Coursework

Machine Learning and Data Mining

MSc Data Science

University of Salford MANCHESTER

MACHINE LEARNING AND DATA MINING

ASSIGNMENT

MOHAMMAD DANISH AHMAD

@00647331

14/12/2022

TASK 1: CLASSIFICATION

analysis using several classifications, including Azure Machine Learning, KNN, and decision trees. Predicting whether or not the consumer will purchase the car based on the provided metrics is the task's goal and aim. Based on characteristics like WHEEL-BASE, WIDTH, HEIGHT, CURB-WEIGHT, ENGINE-TYPE, BORE, STROKE, COMPRESSION-RATIO, HORSEPOWER, PEAK-RPM, and PRICE, it is possible to estimate if the consumer will purchase the automobile or not. In order to do this, a dataset containing the parts of the car will be investigated, the best features will be chosen, and various machine learning models will be built. For the purpose, models like decision trees and KNN will be used. The model's accuracy will be compared once it has been optimised using various parameters. The best model will be used to predict that which car a customer is going to buy.

INTRODUCTION

Cars are the preferred mode of transportation because they allow us to cover greater distances more quickly than walking. The first automobile was created in 1769 by Nicolas Joseph Cugnot in France, although it had only three wheels, was rather heavy, moved very slowly, and relied solely on steam. Later Nikolaus Otto, a German inventor, created the first gasoline-powered automobile in 1876. This technology was more dependable and practical than steam-powered vehicles, which needed hot water to function. Now, vehicles have been developed that run on both gasoline and electricity, which is a remarkably innovative development. Everyone's life has been greatly simplified by this technology, which allows people to travel long distances quickly and comfortably thanks to modern automobile features like air conditioning and comfortable seats. In the past, individuals would travel on animals or on foot, which would take them days or weeks to complete.

Decision tree

An unsupervised machine learning model called a decision tree is utilised for classification and regression. To comprehend the data and teach itself to forecast the target variable, it applied straightforward decision rules. These models are clear and straightforward to describe. They are incredibly easy to manage and require very little data preparation. This methodology can be applied to category and numerical data.

KNN (K-nearest Neighbours)

K Nearest Neighbor is a straightforward algorithm that sorts new information or instances based on similarities between them and all previously stored examples. It mostly serves to categorise a data point according on how its neighbours are categorised.

DATASETS

The dataset used in the task contains car’s data. It has 12 features. The features contain data of different diagnostic characteristics such as WHEEL-BASE, WIDTH, HEIGHT, CURB-WEIGHT, ENGINE-TYPE, BORE, STROKE, COMPRESSION-RATIO, HORSEPOWER, PEAK-RPM, PRICE. The target

variable is BUYING. This column contains the information whether the customer is going to buy i.e. YES or not i.e. NO. The data type of all the features are numerical except the BUYING which has character data type. As discussed above, the dataset has 12 features which are independent variables. They all have numerical data type. The dependent variable is BUYING. It has character data type. BUYING has two values, “YES” and “NO”. YES, states that the customer is going to buy the car and NO, states that the customer is not going to buy the car. The details of all the features are listed below.

Table 1.1: Attributes name and types

Column name	Data type	Column class
WHEEL-BASE	Numeric	Independent variable
BUYING	Character	Dependent variable
HEIGHT	Numeric	Independent variable
CURB-WEIGHT	Numeric	Independent variable
ENGINE-TYPE	Numeric	Independent variable
BORE	Numeric	Independent variable
STROKE	Numeric	Independent variable
COMPRESSION-RATIO	Numeric	Independent variable

HORSEPOWER	Numeric	Independent variable
PEAK-RPM	Numeric	Independent variable
PRICE	Numeric	Independent variable
WIDTH	Numeric	Independent variable

Table 1.1: Attributes name and types

EXPLANATION AND PREPARATION OF DATASETS:

Data Pre-processing

Before handling it for model training, the dataset required to be pre-processed because it contained a lot of information. Python is used to delete the unnecessary columns. But in our case, all the columns were useful so we did have to remove any column, but we should know the process of how we can remove the column which is not needed because every time the data are not same. So, the code for removing a column has been given below. The pre-processing also deals with null values and outliers. Different approaches can be used to deal with null values. During pre-processing, the dependent variable's data type was also modified. As Boruta requires the dependent variable to be in factor format, this was done to satisfy the requirements of feature selection.

```
cols = ['column name1','column name 2']
df = df.drop(cols, axis=1)
```

Figure 1.1: Code for removing the column in python

Implementation in Python

TASK: CLASSIFICATION (Using Decision Tree**)**

Figure 1.2: Workflow for Classification

The flowchart for all the work and implementation done in this classification is shown in the image above. Data processing, feature extraction, training data selection, decision and classification, classification output, post pressing, and accuracy assessment were performed after importing the data.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn as sk
import seaborn as sns
```

Table 1.2: Libraries used in python

The names of the libraries that were imported and used for the task are listed in the above table. As can be seen above, a total of five libraries were imported. The dataset was imported into Python using the read csv command after the library had been imported. Basic exploration was done after the data was imported utilising built-in functions. It provided a description of the data and its current state.

```
df.head()

df.tail()

df.describe(include='all')

df.describe()

df['BUYING'].value_counts()

df.info()

df.isnull().sum()
```

Above is some basic code used to explore data in python

Pre-processing was carried out after the fundamental structure of the data was established. We looked for any missing values in the data.

The dependent variable (BUYINIG), which includes the category value, was updated after the processing work was finished, and the data type was as well. YES, was changed to 0 and NO to 1, respectively. YES, indicates that the customer has bought the car, whereas NO indicates that the buyer has not. The codes are provided below.

```
df['BUYING']=df['BUYING'].replace(['YES','NO'],[0,1])

X=df.drop(columns='BUYING',axis=1)

Y=df['BUYING']
```

Splitting the dataset into the Training set and the Test set:

We will split our data to train and test dataset, for that we have used the following code for data splitting:

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=0)
```

Scaling features:

Features in our dataset come in a range of sizes and dimensions. The modelling of a dataset might be impacted by different sizes of the data characteristics. By taking away the mean and dividing by the standard deviation, we scaled the training data. For the scaling test data, we likewise used the same mean and standard deviation. Python's fit transform() method is applied to the training data so that we can scale it and discover the data's scaling parameters (i.e., the mean and variance of each of the features in the training data). Scaling was then applied to the test data as well. With the use of the convert() method, we can scale the test data using the same mean and variance that are obtained from our training data.

The Following code was used to standardize the train and test dataset:

```
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()

X_train_s=sc.fit_transform(X_train)

X_test_s=sc.transform(X_test)
```

Training the model:

After scaling the data, the data was trained using the following code:

```
from sklearn.tree import DecisionTreeClassifier

classifier=DecisionTreeClassifier(criterion='entropy',random_state = 0)

classifier.fit(X_train_s,Y_train)
```

Evaluating the model:

For evaluating the data, we used predict function. After training the data we need to evaluate our model to make predictions.

```
Y_pred=classifier.predict(X_test_s)

print(Y_pred)
```

Accuracy check:

To check our accuracy, I used confusion_matrix function. This function makes use of Y_pred and test data Y_test to create a matrix. The code used are given below:

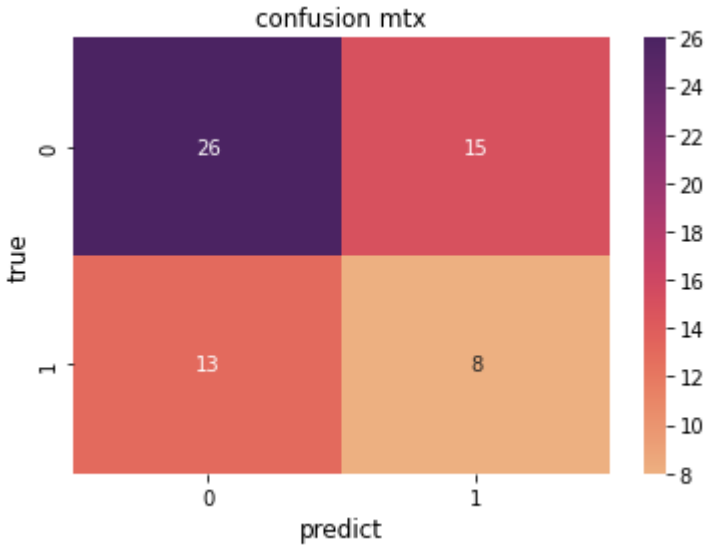
```
from sklearn import metrics

acc=metrics.accuracy_score(Y_test,Y_pred)

print('accuracy:%.2f\n\n'%(acc))

cm=metrics.confusion_matrix(Y_test,Y_pred)

print('confusion Matrix:')

print(cm,'\n\n')

print('--------------------')

result=metrics.classification_report(Y_test,Y_pred)

print('classification')

print(result)
```

Confusion Matrix using seaborn heatmap:

Codes and results are given below.

```
ax=sns.heatmap(cm,cmap='flare',annot=True,fmt='d')
```

```
plt.xlabel("predict",fontsize=12)
plt.ylabel("true",fontsize=12)
plt.title("confusion mtx",fontsize=12)
plt.show()
```



TASK: CLASSIFICATION (Using KNN**):**

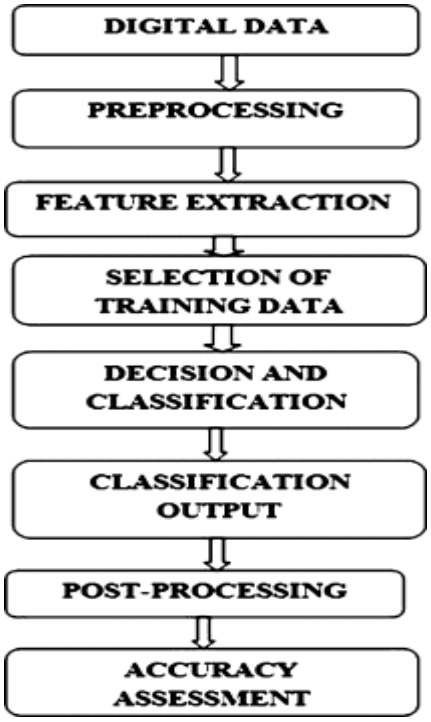


Figure 1.2: Workflow for Classification

The flowchart for all the work and implementation done in this classification is shown in the image above. Data processing, feature extraction, training data selection, decision and classification, classification output, post pressing, and accuracy assessment were performed after importing the data.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn as sk
import seaborn as sns
```

Table 1.2: Libraries used in python

The table above mentions the library names that were imported and used in the task. Total of five libraries were imported as seen above. After importing the library, the dataset was imported in python using read_csv command. Once the data was imported, basic exploration was performed using in-built functions. It showed us how the data is and a summary about the data.

```
df.head()

df.tail()

df.describe(include='all')

df.describe()
```

```
df['BUYING'].value_counts()

df.info()

df.isnull().sum()
```

Above is some basic code used to explore data in python

Once the basic structure of the data was known, the pre-processing was performed. The data were checked for missing values.

After completing the processing task, the dependent variable (BUYINIG) which contains the categorical value was changed, and the data type was also changed. YES, was replaced by 0 and NO was replaced by 1. YES, represents that the customer has purchase the car whereas NO represents that the customer has not purchased the car. The code for this are given below.

```
df['BUYING']=df['BUYING'].replace(['YES','NO'],[0,1])

X=df.drop(columns='BUYING',axis=1)

Y=df['BUYING']
```

Splitting the dataset into the Training set and the Test set:

We will split our data to train and test dataset, for that we have used the following code for data splitting:

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=0)
```

Scaling features:

Features in our dataset come in a range of sizes and dimensions. The modelling of a dataset might be impacted by different sizes of the data characteristics. By taking away the mean and dividing by the standard deviation, we scaled the training data. For the scaling test data, we likewise used the same mean and standard deviation. Python's fit transform() method is applied to the training data so that we can scale it and discover the data's scaling parameters (i.e., the mean and variance of each of the features in the training data). Scaling was then applied to the test data as well. With the use of the convert() method, we can scale the test data using the same mean and variance that are obtained from our training data.

The Following code was used to standardize the train and test dataset:

```
from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

X_train_s=sc.fit_transform(X_train)

X_test_s=sc.transform(X_test)
```

Training the model:

After scaling the data, the data was trained using the following code:

```
from sklearn.neighbors import KNeighborsClassifier

classifier=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)

classifier.fit(X_train_s,Y_train)
```

Evaluating the model:

For evaluating the data, we used predict function. After training the data we need to evaluate our model to make predictions.

```
Y_pred=classifier.predict(X_test_s)

print(Y_pred)
```

Accuracy check:

To check our accuracy, I used confusion_matrix function. This function makes use of Y_pred and test data Y_test to create a matrix. The code used are given below:

```
from sklearn import metrics

acc=metrics.accuracy_score(Y_test,Y_pred)

print('accuracy:%.2f\n\n'%(acc))

cm=metrics.confusion_matrix(Y_test,Y_pred)

print('confusion Matrix:')

print(cm,'\n\n')

print('--------------------')

result=metrics.classification_report(Y_test,Y_pred)

print('classification')

print(result)
```

The results are given below:

```
accuracy:0.55


confusion Matrix:
[[23 18]
 [10 11]]


--------------------
classification
              precision    recall  f1-score   support

           0       0.70      0.56      0.62        41
           1       0.38      0.52      0.44        21

    accuracy                           0.55        62
   macro avg       0.54      0.54      0.53        62
weighted avg       0.59      0.55      0.56        62
```

Confusion Matrix using seaborn heatmap:

Codes and results are given below.

```
ax=sns.heatmap(cm,cmap='flare',annot=True,fmt='d')

plt.xlabel("predict",fontsize=12)

plt.ylabel("true",fontsize=12)

plt.title("confusion mtx",fontsize=12)

plt.show()
```

RESULTS ANALYSIS AND DISCUSSION:

On the data, various classification techniques including Decision Tree and KNN were used. Our dataset fared well in both the categorization methods when the outcomes from the two processes were compared. In both instances, the outcomes and accuracy attained are comparable. The outcomes are listed below.

```
accuracy:0.55


confusion Matrix:
[[23 18]
 [10 11]]


--------------------
classification
              precision    recall  f1-score   support

           0       0.70      0.56      0.62        41
           1       0.38      0.52      0.44        21

    accuracy                           0.55        62
   macro avg       0.54      0.54      0.53        62
weighted avg       0.59      0.55      0.56        62
```

Figure: above shows the accuracy obtained from Decision Tree

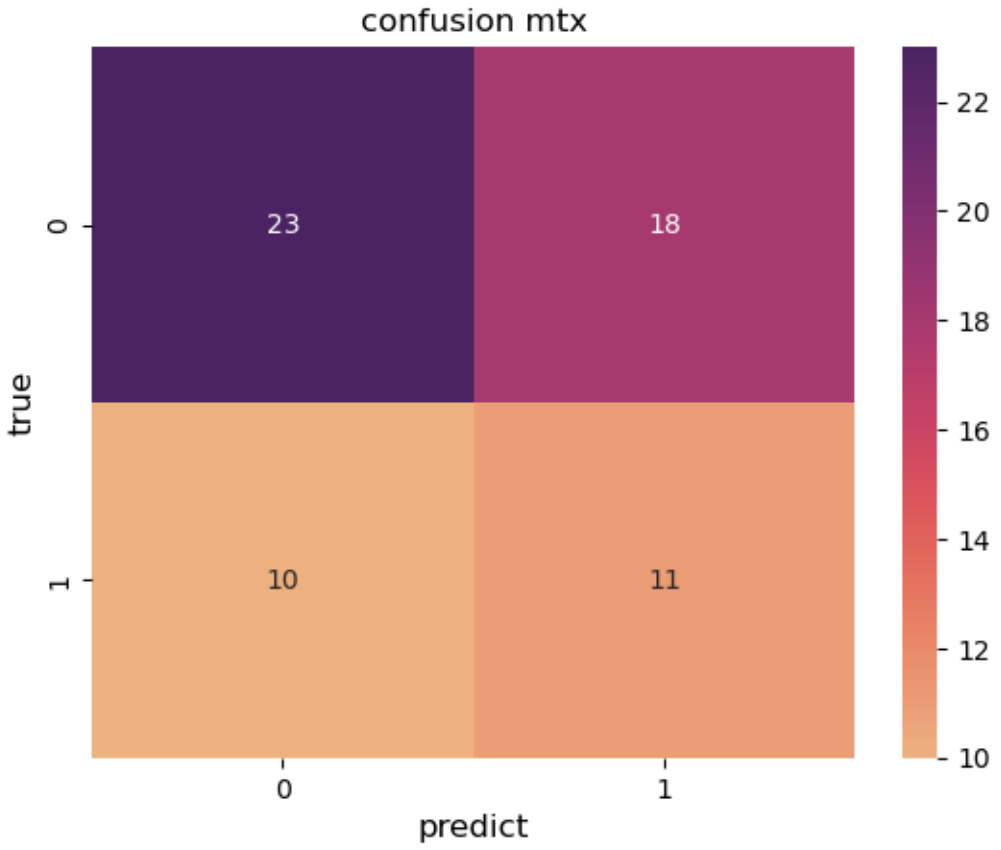


Figure: above shows the confusion matrix obtained from Decision Tree

```
accuracy:0.55


confusion Matrix:
[[26 15]
 [13  8]]


--------------------
classification
              precision    recall  f1-score   support

           0       0.67      0.63      0.65        41
           1       0.35      0.38      0.36        21

    accuracy                           0.55        62
   macro avg       0.51      0.51      0.51        62
weighted avg       0.56      0.55      0.55        62
```

Figure: above shows the accuracy obtained from KNN

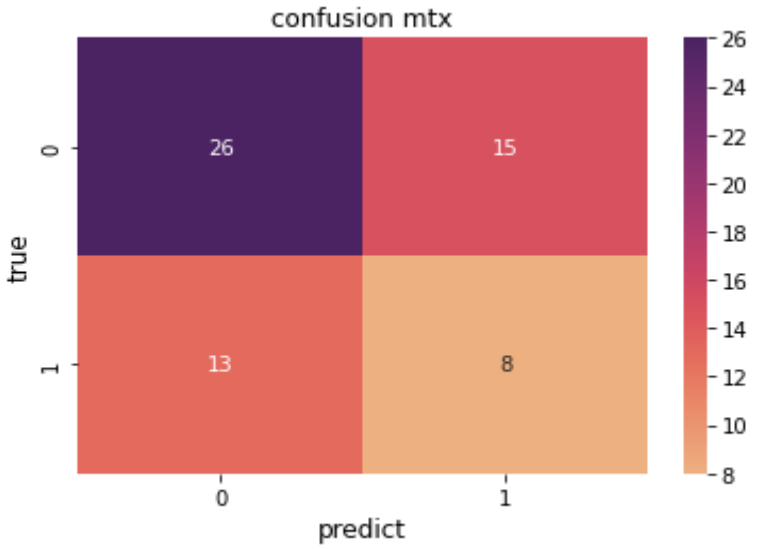


Figure: above shows the confusion matrix obtained from KNN

As we can see from both the Results, the accuracy obtained from both Decision tree and KNN is same which is 0.55. we can see some slight difference in precision, recall, and f1-score, but no difference in support. We can see some slight difference in confusion matrix as well.

As there is not much variation the both the results, we can say that our data performed well in both the classification method, and we can use both the classification method for our data.

TASK: CLASSIFICATION (Using AZURE MACHINE LEARNING**):**

AZURE MACHINE LEARING:

The lifespan of a machine learning project can be accelerated and managed using Azure Machine Learning, a cloud service. Professionals in machine learning, data scientists, and engineers can use it in their regular workflows: Manage MLOps while training and deploying models. Here, the categorization method will make advantage of Azure ML.

DATASETS

The dataset used in the task contains car's data. It has 12 features. The features contain data of different diagnostic characteristics such as WHEEL-BASE, WIDTH, HEIGHT, CURB-WEIGHT, ENGINE-TYPE, BORE, STROKE, COMPRESSION-RATIO, HORSEPOWER, PEAK-RPM, PRICE. The target variable is BUYING. This column contains the information whether the customer is going to buy i.e. YES or not i.e. NO. The data type of all the features are numerical except the BUYING which has character data type. As discussed above, the dataset has 12 features which are independent variables. They all have numerical data type. The dependent variable is BUYING. It has character data type. BUYING has two values, "YES" and "NO". YES, states that the customer is going to buy the car and NO, states that the customer is not going to buy the car. The details of all the features are listed below.

Table 1.1: Attributes name and types

Column name	Data type	Column class
WHEEL-BASE	Numeric	Independent variable
BUYING	Character	Dependent variable
HEIGHT	Numeric	Independent variable
CURB-WEIGHT	Numeric	Independent variable

ENGINE-TYPE	Numeric	Independent variable
BORE	Numeric	Independent variable
STROKE	Numeric	Independent variable
COMPRESSION-RATIO	Numeric	Independent variable
HORSEPOWER	Numeric	Independent variable
PEAK-RPM	Numeric	Independent variable
PRICE	Numeric	Independent variable
WIDTH	Numeric	Independent variable

Table 1.1: Attributes name and types

EXPLANATION AND PREPARATION OF DATASETS:

Data Pre-processing

Before handling it for model training, the dataset required to be pre-processed because it contained a lot of information. Python is used to eliminate the optional columns. Before sending the data to Azure, we changed the dependent variable, BUYING, to a range of 0 to 1. Additionally, integer was changed as the data type. The values of BUYING are "YES" and "NO." YES indicates that the customer is going to purchase the vehicle, whereas NO indicates that they are not. YES was changed to 0 and NO to 1, respectively.

Create an Azure Machine Learning workspace: First, we need to create a workspace in machine learning. Then we need to Create compute. Then we need to create a pipeline in Designer. After creating a designer, we need to import the data set.

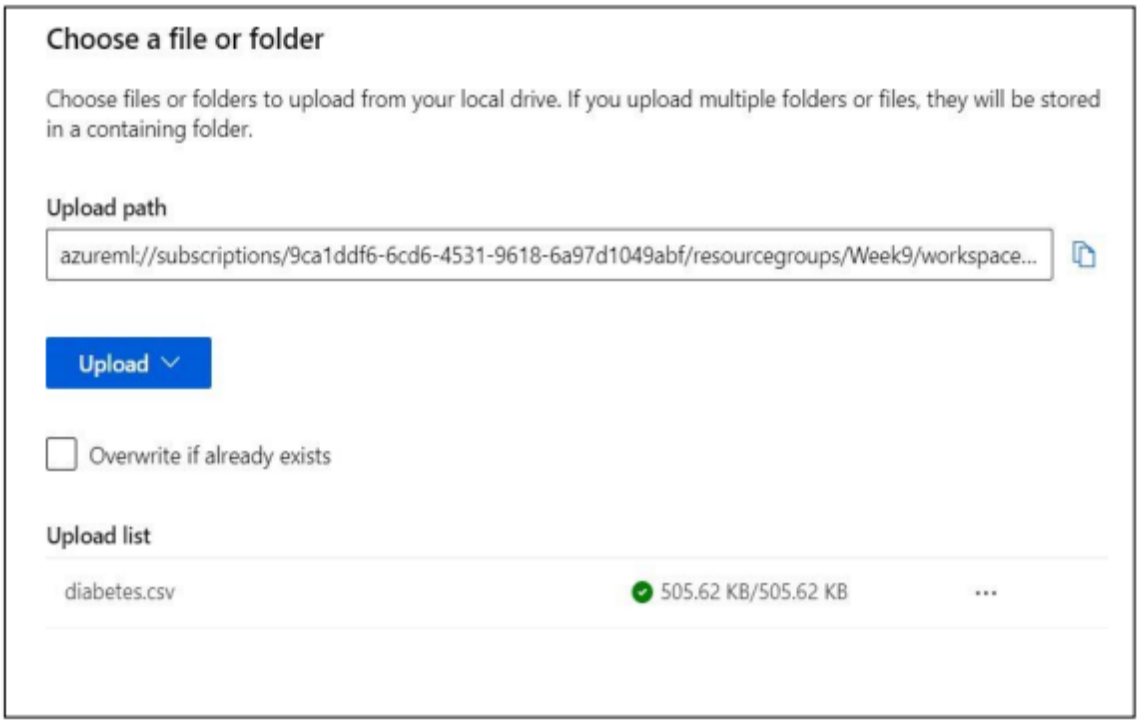
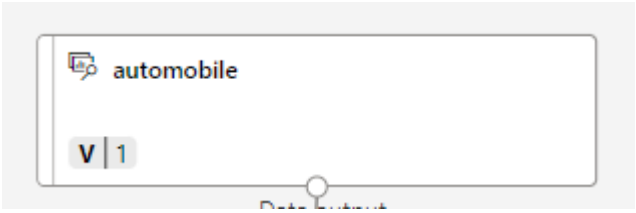


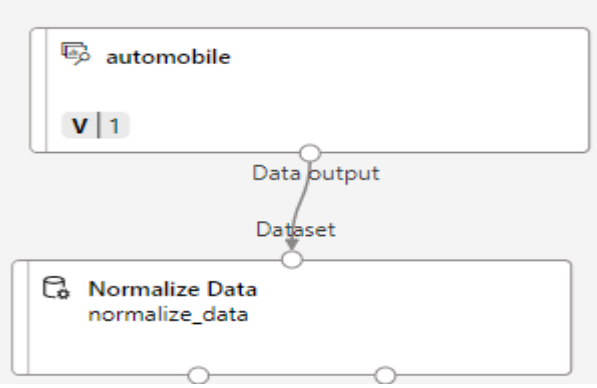
Figure: shows how we need to upload the data set from our device.

Load data to canvas

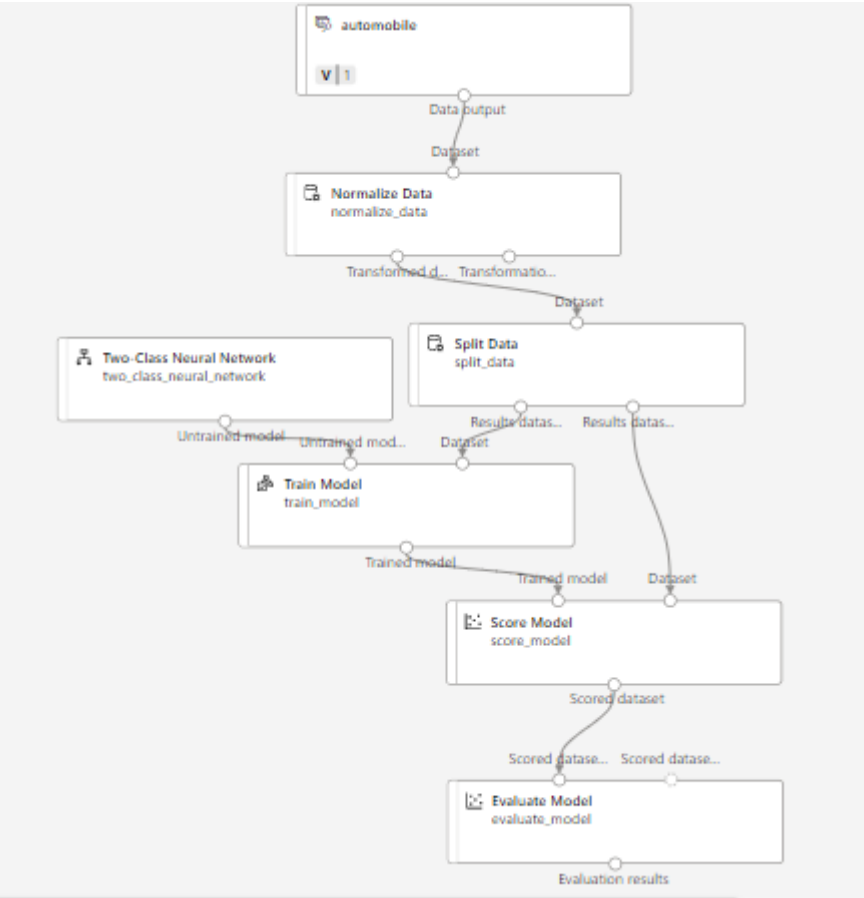
1. After loading the data, we come back to pipeline by selecting Designer. On the Designer page, we can see automobile select the automobile data to load on the canvas.



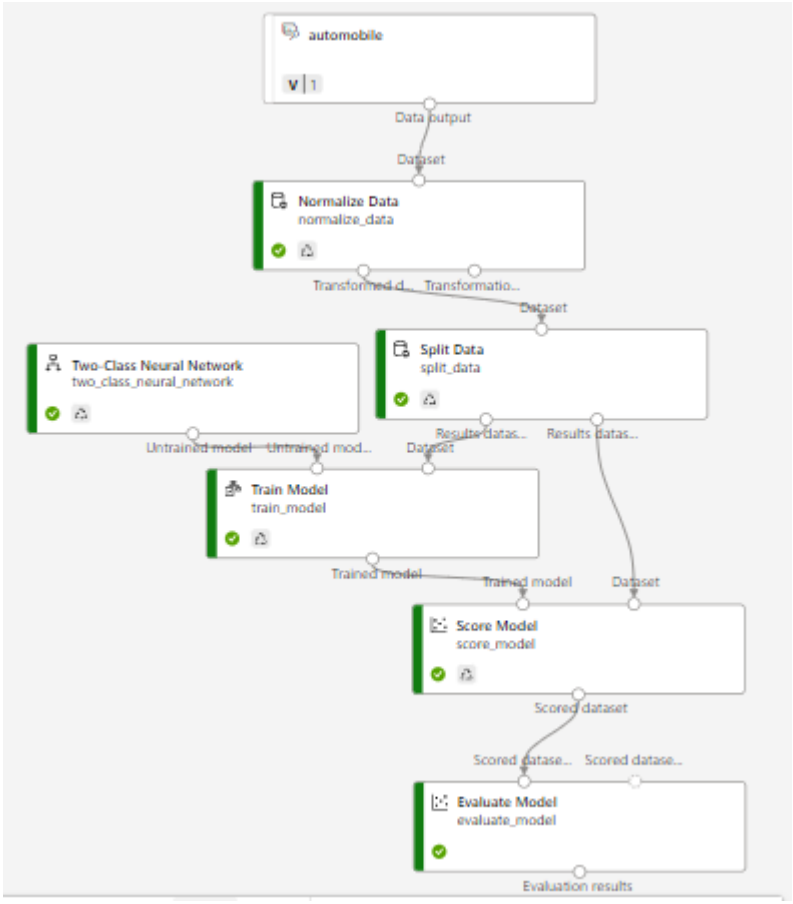
2. Look for the Normalize Data module in the component pane and place it on the canvas, below the automobile dataset. We need to do some basic settings after bringing normalize data to the canvas. For that double click on the normalize data and do some basic settings and column names that needs to be transformed.



3. Run the pipeline to apply our data transformations, we need to run the pipeline. After that we need to add the training module. After adding the training model submit the job.

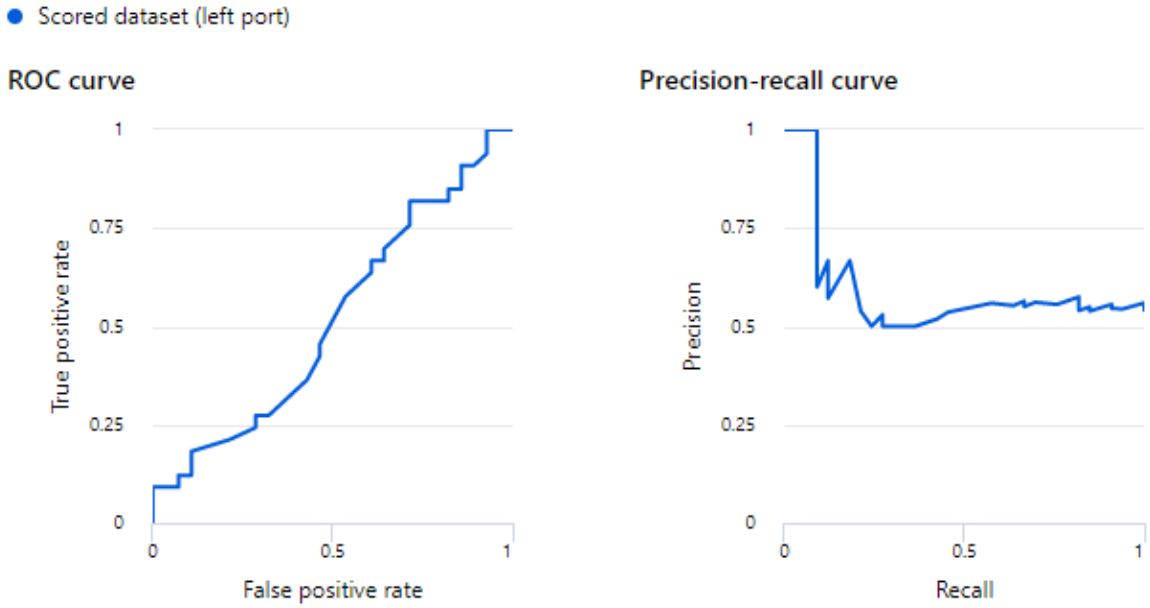


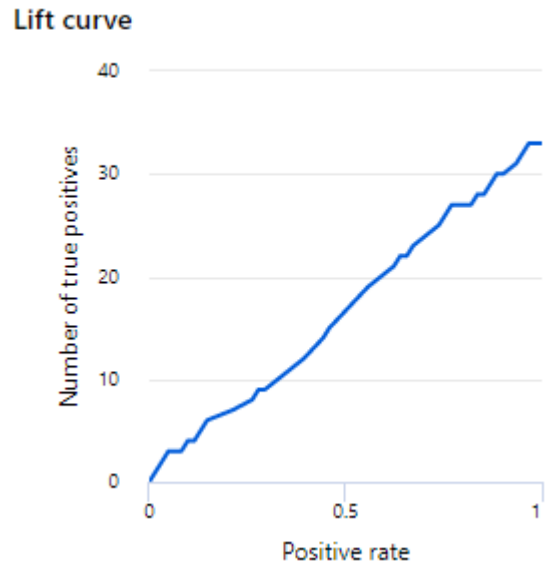
4. After the data transformation the pipeline will look like this.

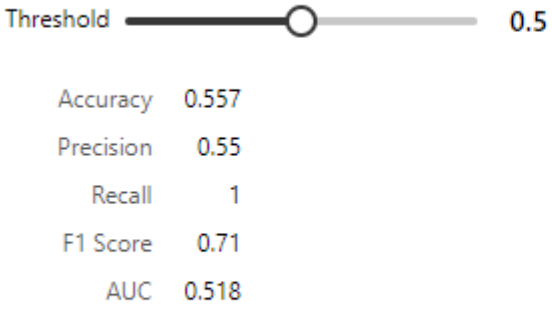


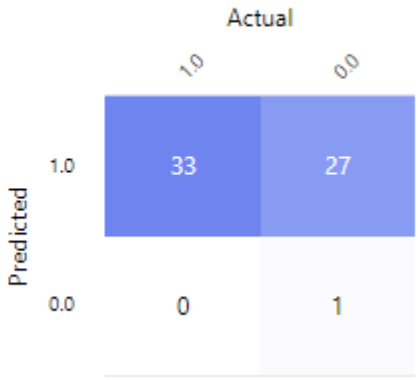
Results analysis and discussion

The results are:









As we can clearly see above in this Azure machine learning also, we have got accuracy 0.55 which is same we got in classification. The result from Python implementation and Azure are listed above. Both implementations have showed almost similar results.

CONCLUSIONS

People in our century place a high value on speed because they are all chasing after their goals and trying not to waste time on anything. As a result, cars are becoming more useful to them because they enable people to achieve their objectives by enabling them to arrive at their destinations early and without wasting any of their valuable time. Banks now assist people in purchasing automobiles through monthly payments because cars can be costly, allowing even middle-class families to buy one and benefit from the comfort it brings. As we can see, having a car has several advantages for us, including the fact that it keeps us independent in case of an emergency. Instead of waiting for the bus or taking another route, we can easily get where we're going. They are helpful for extended road trips with your friends, family, or by yourself to enjoy the cosy environment or discover new locations.

In this study, many models are created to assist individuals in selecting the best cars based on various car parameters. Information about Car was taken from a dataset. The features were chosen. KNN and Decision tree were trained as two models.

In Python, KNN and decision trees both demonstrated accuracy of 0.55 percent, and Azure demonstrated accuracy of 0.55 as well. Results in both software programmes were identical. The results varied according to the Azure confusion matrix, and there was also a small variation between the confusion matrices for the decision tree and the KNN. The outcome shows that Azure performed better than Decision tree and KNN in Python in terms of several criteria, including precision, recall, and f1-score. Therefore, we may conclude that Azure performed more effectively than the decision tree and KNN.

REFERENCES

1. *python - K Means Clustering: What does it mean about my input features if the Elbow Method gives me a straight line?* (n.d.). Stack Overflow. Retrieved December 20, 2022, from https://stackoverflow.com/questions/61820103/k-means-clustering-what-does-it-mean-about-my-input-features-if-the-elbow-metho
2. Kshitiz Sirohi. (2018, December 30). *K-nearest Neighbors Algorithm with Examples in R (Simply Explained knn)*. Medium; Towards Data Science. https://towardsdatascience.com/k-nearest-neighbors-algorithm-with-examples-in-r-simply-explained-knn-1f2c88da405
3. *College Essay about Cars: Topics, Tips & Ideas*. (n.d.). Knowledge Base. Retrieved December 20, 2022, from https://custom-writing.org/blog/essays-on-cars
4. *What is the k-nearest neighbors algorithm? | IBM*. (n.d.). Www.ibm.com. https://www.ibm.com/uk-en/topics/knn
5. *essay on car - Search*. (n.d.). Www.bing.com. Retrieved December 20, 2022, from https://www.bing.com/search?q=essay+on+car&cvid=3663bdf07cf44a9eb1ecc201a021bf29&aqs=edge.1.69i59i450l8...8.60762610j0j1&FORM=ANNTA1&PC=U531&ntref=1

6. sdgilley. (n.d.). *What is Azure Machine Learning? - Azure Machine Learning*. Learn.microsoft.com. https://learn.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-machine-learning

APPENDIX

#important library used

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn as sk
import seaborn as sns
```

#Reading the file in in python

```
df=pd.read_csv('Automobile_Data.csv')
```

 # Converting the dependent value to 0-1

```
df['BUYING']=df['BUYING'].replace(['YES','NO'],[0,1])
X=df.drop(columns='BUYING',axis=1)
Y=df['BUYING']
```

#Training the data

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=0)
```

#Training the model for Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion='entropy',random_state = 0)
classifier.fit(X_train_s,Y_train)
```

#Training the model for KNN:

```
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
classifier.fit(X_train_s,Y_train)
```

 # Creating confusion matrix:

```
from sklearn import metrics
acc=metrics.accuracy_score(Y_test,Y_pred)
print('accuracy:%.2f\n\n'%(acc))
cm=metrics.confusion_matrix(Y_test,Y_pred)
print('confusion Matrix:')
print(cm,'\n\n')
print('--------------------')
result=metrics.classification_report(Y_test,Y_pred)
print('classification')
print(result)
```

TASK 2 – ASSOCIATION RULE MINING

TITLE

Analysis using Association rule mining. In this we have taken some dada and created a data frame on our own. In this our aim will be to analyse the association of purchased items in a single basket or single purchase.

INTRODUCTION

Finding the rules that might control relationships between groups of things is done through a process called association rule mining. In every transaction involving numerous commodities, market basket analysis—a particular application of association rules mining that many businesses utilise for a number of purposes—seeks out the principles that specify how or why such items are frequently purchased together. Association rules are rules that go beyond a minimal support and confidence threshold that the user has set.

Association rules are frequently employed in recommender systems to market basket analysis. Recommender systems are employed by numerous online service providers, including Amazon and Netflix. Association rules can be used by recommender systems to find related items or locate clients with comparable interests. For instance, association rules can imply that clients who purchased product A also purchased product B, or that clients who purchased items A, B, and C are more akin to this client. These findings offer retailers chances to cross-sell their goods. Huge volumes of information are contained in web usage log files created on web servers, and association rules may provide web usage data analyzers with relevant information.

Support:

The percentage of transactions in the dataset that contain the item set is known as the support supp(X) of an item set X.

$$support(B) = Transactions\ containing\ B / Total\ Transactions$$

Confidence:

The formula for a rule's confidence is conf (X Y) = supp(X Y)/supp (X). As a result, the association rule X Y will be met.

$$confidence(A \rightarrow B) = Transactions\ containing\ A\ and\ B\ / Transactions\ containing\ A$$

Lift:

The lift of a rule is defined as:

$$lift(A \rightarrow B) = support(A\ and\ B) / support(A) \times support(B)$$

DATASETS

We have created a data set which contains item from supermarket. The item includes which are some common items bought in supermarket. The data set are given below.

```
Data_set = [['Chicken','Butter','Bread','Milk','Potato'],
            ['Milk','Onion','Potato','Jam','Peanut_Butter'],
            ['Milk','Bread','Butter','Apple','Eggs'],
            ['Tomato','Chilli','Bread','Butter','Juice'],
            ['Potato','Chicken','Tomato','Onion','Eggs','Juice']]
```

EXPLANATION AND PREPARATION OF DATASETS

Data Pre-processing

The dataset contains some unnecessary junk data that needs to be pre-processed and remove unnecessary data which is not useful before transferring the data to the training model. The columns that are not required are removed using Python. But in our case, as we have created our dataset, we don't need to do any pre-processing, but we should know the process of how we can remove of column if not needed because every time

the data are not same. So, the code for removing a column in python has been given below.

```
cols = ['column name1','column name 2']
df = df.drop(cols, axis=1)
```

Figure: codes for removing column in python

Our data set looks like this.

```
Data_set
```

```
[['Chicken', 'Butter', 'Bread', 'Milk', 'Potato'],
 ['Milk', 'Onion', 'Potato', 'Jam', 'Peanut_Butter'],
 ['Milk', 'Bread', 'Butter', 'Apple', 'Eggs'],
 ['Tomato', 'Chilli', 'Bread', 'Butter', 'Juice'],
 ['Potato', 'Chicken', 'Tomato', 'Onion', 'Eggs', 'Juice']]
```

Implementation in Python

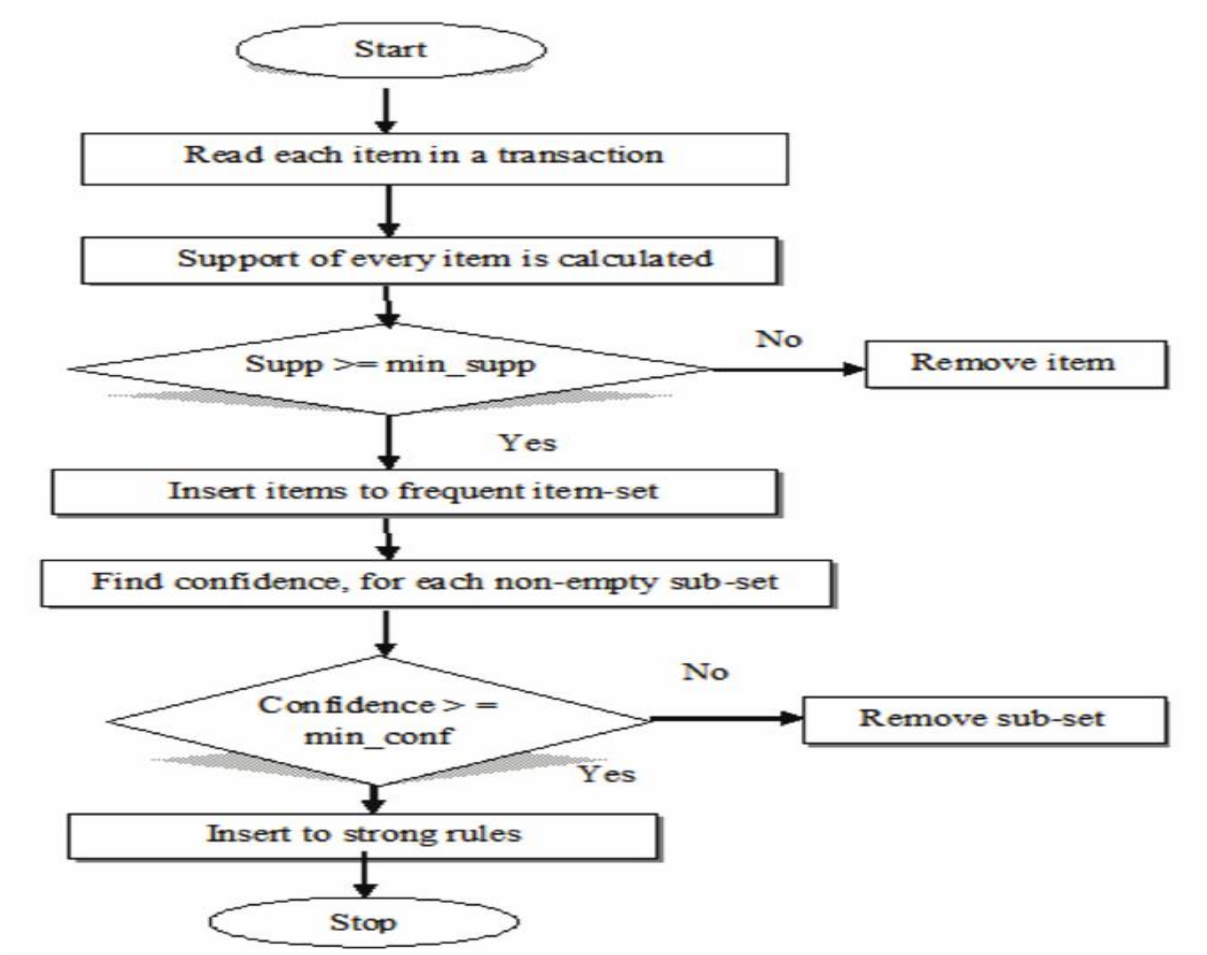


Figure: shows a workflow of Association rule mining

The figure above shows the flowchart of the all the work and implementation done in this Association rule mining. First the data set was created and then and then with the

help of transaction encoder method the data was converted into and array and then the confidence, support, lift etc was determined.

```
import pandas as pd
from      mlxtend.preprocessing
import TransactionEncoder
```

Figure: libraries used

The table above mentions the library names that were imported and used in the task. Total of two libraries were imported as seen above.

```
tra = TransactionEncoder()
tra_array = tra.fit(Data_set).transform(Data_set)
```

We can convert the dataset into an array format that is appropriate for our dataset by using a TransactionEncoder object. The input dataset is transformed into a one-hot encoded NumPy Boolean array by the fit method, the TransactionEncoder learns the distinct labels in the dataset, and the transform method:

```
data = pd.DataFrame(tra_array, columns= tra.columns_)
data
```

	Apple	Bread	Butter	Chicken	Chilli	Eggs	Jam	Juice	Milk	Onion	Peanut_Butter	Potato	Tomato
0	False	True	True	True	False	False	False	False	True	False	False	True	False
1	False	False	False	False	False	False	True	False	True	True	True	True	False
2	True	True	True	False	False	True	False	False	True	False	False	False	False
3	False	True	True	False	True	False	False	True	False	False	False	False	True
4	False	False	False	True	False	True	False	True	False	True	False	True	True

Above we can see the data frame created using the transaction encoder which converted all the values into boolean values and then we created data frame using the pd.DataFrame.

```
from mlxtend.frequent_patterns import apriori
bought_items= apriori(data,min_support = 0.4, use_colnames = True)
bought_items
```

After creating the dataframe we import apriori then we are generating frequent bought items(bought_items) sets witht the help of apriori. And in the apriori we send the same data sets that we created above, and we are sending the minimum support value 0.4 that is the threshold value. This will filter out where the support item is more than 0.4.

Now we will use the Association rule mining and for that we will import the association_rule from the mlxtend.frequent_patterns. Then we will be calling association rule and in that we will pass bought items that we have already created above, and we are giving metrics = confidence and min_threshold=0.6. This will give us the items which have confidence more than 0.6. The code is given below:

```
from mlxtend.frequent_patterns import association_rules

Ass_rule=association_rules(bought_items,metric="confidence",
min_threshold = 0.6)

Ass_rule
```

Now we will filter out on the basis of confidence. We will filter out the items which have confidence greater than the 0.7. the codes are given below.

```
Ass_rule= Ass_rule[Ass_rule['confidence']>0.7]

Ass_rule
```

Now we filter out the columns which are only useful for us and the code for that are given below.

```
Ass_rule1= Ass_rule[['antecedents','consequents','support','confidence','lift']]

Ass_rule1
```

RESULTS ANALYSIS AND DISCUSSION:

Association Rule were carried out on the data set using Mlextend and association rule. The results obtained are given below:

```
data = pd.DataFrame(tra_array, columns= tra.columns_)
data
```

	Apple	Bread	Butter	Chicken	Chilli	Eggs	Jam	Juice	Milk	Onion	Peanut_Butter	Potato	Tomato
0	False	True	True	True	False	False	False	False	True	False	False	True	False
1	False	False	False	False	False	False	True	False	True	True	True	True	False
2	True	True	True	False	False	True	False	False	True	False	False	False	False
3	False	True	True	False	True	False	False	True	False	False	False	False	True
4	False	False	False	True	False	True	False	True	False	True	False	True	True

Array that we created and then transform it into dataframe using the transaction encoder and fit method.

```
from mlxtend.frequent_patterns import apriori
bought_items= apriori(data,min_support = 0.4, use_colnames = True)
bought_items
```

	support	itemsets
0	0.6	(Bread)
1	0.6	(Butter)
2	0.4	(Chicken)
3	0.4	(Eggs)
4	0.4	(Juice)
5	0.6	(Milk)
6	0.4	(Onion)
7	0.6	(Potato)
8	0.4	(Tomato)
9	0.6	(Bread, Butter)
10	0.4	(Bread, Milk)
11	0.4	(Butter, Milk)
12	0.4	(Potato, Chicken)
13	0.4	(Tomato, Juice)
14	0.4	(Potato, Milk)
15	0.4	(Potato, Onion)
16	0.4	(Bread, Butter, Milk)

The above result shows the bought items that has minimum support greater than the 0.4.

```
from mlxtend.frequent_patterns import association_rules
Ass_rule = association_rules(bought_items, metric="confidence", min_threshold = 0.6)
Ass_rule
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Bread)	(Butter)	0.6	0.6	0.6	1.000000	1.666667	0.24	inf
1	(Butter)	(Bread)	0.6	0.6	0.6	1.000000	1.666667	0.24	inf
2	(Bread)	(Milk)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
3	(Milk)	(Bread)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
4	(Butter)	(Milk)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
5	(Milk)	(Butter)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
6	(Potato)	(Chicken)	0.6	0.4	0.4	0.666667	1.666667	0.16	1.8
7	(Chicken)	(Potato)	0.4	0.6	0.4	1.000000	1.666667	0.16	inf
8	(Tomato)	(Juice)	0.4	0.4	0.4	1.000000	2.500000	0.24	inf
9	(Juice)	(Tomato)	0.4	0.4	0.4	1.000000	2.500000	0.24	inf
10	(Potato)	(Milk)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
11	(Milk)	(Potato)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
12	(Potato)	(Onion)	0.6	0.4	0.4	0.666667	1.666667	0.16	1.8
13	(Onion)	(Potato)	0.4	0.6	0.4	1.000000	1.666667	0.16	inf
14	(Bread, Butter)	(Milk)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
15	(Bread, Milk)	(Butter)	0.4	0.6	0.4	1.000000	1.666667	0.16	inf
16	(Butter, Milk)	(Bread)	0.4	0.6	0.4	1.000000	1.666667	0.16	inf
17	(Bread)	(Butter, Milk)	0.6	0.4	0.4	0.666667	1.666667	0.16	1.8
18	(Butter)	(Bread, Milk)	0.6	0.4	0.4	0.666667	1.666667	0.16	1.8
19	(Milk)	(Bread, Butter)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2

Above shows the results that we got after applying the association rule. The items are shortlisted on the basis of confidence, only items are included whose confidence is grater than the min_threshold which is > 0.6. Here we can see antecedents, consequents, antecedent_support, consequent_support, support, confidence, lift, leverage and conviction. Here we can see some items in antecedent and consequents. For example, in first row we can see that bread is under antecedents and the antecedents support is 0.6 and in consequents column butter is there and consequent support s allso0.6 and the support which is a combination of both has a value of 0.6 as well. For some we can see that the support value is < than the consequent and antecedent support.

```
Ass_rule= Ass_rule[Ass_rule['confidence']>0.7]
Ass_rule
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Bread)	(Butter)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
1	(Butter)	(Bread)	0.6	0.6	0.6	1.0	1.666667	0.24	inf
7	(Chicken)	(Potato)	0.4	0.6	0.4	1.0	1.666667	0.16	inf
8	(Tomato)	(Juice)	0.4	0.4	0.4	1.0	2.500000	0.24	inf
9	(Juice)	(Tomato)	0.4	0.4	0.4	1.0	2.500000	0.24	inf
13	(Onion)	(Potato)	0.4	0.6	0.4	1.0	1.666667	0.16	inf
15	(Bread, Milk)	(Butter)	0.4	0.6	0.4	1.0	1.666667	0.16	inf
16	(Butter, Milk)	(Bread)	0.4	0.6	0.4	1.0	1.666667	0.16	inf

Above we have filter out the items whose confidence value is grater than 0.7.

```
Ass_rule1= Ass_rule[['antecedents','consequents','support','confidence','lift']]
Ass_rule1
```

	antecedents	consequents	support	confidence	lift
0	(Bread)	(Butter)	0.6	1.0	1.666667
1	(Butter)	(Bread)	0.6	1.0	1.666667
7	(Chicken)	(Potato)	0.4	1.0	1.666667
8	(Tomato)	(Juice)	0.4	1.0	2.500000
9	(Juice)	(Tomato)	0.4	1.0	2.500000
13	(Onion)	(Potato)	0.4	1.0	1.666667
15	(Bread, Milk)	(Butter)	0.4	1.0	1.666667
16	(Butter, Milk)	(Bread)	0.4	1.0	1.666667

Above we have just generated a subset of columns which are only important to us. In this we can see the antecedents, consequents, support, confidence and lift.

DISCUSSION:

Once the apriori and association rules have been created, we may create a data set that contains just those item sets that contain all of the values, such as support, confidence, lift, etc. We can assist in the market basket analysis based on these item sets that are antecedents and consequents. When we have this much information, we can advise the supermarket or retail establishment on how to organise the products and which ones to pair together in order to increase sales.

CONCLUSIONS:

Association rule mining helped us understand which are the most frequent bought item and which are not. Some items showed higher lift, support and confidence value while some showed lower value. This analysis helps us understand better that which item should be placed together and on which item the store should be more focusing to increase their sales. Association rules help us in increasing the sales.

REFERENCES:

1. *Association Rule Mining: An Overview and its Applications*. (2022, September 29). UpGrad Blog. https://www.upgrad.com/blog/association-rule-mining-an-overview-and-its-applications/#:~:text=Association%20Rule%20Mining%2C%20as%20the%20name%20suggests%2C%20association
2. Mathur, V. (n.d.). *Association Rule Mining: Importance and Steps | Analytics Steps*. Www.analyticssteps.com. https://www.analyticssteps.com/blogs/association-rule-mining-importance-and-steps
3. Garg, A. (2018, September 3). *Complete guide to Association Rules (1/2)*. Towards Data Science; Towards Data Science. https://towardsdatascience.com/association-rules-2-aa9a77241654
4. Grosvenor, M. (2019, July 16). *Market Basket Analysis 101: Anticipating Customer Behavior*. Smartbridge. https://smartbridge.com/market-basket-analysis-101/
5. *TransactionEncoder - mlxtend*. (n.d.). Rasbt.github.io. http://rasbt.github.io/mlxtend/user_guide/preprocessing/TransactionEncoder/

APPENDIX

```
#Important libraries
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
# Creating array with the help of transaction encoder and fit and transform method
tra = TransactionEncoder()
```

```
tra_array = tra.fit(Data_set).transform(Data_set)

tra_array

#Creating the data frame

data = pd.DataFrame(tra_array, columns= tra.columns_)

data

#applying apriori for frequent bought items

from mlxtend.frequent_patterns import apriori

bought_items= apriori(data,min_support = 0.4, use_colnames = True)

bought_items

#applying association rule

from mlxtend.frequent_patterns import association_rules

Ass_rule = association_rules(bought_items, metric="confidence", min_threshold = 0.6)

Ass_rule
```

TASK 3 – CLUSTERING

The objective of the job is to train the clustering model on the Facebook Live dataset and to ideally generate a number of clusters. The properties of the clusters will then be looked into. For this, the dataset will be imported. It will then be ready and normalised. Then, correlation will be looked at to see which columns are correlated. Thereafter, the ideal number of clusters will be established. After the data have been fitted into the model, it will then be trained. After training, cluster properties will be looked at.

The k-means technique and hierarchical clustering were both used to group the Facebook live post dataset. A Jupyter notebook was used to carry out the project. The data was initially fitted to the model after determining the ideal number of clusters. Python has demonstrated that three clusters function nicely. The number of clusters generated by both procedures was 3, and the outcomes were quite similar.

INTRODUCTION

Each day, hundreds of gigabytes of data are generated worldwide. It has changed and is now the new benchmark for exchange for the biggest tech companies on the planet. One of the most crucial things for organisations to have in order to understand their consumers' needs and how to meet them is this ability. Online content is not organised nor categorised, though. Before being used, this unlabeled data need to be removed and converted into a format that can be used. Unsupervised machine learning is one of the most important methods for dealing with these unlabeled data.

Unlabeled data is examined using a machine learning technique called clustering before being divided into groups. These clusters are just groups of data items having comparable features. While having some things in common, the data points in one cluster are different from the data points in other clusters. There are various clustering patterns. Each has special qualities and applications. Hierarchical clustering, K-means, and C-means clustering are a few examples of different clustering techniques.

DATASETS

The most well-known social networking site is Facebook, which has more than 700 million users worldwide. The initial driving force for the creation of the website was the desire to unite a small group of individuals who shared similar interests. The site was quickly made available to people across the nation and then the world, which contributed to the increasing user base of today. Although the website was initially created to connect people, it is now utilised by businesses, communities, non-profit organisations, celebrities, and even corporations to market their products and services.

This data contains the facebook live data which contains 12 columns. This data contains metrics such comments, shares, and reactions.

Column name	**Data type**
status_id	Integer
status_type	Object
status_published	Object
num_reactions	Integer
num_comments	Integer

num_shares	Integer
num_likes	Integer
num_loves	Integer
num_wows	Integer
num_hahas	Integer
num_sads	Integer
num_angrys	Integer

EXPLANATION AND PREPARATION OF DATASETS

The dataset consists of data from Facebook posts, including remarks, likes, shares, and reactions. There are 12 columns in the dataset. Several pre-processing steps were taken before training the model. Columns that aren't needed and null values are deleted throughout the pre-processing stages. After that, the data was normalised.

```
data.head()
```

	status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_ang
0	1	video	4/22/2018 6:00	529	512	262	432	92	3	1	1	
1	2	photo	4/21/2018 22:45	150	0	0	150	0	0	0	0	
2	3	video	4/21/2018 6:17	227	236	57	204	21	1	1	0	
3	4	photo	4/21/2018 2:29	111	0	0	111	0	0	0	0	
4	5	photo	4/18/2018 3:22	213	0	0	204	9	0	0	0	

Figure: above shows the head of the data.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   status_id         199 non-null    int64 
 1   status_type       199 non-null    object
 2   status_published  199 non-null    object
 3   num_reactions     199 non-null    int64 
 4   num_comments      199 non-null    int64 
 5   num_shares        199 non-null    int64 
 6   num_likes         199 non-null    int64 
 7   num_loves         199 non-null    int64 
 8   num_wows          199 non-null    int64 
 9   num_hahas         199 non-null    int64 
 10  num_sads          199 non-null    int64 
 11  num_angrys        199 non-null    int64 
dtypes: int64(10), object(2)
memory usage: 18.8+ KB
```

Figure: above shows the null value and data type of the data.

```
data.describe()
```

	status_id	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
count	199.000000	199.000000	199.000000	199.000000	199.000000	199.000000	199.000000	199.000000	199.000000	199.000000
mean	100.000000	231.020101	134.391960	21.381910	213.492462	15.422111	1.175879	0.391960	0.407035	0.130653
std	57.590508	348.551041	234.353413	61.299273	336.402099	19.031678	2.182164	0.998552	2.313804	0.485148
min	1.000000	8.000000	0.000000	0.000000	8.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	50.500000	91.000000	7.000000	0.000000	84.500000	2.000000	0.000000	0.000000	0.000000	0.000000
50%	100.000000	177.000000	15.000000	2.000000	166.000000	9.000000	0.000000	0.000000	0.000000	0.000000
75%	149.500000	273.500000	177.000000	31.500000	249.500000	23.000000	1.000000	0.000000	0.000000	0.000000
max	199.000000	4410.000000	1979.000000	753.000000	4315.000000	139.000000	17.000000	8.000000	22.000000	4.000000

Figure: above shows mean, median and standard deviation.

IMPLEMENTATION IN PYTHON:

Library Used

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

import os
```

Part One: K-means Clustering

The K-means Clustering algorithm will be the first clustering algorithm we examine. The number of clusters (k) is a user-defined parameter when using K-means clustering.

Here, we will begin by using K-means clustering on a dataset that has only two number variables. The scatter plot will be simple to comprehend and depict.

```
#Importing the dataset
data=pd.read_csv('facebook_live.csv')
```

We use the read_csv() function in the pandas library to read the data set.

```
from sklearn.preprocessing import StandardScaler
X=data.iloc[:,[4,5]].values
sc_X=StandardScaler()
X=sc_X.fit_transform(X)
```

We train the data first. To train the data, we had to utilise a standardscaler. Because the data is so large, we are simply utilising two columns to do the K-means because it will be simple for us to visualise and comprehend. The two columns on which we wish to run the K-means clustering were extracted using the iloc() tools.

```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Here we are using elbow method to determine the number of clusters we can possibly have. The elbow method involves calculating the within cluster sum of squares (wcss) for the data with various values of k.

```
# Fitting K-Means to the dataset
kmeans=KMeans(n_clusters=3,init='k-means++',random_state=42)
y_kmeans=kmeans.fit_predict(X)
```

After knowing the optimal number of clustering we use fit_predict() method to train a k-means.

```
#Visualising the clusters
plt.figure(figsize=(8,8))
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],s=100,c='blue',label='Cluster 1')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],s=100,c='red',label='Cluster 2')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],s=100,c='yellow',label='Cluster 3')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='magenta',label='Centroids')
plt.title('Clusters of Facebook')
plt.ylabel('Num of shares')
plt.xlabel('Num of comments')
plt.legend()
plt.show()
```

After training the k-means we plot the cluster with a centroid using the scatterplot, Using a colour pallet to differentiate the centroid from the clusters.

Part two: Hierarchical Clustering:

A method of distance-based grouping is hierarchical clustering. However, hierarchical clustering operates fundamentally differently from K-means clustering in that it begins by allocating all data points to a cluster and then iteratively updates the cluster centroids and cluster assignments. We may create a tree, or dendrogram, using hierarchical clustering that illustrates the point at which each cluster merges with another. Each colour on the dendrogram corresponds to a separate cluster. The dendrogram created to determine the ideal number of clusters is shown below.

```
#usig the dendrogrm to find out the optimal number of clusters
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(14,6))
dendrogram= sch.dendrogram(sch.linkage(X, method='ward'))
plt.title('Dendogram')
plt.xlabel('facebook_data')
plt.ylabel('Euclidean distance')
plt.show()
```

Above shows the code used to create dendrogram to find out the number of clusters using hierarchical clustering method.

```
#ftting hirearchial clutering to the dataset
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters= 3,affinity='euclidean',linkage= 'ward')
y_hc = hc.fit_predict(X)
```

After finding out the number of clusters in our case its 3. We try to fit the hierarchical clustering to the dataset.

```
#Visualising the clusters
plt.figure(figsize=(8,8))
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],s=100,c='blue',label='Cluster 1')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],s=100,c='red',label='Cluster 2')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],s=100,c='yellow',label='Cluster 3')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='magenta',label='Centroids')
plt.title('Clusters of Facebook')
plt.ylabel('Num of shares')
plt.xlabel('Num of comments')
plt.legend()
plt.show()
```

Above code is used to display each cluster in a different colour, we can plot a scatterplot of the resulting data with the cluster assignments and centroids are shown.

RESULTS ANALYSIS AND DISCUSSION:

```
# Using the elbow method to find th eoptimal number of clusters
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,10):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,10),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

We used the elbow method to find out the optimal number of clusters in K-means. Above is the code for the elbow method. The elbow method tells us the optimal number of clusters using a graph. We see a sudden curve in the graph and the graph where it bends is called the elbow point thus elbow method. The results are shown below. IN our case the number of optimal clusters obtain are three as we can see in the graph given below that the graph is bending at 3 thus the no. of clusters is 3.

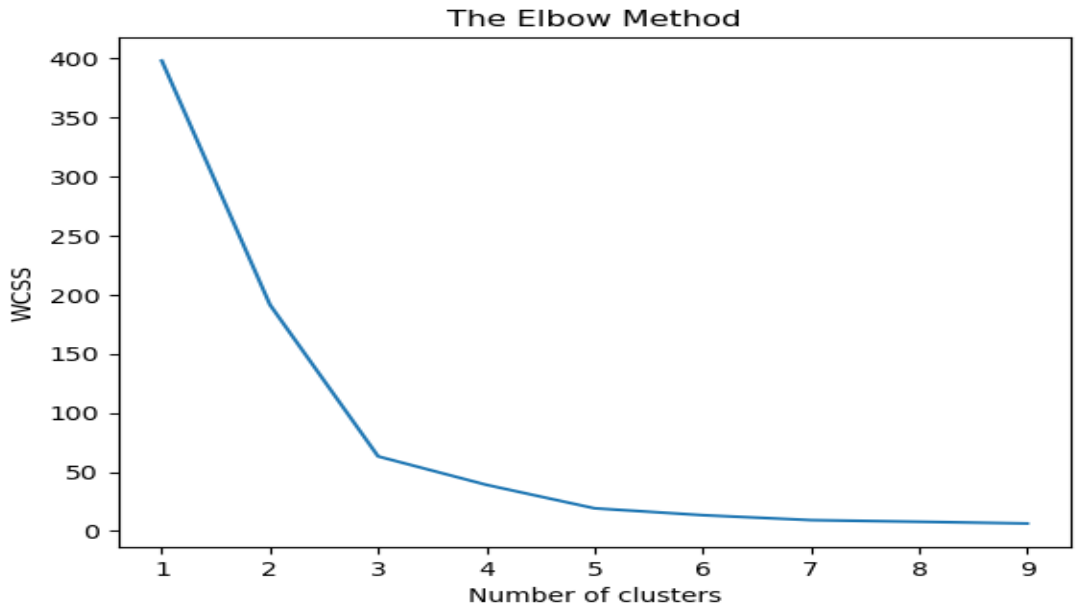


Figure: shows the no. of clusters using elbow method.

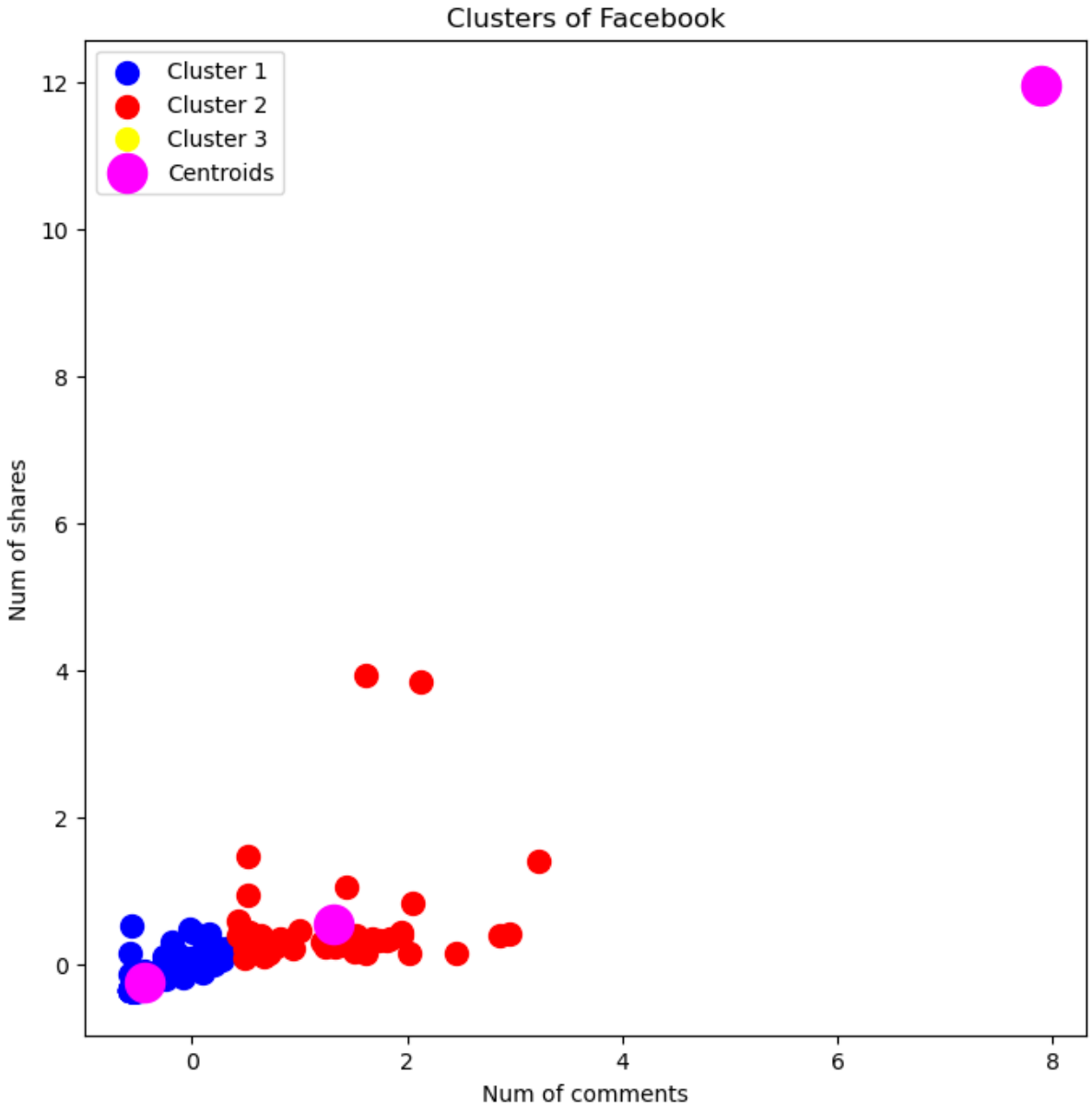


Figure: shows the clusters with centroids

```
#usig the dendrogrm to find out the optimal number of clusters
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(14,6))
dendrogram= sch.dendrogram(sch.linkage(X, method='ward'))
plt.title('Dendogram')
plt.xlabel('facebook_data')
plt.ylabel('Euclidean distance')
plt.show()
```

The code for finding out the optimal number of clusters using Hierarchical method is given above. In hierarchical method we can construct a tree or a dendrogram. Here we

are using dendrogram to find out the optimal number of clusters. Dendrogram of different colour show a different cluster. The result is given below:

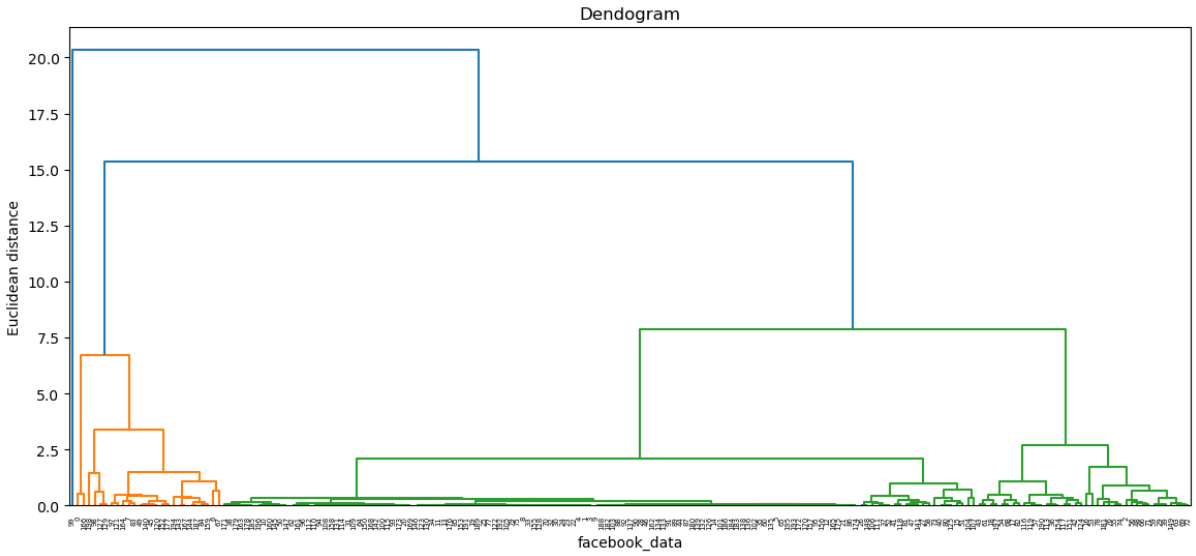


Figure: shows the dendrogram created using hierarchical clustering

Visualising the clusters using centroids, after finding out the optimal number of clusters. The results are given below.

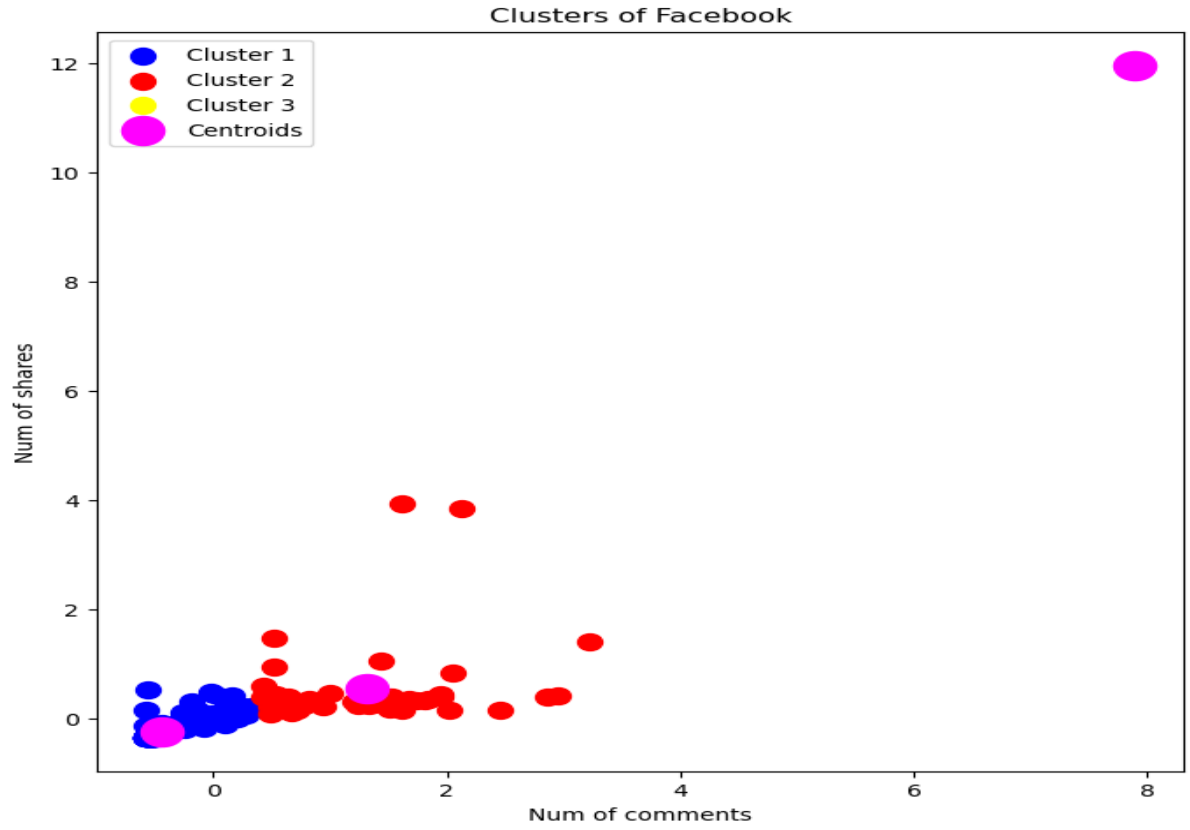


Figure: shows the clusters with centroids

DISCUSSION

Clustering in python were generated and got almost the same results, the number of clusters found using K-means and hierarchical method were same. When the optimum number of clusters were calculated, K-mmeans showed 3 whereas the hierarchical method also gave 3. So, the data was fit to the model to form 3 clusters.

CONCLUSION

As digital technology has developed, a vast amount of data is being produced. The firms depend on these data. A large majority of the generated data are unorganised and unlabeled. Unsupervised learning is crucial for comprehending these data. Data are grouped into groups using one of the unsupervised learning techniques called clustering. The same cluster is formed by the similar data points, whose features are similar to those of the other data points in the cluster but distinct from those of the data points in other clusters. We intended to cluster the dataset including information about Facebook Live posts for this job. Python's K-means method and hierarchical algorithm were employed. K-means determined that three clusters were the ideal number, and Hierarchical also determined that this was the case. The hierarchical conclusion was more precise and tended to align with the dataset's original classification, which was also three.

REFERENCES:

1. *python - K Means Clustering: What does it mean about my input features if the Elbow Method gives me a straight line?* (n.d.). Stack Overflow. https://stackoverflow.com/questions/61820103/k-means-clustering-what-does-it-mean-about-my-input-features-if-the-elbow-method
2. Kaushik, S. (2019, March 11). *An Introduction to Clustering & different methods of clustering*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/
3. *k means - Is it important to scale data before clustering?* (n.d.). Cross Validated. Retrieved December 20, 2022, from https://stats.stackexchange.com/questions/89809/is-it-important-to-scale-data-before-clustering

4. *Elbow Method for optimal value of k in KMeans*. (2019, June 6). GeeksforGeeks. https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/
5. *r - Why Elbow algorithm plot shows a straight line instead of curve line?* (n.d.). Cross Validated. Retrieved December 20, 2022, from https://stats.stackexchange.com/questions/275402/why-elbow-algorithm-plot-shows-a-straight-line-instead-of-curve-line

APPENDIX

```
#Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns


# Run before importing Kmeans
import os
os.environ["OMP_NUM_THREADS"]='1'
#Importing the dataset
data=pd.read_csv('facebook_live.csv')
#Training the data
from sklearn.preprocessing import StandardScaler
X=data.iloc[:,[4,5]].values
sc_X=StandardScaler()
X=sc_X.fit_transform(X)
# Using the elbow method to find th eoptimal number of clusters
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,10):
```

```
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,10),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
# Fitting K-Means to the dataset
kmeans=KMeans(n_clusters=3,init='k-means++',random_state=42)
y_kmeans=kmeans.fit_predict(X)
#Visualising the clusters
plt.figure(figsize=(8,8))
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],s=100,c='blue',label='Cluster 1')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],s=100,c='red',label='Cluster 2')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],s=100,c='yellow',label='Cluster 3')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='magenta',la
bel='Centroids')
plt.title('Clusters of Facebook')
plt.ylabel('Num of shares')
plt.xlabel('Num of comments')
plt.legend()
plt.show()
```

TASK 4 – TEXT MINING AND SENTIMENT ANALYSIS

The task's goal is to perform text mining and sentiment analysis on reviews of 30 hotels that were chosen at random from the dataset's 53645 hotel records. Data from thirty hotels were taken for this project from the main dataset. The data were pre-processed and manipulated in accordance with the required format. After this was finished, word clouds were made, and then text mining and sentiment analysis were carried out to ascertain the reviews' sentiment.

INTRODUCTION

Using natural language processing, text mining is an AI-based technique that turns unlabeled text data into labelled or structured formats that are suitable for study. It is crucial to transform them into a format that will enable understanding of the data, analysis, or use of the data to train a machine learning model as the volume of unlabelled text data generated globally increases. For this, text mining is useful. Different sectors employ various features of text mining. Understanding the language of the text and extrapolating the feelings it conveys is one of the most crucial elements. It is primarily employed in sectors like marketing or hospitality where client feedback is crucial.

Sentiment analysis, often known as opinion mining, is the examination of the attitudes or feelings concealed in the text. It entails computationally processing the language to draw out the emotions of the speaker or writer who created the data. It is a component of natural language processing that enables the computer to comprehend text that has been written by humans in the same way that a person would have. In summary, sentiment analysis helps identify whether a text has a good or negative attitude.

There are various types of sentiment analysis. It involves multilingual sentiment analysis, aspect-based sentiment analysis, fine grain sentiment analysis, and emotion identification. The polarity of the text can be determined using fine-grained sentiment analysis. It indicates if the text is neutral, favourable, extremely favourable, unfavourable, or unfavourable. The text's author's mood or emotion can be ascertained through emotion detection. It involves feelings like happiness, sadness, anger, etc. Similar to the fine grain method, aspect-based sentiment analysis examines customer evaluations to determine the context in which they were written. This aids in the company's comprehension of the comments.

DATASETS

The institution provided the dataset for this challenge. The collection included information for 53645 unique hotels and restaurants. The ID, Review Date, Location, Hotel.name, and review are among its five qualities. The unique ID assigned to each review is contained in the ID attribute. The date for a specific review can be found in the Review Date column. The address of the hotel or restaurant where the user made the review is listed in the Location column. The user-written reviews are displayed in the review column. 53,644 distinct reviews in total are included in the dataset.

EXPLANATION AND PREPARATION OF DATASETS

Data Pre-processing

53,644 hotel reviews from 537 distinct properties were included in the dataset, which required pre-processing in order to prepare it for the assignment. First, the 30 hotels needed for the analysis had to be extracted. The list of distinctive hotel names was culled for this, and 30 hotels were chosen. The data for these 30 hotels was then extracted using a filter.

Some steps involved in data pre-processing are shown below:

```
#Loading the dataset
data = pd.read_csv('Tourist_accommodation_reviews.csv')
```

Loading the data

```
data.head()
```

	ID	Review Date	Location	Hotel/Restaurant name	Review
0	rn579778340	Reviewed 1 week ago	Kathu	Thong Dee The Kathu Brasserie	Just been for sunday roast lamb and beef truly...
1	rn576350875	Reviewed 3 weeks ago	Kathu	Thong Dee The Kathu Brasserie	Quietly set off the main road, nice atmosphere...
2	rn574921678	Reviewed 4 weeks ago	Kathu	Thong Dee The Kathu Brasserie	I made a reservation for a birthday two days i...
3	rn572905503	Reviewed April 12, 2018	Kathu	Thong Dee The Kathu Brasserie	We visit here regularly and never fail to be i...
4	rn572364712	Reviewed April 10, 2018	Kathu	Thong Dee The Kathu Brasserie	Visited this wonderful place on my travels and...

Hotels were categorized on the basis of location.

```
data['Location'].unique()
```

```
array([' Kathu', ' Kata Beach', ' Rawai', ' Choeng Thale', ' Karon Beach',
       ' Phuket Town', ' Patong', ' Mai Khao', ' Karon', ' Chalong',
       ' Nai Harn', ' Cape Panwa', ' Sakhu', ' Pa Khlok', ' Kamala',
       ' Bang Tao Beach', ' Thalang District', ' Talat Nuea',
       ' Kata Noi Beach', ' Wichit', ' Nai Yang', ' Talat Yai',
       ' Koh Kaew', ' Nai Thon', ' Ratsada'], dtype=object)
```

TASK: TEXT MINING AND SENTIMENT ANALYSIS

IMPPLEMENTAION IN PYTHON

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import seaborn as sns
import nltk
```

Above given are some of the libraries used in text mining task

```
nltk.download(['stopwords','punkt','wordnet','omw-1.4','vader_lexicon'])
get_ipython().run_line_magic('matplotlib', 'inline')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\catch\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\catch\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\catch\AppData\Roaming\nltk_data...
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\catch\AppData\Roaming\nltk_data...
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\catch\AppData\Roaming\nltk_data...
```

From nltk download stop word s which are used in tokenization for doing the text mining process.

```
st_words = nltk.corpus.stopwords.words('english')
print(st_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

The first step is tokenization. We have used the RegexpTokenizer from NLTK to do this. This converts our character string into tokens by splitting it into words. In addition, this allows us to define a regular expression, and we can therefore define a regular expression so that we only tokenize alphanumeric characters, this removing punctuation.

Now we will use the sentimentitensityanalyzer to do the sentiment analysis.

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sentiment = SentimentIntensityAnalyzer()
print(sentiment.polarity_scores('The hotel is good'))
print(sentiment.polarity_scores('The hotel is bad'))
```

We have already instantiated the SentimentIntensityAnalyzer above. We can therefore use this to generate polarity scores for the reviews in our dataset. To do this, we use list comprehension to create a new column for each from the dictionaries returned when we use the polarity_scores method.

```
Hotels['neg']= Hotels['Review'].apply(lambda x:sentiment.polarity_scores(x)['neg'])
Hotels['neu']= Hotels['Review'].apply(lambda x:sentiment.polarity_scores(x)['neu'])
Hotels['pos']= Hotels['Review'].apply(lambda x:sentiment.polarity_scores(x)['pos'])
Hotels['compound']= Hotels['Review'].apply(lambda x:sentiment.polarity_scores(x)['compound'])
```

We can also use the describe () method to get more of an insight into sentiment scores for the review data.

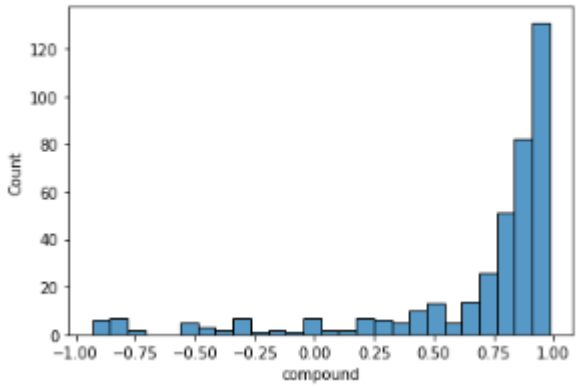
```
Hotels[['compound','neg','neu','pos']].describe()
```

	compound	neg	neu	pos
count	397.000000	397.000000	397.000000	397.000000
mean	0.666339	0.028514	0.731945	0.239549
std	0.451535	0.060700	0.123872	0.133279
min	-0.932100	0.000000	0.331000	0.000000
25%	0.648400	0.000000	0.647000	0.140000
50%	0.855500	0.000000	0.743000	0.234000
75%	0.927200	0.038000	0.821000	0.331000
max	0.985400	0.365000	1.000000	0.669000

Distribution of the compound score.

```
sns.histplot(Hotels['compound'])
```

```
<AxesSubplot:xlabel='compound', ylabel='Count'>
```




```
neg_tokens = [word for review in n_reviews['processed_review'] for word in review]
wordcloud= WordCloud(background_color='green').generate_from_text(' '.join(neg_tokens))
plt.figure(figsize=(12,12))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Above is the codes used for generating word cloud

RESULTS ANALYSIS

```
st_words = nltk.corpus.stopwords.words('english')
print(st_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

With the help of stop words we separated the review and all punctuation marks and stopwords were removed. Stop words are predefined in python library.

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sentiment = SentimentIntensityAnalyzer()
print(sentiment.polarity_scores('The hotel is good'))
print(sentiment.polarity_scores('The hotel is bad'))
```

```
{'neg': 0.0, 'neu': 0.508, 'pos': 0.492, 'compound': 0.4404}
{'neg': 0.538, 'neu': 0.462, 'pos': 0.0, 'compound': -0.5423}
```

Using sentimentintensityanalyzer we get the neg, new and compound value.

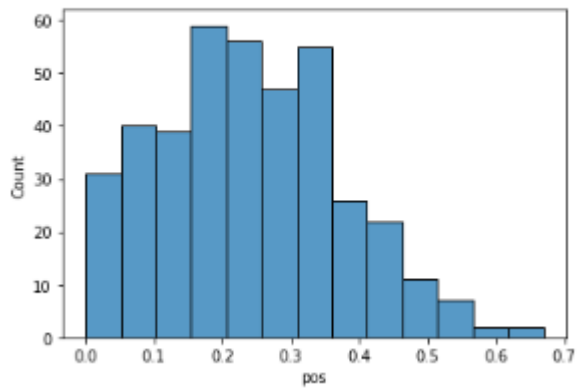
```
Hotels[['compound','neg','neu','pos']].describe()
```

	compound	neg	neu	pos
count	397.000000	397.000000	397.000000	397.000000
mean	0.666339	0.028514	0.731945	0.239549
std	0.451535	0.060700	0.123872	0.133279
min	-0.932100	0.000000	0.331000	0.000000
25%	0.646400	0.000000	0.647000	0.140000
50%	0.855500	0.000000	0.743000	0.234000
75%	0.927200	0.038000	0.821000	0.331000
max	0.985400	0.365000	1.000000	0.669000

Using the describe function we can see that mean of compound is 0.6, neg is 0.02, neu is 0.7 and pos is 0.2.

```
sns.histplot(Hotels['pos'])
```

```
<AxesSubplot:xlabel='pos', ylabel='Count'>
```



We can see positive as well as negative value using histogram plot.

```
(Hotels['compound']<=0).groupby(Hotels['Hotel/Restaurant name']).sum()
```

```
Hotel/Restaurant name
EAT. bar & grill                      4
MaMa Jin Restaurant                   6
On The Rocks - Marina Phuket Resort   9
Veranda                              22
Name: compound, dtype: int64
```

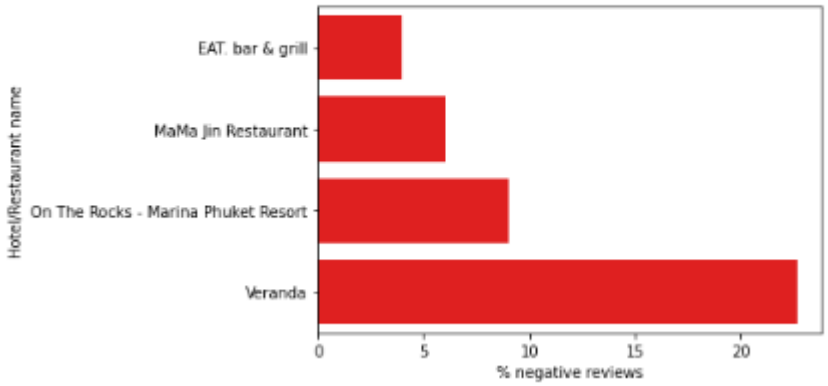
Grouping by Hotel/restaurant name with the help of groupby method whose compound value is less than or equal to zero.

```
negative = pd.DataFrame((Hotels['compound']<=0).groupby(Hotels['Hotel/Restaurant name']).sum()
                        /Hotels['Hotel/Restaurant name'].groupby(Hotels['Hotel/Restaurant name']).count()*100,
                        columns=['% negative reviews']).sort_values(by='% negative reviews')
negative
```

Hotel/Restaurant name	% negative reviews
EAT. bar & grill	4.000000
MaMa Jin Restaurant	6.000000
On The Rocks - Marina Phuket Resort	9.000000
Veranda	22.680412

```
sns.barplot(data=negative, x='% negative reviews', y=negative.index, color='red')
```

```
<AxesSubplot:xlabel='% negative reviews', ylabel='Hotel/Restaurant name'>
```



Bar plot is used to show the negative review of the hotels.

```
Hotels['processed_review'] = Hotels['Review'].apply(preprocess_text)
p_reviews = Hotels.loc[(Hotels['Hotel/Restaurant name'] == 'Veranda') & (Hotels['compound']>0),:]
n_reviews = Hotels.loc[(Hotels['Hotel/Restaurant name'] == 'Veranda' ) & (Hotels['compound']<=0),:]
n_reviews.head()
```

	ID	Review Date	Location	Hotel/Restaurant name	Review	neg	neu	pos	compound	processed_review
27507	rn354063931	Reviewed March 9, 2016	Karon Beach	Veranda	Arrived to be given menus in Russian, not Engl...	0.073	0.927	0.000	-0.5157	[arriv, given, menu, russian, english, without...
27499	rn393362294	Reviewed July 16, 2016	Karon Beach	Veranda	Not sure how this place has managed to get so ...	0.213	0.694	0.093	-0.7638	[sure, place, manag, get, mani, good, review, ...
27488	rn449682768	Reviewed January 5, 2017	Karon Beach	Veranda	Walked in on a moderately busy night. Was not ...	0.208	0.792	0.000	-0.8158	[walk, moder, busi, night, serv, 40, minut, en...
27509	rn346993963	Reviewed February 11, 2016	Karon Beach	Veranda	This is an odd restaurant... you walk up some ...	0.121	0.879	0.000	-0.5423	[odd, restaur, walk, reason, high, step, front...
27516	rn335016228	Reviewed December 28, 2015	Karon Beach	Veranda	This is a restaurant owned/managed by Russians...	0.052	0.948	0.000	-0.3071	[restaur, own, manag, russian, direct, toward,...

Both the figure up and down shows the word cloud for best and worst Restaurant

DISCUSSION

The outcomes of the Python implementation are listed above. The outcomes will be discussed in light of the actual execution. Python was used to determine the top words mentioned in reviews of all 30 hotels. The most often used words were shown to be "Food" and "Place". This makes sense considering that most of the evaluations are for eateries where folks go to enjoy some delectable food. Therefore, it is likely that they have used this word anytime they have submitted a review.

CONCLUSION

Text mining is a popular method for transforming text material into a format that computers can utilise and process. Natural language processing includes sentiment analysis, which employs text data to determine the text's emotional tone. It is frequently used in the hotel industry to comprehend user reviews. It is feasible to accomplish this manually, but if there are a lot of reviews, it becomes impossible. The sentiment score of the reviews is generated using sentiment analysis, which then determines if the review is positive or negative. If the score is greater than 0, the review is considered good; if it is lower than 0, the review is considered unfavourable.

REFERENCES:

1. *pandas.core.groupby.DataFrameGroupBy.nunique — pandas 1.5.2 documentation*. (n.d.). Pandas.pydata.org. Retrieved December 20, 2022, from https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.core.groupby.DataFrameGroupBy.nunique.html
2. Arnal, J. R. (2020, July 29). *Introduction to NLP: Sentiment analysis and Wordclouds ⋆ Quantdare*. Quantdare. https://quantdare.com/introduction-to-nlp-sentiment-analysis-and-wordclouds/
3. Acharya, S. (2022, November 23). *Text Analytics 101 — Word Cloud and Sentiment Analysis*. Medium. https://towardsdatascience.com/text-analytics-101-word-cloud-and-sentiment-analysis-2c3ade81c7e8
4. Ganesan, K. (2019, April 6). *What are Stop Words?* Opinosis Analytics. https://www.opinosis-analytics.com/knowledge-base/stop-words-explained/#:~:text=Stop%20words%20are%20a%20set

APPENDIX:

```
# Importing the libraries


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import seaborn as sns
import nltk
#Loading the dataset
data = pd.read_csv('Tourist_accommodation_reviews.csv')
#Grouping the data on the basis of location
data = data_store.groupby(['Location']).Review.nunique()
data_store
#Grouping the data based on hotel or restaurant name
Location_order= place.groupby(['Hotel/Restaurant name']).Review.nunique()
Location_order
#Using stop word to tokenize the data
st_words = nltk.corpus.stopwords.words('english')
print(st_words)
#Codes used for generating the word cloud
neg_tokens = [word for review in n_reviews['processed_review'] for word in review]
wordcloud=WordCloud(background_color='green').generate_from_text(''.join(neg_tok
ens))
plt.figure(figsize=(12,12))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```