RECEP TAYYIP ERDOGAN UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

COMPUTER ENGINEERING DEPARTMENT

Data Mining

2024-2025

Student Name Surname

Rıdvan Karasubaşı

Student No

201401030

Topic

Using and Analyzing Data Mining Methods

Data Sets Used

Heart Disease Dataset

Wholesale Customer Data

California Housing Database

Applied Methods:

Decision Tree, Random Forest and SVM

K-Means Clustering

Linear Regression and Random Forest Regression

Instructor

Doctor Lecturer Abdulgani Kahraman

RİZE

2024

1. Introduction

This report is about using and analyzing basic methods in data mining. Data mining means finding important information from large and complex sets of data. It is very important in today's world. Because there is a lot of data. This report looks at three main data mining methods and checks how they work on different data sets.

First, decision trees and random forest classification methods are used to predict a target variable that has categories. In this part, data was cleaned, the model was tested, and its performance was compared with additional classifiers such as support vector machines (SVM). In the second part cluster analysis was done using the K-Means method to find natural groups in the data. Lastly, linear regression and random forest regression methods were compared to predict a continuous target variable.

The types of data sets used, the details of the methods, the ways to measure results, and the outcomes are all discussed in the report. The results are shown with visual tools and compared with each other. This report aims to help understand data mining techniques in theory and practice.

2. Datasets Used

This report uses three different datasets. Each dataset is analyzed with different data mining methods. The datasets and their features are shown below:

2.1. Heart Disease Dataset

Source: https://archive.ics.uci.edu/dataset/45/heart+disease

Description: This dataset has health informations from people with and without heart disease. It includes details like age, type of chest pain, and cholesterol levels.

Why I Chose This Data Set: This dataset does not contain any missing values and this data set is easy to understand. We did not eliminate any feature but we updated the target column.

Target Variable(Outcome):

- Target:
  - 1: Heart disease
  - 0: No heart disease

Purpose: To predict the target varible using decision tree, random forest and SVM classification algorithms.

Details:

- **Total Observations:** 303
- **Number of Columns:** 13
- **Independent Variables:**
  - **age:** Age of the individual.
  - **sex:** Gender(1: Male, 0: Female).
  - **cp:** Type of chest pain experienced(typical angina, atypical angina, non anginal pain, asymptomatic).
  - **trestbps:** Resting blood pressure.
  - **chol:** Serum cholesterol level.
  - **fbs:** Fasting blood. If sugar is bigger than 120 than 1(Yes), else 0(No).
  - **restecg:** Resting electrocardiographic results.
  - **thalach:** Maximum heart rate achieved.
  - **exang:** Exercise-induced angina(1: Yes, 0: No).
  - **oldpeak:** ST depression induced by exercise relative to rest.
  - **slope:** Slope of the ST segment during exercise.
  - **ca:** Number of major vessels visible via fluoroscopy.
  - **thal:** Thalassemia type(3: Normal, 6: Fixed defect, 7: Reversible defect).

2.2. Wholesale Customer Data

Source: https://archive.ics.uci.edu/ml/datasets/Wholesale+customers

Description: This dataset provides segmentation data for wholesale customers, including annual spending values on categories such as milk, grocery, and frozen products.

Why I Chose This Data Set: This dataset does not contain any missing values and this data set is easy to understand. We eliminated channel and region columns because they are not necessary for clustering.

Target Variable: None(Clustering analysis does not include a target variable.)

Purpose: To perform customer segmentation using the K-Means algorithm.

Details:

- **Total Observations:** 440
- **Number of Columns:** 8
- **Independent Variables:**
    - **Fresh:** Annual spending on fresh products.
    - **Milk:** Annual spending on milk products.
    - **Grocery:** Annual spending on grocery products.
    - **Frozen:** Annual spending on frozen products.
    - **Detergents_Paper:** Annual spending on detergents and paper products.
    - **Delicassen:** Annual spending on delicatessen products.
    - **Region:** Geographic region (1: Lisbon, 2: Porto, 3: Other).
    - **Channel:** Type of channel (1: Hotel/Restaurant/Cafe, 2: Retail).

2.3. California Housing Database

Source: The dataset can be found in the scikit-learn library.

Description: This dataset contains features related to housing in California, such as income, number of rooms, and house age.

Why I Chose This Data Set: This dataset does not contain any missing values and this data set is easy to understand. We did not eliminate any feature.

Target Variable:

- **Price:** Median house value for California districts.

Purpose: To predict house prices using linear regression and advanced regression techniques.

Details:

- **Total Observations:** 20,640
- **Number of Columns:** 9
- **Independent Variables:**
    - **MedInc:** Median income in the block group.
    - **HouseAge:** Median house age in the block group.
    - **AveRooms:** Average number of rooms per household.
    - **AveBedrms:** Average number of bedrooms per household.
    - **Population:** Population in the block group.
    - **AveOccup:** Average number of occupants per household.

- **Latitude:** Block group latitude.
- **Longitude:** Block group longitude.

3. Question 1: Decision Tree and Comparative Analysis

3.1. Importing Necessary Libraries

The necessary libraries which I imported for this task are shown below:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

- **sklearn.model_selection.train_test_split:** This library used for splitting data to training and test data.
- **sklearn.tree.DecisionTreeClassifier:** Decision tree classifier algorithm.
- **sklearn.tree.plot_tree:** It visualize decision trees and it makes it easier to understand.
- **sklearn.ensemble.RandomForestClassifier:** Random forest classifier algorithm.
- **sklearn.svm.SVC:** Support vector classifier algorithm for additional comparison.
- **sklearn.metrics:** This library includes model evaluation metrics. The metrics are shown below:
  - **accuracy_score:** It measures the accuracy rate of the model.
  - **precision_score:** It measures the accuracy of the positive predictions.
  - **recall_score:** It measures the rate of the true positive.
  - **f1_score:** It calculates the harmonic mean of precision and recall.
  - **confusion_matrix:** It summarizes the difference between actual values and predicted values of the model.
- **seaborn:** This library used for data visualization.
- **matplotlib.pyplot:** This library offers charting tools to visualize the data.
- **pandas:** This library used for data processing and data analyse.

3.2. Data Preprocessing

3.2.1. Loading and Preparing the Data

I loaded my dataset named heart.csv and then assigned the data to the heart_data variable. Now heart_data is a dataframe and it includes the all data in the csv file. Then, I create a map for simplify the target value. 1's will be 1 and 2's will be 0.

```
data = 'data/heart.csv'
heart_data = pd.read_csv(data)
heart_data['target'] = heart_data['target'].map({1: 1, 2: 0})
```

3.2.2. Handling missing Values

The dataset does not have any missing values. This makes it easier to use for training and testing the models.

3.2.3. Feature Selection and Engineering

- The target column is the dependent variable. It has two classes:
    - 1: Patient has heart disease.
    - 0: Patient does not have heart disease.
- All other columns are independent variables used to train the models.
- No unnecessary features were removed. All features were included in the analysis.

```
X = heart_data.drop(columns=['target'])
y = heart_data['target']
```

3.3. Modeling

3.3.1. Splitting Data

- I divided the data two times. In the first I select 80% training data for the model to learn and 20% test data for evaluating the model performance. In the second I select 70% training data for the model to learn and 30% test data for evaluating the model's performance.
- The splitting was done using the train_test_split function with random_state=58 to ensure reproducibility.

First selection:

```
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=58)
```

Second selection:

```
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.3, random_state=58)
```

3.3.2. Decision Tree Classifier

- A decision tree was used to predict the target variable.
- The DecisionTreeClassifier from Python's sklearn library was applied to the dataset.
- The model was trained on the training dataset.

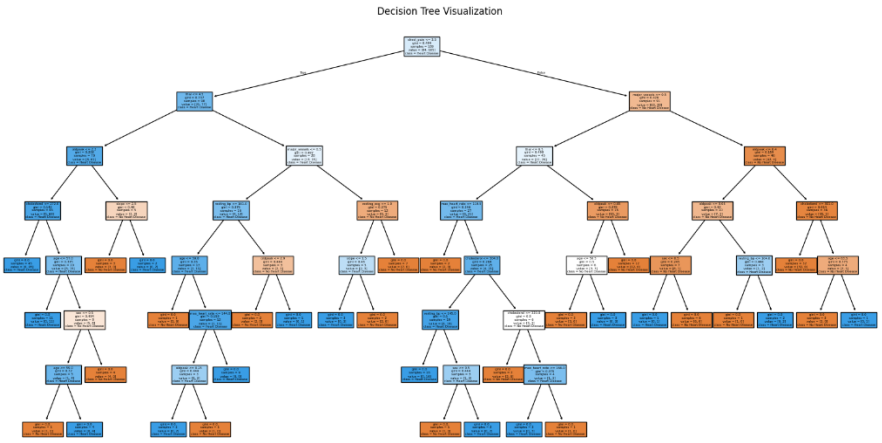
```
dt_model = DecisionTreeClassifier(random_state=58)
dt_model.fit(X_train, y_train)

y_pred_dt = dt_model.predict(X_test)

dt_accuracy = accuracy_score(y_test, y_pred_dt)
dt_precision = precision_score(y_test, y_pred_dt)
dt_recall = recall_score(y_test, y_pred_dt)
dt_f1 = f1_score(y_test, y_pred_dt)
```

Decision Tree Visualization:

In the code below, we visualized the Decision Tree. We setted the figure size with using plt.figure function and then we visualized the decision tree with using plot_tree function. We sent some parameters like feature_names and class_names. Lastly, we setted a plot title.

```
plt.figure(figsize=(20, 10))
plot_tree(dt_model, filled=True, feature_names=X.columns, class_names=["No Heart Disease", "Heart Disease"])
plt.title("Decision Tree Visualization")
plt.show()
```

Result:



3.3.3 Random Forest Classifier

- A random forest was used to predict the target variable.
- The RandomForestClassifier from Python's sklearn library was applied to the dataset.
- The model was trained on the training dataset.

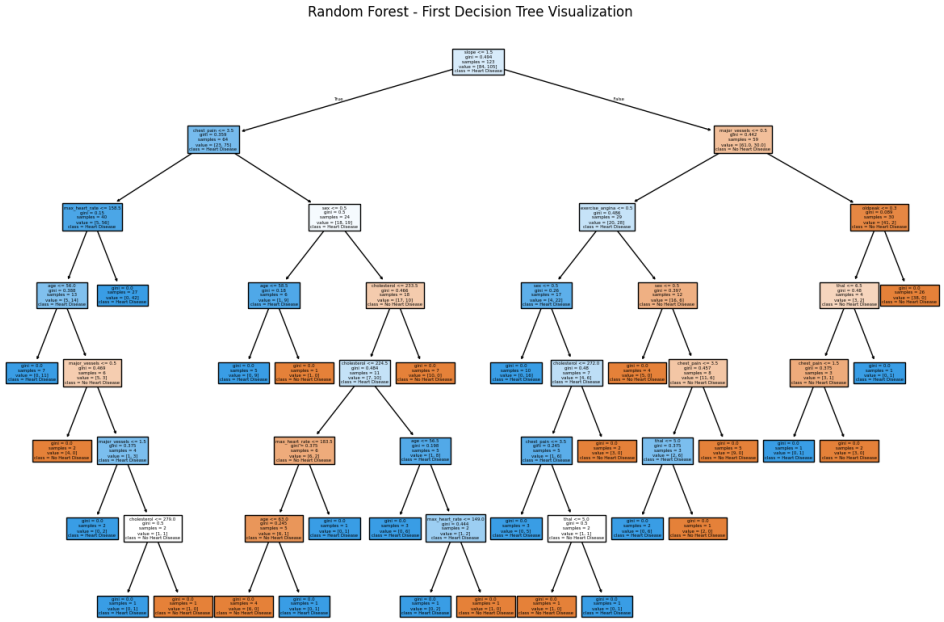
```
rf_model = RandomForestClassifier(random_state=58)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)

rf_accuracy = accuracy_score(y_test, y_pred_rf)
rf_precision = precision_score(y_test, y_pred_rf)
rf_recall = recall_score(y_test, y_pred_rf)
rf_f1 = f1_score(y_test, y_pred_rf)
```

Random Forest Visualization:

In the code below, we visualized the Random Forest's first decision tree. In total, 100 random forest decision trees were created. It was not possible to show all of them so I show only the first tree. We setted the figure size with using plt.figure function and then we visualized the random forest's first decision tree with using plot_tree function. We sent some parameters like feature_names and class_names. Lastly, we setted a plot title.

```
plt.figure(figsize=(15, 10))
plot_tree(first_tree, feature_names=X.columns, class_names=["No Heart Disease", "Heart Disease"], filled=True)
plt.title("Random Forest - First Decision Tree Visualization")
plt.show()
```

Result:



3.3.4. Support Vector Machine Classifier(SVM)

An additional comparison was done using a support vector machine classifier. The SVM algorithm was trained on the same dataset.

- Support Vector Machine (SVM) was used to predict the target variable.
- The SVC model from Python's sklearn library was applied to the dataset.
- The model was trained using the training data.

```
svm_model = SVC(random_state=58)

svm_model.fit(X_train, y_train)

y_pred_svm = svm_model.predict(X_test)

svm_accuracy = accuracy_score(y_test, y_pred_svm)
svm_precision = precision_score(y_test, y_pred_svm)
svm_recall = recall_score(y_test, y_pred_svm)
svm_f1 = f1_score(y_test, y_pred_svm)
```

3.4. Performance Evaluation

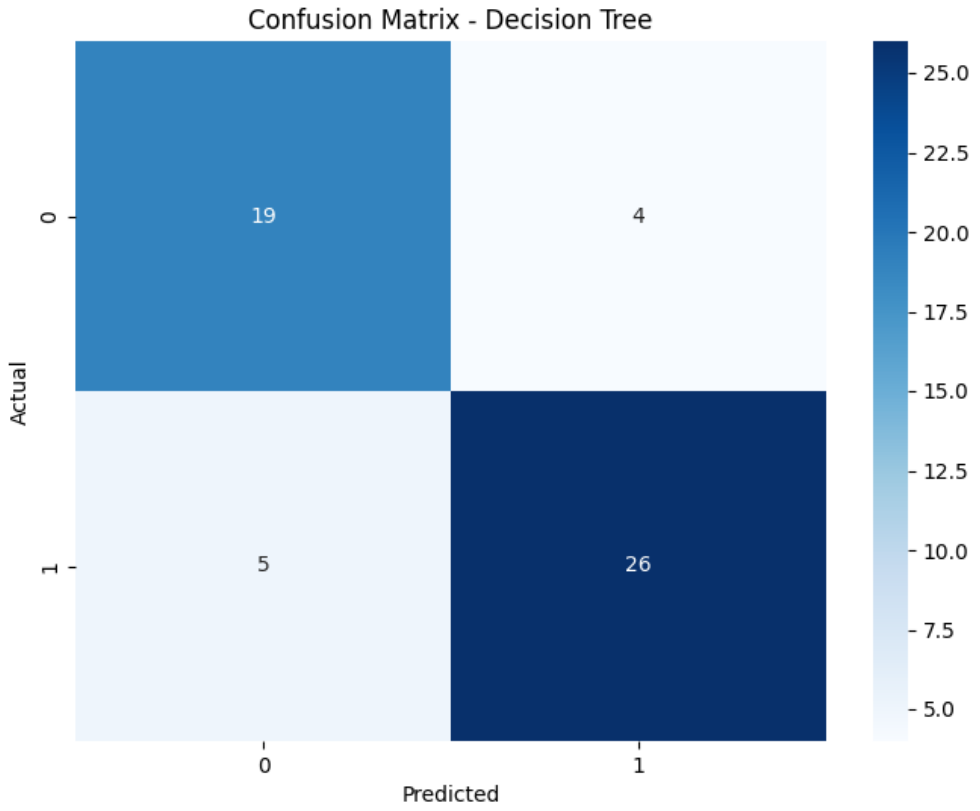
3.4.1. Decision Tree Performance

The performance of two decision tree models was evaluated using accuracy, precision, recall, and F1 score for two selection. For first selection results are shown below:

```
           Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree  0.833333   0.866667  0.838710  0.852459
1  Random Forest  0.944444   0.937500  0.967742  0.952381
2            SVM  0.722222   0.690476  0.935484  0.794521
```

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.83	0.86	0.83	0.85
Random Forest	0.94	0.93	0.96	0.95
SVM	0.72	0.69	0.93	0.79

Confusion Matrix for Decision Tree:



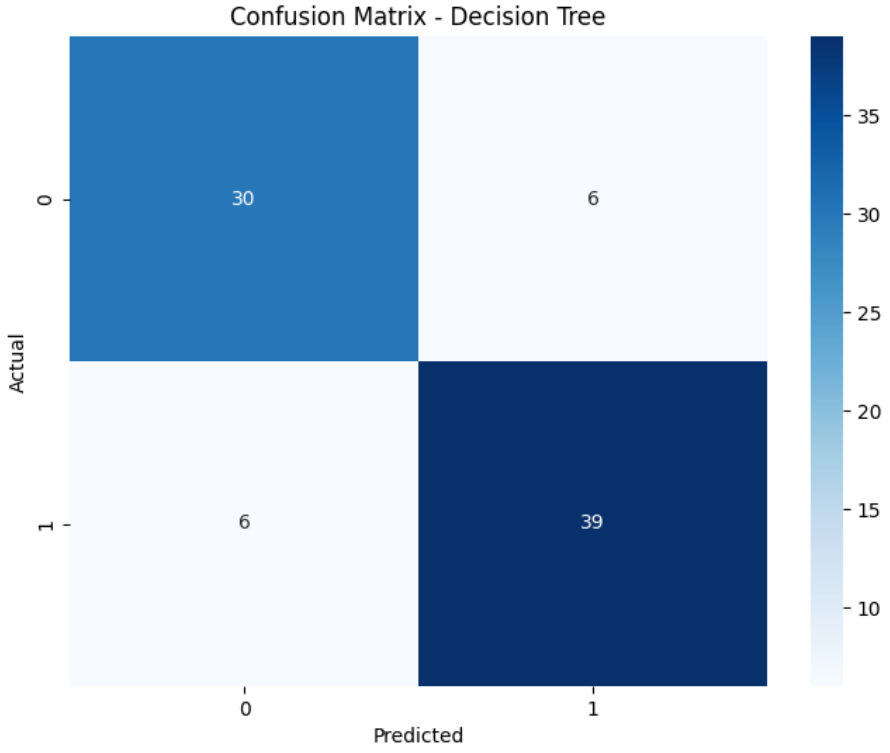
- **True Positives (Correctly predicted heart disease):** 26
- **True Negatives (Correctly predicted no heart disease):** 19
- **False Positives (Wrongly predicted heart disease):** 4
- **False Negatives (Wrongly predicted no heart disease):** 5

For second selection results are shown below:

```
           Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree  0.851852   0.866667  0.866667  0.866667
1  Random Forest  0.938272   0.916667  0.977778  0.946237
2            SVM  0.716049   0.671875  0.955556  0.788991
```

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.85	0.86	0.86	0.86
Random Forest	0.93	0.91	0.97	0.94
SVM	0.71	0.67	0.95	0.78

Confusion Matrix for Decision Tree:



- **True Positives:** 39
- **True Negatives:** 30
- **False Positives:** 6
- **False Negatives:** 6

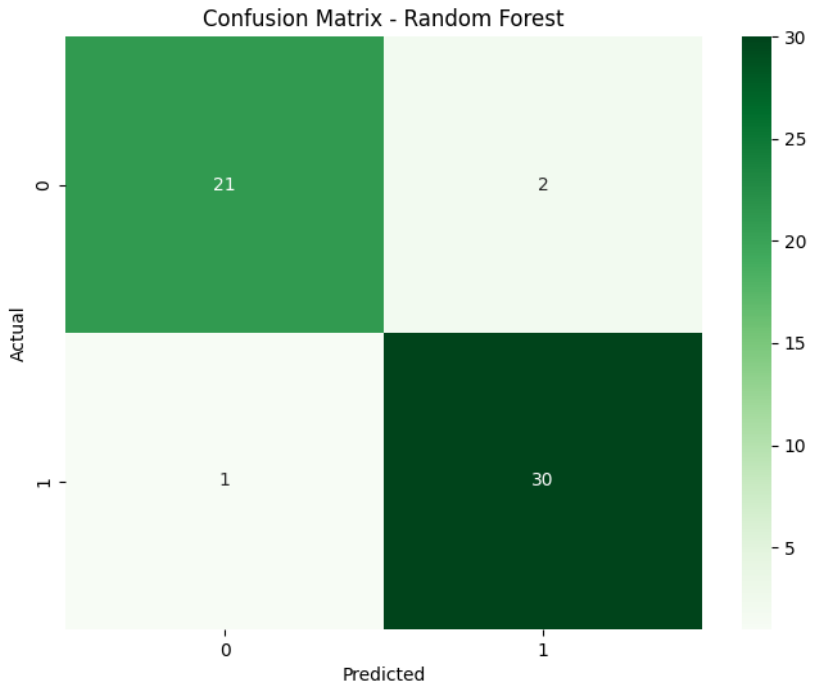
3.4.2. Random Forest Performance

A random forest classifier was used to compare with the decision tree model. For the first selection performance metrics are:

```
           Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree  0.833333   0.866667  0.838710  0.852459
1  Random Forest  0.944444   0.937500  0.967742  0.952381
2            SVM  0.722222   0.690476  0.935484  0.794521
```

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.83	0.86	0.83	0.85
Random Forest	0.94	0.93	0.96	0.95
SVM	0.72	0.69	0.93	0.79

Confusion Matrix for Random Forest:



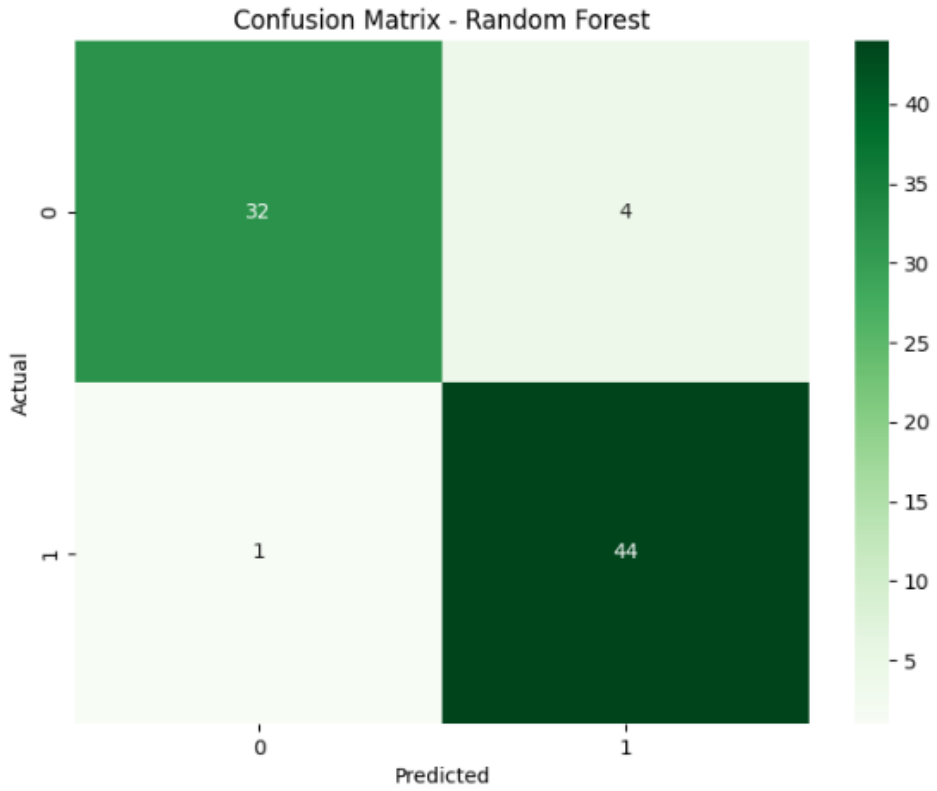
- **True Positives:** 30
- **True Negatives:** 21
- **False Positives:** 2
- **False Negatives:** 1

For the second selection performance metrics are:

```
           Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree  0.851852   0.866667  0.866667  0.866667
1  Random Forest  0.938272   0.916667  0.977778  0.946237
2            SVM  0.716049   0.671875  0.955556  0.788991
```

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.85	0.86	0.86	0.86
Random Forest	0.93	0.91	0.97	0.94
SVM	0.71	0.67	0.95	0.78

Confusion Matrix for Random Forest:



- **True Positives:** 44
- **True Negatives:** 32
- **False Positives:** 4
- **False Negatives:** 1

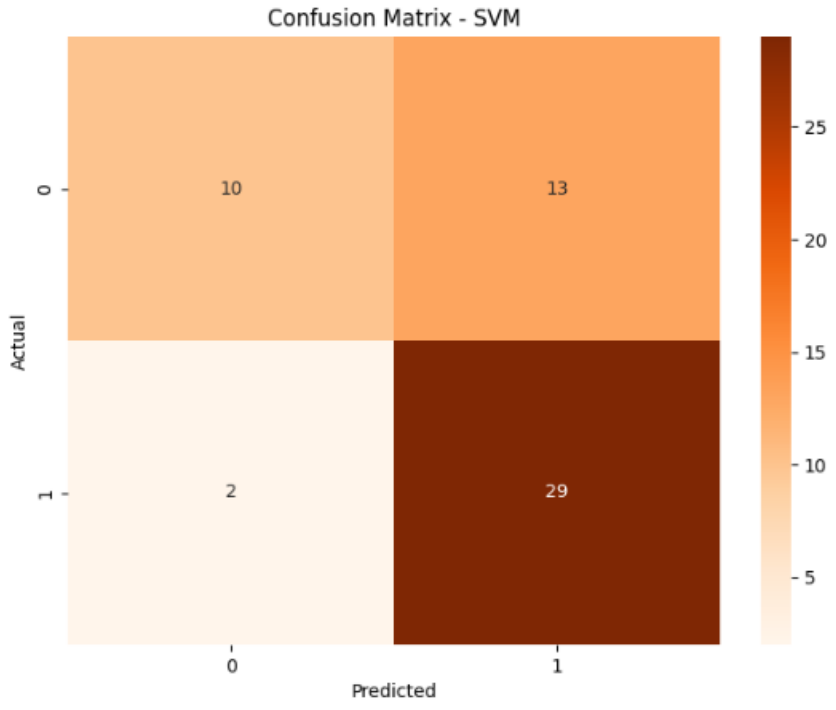
3.4.3. SVM Performance

The performance of two SVM model was evaluated using accuracy, precision, recall, and F1 score for two selection. For first selection results are shown below:

```
           Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree  0.833333   0.866667  0.838710  0.852459
1  Random Forest  0.944444   0.937500  0.967742  0.952381
2            SVM  0.722222   0.690476  0.935484  0.794521
```

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.83	0.86	0.83	0.85
Random Forest	0.94	0.93	0.96	0.95
SVM	0.72	0.69	0.93	0.79

Confusion Matrix for SVM:



- **True Positives:** 19
- **True Negatives:** 10
- **False Positives:** 13
- **False Negatives:** 2

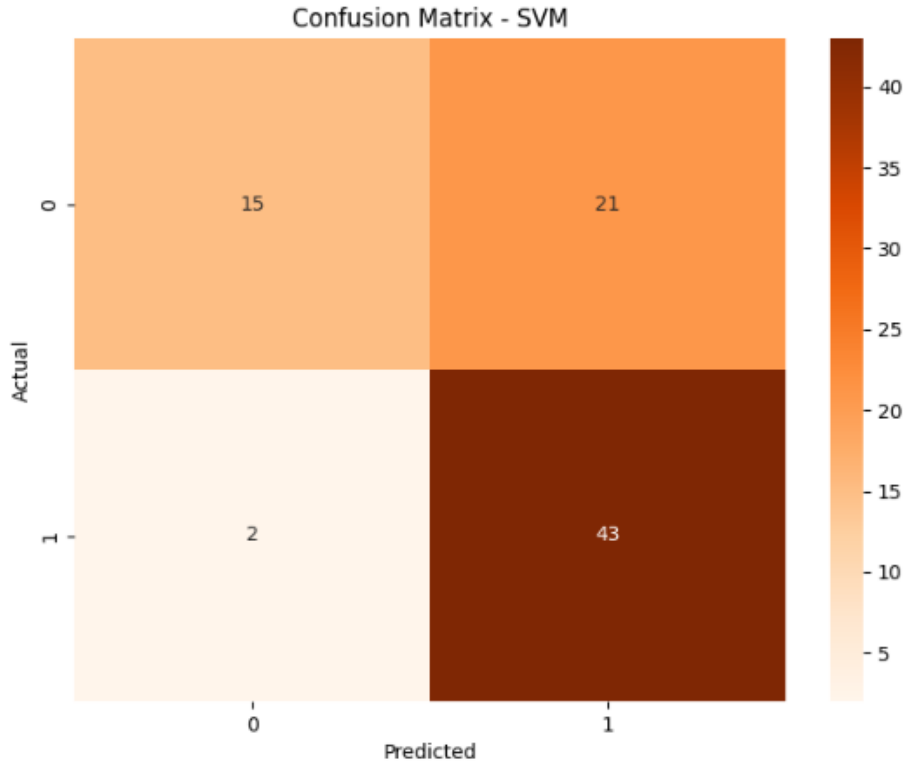
For second selection results are shown below:

```
           Model  Accuracy  Precision    Recall  F1 Score
0  Decision Tree  0.851852   0.866667  0.866667  0.866667
1  Random Forest  0.938272   0.916667  0.977778  0.946237
2            SVM  0.716049   0.671875  0.955556  0.788991
```

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.85	0.86	0.86	0.86
Random Forest	0.93	0.91	0.97	0.94
SVM	0.71	0.67	0.95	0.78

As you can see, the performances of the SVM models is lower than other models.

Confusion Matrix for SVM:



- **True Positives:** 43
- **True Negatives:** 15
- **False Positives:** 21
- **False Negatives:** 2

3.5. Results and Discussion

3.5.1. Comparison of Model Performance

The performance of decision tree, random forest, and SVM models was evaluated using accuracy, precision, recall, and F1 score metrics. Based on the results, random forest outperformed the decision tree and SVM in both train-test splits. Below is a detailed comparison of the models:

Accuracy: Random forest achieved the highest accuracy across both data splits. Its accuracy was significantly higher than decision tree and SVM, indicating that random forest generalizes better and is more reliable for unseen data.

Precision: Precision measures how well the model avoids false positives. Random forest consistently showed the highest precision in both splits. This means random forest is better at reducing false alarms, which is critical in applications like fraud detection or medical diagnoses.

Recall: Recall measures the model's ability to identify true positive cases. Random forest's recall was higher than decision tree and SVM in both splits. This makes random forest a more suitable choice for tasks where identifying true positives is important, such as detecting diseases or other safety-critical issues.

F1 Score: The F1 score, which balances precision and recall, was highest for random forest in both splits. A higher F1 score indicates that random forest maintains a good balance between avoiding false positives and identifying true positives.

Confusion Matrix: The confusion matrix analysis shows that random forest has less false positives and false negatives compared to decision tree and SVM. This contributes to its high overall performance, making it a more reliable model for classification tasks.

Overall Evaluating:

In general, random forest performed significantly better than both decision tree and SVM. Its ability to use multiple decision trees through ensemble learning provides better stability and

more accurate predictions. This makes random forest a better option for applications that require high reliability, such as predicting heart disease, detecting fraud, or other critical classification tasks.

3.5.2. Why Random Forest Performs Better

- **Ensemble Learning:** Random forest combines predictions from multiple decision trees, which reduces errors caused by individual trees. This ensemble approach increases accuracy and makes the model more strong.
- **Lower Overfitting:** Unlike a single decision tree, random forest reduces overfitting by averaging the predictions of multiple trees. This ensures that the model does not memorize the training data but instead learns general patterns.
- **Better Generalization:** Random forest performs well on unseen test data. Its ability to generalize makes it a more reliable choice for real-world applications, where test data may differ significantly from training data.

4. Question 2: K-Means Clustering

4.1 Importing Necessary Libraries

The necessary libraries which I imported for this task are shown below:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
```

- **pandas:** This library used for data processing and data analyse.
- **sklearn.cluster.KMeans:** This library provides the K-Means algorithm.
- **sklearn.preprocessing.StandardScaler:** This library used for scaling the datas.
- **sklearn.metrics.silhouette_score:** This library used for evaluation the clustering algorithms.
- **matplotlib.pyplot:** This library offers charting tools to visualize data.
- **seaborn:** This library used for data visualization.

4.2. Data Preprocessing

4.2.1 Loading and Preparing the Data

I loaded my dataset named wholesale_customers_data.csv. I dropped the Channel and Regions colums because they are not necessary for clustering.

```
data = pd.read_csv("data/wholesale_customers_data.csv")

if 'Channel' in data.columns and 'Region' in data.columns:
    data.drop( labels: ['Channel', 'Region'], axis=1, inplace=True)
```

4.2.2. Scaling and Missing Values

- The wholesale customers dataset was used for clustering.
- The Channel and Region columns were removed because they are not need for clustering.
- The dataset has no missing values.
- The features were scaled using StandardScaler. This helps the clustering algorithm by putting all features on the same scale. I create a scaler for scaling the features.
- It scales the data using fit_transform function.

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
```

4.3. Clustering Process

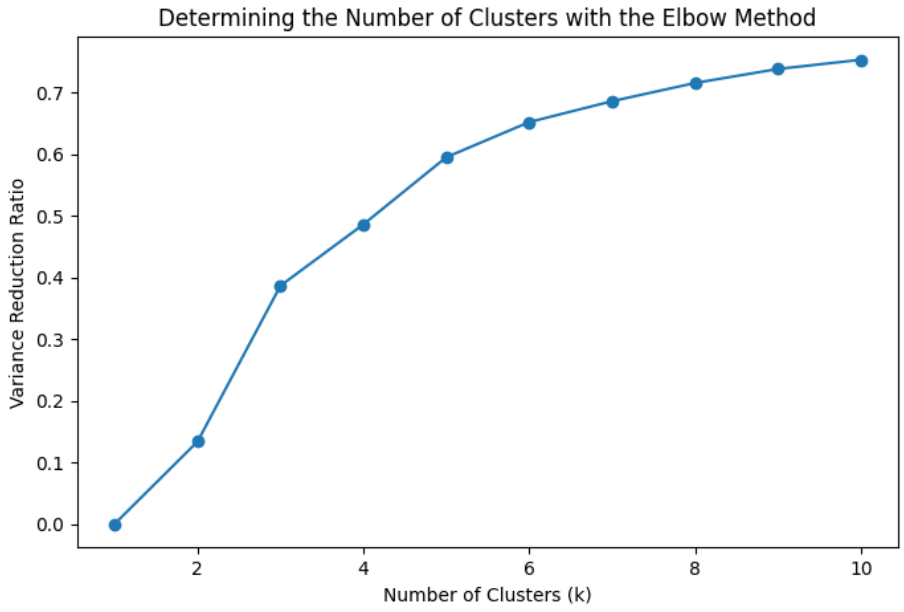
4.3.1. Choosing the Number of Clusters(k)

- The elbow method was used to find the best number of clusters.
- The elbow graph shows how much the variance changes for different k values(1 to 10).
- The “elbow point” shows the best number of clusters. The result is k = 3. As you see in the elbow graph, there is a big break at the point where k is 3. So, the optimal k value is 3. We can select the k as 3.

I created a empty list named internia. This list is used to store the internia values (total error sum of squares) obtained as a result of each clustering. It runs the K-Means algorithm for different cluster numbers and calculates the inertia values. Then, it assigns this value to the internia list. We will use these values later when drawing the elbow graph.

```
inertia = []
k_range = range(1, 11)
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=58)
    kmeans.fit(scaled_data)
    inertia.append(kmeans.inertia_)
```

Elbow Method Graph:



This code calculates the variance ratio using inertia values. The variance ratio indicates how well each clustering solution explains the data.

```
variance_ratio = [1 - (inertia[i] / inertia[0]) for i in range(len(inertia))]
```

You can see the variance values in the graph which shown below. Variance ratio for k=2 is 0.13 and variance ratio for k=3 is 0.38. Almost 3 times. So we can say k=3 is elbow.

```
Variance ratio for k=1: 0.0
Variance ratio for k=2: 0.13355220593411976
Variance ratio for k=3: 0.3863815219043727
Variance ratio for k=4: 0.4858412904081828
Variance ratio for k=5: 0.5948026974363619
Variance ratio for k=6: 0.6517804891616926
Variance ratio for k=7: 0.6856891864834975
Variance ratio for k=8: 0.7152255809430644
Variance ratio for k=9: 0.7379439620388433
Variance ratio for k=10: 0.7529122916105695
```

The optimal number of clusters determined as a result of the previous analysis was calculated as 3. A model created for K-Means algorithm and then K-Means algorithm runned and it separates data to the clusters. Finally, it creates a column named Cluster and it adds cluster values to the Clusters column.

```
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=58)
clusters = kmeans.fit_predict(scaled_data)

data['Cluster'] = clusters
```

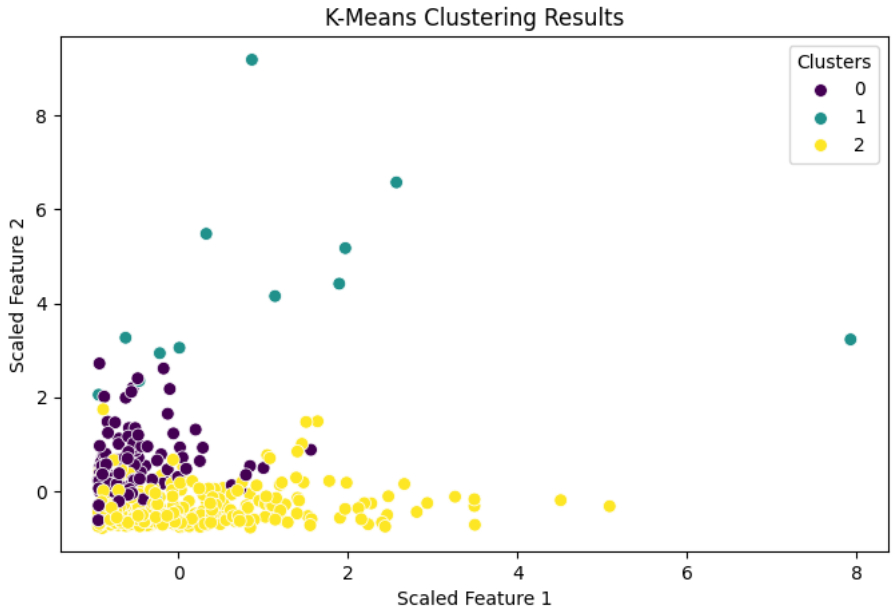
It calculates the silhouette score. Silhouette score will show how well separate the clusters.

```
silhouette_avg = silhouette_score(scaled_data, clusters)
print(f"\nSilhouette Score for k={optimal_k}: {silhouette_avg}")
```

4.3.2. Cluster Results

- The K-Means algorithm was used with k = 3 clusters.
- A scatter plot shows the clusters.
- Each dot is a customer, and each color is a different cluster.

Cluster Results Scatter Plot:



4.4. Performance Evaluation

- **Silhouette Score:** The silhouette score checks how good the clusters are. A higher score means better clustering.

```
Silhouette Score for k=3: 0.33391714199926514
```

4.5. Results and Discussion

4.5.1. Understanding the Clusters

- Cluster 0: Customer who spend a lot on categories like fresh or frozen products.
- Cluster 1: Customers who spend a medium amount on all categories.
- Cluster 2: Customers who spend less in most categories.

4.5.2. Analysis

- The K-Means algorithm worked well with this dataset.
- The elbow method and silhouette score show that k = 3 is the best choice.
- Businesses can user these clusters to understand customer spending behavior and improve their services.

Conclusion

K-Means clustering method was used to group customers into three main clusters based on their spending habits. The number of clusters was determined using the elbow method, which showed that three clusters would give the best results. Each cluster represents a different type of customer: high spenders, medium spenders, and low spenders. These groups help businesses to better understand their customers and make more informed decisions.

The elbow method helped identify the most suitable number of clusters by showing a significant drop in variance when the cluster number reached three. After determining this, the K-Means algorithm was applied, and the data was divided into these groups. Each cluster was visualized with a scatter plot, making it easy to see how customers were grouped based on their spending.

The analysis also included a silhouette score, which showed that the clustering worked well. The results are important for businesses because they can use this information to offer better services, target specific groups of customers, and create strategies for each cluster.

5. Question 3: Regression Analysis

5.1. Importing Necessary Libraries

The necessary libraries which I imported for this task are shown below:

```
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
```

- **pandas:** This library used for data processing and data analyse.
- **sklearn.datasets.fetch_california_housing:** This library loads the California Housing dataset.
- **sklearn.model_selection.train_test_split:** This library used for splitting data to training and test data.

- **sklearn.ensemble.RandomForestRegressor:** This library solves the regression problem using the random forest algorithm.
- **sklearn.linear_model.LinearRegression:** This library provides the linear regression model.
- **sklearn.metrics:** It provides some metrics for measuring of the model's performance. The metrics are shown below:
    - **mean_squared_error(MSE):** Calculates the mean squared error. Smaller values indicate a better model.
    - **mean_absolute_error(MAE):** Calculates the average of the absolute difference between predicted and actual values.
    - **r2_score:** It measures the explanatory power of the model. 1 indicates the ideal model, 0 indicates the average model**.**
- **sklearn.preprocessing.StandardScaler:** This library used for scaling the datas.
- **matplotlib.pyplot:** This library offers charting tools to visualize data.
- **seaborn:** This library used for data visualization.

5.2. Data Preprocessing

5.2.1. Loading Dataset and Preparing the Target Value

- The California housing dataset was used.
- Dataset loaded from sklearn datasets library.
- I create x and y variables. X will store our data as a dataframe and y variable, will store the target variable. The target variable is Price, which represents the median house price in California.

```
data = fetch_california_housing()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target, name="Price")
```

5.2.2. Feature Scaling and Missing Values

- All features were scaled using StandardScaler to make their values comparable.
- There were no missing values in the dataset.
- The features were scaled using StandardScaler. This helps the clustering algorithm by putting all features on the same scale. I create a scaler for scaling the features.
- It scales the x using fit_transform function.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

5.3. Modeling

5.3.1. Splitting Data

- I divided the data two times. In the first I select 80% training data for the model to learn and 20% test data for evaluating the model performance. In the second I select 70% training data for the model to learn and 30% test data for evaluating the model's performance.
- The splitting was done using the train_test_split function with random_state=58 to ensure reproducibility.

First selection:

```
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=58)
```

Second selection:

```
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.3, random_state=58)
```

5.3.2. Lineer Regression

- A lineer regression model was trained on the dataset.
- It uses a straight line to predict house prices based on input features.

```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

y_pred_lr = lr_model.predict(X_test)
print("\nLinear Regression Performance:")
print("MSE:", mean_squared_error(y_test, y_pred_lr))
print("MAE:", mean_absolute_error(y_test, y_pred_lr))
print("R² Score:", r2_score(y_test, y_pred_lr))
```

The linear regression's performance for first selection is shown below:

```
Linear Regression Performance:
MSE: 0.5341302729229414
MAE: 0.532945019389161
R² Score: 0.6057369914736299
```

Model	MSE	MAE	R² Score
Linear Regression	0.53	0.53	0.60

The linear regression's performance for second selection is shown below:

```
Linear Regression Performance:
MSE: 0.5332605094491446
MAE: 0.5325463535548731
R² Score: 0.6060131778750415
```

Model	MSE	MAE	R² Score
Linear Regression	0.53	0.53	0.60

5.3.3. Random Forest Regression

- A random forest regression model was used for comparison.
- Random forest is an advanced technique that uses multiple decision trees to make predictions.

```
rf_model = RandomForestRegressor(random_state=58, n_estimators=100)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)
print("\nRandom Forest Performance:")
print("MSE:", mean_squared_error(y_test, y_pred_rf))
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("R² Score:", r2_score(y_test, y_pred_rf))
```

The random forest's performance for first selection is shown below:

```
Random Forest Performance:
MSE: 0.25671296366751806
MAE: 0.33019635804263586
R² Score: 0.8105098502853849
```

Model	MSE	MAE	R² Score
Random Forest	0.25	0.33	0.81

The random forest's performance for second selection is shown below:

```
Random Forest Performance:
MSE: 0.2643619754114066
MAE: 0.33517376976744206
R² Score: 0.8046824530648102
```

Model	MSE	MAE	R² Score
Random Forest	0.26	0.33	0.80

5.4. Performance Evaluation

The models were evaluated using three metrics:

- Mean Squared Error(MSE)
- Mean Absolute Error(MAE)
- R^2 Score

Linear Regression Comparisons:

Error Metrics(MSE and MAE): In 70% train set selected model, MSE and MAE values are less than the other model. It shows predictions errors are less than the model which 80% train set selected.

R^2 Score: The R^2 score of the model which 70% train set selected is bigger than the other model's R^2 score. It means the model which 70% train set selected is more successful than the other model.

Conclusion

The model trained with 30% test data performance is better than the other model.

Random Forest Comparisons:

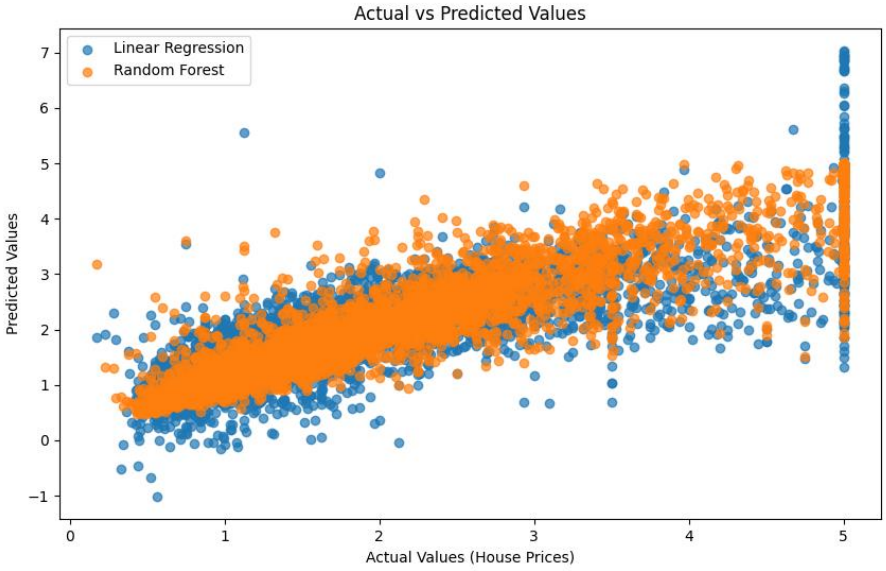
Error Metrics(MSE and MAE): In 80% train set selected model, MSE and MAE values are less than the other model. It shows predictions errors are less than the other model.

R^2 Score: The R^2 score of the model which 80% train set selected is bigger than the other model's R^2 score. It means the model which 80% train set selected is more successful than the other model.

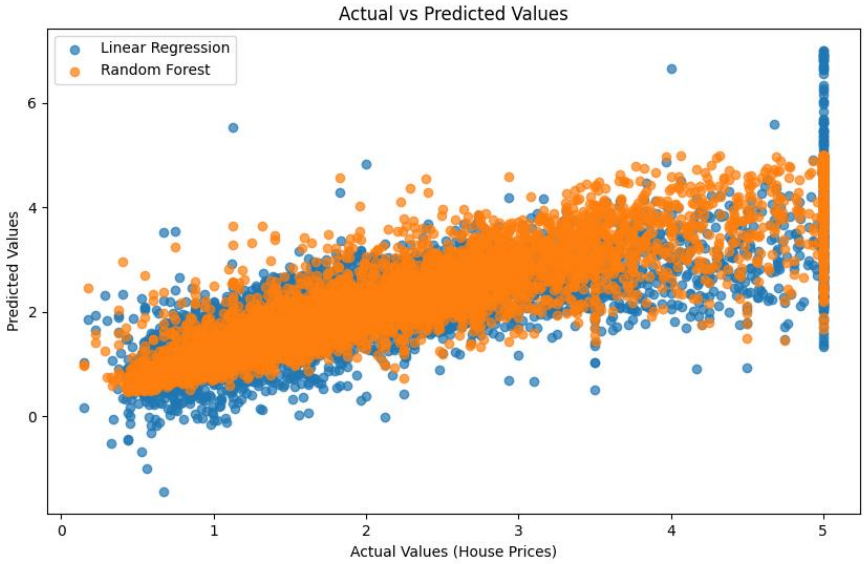
Conclusion

The model trained with 20% test data performance is better than the other model. This means that the model makes less errors and it explains the dependent variable better.

Actual vs Predicted Values Scatter Plot for First Selection:



Actual vs Predicted Values Scatter Plot for Second Selection:



- The random forest model performs better than linear regression on all metrics.
- Its lower MSE and MAE values indicate higher accuracy.

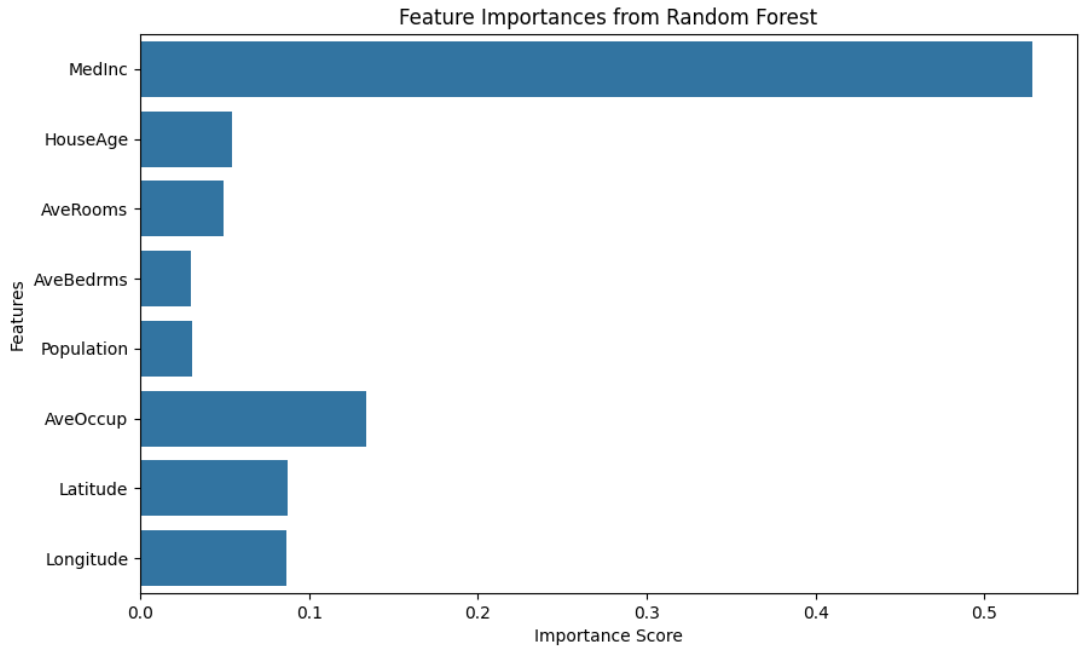
5.5 Results and Discussion

5.5.1. Importance of Features

The random forest model shows the importance of each feature in predicting house prices. Importances are the same in both models. The importance score is shown in the table below:

Feature	Importance Score
MedInc	0.52
AveOccup	0.05
Latitude	0.04
Longitude	0.02
HouseAge	0.03
AveRooms	0.13
AveBedrms	0.09
Population	0.08

Feature Importance Bar Chart for Two Selections:



5.5.2. Model Comparison

- **Lineer Regression:** Linear regression is easy to use and understand. It works well for simple data. But it does not work well with complex data. It is less accurate when the data is not linear.
- **Random Forest:** Random forest uses many decision trees to make better predictions. It works well with complex data and gives more accurate results than linear regression. It also shows which features are important in the data.

Conclusion

The random forest model is better for this task. It gives more accurate predictions and works well with complex data. It is a good choice for predicting house prices. It also helps to understand which features are important.

6. Conclusion

6.1. Summary

In this project, we used three main methods to work with data: classification, clustering, and regression. These methods helped us learn useful things from three different datasets.

The main points are shown below:

- **Classification**: We used decision tree and random forest models to predict if a person has heart disease. Random forest was better because it gave more accurate results and less mistakes.
- **Clustering**: The K-Means method grouped customers into three groups based on how much they spent on products. We used the elbow method to decide that three groups were the best choice.
- **Regression**: We predicted house prices using linear regression and random forest models. Random forest worked better because it understood the data better and made less errors.

6.2. Analysis

Each method worked well for its purpose. Here is why they are important:

- **Classification**: The random forest model was very good for predicting heart disease. It is reliable and can be used in medical studies to help doctors make better decisions.
- **Clustering**: The K-Means method can help businesses understand their customers. For example, they can see which customers spend a lot, which spend less, and then make better plans for these groups.
- **Regression**: The random forest model is strong and works well for complicated data like house prices. It can capture patterns that are not straight lines, making it better than simple models like linear regression.

6.3. Recommendations

Based on what we learned, here are some recommendations:

1. Use random forest when accuracy is very important, like in health or housing data.
2. Use clustering methods like K-Means to group similar data. This can help businesses understand their customers and improve their services.
3. Always check your data carefully before using any method. Make sure there are no missing values and scale the data if needed. Clean data gives better results.
4. Compare different methods to find the one that works best for your problem. For example, random forest was better than decision tree and linear regression in this project.

6.4 Final Thoughts

Data analysis is a powerful tool that helps us solve real-life problems. By finding patterns in data, we can make smarter decisions in fields like healthcare, business, and housing markets.

The methods we used in this project can help in many areas:

- In healthcare, they can predict diseases and improve patient care.
- In business, they can help companies understand customer behavior and make better plans.
- In housing, they can help people and companies predict house prices and make better investments.

Using the right method for the right data is very important. Random forest worked the best in this project, showing that advanced methods can give better results. With these tools, we can turn data into useful information and make better decisions in the future.