STATE CHANGE DETECTION MODEL OF ONLINE GAME PLAYERS

KITTIPAT SAVETRATANAKAREE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE
DEGREE OF PHILOSOPHY IN COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2016

KMITL-2016-SC-D-002-014

STATE CHANGE DETECTION MODEL OF ONLINE GAME PLAYERS

KITTIPAT SAVETRATANAKAREE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE
DEGREE OF PHILOSOPHY IN COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2016
KMITL-2016-SC-D-002-014

หัวข้อวิทยานิพนธ์	ตัวแบบตรวจสอบการเปลี่ยนแปลงสถานะของผู้เล่นเกมออนไลน์
ชื่อนักศึกษา	นายกิตติภัต เศวตรัตนกรี
รหัสประจำตัว	51062904
ปริญญา	วิทยาศาสตรปรัชญาดุษฎีบัณฑิต วิทยาการคอมพิวเตอร์
ภาควิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2559
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร.ศรัณย์ อินทโกสุม
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	ผศ.ดร.กิ่งกาญจน์ สุขณาภิบาล

บทคัดย่อ

บริษัทผลิตเกมออนไลน์ส่วนใหญ่จะแก้ปัญหาที่ผู้เล่นหลักของเกมจะเลิกเล่นเกมโดยการให้โปรโมชั่นกับผู้เล่นกลุ่มดังกล่าว ซึ่งการที่จะทราบว่าผู้เล่นหลักคนใดจะเลิกเล่นเกมจำเป็นต้องมีตัวแบบที่สามารถทำนายพฤติกรรมของผู้เล่นเกมได้อย่างมีประสิทธิภาพ มีงานวิจัยที่ผ่านมามีการพยายามจะสร้างตัวแบบดังกล่าว แต่ยังคงไม่สมบูรณ์ งานวิจัยนี้ได้นำเสนอตัวแบบตรวจสอบเพื่อการทำนายสถานะของผู้เล่นเกมออนไลน์ได้อย่างแม่นยำเที่ยงตรงมากขึ้น โดยได้นำเสนอสองวิธีใหม่คือการนำข้อมูลการเล่นเกมล่าสุดด้วยจำนวนวันเท่ากับตัวแปรเคที่เหมาะสมมาช่วยในการทำนายและเทคนิคใหม่ในการสุ่มตัวอย่างเพิ่มเพื่อสามารถจัดการกับปัญหาความไม่สมดุลกันของข้อมูลผู้เล่นเกมประจำกับผู้ที่มีแนวโน้มจะเลิกเล่นเกม

จากการทดลองจำนวนมากได้ข้อสรุปว่าข้อมูลผู้เล่นเกมล่าสุดในจำนวนวันที่เหมาะสมคือสองวันเป็นจำนวนวันที่ช่วยในการทำนายได้ดีที่สุดและเทคนิควิธีใหม่ในการสุ่มตัวอย่างเพิ่มข้างน้อยสังเคราะห์ที่เส้นขอบ(BOSFS)ได้ถูกพัฒนาขึ้นด้วย ผู้วิจัยได้ทำการทดลองกับตัวแบบที่นำเสนอทั้งสองวิธีโดยเปรียบเทียบกับเทคนิควิธีที่มีในปัจจุบันได้แก่ SMOTE, Borderline-SMOTE และ BOS เป็นต้น ผลการทดลองของตัวแบบที่นำเสนอมีผลการประเมินด้วยค่าเฉลี่ยจีสูงถึง 94.35% และอัตราการรู้จำสูงถึง 94.27% ซึ่งเป็นค่าดีกว่าค่าที่ได้จากเทคนิควิธีที่มีในปัจจุบัน นอกจากนี้ ผู้วิจัยยังได้นำเทคนิควิธีใหม่ในการสุ่มตัวอย่าง(BOSFS)ไปทดลองใช้กับข้อมูลอื่นที่ไม่ใช่เกมได้แก่ ข้อมูลมาตรฐานยูซีไอ ผลการทดลองทั้งหมดได้แสดงให้เห็นว่า เทคนิคใหม่BOSFSมีประสิทธิภาพในการทำนายที่ดีกว่าเทคนิควิธีที่มีในปัจจุบันข้างต้น

คำสำคัญ : การสุ่มตัวอย่างเพิ่มข้างน้อยสังเคราะห์ที่บริเวณเส้นขอบในฟีเจอร์สเปซ, การกลับมาเล่นเกม, เกมออนไลน์แบบเล่นตามบทบาท

Thesis Title	State Change Detection Model of Online Game Players
Student Name	Kittipat Savetratanakaree
Student ID	51062904
Degree	Doctor of Philosophy (Computer Science)
Department	Computer Science
Year	2016
Thesis Advisor	Asst. Prof. Dr. Sarun Intakosum
Thesis Co-advisor	Asst. Prof. Dr. Kingkarn Sookhanaphibarn

Abstract

Most online game companies usually solve the problem of main online game players who might quit the game by giving these players special promotions. In order to know such information, an efficient prediction model is considered necessary. There are some existing prediction models that are developed to solve the problem. The accuracy of each model, though, is acceptable but it could be improved. The purpose of this dissertation is to propose a new state change detection model of online game players that gives higher accuracy. There are two major factors that are taken into consideration in developing the proposed model, firstly, the latest k-day playing information of the players, secondly, a new efficient algorithm to solve the imbalanced data class problem of the players' data set. For the former, many experiments have been conducted. The results have shown that the latest 2-day information gives the best result. For the latter, the novel algorithm called Borderline Oversampling in Feature Space (BOSFS) is developed.

F-measure and g-means are used as the performance metrics. The results have shown that the proposed model outperforms the existing models with the 94.35% and 94.27% of g-means and F-measure respectively. Moreover, experiments on applying BOSFS to five standard UCI data sets have also been conducted. The results have shown that there are significant improvements comparing to the existing algorithms namely SMOTE, Borderline-SMOTE and BOS.

Keywords: borderline over-sampling in feature space, game revisitations, massively multiplayer online role-playing game (MMORPG)

Acknowledgements

I would like to express many thanks to my major advisor, Asst. Prof. Dr. Sarun Intakosum, for his ongoing advice and support. My appreciation goes to my co-advisor, Asst. Prof. Dr. Kingkarn Sookhanaphibarn for reviewing my research and giving valuable advice. I also would like to thank Prof. Dr. Ruck Thawonmas for his useful comments on my research. I highly appreciate with all advice and great support starting from the initial step to the final one.

I am particularly grateful to dissertation committee for comments and suggestions that helped improve my presentation and dissertation.

I would like to thank my family for their devotion, understanding, great help and encouragement along the way.

I am heartily thankful to the Faculty of Computer Science and Software Engineering Department at Bangkok University, Faculty of Computer Science Department and the Office of Academic Administration of King Mongkut's Institute of Technology, Ladkrabang.

Finally, I would like to thank my dear friends and others for their help, support and encouragement in all aspects.

Mr. Kittipat Savetratanakaree

Table of Contents

Table of Contents

List of Tables

List of Figures

List of Figures

Figures **Page**

Abbreviations

BOS	Borderline Over Sampling
BOSFS	Borderline Over Sampling in Feature Space
MMORPG	Massively Multiplayer Online Role-playing Game
SLK days	Staytime of Last K days
SVM	Support Vector Machine
SZO	Shen Zhou Online game
TN	True Negative
TP	True Positive
FN	False Negative
FP	False Positive

1.2 Objectives of the study

To propose a novel state change detection model of online game players in order to have a better prediction performance in learning classifiers.

1.3 Scope(s) of the study

1) The proposed model focuses on game access log files of Shen Zhou Online game dataset as training and testing datasets from year 2004 to 2006.
2) The study focuses on new features extraction from the following attributes of online game access log files as login time, logout time, playing time or staytime.
3) Latest staytime of game playing information will be developed to support the better prediction of learning classifiers.
4) New over-sampling method will be developed to solve Support Vector Machines classification problems dealing with imbalanced class dataset.

1.4 Benefits of the study

1) Online game companies obtains state change detection model of online games players to predict online game players who are in continual-playing state or in quit state. They can give the right promotions out to the right players. This will keep high retention rate of members and the game companies will not lose their income.
2) Latest staytime of playing information of online game players can be considered to improve prediction performance of the proposed model.
3) New over-sampling method can be used to improve SVM classifiers dealing with imbalanced class data environments of online game dataset.
4) New over-sampling method can be applied to other domain datasets not specified only to online game datasets.

Chapter 2
Theory and Literature Reviews

This chapter introduces the main learning algorithm in section 2.1, Support Vector Machines (SVM) that is applied as base learning algorithm for my proposed methods in the state change detection model of online game players. Non-linear Classifier with SVM is explained in section 2.2. Basic kernel methods using in non-linear classifier and kernel methods selection are discussed in section 2.3. Support Vector Machines with imbalanced data environments in section 2.4. Performance metrics used in the proposed methods in section 2.5. Sensitivity and Specificity of SVM in section 2.6, and literature reviews in section 2.7 to detail problems of previous research using game revisitations and existing methods in coping with imbalanced class data problems.

2.1 Support Vector Machines

Support Vector Machines (SVM), first proposed by Vapnik [3], is a successfully supervised learning model with correlated learning algorithms in machine learning. SVM analyzes data for classification and regression analysis.

Suppose, there is a given set of training dataset of n points as:

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2) \ldots, (\vec{x}_n, y_n)$$

where the y_i are either -1 or 1, each data item indicated for which $\vec{x}_i$ appears with one of two classes, an SVM training algorithm builds a model that appoints new instances into one class or the other, this is why SVM is called a binary linear classifier. A representation of the instances in an SVM model is as mapped points in space, so that the instances of the separate classes are divided by a clear area that is as large as possible.

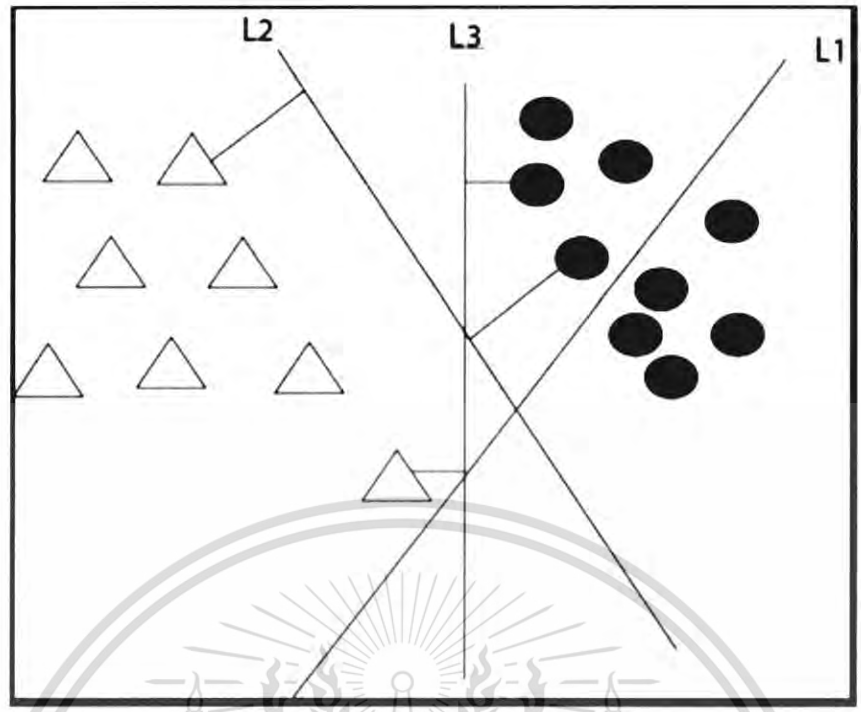


Figure 2.1 Example of lines that separate classes

In Figure 2.1, there are two classes: triangles and black circles. L_1, L_2 and L_3 are three lines separating two classes in different area. The figure illustrated that L_1 can not separate the classes. L_2 separates the classes with the maximum margin. L_3 divides them with the smaller margin than L_2. Line L_2 separates the classes margin with the most possible maximum. L_3 will not be chosen for SVM because it separates the classes with smaller margin than L_2. New instances in testing dataset are classified into the same space and forecasted to match a class according to their side of the space.

The objective of Support Vector Machines (SVM) [4] [5] [6] [7] [8] is to find the optimal boundary that separates one class instance from another class instance with the maximum margin. There are two main cases of SVM as linearly separable case and non-linear separable case.

2.1.1 Linearly separable case (Hard margin)

In case of linearly separable data as explained in [5], two parallel hyperplanes can perfectly divide two classes of data, so the area between two classes is as maximized as possible. These two hyperplanes bound the area, which is called the margin. In Figure 2.2 it illustrates two hyperplanes in the linear separable case which sometimes is also called "hard margin" case.

These two hyperplanes according to [4][5] can be described by the equations

$$\vec{w} \cdot \vec{x} - b = 1 \tag{2.1}$$

In Figure 2.2, the equation (2.1) is the first hyperplane on the left margin with triangle classes

$$\vec{w} \cdot \vec{x} - b = -1 \tag{2.2}$$

The equation (2.2) is the second hyperplane on the right margin with circle classes.

where $\vec{w}$ is weight vector or the normal vector to the hyperplane and the parameter b is the bias of the hyperplanes and the subspace that lies halfway between them is called the maximum-margin hyperplane

The hyperplane as the set of points $\vec{x}$ can be written satisfying

$$\vec{w} \cdot \vec{x} - b = 0 \tag{2.3}$$

The offset of the hyperplane described in [4][5] from the original source along the weight vector $\vec{w}$, determines the parameter as $\frac{b}{||\vec{w}||}$. The equation (2.3) is in the middle line that separates two hyperplanes in Figure 2.2.

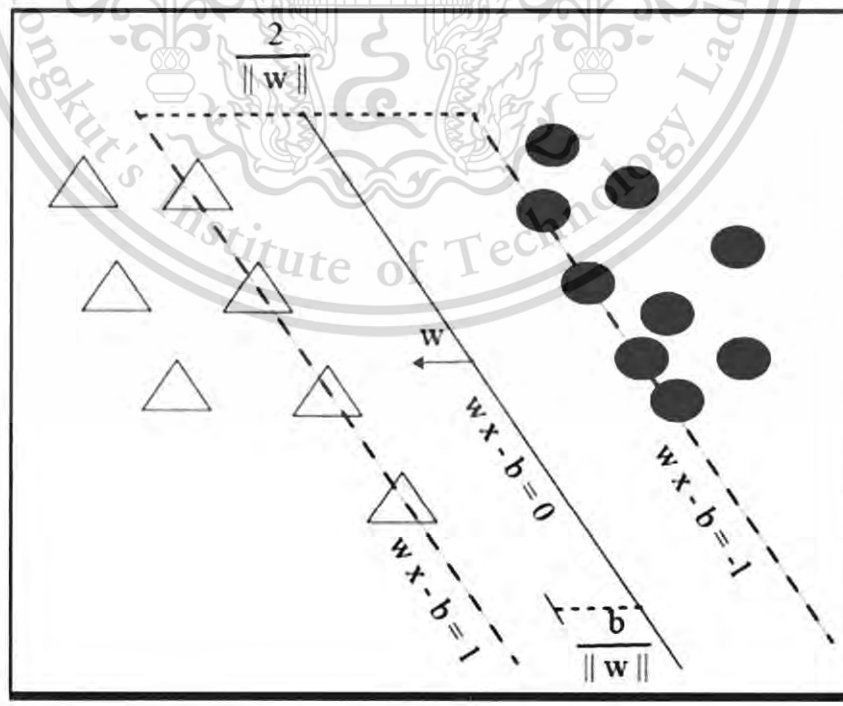


Figure 2.2 the linear separable case and the maximum hyperplane

The distance between two hyperplanes according to [4][5] is $\frac{2}{||\vec{w}||}$, in order to maximize them, it needs to minimize $||\vec{w}||$. In addition, it needs to avoid data points from falling into the margin, so the subsequent constraints need to be added.

for each i either

$$\vec{w} \cdot \vec{x}_i - b \geq 1, \quad if \ y_i = 1 \tag{2.4}$$

or

$$\vec{w} \cdot \vec{x}_i - b \leq -1, \quad if \ y_i = -1 \tag{2.5}$$

Equation (2.4) and (2.5) represent the constraints in which each data point must fall on the right side of the margin.

These two equations can be edited as:

$$y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1, \ for \ i = 1, \ldots . n \tag{2.6}$$

As in [4][5] represents the optimization problem:

Minimize $||\vec{w}||$ subject to

$$y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1, \ for \ i = 1, \ldots . n$$

$\vec{w}$ and b could solve the problem to determine

$$\vec{x} \rightarrow sgn(\vec{w} \cdot \vec{x}_i - b) \tag{2.7}$$

where sgn is the sign function described in [4][5] which can be -1, 1 depends on the variables in the parentheses.

The maximum margin hyperplane is determined by these $\vec{x}_i$ which are nearest to it. These $\vec{x}_i$ are called support vectors.

2.1.2 Non-Linearly separable case (Soft margin)

Support Vector Machines (SVM) [4] [5] [6] [7] [8] could be extended to use for non-linearly separable data. In this case, the hinge loss function needs to be used for training classifiers:

$$\max(0, 1 - y_i\,(\vec{w} \cdot \vec{x}_i - b)) \tag{2.8}$$

This function as described in [5] will be zero if it can satisfy the constraint in Equation (2.6) or $\vec{x}_i$ is on the valid side of the margin. If $\vec{x}_i$ is on the invalid side of the margin, the value of this function is to be proportional to the distance from the margin.

[4] [5] described the objective of this hinge loss function is to minimize this function:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i\,(\vec{w} \cdot \vec{x}_i - b))\right] + \lambda||\vec{w}\,||^2 \tag{2.9}$$

where the variable λ regulates the adjustment between expanding margin-size and assures that the $\vec{x}_i$ was on the valid side of the margin. Hence, for adequately small values of λ, the soft-margin SVM will act similarly to the hard-margin SVM.

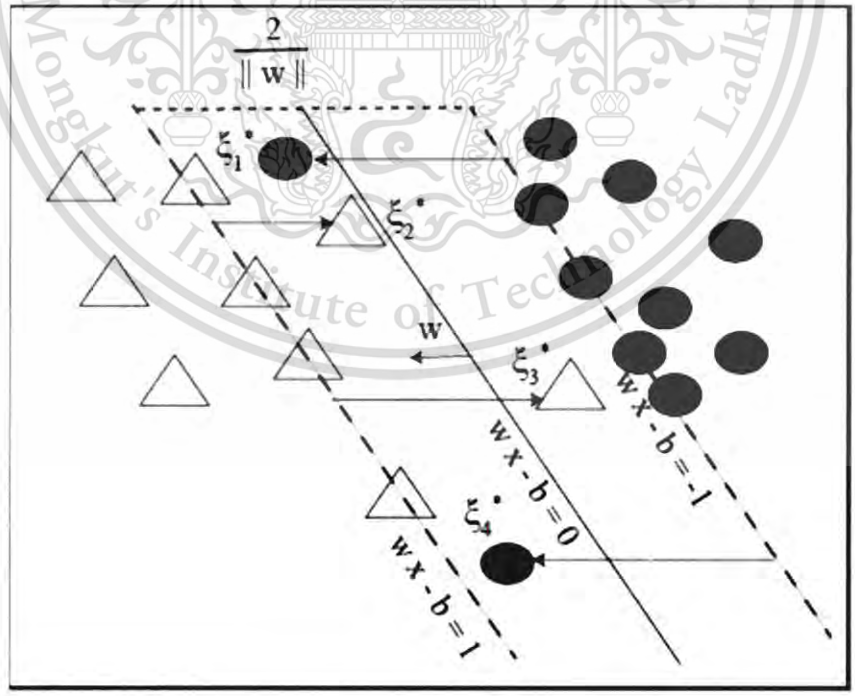


Figure 2.3 the non-separable case (overlap case)

In Figure 2.3 the points tagged ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$ where the points on the right side have $\xi_j^* = 0$.

The margin $M = \frac{1}{||\vec{w}||}$ and the maximum margin of width $2M = \frac{2}{||\vec{w}||}$. The margin maximized is subject to a total allocation $\sum \xi_j \leq constant$.

Thus the $\sum \xi_j^*$ is the entire distance of points on the wrong side of their margin.

2.2 Non-linear Classifier with SVM

In real world problems, classes may not be separable by a linear boundary. Hence, non-linear classifier with SVM needs to be applied to these cases. There are two representations of optimization problems as follows:

2.2.1 Primal problem

According to [4][5], the equation (2.9) could be rewritten as an optimization problem as follows:

For each $i \in \{1, \ldots, n\}$, define the variable

$$\text{₣}_i = \max(0, 1 - y_i (w \cdot x_i - b))$$

If and only if ₣_i is the smallest nonnegative number satisfying

$$y_i (w \cdot x_i - b) \geq 1 - \text{₣}_i \qquad (2.10)$$

Hence, the optimization problem could be in the form as follows

$$minimize \left[\frac{1}{n}\sum_{i=1}^{n} \text{₣}_i\right] + \lambda||\vec{w}||^2 \qquad (2.11)$$

subject to equation (2.10) and $\text{₣}_i \geq 0 \quad for\ i = 1, \ldots . n$

Equation (2.11) could be rewritten in term of w and C parameter, where w is the weight vector and C is the user-specified parameter for the penalty on training instances on the wrong side of the boundary. This C parameter is the weakness of the soft-margin of SVM [9] because C is a fixed number specified by user. If user choose the C parameter to be too high or low, it will cause over-fitting or under-fitting problem.

This equation (2.11) could be computed by minimizing the objective function as described in [4][5] as follows:

$$\min_{w,b,\xi} \frac{1}{2} w \cdot w^T + C \sum_{i=1}^{n} \xi_i \quad (2.12)$$

subject to

$$\begin{cases} \forall_i \, 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{N} \propto_i y_i = 0 \end{cases} \quad (2.13)$$

2.2.2 Dual problem

Solving the Lagrangian dual of the equation (2.11) described in [4][5] and the dual representation of Equation (2.11), which is a quadratic function of c_i is as follows:

$$\text{maximize } f(c_1 \ldots c_n) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i \, c_i (x_i \cdot x_j) y_j c_j \quad (2.14)$$

subject to $\sum_{i=1}^{n} c_i y_i = 0, and \; 0 \leq c_i \leq \frac{1}{2n\lambda} \; for \; i = 1, \ldots . n.$

The variables c_i are defined as

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \vec{x}_i \quad (2.15)$$

The value of variable c_i = 0 when $\vec{x}_i$ falls on the right side of the margin, and $0 \leq c_i \leq \frac{1}{2n\lambda}$ when $\vec{x}_i$ falls on the margin's boundary. The linear combination of the support vectors $\vec{x}_i$, the weight vector $\vec{w}$ and the offset b could be rewritten by finding the $\vec{x}_i$ to solve:

$$y_i (\vec{w} \cdot \vec{x}_i + b) = 1 \Leftrightarrow b = y_i - \vec{w} \cdot \vec{x}_i$$

2.3 Kernel methods

In 1964, nonlinear classifiers were originally proposed by Aizerman et al. [6]. In 1992, Bernhard E. Boser el al. [7] advised a method to create nonlinear classifiers by applying the kernel method to maximize the largest-margin hyperplanes. The simple way [8] of making a non-linear classifier out of a linear classifier to map the training data from the input space X to a feature space $\mathcal{F}$ using a non-linear function

ϕ: $X \rightarrow \mathcal{F}$. In the space $\mathcal{F}$ the discriminant function is

$$f(x) = w^T \phi(x) + b \tag{2.16}$$

The weight vector can be represented as

$$\vec{w} = \sum_{i=1}^{n} \alpha_i x_i$$

The equation (2.16) [7] can be represented as

$$f(x) = \sum_{i=1}^{n} \alpha_i x^T x + b \tag{2.17}$$

In the feature space, $\mathcal{F}$ equation (2.17) [7] could be in the form:

$$f(x) = \sum_{i=1}^{n} \alpha_i \phi(x_i)^T \phi(x) + b \tag{2.18}$$

Kernel methods avoid the step of explicitly mapping the data to a high dimensional feature space by using the kernel function $K(x, x')$, which is defined as

$$K(x, x') = \phi(x_i)^T \phi(x') \tag{2.19}$$

The equation (2.18) could be computed in terms of the kernel function:

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b \tag{2.20}$$

There are four basic kernel methods described in [10]:

2.3.1 **Linear kernel** [10]: where kernel function is defined as follows

$$K(x_i, x_j) = x_i{}^T x_j \tag{2.21}$$

where $x_i, x_j \in R^n$

2.3.2 **Polynomial kernel** [10]: where kernel function is defined as follows

$$K(x_i, x_j) = (\gamma x_i{}^T x_j + r)^d, \ when \ \gamma > 0 \tag{2.22}$$

2.3.3 **Radial basis function (RBF) kernel** [10]: where kernel function is defined as follows

$$K(x_i, x_j) = exp(-\gamma \left\| x_i - x_j \right\|^2), \ when \ \gamma > 0 \tag{2.23}$$

Or It could be rewritten by [39] as

$$K(x_i, x_j) = exp\left(-\frac{\left\| x_i - x_j \right\|^2}{2\sigma^2}\right), \ when \ \gamma = \frac{1}{2\sigma^2} \ , \ \gamma > 0 \tag{2.24}$$

2.3.4 **Sigmoid kernel** [10]: where kernel function is defined as follows

$$K(x_i, x_j) = tanh(\gamma x_i{}^T x_j + r) \tag{2.25}$$

where γ, r, and d are kernel parameters

After preparing the training data set, training SVM model is applied by learning from training dataset with appropriate kernel methods, the unknown instance x could be classified by the equation (2.7) and (2.20) as follows:

$$f(x) = sgn(\sum_{i=1}^{n} \alpha_i K(x, x_i) + b) \tag{2.26}$$

where $sgn(\varsigma) = +1 \; if \; \varsigma \geq 0$, this mean that $f(x) = +1$ then x could be assign to positive class.

In the other case, where $sgn(\varsigma) = -1 \; if \; \varsigma \leq 0$, this mean that $f(x) = -1$ then x could be assigned to negative class.

Chih-Wei Hsu et al. [10] discussed about kernel methods selection about which methods are suitable for which cases. The linear kernel is suitable for linear classification where the others can handle the case when the relation of class labels and attributes is nonlinear. In general case, the RBF kernel is a reasonable method since this kernel maps instances nonlinearly into a high dimensional space for nonlinear separable case. Keerthi and Lin [11] showed that linear kernel is a special case of RBF kernel because the linear kernel with a C penalty parameter has the same result of performance as the RBF kernel with parameter (C, γ). Lin and Lin [12] showed that sigmoid kernel perform like RBF for certain parameter.

As discussed in [10], for the case of the number of hyperparameters effects complicatedness of model selection, the RBF kernel has less hyperparameters than the polynomial kernel. This means that RBF kernel might be less complicated than the polynomial kernel. The RBF kernel also has fewer numerical difficulties comparing to the polynomial kernel that kernel values of polynomial kernel might go far to one of these two cases:

$$\begin{cases} infinity & when \left(\gamma {x_i}^T x_j + r\right) > 1 \\ zero & when \left(\gamma {x_i}^T x_j + r\right) < 1 \end{cases}$$

As discussed in [10], for the cases in which the RBF kernel is not suitable, when there is a very large numbers of features, for example, the number features are large as 3,000 features, we might use the linear kernel instead.

2.4 Support Vector Machines in Imbalanced Data Environments

SVM has been a remarkably successful classifier in a wide range of applications if class data distribution assumed to be balanced. However, SVM is inefficient when it deals with imbalanced datasets as the numbers of majority-class instances greatly exceed the numbers of minority-class instances. The reason of this is shown in the Figure 2.4.

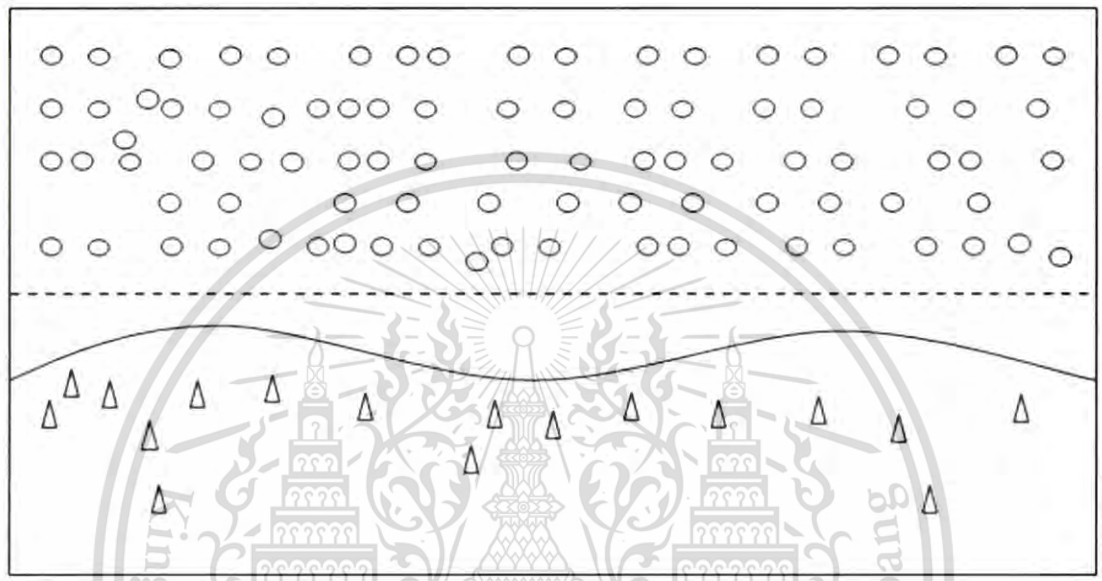


Figure 2.4 Support Vector Machines with Imbalanced Data Class Environments

Figure 2.4 SVM with imbalanced dataset has shown that the majority-class instances (circles) are much more than the minority-class instances (triangles). As a result, classification of SVM could classify with a tendency to be all majority-class instances. The borderline (strong line) is biased to the minority-class instances. The ideal borderline with best classification of two class instances should be in the position of a dashed line. Therefore, the main target to improve SVM with imbalanced dataset is expanding the borderline closer to the ideal borderline for better classification between majority-class and minority class instances.

The term majority-class instances are called the negative instances and the term minority-class instances are called the positive instances. This is because the interesting factor in minority-class instances of most cases in the imbalanced class data environments is rare and more important than the factors in majority-class instances.

2.5 Sensitivity and Specificity of SVM

The ratio between the number of true positive predictions (TP) and the total number of positive instances in the test set is called "Sensitivity" [13] defined as follows:

$$\boldsymbol{sensitivity} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \quad (2.27)$$

where FN is the number of false negatives.

Sensitivity in equation (2.27) is also called "$\boldsymbol{acc^+}$" [13] or the classification of accuracy of positive (majority-class) instances.

The ratio between the number of true negative predictions (TN) and the total number of negative instances in the test set is called "Specificity" [13] defined as follows:

$$\boldsymbol{specificity} = \frac{\boldsymbol{TN}}{\boldsymbol{TN+FP}} \quad (2.28)$$

where FP is the number of false positives.

Specificity in equation (2.28) is also called "$\boldsymbol{acc^-}$" [13] or the classification of accuracy of negative (minority-class) instances.

2.6 Performance metrics

When SVM are dealing with imbalanced class dataset, performance metric such as accuracy is essentially ineffective. For example, if imbalanced ratio of a dataset (Majority-class instances : Minority-class instances) is 96 : 4, after training with SVM the result of classification would be all majority-class or negative instances as accuracy of 96% and the minority-class or positive instances (4%) would be ignored as noise because of the imbalanced class dataset explained in the section 2.4.

Performance metrics of SVM when dealing with imbalanced class dataset are "g-means" metric and another effective metrics as "F-measure".

2.6.1 g-means metric

The g-means metric using by several researches [15] [16] [17] defined as:

$$\boldsymbol{g-means} = \sqrt{\boldsymbol{acc^+ + \ acc^-}} \quad (2.29)$$

The term $\boldsymbol{acc^+}$, $\boldsymbol{acc^-}$ have been previously defined in section 2.5.

2.6.2 **F-measure metric**

The F-measure metric [18] defined as:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (2.30)$$

The term "precision" is defined as the proportion of total predicted positive numbers that are correctly real positives. The term "recall" is another name of sensitivity in equation (2.27) previously defined in section 2.5.

2.7 Literature Reviews

In the recent years, many research [19] [20] has focused on modeling a player's behavior in a particular game. Player Modeling (PM) [21] is the act of tracking game metrics and using information in game metrics afterwards to construct representative models of players. Machine Learning (ML) techniques are usually used in Player Modeling. The types of learning models are varied since PM can be applied to different objectives. In most research, PM is used for particular game with objectives to improve the design of the game studied. A variety of research domains have conduct experiments using online game player's information productively in contributing useful information for their research objectives. Numerous researches related to online game player's information are reviewed as follows.

2.7.1 Game revisitations

Thawonmas et al. [2] proposed the feature "game revisitations" which is defined as the time intervals for how long game players at the time being login to the game comparing to their last logout. If the time interval of game revisitations is short, it means that game player is very enthusiastic to play the game. If the time interval of game revisitations is larger than sixteen days, it means that game player has lost interest in playing the game. The playing frequency on game revisitations is collected by counting the number of times the online game player revisits the same game by logging in within predefined time intervals as classified into 13 bins as shown in table 2.1

Table 2.1 Game revisitations classification of 13 predefined bins in time intervals and approximated time intervals

Bin no.	Time Interval (minutes)	Approximated Time Interval (m = minute, h = hour, d = day)
1	32	30m
2	64	1h
3	98	1.5h
4	136	2h
5	212	3h
6	424	6h
7	848	12h
8	1,696	1d
9	3,392	2d
10	6,784	4d
11	13,568	8d
12	27,136	16d
13	>27,136	>16d

The results of research in game revisitations [2] was not satisfied as the maximum recognition rate was 76 %, maximum precision rate was 76% and maximum recall rate was 78%. There are two problems in game revisitations [2]. First, they use under-sampling method in data preprocessing for their training dataset because weakness of under-sampling method is that it removed majority-class instances which may be important until the balance class numbers of majority-class instances and minority-class instances. Second, game revisitations lacked latest players' information in the recent period before prediction process. We could improve game revisitations with our new proposed additional feature with latest players' information and over-sampling methods in order to gain better performance in terms of accuracy (recognition rate), precision and recall.

2.7.2 Previous researches in Imbalanced data class environments

There were many previous real world applications researches that faced the problem of imbalanced data class distribution. These researches were based on different domains for examples, detection of oil spill in satellite images [22], detection of credit card fraud [23] and detection of fraudulent telephone calls [24]. In dangerous case for life of patients, there was detection of possible breast cancer focused on pixels in mammogram images [25]. There were 98% of pixels in the images were normal pixels. These normal pixels are classified into majority-class instances. In addition, abnormal pixels as minority-class instances were 2% left. If learning algorithms failed to classify abnormal pixels or they did not detect abnormal pixels, it could be dangerous to patients' lives. Unusual cases as abnormal pixels were treated as noise and normally ignored by the machine learning algorithms. If the breast cancer could not be diagnosed as soon as possible, treatment for patients would be much more difficult. These cases turned out to be the most important ones.

There were several approaches [26] to solve the imbalanced class distribution problem of SVM classifiers. There are two main groups of methods; one group is called external methods. External methods are data preprocessing methods, which modify degree of imbalanced class ratio up to the desired level before training SVM classifiers. The other group is called internal methods. Internal methods are modifying the main SVM algorithm to reduce its sensitivity to imbalanced class data distribution. This dissertation focuses on re-sampling approaches, which are in the first group. Resampling methods [27] are random under-sampling and over-sampling methods. Resampling methods adjust the balance of majority-class instances and minority-class instances in the datasets before training SVM classifiers. Under-sampling methods will remove the majority-class instances randomly down to balance number of minority-class instances. Disadvantage of under-sampling method discussed in [28] [29] [30] is that important information of majority-class instances might disappear because under-sampling method randomly gets rid of them. Over-sampling methods will randomly duplicate the minority-class instances to obtain the balanced level of both classes. Advantage of over-sampling methods is that there is no information removed that they could lead to better results. This is the main reason that we proposed a new method by over-sampling approach.

Previous external methods with over-sampling methods are reviewed as follows:

One of the most popular methods is SMOTE [31] that over-samples the minority-class instances by creating new synthetic minority-class instances using interpolation technique. SMOTE interpolates k nearest neighbors of randomly selected existing minority-class instances in random area.

Borderline-SMOTE [32] is an alternative of SMOTE but the difference is that Borderline-SMOTE interpolates k nearest neighbors of randomly selected existing minority-class instances focused on the borderline.

BOS [33], [34] was derived from SMOTE but BOS introduces extrapolation technique to develop synthetic minority-class instances by considering fewer amounts of majority-class instances nearest neighbors along the borderline. Interpolation technique was still used when there are higher amount of majority-class instances nearest neighbors along the borderline.

All of these existing methods have something in common that each method finds k nearest neighbors in the input space rather than feature space as illustrated in the Figure 2.5

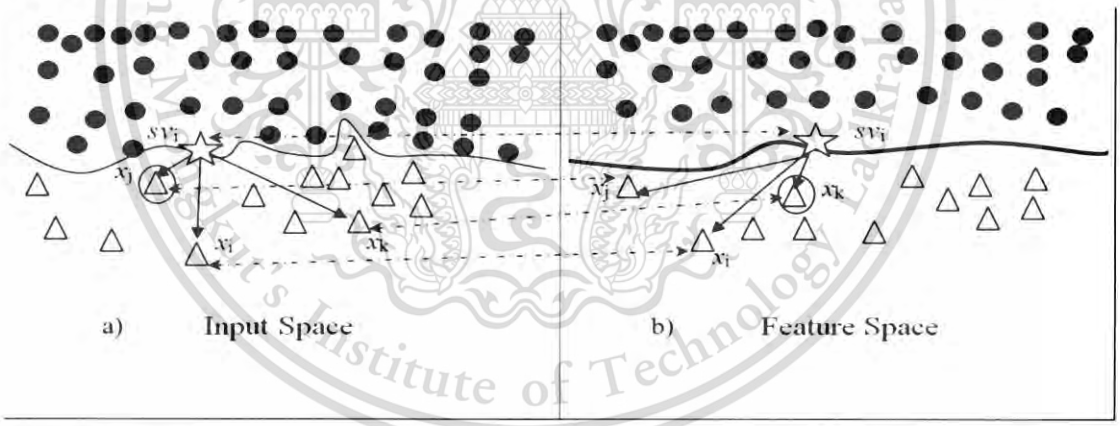


Figure 2.5 Illustration of the difference between previous existing methods and the new proposed method. The arrows represent distance between two points SV_i and any x_i in the input space and this distance representation is also the same in the feature space. The dashed double-sided arrows are between points (x_i , x_j, x_k, SV_i) and their corresponding points in the feature space.

In Figure 2.5 a) Existing methods like SMOTE, Borderline-SMOTE and BOS find the *k* nearest neighbors of minority-class instances (triangles) from any support values (SV_i) on the borderline (solid-line) in the input space. The triangle with circle x_j is the nearest neighbor to the point on the borderline in the input space. x_i , x_j , x_k are

points in the input space In Figure 2.5 b) the newly proposed method finds the k nearest neighbors of minority-class instances (triangles) from any support values (SV_i) on the borderline (solid-line) in the feature space. The triangle with circle x_k is the real nearest neighbor to the point on the borderline in the feature space. x_j is not the real nearest neighbor in the feature space. This is the main difference between previous existing methods and the newly proposed method, which will be described in Chapter 3.

Chapter 3
Research methodology

This chapter describes game dataset and details of main research methodology for state change detection model of online game players in the following section.

3.1 Data description

Shen Zhou Online game dataset (SZO dataset)

We have the data sets from Shen Zhou Online game. Shen Zhou Online (SZO) is MMORPG in Taiwan with 355,706 SZO accounts recording from January 1st, 2004 to April 1st, 2007 as generosity of User Joy Technology Co. Ltd. There are 119,082,865 sessions logged in game access log files. This research study focuses on players whose number of player information records is more than 30 days or 1 month. Since SZO gives all new players free trial period of 30 days. SZO datasets are imbalanced data class because continual-playing states are majority-class instances while quit states are minority-class instances.

The imbalanced ratio of SZO dataset (Minority : Majority) is 1 : 12. For example, suppose there are 100 records of online game players with this ratio, this imbalanced ratio means that there are about 8.33 % (1 divided by 12) of online game players who would be in quit states and 91.67 % (11 divided by 12) of online game players who would be in continue-playing states.

3.2 Conceptual framework of state change detection model of online game players

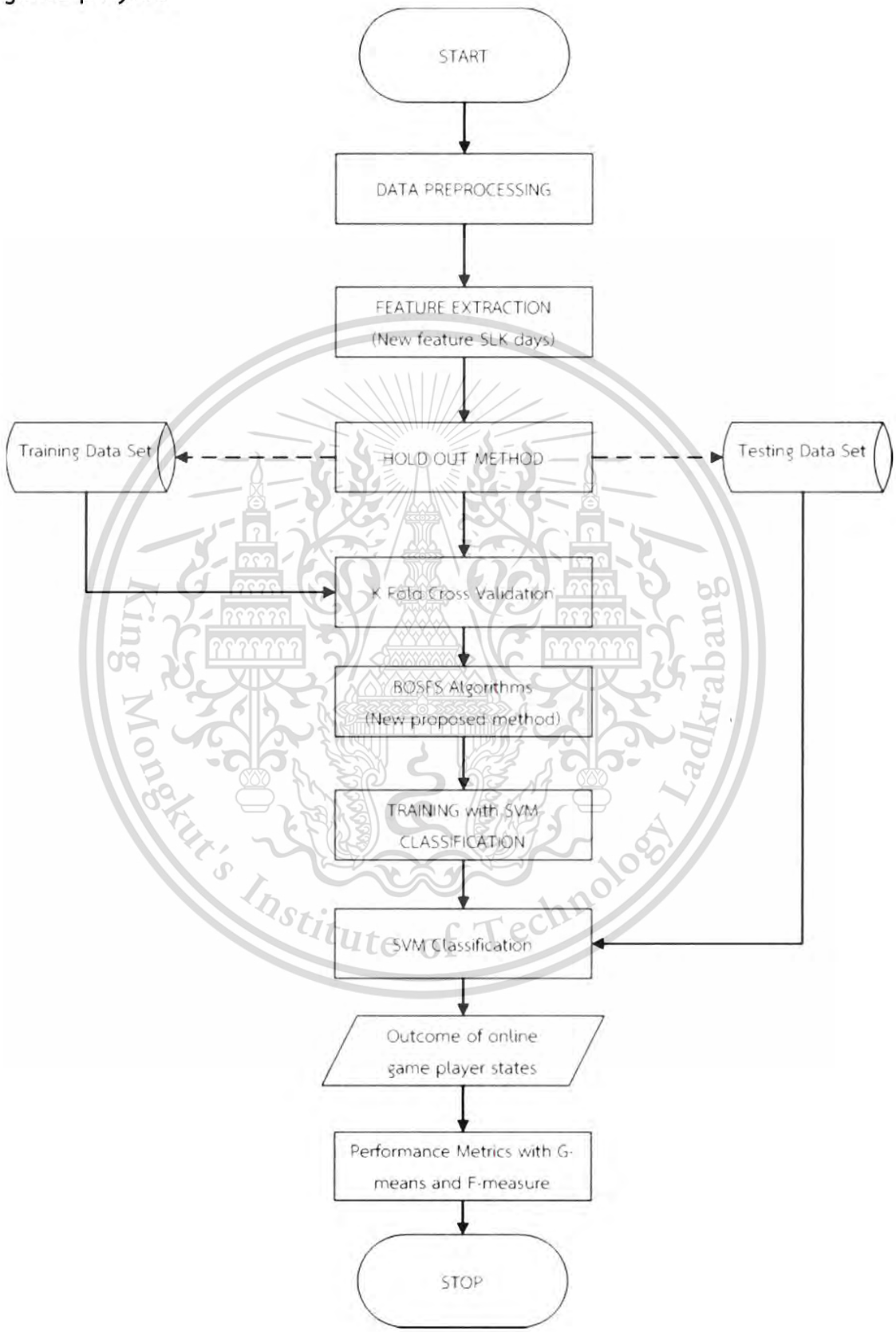


Figure 3.1 State Change Detection Model of Online Game Players

In Figure 3.1, we explained the conceptual framework of the state change detection model of online game players. There are two states of game players as continue-playing state and quit playing state as the outcome of the state change detection model. The model will detect the history access log files of online game players. There are totally eight main steps as data preprocessing, cross validation, BOSFS algorithm, training SVM with training dataset, learning by SVM classifiers, SVM classifiers validating by test dataset, the outcome of state change detection, and performance metrics of the model. The details development of the model is described in the following sections.

3.3 Data preprocessing process

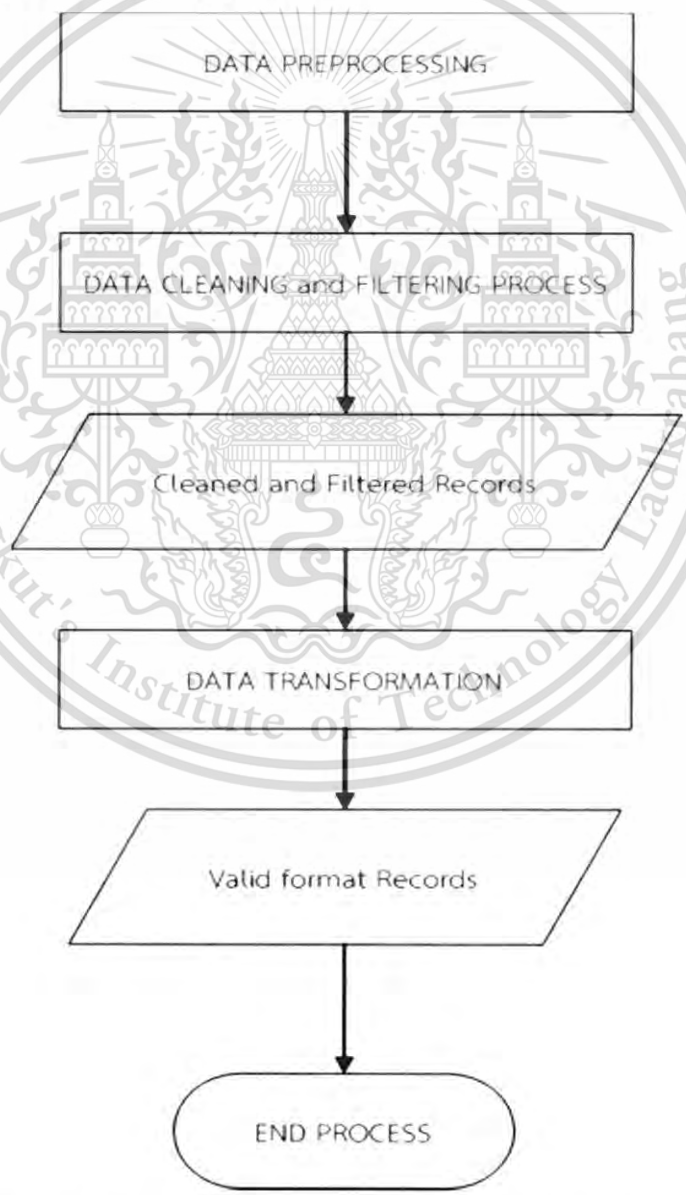


Figure 3.2 Data preprocessing consists of two main processes

At the first step of state change detection model of online game players in Figure 3.2, we need to perform data preprocessing which consists of two main parts. The first part is data cleaning and filtering to SZO datasets in order to gain error-free records, which are important parts. The second part is data transformation of online game player information into the valid format for feature extraction.

3.3.1 Data cleaning and filtering Process

We found that there are some error records in online game player records such as logout time being ahead of login time or duplicate records. These error records happened at the highest network connection, which game servers might record the incorrect time.

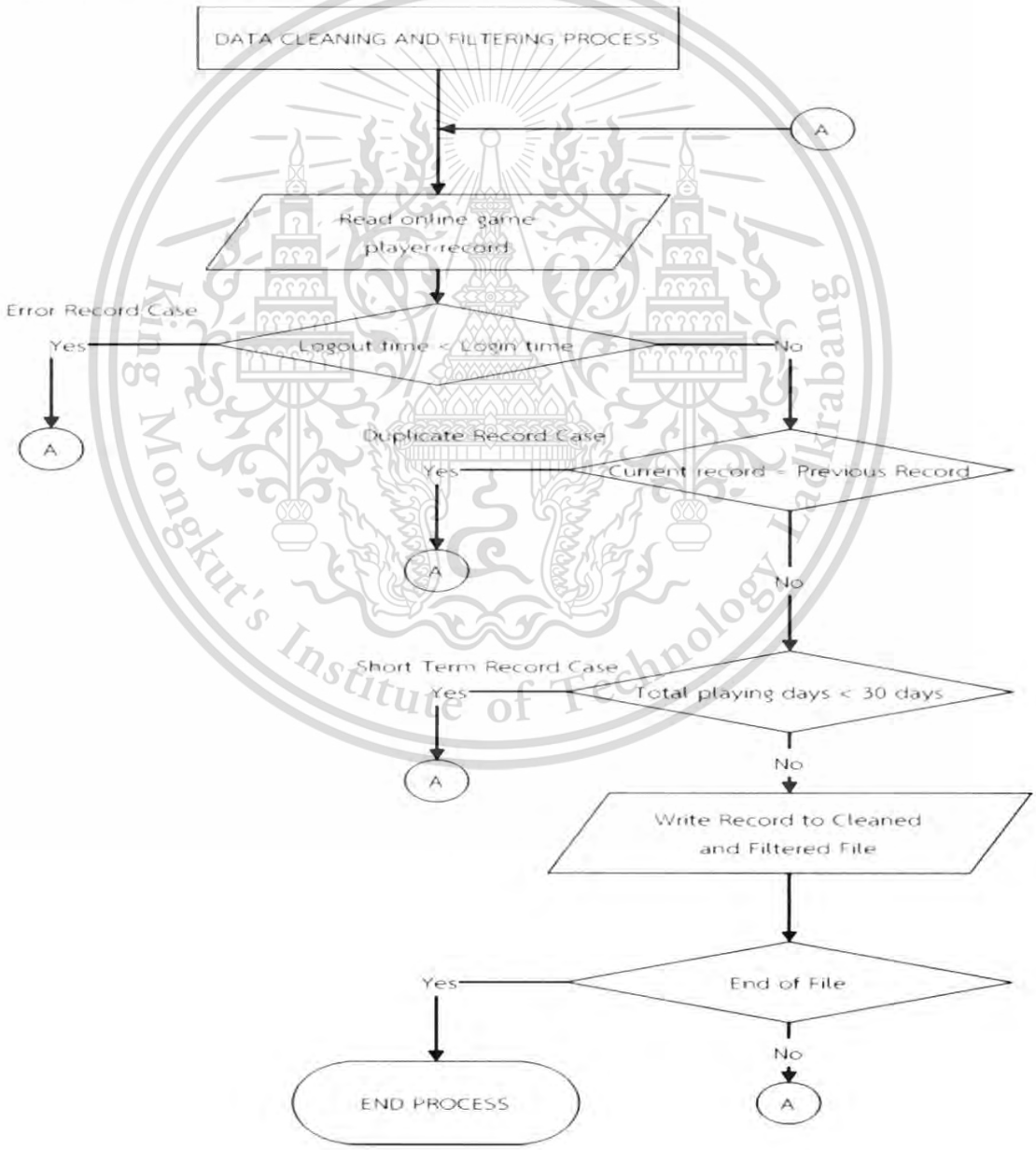


Figure 3.3 Data cleaning and filtering process

In the Figure 3.3, we need to perform filtering out those player information records in short-term period, which were less than 30 days. Since online game companies focus on online game players who played more than 30 days, which was a free-trial period. This process repeats data cleaning and filtering until reading all records to end of file.

3.3.2 Data Transformation

We obtained SZO game access log files. In Table 3.1, it is shown that the original format of login and logout time is in Unix time stamp format which could be incomprehensible. In Table 3.2, data transformation of login and logout time must be conducted to be more understandable as date and time in simplified format.

Table 3.1 SZO game example records of player id = 28 in original format

Record no.	Player ID	Login Time	Staytime (seconds)	Logout Time
100	28	1085934420	9540	1085943960
101	28	1085944140	43080	1085987220
102	28	1086005280	67920	1086073200
103	28	1086118620	10500	1086129120

Table 3.2 SZO game example records of player id = 28 in simplified format

Record no.	Player ID	Login Time	Staytime (seconds)	Logout Time
100	28	30/05/2004 16:27	9540	30/05/2004 19:06
101	28	30/05//2004 19:09	43080	31/05/2004 07:07
102	28	31/05/2004 12:08	67920	01/06/2004 07:00
103	28	01/06/2004 19:37	10500	01/06/2004 22:32

3.4 Feature extraction

This is the second step in the Figure 3.4, which illustrates feature extraction process; there are two important features in our main methodology.

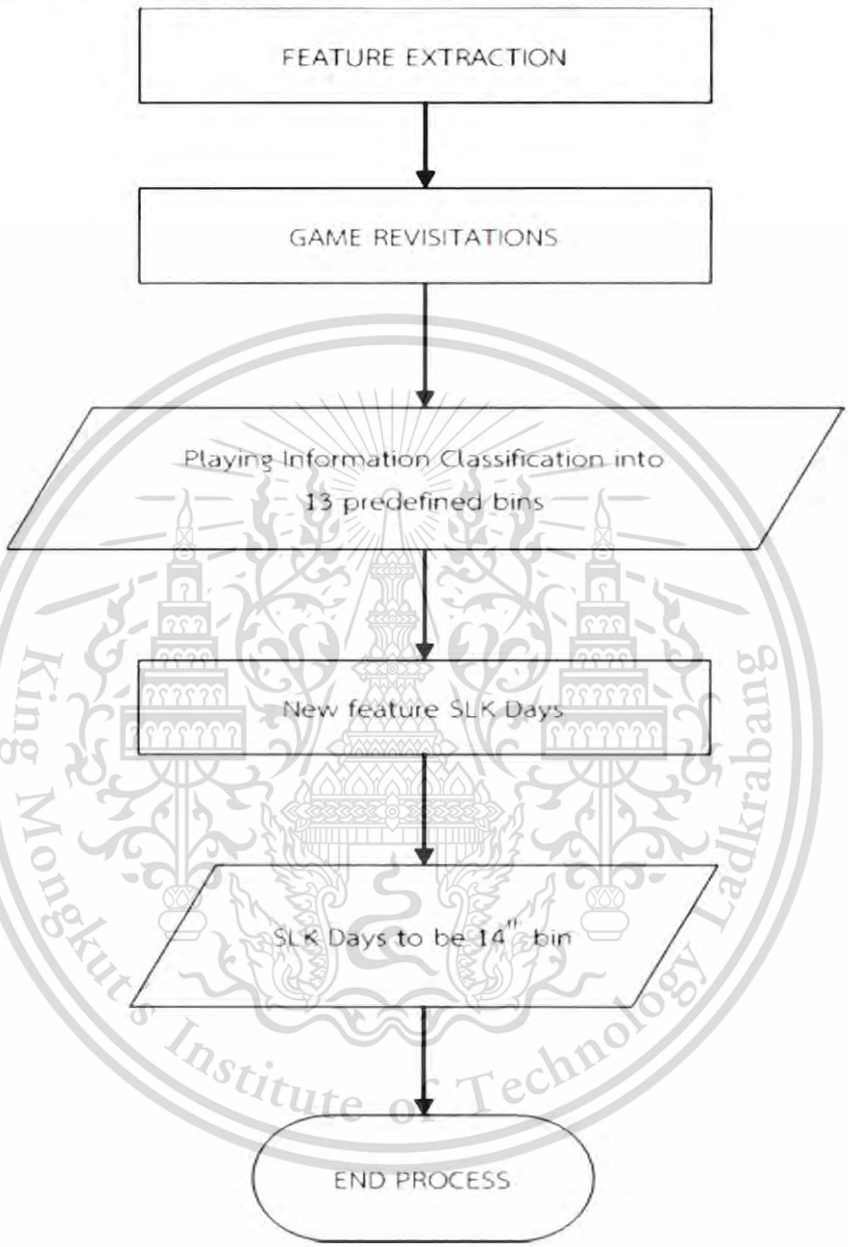


Figure 3.4 Feature Extraction

3.4.1 Game revisitations

Online game players' information are classified into 13 predefined bins of game revisitations [2] as we explained them in section 2.7.3 of Chapter 2. Game revisitations are applied as one important feature to collect online game players' revisitations to the game. This feature needs to have an additional feature for better

prediction for state of online game players for which we proposed a new feature SLK days in the next section to improve game revisitations to be more effective in prediction and classification.

3.4.2 New feature SLK days

New feature was proposed namely "SLK days" stands for "Staytime of Last K days" which means the staytime of game player recently spent in playing the game last K days. SLK days feature [36] is described into three steps as follows:

3.4.2.1 First step, **Staytime** [36] defined as:

$$Staytime = logout\_time - login\_time \tag{3.1}$$

where logout_time is the time game player logout from the game and login_time is the time game player login to the game.

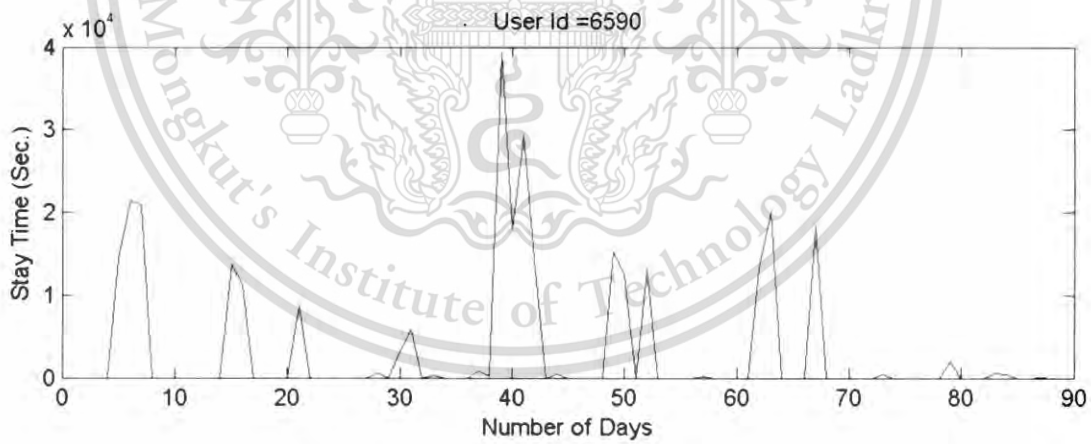


Figure 3.5 Example of daily staytime of user ID 6590

In Figure 3.5, an example of daily staytime of user id 6590 showed a fact that online game players did not play the game frequently. With this reason as discussed in [36], we need to smooth the daily staytime by separating it in an interval such as every five days and averaging it. We performed logarithmic smoothing [35] to our

new feature because it is shown that there was a one-to-one correspondence between the original point that might be in infinite domain and its stationary point.

In each 30-days interval, taking logarithm of average staytime of last k days as represented in [36] could be written as a following equation.

3.4.2.2 Second Step, **SLKdays** [36] defined as:

$$SLKdays = \log\left(\frac{\sum_{i=1}^{k} S_i}{k}\right) \quad (3.2)$$

where S_i is daily SLK days of last day i = 1, 2, 3, ..., k days

3.4.2.3 Third Step, Compute the **normalization** of game revisitations of each 13 predefined bin refer to the equation in [2] as

$$\frac{Bin[j]of\ player[i]}{average\_of\_Bin[j]\_among\_all\_users} \quad (3.3)$$

where bin[j] of player[i] is the number of times player[i] revisiting SZO game within bin[j]'s interval.

SLKdays feature is considered as the 14th bin. Before training SVM, we need to perform normalization to SLK days with the same equation (3.2).

We have done several experiments in [36] for proof of our new feature "SLK days" could be the important feature when we add the SLK days to be 14th bin together with game revisitations classification of 13 predefined bins. It is shown in Figure 4.1 in Chapter 4 that the variable K = 2 could be the best variable for K days, which represents better performance of accuracy, precision and recall than any other numbers as K variable. "SLK days" could be one of the most important feature for state change detection of online game players.

3.5 Hold Out Method

This is the third step in the conceptual framework in Figure 3.1. We randomly separated original data set into training and testing dataset using hold out method based on 80/20, respectively. There are 20 percent of original dataset, which could be enough, and suitable amount to be preserved as the validation dataset for testing

the state change detection model of online game players. There are 80 percent of original dataset to be training dataset for operation in the third step.

3.6 K Fold Cross Validation

This is the fourth step in the conceptual framework in Figure 3.1. There are two purposes in this step; the first purpose is to reduce weakness of soft-margin equation (2.12) in Chapter 2 by using different C parameter ranged from 0.1 to 1. This is because If we chose C parameter to be too high or too low, it would cause over-fitting or under-fitting problem. The second purpose is to determine the optimal value for σ parameter to be used in Radial basis function (RBF) kernel equation (2.24) in Chapter 2. We need to find optimal value of σ parameter range from 0.1 to 1. Hence, this third step is to determine the optimal value of C parameter and σ parameter for best performance metrics, g-means as equation (2.29) and F-measure as equation (2.30).

The prepared dataset in the third step is the training dataset, which is obtained from the second step. K-fold cross validation is evaluation methodology to the prepared dataset. This methodology randomly separated the dataset into k sub-datasets of equal size of which the distribution of the data class stays unchanged. There is a single sub-dataset which is separated as a test dataset. In addition, there are k-1 remaining sub-datasets combining into the training dataset. We set the value of k as 5 in our experiments because each model will be based on an 80/20 separation of the data. There are 20 percent separation of data for validation test, which could be enough, and suitable amount for test data. The rest of 80 percent sub-datasets would become the training dataset for training in SVM learning algorithms.

With this 5-fold cross validation, we repeat 5 times of experiments for training SVM classifier with each value of C parameter and σ parameter range from 0.1 to 1, each step +0.05. There are 20 values of C parameter and σ parameter. We performed almost 100 experiments to determine the optimal value of C parameter and γ parameter. We collected performanc σ e evaluation of each experiment and we chose the optimal value for C parameter and parameter after training SVM classifier with test dataset. We attained the optimal value of C parameter and σ parameter from this third step and we prepare them for the next step.

3.7 BOSFS Algorithm

This is the fifth step in the conceptual framework. We proposed a new BOSFS (Borderline Over-sampling in Feature Space) algorithm [37], which is the external method of data preprocessing method. BOSFS performed over-sampling with generating new synthetic positive instances by finding the nearest existing neighbors on the borderline in the feature space. The new training dataset combined these new synthetic positive instances with the original training dataset, to overcome the imbalanced data class in SVM classifiers.

BOSFS Algorithm

1. Total number of new synthetic positive instances (T_{new_p}) equals to

Over-sampling rate percent (N / 100) * size of positive instances (P_+)in the training dataset

2. Compute Set of positive instances on the borderline(SV_+) in SVM by training RBF kernel SVM on the Training dataset

3. For each positive instance sv_i in Set of positive instances and each positive instance x_p in the set of positive instances (P_+) in the training dataset

   3.1 Perform RBF kernel function $K(sv_i, x_p)$ [10] in equation (2.24) where σ is user-defined parameter.

$$K(sv_i, x_p) = exp\left(-\frac{\|sv_i - x_p\|^2}{2\sigma^2}\right)$$

   3.2 Compute Euclidean distance d_{it} in the feature space [40] [41] as follows

$$d_{ip} = \sqrt{K(sv_i, sv_i) + K(x_t, x_t) - 2K(sv_i, x_t)} \quad (3.4)$$

   where i not equal to p , i is 1,2,3,... size of SV and

   t is 1,2,3, ... size of P

   3.3 Sort d_{it} in ascending order to find $N(i,t)$ array of k nearest neighbors of the positive instance for each sv_i in set SV in the feature space.

4. For each positive instance sv_i in Set of positive instances (SV_+) and each negative instance x_t in the set of negative instance P_- in the training dataset

   Redo Step 3 to find T_{nn} the total number of negative instances nearest to each sv_i in SV_+ on the borderline area.

If T_{nn} is less than half of the maximum number of negative instances near the sv_i then select extrapolation technique as done in BOS by

Create P_+^{new} new positive instances for new borderline by

$$P_+^{new} = sv_i + \rho(sv_i - N(i,t))$$

where $N(i,t)$ is the t^{th} of positive nearest neighbor of each sv_i in SV_+ and ρ is a random number in range of 0 to 1.

Else (Select interpolation technique as done in SMOTE and Borderline-SMOTE)

Create P_+^{new} new positive instances for new borderline by

$$P_+^{new} = sv_i + \rho(N(i,t) - sv_i)$$

5. Combine set of P_+^{new} which is a set of new synthetic positive instances on the borderline with the original training dataset by union these two set as

New Training Dataset = training dataset $(P) \cup \{P_+^{new}\}$

We explained each step in details as follows:

Step 1: User needs to set percent of the over-sampling rate divided by 100,

Total number of new synthetic positive instances =

Over-sampling rate percent(N/100) * number of positive instances in the training dataset (P_+)

This step means that if we set over-sampling rate =100 % and suppose number of positive instances in the training dataset = 500 then we get

Total number of new synthetic positive instances = (100/100) * 500 = 500

BOSFS will create another 500 new synthesis positive instances to be in a new training dataset.

Step 2: Compute Set of positive instances on the borderline(SV_+) in SVM by training RBF kernel SVM on the Training dataset

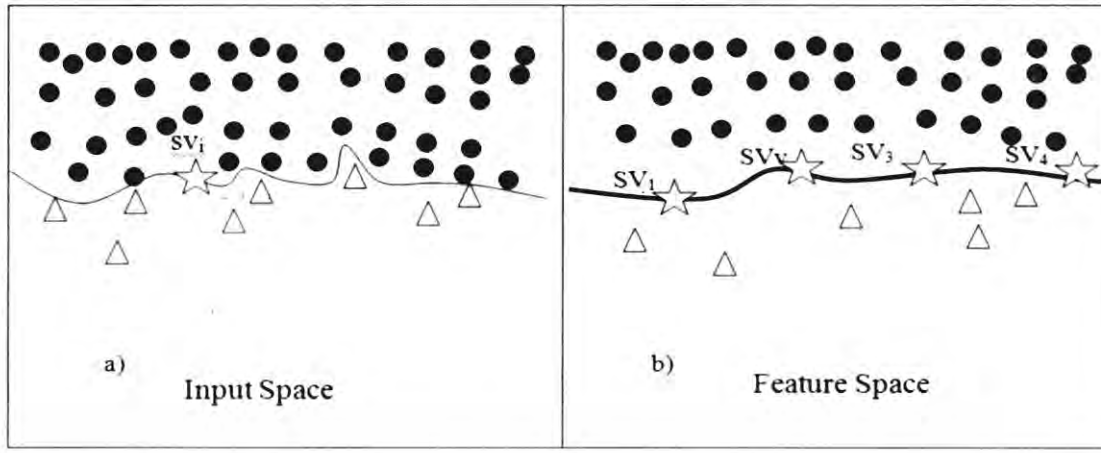


Figure 3.6 Illustration of Step 2 in BOSFS algorithm

In Figure 3.6 a) there is imbalanced class data in the input space. Black circles represent majority-class instances or negative instances. Triangles represent minority-class instances or positive instances. There are any SV_i positive instances(Star) in the input space. In Figure 3.6 b), after we performed step 2 in BOSFS algorithm, there are set of positive instances (Stars) (for example 4 stars: SV_1 , SV_2 , SV_3 , SV_4) on the borderline by training RBF kernel SVM with the training dataset in the feature space. These positive instances (Stars) will be important reference points to create new synthetic positive instances according to number computed at the first step.

Step 3: For each positive instance sv_i in Set of positive instances and each positive instance x_p in the set of positive instances (P_+) in the training dataset

3.1 Perform RBF kernel function $K(sv_i, x_p)$ in equation (2.24) where σ is user-defined parameter.

3.2 Compute Euclidean distance d_{it} in the feature space as equation (3.4)

3.3 Sort d_{it} in ascending order to find $N(i,t)$ array of k nearest neighbors of the positive instance for each sv_i in set SV in the feature space.

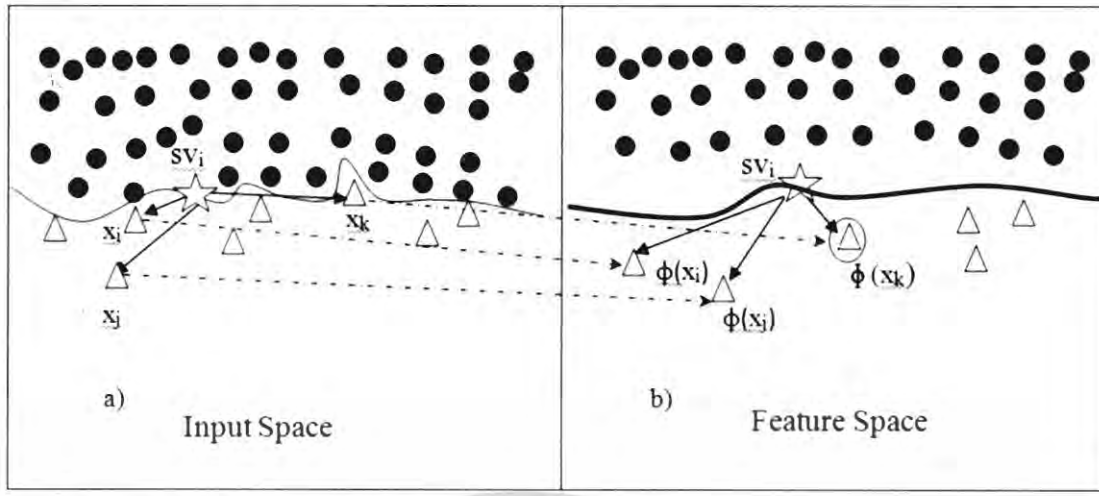


Figure 3.7 Illustration of Step 3 in BOSFS algorithm

In Figure 3.7 a), for any positive instances, (x_i, x_j , x_k) in the input space will be mapped by RBF kernel function to ($\phi(x_i)$, $\phi(x_j)$, $\phi(x_k)$ in the feature space as shown in Figure 3.7 b). Dashed line represents the corresponding point between x_i and $\phi(x_i)$.

In Figure 3.7 a), the arrow represents the distance between SV_i and any nearby positive instances (x_i, x_j , x_k) in the input space. In Figure 3.7 b), the arrow represents the Euclidean distance equation(3.4) between SV_i and any nearby positive instances ($\phi(x_i)$, $\phi(x_j)$, $\phi(x_k)$ in the feature space. We performed k nearest neighbors of positive instance on the borderline (SV_i). We set k to 5 because its performance g-means and F-measure is the best performance of all k values. This is explained in Appendix B.

In Figure 3.7 a), x_i is the nearest neighbor of SV_i on the borderline in the input space while $\phi(x_k)$ (triangle with circle) in Figure 3.7 b) is the real nearest neighbor of SV_i on the borderline in the feature space. With this step, we could find the nearest neighbors of positive instances in the feature space, which appear to be the improvement step for the performance of SVM classifier to deal with imbalanced class data problems.

Step 4: Density of negative instances to determine optimal techniques between Interpolation and Extrapolation Techniques.

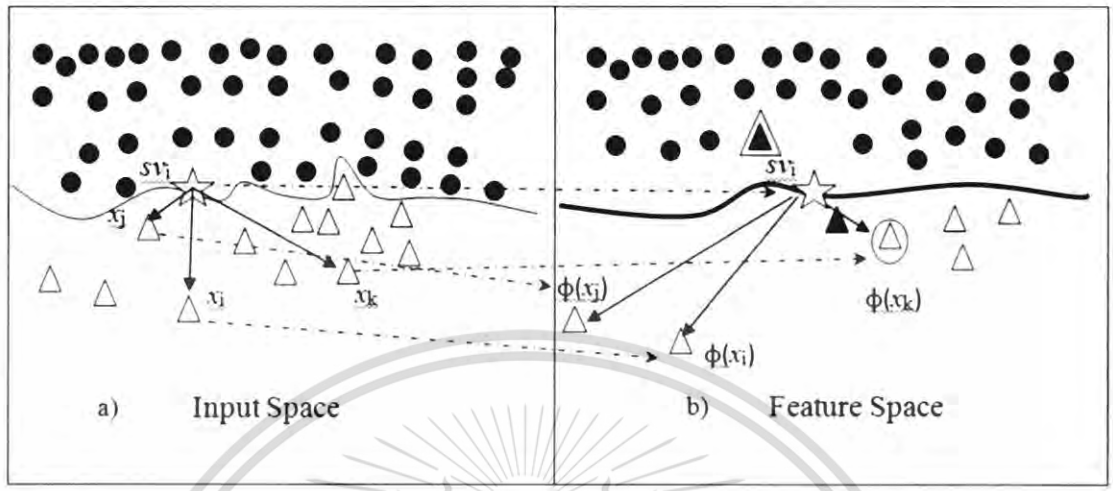


Figure 3.8 Illustration of Step 4 in BOSFS algorithm

Figure 3.8 illustrates that if density of negative instances is less than half of maximum negative instances on the borderline, we select extrapolation technique; the black triangle surrounded by white triangle is the result position of new synthetic instances created by reference points SV_i. It shows that with extrapolation technique, new synthetic instance will go further beyond the borderline, which could expand the new borderline to improve prediction performance of SVM classifiers. If density of negative instances is greater than half of maximum negative instances on the borderline, we select interpolation technique that black triangle is the result position of new synthetic instances created by reference points SV_i.

We repeat Step 3 and 4 until BOSFS generated the total number of new synthetic positive instances according to specified over-sampling rate at the first step.

Step 5: Combined the set of new positive instances with the original training dataset into new training dataset as the result of algorithm. This new training dataset is ready for training SVM algorithms in the next section.

Main Difference between BOSFS method and the existing methods.

There are some existing methods, SMOTE [31], Borderline-SMOTE [32] and BOS [33] [34] algorithms in which the main differences between them and BOSFS Algorithm are

- BOSFS finds the real nearest neighbor of the points on the borderline in the feature space but the existing methods, SMOTE, Borderline-SMOTE and BOS find them in the input space as shown in Figure 2.5 in Chapter 2.

- BOSFS uses kernel method as Radial Basis Function (RBF) Kernel equation (2.24) and Euclidean distance equation (3.4) to find the real nearest neighbors of minority-class instances focusing on the borderline but those existing methods use only Euclidean distance for finding nearest distance in the input space.

- BOSFS applies a combination of interpolation and extrapolation techniques by creating new way for considering suitable technique between interpolation and extrapolation in order to create synthetic minority-class instances on the borderline. BOSFS determines the density of majority-class instances in the feature space by using RBF kernel function equation (2.24) with majority-class instances while those existing methods just collected numbers of majority-class instances along the borderline in the input space. BOSFS applies interpolation technique if the density of majority-class instances is greater than half of the maximum majority-class instances near the borderline (as done in SMOTE and Borderline-SMOTE). BOSFS applies extrapolation technique if the density of majority-class instances is less than half of the maximum majority-class instances near the borderline (as done in BOS).

BOSFS algorithm will synthesize new minority-class instances or new positive instances with focusing on the borderline in the Figure 2.4. The results of new positive instances added to the original training dataset could expand the new borderline for Support Vector Machines (SVM) to be closer to the ideal borderline (dashed line) than the original borderline as shown in Figure 3.9.

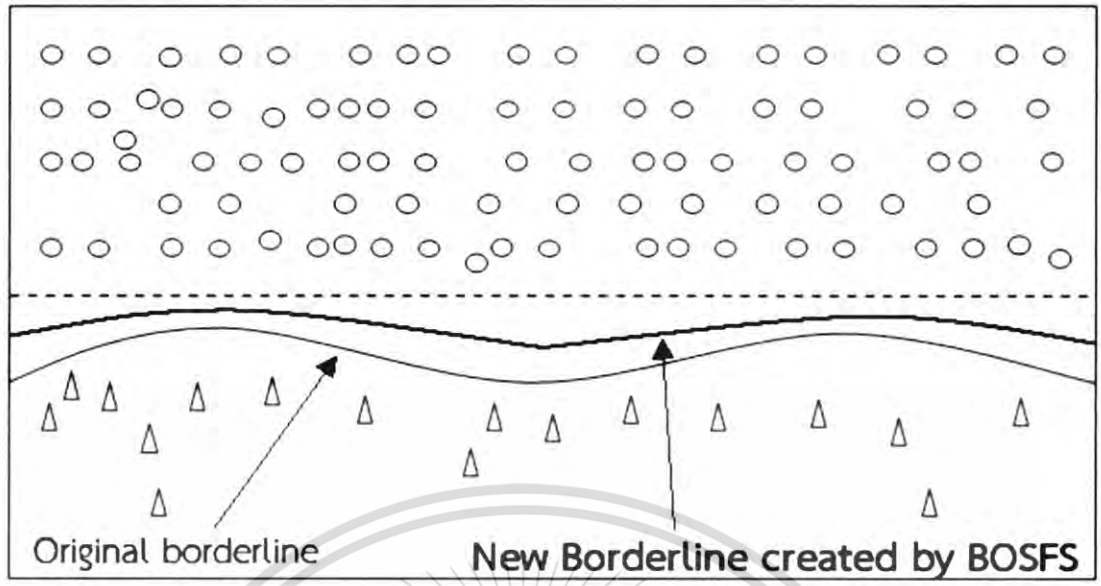


Figure 3.9 New borderline created by BOSFS Algorithm

3.8 New training dataset

New training dataset combines the original training dataset with a set of new synthetic positive instances according to over-sampling rate specified by user at the first step of BOSFS. The new training dataset will be balanced class dataset due to BOSFS algorithm. The new training dataset is well prepared for the learning classifier in the next step.

3.9 Training with SVM classification

This is the sixth step in conceptual framework in Figure 3.1. New training dataset will be training by SVM classifier as explained in Chapter 2 to find set of support vectors in equation (2.18) for non-linear separable case. We use Matlab to train SVM classifier with RBF kernel function in equation (2.24). The output of this step is Learning SVM classifier, which learns new training dataset as support vectors ready to classify the unknown testing dataset.

3.10 SVM classification with new training dataset

This is the seventh step in conceptual framework in Figure 3.1. New training dataset is trained with SVM to attain support vectors. After training SVM with training dataset and two features as game revisitations and SLK days, SVM classifier is ready to classify the test dataset which we use hold out method to separate this test dataset at the previous step of cross validation. At the seventh step, the outcome of state change detection of online game players could be continue-playing state or quit state according to pattern recognition of game revisitations and SLK days, for which SVM learns from the new training dataset.

3.11 Performance metrics

This is the eighth step in conceptual framework in Figure 3.1. We will apply g-means by equation (2.29) and F-measure by equation (2.30). This two performance metrics are suitable for imbalanced data class distribution as done in [13] [14] [15] and [16]. We conduct several experiments with the state change detection model of online game players and the comparison results with performance metrics to the existing methods, SMOTE, Borderline-SMOTE and BOS in Chapter 4.

Chapter 4
Main Results and Discussion

In this chapter, we report main experimental results into three sections. In section 4.1 we present the results of experiments with new proposed feature SLK days in order to find the optimal k days for the best performance of accuracy of all ten cases where k days = 0, 2, 3,... 10 days. This section will demonstrate the optimal k value of SLK days to be used as an important feature in our proposed model. In section 4.2, we combined the new proposed BOSFS algorithms and new feature SLK days with optimal k days from previous section into state change detection of online game players. We report the results of experiments of the proposed model with SZO dataset of different existing methods, SMOTE, BOS comparing to our new proposed BOSFS algorithms. In section 4.3, we applied BOSFS algorithms without SLK days feature to other domain datasets by using five UCI datasets to see the results whether BOSFS could still perform better than the existing methods. We report the results of experiments with different five datasets such as Abalone, Glass, Page, Spect and Yeast dataset. We compared the experimental results of several experiments with existing methods, SMOTE, Borderline-SMOTE, BOS to our new proposed BOSFS as proof of BOSFS is more efficient method than those methods.

4.1 Experimental results of new proposed feature (SLK days) to determine optimal k value for SLK days.

Staytime of Last K days (SLK days) is our new feature extraction which considers staytime of last k days of online game players in history game access log file as we explain SLK days feature in section 3.3.3.2 of Chapter 3. We conducted several experiments with different k days from 2,3,... 10 days to determine the optimal value of k days which will be the best performance of accuracy for SLK days feature. There are four experiments with different datasets for training and testing datasets to test the hypothesis that those additional new feature SLK days to game revisitations, which could improve the prediction performance of SVM classifiers.

In Table 4.1, it shows SZO dataset with various years forming training data set and test data set. We performed four experiments with a variation of k days for ten cases in order to determine the best k days for the best departure prediction of online game players as shown in Table 4.2 through 4.5.

Table 4.1 Different year (y) for training and test data set in four experiments.

Experiment No.	Test dataset (Year)	Training dataset	
		y-2	y-1
1.1	2005	-	2004
1.2	2006	-	2005
2	2006	2004	-
3	2006	2004	2005

From experiment 1.1, we trained SVM classifiers with training dataset in year 2004. We conducted 10 cases for SLK days feature to the test dataset in year 2005. The first nine cases where k = 2, 3, ... 10 and the last case is the case with only game revisitations feature without SLK day feature (case k = 0) for which we want to see the experimental results of the performance of the departure prediction in terms of accuracy, recall and precision as shown in Table 4.2.

Table 4.2 Experimental results of 10 cases in experiment 1.1.

SLK days	k =0	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
Accuracy %	58.05	**92.64**	90.89	87.85	88.05	85.12	82.98	81.62	80.68	79.81
Recall %	69.50	**86.22**	81.96	74.87	74.78	68.81	64.14	61.79	59.70	58.06
Precision %	53.09	**97.35**	95.68	95.74	93.59	91.65	89.72	87.97	87.80	87.87

In Table 4.2, the best performance of experiment 1.1 for feature SLK days of ten cases is when k = 2 days and the performance of accuracy = 90.89%, recall = 86.22 % and precision = 97.35%. When k = 0, it means that we use only game revisitations feature without SLK days. The results of performance metrics is accuracy = 58.05%, recall = 69.50 % and precision = 53.09%. This concludes that without SLK days, the accuracy and precision have dropped to 58.05 % and 53.09 %, respectively.

From experiment 1.2, we trained SVM classifiers with training dataset in year 2005. We conducted totally 10 cases for SLK days feature to the test dataset in year 2006. The experimental results of the performance of the departure prediction in terms of accuracy, recall and precision are shown in Table 4.3.

Table 4.3 Experimental results of 10 cases in experiment 1.2.

SLK days	k =0	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
Accuracy %	47.66	**91.92**	89.93	88.43	86.98	84.77	82.96	81.61	80.59	79.50
Recall %	**99.80**	84.51	79.43	75.43	72.24	68.01	66.26	65.98	65.70	68.60
Precision %	47.54	**98.22**	97.40	96.78	95.39	93.48	90.88	88.40	85.93	81.64

In Table 4.3, the best performance of experiment 1.2 for feature SLK days of ten cases is when k = 2 days that accuracy = 91.92%, recall = 84.51 % and precision = 98.22 %. When k = 0, this means that we use only game revisitations feature without SLK days. The result of performance metrics is accuracy = 47.66 %, recall = 99.80 % and precision = 47.54 %. This means that without SLK days, the accuracy and precision has dropped to 47.66 % and 47.54 %, respectively but Recall = 99.80% which is the highest of all. This might happen because the accuracy and precision are quite low.

From experiment 2, we trained SVM classifiers with training dataset in year 2004. We conducted totally 10 cases for SLK days feature to the test dataset in year 2006. The experimental results of the performance of the departure prediction in terms of accuracy, recall and precision are shown in Table 4.4.

Table 4.4 Experimental results of 10 cases in experiment 2.

SLK days	k =0	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
Accuracy %	54.44	**91.43**	89.63	87.84	86.45	84.30	82.09	80.38	79.49	78.43
Recall %	64.79	**83.63**	78.78	74.87	71.34	66.88	62.23	59.83	57.72	55.81
Precision %	51.59	**98.03**	97.33	95.74	94.82	93.38	93.00	91.77	91.56	91.64

In Table 4.4, the best performance of experiment 2 for feature SLK days of ten cases is when k = 2 days and accuracy = 91.43%, recall = 83.63 % and precision = 98.03 %. When k = 0, it means that we use only game revisitations feature without SLK days. The result of performance metrics is accuracy = 54.44 %, recall = 64.79 % and precision = 51.59 %. This means that without SLK days, the accuracy and precision have dropped to 54.44 % and 51.59 %, respectively

From experiment 3, we trained SVM classifiers with training dataset in both year 2004 and 2005. We conducted totally 10 cases for SLK days feature to the test dataset in year 2006. The experimental results of the performance of the departure prediction in terms of accuracy, recall and precision are shown in Table 4.5.

Table 4.5 Experimental results of 10 cases in experiment 3.

SLk days	k =0	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
Accuracy %	64.67	**91.36**	89.73	88.47	86.64	84.30	84.63	80.69	79.75	78.59
Recall %	57.19	**84.76**	79.75	76.43	72.40	66.88	68.89	60.89	61.44	63.81
Precision %	64.37	**96.63**	96.50	95.72	94.19	93.38	92.06	91.44	87.92	83.36

In Table 4.5, the best performance of experiment 3 for feature SLK days of ten cases is when k = 2 days and accuracy = 91.36%, recall = 84.76 % and precision = 96.63 %. When k = 0, this means that we use only game revisitations feature without

SLK days. The result of performance metrics is accuracy = 64.67 %, recall = 57.19 % and precision = 64.37 %. This means that without SLK days, the accuracy and precision have dropped to 64.67 % and 64.37 %, respectively.

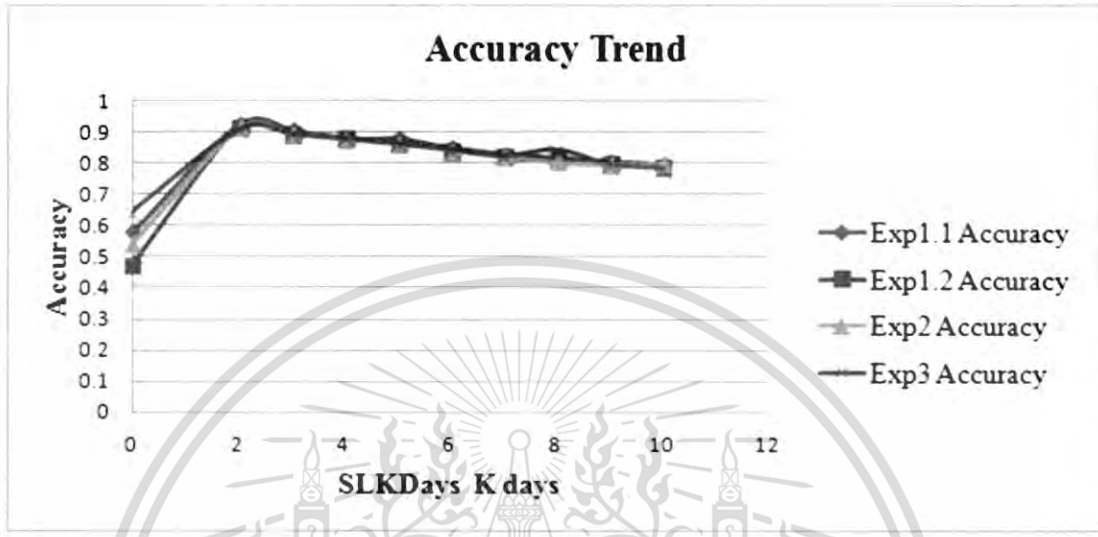


Figure 4.1 Accuracy trend in ten cases of SLK k days of 4 experiments

Figure 4.1 shows the accuracy trend in ten cases of feature SLK days where k = 0 and k = 2, 3 ... 10 of total 4 experiments with different year of training dataset and test dataset in Table 4.1.

In summary, from experiment 1.1, 1.2, 2 and 3, our new proposed feature SLK days when k = 2 days accomplishes the best performance of accuracy of all ten cases of k days. We analyzed that this happened because staytime of last k days if k =2 is the latest information of each online game player. However, if k is greater than 2, it is not the latest information. But when k = 1 day, learning SVM classifier could not classify any output because there is not enough information for learning algorithm of SVM to predict the result. The graph of each experiment also shows that when k = 0 days which means that for the experiments without feature SLK days, the performance accuracy of all experiments have dropped to less than 70 %. This means that our new proposed feature SLK days has the significant efficient feature in the state prediction of online game players.

4.2 Experimental Results with new BOSFS algorithm and SLK days feature in state change detection model of online game players and discussion

We combined new proposed feature SLK days and new proposed BOSFS algorithm to be two important features of state change detection of online game players. The hypothesis we need to validate is that the state change detection of online game players with BOSFS algorithms could perform better prediction than the model applied with the existing methods, SMOTE and BOS. We conducted several experiments using SZO dataset to gain the experimental performance metrics results of three methods as shown in the Table 4.6. The imbalanced ratio of SZO dataset is 1:12.

Table 4.6 SZO dataset: acc^+, acc^-, g-means, and F-measure of SMOTE, BOS and BOSFS algorithm using k =5 nearest neighbors.

Algorithms / metrics	Imbalance Ratio (Minority : Majority)											Average
	2:12	3:12	4:12	5:12	6:12	7:12	8:12	9:12	10:12	11:12	12:12	
SMOTE												
acc^-	70.85	76.93	80.16	82.41	84.58	86.55	87.09	87.87	88.65	89.27	89.49	83.99
acc^+	92.15	89.33	87.00	84.48	83.66	83.61	82.38	81.32	80.12	80.79	78.77	83.96
g-means	80.80	82.90	83.51	83.44	84.11	85.07	84.70	84.52	84.27	84.91	83.96	83.84
F-measure	74.88	79.94	82.51	84.11	86.15	87.86	88.43	89.07	89.58	90.34	90.44	85.76
BOS												
acc^-	81.72	88.43	**91.78**	92.63	93.46	**94.56**	**95.59**	95.34	95.79	95.93	96.42	92.88
acc^+	93.51	93.95	91.79	91.89	92.03	92.63	91.69	91.54	90.89	88.83	90.65	91.76
g-means	87.38	91.14	91.77	92.24	93.15	93.58	93.61	93.70	93.87	92.57	93.73	92.43
F-measure	81.66	89.10	90.76	92.31	93.94	94.65	95.14	95.59	96.10	95.44	96.35	92.82
BOSFS												
acc^-	**83.31**	**88.51**	90.87	**93.46**	**94.30**	94.50	95.54	**95.92**	**96.97**	**96.53**	**96.98**	**93.35**
acc^+	**94.24**	**95.40**	**95.57**	**96.07**	**96.36**	**95.67**	**96.13**	**96.28**	**95.90**	**96.56**	**96.33**	**95.86**
g-means	**88.56**	**91.88**	**93.17**	**94.13**	**94.90**	**95.08**	**95.83**	**95.80**	**95.84**	**96.24**	**96.37**	**94.35**
F-measure	**83.57**	**90.49**	**92.79**	**94.24**	**95.30**	**95.82**	**96.60**	**96.66**	**96.88**	**97.18**	**97.43**	**94.27**

In Table 4.6, the values in bold correspond to the best value attained in each experiment. It is clear that state change detection model of online game players with BOSFS algorithm achieves the greatest number of superior values. We observed that acc^+ of BOSFS algorithm at some imbalanced ratio as (4:12), (7:12) and (8:12) are only inferior to acc^+ of BOS, but still better than acc^+ of SMOTE.

After we conducted several experiments with the state change detection model of online game players using different methods, SMOTE, BOS and our proposed BOSFS, we plotted g-means of different methods as shown in Figure 4.2.

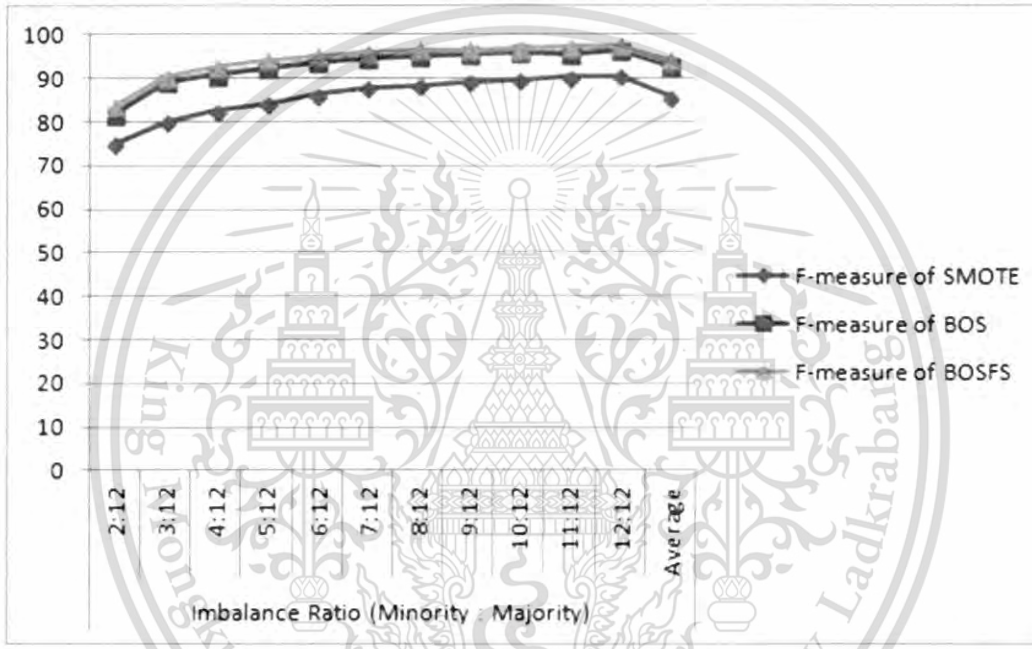


Figure 4.2 the comparison results of g-means graphs using SMOTE, BOS and BOSFS methods with different eleven imbalanced ratios of SZO dataset and average of all ratios.

Figure 4.2 illustrates that g-means of BOSFS are best of all existing methods in each different imbalanced ratio of SZO dataset.

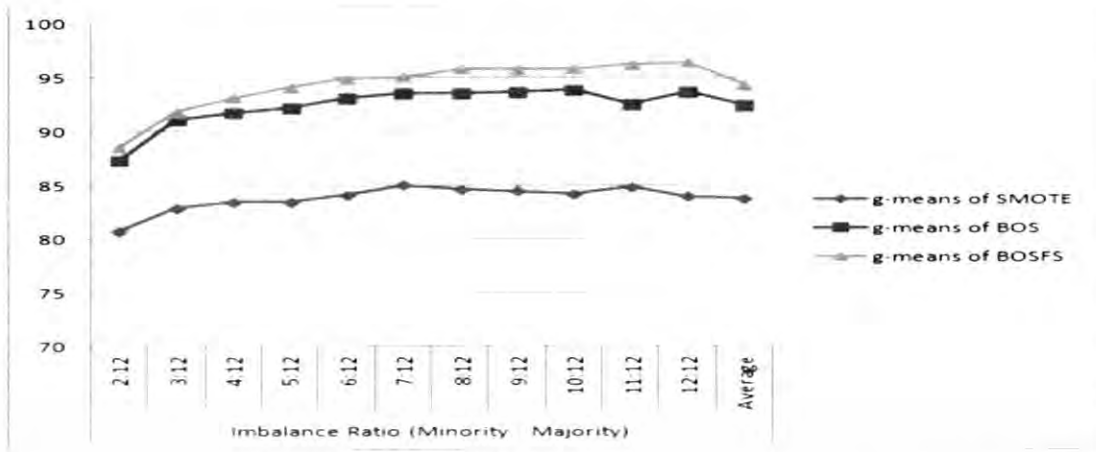


Figure 4.3 the comparison results of F-measure graphs using SMOTE, BOS and BOSFS methods with different eleven imbalanced ratios of SZO dataset and average of all ratios.

In Figure 4.3, it is obviously demonstrated that F-measure graph of BOSFS is the best of F-measure graphs of all existing methods in each different imbalanced ratio of SZO dataset. F-measure graph of BOSFS is slightly better than F-measure graph of BOS method, but it is much better than the graph of SMOTE.

In summary, our proposed state change detection of online game players with BOSFS method accomplishes better g-means and F-measure performance than the SMOTE and BOS at almost over-sampling rates and imbalanced ratios.

4.3 Experimental Results of new BOSFS algorithm with five UCI datasets and discussion

After we obtained the better experimental results of our proposed model with new BOSFS method in section 4.2, we came up with the new hypothesis that BOSFS can apply to other data domain than SZO dataset. Then we used five UCI datasets which are standard datasets from UCI machine learning repository of University of California, Irvine [38] to test this hypothesis. We performed several experiments with different over-sampling rate in Table 4.6 for testing the performance of BOSFS. We conducted experiments by comparison of BOSFS with existing methods, SMOTE and BOS. We performed additional experiments using Borderline-SMOTE. The experimental results are shown in the Table 4.7.

Table 4.7 Five UCI datasets with different over-sampling rate (%)

Dataset	Attributes	No. of Instances	Imbalanced ratio	Over-sampling rate (%)
Abalone	8	4177	35	100, 500, 1000, 1500, 2000, 2500, 3000, 3400
Glass	9	214	6	100, 200, 300, 400, 500
Page-blocks	10	267	61	100, 1000, 2000, 3000, 4000, 5000, 6000
Spect	22	267	4	100, 200, 300
Yeast	8	1484	28	100, 500, 1000, 1500, 2000, 2500, 2700

In Table 4.7, we selected these five UCI datasets because they had different imbalanced ratios, which varied from 4, 6, 28, 35 and 61. Other reasons were the variation numbers of instances and attributes of these five datasets that could be suitable datasets for training and test dataset of our experiments to compare our new BOSFS algorithms to the existing methods like SMOTE, BORDER-SMOTE, BOS.

In Table 4.7, for example, the imbalanced ratio of Abalone dataset is 35. Suppose there are 100 positive instances in Abalone dataset, this means that there are 3,500 negative instances in the dataset as the ratio of minority and majority class instances is 1:35. If we choose over-sampling rate as 100 % of positive class instances after we combined new 100 synthetic positive class instances into the training dataset, there will be 200 positive instances where the new imbalanced ratio of minority and majority would be 2:35. This is the basic concept of over-sampling methods for all existing methods including our new BOSFS method.

4.3.1 Experimental Results of BOSFS with Abalone dataset

We conducted many experiments to compare the performance metrics as g-means, F-measure, acc^{+} and acc of the existing methods, SMOTE, Borderline-SMOTE and BOS with our new proposed BOSFS method as shown in Table 4.8 through Table 4.12 with different over-sampling rates to each dataset as displayed in Table 4.7

Table 4.8 Abalone dataset with performance metrics: acc^+, acc^-, g-means and F-measure of SMOTE, Borderline-SMOTE, BOS and BOSFS using k = 5 nearest neighbors.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)								
	2:35	6:35	11:35	16:35	21:35	26:35	31:35	35:35	Average
SMOTE									
acc^+	31.97	60.18	75.59	82.75	85.91	87.79	89.69	90.39	73.41
acc^-	**98.62**	99.08	97.85	98.15	**98.6**	98.21	98.7	98.89	98.46
g-means	56.08	77.2	85.98	90.08	92.03	92.85	94.09	94.54	84.04
F-measure	44.81	74.42	84.86	89.74	91.95	92.98	94.25	94.69	81.86
Borderline-SMOTE									
acc^+	37.11	63.08	74.16	80.29	85.08	86.98	**93.19**	89.61	76.19
acc^-	95.97	95.33	98.45	98.95	98.41	**98.79**	91.8	**99.58**	97.16
g-means	59.32	77.19	85.44	89.13	91.49	92.69	92.48	94.46	85.28
F-measure	38.94	65.89	84.37	88.67	91.44	92.7	94.04	94.43	81.31
BOS									
acc^+	35.13	61.24	74.79	81.67	85.04	87.75	89.47	90.13	75.66
acc^-	97.61	**99.53**	**99.19**	**99.01**	98.06	98.34	**98.87**	99.42	**98.66**
g-means	58.48	78.06	86.12	89.93	91.31	92.89	94.05	94.66	85.63
F-measure	45.1	75.6	85.14	89.53	91.28	93.02	94.17	94.68	83.51
BOSFS									
acc^+	**47.61**	**72.41**	**81.64**	**86**	**89.09**	**90.93**	91.58	**92.31**	**79.89**
acc^-	97.05	96.16	98.02	98.44	98.31	98.51	97.75	98.86	97.75
g-means	**67.94**	**83.41**	**89.45**	**92.01**	**93.59**	**94.64**	**94.61**	**95.53**	**87.95**
F-measure	**55.54**	**79.22**	**88.7**	**91.79**	**93.64**	**94.8**	**94.99**	**95.75**	**85.53**

In Table 4.8, we conducted experiments with Abalone dataset at the imbalanced ratio of 35. The values of bold numbers relate to the best values obtained in each existing method. It is obvious that BOSFS method accomplishes the greater numbers of bold values than SMOTE, Borderline-SMOTE and BOS. There are only some values that other methods are better. For example, acc^+ of SMOTE at imbalanced ratio of 2:35 and 21:35 is best of all methods while acc^- of Borderline-SMOTE are better at imbalanced ratio of 26:35 and 35:35. acc^+ of BOS at imbalanced ratio of 6:35, 11:35, 16:35 and 31:35 are best of all methods. acc^- is the accuracy of negative instances which means the accuracy of majority-class instances prediction.

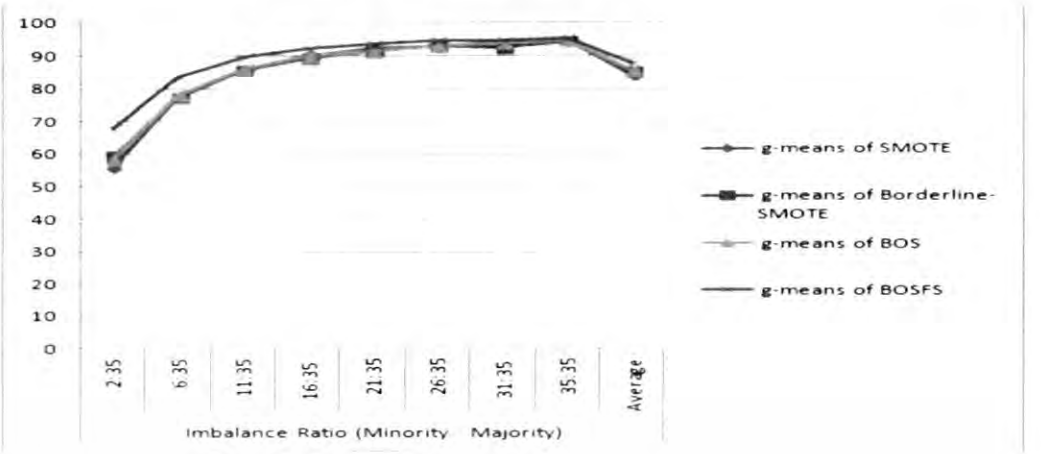


Figure 4.4 the comparison results of g-means graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different eight imbalanced ratios of Abalone dataset and average of all ratios.

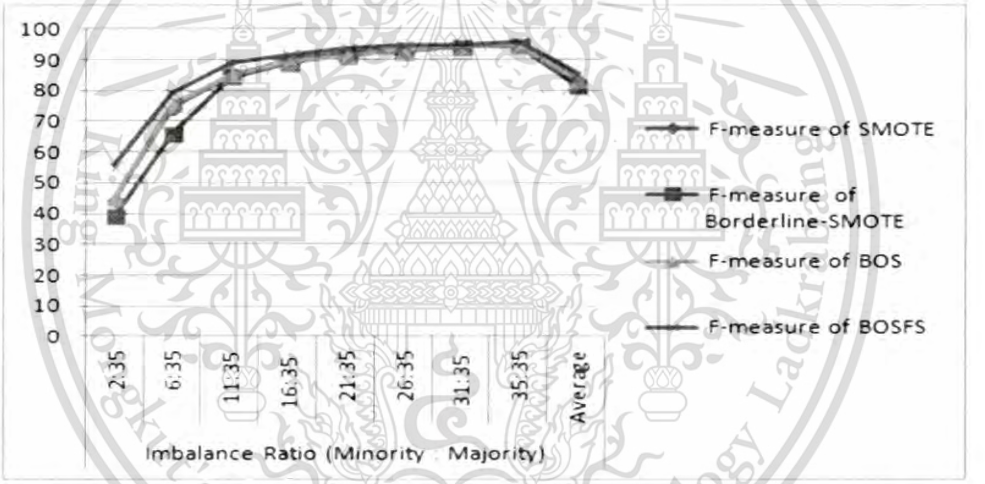


Figure 4.5 the comparison results of F-measure graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different eight imbalanced ratios of Abalone dataset and average of all ratios.

In Figure 4.4 and 4.5, it is obviously demonstrated that g-means and F-measure graph of BOSFS are best of all existing methods in each different imbalanced ratio of Abalone dataset. The best F-measure of all imbalanced ratios is at the imbalanced ratio of 35:35, which means that the ratio of dataset becomes 1:1 or balanced dataset for effective SVM classifiers.

4.3.2 Experimental Results of BOSFS with Page dataset

In Table 4.9, we conducted experiments with Page dataset at the imbalanced ratio of 61. The values of bold numbers relate to the best values obtained in each existing method. It is obvious that BOSFS method accomplishes the greater numbers of bold values than SMOTE, Borderline-SMOTE and BOS. There are only some values that BOS methods are better. For example, acc^+ of BOS at imbalanced ratio of 21:61, 31:61, 41:61 and 51:61 is best of all methods while acc^- and F-measure of BOS is better than BOSFS at imbalanced ratio of 11:61. F-measure and g-means of BOS at imbalanced ratio of 31:61 is also better than BOSFS.

Table 4.9 Page dataset with performance metrics: acc^+, acc^-, g-means and F-measure of SMOTE, Borderline-SMOTE, BOS and BOSFS using k = 5 nearest neighbors.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)							
	2:61	11:61	21:61	31:61	41:61	51:61	61:61	Average
SMOTE								
acc^+	94.02	86.55	89.66	92.44	94.72	95.26	96.36	92.72
acc^-	95.94	93.06	96.29	98.44	98.09	96.61	97.08	96.50
g-means	94.92	89.60	92.87	95.39	96.38	95.93	96.71	94.54
F-measure	38.47	78.28	91.48	95.33	96.57	96.49	97.33	84.85
Borderline-SMOTE								
acc^+	88.58	93.54	93.50	95.67	96.94	96.30	97.31	94.09
acc^-	97.55	91.31	91.31	93.60	93.96	96.22	96.55	93.99
g-means	92.84	92.40	92.38	94.62	95.43	96.26	96.93	93.99
F-measure	67.78	81.73	89.74	94.34	95.80	96.86	97.64	87.71
BOS								
acc^+	97.67	92.55	**94.80**	**97.42**	**96.99**	**97.53**	97.43	**96.34**
acc^-	97.35	**96.05**	94.25	98.36	98.47	98.13	97.81	97.20
g-means	97.48	94.26	94.48	**97.88**	97.72	97.82	97.62	96.75
F-measure	68.42	**90.03**	92.31	**97.85**	97.88	98.13	98.10	91.82
BOSFS								
acc^+	**98.85**	**95.35**	93.37	94.55	96.67	96.96	**97.78**	96.13
acc^-	**97.55**	95.56	**97.72**	**98.62**	**98.98**	**99.09**	**99.08**	**98.16**
g-means	**98.19**	**95.41**	**95.52**	96.56	**97.81**	**98.01**	**98.42**	**97.12**
F-measure	**70.63**	90.02	**94.88**	96.51	**97.91**	**98.16**	**98.62**	**92.38**

In Figure 4.7 and 4.8, it is clearly demonstrated that g-means and F-measure graph of BOSFS are best of all existing methods in each different imbalanced ratio of Page dataset. The best F-measure of all imbalanced ratios is at the imbalanced ratio of 61:61, which means that the ratio of dataset becomes 1:1 or balanced dataset for effective SVM classifiers.

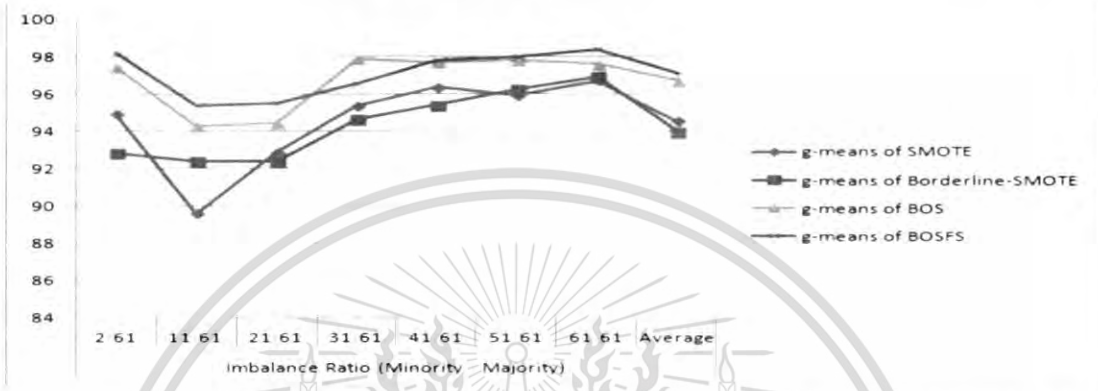


Figure 4.6 the comparison results of g-means graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different seven imbalanced ratios of Page dataset and average of all ratios.

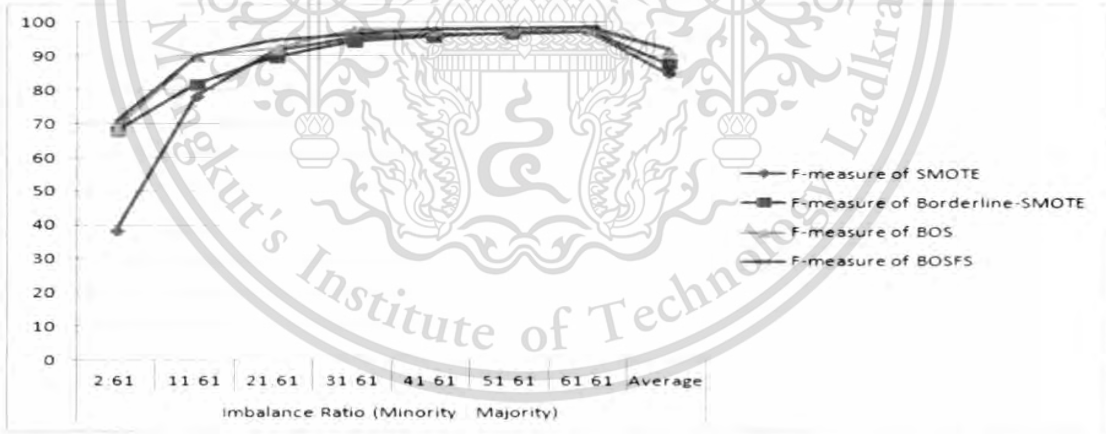


Figure 4.7 the comparison results of F-measure graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different seven imbalanced ratios of Page dataset and average of all ratios.

4.3.3 Experimental Results of BOSFS with Glass dataset

In Table 4.10, we conducted experiments with Glass dataset at the imbalanced ratio of six. The values of bold numbers relate to the best values obtained in each existing method. It is obvious that BOSFS method accomplishes greater numbers of bold values than SMOTE, Borderline-SMOTE and BOS. There are only some better values of SMOTE method. For example, g-means, F-measure of SMOTE at imbalanced ratio of 3:6 and 4:6 are best of all methods while acc⁻ of SMOTE are best of all methods at imbalanced ratio of 4:6 and 5:6. The acc⁺ and g-means of BOS at imbalanced ratio of 6:6 are also better than BOSFS.

Table 4.10 Glass dataset with performance metrics: acc^+, acc^-, g-means and F-measure of SMOTE, Borderline-SMOTE, BOS and BOSFS using k = 5 nearest neighbors.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)					
	2:6	3:6	4:6	5:6	6:6	Average
SMOTE						
acc^+	**100.00**	**100.00**	99.60	92.44	94.72	97.35
acc^-	94.60	**97.28**	**99.33**	**98.44**	98.09	**97.55**
g-means	97.24	**98.62**	**99.46**	95.39	96.38	97.42
F-measure	85.29	**97.28**	**99.19**	95.33	96.57	94.73
Borderline-SMOTE						
acc^+	99.00	96.41	96.83	97.72	96.52	97.30
acc^-	94.41	92.69	88.86	89.09	90.15	91.04
g-means	96.67	94.49	92.72	93.26	93.13	94.05
F-measure	86.35	91.56	89.40	92.60	94.27	90.84
BOS						
acc^+	**100.00**	**100.00**	**100.00**	99.00	96.82	99.16
acc^-	94.00	90.57	90.18	90.93	**98.21**	92.78
g-means	96.92	95.15	94.94	94.78	**97.47**	95.85
F-measure	87.57	89.83	92.54	93.09	97.56	92.12
BOSFS						
acc^+	**100.00**	**100.00**	**100.00**	**99.66**	**100.00**	**99.93**
acc^-	**97.06**	94.46	93.55	95.98	94.93	95.20
g-means	**98.51**	97.17	96.71	**97.79**	97.41	**97.52**
F-measure	**94.80**	93.95	95.23	**97.67**	**97.67**	**95.86**

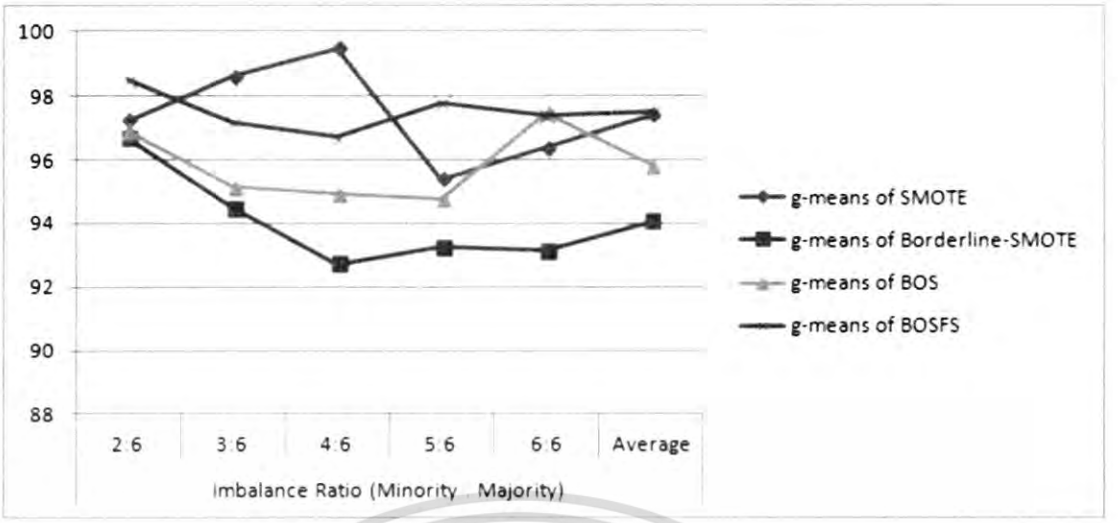


Figure 4.8 the comparison results of g-means graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different five imbalanced ratios of Glass dataset and average of all ratios.

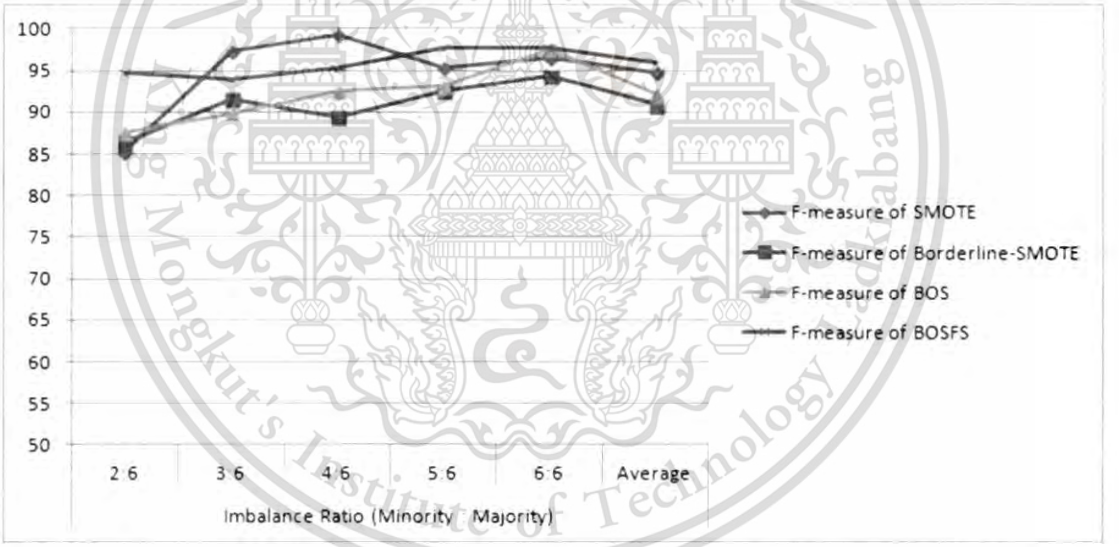


Figure 4.9 the comparison results of F-measure graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different five imbalanced ratios of Glass dataset and average of all ratios.

In Figure 4.8 and 4.9, it is clearly demonstrated that g-means and F-measure graph of BOSFS are better than all existing methods in almost all different imbalanced ratio of Glass dataset. There are some imbalanced ratios at (3:6) and (4:6) that g-means and F-measure of SMOTE are superior than those values of BOSFS. The best F-measure of all imbalanced ratios are closer to the imbalanced ratio of

(5:6) and (6:6) which means that the ratio of dataset becomes closer or equal to 1:1 or balanced dataset for effective SVM classifiers.

4.3.4 Experimental Results of BOSFS with Yeast dataset

In Table 4.11, we conducted experiments with Yeast dataset at the imbalanced ratio of 28. The values of bold numbers are related to the best values obtained in each existing method. It is obvious that BOSFS method accomplishes the greater numbers of bold values than SMOTE, Borderline-SMOTE and BOS. There are only some values of acc^+ of SMOTE and BOS methods that are equal to BOSFS at 100% at the imbalanced ratio of 2:28.

Table 4.11 Yeast dataset with performance metrics: acc^+, acc^-, g-means and F-measure of SMOTE, Borderline-SMOTE, BOS and BOSFS using k = 5 nearest neighbors.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)							
	2:28	6:28	11:28	16:28	21:28	26:28	28:28	Average
SMOTE								
acc^+	**100.00**	92.66	85.61	89.40	92.49	92.79	93.23	92.31
acc^-	93.35	86.58	93.58	92.28	92.78	91.60	92.51	91.81
g-means	96.61	89.37	89.50	90.82	92.61	92.17	92.86	91.99
F-measure	39.86	66.82	88.47	90.48	93.53	93.85	94.43	81.06
Borderline-SMOTE								
acc^+	96.11	94.25	89.16	89.11	91.18	92.65	93.63	92.30
acc^-	92.77	85.04	91.45	95.57	96.71	97.67	97.99	93.89
g-means	94.34	89.34	90.26	92.25	93.89	95.13	95.78	93.00
F-measure	34.56	63.75	88.11	92.33	94.28	95.53	96.20	80.68
BOS								
acc^+	**100.00**	91.86	92.65	93.82	95.45	95.58	96.07	95.06
acc^-	93.71	94.15	96.56	95.83	95.07	95.04	93.93	94.90
g-means	96.80	92.96	94.58	94.80	95.26	95.31	94.99	94.96
F-measure	55.52	86.59	93.65	94.88	95.85	96.13	96.20	88.40
BOSFS								
acc^+	**100.00**	**95.17**	**96.19**	**96.47**	**97.27**	**97.89**	**97.99**	**97.28**
acc^-	**94.96**	**97.18**	**96.71**	**97.13**	**97.57**	**97.39**	**97.79**	**96.96**
g-means	**97.44**	**96.15**	**96.44**	**96.80**	**97.42**	**97.64**	**97.89**	**97.11**
F-measure	**61.00**	**93.70**	**95.74**	**96.88**	**97.67**	**98.13**	**98.35**	**91.64**

In Figure 4.10 and 4.11, it is obviously demonstrated that g-means and F-measure graph of BOSFS are the best of all existing methods in all different imbalanced ratio of Yeast dataset. The best g-means of all imbalanced ratios are at (2:28) and (28:28) but the best F-measure of all imbalanced ratios are the imbalanced ratio of (28:28) which mean that the ratio of dataset become closer or equal to 1:1 or balanced dataset for effective SVM classifiers.

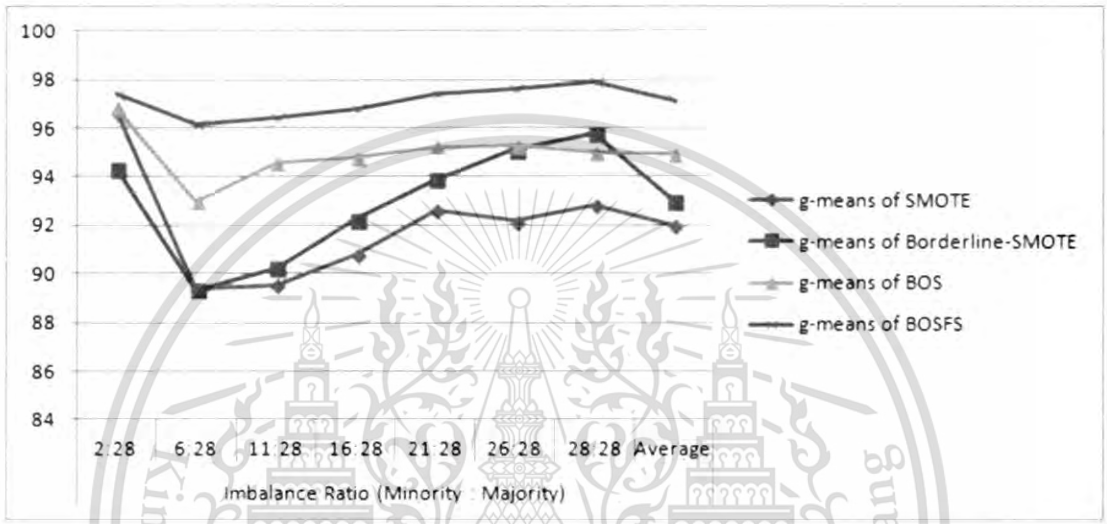


Figure 4.10 the comparison results of g-means graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different seven imbalanced ratios of Yeast dataset and average of all ratios.

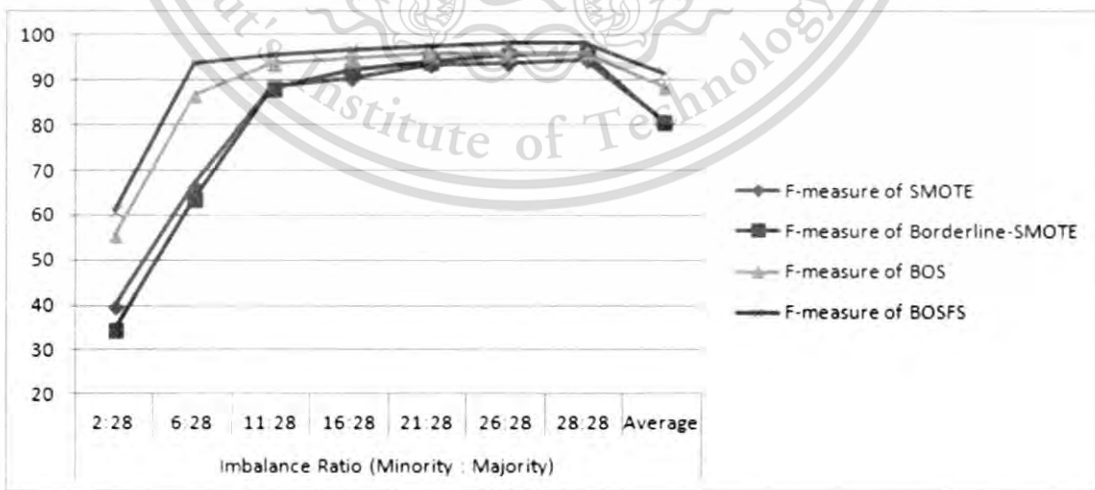


Figure 4.11 the comparison results of F-measure graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different seven imbalanced ratios of Yeast dataset and average of all ratios.

4.3.5 Experimental Results of BOSFS with Spect dataset

In Table 4.12, we conducted experiments with Spect dataset at the imbalanced ratio of four. The values of bold numbers relate to the best values obtained in each existing method. It is obvious that BOSFS method accomplishes the greater numbers of bold values than SMOTE, Borderline-SMOTE and BOS. There are only some values of acc^+ of Borderline-SMOTE and BOS methods that are equal to BOSFS at 100% at the imbalanced ratio of 2:4. Acc^- of BOS at the imbalanced ratio of 4:4 is the best of all methods. From Table 4.8 through Table 4.12, the variation of acc^+ and acc^- values that were obtained by SMOTE, Borderline-SMOTE, BOS and BOSFS methods happened because it depends on the exchange of different C parameter that occurs to be compromised.

Table 4.12 Spect dataset with performance metrics: acc^+, acc^-, g-means and F-measure of SMOTE, Borderline-SMOTE, BOS and BOSFS using k = 5 nearest neighbors.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)			
	2:4	3:4	4:4	Average
SMOTE				
acc^+	99.33	94.83	90.41	94.86
acc^-	87.35	64.91	51.88	68.05
g-means	93.11	77.60	67.09	79.27
F-measure	90.92	76.47	69.97	79.12
Borderline-SMOTE				
acc^+	**100.00**	94.29	93.18	95.82
acc^-	65.61	53.21	49.49	56.10
g-means	80.52	70.28	67.50	72.77
F-measure	60.54	67.37	67.85	65.25
BOS				
acc^+	**100.00**	98.61	96.18	98.26
acc^-	87.16	82.75	**84.04**	84.65
g-means	93.28	90.18	89.62	91.03
F-measure	90.35	93.01	92.08	91.81
BOSFS				
acc^+	**100.00**	**100.00**	**98.66**	**99.55**
acc^-	**89.99**	**86.77**	83.04	**86.60**
g-means	**94.79**	**93.07**	**90.40**	**92.75**
F-measure	**93.30**	**94.19**	**93.74**	**93.74**

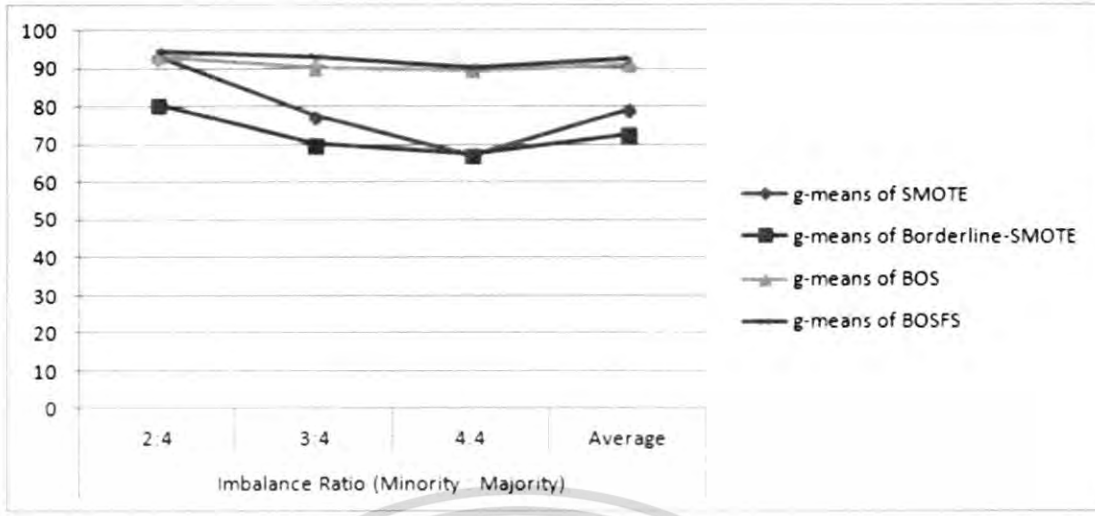


Figure 4.12 the comparison results of g-means graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different three imbalanced ratios of Spect dataset and average of all ratios.

Figure 4.13 the comparison results of F-measure graphs using SMOTE, Borderline-SMOTE, BOS and BOSFS methods with different three imbalanced ratios of Spect dataset and average of all ratios.

In summary, in almost all experiments with five UCI datasets from Table 4.8 through Table 4.12, our new proposed BOSFS method accomplishes the best performance of g-means and F-measure of all existing methods. Graphs of g-means and F-measure from Figure 4.4 through Figure 4.13 illustrate that our new proposed BOSFS method outperformed the existing methods, SMOTE, Borderline-SMOTE and BOS. We could apply BOSFS to UCI dataset, of which the imbalanced class dataset is not limited to SZO dataset. BOSFS can improve SVM classifiers to overcome the imbalanced class data environments with better performance.

Chapter 5
Conclusions

5.1 Conclusions

In this dissertation, we proposed state change detection model of online game players with higher accuracy than existing models for online game companies to solve the problem of giving special promotions to the main online game players who might quit the game. There are two new methods, which are performed in order to develop the proposed model, firstly, we proposed a new feature using the latest k-day playing information of the game players called "SLK days (Staytime of Last K days)". Many experiments have been conducted. The results have shown that the latest-2 day information gives the best prediction performance of SVM classifiers. Secondly, we also proposed a new algorithm called "BOSFS (Borderline Over-sampling in Feature Space)" for SVM classifiers dealing with imbalanced class data environments. F-measure and g-means are used as the performance metrics. The experimental results have shown that the proposed model outperforms the existing models with the 94.35% and 94.27% of g-means and F-measure, respectively.

Moreover, we conducted more experiments on applying BOSFS algorithm to five UCI datasets. The experimental results illustrated that BOSFS is the best method of all existing algorithms namely SMOTE, Borderline-SMOTE and BOS.

5.2 Suggestions

There are three possibilities to develop enhancement of features and algorithms presented in this dissertation.

1) The new feature SLK days could be applied to other domains such as pay cable television, phone and telecommunication, which are based on monthly subscription fees but factors adjusting might be required.
2) The proposed model conducted several experiments based on mapping kernel function from points in the input space to the feature space using linear kernel, polynomial kernel and radial basis function (RBF kernel). The results of RBF kernel function appeared to be the best results of these three kernel function. Further experiments should be performed with sigmoid kernel function to gain better improvement of prediction performance.

3) The proposed model uses the nearest neighbors on the borderline in the feature space. Further experiments could use a variety of distances not specific only nearest distance in order to improve prediction performance of SVM classifiers.

References

[1] Tarng, Pin-Yun, Kuan-Ta Chen, and Polly Huang. "On prophesying online gamer departure.", 2009 8th Annual Workshop on Network and Systems Support for Games (NetGames). IEEE, 2009, doi:10.1109/NETGAMES.2009, 5446225.

[2] R. Thawonmas, K. Yoshida, J. Lou,K. Chen, "Analysis of revisitations in online games", Entertainment Computing 2.4 (2011):pp 2915-221.

[3] V. N. Vapnik, "The nature of statistical learning theory," 1995.

[4] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[5] "Support Vector Machines", Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 31 May 2016

[6] Aizerman, Mark A.; Braverman, Emmanuel M.; and Rozonoer, Lev I. (1964). "Theoretical foundations of the potential function method in pattern recognition learning". Automation and Remote Control 25: 821–837.

[7] Boser, B. E.; Guyon, I. M.; Vapnik, V. N. (1992)., "A training algorithm for optimal margin classifiers"., Proceedings of the fifth annual workshop on Computational learning theory – COLT '92. p.144. doi:10.1145/130385.130401. ISBN 089791497X.

[8] Ben-Hur, Asa, and Jason Weston., "A user's guide to support vector machines." Data mining techniques for the life sciences (2010): 223-239.

[9] Akbani, Rehan, Stephen Kwek, and Nathalie Japkowicz. "Applying support vector machines to imbalanced datasets." Machine learning: ECML 2004. Springer Berlin Heidelberg, 2004. 39-50.

[10] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. "A practical guide to support vector classification." (2003): 1-16.

[11] S. Keerthi and C.J. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation, 15(7):1667-1689, 2003.

[12] Lin, Hsuan-Tien, and Chih-Jen Lin. "A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods." submitted to Neural Computation (2003): 1-32.

[13] Fawcett, Tom. "An introduction to ROC analysis." Pattern recognition letters 27.8 (2006): 861-874.

[14] Barandela, Ricardo, et al. "Learning from Imbalanced sets through resampling and weighting." Pattern Recognition and Image Analysis. Springer Berlin Heidelberg, 2003. 80-88.

[15] M. Kubat and R. Holte, "S. matwin, learning when negative example abound," in Proceedings of the 9th European Conference on Machine Learning, ECML, vol. 97, 1997.

[16] M. Kubat, S. Matwin, et al., "Addressing the curse of imbalanced training sets: one-sided selection," in ICML, vol. 97, pp. 179–186, Nashville, USA, 1997.

[17] G. Wu and E. Y. Chang, "Class-boundary alignment for imbalanced dataset learning," in ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC, pp. 49–56, 2003.

[18] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," Journal of Machine Learning Technologies, vol. 2, pp. 37–63, 2011.

[19] A. M. Smith , C. Lewis , K. Hullett , G. Smith and A. Sullivan, "An Inclusive Taxonomy of Player Modelling", Technical Report UCSC-SOE-11-13 University of California Santa Cruz, CA.,2011, pp.1-18.

[20] B.G. Weber, M. Meteas , "A data mining approach to strategy prediction", In proceedings of the 5th international conference on computational intelligence and games, 2009, pp. 140-147.

[21] M. Shama, M. Mehta, S. Ontanon and A. Ram, " Player modeling evaluation for interactive fiction", In Third Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE), Workshop on Optimizing Player Satisfaction, 2007, pp. 19-24.

[22] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," Machine learning, vol. 30, no. 2-3, pp. 195–215, 1998.

[23] P. K. Chan and S. J. Stolfo, "Toward scalable learning with nonuniform class and cost distributions: A case study in credit card fraud detection.," in KDD, vol. 1998, pp. 164–168, 1998.

[24] T. Fawcett and F. J. Provost, "Combining data mining and machine learning for effective user profiling.," in KDD, pp. 8–13, 1996.

[25] T. FaK. S. Woods, C. C. Doss, K. W. Bowyer, J. L. Solka, C. E. Priebe, and W. P. Kegelmeyer JR, "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," International Journal of Pattern Recognition and Artificial Intelligence, vol. 7, no. 06, pp. 1417–1436, 1993.

[26] R. Batuwita and V. Palade, "Class imbalance learning methods for support vector machines," Imbalanced learning: Foundations, algorithms, and applications, pp. 83–99, 2013.

[27] R. Batuwita and V. Palade, "Efficient resampling methods for training support vector machines with imbalanced datasets," in Neural Networks (IJCNN), The 2010 International Joint Conference on, pp. 1–8, IEEE, 2010.

[28] H. He, E. Garcia, et al., "Learning from imbalanced data," Knowledge and Data Engineering, IEEE Transactions on, vol. 21, no. 9, pp. 1263–1284, 2009.

[29] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 39, no. 2, pp. 539–550, 2009.

[30] I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in Proceedings of Workshop on Learning from Imbalanced Datasets, 2003.

[31] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," Journal of artificial intelligence research, pp. 321–357, 2002.

[32] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new oversampling method in imbalanced data sets learning," in Advances in intelligent computing, pp. 878–887, Springer, 2005.

[33] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline oversampling for imbalanced data classification," in Proc. IEEE SMC Hiroshima Chapter Fifth International Workshop on Computational Intelligence and Applications, (Hiroshima University, Japan), pp. 24–29, 2009.

[34] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline oversampling for imbalanced data classification," International Journal of Knowledge Engineering and Soft Data Paradigms, vol. 3, no. 1, pp. 4–21, 2011.

[35] Guerra-Vazquez, F., and J.J. Rückmann. "A note on Logarithmic Smoothing in Semi-infinite Optimization under Reduction Approach," Croatian Operational Research Review 4.1 (2013): 19-30.

[36] K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum, R. Thawonmas, K.T. Chen, "Departure Prediction of Online Game Players", the Journal of Advanced Materials Research Volume 931-932 (2014) pp.1370-1374, Trans Tech Publication(TTP).

[37] K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum, R. Thawonmas, "Borderline Over-sampling in Feature Space for Learning Algorithms in Imbalanced Data Environments", IAENG International Journal of Computer Science, Volume 43, No. 3, 2016.

[38] M. Lichman, "UCI machine learning repository," 2013.

[39] Vert, Jean-Philippe, Koji Tsuda, and Bernhard Schölkopf. "A primer on kernel methods." Kernel Methods in Computational Biology (2004): 35-70.

[40] H. Zhang, "Distance-based classifier via the kernel trick," in International Journal Software Informatics, Vol.2, pp. 121–133, 2010.

[41] H.-Y. Wang, "Combination approach of smote and biased-svm for imbalanced datasets," in Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on

Computational Intelligence). IEEE International Joint Conference on, pp. 228–231, IEEE, 2008.

Appendices

Appendix A

Publications

1. K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum,"Online Game Player's Behavior by using Hidden Markov Model", ICEAST 2013, International Conference on Engineering, Applied Sciences, and Technology 2013., PID:0144,August 21-24,2013, pp.103.
2. K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum, R. Thawonmas, K.T. Chen, "Departure Prediction of Online Game Players", the Journal of Advanced Materials Research Volume 931-932 (2014) pp.1370-1374, Trans Tech Publication(TTP).
3. K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum, R. Thawonmas, K.T. Chen, "Departure Prediction of Online Game Players", the proceedings 5th KKU Engineering Conference (KKU-IENC 2014).
4. K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum, R. Thawonmas, "Borderline Over-sampling in Feature Space for Learning Algorithms in Imbalanced Data Environments", IAENG International Journal of Computer Science, Volume 43, No. 3, 2016.

Online Game Player's Behavior by using Hidden Markov Model

Kittipat Savetratanakaree[a], Kingkarn Sookhanaphibarn[b], Sarun Intakosum[a]

[a] Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology, Ladkrabang, Thailand.
[b] Department of Computer Science and Software Engineering, School of Science and Technology, Bangkok University, Thailand.

Abstract*—Most game operators revenues come from the subscription fees and sale of virtual items. Game operators need to understand the player's behavior and prediction of player's departure. This paper presents Hidden Markov Models (HMM) approach for pattern recognition of online game player's behavior in game revisitation, classified into player's behavioral state. HMMs have been used for pattern recognition and classification problems since HMMs are proven suitable for modeling dynamic systems. Online Games are dynamic systems which games can be changed according to players can play games with selected different characteristics or roles in game (role-playing game). A large-scale analysis to find player behavior in game revisitations is conducted in our research by using Shen Zhou Online access log. These logs were collected for nearly 4 years consisting of 50,000 player accounts. We use the information on game revisitations, together with daily playing time such the login time, staying time and login frequency information. In this paper, we focus on the predictable records on the past data set of player's behavior in revisitation to the game using HMMs into behavioral states.

Keywords: HMM, Revisitation, Player Behavior, Online game, Shen Zhou Online

I. INTRODUCTION

Massively Multiplayer Online Role-Playing Game (MMORPG) has become increasingly popular in recent years. Shen Zhou Online (SZO) in Taiwan is categorized as MMORPG. SZO is developed and distributed by UserJoy Technology Co.Ltd., SZO maintains at any moment thousands of online players, who must purchase "game points" if they wish to continue their game adventures in the virtual world beyond the 30-day free trial period. Game industry's revenues mostly come from the subscription fee and virtual items sale. The game operator prefer to have more loyal "hardcore players" who would stay in a game more than a year rather than players temporarily or permanently absent from the game after the end of 30-day free trial period.

The Online game player's behavior model [1] is needed for prediction when player's departure from the game. The game operator might give any special promotions to those players who are expected to be absent from the game in order to continue playing the online game which increases the player retention rate of the game. This analysis is important to further study in prediction when the player will absent from the game.

This paper objectives are 1) Understanding the player's behavior in observed player's information on game revisitation 2) Developing a set of states presenting online game player's behavior by using Hidden Markov Model

According to the previous studies [1, 2, 3, 4, 5], game revisitation is the situation that game players returns to play the game again after they might stop playing game for some periods of time. They might be busy or they might do some other things more interested during the stopped periods. It is interesting to know that how long they will return to play the game or they might quit the game. There are typical patterns exist in online-game players' revisiting to a game of interest and its areas.

The Hidden Markov Models(HMM) were first introduced in 1970 as a tool in speech recognition which has become increasingly popular in the recent several years because of its strong statistical foundation for use in a wide range of application as in pattern recognition such as speech signal recognition [7,8,9], handwriting recognition [10], gesture recognition, stock market forcasting [11]. A HMM can be considered the dynamic Bayesian network which provides a probabilities framework for modeling a time series of multivariate observations. We applied HMM with online game player's information on game revisitation because players can change their role-playing as different characteristics in each login which HMM has ability to handle new data robustly and efficient to develop observed sequences to hidden states we are considering.

In this paper, we explain our definition of HMM in Section II. We focus on predictable dataset of online game players records. We filtered unpreferable records out of the predictable dataset and use HMM based model with observed sequences to develop a set of states of game player's behaviors in revisitation to the game displayed as a finite state machine diagram in Section III. We will extend our approach for further study to predict the player's departure from the game which will be our future research.

II. HIDDEN MARKOV MODEL

A Hidden Markov Model (HMM) [6] is a Markov Model in Statistics that the system being observed or modeled assumed as a Markov process. There are observed sequences that can be evaluated as hidden states. These states

represented in a finite state machine that has some fixed number of states. In HMM, the state is not directly visible (hidden), but output, dependent on the state, is visible. Each state has a probability distribution over the possible output sequences.

Regarding HMM, the following notations will be used:
N = number of states in the model
K = number of distinct observation symbols per state
T = length of observation sequence
Y = observation sequence, $Y_1, Y_2, Y_3, \ldots Y_t$
S = state sequence $s_1, s_2, s_3, \ldots s_t$ in the Markov model
$A = \{a_{ij}\}$ transition matrix, where a_{ij} represents the transition probability from state i to state j.

$B = \{b_j(Y_t)\}$ observation emission matrix, where $b_j(Y_t)$ represent the probability of observing Y_t at state j
$\alpha = \{\alpha_j\}$ the prior probability, where α_j represent the probability of being in state$c$ at the beginning of the experiment, i.e., at time $t = 1$
$\psi = (A, B, \alpha)$ the overall HMM model

As denoted above the HMM is characterized by N, K, A, B and ψ. The $a_{ij}, b_i(Y_t)$, and α_j have the properties

$$\sum_j a_{ij} = 1,$$

$$\sum_i b_j(Y_t) = 1,$$

$$\sum_i \alpha_i = 1,$$

and $a_{ij}, b_i(Y_t), \alpha_i \geq 0$ for all i, j, t.

To work with HMM, the following three fundamental problems should be resolved.

1. Given the model $\psi = (A, B, \alpha)$, we will compute $P(Y|\alpha)$, the probability of occurrence of the observation sequence $Y = Y_1, Y_2, Y_3, \ldots Y_t$

2. Given the observation sequence Y and a model ψ, we will choose a state sequence $s_1, s_2, s_3, \ldots s_t$ that best explains the observations.

3. Given the observation sequence Y and a space of models found by varying the model parameters, B and α. The appropriate model will be selected to best explain the observation data.

$$\text{Transition Probability} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$\text{Emission Probability} = \begin{pmatrix} b_1 Y_1 & b_1 Y_2 & b_1 Y_3 \\ b_2 Y_1 & b_2 Y_2 & b_2 Y_3 \\ b_3 Y_1 & b_3 Y_2 & b_3 Y_3 \end{pmatrix}$$

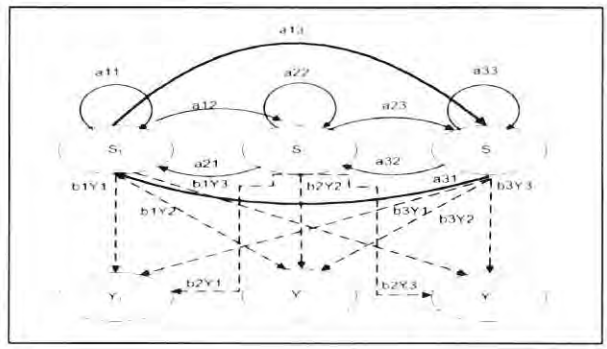


Figure 1. Our proposed model for Online Player's behavior using HMM (Probabilistic parameters $s_i, a_{ij}, b_i(Y_t)$ as explained in the paper)

III. Proposed Methodology

A. Data description

As courtesy of UserJoy, we were able to obtain the traces of 355,706 Shen Zhou Online accounts from January 1st ,2004 to December 3rd ,2007. A total of 119,082,865 sessions is logged. Not all of the sessions are suitable for our analysis. To get suitable data set for our analysis, we need to filter these 5 kinds of records:

1) Hardcore players' records. Since one of our goals is to predict when the player will absent from the game so we need to filter all hardcore players' records which stay more than 90% throughout the year and stay more than one year out of our data set. These hardcore players will never be absent from the game.

2) Error records, which player logout time are less than player login time.

3) Duplicate records, which field values of player id, login time, stay time, logout time are identical. In case of Record Type 2 and 3 were happened because the host server might report error field values due to highest network connections rate at the time which players login. There were a lot of login connections concurrently to the host server.

4) Short-term records, which are records of some players who login to the game less than 5% of average players records or number of records which login less than a month.

5) Unpredictable data set separation from overall data set. Unpredictable data mean all player's records of daily play time in which after plotting graph, the result curve are not in normal curve as shown figure 1 but may be shown in figure 2 instead.

These five kinds of records which are not suitable for our analysis are filtered out from the data set.

B. User of Account of Interest

As in Table I, we figure that some players did not play online game regularly or every day. So we consider moving average player's daily stay time into 10-days intervals with 5 days overlap in each intervals. We calculate logarithmic to daily stay time and plot these data as shown in figure 2 which is sample data of predictable dataset that are in the account of interest. We filter unpredictable dataset out as in figure 3.

TABLE I EXAMPLE OF SIMPLIFIED SZO ACCESS LOG

Record no.	Player ID	Login Time	Logout Time
205	32	24/05/2004 16:27	24/05/2004 19:06
206	32	24/05/2004 19:09	25/05/2004 07:07
207	32	25/05/2004 12:08	26/05/2004 07:00
208	32	26/05/2004 19:37	26/05/2004 22:32
209	32	26/05/2004 22:46	27/05/2004 07:03
210	32	27/05/2004 18:48	27/05/2004 19:19
211	32	30/05/2004 6:06	30/05/2004 07:09

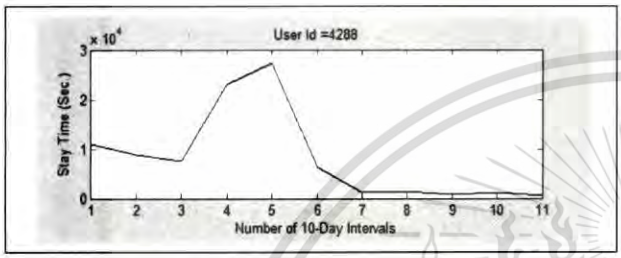


Figure 2 Sample data having normal distribution curve

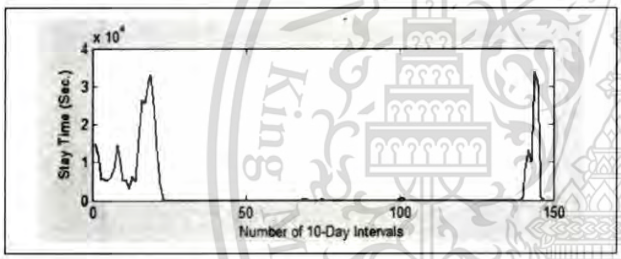


Figure 3 Sample data having unpreferable curve of unpredictable data set which will be filtered all out from the predictable data set collected in the year 2004-2007

C. HMM BASED MODEL

We define online game player's behavioral state into 4 states which are S_1, S_2, S_3, S_4 as

1) Start State (S_1) is the state that player begins to play game increasingly

2) Beginning State (S_2) is the state that player begins to play game increasingly.

3) Interesting State (S_3) is the state that player regularly plays game with the most stay time.

4) Boring State (S_4) is the state that player plays game with less stay time and rarely login to play game.

We consider a stay time of logout time of the of i^{th} logout subtract from the time of i^{th} login of each player to create relationship between each states of the player as shown in TABLE I. example of SZO Access Log of one player.

X_i is denoted as 10-days intervals no. i of each player.

Y_i is denoted as Moving average stay time in 10-days interval no. i of each player .

Z_t is denoted as slope of any stay time a_t and a_{t-1} of any 10-days interval will be Observed Parameters (Y_t) such as Y_1, Y_2, Y_3 when , $t = 1, 2, 3$ where $j = i + 1$ and slope (Z_t) can be calculated as

$$(Z_t) = a_t - a_{t-1}$$

The relationship between player's behavioral states and the observed slope as Table II

TABLE II THE RELATIONSHIP BETWEEN THE OBSERVED SLOPE VALUES AND PLAYER'S BEHAVIORAL STATE

	Slope Z_t between any two points between a_t and a_{t-1}			
OBSERVED SLOPE				
meaning	Start	Increasing trend	High level trend	Decreasing Trend
Observed Value	$Y_t = 0$	$Z_t > 2.469$, $y_t \neq 0$	$-0.22 \leq Z_t \leq 2.469$, $y_t \neq 0$	$Z_t < -0.22$, $y_t \neq 0$
Seq_t	1	2	3	4
Hidden Player's State	Start	Beginning State	Interesting State	Boring State
S_i	1	2	3	4

$$Seq_t = \begin{cases} 1 \text{ where } y_t = 0 \\ 2 \text{ where } Z_t > 2.469 \text{ and } y_t \neq 0 \\ 3 \text{ where } -0.22 \leq Z_t \leq 2.469 \text{ and } y_t \neq 0 \\ 4 \text{ where } Z_t < -0.22 \text{ and } y_t \neq 0 \end{cases}$$

IV. EXPERIMENTAL AND RESULTS

We trained an HMM using the daily player's stay time data for the period from January 1st, 2004 to December 3rd, 2007 to observe and create a finite state diagram of player's behavior as shown in figure 4. We write a Matlab script to estimate the probability of observing sequence and probability of hidden state which represent player's behavioral state. After we put the final Transition Probability and Emission Probability in figure 1 by omitting the line with zero probabilty, the result finite state of player's behavioral state as in figure 4.

In figure 5, we found that some players start playing game with high stay time and gradually decrease stay time that explain why in figure 4, there is a start state directly going to boring state.

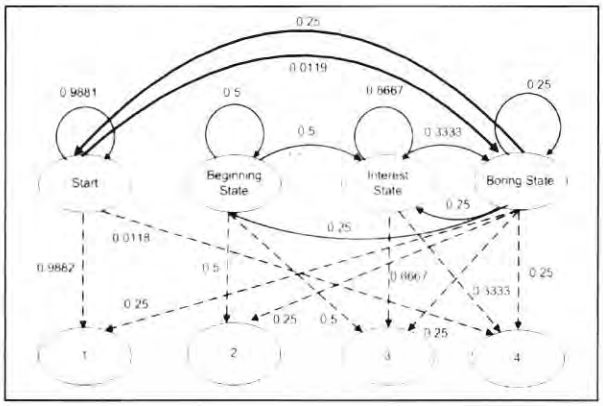


Figure 4. A finite state diagram represents the online game player's behavior using Hidden Markov Model

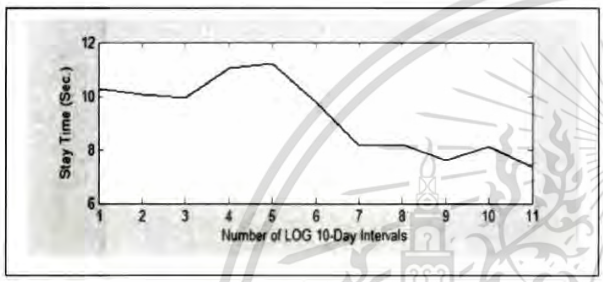


Figure 5. an example graph of player id 4288

V. CONCLUSION AND DISCUSSION

In this paper, we proposed the use of HMM, a new approach , to create a finite state diagram as in Figure 4 represent the online game player's behavioral state. There is significant in changing of player's behavior in revisitation to the game which can define as player's behavioral states, such as Start state, Beginning state, Interesting state, Boring state. It is clear that Table 2 the observed values Z_t which are slope of player's daily usage time, can represent the hidden player's behavioral states and our approach with HMM model, the probability in each state can be shown in the Figure 4.

The finite state diagram explains the result of observed data which most players keep playing online game in the interesting state and laterally they are in boring state and then they finally might quit from the game. There is a chance that players in boring state might go back in interesting state or beginning state.

In our future work, we will further our study in using this new approach with finite state diagram to predict the player's departure from the game. We will extend our study for predictable data set other than normal distribution curve and more player's behavioral states.

REFERENCES

[1] g. K.T. Chen, and P. Huang, "On Prophesying Online Gamer Departure", *Network and Systems Support for Games (NetGames), 2009 8th Annual Workshop on*, vol., no., pp.1,2, 23-24 Nov.2009 doi: 10.1109/NETGAMES.2009.5446235

[2] a ntations in online games", *Entertainment Computing* 2.4 (2011): 215-221.

[3] H. Obendorf, H. Weinreich, E. Herder, M. Mayer, "Web page revisitation revisited : implications of a long-term Click-stream Study of Browser Usage," in Proc. Of the 25th Annual SIGCHI Conference on Human Factors in Computing Systems (CHI'07), 2007, pp. 597-606.

[4] E. Adar, J. Teevan, S Dumais, "Large scale analysis of web revisitation patterns", in Proc. of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems (CHI'08), 2008, pp. 1197-1206.

[5] E. Adar, J. Teevan, S. Dumais, "Resonance on the web: web dynamics and revisitation patterns" , in Proc. of the 27th Annual SIGCHI Conference on Human Factors in Computing Systems (CHI'09), 2009, pp. 1381-1390.

[6] O. Cappe, E. Moulines,T. Ryden,Inference in Hidden Markov Models, ISBN 978-0-387-40264-2. Springer Series in Statistics 2005

[7] Huang X, Ariki Y, Jack M (1990). Hidden Markov Models for speech recognition. Edinburgh University Press.

[8] Jelinek F, Kaufmann M, Mateo C S (1990). Self organized language modelling for speech recognition. in Readings in Speech Recognition (Eds. Alex Waibel and Kai-Fu Lee), Morgan Kaufmann, San Mateo, California,pp. 450-506

[9] Xie H, Anreae P, Zhang M, Warren P (2004). Learning Models for English Speech Recognition, Proceedings of the 27th Conference on Australasian Computer Science, pp. 323-329.

[10] Vinciarelli A and Luettin J(2000). Off-line cursive script recognition based on continuous densit HMM, Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition, Amsterdam, pp. 493-198.

[11] Md. Rafiul H and Baikunth N, Stock Market Forecasting Using Hidden Markov Model : A New Approach, Proceedings of the 2005 5th International Conference on Intelligent Systems Design and Applications (ISDA'05)

Advanced Materials Research Vols. 931-932 (2014) pp 1370-1374
© (2014) Trans Tech Publications, Switzerland
doi:10.4028 www.scientific.net AMR.931-932.1370

Departure Prediction of Online Game Players

Kittipat Savetratanakaree[1,a*], Kingkarn Sookhanaphibarn[2,b], Sarun Intakosum[1,c], Ruck Thawonmas[3,d], Kuan-Ta Chen[4,e]

[1] Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology, Ladkrabang, Thailand.

[2]School of Science and Technology, Bangkok University, Thailand.

[3]Department of Human and Computer Intelligence, Ritsumeikan University, Kusatsu, Shiga, Japan

[4] Institute of Information Science, Academia Sinica, Nankang District, Taipei, Taiwan

[a] kittipatsavet@gmail.com, [b] kingkarns@gmail.com, [c] intakosumsarun@gmail.com, [d] ruck@ci.ritsumei.ac.jp, [e] ktchen@iis.sinica.edu.tw

Keywords: data mining, user behaviors, game revisitations, player behavior, massively multiplayer online role-playing game(MMORPG), Shen Zhou online, SLKdays

Abstract. Most business models of online game companies usually depend on sale of virtual items and the monthly subscription fees. The prediction of player departure could increase revenues by giving special promotions out to the players who are expected to unsubscribe or quit playing the game. This paper proposes a departure prediction approach by using a new feature called "*SLKdays*" and a *game revisitation*. The feature "*SLKdays*" is defined as "*Staytime*", which is the time each player spending in an online game, of the last *k* days, and a *game revisitation* is the playing frequency in the last month to predict the next month subscription. We explore our new feature "*SLKdays*" to determine the optimal number of *k* for the departure prediction. With our proposed feature, the accuracy of the departure prediction is high, 91.92%, and the precision and recall rate are 98.22% and 84.51%.

Introduction

Analyses in games and players [1, 2, 3, 4, 5, 6, 7, 8] are being enthusiastically conducted by a game research community to improve the game design and performance, and the business strategy. This paper is the extending study in the analysis of revisitations in online games [2]. Thawonmas et al. [2] analyzed game revisitations focusing on how long online-game players revisiting the game after their last logout. In this paper, we hypothesize that online-game player's information of last month (last 30-days interval) can predict that a player will continue or absent from the game in the next month (next 30-days interval).

In order to validate our hypothesis, we conduct four experiments with our new feature *SLKdays*, where *SLKdays* is defined as the stay time of the last *k* days of the last month, to predict online-game players' behaviors in the next 30-days interval. Four experiments are designed for a real-world use, i.e., the training set will use the information of the previous year as called one year-ahead prediction. We also tested our approach for the two-years ahead prediction. To build the prediction model, we use *SLKdays* combined with the game revisitation as the extracted feature. Our prediction approach will be explained in the rest of this paper.

Background and Related Works

Online game player's information has been shown effectively in providing useful information in a variety of research domains. There seems to be an increased interest in online-game player's information in game-oriented AI research, adaptive game research, player experience modeling, and game user research. We review various analyses related to online game player's information as follows.

Table 1: Four experiments with different year (y) for training set and test set.

Experiments	Test set (year)	Training set	
		$y-2$	$y-1$
1.1	2005	-	2004
1.2	2006	-	2005
2	2006	2004	-
3	2006	2004	2005

Based on a set of EverQuest II, Shim et al. [4] used the game's player activity data in game to construct profiles of online game players' behaviors for the pattern recognition of normal and abnormal behaviors. Analyzing the history of large behavioral data in a game provide valuable insight into a range of character types (i.e. archetype, classes, sub-classes, race) which exist in the game. The authors examined the player efficiency in terms of total experience point (XP point) and find that there is an overall trend with respect to how fast particular sub-class advance throughout the game and changes in task types as players advance throughout the game.

Whang et al. [5] explored different players' behaviors from players' lifestyles, focusing on how a game player adopted the virtual game world as part of their life. The lifestyles in the online game, Lineage, were classified into three categories as off-real world gamer, community-oriented player, single-oriented player. The study model is to understand how players with different real life backgrounds will play to the various features of a game and how they can adopt their new social identities in the virtual game world.

Chen et al.[6] focused on how network quality and network loss affect a player's decision to unsubscribe an online game prematurely. Trace collection and traffice measurement was conducted by monitoring the traffic of game servers. The results indicate that both network loss and network delay significantly affect a player's decision to quit playing a game too early.

Lou et al. [7] proposed a forecast model for online game addictiveness according to players' emotional responses. The model using additional hardware, elctro-myographic measures (EMG), attached players' two facial muscles which the device can indicate brain signal in order to know humans' positive and negative emotions. The work can ensure that a game's design is on the right track in the early of game development phase.

All papers above focus on different aspects from our paper.

Proposed Methodology

Data description

We obtained the traces of 355,706 Shen Zhou Online (SZO) accounts from January 1st, 2004 to April 1st,2007 as courtesy of User Joy Technology Co.Ltd. SZO is a massively multiplayer online role-playing game (MMORPG) in Taiwan. There are 119,082,865 sessions logged but not all of the sessions are focused. Data pre-processing are required to obtain a suitable data set for our analysis. We filtered the following kinds of records out from the data set.

1. Error records: They are those whose logout time is before the login time.
2. Duplicate records: player id, login-time, stay time, and logout-time are the same. This happens when the player's login time is during the highest network connection rate.
3. Short-term records: They are those whose players logged in less than a month. These players are absent from the game before the end of 30-day free trial period. We focus on players who login and stay in the game for more than a month, or the free trial period of time.

Table 2: Classification of 13 predefined bins of *game revisitations*.

Bin (j_{th}).	1	2	3	4	5	6	7	8	9	10	11	12	13
Time Intervals (m)	32	64	98	136	212	424	848	1696	3392	6784	13568	27136	> 27136
Time Intervals (approx.)	30m	1h	1.5h	2h	3h	6h	12h	1d	2d	4d	8d	16d	≥ 16d

Note that m = minute or minutes, h = hour or hours, d = day or days

4. In-2007 records: The players information of interest is the records of 12 months in each year. Since there are the data log of only 3 months in the year 2007, the players' information in year 2007 is excluded from our consideration.

In this study, we conducted four experiments, each with a different pair of training and test data set as shown in Table 1.

Departure prediction approach

First, we consider the playing frequency on *game revisitations* [2], which it is defined as the number of times each player revisits to play the same game within the predefined time intervals (bins) of N days. We base our prediction method on the daily play time on *game revisitations* within 30-day intervals classified into 13 predefined bins as shown in Table 2. Each 30-day intervals of daily play time is 5-days overlap.

Proposed feature *SLKdays*

Second, we calculate the stay time per day (*Staytime*) for each player as follows :

$$Staytime = Logout\_time - Login\_time \quad . \tag{1}$$

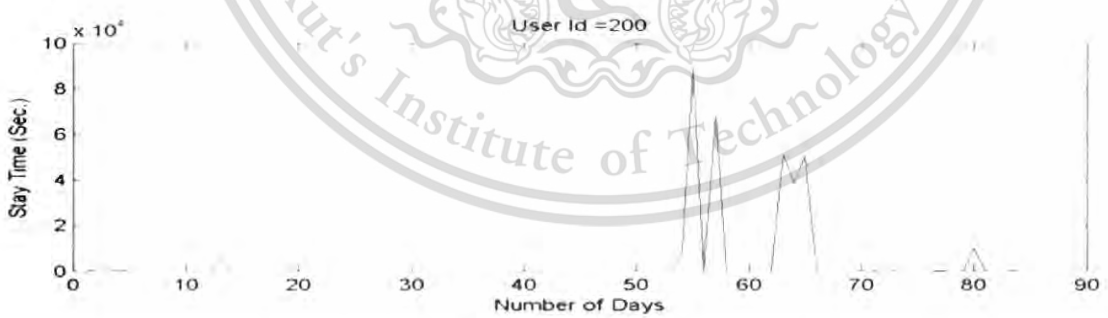


Fig. 1: Daily *Staytime* of user ID 200.

Fig. 1 shows that some players did not play the online game regularly. We smooth the daily play by averaging it in an interval such as every 5 days. From our study, we discovered that the stay time of last k days is significantly related to their continuing or absent in the next month. We also applied the logarithm to transform the linear data. In summary, we calculate the logarithm of average *Staytime* of last k days (*SLKdays*) in each 30-days interval written as follows :

$$SLKdays = \log\left(\frac{\sum_{i=2}^{k} Si}{k}\right) . \tag{2}$$

where S_i is daily *SLKdays* of last day $i = 2, 3, \ldots, k$ days.

Third, the normalization of *game revisitations* in each bin and that of *SLKdays* are required and its equation is shown below.

$$\frac{Bin[j]\_of\_player[i]}{average\_of\_Bin[j]\_among\_all\_users}. \quad (3)$$

where *bin*[*j*] of player [*i*] is the number of times *player*[*i*] revisiting SZO within *bin*[*j*]'s interval. Note that the normalization of *SLKdays* of each player[i] is the last bin ($j = 14$) also using Eq.3.

Lastly, we use the support vector machine (SVM) as the classifier for the departure prediction.

Experimental and Results

In our experiments, we analyzed a variety of k days for finding the best departure prediction. In every experiment, we explored 13 bins of game revisitations as shown in Table 2. We tested totally 10 cases: nine cases of k numbers, where $k = 2, 3, \ldots, 10$, one case excluding *SLKdays*, and the last case, where $k = 0$, i.e., we use only 13 bins of game revisitations without our new feature *SLKdays*. In summary, there are 10 cases analyzed in each of the four experiments as shown in Table 1. In each experiment, we show the comparison of accuracy for *SLKdays* in k days and the last case ($k = 0$).

After we applied SVM in Matlab with the training and test data sets as experiments shown in Table 1, the accuracy percentage of all experiments is shown in Fig. 2. We found that accuracy trend.

Fig. 2: Accuracy trend of *SLKdays* on different k where $k = 0$ indicates the case that uses only the *game revisitation* without our new feature *SLKdays*.

without our new feature are dropped to 47.66% - 64.67%. To compare them with our proposed feature, we included *SLKdays* from $k = 2, 3, \ldots, 10$ days and found that the highest accuracy is *SLKdays* on $k = 2$. These results can imply that the most recent information of *staytime* yields the best departure prediction. We calculate the accuracy, precision and recall of the 10 cases, whose results are shown in Table 3.

The accuracy, recall and precision of *SLKdays* when $k = 2$ is the best of all 10 cases of experiment
1.2, i.e., accuracy (91.92%) , recall (84.51%) and precision (98.22%). From the trend shown in Fig. 2, the decreasing k, the highest percentage of accuracy of the prediction is found.

Conclusion and Future Work

In this paper, we proposed departure prediction with new feature *SLKdays* when $k = 2$ days as in the *Eq.*2 with accuracy 91.92% ,recall 84.51% and precision 98.22%, appears to be the best departure prediction of all other different days studied . The high percentage of recall means that our new feature can predict the right players (up to 84.51%) who are going to quit the game.

Table 3: The accuracy, recall and precision of 10 cases of experiment 1.2.

SLKdays *k days*	Accuracy %	Recall %	Precision %
No new feature	47.66	99.80	47.54
$k = 2$	91.92	84.51	98.22
$k = 3$	89.93	79.43	97.40
$k = 4$	88.43	75.43	96.78
$k = 5$	86.98	72.24	95.39
$k = 6$	84.77	68.01	93.48
$k = 7$	82.96	66.26	90.88
$k = 8$	81.61	65.98	88.40
$k = 9$	80.59	65.70	85.93
$k = 10$	79.50	68.60	81.64

The ability to predict online game players'departure is important to both game operators and game developers in terms of current game improvement and the new design of future games. With accuracy of departure prediction approach, game operators might give special promotions to the right target players who plan to quit the game in order to keep their subscriptions and online game players'retention rate. The percentage of precision with new feature *SLKdays* is up to 98.22%. This result means that promotions given to the predicted players are worth because the predicted players (98.22%) are the right players who plan to unsubscribe the game. In future work, we will further our study using this approach combine with new features of online game players' behavioral states[3] to get better model performance in departure prediction of online game players.

References

[1] P.Y. Tarng, K.T. Chen, and P. Huang, "On Prophesying Online Gamer Departure", Network and Systems Support for Games(NetGames), 2009 8th Annual Workshop on , vol.1, no.2, pp.23-24 Nov.2009 doi:10.1109/NETGAMES.2009. 5446225.

[2] R. Thawonmas,K. Yoshida, J. Lou,K. Chen, "Analysis of revisitations in online games", Entertainment Computing 2.4 (2011):pp 215-221.

[3] K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum,"Online Game Player's Behavior by using Hid- den Markov Model", ICEAST 2013, International Conference on Engineering, Applied Sciences, and Technology 2013., PID:0144,August 21-24,2013, pp.103.

[4] K.J. Shim, J. Srivastava, "Behavioral Profiles of Character Types in EverQuest II",IEEE Conference on Computational Intelligence and Game (CIG'10), 2010, pp.186-194.

[5] L.S. Whang and G.Y. Chang, "Lifestyles of Virtual World Residents, Living in the on-line game, Lineage", Proceedings of the 2003 International Conference on Cyberworlds (CW'03), 2003, pp.18-25.

[6] K.T. Chen, P. Huang, and C.L. Lei, "Effect of Network Quality on Player Departure Behavior in Online Games", IEEE Transactions on Parallel and Distributed Systems, 2009, pp.593-606.

[7] J.K. Lou,K.T.Chen, H.J. Hsu, and C.L. Lei, "Forecasting Online Game Addictiveness", 11th Annual Workshop on Network and Systems Support for Games (NetGames), 2012, pp. 1-6.

[8] Z.H. Borbora and J. Srivastava, "User Behavior Modelling Approach for Churn Prediction in Online Games", 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust. pp.51-60.

Borderline Over-sampling in Feature Space for Learning Algorithms in Imbalanced Data Environments

Kittipat Savetratanakaree, *Member, IAENG*
Kingkarn Sookhanaphibarn, Sarun Intakosum and Ruck Thawonmas

***Abstract*—In this paper, we propose a new approach to over-sample new minority-class instances along the borderline using the Euclidean distance in the feature space to improve support vector machine (SVM) performance in imbalanced data environments. SVM has been an outstandingly successful classifier in a wide variety of applications where balanced class data distribution is assumed. However, SVM is ineffective when coping with imbalanced datasets whereby the majority-class instances far outnumber the minority-class instances. Our new approach, called Borderline Over-sampling in the Feature Space, can deal with imbalanced data to effectively recognize new minority-class instances for better classification with SVM. The results of our class prediction experiments using the proposed approach demonstrate better performance than the existing SMOTE, Borderline-SMOTE and borderline over-sampling methods in terms of the g-mean and F-measure.**

***Index Terms*—Borderline Over-sampling in the Feature Space, Imbalanced Dataset, Over-sampling, SVM in Imbalanced Data Environments**

I. Introduction

SUPPORT vector machine (SVM) is an extremely successful classifier proposed by Vapnik [1] under the presumed condition of balanced data distributions among classes. However, SVM is ineffective when mining data with imbalanced classes. An imbalanced dataset in which the representation between classes is not approximately equal. There are many applications in real-world domains that have innately imbalanced datasets including fraudulent telephone call detection [2], oil spill detection in satellite images [3], telecommunications risk management [4], credit card fraud detection [5], IVF embryos implantation [6], balancing class in clinical dataset[7], text categorization, and unusual disease diagnosis [8].

By mining a large amount of balanced data, SVM classifiers can extract valuable knowledge for decision making support and other objectives. Hidden valuable knowledge sometimes resides in minority-class instances. Minority-class instances are thus often more useful than the majority-class instances and are also called positive instances. Majority-class instances are also called negative instances. On imbalanced datasets, the positive instances are generally misclassified by SVM classifiers because they can be treated as noise.

In some cases, the issue of class imbalance is critical and cannot be ignored. One example is the classification of pixels in mammogram images for possible breast cancer [9]. In this application, the majority-class of normal pixels might contain 98% of the data, whereas the minority-class of abnormal pixels may contain only 2%. If the machine learning algorithm ignores the abnormal pixels, patients' lives could be threatened. Classification algorithms often perform worse in the detection of such unusual cases, which tend to be the most important ones.

There are several methods [10] for overcoming the imbalanced class problem in SVM. The methods are classified into two main groups. The first group comprises external methods: data preprocessing methods that adjust the distribution of class datasets before training SVM classifiers. The second group comprises internal methods: algorithmic modifications to SVM to decrease its sensitivity to imbalanced classes.

In this paper, we propose an over-sampling method called Borderline Over-sampling in the Feature Space (BOSFS) that fits into the first group of data preprocessing methods. BOSFS conducts over-sampling by generating new synthetic minority-class instances with the nearest existing neighbors focused on the borderline in the feature space. These new synthetic instances are combined with the original imbalanced training dataset to form a new training dataset. SVM is trained using the new training dataset, and then assessed on an independent testing dataset. With this new BOSFS method, the SVM classifier achieves higher recognition performance for the minority-class instances in the imbalanced testing dataset.

II. Background and Related Work

In the external methods category, there are two different approaches: resampling and ensemble learning. First, resampling methods [11] consist of random, focused under- or over-sampling methods. These methods balance the minority-class instances and majority-class instances in the datasets before training SVM models. In the under-sampling approach, the random instances of the majority-class are removed until the datasets are balanced. In the over-sampling approach, the minority-class instances are randomly duplicated to achieve an approximately one-to-one ratio with the majority-class instances. Some research [12] [13] [14]

Manuscript received February 12, 2016, revised May 30, 2016.
K. Savetratanakaree and S. Intakosum are with the Computer Science Department, School of Science, King Mongkut's Institute of Technology, Ladkrabang, Bangkok, Thailand.
K. Sookhanaphibarn is with the Computer Science and Software Engineering Department, School of Science and Technology, Bangkok University, Bangkok, Thailand.
R. Thawonmas is with Ritsumeikan University, Shiga, Japan.
Part of this work by the fourth author was supported in part by Grant-in-Aid for Scientific Research (C), Number 26330421, JSPS. Corresponding author is K. Savetratanakaree, Email: kittipatsavet@gmail.com.

has pointed out that information might be lost by using the under-sampling method because important minority-class information might be randomly removed. In contrast, there is no information loss when using the over-sampling method, which may lead to better results. Over-sampling methods include synthetic data generation methods such as SMOTE [15], Borderline-SMOTE [16], and borderline over-sampling (BOS) [17] [18].

Second, ensemble learning methods divide the majority-class dataset into many subdatasets. The number of majority-class instances in each of these subdatasets is equal to the number of minority-class instances. These approaches may use clustering methods or random sampling with or without replacement (bootstrapping). Different SVM classifiers are used for each training dataset, which consists of the same positive dataset combined with a different negative subdataset.

In this paper, we focus on the external (data preprocessing) over-sampling methods. SMOTE [15] is one of the most popular methods that over-samples the minority class by generating new synthetic minority-class instances rather than by over-sampling with replacement. This is done interpolating the k nearest neighbors of randomly-chosen existing minority-class instances. Borderline-SMOTE [16], a variant of SMOTE, performs over-sampling by interpolating the k nearest neighbors of each minority-class instance focused on the borderline. Concentrating on instances in the borderline area has been proven to achieve higher SVM performance. The rest of the minority-class instances in areas other than the borderline are removed from consideration.

BOS [17] [18], yet another variant of SMOTE, focuses on using both interpolation and extrapolation techniques to generate synthetic minority-class instances along the borderline from existing minority-class instances that are minority-class support vectors of SVM and a number of their nearest neighbors. All of these algorithms use the over-sampling method to generate new synthetic minority-class instances, but the BOS Algorithm in [17] [18] outperforms both SMOTE and Borderline-SMOTE.

When SVM is used as the classifier, SMOTE, Borderline-SMOTE, and BOS find nearest neighbors of interest in the input space, not feature space, of SVM. Our proposed BOSFS uses the kernel function as a mapping function of those neighbors or minority-class instances in the input space to the feature space. BOSFS uses specific equations of Euclidean distance with the kernel function to find such nearest neighbors to the borderline directly in the feature space. BOSFS also applies a combination of interpolation and extrapolation techniques with the Euclidean distance in the feature space to create new synthetic minority-class instances or positive instances along the borderline. Thus, we expect that such synthetic positive instances could become new support vectors contributing to better SVM performance.

III. Imbalanced Data Classification with Support Vector Machines

A. Support Vector Machines

The aim of support vector machines [19][20] is to find the optimal boundary that separates the negative and positive instances with the largest margin. Consider the training dataset $\{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_i, y_i)\}$ where $\mathbf{x}_i$ represents the training instance and y_i represents the label of the instance: $y_i \in \{-1, +1\}$. Using the training dataset, SVM creates an optimal boundary. This boundary can be computed by minimizing the objective function as follows:

$$\min_{w,b,\xi} \frac{1}{2}\mathbf{w} \cdot \mathbf{w}^T + C\sum_{i=1}^{N} \xi_i \tag{1}$$

subject to

$$\begin{cases} \forall_i y_i(\mathbf{w}^T \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ \forall_i \xi_i \geq 0 \end{cases} \tag{2}$$

where $\mathbf{w}$ is the weight vector, y_i are the labels, b is the offset or bias of the hyperplane, $\Phi(\cdot)$ is the mapping function from a point in the input space to a corresponding point in the feature space, ξ_i are the slack variables (considering the non-separable case by admitting misclassification of training instances), and C is the user-specified parameter for the penalty on training instances on the wrong side of the boundary. If the C parameter is very small, SVM classifies all instances as negative to maximize the margin. Clearly, this causes more training errors with respect to positive instances. [21] proposed a solution to this problem by using different parameters C, such that C^+ corresponds to the minority-class instances and C^- corresponds to the majority-class instances. The tradeoff C^+ is chosen to be larger than C^- according to the data imbalance ratio.

In Equation (1), minimizing the objective function by minimizing the first and second terms corresponds to maximizing the margin and minimizing the training errors, respectively. The dual representation of Equation 1 is as follows:

$$\max W(\alpha) \equiv \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{3}$$

subject to

$$\begin{cases} \forall_i 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{4}$$

where y_i are the labels, $K(\mathbf{x}_i, \mathbf{x}_j)$ represents a Kernel function (as shown in (7) and (9), also known as the Kernel trick to compute dot products in the feature space without knowing the real Φ mapping), and α_i's are the Lagrange multipliers which are nonzero only for the training instances that fall within the margin. These training instances are called support vectors [19] and they are crucial instances of the training dataset. They define the optimal hyperplane of the decision boundary to separate positive and negative instances.

B. SVM and imbalanced class data

There are two main problems with using SVM dealing to classify imbalanced datasets [22]. First, the SVM classifier is biased towards the majority-class instances and thereby achieves a poor classification rate on minority-class instances. Fig. 1 shows that, in imbalanced data environments, the borderline (solid line) is skewed towards the minority class instances. In other words, the positive instances (white triangles) are further from the ideal boundary. This problem occurs because the number of negative instances is higher than the number of positive instances around

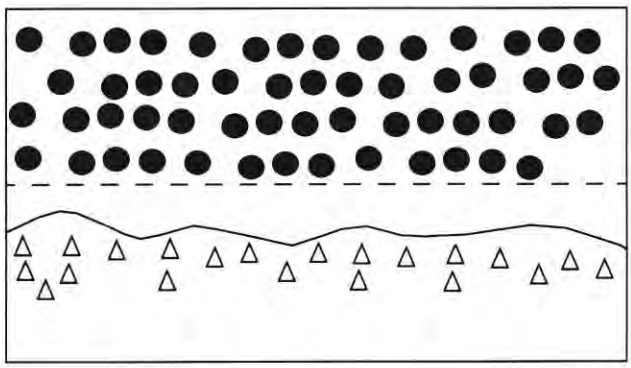


Fig. 1. The borderline (solid line) when training with SVM is skewed towards the minority instances. The dashed line shows an ideal borderline.

the ideal boundary. This may cause the misclassification of positive instances by the SVM such that the prediction results yield all negative instances. In this case, the example of an ideal borderline (dashed line in Fig. 1) is much less skewed and should have more generalization ability in classifying unknown instances.

To overcome this problem in classifying imbalanced datasets using SVM, the main target is to expand the borderline derived by SVM by over-sampling new synthetic minority instances (new positive instances) shifted towards the ideal borderline, as shown in Fig. 1. This has been approached by using an interpolation technique as proposed by [23], and by a combination of interpolation and extrapolation techniques in BOS [17] [18], as further explained in Section IV C.

Second, soft-margins [22] minimize the objective function in (1) by minimizing the first and second terms corresponding to maximizing the margin and minimizing the training errors, respectively. The weakness of soft-margins lies in the fact that the C parameter in the second term ($C\sum_{i=1}^{N}\xi_i$) is a constant chosen by the user. The C parameter specifies the tradeoff that the user is willing to allow between maximizing the margin and minimizing the training errors. If C is too high or low, it may cause over- or under-fitting problem, respectively. We solve this problem by controlling the sensitivity of SVM using different parameters C, as proposed by [21]. The sensitivity of SVM is the ratio between the number of true positive predictions (TP) and the number of positive instances in the test set:

$$sensitivity = \frac{TP}{TP + FN} \tag{5}$$

where FN is the number of false negatives.

The specificity of SVM is the ratio between the number of true negative predictions (TN) and the number of negative instances in the test set:

$$specificity = \frac{TN}{TN + FP} \tag{6}$$

where FP is the number of false positives.

We must control the sensitivity of SVM with different parameters C by using the cross-validation method. We thus determine the optimal C to solve this problem.

C. Main drawbacks of existing methods

Before over-sampling by generating new synthetic positive instances with interpolation or extrapolation techniques, there is the crucial step of finding the nearest neighbors of the positive instances along the borderline to determine the optimal boundary of SVM. Fig. 2 provides an illustration of the difference between our proposed method and the previous methods.

In Fig. 2a, the lighter line is a border line in the input space. The negative instances (black circles) and positive instances (white triangles) are in the original imbalanced training dataset or in the input space. $\mathbf{x}_i, \mathbf{x}_j$, and $\mathbf{x}_k$ are examples of any positive instances (white triangles) in the input space. The arrows depict the distances between each positive instance for each $\mathbf{sv}_i$ (stars) in the input space. $\mathbf{x}_j$ (white triangle with circle) is selected to be the nearest neighbor for $\mathbf{sv}_i$ in the input space. In Fig. 2b the standard SVM creates the borderline by using the original imbalanced training dataset. The borderline (solid line) classifies the negative instances (black circles) and positive instances (white triangles) in the feature space. The $\mathbf{sv}_i$ (star) is any positive instance on the borderline in the feature space. Some positive instances in the input space are individually associated to their corresponding positions in the feature space by double-sided arrows.

In Fig. 2a the Borderline-SMOTE [16] and BOS [17], [18] algorithms find the nearest neighbors for $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$ which are in the input space to each $\mathbf{sv}_i$ (a corresponding point) in the input space. The algorithms consider only the positive instances that are closest to the $\mathbf{sv}_i$ (star) corresponding point in the input space rather than the $\mathbf{sv}_i$ (star) on the borderline in the feature space. The white triangle with circle ($\mathbf{x}_j$) is selected to be the nearest neighbors to $\mathbf{sv}_i$ in the input space. In Fig. 2b The real nearest neighbor to $\mathbf{sv}_i$ in the feature space is ($\mathbf{x}_k$), which is not selected by Borderline-SMOTE or BOS. The SMOTE [15] algorithm also finds the nearest neighbors in the same way as Borderline-SMOTE and BOS, but it considers the existing positive instances of the entire area rather than just in the borderline area.

The main issue in SMOTE, Borderline-SMOTE, and BOS is that the nearest neighbor of $\mathbf{x}_i$ and $\mathbf{sv}_i$ should be found in the feature space rather than the input space. The proposed BOSFS uses the kernel function in Equations (7) and (9), and the Euclidean distance in Equation (8) between any $\mathbf{sv}_i$ and $\mathbf{x}_i$ to find the new nearest neighbors of any $\mathbf{x}_i$ for each $\mathbf{sv}_i$ in the feature space. [24] showed that the Euclidean distance using the kernel function in the feature space can be used effectively with the SVM classifier. These new nearest neighbors are combined with interpolation and extrapolation techniques to generate new synthetic positive instances for the new borderline. Our BOSFS method thus achieves the main target of expanding the new borderline for improved SVM performance when dealing with imbalanced class datasets.

IV. Proposed Method

A. Main concept of the BOSFS algorithm

We propose BOSFS to allow SVM to better deal with imbalanced data environments. It has been shown that focusing on positive instances in the borderline area [16] [25] is important to achieve better SVM performance. There are several kernel functions used in SVM including the linear kernel, polynomial kernel, and radial basis function (RBF)

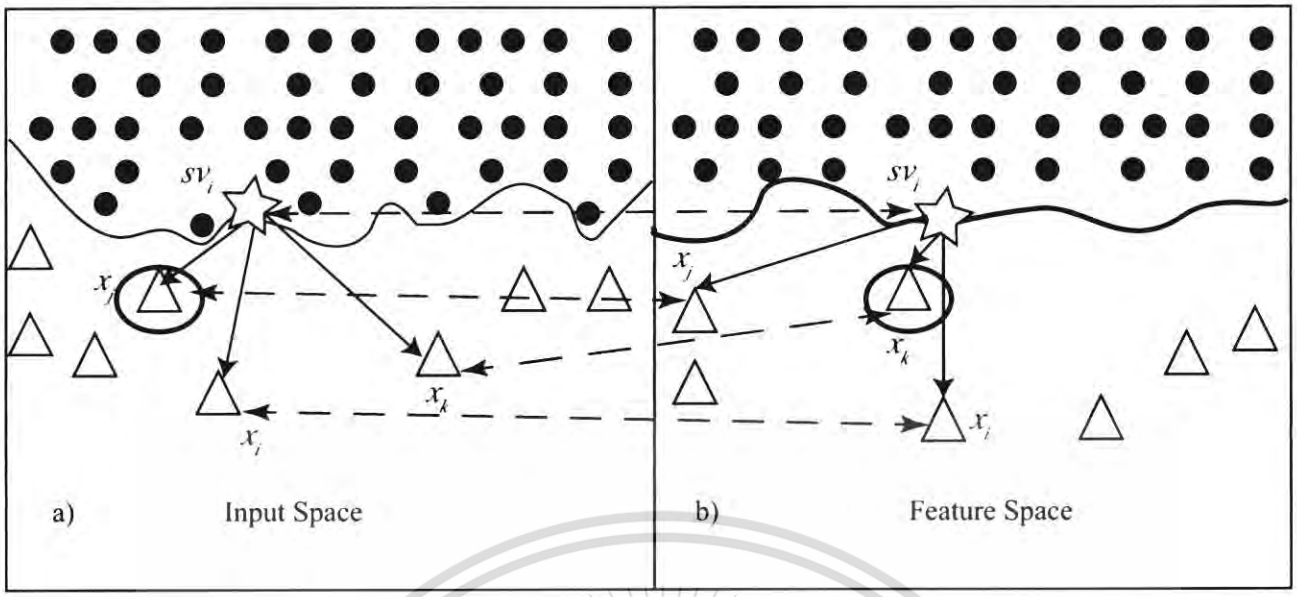


Fig. 2. Illustration of the difference between the previous methods and our proposed BOSFS method. a) The arrows depict the distances between positive instances $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$ to each $\mathbf{sv}_i$ on the borderline in the input space. This figure illustrates that the SMOTE, Borderline-SMOTE, and BOS algorithms find their nearest neighbors to $\mathbf{sv}_i$ (star) in the input space. $\mathbf{x}_j$ (white triangle with circle) is selected to be the nearest neighbor for $\mathbf{sv}_i$ in the input space. b) There is a dashed double-sided arrow between $\mathbf{sv}_i$ (star) in the input space and its corresponding point in the feature space. The arrows depict the distances between corresponding positive instances $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$ to each $\mathbf{sv}_i$ on the borderline in the feature space. The real nearest neighbor $\mathbf{x}_k$ (white triangle with circle) is selected for $\mathbf{sv}_i$ (star) in the feature space. This figure illustrates that the new BOSFS algorithm finds the real nearest neighbors to $\mathbf{sv}_i$ (stars) directly in the feature space. This process is an important step in generating new synthetic positive instances for the new borderline to improve the performance of SVM.

kernel. We conducted several experiments on UCI datasets using different kernel methods. The best result was obtained when using the RBF kernel. The RBF kernel may be more suitable for nonlinear relationships between class labels and attributes [26] if the number of features is not too large. Hence, we use the RBF kernel function in SVM and in the kernel functions in Equations (7), (8), and (9).

BOSFS creates a borderline after training an RBF kernel SVM on the original training dataset. BOSFS introduces a new way of computing the nearest neighbor by using the kernel function in Equations (7) and (9) and the Euclidean distance in Equation (8) to select the k nearest neighbors with the nearest distances of positive instances along the borderline in the feature space. New synthetic positive instances are generated by these new k nearest neighbors with a combination of interpolation and extrapolation techniques, as in BOS. The selection between the interpolation or extrapolation technique depends on the density of negative instances along the borderline[17] [18]. The BOSFS approach finds the density of negative instances along the borderline by nonlinear mapping of these negative instances of the input space to their corresponding points in the feature space. Then, our algorithm over-samples to obtain new synthetic positive instances and adds these new instances to the original training dataset. Using the new training dataset, the improved performance of SVM is assessed by testing on an independent testing dataset. We explain these steps in detail below.

B. Euclidean distance in the feature space

Kernel methods [1],[27],[28] are performed on the feature space H that is generated from the input space X by using a nonlinear map $\Phi(\mathbf{x}_i)$. BOSFS uses the following kernel function (K) representation.

$$K : X \times X \to \mathbb{R}, \qquad K(\mathbf{a}, \mathbf{b}) = \Phi(\mathbf{a})'\Phi(\mathbf{b}) \tag{7}$$

The Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space [24] [29] is d_{ij} and can be computed as follows:

$$\begin{aligned}(d_{ij})^2 &= \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 \\ &= \|\Phi(\mathbf{x}_i)\|^2 + \|\Phi(\mathbf{x}_j)\|^2 + 2\|\Phi(\mathbf{x}_i)\|\|\Phi(\mathbf{x}_j)\| \\ &= K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)\end{aligned} \tag{8}$$

With Equation (8) we can now find the Euclidean distance in the feature space of any $\mathbf{x}_i$, and $\mathbf{x}_j$ in the input space. The nearest neighbor $\mathbf{x}_j$ for each $\mathbf{sv}_i$ on the borderline is determined by the nearest Euclidean distance in the feature space.

Fig. 3 illustrates how our BOSFS algorithm finds the nearest neighbor of each $\mathbf{sv}_i$ on the borderline in the feature space. The k nearest neighbors ($\mathbf{x}_k$) in the input space for each $\mathbf{sv}_i$ on the borderline are selected by BOSFS to create new synthetic positive instances for the new training dataset. The black triangle is an example of a new synthetic positive instance created by the interpolation technique, while the black triangle within white triangle is an example of a new synthetic positive instance created by the extrapolation technique. These new synthetic positive instances in the new training dataset serve to improve the performance of SVM for imbalanced datasets.

We have conducted several experiments with different k = 3, 5, and 7 for k nearest neighbors. There is no significant difference between the results achieved by using k = 3, 5, and 7. Hence, we set k to 5 in all experiments, as was done in SMOTE, Borderline-SMOTE, and BOS.

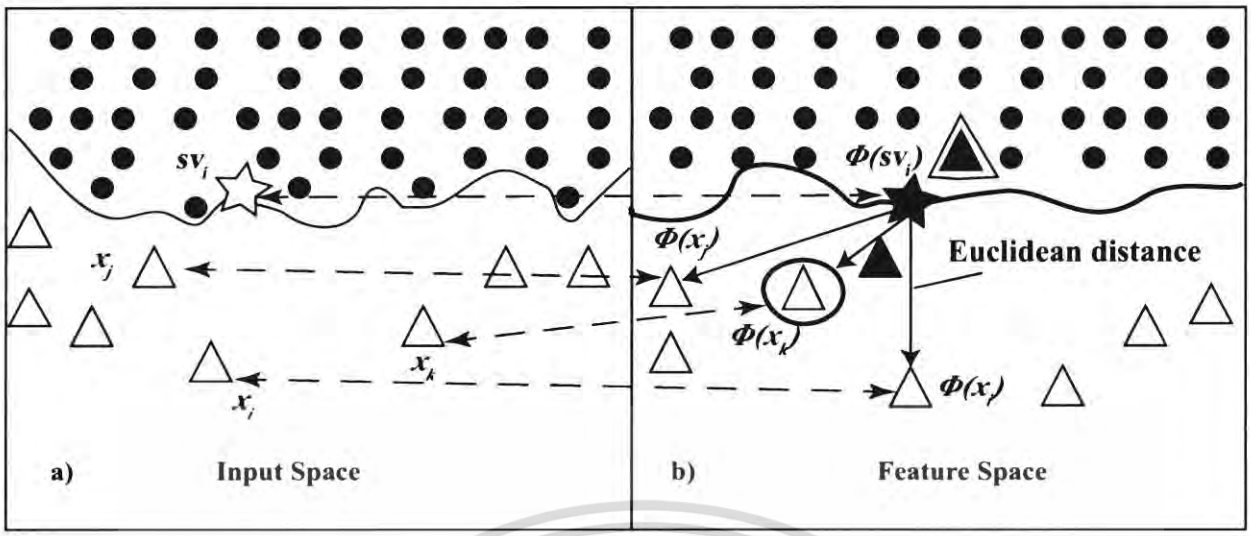
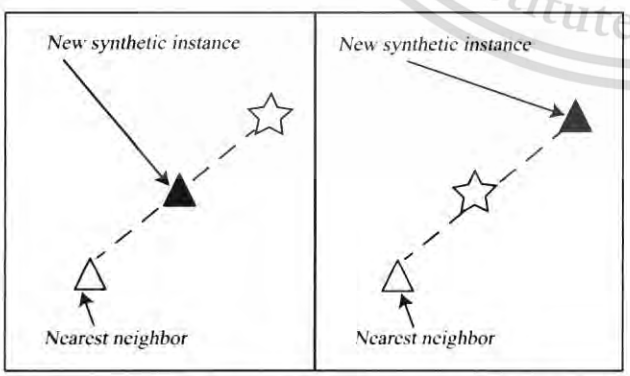


Fig. 3. Illustration of creating the new synthetic positive instances by using the proposed BOSFS. a) $\mathbf{sv}_i$ (white star) is an example of any positive instance in the input space and its corresponding point (black star) is on the borderline created by SVM in the feature space. b) BOSFS uses the RBF kernel function to map any $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_k$, and $\mathbf{sv}_i$ of positive instances in the input space to their corresponding points $\Phi(\mathbf{x}_i)$, $\Phi(\mathbf{x}_j)$, $\Phi(\mathbf{x}_k)$, and $\Phi(\mathbf{sv}_i)$ in the feature space. The dashed double-sided arrows show the non-linear mapping from points in the input space to their corresponding points in the feature space. The solid arrows show the Euclidean distance between $\Phi(\mathbf{sv}_i)$ and any other positive instances $\Phi(\mathbf{x}_i)$, $\Phi(\mathbf{x}_j)$, and $\Phi(\mathbf{x}_k)$ in the feature space. The nearest neighbor of $\Phi(\mathbf{sv}_i)$ is $\Phi(\mathbf{x}_k)$ (white triangle with circle) in the feature space. $\mathbf{x}_k$ is selected for the corresponding point in the input space for the real nearest neighbors of $\Phi(\mathbf{sv}_i)$ on the borderline in the feature space. The black triangle is an example of a new synthetic positive instance created by the interpolation technique. Its position remains in between the black star and the nearest neighbor $\Phi(\mathbf{x}_k)$. The black triangle within the white triangle is an example of a new synthetic positive instance created by the extrapolation technique. Its position is further from the black star and its nearest neighbor $\Phi(\mathbf{x}_k)$.

C. Interpolation and extrapolation techniques in BOSFS

We used an interpolation technique as used in SMOTE [15] to create new data points for a discrete set of known data points in Fig. 4a. A new synthetic positive instance (black triangle) is randomly created using the interpolation technique within the area between the positive instance on the borderline (star) and the nearest neighbor positive instance (white triangle). We determine the difference between the white triangle and star where the minuend is white triangle and the subtrahend is star. A new synthetic positive instance is created by multiplying this difference by a random number between 0 to 1 and adding it to a positive instance (star) under consideration. This computation step of the interpolation technique is shown in step 4 in the BOSFS algorithm. The interpolation technique is applied when there is a crowd density of negative instances near the borderline. This technique then increases the number of positive instances in the crowd density area of negative instances near the borderline.



a) Interpolation like in SMOTE b) Extrapolation like in BOS

Fig. 4. Illustration of the difference between interpolation and extrapolation technique. a) The star is the instance under consideration. The white triangle is the nearest neighbor of the positive instance to the star. The new synthetic instance (black triangle) found by the interpolation technique will be located between the star and white triangle. b) The new synthetic instance found by the extrapolation technique will be located further from the star and white triangle.

In addition, we used an extrapolation technique as in BOS[17] [18] to create new data points by the extension process of estimating beyond the original observation range in Fig. 4b. The extrapolation technique performs the same calculation process as the interpolation technique but the aforementioned difference between the white triangle and star is reversed such that the minuend is the star and the subtrahend is the white triangle. The extrapolation technique can expand the area of the new synthetic positive instance (black triangle) further from the positive instance (star) and its nearest neighbor (white triangle) to the ideal borderline. This technique is applied when there is a lower density of negative instances near the borderline. A low density of negative instances is determined when the number of negative instances is less than half of the number of its nearest neighbors in the feature space. It has been shown in BOS that the new borderline with the extrapolation technique can be shifted towards the ideal borderline. This computation step of the extrapolation technique is shown in step 4 in the BOSFS algorithm.

The BOSFS algorithm uses a combination of Interpolation and extrapolation techniques, as in SMOTE and BOS. The main difference between BOSFS and SMOTE, Borderline-SMOTE and BOS is that BOSFS considers the density of negative instances near the borderline in the feature space rather than in the input space. BOSFS maps all

of the negative instances near the borderline to their corresponding points in the feature space. It computes the Euclidean distance to find the nearest negative instances to the borderline in the feature space. This can be seen in Fig. 3 but the dashed double-sided arrows map the negative instances (black circles) instead of the positive instances (white triangles). The Euclidean distance in Equation (8) using the kernel function in Equations (7) and (9) is recomputed to find the nearest negative instances to the borderline in the feature space.

If the total number of nearest negative instances to the borderline in the feature space is less than half of the number of its nearest neighbors (lower density case), we apply the extrapolation technique. Otherwise, in the crowd density case, we apply the interpolation technique. We must determine the total number of nearest negative instances to the borderline to select the appropriate technique between interpolation and extrapolation to create the new synthetic positive instance.

D. BOSFS algorithm

BOSFS is described as follows:

The notations used for our BOSFeatureSpace algorithm are as follows.

Main variables:

- X : Training dataset
- P^+ : Set of positive instances in X
- P^- : Set of negative instances in X
- N : Over-sampling rate (N is a percent rate such as 100, 200, $\cdots$)
- SV^+: Set of positive instances on the borderline in support vectors (SVs).
- k : Number of nearest neighbors of positive instances to the borderline.
- ϖ : Maximum number of negative instances indicating the threshold density of negative instances to select the interpolation or extrapolation technique.
- np : Total number of new synthetic positive instances to create.
- nn : Total number of negative instances nearest to each $\mathbf{sv}_i$ in SV^+ for the borderline.
- $\S$: Array containing the k nearest neighbors of the positive instances for each $\mathbf{sv}_i^+ \in SV^+$.
- $\mathfrak{D}$: Array of Euclidean distances (d_{ij}) for each $\mathbf{sv}_i^+ \in SV^+$ and its nearest neighbors $\mathbf{x}_j$ in P^+ in the feature space.
- K : RBF kernel function computed as in Equation (9).
- p_{new}^+: New synthetic positive instances for new borderline.
- X_{new}: New over-sampled training dataset

Algorithm: BOSFeatureSpace

Input Parameters: X, N, k, ϖ

Output Parameters: X_{new}

Begin

1) $$np = \left(\frac{N}{100} \times |X|\right)$$

where $|X|$ is the size of the training dataset.

2) Compute SV^+ by training RBF kernel SVMs on X.
3) For each $\mathbf{sv}_i^+ \in SV^+$ and each $\mathbf{x}_j \in P^+$
Perform the mapping function ($\mathbf{x}$, $\mathbf{x}'$) to map $\mathbf{x}$, $\mathbf{x}'$ into the feature space by using the RBF kernel function as follows:

$$K(\mathbf{x}, \mathbf{x}') = exp\left(\frac{-\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{9}$$

where $Sigma$ (σ) is a user-specified parameter. Compute $\mathfrak{D}$ to keep the Euclidean distance d_{ij} in the feature space as follows:

$$d_{ij} = \sqrt{K(\mathbf{sv}_i, \mathbf{sv}_j) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{sv}_i, \mathbf{x}_j)}$$

where $i \neq j$, i = 1, 2, $\cdots$ $|SV^+|$ and j = 1, 2, $\cdots$ $|P^+|$, $|SV^+|$ is the size of the set of positive instances on the borderline, and $|P^+|$ is the size of the set of positive instances in X.
d_{ij} are sorted in ascending order to find $\S[i][k]$ where k signifies the nearest neighbors of the positive instances for each $\mathbf{sv}_i^+$ in the feature space.

4) For each $\mathbf{sv}_i^+ \in SV^+$ and each $\mathbf{x}_j \in P^-$,
repeat step 3 to find the nn negative instances nearest to $\mathbf{x}_j$ for each $\mathbf{sv}_i$ in SV^+ for the borderline.

If nn is less than half of the ϖ nearest neighbors of the negative instances (extrapolation case as in BOS)

Create p_{new}^+ according to np with each $\S[i]$ for synthetic positive instances i using the following formula:

$$p_{new}^+ = \mathbf{sv}_i^+ + \rho(\mathbf{sv}_i^+ - \S[i][j])$$

where $\S[i][j]$ is the j-th positive nearest neighbor of $\mathbf{sv}_i^+$ and ρ is a random number in the range [0,1].

else (interpolation case as in SMOTE)

$$p_{new}^+ = \mathbf{sv}_i^+ + \rho(\S[i][j] - \mathbf{sv}_i^+)$$

5) Combine $\{p_{new}^+\}$, a set of new synthetic positive instances along the borderline, with the original training dataset to form the new training dataset X_{new} as follows:

$$X_{new} = X \cup \{p_{new}^+\} \tag{10}$$

End

V. Experiments and Results

A. Data description

In our experiments, we use a total of five datasets from the UCI machine learning repository [8]: Abalone (5), Glass (7), Page-blocks (4), Spect (0), and Yeast (5). The numbers in the parentheses indicate the class numbers that are selected as positive instances whereby the remaining classes become the negative instances. The dataset statistics and the over-sampling rates performed in our experiments are shown in Table I. The reason we use these five datasets is because

TABLE I
FIVE UCI DATASETS WITH OVER-SAMPLING RATE (%).

Dataset	Attributes	No of Instances	Imbalance ratio	Over-Sampling rate(%)
Abalone	8	4177	35	100, 500, 1000, 1500, 2000, 2500, 3000, 3400
Glass	9	214	6	100, 200, 300, 400, 500
Page-blocks	10	5473	61	100, 1000, 2000, 3000, 4000, 5000, 6000
Spect	22	267	4	100, 200, 300
Yeast	8	1484	28	100, 500, 1000, 1500, 2000, 2500, 2700

they cover a variety of imbalance ratios (Minority:Majority): Spect (1:4), Glass (1:6), Yeast (1:28), Abalone (1:35) and Page (1:61).

B. Experimental setting

We compare our BOSFS algorithm with the SMOTE, Borderline-SMOTE and BOS algorithms. It has already been shown in [17] [18] that BOS outperforms SMOTE [15], Borderline-SMOTE [16], and standard SVM. The borderline instances are derived by the support vectors after training an SVM on the original training set. The number k of nearest neighbors to positive instances along the borderline is to five in all experiments, as was done in SMOTE, Borderline-SMOTE and BOS. We use the RBF kernel SVM as described in Section IV.

We select the over-sampling rates in each experiment according to the imbalance ratio of each dataset in Table I. For example, the imbalance ratio of the Abalone dataset is 1:35 (Minority:Majority). Suppose there is only one minority-class instance in the original training dataset. If we use an over-sampling rate of 100% for the minority-class instances, we will randomly over-sample one minority-class instance, and the new imbalance ratio (Minority:Majority) of the new training dataset after adding the new synthetic minority-class instances will be (2:35). Hence, the over sampling rates of Abalone dataset are varied as follows: 100% (2:35), 500% (6:35), $\cdots$ to the highest rate of 3400% (35:35) to achieve the balanced ratio (1:1).

C. Performance metrics

Using accuracy as a metric to evaluate SVM classification performance is practically useless when coping with significantly imbalanced datasets. This is because if a dataset has an imbalance ratio of 95:5, an SVM classifier that classifies all instances as negative achieves 95% accuracy but is absolutely useless for the task at hand. Several publications use the g-means metric [30], [31], [32] to evaluate classifiers on imbalanced datasets. The g-means metric is defined as follows in [31]:

$$g\text{-}means = \sqrt{acc^+ \cdot acc^-} \tag{11}$$

where *sensitivity* (5) is acc^+ and *specificity* (6) is acc^-.

Another useful performance metric, the F-measure [33], is the harmonic mean of the precision and recall and is defined as:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{12}$$

We use these two performance metrics, g-means and F-measure, to compare SMOTE, Borderline-SMOTE, BOS, and BOSFS. Tables II through VI show our experimental results.

D. Experimental process

First, we use a holdout method to separate each dataset in Table I into two sets: the training dataset and the test dataset.

Second, in Section IV, we conduct the experiments comparing our BOSFS algorithm to SMOTE [15], Borderline-SMOTE [16] and BOS algorithms [17], [18] using MATLAB with the same over-sampling rate or imbalance ratio listed in Table I for each training dataset. we add the set of new synthetic positive instances $\{p^+_{new}\}$ determined by BOSFS, SMOTE, Borderline-SMOTE or BOS to the original training dataset X to form the new training dataset X_{new}, as in Equation (10).

Third, we perform k-fold cross-validation where k = 5 for each dataset with a variety of values for the C parameter in the range of [0,1] and a variety of values for the $Sigma$ (σ) parameter in the range of [0,1] to determine the suitable C and $Sigma$ parameters to achieve the best g-means (Equation 11) and F-measure (Equation 12) using SMOTE, Borderline-SMOTE, BOS, or BOSFS. This step solves SVMs soft-margin problem by using different C and $Sigma$ parameters in the RBF kernel function (Equation 9), as explained above.

Fourth, we train the SVM on the new training dataset X_{new} with the suitable C and $Sigma$ parameters computed in the previous step.

Fifth, we execute SVM with the suitable C and $Sigma$ parameters on the testing dataset previously prepared using the holdout method in the first step to finally obtain the acc^+, acc^-, g-means (Equation 11), and F-measure (Equation 12) for each experiment. These metrics are recorded and averaged for each imbalance ratio.

To reduce the effect of randomness in the data division and sampling, we perform the first step through the fifth step 10 times for each over-sampling rate and dataset combination. The values of acc^+, acc^-, g-means, and F-measure are averaged after 10 experiments.

E. Experimental results and evaluation

In Tables II through VI, the values in bold correspond to the best values attained in each experiment. It is clear that BOSFS achieves the greatest number of superior values. For the Glass dataset with imbalance ratios of (3:6) and (4:6), we observe that the BOSFS method is only inferior to SMOTE according to the g-means and F-measure but still better than BOS. The acc^+ values attained by our BOSFS method are generally better than those of SMOTE, Borderline-SMOTE and BOS, especially for the Abalone, Glass, Spect, and Yeast datasets. We observe that the acc^+ values of all of the methods can reach 100% for the low degrees of imbalance shown in the Glass and Spect datasets. This means that all of the methods can perform well for low degrees of imbalance, while the BOSFS method outperforms the other methods for higher degrees of imbalance such as Abalone (2:35), Page (2:61), and Yeast (2:28). The values of acc^+ and acc^- attained by SMOTE, Borderline-SMOTE, BOS, and BOSFS vary because of the dependency on the tradeoff between the C^+ and C^- parameter and the imbalance ratio of each dataset. In summary, our BOSFS method accomplishes better g-means and F-measure performance than the SMOTE, Borderline-SMOTE and BOS methods at almost all over-sampling rates and imbalance ratios.

TABLE II
ABALONE DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS, AND BOSFS ALGORITHM USING k = 5 NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)								
	2:35	6:35	11:35	16:35	21:35	26:35	31:35	35:35	Average
SMOTE									
acc^+	31.97	60.18	75.59	82.75	85.91	87.79	89.69	90.39	73.41
acc^-	**98.62**	99.08	97.85	98.15	**98.6**	98.21	98.7	98.89	98.46
g-means	56.08	77.2	85.98	90.08	92.03	92.85	94.09	94.54	84.04
F-measure	44.81	74.42	84.86	89.74	91.95	92.98	94.25	94.69	81.86
Borderline-SMOTE									
acc^+	37.11	63.08	74.16	80.29	85.08	86.98	**93.19**	89.61	76.19
acc^-	95.97	95.33	98.45	98.95	98.41	**98.79**	91.8	**99.58**	97.16
g-means	59.32	77.19	85.44	89.13	91.49	92.69	92.48	94.46	85.28
F-measure	38.94	65.89	84.37	88.67	91.44	92.7	94.04	94.43	81.31
BOS									
acc^+	35.13	61.24	74.79	81.67	85.04	87.75	89.47	90.13	75.66
acc^-	97.61	**99.53**	**99.19**	**99.01**	98.06	98.34	**98.87**	99.42	**98.66**
g-means	58.48	78.06	86.12	89.93	91.31	92.89	94.05	94.66	85.63
F-measure	45.1	75.6	85.14	89.53	91.28	93.02	94.17	94.68	83.51
BOSFS									
acc^+	**47.61**	**72.41**	**81.64**	**86**	**89.09**	**90.93**	91.58	**92.31**	**79.89**
acc^-	97.05	96.16	98.02	98.44	98.31	98.51	97.75	98.86	97.75
g-means	**67.94**	**83.41**	**89.45**	**92.01**	**93.59**	**94.64**	**94.61**	**95.53**	**87.95**
F-measure	**55.54**	**79.22**	**88.7**	**91.79**	**93.64**	**94.8**	**94.99**	**95.75**	**85.53**

TABLE III
PAGE DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS, AND BOSFS ALGORITHM USING k = 5 NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)							
	2:61	11:61	21:61	31:61	41:61	51:61	61:61	Average
SMOTE								
acc^+	94.02	86.55	89.66	92.44	94.72	95.26	96.36	92.72
acc^-	95.94	93.06	96.29	98.44	98.09	96.61	97.08	96.50
g-means	94.92	89.60	92.87	95.39	96.38	95.93	96.71	94.54
F-measure	38.47	78.28	91.48	95.33	96.57	96.49	97.33	84.85
Borderline-SMOTE								
acc^+	88.58	93.54	93.50	95.67	96.94	96.30	97.31	94.09
acc^-	97.55	91.31	91.31	93.60	93.96	96.22	96.55	93.99
g-means	92.84	92.40	92.38	94.62	95.43	96.26	96.93	93.99
F-measure	67.78	81.73	89.74	94.34	95.80	96.86	97.64	87.71
BOS								
acc^+	97.67	92.55	**94.80**	**97.42**	**96.99**	**97.53**	97.43	**96.34**
acc^-	97.35	**96.05**	94.25	98.36	98.47	98.13	97.81	97.20
g-means	97.48	94.26	94.48	**97.88**	97.72	97.82	97.62	96.75
F-measure	68.42	**90.03**	92.31	**97.85**	97.88	98.13	98.10	91.82
BOSFS								
acc^+	**98.85**	**95.35**	93.37	94.55	96.67	96.96	**97.78**	96.13
acc^-	**97.55**	95.56	**97.72**	**98.62**	**98.98**	**99.09**	**99.08**	**98.16**
g-means	**98.19**	**95.41**	**95.52**	96.56	**97.81**	**98.01**	**98.42**	**97.12**
F-measure	**70.63**	90.02	**94.88**	96.51	**97.91**	**98.16**	**98.62**	**92.38**

TABLE IV
GLASS DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS, AND BOSFS ALGORITHM USING k = 5 NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)					
	2:6	3:6	4:6	5:6	6:6	Average
SMOTE						
acc^+	**100.00**	**100.00**	99.60	92.44	94.72	97.35
acc^-	94.60	**97.28**	**99.33**	**98.44**	98.09	**97.55**
g-means	97.24	**98.62**	**99.46**	95.39	96.38	97.42
F-measure	85.29	**97.28**	**99.19**	95.33	96.57	94.73
Borderline-SMOTE						
acc^+	99.00	96.41	96.83	97.72	96.52	97.30
acc^-	94.41	92.69	88.86	89.09	90.15	91.04
g-means	96.67	94.49	92.72	93.26	93.13	94.05
F-measure	86.35	91.56	89.40	92.60	94.27	90.84
BOS						
acc^+	**100.00**	**100.00**	**100.00**	99.00	96.82	99.16
acc^-	94.00	90.57	90.18	90.93	**98.21**	92.78
g-means	96.92	95.15	94.94	94.78	**97.47**	95.85
F-measure	87.57	89.83	92.54	93.09	97.56	92.12
BOSFS						
acc^+	**100.00**	**100.00**	**100.00**	**99.66**	**100.00**	**99.93**
acc^-	**97.06**	94.46	93.55	95.98	94.93	95.20
g-means	**98.51**	97.17	96.71	**97.79**	97.41	**97.52**
F-measure	**94.80**	93.95	95.23	**97.67**	**97.67**	**95.86**

TABLE V
YEAST DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS, AND BOSFS ALGORITHM USING k = 5 NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)							
	2:28	6:28	11:28	16:28	21:28	26:28	28:28	Average
SMOTE								
acc^+	**100.00**	92.66	85.61	89.40	92.49	92.79	93.23	92.31
acc^-	93.35	86.58	93.58	92.28	92.78	91.60	92.51	91.81
g-means	96.61	89.37	89.50	90.82	92.61	92.17	92.86	91.99
F-measure	39.86	66.82	88.47	90.48	93.53	93.85	94.43	81.06
Borderline-SMOTE								
acc^+	96.11	94.25	89.16	89.11	91.18	92.65	93.63	92.30
acc^-	92.77	85.04	91.45	95.57	96.71	97.67	97.99	93.89
g-means	94.34	89.34	90.26	92.25	93.89	95.13	95.78	93.00
F-measure	34.56	63.75	88.11	92.33	94.28	95.53	96.20	80.68
BOS								
acc^+	**100.00**	91.86	92.65	93.82	95.45	95.58	96.07	95.06
acc^-	93.71	94.15	96.56	95.83	95.07	95.04	93.93	94.90
g-means	96.80	92.96	94.58	94.80	95.26	95.31	94.99	94.96
F-measure	55.52	86.59	93.65	94.88	95.85	96.13	96.20	88.40
BOSFS								
acc^+	**100.00**	**95.17**	**96.19**	**96.47**	**97.27**	**97.89**	**97.99**	**97.28**
acc^-	**94.96**	**97.18**	**96.71**	**97.13**	**97.57**	**97.39**	**97.79**	**96.96**
g-means	**97.44**	**96.15**	**96.44**	**96.80**	**97.42**	**97.64**	**97.89**	**97.11**
F-measure	**61.00**	**93.70**	**95.74**	**96.88**	**97.67**	**98.13**	**98.35**	**91.64**

TABLE VI
SPECT DATASET: acc^+, acc^-, G-MEANS, AND F-MEASURE OF SMOTE, BORDERLINE-SMOTE, BOS, AND BOSFS ALGORITHM USING $k = 5$ NEAREST NEIGHBORS.

Algorithms / Metrics	Imbalance Ratio (Minority : Majority)			
	2:4	3:4	4:4	Average
SMOTE				
acc^+	99.33	94.83	90.41	94.86
acc^-	87.35	64.91	51.88	68.05
g-means	93.11	77.60	67.09	79.27
F-measure	90.92	76.47	69.97	79.12
Borderline-SMOTE				
acc^+	**100.00**	94.29	93.18	95.82
acc^-	65.61	53.21	49.49	56.10
g-means	80.52	70.28	67.50	72.77
F-measure	60.54	67.37	67.85	65.25
BOS				
acc^+	**100.00**	98.61	96.18	98.26
acc^-	87.16	82.75	**84.04**	84.65
g-means	93.28	90.18	89.62	91.03
F-measure	90.35	93.01	92.08	91.81
BOSFS				
acc^+	**100.00**	**100.00**	**98.66**	**99.55**
acc^-	**89.99**	**86.77**	83.04	**86.60**
g-means	**94.79**	**93.07**	**90.40**	**92.75**
F-measure	**93.30**	**94.19**	**93.74**	**93.74**

VI. CONCLUSION

The imbalance class data problem has impacted the prediction performance of SVM classifiers. In this paper, we proposed a new over-sampling method called BOSFS that focuses on the k nearest neighbors of the positive instances along the borderline. BOSFS finds these nearest neighbors using the Euclidean distance and kernel function in the feature space, rather than in the input space as is done in SMOTE, Borderline-SMOTE, and BOS. We also introduce a new way to find the density of the nearest negative instances in the feature space along the borderline. We determine the appropriate technique to generate new synthetic instances (i.e., interpolation or extrapolation) by considering the density of negative instances in the feature space. Thus, the BOSFS algorithm achieves superior SVM classification performance in terms of g-means and the F-measure for imbalanced datasets, as shown in Tables II through VI. We conclude that our BOSFS algorithm is better suited than the existing SMOTE, Borderline-SMOTE, and BOS algorithms for effectively determining new positive instances to improve SVM prediction in imbalanced data environments.

REFERENCES

[1] V. N. Vapnik, "The nature of statistical learning theory," 1995.

[2] T. Fawcett and F. J. Provost, "Combining data mining and machine learning for effective user profiling.," in *KDD*, pp. 8–13, 1996.

[3] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine learning*, vol. 30, no. 2-3, pp. 195–215, 1998.

[4] K. J. Ezawa, M. Singh, and S. W. Norton, "Learning goal oriented bayesian networks for telecommunications risk management," in *ICML*, pp. 139–147, 1996.

[5] P. K. Chan and S. J. Stolfo, "Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection.," in *KDD*, vol. 1998, pp. 164–168, 1998.

[6] A. Uyar, A. Bener, H. Ciracy, and M. Bahceci, "Handling the imbalance problem of ivf implantation prediction," *IAENG International Journal of Computer Science*, vol. 37, pp. 164–170, 2010.

[7] N. Poolsawad, C. Kambhampati, and J. Cleland, "Balancing class for performance of classification with a clinical dataset," *Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering 2014, WCE 2014*, 2-4 July, 2014, London, U.K., pp 237-242.

[8] M. Lichman, "UCI machine learning repository," 2013.

[9] K. S. Woods, C. C. Doss, K. W. Bowyer, J. L. Solka, C. E. Priebe, and W. P. Kegelmeyer JR, "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 06, pp. 1417–1436, 1993.

[10] R. Batuwita and V. Palade, "Class imbalance learning methods for support vector machines," *Imbalanced learning: Foundations, algorithms, and applications*, pp. 83–99, 2013.

[11] R. Batuwita and V. Palade, "Efficient resampling methods for training support vector machines with imbalanced datasets," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pp. 1–8, IEEE, 2010.

[12] H. He, E. Garcia, *et al.*, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, 2009.

[13] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 2, pp. 539–550, 2009.

[14] I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of Workshop on Learning from Imbalanced Datasets*, 2003.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, pp. 321–357, 2002.

[16] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *Advances in intelligent computing*, pp. 878–887, Springer, 2005.

[17] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," in *Proc. IEEE SMC Hiroshima Chapter Fifth International Workshop on Computational Intelligence and Applications*, (Hiroshima University, Japan), pp. 24–29, 2009.

[18] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *International Journal of Knowledge Engineering and Soft Data Paradigms*, vol. 3, no. 1, pp. 4–21, 2011.

[19] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[20] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the border: active learning in imbalanced data classification," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 127–136, ACM, 2007.

[21] K. Veropoulos, C. Campbell, N. Cristianini, *et al.*, "Controlling the sensitivity of support vector machines," in *Proceedings of the international joint conference on AI*, pp. 55–60, 1999.

[22] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," in *Machine Learning: ECML 2004*, pp. 39–50, Springer, 2004.

[23] G. Wu and E. Y. Chang, "Kba: Kernel boundary alignment considering imbalanced data distribution," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 786–795, 2005.

[24] H. Zhang, "Distance-based classifier via the kernel trick," in *International Journal Software Informatics*, Vol.2, pp. 121–133, 2010.

[25] H.-Y. Wang, "Combination approach of smote and biased-svm for imbalanced datasets," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pp. 228–231, IEEE, 2008.

[26] C.-W. Hsu, C.-C. Chang, C.-J. Lin, *et al.*, "A practical guide to support vector classification," 2003.

[27] R. Herbrich, "Learning kernel classifiers," MIT Press, Cambridge, 2002.

[28] B. Schölkopf and A. J. Smola, "Learning with kernels. 2002.", MIT Press, Cambridge, 2002.

[29] Y. Li, Z. Hu, Y. Cai, and W. Zhang, "Support vector based prototype selection method for nearest neighbor rules," in *Advances in Natural Computation*, pp. 528–535, Springer, 2005.

[30] M. Kubat and R. Holte, "S. matwin,learning when negative example abound," in *Proceedings of the 9th European Conference on Machine Learning, ECML*, vol. 97, 1997.

[31] M. Kubat, S. Matwin, *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *ICML*, vol. 97, pp. 179–186, Nashville, USA, 1997.

[32] G. Wu and E. Y. Chang, "Class-boundary alignment for imbalanced dataset learning," in *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*, pp. 49–56, 2003.
[33] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, pp. 37–63, 2011.

Appendix B
Experiments for different k nearest neighbors

We explained the reason that we set k nearest neighbors, k = 5 because we conducted three main experiments using five UCI dataset [37] with different k nearest neighbors as k = 3, 5 and 7, respectively. The result of performance metrics, g-means and F-measure of different k nearest neighbors using Multivariate analysis in SPSS package as shown in Figure Appendix B_1 and Appendix B_2.

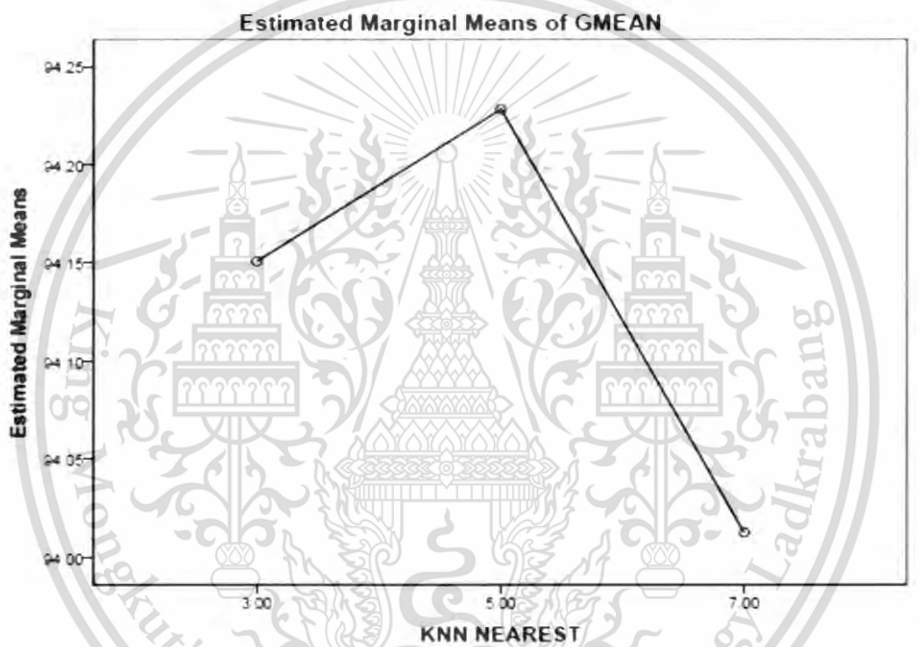


Figure Appendix B_1 Illustration of Graph represents g-means of different k = 3, 5 and 7.

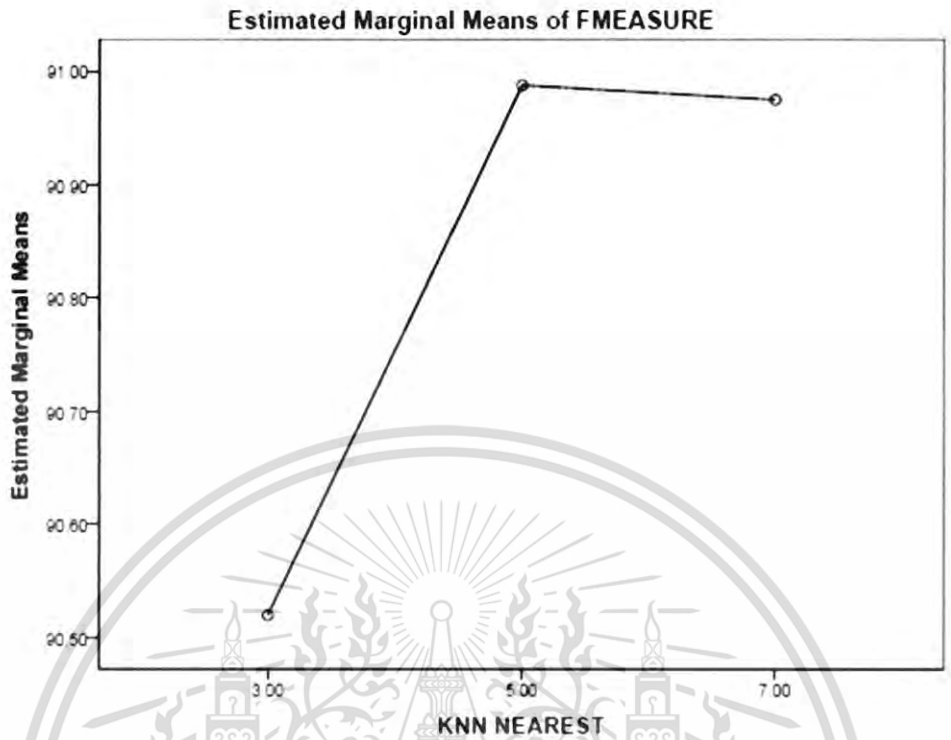


Figure Appendix B_2 Illustration of Graph represents F-measure of different k = 3, 5 and 7.

In Figure Appendix B_1 and Figure Appendix B_2, It showed that experiments with k = 5 for k nearest neighbors is the best performance metrics, g-means and F-measure of different k nearest neighbors.

Details of Multivariate analysis with a 99% confidence interval in SPSS package on the next seventeen pages using dataset code according to five UCI datasets in Table 4.7 of chapter 4.

DATASET CODE	UCI DATASET NAME
11	Abalone
22	Glass
33	Page
44	Spect
55	Yeast

```
GLM GMEAN FMEASURE BY K METHOD DATASET
  /METHOD=SSTYPE(3)
  /INTERCEPT=INCLUDE
  /POSTHOC=K METHOD DATASET(FREGW)
  /PLOT=PROFILE(K METHOD DATASET)
  /EMMEANS=TABLES(K)
  /EMMEANS=TABLES(METHOD)
  /EMMEANS=TABLES(DATASET)
  /PRINT=DESCRIPTIVE HOMOGENEITY
  /CRITERIA=ALPHA(.01)
  /DESIGN= K METHOD DATASET K*METHOD K*DATASET METHOD*DATASET K*METHOD*DAT
ASET.
```

General Linear Model

```
[DataSet1] G:\NEWPHD\ALL_RESULTS\New Folder\ALL_UCI_DATASET_TRANSFORM.sav
```

Warnings

Post hoc tests are not performed for METHOD because there are fewer than three groups.

Between-Subjects Factors

		N
KNN NEAREST	3.00	60
	5.00	60
	7.00	60
METHOD	1	90
	2	90
DATASET	11	48
	22	30
	33	18
	44	42
	55	42

Descriptive Statistics

	KNN NEAREST	METHOD	DATASET	Mean	Std. Deviation	N
GMEAN	3.00	1	11	85.5363	12.55477	8
			22	94.7220	1.20891	5
			33	89.8967	2.06558	3
			44	96.1186	1.61753	7
			55	95.6786	1.21619	7
			Total	92.3390	7.73902	30
		2	11	90.9025	11.54829	8
			22	97.2540	1.81817	5
			33	94.9067	3.34032	3
			44	98.7800	1.12897	7
			55	97.7129	.44980	7
			Total	95.7887	6.61910	30
		Total	11	88.2194	11.97799	16
			22	95.9880	1.97474	10
			33	92.4017	3.70132	6
			44	97.4493	1.92428	14
			55	96.6957	1.37485	14
			Total	94.0638	7.34841	60
	5.00	1	11	85.6875	12.25059	8
			22	95.8520	1.24823	5
			33	91.0267	1.97143	3
			44	96.7514	1.63327	7
			55	94.9571	1.13790	7
			Total	92.6600	7.63456	30
		2	11	91.2838	11.75050	8
			22	97.5220	1.43542	5
			33	93.3167	1.80362	3
			44	98.7229	1.18073	7
			55	97.1614	.54566	7
			Total	95.6340	6.59119	30
		Total	11	88.4856	11.95082	16
			22	96.6870	1.54367	10
			33	92.1717	2.10453	6
			44	97.7371	1.70910	14
			55	96.0593	1.42940	14
			Total	94.1470	7.22853	60
	7.00	1	11	85.4300	12.49314	8
			22	95.9960	.68226	5
			33	89.9167	3.91838	3
			44	95.5914	1.74102	7
			55	94.0471	1.44365	7
			Total	92.0213	7.68698	30

Descriptive Statistics

	KNN NEAREST	METHOD	DATASET	Mean	Std. Deviation	N
		2	11	91.5313	11.85617	8
			22	97.8600	2.22410	5
			33	93.4533	3.21405	3
			44	98.8829	1.18534	7
			55	97.4171	.49115	7
			Total	95.8670	6.70183	30
		Total	11	88.4806	12.18040	16
			22	96.9280	1.83589	10
			33	91.6850	3.74512	6
			44	97.2371	2.22804	14
			55	95.7321	2.03245	14
			Total	93.9442	7.40815	60
	Total	1	11	85.5512	11.88114	24
			22	95.5233	1.15906	15
			33	90.2800	2.48805	9
			44	96.1538	1.65248	21
			55	94.8943	1.38754	21
			Total	92.3401	7.60463	90
		2	11	91.2392	11.20103	24
			22	97.5453	1.73562	15
			33	93.8922	2.60146	9
			44	98.7952	1.10757	21
			55	97.4305	.52507	21
			Total	95.7632	6.56326	90
		Total	11	88.3952	11.77862	48
			22	96.5343	1.77768	30
			33	92.0861	3.09060	18
			44	97.4745	1.92802	42
			55	96.1624	1.64952	42
			Total	94.0517	7.28816	180
FMEASURE	3.00	1	11	83.1975	17.42320	8
			22	87.9560	10.50189	5
			33	90.7767	3.52750	3
			44	91.2114	10.17611	7
			55	87.5043	18.00350	7
			Total	87.6233	13.68916	30
		2	11	89.4450	15.61366	8
			22	91.5320	12.95104	5
			33	95.0767	4.52107	3
			44	96.3371	6.59110	7
			55	92.1600	14.02704	7
			Total	92.5977	11.83560	30

Descriptive Statistics

KNN NEAREST	METHOD	DATASET	Mean	Std. Deviation	N
	Total	11	86.3213	16.30461	16
		22	89.7440	11.27459	10
		33	92.9267	4.32439	6
		44	93.7743	8.65550	14
		55	89.8321	15.69215	14
		Total	90.1105	12.93263	60
5.00	1	11	83.5650	16.75874	8
		22	92.1180	3.76356	5
		33	91.8133	1.34990	3
		44	91.8171	10.82700	7
		55	88.4029	14.88858	7
		Total	88.8697	12.35378	30
	2	11	89.8225	15.79056	8
		22	92.6480	9.18551	5
		33	93.6633	4.96754	3
		44	96.0757	6.65873	7
		55	89.9486	18.84512	7
		Total	92.1660	12.76086	30
	Total	11	86.6938	16.05824	16
		22	92.3830	6.62365	10
		33	92.7383	3.40972	6
		44	93.9464	8.91347	14
		55	89.1757	16.33592	14
		Total	90.5178	12.56252	60
7.00	1	11	82.8575	18.24738	8
		22	92.5840	3.61164	5
		33	91.4933	2.83784	3
		44	90.8657	10.95447	7
		55	87.4229	12.91486	7
		Total	88.2760	12.49908	30
	2	11	90.0863	15.89783	8
		22	92.1120	13.33694	5
		33	94.5600	4.15966	3
		44	96.3671	7.37984	7
		55	91.3971	15.39938	7
		Total	92.6427	12.37013	30
	Total	11	86.4719	16.94889	16
		22	92.3480	9.21489	10
		33	93.0267	3.60053	6
		44	93.6164	9.41646	14
		55	89.4100	13.80883	14
		Total	90.4593	12.52400	60

Descriptive Statistics

KNN NEAREST	METHOD	DATASET	Mean	Std. Deviation	N
Total	1	11	83.2067	16.71205	24
		22	90.8860	6.62744	15
		33	91.3611	2.40646	9
		44	91.2981	10.11907	21
		55	87.7767	14.62819	21
		Total	88.2563	12.72623	90
	2	11	89.7846	15.06904	24
		22	92.0973	11.09383	15
		33	94.4333	3.99855	9
		44	96.2600	6.53383	21
		55	91.1686	15.41410	21
		Total	92.4688	12.19064	90
	Total	11	86.4956	16.08865	48
		22	91.4917	8.99991	30
		33	92.8972	3.57037	18
		44	93.7790	8.77948	42
		55	89.4726	14.94083	42
		Total	90.3626	12.60467	180

Box's Test of Equality of Covariance Matrices[a]

Box's M	622.597
F	5.706
df1	87
df2	4434.477
Sig.	.000

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + K + METHOD + DATASET + K * METHOD + K * DATASET + METHOD *

a. Design: Intercept + K + METHOD + DATASET + K * METHOD + K * DATASET + METHOD * DATASET + K * METHOD * DATASET

Multivariate Tests[a]

Effect		Value	F	Hypothesis df	Error df
Intercept	Pillai's Trace	.996	17793.262[b]	2.000	149.000
	Wilks' Lambda	.004	17793.262[b]	2.000	149.000
	Hotelling's Trace	238.836	17793.262[b]	2.000	149.000
	Roy's Largest Root	238.836	17793.262[b]	2.000	149.000
K	Pillai's Trace	.001	.038	4.000	300.000
	Wilks' Lambda	.999	.038[b]	4.000	298.000
	Hotelling's Trace	.001	.037	4.000	296.000
	Roy's Largest Root	.001	.067[c]	2.000	150.000
METHOD	Pillai's Trace	.065	5.162[b]	2.000	149.000
	Wilks' Lambda	.935	5.162[b]	2.000	149.000
	Hotelling's Trace	.069	5.162[b]	2.000	149.000
	Roy's Largest Root	.069	5.162[b]	2.000	149.000
DATASET	Pillai's Trace	.367	8.432	8.000	300.000
	Wilks' Lambda	.643	9.219[b]	8.000	298.000
	Hotelling's Trace	.541	10.009	8.000	296.000
	Roy's Largest Root	.511	19.178[c]	4.000	150.000
K * METHOD	Pillai's Trace	.002	.059	4.000	300.000
	Wilks' Lambda	.998	.059[b]	4.000	298.000
	Hotelling's Trace	.002	.058	4.000	296.000
	Roy's Largest Root	.001	.076[c]	2.000	150.000
K * DATASET	Pillai's Trace	.004	.034	16.000	300.000
	Wilks' Lambda	.996	.034[b]	16.000	298.000
	Hotelling's Trace	.004	.034	16.000	296.000
	Roy's Largest Root	.002	.042[c]	8.000	150.000
METHOD * DATASET	Pillai's Trace	.019	.351	8.000	300.000
	Wilks' Lambda	.982	.349[b]	8.000	298.000
	Hotelling's Trace	.019	.347	8.000	296.000
	Roy's Largest Root	.015	.571[c]	4.000	150.000
K * METHOD * DATASET	Pillai's Trace	.003	.029	16.000	300.000
	Wilks' Lambda	.997	.029[b]	16.000	298.000
	Hotelling's Trace	.003	.029	16.000	296.000
	Roy's Largest Root	.002	.038[c]	8.000	150.000

Multivariate Tests[a]

Effect		Sig.
Intercept	Pillai's Trace	.000
	Wilks' Lambda	.000
	Hotelling's Trace	.000
	Roy's Largest Root	.000
K	Pillai's Trace	.997
	Wilks' Lambda	.997
	Hotelling's Trace	.997
	Roy's Largest Root	.935
METHOD	Pillai's Trace	.007
	Wilks' Lambda	.007
	Hotelling's Trace	.007
	Roy's Largest Root	.007
DATASET	Pillai's Trace	.000
	Wilks' Lambda	.000
	Hotelling's Trace	.000
	Roy's Largest Root	.000
K * METHOD	Pillai's Trace	.993
	Wilks' Lambda	.994
	Hotelling's Trace	.994
	Roy's Largest Root	.927
K * DATASET	Pillai's Trace	1.000
	Wilks' Lambda	1.000
	Hotelling's Trace	1.000
	Roy's Largest Root	1.000
METHOD * DATASET	Pillai's Trace	.945
	Wilks' Lambda	.946
	Hotelling's Trace	.947
	Roy's Largest Root	.684
K * METHOD * DATASET	Pillai's Trace	1.000
	Wilks' Lambda	1.000
	Hotelling's Trace	1.000
	Roy's Largest Root	1.000

a. Design: Intercept + K + METHOD + DATASET + K * METHOD + K * DATASET + METHOD * DATASET + K * METHOD * DATASET

b. Exact statistic

c. The statistic is an upper bound on F that yields a lower bound on the significance level.

Levene's Test of Equality of Error Variances[a]

	F	df1	df2	Sig.
GMEAN	3.381	29	150	.000
FMEASURE	.851	29	150	.687

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + K + METHOD + DATASET + K * METHOD + K * DATASET + METHOD * DATASET + K * METHOD * DATASET

Tests of Between-Subjects Effects

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square
Corrected Model	GMEAN	3117.897[a]	29	107.514
	FMEASURE	2449.051[b]	29	84.450
Intercept	GMEAN	1407855.867	1	1407855.867
	FMEASURE	1310779.242	1	1310779.242
K	GMEAN	1.263	2	.632
	FMEASURE	7.520	2	3.760
METHOD	GMEAN	432.566	1	432.566
	FMEASURE	586.666	1	586.666
DATASET	GMEAN	2469.420	4	617.355
	FMEASURE	1395.145	4	348.786
K * METHOD	GMEAN	6.169	2	3.084
	FMEASURE	23.720	2	11.860
K * DATASET	GMEAN	14.355	8	1.794
	FMEASURE	45.243	8	5.655
METHOD * DATASET	GMEAN	91.112	4	22.778
	FMEASURE	153.513	4	38.378
K * METHOD * DATASET	GMEAN	8.752	8	1.094
	FMEASURE	29.165	8	3.646
Error	GMEAN	6390.103	150	42.601
	FMEASURE	25990.071	150	173.267
Total	GMEAN	1601736.880	180	
	FMEASURE	1498209.582	180	
Corrected Total	GMEAN	9508.000	179	
	FMEASURE	28439.122	179	

Tests of Between-Subjects Effects

Source	Dependent Variable	F	Sig.
Corrected Model	GMEAN	2.524	.000
	FMEASURE	.487	.987
Intercept	GMEAN	33047.729	.000
	FMEASURE	7565.077	.000
K	GMEAN	.015	.985
	FMEASURE	.022	.979
METHOD	GMEAN	10.154	.002
	FMEASURE	3.386	.068
DATASET	GMEAN	14.492	.000
	FMEASURE	2.013	.095
K * METHOD	GMEAN	.072	.930
	FMEASURE	.068	.934
K * DATASET	GMEAN	.042	1.000
	FMEASURE	.033	1.000
METHOD * DATASET	GMEAN	.535	.710
	FMEASURE	.221	.926
K * METHOD * DATASET	GMEAN	.026	1.000
	FMEASURE	.021	1.000
Error	GMEAN		
	FMEASURE		
Total	GMEAN		
	FMEASURE		
Corrected Total	GMEAN		
	FMEASURE		

a. R Squared = .328 (Adjusted R Squared = .198)

b. R Squared = .086 (Adjusted R Squared = -.091)

Estimated Marginal Means

1. KNN NEAREST

Dependent Variable	KNN NEAREST	Mean	Std. Error	99% Confidence Interval	
				Lower Bound	Upper Bound
GMEAN	3.00	94.151	.897	91.811	96.491
	5.00	94.228	.897	91.888	96.568
	7.00	94.013	.897	91.673	96.352
FMEASURE	3.00	90.520	1.809	85.801	95.239
	5.00	90.987	1.809	86.269	95.706
	7.00	90.975	1.809	86.256	95.694

2. METHOD

Dependent Variable	METHOD	Mean	Std. Error	99% Confidence Interval	
				Lower Bound	Upper Bound
GMEAN	1	92.481	.732	90.570	94.391
	2	95.780	.732	93.870	97.691
FMEASURE	1	88.906	1.477	85.053	92.759
	2	92.749	1.477	88.896	96.602

3. DATASET

Dependent Variable	DATASET	Mean	Std. Error	99% Confidence Interval	
				Lower Bound	Upper Bound
GMEAN	11	88.395	.942	85.937	90.853
	22	96.534	1.192	93.425	99.643
	33	92.086	1.538	88.072	96.100
	44	97.475	1.007	94.847	100.102
	55	96.162	1.007	93.535	98.790
FMEASURE	11	86.496	1.900	81.539	91.453
	22	91.492	2.403	85.222	97.762
	33	92.897	3.103	84.803	100.992
	44	93.779	2.031	88.480	99.078
	55	89.473	2.031	84.173	94.772

Post Hoc Tests

KNN NEAREST

Homogeneous Subsets

GMEAN

Ryan-Einot-Gabriel-Welsch F[a]

KNN NEAREST	N	Subset 1
7.00	60	93.9442
3.00	60	94.0638
5.00	60	94.1470
Sig.		.985

Means for groups in homogeneous subsets are displayed.
Based on observed means.
The error term is Mean Square(Error) = 42.601.

a. Alpha = .01.

FMEASURE

Ryan-Einot-Gabriel-Welsch F[a]

KNN NEAREST	N	Subset 1
3.00	60	90.1105
7.00	60	90.4593
5.00	60	90.5178
Sig.		.983

Means for groups in homogeneous subsets are displayed.
Based on observed means.
The error term is Mean Square(Error) = 173.267.

a. Alpha = .01.

DATASET

Homogeneous Subsets

GMEAN

Ryan-Einot-Gabriel-Welsch F[a]

DATASET	N	Subset 1	Subset 2
11	48	88.3952	
33	18	92.0861	92.0861
55	42		96.1624
22	30		96.5343
44	42		97.4745
Sig.		.103	.035

Means for groups in homogeneous subsets are displayed.
Based on observed means.
The error term is Mean Square(Error) = 42.601.

a. Alpha = .01.

FMEASURE

Ryan-Einot-Gabriel-Welsch F[a]

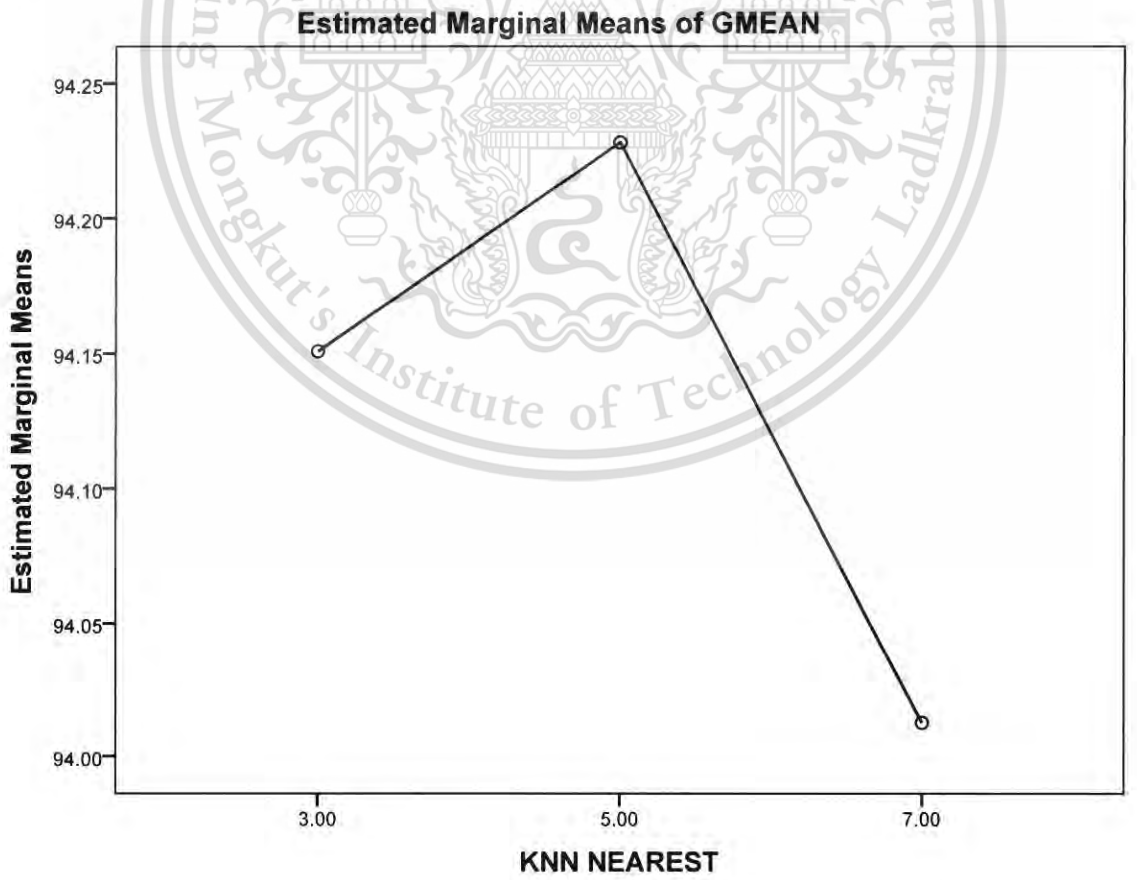
DATASET	N	Subset 1
11	48	86.4956
55	42	89.4726
22	30	91.4917
33	18	92.8972
44	42	93.7790
Sig.		.095

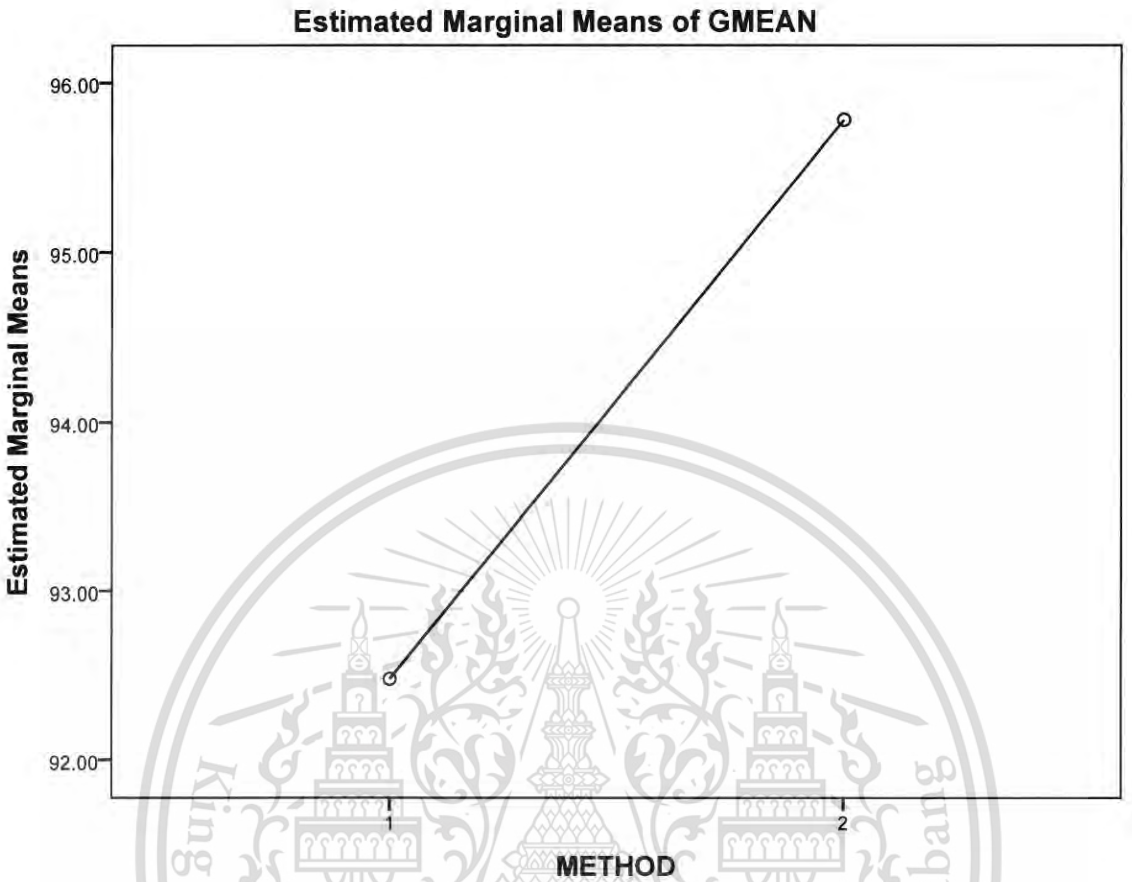
Means for groups in homogeneous subsets are displayed.
Based on observed means.
The error term is Mean Square (Error) = 173.267.

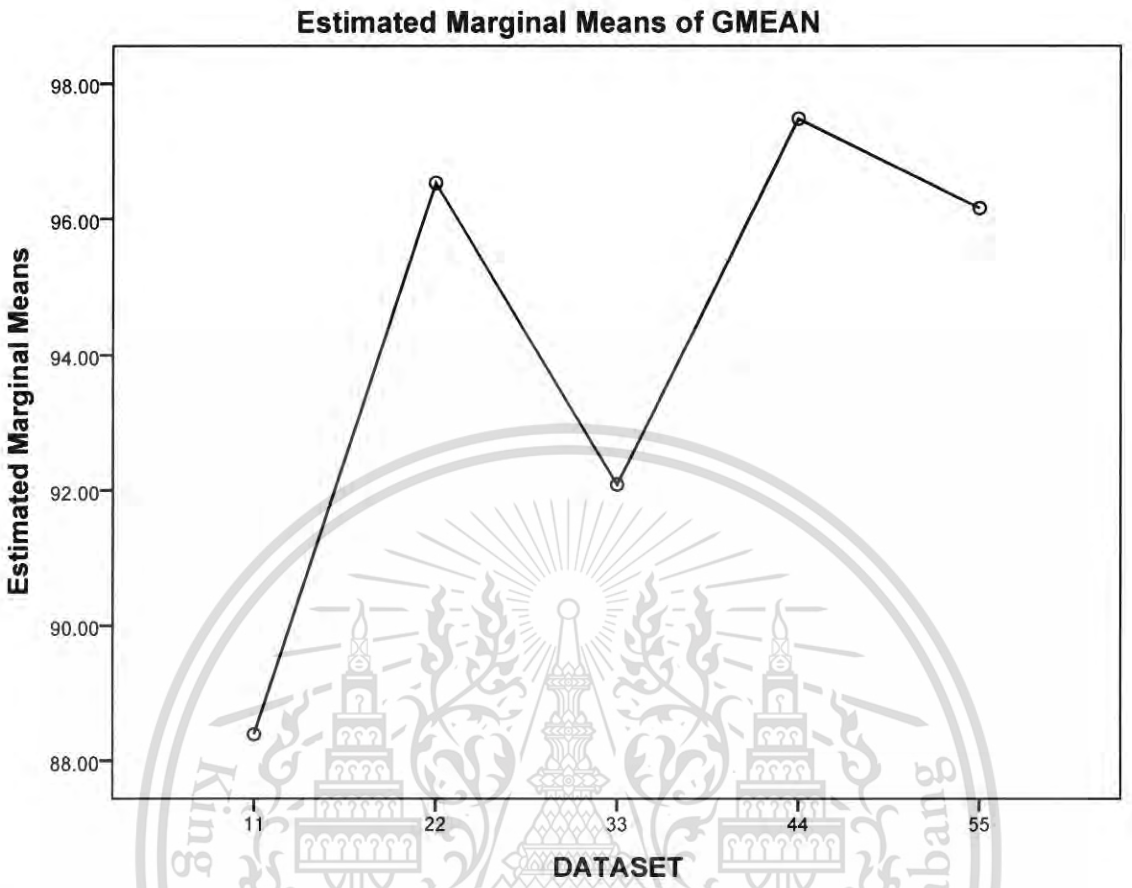
a. Alpha = .01.

Profile Plots

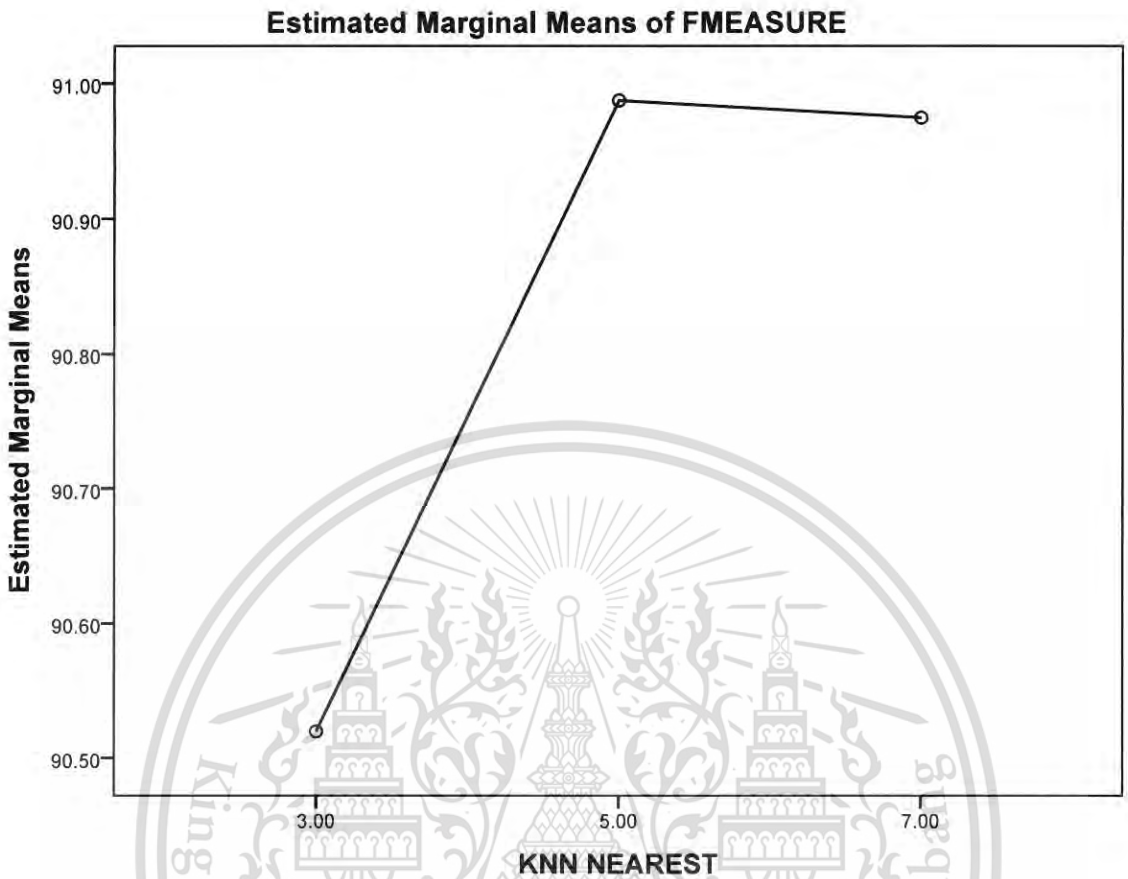
GMEAN







FMEASURE



Author Biography

Name	Mr. Kittipat Savetratanakaree
Date of Birth	5 March 1965.
Address	39/28 Preecha Village-Srinakarin soi 7/3 Bangkaew, Bangplee, Samuthprakarn 10540 , Thailand.
Education	(1987) Bachelor of Science in Applied Statistics. GPA 3.05 Department of Applied Statistics, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand 10520. (1990) Master of Science in Computer Science. GPA 3.9 Department of Computer and Information Science, School of Engineering, University of Mississippi, Oxford Campus, Mississippi 38677, U.S.A.
Scholarship	Master Degree Scholarship in Computer Science, Faculty development program of Bangkok University Year 1989. Ph.D. Scholarship in Computer Science, Faculty development program of Bangkok University Year 2008.

Academic Publication(s)

1.K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum,"Online Game Player's Behavior by using Hidden Markov Model", ICEAST 2013, International Conference on Engineering, Applied Sciences, and Technology 2013., PID:0144,August 21-24,2013, pp.103.

2. K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum, R. Thawonmas, K.T. Chen, "Departure Prediction of Online Game Players",the Journal of Advanced Materials Research Volume 931-932 (2014) pp.1370-1374, Trans Tech Publication(TTP)

3.K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum, R. Thawonmas, K.T. Chen, "Departure Prediction of Online Game Players", the proceedings 5th KKU Engineering Conference (KKU-IENC 2014).

4. K. Savetratanakaree, K. Sookhanaphibarn, S. Intakosum, R. Thawonmas, "Borderline Over-sampling in Feature Space for Learning Algorithms in Imbalanced Data Environments", IAENG International Journal of Computer Science, Volume 43, No. 3, 2016.