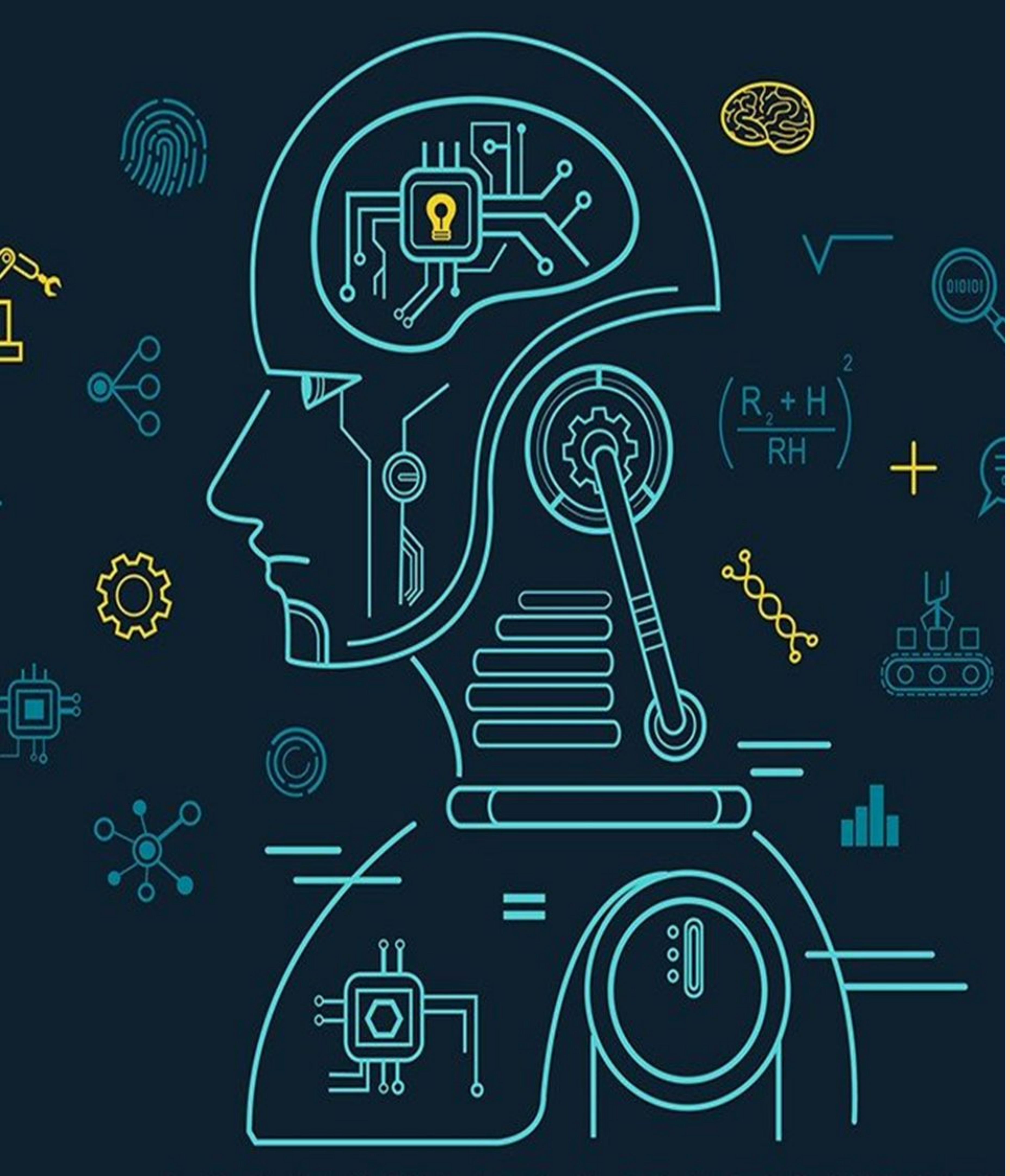


SCM 516 - Final Project

Team 401 - Divyansh Shrivastava, Lucas Smith, Kavya Murugan, Sami Fahim, Sravani Bolla

Predictive Modeling Techniques →

Agenda

- Introduction
- Classification
  - Analysis
  - Insights
- Regression
  - Analysis
  - Insights

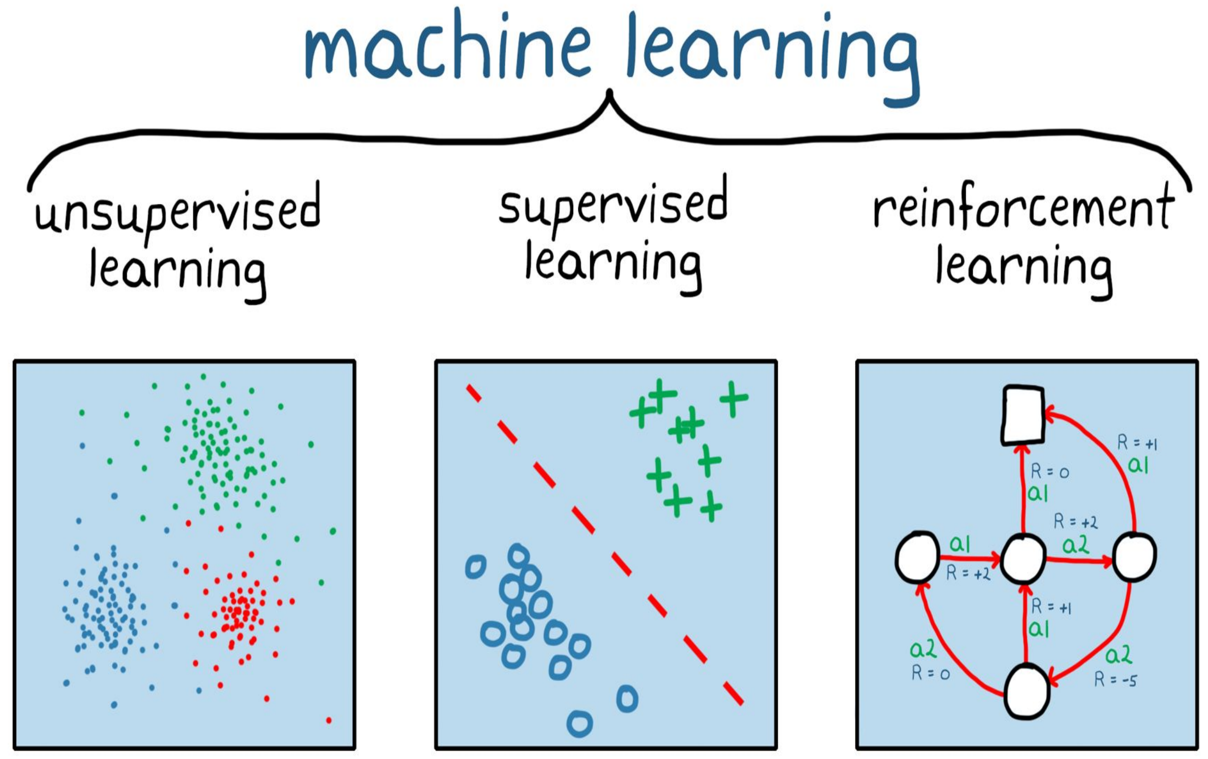
INTRODUCTION

- **Machine Learning**: Helps businesses make data-driven decisions.

- **Types of Machine Learning**:
  - **Supervised Learning**: Uses labeled data for predictions.
    - **Classification**: Categorizes data (e.g., predicting wine quality).
    - **Regression**: Predicts continuous values (e.g., estimating property prices).
  - **Unsupervised Learning**: Identifies patterns in unlabeled data (e.g., customer segmentation).

- **Project Overview**:
- **Wine Dataset (Classification)**: Predict wine quality based on features.
- **Real Estate Dataset (Regression)**: Predict property prices based on attributes.

Source: Find Open Datasets and Machine Learning Projects | Kaggle



Source: MathWorks

CLASSIFICATION

- ➔ Classification is a supervised learning technique used to categorize data into predefined labels or classes. It involves training a model on labeled data so it can predict the class of new, unseen instances.

CASE STUDY

- ➔ In this case study, we are applying classification to the Wine Quality dataset.
- ➔ The Wine Quality dataset consists of various chemical properties of red wine samples, including alcohol content, pH, acidity, and sugar levels. Each wine sample is labeled with a quality score, making it suitable for classification tasks to predict wine quality based on its chemical attributes.
- ➔ Models used: Naive Bayes, Decision Tree, Random Forest & KNN

5-Step Process

01	**Importing and Exploration**	Importing Python libraries, Dataset & exploring basic statistics
02	**Preprocessing**	Data cleaning, encoding categorical variables
03	**Scaling**	Normalizing feature values & splitting into training and testing sets
04	**Training and Evaluation**	Training a classification model & evaluating its performance using accuracy metrics
05	**Tuning and Predictions**	Optimize model parameters and generate predictions with a classification report

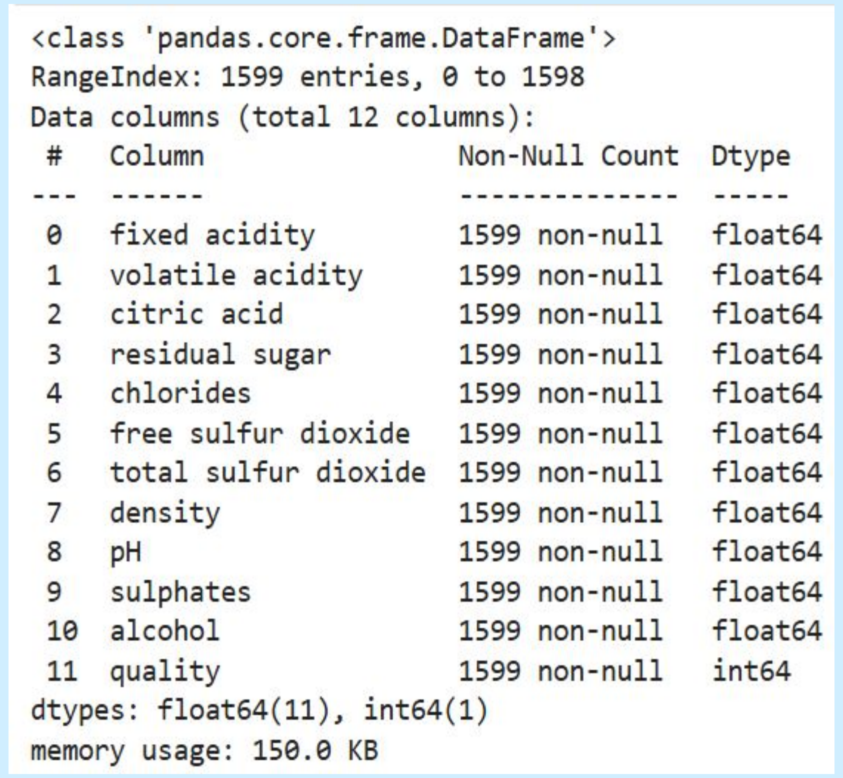
01

Importing and Exploration

Importing Python libraries, Dataset & exploring basic statistics

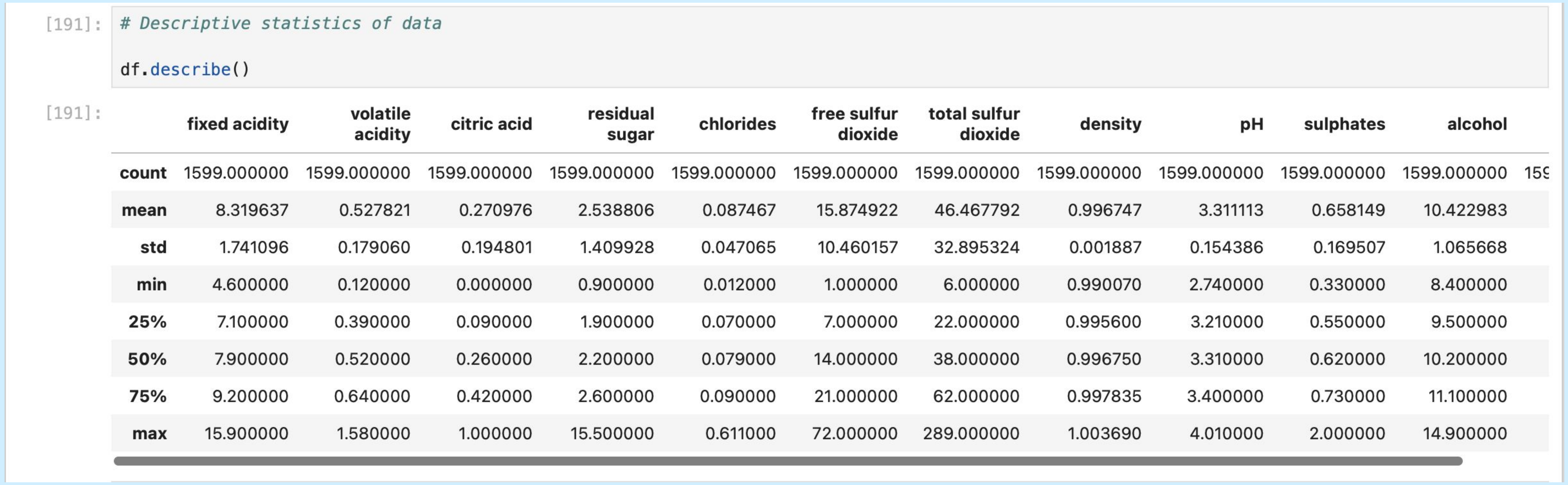
➔ In the first step, we imported necessary libraries such as numpy and pandas for data manipulation, and scikit-learn modules like train_test_split for splitting the dataset, MinMaxScaler for scaling, and classifiers like DecisionTreeClassifier and RandomForestClassifier. We also included metrics for evaluating model performance, such as accuracy and F1 score. Additionally, matplotlib and seaborn are used for data visualization.The dataset is read using pandas from a CSV file containing information on wine characteristics, such as acidity, sugar content, and alcohol levels, which will be used to predict the quality of wine.

➔ Next, the df.describe() function provides summary statistics for each feature, helping us understand the data distribution and detect any outliers. The df.info() function displays the dataset's structure, including column names, data types, and non-null counts, giving an overview of missing values and data types.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

Checking the data types



```
[191]: # Descriptive statistics of data

       df.describe()
```

[191]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	159
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	

Descriptive Statistics

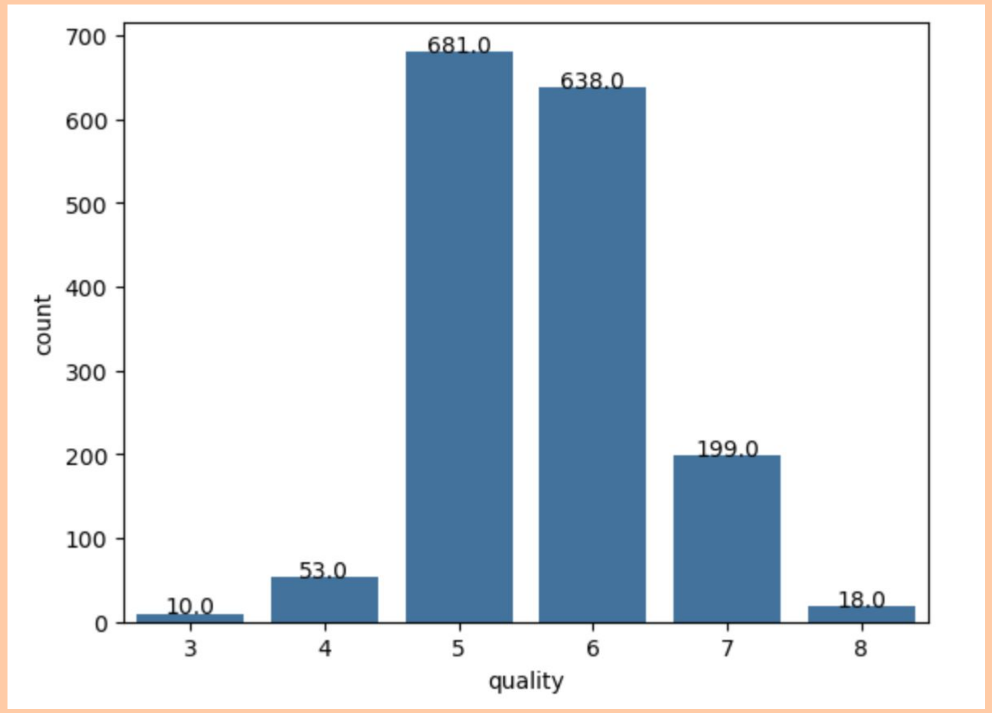
02

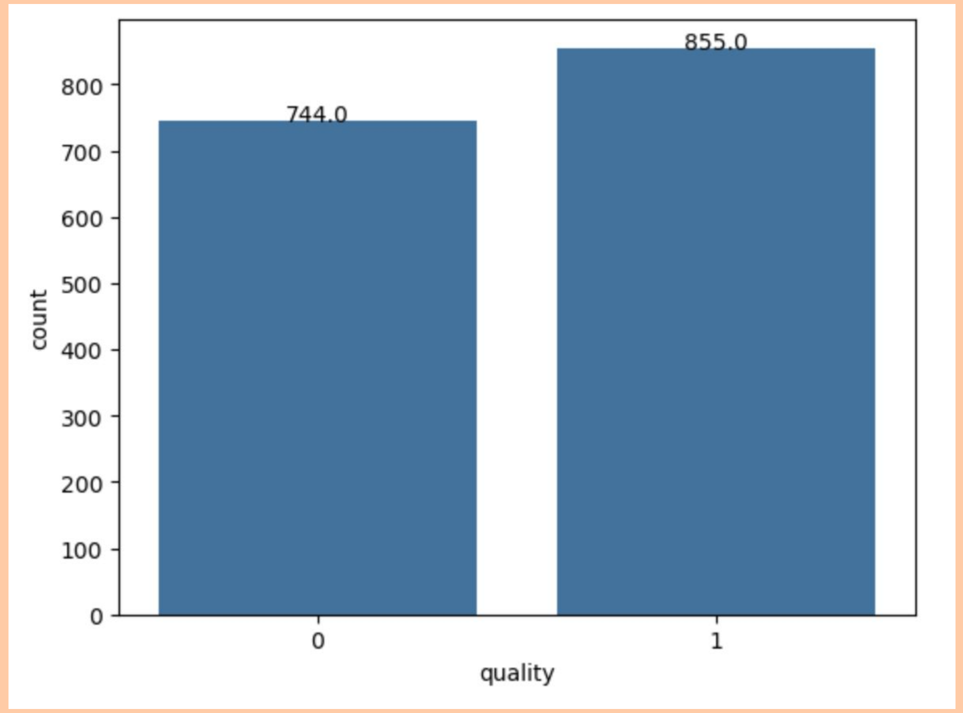
Preprocessing

Data cleaning, encoding categorical variables

➔ In step 2 we started inspecting its structure to understand the features and target variables. We handled any missing values by employing appropriate imputation techniques to ensure data integrity. Categorical variables were encoded to convert them into a numerical format suitable for model training, enhancing the dataset's usability.

Data Balancing





Split Quality column into two labels:
- High quality: 6 to 8 is represented by 1
- Low quality: 3 to 5 is represented by 0

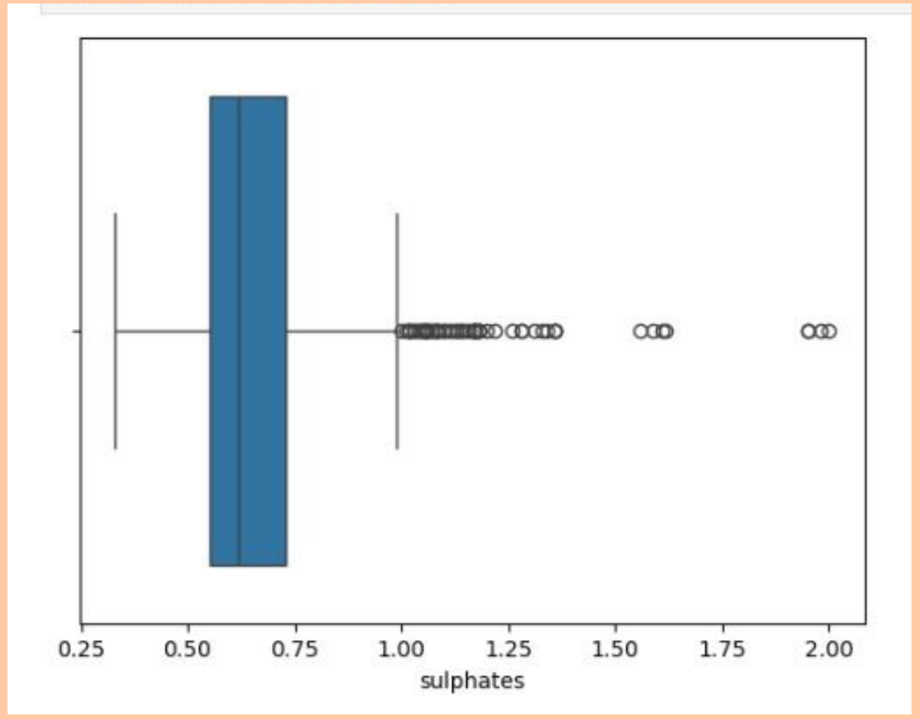
02

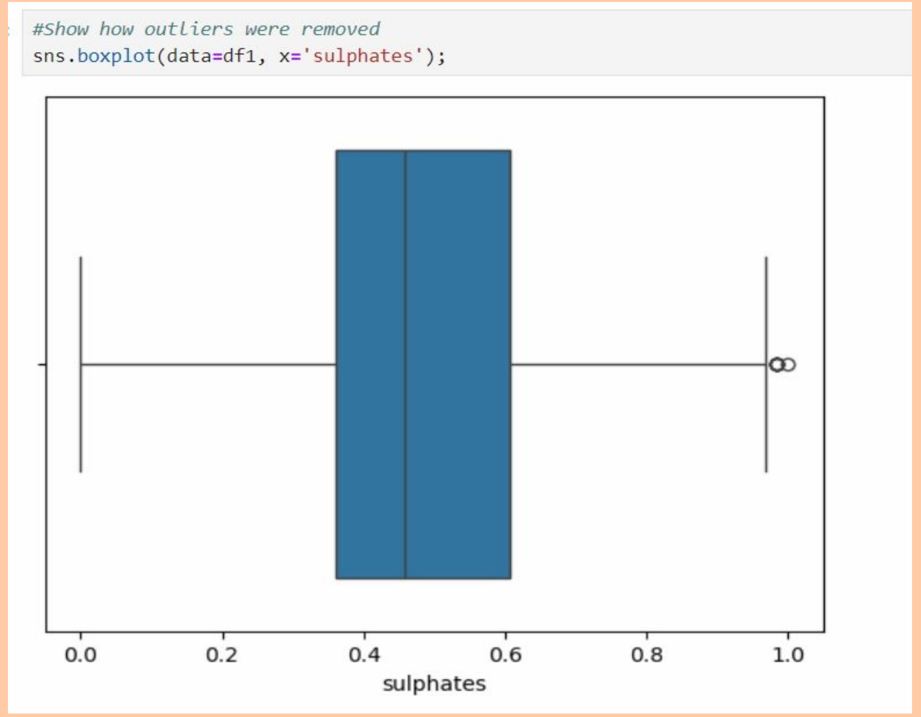
Preprocessing

Data cleaning, encoding categorical variables

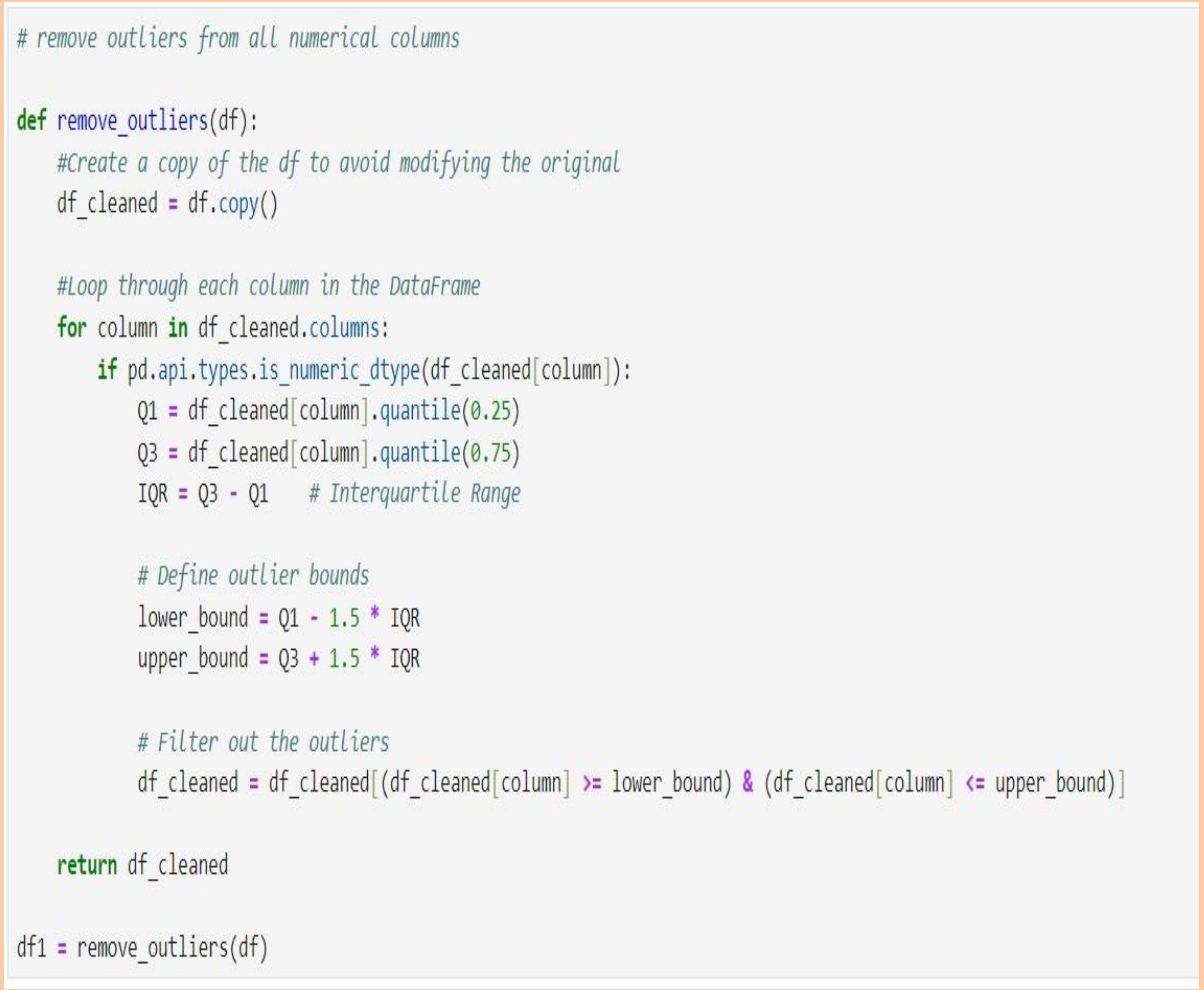
➔ We have observed outliers in this data set and we've removed the outliers using ***interquartile range*** method.

Outliers





```
#Show how outliers were removed
sns.boxplot(data=df1, x='sulphates');
```



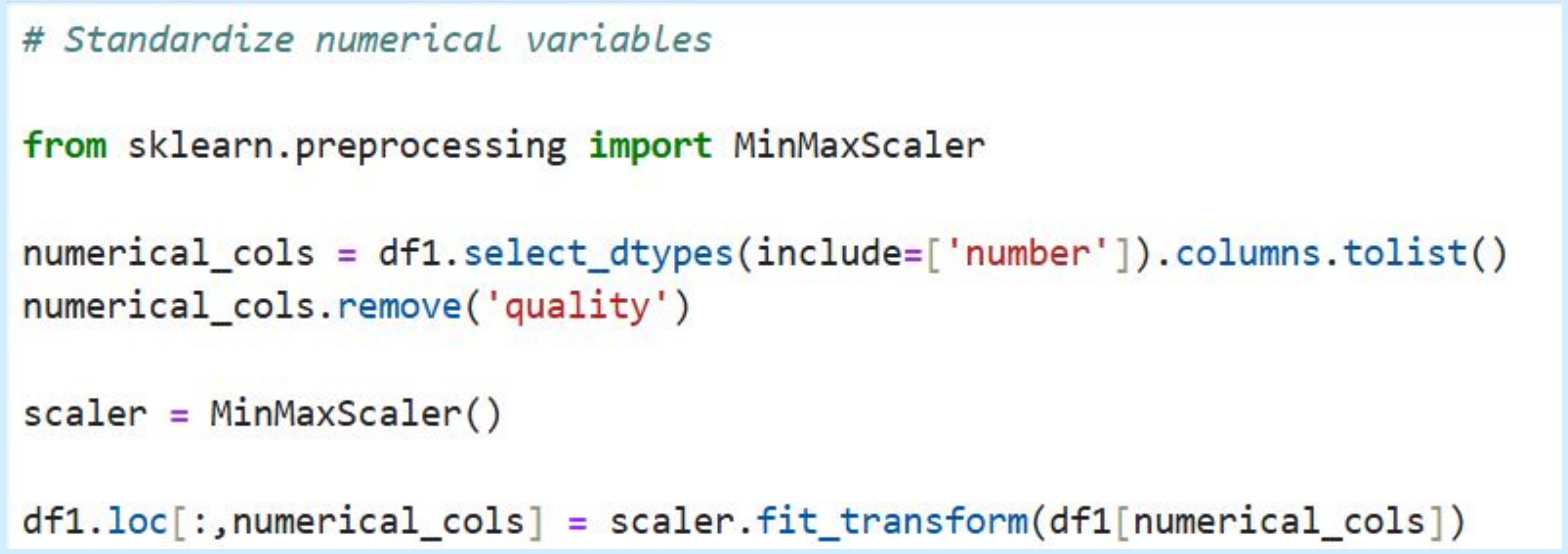
```
# remove outliers from all numerical columns

def remove_outliers(df):
    #Create a copy of the df to avoid modifying the original
    df_cleaned = df.copy()

    #Loop through each column in the DataFrame
    for column in df_cleaned.columns:
        if pd.api.types.is_numeric_dtype(df_cleaned[column]):
            Q1 = df_cleaned[column].quantile(0.25)
            Q3 = df_cleaned[column].quantile(0.75)
            IQR = Q3 - Q1    # Interquartile Range

            # Define outlier bounds
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR

            # Filter out the outliers
            df_cleaned = df_cleaned[(df_cleaned[column] >= lower_bound) & (df_cleaned[column] <= upper_bound)]

    return df_cleaned

df1 = remove_outliers(df)
```

03

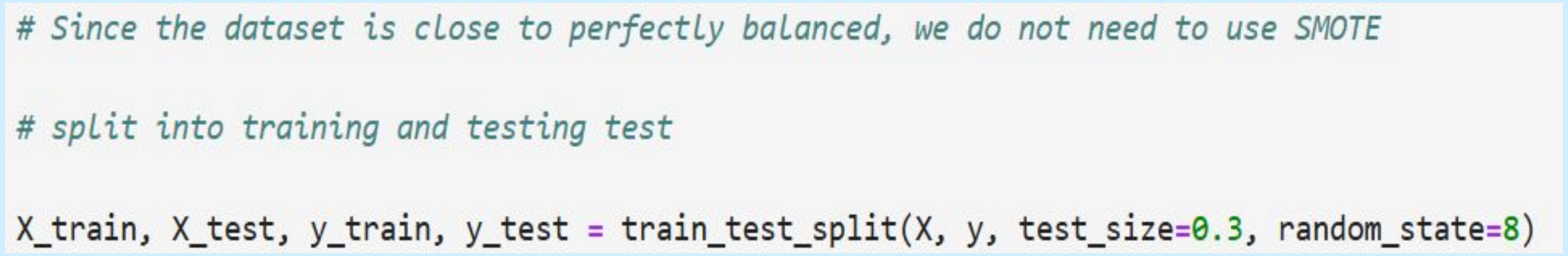
Scaling

Normalizing feature values & splitting into training and testing sets

➔ Subsequently, we scaled the features using standardization to normalize the feature values, which is crucial for many machine learning algorithms that are sensitive to the scale of input data. Finally, the dataset was split into training and testing subsets to facilitate the evaluation of model performance while preventing overfitting.



```
# Standardize numerical variables

from sklearn.preprocessing import MinMaxScaler

numerical_cols = df1.select_dtypes(include=['number']).columns.tolist()
numerical_cols.remove('quality')

scaler = MinMaxScaler()

df1.loc[:,numerical_cols] = scaler.fit_transform(df1[numerical_cols])
```



```
# Since the dataset is close to perfectly balanced, we do not need to use SMOTE

# split into training and testing test

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=8)
```

30% of the dataset is for training and 70% of the dataset is for testing

➔ Features / Input variable: X = df1. drop(columns=[‘quality’]) - (independent)

➔ Target / Output variable: Y = df1[‘quality’] - (dependent)

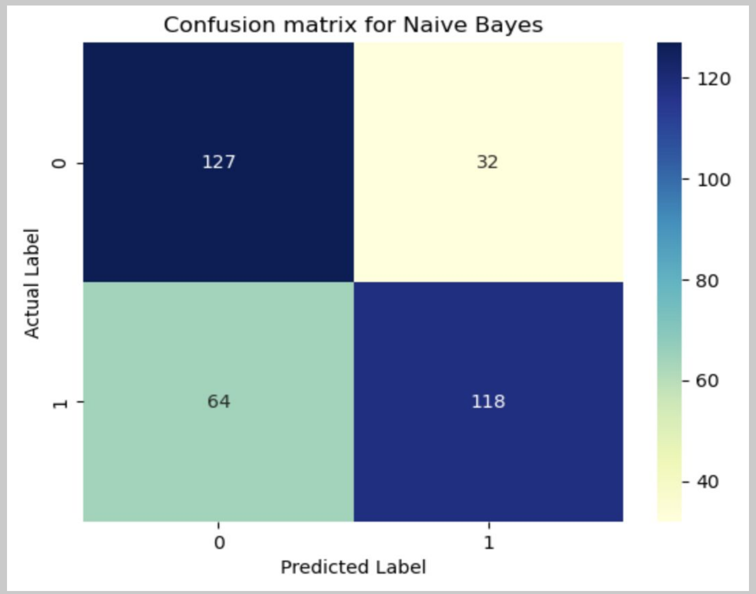
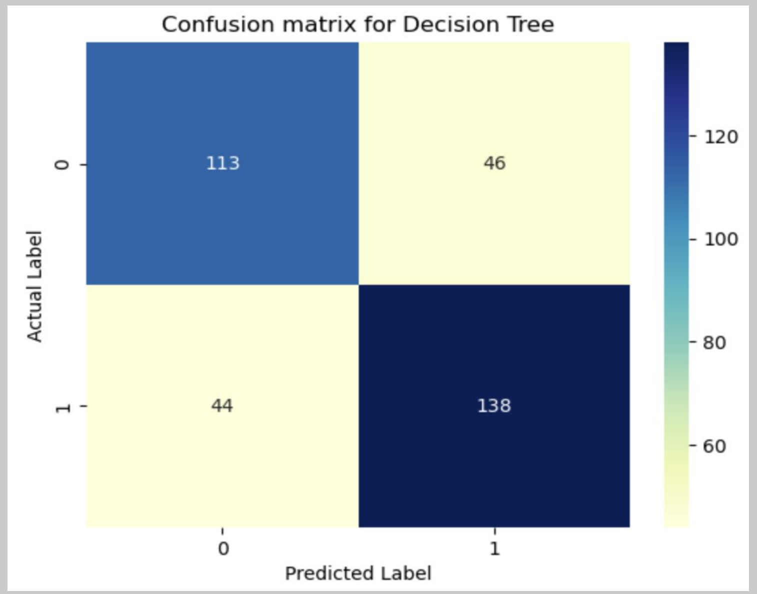
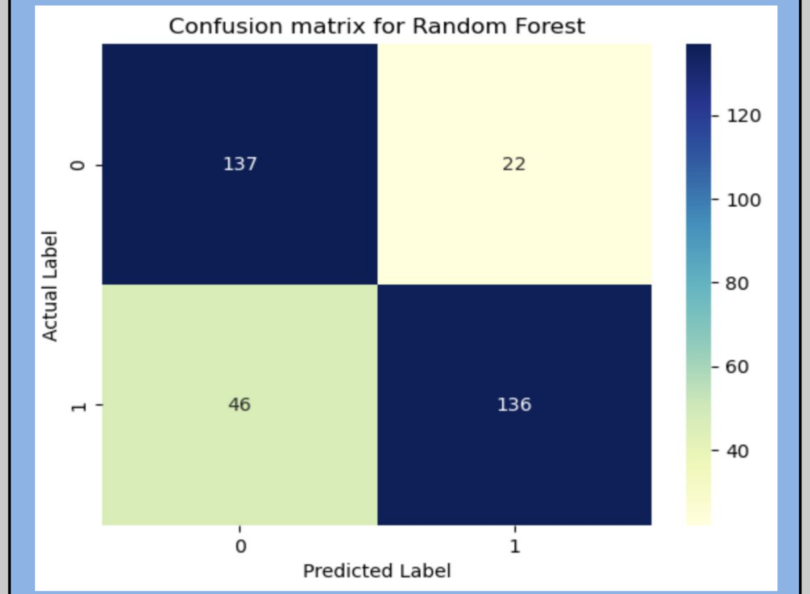
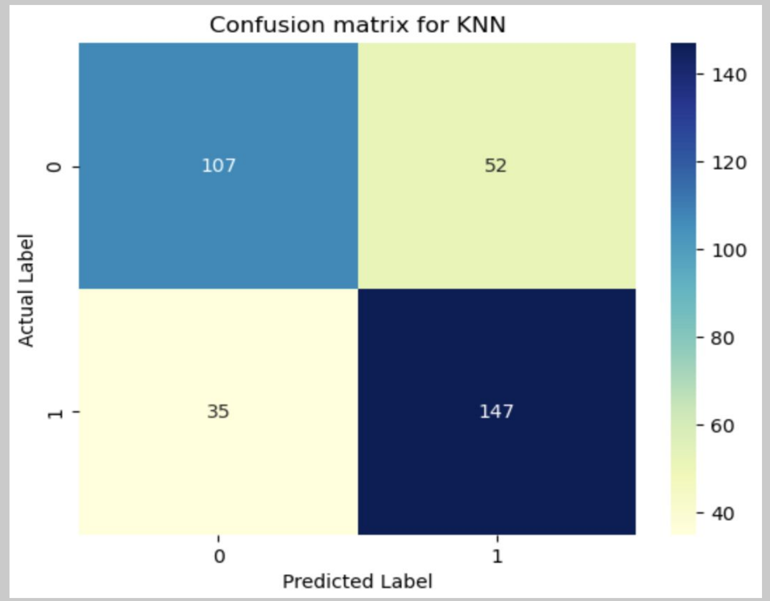
04

Training and Evaluation

Training a classification model & evaluating its performance using accuracy metrics

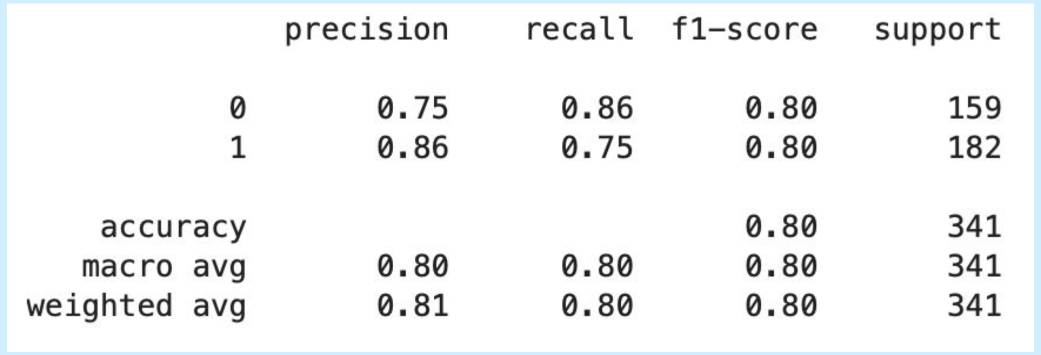
CONDITIONAL PROBABILITIES

● P (quality = 1 | 'alcohol' >= 0.325581)	● 431/586 = 0.74 or 74%
● P (quality = 1 | 'sulphates' >= 0.459016)	● 400/590 = 0.68 or 68%
● P (quality = 1 | 'residual sugar' >= 0.0.375000)	● 363/651 = 0.56 or 56%
● P (quality = 1 | 'citric acid' >= 0.328767)	● 362/576 = 0.63 or 63%
● P (quality = 1 | 'alcohol' >= 0.325581, 'sulphates' >= 0.459016)	● 302/368 = 0.82 or 82%

METRICS \ MODELS	Naïve Bayes	Decision Tree	Random Forest	KNN
Accuracy	0.72	0.74	0.80	0.74
Precision	0.79	0.75	0.86	0.74
Recall or Sensitivity	0.65	0.76	0.75	0.81
F1 Score	0.71	0.75	0.80	0.77
Specificity	0.79	0.71	0.86	0.67
Confusion Matrix	[[127 32] [64 118]]	[[113 46] [44 138]]	[[137 22] [46 136]]	[[107 52] [35 147]]
				

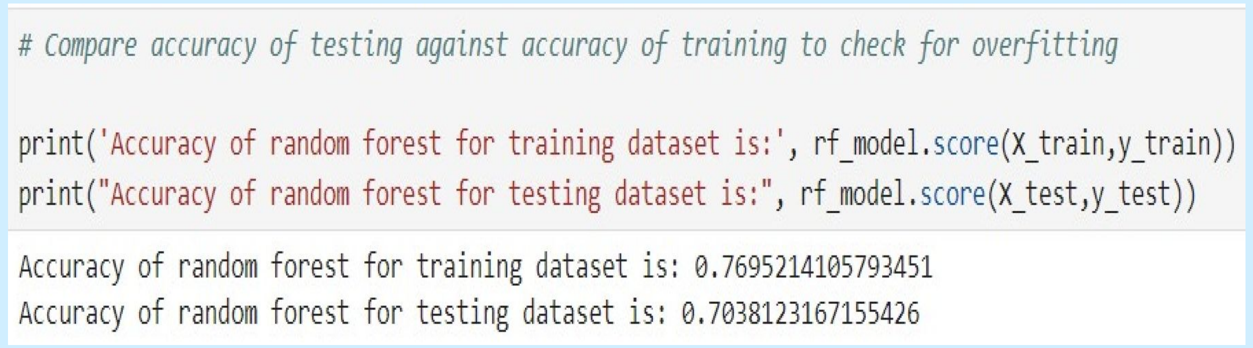
Random Forest has the best performance as it has highest scores

- RF Classification Report:



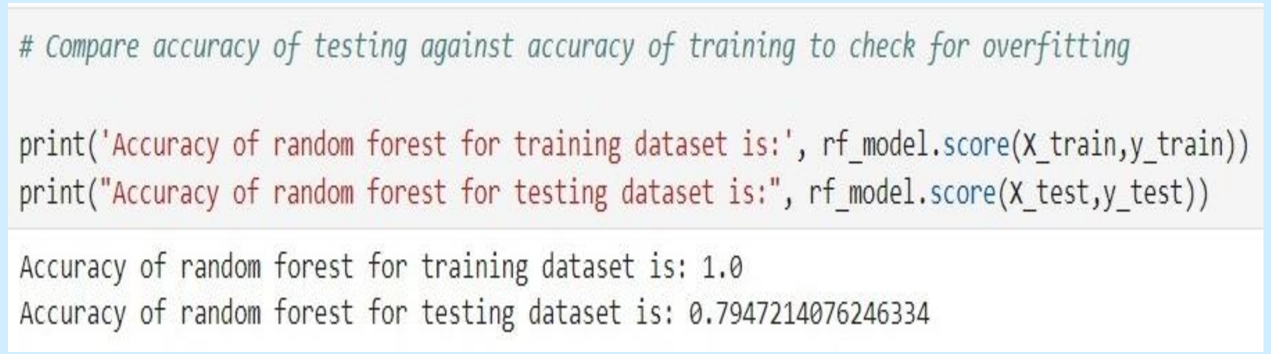
```
              precision    recall  f1-score   support

           0       0.75      0.86      0.80       159
           1       0.86      0.75      0.80       182

    accuracy                           0.80       341
   macro avg       0.80      0.80      0.80       341
weighted avg       0.81      0.80      0.80       341
```

- Compare accuracy of testing against accuracy of training to check for overfitting:



```
# Compare accuracy of testing against accuracy of training to check for overfitting

print('Accuracy of random forest for training dataset is:', rf_model.score(X_train,y_train))
print("Accuracy of random forest for testing dataset is:", rf_model.score(X_test,y_test))
```

```
Accuracy of random forest for training dataset is: 0.7695214105793451
Accuracy of random forest for testing dataset is: 0.7038123167155426
```

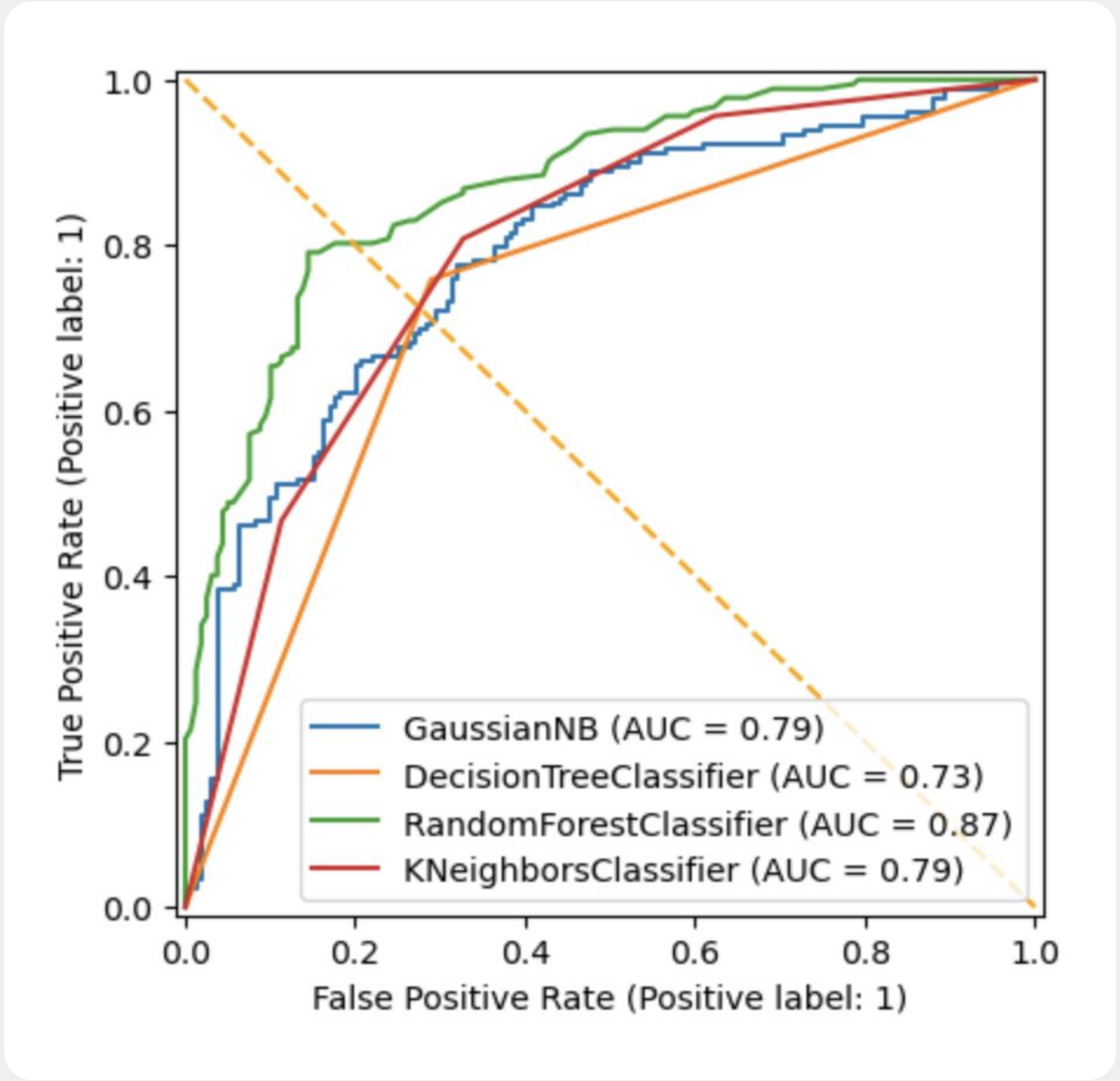


```
# Compare accuracy of testing against accuracy of training to check for overfitting

print('Accuracy of random forest for training dataset is:', rf_model.score(X_train,y_train))
print("Accuracy of random forest for testing dataset is:", rf_model.score(X_test,y_test))
```

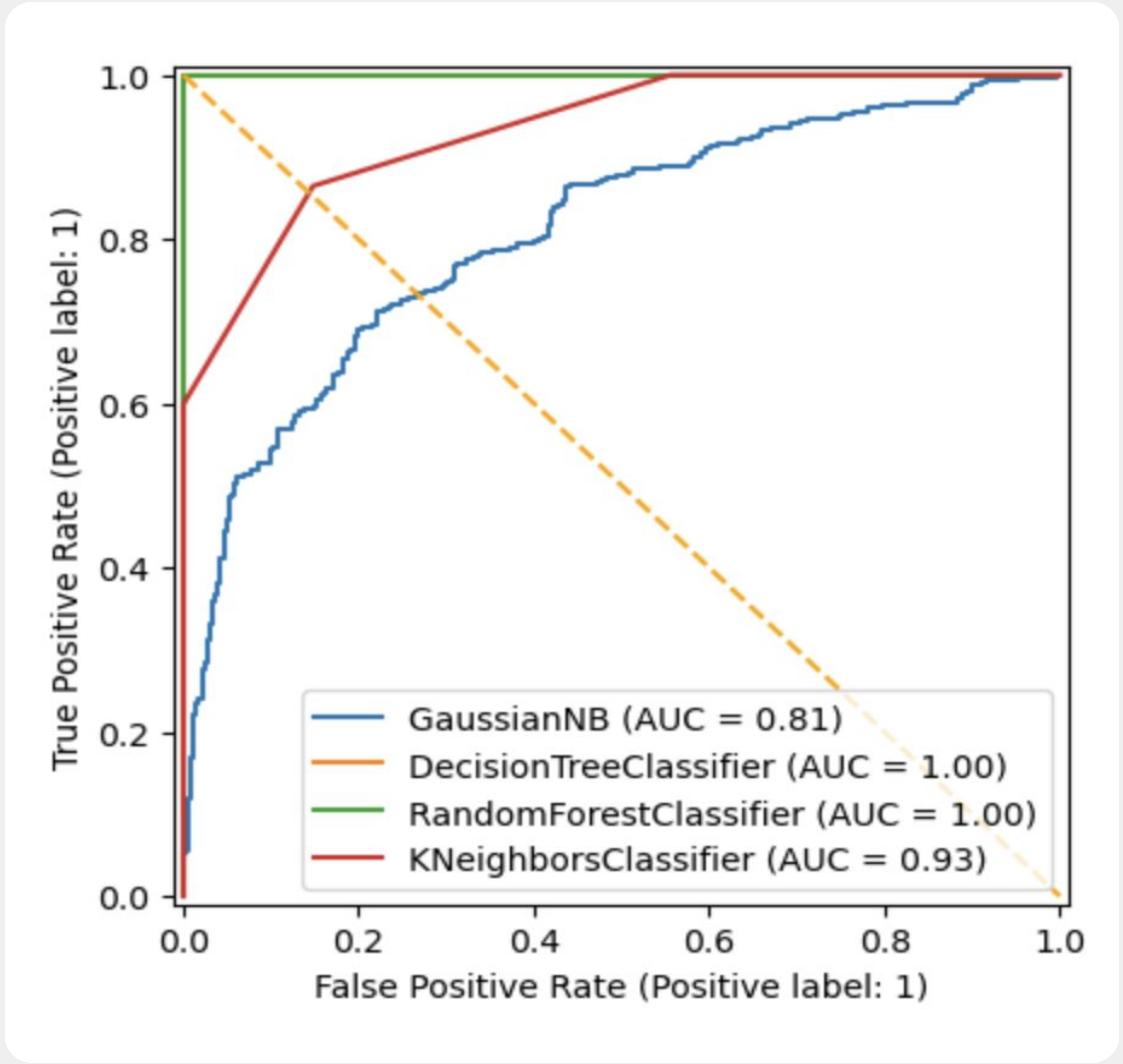
```
Accuracy of random forest for training dataset is: 1.0
Accuracy of random forest for testing dataset is: 0.7947214076246334
```

ROC Curve

Testing

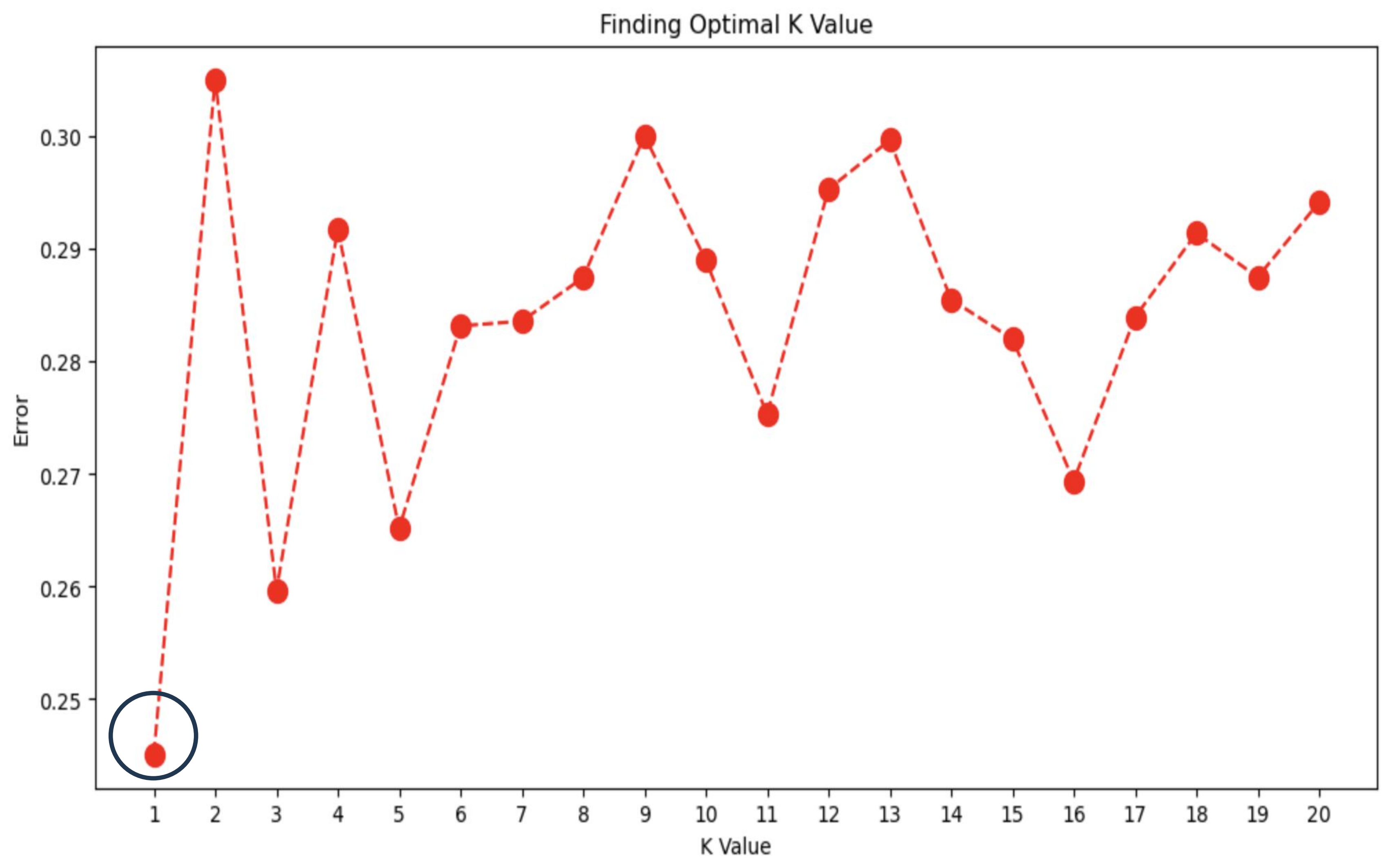


Training



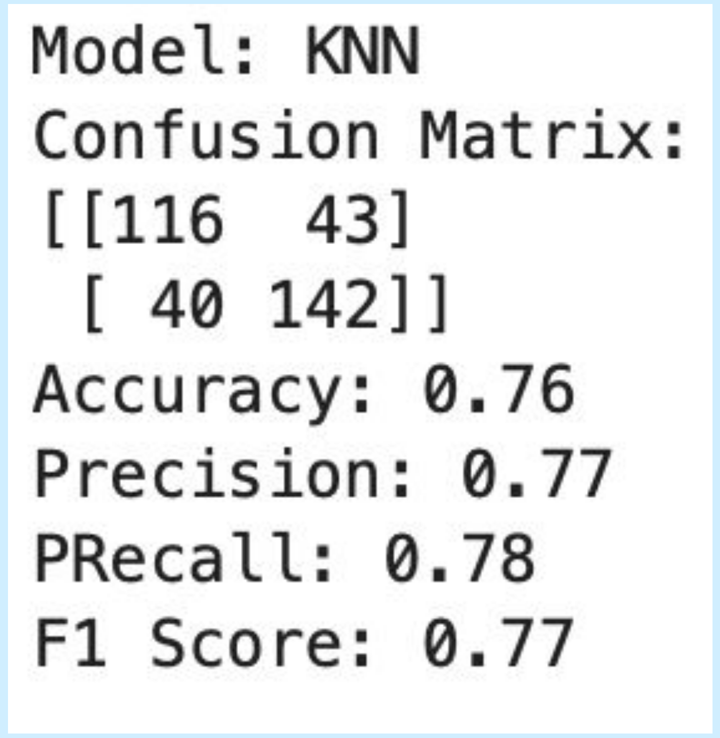
Our random forest model has a AUC of 0.88. While not excellent, this value is high enough to confirm that the models possess a good amount discriminative ability.

Finding the best value of K in KNN



KNN with k=1 (accuracy is 76%) performs better than
KNN with k=3 (accuracy is 74%)

Create a new KNN model using the optimal number of neighbors



```
Model: KNN
Confusion Matrix:
[[116  43]
 [ 40 142]]
Accuracy: 0.76
Precision: 0.77
PRecall: 0.78
F1 Score: 0.77
```

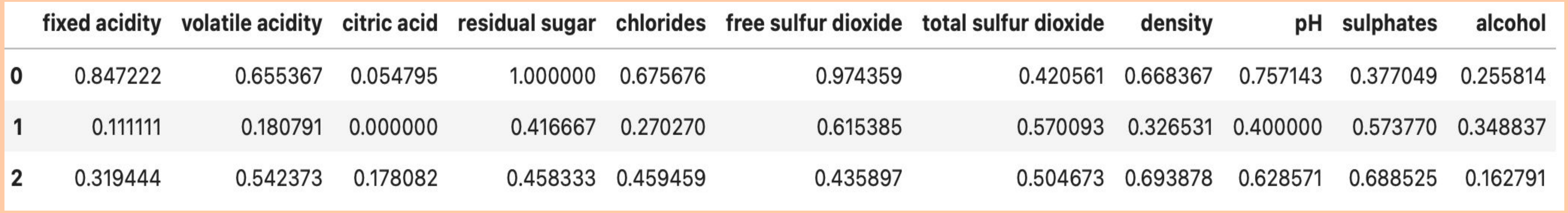
- ➔ These results indicate that the KNN model performs reasonably well with an accuracy of 76%, meaning it correctly classifies 76% of the data.
- ➔ The precision of 77% suggests that out of all predicted positive instances, 77% are correctly classified.
- ➔ The recall of 78% shows the model captures 78% of actual positive cases.
- ➔ The F1 score of 0.77 balances precision and recall, indicating good overall performance.

05

Tuning and Predictions

Optimize model parameters and generate predictions with a classification report

➔ Code reads a CSV file into new_data and uses a pre-trained Random Forest model (rf_model) to predict outcomes for this data. A new Random Forest Classifier (rf) is created and trained on features (X) and labels (y). After training, the feature importances (I) are extracted to identify which features contribute most to the model. The feature names are obtained by dropping the target column, and a new DataFrame is created to pair the importance scores with the feature names.



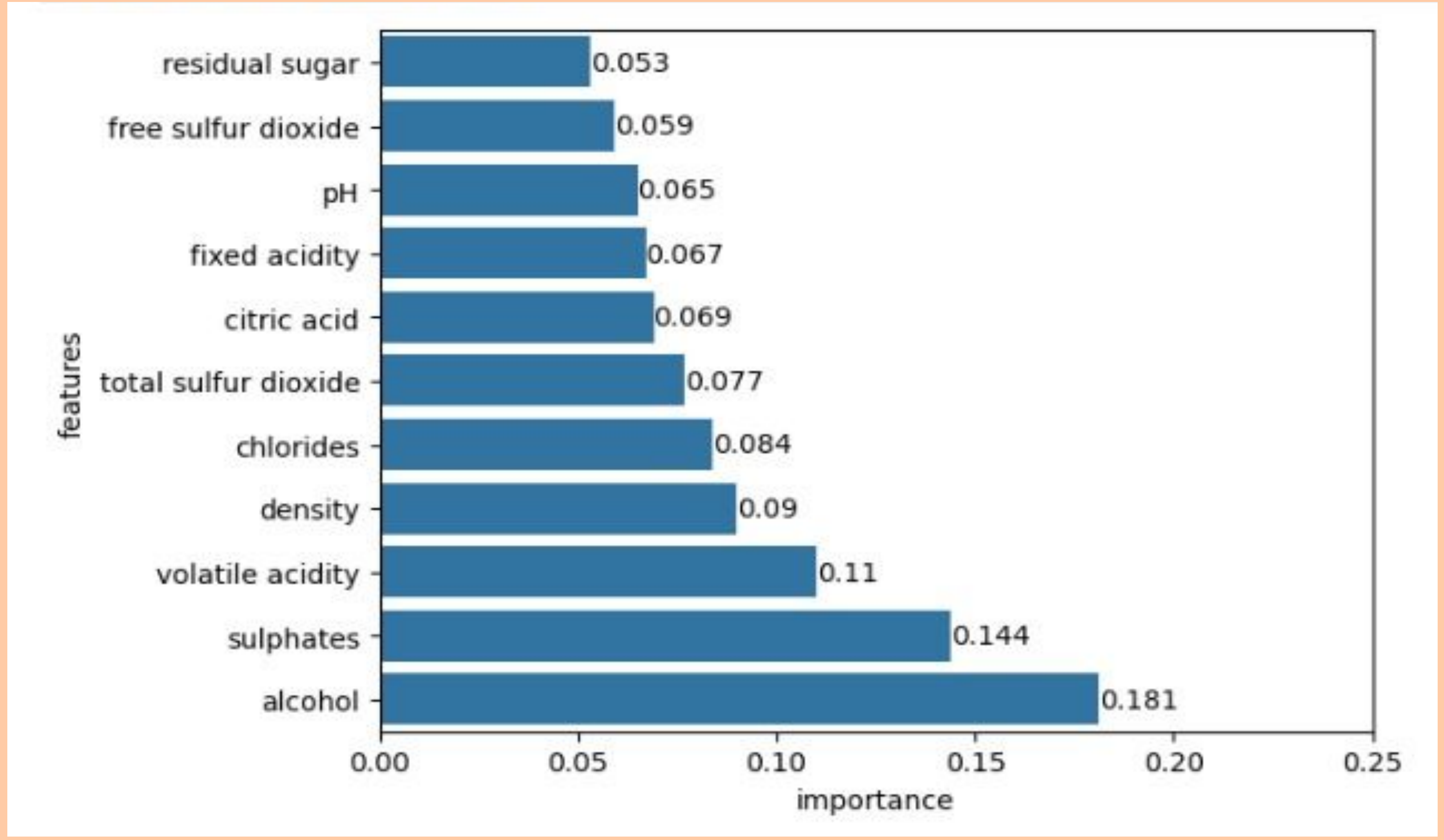
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	0.847222	0.655367	0.054795	1.000000	0.675676	0.974359	0.420561	0.668367	0.757143	0.377049	0.255814
1	0.111111	0.180791	0.000000	0.416667	0.270270	0.615385	0.570093	0.326531	0.400000	0.573770	0.348837
2	0.319444	0.542373	0.178082	0.458333	0.459459	0.435897	0.504673	0.693878	0.628571	0.688525	0.162791

➔

Predicted Quality
0
1
0

Find the feature importance using RF

➔ "Feature Importance" shows that 'alcohol' is the most important feature, followed by 'sulphates' and 'volatile acidity'. The least important features are 'residual sugar' and 'free sulfur dioxide'. This suggests that 'alcohol' has the highest impact on the model's predictions, while 'residual sugar' has the least.



INSIGHTS

- ➔ After comparing all the models we can say that **Random Forest** consistently delivered the best performance across most metrics, particularly excelling in precision, accuracy, and F1 score. This model demonstrated high reliability in classifying both positive and negative instances, making it the best choice for this dataset. The KNN model also performed well after parameter tuning, and it is notable for its ability to reduce false negatives. However, the Random Forest model is preferable for deployment due to its strong overall performance and generalizability.

- ➔ From the dataset we tried to add new data and predict the quality which had given results [0,1,0]

- ➔ Feature Importance" shows that 'alcohol' is the most important feature, followed by 'sulphates' and 'volatile acidity'. The least important features are 'residual sugar' and 'free sulfur dioxide'. This suggests that 'alcohol' has the highest impact on the model's predictions, while 'residual sugar' has the lowest.

REGRESSION

- ➔ Regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It helps predict outcomes based on input features.
- ➔ Common regression techniques include linear regression, polynomial regression, and logistic regression. Regression analysis is widely used in fields like economics, medicine, and machine learning to forecast trends and analyze data patterns.

CASE STUDY

- ➔ The Real Estate Valuation Data Set from UCI contains information on house prices in Taiwan. It includes six features such as the age of the house, the distance to the nearest MRT station, number of convenience stores, latitude, and longitude to predict real estate prices.

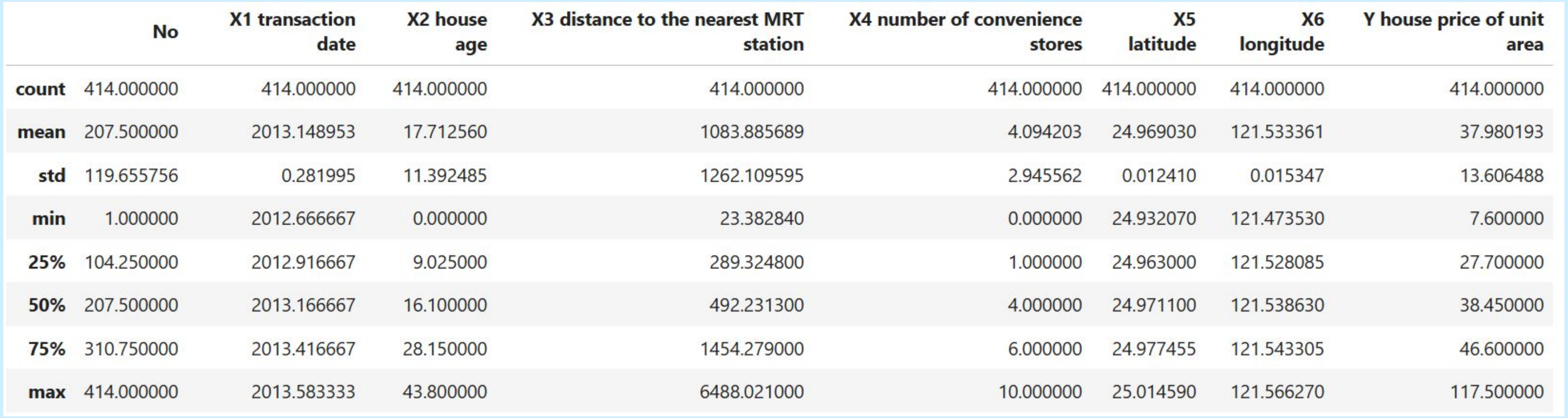
5-Step Process

01	**Data Loading and Preprocessing**	Load the dataset and preprocess it (handle missing values, normalization).
02	**Model Selection and Training**	Calculate correlation between features and the target variable.
03	**Equation of Regression Model**	Select regression models (Linear, Polynomial, Exponential) and train on the data
04	**Model Evaluation**	Display the regression equation for each model.
05	**Model Interpretation**	Evaluate the models using R^2 and adjusted R^2, and interpret the results.

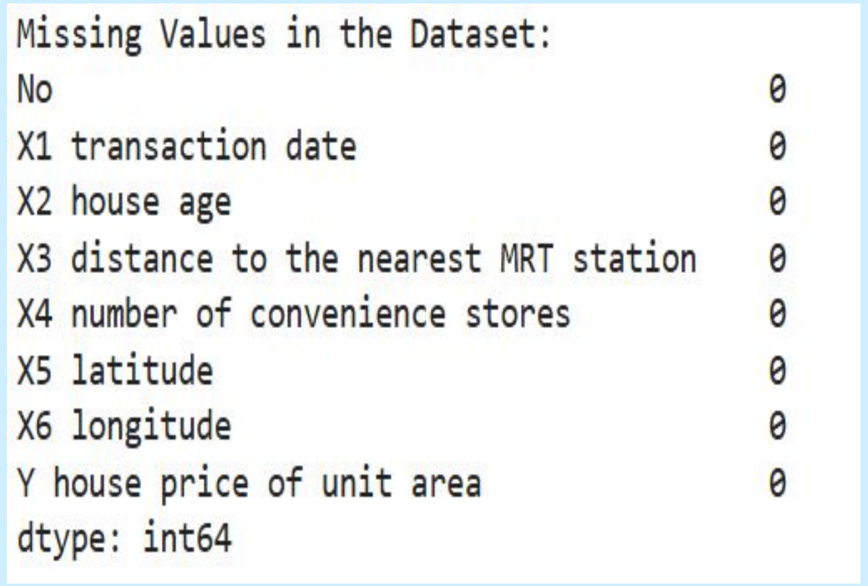
01

Data Loading and Preprocessing

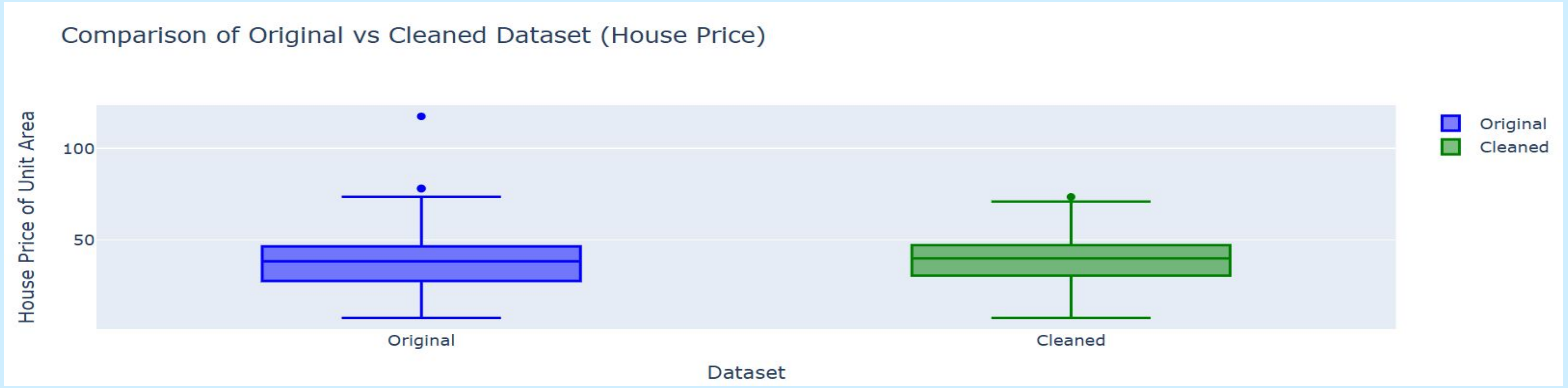
Load the dataset and preprocess it (handle missing values, normalization).



	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
count	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000
mean	207.500000	2013.148953	17.712560	1083.885689	4.094203	24.969030	121.533361	37.980193
std	119.655756	0.281995	11.392485	1262.109595	2.945562	0.012410	0.015347	13.606488
min	1.000000	2012.666667	0.000000	23.382840	0.000000	24.932070	121.473530	7.600000
25%	104.250000	2012.916667	9.025000	289.324800	1.000000	24.963000	121.528085	27.700000
50%	207.500000	2013.166667	16.100000	492.231300	4.000000	24.971100	121.538630	38.450000
75%	310.750000	2013.416667	28.150000	1454.279000	6.000000	24.977455	121.543305	46.600000
max	414.000000	2013.583333	43.800000	6488.021000	10.000000	25.014590	121.566270	117.500000



```
Missing Values in the Dataset:
No                                        0
X1 transaction date                       0
X2 house age                              0
X3 distance to the nearest MRT station    0
X4 number of convenience stores           0
X5 latitude                               0
X6 longitude                              0
Y house price of unit area                0
dtype: int64
```

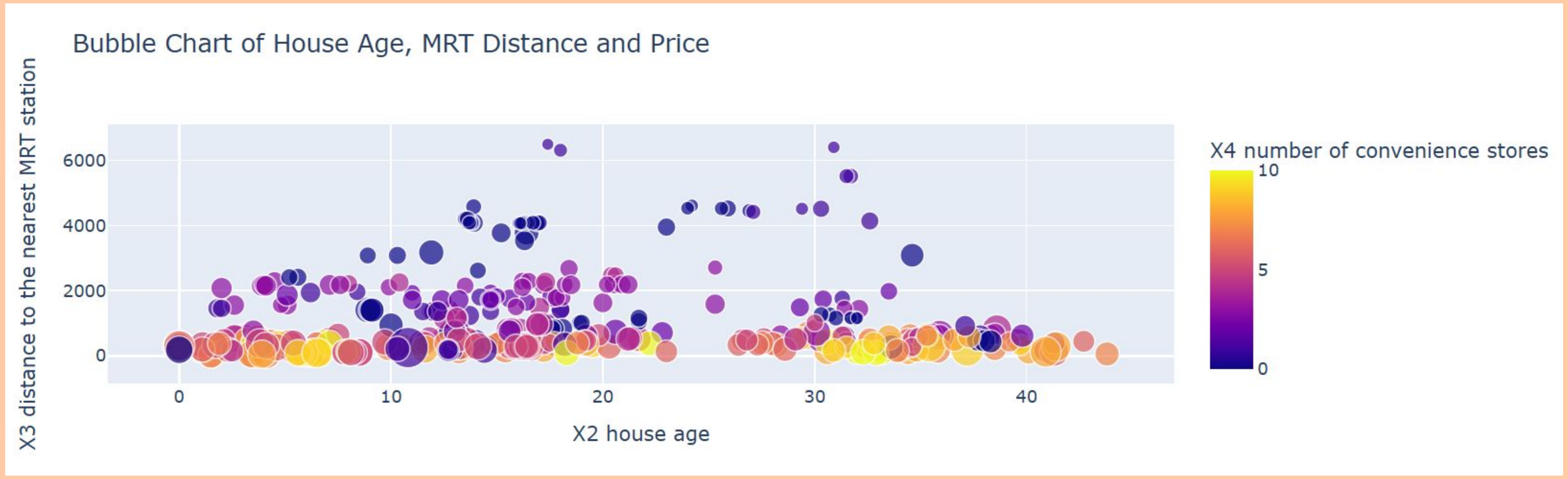


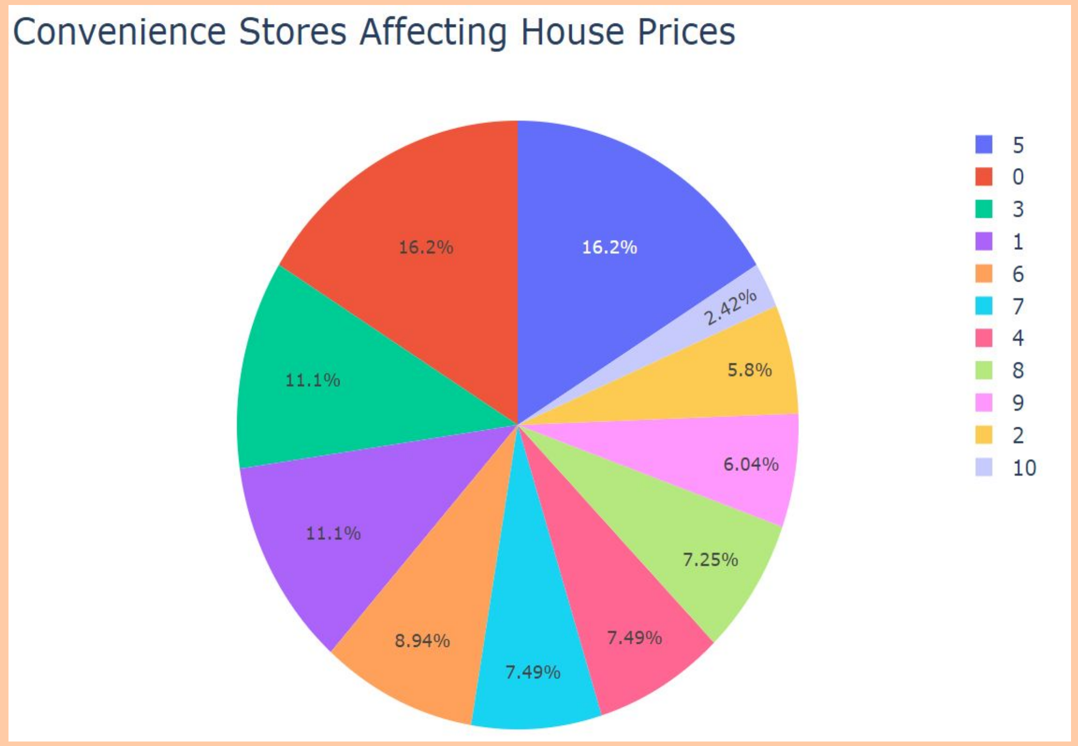
Original Dataset Shape: (414, 8), Cleaned Dataset Shape: (371, 8)

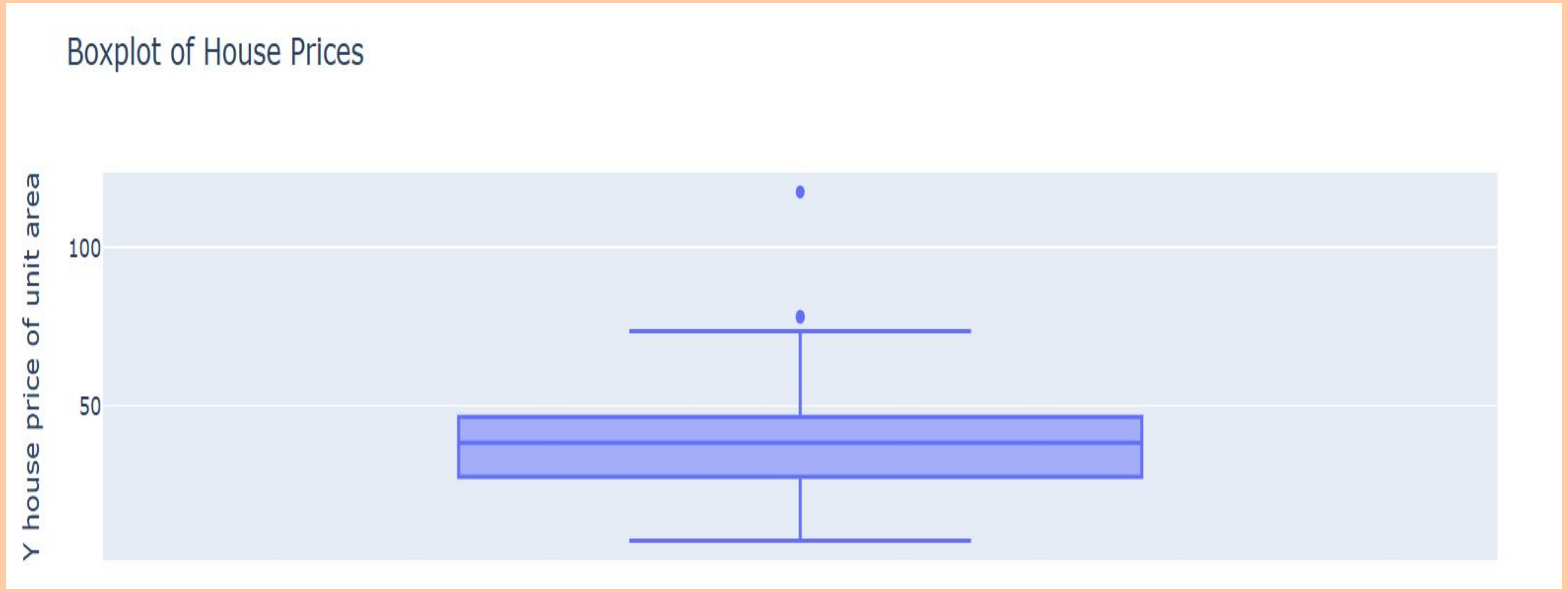
02

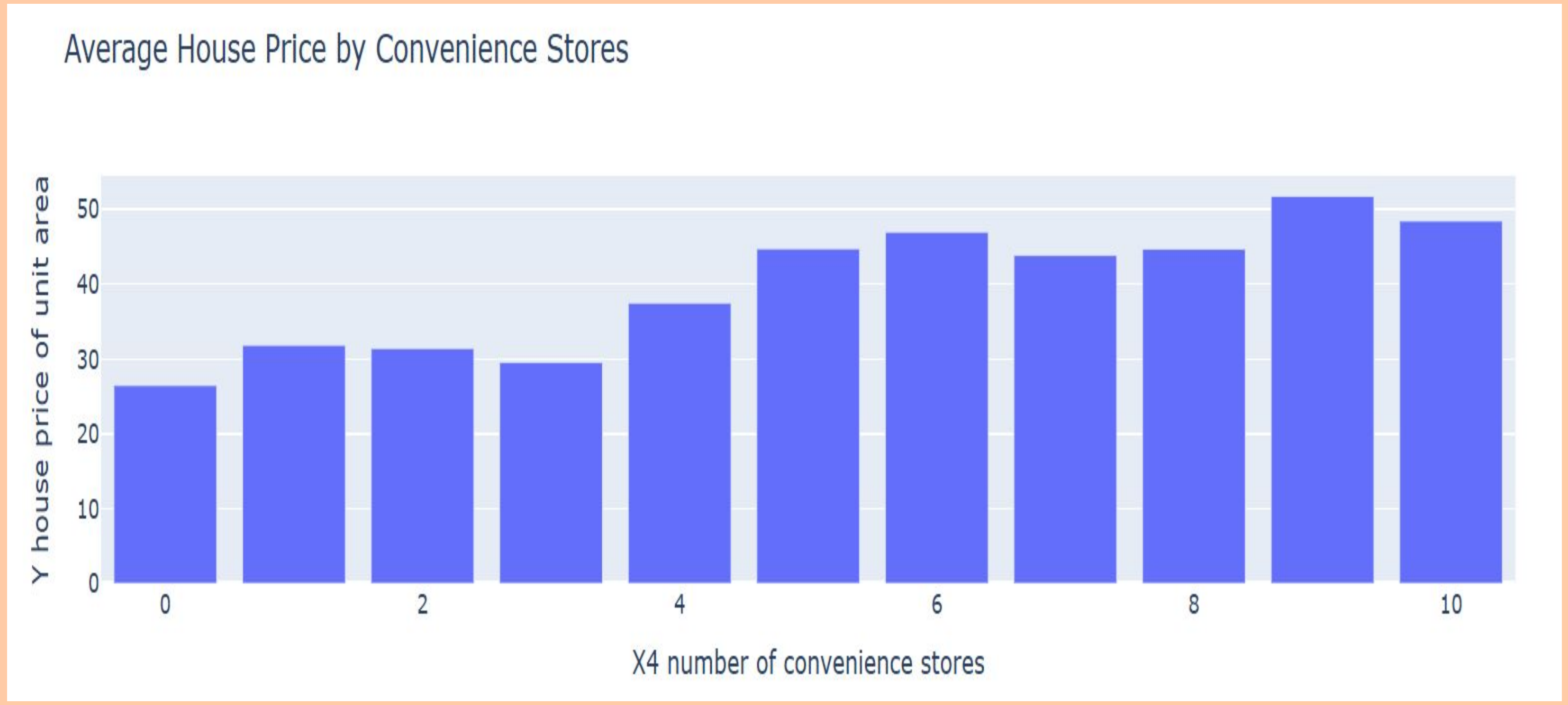
Model Selection and Training

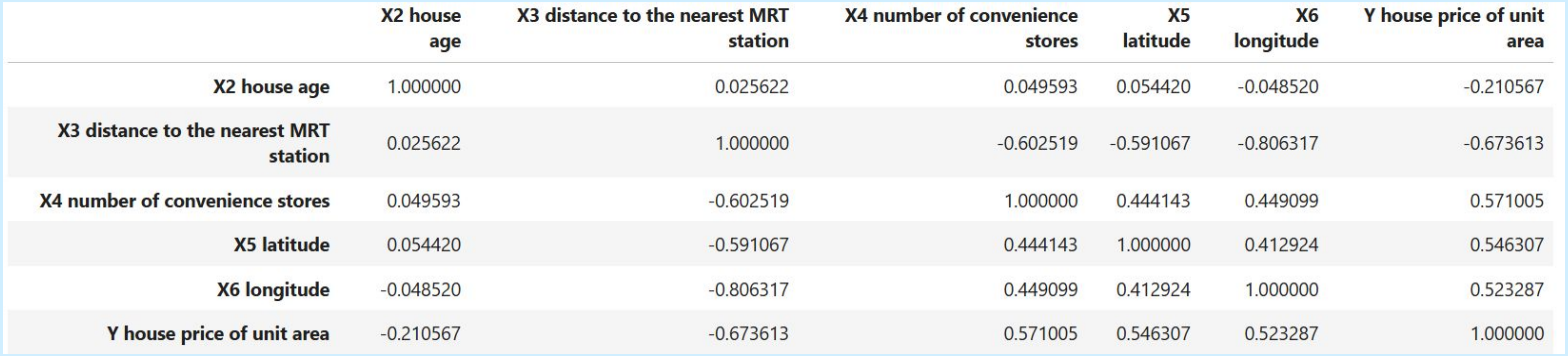
Calculate correlation between features and the target variable.



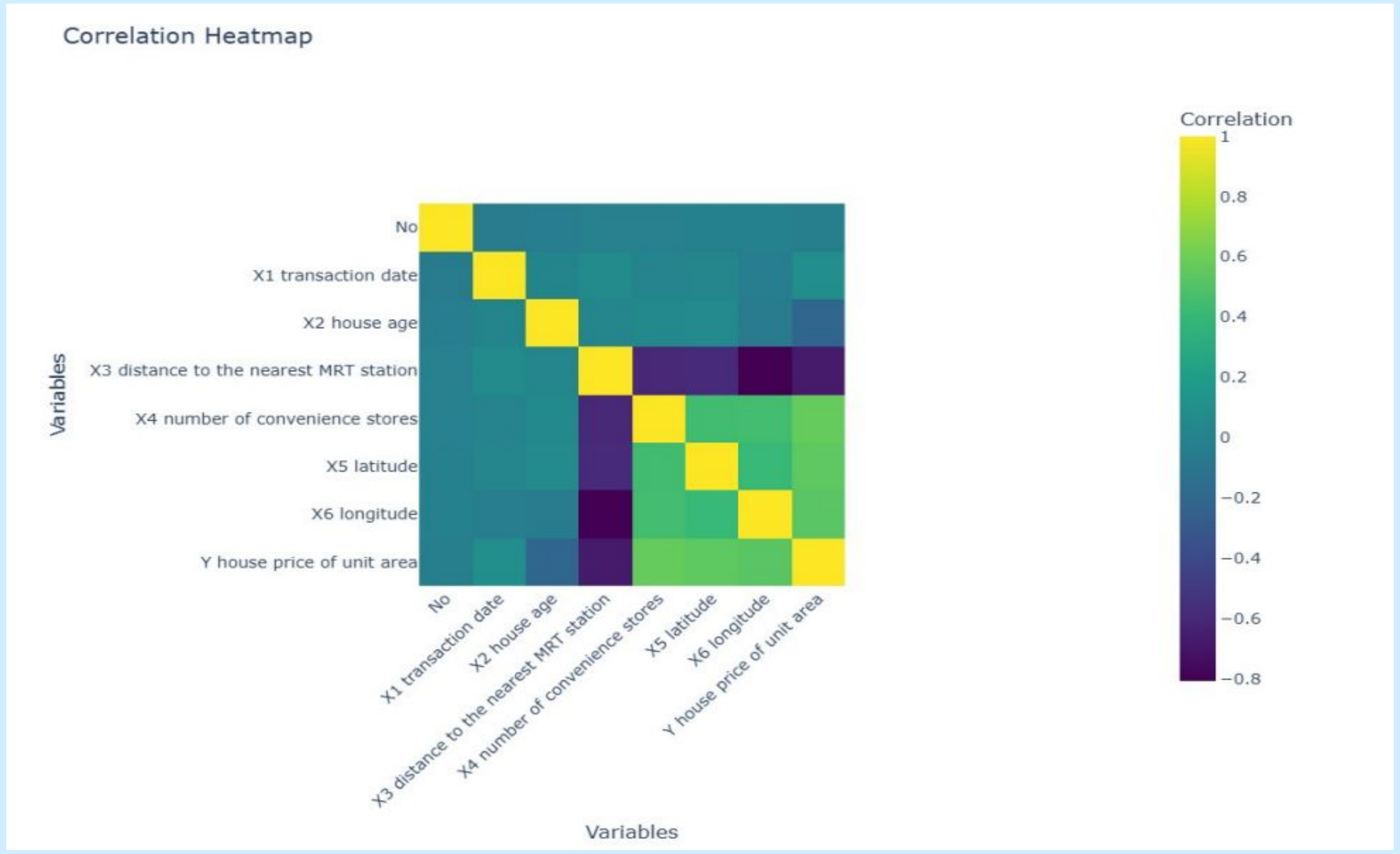








	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
X2 house age	1.000000	0.025622	0.049593	0.054420	-0.048520	-0.210567
X3 distance to the nearest MRT station	0.025622	1.000000	-0.602519	-0.591067	-0.806317	-0.673613
X4 number of convenience stores	0.049593	-0.602519	1.000000	0.444143	0.449099	0.571005
X5 latitude	0.054420	-0.591067	0.444143	1.000000	0.412924	0.546307
X6 longitude	-0.048520	-0.806317	0.449099	0.412924	1.000000	0.523287
Y house price of unit area	-0.210567	-0.673613	0.571005	0.546307	0.523287	1.000000



- ➔ House age (X2) has a coefficient of -0.2418, meaning for every additional year of house age, the house price decreases by 0.2418 units.
- ➔ Distance to MRT station (X3) has a coefficient of -0.0049, indicating that house prices decrease by 0.0049 units for each additional meter of distance from the MRT.
- ➔ Number of convenience stores (X4) has a coefficient of 0.57, suggesting that for each additional convenience store, house prices increase by 0.57 units.
- ➔ Latitude (X5) and longitude (X6) coefficients show that house prices are positively influenced by latitude (0.54) and negatively influenced by longitude (0.52), reflecting geographic effects.

03 Equation of Regression Model

Select regression models (Linear, Polynomial, Exponential) and train on the data

Y= −530.1680+(−0.2418×X2 house age)+(−0.0049×X3 distance to the nearest MRT station)+(1.0860×X4 number of convenience stores)+(252.0608×X5 latitude)+(−47.0680×X6 longitude)

➔ This equation shows how each independent variable affects house prices. For example, house age and distance to the MRT station negatively affect prices, while the number of convenience stores and latitude positively influence house prices.

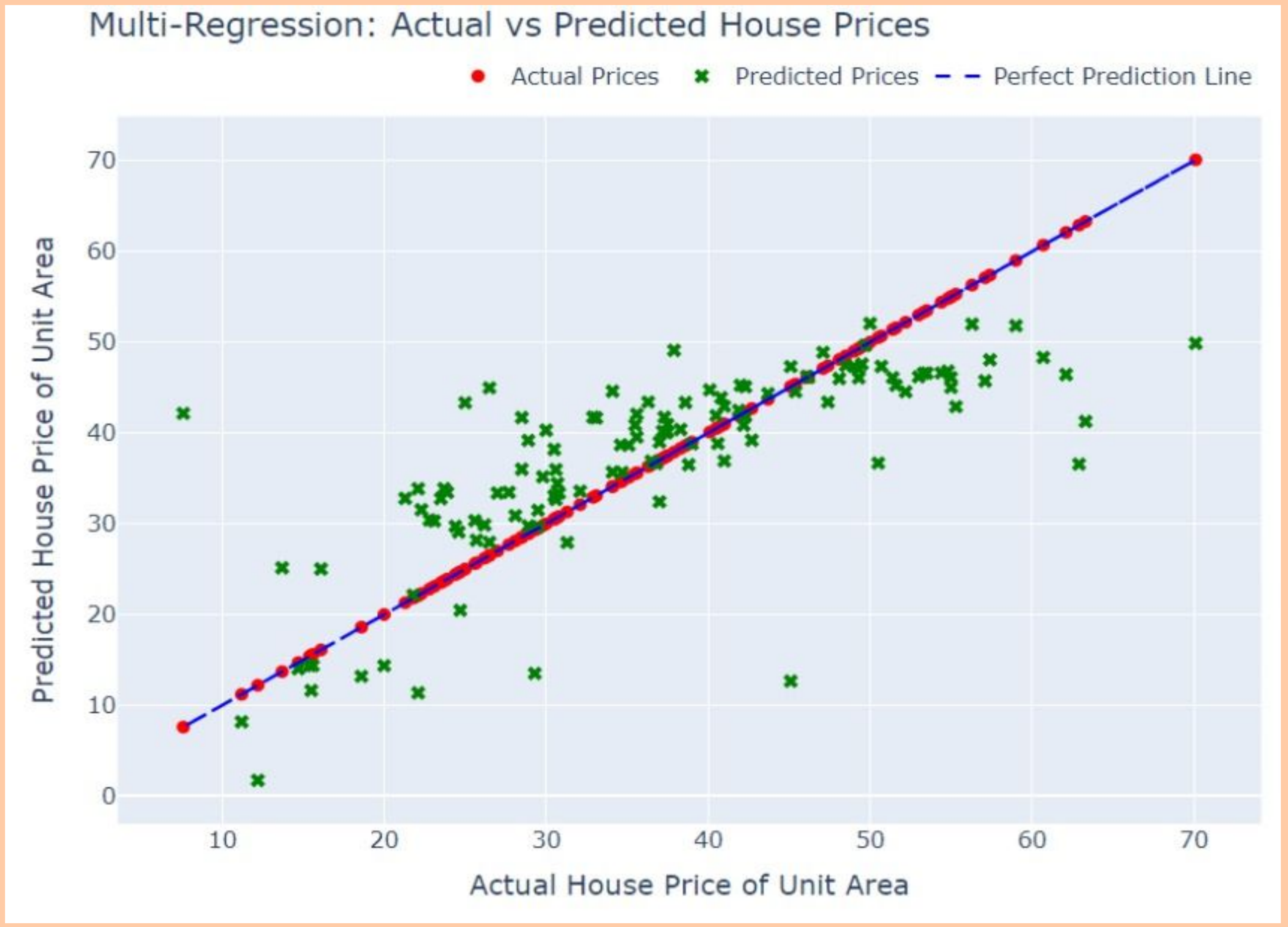
Intercept (-530.168):

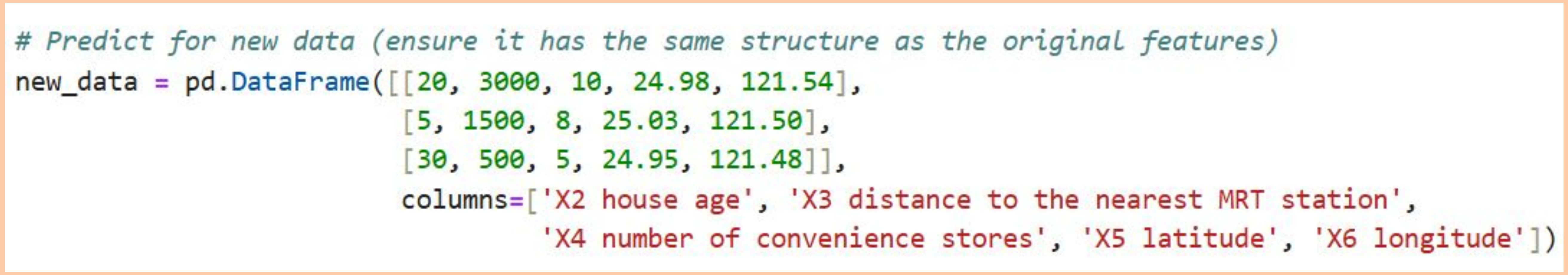
➔ This is the baseline value of house price per unit area when all independent variables (house age, MRT distance, number of convenience stores, latitude, and longitude) are zero. It's a theoretical point, not practically meaningful, but necessary for the equation.

04 Model Evaluation

Display the regression equation for each model.

➔ Chart helps visualize the performance of the regression model by showing how closely the predicted values match the actual values. Ideally, the predicted values (orange 'X' markers) should lie close to the red dashed line, indicating a well-fitted model. The more scattered the orange markers are from the line, the more errors are present in the model's predictions. The actual price is based on the dataset. The predicted price is considering all the dependent variables like these and helping the model predict the price





```python
# Predict for new data (ensure it has the same structure as the original features)
new_data = pd.DataFrame([[20, 3000, 10, 24.98, 121.54],
                         [5, 1500, 8, 25.03, 121.50],
                         [30, 500, 5, 24.95, 121.48]],
                         columns=['X2 house age', 'X3 distance to the nearest MRT station',
                                  'X4 number of convenience stores', 'X5 latitude', 'X6 longitude'])
```

Predicted House Price per unit area
37
60
36

05 Model Interpretation

Evaluate the models using R² and adjusted R², and interpret the results.

➔ **R-squared (0.5571)**:
This value indicates that approximately 55.71% of the variability in house prices is explained by the independent variables in the model, suggesting a moderate fit.

➔ **Adjusted R-squared (0.5385)**:
This value accounts for the number of predictors in the model and shows that 53.85% of the variability in house prices is explained after adjusting for the number of variables. This value is slightly lower than the R-squared, indicating that some variables might have limited predictive power.

➔ **Mean Absolute Error & Mean Squared Error:**:
The Mean Absolute Error (MAE) of 6.15 indicates that, on average, the predicted house prices are off by about $6.15 from the actual prices. The Mean Squared Error (MSE) of 74.06 suggests that the average of the squared differences between predicted and actual prices is relatively small, indicating a decent model fit. The average prediction error is relatively small and that the model is performing well in predicting house prices.

INSIGHTS

- ➔ The model suggests that proximity to MRT stations and the number of convenience stores significantly affect house prices, with location (latitude and longitude) also playing a role. House age has a smaller but still negative impact. The moderate **R-squared** values suggest that while the model explains a good portion of price variance.

- ➔ The **MAE - Mean Absolute Error** indicates that the average prediction error is relatively small and that the model is performing well in predicting house prices. The **MSE - Mean Square Error** suggests that while the model makes some predictions that are off by larger amounts (due to squaring), the average error squared is still acceptable.

Thanks for listening!