โปรแกรมสำเร็จรูปสำหรับวิเคราะห์หุ้นเชิงเทคนิคด้วยแมคดี โดยใช้ภาษาไพทอน

SOFTWARE FOR TECHNICAL STOCK ANALYSIS WITH MACDs INDICATOR BY USING PYTHON.

กฤตนันท์ บุญเดช
ณัฐวุฒิ บุญวิบูลวัฒน์
ถิรวัฒน์ นานอก

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต สาขาคณิตศาสตร์ประยุกต์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

โปรแกรมสำเร็จรูปสำหรับวิเคราะห์หุ้นเชิงเทคนิคด้วยแมคดี โดยใช้ภาษาไพทอน

SOFTWARE FOR TECHNICAL STOCK ANALYSIS WITH MACDs INDICATOR BY USING PYTHON.

กฤตนันท์ บุญเดช
ณัฐวุฒิ บุญวิบูลวัฒน์
ถิรวัฒน์ นานอก

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต สาขาคณิตศาสตร์ประยุกต์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

SOFTWARE FOR TECHNICAL STOCK ANALYSIS WITH MACDs INDICATOR BY USING PYTHON.

KRITTANAN BOONDECH
NATTAWUT BOONWIBULWAT
THIRAWAT NANORK

A SPECIAL PROBLEM SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIRMENTS FOR THE DEGREE OF BACHELOR OF
SCIENCE IN APPLIED MATHEMATICS
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2015

หัวข้อปัญหาพิเศษ	โปรแกรมสำเร็จรูปสำหรับวิเคราะห์หุ้นเชิงเทคนิคด้วยแมคดี โดยใช้ภาษาไพทอน Software for Technical Stock Analysis with MACDs Indicator by Using PYTHON
ชื่อนักศึกษา	นายกฤตนันท์ บุญเดช 55050010 นายณัฐวุฒิ บุญวิบูลวัฒน์ 55050054 นายถิรวัฒน์ นานอก 55050060
ปริญญา	วิทยาศาสตรบัณฑิต (คณิตศาสตร์ประยุกต์)
ภาควิชา	คณิตศาสตร์
ปีการศึกษา	2558
อาจารย์ที่ปรึกษา	ผศ. ดร.วิชัย วิทยาเกียรติเลิศ

คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต (คณิตศาสตร์ประยุกต์) ประจำปีการศึกษา 2558

คณะกรรมการสอบ	**ลายมือชื่อ**
ดร.ธวัชชัย คำประภัสสร ประธานกรรมการ	
ดร.สิริพร แฮนน่า วินเทอร์ กรรมการ	
ผศ. ดร.วิชัย วิทยาเกียรติเลิศ กรรมการและอาจารย์ที่ปรึกษา	

หัวข้อปัญหาพิเศษ	โปรแกรมสำเร็จรูปสำหรับวิเคราะห์หุ้นเชิงเทคนิคด้วยแมคดี โดยใช้ภาษาไพทอน
ชื่อนักศึกษา	นายกฤตนันท์ บุญเดช 55050010
	นายณัฐวุฒิ บุญวิบูลวัฒน์ 55050054
	นายถิรวัฒน์ นานอก 55050060
ปริญญา	วิทยาศาสตรบัณฑิต (คณิตศาสตร์ประยุกต์)
ภาควิชา	คณิตศาสตร์
ปีการศึกษา	2558
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.วิชัย วิทยาเกียรติเลิศ

บทคัดย่อ

ปัญหาพิเศษนี้ทำโปรแกรมสำเร็จรูปในการวิเคราะห์การซื้อขายหุ้นด้วยการวิเคราะห์เชิงเทคนิคด้วยดัชนีแมคดี โดยสร้างโปรแกรมที่เขียนด้วยภาษาไพทอน ซึ่งในโปรแกรมจะประกอบด้วยการคำนวณแบบ MACD, MACDR1, MACDR2, MACDP, MACDP1, MACDP2, MACDK, MACDK1, MACDK2, MACDF, MACDF1, MACDF2 และสามารถรับข้อมูลเรียลไทม์มาคำนวณได้

คำสำคัญ: ดัชนีแมคดี, โปรแกรมสำเร็จรูป, ภาษาไพทอน

Special Problem Title	Software for Technical Stock Analysis with MACDs Indicator by Using PYTHON		
Students	Mr.Krittanan	Boondech	55050010
	Mr.Nattawut	Boonwibulwat	55050054
	Mr.Thirawat	Nanork	55050060
Degree	Bachelor of Science (Applied Mathematics)		
Major Program	Mathematics		
Academic Year	2015		
Advisor	Assist.Prof.Dr. Wichai Witayakiattilerd		

Abstract

This special problem, Software of stock analysis by using Technical Analysis with MACDs indicator. This Program created by Python, The program include composed calculate of MACD, MACDR1, MACDR2, MACDP, MACDP1, MACDP2, MACDK, MACDK1, MACDK2, MACDF, MACDF1, MACDF2 and it can received real time data for calculate.

Keywords: MACDs indicator, Python, Software

กิตติกรรมประกาศ

ปัญหาพิเศษเรื่องโปรแกรมสำเร็จรูปสำหรับวิเคราะห์หุ้นเชิงเทคนิคด้วยแมคดีโดยใช้ภาษาไพทอน คณะผู้จัดทำขอขอบคุณ ผศ. ดร.วิชัย วิทยาเกียรติเลิศ อาจารย์ที่ปรึกษาโครงงานปัญหาพิเศษเล่มนี้ ที่ได้กรุณาช่วยแนะนำ เสนอแนะและให้คำปรึกษาต่างๆ รวมทั้งให้ข้อคิดเห็นเกี่ยวกับแนวทางการดำเนินงาน เอื้อเฟื้อเอกสารความรู้เพื่อมาใช้ประกอบในการดำเนินงาน และอุปกรณ์ที่เกี่ยวข้องในการดำเนินงาน ช่วยแก้ไขข้อบกพร่องหรือปัญหาที่ต่างๆที่เกิดขึ้นระหว่างการดำเนินงานแก่คณะผู้จัดทำได้เป็นอย่างดี

นอกจากนี้ คณะผู้จัดทำขอขอบคุณคณาจารย์สาขาวิชาคณิตศาสตร์ทุกท่าน ที่ได้ให้ความรู้และประสบการณ์ต่างๆตลอดระยะเวลา 4 ปีที่ผ่านมา คอยอบรมสั่งสอนให้เป็นคนดีมีความรับผิดชอบ คอยให้คำแนะนำต่างๆรวมถึงการใช้ชีวิตในมหาวิทยาลัย และขอบคุณเพื่อนๆทุกคนที่ให้กำลังใจคอยช่วยเหลือ ทำให้ปัญหาพิเศษเล่มนี้สัมฤทธิ์ผล

นายกฤตนันท์ บุญเดช
นายณัฐวุฒิ บุญวิบูลวัฒน์
นายถิรวัฒน์ นานอก

สารบัญ

หน้า

สารบัญตาราง

สารบัญตาราง(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในปัจจุบันการลงทุนในตลาดหลักทรัพย์แห่งประเทศไทยเป็นอีกหนึ่งทางเลือกที่เป็นการลงทุนที่ได้รับผลตอบแทนที่สูงแต่ก็มีความเสี่ยงที่สูงมากอีกเช่นกัน และเพื่อที่จะลดความเสี่ยงที่จะเกิดขึ้นจึงได้มีการสร้างเครื่องมือที่ช่วยในการวิเคราะห์ที่ใช้ชี้วัดแนวโน้มของราคาหุ้นและกำหนดสัญญาณซื้อ สัญญาณขาย เพื่อให้ง่ายต่อการตัดสินใจในการลงทุนที่มีความเสี่ยงนี้

ในปี ค.ศ.1970 Gerald Appel ได้คิดค้น MACD ซึ่งเป็นเครื่องมือที่ใช้ชี้วัดแนวโน้มของราคาหุ้นและกำหนดสัญญาณซื้อสัญญาณขาย ที่ใช้การวิเคราะห์ทางเทคนิคที่สร้างขึ้นโดยอาศัยความแตกต่างของเส้นค่าเฉลี่ยระยะสั้นและเส้นค่าเฉลี่ยระยะยาว ที่สามารถให้อัตราความสำเร็จที่ 36.44%

ในปี ค.ศ.2000 Gunter Meissner ได้เสนอ MACD ปรับปรุงเรียกว่า MACDR1 สามารถให้อัตราความสำเร็จ 56.04% และMACDR2 ที่สามารถให้อัตราความสำเร็จ 74.57%

ในปี ค.ศ. 2013 นายพิพรรธน์ สำเภากิจ และคณะที่จัดทำงานวิจัยปัญหาพิเศษทางด้านการปรับปรุง MACD โดยการดัดแปลงเส้นสัญญาณโดยการอนุมานค่าน้ำหนักการซื้อ-ขายด้วยตรรกศาสตร์ฟัซซีโดยให้ชื่อว่า MACDP โดยทำการวิเคราะห์กับหุ้นแต่ละตัวในกลุ่ม SET-100 ในตลาดหลักทรัพย์แห่งประเทศไทยตั้งแต่วันที่ 5 กรกฎาคม ถึง 28 พฤศจิกายน พ.ศ.2556 ที่สามารถให้อัตราความสำเร็จ 61.02%

ในปี ค.ศ. 2014 นายอานัส ตือบิงหมัะ และคณะได้จัดทำงานวิจัยปัญหาพิเศษทางด้านการปรับปรุง MACD โดยใช้แนวคิดของการยืดขยายเส้นสัญญาณด้วยฟังก์ชั่นยืดขยาย K โดยนิยามใหม่ว่า “ MACDK” โดยทำการวิเคราะห์กับข้อมูลราคาของหุ้นย้อนหลัง 5 ปี ของหุ้นแต่ละตัวในกลุ่ม SET-50 และ SET-100 ในตลาดหลักทรัพย์แห่งประเทศไทยระหว่างวันที่ 6 พฤศจิกายน พ.ศ. 2552 ถึง 7 พฤศจิกายน พ.ศ. 2557 ที่สามารถให้อัตราความสำเร็จ 47.72%

ในปี ค.ศ. 2014 นายอนิรุธ หนูตอและคณะได้จัดทำงานวิจัยปัญหาพิเศษทางด้านการปรับปรุง MACD โดยใช้แนวคิดของตรรกศาสตร์ฟัซซีเข้ามาช่วยในการปรับปรุงเพื่อเพิ่มประสิทธิภาพในการตัดสินใจโดยนิยามใหม่ว่า “MACDF” โดยทำการวิเคราะห์กับข้อมูลราคาของหุ้นแต่ละตัวในกลุ่ม SET-50 ในตลาดหลักทรัพย์แห่งประเทศไทยระหว่างวันที่ 4 มกราคม พ.ศ. 2555 ถึง 30 ธันวาคม พ.ศ. 2557 ที่สามารถให้อัตราความสำเร็จ 61.02%

เราจึงเห็นความสำคัญของวิธีการวิเคราะห์แต่ละวิธี จึงได้ทำการออกแบบและเขียนโปรแกรมขึ้นมาเพื่อรวบรวมวิธีการวิเคราะห์แต่ละวิธีเพื่อช่วยให้ง่ายต่อการตัดสินใจและหลากหลายและสามารถเข้าถึงได้ง่ายจากการหาเส้นสัญญาณซื้อ-ขายจากวิธีการวิเคราะห์แต่ละวิธี

1.2 วัตถุประสงค์ของปัญหาพิเศษ

เพื่อสร้างโปรแกรมสำเร็จรูปสำหรับวิเคราะห์หุ้นเชิงเทคนิคด้วยแมคดีโดยใช้ภาษาไพทอน โดยรวบรวมวิธีการวิเคราะห์ที่ได้ศึกษามาก่อนหน้านี้ ได้แก่ MACD, MACDR1, MACDR2, MACDP, MACDK, MACDF ที่ช่วยในการหาสัญญาณซื้อ-ขายในตลาดหลักทรัพย์แห่งประเทศไทย

1.3 ขอบเขตของปัญหา

ในการจัดทำโครงการเรื่องโปรแกรมสำเร็จรูปสำหรับวิเคราะห์หุ้นเชิงเทคนิคด้วยแมคดีโดยใช้ภาษาไพทอนดังนี้

ออกแบบและเขียนโปรแกรมโดยใช้ spyder เพื่อทำให้โปรแกรมสามารถแสดงเส้นสัญญาณซื้อ-ขาย ออกมาในรูปแบบของกราฟและข้อมูล โดยอาศัยข้อมูลในตลาดหลักทรัพย์แห่งประเทศไทย

1.4 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่คาดว่าจะได้รับจากการทำโครงการเรื่องโปรแกรมสำเร็จรูปสำหรับวิเคราะห์หุ้นเชิงเทคนิคด้วยแมคดีโดยใช้ภาษาไพทอนมีดังนี้

1.4.1) ได้รับความรู้เกี่ยวกับการออกแบบและเขียนโปรแกรมโดยใช้ spyder

1.4.2) ได้รับความรู้เกี่ยวกับการวิเคราะห์แนวโน้มราคาหุ้นโดยอาศัยเส้นสัญญาณซื้อ-ขาย ในแต่ละวิธี

1.5 ขั้นตอนในการดำเนินงาน

1.5.1) ศึกษาบทความปัญหาพิเศษที่เกี่ยวกับ MACD, MACDR1, MACDR2, MACDP, MACDF และ MACDK

1.5.2) ศึกษาเกี่ยวกับการออกแบบและเขียนโปรแกรมเพื่อนำมาใช้ในการทำบทความปัญหาพิเศษนี้

1.5.3) ศึกษาและทำความเข้าใจเกี่ยวกับการทำงานของเครื่องมือ

1.5.4) ออกแบบและลงมือเขียนโปรแกรมเพื่อรวบรวมวิธีการวิเคราะห์ทั้งหมด

1.5.5) ทดลองใช้จริง

1.5.6) วิเคราะห์ผลการทดลอง

1.5.7) สรุปผลการทดลองและเรียบเรียงโครงการปัญหาพิเศษ

1.6 ระยะเวลาการดำเนินงาน

ปัญหาพิเศษใช้เวลาดำเนินการทั้งสิ้น 8 เดือนตั้งแต่เดือนกันยายน พ.ศ.2558 ถึงเดือนเมษายน พ.ศ.2559 ระยะเวลาที่ใช้ในขั้นตอนต่างๆ แสดงดังตารางที่ 1.1

ตารางที่ 1.1 ตารางแสดงระยะเวลาการดำเนินงานในขั้นตอนต่างๆ

การดำเนินงาน	ระยะเวลา							
	2558				2559			
	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1) ศึกษาบทความปัญหาพิเศษที่เกี่ยวกับ MACD, MACDR1, MACDR2, MACDP, MACDF และ MACDK	←	—	→					
2)ศึกษาเกี่ยวกับการออกแบบและเขียนโปรแกรมเพื่อนำมาใช้ในการทำบทความปัญหาพิเศษนี้		←	—	→				
3)ศึกษาและทำความเข้าใจเกี่ยวกับการทำงานของเครื่องมือ		←	—	—	→			
4)ออกแบบและลงมือเขียนโปรแกรมเพื่อรวบรวมวิธีการวิเคราะห์ทั้งหมด			←	—	—	→		
5)ทดลองใช้จริง				←	—	—	→	
6)วิเคราะห์ผลการทดลอง					←	—	→	
7)สรุปผลการทดลองและเรียบเรียงโครงการปัญหาพิเศษ						←	—	→

บทที่ 2
ภาษาไพทอน

ในบทนี้จะกล่าวถึงความรู้พื้นฐานของภาษาไพธอน ซึ่งผู้วิจัยเลือกใช้ในงานวิจัยเนื่องจากเป็นภาษาโปรแกรมชั้นสูงในลักษณะภาษาอินเตอร์พรีเตอร์โปรแกรมมิ่ง (Interpreted programming language) ที่เข้าใจง่าย และมีประสิทธิภาพสูงที่เป็นที่นิยมกันในปัจจุบัน

2.1 ความหมาย

ไพธอน (Python) เป็นภาษาโปรแกรมในลักษณะภาษาอินเตอร์พรีเตอร์โปรแกรมมิ่ง (Interpreted programming language) ผู้คิดค้นคือ Guido van Rossum ในปี 1990 ซึ่งไพธอนเป็น การจัดการชนิดของตัวแปรแบบแปรผันตามข้อมูลที่บรรจุอยู่ (Fully dynamically typed) และใช้การจัดการหน่วยความจำเป็นอัตโนมัติ (Automatic memory management) โดยได้เป็นการพัฒนาและผสมผสานของภาษาอื่น ๆ ได้แก่ ABC, Modula-3, Icon, ANSI C, Perl, Lisp, Smalltalk และ Tcl และภาษาไพธอนยังเป็นแนวคิดที่ทำให้เกิดภาษาใหม่ ๆ ซึ่งได้แก่ Ruby และ Boo เป็นต้น

ไพธอนนั้นพัฒนาเป็นโครงการ Open source โดยมีการจัดการแบบไม่หวังผลกำไรโดย Python Software Foundation และสามารถหาข้อมูลและตัวแปรภาษาได้จากเว็บไซต์ของไพธอนเอง ที่ http://www.python.org/

ความสามารถของภาษา Python

1. เรียนรู้ได้ง่าย โดยภาษา Python มีโครงสร้างของภาษาไม่ซับซ้อนเข้าใจง่าย ซึ่งโครงสร้างภาษา Python จะคล้ายกับภาษา C มาก เพราะภาษา Python สร้างขึ้นมาโดยใช้ภาษา C ทำให้ผู้ที่คุ้นเคยภาษา C อยู่แล้วใช้งานภาษา Python ได้ไม่ยาก นอกจากนี้โดยตัวภาษาเองมีความยืดหยุ่นสูงทำให้การจัดการกับงานด้านข้อความ และ Text File ได้เป็นอย่างดี
2. ไม่ต้องเสียค่าใช้จ่าย เพราะตัวแปรภาษา Python อยู่ภายใต้ลิขสิทธิ์ GNU
3. ใช้ได้หลายแพลตฟอร์ม ในช่วงแรกภาษา Python ถูกออกแบบใช้งานกับระบบ Unix อยู่ก็จริง แต่ในปัจจุบันได้มีการพัฒนาตัวแปลภาษา Python ให้สามารถใช้กับระบบปฏิบัติการอื่นๆ เช่น Linux, Windows 95/98/ME, Windows NT, Windows 2000, OS/2
4. ภาษา Python สร้างขึ้นโดยนำข้อดีของภาษาต่างๆมาใช้ เช่น ภาษา C, C++, JAVA, Perl
5. ภาษา Python เป็นภาษาประเภท Server side Script คือการทำงานของภาษา Python จะทำงานด้านฝั่ง Server แล้วส่งผลลัพธ์กลับมายัง Client ทำให้มีความปลอดภัยสูง

สำหรับโปรแกรมสำเร็จรูปนี้ได้ใช้ Python เป็นโปรแกรมหลักในการเขียนโปรแกรม โดยใช้ GUI ของ Python ในการออกแบบหน้าต่างการทำงานของโปรแกรมและนำโค้ดภาษา Python จากปัญหาพิเศษของรุ่นก่อนหน้ามาดัดแปลงแก้ไขเพื่อให้ใช้งานได้ดีขึ้นสำหรับโปรแกรมนี้ โดยทำให้สามารถรับค่าเรียลไทม์ของราคาหุ้นตัวต่างๆสามารถระบุค่าเฉลี่ย วันสั้น วันยาวและสามารถแสดงออกมาเป็นกราฟการทำนายได้

2.2 GUI Python

2.2.1 การสร้าง Frame

Syntax ที่สำคัญในการสร้าง Frame นี้คือ

```
W=Frame(master,option,...)
```

โดย master จะเป็นส่วนของหน้าต่างหลักที่จะใช้แสดง Frame ที่ได้ทำการสร้างขึ้นมา และ option เป็นรายการตัวเลือกที่ช่วยในการสร้างรายละเอียดและการทำงานของ Frame

Option	Description
Bg	เป็นตัวกำหนดสีพื้นหลังของ Frame
Bd	เป็นตัวกำหนดขอบ Frame เพื่อเพิ่มรายละเอียดมิติของ Frame
Cursor	เป็นการตั้งค่า cursor เมื่อ cursor เข้ามาใน Frame cursor จะเปลี่ยนรูป (ลูกศร, จุด, ฯลฯ) ไปตามที่กำหนด
Height	เป็นการกำหนดขนาดแนวตั้งของ Frame ที่ถูกสร้างขึ้นมา
highlightbackground	เป็นการกำหนดสีของพื้นหลังเมื่อไม่ได้มีการใช้งาน
Highlightcolor	เป็นการกำหนดสีของพื้นหลังเมื่อมีการใช้งาน
Highlightthickness	เป็นการกำหนดความหนาของ highlight เมื่อมีการโฟกัสของอักษรบน Frame
Width	เป็นการกำหนดขนาดแนวนอนของ Frame ที่ถูกสร้างขึ้นมา

ตารางที่ 2.1 ตารางแสดงการสร้างเฟรม

ตัวอย่างที่2.1 ตัวอย่างการสร้างเฟรม

```
Example=Frame(root,relief=GROOVE,
borderwidth=4,width=200,height=620,bg='white')

Example.place(relx=0.01,rely=0.02,anchor=NW,width=200
,height=620)
```

2.1.2 การสร้าง Button

Syntax ที่สำคัญในการสร้าง Button นี้คือ

```
W=Button(master,option=value,...)
```

โดย master จะเป็นส่วนของหน้าต่างหลักที่จะใช้แสดง Button ที่ได้ทำการสร้างขึ้นมา และ option เป็นรายการตัวเลือกที่ช่วยในการสร้างรายละเอียดและการทำงานของ Button

Option	Description
Bg	เป็นตัวกำหนดสีพื้นหลังของ Button
Bd	เป็นตัวกำหนดขอบ Button เพื่อเพิ่มรายละเอียดมิติของ Button
activebackground	เป็นการกำหนดสีพื้นหลังหลักของ Button โดยจะทำการผสมสีกันระหว่าง activebackground และ activeforeground
activeforeground	เป็นการกำหนดสีพื้นหลังรองของ Button โดยจะทำการผสมสีกันระหว่าง activebackground และ activeforeground
Command	เป็นการเรียกใช้ฟังก์ชั่นเมื่อมีการคลิกปุ่มที่ถูกสร้างขึ้นมา
Fg	เป็นการกำหนดสีของข้อความที่จะแสดงบน Button
Font	เป็นการกำหนดรูปแบบอักษรของข้อความที่จะแสดงบน Button
height	เป็นการกำหนดขนาดแนวตั้งของ Button ที่ถูกสร้างขึ้นมา
Image	เป็นการกำหนดรูปภาพที่จะใช้แสดงบน Button แทนข้อความ
Justify	เป็นการกำหนดให้แสดงข้อความบน Button แบบหลายบรรทัด โดยเริ่มจากทางซ้าย(LEFT) เริ่มจากทางขวา(RIGHT) หรือเริ่มจากตรงกลาง(CENTER)
Padx	เป็นการกำหนดระยะความห่างทางซ้ายและขวาของข้อความที่แสดงจากขอบ Button
Pady	เป็นการกำหนดระยะความห่างด้านบนและล่างของข้อความที่แสดงจากขอบ Button

Width	เป็นการกำหนดขนาดแนวนอนของ Button ที่ถูกสร้างขึ้นมา
Underline	เป็นการกำหนดจำนวนเส้นที่ถูกขีดไว้ใต้ข้อความที่แสดงบน Button
State	เป็นการกำหนดการตอบสนองของ Button ถ้าจะให้สามารถใช้งานได้ปกติใช้ NORMAL แต่ถ้าไม่ต้องการให้ Button นี้สามารถคลิกได้ใช้ DISABLED

ตารางที่ 2.2 ตารางแสดงการสร้าง Button

ตัวอย่างที่2.2 ตัวอย่างการสร้าง Button

```
Example=Button(root,text="Hello",background='white',
command=hello world,fg="black",font=('forte', 11))

Example.place(relx=0.845,rely=0.95,anchor=SW,width=10
0,height=30)
```

2.1.3 การสร้าง Entry Box

Syntax ที่สำคัญในการสร้าง Entry Box นี้คือ

```
W=Entry(master,option,...)
```

โดย master จะเป็นส่วนของหน้าต่างหลักที่จะใช้แสดง Entry Box ที่ได้ทำการสร้างขึ้นมา และ option เป็นรายการตัวเลือกที่ช่วยในการสร้างรายละเอียดและการทำงานของ Entry Box

Option	Description
Bg	เป็นตัวกำหนดสีพื้นหลังของ Entry
Bd	เป็นตัวกำหนดขอบ Entry เพื่อเพิ่มรายละเอียดมิติของ Entry
Command	เป็นการเรียกใช้ฟังก์ชั่นเมื่อมีการใส่ค่าลงไป
Justify	เป็นการกำหนดให้แสดงข้อความบน Entry แบบหลายบรรทัด โดยเริ่มจากทางซ้าย(LEFT) เริ่มจากทางขวา(RIGHT) หรือเริ่มจากตรงกลาง(CENTER)
Font	เป็นการกำหนดรูปแบบอักษรของข้อความที่จะแสดงบน Entry

Cursor	เป็นการตั้งค่า cursor เมื่อ cursor เข้ามาใน Entry cursor จะเปลี่ยนรูป (ลูกศร, จุด, ฯลฯ) ไปตามที่กำหนด
Fg	เป็นการกำหนดสีของข้อความที่จะแสดงบน Entry
Show	เป็นการกำหนดการแสดงของข้อความที่ใส่ลงไปหรือแสดงบน Entry โดยปกติจะแสดงในรูปแบบของอักษรที่ใส่เข้าไป เพื่อสามารถทำให้เป็นการใส่แบบ password ได้จึงต้องใช้ option นี้โดยตั้งค่าดังนี้ show = "*"
State	โดยปกติ state = NORMAL คือมีการตอบสนองของ Entry เพื่อให้สามารถใส่ค่าลงไปได้ แต่หาก state = DISABLED พื้นหลังจะเป็นสีเทาและจะไม่มีการตอบสนองของ Entry คือไม่สามารถใส่ค่าลงไปได้
Width	เป็นการกำหนดขนาดแนวนอนของ Entry ที่ถูกสร้างขึ้นมา
xscrollcommand	ถ้าหากมีคาดการณ์ไว้ว่าข้อมูลที่จะถูกใส่ลงหรือแสดงภายใน Entry จะมีความยาวเกินกว่าขนาดที่ตั้งไว้ option จะทำให้ Entry ปรากฏ scrollbar เพื่อเชื่องโยงกับขนาดความยาวของข้อมูลที่ถูกใส่ลงไป

ตารางที่ 2.3 ตารางแสดงการสร้าง Entry Box

ตัวอย่างที่2.3 ตัวอย่างการสร้าง Entry Box

```
Example=Entry(root,bd =5,justify=CENTER)

Example.place(relx=0.2,rely=0.72,anchor=NW,width=80)
```

2.1.4 การสร้าง Check Button

Syntax ที่สำคัญในการสร้าง Check Button นี้คือ

```
W=Checkbutton(master,option,...)
```

โดย master จะเป็นส่วนของหน้าต่างหลักที่จะใช้แสดง Check Button ที่ได้ทำการสร้างขึ้นมาและ option เป็นรายการตัวเลือกที่ช่วยในการสร้างรายละเอียดและการทำงานของ Check Button

Option	Description
Bg	เป็นตัวกำหนดสีพื้นหลังของ Check Button
Bd	เป็นตัวกำหนดขอบ Check Button เพื่อเพิ่ม

	รายละเอียดมิติของ Check Button
activebackground	เป็นการกำหนดสีพื้นหลังหลักของ Check Button โดยจะทำการผสมสีกันระหว่าง activebackground และ activeforeground
activeforeground	เป็นการกำหนดสีพื้นหลังรองของ Check Button โดยจะทำการผสมสีกันระหว่าง activebackground และ activeforeground
Command	เป็นการเรียกใช้ฟังก์ชั่นเมื่อมีการคลิกเลือก Check Button ที่ถูกสร้างขึ้นมา
Fg	เป็นการกำหนดสีของข้อความที่จะแสดงบน Check Button
Font	เป็นการกำหนดรูปแบบอักษรของข้อความที่จะแสดงบน Check Button
Height	เป็นการกำหนดขนาดแนวตั้งของ Check Button ที่ถูกสร้างขึ้นมา
Image	เป็นการกำหนดรูปภาพที่จะใช้แสดงบน Check Button แทนข้อความ
Justify	เป็นการกำหนดให้แสดงข้อความบน Check Button แบบหลายบรรทัด โดยเริ่มจากทางซ้าย(LEFT) เริ่มจากทางขวา(RIGHT) หรือเริ่มจากตรงกลาง(CENTER)
Padx	เป็นการกำหนดระยะความห่างทางซ้ายและขวาของข้อความที่แสดงจากขอบ Check Button
Pady	เป็นการกำหนดระยะความห่างด้านบนและล่างของข้อความที่แสดงจากขอบ Check Button
Width	เป็นการกำหนดขนาดแนวนอนของ Check Button ที่ถูกสร้างขึ้นมา
Underline	เป็นการกำหนดจำนวนเส้นที่ถูกขีดไว้ใต้ข้อความที่แสดงบน Check Button
State	เป็นการกำหนดการตอบสนองของ Check Button ถ้าจะให้สามารถใช้งานได้ปกติใช้ NORMAL แต่ถ้าไม่ต้องการให้ Check Button นี้สามารถคลิกได้ใช้ DISABLED
Cursor	เป็นการตั้งค่า cursor เมื่อ cursor เข้ามาใน Check Button cursor จะเปลี่ยนรูป (ลูกศร, จุด, ฯลฯ) ไปตามที่กำหนด

disabledforeground	เป็นการกำหนดสีพื้นหลังของ Check Button ที่มีสถานะของ state เป็น disabled จากที่มีเพียงแค่สีเทา
Offvalue	โดยปกติ Check Button จะมีตัวแปรที่คอยควบคุมเริ่มต้นคือ 0 แต่เมื่อถูกเลือกหรือถูกคลิกตัวแปรที่คอยควบคุมจะมีค่าเท่ากับ 1 ดังนั้น option นี้จะทำการกำหนดค่าให้ Check Button นี้มีตัวแปรเป็น 0 ตลอดไม่ว่าจะถูกเลือกหรือไม่ก็ตาม
onvalue	โดยปกติ Check Button จะมีตัวแปรที่คอยควบคุมเริ่มต้นคือ 0 แต่เมื่อถูกเลือกหรือถูกคลิกตัวแปรที่คอยควบคุมจะมีค่าเท่ากับ 1 ดังนั้น option นี้จะทำการกำหนดค่าให้ Check Button นี้มีตัวแปรเป็น 1 ตลอดไม่ว่าจะถูกเลือกหรือไม่ก็ตาม
Selectcolor	เป็นการกำหนดสีในช่องของ Check Button
Selectimage	เป็นการกำหนดรูปภาพในช่องของ Check Button

ตารางที่ 2.4 ตารางแสดงการสร้าง Check Button

ตัวอย่างที่2.4 ตัวอย่างการสร้าง Check Button

```
Checkbutton(CK,text='MACD',variable=var1,bg='#f3fbfa'
).grid(row=0,column=0,sticky=W)

Checkbutton(CK,text='MACDR1',variable=var2,bg='#f3fbf
a').grid(row=0,column=1,sticky=W)

Checkbutton(CK,text='MACDR2',variable=var3,bg='#f3fbf
a').grid(row=0,column=2, sticky=W)

Checkbutton(CK,text='MACDP',variable=var4,bg='#f3fbfa
').grid(row=1,column=0,sticky=W)

Checkbutton(CK,text='MACDP1',variable=var5,bg='#f3fbf
a').grid(row=1,column=1,sticky=W)

Checkbutton(CK,text='MACDP2',variable=var6,bg='#f3fbf
a').grid(row=1,column=2,sticky=W)
```

```
Checkbutton(CK,text='MACDK',variable=var7,bg='#f3fbfa
').grid(row=2,column=0,sticky=W)

Checkbutton(CK,text='MACDK1',variable=var8,bg='#f3fbf
a').grid(row=2,column=1,sticky=W)

Checkbutton(CK,text='MACDK2',variable=var9,bg='#f3fbf
a').grid(row=2,column=2,sticky=W)

Checkbutton(CK,text='MACDF',variable=var10,bg='#f
3fbfa').grid(row=3,column=0,sticky=W)

Checkbutton(CK,text='MACDF1',variable=var11,bg='#
f3fbfa').grid(row=3,column=1,sticky=W)

Checkbutton(CK,text='MACDF2',variable=var12,bg='#
f3fbfa').grid(row=3,column=2,sticky=W)
```

2.1.5 การสร้าง Label

Syntax ที่สำคัญในการสร้าง Label นี้คือ

```
W = Label (master, option,...)
```

โดย master จะเป็นส่วนของหน้าต่างหลักที่จะใช้แสดง Label ที่ได้ทำการสร้างขึ้นมา และ option เป็นรายการตัวเลือกที่ช่วยในการสร้างรายละเอียดและการทำงานของ Label

Option	Desciption
Bg	เป็นตัวกำหนดสีพื้นหลังของ Label
Bd	เป็นตัวกำหนดขอบ Label เพื่อเพิ่มรายละเอียดมิติของ Label
activebackground	เป็นการกำหนดสีพื้นหลังหลักของ Label โดยจะทำการผสมสีกันระหว่าง activebackground และ activeforeground
activeforeground	เป็นการกำหนดสีพื้นหลังรองของ Label โดยจะทำการผสมสีกันระหว่าง activebackground และ activeforeground
Fg	เป็นการกำหนดสีของข้อความที่จะแสดงบน Label

Font	เป็นการกำหนดรูปแบบอักษรของข้อความที่จะแสดงบน Label
Height	เป็นการกำหนดขนาดแนวตั้งของ Label ที่ถูกสร้างขึ้นมา
Justify	เป็นการกำหนดให้แสดงข้อความบน Button แบบหลายบรรทัด โดยเริ่มจากทางซ้าย(LEFT) เริ่มจากทางขวา(RIGHT) หรือเริ่มจากตรงกลาง(CENTER)
Padx	เป็นการกำหนดระยะความห่างทางซ้ายและขวาของข้อความที่แสดงจากขอบ Label
Pady	เป็นการกำหนดระยะความห่างด้านบนและล่างของข้อความที่แสดงจากขอบ Label
Width	เป็นการกำหนดขนาดแนวนอนของ Label ที่ถูกสร้างขึ้นมา
Underline	เป็นการกำหนดจำนวนเส้นที่ถูกขีดไว้ใต้ข้อความที่แสดงบน Label
Cursor	เป็นการตั้งค่า cursor เมื่อ cursor เข้ามาใน Label cursor จะเปลี่ยนรูป (ลูกศร, จุด, ฯลฯ) ไปตามที่กำหนด
Image	เป็นการกำหนดรูปภาพที่จะใช้แสดงบน Label แทนข้อความ
Anchor	เป็นการกำหนดตำแหน่งที่จะแสดงข้อความว่าจะอยู่ตำแหน่งไหนของพื้นที่ แม้ว่าพื้นที่ที่จะแสดงข้อความจะมีการขยายขนาดเพิ่มมากขึ้นก็ตาม

ตารางที่ 2.5 ตารางแสดงการสร้าง label

ตัวอย่างที่2.5 ตัวอย่างการสร้าง label

```
Example=Label(root,text="Indicator",fg="black",bg='wh
ite',Font=('Monotype Corsiva',11))

Example.place(relx=0.01,rely=0.06,anchor=NW)
```

2.1.6 การสร้าง Option Menu

Syntax ที่สำคัญในการสร้าง Option Menu นี้คือ

```
W=Optionmenu(master,option,...)
```

โดย master จะเป็นส่วนของหน้าต่างหลักที่จะใช้แสดง Option Menu ที่ได้ทำการสร้างขึ้นมา และ option เป็นรายการตัวเลือกที่ช่วยในการสร้างรายละเอียดและการทำงานของ Option Menu

Option	Description
Bg	เป็นตัวกำหนดสีพื้นหลังของ Option Menu
Bd	เป็นตัวกำหนดขอบ Option Menu เพื่อเพิ่มรายละเอียดมิติของ Option Menu
activebackground	เป็นการกำหนดสีพื้นหลังหลักของ Option Menu โดยจะทำการผสมสีกันระหว่าง activebackground และ activeforeground
activeforeground	เป็นการกำหนดสีพื้นหลังรองของ Option Menu โดยจะทำการผสมสีกันระหว่าง activebackground และ activeforeground
Command	เป็นการเรียกใช้ฟังก์ชั่นเมื่อมีการคลิกปุ่มที่ถูกสร้างขึ้นมา
Fg	เป็นการกำหนดสีของข้อความที่จะแสดงบน Option Menu
Font	เป็นการกำหนดรูปแบบอักษรของข้อความที่จะแสดงบน Option Menu
Height	เป็นการกำหนดขนาดแนวตั้งของ Option Menu ที่ถูกสร้างขึ้นมา
Image	เป็นการกำหนดรูปภาพที่จะใช้แสดงบน Option Menu แทนข้อความ
Justify	เป็นการกำหนดให้แสดงข้อความบน Option Menu แบบหลายบรรทัด โดยเริ่มจากทางซ้าย(LEFT) เริ่มจากทางขวา(RIGHT) หรือเริ่มจากตรงกลาง(CENTER)
Padx	เป็นการกำหนดระยะความห่างทางซ้ายและขวาของข้อความที่แสดงจากขอบ Option Menu
Pady	เป็นการกำหนดระยะความห่างด้านบนและล่างของข้อความที่แสดงจากขอบ Option Menu
Width	เป็นการกำหนดขนาดแนวนอนของ Option Menu ที่ถูกสร้างขึ้นมา

underline	เป็นการกำหนดจำนวนเส้นที่ถูกขีดไว้ใต้ข้อความที่แสดงบน Option Menu
State	เป็นการกำหนดการตอบสนองของ Option Menu ถ้าจะให้สามารถใช้งานได้ปกติใช้ NORMAL แต่ถ้าไม่ต้องการให้ Option Menu นี้สามารถคลิกได้ใช้ DISABLED

ตารางที่ 2.6 ตารางแสดงการสร้าง Option Menu

ตัวอย่างที่2.6 ตัวอย่างการสร้าง Option Menu

```
group=['SET50','SET100','SETHD','AGRO','--AGRI','--
FOOD','CONSUMP','--FASHION','--HOME','--
PERSON','FINCIAL','--BANK','--FIN','--
INSUR','INDUS','--AUTO','--IMM','--PAPER','--
PETRO','--PKG','--STEEL','PROPCON','--CONMAT','--
PROP','--PF&REIT','--CONS','RESOURC','--ENERG','--
MINE','SERVICE','--COMM','--HELTH','--MEDIA','--
PROF','--TOURISM','--TRANS','TECH','--ETRON','--ICT']

varindicator2=StringVar()

Example=OptionMenu(root,varindicator2,*group,command=
groupstock)
```

2.1.7 การรับข้อมูล Real time.

รับข้อมูล Real time จากที่อื่นโดยกำหนดแหล่งข้อมูลโดยใช้คำสั่ง **Url=...** และเปิดจากคำสั่ง

```
htmlfile=urllib.urlopen(url)
```

และอ่านข้อมูลจากคำสั่ง

```
htmltext=htmlfile.read()
```

และเขียนข้อมูลใหม่จากคำสั่ง

```
regex='<td>(.+?)</td>'
pattern=re.compile(regex)
new_date=re.findall(pattern,htmltext)
```

2.1.8 การ Open File PDF

Syntax ที่สำคัญในการใช้งาน Link Web นี้คือ

```
os.system("File.pdf")
```

ตัวอย่างที่2.6 ตัวอย่าง Open File PDF

```
def callpdf1(event):
os.system("MACD.pdf")
```

บทที่ 3
การวิเคราะห์หลักทรัพย์เชิงเทคนิค

การลงทุนในหุ้นผู้ลงทุนย่อมต้องคาดหวังผลตอบแทนในแต่ละครั้ง ในการตัดสินใจที่จะเลือกลงทุนในหุ้นตัวใดย่อมต้องมีความรู้พื้นฐานและจังหวะเวลาที่ดี สิ่งสำคัญสาหรับการลงทุนในตลาดหุ้นนั้นก็คือ จังหวะเวลา คำ ว่าจังหวะนั้น หมายถึง การซื้อหุ้นในราคาที่ต่ำ และสามารถขายได้ในราคาที่สูงกว่า ดังนั้นนักลงทุนจะทำ อย่างไรเพื่อให้ทราบจังหวะที่เหมาะสมในการซื้อขายหุ้น จากผลงานวิจัยที่ผ่านมามีการพยายามคิดหลักการที่ใช้พยากรณ์การซื้อขายหุ้น ซึ่งสามารถแบ่งหลักการดังกล่าวออกเป็น 2 หลักการสำคัญ คือ การวิเคราะห์หุ้น และการใช้เทคนิคการวิเคราะห์ตัวชี้วัดการซื้อขายและราคาหุ้น ซึ่ง 2 หลักการสำคัญนั้นมักจะมาในรูปแบบของแอพพลิเคชั่นหรือโปรแกรม

ในบทนี้จะกล่าวถึงการวิเคราะห์หลักทรัพย์เชิงเทคนิคด้วยวิธีต่างๆซึ่งปรากฏในงานวิจัยที่กล่าวไว้ในบทที่ 1 ซึ่งได้แก่ แม็คดีหรือMACD, MACDK, MACDP และ MACDF

3.1 MACD (Moving Average Convergence Divergence)

เส้นค่าเฉลี่ยเคลื่อนที่ร่วมทางแยกทาง หรือ MACD เป็นดัชนีที่นิยมใช้ในการวิเคราะห์หุ้นอย่างมากในปัจจุบัน เนื่องจากเป็นดัชนีที่นักลงทุนหรือนักวิเคราะห์สามารถใช้งานได้ง่าย ไม่ซับซ้อน และมีประสิทธิภาพที่นักลงทุนยอมรับได้ MACD ถูกใช้เป็นดัชนีเพื่อวิเคราะห์ ดูแนวโน้มการเปลี่ยนแปลงราคา สภาวะตลาด การกำหนดสัญญาณซื้อ-สัญญาณขาย เป็นต้น

3.1.1 ค่าเฉลี่ยเอ็กซ์โปเนนเชียลเคลื่อนที่ (Exponential Moving Average)

นิยามที่3.1 กำหนดให้ (τ) เป็นลำดับของจุดเวลาทั้งหมดที่ทำการเก็บข้อมูล และ p_t เป็นราคาหุ้นของวันที่ $t \in (\tau)$ เรียกลำดับ $(p) = (p_t \mid t \in (\tau))$ ว่า *เส้นราคา* (Price line)

นิยามที่3.2 กำหนดให้ $w_k = \alpha(1-\alpha)^{t-k}$ เมื่อ $k = t, t-1, t-2, \ldots$ โดยที่ $0 < \alpha < 1$ และ กำหนดให้ (τ) เป็นลำดับของจุดเวลาทั้งหมดที่ทำการเก็บข้อมูล ค่าเฉลี่ยเอ็กซ์โปเนนเชียลเคลื่อนที่ของวันที่ $t \in (\tau)$ เขียนแทนสัญลักษณ์ด้วย $EMA(t)$ นิยามโดย

$$EMA(t) = \mathrm{w}_t\, p_t + w_{t-1} p_{t-1} + \ldots = \sum_{k=-\infty}^{t} \alpha(1-\alpha)^{t-k} p_k \qquad (3.1)$$

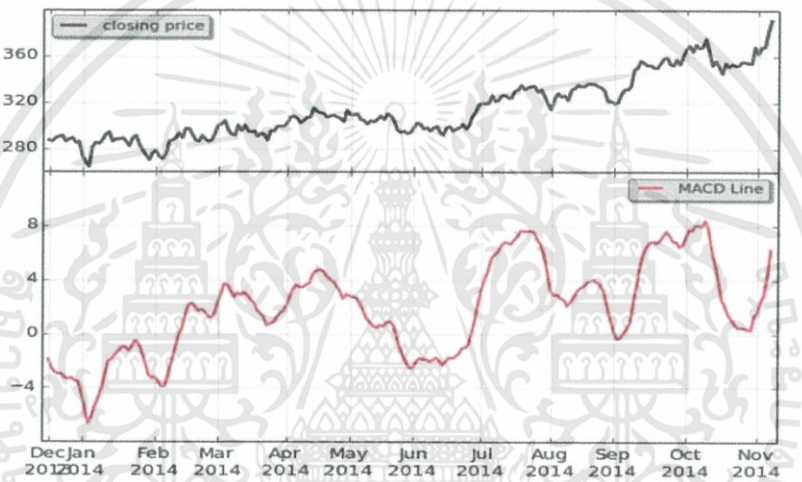
โดยทั่วไป นักวิเคราะห์นิยมกำหนด $\alpha = \frac{2}{n \mid 1}$ สำหรับการคำนวณค่าเฉลี่ยเอ็กซ์โปเนนเชียลเคลื่อนที่ของราคาหุ้น ซึ่งเรียก α ว่า *ค่าปรับเรียบ* เมื่อ $2 < n < t$

3.1.2 เส้นค่าเฉลี่ยเคลื่อนที่ร่วมทางแยกทาง หรือ MACD

นิยามที่3.3 MACD หรือเส้น MACD คือลำดับผลต่างของค่าเฉลี่ยเคลื่อนที่แบบเอ็กซ์โปเนนเชียลสองลำดับที่มีค่าพารามิเตอร์ n_1 และ n_2 ที่ซึ่ง $n_1 < n_2$ เขียนแทนด้วย $m = (m_{t_i} \mid t_i \in (\tau))$ โดยที่

$$m_{t_i} = EMA^{price}_{n_1}(t_i) - EMA^{price}_{n_2}(t_i) \tag{3.2}$$

หมายเหตุ ในที่นี้จะเรียก n_1 ว่า ***วันสั้น*** และเรียก n_2 ว่า *วันยาว* นั่นคือ เส้น MACD คือผลต่างระหว่างเส้นค่าเฉลี่ยเอ็กซ์โปเนนเชียลเคลื่อนที่วันสั้นและวันยาวของเส้นราคา



รูปที่ 3.1 แสดง กราฟของเส้น MACD

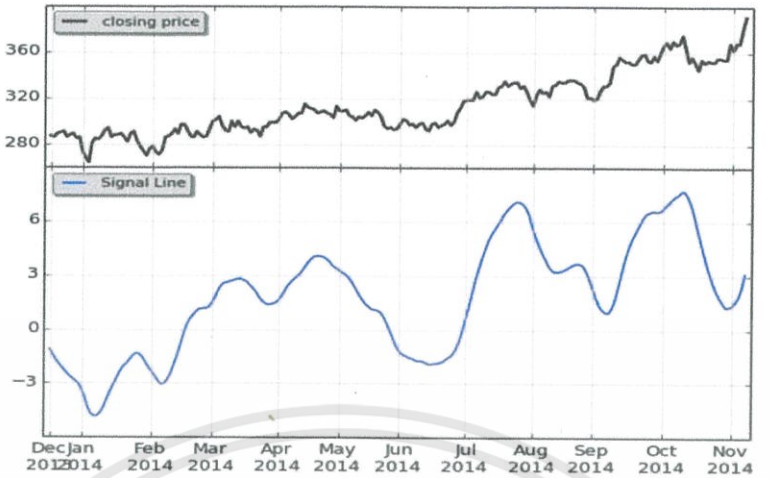
การใช้เส้น MACD ในการซื้อขายหุ้นจะต้องใช้ร่วมกับลำดับอีกลำดับหนึ่งเรียกว่า*เส้นสัญญาณ* หรือ Signal line ซึ่งเป็นเส้นค่าเฉลี่ยเคลื่อนที่แบบเอ็กซ์โปเนนเชียลของเส้น MACD

นิยามที่3.4 เส้นสัญญาณ $s = (s_{t_i} \mid t_i \in (\tau))$ คือ ลำดับซึ่งกำหนดโดย

$$s_{t_i} = EMA^{MACD}_{n_3}(t_i) \tag{3.3}$$

เมื่อ $EMA^{MACD}_{n_3}(t_i) = \sum_{j=0}^{i-1} \alpha_{n_3}(i-\alpha_{n_3})^j m_{t_{i-j}}$

จากนิยามที่3.4 จะได้ว่า เส้นสัญญาณคือเส้นค่าเฉลี่ยเอ็กซ์โปเนนเชียลเคลื่อนที่ n_3 วันของเส้น MACD



รูปที่ 3.2 แสดง กราฟของลำดับเส้นสัญญาณ

3.1.3 จุดตัดขึ้น-จุดตัดลงและนิยามคาบการซื้อขาย

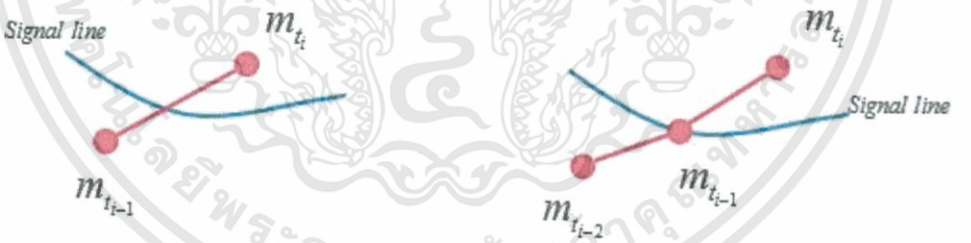
ในหัวข้อนี้จะนิยามจุดตัดขึ้น-จุดตัดลงและนิยามคาบการซื้อขายในงานวิจัยนี้

นิยามที่3.5 จุดตัดขึ้นที่ k เขียนแทนด้วยสัญลักษณ์ $t^{(\uparrow k)}$ คือจุดเวลา $t_i \in (\tau)$ จุดที่ k ที่สอดคล้องกับเงื่อนไขข้อใดข้อหนึ่งต่อไปนี้

1) $m_{t_i} > s_{t_i}$ และ $m_{t_{i-1}} < s_{t_{i-1}}$

2) $m_{t_i} > s_{t_i}$ และ $m_{t_{i-1}} = s_{t_{i-1}}$ และ $m_{t_{i-2}} < s_{t_{i-2}}$

จุดตัดขึ้นตามบทนิยามที่ 3.5 แสดงได้ดังรูปต่อไปนี้



รูปที่ 3.3 แสดง จุดตัดขึ้น

นิยามที่3.6 จุดตัดลง ที่ k เขียนแทนด้วยสัญลักษณ์ $t^{(\downarrow k)}$ คือจุดเวลา $t_i \in (\tau)$ จุดที่ k ที่สอดคล้องกับเงื่อนไขข้อใดข้อหนึ่งต่อไปนี้

1) $m_{t_i} < s_{t_i}$ และ $m_{t_{i-1}} > s_{t_{i-1}}$

2) $m_{t_i} < s_{t_i}$ และ $m_{t_{i-1}} = s_{t_{i-1}}$ และ $m_{t_{i-2}} > s_{t_{i-2}}$

จุดตัดลงตามบทนิยามที่ 3.6 แสดงได้ดังรูปต่อไปนี้

รูปที่ 3.4 แสดง จุดตัดลง

กำหนดสัญลักษณ์ ให้ $t^{(\downarrow k)}$ แทนจุดตัดลงและจุดตัดขึ้นที่ k ใน (τ) เขียนแทนด้วย $t^{(\uparrow k)}$ สำหรับทุกๆ $t \in (\tau)$ ต่อไปจะนิยามคาบการซื้อขายโดยใช้จุดตัดลงเป็นตัวกำหนด

นิยามที่3.7 คาบการซื้อขายที่ k ใน (τ) คือลำดับย่อยของลำดับเวลา (τ) ซึ่งมีจุดเริ่มต้นคาบอยู่ที่ $t_{(k,1)} = t^{(\downarrow k)}$ และจุดปลายคาบที่ $t_{(k,\theta_k)} = t^{(\downarrow k+1)}$ เขียนแทนด้วยสัญลักษณ์ $(\tau_k) = \{t_{(k,1)}, t_{(k,2)}, \ldots, t_{(k,\theta_k)}\}$ เมื่อ θ_k คือขนาดคาบการซื้อขายที่ k และ $t_{(k,j)}, j = 1,2,\ldots,\theta_k$ จุดเวลาที่ k ในคาบที่ k

นิยามที่3.8 รอบการซื้อขาย 1 รอบคือการซื้อ 1 ครั้ง และการขาย 1 ครั้ง ในคาบเดียวกันด้วยเงินทุนจำนวนหนึ่งโดยการขายจะขายหุ้นทั้งหมดจากการซื้อ โดยที่ถ้ากำหนดให้ (τ_k) เป็นคาบการซื้อขาย การซื้อขายรอบที่ j ในคาบ (τ_k) จะเขียนแทนด้วย $({}_{jk}t^{b\&s}) = ({}_{jk}t^{b}, {}_{jk}t^{s}, {}_{jk}C)$ เมื่อ

${}_{jk}t^{b} \in (\tau_k)$ คือ จุดเวลาซื้อ

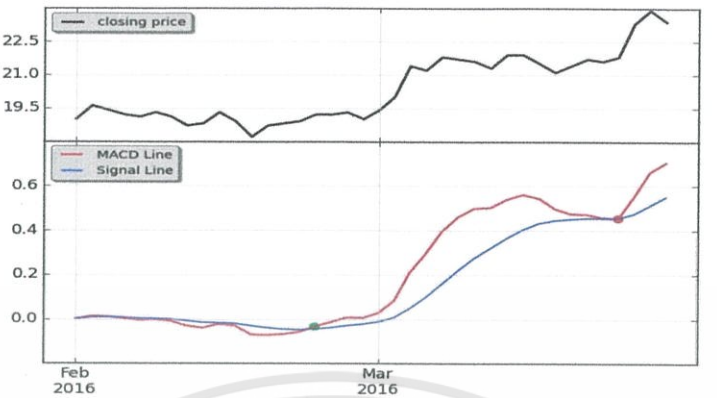
${}_{jk}t^{s} \in (\tau_k)$ คือ จุดเวลาขาย

${}_{jk}C > 0$ คือ เงินทุนที่ใช้ในการซื้อ

3.1.4 นิยามการซื้อขายด้วย MACD แบบดั้งเดิม

ในงานวิจัยนี้นิยามการซื้อขายด้วย *MACD* แบบดั้งเดิมด้วยซื้อที่จุดตัดขึ้นและขายที่จุดตัดลงตามนิยามต่อไปนี้

นิยามที่3.9 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k = 1,2,\ldots,r$ กำหนดให้ $t^{(\uparrow k)}$ และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_{k}t^{b}, {}_{k}t^{s}, C) = ({}_{1k}t^{b}, {}_{1k}t^{s}, C)$ ในคาบ (τ_k) โดยให้จุดซื้อคือจุด ${}_{k}t^{b} = t^{(\uparrow k)}$ และมีจุดขายคือจุด ${}_{k}t^{s} = t^{(\downarrow k)}$ เมื่อ $C > 0$ เป็นค่าคงที่ใดๆ



รูปที่ 3.5 แสดง กราฟของจุดซื้อ จุดขาย

3.2 การซื้อขายด้วย MACDR1

วิธีการซื้อขายด้วย MACDR1 จะกำหนดให้จุดซื้อคือจุดเวลาที่ 3 นับจากจุดตัดขึ้นถ้าเส้น MACD ยังคงอยู่เหนือเส้นสัญญาณ และกำหนดจุดขายเป็นจุดที่อัตรากำไรมีค่าถึง 3% แต่หากไม่พบจุดดังกล่าวก็ให้ขายที่จุดตัดลง

นิยามที่3.10 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ และ กำหนดให้ $t^{(\uparrow k)}$และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C)$ ในคาบ(τ_k) จุดซื้อ ${}_kt^b$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

1) $m_{t_{(k,j)}} > s_{t_{(k,j)}}$

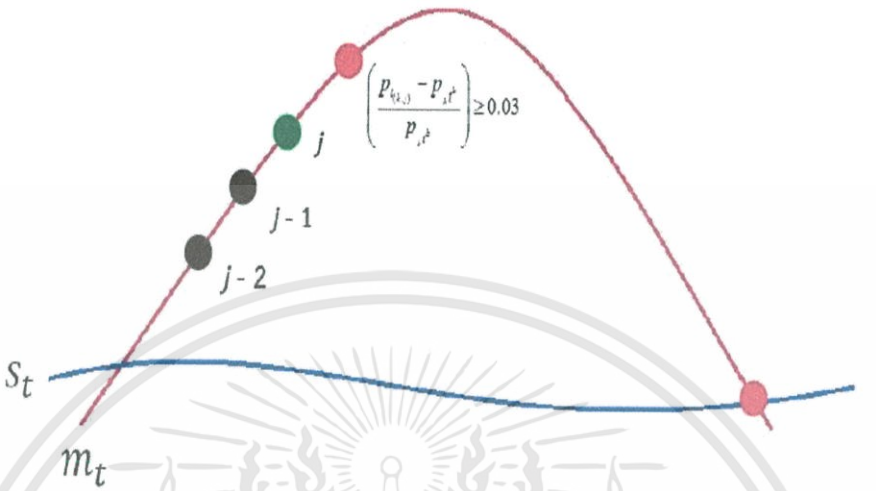
2) $m_{t_{(k,j-1)}} > s_{t_{(k,j-1)}}$

3) $t_{(k,j-2)}$ เป็นจุดตัดขึ้น

และถ้ามีจุดซื้อ ${}_kt^b$ ในคาบแล้ว จุดขาย ${}_kt^s$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

$$\left(\frac{p_{t_{(k,j)}} - p_{{}_kt^b}}{p_{{}_kt^b}}\right) \geq 0.03 \text{ หรือ } t_{(k,j)} = t_{(k,\theta_k)}$$

วิธีการซื้อขายด้วย MACDR1ตามบทนิยามที่ 3.10 แสดงได้ดังรูปต่อไปนี้



รูปที่ 3.6 แสดง การซื้อขายด้วยวิธี MACDR1

3.3 การซื้อขายด้วย MACDR2

วิธีการซื้อขาย MACDR2 มีหลักการกำหนดจุดซื้อขายเช่นเดียวกับ MACDR1โดยมีข้อกำหนดว่าจะซื้อถ้าผลต่างค่าของเส้น MACD กับค่าของเส้นสัญญาณ ณ จุดซื้อมีค่ามากกว่า หรือ เท่ากับร้อยละ 0.5 ของราคาหุ้น

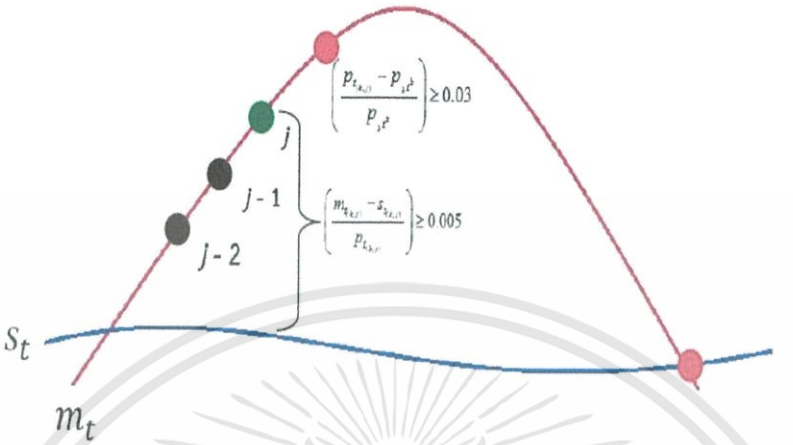
นิยาม3.11 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k = 1,2,\ldots,r$ และ กำหนดให้ $t^{(\uparrow k)}$และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_k t^b, {}_k t^s, C)$ในคาบ(τ_k) จุดซื้อ ${}_k t^b$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

1) $m_{t_{(k,j)}} > s_{t_{(k,j)}}$
2) $m_{t_{(k,j-1)}} > s_{t_{(k,j-1)}}$
3) $t_{(k,j-2)}$ เป็นจุดตัดขึ้น
4) $(\frac{m_{t_{(k,j)}} - s_{t_{(k,j)}}}{p_{t_{(k,j)}}}) \geq 0.005$

และถ้ามีจุดซื้อ ${}_k t^b$ ในคาบแล้ว จุดขาย ${}_k t^s$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

$$(\frac{p_{t_{(k,j)}} - p_{{}_k t^b}}{p_{{}_k t^b}}) \geq 0.03 \text{ หรือ } t_{(k,j)} = t_{(k,\theta_k)}$$

วิธีการซื้อขายด้วย MACDR2ตามบทนิยามที่ 3.11 แสดงได้ดังรูปต่อไปนี้



รูปที่ 3.7 แสดงการซื้อขายด้วยวิธี MACDR2

3.4 การซื้อขายด้วย MACDP, MACDP1 และ MACDP2

ในปี 2556 นายพิพรรธน์ สำเภากิจและคณะได้จัดทำปัญหาพิเศษเรื่อง *"การวิเคราะห์การซื้อขายหุ้นในตลาดหลักทรัพย์แห่งประเทศไทยโดยใช้เส้นสัญญาณดัดแปลงและการอนุมานค่าน้ำหนักการซื้อขายด้วยตรรกศาสตร์ฟัซซี"* ขึ้นแสดงให้เห็นว่าการใช้เส้นสัญญาณดัดแปลงแทนเส้นสัญญาณเดิม สามารถทำให้การซื้อขายหุ้นด้วยวิธี MACD มีสัญญาณซื้อขายที่เร็วขึ้น ส่งผลให้อัตราความสำเร็จและอัตรากำไรสูงขึ้น และเมื่อทำการทดสอบเปรียบเทียบกับ MACDR2 พบว่าแม้อัตราความสำเร็จจะด้อยกว่า แต่สามารถให้อัตรากำไรเฉลี่ยที่สูงกว่าในการทดสอบกับข้อมูลราคาปิดรายวันของหุ้นแต่ละตัวในกลุ่ม SET-100 ตั้งแต่วันที่ 28 พฤศจิกายน ถึง 28 พฤศจิกายน พ.ศ. 2556

การใช้เส้นสัญญาณดัดแปลงมีจุดอ่อนคือจะเกิดสัญญาณหลอกที่มักจะทำให้การซื้อขายขาดทุน การซื้อขายแบบถ่วงน้ำหนักโดยใช้ค่าน้ำหนักการซื้อขายที่อนุมานจากตรรกศาสตร์ฟัซซีสามารถช่วยบรรเทาจุดอ่อนนี้และทำให้อัตรากำไรเฉลี่ยและอัตราความสำเร็จสูงขึ้นได้

ในบทวิจัยนี้ใช้ข้อมูลราคาระยะเวลาเพียง 5 เดือน ซึ่งอาจทำให้ผลลัพธ์ที่ได้ไม่รัดกุมมาก แนวทางการซื้อขายหุ้นโดยใช้เส้นสัญญาณดัดแปลงนี้ยังควรต้องทดสอบข้อมูลที่หลากหลายและมีระยะเวลานานกว่านี้

ทั้ง MACDP, MACDP1 และ MACDP2 เป็นดัชนีที่ปรับปรุงให้ดัชนีส่งสัญญาณให้มีการซื้อขายเร็วขึ้น ซึ่งเป็นการแก้ไขจุดด้อยของการซื้อขายด้วยMACD ดั้งเดิม โดยวิธียืดขยายเส้นสัญญาณให้ตัดกับเส้น MACD ก่อนจุดตัดเดิมโดยการประยุกต์ใช้ตรรกศาสตร์ฟัซซีเพื่อหาคาดการณ์จุดตัดของเส้น MACD ดั้งเดิม

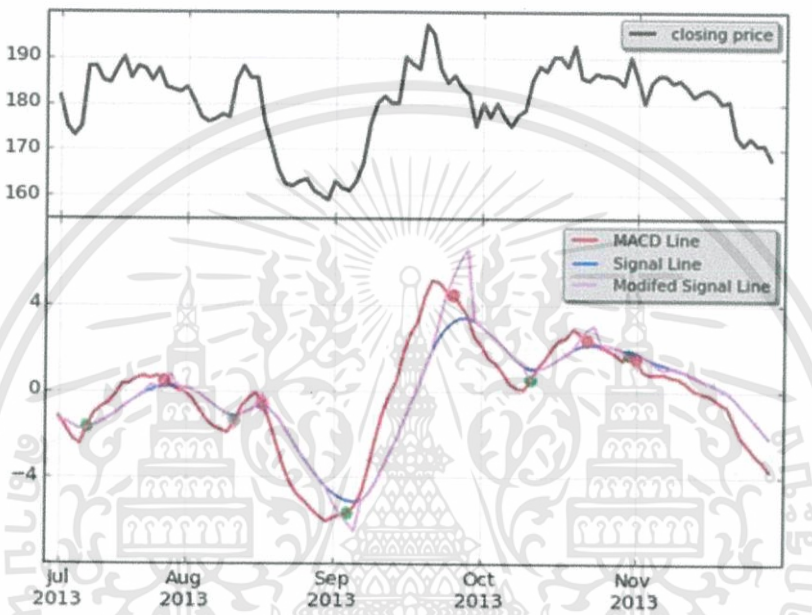
วิธีการซื้อขายด้วย MACDP,MACDP1 และ MACDP2 มีหลักการกำหนดจุดซื้อขายเช่นเดียวกับ MACD แบบดั้งเดิม, MACDR1และ MACDR2 ตามลำดับ

ถ้าให้ $s^d = (s^d_{t_i} \mid t_i \in (\tau))$ เป็นเส้นสัญญาณดัดแปลง

การซื้อขายด้วย MACDP

นิยามที่3.12 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ กำหนดให้ $t^{(\uparrow k)}$ และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C)=({}_{1k}t^b, {}_{1k}t^s, C)$ ในคาบ (τ_k) โดยให้จุดซื้อคือจุด ${}_kt^b = t^{(\uparrow k)}$ และมีจุดขายคือจุด ${}_kt^s = t^{(\downarrow k)}$ เมื่อ $C > 0$ เป็นค่าคงที่ใดๆ

วิธีการซื้อขายด้วย MACDP ตาม MACD แบบดั้งเดิม แสดงได้ดังรูปต่อไปนี้



รูปที่ 3.8 แสดง การซื้อขาย MACDP

การซื้อขายด้วย MACDP1

นิยามที่3.13 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ และ กำหนดให้ $t^{(\uparrow k)}$ และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C)$ ในคาบ (τ_k) จุดซื้อ ${}_kt^b$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

1) $m_{t_{(k,j)}} > s^d{}_{t_{(k,j)}}$

2) $m_{t_{(k,j-1)}} > s^d{}_{t_{(k,j-1)}}$

3) $t_{(k,j-2)}$ เป็นจุดตัดขึ้น

และถ้ามีจุดซื้อ ${}_kt^b$ ในคาบแล้ว จุดขาย ${}_kt^s$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

$$\left(\frac{p_{t_{(k,j)}} - p_{{}_kt^b}}{p_{{}_kt^b}}\right) \geq 0.03 \text{ หรือ } t_{(k,j)} = t_{(k,\theta_k)}$$

การซื้อขายด้วย MACDP2

นิยาม3.14 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ และ กำหนดให้ $t^{(\uparrow k)}$และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C)$ ในคาบ(τ_k) จุดซื้อ ${}_kt^b$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

1) $m_{t_{(k,j)}} > s^d{}_{t_{(k,j)}}$

2) $m_{t_{(k,j-1)}} > s^d{}_{t_{(k,j-1)}}$

3) $t_{(k,j-2)}$ เป็นจุดตัดขึ้น

4) $(\dfrac{m_{t_{(k,j)}} - s^d{}_{t_{(k,j)}}}{p_{t_{(k,j)}}}) \geq 0.005$

และถ้ามีจุดซื้อ ${}_kt^b$ ในคาบแล้ว จุดขาย ${}_kt^s$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

$$(\frac{p_{t_{(k,j)}} - p_{{}_kt^b}}{p_{{}_kt^b}}) \geq 0.03 \text{ หรือ } t_{(k,j)} = t_{(k,\theta_k)}$$

3.5 การซื้อขายด้วย MACDK, MACDK1 และ MACDK2

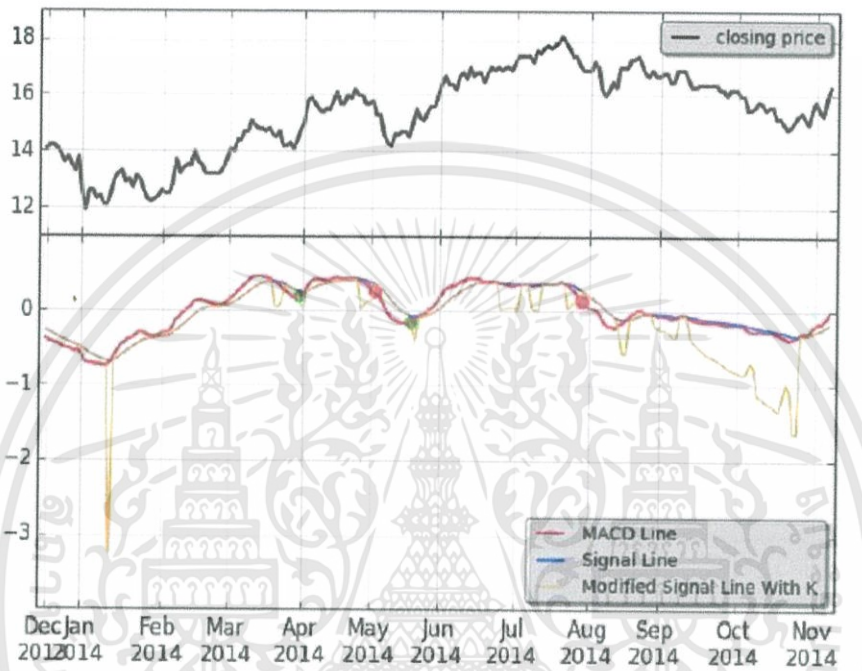
ในปี 2557 นายวุฒิชัย บุญชมและคณะได้จัดทำปัญหาพิเศษเรื่อง *"การพัฒนาเส้นสัญญาณในดัชนี MACD ด้วยฟังก์ชันยืดขยาย"* ขึ้นแสดงให้เห็นว่าการใช้เส้นสัญญาณใหม่ในการซื้อขายแทนการซื้อขายในเส้นสัญญาณเดิม เส้นสัญญาณใหม่สามารถส่งสัญญาณซื้อได้เร็วขึ้น ส่งผลให้มีอัตรากำไรที่สูงขึ้นเมื่อเทียบกับวิธีการซื้อขายด้วยเส้นสัญญาณอื่นๆ แต่พบว่าเส้นสัญญาณใหม่มีอัตราความสำเร็จที่ต่ำกว่า MACDR1 MACDR2 และ MACDP โดย MACDK สามารารถทำอัตรากำไรเฉลี่ยได้สูงมากกว่าเส้นสัญญาณแบบดั้งเดิมถึงสองเท่า อย่างไรก็ตาม อัตราความสำเร็จสามารถเพิ่มขึ้นด้วยวิธีการซื้อขายแบบ MACDK0.011 ที่ให้อัตราความสำเร็จสูงที่สุด แต่มีอัตรากำไรเฉลี่ยลดลงจากปัญหาพิเศษนี้ สามารถนำ MACD วิธีต่างๆที่พัฒนามาประยุกต์ใช้ในการลงทุนรูปแบบต่างๆตามความเหมาะสมของผู้ลงทุน เช่น ในกรณีที่ผู้ลงทุนมีเงินลงทุนที่จำกัด สามารถนำวิธี MACDที่ให้อัตรากำไรที่ต่ำ แต่อัตราความสำเร็จที่สูง เพื่อป้องกันความเสี่ยงในการลงทุน ส่วนในกรณีที่ผู้ลงทุนยอมรับความเสี่ยงได้ และมีเงินลงทุนสารองมาก ก็สามารถเลือกวิธี MACD ที่ให้อัตราความสำเร็จที่ต่ำ แต่อัตรากำไรที่สูง ซึ่งทั้งสองวิธีนี้สามารถนำไปใช้ได้ตามความเหมาะสมของผู้ลงทุนนั้นๆ โดยทำการทดสอบกับข้อมูลราคาปิดของหุ้นย้อนหลัง 5 ปี ของหุ้นแต่ละตัวในกลุ่ม SET50 และSET100 ระหว่างวันที่ 6 พฤศจิกายน พ.ศ. 2552 ถึง 7 พฤศจิกายน พ.ศ. 2557

ทั้ง MACDK, MACDK1 และ MACDK2 เป็น MACD ซึ่งทำการปรับปรุงเส้นสัญญาณเพื่อแก้ปัญหาการการส่งสัญญาณซื้อ-ขายล่าช้า เช่นเดียวกับ MACDP แต่มีข้อแตกต่างกันที่ MACDK จะทำการยืดขยาดหรือ หดเส้นสัญญาณเดิมด้วยฟังก์ชันยืดขยายมาจนทำให้เกิดเส้นสัญญาณใหม่ โดยกำหนดวิธีการซื้อขายเช่นเดียวกับ MACD แบบดั้งเดิม, MACDR1และ MACDR2 ตามลำดับ

ถ้าให้ $\tilde{s} = (\tilde{s}_{t_i} \mid t_i \in (\tau))$ เป็นเส้นสัญญาณใหม่ที่ถูกยืดขยายด้วยฟังก์ชันยืดขยาย

วิธีการซื้อขาย MACDK

นิยามที่3.15 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ กำหนดให้ $t^{(\uparrow k)}$ และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C)=({}_{1k}t^b, {}_{1k}t^s, C)$ ในคาบ (τ_k) โดยให้จุดซื้อคือจุด ${}_kt^b = t^{(\uparrow k)}$ และมีจุดขายคือจุด ${}_kt^s = t^{(\downarrow k)}$ เมื่อ $C > 0$ เป็นค่าคงที่ใดๆ



รูปที่ 3.9 แสดง การซื้อขาย MACDK

วิธีการซื้อขาย MACDK1

นิยามที่ 3.16 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ และ กำหนดให้ $t^{(\uparrow k)}$ และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C)$ ในคาบ (τ_k) จุดซื้อ ${}_kt^b$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

1) $m_{t_{(k,j)}} > \tilde{s}_{t_{(k,j)}}$

2) $m_{t_{(k,j-1)}} > \tilde{s}_{t_{(k,j-1)}}$

3) $t_{(k,j-2)}$ เป็นจุดตัดขึ้น

ถ้ามีจุดซื้อ ${}_kt^b$ ในคาบแล้ว จุดขาย ${}_kt^s$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

$$\left(\frac{p_{t_{(k,j)}} - p_{{}_kt^b}}{p_{{}_kt^b}}\right) \geq 0.03 \text{ หรือ } t_{(k,j)} = t_{(k,\theta_k)}$$

วิธีการซื้อขาย MACDK2

นิยามที่ 3.17 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ และ กำหนดให้ $t^{(\uparrow k)}$ และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C)$ ในคาบ (τ_k) จุดซื้อ ${}_kt^b$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

1) $m_{t_{(k,j)}} > \tilde{s}_{t_{(k,j)}}$

2) $m_{t_{(k,j-1)}} > \tilde{s}_{t_{(k,j-1)}}$

3) $t_{(k,j-2)}$ เป็นจุดตัดขึ้น

4) $(\frac{m_{t_{(k,j)}} - s_{t_{(k,j)}}}{p_{t_{(k,j)}}}) \geq 0.005$

ถ้ามีจุดซื้อ ${}_kt^b$ ในคาบแล้ว จุดขาย ${}_kt^s$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

$$(\frac{p_{t_{(k,j)}} - p_{{}_kt^b}}{p_{{}_kt^b}}) \geq 0.03 \quad \text{หรือ} \quad t_{(k,j)} = t_{(k,\theta_k)}$$

3.6 การซื้อขายด้วย MACDF, MACDF1 และ MACDF2

ในปี 2557 นายอนิรุธ หนูตอและคณะได้จัดทำปัญหาพิเศษเรื่อง "*ดัชนี MACD ฟัซซีประยุกต์กับ SET50 ในตลาดหลักทรัพย์แห่งประเทศไทย*" ขึ้นแสดงให้เห็นว่าในการสร้างแมคดฟัซซีโดยใช้ความรู้ MACD ดั้งเดิมและตรรกศาสตร์ฟัซซีและทำการทดสอบหุ้นในกลุ่ม SET50 ตั้งแต่วันที่ 4 มกราคม 2555 ถึง 30 ธันวาคม พ.ศ.2557 พบว่าแมคดีฟัซซีสามารถให้อัตราความสำเร็จและอัตรากำไรสูงกว่า MACD ดั้งเดิม และเมื่อทดสอบเปรียบเทียบกับ MACDR2 ให้อัตราความสำเร็จและอัตรากำไรเท่ากัน แม้ว่าหากทดสอบเปรียบเทียบกับ MACDR1 มีอัตราความสำเร็จและอัตรากำไรต่ำกว่า ดังนั้นแมคดีฟัซซีจึงเป็นตัวเลือกหนึ่งในการนำไปตัดสินใจลงทุนซื้อขายหุ้น

การเลือกใช้เครื่องมือวิเคราะห์ในการซื้อขายหุ้นนั้น ตัวชี้วัดคืออัตราความสำเร็จ เพราะ นักลงทุนนั้นต้องการอัตราความสำเร็จที่สูง จึงเลือกใช้เครื่องมือวิเคราะห์ตามอัตราความสำเร็จและวางกลยุทธ์ในการลงทุนด้วยอัตรากำไร ถ้าเครื่องมือวิเคราะห์นั้นให้อัตราความสำเร็จต่ำควรใช้เครื่องมือวิเคราะห์อื่นมาช่วยในการตัดสินใจในการซื้อขาย

ในบทวิจัยนี้ใช้ข้อมูลราคาหุ้นกลุ่ม SET50 ซึ่งคือหุ้นเพียงหุ้นเดียวอาจทาให้ผลลัพธ์ที่ได้ไม่รัดกุมมาก แนวทางการซื้อขายหุ้นโดยใช้เส้นแมคดีฟัซซีนี้ควรทดสอบข้อมูลที่หลากหลายมากกว่านี้

MACDF เป็นการปรับปรุง MACD ดั้งเดิม โดยที่ MACD ดั้งเดิมจะพิจารณาแค่ราคาปิดในการนำมาสร้างเส้น MACD และเส้นสัญญาณ แต่ MACDF พิจารณา ราคาปิด ราคาเปิด และราคาเฉลี่ย ในการนำมาสร้างเส้น MACDF และเส้น SignalF โดยนำตรรกศาสตร์ฟัซซีมาประยุกต์ใช้ค่าเฉลี่ยเอ็กซ์โปเนนเชียลเคลื่อนที่ฟัซซี (Fuzzy exponential movingaverage : FEMA)

นิยามที่ 3.18 กำหนดให้ (τ) เป็นลำดับของจุดเวลาทั้งหมดที่มีการเก็บข้อมูลและกำหนดให้แทน $(\mathrm{P})-\left\{\hat{p}_i \mid \hat{p}_i - Tri(a_1^{(i)}, a_m^{(i)}, a_2^{(i)}), i=1,2,...,t\right\}$ ลำดับของราคา *ค่าเฉลี่ยเอ็กซ์โปเนนเชียลเคลื่อนที่ฟัซซี* n $(n<t)$ วัน ณ วันที่ t เขียนแทนสัญลักษณ์ $FEMA_n(t)$ นิยามโดย

$$FEMA_n(t) = Tri(EMA_n^{a_1}(t), EMA_n^{a_m}(t), EMA_n^{a_2}(t)) \qquad ;t>n$$

เมื่อ $$EMA_n^{a_k}(t) = EMA_n^{a_k}(t-1) + \frac{2}{n+1}(a_k^{(t)} + (-1)EMA_n^{a_k}(t-1))$$

$$EMA_n^{a_k}(n) = w_1 a_k^{(1)} + w_2 a_k^{(2)} + ... + w_n a_k^{(n)} \;; k = 1, m, 2$$

และ $$w_i = \frac{2}{n+1}\left(1-\frac{2}{n+1}\right)^{n-i} \qquad ;i=1,2,...,n$$

นิยามที่ 3.19 ตัวดำเนินการดีฟัซซีฟิเคชันสำหรับค่าเฉลี่ยเอ็กซ์โปเนนเชียลเคลื่อนที่ฟัซซี เขียนแทนด้วยสัญลักษณ์ $D_{FEMA}(t)$ นิยามโดย

$$D_{FEMA}(t) = \frac{EMAa_n^{a_1}(t) + 8EMAa_n^{a_m}(t) + EMAa_n^{a_2}(t)}{10}$$

นิยามที่ 3.20 MACDF หรือเส้น MACDF คือลำดับผลต่างของค่าเฉลี่ยเคลื่อนที่แบบเอ็กซ์โปเนนเชียลฟัซซีสองลำดับที่มีค่าพารามิเตอร์ n_1 และ n_2 ที่ซึ่ง $\mathrm{n}_1 < \mathrm{n}_2$ เขียนแทนด้วย $MACDF = \left(MACDF(t) \middle| t \in (\tau)\right)$ โดยที่

$$MACDF(t) = FEMA_{n_1}^{price}(t) - FEMA_{n_2}^{price}(t)$$

การใช้เส้น MACD ในการซื้อขายหุ้นจะต้องใช้ร่วมกับลำดับอีกลำดับหนึ่งเรียกว่า*เส้นสัญญาณ*ซึ่งคือค่าเฉลี่ยเคลื่อนที่แบบเอ็กซ์โปเนนเชียลของเส้น MACD

นิยามที่ 3.21 ให้ $SignalF = \left(SignalF(t) \middle| t \in (\tau)\right)$ แทนลำดับของเส้นสัญญาณฟัซซี โดยที่

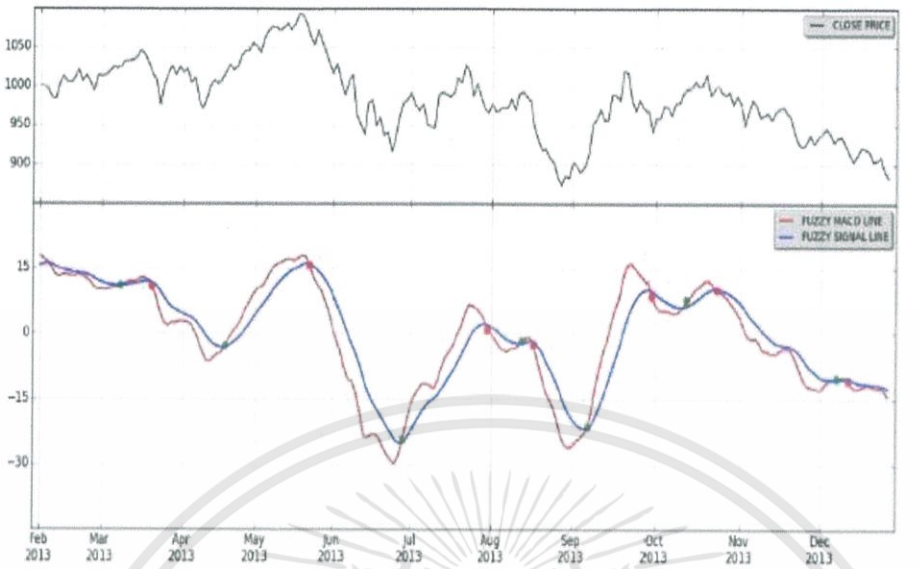
$$SignalF(t) = EMA_{n_3}^{MACDF}(t)$$

ถ้าให้ $sf = (sf_{\mathrm{t_i}} \mid \mathrm{t_i} \in (\tau))$ เป็นเส้นสัญญาณฟัซซี และ $mf = (mf_{\mathrm{t_i}} \mid \mathrm{t_i} \in (\tau))$ เป็นเส้นแม็คดีฟัซซี

วิธีซื้อขายด้วย MACDF

วิธีการซื้อขายด้วย MACDF จะซื้อที่จุดตัดขึ้นและขายที่จุดตัดลง

นิยามที่ 3.22 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ กำหนดให้ $t^{(\uparrow k)}$ และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C) = ({}_{1k}t^b, {}_{1k}t^s, C)$ ในคาบ (τ_k) โดยให้จุดซื้อคือจุด ${}_kt^b = t^{(\uparrow k)}$ และมีจุดขายคือจุด ${}_kt^s = t^{(\downarrow k)}$ เมื่อ $C > 0$ เป็นค่าคงที่ใดๆ



รูปที่ 3.10 แสดง วิธีการซื้อขาย MACDF

วิธีการซื้อขาย MACDF1

นิยามที่ 3.23 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ และ กำหนดให้ $t^{(\uparrow k)}$ และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C)$ ในคาบ (τ_k) จุดซื้อ ${}_kt^b$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

1) $mf_{t_{(k,j)}} > sf_{t_{(k,j)}}$

2) $mf_{t_{(k,j-1)}} > sf_{t_{(k,j-1)}}$

3) $t_{(k,j-2)}$ เป็นจุดตัดขึ้น

และถ้ามีมีจุดซื้อ $t^b_{(k,j)}$ ในคาบ จุดขาย $t^s_{(k,j)}$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

$$\left(\frac{p_{t_{(k,j)}} - p_{t^b_{(k,j)}}}{p_{t^b_{(k,j)}}}\right) \geq 0.03 \text{ หรือ } t_{(k,j)} = t_{(k,\theta_k)}$$

วิธีการซื้อขาย MACDF2

นิยามที่ 3.24 กำหนดให้ (τ_k) แทนคาบเวลาซื้อขายที่ $k=1,2,\ldots,r$ และ กำหนดให้ $t^{(\uparrow k)}$ และ $t^{(\downarrow k)}$ แทนจุดตัดขึ้นและจุดตัดลงตามลำดับที่ k ในคาบเวลา (τ) กำหนดการซื้อขาย $({}_kt^b, {}_kt^s, C)$ ในคาบ (τ_k) จุดซื้อ ${}_kt^b$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

1) $mf_{t_{(k,j)}} > sf_{t_{(k,j)}}$

2) $mf_{t_{(k,j-1)}} > sf_{t_{(k,j-1)}}$

3) $t_{(k,j-2)}$ เป็นจุดตัดขึ้น

4) $$\left(\frac{mf_{t_{(k,j)}} - sf_{t_{(k,j)}}}{p_{t_{(k,j)}}}\right) \geq 0.005$$

และถ้ามีมีจุดซื้อ $t^b_{(k,j)}$ ในคาบ จุดขาย $t^s_{(k,j)}$ คือจุด $t_{(k,j)} \in (\tau_k)$ ใดๆที่ซึ่ง

$$\left(\frac{p_{t_{(k,j)}} - p_{t^b_{(k,j)}}}{p_{t^b_{(k,j)}}}\right) \geq 0.03 \text{ หรือ } t_{(k,j)} = t_{(k,\theta_k)}$$

3.7 อัตราส่วนทางการเงินที่สำคัญ

อัตราส่วนราคาต่อกำไรต่อหุ้น (Price to Earnings Ratio: P/E Ratio)

อัตราส่วนนี้คำนวณจาก ราคาตลาดของหุ้น หารด้วย กำไรสุทธิต่อหุ้น (Earnings Per Share : EPS) แสดงให้เห็นว่าผู้ลงทุนยินดีจะจ่ายเงินซื้อหุ้นนั้นเป็นกี่เท่าของทุกๆ 1 บาทของกำไรสุทธิของบริษัท มีสูตรการคำนวณดังนี้

$$\text{P/E ratio} = \frac{\text{ราคาหุ้นปัจจุบัน} \times (\text{(จำนวนหุ้นสามัญ} + \text{จำนวนหุ้นบุริมสิทธิ์)} - \text{จำนวนหุ้นซื้อคืน})}{\text{กำไรงวด } 12 \text{ เดือนล่าสุด}}$$

ซึ่งตลาดหลักทรัพย์แห่งประเทศไทยไม่ใช้ "กำไรสุทธิต่อหุ้นสามัญ" ในการคำนวณค่า P/E แต่จะใช้ "กำไรสุทธิที่บริษัทนั้นทำได้ในรอบระยะเวลา 12 เดือนล่าสุด"

อัตราส่วนราคาต่อมูลค่าทางบัญชีต่อหุ้น (Price to Book Value : P/BV Ratio)

อัตราส่วนนี้คำนวณจาก ราคาตลาดของหุ้น หารด้วย มูลค่าทางบัญชีต่อหุ้น แสดงให้ผู้ลงทุนเห็นว่า ราคาหุ้น ณ ขณะนั้น สูงเป็นกี่เท่าของมูลค่าทางบัญชีของหุ้นดังกล่าว มีสูตรการคำนวณดังนี้

$$\text{P/BV ratio} = \frac{\text{ราคาตลาดของหุ้น}}{\text{มูลค่าทางบัญชีต่อหุ้น}}$$

โดย

$$\text{มูลค่าทางบัญชีต่อหุ้น} = \frac{\text{สินทรัพย์} - \text{หนี้สิน}}{\text{จำนวนหุ้นจดทะเบียนที่ชำระแล้ว}}$$

เงินปันผลตอบแทน (Dividend Yield)

เงินปันผลตอบแทนใช้เปรียบเทียบ อัตราผลตอบแทนการลงทุนที่ได้จากเงินปันผล ของหุ้นที่มีราคาแตกต่างกัน คำนวณได้จาก

$$\text{เงินปันผลตอบแทน (Dividend Yield)} = \frac{\text{เงินปันผล(ต่อปี)ต่อหุ้น}}{\text{ราคาหุ้น}}$$

เงินปันผลตอบแทน (Dividend Yield) นอกจากจะใช้เปรียบเทียบอัตราผลตอบแทน จากเงินปันผลของหุ้น ที่มีราคาต่างกันแล้ว ยังสามารถใช้เป็นปัจจัยในการเปรียบเทียบหุ้นนั้นๆ กับการลงทุนในหลักทรัพย์ประเภทอื่น เช่น หุ้นกู้ การฝากเงิน หรือพันธบัตร

นักลงทุนที่ไม่ปรารถนาที่จะรับความเสี่ยงจากความผันผวนของราคาหุ้น เช่น นักลงทุนรายใหม่ หรือนักลงทุนที่ถอนเงินจากการฝากเงิน มาลงทุนในหุ้น อาจมองหาหุ้นที่มี เงินปันผลตอบแทน (Dividend Yield) ที่สูง

อย่างไรก็ตาม เงินปันผลตอบแทน (Dividend Yield) คำนวณจากการจ่ายเงินปันผลในอดีต ในรอบ 1 ปี ที่ผ่านมา ไม่สามารถรับรองได้ว่า การจ่ายเงินปันผลจะเหมือนเดิมในอนาคต นอกจากนี้เงินต้นจากการลงทุนจะเปลี่ยนแปลงไปตามราคาหุ้นที่เปลี่ยนไปเมื่อทำการขายหุ้น ดังนั้นตัวเลขจึงเป็นเพียงตัวบ่งชี้ เพื่อการตัดสินใจของนักลงทุน

กำไรต่อหุ้น (EARNING PER SHARE)

กำไรต่อหุ้น (Earning per Share หรือ EPS) คืออัตราส่วนกำไรสุทธิต่อหุ้นหรือส่วนกำไรสุทธิแบ่งเฉลี่ยให้แก่หุ้นสามัญแต่ละหุ้นของบริษัท แสดงให้เห็นถึงกำไรของบริษัทเมื่อเทียบกับจำนวนหุ้นทั้งหมด สามารถคำนวณได้โดยการนำกำไรสุทธิรอบ 12 เดือนล่าสุดเป็นตัวตั้งแล้วหารด้วยส่วนของผู้ถือหุ้น ใช้สูตรเบื้องต้นดังต่อไปนี้

กำไรต่อหุ้น = กำไรสุทธิ / จำนวนหุ้นของบริษัทที่ชำระแล้ว

หลักการตีความจาก EPS นั้นไม่ตายตัว แต่โดยทั่วไปแล้วยิ่งเยอะเท่าไหร่ก็ยิ่งดี EPS เยอะหมายความว่าบริษัทมีกำไรมาก มีการเงินที่แข็งแรงมั่นคง จึงเป็นบริษัทที่น่าเชื่อถือสำหรับการลงทุน ในส่วนของการวิเคราะห์อาจดู EPS ย้อนหลังและเปรียบเทียบกับบริษัทที่อยู่ในกลุ่มอุตสาหกรรมและหมวดธุรกิจเดียวกัน อย่างไรก็ตาม การดูเพียงแค่ EPS อย่างเดียวคงไม่สะท้อนถึงแง่มุมต่าง ๆ ของบริษัทออกมาทั้งหมดเช่น สองบริษัทมี EPS ที่เท่ากันแต่อีกบริษัทใช้ต้นทุนน้อยกว่า อาจตีความได้ว่าสามารถใช้เงินลงทุนได้อย่างมีประสิทธิภาพมากกว่าเป็นต้น

ปันผลต่อหุ้น (dividend per share)

คือเงินปันผลต่อหุ้นที่จะแบ่งสรรให้ผู้ถือหุ้น โดยอาจแบ่ง EAT ส่วนหนึ่งสำหรับไว้ลงทุนต่อหรือขยายกิจการตามความเหมาะสม และอีกส่วนหนึ่งแบ่งสรรให้แก่ผู้ถือหุ้น มีค่ายิ่งสูงยิ่งดี ซึ่งจะมีสูตรการคำนวณ คือ

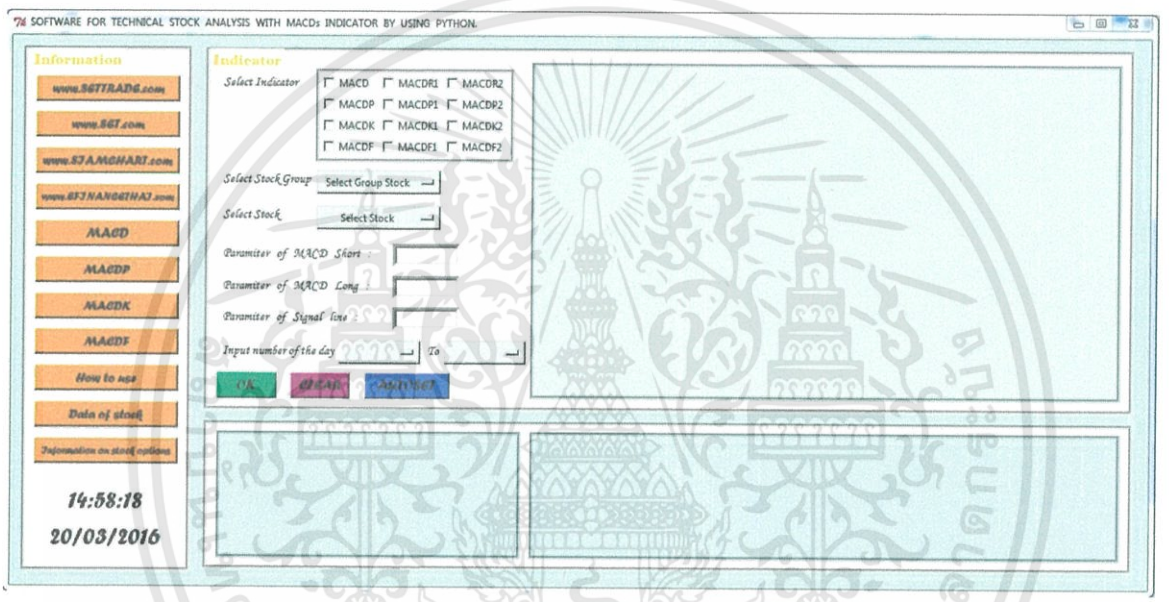
ปันผลต่อหุ้น = กำไรสุทธิของบริษัท / จำนวนหุ้นสามัญที่เรียกชำระของบริษัท

บทที่4

ผลการวิจัยดำเนินงานและอภิปรายผล

ผลการวิจัยดำเนินงาน

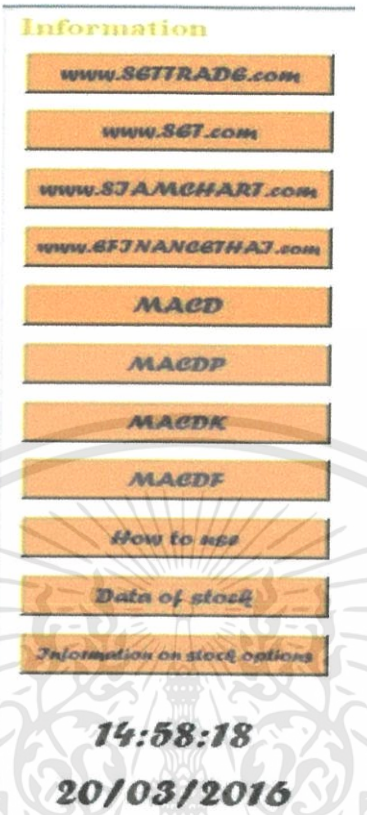
เมื่อทำการสร้าง GUI ใส่โค้ดต่างๆลงในโปรแกรมและตรวจสอบความถูกต้องของโปรแกรมแล้วจะได้โปรแกรมออกมาดังรูปที่ 4.1



รูปที่ 4.1 แสดง หน้าต่างของโปรแกรมเมื่อเริ่มต้น

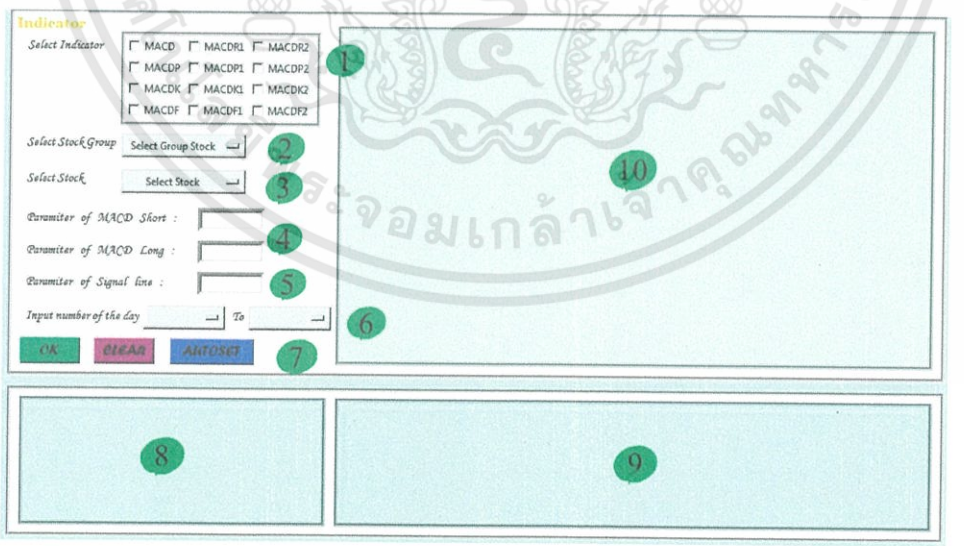
จากนั้นจะได้คู่มือการใช้งานโปรแกรมดังนี้

<u>ส่วนที่ 1</u> เราเรียกว่าส่วน Decoration คือส่วนที่เราจะนำเว็บไซต์ที่จำเป็น วัน เวลา วิธีใช้งานโปรแกรม รวมถึงความรู้เกี่ยวกับการวิเคราะห์แบบต่างๆมาใส่ไว้เพื่อให้ user ได้ศึกษา



รูปที่ 4.2 แสดง หน้าต่างโปรแกรมส่วนที่ 1

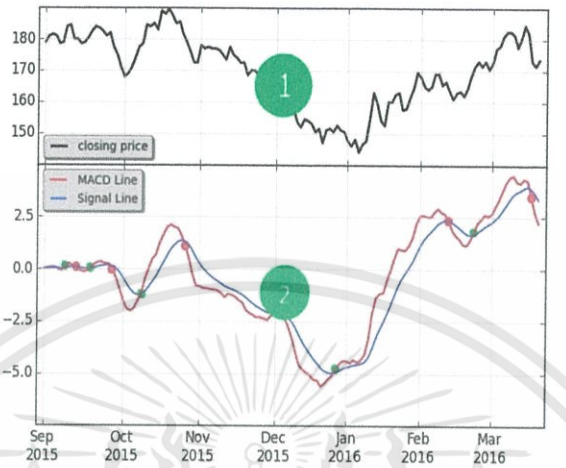
<u>ส่วนที่ 2</u> เราเรียกว่าส่วน Indicator คือส่วนที่เกี่ยวข้องกับการคำนวณในทุกๆแบบซึ่งในส่วนนี้จะมีองค์ประกอบดังนี้



รูปที่ 4.3 แสดง หน้าต่างโปรแกรมส่วนที่ 2

1. เป็นส่วนของการเลือกวิธีที่เราจะใช้ในการคำนวณหาผลลัพธ์ได้แก่ MACD, MACDR1, MACDR2, MACDP, MACDP1, MACDP2, MACDK, MACDK1, MACDK2, MACDF, MACDF1, MACDF2
2. เป็นส่วนของการเลือกกลุ่มชนิดของหุ้นที่เราต้องการจะใช้ในการคำนวณ เช่น SET50, SET100, SETHD, etc.
3. เป็นส่วนของการเลือกรายชื่อหุ้นที่เราต้องการจะใช้ในการคำนวณ เช่น ADVANC, AOT, BA, BANPU, BBL, etc. ตามกลุ่มชนิดของหุ้นที่เราได้ทำการเลือกในหมายเลข 2.
4. เนื่องจากการสร้างเส้น MACD เกิดการเส้นค่าเฉลี่ยราคาหุ้นในวันสั้นกับวันยาว ส่วนนี้จึงเป็นส่วนที่ใช้ในการกำหนดจำนวนของวันสั้นและวันยาวในการสร้างเส้น MACD
5. เป็นส่วนที่ใช้ในการกำหนดจำนวนวันของเส้น Signal Line
6. เป็นส่วนที่ใช้ในการกำหนดวันที่เราต้องการนำข้อมูลราคาหุ้นใช้จากวันที่เท่าไหร่ถึงวันที่เท่าไหร่
7. เป็นส่วนของปุ่ม
   - OK ใช้สั่งรันโปรแกรมเพื่อทำการคำนวณซึ่งผลการคำนวณจะไปแสดงที่หมายเลข 10
   - CLEAR ที่ใช้ในการล้างข้อมูลที่เราได้ทำการเลือกและใส่ค่าเข้าไปในแต่ละส่วนให้กลับสู่ค่าเริ่มต้นใหม่อีกครั้ง
   - Auto set ใช้ตั้งค่าพื้นฐานต่างๆที่นักลงทุนนิยมใช้
8. เป็นส่วนที่แสดงข้อมูลหุ้น
9. เป็นส่วนของการแสดงการเปลี่ยนแปลงของหุ้น
10. เป็นส่วนที่แสดงผลลัพธ์การคำนวณ

<u>ส่วนที่3</u> เมื่อโปรแกรมได้ทำการคำนวณในส่วนที 2 แล้วจะมีหน้าต่างกราฟของแต่ละวิธีปรากฏขึ้นมาบนหน้าจอ



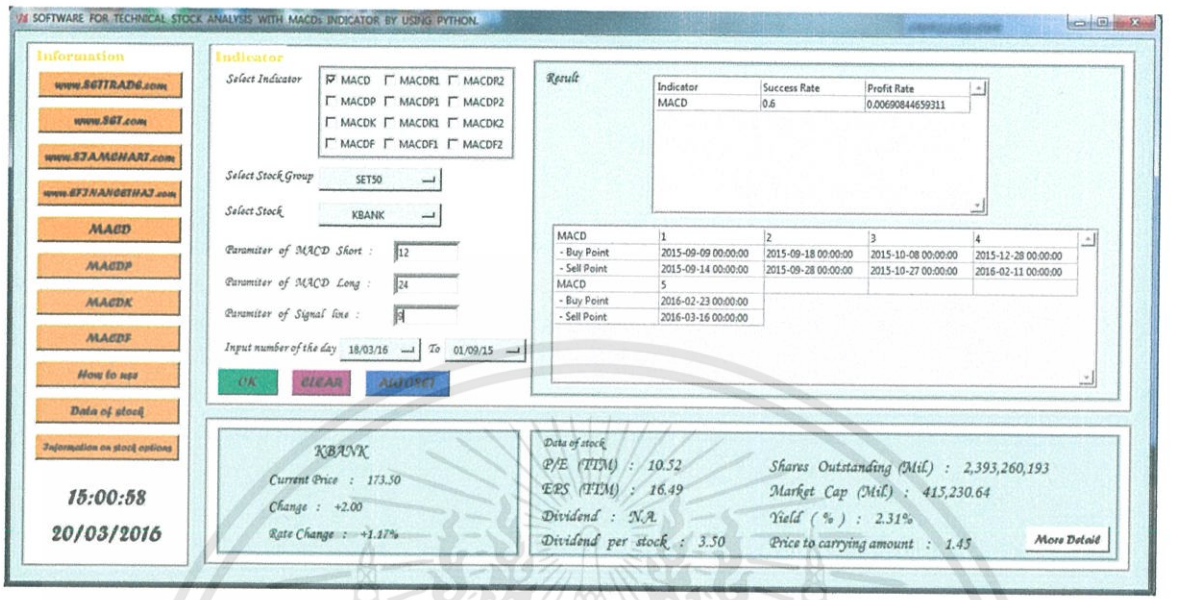
รูปที่ 4.4 แสดง หน้าต่างโปรแกรมส่วนที่ 3

1. กราฟส่วนนี้คือกราฟของราคาปิดของหุ้นที่เราเลือกวิเคราะห์
2. กราฟส่วนนี้คือกราฟของMACDและเส้นสัญญาณซึ่งจะมีกันตัดกันเกิดขึ้นทำให้มีจุดซื้อจุดขายซึ่งกราฟของแต่ละวิธีที่วิเคราะห์ก็จะเกิดจุดตัดไม่เหมือนกัน

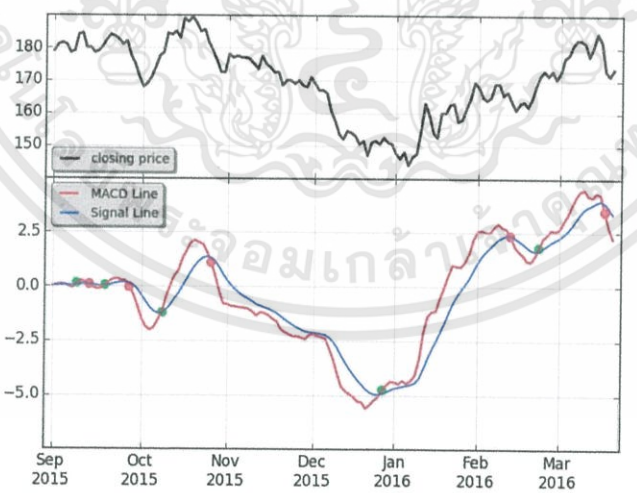
อภิปรายผล

เมื่อทำการทดสอบโปรแกรมในการวิเคราะห์แบบ MACDs MACDR1 MACDR2 MACDP MACDP1 MACDP2 MACDK MACDK1 MACDK2 MACDF MACDF1 MACDF2 โดยใช้หุ้น KBANK เป็นตัวอย่างในการวิเคราะห์และกำหนดวันสั้นเท่ากับ 12 วันยาวเท่ากับ 24 เส้นสัญญาณกับ 9 เริ่มคิดจากวันที่ 1 ก.ย. 2558 ถึงวันที่ 18 มี.ค. 2559 จะได้ผลลัพธ์ดังนี้

รูปที่ 4.5 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDs

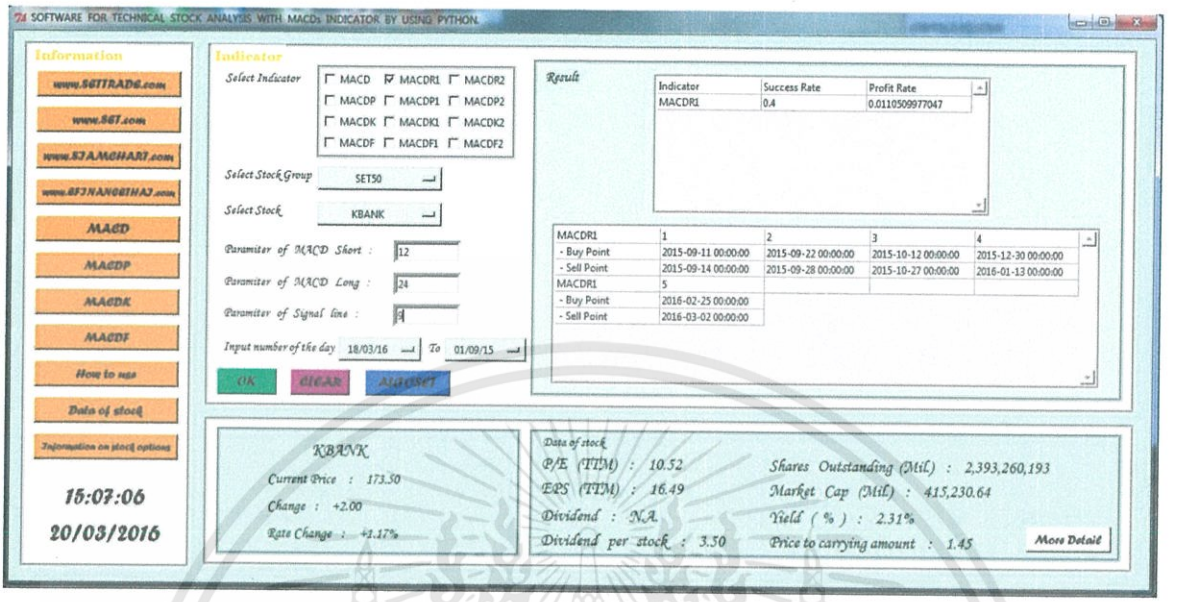


รูปที่ 4.6 กราฟของหุ้น KBANK เมื่อใช้ MACDs

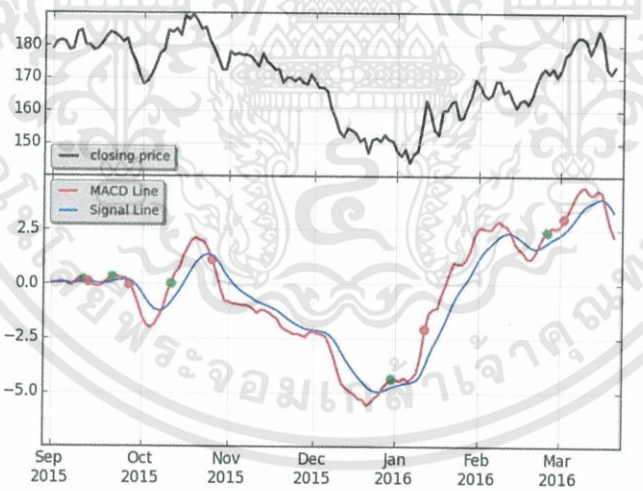


จากรูปที่ 4.5 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDs จะมีอัตราความสำเร็จอยู่ที่ 60% ผลกำไร 0.6% และมีคาบการซื้อขายทั้งหมด 5 คาบ

รูปที่ 4.7 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDR1

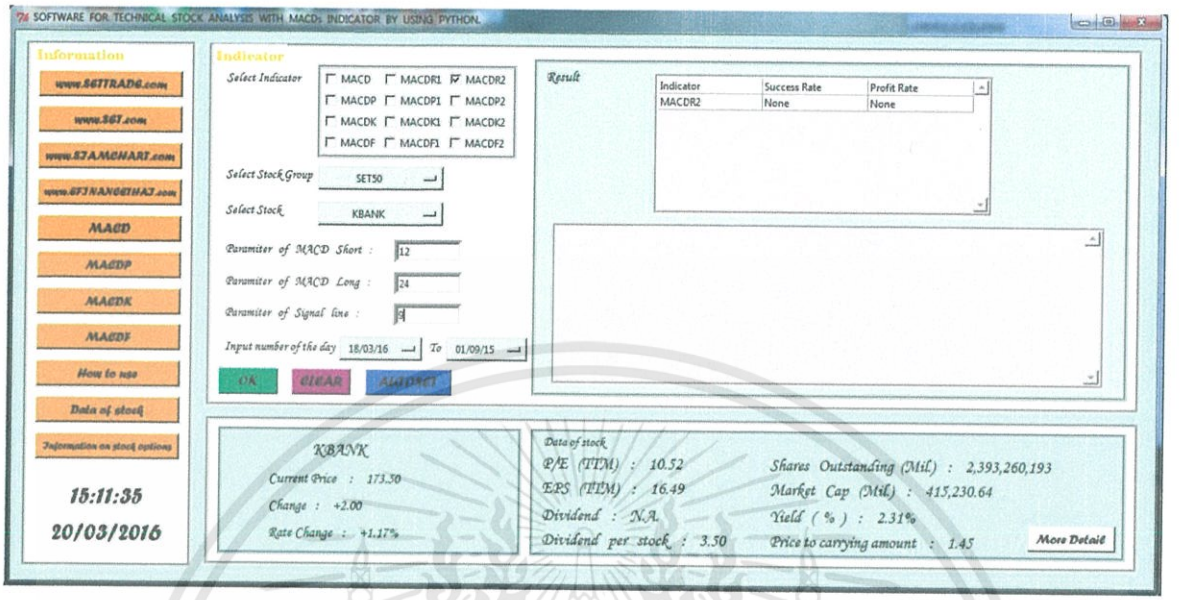


รูปที่ 4.8 กราฟของหุ้น KBANK เมื่อใช้ MACDR1

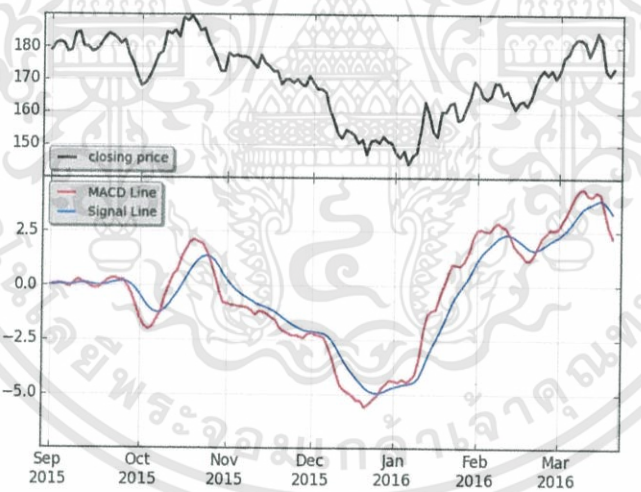


จากรูปที่ 4.7 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDR1 จะมีอัตราความสำเร็จอยู่ที่ 40% ผลกำไร 1.1% และมีคาบการซื้อขายทั้งหมด 6 คาบ

รูปที่ 4.9 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDR2

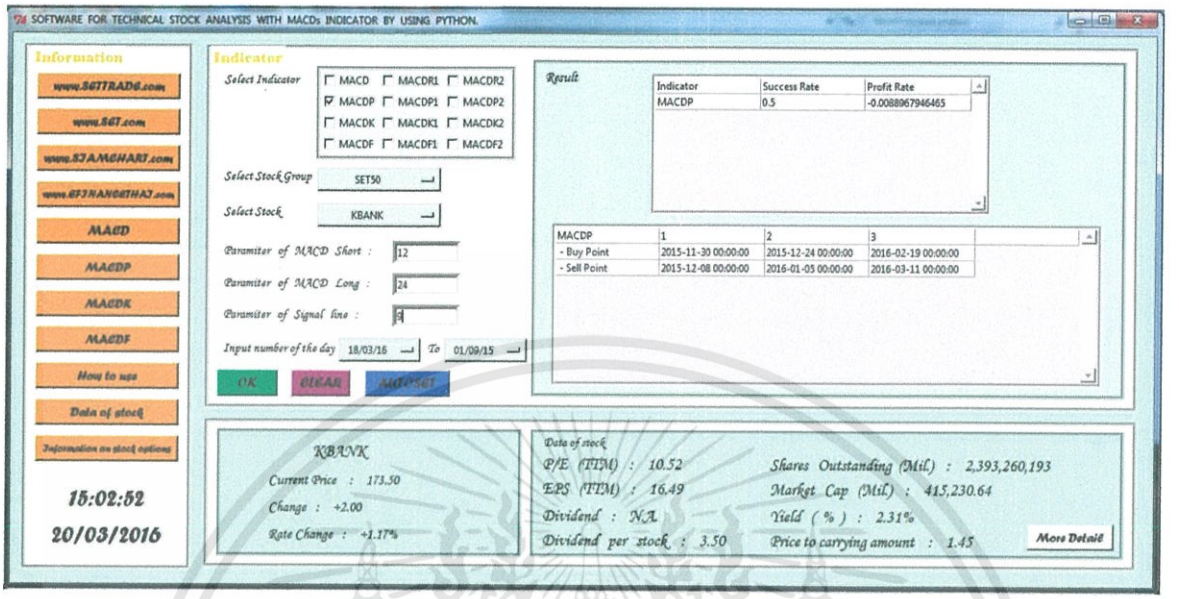


รูปที่ 4.10 กราฟของหุ้น KBANK เมื่อใช้ MACDR2

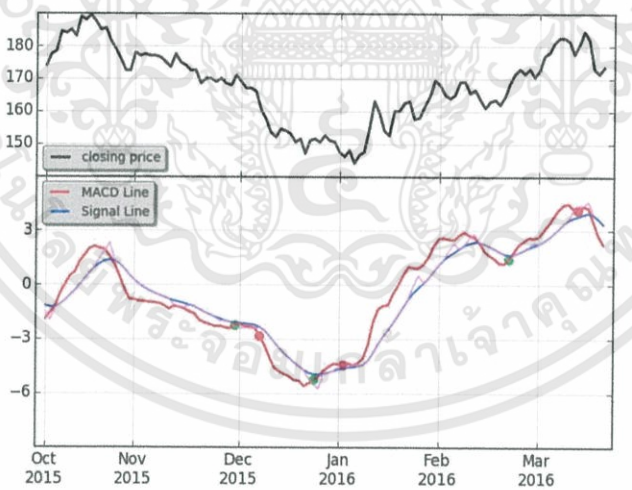


จากรูปที่ 4.9 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDR2 ไม่สามารถวิเคราะห์ได้เนื่องจากระยะเวลาที่ทำการวิเคราะห์สั้นเกินไป

รูปที่ 4.11 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDP

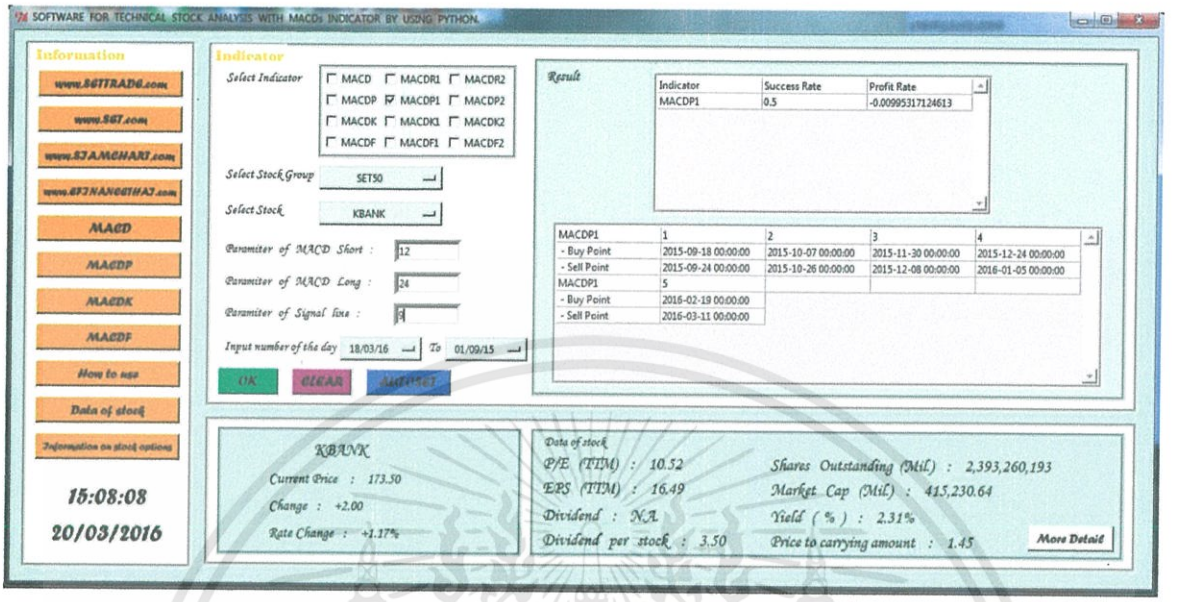


รูปที่ 4.12 กราฟของหุ้น KBANK เมื่อใช้ MACDP

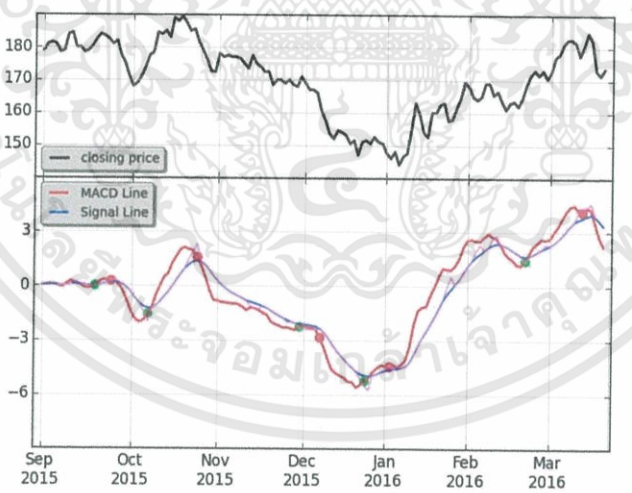


จากรูปที่ 4.11 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDP จะมีอัตราความสำเร็จอยู่ที่ 50% ผลกำไร -0.8% และมีคาบการซื้อขายทั้งหมด 3 คาบ

รูปที่ 4.13 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDP1

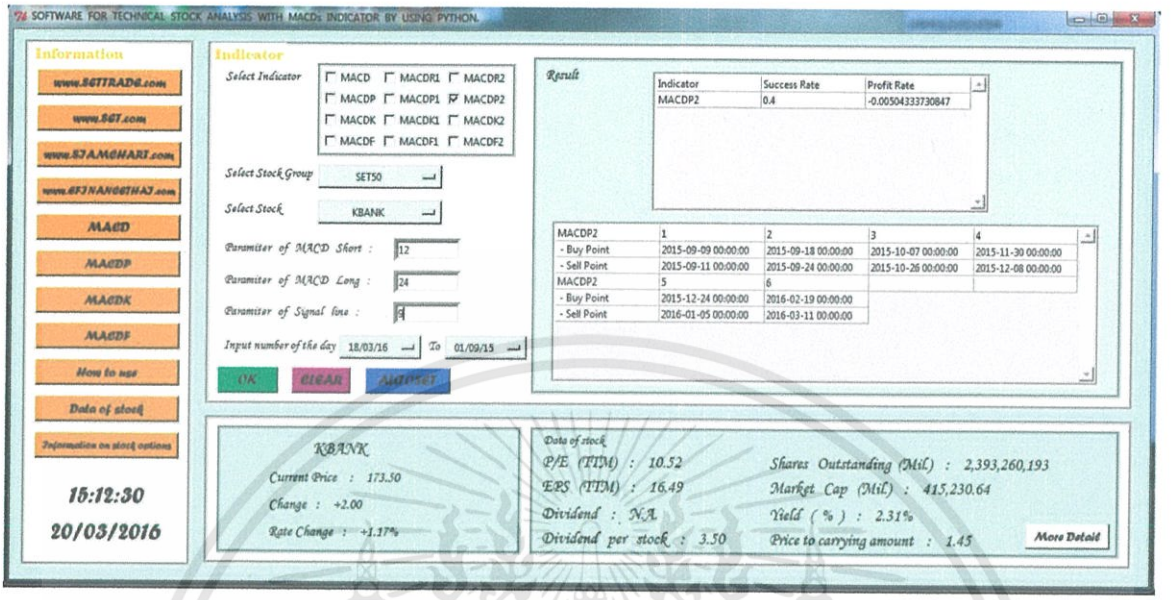


รูปที่ 4.14 กราฟของหุ้น KBANK เมื่อใช้ MACDP1

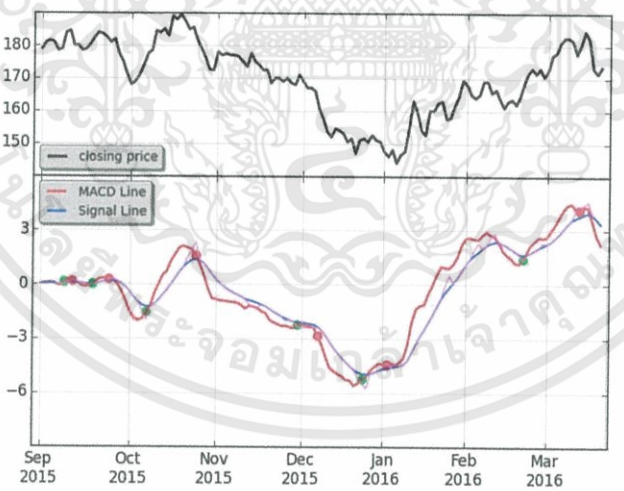


จากรูปที่ 4.13 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDP1 จะมีอัตราความสำเร็จอยู่ที่ 50% ผลกำไร -0.9% และมีคาบการซื้อขายทั้งหมด 5 คาบ

รูปที่ 4.15 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDP2

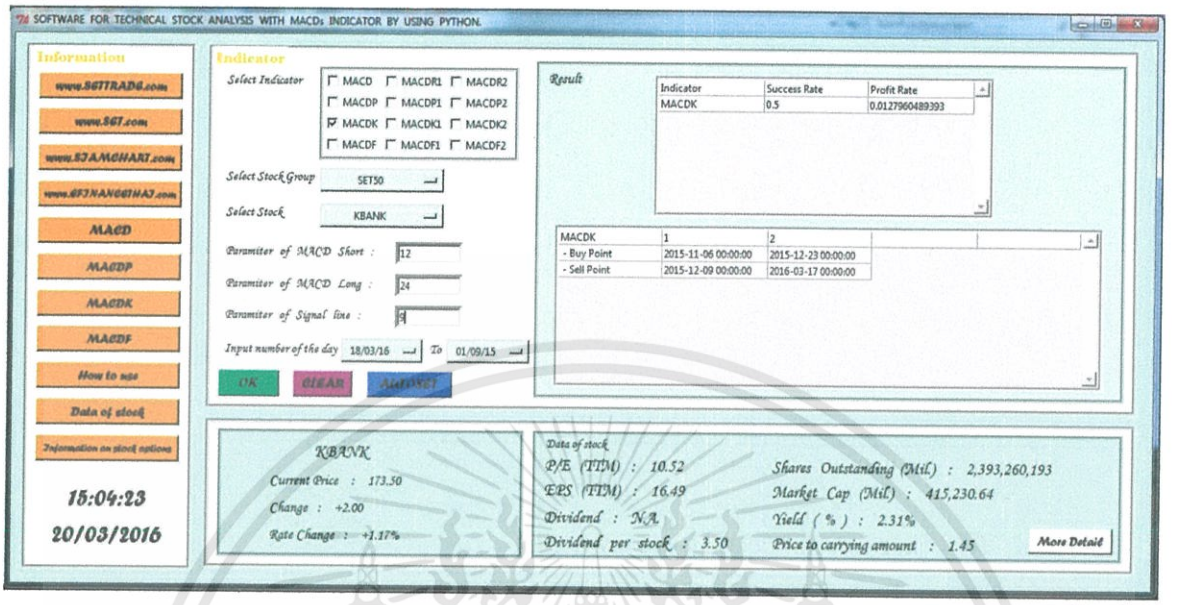


รูปที่ 4.16 กราฟของหุ้น KBANK เมื่อใช้ MACDP2

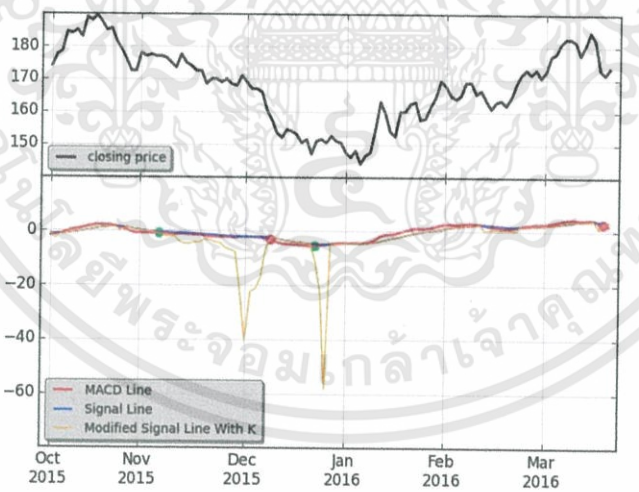


จากรูปที่ 4.15 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDP2 จะมีอัตราความสำเร็จอยู่ที่ 40% ผลกำไร -0.5% และมีคาบการซื้อขายทั้งหมด 6 คาบ

รูปที่ 4.17 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDK

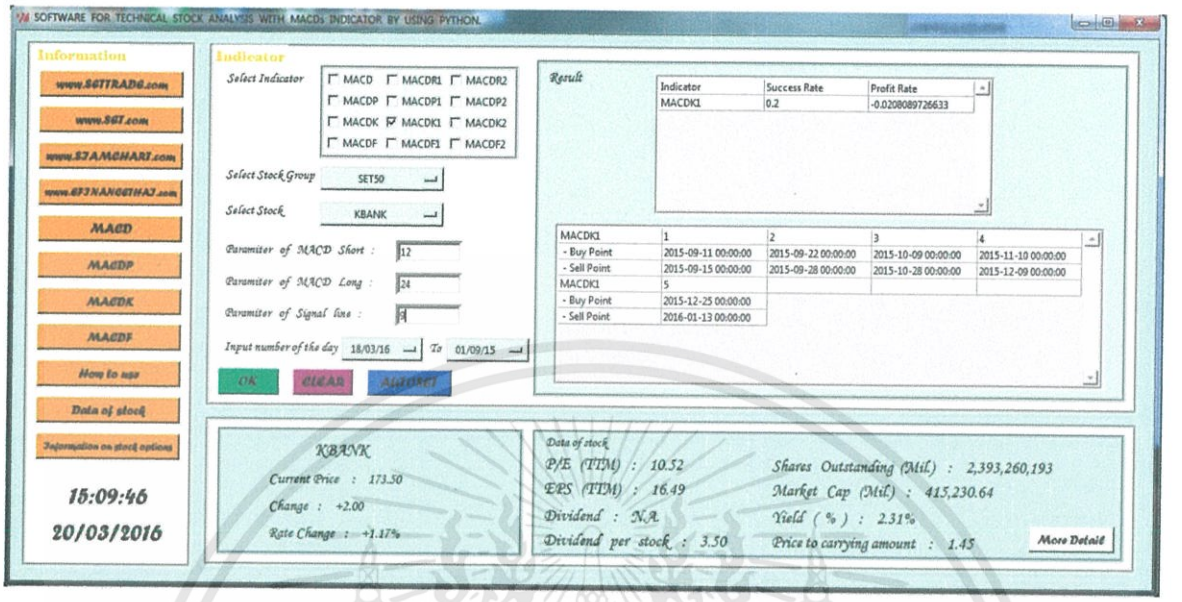


รูปที่ 4.18 กราฟของหุ้น KBANK เมื่อใช้ MACDK

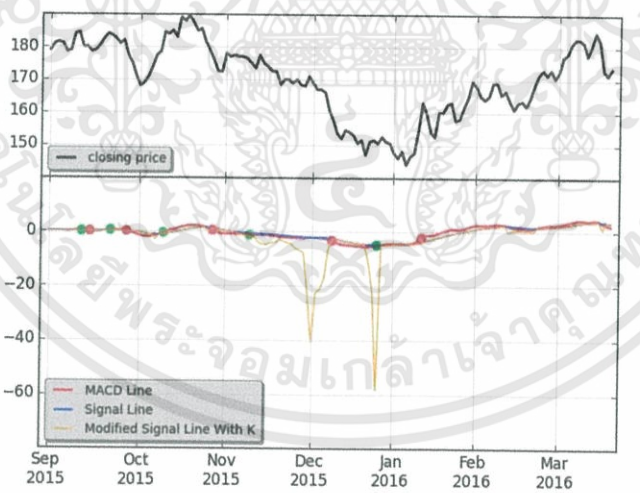


จากรูปที่ 4.17 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDK จะมีอัตราความสำเร็จอยู่ที่ 50% ผลกำไร 1.2% และมีคาบการซื้อขายทั้งหมด 2 คาบ

รูปที่ 4.19 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDK1

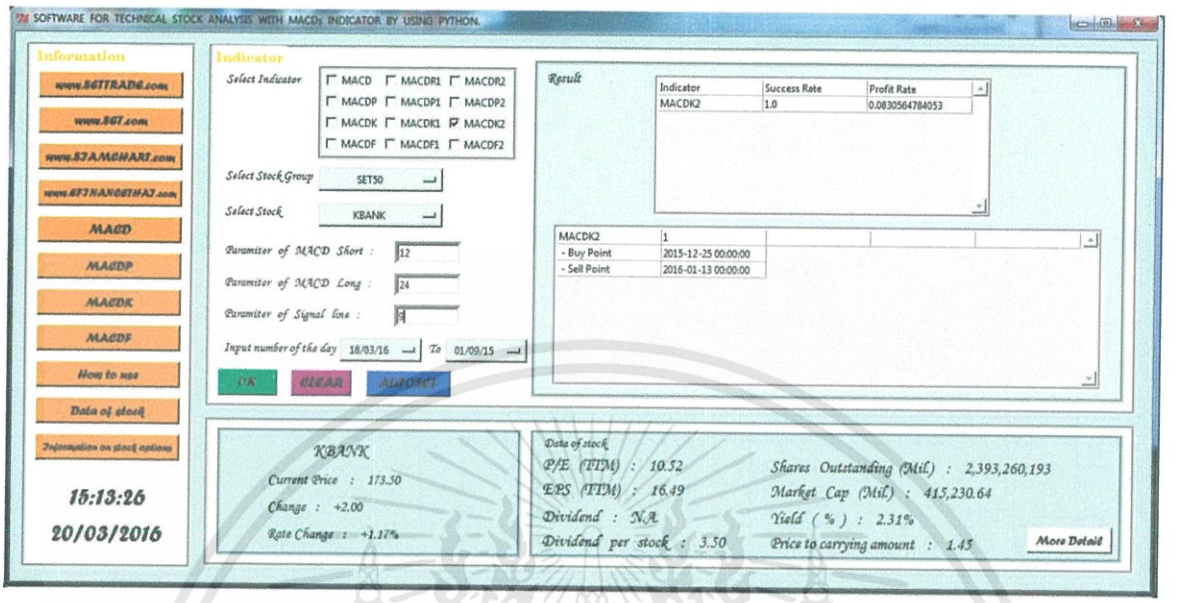


รูปที่ 4.20 กราฟของหุ้น KBANK เมื่อใช้ MACDK1

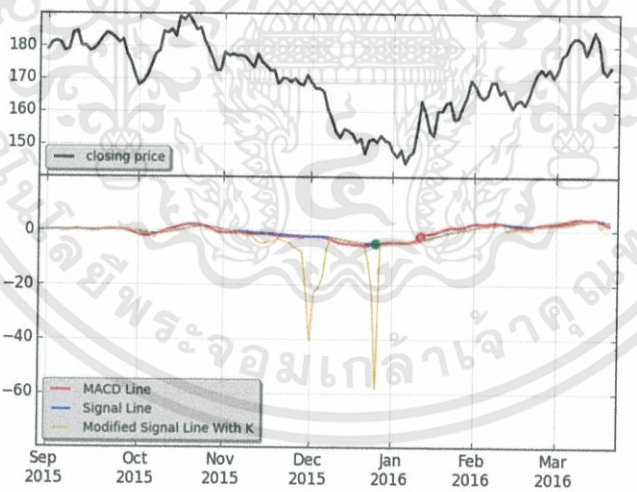


จากรูปที่ 4.19 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDK1 จะมีอัตราความสำเร็จอยู่ที่ 20% ผลกำไร -2% และมีคาบการซื้อขายทั้งหมด 5 คาบ

รูปที่ 4.21 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDK2

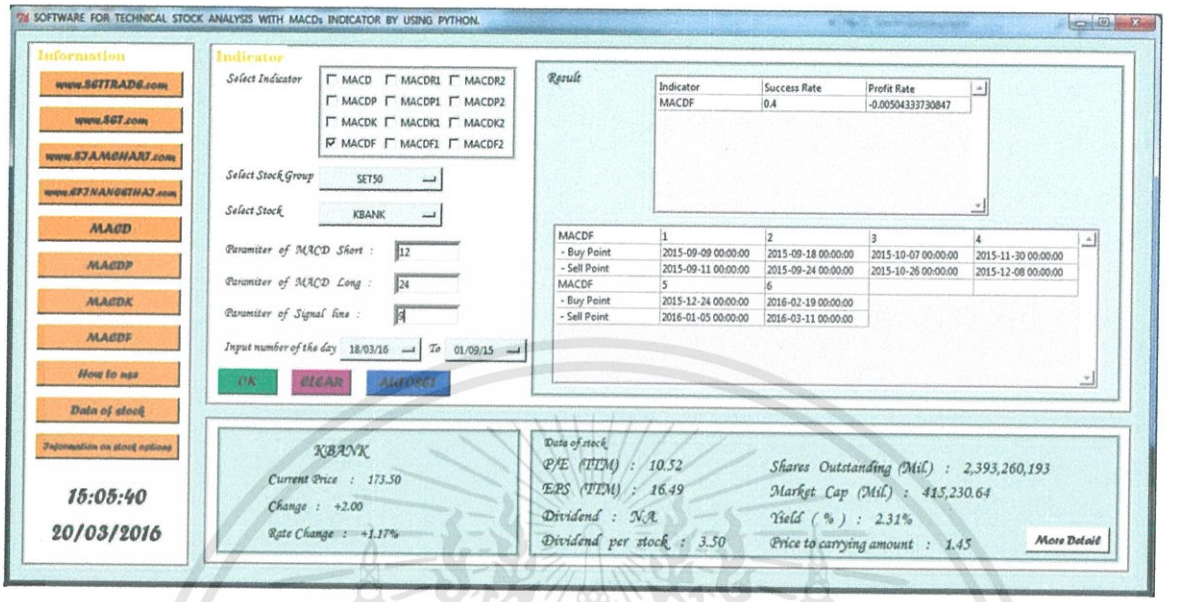


รูปที่ 4.22 กราฟของหุ้น KBANK เมื่อใช้ MACDK2

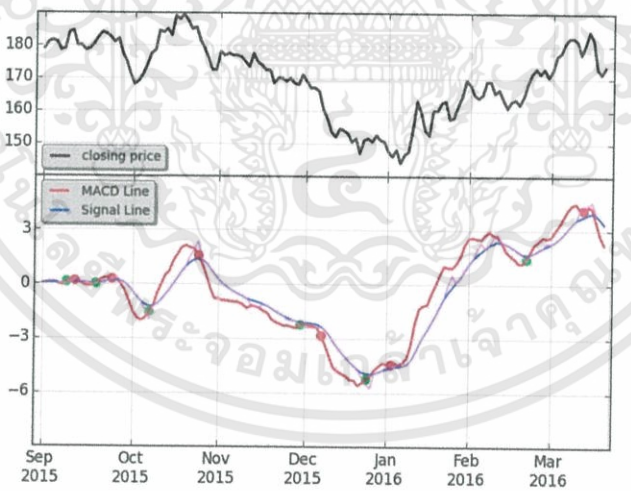


จากรูปที่ 4.21 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDK2 จะมีอัตราความสำเร็จอยู่ที่ 100% ผลกำไร 8.3% และมีคาบการซื้อขายทั้งหมด 1 คาบ

รูปที่ 4.23 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDF

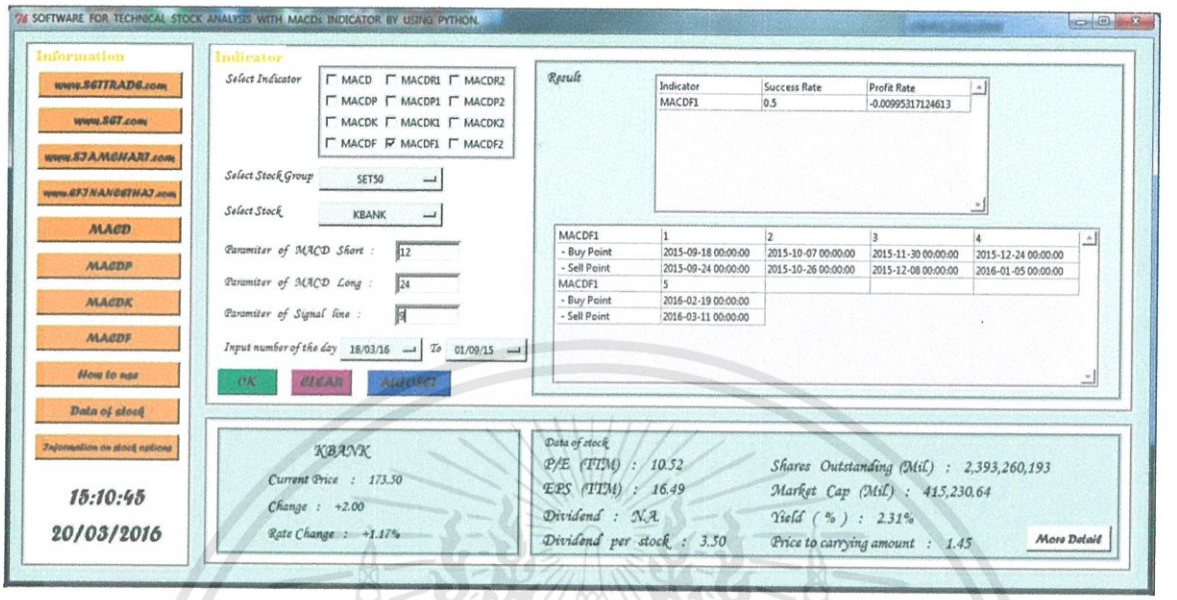


รูปที่ 4.24 กราฟของหุ้น KBANK เมื่อใช้ MACDF

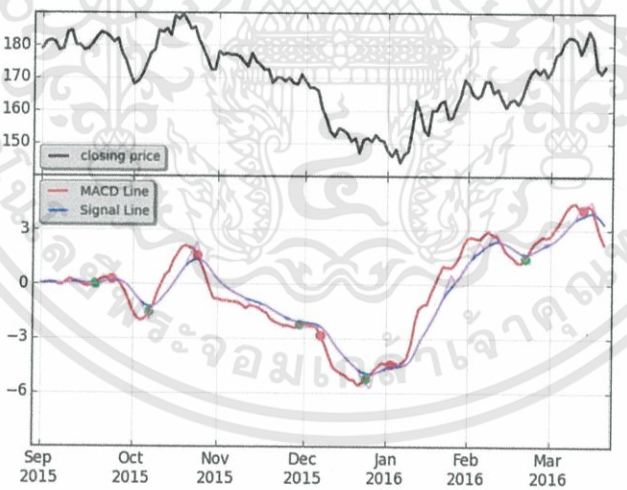


จากรูปที่ 4.23 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDF จะมีอัตราความสำเร็จอยู่ที่ 40% ผลกำไร 0.5% และมีคาบการซื้อขายทั้งหมด 6 คาบ

รูปที่ 4.25 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDF1

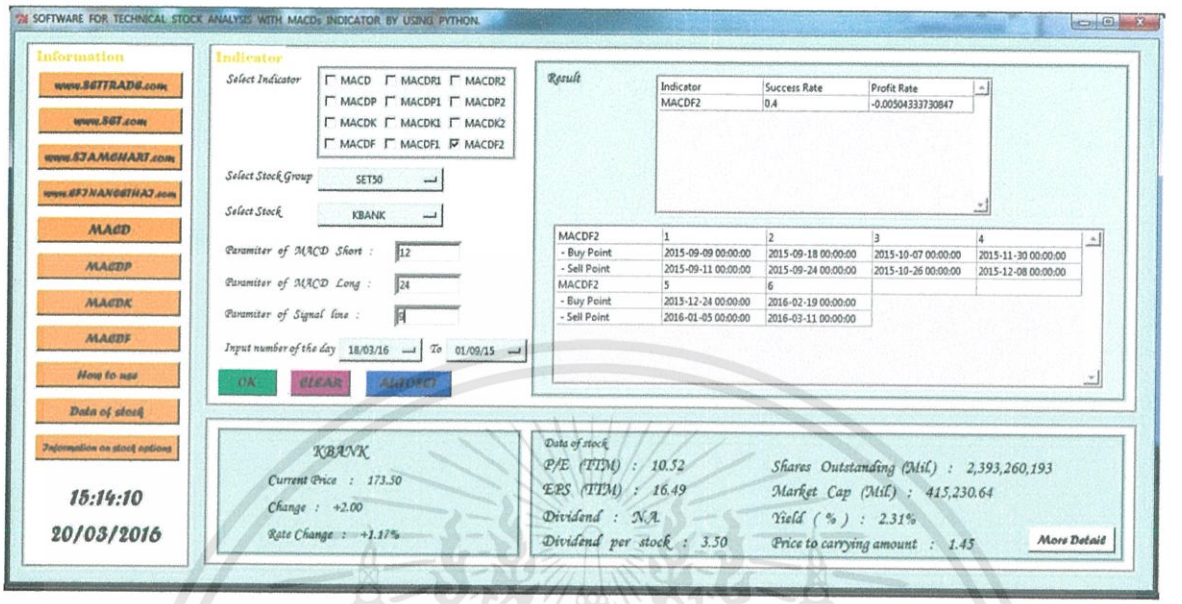


รูปที่ 4.26 กราฟของหุ้น KBANK เมื่อใช้ MACDF1

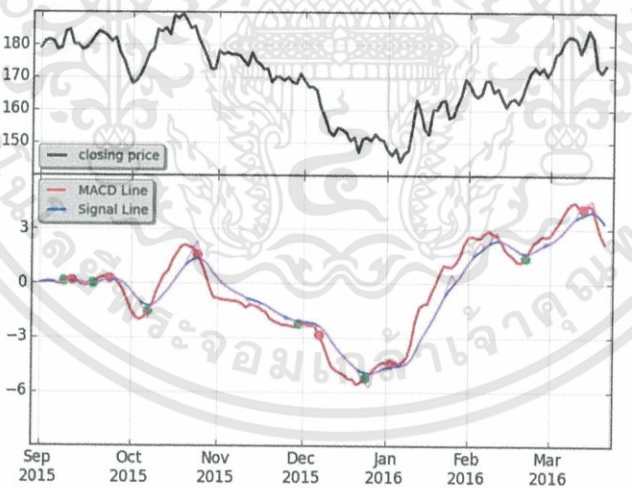


จากรูปที่ 4.25 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDF1 จะมีอัตราความสำเร็จอยู่ที่ 50% ผลกำไร 0.9% และมีคาบการซื้อขายทั้งหมด 5 คาบ

รูปที่ 4.27 ผลลัพธ์ของหุ้น KBANK เมื่อใช้ MACDF2



รูปที่ 4.28 กราฟของหุ้น KBANK เมื่อใช้ MACDF2



จากรูปที่ 4.27 จะเห็นว่าการวิเคราะห์หุ้น KBANK โดยใช้ MACDF2 จะมีอัตราความสำเร็จอยู่ที่ 40% ผลกำไร -0.5% และมีคาบการซื้อขายทั้งหมด 5 คาบ

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

การสร้างโปรแกรมสำเร็จรูปเพื่อที่จะรวบรวมวิธีวิเคราะห์แบบ MACDs MACDR1 MACDR2 MACDP MACDP1 MACDP2 MACDK MACDK1 MACDK2 MACDF MACDF1 MACDF2 จะทำให้วิเคราะห์ได้อย่างมีประสิทธิภาพเพราะโปรแกรมจะคำนวณจากข้อมูลเรียลไทม์ ซึ่งจะสามารถกลบข้อด้อยของปัญหาพิเศษของรุ่นก่อนๆ

จากปัญหาพิเศษนี้ สามารถนำโปรแกรมที่สร้างขึ้นมาประยุกต์ใช้ในการลงทุนรูปแบบต่างๆตามความเหมาะสมของผู้ลงทุน เช่น ในกรณีที่ผู้ลงทุนยังไม่มีความชำนาญในการวิเคราะห์สามารถใช้โปรแกรมเป็นตัวช่วยในการตัดสินใจลงทุน โดยเลือกการวิเคราะห์ในแบบต่างๆตามความต้องการ ซึ่งแต่ละแบบก็มีอัตราความสำเร็จที่แตกต่างกัน ซึ่งผู้ลงทุนสามารถทำการทดสอบกับข้อมูลราคาปิดของหุ้นได้อย่างเป็นปัจจุบัน

5.2 ข้อเสนอแนะ

ปัญหาพิเศษนี้สามารถนำไปศึกษาและพัฒนาต่อได้โดยอาจจะนำไปเขียนโปรแกรมใหม่โดยใช้ภาษาอื่นที่เป็นที่นิยม หรือปรับปรุงวิธีคิดหรือรูปแบบของโปรแกรมให้มีความสวยงามและใช้งานง่ายขึ้น เพื่อให้เกิดประสิทธิภาพมากขึ้น

เอกสารอ้างอิง

[1] นายพิพรรธน์ สำเภากิจ และคณะ. *การวิเคราะห์การซื้อขายหุ้นในตลาดหลักทรัพย์แห่งประเทศไทยโดยใช้เส้นสัญญาณดัดแปลงและการอนุมานค่าน้ำหนักการซื้อขายด้วยตรรกศาสตร์ฟัซซี* [หัวข้อปัญหาพิเศษสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง]. กรุงเทพฯ:สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง; 2556.

[2] นางสาวปุณนิมา บำรุงพงษ์ และคณะ. *FUZZY QUANTITATIVE ANALYSIS OF THE PROPERTY AND CONSTRUCTION INDUSTRIAL GROUP IN THE STOCK EXCHANGE OF THAILAND* [หัวข้อปัญหาพิเศษสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง]. กรุงเทพฯ:สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง; 2557.

[3] นายอานัส ตือบิงหมัะและคณะ. *DEVELOPING SIGNAL LINE IN MACD INDICATOR WITH EXCHANGE FUNCTION* [หัวข้อปัญหาพิเศษสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง]. กรุงเทพฯ:สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง; 2557.

[4] นายอนิรุธ หนูตอและคณะ. *FUZZY MACD WITH APPLICATION TO SET50 IN STOCK EXCHANG OF THAILAND* [หัวข้อปัญหาพิเศษสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง]. กรุงเทพฯ: สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง; 2557.

ภาคผนวก

ภาคผนวก ก.

รายชื่อหุ้นในกลุ่ม SET50 SET100 และ SETHD

SET50 คือ หุ้น 50 ตัวที่ตลาดหลักทรัพย์คัดเลือกมาเป็นหุ้นที่มีปัจจัยพื้นฐานดี มีอัตราการเติบโตของบริษัทอย่างต่อเนื่อง และมีปริมาณการซื้อขายอย่างสม่ำเสมอ ราคาไม่แกว่งมากเกินไป

ชื่อย่อ	ชื่อบริษัท
ADVANC	บริษัท แอดวานซ์ อินโฟร์ เซอร์วิส จำกัด (มหาชน)
AOT	บริษัท ท่าอากาศยานไทย จำกัด (มหาชน)
BA	บริษัท การบินกรุงเทพ จำกัด (มหาชน)
BANPU	บริษัท บ้านปู จำกัด (มหาชน)
BBL	ธนาคารกรุงเทพ จำกัด (มหาชน)
BCP	บริษัท บางจากปิโตรเลียม จำกัด (มหาชน)
BDMS	บริษัท กรุงเทพดุสิตเวชการ จำกัด(มหาชน)
BEC	บริษัท บีอีซี เวิลด์ จำกัด (มหาชน)
BH	บริษัท โรงพยาบาลบำรุงราษฎร์ จำกัด (มหาชน)
BMCL	บริษัท รถไฟฟ้ากรุงเทพ จำกัด (มหาชน)
BTS	บริษัท บีทีเอส กรุ๊ป โฮลดิ้งส์ จำกัด (มหาชน)
CBG	บริษัท คาราบาวกรุ๊ป จำกัด (มหาชน)
CENTEL	บริษัท โรงแรมเซ็นทรัลพลาซา จำกัด (มหาชน)
CK	บริษัท ช.การช่าง จำกัด (มหาชน)
CPALL	บริษัท ซีพี ออลล์ จำกัด (มหาชน)
CPF	บริษัท เจริญโภคภัณฑ์อาหาร จำกัด (มหาชน)
CPN	บริษัท เซ็นทรัลพัฒนา จำกัด (มหาชน)
DELTA	บริษัท เดลต้า อีเลคโทรนิคส์ (ประเทศไทย) จำกัด (มหาชน)
DTAC	บริษัท โทเทิ่ล แอ็คเซ็ส คอมมูนิเคชั่น จำกัด (มหาชน)
EGCO	บริษัท ผลิตไฟฟ้า จำกัด (มหาชน)
GLOW	บริษัท โกลว์ พลังงาน จำกัด (มหาชน)
HMPRO	บริษัท โฮม โปรดักส์ เซ็นเตอร์ จำกัด (มหาชน)
INTUCH	บริษัท อินทัช โฮลดิ้งส์ จำกัด (มหาชน)
IRPC	บริษัท ไออาร์พีซี จำกัด (มหาชน)
ITD	บริษัท อิตาเลียนไทย ดีเวล็อปเมนต์ จำกัด (มหาชน)
IVL	บริษัท อินโดรามา เวนเจอร์ส จำกัด (มหาชน)
JAS	บริษัท จัสมิน อินเตอร์เนชั่นแนล จำกัด (มหาชน)
KBANK	ธนาคารกสิกรไทย จำกัด (มหาชน)
KTB	ธนาคารกรุงไทย จำกัด (มหาชน)

ชื่อย่อ	ชื่อบริษัท
LH	บริษัท แลนด์แอนด์เฮ้าส์ จำกัด (มหาชน)
M	บริษัท เอ็มเค เรสโตรองต์ กรุ๊ป จำกัด (มหาชน)
MINT	บริษัท ไมเนอร์ อินเตอร์เนชั่นแนล จำกัด (มหาชน)
PS	บริษัท พฤกษา เรียลเอสเตท จำกัด (มหาชน)
PTT	บริษัท ปตท. จำกัด (มหาชน)
PTTEP	บริษัท ปตท. สำรวจและผลิตปิโตรเลียม จำกัด (มหาชน)
PTTGC	บริษัท พีทีที โกลบอล เคมิคอล จำกัด (มหาชน)
RATCH	บริษัท ผลิตไฟฟ้าราชบุรีโฮลดิ้ง จำกัด (มหาชน)
ROBINS	บริษัท ห้างสรรพสินค้าโรบินสัน จำกัด (มหาชน)
SAWAD	บริษัท ศรีสวัสดิ์ พาวเวอร์ 1979 จำกัด (มหาชน)
SCB	ธนาคารไทยพาณิชย์ จำกัด (มหาชน)
SCC	บริษัท ปูนซิเมนต์ไทย จำกัด(มหาชน)
TCAP	บริษัท ทุนธนชาต จำกัด (มหาชน)
THCOM	บริษัท ไทยคม จำกัด (มหาชน)
TMB	ธนาคารทหารไทย จำกัด (มหาชน)
TOP	บริษัท ไทยออยล์ จำกัด (มหาชน)
TPIPL	บริษัท ทีพีไอ โพลีน จำกัด (มหาชน)
TRUE	บริษัท ทรู คอร์ปอเรชั่น จำกัด (มหาชน)
TTW	บริษัท ทีทีดับบลิว จำกัด (มหาชน)
TU	บริษัท ไทยยูเนี่ยน กรุ๊ป จำกัด (มหาชน)
WHA	บริษัท ดับบลิวเอชเอ คอร์ปอเรชั่น จำกัด (มหาชน)

SET100 มีลักษณะทั่วไปเหมือนกับ SET50 แต่มีหุ้น 100 ตัวที่ตลาดหลักทรัพย์คัดเลือกมา โดยหุ้น 1 ตัวสามารถอยู่ได้ทั้งใน SET50 และ SET100

ชื่อย่อ	ชื่อบริษัท
AAV	บริษัท เอเชีย เอวิเอชั่น จำกัด (มหาชน)
ADVANC	บริษัท แอดวานซ์ อินโฟร์ เซอร์วิส จำกัด (มหาชน)
AMATA	บริษัท อมตะ คอร์ปอเรชัน จำกัด (มหาชน)
ANAN	บริษัท อนันดา ดีเวลลอปเม้นท์ จำกัด (มหาชน)
AOT	บริษัท ท่าอากาศยานไทย จำกัด (มหาชน)
AP	บริษัท เอพี (ไทยแลนด์) จำกัด (มหาชน)
ASP	บริษัท เอเชีย พลัส กรุ๊ป โฮลดิ้งส์ จำกัด (มหาชน)
BA	บริษัท การบินกรุงเทพ จำกัด (มหาชน)
BANPU	บริษัท บ้านปู จำกัด (มหาชน)
BBL	ธนาคารกรุงเทพ จำกัด (มหาชน)
BCP	บริษัท บางจากปิโตรเลียม จำกัด (มหาชน)
BDMS	บริษัท กรุงเทพดุสิตเวชการ จำกัด(มหาชน)
BEAUTY	บริษัท บิวตี้ คอมมูนิตี้ จำกัด (มหาชน)
BEC	บริษัท บีอีซี เวิลด์ จำกัด (มหาชน)
BECL	บริษัท ทางด่วนกรุงเทพ จำกัด (มหาชน)
BH	บริษัท โรงพยาบาลบารุงราษฎร์ จำกัด (มหาชน)
BJCHI	บริษัท บีเจซี เฮฟวี่ อินดัสทรี จำกัด (มหาชน)
BLAND	บริษัท บางกอกแลนด์ จำกัด (มหาชน)
BMCL	บริษัท รถไฟฟ้ากรุงเทพ จำกัด (มหาชน)
BTS	บริษัท บีทีเอส กรุ๊ป โฮลดิ้งส์ จำกัด (มหาชน)
CBG	บริษัท คาราบาวกรุ๊ป จำกัด (มหาชน)
CENTEL	บริษัท โรงแรมเซ็นทรัลพลาซา จำกัด (มหาชน)
CK	บริษัท ช.การช่าง จำกัด (มหาชน)
CKP	บริษัท ซีเค พาวเวอร์ จำกัด (มหาชน)
CPALL	บริษัท ซีพี ออลล์ จำกัด (มหาชน)
CPF	บริษัท เจริญโภคภัณฑ์อาหาร จำกัด (มหาชน)
CPN	บริษัท เซ็นทรัลพัฒนา จำกัด (มหาชน)
DELTA	บริษัท เดลต้า อีเลคโทรนิคส์ (ประเทศไทย) จำกัด (มหาชน)
DEMCO	บริษัท เด็มโก้ จำกัด (มหาชน)
DTAC	บริษัท โทเทิ่ล แอ็คเซ็ส คอมมูนิเคชั่น จำกัด (มหาชน)
EARTH	บริษัท เอ็นเนอร์ยี่ เอิร์ธ จำกัด (มหาชน)
EGCO	บริษัท ผลิตไฟฟ้า จำกัด (มหาชน)

ชื่อย่อ	ชื่อบริษัท
ERW	บริษัท ดิ เอราวัณ กรุ๊ป จำกัด (มหาชน)
GFPT	บริษัท จีเอฟพีที จำกัด (มหาชน)
GLOBAL	บริษัท สยามโกลบอลเฮ้าส์ จำกัด (มหาชน)
GLOW	บริษัท โกลว์ พลังงาน จำกัด (มหาชน)
GUNKUL	บริษัท กันกุลเอ็นจิเนียริ่ง จำกัด (มหาชน)
HANA	บริษัท ฮานา ไมโครอิเล็คโทรนิคส จำกัด (มหาชน)
HMPRO	บริษัท โฮม โปรดักส์ เซ็นเตอร์ จำกัด (มหาชน)
ICHI	บริษัท อิชิตัน กรุ๊ป จำกัด (มหาชน)
INTUCH	บริษัท อินทัช โฮลดิ้งส์ จำกัด (มหาชน)
IRPC	บริษัท ไออาร์พีซี จำกัด (มหาชน)
ITD	บริษัท อิตาเลียนไทย ดีเวล๊อปเมนต์ จำกัด (มหาชน)
IVL	บริษัท อินโดรามา เวนเจอร์ส จำกัด (มหาชน)
JAS	บริษัท จัสมิน อินเตอร์เนชั่นแนล จำกัด (มหาชน)
KBANK	ธนาคารกสิกรไทย จำกัด (มหาชน)
KCE	บริษัท เคซีอี อีเลคโทรนิคส์ จำกัด (มหาชน)
KKP	ธนาคารเกียรตินาคิน จำกัด (มหาชน)
KTB	ธนาคารกรุงไทย จำกัด (มหาชน)
KTC	บริษัท บัตรกรุงไทย จำกัด (มหาชน)
LH	บริษัท แลนด์แอนด์เฮ้าส์ จำกัด (มหาชน)
LHBANK	บริษัท แอล เอช ไฟแนนซ์เชียล กรุ๊ป จำกัด (มหาชน)
LOXLEY	บริษัท ล็อกซเล่ย์ จำกัด (มหาชน)
LPN	บริษัท แอล.พี.เอ็น.ดีเวลลอปเมนท์ จำกัด (มหาชน)
M	บริษัท เอ็มเค เรสโตรองต์ กรุ๊ป จำกัด (มหาชน)
MAJOR	บริษัท เมเจอร์ ซีนีเพล็กซ์ กรุ๊ป จำกัด (มหาชน)
MC	บริษัท แม็คกรุ๊ป จำกัด (มหาชน)
MINT	บริษัท ไมเนอร์ อินเตอร์เนชั่นแนล จำกัด (มหาชน)
MONO	บริษัท โมโน เทคโนโลยี จำกัด (มหาชน)
PS	บริษัท พฤกษา เรียลเอสเตท จำกัด (มหาชน)
PSL	บริษัท พรีเซียส ชิพปิ้ง จำกัด (มหาชน)
PTT	บริษัท ปตท. จำกัด (มหาชน)
PTTEP	บริษัท ปตท. สำรวจและผลิตปิโตรเลียม จำกัด (มหาชน)
PTTGC	บริษัท พีทีที โกลบอล เคมิคอล จำกัด (มหาชน)
QH	บริษัท ควอลิตี้เฮ้าส์ จำกัด (มหาชน)
RATCH	บริษัท ผลิตไฟฟ้าราชบุรีโฮลดิ้ง จำกัด (มหาชน)
ROBINS	บริษัท ห้างสรรพสินค้าโรบินสัน จำกัด (มหาชน)

ชื่อย่อ	ชื่อบริษัท
RS	บริษัท อาร์เอส จำกัด (มหาชน)
S	บริษัท สิงห์ เอสเตท จำกัด (มหาชน)
SAMART	บริษัท สามารถคอร์ปอเรชั่น จำกัด (มหาชน)
SAPPE	บริษัท เซ็ปเป้ จำกัด (มหาชน)
SAWAD	บริษัท ศรีสวัสดิ์ พาวเวอร์ 1979 จำกัด (มหาชน)
SCB	ธนาคารไทยพาณิชย์ จำกัด (มหาชน)
SCC	บริษัท ปูนซิเมนต์ไทย จำกัด(มหาชน)
SF	บริษัท สยามฟิวเจอร์ ดีเวลอปเมนท์ จำกัด (มหาชน)
SGP	บริษัท สยามแก๊ส แอนด์ ปิโตรเคมีคัลส์ จำกัด (มหาชน)
SIRI	บริษัท แสนสิริ จำกัด (มหาชน)
SPALI	บริษัท ศุภาลัย จำกัด (มหาชน)
SPCG	บริษัท เอสพีซีจี จำกัด (มหาชน)
STEC	บริษัท ซิโน-ไทย เอ็นจีเนียริ่งแอนด์คอนสตรัคชั่น จำกัด(มหาชน)
STPI	บริษัท เอสทีพี แอนด์ ไอ จำกัด (มหาชน
SVI	บริษัท เอสวีไอ จำกัด (มหาชน)
TCAP	บริษัท ทุนธนชาต จำกัด (มหาชน)
THAI	บริษัท การบินไทย จำกัด (มหาชน)
THCOM	บริษัท ไทยคม จำกัด (มหาชน)
TICON	บริษัท ไทคอน อินดัสเทรียล คอนเน็คชั่น จำกัด (มหาชน)
TISCO	บริษัท ทิสโก้ไฟแนนเชียลกรุ๊ป จำกัด (มหาชน)
TMB	ธนาคารทหารไทย จำกัด (มหาชน)
TOP	บริษัท ไทยออยล์ จำกัด (มหาชน)
TPIPL	บริษัท ทีพีไอ โพลีน จำกัด (มหาชน)
TRUE	บริษัท ทรู คอร์ปอเรชั่น จำกัด (มหาชน)
TTA	บริษัท โทรีเซนไทย เอเยนต์ซีส์ จำกัด (มหาชน)
TTCL	บริษัท ทีทีซีแอล จำกัด (มหาชน)
TTW	บริษัท ทีทีดับบลิว จำกัด (มหาชน)
TU	บริษัท ไทยยูเนี่ยน กรุ๊ป จำกัด (มหาชน)
U	บริษัท ยู ซิตี้ จำกัด (มหาชน)
UNIQ	บริษัท ยูนิค เอ็นจีเนียริ่ง แอนด์ คอนสตรัคชั่น จำกัด (มหาชน)
UV	บริษัท ยูนิ เวนเจอร์ จำกัด (มหาชน)
VGI	บริษัท วี จี ไอ โกลบอล มีเดีย จำกัด (มหาชน)
WHA	บริษัท ดับบลิวเอชเอ คอร์ปอเรชั่น จำกัด (มหาชน)

SETHD หรือ SET High Dividend หมายถึงหุ้นที่มีอัตราผลตอบแทนย้อนหลัง 3 ปีอยู่ในเกณฑ์ดีและสม่ำเสมอ หุ้นในกลุ่ม SETHD มีจำนวนทั้งสิ้น 30 ตัว

ชื่อย่อ	ชื่อบริษัท
AMATA	บริษัท อมตะ คอร์ปอเรชัน จำกัด (มหาชน)
AP	บริษัท เอพี (ไทยแลนด์) จำกัด (มหาชน)
ASP	บริษัท เอเซีย พลัส กรุ๊ป โฮลดิ้งส์ จำกัด (มหาชน)
BBL	ธนาคารกรุงเทพ จำกัด (มหาชน)
BECL	บริษัท บางจากปิโตรเลียม จำกัด (มหาชน)
CPF	บริษัท เซ็นทรัลพัฒนา จำกัด (มหาชน)
DELTA	บริษัท เดลต้า อีเลคโทรนิคส์ (ประเทศไทย) จำกัด (มหาชน)
EGCO	บริษัท ผลิตไฟฟ้า จำกัด (มหาชน)
GLOW	บริษัท โกลว์ พลังงาน จำกัด (มหาชน)
HANA	บริษัท ฮานา ไมโครอิเล็คโทรนิคส จำกัด (มหาชน)
JAS	บริษัท จัสมิน อินเตอร์เนชั่นแนล จำกัด (มหาชน)
KKP	ธนาคารเกียรตินาคิน จำกัด (มหาชน)
KTB	ธนาคารกรุงไทย จำกัด (มหาชน)
LH	บริษัท แลนด์แอนด์เฮ้าส์ จำกัด (มหาชน)
LPN	บริษัท แอล.พี.เอ็น.ดีเวลลอปเมนท์ จำกัด (มหาชน)
PS	บริษัท พฤกษา เรียลเอสเตท จำกัด (มหาชน)
PTT	บริษัท ปตท. จำกัด (มหาชน)
PTTEP	บริษัท ปตท. สำรวจและผลิตปิโตรเลียม จำกัด (มหาชน)
PTTGC	บริษัท พีทีที โกลบอล เคมิคอล จำกัด (มหาชน)
QH	บริษัท ควอลิตี้เฮ้าส์ จำกัด (มหาชน)
RATCH	บริษัท ผลิตไฟฟ้าราชบุรีโฮลดิ้ง จำกัด (มหาชน)
SAMART	บริษัท สามารถคอร์ปอเรชั่น จำกัด (มหาชน)
SCB	ธนาคารไทยพาณิชย์ จำกัด (มหาชน)
SCC	บริษัท ปูนซิเมนต์ไทย จำกัด(มหาชน)
SIRI	บริษัท แสนสิริ จำกัด (มหาชน)
SPALI	บริษัท ศุภาลัย จำกัด (มหาชน)
TCAP	บริษัท ทุนธนชาต จำกัด (มหาชน)
TICON	บริษัท ไทคอน อินดัสเทรียล คอนเน็คชั่น จำกัด (มหาชน)
TISCO	บริษัท ทิสโก้ไฟแนนเชียลกรุ๊ป จำกัด (มหาชน)
TU	บริษัท ไทยยูเนี่ยน กรุ๊ป จำกัด (มหาชน)

ภาคผนวก ข.

ไฟล์รหัส

โปรแกรมสำเร็จรูปในปัญหาพิเศษนี้ทำโดยใช้ซอฟท์แวร์ภาษาไพธอน 2.7.11 ซึ่งเป็นซอฟต์แวร์เปิดภายใต้ลิขสิทธิ์ GNU General Public License ซึ่งให้อิสระในการนาไปใช้อย่างกว้างขวาง สามารถดาวน์โหลดโดยไม่เสียค่าใช้จ่าย

ในภาคผนวก ข จะแสดงถึงรหัสต้นฉบับของโปรแกรมทั้งหมดที่ใช้ในการทดสอบการซื้อขายหุ้นด้วยวิธีการต่างๆ ซึ่งจากรหัสต้นฉบับดังกล่าวนี้สามารถนาไปแก้ไขดัดแปลงเพื่อวิเคราะห์เพิ่มเติมได้

ในรหัสต้นฉบับจะแบ่งการทำงานของโปรแกรมออกเป็น 14 ไฟล์ แต่ละไฟล์จะทำหน้าที่ที่แตกต่างกันออกไป โดยแต่ละไฟล์มีจุดประสงค์การทำงานดังแสดงในตารางที่ ข.1

ตารางที่ ข.1 ตารางอธิบายจุดประสงค์ในการทำงานของไฟล์ในแต่ละโฟล์เดอร์

ชื่อไฟล์ในโฟลเดอร์หลัก	จุดประสงค์ในการทำงาน
Table.py	ใช้สำหรับการแสดงหน้าต่างของโปรแกรมที่ได้ทำการสร้างขึ้นมาและรับค่า Input เพื่อส่งค่าต่อไปที่ Mainmacdf, Mainmacdp, Mainmacdk เพื่อทำการคำนวณในแต่ละวิธีเพื่อหาอัตราความสำเร็จของการขายและอัตรากำไรที่ได้รับเพื่อมาแสดงผลบนหน้าต่างของโปรแกรม
Setup.py	ใช้สำหรับการสร้างไฟล์ .exe เพื่อช่วยให้ง่ายต่อการเรียกใช้โปรแกรมและง่ายต่อการ setup ไฟล์โปรแกรม
Updatepriceandpricehigh.py	ใช้สำหรับการการอ่าน data base จากเว็ปไซต์ เพื่อที่จะนำข้อมูลราคาเปิด , ราคาปิด , ราคาสูงสุดและราคาต่ำสุดที่ได้มาทำการ update ไฟล์ .csv โดยจะมีคำสั่งในการสร้างไฟล์ .csv เป็นหลักในการ update และ การสร้างไฟล์ขึ้นมาใหม่เพื่อที่จะได้มีข้อมูลที่เป็น real time เสมอ
Lenday.py	ใช้สำหรับเปิดไฟล์ .csv หลักของหุ้นที่ได้ทำการเลือกจากหน้าต่างของโปรแกรมโดยจะเข้าไปที่ไฟล์หลักของหุ้นที่ได้ทำการเลือกและจะอ่านค่าตามจำนวนวันที่ได้ทำการเลือกมาจากหน้าต่างโปรแกรมและนำไปเขียนเป็นไฟล์ .csv ใหม่ที่มีขนาดที่เล็กกว่าเพื่อที่ Mainmacdf, Mainmacdp หรือ Mainmacdkสามารถเลือกใช้ทั้งหมดได้เลยเพื่อขจัดความล่าช้าของโปรแกรม
Main_update.py	ใช้สำหรับเปิดเว็บไซต์เพื่ออ่าน data base หากลุ่มของหุ้น

	และรายชื่อหุ้นมาเพื่อที่โปรแกรมในส่วนของ Updatepriceandpricehigh.py จะได้ทำการ update ตามกลุ่มและรายชื่อของหุ้นที่ได้รับมา
Mainmacdf	ใช้สำหรับทำการคำนวณเพื่อหาผลลัพธ์ตาม indicator จากข้อมูลที่ได้ทำการส่งมาจากหน้าต่างของโปรแกรมและเพื่อ plot graph โดยค่าที่ได้จะถูกส่งกลับไปที่หน้าต่างของโปรแกรมเพื่อแสดงผล
Mainmacdk	ใช้สำหรับทำการคำนวณเพื่อหาผลลัพธ์ตาม indicator จากข้อมูลที่ได้ทำการส่งมาจากหน้าต่างของโปรแกรมและเพื่อ plot graph โดยค่าที่ได้จะถูกส่งกลับไปที่หน้าต่างของโปรแกรมเพื่อแสดงผล
Mainmacdp	ใช้สำหรับทำการคำนวณเพื่อหาผลลัพธ์ตาม indicator จากข้อมูลที่ได้ทำการส่งมาจากหน้าต่างของโปรแกรมและเพื่อ plot graph โดยค่าที่ได้จะถูกส่งกลับไปที่หน้าต่างของโปรแกรมเพื่อแสดงผล
ชื่อไฟล์ในโฟลเดอร์macdtesterK	**จุดประสงค์ในการทำงาน**
data.py	ใช้สำหรับจัดการเกี่ยวกับการแปลงข้อมูลราคาหุ้นที่จัดเก็บในไฟล์อักษร (text file) ให้เป็นข้อมูลในรูปอนุกรมเวลา (time series) ที่พร้อมใช้งาน (เป็นไฟล์ .csv)
macd.py	ใช้สำหรับการคำนวณ MACD และ Signal Line
basic.py	ใช้สำหรับการวิเคราะห์ขั้นพื้นฐาน เช่นการแบ่งคาบการซื้อขาย การนิยามการซื้อขาย การหาอัตราความสำเร็จและอัตรากำไร
ema.py	ใช้สำหรับการคำนวณค่าเฉลี่ยเคลื่อนที่แบบเอ็กซ์โปเนนเชียล
modifiedsignalline.py	ใช้สาหรับการคำนวณเส้นสัญญาณดัดแปลง แบบต่างๆ ได้แก่ เส้นสัญญาณดัดแปลง MACDP เส้นสัญญาณดัดแปลง MACDK
plot.py	ใช้สำหรับสร้างกราฟ แสดงแนวโน้มราคาและจุดซื้อขาย
tradingtest_originalmacd.py	ใช้สำหรับการทดสอบวิธีการซื้อขายแบบดั้งเดิม (Original MACD)
tradingtest_macdr1.py	สำหรับการทดสอบวิธีการซื้อขาย MACDR1
tradingtest_macdr2.py	สำหรับการทดสอบวิธีการซื้อขาย MACDR2
tradingtest_modifiedsignalline.py	สำหรับการทดสอบวิธีการซื้อขายด้วยเส้นสัญญาณดัดแปลง (MACDP)
Tradingtest_modifiedsignallinewithK.py	สำหรับการทดสอบวิธีการซื้อขายหุ้นด้วยเส้นสัญญาณดัดแปลง (MACDK)

Tradingtest_modifiedsignallinewithK_1.py	สำหรับทดสอบวิธีการซื้อขายหุ้นด้วยเส้นสัญญาณดัดแปลง (MACDK1)
Tradingtest_modifiedsignallinewithK_2.py	ใช้สำหรับทดสอบวิธีการซื้อขายหุ้นด้วยเส้นสัญญาณดัดแปลง(MACDK2)
Mainmacdk.py	เป็นไฟล์หลักในการทำงาน ใช้สำหรับเรียกใช้ฟังก์ชันต่างๆ ในไฟล์อื่นๆ เพื่อวิเคราะห์ผลและแสดงผลลัพธ์
ชื่อไฟล์ในโฟลเดอร์macdtesterF	**จุดประสงค์ในการทำงาน**
basic.py	เป็นที่รวมฟังก์ชันสำหรับการวิเคราะห์เบื้องต้น เช่น การแบ่งคาบการซื้อขาย การหาจุดสูงสุดและจุดต่ำสุดในคาบ การนิยามการซื้อขายและการคำนวณอัตรากำไรเป็นต้น สำหรับ MACD ดั้งเดิม
Fuzzy.py	เป็นที่รวมฟังก์ชันสาหรับการวิเคราะห์เบื้องต้น เช่น การแบ่งคาบการซื้อขาย การหาจุดสูงสุดและจุดต่ำสุดในคาบ การนิยามการซื้อขายและการคำนวณอัตรากำไรเป็นต้น สำหรับแมคดีฟัซซี
data.py	สำหรับจัดการเกี่ยวกับการแปลงข้อมูลราคาหุ้นที่จัดเก็บในไฟล์อักษร (text file) ให้เป็นข้อมูลในรูปอนุกรมเวลา (time series) ที่พร้อมใช้
ema.py	สำหรับการคำนวณค่าเฉลี่ยเคลื่อนที่แบบเอ็กซ์โปเนนเชียล
macd.py	สำหรับการคำนวณ MACD และเส้นสัญญาณ
plot.py	สำหรับพล็อตกราฟ MACD ดั้งเดิม
modifiedsignalline.py	ใช้สาหรับการคำนวณเส้นสัญญาณดัดแปลง แบบต่างๆ ได้แก่ เส้นสัญญาณดัดแปลง MACDP เส้นสัญญาณดัดแปลง MACDK
tradingtest_originalmacd.py	สำหรับการทดสอบวิธีการซื้อขาย MACD แบบดั้งเดิม
tradingtest_macdr1.py	สำหรับการทดสอบวิธีการซื้อขาย MACDR1
tradingtest_macdr2.py	สำหรับการทดสอบวิธีการซื้อขาย MACDR2
tradingtest_modifiedsignalline.py	สำหรับการทดสอบวิธีการซื้อขายด้วยเส้นสัญญาณดัดแปลง
tradingtest_modsigtradingweight.py	สำหรับการทดสอบวิธีการซื้อขาย MACDF1 และ MACDF2
mainmacdf.py	เป็นไฟล์หลักสำหรับเรียกใช้ฟังก์ชันต่างๆ ในไฟล์อื่น และแสดงผลลัพธ์ ผู้ใช้สามารถทำการทดสอบซ้ำหรือดัดแปลงเพื่อวิเคราะห์เพิ่มเติมได้ที่ไฟล์นี้
ชื่อไฟล์ในโฟลเดอร์macdtesterP	**จุดประสงค์ในการทำงาน**
basic.py	ใช้สำหรับจัดการเกี่ยวกับการแปลงข้อมูลราคาหุ้นที่จัดเก็บในไฟล์อักษร (text file) ให้เป็นข้อมูลในรูปอนุกรมเวลา (time series) ที่พร้อมใช้งาน (เป็นไฟล์ .csv)

Mainmacdp.py	เป็นไฟล์หลักสำหรับเรียกใช้ฟังก์ชันต่างๆ ในไฟล์อื่น และแสดงผลลัพธ์ ผู้ใช้สามารถทำการทดสอบซ้ำหรือดัดแปลงเพื่อวิเคราะห์เพิ่มเติมได้ที่ไฟล์นี้
data.py	ใช้สำหรับการวิเคราะห์ขั้นพื้นฐาน เช่นการแบ่งคาบการซื้อขาย การนิยามการซื้อขาย การหาอัตราความสำเร็จและอัตรากำไร
ema.py	ใช้สำหรับการคำนวณค่าเฉลี่ยเคลื่อนที่แบบเอ็กซ์โปเนนเชียล
Fuzzy.py	เป็นที่รวมฟังก์ชันสาหรับการวิเคราะห์เบื้องต้น เช่น การแบ่งคาบการซื้อขาย การหาจุดสูงสุดและจุดต่ำสุดในคาบ การนิยามการซื้อขายและการคำนวณอัตรากำไรเป็นต้น สำหรับแมคดีฟัซซี
macd.py	ใช้สำหรับการคำนวณ MACD และ Signal Line
plot.py	ใช้สำหรับสร้างกราฟ แสดงแนวโน้มราคาและจุดซื้อขาย
tradingtest_originalmacd.py	สำหรับการทดสอบวิธีการซื้อขาย MACD แบบดั้งเดิม
tradingtest_macdr1.py	สำหรับการทดสอบวิธีการซื้อขาย MACDR1
tradingtest_macdr2.py	สำหรับการทดสอบวิธีการซื้อขาย MACDR2
Modifiedsignalline.py	ใช้สาหรับการคำนวณเส้นสัญญาณดัดแปลง แบบต่างๆ ได้แก่ เส้นสัญญาณดัดแปลง MACDP
tradingtest_modifiedsignalline.py	สำหรับการทดสอบวิธีการซื้อขายด้วยเส้นสัญญาณดัดแปลง (MACDP)
tradingtest_modsigtradingweight.py	สำหรับการทดสอบวิธีการซื้อขาย MACDP1 และ MACDP2

ตารางที่ ข.2 ตารางแสดงรหัสต้นฉบับในไฟล์ Table.py

```
from Tkinter import *
import Tkinter as tk
import time
import os
import urllib
import csv
```

```
from mainmacdk import *
from mainmacdf import *
from mainmacdp import *
from lenday import *
import webbrowser
from HTMLParser import HTMLParser
from main_update import *
from tkMessageBox import *

#runupdate()

root = Tk()
root.title('SOFTWARE FOR TECHNICAL STOCK ANALYSIS WITH MACDs INDICATOR BY
USING PYTHON.')
f = Frame(root, width=1350, height=650,bg='#d4f4f2')

#tableDecoration
T1 = Frame(f, relief=GROOVE, borderwidth=4,width=200, height=620, bg='white')
T1.place(relx=0.01, rely=0.02, anchor=NW, width=200, height=620)
Label(T1, text='Information', fg="#ffd700", bg='white', font=('Elephant', 13)).place(relx=0.02,
rely=0.02,anchor=W)

######################## BUTTONPDF
#############################################
def callpdf1(event):
    os.system("MACD.pdf")
def callpdf2(event):
    os.system("MACDP.pdf")
def callpdf3(event):
    os.system("MACDK.pdf")
def callpdf4(event):
    os.system("MACDF.pdf")
def callpdf5(event):
    os.system("How-to-use.pdf")
def callpdf6(event):
    os.system("Dataofstock.pdf")
def callpdf7(event):
    os.system("Information.pdf")
```

```
buttonfuzzy1 = Button(T1, text = " MACD ", background="orange", fg="black", font=('forte',
12)) #buttonmacd in fuzzy
buttonfuzzy1.place(relx=0.5, rely=0.33,anchor=N, width=170, height=30)
buttonfuzzy1.bind('<Button-1>', callpdf1)


buttonfuzzy2 = Button(T1, text = " MACDP ", background="orange", fg="black", font=('forte',
11)) #buttonmacdp in fuzzy
buttonfuzzy2.place(relx=0.5, rely=0.40,anchor=N, width=170, height=30)
buttonfuzzy2.bind('<Button-1>', callpdf2)

buttonfuzzy3 = Button(T1, text = " MACDK ", background="orange", fg="black", font=('forte',
11)) #buttonmacdk in fuzzy
buttonfuzzy3.place(relx=0.5, rely=0.47,anchor=N, width=170, height=30)
buttonfuzzy3.bind('<Button-1>', callpdf3)

buttonfuzzy4 = Button(T1, text = " MACDF ", background="orange", fg="black", font=('forte',
11)) #buttonmacdf in fuzzy
buttonfuzzy4.place(relx=0.5, rely=0.54,anchor=N, width=170, height=30)
buttonfuzzy4.bind('<Button-1>', callpdf4)

buttonfuzzy5 = Button(T1, text = " How to use ", background="orange", fg="black",
font=('forte', 11)) #buttonhow to use in fuzzy
buttonfuzzy5.place(relx=0.5, rely=0.61,anchor=N, width=170, height=30)
buttonfuzzy5.bind('<Button-1>', callpdf5)

buttonfuzzy6 = Button(T1, text = " Data of stock ", background="orange", fg="black",
font=('forte', 11)) #buttondata of stock in fuzzy
buttonfuzzy6.place(relx=0.5, rely=0.68,anchor=N, width=170, height=30)
buttonfuzzy6.bind('<Button-1>', callpdf6)

buttonfuzzy7 = Button(T1, text = "Information on stock options", background="orange",
fg="black", font=('forte', 9)) #buttoninformation in fuzzy
buttonfuzzy7.place(relx=0.5, rely=0.75,anchor=N, width=170, height=30)
buttonfuzzy7.bind('<Button-1>', callpdf7)
################################### BUTTONWEB
#############################################################
```

```
def openwebsettrade():
    url = "http://www.settrade.com/login.jsp?txtBrokerId=IPO"
    webbrowser.open(url)
def openwebset():
    url = "http://www.set.or.th/education/th/education.html"
    webbrowser.open(url)
def openwebsiamchart():
    url = "http://siamchart.com/"
    webbrowser.open(url)
def openwebefinancethai():
    url = "http://www.efinancethai.com/"
    webbrowser.open(url)
buttonsettrade = Button(T1, text = "www.SETTRADE.com", background="orange",
command=openwebsettrade, fg="black", font=('forte', 11)) #buttonwebsettrade in fuzzy
buttonsettrade.place(relx=0.5, rely=0.05,anchor=N, width=170, height=30)

buttonset = Button(T1, text = "www.SET.com", background="orange",
command=openwebset,  fg="black", font=('forte', 11)) #buttonwebset in fuzzy
buttonset.place(relx=0.5, rely=0.12,anchor=N, width=170, height=30)

buttonsiam = Button(T1, text = "www.SIAMCHART.com", background="orange",
command=openwebsiamchart,  fg="black", font=('forte', 11)) #buttonwebsiamchart in fuzzy
buttonsiam.place(relx=0.5, rely=0.19,anchor=N, width=170, height=30)

buttonefinancethai = Button(T1, text = "www.EFINANCETHAI.com", background="orange",
command=openwebefinancethai,  fg="black", font=('forte', 10)) #buttonwebfinance
buttonefinancethai.place(relx=0.5, rely=0.26,anchor=N, width=170, height=30)
##########################################################################
########################################

###TIME###
def update_timeText():
    # Get the current time, note you can change the format as you wish
    current = time.strftime("%H:%M:%S")
    # Update the timeText Label box with the current time
    timeText.configure(text=current)
    # Call the update_timeText() function after 1 second
    root.after(1000, update_timeText)
```

```
# Create a timeText Label (a text box)
timeText = Label(T1, text="",fg="black", bg='white', font=("forte", 20))
timeText.place(relx=0.5, rely=0.9,anchor=S)
update_timeText()

###DATE###
date = time.strftime("%d/%m/%Y")
text = Label(T1, text=date,fg="black", bg='white', font=("forte", 20))
text.place(relx=0.5, rely=0.97,anchor=S)

#####################################tablefuzzy#############################
##########################

T2 = Frame(f, relief=GROOVE, borderwidth=4,width=1100, height=175,bg='white')
T2.place(relx=0.17, rely=0.7, anchor=NW, width=1100, height=175)
#Label(f, text='Fuzzy', fg="blue", font=('Monotype Corsiva', 11)).place(relx=0.185,
rely=0.02,anchor=W)

tablefuzzy1 = Frame(T2, relief=GROOVE, borderwidth=4,width=700, height=150,bg='#d4f4f2')
#table1 in fuzzy
tablefuzzy1.place(relx=0.35, rely=0.05, anchor=NW, width=700, height=150)

tablefuzzy2 = Frame(T2, relief=GROOVE, borderwidth=4,width=700, height=150,bg='#d4f4f2')
tablefuzzy2.place(relx=0.01, rely=0.05, anchor=NW, width=360, height=150)

#Label(tablefuzzy1, text='Data of stock', fg="black",bg='#d4f4f2', font=('Monotype Corsiva',
11)).place(relx=0.125, rely=0.07,anchor=E)

#########################################tableindicator###################
##############################

T3 = Frame(f, relief=GROOVE, borderwidth=4,width=1100, height=350, bg='white')
T3.place(relx=0.17, rely=0.68, anchor=SW, width=1100, height=428)
Label(T3, text='Indicator', fg="#ffd700", bg='white', font=('Elephant', 13)).place(relx=0,
rely=0.025,anchor=W)


labelindicator1 = Label(T3, text=" Select Indicator  ", fg="black", bg='white', font=('Monotype
```

```
Corsiva', 11)) #label1 in indicatior
labelindicator1.place(relx=0.01, rely=0.06,anchor=NW)

###################### Checkbutton
########################################################
class Dummy: pass
var = Dummy()
CK = Frame(T3, relief=GROOVE, borderwidth=4,bg='#f3fbfa')
CK.place(relx=0.115, rely=0.05, anchor=NW, width=235, height=110)
var1 = IntVar()
Checkbutton (CK,text = 'MACD', variable=var1,bg='#f3fbfa').grid(row = 0, column = 0, sticky =
W)
var2 = IntVar()
Checkbutton (CK,text = 'MACDR1', variable=var2,bg='#f3fbfa').grid(row = 0, column = 1, sticky
= W)
var3 = IntVar()
Checkbutton (CK,text = 'MACDR2', variable=var3,bg='#f3fbfa').grid(row = 0, column = 2, sticky
= W)
var4 = IntVar()
Checkbutton (CK,text = 'MACDP', variable=var4,bg='#f3fbfa').grid(row = 1, column = 0, sticky =
W)
var5 = IntVar()
Checkbutton (CK,text = 'MACDP1', variable=var5,bg='#f3fbfa').grid(row = 1, column = 1, sticky
= W)
var6 = IntVar()
Checkbutton (CK,text = 'MACDP2', variable=var6,bg='#f3fbfa').grid(row = 1, column = 2, sticky
= W)
var7 = IntVar()
Checkbutton (CK,text = 'MACDK', variable=var7,bg='#f3fbfa').grid(row = 2, column = 0, sticky =
W)
var8 = IntVar()
Checkbutton (CK,text = 'MACDK1', variable=var8,bg='#f3fbfa').grid(row = 2, column = 1, sticky
= W)
var9 = IntVar()
Checkbutton (CK,text = 'MACDK2', variable=var9,bg='#f3fbfa').grid(row = 2, column = 2, sticky
= W)
var10 = IntVar()
Checkbutton (CK,text = 'MACDF', variable=var10,bg='#f3fbfa').grid(row = 3, column = 0, sticky
```

```
= W)
var11 = IntVar()
Checkbutton (CK,text = 'MACDF1', variable=var11,bg='#f3fbfa').grid(row = 3, column = 1,
sticky = W)
var12 = IntVar()
Checkbutton (CK,text = 'MACDF2', variable=var12,bg='#f3fbfa').grid(row = 3, column = 2,
sticky = W)

#########################################################################
##################

labelindicator2 = Label(T3, text=" Select Stock Group  ", fg="black", bg='white',
font=('Monotype Corsiva', 11)) #label2 in indicator
labelindicator2.place(relx=0.01, rely=0.33,anchor=NW)

labelindicator3 = Label(T3, text=" Select Stock   ", fg="black", bg='white', font=('Monotype
Corsiva', 11)) #label3 in indicator
labelindicator3.place(relx=0.01, rely=0.43,anchor=NW)
varindicator3 = StringVar() #option menu3 in indicator
optindicator3 = OptionMenu(T3, varindicator3, '')
varindicator3.set(' Select Stock ')
optindicator3.config(bg = '#e7f9f7')
optindicator3.place(relx=0.115, rely=0.43,anchor=NW, width=150)

varindicator4 = StringVar() #option menu4 in indicator
optindicator4 = OptionMenu(T3, varindicator4, '')
optindicator4.config(bg = '#e7f9f7')

varindicator5 = StringVar() #option menu5 in indicator
optindicator5 = OptionMenu(T3, varindicator5,'')
optindicator5.config(bg = '#e7f9f7')

############################### input value web
#########################################################

def stock(event):
    G = varindicator2.get()
```

```
    url =
"http://www.settrade.com/C04_01_stock_quote_p1.jsp?txtSymbol="+varindicator3.get()+"&ss
oPageId=9&selectPage=1"
    htmlfile = urllib.urlopen(url)
    htmltext = htmlfile.read()

    regex = '<div class="col-xs-12 colorGreen"><h1>(.+?)</h1></div>'
    pattern = re.compile(regex)
    cp = re.findall(pattern,htmltext)#CurrentPrice
    if cp==[]:
        regex = '<div class="col-xs-12 colorRed"><h1>(.+?)</h1></div>'
        pattern = re.compile(regex)
        cp = re.findall(pattern,htmltext) #CurrentPrice
        if cp==[]:
            regex = '<div class="col-xs-12 "><h1>(.+?)</h1></div>'
            pattern = re.compile(regex)
            cp = re.findall(pattern,htmltext) #CurrentPrice

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    pe = re.findall(pattern,htmltext) #PE

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    eps = re.findall(pattern,htmltext) #eps

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    div = re.findall(pattern,htmltext)#div

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    divperstock = re.findall(pattern,htmltext)#div

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    yie = re.findall(pattern,htmltext)#yie
```

```
    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    mc = re.findall(pattern,htmltext)#mc

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    so = re.findall(pattern,htmltext)#so

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    pac = re.findall(pattern,htmltext)#pac
################################## Change and Change rate
########################################
    global chk_starttag
    chk_starttag = False

    global change_value
    change_value = []

    class MyHTMLParser(HTMLParser):

        def handle_starttag(self, tag, attrs):
            if attrs != [] and tag == "h1" and attrs[0][1] == "colorRed":
                global chk_starttag
                chk_starttag = True
            if attrs != [] and tag == "h1" and attrs[0][1] == "colorGreen":
                global chk_starttag
                chk_starttag = True

        def handle_data(self, data):
            if(chk_starttag):
                change_value.append(data)
            global chk_starttag
            chk_starttag = False
    parser = MyHTMLParser()
    parser.feed(htmltext)

    if(change_value == []):
```

```
        change_value.append(0)
        change_value.append(0)
        change_value.append(0)

    change = str(change_value[0]).replace(" ","").replace("\r","").replace("\n","")
    rate_change = str(change_value[1]).replace(" ","").replace("\r","").replace("\n","")
##################################### Label data of stock
#############################################################

    Label(tablefuzzy1, text='Data of stock', fg="black",bg='#d4f4f2', font=('Monotype Corsiva',
11)).place(relx=0.125, rely=0.07,anchor=E)

    buttonfuzzydetail = Button(tablefuzzy1, text = "More Detail", background='white',
command = openweb, fg="black", font=('forte', 11)) #buttonmoredetail
    buttonfuzzydetail.place(relx=0.845, rely=0.95,anchor=SW, width=100, height=30)

    labelindicator5 = Label(tablefuzzy2, text = varindicator3.get()+"                ",
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label5 in indicator
    labelindicator5.place(relx=0.3, rely=0.05,anchor=NW)

    labelindicator6 = Label(tablefuzzy2, text = "Change   :"+"    "+change+"                ",
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 13))#label5 in indicator
    labelindicator6.place(relx=0.15, rely=0.525,anchor=NW)

    labelindicator7 = Label(tablefuzzy2, text = "Rate Change   :"+"    "+rate_change+"
", fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 13))#label5 in indicator
    labelindicator7.place(relx=0.15, rely=0.75,anchor=NW)

    labelindicator18 = Label(tablefuzzy2, text = "Current Price    :"+"    "+cp[0]+"          ",
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 13))#label5 in indicator
    labelindicator18.place(relx=0.15, rely=0.3,anchor=NW)

    labelindicator9 = Label(tablefuzzy1, text = "P/E   (TTM)    :"+"    "+pe[0],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator9.place(relx=0.01, rely=0.15,anchor=NW)

    labelindicator10 = Label(tablefuzzy1, text = "EPS   (TTM)    :"+"    "+eps[2],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
```

```
    labelindicator10.place(relx=0.01, rely=0.35,anchor=NW)

    labelindicator11 = Label(tablefuzzy1, text = "Dividend    :"+"    "+div[5],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator11.place(relx=0.01, rely=0.58,anchor=NW)

    labelindicator15 = Label(tablefuzzy1, text = "Dividend  per  stock   :"+"
"+divperstock[3], fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator15.place(relx=0.01, rely=0.78,anchor=NW)

    labelindicator12 = Label(tablefuzzy1, text = "Shares  Outstanding  (Mil.)   :"+"    "+so[7],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator12.place(relx=0.4, rely=0.15,anchor=NW)

    labelindicator13 = Label(tablefuzzy1, text = "Market  Cap  (Mil.)   :"+"    "+mc[6],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator13.place(relx=0.4, rely=0.35,anchor=NW)

    labelindicator14 = Label(tablefuzzy1, text = "Yield  (  %  )   :"+"    "+yie[1],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator14.place(relx=0.4, rely=0.58,anchor=NW)

    labelindicator16 = Label(tablefuzzy1, text = "Price to carrying amount    :"+"    "+pac[4],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator16.place(relx=0.4, rely=0.78,anchor=NW)


##########################################################################
##############################################

    day = []
    text_file = open("quoteName"+G+"/"+varindicator3.get()+".csv", "r")
    try:
        reader = csv.reader(text_file)
        for row in reader:
            if "data" in row:
                continue
            day.append(row[0])
```

```
    finally:
        text_file.close()



    optindicator4 = OptionMenu(T3, varindicator4, *day)
    varindicator4.set('  ')
    optindicator4.config(bg = '#e7f9f7')
    optindicator4.place(relx=0.14, rely=0.81,anchor=NW, width=100) #option menu4 in
indicator


    optindicator5 = OptionMenu(T3, varindicator5, *day)
    varindicator5.set('  ')
    optindicator5.config(bg = '#e7f9f7')
    optindicator5.place(relx=0.255, rely=0.81,anchor=NW, width=100) #option menu5 in
indicator

def groupstock(event):
    G = varindicator2.get()
    text_file = open("quoteName"+G+".txt", "r")
    Glist = text_file.readlines()
    Glist = map(lambda s: s.strip(), Glist)
    text_file.close()
    optindicator3 = OptionMenu(T3, varindicator3, *Glist, command = stock)
    optindicator3.config(bg = '#e7f9f7')
    varindicator3.set(' Select Stock ')
    optindicator3.place(relx=0.115, rely=0.43,anchor=NW, width=150)



group = ['SET50', 'SET100', 'SETHD','AGRO','--AGRI',
                '--FOOD','CONSUMP','--FASHION','--HOME','--PERSON','FINCIAL',
                '--BANK','--FIN','--INSUR','INDUS','--AUTO','--IMM','--PAPER',
                '--PETRO','--PKG','--STEEL','PROPCON','--CONMAT','--PROP','--PF&REIT',
                '--CONS','RESOURC','--ENERG','--MINE','SERVICE','--COMM','--HELTH',
                '--MEDIA','--PROF','--TOURISM','--TRANS','TECH','--ETRON','--ICT']
varindicator2 = StringVar() #option menu2 in indicator
optindicator2 = OptionMenu(T3 ,varindicator2, *group, command = groupstock)
varindicator2.set(' Select Group Stock ')
```

```
optindicator2.config(bg = '#e7f9f7')
optindicator2.place(relx=0.115, rely=0.33,anchor=NW, width=150)


labelindicator5 = Label(T3, text = '', fg="black", bg='white', font=('Monotype Corsiva',
11))#label5 in indicator
labelindicator5.place(relx=0.01, rely=0.415,anchor=NW)

labelindicator8 = Label(T3, text="", fg="black", bg='white', font=('Monotype Corsiva',
11))#label8 in indicator
labelindicator8.place(relx=0.01, rely=0.485,anchor=NW)


labelindicator4 = Label(T3, text=" Paramiter of MACD Short :", fg="black", bg='white',
font=('Monotype Corsiva', 11)) #label4 in indicator
labelindicator4.place(relx=0.01, rely=0.54,anchor=NW)

labelindicator4 = Label(T3, text=" Paramiter of MACD Long :", fg="black", bg='white',
font=('Monotype Corsiva', 11)) #label4 in indicator
labelindicator4.place(relx=0.01, rely=0.63,anchor=NW)

entryindicator1 = Entry(T3, bd =5, justify = CENTER) #entry1 in indicator
entryindicator1.place(relx=0.2, rely=0.54,anchor=NW,width=80)

entryindicator2 = Entry(T3, bd =5, justify = CENTER) #entry2 in indicator
entryindicator2.place(relx=0.2, rely=0.63,anchor=NW,width=80)

labelindicator6 = Label(T3, text=" Paramiter of Signal line :", fg="black", bg='white',
font=('Monotype Corsiva', 11)) #label6 in indicator
labelindicator6.place(relx=0.01, rely=0.72,anchor=NW)
entryindicator3 = Entry(T3, bd =5, justify = CENTER) #entry3 in indicator
entryindicator3.place(relx=0.2, rely=0.72,anchor=NW,width=80)

labelindicator7 = Label(T3, text=" Input number of the day ", fg="black", bg='white',
font=('Monotype Corsiva', 11)) #label7 in indicator
labelindicator7.place(relx=0.01, rely=0.815,anchor=NW)

varindicator4.set(' ')
```

```
optindicator4.place(relx=0.14, rely=0.81,anchor=NW, width=100) #option menu4 in indicator
optindicator4.config(bg = '#e7f9f7')

labelindicator11 = Label(T3, text=" To ", fg="black", bg='white', font=('Monotype Corsiva',
11))#label11 in indicator
labelindicator11.place(relx=0.232, rely=0.815,anchor=NW)

varindicator5.set(' ')
optindicator5.place(relx=0.255, rely=0.81,anchor=NW, width=100) #option menu5 in
indicator
optindicator5.config(bg = '#e7f9f7')

######################################## command for ok button
#########################################################################
####
def clearTable():
    tableindicator2 = Frame(T3, relief=GROOVE, borderwidth=4,width=700, height=85) #table1
in indicator
    tableindicator2.place(relx=0.48, rely=0.07, anchor=NW, width=400, height=165)

    scframe = VerticalScrolledFrame(tableindicator2)
    scframe.place( width=390, height=155)

    tableindicator3 = Frame(T3, relief=GROOVE, borderwidth=4,width=700, height=397)
#table2 in indicator
    tableindicator3.place(relx=0.372, rely=0.49, anchor=NW, width=650, height=190)

    scframe = VerticalScrolledFrame(tableindicator3)
    scframe.place( width=640, height=180)

def opt():
    #print varindicator1.get()
    Label(tableindicator4, text='Result', fg="black", bg='#e1f9f7', font=('Monotype Corsiva',
13)).place(relx=0.01, rely=0.04,anchor=W)
    clearTable()

    print varindicator2.get()
    print varindicator3.get()
```

```
    print varindicator4.get()
    print varindicator5.get()
    print entryindicator1.get()
    print entryindicator2.get()
    print entryindicator3.get()

    if int(entryindicator1.get())<int(entryindicator2.get()):

        successRate = []
        profitRate = []
        allBuyPoints = []
        allSellPoints = []

        checkCB = [""]
        countCB = 0

        if var1.get() == 1:

            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())

            suc , pro , buy , sell =
runmacd(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACD')
            countCB += 1;

        if var2.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdr1(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())
```

```
            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDR1')
            countCB += 1;

        if var3.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdr2(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDR2')
            countCB += 1;

        if var4.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdp(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDP')
            countCB += 1;

        if var5.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
```

```
varindicator5.get())
            suc , pro , buy , sell =
runmacdp1(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDP1')
            countCB += 1;

        if var6.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdf2(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDP2')
            countCB += 1;

        if var7.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdk(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDK')
```

```
            countCB += 1;

        if var8.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdk1(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDK1')
            countCB += 1;

        if var9.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdk2(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDK2')
            countCB += 1;

        if var10.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdf(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
```

```
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDF')
            countCB += 1;

        if var11.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdf1(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDF1')
            countCB += 1;

        if var12.get() == 1:
            length_date(varindicator2.get(), varindicator3.get(), varindicator4.get(),
varindicator5.get())
            suc , pro , buy , sell =
runmacdf2(varindicator3.get(),entryindicator1.get(),entryindicator2.get(),entryindicator3.get())

            successRate.append(suc)
            profitRate.append(pro)
            allBuyPoints.append(buy)
            allSellPoints.append(sell)

            checkCB.append('MACDF2')
            countCB += 1;

        print successRate
        print profitRate
        print allBuyPoints
        print allSellPoints
```

```
        ############################## result Success and Profit Rate
#####################################################

        tableindicator2 = Frame(T3, relief=GROOVE, borderwidth=4,width=700, height=85)
#table1 in indicator
        tableindicator2.place(relx=0.48, rely=0.07, anchor=NW, width=400, height=165)

        scframe = VerticalScrolledFrame(tableindicator2)
        scframe.place( width=390, height=155)

        headTable1 = ["Indicator", "Success Rate", "Profit Rate"]

        rows = []
        for i in range(countCB+1):
            cols = []
            for j in range(3):
                e = Entry(scframe.interior,relief=RIDGE)
                e.grid(row=i, column=j, sticky=NSEW)

                #Print Indicator Name
                if(j == 0 and i!=0):
                    e.insert(END, '%s' % checkCB[i])

                #Print Success Rate
                if(j == 1 and i!=0):
                    e.insert(END, '%s' % successRate[i-1])

                #Print Profit Rate
                if(j == 2 and i!=0):
                    e.insert(END, '%s' % profitRate[i-1])

                #Print Head
                if(i == 0):
                    e.insert(END, '%s' % headTable1[j])
                else:
                    if(j!=0 and j!=1 and j!=2):
                        e.insert(END, '%d.%d' % (i, j))
```

```
            cols.append(e)
        rows.append(cols)

    ############################## result table buy and sell point
#####################################################

    tableindicator3 = Frame(T3, relief=GROOVE, borderwidth=4,width=700, height=397)
#table2 in indicator
    tableindicator3.place(relx=0.372, rely=0.49, anchor=NW, width=650, height=190)

    scframe = VerticalScrolledFrame(tableindicator3)
    scframe.place( width=640, height=180)

    checkCB.remove('')
    #Fix None Value Point
    noneIndex = []
    for cnv in range(len(successRate)):
        if(successRate[cnv] == None):
            noneIndex.append(cnv)
    print "noneIndex : "+str(noneIndex)
    for rnv in noneIndex:
        checkCB[rnv] = ''
        successRate.remove(None)
        profitRate.remove(None)
        allBuyPoints.remove([])
        allSellPoints.remove([])
    print "checkCB : "+str(checkCB)
    for rnv in noneIndex:
        checkCB.remove('')


    print "-----Before----------"
    print checkCB
    print successRate
    print profitRate
    print allBuyPoints
    print allSellPoints
```

```
    tableNumber = []
    tableValue = []

    for ftn in range(len(allBuyPoints)):
        if(len(allBuyPoints[ftn]) == 0):
            continue
        tableValue.append(len(allBuyPoints[ftn]))

        if(len(allBuyPoints[ftn]) % 4 == 0):
            tableNumber.append(len(allBuyPoints[ftn]) / 4)
        else:
            tableNumber.append(len(allBuyPoints[ftn])/ 4 + 1)

    print tableNumber
    print tableValue

    colNum = 0
    for cn in range(len(tableNumber)):
        colNum = colNum + tableNumber[cn]
    colNum *= 3
    print colNum

    tableType = 0 #For Change Type
    colCount = 1
    valueBuyCount = 0
    valueSellCount = 0
    valueNumber = 1

    rows = []
    buyandsell = ["Buy Point", "Sell Point"]
    for i in range(colNum):
        cols = []

        print "tableType : "+str(tableType)+" Name : "+str(checkCB[tableType])

        if(valueBuyCount >= tableValue[tableType]):
            valueBuyCount = 0
        if(valueSellCount >= tableValue[tableType]):
```

```
        valueSellCount = 0
        if(colCount >= tableNumber[tableType]*3):
            tableType += 1
            colCount = 0
            valueNumber = 1

    for j in range(5):

        if(valueBuyCount >= tableValue[tableType]):
            continue
        if(valueSellCount >= tableValue[tableType]):
            continue

        e = Entry(scframe.interior,relief=RIDGE)
        e.grid(row=i, column=j, sticky=NSEW)

        if (i%3==1 and j == 0):
            e.insert(END, '%s' % " - "+buyandsell[0])

        if (i%3==2 and j == 0):
            e.insert(END, '%s' % " - "+buyandsell[1])

        if(j == 0 and i%3 == 0): #Print Type
            e.insert(END, '%s' % checkCB[tableType])
        else:
            #'''
            if(i%3 == 0):
                if(valueNumber > tableValue[tableType]):
                    continue
                e.insert(END, '%d' % (valueNumber))

                valueNumber += 1

            if(i!=0 and j!=0 and i%3!=0):

                if(i%3 == 1):
                    print str(i)+" "+str(j)
                    print "Data Buy "+str(tableType)+" "+str(valueBuyCount)
```

```
                    e.insert(END, '%s' % str(allBuyPoints[tableType][valueBuyCount]))
                    #e.insert(END, '%d %d' % (i,j))
                    valueBuyCount+=1
                if(i%3 == 2):
                    print str(i)+" "+str(j)
                    print "Data Sell "+str(tableType)+" "+str(valueSellCount)
                    e.insert(END, '%s' % str(allSellPoints[tableType][valueSellCount]))
                    #e.insert(END, '%d %d' % (i,j))
                    valueSellCount+=1

            cols.append(e)



        rows.append(cols)

        colCount += 1
    else :
        showerror("Error", "Error, Paramiter of MACD short over Paramiter of MACD long")

#################################### command for clear button
#########################################################################
######

def clear():
    #varindicator1.set(' Select Indicator  ')
    var1.set('0')
    var2.set('0')
    var3.set('0')
    var4.set('0')
    var5.set('0')
    var6.set('0')
    var7.set('0')
    var8.set('0')
    var9.set('0')
    var10.set('0')
    var11.set('0')
    var12.set('0')
```

```
    varindicator2.set(' Select Stock Group  ')
    varindicator3.set(' Select Stock   ')
    varindicator4.set('')
    varindicator5.set('')
    entryindicator1.delete(0, 'end')
    entryindicator2.delete(0, 'end')
    entryindicator3.delete(0, 'end')

######################################## command for auto button
################################################

def auto():

    Label(tablefuzzy1, text='Data of stock', fg="black",bg='#d4f4f2', font=('Monotype Corsiva',
11)).place(relx=0.125, rely=0.07,anchor=E)

    buttonfuzzydetail = Button(tablefuzzy1, text = "More Detail", background='white',
command = openweb, fg="black", font=('forte', 11)) #buttonmoredetail
    buttonfuzzydetail.place(relx=0.845, rely=0.95,anchor=SW, width=100, height=30)

    day = []

    text_file = open("quoteNameSET50/KBANK.csv", "r")
    try:
        reader = csv.reader(text_file)
        for row in reader:
            if "data" in row:
                continue
            day.append(row[0])
    finally:
        text_file.close()

    var1.set('1')
    varindicator2.set('SET50')
    varindicator3.set('KBANK')
    varindicator4.set(day[0])
    varindicator5.set(day[180])
    entryindicator1.delete(0,'12')
```

```
    entryindicator1.insert(0,'12')
    entryindicator2.delete(0,'24')
    entryindicator2.insert(0,'24')
    entryindicator3.delete(0,'9')
    entryindicator3.insert(0,'9')
    G = varindicator2.get()

    url =
"http://www.settrade.com/C04_01_stock_quote_p1.jsp?txtSymbol="+varindicator3.get()+"&ss
oPageId=9&selectPage=1"
    htmlfile = urllib.urlopen(url)
    htmltext = htmlfile.read()

    regex = '<div class="col-xs-12 colorGreen"><h1>(.+?)</h1></div>'
    pattern = re.compile(regex)
    cp = re.findall(pattern,htmltext)#CurrentPrice
    if cp==[]:
        regex = '<div class="col-xs-12 colorRed"><h1>(.+?)</h1></div>'
        pattern = re.compile(regex)
        cp = re.findall(pattern,htmltext) #CurrentPrice
        if cp==[]:
            regex = '<div class="col-xs-12 "><h1>(.+?)</h1></div>'
            pattern = re.compile(regex)
            cp = re.findall(pattern,htmltext) #CurrentPrice

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    pe = re.findall(pattern,htmltext) #PE

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    eps = re.findall(pattern,htmltext) #eps

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    div = re.findall(pattern,htmltext)#div

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
```

```
    pattern = re.compile(regex)
    divperstock = re.findall(pattern,htmltext)#div

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    yie = re.findall(pattern,htmltext)#yie

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    mc = re.findall(pattern,htmltext)#mc

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    so = re.findall(pattern,htmltext)#so

    regex = '<div class="text-right col-xs-4"><strong>(.+?)</strong></div>'
    pattern = re.compile(regex)
    pac = re.findall(pattern,htmltext)#pac
################################## Change and Change rate
########################################
    global chk_starttag
    chk_starttag = False

    global change_value
    change_value = []

    class MyHTMLParser(HTMLParser):

        def handle_starttag(self, tag, attrs):
            if attrs != [] and tag == "h1" and attrs[0][1] == "colorRed":
                global chk_starttag
                chk_starttag = True
            if attrs != [] and tag == "h1" and attrs[0][1] == "colorGreen":
                global chk_starttag
                chk_starttag = True

        def handle_data(self, data):
            if(chk_starttag):
```

```
            change_value.append(data)
        global chk_starttag
        chk_starttag = False
    parser = MyHTMLParser()
    parser.feed(htmltext)

    if(change_value == []):
        change_value.append(0)
        change_value.append(0)
        change_value.append(0)

    change = str(change_value[0]).replace(" ","").replace("\r","").replace("\n","")
    rate_change = str(change_value[1]).replace(" ","").replace("\r","").replace("\n","")
#########################################################################
#########################

    labelindicator5 = Label(tablefuzzy2, text = varindicator3.get()+"                ",
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label5 in indicator
    labelindicator5.place(relx=0.3, rely=0.05,anchor=NW)

    labelindicator6 = Label(tablefuzzy2, text = "Change  :"+"   "+change+"              ",
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 13))#label5 in indicator
    labelindicator6.place(relx=0.15, rely=0.525,anchor=NW)

    labelindicator7 = Label(tablefuzzy2, text = "Rate Change  :"+"   "+rate_change+"
", fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 13))#label5 in indicator
    labelindicator7.place(relx=0.15, rely=0.75,anchor=NW)

    labelindicator18 = Label(tablefuzzy2, text = "Current Price   :"+"   "+cp[0]+"          ",
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 13))#label5 in indicator
    labelindicator18.place(relx=0.15, rely=0.3,anchor=NW)

    labelindicator9 = Label(tablefuzzy1, text = "P/E  (TTM)   :"+"   "+pe[0],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator9.place(relx=0.01, rely=0.15,anchor=NW)

    labelindicator10 = Label(tablefuzzy1, text = "EPS  (TTM)   :"+"   "+eps[2],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
```

```
    labelindicator10.place(relx=0.01, rely=0.35,anchor=NW)

    labelindicator11 = Label(tablefuzzy1, text = "Dividend    :"+"    "+div[5],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator11.place(relx=0.01, rely=0.58,anchor=NW)

    labelindicator15 = Label(tablefuzzy1, text = "Dividend  per  stock   :"+"
"+divperstock[3], fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator15.place(relx=0.01, rely=0.78,anchor=NW)

    labelindicator12 = Label(tablefuzzy1, text = "Shares  Outstanding  (Mil.)   :"+"    "+so[7],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator12.place(relx=0.4, rely=0.15,anchor=NW)

    labelindicator13 = Label(tablefuzzy1, text = "Market  Cap  (Mil.)   :"+"    "+mc[6],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator13.place(relx=0.4, rely=0.35,anchor=NW)

    labelindicator14 = Label(tablefuzzy1, text = "Yield  ( % )   :"+"    "+yie[1],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator14.place(relx=0.4, rely=0.58,anchor=NW)

    labelindicator16 = Label(tablefuzzy1, text = "Price to carrying amount   :"+"    "+pac[4],
fg="black",bg='#d4f4f2', font=('Monotype Corsiva', 15))#label8 in indicator
    labelindicator16.place(relx=0.4, rely=0.78,anchor=NW)

######################################## command for more detail button
#################################################

def openweb():
    url =
"http://www.settrade.com/C04_01_stock_quote_p1.jsp?txtSymbol="+varindicator3.get()+"&ss
oPageId=9&selectPage=1"
    webbrowser.open(url)

buttonindicator1 = Button(T3, text = " OK ", background="#00fa9a", command = opt,
fg="black", font=('forte', 13)) #button1 in indicator
buttonindicator1.place(relx=0.01, rely=0.975,anchor=SW, width=70, height=30)
```

```
buttonindicator2 = Button(T3, text = " CLEAR ", background="#c71585", command = clear,
fg="black", font=('forte', 13)) #button2 in indicator
buttonindicator2.place(relx=0.09, rely=0.975,anchor=SW, width=70, height=30)

buttonindicator3 = Button(T3, text = " AUTOSET ", background="#7b68ee", command = auto,
fg="black", font=('forte', 13)) #button2 in indicator
buttonindicator3.place(relx=0.17, rely=0.975,anchor=SW, width=100, height=30)

tableindicator4 = Frame(T3, relief=GROOVE, borderwidth=4,width=700, height=85,
bg='#e1f9f7') #table3 in indicator
tableindicator4.place(relx=0.35, rely=0.03, anchor=NW, width=700, height=397)

################################## table result success
profit##############################################


class VerticalScrolledFrame(tk.Frame):

    def __init__(self, parent, *args, **kw):
        tk.Frame.__init__(self, parent, *args, **kw)

        # create a canvas object and a vertical scrollbar for scrolling it
        vscrollbar = tk.Scrollbar(self, orient=tk.VERTICAL)
        vscrollbar.pack(fill=tk.Y, side=tk.RIGHT, expand=tk.FALSE)

        canvas = tk.Canvas(self, bd=0, highlightthickness=0, yscrollcommand=vscrollbar.set)
        canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=tk.TRUE)

        vscrollbar.config(command=canvas.yview)

        # reset the view
        canvas.yview_moveto(0)

        # create a frame inside the canvas which will be scrolled with it
        self.interior = interior = tk.Frame(canvas)
        interior_id = canvas.create_window(0, 0, window=interior,anchor=tk.NW)
```

```
        # track changes to the canvas and frame width and sync them,
        # also updating the scrollbar
        def _configure_interior(event):
            # update the scrollbars to match the size of the inner frame
            size = (interior.winfo_reqwidth(), interior.winfo_reqheight())
            canvas.config(scrollregion="0 0 %s %s" % size)
            if interior.winfo_reqwidth() != canvas.winfo_width():
                # update the canvas's width to fit the inner frame
                canvas.config(width=interior.winfo_reqwidth())

        interior.bind('<Configure>', _configure_interior)

        def _configure_canvas(event):
            if interior.winfo_reqwidth() != canvas.winfo_width():
                # update the inner frame's width to fill the canvas
                canvas.itemconfigure(interior_id, width=canvas.winfo_width())
        canvas.bind('<Configure>', _configure_canvas)



########################################## table result buy sell
#################################################

class VerticalScrolledFrame(tk.Frame):

    def __init__(self, parent, *args, **kw):
        tk.Frame.__init__(self, parent, *args, **kw)

        # create a canvas object and a vertical scrollbar for scrolling it
        vscrollbar = tk.Scrollbar(self, orient=tk.VERTICAL)
        vscrollbar.pack(fill=tk.Y, side=tk.RIGHT, expand=tk.FALSE)

        canvas = tk.Canvas(self, bd=0, highlightthickness=0, yscrollcommand=vscrollbar.set)
        canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=tk.TRUE)

        vscrollbar.config(command=canvas.yview)

        # reset the view
```

```
        canvas.yview_moveto(0)

        # create a frame inside the canvas which will be scrolled with it
        self.interior = interior = tk.Frame(canvas)
        interior_id = canvas.create_window(0, 0, window=interior,anchor=tk.NW)

        # track changes to the canvas and frame width and sync them,
        # also updating the scrollbar
        def _configure_interior(event):
            # update the scrollbars to match the size of the inner frame
            size = (interior.winfo_reqwidth(), interior.winfo_reqheight())
            canvas.config(scrollregion="0 0 %s %s" % size)
            if interior.winfo_reqwidth() != canvas.winfo_width():
                # update the canvas's width to fit the inner frame
                canvas.config(width=interior.winfo_reqwidth())

        interior.bind('<Configure>', _configure_interior)

        def _configure_canvas(event):
            if interior.winfo_reqwidth() != canvas.winfo_width():
                # update the inner frame's width to fill the canvas
                canvas.itemconfigure(interior_id, width=canvas.winfo_width())
        canvas.bind('<Configure>', _configure_canvas)



f.pack()
root.mainloop()
```

ตารางที่ ข.3 ตารางแสดงรหัสต้นฉบับในไฟล์ Setup.py

```
from distutils.core import setup
import py2exe
import sys

sys.setrecursionlimit(10000)
from glob import glob
```

```
import matplotlib       #Import then use get_py2exe_datafiles() to collect numpy datafiles.
matplotlib.use('wxagg') #Specify matplotlib backend. tkagg must still be included else
error is thrown.

data_files = [
        ("Stuff", glob(r'C:\ProjectFolder\Stuff\*.*'))
        ,("dlls", glob(r'C:\ProjectFolder\dlls\*.dll'))
        ,("pyds", glob(r'C:\ProjectFolder\pyds\*.pyd')) # py2exe specified pyd's
        ]
# Extend the tuple list because matplotlib returns a tuple list.
data_files.extend(matplotlib.get_py2exe_datafiles())  #Matplotlib - pulls it's own files

options =   {'py2exe':{#'bundle_files': 1,                    # Bundle files to exe
                'includes':
["matplotlib.backends.backend_tkagg","matplotlib.backends.backend_qt4agg"]  #
Specifically include missing modules
                ,'excludes': ['_gtkagg', 'tkagg']           # Exclude dependencies.
Reduce size.
                }
        }

setup(name='table'
,options = options
,data_files=data_files,console=['table.py'])
```

ตารางที่ ข.4 ตารางแสดงรหัสต้นฉบับในไฟล์ Updatepriceandpricehigh.py

```
import csv
import sys
import urllib
import re
import os.path


def update(stockgroup="quoteNameSET50"):

    text_file = open(stockgroup+".txt", "r")
    symbolslist = text_file.readlines()
```

```
    symbolslist = map(lambda s: s.strip(), symbolslist)
    text_file.close()

    i=0
    while i<len(symbolslist):

        ###################### Read HTML ################################

        url =
"https://www.settrade.com/C04_02_stock_historical_p1.jsp?txtSymbol="+symbolslist[i]+"&ma
x=99999999&offset=0"
        htmlfile = urllib.urlopen(url)
        htmltext = htmlfile.read()

        new_date_tem = []

        regex = '<td>(.+?)</td>'
        pattern = re.compile(regex)
        new_date = re.findall(pattern,htmltext) #date

        for j in xrange(0,len(new_date)):
            if((j*10)>=len(new_date)):
                break
            new_date_tem.append(new_date[j*10])
        new_date = new_date_tem
        ###############################################################

        new_price_tem = [] #New Price Template

        regex = '<td>(.+?)</td>'
        pattern = re.compile(regex)
        new_price = re.findall(pattern,htmltext)#newprice

        for j in xrange(0,len(new_price)):
            if(((j*10)+1)>=len(new_price)):
                break
            new_price_tem.append(new_price[(j*10)+1])
        new_price = new_price_tem
```

```
#####################################################################

new_price_high_tem = [] #New Price High Template

regex = '<td>(.+?)</td>'
pattern = re.compile(regex)
new_price_high = re.findall(pattern,htmltext)#newpriceclose

for j in xrange(0,len(new_price_high)):
    if((j*10)+2>=len(new_price_high)):
        break
    new_price_high_tem.append(new_price_high[(j*10)+2])
new_price_high = new_price_high_tem

#####################################################################

new_price_low_tem = [] #New Price Low Template

regex = '<td>(.+?)</td>'
pattern = re.compile(regex)
new_price_low = re.findall(pattern,htmltext)#newpricelow

for j in xrange(0,len(new_price_low)):
    if((j*10)+3>=len(new_price_low)):
        break
    new_price_low_tem.append(new_price_low[(j*10)+3])
new_price_low = new_price_low_tem

#####################################################################

new_price_close_tem = [] #New Price Close Template

regex = '<td>(.+?)</td>'
pattern = re.compile(regex)
new_price_close = re.findall(pattern,htmltext)#newpriceclose

for j in xrange(0,len(new_price_close)):
```

```
    if((j*10)+5>=len(new_price_close)):
        break
    new_price_close_tem.append(new_price_close[(j*10)+5])
new_price_close = new_price_close_tem

#####################################################################

print symbolslist[i]

if(os.path.exists(stockgroup+"/"+symbolslist[i]+".csv")): #Have CSV File

    print "Update "+symbolslist[i]+" File"+stockgroup

    f = open(stockgroup+"/"+symbolslist[i]+".csv", 'rt')
    try:
        reader = csv.reader(f)
        for row in reader:
            if "data" in row:
                continue
            last_dat_old = row[0]
            print "Last Date In Old CSV File is ",row[0]
            break
    finally:
        f.close()

    for pos_new_data in xrange(0,len(new_date)):
        if(new_date[pos_new_data] == last_dat_old):
            position_in_new = pos_new_data #Position of last date in new data

    update_date = []
    update_price = []
    update_price_high = []
    update_price_low = []
    update_price_close = []

    #Update New Data
    for up_new_data in xrange(0,position_in_new):
        print "Update Date [" + new_date[up_new_data] + "]"
```

```
            update_date.append(new_date[up_new_data])
            update_price.append(new_price[up_new_data])
            update_price_high.append(new_price_high[up_new_data])
            update_price_low.append(new_price_low[up_new_data])
            update_price_close.append(new_price_close[up_new_data])

        #Copy Old Data
        f = open(stockgroup+"/"+symbolslist[i]+".csv", 'rt')
        try:
            reader = csv.reader(f)
            for row in reader:
                if "data" in row:
                    continue
                update_date.append(row[0])
                update_price.append(row[1])
                update_price_high.append(row[2])
                update_price_low.append(row[3])
                update_price_close.append(row[4])
        finally:
            f.close()

        f = open(stockgroup+"/"+symbolslist[i]+".csv", 'w')

        try:
            writer = csv.writer(f,lineterminator='\n')
            writer.writerow( ('data','open', 'high', 'low', 'close') )
            for k in  xrange(0,len(update_price)):

writer.writerow((update_date[k],update_price[k],update_price_high[k],update_price_low[k],up
date_price_close[k]))
        finally:
            print("Finish Update "+symbolslist[i])
            print('')
            print('')
            f.close()

    else:
```

```
            print "Create "+symbolslist[i]+" File"+stockgroup

            f = open(stockgroup+"/"+symbolslist[i]+".csv", 'w')

            try:
                writer = csv.writer(f,lineterminator='\n')
                writer.writerow( ('data','open', 'high', 'low', 'close') )
                for k in  xrange(0,len(new_price)):

writer.writerow((new_date[k],new_price[k],new_price_high[k],new_price_low[k],new_price_clo
se[k]))
            finally:
                print("Finish Update "+symbolslist[i])
                f.close()


##################################################################

            i+=1
```

ตารางที่ ข.5 ตารางแสดงรหัสต้นฉบับในไฟล์ Lenday.py

```
import csv
import sys
import urllib
import re
import os.path
import math


def length_date(stockgroup, symbols, start_date, end_date):

    date_list = []

    f = open("quoteName"+stockgroup+"/"+symbols+".csv", 'rt')
    try:
        reader = csv.reader(f)
        for row in reader:
```

```
            if "data" in row:
                continue
            date_list.append([row[0],row[1],row[2],row[3],row[4]])

        finally:
            f.close()

        start_date_position = [i for i,x in enumerate(date_list) if x[0] == start_date]
        end_date_position = [i for i,x in enumerate(date_list) if x[0] == end_date]

        positionstart = start_date_position[0]
        positionend = end_date_position[0]+1

        print positionstart
        print positionend

        for up_new_data in xrange(positionstart,positionend):
            date_list[up_new_data]

        #Write To CSV
        f = open("example/ex.csv", 'w')
        try:
            writer = csv.writer(f,lineterminator='\n')
            writer.writerow( ('date','open', 'high', 'low', 'close') )
            for k in xrange(positionstart,positionend):
                date_split = date_list[k][0].split("/")
                date_split[2] = '20'+date_split[2]
                date_string = date_split[2]+date_split[1]+date_split[0]
                #print date_string

writer.writerow((date_string,date_list[k][1],date_list[k][2],date_list[k][3],date_list[k][4]))
        finally:
            print("Finish Write Example To CSV")
            print('')
            print('')
            f.close()

#length_date("SET50","ADVANC", "04/12/15","10/11/15")
```

ตารางที่ ข.6 ตารางแสดงรหัสต้นฉบับในไฟล์ Main_update.py

```
from updatepriceandpricehigh import update

group_list = ["quoteNameSET50","quoteNameSET100","quoteNameSETHD",
              "quoteName--AGRI","quoteNameAGRO","quoteName--AUTO",
              "quoteName--BANK","quoteName--COMM","quoteName--CONMAT",
              "quoteName--CONS","quoteNameCONSUMP","quoteName--ENERG",
              "quoteName--ETRON","quoteName--FASHION","quoteName--FIN",
              "quoteNameFINCIAL","quoteName--FOOD","quoteName--HELTH",
              "quoteName--HOME","quoteName--ICT","quoteName--IMM",
              "quoteNameINDUS","quoteName--INSUR","quoteName--MEDIA",
              "quoteName--MINE","quoteName--PAPER","quoteName--PERSON",
              "quoteName--PETRO","quoteName--PF&REIT","quoteName--PKG",
              "quoteName--PROF","quoteName--PROP","quoteNamePROPCON",
              "quoteNameRESOURC","quoteNameSERVICE","quoteName--STEEL",
              "quoteNameTECH","quoteName--TOURISM","quoteName--TRANS",
              ]

for i in group_list:
    update(i)
```

ตารางที่ ข.7 ตารางแสดงรหัสต้นฉบับในไฟล์ Mainmacdf

```
from macdtesterF.data import getPricesData
from macdtesterF.tradingtest_originalmacd import originalMACDTest, getAllBuyPoints as
org_getAllBuyPoints,\
    getAllSellPoints as org_getAllSellPoints, originalMACDTestWithSET100
from macdtesterF.macd import getMACDAndSignalLine
from macdtesterF.modifiedsignalline import getModifiedSignalLine
from macdtesterF.plot import standardPlot, addMarkers, addModifiedSignalLine
from macdtesterF.tradingtest_macdr1 import MACDR1Test, getAllBuyPoints as
macdr1_getAllBuyPoints,\
    getAllSellPoints as macdr1_getAllSellPoints, MACDR1TestWithSET100
from macdtesterF.tradingtest_macdr2 import MACDR2Test, getAllBuyPoints as
macdr2_getAllBuyPoints,\
```

```
    getAllSellPoints as macdr2_getAllSellPoints, MACDR2TestWithSET100
from macdtesterF.tradingtest_modifiedsignalline import modifiedSignalLineTest,
getAllBuyPoints as modsig_getAllBuyPoints,\
    getAllSellPoints as modsig_getAllSellPoints,\
    modifiedSignalLineTestWithSET100
from macdtesterF.basic import getTradingPeriods, getMaximumPointsInAPeriod,\
    getMinimumPointsInAPeriod
from macdtesterF.tradingtest_modsigtradingweight import modsigTradingWeightTest,
getAllBuyPoints as modsigw_getAllBuyPoints,\
    getAllSellPoints as modsigw_getAllSellPoints,\
    modsigTradingWeightTestWithSET100



def runmacd(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
    print paraMACDLong
    print parasignalShort

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)

    print 'preparing data...'

    #macdDef()

    #def macdDef():
    ## orlginal MACD test with a single data
    print 'original MACD test for ' + symbol
    successRate, profitRate = originalMACDTest(pricesData, macd, signalLine)
    ############################## Sell and buy point Error
####################################
```

```
    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


##########################################################################
######################

    #raw_input("Press Enter to continue...")

    allBuyPoints = org_getAllBuyPoints(pricesData, macd, signalLine)
    allSellPoints = org_getAllSellPoints(pricesData, macd, signalLine)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints

def runmacdr1(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
    print paraMACDLong
    print parasignalShort

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)
    ## MACDR1 test with a single data
    print 'MACDR1 test for ' + symbol
    successRate, profitRate = MACDR1Test(pricesData, macd, signalLine)
```

```
    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


#########################################################################
######################

    # raw_input("Press Enter to continue...")

    allBuyPoints = macdr1_getAllBuyPoints(pricesData, macd, signalLine)
    allSellPoints = macdr1_getAllSellPoints(pricesData, macd, signalLine)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints

def runmacdr2(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
    print paraMACDLong
    print parasignalShort

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)
```

```
    ## MACDR2 test with a single data
    print 'MACDR2 test for ' + symbol
    successRate, profitRate = MACDR2Test(pricesData, macd, signalLine)

    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


#########################################################################
######################
    # raw_input("Press Enter to continue...")

    allBuyPoints = macdr2_getAllBuyPoints(pricesData, macd, signalLine)
    allSellPoints = macdr2_getAllSellPoints(pricesData, macd, signalLine)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints

def runmacdf(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
    print paraMACDLong
    print parasignalShort

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
```

```
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)
    ## modified signal line test with a single data
    print 'modified signal line test for ' + symbol
    successRate, profitRate = modifiedSignalLineTest(pricesData, macd, signalLine,
modifiedSignalLine)
    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


#########################################################################
######################

    # raw_input("Press Enter to continue...")

    allBuyPoints = modsig_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine)
    allSellPoints = modsig_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addModifiedSignalLine(fig, modifiedSignalLine, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints

def runmacdf1(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
```

```
    print paraMACDLong
    print parasignalShort

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)
    ## modified signal line with trading weight test with a single data
    print 'modified signal line with trading weight test for ' + symbol
    successRate, profitRate = modsigTradingWeightTest(pricesData, macd, signalLine,
modifiedSignalLine, 1)
    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


#########################################################################
#####################

  # raw_input("Press Enter to continue...")

    allBuyPoints = modsigw_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)
    allSellPoints = modsigw_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addModifiedSignalLine(fig, modifiedSignalLine, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)
```

```
    return successRate, profitRate, allBuyPoints, allSellPoints
```

ตารางที่ ข.8 ตารางแสดงรหัสต้นฉบับในไฟล์ Mainmacdk

```
from macdtesterK.data import getPricesData
from macdtesterK.tradingtest_originalmacd import originalMACDTest, getAllBuyPoints as
org_getAllBuyPoints,\
    getAllSellPoints as org_getAllSellPoints, originalMACDTestWithSET100
from macdtesterK.macd import getMACDAndSignalLine
from macdtesterK.modifiedsignalline import
getModifiedSignalLine,getModifiedSignalLineWithK
from macdtesterK.plot import standardPlot, addMarkers,
addModifiedSignalLine,addModifiedSignalLineWithK
from macdtesterK.tradingtest_macdr1 import MACDR1Test, getAllBuyPoints as
macdr1_getAllBuyPoints,\
    getAllSellPoints as macdr1_getAllSellPoints, MACDR1TestWithSET100
from macdtesterK.tradingtest_macdr2 import MACDR2Test, getAllBuyPoints as
macdr2_getAllBuyPoints,\
    getAllSellPoints as macdr2_getAllSellPoints, MACDR2TestWithSET100
from macdtesterK.tradingtest_modifiedsignalline import modifiedSignalLineTest,
getAllBuyPoints as modsig_getAllBuyPoints,\
    getAllSellPoints as modsig_getAllSellPoints,\
    modifiedSignalLineTestWithSET100
from macdtesterK.basic import getTradingPeriods, getMaximumPointsInAPeriod,\
    getMinimumPointsInAPeriod
from macdtesterK.tradingtest_modsigtradingweight import modsigTradingWeightTest,
getAllBuyPoints as modsigw_getAllBuyPoints,\
    getAllSellPoints as modsigw_getAllSellPoints,\
    modsigTradingWeightTestWithSET100
from macdtesterK.tradingtest_originalmacd import originalMACDTest, getAllBuyPoints as
org_getAllBuyPoints,\
    getAllSellPoints as org_getAllSellPoints, originalMACDTestWithSET100
from macdtesterK.tradingtest_modifiedsignalline_withK import modifiedSignalLineWithKtest,
getAllBuyPoints as org_getAllBuyPoints,\
    getAllSellPoints as org_getAllSellPoints, modifiedSignalLineWithKtestInSET100
from macdtesterK.tradingtest_modifiedsignalline_withK_1 import MACDK1Test,
getAllBuyPoints as macdr1_getAllBuyPoints,\
```

```
    getAllSellPoints as macdr1_getAllSellPoints, MACDK1TestWithSET100
from macdtesterK.tradingtest_modifiedsignalline_withK_2 import MACDK2Test,
getAllBuyPoints as macdr2_getAllBuyPoints,\
    getAllSellPoints as macdr2_getAllSellPoints, MACDK2TestWithSET100

def runmacdk(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
    print paraMACDLong
    print parasignalShort

    print 'preparing data...'

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)
    modifiedSignalLineWithK = getModifiedSignalLineWithK(macd, signalLine)

    pricesData = pricesData[25:]
    macd = macd[25:]
    signalLine = signalLine[25:]
    modifiedSignalLine = modifiedSignalLine[25:]
    modifiedSignalLineWithK = modifiedSignalLineWithK[25:]

    ## modified signal line test with a single data
    print 'modified signal line with K test for ' + symbol
    successRate, profitRate = modifiedSignalLineWithKtest(pricesData, macd,
modifiedSignalLineWithK)
    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
```

```
        print "Not have buy point sell point!!"


##########################################################################
######################

        #raw_input("Press Enter to continue...")

    allBuyPoints = org_getAllBuyPoints(pricesData, macd, modifiedSignalLineWithK)
    allSellPoints = org_getAllSellPoints(pricesData, macd, modifiedSignalLineWithK)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addModifiedSignalLineWithK(fig, modifiedSignalLineWithK, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints

def runmacdk1(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
    print paraMACDLong
    print parasignalShort

    print 'preparing data...'

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)
    modifiedSignalLineWithK = getModifiedSignalLineWithK(macd, signalLine)
        ## Modified signal Line with K-1 test with a single data
    print 'MACDK1 test for ' + symbol
    successRate, profitRate = MACDK1Test(pricesData, macd, modifiedSignalLineWithK)
```

```
    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


#########################################################################
######################

        #raw_input("Press Enter to continue...")

    allBuyPoints = macdr1_getAllBuyPoints(pricesData, macd, modifiedSignalLineWithK)
    allSellPoints = macdr1_getAllSellPoints(pricesData, macd, modifiedSignalLineWithK)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addModifiedSignalLineWithK(fig, modifiedSignalLineWithK, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints

def runmacdk2(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
    print paraMACDLong
    print parasignalShort

    print 'preparing data...'

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
```

```
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)
    modifiedSignalLineWithK = getModifiedSignalLineWithK(macd, signalLine)
        ## Modified signal Line with K-2 test with a single data
    print 'MACDK2 test for ' + symbol
    successRate, profitRate = MACDK2Test(pricesData, macd, modifiedSignalLineWithK)
    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


#########################################################################
######################

    #raw_input("Press Enter to continue...")

    allBuyPoints = macdr2_getAllBuyPoints(pricesData, macd, modifiedSignalLineWithK)
    allSellPoints = macdr2_getAllSellPoints(pricesData, macd, modifiedSignalLineWithK)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addModifiedSignalLineWithK(fig, modifiedSignalLineWithK, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints

def runmacdf2(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
```

```
    print paraMACDLong
    print parasignalShort

    print 'preparing data...'

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)
    modifiedSignalLineWithK = getModifiedSignalLineWithK(macd, signalLine)
        ## Modified signal line test with a single data
    print 'Modified signal line test for ' + symbol
    successRate, profitRate = modifiedSignalLineTest(pricesData, macd, signalLine,
modifiedSignalLine)
    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


#########################################################################
######################

        #raw_input("Press Enter to continue...")

    allBuyPoints = modsig_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine)
    allSellPoints = modsig_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addModifiedSignalLine(fig, modifiedSignalLine, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
```

```
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints
```

ตารางที่ ข.9 ตารางแสดงรหัสต้นฉบับในไฟล์ Mainmacdp

```
from macdtesterP.data import getPricesData
from macdtesterP.tradingtest_originalmacd import originalMACDTest, getAllBuyPoints as
org_getAllBuyPoints,\
    getAllSellPoints as org_getAllSellPoints, originalMACDTestWithSET100
from macdtesterP.macd import getMACDAndSignalLine
from macdtesterP.modifiedsignalline import getModifiedSignalLine
from macdtesterP.plot import standardPlot, addMarkers, addModifiedSignalLine
from macdtesterP.tradingtest_macdr1 import MACDR1Test, getAllBuyPoints as
macdr1_getAllBuyPoints,\
    getAllSellPoints as macdr1_getAllSellPoints, MACDR1TestWithSET100
from macdtesterP.tradingtest_macdr2 import MACDR2Test, getAllBuyPoints as
macdr2_getAllBuyPoints,\
    getAllSellPoints as macdr2_getAllSellPoints, MACDR2TestWithSET100
from macdtesterP.tradingtest_modifiedsignalline import modifiedSignalLineTest,
getAllBuyPoints as modsig_getAllBuyPoints,\
    getAllSellPoints as modsig_getAllSellPoints,\
    modifiedSignalLineTestWithSET100
from macdtesterP.basic import getTradingPeriods, getMaximumPointsInAPeriod,\
    getMinimumPointsInAPeriod
from macdtesterP.tradingtest_modsigtradingweight import modsigTradingWeightTest,
getAllBuyPoints as modsigw_getAllBuyPoints,\
    getAllSellPoints as modsigw_getAllSellPoints,\
    modsigTradingWeightTestWithSET100

def runmacdp(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
    print paraMACDLong
    print parasignalShort

    print 'preparing data...'
```

```
    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)

    ## prepare data for a single data

    pricesData = pricesData[25:]
    macd = macd[25:]
    signalLine = signalLine[25:]
    modifiedSignalLine = modifiedSignalLine[25:]

    ## modified signal line test with a single data
    print 'modified signal line test for ' + symbol
    successRate, profitRate = modifiedSignalLineTest(pricesData, macd, signalLine,
modifiedSignalLine)
    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


#########################################################################
######################
    #raw_input("Press Enter to continue...")

    allBuyPoints = modsig_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine)
    allSellPoints = modsig_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
```

```
    addModifiedSignalLine(fig, modifiedSignalLine, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints

def runmacdp1(getSymbol,paraMACDShort,paraMACDLong,parasignalShort) :
    ## prepare data for a single data
    print getSymbol
    print paraMACDShort
    print paraMACDLong
    print parasignalShort

    print 'preparing data...'

    symbol = str(getSymbol)
    pricesData = getPricesData(symbol)
    macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
    modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)
    ## modified signal line with trading weight test with a single data
    print 'modified signal line with trading weight test for ' + symbol
    successRate, profitRate = modsigTradingWeightTest(pricesData, macd, signalLine,
modifiedSignalLine, 1)
    ############################## Sell and buy point Error
#####################################

    try:
        print 'success rate: %f' % successRate
        print 'profit rate: %f' % profitRate
    except (RuntimeError, TypeError, NameError):
        print "Not have buy point sell point!!"


#########################################################################
######################
    #raw_input("Press Enter to continue...")
```

```
    allBuyPoints = modsigw_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)
    allSellPoints = modsigw_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)
    #print allBuyPoints
    #print allSellPoints

    fig = standardPlot(pricesData, macd, signalLine, show=False)
    addModifiedSignalLine(fig, modifiedSignalLine, show=False)
    addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
    addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

    return successRate, profitRate, allBuyPoints, allSellPoints
```

ตารางที่ ข.10 ตารางแสดงรหัสต้นฉบับในไฟล์ data.py ในโฟลเดอร์macdtesterK

```
import pandas as pd

class DataDefinition (str):
    def __new__(cls, name, colNumber, dtype):
        return str.__new__(cls, name)

    def __init__(self, name, colNumber, dtype):
        super(DataDefinition, self).__init__(name)
        self.colNumber = colNumber
        self.dtype = dtype


DATE = DataDefinition('date', 0, 'S8')
OPEN = DataDefinition('open', 1, 'f8')
HIGH = DataDefinition('high', 2, 'f8')
LOW = DataDefinition('low', 3, 'f8')
CLOSE = DataDefinition('close', 4, 'f8')

def _getRawData(uri="./example/ex.csv"):
    data = pd.read_csv(uri, dtype={DATE: DATE.dtype,
                        OPEN: OPEN.dtype, HIGH: HIGH.dtype,
                        LOW: LOW.dtype, CLOSE: CLOSE.dtype,
```

```
                }, parse_dates=[0], index_col=0, na_values='-')

    return data.sort_index(ascending=True).dropna()

def getPricesData(symbol='KBANK', type='close'):
    if type not in [OPEN, HIGH, LOW, CLOSE]:
        raise Exception('param-value error: type= "' + type +'". Hint, type=x for x in {"open",
"high", "low", "close"}')
    uri = "./example/ex.csv"

    rawData = None
    try:
        rawData = _getRawData(uri)
    except:
        raise Exception('error occur while parsing raw data. Hint, please check if symbol="' +
symbol + '" exists and matches the name of csv file.')

    data = rawData[type]
    return data
```

ตารางที่ ข.11 ตารางแสดงรหัสต้นฉบับในไฟล์ macd.py ในโฟลเดอร์macdtesterK

```
import pandas as pd
import pandas.stats.moments as mt
from ema import getEMA

def getMACDAndSignalLine(pricesData, n1, n2, n3):
    ema_n1 = getEMA(pricesData, n1)
    ema_n2 = getEMA(pricesData, n2)
    macd = ema_n1 - ema_n2
    signalLine = pd.Series(mt.ewma(macd, span=n3), index=pricesData.index)
    return macd, signalLine
```

ตารางที่ ข.12 ตารางแสดงรหัสต้นฉบับในไฟล์ basic.py ในโฟลเดอร์macdtesterK

```
import numpy as np
```

```
def getTradingPeriods(macd, signalLine):
    downCrossingPoints = getDownCrossingPoints(macd, signalLine)
    periods = []
    for i in xrange(len(downCrossingPoints)-1):
        periods.append(macd[downCrossingPoints[i]:downCrossingPoints[i+1]].index)

    return periods


def isAnUpCrossingPoint(macd, signalLine, t):
    i = macd.index.get_loc(t)

    if i >= 1 and (macd.ix[i] > signalLine.ix[i]) and (macd.ix[i-1] < signalLine.ix[i-1]):
        return True
    elif i >= 2 and (macd.ix[i] > signalLine.ix[i]) and (macd.ix[i-1] == signalLine.ix[i-1]) and
(macd.ix[i-2] < signalLine.ix[i-2]):
        return True
    else:
        False

def isADownCrossingPoint(macd, signalLine, t):
    i = macd.index.get_loc(t)

    if i >= 1 and (macd.ix[i] < signalLine.ix[i]) and (macd.ix[i-1] > signalLine.ix[i-1]):
        return True
    elif i >= 2 and (macd.ix[i] < signalLine.ix[i]) and (macd.ix[i-1] == signalLine.ix[i-1]) and
(macd.ix[i-2] > signalLine.ix[i-2]):
        return True
    else:
        False

def getUpCrossingPoints(macd, signalLine):
    upCrossingPointList = []
    firstDownCrossingExists = False
    for t in macd.index:
        if not firstDownCrossingExists and isADownCrossingPoint(macd, signalLine, t):
            firstDownCrossingExists = True
```

```
        if firstDownCrossingExists and isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPointList.append(t)

    return upCrossingPointList

def getDownCrossingPoints(macd, signalLine):
    downCrossingPointList = []
    for t in macd.index:
        if isADownCrossingPoint(macd, signalLine, t):
            downCrossingPointList.append(t)

    return downCrossingPointList

class Trade(object):
    def __init__(self, buyPoint, sellPoint, capital, weight=1):
        self.buyPoint = buyPoint
        self.sellPoint = sellPoint
        self.capital = capital
        self.weight = weight

    def getProfit(self, pricesData):
        return ((pricesData[self.sellPoint] -
pricesData[self.buyPoint])/float(pricesData[self.buyPoint]))*self.capital*self.weight

class TotalTrade(object):
    def __init__(self, tradeList):
        self.initialCapital = tradeList[0].capital
        for trade in tradeList:
            if trade.capital != self.initialCapital:
                raise Exception('All trade in tradeList must have the same capital.')

        self.tradeList = tradeList

    def getProfitRate(self, pricesData):
        if len(self.tradeList) == 1:
            profitRate = self.tradeList[0].getProfit(pricesData)/float(self.initialCapital)
        else:
            profits = np.array([trade.getProfit(pricesData) for trade in self.tradeList])
```

```
        profitRate = np.sum(profits)/float(self.initialCapital - np.sum(profits[profits < 0]))

        return profitRate

def getUpCrossingPointInAPeriod(macd, signalLine, period):
    upCrossingPoint = None
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t
            break

    return upCrossingPoint

def getMaximumPointsInAPeriod(macd, signalLine, period):
    upCrossingPoint = None
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t
            break

    maximumPoint = macd[upCrossingPoint:period[-1]].idxmax()
    return maximumPoint

def getMinimumPointsInAPeriod(macd, signalLine, period):
    upCrossingPoint = None
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t
            break

    minimumPoint = macd[period[0]:upCrossingPoint].idxmin()
    return minimumPoint
```

ตารางที่ ข.13 ตารางแสดงรหัสต้นฉบับในไฟล์ ema.py ในโฟลเดอร์macdtesterK

```
import pandas as pd
import pandas.stats.moments as mt
```

```
def getEMA(pricesData, n=12):
    ema = pd.Series(mt.ewma(pricesData, span=n), index=pricesData.index)
    return ema
```

ตารางที่ ข.14 ตารางแสดงรหัสต้นฉบับในไฟล์ modifiedsignalline.py ในโฟลเดอร์macdtesterK

```
import pandas as pd
from macdtesterK.basic import getTradingPeriods,
isAnUpCrossingPoint,isADownCrossingPoint


def getModifiedSignalLine(macd, signalLine, K):
    periods = getTradingPeriods(macd, signalLine)
    modifiedSignalLine = pd.Series(float('nan'), index=signalLine.index)

    for period in periods:
        avmin = getAvailablyMinimumPointsInAPeriod(macd, signalLine, period)
        avmax = getAvailablyMaximumPointsInAPeriod(macd, signalLine, period)
        dmin = getMinimumCertainties(macd, signalLine, period)
        dmax = getMaximumCertainties(macd, signalLine, period)
        upCrossingPoint = None
        for t in period:
            if upCrossingPoint is None and isAnUpCrossingPoint(macd, signalLine, t):
                upCrossingPoint = t

            if upCrossingPoint is None:
                hMinimum = macd[avmin[t]] - signalLine[avmin[t]]
                modifiedSignalLine[t] = signalLine[t] + hMinimum*(dmin[t]**K)
            elif t == period[-1]:
                modifiedSignalLine[t] = signalLine[t]
            else:
                hMaximum = macd[avmax[t]] - signalLine[avmax[t]]
                modifiedSignalLine[t] = signalLine[t] + hMaximum*(dmax[t]**K)

    return modifiedSignalLine

def getModifiedSignalLineWithK(macd, signalLine):
```

```
modifiedSignalLineWithK = pd.Series(float('nan'), index=signalLine.index)

hmax=-9999999
hmin=-9999999
ht=1

for t in signalLine.index:

    if (macd[t]>=signalLine[t]):
        ht = macd[t] - signalLine[t]
    else:
        ht = signalLine[t] - macd[t]

    if (macd[t] >= signalLine[t]) and (macd[t] - signalLine[t])> hmax :
        hmax = macd[t] - signalLine[t]
    elif (signalLine[t]>=macd[t]) and (signalLine[t]-macd[t])> hmin:
        hmin = signalLine[t] - macd[t]
    if (hmax==-9999999 or hmin==-9999999):
        k=1
        m=1
    else:
        k=(hmax+hmin)/(2*ht)
        m=(2*ht)/(hmax+hmin)

    if (signalLine[t] >= 0):
        if (macd[t] >= signalLine[t]):
            if (k<2):
                modifiedSignalLineWithK[t]=signalLine[t]
            else:
                modifiedSignalLineWithK[t]=signalLine[t]

        elif (signalLine[t]>=macd[t]):
            if (m<0.5):
                modifiedSignalLineWithK[t]=signalLine[t]*m
            else:
                    modifiedSignalLineWithK[t]=signalLine[t]

    elif (signalLine[t] < 0):
```

```
            if (macd[t] >= signalLine[t]):
                if (m<0.7):
                    modifiedSignalLineWithK[t]=signalLine[t]
                else:
                    modifiedSignalLineWithK[t]=signalLine[t]

            elif (signalLine[t]>=macd[t]):
                if (k<2):
                    modifiedSignalLineWithK[t]=signalLine[t]
                else:
                    modifiedSignalLineWithK[t]=signalLine[t]*k

        return modifiedSignalLineWithK


def getAvailablyMaximumPointsInAPeriod(macd, signalLine, period):
    avialablyMaximumPoints = pd.Series('', index=period)
    upCrossingPoint = None
    for t in period:
        if upCrossingPoint is None and isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t

        if upCrossingPoint is not None:
            avialablyMaximumPoints[t] = macd[upCrossingPoint:t].idxmax()

    return avialablyMaximumPoints

def getAvailablyMinimumPointsInAPeriod(macd, signalLine, period):
    avialablyMinimumPoints = pd.Series('', index=period)
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            avialablyMinimumPoints[t] = macd[period[0]:t].idxmin()
            break

        avialablyMinimumPoints[t] = macd[period[0]:t].idxmin()

    return avialablyMinimumPoints
```

```
def getMaximumCertainties(macd, signalLine, period):
    avmax = getAvailablyMaximumPointsInAPeriod(macd, signalLine, period)
    maximumCertainties = pd.Series('', index=period)
    upCrossingPoint = None
    for t in period:
        if t == period[-1]:
            maximumCertainties[t] = 1
            break

        if upCrossingPoint is None and isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t

        if upCrossingPoint is not None:
            hCurrent = macd[t] - signalLine[t]
            hMaximum = macd[avmax[t]] - signalLine[avmax[t]]
            maximumCertainties[t] = 1 - (hCurrent/float(hMaximum)) if hMaximum != 0 else 1

    return maximumCertainties

def getMinimumCertainties(macd, signalLine, period):
    avmin = getAvailablyMinimumPointsInAPeriod(macd, signalLine, period)
    minimumCertainties = pd.Series('', index=period)
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            minimumCertainties[t] = 1
            break

        hCurrent = macd[t] - signalLine[t]
        hMinimum = macd[avmin[t]] - signalLine[avmin[t]]
        minimumCertainties[t] = 1 - (hCurrent/float(hMinimum)) if hMinimum != 0 else 1

    return minimumCertainties
```

ตารางที่ ข.15 ตารางแสดงรหัสต้นฉบับในไฟล์ plot.py ในโฟลเดอร์macdtesterK

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
import numpy as np
from matplotlib import ticker, font_manager

class IndexManager(object):
    def __init__(self, indexSeries):
        self.index = indexSeries
        self.evenlyIndex = pd.Series(np.arange(len(indexSeries)), index=indexSeries)

    def evenlyIndexToDateTimeIndex(self, x):
        try:
            return self.index[x]
        except:
            return None

    def datetimeIndexToEvenlyIndex(self, d):
        try:
            return self.evenlyIndex[d]
        except:
            return None

    def dateMappingFormatter(self, x, pos=None):
        d = self.evenlyIndexToDateTimeIndex(int(x + 0.5))
        if d is not None:
            return d.strftime('%b\n%Y')
        else:
            return ''

def standardPlot(pricesData, macd, signalLine, show=True):
    idm = IndexManager(pricesData.index)
    plt.rc('axes', grid=True)
    plt.rc('grid', color='0.75', linestyle='-', linewidth=0.5)

    left, width = 0.1, 0.8
    rect1 = [left, 0.6, width, 0.3]
    rect2 = [left, 0.1, width, 0.5]

    fig = plt.figure(facecolor='white')
    axescolor  = '#f9f9f9'  # the axes background color
```

```
    ax1 = fig.add_axes(rect1, axisbg=axescolor)  #left, bottom, width, height
    ax2 = fig.add_axes(rect2, axisbg=axescolor, sharex=ax1)


    #we don't want to label all indices, just the first day in each month are enough
    #this function return the list of the (evenly) indices of the first day in each month
    def getIndexOfFirstDayEachMonth(datearr):
        dateSeries = pd.Series(datearr.month)
        isFirstDay = np.ones_like(dateSeries, dtype=np.bool)
        isFirstDay[1:] = (dateSeries[1:] != dateSeries[:-1])
        return dateSeries.index[isFirstDay]

    #we then use the above mentioned list as a 'FixedLocator' and set it as the major locator

ax1.xaxis.set_major_locator(ticker.FixedLocator(getIndexOfFirstDayEachMonth(pricesData.inde
x)))

    #only the indices in the locator will be labeled, the plotter will ask the 'formatter' which
labels corresponds to which indices
    #this is the place where our mapping is used
    ax1.xaxis.set_major_formatter(ticker.FuncFormatter(idm.dateMappingFormatter))

    #set nice left and right spaces
    ax1.set_xlim(idm.evenlyIndex[0] - 2, idm.evenlyIndex[-1] + 2)

    ax1.plot(idm.evenlyIndex, pricesData, color='black', lw=2, label='closing price')
    ax2.plot(idm.evenlyIndex, macd, color='red', lw=1.5, label='MACD Line')
    ax2.plot(idm.evenlyIndex, signalLine, color='blue', lw=1, label='Signal Line')

    for label in ax1.get_xticklabels():
        label.set_visible(False)

    ax1.yaxis.set_major_locator(ticker.MaxNLocator(5, prune='both'))
    ax2.yaxis.set_major_locator(ticker.MaxNLocator(5, prune='both'))

    props = font_manager.FontProperties(size=10)
    leg1 = ax1.legend(loc='best', shadow=True, fancybox=True, prop=props, scatterpoints=1,
markerscale=1)
```

```
    leg1.get_frame().set_alpha(0.5)
    leg2 = ax2.legend(loc='best', shadow=True, fancybox=True, prop=props, scatterpoints=1,
markerscale=1)
    leg2.get_frame().set_alpha(0.5)
    if show: plt.show()
    return fig

def addModifiedSignalLine(fig, modifiedSignalLine, color='violet', show=True):
    idm = IndexManager(modifiedSignalLine.index)
    ax2 = fig.get_axes()[1]
    ax2.plot(idm.evenlyIndex, modifiedSignalLine, color=color, lw=1, label='Modified Signal
Line')
    if show: plt.show()
    return fig

def addMarkers(fig, macd, tList, markerSize=40, color='#00CC00', show=True):
    idm = IndexManager(macd.index)
    ax2 = fig.get_axes()[1]
    ax2.scatter(idm.evenlyIndex[tList], macd[tList], color=color, s=markerSize)
    if show: plt.show()
    return fig
def addModifiedSignalLineWithK(fig, modifiedSignalLineWithK, color='orange', show=True):
    idm = IndexManager(modifiedSignalLineWithK.index)
    ax2 = fig.get_axes()[1]
    ax2.plot(idm.evenlyIndex, modifiedSignalLineWithK, color='orange', lw=1, label='Modified
Signal Line With K')
    props = font_manager.FontProperties(size=10)
    leg2 = ax2.legend(loc='best', shadow=True, fancybox=True, prop=props, scatterpoints=1,
markerscale=1)
    leg2.get_frame().set_alpha(0.5)
    if show: plt.show()
    return fig
```

ตารางที่ ข.16 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_originalmacd.py ในโฟลเดอร์macdtesterK

```
from macdtesterK.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
from macdtesterK.macd import getMACDAndSignalLine
```

```
import numpy as np
from macdtesterK.data import getPricesData
import os
import pandas as pd

def isABuyPoint(macd, signalLine, t):
    if isAnUpCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def isASellPoint(macd, signalLine, t):
    if isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(macd, signalLine, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, signalLine, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(macd, signalLine, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None
```

```
def originalMACDTest(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, signalLine, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
    print 'total buy' ,len(allBuyPoints)
    #print allBuyPoints
    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, signalLine, period)
        if totalTrade is not None:
```

```
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)
    print 'total sell' ,len(allSellPoints)
    #print allSellPoints
    return allSellPoints


def originalMACDTestWithSET100(verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        successRate, profitRate = originalMACDTest(pricesData, macd, signalLine)

        getAllBuyPoints(pricesData, macd, signalLine)
        getAllSellPoints(pricesData, macd, signalLine)



        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
```

```
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)


    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.17 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_macdr1.py ในโฟลเดอร์macdtesterK

```
from macdtesterK.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
from macdtesterK.macd import getMACDAndSignalLine
import numpy as np
import os
import pandas as pd
from macdtesterK.data import getPricesData

def isABuyPoint(macd, signalLine, t):
    i = macd.index.get_loc(t)
    if (i >=2 and
        macd.ix[i] > signalLine.ix[i] and
        macd.ix[i-1] > signalLine.ix[i-1] and
        isAnUpCrossingPoint(macd, signalLine, macd.index[i-2])):
        return True
    else:
        return False

def isASellPoint(pricesData, macd, signalLine, buyPoint, t):
    profitRate = (pricesData[t] - pricesData[buyPoint])/float(pricesData[buyPoint])
```

```
    if profitRate >= 0.03 or isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(pricesData, macd, signalLine, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, signalLine, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(pricesData, macd, signalLine, buyPoint, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def MACDR1Test(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
```

```
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
    print 'total buy' ,len(allBuyPoints)
    #print allBuyPoints
    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)
    print 'total sell' ,len(allSellPoints)
    #print allSellPoints
    return allSellPoints

def MACDR1TestWithSET100(verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]
```

```
    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData,  int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        successRate, profitRate = MACDR1Test(pricesData, macd, signalLine)

        getAllBuyPoints(pricesData, macd, signalLine)
        getAllSellPoints(pricesData, macd, signalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('MACDR1 test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.18 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_macdr2.py ในโฟลเดอร์macdtesterK

```
from macdtesterK.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
import numpy as np
```

```
import os
import pandas as pd
from macdtesterK.data import getPricesData
from macdtesterK.macd import getMACDAndSignalLine

def isABuyPoint(pricesData, macd, signalLine, t):
    i = macd.index.get_loc(t)
    if (i >=2 and
        macd.ix[i] > signalLine.ix[i] and
        macd.ix[i-1] > signalLine.ix[i-1] and
        isAnUpCrossingPoint(macd, signalLine, macd.index[i-2]) and
        (macd.ix[i] - signalLine.ix[i])/float(pricesData.ix[i]) >= 0.005):
        return True
    else:
        return False

def isASellPoint(pricesData, macd, signalLine, buyPoint, t):
    profitRate = (pricesData[t] - pricesData[buyPoint])/float(pricesData[buyPoint])
    if profitRate >= 0.03 or isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(pricesData, macd, signalLine, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(pricesData, macd, signalLine, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(pricesData, macd, signalLine, buyPoint, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
```

```
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def MACDR2Test(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
    print 'total buy' ,len(allBuyPoints)
    #print allBuyPoints
    return allBuyPoints
```

```
def getAllSellPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)
    print 'total sell' ,len(allSellPoints)
    #print allSellPoints
    return allSellPoints


def MACDR2TestWithSET100(verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        successRate, profitRate = MACDR2Test(pricesData, macd, signalLine)

        getAllBuyPoints(pricesData, macd, signalLine)
        getAllSellPoints(pricesData, macd, signalLine)
        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
```

```
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('MACDR2 test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.19 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_modifiedsignalline.py ในโฟลเดอร์ macdtesterK

```
from macdtesterK.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods, getUpCrossingPointInAPeriod
from macdtesterK.macd import getMACDAndSignalLine
import numpy as np
import os
import pandas as pd
from macdtesterK.data import getPricesData
from macdtesterK.modifiedsignalline import getModifiedSignalLine

def isABuyPoint(macd, modifiedSignalLine, upPoint, t):
    iup = macd.index.get_loc(upPoint)
    i = macd.index.get_loc(t)
    if i <= iup and isAnUpCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
        return False

def isASellPoint(macd, modifiedSignalLine, t):
    if isADownCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
```

```
        return False

def getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, modifiedSignalLine, upPoint, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(macd, modifiedSignalLine, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def modifiedSignalLineTest(pricesData, macd, signalLine, modifiedSignalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for period in periods:
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)
```

```
    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
    print 'total buy' ,len(allBuyPoints)
    #print allBuyPoints
    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)
    print 'total sell' ,len(allSellPoints)
    #print allSellPoints
    return allSellPoints

def modifiedSignalLineTestWithSET100(K=1, verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]
```

```
    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
        modifiedSignalLine = getModifiedSignalLine(macd, signalLine, K)
        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        modifiedSignalLine = modifiedSignalLine[25:]
        successRate, profitRate = modifiedSignalLineTest(pricesData, macd, signalLine,
modifiedSignalLine)

        getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine)
        getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('MACDP test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.20 ตารางแสดงรหัสต้นฉบับในไฟล์ Tradingtest_modifiedsignallinewithK ในโฟลเดอร์ macdtesterK

```
from macdtesterK.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
from macdtesterK.macd import getMACDAndSignalLine
from macdtesterK.modifiedsignalline import getModifiedSignalLineWithK
import numpy as np
from macdtesterK.data import getPricesData
import os
import pandas as pd

def isABuyPoint(macd,modifiedSignalLineWithK, t):
    if isAnUpCrossingPoint(macd, modifiedSignalLineWithK, t):
        return True
    else:
        return False

def isASellPoint(macd, signalLine, t):
    if isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(macd, signalLine, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, signalLine, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(macd, signalLine, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None
```

```
    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def modifiedSignalLineWithKtest(pricesData, macd, modifiedSignalLineWithK):
    periods = getTradingPeriods(macd, modifiedSignalLineWithK)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLineWithK, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, modifiedSignalLineWithK):
    periods = getTradingPeriods(macd, modifiedSignalLineWithK)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLineWithK, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
    print 'total buy' ,len(allBuyPoints)
    #print allBuyPoints
    return allBuyPoints

def getAllSellPoints(pricesData, macd, modifiedSignalLineWithK):
    periods = getTradingPeriods(macd,modifiedSignalLineWithK)
```

```
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd,modifiedSignalLineWithK, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)
    print 'total sell',len(allSellPoints)
    #print allSellPoints
    return allSellPoints


def modifiedSignalLineWithKtestInSET100(verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)

        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
        modifiedSignalLineWithK = getModifiedSignalLineWithK(macd, signalLine)

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        modifiedSignalLineWithK =modifiedSignalLineWithK[25:]

        successRate, profitRate = modifiedSignalLineWithKtest(pricesData, macd,
modifiedSignalLineWithK)

        getAllBuyPoints(pricesData, macd, modifiedSignalLineWithK)
        getAllSellPoints(pricesData, macd, modifiedSignalLineWithK)
```

```
        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Modifiedsignalline withK test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.21 ตารางแสดงรหัสต้นฉบับในไฟล์ Tradingtest_modifiedsignallinewithK_1 ในโฟลเดอร์ macdtesterK

```
from macdtesterK.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
from macdtesterK.macd import getMACDAndSignalLine
from macdtesterK.modifiedsignalline import getModifiedSignalLineWithK
import numpy as np
import os
import pandas as pd
from macdtesterK.data import getPricesData

def isABuyPoint(macd,modifiedSignalLineWithK , t):
```

```
    i = macd.index.get_loc(t)
    if (i >=2 and
        macd.ix[i] > modifiedSignalLineWithK.ix[i] and
        macd.ix[i-1] > modifiedSignalLineWithK.ix[i-1] and
        isAnUpCrossingPoint(macd, modifiedSignalLineWithK, macd.index[i-2])):
        return True
    else:
        return False

def isASellPoint(pricesData, macd, modifiedSignalLineWithK, buyPoint, t):
    profitRate = (pricesData[t] - pricesData[buyPoint])/float(pricesData[buyPoint])
    if profitRate >= 0.03 or isADownCrossingPoint(macd, modifiedSignalLineWithK, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(pricesData, macd, modifiedSignalLineWithK, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, modifiedSignalLineWithK, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(pricesData, macd,modifiedSignalLineWithK,
buyPoint, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None
```

```
def MACDK1Test(pricesData, macd, modifiedSignalLineWithK):
    periods = getTradingPeriods(macd, modifiedSignalLineWithK)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, modifiedSignalLineWithK,
period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, modifiedSignalLineWithK):
    periods = getTradingPeriods(macd, modifiedSignalLineWithK)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, modifiedSignalLineWithK,
period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
    print 'total sell' ,len(allBuyPoints)
    #print allBuyPoints
    return allBuyPoints

def getAllSellPoints(pricesData, macd, modifiedSignalLineWithK):
    periods = getTradingPeriods(macd, modifiedSignalLineWithK)
    allSellPoints = []
    for period in periods:
```

```
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, modifiedSignalLineWithK,
period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)
    print 'total sell' ,len(allSellPoints)
    #print allSellPoints
    return allSellPoints

def MACDK1TestWithSET100(verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
        modifiedSignalLineWithK=getModifiedSignalLineWithK(macd,signalLine)

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        modifiedSignalLineWithK=modifiedSignalLineWithK[25:]

        successRate, profitRate = MACDK1Test(pricesData, macd, modifiedSignalLineWithK)

        getAllBuyPoints(pricesData, macd, modifiedSignalLineWithK)
        getAllSellPoints(pricesData, macd, modifiedSignalLineWithK)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
```

```
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Modified_K1 test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.22 ตารางแสดงรหัสต้นฉบับในไฟล์ Tradingtest_modifiedsignallinewithK_2 ในโฟลเดอร์ macdtesterK

```
from macdtesterK.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
import numpy as np
import os
import pandas as pd
from macdtesterK.data import getPricesData
from macdtesterK.macd import getMACDAndSignalLine
from macdtesterK.modifiedsignalline import getModifiedSignalLineWithK

def isABuyPoint(pricesData, macd, modifiedSignalLineWithK, t):
    i = macd.index.get_loc(t)
    if (i >=2 and
        macd.ix[i] > modifiedSignalLineWithK.ix[i] and
        macd.ix[i-1] > modifiedSignalLineWithK.ix[i-1] and
        isAnUpCrossingPoint(macd, modifiedSignalLineWithK, macd.index[i-2]) and
        (macd.ix[i] - modifiedSignalLineWithK.ix[i])/float(pricesData.ix[i]) >= 0.005):
        return True
    else:
        return False

def isASellPoint(pricesData, macd, modifiedSignalLineWithK, buyPoint, t):
    profitRate = (pricesData[t] - pricesData[buyPoint])/float(pricesData[buyPoint])
    if profitRate >= 0.03 or isADownCrossingPoint(macd, modifiedSignalLineWithK, t):
        return True
```

```
    else:
        return False

def getTotalTradesInAPeriod(pricesData, macd,modifiedSignalLineWithK, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(pricesData, macd, modifiedSignalLineWithK, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(pricesData, macd, modifiedSignalLineWithK,
buyPoint, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def MACDK2Test(pricesData, macd, modifiedSignalLineWithK):
    periods = getTradingPeriods(macd, modifiedSignalLineWithK)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, modifiedSignalLineWithK,
period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
```

```
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, modifiedSignalLineWithK):
    periods = getTradingPeriods(macd, modifiedSignalLineWithK)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, modifiedSignalLineWithK,
period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
    print 'total buy' ,len(allBuyPoints)
    #print allBuyPoints
    return allBuyPoints

def getAllSellPoints(pricesData, macd, modifiedSignalLineWithK):
    periods = getTradingPeriods(macd, modifiedSignalLineWithK)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, modifiedSignalLineWithK,
period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)
    print 'total sell' ,len(allSellPoints)
    #print allSellPoints
    return allSellPoints


def MACDK2TestWithSET100(verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
```

```
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData,  int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
        modifiedSignalLineWithK=getModifiedSignalLineWithK(macd,signalLine)

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        modifiedSignalLineWithK=modifiedSignalLineWithK[25:]
        successRate, profitRate = MACDK2Test(pricesData, macd, modifiedSignalLineWithK)

        getAllBuyPoints(pricesData, macd, modifiedSignalLineWithK)
        getAllSellPoints(pricesData, macd, modifiedSignalLineWithK)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Modified_K2 test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.23 ตารางแสดงรหัสต้นฉบับในไฟล์ Mainmacdk.py ในโฟลเดอร์macdtesterK

```
from macdtesterK.data import getPricesData
from macdtesterK.tradingtest_originalmacd import originalMACDTest, getAllBuyPoints as
org_getAllBuyPoints,\
    getAllSellPoints as org_getAllSellPoints, originalMACDTestWithSET100
from macdtesterK.macd import getMACDAndSignalLine
from macdtesterK.modifiedsignalline import getModifiedSignalLine
from macdtesterK.plot import standardPlot, addMarkers, addModifiedSignalLine
from macdtesterK.tradingtest_macdr1 import MACDR1Test, getAllBuyPoints as
macdr1_getAllBuyPoints,\
    getAllSellPoints as macdr1_getAllSellPoints, MACDR1TestWithSET100
from macdtesterK.tradingtest_macdr2 import MACDR2Test, getAllBuyPoints as
macdr2_getAllBuyPoints,\
    getAllSellPoints as macdr2_getAllSellPoints, MACDR2TestWithSET100
from macdtesterK.tradingtest_modifiedsignalline import modifiedSignalLineTest,
getAllBuyPoints as modsig_getAllBuyPoints,\
    getAllSellPoints as modsig_getAllSellPoints,\
    modifiedSignalLineTestWithSET100
from macdtesterK.basic import getTradingPeriods, getMaximumPointsInAPeriod,\
    getMinimumPointsInAPeriod
from macdtesterK.tradingtest_modsigtradingweight import modsigTradingWeightTest,
getAllBuyPoints as modsigw_getAllBuyPoints,\
    getAllSellPoints as modsigw_getAllSellPoints,\
    modsigTradingWeightTestWithSET100

## prepare data for a single data
print 'preparing data...'
symbol = 'KBANK'
pricesData = getPricesData(symbol)
macd, signalLine = getMACDAndSignalLine(pricesData, 12, 26, 9)
modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)

pricesData = pricesData[25:]
macd = macd[25:]
signalLine = signalLine[25:]
modifiedSignalLine = modifiedSignalLine[25:]
```

```
## original MACD test with a single data
print 'original MACD test for ' + symbol
successRate, profitRate = originalMACDTest(pricesData, macd, signalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = org_getAllBuyPoints(pricesData, macd, signalLine)
allSellPoints = org_getAllSellPoints(pricesData, macd, signalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## MACDR1 test with a single data
print 'MACDR1 test for ' + symbol
successRate, profitRate = MACDR1Test(pricesData, macd, signalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = macdr1_getAllBuyPoints(pricesData, macd, signalLine)
allSellPoints = macdr1_getAllSellPoints(pricesData, macd, signalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## MACDR2 test with a single data
print 'MACDR2 test for ' + symbol
successRate, profitRate = MACDR2Test(pricesData, macd, signalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = macdr2_getAllBuyPoints(pricesData, macd, signalLine)
allSellPoints = macdr2_getAllSellPoints(pricesData, macd, signalLine)
```

```
fig = standardPlot(pricesData, macd, signalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## modified signal line test with a single data
print 'modified signal line test for ' + symbol
successRate, profitRate = modifiedSignalLineTest(pricesData, macd, signalLine,
modifiedSignalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = modsig_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine)
allSellPoints = modsig_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addModifiedSignalLine(fig, modifiedSignalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## modified signal line with trading weight test with a single data
print 'modified signal line with trading weight test for ' + symbol
successRate, profitRate = modsigTradingWeightTest(pricesData, macd, signalLine,
modifiedSignalLine, 1)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = modsigw_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)
allSellPoints = modsigw_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addModifiedSignalLine(fig, modifiedSignalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)
```

```
## original MACD test in SET-100
print 'original MACD test in SET-100'
successRate, profitRate = originalMACDTestWithSET100(verbose=True)
print 'original MACD test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## MACDR1 test in SET-100
print 'MACDR1 test in SET-100'
successRate, profitRate = MACDR1TestWithSET100(verbose=True)
print 'MACDR1 test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## MACDR2 test in SET-100
print 'MACDR2 test in SET-100'
successRate, profitRate = MACDR2TestWithSET100(verbose=True)
print 'MACDR2 test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## modified signal line test in SET-100
print 'modified signal line test in SET-100'
successRate, profitRate = modifiedSignalLineTestWithSET100(K=0.1, verbose=True)
print 'modified signal line test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## modified signal line with trading weight test in SET-100
print 'modified signal line with trading weight test in SET-100'
successRate, profitRate = modsigTradingWeightTestWithSET100(K=0.1, R=1, verbose=True)
print 'modified signal line with trading weight test in SET-100'
```

```
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
```

ตารางที่ ข.24 ตารางแสดงรหัสต้นฉบับในไฟล์ basic.py ในโฟลเดอร์macdtesterF

```
import numpy as np

def getTradingPeriods(macd, signalLine):
    downCrossingPoints = getDownCrossingPoints(macd, signalLine)
    periods = []
    for i in xrange(len(downCrossingPoints)-1):
        periods.append(macd[downCrossingPoints[i]:downCrossingPoints[i+1]].index)

    return periods


def isAnUpCrossingPoint(macd, signalLine, t):
    i = macd.index.get_loc(t)

    if i >= 1 and (macd.ix[i] > signalLine.ix[i]) and (macd.ix[i-1] < signalLine.ix[i-1]):
        return True
    elif i >= 2 and (macd.ix[i] > signalLine.ix[i]) and (macd.ix[i-1] == signalLine.ix[i-1]) and
(macd.ix[i-2] < signalLine.ix[i-2]):
        return True
    else:
        False

def isADownCrossingPoint(macd, signalLine, t):
    i = macd.index.get_loc(t)

    if i >= 1 and (macd.ix[i] < signalLine.ix[i]) and (macd.ix[i-1] > signalLine.ix[i-1]):
        return True
    elif i >= 2 and (macd.ix[i] < signalLine.ix[i]) and (macd.ix[i-1] == signalLine.ix[i-1]) and
(macd.ix[i-2] > signalLine.ix[i-2]):
        return True
    else:
        False
```

```
def getUpCrossingPoints(macd, signalLine):
    upCrossingPointList = []
    firstDownCrossingExists = False
    for t in macd.index:
        if not firstDownCrossingExists and isADownCrossingPoint(macd, signalLine, t):
            firstDownCrossingExists = True

        if firstDownCrossingExists and isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPointList.append(t)

    return upCrossingPointList

def getDownCrossingPoints(macd, signalLine):
    downCrossingPointList = []
    for t in macd.index:
        if isADownCrossingPoint(macd, signalLine, t):
            downCrossingPointList.append(t)

    return downCrossingPointList

class Trade(object):
    def __init__(self, buyPoint, sellPoint, capital, weight=1):
        self.buyPoint = buyPoint
        self.sellPoint = sellPoint
        self.capital = capital
        self.weight = weight

    def getProfit(self, pricesData):
        return ((pricesData[self.sellPoint] -
pricesData[self.buyPoint])/float(pricesData[self.buyPoint]))*self.capital*self.weight

class TotalTrade(object):
    def __init__(self, tradeList):
        self.initialCapital = tradeList[0].capital
        for trade in tradeList:
            if trade.capital != self.initialCapital:
                raise Exception('All trade in tradeList must have the same capital.')
```

```
        self.tradeList = tradeList

    def getProfitRate(self, pricesData):
        if len(self.tradeList) == 1:
            profitRate = self.tradeList[0].getProfit(pricesData)/float(self.initialCapital)
        else:
            profits = np.array([trade.getProfit(pricesData) for trade in self.tradeList])
            profitRate = np.sum(profits)/float(self.initialCapital - np.sum(profits[profits < 0]))

        return profitRate

def getUpCrossingPointInAPeriod(macd, signalLine, period):
    upCrossingPoint = None
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t
            break

    return upCrossingPoint

def getMaximumPointsInAPeriod(macd, signalLine, period):
    upCrossingPoint = None
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t
            break

    maximumPoint = macd[upCrossingPoint:period[-1]].idxmax()

    return maximumPoint

def getMinimumPointsInAPeriod(macd, signalLine, period):
    upCrossingPoint = None
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t
            break
```

```
    minimumPoint = macd[period[0]:upCrossingPoint].idxmin()

    return minimumPoint
```

ตารางที่ ข.25 ตารางแสดงรหัสต้นฉบับในไฟล์ Fuzzy.py ในโฟลเดอร์macdtesterF

```
import numpy as np
import matplotlib.pyplot as plt

class Universe(object):
    def __init__(self, lower, upper):
        assert lower <= upper
        self.lower = lower
        self.upper = upper

    def contains(self, value):
        return self.lower <= value and value <= self.upper

    def contains_all(self, values):
        values = np.array(values)
        return (self.lower <= values).all() and (values <= self.upper).all()

    def sample(self, step=None):
        if step is None:
            step = 0.01 if self.upper - self.lower <= 10 else 0.1

        return np.arange(self.lower, self.upper, step)

    def __eq__(self, other):
        return other is not None and isinstance(other, Universe) and other.lower ==
self.lower and other.upper == self.upper

class Variable(object):
    def __init__(self, universe):
        assert isinstance(universe, Universe)
        self.universe = universe
        self.value = None
```

```
        self.terms = dict()
        self.abcd_for_term = dict()

    def set_value(self, value):
        assert self.universe.contains(value) or value is None
        self.value = value

    def set_term(self, term_name, abcd=(None,None,None,None), description=''):
        assert len(abcd) == 4
        if self.abcd_for_term.has_key(term_name):
            for i in xrange(len(abcd)):
                if abcd[i] is not None and abcd[i] != '':
                    self.abcd_for_term[term_name][i] = abcd[i]
        else:
            self.abcd_for_term[term_name] = list(abcd)

        if all((param is not None and param != '') for param in
self.abcd_for_term[term_name]):
            self.terms[term_name] = FuzzySet(self.universe, self.abcd_for_term[term_name],
description=description)
        else:
            self.terms[term_name] = None

    def fuzzify(self, term_name):
        if self.terms[term_name] is None:
            print [type(param) for param in self.abcd_for_term[term_name]]
            assert all((type(param) in [int, float]) for param in self.abcd_for_term[term_name])
            raise Exception("Some parameters have not been set for term: %s, params: %s" %
(term_name, str(self.abcd_for_term[term_name])))

        return self.terms[term_name](self.value)

    def defuzzify(self, accum_mf):
        outs = self.universe.sample()
        weights = accum_mf(outs)
        sum_weights = weights.sum()
        out = (outs * weights).sum()/sum_weights if sum_weights != 0 else None
        self.set_value(out)
```

```
        return out

    def displayterm(self, term_name, xlim):
        if self.terms[term_name] is None:
            raise Exception("Some parameters have not been set for term: %s, params: %s" %
(term_name, str(self.abcd_for_term[term_name])))

        fig = plt.figure(facecolor='white')
        ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.set_xlim(xlim)
        ax.set_ylim(0, 1.2)

        x = self.universe.sample(step=0.005)
        y = self.terms[term_name](x)

        ax.plot(x,y,lw=3)
        ax.set_title(term_name, fontsize=20)
        plt.show()

class FuzzySet(object):

    def __init__(self, universe, abcd, mf=None, description=''):
        assert isinstance(universe, Universe)
        assert (abcd is not None and mf is None) or (abcd is None and mf is not None)

        if abcd is not None:
            assert len(abcd) == 4 and universe.contains_all(abcd)
            self.mf = FuzzySet.fuzzy_number_mf(*abcd)
        else:
            self.mf = mf

        self.universe = universe
        self.description = description


    def __call__(self, values):
        assert self.universe.contains_all(values)
```

```
        return self.mf(values)

    def __and__(self, other):
        assert isinstance(other, (int, float, FuzzySet))
        if isinstance(other, FuzzySet):
            assert self.universe == other.universe
            def new_mf(x):
                return FuzzyOperation.AND(self(x), other(x))
        else:
            def new_mf(x):
                return FuzzyOperation.AND(self(x), other)

        return FuzzySet(self.universe, None, mf=new_mf)

    def __or__(self, other):
        assert isinstance(other, (int, float, FuzzySet))
        if isinstance(other, FuzzySet):
            assert self.universe == other.universe
            def new_mf(x):
                return FuzzyOperation.OR(self(x), other(x))
        else:
            def new_mf(x):
                return FuzzyOperation.OR(self(x), other)

        return FuzzySet(self.universe, None, mf=new_mf)

    class fuzzy_number_mf(object):
        def __init__(self, a, b, c, d):
            #assert a <= b and b <= c and c <= d
            if not (a <= b and b <= c and c <= d): raise Exception, 'not! (%f <= %f <= %f <=
%f)' % (a, b, c, d)



            def mf(x):
                if x < a: return 0
                elif a <= x and x < b: return (x-a)/float(b-a)
                elif b <= x and x < c: return 1
                elif c <= x and x < d: return (d-x)/float(d-c)
```

```
            elif x == d and d == c: return 1
            else: return 0

        self.func = np.frompyfunc(mf, 1, 1)

    def __call__(self, x):
        return self.func(x)

class FuzzyOperation(object):

    @classmethod
    def define_operation_and(cls, operation):
        assert isinstance(operation, FuzzyOperation.Interface)
        cls.AND = operation

    @classmethod
    def define_operation_or(cls, operation):
        assert isinstance(operation, FuzzyOperation.Interface)
        cls.OR = operation

    class Interface(object):
        def __call__(self, a, b): raise NotImplementedError

    class MAX(Interface):
        def __call__(self, a, b):
            return np.maximum(a, b)

    class MIN(Interface):
        def __call__(self, a, b):
            return np.minimum(a, b)

    class PROD(Interface):
        def __call__(self, a, b):
            return a * b

    class SUM(Interface):
        def __call__(self, a, b):
            return a + b
```

```
    AND = MIN()
    OR = MAX()

class FuzzyRule(object):
    def __init__(self, antecedence, consequences):
        self.antecedence = antecedence
        self.consequences = consequences

    def evaluate(self, inputs, outputs):
        '''

        :param variable
        :return a dictionary of the form {'z1': activ_mf_z1, 'z2': activ_mf_z2, ...}
            where zi is the output-variable's name and activ_mf_zi is the corresponding result
membership function from this rule's evaluation.
        '''

        w = self._aggregate(inputs)
        activ_mf = self._activate(w, outputs)
        return activ_mf

    def _aggregate(self, inputs):

        stack = []
        for element in self.antecedence:
            if isinstance(element, FuzzyOperation.Interface):
                operation = element
                token1 = stack.pop()
                fuzzified_value1 = token1 if not isinstance(token1, tuple) else
inputs[token1[0]].fuzzify(token1[1])

                token2 = stack.pop()
                fuzzified_value2 = token2 if not isinstance(token2, tuple) else
inputs[token2[0]].fuzzify(token2[1])

                result_token = operation(fuzzified_value1, fuzzified_value2)
                stack.append(result_token)
```

```
            else:
                stack.append(element)

        if len(stack) != 1: print stack
        assert len(stack) == 1

        token = stack.pop()
        w = token if not isinstance(token, tuple) else inputs[token[0]].fuzzify(token[1])

        return w

    def _activate(self, w, outputs):
        activ_mf = dict()
        for element in self.consequences:
            variable_name, term_name = element
            activ_mf[variable_name] = outputs[variable_name].terms[term_name] & w

        return activ_mf

class FIS(object):
    def __init__(self):
        self._define(self.fuzzydefinition)

    def get_inputs(self):
        return self.inputs

    def get_outputs(self):
        return self.outputs

    def set_inputs(self, **kwargs):
        assert set(kwargs.keys()) == set(self.inputs.keys())

        for key, value in kwargs.iteritems():
            assert self.inputs[key].universe.contains(value)
            self.inputs[key].set_value(value)

        return self
```

```
    def set_term_param(self, variable_name, term_name, abcd):
        self.inputs[variable_name].set_term(term_name, abcd=abcd)

        return self

    def evaluate(self):
        accum_mf = dict()
        for output_variable_name in self.outputs:
            accum_mf[output_variable_name] = None

        for rule in self.fuzzy_rules:
            for output_variable_name in self.outputs:
                activ_mf = rule.evaluate(self.inputs, self.outputs)
                if output_variable_name in activ_mf.keys():
                    accum_mf[output_variable_name] = (activ_mf[output_variable_name]
                                                      if accum_mf[output_variable_name] is None
                                                      else accum_mf[output_variable_name] |
activ_mf[output_variable_name])

        for output_variable_name, output_variable in self.outputs.iteritems():
            output_variable.defuzzify(accum_mf[output_variable_name])

        return self

    def _define(self, definition):
        properties = {'inputs': None, 'outputs': None, 'fuzzy_rules': None}
        definition(properties)
        assert properties['inputs'] is not None
        assert properties['outputs'] is not None
        assert properties['fuzzy_rules'] is not None
        self.inputs = properties['inputs']
        self.fuzzy_rules = properties['fuzzy_rules']
        self.outputs = properties['outputs']

        return self

    def fuzzydefinition(self, properties):
        REAL = Universe(-999, 999)
```

```
        NORM = Universe(0, 1)

        #inputs
        properties['inputs'] = {'dmin': Variable(NORM),
                                'macd_avmin': Variable(REAL)}

        #outputs
        properties['outputs'] = {'z': Variable(NORM)}

        #input_terms
        properties['inputs']['dmin'].set_term('MIN-CONFIDENCE', abcd=(0, 1, 1, 1))

        properties['inputs']['macd_avmin'].set_term('MACD-HIGH', abcd=(None, None,
REAL.upper, REAL.upper))
        properties['inputs']['macd_avmin'].set_term('MACD-LOW', abcd=(REAL.lower,
REAL.lower, None, None))
        properties['inputs']['macd_avmin'].set_term('MACD-MEDIUM', abcd=(None, None,
None, None))


        #output_terms
        properties['outputs']['z'].set_term('TRADE-CONFIDENCE', abcd=(0, 1, 1, 1))
        properties['outputs']['z'].set_term('TRADE-DOUBT', abcd=(0, 0, 0, 1))


        #fuzzy_rules
        AND = FuzzyOperation.MIN()
        OR = FuzzyOperation.MAX()
        properties['fuzzy_rules'] = [
                    FuzzyRule(antecedence = (('dmin', 'MIN-CONFIDENCE'),),
                        consequences = (('z', 'TRADE-CONFIDENCE'),)),

                    FuzzyRule(antecedence = (('macd_avmin', 'MACD-LOW'),),
                        consequences = (('z', 'TRADE-CONFIDENCE'),)),

                    FuzzyRule(antecedence = (('macd_avmin', 'MACD-HIGH'), ('macd_avmin',
'MACD-MEDIUM'), OR),
                        consequences = (('z', 'TRADE-DOUBT'),))
```

```
]
```

ตารางที่ ข.26 ตารางแสดงรหัสต้นฉบับในไฟล์ data.py ในโฟลเดอร์macdtesterF

```
import pandas as pd

class DataDefinition (str):
    def __new__(cls, name, colNumber, dtype):
        return str.__new__(cls, name)

    def __init__(self, name, colNumber, dtype):
        super(DataDefinition, self).__init__(name)
        self.colNumber = colNumber
        self.dtype = dtype


DATE = DataDefinition('date', 0, 'S8')
OPEN = DataDefinition('open', 1, 'f8')
HIGH = DataDefinition('high', 2, 'f8')
LOW = DataDefinition('low', 3, 'f8')
CLOSE = DataDefinition('close', 4, 'f8')

def _getRawData(uri="./example/ex.csv"):
    data = pd.read_csv(uri, dtype={DATE: DATE.dtype,
                        OPEN: OPEN.dtype, HIGH: HIGH.dtype,
                        LOW: LOW.dtype, CLOSE: CLOSE.dtype,
                        }, parse_dates=[0], index_col=0, na_values='-')

    return data.sort_index(ascending=True).dropna()

def getPricesData(symbol='CHOTI', types='close'):
    if types not in [OPEN, HIGH, LOW, CLOSE]:
        raise Exception('param-value error: type= "' + types +'". Hint, type=x for x in {"open",
"high", "low", "close"}')
    uri = "./example/ex.csv"

    rawData = None
```

```
    try:
        rawData = _getRawData(uri)
    except:
        raise Exception('error occur while parsing raw data. Hint, please check if symbol="' +
symbol + '" exists and matches the name of csv file.')

    data = rawData[types]
    return data
```

ตารางที่ ข.27 ตารางแสดงรหัสต้นฉบับในไฟล์ ema.py ในโฟลเดอร์macdtesterF

```
import pandas as pd
import pandas.stats.moments as mt

def getEMA(pricesData, n=12):
    ema = pd.Series(mt.ewma(pricesData, span=n), index=pricesData.index)
    return ema
```

ตารางที่ ข.28 ตารางแสดงรหัสต้นฉบับในไฟล์ macd.py ในโฟลเดอร์macdtesterF

```
import pandas as pd
import pandas.stats.moments as mt
from ema import getEMA

def getMACDAndSignalLine(pricesData, n1, n2, n3):
    ema_n1 = getEMA(pricesData, n1)
    ema_n2 = getEMA(pricesData, n2)
    macd = ema_n1 - ema_n2
    signalLine = pd.Series(mt.ewma(macd, span=n3), index=pricesData.index)
    return macd, signalLine
```

ตารางที่ ข.29ตารางแสดงรหัสต้นฉบับในไฟล์ plot.py ในโฟลเดอร์macdtesterF

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
import numpy as np
from matplotlib import ticker, font_manager

class IndexManager(object):
    def __init__(self, indexSeries):
        self.index = indexSeries
        self.evenlyIndex = pd.Series(np.arange(len(indexSeries)), index=indexSeries)

    def evenlyIndexToDateTimeIndex(self, x):
        try:
            return self.index[x]
        except:
            return None

    def datetimeIndexToEvenlyIndex(self, d):
        try:
            return self.evenlyIndex[d]
        except:
            return None

    def dateMappingFormatter(self, x, pos=None):
        d = self.evenlyIndexToDateTimeIndex(int(x + 0.5))
        if d is not None:
            return d.strftime('%b\n%Y')
        else:
            return ''

def standardPlot(pricesData, macd, signalLine, show=True):
    idm = IndexManager(pricesData.index)
    plt.rc('axes', grid=True)
    plt.rc('grid', color='0.75', linestyle='-', linewidth=0.5)

    left, width = 0.1, 0.8
    rect1 = [left, 0.6, width, 0.3]
    rect2 = [left, 0.1, width, 0.5]


    fig = plt.figure(facecolor='white')
```

```
    axescolor  = '#f9f9f9'  # the axes background color
    ax1 = fig.add_axes(rect1, axisbg=axescolor)  #left, bottom, width, height
    ax2 = fig.add_axes(rect2, axisbg=axescolor, sharex=ax1)


    #we don't want to label all indices, just the first day in each month are enough
    #this function return the list of the (evenly) indices of the first day in each month
    def getIndexOfFirstDayEachMonth(datearr):
        dateSeries = pd.Series(datearr.month)
        isFirstDay = np.ones_like(dateSeries, dtype=np.bool)
        isFirstDay[1:] = (dateSeries[1:] != dateSeries[:-1])
        return dateSeries.index[isFirstDay]

    #we then use the above mentioned list as a 'FixedLocator' and set it as the major locator

ax1.xaxis.set_major_locator(ticker.FixedLocator(getIndexOfFirstDayEachMonth(pricesData.inde
x)))

    #only the indices in the locator will be labeled, the plotter will ask the 'formatter' which
labels corresponds to which indices
    #this is the place where our mapping is used
    ax1.xaxis.set_major_formatter(ticker.FuncFormatter(idm.dateMappingFormatter))

    #set nice left and right spaces
    ax1.set_xlim(idm.evenlyIndex[0] - 2, idm.evenlyIndex[-1] + 2)

    ax1.plot(idm.evenlyIndex, pricesData, color='black', lw=2, label='closing price')
    ax2.plot(idm.evenlyIndex, macd, color='red', lw=1.5, label='MACD Line')
    ax2.plot(idm.evenlyIndex, signalLine, color='blue', lw=1, label='Signal Line')

    for label in ax1.get_xticklabels():
        label.set_visible(False)

    ax1.yaxis.set_major_locator(ticker.MaxNLocator(5, prune='both'))
    ax2.yaxis.set_major_locator(ticker.MaxNLocator(5, prune='both'))

    props = font_manager.FontProperties(size=10)
    leg1 = ax1.legend(loc='best', shadow=True, fancybox=True, prop=props, scatterpoints=1,
```

```
markerscale=1)
    leg1.get_frame().set_alpha(0.5)
    leg2 = ax2.legend(loc='best', shadow=True, fancybox=True, prop=props, scatterpoints=1,
markerscale=1)
    leg2.get_frame().set_alpha(0.5)
    if show: plt.show()
    return fig

def addModifiedSignalLine(fig, modifiedSignalLine, color='violet', show=True):
    idm = IndexManager(modifiedSignalLine.index)
    ax2 = fig.get_axes()[1]
    ax2.plot(idm.evenlyIndex, modifiedSignalLine, color=color, lw=1, label='Modified Signal
Line')
    if show: plt.show()
    return fig

def addMarkers(fig, macd, tList, markerSize=40, color='#00CC00', show=True):
    idm = IndexManager(macd.index)
    ax2 = fig.get_axes()[1]
    ax2.scatter(idm.evenlyIndex[tList], macd[tList], color=color, s=markerSize)
    if show: plt.show()
    return fig
```

ตารางที่ ข.30 ตารางแสดงรหัสต้นฉบับในไฟล์ modifiedsignalline.py ในโฟลเดอร์macdtesterF

```
import pandas as pd
from macdtesterF.basic import getTradingPeriods, isAnUpCrossingPoint


def getModifiedSignalLine(macd, signalLine, K):
    periods = getTradingPeriods(macd, signalLine)
    modifiedSignalLine = pd.Series(float('nan'), index=signalLine.index)

    for period in periods:
        avmin = getAvailablyMinimumPointsInAPeriod(macd, signalLine, period)
```

```
        avmax = getAvailablyMaximumPointsInAPeriod(macd, signalLine, period)
        dmin = getMinimumCertainties(macd, signalLine, period)
        dmax = getMaximumCertainties(macd, signalLine, period)
        upCrossingPoint = None
        for t in period:
            if upCrossingPoint is None and isAnUpCrossingPoint(macd, signalLine, t):
                upCrossingPoint = t

            if upCrossingPoint is None:
                hMinimum = macd[avmin[t]] - signalLine[avmin[t]]
                modifiedSignalLine[t] = signalLine[t] + hMinimum*(dmin[t]**K)
            elif t == period[-1]:
                modifiedSignalLine[t] = signalLine[t]
            else:
                hMaximum = macd[avmax[t]] - signalLine[avmax[t]]
                modifiedSignalLine[t] = signalLine[t] + hMaximum*(dmax[t]**K)

    return modifiedSignalLine


def getAvailablyMaximumPointsInAPeriod(macd, signalLine, period):
    avialablyMaximumPoints = pd.Series('', index=period)
    upCrossingPoint = None
    for t in period:
        if upCrossingPoint is None and isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t

        if upCrossingPoint is not None:
            avialablyMaximumPoints[t] = macd[upCrossingPoint:t].idxmax()

    return avialablyMaximumPoints

def getAvailablyMinimumPointsInAPeriod(macd, signalLine, period):
    avialablyMinimumPoints = pd.Series('', index=period)
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            avialablyMinimumPoints[t] = macd[period[0]:t].idxmin()
            break
```

```
        avialablyMinimumPoints[t] = macd[period[0]:t].idxmin()

    return avialablyMinimumPoints

def getMaximumCertainties(macd, signalLine, period):
    avmax = getAvailablyMaximumPointsInAPeriod(macd, signalLine, period)
    maximumCertainties = pd.Series('', index=period)
    upCrossingPoint = None
    for t in period:
        if t == period[-1]:
            maximumCertainties[t] = 1
            break

        if upCrossingPoint is None and isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t

        if upCrossingPoint is not None:
            hCurrent = macd[t] - signalLine[t]
            hMaximum = macd[avmax[t]] - signalLine[avmax[t]]
            maximumCertainties[t] = 1 - (hCurrent/float(hMaximum)) if hMaximum != 0 else 1

    return maximumCertainties

def getMinimumCertainties(macd, signalLine, period):
    avmin = getAvailablyMinimumPointsInAPeriod(macd, signalLine, period)
    minimumCertainties = pd.Series('', index=period)
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            minimumCertainties[t] = 1
            break

        hCurrent = macd[t] - signalLine[t]
        hMinimum = macd[avmin[t]] - signalLine[avmin[t]]
        minimumCertainties[t] = 1 - (hCurrent/float(hMinimum)) if hMinimum != 0 else 1

    return minimumCertainties
```

ตารางที่ ข.31 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_originalmacd.py ในโฟลเดอร์macdtesterF

```
from macdtesterF.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
from macdtesterF.macd import getMACDAndSignalLine
import numpy as np
from macdtesterF.data import getPricesData
import os
import pandas as pd

def isABuyPoint(macd, signalLine, t):
    if isAnUpCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def isASellPoint(macd, signalLine, t):
    if isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(macd, signalLine, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, signalLine, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(macd, signalLine, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
```

```
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def originalMACDTest(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, signalLine, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
    #print allBuyPoints
    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine):
```

```
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)
    #print allSellPoints
    return allSellPoints


def originalMACDTestWithSET100(verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        successRate, profitRate = originalMACDTest(pricesData, macd, signalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
```

```
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.32 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_macdr1.py ในโฟลเดอร์macdtesterF

```
from macdtesterF.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
from macdtesterF.macd import getMACDAndSignalLine
import numpy as np
import os
import pandas as pd
from macdtesterF.data import getPricesData

def isABuyPoint(macd, signalLine, t):
    i = macd.index.get_loc(t)
    if (i >=2 and
        macd.ix[i] > signalLine.ix[i] and
        macd.ix[i-1] > signalLine.ix[i-1] and
        isAnUpCrossingPoint(macd, signalLine, macd.index[i-2])):
        return True
    else:
        return False

def isASellPoint(pricesData, macd, signalLine, buyPoint, t):
    profitRate = (pricesData[t] - pricesData[buyPoint])/float(pricesData[buyPoint])
    if profitRate >= 0.03 or isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(pricesData, macd, signalLine, period):
```

```
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, signalLine, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(pricesData, macd, signalLine, buyPoint, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def MACDR1Test(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
```

```
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)

    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)

    return allSellPoints


def MACDR1TestWithSET100(verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
```

```
        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        successRate, profitRate = MACDR1Test(pricesData, macd, signalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.33 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_macdr2.py ในโฟลเดอร์macdtesterF

```
from macdtesterF.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
import numpy as np
import os
import pandas as pd
from macdtesterF.data import getPricesData
from macdtesterF.macd import getMACDAndSignalLine

def isABuyPoint(pricesData, macd, signalLine, t):
    i = macd.index.get_loc(t)
    if (i >=2 and
        macd.ix[i] > signalLine.ix[i] and
        macd.ix[i-1] > signalLine.ix[i-1] and
```

```
        isAnUpCrossingPoint(macd, signalLine, macd.index[i-2]) and
        (macd.ix[i] - signalLine.ix[i])/float(pricesData.ix[i]) >= 0.005):
        return True
    else:
        return False

def isASellPoint(pricesData, macd, signalLine, buyPoint, t):
    profitRate = (pricesData[t] - pricesData[buyPoint])/float(pricesData[buyPoint])
    if profitRate >= 0.03 or isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(pricesData, macd, signalLine, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(pricesData, macd, signalLine, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(pricesData, macd, signalLine, buyPoint, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def MACDR2Test(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
```

```
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)

    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)

    return allSellPoints
```

```
def MACDR2TestWithSET100(verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        successRate, profitRate = MACDR2Test(pricesData, macd, signalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.34 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_modifiedsignalline.py ในโฟลเดอร์ macdtesterF

```
from macdtesterF.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods, getUpCrossingPointInAPeriod
from macdtesterF.macd import getMACDAndSignalLine
import numpy as np
import os
import pandas as pd
from macdtesterF.data import getPricesData
from macdtesterF.modifiedsignalline import getModifiedSignalLine

def isABuyPoint(macd, modifiedSignalLine, upPoint, t):
    iup = macd.index.get_loc(upPoint)
    i = macd.index.get_loc(t)
    if i <= iup and isAnUpCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
        return False

def isASellPoint(macd, modifiedSignalLine, t):
    if isADownCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, modifiedSignalLine, upPoint, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(macd, modifiedSignalLine, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
```

```
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def modifiedSignalLineTest(pricesData, macd, signalLine, modifiedSignalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for period in periods:
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)

    return allBuyPoints
```

```
def getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)

    return allSellPoints

def modifiedSignalLineTestWithSET100(K=1, verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and f.split('.')[0]
not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
        modifiedSignalLine = getModifiedSignalLine(macd, signalLine, K)
        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        modifiedSignalLine = modifiedSignalLine[25:]
        successRate, profitRate = modifiedSignalLineTest(pricesData, macd, signalLine,
modifiedSignalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
```

```
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.35 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_modsigtradingweight.py ในโฟลเดอร์ macdtesterF

```
from macdtesterF.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods, getMaximumPointsInAPeriod,\
    getMinimumPointsInAPeriod, getUpCrossingPointInAPeriod
import numpy as np
from macdtesterF.modifiedsignalline import getAvailablyMinimumPointsInAPeriod,\
    getMinimumCertainties
from macdtesterF.fuzzy import FIS
import os
import pandas as pd
from macdtesterF.data import getPricesData
from macdtesterF.modifiedsignalline import getModifiedSignalLine
from macdtesterF.macd import getMACDAndSignalLine

def isABuyPoint(macd, modifiedSignalLine, upPoint, t):
    iup = macd.index.get_loc(upPoint)
    i = macd.index.get_loc(t)
    if i <= iup and isAnUpCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
        return False

def isASellPoint(macd, modifiedSignalLine, t):
```

```
    if isADownCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint, overBought,
overSold, avmin, dmin):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    weight = None
    for t in period:
        if buyPoint is None and isABuyPoint(macd, modifiedSignalLine, upPoint, t):
            buyPoint = t
            weight = fuzzyInfer(overBought, overSold, macd[avmin[t]], dmin[t])

        if buyPoint is not None and isASellPoint(macd, modifiedSignalLine, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital, weight))
            buyPoint = None
            sellPoint = None
            weight = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def modsigTradingWeightTest(pricesData, macd, signalLine, modifiedSignalLine, R):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for i in xrange(len(periods)):
        if i-R < 0: continue
```

```
        period = periods[i]
        avmin = getAvailablyMinimumPointsInAPeriod(macd, signalLine, period)
        dmin = getMinimumCertainties(macd, signalLine, period)
        overBought = getOverBought(macd, signalLine, periods, i, R)
        overSold = getOverSold(macd, signalLine, periods, i, R)
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint,
overBought, overSold, avmin, dmin)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine, R):
    from tradingtest_modifiedsignalline import getTotalTradesInAPeriod as
modsig_getTotalTradesInAPeriod
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []

    for i in xrange(len(periods)):
        if i-R < 0: continue

        period = periods[i]
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = modsig_getTotalTradesInAPeriod(macd, modifiedSignalLine, period,
upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
```

```
    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine, R):
    from tradingtest_modifiedsignalline import getTotalTradesInAPeriod as
modsig_getTotalTradesInAPeriod
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []

    for i in xrange(len(periods)):
        if i-R < 0: continue

        period = periods[i]
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = modsig_getTotalTradesInAPeriod(macd, modifiedSignalLine, period,
upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)

    return allSellPoints

def getOverBought(macd, signalLine, periods, i, R):
    if i-R < 0:
        raise Exception('R is too much.')

    maxPoint = getMaximumPointsInAPeriod(macd, signalLine, periods[i-R])
    overBought = macd[maxPoint]
    return overBought

def getOverSold(macd, signalLine, periods, i, R):
    if i-R < 0:
        raise Exception('R is too much.')

    minPoint = getMinimumPointsInAPeriod(macd, signalLine, periods[i-R])
    overSold = macd[minPoint]
    return overSold
```

```
fis = FIS()
MaxDefuzz = fis.outputs['z'].defuzzify(fis.outputs['z'].terms['TRADE-CONFIDENCE'])
MinDefuzz = fis.outputs['z'].defuzzify(fis.outputs['z'].terms['TRADE-DOUBT'])
def fuzzyInfer(overBought, overSold, macd_avmin, dmin):
    half = (overBought + overSold)/2.0
    oneForth = (overSold + half)/2.0
    threeForth = (overBought + half)/2.0
    fis.set_term_param('macd_avmin', 'MACD-HIGH', abcd=(half, overBought, '', ''))
    fis.set_term_param('macd_avmin', 'MACD-LOW', abcd=('', '', overSold, half))
    fis.set_term_param('macd_avmin', 'MACD-MEDIUM', abcd=(oneForth, half, half,
threeForth))

    fis.set_inputs(macd_avmin=macd_avmin, dmin=dmin)
    z = fis.evaluate().get_outputs()['z'].value
    w = (z-MinDefuzz)/float(MaxDefuzz-MinDefuzz)

    return w

def modsigTradingWeightTestWithSET100(K=1, R=1, verbose=False):
    directory = './quoteName--AGRI/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, int(paraMACDShort),
int(paraMACDLong), int(parasignalShort))
        modifiedSignalLine = getModifiedSignalLine(macd, signalLine, K)
        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        modifiedSignalLine = modifiedSignalLine[25:]
        successRate, profitRate = modsigTradingWeightTest(pricesData, macd, signalLine,
modifiedSignalLine, R)
```

```
        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.36 ตารางแสดงรหัสต้นฉบับในไฟล์ mainmacdf.py ในโฟลเดอร์macdtesterF

```
from macdtesterF.data import getPricesData
from macdtesterF.tradingtest_originalmacd import originalMACDTest, getAllBuyPoints as
org_getAllBuyPoints,\
    getAllSellPoints as org_getAllSellPoints, originalMACDTestWithSET100
from macdtesterF.macd import getMACDAndSignalLine
from macdtesterF.modifiedsignalline import getModifiedSignalLine
from macdtesterF.plot import standardPlot, addMarkers, addModifiedSignalLine
from macdtesterF.tradingtest_macdr1 import MACDR1Test, getAllBuyPoints as
macdr1_getAllBuyPoints,\
    getAllSellPoints as macdr1_getAllSellPoints, MACDR1TestWithSET100
from macdtesterF.tradingtest_macdr2 import MACDR2Test, getAllBuyPoints as
macdr2_getAllBuyPoints,\
    getAllSellPoints as macdr2_getAllSellPoints, MACDR2TestWithSET100
from macdtesterF.tradingtest_modifiedsignalline import modifiedSignalLineTest,
getAllBuyPoints as modsig_getAllBuyPoints,\
    getAllSellPoints as modsig_getAllSellPoints,\
    modifiedSignalLineTestWithSET100
from macdtesterF.basic import getTradingPeriods, getMaximumPointsInAPeriod,\
    getMinimumPointsInAPeriod
```

```
from macdtesterF.tradingtest_modsigtradingweight import modsigTradingWeightTest,
getAllBuyPoints as modsigw_getAllBuyPoints,\
    getAllSellPoints as modsigw_getAllSellPoints,\
    modsigTradingWeightTestWithSET100

## prepare data for a single data
print 'preparing data...'
symbol = 'KBANK'
pricesData = getPricesData(symbol)
macd, signalLine = getMACDAndSignalLine(pricesData, 12, 26, 9)
modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)

pricesData = pricesData[25:]
macd = macd[25:]
signalLine = signalLine[25:]
modifiedSignalLine = modifiedSignalLine[25:]

## original MACD test with a single data
print 'original MACD test for ' + symbol
successRate, profitRate = originalMACDTest(pricesData, macd, signalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = org_getAllBuyPoints(pricesData, macd, signalLine)
allSellPoints = org_getAllSellPoints(pricesData, macd, signalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## MACDR1 test with a single data
print 'MACDR1 test for ' + symbol
successRate, profitRate = MACDR1Test(pricesData, macd, signalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")
```

```
allBuyPoints = macdr1_getAllBuyPoints(pricesData, macd, signalLine)
allSellPoints = macdr1_getAllSellPoints(pricesData, macd, signalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## MACDR2 test with a single data
print 'MACDR2 test for ' + symbol
successRate, profitRate = MACDR2Test(pricesData, macd, signalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = macdr2_getAllBuyPoints(pricesData, macd, signalLine)
allSellPoints = macdr2_getAllSellPoints(pricesData, macd, signalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## modified signal line test with a single data
print 'modified signal line test for ' + symbol
successRate, profitRate = modifiedSignalLineTest(pricesData, macd, signalLine,
modifiedSignalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = modsig_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine)
allSellPoints = modsig_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addModifiedSignalLine(fig, modifiedSignalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## modified signal line with trading weight test with a single data
```

```
print 'modified signal line with trading weight test for ' + symbol
successRate, profitRate = modsigTradingWeightTest(pricesData, macd, signalLine,
modifiedSignalLine, 1)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = modsigw_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)
allSellPoints = modsigw_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addModifiedSignalLine(fig, modifiedSignalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)



## original MACD test in SET-100
print 'original MACD test in SET-100'
successRate, profitRate = originalMACDTestWithSET100(verbose=True)
print 'original MACD test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## MACDR1 test in SET-100
print 'MACDR1 test in SET-100'
successRate, profitRate = MACDR1TestWithSET100(verbose=True)
print 'MACDR1 test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## MACDR2 test in SET-100
print 'MACDR2 test in SET-100'
successRate, profitRate = MACDR2TestWithSET100(verbose=True)
```

```
print 'MACDR2 test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## modified signal line test in SET-100
print 'modified signal line test in SET-100'
successRate, profitRate = modifiedSignalLineTestWithSET100(K=0.1, verbose=True)
print 'modified signal line test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## modified signal line with trading weight test in SET-100
print 'modified signal line with trading weight test in SET-100'
successRate, profitRate = modsigTradingWeightTestWithSET100(K=0.1, R=1, verbose=True)
print 'modified signal line with trading weight test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
```

ตารางที่ ข.37 ตารางแสดงรหัสต้นฉบับในไฟล์ basic.py ในโฟลเดอร์macdtesterP

```
import numpy as np

def getTradingPeriods(macd, signalLine):
    downCrossingPoints = getDownCrossingPoints(macd, signalLine)
    periods = []
    for i in xrange(len(downCrossingPoints)-1):
        periods.append(macd[downCrossingPoints[i]:downCrossingPoints[i+1]].index)

    return periods

def isAnUpCrossingPoint(macd, signalLine, t):
    i = macd.index.get_loc(t)

    if i >= 1 and (macd.ix[i] > signalLine.ix[i]) and (macd.ix[i-1] < signalLine.ix[i-1]):
        return True
    elif i >= 2 and (macd.ix[i] > signalLine.ix[i]) and (macd.ix[i-1] == signalLine.ix[i-1]) and
```

```
(macd.ix[i-2] < signalLine.ix[i-2]):
        return True
    else:
        False

def isADownCrossingPoint(macd, signalLine, t):
    i = macd.index.get_loc(t)

    if i >= 1 and (macd.ix[i] < signalLine.ix[i]) and (macd.ix[i-1] > signalLine.ix[i-1]):
        return True
    elif i >= 2 and (macd.ix[i] < signalLine.ix[i]) and (macd.ix[i-1] == signalLine.ix[i-1]) and
(macd.ix[i-2] > signalLine.ix[i-2]):
        return True
    else:
        False

def getUpCrossingPoints(macd, signalLine):
    upCrossingPointList = []
    firstDownCrossingExists = False
    for t in macd.index:
        if not firstDownCrossingExists and isADownCrossingPoint(macd, signalLine, t):
            firstDownCrossingExists = True

        if firstDownCrossingExists and isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPointList.append(t)

    return upCrossingPointList

def getDownCrossingPoints(macd, signalLine):
    downCrossingPointList = []
    for t in macd.index:
        if isADownCrossingPoint(macd, signalLine, t):
            downCrossingPointList.append(t)

    return downCrossingPointList

class Trade(object):
    def __init__(self, buyPoint, sellPoint, capital, weight=1):
```

```
        self.buyPoint = buyPoint
        self.sellPoint = sellPoint
        self.capital = capital
        self.weight = weight

    def getProfit(self, pricesData):
        return ((pricesData[self.sellPoint] -
pricesData[self.buyPoint])/float(pricesData[self.buyPoint]))*self.capital*self.weight

class TotalTrade(object):
    def __init__(self, tradeList):
        self.initialCapital = tradeList[0].capital
        for trade in tradeList:
            if trade.capital != self.initialCapital:
                raise Exception('All trade in tradeList must have the same capital.')

        self.tradeList = tradeList

    def getProfitRate(self, pricesData):
        if len(self.tradeList) == 1:
            profitRate = self.tradeList[0].getProfit(pricesData)/float(self.initialCapital)
        else:
            profits = np.array([trade.getProfit(pricesData) for trade in self.tradeList])
            profitRate = np.sum(profits)/float(self.initialCapital - np.sum(profits[profits < 0]))

        return profitRate

def getUpCrossingPointInAPeriod(macd, signalLine, period):
    upCrossingPoint = None
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t
            break

    return upCrossingPoint

def getMaximumPointsInAPeriod(macd, signalLine, period):
    upCrossingPoint = None
```

```
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t
            break

    maximumPoint = macd[upCrossingPoint:period[-1]].idxmax()

    return maximumPoint

def getMinimumPointsInAPeriod(macd, signalLine, period):
    upCrossingPoint = None
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t
            break

    minimumPoint = macd[period[0]:upCrossingPoint].idxmin()

    return minimumPoint
```

ตารางที่ ข.38 ตารางแสดงรหัสต้นฉบับในไฟล์ Mainmacdp.py ในโฟลเดอร์macdtesterP

```
from macdtesterP.data import getPricesData
from macdtesterP.tradingtest_originalmacd import originalMACDTest, getAllBuyPoints as
org_getAllBuyPoints,\
    getAllSellPoints as org_getAllSellPoints, originalMACDTestWithSET100
from macdtesterP.macd import getMACDAndSignalLine
from macdtesterP.modifiedsignalline import getModifiedSignalLine
from macdtesterP.plot import standardPlot, addMarkers, addModifiedSignalLine
from macdtesterP.tradingtest_macdr1 import MACDR1Test, getAllBuyPoints as
macdr1_getAllBuyPoints,\
    getAllSellPoints as macdr1_getAllSellPoints, MACDR1TestWithSET100
from macdtesterP.tradingtest_macdr2 import MACDR2Test, getAllBuyPoints as
macdr2_getAllBuyPoints,\
    getAllSellPoints as macdr2_getAllSellPoints, MACDR2TestWithSET100
from macdtesterP.tradingtest_modifiedsignalline import modifiedSignalLineTest,
getAllBuyPoints as modsig_getAllBuyPoints,\
```

```
    getAllSellPoints as modsig_getAllSellPoints,\
    modifiedSignalLineTestWithSET100
from macdtesterP.basic import getTradingPeriods, getMaximumPointsInAPeriod,\
    getMinimumPointsInAPeriod
from macdtesterP.tradingtest_modsigtradingweight import modsigTradingWeightTest,
getAllBuyPoints as modsigw_getAllBuyPoints,\
    getAllSellPoints as modsigw_getAllSellPoints,\
    modsigTradingWeightTestWithSET100

## prepare data for a single data
print 'preparing data...'
symbol = 'KBANK'
pricesData = getPricesData(symbol)
macd, signalLine = getMACDAndSignalLine(pricesData, 12, 26, 9)
modifiedSignalLine = getModifiedSignalLine(macd, signalLine, 1)

pricesData = pricesData[25:]
macd = macd[25:]
signalLine = signalLine[25:]
modifiedSignalLine = modifiedSignalLine[25:]

## original MACD test with a single data
print 'original MACD test for ' + symbol
successRate, profitRate = originalMACDTest(pricesData, macd, signalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = org_getAllBuyPoints(pricesData, macd, signalLine)
allSellPoints = org_getAllSellPoints(pricesData, macd, signalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## MACDR1 test with a single data
print 'MACDR1 test for ' + symbol
successRate, profitRate = MACDR1Test(pricesData, macd, signalLine)
```

```
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = macdr1_getAllBuyPoints(pricesData, macd, signalLine)
allSellPoints = macdr1_getAllSellPoints(pricesData, macd, signalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## MACDR2 test with a single data
print 'MACDR2 test for ' + symbol
successRate, profitRate = MACDR2Test(pricesData, macd, signalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = macdr2_getAllBuyPoints(pricesData, macd, signalLine)
allSellPoints = macdr2_getAllSellPoints(pricesData, macd, signalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## modified signal line test with a single data
print 'modified signal line test for ' + symbol
successRate, profitRate = modifiedSignalLineTest(pricesData, macd, signalLine,
modifiedSignalLine)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = modsig_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine)
allSellPoints = modsig_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addModifiedSignalLine(fig, modifiedSignalLine, show=False)
```

```
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)

## modified signal line with trading weight test with a single data
print 'modified signal line with trading weight test for ' + symbol
successRate, profitRate = modsigTradingWeightTest(pricesData, macd, signalLine,
modifiedSignalLine, 1)
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

allBuyPoints = modsigw_getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)
allSellPoints = modsigw_getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine,
1)

fig = standardPlot(pricesData, macd, signalLine, show=False)
addModifiedSignalLine(fig, modifiedSignalLine, show=False)
addMarkers(fig, macd, allBuyPoints, color='#00CC00', show=False)
addMarkers(fig, macd, allSellPoints, color='#FF0000', show=True)



## original MACD test in SET-100
print 'original MACD test in SET-100'
successRate, profitRate = originalMACDTestWithSET100(verbose=True)
print 'original MACD test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## MACDR1 test in SET-100
print 'MACDR1 test in SET-100'
successRate, profitRate = MACDR1TestWithSET100(verbose=True)
print 'MACDR1 test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")
```

```
## MACDR2 test in SET-100
print 'MACDR2 test in SET-100'
successRate, profitRate = MACDR2TestWithSET100(verbose=True)
print 'MACDR2 test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## modified signal line test in SET-100
print 'modified signal line test in SET-100'
successRate, profitRate = modifiedSignalLineTestWithSET100(K=0.1, verbose=True)
print 'modified signal line test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
raw_input("Press Enter to continue...")

## modified signal line with trading weight test in SET-100
print 'modified signal line with trading weight test in SET-100'
successRate, profitRate = modsigTradingWeightTestWithSET100(K=0.1, R=1, verbose=True)
print 'modified signal line with trading weight test in SET-100'
print 'success rate: %f' % successRate
print 'profit rate: %f' % profitRate
```

ตารางที่ ข.39 ตารางแสดงรหัสต้นฉบับในไฟล์ data.py ในโฟลเดอร์macdtesterP

```
import pandas as pd

class DataDefinition (str):
    def __new__(cls, name, colNumber, dtype):
        return str.__new__(cls, name)

    def __init__(self, name, colNumber, dtype):
        super(DataDefinition, self).__init__(name)
        self.colNumber = colNumber
        self.dtype = dtype
```

```
DATE = DataDefinition('date', 0, 'S8')
OPEN = DataDefinition('open', 1, 'f8')
HIGH = DataDefinition('high', 2, 'f8')
LOW = DataDefinition('low', 3, 'f8')
CLOSE = DataDefinition('close', 4, 'f8')

def _getRawData(uri="./example/ex.csv"):
    data = pd.read_csv(uri, dtype={DATE: DATE.dtype,
                        OPEN: OPEN.dtype, HIGH: HIGH.dtype,
                        LOW: LOW.dtype, CLOSE: CLOSE.dtype,
                        }, parse_dates=[0], index_col=0, na_values='-')

    return data.sort_index(ascending=True).dropna()

def getPricesData(symbol='KBANK', type='close'):
    if type not in [OPEN, HIGH, LOW, CLOSE]:
        raise Exception('param-value error: type=' + type +'". Hint, type=x for x in {"open",
"high", "low", "close"}')
    uri = "./example/ex.csv"

    rawData = None
    try:
        rawData = _getRawData(uri)
    except:
        raise Exception('error occur while parsing raw data. Hint, please check if symbol=' +
symbol + ' exists and matches the name of csv file.')

    data = rawData[type]
    return data
```

ตารางที่ ข.40 ตารางแสดงรหัสต้นฉบับในไฟล์ ema.py ในโฟลเดอร์macdtesterP

```
import pandas as pd
import pandas.stats.moments as mt

def getEMA(pricesData, n=12):
```

```
    ema = pd.Series(mt.ewma(pricesData, span=n), index=pricesData.index)
    return ema
```

ตารางที่ ข.41 ตารางแสดงรหัสต้นฉบับในไฟล์ Fuzzy.py ในโฟลเดอร์macdtesterP

```
import numpy as np
import matplotlib.pyplot as plt

class Universe(object):
    def __init__(self, lower, upper):
        assert lower <= upper
        self.lower = lower
        self.upper = upper

    def contains(self, value):
        return self.lower <= value and value <= self.upper

    def contains_all(self, values):
        values = np.array(values)
        return (self.lower <= values).all() and (values <= self.upper).all()

    def sample(self, step=None):
        if step is None:
            step = 0.01 if self.upper - self.lower <= 10 else 0.1

        return np.arange(self.lower, self.upper, step)

    def __eq__(self, other):
        return other is not None and isinstance(other, Universe) and other.lower ==
self.lower and other.upper == self.upper

class Variable(object):
    def __init__(self, universe):
        assert isinstance(universe, Universe)
        self.universe = universe
        self.value = None
        self.terms = dict()
        self.abcd_for_term = dict()
```

```
    def set_value(self, value):
        assert self.universe.contains(value) or value is None
        self.value = value

    def set_term(self, term_name, abcd=(None,None,None,None), description=''):
        assert len(abcd) == 4
        if self.abcd_for_term.has_key(term_name):
            for i in xrange(len(abcd)):
                if abcd[i] is not None and abcd[i] != '':
                    self.abcd_for_term[term_name][i] = abcd[i]
        else:
            self.abcd_for_term[term_name] = list(abcd)

        if all((param is not None and param != '') for param in
self.abcd_for_term[term_name]):
            self.terms[term_name] = FuzzySet(self.universe, self.abcd_for_term[term_name],
description=description)
        else:
            self.terms[term_name] = None

    def fuzzify(self, term_name):
        if self.terms[term_name] is None:
            print [type(param) for param in self.abcd_for_term[term_name]]
            assert all((type(param) in [int, float]) for param in self.abcd_for_term[term_name])
            raise Exception("Some parameters have not been set for term: %s, params: %s" %
(term_name, str(self.abcd_for_term[term_name])))

        return self.terms[term_name](self.value)

    def defuzzify(self, accum_mf):
        outs = self.universe.sample()
        weights = accum_mf(outs)
        sum_weights = weights.sum()
        out = (outs * weights).sum()/sum_weights if sum_weights != 0 else None
        self.set_value(out)

        return out
```

```
    def displayterm(self, term_name, xlim):
        if self.terms[term_name] is None:
            raise Exception("Some parameters have not been set for term: %s, params: %s" %
(term_name, str(self.abcd_for_term[term_name])))

        fig = plt.figure(facecolor='white')
        ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
        ax.set_xlim(xlim)
        ax.set_ylim(0, 1.2)

        x = self.universe.sample(step=0.005)
        y = self.terms[term_name](x)

        ax.plot(x,y,lw=3)
        ax.set_title(term_name, fontsize=20)
        plt.show()

class FuzzySet(object):

    def __init__(self, universe, abcd, mf=None, description=''):
        assert isinstance(universe, Universe)
        assert (abcd is not None and mf is None) or (abcd is None and mf is not None)

        if abcd is not None:
            assert len(abcd) == 4 and universe.contains_all(abcd)
            self.mf = FuzzySet.fuzzy_number_mf(*abcd)
        else:
            self.mf = mf

        self.universe = universe
        self.description = description


    def __call__(self, values):
        assert self.universe.contains_all(values)
        return self.mf(values)
```

```
    def __and__(self, other):
        assert isinstance(other, (int, float, FuzzySet))
        if isinstance(other, FuzzySet):
            assert self.universe == other.universe
            def new_mf(x):
                return FuzzyOperation.AND(self(x), other(x))
        else:
            def new_mf(x):
                return FuzzyOperation.AND(self(x), other)

        return FuzzySet(self.universe, None, mf=new_mf)

    def __or__(self, other):
        assert isinstance(other, (int, float, FuzzySet))
        if isinstance(other, FuzzySet):
            assert self.universe == other.universe
            def new_mf(x):
                return FuzzyOperation.OR(self(x), other(x))
        else:
            def new_mf(x):
                return FuzzyOperation.OR(self(x), other)

        return FuzzySet(self.universe, None, mf=new_mf)

    class fuzzy_number_mf(object):
        def __init__(self, a, b, c, d):
            #assert a <= b and b <= c and c <= d
            if not (a <= b and b <= c and c <= d): raise Exception, 'not! (%f <= %f <= %f <=
%f)' % (a, b, c, d)


            def mf(x):
                if x < a: return 0
                elif a <= x and x < b: return (x-a)/float(b-a)
                elif b <= x and x < c: return 1
                elif c <= x and x < d: return (d-x)/float(d-c)
                elif x == d and d == c: return 1
                else: return 0
```

```
            self.func = np.frompyfunc(mf, 1, 1)

        def __call__(self, x):
            return self.func(x)

class FuzzyOperation(object):

    @classmethod
    def define_operation_and(cls, operation):
        assert isinstance(operation, FuzzyOperation.Interface)
        cls.AND = operation

    @classmethod
    def define_operation_or(cls, operation):
        assert isinstance(operation, FuzzyOperation.Interface)
        cls.OR = operation

    class Interface(object):
        def __call__(self, a, b): raise NotImplementedError

    class MAX(Interface):
        def __call__(self, a, b):
            return np.maximum(a, b)

    class MIN(Interface):
        def __call__(self, a, b):
            return np.minimum(a, b)

    class PROD(Interface):
        def __call__(self, a, b):
            return a * b

    class SUM(Interface):
        def __call__(self, a, b):
            return a + b

    AND = MIN()
```

```
    OR = MAX()

class FuzzyRule(object):
    def __init__(self, antecedence, consequences):
        self.antecedence = antecedence
        self.consequences = consequences

    def evaluate(self, inputs, outputs):
        '''

        :param variable
        :return a dictionary of the form {'z1': activ_mf_z1, 'z2': activ_mf_z2, ...}
            where zi is the output-variable's name and activ_mf_zi is the corresponding result
membership function from this rule's evaluation.
        '''

        w = self._aggregate(inputs)
        activ_mf = self._activate(w, outputs)
        return activ_mf

    def _aggregate(self, inputs):

        stack = []
        for element in self.antecedence:
            if isinstance(element, FuzzyOperation.Interface):
                operation = element
                token1 = stack.pop()
                fuzzified_value1 = token1 if not isinstance(token1, tuple) else
inputs[token1[0]].fuzzify(token1[1])

                token2 = stack.pop()
                fuzzified_value2 = token2 if not isinstance(token2, tuple) else
inputs[token2[0]].fuzzify(token2[1])

                result_token = operation(fuzzified_value1, fuzzified_value2)
                stack.append(result_token)
            else:
                stack.append(element)
```

```
        if len(stack) != 1: print stack
        assert len(stack) == 1

        token = stack.pop()
        w = token if not isinstance(token, tuple) else inputs[token[0]].fuzzify(token[1])

        return w

    def _activate(self, w, outputs):
        activ_mf = dict()
        for element in self.consequences:
            variable_name, term_name = element
            activ_mf[variable_name] = outputs[variable_name].terms[term_name] & w

        return activ_mf

class FIS(object):
    def __init__(self):
        self._define(self.fuzzydefinition)

    def get_inputs(self):
        return self.inputs

    def get_outputs(self):
        return self.outputs

    def set_inputs(self, **kwargs):
        assert set(kwargs.keys()) == set(self.inputs.keys())

        for key, value in kwargs.iteritems():
            assert self.inputs[key].universe.contains(value)
            self.inputs[key].set_value(value)

        return self

    def set_term_param(self, variable_name, term_name, abcd):
        self.inputs[variable_name].set_term(term_name, abcd=abcd)
```

```
        return self

    def evaluate(self):
        accum_mf = dict()
        for output_variable_name in self.outputs:
            accum_mf[output_variable_name] = None

        for rule in self.fuzzy_rules:
            for output_variable_name in self.outputs:
                activ_mf = rule.evaluate(self.inputs, self.outputs)
                if output_variable_name in activ_mf.keys():
                    accum_mf[output_variable_name] = (activ_mf[output_variable_name]
                                                      if accum_mf[output_variable_name] is None
                                                      else accum_mf[output_variable_name] |
activ_mf[output_variable_name])

        for output_variable_name, output_variable in self.outputs.iteritems():
            output_variable.defuzzify(accum_mf[output_variable_name])

        return self

    def _define(self, definition):
        properties = {'inputs': None, 'outputs': None, 'fuzzy_rules': None}
        definition(properties)
        assert properties['inputs'] is not None
        assert properties['outputs'] is not None
        assert properties['fuzzy_rules'] is not None
        self.inputs = properties['inputs']
        self.fuzzy_rules = properties['fuzzy_rules']
        self.outputs = properties['outputs']

        return self

    def fuzzydefinition(self, properties):
        REAL = Universe(-999, 999)
        NORM = Universe(0, 1)
```

```
        #inputs
        properties['inputs'] = {'dmin': Variable(NORM),
                                'macd_avmin': Variable(REAL)}

        #outputs
        properties['outputs'] = {'z': Variable(NORM)}

        #input_terms
        properties['inputs']['dmin'].set_term('MIN-CONFIDENCE', abcd=(0, 1, 1, 1))

        properties['inputs']['macd_avmin'].set_term('MACD-HIGH', abcd=(None, None,
REAL.upper, REAL.upper))
        properties['inputs']['macd_avmin'].set_term('MACD-LOW', abcd=(REAL.lower,
REAL.lower, None, None))
        properties['inputs']['macd_avmin'].set_term('MACD-MEDIUM', abcd=(None, None,
None, None))


        #output_terms
        properties['outputs']['z'].set_term('TRADE-CONFIDENCE', abcd=(0, 1, 1, 1))
        properties['outputs']['z'].set_term('TRADE-DOUBT', abcd=(0, 0, 0, 1))


        #fuzzy_rules
        AND = FuzzyOperation.MIN()
        OR = FuzzyOperation.MAX()
        properties['fuzzy_rules'] = [
                    FuzzyRule(antecedence = (('dmin', 'MIN-CONFIDENCE'),),
                        consequences = (('z', 'TRADE-CONFIDENCE'),)),

                    FuzzyRule(antecedence = (('macd_avmin', 'MACD-LOW'),),
                        consequences = (('z', 'TRADE-CONFIDENCE'),)),

                    FuzzyRule(antecedence = (('macd_avmin', 'MACD-HIGH'), ('macd_avmin',
'MACD-MEDIUM'), OR),
                        consequences = (('z', 'TRADE-DOUBT'),))

            ]
```

ตารางที่ ข.42ตารางแสดงรหัสต้นฉบับในไฟล์ macd.py ในโฟลเดอร์macdtesterP

```
import pandas as pd
import pandas.stats.moments as mt
from ema import getEMA

def getMACDAndSignalLine(pricesData, n1, n2, n3):
    ema_n1 = getEMA(pricesData, n1)
    ema_n2 = getEMA(pricesData, n2)
    macd = ema_n1 - ema_n2
    signalLine = pd.Series(mt.ewma(macd, span=n3), index=pricesData.index)
    return macd, signalLine
```

ตารางที่ ข.43 ตารางแสดงรหัสต้นฉบับในไฟล์ plot.py ในโฟลเดอร์macdtesterP

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import ticker, font_manager

class IndexManager(object):
    def __init__(self, indexSeries):
        self.index = indexSeries
        self.evenlyIndex = pd.Series(np.arange(len(indexSeries)), index=indexSeries)

    def evenlyIndexToDateTimeIndex(self, x):
        try:
            return self.index[x]
        except:
            return None

    def datetimeIndexToEvenlyIndex(self, d):
        try:
            return self.evenlyIndex[d]
        except:
            return None
```

```
    def dateMappingFormatter(self, x, pos=None):
        d = self.evenlyIndexToDateTimeIndex(int(x + 0.5))
        if d is not None:
            return d.strftime('%b\n%Y')
        else:
            return ''

def standardPlot(pricesData, macd, signalLine, show=True):
    idm = IndexManager(pricesData.index)
    plt.rc('axes', grid=True)
    plt.rc('grid', color='0.75', linestyle='-', linewidth=0.5)

    left, width = 0.1, 0.8
    rect1 = [left, 0.6, width, 0.3]
    rect2 = [left, 0.1, width, 0.5]


    fig = plt.figure(facecolor='white')
    axescolor  = '#f9f9f9'  # the axes background color
    ax1 = fig.add_axes(rect1, axisbg=axescolor)  #left, bottom, width, height
    ax2 = fig.add_axes(rect2, axisbg=axescolor, sharex=ax1)


    #we don't want to label all indices, just the first day in each month are enough
    #this function return the list of the (evenly) indices of the first day in each month
    def getIndexOfFirstDayEachMonth(datearr):
        dateSeries = pd.Series(datearr.month)
        isFirstDay = np.ones_like(dateSeries, dtype=np.bool)
        isFirstDay[1:] = (dateSeries[1:] != dateSeries[:-1])
        return dateSeries.index[isFirstDay]

    #we then use the above mentioned list as a 'FixedLocator' and set it as the major locator

ax1.xaxis.set_major_locator(ticker.FixedLocator(getIndexOfFirstDayEachMonth(pricesData.inde
x)))


    #only the indices in the locator will be labeled, the plotter will ask the 'formatter' which
```

```
labels corresponds to which indices
    #this is the place where our mapping is used
    ax1.xaxis.set_major_formatter(ticker.FuncFormatter(idm.dateMappingFormatter))

    #set nice left and right spaces
    ax1.set_xlim(idm.evenlyIndex[0] - 2, idm.evenlyIndex[-1] + 2)

    ax1.plot(idm.evenlyIndex, pricesData, color='black', lw=2, label='closing price')
    ax2.plot(idm.evenlyIndex, macd, color='red', lw=1.5, label='MACD Line')
    ax2.plot(idm.evenlyIndex, signalLine, color='blue', lw=1, label='Signal Line')

    for label in ax1.get_xticklabels():
        label.set_visible(False)

    ax1.yaxis.set_major_locator(ticker.MaxNLocator(5, prune='both'))
    ax2.yaxis.set_major_locator(ticker.MaxNLocator(5, prune='both'))

    props = font_manager.FontProperties(size=10)
    leg1 = ax1.legend(loc='best', shadow=True, fancybox=True, prop=props, scatterpoints=1,
markerscale=1)
    leg1.get_frame().set_alpha(0.5)
    leg2 = ax2.legend(loc='best', shadow=True, fancybox=True, prop=props, scatterpoints=1,
markerscale=1)
    leg2.get_frame().set_alpha(0.5)
    if show: plt.show()
    return fig

def addModifiedSignalLine(fig, modifiedSignalLine, color='violet', show=True):
    idm = IndexManager(modifiedSignalLine.index)
    ax2 = fig.get_axes()[1]
    ax2.plot(idm.evenlyIndex, modifiedSignalLine, color=color, lw=1, label='Modified Signal
Line')
    if show: plt.show()
    return fig

def addMarkers(fig, macd, tList, markerSize=40, color='#00CC00', show=True):
    idm = IndexManager(macd.index)
    ax2 = fig.get_axes()[1]
```

```
    ax2.scatter(idm.evenlyIndex[tList], macd[tList], color=color, s=markerSize)
    if show: plt.show()
    return fig
```

ตารางที่ ข.44 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_originalmacd.py ในโฟลเดอร์macdtesterP

```
from macdtesterP.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
from macdtesterP.macd import getMACDAndSignalLine
import numpy as np
from macdtesterP.data import getPricesData
import os
import pandas as pd

def isABuyPoint(macd, signalLine, t):
    if isAnUpCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def isASellPoint(macd, signalLine, t):
    if isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(macd, signalLine, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, signalLine, t):
            buyPoint = t
```

```
            if buyPoint is not None and isASellPoint(macd, signalLine, t):
                sellPoint = t

            if sellPoint is not None:
                tradeList.append(Trade(buyPoint, sellPoint, capital))
                buyPoint = None
                sellPoint = None

        if len(tradeList) > 0:
            return TotalTrade(tradeList)
        else:
            return None

def originalMACDTest(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, signalLine, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, signalLine, period)
        if totalTrade is not None:
```

```
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)

    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)

    return allSellPoints


def originalMACDTestWithSET100(verbose=False):
    directory = './resources/stockdata/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, 12, 26, 9)

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        successRate, profitRate = originalMACDTest(pricesData, macd, signalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate
```

```
    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.45 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_macdr1.py ในโฟลเดอร์macdtesterP

```
from macdtesterP.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
from macdtesterP.macd import getMACDAndSignalLine
import numpy as np
import os
import pandas as pd
from macdtesterP.data import getPricesData

def isABuyPoint(macd, signalLine, t):
    i = macd.index.get_loc(t)
    if (i >=2 and
```

```
        macd.ix[i] > signalLine.ix[i] and
        macd.ix[i-1] > signalLine.ix[i-1] and
        isAnUpCrossingPoint(macd, signalLine, macd.index[i-2])):
        return True
    else:
        return False

def isASellPoint(pricesData, macd, signalLine, buyPoint, t):
    profitRate = (pricesData[t] - pricesData[buyPoint])/float(pricesData[buyPoint])
    if profitRate >= 0.03 or isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(pricesData, macd, signalLine, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, signalLine, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(pricesData, macd, signalLine, buyPoint, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def MACDR1Test(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
```

```
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)

    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)

    return allSellPoints
```

```
def MACDR1TestWithSET100(verbose=False):
    directory = './resources/stockdata/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, 12, 26, 9)

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        successRate, profitRate = MACDR1Test(pricesData, macd, signalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.46 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_macdr2.py ในโฟลเดอร์macdtesterP

```
from macdtesterP.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods
import numpy as np
import os
import pandas as pd
from macdtesterP.data import getPricesData
from macdtesterP.macd import getMACDAndSignalLine

def isABuyPoint(pricesData, macd, signalLine, t):
    i = macd.index.get_loc(t)
    if (i >=2 and
        macd.ix[i] > signalLine.ix[i] and
        macd.ix[i-1] > signalLine.ix[i-1] and
        isAnUpCrossingPoint(macd, signalLine, macd.index[i-2]) and
        (macd.ix[i] - signalLine.ix[i])/float(pricesData.ix[i]) >= 0.005):
        return True
    else:
        return False

def isASellPoint(pricesData, macd, signalLine, buyPoint, t):
    profitRate = (pricesData[t] - pricesData[buyPoint])/float(pricesData[buyPoint])
    if profitRate >= 0.03 or isADownCrossingPoint(macd, signalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(pricesData, macd, signalLine, period):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(pricesData, macd, signalLine, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(pricesData, macd, signalLine, buyPoint, t):
            sellPoint = t
```

```
        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def MACDR2Test(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)
```

```
    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        totalTrade = getTotalTradesInAPeriod(pricesData, macd, signalLine, period)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)

    return allSellPoints


def MACDR2TestWithSET100(verbose=False):
    directory = './resources/stockdata/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, 12, 26, 9)

        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        successRate, profitRate = MACDR2Test(pricesData, macd, signalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
```

```
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.47 ตารางแสดงรหัสต้นฉบับในไฟล์ Modifiedsignalline.py ในโฟลเดอร์macdtesterP

```
import pandas as pd
from macdtesterP.basic import getTradingPeriods, isAnUpCrossingPoint


def getModifiedSignalLine(macd, signalLine, K):
    periods = getTradingPeriods(macd, signalLine)
    modifiedSignalLine = pd.Series(float('nan'), index=signalLine.index)

    for period in periods:
        avmin = getAvailablyMinimumPointsInAPeriod(macd, signalLine, period)
        avmax = getAvailablyMaximumPointsInAPeriod(macd, signalLine, period)
        dmin = getMinimumCertainties(macd, signalLine, period)
        dmax = getMaximumCertainties(macd, signalLine, period)
        upCrossingPoint = None
        for t in period:
            if upCrossingPoint is None and isAnUpCrossingPoint(macd, signalLine, t):
                upCrossingPoint = t

            if upCrossingPoint is None:
                hMinimum = macd[avmin[t]] - signalLine[avmin[t]]
                modifiedSignalLine[t] = signalLine[t] + hMinimum*(dmin[t]**K)
            elif t == period[-1]:
                modifiedSignalLine[t] = signalLine[t]
            else:
                hMaximum = macd[avmax[t]] - signalLine[avmax[t]]
```

```
            modifiedSignalLine[t] = signalLine[t] + hMaximum*(dmax[t]**K)

    return modifiedSignalLine


def getAvailablyMaximumPointsInAPeriod(macd, signalLine, period):
    avialablyMaximumPoints = pd.Series('', index=period)
    upCrossingPoint = None
    for t in period:
        if upCrossingPoint is None and isAnUpCrossingPoint(macd, signalLine, t):
            upCrossingPoint = t

        if upCrossingPoint is not None:
            avialablyMaximumPoints[t] = macd[upCrossingPoint:t].idxmax()

    return avialablyMaximumPoints

def getAvailablyMinimumPointsInAPeriod(macd, signalLine, period):
    avialablyMinimumPoints = pd.Series('', index=period)
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            avialablyMinimumPoints[t] = macd[period[0]:t].idxmin()
            break

        avialablyMinimumPoints[t] = macd[period[0]:t].idxmin()

    return avialablyMinimumPoints

def getMaximumCertainties(macd, signalLine, period):
    avmax = getAvailablyMaximumPointsInAPeriod(macd, signalLine, period)
    maximumCertainties = pd.Series('', index=period)
    upCrossingPoint = None
    for t in period:
        if t == period[-1]:
            maximumCertainties[t] = 1
            break

        if upCrossingPoint is None and isAnUpCrossingPoint(macd, signalLine, t):
```

```
            upCrossingPoint = t

        if upCrossingPoint is not None:
            hCurrent = macd[t] - signalLine[t]
            hMaximum = macd[avmax[t]] - signalLine[avmax[t]]
            maximumCertainties[t] = 1 - (hCurrent/float(hMaximum)) if hMaximum != 0 else 1

    return maximumCertainties

def getMinimumCertainties(macd, signalLine, period):
    avmin = getAvailablyMinimumPointsInAPeriod(macd, signalLine, period)
    minimumCertainties = pd.Series('', index=period)
    for t in period:
        if isAnUpCrossingPoint(macd, signalLine, t):
            minimumCertainties[t] = 1
            break

        hCurrent = macd[t] - signalLine[t]
        hMinimum = macd[avmin[t]] - signalLine[avmin[t]]
        minimumCertainties[t] = 1 - (hCurrent/float(hMinimum)) if hMinimum != 0 else 1

    return minimumCertainties
```

ตารางที่ ข.48 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_modifiedsignalline.py ในโฟลเดอร์ macdtesterP

```
from macdtesterP.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods, getUpCrossingPointInAPeriod
from macdtesterP.macd import getMACDAndSignalLine
import numpy as np
import os
import pandas as pd
from macdtesterP.data import getPricesData
from macdtesterP.modifiedsignalline import getModifiedSignalLine

def isABuyPoint(macd, modifiedSignalLine, upPoint, t):
    iup = macd.index.get_loc(upPoint)
```

```
    i = macd.index.get_loc(t)
    if i <= iup and isAnUpCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
        return False

def isASellPoint(macd, modifiedSignalLine, t):
    if isADownCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    for t in period:
        if buyPoint is None and isABuyPoint(macd, modifiedSignalLine, upPoint, t):
            buyPoint = t

        if buyPoint is not None and isASellPoint(macd, modifiedSignalLine, t):
            sellPoint = t

        if sellPoint is not None:
            tradeList.append(Trade(buyPoint, sellPoint, capital))
            buyPoint = None
            sellPoint = None

    if len(tradeList) > 0:
        return TotalTrade(tradeList)
    else:
        return None

def modifiedSignalLineTest(pricesData, macd, signalLine, modifiedSignalLine):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
```

```
    for period in periods:
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine):
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []
    for period in periods:
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)

    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine):
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []
    for period in periods:
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)
```

```
    return allSellPoints

def modifiedSignalLineTestWithSET100(K=1, verbose=False):
    directory = './resources/stockdata/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, 12, 26, 9)
        modifiedSignalLine = getModifiedSignalLine(macd, signalLine, K)
        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        modifiedSignalLine = modifiedSignalLine[25:]
        successRate, profitRate = modifiedSignalLineTest(pricesData, macd, signalLine,
modifiedSignalLine)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ตารางที่ ข.49 ตารางแสดงรหัสต้นฉบับในไฟล์ tradingtest_modsigtradingweight.py ในโฟลเดอร์ macdtesterP

```
from macdtesterP.basic import isAnUpCrossingPoint, isADownCrossingPoint, Trade,\
    TotalTrade, getTradingPeriods, getMaximumPointsInAPeriod,\
    getMinimumPointsInAPeriod, getUpCrossingPointInAPeriod
import numpy as np
from macdtesterP.modifiedsignalline import getAvailablyMinimumPointsInAPeriod,\
    getMinimumCertainties
from macdtesterP.fuzzy import FIS
import os
import pandas as pd
from macdtesterP.data import getPricesData
from macdtesterP.modifiedsignalline import getModifiedSignalLine
from macdtesterP.macd import getMACDAndSignalLine

def isABuyPoint(macd, modifiedSignalLine, upPoint, t):
    iup = macd.index.get_loc(upPoint)
    i = macd.index.get_loc(t)
    if i <= iup and isAnUpCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
        return False

def isASellPoint(macd, modifiedSignalLine, t):
    if isADownCrossingPoint(macd, modifiedSignalLine, t):
        return True
    else:
        return False

def getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint, overBought,
overSold, avmin, dmin):
    tradeList = []
    buyPoint = None
    sellPoint = None
    capital = 1
    weight = None
    for t in period:
```

```
            if buyPoint is None and isABuyPoint(macd, modifiedSignalLine, upPoint, t):
                buyPoint = t
                weight = fuzzyInfer(overBought, overSold, macd[avmin[t]], dmin[t])

            if buyPoint is not None and isASellPoint(macd, modifiedSignalLine, t):
                sellPoint = t

            if sellPoint is not None:
                tradeList.append(Trade(buyPoint, sellPoint, capital, weight))
                buyPoint = None
                sellPoint = None
                weight = None

        if len(tradeList) > 0:
            return TotalTrade(tradeList)
        else:
            return None

def modsigTradingWeightTest(pricesData, macd, signalLine, modifiedSignalLine, R):
    periods = getTradingPeriods(macd, signalLine)
    profitRateList = []
    successList = []
    for i in xrange(len(periods)):
        if i-R < 0: continue

        period = periods[i]
        avmin = getAvailablyMinimumPointsInAPeriod(macd, signalLine, period)
        dmin = getMinimumCertainties(macd, signalLine, period)
        overBought = getOverBought(macd, signalLine, periods, i, R)
        overSold = getOverSold(macd, signalLine, periods, i, R)
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = getTotalTradesInAPeriod(macd, modifiedSignalLine, period, upPoint,
overBought, overSold, avmin, dmin)
        if totalTrade is not None:
            profitRate = totalTrade.getProfitRate(pricesData)
            profitRateList.append(profitRate)
            if profitRate > 0:
                successList.append(1)
```

```
            else:
                successList.append(0)

    if len(profitRateList) > 0:
        return np.mean(successList), np.mean(profitRateList)
    else:
        return None, None

def getAllBuyPoints(pricesData, macd, signalLine, modifiedSignalLine, R):
    from tradingtest_modifiedsignalline import getTotalTradesInAPeriod as
modsig_getTotalTradesInAPeriod
    periods = getTradingPeriods(macd, signalLine)
    allBuyPoints = []

    for i in xrange(len(periods)):
        if i-R < 0: continue

        period = periods[i]
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
        totalTrade = modsig_getTotalTradesInAPeriod(macd, modifiedSignalLine, period,
upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allBuyPoints.append(trade.buyPoint)

    return allBuyPoints

def getAllSellPoints(pricesData, macd, signalLine, modifiedSignalLine, R):
    from tradingtest_modifiedsignalline import getTotalTradesInAPeriod as
modsig_getTotalTradesInAPeriod
    periods = getTradingPeriods(macd, signalLine)
    allSellPoints = []

    for i in xrange(len(periods)):
        if i-R < 0: continue

        period = periods[i]
        upPoint = getUpCrossingPointInAPeriod(macd, signalLine, period)
```

```
        totalTrade = modsig_getTotalTradesInAPeriod(macd, modifiedSignalLine, period,
upPoint)
        if totalTrade is not None:
            for trade in totalTrade.tradeList:
                allSellPoints.append(trade.sellPoint)

    return allSellPoints

def getOverBought(macd, signalLine, periods, i, R):
    if i-R < 0:
        raise Exception('R is too much.')

    maxPoint = getMaximumPointsInAPeriod(macd, signalLine, periods[i-R])
    overBought = macd[maxPoint]
    return overBought

def getOverSold(macd, signalLine, periods, i, R):
    if i-R < 0:
        raise Exception('R is too much.')

    minPoint = getMinimumPointsInAPeriod(macd, signalLine, periods[i-R])
    overSold = macd[minPoint]
    return overSold


fis = FIS()
MaxDefuzz = fis.outputs['z'].defuzzify(fis.outputs['z'].terms['TRADE-CONFIDENCE'])
MinDefuzz = fis.outputs['z'].defuzzify(fis.outputs['z'].terms['TRADE-DOUBT'])
def fuzzyInfer(overBought, overSold, macd_avmin, dmin):
    half = (overBought + overSold)/2.0
    oneForth = (overSold + half)/2.0
    threeForth = (overBought + half)/2.0
    fis.set_term_param('macd_avmin', 'MACD-HIGH', abcd=(half, overBought, '', ''))
    fis.set_term_param('macd_avmin', 'MACD-LOW', abcd=('', '', overSold, half))
    fis.set_term_param('macd_avmin', 'MACD-MEDIUM', abcd=(oneForth, half, half,
threeForth))

    fis.set_inputs(macd_avmin=macd_avmin, dmin=dmin)
```

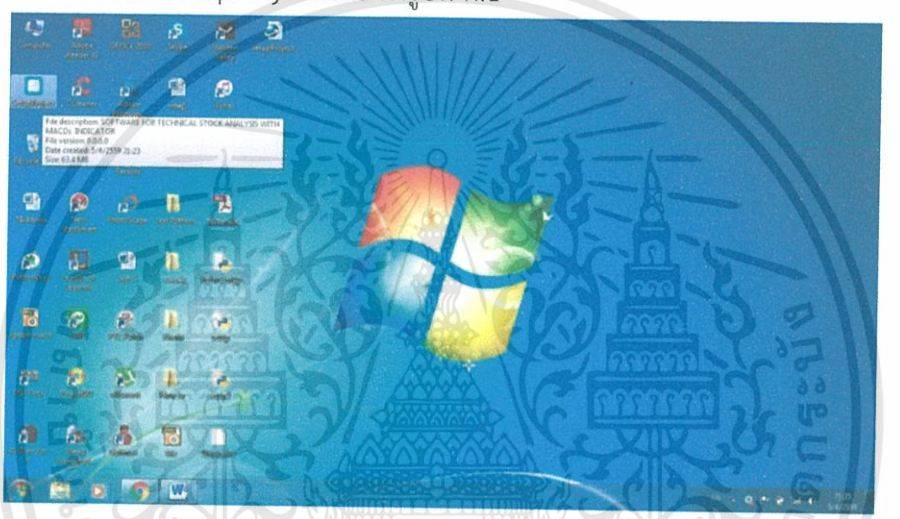
```
    z = fis.evaluate().get_outputs()['z'].value
    w = (z-MinDefuzz)/float(MaxDefuzz-MinDefuzz)

    return w

def modsigTradingWeightTestWithSET100(K=1, R=1, verbose=False):
    directory = './resources/stockdata/'
    ignoreList = ['BANPU', 'MAKRO', 'VGI', 'STPI']
    symbolList = [f.replace('.csv','') for f in os.listdir(directory) if f.endswith('.csv') and
f.split('.')[0] not in ignoreList]

    resultList = pd.DataFrame({'success rate': None, 'profit rate': None}, index=symbolList)

    for symbol in symbolList:
        if verbose: print 'testing ' + symbol + '...'
        pricesData = getPricesData(symbol=symbol)
        macd, signalLine = getMACDAndSignalLine(pricesData, 12, 26, 9)
        modifiedSignalLine = getModifiedSignalLine(macd, signalLine, K)
        pricesData = pricesData[25:]
        macd = macd[25:]
        signalLine = signalLine[25:]
        modifiedSignalLine = modifiedSignalLine[25:]
        successRate, profitRate = modsigTradingWeightTest(pricesData, macd, signalLine,
modifiedSignalLine, R)

        resultList['profit rate'][symbol] = profitRate
        resultList['success rate'][symbol] = successRate

    if verbose:
        import webbrowser
        f = open(os.getcwd() + '/temp.html', 'w')
        f.write('Original MACD test with SET-100')
        f.write(resultList.to_html())
        f.close()
        webbrowser.open(os.getcwd() + '/temp.html', new=2)

    resultList = resultList.dropna()
    avgSuccessRate = resultList['success rate'].mean()
```

```
    avgProfitRate = resultList['profit rate'].mean()
    return avgSuccessRate, avgProfitRate
```

ภาคผนวก ค
ความรู้พื้นฐานในการใช้โปรแกรม

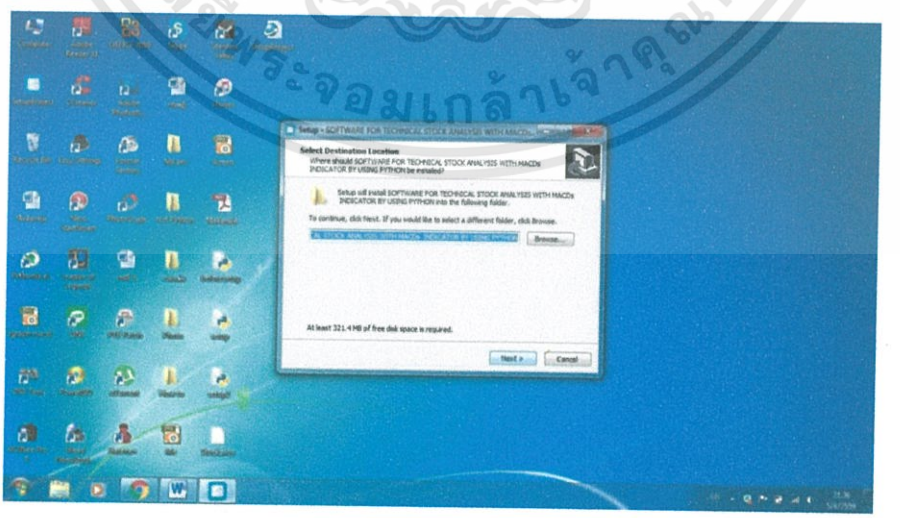
ภาคผนวก ค จะกล่าวถึงวิธีใช้งานโปรแกรม โดยจะสมมติว่าเครื่องคอมพิวเตอร์ที่ใช้มีระบบปฏิบัติการคือ Windows 7 หรือ Windows 8 และได้ทำการติดตั้งชุดซอฟต์แวร์ SOFTWARE FOR TECHNICAL STOCK ANALYSIS WITH MACDs INDICATOR BY USING PYTHON โดยมีขั้นตอนดังนี้

1. เลือกไฟล์ติดตั้งชื่อ SetupProject.exe ดังรูปที่ ค.1



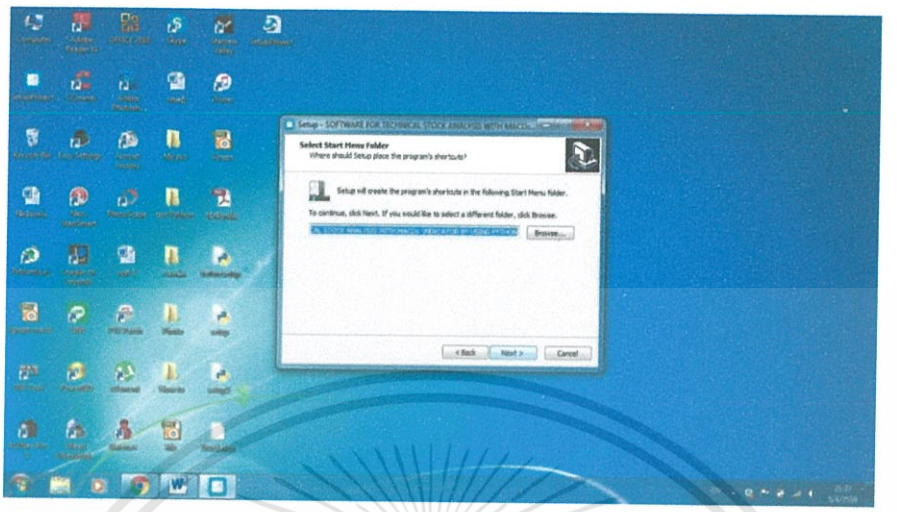
รูปที่ ค.1

2. เมื่อปรากฏหน้าต่างติดตั้งขึ้นมาให้กด “next” ดังรูปที่ ค.2



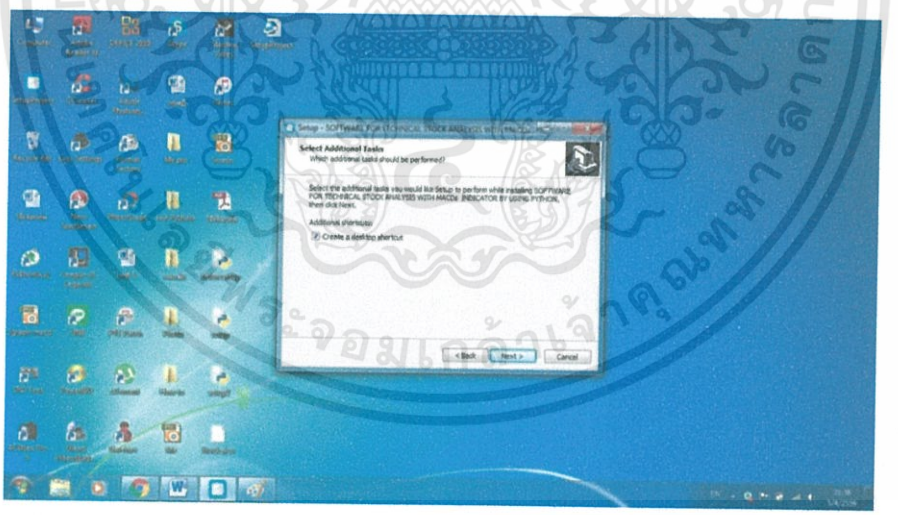
รูปที่ ค.2

3. จากนั้นให้กด “next” ดังรูปที่ ค.3



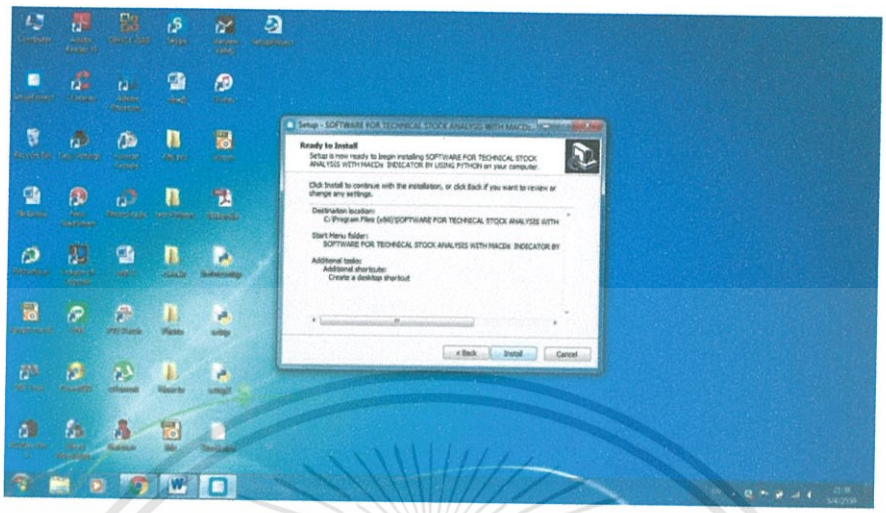
รูปที่ ค.3

4. ติ๊กถูกในช่องว่างจากนั้นให้กด “next” ดังรูปที่ ค.4



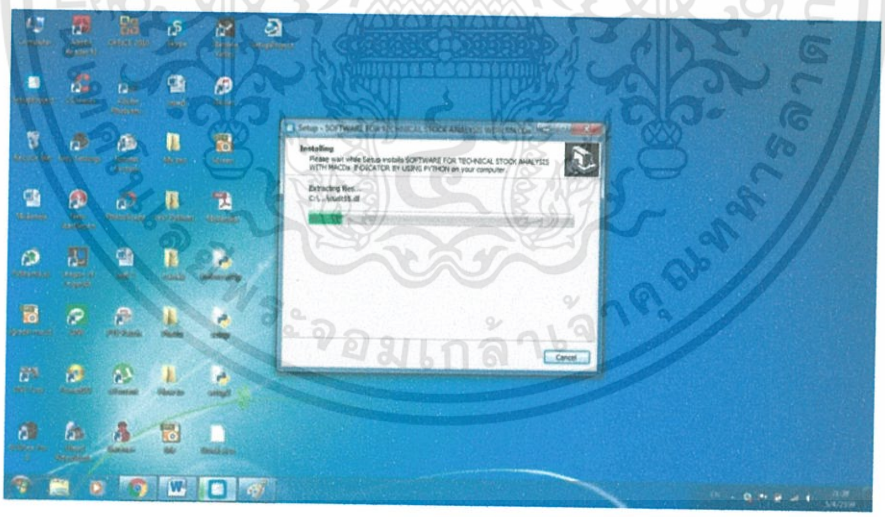
รูปที่ ค.4

5. จากนั้นให้กด “Install” ดังรูปที่ ค.5



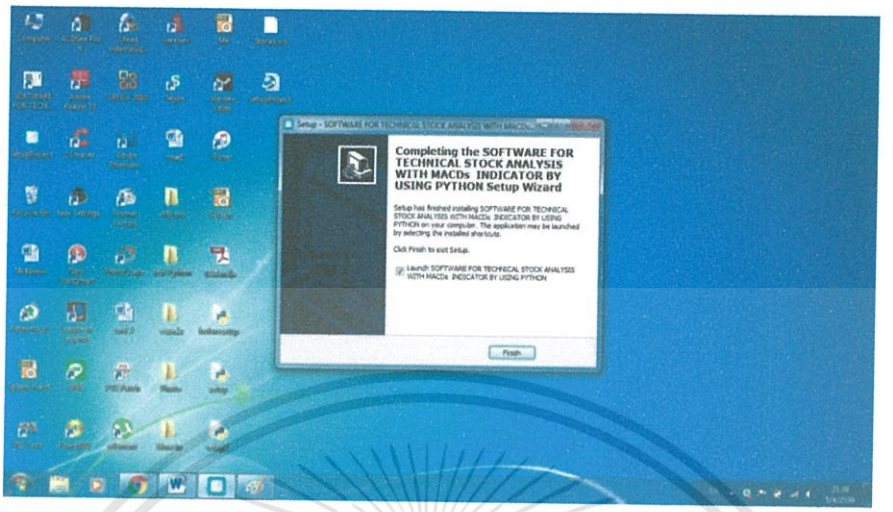
รูปที่ ค.5

6. รอสักครู่เพื่อติดตั้งโปรแกรม ดังรูปที่ ค.6



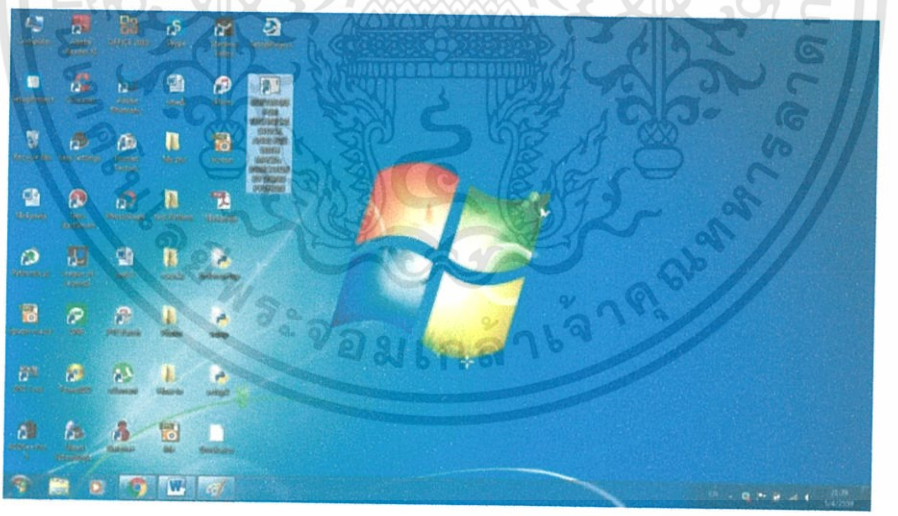
รูปที่ ค.6

7. เมื่อเสร็จสิ้นการติดตั้งให้กด “Finish” ดังรูปที่ ค.7



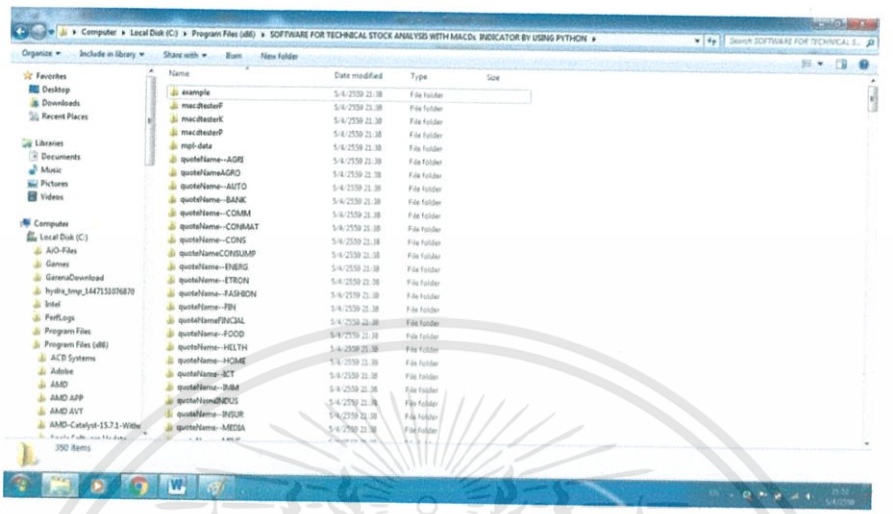
รูปที่ ค.7

8. บนหน้าจอจะปรากฏไอคอนของโปรแกรม สามารถเข้าใช้งานจากไอคอนนี้ได้ ดังรูปที่ ค.8



รูปที่ ค.8

เมื่อลงโปรแกรมเสร็จสิ้นไฟล์ของโปรแกรมทั้งหมดจะอยู่ใน C:\Program Files (x86)\SOFTWARE FOR TECHNICAL STOCK ANALYSIS WITH MACDs INDICATOR BY USING PYTHON ดังรูปที่ ค.9



รูปที่ ค.9