ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

PENALIZED STABLE REGRESSION

M.Sc. THESIS

İrem SARIBAŞ

Department of Mathematics Engineering

Mathematics Engineering Programme

JUNE 2024

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

PENALIZED STABLE REGRESSION

M.Sc. THESIS

İrem SARIBAŞ
(509211209)

Department of Mathematics Engineering

Mathematics Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Gül İNAN

JUNE 2024

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

CEZALANDIRILMIŞ STABİL REGRESYON

YÜKSEK LİSANS TEZİ

İrem SARIBAŞ
(509211209)

Matematik Mühendisliği Anabilim Dalı

Matematik Mühendisliği Programı

Tez Danışmanı: Assoc. Prof. Dr. Gül İNAN

HAZİRAN 2024

İrem SARIBAŞ, a M.Sc. student of ITU Graduate School student ID 509211209 successfully defended the thesis entitled "PENALIZED STABLE REGRESSION", which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Gül İNAN** İstanbul Technical University	.....
Jury Members :	**Prof. Dr. Atabey KAYGUN** İstanbul Technical University	.....
	Prof. Dr. Özgür MARTİN Mimar Sinan Fine Arts University	.....

Date of Submission : **16 May 2024**
Date of Defense : **24 June 2024**

To my family,

FOREWORD

I would like to express my gratitude to my advisor Assoc. Prof. Dr. Gül İnan for her guidance and unwavering support throughout my study. Additionally, I extend my deepest thanks to my family and friends for their invaluable financial and moral support.

June 2024 İrem SARIBAŞ

TABLE OF CONTENTS

ABBREVIATIONS

BMI	**:** Body Mass Index
CV	**:** Cross-Validation
DBSCAN	**:** Density-Based Spatial Clustering of Applications with Noise
LASSO	**:** Least Absolute Shrinkage and Selection Operator
LQA	**:** Local Quadratic Approximation
MCP	**:** Minimax Concave Penalty
MSE	**:** Mean Squared Error
OLS	**:** Ordinary Least Squares
SCAD	**:** Smoothly Clipped Absolute Deviation

LIST OF TABLES

Page

LIST OF FIGURES

Page

PENALIZED STABLE REGRESSION

SUMMARY

In machine learning, the process of data splitting is critical for developing both accurate and consistent models. This process involves dividing the data into separate sets for training, validation, and testing. The training set enables the training of models, the validation set assists in selecting the best parameters, and the test data allows for the assessment of performance of the model in real-world scenarios. Various data splitting techniques exist, each serving specific characteristics of the data set and modeling objectives, such as one-time split and k-fold cross-validation. In the method of one-time split, the data set is randomly divided into two subsets at a predetermined ratio. In the method of k-fold cross-validation, the data set is randomly divided into k equal parts. The model is trained on $k-1$ parts, and the remaining part is used for validation or testing. This process is repeated such that each part is used for training exactly once.

Over-fitting is an issue in machine learning where a model learns the details and noise in the training data to an extent that adversely affects its performance on previously unseen data. Regularized regression methods play a crucial role in addressing the problem of over-fitting, especially for models that perform excellently on training data but fail on new and previously unseen data. Techniques such as Ridge regression, the Least Absolute Shrinkage and Selection Operator (LASSO), Smoothly Clipped Absolute Deviation (SCAD), and the Minimax Concave Penalty (MCP) hold significant places in enhancing model training. By penalizing the coefficients of features, these methods help reduce over-fitting, encouraging the development of simpler models because such models are more likely to generalize better to new data sets. The penalty implemented by Ridge regression is proportional to the sum of the squares of the coefficients, which reduces their effect while retaining all features in the model, but does not eliminate any feature completely. LASSO aims both to shrink regression coefficients and to remove insignificant features from the model. It employs the sum of the absolute values of the coefficients as the penalty term. This method zeroes out the coefficients of insignificant features, thereby automatically performing feature selection. SCAD applies a penalty similar to that of LASSO to small coefficients but avoids penalizing large coefficients, allowing the model to retain large coefficients that are significantly different from zero. MCP is a method developed to address variable selection in high-dimensional data, offering a non-convex penalty mechanism and promoting sparse solutions while penalizing large coefficient values with less bias, thus reducing the effect on large coefficients differently than Ridge.

In this thesis, we propose an optimization-based algorithmic data splitting method to effectively select training and validation sets. Our proposed method systematically

assigns data points to training or validation sets based on their contributions to the performance of the model. If the contribution of a data point to the performance of the model is high, it is placed in the training set; if low, in the validation set.

In this study, the proposed approach is tested on various regression models using penalties such as Ridge, LASSO, SCAD, and MCP. The approach is compared with traditional data splitting techniques like one-time split method and k-fold cross-validation, applied to two different data sets using various evaluation metrics. Each data splitting scenario has been repeated one thousand times to ensure the consistency of the results and to obtain statistically reliable outcomes. The evaluation metrics include the runtime, the average value of the regularization parameter λ, the standard deviation of the regularization parameter λ, errors in prediction for the validation, training, and test sets, average coefficients, and the standard deviation of the coefficients.

In the scenario of one-time split method, the data set is randomly divided such that 80% of the observations are in the training and validation set, and 20% are in the test set. Then, based on a predetermined ratio, the training and validation sets are further randomly split. Models are constructed using these sets, and performance is measured.

In the scenario of k-fold cross-validation, the data set is randomly divided so that 80% of the observations are in the training and validation set, and 20% are in the test set. The training and validation set is then is divided into k equal parts. $k-1$ parts are used for training while the remaining part is used for validation. This process is repeated k times, each time with a different part used as the validation set, and the performance of the models is measured.

In the scenario evaluating the optimization-based data splitting approach, the data set is randomly divided so that 80% of the observations are in the training and validation set, and 20% are in the test set. Then, considering the contribution of each data point to the performance of the model, the training and validation sets are split according to a predetermined ratio. Models are built using these sets, and their performance is measured.

The findings obtained from the tests conducted over the mentioned scenarios are as follows:

When evaluated in terms of the runtime, the proposed method, although requiring more time compared to the method of single random splitting, provides effective results in similar or less time when compared to k-fold cross-validation. This situation becomes more pronounced especially when working with large data sets or complex models. Our method optimizes data splitting processes, balancing the cost of time while maximizing the accuracy and performance of the model.

In terms of the average value of the regularization parameter λ, variability in λ values across different scenarios indicates that regularization methods such as Ridge and SCAD significantly impact the fit of the models to the data. In the case of LASSO, low λ values result in outcomes similar to those of unregularized regression models, suggesting a minimal impact of the regularization.

When evaluating the standard deviation of the regularization parameter λ, the proposed method reduces the standard deviations of λ values, ensuring more consistent data fit by the model. This reduction indicates an enhancement in the generalization ability of the model and a better fit to the data.

In terms of prediction errors (MSE) evaluated scenario-wise, the proposed method maintains consistency in MSE values across the validation, training, and test sets. Notably, tests conducted with both k-fold cross-validation and the proposed optimization approach enhance the generalization capacity of the model, offering the lowest MSE values.

The results demonstrate that the proposed optimization-based data splitting method can produce models with prediction errors comparable to, and in some cases more successful than, those developed using k-fold cross-validation. When compared in terms of computational cost, the optimization-based data splitting method appears to be more advantageous than the time spent on k-fold cross-validation. Furthermore, the models have exhibited significantly lower standard deviations in predictions, model coefficients, and hyperparameters. This indicates a marked increase in model stability and suggests that the proposed method can contribute to the development of more reliable and consistent machine learning models. These findings offer promising perspectives on the applicability and effectiveness of the method.

Keywords: Regularization, Ridge, LASSO, SCAD, MCP, Cross-Validation, Over-fitting, Data Splitting, Optimization.

CEZALANDIRILMIŞ STABİL REGRESYON

ÖZET

Makine öğrenmesinde veri bölme, hem doğru hem de uyumlu modeller geliştirmek için kritik bir süreçtir. Bu süreç, verileri eğitim, doğrulama ve test olmak üzere ayrı kümeler halinde bölmeyi içerir. Bu bölümlemede eğitim kümesi modellerin eğitilmesini sağlar, doğrulama kümesi en iyi parametrelerin seçilmesine yardımcı olur ve test verisi ise modelin gerçek dünya senaryolarında ne kadar iyi performans gösterdiğini değerlendirmeye olanak tanır. Tek seferlik rastgele bölme veya k-katlı çapraz doğrulama gibi çeşitli veri bölme teknikleri vardır, her biri farklı veri seti özelliklerine ve modelleme amaçlarına özel olarak hizmet eder. Tek seferlik rastgele bölme yönteminde veri seti belirlenen bir oranda rastgele iki kümeye ayrılır. k-katlı çapraz doğrulama yönteminde veri seti rastgele olarak k eşit parçaya ayrılır. Model, $k-1$ parça ile eğitilir ve kalan parça doğrulama veya test için kullanılır. Bu işlem, her bir parça bir kez eğitim dışında kalacak şekilde tekrarlanır.

Aşırı uyum, makine öğreniminde bir modelin eğitim verisi üzerindeki detayları ve gürültüyü modelin daha önceden görmediği veriler üzerindeki performansını olumsuz etkileyecek şekilde öğrenmesi problemidir. Düzenlileştirilmiş regresyon yöntemleri, özellikle eğitim verileri üzerinde mükemmel sonuçlar veren ancak yeni ve daha önce görülmemiş veriler üzerinde başarısız olan modellerin aşırı uyum sorununu çözmede kritik bir rol oynamaktadır. Ridge regresyonu, LASSO (En Az Mutlak Küçültme ve Seçme Operatörü), SCAD (Pürüzsüz Kırpılmış Mutlak Sapma) ve MCP (Minimax Konkav Ceza) gibi düzenlileştirme yöntemleri, model eğitimini geliştirmede önemli bir yere sahiptir. Özelliklerin katsayılarının cezalandırılması yoluyla, bu yöntemler, aşırı uyum sorununu azaltmaya yardımcı olur, daha basit modellerin geliştirilmesini teşvik ederler, çünkü bu tür modellerin yeni veri setlerine daha iyi genelleme yapma olasılığı daha yüksektir. Ridge regresyonunun uyguladığı ceza, katsayıların karelerinin toplamına orantılıdır. Bu, tüm özellikleri modelde tutarken onların etkilerini azaltır, fakat hiçbir özelliği tamamen sıfırlamaz. LASSO, hem regresyon katsayılarını küçültmeyi hem de önemsiz özellikleri modelden çıkarmayı hedefleyen bir yöntemdir. LASSO, katsayıların mutlak değerlerinin toplamını ceza terimi olarak kullanır. Bu yöntem, önemsiz özelliklerin katsayılarını sıfıra eşitleyerek onları modelden çıkarır ve böylece özellik seçimini otomatik olarak gerçekleştirmiş olur. SCAD, küçük katsayılara LASSO'ya benzer bir ceza uygularken, büyük katsayıları cezalandırmaktan kaçınır. Bu yöntem, modelin sıfırdan farklı büyük katsayılara sahip olmasını sağlar. MCP, yüksek boyutlu verilerdeki değişken seçimini ele almak için geliştirilmiş bir yöntemdir. Konveks olmayan bir ceza mekanizması sunar ve benzer şekilde LASSO gibi seyrek çözümleri teşvik ederken, büyük katsayı değerlerini az bir önyargı ile cezalandırır. Bu, özellikle çok değişkenli regresyon modellerinde Ridge'den farklı olarak büyük katsayılara olan etkiyi azaltır.

Bu tezde, eğitim ve doğrulama kümelerini etkin bir şekilde seçmek için optimizasyon tabanlı algoritmik bir veri bölme yöntemi önermekteyiz. Önerdiğimiz yöntem, veri noktalarını model performansına katkılarına göre sistematik olarak eğitim veya doğrulama kümelerine atar. Eğer veri noktasının model performansına katkısı yüksekse eğitim kümesinde, düşükse doğrulama kümesinde bulunması mantığı esas alınır.

Bu çalışmada, önerilen yaklaşım Ridge, LASSO, SCAD ve MCP regresyon cezalarını kullanarak çeşitli regresyon modelleri üzerinde test edilmiştir. Yaklaşım, çeşitli değerlendirme metrikleri kullanılarak iki farklı veri setinde uygulanan tek seferlik rastgele veri bölme ve k-katlı çapraz doğrulama gibi geleneksel veri bölme teknikleriyle karşılaştırılmıştır. Her bir veri bölme senaryosu, sonuçların tutarlılığını sağlamak ve istatistiksel olarak güvenilir sonuçlar elde etmek için 1000 kez tekrar çalıştırılmıştır. Değerlendirme metrikleri arasında çalışma süresi, düzenlileştirme hiperparametresi λ'nın ortalama değeri, düzenlileştirme hiperparametresi λ'nın standart sapması, doğrulama, eğitim ve test kümelerinin tahmin hatası, ortalama katsayılar ve katsayıların ortalama standart sapması yer almaktadır.

Tek seferlik rastgele veri bölme senaryosunda veri seti, gözlemlerin %80'i eğitim ve doğrulama kümesinde ve %20'si test kümesinde olacak şekilde rastgele bölünmüştür. Ardından, belirlenen bir orana göre eğitim ve doğrulama kümeleri yine rastgele gözlemler seçilerek ayrılmıştır. Bu kümeler kullanılarak model kurulmuş ve performans ölçülmüştür.

k-katlı çapraz doğrulama senaryosunda veri seti, gözlemlerin %80'i eğitim ve doğrulama kümesinde ve %20'si test kümesinde olacak şekilde rastgele bölünmüştür. Eğitim ve doğrulama kümeleri daha sonra k eşit parçaya bölünmüştür. $k-1$ parça eğitim için kullanılırken kalan 1 parça doğrulama için kullanılmıştır Bu şekilde her parça bir kez doğrulama kümesi olacak şekilde k defa model kurulmuştur ve performans ölçülmüştür.

Optimizasyon tabanlı veri bölme yaklaşımının değerlendirildiği senaryoda veri seti, gözlemlerin %80'i eğitim ve doğrulama kümesinde ve %20'si test kümesinde olacak şekilde rastgele bölünmüştür. Ardından, her bir veri noktasının model performansına katkısı göz önüne alınarak eğitim ve doğrulama kümeleri belirlenen bir orana göre ayrılmıştır. Bu kümeler kullanılarak model kurulmuş ve performansı ölçülmüştür.

Belirtilen senaryolar üzerinden yapılan testler sonucunda elde edilen bulgular aşağıdaki gibidir:

Çalışma süresi bakımından değerlendirildiğinde önerilen yöntem, tek seferlik rastgele bölme yöntemine kıyasla daha fazla zaman gerektirse de, k-katlı çapraz doğrulama ile karşılaştırıldığında benzer veya daha az sürede etkili sonuçlar sunmaktadır. Bu durum, özellikle büyük veri setleri veya karmaşık modellerle çalışıldığında daha belirgin hale gelmektedir. Yöntemimiz, veri bölme süreçlerini optimize ederek, zaman maliyetini dengelerken doğruluk ve model performansını maksimize etme potansiyeline sahiptir.

Düzenleme hiperparametresi λ'nın ortalaması bakımından değerlendirildiğinde; farklı senaryolarda, λ değerlerinin değişkenlik göstermesi, özellikle Ridge ve SCAD gibi düzenlileştirme yöntemlerinin, modellerin veriye uyumunda önemli bir etkisi olduğunu göstermiştir. LASSO'da ise düşük λ değerleri, düzenlemesiz regresyon

modellerine benzer sonuçlar elde edilmesine neden olmuştur, bu da düzenlemenin etkisinin minimal olduğunu göstermektedir.

Düzenleme hiperparametresi λ'nın standart sapması bakımından değerlendirildiğinde önerilen yöntem, λ değerlerinin standart sapmalarını azaltarak, modelin verilere daha tutarlı bir şekilde uyum sağlamasını sağlamıştır. Bu azalma, modelin genelleştirme yeteneğinin arttığını ve veriye daha iyi uyduğunu göstermektedir.

Tahmin hataları (MSE) bakımından senaryo bazında yapılan analizlerde, önerilen yöntemin doğrulama, eğitim ve test setleri üzerindeki MSE değerlerinde tutarlılık sağladığı görülmüştür. Özellikle, *k*-katlı çapraz doğrulama ve önerilen optimizasyon yaklaşımıyla yapılan testler, modelin genelleştirme kapasitesini artırarak en düşük MSE değerlerini sunmuştur.

Sonuçlar, önerilen optimizasyon tabanlı veri bölme yönteminin, *k*-katlı çapraz doğrulama ile kurulmuş modellere kıyasla benzer ve bazı durumlarda daha başarılı tahmin hataları ile modeller üretebildiğini göstermektedir. *k*-katlı çapraz doğrulama yönteminde harcanan süre ile optimizasyon tabanlı veri bölme yöntemi kıyaslandığında hesaplama maliyeti olarak optimizasyon tabanlı veri bölme yöntemi daha avantajlı görülmektedir. Bununla birlikte, modeller tahminlerde, model katsayılarında ve hiperparametrelerde önemli ölçüde daha düşük standart sapmalar sergilemiştir. Bu durum, modelin istikrarında belirgin bir artış olduğunu işaret etmekte ve önerilen yöntemin daha güvenilir ve tutarlı makine öğrenimi modellerinin geliştirilmesine katkı sağlayabileceğini öne sürmektedir. Bu bulgular, yöntemin uygulanabilirliği ve etkinliği açısından umut verici perspektifler sunmaktadır.

Anahtar Kelimeler: Düzenlileştirme, Ridge, LASSO, SCAD, MCP, Çapraz Doğrulama, Aşırı Uyum, Veri Bölme, Optimizasyon.

1. INTRODUCTION

Machine learning is transforming business processes across various sectors by accelerating data-driven decision-making processes. Especially in data-intensive industries such as finance, healthcare, and retail, machine learning models have become indispensable for analyzing complex relationships between data and predicting future trends. These models can process large data sets, identifying patterns and relationships beyond human analytical capabilities.

Data splitting is a fundamental method in machine learning projects for training and evaluating models. Separating data into training, validation, and testing sets allows for objective testing of the performance of the models on real-world data and measuring their ability to generalize. This process helps ensure that models avoid issues like over-fitting, thereby producing more reliable results.

On the other hand, regularization techniques play a critical role in addressing the over-fitting problem in regression models. Techniques such as Ridge and LASSO regularization help improve the performance of models on both training and unseen data by reducing model complexity and penalizing unnecessary parameters. These methods enhance the robustness and stability of machine learning models, increasing their applicability to broader data sets.

In conclusion, machine learning, with its advanced data analysis and modeling techniques, plays a significant role in shaping modern business processes. The influence of these technologies is expanding daily, penetrating more sectors and enhancing decision-support systems.

1.1 Purpose of Thesis

The aim of this thesis is to present an optimization-based algorithmic data splitting method that assigns data points to training and validation sets with the objective of enhancing model performance. This approach will be rigorously tested along

with various regularized regression models, specifically Ridge [1], Least Absolute Shrinkage and Selection Operator (LASSO) [2], Smoothly Clipped Absolute Deviation (SCAD) [3], Minimax Concave Penalty (MCP) [4]. The Ridge and and LASSO regression models are readily available in the Scikit-learn library [5] of Python [6]. However, due to the absence of SCAD and MCP models within the Scikit-learn suite, custom implementations have been created for comprehensive evaluation. The potential of the proposed method to improve model generalization and yield more consistent and reliable results will be explored. Furthermore, this method will be compared with existing data splitting techniques, such as one-time split and k-fold cross-validation, to assess its suitability for model selection and tuning processes.

1.2 Literature Review

In the rapidly evolving field of data science, the quest for more precise and stable predictive models has led researchers to explore beyond traditional methodologies.

Bertsimas and Paskov [7] argue that optimization techniques in the training of regression models can produce more consistent and reliable outcomes compared to traditional random sampling methods, presenting the advantages of these techniques through both theoretical and empirical studies. They compared optimization and randomization scenarios using regularized and unregularized regression models across 12 different data sets. It was observed that the optimization method offers a distinct advantage over the random method when using a single validation set. While the advantage of optimization over cross-validation is smaller, it still provides significant benefits.

Hoerl and Kennard [1] introduced Ridge regression, providing a solution to the multicollinearity problem in regression analysis by imposing a penalty that shrinks the coefficients towards zero, thereby reducing model complexity and helping to avoid over-fitting.

Tibshirani [2] developed the LASSO, which not only penalizes the magnitude of the coefficients but also enables variable selection by setting some coefficients to zero, thus enhancing interpretability and prediction accuracy of the model.

Fan and Li [3] are credited with the foundational work on the SCAD approach, which addresses some of the limitations found in the LASSO for variable selection in statistical models.

On the other hand, Zhang [4] proposed the MCP as an innovative alternative to both LASSO and SCAD, providing a different perspective on the variable selection issue with certain theoretical improvements.

Andy Jones [8] provides a detailed explanation of how the SCAD penalty and its derivative are defined, how the derivative of the penalty function is solved for different values of λ, and how this approach can be applied to regression problems. The use of local quadratic approximations as a general approach for fitting models penalized with SCAD, and how this approach can be integrated into the full least squares objective function that includes the SCAD penalty, is discussed. The article concludes that the SCAD penalty and its quadratic approximation can be solved in a manner similar to the Ridge regression problem, specifically showing how this approach can yield an approximate solution to the SCAD problem.

Similarly, in Statistical Foundations of Data Science book by Fan et al. [9] elucidate on the SCAD and MCP penalty functions, presenting the application of local quadratic approximations for their implementation within regression frameworks.

1.3 Structure of the Thesis

This thesis is structured into five chapters. In Chapter 1, the aim of the thesis is introduced, along with a literature review.

In Chapter 2, an introductory overview of the concept of machine learning commences. Various steps in data pre-processing, including feature scaling, management of dummy variables, and techniques such as logarithmic transformation, are addressed. Validation methods like the one-time split and k-fold cross-validation are explored. Regularized regression models with Ridge, LASSO, SCAD, and MCP penalty functions are examined.

In Chapter 3, a new data splitting algorithm, developed as an alternative to traditional random data splitting techniques, is introduced. The methodology detailing how

data points are systematically assigned to training and validation sets to optimize the performance of the model is presented.

In Chapter 4, two distinct data sets named Insurance and Credit are introduced, which are used to evaluate the proposed data splitting methodology. Detailed scenarios designed to assess the performance of various data splitting methods on regularized regression models including Ridge, LASSO, SCAD, and MCP are elaborately explained. These scenarios include one-time split, k-fold cross-validation, and optimization-based data splitting methods. Additionally, the effects of these methods on the regression models are evaluated using various performance metrics—computational runtime, the average and standard deviation of the regularization hyperparameter λ, prediction errors for validation, training, and test sets, and the average and average standard deviation of the coefficients. Each scenario is thoroughly examined using specific data splitting ratios, and the results are presented in a comparative format. Finally, in Chapter 5, the conclusion of our study is presented.

2. REVIEW OF MACHINE LEARNING METHODS

Before diving into the specifics of the thesis, an overview of machine learning is given. Subsequent sections delve into the critical steps in data pre-processing, including standardization, dummy variable creation, and logarithmic transformation. The chapter further explores various data splitting techniques such as one-time split method, k-fold cross-validation, and grid search. Additionally, the thesis provides insights into the estimation of regression coefficients using least squares methods, leading to a discussion on regularization methods such as Ridge, LASSO, SCAD, and MCP, and how these methods are applied in fitting models to data.

2.1 Machine Learning

Machine learning is an artificial intelligence field that aims to enable machines to make intelligent predictions by extracting knowledge from data, without being explicitly programmed. It discovers hidden patterns and relationships in the data through algorithms [10].

In most of the machine learning problems, data is presented in tabular form where each column represents a variable and each row represents a data point or observation. Among these variables, one is designated as the response variable, also known as the dependent variable in statistics, and referred to as a label. This variable is considered to be associated with other variables, termed independent variables, which collectively affect the outcome of the response variable.

A mathematical model is constructed using a subset of data known as training data. The model is trained to recognize and generalize the relationships identified within this training set to predict outcomes for the entire population. This process of generalization from training data to unseen situations is critical for applying machine learning in real-world scenarios.

After processing input data, a machine learning model produces predictions or responses based on the patterns and relationships it has learned. The model can make informed decisions, classify new data points, or provide insights into the specific tasks for which it was trained.

Machine learning is commonly categorized into two main types: "Supervised Learning" and "Unsupervised Learning".

2.1.1 Supervised learning algorithms

In this type of learning, the machine learning model is trained using labeled data sets where each input data point is paired with a corresponding output label. Throughout the training process, the model continuously adjusts its weights to closely fit the underlying patterns found in the labeled data. Subsequently, armed with this acquired knowledge, the model becomes capable of predicting outcomes for new, previously unseen input data by applying the learned patterns.

Supervised learning is frequently used for two primary types of tasks: classification and regression.

Commonly used methods for classification include logistic regression (illustrated in Figure 2.1), decision trees, support vector machines, *k*-nearest neighbors, random forests, naive bayes, and neural networks.

For regression, popular approaches are linear regression (illustrated in Figure 2.2), polynomial regression, Ridge regression, LASSO regression, elastic net, decision trees for regression, and support vector regression [12].

2.1.2 Unsupervised learning algorithms

In this type of learning, machine learning algorithms is used to extract meaning from unlabeled data sets. There are no associated output labels for input data during training process of the model; instead, it autonomously seeks to uncover inherent patterns or groupings in the data.

Unsupervised learning is frequently used for two primary types of tasks: clustering and dimensionality reduction.

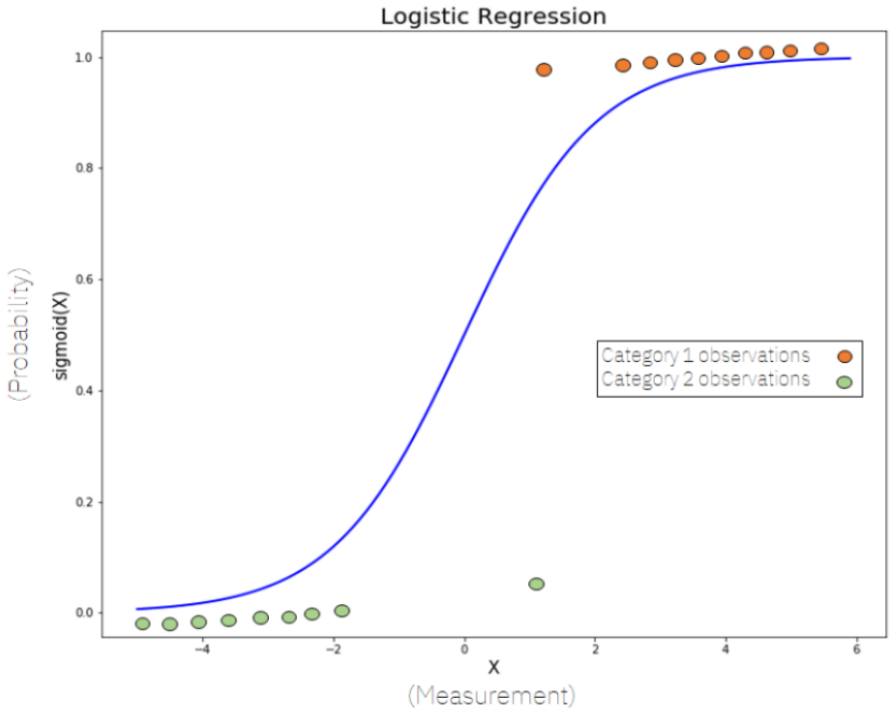

Figure 2.1 : An illustration of classification with logistic regression.

Source: [11].

Clustering tasks often utilize k-means (illustrated in Figure 2.3), hierarchical clustering, DBSCAN, Gaussian mixture models, and mean shift.

Dimension reduction is frequently achieved through principal component analysis, t-distributed stochastic neighbor embedding, auto-encoders, linear discriminant analysis, and singular value decomposition [12].

2.1.3 Mathematical optimization

Machine learning is an interdisciplinary field that integrates principles from statistics, data mining, and mathematical optimization. Specifically, mathematical optimization involves maximizing or minimizing an objective function $f(x)$ by systematically choosing input values from an allowed set and computing the value of the function. The general form of an optimization problem can be expressed as:

$$\begin{aligned} &\max_{x} \text{ (or } \min\text{)} \quad f(x) \\ &\text{subject to} \qquad g_i(x) \leq b_i,\ i = 1,\ldots,m, \qquad h_j(x) = 0,\ j = 1,\ldots,p, \end{aligned}$$

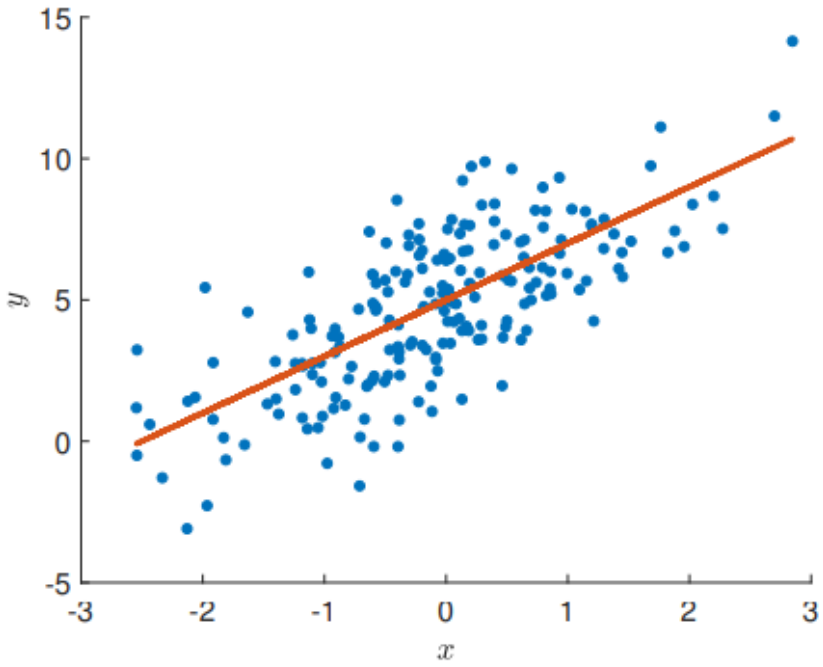

Figure 2.2 : An illustration of simple linear regression.

Source: [13].

where $g_i(x)$ represents the inequality constraints, and $h_j(x)$ represent the equality constraints that the solution x must satisfy. The variables b_i and c_j are the bounds of these constraints.

2.2 Feature Pre-processing

Before training machine learning models, feature pre-processing involves operations on raw data such as filling in missing values, converting categorical data to numerical format, scaling features, correcting noisy data, deriving new features, selecting important features, and transforming data. These processes help in efficiently processing the data set by machine learning algorithms and improving the overall performance of the model. Below, some of the feature pre-processing steps used in this study are described.

2.2.1 Standardization

Standardization involves rescaling all variables to similar scales by using the formula:

$$z_i = \frac{x_i - \mu}{\sigma},$$

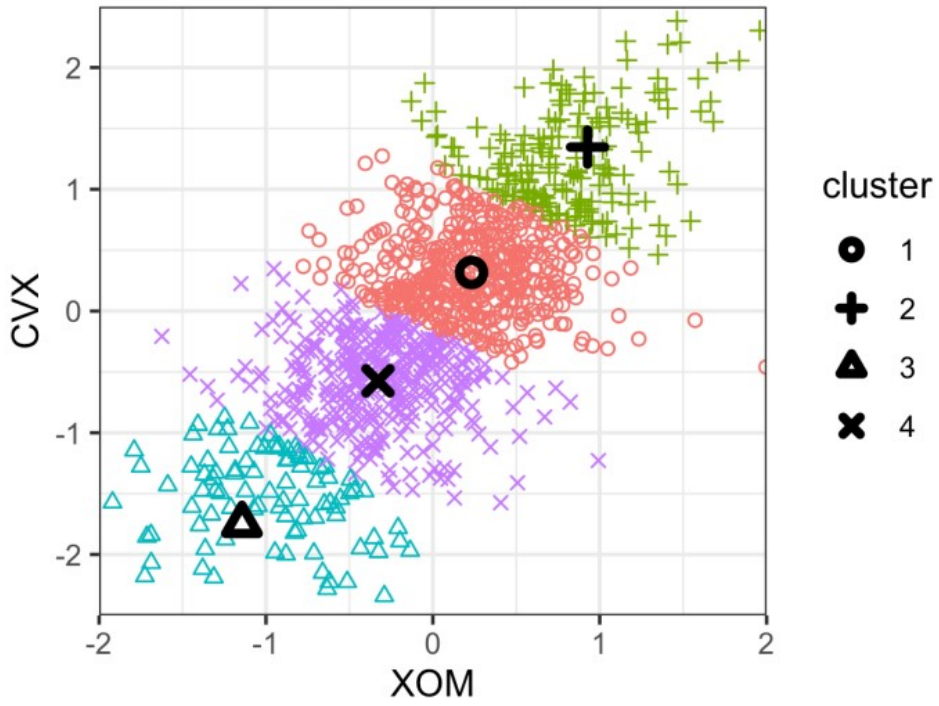

Figure 2.3 : An illustration of clustering with k-means algorithm.
Source: [14].

where x_i is the value of the ith observation, for $i = 1, \ldots, n$. The term μ denotes the mean of n observations for the variable being considered, the term σ denotes the standard deviation of n observations for the variable being considered, and z_i is the standardized value of the ith observation for the variable.

2.2.2 Dummy variable

Creating dummy variables is a technique for transforming categorical data into numerical form, which is frequently employed in statistical models that require numerical computations, such as regression analysis.

For a categorical variable comprising K distinct categories, one category is selected as the reference and $K - 1$ dummy variables are defined to prevent multicollinearity [14]. These dummy variables are capable of assuming either 0 or 1 as their values.

2.2.3 Logarithmic transformation

The logarithmic transformation involves applying the logarithm to the values of one or more numerical variables within a data set. This process is typically employed to

bring the data closer to a normal distribution, to diminish the influence of outliers, or to facilitate modeling when dealing with data that spans a wide range of values such as:

$$y' = \log_b(y),$$

where y' is the transformed value, y is the original value, and $\log_b$ denotes the base b of the logarithmic operation. In this study, the natural logarithm ln with base e (Euler's number) is used for data transformation.

For instance, in Figure 2.4, the original distribution of a variable from the data set used in our study is compared with its distribution after a logarithmic transformation.

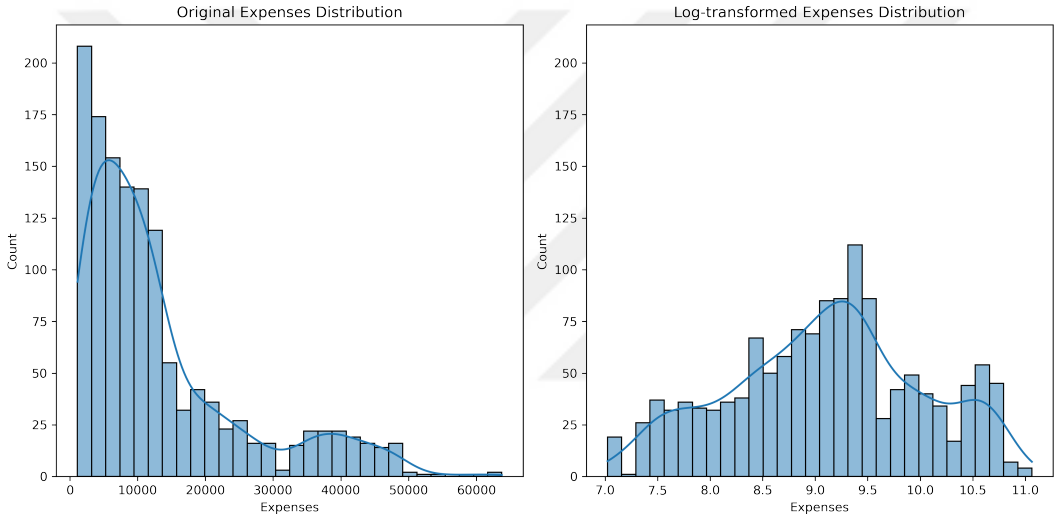

Figure 2.4 : An illustration of the impact of logarithmic transformation.

The leftmost graph illustrates the distribution of the variable in the original data set as being heavily left-skewed, with a significant sparsity observed at higher values. In the rightmost graph, following the logarithmic transformation, the range of values for the variable has been compressed, and the distribution shape has become more similar to a normal distribution.

2.3 Data Splitting Methods

In machine learning, data can be divided into three subsets: training, validation, and test sets. The training set is used to train models and allows the models to adjust their parameters to fit the data. The validation set is used during the training phase to

optimize the hyperparameters of the model and select the best version of the model. The test data set is then used to evaluate how well the model generalizes to new and previously unseen data. Data splitting methods are used to create random subsets of the data sets, which help assess and validate the generalization performance of the model. The most common data splitting techniques include the one-time split method and k-fold cross-validation.

An important aspect of model evaluation is understanding the prediction error, which is the differences between the actual values and the predictions made by the model. One commonly used metric to calculate this error is the Mean Squared Error (MSE).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

where y_i represents the actual values, $\hat{y}_i$ represents the values predicted by the model, and n is the number of observations.

Lower MSE values are desirable and indicate that the model fits the data better and provides highly accurate predictions. This criterion is frequently used in model selection and tuning to assess the generalization ability of a model.

2.3.1 One-time split method

The one-time split method is one of the simplest validation techniques used for assessing the performance of machine learning models. The data set is randomly split into a training set and a validation set at a specific ratio. The model is trained on the training set, and the validation set is used for tuning model hyperparameters and evaluating the generalization performance of the model. The test set, on the other hand, is used for the final check of the overall performance of the model. Figure 2.5 visualizes the one-time split method.

The one-time split method can lead to variability in model performance due to the way data is split, the potential omission of crucial data in training, and the limitation of data available for training.

Due to its simplicity and rapid applicability, the one-time split method is especially preferred for large data sets.

Figure 2.5 : An illustration of one-time split method.

2.3.2 k-fold cross-validation

Cross-validation is statistical technique for evaluating the generalization performance of a model, offering greater stability and comprehensiveness compared to a simple split into training and test sets.

In the k-fold cross-validation method, first a certain portion of the data is split for testing, which must be kept for final evaluations.

The remaining set is randomly divided into k smaller groups of approximately equal size and each group called as fold. A model is trained using $k-1$ of the folds as training data and the rest one is treated as a validation set. This process is repeated k times, each time a different fold is selected as validation data. At the end of each iteration, evaluation metrics, such as the mean squared error, are calculated and the average of these k values forms the mean squared error value reported for k-fold cross-validation.

$$CV_{(k)} = \frac{1}{k}\sum_{t=1}^{k} MSE_t,$$

where MSE_t is the Mean Squared Error for the tth fold in k-fold cross-validation. Figure 2.6 visualizes the 5-fold cross-validation method.

The choice of the fold number k in k-fold cross-validation significantly influences model performance. Selecting high k values results in models with lower bias and provides more reliable and stable performance measurements; however, this leads to higher computational costs. It is advisable to select an appropriate k value based on the size of the data set and available computational resources. Generally, the values 5 and 10 are preferred for k, as they provide a good compromise between computation time and model validation.

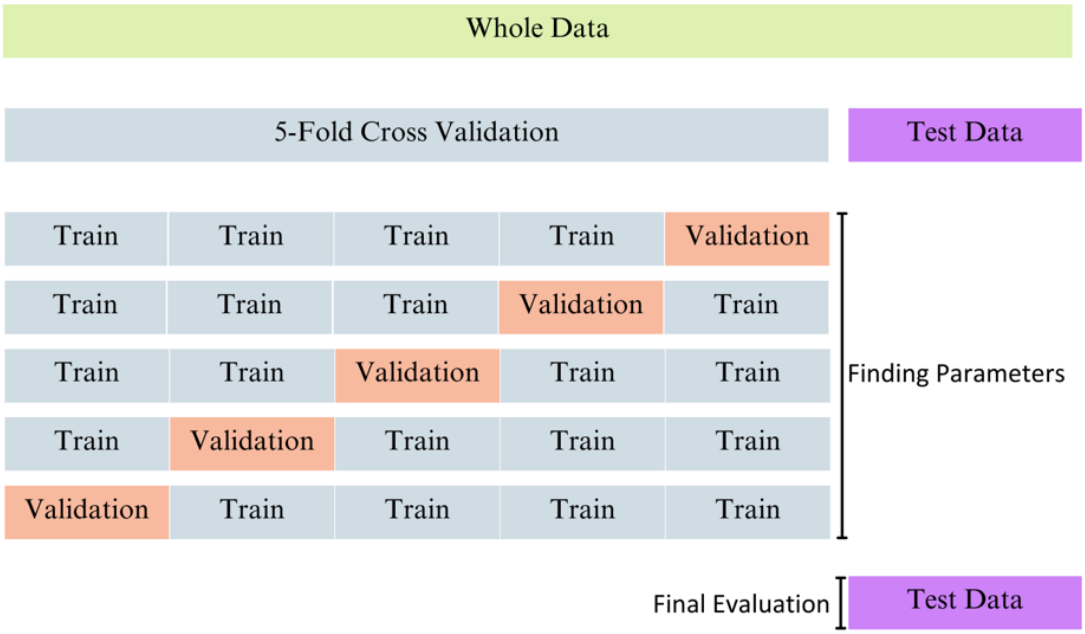

Figure 2.6 : An illustration of 5-fold cross-validation.

Source: [5].

2.3.3 Grid search

Grid search is an approach used for tuning hyperparameters of a model to find the combination that performs best. For example, in the context of this thesis where regularized regression models are employed, parameters such as the regularization strength λ could be explored. Grid search essentially tests every combination of these parameters using a validation data set to determine which configuration yields the highest performance [15].

Grid search is commonly paired with cross-validation. As a result, the term cross-validation is used to refer the process of performing grid search to optimize model parameters [15]. The diagram in Figure 2.7 illustrates the process of cross-validation used alongside grid search for hyperparameter tuning.

This process can be implemented using Python's `scikit-learn` library [5] `GridSearchCV()` function, while data splitting can be performed using either `ShuffleSplit(n_splits)` or `KFold(n_splits)` in `scikit-learn`.

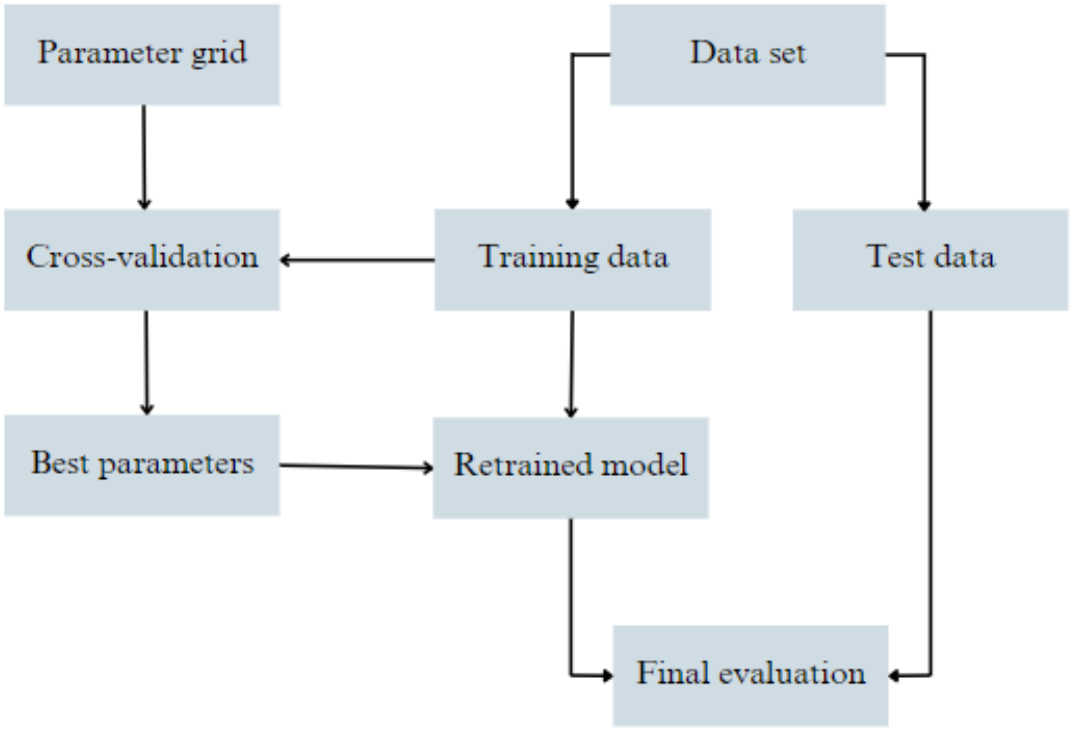

Figure 2.7 : An illustration of the workflow of cross-validation along with grid search.

Source: [5].

2.4 Linear Regression

Linear regression is a commonly used supervised learning tool for predicting quantitative responses. Many statistical learning approaches can be expressed as generalizations or extensions of linear regression.

It is assumed that the data set contains n observations, each characterized by p features and a response variable. The linear regression model is expressed as:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{2.1}$$

where $\mathbf{Y}$ is an $n \times 1$ dimensional column vector representing the response variable such as:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

the term $\mathbf{X}$ is an $n \times (p+1)$ dimensional matrix containing the features such as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_n}^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix},$$

the term $\boldsymbol{\beta}$ is a $(p+1) \times 1$ dimensional column vector representing the coefficients of the model:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \text{and}$$

the term $\boldsymbol{\varepsilon}$ is a $n \times 1$ dimensional column vector representing stochastic errors in the observations:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}.$$

Additionally, Equation 2.1 is commonly expressed at the individual level as follows:

$$\begin{aligned} y_i &= \mathbf{x_i}^T \boldsymbol{\beta} \\ &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \varepsilon_i \quad \text{for} \quad i = 1, \ldots, n, \end{aligned}$$

where β_0 is the intercept term, and $\beta_1, \beta_2, \ldots, \beta_p$ are the regression coefficients associated with the independent variables $x_1, x_2, \ldots, x_p$, respectively.

The coefficients $\beta_1, \beta_2, \ldots, \beta_p$ determine the strength and direction of the relationship between the dependent variable and corresponding independent variable in the regression model, with each coefficient reflecting the change in the dependent variable for a one-unit change in the corresponding independent variable, assuming all other variables remain constant.

The coefficients $\beta_1, \beta_2, \ldots, \beta_p$ are unknown and must be estimated. To obtain the coefficient estimates $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p$ such that the linear model 2.4 fits the data well, the least squares method is utilized.

2.4.1 Estimation of regression coefficients through ordinary least squares methods

A primary application of the ordinary least squares method (OLS) is in the estimation of regression coefficients. By adopting this method, the aim is to minimize the sum of squared errors:

$$\begin{aligned} \min_{\boldsymbol{\beta}} Q &= \min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \varepsilon_i^2 \\ &= \min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \mathbf{x_i}^T \boldsymbol{\beta})^2 \\ &= \min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \big(y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip})\big)^2, \end{aligned} \tag{2.2}$$

where Q represents the aggregate of squared differences between the observed values and those estimated by the model. In this formula, y_i denotes the observed value, x_{ij} represents the jth predictor of the ith observation, and $\beta_0, \beta_1, \ldots, \beta_p$ are the coefficients of the regression model that need to be estimated.

The goal of the least squares estimation method is to find the values of the regression coefficients that minimize Q, thereby ensuring that the predicted values are as close as possible to the actual observed values [16].

In matrix notation, Equation 2.2 can be written as:

$$Q = (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}).$$

The derivative of Q with respect to $\boldsymbol{\beta}$ (also called the gradient) is taken:

$$\frac{\partial Q}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^T\mathbf{Y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}. \tag{2.3}$$

Setting Equation 2.3 to 0 and solving it for $\boldsymbol{\beta}$ yields the estimation of the ordinary least squares regression coefficients $\hat{\boldsymbol{\beta}}$:

$$\begin{aligned} -2\mathbf{X}^T\mathbf{Y} + 2\mathbf{X}^T\mathbf{X}\hat{\beta} &= 0 \\ \mathbf{X}^T\mathbf{X}\hat{\beta} &= \mathbf{X}^T\mathbf{Y} \\ \hat{\boldsymbol{\beta}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \end{aligned} \tag{2.4}$$

Equation 2.5 in the linear regression model represents the predicted values of the dependent variable Y as calculated by the model:

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}}. \tag{2.5}$$

This expression shows how the model computes the predicted values of the dependent variable for each observation in the data set.

Furthermore, Equation 2.5 are often detailed at the individual level using the following formula:

$$\begin{aligned} \hat{y}_i &= \mathbf{x_i}^T \hat{\boldsymbol{\beta}} \\ &= \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \ldots + \hat{\beta}_p x_{ip} \quad \text{for} \quad i = 1, \ldots, n, \end{aligned}$$

where y_i denotes the predicted value of the dependent variable for the ith observation, $\hat{\beta}_0$ represents the intercept term (the value of the model when all independent variables are zero), and $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p$ correspondingly represent the coefficients indicating the effect of the independent variables $x_{i1}, x_{i2}, \ldots, x_{ip}$ on the dependent variable.

2.5 Regularization Methods

Over-fitting is a critical challenge in machine learning model training, leading to reduced model accuracy. It occurs when a model overly adapts to the noise in the training data, failing to generalize well to new, unseen data. To ensure effectiveness and reliability of a model, it is crucial to prevent over-fitting. Regularization techniques, which fit a model using all predictors while constraining the magnitude of coefficient estimates or shrinking them towards zero, offer a solution to mitigate over-fitting. This approach encourages simpler models that are more likely to generalize well [17].

The essence of shrinkage methods lies in minimizing the sum of squared differences between the observed values and the model predicted values, augmented by a penalty function. Coefficients are estimated in this manner.

Consider a general form of penalized least squares:

$$\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (y_i - \mathbf{x_i}^T \boldsymbol{\beta})^2 + \sum_{j=0}^{p} \Gamma_\lambda(\beta_j) \right), \tag{2.6}$$

where y_i represents the observed values, $\mathbf{x_i}$ is the vector of predictor variables, $\boldsymbol{\beta}$ is the vector of coefficients to be estimated, and n is the number of observations. The

function $\Gamma_\lambda(\beta_j)$ is a penalty applied to the coefficients, where λ controls the strength of this penalty.

The most commonly used regularization methods are Ridge, LASSO, and Elastic Net, which is a combination of the two. SCAD and MCP are also one of the regularization methods.

2.5.1 Ridge

Ridge regression, commonly known as ℓ_2 regularization, was first introduced by Arthur E. Hoerl and Robert W. Kennard in 1970 [1] with the following penalized objective function:

$$\hat{\boldsymbol{\beta}}_{Ridge} = \arg\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (y_i - \mathbf{x_i}^T \boldsymbol{\beta})^2 + \lambda \sum_{j=0}^{p} \beta_j^2 \right), \tag{2.7}$$

where $\lambda \geq 0$ is a tuning parameter, which controls the impact of shrinkage on the regression coefficient estimates. When $\lambda = 0$, the penalty term has no effect, and Ridge regression produces the same results as the least squares estimation. However, as $\lambda \to \infty$, the impact of the shrinkage penalty in Ridge regression increases, causing the coefficient estimates to approach zero, although they do not become exactly zero [17].

Solving Equation 2.7 for $\boldsymbol{\beta}$ yields the following closed-form solution:

$$\hat{\boldsymbol{\beta}}_{Ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y},$$

where $\hat{\boldsymbol{\beta}}_{Ridge}$ represents the estimated coefficients, $\mathbf{X}$ is the design matrix of input features, $\mathbf{Y}$ is the vector of target values, λ is the regularization parameter, and $\mathbf{I}$ is the $(p+1) \times (p+1)$ identity matrix.

The graph shown in Figure 2.8 represents the Ridge penalty function in relation to the coefficient values for Ridge regression. The U-shaped curve represents the penalty that escalates with the magnitude of the coefficient values; as coefficients move away from zero, the penalty increases quadratically.

2.5.2 Least absolute shrinkage and selection operator (LASSO)

LASSO regression, commonly known as ℓ_1 regularization, was introduced by Robert Tibshirani in 1996 [2] as a new method for variable selection and shrinkage in

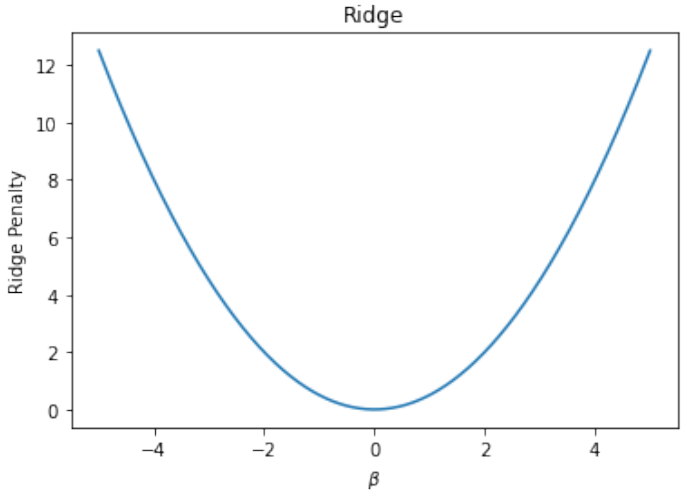

Figure 2.8 : Graph of Ridge penalty function.

regression with the following penalized objective function:

$$\hat{\boldsymbol{\beta}}_{LASSO} = \arg\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (y_i - \mathbf{x_i}^T \boldsymbol{\beta})^2 + \lambda \sum_{j=0}^{p} |\beta_j| \right), \tag{2.8}$$

where $\lambda \geq 0$ is a tuning parameter that controls the effect of shrinkage on the coefficient estimates similar to its role in Ridge regression. Unlike Ridge regression, LASSO has the effect of forcing some of the coefficient estimates to become exactly zero when the tuning parameter λ is sufficiently large. This feature of LASSO facilitates feature selection within the model [17].

There is no closed form expression as in the Ridge regression. Computing the LASSO solution is a quadratic programming problem [18].

The graph in Figure 2.9 is a visualization of the LASSO penalty as a function of regression coefficients (β). The V-shaped plot highlights the absolute value penalty that LASSO applies to coefficients. The point at which the lines meet at the bottom reflects the minimum penalty at zero, which is a key aspect of LASSO, differentiating it from Ridge regression that has a parabolic shape.

2.5.3 Smoothly clipped absolute deviation (SCAD)

The smoothly clipped absolute deviation (SCAD) penalty, proposed by Fan and Li in 2001 [3], was formulated to promote sparse solutions in least squares problems,

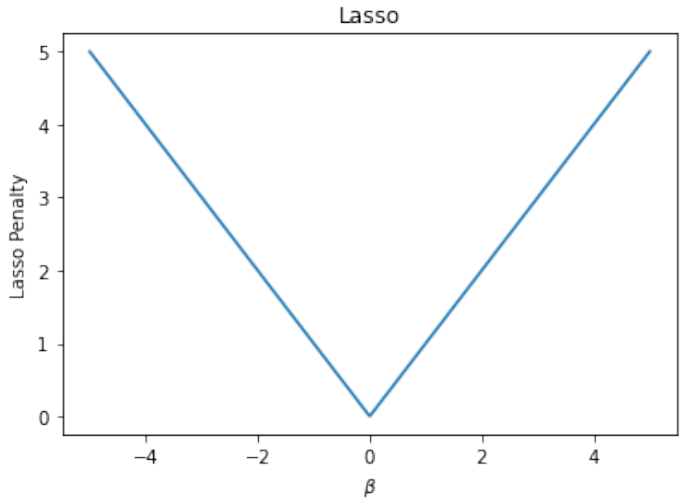

Figure 2.9 : Graph of LASSO penalty function.

while also permitting the inclusion of substantial β values. It is designed to overcome some of the limitations of LASSO and to prevent excessive shrinkage applied to large coefficients. Unlike LASSO, which shrinks all coefficients equally, SCAD applies a continuous non-linear penalty that varies based on the magnitude of the coefficients.

SCAD estimates the coefficients using the augmented objective function given below:

$$\hat{\boldsymbol{\beta}}_{SCAD} = \arg\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (y_i - \mathbf{x_i}^T \boldsymbol{\beta})^2 + \sum_{j=0}^{p} \Gamma_\lambda(\beta_j) \right), \tag{2.9}$$

where the penalty function $\Gamma_\lambda(\beta_j)$ for SCAD is defined as:

$$\Gamma_\lambda(\beta_j) = \begin{cases} \lambda|\beta_j| & \text{if } |\beta_j| \leq \lambda \\ \frac{2a\lambda|\beta_j| - \beta_j^2 - \lambda^2}{2(a-1)} & \text{if } \lambda < |\beta_j| \leq a\lambda \\ \frac{\lambda^2(a+1)}{2} & \text{if } |\beta_j| > a\lambda \end{cases}, \tag{2.10}$$

with $a > 2$ and $\lambda > 0$.

Curiously, the SCAD penalty is frequently delineated primarily through its first derivative, denoted as $\Gamma'(\beta)$, rather than $\Gamma(\beta)$ [8]:

$$\Gamma'_\lambda(\beta_j) = \begin{cases} \lambda sign(\beta_j) & \text{if } |\beta_j| \leq \lambda \\ \frac{a\lambda sign(\beta_j) - \beta_j}{a-1} & \text{if } \lambda < |\beta_j| \leq a\lambda \\ 0 & \text{if } |\beta_j| > a\lambda \end{cases}, \tag{2.11}$$

for $j = 0, 1, \ldots, p$, with $a > 2$ and $\lambda > 0$. The $sign()$ function is a mathematical function that returns the sign of a given number.

This is indicative of a quadratic spline function with knots at λ and $a\lambda$. This penalty function ensures that the value of β is not excessively penalized, contributing to the continuity of the solution.

Here, λ and a are unknown parameters. In practice, the cross-validation method can be employed to search for the optimal (λ, a) pair. Implementations of such procedures tend to be computationally expensive. Therefore, the value of a has been set to 3.7, as suggested in [3].

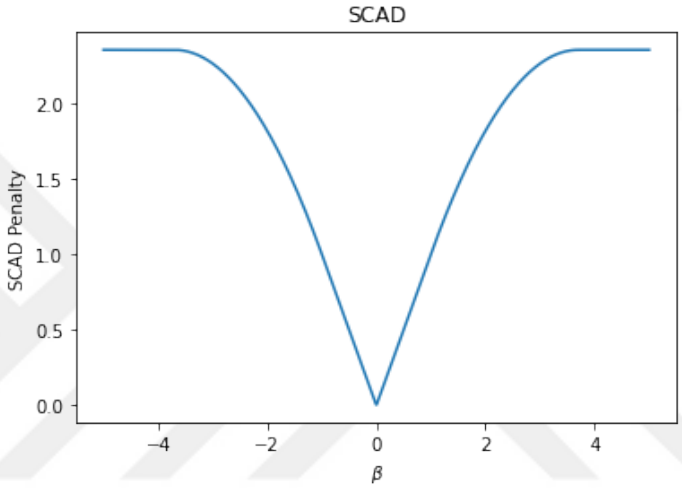

Figure 2.10 : Graph of SCAD penalty function with $a = 3.7$.

The graph presented in Figure 2.10 showcases relationship of the SCAD penalty function with the regression coefficients, denoted by β. As can be seen from the graph, it is evident that the SCAD penalty function, as defined in Equation 2.10, coincides with the LASSO penalty up to a certain threshold $|\beta| = \lambda$. After this point, it smoothly transitions to a quadratic function until approximately $|\beta| = a\lambda$, beyond which it stabilizes and remains constant for all $|\beta| > a\lambda$ [19].

2.5.4 Minimax concave penalty (MCP)

Minimax Concave Penalty (MCP), designed to handle variable selection in high-dimensional data, was introduced by Zhang in 2010 [4]. MCP offers a non-convex penalty mechanism which, akin to LASSO, promotes sparse solutions

by applying penalties to the coefficients, but does so with reduced bias on larger coefficient values, setting it apart from Ridge.

MCP estimates the coefficients by employing the augmented objective function detailed below:

$$\hat{\boldsymbol{\beta}}_{MCP} = \arg\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (y_i - \mathbf{x_i}^T \boldsymbol{\beta})^2 + \sum_{j=0}^{p} \Gamma_\lambda(\beta_j) \right), \tag{2.12}$$

where the penalty function $\Gamma_\lambda(\beta_j)$ for MCP is defined as:

$$\Gamma_\lambda(\beta_j) = \begin{cases} \lambda|\beta_j| - \frac{\beta_j^2}{2a} & \text{if } |\beta_j| \leq a\lambda \\ \frac{1}{2}a\lambda^2 & \text{if } |\beta_j| > a\lambda \end{cases}, \tag{2.13}$$

with $a \geq 1$ and $\lambda > 0$.

The MCP penalty is defined via its first derivative, usually denoted $\Gamma'(\beta_j)$ instead of $\Gamma(\beta_j)$, similar to the SCAD penalty.

$$\Gamma'_\lambda(\beta_j) = \begin{cases} (\lambda - \frac{|\beta_j|}{a} sign(\beta_j) & \text{if } |\beta_j| \leq a\lambda \\ 0 & \text{if } |\beta_j| > a\lambda \end{cases}, \tag{2.14}$$

for $j = 0, 1, \ldots, p$, with $a > 1$ and $\lambda > 0$. In this study, the value of a has been set to 3. The $sign()$ function is a mathematical function that returns the sign of a given number.

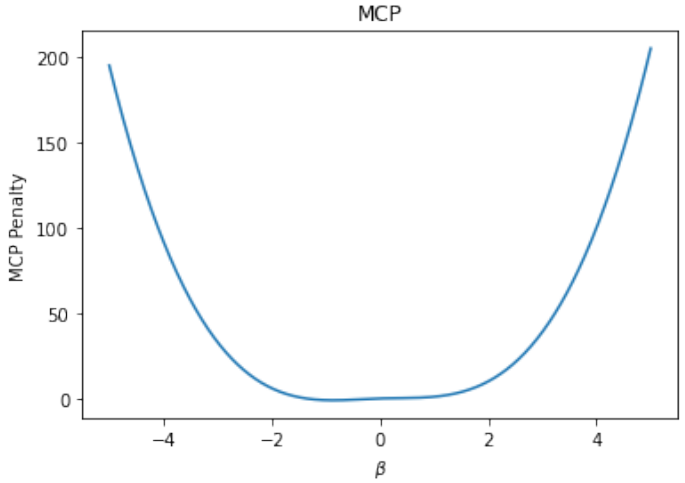

Figure 2.11 : Graph of MCP penalty function with $a = 1.5$.

The graph in Figure 2.11 illustrates the MCP function in relation to the regression coefficient values.

2.5.5 Fitting regression models with SCAD and MCP penalty functions

A common strategy for fitting penalized least squares models, which includes SCAD and MCP penalty functions, involves employing local quadratic approximations (LQAs) [8,9]. This method focuses on fitting a quadratic function $q(\beta_j)$ around an initial point β_{0j}, ensuring that the approximation is symmetric around 0, and that it meets the following two conditions: $q(\beta_{0j}) = \Gamma_\lambda(|\beta_{0j}|)$ and $q'(\beta_{0j}) = \Gamma'_\lambda(|\beta_{0j}|)$ for $j = 0, 1, \ldots, p$ [8,9].

This quadratic function is defined as:

$$q(\beta_j) = a + b\beta_j^2, \tag{2.15}$$

where a and b are coefficients, which are independent of β_j and $q'(\beta_j) = 2b\beta_j$ for $j = 0, 1, \ldots, p$. Assuming that this quadratic function form satisfies the conditions mentioned above, when $q(\beta_j)$ and $q'(\beta_j)$ are equated to $\Gamma_\lambda(|\beta_j|)$ and $\Gamma'_\lambda(|\beta_j|)$, respectively, at the initial point β_{0j}, the following two equations are obtained:

$$a + b\beta_{0j}^2 = \Gamma_\lambda(|\beta_{0j}|) \quad \text{and} \tag{2.16}$$

$$2b\beta_{0j} = \Gamma'_\lambda(|\beta_{0j}|). \tag{2.17}$$

The b value is obtained from Equation 2.17 as follows:

$$b = \frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|}. \tag{2.18}$$

When the b value in Equation 2.18 is substituted into Equation 2.16, the a value is obtained as follows:

$$a + \frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|}\beta_{0j}^2 = \Gamma_\lambda(|\beta_{0j}|) \Longrightarrow$$

$$a = \Gamma_\lambda(|\beta_{0j}|) - \frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|}\beta_{0j}^2.$$

These derived variables a and b can be substituted into the $q(\beta_j)$ quadratic approximation specified in Equation 2.15 given below:

$$\begin{aligned} q(\beta_j) &= \Gamma_\lambda(|\beta_{0j}|) - \frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|}\beta_{0j}^2 + \frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|}\beta_j^2 \\ &= \Gamma_\lambda(|\beta_{0j}|) + \frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|}(\beta_j^2 - \beta_{0j}^2). \end{aligned} \tag{2.19}$$

For any initial estimate of the coefficient value β_{0j}, we can construct a quadratic estimate of the penalty using $q(\beta_j)$ given in Equation 2.19. For instance, Figure 2.12 shows the local quadratic approximation function and the SCAD penalty function graphically.

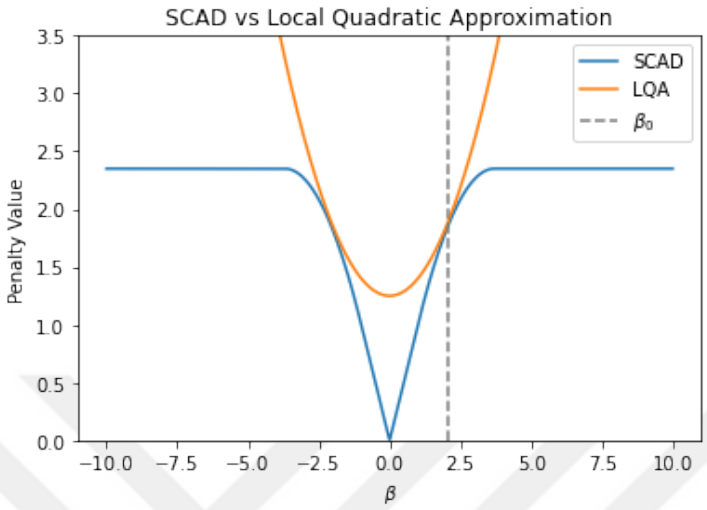

Figure 2.12 : Graph for comparison between SCAD penalty function (with $a = 3.7$) and its local quadratic approximation.

The augmented objective function is obtained by substituting the quadratic approximation function $q(\beta)$ in Equation 2.19 into $\Gamma_\lambda()$ in the penalized least squares objective function in Equation 2.6 as follows:

$$\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (y_i - \mathbf{x_i}^T \boldsymbol{\beta})^2 + \sum_{j=0}^{p} \left[\Gamma_\lambda(|\beta_{0j}|) + \frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|} (\beta_j^2 - \beta_{0j}^2) \right] \right).$$

When the terms that do not depend on β are ignored in this expression, the minimization function reduces to the following form:

$$\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (y_i - \mathbf{x_i}^T \boldsymbol{\beta})^2 + \sum_{j=0}^{p} \frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|} \beta_j^2 \right). \tag{2.20}$$

Equation 2.20 represents a Ridge regression problem, with the solution computed analytically as:

$$\hat{\boldsymbol{\beta}}_{SCAD/MCP} = \left(\mathbf{X}^T\mathbf{X} + \left(\text{diag} \left\{ \frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|} \right\} \right) \right)^{-1} \mathbf{X}^T\mathbf{Y},$$

where $\hat{\boldsymbol{\beta}}_{SCAD/MCP}$ represents the estimated coefficients under either SCAD or MCP penalty function, $\mathbf{X}$ is the design matrix of input features, $\mathbf{Y}$ is the vector of target values, *diag()* is a function that creates a diagonal matrix with the values of $\frac{\Gamma'_\lambda(|\beta_{0j}|)}{2|\beta_{0j}|}$ on its main diagonal, $\Gamma'_\lambda()$ is the derivative of the corresponding penalty function depending on the λ regularization parameter. Here, Equation 2.11 is employed for SCAD penalty and Equation 2.14 is applied for MCP penalty to compute these estimates [8,9].

In this research, classes of the Scikit-learn library for Ridge and LASSO were employed during the experiments conducted in Python. However, due to the absence of SCAD and MCP penalty functions in the Scikit-learn library, the method described above was utilized for estimating the coefficients.

3. PROPOSED DATA SPLITTING APPROACH

In this chapter, an optimization-based data splitting method for training and validation sets is presented for use with regularized regression models, including those with Ridge, LASSO, SCAD, and MCP penalty functions. This method is a departure from conventional random splitting techniques, offering a more systematic allocation of data points according to their impact on model performance.

This approach is motivated by the insights from the work of Bertsimas and Paskov on "Stable Regression on the Power of Optimization" [7], which highlights the robust capabilities of optimization in statistical modeling.

Bertsimas and Paskov [7] have advanced the optimization problem in their article by applying robust optimization techniques, transforming the LASSO problem into a linear optimization challenge and the Ridge problem into a convex optimization issue. They have resolved these using commercial optimization software, comparing their approach to typical randomization methods.

In our study, we iteratively solve this optimization method as outlined in Algorithm 1. Additionally, we include SCAD and MCP penalty functions to the comparisons of Ridge and LASSO penalties originally discussed in the article. By adopting this approach, we aim to significantly enhance the predictive accuracy of the models

3.1 The Alternating Optimization of an Objective Function

In this section, we will talk about the optimization-based approach we propose for training general regularized regression models. Ridge, LASSO, SCAD, and MCP regression models are special cases of the model presented below. Rather than randomly dividing the data into training and validation sets, we can do this selection solving the following optimization problem:

$$\arg\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} z_i (y_i - \mathbf{x}_i{}^T \boldsymbol{\beta})^2 + \sum_{j=0}^{p} \Gamma_\lambda(\beta_j) \right) \quad \text{with} \quad \mathcal{Z} = \left\{ z_i : \sum_{i=1}^{n} z_i = R, \quad z_i \in \{0,1\} \right\}, \tag{3.1}$$

where $\Gamma_\lambda()$ is the corresponding penalty function characterized by the λ regularization parameter. At the optimal solution of Equation 3.1, each data point is either part of the training set or the validation set, depending on the value of a binary variable z_i. If $z_i = 1$, the data point $(\mathbf{x}_i^T, y_i)$ is included in the training set; if $z_i = 0$, the data point is placed in the validation set. This binary variable effectively determines the allocation of each data point. The constraint $\sum_{i=1}^{n} z_i = R$ specifies that exactly R data points are selected for the training set.

3.1.1 Algorithm

The proposed computational solution to the optimization problem in Equation 3.1 is presented in the following algorithm.

Algorithm 1 Optimization-based Data Splitting for Regularized Regression Models.

Input: $\mathcal{D}^{Train+Validation} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{n}$, Rate = (1-R/n), $\mathcal{D}^{Train}_{Initial} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{n\times(1-\text{Rate})}$, $RegularizedRegression$, λ, $\hat{\boldsymbol{\beta}}_{Initial} = [\hat{\beta}_{Initial_0}, \hat{\beta}_{Initial_1}, ..., \hat{\beta}_{Initial_p}]^T$, MaxIterations.

Output: $\mathcal{D}^{Train} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{n\times(1-\text{Rate})}$, $\mathcal{D}^{Validation} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{n\times\text{Rate}}$.

$\mathcal{D}^{Train} \leftarrow \mathcal{D}^{Train}_{Initial}$
/* Determine the number of observations for training data set.*/
$n_{Train} \leftarrow n \times (1 - \text{Rate})$
$counter \leftarrow 0$
while $counter \leq$ MaxIterations **do**
    /* Computation of regularized regression estimates on $\mathcal{D}^{Train}$ */
    $\hat{\boldsymbol{\beta}} \leftarrow RegularizedRegression(\mathcal{D}^{Train}, \lambda)$ where $\hat{\boldsymbol{\beta}} = [\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p]^T$
    /* Calculate the tolerance criterion. */
    $tol\_cri \leftarrow \sum_{j=0}^{p} (\hat{\beta}_{Initial_j} - \hat{\beta}_j)^2$
    if $tol\_cri > 0.001$ **then**
        /* Calculate squared residuals. */
        $\mathbf{r} \leftarrow (\mathbf{Y}^{Train+Validation} - \mathbf{X}^{Train+Validation}\hat{\boldsymbol{\beta}})^2$
        /* Sort the set of residuals in ascending order. */
        $\mathbf{r}_{\mathbf{sorted}} \leftarrow sort(\mathbf{r})$
        Identify the indices of the first n_{Train} elements from $\mathbf{r}_{\mathbf{sorted}}$ to compose the training data set $\mathcal{D}^{Train}$ and the remaining elements form the validation data set $\mathcal{D}^{Validation}$.
        $counter \leftarrow counter + 1$

17: $\hat{\boldsymbol{\beta}}_{Initial} \leftarrow \hat{\boldsymbol{\beta}}$
18: **else**
19: **break**
20: **end if**
21: **end while**
22: **return** $\mathcal{D}^{Train}$ and $\mathcal{D}^{Validation}$.

The Algorithm 1 takes the following inputs:

- $\mathcal{D}^{Train+Validation}$: The portion of the whole data set reserved for the training and validation set, which consists of n data points,
- Rate: The proportion of the data set to include in the validation set,
- $\mathcal{D}^{Train}_{Initial}$: The initial training data set, split according to the Rate from the training and validation set, $\mathcal{D}^{Train+Validation}$,
- $\hat{\boldsymbol{\beta}}_{Initial}$: The preliminary estimated coefficients, derived from an ordinary least squares estimation as given in Equation 2.4 that has been applied to the $\mathcal{D}^{Train}_{Initial}$ data, serving as the starting point for the parameter optimization of the model,
- *RegularizedRegression*: A choice of a regularized regression model such as Ridge, LASSO, SCAD, or MCP,
- λ: A choice of regularization strength, and
- MaxIterations: The maximum number of iterations that the algorithm is permitted to execute.

The Algorithm 1 then proceeds to apply the following steps utilizing these specified inputs:

- It begins by updating the initial training data set ($\mathcal{D}^{Train}_{Initial}$) to serve as the training data ($\mathcal{D}^{Train}$).
- It calculates the number of observations (n_{Train}) will be in $\mathcal{D}^{Train}$ based on given validation set proportion (Rate),
- A $counter$ is initialized to zero, which will track the number of iterations performed.

- A **while** loop commences and continues as long as the *counter* is less or equal to the specified maximum number of iterations (MaxIterations). This ensures that the algorithm halts if the coefficients of the model do not converge after a certain number of iterations.

- Within the loop, the expression $RegularizedRegression(\mathcal{D}^{Train}, \lambda)$ solves the Equation 2.7 for Ridge regression, the Equation 2.8 for LASSO regression, the Equation 2.20 along with 2.11 for SCAD regression, and the Equation 2.20 along with 2.14 for MCP regression on the data set $\mathcal{D}^{Train}$ with given λ, and coefficients of the corresponding model, denoted by $\hat{\boldsymbol{\beta}}$, are obtained.

- The tolerance criterion tol_cri is calculated as the sum of the squared differences between the initial coefficients $\hat{\boldsymbol{\beta}}_{Initial}$ and the newly obtained coefficients $\hat{\boldsymbol{\beta}}$. This measures the extent of change in the coefficients during the last iteration.

- If the tolerance criterion (tol_cri) exceeds 0.001, this indicates a significant change in coefficients and we go as follows:

  - Residuals ($\mathbf{r}$) are calculated as the squared differences between the actual values ($\mathbf{Y}^{Train+Validation}$) and the predicted values obtained by multiplying coefficients of the model ($\mathbf{X}^{Train+Validation}$).
  - The expression $sort(\mathbf{r})$ sorts the residuals in ascending order and returns $\mathbf{r_{sorted}}$.
  - The indices of the first n_{Train} number of elements from the sorted residual list, $\mathbf{r_{sorted}}$, are used to identify the data points that will form the new training set, $\mathcal{D}^{\text{Train}}$.
  - Conversely, the indices of the remaining elements in $\mathbf{r_{sorted}}$ are used to form the validation set, $\mathcal{D}^{\text{Validation}}$.
  - This method assumes that observations with smaller residuals, which are indicative of a better fit by the model, should be included in the training set to potentially enhance model performance, while those with larger residuals are set aside for validation to assess the predictive ability of the model on more challenging cases.

- The *counter* is incremented and the initial coefficients ($\hat{\boldsymbol{\beta}}_{Initial}$) are updated to the latest coefficients ($\hat{\boldsymbol{\beta}}$).
- If the tolerance criterion (tol_cri) is not greater than 0.001, this indicates that coefficients of the model have stabilized and the loop is exited.
- Finally, the while loop continues until one of the stopping conditions is met: either the maximum number of iterations (MaxIterations) is reached or the change in the coefficients of the model falls below a certain level.
- As a result of all these steps, the outputs $\mathcal{D}^{Train}$ set for training and $\mathcal{D}^{Validation}$ set for validation purposes are obtained.

This adaptive approach ensures that the division of data into training and validation sets is not arbitrary but informed by predictive performance of the model, leading to more reliable and robust model outcomes. By continuously refining the data split based on the performance of the model, the algorithm provides a more nuanced and effective method for model validation compared to existing data splitting methods.

4. APPLICATION WITH DATA

This chapter begins by introducing two different data sets, namely Insurance and Credit, which are used to test the proposed data splitting methodology. It then outlines the scenarios used in the testing methodology to evaluate the performance of various data splitting methods on regularized regression models. The final part of this chapter presents comparative results of experiments applied to the Insurance and Credit data sets across various performance metrics.

4.1 Data Sets

This study was tested on two open source data sets named Insurance and Credit. The Insurance data set utilized in this research draws inspiration from the "Machine Learning with R" book [20]. This data set is accessible via the following directory: `https://github.com/stedy/Machine-Learning-with-R-datasets` [21]. The Credit data set, on the other hand, is sourced from the materials presented in "An Introduction to Statistical Learning" [17]. This data set is accessible via the following directory: `https://www.statlearning.com/resources-second-edition` [22].

The Insurance and Credit data sets are described in detail in Sections 4.1.1 and 4.1.2, respectively.

4.1.1 Insurance data set

Pricing policies is a critical aspect for insurance companies to ensure profitability. The annual total of the premiums set by the company for the policies it sells must exceed the amount spent on its insured clients. The Insurance data set has been utilized to estimate medical care expenses for a health insurance policy. The data set contains 1338 observations pertaining to individuals enrolled in an insurance plan. These observations include data on the patients' age, sex, BMI (Body Mass Index),

number of children, smoking status, region, and the total medical expenses charged to the insurance plan within a calendar year. The target variable in this data set is the total expenses incurred by each insured individual. There is no missing observations in the data set.

While the variable descriptions from the Insurance data set are given in Table 4.1, the first five observations of the data set are presented in Table 4.2.

Table 4.1 : Description of variables in the Insurance data set.

Variable	Type	Description
Age	Quantitative	Indicates the integer age of the primary beneficiary, excluding those over 64 who are typically covered by government insurance.
Sex	Qualitative	Specifies the policy holder's gender, categorized as male or female.
BMI	Quantitative	Represents the body mass index, a ratio of weight in kilograms to height in meters squared, indicating relative weight status.
Children	Quantitative	Counts the number of children or dependents covered by the insurance plan.
Smoker	Qualitative	Indicates with a yes or no whether the insured individual smokes tobacco.
Region	Qualitative	Designates the beneficiary's residence within the U.S., categorized into one of four regions: northeast, southeast, southwest, or northwest.
Expenses	Quantitative	Measures the total annual medical expenses charged to the insurance plan. This variable has been subjected to logarithmic transformation.

Source: [20].

Table 4.2 : The first 5 observations of the Insurance data set.

Age	Sex	BMI	Children	Smoker	Region	Expenses
19	Female	27.9	0	Yes	Southwest	16884.92
18	Male	33.8	1	No	Southwest	1725.55
28	Male	33.0	3	No	Southwest	4449.46
33	Male	22.7	0	No	Northwest	21984.47
32	Male	28.9	0	No	Northwest	3866.86

4.1.2 Credit data set

Understanding their customers' spending behaviors and credit utilization is of significant importance for banks. Such insights are vital for creating strategies that

balance risk management with customer satisfaction in the competitive credit market. The Credit data set is used for the purpose of analyzing customers' credit card usage and payment patterns. The data set contains 400 observations and records variables related to the credit card holders. The data encompasses the cardholder's income, authorized credit limit, credit rating score, number of credit cards, age, years of education, home ownership status, student status, marital status, geographic region of residence and average credit card balance. There is no missing observations in the data set.

While the variable descriptions from the Credit data set are given in Table 4.3, the first five observations of the data set are presented in Table 4.4.

Table 4.3 : Description of variables in the Credit data set.

Variable	Type	Description
Income	Quantitative	Represents the credit card holder's income in thousands of dollars as an integer.
Limit	Quantitative	The credit card holder's authorized credit limit as an integer.
Ratings	Quantitative	An integer reflecting the credit card holder's credit rating score.
Cards	Quantitative	The number of credit cards possessed by an individual as an integer.
Age	Quantitative	Denotes the credit card holder's age as an integer.
Education	Quantitative	Signifies the number of years of education the credit card holder has completed as an integer.
Own	Qualitative	Specifies whether the credit card holder owns a house, with possible values of yes or no.
Student	Qualitative	Indicates the student status of the individual, with potential responses being yes or no.
Married	Qualitative	Denotes the marital status of the credit card holder, categorized as yes or no.
Region	Qualitative	Categorizes the credit card holder's region of residence as East, West, or South.
Balance	Quantitative	Represents the average credit card debt per individual. This variable has been subjected to logarithmic transformation.

Source: [17].

Table 4.4 : The first 5 observations of the Credit data set.

Income	Limit	Rating	Cards	Age	Education	Own	Student	Married	Region	Balance
14.89	3606	283	2	34	11	No	No	Yes	South	333
106.03	6645	483	3	82	15	Yes	Yes	Yes	West	903
104.59	7075	514	4	71	11	No	No	No	West	580
148.92	9504	1681	3	36	11	Yes	No	No	West	964
55.88	4897	357	2	68	16	No	No	Yes	South	331

4.2 Testing Methodology

In this subsection, we introduce scenarios aimed at contrasting the optimization-based data splitting method detailed in Chapter 3 with the one-time split and k-fold cross-validation methods outlined in Sections 2.3.1 and 2.3.2, respectively. The scenarios are depicted below.

- Scenario 1 (One-time Split Method): In Scenario 1, the one-time split method is employed. It starts by allocating 20% of the data as the test set. The remaining data is then divided into training and validation sets using different ratios: 50%-50%, 60%-40%, 70%-30%, 80%-20%, and 90%-10%, respectively. Subsequently, the regularized regression models are trained using Ridge, LASSO, SCAD, and MCP penalty functions under each splitting ratio, resulting in twenty different sub-scenarios.

- Scenario 2 (k-fold Cross-validation): In Scenario 2, k-fold cross-validation is applied. Initially, 20% of the data is reserved for testing purposes. The remainder undergoes k-fold cross-validation with k values set to 5 and 10, and the regularized regression models are trained using the aforementioned penalty functions under each k, resulting in eighth different sub-scenarios.

- Scenario 3 (Optimization-based Approach): Scenario 3 employs the proposed optimization-based data splitting method. It reserves 20% as the test set and uses splitting ratios of 50%-50%, 60%-40%, 70%-30%, 80%-20%, and 90%-10% for dividing the training and validation sets. The models are trained similarly to Scenario 1, resulting in twenty different sub-scenarios.

Each of the forty-eight sub-scenarios described above is applied to the Insurance and Credit data sets described in Section 4.1, respectively. Furthermore, each sub-scenario is executed $M = 1000$ times with different random seeds to examine the robustness and accuracy of the predictions.

Let $\mathcal{D} = (\mathbf{x}_i^T, y_i)_{i=1}^{n'}$ be the general notation used to represent Insurance and Credit data sets, where n' is the total number of observations, $\mathbf{x}_i$ is the p-dimensional column vector of features of ith observation, y_i is the target value of ith observation. The term $\mathbf{Y} = (y_1, \ldots, y_{n'})^T$ is the column vector of target values. Hence, the outlines of the workflows in Scenarios 1-3 are given below.

4.2.1 Outline of the workflow in scenario 1

Step 1:

1. Dummy variables are created for categorical features in the data set.
2. Logarithmic transformation is applied to the target variable, $\mathbf{Y}$.
3. The initial coefficients $\hat{\boldsymbol{\beta}}_{Initial} = \left[\hat{\beta}_{Initial_0}, \hat{\beta}_{Initial_1}, \ldots, \hat{\beta}_{Initial_p}\right]^T$ are computed on $\mathcal{D}$ through Equation 2.4, which serve as the starting point for fitting the SCAD and MCP regularized regression models.

Step 2: For each validation splitting ratio in the array $[0.1, 0.2, 0.3, 0.4, 0.5]$, a loop begins.

Step 3: For each regularized regression model in the set of estimators, which includes Ridge, LASSO, SCAD, and MCP, a loop begins.

Step 4: The following steps are repeated 1000 times.

1. The data set $\mathcal{D} = (\mathbf{x}_i^T, y_i)_{i=1}^{n'}$ is randomly split into 80% for training and validation $\mathcal{D}^{Train+Validation} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{0.8n'}$ and 20% for testing $\mathcal{D}^{Test} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{0.2n'}$, where $n = 0.8n'$.
2. The numerical features are standardized.

3. A parameter grid for the regularization parameter λ is defined using a logarithmic scale from 10^{-3} to $10^{1.5}$, spanning 50 values to systematically search through a wide range of possible settings.

4. `ShuffleSplit(n_splits=1)` is employed to randomly divide $\mathcal{D}^{Train+Validation}$ into $\mathcal{D}^{Train} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{n\times(1-\text{Rate})}$ and $\mathcal{D}^{Validation} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{n\times\text{Rate}}$ based on the specified validation splitting ratio.

5. `GridSearchCV()`, as illustrated in Figure 2.7, is applied with the parameter grid given in Step 4.3 on $\mathcal{D}^{Train}$ obtained in Step 4.4 to train the model, to ascertain the optimal hyperparameter combination on $\mathcal{D}^{Validation}$ obtained in Step 4.4, and then return to regression coefficients obtained from re-trained best model on $\mathcal{D}^{Train+Validation}$ obtained in Step 4.4.

6. Finally, total runtime, optimum λ value, MSE on $\mathcal{D}^{Validation}$, MSE on $\mathcal{D}^{Train+Validation}$, and MSE on $\mathcal{D}^{Test}$ are recorded.

4.2.2 Outline of the workflow in scenario 2

Step 1:

1. Dummy variables are created for categorical features in the data set.

2. Logarithmic transformation is applied to the target variable, $\mathbf{Y}$.

3. The initial coefficients $\hat{\boldsymbol{\beta}}_{Initial} = \left[\hat{\beta}_{Initial_0}, \hat{\beta}_{Initial_1}, ..., \hat{\beta}_{Initial_p}\right]^T$ are computed on $\mathcal{D}$ through Equation 2.4, which serve as the starting point for fitting the SCAD and MCP regularized regression models.

Step 2: For each k-fold number in the array [5, 10], a loop begins.

Step 3: For each regularized regression model in the set of estimators, which includes Ridge, LASSO, SCAD, and MCP, a loop begins.

Step 4: The following steps are repeated 1000 times:

1. The data set $\mathcal{D} = (\mathbf{x}_i^T, y_i)_{i=1}^{n'}$ is randomly split into 80% for training and validation $\mathcal{D}^{Train+Validation} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{0.8n'}$ and 20% for testing $\mathcal{D}^{Test} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{0.2n'}$, where $n = 0.8n'$.

2. The numerical features are standardized.

3. A parameter grid for the regularization parameter λ is defined using a logarithmic scale from 10^{-3} to $10^{1.5}$, spanning 50 values to systematically search through a wide range of possible settings.

4. k-fold cross-validation is utilized to divide $\mathcal{D}^{Train+Validation}$ into k equally sized folds, ensuring that each fold is used as a validation set once while the remaining folds form the training set. This process is implemented by applying `KFold(n_splits = k)` to ensure the data is split appropriately.

5. `GridSearchCV()`, as illustrated in Figure 2.7, is applied with the parameter grid given in Step 4.3 to ascertain the optimal hyperparameter combination, and then return to regression coefficients obtained from re-trained best model on $\mathcal{D}^{Train+Validation}$ obtained in Step 4.4.

6. Finally, total runtime, optimum λ value, cross-validated MSE on $\mathcal{D}^{Validation}$, MSE on $\mathcal{D}^{Train+Validation}$, and MSE on $\mathcal{D}^{Test}$ are recorded.

4.2.3 Outline of the workflow in scenario 3

Step 1: For each regularized regression model in the set of estimators, which includes Ridge, LASSO, SCAD, and MCP, the following steps are implemented.

1. Dummy variables are created for categorical features in the data set.

2. Logarithmic transformation is applied to the target variable, $\mathbf{Y}$.

Step 2: For each validation splitting ratio in the array $[0.1, 0.2, 0.3, 0.4, 0.5]$, a loop begins.

Step 3: The following steps are repeated 1000 times.

1. The data set $\mathcal{D} = (\mathbf{x}_i^T, y_i)_{i=1}^{n'}$ is randomly split into 80% for training and validation $\mathcal{D}^{Train+Validation} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{0.8n'}$ and 20% for testing $\mathcal{D}^{Test} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{0.2n'}$, where $n = 0.8n'$.

2. The numerical features are standardized.

3. The initial coefficients $\hat{\boldsymbol{\beta}}_{Initial} = [\hat{\beta}_{Initial_0}, \hat{\beta}_{Initial_1}, ..., \hat{\beta}_{Initial_p}]^T$ are computed on $\mathcal{D}^{Train+Validation}$ through Equation 2.4, which serve as the starting point for optimization-based data splitting method and fitting the SCAD and MCP regularized regression models.

4. Set the best score to a high number to retain the lowest MSE discovered during the hyperparameter tuning phase.

Step 4: A parameter grid for the regularization parameter λ is defined using a logarithmic scale from 10^{-3} to $10^{1.5}$, spanning 50 values to systematically search through a wide range of possible settings.

1. Using optimization-based data splitting Algorithm 1, split $\mathcal{D}^{Train+Validation}$ into $\mathcal{D}^{Train} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{0.8n\times(1-\text{Rate})}$, and $\mathcal{D}^{Validation} = \{(\mathbf{x}_i^T, y_i)\}_{i=1}^{0.8n\times\text{Rate}}$.

2. For each λ value, fit the model using $\mathcal{D}^{Train}$ and appraise the MSE on $\mathcal{D}^{Validation}$. Update the best score and corresponding parameters should the MSE prove to be inferior to the current best score.

Step 5:

1. Re-train the model with the identified best λ parameter upon the set $\mathcal{D}^{Train+Validation}$ obtained in Step 3.1 once the hyperparameter tuning concludes.

2. Finally, total runtime, optimum λ value, cross-validated MSE on $\mathcal{D}^{Validation}$, MSE on $\mathcal{D}^{Train+Validation}$, and MSE on $\mathcal{D}^{Test}$ are recorded.

The results are detailed in Section 4.3.

4.3 Results

In this subsection, the predictions over $M = 1000$ repetitions of each sub-scenario under Scenario 1, under Scenario 2, and under Scenario 3 for Insurance and Credit data sets are summarized using eight different performance metrics: computational runtime, average value of the regularization hyperparameter λ, standard deviation of the regularization hyperparameter λ, prediction error of the validation, training, and test sets, grand average of coefficients, and grand average of standard deviation of the coefficients.

4.3.1 Computational runtime

Considering the importance of computational costs in evaluating model performance, the measurement of runtime is critical. The runtime of the model training process for a given sub-scenario is calculated by recording the time consumed in Step 4.5 under Scenario 1, Step 4.5 under Scenario 2, and Steps 4 - 5.1 under Scenario 3. This is accomplished utilizing Python's `time()` function. Subsequently, these individual runtimes are then averaged over $M = 1000$ repetitions accordingly to provide a comprehensive measure of the computational cost associated with the model training under a given sub-scenario.

Let T_m represent the runtime (in seconds) for the mth repetition of the specified steps under a given sub-scenario. For a total of $M = 1000$ repetitions, the average runtime $\overline{T}$ (in seconds) is calculated as:

$$\overline{T} = \frac{1}{M} \sum_{m=1}^{M} T_m,$$

where $\overline{T}$ represents the average time taken to execute the specified steps across all repetitions of that given sub-scenario, providing a measure of the computational cost for the model training process. Average runtime results over 1000 repetitions across forty-eight sub-scenario for each Insurance and Credit data sets are given in Table 4.5 and illustrated in Figures A.1 - A.6.

As expected, Scenario 1, which represents the one-time split method, consumes less time compared to Scenario 3, which represents the proposed optimization-based data

splitting approach. Scenario 2, which involves k-fold cross-validation, increases the runtime proportionally with the value of k, as it requires splitting the data and fitting the model k times, making it the most time-consuming scenario.

In the Insurance data set, for Scenario 1, an increase in the size of the training set leads to an increase in runtime.

The impact of the utilized penalty function on runtime has not been observed.

It should be noted that the complexity of the model and the size of the data can also influence the runtime, especially in scenarios with large data sets or complex models. While the optimized split is expected to yield more accurate results, this increase in accuracy needs to be balanced against the time cost. The choice of the k value in cross-validation can increase computational time but provides a more accurate assessment of the model.

Table 4.5 : Average runtime (in seconds) value over 1000 repetitions.

Data Set	Regularization Method	Scenario 1					Scenario 2		Scenario 3				
		Train/Validation Split Ratio					Number of Folds		Train/Validation Split Ratio				
		50/50	60/40	70/30	80/20	90/10	5	10	50/50	60/40	70/30	80/20	90/10
Insurance	Ridge	1.1643	1.2536	1.3767	1.8108	1.2516	7.7277	13.8188	1.9716	2.2515	3.4591	3.5365	3.8714
	LASSO	1.0263	1.0229	1.0517	1.0906	1.0390	5.4021	14.3813	2.0214	2.2993	3.2053	3.3978	3.1753
	SCAD	1.1160	1.1235	1.2157	1.3398	1.8996	6.8921	16.7354	2.0676	1.6427	1.5997	2.1704	2.6744
	MCP	1.1425	1.1388	1.2918	1.3819	1.9672	6.8625	16.3342	2.2878	1.9511	1.7781	2.1611	2.7441
Credit	Ridge	1.0261	1.0028	1.0572	1.4168	1.0598	5.7208	7.2080	2.4315	3.3467	3.7940	2.9784	2.4538
	LASSO	1.0021	0.9785	0.9927	1.5033	1.0136	6.4883	6.9692	3.0774	3.0391	2.9347	2.4813	2.1557
	SCAD	1.0704	1.0460	1.0449	1.9804	1.2039	5.1074	9.1714	1.7050	3.9599	3.2258	2.6900	2.1097
	MCP	1.0673	1.3917	1.0384	1.0408	1.1485	5.5161	10.9986	3.4505	3.0274	3.0755	2.6097	2.0653

4.3.2 Average value of regularization hyperparameter (λ)

For each sub-scenario under Scenario 1, Scenario 2, and Scenario 3, the average optimal regularization parameter, denoted as $\overline{\lambda}$, is calculated as follows:

$$\overline{\lambda} = \frac{1}{M} \sum_{m=1}^{M} \lambda_m,$$

where λ_m is the optimal lambda value from the mth repetition and $M = 1000$ is the total number of repetitions. Average optimal λ value results over 1000 repetitions across forty-eight sub-scenario for each Insurance and Credit data sets are given in Table 4.6.

Low λ values indicate a minimal effect of the penalty term, showing that the model resembles the ordinary least squares method more closely.

Under each splitting scenario given a data set and penalty function, it was observed that the estimated values of λ (regularization parameter) differ.

The fact that the λ values are distinct from zero indicates that regularization methods like Ridge, SCAD, and MCP have a significant impact on the models, suggesting that these models fit the data better. On the other hand, in the LASSO model, the λ values being close to zero imply that the model resembles an unpenalized ordinary least squares regression, suggesting almost no effect of regularization.

In Scenario 1, for both data sets, it was observed that as the size of the training set increased, the optimal λ values also increased. A similar situation has been observed for Scenario 3 in the Credit data set.

Table 4.6 : Average optimum λ value over 1000 repetitions.

Data Set	Regularization Method	Scenario 1					Scenario 2		Scenario 3				
		Train/Validation Split Ratio					Number of Folds		Train/Validation Split Ratio				
		50/50	60/40	70/30	80/20	90/10	5	10	50/50	60/40	70/30	80/20	90/10
Insurance	Ridge	1.6898	2.0335	2.4280	3.1839	4.6907	0.2775	0.2430	0.2951	0.5884	0.5023	0.1914	0.1054
	LASSO	0.0040	0.0039	0.0040	0.0047	0.0061	0.0010	0.0010	0.0011	0.0016	0.0017	0.0016	0.0013
	SCAD	3.7819	4.5001	5.6401	7.1253	9.2687	0.4384	0.3743	2.6398	13.2170	26.0488	23.8324	8.7603
	MCP	4.1005	4.9342	5.8638	7.5815	9.4670	0.3965	0.2849	0.4421	1.1038	18.9624	29.1557	20.0810
Credit	Ridge	1.1202	1.3284	1.6112	2.0369	3.0054	0.4787	0.4770	0.0200	0.0342	0.0320	0.0110	0.0089
	LASSO	0.0089	0.0084	0.0087	0.0091	0.0108	0.0040	0.0036	0.0012	0.0012	0.0040	0.0047	0.0010
	SCAD	3.0864	3.5908	4.7311	6.2209	8.2409	3.3925	3.9237	1.9401	5.1927	9.1716	14.8707	17.7825
	MCP	3.1080	3.7933	5.0138	6.6631	8.5675	3.44978	4.0376	1.8720	4.6214	8.4188	14.7421	18.3897

4.3.3 Standard deviation of regularization hyperparameter (λ)

The standard deviation of the optimal regularization parameter, denoted as σ_λ, quantifies the variation in the optimal λ values obtained from each repetition under a given sub-scenario. For each sub-scenario under Scenario 1, Scenario 2, and Scenario 3, it is calculated as follows:

$$\sigma_\lambda = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}(\lambda_m - \overline{\lambda})^2},$$

where λ_m is the optimal lambda value from the mth repetition, $\overline{\lambda}$ is the mean of the optimal lambda values across all 1000 repetitions, obtained in Section 4.3.2, and $M = 1000$ is the total number of repetitions. The results for standard deviation of optimal λ value over 1000 repetitions across forty-eight sub-scenario for each Insurance and Credit data sets are given in Table 4.7 and illustrated in Figures A.7 - A.12.

In terms of standard deviation, we observed that cross-validation provides the least variable lambda values across 1000 runs, followed by the proposed optimization approach in Scenario 3. In Scenario 2, where k-fold cross-validation is applied, variability decreases as the value of k increases.

When comparing Scenario 1, the traditional one-time split method, with Scenario 3, the proposed optimization-based data splitting method, using the same training/validation ratios (e.g., comparing Insurance data set - Scenario 1 - Ridge regression - 50/50 with Insurance data set - Scenario 3 - Ridge regression - 50/50), we generally observed that Scenario 3 has lower standard deviations of λ values in Ridge regression and LASSO, and usually in SCAD and MCP as well.

Since the best λ value is chosen as the one that minimizes the MSE on the validation set, increasing the size of the validation set relative to the training set in Scenario 1 leads to a decrease in the standard deviation of λ values in both data sets, using traditional and randomized one-time split approaches. A similar trend can be suggested for Scenario 3, though it is not as clear as in Scenario 1. These observations suggest that a reduction in the standard deviation of λ values indicates a better fit of the model to the data and more consistent results.

Table 4.7 : Standard deviation of optimum λ value over 1000 repetitions.

Data Set	Regularization Method	Scenario 1					Scenario 2		Scenario 3				
		Train/Validation Split Ratio					Number of Folds		Train/Validation Split Ratio				
		50/50	60/40	70/30	80/20	90/10	5	10	50/50	60/40	70/30	80/20	90/10
Insurance	Ridge	2.6195	3.0833	3.8468	5.0095	7.5027	0.2895	0.2158	0.1750	0.2910	0.3746	0.3125	0.2565
	LASSO	0.0048	0.0048	0.0050	0.0062	0.0092	0.0001	0.0000	0.0002	0.0005	0.0007	0.0008	0.0006
	SCAD	5.1323	6.2421	7.5435	9.5169	11.9087	0.4736	0.3482	4.3177	5.8984	7.9682	11.6098	10.7097
	MCP	5.7483	6.7768	8.0243	9.8427	12.1205	0.5793	0.4054	0.4702	3.1183	12.7655	5.7626	10.5801
Credit	Ridge	1.5877	1.7726	2.0938	2.7197	4.3544	0.3671	0.2722	0.0357	0.0612	0.0831	0.0432	0.0499
	LASSO	0.0094	0.0086	0.0089	0.0093	0.0122	0.0024	0.0020	0.0004	0.0005	0.0040	0.0073	0.0002
	SCAD	3.3616	3.9915	5.3560	6.9076	8.3934	2.1999	2.2881	4.3542	5.0322	5.2643	6.0127	8.2857
	MCP	3.3408	4.3545	5.7880	7.3379	8.4745	2.4339	2.6028	4.5865	5.3294	4.2356	6.4392	8.7198

4.3.4 Prediction error for validation, training, and test sets

After finding the optimum values for each run, models were re-trained using the optimum λ and prediction error were calculated for the big training set, and performance was tested on the test data. The prediction error for the validation, big training, and test sets is quantified using the Mean Squared Error (MSE). It is computed for each set as follows:

- Validation MSE: Measures the average of the squares of the prediction errors, which are the differences between actual and predicted values in the validation data set.
- Big Training MSE: Similarly, after determining the optimal hyperparameters, this metric represents the average of the squares of the prediction errors in the big training data set.
- Test MSE: This metric evaluates the prediction error in the test data set, providing an estimation of performance of the model on new, unseen data.

$$\overline{\text{MSE}} = \frac{1}{M} \sum_{m=1}^{M} \text{MSE}_m,$$

where MSE_m is the MSE computed in the mth repetition under a give sub-scenario and $\overline{\text{MSE}}$ is the average MSE over $M = 1000$ repetitions. The average results for the MSE of the validation set, big training set, and test set over 1000 repetitions across forty-eight sub-scenario for each Insurance and Credit data sets are given in Tables 4.8 - 4.10 and illustrated Figures A.13 - A.30. In both data sets, Scenario 1 consistently yielded the lowest validation MSE, typically with a $90/10$ training/validation ratio.

In Scenario 1, as the size of the training set increased and the size of the validation set decreased, a reduction in validation MSE was observed across both data sets and all regularization methods.

In Scenario 2, the k-fold cross-validation, higher validation MSE values were consistently observed for $k = 5$ compared to $k = 10$.

In Scenario 3, data points were assigned to the training set if they had a high contribution to model performance and to the validation set if their contribution was low. As expected, higher validation MSE values were seen compared to randomly split scenarios. Contrary to Scenario 1, an increase in training set size and a decrease in validation set size led to increased validation MSE values.

Generally, it was observed that test MSE values are close to the MSE values for the large training set, indicating that the models are well-tuned and there is no over-fitting.

On the other hand, in terms of test MSE, cross-validation (Scenario 2) effectively increases the size of the training set by using the same data set multiple times, providing the lowest MSE values. However, Scenario 3 competes closely with Scenario 2 and even outperforms it in instances like the credit data set under Ridge and LASSO regularization methods. The success demonstrated in the test MSE indicates that the proposed optimization-based data splitting method is effective. As expected, the traditional one-time split method (Scenario 1) leads to slightly higher test MSE values.

Observations made in MSE tests can also be seen in the standard deviation of coefficients. The greater the variability in coefficients, the higher the MSE on test data.

Table 4.8 : Average MSE validation value over 1000 repetitions.

Data Set	Regularization Method	Scenario 1					Scenario 2		Scenario 3				
		Train/Validation Split Ratio					Number of Folds		Train/Validation Split Ratio				
		50/50	60/40	70/30	80/20	90/10	5	10	50/50	60/40	70/30	80/20	90/10
Insurance	Ridge	0.2006	0.1996	0.1997	0.1993	0.1978	0.1998	0.1997	0.4086	0.5140	0.6954	1.0699	1.7665
	LASSO	0.2007	0.1997	0.1998	0.1993	0.1978	0.1999	0.1997	0.4140	0.5142	0.6946	1.0726	1.7719
	SCAD	0.2008	0.1998	0.1999	0.1995	0.1983	0.1998	0.1997	0.4096	0.5134	0.6858	1.0554	1.7512
	MCP	0.2008	0.1998	0.1999	0.1995	0.1983	0.1998	0.1997	0.4097	0.5158	0.6902	1.0486	1.7428
Credit	Ridge	0.2158	0.2092	0.2073	0.2058	0.2059	0.2119	0.2106	0.5019	0.7613	1.0970	1.4682	2.0131
	LASSO	0.2144	0.2081	0.2061	0.2046	0.2047	0.2112	0.2100	0.5772	0.7943	1.1192	1.4690	2.0367
	SCAD	0.2148	0.2082	0.2054	0.2022	0.1961	0.2105	0.2092	0.4338	0.5766	0.7769	1.1199	1.8149
	MCP	0.2147	0.2081	0.2055	0.2019	0.1949	0.2105	0.2091	0.43253	0.5917	0.7943	1.1217	1.8216

Table 4.9 : Average MSE big train value over 1000 repetitions.

Data Set	Regularization Method	Scenario 1					Scenario 2		Scenario 3				
		Train/Validation Split Ratio					Number of Folds		Train/Validation Split Ratio				
		50/50	60/40	70/30	80/20	90/10	5	10	50/50	60/40	70/30	80/20	90/10
Insurance	Ridge	0.1969	0.1970	0.1971	0.1972	0.1977	0.1968	0.1968	0.1968	0.1968	0.1968	0.1968	0.1968
	LASSO	0.1974	0.1974	0.1974	0.1976	0.1983	0.1968	0.1968	0.1968	0.1969	0.1969	0.1969	0.1969
	SCAD	0.1969	0.1969	0.1969	0.1970	0.1971	0.1968	0.1968	0.1969	0.1971	0.1979	0.1978	0.1971
	MCP	0.1969	0.1969	0.1970	0.1970	0.1971	0.1968	0.1968	0.1968	0.1968	0.1975	0.1980	0.1975
Credit	Ridge	0.1906	0.1909	0.1915	0.1925	0.1951	0.1896	0.1896	0.1895	0.1895	0.1895	0.1895	0.1895
	LASSO	0.1945	0.1939	0.1943	0.1947	0.1971	0.1903	0.1901	0.1896	0.1896	0.1906	0.1919	0.1895
	SCAD	0.1911	0.1932	0.1975	0.2044	0.2156	0.1905	0.1901	0.1914	0.1922	0.1961	0.2140	0.2423
	MCP	0.1922	0.1932	0.1983	0.2020	0.2124	0.1902	0.1904	0.1916	0.1925	0.1966	0.2182	0.2438

Table 4.10 : Average MSE test value over 1000 repetitions.

Data Set	Regularization Method	Scenario 1					Scenario 2		Scenario 3				
		Train/Validation Split Ratio					Number of Folds		Train/Validation Split Ratio				
		50/50	60/40	70/30	80/20	90/10	5	10	50/50	60/40	70/30	80/20	90/10
Insurance	Ridge	0.1985	0.1985	0.1987	0.1988	0.1991	0.1984	0.1984	0.1984	0.1984	0.1984	0.1984	0.1984
	LASSO	0.1990	0.1989	0.1990	0.1992	0.1998	0.1984	0.1984	0.1985	0.1985	0.1985	0.1985	0.1985
	SCAD	0.1985	0.1985	0.1985	0.1986	0.1988	0.1984	0.1984	0.1986	0.1989	0.1996	0.1996	0.1988
	MCP	0.1985	0.1985	0.1985	0.1986	0.1987	0.1984	0.1984	0.1984	0.1984	0.1992	0.1997	0.1993
Credit	Ridge	0.2120	0.2122	0.2126	0.2132	0.2155	0.2114	0.2114	0.2113	0.2113	0.2113	0.2114	0.2114
	LASSO	0.2144	0.2140	0.2142	0.2144	0.2161	0.2117	0.2116	0.2111	0.2111	0.2116	0.2138	0.2111
	SCAD	0.2124	0.2150	0.2198	0.2273	0.2369	0.2120	0.2113	0.2135	0.2139	0.2177	0.2373	0.2699
	MCP	0.2142	0.2153	0.2202	0.2259	0.2360	0.2112	0.2115	0.2137	0.2142	0.2184	0.2458	0.2724

4.3.5 Average coefficients and average standard deviation of the coefficients

Let $\hat{\beta}_{jm}$ represents the value of the jth coefficient estimated for a data set at the mth repetition of a given sub-scenario. Then, the average value of the jth coefficient over $M = 1000$ repetitions is computed as:

$$\overline{\hat{\beta}}_j = \frac{1}{M} \sum_{m=1}^{M} \hat{\beta}_{jm},$$

where $j = 1, ..., p$ with p is the total number of parameters in the given data set, and $\overline{\hat{\beta}}_j$ is the mean value of the jth coefficient over $M = 1000$ repetitions in that specific data set for the given sub-scenario.

The standard deviation of the jth coefficient $\hat{\beta}_j$ is calculated to measure the variability of the coefficient values over $M = 1000$ repetitions:

$$\hat{\sigma}_{\hat{\beta}_j} = \sqrt{\frac{1}{M-1} \sum_{m=1}^{M} (\hat{\beta}_{jm} - \overline{\hat{\beta}}_j)^2},$$

where $\hat{\beta}_{jm}$ is the value of the estimated jth coefficient for the mth repetition, $\overline{\hat{\beta}}_j$ is the mean of the estimated jth coefficient computed previously, and $\hat{\sigma}_{\hat{\beta}_j}$ represents the standard deviation of the estimated jth coefficient over $M = 1000$ repetitions in that specific data set for the given sub-scenario.

Lastly, each $\overline{\hat{\beta}}_j$ is averaged over p parameters to get an overall average coefficient $\overline{\hat{\beta}}$:

$$\overline{\hat{\beta}} = \frac{1}{p} \sum_{j=1}^{p} \overline{\hat{\beta}}_j.$$

In a similar fashion, each $\hat{\sigma}_{\hat{\beta}_j}$ is averaged over p parameters to get an overall average coefficient $\overline{\hat{\sigma}}_{\hat{\beta}_j}$:

$$\overline{\hat{\sigma}}_{\hat{\beta}} = \frac{1}{p} \sum_{j=1}^{p} \hat{\sigma}_{\hat{\beta}_j}.$$

The grand average of the coefficients and the grand average of standard deviations over 1000 repetitions across forty-eight sub-scenario for each Insurance and Credit data sets are given in Tables 4.11, and 4.12.

The grand average standard deviations of the coefficients, as presented in Table 4.12, reflect how consistently each model and scenario fits the data. Lower grand average

standard deviations indicate that model coefficients produce similar values across different runs, providing more reliable predictions. Higher grand average standard deviations suggest that the model is more variable, indicating less stability across different data subsets or instances.

As discussed in the MSE test analysis, the variability in coefficients and MSE test exhibit similar behaviors. Cross-validation (Scenario 2), which involves training on the data multiple times, generally results in lower standard deviations, indicating better generalizability of the model. However, Scenario 3, the proposed optimization approach, competes with Scenario 2 and even surpasses it in MSE test performance for the Credit data set using Ridge and LASSO regularization methods, similarly to cross-validation.

Table 4.11 : Grand average of coefficients over 1000 repetitions.

Data Set	Regularization Method	Scenario 1					Scenario 2		Scenario 3				
		Train/Validation Split Ratio					Number of Folds		Train/Validation Split Ratio				
		50/50	60/40	70/30	80/20	90/10	5	10	50/50	60/40	70/30	80/20	90/10
Insurance	Ridge	-0.1217	-0.1213	-0.1208	-0.1200	-0.1184	-0.1234	-0.1234	-0.1234	-0.1230	-0.1231	-0.1235	-0.1236
	LASSO	-0.1249	-0.1249	-0.1249	-0.1250	-0.1248	-0.1241	-0.1241	-0.1241	-0.1243	-0.1243	-0.1243	-0.1242
	SCAD	-0.1257	-0.1259	-0.1263	-0.1268	-0.1275	-0.1241	-0.1240	-0.1248	-0.1285	-0.1329	-0.1324	-0.1281
	MCP	-0.1262	-0.1265	-0.1269	-0.1276	-0.1282	-0.1241	-0.1240	-0.1241	-0.1245	-0.1321	-0.1360	-0.1329
Credit	Ridge	-0.0334	-0.0326	-0.0316	-0.0303	-0.0276	-0.0359	-0.0359	-0.0380	-0.0379	-0.0379	-0.0380	-0.0381
	LASSO	-0.0255	-0.0261	-0.0256	-0.0249	-0.0223	-0.0326	-0.0331	-0.0364	-0.0364	-0.0328	-0.0311	-0.0367
	SCAD	-0.0443	-0.0443	-0.0405	-0.0377	-0.0296	-0.0464	-0.0472	-0.0415	-0.0459	-0.0475	-0.0420	-0.0290
	MCP	-0.0462	-0.0453	-0.0428	-0.0391	-0.0321	-0.0484	-0.0495	-0.0417	-0.0471	-0.0539	-0.0472	-0.0324

Table 4.12 : Grand average of standard deviation of coefficients over 1000 repetitions.

Data Set	Regularization Method	Scenario 1					Scenario 2		Scenario 3				
		Train/Validation Split Ratio					Number of Folds		Train/Validation Split Ratio				
		50/50	60/40	70/30	80/20	90/10	5	10	50/50	60/40	70/30	80/20	90/10
Insurance	Ridge	0.0109	0.0113	0.0124	0.0137	0.0167	0.0090	0.0090	0.0090	0.0090	0.0090	0.0091	0.0090
	LASSO	0.0151	0.0149	0.0152	0.0173	0.0221	0.0090	0.0090	0.0091	0.0092	0.0091	0.0094	0.0093
	SCAD	0.0095	0.0097	0.0099	0.0104	0.0111	0.0090	0.0090	0.0096	0.0100	0.0104	0.0112	0.0110
	MCP	0.0097	0.0098	0.0101	0.0106	0.0113	0.0090	0.0090	0.0091	0.0093	0.0116	0.0097	0.0112
Credit	Ridge	0.0354	0.0366	0.0387	0.0426	0.0509	0.0298	0.0294	0.0292	0.0292	0.0293	0.0292	0.0292
	LASSO	0.0455	0.0435	0.0447	0.0461	0.0528	0.0304	0.0300	0.0286	0.0287	0.0329	0.0434	0.0287
	SCAD	0.0366	0.0457	0.0591	0.0748	0.0968	0.0308	0.0288	0.0414	0.0405	0.0514	0.0851	0.1218
	MCP	0.0425	0.0469	0.0645	0.0690	0.0864	0.0298	0.0304	0.0411	0.0424	0.0573	0.0912	0.1277

5. CONCLUSION

This thesis presents an optimization-based approach aimed at enhancing the effectiveness of data splitting methods used in machine learning experiments. This approach assigns data points to training and validation sets based on model performance. When compared with traditional one-time split and k-fold cross-validation methods, the proposed method competes effectively in improving the accuracy and stability of models.

The study also aimed to compare the effects of regularization in machine learning algorithms. Among regularization models, Ridge regression and LASSO stand out and are readily implemented using the Scikit-learn library of Python. Additionally, less commonly discussed in the literature, the SCAD and MCP penalty functions were included in this study. Since these functions are not readily available in Python libraries, functions were developed to fit SCAD and MCP models, thus incorporating them into the study.

Furthermore, the impact of different data splitting ratios on model learning and validation capabilities was assessed. Dynamically adjustable training/validation splits have enhanced accuracy by better adapting to the nuances of each data set. These experiments were conducted on two distinct data sets, Insurance and Credit.

The results of these experiments demonstrated that, as expected, the k-fold cross-validation method facilitated the development of more accurate models compared to the one-time split method, and the proposed optimization-based data splitting method performed as well as k-fold cross-validation.

In the analysis of computational runtime, Scenario 1, which employs the one-time split method, requires less time compared to the other scenarios, underscoring its efficiency though potentially at the cost of less detailed analysis. Scenario 2, utilizing k-fold cross-validation, shows a direct correlation between increases in the value of k and

longer runtime, marking it as the most time-intensive scenario due to its repeated data splitting and model fitting. Scenario 3, which introduces an optimization-based approach for data splitting, is expected to yield the most accurate results, especially with large or complex data sets, but at a high time cost.

The analysis of the average value of the regularization hyperparameter λ across different scenarios indicates that the λ values vary, highlighting the significant impact of regularization methods such as Ridge, SCAD, MCP, and LASSO on the models. For most methods except LASSO, λ values significantly different from zero suggest a better model fit to the data. In contrast, λ values of LASSO, which are close to zero, imply that it approximates an unpenalized ordinary least squares regression, indicating minimal regularization effect.

Regarding the standard deviation of the regularization hyperparameter λ, both cross-validation and the proposed optimization approach show the least variability in λ values, pointing to their robustness. A comparative analysis between Scenario 1 and Scenario 3 under the same training/validation ratios demonstrates that Scenario 3 generally achieves lower standard deviations of λ values, emphasizing its consistency and reliability in modeling outcomes.

The prediction error analysis for validation, training, and test sets, measured by Mean Squared Error (MSE), reveals that Scenario 1 consistently yields the lowest validation MSE, particularly noticeable with a 90/10 training/validation ratio which optimizes performance. Conversely, in Scenario 2, configurations with $k = 5$ consistently result in higher MSE values compared to $k = 10$, showcasing the influence of fold count on model performance. Scenario 3, which strategically assigns data points to training or validation sets based on their contribution to model performance, often results in higher validation MSE values compared to scenarios with random data splits. Interestingly, as the size of the training set increases and the validation set decreases, an increase in validation MSE is observed, which highlights the intricate impact of data distribution on model training effectiveness.

In conclusion, this thesis focuses on the optimization of data splitting methods, offering an effective alternative to existing approaches in machine learning applications. The

proposed methodology stands out for its potential to improve model performance and provides a solid foundation for future research. Future studies could explore the applicability and effectiveness of this approach on different data types and more extensive data sets, thereby deepening the understanding of the impact of data splitting strategies on model performance.

REFERENCES

[1] **Hoerl, A.E. and Kennard, R.W.** (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems, *Technometrics*, *12*(1), 55–67.

[2] **Tibshirani, R.** (1996). Regression Shrinkage and Selection via the Lasso, *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *58*(1), 267–288.

[3] **Fan, J. and Li, R.** (2001). Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties, *Journal of the American Statistical Association*, *96*(456), 1348–1360.

[4] **Zhang, C.H.** (2010). Nearly Unbiased Variable Selection Under Minimax Concave Penalty, *The Annals of Statistics*, *38*(2), 894–942.

[5] **Pedregosa, F.**, **Varoquaux, G.**, **Gramfort, A.**, **Michel, V.**, **Thirion, B.**, **Grisel, O.**, **Blondel, M.**, **Prettenhofer, P.**, **Weiss, R.**, **Dubourg, V.**, **Vanderplas, J.**, **Passos, A.**, **Cournapeau, D.**, **Brucher, M.**, **Perrot, M. and Duchesnay, E.** (2011). Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, *12*, 2825–2830.

[6] **Van Rossum, G. and Drake, F.L.** (2009). *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA.

[7] **Bertsimas, D. and Paskov, I.** (2020). Stable Regression: On the Power of Optimization over Randomization in Training Regression Problems, *Journal of Machine Learning Research*, *21*(230), 1–25.

[8] **Jones, A.** *The Smoothly Clipped Absolute Deviation (SCAD) penalty*, `https://andrewcharlesjones.github.io/journal/scad.html`, accessed: 2024-05-09.

[9] **Fan, J.**, **Li, R.**, **Zhang, C.H. and Zou, H.** (2020). *Statistical Foundations of Data Science*, Chapman and Hall/CRC.

[10] **Wikipedia** (2023). *Machine Learning*, `https://en.wikipedia.org/wiki/Machine_learning`, accessed: 2023-04-12.

[11] **Gupta, S.** *Why Is Logistic Regression a Classification Algorithm?*, `https://builtin.com/machine-learning/logistic-regression-classification-algorithm`, accessed: 2024-05-09.

[12] **IBM**. *What is machine learning (ML)?*, https://www.ibm.com/topics/machine-learning, accessed: 2024-05-09.

[13] **Kroese, D.P.**, **Botev, Z.I. and Taimre, T.** (2019). *Data Science and Machine Learning: Mathematical and Statistical Methods*, Chapman Hall/CRC.

[14] **Bruce, P.**, **Bruce, A. and Gedeck, P.** (2020). *Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python*, O'Reilly Media.

[15] **Müller, A.C. and Guido, S.** (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*, O'Reilly Media.

[16] **Neter, J.**, **Kutner, M.H.**, **Nachtsheim, C.J.**, **Wasserman, W.** *et al.* (1996). *Applied Linear Statistical Models*, Irwin Chicago.

[17] **James, G.**, **Witten, D.**, **Hastie, T.**, **Tibshirani, R. and Taylor, J.** (2023). *An Introduction to Statistical Learning: With Applications in Python*, Springer Nature.

[18] **Hastie, T.**, **Tibshirani, R.**, **Friedman, J.H. and Friedman, J.H.** (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2, Springer.

[19] **Breheny, P.** (2016). Adaptive Lasso, MCP, and SCAD, *Slides. High Dimensional Data Analysis, University of Iowa, United States of America*.

[20] **Lantz, B.** (2019). *Machine Learning with R: Expert Techniques for Predictive Modeling*, Packt Publishing Limited.

[21] **Stedy**. *Machine Learning with R Datasets*, https://github.com/stedy/Machine-Learning-with-R-datasets, accessed: 2024-05-09.

[22] **James, G.**, **Witten, D.**, **Hastie, T. and Tibshirani, R.** *An Introduction to Statistical Learning*, https://www.statlearning.com/resources-second-edition, accessed: 2024-05-09.

APPENDICES

APPENDIX A : Boxplots for Computational Runtime, Regularization Hyperparameter (λ), and Prediction Error for Validation, Training, and Test Sets

APPENDIX A : Boxplots for Computational Runtime, Regularization Hyperparameter (λ), and Prediction Error for Validation, Training, and Test Sets

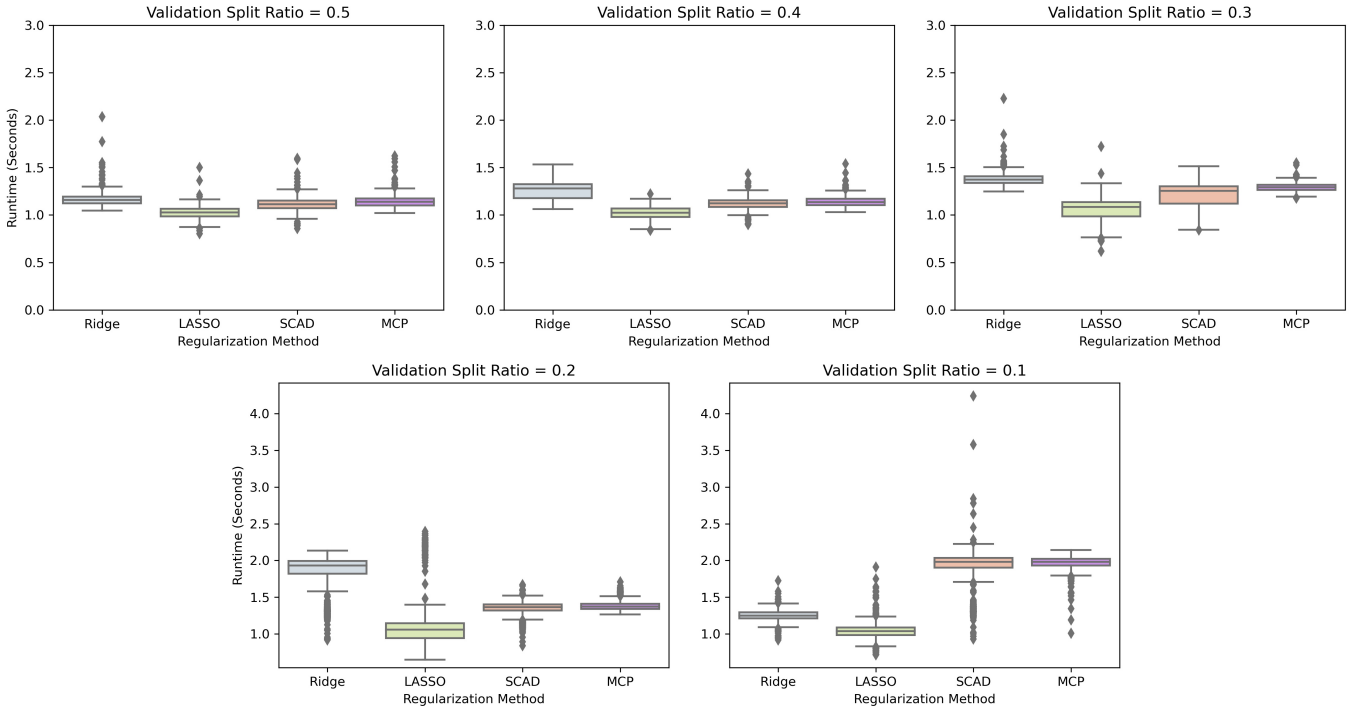

Figure A.1 : The distribution of runtime (in seconds) under Scenario 1 for the Insurance data set.

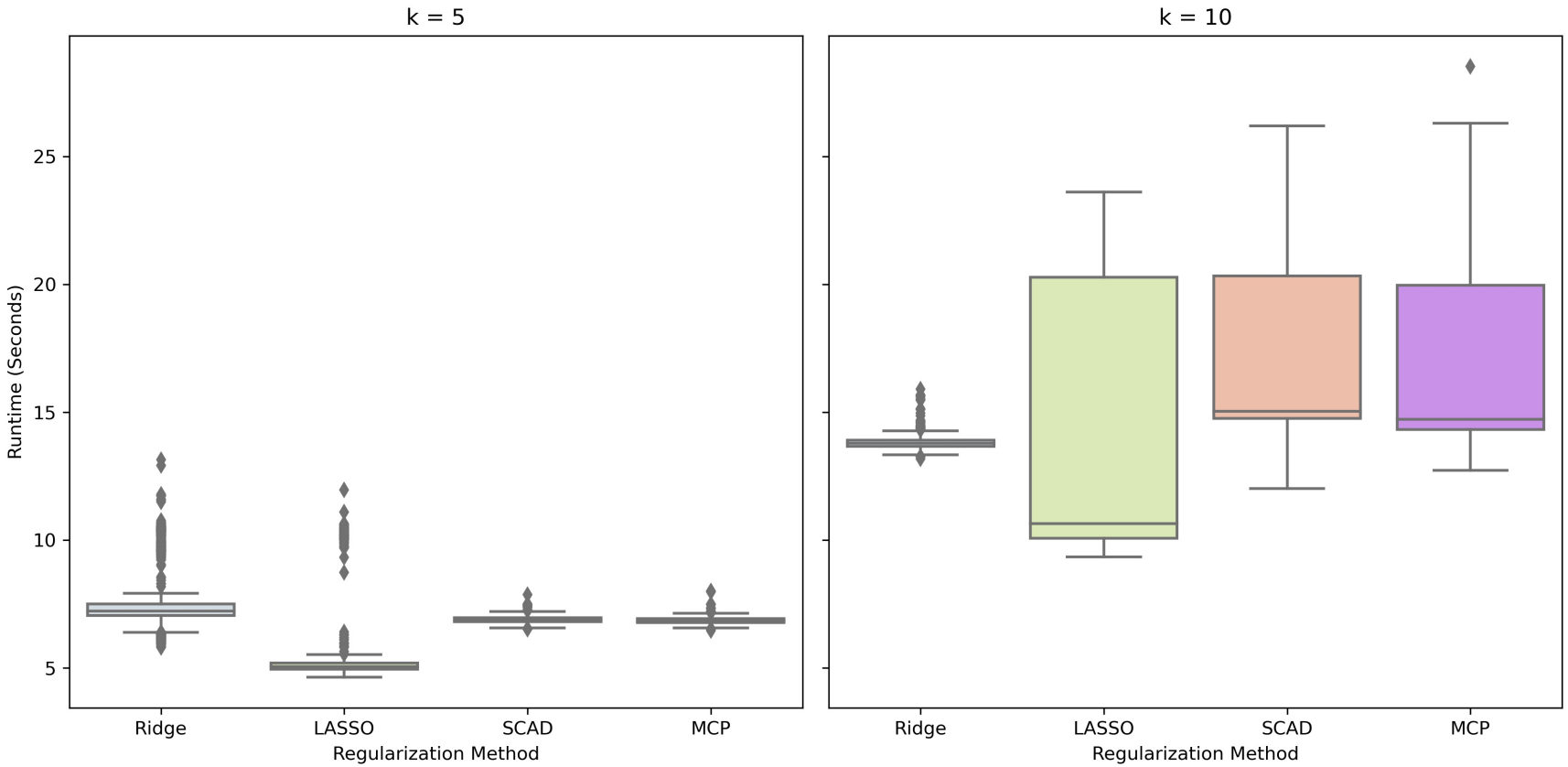

Figure A.2 : The distribution of runtime (in seconds) under Scenario 2 for the Insurance data set.

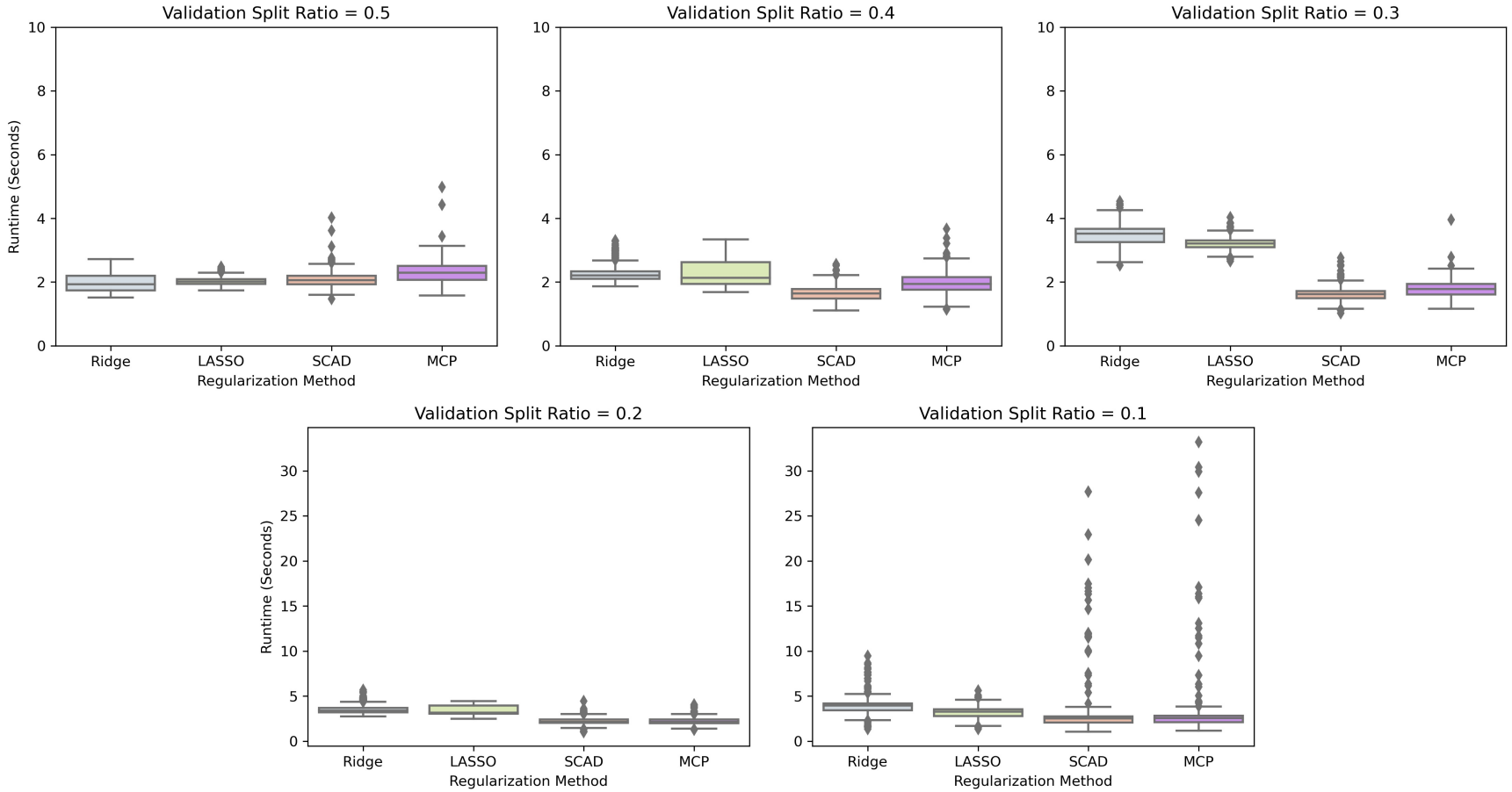

Figure A.3 : The distribution of runtime (in seconds) under Scenario 3 for the Insurance data set.

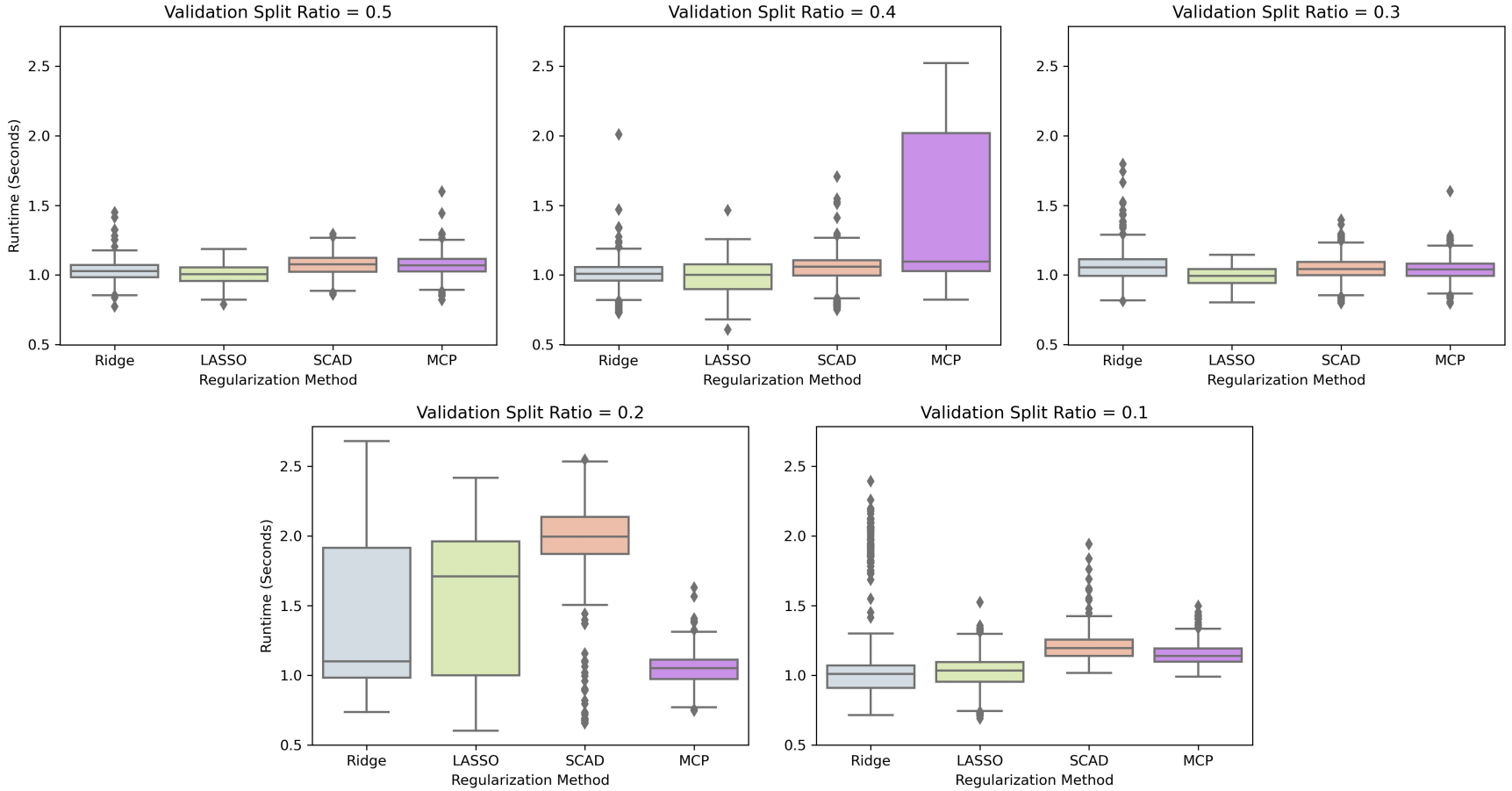

Figure A.4 : The distribution of runtime (in seconds) under Scenario 1 for the Credit data set.

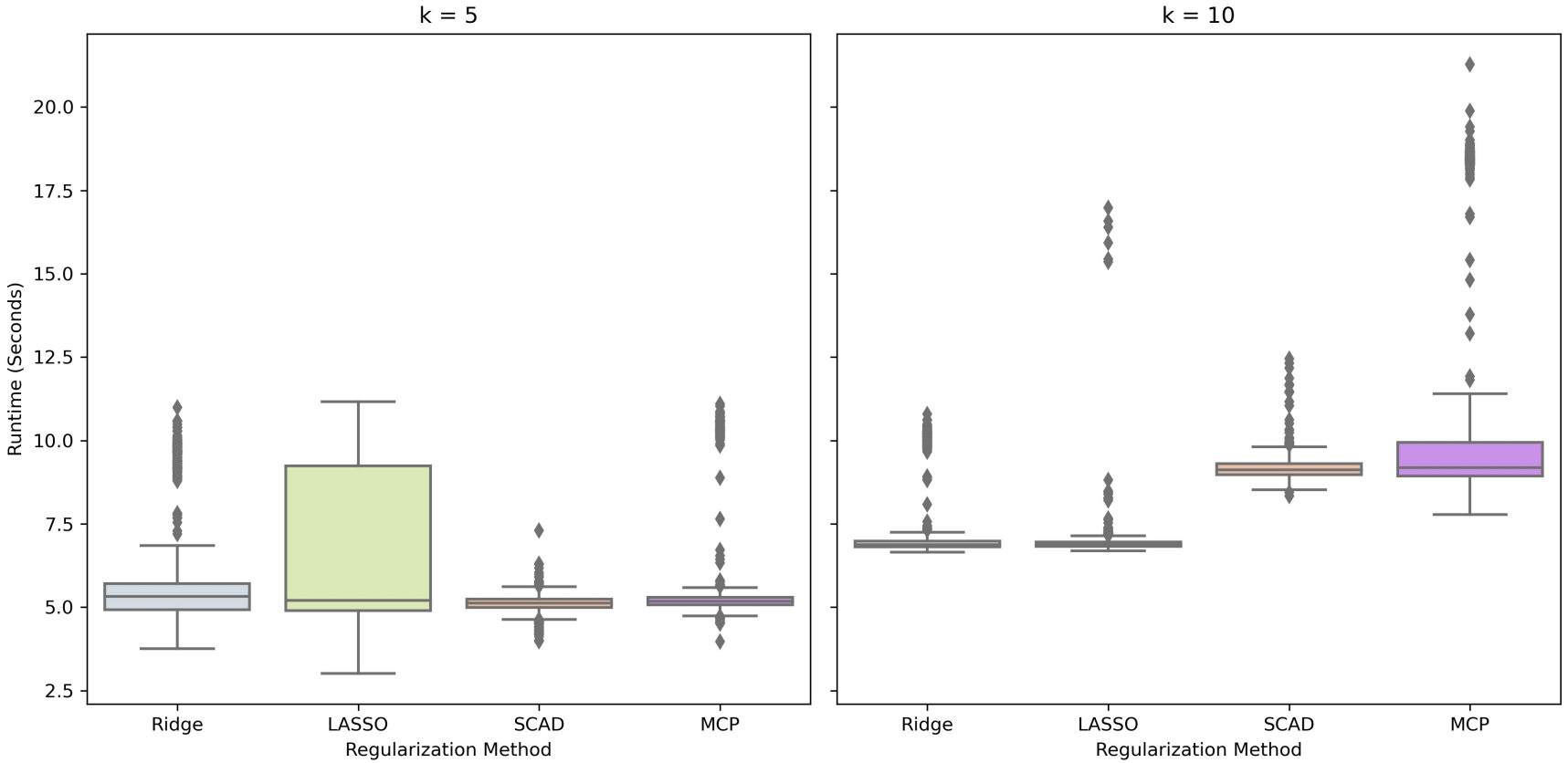

Figure A.5 : The distribution of runtime (in seconds) under Scenario 2 for the Credit data set.

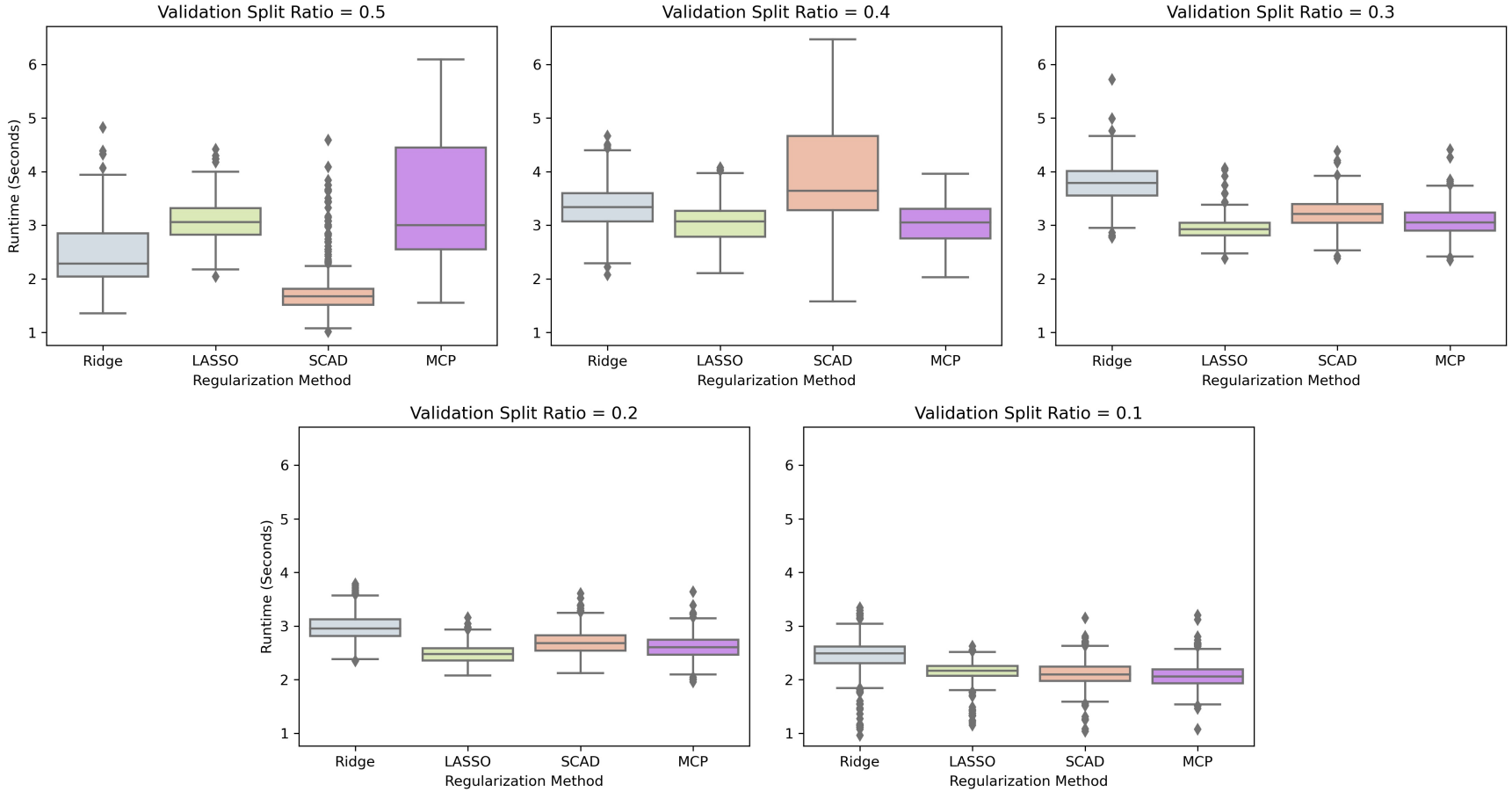

Figure A.6 : The distribution of runtime (in seconds) under Scenario 3 for the Credit data set.

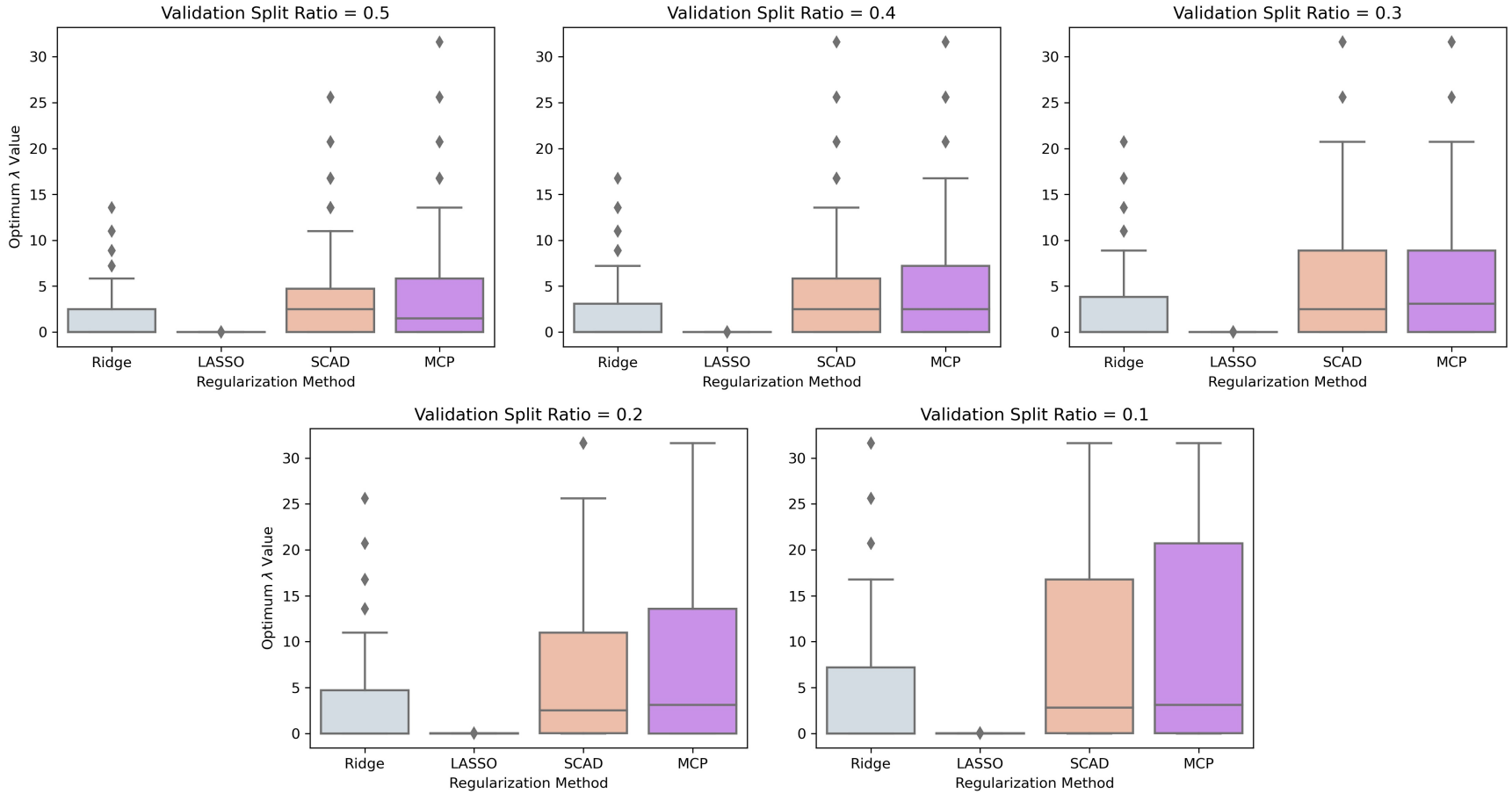

Figure A.7 : The distribution of optimum λ parameter under Scenario 1 for the Insurance data set.

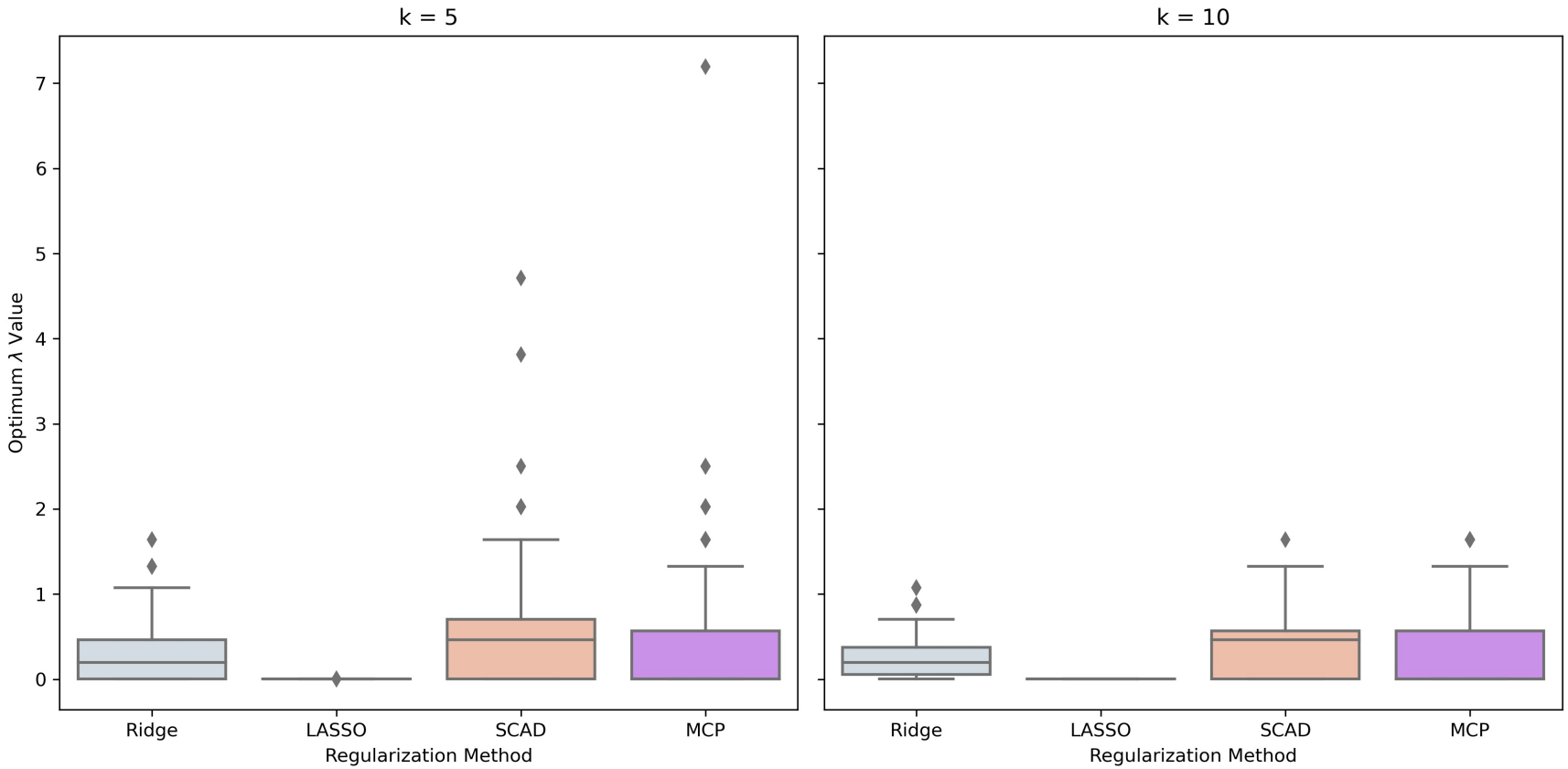

Figure A.8 : The distribution of optimum λ parameter under Scenario 2 for the Insurance data set.

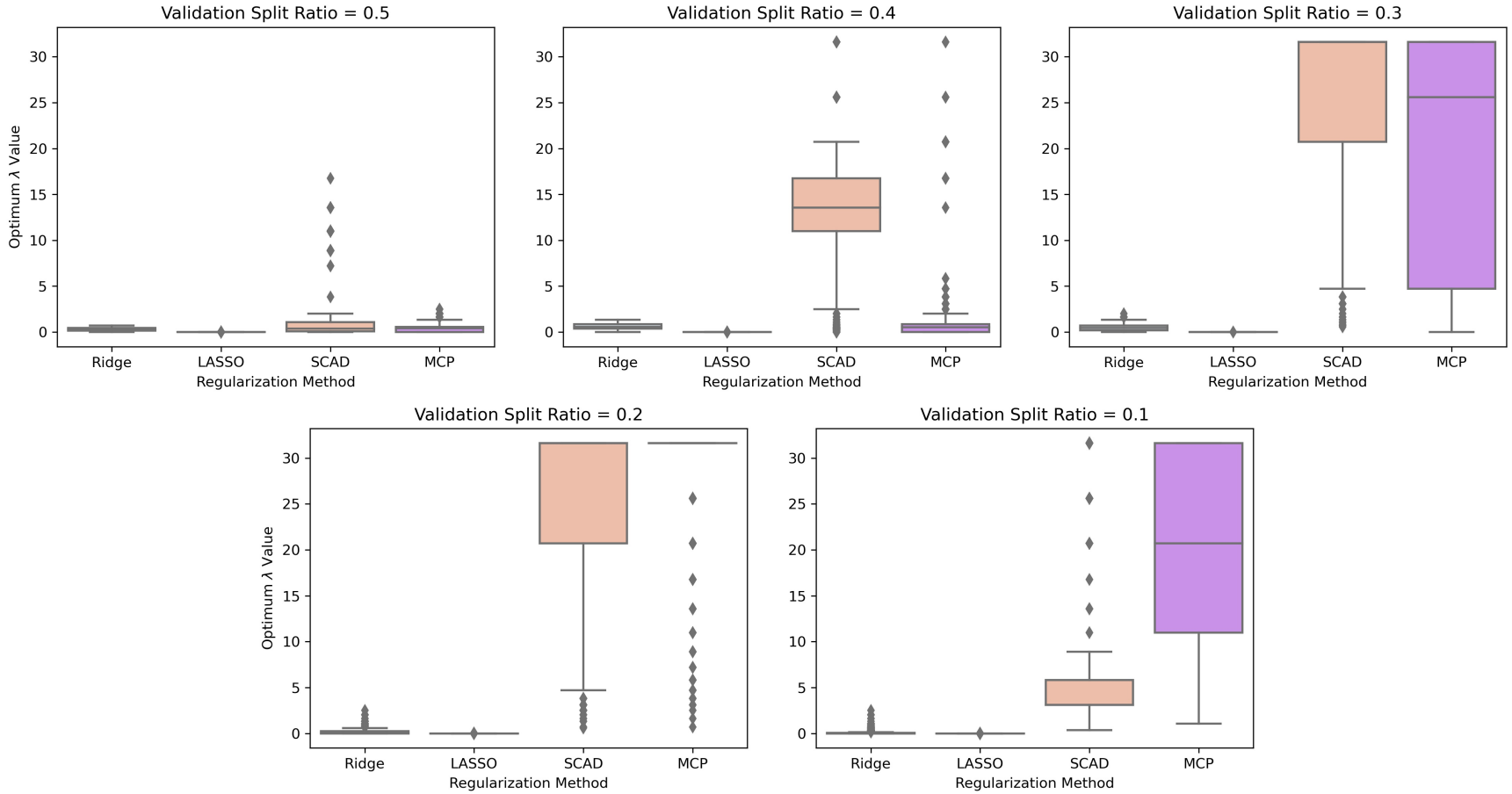

Figure A.9 : The distribution of optimum λ parameter under Scenario 3 for the Insurance data set.

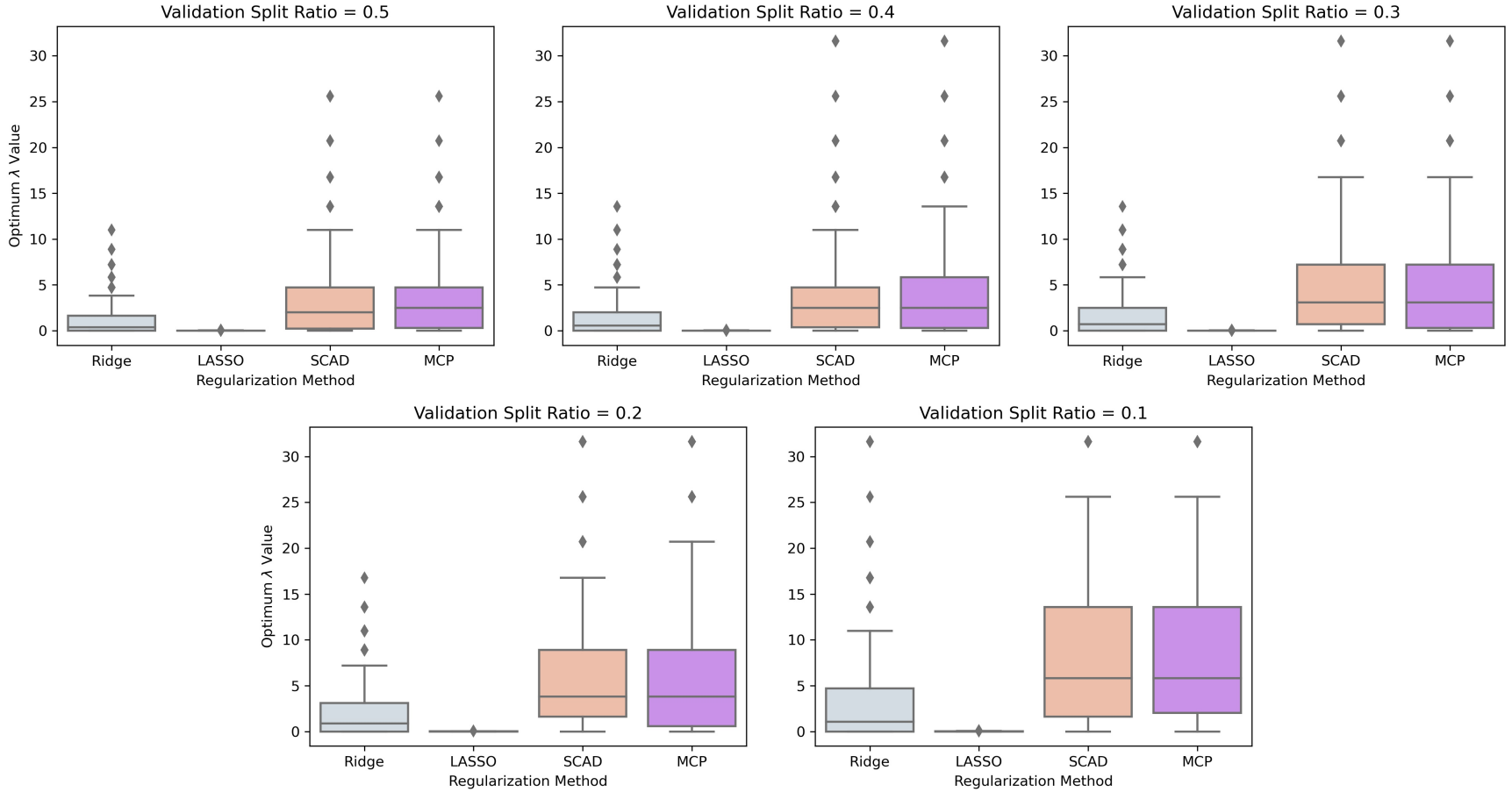

Figure A.10 : The distribution of optimum λ parameter under Scenario 1 for the Credit data set.

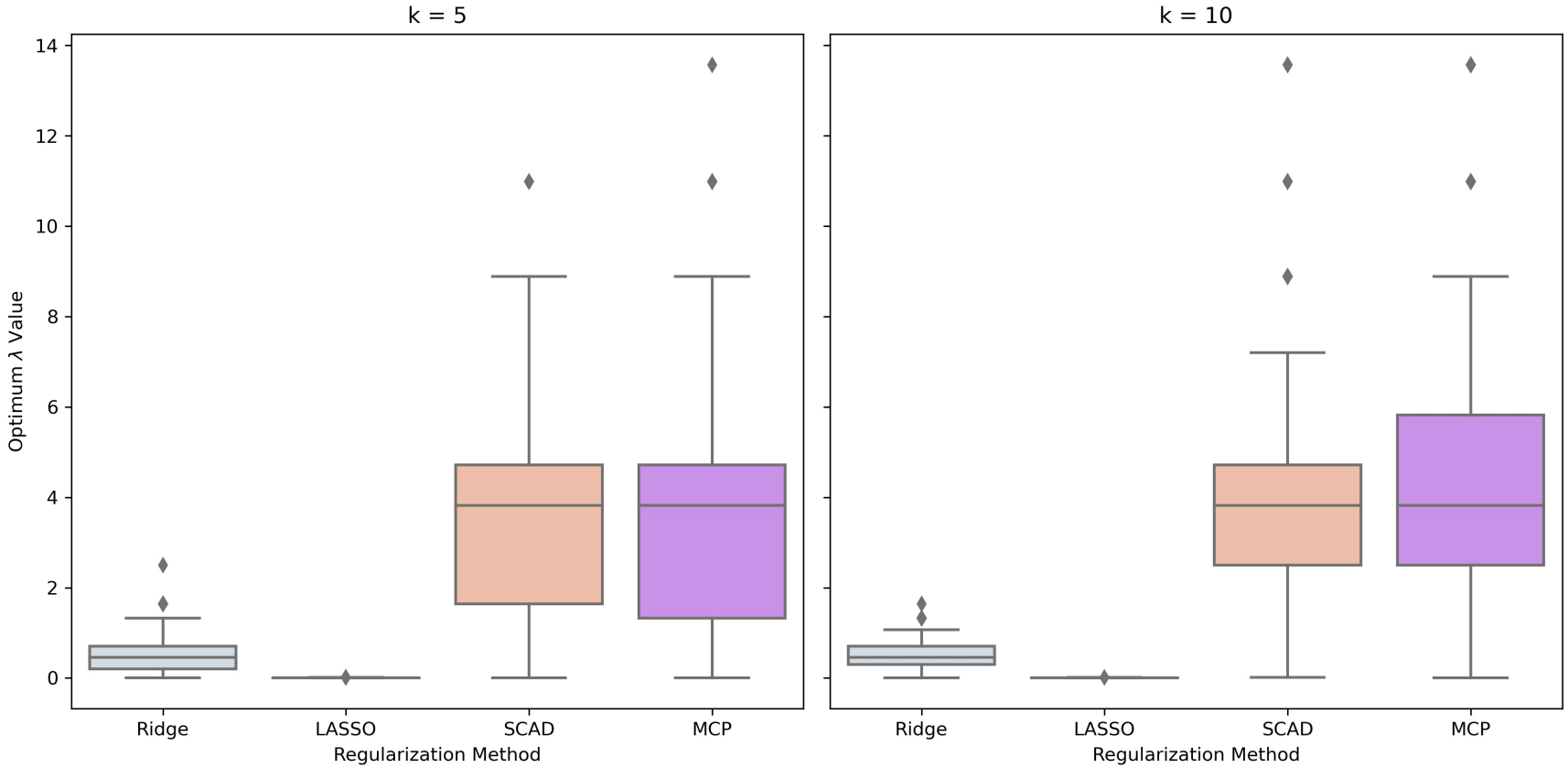

Figure A.11 : The distribution of optimum λ parameter under Scenario 2 for the Credit data set.

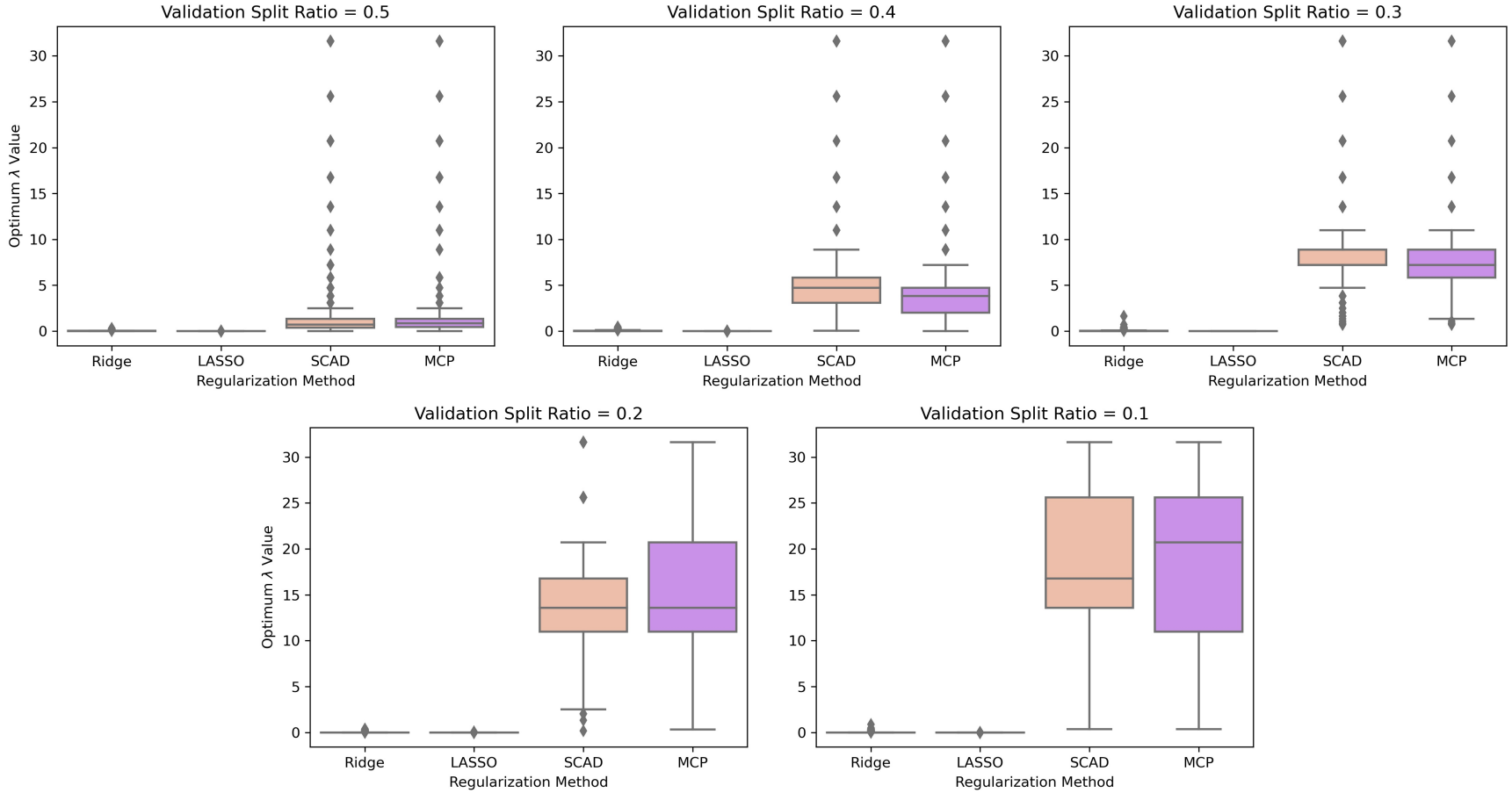

Figure A.12 : The distribution of optimum λ parameter under Scenario 3 for the Credit data set.

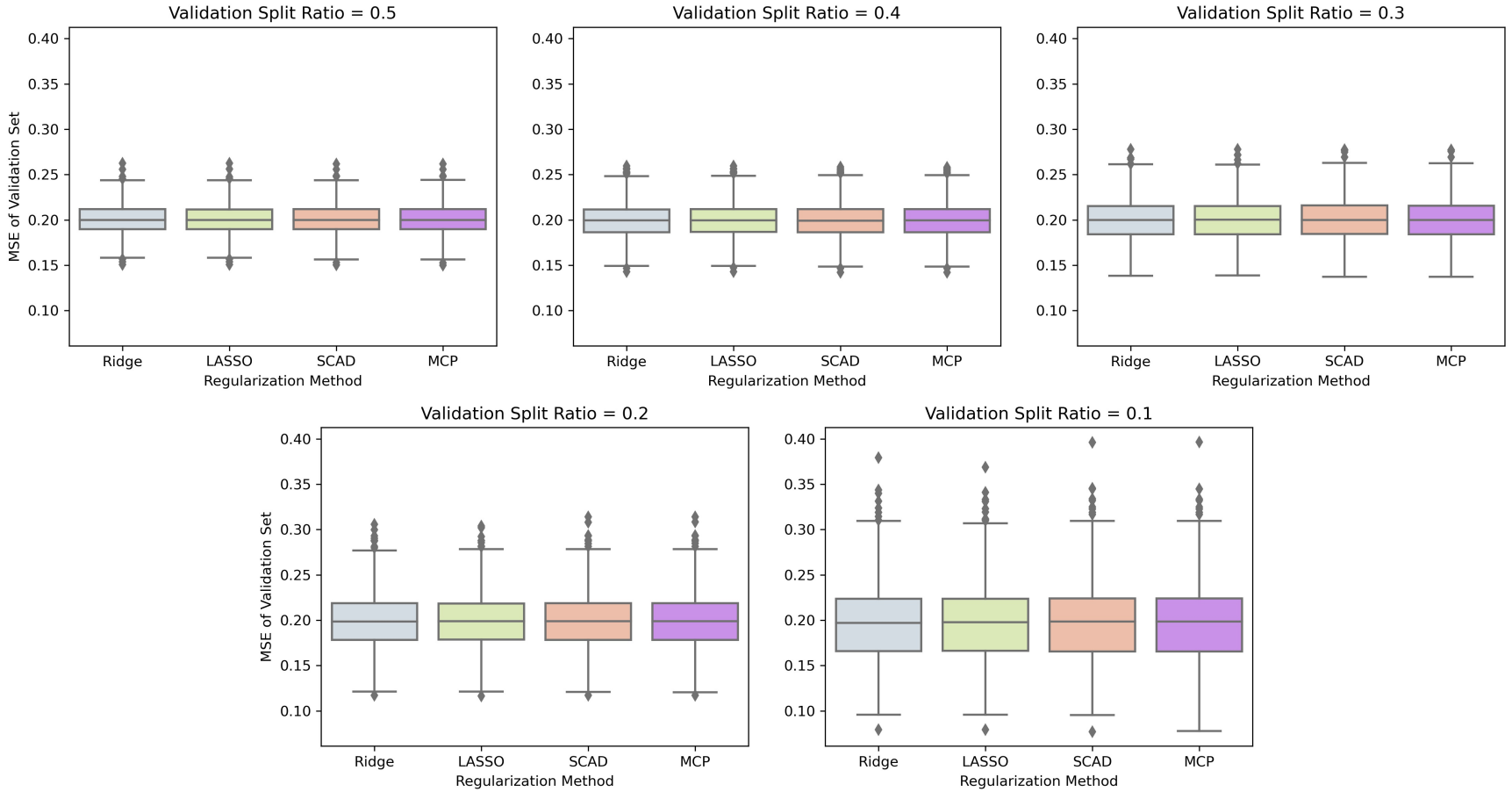

Figure A.13 : The distribution of MSE validation under Scenario 1 for the Insurance data set.

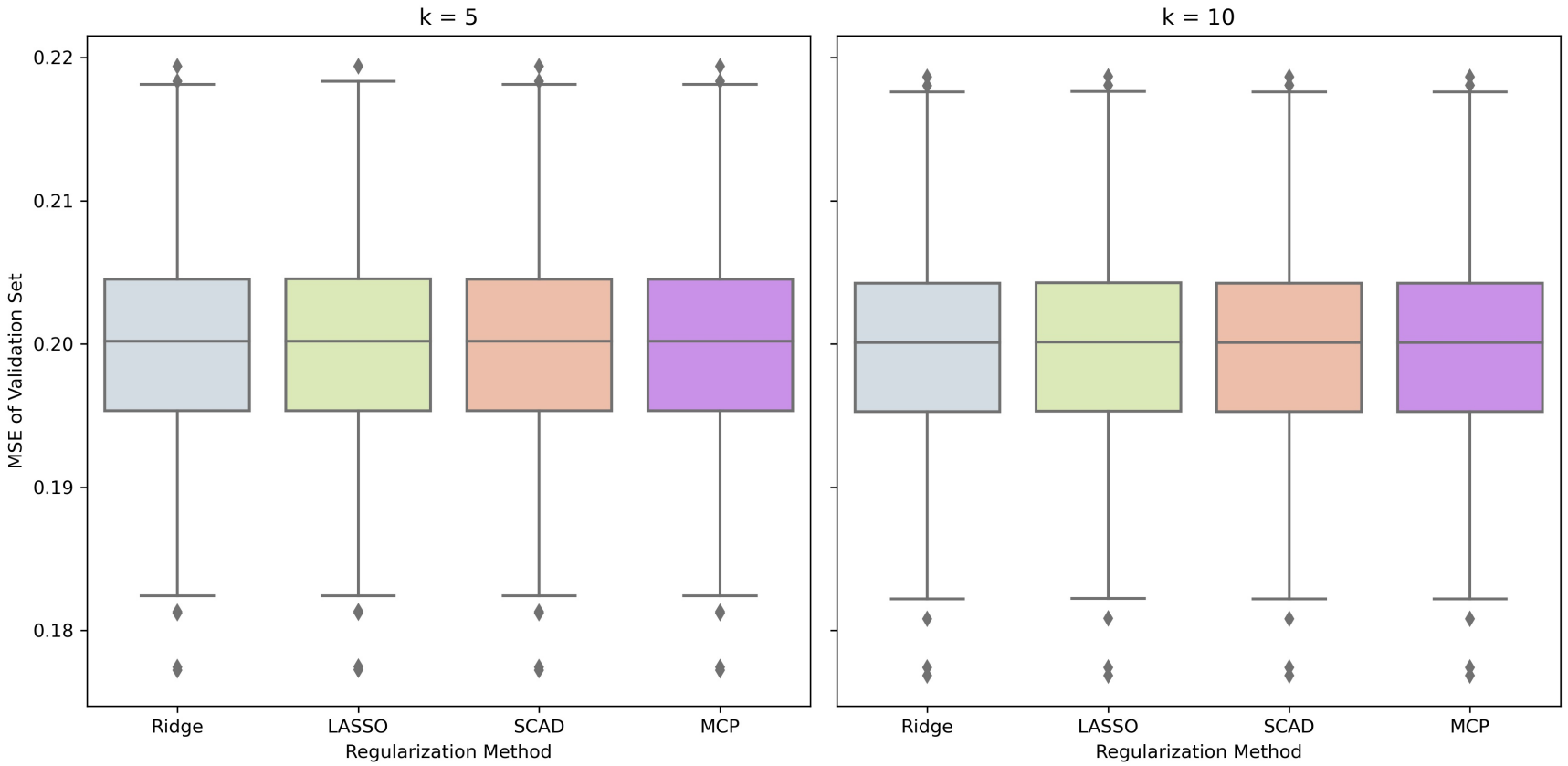

Figure A.14 : The distribution of MSE validation under Scenario 2 for the Insurance data set.

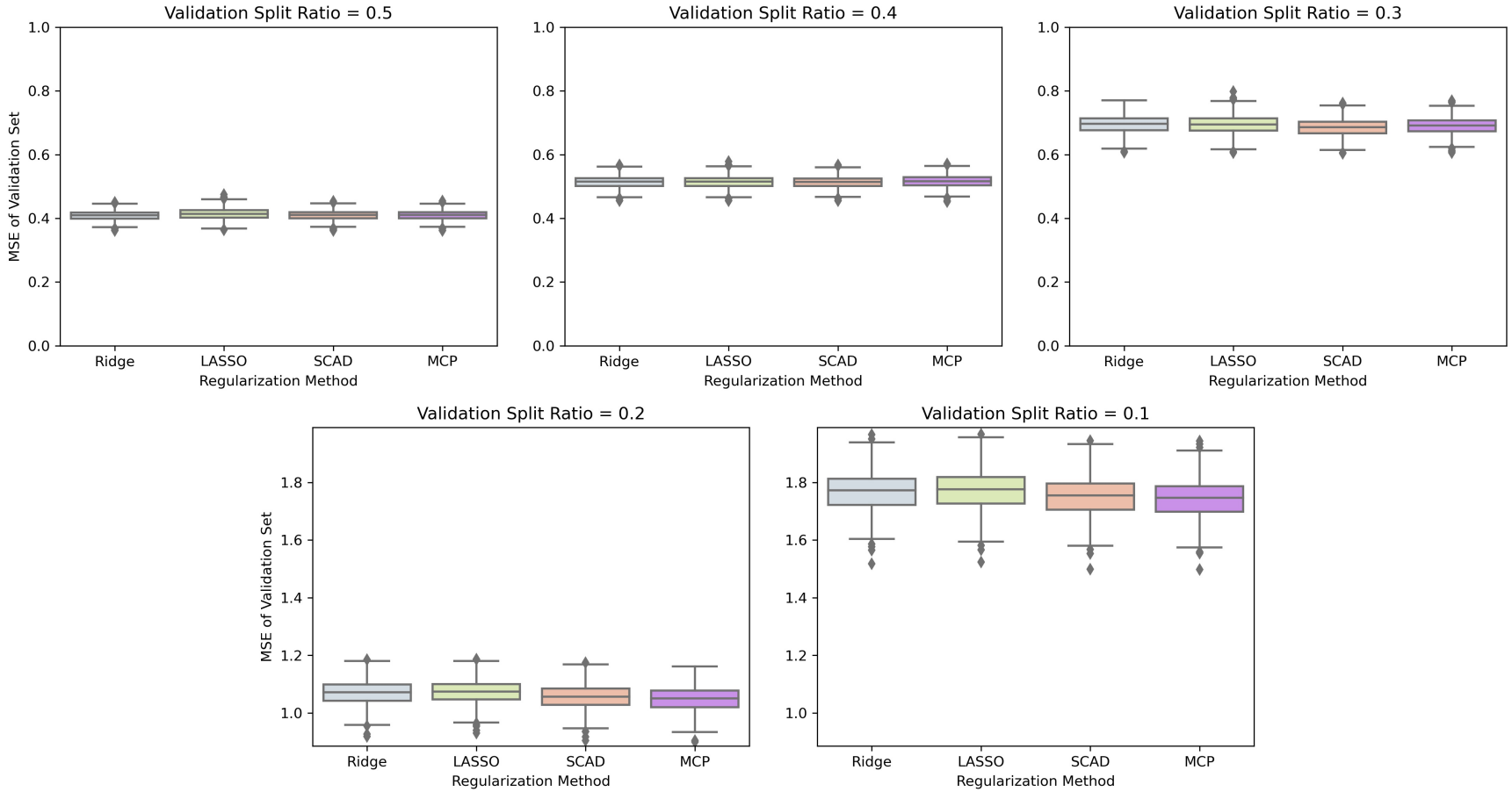

Figure A.15 : The distribution of MSE validation under Scenario 3 for the Insurance data set.

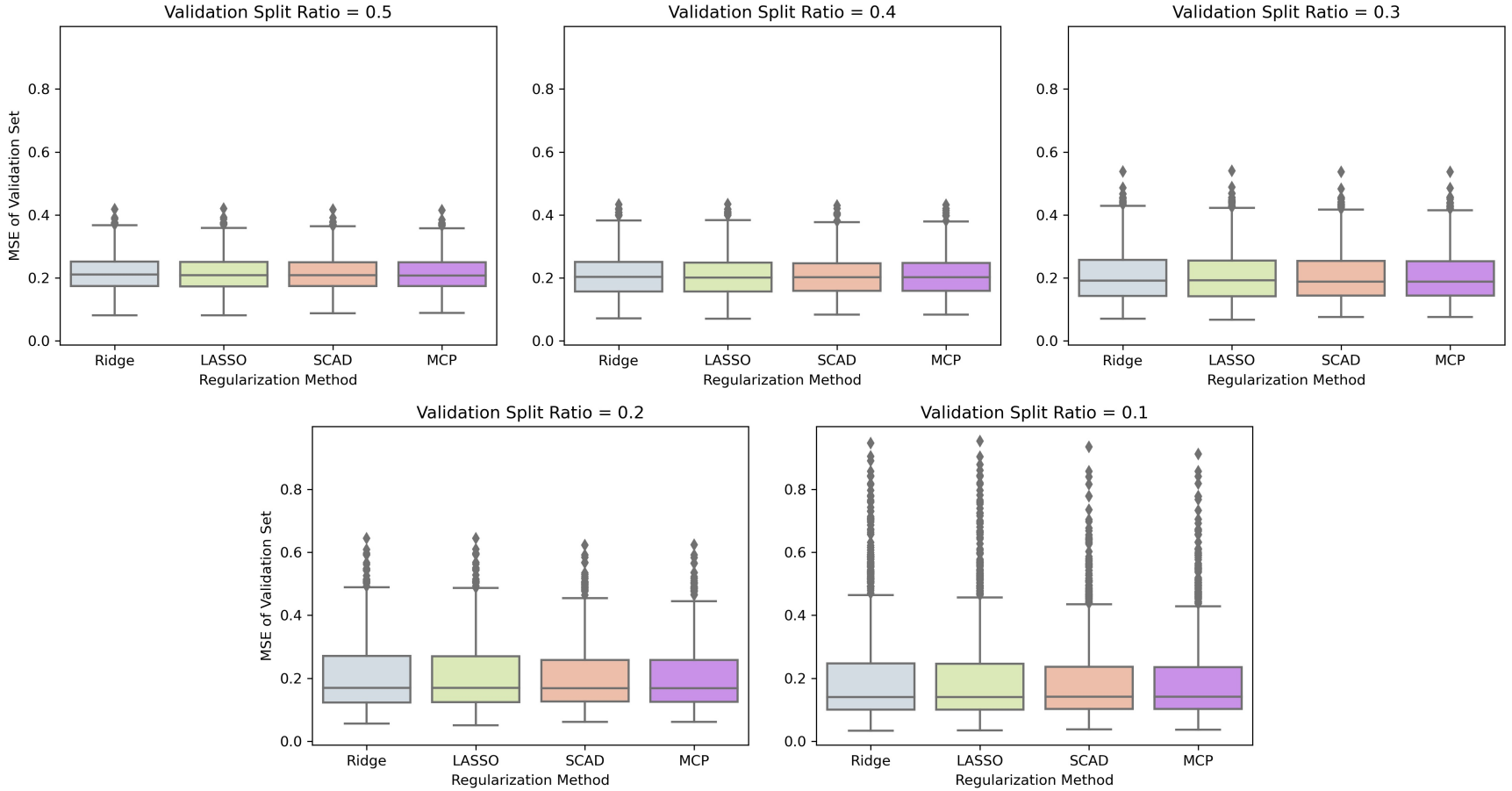

Figure A.16 : The distribution of MSE validation under Scenario 1 for the Credit data set.

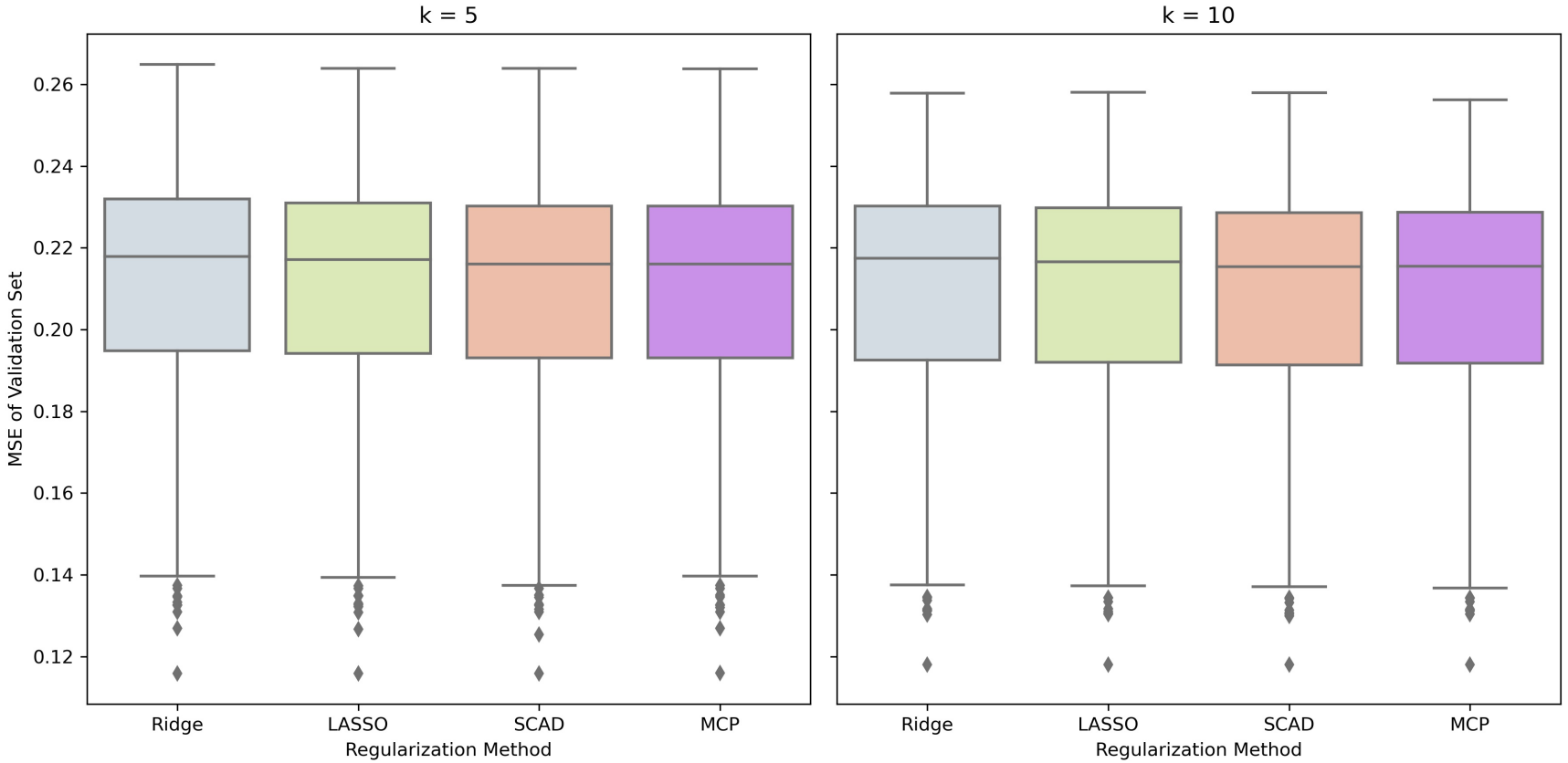

Figure A.17 : The distribution of MSE validation under Scenario 2 for the Credit data set.

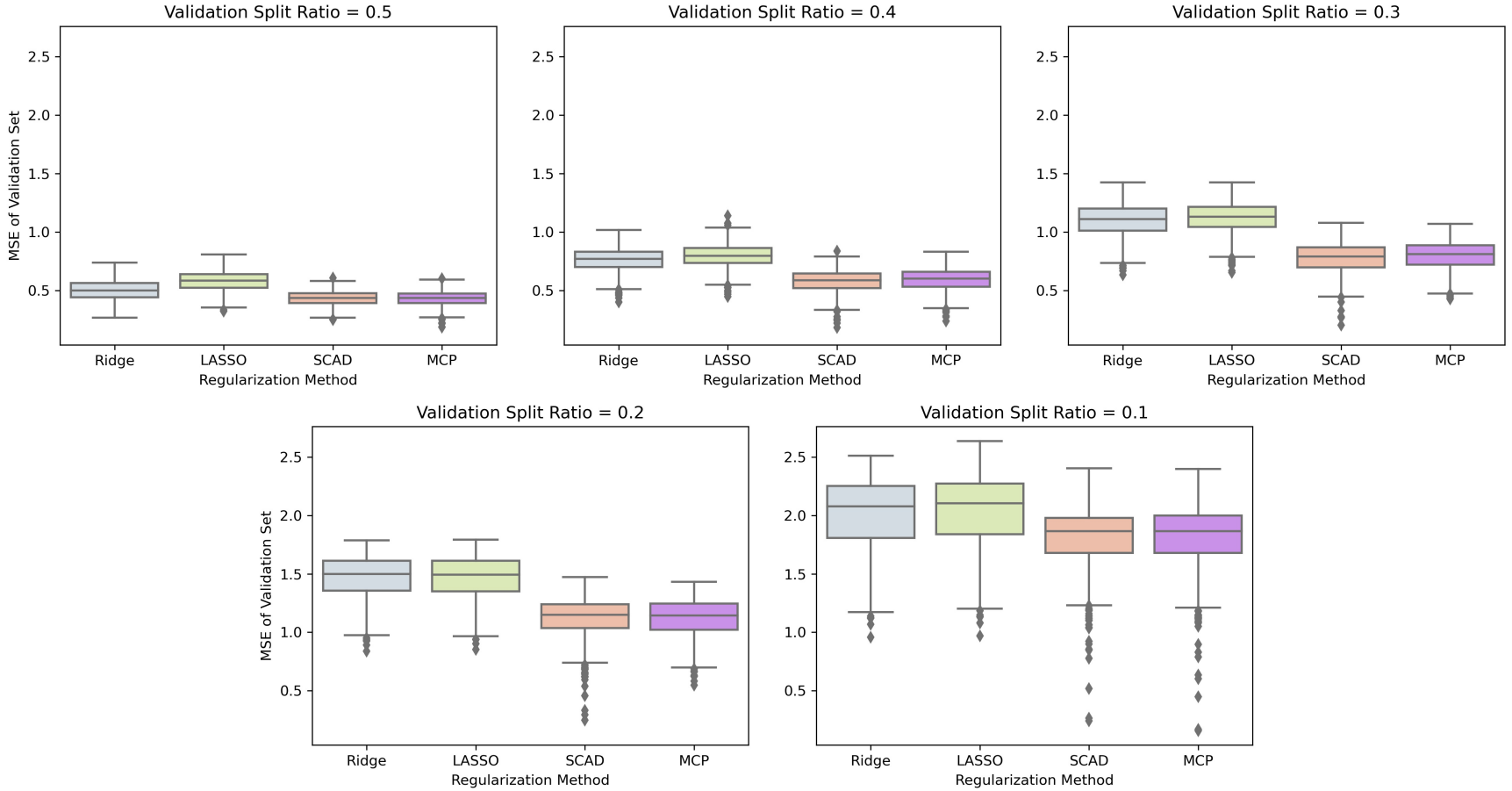

Figure A.18 : The distribution of MSE validation under Scenario 3 for the Credit data set.

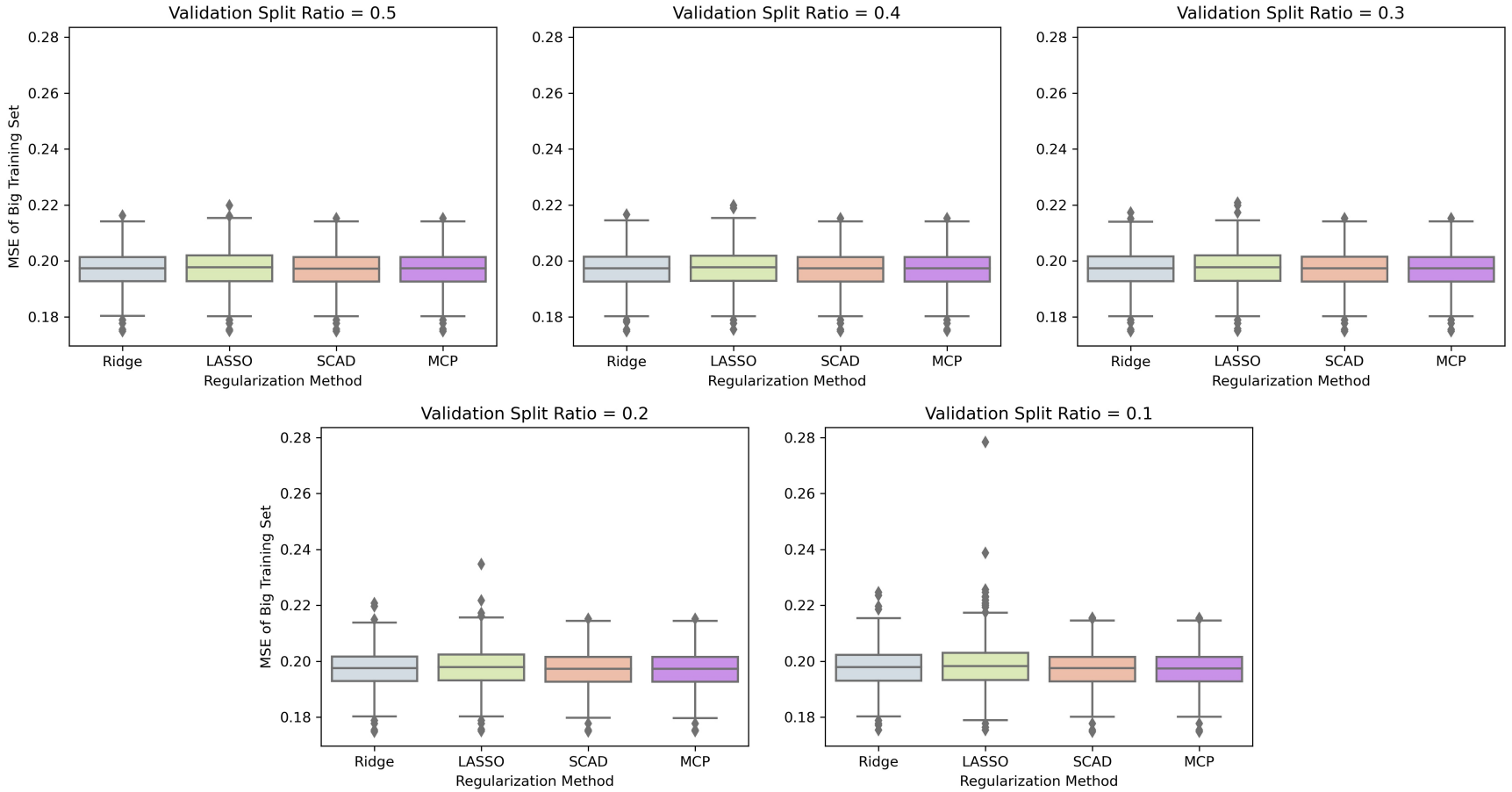

Figure A.19 : The distribution of MSE big train under Scenario 1 for the Insurance data set.

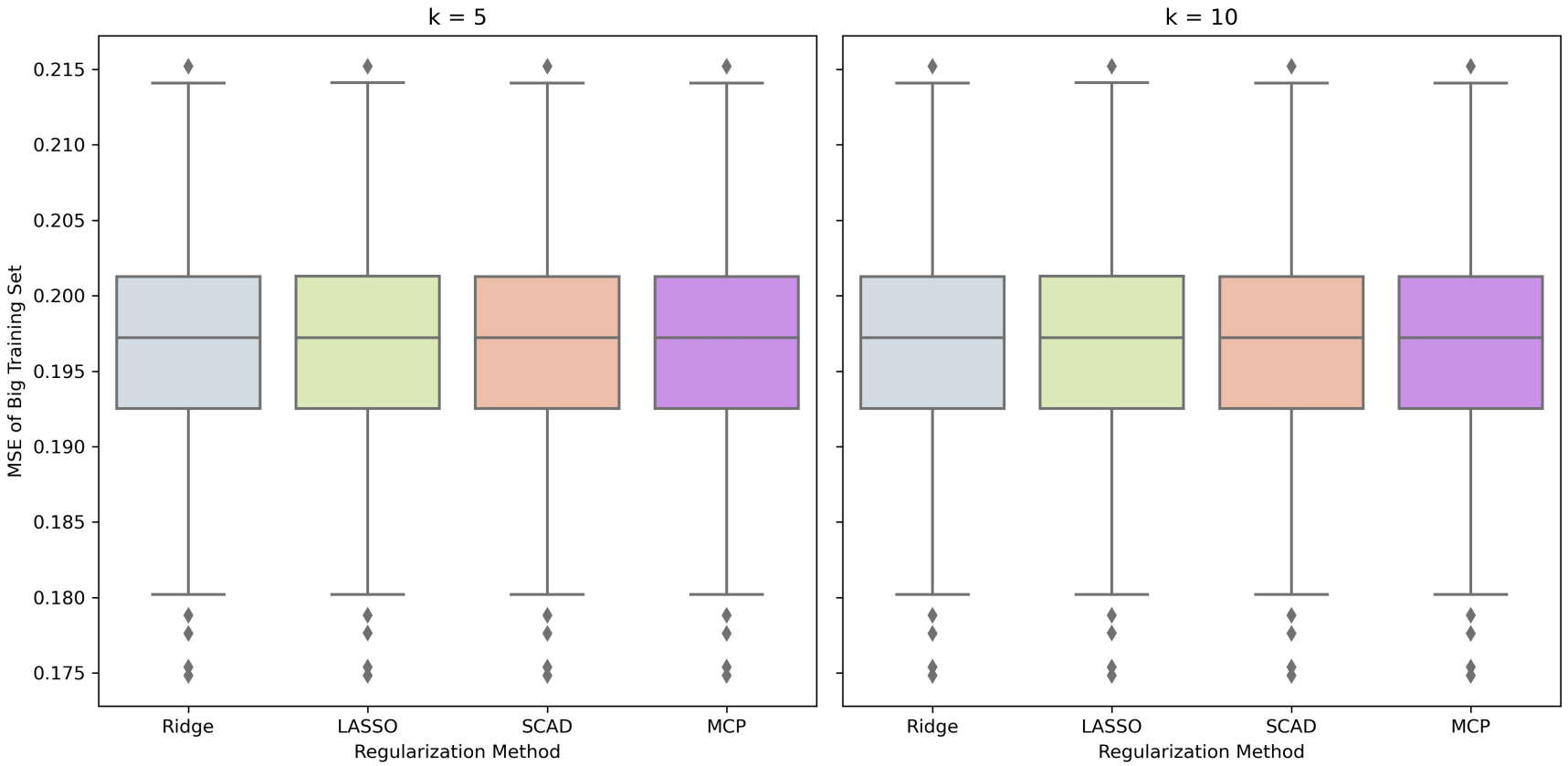

Figure A.20 : The distribution of MSE big train under Scenario 2 for the Insurance data set.

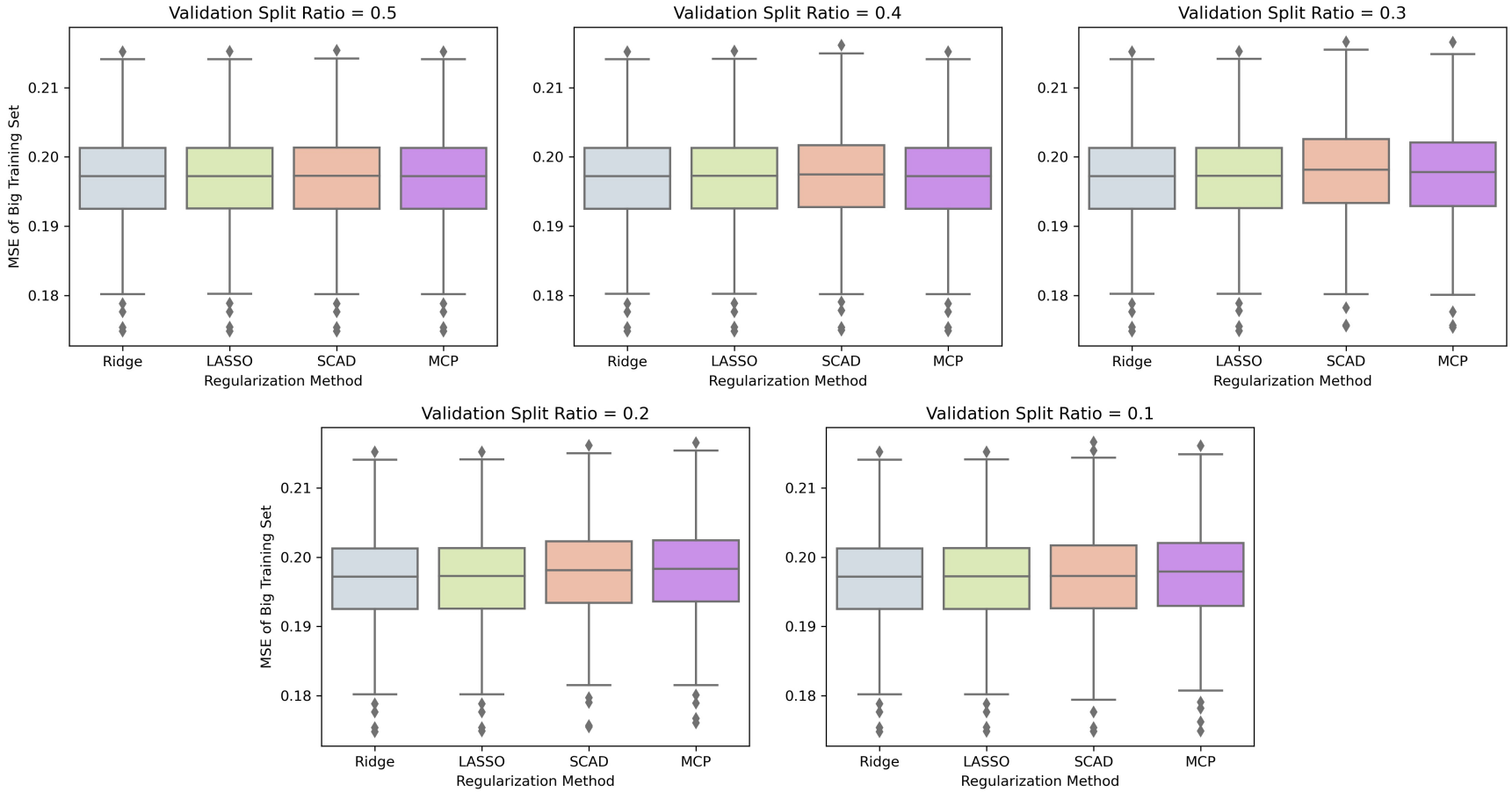

Figure A.21 : The distribution of MSE big train under Scenario 3 for the Insurance data set.

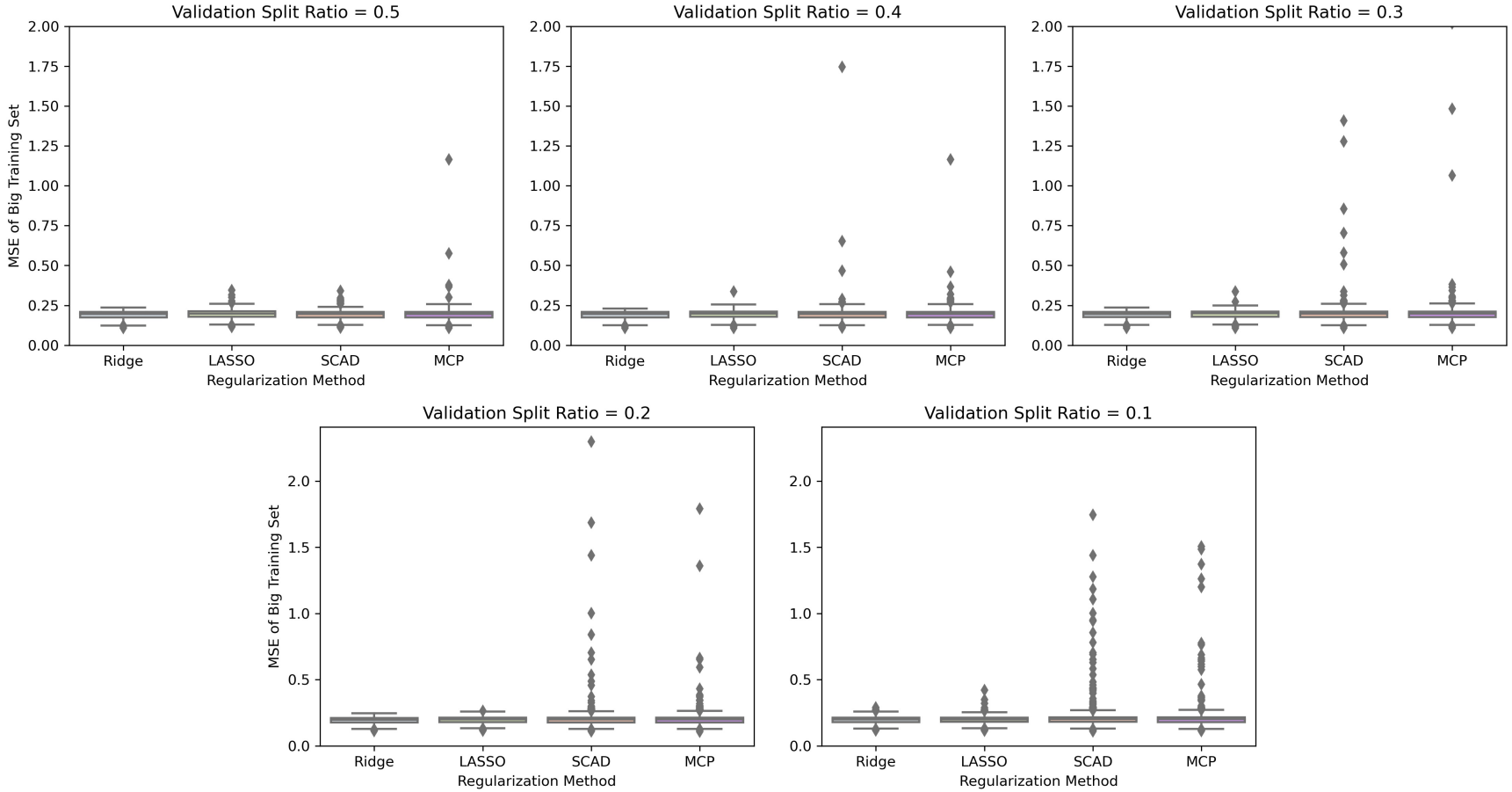

Figure A.22 : The distribution of MSE big train under Scenario 1 for the Credit data set.

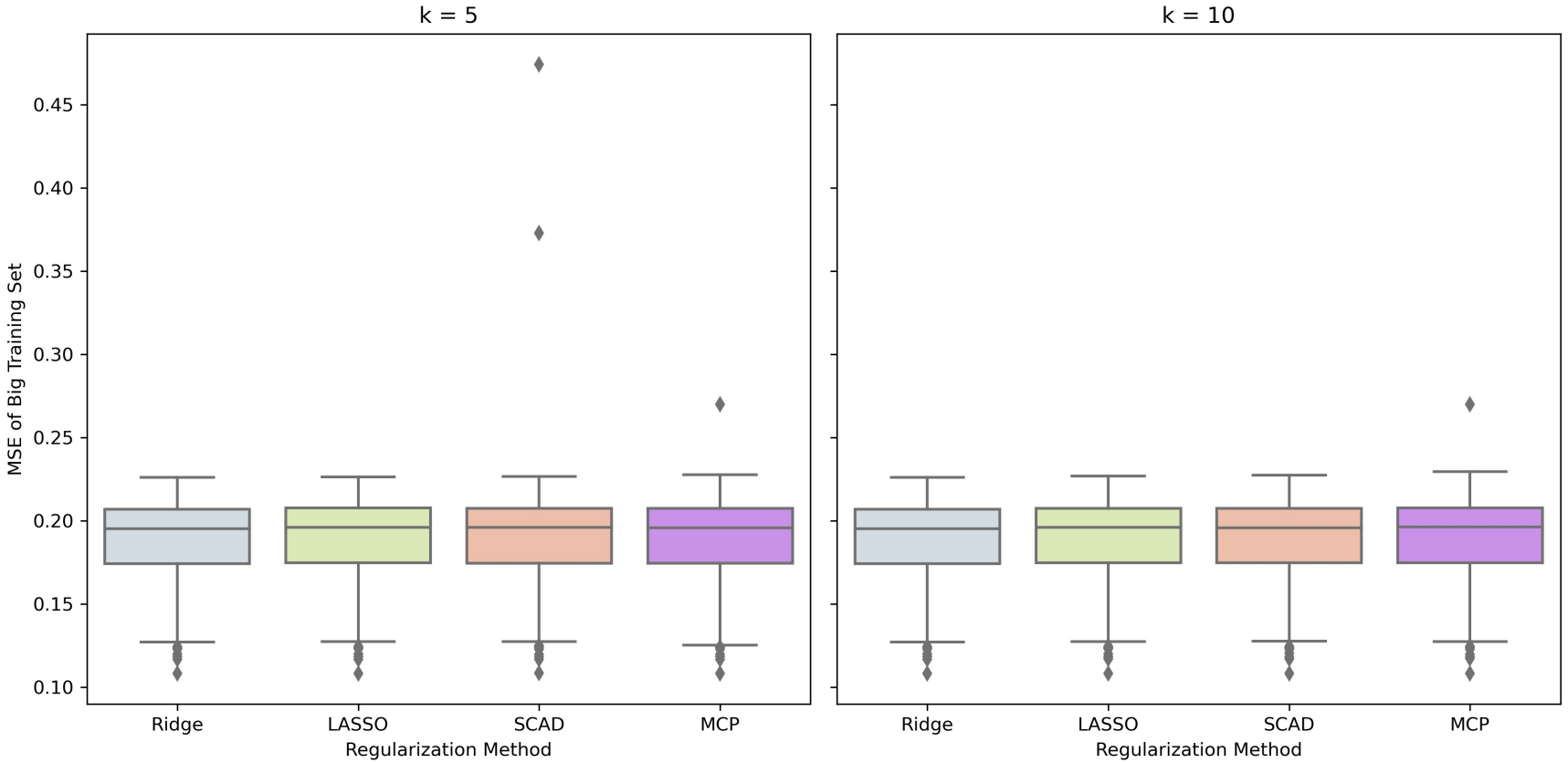

Figure A.23 : The distribution of MSE big train under Scenario 2 for the Credit data set.

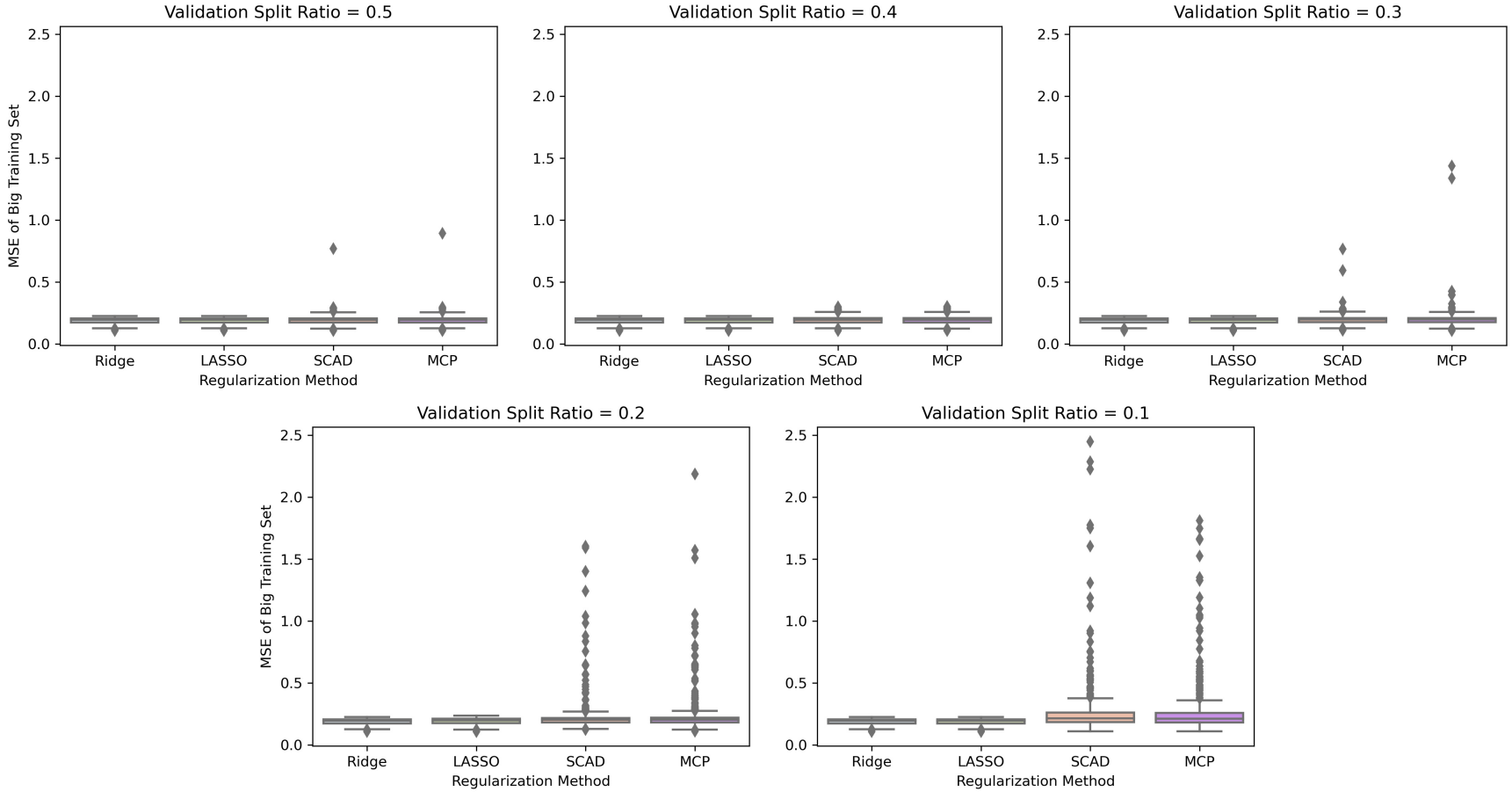

Figure A.24 : The distribution of MSE big train under Scenario 3 for the Credit data set.

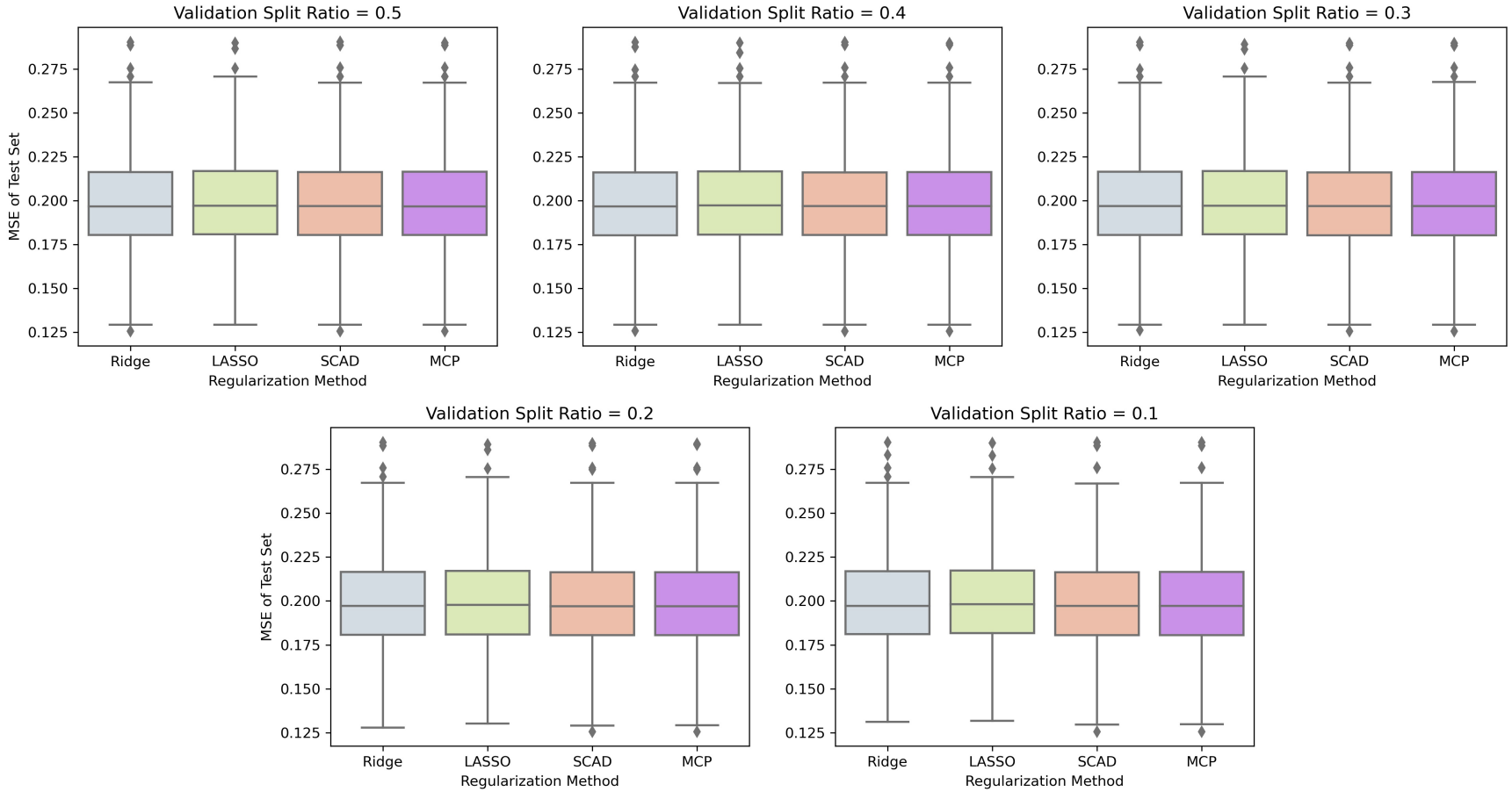

Figure A.25 : The distribution of MSE test under Scenario 1 for the Insurance data set.

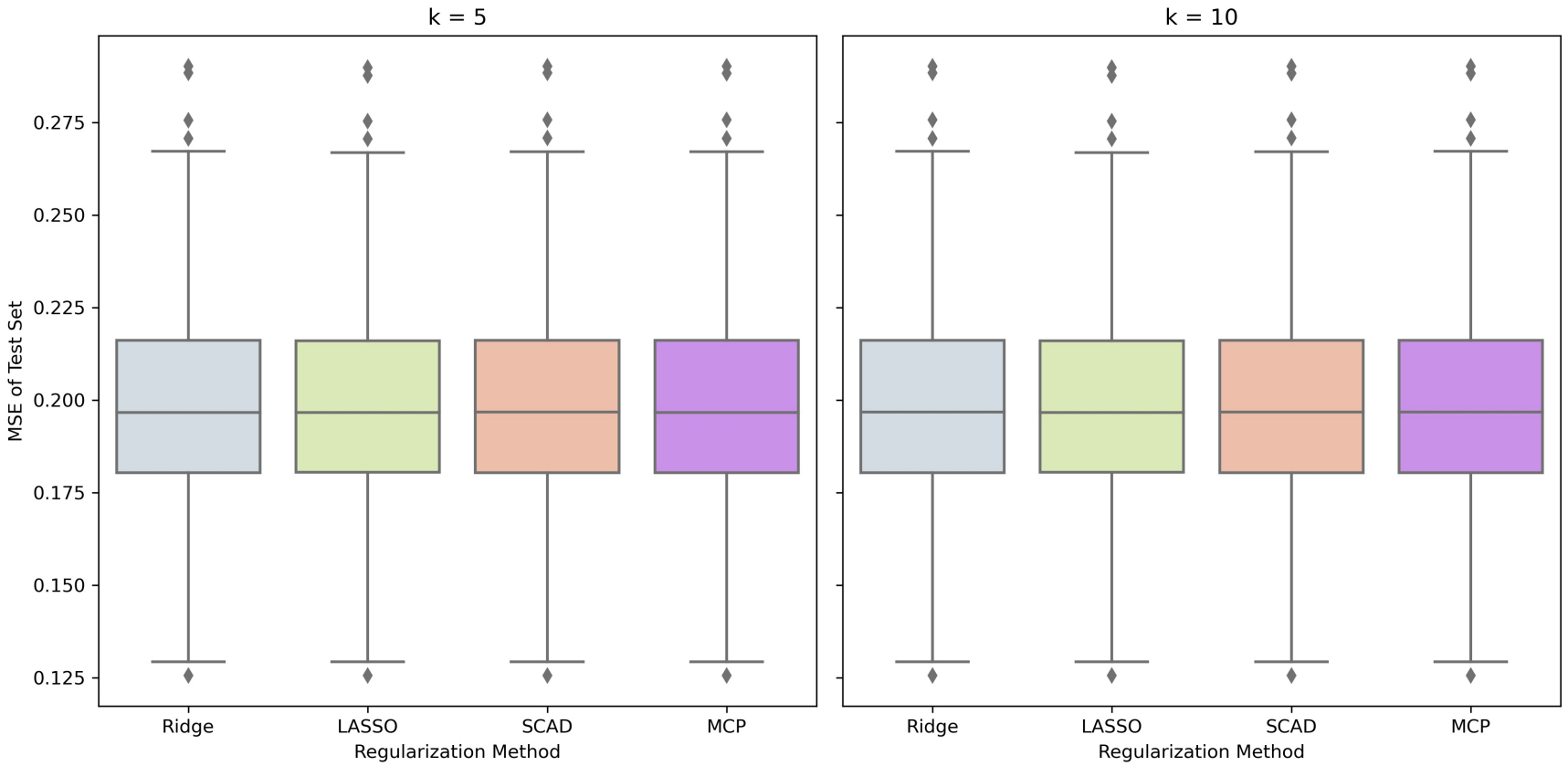

Figure A.26 : The distribution of MSE test under Scenario 2 for the Insurance data set.

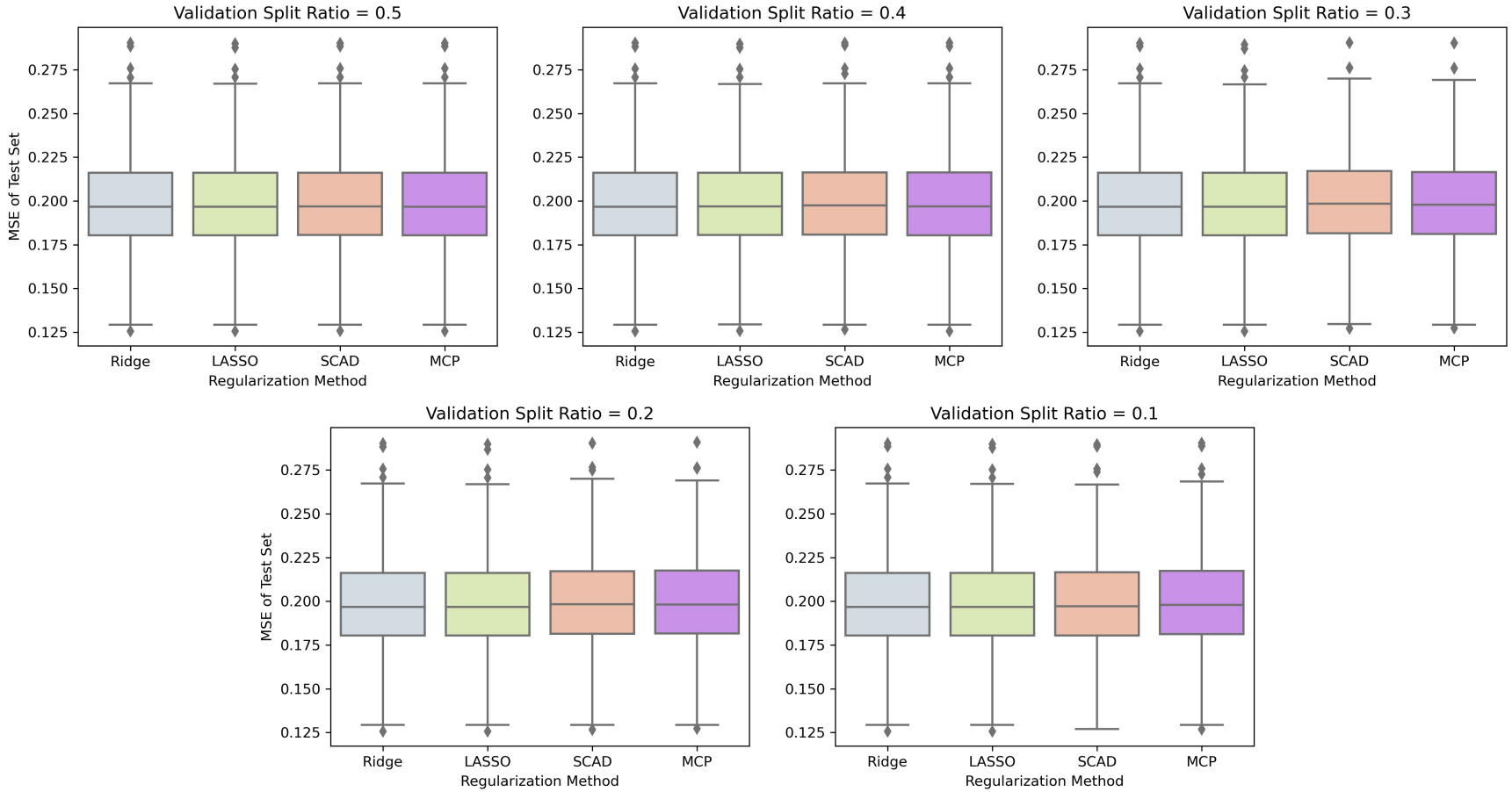

Figure A.27 : The distribution of MSE test under Scenario 3 for the Insurance data set.

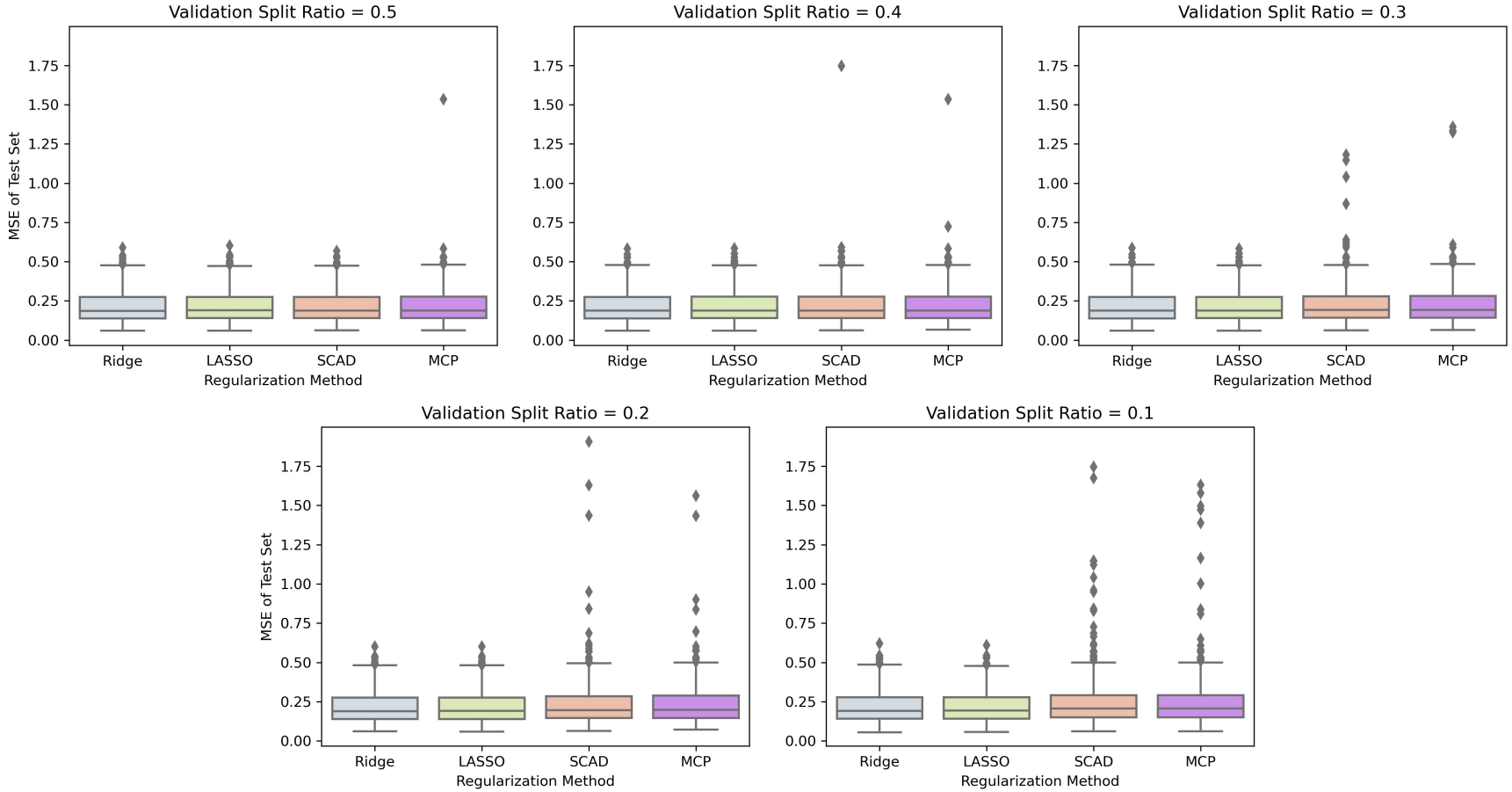

Figure A.28 : The distribution of MSE test under Scenario 1 for the Credit data set.

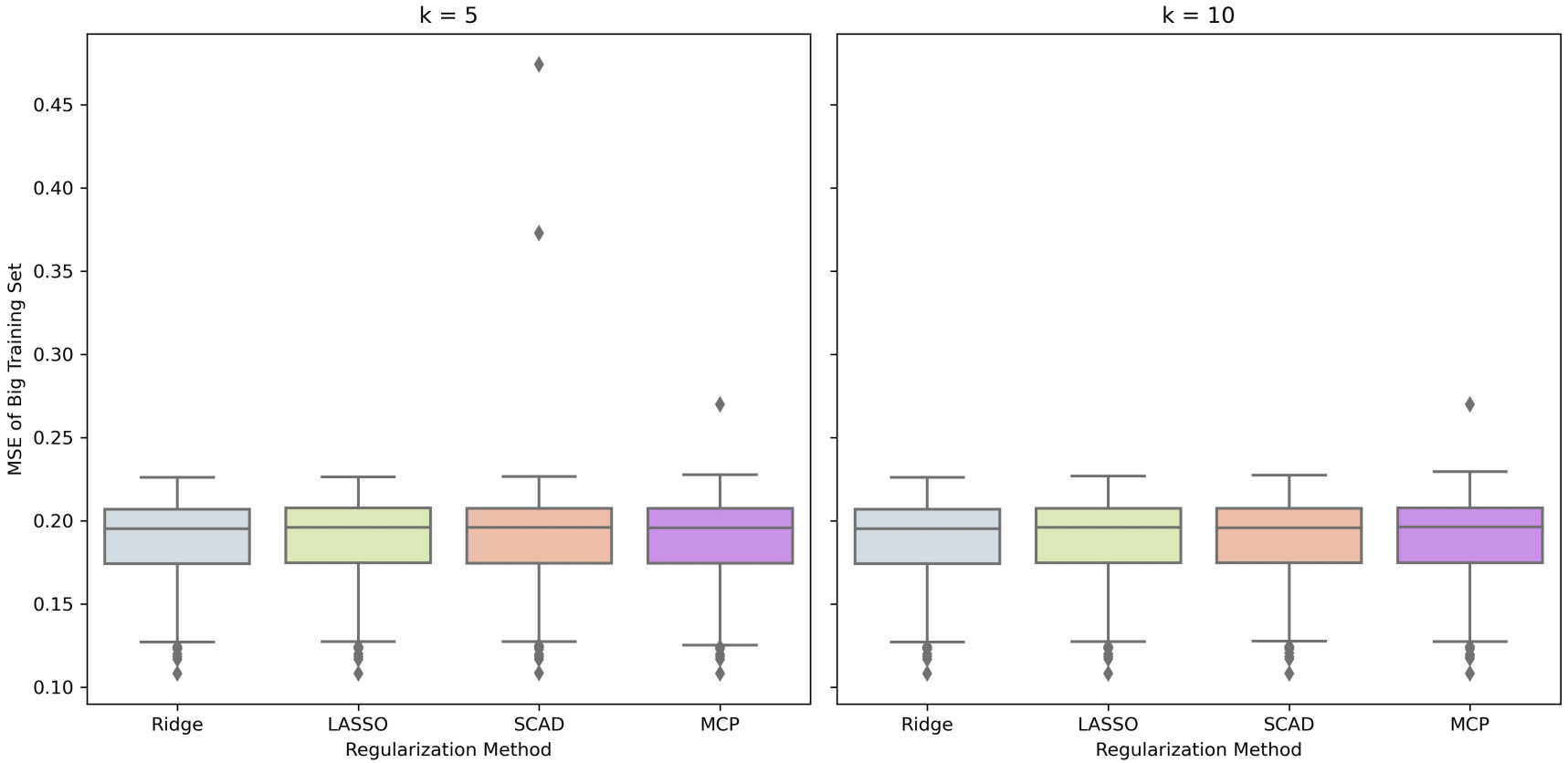

Figure A.29 : The distribution of MSE test under Scenario 2 for the Credit data set.

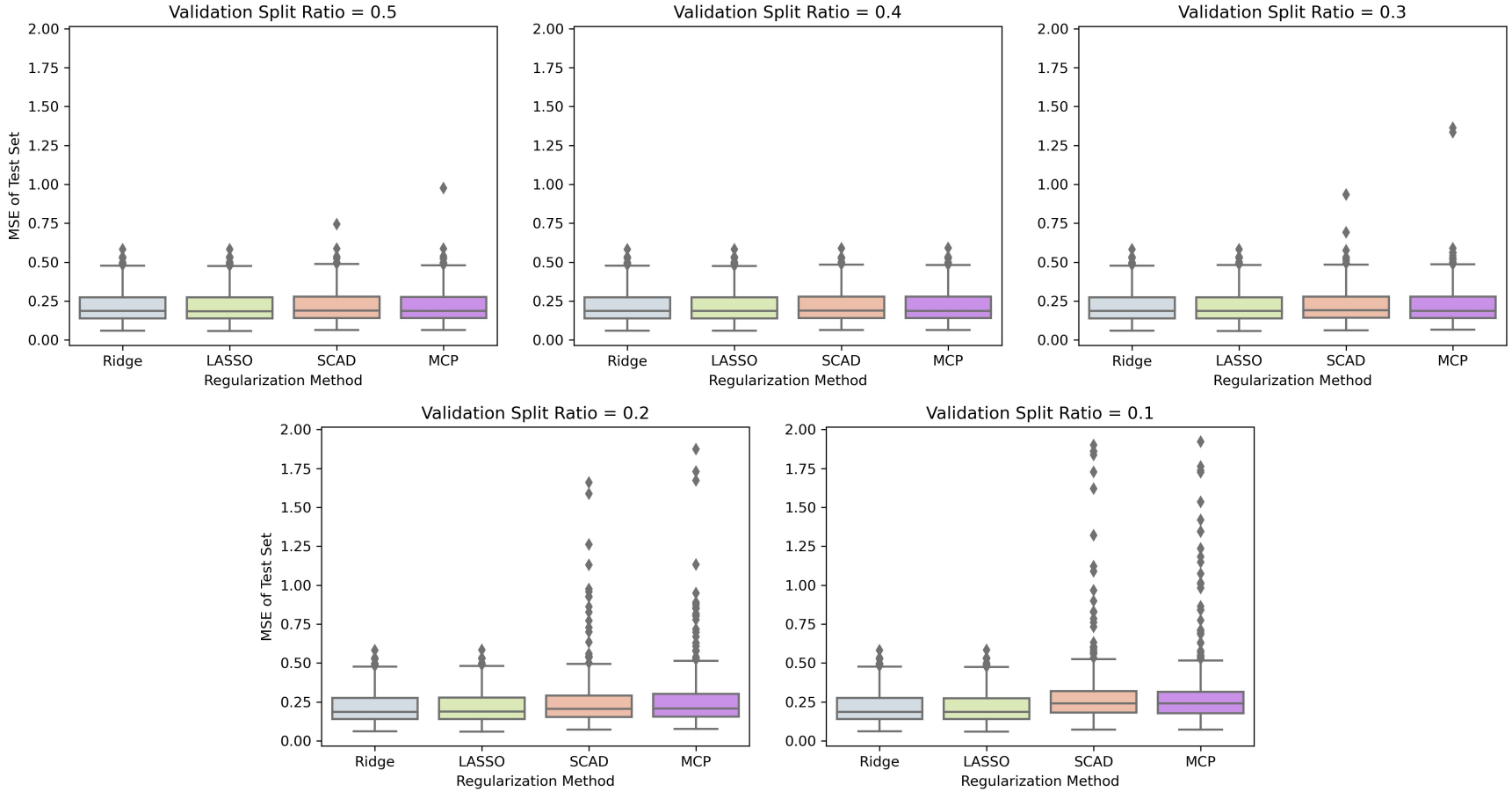

Figure A.30 : The distribution of MSE test under Scenario 3 for the Credit data set.

CURRICULUM VITAE

İrem SARIBAŞ:

EDUCATION:

- **M.Sc.:** 2021-Present, Department of Mathematics Engineering, Graduate School, Istanbul Technical University, Istanbul, Turkey.
- **B.Sc.:** 2016, Department of Mathematics, Faculty of Arts and Sciences, Yıldız Technical University, Istanbul, Turkey.

PROFESSIONAL EXPERIENCE:

- 2023 - Present, Model Risk and Validation Unit, Risk Management Directorate, Anadolu Sigorta, Istanbul, Turkey.

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Sarıbaş İ.**, İnan G. (2024). Penalized Stable Regression. *International Graduate Research Symposium*, May 8-10, 2024 Istanbul, Turkey.