T. C.
RECEP TAYYİP ERDOĞAN UNIVERSITY
FACULTY OF ARCHITECTURE AND ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

CEN411 – Data Mining
P-1: Classification with Decision Tree and Model Comparison
P-2: K-Means Clustering and Analysis
P-3: Continuous Variable Prediction and Regression Models

Student
Emirhan AŞIK – 201401009

Instructor
Dr. Abdulgani KAHRAMAN

2024

Contents

1. INTRODUCTION

This assignment aims to solve different types of problems using data mining techniques. Data mining is a process that aims to extract meaningful information from large and complex datasets and is one of the cornerstones of modern data analytics processes. In this study, various data mining methods and algorithms were applied to model, analyze, and evaluate different types of data. This assignment focuses on three main problems.

1. Classification Problem: The classification performances of algorithms such as decision trees, random forests, and support vector machines (SVM) have been compared..
2. Clustering Problem: The dataset was clustered using the K-Means algorithm, and these clusters were analyzed.
3. Regression Problem: For a continuous target variable, linear regression, Ridge regression, and random forest regression models were compared to determine the most suitable model.

This report thoroughly discusses the selection criteria for the datasets used, data preprocessing steps, modeling processes, and the results obtained. Appropriate algorithms were chosen for each problem, and their performance was evaluated both visually and statistically. The primary aim of the report is to demonstrate the effectiveness of data mining techniques and introduce best practices for addressing different problems.

2. P-1: CLASSIFICATION WITH DECISION TREE AND MODEL COMPARISON

In this section, a classification problem was solved on the Titanic dataset, and three different machine learning algorithms (Decision Tree, Random Forest, and Support Vector Machines) were compared. As part of the data mining process, data preprocessing, model training, and performance evaluation stages were carried out.

Each algorithm was evaluated using metrics such as accuracy, precision, recall, and F1 score, and the results were visualized. The analyses were conducted to understand the strengths and weaknesses of each algorithm and to select the most suitable model for specific types of problems.

In this context, the analysis of the Titanic dataset and the evaluation of the models provided a comprehensive foundation for understanding different approaches to classification problems and their successes.

2.1 Selection Criteria and Sources of the Dataset Used

The Titanic dataset was downloaded from the Kaggle platform. You can access the dataset via the following link:

Kaggle Titanic Dataset

This dataset contains various demographic and travel information about passengers and presents whether a passenger survived the Titanic disaster ("Survived" variable) as the target variable. The dataset consists of 891 rows and 12 columns. The selection criteria for the dataset used are as follows:

- Various Data Types: The dataset contains both categorical ("Sex", "Embarked") and numerical ("Age", "Fare") variables, making it suitable for testing different data preprocessing steps and evaluating algorithm performance against these steps.
- Missing Data Management: Since the dataset contains missing values, it provides an ideal opportunity to apply missing data management strategies and assess their impact on model performance.
- Binary Classification Problem: The target variable ("Survived") creates a binary classification problem, making the dataset suitable for evaluating classification algorithms.
- Popular and Well-Known Dataset: The Titanic dataset is a commonly used benchmark in the field of data science, making it a standard for testing and comparing machine learning models.

2.2 Data Preprocessing Steps

Data preprocessing is a critical step that directly impacts the performance of machine learning models. Before applying classification algorithms to the Titanic dataset, several preprocessing operations were performed on the dataset. This section provides a detailed explanation of how missing data was handled, feature selection was carried out, and categorical variables were transformed.

2.2.1 Data Pre-Processing Stages

Some columns in the Titanic dataset contain missing values, particularly:

- Cabin: A large portion is missing, so it has been completely removed during the modeling process.
- Age: Some rows have missing age information.
- Embarked: Some passengers have missing embarkation port information.

Missing data can prevent machine learning models from producing accurate results and, in some cases, can even make algorithms impossible to run. Therefore, missing data has been processed. Missing values have been filled using the forward fill method. This method ensures that missing cells are filled with the value from the previous row. This way, missing data is consistently filled, and data integrity is maintained. The code used is as follows:

```
# Filling missing values
# Forward fill method is applied to handle missing values, replacing them with the value from the
previous row.
data.fillna(method='ffill', inplace=True)
Neden Forward Fill Seçildi?
```

Other methods, such as filling with the mean or median, could also be used. However, these methods are generally more suitable for numerical data. The forward fill method, on the other hand, can be applied to both categorical and numerical variables.

The forward fill method preserves the meaningfulness of the data flow by filling missing values with the previous value in the dataset. In the Titanic dataset, this method is particularly useful for categorical variables like Embarked.

Note: A FutureWarning was received in the console output for the fillna method. This warning indicates that a different alternative (such as bfill) may need to be used in future versions of pandas. However, this does not affect the accuracy of the current implementation.

.

2.2.2 Feature Selection

Not all columns in the dataset are meaningful for machine learning models. Feature selection is important for improving model accuracy and reducing unnecessary computational load. In the Titanic dataset, some columns have been removed:

Extracted Features::

- Name: The passengers' name information is not directly related to the survival status. Additionally, it does not provide meaningful contribution to the model due to containing a high degree of unique values.
- Ticket: The passenger ticket numbers have been removed as they contain randomness and do not carry meaningful information.
- Cabin: This column has been removed due to containing a large amount of missing data.

Features Used:

- Numeric columns: Age, Fare, Pclass, SibSp, Parch.
- Categorical columns: Sex, Embarked.

```
# Selecting features and target variable
# The columns 'Name', 'Ticket', and 'Cabin' are removed as they are either irrelevant or contain
excessive missing values.
target_column = "Survived"
X = data.drop(columns=[target_column, "Name", "Ticket", "Cabin"])  # Selecting features
y = data[target_column]  # Separating the target variable
```

Why were these features chosen?

- Categorical Variables: Gender (Sex) and port of embarkation (Embarked) are important factors affecting passenger survival.
- Numerical Variables: Age and ticket price (Fare) contain important demographic and economic information that can affect the probability of survival.

2.2.3 Transforming Categorical Variables

Since machine learning models work with numerical data, categorical variables (e.g. “Sex” and “Embarked”) were converted into numerical format:

1. One-Hot Encoding:
   - Reason for Use: It creates separate columns for each category of categorical variables like "Sex" and "Embarked" and represents each column with binary (0 or 1) values.
   - Application: This method has been applied to the "Embarked" and "Sex" variables.
2. Label Encoding:
   - Reason for Use: The target variable "Survived" (0: Did not survive, 1: Survived) has been converted to a numerical format as it is a binary classification problem.
   - Application: The "Survived" column has been labeled using LabelEncoder.

```
# Encoding categorical variables
# One-hot encoding is applied to 'Sex' and 'Embarked' columns, while the target variable
'Survived' is label-encoded.
X = pd.get_dummies(X, drop_first=True)  # Applying one-hot encoding to categorical variables
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)  # Encoding the target variable
```

- The "Sex" variable has been converted into a column named "Sex_male" (1: Male, 0: Female).
- The "Embarked" variable has been split into "Embarked_Q" and "Embarked_S" columns. The first category ("C") has been left as the reference category.

2.3 Modeling Process

The modeling process involves training and evaluating three different machine learning algorithms (Decision Tree, Random Forest, and Support Vector Machines) on the dataset processed through data preprocessing stages. In this section, the methods and hyperparameters used for training each algorithm are discussed in detail.

2.3.1 Decision Tree Model Training and Hyperparameter Settings

Why Decision Tree?

- Decision trees are simple and interpretable models that can effectively work with both numerical and categorical data.
- A single tree structure makes predictions by splitting into branches to learn all classes of the data.

Training Process: The model has been created using the DecisionTreeClassifier class and trained on the training portion of the dataset.

Hyperparameters:

- random_state=42: Ensures the reproducibility of the model's results.
- Other hyperparameters have been left at their default values.

```
# Decision Tree model training
clf = DecisionTreeClassifier(random_state=42)  # Initialize the model
clf.fit(X_train, y_train)  # Train the model on training data
```

Evaluation:

- The model's performance has been evaluated on the test dataset.
- Metrics such as accuracy, precision, recall, and F1 score have been calculated.

```
# Decision Tree model evaluation
y_pred_clf = clf.predict(X_test)  # Predict the target variable for the test set
print("\nDecision Tree Model Performance:")
print(classification_report(y_test, y_pred_clf))  # Output evaluation metrics
```

2.3.2 Training the Random Forest Model and Hyperparameter Settings

Why Random Forest?

- It creates multiple decision trees and increases generalization ability by combining the predictions of these trees.
- It provides higher accuracy compared to decision trees and reduces overfitting.

Training Process: The model has been created using the RandomForestClassifier class and trained on the training data.

Hyperparameters:

- random_state=42: Ensures the reproducibility of the results.
- Default hyperparameters have been used.

```
# Random Forest model training
rf_clf = RandomForestClassifier(random_state=42)  # Initialize the model
rf_clf.fit(X_train, y_train)  # Train the model on training data
```

Evaluation:

- The model's performance has been evaluated on the test dataset.
- The following code generates the performance evaluation metrics:

```
# Random Forest model evaluation
y_pred_rf = rf_clf.predict(X_test)  # Predict the target variable for the test set
print("\nRandom Forest Model Performance:")
print(classification_report(y_test, y_pred_rf))  # Output evaluation metrics
```

2.3.3 Support Vector Machines (SVM) Model Training and Hyperparameter Settings

Why SVM?

- It performs effectively on high-dimensional datasets.
- It can capture complex nonlinear relationships.
  It carries a lower risk of overfitting.

Training Process: The model has been created using the SVC class and trained on the training portion of the dataset.

Hyperparameters:

- random_state=42: Ensures the reproducibility of the results.
- Default hyperparameters have been used (e.g., kernel='rbf').

```
# SVM model training
svm_clf = SVC(random_state=42)  # Initialize the model
svm_clf.fit(X_train, y_train)  # Train the model on training data
```

Evaluation:

- The performance of the model was evaluated on the test dataset:

```
# SVM model evaluation
y_pred_svm = svm_clf.predict(X_test)  # Predict the target variable for the test set
print("\nSupport Vector Machine (SVM) Model Performance:")
print(classification_report(y_test, y_pred_svm))  # Output evaluation metrics
```

2.4 Model Performance and Evaluation

In this section, the performance of three different models (Decision Tree, Random Forest, and SVM) on the test dataset is evaluated using metrics such as accuracy, precision, recall, and F1 score. Additionally, the classification successes of the models are visualized with a confusion matrix.

2.4.1 Performance Metrics (Accuracy, Precision, Recall, F1 Score)

The performance of each model was measured using the classification_report function. This report includes the following metrics:

- Accuracy: The ratio of correct classifications to the total classifications.
- Precision: Indicates how many of the predicted positive examples are actually positive.
- Recall: It shows how many of the actual positive examples were correctly predicted.
- F1 Score: The harmonic mean of precision and recall.

The code below calculates the performance metrics for each model:

```
# Evaluate each model and print classification report
print("\nDecision Tree Model Performance:")
print(classification_report(y_test, y_pred_clf))

print("\nRandom Forest Model Performance:")
print(classification_report(y_test, y_pred_rf))

print("\nSupport Vector Machine (SVM) Model Performance:")
print(classification_report(y_test, y_pred_svm))
```

The outputs in the console are given in the tables below.

Decision Tree Model Performance

	PRECISION	RECALL	F1-SCORE	SUPPORT
0	0.79	0.77	0.78	105
1	0.68	0.70	0.69	74
ACCURACY			0.74	179
MACRO AVG	0.74	0.74	0.74	179
WEIGHTED AVG	0.74	0.74	0.74	179

Random Forest Model Performance

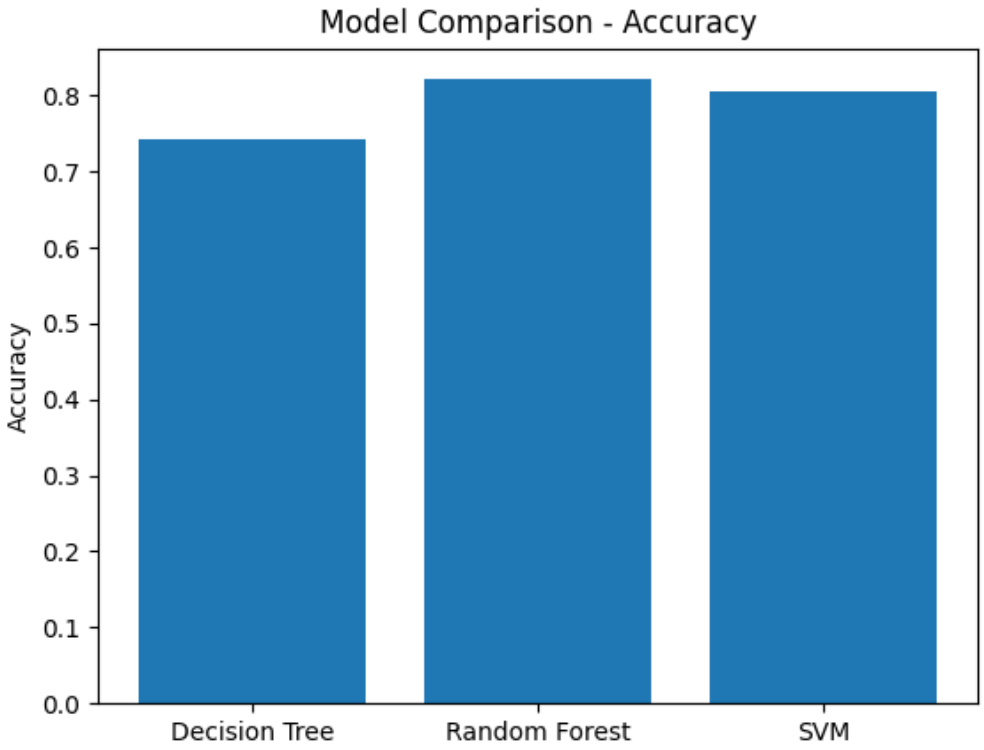
	PRECISION	RECALL	F1-SCORE	SUPPORT
0	0.82	0.89	0.85	105
1	0.82	0.73	0.77	74
ACCURACY			0.82	179
MACRO AVG	0.82	0.81	0.81	179
WEIGHTED AVG	0.82	0.82	0.82	179

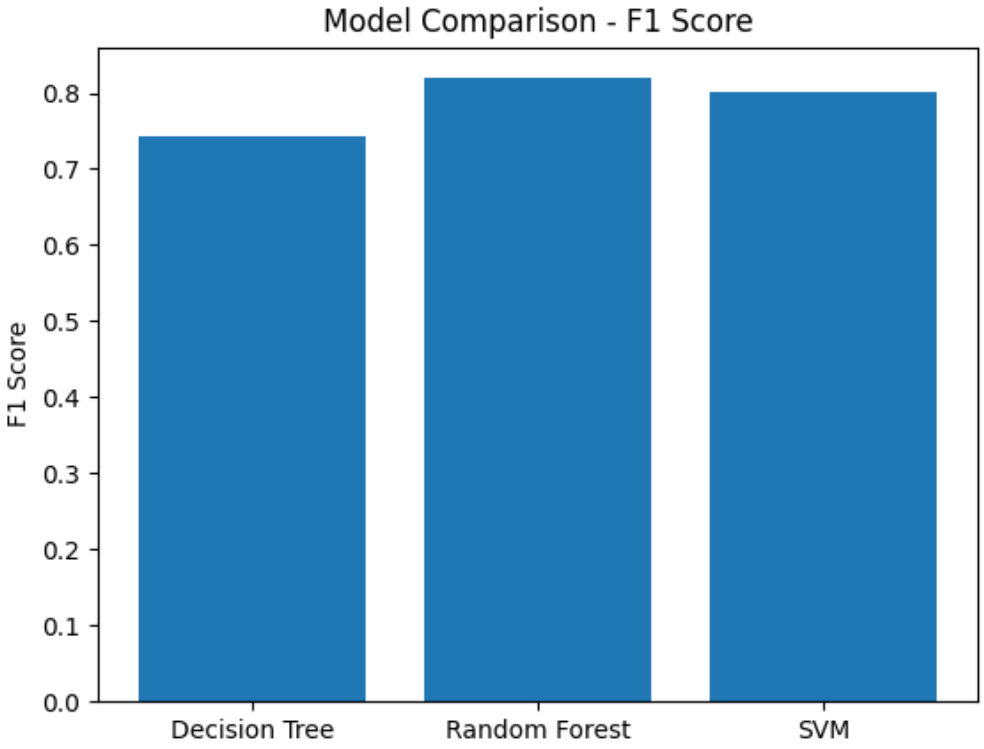
Support Vector Machine (SVM) Model Performance

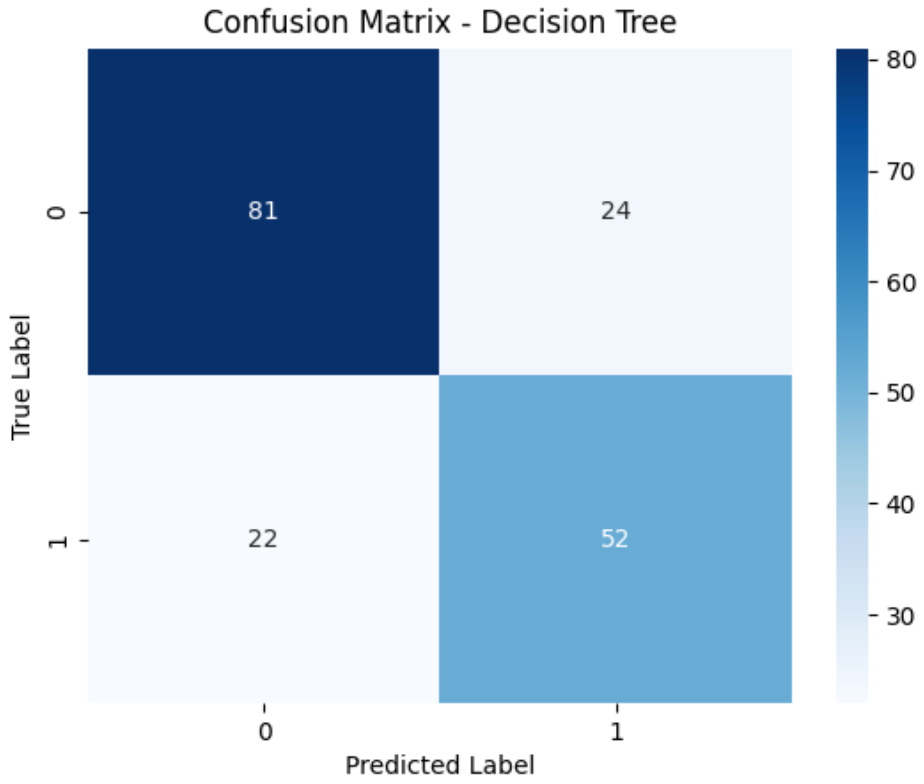
	PRECISION	RECALL	F1-SCORE	SUPPORT
0	0.80	0.89	0.84	105
1	0.81	0.69	0.74	74
ACCURACY			0.80	179
MACRO AVG	0.81	0.79	0.79	179
WEIGHTED AVG	0.80	0.80	0.80	179

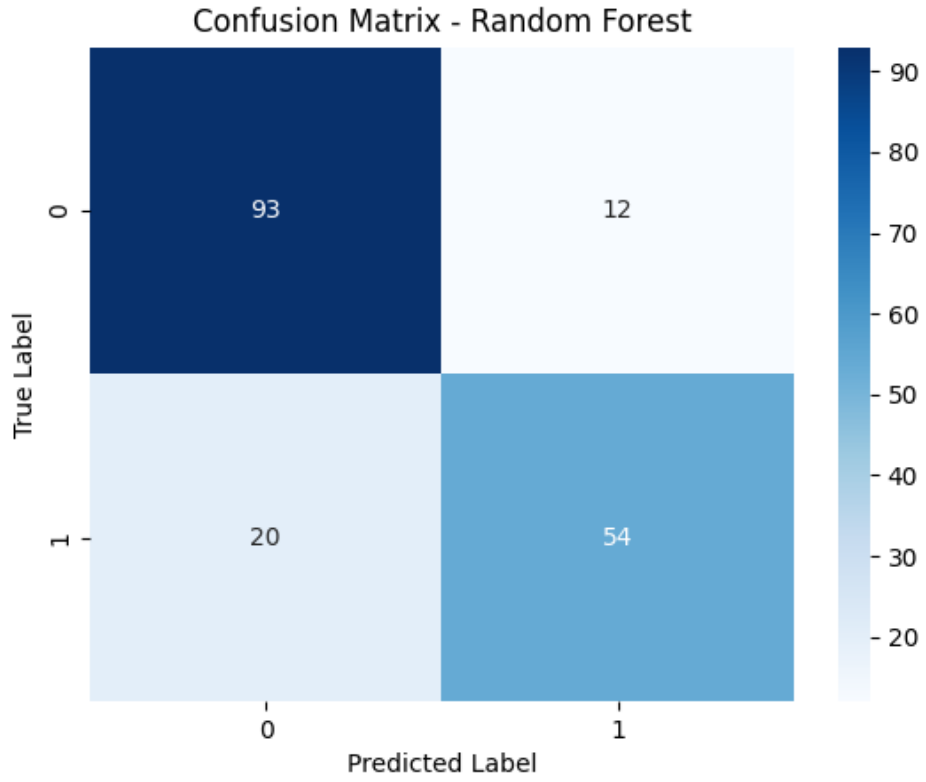
2.4.2 Confusion Matrix and Visualizations

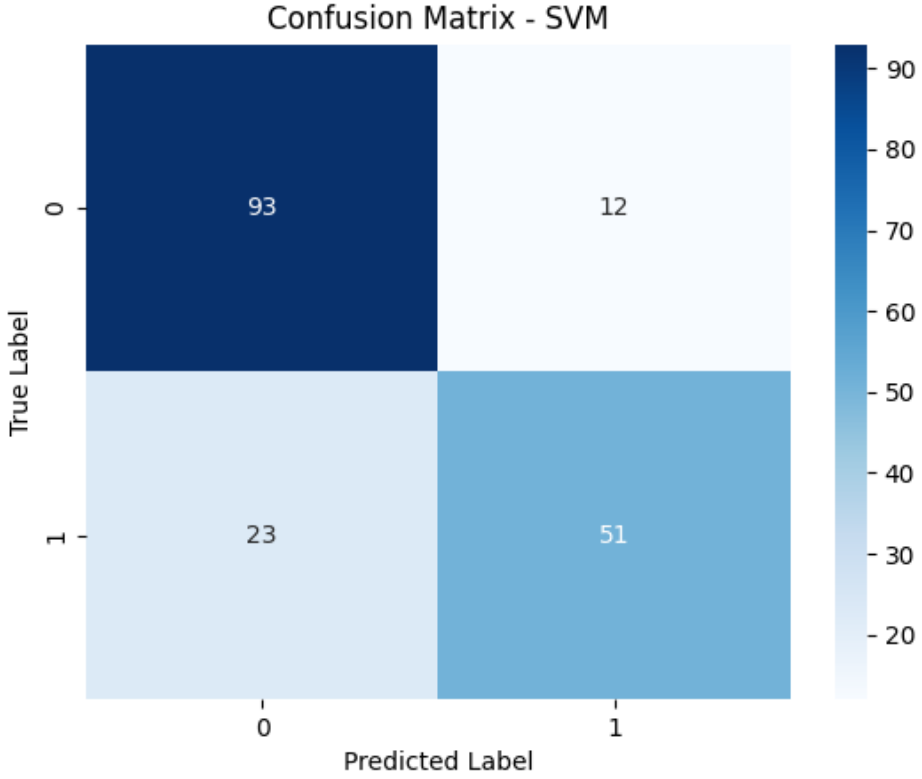
The graphical visuals when the code runs are as follows:




Model Comparison - F1 Score
F1 Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Decision Tree
Random Forest
SVM


Confusion Matrix - Decision Tree
True Label
0
1
81
24
22
52
0
1
Predicted Label
80
70
60
50
40
30


Confusion Matrix - Random Forest
93
12
20
54
True Label
Predicted Label
0
1
90
80
70
60
50
40
30
20


Confusion Matrix - SVM
93
12
23
51
True Label
Predicted Label
0
1
90
80
70
60
50
40
30
20

2.5 Conclusion

In this section, the findings and general results from the analysis of three different classification algorithms (Decision Tree, Random Forest, and Support Vector Machines) on the Titanic dataset are presented.

Decision Tree:

- The Decision Tree model, being a fast and interpretable algorithm, offers an entry-level approach for classification problems.
- However, it carries the risk of overfitting, which limits its generalization ability. This is reflected in the relatively low accuracy rate on the test data (74%).
- The misclassification rate is higher compared to the proportion of correct positive and negative predictions.

Random Forest:

- The Random Forest model, an ensemble model of decision trees, has shown the best performance. The accuracy on the test data was 82%, and the F1 score was also 82%.
- The model's low false positive and false negative rates highlight its ability to generalize well. It has proven effective in reducing misclassifications.
- Due to its capacity to learn more complex relationships, this model can be considered the most suitable algorithm for the Titanic dataset.

SVM:

- The SVM model demonstrated a performance close to the Random Forest model, with an accuracy of 80% and an F1 score of 80%.
- SVM is particularly effective with high-dimensional datasets and has also produced generally successful results with the Titanic dataset.
- However, the false negative rate is higher compared to the Random Forest model, indicating limited performance in predicting survivors.

Which Model is More Suitable?

The most suitable algorithm for the Titanic dataset is the Random Forest model, which outperforms others in metrics such as accuracy and F1 score. This model effectively demonstrates its generalization ability with a low misclassification rate.

When Can the Decision Tree Be Used?

The Decision Tree model is suitable for problems that require fast predictions and prioritize interpretability. However, for large and complex datasets, models like Random Forest should be preferred.

Conclusion

This analysis demonstrates that the choice of algorithm in classification problems depends on the type of problem and the characteristics of the dataset. In the Titanic dataset:

- The Random Forest model stands out as the strongest algorithm in terms of accuracy and generalization capacity.
- The Decision Tree provides a fast-starting point as a basic model but is limited in generalization.
- SVM emerges as a strong alternative in terms of accuracy, but due to its lower recall rate, it performs weaker in predicting false negatives compared to the Random Forest model.

In conclusion, for datasets of moderate complexity like the Titanic dataset, the Random Forest model is recommended. In future studies, it is advised to apply more advanced techniques, such as hyperparameter optimization, taking into account the size and structure of the dataset.

3. P-2: K-MEANS CLUSTERING AND ANALYSIS

In this section, clustering analysis has been performed on a dataset using the K-Means clustering algorithm. The dataset was first appropriately processed, and then clusters were formed by applying the algorithm. Finally, the quality of the clusters was evaluated, and the results were visualized.

3.1. Selection Criteria and Sources of the Dataset Used

The dataset used in this study is one that has various features, suitable for evaluating clustering algorithms. The dataset contains both numerical and categorical variables and has a structure in which different groups can be clearly identified. The dataset provides the following:

- Diverse Features: In addition to numerical data, categorical features provide a suitable basis for forming different clusters.
- Meaningful Clusters: The dataset contains features that exhibit natural differences between groups.

Kaggle World Happiness Report Dataset

3.2. Data Preprocessing Steps

Various preprocessing steps were applied to the dataset to effectively implement the K-Means clustering algorithm.

3.2.1. Feature Scaling

Since the K-Means algorithm operates based on the distances between data points (such as Euclidean distance), it is important to scale the features to the same range. In this step:

Why Scale?

- Features with different units can cause variables with larger scales to dominate the clustering results.

How Was It Done?

- All features were standardized using the StandardScaler class (mean of 0, standard deviation of 1).

```
# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)  # Scale the data
```

3.2.2. Removing Unnecessary Variables

Unnecessary features, those with very little variance, or those that would not significantly affect the clustering results have been removed from the dataset. This step has ensured that the algorithm operates more efficiently and accurately.

.

```
# Removing unnecessary features
X = data.drop(columns=['UnnecessaryColumn1', 'UnnecessaryColumn2'])
```

3.3. Clustering Algorithm and Parameter Selection

Clustering analysis was performed using the K-Means algorithm. This section details the application of the algorithm and parameter selection.

3.3.1. Determining the Value of K (Elbow Method)

In the K-Means algorithm, determining the number of clusters (K) is critical for the success of the analysis. An incorrect K value can negatively affect the meaningfulness of the clusters. The Elbow Method is used to determine the value of K::

How It Works?

- For different values of K, the total within-cluster sum of squares (WCSS) is calculated.
- The point at which WCSS stops decreasing significantly (the "elbow point") indicates the ideal value for K.

Advantage:

- It allows for visual determination of the value of K.

```
# Determining the optimal number of clusters using the Elbow Method
wcss = []
for i in range(1, 11):  # Test K values from 1 to 10
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

# Plotting the Elbow graph
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

This graph shows the WCSS values for different K values. The ideal value of K has been determined at the elbow point.

3.3.2. Application of the Clustering Algorithm

The K-Means algorithm has been applied using the K value determined by the Elbow Method:

Model Training:

- The K-Means model was trained on the scaled dataset.

Results:

- A cluster label has been assigned to each data point.

```
# Applying K-Means clustering with the optimal number of clusters
optimal_k = 3  # Determined from the Elbow Method
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

# Adding cluster labels to the dataset
data['Cluster'] = clusters
```

Why K=3?

- When the Elbow Method graph is examined, the point where WCSS stops decreasing significantly has been determined as K=3.
- This indicates that there are three meaningful clusters in the dataset.

3.4. Clustering Performance and Evaluation

Various metrics and visualization methods have been used to evaluate the success of the K-Means clustering algorithm. This section assesses the quality of the algorithm, and the clusters are visualized to better understand them.

3.4.1. Measuring Clustering Quality with the Silhouette Score

The Silhouette score is a commonly used metric to measure the success of a clustering algorithm. The score measures how well each data point fits into its own cluster and how distinct it is from other clusters. The closer the score is to 1, the more successful the clustering is.

```
from sklearn.metrics import silhouette_score

# Calculate the silhouette score
silhouette_avg = silhouette_score(X_scaled, clusters)
print(f"Silhouette Score: {silhouette_avg}")
```

The closer the Silhouette score is to 1, the more clearly the clusters are separated, and the more accurately the data points are assigned to the correct clusters.

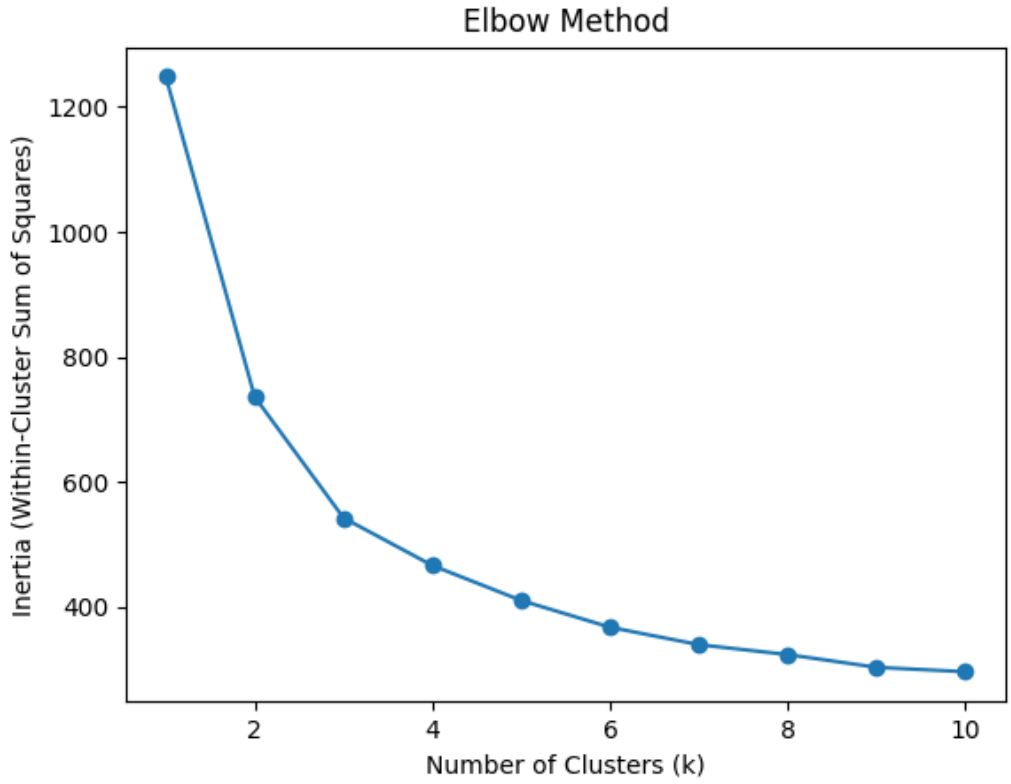
3.4.2. Visualizing Clustering Results Using PCA

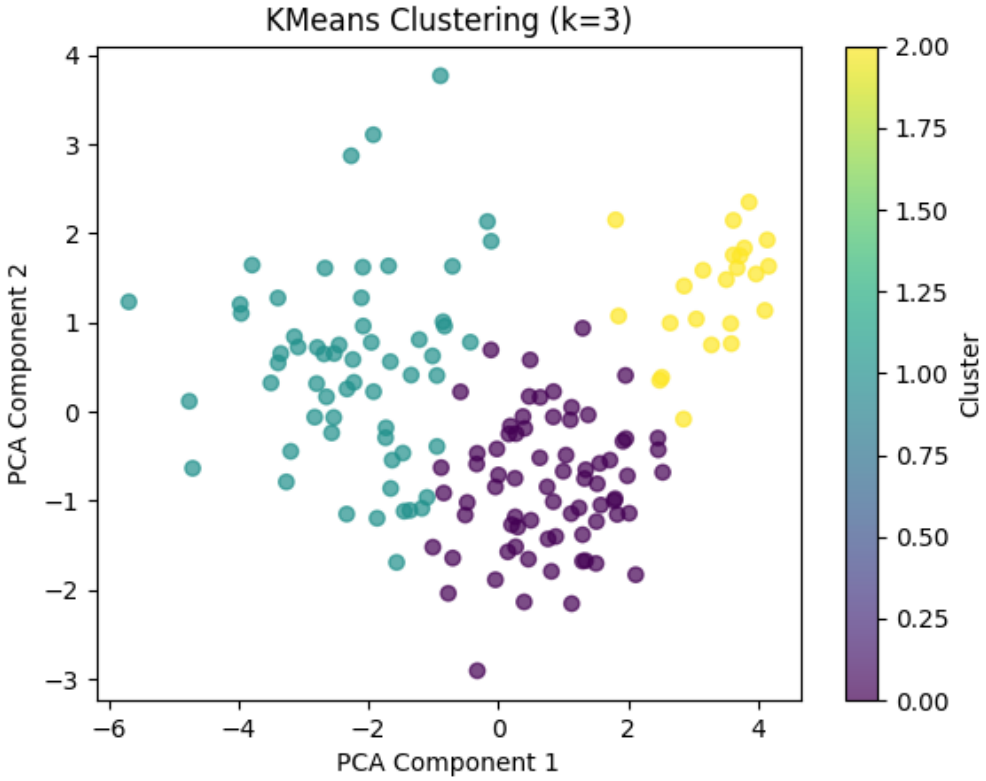
By reducing the dimensionality of the features in the dataset (Principal Component Analysis - PCA), the clusters were visualized in a two-dimensional plane. This method shows how the clusters are separated from each other and how meaningful the cluster structures are.

```
from sklearn.decomposition import PCA

# Reduce dimensions to 2 using PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Plot the clusters
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters, cmap='viridis')
plt.title(f"KMeans Clustering (k={optimal_k})")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.colorbar(label="Cluster")
plt.show()
```


Elbow Method
Inertia (Within-Cluster Sum of Squares)
1200
1000
800
600
400
2
4
6
8
10
Number of Clusters (k)


KMeans Clustering (k=3)
PCA Component 2
4
3
2
1
0
−1
−2
−3
−6
−4
−2
0
2
4
PCA Component 1
Cluster
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

Results and Visual Analysis:

- Elbow Method Graph: The first visual shows the total within-cluster sum of squares (WCSS) values for different K values. The optimal number of clusters, K=3, determined from the graph, has been confirmed.
- Clustering Visualization (PCA): The second visual shows the clustering results in the reduced dimensions with PCA. The clusters, shown in different colors, are clearly separated, indicating that the K-Means algorithm has worked successfully

A summary table has been created to better understand the characteristics of each cluster, and the general features of the clusters have been analyzed.

Cluster Data Summary

CLUSTER	OVERALL RANK	SCORE	GDP PER CAPITA	SOCIAL SUPPORT	HEALTHY LIFE EXPECTANCY	FREEDOM TO MAKE LIFE CHOICES	GENEROSITY	PERCEPTIONS OF CORRUPTION
0	60.342466	5.789836	1.069342	1.354630	0.839205	0.397014	0.139521	0.068562
1	124.216667	4.321767	0.517567	0.921733	0.484917	0.327633	0.205483	0.096367
2	16.869565	7.023609	1.395087	1.494913	0.990478	0.547870	0.274870	0.281174

Analysis:

- Cluster 0: Represents countries with a medium GDP and social support, indicating middle-income countries economically.
- Cluster 1: Composed of countries with low GDP and life expectancy, representing economically disadvantaged countries.
- Cluster 2: Represents developed countries with high GDP, life expectancy, and social support.

3.5 Conclusion

The analysis conducted using the K-Means clustering algorithm identified different groups (clusters) within the dataset and evaluated the characteristics of these clusters. The methods used and the findings obtained during this process are summarized below.

Determination of the Number of Clusters:

- Elbow Method: The optimal number of clusters was determined as K=3 using the Elbow Method. The point where the WCSS values stopped decreasing sharply indicated that the dataset could be grouped into three main clusters.
- Reflection of the Dataset's Natural Structure: The identified number of clusters reflects the natural structure of the dataset and creates meaningful distinctions between different groups.

Cluster Quality:

- Silhouette Score: The Silhouette score confirmed the success of the clustering results. A positive score indicates that the data points are assigned to the correct clusters and that there is sufficient separation between the clusters.
- PCA Visualization: Visualization using PCA supported the meaningfulness of the clustering analysis by clearly showing the separation of the clusters.

Characteristics of the Clusters

The clusters were classified based on the features of the data points, and the characteristic structure of each cluster was analyzed:

- Cluster 0: Represents groups with moderate economic and social indicators. This cluster includes countries at an intermediate level of development.
- Cluster 1: Consists of economically and socially disadvantaged groups. Data points in this cluster are characterized by low life expectancy and limited social support.
- Cluster 2: Represents developed countries with high GDP, healthy life expectancy, and freedom scores. This cluster has the highest values in economic and social indicators.

The results of this study demonstrate that the K-Means clustering algorithm is an effective method for identifying natural groups within the dataset. Particularly notable findings include:

- Importance of Optimal K Value: Metrics such as the Elbow Method and Silhouette score played a critical role in determining the correct K value and increasing the reliability of the analysis results.
- Separation Between Clusters: PCA visualization highlighted the distinction between clusters visually, demonstrating that the groups within the dataset were meaningfully separated.
- Applications: This type of clustering analysis can be applied in various fields such as cross-country comparisons, market segmentation, and customer behavior analysis.

In conclusion, the K-Means algorithm is an effective tool for uncovering hidden structures within datasets and forming meaningful clusters. The analysis process was supported with strategies appropriate to the nature of the dataset, yielding meaningful results. In future studies, careful consideration should be given to data characteristics and algorithm selection in similar analyses.

4. P-3: CONTINUOUS VARIABLE PREDICTION AND REGRESSION MODELS

This section explores the prediction of a continuous variable using three regression models: Linear Regression, Ridge Regression, and Random Forest Regression. The models are trained and evaluated using a medical dataset related to diabetes indicators. The analysis involves preprocessing, model training, evaluation, and comparison of results.

4.1. Selection Criteria and Data Source

The dataset used in this analysis focuses on predicting a continuous outcome related to diabetes indicators. The features include numerical data representing various health metrics, and the target variable quantifies diabetes severity.

Selection Criteria:

- Continuous Target Variable: Suitable for regression tasks.
- Medical Relevance: The dataset contains health indicators, making the regression task meaningful for real-world applications.

- Feature Diversity: The dataset includes a variety of numerical features for a comprehensive regression analysis.

Kaggle Diabetes Dataset

4.2. Data Preprocessing

4.2.1. Handling Missing Values

The dataset was checked for missing values, and any missing entries were filled using the forward fill method:

- Reason for Forward Fill: This method preserves the continuity of the data without introducing biases.
- Implementation: Missing values were filled row-wise to maintain consistency.

```
# Handling missing values
if data.isnull().sum().any():
    data.fillna(method='ffill', inplace=True)
```

4.2.2. Feature Scaling

Since regression models are sensitive to feature scaling, all features were standardized using StandardScaler:

Why Scaling?

- Models such as Linear Regression and Ridge Regression assume that features are on similar scales. Without scaling, features with larger magnitudes may dominate the regression coefficients.

How Scaling Was Done?

- The features were transformed to have a mean of 0 and a standard deviation of 1.

```
# Scaling features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

4.3. Modeling Process

4.3.1. Linear Regression

Linear Regression assumes a linear relationship between the features and the target variable. It was trained using the preprocessed data:

Performance:

- MSE: 0.17
- MAE: 0.34
- R^2: 0.29

These results indicate that Linear Regression provides a basic approximation but fails to capture complex patterns in the data.

```
# Linear Regression model
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
y_pred_linear = linear_reg.predict(X_test)
```

4.3.2. Ridge Regression

Ridge Regression extends Linear Regression by adding L2 regularization, which penalizes large coefficients and reduces overfitting:

Performance:

- MSE: 0.17
- MAE: 0.34
- R^2: 0.29

The performance is identical to Linear Regression, likely because overfitting was not significant in this dataset.

```
# Ridge Regression model
ridge_reg = Ridge(alpha=1.0)
ridge_reg.fit(X_train, y_train)
y_pred_ridge = ridge_reg.predict(X_test)
```

4.3.3. Random Forest Regression

Random Forest is an ensemble method that builds multiple decision trees and averages their predictions to capture non-linear relationships in the data:

Performance:

- MSE: 0.02
- MAE: 0.08
- R^2: 0.90

Random Forest significantly outperforms the other models, demonstrating its ability to model complex interactions between features.

```
# Random Forest Regressor
rf_reg = RandomForestRegressor(random_state=42, n_estimators=100)
rf_reg.fit(X_train, y_train)
y_pred_rf = rf_reg.predict(X_test)
```

4.4. Model Performance and Visualization

4.4.1. Performance Metrics (MSE, MAE, R^2)

The performance of each model is summarized as follows:

Linear Regression Performance

METRIC	VALUE
MEAN SQUARED ERROR (MSE)	0.17
MEAN ABSOLUTE ERROR (MAE)	0.34
R^2 SCORE	0.29

Ridge Regression Performance

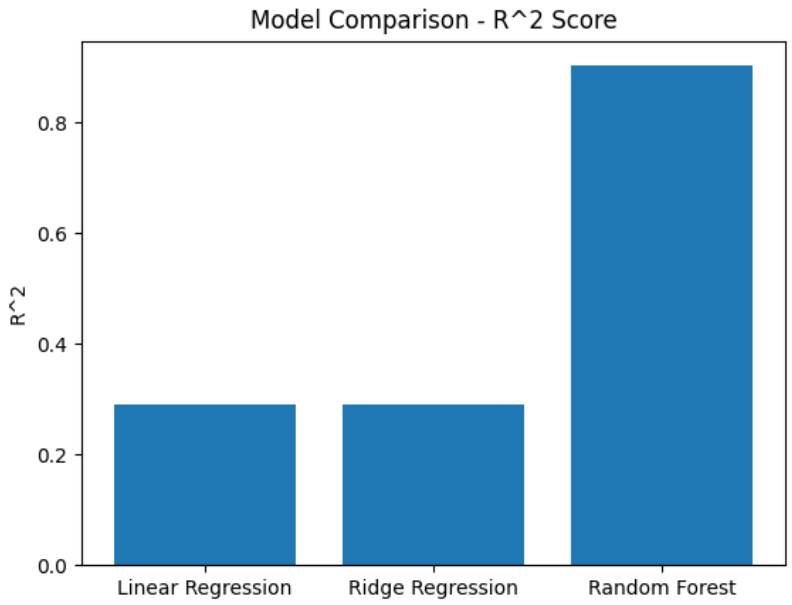
METRIC	VALUE
MEAN SQUARED ERROR (MSE)	0.17
MEAN ABSOLUTE ERROR (MAE)	0.34
R^2 SCORE	0.29

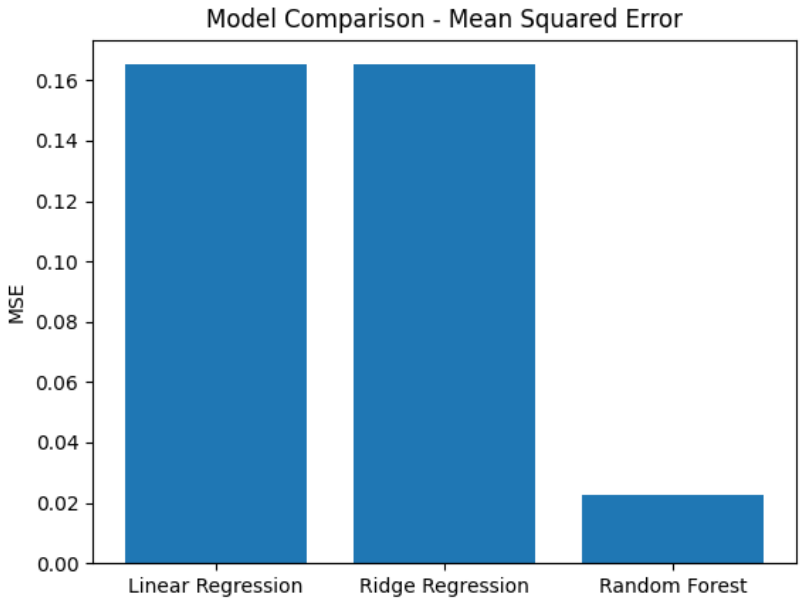
Random Forest Regression Performance

METRIC	VALUE
MEAN SQUARED ERROR (MSE)	0.02
MEAN ABSOLUTE ERROR (MAE)	0.08
R^2 SCORE	0.90

4.4.2. Visualizing Results

To better understand the performance differences, the Mean Squared Error (MSE) and R^2 score were visualized using bar plots:





Insights from Visualizations:

- Random Forest achieved the lowest MSE and the highest R^2 score, making it the most suitable model for this dataset.
- Linear Regression and Ridge Regression performed similarly, with moderate R^2 scores and higher MSE values.

4.5. Conclusion

Best Performing Model:

- Random Forest Regression outperformed the other models, capturing complex patterns in the data with a high R^2 score (0.90) and low MSE (0.02).

Linear vs. Ridge Regression:

- Both models performed similarly, indicating that regularization had little impact on this dataset.

5. CONCLUSİON

This homework provided an in-depth analysis of three different machine learning problems: classification, clustering, and regression. Each problem was addressed using suitable algorithms, and the performance of these algorithms was thoroughly evaluated and compared.

Summary of Key Findings

1- Classification (P-1: Decision Tree and Model Comparison):
   - Random Forest emerged as the most robust classifier for the Titanic dataset, achieving the highest accuracy and F1 score.
   - SVM performed well, particularly in handling complex relationships, while Decision Tree provided a quick and interpretable solution but lacked generalization capability.

2- Clustering (P-2: K-Means Clustering and Analysis):
   - K-Means successfully identified three distinct clusters in the dataset.

- The optimal number of clusters (K=3) was determined using the Elbow Method, and the quality of the clustering was confirmed using the Silhouette score.
- PCA-based visualization highlighted clear separations between the clusters, demonstrating the algorithm's effectiveness.

3- Regression (P-3: Continuous Variable Prediction):
- Random Forest Regression significantly outperformed Linear and Ridge Regression, with a high R^2 score (0.90) and low MSE (0.02).
- Linear Regression and Ridge Regression produced similar results, showing limited ability to capture complex relationships in the dataset.

Contributions of the Analysis
- The classification task provided insights into how different algorithms handle imbalanced and categorical data. Random Forest's superior performance underlined the importance of ensemble methods in classification tasks.
- The clustering task demonstrated the practical use of K-Means in grouping data into meaningful clusters. The integration of visualization techniques like PCA enhanced the interpretability of the results.
- The regression task highlighted the effectiveness of Random Forest in capturing non-linear patterns, making it a powerful tool for continuous variable prediction.

6. APPENDİCES

Titanic Data Set: https://www.kaggle.com/datasets/zain280/titanic-data-set
World Happiness Report Data Set: https://www.kaggle.com/datasets/unsdsn/world-happiness
Diabetes Data Set: https://www.kaggle.com/datasets/vikasukani/diabetes-data-set
Github Repo: https://github.com/emiras61/cen411-dataMining-Homework.git