DATA MINING

BUSINESS REPORT

THAKUR ARUN SINGH

MAY 2021

This Business Report shall provide detailed explanation of how we approached each problem given in the assignment. It shall also provide relative resolution and explanation with regards to the problems

CONTENTS

Problem 1: 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.

PROBLEM 1.1

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

Resolution:

Describing the data:

- First we import all the necessary libraries in Python, and then import the data file which is 'bank_marketing_part1_Data'. Once we import the file we confirm whether the data has been uploaded correctly or not using 'head' function. Using this function we can view the data and all the columns and headers whether they are aligning correctly or not.
- Then using the 'shape' function we can understand how many row and columns are there in our data set.
- To check the data type of all the columns and also to check the null values, 'info' function. Has been used.
- To see the detail description of the data such as, Count, Mean, Median, Min, Max, Standard Deviations etc,

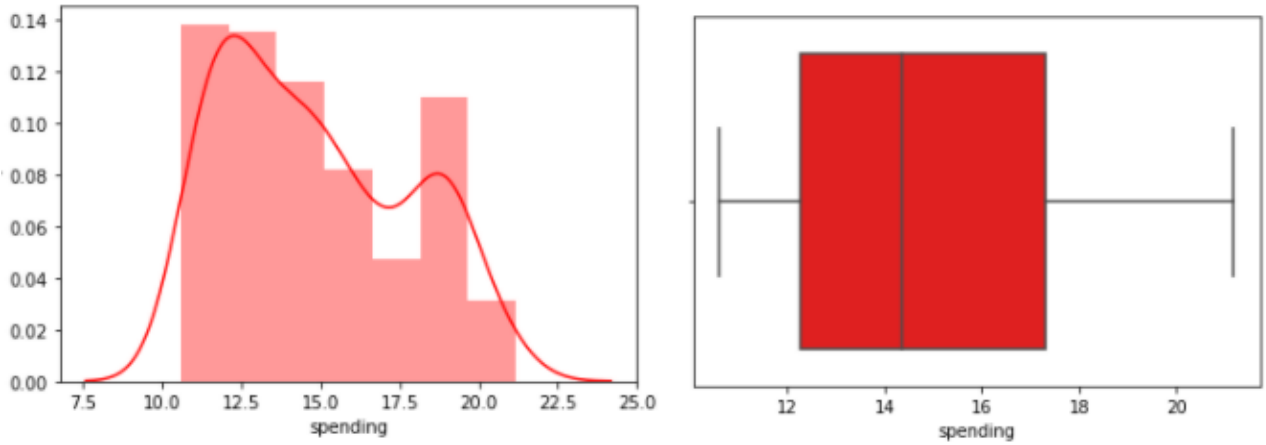
	count	mean	std	min	25%	50%	75%	max
spending	210	14.84752	2.909699	10.59	12.27	14.355	17.305	21.18
advance_payments	210	14.55929	1.305959	12.41	13.45	14.32	15.715	17.25
probability_of_full_payment	210	0.870999	0.023629	0.8081	0.8569	0.87345	0.887775	0.9183
current_balance	210	5.628533	0.443063	4.899	5.26225	5.5235	5.97975	6.675
credit_limit	210	3.258605	0.377714	2.63	2.944	3.237	3.56175	4.033
min_payment_amt	210	3.700201	1.503557	0.7651	2.5615	3.599	4.76875	8.456
max_spent_in_single_shopping	210	5.408071	0.49148	4.519	5.045	5.223	5.877	6.55

- Using the 'isnull' function, one can understand if there are any null values in the data set. And we do not have any null values in the existing data set.
- Using the 'dups' function we check for the duplicates and there were no duplicate values.

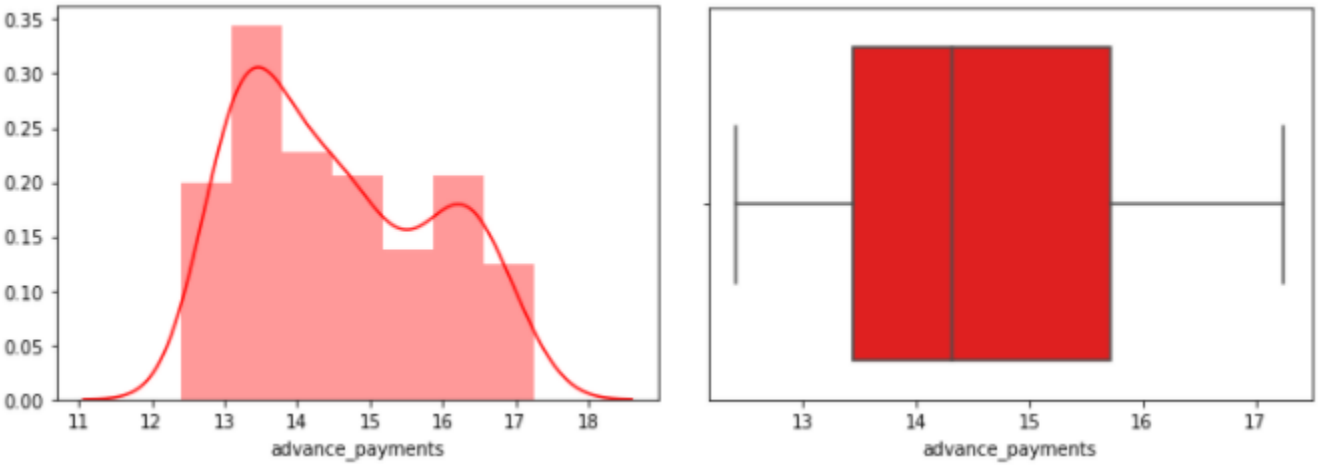
After reviewing the data thoroughly, and based on the above analysis we can say that, we have seven variables, Mean and Median values are almost equal, and Standard deviation for 'Spending' is higher than other variables. There are no duplicates in the data set.

Exploratory data analysis

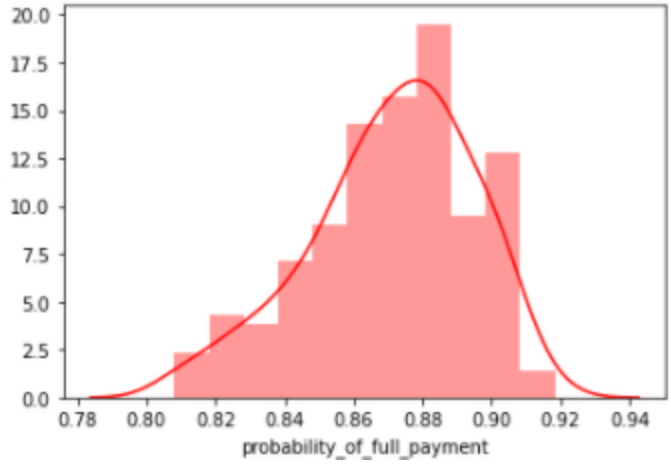
Univariate and multivariate analysis

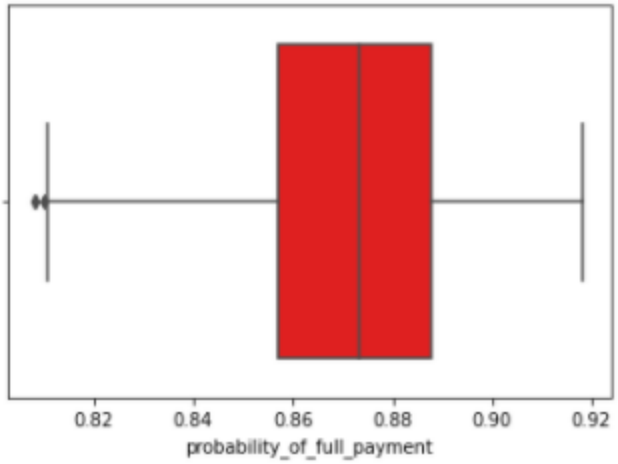


The above dist plot shows the distribution of data from 10 – 22 and is positively skewed. Boxplot shows that there are no outliers.

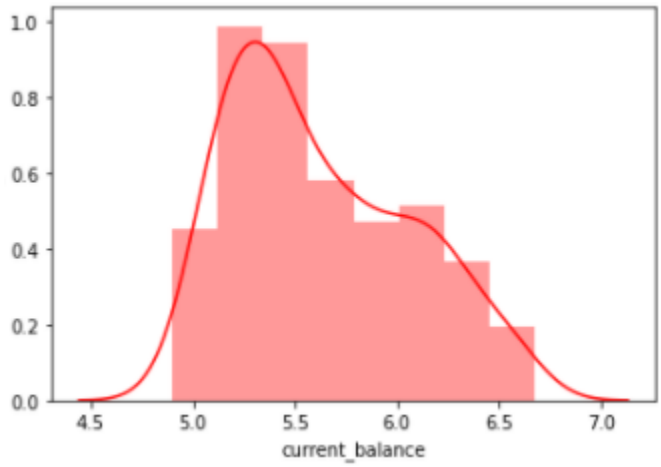


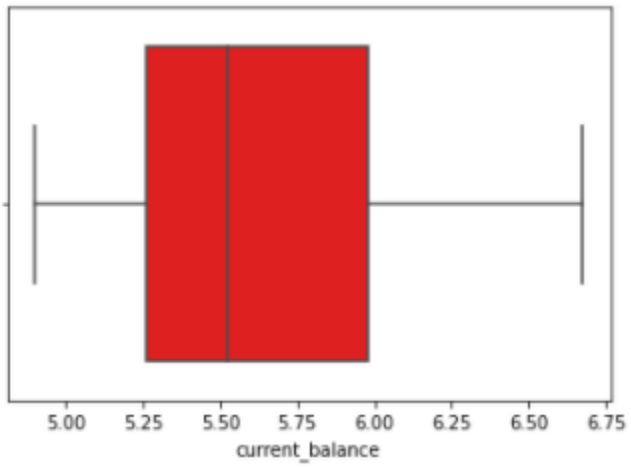
The above dist plot shows the distribution of data from 12 – 17 and is positively skewed. Boxplot shows that there are no outliers



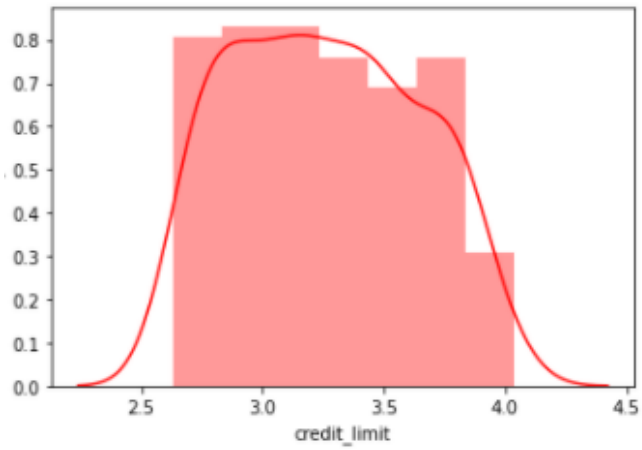


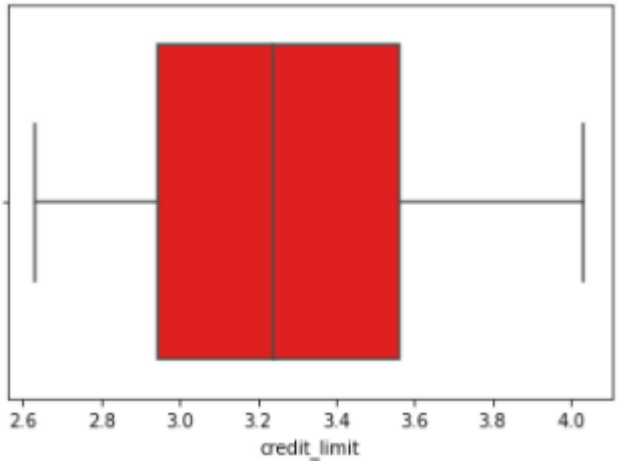
The above dist plot shows the distribution of data from 0.80 – 0.92 and is negatively skewed. Boxplot shows that there are a few outliers. Probably values is good above 80%.



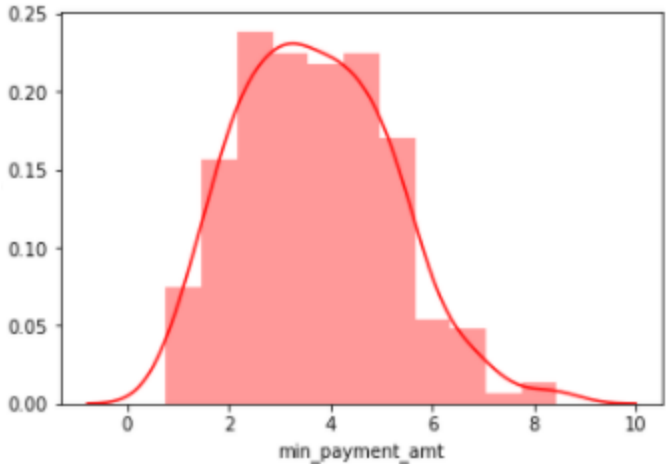


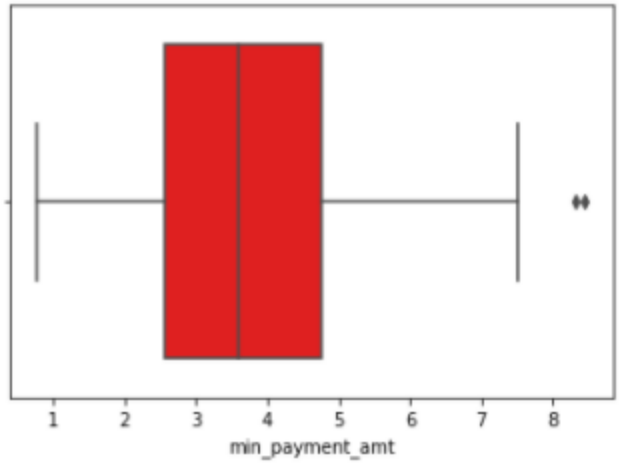
The above dist plot shows the distribution of data from 5.0 – 6.5 and is positively skewed. Boxplot shows that there are no outliers.



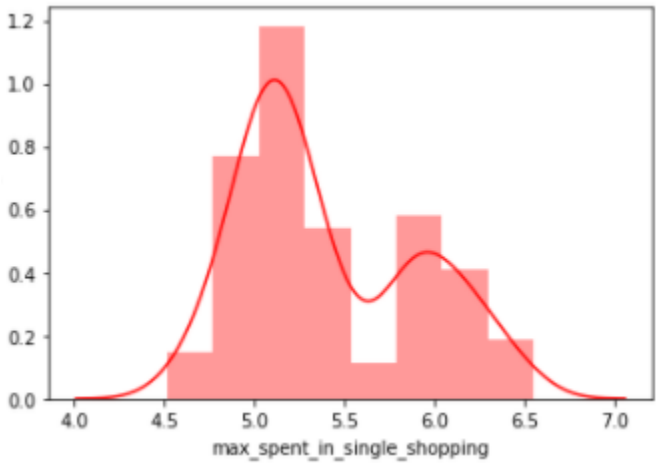


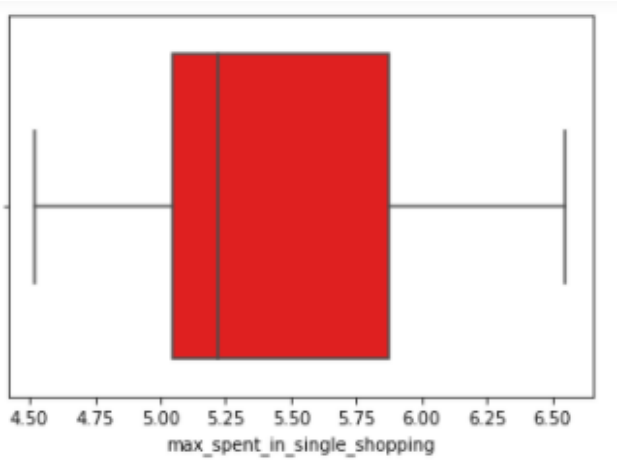
The above dist plot shows the distribution of data from 2.5 – 4.0 and is positively skewed. Boxplot shows that there are no outliers.





The above dist plot shows the distribution of data from 1 – 8 and is positively skewed. Boxplot shows that there are a few outliers.

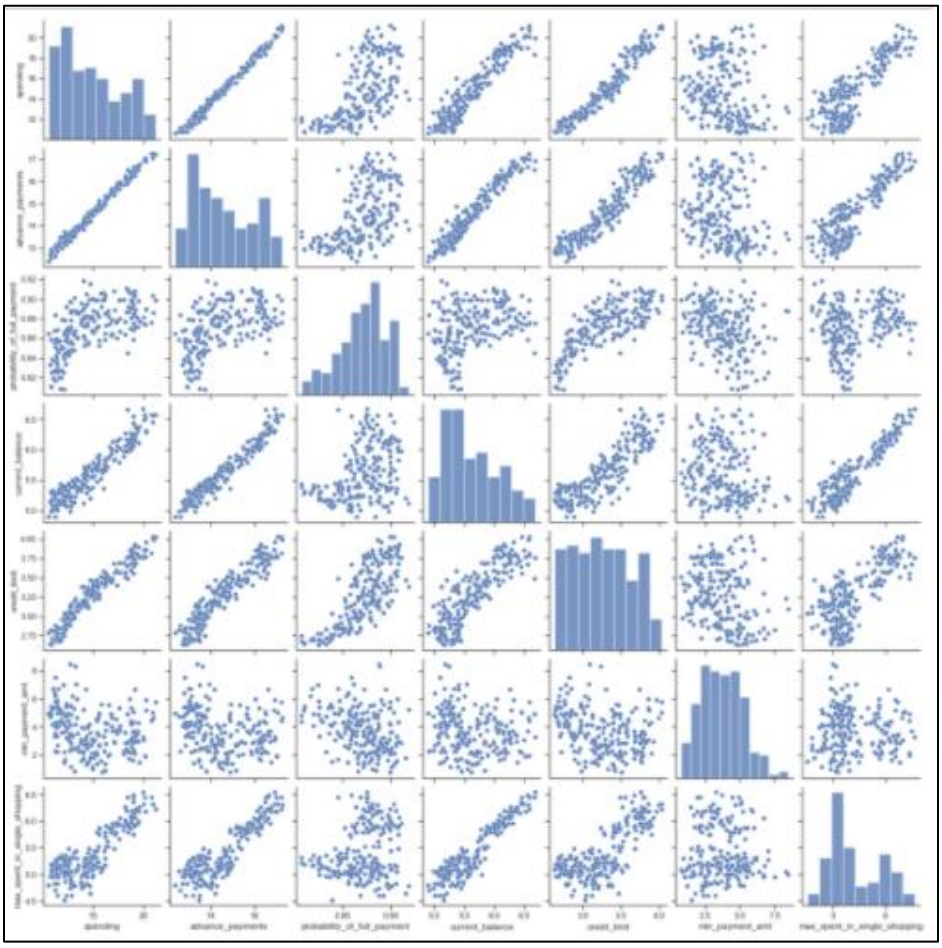


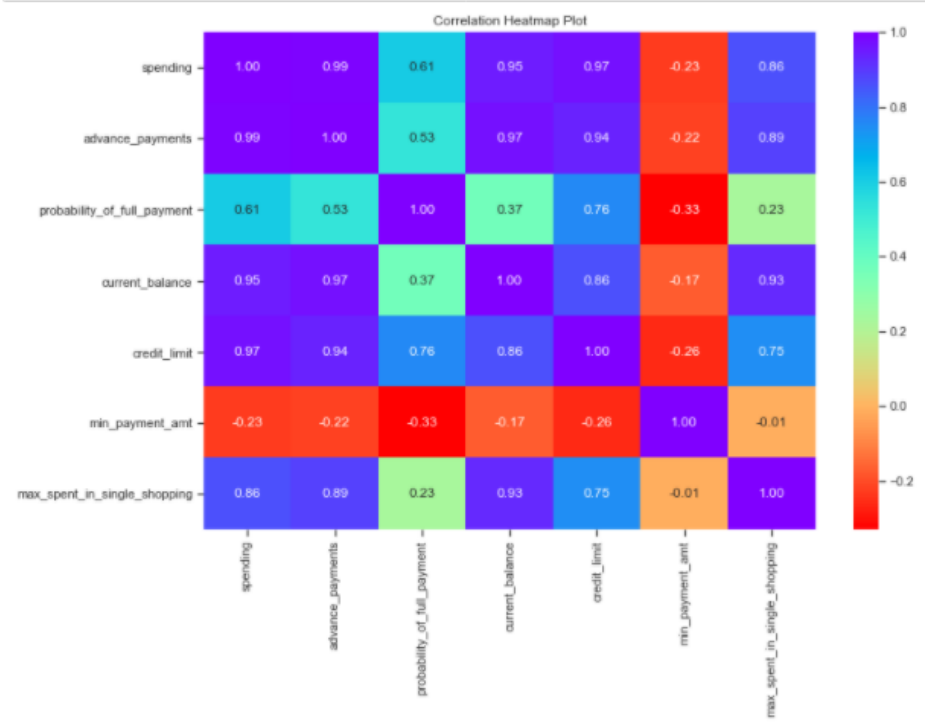


The above dist plot shows the distribution of data from 4.5 – 6.5 and is positively skewed. Boxplot shows that there are a few outliers.

Outlier will not be treated as there are only 3 to 4 values which were observed in the data set.

Multivariate Analysis





From both the analysis we can see that there is strong positive correlation between all the variables except ‘min_payment_amt’

PROBLEM 1.2

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

Resolution:

As the data is unscaled,it is imperative that we perform the scaling as this model works on distance based components. As we know that the values of all the variables are in different scale.

For Example: ‘Spending’ and ‘advance payments’ are in different values and this may get more weight.

Scaling will bring down the values to relatively same range. Using standard scalar below is the data how it looks after scaling.

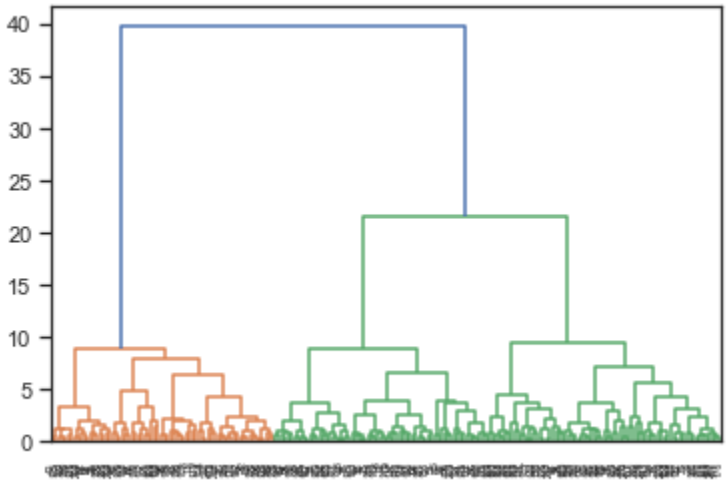
	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
0	1.754355	1.811968	0.17823	2.367533	1.338579	-0.29881	2.328998
1	0.393582	0.25384	1.501773	-0.600744	0.858236	-0.24281	-0.538582
2	1.4133	1.428192	0.504874	1.401485	1.317348	-0.22147	1.509107
3	-1.38403	-1.22753	-2.591878	-0.793049	-1.63902	0.987884	-0.454961
4	1.082581	0.998364	1.19634	0.591544	1.155464	-1.08815	0.874813

PROBLEM 1.3

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

Resolution:

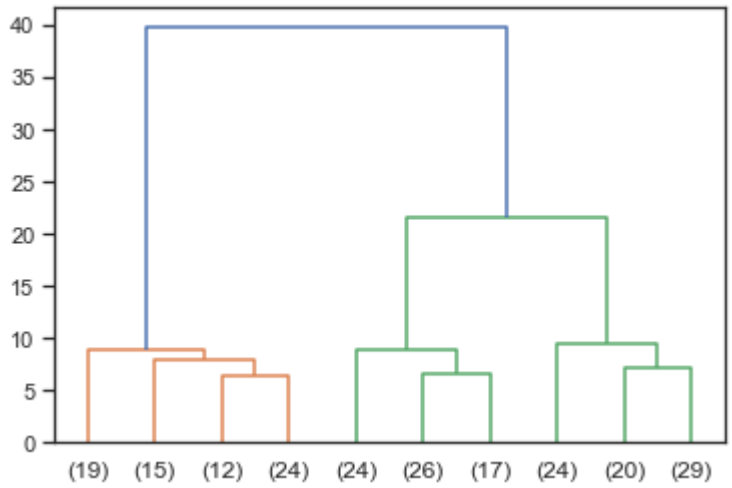
For Hierarchical clustering we can use ward’s method for scaled data



From the above dendrogram, we can see analyze that all the data points has clustered in different clusters.

To achieve the business objective and to obtain the optimal number of clusters we can use ‘truncate mode as lastp.’

Where we can give the last p = 10 according to industry standards.



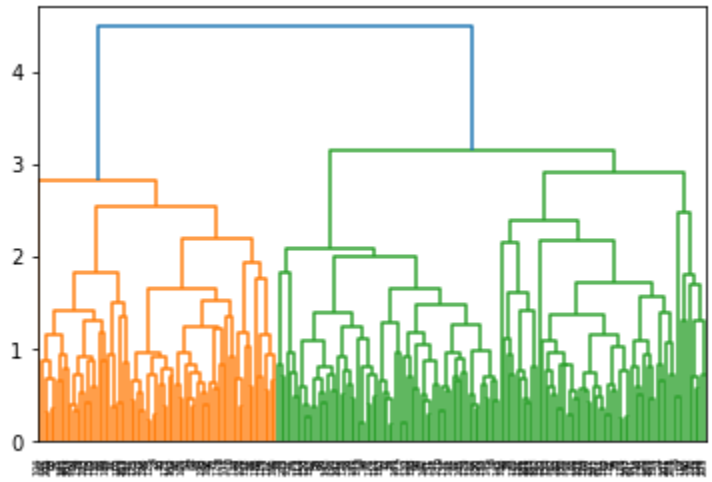
Here we can see that the data has clustered into three clusters.

Using the criterion as ‘maxclust’ and fclusters we can map these clusters to the data set.

	spending	advance_ payments	probability_of _full_payment	current_bal ance	credit_lim it	min_pay ment_amt	max_spent_in_s ingle_shopping	fcluster
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.55	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	5.89	3.694	2.068	5.837	1

We can now look at the cluster frequency in our data set

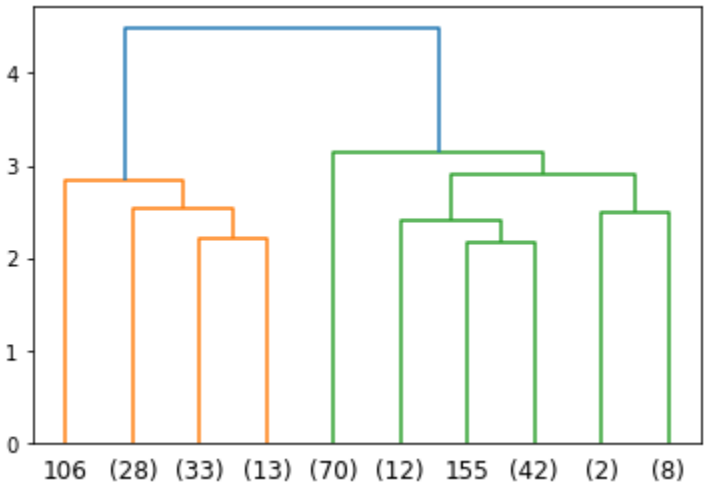
We can choose average method to the scaled data



From the above dendrogram, we can see analyze that all the data points has clustered in different clusters by average method.

To achieve the business objective and to obtain the optimal number of clusters we can use ‘truncate mode as lastp.’

Where we can give the last p = 10 according to industry standards.



Here we can see that the data has clustered into three clusters.

Using the criterion as ‘maxclust’ and fclusters we can map these clusters to the data set.

Clustering –

Analysis – From the above analysis we can say that there is not much variation in both methods. Both methods has similar means, minor variation which is predictable.

Clustering – Based on the above dendrograms 3 to 4 clustering groups looks perfect. As per the dataset what we have we can go with 3 groups of clusters. This gives us a patter based on

high/medium/low spending with max_spent_in_single_shopping (high value item) and probability_of_full_payment (payment made).

PROBLEM 1.4

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

Resolution:

For K-Means clustering, we can randomly decide to give n_clusters = 3 and we look at the distribution of clusters according to the n_clusters.

We apply K-means technique to the scaled data.

Cluster output for all the observations in the dataset.

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

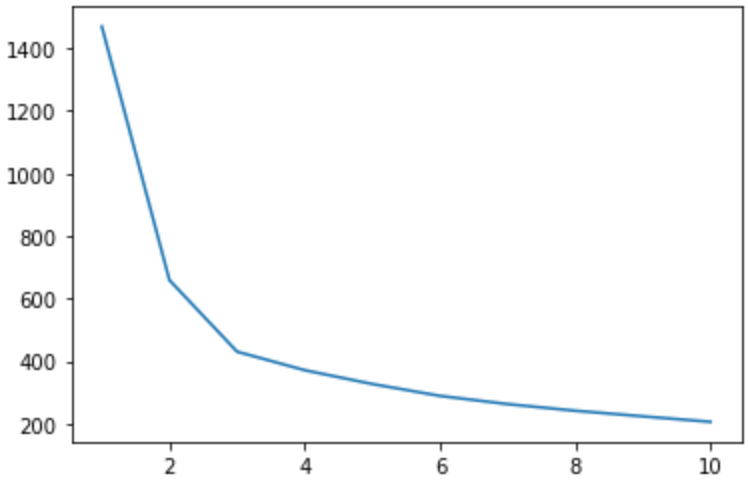
Now as we have 3 clusters 0,1,2 and to find the optimal number of clusters, we can use k-elbow method.

```
wss
```

```
[1469.9999999999998,
 659.171754487041,
 430.6589731513006,
 371.38509060801096,
 327.21278165661346,
 289.31599538959495,
 262.98186570162267,
 241.81894656086033,
 223.91254221002725,
 206.39612184786694]
```

To obtain the inertia value for all the clusters from 1 to 11, we can use a ‘for loop’ to find the optimal number of clusters.

The silhouette score for 3 clusters is good - 0.4007270552751299



From the above elbow curve we can see that after 3 clusters there is no huge drop in the values, so we select 3 clusters.

So adding the cluster results to our dataset to solve our business objective

ıce_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	fcluster	Kmeans_clusters	(silhouet Kmeans_
16.92	0.8752	6.675	3.763	3.252	6.550	1	2	
14.89	0.9064	5.363	3.582	3.336	5.144	3	0	
16.42	0.8829	6.248	3.755	3.368	6.148	1	2	
12.96	0.8099	5.278	2.641	5.182	5.185	2	1	
15.86	0.8992	5.890	3.694	2.068	5.837	1	2	

Observations - By K-Means method we came at cluster 3. We find it optimal after there is no huge drop in inertia values. Also the elbow curve seems to show similar results.

The silhouette width score of the K–Means also seems to very less value that indicates all the data points are properly clustered. There is no mismatch in the data points with regards to clustering.

Based on the above dendrograms 3 to 4 clustering groups looks perfect. As per the dataset what we have we can go with 3 groups of clusters.

This gives us a patter based on high/medium/low spending with max_spent_in_single_shopping (high value item) and probability_of_full_payment (payment made).

PROBLEM 1.5

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

Resolution:

Based on the above analysis we can divide the cluster profiles in three groups.

High Spending, Medium Spending and Low Spending.

Group 1 – High Spending:

- Offering reward points might increase their purchases / spend.
- Maximum max_spent_in_single_shopping is high for this group, so we can offer discount / offer on next transactions upon full payment.
- Increase their credit limit which in turn increases the spending habits.
- Can offer loans against the credit card, as they are high value customers with good repayment record.
- Tie up with luxury brands, which will drive more one_time_maximun spending.

Group 2 – Medium Spending:

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

Group 3 – Low Spending:

- These customers should be given remainders for payments.
- Offers can be provided on early payments to improve their payment rate
- Increase their spending habits by tying up with grocery stores, utilities (electricity, phone, gas, others)

***^^**

Problem 2:

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.

PROBLEM 2.1

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

Resolution:

- First we import all the necessary libraries in Python, and then import the data file which is ‘insurance_part2_data’. Once we import the file we confirm whether the data has been uploaded correctly or not using ‘head’ function. Using this function we can view the data and all the columns and headers whether they are aligning correctly or not.

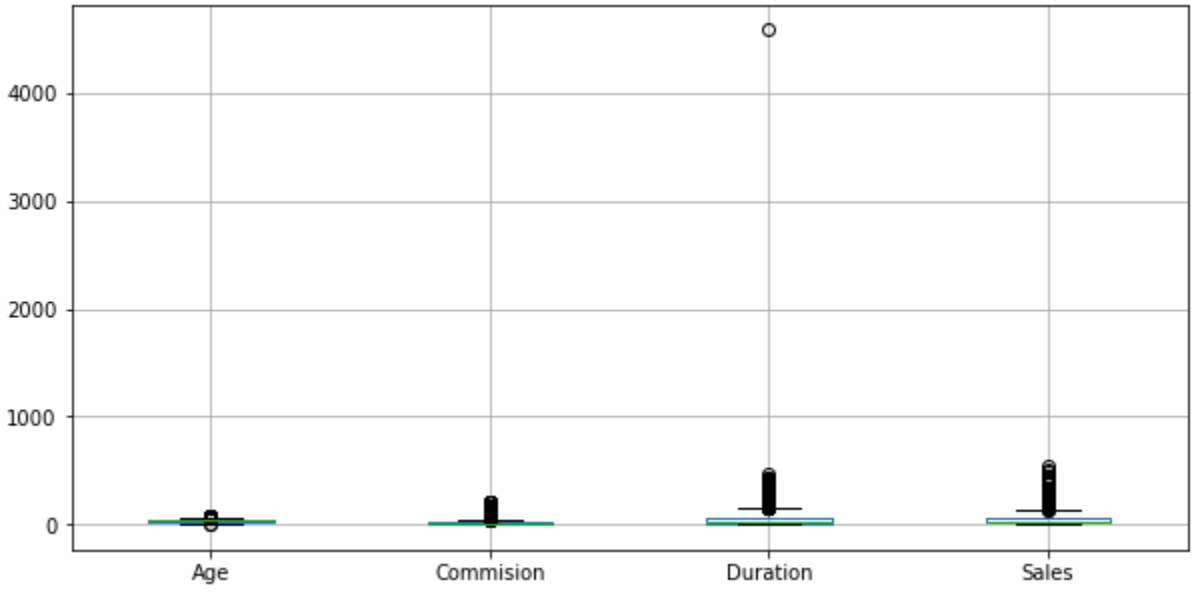
- Then using the ‘shape’ function we can understand how many row and columns are there in our data set.
- To check the data type of all the columns and also to check the null values, ‘info’ function. Has been used.
- To see the detail description of the data such as, Count, Mean, Median, Min, Max, Standard Deviations etc,

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Age	3000	NaN	NaN	NaN	38.091	10.4635	8	32	36	42	84
Agency_Code	3000	4	EPX	1365	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Type	3000	2	Travel Agency	1837	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Claimed	3000	2	No	2076	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Commission	3000	NaN	NaN	NaN	14.5292	25.4815	0	0	4.63	17.235	210.21
Channel	3000	2	Online	2954	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Duration	3000	NaN	NaN	NaN	70.0013	134.053	-1	11	26.5	63	4580
Sales	3000	NaN	NaN	NaN	60.2499	70.734	0	20	33	69	539
Product Name	3000	5	Customized Plan	1136	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Destination	3000	3	ASIA	2465	NaN	NaN	NaN	NaN	NaN	NaN	NaN

- Using the ‘isnull’ function, one can understand if there are any null values in the data set. And we do not have any null values in the existing data set.
- Using the ‘dups’ function we check for the duplicates and there were few duplicate values which are noted.
- Using the ‘drop_duplicates’ function, we can exclude the duplicate values. Then check for the data.

Checking for Outliers

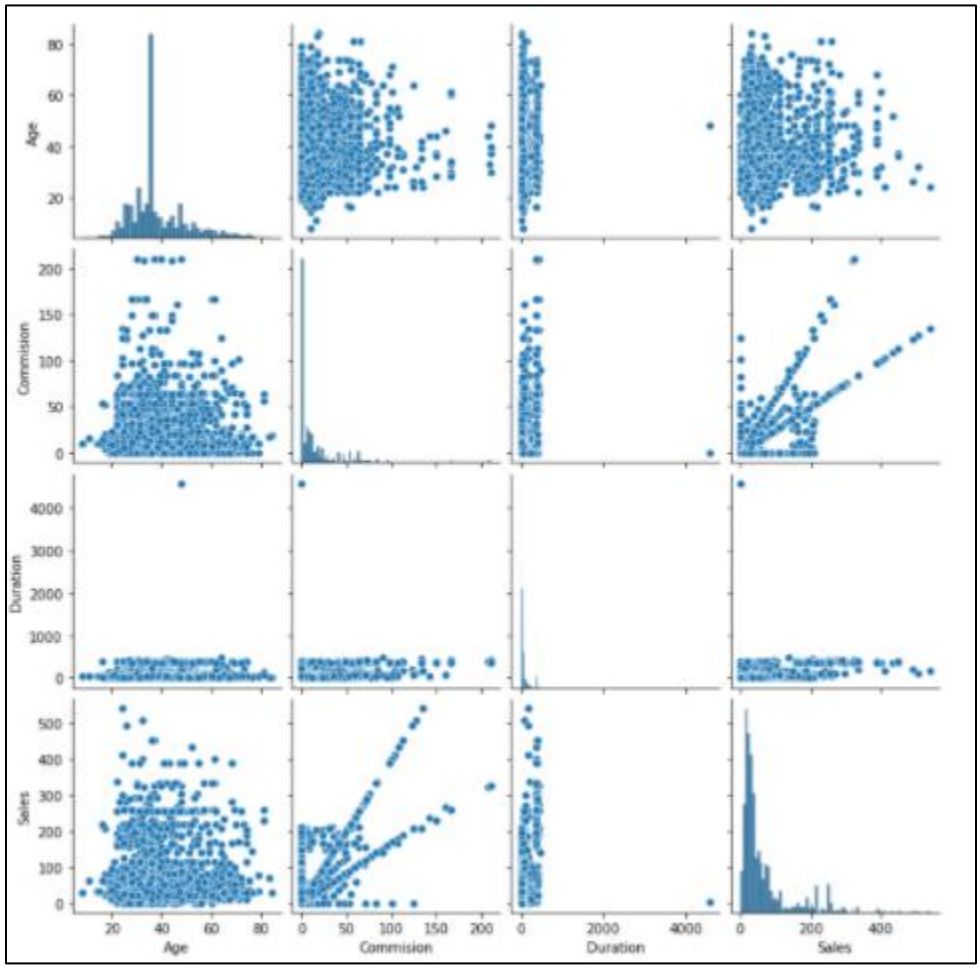
As there is no unique identifier I’m not dropping the duplicates it may be different customer’s data.



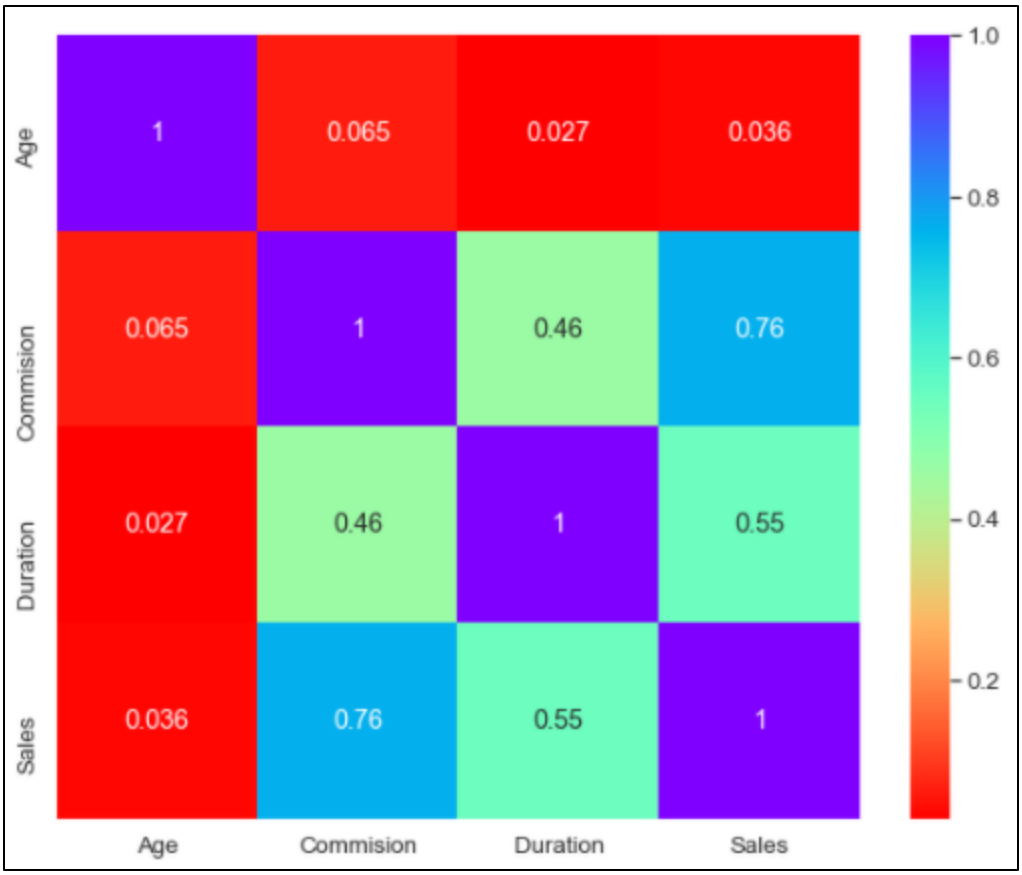
Outliers exist in almost all the numeric values.

We can treat outliers in random forest classification.

Checking pairplot distribution of the continuous variables

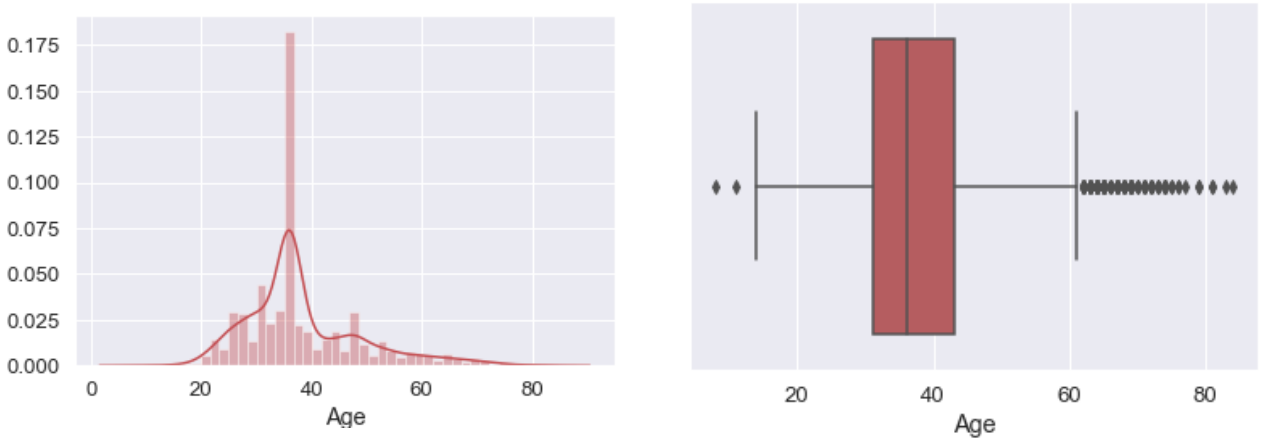


Checking for Correlations

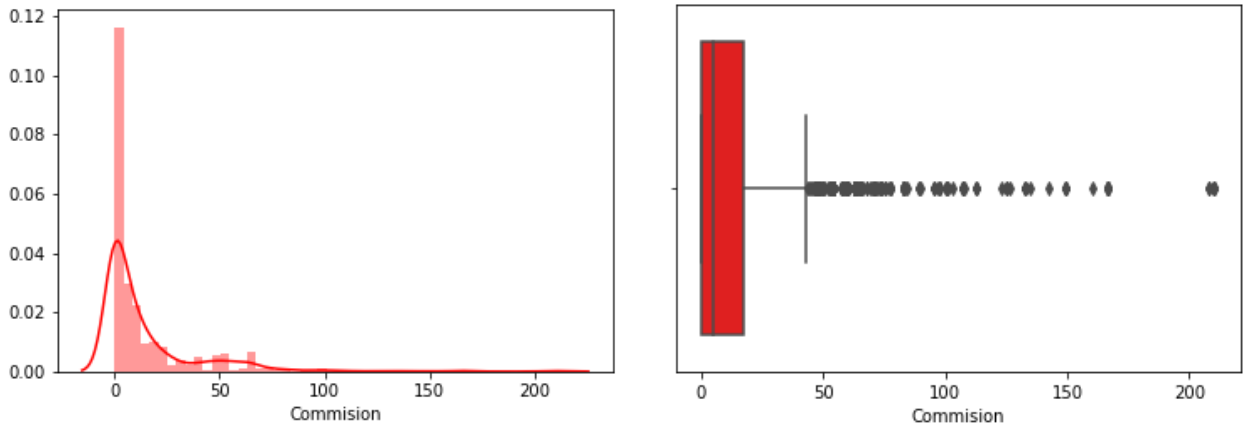


AGENCY_CODE: 4
JZI 239
CWT 472
C2B 924
EPX 1365
TYPE: 2
Airlines 1163
Travel Agency 1837
CLAIMED: 2
Yes 924
No 2076
CHANNEL: 2
Offline 46
Online 2954
PRODUCT NAME: 5
Gold Plan 109
Silver Plan 427
Bronze Plan 650
Cancellation Plan 678
Customised Plan 1136
DESTINATION: 3
EUROPE 215
Americas 320
ASIA 2465

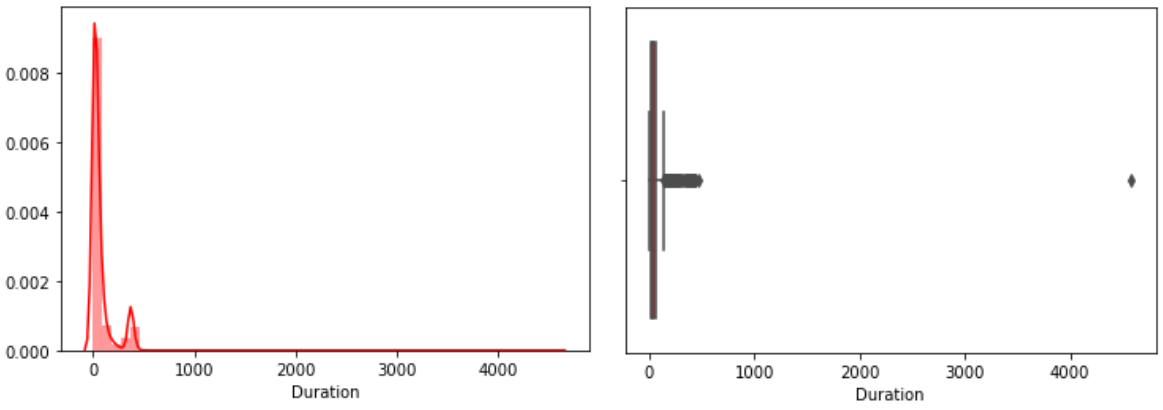
Univariate



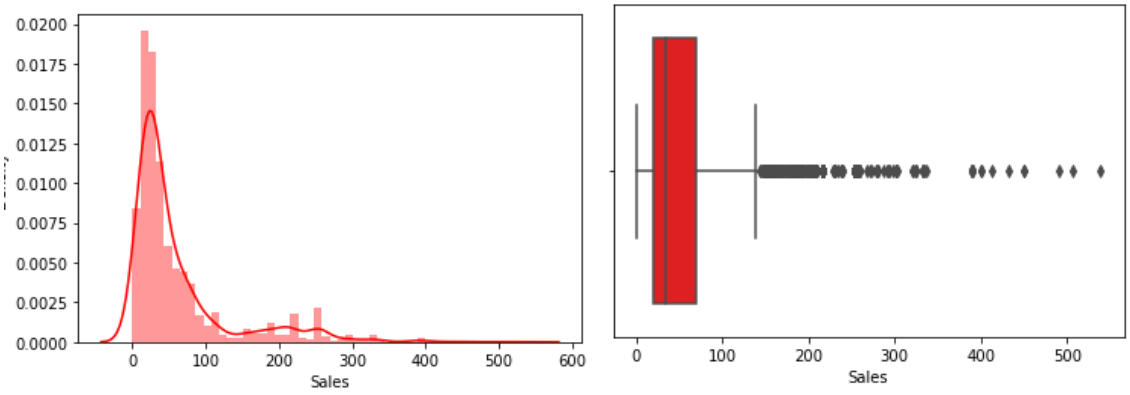
The above dist plot shows the distribution of data from 20 – 80 and is positively skewed. Boxplot shows that there are no outliers. Majority of the distribution lies in the range of 30 – 40.



The above dist plot shows the distribution of data from 0 – 30 and is positively skewed. Boxplot shows that there a few outliers.

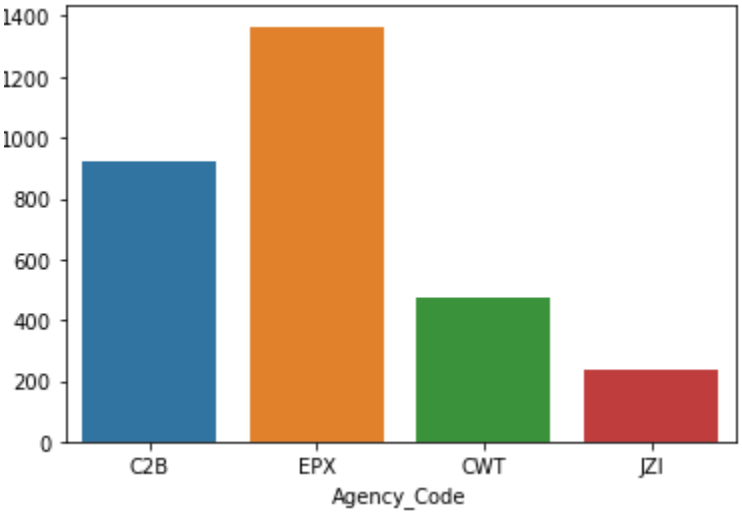


The above dist plot shows the distribution of data from 0 – 100 and is positively skewed. Boxplot shows that there are a few outliers.

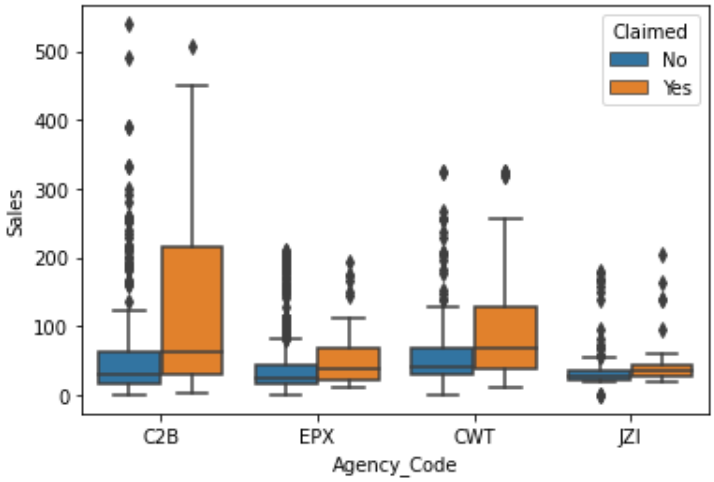


The above dist plot shows the distribution of data from 30 – 300 and is positively skewed. Boxplot shows that there are a few outliers.

Categorical variables

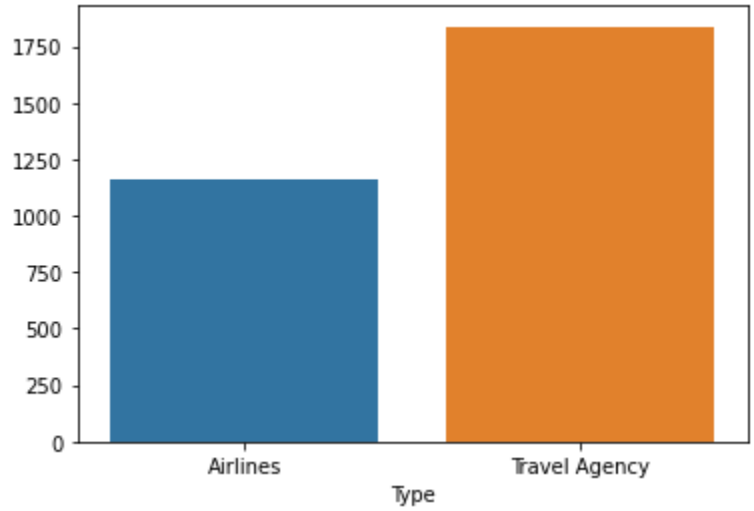


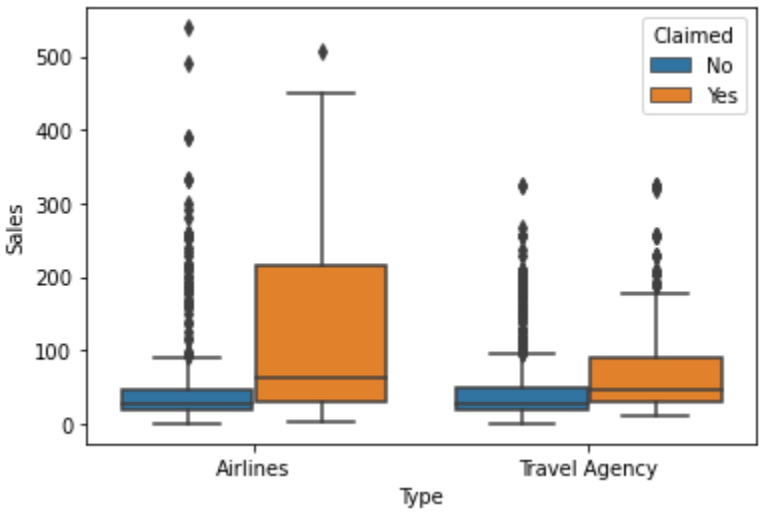
The distribution of the agency code shows us EXP with maximum frequency.



The box plot shows the split of sales with different agency code and also hue having claimed column.

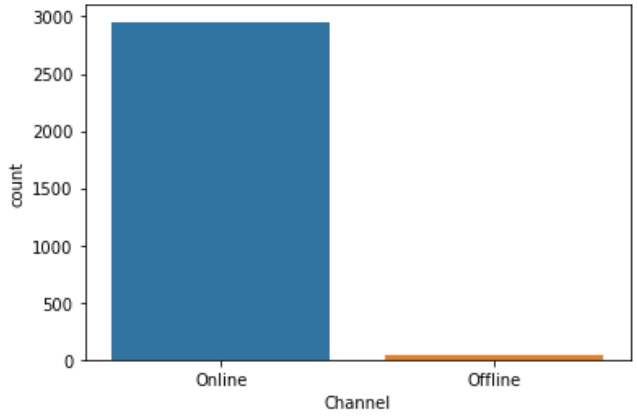
It seems that C2B have claimed more claims than other agency.



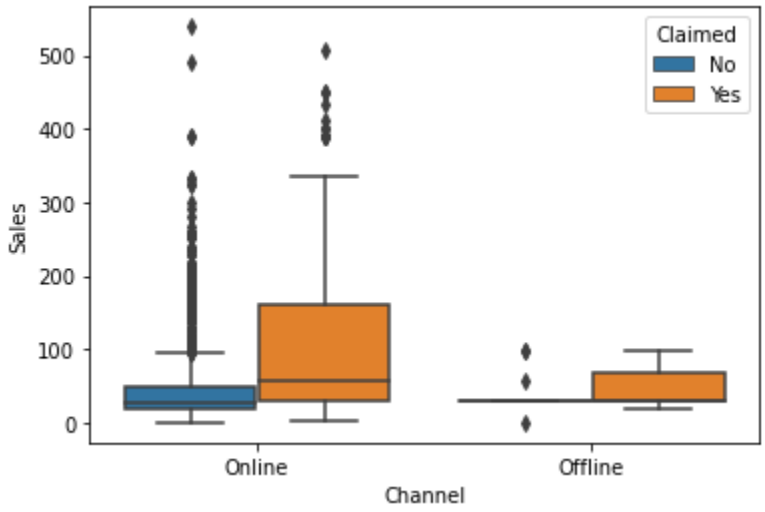


The box plot shows the split of sales with different type and also hue having claimed column.

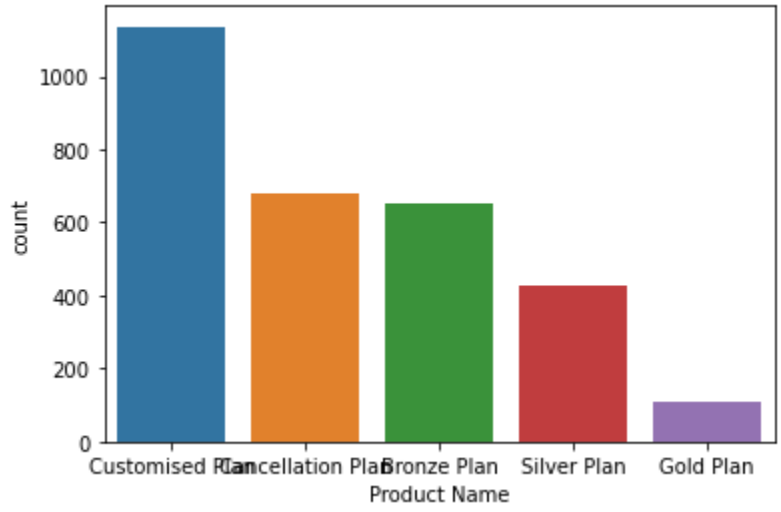
We could understand airlines type has more claims.



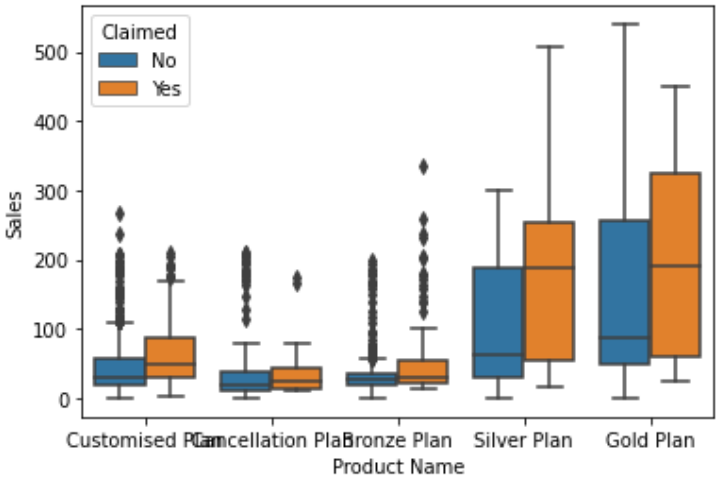
The majority of customers have used online medium, very less with offline medium



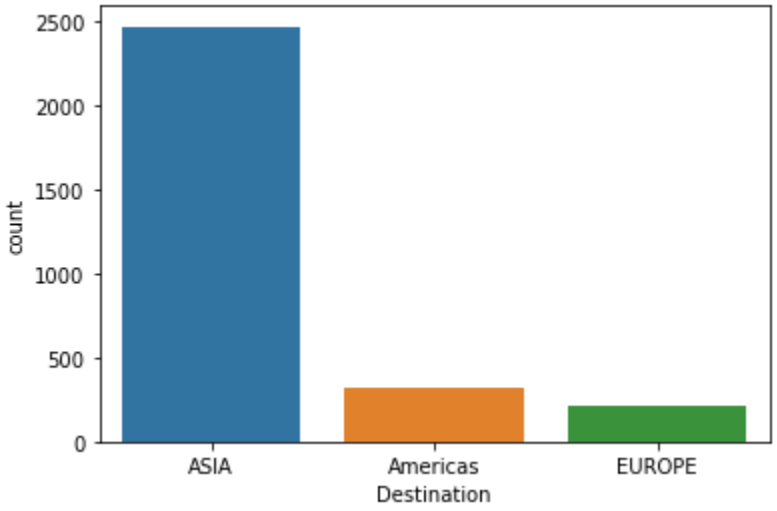
The box plot shows the split of sales with different channel and also hue having claimed column.



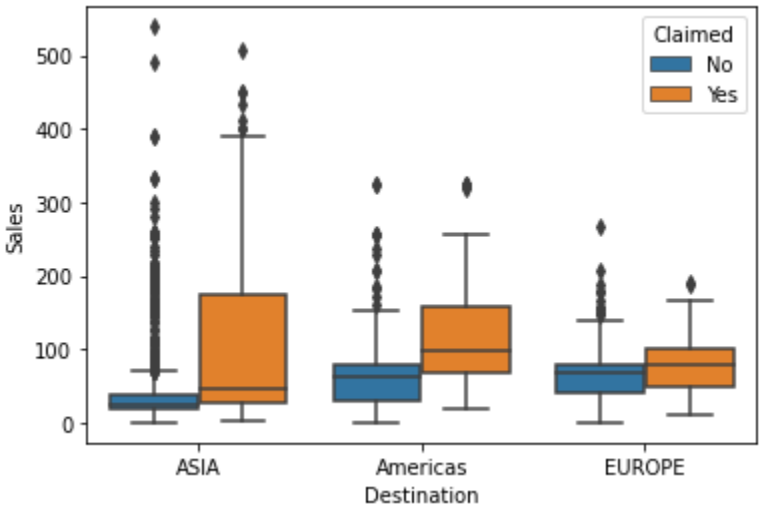
Customized plan seems to be most liked plan by customers when compared to all other plans.



The box plot shows the split of sales with different product name and also hue having claimed column.



Asia is where customers choose when compared with other destination places.



The box plot shows the split of sales with different destination and also hue having claimed column.

PROBLEM 2.2

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

Resolution:

For training and testing purpose we are splitting the dataset into train and test data in the ratio 70:30.

We have divided the dataset into train and test.

MODEL 1

Building A Decision Tree Classifier

```
dt_model = DecisionTreeClassifier(criterion = 'gini' )
```

```
dt_model.fit(X_train, train_labels)
```

35]: DecisionTreeClassifier()

Checking for the feature

```
print (pd.DataFrame(dt_model.feature_importances_, columns = ["Imp"],
                    index = X_train.columns).sort_values('Imp',ascending=False))
```

	Imp
Duration	0.276811
Agency_Code	0.194356
Sales	0.194228
Age	0.163714
Commision	0.102841
Product Name	0.038334
Destination	0.019359
Channel	0.007262
Type	0.003095

Grid Search for finding out the optimal values for the hyper parameters

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'max_depth': [4, 5,6],
    'min_samples_leaf': [20, 40, 60, 70],
    'min_samples_split': [150, 200, 250, 300,]
}

dt_model = DecisionTreeClassifier()

grid_search = GridSearchCV(estimator = dt_model, param_grid = param_grid, cv = 10)
```

.

```
grid_search.fit(X_train, train_labels)
```

: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
param_grid={'max_depth': [4, 5, 6],
'min_samples_leaf': [20, 40, 60, 70],
'min_samples_split': [150, 200, 250, 300]})

```
best_grid
```

DecisionTreeClassifier(max_depth=4, min_samples_leaf=20, min_samples_split=150)

Regularizing the Decision Tree

Adding Tuning Parameters

```
reg_dt_model = DecisionTreeClassifier(criterion = 'gini', max_depth = 4,min_samples_leaf=20,min_samples_split=150)
```

```
reg_dt_model.fit(X_train, train_labels)
```

```
DecisionTreeClassifier(max_depth=4, min_samples_leaf=20, min_samples_split=150)
```

Variable Importance

```
print (pd.DataFrame(reg_dt_model.feature_importances_, columns = ["Imp"],
                    index = X_train.columns).sort_values('Imp',ascending=False))
```

```
                    Imp
Agency_Code    0.616392
Sales          0.252286
Product Name   0.077771
Commision      0.022912
Duration       0.022624
Age            0.008015
Type           0.000000
Channel        0.000000
Destination    0.000000
```

MODEL 2:

Building a Ensemble Random Forest Classifier

```
df_insured_rf=df_original.copy()
df_insured_rf.head()
```

9]:

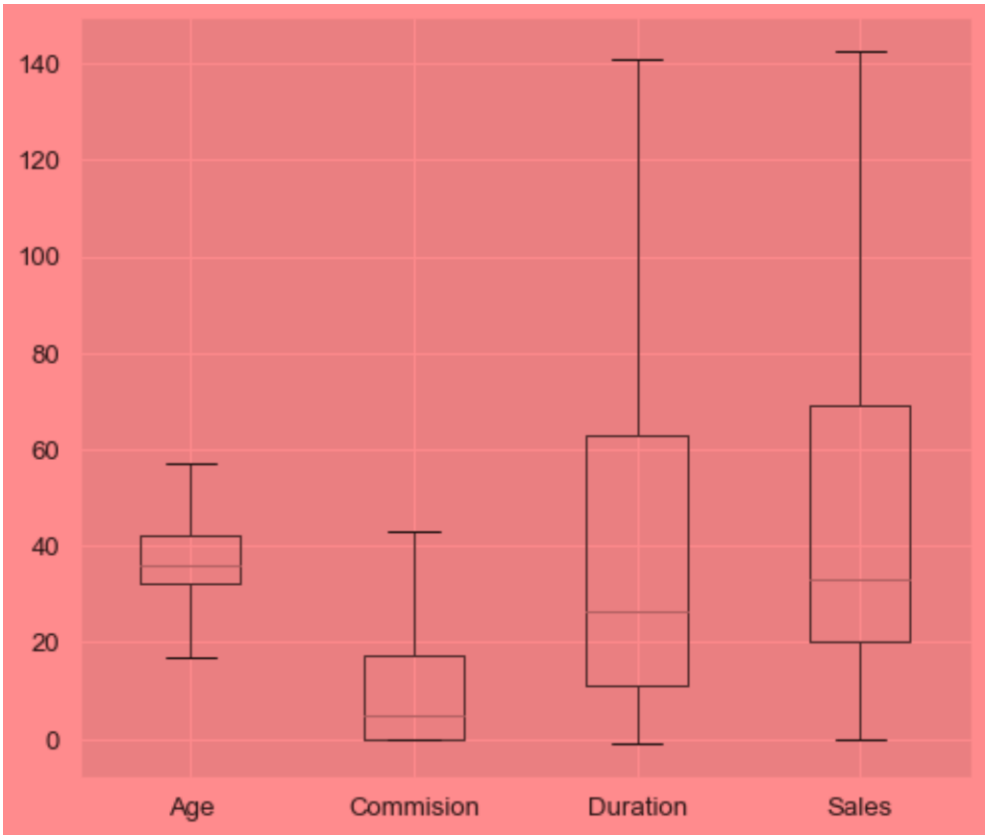
	Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
0	48	C2B	Airlines	No	0.70	Online	7	2.51	Customised Plan	ASIA
1	36	EPX	Travel Agency	No	0.00	Online	34	20.00	Customised Plan	ASIA
2	39	CWT	Travel Agency	No	5.94	Online	3	9.90	Customised Plan	Americas
3	36	EPX	Travel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

Treating Outliers from Random Forest

```
def treat_outlier(col):
    sorted(col)
    Q1,Q3=np.percentile(col,[25,75])
    IQR=Q3-Q1
    lower_range= Q1-(1.5 * IQR)
    upper_range= Q3+(1.5 * IQR)
    return lower_range, upper_range
```

```
for feature in df_insured_rf[['Age','Commision', 'Duration', 'Sales']]:
    lr,ur=treat_outlier(df_insured_rf[feature])
    df_insured_rf[feature]=np.where(df_insured_rf[feature]>ur,ur,df_insured_rf[feature])
    df_insured_rf[feature]=np.where(df_insured_rf[feature]<lr,lr,df_insured_rf[feature])
```

Boxplot to check the outliers



Random Forest Classifier

```
X_train, X_test, train_labels, test_labels = train_test_split(X_rf, y_rf, test_size=.30, random_state=1)
```

```
rfcl = RandomForestClassifier(n_estimators = 100,max_features=6,random_state=1)
rfcl = rfcl.fit(X_train, train_labels)
```

```
rfcl
```

Finding the optimal numbers using grid search

```
param_grid_rfcl = {
    'max_depth': [6],#20,30,40
    'max_features': [4],## 7,8,9
    'min_samples_leaf': [8],## 50,100
    'min_samples_split': [45], ## 60,70
    'n_estimators': [100] ## 100,200
}

rfcl = RandomForestClassifier(random_state=1)

grid_search_rfcl = GridSearchCV(estimator = rfcl, param_grid = param_grid_rfcl, cv = 10)
```

Fitting the Model To RFCL Values Obtained By Optimal Grid

Search Method

Best grid values

```
best_grid_rf
```

```
RandomForestClassifier(max_depth=6, max_features=4, min_samples_leaf=8,
                       min_samples_split=45, random_state=1)
```

Predicting on traning dataset for Random Forest

```
ytrain_predict_rf = best_grid_rf.predict(X_train)
```

```
ytest_predict_rf = best_grid_rf.predict(X_test)
```

MODEL 3

Building a neural network classifier

Before building the model

We have scale the values, to standard scale using minmaxscaler

```
sc = StandardScaler()
```

```
X_trains = sc.fit_transform(X_train)
X_tests = sc.transform (X_test)
```

After scaling the data we are transforming the same to the test data

```
X_trains = sc.fit_transform(X_train)
X_tests = sc.transform (X_test)
```

MLP classifier

```
clf = MLPClassifier(hidden_layer_sizes=100, max_iter=5000,
                    solver='sgd', verbose=True,  random_state=21,tol=0.01)
```

Training the

```
clf.fit(X_trains, train_labels)
```

```
Iteration 1, loss = 0.64244509
Iteration 2, loss = 0.62392631
Iteration 3, loss = 0.60292414
Iteration 4, loss = 0.58458220
Iteration 5, loss = 0.56914550
Iteration 6, loss = 0.55651481
Iteration 7, loss = 0.54598011
Iteration 8, loss = 0.53752961
Iteration 9, loss = 0.53051147
Iteration 10, loss = 0.52440802
Iteration 11, loss = 0.51934384
Iteration 12, loss = 0.51483466
Iteration 13, loss = 0.51108343
Iteration 14, loss = 0.50763356
Iteration 15, loss = 0.50476577
Iteration 16, loss = 0.50218466
Iteration 17, loss = 0.49989583
Iteration 18, loss = 0.49786338
Training loss did not improve more than tol=0.010000 for 10 consecutive epochs. Stopping.
```

Grid Search

```
param_grid = {
    'hidden_layer_sizes': [200], # 50, 200
    'max_iter': [2500], #5000,2500
    'solver': ['adam'], #sgd
    'tol': [0.01],
}

nncl = MLPClassifier(random_state=1)

grid_search = GridSearchCV(estimator = nncl, param_grid = param_grid, cv = 10)
```

Fitting the model using the optimal values from grid search

```
grid_search.fit(X_trains, train_labels)
```

```
GridSearchCV(cv=10, estimator=MLPClassifier(random_state=1),
             param_grid={'hidden_layer_sizes': [200], 'max_iter': [2500],
                         'solver': ['adam'], 'tol': [0.01]})
```

Best grid values,

```
best_grid_ann= grid_search.best_estimator_
best_grid_ann
```

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

Predicting on Training dataset for Neural Network Classifier

```
ytrain_predict = best_grid_ann.predict(X_trains)
ytest_predict = best_grid_ann.predict(X_tests)
```

PROBLEM 2.3

Performance Metrics: Comment and Check the performance of Predictions on Train and Test sets using Accuracy,

Resolution:

Decision tree prediction

```
ytrain_predict_dt = reg_dt_model.predict(X_train)
```

```
print('ytrain_predict',ytrain_predict_dt.shape)
```

```
ytrain_predict (2100,)
```

Accuracy

```
cart_train_acc = reg_dt_model.score(X_train,train_labels)
cart_train_acc
```

```
0.7933333333333333
```

Confusion Matrix

```
print(classification_report(train_labels, ytrain_predict_dt))
```

```
              precision    recall  f1-score   support

           0       0.84      0.87      0.85      1471
           1       0.67      0.62      0.64       629

    accuracy                           0.79      2100
   macro avg       0.75      0.74      0.75      2100
weighted avg       0.79      0.79      0.79      2100
```

```
confusion_matrix(train_labels, ytrain_predict_dt)
```

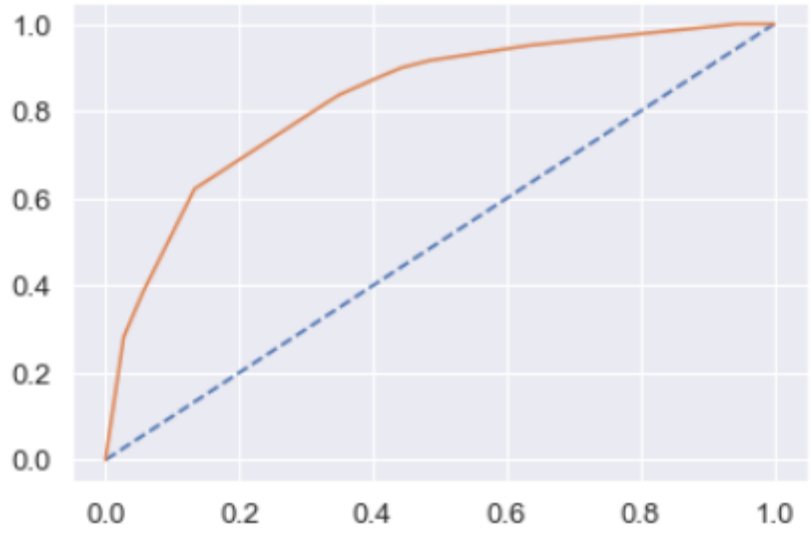
```
3]: array([[1275,  196],
           [ 238,  391]], dtype=int64)
```

Model Evaluation for Decision Tree
AUC and ROC for the training data for Decision Tree

```
# predict probabilities
probs = reg_dt_model.predict_proba(X_train)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
cart_train_auc = roc_auc_score(train_labels, probs)
print('AUC: %.3f' % cart_train_auc)
# calculate roc curve
cart_train_fpr, cart_train_tpr, cart_train_thresholds = roc_curve(train_labels, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(cart_train_fpr, cart_train_tpr)
```

```
AUC: 0.827
```

```
[<matplotlib.lines.Line2D at 0x22d36d29fc8>]
```




```
cart_metrics=classification_report(train_labels, ytrain_predict_dt,output_dict=True)
df=pd.DataFrame(cart_metrics).transpose()
cart_train_f1=round(df.loc["1"][2],2)
cart_train_recall=round(df.loc["1"][1],2)
cart_train_precision=round(df.loc["1"][0],2)
print ('cart_train_precision ',cart_train_precision)
print ('cart_train_recall ',cart_train_recall)
print ('cart_train_f1 ',cart_train_f1)
```

```
cart_train_precision  0.67
cart_train_recall  0.62
cart_train_f1  0.64
```

AUC and ROC for the test data for Decission Tree

```
# predict probabilities
probs = reg_dt_model.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
cart_test_auc = roc_auc_score(test_labels, probs)
print('AUC: %.3f' % cart_test_auc)
# calculate roc curve
cart_test_fpr, cart_test_tpr, cart_testthresholds = roc_curve(test_labels, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(cart_test_fpr, cart_test_tpr)
```

```
AUC: 0.790
```

```
cart_metrics=classification_report(test_labels, ytest_predict_dt,output_dict=True)
df=pd.DataFrame(cart_metrics).transpose()
cart_test_precision=round(df.loc["1"][0],2)
cart_test_recall=round(df.loc["1"][1],2)
cart_test_f1=round(df.loc["1"][2],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  0.71
cart_test_recall  0.53
cart_test_f1  0.6
```

Model 2 prediction random forest

```
ytrain_predict_rf = best_grid_rf.predict(X_train)
```

```
print('ytrain_predict',ytrain_predict_rf.shape)
```

```
ytrain_predict (2100,)
```

Accuracy

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

```
]:  0.8123809523809524
```

Confusion Matrix

```
print(classification_report(train_labels, ytrain_predict_rf))
```

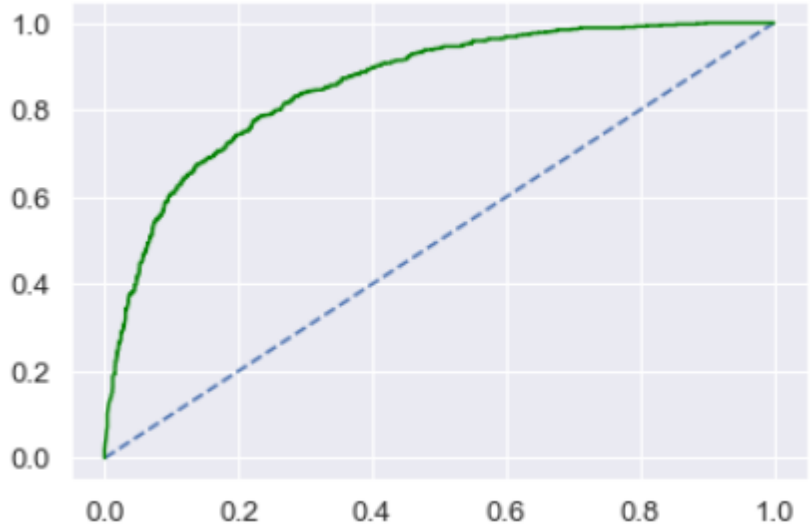
```
              precision    recall  f1-score   support

           0       0.84      0.90      0.87      1471
           1       0.73      0.60      0.66       629

    accuracy                           0.81      2100
   macro avg       0.78      0.75      0.76      2100
weighted avg       0.81      0.81      0.81      2100
```

```
confusion_matrix(train_labels, ytrain_predict_rf)
```

```
array([[1331,  140],
       [ 254,  375]], dtype=int64)
```

Model Evaluation for Random Forest
AUC and ROC for the training data for Random Forest

```
# predict probabilities
probs = best_grid_rf.predict_proba(X_train)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
rf_train_auc = roc_auc_score(train_labels, probs)
print('AUC: %.3f' % rf_train_auc)
# calculate roc curve
rf_train_fpr, rf_train_tpr, rf_train_thresholds = roc_curve(train_labels, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(rf_train_fpr, rf_train_tpr, color='green')
```




```
rf_metrics=classification_report(train_labels, ytrain_predict_rf,output_dict=True)
df=pd.DataFrame(rf_metrics).transpose()
rf_train_f1=round(df.loc["1"][2],2)
rf_train_recall=round(df.loc["1"][1],2)
rf_train_precision=round(df.loc["1"][0],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  0.73
rf_train_recall  0.6
rf_train_f1  0.66
```

Predicting on Test dataset for Random Forest

```
ytest_predict_rf = best_grid_rf.predict(X_test)
```

```
print('ytest_predict_rf',ytest_predict_rf.shape)
```

```
ytest_predict_rf (900,)
```

Accuracy

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

```
0.7733333333333333
```

Confusion Matrix

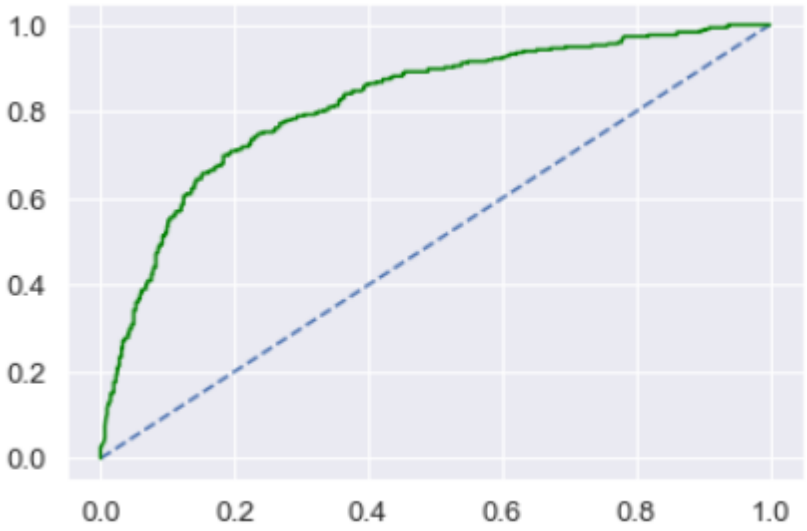
```
print(classification_report(test_labels, ytest_predict_rf))
```

```
              precision    recall  f1-score   support

           0       0.79      0.91      0.84       605
           1       0.73      0.49      0.59       295

    accuracy                           0.77       900
   macro avg       0.76      0.70      0.71       900
weighted avg       0.77      0.77      0.76       900
```

AUC and ROC for the test data for Random Forest

```
# predict probabilities
probs = best_grid_rf.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
rf_test_auc = roc_auc_score(test_labels, probs)
print('AUC: %.3f' % rf_test_auc)
# calculate roc curve
rf_test_fpr, rf_test_tpr, rf_testthresholds = roc_curve(test_labels, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(rf_test_fpr, rf_test_tpr, color='green')
```

```
AUC: 0.818
```




```
rf_metrics=classification_report(test_labels, ytest_predict_rf,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  0.73
rf_test_recall  0.49
rf_test_f1  0.59
```

MODEL 3

Predicting on Training dataset for Neural Network Classifier

```
ytrain_predict_ann = best_grid_ann.predict(X_trains)
```

```
print('ytrain_predict',ytrain_predict_ann.shape)
```

```
ytrain_predict (2100,)
```

CONFUSION MATRIX

```
confusion_matrix(train_labels,ytrain_predict_ann)
```

```
array([[1317,  154],
       [ 303,  326]], dtype=int64)
```

ACCURACY

```
ann_train_acc=best_grid_ann.score(X_trains,train_labels)
ann_train_acc
```

```
0.7823809523809524
```

```
print(classification_report(train_labels,ytrain_predict_ann))
```

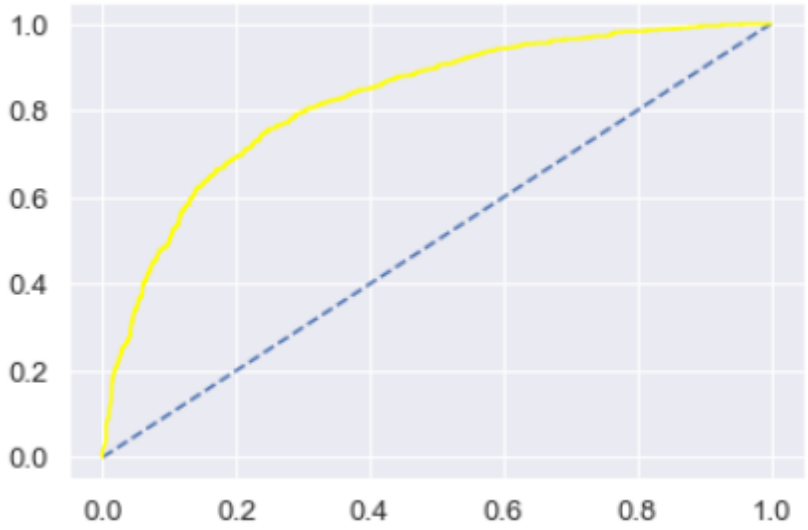
```
              precision    recall  f1-score   support

           0       0.81      0.90      0.85      1471
           1       0.68      0.52      0.59       629

    accuracy                           0.78      2100
   macro avg       0.75      0.71      0.72      2100
weighted avg       0.77      0.78      0.77      2100
```

Model evaluation for neural network classifier

AUC and ROC for the training data for Neural Network Classifier

```
# predict probabilities
probs = best_grid_ann.predict_proba(X_trains)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
ann_train_auc = roc_auc_score(train_labels, probs)
print('AUC: %.3f' % ann_train_auc)
# calculate roc curve
ann_train_fpr, ann_train_tpr, ann_train_thresholds = roc_curve(train_labels, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(ann_train_fpr, ann_train_tpr, color='yellow')
```

```
AUC: 0.823
```



Predicting on Test dataset for Neural Network Classifier

```
ytest_predict_ann = best_grid_ann.predict(X_tests)
```

```
print('ytest_predict_ann',ytest_predict_ann.shape)
```

```
ytest_predict_ann (900,)
```

Accuracy

```
ann_test_acc = best_grid_ann.score(X_tests,test_labels)
ann_test_acc
```

```
0.7622222222222222
```

Confusion Matrix

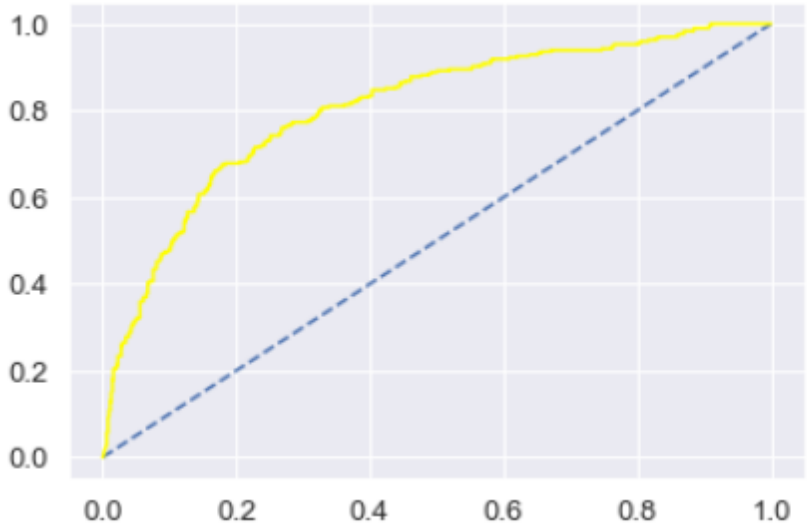
```
confusion_matrix(test_labels, ytest_predict_ann)
```

```
array([[557,  48],
       [166, 129]], dtype=int64)
```

AUC and ROC for the test data for Neural Network Classifier

```
# predict probabilities
probs = best_grid_ann.predict_proba(X_tests)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
ann_test_auc = roc_auc_score(test_labels, probs)
print('AUC: %.3f' % ann_test_auc)
# calculate roc curve
ann_test_fpr, ann_test_tpr, ann_testthresholds = roc_curve(test_labels, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(ann_test_fpr, ann_test_tpr, color='yellow')
```

```
AUC: 0.806
```

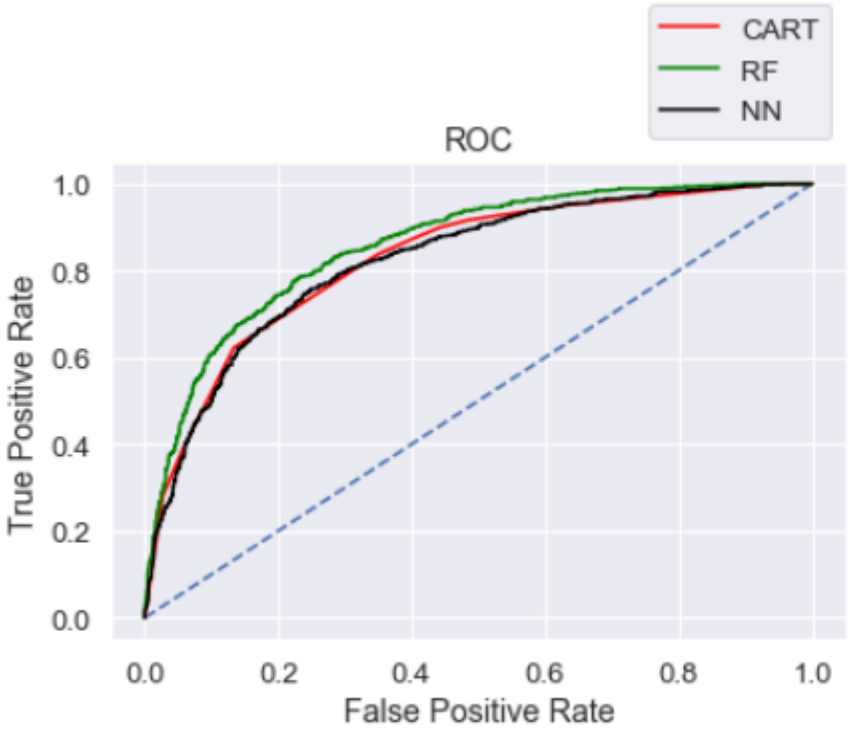


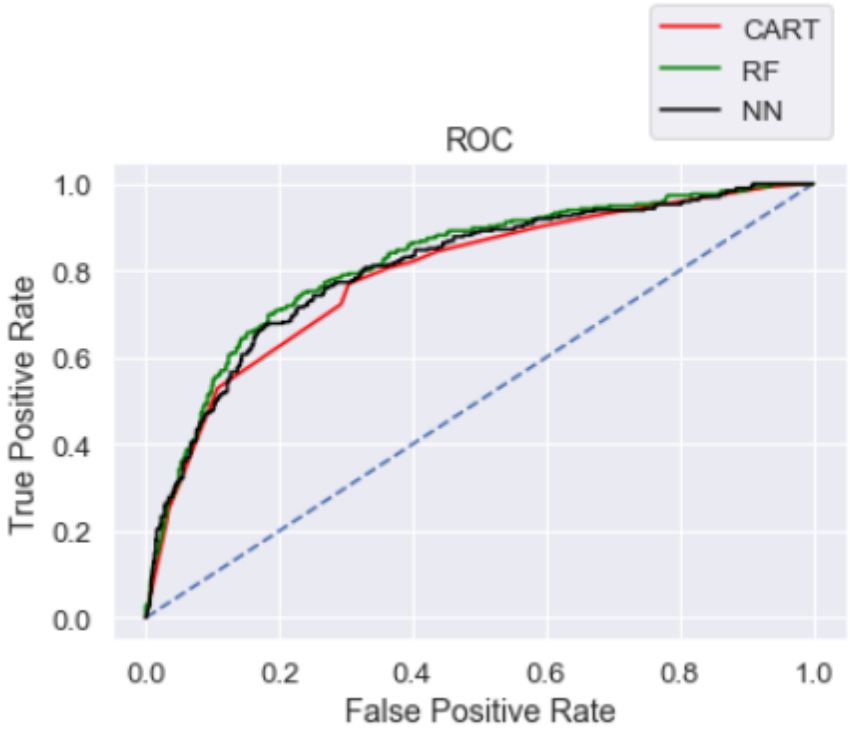
PROBLEM 2.4

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

Resolution:

	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.79	0.77	0.81	0.77	0.78	0.76
AUC	0.83	0.79	0.86	0.82	0.82	0.81
Recall	0.62	0.53	0.60	0.49	0.52	0.44
Precision	0.67	0.71	0.73	0.73	0.68	0.73
F1 Score	0.64	0.60	0.66	0.59	0.59	0.55





CONCLUSION:

Here we are selecting the RF model, as it has better accuracy, precision, recall, and f1 score better than other two CART & NN.

PROBLEM 2.5

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

Resolution:

After thoroughly analyzing the model, more data will help us understand and predict models better.

Streamlining online experiences benefitted customers which lead to an increase in conversions, and subsequently raised profits.

• 90% of the insurance is done by online channel.

• Almost all the offline business has a claimed associated.

• 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.

• 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 perform a deep dive analysis into the process to understand the workflow and why?

Key performance indicators (KPI) The KPI's of insurance claims are

• Increase customer satisfaction which in fact will give more revenue

• Combat fraud transactions; deploy measures to avoid fraudulent transactions at earliest

• Optimize claims recovery method

• Reduce claim handling costs.

The End

Thakur Arun Singh

****************************^^^**************************