BERLIN SCHOOL OF BUSINESS & INNOVATION

University for the Creative Arts

Assignment Title : Problematisation of a Data-based Scenario and using Python for Problem Solving

Programme title : MSc Data Analytics

Name : Sachin Mehta

Year : 2023

CONTENTS

Statement of compliance with academic ethics and the avoidance of plagiarism

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people's texts I clearly declare it through the good use of references following academic ethics.
(In the case that it is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the postgraduate program).

Name and Surname (Capital letters):

SACHIN MEHTA

Date: **2023/07/02**

INTRODUCTION

Utilising the data to gather insightful knowledge, comprehend consumer behaviour, enhance marketing tactics, and make data-driven decisions, It assists marketers in trend identification, audience targeting, performance measurement, and increased marketing effectiveness. On a dataset of consumer actions and advertisement categories, the provided code illustrates the comparison of three distinct methods (Alsubari *et al*. 2022).

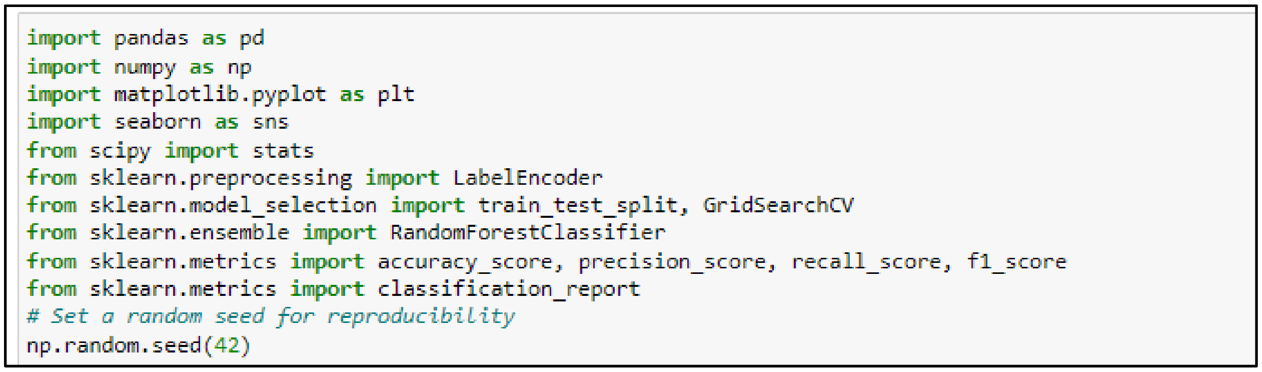
In order to ensure reproducibility, it creates synthetic data to mimic client behaviour and ad categories. A CSV file is then created and saved with the dataset for further usage. The data is then preprocessed after being loaded from the CSV file.

The categorization task is then carried out by the code. The dataset is divided into training and testing sets, and the training data is used to initialize and train a Random Forest classifier.

Data Collection and Structuring

The key steps in the data analytics process are data collecting and structuring, which include systematic gathering of data from various sources, including databases, surveys, web scraping, social media platforms, consumer interactions, and more, is known as data collection (Wang *et al*. 2022).

Once the data has been gathered, it must be organized and arranged in a way that makes efficient analysis possible. Data cleaning entails locating and fixing these problems to guarantee data integrity. Data integration is frequently gathered from several sources, some of which may have various structures or file formats. Data transformation is the process of transforming data into a format that is standardized and acceptable for analysis. Data formatting is concerned with arranging the data in a systematic manner. In order to do this, variables must be labelled, proper data types must be assigned, and the data must be set up in a tabular arrangement with rows and columns.

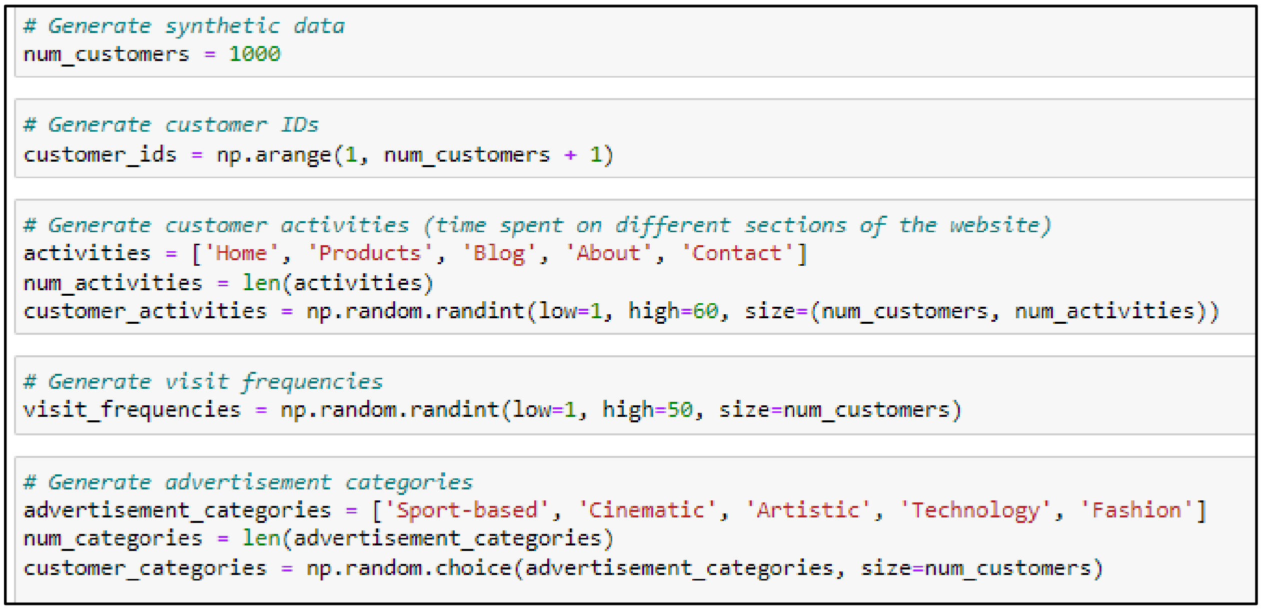


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
# Set a random seed for reproducibility
np.random.seed(42)
```

Figure 1: Loading required libraries

(Source: Observed and acquired by the learner)

The act of importing the essential software libraries or packages into a programming environment in order to use its features is referred to as loading required libraries. Libraries provide per-written functions and code that can be utilized to carry out particular processes or activities.



```
# Generate synthetic data
num_customers = 1000

# Generate customer IDs
customer_ids = np.arange(1, num_customers + 1)

# Generate customer activities (time spent on different sections of the website)
activities = ['Home', 'Products', 'Blog', 'About', 'Contact']
num_activities = len(activities)
customer_activities = np.random.randint(low=1, high=60, size=(num_customers, num_activities))

# Generate visit frequencies
visit_frequencies = np.random.randint(low=1, high=50, size=num_customers)

# Generate advertisement categories
advertisement_categories = ['Sport-based', 'Cinematic', 'Artistic', 'Technology', 'Fashion']
num_categories = len(advertisement_categories)
customer_categories = np.random.choice(advertisement_categories, size=num_customers)
```

Figure 2: Steps for generating Synthetic data

(Source: Observed and acquired by the learner)

Synthetic data generation entails producing artificial data that closely resembles the traits and trends of actual data. Typically, the procedures for producing synthetic data consist of:

Establish the statistical characteristics of the requested data, such as mean, variance, and distribution type, to define the data distribution.

Create random values: Create random values depending on the specified distribution parameters by using random number generators.

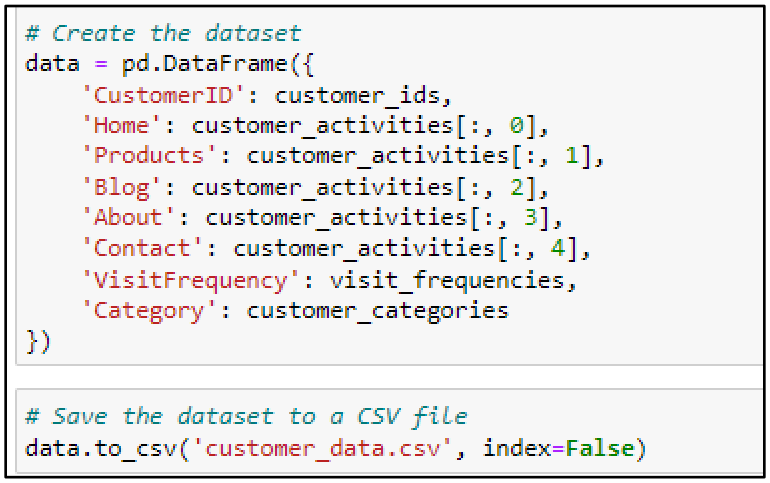
Include dependencies: Make sure that the generation process takes into account any dependencies or links between variables.

Add noise: Add noise or variability to the generated values to make the data more realistic. This can mimic the flaws and unpredictability found in real-world data.

Validate the synthetic data: Evaluate the generated data's accuracy and reliability by contrasting it with the traits of actual data.

Iterate and improve: Make changes to the distribution settings, add more dependencies, or improve the noise creation if the generated data does not satisfy the specified requirements or is unrealistic.

These procedures enable academics and data analysts to create artificial datasets that substantially resemble real-world data, enabling a variety of modelling and data analysis activities without disclosing confidential or restricted data.



```
# Create the dataset
data = pd.DataFrame({
    'CustomerID': customer_ids,
    'Home': customer_activities[:, 0],
    'Products': customer_activities[:, 1],
    'Blog': customer_activities[:, 2],
    'About': customer_activities[:, 3],
    'Contact': customer_activities[:, 4],
    'VisitFrequency': visit_frequencies,
    'Category': customer_categories
})
```

```
# Save the dataset to a CSV file
data.to_csv('customer_data.csv', index=False)
```

Figure 3: Creating and saving the dataset

(Source: Observed and acquired by the learner)

It is necessary to construct and preserve the synthetic data as a dataset when it has been generated. This entails structuring the collected data into a dataframe or CSV file and saving it to a designated location for use in upcoming modelling or data analysis tasks.

Data Pre-processing and Visualization

The workflow for data analytics must include important phases like data pre-processing and visualisation. In these phases, the data is transformed and prepared for analysis so that insights can be gained through visual representations.

Data cleaning: This stage involves dealing with the dataset's missing values, outliers, and inconsistencies. Depending on the needs of the analysis, missing values can either be imputed or deleted (Araz *et al*. 2020).

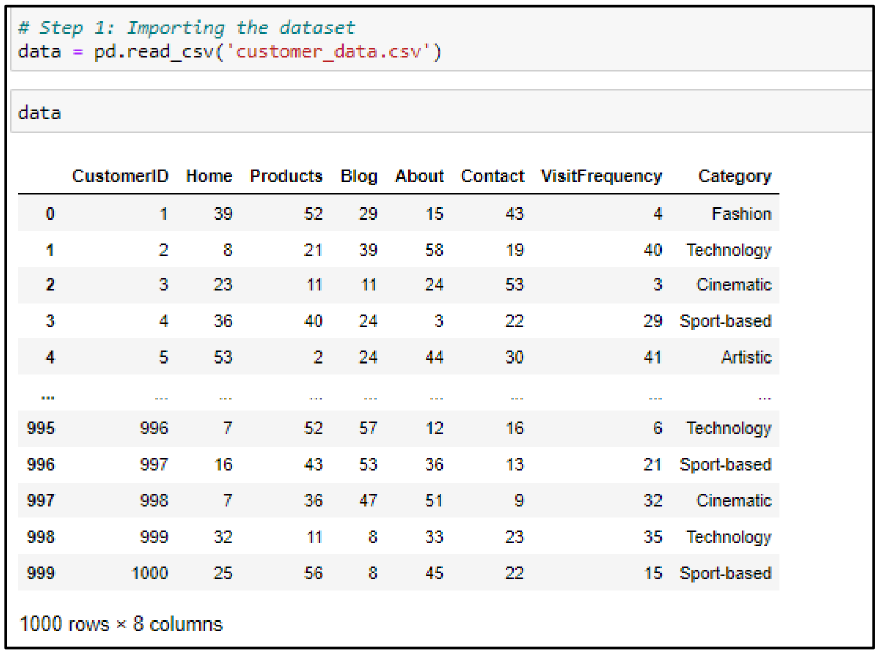
Data transformation: In certain conditions, it is important to change information to better the dispersion of factors or to adjust to the suppositions of measurable models. Scaling, standardization, and log changes are regularly utilized changes.

Feature engineering: It is the most common way of adding new elements or changing existing ones to upgrade the usefulness of AI models.

Data visualisation: Information representation is fundamental for perceiving examples, associations, and patterns.

Exploratory Data Analysis (EDA): EDA involves a more intensive examination of the dataset to grasp its elements. This incorporate working out rundown measurements, taking a gander at connections between information, spotting significant patterns, and concocting groundbreaking thoughts.

Data Visualization Libraries: Python has various libraries for information representation, including Matplotlib, Seaborn, and Plotly. These libraries offer a broad determination of portable plots and outlines to meet different information sorts and examination needs (Bhattarai *et al*. 2019).



```python
# Step 1: Importing the dataset
data = pd.read_csv('customer_data.csv')
```

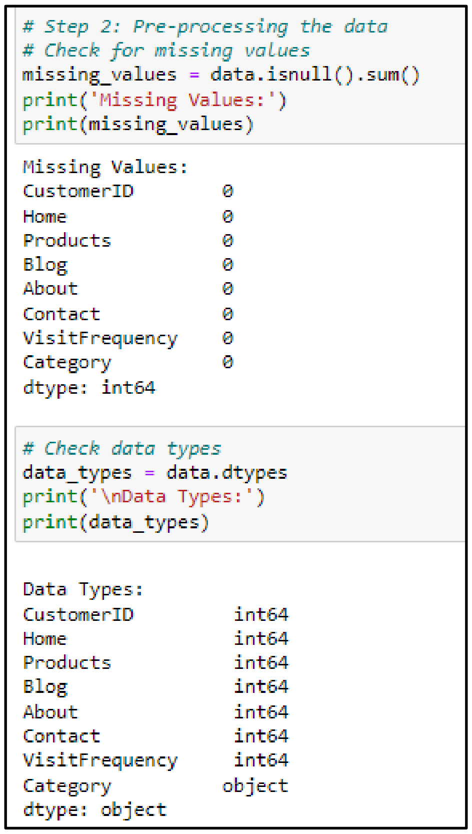
```python
data
```

	CustomerID	Home	Products	Blog	About	Contact	VisitFrequency	Category
0	1	39	52	29	15	43	4	Fashion
1	2	8	21	39	58	19	40	Technology
2	3	23	11	11	24	53	3	Cinematic
3	4	36	40	24	3	22	29	Sport-based
4	5	53	2	24	44	30	41	Artistic
...	...	...	...	...	...	...	...	...
995	996	7	52	57	12	16	6	Technology
996	997	16	43	53	36	13	21	Sport-based
997	998	7	36	47	51	9	32	Cinematic
998	999	32	11	8	33	23	35	Technology
999	1000	25	56	8	45	22	15	Sport-based

1000 rows × 8 columns

Figure 4: Importing the created dataset

(Source: Observed and acquired by the learner)



```python
# Step 2: Pre-processing the data
# Check for missing values
missing_values = data.isnull().sum()
print('Missing Values:')
print(missing_values)
```

```
Missing Values:
CustomerID        0
Home              0
Products          0
Blog              0
About             0
Contact           0
VisitFrequency    0
Category          0
dtype: int64
```

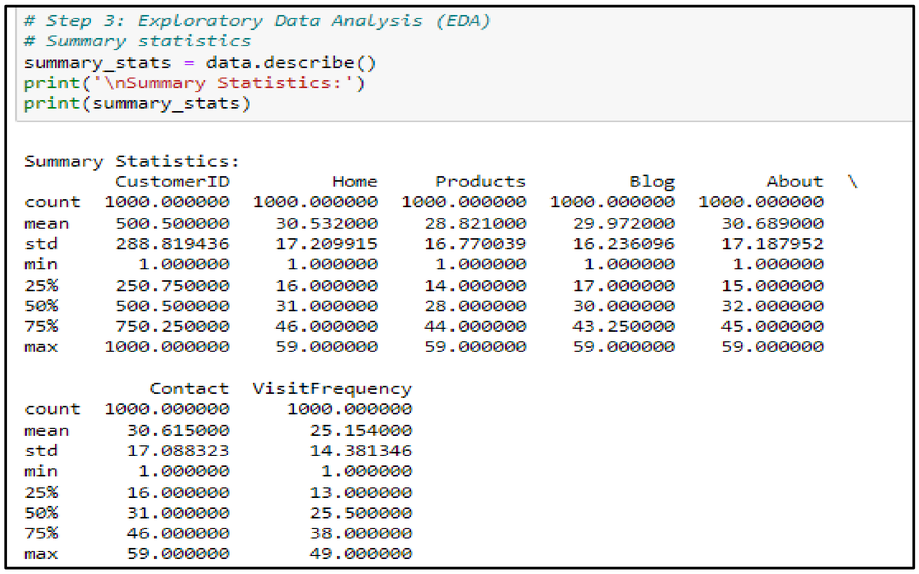
```python
# Check data types
data_types = data.dtypes
print('\nData Types:')
print(data_types)
```

```
Data Types:
CustomerID         int64
Home               int64
Products           int64
Blog               int64
About              int64
Contact            int64
VisitFrequency     int64
Category          object
dtype: object
```

Figure 5: Basic EDA steps

(Source: Observed and acquired by the learner)

Exploratory information examination (EDA) is the method involved with investigating connections between factors utilizing devices like dissipate plots, histograms, box plots, and relationship lattices. The essential EDA steps incorporate looking at the dataset to grasp its construction, distinguishing missing qualities, deciding information types, figuring outline measurements, picturing information disseminations, and investigating information circulations.



```
# Step 3: Exploratory Data Analysis (EDA)
# Summary statistics
summary_stats = data.describe()
print('\nSummary Statistics:')
print(summary_stats)
```

```
Summary Statistics:
        CustomerID         Home     Products         Blog        About  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000   
mean    500.500000    30.532000    28.821000    29.972000    30.689000   
std     288.819436    17.209915    16.770039    16.236096    17.187952   
min       1.000000     1.000000     1.000000     1.000000     1.000000   
25%     250.750000    16.000000    14.000000    17.000000    15.000000   
50%     500.500000    31.000000    28.000000    30.000000    32.000000   
75%     750.250000    46.000000    44.000000    43.250000    45.000000   
max    1000.000000    59.000000    59.000000    59.000000    59.000000   

           Contact  VisitFrequency  
count  1000.000000     1000.000000  
mean     30.615000       25.154000  
std      17.088323       14.381346  
min       1.000000        1.000000  
25%      16.000000       13.000000  
50%      31.000000       25.500000  
75%      46.000000       38.000000  
max      59.000000       49.000000  
```

Figure 6: Summary statistics

(Source: Observed and acquired by the learner)

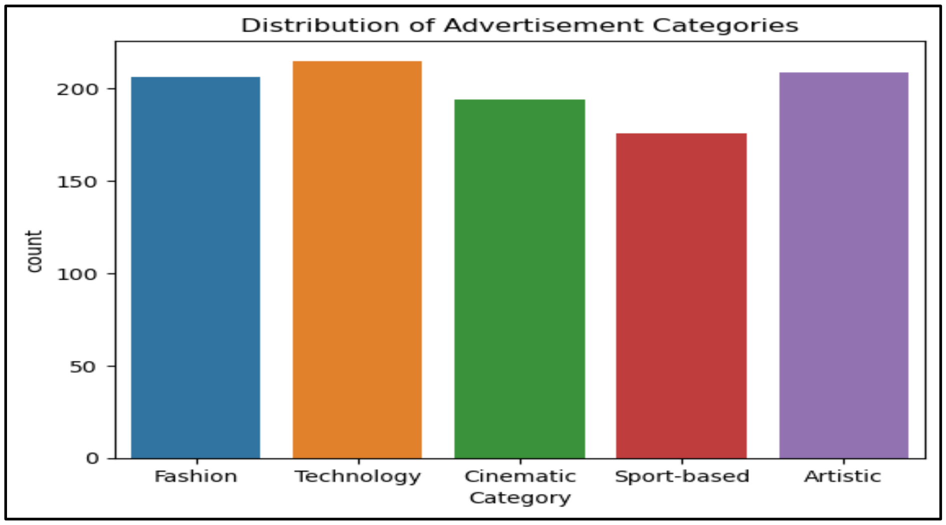
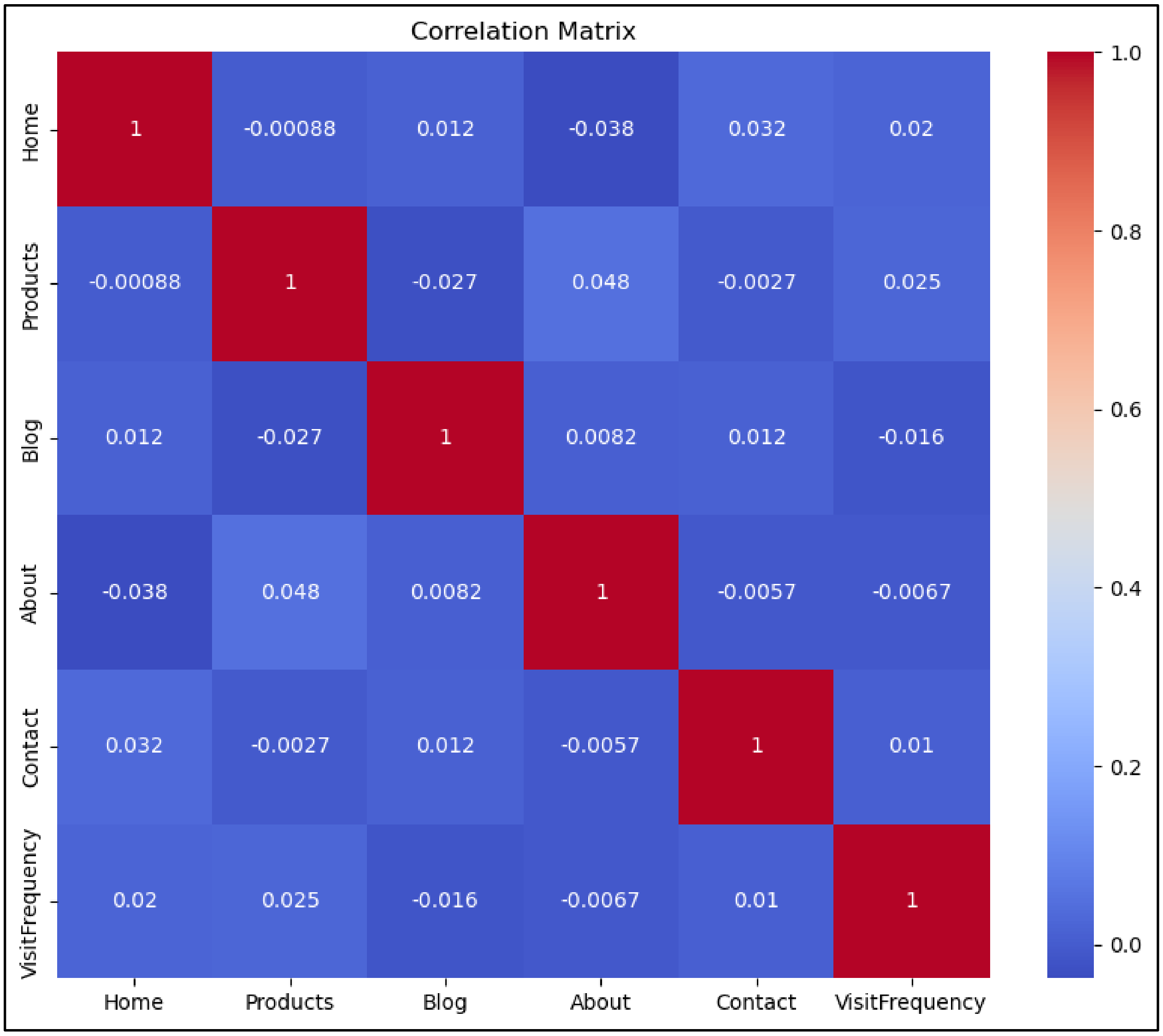


Figure 7: Distribution of Advertisement Categories

(Source: Observed and acquired by the learner)



	Home	Products	Blog	About	Contact	VisitFrequency
Home	1	-0.00088	0.012	-0.038	0.032	0.02
Products	-0.00088	1	-0.027	0.048	-0.0027	0.025
Blog	0.012	-0.027	1	0.0082	0.012	-0.016
About	-0.038	0.048	0.0082	1	-0.0057	-0.0067
Contact	0.032	-0.0027	0.012	-0.0057	1	0.01
VisitFrequency	0.02	0.025	-0.016	-0.0067	0.01	1

Figure 8: Correlation Matrix

(Source: Observed and acquired by the learner)

The pairwise relationships between factors in a dataset are shown in a square framework called a connection grid (ur Rehman *et al*. 2019). To recognize examples and conditions in the information, it gives a visual portrayal of the strength and course of the straight connection between factors.

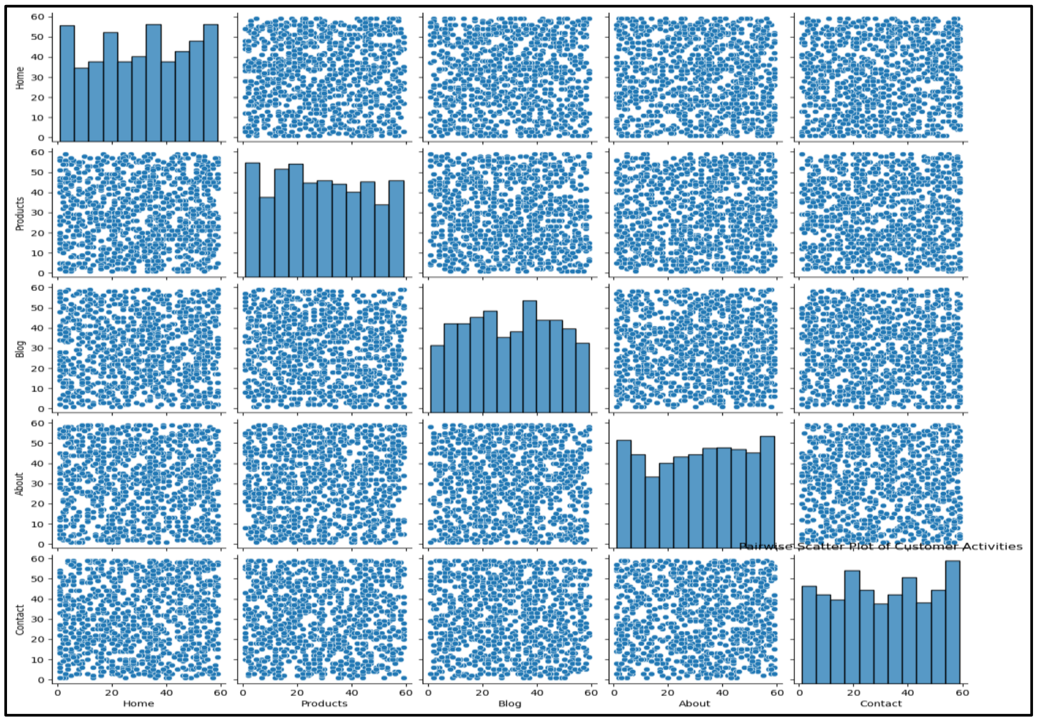


Figure 9: Pairwise Scatter Plot of Customer Activities

(Source: Observed and acquired by the learner)

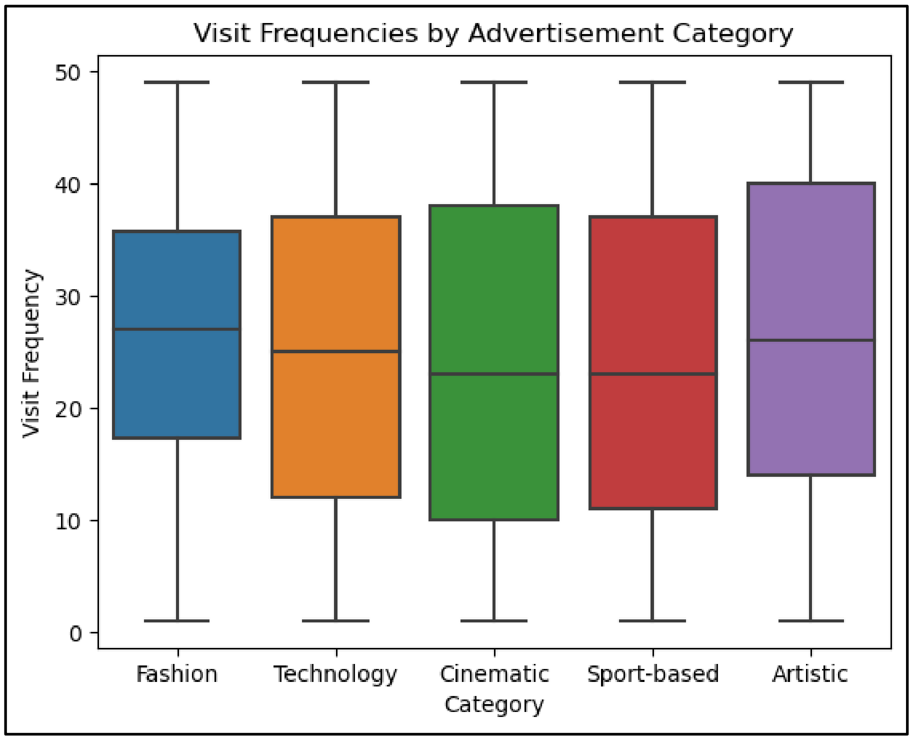


Figure 10: Visit Frequencies by Advertisement Category

(Source: Observed and acquired by the learner)

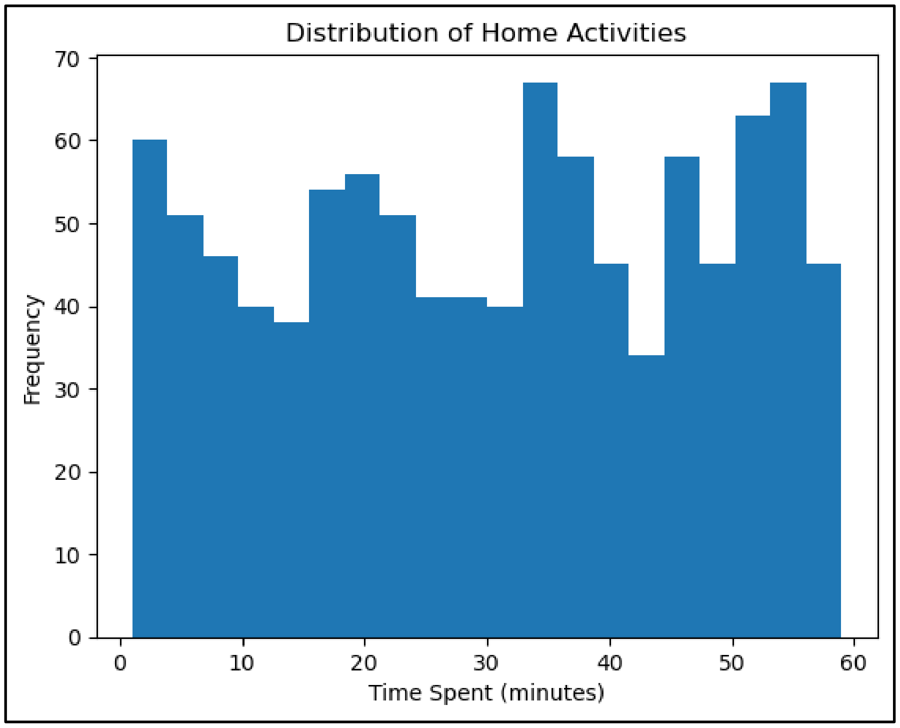
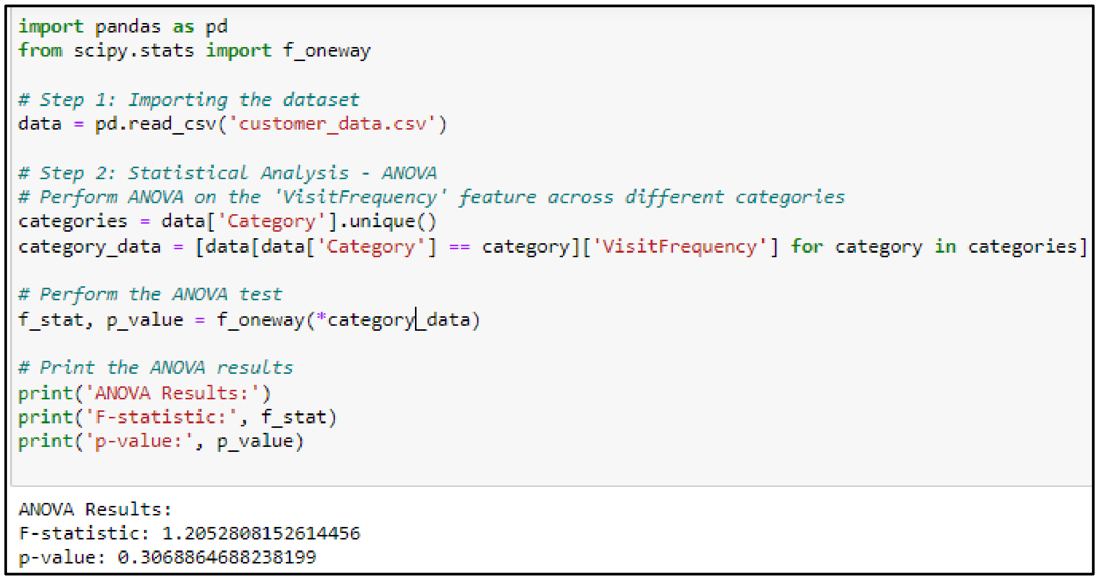


Figure 11: Distribution of Home Activities

(Source: Observed and acquired by the learner)



```
import pandas as pd
from scipy.stats import f_oneway

# Step 1: Importing the dataset
data = pd.read_csv('customer_data.csv')

# Step 2: Statistical Analysis - ANOVA
# Perform ANOVA on the 'VisitFrequency' feature across different categories
categories = data['Category'].unique()
category_data = [data[data['Category'] == category]['VisitFrequency'] for category in categories]

# Perform the ANOVA test
f_stat, p_value = f_oneway(*category_data)

# Print the ANOVA results
print('ANOVA Results:')
print('F-statistic:', f_stat)
print('p-value:', p_value)
```

```
ANOVA Results:
F-statistic: 1.2052808152614456
p-value: 0.3068864688238199
```

Figure 12: Statistical Analysis - ANOVA

(Source: Observed and acquired by the learner)

ANOVA (Examination of Change) is a measurable strategy used to look at the method for at least two gatherings to distinguish any distinctions that are genuinely critical.

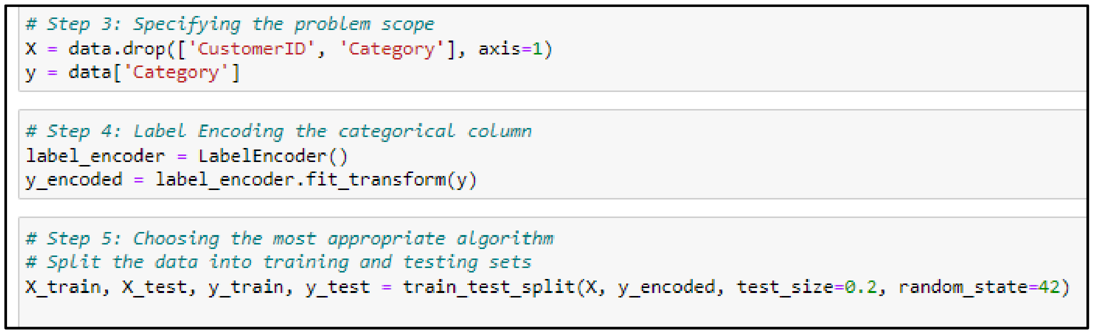
Problem Scope and Algorithm Selection

It is essential to specify the problem's scope and choose the right algorithm before employing data analytics to address a problem. Understanding the goals, limitations, and desired results of the analysis is part of the problem scope (Watson *et al*. 2019).

The qualities of the data that is currently available must also be taken into account. This entails evaluating the data's accuracy, comprehensiveness, and applicability to the issue at hand. The next step is to choose an appropriate algorithm after the problem's parameters and data have been established.

The assumptions and constraints connected to the technique of choice also influence the choice of an algorithm. Interpretability, scalability, computational effectiveness, and the capacity for handling big datasets are essential considerations.

It is crucial to pick a method that complies with the specifications of the problem, makes good use of the data at hand, and offers precise and trustworthy insights.



```python
# Step 3: Specifying the problem scope
X = data.drop(['CustomerID', 'Category'], axis=1)
y = data['Category']
```

```python
# Step 4: Label Encoding the categorical column
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

```python
# Step 5: Choosing the most appropriate algorithm
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
```

Figure 13: Specifying the problem scope

(Source: Observed and acquired by the learner)

Algorithm Selection and Model Building

In data analytics, choosing an algorithm and creating a model are essential phases since they dictate how to approach an issue and create a predictive or descriptive model. The primary steps in algorithm selection and model construction are listed below:

Data Preparation: Handling missing values, encoding categorical variables, normalising or scaling numerical features, and dividing the data into training and testing sets are all crucial steps in preprocessing the data before choosing an algorithm (Buhalis *et al*. 2021).

Define Evaluation Metrics: Choose the metrics that will be used to gauge how well the models are working. Depending on the type of problem (classification, regression, etc.) and the specific goals, this may contain measurements like accuracy, precision, recall, F1-score, mean squared error, or area under the ROC curve.

Algorithm Selection: Choose the best appropriate algorithm for the task based on the problem type, the data at hand, and the objectives. Think about things like model complexity, scalability, interpretability, and the capacity to manage the data attributes (Rialti *et al*. 2019).

Model Construction: Using the training data and the chosen algorithm, construct the model. This entails tuning the hyperparameters as needed and training the model to recognise patterns **Model evaluation:** Use the testing set to gauge the model's effectiveness. Determine the assessment criteria of choice to determine how well the model generalises to new data.

Model Improvement: If the model's performance is unsatisfactory, think about improving the model by changing its features, hyperparameters, or ensemble approaches.

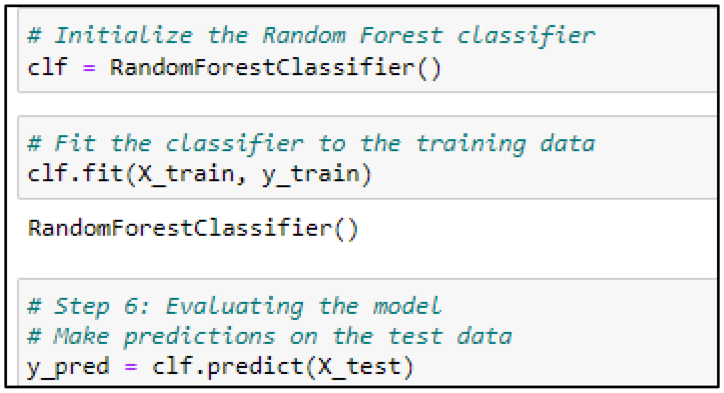
Cross-Validation: In order to make sure that the model is robust, divide the data into several folds and test the model on each fold.

Model Interpretation: Depending on the algorithm you've chosen, interpret the model to learn more about the key characteristics or factors that influence your forecasts.

Implementation: The model can be used in a production context for making predictions on fresh, unforeseen data after it has been developed and evaluated.

Model Monitoring and Maintenance: Monitor the performance of the deployed model continuously, and periodically retrain or update the model as new data becomes available (Aryal *et al*. 2020).

Overall, the problem, the data, and the assessment criteria must be carefully taken into account while choosing an algorithm and developing a model. The model is improved and refined through an iterative approach until it performs as anticipated and offers helpful information for making decisions.

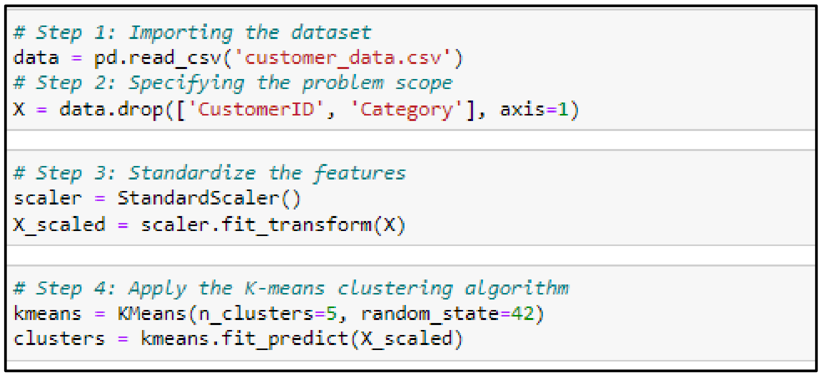


```
# Initialize the Random Forest classifier
clf = RandomForestClassifier()

# Fit the classifier to the training data
clf.fit(X_train, y_train)

RandomForestClassifier()

# Step 6: Evaluating the model
# Make predictions on the test data
y_pred = clf.predict(X_test)
```

Figure 14: Random Forest Classifier

(Source: Observed and acquired by the learner)

Widely used for classification problems, the Random Forest Classifier is an ensemble machine learning technique that blends various decision trees to create predictions.

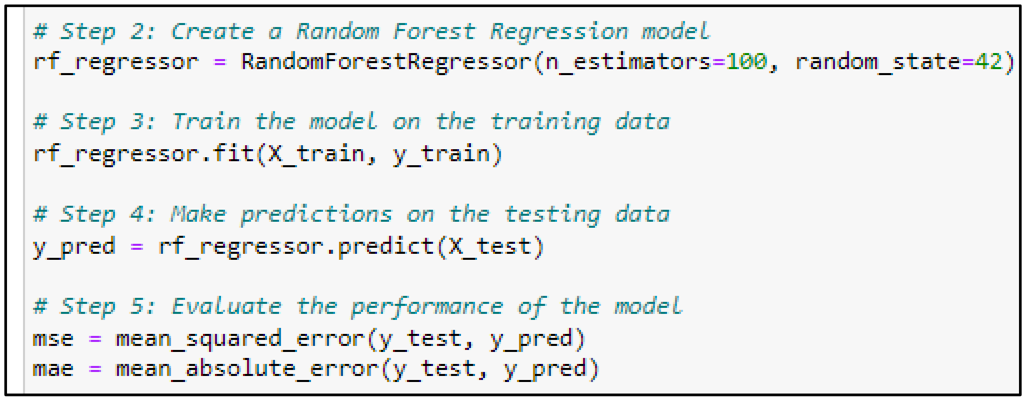


```
# Step 1: Importing the dataset
data = pd.read_csv('customer_data.csv')
# Step 2: Specifying the problem scope
X = data.drop(['CustomerID', 'Category'], axis=1)

# Step 3: Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 4: Apply the K-means clustering algorithm
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X_scaled)
```

Figure 15: KMeans clustering

(Source: Observed and acquired by the learner)

Data is divided into groups based on similarity using the unsupervised machine learning approach KMeans clustering, where each cluster is represented by its centroid. It is frequently employed in cluster analysis (Dremel *et al*. 2020).



```python
# Step 2: Create a Random Forest Regression model
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

# Step 3: Train the model on the training data
rf_regressor.fit(X_train, y_train)

# Step 4: Make predictions on the testing data
y_pred = rf_regressor.predict(X_test)

# Step 5: Evaluate the performance of the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
```

Figure 16: Random forest regressor

(Source: Observed and acquired by the learner)

A machine learning ensemble approach called the Random Forest Regressor combines several decision trees to produce predictions on continuous numerical data. Regression problems frequently make use of it.

Model Evaluation

A trained model's performance and dependability are evaluated as a crucial part of machine learning. It assists in determining the model's suitability for the intended job and how effectively it generalises to unobserved data (Ciampi *et al*. 2021). Depending on the nature of the issue, different evaluation metrics might be applied.

Common evaluation criteria for classification tasks include accuracy, precision, recall, and F1-score. Metrics like mean squared error (MSE) and mean absolute error (MAE) are frequently used to assess regression jobs. MSE gives more weight to larger errors by measuring the average squared difference between the predicted and actual values (Akter *et al*. 2019). MAE provides a more accurate picture of the mistakes by measuring the average absolute difference between the projected and actual numbers.

The Random Forest Classifier model performs relatively poorly, with an accuracy of only 26%, according to the evaluation measures. This shows that just 26% of the time do the model's predictions match the actual labels. The poor F1-score, precision, and recall also indicate difficulties in accurately detecting positive events.

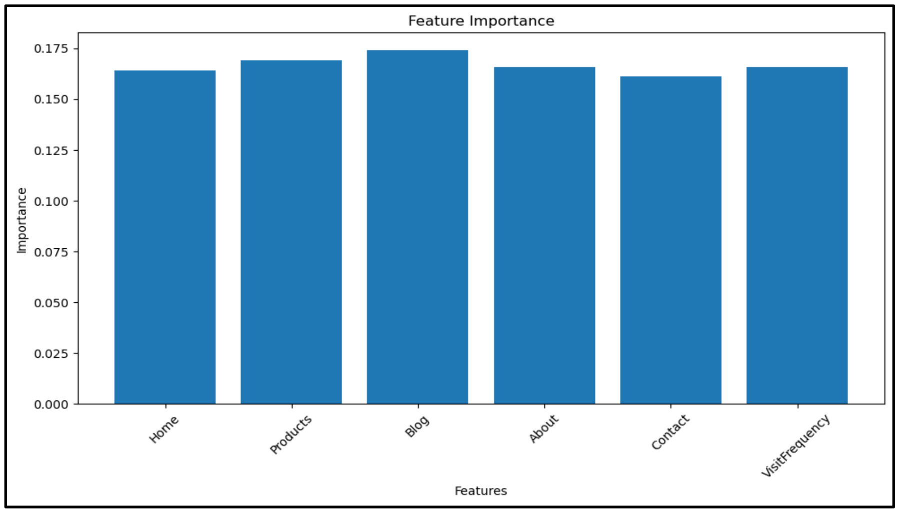
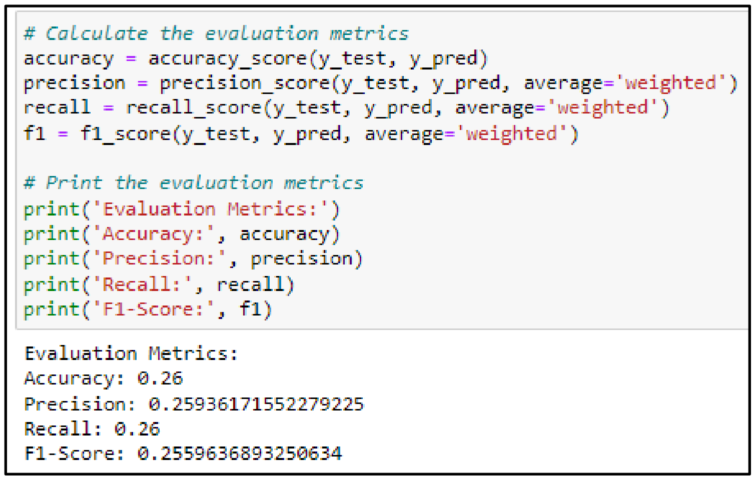


Figure 17: Feature Importance

(Source: Observed and acquired by the learner)



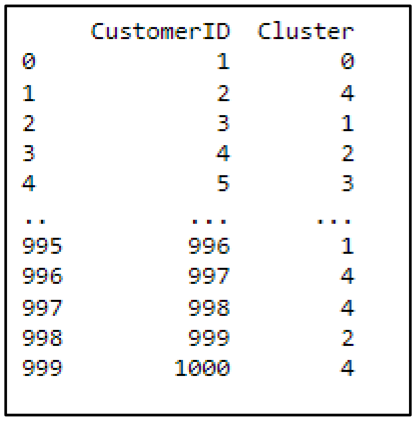
```
# Calculate the evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Print the evaluation metrics
print('Evaluation Metrics:')
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-Score:', f1)
```

```
Evaluation Metrics:
Accuracy: 0.26
Precision: 0.25936171552279225
Recall: 0.26
F1-Score: 0.2559636893250634
```

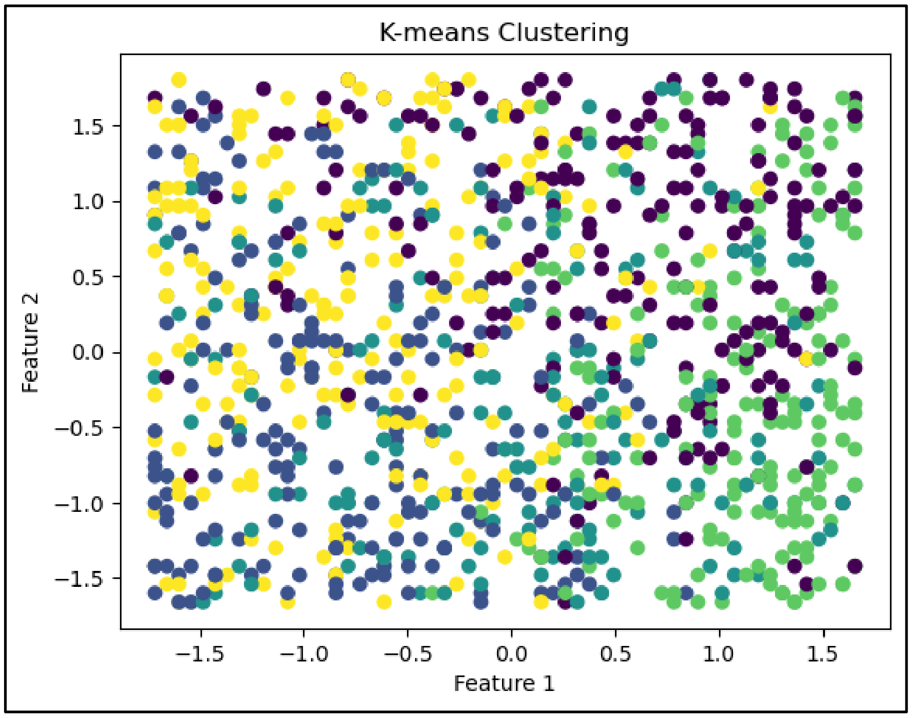
Figure 18: Random Forest Classifier model evaluation

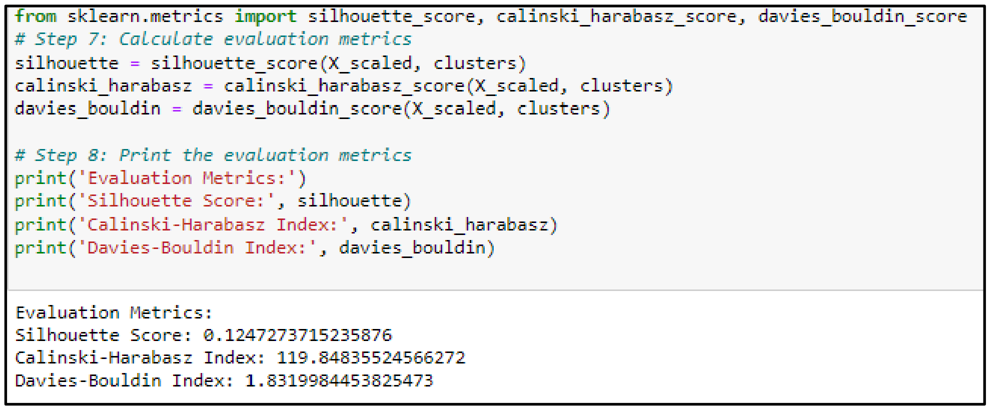
(Source: Observed and acquired by the learner)

The accuracy, precision, review, and F1-score of the arbitrary woodland classifier model were all sub optimal, at around 0.26. This proposes that the model experienced issues accurately ordering the information and that its forecasts were not exact. The presentation of the model should be worked on by extra innovative work.



```
     CustomerID  Cluster
0             1        0
1             2        4
2             3        1
3             4        2
4             5        3
..          ...      ...
995         996        1
996         997        4
997         998        4
998         999        2
999        1000        4
```





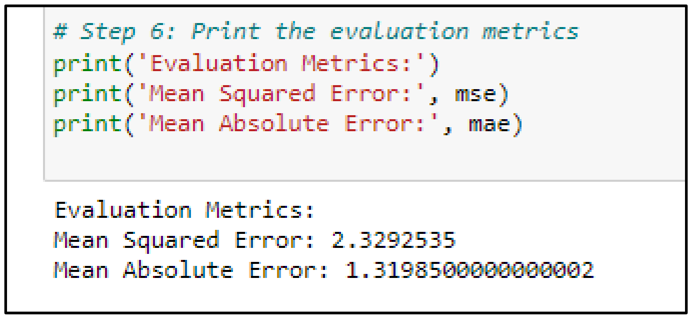
```
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
# Step 7: Calculate evaluation metrics
silhouette = silhouette_score(X_scaled, clusters)
calinski_harabasz = calinski_harabasz_score(X_scaled, clusters)
davies_bouldin = davies_bouldin_score(X_scaled, clusters)

# Step 8: Print the evaluation metrics
print('Evaluation Metrics:')
print('Silhouette Score:', silhouette)
print('Calinski-Harabasz Index:', calinski_harabasz)
print('Davies-Bouldin Index:', davies_bouldin)
```

```
Evaluation Metrics:
Silhouette Score: 0.1247273715235876
Calinski-Harabasz Index: 119.84835524566272
Davies-Bouldin Index: 1.8319984453825473
```

Figure 19: Clustering results

(Source: Observed and acquired by the learner)

The outline score of 0.1247, acquired by the KMeans bunching procedure, demonstrates that the groups are not obviously separated. The good cluster density and separation shown by the Calinski-Harabasz index of 119.85. The clusters are moderately different, as shown by the 1.832 Davies-Bouldin index.



```
# Step 6: Print the evaluation metrics
print('Evaluation Metrics:')
print('Mean Squared Error:', mse)
print('Mean Absolute Error:', mae)
```

```
Evaluation Metrics:
Mean Squared Error: 2.3292535
Mean Absolute Error: 1.3198500000000002
```

Figure 20: Regression evaluation

(Source: Observed and acquired by the learner)

The Random Forest Regression model performed averagely, with mean absolute error of 1.3199 and mean squared error of 2.3293. These metrics show a modest level of variation between the model's predictions and the actual values, with the mean squared error displaying the average squared difference and the mean absolute error displaying the average absolute difference between the two.

Conclusion

Data analytics plays a significant part in marketing systems by offering insights and useful information for wise decision-making. The evaluation's findings show the three models' advantages and disadvantages. There is a need for more advancements as the Random Forest Classifier struggles to categorise cases correctly. The Calinski-Harabasz Index indicates good clustering for the KMeans clustering model, however the Silhouette Score reveals fewer distinct groups.

These results imply that the Random Forest Regression model might be the best fit for the dataset and issue at hand. However, more research and testing should be done to confirm the efficacy of the models and investigate alternative viable algorithms for comparison. It makes data-driven decisions possible, raises client happiness, and helps marketing activities succeed overall. Businesses can obtain a competitive edge in today's dynamic and data-driven marketing environment by utilising data analytics strategies.

References

Akter, S., Bandara, R., Hani, U., Wamba, S.F., Foropon, C. and Papadopoulos, T., 2019. Analytics-based decision-making for service systems: A qualitative study and agenda for future research. *International Journal of Information Management*, *48*, pp.85-95.

Alsubari, S.N., Deshmukh, S.N., Alqarni, A.A., Alsharif, N., Aldhyani, T.H., Alsaade, F.W. and Khalaf, O.I., 2022. Data analytics for the identification of fake reviews using supervised learning. *Computers, Materials & Continua*, *70*(2), pp.3189-3204.

Araz, O.M., Choi, T.M., Olson, D.L. and Salman, F.S., 2020. Data analytics for operational risk management. *Decis. Sci.*, *51*(6), pp.1316-1319.

Aryal, A., Liao, Y., Nattuthurai, P. and Li, B., 2020. The emerging big data analytics and IoT in supply chain management: a systematic review. *Supply Chain Management: An International Journal*, *25*(2), pp.141-156.

Bhattarai, B.P., Paudyal, S., Luo, Y., Mohanpurkar, M., Cheung, K., Tonkoski, R., Hovsapian, R., Myers, K.S., Zhang, R., Zhao, P. and Manic, M., 2019. Big data analytics in smart grids: state-of-the-art, challenges, opportunities, and future directions. *IET Smart Grid*, *2*(2), pp.141-154.

Buhalis, D. and Volchek, K., 2021. Bridging marketing theory and big data analytics: The taxonomy of marketing attribution. *International Journal of Information Management*, *56*, p.102253.

Ciampi, F., Demi, S., Magrini, A., Marzi, G. and Papa, A., 2021. Exploring the impact of big data analytics capabilities on business model innovation: The mediating role of entrepreneurial orientation. *Journal of Business Research*, *123*, pp.1-13.

Dagilienė, L. and Klovienė, L., 2019. Motivation to use big data and big data analytics in external auditing. *Managerial Auditing Journal*.

Dong, J.Q. and Yang, C.H., 2020. Business value of big data analytics: A systems-theoretic approach and empirical test. *Information & Management*, *57*(1), p.103124.

Dremel, C., Herterich, M.M., Wulf, J. and Vom Brocke, J., 2020. Actualizing big data analytics affordances: A revelatory case study. *Information & Management*, *57*(1), p.103121.

Khanra, S., Dhir, A. and Mäntymäki, M., 2020. Big data analytics and enterprises: a bibliometric synthesis of the literature. *Enterprise Information Systems*, *14*(6), pp.737-768.

Rialti, R., Zollo, L., Ferraris, A. and Alon, I., 2019. Big data analytics capabilities and performance: Evidence from a moderated multi-mediation model. *Technological Forecasting and Social Change*, *149*, p.119781.

ur Rehman, M.H., Yaqoob, I., Salah, K., Imran, M., Jayaraman, P.P. and Perera, C., 2019. The role of big data analytics in industrial Internet of Things. *Future Generation Computer Systems*, *99*, pp.247-259.

Wang, J., Xu, C., Zhang, J. and Zhong, R., 2022. Big data analytics for intelligent manufacturing systems: A review. *Journal of Manufacturing Systems*, *62*, pp.738-752.

Watson, H.J., 2019. Update tutorial: Big Data analytics: Concepts, technology, and applications. *Communications of the Association for Information Systems*, *44*(1), p.21.