DATA MİNİNG HOMEWORK REPORT

Fahrettin Solak
201401053

08.12.2024

Contents

Question 1 Report: Air Quality Prediction and Model Comparison 2
1. Introduction 2
2. Data Set and Selection 2
3. Data Pre-Processing 8
4. Model Installation and Training Stages 10
5. Model Performance Evaluation and Comparison 14
6. Hyperparameter Settings and Results 15
7. Results and Learnings 16
8. Conclusion: General Evaluation 17
Question 2 Report: Clustering Analysis of Weather Data with K-means Algorithm 18
1. Introduction and Data Set 18
2. Data Pre-Processing 19
3. Model Training and Evaluation 20
4. Hyperparameter Adjustment and Optimization 21
5. Evaluation of Clustering Quality 22
6. Analysis of Cluster Properties 22
7. Comparison Before and After the Hyperparameter Setting 22
8. Results and Learnings 25
Question 3 Report: Analysis by Linear Regression and Alternative Methods for the Estimation of Continuous Target Variable 26
1. Introduction 26
2. Data Set and Characteristics 27
3. Data Pre-Processing 29
4. Model Training and Evaluation 31
5. Model Performance Comparison 34
6. Results and Learnings 35

Question 1 Report: Air Quality Prediction and Model Comparison

---

- **The Subject of the Problem**: Air Quality Prediction with Decision Tree and Comparison with Different Models
- **The Data Set and Link Used**:
    - **Data Set Source**: Kaggle
    - **Link**: Air Quality and Pollution Assessment
- **Methods Used**:
    - Decision Tree,
    - Random Forest,
    - Support Vector Machine (SVM),
    - Logistic Regression,
    - Data Preprocessing,
    - Hyperparameter Adjustment,
    - Ensemble Learning (Voting Classifier)

---

1. Introduction

Project Objective: The aim of this project is to classify air quality using various environmental factors. Different models of machine learning have been applied and compared to improve the accuracy of the classification process. Predicting air quality plays an important role in protecting environmental health, reducing pollution and improving people's quality of life. In this study, a decision tree model for air quality classification was first used. Its performance was then compared with other popular classifiers such as Random Forest, Logistic Regression and Support Vector Machine. In addition, a community learning method was used by creating a voting classifier model that combines the strengths of different models to achieve better results

Data Mining and its Importance: Data Mining extracts valuable information from large data sets by identifying patterns, relationships, and trends. It helps to transform complex data into meaningful information and allows for accurate forecasts and data-based decisions. In environmental research, data mining plays a crucial role in analyzing environmental factors to address critical issues such as air pollution. In this project, data mining techniques were applied to study air quality. By studying various environmental factors, we have studied how these elements affect air quality and how they can be classified to support better environmental management and pollution control.

---

2. Data Set and Selection

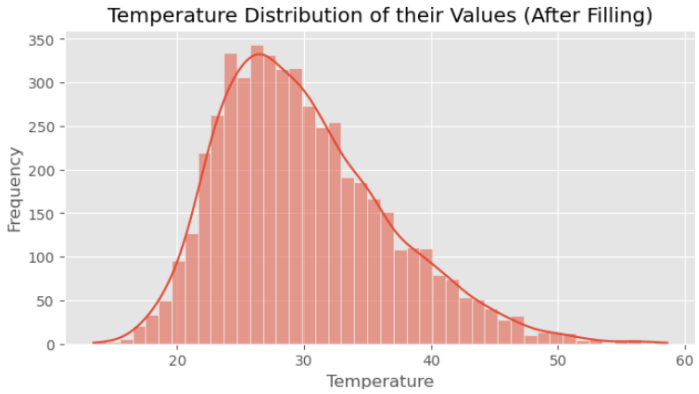
The "Air Quality and Pollution Assessment" dataset published on the Kaggle platform was used in the project. This data set is designed to assess air quality and pollution level based on various environmental parameters. The dataset has 5000 samples and 10 features and contains important parameters related to air pollution. The first 5 lines of the dataset are as follows:

The First 5 Lines of the Data Set:

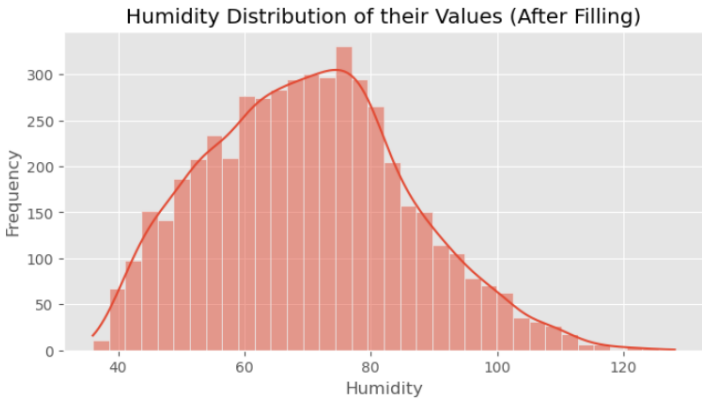
	Temperature	Humidity	PM2.5	PM10	NO2	SO2	CO	Proximity_to_Industrial_Areas	Population_Density	Air Quality
0	29.8	59.1	5.2	17.9	18.9	9.2	1.72	6.3	319	Moderate
1	28.3	75.6	2.3	12.2	30.8	9.7	1.64	6.0	611	Moderate
2	23.1	74.7	26.7	33.8	24.4	12.6	1.63	5.2	619	Moderate
3	27.1	39.1	6.1	6.3	13.5	5.3	1.15	11.1	551	Good
4	26.5	70.7	6.9	16.0	21.9	5.6	1.01	12.7	303	Good

Features Found in the Data Set:

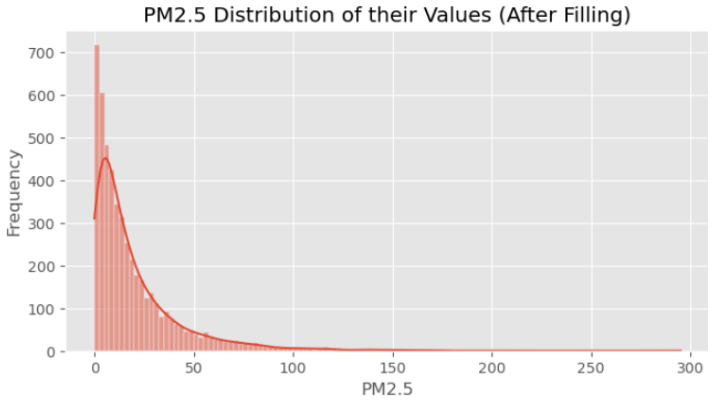
1. **Temperature**: The value of the temperature in the air (in Celsius).



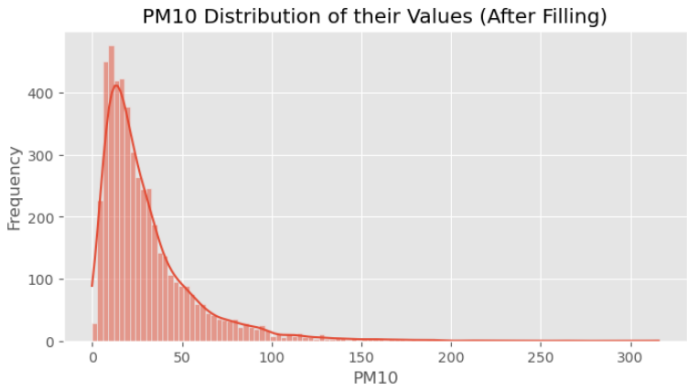
2. **Humidity**: The humidity in the air (%).



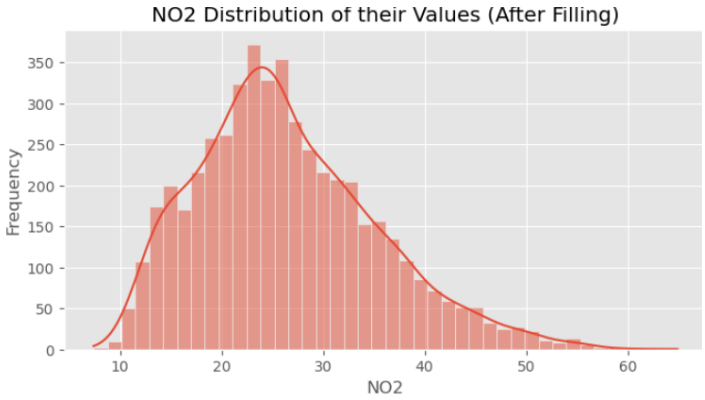
3. **PM2.5**: The amount of particulate matter smaller than 2.5 micrometers.



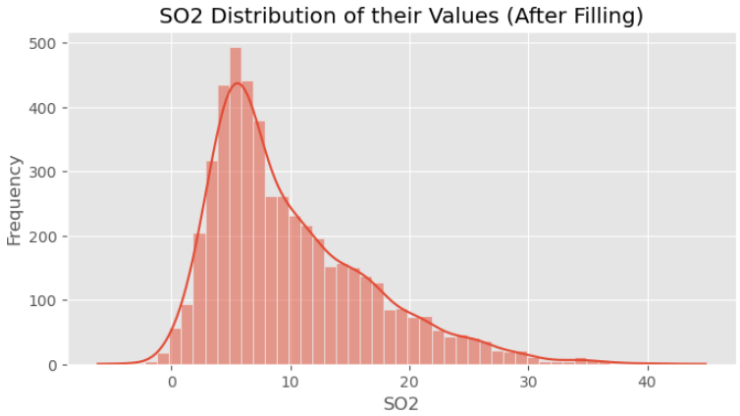
4. **PM10**: The amount of particulate matter smaller than 10 micrometers.



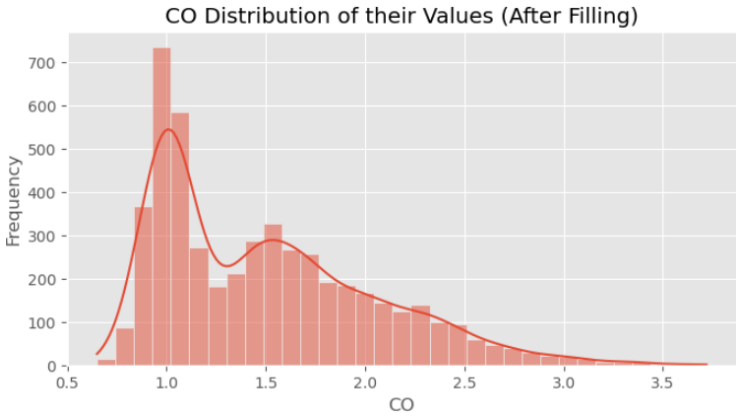
5. **NO2 (Nitrogen Dioxide)**: The concentration of nitrogen dioxide gas in the air.



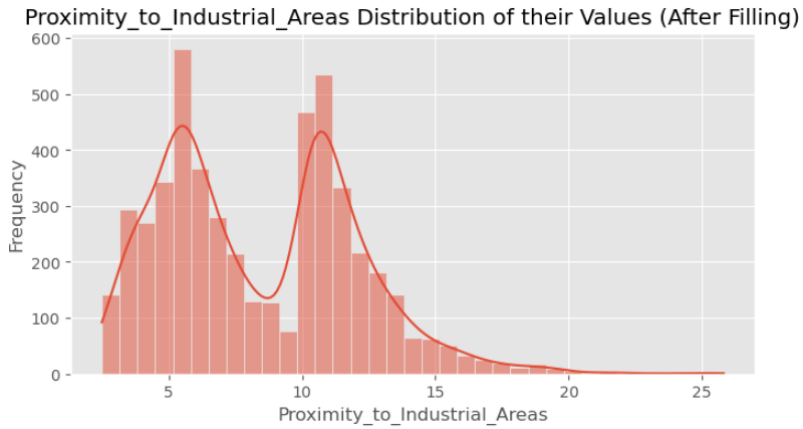
6. **SO2 (Sulfur Dioxide)**: The concentration of sulfur dioxide gas in the air.



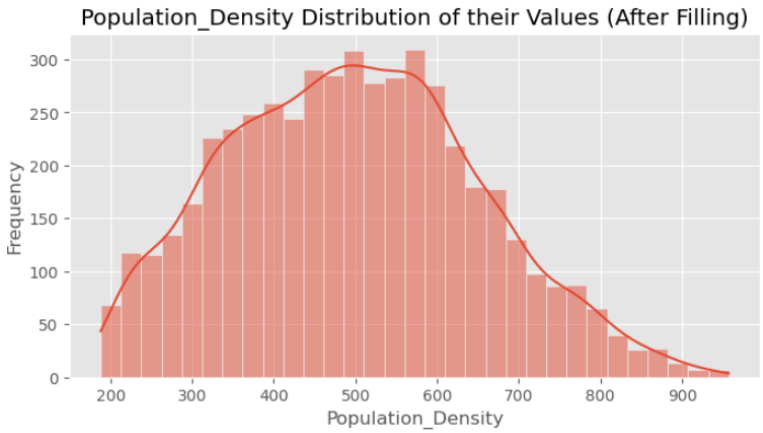
7. **CO (Carbon Monoxide)**: The concentration of carbon monoxide gas in the air.



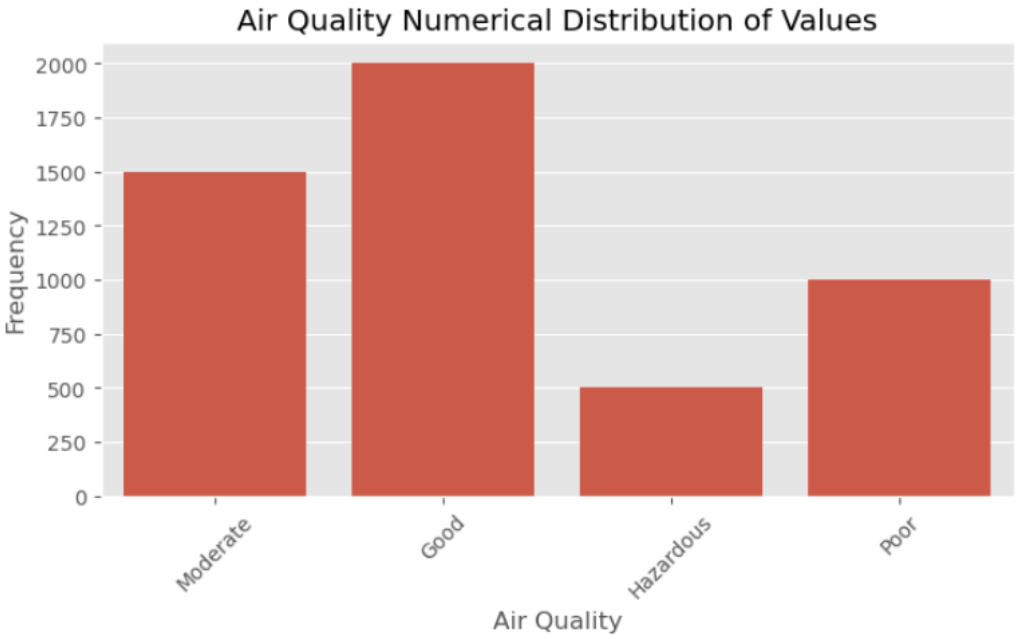
8. **Proximity_to_Industrial_Areas:** The distance to the industrial areas of the region (in km).



9. **Population_Density**: The density of people in the area.



10. **Air Quality**: The target variable indicating air quality. This variable is divided into four classes: "Good", "Moderate", "Poor" and "Hazardous".



General Review of the Data Set:

- **Missing Value Analysis:** It was checked whether there were missing values in the data set, and it was found that there were no missing values. This means that the model can be trained without the need for additional steps such as data cleaning or filling in missing data. Oct.

```
The Number of Missing Values:
Temperature                    0
Humidity                       0
PM2.5                          0
PM10                           0
NO2                            0
SO2                            0
CO                             0
Proximity_to_Industrial_Areas  0
Population_Density             0
Air Quality                    0
dtype: int64
```

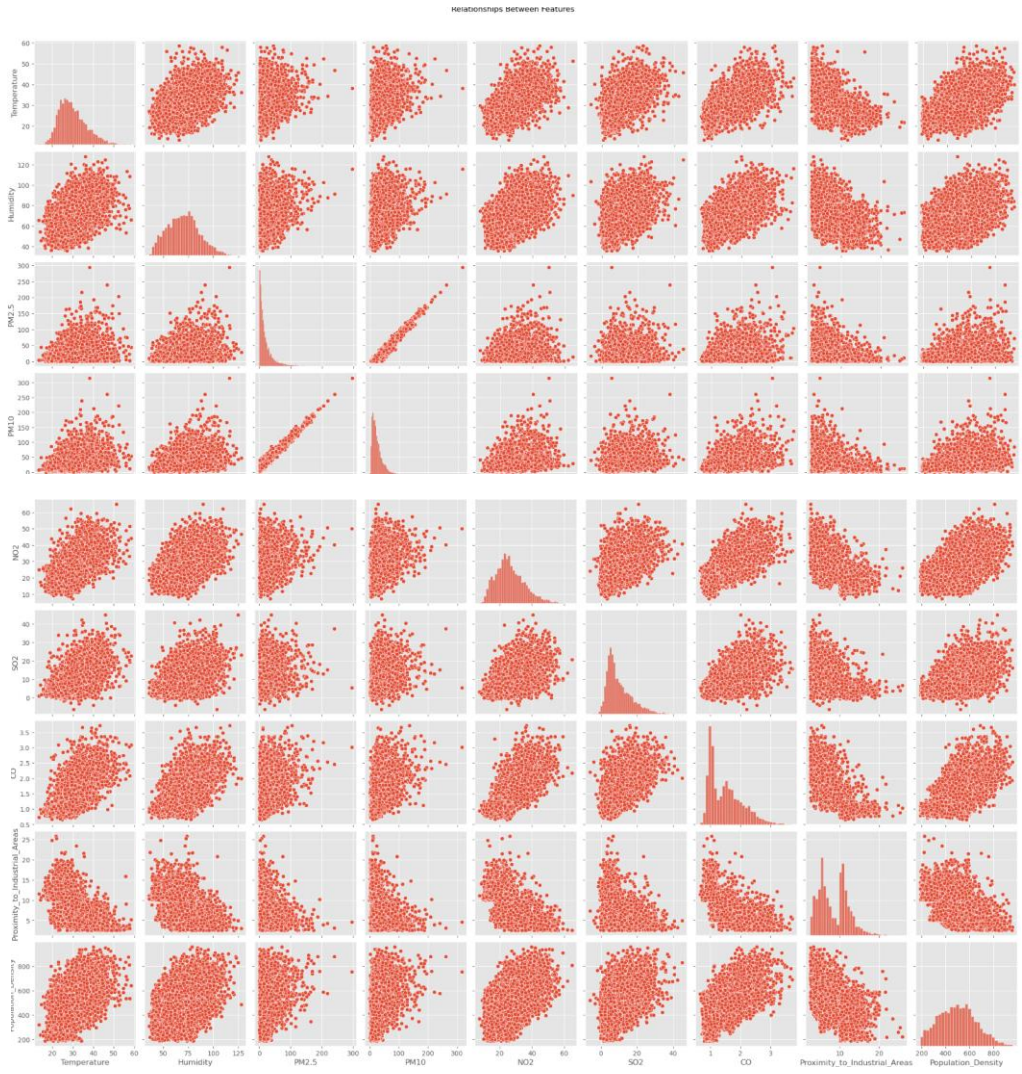
- **Statistical Information:** Statistical information such as minimum, maximum, average and standard deviation have been extracted for each feature in the data set. This information helped us to understand the overall structure of the data and revealed that some variables may have excessive values. It means that it can be trained.

Statistical Summary of the Data Set:

	Temperature	Humidity	PM2.5	PM10	NO2	SO2	CO	Proximity_to_Industrial_Areas	Population_Density
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	30.029020	70.056120	20.142140	30.218360	26.412100	10.014820	1.500354	8.425400	497.423800
std	6.720661	15.863577	24.554546	27.349199	8.895356	6.750303	0.546027	3.610944	152.754084
min	13.400000	36.000000	0.000000	-0.200000	7.400000	-6.200000	0.650000	2.500000	188.000000
25%	25.100000	58.300000	4.600000	12.300000	20.100000	5.100000	1.030000	5.400000	381.000000
50%	29.000000	69.800000	12.000000	21.700000	25.300000	8.000000	1.410000	7.900000	494.000000
75%	34.000000	80.300000	26.100000	38.100000	31.900000	13.725000	1.840000	11.100000	600.000000
max	58.600000	128.100000	295.000000	315.800000	64.900000	44.900000	3.720000	25.800000	957.000000

Relationships Between Variables and Visualization:

- **Pair Plot Graph:** A pair plot graph was created in order to better understand the relationship of properties to each other. In particular, the relationships of variables such as PM2.5, NO2 and CO with other factors were observed thanks to this graph.



3. Data Pre-Processing

Data preprocessing is a very important stage in the machine learning process and directly affects model performance. During data preprocessing in this project, the following steps were applied:

3.1 Filling in the Missing Values:

- It was checked whether there are missing values in the data set. The absence of missing values provided a great advantage for the model, because at this stage there was no need to fill in missing values in order to prevent any data loss or misleading data formation.

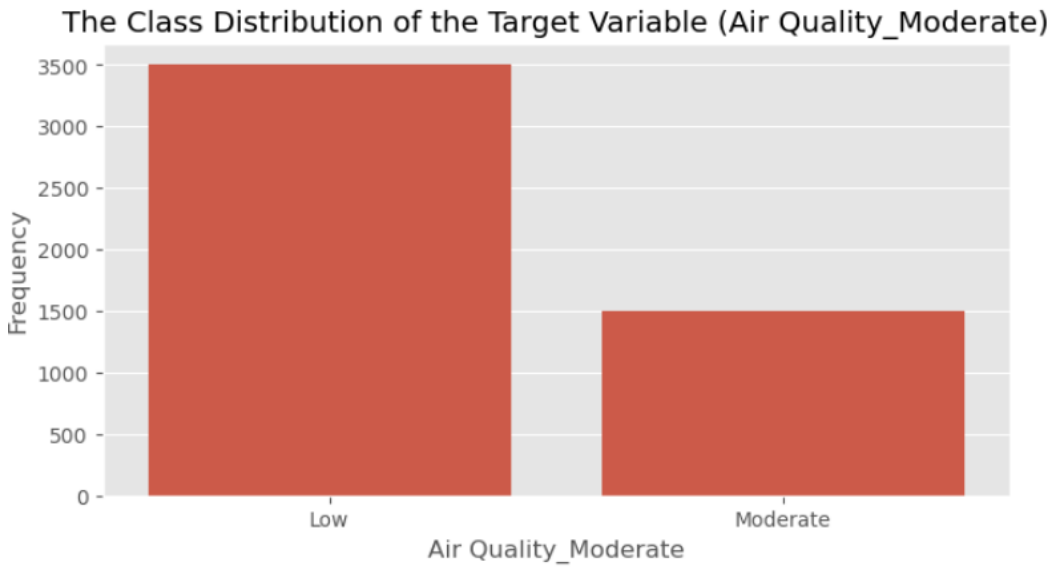
```
The Number of Missing Values:
Temperature                    0
Humidity                       0
PM2.5                          0
PM10                           0
NO2                            0
SO2                            0
CO                             0
Proximity_to_Industrial_Areas  0
Population_Density             0
Air Quality                    0
dtype: int64
```

3.2 Digitization of Categorical Variables (One-Hot Encoding):

- Categorical variables such as **Air Quality** had to be numericized in order to be processed by machine learning algorithms. For this reason, using One-Hot Encoding, each category was expressed with a value of 0 or 1.

The First 5 Lines of the Data Set (After One-Hot Encoding):

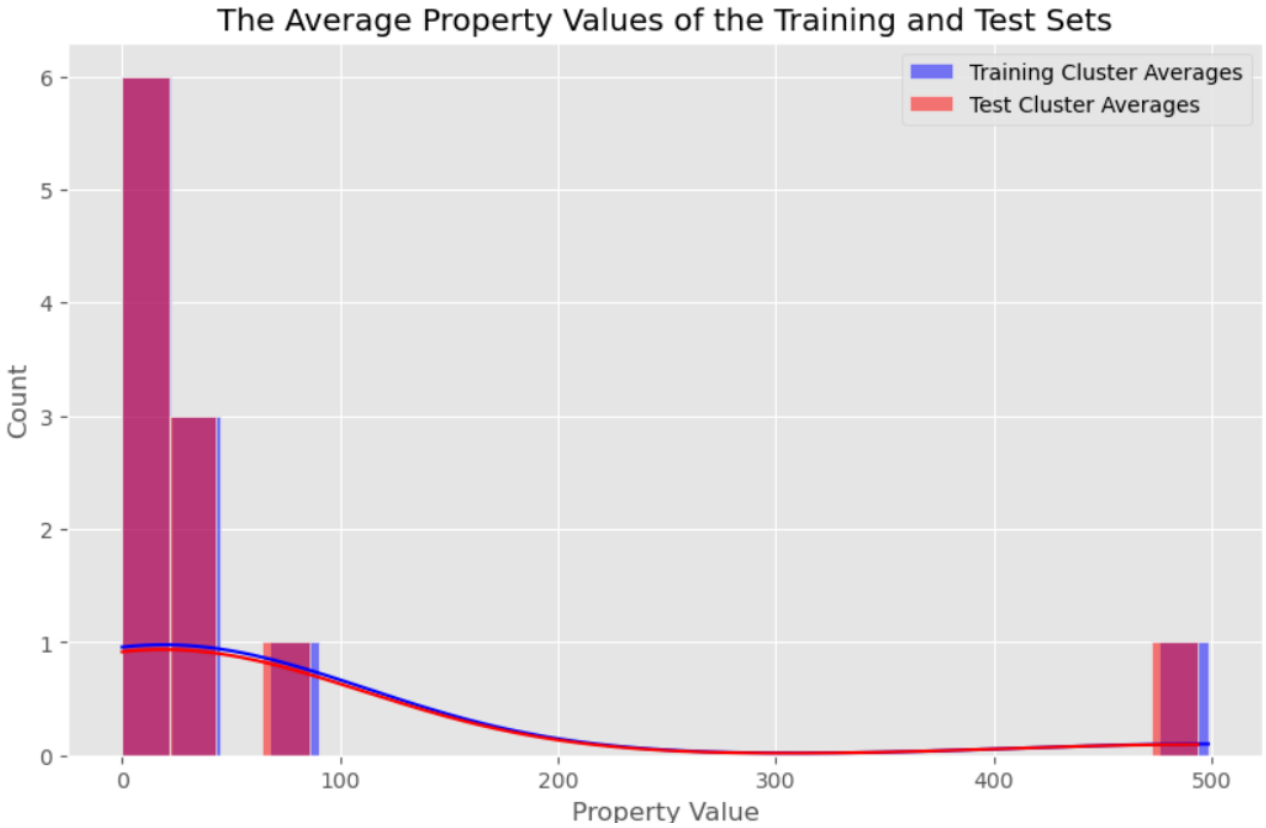
	Temperature	Humidity	PM2.5	PM10	NO2	SO2	CO	Proximity_to_Industrial_Areas	Population_Density	Air Quality_Hazardous	Air Quality_Moderate	Air Quality_Poor
0	29.8	59.1	5.2	17.9	18.9	9.2	1.72	6.3	319	False	True	False
1	28.3	75.6	2.3	12.2	30.8	9.7	1.64	6.0	611	False	True	False
2	23.1	74.7	26.7	33.8	24.4	12.6	1.63	5.2	619	False	True	False
3	27.1	39.1	6.1	6.3	13.5	5.3	1.15	11.1	551	False	False	False
4	26.5	70.7	6.9	16.0	21.9	5.6	1.01	12.7	303	False	False	False



3.3 Dividing the Data Set into Training and Testing:

- The data was divided into 80% training and 20% testing. This ratio allowed us to test the overall performance of the model and its accuracy on the new data. While the training set was used for the model to learn the data, the test set was used to measure the performance of the model.

```
Training set size: (4000, 11)
Test set size: (1000, 11)
```

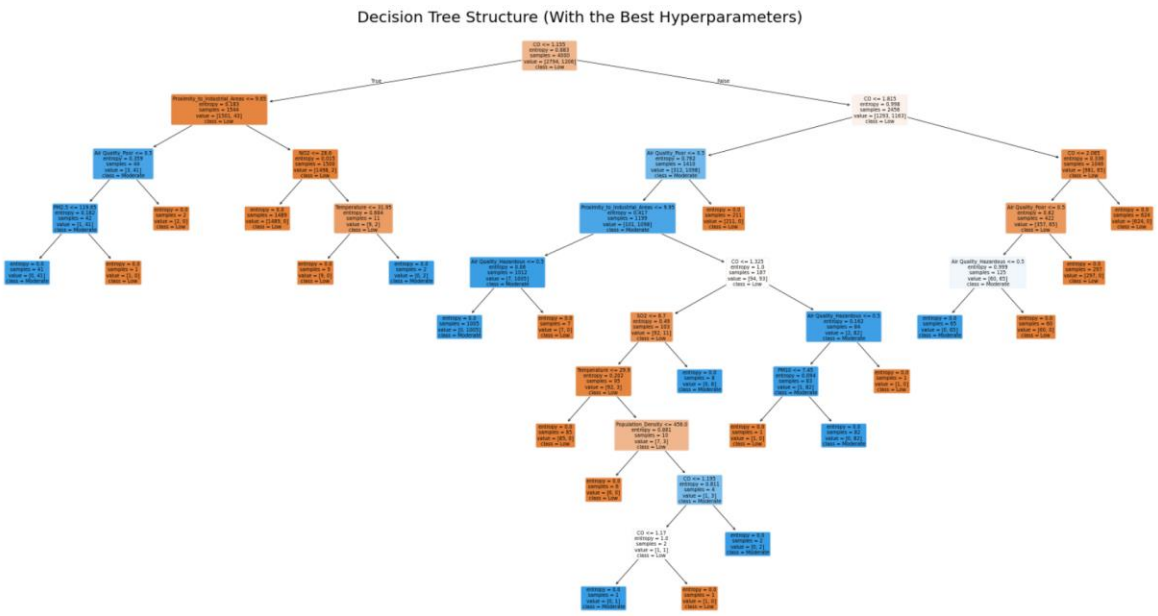


---

4. Model Installation and Training Stages

4.1 The Decision Tree Model:

- **Model Setup**: The Decision Tree is a highly explainable model that classifies data by dividing it into branches. The Decision Tree continues the prediction process by making a division according to a certain property at each node.



- **Hyperparameter Setting**: Using GridSearchCV, parameters such as **criterion**, **max_depth**, **min_samples_split** and **min_samples_leaf** are set to find the most appropriate values.
    - **The Best Parameters**: Criterion = 'entropy', Max Depth = None, Min Samples Split = 2, Min Samples Leaf = 1.

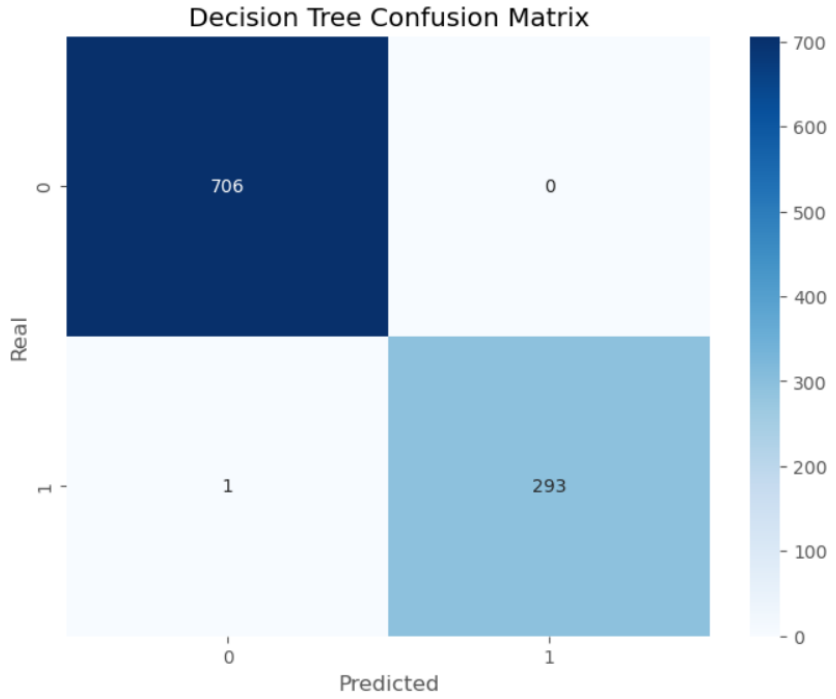
```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

- **Evaluation of Model Performance**: The performance of the Decision Tree model was evaluated using metrics such as accuracy, precision, recall, F1 score.

```
Decision Tree Model Performance Metrics:
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
```

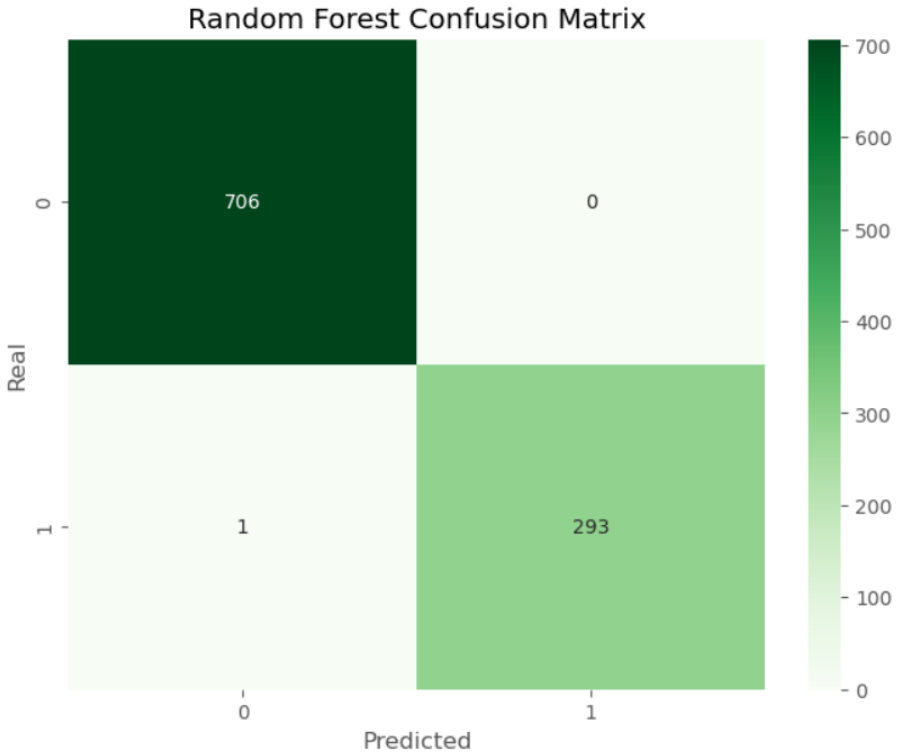
The best values:

```
The Best Decision Tree Hyperparameters Are: {'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}
```



4.2 Random Forest Model:

- **Model Setup**: Random Forest combines many Decision Trees to create a model that is more balanced and has a high generalization ability. Dec. In this way, it increases the overall accuracy by compensating for the weaknesses of different trees.



- **Hyperparameter Setting**: Using GridSearchCV, the optimal combination of parameters such as **n_estimators**, **max_depth**, **min_samples_split**, **min_samples_leaf** was determined.
    - **The Best Parameters**: n_estimators = 50, Max Depth = None, Min Samples Split = 10, Min Samples Leaf = 1.

```
param_grid_rf = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

- **Evaluation of Model Performance**: The Random Forest model has shown a more balanced and better generalization performance compared to the Decision Tree.

```
Random Forest Model Performance Metrics:
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
```

The best values:

```
The Best Random Forest Hyperparameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 50}
```

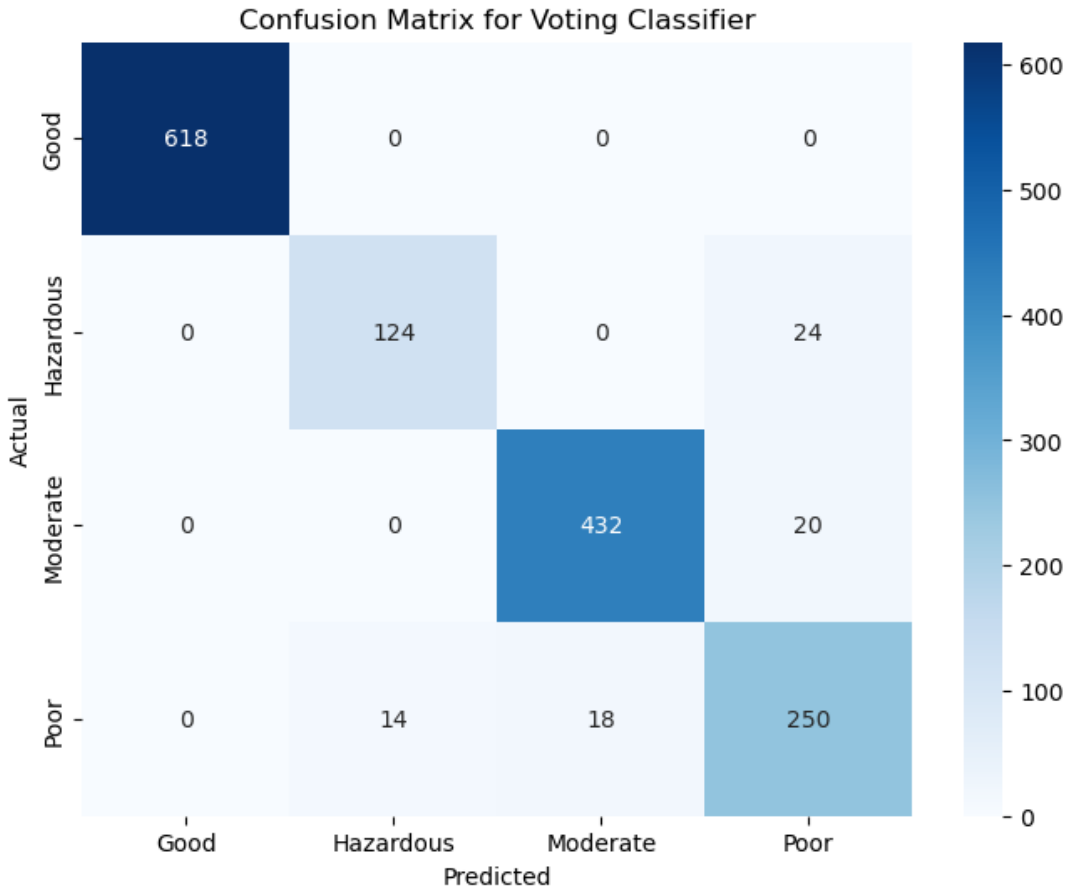
4.3 Other Classifiers:

- **Logistic Regression and SVM**: Both of these models were trained on the data set and their performance was evaluated. Logistic Regression was successful in determining linear boundaries between classes, while SVM was effective in learning more complex classroom boundaries.

4.4 Community Learning with Voting Classifier:

- **Ensemble Learning**: Voting Classifier has created a more balanced and powerful prediction model by combining Decisional Tree, Random Forest, Logistic Regression and SVM models. This method provides higher accuracy and balance by combining the strengths of the models.

```
Voting Classifier Performansı:
Accuracy: 0.95
Precision: 0.95
Recall: 0.95
F1 Score: 0.95
```



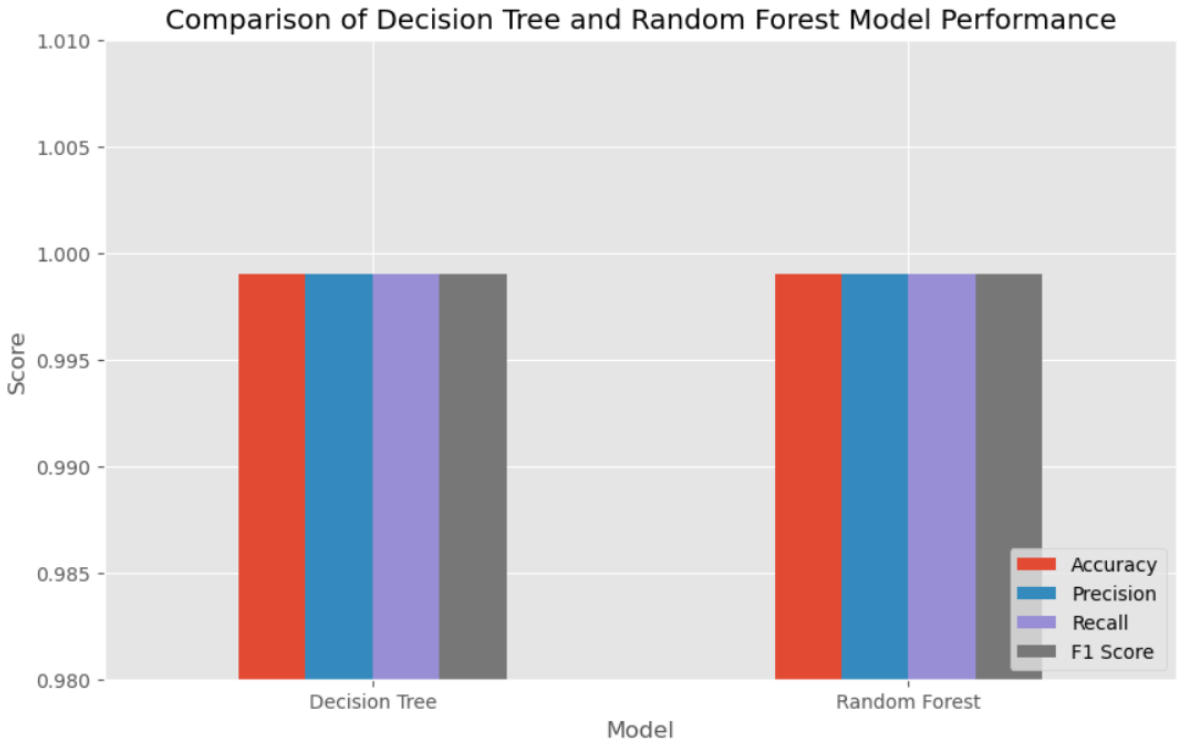
5. Model Performance Evaluation and Comparison

Model Comparison:

- We analyzed the performance of each model with metrics such as accuracy, accuracy, recall and F1 score. As a result of the comparison:
    - **Decision Tree**: It provided high accuracy, but excessive compliance problems were observed in some classes.
    - **Random Forest:** It gave results that were more balanced and had a high generalization ability.
    - **SVM and Logistic Regression**: It was more effective in determining linear boundaries.
    - **Voting Classifier**: It has achieved the highest accuracy by combining the strengths of all models.

Model Performance Comparison:

	Model	Accuracy	Precision	Recall	F1 Score
0	Decision Tree	0.999	0.999001	0.999	0.999
1	Random Forest	0.999	0.999001	0.999	0.999



6. Hyperparameter Settings and Results

Hyperparameter optimization is a critical step to ensure that machine learning models work in a more balanced and effective way. In this project, performance improvement was achieved by optimizing the hyperparameters of **Decision Tree** and **Random Forest** models in particular.

Stages of Hyperparameter Settings:

6.1 Determination of Hyperparameters for Decision Tree and Random Forest:

- Parameters such as **max_depth**, **min_samples_split**, **n_estimators** have been determined for **Decision Tree** and **Random Forest** models and the most appropriate combinations of these parameters have been determined using GridSearchCV.
- **The Best Parameters For the Decision Tree**: Criterion = 'entropy', Max Depth = None, Min Samples Split = 2, Min Samples Leaf = 1.

- **The Best Parameters For Random Forest**: n_estimators = 50, Max Depth = None, Min Samples Split = 10, Min Samples Leaf = 1.

6.2 Finding the Best Hyperparameters Using RandomizedSearchCV:

- **RandomizedSearchCV**, provided a faster optimization compared to GridSearchCV, as it tried a certain number of random hyperparameter combinations. In this way, the best results were found quickly without trying too many parameter combinations.

6.3 Performance Improvements and Comparisons:

- After the hyperparameter optimization, significant increases were observed in metrics such as accuracy, precision, recall and F1 score of each model.
    - Especially as a result of the hyperparameter optimization performed in the **Random Forest** model, there was a significant improvement in recall and F1 score. This means that the model is able to recognize especially positive classes more accurately.
- As a result of performance comparison, it has been seen that optimization increases the generalization ability of the model and better results are obtained on the test data.

---

7. Results and Learnings

In this project, various classification operations were performed using different machine learning models in order to classify air quality. The training process, performance evaluation and hyperparameter optimization of each model were performed. Below are the important points that we have extracted from this project:

7.1 The Importance of Data Pre-Processing:

- It was seen once again during this project that the data preprocessing step directly affects the success of the model in data mining projects. Filling in the missing values, digitizing the categorical variables and correctly dividing the data as training/testing have greatly improved the performance of the model.

7.2 Strengths and Weaknesses of the Models:

- **Decision Tree** is a very useful model in data classification with its simple and explainable structure; however, it tends to over-adapt in complex data.
- **Random Forest** is a powerful model that increases the ability to generalize using multiple Decision Trees. In this way, he learned the complex relationships observed in the data set better.
- **SVM and Logistic Regression** are the models that are more successful in linear classifications. SVM has been especially effective in data sets with complex boundaries.
- Ensemble Learning performed using **Voting Classifier** has improved the overall performance by taking advantage of the strengths of the models. This has been especially effective in Decoupling the imbalance between classes and improving the accuracy of forecasts.

7.3 The Effect of Hyperparameter Adjustment on Performance:

- The performance of the models has been significantly improved with the hyperparameter settings. In particular, the increase in recall and F1 scores has shown that the optimized model reduces performance differences between classes and gives more balanced results.

7.4 The Learned:

- In this project, I tried to predict air quality levels using different machine learning models. I worked on data preprocessing steps such as filling in missing values, encoding categorical variables, and dividing the data into training and test sets. I used Decision Tree, Random Forest, Logistic Regression and SVM. The Random Forest and the Voting Classifier gave the best accuracy (95%). I also adjusted the hyperparameters to make the Random Forest model better, which helped me improve its performance and stability. This project taught me how different models work and the importance of adjusting hyper parameters to achieve better results. Hyperparameters to get better results.

---

8. Conclusion: General Evaluation

In this project, different machine learning algorithms, data preprocessing process and hyperparameter optimizations used to improve air quality prediction were analyzed in detail. Data mining played a critical role in this project and made an important contribution to the process of classifying and analyzing environmental factors.

In particular, the use of ensemble methods has increased the accuracy and generalization ability of air quality forecasting by taking advantage of the strengths of different models. Hyperparameter optimization has improved the performance of the models, allowing for more balanced and accurate forecasts.

The most important thing I learned during the project was how big a role data preprocessing, model optimization and hyperparameter adjustment play in the success of machine learning models. These processes directly affect not only the accuracy of the model, but also its generalization ability on the test data. These comparisons with different models and optimization steps clearly showed that we need to develop more reliable machine learning models for real-world applications.

Question 2 Report: Clustering Analysis of Weather Data with K-means Algorithm

---

- **The Subject of the Problem**: Grouping of Weather Data with K-means Clustering
- **The Data Set and Link Used**:
    - **Data Set Source**: Kaggle
    - **Link**: Weather Forecast Data on Kaggle

Methods Used:

- K-means Clustering
- Data Pre-Processing
- Hyperparameter Optimization
- Evaluation Metrics:
    - Silhouette Score
    - Davies-Bouldin Score
    - Calinski-Harabasz Score

---

1. Introduction and Data Set

Project Objective and Data Set Selection:

The aim of this project is to divide similar weather conditions into clusters based on weather-related data and to make inferences about these clusters. The K-means clustering algorithm has been used to divide this data into a certain number of clusters. Our goal is to make this data more meaningful by analyzing weather data and observing which clusters form under certain conditions.

Our data set is the "**Weather Forecast Data**" data set available on the **Kaggle** platform. This data set includes various parameters related to the weather and makes it possible to cluster these parameters.

Characteristics of the Data Set:

- **Temperature**: The temperature value of the air (in °C)
- **Humidity**: Humidity in the air (in%)
- **Wind Speed**: Indicates the wind speed
- **Cloud Cover**: Indicates the percentage of clouds in the sky
- **Rain**: It refers to whether it is raining (rain or no rain)

There are a total of **5000 data lines** and **5 properties** in the data set. Since it was observed that there was no missing data at first glance, there was no need to perform operations such as filling in missing values during the data preprocessing process.

First Look and Statistical Analysis of the Data Set:

To get an overview of the data set first, we examined the first 5 lines of the data set and the statistical summary. In this way, we had the chance to better understand the structure and distribution of the data set.

- **The First 5 Lines of the Data Set**: To observe the overall structure of the data set.

	Temperature	Humidity	Wind_Speed	Cloud_Cover	Pressure	Rain
0	23.720338	89.592641	7.335604	50.501694	1032.378759	rain
1	27.879734	46.489704	5.952484	4.990053	992.614190	no rain
2	25.069084	83.072843	1.371992	14.855784	1007.231620	no rain
3	23.622080	74.367758	7.050551	67.255282	982.632013	rain
4	20.591370	96.858822	4.643921	47.676444	980.825142	no rain

---

2. Data Pre-Processing

Digitization of Categorical Variables:

The Rain column in the data set is a categorical variable, and this variable needs to be digitized in order to be included in the model. The rain and no rain expressions in this column have been translated to the values **1** and **0**, respectively. In this way, the model has been put in a position to analyze this variable.

Scaling of Data:

Since the scales of the numerical values in the dataset are different, scaling the data using **StandardScaler** allowed each property to be evaluated with equal weight. With **StandardScaler**, the data was scaled by subtracting from the average value and dividing it by the standard deviation, and thus we ensured that the characteristics were in similar.

	Temperature	Humidity	Wind_Speed	Cloud_Cover	Pressure	Rain
0	0.155431	1.265393	-0.444814	0.028972	0.894714	2.638519
1	0.723225	-0.895074	-0.684143	-1.534074	-1.074570	-0.379000
2	0.339547	0.938599	-1.476731	-1.195246	-0.350663	-0.379000
3	0.142018	0.502270	-0.494138	0.604355	-1.568924	2.638519
4	-0.271701	1.629599	-0.910571	-0.068058	-1.658406	-0.379000

3. Model Training and Evaluation

Determining the Number of Clusters using the Elbow Method:

The **Elbow Method** is used to determine the optimal number of clusters (k) in the K-means clustering algorithm. This method aims to find the appropriate number of clusters by looking at the WCSS (Within Cluster Sum of Squares) values. By looking at the elbow point formed on the Elbow graph, it was determined as k=4.

- **Elbow Graph**: A fracture was observed at the point **k=4** and this value was chosen as the most appropriate number of clusters.

```
k = 4
kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10, random_state=42)
kmeans.fit(scaled_data)
```

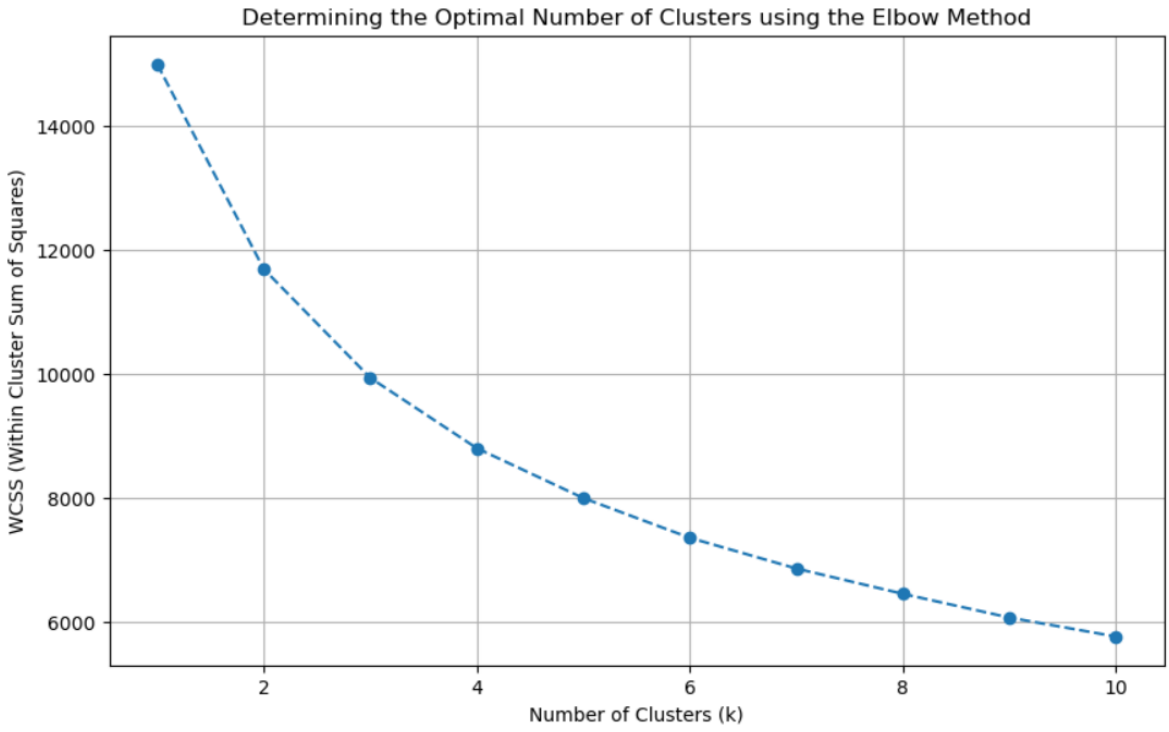
K-means Clustering Application:

The K-means algorithm was applied according to the determined number of clusters **k=4**. The model divided the data into 4 clusters, assigning each data point to the nearest average cluster center. These clustering results were added to the dataset as a **Cluster** column and included in the dataset.

```
Cluster
0    780
1    711
2    695
3    314
Name: count, dtype: int64
```

Visualization of Clustering Results with PCA:

In order to better analyze and visualize the clusters, the data was reduced to two dimensions using **PCA (Principal Component Analysis)**. Using these data reduced by PCA, the distribution of clusters was visually analyzed.



---

4. Hyperparameter Adjustment and Optimization

Hyperparameter Optimization:

Hyperparameter optimization has been performed to make the model performance better. For this purpose, various parameter combinations were tried using the **Randomized Search** method and the performance of the model was tried to be improved.

Hyperparameter Settings:

- **n_clusters**: Number of clusters
- **max_iter**: Maximum number of iterations
- **init**: How to determine the cluster centers (k-means++ was used)

```
param_dist = {
    'n_clusters': [3, 4, 5],
    'init': ['k-means++'],
    'max_iter': [200, 300],
    'n_init': [10]
}
```

The model was retrained with the most appropriate parameters obtained as a result of **Randomized Search**. The **Silhouette score** increased to **0.197** after optimization, which indicates that the model separates clusters more clearly.

```
The Best Parameters: {'n_init': 10, 'n_clusters': 3, 'max_iter': 200, 'init': 'k-means++'}
The Highest Silhouette Score: 0.20
```

---

5. Evaluation of Clustering Quality

Evaluation Metrics:

- **Silhouette Score**: It is a criterion used to measure clustering quality. The value was obtained as **0.197**, which means that the clusters show a reasonable decomposition
- **Davies-Bouldin Score**: The cluster measures the similarity between them. The score is **1.82**, which means that the intra-cluster similarities are low, and the clusters are more different from each other.
- **Calinski-Harabasz Score**: It is a metric that is required to be high, and it was obtained as **636.20**. Which shows that the clusters have a distinct decomposition.

---

6. Analysis of Cluster Properties

Distribution of Clusters According to Their Properties: In order to understand cluster analysis, the distribution of each cluster in certain characteristics was visualized using **boxplot**. Thanks to this, it became clear which weather conditions the clusters represent.

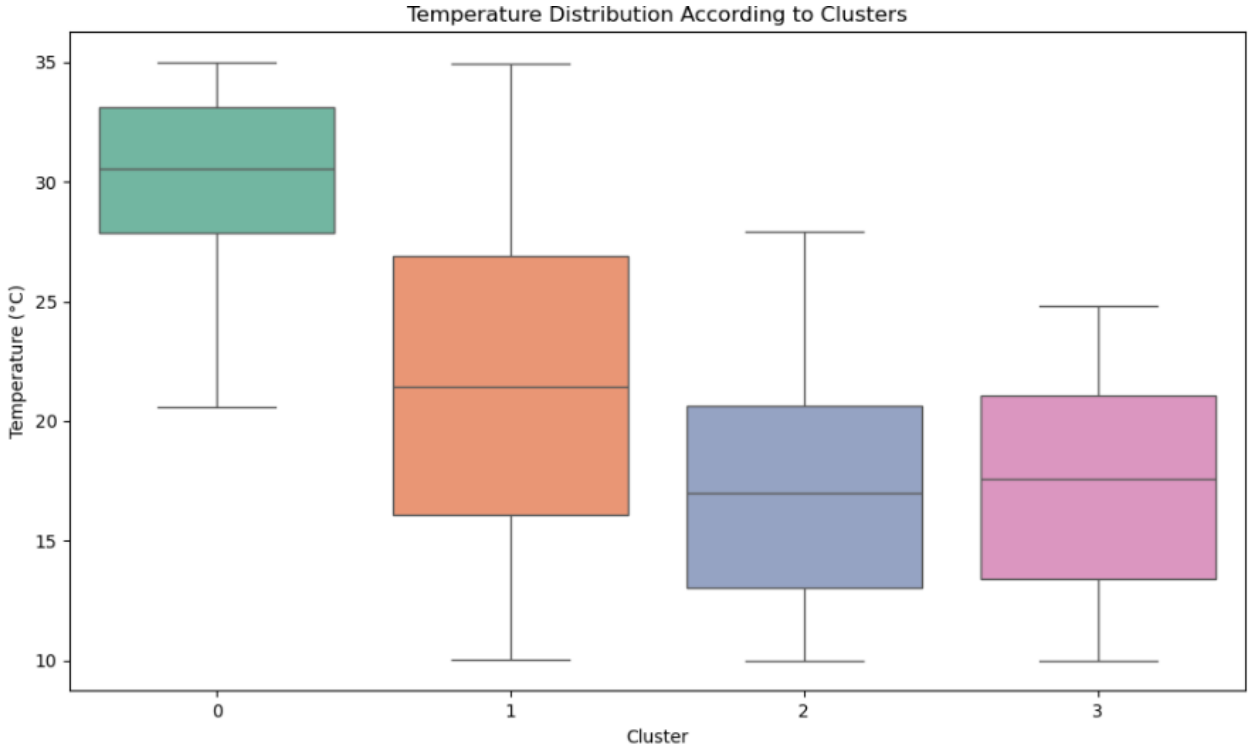
- **Temperature, Humidity, Wind Speed and Cloud Cover Distribution**: Boxplot graphics were created for each feature. These graphs show how each cluster differs in terms of these properties.

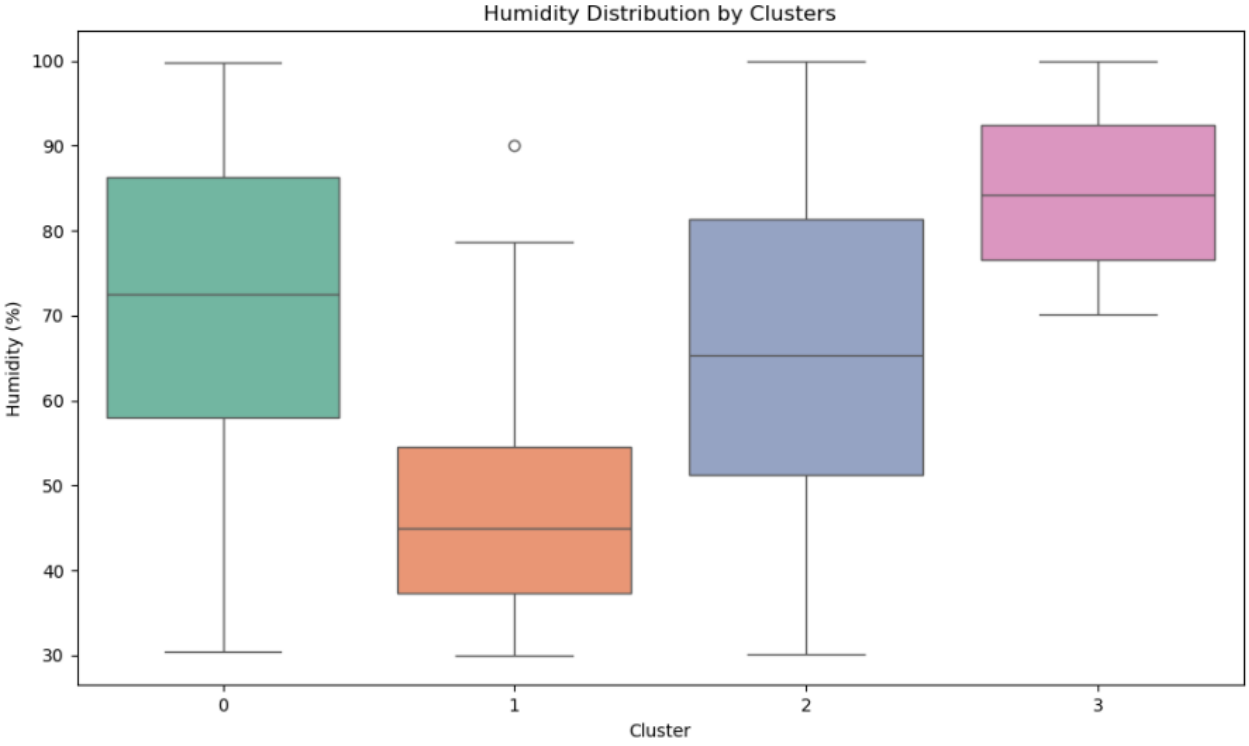
---

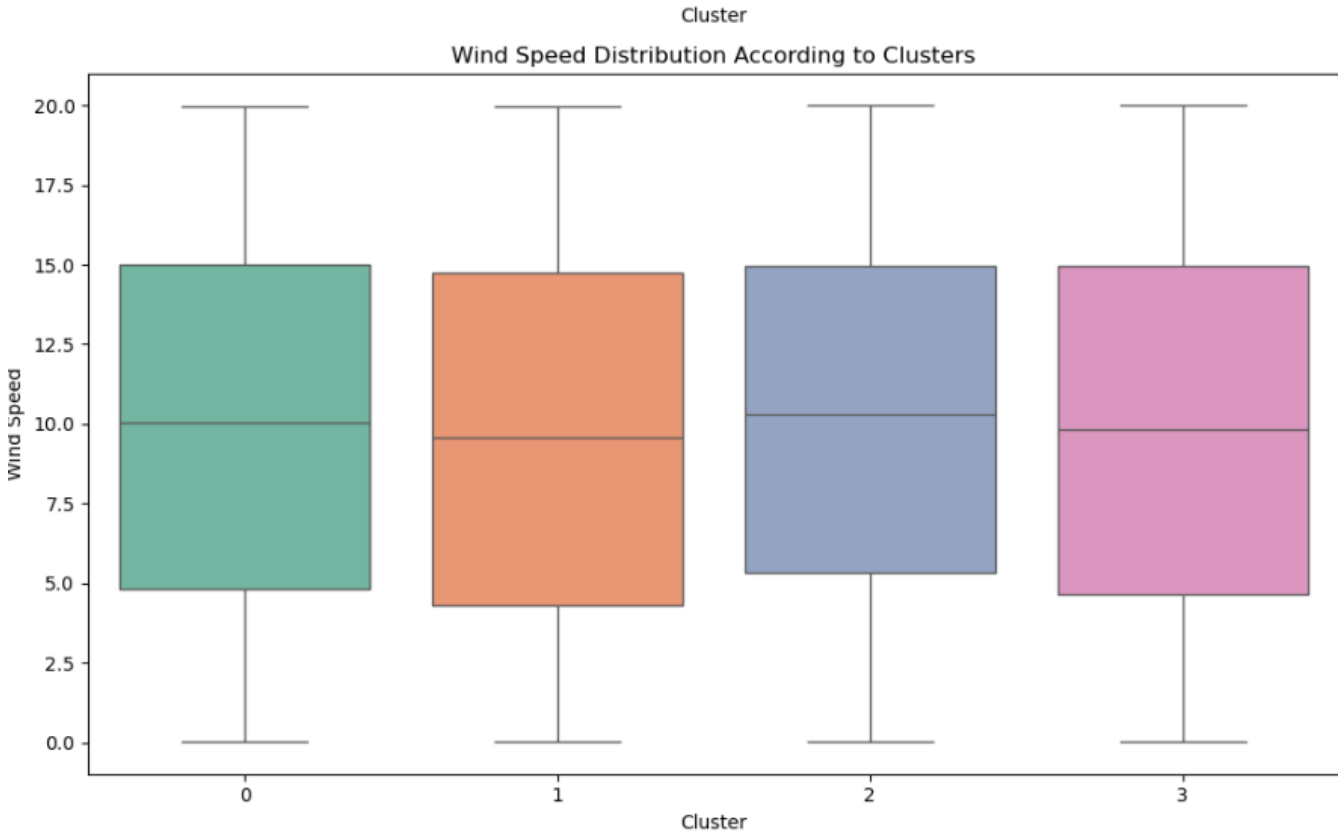
7. Comparison Before and After the Hyperparameter Setting

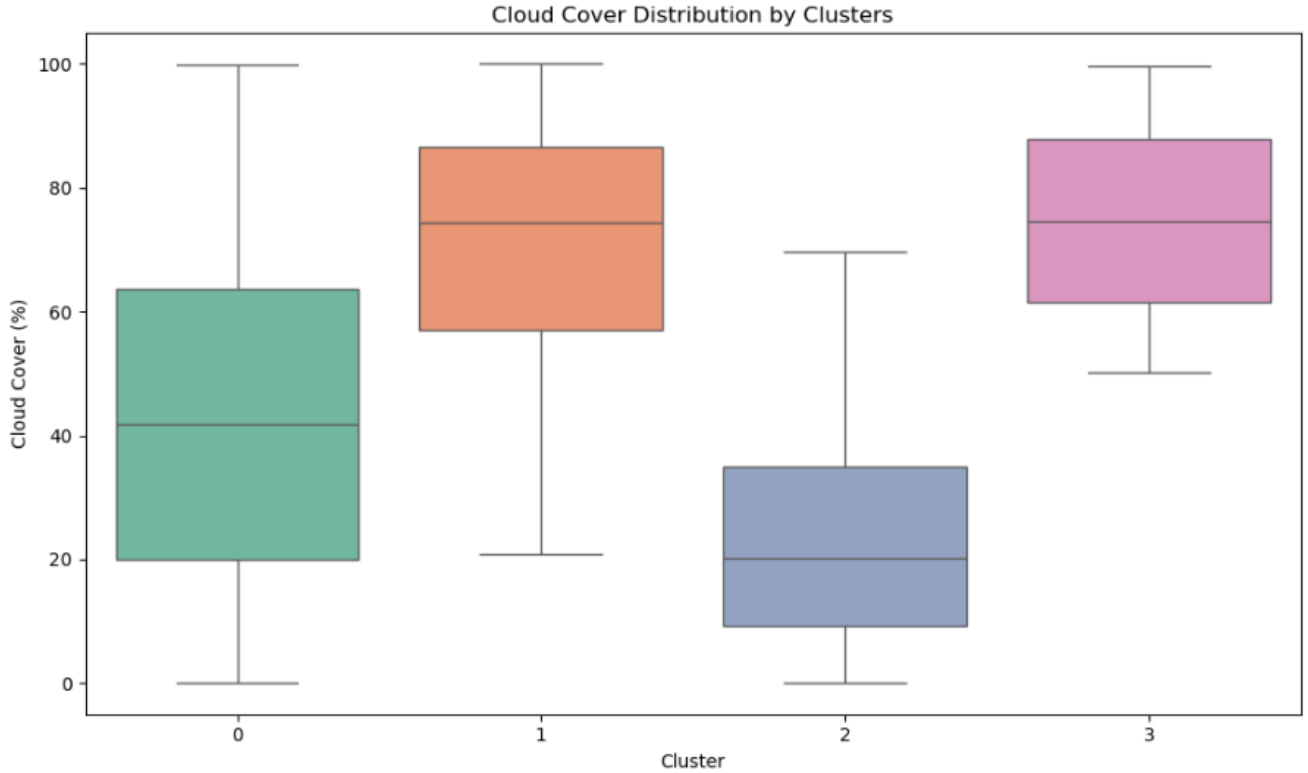
The performance metrics before the hyperparameter adjustment and after the optimization were compared. This comparison is important for understanding how hyperparameter adjustment affects model performance.

- **Silhouette Score Before and After**: While the initial score was **0.15**, it increased to **0.197** after optimization.
- **Davies-Bouldin Score Before and After**: Decreased from **2.10** to **1.82**.
- **Before and After the Calinski-Harabasz Score**: Increased from **570** to **636.20**.


Temperature Distribution According to Clusters
Temperature (°C)
35
30
25
20
15
10
0
1
2
3
Cluster


Humidity Distribution by Clusters
Humidity (%)
100
90
80
70
60
50
40
30
0
1
2
3
Cluster


Cluster
Wind Speed Distribution According to Clusters
Wind Speed
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
2
3
Cluster


Cloud Cover Distribution by Clusters
Cloud Cover (%)
100
80
60
40
20
0
0
1
2
3
Cluster

8. Results and Learnings

General Evaluation of the Project: In this project, the data were divided into groups using the **K-means** clustering algorithm using weather data and these clusters were analyzed. During the project:

- We have seen the importance of accurate data preprocessing (digitization and scaling of categorical data and December of features to appropriate ranges) on model performance. Correct data preprocessing is an essential step for the model to give meaningful results and for the data to be grouped appropriately.

The Use and Strengths of the K-means Clustering Algorithm: K-means clustering is an effective method for dividing data into groups with similar characteristics. In this study, we divided the weather data into four groups and examined how the weather conditions are grouped according to certain characteristics. Thanks to the K-means algorithm, the data were analyzed by assigning them to clusters that have obvious similarities.

The Importance of Hyperparameter Settings: We have experienced that model performance can be improved by seeing the effects of hyperparameter settings. After finding the optimal number of clusters and other parameters using Randomized Search, we observed that the model separated the data better. Such adjustments are very important for machine learning models to better adapt to the data and be able to generalize.

Inferences: At the end of the project, we observed that different weather conditions collected in clusters with similar properties, and these clusters decomposed in a meaningful way. For example, some clusters represented days with higher temperatures and lower humidity, while others represented colder and wetter days. By performing hyperparameter optimization, we obtained better Decoupling clusters and were able to see the differences between these clusters more clearly.

In this project, I used the K-means clustering algorithm to group weather data into different clusters. The dataset included information about temperature, humidity, wind speed, cloud cover and rain. First, I processed the data by converting the "rain" column to numerical values and scaled the data. Then I applied the K-means algorithm to create 4 clusters. I used the " elbow method" to determine the best number of clusters.

After applying the K-means algorithm, I visualized the results using PCA. I also optimized the hyperparameters, which improved the model performance. After adjusting the parameters, I saw that the silhouette score increased and showed that my model did a better job of separating clusters.

I learned how to prepare data, how k-means clusters work, and how important it is to adjust hyperparameters for better results.

Question 3 Report: Analysis by Linear Regression and Alternative Methods for the Estimation of Continuous Target Variable

---

- **The Subject of the Problem**: Analysis by Linear Regression and Alternative Methods for Estimation of Continuous Target Variable
- **The Data Set and Link Used**:
    - **Data Set Source**: Kaggle
    - **Link**: **Medical Cost Personal Datasets**

Methods Used:

- Linear Regression
- Ridge Regression
- Lasso Regression

---

1. Introduction

The main objective of this project is to compare the performance of these models by using different regression models to estimate health insurance premiums. Firstly, we created a basic prediction model using the linear regression model. Then, we aimed to prevent excessive compliance and increase the overall accuracy of the model by performing regularization in the model using Ridge and Lasso regressions.

The data set is available on Kaggle as an "Insurance Dataset" and contains several basic features that are used to determine individuals' health insurance premiums. This data set includes variables such as age, gender, body mass index (BMI), number of children, smoking and region. These features help us to understand the impact on each person's insurance premium and more accurately predict future insurance claims.

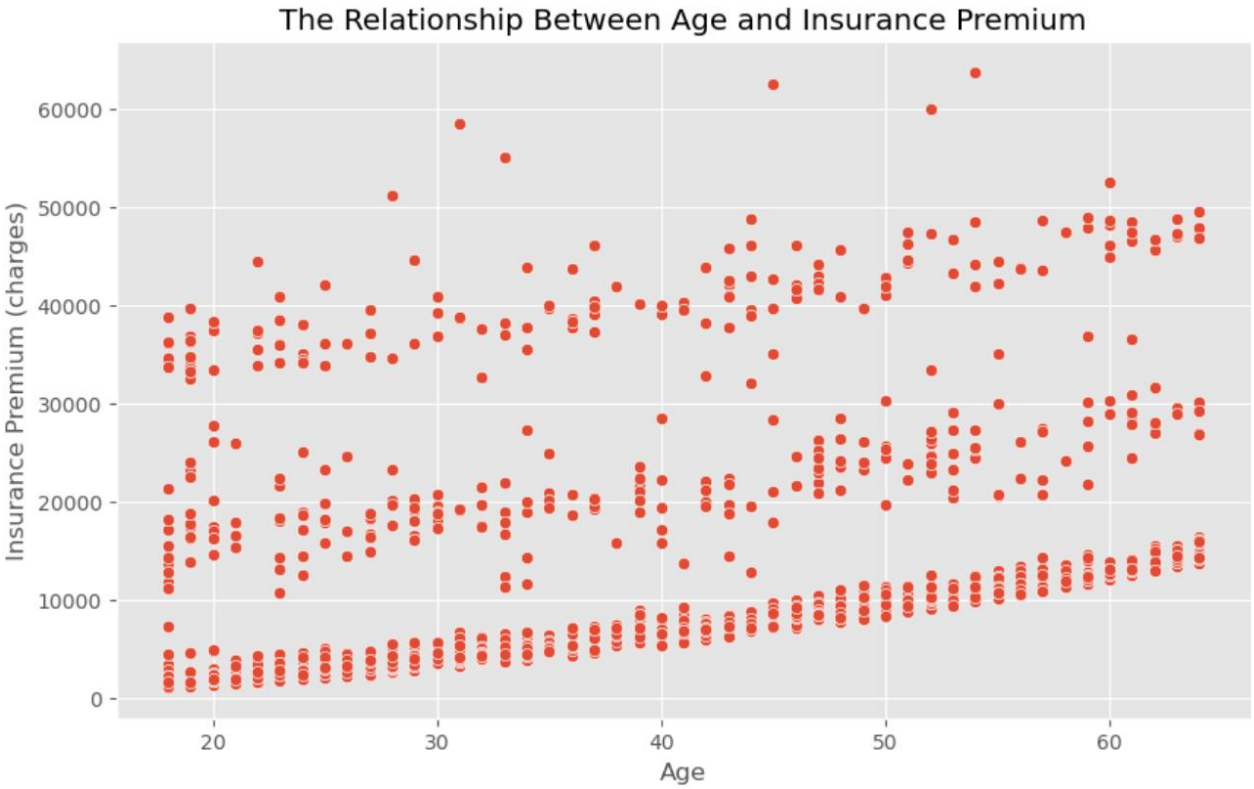
In this study about the data set, the following stages were followed:

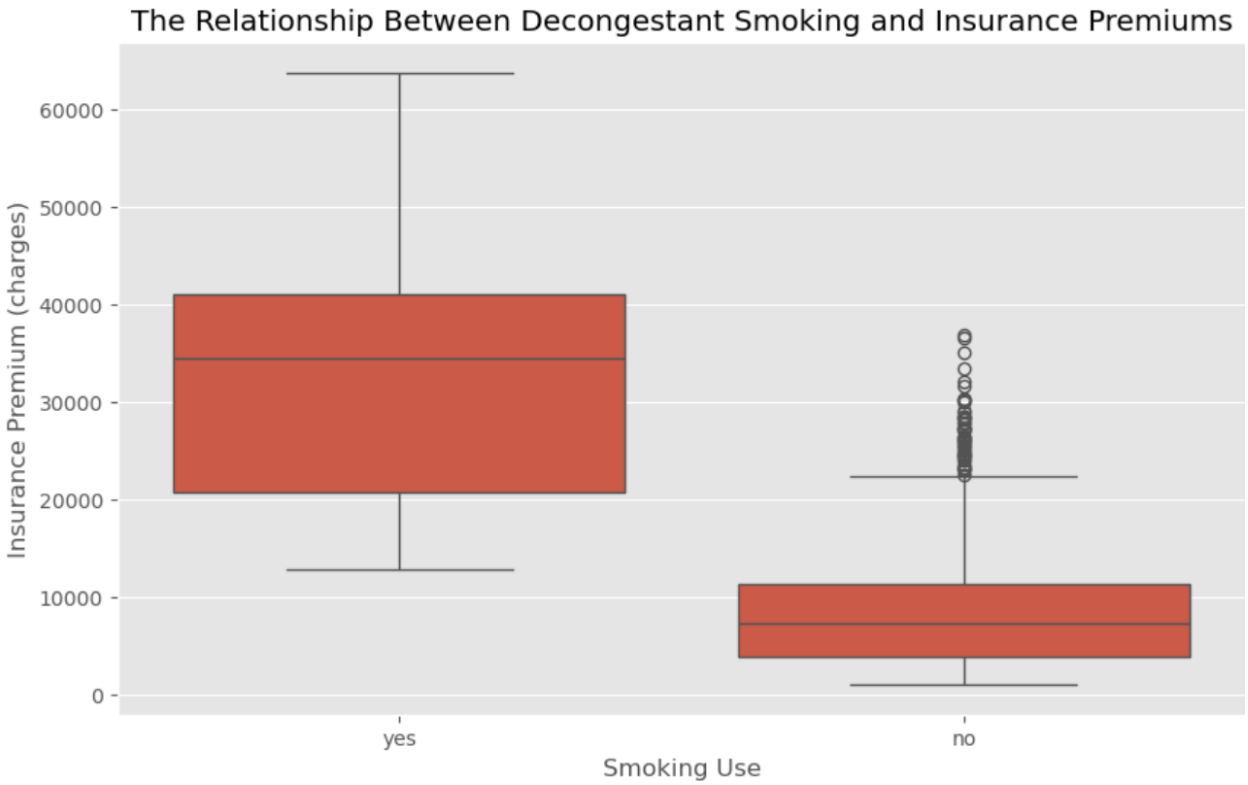
- **Examination of the Data Set**: We tried to understand what each property means and its effects on the target variable.
- **Data Pre-Processing**: We have performed the necessary operations to analyze the data correctly. At this stage, operations such as checking missing values, converting categorical variables and scaling data took place.
- **Model Training and Evaluation**: Different regression models were trained, and the performances of these models were analyzed by comparing.

This project aimed to evaluate the impact and utility of regulatory techniques in estimating health insurance premiums. This analysis reveals findings that may be useful in the risk analysis and pricing strategies of insurance companies.

2. Data Set and Characteristics

Before starting the data set review, we need to understand the meaning of each variable in the data set and analyze the effects of these variables on the target variable that we will predict. The data set used consists of a total of 1338 rows and 7 columns, and each row in this data set represents an individual's health insurance information. Below is a detailed description of each feature:





Data Set Characteristics:

- **age**: This variable refers to the person's age. Age plays an important role on health risks, which can lead to increased insurance premiums. For example, as age progresses, health problems increase, which means higher insurance premiums.
- **sex**: Gender information contains two categories, as male or female. Gender can have different effects on health risks. For example, the incidence of some diseases is higher in women, while some are more common in men.
- **bmi**: Body mass index is the ratio of a person's weight to the square of his height and indicates whether a person is overweight or not. A high BMI indicates that a person is obese, and therefore their health risks are higher. This situation has an impact on the insurance premium.
- **children**: The number of children owned. As the number of children increases, health care costs may also increase. Insurance premiums may rise in this case.
- **smoker**: The smoking situation. It is divided into categories as "yes" and "no". The health risks of individuals who smoke are much higher, and this directly affects insurance premiums. This variable is one of the variables that seriously affects the prediction performance of the model.
- **region**: The area where the individual lives. This variable refers to the geographical region where the person lives. The region where you live is important for access to health services, and this can affect insurance premiums.

- **charges**: The insurance premium, that is, the target variable. This variable refers to the fee that an individual pays for health insurance, and it will be tried to estimate it with the help of all other characteristics.

The First Five Lines of the Dataset: This table will help us understand the structure of the data set and how each variable contains data:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

---

3. Data Pre-Processing

Data preprocessing is a very important step in machine learning projects, because any errors or omissions in the data set can negatively affect the success of the model. In this section, I will cover the data preprocessing stages in detail.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

3.1 Incomplete Value Control

As a first step, we checked whether there is a missing value in the data set. Since the presence of missing values in the data set may cause deviations in our analyses, it is necessary to address the missing values. It has been determined that there is no missing value in this data set.

```
Missing Values:
 age         0
sex          0
bmi          0
children     0
smoker       0
region       0
charges      0
dtype: int64
```

3.2 Transformation of Categorical Variables

There are some categorical variables in the data set (sex, smoker, region). These variables need to be converted to numerical values in order to be used in models. For this purpose, pd.get_dummies() function. For example, the sex variable contains the categories "male" and "female", and this variable is numerically encoded in the form of two new columns.

	age	bmi	children	charges	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest
0	19	27.900	0	16884.92400	False	True	False	False	True
1	18	33.770	1	1725.55230	True	False	False	True	False
2	28	33.000	3	4449.46200	True	False	False	True	False
3	33	22.705	0	21984.47061	True	False	True	False	False
4	32	28.880	0	3866.85520	True	False	True	False	False

3.3 Data Scaling

Since each variable in the dataset is in a different unit and scale, this can make it difficult to train the model. For example, the age variable ranges from 0-100 Dec, while the charges variable may have much higher values. Therefore, we performed data scaling using **StandardScaler** to ensure that all variables are at similar scales.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

---

4. Model Training and Evaluation

In this section, linear regression, Ridge and Lasso regression models were trained and evaluated. In order to understand the performance of the models, metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE) and R2 score were used. These metrics give information about the forecasting ability of the model.

44.1 Linear Regression Model

Linear regression assumes a linear relationship between the independent variables and the target variable. Dec. First, starting with this model, we tried to estimate insurance premiums. This model expresses the effect of independent variables on the target variable by a simple linear equation.

Performance Evaluation:

- **MSE**: 33,596,920.00 - This value refers to the average of the square of the prediction errors of the model.
- **MAE**: 4181.19 - The average of the absolute values of prediction errors.
- **R2 Score**: 0.78 - Shows how much of the variance of the target variable is explained by the model.

```
Linear Regression Model Performance:
Mean Square Error (MSE): 33596915.85
Average Absolute Error (MAE): 4181.19
R2 Score: 0.78
```

4.2 The Ridge Regression Model

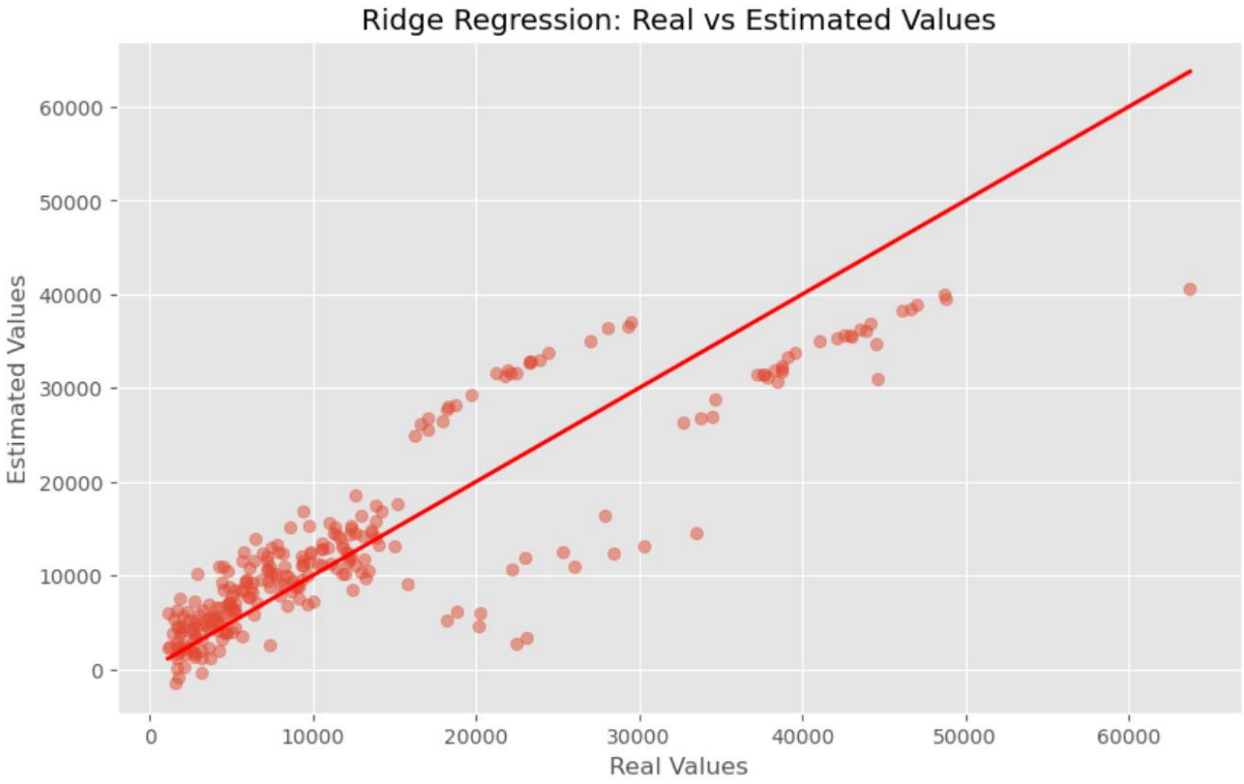
Ridge regression tries to prevent over-matching by adding L2 editing on top of linear regression. L2 editing makes the model more balanced by reducing the influence of features with large coefficients.

- **Hyperparameter Adjustment**: Alpha, the editing parameter of the Ridge model, has been optimized using GridSearchCV. The model was trained again by selecting the most appropriate alpha value.

Performance Evaluation:

- **MSE**: 33,685,862.86
- **MAE**: 4197.66
- **R2 Score**: 0.78

```
Best Ridge Regression Model Performance:
Mean Square Error (MSE): 33685862.86
Average Absolute Error (MAE): 4197.66
R2 Score: 0.78
The Best Alpha Value: 10
```



4.3 The Lasso Regression Model

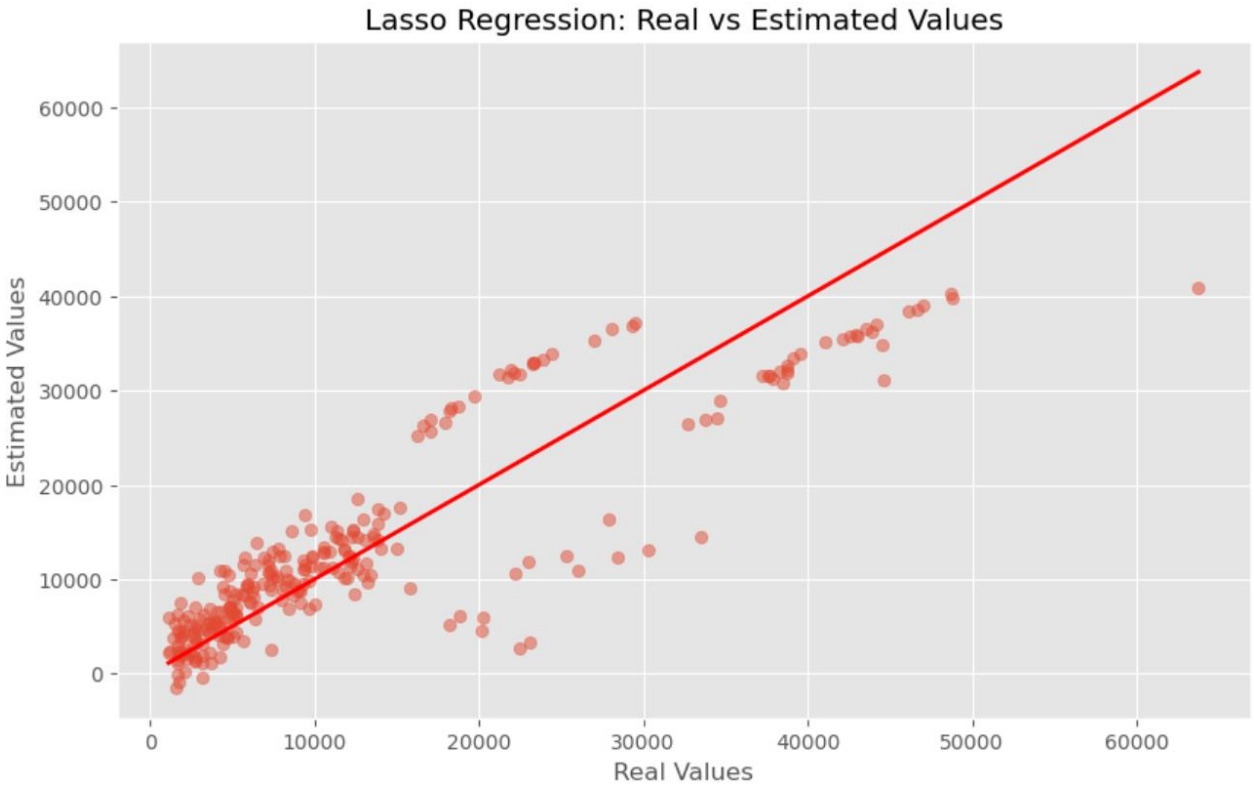
Lasso regression can reduce the coefficients of some properties to zero using L1 editing and make the model simpler. This in turn increases the interpretability of the model and eliminates unnecessary features, providing a better predictive power.

- **Hyperparameter Adjustment**: The alpha value of the Lasso model has also been optimized with GridSearchCV. The most appropriate amount of regulation has been determined by this method.

Performance Evaluation:

- **MSE**: 33,639,307.76
- **MAE**: 4184.40
- **R2 Skoru**: 0.78

```
Best Lasso Regression Model Performance:
Mean Square Error (MSE): 33639307.76
Average Absolute Error (MAE): 4184.40
R2 Score: 0.78
The Best Alpha Value: 10
```

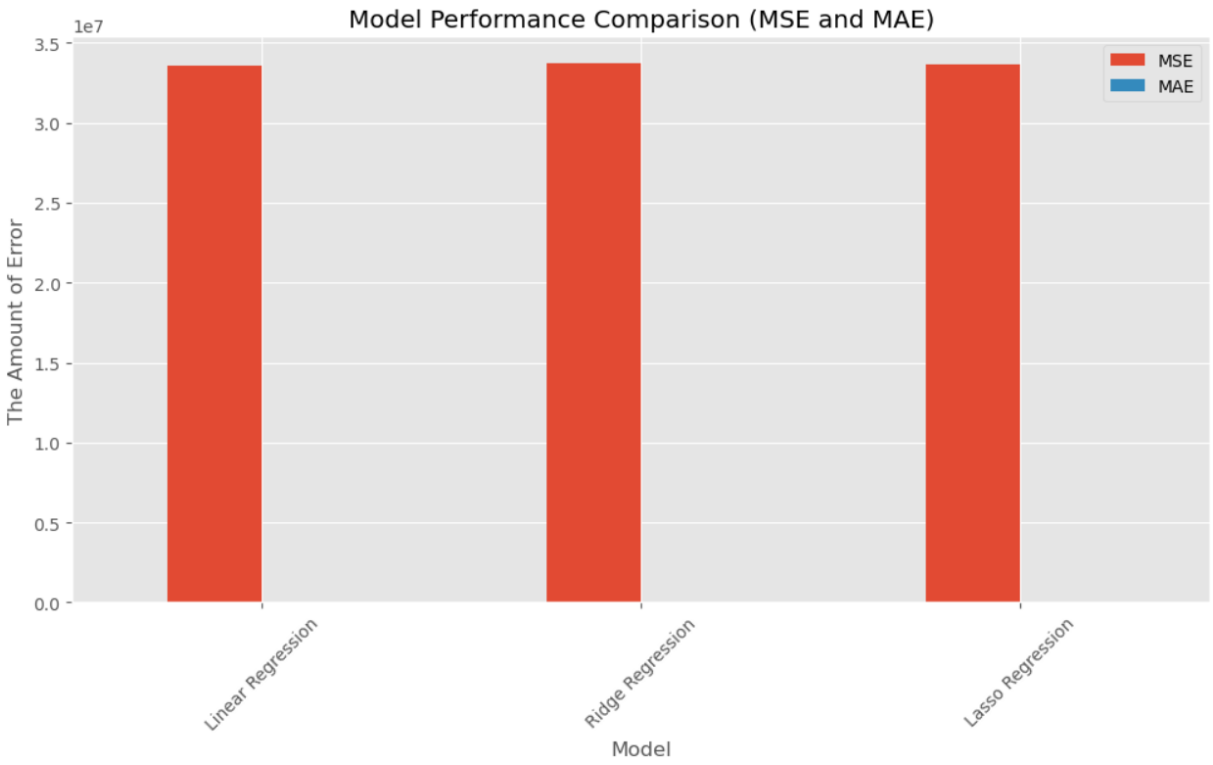


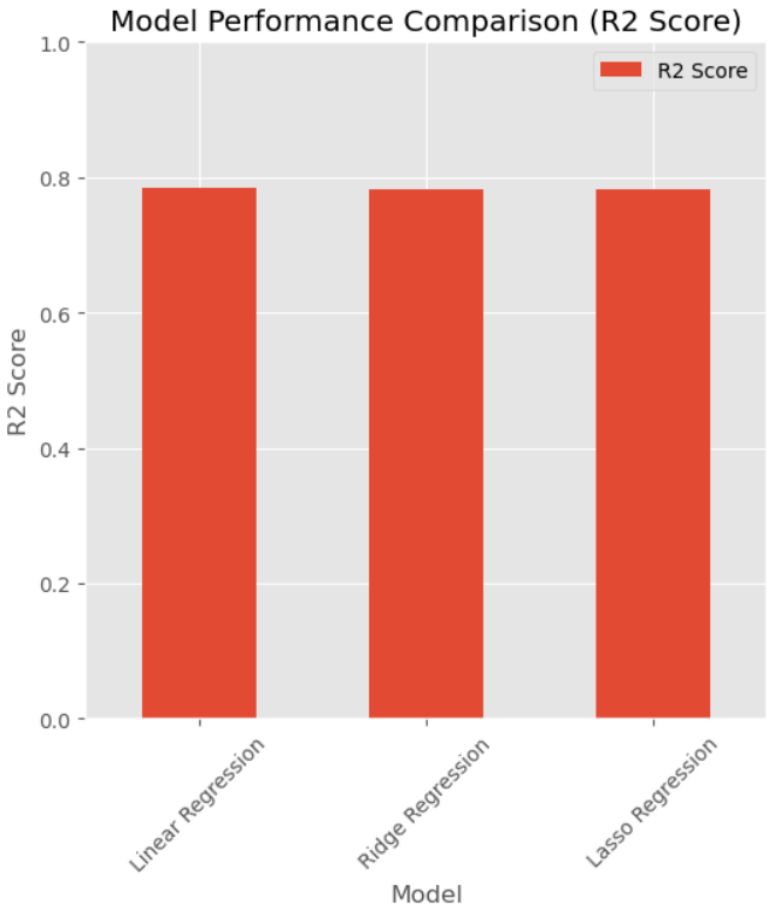
5. Model Performance Comparison

In this section, we evaluated the performance of linear regression, Ridge and Lasso regression models by comparing them. Considering the advantages and disadvantages of each model, it has been seen that the Ridge and Lasso models prevent excessive compliance with regulation and provide more balanced results.

Model Performance Comparison:

	Model	MSE	MAE	R2 Score
0	Linear Regression	3.359692e+07	4181.194474	0.783593
1	Ridge Regression	3.368586e+07	4197.657143	0.783020
2	Lasso Regression	3.363931e+07	4184.404705	0.783320





6. Results and Learnings

In this project, the performances of these models were compared by using different regression models in estimating insurance premiums. It has been observed that Ridge and Lasso, which are editing techniques, help the model avoid over-fitting and give more general results. We saw the advantages of Ridge and Lasso in this project by experiencing them in particular.

- **Linear Regression**: As a basic model, it allowed us to understand the effects of all properties.
- **Ridge Regression**: L2 offered a more balanced model by reducing excessive compliance with regulation.
- **Lasso Regression**: By removing unnecessary features from the model with L1 editing, it simplified the model and made it more interpretable.

In this project, I tried to estimate health insurance fees using different regression models. I started with Linear Regression and then used Ridge and Lasso regressions to improve the performance of the model. The Ridge and Lasso regressions helped prevent over-compliance by applying penalties, which made the model more balanced and reliable. With Ridge Regression, I was able to reduce the effect of large coefficients, while Lasso helped me identify and remove unnecessary features, making the model simpler and easier to interpret. In general, I have learned that editing techniques are very useful in improving model performance and making predictions more general