Project

Data Mining

Haresh p Tayade

PGP-DSBA Online

Sept’2021

Date-26/01/2022

Project: Data Mining

Contents

CLUSTERING

(Problem 1)

1. Read the data, do the necessary initial steps, and exploratory data analysis Univariate, Bi-variate, and multivariate analysis).
2. Do you think Scaling is necessary for clustering in this case? Justify
3. Apply Hierarchical clustering to scaled data. Identify the number of optimum clusters using Dendrogram and briefly describe them.
4. Apply K-means Clustering on scaled data and determine optimum clusters. Apply elbow curve and silhouette score. Explain the results properly. Interpret and write inferences on the finalized clusters.
5. Describe the cluster profile for the clusters defined. Recommend different promotional strategies for different clusters.

CART - RF - ANN

(Problem 2)

1. Read the data, do the necessary initial steps, and exploratory data analysis Univariate, Bi-variate, and multivariate analysis).
2. Data Split: Split the data into test and train, build classification model CART, Random Forest, Artificial Neural Network.
3. Performance Metrics: Comment and check performance of predictions on train and test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score, classification reports for each model.
4. Final model: Compare all the models and write and inference which model is best optimized.
5. Inference: Based on the whole analysis, what are the business insights and recommendations.

Executive Summary:

Clustering¶

A leading bank wants to develop a customer segmentation to give promotional offers to its customers. They collected a sample that summarizes the activities of users during the past few months. You are given the task to identify the segments based on credit card usage.

CART-RF-ANN

An Insurance firm providing tour insurance is facing higher claim frequency. The management decides to collect data from the past few years. You are assigned the task to make a model which predicts the claim status and provide recommendations to management. Use CART, RF & ANN and compare the models' performances in train and test sets.

1.1

Read the data, do the necessary initial steps, and exploratory data analysis Univariate, Bi-variate, and multivariate analysis).

Answer:

As below checked the dataset:

```
df.head()
```

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.550
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144
2	18.95	16.42	0.8829	6.248	3.755	3.368	6.148
3	10.83	12.96	0.8099	5.278	2.641	5.182	5.185
4	17.99	15.86	0.8992	5.890	3.694	2.068	5.837

We found that the dataset has 210 rows and 7 columns.

```
df.shape
```

```
(210, 7)
```

As description of data set

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
spending	210.0	14.847524	2.909699	10.5900	12.27000	14.35500	17.305000	21.1800
advance_payments	210.0	14.559286	1.305959	12.4100	13.45000	14.32000	15.715000	17.2500
probability_of_full_payment	210.0	0.870999	0.023629	0.8081	0.85690	0.87345	0.887775	0.9183
current_balance	210.0	5.628533	0.443063	4.8990	5.26225	5.52350	5.979750	6.6750
credit_limit	210.0	3.258605	0.377714	2.6300	2.94400	3.23700	3.561750	4.0330
min_payment_amt	210.0	3.700201	1.503557	0.7651	2.56150	3.59900	4.768750	8.4560
max_spent_in_single_shopping	210.0	5.408071	0.491480	4.5190	5.04500	5.22300	5.877000	6.5500

There are No null value in dataset.

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

```
False
```

There are no duplicates in given dataset.

```
df.duplicated().sum()
```

```
0
```

Univariate analysis/bi-variate analysis

Univariate analysis **explores each variable in a data set**, separately. It looks at the range of values, as well as the central tendency of the values. It describes the pattern of response to the variable. It describes each variable on its own.

Bivariate analysis is one **of the simplest forms of quantitative** (statistical) analysis. It involves the analysis of two variables (often denoted as X, Y), for the purpose of determining the empirical relationship between them. ... It is the analysis of the relationship between the two variables.

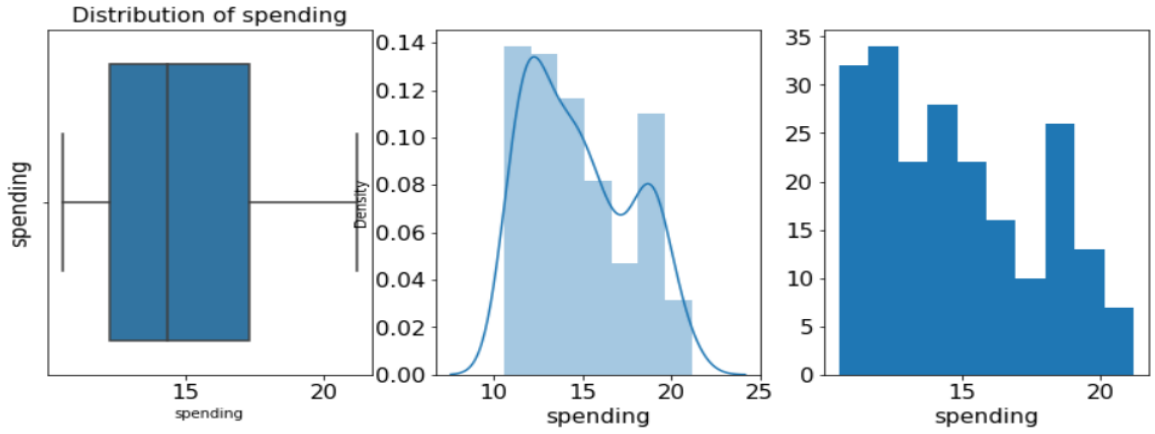
As need to check the min, max, mean, median, std for all the above 7 variables with BOX PLOT, DIST PLOT, HISTOGRAM

7 variables (spending, advance_payments, probability_of_full_payment, current_balance, credit_limit, min_payment_amt, max_spent_in_single_shopping)

SPENDING

```
Minimum spending:  10.59
Maximum spending : 21.18
Mean spending : 14.847523809523818
Median spending : 14.355
STD spending : 2.909699430687361
```

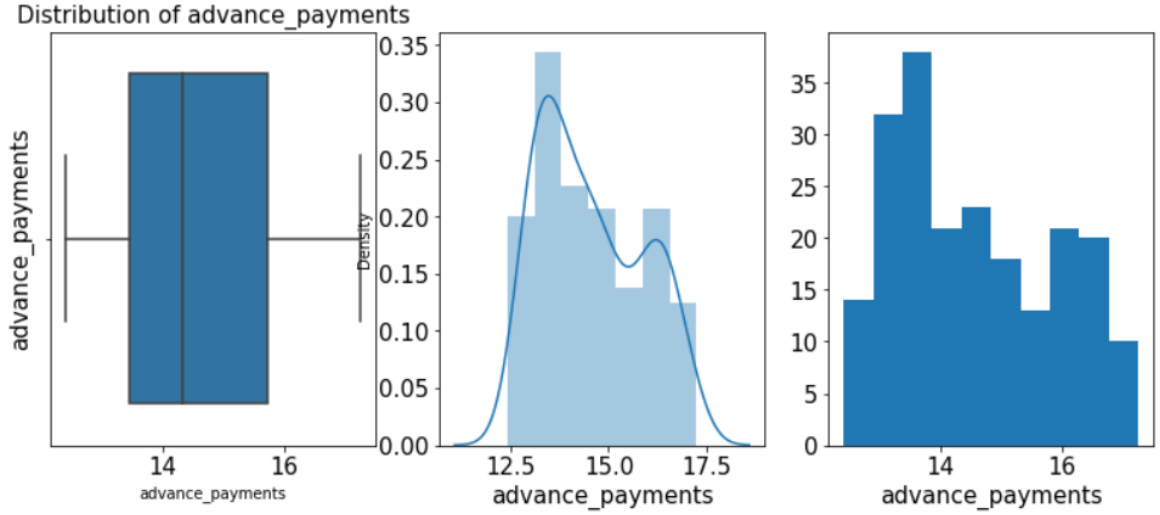
As from above table we can see the min, max, mean, median, STD deviation of spending



As from above figure we can check the outliers and histogram

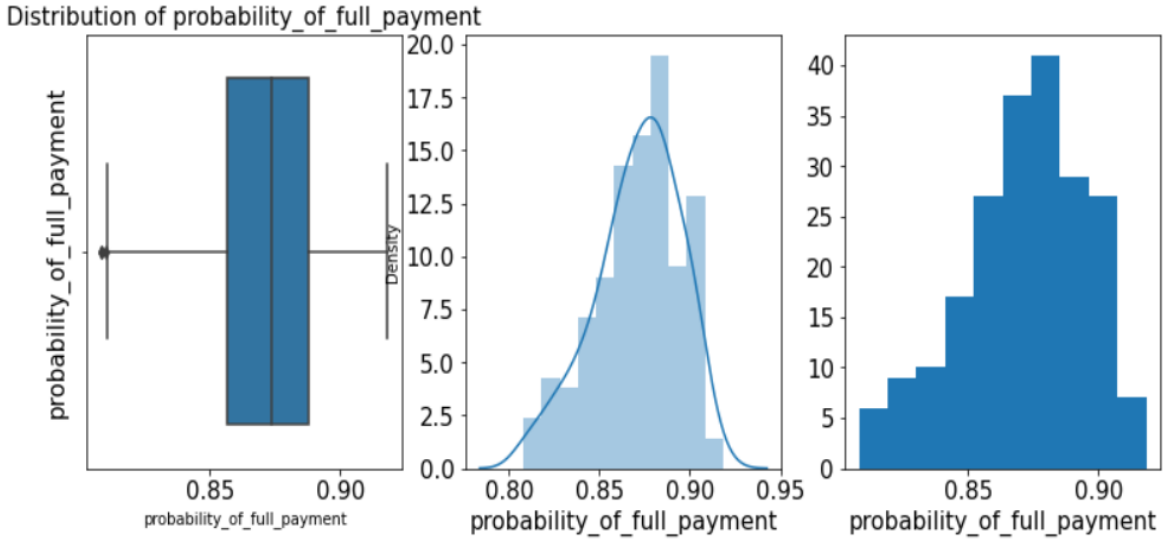
Advance_payment

```
Minimum advance_payments:  12.41
Maximum advance_payments : 17.25
Mean advance_payments : 14.559285714285727
Median advance_payments : 14.32
STD advance_payments : 1.305958726564022
```



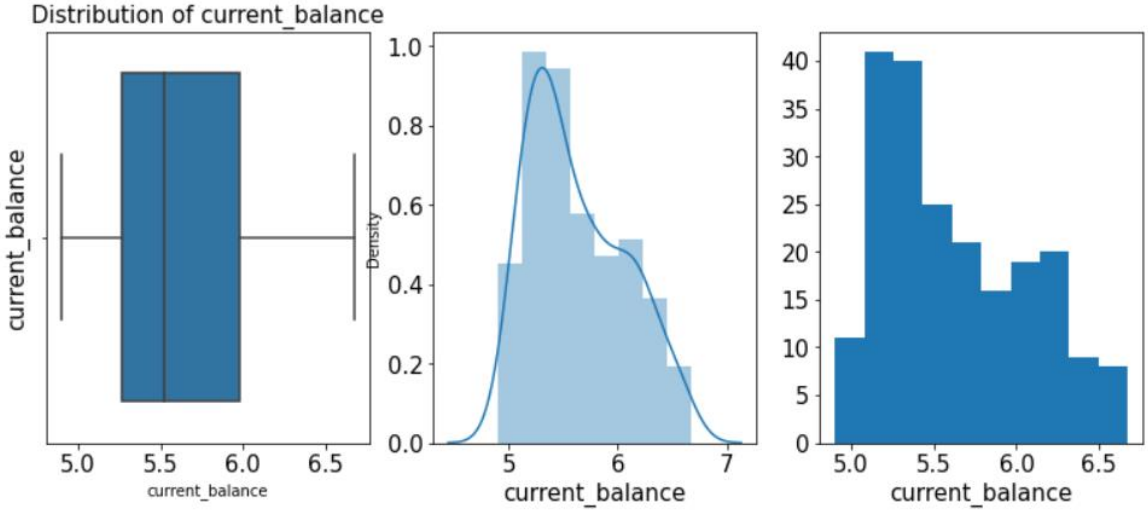
probability_of_full_payment

```
Minimum probability_of_full_payment:  0.8081
Maximum probability_of_full_payment : 0.9183
Mean probability_of_full_payment : 0.8709985714285714
Median probability_of_full_payment : 0.8734500000000001
STD probability_of_full_payment : 0.0236294165838465
```



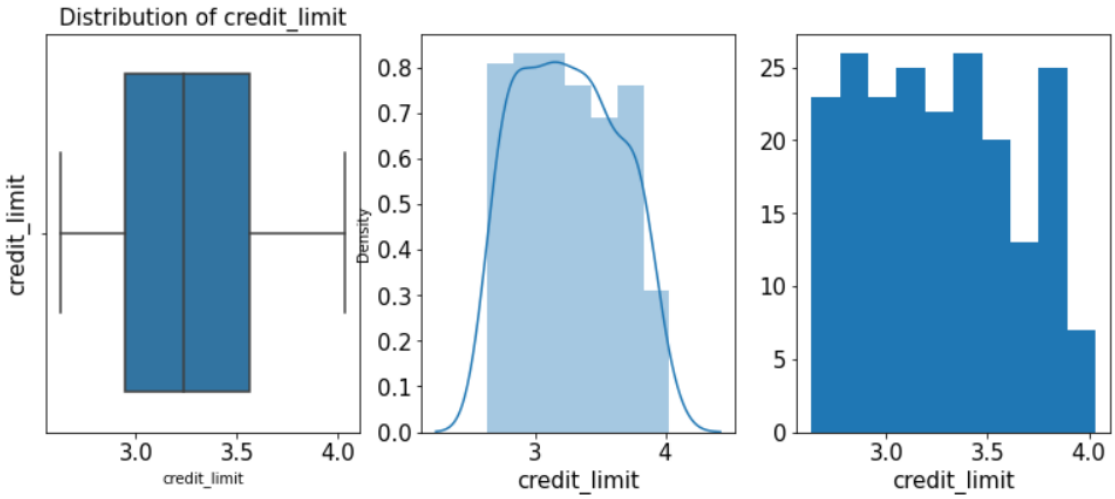
current_balance

```
Minimum current_balance:  4.899
Maximum current_balance : 6.675
Mean current_balance : 5.628533333333335
Median current_balance : 5.5235
STD current_balance : 0.44306347772644944
```



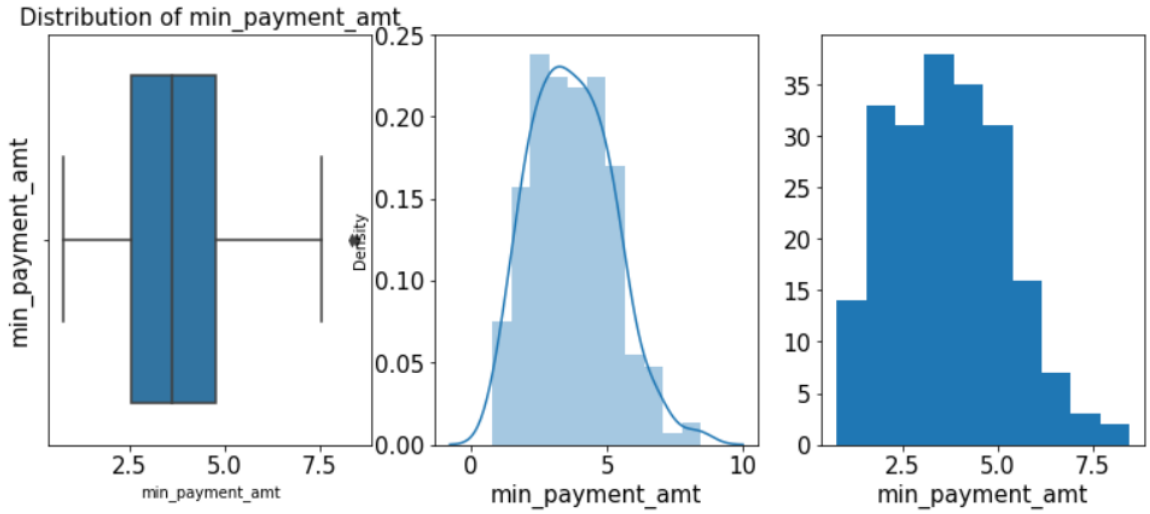
credit_limit

```
Minimum credit_limit:  2.63
Maximum credit_limit : 4.033
Mean credit_limit : 3.258604761904763
Median credit_limit : 3.237
STD credit_limit : 0.37771444490658734
```



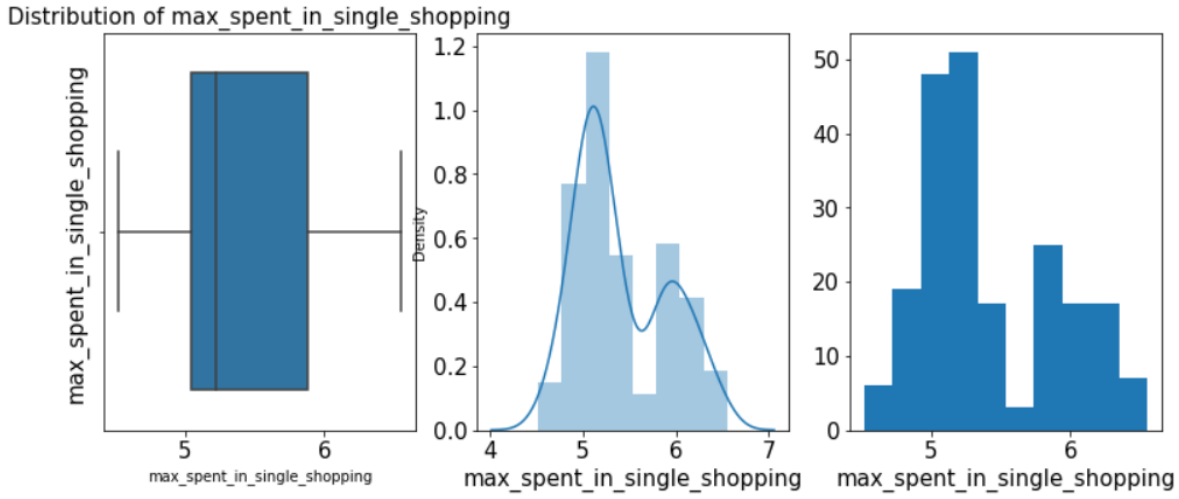
min_payment_amt

```
Minimum min_payment_amt:  0.7651
Maximum min_payment_amt : 8.456
Mean min_payment_amt : 3.7002009523809503
Median min_payment_amt : 3.599
STD min_payment_amt : 1.5035571308217792
```



max_spent_in_single_shopping

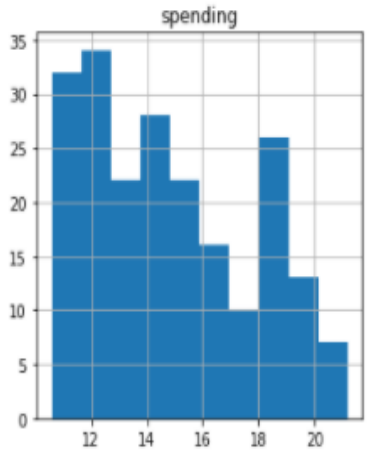
```
Minimum max_spent_in_single_shopping:  4.519
Maximum max_spent_in_single_shopping : 6.55
Mean max_spent_in_single_shopping : 5.408071428571429
Median max_spent_in_single_shopping : 5.223000000000001
STD max_spent_in_single_shopping : 0.49148049910240543
```

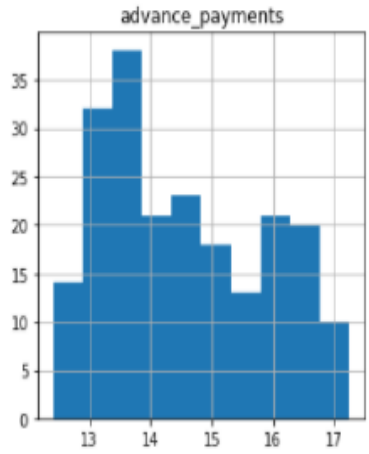


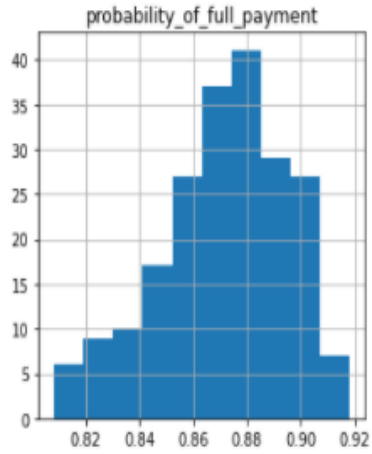
As from all 7 Variables we have seen their min value ,max value etc.

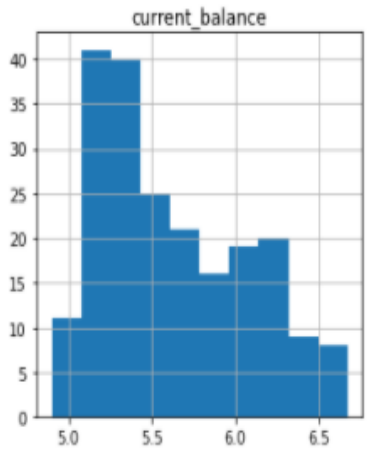
AS we can check histogram for all

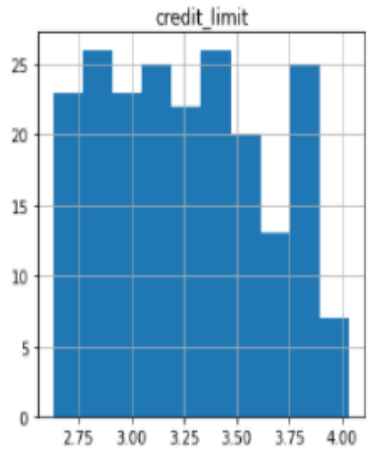
```
<AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```

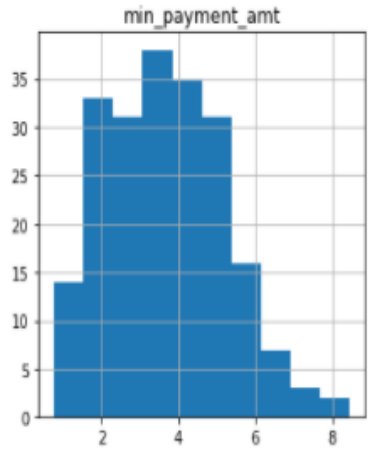


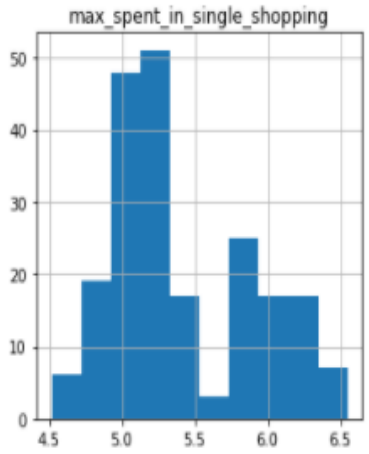













```
probability_of_full_payment    -0.537954
credit_limit                    0.134378
advance_payments                0.386573
spending                        0.399889
min_payment_amt                 0.401667
current_balance                 0.525482
max_spent_in_single_shopping    0.561897
dtype: float64
```

checking the skew value .

Multivariate analysis

Pairplot for given data.

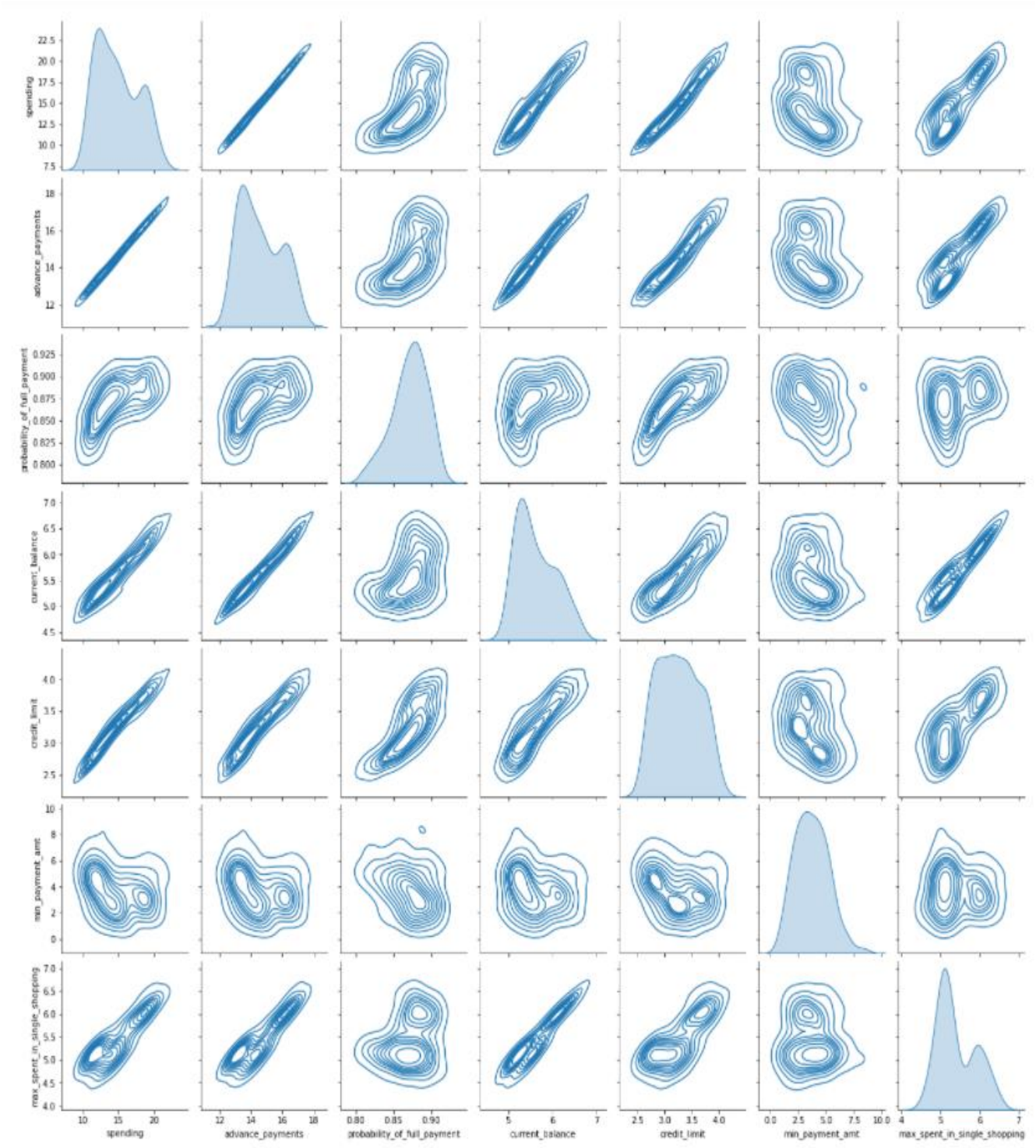
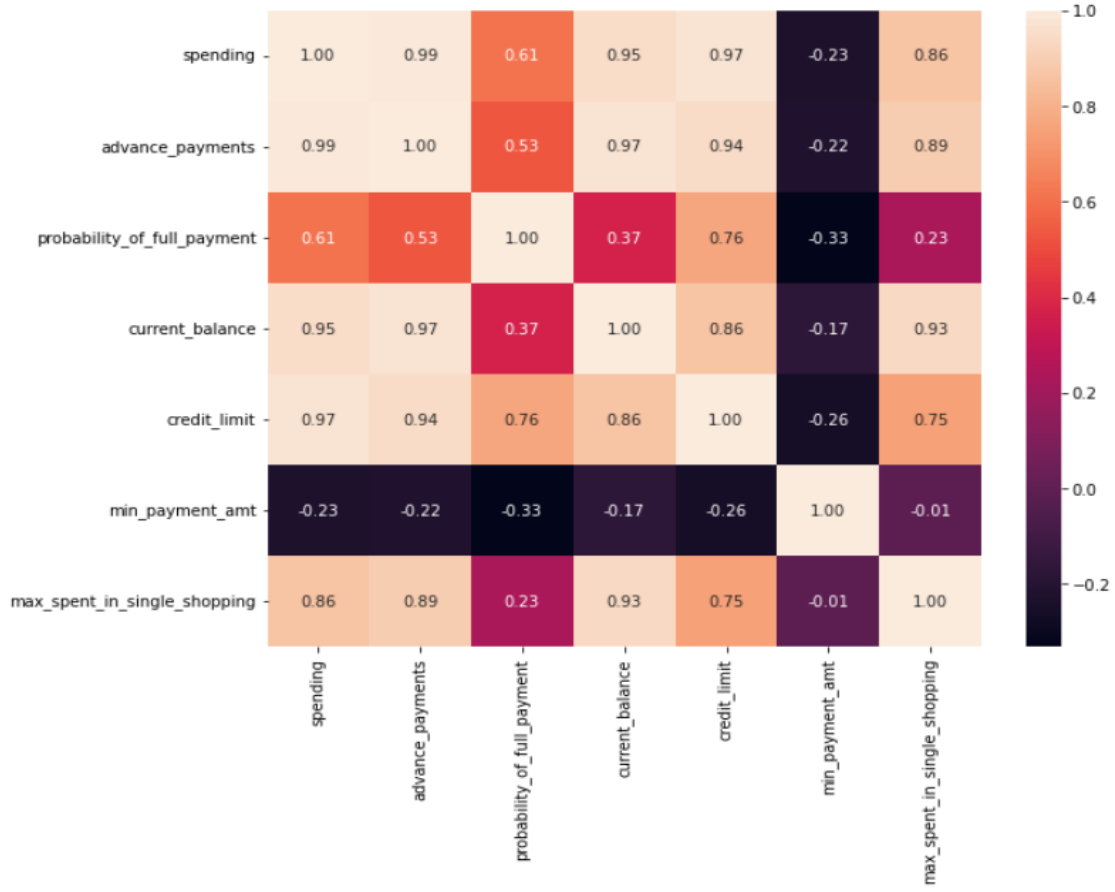


Figure : Pair plot for relation of variables

Observation:

- Strong positive correlation between
- spending & advance_payments,
- advance_payments & current_balance,
- credit_limit & spending
- spending & current_balance
- credit_limit & advance_payments
- max_spent_in_single_shopping current_balance

Now we need to find out the correlation between the variables.



	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
spending	1.000000	0.994341	0.608288	0.949985	0.970771	-0.229572	0.863693
advance_payments	0.994341	1.000000	0.529244	0.972422	0.944829	-0.217340	0.890784
ability_of_full_payment	0.608288	0.529244	1.000000	0.367915	0.761635	-0.331471	0.226825
current_balance	0.949985	0.972422	0.367915	1.000000	0.860415	-0.171562	0.932806
credit_limit	0.970771	0.944829	0.761635	0.860415	1.000000	-0.258037	0.749131
min_payment_amt	-0.229572	-0.217340	-0.331471	-0.171562	-0.258037	1.000000	-0.011079
ent_in_single_shopping	0.863693	0.890784	0.226825	0.932806	0.749131	-0.011079	1.000000

Table for correlation for all variables.

From above as we can see that

- the correlation between spending and advance payment is 0.99.
- correlation between spending and probability of full payment is 0.61.
- correlation between spending and current balance is 0.95
- correlation between spending and credit limit is 0.97.
- correlation between spending and min payment amt is -0.23.
- correlation between spending and max spent in single shopping is 0.86.

- Correlation between advance payment and probability of full payment is 0.53.
- Correlation between advance payment and current balance is 0.97.
- Correlation between advance payment and credit limit is 0.94.
- Correlation between advance payment and min payment amt is -0.22
- Correlation between advance payment and max spent in single shopping is 0.89

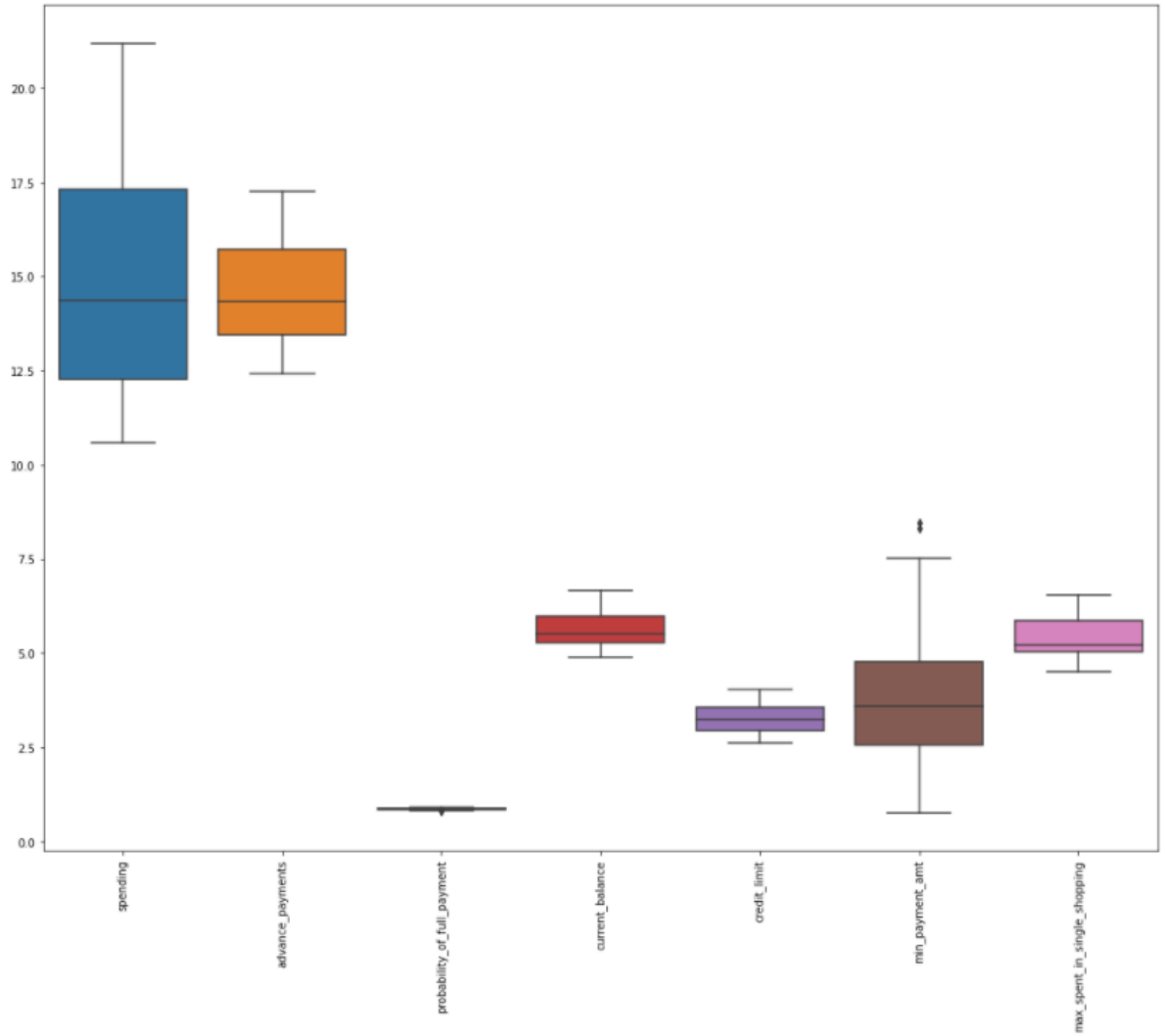
- Correlation between probability of full payment and current balance is 0.37
- Correlation between probability of full payment and credit limit is 0.76
- Correlation between probability of full payment and min payment amount is -0.33
- Correlation between probability of full payment and max spent in single shopping is 0.23

- Correlation between current balance and credit limit is 0.86
- Correlation between current balance and min payment amount is -0.17
- Correlation between current balance and max spent in single shopping is 0.93

- Correlation between credit limit and min payment amount is -0.26
- Correlation between credit limit and max spent in single shopping is 0.75
- Correlation between min payment amount and max spent in single shopping is -0.01

As from above we have done the correlation inbetween all the 7 variables.

Now need to check the outliers



As from the above table we can see that we have 2 outliers

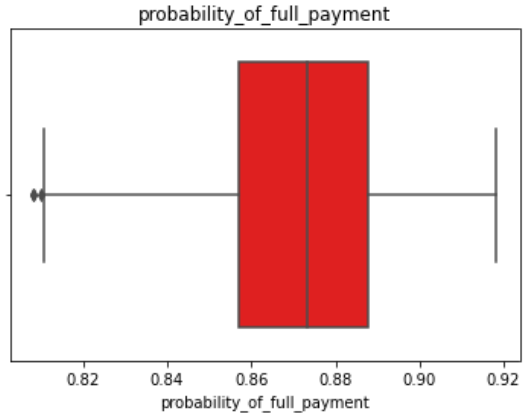
1) probability_of_full_payment

2) min_payment_amt

Observation

Most of the outlier has been treated and now we are good to go.

```
<AxesSubplot:title={'center':'probability_of_full_payment'}, xlabel='probability_of_full_payment'>
```



Though we did treated the outlier, we still see one as per the boxplot, it is okay, as it is no extreme and on lower band.

Question 1.2

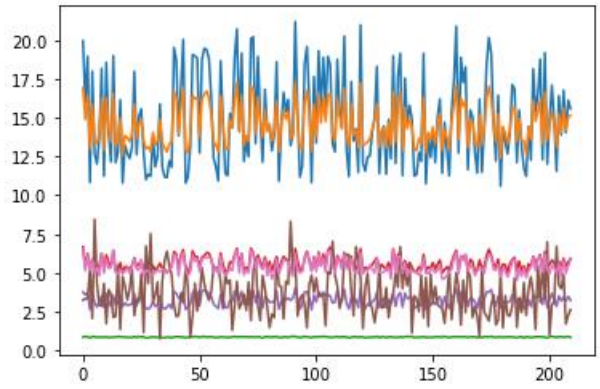
Do you think scaling is necessary for clustering in this case? Justify

Answer:

Yes. Clustering algorithms such as K-means do need feature scaling before they are fed to the algo.

scaling needs to be done as the values of the variables are different after scaling we will have all the values in the relative same range. I have used zscore to standarised the data to relative same scale -3 to +3.

As we need to see the data prior to scaling



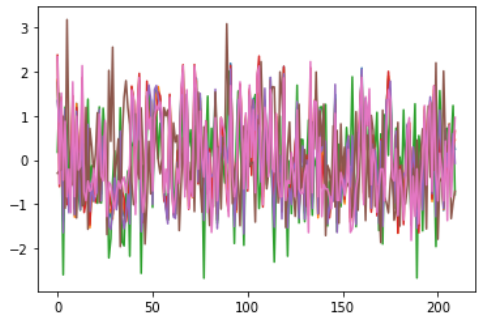
Before Scaling

And after applying the Zscore what value we got in dataset.

```
data=df.apply(zscore)
data.head()
```

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
0	1.754355	1.811968	0.178230	2.367533	1.338579	-0.298806	2.328998
1	0.393582	0.253840	1.501773	-0.600744	0.858236	-0.242805	-0.538582
2	1.413300	1.428192	0.504874	1.401485	1.317348	-0.221471	1.509107
3	-1.384034	-1.227533	-2.591878	-0.793049	-1.639017	0.987884	-0.454961
4	1.082581	0.998364	1.196340	0.591544	1.155464	-1.088154	0.874813

```
plt.plot(data)
plt.show()
```



After scaling

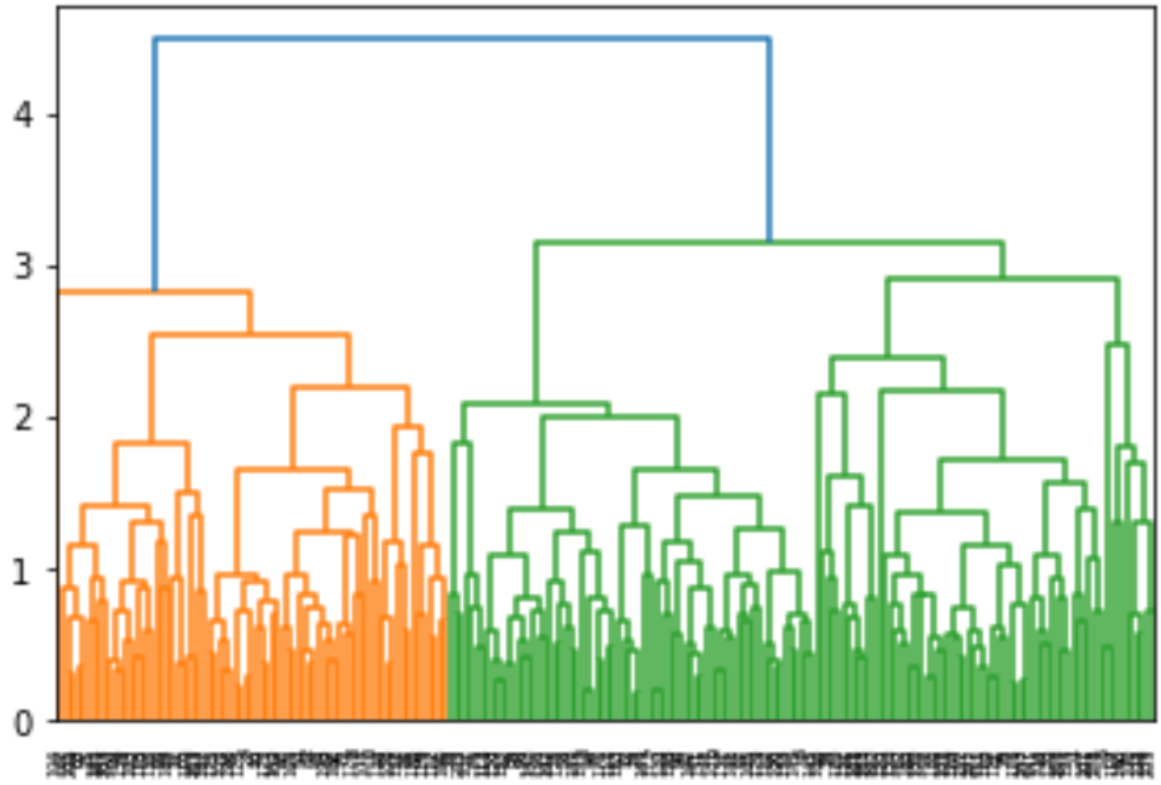
Question 1.3

Apply hierarchical clustering to scaled data. Identify the number of optimum clusters using Dendrogram and briefly describe them.

Answer:

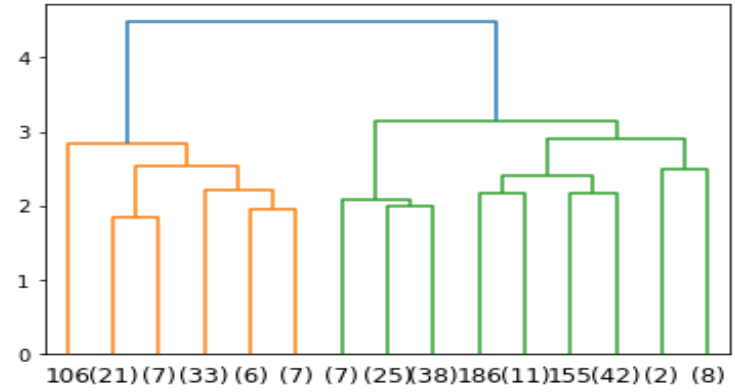
After scaling the data we need to do hierarchical clustering we need to import the libraries for it.

Dendogram:



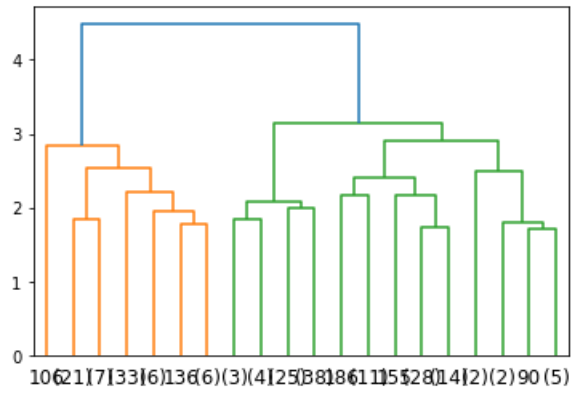
After getting the result of dendrogram we need to get it more clear .

```
dend = dendrogram(link_method,truncate_mode='lastp',p=15)
```

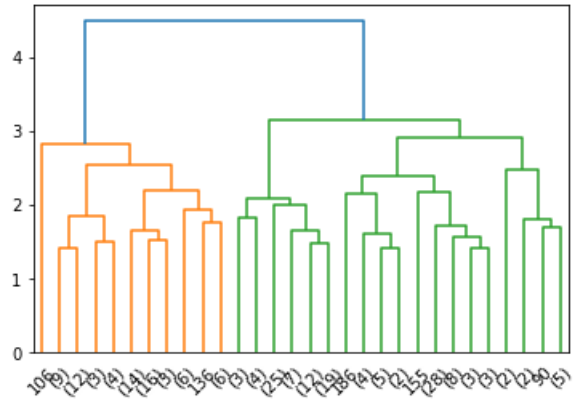


As we can take the P value 30,20,or 25 so we can see the result.

```
dend = dendrogram(link_method,truncate_mode='lastp',p=20)
```



```
dend = dendrogram(link_method,truncate_mode='lastp',p=30)
```



After clustering the value we get

```
array([1, 3, 1, 2, 1, 3, 2, 2, 1, 2, 1, 1, 2, 1, 3, 3, 3, 2, 2, 2, 2, 2,
       1, 2, 3, 1, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 1, 1, 3, 1, 1,
       2, 2, 3, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 3, 2, 2, 1, 3, 1,
       1, 3, 1, 2, 3, 2, 1, 1, 2, 1, 3, 2, 1, 3, 3, 3, 3, 1, 2, 1, 1, 1,
       1, 3, 3, 1, 3, 2, 2, 1, 1, 1, 2, 1, 3, 1, 3, 1, 3, 1, 1, 2, 3, 1,
       1, 3, 1, 2, 2, 1, 3, 3, 2, 1, 3, 2, 2, 2, 3, 3, 1, 2, 3, 3, 2, 3,
       3, 1, 2, 1, 1, 2, 1, 3, 3, 3, 2, 2, 2, 2, 1, 2, 3, 2, 3, 2, 3, 1,
       3, 3, 2, 2, 3, 1, 1, 2, 1, 1, 1, 2, 1, 3, 3, 2, 3, 2, 3, 1, 1, 1,
       3, 2, 3, 2, 3, 2, 3, 3, 1, 1, 3, 1, 3, 2, 3, 3, 2, 1, 3, 1, 1, 2,
       1, 2, 3, 3, 3, 2, 1, 3, 1, 3, 3, 1], dtype=int32)
```

After clustering the data we get

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	clusters-3
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.550	1
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144	3
2	18.95	16.42	0.8829	6.248	3.755	3.368	6.148	1
3	10.83	12.96	0.8099	5.278	2.641	5.182	5.185	2
4	17.99	15.86	0.8992	5.890	3.694	2.068	5.837	1

After aggdata we get the value

```
aggdata
```

clusters-3	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Freq
1	18.129200	16.058000	0.881595	6.135747	3.648120	3.650200	5.987040	75
2	11.916857	13.291000	0.846766	5.258300	2.846000	4.619000	5.115071	70
3	14.217077	14.195846	0.884869	5.442000	3.253508	2.768418	5.055569	65

As we can see from the above table that for value 1 there are a total 75 number of count for 2 we have total 70 number of count and for 3 we have 65 number of count.

Both the method are almost similer means , minor variation, which we know it occurs.

We for cluster grouping based on the dendrogram, 3 or 4 looks good. Did the further analysis, and based on the dataset had gone for 3 group cluster solution based on the hierarchical clustering

Also in real time, there colud have been more variables value captured - tenure, BALANCE_FREQUENCY, balance, purchase, installment of purchase, others.

And three group cluster solution gives a pattern based on high/medium/low spending with max_spent_in_single_shopping (high value item) and probability_of_full_payment(payment made).

Question 1.4

Apply K-Means clustering on scaled data and determine optimum clusters. Apply elbow curve and silhouette score?

Answer:

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, serving as a prototype of the cluster.

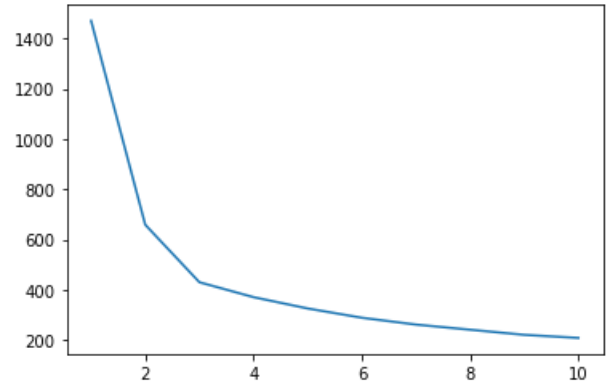
For Cluster 1 we get the value: 1469.9999999999995
For Cluster 2 we get the value: 659.1717544870411
For Cluster 3 we get the value: 430.65897315130064
For Cluater 4 we get the value: 371.6531439995162

For cluster within cluster Sum of Square (WSS)

```
wss
```

```
[1469.9999999999995,
 659.1717544870411,
 430.65897315130064,
 371.301721277542,
 326.36254154106956,
 289.46717056412893,
 262.3445504777467,
 241.92009469832294,
 221.569390877319,
 209.28295981087177]
```

```
[<matplotlib.lines.Line2D at 0x197ab32a400>]
```



After that we get the variable clus_kmeans in dataset.

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Clus_kmeans
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.550	2
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144	0
2	18.95	16.42	0.8829	6.248	3.755	3.368	6.148	2
3	10.83	12.96	0.8099	5.278	2.641	5.182	5.185	1
4	17.99	15.86	0.8992	5.890	3.694	2.068	5.837	2

Now we get the Silhouette sample and Silhouette score

Silhouette score: `0.3347542296283262`
Silhouette sample:

```
silhouette_samples(data,labels_4)
```

```
array([ 4.72459898e-01,  5.36652637e-02,  4.94127371e-01,  5.33637257e-01,
        1.52686050e-01,  2.34941508e-01,  4.72104986e-01,  4.35787998e-01,
        2.65426327e-01,  5.09484040e-01,  3.11333104e-01,  4.40049096e-02,
        3.80469250e-01,  3.84889924e-01,  2.69597441e-01,  1.83848163e-01,
        1.29400208e-01,  4.71941919e-01,  5.86913951e-03,  2.01039060e-01,
        2.95623878e-01,  3.95938728e-01,  2.24937886e-01,  2.23852949e-01,
        1.21410307e-01,  4.51665132e-01,  3.93832703e-01,  5.33898611e-01,
        5.61772416e-01,  4.46650270e-01,  4.40534970e-01,  3.72882000e-01,
        3.73628032e-01,  2.26351415e-01,  5.52414481e-01,  5.84695490e-01,
        5.14887114e-01,  4.22719874e-01,  5.34279492e-01,  5.19629801e-01,
        4.15379157e-01,  4.69103841e-01,  1.70669590e-01,  5.28858661e-01,
        5.37382822e-01,  2.86760740e-01,  3.92074967e-01,  4.82260696e-01,
        2.70382701e-01,  3.67881765e-01,  2.80636751e-01,  1.98019629e-01,
        4.20707965e-01,  4.22315570e-01,  4.67085507e-01,  3.50988316e-01,
        4.39328038e-01,  5.56452647e-01,  5.87599144e-01,  2.83467459e-01,
        3.31999104e-01,  4.91023786e-01,  5.43435665e-01,  2.63352076e-01,
        3.78589860e-01,  4.47017132e-01,  4.60812791e-01,  4.48090837e-01,
        5.24720172e-01,  6.86085016e-02,  4.61199284e-01,  2.59948132e-01,
        4.58315291e-01,  4.24793021e-01,  3.05425646e-01,  4.51084448e-01,
        8.52960281e-02,  4.84480774e-01,  2.94294909e-01,  4.15640459e-01,
        1.17927156e-01,  4.10483068e-01,  4.48218942e-01,  1.42359519e-01,
        5.29848612e-01,  1.41785120e-01,  1.22835794e-01,  4.79087086e-01,
        4.75880547e-01,  1.55682164e-01,  1.77514908e-01,  3.88072821e-01,
        1.11271354e-01,  4.41337064e-01,  6.27581568e-03,  2.87353895e-01,
        4.36224321e-01,  4.71592144e-01,  5.74098323e-01, -2.04129136e-02,
        1.26497048e-01,  5.25507095e-01,  7.12852210e-02,  4.29077507e-01,
        2.46265543e-01,  3.51681970e-01,  1.94115949e-01,  2.10699587e-01,
        2.00475094e-02,  3.78472577e-01,  3.22053865e-01,  2.79671585e-01,
        3.64739162e-01,  3.82625447e-01,  5.49990852e-01,  1.96261885e-01,
        4.76890622e-01,  2.12609456e-02,  5.75002325e-01,  4.66283838e-01,
        2.44849410e-01,  5.04868658e-01,  2.83613924e-01,  2.22315288e-01,
        3.54450865e-01,  2.54536095e-01,  2.77625714e-01,  6.02950040e-01,
        4.29404121e-01,  3.87174154e-01,  5.32162450e-01,  1.47706183e-01,
        2.73234039e-01,  3.78970308e-01,  5.68436050e-01,  6.83135844e-02,
        1.62997612e-01,  4.25632589e-01,  1.51906470e-01,  2.70014217e-01,
        2.32154742e-01,  3.58966943e-01,  4.68466061e-01,  2.64382680e-01,
        1.41896386e-01,  4.71948169e-01,  4.12507199e-01,  5.64454540e-01,
        5.05314434e-01,  5.17972825e-01,  1.85480871e-01,  2.65744552e-01,
        2.05069373e-01,  3.52072887e-01, -1.04648258e-04,  3.61368632e-01,
        1.36413103e-01,  6.06816819e-01,  1.81249117e-01,  4.40925023e-02,
        4.37963053e-01,  5.42559055e-02,  3.83046379e-01,  2.26274167e-01,
        3.93220065e-01,  5.24775154e-01,  4.44824568e-01,  1.15020048e-01,
        3.99333641e-01,  2.53205470e-01,  6.29320748e-02,  5.82801315e-01,
        4.46286018e-01,  3.11038693e-01,  4.53789139e-01,  4.44186912e-01,
        1.30624241e-01,  3.82035801e-02,  2.10641318e-01,  3.95216844e-01,
        3.45624196e-01,  1.23745350e-01,  3.95333954e-01,  3.47977951e-01,
        1.79504048e-01,  3.05943808e-01,  6.03931547e-02,  1.45516125e-01,
        4.63857798e-01,  4.69747209e-01,  1.53931691e-01,  9.70793807e-02,
        5.24056679e-01,  1.54032358e-01,  2.56852026e-01,  3.28443497e-01,
        4.60516268e-01,  3.87172496e-01,  4.62421868e-01,  5.07257731e-01,
        3.88204647e-01,  1.28932501e-01,  8.64172398e-02,  5.30780543e-01,
        4.41072882e-01,  3.60573116e-01,  3.75602469e-01,  4.91146028e-01,
        7.14686264e-02,  3.20045513e-01])
```

Silhouette sample minimum: `-0.020412913640457993`
Silhouette sample maximum: `0.60681681919254`

the optimal cluster should be 3 or 4

Sil width value in the data

spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Clus_kmeans	sil_width
19.94	16.92	0.8752	6.675	3.763	3.252	6.550	2	0.472460
15.99	14.89	0.9064	5.363	3.582	3.336	5.144	0	0.053665
18.95	16.42	0.8829	6.248	3.755	3.368	6.148	2	0.494127
10.83	12.96	0.8099	5.278	2.641	5.182	5.185	1	0.533637
17.99	15.86	0.8992	5.890	3.694	2.068	5.837	2	0.152686

For getting the Kmeans mean cluster-3

```
array([1, 0, 1, 2, 1, 2, 2, 0, 1, 2, 1, 0, 2, 1, 0, 2, 0, 2, 2, 2, 2, 2,
       1, 2, 0, 1, 0, 2, 2, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 1, 1, 0, 1, 1,
       2, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 0, 2, 2, 0, 0, 1,
       1, 0, 1, 2, 0, 2, 1, 1, 2, 1, 0, 2, 1, 0, 0, 0, 0, 1, 2, 0, 1, 0,
       1, 2, 0, 1, 0, 2, 2, 1, 1, 1, 2, 1, 0, 1, 0, 1, 0, 1, 1, 2, 2, 1,
       0, 0, 1, 2, 2, 1, 0, 0, 2, 1, 0, 2, 2, 2, 0, 0, 1, 2, 0, 0, 2, 0,
       0, 1, 2, 1, 1, 2, 1, 0, 0, 0, 2, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 0,
       2, 0, 0, 2, 0, 1, 1, 2, 1, 1, 1, 2, 0, 0, 0, 2, 0, 2, 0, 1, 1, 1,
       0, 2, 0, 2, 0, 0, 0, 0, 1, 1, 2, 0, 0, 2, 2, 0, 2, 1, 0, 1, 1, 2,
       1, 2, 0, 1, 0, 2, 1, 0, 1, 0, 0, 0])
```

cluster	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
1	14.4	14.3	0.9	5.5	3.3	2.7	5.1
2	11.9	13.2	0.8	5.2	2.8	4.7	5.1
3	18.5	16.2	0.9	6.2	3.7	3.6	6.0

cluster	1	2	3
spending	14.4	11.9	18.5
advance_payments	14.3	13.2	16.2
probability_of_full_payment	0.9	0.8	0.9
current_balance	5.5	5.2	6.2
credit_limit	3.3	2.8	3.7
min_payment_amt	2.7	4.7	3.6
max_spent_in_single_shopping	5.1	5.1	6.0

Going with 3 clusters via Kmeans but showing the analysis of 4 & 5 Kmeans cluster.

For getting the Kmeans mean cluster-4

```
array([0, 3, 0, 1, 0, 1, 1, 3, 0, 1, 0, 2, 1, 0, 3, 1, 3, 1, 3, 1, 1, 1,
       0, 1, 3, 2, 3, 1, 1, 1, 3, 1, 1, 3, 1, 1, 1, 1, 1, 0, 0, 3, 2, 0,
       1, 1, 3, 0, 0, 0, 1, 0, 0, 0, 0, 2, 1, 1, 1, 0, 3, 1, 1, 2, 3, 0,
       0, 3, 0, 3, 3, 1, 0, 0, 1, 0, 3, 1, 2, 3, 3, 3, 3, 0, 1, 2, 2, 2,
       2, 1, 3, 0, 3, 1, 3, 0, 0, 2, 1, 0, 3, 0, 2, 0, 3, 0, 0, 1, 3, 0,
       2, 3, 0, 1, 1, 2, 3, 2, 1, 0, 3, 1, 1, 1, 3, 3, 0, 1, 3, 3, 1, 3,
       3, 0, 1, 0, 0, 1, 2, 3, 2, 3, 1, 1, 3, 1, 0, 1, 3, 1, 3, 1, 3, 2,
       3, 3, 3, 1, 3, 0, 0, 1, 0, 2, 0, 1, 2, 3, 3, 1, 3, 1, 3, 0, 0, 0,
       3, 3, 2, 1, 3, 3, 3, 3, 2, 2, 3, 2, 3, 1, 3, 3, 1, 0, 3, 2, 0, 1,
       0, 1, 3, 2, 3, 1, 2, 3, 2, 3, 2, 2])
```

cluster	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
1	19.1	16.4	0.9	6.3	3.8	3.5	6.1
2	11.9	13.3	0.8	5.2	2.8	4.9	5.1
3	16.1	15.2	0.9	5.8	3.4	4.0	5.6
4	14.1	14.1	0.9	5.4	3.2	2.4	5.0

```
cluster_4_T = kmeans_mean_cluster.T
cluster_4_T
```

cluster	1	2	3	4
spending	19.1	11.9	16.1	14.1
advance_payments	16.4	13.3	15.2	14.1
probability_of_full_payment	0.9	0.8	0.9	0.9
current_balance	6.3	5.2	5.8	5.4
credit_limit	3.8	2.8	3.4	3.2
min_payment_amt	3.5	4.9	4.0	2.4
max_spent_in_single_shopping	6.1	5.1	5.6	5.0

For getting the Kmeans mean cluster-4

```
array([1, 0, 1, 4, 1, 2, 2, 0, 1, 2, 1, 2, 2, 1, 2, 2, 0, 2, 2, 2, 2, 4,
       1, 2, 0, 3, 0, 4, 4, 4, 0, 2, 2, 0, 4, 4, 4, 2, 4, 1, 1, 0, 3, 1,
       4, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 3, 2, 4, 4, 1, 0, 2, 4, 3, 0, 1,
       1, 0, 1, 2, 0, 2, 1, 1, 2, 1, 0, 4, 3, 0, 0, 0, 0, 1, 4, 3, 3, 3,
       3, 2, 0, 1, 0, 4, 2, 1, 1, 3, 4, 3, 2, 1, 0, 1, 0, 1, 1, 4, 2, 1,
       3, 0, 1, 4, 4, 3, 0, 2, 4, 1, 2, 4, 2, 2, 0, 0, 1, 4, 0, 0, 4, 0,
       2, 1, 4, 3, 1, 2, 3, 0, 3, 0, 4, 2, 2, 2, 1, 4, 0, 4, 0, 2, 0, 3,
       2, 0, 2, 4, 0, 3, 1, 2, 1, 3, 1, 2, 3, 0, 0, 2, 0, 4, 0, 1, 1, 1,
       0, 2, 3, 2, 0, 2, 0, 0, 3, 3, 2, 3, 0, 4, 2, 0, 4, 1, 0, 3, 1, 2,
       1, 4, 0, 3, 0, 4, 3, 0, 3, 0, 0, 3])
```

cluster	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
1	11.6	13.2	0.8	5.3	2.8	4.7	5.2
2	19.1	16.4	0.9	6.3	3.8	3.5	6.1
3	14.2	14.2	0.9	5.5	3.2	2.3	5.0
4	12.3	13.3	0.9	5.2	3.0	5.0	5.0
5	16.2	15.2	0.9	5.9	3.4	3.9	5.7

```
cluster_5_T = kmeans_mean_cluster.T
cluster_5_T
```

cluster	1	2	3	4	5
spending	11.6	19.1	14.2	12.3	16.2
advance_payments	13.2	16.4	14.2	13.3	15.2
probability_of_full_payment	0.8	0.9	0.9	0.9	0.9
current_balance	5.3	6.3	5.5	5.2	5.9
credit_limit	2.8	3.8	3.2	3.0	3.4
min_payment_amt	4.7	3.5	2.3	5.0	3.9
max_spent_in_single_shopping	5.2	6.1	5.0	5.0	5.7

Question 1.5

Describe cluster profiles for the clusters defined. Recommend different promotional strategies for different clusters?

Answer:

3 groups clusters via kmeans

cluster	1	2	3
spending	14.4	11.9	18.5
advance_payments	14.3	13.2	16.2
probability_of_full_payment	0.9	0.8	0.9
current_balance	5.5	5.2	6.2
credit_limit	3.3	2.8	3.7
min_payment_amt	2.7	4.7	3.6
max_spent_in_single_shopping	5.1	5.1	6.0

3 groups cluster via hierarchical clustering

clusters-3	1	2	3
spending	18.129200	11.916857	14.217077
advance_payments	16.058000	13.291000	14.195846
probability_of_full_payment	0.881595	0.846766	0.884869
current_balance	6.135747	5.258300	5.442000
credit_limit	3.648120	2.846000	3.253508
min_payment_amt	3.650200	4.619000	2.768418
max_spent_in_single_shopping	5.987040	5.115071	5.055569
Freq	75.000000	70.000000	65.000000

Cluster Groups Profiles

Group 1 - High Spending

- Giving any reward points might increase their purchases.
- maximum max_spent_in_single_shopping is high for this group, so can be o ffered discount/offer on next transactions upon full payment. Increase ther e credit limit and Increase spending habits.
- Give loan against the credit card, as they are customers with good repayme nt record.
- Tie up with luxary brands, which will drive more one_time_maximun spendin g

Group 3 - Medium Spending

- They are potential target customer who are paying bills and doing purchases and maintaining comparatively good credit score. So we can increase credit limit or can lower down interest rate.
- Promote premium cards loyalty cars to increase transcations.
- Increase spending habits by trying with premium ecommerce sites ,travel portal, travel airlines/hotels as this will encourage them to spend more .

Group 2 - Low Spending

- Customers should be given remainders for payments .offers can be provided on early payments to improve their payment rate.
- Increase there spending habits by tieing up with grocery store utilities.

Problem 2 -
CART-RF-ANN

An Insurance firm providing tour insurance is facing higher claim frequency. The management decides to collect data from the past few years. You are assigned the task to make a model which predicts the claim status and provide recommendations to management. Use CART, RF & ANN and compare the models' performances in train and test sets.

Attribute Information:

1. Target: Claim Status (Claimed)
2. Code of tour firm (Agency_Code)
3. Type of tour insurance firms (Type)
4. Distribution channel of tour insurance agencies (Channel)
5. Name of the tour insurance products (Product)
6. Duration of the tour (Duration in days)
7. Destination of the tour (Destination)
8. Amount worth of sales per customer in procuring tour insurance policies in rupees (in 100’s)
9. The commission received for tour insurance firm (Commission is in percentage of sales)
10. Age of insured (Age)

Importing the important libraries.

Importing the csv file.

Checked the dataset:

	Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
0	48	C2B	Airlines	No	0.70	Online	7	2.51	Customised Plan	ASIA
1	36	EPX	Travel Agency	No	0.00	Online	34	20.00	Customised Plan	ASIA
2	39	CWT	Travel Agency	No	5.94	Online	3	9.90	Customised Plan	Americas
3	36	EPX	Travel Agency	No	0.00	Online	4	26.00	Cancellation Plan	ASIA
4	33	JZI	Airlines	No	6.30	Online	53	18.00	Bronze Plan	ASIA

We found that the data set has 3000 rows and 10 columns.

```
harish.shape
```

```
(3000, 10)
```

The description of dataset

```
harish.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	3000.0	38.091000	10.463518	8.0	32.0	36.00	42.000	84.00
Commision	3000.0	14.529203	25.481455	0.0	0.0	4.63	17.235	210.21
Duration	3000.0	70.001333	134.053313	-1.0	11.0	26.50	63.000	4580.00
Sales	3000.0	60.249913	70.733954	0.0	20.0	33.00	69.000	539.00

Checking for duplicates in dataset

```
harish.duplicated().sum()
```

```
139
```

As we can see there are 139 duplicates value .

Checking for null values

```
harish.isnull().sum()
```

```
Age             0
Agency_Code     0
Type            0
Claimed         0
Commision       0
Channel         0
Duration        0
Sales           0
Product Name    0
Destination     0
dtype: int64
```

```
harish.isnull().any()
```

```
Age             False
Agency_Code     False
Type            False
Claimed         False
Commision       False
Channel         False
Duration        False
Sales           False
Product Name    False
Destination     False
dtype: bool
```

As checked we don’t have any null values .

Checked info :

```
harish.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Age           3000 non-null   int64  
 1   Agency_Code   3000 non-null   object 
 2   Type          3000 non-null   object 
 3   Claimed       3000 non-null   object 
 4   Commision     3000 non-null   float64
 5   Channel       3000 non-null   object 
 6   Duration      3000 non-null   int64  
 7   Sales         3000 non-null   float64
 8   Product Name  3000 non-null   object 
 9   Destination   3000 non-null   object 
dtypes: float64(2), int64(2), object(6)
memory usage: 234.5+ KB
```

Observations:

10 variables along with 3000 observations

Age, Commision, Duration, Sales are numeric variable
Agency_code, Type, Claimed, Channel, Product Names & Destination,
No missing record
139 Duplicate record

Question 2.1

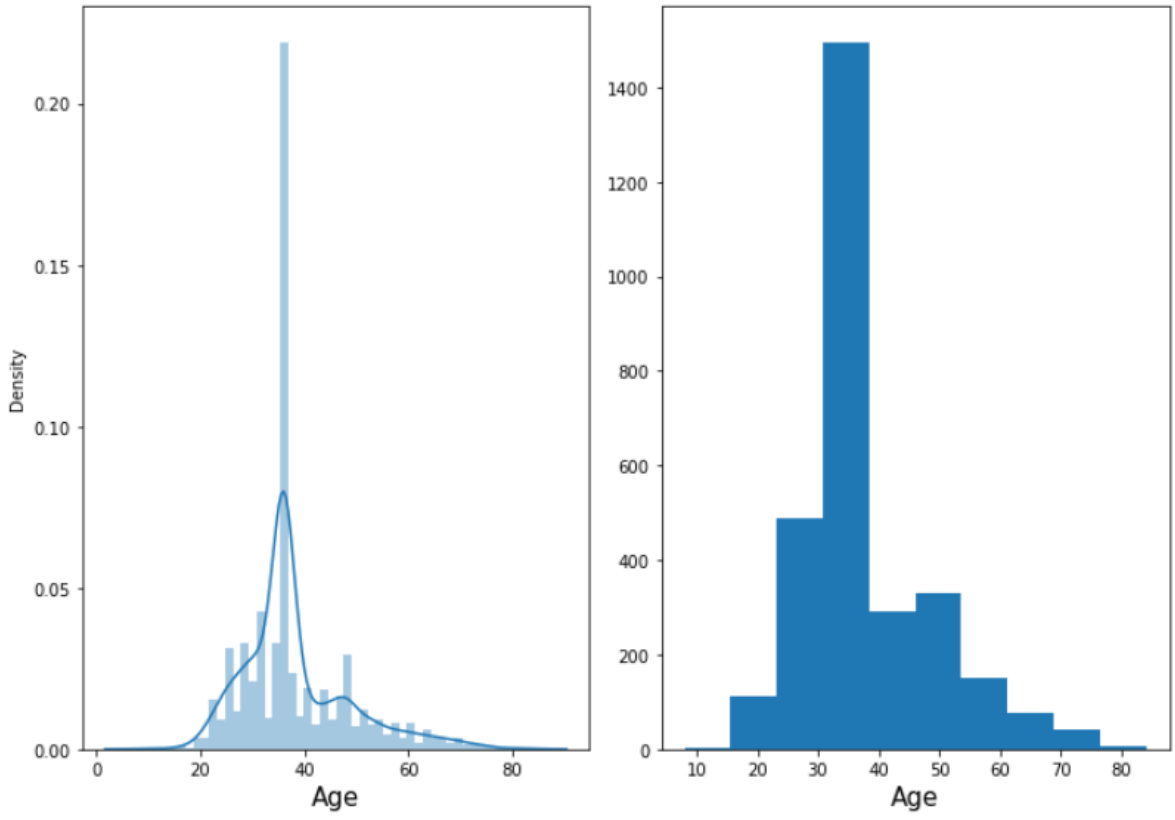
Read the data, do the necessary initial steps, and exploratory data analysis (Univariate, Bi-variate, and multivariate analysis).

Answer:

We will get min, max, median, mean, Std deviation for all variables in data set with that we also take 1st quartile,3rd quartile, inter quartile range(IQR),Lower outliers, upper outliers and distplot and histplot for all variables

Age variable:

```
Minimum Age:  8
Maximum Age:  84
Mean value:  38.091
Median value:  36.0
Standard deviation:  10.463518245377944

1st Quartile:  32.0
3st Quartile:  42.0
IQR 10.0

Lower outliers:  17.0
Upper outliers:  57.0
```

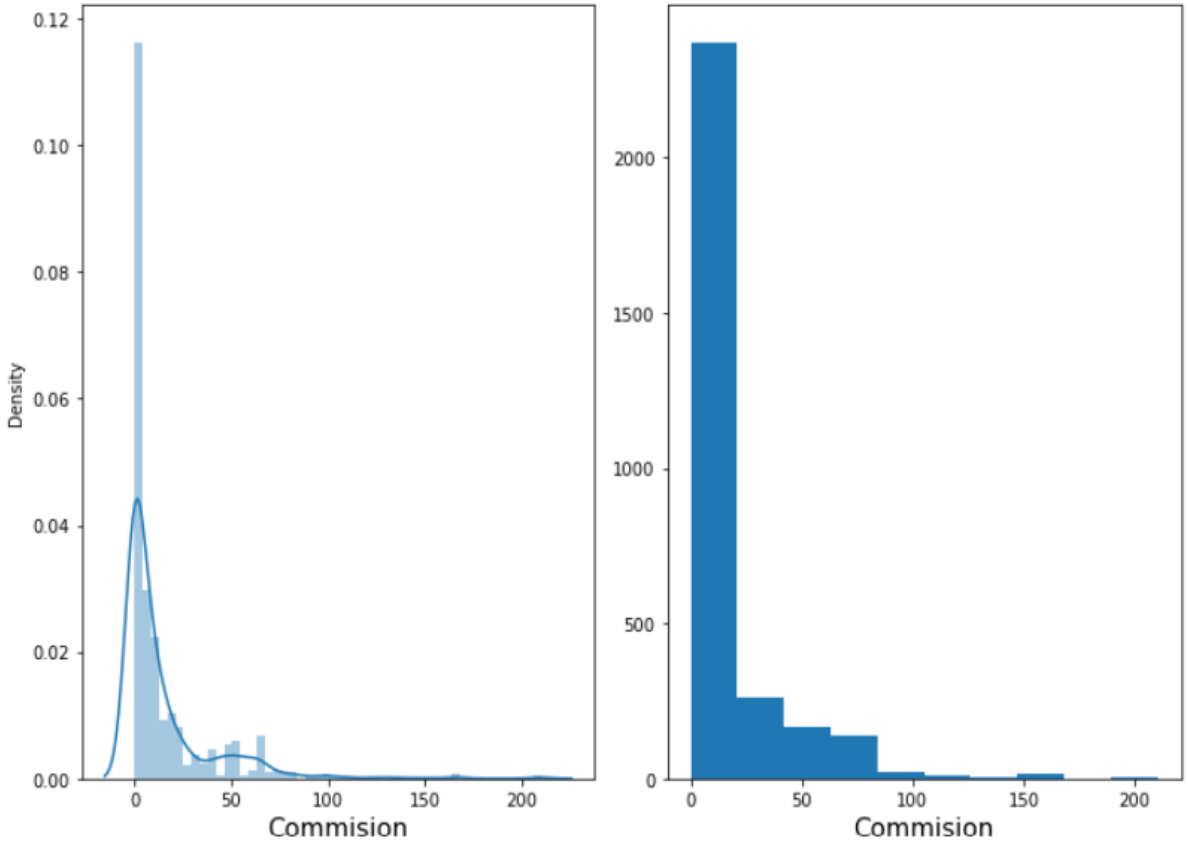


Commision Variable:

```
Minimum Commision:  0.0
Maximum Commision:  210.21
Mean Commision:  14.529203333333266
Median Commision:  4.63
Standard deviation:  25.48145450662553
```

```
1st Quartile is: 0.0
3rd Quartile is: 17.235
IQR 17.235
```

```
Lower outliers:  -25.8525
Upper outliers:  43.0875
```

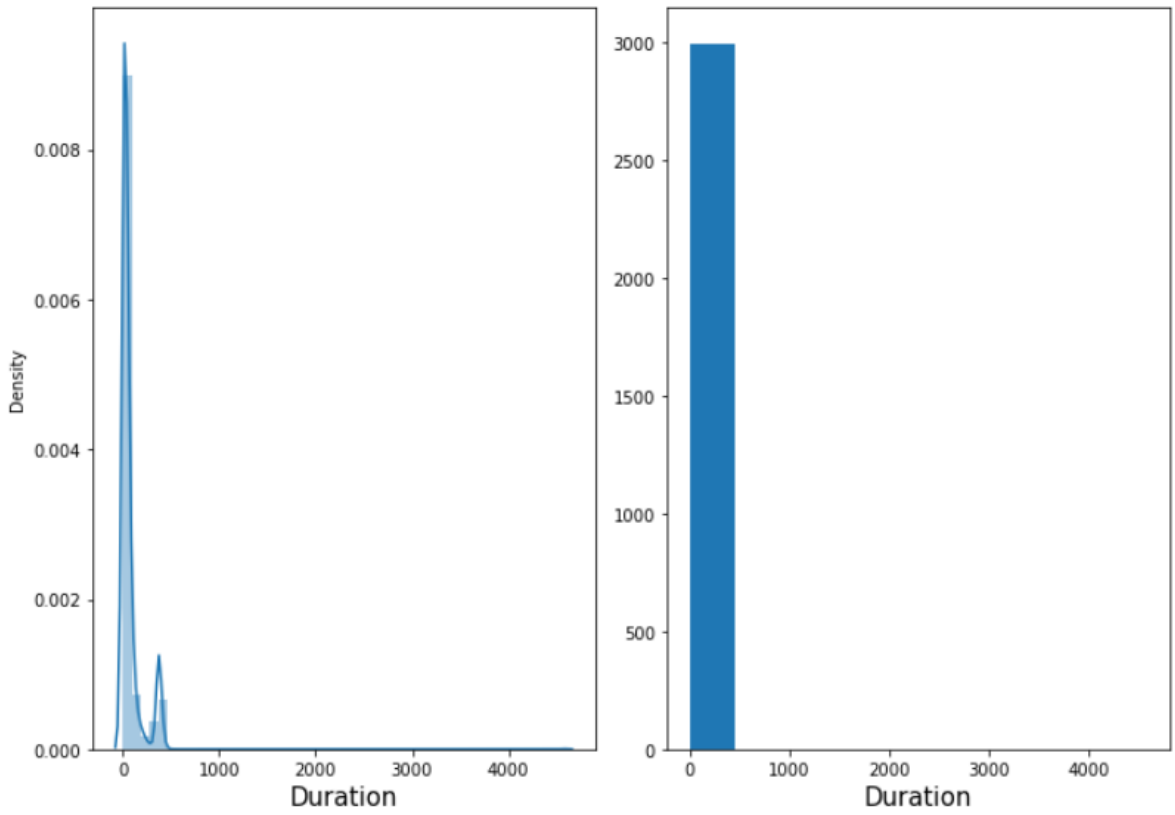


Duration Variable:

```
Minimum Duration:  -1
Maximum Duration:  4580
Mean Duration:  70.00133333333333
Median Duration:  26.5
Standard Duration:  134.05331313253495
```

```
1st Quartile is: 11.0
3rd Quartile is: 63.0
IQR 52.0
```

```
Lower outliers:  -67.0
Upper outliers:  141.0
```



Sales Variables:

Minimum Sales: 0.0

Maximum Sales: 539.0

Mean Sales: 60.24991333333344

Median Sales: 33.0

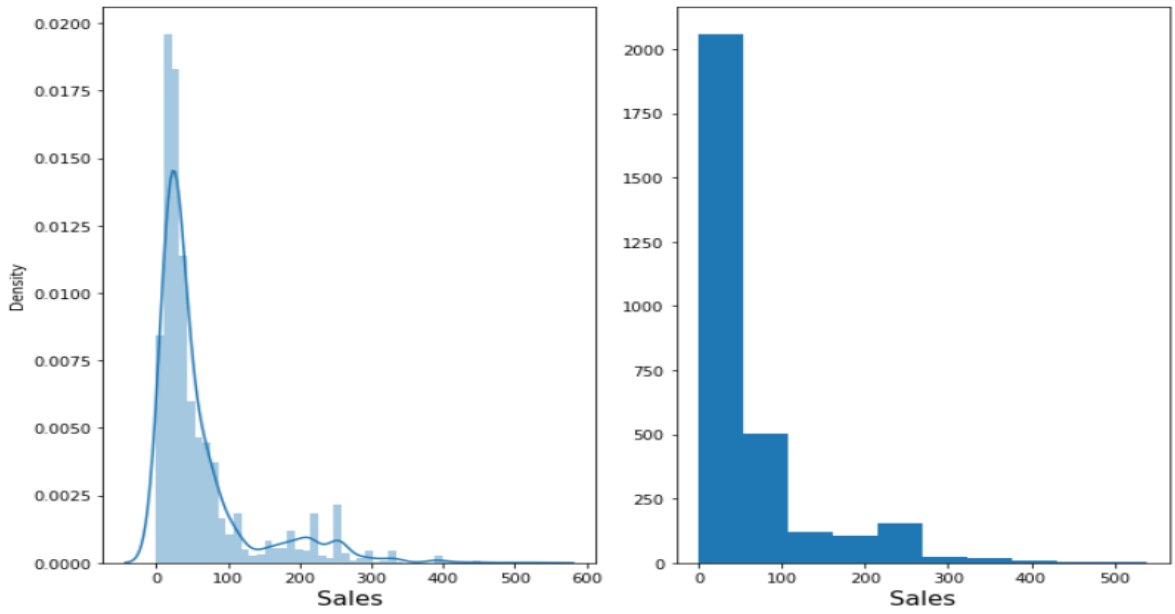
Standard Sales: 70.73395353143047

1st Quartile is: 20.0

3rd Quartile is: 69.0

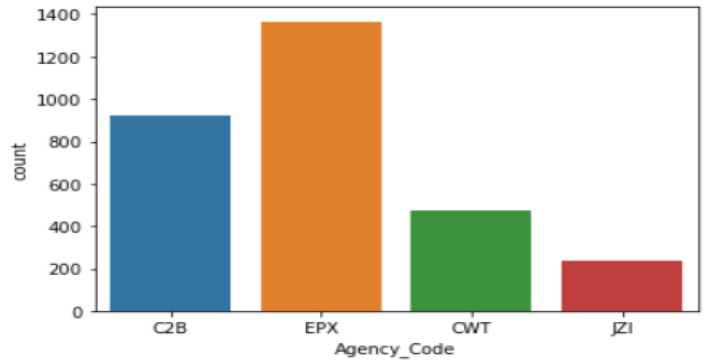
IQR 49.0

```
Lower outliers:  -53.5
Upper outliers:  142.5
```

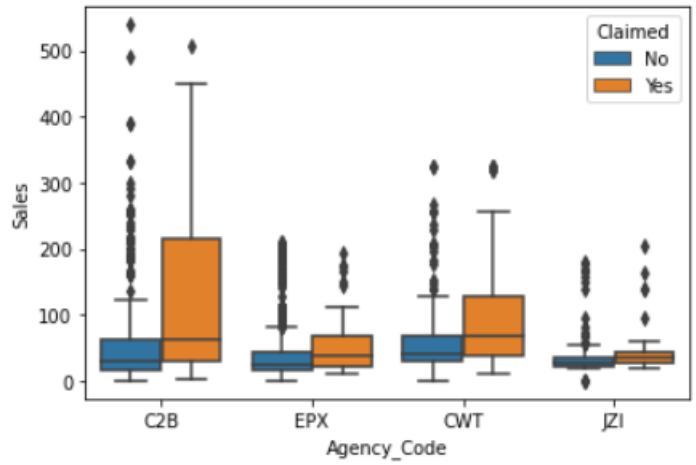


COUNT PLOT and BOX PLOT for agency code

```
<AxesSubplot:xlabel='Agency_Code', ylabel='count'>
```

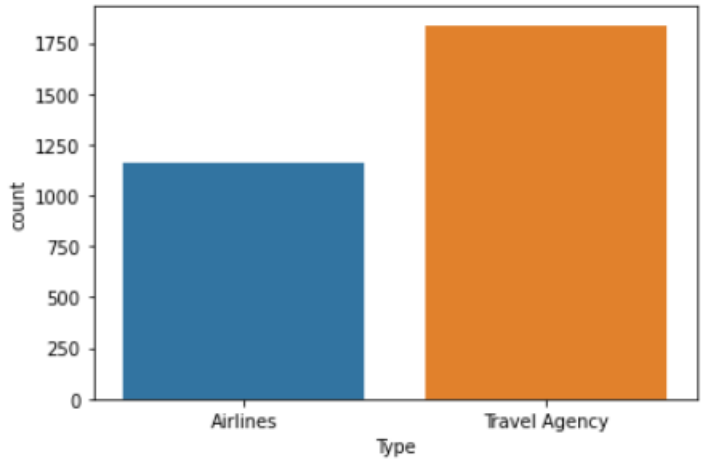



```
<AxesSubplot:xlabel='Agency_Code', ylabel='Sales'>
```



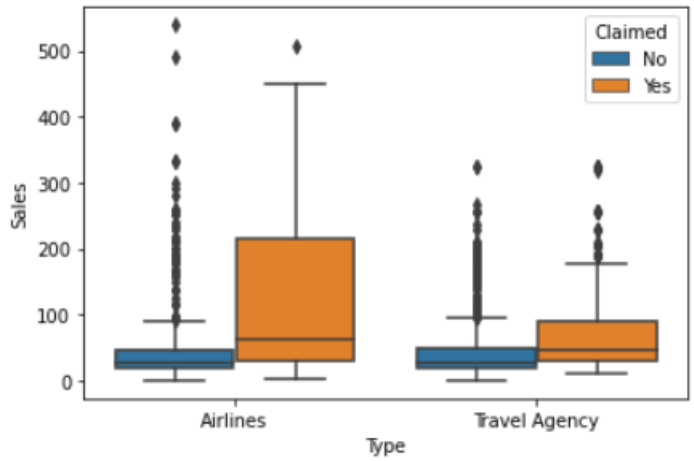
COUNT PLOT and BOX PLOT for type

```
<AxesSubplot:xlabel='Type', ylabel='count'>
```



Count plot and box plot for airlines and travel agency

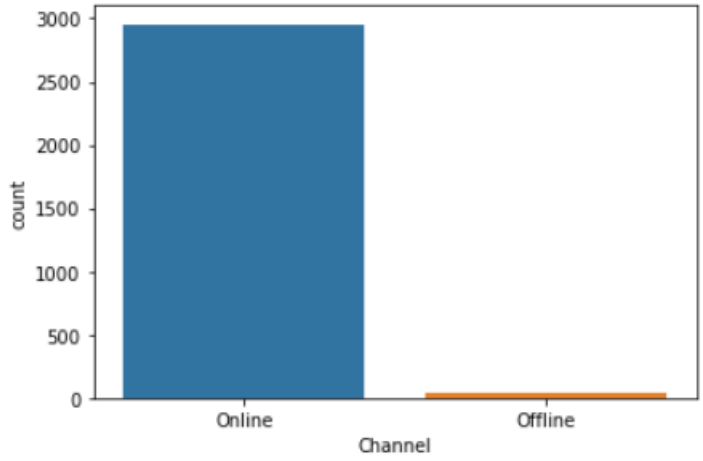
```
<AxesSubplot:xlabel='Type', ylabel='Sales'>
```



Boxplot for checking outliers for airlines and travel agency.

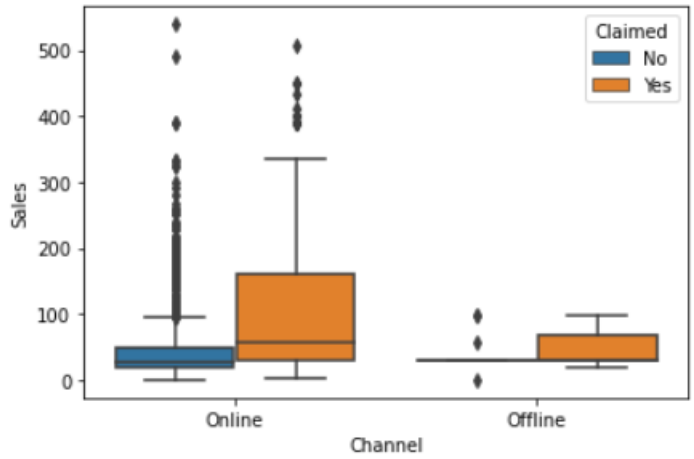
Count plot and box plot for channel

```
<AxesSubplot:xlabel='Channel', ylabel='count'>
```



Count plot for checking online and offline

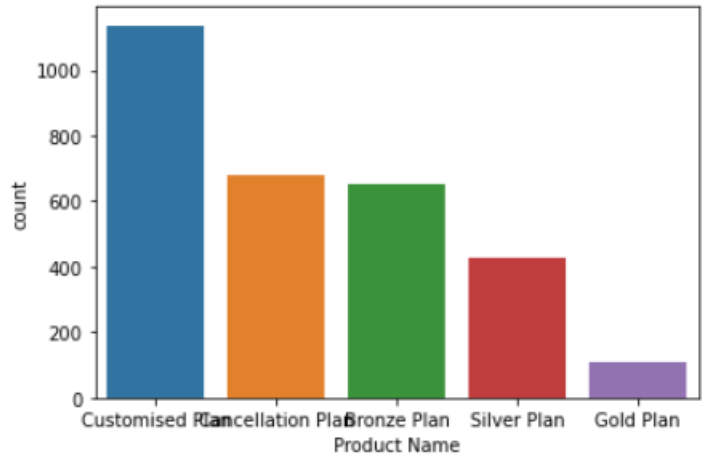
```
<AxesSubplot:xlabel='Channel', ylabel='Sales'>
```



Checking outliers for online and offline

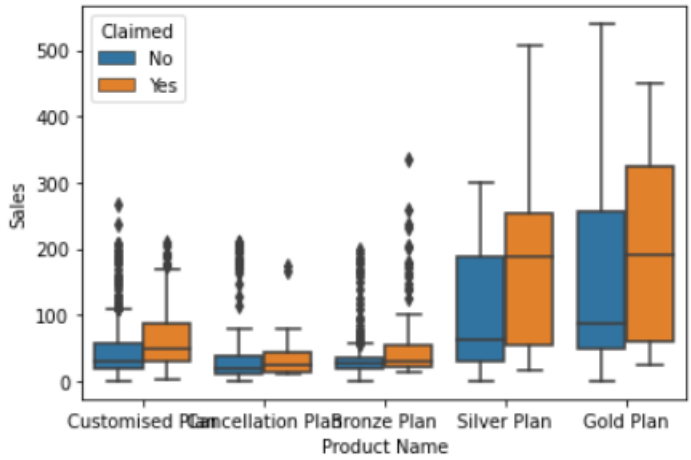
Count plot and boxplot for product name

```
<AxesSubplot:xlabel='Product Name', ylabel='count'>
```



Countplot for product name such as (customised plan, cancellation plan, bronze plan, silver plan, gold plan)

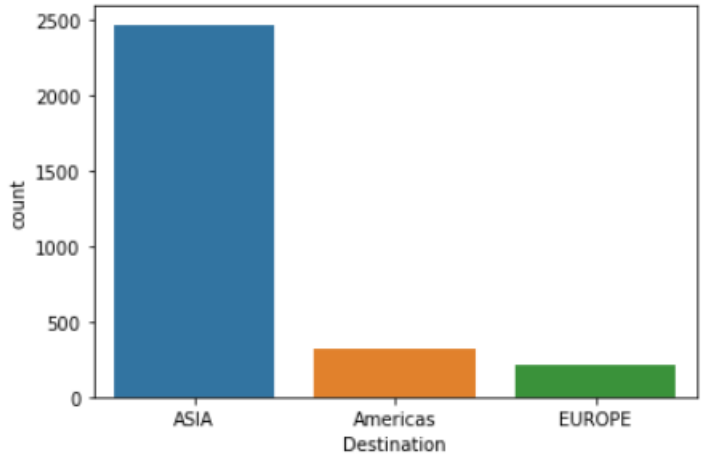
```
<AxesSubplot:xlabel='Product Name', ylabel='Sales'>
```



Box plot for all the outliers in product name.

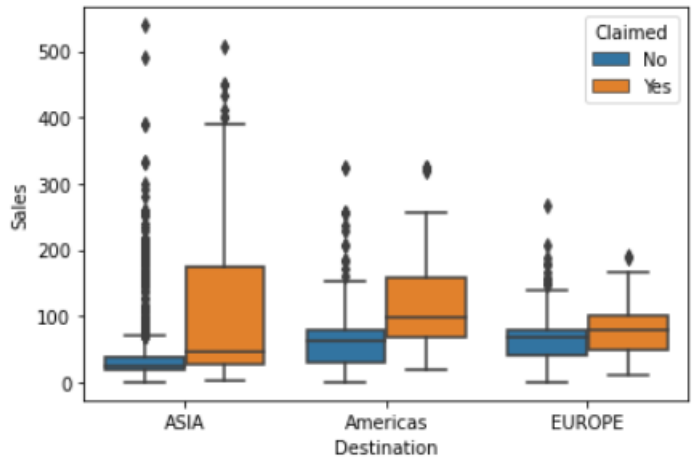
Count plot and box plot for Destination

```
<AxesSubplot:xlabel='Destination', ylabel='count'>
```



Count plot for all the destinations (Asia , Americas Destination , EUROPE)

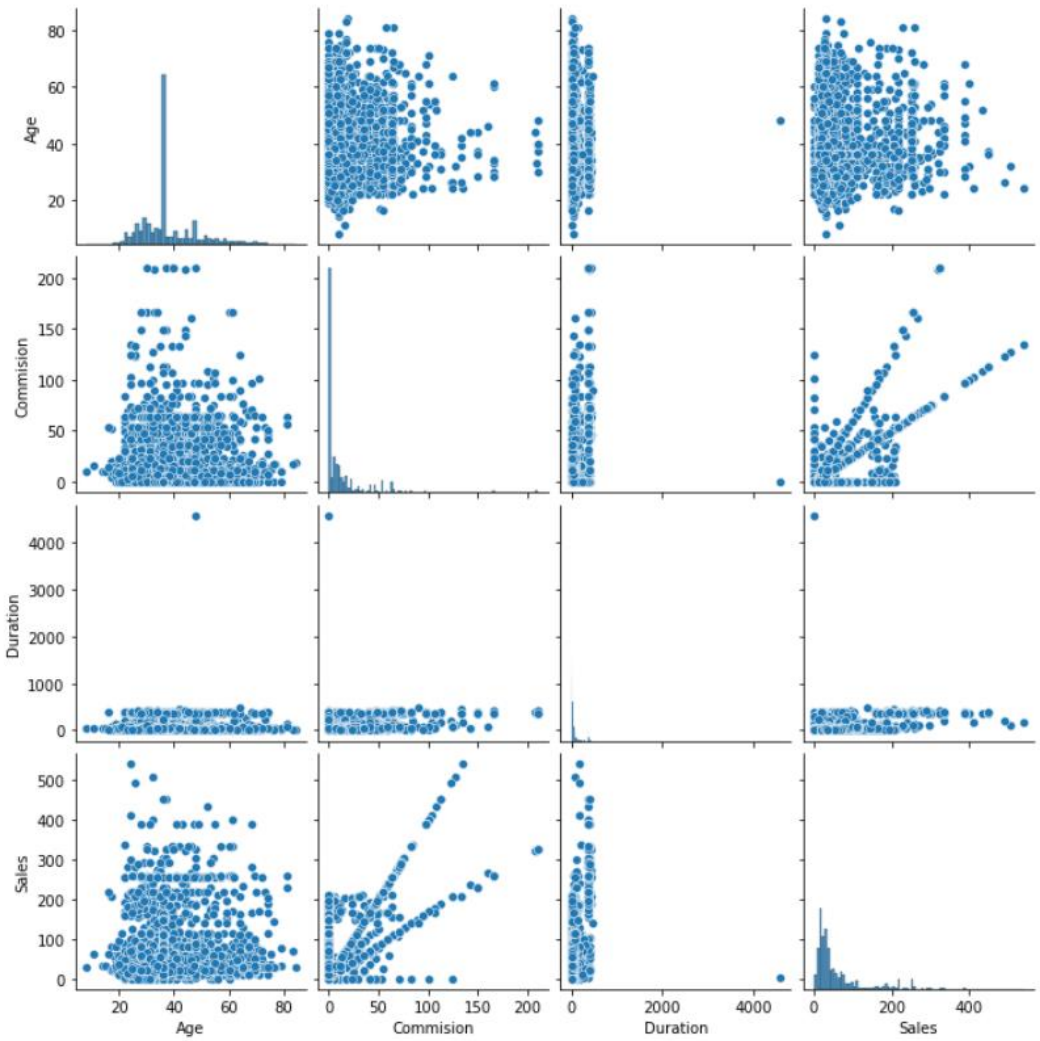
```
<AxesSubplot:xlabel='Destination', ylabel='Sales'>
```



Outliers for given destination (ASIA , Americas Destination, EUROPE)

Checking pairwise distribution of the continuous variable

```
<Figure size 720x576 with 0 Axes>
```



Checking for correlations

```
<AxesSubplot:>
```

As per checking the info from dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Age           3000 non-null   int64  
 1   Agency_Code   3000 non-null   object 
 2   Type          3000 non-null   object 
 3   Claimed       3000 non-null   object 
 4   Commision     3000 non-null   float64
 5   Channel       3000 non-null   object 
 6   Duration      3000 non-null   int64  
 7   Sales         3000 non-null   float64
 8   Product Name  3000 non-null   object 
 9   Destination   3000 non-null   object 
dtypes: float64(2), int64(2), object(6)
memory usage: 234.5+ KB
```

As checked in info we have object datatype

We need to convert the object datatype into categorical codes.

```
for feature in harish.columns:
    if harish[feature].dtype == 'object':

        print('feature:',feature)
        print(pd.Categorical(harish[feature].unique()))
        print(pd.Categorical(harish[feature].unique()).codes)
        harish[feature] = pd.Categorical(harish[feature]).codes
```

```
feature: Agency_Code
['C2B', 'EPX', 'CWT', 'JZI']
Categories (4, object): ['C2B', 'CWT', 'EPX', 'JZI']
[0 2 1 3]
feature: Type
['Airlines', 'Travel Agency']
Categories (2, object): ['Airlines', 'Travel Agency']
[0 1]
feature: Claimed
['No', 'Yes']
Categories (2, object): ['No', 'Yes']
[0 1]
feature: Channel
['Online', 'Offline']
Categories (2, object): ['Offline', 'Online']
[1 0]
feature: Product Name
['Customised Plan', 'Cancellation Plan', 'Bronze Plan', 'Silver Plan', 'Gold Plan']
Categories (5, object): ['Bronze Plan', 'Cancellation Plan', 'Customised Plan', 'Gold Plan', 'Silver Plan']
[2 1 0 4 3]
feature: Destination
['ASIA', 'Americas', 'EUROPE']
Categories (3, object): ['ASIA', 'Americas', 'EUROPE']
[0 1 2]
```

after converting the object to categorical codes. Need to check the dataset and info

check the dataset:

	Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
0	48	0	0	0	0.70	1	7	2.51	2	0
1	36	2	1	0	0.00	1	34	20.00	2	0
2	39	1	1	0	5.94	1	3	9.90	2	1
3	36	2	1	0	0.00	1	4	26.00	1	0
4	33	3	0	0	6.30	1	53	18.00	0	0

Now checking the info of dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Age            3000 non-null   int64  
 1   Agency_Code    3000 non-null   int8   
 2   Type           3000 non-null   int8   
 3   Claimed        3000 non-null   int8   
 4   Commision      3000 non-null   float64
 5   Channel        3000 non-null   int8   
 6   Duration       3000 non-null   int64  
 7   Sales          3000 non-null   float64
 8   Product Name   3000 non-null   int8   
 9   Destination    3000 non-null   int8   
dtypes: float64(2), int64(2), int8(6)
memory usage: 111.5 KB
```

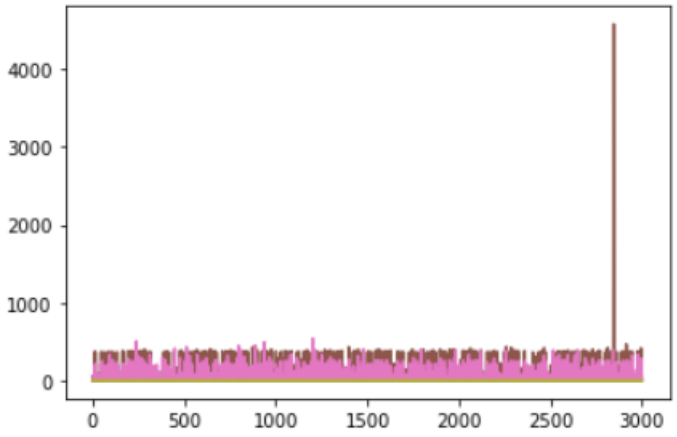
Question 2.2

Data Split: Split the data into test and train, build classification model CART, Random Forest, Artificial Neural Network

Answer:

Extracting the target column into separate vectors for training set and test set

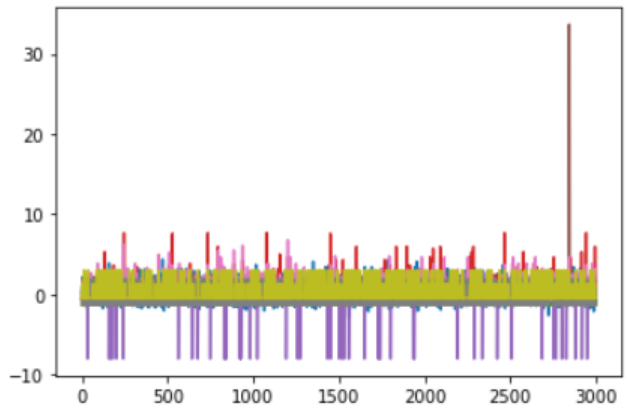
	Age	Agency_Code	Type	Commision	Channel	Duration	Sales	Product Name	Destination
0	48	0	0	0.70	1	7	2.51	2	0
1	36	2	1	0.00	1	34	20.00	2	0
2	39	1	1	5.94	1	3	9.90	2	1
3	36	2	1	0.00	1	4	26.00	1	0
4	33	3	0	6.30	1	53	18.00	0	0



After applying Zscore to the dataset.

	Age	Agency_Code	Type	Commision	Channel	Duration	Sales	Product Name	Destination
0	0.947162	-1.314358	-1.256796	-0.542807	0.124788	-0.470051	-0.816433	0.268835	-0.434646
1	-0.199870	0.697928	0.795674	-0.570282	0.124788	-0.268605	-0.569127	0.268835	-0.434646
2	0.086888	-0.308215	0.795674	-0.337133	0.124788	-0.499894	-0.711940	0.268835	1.303937
3	-0.199870	0.697928	0.795674	-0.570282	0.124788	-0.492433	-0.484288	-0.525751	-0.434646
4	-0.486629	1.704071	-1.256796	-0.323003	0.124788	-0.126846	-0.597407	-1.320338	-0.434646

```
plt.plot(X_scaled)
plt.show()
```



Splitting the data into training and test

```
X_train, X_test, train_labels, test_labels = train_test_split(X_scaled, y, test_size=.30, random_state=5)
```

```
print('X_train',X_train.shape)
print('X_test',X_test.shape)
print('train_labels',train_labels.shape)
print('test_labels',test_labels.shape)
```

```
X_train (2100, 9)
X_test (900, 9)
train_labels (2100,)
test_labels (900,)
```

```
reg_dt_model = {
    'criterion': ['gini'],
    'max_depth': [10,20,30,50],
    'min_samples_leaf': [50,100,150],
    'min_samples_split': [150,300,450],
}

dtcl = DecisionTreeClassifier(random_state=1)

grid_search_dtcl = GridSearchCV(estimator = dtcl, param_grid = reg_dt_model, cv = 10)
```

```
grid_search_dtcl.fit(X_train, train_labels)
print(grid_search_dtcl.best_params_)
best_grid_dtcl = grid_search_dtcl.best_estimator_
best_grid_dtcl
```

```
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 450}

DecisionTreeClassifier(max_depth=10, min_samples_leaf=50, min_samples_split=450,
                       random_state=1)
```

```
train_char_label=['No','Yes']
# after that we need to create a dot file in d,c,e any drive eg.
Credit_Tree_File=open('d:\credit_tree.dot','w')
dot_data=tree.export_graphviz(best_grid_dtcl,out_file=Credit_Tree_File,feature_names=list(X_train),class_names=list(train_char_la
Credit_Tree_File.close()
```

```
# as i have added this file while submitting the project .
# file name -credit_tree
# as we can check this file in webgraphviz.com
```

As the file credit_tree is added while submitting the file.

Predicting on training and testing Dataset

	0	1
0	0.656751	0.343249
1	0.935593	0.064407
2	0.935593	0.064407
3	0.656751	0.343249
4	0.935593	0.064407

building a random forest classifier

```
grid_search_rfcl.fit(X_train, train_labels)
print(grid_search_rfcl.best_params_)
best_grid_rfcl = grid_search_rfcl.best_estimator_
best_grid_rfcl
```

```
{'max_depth': 6, 'max_features': 3, 'min_samples_leaf': 8, 'min_samples_split': 46, 'n_estimators': 350}

RandomForestClassifier(max_depth=6, max_features=3, min_samples_leaf=8,
                       min_samples_split=46, n_estimators=350, random_state=1)
```

predicting training and testing data

	0	1
0	0.778010	0.221990
1	0.971910	0.028090
2	0.904401	0.095599
3	0.651398	0.348602
4	0.868406	0.131594

Artificial Neural Network

```
grid_search_nncl.fit(X_train, train_labels)
grid_search_nncl.best_params_
best_grid_nncl = grid_search_nncl.best_estimator_
best_grid_nncl
```

```
MLPClassifier(hidden_layer_sizes=200, max_iter=2500, random_state=1, tol=0.01)
```

predicting training and testing data

	0	1
0	0.822676	0.177324
1	0.933407	0.066593
2	0.918772	0.081228
3	0.688933	0.311067
4	0.913425	0.086575

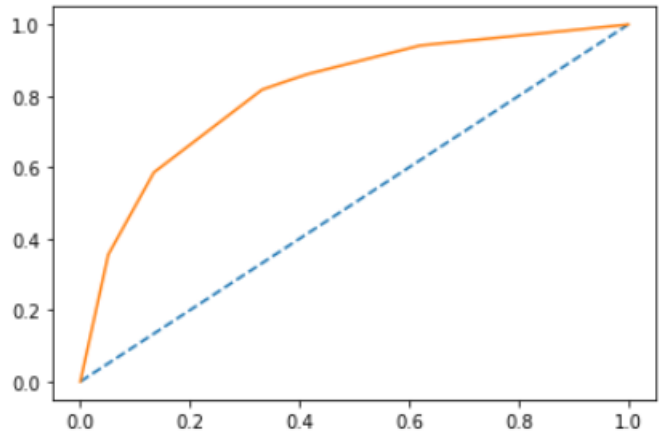
Question 2.3

Performance Metrics: Comment and Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score, classification reports for each model.

ANSWER:

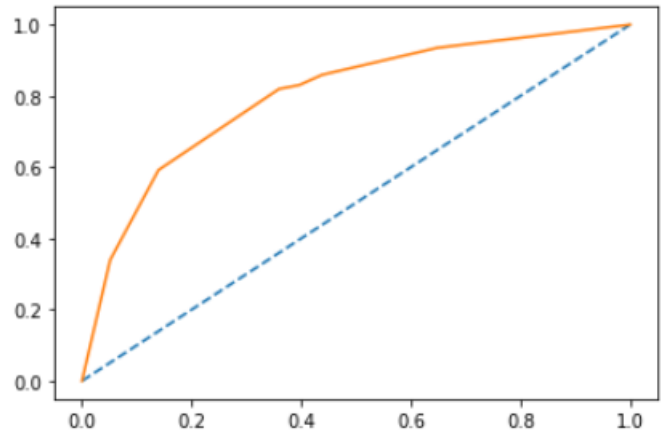
AUC and ROC for training data

```
AUC: 0.810

[<matplotlib.lines.Line2D at 0x197b102bc10>]
```



AUC and ROC for testing data

```
AUC: 0.799

[<matplotlib.lines.Line2D at 0x197b101fb80>]
```



Confusion matrix and classification report for testing data

```
confusion_matrix(test_labels, ytest_predict_dtcl)
```

```
array([[536,  87],
       [113, 164]], dtype=int64)
```

```
cart_test_acc=best_grid_dtcl.score(X_test,test_labels)
cart_test_acc
```

```
0.7777777777777778
```

```
print(classification_report(test_labels, ytest_predict_dtcl))
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       623
           1       0.65      0.59      0.62       277

    accuracy                           0.78       900
   macro avg       0.74      0.73      0.73       900
weighted avg       0.77      0.78      0.77       900
```

```
cart_metrics=classification_report(test_labels, ytest_predict_dtcl,output_dict=True)
df=pd.DataFrame(cart_metrics).transpose()
cart_test_f1=round(df.loc["1"][2],2)
cart_test_recall=round(df.loc["1"][1],2)
cart_test_precision=round(df.loc["1"][0],2)
print ('cart_test_precision ',cart_test_precision)
print ('cart_test_recall ',cart_test_recall)
print ('cart_test_f1 ',cart_test_f1)
```

```
cart_test_precision  0.65
cart_test_recall  0.59
cart_test_f1  0.62
```

conclusion for Training

```
AUC - 81%
Accuracy - 77%
Precision - 66%
f1-score - 62%
```

Conclusion for Testing

```
AUC - 80%
Accuracy - 77%
Precision - 83%
f1-score - 84%
```

Training and Test set results are almost similar, and with the overall measures high

RF model for training data

```
rf_train_acc=best_grid_rfcl.score(X_train,train_labels)
rf_train_acc
```

```
0.8042857142857143
```

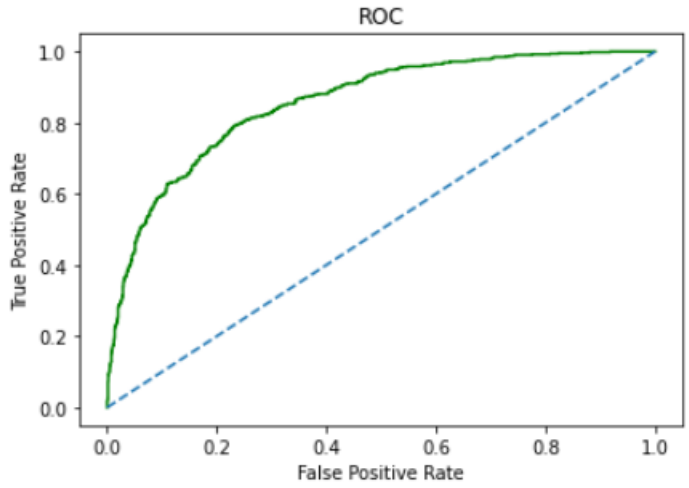
```
print(classification_report(train_labels,ytrain_predict_rfcl))
```

```
              precision    recall  f1-score   support

           0       0.84      0.89      0.86      1453
           1       0.72      0.61      0.66       647

    accuracy                           0.80      2100
   macro avg       0.78      0.75      0.76      2100
weighted avg       0.80      0.80      0.80      2100
```

```
rf_metrics=classification_report(train_labels, ytrain_predict_rfcl,output_dict=True)
df=pd.DataFrame(rf_metrics).transpose()
rf_train_precision=round(df.loc["1"][0],2)
rf_train_recall=round(df.loc["1"][1],2)
rf_train_f1=round(df.loc["1"][2],2)
print ('rf_train_precision ',rf_train_precision)
print ('rf_train_recall ',rf_train_recall)
print ('rf_train_f1 ',rf_train_f1)
```

```
rf_train_precision  0.72
rf_train_recall  0.61
rf_train_f1  0.66
```

```
Area under Curve is 0.8563713512840778
```



RF model for testing data

```
rf_test_acc=best_grid_rfcl.score(X_test,test_labels)
rf_test_acc
```

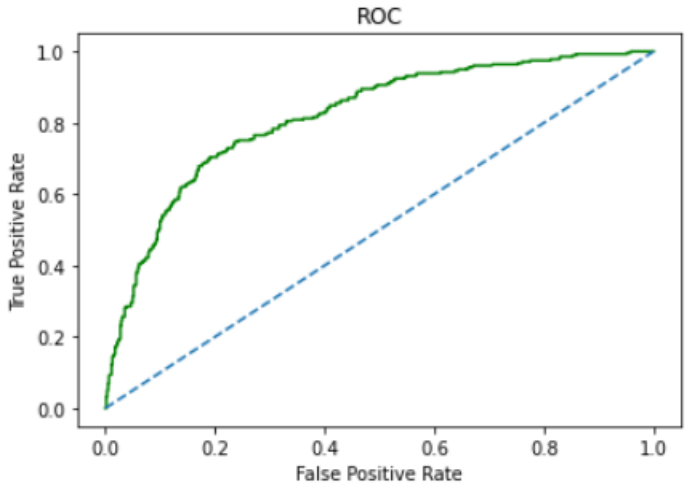
```
0.7844444444444445
```

```
print(classification_report(test_labels,ytest_predict_rfcl))
```

```
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       623
           1       0.68      0.56      0.62       277

    accuracy                           0.78       900
   macro avg       0.75      0.72      0.73       900
weighted avg       0.78      0.78      0.78       900
```

```
rf_metrics=classification_report(test_labels, ytest_predict_rfcl,output_dict=True)
df=pd.DataFrame(rf_metrics).transpose()
rf_test_precision=round(df.loc["1"][0],2)
rf_test_recall=round(df.loc["1"][1],2)
rf_test_f1=round(df.loc["1"][2],2)
print ('rf_test_precision ',rf_test_precision)
print ('rf_test_recall ',rf_test_recall)
print ('rf_test_f1 ',rf_test_f1)
```

```
rf_test_precision  0.68
rf_test_recall  0.56
rf_test_f1  0.62

Area under Curve is 0.8181994657271499
```



random forest conclusion

for training

- AUC: 86%
- Accuracy: 80%
- Precision: 72%
- f1-Score: 66%

for testing

- AUC: 82%
- Accuracy: 78%
- Precision: 68%
- f1-Score: 62

NM model for Training data

```
nn_train_acc=best_grid_nncl.score(X_train,train_labels)
nn_train_acc
```

```
0.7761904761904762
```

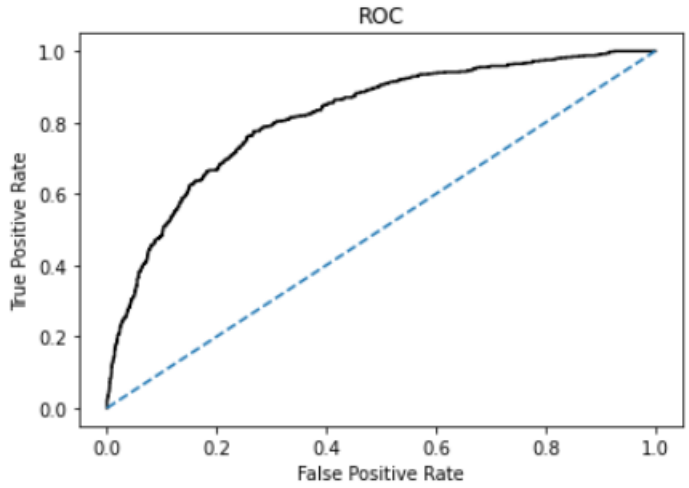
```
print(classification_report(train_labels,ytrain_predict_nncl))
```

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.85      1453
           1       0.68      0.51      0.59       647

    accuracy                           0.78      2100
   macro avg       0.74      0.70      0.72      2100
weighted avg       0.77      0.78      0.77      2100
```

```
nn_metrics=classification_report(train_labels, ytrain_predict_nncl,output_dict=True)
df=pd.DataFrame(nn_metrics).transpose()
nn_train_precision=round(df.loc["1"][0],2)
nn_train_recall=round(df.loc["1"][1],2)
nn_train_f1=round(df.loc["1"][2],2)
print ('nn_train_precision ',nn_train_precision)
print ('nn_train_recall ',nn_train_recall)
print ('nn_train_f1 ',nn_train_f1)
```

```
nn_train_precision  0.68
nn_train_recall  0.51
nn_train_f1  0.59
```

```
Area under Curve is 0.8166831721609928
```



NM model for testing data

```
nn_test_acc=best_grid_nncl.score(X_test,test_labels)
nn_test_acc
```

```
0.7688888888888888
```

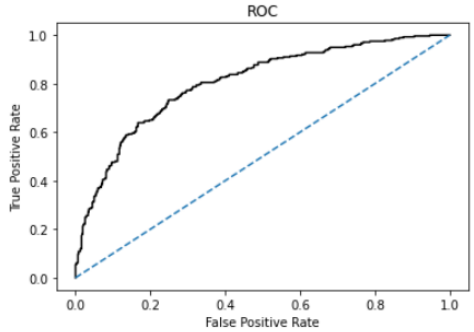
```
print(classification_report(test_labels,ytest_predict_nncl))
```

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       623
           1       0.67      0.50      0.57       277

    accuracy                           0.77       900
   macro avg       0.73      0.69      0.71       900
weighted avg       0.76      0.77      0.76       900
```

```
nn_metrics=classification_report(test_labels, ytest_predict_nncl,output_dict=True)
df=pd.DataFrame(nn_metrics).transpose()
nn_test_precision=round(df.loc["1"][0],2)
nn_test_recall=round(df.loc["1"][1],2)
nn_test_f1=round(df.loc["1"][2],2)
print ('nn_test_precision ',nn_test_precision)
print ('nn_test_recall ',nn_test_recall)
print ('nn_test_f1 ',nn_test_f1)
```

```
nn_test_precision  0.67
nn_test_recall  0.5
nn_test_f1  0.57
```

```
Area under Curve is 0.8044225275393896
```



NM conclusion

---

for training

- AUC: 82%
- Accuracy: 78%
- Precision: 68%
- f1-Score: 59

for testing

- AUC: 80%
- Accuracy: 77%
- Precision: 67%
- f1-Score: 57%

Question 2.4

Final Model: Compare all the models and write an inference which model is best/optimized

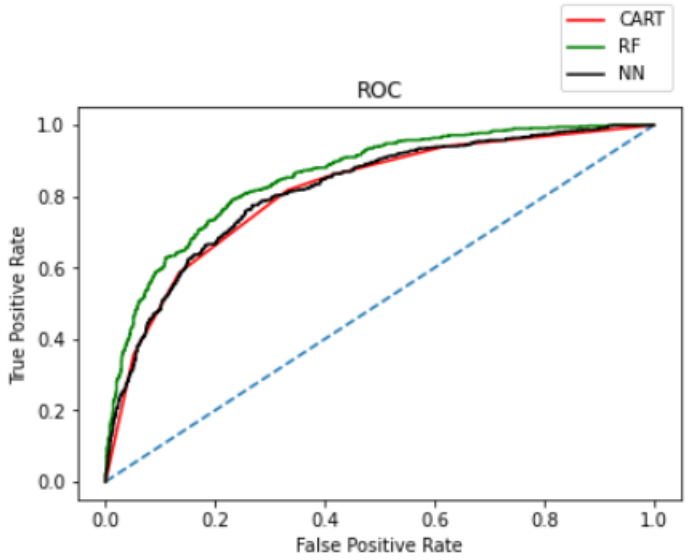
Answer:

By comparing all the models we get a data

	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.78	0.78	0.80	0.78	0.78	0.77
AUC	0.81	0.80	0.86	0.82	0.82	0.80
Recall	0.59	0.59	0.61	0.56	0.51	0.50
Precision	0.66	0.65	0.72	0.68	0.68	0.67
F1 Score	0.62	0.62	0.66	0.62	0.59	0.57

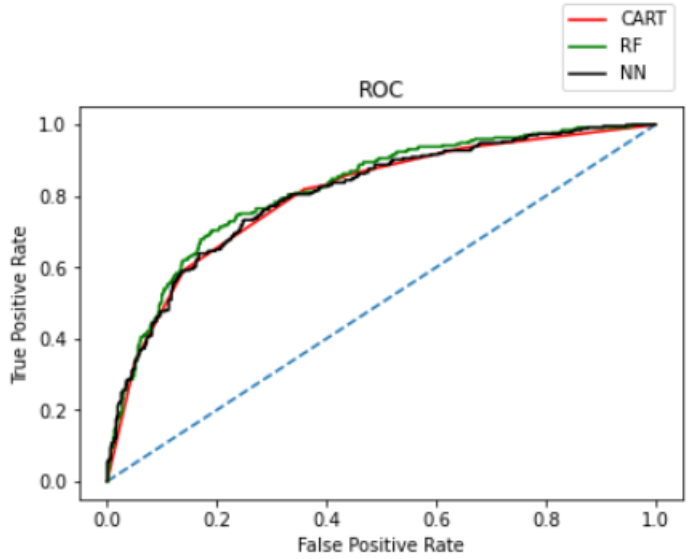
ROC curve for 3 model on training data

```
<matplotlib.legend.Legend at 0x197b16d2fd0>
```



ROC curve for 3 model on testing data

```
<matplotlib.legend.Legend at 0x197b1708ee0>
```



CONCLUSION :

#I am selecting the RF model, as it has better accuracy, precsion, recall, f1 score better than other two CART & NN.

Question 2.5

Inference: Based on the whole Analysis, what are the business insights and recommendations

Answer:

#This is understood by looking at the insurance data by drawing relations between different variables such as day of the incident, time, age group, and associating it with other external information such as location, behavior patterns, weather information, airline/vehicle types, etc. #Streamlining online experiences benefitted customers, leading to an increase in conversions, which subsequently raised profits. • As per the data 90% of insurance is done by online channel. • Other interesting fact, is almost all the offline business has a claimed associated, need to find why? • Need to train the JZI agency resources to pick up sales as they are in bottom, need to run promotional marketing campaign or evaluate if we need to tie up with alternate agency • Also based on the model we are getting 80%accuracy, so we need customer books airline tickets or plans, cross sell the insurance based on the claim data pattern. • Other interesting fact is more sales happen via Agency than Airlines and the trend shows the claim are processed more at Airline. So we may need to deep dive into the process to understand the workflow and why? #Key performance indicators (KPI) The KPI's of insurance claims are: • Reduce claims cycle time • Increase customer satisfaction • Combat fraud • Optimize claims recovery • Reduce claim handling costs Insights gained from data and AI-powered analytics could expand the boundaries of insurability, extend existing products, and give rise to new risk transfer solutions in areas like a non-damage business interruption and reputational damage.

END