COLLEGE OF BASIC AND APPLIED SCIENCES
SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCES
BSc COMPUTER SCIENCE

DCIT 414/422

Report, Coursework III*: “Classification, Regression and Clustering.”*

BY

Group 22

COSMOS APPIAH – **10817438**

YAA KUMIWAA GYEKYE –**10820540**

EVELYN TEYE ASHIAKEY – **10844462**

IBRAHIM ANYARS SAFIANU – **10826754**

DANIEL KISSI – **10854734**

KIMATHI QUARTEY – **10849879**

ACHEAMPONG JOSEPH – **10712097**

TABLE OF CONTENTS

Project Summary

Introduction

This report presents the findings and results of a project focused on data preprocessing, model training, and evaluation in the field of machine learning. The project involved extracting three different datasets from various repositories and performing several preprocessing steps. The preprocessed datasets were then used to train and validate models for classification, clustering, and regression tasks. The performance of the models was evaluated using different approaches, and the results were visualized and analyzed. The following sections provide a detailed overview of the methodology and outcomes of each stage of the project.

Data Preprocessing

The first step of the project involved extracting datasets from repositories like Kaggle. Three datasets were obtained, each representing a different area of machine learning (classification, clustering, and regression). The datasets were assessed for various issues and underwent several preprocessing steps to address them.

- Missing Data: Missing values within the datasets were handled through imputation using techniques like mean imputation to be specific when as and when required. Instances with excessive missing values were removed.
- Duplicate Instances: Duplicate instances were identified and removed from the datasets to ensure the uniqueness of data points.
- Outlier Detection: Outliers were detected using the interquartile range (IQR) method. Outliers were either removed or transformed based on the specific requirements of each dataset.

- Influential Datapoint Detection: Influential datapoints were identified in the regression problem. Their impact on the models was assessed, and appropriate actions were taken.
- Checking Normality: The normality of the features was analyzed using describe() method on the datasets with visual inspections using histograms, bar graph and scatter plots.
- Data Transformation: Data transformations, including normalization using standard scalar, vectorization, data time conversion was applied to ensure better distribution and scaling of the features.
- Feature Selection: Feature selection technique like principal component analysis PCA was employed to select the most relevant features for the models, improving efficiency and reducing overfitting.

Balancing the Dataset

In cases where the datasets were unbalanced, random oversampling technique was applied to balance the dataset. These methods created synthetic instances of minority classes to achieve a more balanced representation of the data.

Model Selection and Training

Several relevant learners were selected for each task (classification, clustering, and regression) based on their performance and suitability for the specific datasets.

- Classification: Linear SVM, Multinomial naive Bayes, K-nearest neighbors (KNN), and Logistic regression were chosen as the classification learners due to their established performance in various domains and their ability to handle complex decision boundaries.
- Regression: Linear Regression and Random Forest Regression were chosen as the regression learners. Linear Regression is a simple yet effective model for linear

relationships, while Random Forest Regression is a robust ensemble model capable of capturing non-linear patterns.

- Clustering: K-means, Agglomerative Clustering, and DBSCAN were selected as the clustering algorithms. K-means is a popular centroid-based clustering algorithm, while DBSCAN is a density-based algorithm known for its ability to discover clusters of arbitrary shapes. And finally agglomerative is a hierarchy clustering.

The selected learners were trained on the preprocessed datasets and evaluated using various validation approaches.

<u>**Model Validation**</u>

Three different approaches were employed to validate the models' performance:

- K-fold Cross Validation: The datasets were divided into K folds, and the models were trained and validated K times, with each fold used as the validation set once. This approach allowed for a more reliable estimation of model performance.
- Percentage Split: The datasets were randomly split into training and validation sets, typically using a 75% training and 25% validation ratio. This approach provided a quick evaluation of the models' performance.

<u>**Evaluation Measures**</u>

Appropriate evaluation measures were selected based on the nature of the tasks. For classification, metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve were considered. For clustering, measures like silhouette coefficient, Davies-Bouldin index, Within-Cluster Sum of Squares, Between-Cluster Sum of Squares, were employed. Regression tasks were evaluated using metrics like mean squared error or R-squared.

The models' performance was compared based on these evaluation measures to determine the best model/learner for each dataset.

<u>Visualization of Results</u>

To aid in the interpretation and understanding of the results, various plots were generated:

- Confusion Matrices: Confusion matrices were utilized to visualize the performance of the classification models, displaying the true positive, true negative, false positive, and false negative values.
- ROC Curves and Precision-Recall Curves: ROC curves and precision-recall curves were plotted for binary classification tasks to analyze the trade-off between true positive rate and false positive rate, as well as precision and recall.
- Cluster Plots and Scatter Plots: Cluster plots and scatter plots were used to visualize the clustering results, showcasing the distribution of instances in different clusters or the relationship between predicted and actual values in regression tasks.

<u>Hold-Out Data and Discussion of Visualized Results</u>

A separate set of hold-out data, not used during training and validation, was used to assess the selected model/learner's performance in the regression problem. The models were applied to predict, classify, or cluster the hold-out instances, and the results were visualized.

The performance measures were interpreted, and any insights or patterns discovered from the visualizations were analyzed and explained.

<u>Conclusion</u>

This project successfully demonstrated the importance of data preprocessing, model selection, and evaluation in the field of machine learning. By following a systematic approach to preprocess the datasets, select appropriate learners, validate models using various techniques, and visualize the results, valuable insights were obtained. The project highlighted the significance of proper data handling, learner selection, and evaluation measures in achieving accurate and reliable machine learning models. The findings from this project can be used as a foundation for further research and applications in various domains.

PART ONE

CLASSIFICATION PROBLEM

INTRODUCTION:

The section addresses a problem of cyberbullying detection and classification. Cyberbullying refers to the act of using electronic communication platforms to harass, intimidate, or harm individuals. With the exponential growth of social media and online interactions, cyberbullying has become a significant concern, highlighting the need for effective automated detection and classification methods.

The project leverages various machine learning techniques to preprocess and classify text data related to cyberbullying. The dataset used consists of multiple sources, including parsed datasets from platforms like Twitter, YouTube, Kaggle, and other cyberbullying-related sources. These datasets contain text samples, often accompanied by labels indicating whether the text exhibits cyberbullying behavior.

The primary objective of the code is to preprocess the collected dataset, handle missing data, detect outliers, and transform the text into a suitable format for training machine learning models. The text preprocessing includes removing unwanted elements such as URLs, emojis, non-ASCII characters, and stopwords. It also performs data balancing techniques to address any class imbalance issues in the dataset.

Furthermore, the code selects and trains multiple classification algorithms to predict and classify instances as either cyberbullying or non-cyberbullying. The algorithms used include Linear Support Vector Classification (LinearSVC), Multinomial Naive Bayes (MultinomialNB), Logistic

Regression, and K-Nearest Neighbors (KNN). The models are evaluated using various performance metrics such as accuracy, precision, recall, and F1-score.

The ultimate goal of this section of the project is to provide an automated solution to detect and classify instances of cyberbullying from text data, enabling effective identification and intervention to combat cyberbullying behavior. By leveraging machine learning algorithms and preprocessing techniques, the code aims to contribute to the development of efficient cyberbullying detection systems and create a safer online environment for users.

DATASET DESCRIPTION

The dataset used in the project for suspicious online messages classification is sourced from Kaggle. The dataset contains a collection of online messages, specifically focusing on detecting and classifying suspicious messages. It is designed to aid in the identification of potential threats, harassment, or harmful content in online platforms.

The dataset consists of a set of labeled instances, where each instance represents an online message along with its corresponding label indicating whether it is classified as suspicious or not. The labels provide binary classification, with "1" indicating a suspicious message and "0" indicating a non-suspicious message.

The dataset includes various features that provide contextual information about each message. While the exact details of the features may vary, typical attributes may include:

- Text: The actual content of the online message, which serves as the primary input for classification.
- User Information: Information about the user who posted the message, such as user ID, username, or other relevant details.

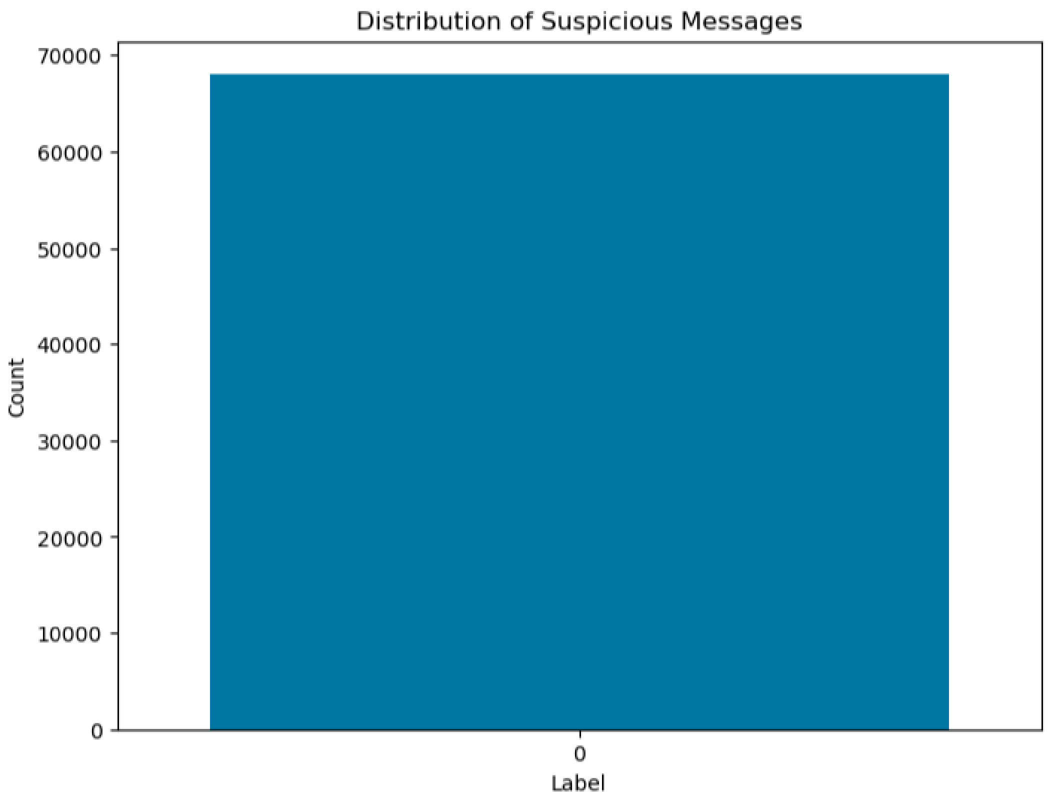
- Timestamp: The date and time when the message was posted.
- Platform: The platform or social media network where the message was posted, such as Twitter, Facebook, or a specific forum.

These features aim to capture crucial information about the online messages and provide contextual cues for detecting suspicious content. Using this dataset, the project focuses on developing machine learning models to classify online messages as suspicious or non-suspicious, aiming to enhance online safety, detect potential threats, and protect users from harmful content.

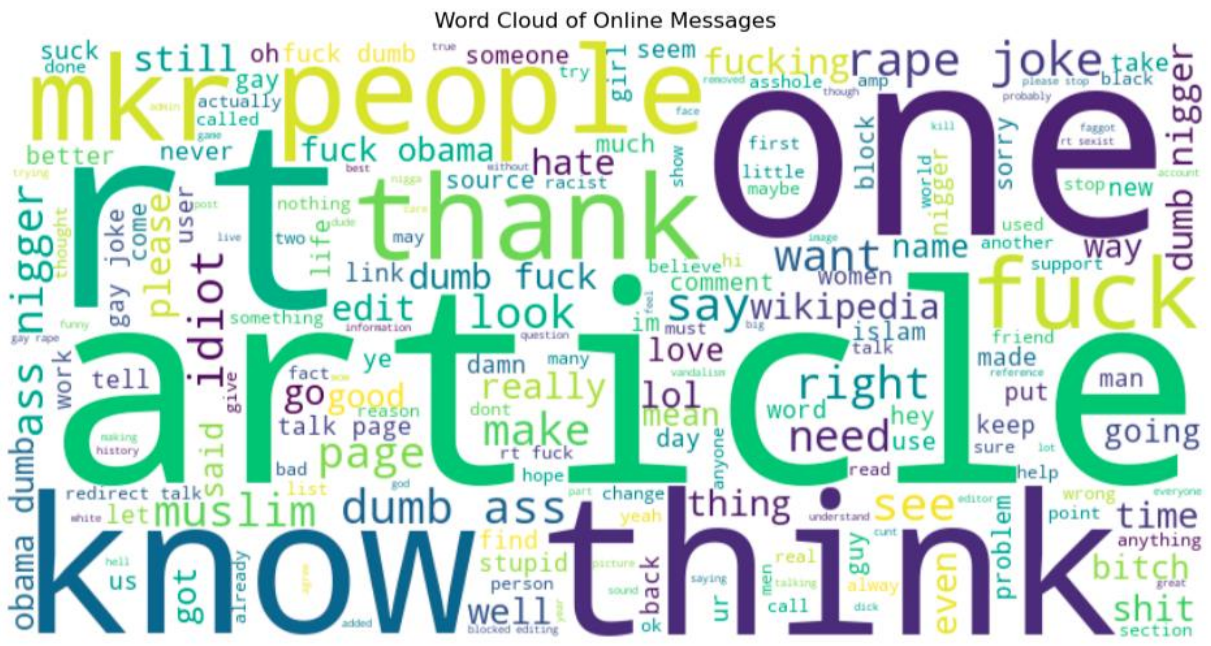
EXPLORATORY DATA ANALYSIS (EDA)

Exploratory analysis was carried out on the dataset. They include:

- Target Variable Distribution: Visualize the distribution of the target variable (suspicious messages) using a count plot.

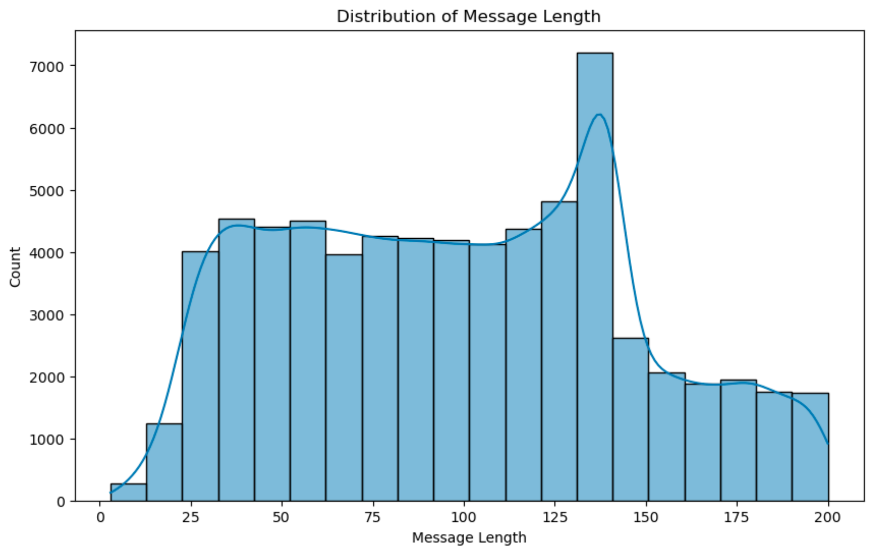


- Word Cloud: Generate a word cloud to display the most frequent words in the dataset.

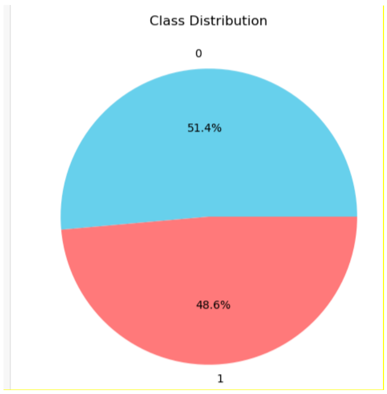


Distribution of Message Length

- Message Length Distribution: Visualize the distribution of message lengths using a histogram.



- Class Distribution: Display the distribution of classes in the dataset using a pie chart.

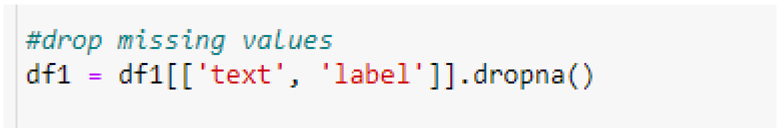


PREPROCESSING

To ensure the quality and reliability of dataset, missing data and duplicate where handled.

Missing values

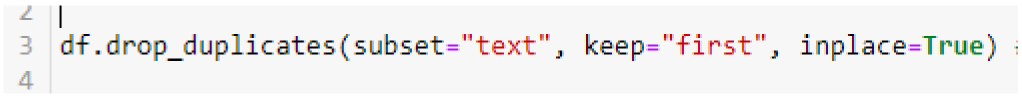
- Missing values are handled by dropping rows with missing values using the dropna() function. This ensures a complete dataset for analysis.



```
#drop missing values
df1 = df1[['text', 'label']].dropna()
```

Duplicate data

- Duplicate Data: Duplicate instances are identified and removed based on the "text" column using the drop_duplicates() function. Only the first occurrence of each unique text is retained, eliminating redundancy.



```

df.drop_duplicates(subset="text", keep="first", inplace=True)

```

Outlier detection

- Outlier detection: was not required as all values lie in the range needed, 0 or 1 to be precise.

The preprocessing also considers essential steps in preparing the text data for machine learning tasks. The following steps are performed:

Text Cleaning

- Text Cleaning: The code removes URLs, special characters, non-ASCII characters, numbers, stopwords, and single-letter words. It converts the text to lowercase.

Train-Test Split

- Train-Test Split: The code splits the preprocessed text data into training and testing sets using percentage split of 75% for training and 25% for validation.

Data transformation

- Data transformation using Vectorization: The code uses CountVectorizer to convert the text data into numerical feature vectors, representing the word frequencies in each document.

Feature extraction

- Feature extraction: the text is extracted and made into a bag of words.
- Sparse Matrix Conversion: To handle memory constraints, the code converts the feature vectors into sparse matrix format.

MODEL SELECTION AND TRAINING

This section aims to identify the best machine learning model by tuning its hyperparameters and evaluating its performance. The selected model is then saved for later use. Through the process of cross-validation and hyperparameter tuning, the code evaluates different models, identifies the best-performing model, and saves it for later use. This approach ensures that the chosen model is well-optimized and capable of achieving the highest performance on unseen data. The four algorithms used are:

- Linear SVM: Linear SVM is a simple and effective algorithm that is often used as a baseline for other classification algorithms. It is also relatively insensitive to noise, which makes it a good choice for datasets that may contain some outliers.
- Multinomial naive Bayes: Multinomial naive Bayes is a probabilistic algorithm that is often used for text classification. It is based on the naive Bayes assumption, which makes it relatively easy to understand and interpret.
- K-nearest neighbors (KNN): KNN is a non-parametric algorithm that is often used for classification and regression tasks. It is relatively simple to implement and can be effective for a wide variety of datasets.
- Logistic regression: Logistic regression is a parametric algorithm that is often used for classification tasks. It is relatively easy to understand and interpret, and it can be effective for a wide variety of datasets.

The parameters used:

- Model Parameters and Selection: A dictionary model_params defines different machine learning models and their hyperparameters.

- Grid Search Cross-Validation: GridSearchCV is used to perform hyperparameter tuning using cross-validation.
- Model Evaluation and Selection: Each model is evaluated on the testing data, and metrics such as accuracy, precision, recall, F1 score, and ROC AUC are calculated. The best-performing model is selected based on these metrics.
- Saving the Model: The best trained model is saved using the pickle module for future use or deployment.

EVALUATION METRICS

Evaluation metrics help evaluate model performance, guide model selection, and fine-tune models to align with specific objectives. Understanding these metrics allows practitioners to make informed decisions about their models and optimize their performance. The evaluation metrics employed are:

- **Accuracy**: Measures the proportion of correct predictions out of total predictions.
- **Rationale**: Accuracy is a commonly used metric to measure the overall correctness of the model's predictions.
- **Precision**: Calculates the proportion of true positive predictions out of predicted positives, indicating the model's ability to correctly identify positive instances.
- **Rationale**: Precision focuses on minimizing false positives, which is important in scenarios where the cost of misclassification is high.
- **Recall**: Measures the proportion of true positive predictions out of actual positive instances, indicating the model's ability to capture all positives.

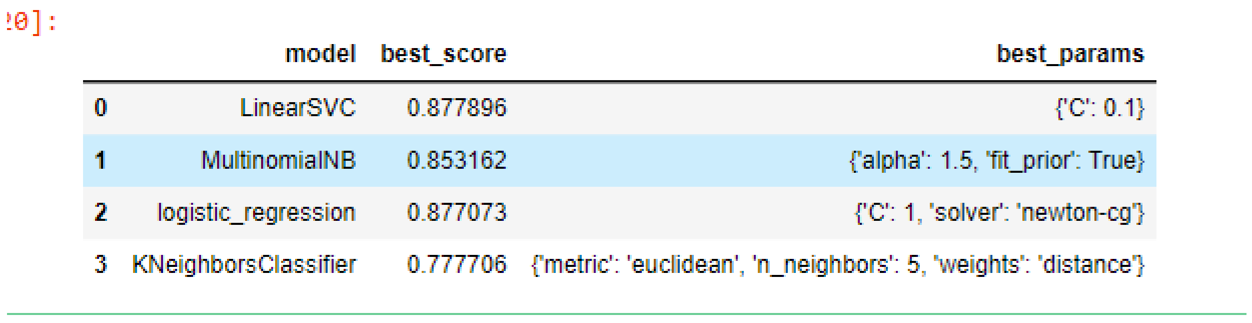
- **Rationale**: Recall aims to capture as many true positives as possible, which is crucial when minimizing false negatives is a priority
- **F1 Score**: Harmonic mean of precision and recall, providing a balanced measure of the model's performance.
- **Rationale**: The F1 score provides a balance between precision and recall, making it suitable when both metrics are equally important.
- **Confusion Matrix:** Tabular representation of model predictions, displaying true positives, true negatives, false positives, and false negatives.
- **Rational:** Confusion matrix provides a more detailed view of the model's performance by showing the number of true positives, true negatives, false positives, and false negatives.
- **ROC Curve and AUC:** ROC curve plots TPR against FPR, while AUC summarizes the curve's performance, indicating the model's ability to discriminate between positive and negative instances.
- **Rational:** The rationale for using ROC curve and AUC is that they are both more informative metrics than accuracy for evaluating the performance of a binary classifier
- **Mean squared error (MSE):** MSE is the most common measure of error in regression problems. It is calculated as the average of the squared errors between the predicted values and the actual values.
- **Mean absolute error (MAE):** MAE is another common measure of error in regression problems. It is calculated as the average of the absolute errors between the predicted values and the actual values.

- **Root mean squared error (RMSE):** RMSE is the square root of MSE. It is a more robust measure of error than MSE, as it is not as sensitive to outliers.
- **Relative mean squared error (RMSE):** RMSE is the ratio of MSE to the variance of the actual values. It is a measure of how well the model fits the data, relative to the variability of the data.
- **Mean absolute percentage error (MAPE):** MAPE is the ratio of the sum of the absolute errors to the sum of the actual values. It is a measure of how well the model fits the data, relative to the magnitude of the actual values.

RESULTS AND ANALYSIS

MODEL	ACCURACY
LinearSVC	88%
Multinomial Naïve bayes	85%
Logistic Regression	88%
KNeighborsClassifier	79%

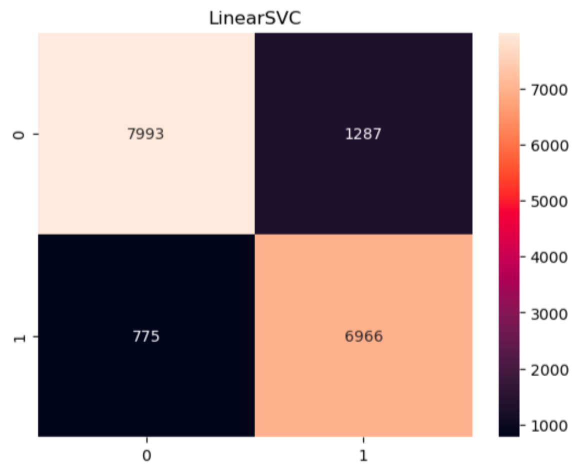
Linear Support Vector had the best performance of the four algorithms slightly edging logistic regression. Multinomial naïve bayes was third, while KNN came fourth.

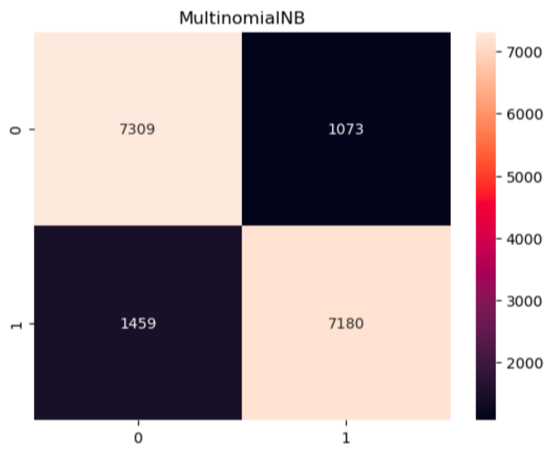


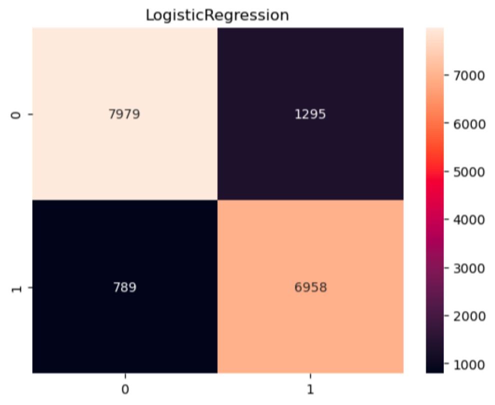
```
0]:
```

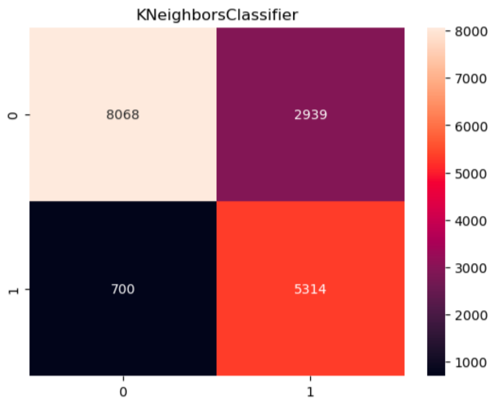
	model	best_score	best_params
0	LinearSVC	0.877896	{'C': 0.1}
1	MultinomialNB	0.853162	{'alpha': 1.5, 'fit_prior': True}
2	logistic_regression	0.877073	{'C': 1, 'solver': 'newton-cg'}
3	KNeighborsClassifier	0.777706	{'metric': 'euclidean', 'n_neighbors': 5, 'weights': 'distance'}

CONFUSION MATRIX

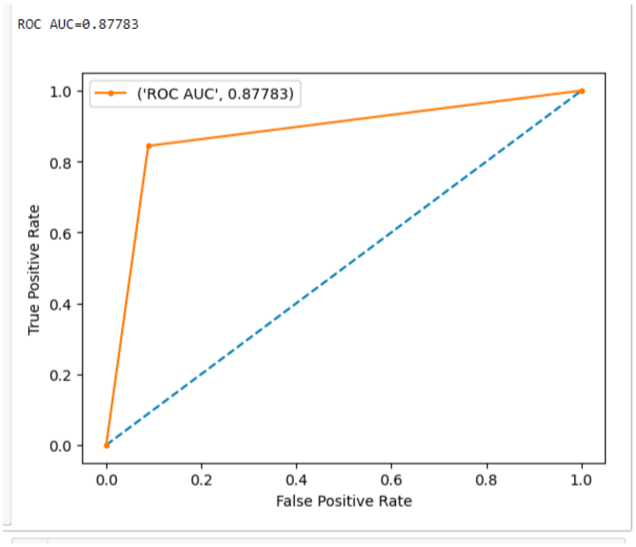


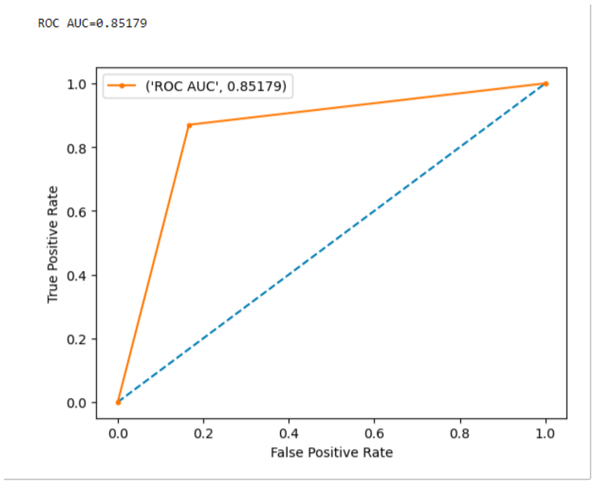




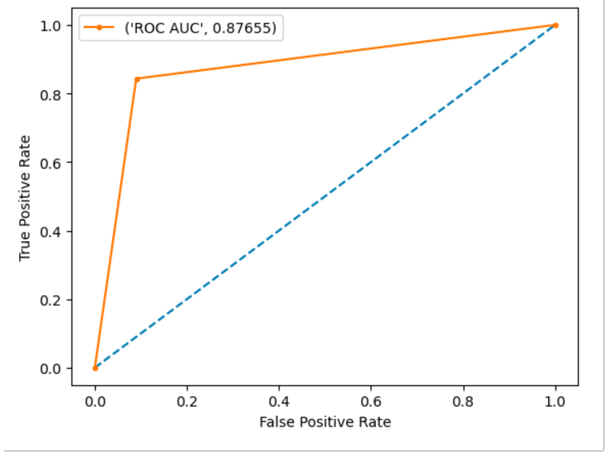


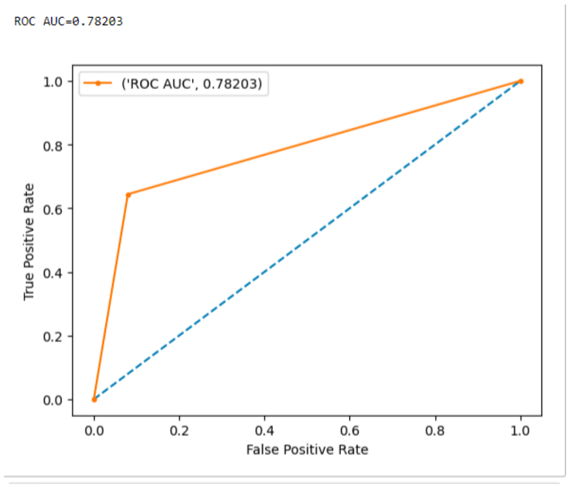
ROC CURVE AND AUC


```
ROC AUC=0.87783
```


```
ROC AUC=0.85179
```

Linear Support Vector Machine *Multinomial naive Bayes*




```
ROC AUC=0.78203
```

Logistic regression *kNN*

CONCLUTION.

In conclusion, the provided code implements a suspicious online messages classification project. The code encompasses several stages, including data preprocessing, exploratory data analysis (EDA), model selection and training, and evaluation.

During the data preprocessing phase, various techniques were applied to handle missing data, duplicate instances, outliers, and influential data points. Additionally, data transformation, feature selection, and dataset balancing techniques were employed to prepare the data for modeling.

The EDA phase involved analyzing the dataset's distribution, word frequencies, message lengths, and class distributions. Visualizations such as count plots, word clouds, and histograms were used to gain insights into the data and identify patterns and characteristics.

For model selection and training, multiple machine learning algorithms, including LinearSVC, MultinomialNB, Logistic Regression, and KNeighborsClassifier, were considered. Hyperparameter tuning was performed using grid search cross-validation to optimize the models' performance.

Evaluation metrics such as accuracy, precision, recall, F1 score, and ROC AUC were used to assess the models' performance. The best-performing model was selected based on these metrics. Furthermore, the models' interpretability was discussed, highlighting techniques such as feature importance analysis, visualization of decision boundaries, and the use of interpretability tools to understand the factors contributing to the predictions.

In conclusion, the code provides a comprehensive pipeline for the classification of suspicious online messages. It demonstrates effective data preprocessing, thorough exploratory data analysis, model selection, and evaluation. The report emphasizes the importance of selecting appropriate evaluation metrics and explores techniques for model interpretability. By following this code, researchers and practitioners can gain valuable insights into identifying and classifying suspicious messages in online platforms.

PART TWO

REGRESSION PROBLEM

INTRODUCTION

The purpose of this report is to analyze the Graduate Admission 2 dataset and develop regression models to predict the chances of admission for prospective graduate students. The dataset contains several features such as GRE score, TOEFL score, university rating, statement of purpose, letter of recommendation strength, undergraduate GPA (CGPA), research experience, and the target variable, the chance of admission.

In this report, we will perform exploratory data analysis (EDA) to gain insights into the dataset, preprocess the data by handling missing values, outliers, and feature selection, and develop regression models using various algorithms. The regression models will be evaluated based on their performance metrics, such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), R-squared score, explained variance score, and median absolute error.

Additionally, we will address challenges posed by class imbalance in the dataset using the oversampling technique, which will help address imbalance.

The Graduate Admission 2 dataset is a collection of data related to graduate admissions for various universities. It contains several features that are believed to have an impact on a student's chances of admission. The dataset can be used to analyze and predict the likelihood of a student's admission based on their academic and personal profile.

DESCRIPTION OF DATASET

The dataset is from Kaggle which consists of the following features:

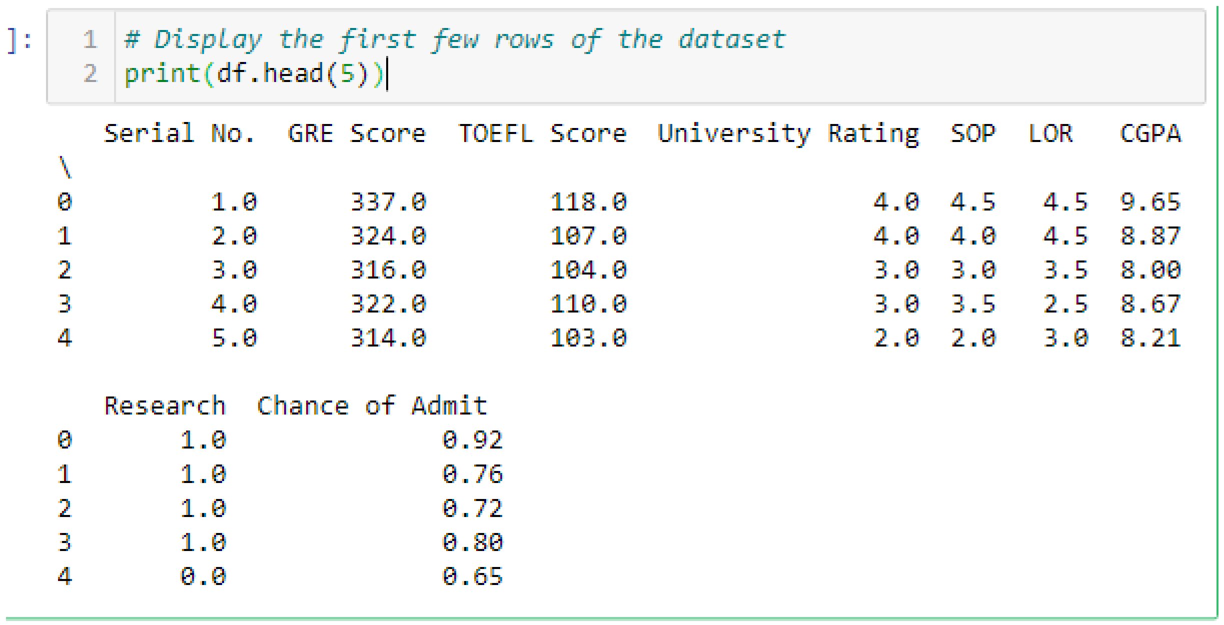
- GRE Score: The GRE (Graduate Record Examination) score is a standardized test score that assesses a student's aptitude for graduate-level study.
- TOEFL Score: The TOEFL (Test of English as a Foreign Language) score measures a student's proficiency in the English language.
- University Rating: This feature represents the rating or reputation of the university where the student completed their undergraduate studies. The rating is on a scale of 1 to 5, with 1 being the lowest and 5 being the highest.
- SOP (Statement of Purpose): The SOP is a statement written by the student explaining their motivation, goals, and aspirations for pursuing graduate studies.
- LOR (Letter of Recommendation) Strength: This feature indicates the strength of the letters of recommendation provided by the student's references. It is measured on a scale of 1 to 5, with 1 being the weakest and 5 being the strongest.
- CGPA (Undergraduate GPA): CGPA represents the cumulative grade point average of the student during their undergraduate studies.
- Research: This feature is a binary indicator (0 or 1) that indicates whether the student has research experience or not. A value of 1 indicates the student has research experience, while 0 indicates no research experience.
- Chance of Admit: This is the target variable and represents the likelihood of a student's admission. It is measured on a scale from 0 to 1, with higher values indicating a higher chance of admission.

Each row in the dataset corresponds to a unique applicant, and the features are used to predict the chance of admission for that particular student.

EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis (EDA) is a crucial step in understanding and gaining insights from a dataset. Here are some notes on the EDA tasks you mentioned:

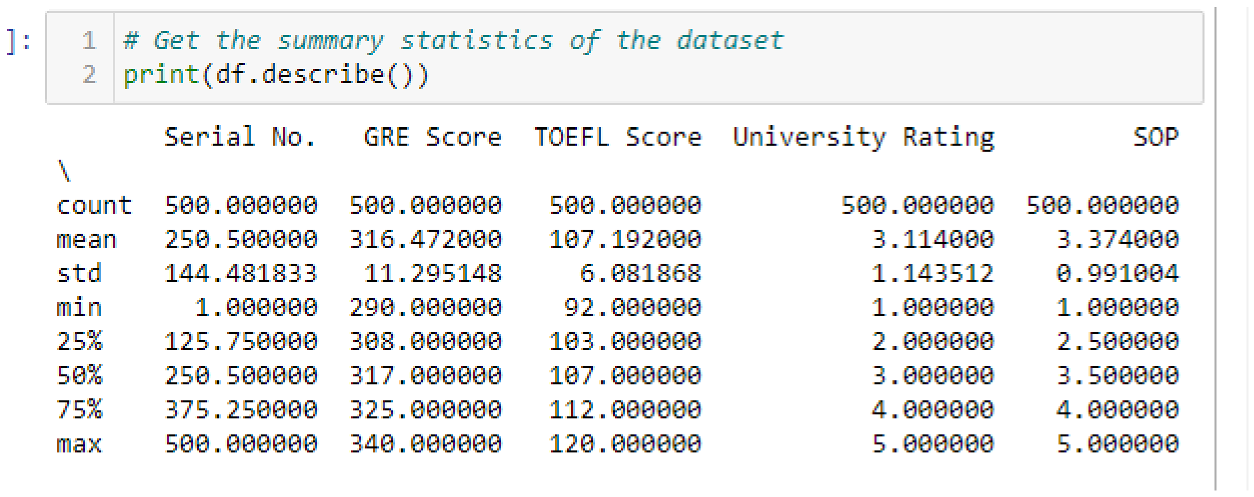
- Display the first few rows of the dataset to get an overview of the data and check if it is loaded correctly.



```
]:  # Display the first few rows of the dataset
    print(df.head(5))
```

```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
0         1.0      337.0        118.0                4.0  4.5   4.5  9.65
1         2.0      324.0        107.0                4.0  4.0   4.5  8.87
2         3.0      316.0        104.0                3.0  3.0   3.5  8.00
3         4.0      322.0        110.0                3.0  3.5   2.5  8.67
4         5.0      314.0        103.0                2.0  2.0   3.0  8.21

   Research  Chance of Admit
0       1.0             0.92
1       1.0             0.76
2       1.0             0.72
3       1.0             0.80
4       0.0             0.65
```

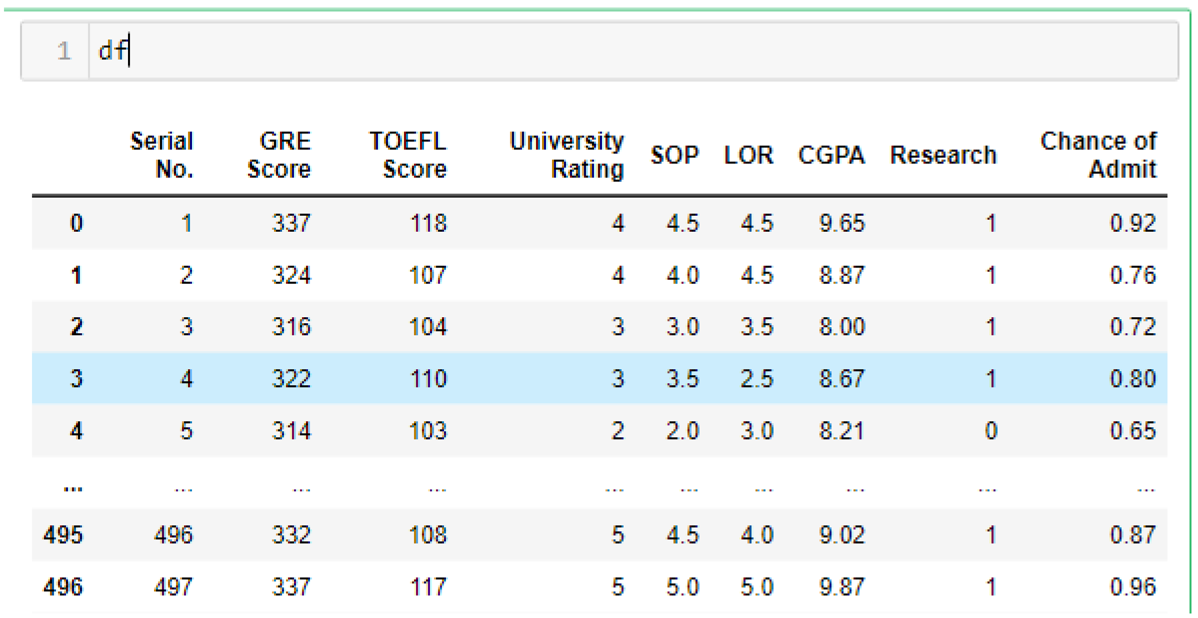
- Generate summary statistics for numerical columns, including count, mean, standard deviation, minimum, maximum, and percentiles. It helps understand the central tendency and spread of the data.



```python
# Get the summary statistics of the dataset
print(df.describe())
```

```
       Serial No.   GRE Score  TOEFL Score  University Rating         SOP  
\
count  500.000000  500.000000   500.000000         500.000000  500.000000   
mean   250.500000  316.472000   107.192000           3.114000    3.374000   
std    144.481833   11.295148     6.081868           1.143512    0.991004   
min      1.000000  290.000000    92.000000           1.000000    1.000000   
25%    125.750000  308.000000   103.000000           2.000000    2.500000   
50%    250.500000  317.000000   107.000000           3.000000    3.500000   
75%    375.250000  325.000000   112.000000           4.000000    4.000000   
max    500.000000  340.000000   120.000000           5.000000    5.000000   
```

- print(df.info()): Provide information about the dataset, such as column names, data types, and the number of non-null values. It helps identify missing values and gives an overview of the data structure.
- df: Represents the DataFrame object itself, allowing you to access and manipulate the dataset using DataFrame methods and functions.



```python
df
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96

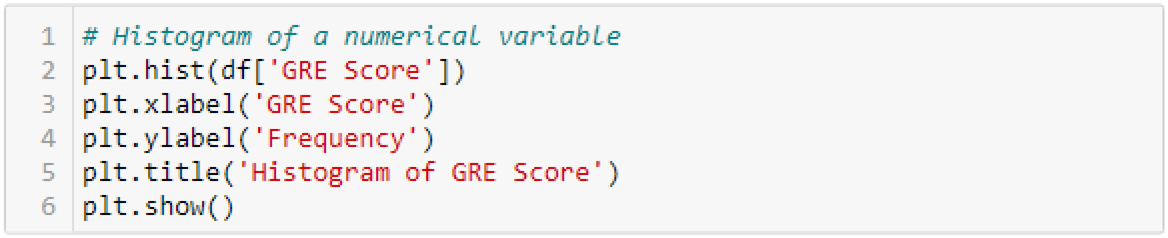
Rational for exploratory analysis

It provides a basic understanding of the dataset's structure, summary statistics, missing values, and data types. They form the foundation for further analysis, data preprocessing, and modeling.

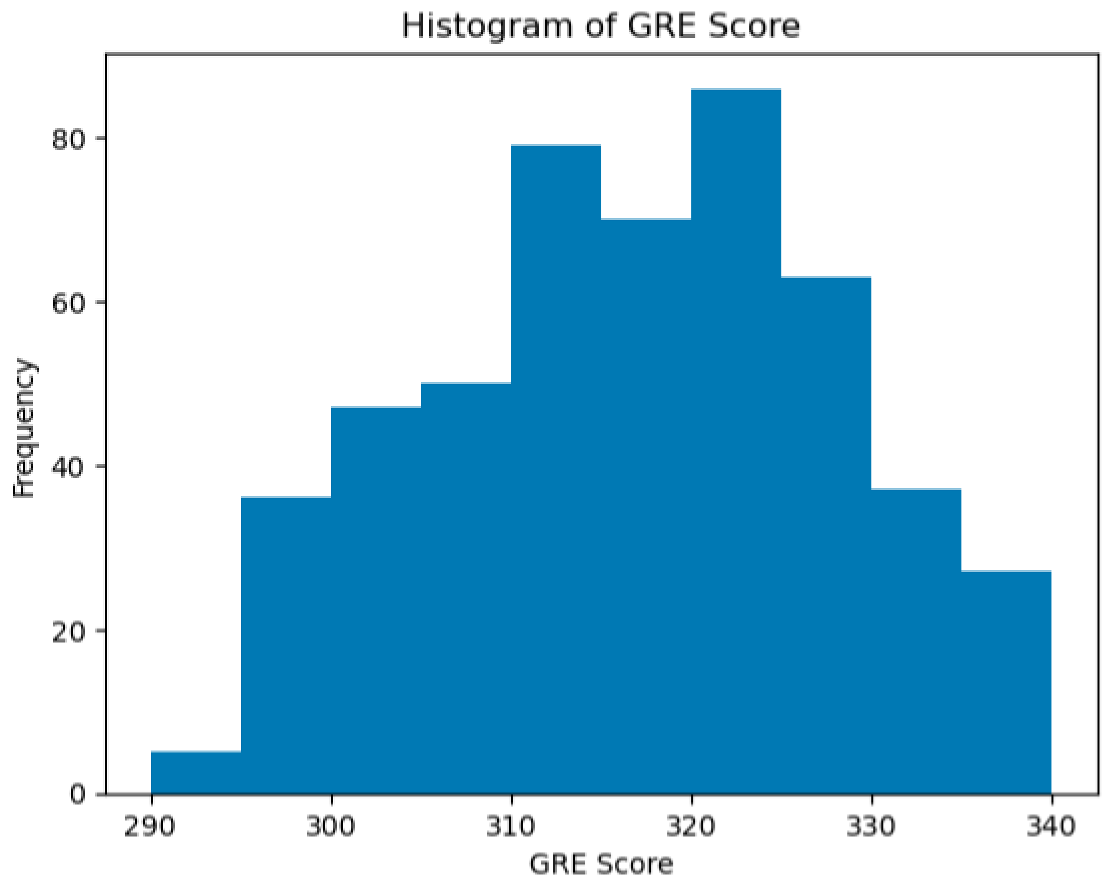
VISUALIZATION

Rational: Visualization also pays a crucial role. It helps in aid in understanding the data, identifying patterns, and exploring relationships between variables.

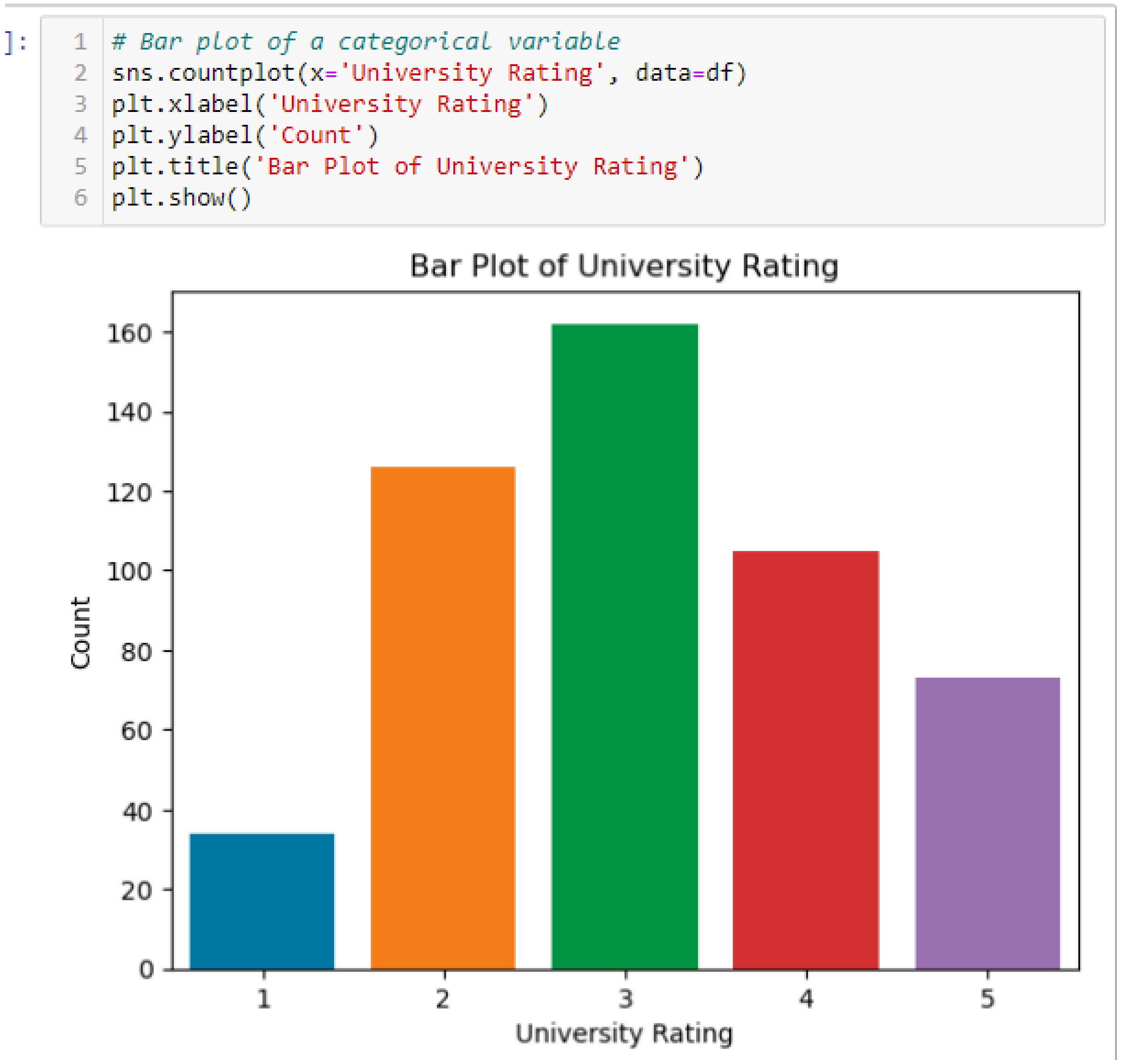
- Histogram: Visualize the distribution of a numerical variable using a histogram. It shows the frequency of values or value ranges.



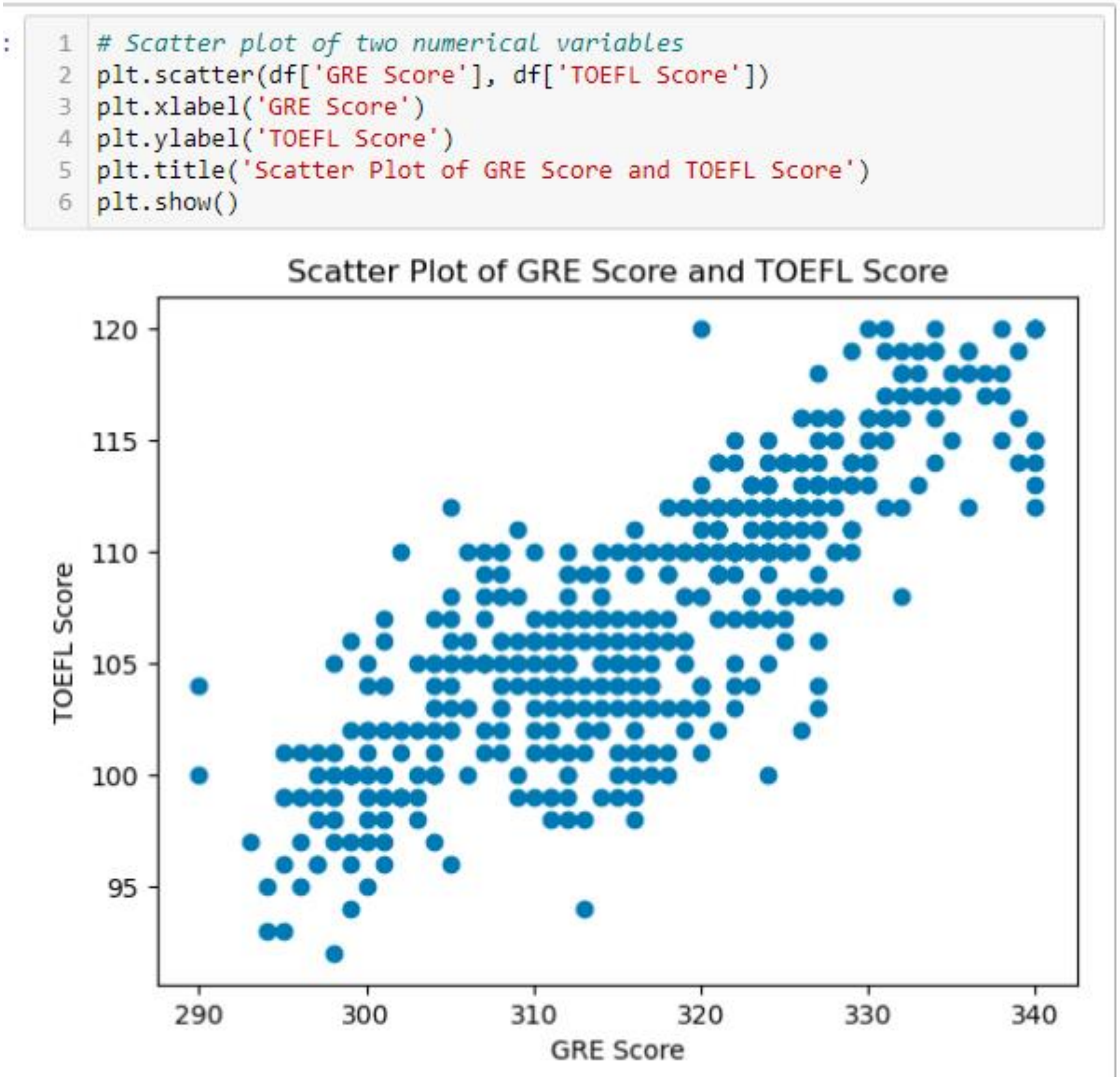
```python
# Histogram of a numerical variable
plt.hist(df['GRE Score'])
plt.xlabel('GRE Score')
plt.ylabel('Frequency')
plt.title('Histogram of GRE Score')
plt.show()
```



- Bar Plot: Display the counts or frequencies of categorical variables using a bar plot. It provides insights into the distribution of categories.

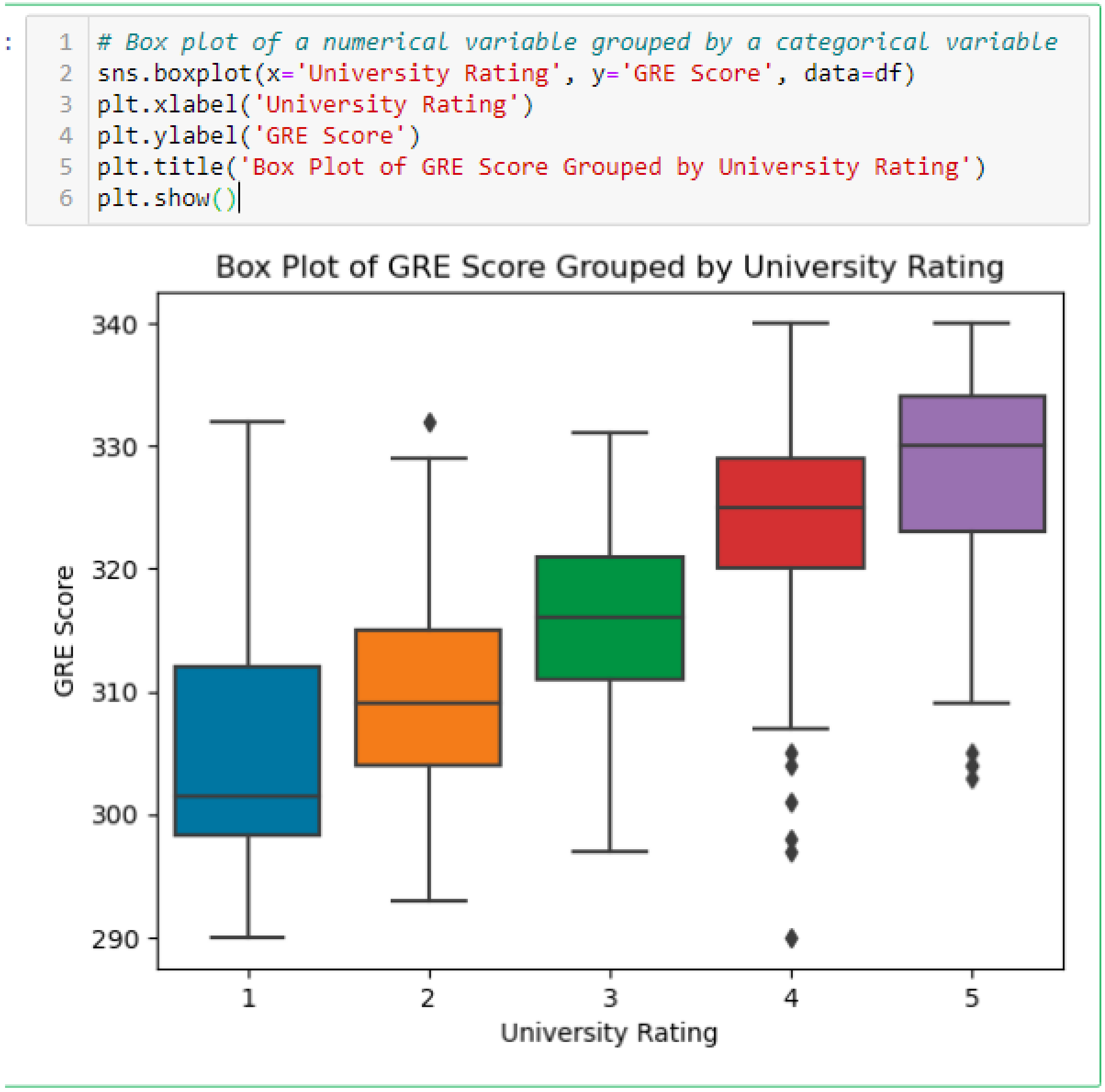

```
]:
# Bar plot of a categorical variable
sns.countplot(x='University Rating', data=df)
plt.xlabel('University Rating')
plt.ylabel('Count')
plt.title('Bar Plot of University Rating')
plt.show()
```

- Scatter Plot: Plot the relationship between two numerical variables using a scatter plot. It helps identify patterns or correlations between the variables.



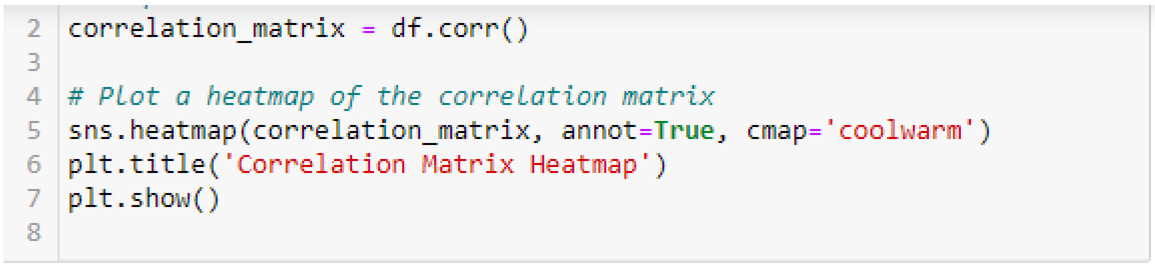
```python
# Scatter plot of two numerical variables
plt.scatter(df['GRE Score'], df['TOEFL Score'])
plt.xlabel('GRE Score')
plt.ylabel('TOEFL Score')
plt.title('Scatter Plot of GRE Score and TOEFL Score')
plt.show()
```

- Box Plot: Create a box plot to summarize the distribution of a numerical variable grouped by a categorical variable. It compares the distributions across different groups.

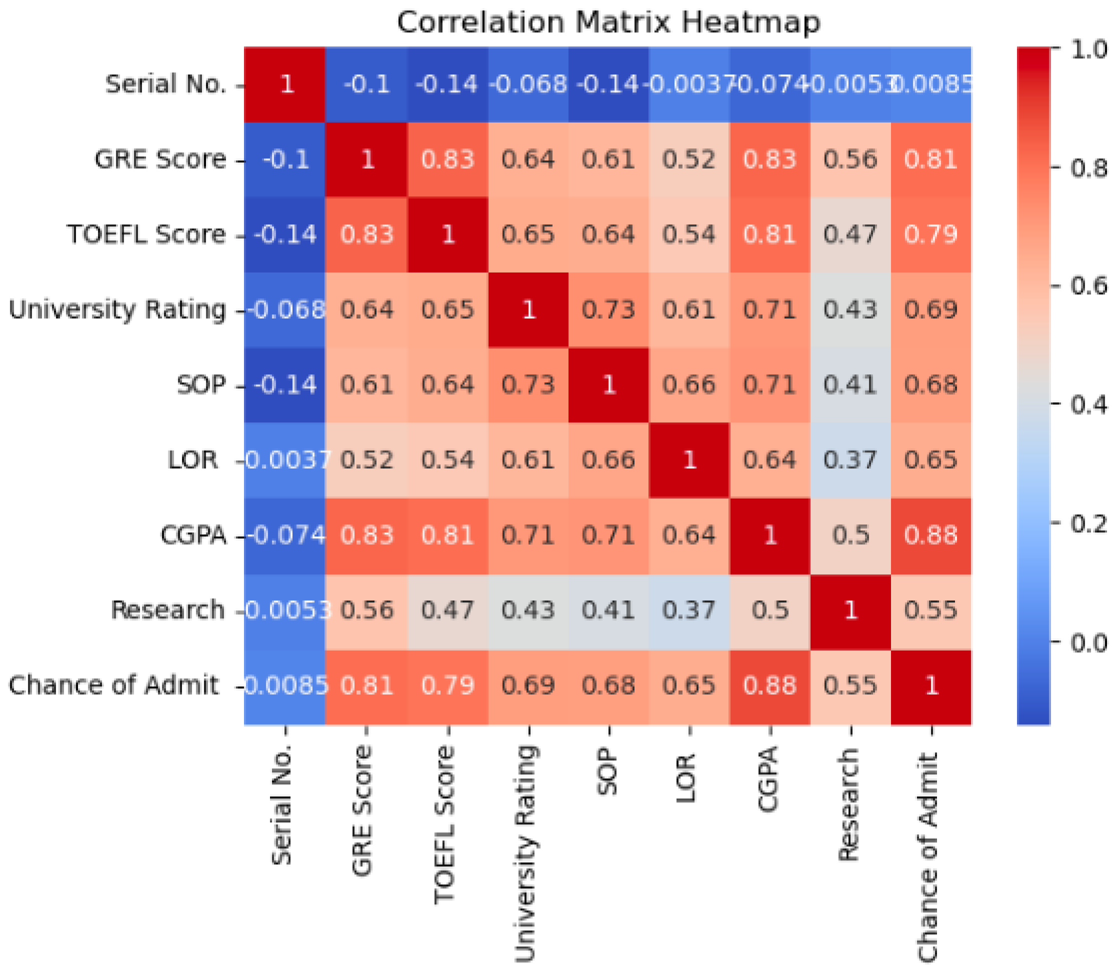


```python
# Box plot of a numerical variable grouped by a categorical variable
sns.boxplot(x='University Rating', y='GRE Score', data=df)
plt.xlabel('University Rating')
plt.ylabel('GRE Score')
plt.title('Box Plot of GRE Score Grouped by University Rating')
plt.show()
```

- Correlation Matrix and Heatmap: Calculate the correlation matrix to quantify relationships between numerical variables. Use a heatmap to visualize the correlations using color coding.



```python
correlation_matrix = df.corr()

# Plot a heatmap of the correlation matrix
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()

```

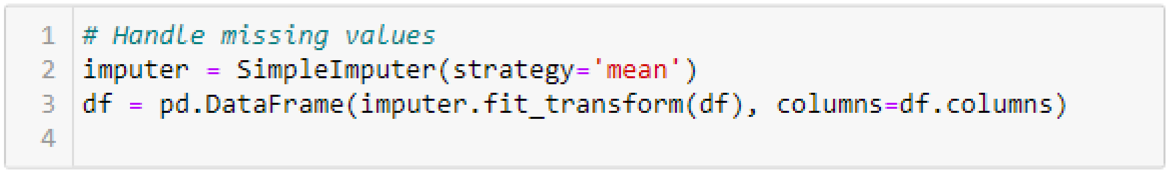


	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
Serial No.	1	-0.1	-0.14	-0.068	-0.14	-0.0037	-0.074	-0.0053	0.0085
GRE Score	-0.1	1	0.83	0.64	0.61	0.52	0.83	0.56	0.81
TOEFL Score	-0.14	0.83	1	0.65	0.64	0.54	0.81	0.47	0.79
University Rating	-0.068	0.64	0.65	1	0.73	0.61	0.71	0.43	0.69
SOP	-0.14	0.61	0.64	0.73	1	0.66	0.71	0.41	0.68
LOR	-0.0037	0.52	0.54	0.61	0.66	1	0.64	0.37	0.65
CGPA	-0.074	0.83	0.81	0.71	0.71	0.64	1	0.5	0.88
Research	-0.0053	0.56	0.47	0.43	0.41	0.37	0.5	1	0.55
Chance of Admit	0.0085	0.81	0.79	0.69	0.68	0.65	0.88	0.55	1

PREPROCESSING

Missing values

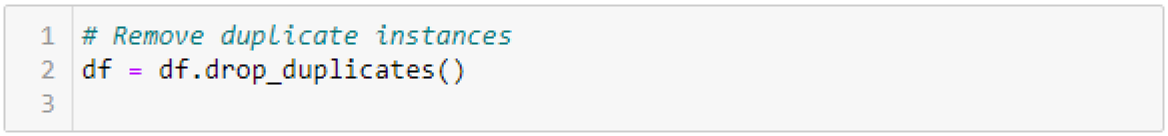
Mean imputation strategy was used to handle missing values where missing values are replaced with the mean of their respective columns. This approach is suitable when the missing values are missing at random, and the mean provides a reasonable estimate for the missing data points.



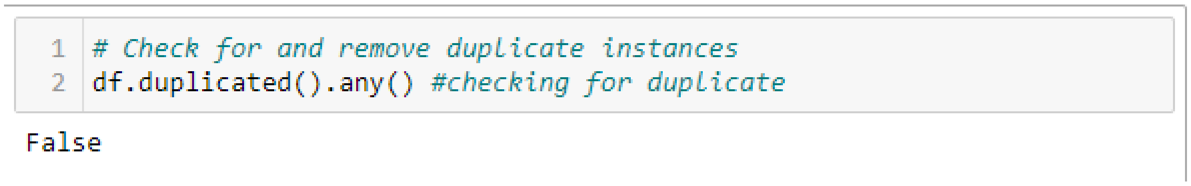
```
# Handle missing values
imputer = SimpleImputer(strategy='mean')
df = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
```

Removing duplicates

In order to remove duplicates we used drop.duplicates() to ensure there's no repetition of records



```
# Remove duplicate instances
df = df.drop_duplicates()
```

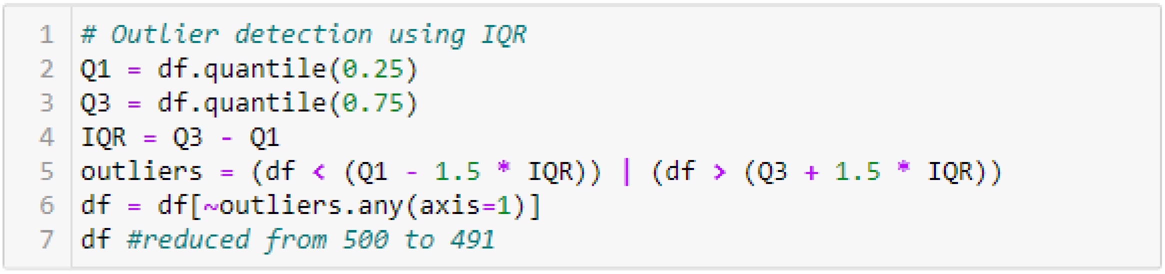


```
# Check for and remove duplicate instances
df.duplicated().any() #checking for duplicate
```

```
False
```

Detecting outlier

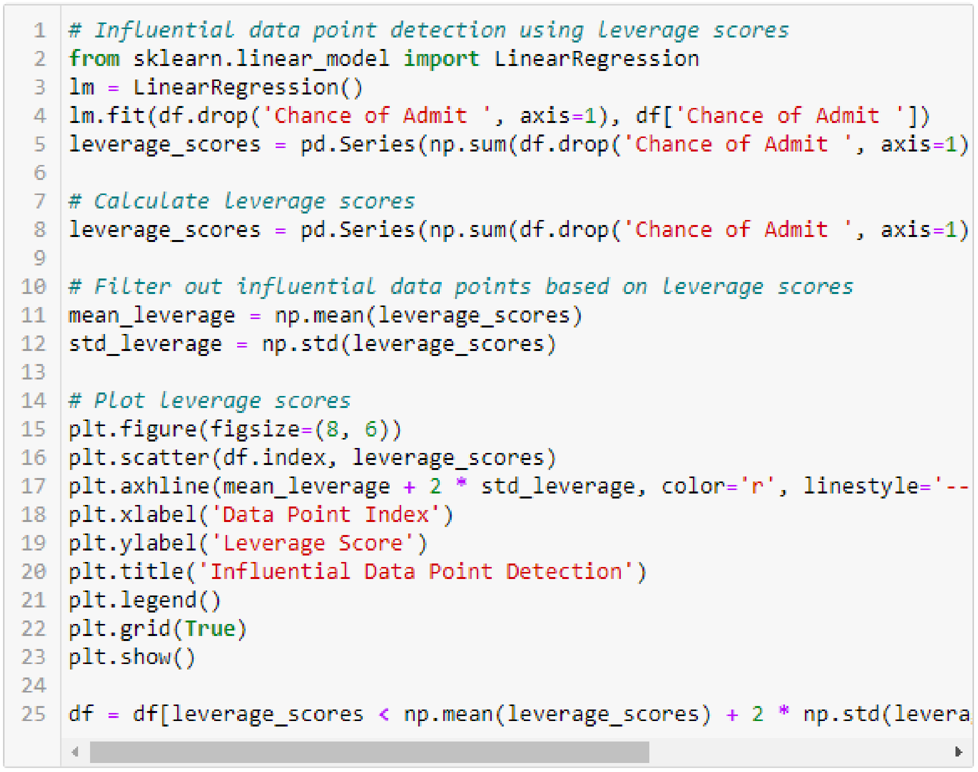
In this project, outliers were handled. We detected and handled outliers in the dataset using the Interquartile Range (IQR) method.



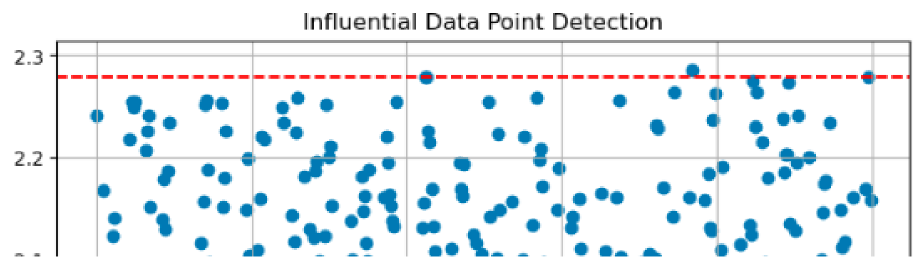
```
# Outlier detection using IQR
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
outliers = (df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))
df = df[~outliers.any(axis=1)]
df #reduced from 500 to 491
```

Finding influential data points

Influential datapoints were found using leverage scores.

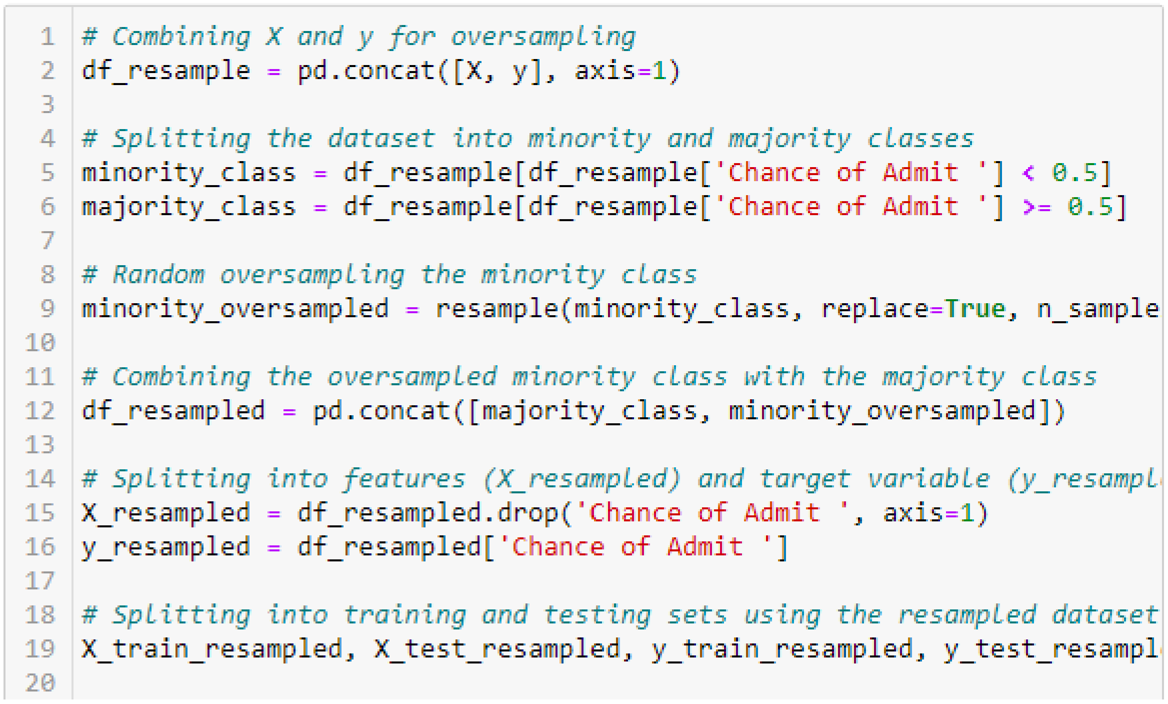


```
# Influential data point detection using leverage scores
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(df.drop('Chance of Admit ', axis=1), df['Chance of Admit '])
leverage_scores = pd.Series(np.sum(df.drop('Chance of Admit ', axis=1)

# Calculate leverage scores
leverage_scores = pd.Series(np.sum(df.drop('Chance of Admit ', axis=1)

# Filter out influential data points based on leverage scores
mean_leverage = np.mean(leverage_scores)
std_leverage = np.std(leverage_scores)

# Plot leverage scores
plt.figure(figsize=(8, 6))
plt.scatter(df.index, leverage_scores)
plt.axhline(mean_leverage + 2 * std_leverage, color='r', linestyle='--
plt.xlabel('Data Point Index')
plt.ylabel('Leverage Score')
plt.title('Influential Data Point Detection')
plt.legend()
plt.grid(True)
plt.show()

df = df[leverage_scores < np.mean(leverage_scores) + 2 * np.std(levera
```



Address data imbalance

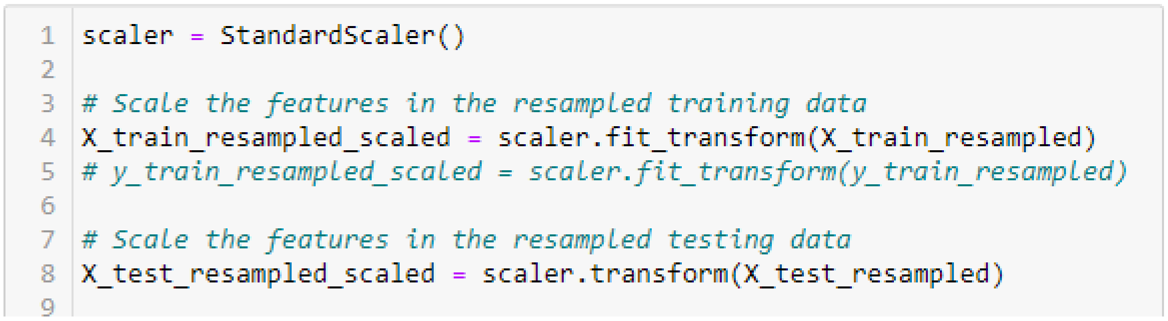
Oversampling technique was used to address issues pertaining to data imbalance.



```
# Combining X and y for oversampling
df_resample = pd.concat([X, y], axis=1)

# Splitting the dataset into minority and majority classes
minority_class = df_resample[df_resample['Chance of Admit '] < 0.5]
majority_class = df_resample[df_resample['Chance of Admit '] >= 0.5]

# Random oversampling the minority class
minority_oversampled = resample(minority_class, replace=True, n_sample

# Combining the oversampled minority class with the majority class
df_resampled = pd.concat([majority_class, minority_oversampled])

# Splitting into features (X_resampled) and target variable (y_resampl
X_resampled = df_resampled.drop('Chance of Admit ', axis=1)
y_resampled = df_resampled['Chance of Admit ']

# Splitting into training and testing sets using the resampled dataset
X_train_resampled, X_test_resampled, y_train_resampled, y_test_resampl

```

Normalizing features

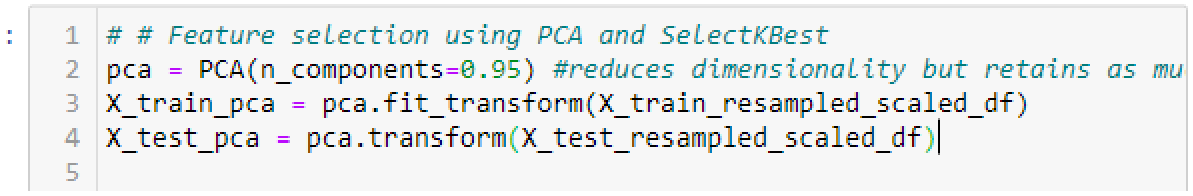
In order to balance the training dataset, Standard Scalar was used.



```
scaler = StandardScaler()

# Scale the features in the resampled training data
X_train_resampled_scaled = scaler.fit_transform(X_train_resampled)
# y_train_resampled_scaled = scaler.fit_transform(y_train_resampled)

# Scale the features in the resampled testing data
X_test_resampled_scaled = scaler.transform(X_test_resampled)

```

Feature selection

PCA was used for feature selection. It reduces dimensionality but retains as much info as possible.


```
:  # # Feature selection using PCA and SelectKBest
   pca = PCA(n_components=0.95) #reduces dimensionality but retains as mu
   X_train_pca = pca.fit_transform(X_train_resampled_scaled_df)
   X_test_pca = pca.transform(X_test_resampled_scaled_df)
```

MODEL SELECTION AND TRAINING

Four regression algorithms were used. They include Decision Tree, Random Forest, K-Nearest Neighbors, and XGBoots. The models were trained on the independent variables that went through the data preprocessing.

Cross-validation technique was used for evaluating the performance of a model on a holdout dataset. It works by dividing the data into k folds, and then training the model on k-1 folds and evaluating the model on the remaining fold. This process is repeated k times, and the average of the k scores is used as the cross-validation score.

Rational for the choices of algorithms

The algorithms Decision trees, random forests, K-nearest neighbors (KNN), and XGBoost are all popular machine learning algorithms that can be used for regression problems. Hence the reason for selecting them for this project.

Rational for the choice of cross-validation

Cross validation was used because it provides a more accurate estimate of the model's performance than simply evaluating the model on the training data.

Default parameters used in the training.

Scoring=’neg_mean_sqauared_error’

Cv=5. This means the models will be trained on 4 of the folds and then evaluated on the remaining fold. This process will be repeated 5 times, and the average of the 5 scores will be used as the cross-validation score.

All other parameters were kept as default.

EVALUATION METRICS

The metrics used in the valuation of the above model are:

- **Mean Squared Error (MSE):** The mean squared error is a measure of the average squared difference between the predicted values and the actual values. A low MSE indicates that the model is making accurate predictions.
- **Root Mean Squared Error (RMSE):** The root mean squared error is the square root of the mean squared error. It is a measure of the average error between the predicted values and the actual values. A low RMSE indicates that the model is making accurate predictions.
- **Mean Absolute Error (MAE):** The mean absolute error is a measure of the average absolute difference between the predicted values and the actual values. A low MAE indicates that the model is making accurate predictions.

- **R-squared Score:** The R-squared score is a measure of how well the model fits the data. A high R-squared score indicates that the model is making accurate predictions.
- **Explained Variance Score**: The explained variance score is a measure of how much of the variance in the data is explained by the model. A high explained variance score indicates that the model is making accurate predictions.
- **Median Absolute Error:** The median absolute error is a measure of the median absolute difference between the predicted values and the actual values. It is less sensitive to outliers than the mean absolute error.

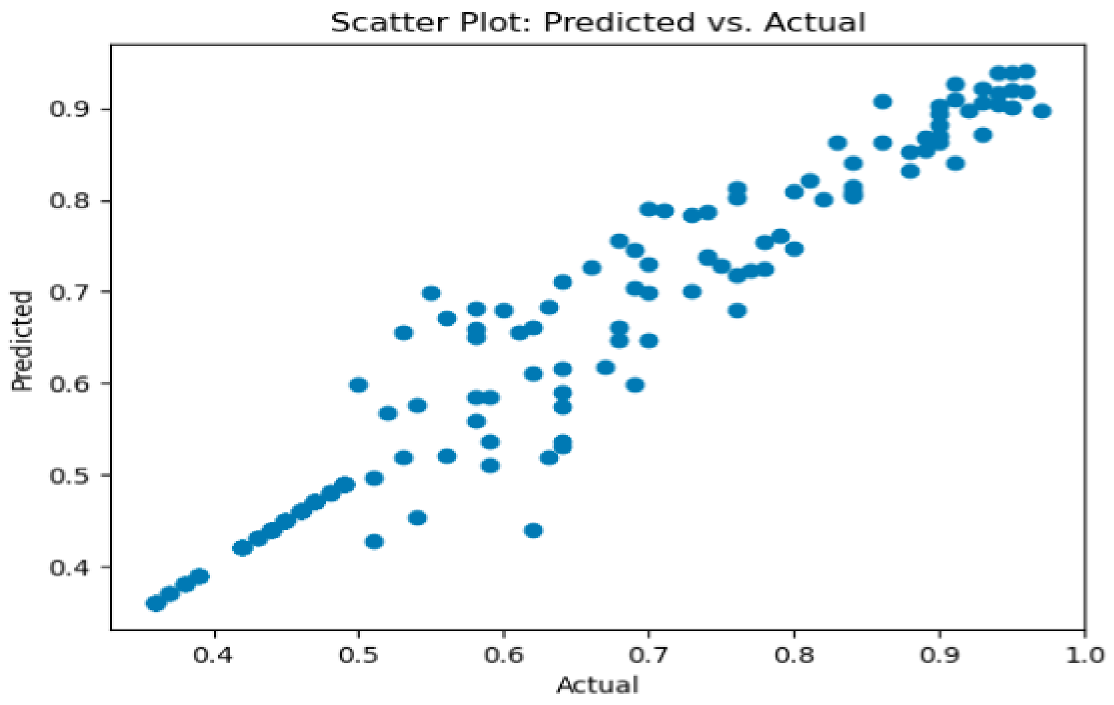
<u>RESULTS AND ANALYSIS</u>

PERFORMANCE OF MODELS

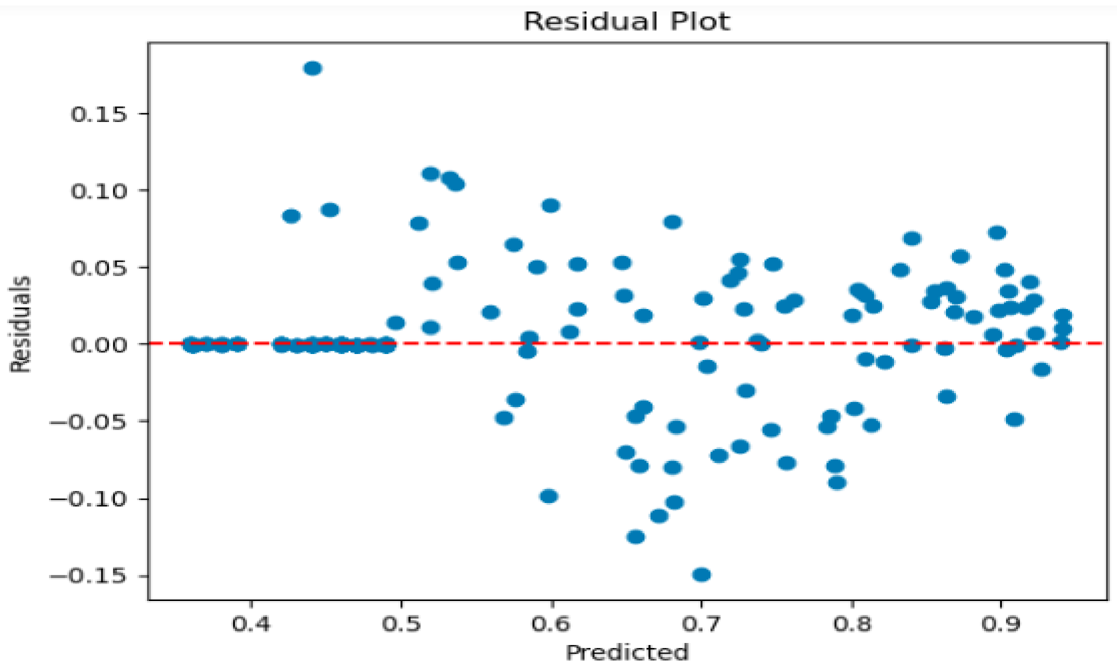
MODELS	MSE	RMSE	MAE	RSS	ESV	MEDIAN ABSOLUTE ERROR
Decision Tree	0.003	0.055	0.028	0.912	0.912	3.330
Random forest	0.002	0.044	0.0245	0.942	0.943	0.0023
KNN	0.0023	0.050	0.0274	0.924	0.926	0.0060
XGBoost	0.0017	0.041	0.0237	0.949	0.950	0.0014

CV=5 PERFORMANCE OF THE MODELS

Cross-validation	**ACTUAL**	DECISION TREE	RANDOM FOREST	KNN	XGBoost
K VALUE		**PREDICTED**			
K=0	0.92	0.92	0.93	0.93	0.92
K=1	0.76	0.76	0.77	0.78	0.76
K=2	0.72	0.72	0.67	0.51	0.72
K=3	0.80	0.80	0.75	0.74	0.80
K=4	0.65	0.65	0.62	0.51	0.65



Scatter plot of the actual vs predicted results.



Residual plot

PART THREE

CLUSTERING PROBLEM

INTRODUCTION

In the increasingly competitive business landscape, effective marketing campaigns play a vital role in attracting and retaining customers. However, designing successful marketing strategies requires a deep understanding of customer segments and their preferences. Clustering analysis offers a powerful approach to uncover underlying patterns and group similar customers together based on their behaviors, demographics, and responses to marketing campaigns. This report aims to analyze the Marketing Campaign dataset using clustering techniques to gain insights into customer segmentation and enhance marketing strategies.

DATASET DESCRIPTION

Dataset Structure:

The dataset contains a total of 2,240 entries or rows and 29 columns or attributes in the dataset.

Attribute Description:

- ID: Represents a unique identifier for each customer.
- Year_Birth: Captures the birth year of the customer.
- Education: Represents the education level of the customer.
- Marital_Status: Indicates the marital status of the customer.
- Income: Represents the income of the customer (with 24 missing values).
- Kidhome: Indicates the number of young children in the customer's household.
- Teenhome: Represents the number of teenagers in the customer's household.

- Dt_Customer: Captures the date when the customer was added to the database.
- Recency: Measures the number of days since the last contact with the customer.
- MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts, MntGoldProds: These attributes represent the amount of money spent by the customer on different product categories.
- NumDealsPurchases, NumWebPurchases, NumCatalogPurchases, NumStorePurchases, NumWebVisitsMonth: Capture the number of purchases or visits made by the customer.
- AcceptedCmp1, AcceptedCmp2, AcceptedCmp3, AcceptedCmp4, AcceptedCmp5: Indicate whether the customer accepted a specific marketing campaign (binary values: 0 or 1).
- Complain: Represents whether the customer made any complaints (binary values: 0 or 1).
- Z_CostContact, Z_Revenue: Additional cost and revenue information (constant values for all entries).
- Response: Indicates whether the customer responded to the marketing campaign (binary values: 0 or 1).

Data Types:

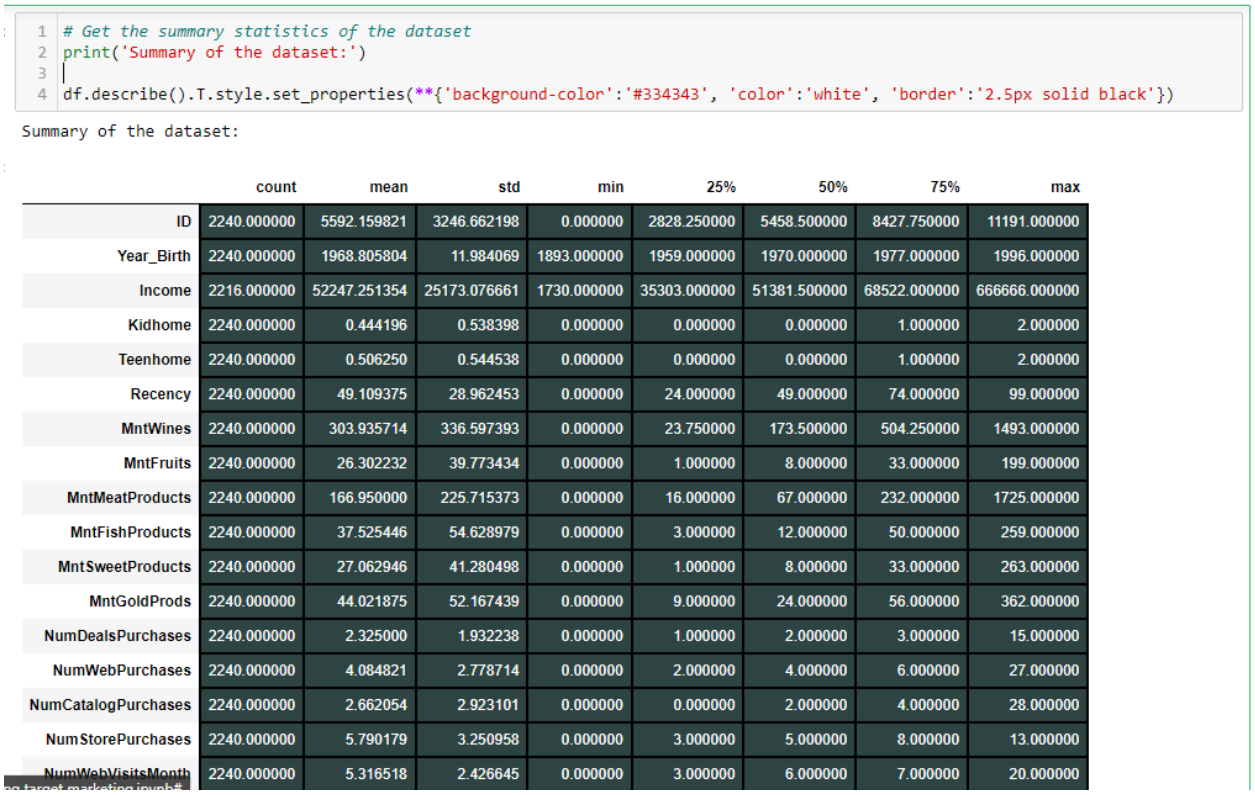
- The dataset contains attributes of different data types:
- float64: 1 attribute (Income)
- int64: 25 attributes
- object: 3 attributes (Education, Marital_Status, Dt_Customer)

Missing Values:

The 'Income' attribute has 24 missing values.

EXPLORATORY DATA ANALYSIS

- Looking at the head of the dataset

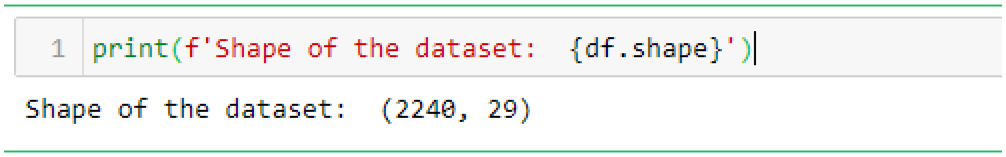


```python
# Get the summary statistics of the dataset
print('Summary of the dataset:')

df.describe().T.style.set_properties(**{'background-color':'#334343', 'color':'white', 'border':'2.5px solid black'})
```

```
Summary of the dataset:
```

	count	mean	std	min	25%	50%	75%	max
ID	2240.000000	5592.159821	3246.662198	0.000000	2828.250000	5458.500000	8427.750000	11191.000000
Year_Birth	2240.000000	1968.805804	11.984069	1893.000000	1959.000000	1970.000000	1977.000000	1996.000000
Income	2216.000000	52247.251354	25173.076661	1730.000000	35303.000000	51381.500000	68522.000000	666666.000000
Kidhome	2240.000000	0.444196	0.538398	0.000000	0.000000	0.000000	1.000000	2.000000
Teenhome	2240.000000	0.506250	0.544538	0.000000	0.000000	0.000000	1.000000	2.000000
Recency	2240.000000	49.109375	28.962453	0.000000	24.000000	49.000000	74.000000	99.000000
MntWines	2240.000000	303.935714	336.597393	0.000000	23.750000	173.500000	504.250000	1493.000000
MntFruits	2240.000000	26.302232	39.773434	0.000000	1.000000	8.000000	33.000000	199.000000
MntMeatProducts	2240.000000	166.950000	225.715373	0.000000	16.000000	67.000000	232.000000	1725.000000
MntFishProducts	2240.000000	37.525446	54.628979	0.000000	3.000000	12.000000	50.000000	259.000000
MntSweetProducts	2240.000000	27.062946	41.280498	0.000000	1.000000	8.000000	33.000000	263.000000
MntGoldProds	2240.000000	44.021875	52.167439	0.000000	9.000000	24.000000	56.000000	362.000000
NumDealsPurchases	2240.000000	2.325000	1.932238	0.000000	1.000000	2.000000	3.000000	15.000000
NumWebPurchases	2240.000000	4.084821	2.778714	0.000000	2.000000	4.000000	6.000000	27.000000
NumCatalogPurchases	2240.000000	2.662054	2.923101	0.000000	0.000000	2.000000	4.000000	28.000000
NumStorePurchases	2240.000000	5.790179	3.250958	0.000000	3.000000	5.000000	8.000000	13.000000
NumWebVisitsMonth	2240.000000	5.316518	2.426645	0.000000	3.000000	6.000000	7.000000	20.000000

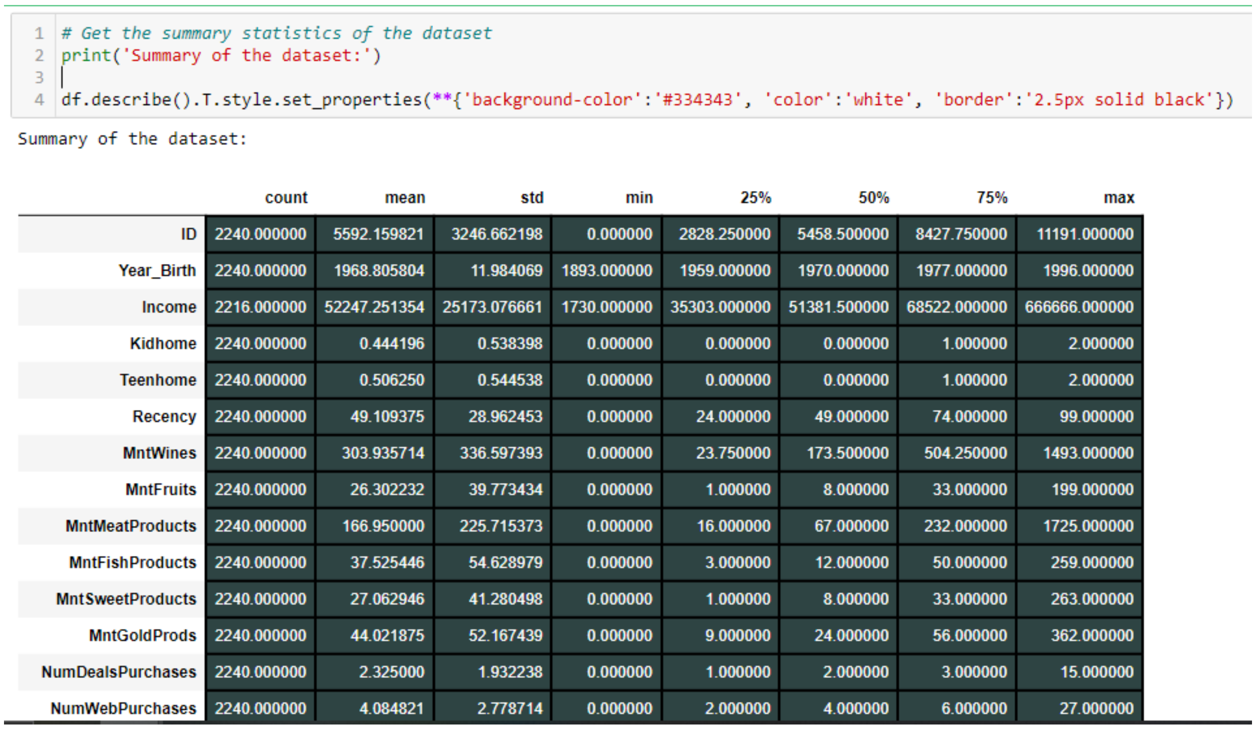
- Exploring the shape of the dataset



```python
print(f'Shape of the dataset:  {df.shape}')
```

```
Shape of the dataset:  (2240, 29)
```

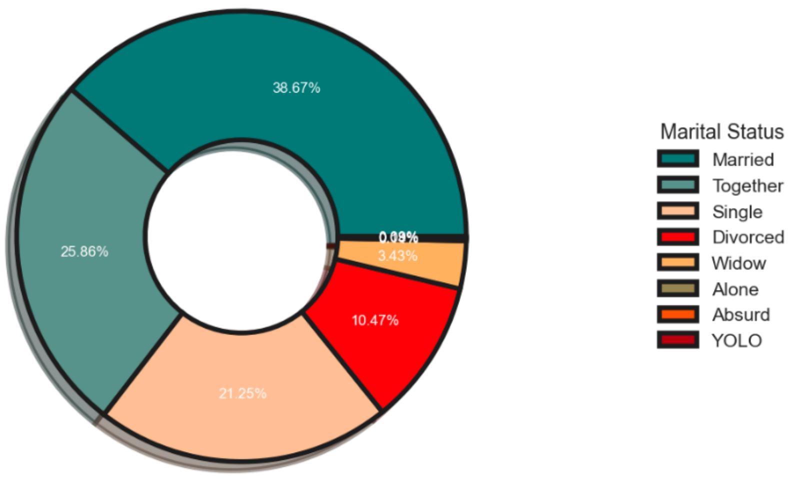
- Summary statistics of the dataset



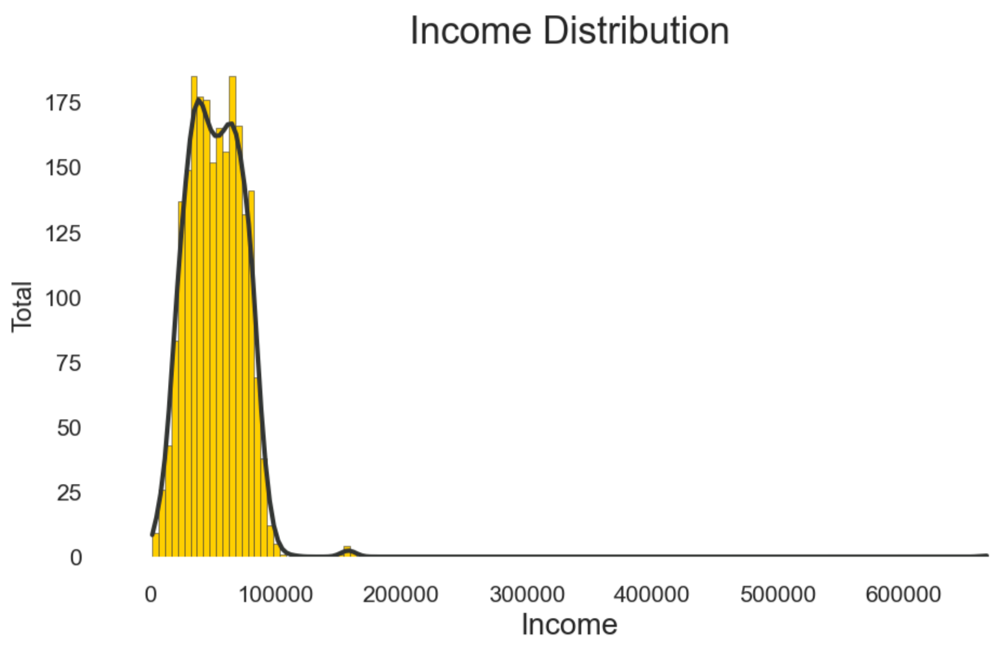
```
# Get the summary statistics of the dataset
print('Summary of the dataset:')

df.describe().T.style.set_properties(**{'background-color':'#334343', 'color':'white', 'border':'2.5px solid black'})
```

```
Summary of the dataset:
```

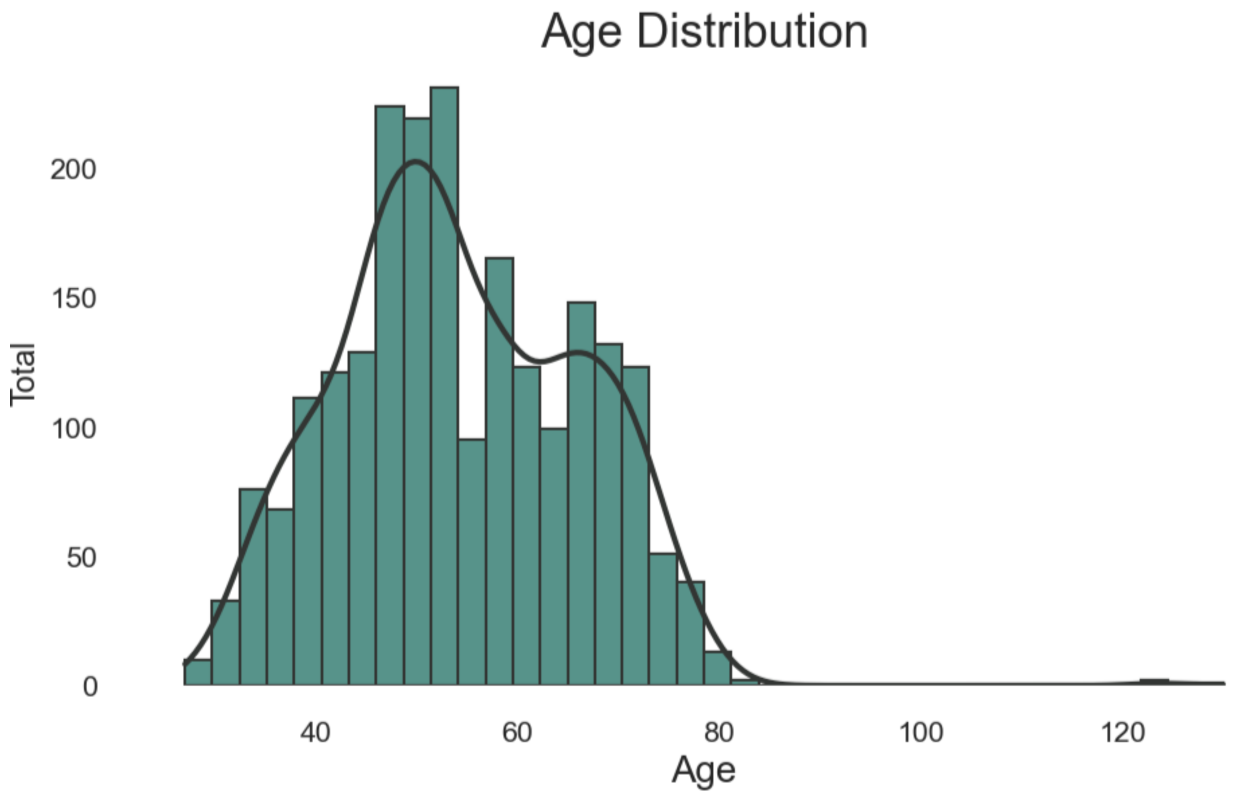
	count	mean	std	min	25%	50%	75%	max
ID	2240.000000	5592.159821	3246.662198	0.000000	2828.250000	5458.500000	8427.750000	11191.000000
Year_Birth	2240.000000	1968.805804	11.984069	1893.000000	1959.000000	1970.000000	1977.000000	1996.000000
Income	2216.000000	52247.251354	25173.076661	1730.000000	35303.000000	51381.500000	68522.000000	666666.000000
Kidhome	2240.000000	0.444196	0.538398	0.000000	0.000000	0.000000	1.000000	2.000000
Teenhome	2240.000000	0.506250	0.544538	0.000000	0.000000	0.000000	1.000000	2.000000
Recency	2240.000000	49.109375	28.962453	0.000000	24.000000	49.000000	74.000000	99.000000
MntWines	2240.000000	303.935714	336.597393	0.000000	23.750000	173.500000	504.250000	1493.000000
MntFruits	2240.000000	26.302232	39.773434	0.000000	1.000000	8.000000	33.000000	199.000000
MntMeatProducts	2240.000000	166.950000	225.715373	0.000000	16.000000	67.000000	232.000000	1725.000000
MntFishProducts	2240.000000	37.525446	54.628979	0.000000	3.000000	12.000000	50.000000	259.000000
MntSweetProducts	2240.000000	27.062946	41.280498	0.000000	1.000000	8.000000	33.000000	263.000000
MntGoldProds	2240.000000	44.021875	52.167439	0.000000	9.000000	24.000000	56.000000	362.000000
NumDealsPurchases	2240.000000	2.325000	1.932238	0.000000	1.000000	2.000000	3.000000	15.000000
NumWebPurchases	2240.000000	4.084821	2.778714	0.000000	2.000000	4.000000	6.000000	27.000000



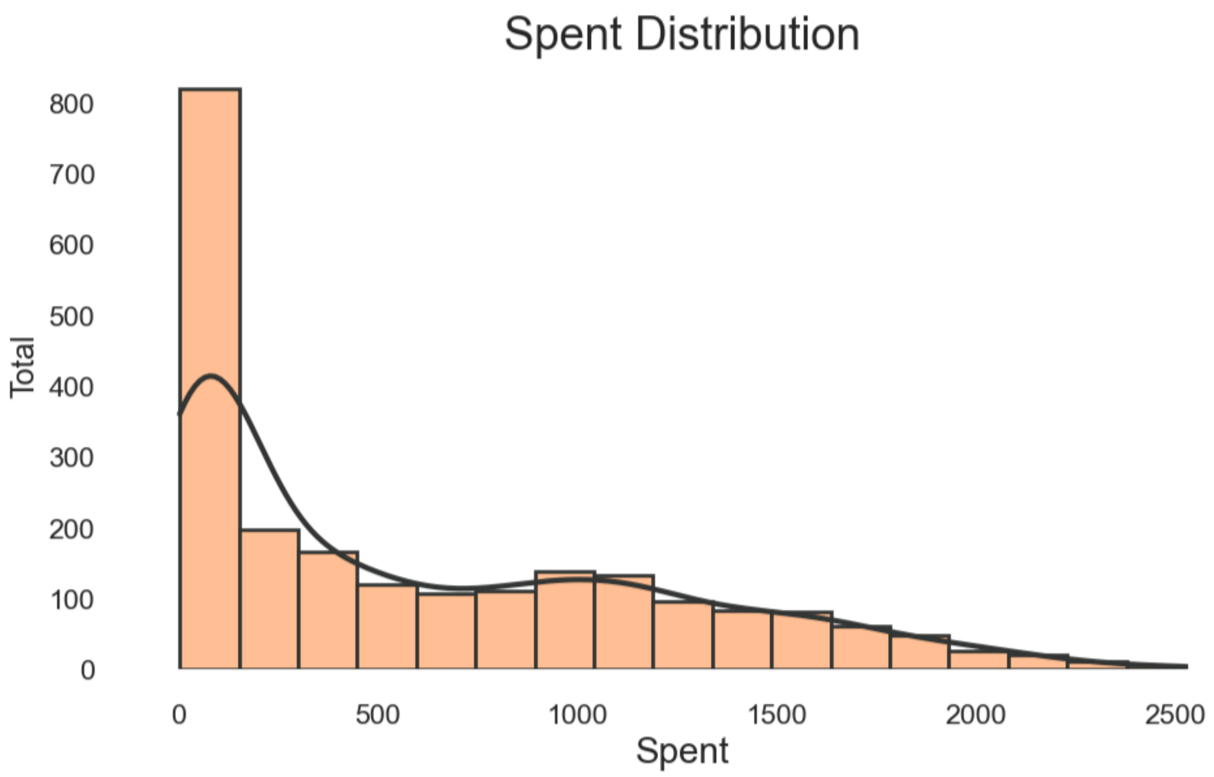
Analysis of the material status of the dataset after preprocessing



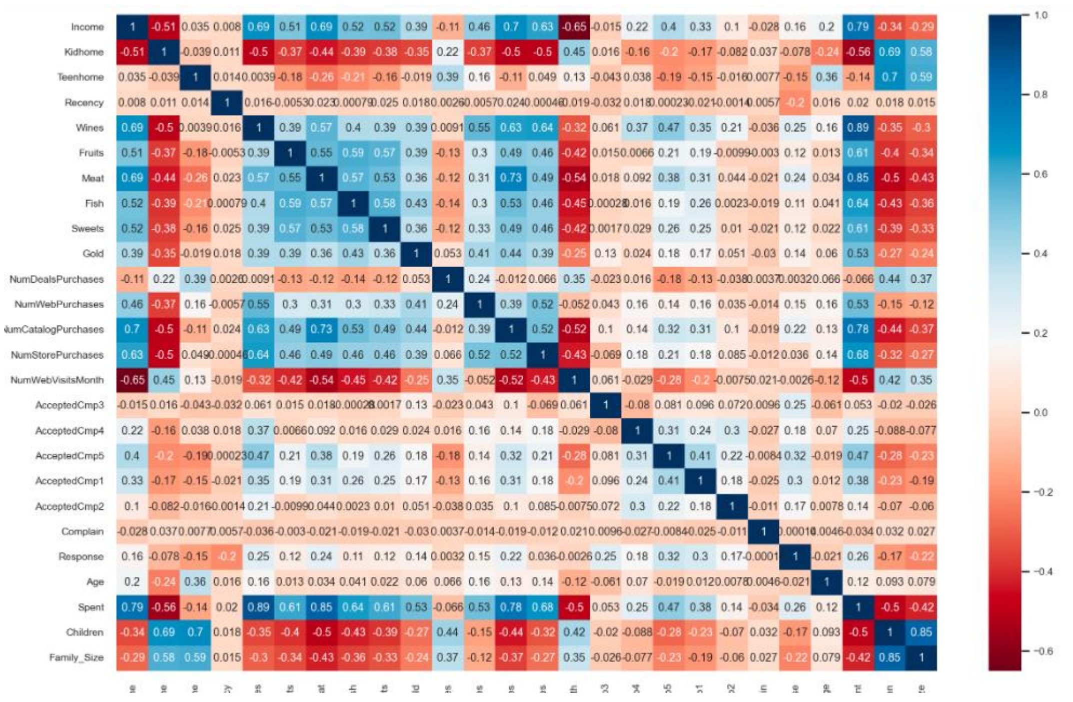
Income distributions after preprocessing



Age distributions after preprocessing



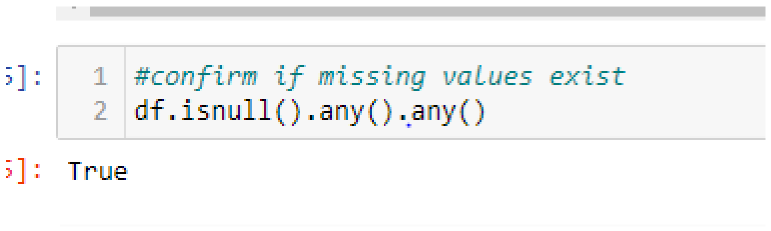
Spending distribution.



Confusion matrix of the dataset after preprocessing

PREPROCESSING

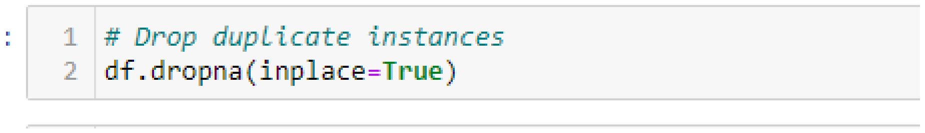
Missing Values



```
#confirm if missing values exist
df.isnull().any().any()
```

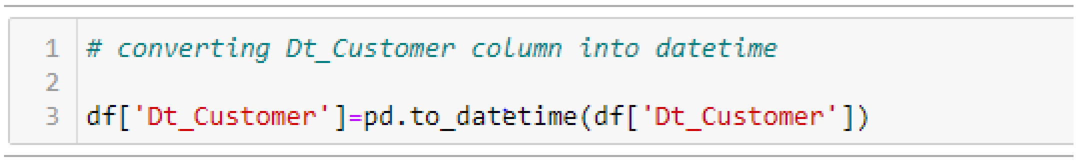
```
True
```

Removing duplicates



```
# Drop duplicate instances
df.dropna(inplace=True)
```

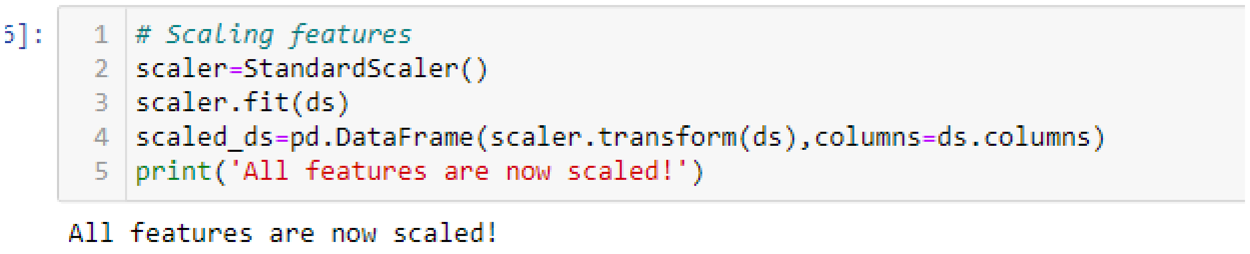
Feature transformation



```
# converting Dt_Customer column into datetime

df['Dt_Customer']=pd.to_datetime(df['Dt_Customer'])
```

The rational, we are trying to build a machine learning model to predict customer churn, you would need to know the date that the customer became a customer. If the Dt_Customer column is not a datetime datatype, then the machine learning algorithm will not be able to understand the data.

Normalizing data



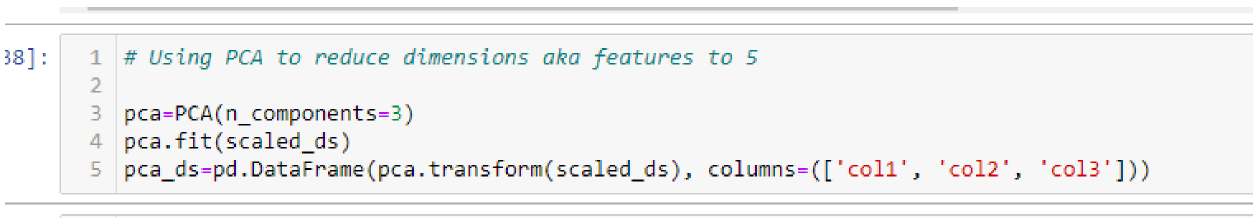
5]:

```
# Scaling features
scaler=StandardScaler()
scaler.fit(ds)
scaled_ds=pd.DataFrame(scaler.transform(ds),columns=ds.columns)
print('All features are now scaled!')
```

```
All features are now scaled!
```

Dimensionality reduction

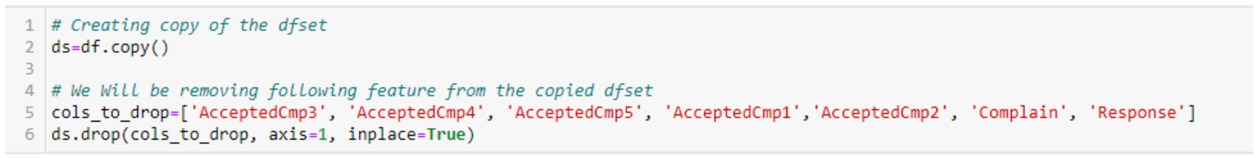
Principal component analysis (PCA) is used to reduce the dimensionality. It is a mathematical technique that can be used to reduce the dimensionality of a dataset while preserving as much of the variation in the data as possible. This can be useful for machine learning algorithms that are not able to handle high-dimensional data.



38]:

```
# Using PCA to reduce dimensions aka features to 5

pca=PCA(n_components=3)
pca.fit(scaled_ds)
pca_ds=pd.DataFrame(pca.transform(scaled_ds), columns=(['col1', 'col2', 'col3']))
```

Feature selection

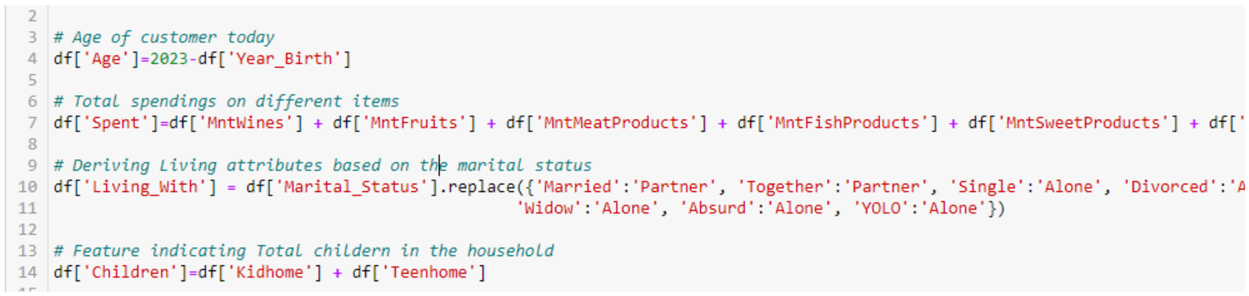


```
# Creating copy of the dfset
ds=df.copy()

# We Will be removing following feature from the copied dfset
cols_to_drop=['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']
ds.drop(cols_to_drop, axis=1, inplace=True)
```

Feature engineering

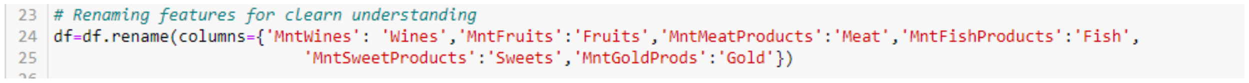
Feature engineering is the process of transforming raw data into features that are more informative and useful for machine learning algorithms. This can involve a variety of tasks, such as:

- Creating new features: This can involve combining existing features, transforming features, or creating entirely new features. For example, the code you provided creates the Age feature by subtracting the Year_Birth feature from the current year.



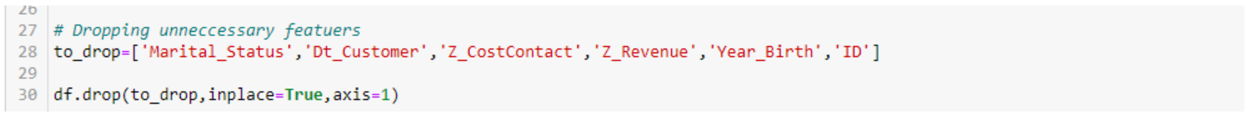
```

# Age of customer today
df['Age']=2023-df['Year_Birth']

# Total spendings on different items
df['Spent']=df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntFishProducts'] + df['MntSweetProducts'] + df['

# Deriving Living attributes based on the marital status
df['Living_With'] = df['Marital_Status'].replace({'Married':'Partner', 'Together':'Partner', 'Single':'Alone', 'Divorced':'A
                                              'Widow':'Alone', 'Absurd':'Alone', 'YOLO':'Alone'})

# Feature indicating Total childern in the household
df['Children']=df['Kidhome'] + df['Teenhome']
```

- Renaming features: This can make the features more readable and understandable. For example, the code you provided renames the MntWines feature to Wines.



```
# Renaming features for clearn understanding
df=df.rename(columns={'MntWines': 'Wines','MntFruits':'Fruits','MntMeatProducts':'Meat','MntFishProducts':'Fish',
                      'MntSweetProducts':'Sweets','MntGoldProds':'Gold'})
```

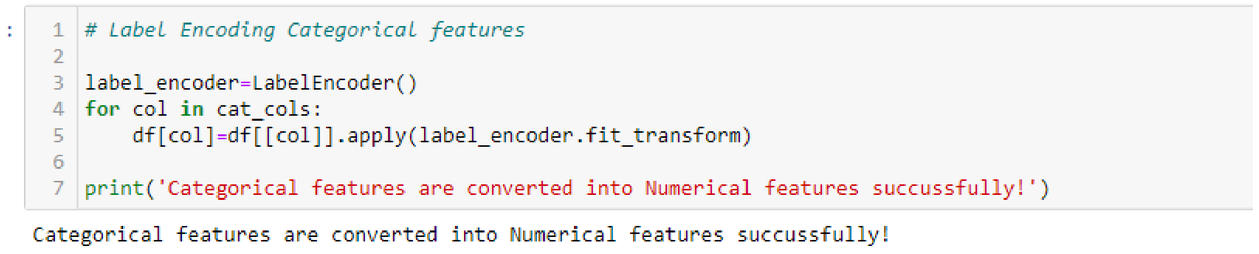
- Dropping features: This can be done to remove features that are not informative or that are correlated with other features. For example, the code you provided drops the Marital_Status feature because it is correlated with the Living_With feature.



```
# Dropping unneccessary featuers
to_drop=['Marital_Status','Dt_Customer','Z_CostContact','Z_Revenue','Year_Birth','ID']

df.drop(to_drop,inplace=True,axis=1)
```

- Encoding of categorical features:

Label_encoder function is used to encode categorical features into numerical features. This is done because machine learning algorithms typically work better with numerical features. This is because numerical features are easier to understand and process



```
# Label Encoding Categorical features

label_encoder=LabelEncoder()
for col in cat_cols:
    df[col]=df[[col]].apply(label_encoder.fit_transform)

print('Categorical features are converted into Numerical features succussfully!')
```

```
Categorical features are converted into Numerical features succussfully!
```

MODEL SECTION AND TRAINING

Three alogrithms were used:

Agglomerative Clustering: This is a hierarchical clustering algorithm that works by merging similar clusters together.

The default parameters for Agglomerative Clustering are:

- ✓ n_clusters: The number of clusters to be created.
- ✓ linkage: The linkage criterion to be used. The linkage criterion determines how clusters are merged together. The default linkage criterion is ward.

DBSCAN: This is a density-based clustering algorithm that works by finding clusters of high density.

The default parameters for DBSCAN are:

- eps: The maximum distance between two points to be considered as part of the same cluster.
- min_samples: The minimum number of points required to form a cluster.

K-Means: This is a partitional clustering algorithm that works by grouping data points into k clusters.

The default parameters for K-Means are:

- n_clusters: The number of clusters to be created.
- init: The method used to initialize the cluster centroids. The default initialization method is random.
- n_init: The number of times the K-Means algorithm is run with different random initializations. The best model is then chosen based on the inertia score.

EVALUATION METRICS

These evaluation metrics help assess the performance and quality of clustering algorithms. They provide insights into the compactness, separation, and overall effectiveness of the clustering solution. By considering different aspects of the clustering results, these metrics assist in selecting the optimal number of clusters and evaluating the clustering algorithm's performance.

- ✓ **WCSS (Within-Cluster Sum of Squares):** WCSS measures the compactness of clusters by calculating the sum of squared distances between each data point and the centroid of its assigned cluster. Lower WCSS indicates tighter and more homogeneous clusters.
- ✓ **BCSS (Between-Cluster Sum of Squares):** BCSS measures the separation between clusters by calculating the sum of squared distances between the centroids of different clusters. Higher BCSS indicates distinct and well-separated clusters.
- ✓ **Silhouette Score:**The Silhouette Score considers both cohesion within clusters and separation between clusters. It calculates the average silhouette coefficient for all data

points, ranging from -1 to 1. Higher scores indicate better-defined and well-separated clusters.

- ✓ **Davis-Bouldin Score:** The Davis-Bouldin Score measures the average dissimilarity between clusters, considering both compactness within clusters and separation between clusters. Lower scores indicate better clustering with more distinct and well-separated clusters.

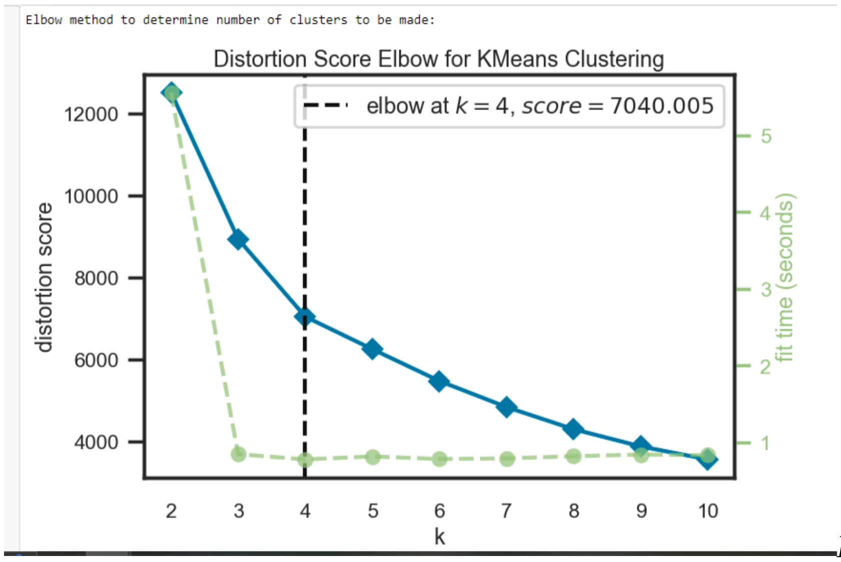
RESULTS AND ANALYSIS

Performance score on col1

Algorithm	Kmeans	Agglomerative clustering	Dbscan
WCSS	83.813	83.813	83.813
BCSS	24.549	24.549	24.549
silhouette_score	0.45	0.112	0.1670
davies_bouldin_score	0.8269	3.3548	1.9515

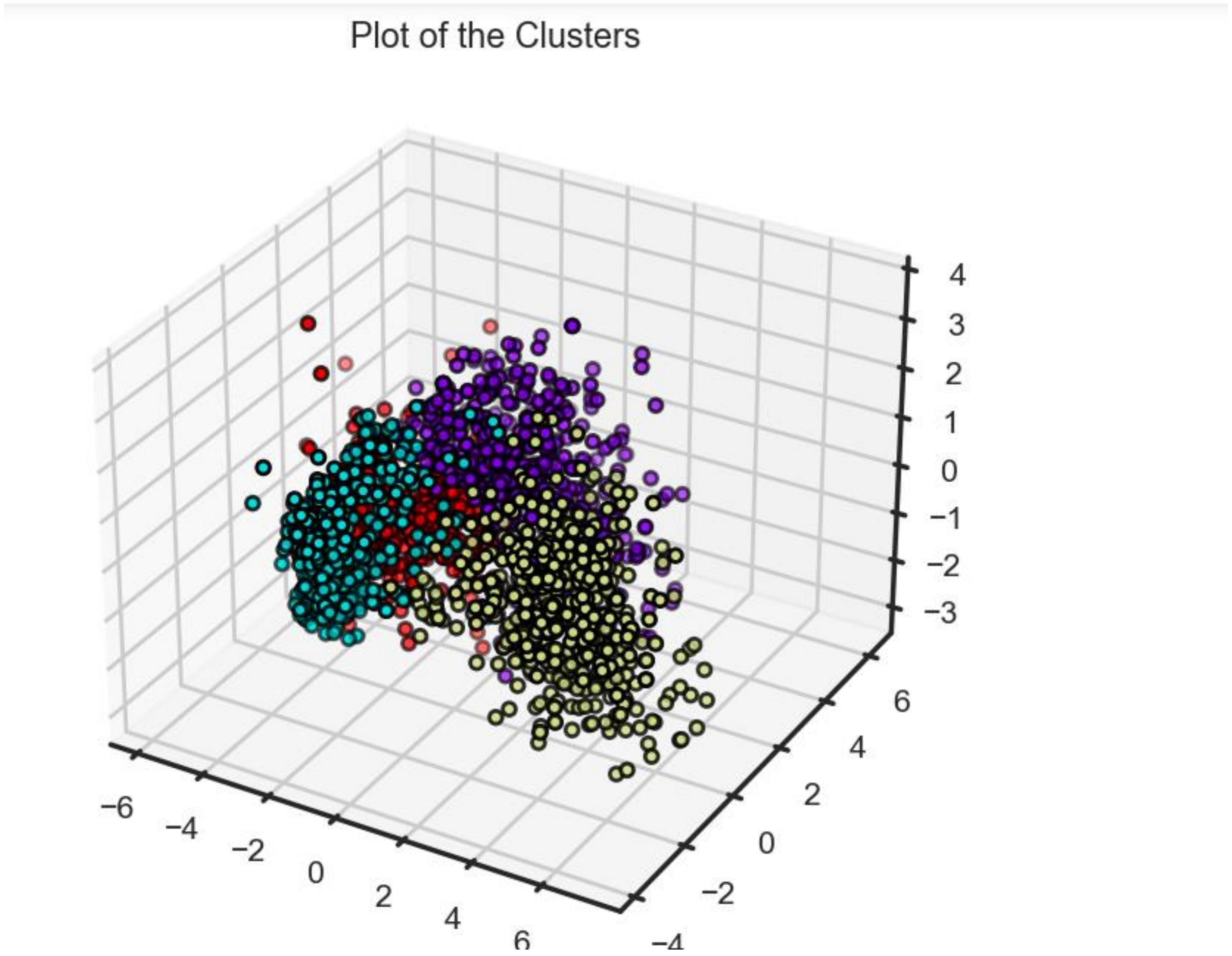
Performance score

SCORE		Kmeans	Agglomerative clustering	DBSCAN
WCSS	Col1 Col2 Col3	83.813207 10.220810 1.988349	83.813207 10.220810 1.988349	83.813207 10.220810 1.988349
BCSS	Col1 Col2 Col3	24.548848 7.127131 1.649267	24.548848 7.127131 1.649267	24.548848 7.127131 1.649267
Silhouette_Score		0.45	0.112	0.1670
Davies_Bouldin_Score		0.8269	3.3548	1.9515

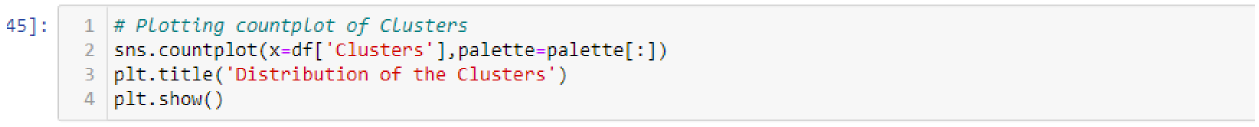


```
Elbow method to determine number of clusters to be made:
```

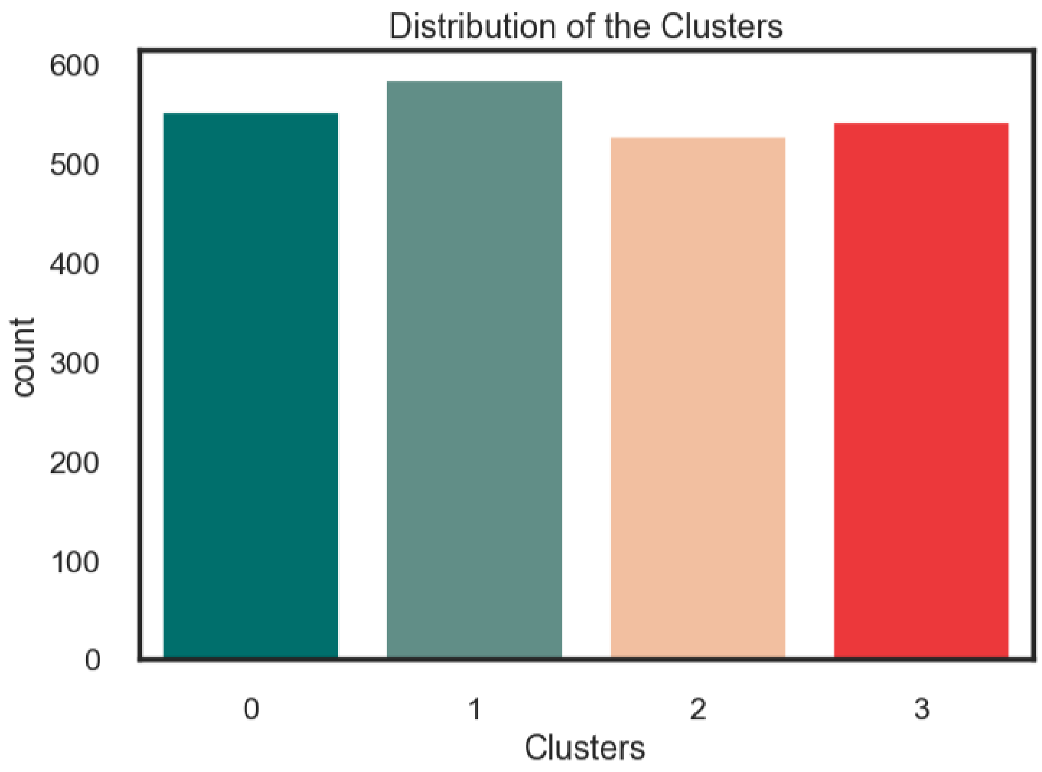
Number of clusters



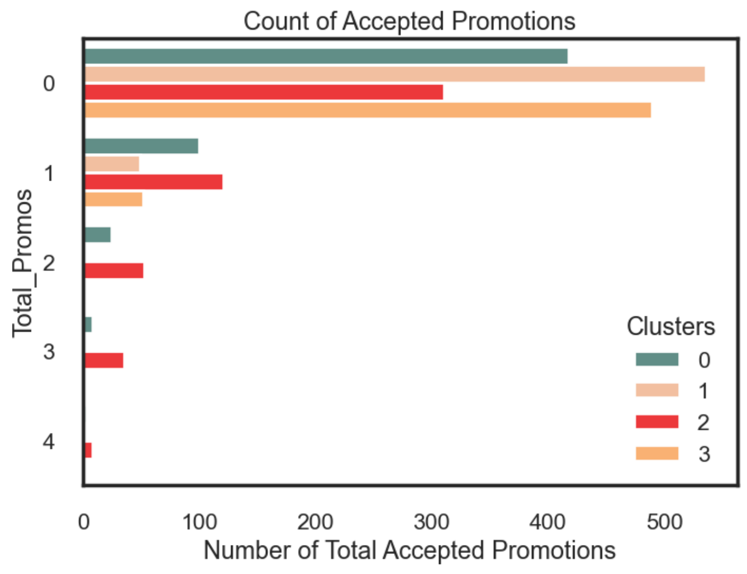
Scatter plot of clusters



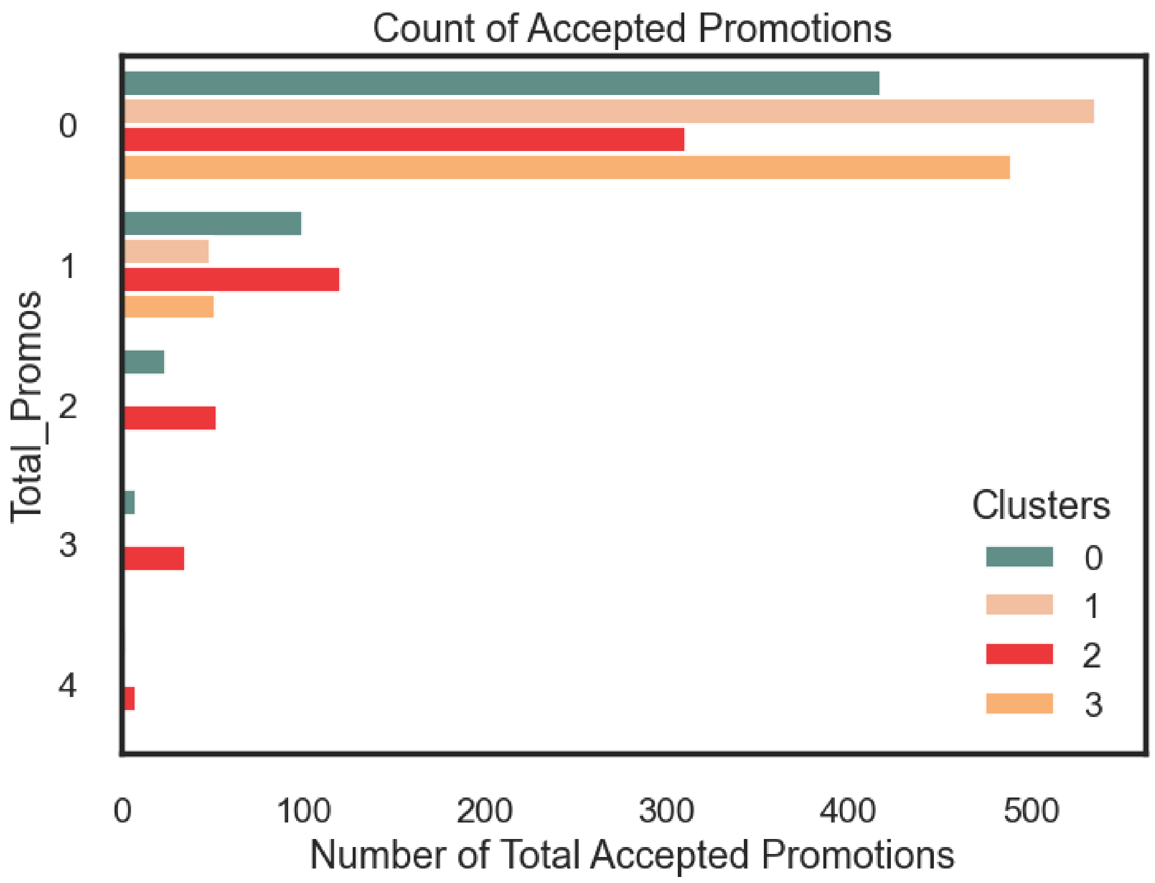
```
45]:
# Plotting countplot of Clusters
sns.countplot(x=df['Clusters'],palette=palette[:])
plt.title('Distribution of the Clusters')
plt.show()
```



Distribution of clusters



Acceptable promotions by customers



Cluster profiling

CONCLUSION.

The four clusters identified in the income vs. spending analysis can be described as follows:

Group 0: Medium income and high spending. This group represents customers who have a medium level of income but spend a relatively high amount of money. This may be due to a number of factors, such as having a large family or living in a high-cost area.

Group 1: Low income and low spending. This group represents customers who have a low level of income and spend a relatively low amount of money. This may be due to a number of factors, such as being retired or living on a fixed income.

Group 2: High income and high spending. This group represents customers who have a high level of income and spend a relatively high amount of money. This may be due to a number of factors, such as having a large disposable income or being interested in luxury goods.

Group 3: Low income and average spending. This group represents customers who have a low level of income and spend a relatively average amount of money. This may be due to a number of factors, such as being young or being careful with their money.

The different clusters can be seen as different customer segments. By understanding the characteristics of each segment, businesses can better target their marketing and sales efforts. For example, businesses may want to focus on offering discounts or promotions to customers in Group 1, while they may want to focus on providing high-quality products or services to customers in Group 2.

The income vs. spending clusters analysis can also be used to identify potential opportunities for growth. For example, businesses may find that there is a large underserved market of customers in Group 3. By targeting this market, businesses could potentially increase their sales and profits.

Overall, the income vs. spending clusters analysis is a valuable tool that can help businesses better understand their customers and identify potential opportunities for growth.