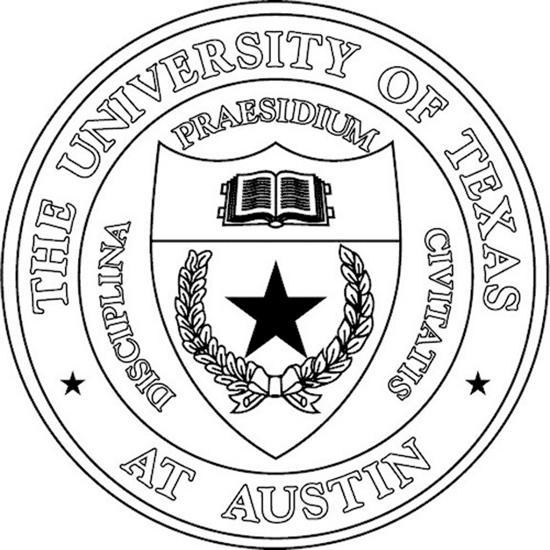
**OPTIMIZING VARIABLE SELECTION IN PREDICTIVE ANALYTICS**

**A Comparative Study of MIQP and LASSO Methods**



**Nov 17, 2023**

**Group 10**

**Barnana Ganguly, Jordan Ehlinger, Santhosh Kumar Ramkumar, Vaibhav Nagar**

Table of Contents

[**1. Introduction**](#_q8p8nsv5w9wf) **2**

[**2. Methodology**](#_ehkai1clh0gd) **3**

[2.1. Data](#_5rlujp8le6wz) 3

[2.2. Direct Variable Selection: The MIQP Approach](#_kz82by4was9) 3

[2.3. Indirect Variable Selection: The LASSO Approach](#_mfcuw1bnmozw) 4

[2.4. Cross-Validation](#_z6q5q61rpehr) 5

[2.4.1 Manual Cross Validation for MIQP](#_6axrj5ewg9q) 5

[2.4.2 Automated K-Fold Cross Validation for LASSO](#_jb0gnamthn9r) 5

[**3. Code and Results**](#_jzil4zt85rjg) **6**

[3.1. MIQP](#_x9jtfo15nmex) 6

[3.2. LASSO](#_f370r4aavt5) 14

[**4. MIQP vs LASSO - A Comparison**](#_73x50vhqsru) **17**

[4.1 Accuracy](#_t5vwxsvyd4i5) 17

[4.2 Variable Selection](#_mwkv7r6i02a4) 18

[4.3 Comparison of Coefficients](#_bko9208cyrxi) 19

[4.4 Computational Efficiency](#_e3cbfdqvld0y) 20

[4.5 Summary of Comparison](#_slrdwk6zvbgj) 22

[**5. Conclusion and Recommendations**](#_z1o3ib14bqo8) **22**

# **1. Introduction**

In the field of predictive analytics, variable selection plays a pivotal role in enhancing the performance of regression models. This project is centered around comparing two distinct approaches to variable selection: **Mixed Integer Quadratic Programming (MIQP**) and the **Least Absolute Shrinkage and Selection Operator (LASSO)**.

MIQP is a mathematical optimization technique used for selecting variables in a model to optimize a quadratic objective under linear constraints.

LASSO was originally developed in response to the computational challenges associated with direct variable selection methods. It is a regression analysis method that performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the resulting statistical model.

Its appeal lies in its computational efficiency, making it particularly suitable for handling large datasets or scenarios with constrained computing resources. In addition, its shrinkage component, beneficial for managing overfitting through regularization, adds to its appeal.

However, advancements in computational technology have renewed interest in direct variable selection methods like MIQP. This study aims to compare the effectiveness of MIQP and LASSO, evaluating the contexts where each method excels based on computational efficiency and the advantages of shrinkage in variable selection.

The methodology involves applying both MIQP and LASSO to the same datasets, comprising a training and a test set. The primary focus is on optimizing the critical parameters: **the number of variables (k) for MIQP** and **the regularization parameter (λ) for LASSO**, using **10-fold cross-validation**.

The objective of this study is to determine whether LASSO's 'shrinkage' component, which also helps in reducing overfitting, is more beneficial than the direct selection of the most suitable variables using MIQP. This comparative analysis aims to provide insights into the relative advantages of each approach, thereby contributing to both the theoretical understanding and practical application of variable selection in the field of predictive analytics.

# **2. Methodology**

## **2.1. Data**

This study utilizes two distinct datasets containing a set of 50 independent variables (X) and a dependent variable (y), each serving specific purposes in the analysis:

**The Training Dataset (training\_data.csv):**

* **Composition:** Consists of 250 rows.
* **Purpose:** The training dataset is used for building the regression models and optimizing their hyperparameters. It is essential for training both the MIQP and LASSO models, determining the optimal number of variables (k) for MIQP, and the regularization parameter (λ) for LASSO.

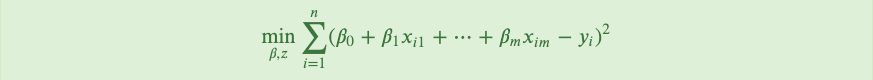
**Testing Dataset (test\_data.csv)**:

* **Composition:** Comprises 50 rows.
* **Purpose:** The testing dataset is employed to evaluate the performance of the trained models. It is used to test the models' ability to predict y values accurately and to compare these predictions with the true y values, thus assessing the effectiveness of the models on new, unseen data.

## **2.2.** **Direct Variable Selection: The MIQP Approach**

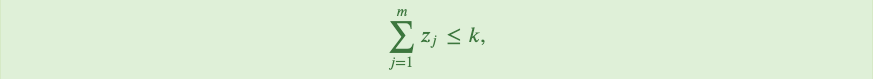
The core idea behind Mixed Integer Quadratic Programming (MIQP) is to pose the variable selection problem as an optimization task. Given a dataset of m independent variables (X) and a dependent variable (y), we seek to identify the optimal subset of variables to include in our regression model while minimizing the least squares error.

To incorporate variable selection into this problem, binary variables, 𝑧𝑗, are introduced. These force the corresponding values of 𝛽𝑗 to be zero if 𝑧𝑗 is zero, using the big-M method. If we want to include at most 𝑘 variables from 𝑋, then we can pose this as:



subject to:





Note that 𝑚 and 𝑀 are different entities; 𝑚 represents the number of independent variables while 𝑀 is a large constant used in the big-M method. This Big-M value must be large enough so as not to limit the range of the beta coefficients.

In MIQP, we don't restrict the model from having an intercept term (𝛽0) because forcing 𝛽0 to be zero could lead to biased and inaccurate model predictions, especially when an intercept is theoretically justified or necessary for accurate regression.

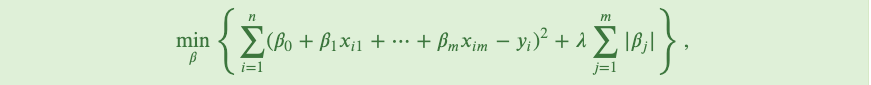
The hyperparameter 𝑘, representing the maximum number of variables to be included, can be chosen using cross-validation.

## **2.3. Indirect Variable Selection: The LASSO Approach**

**LASSO,** or **Least Absolute Shrinkage and Selection Operator,** is a robust technique for variable selection in regression analysis, adept at identifying the most relevant features while reducing overfitting.

LASSO's standout advantage lies in its automatic feature selection ability, autonomously identifying and retaining influential variables for predicting the target variable and discarding less relevant ones. It combats overfitting through a shrinkage effect, which pulls coefficients closer to zero, enhancing model accuracy and reliability.

LASSO does all of this by introducing a penalty term to the traditional least squares regression problem. The objective is to minimize the sum of squared errors, like ordinary least squares, while adding a penalty based on the absolute values of the regression coefficients, excluding the intercept term. This penalty term encourages some coefficients to become exactly zero, effectively selecting a subset of the available variables.



Crucially, LASSO retains the intercept term (β0), ensuring that the model includes a baseline value. The hyperparameter λ controls the trade-off between regularization strength and model fit. The selection of an appropriate λ value is often determined through cross-validation.

In our analysis, we implement the LASSO regression model using the scikit-learn library, a widely used Python toolkit for machine learning. The next sections of this report will present a comparative analysis of the results obtained from LASSO and the direct variable selection approach using MIQP. This analysis will shed light on whether LASSO's unique combination of feature selection and shrinkage is more advantageous for our specific regression problem.

## **2.4. Cross-Validation**

### **2.4.1 Manual Cross Validation for MIQP**

**Adapting to MIQP's Unique Requirements:**

MIQP presents unique challenges in cross-validation due to its nature as an optimization technique, as opposed to standard predictive models. Currently, there isn't a standard package designed for applying K-Fold Cross-Validation (KFold) directly to MIQP problems. This stems from the distinct purposes of MIQP and KFold: MIQP focuses on finding optimal solutions under constraints, while KFold is used for assessing a model's generalizability.

**Implementing a Manual Approach:**

Given these constraints and the lack of a standard tool, a manual cross-validation method was devised for MIQP. This involved randomly shuffling the training dataset, followed by a division into 10 equal-sized segments, each serving as a validation set in turn.

For each specified k value, the MIQP model was applied to the respective training set segments, and its performance was assessed on the validation set.

This approach allowed for a detailed evaluation of MIQP's predictive ability across different data subsets, aligning with the general principles of cross-validation but tailored to the specificities of MIQP.

### **2.4.2 Automated K-Fold Cross Validation for LASSO**

**Standard K-Fold for LASSO:**

Unlike MIQP, the LASSO regression model is well-suited for the standard K-Fold cross-validation method. LASSO, being a predictive modeling technique, aligns with the objectives of KFold, which evaluates model performance on unseen data.

**Cross Validation Process:**

The LassoCV function from scikit-learn was employed to perform 10-fold cross-validation, automatically determining the optimal value of the regularization parameter alpha (λ). Prior to applying KFold, the training data was normalized using StandardScaler, a necessary preprocessing step for LASSO regression. This automated process efficiently optimized the LASSO model, ensuring a thorough assessment of its predictive accuracy.

# **3. Code and Results**

## **3.1. MIQP**

In this coding section of the report, we detail the practical implementation of MIQP in feature selection for predictive modeling, structured in the following key steps:

1. **Define the MIQP Optimization Function:** This involves setting up an optimization function specific to MIQP, which selects the most relevant features from the dataset.
2. **Define the Manual Cross-Validation Function:** A crucial step in the process, manual cross-validation ensures the robustness and reliability of the feature selection. This part covers the implementation details for conducting cross-validation manually, iterating over a range of k-values and evaluating the performance across different data subsets.
3. **Determine the Optimal Number of Features (k):** This step involves analyzing the results from the cross-validation process to identify the optimal k value. The focus is on striking a balance between model complexity and predictive accuracy, ensuring the model is neither overfit nor underfit.
4. **Utilize the Optimal k Value:** The final step is applying the optimal k value determined from the previous steps to estimate model coefficients using the entire training dataset and then evaluating the model's performance on a separate test set to gauge its predictive power on unseen data.

Let's now dive in and take a closer look at the code and output.

**Step 1: Define the MIQP Optimization Function**

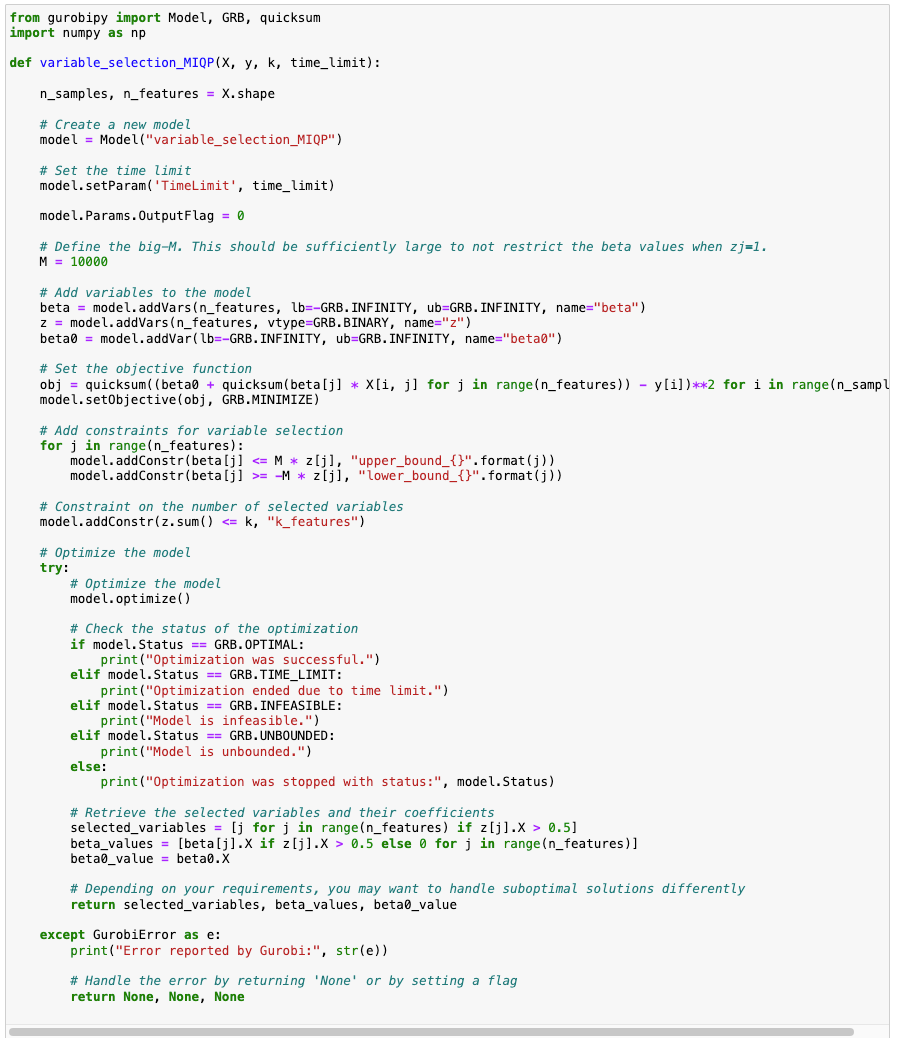
We define a function to perform the MIQP optimization, specifying the following parameters:

1. **X**: independent variables
2. **y**: dependent variable
3. **k**: maximum number of variables to select
4. **time\_limit**: maximum time allowed for the optimization process

As mentioned earlier, the Big-M value must be large enough so as not to limit the range of the beta coefficients. So we picked a value of 10000.

This function returns **the selected variables, their corresponding coefficients, and the intercept term**.

**Code:**



**Step 2: Define the Manual Cross-Validation Function**

We implement a function for conducting manual cross-validation for MIQP (for reasons mentioned above).

The parameters include:

1. **data:** This is the dataset that will be used in the process.
2. **target:** This refers to the target variable in the dataset.
3. **k\_values:** A list of k-values that the function will iterate over.
4. **output\_csv:** The filename for the CSV file where results will be written.
5. **total\_time\_limit:** The overall time constraint imposed on the cross-validation process.
6. **n\_folds:** The number of folds to be used in the cross-validation.

This function produces a DataFrame named **results\_df**, which encapsulates the performance details of each fold in the cross-validation process. The DataFrame is structured with the following columns:

1. **Fold:** Identifies the specific fold in the cross-validation.
2. **K:** Represents the value of k used in that fold.
3. **Betas:** Contains the beta coefficients determined during the fold.
4. **Intercept:** The intercept value computed in the fold.
5. **Squared\_Error:** The squared error for the fold's predictions.
6. **Mean\_Squared\_Error:** The mean squared error calculated for the fold.
7. **Fold\_Size:** Indicates the number of data points in the fold.

An important aspect of the Fold\_Size column is its variability. This occurs because when the total number of data points is not perfectly divisible by the number of folds, the leftover data points are allocated to the last fold. As a result, the size of the folds may vary.

To ensure data integrity and continuity, the results from each fold are continuously saved to a specified CSV file. This approach guarantees that no data is lost after each fold's computation.

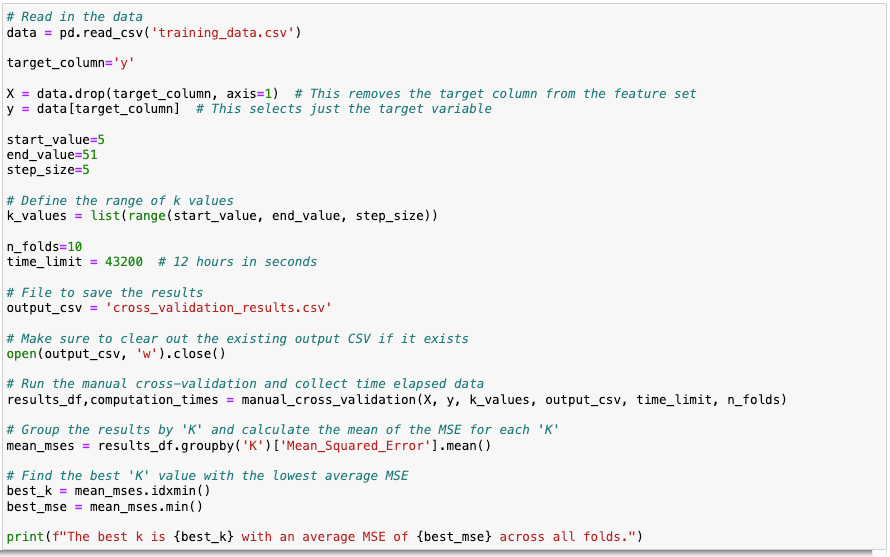


**Step 3: Determine the Best k**

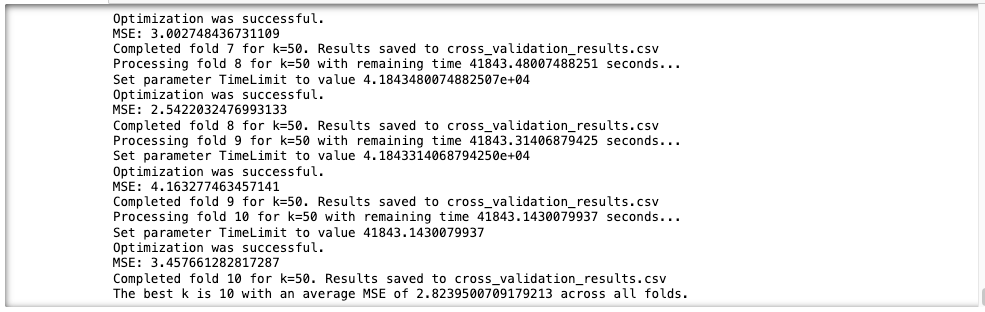
Now that we’ve created the relevant functions, we want to use them to find the optimal value of k. We begin by importing the training dataset. Then, using the defined range of potential k values, we execute a thorough cross-validation procedure.

Our objective is to identify the k that yields the lowest average Mean Squared Error (MSE).

**Code:**



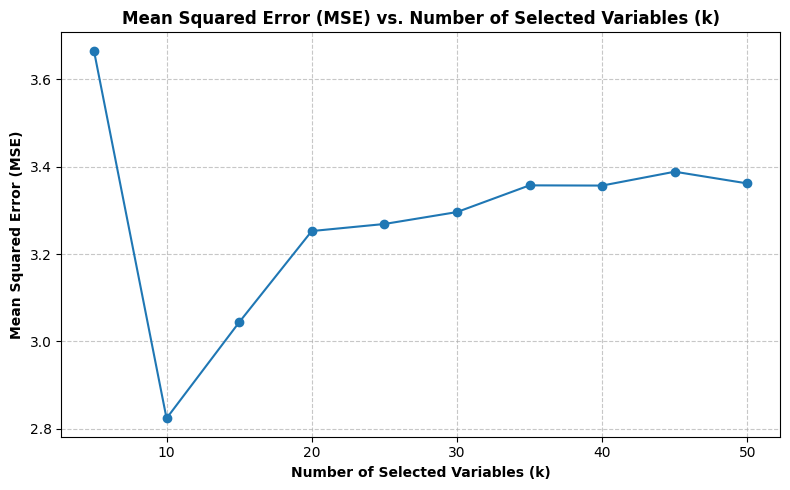
**Output:**

****

Note that the output provided is just a part of the complete result, which includes the model running across all 10 folds for each k value.

The following chart demonstrates the variation in MSE observed across different values of k

**Figure 1:**



***The results indicate that a k of 10 is optimal, achieving an average Mean Squared Error (MSE) of 2.82***

**Step 4: Utilize the Optimal Value of k**

With our optimal k identified, we estimate the model coefficients using the entire training data.

These coefficients are then used to predict the target variable in the test dataset.

The Test MSE provides a measure of the model's predictive power.

**Code:**



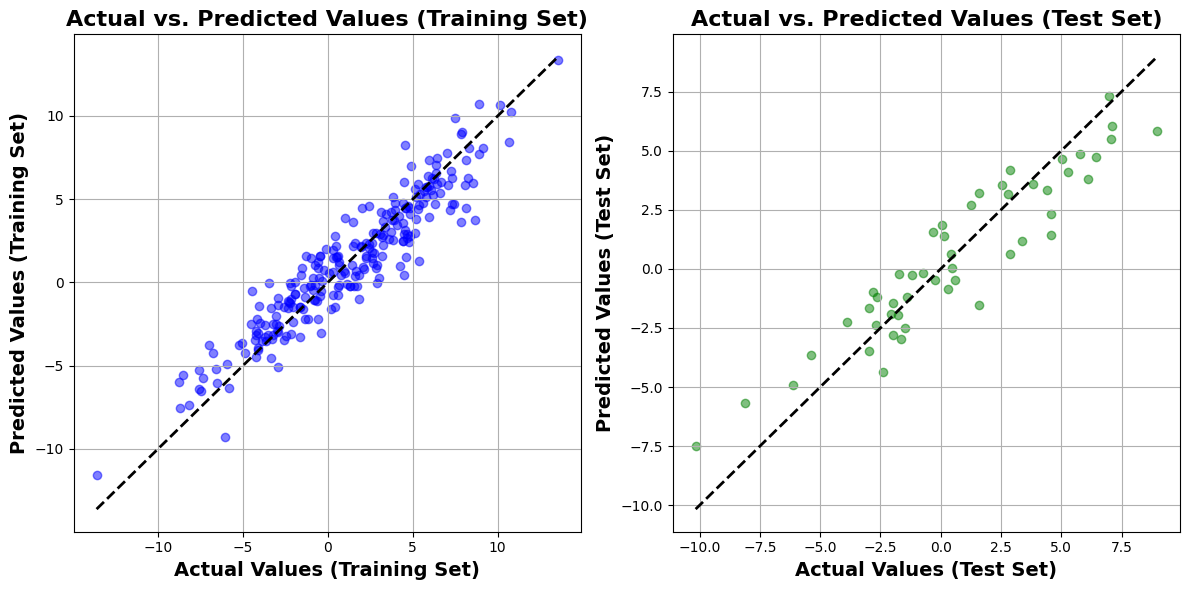
**Output:**

****

***The output shows that the Mean Squared Error (MSE) for the test set, using MIQP optimization, is 2.33***

To further understand and assess the performance of the MIQP model on both the training and test datasets, we turn to the following visualization.

**Figure 2:**



On the left, the scatter plot compares actual versus predicted values for the training set, demonstrating the model's learning capabilities. On the right, a similar plot for the test set highlights the model's ability to generalize to new data. **In both plots, the proximity of points to the diagonal line is indicative of the model's predictive accuracy, a key measure in assessing the success of MIQP models**.

## **3.2. LASSO**

In this section, we present the coding implementation of LASSO Regression, focusing on optimizing the alpha (lambda) parameter and assessing the model's performance through Mean Squared Error (MSE) on the test set.

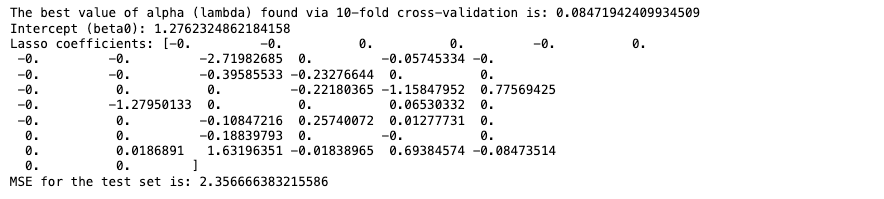
We use KFold from scikit-learn to perform 10-fold cross-validation and identify the optimal λ for our LASSO model. Once the best λ is determined, we fit the model to the entire training dataset.

Next, we use the resulting β coefficients to predict y values on the test set, and finally calculate the Test MSE to assess the model's predictive accuracy.

**Code:**



**Output:**

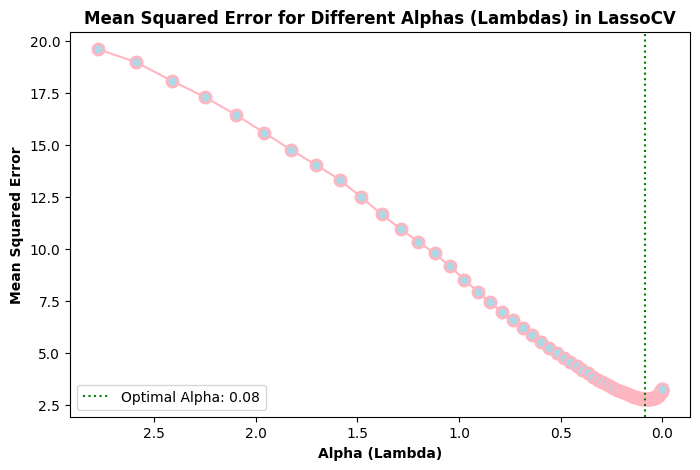
****

***The LASSO regression output reveals an optimal alpha (lambda) value of 0.085 and Test MSE of 2.36***

The 10-fold cross-validation attempts to balance the trade-off between model complexity and fit.

To enhance our understanding and provide a clearer picture of this trade-off, we include the following chart that displays the Mean Squared Error (MSE) for various alpha (lambda) values. This visualization is crucial as it helps better grasp how changes in lambda affect the model's performance, clearly illustrating the relationship between regularization strength and prediction accuracy.

**Figure 3:**



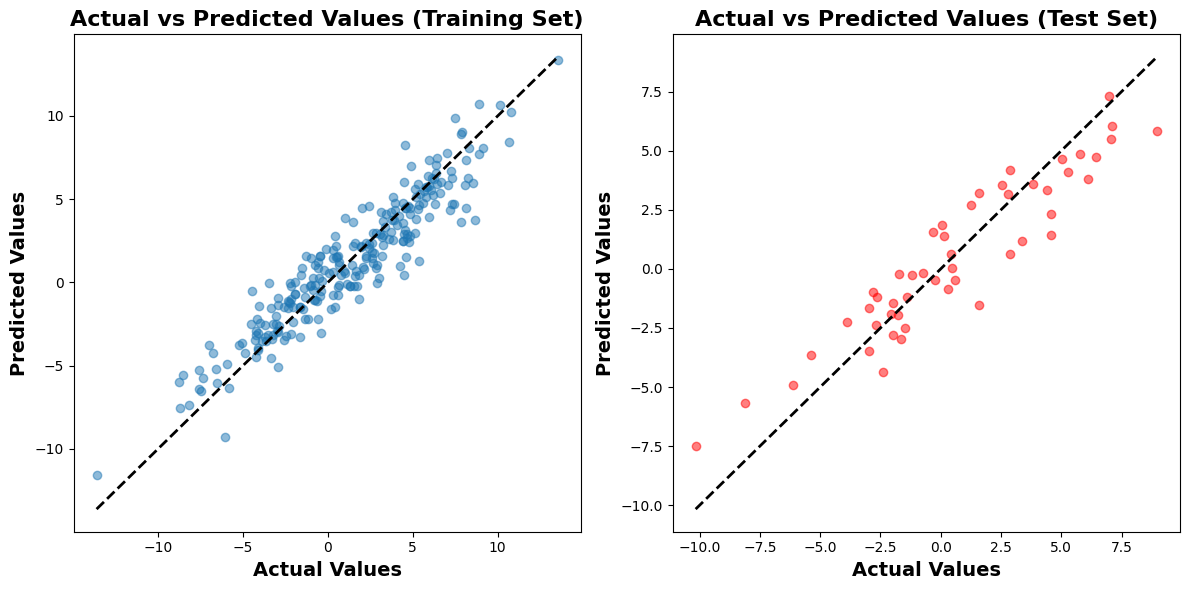
Let’s look at the model output more closely now.

The intercept of the model, β0, is 1.28, setting the baseline prediction.

The LASSO coefficients show a mixture of zero and non-zero values, demonstrating LASSO's feature selection capability by reducing the coefficients of less influential variables to zero. This effectively simplifies the model by focusing on the most significant predictors. Only 18 coefficients were non-zero, signifying their more substantial influence on the dependent variable in the model.

Mirroring the approach taken with the MIQP model, we have created a similar chart for the LASSO regression model.

**Figure 4:**



We observed that the LASSO approach achieves a **Mean Squared Error (MSE) of 2.36** on the test set, demonstrating a high level of accuracy in its predictions.

Note that this is only slightly higher than the Test set MSE of 2.33 obtained by using MIQP.

In the upcoming section, we will compare the LASSO and MIQP approaches in detail.

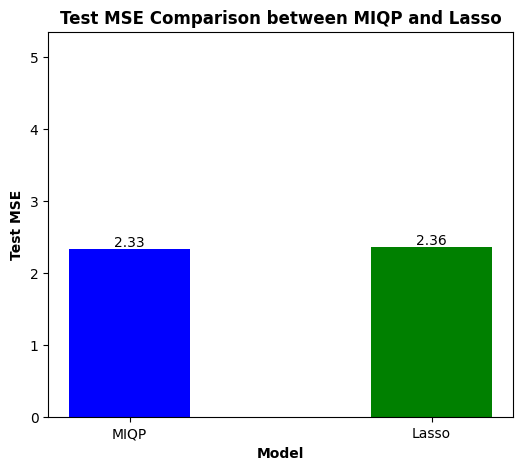
# **4. MIQP vs LASSO - A Comparison**

## **4.1 Accuracy**

A key metric in our analysis for evaluating model performance is the Mean Squared Error (MSE) on the test sets. This measure provides a quantitative foundation for comparing the accuracy of the MIQP and LASSO models.

The results indicate a slight variation in the MSE values between the two approaches. The MIQP model exhibits an MSE of 2.33, marginally outperforming the LASSO model, which has an MSE of 2.36. Although this difference is small, it suggests a slightly higher accuracy in the predictions made by the MIQP model. The below chart demonstrates this.

**Figure 5:**

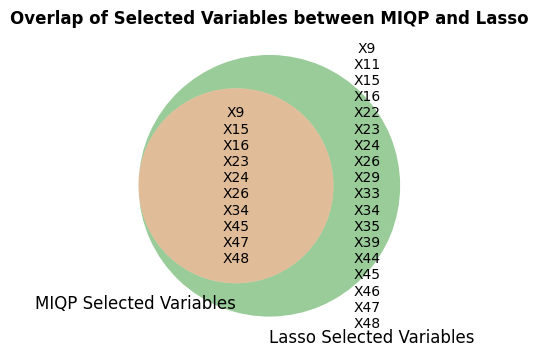


In summary, while the MIQP model demonstrates a marginally higher accuracy as reflected in the MSE values, the difference in performance is relatively minimal.

## **4.2 Variable Selection**

In our comparative analysis of variable selection, the MIQP model opted for a more concise set of variables: **X9, X15, X16, X23, X24, X26, X34, X45, X47, X48**, emphasizing a streamlined approach. Conversely, the LASSO model selected a broader array, including all variables chosen by MIQP plus additional ones like **X11, X22, X29, X33, X35, X39, X44, X46**. This difference highlights MIQP's focus on a smaller, potentially more impactful subset, whereas LASSO's strategy aims to capture a wider spectrum of influences.

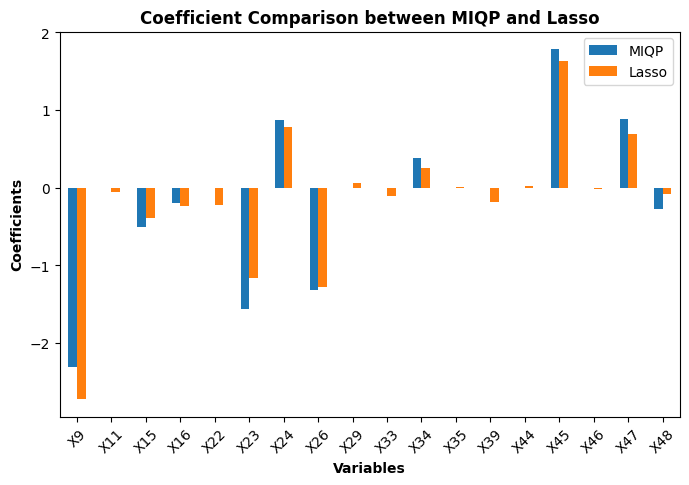
**Figure 6:**



The above Venn diagram visually encapsulates this overlap and distinction in variable selection, illustrating the choices made by each model.

## **4.3 Comparison of Coefficients**

**Figure 7:**

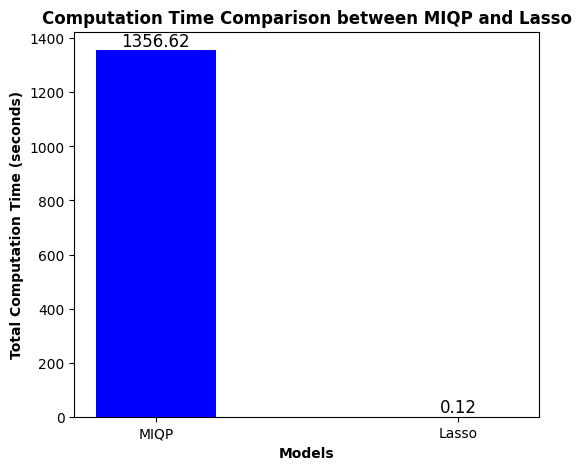


In the comparison of coefficient magnitudes between the MIQP and LASSO models, LASSO demonstrates higher absolute values for 11 variables, while showing lower absolute values for 2 variables. Moreover, for some variables, both models yield coefficients that are either zero or very similar in magnitude, underscoring a level of agreement between the two methods.

The overall trend does not decisively favor one model over the other in terms of consistently producing higher coefficient magnitudes. While Lasso tends to produce non-zero coefficients for more variables, suggesting a broader selection of influential factors, MIQP's approach results in a higher magnitude for certain key variables. The consistency in the sign of coefficients across both models indicates agreement on the direction of influence for these variables.

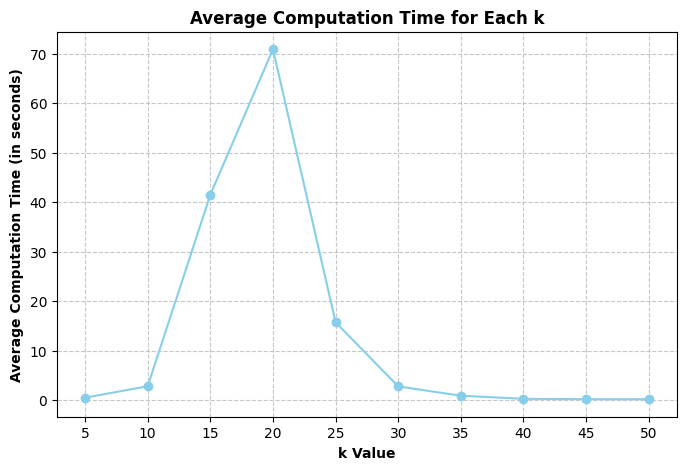
## **4.4 Computational Efficiency**

**Figure 8:**



The comparison between MIQP and LASSO in terms of computational efficiency reveals a clear distinction in their practical applicability. MIQP, known for its nuanced approach in variable selection, incurs significantly longer computation times. This is best visualized in the above chart, which starkly contrasts MIQP's total computation time of 1356.62 seconds against LASSO's almost instantaneous processing of 0.12 seconds.

**Figure 9:**



On further examination we see that MIQP's computation times across different numbers of variables (k) show considerable fluctuation. As illustrated in the above chart, the time required for MIQP peaks notably at k=20 with about 70.94 seconds before decreasing for larger values of k. This nonlinear pattern in computation time with the increase in the number of variables suggests that MIQP can be more efficient for either a very low or high number of variables, while it becomes markedly less efficient for mid-range values.

In contrast, LASSO demonstrates a consistently low computation time, regardless of the number of variables.

## 

## 

## 

## 

## **4.5 Summary of Comparison**

Before proceeding with our recommendations, let's present a table that summarizes the differences between MIQP and LASSO across relevant dimensions:

| **Dimension** | **LASSO** | **MIQP** |
| --- | --- | --- |
| **Computational Efficiency** | Faster, especially with large datasets. Efficient for quick model updates. | Slower, with increased computational time as dataset size grows. |
| **Handling Overfitting** | Provides regularization which helps in reducing overfitting. | Does not inherently provide regularization against overfitting. |
| **Scalability** | Well-suited for high-dimensional data. Scalable with the number of predictors. | May face scalability issues with very large datasets. |
| **Implementation Ease** | Widely supported and easy to implement in various software tools. | Requires sophisticated optimization solvers; more complex to implement. |
| **Flexibility in Variable Selection** | Limited control over the exact number of variables selected. | More explicit control over the number of variables included. |
| **Accuracy** | Generally provides robust predictions. Shrinkage factor might introduce bias in large coefficients. | Potentially more accurate but the difference might be marginal. |

# **5. Conclusion and Recommendations**

Based on our analysis and the insights gained from comparing the MIQP and LASSO models in terms of accuracy, variable selection, coefficient magnitudes, and computational efficiency, we have formulated the following recommendations:

1. **Prioritize LASSO for Efficiency:** In most scenarios, especially where quick processing and computational efficiency are essential, LASSO should be the primary choice. Its ability to handle high-dimensional data swiftly makes it suitable for standard applications.
2. **Use MIQP for Accuracy-Critical Projects:** For projects where utmost accuracy is crucial and computational time and resources are less of a concern, the MIQP model becomes more relevant. This approach is particularly beneficial in situations demanding precise control over variable selection.
3. **Balance Model Selection Based on Needs:** The choice between MIQP and LASSO should be guided by the specific requirements of each project. Factors like the desired level of accuracy, computational limitations, and project timelines should inform the decision-making process, ensuring the most appropriate model is utilized for each task.

To sum up, the decision to use LASSO or MIQP should be carefully aligned with project-specific needs—LASSO for its rapid processing and efficiency in standard settings, and MIQP for its higher accuracy in more complex, detail-oriented tasks. This tailored approach in model selection is key to achieving the most effective and efficient analytical results.

# 

# 

# 

# 

# 

**6. Appendix**

**6.1 MIQP - Output of Step 3 i.e. Determine the Best k**

**The following data represents the results of MIQP optimization across each fold and for every value of k:**

Processing fold 1 for k=5 with remaining time 43199.99999880791 seconds...

Set parameter Username

Academic license - for non-commercial use only - expires 2024-10-17

Set parameter TimeLimit to value 4.3199999998807907e+04

Optimization was successful.

MSE: 4.850142254805167

Completed fold 1 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=5 with remaining time 43199.52303481102 seconds...

Set parameter TimeLimit to value 4.3199523034811020e+04

Optimization was successful.

MSE: 3.0437345642799585

Completed fold 2 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=5 with remaining time 43199.124369859695 seconds...

Set parameter TimeLimit to value 4.3199124369859695e+04

Optimization was successful.

MSE: 3.0590033245513464

Completed fold 3 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=5 with remaining time 43198.65921878815 seconds...

Set parameter TimeLimit to value 4.3198659218788147e+04

Optimization was successful.

MSE: 2.9080995654868

Completed fold 4 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=5 with remaining time 43198.29382300377 seconds...

Set parameter TimeLimit to value 4.3198293823003769e+04

Optimization was successful.

MSE: 4.799771765406785

Completed fold 5 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=5 with remaining time 43197.9025850296 seconds...

Set parameter TimeLimit to value 43197.9025850296

Optimization was successful.

MSE: 3.3468721354361866

Completed fold 6 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=5 with remaining time 43197.31345081329 seconds...

Set parameter TimeLimit to value 4.3197313450813293e+04

Optimization was successful.

MSE: 3.0405835387131197

Completed fold 7 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=5 with remaining time 43196.890333890915 seconds...

Set parameter TimeLimit to value 4.3196890333890915e+04

Optimization was successful.

MSE: 2.4234568824393197

Completed fold 8 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=5 with remaining time 43196.193547964096 seconds...

Set parameter TimeLimit to value 4.3196193547964096e+04

Optimization was successful.

MSE: 5.347921748443977

Completed fold 9 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=5 with remaining time 43195.853942871094 seconds...

Set parameter TimeLimit to value 4.3195853942871094e+04

Optimization was successful.

MSE: 3.8372359129724924

Completed fold 10 for k=5. Results saved to cross\_validation\_results.csv

Processing fold 1 for k=10 with remaining time 43195.28674387932 seconds...

Set parameter TimeLimit to value 4.3195286743879318e+04

Optimization was successful.

MSE: 3.3185880753533

Completed fold 1 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=10 with remaining time 43192.52545309067 seconds...

Set parameter TimeLimit to value 4.3192525453090668e+04

Optimization was successful.

MSE: 2.7217541467842956

Completed fold 2 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=10 with remaining time 43187.07724380493 seconds...

Set parameter TimeLimit to value 4.3187077243804932e+04

Optimization was successful.

MSE: 2.5313693765794185

Completed fold 3 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=10 with remaining time 43184.78655695915 seconds...

Set parameter TimeLimit to value 4.3184786556959152e+04

Optimization was successful.

MSE: 2.5613582652518225

Completed fold 4 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=10 with remaining time 43178.59696292877 seconds...

Set parameter TimeLimit to value 4.3178596962928772e+04

Optimization was successful.

MSE: 3.2634575067347633

Completed fold 5 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=10 with remaining time 43177.08860373497 seconds...

Set parameter TimeLimit to value 4.3177088603734970e+04

Optimization was successful.

MSE: 2.4669426529536493

Completed fold 6 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=10 with remaining time 43176.17920303345 seconds...

Set parameter TimeLimit to value 4.3176179203033447e+04

Optimization was successful.

MSE: 2.60306227268367

Completed fold 7 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=10 with remaining time 43175.22890281677 seconds...

Set parameter TimeLimit to value 4.3175228902816772e+04

Optimization was successful.

MSE: 1.9504673678932605

Completed fold 8 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=10 with remaining time 43172.274497032166 seconds...

Set parameter TimeLimit to value 4.3172274497032166e+04

Optimization was successful.

MSE: 3.8080252088999886

Completed fold 9 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=10 with remaining time 43169.95585465431 seconds...

Set parameter TimeLimit to value 4.3169955854654312e+04

Optimization was successful.

MSE: 3.0144758360450457

Completed fold 10 for k=10. Results saved to cross\_validation\_results.csv

Processing fold 1 for k=15 with remaining time 43167.291279792786 seconds...

Set parameter TimeLimit to value 4.3167291279792786e+04

Optimization was successful.

MSE: 3.4247538341299992

Completed fold 1 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=15 with remaining time 43135.67313671112 seconds...

Set parameter TimeLimit to value 4.3135673136711121e+04

Optimization was successful.

MSE: 3.4141888946456436

Completed fold 2 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=15 with remaining time 42977.1494038105 seconds...

Set parameter TimeLimit to value 42977.1494038105

Optimization was successful.

MSE: 2.5071669320249144

Completed fold 3 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=15 with remaining time 42962.58872079849 seconds...

Set parameter TimeLimit to value 4.2962588720798492e+04

Optimization was successful.

MSE: 2.8537009093594685

Completed fold 4 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=15 with remaining time 42946.41210794449 seconds...

Set parameter TimeLimit to value 4.2946412107944489e+04

Optimization was successful.

MSE: 3.1921713193955923

Completed fold 5 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=15 with remaining time 42937.89082789421 seconds...

Set parameter TimeLimit to value 4.2937890827894211e+04

Optimization was successful.

MSE: 2.4989929341292325

Completed fold 6 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=15 with remaining time 42811.45481085777 seconds...

Set parameter TimeLimit to value 4.2811454810857773e+04

Optimization was successful.

MSE: 3.1328070797945577

Completed fold 7 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=15 with remaining time 42805.118061065674 seconds...

Set parameter TimeLimit to value 4.2805118061065674e+04

Optimization was successful.

MSE: 2.5788472766404937

Completed fold 8 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=15 with remaining time 42783.511541843414 seconds...

Set parameter TimeLimit to value 4.2783511541843414e+04

Optimization was successful.

MSE: 3.6426526797015457

Completed fold 9 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=15 with remaining time 42760.80212187767 seconds...

Set parameter TimeLimit to value 4.2760802121877670e+04

Optimization was successful.

MSE: 3.193134299031582

Completed fold 10 for k=15. Results saved to cross\_validation\_results.csv

Processing fold 1 for k=20 with remaining time 42752.078968048096 seconds...

Set parameter TimeLimit to value 4.2752078968048096e+04

Optimization was successful.

MSE: 3.614072978918192

Completed fold 1 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=20 with remaining time 42661.53250670433 seconds...

Set parameter TimeLimit to value 4.2661532506704330e+04

Optimization was successful.

MSE: 3.7752952703551914

Completed fold 2 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=20 with remaining time 42258.35064172745 seconds...

Set parameter TimeLimit to value 4.2258350641727448e+04

Optimization was successful.

MSE: 3.3942107381964566

Completed fold 3 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=20 with remaining time 42238.88775396347 seconds...

Set parameter TimeLimit to value 4.2238887753963470e+04

Optimization was successful.

MSE: 2.9841793728503365

Completed fold 4 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=20 with remaining time 42223.79138302803 seconds...

Set parameter TimeLimit to value 4.2223791383028030e+04

Optimization was successful.

MSE: 3.173932165760675

Completed fold 5 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=20 with remaining time 42216.794296979904 seconds...

Set parameter TimeLimit to value 4.2216794296979904e+04

Optimization was successful.

MSE: 2.7337443915790858

Completed fold 6 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=20 with remaining time 42123.18860578537 seconds...

Set parameter TimeLimit to value 4.2123188605785370e+04

Optimization was successful.

MSE: 2.972461708763836

Completed fold 7 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=20 with remaining time 42113.406563043594 seconds...

Set parameter TimeLimit to value 4.2113406563043594e+04

Optimization was successful.

MSE: 2.425050151056745

Completed fold 8 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=20 with remaining time 42077.4822678566 seconds...

Set parameter TimeLimit to value 42077.4822678566

Optimization was successful.

MSE: 3.9630766109005164

Completed fold 9 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=20 with remaining time 42053.48014783859 seconds...

Set parameter TimeLimit to value 4.2053480147838593e+04

Optimization was successful.

MSE: 3.4915968756118367

Completed fold 10 for k=20. Results saved to cross\_validation\_results.csv

Processing fold 1 for k=25 with remaining time 42042.55409669876 seconds...

Set parameter TimeLimit to value 4.2042554096698761e+04

Optimization was successful.

MSE: 3.7119599991319543

Completed fold 1 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=25 with remaining time 42030.572757959366 seconds...

Set parameter TimeLimit to value 4.2030572757959366e+04

Optimization was successful.

MSE: 4.135798358857802

Completed fold 2 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=25 with remaining time 41971.67784905434 seconds...

Set parameter TimeLimit to value 4.1971677849054337e+04

Optimization was successful.

MSE: 3.326874457360343

Completed fold 3 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=25 with remaining time 41953.86353302002 seconds...

Set parameter TimeLimit to value 4.1953863533020020e+04

Optimization was successful.

MSE: 2.941608160787275

Completed fold 4 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=25 with remaining time 41944.61615085602 seconds...

Set parameter TimeLimit to value 4.1944616150856018e+04

Optimization was successful.

MSE: 3.4937455362389835

Completed fold 5 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=25 with remaining time 41941.26429891586 seconds...

Set parameter TimeLimit to value 4.1941264298915863e+04

Optimization was successful.

MSE: 2.9615832211996693

Completed fold 6 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=25 with remaining time 41910.56849503517 seconds...

Set parameter TimeLimit to value 4.1910568495035172e+04

Optimization was successful.

MSE: 2.767581044831384

Completed fold 7 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=25 with remaining time 41907.429294109344 seconds...

Set parameter TimeLimit to value 4.1907429294109344e+04

Optimization was successful.

MSE: 2.3554571319615425

Completed fold 8 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=25 with remaining time 41901.16346287727 seconds...

Set parameter TimeLimit to value 4.1901163462877274e+04

Optimization was successful.

MSE: 3.818251372371366

Completed fold 9 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=25 with remaining time 41896.701279878616 seconds...

Set parameter TimeLimit to value 4.1896701279878616e+04

Optimization was successful.

MSE: 3.1764114853384733

Completed fold 10 for k=25. Results saved to cross\_validation\_results.csv

Processing fold 1 for k=30 with remaining time 41885.20294690132 seconds...

Set parameter TimeLimit to value 4.1885202946901321e+04

Optimization was successful.

MSE: 3.8929547506089244

Completed fold 1 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=30 with remaining time 41884.271481752396 seconds...

Set parameter TimeLimit to value 4.1884271481752396e+04

Optimization was successful.

MSE: 4.1302228997504695

Completed fold 2 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=30 with remaining time 41875.89033079147 seconds...

Set parameter TimeLimit to value 4.1875890330791473e+04

Optimization was successful.

MSE: 2.9003979007390983

Completed fold 3 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=30 with remaining time 41874.74344277382 seconds...

Set parameter TimeLimit to value 4.1874743442773819e+04

Optimization was successful.

MSE: 3.08368565831253

Completed fold 4 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=30 with remaining time 41871.342206954956 seconds...

Set parameter TimeLimit to value 4.1871342206954956e+04

Optimization was successful.

MSE: 3.3758581898178392

Completed fold 5 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=30 with remaining time 41869.25074386597 seconds...

Set parameter TimeLimit to value 4.1869250743865967e+04

Optimization was successful.

MSE: 2.943666104950675

Completed fold 6 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=30 with remaining time 41866.39153289795 seconds...

Set parameter TimeLimit to value 4.1866391532897949e+04

Optimization was successful.

MSE: 2.8610040405926718

Completed fold 7 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=30 with remaining time 41865.49041104317 seconds...

Set parameter TimeLimit to value 4.1865490411043167e+04

Optimization was successful.

MSE: 2.2367980014683524

Completed fold 8 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=30 with remaining time 41862.009301900864 seconds...

Set parameter TimeLimit to value 4.1862009301900864e+04

Optimization was successful.

MSE: 4.3215730858363335

Completed fold 9 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=30 with remaining time 41859.67676782608 seconds...

Set parameter TimeLimit to value 4.1859676767826080e+04

Optimization was successful.

MSE: 3.216171054510333

Completed fold 10 for k=30. Results saved to cross\_validation\_results.csv

Processing fold 1 for k=35 with remaining time 41857.588013887405 seconds...

Set parameter TimeLimit to value 4.1857588013887405e+04

Optimization was successful.

MSE: 3.8797079783321133

Completed fold 1 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=35 with remaining time 41857.3373799324 seconds...

Set parameter TimeLimit to value 41857.3373799324

Optimization was successful.

MSE: 3.7218821184940696

Completed fold 2 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=35 with remaining time 41855.97883486748 seconds...

Set parameter TimeLimit to value 4.1855978834867477e+04

Optimization was successful.

MSE: 3.1406027160908185

Completed fold 3 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=35 with remaining time 41855.66286087036 seconds...

Set parameter TimeLimit to value 4.1855662860870361e+04

Optimization was successful.

MSE: 3.2953573040291193

Completed fold 4 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=35 with remaining time 41854.472468853 seconds...

Set parameter TimeLimit to value 41854.472468853

Optimization was successful.

MSE: 3.65954543025446

Completed fold 5 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=35 with remaining time 41853.965609788895 seconds...

Set parameter TimeLimit to value 4.1853965609788895e+04

Optimization was successful.

MSE: 2.942755826629264

Completed fold 6 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=35 with remaining time 41851.636930942535 seconds...

Set parameter TimeLimit to value 4.1851636930942535e+04

Optimization was successful.

MSE: 2.918664091043122

Completed fold 7 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=35 with remaining time 41851.22527575493 seconds...

Set parameter TimeLimit to value 4.1851225275754929e+04

Optimization was successful.

MSE: 2.2711794157238856

Completed fold 8 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=35 with remaining time 41849.46880483627 seconds...

Set parameter TimeLimit to value 4.1849468804836273e+04

Optimization was successful.

MSE: 4.327743054737564

Completed fold 9 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=35 with remaining time 41849.104007959366 seconds...

Set parameter TimeLimit to value 4.1849104007959366e+04

Optimization was successful.

MSE: 3.4166518536671036

Completed fold 10 for k=35. Results saved to cross\_validation\_results.csv

Processing fold 1 for k=40 with remaining time 41848.828422784805 seconds...

Set parameter TimeLimit to value 4.1848828422784805e+04

Optimization was successful.

MSE: 4.031002133864027

Completed fold 1 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=40 with remaining time 41848.65303397179 seconds...

Set parameter TimeLimit to value 4.1848653033971786e+04

Optimization was successful.

MSE: 3.8766342769321267

Completed fold 2 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=40 with remaining time 41848.33926296234 seconds...

Set parameter TimeLimit to value 4.1848339262962341e+04

Optimization was successful.

MSE: 2.983531032004351

Completed fold 3 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=40 with remaining time 41848.118137836456 seconds...

Set parameter TimeLimit to value 4.1848118137836456e+04

Optimization was successful.

MSE: 3.260083613934959

Completed fold 4 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=40 with remaining time 41847.82401895523 seconds...

Set parameter TimeLimit to value 4.1847824018955231e+04

Optimization was successful.

MSE: 3.646435537635532

Completed fold 5 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=40 with remaining time 41847.588471889496 seconds...

Set parameter TimeLimit to value 4.1847588471889496e+04

Optimization was successful.

MSE: 2.650143097604909

Completed fold 6 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=40 with remaining time 41847.333746910095 seconds...

Set parameter TimeLimit to value 4.1847333746910095e+04

Optimization was successful.

MSE: 3.172607643921931

Completed fold 7 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=40 with remaining time 41847.12951397896 seconds...

Set parameter TimeLimit to value 4.1847129513978958e+04

Optimization was successful.

MSE: 2.4419702510796553

Completed fold 8 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=40 with remaining time 41846.77714776993 seconds...

Set parameter TimeLimit to value 4.1846777147769928e+04

Optimization was successful.

MSE: 3.988627303969263

Completed fold 9 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=40 with remaining time 41846.54183077812 seconds...

Set parameter TimeLimit to value 4.1846541830778122e+04

Optimization was successful.

MSE: 3.517941998550394

Completed fold 10 for k=40. Results saved to cross\_validation\_results.csv

Processing fold 1 for k=45 with remaining time 41846.36831688881 seconds...

Set parameter TimeLimit to value 4.1846368316888809e+04

Optimization was successful.

MSE: 4.031002133864027

Completed fold 1 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=45 with remaining time 41846.202182769775 seconds...

Set parameter TimeLimit to value 4.1846202182769775e+04

Optimization was successful.

MSE: 4.031036147600557

Completed fold 2 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=45 with remaining time 41846.029357910156 seconds...

Set parameter TimeLimit to value 4.1846029357910156e+04

Optimization was successful.

MSE: 3.2455087097158057

Completed fold 3 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=45 with remaining time 41845.835705041885 seconds...

Set parameter TimeLimit to value 4.1845835705041885e+04

Optimization was successful.

MSE: 3.206672806390462

Completed fold 4 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=45 with remaining time 41845.671874046326 seconds...

Set parameter TimeLimit to value 4.1845671874046326e+04

Optimization was successful.

MSE: 3.5368085948635013

Completed fold 5 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=45 with remaining time 41845.505183935165 seconds...

Set parameter TimeLimit to value 4.1845505183935165e+04

Optimization was successful.

MSE: 2.885551699108614

Completed fold 6 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=45 with remaining time 41845.33733987808 seconds...

Set parameter TimeLimit to value 4.1845337339878082e+04

Optimization was successful.

MSE: 2.8468828927119123

Completed fold 7 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=45 with remaining time 41845.17719388008 seconds...

Set parameter TimeLimit to value 4.1845177193880081e+04

Optimization was successful.

MSE: 2.676785271657453

Completed fold 8 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=45 with remaining time 41844.97952103615 seconds...

Set parameter TimeLimit to value 4.1844979521036148e+04

Optimization was successful.

MSE: 3.9082508755832865

Completed fold 9 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=45 with remaining time 41844.806701898575 seconds...

Set parameter TimeLimit to value 4.1844806701898575e+04

Optimization was successful.

MSE: 3.517941998550394

Completed fold 10 for k=45. Results saved to cross\_validation\_results.csv

Processing fold 1 for k=50 with remaining time 41844.648111104965 seconds...

Set parameter TimeLimit to value 4.1844648111104965e+04

Optimization was successful.

MSE: 3.902812296594459

Completed fold 1 for k=50. Results saved to cross\_validation\_results.csv

Processing fold 2 for k=50 with remaining time 41844.48473787308 seconds...

Set parameter TimeLimit to value 4.1844484737873077e+04

Optimization was successful.

MSE: 4.045508627464424

Completed fold 2 for k=50. Results saved to cross\_validation\_results.csv

Processing fold 3 for k=50 with remaining time 41844.318063020706 seconds...

Set parameter TimeLimit to value 4.1844318063020706e+04

Optimization was successful.

MSE: 3.1986246021676696

Completed fold 3 for k=50. Results saved to cross\_validation\_results.csv

Processing fold 4 for k=50 with remaining time 41844.14331078529 seconds...

Set parameter TimeLimit to value 4.1844143310785294e+04

Optimization was successful.

MSE: 3.133484223441435

Completed fold 4 for k=50. Results saved to cross\_validation\_results.csv

Processing fold 5 for k=50 with remaining time 41843.97967791557 seconds...

Set parameter TimeLimit to value 4.1843979677915573e+04

Optimization was successful.

MSE: 3.4772917948348105

Completed fold 5 for k=50. Results saved to cross\_validation\_results.csv

Processing fold 6 for k=50 with remaining time 41843.80910778046 seconds...

Set parameter TimeLimit to value 4.1843809107780457e+04

Optimization was successful.

MSE: 2.6945486464962927

Completed fold 6 for k=50. Results saved to cross\_validation\_results.csv

Processing fold 7 for k=50 with remaining time 41843.64865875244 seconds...

Set parameter TimeLimit to value 4.1843648658752441e+04

Optimization was successful.

MSE: 3.002748436731109

Completed fold 7 for k=50. Results saved to cross\_validation\_results.csv

Processing fold 8 for k=50 with remaining time 41843.48007488251 seconds...

Set parameter TimeLimit to value 4.1843480074882507e+04

Optimization was successful.

MSE: 2.5422032476993133

Completed fold 8 for k=50. Results saved to cross\_validation\_results.csv

Processing fold 9 for k=50 with remaining time 41843.31406879425 seconds...

Set parameter TimeLimit to value 4.1843314068794250e+04

Optimization was successful.

MSE: 4.163277463457141

Completed fold 9 for k=50. Results saved to cross\_validation\_results.csv

Processing fold 10 for k=50 with remaining time 41843.1430079937 seconds...

Set parameter TimeLimit to value 41843.1430079937

Optimization was successful.

MSE: 3.457661282817287

Completed fold 10 for k=50. Results saved to cross\_validation\_results.csv

The best k is 10 with an average MSE of 2.8239500709179213 across all folds.

**6.2 MIQP: Mean MSE values for each k**

| **k** | **Average MSE** |
| --- | --- |
| **5** | **3.665682** |
| **10** | **2.823950** |
| **15** | **3.043842** |
| **20** | **3.252762** |
| **25** | **3.268927** |
| **30** | **3.296233** |
| **35** | **3.357409** |
| **40** | **3.356898** |
| **45** | **3.388644** |
| **50** | **3.361816** |

**6.3 MIQP: Mean Computation Time for each k**

| **k** | **Computation Time** |
| --- | --- |
| **5** | **0.46932220458984375** |
| **10** | **2.7973876714706423** |
| **15** | **41.51614489555359** |
| **20** | **70.94480872154236** |
| **25** | **15.724645566940307** |
| **30** | **2.758376383781433** |
| **35** | **0.8732364654541016** |
| **40** | **0.24390151500701904** |
| **45** | **0.16978588104248046** |
| **50** | **0.1647702693939209** |