INDIVIDUAL REPORT

INTRODUCTION

A UK-based online retail store has captured the sales data for different products for the period of one year (Nov 2016 to Dec 2017) and updated the dataset (in 2020). The organization sells gifts primarily on the online platform. The customers who make a purchase, consume directly for themselves. There are small businesses that buy in bulk and sell to other customers through the retail outlet channel. The project objective is to find significant customers for the business who make high purchases of their favorite products. The organization wants to roll out a loyalty program to high-value customers after the identification of segments. The Data mining algorithm used is K-means clustering and it is in standard form (see Appendix A for in depth knowledge). The packages used to implement the network are NumPy, Pandas, Seaborn, Matplotlib, Sklearn and Scipy. For the K-means clustering method, the most common approach for metric evaluation is the so-called elbow method. It involves running the algorithm multiple times over a loop, with an increasing number of cluster choices, and then plotting a clustering score as a function of the number of clusters. My Work for the project is performing the Exploratory Data Analysis (EDA).

ALGORITHM DEVELOPMENT

K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean dis

Given a set of observations (x_1, x_2, ..., x_n), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k ($\leq$ n) sets S = {S_1, S_2, ..., S_k} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where μ_i is the mean of points in S_i.

The equivalence can be deduced from identity

$$\sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} (\mathbf{x} - \boldsymbol{\mu}_i)^T (\boldsymbol{\mu}_i - \mathbf{y})$$

WORKING :

To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids
It halts creating and optimizing clusters when either:

- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

My portion of work was plotting the Monthly Expenses, Calculating the Expenses, Invoices for each Customer ID and Unique Items purchased for each Customer.

```
print("PLotting monthly TotalExpense distribution")
plt.figure(figsize=(12, 8))
sns.barplot('Month', 'TotalExpense', data=df, hue='Year')
plt.title("Monthly TotalExpense distribution")
plt.show()
```

Here, I've plotted a Bar chart using Seaborn. The title of the chart is "Monthly Total Expense Distribution" with x-axis as Month and y-axis as Total Expense respectively.

```
print("Calculating TotalExpense for each CustomerID")
customer_stat =
df.groupby(['CustomerID'])['TotalExpense'].agg([np.sum, np.mean,
np.max, np.min])
df1 = pd.DataFrame(customer_stat)
df1.columns = ['Total Expenditure', 'MeanAmt', 'MaxAmt', 'MinAmt']
print(df1.head(5))
```

Here, I've calculated the Total Expense for each Customer ID. I implemented it using groupby by grouping the Customer ID and Total Expense with columns as Total Expenditure, Mean Amount, Maximum Amount and Minimum Amount.

```
print("Calculating number of invoices for each CustomerID")
transactions = df[['InvoiceNo', 'CustomerID']]
transactions = transactions.drop_duplicates()
transactions = transactions.groupby(by='CustomerID',
as_index=False).count()
transactions = transactions.rename(columns={'InvoiceNo':
'Total Purchases'})
print(transactions.head(5))
```

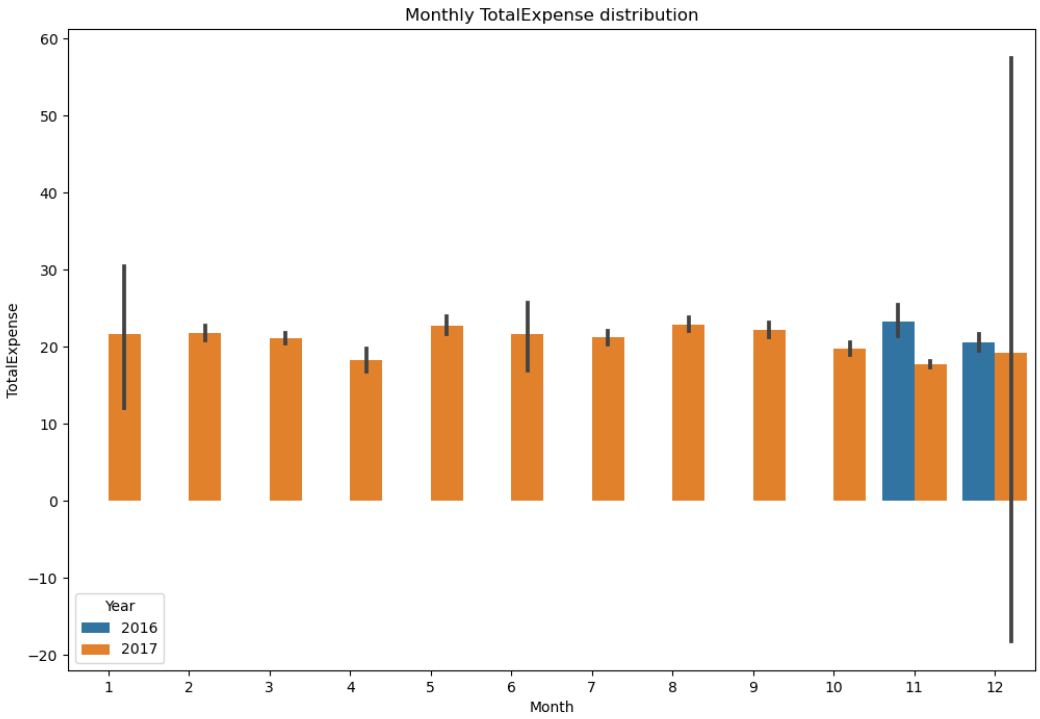
Here, I've calculated the number of invoices for each Customer ID. I implemented it using the Invoice Number and Customer ID. Also, I renamed the Invoice Number to Total Purchases for a better view of the result.

```
print("Calculating number of unique items purchased by each customer")
unique_items = df[['StockCode', 'CustomerID']]
unique_items = unique_items.drop_duplicates()
unique_items = unique_items.groupby(by='CustomerID', as_index=False).count()
unique_items = unique_items.rename(columns={'StockCode': 'No. of unique items'})
print(unique_items.head(5))
```

Here, I've calculated the number of unique items purchased by each customer. I implemented it using the Stock code and Customer ID. Also, I renamed the Stock Code to Number of Unique Items for a better view of the result.

RESULTS :

- Plotting monthly Total Expense Distribution :



From the above graph we can determine the Monthly Total Expenses for the Year 2016 and 2017. The Vertical Bar Charts are distinguished from histograms. We can determine that the month 5 of the year 2017 carries highest expense and for the month 11 of the year 2016 carries highest expense.

- **Calculating Total Expense for each Customer ID:**

```
Calculating TotalExpense for each CustomerID
            Total Expenditure    MeanAmt   MaxAmt    MinAmt
CustomerID
12346                    0.00   0.000000  77183.6 -77183.60
12347                 4310.00  23.681319    249.6      5.04
12348                 1797.24  57.975484    240.0     13.20
12349                 1757.55  24.076027    300.0      6.64
12350                  334.40  19.670588     40.0      8.50
# ------------------------------------------------
```

From the above table we can determine the first few rows of the Total Expenditure, Mean Amount, Maximum Amount and Minimum Amount of each Customer ID.

- **Calculating Number of Invoices for each Customer ID:**

```
 Calculating number of invoices for each CustomerID
    CustomerID  Total Purchases
 0       12346                2
 1       12347                7
 2       12348                4
 3       12349                1
 4       12350                1
 # ------------------------------------------------
```

From the above table we can determine the first few rows of the Total Purchases of each Customer ID. We can say that the Customer ID 12347 has 7 purchases. Which carries the highest.

- **Calculating Number of Unique Items Purchased by Each Customer:**

```
Calculating number of unique items purchased by each customer
   CustomerID  No. of unique items
0       12346                   1
1       12347                 103
2       12348                  22
3       12349                  73
4       12350                  17
# ------------------------------------------------
```

From the above table we can determine the first few rows of the Number of Unique Items purchased by each Customer. We can say that the Customer ID 12347 has 103 unique items. Which carries the highest.

```
import sys
import matplotlib
matplotlib.use('Qt5Agg')
from PyQt5 import QtCore, QtWidgets
from matplotlib.backends.backend_qt5agg import
FigureCanvasQTAgg
from matplotlib.figure import Figure
class MplCanvas(FigureCanvasQTAgg):
   def __init__(self, parent=None, width=5, height=4,
dpi=100):
       fig = Figure(figsize=(width, height), dpi=dpi)
       self.axes = fig.add_subplot(111)
       super(MplCanvas, self).__init__(fig)
class MainWindow(QtWidgets.QMainWindow):
   def __init__(self, *args, **kwargs):
       super(MainWindow, self).__init__(*args, **kwargs)
       # Create the maptlotlib FigureCanvas object,
       # which defines a single set of axes as self.axes.
       sc = MplCanvas(self, width=5, height=4, dpi=100)
       sc.axes.plot([0,1,2,3,4], [10,1,20,3,40]
  self.setCentralWidget(sc)
       self.show()
app = QtWidgets.QApplication(sys.argv)
w = MainWindow()
app.exec_()
```

Even though I haven't contributed GUI for the project I have gone through the GUI and tried executing it.

SUMMARY

K-means clustering is one of the most popular clustering algorithms and usually the first thing practitioners apply when solving clustering tasks to get an idea of the structure of the dataset. The goal of k-means is to group data points into distinct non-overlapping subgroups. Here, we are talking about performing the EDA. Plotting was done for the monthly total expense distribution for the Year 2016 and 2017. From, the chart it can be determined that the month 5 of the Year 2017 carries Highest Expense and Year 2016 carries Highest Expense. The calculation of the Total Expense, Number of Invoices and Number of Unique Items Purchased for the first few rows determines that the Customer ID 12347 carries the highest purchases and unique items purchased.

PERCENTAGE CALCULATION:

I have used 43 lines of code from the internet and then I modified 10 of lines and added another 20 lines of my own code, The percentage of the code that I've found (or) copied from the internet is 52.3%.

References

https://www.kaggle.com/vik2012kvs/high-value-customers-identification

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/