Group 6

MT413 Data Mining & Predictive Analytics

The Use of Unsupervised Learning in Customer Segmentation

Jeff Bowers – 19310243

Oluwadamilola Daramola – 20351291

Daniel Heather – 20428824

Lauren McConnell – 20332536

Cian Murray-Doyle – 20501873

Contents

Introduction

Today’s businesses have access to more data about customers than ever before (Plangger et al., 2023). Businesses need to use this data to increase the value provided for their end users and to generate more profit (Florez-Lopez, 2016). One sector that we have identified that could benefit from this availability of data is the marketing sector. With today’s consumers preferring and expecting to receive a personalized advertising experience, marketing teams are tasked with finding ways to use the data available to meet these advertising expectations (O. Pappas, 2018). An effective way to provide customers with this customized experience is to create customer profiles and group users within these profiles and provide them with advertisements accordingly.

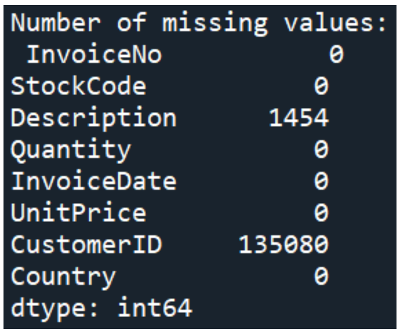
Companies use customer segmentation to effectively use their resources to focus marketing on loyal customers (Kotler P et al, 2014). This involves grouping customers into groups based on shared characteristics, for example, location, behaviours, or demographics (Alsayat, 2023). This topic has been widely researched with many studies conducted on customer segmentation in different industries. Elrod, al. (2015) conducted a study where he focused on demographic variables such as income, age, and education to segment customers into clusters. (Huseynov et al, 2017) conducted a study on their e-commerce customers. Unlike Elrod, they instead used behaviour to segment customers into clusters. They did this segmentation by observing their customers' spending on the site along with factors such as usage. They then used this information to develop strategies to improve their marketing to each customer segment. This study is similar to what we hope to demonstrate in this assignment.

We used unsupervised machine learning algorithms to create these segments. This is far more effective than the use of supervised learning. If a business were to use supervised learning to make these groups, they could use methods such as decision trees. However, it is difficult for an individual to create these groupings. It is both time-consuming and open to bias, the groupings of customers are up to the interpretation of the individual creating them. Using machine learning algorithms cuts out this human bias and makes the entire process more efficient. Considering the above use case, we believe that the Recency, Frequency and Monetary (RFM) model and K-means clustering algorithm would be most appropriate. The K-means algorithm is the most commonly used clustering algorithm, which is used to minimize the sum of squared errors (SEE) (Alsayat, 2023). Variables such as consumers' previous purchases, time of purchase, age, location and other consumer attributes can all be used to cluster consumers by similarity. However, we will use these methods to group customers based on similarities in their behaviour, their most recent purchase, how often they might purchase products on the website and how much they spend on average on these purchases.

Process of using K-Means Clustering Algorithm in Python

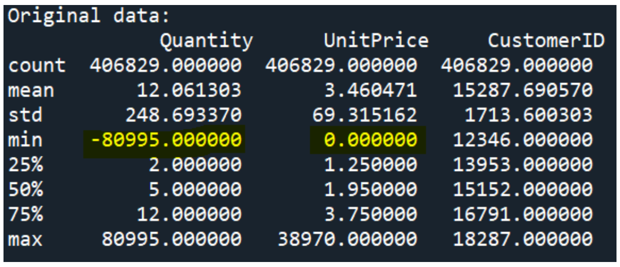
Data Cleaning:

We will now begin the process of segmenting our data. For the purpose of demonstration, this includes the Customer ID and relevant data such as the quantity of goods bought or the country in which the purchase was made. To prepare and analyse this data we used a python IDE. This is how we will be demonstrating our data preparation, RFM analysis, and K-means clustering algorithm. All coding demonstrations will be performed on the dataset provided with the submission and all relevant coding can be viewed in the appendix of this document.



```
Number of missing values:
 InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

Before any algorithm can be applied the data being used must be prepared. Once the data has been imported into Python and transformed into a data frame, we can begin our preparation process. The first step in cleaning the data is removing any missing values. Some examples of dealing with missing values are; removing them completely, replacing them with the mean/median or inferring the missing values. Due to the k-means clustering's sensitivity to missing values, our example will opt to remove all missing values from the dataset. (See Appendix A)



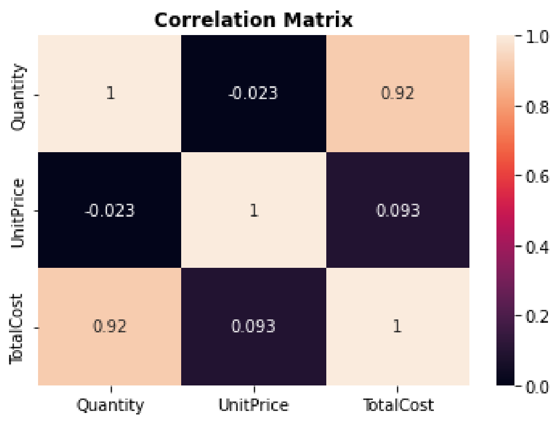
```
Original data:
             Quantity      UnitPrice     CustomerID
count  406829.000000  406829.000000  406829.000000
mean       12.061303       3.460471   15287.690570
std       248.693370      69.315162    1713.600303
min    -80995.000000       0.000000   12346.000000
25%         2.000000       1.250000   13953.000000
50%         5.000000       1.950000   15152.000000
75%        12.000000       3.750000   16791.000000
max     80995.000000   38970.000000   18287.000000
```

Next, outliers will need to be removed. To detect outliers first we will describe our data in Python. In our example, we can see there are negative values for "Quantity" and values of 0 for "UnitPrice", as we know these are not possible values and therefore need to be removed. (See Appendix B)

After these impossible values have been removed, we can use methods like z-score and the interquartile range. The Z-score represents the number of standard deviations the value is from the mean, outliers are defined as values that are further than 3 standard deviations from the mean. For the interquartile range, outliers will be values greater than the calculated upper bound and values lower than the calculated lower bound. In our dataset we can see that the outliers are difficult to determine without further context, we will need to create new features to more accurately remove outliers. (See Appendix C)

Once the outliers are removed, we will now look to remove duplicate entries. For example, the same purchase could be recorded multiple times, this would lead to errors in the clustering accuracy. We'll find the feature we need the duplicates removed from and use a line of code to remove the duplicated entries. Our example data doesn't require any duplicates to be removed but we have provided a sample code to demonstrate how this would be achieved. (See Appendix D)

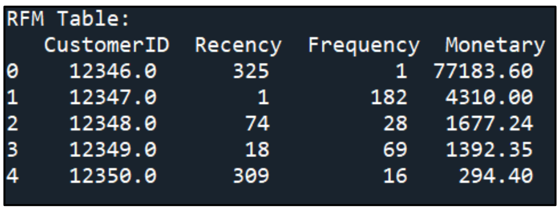
After duplicates have been removed, we will look to remove some of the irrelevant features from the dataset. By creating a correlation matrix and plotting it with a heatmap, we can see which features are double-counting data. Not including these features in the model will help avoid misrepresentation of the data. The correlation matrix can only be numeric so categorical features must be omitted. (See Appendix E)



The final step in cleaning the data will be formatting. Ensuring all data is in the correct format to be analysed. Some examples of correcting formatting can be seen in our example, changing "InvoiceDate" to be the correct date time data type and converting "CustomerID" from an integer to a string as we know this is not intended to be analysed as a numeric value. (See Appendix F)

Feature Engineering:

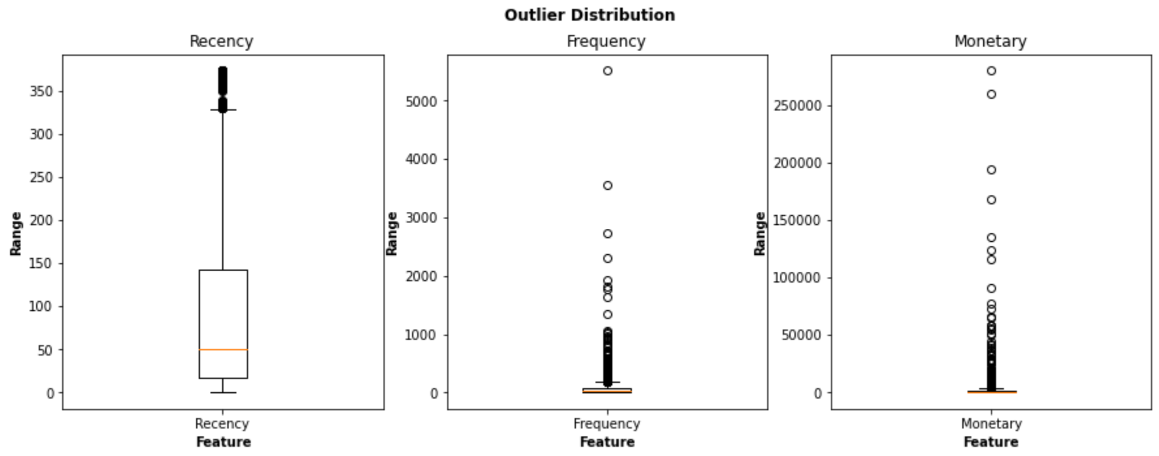
When training a machine learning model not only does the data need to be clean it also needs to be relevant. Once the data has been cleaned, we move onto feature engineering, this refers to the process of creating new features from existing features, transforming existing features and/or selecting which features are relevant.

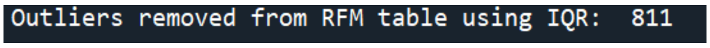


```
RFM Table:
   CustomerID  Recency  Frequency  Monetary
0     12346.0      325          1  77183.60
1     12347.0        1        182   4310.00
2     12348.0       74         28   1677.24
3     12349.0       18         69   1392.35
4     12350.0      309         16    294.40
```

In our example we will begin by using the available features to create more relevant features, the first of which is the total cost of each transaction (unit cost * quantity), we will use Recency, Frequency and Monetary (RFM) analysis. RFM analysis is a quantitative model often used to assist in customer segmentation and analysing of profit. It is used in many industries such as telecommunications, banking, insurance and retail (Mahfuza, et. al, 2022).

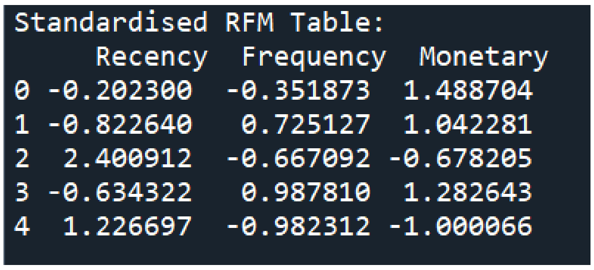
To apply this RFM analysis to our dataset, we will need to create features measuring the time since the last transaction for each customer ID (Recency), total transactions for each customer ID (Frequency) and total spent for each customer ID (Monetary). To improve the accuracy of the RFM analysis we need to remove outliers once. First, we will visualise the outliers with boxplots and then use the interquartile range method to remove outliers. (See Appendix G & Appendix H)





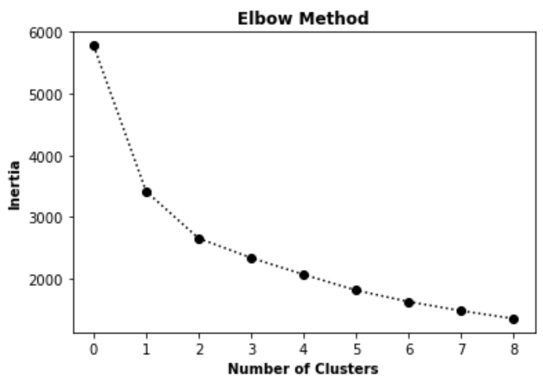
```
Outliers removed from RFM table using IQR:   811
```

Another data preparation process used during this stage is dealing with categorical variables. Machine learning algorithms need categorical values transformed into numeric values, one common way of doing this is through one-hot encoding, this can be done through the Pandas library in Python. One-hot encoding this way will add a column for each categorical variable and state in binary whether it is true or false. The method of analysis we have opted for does not require this to be done but an example code can be found in the appendix**.** The final step in preparing data for a machine learning algorithm would be to standardize/bring data to a common scale to improve the performance of the machine learning model. There are a few ways to achieve this, the method we will demonstrate is using the z-score. (See Appendix I & Appendix J)



```
Standardised RFM Table:
     Recency  Frequency  Monetary
0 -0.202300  -0.351873  1.488704
1 -0.822640   0.725127  1.042281
2  2.400912  -0.667092 -0.678205
3 -0.634322   0.987810  1.282643
4  1.226697  -0.982312 -1.000066
```

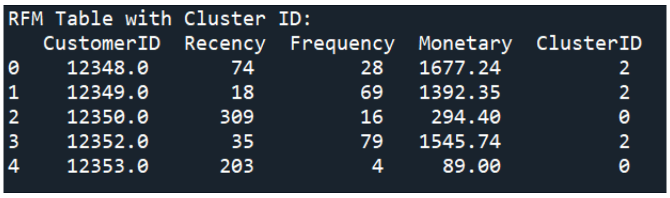
Definition of Model Chosen (K-Means Clustering):



Now that the data has been cleaned and prepared the clustering algorithm can be implemented (Vara et al., 2022). The algorithm will use customer behavioural data to group customers based on similarities. First, the optimal number of clusters needs to be found, the method we recommend for this is the elbow method.

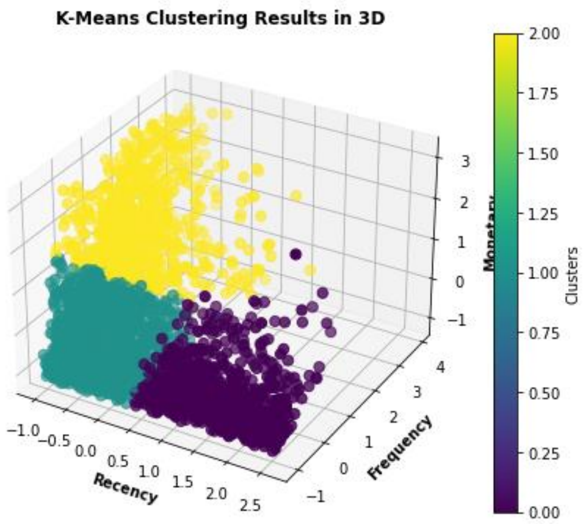
The elbow method is a graphical method of finding the optimal number of clusters, it works by taking the sum of the squared distance between centroids and each point and plotting them against k from 1-10, the optimal number of clusters is where the graph forms an "elbow". The algorithm goes through two steps:

1. Initial centroids for the clusters are chosen and then each point is assigned to the nearest centroid based on the squared Euclidean distance**.**
2. The centroids of each cluster are then updated, this is done by calculating the mean of all points assigned to the centroids cluster.



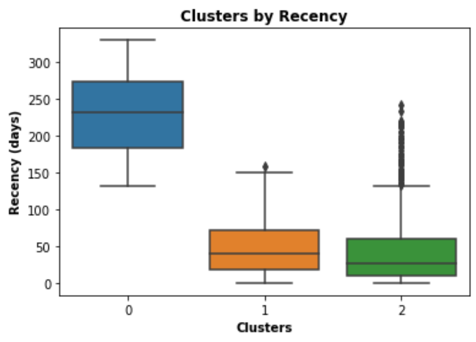
```
RFM Table with Cluster ID:
   CustomerID  Recency  Frequency  Monetary  ClusterID
0     12348.0       74         28   1677.24          2
1     12349.0       18         69   1392.35          2
2     12350.0      309         16    294.40          0
3     12352.0       35         79   1545.74          2
4     12353.0      203          4     89.00          0
```

These two steps repeat until data points no longer change cluster, the sum of the distance is minimised or the maximum number of iterations has been reached. For our dataset, 3 clusters will be optimal. Using the optimal number of clusters we can fit the k-means model to our standardised data set. This will group each customer in one of the 3 clusters, we take these cluster labels and add them to the RFM tables to begin visualizing the clusters. (See Appendix K, Appendix L & Appendix M)



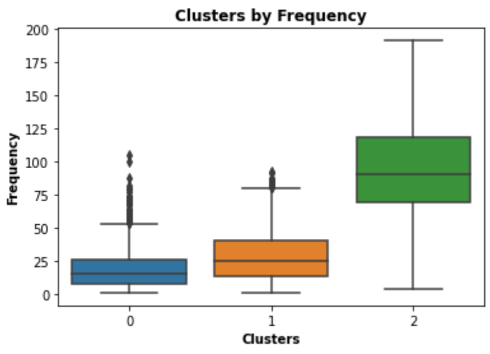
Findings:

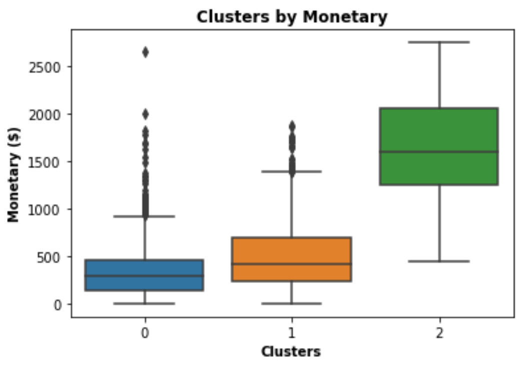
Now that the k-means clustering model has been fit to the dataset, the customer groupings can be analysed to gain insight and improve understanding of consumer behaviour. For our example, we will use box plots to analyse our RFM consumer groupings. (See Appendix N)



We can see that customers in cluster "0" have purchased the least recently and have a low frequency and a lot of outliers on spend. There are outliers in regards to number of purchases and cost of these purchases, however, the average customer in this cluster spends little, infrequently and has not made a purchase recently, for these reasons it would be best to avoid marketing towards these customers.

Customers in cluster "1" have mostly made purchases within the last 60 days, have made fewer total purchases and have lower spending. There are outliers in spending and frequency, however, the data indicates that customers within this cluster are probably new customers. From a business perspective, this would be an attractive segment to market towards as it could lead to an increased consumer base.





Cluster "2" has a high total spend, and number of purchases and has made purchases most recently. Although there seem to be many outliers in recency of last purchase, the high spend and frequency indicate these are customers familiar with the business. This would be a good segment to focus marketing on as they are already accustomed to the business and will be easy to entice to make recurring purchases.

After analysing this data, we can conclude that the marketing teams of businesses in this industry are most likely to use this information to market towards customers in clusters 1 and 2. They are likely to put most spending into customers in cluster 2 as they are most likely to repurchase. They may promote loyalty cards or discounts for repeat or subscription purchases. They also may decide they want to encourage customers in cluster 1 to purchase more items. This may be through discounts as seen in the 'Clusters by Monetary' graph, they spend on average much less than those in cluster 2, therefore it could be inferred that they have less disposable income. These discounts will also promote brand loyalty. They are likely to avoid customers in cluster 0 as they have likely only made a once-off purchase and are unlikely to be loyal to the brand. It would be more beneficial for the business to focus its efforts on customers who already enjoy their products. As this segmentation is done by behaviour and does not consider other factors such as location or age, it may be difficult to decide on which platform advertisements could be done. This is a potential limit of this method.

Limitations

One of the primary limitations of K-means clustering is its sensitivity to outliers in the dataset. To mitigate this limitation, we have implemented code to remove outliers by defining z-scores and interquartile ranges. If we did not use these methods, the result would display a skewed cluster structure in the graph, the centroids would be placed disproportionately, and subsequently, our box plots determining customer behaviour would display incorrect data. If a marketing team was using this data to influence its marketing strategy, it could result in an ineffective effort to influence consumer behaviour. Similarly, the algorithm may try to fit clusters to these outliers which could result in poor results.

K-means clustering assigns data points to one cluster only. In the real world, some data points may belong to multiple clusters. This limitation is compounded by the algorithm struggling with variation in datasets with high-density clusters, where the algorithm will pull data points into these dense clusters when they should be in other, more sparsely populated clusters

Finally, by setting an initial number of clusters to three using the elbow method, we potentially limit the outcome of our algorithm as we cannot be certain of our choice of clusters. It can be difficult to justify the number of clusters used without a deep understanding of the data. In the case of our data, using the elbow method, we subjectively decided that the number of clusters is three however it could also be two. The graph is slightly ambiguous, meaning that we need to use our understanding of the source data to determine the number of clusters we believe will represent our data most effectively (Itokun et al, 2018). The graph is slightly ambiguous, meaning that to determine the correct number of clusters we need to use our understanding of the source data. (Itokun et al, 2018).

These are only three of several limitations associated with K-means clustering and from a business perspective, it is crucial to understand these limitations to minimize the risk. As useful as unsupervised learning is, there can be a lack of accuracy in its interpretations, and due to the outputs being unspecified there can also be a lack of context involved and it can be difficult to evaluate the results of the algorithm used.

Presentation of findings + interpretation of results + Conclusion

Python and K-means clustering are both simple, easy-to-use tools. They grant useful initial data exploration capabilities to a business. Allowing businesses to understand patterns in customer data before more sophisticated analyses are performed. These valuable insights into customer behaviour then complement the decision-making processes.

The ease of use enables seamless integration of preliminary data analysis into business workflows. The results of K-means clustering are easily interpretable, and K-means is an iterative algorithm meaning results can be easily refined based on changing business needs.

We advise the business to focus marketing campaigns on Clusters 1 and 2, implementing strategies such as promotions and loyalty programs to incentivise repeat purchasing. Cluster 0 is not cost-effective to target.

Various limitations of K-Means clustering must be factored in when utilising analysis results. We utilised several methods in an attempt to circumvent said limitations. Skewed cluster structures and disproportionately placed centroids were avoided through the implementation of code to remove outliers by defining z-scores and interquartile ranges.

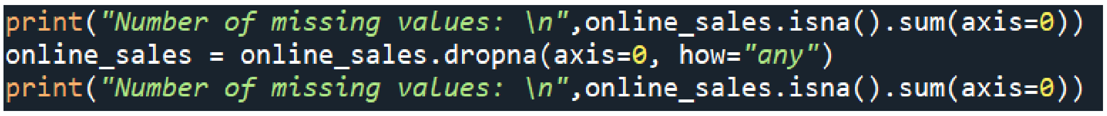
Regarding process limitations, if we better understood the source data, we could have better chosen the number of clusters/customer types. Customer behaviour was the sole segment divider. Choosing to utilize the elbow method did not allow precise customer knowledge to factor into the cluster amount decision. As behaviour-driven segmentation also does not consider factors such as location or age, it may be difficult to decide on which platform to advertise our recommended promotions. Again, graph ambiguity can be counteracted with well-understood source data.

Although customer segmentation lacks demographic considerations, the insights provide a starting point for strategic decision-making in marketing optimisation efforts. K-means will meet your demand for an initial impression of your various consumer types. A caveat of the algorithm is that it returns an interpretive answer – someone must have the knowledge/industry experience to correctly interpret the clusters.

If you require a similar analysis of your data, Hierarchical clustering is a method of unsupervised learning which could be used. It does not require a specific number of clusters and the resulting data dendrogram can be pruned at different levels to form clusters. The dataset we used contained 542,000 lines and therefore hierarchical clustering could have been too computationally intensive.

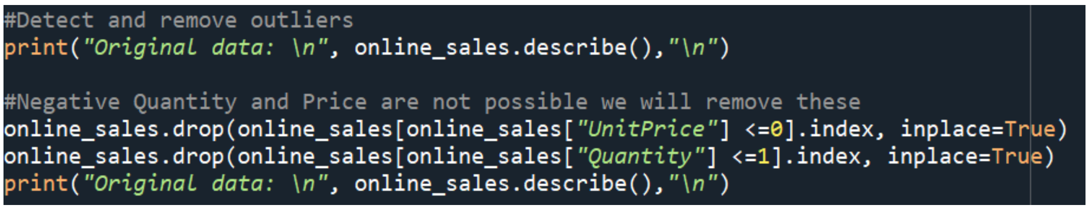
Appendix

A. Code for missing values



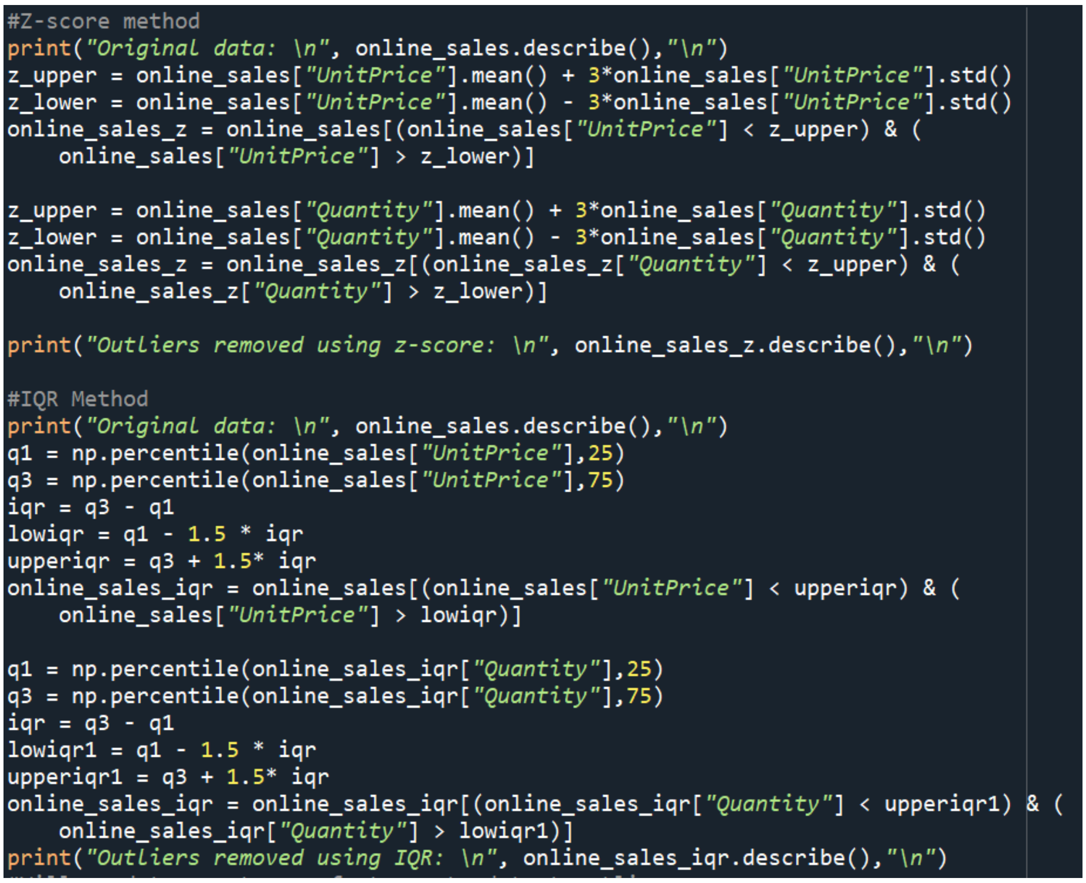
```
print("Number of missing values: \n",online_sales.isna().sum(axis=0))
online_sales = online_sales.dropna(axis=0, how="any")
print("Number of missing values: \n",online_sales.isna().sum(axis=0))
```

B. Code for describing data and removing impossible values



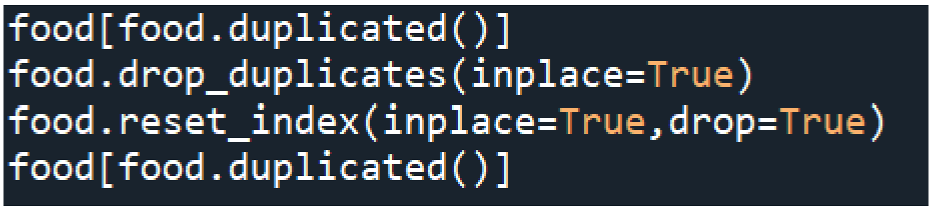
```
#Detect and remove outliers
print("Original data: \n", online_sales.describe(),"\n")

#Negative Quantity and Price are not possible we will remove these
online_sales.drop(online_sales[online_sales["UnitPrice"] <=0].index, inplace=True)
online_sales.drop(online_sales[online_sales["Quantity"] <=1].index, inplace=True)
print("Original data: \n", online_sales.describe(),"\n")
```

C. Code for removing outliers using IQR & Z-Score method



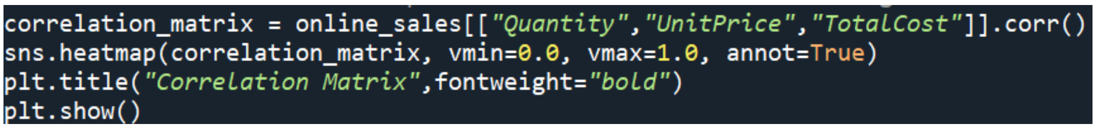
```
#Z-score method
print("Original data: \n", online_sales.describe(),"\n")
z_upper = online_sales["UnitPrice"].mean() + 3*online_sales["UnitPrice"].std()
z_lower = online_sales["UnitPrice"].mean() - 3*online_sales["UnitPrice"].std()
online_sales_z = online_sales[(online_sales["UnitPrice"] < z_upper) & (
    online_sales["UnitPrice"] > z_lower)]

z_upper = online_sales["Quantity"].mean() + 3*online_sales["Quantity"].std()
z_lower = online_sales["Quantity"].mean() - 3*online_sales["Quantity"].std()
online_sales_z = online_sales_z[(online_sales_z["Quantity"] < z_upper) & (
    online_sales_z["Quantity"] > z_lower)]

print("Outliers removed using z-score: \n", online_sales_z.describe(),"\n")

#IQR Method
print("Original data: \n", online_sales.describe(),"\n")
q1 = np.percentile(online_sales["UnitPrice"],25)
q3 = np.percentile(online_sales["UnitPrice"],75)
iqr = q3 - q1
lowiqr = q1 - 1.5 * iqr
upperiqr = q3 + 1.5* iqr
online_sales_iqr = online_sales[(online_sales["UnitPrice"] < upperiqr) & (
    online_sales["UnitPrice"] > lowiqr)]

q1 = np.percentile(online_sales_iqr["Quantity"],25)
q3 = np.percentile(online_sales_iqr["Quantity"],75)
iqr = q3 - q1
lowiqr1 = q1 - 1.5 * iqr
upperiqr1 = q3 + 1.5* iqr
online_sales_iqr = online_sales_iqr[(online_sales_iqr["Quantity"] < upperiqr1) & (
    online_sales_iqr["Quantity"] > lowiqr1)]
print("Outliers removed using IQR: \n", online_sales_iqr.describe(),"\n")
```

D. Sample code showing how duplicates would be removed



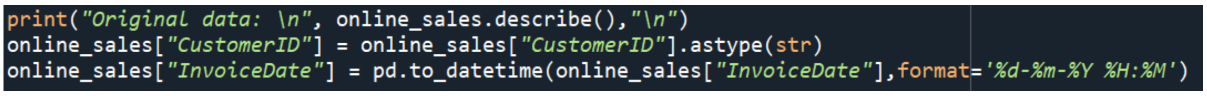
```
food[food.duplicated()]
food.drop_duplicates(inplace=True)
food.reset_index(inplace=True,drop=True)
food[food.duplicated()]
```

E. Code to create a correlation matrix and graph with a heatmap



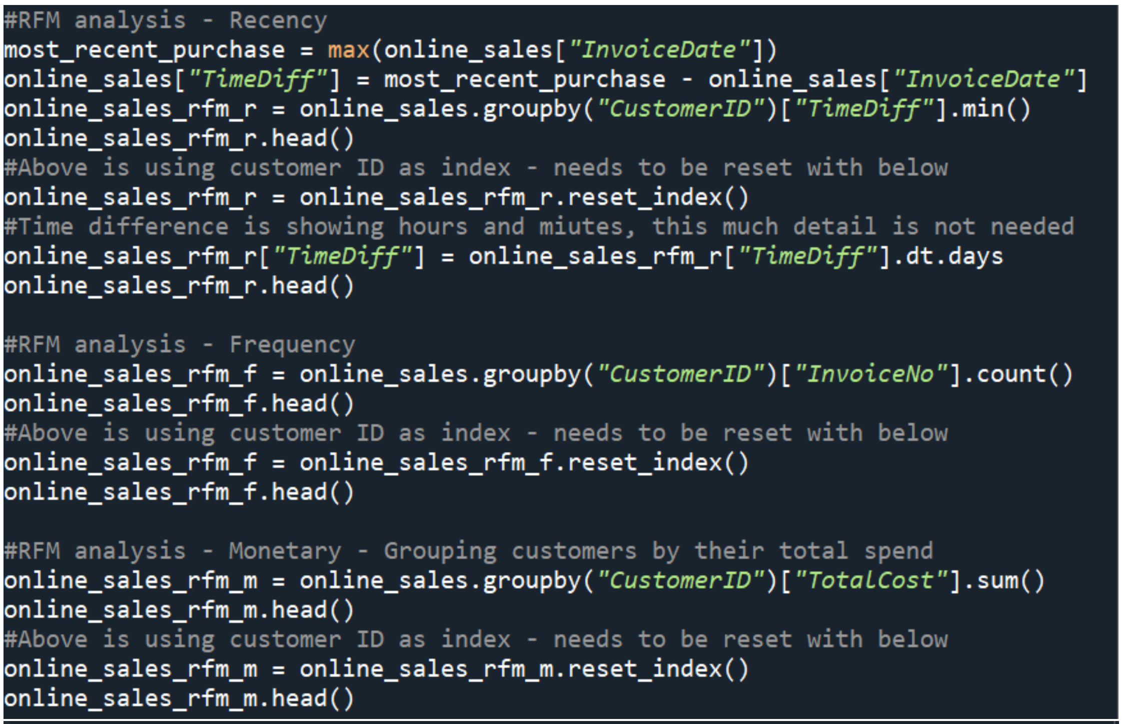
```
correlation_matrix = online_sales[["Quantity","UnitPrice","TotalCost"]].corr()
sns.heatmap(correlation_matrix, vmin=0.0, vmax=1.0, annot=True)
plt.title("Correlation Matrix",fontweight="bold")
plt.show()
```

F. Code adjusting formatting on InvoiceDate and CustomerID features

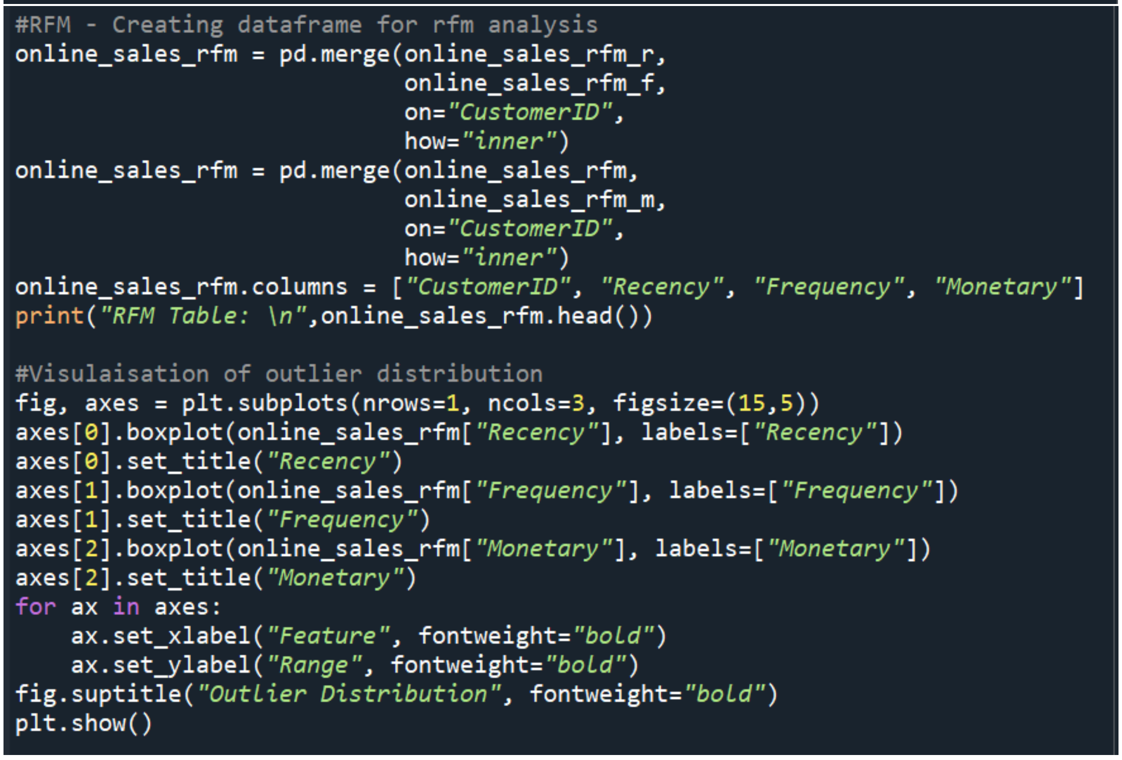


```
print("Original data: \n", online_sales.describe(),"\n")
online_sales["CustomerID"] = online_sales["CustomerID"].astype(str)
online_sales["InvoiceDate"] = pd.to_datetime(online_sales["InvoiceDate"],format='%d-%m-%Y %H:%M')
```

G. Code to create RFM table and show outliers on the boxplot

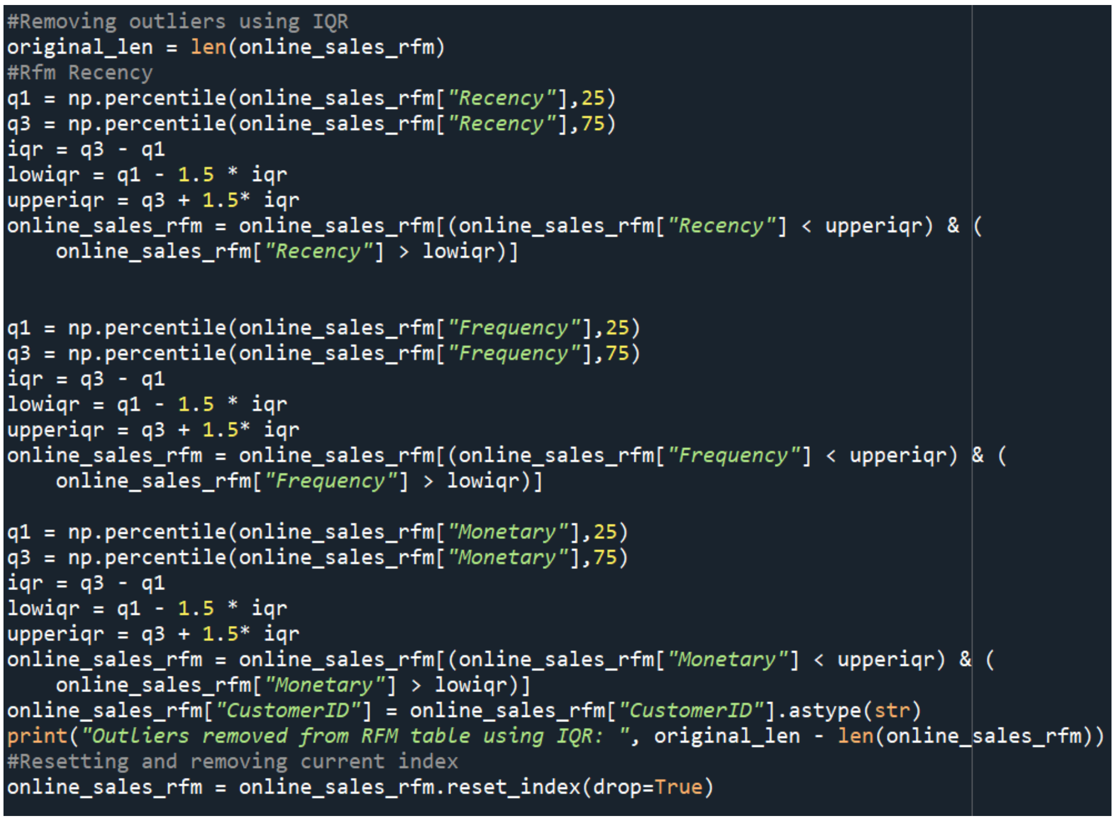


```
#RFM analysis - Recency
most_recent_purchase = max(online_sales["InvoiceDate"])
online_sales["TimeDiff"] = most_recent_purchase - online_sales["InvoiceDate"]
online_sales_rfm_r = online_sales.groupby("CustomerID")["TimeDiff"].min()
online_sales_rfm_r.head()
#Above is using customer ID as index - needs to be reset with below
online_sales_rfm_r = online_sales_rfm_r.reset_index()
#Time difference is showing hours and miutes, this much detail is not needed
online_sales_rfm_r["TimeDiff"] = online_sales_rfm_r["TimeDiff"].dt.days
online_sales_rfm_r.head()

#RFM analysis - Frequency
online_sales_rfm_f = online_sales.groupby("CustomerID")["InvoiceNo"].count()
online_sales_rfm_f.head()
#Above is using customer ID as index - needs to be reset with below
online_sales_rfm_f = online_sales_rfm_f.reset_index()
online_sales_rfm_f.head()

#RFM analysis - Monetary - Grouping customers by their total spend
online_sales_rfm_m = online_sales.groupby("CustomerID")["TotalCost"].sum()
online_sales_rfm_m.head()
#Above is using customer ID as index - needs to be reset with below
online_sales_rfm_m = online_sales_rfm_m.reset_index()
online_sales_rfm_m.head()
```



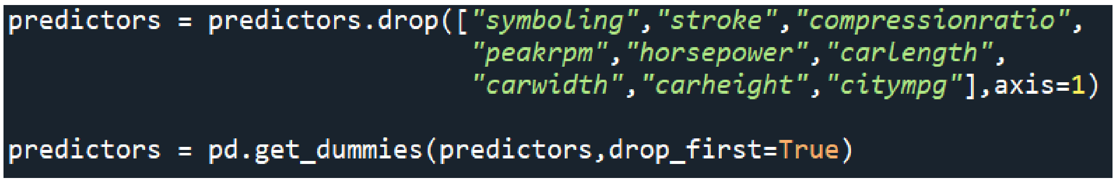
```
#RFM - Creating dataframe for rfm analysis
online_sales_rfm = pd.merge(online_sales_rfm_r,
                            online_sales_rfm_f,
                            on="CustomerID",
                            how="inner")
online_sales_rfm = pd.merge(online_sales_rfm,
                            online_sales_rfm_m,
                            on="CustomerID",
                            how="inner")
online_sales_rfm.columns = ["CustomerID", "Recency", "Frequency", "Monetary"]
print("RFM Table: \n",online_sales_rfm.head())

#Visulaisation of outlier distribution
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15,5))
axes[0].boxplot(online_sales_rfm["Recency"], labels=["Recency"])
axes[0].set_title("Recency")
axes[1].boxplot(online_sales_rfm["Frequency"], labels=["Frequency"])
axes[1].set_title("Frequency")
axes[2].boxplot(online_sales_rfm["Monetary"], labels=["Monetary"])
axes[2].set_title("Monetary")
for ax in axes:
    ax.set_xlabel("Feature", fontweight="bold")
    ax.set_ylabel("Range", fontweight="bold")
fig.suptitle("Outlier Distribution", fontweight="bold")
plt.show()
```

H. Code for removing outliers using IQR from new data frame



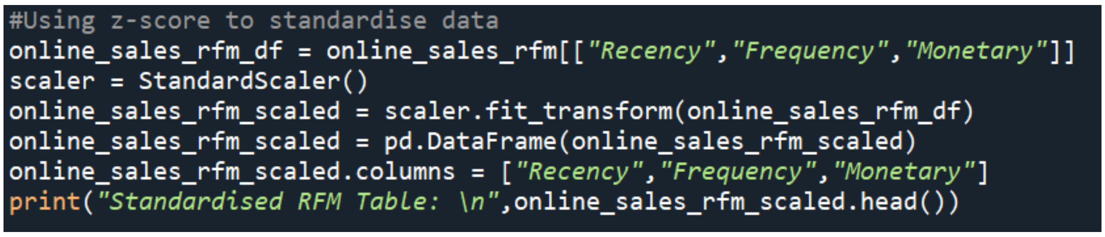
```
#Removing outliers using IQR
original_len = len(online_sales_rfm)
#Rfm Recency
q1 = np.percentile(online_sales_rfm["Recency"],25)
q3 = np.percentile(online_sales_rfm["Recency"],75)
iqr = q3 - q1
lowiqr = q1 - 1.5 * iqr
upperiqr = q3 + 1.5* iqr
online_sales_rfm = online_sales_rfm[(online_sales_rfm["Recency"] < upperiqr) & (
    online_sales_rfm["Recency"] > lowiqr)]


q1 = np.percentile(online_sales_rfm["Frequency"],25)
q3 = np.percentile(online_sales_rfm["Frequency"],75)
iqr = q3 - q1
lowiqr = q1 - 1.5 * iqr
upperiqr = q3 + 1.5* iqr
online_sales_rfm = online_sales_rfm[(online_sales_rfm["Frequency"] < upperiqr) & (
    online_sales_rfm["Frequency"] > lowiqr)]

q1 = np.percentile(online_sales_rfm["Monetary"],25)
q3 = np.percentile(online_sales_rfm["Monetary"],75)
iqr = q3 - q1
lowiqr = q1 - 1.5 * iqr
upperiqr = q3 + 1.5* iqr
online_sales_rfm = online_sales_rfm[(online_sales_rfm["Monetary"] < upperiqr) & (
    online_sales_rfm["Monetary"] > lowiqr)]
online_sales_rfm["CustomerID"] = online_sales_rfm["CustomerID"].astype(str)
print("Outliers removed from RFM table using IQR: ", original_len - len(online_sales_rfm))
#Resetting and removing current index
online_sales_rfm = online_sales_rfm.reset_index(drop=True)
```

I. One hot encoding example code



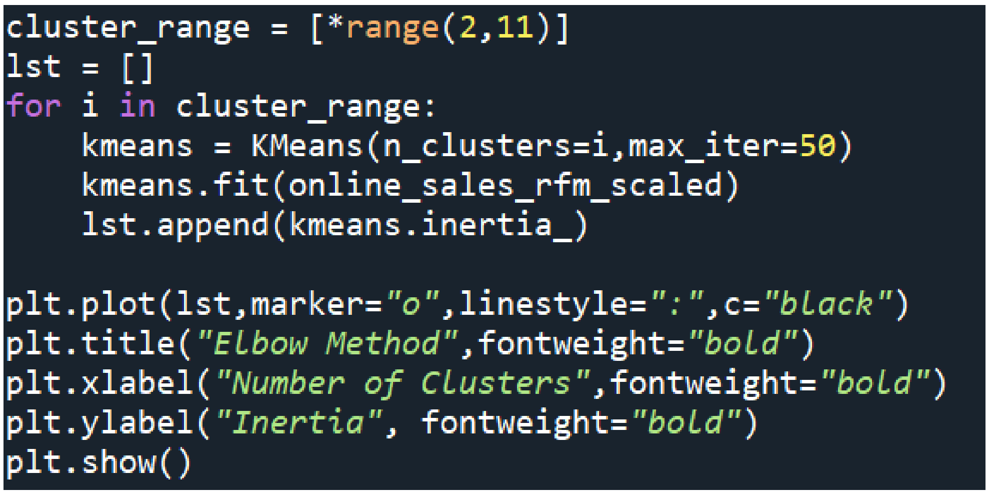
```
predictors = predictors.drop(["symboling","stroke","compressionratio",
                              "peakrpm","horsepower","carlength",
                              "carwidth","carheight","citympg"],axis=1)

predictors = pd.get_dummies(predictors,drop_first=True)
```

J. Code to standardise data using Z-Score



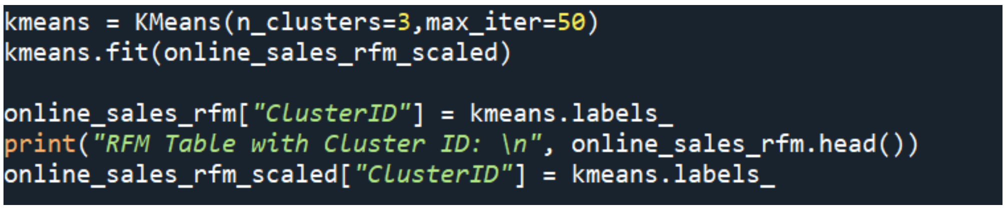
```
#Using z-score to standardise data
online_sales_rfm_df = online_sales_rfm[["Recency","Frequency","Monetary"]]
scaler = StandardScaler()
online_sales_rfm_scaled = scaler.fit_transform(online_sales_rfm_df)
online_sales_rfm_scaled = pd.DataFrame(online_sales_rfm_scaled)
online_sales_rfm_scaled.columns = ["Recency","Frequency","Monetary"]
print("Standardised RFM Table: \n",online_sales_rfm_scaled.head())
```

K. Code to create elbow graph for k-means clustering



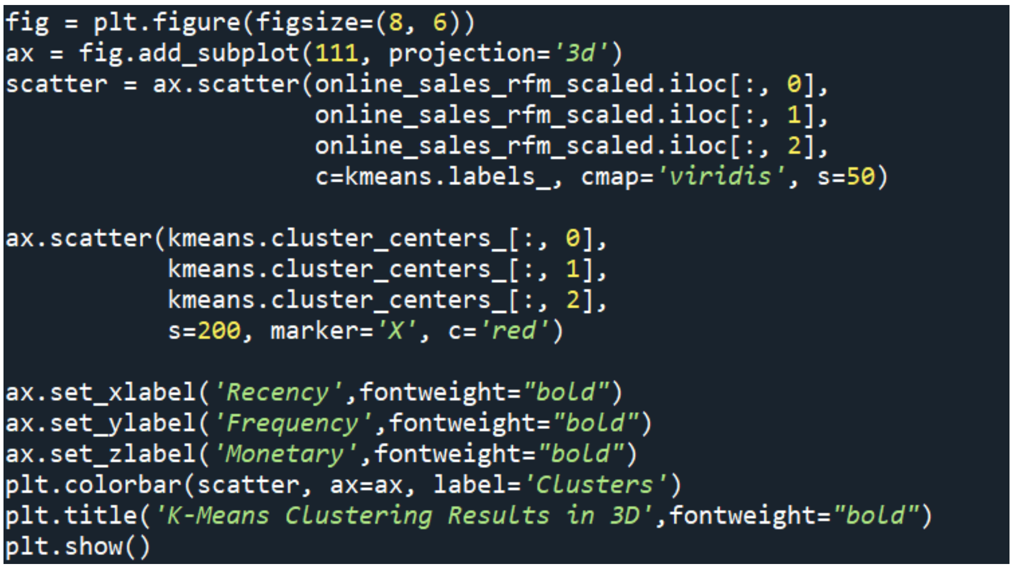
```
cluster_range = [*range(2,11)]
lst = []
for i in cluster_range:
    kmeans = KMeans(n_clusters=i,max_iter=50)
    kmeans.fit(online_sales_rfm_scaled)
    lst.append(kmeans.inertia_)

plt.plot(lst,marker="o",linestyle=":",c="black")
plt.title("Elbow Method",fontweight="bold")
plt.xlabel("Number of Clusters",fontweight="bold")
plt.ylabel("Inertia", fontweight="bold")
plt.show()
```

L. Code to apply k-means to the dataset



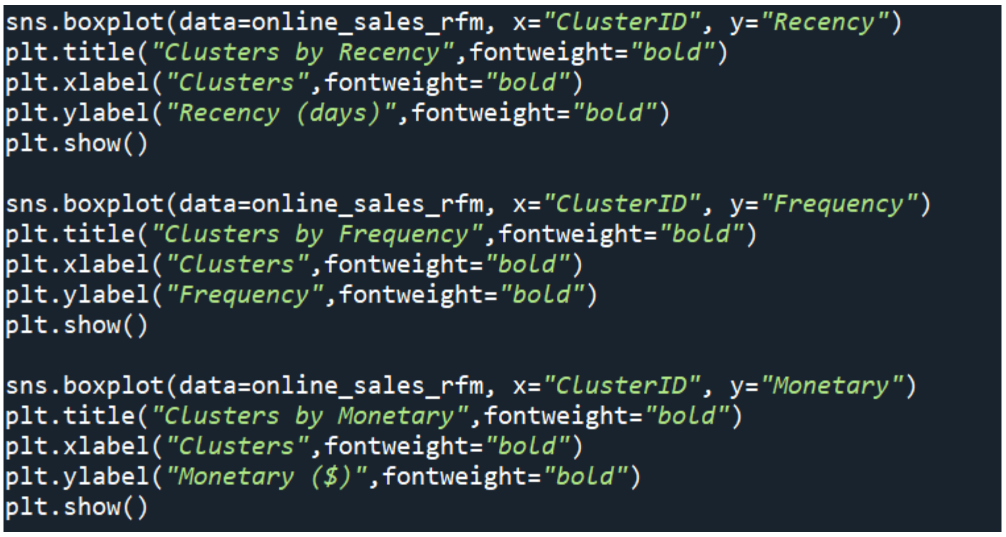
```python
kmeans = KMeans(n_clusters=3,max_iter=50)
kmeans.fit(online_sales_rfm_scaled)

online_sales_rfm["ClusterID"] = kmeans.labels_
print("RFM Table with Cluster ID: \n", online_sales_rfm.head())
online_sales_rfm_scaled["ClusterID"] = kmeans.labels_
```

M. Code to create a 3D clustering graph



```python
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(online_sales_rfm_scaled.iloc[:, 0],
                     online_sales_rfm_scaled.iloc[:, 1],
                     online_sales_rfm_scaled.iloc[:, 2],
                     c=kmeans.labels_, cmap='viridis', s=50)

ax.scatter(kmeans.cluster_centers_[:, 0],
           kmeans.cluster_centers_[:, 1],
           kmeans.cluster_centers_[:, 2],
           s=200, marker='X', c='red')

ax.set_xlabel('Recency',fontweight="bold")
ax.set_ylabel('Frequency',fontweight="bold")
ax.set_zlabel('Monetary',fontweight="bold")
plt.colorbar(scatter, ax=ax, label='Clusters')
plt.title('K-Means Clustering Results in 3D',fontweight="bold")
plt.show()
```

N. Code to create boxplots for the finalised dataset



```python
sns.boxplot(data=online_sales_rfm, x="ClusterID", y="Recency")
plt.title("Clusters by Recency",fontweight="bold")
plt.xlabel("Clusters",fontweight="bold")
plt.ylabel("Recency (days)",fontweight="bold")
plt.show()

sns.boxplot(data=online_sales_rfm, x="ClusterID", y="Frequency")
plt.title("Clusters by Frequency",fontweight="bold")
plt.xlabel("Clusters",fontweight="bold")
plt.ylabel("Frequency",fontweight="bold")
plt.show()

sns.boxplot(data=online_sales_rfm, x="ClusterID", y="Monetary")
plt.title("Clusters by Monetary",fontweight="bold")
plt.xlabel("Clusters",fontweight="bold")
plt.ylabel("Monetary ($)",fontweight="bold")
plt.show()
```

References

Alsayat, A. (2023). Customer decision-making analysis based on big social data using machine learning: A case study of hotels in Mecca. *Neural Computing & Applications, 35*(6), 4701–4722. https://doi.org/10.1007/s00521-022-07992-x

Elrod, C., Stanley, S., Cudney, E., & Fisher, C. (2015). Empirical study utilizing QFD to develop an international marketing strategy. *Sustainability, 7*(8), 10756–10769. doi:10.3390/su70810756

Florez-Lopez, R. (2016). Marketing segmentation through machine learning models: An approach based on customer relationship management and customer profitability accounting. *Social Science Computer Review*, *27*(3), 297–317. doi:10.1177/0894439308321592

Huseynov, Farid, & Ozkan, Sevgi. (2017). Behavioural segmentation analysis of online consumer audience in Turkey by using real e-commerce transaction data. *International Journal of Economics and Business Research, 14*(12). doi:10.1504/IJEBR.2017.085549

Ikotun, A. M., Ezugwu, A. E., Abualigah, L., Abuhaija, B., & Heming, J. (2023). K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences, 622*, 178–210. https://doi.org/10.1016/j.ins.2022.11.139

Kotler, P., et al. (2014). *Marketing management* (14th ed.). Pearson.

Mahfuza, R., Islam, N., Toyeb, M., Emon, M.A.F., Chowdhury, S.A., & Alam, M.G.R. (2022). LRFMV: An efficient customer segmentation model for superstores. *PLoS ONE, 17*(12), e0279262. https://doi.org/10.1371/journal.pone.0279262

Pappas, I. O. (2018). User experience in personalized online shopping: A fuzzy-set analysis. *European Journal of Marketing, 52*(7/8), 1679–1703. doi:https://doi.org/10.1108/EJM.

Plangger, K., Marder, B., Matteo Montecchi, Watson, R.T., & Pitt, L. (2023). Does (customer data) size matter? Generating valuable customer insights with less customer relationship risk. *Psychology & Marketing, 40*(10), 2016–2028. doi:10.1002/mar.21866

Vara, N., Mahdieh Mirzabeigi, Hajar Sotudeh, & Seyed Mostafa Fakhrahmad. (2022). Application of k-means clustering algorithm to improve effectiveness of the results recommended by journal recommender system. Scientometrics, 127(6), 3237–3252. https://doi.org/10.1007/s11192-022-04397-4