AN ACCURATE EVALUATING ALGORITHM TO CONSTRUCT MODIFIED SVM KERNEL METHODS

IRFAN NURHIDAYAT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY IN APPLIED MATHEMATICS
DEPARTMENT OF MATHEMATICS SCHOOL OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2023

KMITL-2023-SC-D-001-061

Thesis Title	An Accurate Evaluating Algorithm to Construct Modified SVM Kernel Methods
Student Name	Irfan Nurhidayat
Student ID	64605130
Degree	Doctor of Philosophy (Applied Mathematics)
Department	Mathematics
Year	2023
Thesis Advisor	Asst. Prof. Dr. Busayamas Pimpunchat

Abstract

This thesis seeks to introduce an MSVM (Modified Support Vector Machine) derived from the CSVM (Classical Support Vector Machine) for the assessment of insurance-related data. The research methodology involves an exploration of CSVM and MSVM using R programming. To gauge the effectiveness of both MSVM and CSVM, performance profiles serve as a tool for assessing and comparing the performance of different algorithms, methods, or models in a systematic and visual meaning. The novel algorithm, referred to as an accurate evaluating algorithm, is developed for constructing the MSVM. The core concept in developing the MSVM is rooted in CSVM. Additionally, an accurate evaluating algorithm is established using tools including Model Performance Evaluation (MPE), Receiver Operating Characteristics (ROCs) Curve, Area Under Curve (AUC), partial AUC (pAUC), smoothing, confidence intervals, and thresholds. These components contribute to the formulation of the MSVM. The study also presents superior performance profiles based on factors like computational time and iteration count for both MSVM and CSVM. By means of performance profiles, quantitative comparisons are drawn between CSVM and MSVM. This accurate evaluating algorithm can be adapted for diverse datasets by leveraging R programming and modifying appropriate kernels as needed.

Keywords : Accurate Evaluating Algorithm, CSVM, Kernels, MSVM.

Acknowledgments

I wish to extend my sincere gratitude to my esteemed advisor, Ajarn Busayamas Pimpunchat, of the Department of Mathematics at King Mongkut's Institute of Technology Ladkrabang. Her invaluable guidance, positive endorsement, and scholarship recommendations have been pivotal in facilitating my journey to Thailand. Ajarn Busayamas' unwavering support is evident through her open-door policy, which provides me a sanctuary for resolving challenges and seeking insights related to my research and scholarly writing. Her judicious guidance steers me toward the correct course while respecting the autonomy of my work. I am profoundly indebted to her for her astute observations on this thesis and her unwavering mentorship throughout my doctoral program at King Mongkut's Institute of Technology Ladkrabang.

I would like to express my sincere gratitude to the distinguished experts who played a pivotal role in the validation survey of this research thesis. I extend my profound appreciation to Ajarn Wannapong Triampo from Mahidol University, as well as the dedicated committee members from King Mongkut's Institute of Technology Ladkrabang: Ajarn Kanchana Kumnungkit, Ajarn Atid Kangtunyakarn, and Ajarn Kanognudge Wuttanachamsri. Their enthusiastic participation and insightful contributions are instrumental in the successful execution of the validation survey. Additionally, I wish to acknowledge Ajarn Sukrawan Mavecha, the Head of the Department of Mathematics, for fostering an enabling academic environment during my academic tenure.

In addition, I hereby extend my sincerest gratitude to the Dean, Vice-Dean, and the entire staff of the School of Science for their exceptional kindness and invaluable support of administrative processes. Their assistance, characterized by kindness, warmth, and professionalism, has been instrumental in facilitating a conducive learning and research environment.

I am deeply grateful to my parents, Indonesian colleagues at King Mongkut's Institute of Technology Ladkrabang, and research collaborators for their unwavering encouragement and steadfast support during every phase of my academic journey, from the initiation of research to the culmination of this thesis. Their unwavering faith in my capabilities and consistent motivation has been instrumental in achieving this milestone.

On a final note, It is with great honor that I dedicate my Ph.D. degree in Applied Mathematics to my beloved parents, Drs. H. Moh. Hidayat, M.Pd., and Hj. Dedeh Nuridah. Their steadfast love, genuine affection, and unwavering prayers have laid the unshakeable groundwork upon which my struggles have been fruitful. The depth of my lovely stands as a testament to the profound bond I share with them forever. Thank you very much.

Table of Contents

List of Tables

Table **Page**

List of Figures

Figure **Page**

Abbreviations

Throughout this thesis, the following abbreviations are adopted.

Words	Meaning
MPE	Model Performance Evaluation
ROCs	Reciever Operating Characteristics
AUC	Area Under Curve
pAUC	Partial Area Under the Curve
SVM	Support Vector Machine
CSVM	Classical Support Vector Machine
MSVM	Modified Support Vector Machine
RBF	Radial Basis Function

Chapter 1

Introduction

This chapter has five sections consisting: research motivation, study objectives, scopes of the study, benefits, and leading organization of the thesis.

1.1 Research Motivation

Data analysis plays a crucial role across various business sectors in today's landscape [1]. The insurance industry has been slow to embrace cutting-edge technologies and applications, often viewed as resistant to change due to outdated methods and constraints [1, 2]. Digital insurance providers have introduced multiple perspectives, encompassing insights, governance, data sources, and cultural aspects. Insights involve artificial intelligence, data science, and machine learning; Governance incorporates elements of the blockchain model and management explainability; Data sources span virtualization, the Internet of Things (IoT), enterprise resource planning (ERP), and customer relationship management (CRM). Data culture encompasses citizen data, data scientists, and data literacy. Digital insurance data represents a platform enabling users to engage with their insurance data from a digital standpoint. Despite its potential, digital insurance data analysis often encounters errors. This thesis focuses on the viewpoint of digital insurers' insights, particularly in relation to machine learning.

The digital insurance market has undergone a significant transformation in recent years, as carriers recognize the imperative to offer speed, flexibility, accessibility, and user-friendly experiences unique to digital insurance [2]. This entails delivering an array of services both externally for clients and internally, entirely online. The maturation of digital insurance innovation is particularly evident in client-facing operations. Insurers are investing in refining their processes and enhancing the customer experience on digital platforms, recognizing that these platforms are vital for their growth and success. Machine learning emerges as a pivotal tool in the realm of digital insurance [3]. As insurers shift towards streamlined service operations and customer-centric interactions, machine learning serves as an enabler of this transition, serving as a fundamental tool for data collection and analysis for insurers.

Machine learning offers the potential to automate previously time-consuming tasks within the insurance operations. An illustrative instance of this is the enhancement of claims processing, which can be significantly improved by leveraging preprogrammed data, algorithms, and customer-provided data to automatically assist clients during critical moments [4]. Insurance enterprises have been navigated through multiple legislative changes, intricate distribution systems, financial crises, and various

other influencing factors. However, the advent of new technologies, the impact of social media, increasingly discerning and empowered customers, ongoing digitization, heightened competition, regulatory shifts, and emergent crises have compounded challenges. Concurrently, the widespread adoption of mobile devices and the simultaneous use of multiple channels have been ongoing trends, resulting in substantial implications that will continue to test the prevailing status quo in the years ahead.

As mentioned earlier, the value of data lies in its correct processing. Effective data processing carries numerous advantages, particularly for organizational management. Decision-making becomes more objective when grounded in field data, eliminating subjectivity. Successful data processing necessitates technological intervention, which can be achieved through learning and adaptation within machine learning frameworks. A viable method for managing company data involves organizing raw or refined data in substantial repositories with appropriate naming conventions. This approach streamlines data retrieval and facilitates the transformation of untidy data into actionable strategies for the company. Data can serve as a compelling motivation for consumers to engage with a company's products or services. A company's ability to process data efficiently signifies its maturity. Conversely, a company's inability to demonstrate proficient data processing raises concerns when considering their offerings, especially in the context of purchasing health insurance. It becomes imperative to ensure the insurance company's resilience, which can often be evaluated through existing data, such as financial reports.

However, obtaining data from consumers or the field is a time-intensive endeavor. Gathering comprehensive data takes considerable effort, and the incoming data is often raw and unstructured. Therefore, the application of data cleansing techniques becomes essential to identify and rectify corrupt or inaccurate databases. Proper data processing expedites the retrieval of essential information, thanks to meticulous categorization [4]. This proves particularly valuable when data analysts require access to data stored in extensive databases.

Support Vector Machine (SVM) is a widely used method in supervised learning, commonly applied for tasks like classification (Support Vector Classification - SVC) and regression (Support Vector Regression - SVR) [5]. It serves as a framework for creating supervised machine learning models based on classification algorithms. Among various classification techniques, SVM stands out with a well-established and comprehensible mathematical foundation. It is proficient in addressing both linear and non-linear classification and regression problems. The Classical SVM (CSVM) is a method developed without integrating a precise evaluation algorithm during the simulation process. On the other hand, the Modified SVM (MSVM) involves the incorporation of an accurate evaluating algorithm before conducting simulations [32].

In order to keep pace with the rapidly evolving technological landscape, the

MSVM method approach introduces techniques for validating a range of data sets and simulations. The utilization of insurance data in this context is of particular interest, as the development of Insurance Technology (InsurTech) lags behind other Financial Technology (FinTech) services in terms of business practices [4].

In a study conducted by Kabran and Ünlü [5], SVM was employed to categorize data for stock market analysis. The researchers used a two-step machine learning technique, including real-time bubble detection and SVM, to forecast bubbles in the S&P 500 [5]. The proposed methodology has the potential for further exploration in data forecasting, kernel analysis, and application to complex financial systems [4, 5]. They used the SVM algorithm to forecast datasets in their study. This thesis will test datasets based on MSVM to predict and classify insurance phenomena.

This research contributes to enhancing our understanding of SVM's applicability and significance, particularly in the context of insurance phenomena and the utilization of advanced data analysis techniques.

1.2 Objectives of the Study

The main objectives of this thesis are as follows:

1. We develop MSVM by employing some statistical tools calls the evaluating algorithm.
2. We introduce MSVM on data classification with linear, polynomial, and RBF kernels.
3. We estimate that MSVM is a developed concept from CSVM over three kernels inside by evaluating the root-mean-square error (RMSE) and density approaches.
4. We apply for MSVM to classify the sum insured datasets in recommending insurance premiums to pay more (0) or have to pay less (1).

1.3 Scope of the Study

The scope of this thesis is such that the MSVM method with the accurate evaluating algorithm features three kernels: linear, polynomial, and radial basis functions. These kernels also find prominence in simulations and visualizations through their implementation using R programming. Insurance datasets from the Office of the Insurance Commission (OIC) in Thailand were limited to the years 2011, 2012, and 2014 and capable computerized at their official site. There is no dataset in 2013 in this study based on information from the OIC site.

1.4 Benefits of the Study

The benefits of the study are as follows:

1. MSVM is expected to analyze data more accurately than CSVM in this work.
2. Various large-sized data are used, following a variety of suitable kernels for each data.
3. This work points out MSVM approaches for faster data analysis than others.
4. MSVM is a new approach for predicting and classifying datasets.

1.5 Organizational Thesis

The thesis reveals chapters, i.e., Chapter 1 discusses research motivation, objective, scope, benefits, and the whole organization of this thesis. Chapter 2 focuses on literature reviews which will give the small theories of the accurate evaluating algorithm with a mathematical work of the CSVM. Chapter 3 reveals the research methodology with most concepts for research questions, methods, and steps. Chapter 4 delves into the main discussions of this thesis, divided into three parts:

1. CSVM and MSVM are underlined with a polynomial kernel and simulated performance profiles of these methods.
2. MSVM is stressed into three main kernels and ended by numerical comparisons via the root-mean-square error (RMSE).
3. MSVM applications are to classify the sum insured in Thailand.

Section 5 concludes all of this thesis work with a future work.

Chapter 2

Theory and Literature Reviews

2.1 Model Performance Evaluation

Predictive models help project the future growth of enterprises since they predict outcomes using data mining and probability, with each model containing several predictors or variables. As a result, a simulation approach may be developed by gathering data on influential factors. Depending on the type of company, a predictive model is able to tackle two types of problems: classification and regression. These two areas serve as the starting point for a data science team to select the appropriate measurements and create a solid functioning model. It is critical to assess appropriate prediction models since different datasets will be used for the same predictive model. Database tuning is required to be conveniently organized and accessed. Developing a model with the performance evaluation (MPE) through evaluating the model is one of the highly prominent things. The algorithm for MPE testing in R is as follows.

Algorithm 1 Plot E with MPE

Require: Install ROCR

Ensure: plot(E)

1: predict(M,b,t) to get p
2: prediction(p,b$Lv) to get p
3: We have the performance(p) is E.

Model Performance Evaluation is a fundamental process in machine learning that involves assessing the effectiveness and quality of a trained model's predictions. This process is crucial for understanding how well a model generalizes to new, unseen data and for making informed decisions about model deployment and improvement.

Here is an in-depth explanation of the Model Performance Evaluation algorithm:

1. Dataset Splitting: Before evaluating model performance, the dataset is typically split into two or more subsets: a training set and one or more evaluation sets. The training set is used to train the model, while the evaluation sets are used to assess the model's performance. Common splits include a training-validation-test split or cross-validation.

2. Metric Selection: Choose appropriate evaluation metrics based on the problem type. Classification problems might use metrics like accuracy, precision, recall,

F1-score, ROC-AUC, etc., while regression problems may use metrics like mean squared error (MSE), mean absolute error (MAE), or R-squared.

3. Model Training: Train the chosen machine learning model using the training dataset. The model learns patterns and relationships in the data that it can later use for making predictions.

4. Prediction Generation: Apply the trained model to the evaluation dataset to generate predictions. For classification tasks, the model assigns class labels, while for regression tasks, it predicts numerical values.

5. Comparison with Ground Truth: Compare the model's predictions with the ground truth (actual) values from the evaluation dataset. This involves calculating the chosen evaluation metrics using the predicted and actual values.

6. Visualization and Analysis: Visualize the evaluation results to gain insights into the model's performance. This could involve creating confusion matrices, ROC curves, precision-recall curves, or scatter plots of predicted vs. actual values. These visualizations help in understanding how the model behaves under different circumstances.

7. Model Tuning and Improvement: Based on the evaluation results, make informed decisions about model tuning and hyperparameter adjustments. This might involve iterating through different model architectures, feature selections, or hyperparameter settings to improve performance.

8. Performance Robustness: Assess the model's robustness by evaluating its performance on multiple evaluation sets, such as using cross-validation or a time-based split. This helps ensure that the model's performance is consistent and not heavily influenced by specific data subsets.

9. Interpreting Results: Interpret the evaluation metrics and analysis to understand the strengths and weaknesses of the model. This step involves determining where the model excels and where it might struggle, which can guide further improvements or inform decision-making.

10. Reporting and Communication: Summarize the model's performance and findings in a clear and concise manner. This includes presenting the evaluation metrics, visualizations, and any insights gained from the analysis. Effective communication is important for collaboration and decision-making among stakeholders.

In essence, the Model Performance Evaluation algorithm is a systematic process of training, testing, and analyzing machine learning models to determine their effectiveness in making predictions. It enables data scientists and practitioners to gauge

the model's ability to generalize to new data and make informed choices about its deployment or refinement.

2.2 Reciever Operating Characteristics Curve

Receiver Operating Characteristics (ROCs) are employed to establish thresholds for a classifier. These characteristics typically gauge the accuracy of positive classifications. ROC curves are instrumental in assessing models using various probability thresholds. While initially developed for signal detection, this technology has found applications in diverse domains such as medicine, radiology, natural disaster prediction, and machine learning.

A discrete classifier yields the predicted class, representing a single point on the ROC curve. Conversely, probabilistic classifiers can generate curves by varying the score threshold. Thresholds for classification criteria are often visualized to evaluate the performance of both ROC and AUC (Area Under Curve) [6, 7].

The ROC metric offers a visual representation of a classification model's performance, based on its accuracy and rate of misclassification. Below is an explanation of the ROC algorithm:

Algorithm 2 Output of ROCs is rocplot

Require: Install ROCR

Ensure: Output is rocplot

```
1:  while f(p,t) to rocplot do
2:  prediction(p,t) to predob
3:  performance(predob) to perf
4:  perf to plot
5:  end while
6:  svm(Lv, d, k, d, c) to svmfit opt
7:  svm(Lv, d, k, d, c) to svmfit flex
8:  attributes(p(svmfit opt,b)) to fit opt train
9:  (fit opt train, b$Lv, m) to rocplot                ▷ training data
10: attributes(p(svmfit flex, b)) to fit flex train
11: (fit flex train, b$Lv) to rocplot
12: attributes(p(svmfit opt, b)) to fit opt test
13: (fit opt test, b$Lv, m) to rocplot                 ▷ test data
14: attributes(p(svmfit flex, b)) to fit flex test
15: (fit flex test, b$Lv) to rocplot
```

The ROC Curve is a graphical representation commonly used in the field of machine learning and statistics to evaluate and visualize the performance of binary

classification models.

Here is a detailed explanation of the ROC Curve algorithm:

1. Binary Classification Problem: The ROC Curve is typically used for binary classification problems, where the goal is to categorize instances into one of two classes, often referred to as ”positive” and ”negative”.

2. Threshold Variation: In a binary classification model, predictions are often made using a decision threshold. Observations with predicted probabilities above this threshold are classified as the positive class, and those below are classified as the negative class. The ROC Curve is constructed by varying this threshold across a range of values from 0 to 1.

3. True Positive Rate (Sensitivity): For each threshold value, calculate the True Positive Rate (TPR), also known as sensitivity or recall. TPR measures the proportion of actual positive instances that are correctly classified as positive by the model.

$$TPR = TP/(TP + FN)$$

where TP is the number of true positives (correctly predicted positive instances) and FN is the number of false negatives (positive instances incorrectly predicted as negative).

4. False Positive Rate (Fallout): Calculate the False Positive Rate (FPR), also known as fallout or the probability of false alarm. FPR measures the proportion of actual negative instances that are incorrectly classified as positive by the model.

$$FPR = FP/(FP + TN)$$

where FP is the number of false positives (negative instances incorrectly predicted as positive) and TN is the number of true negatives (correctly predicted negative instances).

5. ROC Curve Plotting: Plot the calculated TPR values on the y-axis and the corresponding FPR values on the x-axis. Each point on the ROC Curve represents a specific threshold value.

6. Diagonal Line: The diagonal line from the bottom left corner to the top right corner of the ROC space represents random guessing, where the model’s performance is no better than chance. Points above the diagonal line represent better-than-random performance, while points below the line represent worse performance.

7. AUC-ROC Score: The Area Under the ROC Curve (AUC-ROC) is a scalar value that quantifies the overall performance of the classification model. AUC-ROC ranges

from 0 to 1, where a value of 0.5 indicates random guessing, and higher values indicate better model performance. A perfect classifier would have an AUC-ROC of 1.

8. Model Comparison: The ROC Curve provides a visual and quantitative way to compare the performance of different classification models. Models with higher AUC-ROC values generally have better discrimination power between positive and negative classes.

9. Selecting a Threshold: The choice of threshold depends on the trade-off between sensitivity and specificity (1 - FPR). A lower threshold increases sensitivity but may also increase the false positive rate, while a higher threshold increases specificity at the cost of sensitivity.

10. Limitations and Extensions: The ROC Curve assumes that the class distribution remains balanced across different thresholds. For imbalanced datasets, Precision-Recall curves might be more appropriate. Extensions like the Multi-Class ROC Curve and Confidence Intervals for ROC Curves are used for multi-class classification and uncertainty estimation, respectively.

In summary, the Receiver Operating Characteristic (ROC) Curve algorithm is a powerful tool for evaluating and comparing binary classification models based on their trade-off between true positive rate (sensitivity) and false positive rate (fallout). The ROC Curve provides insights into the model's discriminatory power and allows for informed decision-making regarding threshold selection and model performance.

2.3 Area under the ROC Curve

AUC-ROC is a valuable metric employed to evaluate the performance of classification models. It quantifies the model's capacity to distinguish between different classes. A key principle guiding the interpretation of AUC-ROC is that a larger AUC value corresponds to a more accurate model. AUC-ROC curves are commonly used to assess classification performance across various threshold settings.

In order to provide an in-depth understanding of the terms uses in AUC and the ROC curve, as well as the mathematical formulas associated with them, we recommend to referring the sources cited: [8, 9, 10].

The AUC algorithm is as the following:

Algorithm 3 AUC for plot(roc)

Require: Install ROCR

Ensure: plot(roc)

1: performance(p) to roc
2: performance(p) to auc
3: unlist(s(auc)) to auc
4: round(auc) to auc

The target is to get a plot of the AUC with installing ROCR:

1. We calculate the performance of p to get ROC.
2. We calculate the performance of p to get AUC.
3. We calculate unlist(s(auc)) to get the AUC.
4. We calculate round(auc) to get the AUC.
5. Then, the output from the algorithm is a plot(roc).

The AUC-ROC is a widely used metric in machine learning and statistics to assess the performance of binary classification models. It quantifies the ability of a model to distinguish between positive and negative instances across various threshold settings.

Here is an in-depth explanation of the AUC-ROC algorithm:

1. Binary Classification Context: The AUC-ROC is specifically applicable to binary classification problems, where the goal is to categorize instances into one of two classes: "positive" and "negative."
2. Threshold Variation: In a binary classification model, predictions are made based on a decision threshold. By varying this threshold across a range of values from 0 to 1, the model can be evaluated under different levels of sensitivity and specificity.
3. True Positive Rate (Sensitivity): For each threshold value, calculate the True Positive Rate (TPR), also known as sensitivity or recall. TPR measures the proportion of actual positive instances that are correctly classified as positive by the model.

$$TPR = TP/(TP + FN)$$

where TP is the number of true positives (correctly predicted positive instances) and FN is the number of false negatives (positive instances incorrectly predicted as negative).

4. False Positive Rate (Fallout): Calculate the False Positive Rate (FPR), also known as fallout or the probability of false alarm. FPR measures the proportion of actual negative instances that are incorrectly classified as positive by the model.

$$FPR = FP/(FP + TN)$$

where FP is the number of false positives (negative instances incorrectly predicted as positive) and TN is the number of true negatives (correctly predicted negative instances).

5. ROC Curve Plotting: Plot the calculated TPR values on the y-axis against the corresponding FPR values on the x-axis. This results in the Receiver Operating Characteristic (ROC) Curve. The ROC Curve visually represents the trade-off between sensitivity and specificity as the threshold varies.

6. AUC-ROC Calculation: The Area Under the ROC Curve (AUC-ROC) is the integral (area) under the ROC Curve. It quantifies the model's ability to discriminate between positive and negative instances across all possible threshold settings. A perfect classifier would have an AUC-ROC of 1, while random guessing would result in an AUC-ROC of 0.5.

7. Interpreting AUC-ROC: A higher AUC-ROC value indicates better overall model performance in distinguishing between the two classes. It implies that the model is better at correctly ranking positive instances above negative instances across various threshold choices.

8. Model Comparison: The AUC-ROC provides a single scalar value that allows for straightforward comparison between different classification models. Models with higher AUC-ROC values are generally better at discriminating between positive and negative instances.

9. Limitations and Extensions: The AUC-ROC assumes that the class distribution remains balanced across different threshold settings. It might not be the ideal metric for imbalanced datasets. In such cases, precision-recall curves and the area under the precision-recall curve (AUC-PR) are often used.

10. Practical Application: AUC-ROC is commonly used for evaluating the performance of machine learning models, especially in medical diagnostics, fraud detection, and other domains where correctly identifying positive instances is crucial.

In summary, AUC serves as a crucial metric for evaluating the performance of classification models, particularly in terms of their ability to distinguish between classes. The AUC-ROC curve provides valuable insights into the model's accuracy

across different threshold settings, aiding in the interpretation and comparison of classification results. The Area Under the ROC Curve (AUC-ROC) algorithm is a powerful tool for quantifying and comparing the performance of binary classification models by measuring their ability to discriminate between positive and negative instances across varying threshold choices. It provides a comprehensive evaluation of a model's ability to rank instances and is widely used for model selection and comparison.

2.4 Partial AUC

The Partial Area Under the Curve (pAUC) serves as a method for analyzing data, its application, and addressing the question of why a partial ROC will be necessary for data validation [11, 12]. The partial AUC is a valuable tool for data analysis and validation, offering insights into specific aspects of a classifier's performance. Its application contributes to a comprehensive understanding of the classifier's behavior, particularly when evaluating performance across various thresholds. For a detailed understanding of the need for partial ROC and its significance, readers may refer to various related sources.

The pAUC algorithm is as the following:

Algorithm 4 Partial AUC for plot.roc

Require: Install ROCR

Ensure: plot.roc

1: (bLv, by, percent, partial.auc, partial.auc.correct, print.auc) to plot.roc
2: if print.auc then
3: corrected pauc to print.auc.pattern
4: polygon to auc.polygon
5: max.polygon to max.auc.polygon
6: else if sensitivity then
7: SE to partial.auc.focus
8: end if

The AUC rate is frequently utilized in meta-analyses, particularly in situations where each component of a study estimates the test's sensitivity and specificity [9]. This approach recognizes the value of partial AUC (pAUC), which focuses on specific regions of the ROC space where observed data or clinically significant values of test sensitivity and specificity exist. In cases where a partial AUC is employed, it takes into consideration those segments of the ROC curve that are most relevant to the study's objectives.

In an ideal scenario where the ROC curve is perfect, the pAUC result tends to be 1, signifying optimal discrimination. On the other hand, a non-discriminant ROC

curve typically yields a pAUC value of 0.5, indicating chance-level performance [8].

The AUC rate plays a crucial role in meta-analyses by quantifying the balance between sensitivity and specificity in various study components. The use of partial AUC offers a refined perspective, focusing on specific regions of the ROC curve to provide a more relevant evaluation of test performance. The resulting pAUC values offer valuable insights into the discriminative capabilities of the tests being assessed.

The Partial AUC is a specific metric used to evaluate the performance of binary classification models by considering a restricted portion of the Receiver Operating Characteristic (ROC) curve.

Here is a comprehensive explanation of the Partial AUC algorithm:

1. Background: The Receiver Operating Characteristic (ROC) curve is a graphical representation of a binary classification model's performance across various threshold settings. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) as the threshold varies.

2. Partial AUC Definition: The Partial Area Under the Curve (Partial AUC) focuses on a specific segment of the ROC curve, usually between two predefined FPR values. It measures the area under this segment of the curve, representing the model's performance within that specific range of FPR.

3. Application: Partial AUC is particularly useful when certain regions of the ROC curve are more important or relevant for a given problem. It might be used when false positives are costlier in certain scenarios, or when emphasizing performance at specific operating points.

4. Calculation of Partial AUC: To calculate the Partial AUC, follow these steps:

    - Choose a range of interest for FPR (e.g., FPR from a lower bound to an upper bound).
    - Calculate the area under the ROC curve within this specified FPR range. This can be done using integration methods or numerical approximation techniques.

5. Interpretation of Partial AUC: Similar to the AUC-ROC, the Partial AUC ranges from 0 to 1, where a higher value indicates better model performance within the chosen FPR range. A Partial AUC of 1 represents perfect performance in that specific FPR interval.

6. Applications and Use Cases: Partial AUC is especially relevant in situations where certain false positive rates have more significant implications than others. For instance, in medical diagnosis, a test with a lower false positive rate might be more desirable even if sensitivity is sacrificed to some extent.

7. Performance Comparison: Partial AUC allows for comparing different models' performance within a specific region of the ROC curve, providing insights into how well models perform at certain operating points.

8. Implementations: Various software libraries and tools offer functions to calculate Partial AUC directly from ROC curve data. These libraries often provide options to customize the FPR range of interest.

9. Limitations: Partial AUC can provide valuable insights, but it is sensitive to the chosen FPR range. Different ranges might yield different results and interpretations.

10. Trade-offs: Emphasizing performance in a specific region of the ROC curve might lead to a trade-off in other areas. Models optimized for one region might perform less well in other regions.

In summary, the Partial Area Under the Curve (Partial AUC) algorithm is a specialized metric used to evaluate binary classification models by focusing on a specific segment of the ROC curve. It offers insights into model performance within a particular false positive rate range, making it a useful tool for situations where certain operating points have specific importance or implications.

2.5 Smoothing

Smoothing serves as a statistical technique aimed at eliminating unnecessary data from datasets, thereby enhancing the visibility of underlying patterns. This method proves effective in reducing statistical errors within insurance data. It is also referred to as curve fitting and low-pass filtering. The concept of smoothing represents a pivotal approach extensively used in data processing. Its primary objective is to unveil trends, particularly when the exact nature of the trend remains uncertain.

The significance of smoothing techniques extends to the realm of machine learning, where they are valuable for addressing conditional expectations and probabilities that may exhibit complex shapes necessitating estimation in the presence of uncertainty [13]. These techniques provide a means to effectively handle challenging data patterns in the context of uncertainty.

Here is an explanation of the smoothing algorithm:

Algorithm 5 Smoothing for rocobj(plot.roc)

Require: Install ROCR

Ensure: rocobj(plot.roc)

1: smooth(rocobj) to lines ▷ binormal
2: smooth(rocobj) to lines ▷ density
3: smooth(rocobj) to lines ▷ fitdistr
4: smooth(rocobj) to lines ▷ lognormal

Smoothing is a powerful statistical strategy employed to enhance pattern visibility by eliminating unnecessary data from datasets. This approach is crucial for data processing, aiding in the discovery of trends, even in situations where the exact form of the trend is uncertain. Its relevance extends to machine learning, where smoothing techniques are harnessed to estimate complex shapes in the presence of uncertainty.

Smoothing is a technique used in various fields, including signal processing, statistics, and machine learning, to reduce noise, fluctuations, or irregularities in data, resulting in a clearer and more understandable representation of underlying patterns or trends.

Here is a comprehensive explanation of the Smoothing algorithm:

1. Introduction to Smoothing: Smoothing is employed to transform raw, noisy data into a smoother version that highlights important features while minimizing random fluctuations. This is especially useful when dealing with data that contains random variations or measurement errors.

2. Moving Average Smoothing: One of the most straightforward smoothing techniques is the Moving Average. It involves calculating the average value of a set of adjacent data points within a defined window or interval. The resulting smoothed value replaces the original value at the center of the window. Moving average smoothing helps reduce high-frequency noise.

3. Exponential Smoothing: Exponential Smoothing is a popular technique that assigns exponentially decreasing weights to older data points while emphasizing recent data. This approach gives more importance to recent observations and is suitable for data with trends and seasonality.

4. Kernel Smoothing: Kernel smoothing, often used in non-parametric statistics, involves placing a kernel (a weighting function) over each data point and computing a weighted average of neighboring points. The choice of kernel and bandwidth determines the smoothness level.

5. Savitzky-Golay Smoothing: This method involves fitting a polynomial to local

subsets of the data and using the polynomial to estimate smoothed values. It is particularly useful for preserving important features like peaks and valleys.

6. Lowess (Locally Weighted Scatterplot Smoothing): Lowess is a non-parametric regression technique that fits a locally weighted polynomial regression to the data. It adapts to local trends and is robust against outliers.

7. Gaussian Smoothing: Gaussian smoothing convolves the data with a Gaussian kernel, effectively applying a weighted moving average with a Gaussian-shaped window. This method is often used in image processing to blur or smooth images.

8. Applications of Smoothing: Smoothing is applied in various domains. In time series analysis, it helps identify trends and seasonality. In image processing, it reduces noise and enhances edges. In signal processing, it improves the quality of signals. In machine learning, smoothing can help preprocess data before applying learning algorithms.

9. Trade-offs: While smoothing can enhance data clarity, it may also lead to loss of fine details or important fluctuations. The choice of smoothing technique and parameters should be guided by the characteristics of the data and the intended analysis.

10. Optimizing Smoothing: In many cases, the level of smoothing is determined empirically or through optimization methods to strike a balance between reducing noise and preserving important features.

11. Interpolation vs. Smoothing: Smoothing is distinct from interpolation, which aims to estimate values between known data points. Smoothing focuses on reducing noise and revealing trends.

In summary, the smoothing algorithm is a valuable tool for improving data quality and revealing underlying patterns by reducing noise and fluctuations. Various smoothing techniques offer flexibility in adapting to different data characteristics and analytical goals, making them indispensable tools in data analysis, signal processing, and more.

2.6 Confidence Intervals

The calculation of confidence intervals involves various factors, and the sample size is one crucial component of the equation. Sensitivities, also known as the True Positive Rate (TPR), denote the percentage of individuals correctly identified as having positive states, when the predicted condition is also positive. On the other hand, specificity, referred to as the False Positive Rate (FPR), represents the proportion of

individuals identified as having an adverse condition, when the actual condition is approachable. The following is an explanation of the algorithm:

Algorithm 6 Confidence Intervals

Require: Install pROC

Ensure: rocobj(plot.roc)

1: b$Lv to plot.roc
2: b$y to plot.roc
3: ci.sp(rocobj, sensitivities) to plot

Confidence intervals are computed by taking into account various factors, with the sample size being a significant consideration. Sensitivities (TPR) and specificities (FPR) are fundamental metrics used to assess the performance of classifiers in binary classification tasks. These metrics provide insights into the ability of the model to correctly identify positive and negative instances. The algorithm mentions in the paragraph likely elaborates on the specific calculations involved in these metrics and their interpretation.

Confidence Intervals are a fundamental statistical concept used to estimate the range within which a population parameter, such as a mean or proportion, is likely to lie based on a sample of data. They provide a measure of uncertainty and help infer the true characteristics of a population from a limited sample.

Here is an in-depth explanation of the Confidence Intervals algorithm:

1. Introduction to Confidence Intervals: Confidence Intervals (CIs) are a range of values around a sample statistic (e.g., mean, proportion) that is likely to contain the true population parameter. CIs take into account the variability in sample data and provide an estimate of the level of uncertainty associated with the sample estimate.

2. Sample Data Collection: To construct a Confidence Interval, start by collecting a representative sample from the population of interest. The sample should be randomly selected and sufficiently large to ensure valid inference.

3. Select a Confidence Level: Choose a desired confidence level, often denoted as (1 - α), where α is the significance level or the probability of making a Type I error. Commonly used confidence levels are 90%, 95%, and 99%.

4. Sample Statistic Calculation: Calculate the sample statistic of interest, such as the sample mean or proportion.

5. Standard Error Calculation: Calculate the standard error (SE) of the sample statistic. The standard error represents the average amount of variability expected between sample estimates and the true population parameter.

6. Critical Value Calculation: Determine the critical value(s) based on the chosen confidence level and the distribution of the sample statistic. The critical value(s) define the boundaries of the Confidence Interval.

7. Confidence Interval Calculation: Calculate the Confidence Interval by adding and subtracting the product of the standard error and the critical value(s) from the sample statistic.

   Confidence Interval = Sample Statistic ± (Critical Value * Standard Error)

8. Interpretation of Confidence Intervals: The Confidence Interval provides a range within which the true population parameter is likely to lie with the chosen confidence level. For example, a 95% Confidence Interval implies that if we were to collect multiple samples and construct intervals in the same way, about 95% of those intervals would contain the true population parameter.

9. Narrowing the Interval: Increasing the sample size generally results in a narrower Confidence Interval since larger samples provide more precise estimates of the population parameter.

10. Applications of Confidence Intervals: Confidence Intervals are used in a wide range of fields, including social sciences, economics, medicine, and more. They are essential for making informed decisions based on sample data and assessing the reliability of statistical findings.

11. Limitations and Considerations: Confidence Intervals assume that the sample is representative of the population and that the data follows certain statistical distributions. Additionally, CIs do not tell us anything about individual samples; they are about the long-term behavior of intervals constructed from similar samples.

In summary, Confidence Intervals are a statistical tool used to estimate the range within which a population parameter is likely to fall based on sample data. They provide a measure of the uncertainty associated with the estimate and play a critical role in making reliable inferences from sample statistics to population parameters.

2.7 Confidence Intervals of a Threshold

The confidence threshold is typically represented as a percentage and is the reciprocal of the significance threshold. For instance, a significance threshold of 0.05 corresponds to a 95% confidence level. This threshold indicates the level of certainty required to exclude all values under the null hypothesis. The specific probability chosen must align with the circumstances of the test being conducted. The threshold is utilized to calculate the necessary sample size for conducting a powerful test at that threshold, considering the specified minimal impact of interest [16, 17].

The confidence threshold reverses the elements and is stated in terms of percentages. In practice, the confidence threshold is often set at 95% when determining the threshold for a particular test.

Here is an explanation of the algorithm:

Algorithm 7 Confidence Intervals of a Threshold for plot.roc

Require: Install pROC

Ensure: plot.roc

1: b$Lv to plot.roc
2: b$y to plot.roc
3: percent to plot.roc
4: ci to plot.roc
5: of to plot.roc ▷ thresholds
6: thresholds to plot.roc ▷ best thresholds
7: print.thres to plot.roc

The confidence threshold is a fundamental parameter in statistical hypothesis testing, indicating the level of confidence required to draw conclusions from the results. It is expressed as a percentage and is closely related to the significance threshold. The algorithm mentions in the paragraph likely elaborates on the calculations and decisions involved in setting the confidence threshold for a specific test.

Confidence Intervals of a Threshold is a statistical concept used in binary classification to estimate a range of thresholds for making predictions that yields a desired level of confidence in the model's performance.

Here is a comprehensive explanation of the Confidence Intervals of a Threshold algorithm:

1. Context of Binary Classification: In binary classification, a threshold is used to determine the class label assigned to an instance based on the model's predicted probability. For example, if the predicted probability of a positive class is greater than the threshold, the instance is classified as positive; otherwise, it's classified as negative.

2. Importance of Threshold Selection: The choice of threshold can significantly impact the balance between precision and recall, leading to different trade-offs between false positives and false negatives. Selecting an appropriate threshold is crucial for aligning the model's predictions with the specific goals of the problem.

3. Uncertainty in Threshold Selection: The optimal threshold may not be known with certainty due to factors such as limited data or inherent uncertainty in the problem domain.

4. Confidence Intervals for Thresholds: Confidence Intervals of a Threshold provide a range of threshold values within which the ”true” optimal threshold likely lies with a specified level of confidence.

5. Algorithm Steps:

   - Select a Confidence Level: Choose a desired level of confidence, often denoted as (1 - α), where α is the significance level.
   - Sample Thresholds: Randomly sample multiple thresholds from a predefined range or distribution.
   - Model Performance Evaluation: For each sampled threshold, evaluate the model’s performance using appropriate metrics such as accuracy, precision, recall, F1-score, or others, on a holdout dataset or through cross-validation.
   - Confidence Interval Calculation: Calculate the Confidence Interval for the threshold values based on the distribution of the sampled performance metrics. This interval indicates the range of thresholds that are likely to yield similar performance with the chosen confidence level.

6. Interpretation of Confidence Intervals: The Confidence Intervals of a Threshold provide insights into the range of thresholds that are consistent with the desired model performance level at the selected confidence level.

7. Threshold Selection Process: Once you have the Confidence Intervals, you can choose a threshold from within the interval based on the desired trade-offs between precision and recall or other relevant factors.

8. Applications: Confidence Intervals of a Threshold can be particularly useful when the cost of false positives and false negatives varies or when the optimal threshold is uncertain due to limited data.

9. Implementation and Software: This algorithm can be implemented using programming languages like Python or statistical software packages. Libraries for statistical inference and machine learning may offer tools for evaluating and estimating Confidence Intervals of a Threshold.

10. Considerations and Trade-offs: While Confidence Intervals of a Threshold provide valuable insights, they are based on assumptions about the distribution of performance metrics and the sampling procedure. It is important to interpret the results in the context of these assumptions.

In summary, Confidence Intervals of a Threshold is a statistical technique used in binary classification to estimate a range of threshold values that are likely to yield similar model performance with a specified level of confidence. This approach helps

address uncertainty in threshold selection and facilitates making informed decisions based on the trade-offs between different performance metrics.

2.8 Support Vector Machine

The SVM model is derived with the main idea from understanding in [18] and other related sources. The main formula of SVM defines $f(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x} + b$ when $\mathbf{w}$ is a parameter for a weight vector and b is a bias. Support vectors are the nearest hyperplane data. The form of the margin is

$$M = \frac{|f(\mathbf{x})|}{\|\mathbf{w}\|}. \tag{2.1}$$

It recalls that the optimal hyperplane is $\mathbf{w} \cdot x + b = 0$. The goal, then, is simply to find the maximum margin M, which is

$$M = \frac{2}{\|\mathbf{w}\|}. \tag{2.2}$$

It has (2.1), which can be generalized and afforded a small error. The equation can be redefined as follows

$$\max(M) = \max_{\mathbf{w},b} \left(\frac{|f(\mathbf{x})|}{\|\mathbf{w}\|} \right) = \min_{\mathbf{w}}(\|\mathbf{w}\|). \tag{2.3}$$

We let the vectors $\mathbf{u}$ and $\mathbf{v}$ be parallel vectors on the straight lines successively

$$\mathbf{w} \cdot \mathbf{x} + b = -1, \ \mathbf{w} \cdot \mathbf{x} + b = 1. \tag{2.4}$$

Then, $\mathbf{v} = \mathbf{u} + \mathbf{q}$ where the vector $\mathbf{q}$ is a straight line perpendicular to the vector $\mathbf{u}$ and $\mathbf{v}$. The edge of the vector $\mathbf{q}$ is M as follow $\mathbf{q} = M \cdot \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \right)$. This means that $\mathbf{q}$ go in the same direction as a vector $\mathbf{w}$. It has a fact that $\mathbf{v}$ and $\mathbf{u}$ are on (2.4), then

$$\mathbf{w} \cdot \mathbf{u} + b = -1, \ \mathbf{w} \cdot \mathbf{v} + b = 1. \tag{2.5}$$

From (2.5), for $\mathbf{v}$, it obtains $\mathbf{w} \cdot (\mathbf{u} + \mathbf{q}) + b = 1 \iff M = \frac{2}{\|\mathbf{w}\|}$. From (2.3), it has $\max(M) = \min_{\mathbf{w}} \left(\frac{1}{2}\|\mathbf{w}\| \right)$ with the inequality condition as $y_i(\mathbf{w}^T \cdot x_i + b) \geq 1, \ [i = 1, 2, ..., N]$. It is referred to as SVM optimization with hard margin and rewritten as

$$\min_{\mathbf{w},b} \left(\frac{1}{2}\|\mathbf{w}\|^2 \right), \ \text{s.t. } y_i(\mathbf{w}^T \cdot x_i + b) \geq 1, \tag{2.6}$$

where (2.6) is a generalization of (2.2), also known as the edge of (2.2). There are points at which are still trying to minimize $\|\mathbf{w}\|$. The goal is to keep the edges as large as possible and the slack variables as small as possible. To avoid the strictness of the edges, it presents the objective SVM optimization problem with soft edges

$$\min_{\mathbf{w},b,\xi} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i \right), \ \text{s.t. } y_i(\mathbf{x}^T x_i + b) \geq 1 - \xi_i, \ \xi_i \geq 0, \ [i = 1, ..., N]. \tag{2.7}$$

It write the constraints from (2.7) as $-y_i(\mathbf{w}^T \cdot x_i + b) + 1 - \xi_i \leq 0,\ -\xi_i \leq 0$. It has the Lagrangian as

$$L(\mathbf{w}, b, \xi, \alpha, \lambda; C) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(\mathbf{w}^T \cdot x_i + b) + \xi_i - 1] \tag{2.8}$$

with $\mathbf{w}$ as a primary variable and α, λ for dual variables or the Lagrange multipliers. The Lagrange multiplier solutions are the solution of the original optimization and all of them should satisfy the Karush-Khun-Tucker (KKT) optimality conditions. Through the KKT it obtains

$$\begin{aligned}
&\nabla_w L(\mathbf{w}, b, \xi, \alpha, \lambda; C) = 0 \Longleftrightarrow \mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i x_i = 0 \text{ and } \sum_{i=1}^{N}\alpha_i y_i = 0,\\
&\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \lambda; C)}{\partial b} = \sum_{i=1}^{N}\alpha_i y_i = 0,\\
&\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \lambda; C)}{\partial \xi_i} = C - \lambda_i - \alpha_i = 0.
\end{aligned} \tag{2.9}$$

It has a fact from the dual admissibility condition, i.e. $\alpha \geq 0,\ \lambda \geq 0 \Longleftrightarrow C \geq \alpha$. Consequently, $0 \leq \alpha_i \leq C,\ [i = 1, ..., N]$. From (2.9), constraints for the soft margin SVM optimization problem are

$$\sum_{i=1}^{N}\alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C,\ [i = 1, ..., N]. \tag{2.10}$$

We now consider the last term of the Lagrangian sum from (2.8), which is

$$\sum_{i=1}^{N}\alpha_i[y_i(\mathbf{w}^T \cdot x_i + b) + \xi_i - 1]$$

and substitute $\mathbf{w}$ form of (2.9) such as

$$\begin{aligned}
&\sum_{i=1}^{N}\alpha_i\left[y_i\left(\left(\sum_{j=1}^{N}\alpha_j t_j x_j\right)^T \cdot x_i + b\right) + \xi_i - 1\right]\\
&= \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i t_j x_i^T \cdot x_j + b\sum_{i=1}^{N}\alpha_i y_i + \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}\alpha_i
\end{aligned} \tag{2.11}$$

with letting $R = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i t_j x_i^T \cdot x_j$. Then, it applies for (2.11) into (2.8) so that

$$\begin{aligned}
&L(\mathbf{w}, b, \xi, \alpha, \lambda; C)\\
&= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\xi_i - R - b\sum_{i=1}^{N}\alpha_i y_i - \sum_{i=1}^{N}\alpha_i\xi_i + \sum_{i=1}^{N}\alpha_i.
\end{aligned} \tag{2.12}$$

It substitutes for stationary cases of (2.9) into (2.12), so as

$$L(\mathbf{w}, b, \xi, \alpha, \lambda; C) = \sum_{i=1}^{N}\alpha_i - \frac{R}{2} - b\sum_{i=1}^{N}\alpha_i y_i + \sum_{i=1}^{N}\xi_i(C - \lambda_i - \alpha_i). \tag{2.13}$$

It recalls a fact of (2.9) of both primary and dual admissibilities. All of these are used in the manipulation of (2.13) as

$$L(\mathbf{w}, b, \xi, \alpha, \lambda; C) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i t_j x_i^T \cdot x_j. \tag{2.14}$$

Therefore, (2.10) and (2.14) create a dual form of the SVM optimization problem with $i = 1, ..., N$ as follows

$$\min_{\alpha} \left(\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i t_j x_i^T \cdot x_j - \sum_{i=1}^{N} \alpha_i \right) \text{ s.t. } \sum_{i=1}^{N} \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq C.$$

It obtains the Lagrangian formula from (2.14) as follows

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i t_j \kappa(x_i, x_j)$$

wherever $\kappa(x_i, x_j)$ is kernel functions. Kernels that will be used in this research are in Table 2.1.

Table 2.1*:* Kernel Functions

Types	$\kappa(x_i, x_j)$
Linear	$x_i^T x_j + c$
Polynomial	$(\gamma x_i^T x_j + c)^d, \ \gamma > 0$
RBF	$\exp(-\gamma \|x_i - x_j\|^2), \ \gamma > 0$

Immediate access to hardware is a complex problem, as it was never derived from early-generation computers. One of the resulting issues is their inability to run programs simultaneously. Then comes a kernel, an essential component for implementing a set abstraction layer. Abstraction is a method used to hide or simplify existing complexity, making access to the hardware device easier and more consistent. An example of the blessing of abstraction is a programmer's ease of work. There is no need to code binary to command the hardware but simply code accordingly mastered programming language. The kernel is likened to life, while the operating system is physical and the two elements need each other to work together. The kernel is software that contains all the services provided by the operating system for users. Contains some of the most basic components of an operating system. The kernel manages scheduling and context switching, exception handling and interrupt handling, and synchronization multiprocessor. The main function of the kernel is to manage computer resources and enable other programs to run and use the computer's resources.

2.9 Literature Review

Support Vector Machine (SVM), initially introduced by Vapnik in 1992, constitutes a collection of sophisticated and harmonized concepts within the realm of pattern recognition. Although relatively young as a pattern recognition method, SVM's assessment across diverse applications has positioned it as a cutting-edge approach in

this field, contributing to its rapid growth. SVM operates as a machine learning technique founded on the principle of Structural Risk Minimization (SRM), aiming to identify the optimal hyperplane that effectively separates two classes within input spaces.

The development of the Support Vector Machine (SVM) can be attributed to the collaborative efforts of Boser, Guyon, and Vapnik. It was first unveiled in 1992 at the Annual Workshop on Computational Learning Theory. The essence of SVM embodies the synthesis of several computational theories from previous years, including hyperplane margins introduced by Widrow and Hart (1973), Cover (1965), and Vapnik (1964), as well as the concept of kernels introduced by Aronszajn in 1950, among others. However, it was only in 1992 that a comprehensive attempt was made to unify these components [19, 20].

Diverging from the approach of neural networks that seeks a dividing hyperplane between classes, SVM strives to determine the best-fitting hyperplane in the input space. At its core, SVM operates as a linear classifier, yet it has evolved to address non-linear problems by integrating the concept of the kernel trick within high-dimensional spaces. This advancement has spurred research interest, driving investigations into SVM's theoretical capabilities and practical applications.

The key characteristics of SVM can be summarized as follows:

1. Linear Classifier: SVM functions as a linear classifier, aiming to identify a hyperplane that effectively separates data points belonging to different classes within the input space.

2. Higher-Dimensional Transformation: SVM conducts pattern recognition by transforming data from the input space into a higher-dimensional space. The optimization is then performed within this transformed vector space, setting SVM apart from many other pattern recognition solutions that optimize parameters in lower-dimensional transformed spaces.

3. Structural Risk Minimization (SRM): SVM employs the Structural Risk Minimization strategy, a principle that focuses on minimizing the generalization error by controlling the capacity of the learning algorithm. This approach helps to enhance the model's performance on unseen data.

4. Binary Classification: SVM is primarily designed for binary classification tasks, involving the separation of data points into two distinct classes. While its fundamental operation revolves around two classes, extensions and adaptations have been developed to address multi-class classification as well.

These summarized points capture the essential characteristics of SVM, highlighting its unique approach to classification and its utilization of higher-dimensional transformations for effective pattern recognition.

Certainly, here is a concise summary of the limitations of Support Vector Machine (SVM) based on the provided points:

1. Scalability Issues: SVM can encounter difficulties when applied to large-scale problems, particularly when dealing with a high number of samples to be processed. The computational demands of SVM may become challenging in scenarios involving a significant volume of data.

2. Binary Classification Emphasis: Initially, SVM is designed for binary classification problems involving two classes. Although modifications have been made to extend SVM to handle multi-class problems, these adaptations, such as the one-vs-rest and Tree Structure strategies, each come with their own drawbacks. As a result, ongoing research and development are dedicated to enhancing SVM's applicability to multi-class problems, reflecting an open research theme.

These summarized points provide insights into the limitations of SVM, particularly in terms of scalability and its original focus on binary classification. The ongoing efforts to address multi-class problems underscore the evolving nature of SVM research and its pursuit of improved capabilities.

In [21], their research explained that the latest machine learning for classification consists of two groups. Implementation in their research that the input was a non-linear vector to high-dimensional space. The analysis presents experimental data that was not separate.

In [22], an algorithm that maximizes the margin between the trial pattern and the decision boundary. This algorithm was applicable to various classification functions. The results of this study demonstrated a good generalization when compared to other algorithms.

In [23], the research presented the application of a support vector machine (SVM) to predict student educational completion using STEM. The results of this experimental analysis included prediction and ranking, as well as the development of a simpler system to support students.

In [24], a study carried out an exploration of the support vector method in predicting critical heat loads, namely heat transfer crises using the support-vector method was a model that has strong predictive power and works well on many datasets.

In [25], conducting artificial intelligence (AI) studied to analyze mathematical models for water quality in rivers. In this research, using root mean square error (RMSE), mean square error (MSE), and artificial neural network (ANN) in building the six regression models.

In [26], researching consumer cases in the Information Centric Network (ICN) which produced an update in predicting the probability of an identified location. The

purpose of this was to optimize the existence of a support vector machine that provided a level of success in minimizing content search time. In the exploration used polynomial kernels, linear and radial basis where the accuracy of polynomial kernels was the best.

In [27], educational data mining (EDM) study which was a branch of in-depth knowledge of computer science. This study conducted an analysis of student performance. Comparing linear support vector machine (LSVM) performance with classical machine learning algorithm performance.

In [28], behavior analysis of the SVM classifier when its parameters were assumed to have different values. The analysis consisted of illustrative, visual, and mathematical examples and geometric interpretations. The goal was to prove basic kernel functionality with SVM.

SVM has successfully applied to real-world problems and generally provides solutions that are more effective than classical methods such as the artificial neural network. This thesis introduces the explored concept of SVM and discusses its application in insurance, which is recently one of the fastest-growing fields.

Chapter 3

Research Methodology

3.1 Research Questions

This study focuses on the analysis of insurance data, specifically addressing various aspects of model performance evaluation (MPE), including receiver operating characteristics (ROCs), area under curve (AUC), partial AUC (pAUC), smoothing, confidence intervals, and thresholds. The significance of data analysis is emphasized across diverse sectors, where companies require accurate insights to evaluate their performance outcomes. However, many companies struggle with effectively analyzing data they possess and collect. This challenge may arise due to a lack of familiarity with suitable analytical methods for processing data.

In today's era of big data, numerous companies have transitioned into becoming data literate organizations, recognizing the crucial role of data. Despite this awareness, many companies still lack the necessary tools to efficiently organize, process, and analyze their data. Data analysis encompasses the comprehensive process of cleansing, interpreting, analyzing, and visualizing data using a range of techniques, methods, and tools. These data analysis tools play a pivotal role in enabling companies to extract pertinent insights, facilitating informed and intelligent decision-making.

In essence, this research sheds light on the importance of data analysis in the context of insurance data and its broader application in diverse sectors, underlining how effective data analysis can empower organizations to enhance their decision-making processes.

In data processing, two terms are known, namely analytics and analysis. This study will describe both the concept of analytic and analysis. Here an accurate algorithm evaluation is offered which is an algorithm formed from several statistical tools namely MPE, ROC, AUC, pAUC, smoothing, confidence intervals, and thresholds. CSVM is a classical SVM method without involving accurate evaluation algorithms in conducting data analysis. MSVM is built through the formation of CSVM using an accurate evaluation algorithm. The criterion of using an accurate evaluating algorithm produces accurate output and accuracy with the lowest iterations and the fastest computation time compared to CSVM. This thesis will focus on the introduction of an accurate evaluation algorithm that is offered so that a new concept from CSVM to MSVM is produced. The kernel used will focus on linear, polynomial, and radial basis. This thesis leads us to the following questions:

1. What is MSVM?

2. Does MSVM offer a new concept of CSVM?
3. How is the application of MSVM in classification?

Those questions will be presented in Chapter 4 of this thesis. The steps by steps for MSVM are as the following:

1. Model Performance Evaluation: Model Performance Evaluation is a critical process in machine learning that assesses how well a model generalizes to new, unseen data. It involves various techniques and metrics to quantitatively and qualitatively measure a model's predictive ability. Common steps in Model Performance Evaluation include:
    - Data Splitting: The dataset is divided into training and testing subsets or using techniques like cross-validation to prevent overfitting and ensure unbiased evaluation.
    - Metric Selection: Relevant evaluation metrics are chosen based on the problem type. For classification, these metrics could include accuracy, precision, recall, F1-score, ROC-AUC, etc. In regression, metrics like mean squared error (MSE) or R-squared are often used.
    - Model Training and Prediction: The model is trained on the training set and then used to make predictions on the test set.
    - Metric Calculation: The chosen metrics are calculated using the model's predictions and the ground truth labels or values from the test set.
    - Visualization: Visualization techniques like confusion matrices, ROC curves, precision-recall curves, or calibration plots help interpret and communicate the model's performance.
    - Comparative Analysis: Multiple models can be evaluated and compared using the same metrics to select the best-performing model.
2. ROC (Receiver Operating Characteristic) Curve: The ROC Curve is a graphical representation of a binary classification model's performance across different threshold settings. It plots the True Positive Rate (sensitivity) against the False Positive Rate (fallout). Key aspects of the ROC Curve include:
    - Threshold Variation: As the threshold for classifying positive instances changes, the trade-off between TPR and FPR is visualized on the ROC Curve.
    - Random Guessing Line: The diagonal line represents random guessing, and points above it indicate better-than-random performance.
    - AUC (Area Under the Curve): The AUC quantifies the overall performance of the model. AUC values range from 0.5 (random) to 1 (perfect classifier).

3. AUC (Area Under the Curve): The Area Under the ROC Curve (AUC) is a scalar value representing the extent to which a binary classification model can distinguish between positive and negative instances. A high AUC indicates good separation ability:
    - Interpretation: An AUC of 0.8 means the model has an 80% chance of ranking a randomly chosen positive instance higher than a randomly chosen negative instance.
    - Comparison: AUC allows for comparing different models' overall performance, independent of the chosen threshold.
4. pAUC (Partial Area Under the Curve): Partial AUC focuses on a specific segment of the ROC curve, between two predefined FPR values. It measures the model's performance within that FPR range and is useful when certain regions of the ROC curve are more relevant:
    - Calculation: Partial AUC is calculated by integrating the ROC curve over the specified FPR interval.
5. Smoothing: Smoothing techniques are used to reduce noise and fluctuations in data while preserving essential patterns:
    - Moving Average: Calculates the average of adjacent data points in a defined window to reduce high-frequency noise.
    - Exponential Smoothing: Assigns exponentially decreasing weights to older data points while emphasizing recent data.
    - Kernel Smoothing: Applies a weighting function (kernel) to data points to calculate weighted averages.
6. Confidence Intervals: Confidence Intervals provide a range within which a population parameter is likely to lie with a specified level of confidence. It considers the variability in sample data and the sampling procedure:
    - Calculation: The Confidence Interval is calculated based on the sample statistic, standard error, and chosen confidence level.
7. Confidence Intervals of a Threshold: Confidence Intervals of a Threshold estimate a range of threshold values for making predictions that yield a desired level of confidence in model performance:
    - Algorithm Steps: The process involves selecting a confidence level, sampling thresholds, evaluating model performance, and calculating Confidence Intervals for the thresholds based on performance distribution.

- Interpretation: The Confidence Intervals of a Threshold indicate a range of thresholds that likely lead to similar performance with the selected confidence level.

3.2 Methods

The researchers involved in this study are well-versed in and have extensively utilized a range of methodologies, including model performance evaluation (MPE), receiver operating characteristics (ROCs), area under curve (AUC), partial AUC (pAUC), smoothing, confidence intervals, and threshold confidence intervals. The preceding chapter thoroughly discusses the detailed descriptions and algorithmic implementations of these techniques. These methods have been well-established and applied across various fields of research, spanning engineering, economics, data analysis, and data science, among others.

In this thesis, an innovative approach is taken by amalgamating these methodologies into a unified framework, referred to as the "accurate evaluating algorithm" implemented in R. The process of this approach is structured across eight distinct steps, aimed at achieving the highest level of accuracy based on the analysis of insurance data conducted in this study. This integration represents a novel contribution, offering enhanced convenience and time efficiency for data accuracy analysis within the R environment.

Conventionally, data analysis involves examining each method individually. However, this research introduces a novel approach by performing data analysis using the comprehensive accurate evaluating algorithm, combining all eight processes in a single experiment. The results demonstrate that this approach led to faster computational times, with significantly reduced iteration counts.

The study performs a comparison between CSVM (Classical Support Vector Machine) and MSVM (Modified Support Vector Machine) using the accurate evaluating algorithm. CSVM represents a standard SVM data analysis step, while MSVM involves data analysis with the accurate evaluating algorithm integrated beforehand. The ensuing chapter will present the outcomes of this comparison through the performance profiles of these two methods.

The motivation behind this research stemms from the challenges faced by companies and governmental entities in managing complex data. The proposed algorithms aim to address these challenges by providing faster computation times and reducing the number of iterations required. The applicability of accurate evaluating algorithms extends beyond the academic realm, proving beneficial in non-academic sectors, particularly aiding data analysts in expediting the analysis of large datasets.

Both CSVM and MSVM methods are developed and executed using R program-

ming, with a focus on enhancing accuracy in data processing. Notably, both methods employ three popular kernels within SVM: linear, polynomial, and radial basis function (RBF) kernels. These kernels are selected based on their efficacy in achieving accuracy, as demonstrated in prior research such as that by Nurhidayat et al. [30], who explored kernel performance using RMSE and density metrics. All numerical experiments are conducted on a computing setup featuring Windows 10 Pro, an Intel Core i5-7500 CPU, 8GB RAM, and 64-bits.

In summary, this thesis primarily revolves around an innovative research methodology that combines various methodologies into a cohesive accurate evaluating algorithm. The approach is applied to insurance data analysis and holds promise for streamlining data analysis processes across different sectors. The subsequent chapters delve into the specifics of the research methodology and its outcomes.

1. Data processing is limited in 2011, 2012, and 2014.
2. Data processing with the accurate evaluating algorithm to build MSVM with three kernels is involved.
3. Simulations are both CSVM and MSVM.

There are four phases for the MSVM of this thesis as the following.

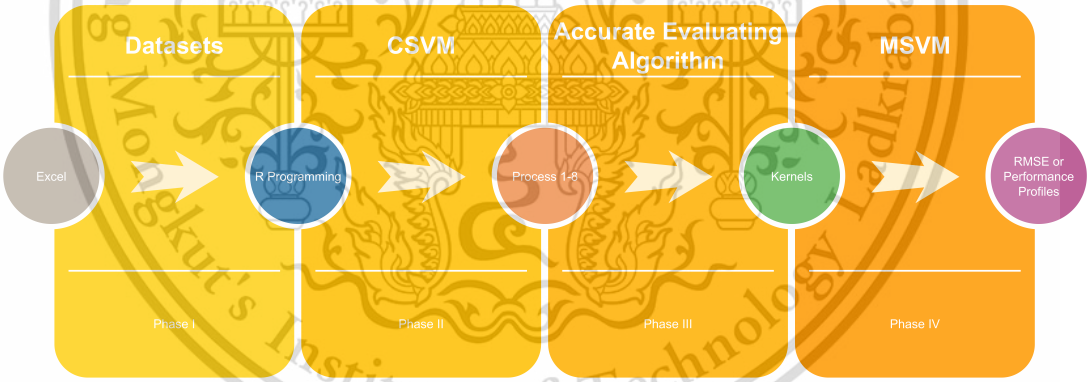

Figure 3.1: Modified Support Vector Machine (MSVM) Constructions.

Performance profiles are a valuable tool for comparing and visualizing the performance of multiple algorithms or methods across various problem instances. They provide insights into how different algorithms behave under different conditions and help researchers and practitioners make informed decisions when selecting an algorithm for a specific problem. Performance profiles are particularly useful when dealing with optimization or decision-making problems, where the choice of algorithm can have a significant impact on outcomes. Here are the key reasons for using performance profiles:

1. Comparison Across Algorithms: Performance profiles allow us to compare the performance of multiple algorithms on a set of problem instances. This is crucial for identifying which algorithms consistently perform well and which ones struggle in certain scenarios. It helps researchers understand the strengths and weaknesses of each algorithm.

2. Robustness Assessment: By analyzing performance profiles, we can assess the robustness of algorithms across different instances. Some algorithms might perform well on average but fail to handle specific instances effectively. Performance profiles reveal these instances, enabling us to target improvements or fine-tuning for those cases.

3. Algorithm Selection: Performance profiles aid in algorithm selection for specific problems. Instead of relying solely on average performance metrics, we can identify which algorithm is more likely to provide good results for a given instance. This can lead to better decisions when choosing an algorithm for practical applications.

4. Parameter Tuning: Performance profiles can guide parameter tuning efforts. Different algorithms might have specific parameter settings that perform better under certain conditions. By studying performance profiles, we can identify parameter configurations that lead to optimal results.

5. Insight into Problem Characteristics: Performance profiles can reveal patterns in algorithm behavior based on problem characteristics. This insight helps in understanding why certain algorithms excel in particular scenarios, providing valuable information for further algorithm development.

6. Performance Trade-offs: Performance profiles show the trade-offs between different performance metrics. Algorithms might excel in some metrics while lagging in others. Performance profiles help in identifying the algorithms that strike the right balance between different objectives.

7. Visualization of Performance: Graphical representations of performance profiles make it easier to interpret and communicate the results. They provide a clear visual overview of how algorithms compare on different instances and metrics.

8. Publication and Reporting: In research and reporting, performance profiles enhance the transparency and rigor of evaluations. They provide a more comprehensive view of algorithm performance compared to simple summary statistics.

To create a performance profile, we typically plot the cumulative distribution functions (CDFs) of the ratios of algorithm performance to the best-known performance on each instance. This allows us to observe how closely each algorithm approaches

the best-known performance for various levels of optimality. The closer an algorithm's curve is to the top-right corner of the graph which is its overall performance. In summary, performance profiles are a valuable tool for assessing, comparing, and selecting algorithms based on their performance across different problem instances. They offer a holistic view of algorithm behavior and help guide decision-making in algorithmic research, optimization, and real-world applications.

Performance profiles offer several advantages when evaluating and comparing the performance of different algorithms or methods on various problem instances. Here are some key advantages of using performance profiles:

1. Comprehensive Comparison: Performance profiles provide a comprehensive and holistic view of algorithm performance across multiple instances and metrics. Instead of relying solely on average or summary metrics, performance profiles allow us to assess how algorithms behave under different conditions and on different types of instances.
2. Objective Algorithm Selection: Performance profiles help us objectively select the most suitable algorithm for a specific problem. By considering performance across a range of instances and metrics, we can make informed decisions about which algorithm is likely to perform well in practice.
3. Robustness Assessment: Performance profiles help identify the robustness of algorithms. We can see how algorithms handle challenging or outlier instances, enabling us to target improvements or adjustments for specific cases.
4. Parameter Tuning Guidance: Performance profiles guide parameter tuning efforts by highlighting the parameter configurations that lead to better performance. We can identify parameter settings that consistently yield good results across different instances.
5. Insights into Algorithm Behavior: Performance profiles offer insights into how algorithms respond to different problem characteristics. This information can be valuable for understanding the strengths and weaknesses of each algorithm and for guiding further algorithm development.
6. Transparency and Reproducibility: Using performance profiles enhances the transparency and reproducibility of your evaluations. They provide a detailed record of algorithm performance on various instances, making it easier for others to understand and validate your results.
7. Balanced Performance Evaluation: Performance profiles help strike a balance between different performance metrics. Algorithms might excel in some metrics while lagging in others. Performance profiles provide a clear visualization of these trade-offs.

8. Graphical Representation: The graphical nature of performance profiles makes it easier to visualize and communicate complex information. The curves on the profile graph provide an intuitive way to compare algorithms and understand their relative performance.

9. Enhanced Decision-Making: Performance profiles enable better decision-making by considering both the strengths and weaknesses of each algorithm. This is especially important when making choices that impact real-world applications.

10. Publication and Reporting: Performance profiles enhance the quality of research publications and reports by providing a richer and more detailed evaluation of algorithm performance. They add depth to the analysis and strengthen the credibility of your findings.

11. Adaptability to Multiple Metrics: Performance profiles can be adapted to various performance metrics, allowing us to assess algorithm performance from different angles. This flexibility accommodates a wide range of evaluation criteria.

In summary, performance profiles offer a powerful and versatile approach to evaluating and comparing algorithms. They provide a nuanced understanding of algorithm behavior and enable data-driven decision-making in algorithm selection, parameter tuning, and problem understanding.

While performance profiles offer valuable insights into algorithm behavior and performance, they also have certain limitations and potential drawbacks. It is important to be aware of these disadvantages when using performance profiles for evaluating and comparing algorithms:

1. High-Dimensional Data: Performance profiles become complex and difficult to interpret when dealing with a large number of algorithms, instances, and metrics. Visualizing and analyzing high-dimensional performance profiles can be challenging and may require specialized tools.

2. Limited to Known Instances: Performance profiles rely on a predefined set of problem instances for evaluation. If the instances do not represent the full spectrum of real-world scenarios, the insights gained from the profiles may not generalize well.

3. Metric Dependency: The choice of performance metrics can influence the shape and interpretation of performance profiles. Different metrics may lead to different conclusions about algorithm performance. It is important to carefully select and justify the chosen metrics.

4. Misleading Rankings: In some cases, a small difference in performance on a few instances can lead to significant changes in the ranking of algorithms on a perfor-

mance profile. This may not always reflect practical significance or generalizability.

5. Overemphasis on Outliers: Performance profiles can disproportionately emphasize the behavior of algorithms on outlier instances, leading to skewed perceptions of algorithm performance.

6. Threshold Sensitivity: Performance profiles can be sensitive to the choice of threshold values, especially when algorithms exhibit similar performance. Small changes in threshold settings may lead to different conclusions about algorithm rankings.

7. Algorithm Complexity: The complexity of algorithms can impact their performance on different instances. A more complex algorithm might perform well on some instances but struggle on others, affecting its placement on a performance profile.

8. Data Variability: The variability of results across different runs or data samples can influence performance profiles. Stochastic behavior or randomness in algorithms can lead to inconsistent profiles.

9. Data Quality and Noise: Noise in data, inaccuracies in measurement, or errors in problem instance representation can introduce uncertainty and affect the reliability of performance profiles.

10. Interpretation Challenges: Interpreting performance profiles requires a solid understanding of the underlying algorithms, problem domains, and performance metrics. Misinterpretation of profile patterns can lead to incorrect conclusions.

11. Resource Intensive: Constructing performance profiles may require substantial computational resources and time, especially when dealing with large datasets or complex algorithms.

12. Subjective Thresholds: Deciding what constitutes an acceptable level of performance can be subjective and context-dependent. Performance profiles do not provide a definitive answer to the question of which algorithm is ”best”.

Despite these disadvantages, performance profiles remain a valuable tool for algorithm evaluation and comparison. To mitigate these limitations, it is important to carefully design experiments, select appropriate problem instances, consider multiple performance metrics, and critically interpret the results in the context of the specific problem and application.

Chapter 4

Results and Discussion

In the first and second chapters, we have discussed the general idea of this thesis including some literature reviews necessary for the discussion of this paper. The main ideas of this thesis have been informed in the methodology section of the third chapter. We have now come to the main chapter of this thesis which will consist of three parts. These sections will connect with each other and lead the reader to the conclusion and further work of this thesis. The first part, CSVM and MSVM over the polynomial kernel is of interest to the reader to review. Then, ends with the performance of CSVM and MSVM based on computation time and number of iterations. The second part examines some of the simulation results from the first part including an introduction to the root-mean-square error (RMSE). The analysis for the density and RMSE of the three kernels is reviewed at the end of the second section. Finally, SVM kernels for the sum insured close this chapter.

4.1 Simulation based on Modified SVM Polynomial Methods

This study encompasses a comprehensive set of methods and algorithms for data analysis prior to the classification process using the Modified Support Vector Machine (MSVM). The process begins by designing insurance databases within Microsoft Excel. Subsequently, these databases are transformed into the R programming language utilizing caret packages and various interfaces. Both classical and modified versions of MSVM are implemented, employing a polynomial kernel for the classification task. The algorithm for testing the polynomial kernel in both Classical Support Vector Machine (CSVM) and Modified Support Vector Machine (MSVM) is provided below:

Algorithm 8 MSVM polynomial

Require: Install caret, readxl ▷ library

Ensure: plot(svm poly)

1: read excel to data
2: data to train
3: data to test
4: train(data) to svm poly
5: test(data) to svm poly

SVM classifiers in this study will be constructed using non-linear kernels, specifically the polynomial kernel. The utilization of polynomial function SVM non-linear models is facilitated through the caret packages in R. The caret package automates

the selection of optimal model tuning parameters, thereby enhancing the accuracy of the model. During the experimentation, various trials are conducted with different scale values (2, 4, 8, 10) and degrees (1, 2, 3, 4). After executing the program, the optimal values are determined: degree = 1, scale = 0.1, and C = 0.25, resulting in a 92% accuracy rate [31].

To assess and compare the accuracy of the models, a numerical comparison known as a performance profile will be employed, drawing insights from research such as [29] and adapting their approach. The concept of performance profiles will be summarized, drawing inspiration from sources like [29, 34, 35, 36].

A specific numerical test will be conducted to compare the performance of the CSVM and MSVM polynomial methods. The performance ratio, which quantifies the relative performance of the two methods, is defined as:

$$r_{p,s} := \frac{f_{p,s}}{\min\{f_{p,s} : 1 \leq f \leq n_s\}}$$

where $f_{p,s}$ referred to the iteration and function evaluations that solver s spent on problem p and n_s refers to the number of problems in the model test. The cumulative distribution function is defined as

$$\rho_s(\tau) := \frac{1}{n_p}\text{size}\{p \in r_{p,s} \leq \tau\}$$

where $\rho_s(\tau)$ is the probability that a performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$.

The computing time is measured in seconds and is utilized for insurance data analysis across the four regions of Thailand: central, north, northeast, and south. The objective is to determine the computing time and number of iterations for both Classical Support Vector Machine (CSVM) and Modified Support Vector Machine (MSVM) with various values of τ: 2, 5, 7, 9, 11, 16, 20, 24, 28, and 31.

The results indicate that the integration of the accurate evaluating algorithm into MSVM leads to a substantial increase in data accuracy, reaching up to 86% according to performance profiles. In contrast, the CSVM polynomial approach achieves an accuracy of only 79%. The performance profiles illustrate this distinction clearly.

Performance profiles for computing time and the number of iterations in the MSVM polynomial method are depicted in Figures 4.1, 4.2, and 4.3, highlighting the superior performance of this approach. The incorporation of the accurate evaluating algorithm into the MSVM polynomial technique brings it closer to the performance index, underscoring its potential efficacy and high-performance characteristics.

In summary, the study demonstrates that the accurate evaluating algorithm integrated with the MSVM polynomial method significantly enhances data accuracy, outperforming the CSVM polynomial approach and approaching the performance index. The results are substantiated through performance profiles and graphical representations of computing time and iteration counts.

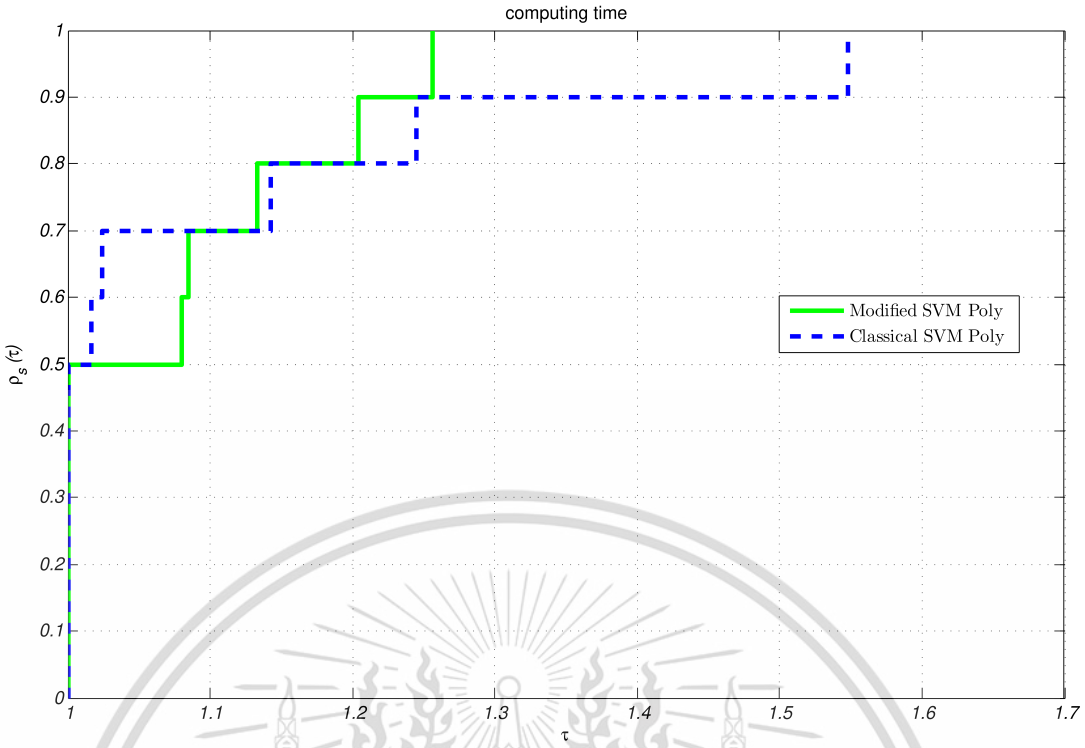

Figure 4.1: Probability Computing of $0 < \rho_s(\tau) < 1$ over MSVM and CSVM Polynomials.

In Figure 4.1, it seems to be that when discussing computational time, MSVM is higher accuracy than CSVM. Likewise, the iteration results from Figure 4.2 show an effective level of SVM modification efficiency. MSVM Polynomial runs concurrently with CSVM Polynomial and ends with MSVM Polynomial which is close to 1. This means that MSVM Polynomial is more effective in computing speed than CSVM Polynomial.

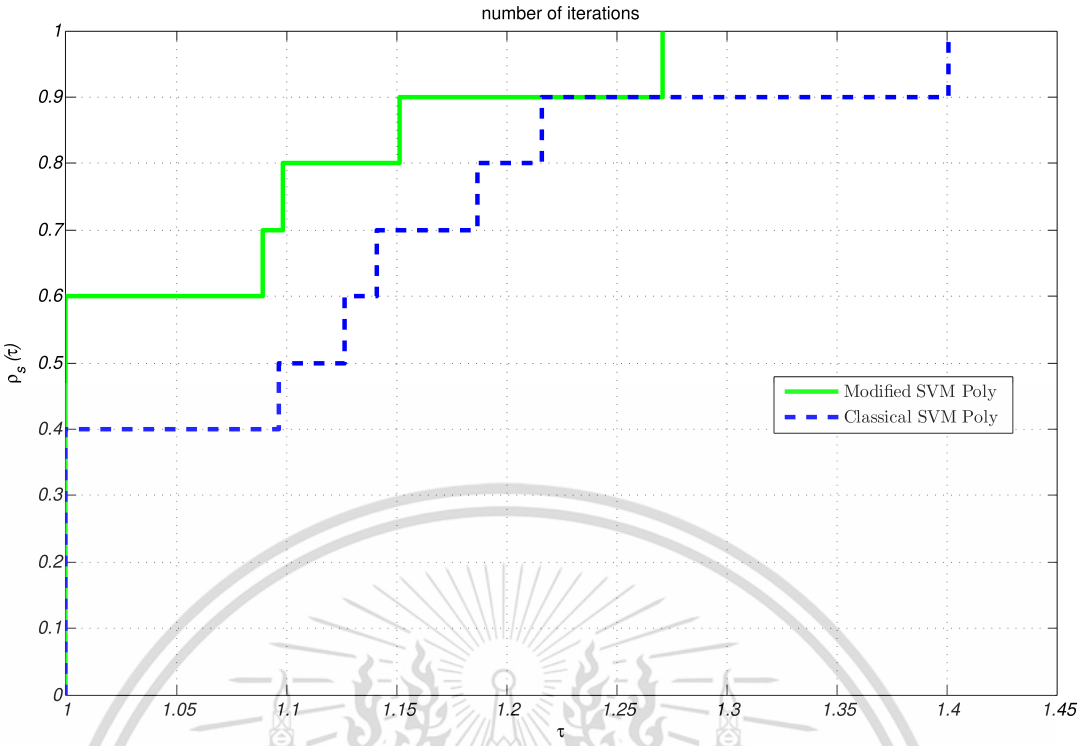

Figure 4.2: Probability Iterations of $0 < \rho_s(\tau) < 1$ over MSVM and CSVM Polynomials.

The comparison of accuracy between the two methods are in Figure 4.3. The computational performance with the addition of an accurate evaluating algorithm is an effective tool for data analysis without using it. It must be noted that Figures 4.1, 4.2, and 4.3 demonstrated on both methods, the best ones can be seen with their green color i.e., MSVM close to $0 < \rho_s(\tau) < 1$. It is confirmed that adding the accurate evaluating algorithm will impact the accuracy of the methods. MSVM Polynomial goes concurrently with the CSVM Polynomial and ends with the MSVM Polynomial which is close to 1. That is, the MSVM Polynomial is more effective in iteration speed than the CSVM Polynomial.

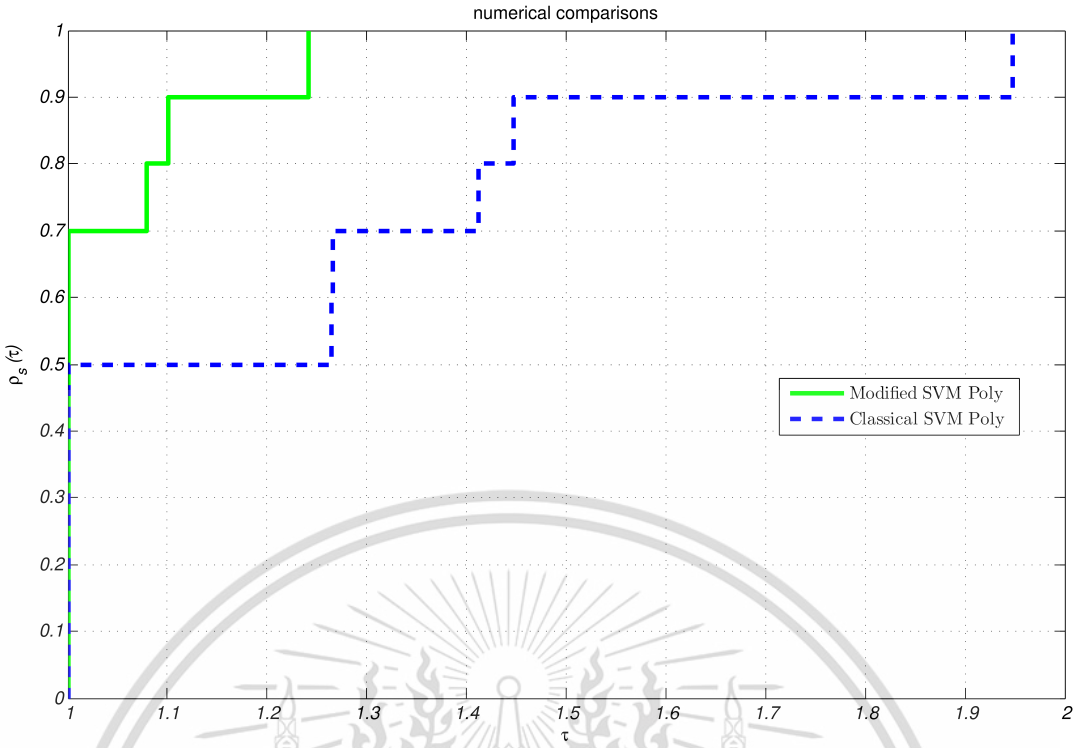

Figure 4.3: Probability Comparison of $0 < \rho_s(\tau) < 1$ over MSVM and CSVM Polynomials.

Figure 4.1 seems to be that when discussing computational time, MSVM polynomial method is an efficient tool. Likewise, iteration results from Figure 4.2 show a high performance of SVM modification efficiency. The comparison of accuracy between the two methods are in Figure 4.3. The computational performance with the addition of an accurate evaluating algorithm is more effective for data analysis than without using it. The best ones can be seen with their green color i.e., the MSVM polynomial closes to $0 < \rho_s(\tau) < 1$. MSVM Polynomial goes concurrently with the CSVM Polynomial and ends with the MSVM Polynomial which is close to 1. That is, the MSVM Polynomial is the best method than the CSVM Polynomial.

The incorporation of the accurate evaluating algorithm has been verified to have a significant impact on the accuracy of the methods. The results obtain from this study, along with a review of previous literature [1–4], demonstrate that classical approaches have not led to a substantial increase in accuracy for insurance data analysis. Notably, the MSVM method outperforms others in terms of effective performance, particularly concerning the number of iterations and computation time. This superiority can be attributed to the analysis of the accurate evaluating algorithm integrated within the MSVM method.

In the work of Qiao [37], a prediction model utilizing chaos theory is employed to predict stock market data. Interestingly, this prediction model can be effectively solved using the MSVM method with a polynomial kernel. Similarly, SVM has been

applied to stock market prediction in previous research [38], while the MSVM has been shown to enhance prediction model accuracy in stock market datasets [39]. Notably, this research goes beyond the scope of forecasting based solely on chaos theory [40]. While sentiment analysis and quantum computing theories have limitations, the MSVM method with its accurate evaluating algorithm has provided a useful approach for accurate data forecasting, verified through performance profiles.

Performance profiles emerge as a valuable tool for predicting accuracy across various computational and iterative scenarios. This technique has implications beyond insurance data analysis, finding applicability in engineering domains. Specifically, the accurate evaluating algorithm employs in building the MSVM method can potentially replace existing algorithms such as GSAPSO [42]. The potential applications of the MSVM method extend to power systems [43], energy consumption forecasting [44], predicting building energy consumption, forecasting photovoltaic output power [45], and even analyzing home presence using active power in energy research [46].

In light of these applications, the MSVM method represents a promising avenue for forecasting across diverse fields. It offers enhanced effectiveness compared to the CSVM method and presents a new approach to accurate forecasting that can be harnessed across various domains.

4.2 Simulation based on Modified SVM RBF Methods

This chapter focuses on the accurate evaluating algorithm used to validate data before categorizing it with the MSVM RBF kernel method. The algorithm makes use of sum-insured data for conducting these scientific simulations. Model Performance Evaluation (MPE) is utilized to assess a company's future progress. Graphical simulation estimates can be developed by considering data on influencing factors. The prediction model can be influenced by the type of company. Classification is based on the incorrect anticipation of inclusion in the sample group, while regression is performed to estimate the quantity. These are used as starting points for selecting appropriate measures and constructing a robust functional model in both areas. Easy access to data can be achieved by making the necessary adjustments to the database. The performance model evaluation in Figure 4.4 indicates that if the cut-off is set to 0, the precision remains consistently around 0.96.

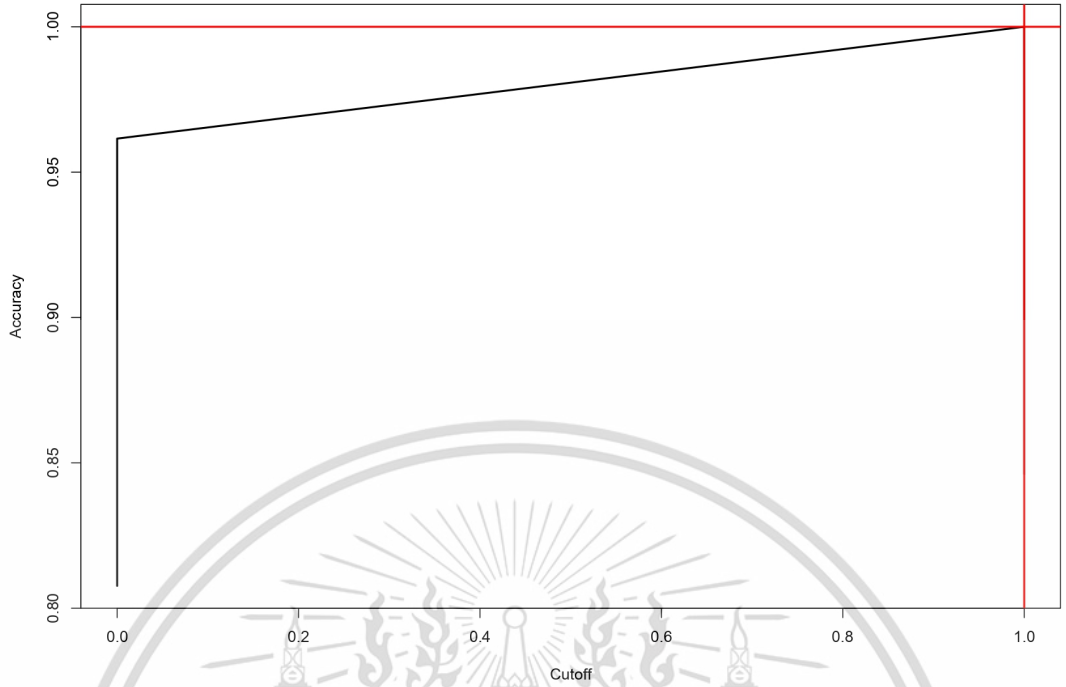

Figure 4.4*:* Performance Evaluation (1,1).

The cut-off state shows a rapid increase, while the accuracy demonstrates a slower increase. It appears that the horizontal line corresponds to the best accuracy value for this model, which is equal to 1. Additionally, the vertical red line is positioned at 0.9999326, representing the optimal cut-off value. Therefore, the point where the red horizontal and vertical lines intersect is considered the best estimation point for this model. The algorithm for MPE testing in R can be found in reference [31], with an expected output greater than 70%. Figure 4.5 illustrates the two approaches through the simulation of test data and training data. Receiver Operating Characteristics (ROCs) serve as a tool for determining thresholds in classifiers while optimizing True Positive Rate (TPR) and minimizing False Positive Rate (FPR). The threshold value in this case consists of the actual number 365, with a corresponding threshold number of 283. The ROC metric measures the number of correct positive classifications generated while also considering the increase in imprecise positives. ROC curves can be generated through multiple threshold score trials using probabilistic classifiers [6, 7]. In general classification, ROC-AUC performance is simulated for various thresholds. ROC-AUC evaluates the performance of a classification model based on the accuracy of both correct and incorrect classifications. Further details on the ROC algorithm can be found in reference [31], with an expected output of 100%. Plots for TPR and FPR are depicted as follows.

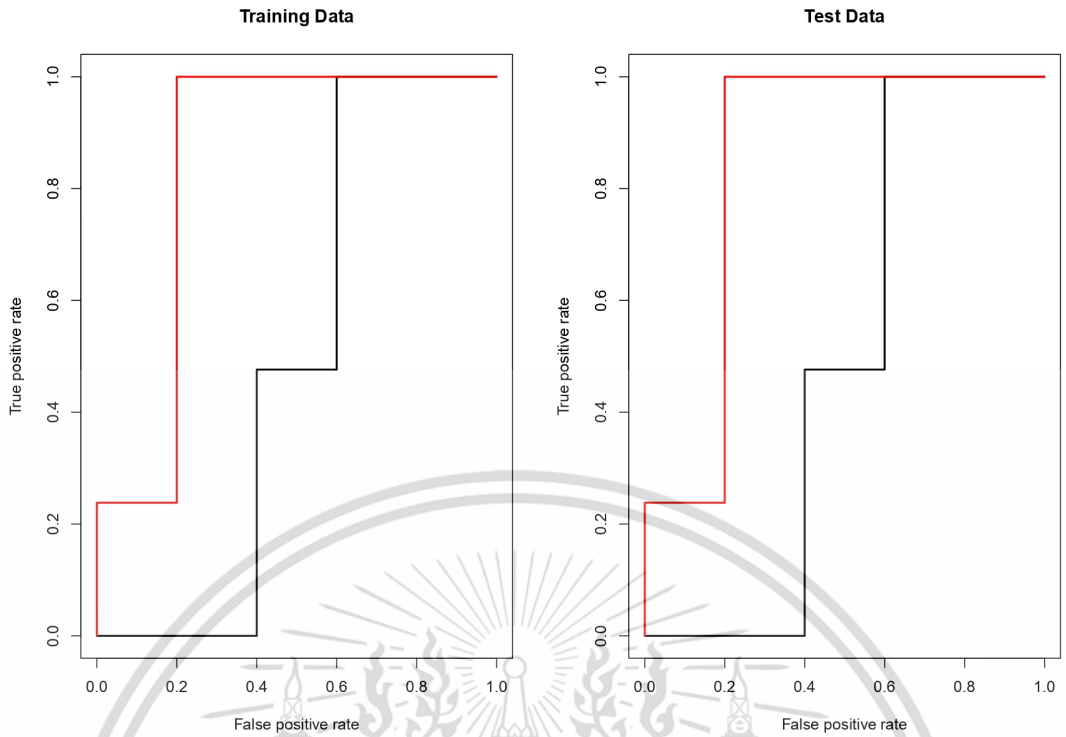

Figure 4.5: TPR and FPR based on ROCs.

When considering the curve ranging from 0.0 to 0.1 and ultimately to 1.1, the accuracy reaches 100%. Although it is not entirely in line with the ideal state in graphical simulations using this data, an attempt is made to input an intercept of 0 and a slope of 1. This results in Figure 4.5, which illustrates the position of the line above the straight-line p-value. Drawing a straight-line p-value ensures the evaluation of data suitability for the classification study. The Area Under Curve (AUC) illustrates the ROC curve, enabling classifiers to distinguish between classes [7]. AUC-ROC is a metric used to evaluate the performance of a classification model, with a higher AUC indicating a more accurate model. The AUC algorithm is explained in reference [31], with an expected output greater than 50%. This test enhances the value and reliability of the data, thereby contributing to the improvement of data accuracy.

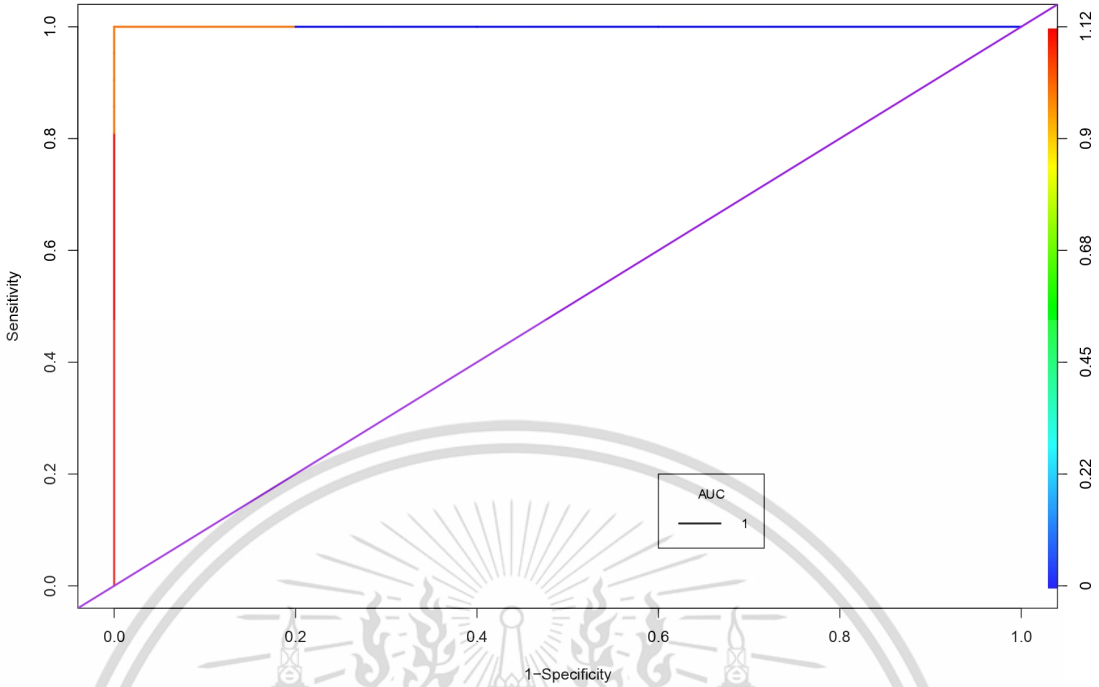

Figure 4.6*:* Area under the ROC Curve.

In Figure 4.6, data is depicted above the p-value line, indicating its suitability for classification. The total area in Figure 4.6 amounts to one, and when considering the area below the p-value line, it corresponds to 50%. Therefore, the area under the curve for the constructed model will exceed 50%. Partial AUC serves as a data analysis tool and aids in checking data accuracy. For additional information on the validation of partial ROC and related matters, please refer to references [11, 12]. The pAUC algorithm is provided in [31] with an expected output of 1. This process is undertaken to evaluate the sensitivity and specificity of the examined data, aiming to prevent errors.

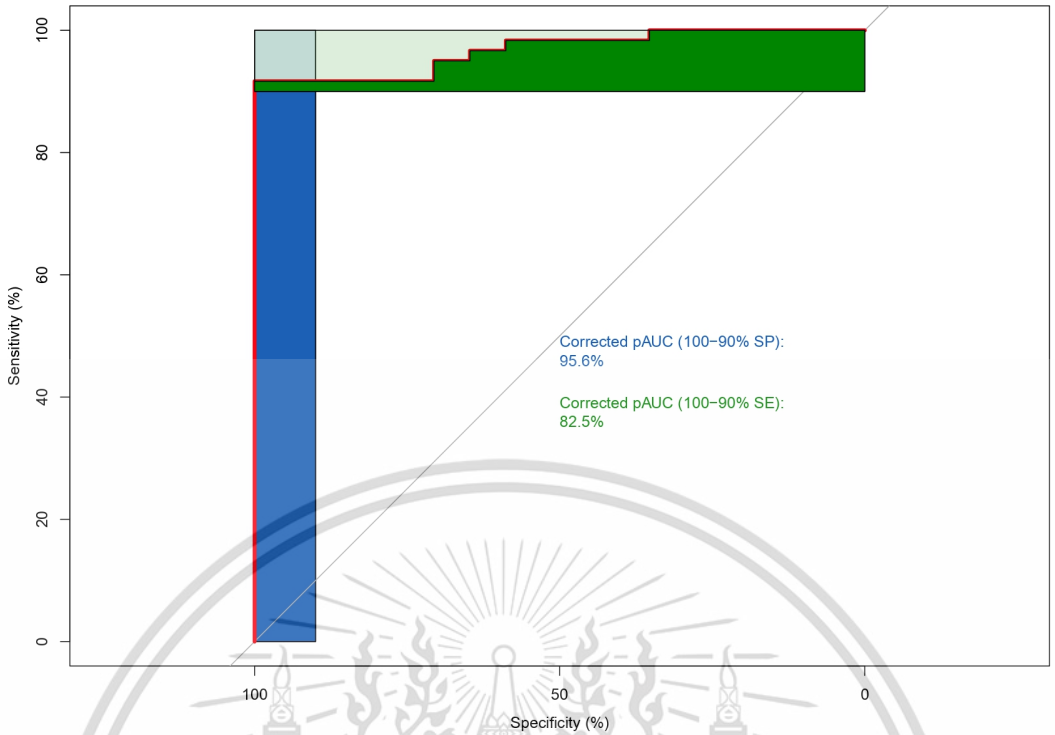

Figure 4.7*:* Sensitivity and Specificity based on pAUC.

In Figure 4.7, it has obtained the performance of AUC which is 1. Smoothing is a technique employed to eliminate unused data points from a dataset, resulting in a more regular data pattern. It finds extensive usage in insurance data processing, and the underlying concepts of smoothing techniques are widely applied in machine learning [13, 31]. The smoothing algorithm is detailed in [31]. The output of all lines in the graph is positioned above the straight line, indicating the effectiveness of the smoothing process. Figure 4.8 demonstrates the data smoothing effect. Several studies have employed this technique to identify trends in the realms of business and finance.

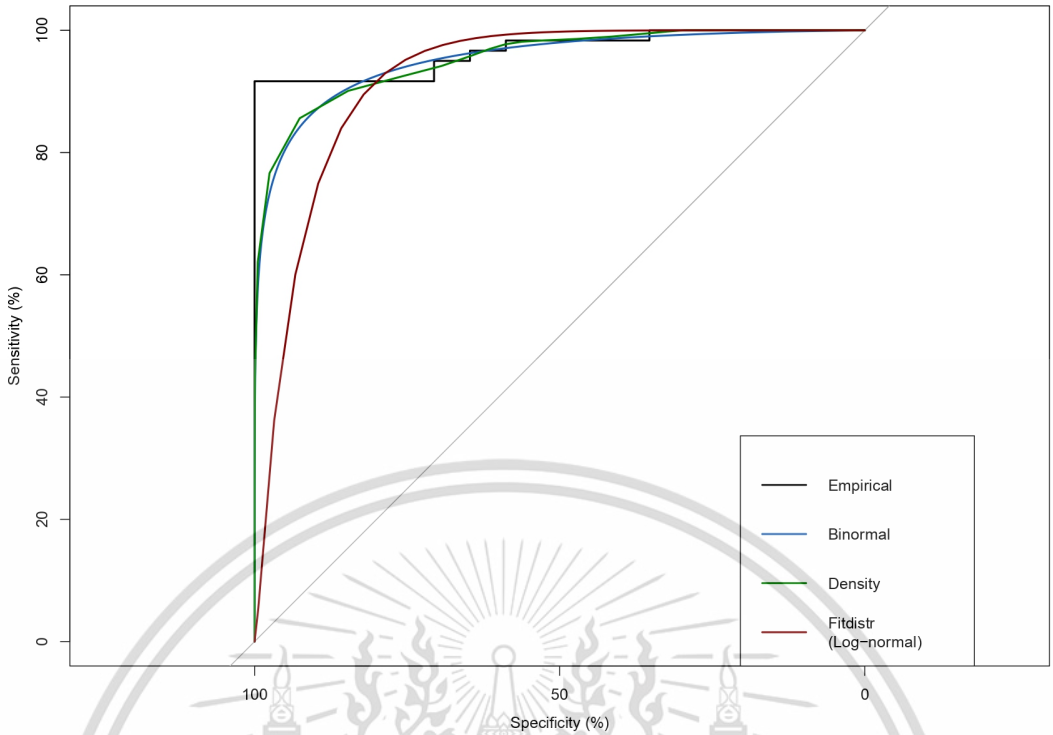

Figure 4.8: Comparisons of Empirical, Binormal, Density, and Fitdistr.

From Figure 4.8, the empirical ROC curve for the sum insured is represented in gray, showcasing three different smoothing methods. The smoothing analysis for sum insured data is presented in Figure 4.9. The percentage of individuals with positive intervals and conditions expected to be positive is known as sensitivity (TPR). Conversely, the percentage of individuals with negative conditions and conditions predicted to be perilous is referred to as specificity (FPR). The algorithm is explained in detail in [31]. The expected output is 100% for specificity and greater than 91% for sensitivity.

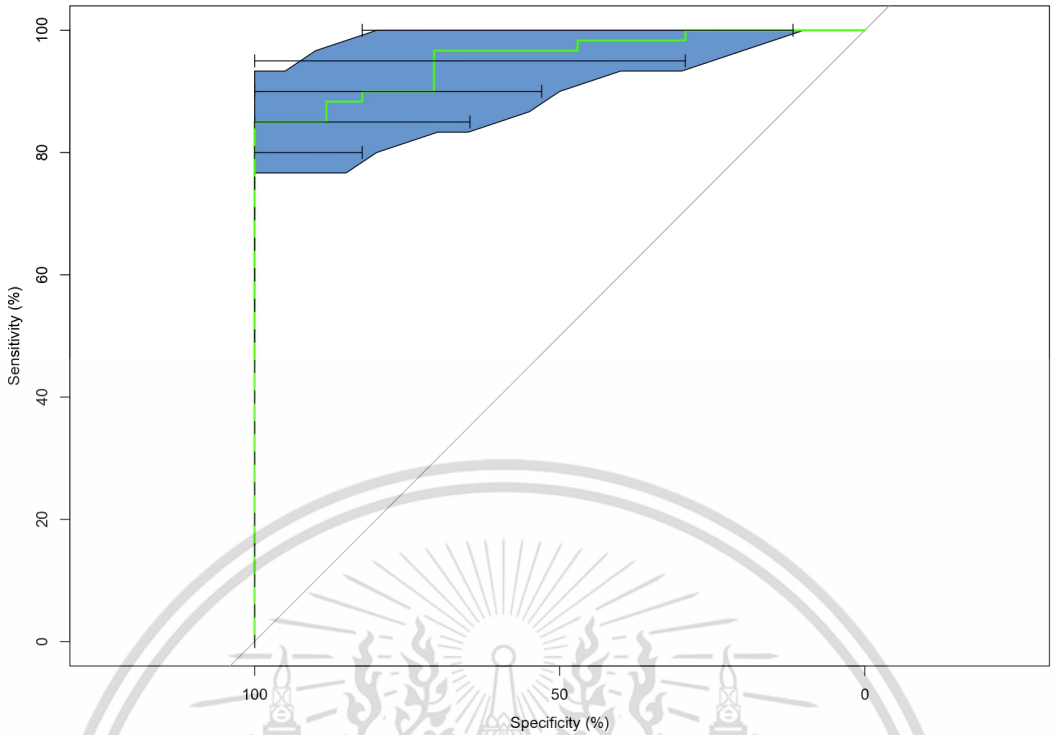

Figure 4.9*:* Percentages of Specificity and Sensitivity based on Confidence Intervals.

The confidence interval for the sum insured is established with a threshold of approximately 1914474.2, achieving a specification of up to 100% and a sensitivity of up to 91%. Further details can be found in Figure 4.9. To determine the most accurate threshold, it is necessary to conduct sample size testing, which has been described in references [16, 17]. The algorithm, illustrated in [31], is expected to yield an output greater than 97%. Certain critical decisions warrant a confidence level of up to 99.9%. The results of the sum insured data test for the threshold are presented in Figure 4.10.

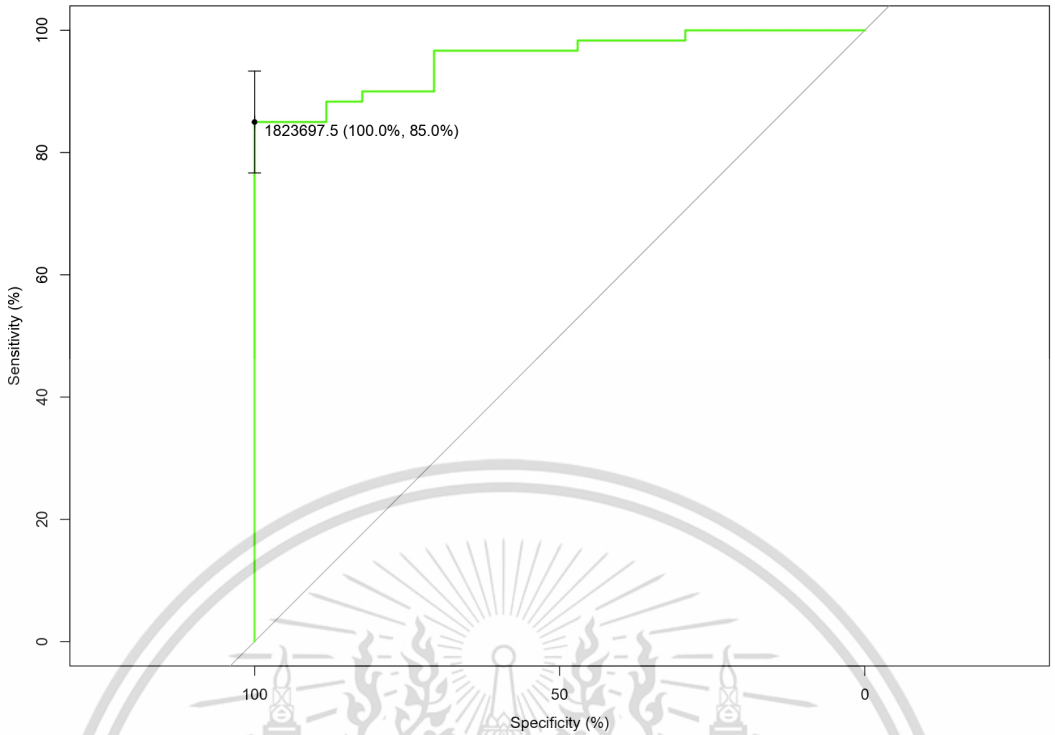

Figure 4.10: Confidence Intervals of Insurance Data with a Threshold.

The exploration of this numerical simulation is based on [30]. It is conducted by designing sum-insured databases in Microsoft Excel using the caret package for the MSVM RBF kernel method. The following algorithm outlines the testing procedure for MSVM RBF.

Algorithm 9 MSVM RBF Kernel Method

Require: caret, readxl

Ensure: plot(svm radial)

1: read excel to data
2: data to train
3: data to test
4: train(data) to svm radial
5: test(data) to svm radial

MSVM classifiers will be constructed using non-linear kernels, specifically the RBF kernel. The caret package is employed to facilitate the computation of SVM-RBF non-linear models.

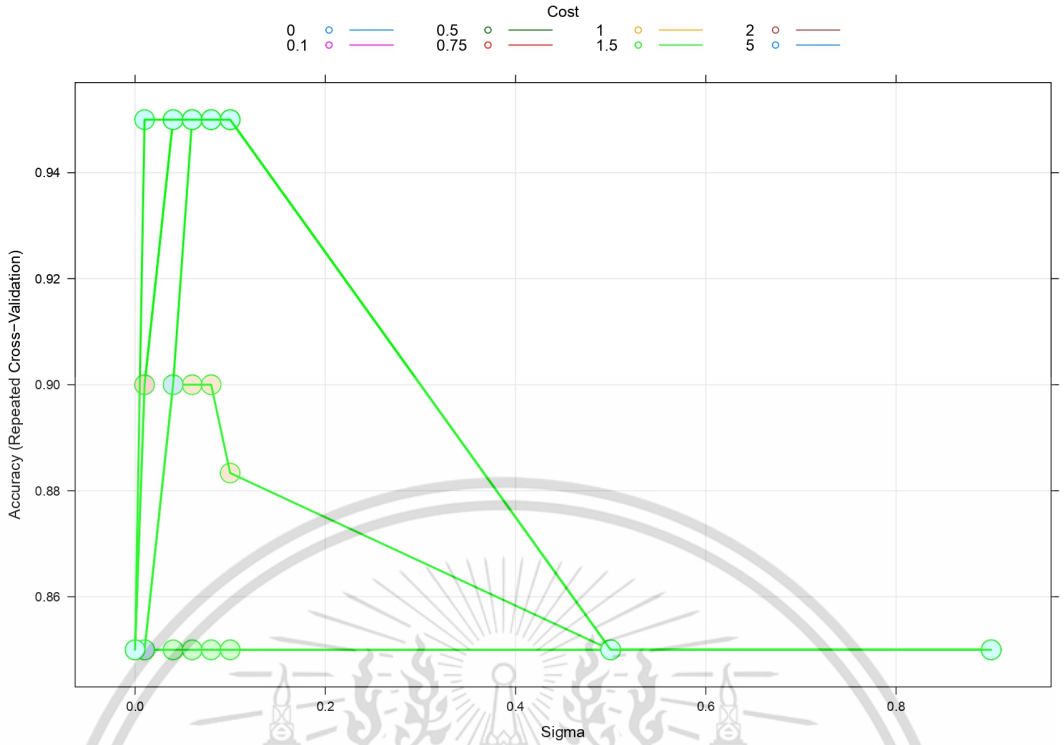

Figure 4.11: MSVM RBF Kernel.

The final values after executing the program are sigma is equal to (0,1), the cost is 0, 0.1, 0.5, 0.75, 1, 1.5, 2, and 5. The accuracy of this kernel is up to 83%.

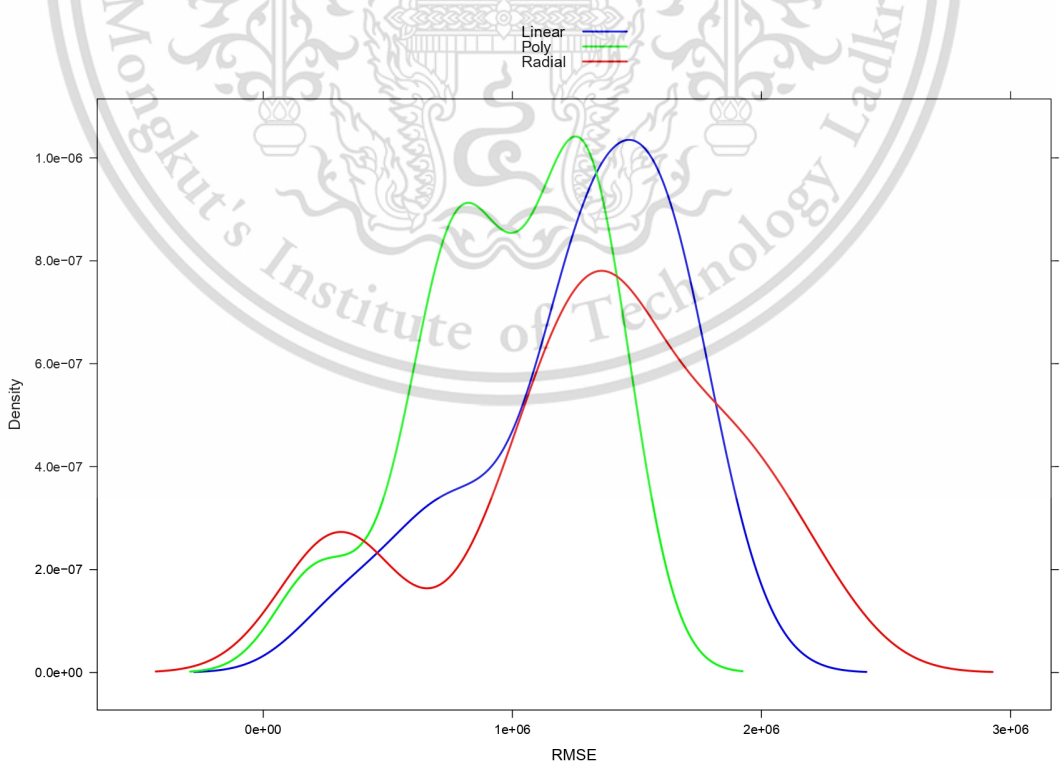

Figure 4.12: Kernel Comparisons.

The results of the Root-Mean-Square Error (RMSE) shown in Figure 4.12 demonstrate that the RBF kernel outperforms the linear and polynomial kernels. The SVM RBF kernel method exhibits the highest accuracy. A comparative study with other methods has been discussed in [30]. Additionally, further research has explored the utilization of SVM prediction with alternative kernels in the stock market and financial chaotic systems [47].

4.3 Insurance Data Clustering based on SVM Kernel Methods

In this section, we will conduct statistical analysis on the sum insured data utilized for Support Vector Machine (SVM) classification, specifically the Modified Support Vector Machine (MSVM) classification, as referenced in previous studies [48, 49]. This analysis marks the continuation of the data validation process that commenced in the model performance evaluation section, extending to the statistical comparison of the selected data samples.

The datasets under consideration include information spanning the years 2011, 2012, and 2014. Our study focuses on data from distinct regions in Thailand, and for each region, we employ models from the mentioned years (2011, 2012, and 2014). The aim is to perform statistical comparisons on the sum insured data, as illustrated in Figure 4.13. To carry out this analysis, we employ R programming to execute a series of data checks, which are categorized into four main groups. Subsequently, these checks undergo further validation through additional scientific computations. The specific R code used for testing the statistical comparisons is provided in the reference [50].

Throughout the process, we utilize three kernels in our analysis, and our objective is to identify any minimal discrepancies to simulate. The outcome of the analysis leads us to categorize areas into two classes: ”0” represents regions with high insurance costs, while ”1” signifies excellent areas characterized by reduced insurance prices.

In summary, this section delves into the implementation of comprehensive statistical analysis on the sum insured data, focusing on SVM classification (specifically MSVM classification) across different regions in Thailand. Through meticulous data checks, scientific computations, and the utilization of various kernels, our aim is to discern patterns and relationships in the sum insured data that contribute to the classification process.

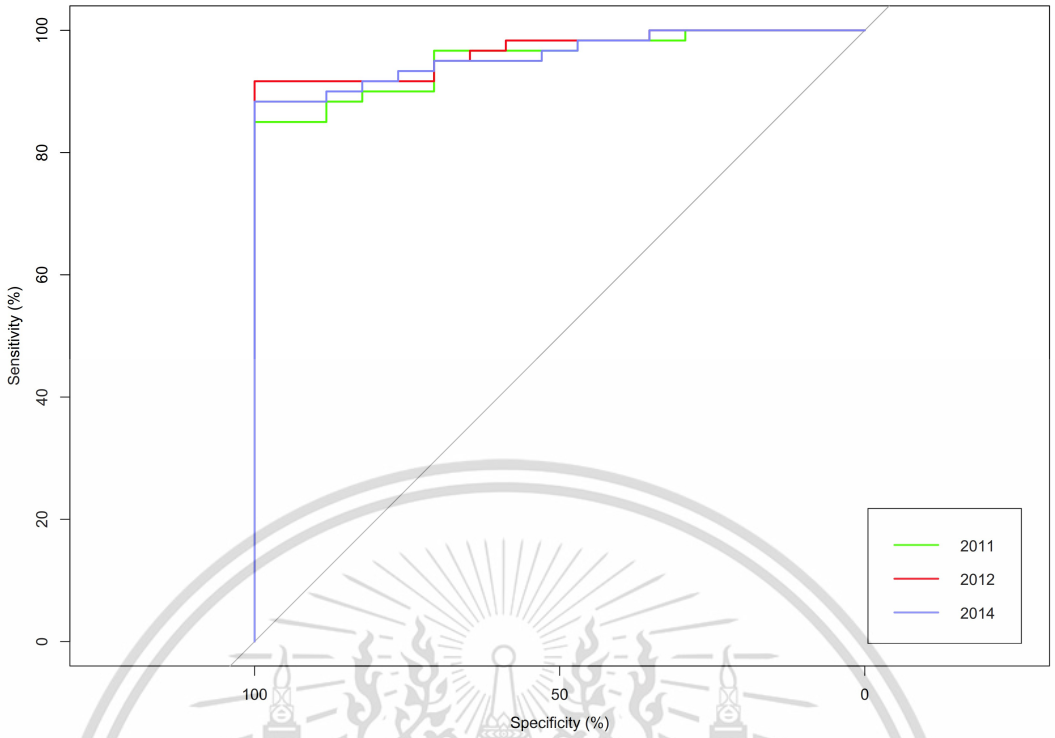

Figure 4.13: Statistical Comparison.

In our simulations, we utilize three different kernels for Modified Support Vector Machine (MSVM): linear, polynomial, and radial basis function (RBF) kernels. Throughout the simulation process, we apply the collected sum insured data for the years 2011, 2012, and 2014.

For example, when considering data for the year 2011, it encompasses the total sum insured data from every province in Thailand. The country is segmented into four distinct regions: central, north, northeast, and south. The 77 areas within Thailand are categorized based on their geographical locations, with 26 parts in the central region, 17 in the north, 20 in the northeast, and 14 in the south. The same categorization technique is applied to the data from 2012 and 2014.

The primary objective of this simulation is to identify the optimal kernel for categorizing insured data and informing future studies. We achieve this by evaluating data using the MSVM algorithm with three alternative kernels. The two classification levels are defined as follows: level 0 signifies areas where higher sum insured payments are recommended, while level 1 indicates areas where reduced sum insured payments are preferred.

The simulation process initiates with the creation of a sum insured database within Microsoft Excel. Subsequently, this database is utilized in R programming through the application of MSVM classification. We employ the caret packages and implement the three selected kernels: linear, polynomial, and RBF.

For the MSVM linear testing, the specific R code can be found in the reference [30]. Figure 4.15 depicts the results obtained using the MSVM linear kernel for each region in Thailand, utilizing sum insured data samples from the years 2011, 2012, and 2014.

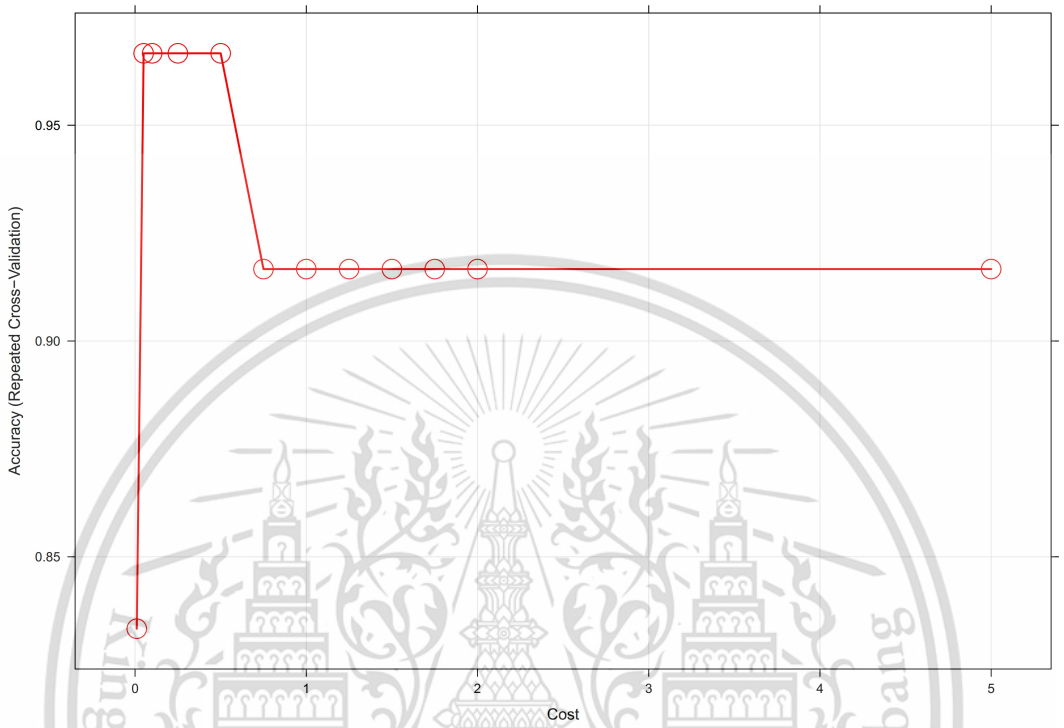

Figure 4.14: MSVM Linear Kernel.

Figure 4.15 illustrates the sum insured data in Thailand, showcasing an impressive accuracy of 90% achieved through the MSVM linear kernel algorithm. A crucial tuning parameter, denoted as C (cost), plays a significant role in this accuracy. C serves as a means to manage potential misclassifications by imposing a penalty on the model for making errors. Essentially, a higher value of C results in a reduced likelihood of the MSVM linear kernel algorithm misclassifying data points. In the specific context presented, caret, the software package, generates an MSVM linear classifier with C set to 1. However, it is worth noting that multiple values of C can be input to determine the optimal value that maximizes cross-validation accuracy of the model.

The MSVM with a linear kernel is a powerful machine learning algorithm used for classification and regression tasks. Here is an in-depth explanation of the SVM Linear algorithm:

1. Introduction to MSVM: MSVM is a supervised learning algorithm that aims to find the optimal hyperplane that best separates classes in a feature space. It is particularly effective when dealing with complex data that is not linearly separable.

2. Linear Separability and Hyperplane: MSVM with a linear kernel focuses on data that is linearly separable, which means that classes can be separated by a straight line (or hyperplane in higher dimensions).

3. Kernel Trick: MSVM algorithm can be enhanced by using kernels, which allow it to implicitly map the input features into a higher-dimensional space. In the case of MSVM with a linear kernel, no such mapping is applied, and the algorithm works in the original feature space.

4. Maximizing Margin: MSVM aims to maximize the margin, which is the distance between the hyperplane and the nearest data points of each class. These nearest points are called support vectors.

5. Hyperplane Equation: In a binary classification problem, the equation of the hyperplane is represented as $w * x + b = 0$, where w is the weight vector, x is the input feature vector, and b is the bias term.

6. Soft Margin: MSVM allows for some misclassification by introducing a soft margin, which permits a certain number of data points to be on the wrong side of the margin. This is helpful when the data is not perfectly separable.

7. Cost Parameter: The trade-off between maximizing the margin and minimizing the misclassification is controlled by the cost parameter C. Higher C values penalize misclassification more heavily, potentially leading to a smaller margin.

8. Decision Function: To make predictions, MSVM calculates the sign of the decision function $f(x) = w * x + b$. If $f(x)$ is positive, the instance is classified as one class; if negative, it is classified as the other class.

9. Dual Problem and Lagrange Multipliers: The optimization problem for MSVM can be reformulated into the dual problem involving Lagrange multipliers. Solving the dual problem leads to finding the support vectors and the optimal hyperplane.

10. Application to Multi-Class Problems: MSVM with a linear kernel can be extended to handle multi-class classification using techniques like One-vs-Rest or One-vs-One.

11. Regularization: MSVM algorithm naturally includes L2 regularization, which helps prevent overfitting.

12. Advantages and Limitations: MSVM linear kernel is efficient for high-dimensional data, offers good generalization, and works well when classes are separable. However, it may struggle with highly non-linear data.

In summary, MSVM with a linear kernel is a powerful algorithm for binary classification that aims to find an optimal hyperplane that separates classes in the original

feature space. By maximizing the margin between classes, MSVM can achieve robust and accurate classification, making it a valuable tool in various applications.

For MSVM polynomial testing, the R code can be accessed from the reference [30]. This code facilitates the creation of MSVM classifiers using non-linear kernels such as polynomial or radial basis functions. With the aid of the caret package, the polynomial and radial basis function SVM non-linear models can be swiftly computed, as depicted in Figure 4.15.

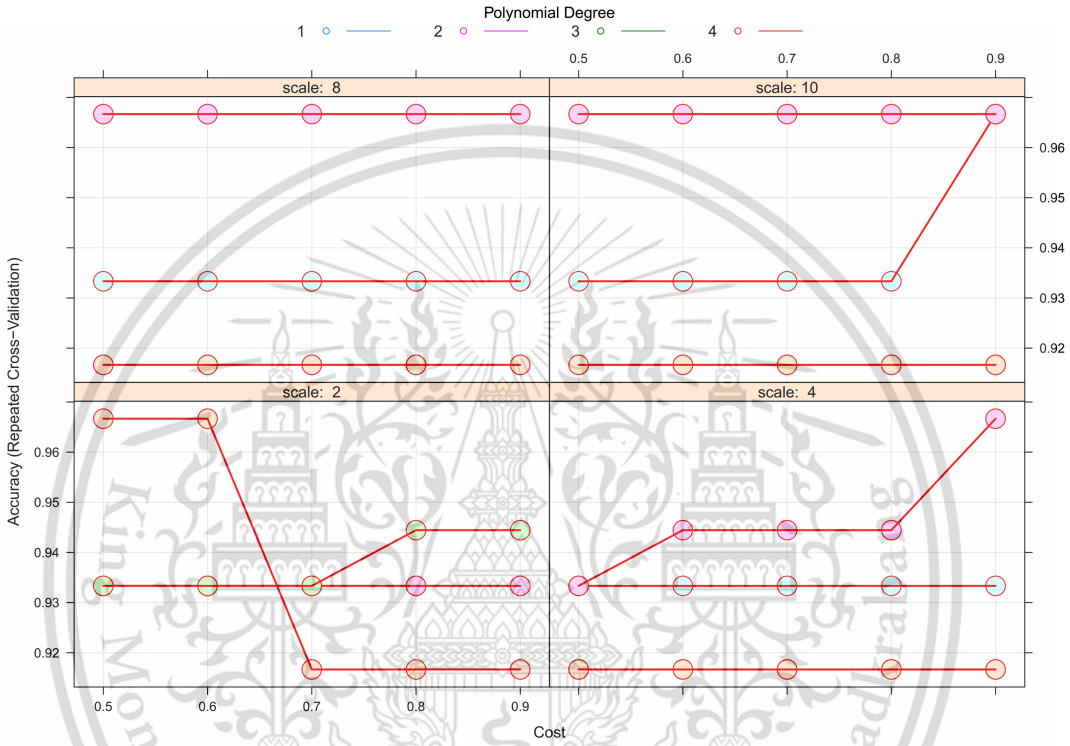

Figure 4.15: MSVM Polynomial Kernel.

MSVM with a polynomial kernel is a machine learning technique used for classification and regression tasks. Here is an in-depth explanation of the SVM polynomial algorithm:

1. Introduction to MSVM with Polynomial Kernel: MSVM is a supervised learning algorithm used for binary and multi-class classification, as well as regression tasks. Polynomial kernel is one of the kernel functions that MSVM can use to map data into a higher-dimensional space.

2. Kernel Trick and Polynomial Kernel: MSVM algorithm can work effectively in higher-dimensional feature spaces without explicitly transforming the data. Polynomial kernel is used to implicitly transform the data into a higher-dimensional space to make it linearly separable.

3. Polynomial Feature Mapping: In the case of a polynomial kernel, the feature

space is expanded using polynomial combinations of the original features. This allows the MSVM to capture non-linear relationships between the features and the target variable.

4. Kernel Function: Polynomial kernel function is defined as $K(x, y) = (x * y + c)^d$, where x and y are input feature vectors, c is a constant, and d is the degree of the polynomial.

5. Degree and Coefficient Parameters: The degree d and coefficient c are hyperparameters of the polynomial kernel. The degree determines the complexity of the polynomial transformation, while the coefficient influences the impact of higher-degree terms.

6. Maximizing Margin: Like the linear MSVM, the polynomial MSVM aims to find the hyperplane that maximizes the margin between classes while allowing for a certain amount of misclassification (soft margin).

7. Kernel Matrix: To work in the higher-dimensional space, MSVM computes a kernel matrix that represents the inner products between transformed data points. The polynomial kernel matrix is calculated using the polynomial function.

8. Kernel Trick and Computation Efficiency: The kernel trick avoids explicitly computing the higher-dimensional feature vectors, making the computation more efficient.

9. Support Vectors: Support vectors are the data points closest to the decision boundary and play a crucial role in defining the hyperplane.

10. Solving the Optimization Problem: MSVM with a polynomial kernel involves solving the dual optimization problem using techniques like Quadratic Programming. Lagrange multipliers are used to find the optimal hyperplane.

11. Classification Decision: The classification decision is made based on the sign of the decision function, similar to the linear MSVM. The decision function incorporates the support vectors and their corresponding Lagrange multipliers.

12. Advantages and Limitations: MSVM with a polynomial kernel is effective in capturing non-linear relationships in the data. It can handle moderately complex decision boundaries. However, selecting appropriate values for the hyperparameters and avoiding overfitting can be challenging.

13. Regularization and Overfitting: The cost parameter C controls the trade-off between maximizing the margin and minimizing misclassification. A small C encourages a wider margin but allows more misclassification, while a large C emphasizes accurate classification but may lead to overfitting.

In summary, MSVM with a polynomial kernel is a powerful algorithm that employs the polynomial kernel trick to capture non-linear relationships between features and target variables. It transforms data into a higher-dimensional space and constructs an optimal hyperplane to separate classes. Careful tuning of hyperparameters is essential for achieving good performance with the MSVM polynomial algorithm.

The caret package plays a pivotal role in automating the selection of optimal model tuning parameters, thereby enhancing the overall accuracy of the model. In the course of experimentation, various combinations of parameters are tested. Specifically, for the scales, values of 2, 4, 8, and 10 are explored along with degrees 1, 2, 3, and 4. After conducting these trials, the final selected values are as follows: degree is set to 1, scale is set to 0.1, and C (cost) is set to 0.25. This configuration yields an impressive accuracy of 92%. For those interested in the specifics of MSVM RBF (Radial Basis Function) testing, the corresponding R code can be accessed via the reference [30]. This code serves as a valuable resource for understanding and implementing MSVM classifiers using the RBF kernel, a non-linear approach that can effectively capture intricate relationships within data.

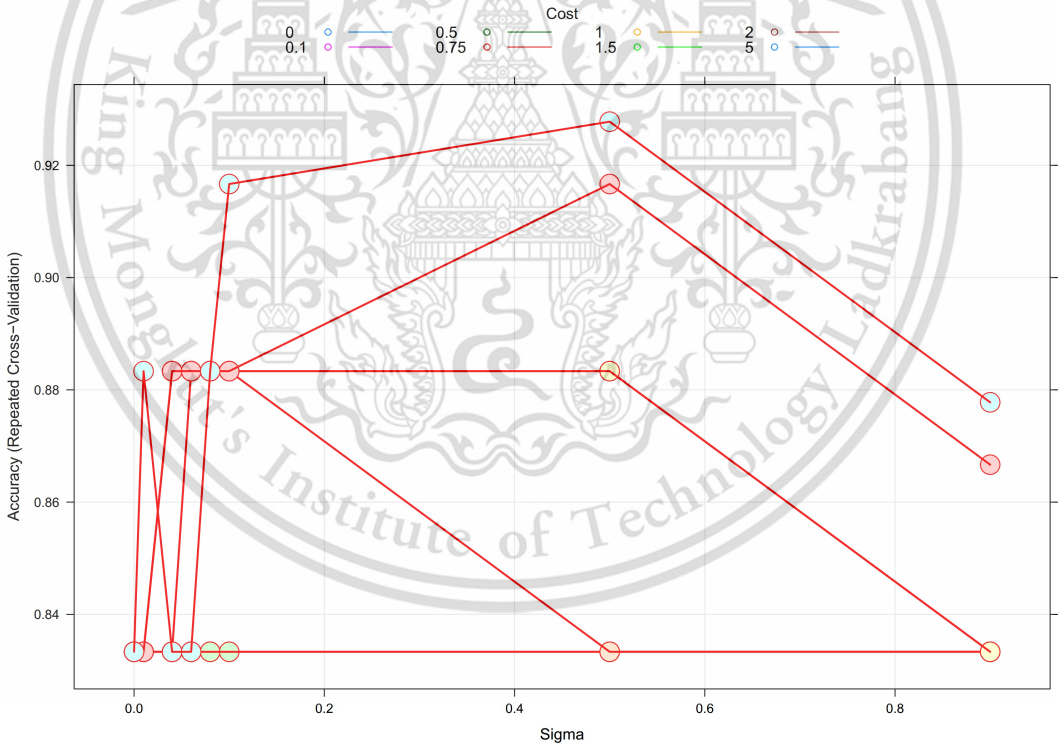

Figure 4.16*:* MSVM RBF Kernel.

The caret package is routinely employed for MSVM analysis using different kernels. This program demonstrates efficiency by effectively categorizing data to minimize errors and enhance accuracy. Figure 4.16 showcases the impact of tuning parameter sigma, which optimizes the model at a value of 0.2714456 with a corresponding C

value of 16, resulting in an impressive accuracy of 96%.

The MSVM with RBF kernel is a widely used machine learning technique for classification and regression tasks. Here is a comprehensive explanation of the MSVM RBF algorithm:

1. Introduction to MSVM with RBF Kernel: MSVM is a powerful supervised learning algorithm used for various tasks, including classification and regression. The RBF kernel, also known as the Gaussian kernel, is one of the most popular kernel functions used to transform data into a higher-dimensional space.

2. Kernel Trick and RBF Kernel: The kernel trick allows MSVM to work efficiently in high-dimensional spaces without explicitly transforming the data. The RBF kernel maps data into a feature space where the data points are represented as vectors with similarities calculated using the Gaussian function.

3. Gaussian Function and Similarity: The Gaussian function calculates the similarity between two data points based on their Euclidean distance. It assigns higher similarity to points that are closer to each other and lower similarity to points that are farther apart.

4. Kernel Function: The RBF kernel function is defined as $K(x, y) = \exp(-\gamma * ||x-y||^2)$, where x and y are input feature vectors, and γ (gamma) is a hyperparameter that controls the shape of the Gaussian function.

5. Gamma Parameter: The gamma parameter influences the spread of the Gaussian function. Smaller values make the kernel wider, leading to a smoother decision boundary, while larger values make the kernel narrower, resulting in a more complex decision boundary.

6. Maximizing Margin and Non-Linearity: Similar to other MSVMs, MSVM with RBF kernel aims to find the hyperplane that maximizes the margin between classes. The RBF kernel effectively introduces non-linearity into the decision boundary, allowing MSVM to capture complex relationships in the data.

7. Kernel Matrix: To operate in the higher-dimensional space, MSVM computes a kernel matrix that represents the pairwise similarities between data points using the RBF kernel function.

8. Support Vectors: Support vectors are the data points that lie on the margin or are misclassified. They play a crucial role in defining the optimal decision boundary.

9. Solving the Optimization Problem: MSVM with RBF kernel involves solving the dual optimization problem, similar to other MSVM variants. The goal is to find the optimal hyperplane that best separates classes while minimizing misclassification.

10. Classification Decision: Classification decisions are made based on the sign of the decision function, which incorporates the support vectors and their corresponding Lagrange multipliers.

11. Regularization and Overfitting: MSVM with RBF kernel includes the cost parameter C, which balances the trade-off between maximizing the margin and minimizing misclassification. Proper tuning of C and γ is crucial to prevent overfitting.

12. Advantages and Limitations: MSVM with RBF kernel is powerful in capturing complex, non-linear relationships in data. It performs well on a variety of datasets but requires careful hyperparameter tuning and can be computationally intensive.

13. Applications: MSVM with RBF kernel is widely used in various fields, including image classification, text categorization, and bioinformatics.

In summary, MSVM with Radial Basis Function (RBF) kernel is a versatile algorithm that employs the RBF kernel trick to transform data into a higher-dimensional space and create complex, non-linear decision boundaries. Careful selection of hyperparameters, particularly C and γ, is essential for achieving optimal performance with the MSVM RBF algorithm.

Histograms and stem-and-leaf plots are both valuable tools used to provide an overview of central tendency and symmetry metrics for observed data. For detailed RMSE (Root-Mean-Square Error) testing, the corresponding R code can be accessed through the reference [30].

Furthermore, Box and Whisker Plots, also known as BoxPlots, offer an alternative graphical representation that succinctly summarizes specific information about the distribution of observed data values. This plot comprises a "Box" and a "Whisker." BoxPlots, as demonstrated in Figure 6 of the Root-Mean-Square Error (RMSE), present a visual summary of sample distribution characteristics, including skewness, central tendency, and data spread. The outcomes for Mean Absolute Error (MAE), Root-Mean-Square Error (RMSE), and R-Squared are obtained and can be observed in Figure 4.17.

In essence, the program's usage of the caret package for MSVM analysis, coupled with various visualization techniques such as histograms, stem-and-leaf plots, and BoxPlots, contributes to a comprehensive assessment of the model's performance across different kernels and tuning parameters. This approach aids in understanding data distribution, accuracy, and error metrics, facilitating informed decision-making in data analysis.

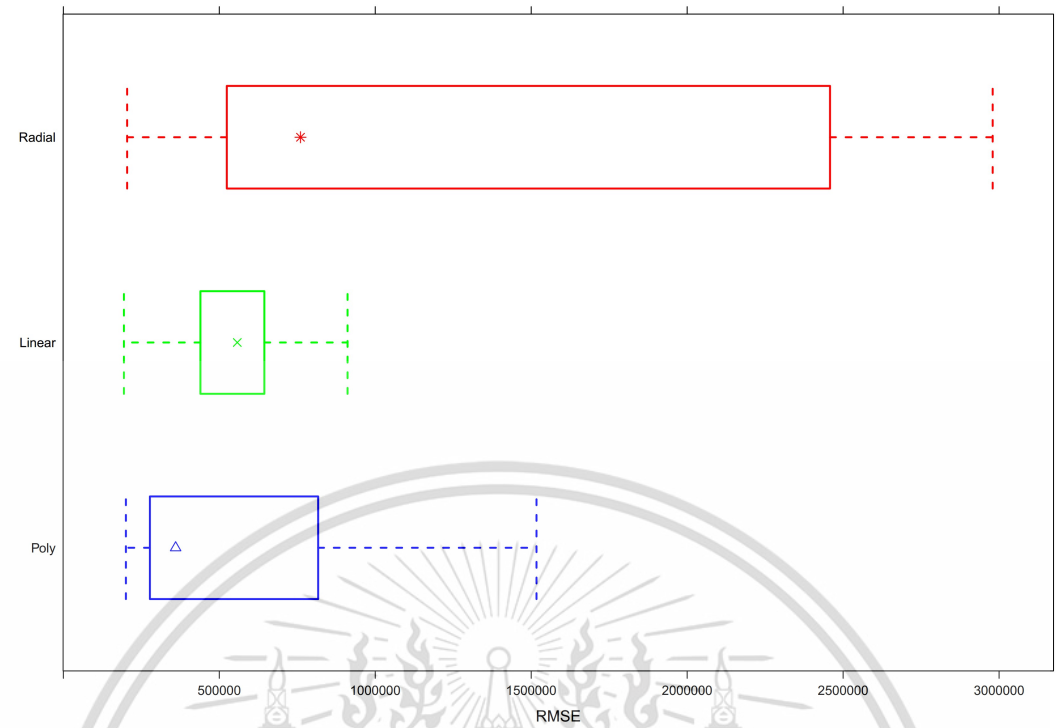

Figure 4.17*:* BloxPlot of RMSE for each of the Kernels.

The utilization of SVM classification with multiple related kernels is a strategic choice aimed at conducting thorough and comprehensive research while minimizing errors. SVM classification stands out as the optimal approach for delving deep into the current dataset, offering a robust foundation for analysis. The rigorous data validation and verification processes conducted through simulations serve to ensure prudence and accuracy in the research findings.

In the pursuit of further insights, additional avenues of analysis can be explored. One potential direction involves incorporating premium data, which can enhance the scope and richness of the analysis. Additionally, there is potential to refine and modify the categorization algorithm, fine-tuning it to capture subtle nuances within data and potentially improving the accuracy of results.

By adopting this approach and considering potential extensions, the research not only builds a solid foundation but also opens doors for future exploration and enhancement of the analysis. This holistic and iterative process contributes to the ongoing refinement and advancement of data analysis methodologies.

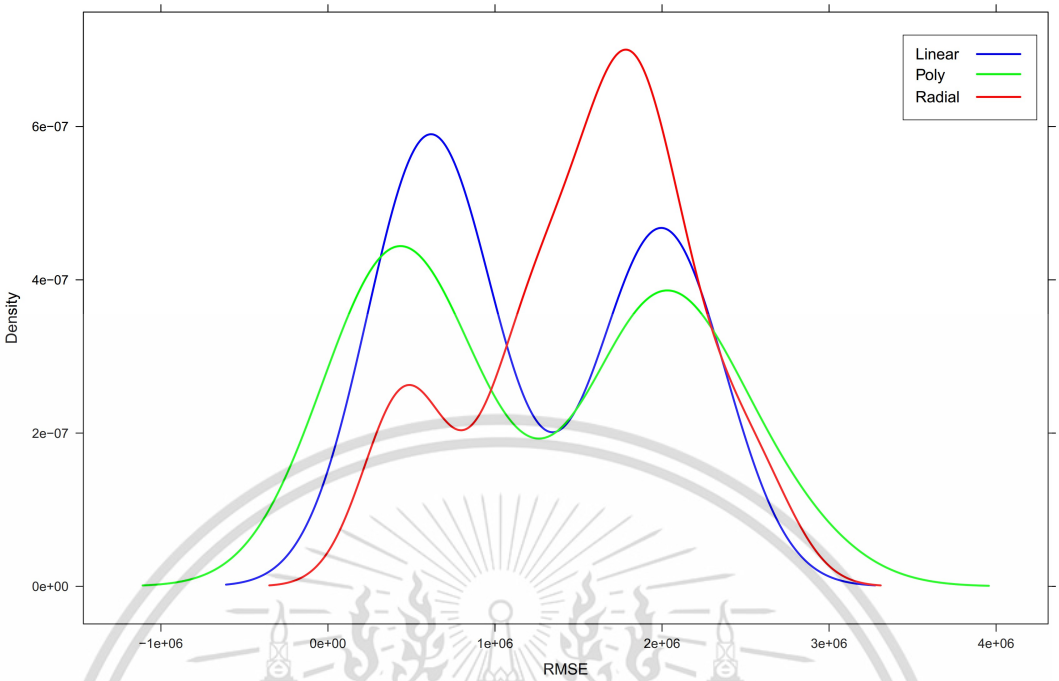

Figure 4.18*:* RMSE and Density for MSVM Kernel Comparisons.

Table 4.1*:* Sum Insured Level for Central Regions in Thailand

No	Central Province	Sum Insured Level	Recommendation
1	Bangkok	0	Pay More
2	Kanchanaburi	1	Pay Less
3	Chathaburi	1	Pay Less
4	Chachoengsao	1	Pay Less
5	Chonburi	0	Pay More
6	Chai Nat	1	Pay Less
7	Trat	1	Pay Less
8	Nakhon Nayok	1	Pay Less
9	Nakhon Pathom	1	Pay Less
10	Nonthaburi	0	Pay More
11	Pathum Thani	0	Pay More
12	Prachuap Khiri Khan	1	Pay Less
13	Prachin Buri	1	Pay Less
14	Phetchaburi	1	Pay Less
15	Ayutthaya	1	Pay Less
16	Rayong	1	Pay Less
17	Ratchaburi	1	Pay Less
18	Lop Buri	1	Pay Less
19	Samut Prakan	0	Pay More
20	Samut Songkhram	1	Pay Less
21	Samut Sakhon	1	Pay Less
22	Sa Kaeo	1	Pay Less
23	Saraburi	1	Pay Less
24	Sing Buri	1	Pay Less
25	Suphan Buri	1	Pay Less
26	Ang Thong	1	Pay Less

Source: Office of the Insurance Commission (OIC) in Thailand

Table 4.2*:* Sum Insured Level for North Regions in Thailand

No	Central Province	Sum Insured Level	Recommendation
1	Kamphaeng Phet	1	Pay Less
2	Chiang Rai	0	Pay More
3	Chiang Mai	0	Pay More
4	Tak	1	Pay Less
5	Nakhon Sawan	0	Pay More
6	Nan	1	Pay Less
7	Phetchabun	1	Pay Less
8	Phrae	1	Pay Less
9	Phyao	1	Pay Less
10	Phichit	1	Pay Less
11	Phitsanulok	0	Pay More
12	Mae Hong Son	1	Pay Less
13	Lampang	1	Pay Less
14	Lamphun	1	Pay Less
15	Sukhothai	1	Pay Less
16	Uttaradit	1	Pay Less
17	Uthai Thani	1	Pay Less

Source: Office of the Insurance Commission (OIC) in Thailand

Table 4.3*:* Sum Insured Level for Northeast Regions in Thailand

No	Central Province	Sum Insured Level	Recommendation
1	Kalasin	1	Pay Less
2	Khon Khaen	0	Pay More
3	Chaiyaphum	1	Pay Less
4	Nakhon Phanom	1	Pay Less
5	Nakhon Ratchasima	0	Pay More
6	Buriram	1	Pay Less
7	Maha Sarakham	1	Pay Less
8	Mukdahan	1	Pay Less
9	Yasothon	1	Pay Less
10	Roi Et	1	Pay Less
11	Loei	1	Pay Less
12	Si Sa Ket	1	Pay Less
13	Sakhon Nakhon	1	Pay Less
14	Surin	1	Pay Less
15	Nong Khai	1	Pay Less
16	Nong Bua Lam Phu	1	Pay Less
17	Amnat Charoen	1	Pay Less
18	Udon Thani	0	Pay More
19	Ubon Ratchathani	0	Pay More
20	Bueng Kan	1	Pay Less

Source: Office of the Insurance Commission (OIC) in Thailand

Table 4.4*:* Sum Insured Level for South Regions in Thailand

No	Central Province	Sum Insured Level	Recommendation
1	Krabi	1	Pay Less
2	Chumphon	1	Pay Less
3	Trang	1	Pay Less
4	Nakhon Si Thammarat	0	Pay More
5	Narathiwat	1	Pay Less
6	Pattani	1	Pay Less
7	Phangnga	1	Pay Less
8	Phatthalung	1	Pay Less
9	Phuket	0	Pay More
10	Yala	1	Pay Less
11	Ranong	1	Pay Less
12	Songkhla	0	Pay More
13	Satun	1	Pay Less
14	Surat Thani	0	Pay More

Source: Office of the Insurance Commission (OIC) in Thailand

Chapter 5

Conclusion and Future Work

5.1 Conclusion

A significant contribution of this research emerges in the context of existing literature, particularly within the context of Thailand. This innovative approach aligns well with the needs of Thai insurance companies and holds the potential to enhance service quality within the country. The study's outcomes propose a comprehensive framework for evaluating data prior to its simulation through the modified SVM with polynomial method. This framework offers professionals insights into the factors influencing the quality of insurance data services. Employing statistical techniques such as MPE, ROC, AUC, pAUC, smoothing, confidence intervals, and threshold analysis, the MSVM polynomial method accurately discerns insurance datasets. The simulations yield insightful answers that are invaluable for predicting risks associated with fallacious data processing, thereby aiding insurance companies in making informed decisions. Notably, the CSVM polynomial method lags significantly behind the MSVM polynomial method across multiple facets of data analysis. The research suggests the possibility of further exploration by applying various kernels to the MSVM polynomial method to identify the optimal kernel for classification. It is important to acknowledge that the study's limitations stem from its focus on the polynomial kernel and the application of random τ values. This research centers on data processing within the context of insurance in Thailand.

The creation of an accurate evaluating algorithm incorporating MPE, ROC, AUC, pAUC, smoothing, confidence intervals, and threshold analysis leads to a remarkable increase in accuracy, reaching 83%. This simulation enables the prediction of error risks in data processing, proving advantageous for corporations, governments, and researchers in refining data accuracy prior to decision-making. The study introduces a precise evaluating algorithm specifically tailored to data analysis using the MSVM RBF kernel method. This contribution holds substantial implications for the digital insurance sector and its data processing methodologies.

An error is pinpointed in the calculation of sum insured quantities for specific locations. In response, the MSVM method emerges as a more accurate and effective classification solution for sum insured, following a comprehensive exploration of the three prominent SVM kernels. The findings indicate that the MSVM approach should primarily utilize the radial kernel for future research. The key observation is that the radial kernel exhibits a low RMSE and a high density, resulting in data analysis outcomes closely approximating the likelihood of correctness within each zone. Tables

4.1 to 4.4 present data analysis outcomes for each region in Thailand. Notably, the decision to pay a higher or lower sum insured premium is influenced by an individual's address as indicated on their identity card. Even if someone works in a different area of Thailand, the contribution is determined by the original residential address. For instance, a student from Pattani, initially assigned to level 1, implies a small insurance coverage. If they relocate to Bangkok (level 0) for studies, the charged sum insured remains at level 1 due to the original Pattani address on their identity card. However, a permanent change of residential address would result in a level 0 sum insured charge based on the Bangkok area. This study provides a detailed and precise classification of insurance data, illuminating the imperfections still present in Thailand's data classification. It endeavors to introduce more effective and efficient methods for data clustering, as evidenced by the findings presented in this thesis. Future extensions of this thesis might encompass comparisons between the MSVM method and alternative classification approaches or the replacement of sum insured data with other insurance-related data, such as the analysis of insurance stock markets.

5.2 Future Work

Potential avenues for further research are highlighted. One avenue involves delving into data analysis using derived SVM models in fractional differential equations with varying orders. This approach can be applied to areas such as the stock market or the analysis of chaotic financial systems, drawing inspiration from chaos theory [5]. Another intriguing possibility is the application of the results from this thesis to categorize insurance data based on province regions in Thailand. This can involve exploring which regions will pay more or less for the sum insured or premium, contributing to a deeper understanding of regional insurance dynamics. Additionally, further investigation into identifying the optimal kernel for classifying insurance data using the MSVM polynomial method is suggested. While the current methods have shown statistical fitness, there is room for exploring alternative approaches. The implementation of the accurate evaluating algorithm for other types of datasets, such as stock market data or banking data, is considered. Expanding the research to evaluate the accuracy of the algorithm in classifying insurance data using both the MSVM and chaos theory approaches is also recommended. This expansion can potentially extend to applications in stock market research, where the algorithm can aid in predicting stock index fluctuations. Similarly, in geotechnical engineering, the algorithm can contribute to earthquake mitigation strategies. It can also find applications in forecasting and trading stock indices, tax forecasting systems, insurance marketing evaluation, and disease-related research. These extensions will demonstrate the versatility and potential impact of the accurate evaluating algorithm across diverse domains.

References

[1] Cappiello, A. 2020. "The Technological Disruption of Insurance Industry: A Review". *International Journal of Business and Social Science* 11(1).

[2] Pisoni, G. 2020. "Going digital: case study of an Italian insurance company". *Journal of Business Strategy* 42(2): 106-115.

[3] Eckert, C. and Osterrieder, K. 2020. "How digitalization affects insurance companies: overview and use cases of digital technologies". *Zeitschrift für die gesamte Versicherungswissenschaft* 109(5): 333-360.

[4] Riikkinen, M., Saarijärvi, H., Sarlin, P. and Lähteenmäki, I. 2018. "Using artificial intelligence to create value in insurance". *International Journal of Bank Marketing* 36(6): 1145-1168.

[5] Kabran, F.B. and Ünlü, K.D. 2020. "A two-step machine learning approach to predict S&P 500 bubbles". *Journal of Applied Statistics* 2776-2794.

[6] Yu, W. and Park, T. 2014. "AucPR: An AUC-based approach using penalized regression for disease prediction with high-dimensional omics data". *BMC Genomics* 15(S10).

[7] Yu, W., Kim, J.K. and Park, T. 2018. "Estimation of Area Under the ROC Curve under nonignorable verication bias". *Statistica Sinica*.

[8] Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J-C. and Müller, M. 2011. "pROC: an open-source package for R and S+ to analyze and compare ROC curves". *BMC Bioinformatics* 12(1).

[9] Walter, S.D. 2005. "The partial area under the summary ROC curve". *Statistics in Medicine* 24(13): 2025-2040.

[10] McClish, D.K. 1989. "Analyzing a Portion of the ROC Curve". *Medical Decision Making* 9(3): 190-195.

[11] Carrington, A.M., Fieguth, P.W., Qazi, H., Holzinger, A., Chen, H.H., Mayr, F. and Manuel, D.G. 2020. "A new concordant partial AUC and partial c statistic for imbalanced data in the evaluation of machine learning algorithms". *BMC Medical Informatics and Decision Making* 20(1).

[12] Ma, H., Bandos, A.I., Rockette, H.E. and Gur, D. 2013. "On use of partial area under the ROC curve for evaluation of diagnostic performance". *Statistics in Medicine* 32(20): 3449-3458.

[13] Irizarry, R.A. 2019. **Introduction to Data Science: Data Analysis and Prediction Algorithms with R**. CRC Press.

[14] DeLong, E.R., DeLong, D.M. and Clarke-Pearson, D.L. 1988. "Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach". *Biometrics* 44(3): 837.

[15] Fawcett, T. 2006. "An introduction to ROC analysis". *Pattern Recognition Letters* 27(8): 861-874.

[16] Jakobsen, J.C., Gluud, C., Winkel, P., Lange, T. and Wetterslev, J. 2014. "The thresholds for statistical and clinical significance – a five-step procedure for evaluation of intervention effects in randomised clinical trials". *BMC Medical Research Methodology* 14(1).

[17] Kist, M.J. and Silvestrini, R.T. 2015. "Incorporating Confidence Intervals on the Decision Threshold in Logistic Regression". *Quality and Reliability Engineering International* 32(5): 1769-1784.

[18] Steinwart I. and Christmann, A. 2008. **Support Vector Machines**. Springer Science & Business Media.

[19] Vapnik, V.N. 1999. **The Nature of Statistical Learning Theory**. Springer-Verlag.

[20] Cristianini, N. and Taylor J.S. 2000. **An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods**. Cambridge University Press.

[21] Cortes, C. and Vapnik, V. 1995. "Support-Vector Networks". *Machine Learning* 20: 273-297.

[22] Boser, B.E., Guyon, I.M. and Vapnik, V.N. 1992. "A Training Algorithm for Optimal Margin Classifiers". *ACM* 144-152.

[23] Cardona, T.A. and Cudney, E.A. 2019. "Predicting Student Retention Using Support Vector Machines". *Procedia Manufacturing* 39: 1827-1833.

[24] Ryabtseva, V. and Skomorokhov, A. 2020. "Critical power prediction using SVM algorithms". *Procedia Computer Science* 169: 198-202.

[25] Rizal, N.N.M., Hayder, G., Mnzool, M., Elnaim, B.M.E., Mohammed, A.O.Y. and Khayyat, M.M. 2020. "Comparison between Regression Models, Support Vector Machine (SVM), and Artificial Neural Network (ANN) in River Water Quality Prediction". *Processes* 10(1652): 1-11.

[26] Dutta, N., Tanwar, S., Patel, S.K. and Ghinea, G. 2022. "SVM-based Analysis for Predicting Success Rate of Interest Packets in Information Centric Networks". *Applied Artificial Intelligence* 36(1): 1562-1583.

[27] Naicker, N., Adeliyi, T. and Wing, J. 2020. "Linear Support Vector Machines for Prediction of Student Performance in School-Based Education". *Mathematical Problems in Engineering* 1-7.

[28] Tharwat, A. 2019. "Parameter investigation of support vector machine classifier with kernel functions". *Knowledge and Information Systems* 61: 1269-1302.

[29] Dolan, E.D. and Moré, J.J. 2002. "Benchmarking optimization soft-ware with performance profiles". *Mathematical Programming* 91(2): 201-213.

[30] Nurhidayat, I., Pimpunchat, B., Noeiaghdam, S. and Fernández-Gámiz, U. 2022. "Comparisons of SVM Kernels for Insurance Data Clustering". *Emerging Science Journal* 6(4): 866-880.

[31] Nurhidayat, I., Pimpunchat, B. and Klomsungcharoen, W. 2023. "More accurate simulation for insurance data based on a modified SVM polynomial method". *Journal of Intelligence & Fuzzy Systems* 44(6): 9129-9141.

[32] Nurhidayat, I. and Pimpunchat, B. 2023. "A comparative approach to SVM kernel functions via accurate evaluating algorithms". *Journal of Engineering Science & Technology* 18(4): 2078-2090.

[33] Rubio, A. and Villar, F. 2015. "Code profiling in R: A review of existing methods and an introduction to package GUIProfiler". *The R Journal* 7(2): 275-287.

[34] Mishra, S.K., Rajković, P., Samei, M.E., Chakraborty, S.K., Ram, B. and Kaabar, M.K. 2021. "A q-gradient descent algorithm with quasi-fejér convergence for unconstrained optimization problems". *Fractal and Fractional* 5(3): 110.

[35] Neculai, A. 2008. "An Unconstrained Optimization Test Functions Collection". *Adv. Model. Optim.* 10: 147-161.

[36] Sedkaoui, S. 2018. **Data Analytics and Big Data**. John Wiley & Sons.

[37] Qiao, W. 2021. "Prediction model of stock market based on chaos theory". *E3S Web of Conferences* 275, *EILCD*.

[38] Hu, M., Tang, Z., Xie, X. and Jiang, M. 2022. "Stock prediction and analysis based on support vector machine". *Frontiers in Business, Economics and Management* 5(2): 98-101.

[39] Soofi, A.S., Li, Z. and Hui, X. 2011. "Nonlinear interdependence of the Chinese stock markets". *Quantitative Finance* 1-27.

[40] Bulusu, A., Palle, A., Austin, G., Kainth, K., Saraf, M., Talukder, S. and McMahan, L. 2020. "Near future stock market forecasting based on chaos theory, sentiment analysis, and quantum computing". *Banking & Finance* 332-336.

[41] Ding, C., Bao, T.-Y. and Huang, H.-L. 2020. "Quantum-inspired support vector machine". *Journal of Latex Class Files* 14(8): 1-13.

[42] Ding, S., Hao, M., Cui, Z., Wang, Y., Hang, J. and Li, X. 2022. "Application of multi-SVM classifier and hybrid GSAPSO algorithm for fault diagnosis of electrical machine drive system". *ISA Transactions*.

[43] Babu, N.R. and Mohan, B.J. 2017. "Fault classification in power systems using EMD and SVM". *Ain Shams Engineering Journal* 8: 103-111.

[44] Ma, Z., Ye, C., Li, H. and Ma, W. 2018. "Applying support vector machines to predict building energy consumption in China". *Energy Procedia* 152: 780-786.

[45] Xiao, B., Zhu, H., Zhang, S., OuYang, Z., Wang, T. and Sarvazizi, S. 2022. "Gray-related support vector machine optimization strategy and its implementation in forecasting photovoltaic output power". *International Journal of Photoenergy* 1-9.

[46] Varela-Aldás, J., Toasa, R.M. and Egas, P.F.B. 2022. "Support vector machine binary classifiers of home presence using active power". *Designs* 6(6): 1-11.

[47] Ramadevi, B. and Bingi, K. 2022. "Chaotic time series forecasting approaches using machine learning techniques: A review". *Symmetry* 14(5), 1-43.

[48] Zanaty, E.A. and Afifi, A. 2011. "Support vector machines (SVMs) with universal kernels". *Applied Artificial Intelligence* 25(7): 575–589.

[49] James, G., Witten, D., Hastie, T. and Tibshirani, R. 2013. **An introduction to statistical learning**. Springer.

[50] Puenpatom, R.A. and Rosenman, R. 2008. "Efficiency of Thai provincial public hospitals during the introduction of universal health coverage using capitation". *Health Care Management Science* 11(4): 319–338.

[51] Kang, H., Zong, X., Wang, J. and Chen, H. 2023. "Binary gravity search algorithm and support vector machine for forecasting and trading stock indices". *International Review of Economics & Finance* 84: 507-526.

[52] Xin, Y. 2022. "Application of optimizaed support vector machine model in tax forecasting system". *Journal of Function Spaces* 1-10.

[53] Wu, X. and Liu, H. 2022. "Evaluation and selection of insurance marketing schemes driven by multisource big data". *Mathematical Problems in Engineering* 1-9.

[54] Wang, J., He, F. and Sun, S. 2023. "Construction of a new smooth support vector machine model and its application in heart disease diagnosis". *Plos One* 18(2): 1-14.

Author Biography

Name	Mr. Irfan Nurhidayat
Date of Birth	9th October 1994
Address	Jalan Kapur, Dusun 01, RT. 002, RW. 001, Desa Sutawangi, Kecamatan Jatiwangi, Kabupaten Majalengka 45454, Jawa Barat, Indonesia
Education	B.Sc. in Mathematics in 2016 Honors cum laude of GPA 3.65/4.00 with the best graduate Jenderal Soedirman University, Purwokerto, Indonesia M.S. in Mathematics in 2019 Honors cum laude of GPA 3.88/4.00 National Taiwan Normal University, Taipei City, Taiwan NTNU Scholarships in 2017-2019, Taiwan Ph.D. in Applied Mathematics in 2023 Graduated with Outstanding King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand KMITL Doctoral Scholarships in 2021-2024, Thailand
Publications	(with B. Pimpunchat, S. Noeiaghdam, and U. Fernandez-Gamiz) Comparisons of SVM Kernels for Insurance Data Clustering (Q1, Emerging Science Journal, Italy). (with B. Pimpunchat and W. Klomsungcharoen) More accurate simulation of insurance data based on a modified SVM polynomial method (Q2, Journal of Intelligence and Fuzzy Systems, The Netherlands). (with B. Pimpunchat) A comparative approach to SVM kernel functions via accurate evaluating algorithms (Q3, Journal of Engineering Science and Technology, Malaysia). (with P. Meeklueb and B. Pimpunchat)

SVM Classifications for Insurance Data Processing
Mathematics for Digital Economy, VIASM
On December 14, 2021, Hanoi, Vietnam.