Data Mining Assignment

Submitted By

Vinit Sharma

Table of Contents

Figures of 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.

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

As an initial steps, all the necessary libraries have been imported. Data based on bank marketing has read successfully. Data contains 210 rows with 7 variables. Within these 7 variables, all variable has float as dtype.

```
bank_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

```
bank_df.describe()
```

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000
mean	14.847524	14.559286	0.870999	5.628533	3.258605	3.700201	5.408071
std	2.909699	1.305959	0.023629	0.443063	0.377714	1.503557	0.491480
min	10.590000	12.410000	0.808100	4.899000	2.630000	0.765100	4.519000
25%	12.270000	13.450000	0.856900	5.262250	2.944000	2.561500	5.045000
50%	14.355000	14.320000	0.873450	5.523500	3.237000	3.599000	5.223000
75%	17.305000	15.715000	0.887775	5.979750	3.561750	4.768750	5.877000
max	21.180000	17.250000	0.918300	6.675000	4.033000	8.456000	6.550000

```
bank_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210 entries, 0 to 209
Data columns (total 7 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   spending                    210 non-null    float64
 1   advance_payments            210 non-null    float64
 2   probability_of_full_payment 210 non-null    float64
 3   current_balance             210 non-null    float64
 4   credit_limit                210 non-null    float64
 5   min_payment_amt             210 non-null    float64
 6   max_spent_in_single_shopping 210 non-null   float64
dtypes: float64(7)
memory usage: 11.6 KB
```

FIGURE 1 BANK DATASET DESCRIPTION AND INFO

There is no missing and duplicated row present in the dataset. In the description of the dataset, we have converted the five-point summary based on dataset.

```
bank_df.isnull().sum()
```

```
spending                      0
advance_payments              0
probability_of_full_payment   0
current_balance               0
credit_limit                  0
min_payment_amt               0
max_spent_in_single_shopping  0
dtype: int64
```

```
bank_df.duplicated().sum()
```

```
0
```

FIGURE 2 NULL AND DUPLICATED DATA INTO DATASET

Distplot has been plotted for all the variables. Frequency measures also complete for the dataset variables.

FIGURE 3 DISTPLOT FOR THE DATASET VARIABLES

No outlier presence identified in the dataset. Boxplot shows there is no outlier availability for the variables.

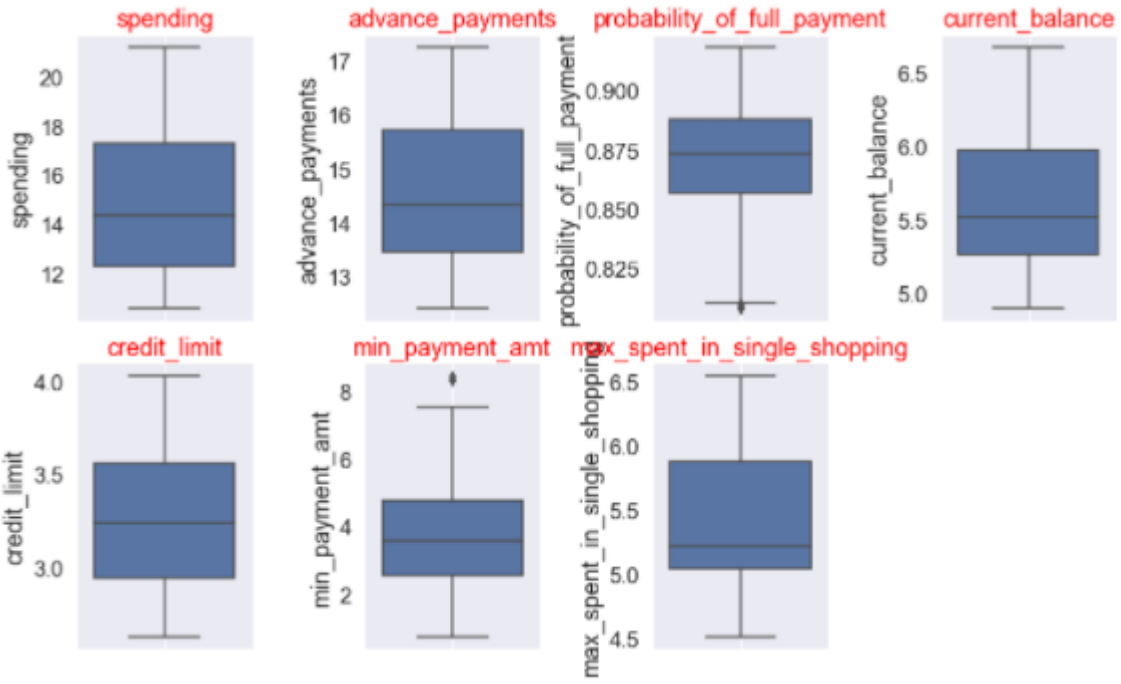


FIGURE 4 BOXPLOT FOR THE DATASET VARIABLES

Heatmap is plotted to represent the correlations among variables. Based on map, many variables are in the range of 0.80 to 1 which indicates strong relation between variables. Similarly, some negative correlated pairs also found.

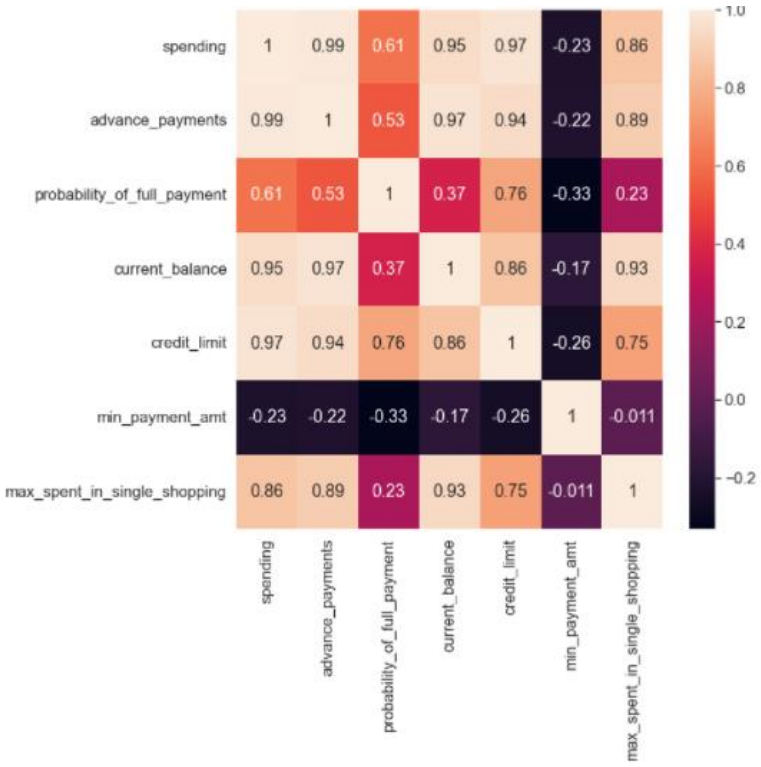


FIGURE 5 HEATMAP PLOT FOR VARIABLES

Below plot shows the pair plot between variables. In pair plot also, we can see many pairs are highly correlated and also shows linear variations.

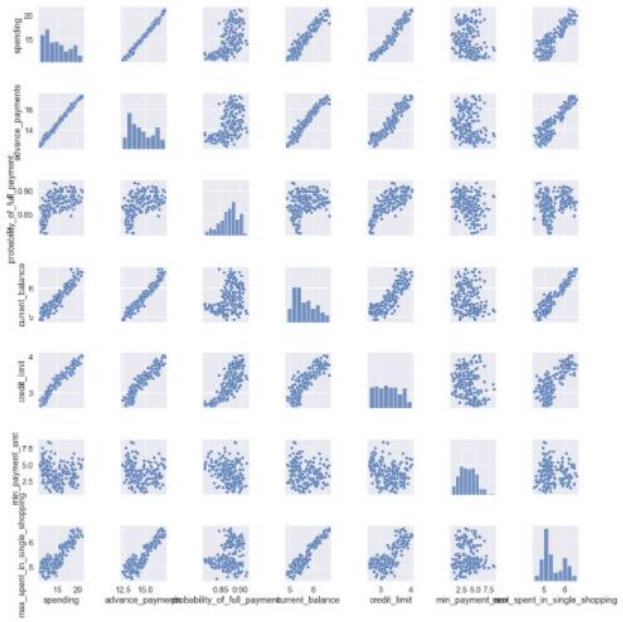


FIGURE 6 PAIRPLOT FOR DATASET

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

Scaling is needed when the values of the variables are different and varied much. As in this case study, spending, advance payments have different values and may get more weightage in analysis so to provide the equal importance to all the variables scaling is necessary. Here, we have used standard scaler to do the scaling process on dataset.

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

```
X = StandardScaler()
scaled_bank_df = X.fit_transform(bank_df)
scaled_bank_df
```

```
array([[ 1.75435461,  1.81196782,  0.17822987, ...,  1.33857863,
        -0.29880602,  2.3289982 ],
       [ 0.39358228,  0.25383997,  1.501773  , ...,  0.85823561,
        -0.24280501, -0.53858174],
       [ 1.41330028,  1.42819249,  0.50487353, ...,  1.317348  ,
        -0.22147129,  1.50910692],
       ...,
       [-0.2816364 , -0.30647202,  0.36488339, ..., -0.15287318,
        -1.3221578 , -0.83023461],
       [ 0.43836719,  0.33827054,  1.23027698, ...,  0.60081421,
        -0.95348449,  0.07123789],
       [ 0.24889256,  0.45340314, -0.77624835, ..., -0.07325831,
        -0.70681338,  0.96047321]])
```

FIGURE 7 STANDARD SCALER PROCESS

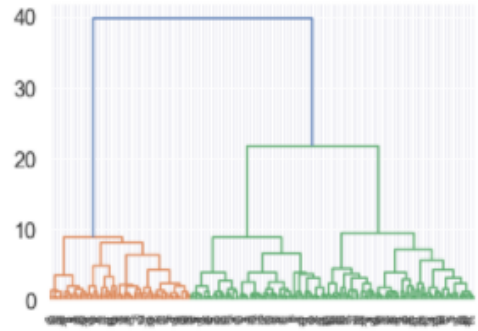
1.3 Apply hierarchical clustering to scaled data. Identify the number of optimum clusters using Dendrogram and briefly describe them

Hierarchical clustering is applied to scaled data and respective dendrograms also plotted.

We have done the cluster grouping based on the dendrogram. Hierarchical clustering analysis process provide a better result for 3 group cluster for MAX cluster based. In the other distance based analysis, 2 clusters are found.

All the three cluster has segregated into three form of clusters such as low, medium and high.

```
dend = dendrogram(wardlink)
```




```
dend = dendrogram(wardlink,
                  truncate_mode='lastp',
                  p = 10,
                  )
```

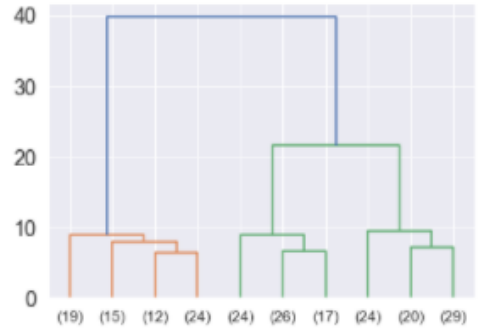


FIGURE 8 DENDROGRAM FOR DATASET

```
from scipy.cluster.hierarchy import fcluster
```

```
#Method 1
clusters = fcluster(wardlink, 3, criterion='maxclust')
clusters
```

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

```
clusters = fcluster(wardlink,23, criterion='distance')
clusters
```

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

FIGURE 9 CLUSTERING BASED ON DIFFERENT METHODS

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.

KMeans clustering is applied to scaled data and respective elbow curve also plotted.

We have done the cluster grouping based on the KMeans. KMeans clustering analysis process provide a better result for 3 group cluster based on elbow curve. This elbow shows sharp fall till 3 cluster after that not much fall has seen.

All the three cluster has segregated into three form of clusters such as low, medium and high.

```
wss=[]
for i in range(1,11):
    KM = KMeans(n_clusters=i)
    KM.fit(scaled_bank_df)
    wss.append(KM.inertia_)
```

```
plt.plot(range(1,11),wss)
```

```
[<matplotlib.lines.Line2D at 0x1e86b45f7b8>]
```

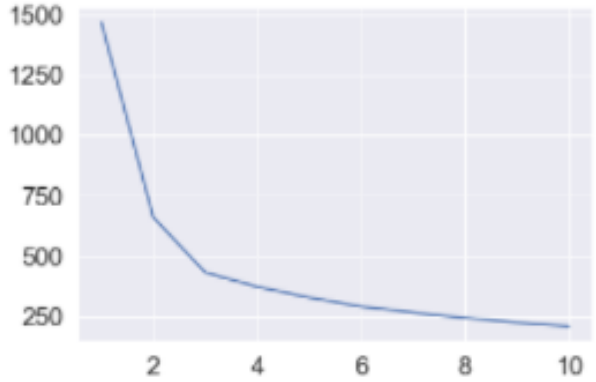


FIGURE 10 ELBOW CURVE

```
from sklearn.metrics import silhouette_samples, silhouette_score

#X,Y =make_blobs()

num_of_clusters=[2,3,4,5,6]

sl=[]

for i in num_of_clusters:
    k_means = KMeans(n_clusters = i)
    labels=k_means.fit_predict(scaled_bank_df)
    silhouette_avg=silhouette_score(scaled_bank_df,labels)
    sl.append(silhouette_avg)
    print("cluster number:",i,"silhouette average score:",silhouette_avg)
```

```
cluster number: 2 silhouette average score: 0.46577247686580914
cluster number: 3 silhouette average score: 0.40072705527512986
cluster number: 4 silhouette average score: 0.32757426605518075
cluster number: 5 silhouette average score: 0.28621461554288646
cluster number: 6 silhouette average score: 0.28617186533810457
```

```
k_means = KMeans(n_clusters = 3)
k_means.fit(scaled_bank_df)
labels = k_means.labels_
```

```
bank_df['clusters']=labels
bank_df['clusters'].unique()
```

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

FIGURE 11 SILHOUETTE SCORE AND KMEANS CLUSTERS PROCESS

```
silhouette_score(scaled_bank_df,labels)
```

```
0.40072705527512986
```

```
sil_width = silhouette_samples(scaled_bank_df,labels)
silhouette_samples(scaled_bank_df,labels).min()
```

```
0.002713089347678376
```

FIGURE 12 KMEANS CLUSTERS SILHOUETTE SCORE ON 3 GROUP KMEANS

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

3Group Clustering via Hierarchical Clustering:

clusters	1.0	2.0	3.0
spending	18.371429	11.872388	14.199041
advance_payments	16.145429	13.257015	14.233562
probability_of_full_payment	0.884400	0.848072	0.879190
current_balance	6.158171	5.238940	5.478233
credit_limit	3.684629	2.848537	3.226452
min_payment_amt	3.639157	4.949433	2.612181
max_spent_in_single_shopping	6.017371	5.122209	5.086178

FIGURE 13 3GROUP CLUSTERING VIA HIERARCHICAL CLUSTERING

3Group Clustering via KMeans Clustering:

	1	2	3
spending	18.371429	11.872388	14.199041
advance_payments	16.145429	13.257015	14.233562
probability_of_full_payment	0.884400	0.848072	0.879190
current_balance	6.158171	5.238940	5.478233
credit_limit	3.684629	2.848537	3.226452
min_payment_amt	3.639157	4.949433	2.612181
max_spent_in_single_shopping	6.017371	5.122209	5.086178

FIGURE 14 3GROUP CLUSTERING VIA KMEANS CLUSTERING

Cluster Group Profiles are as followings:

Group1: High Spending

Group 2: Medium Spending

Group 3: Low Spending

Here are some promotional strategies for each cluster:

Group 1: High Spending Group

- Increase spending limits
- Increase credit limit
- Provide more reward points
- Provide tie up privilege on luxury brands shopping which will boost one_time_maximum_spending

Group 2: Medium Spending Group

- Promote premium cards usage to boost transactions
- Such customers are good and potential to enter into high spending group members. So, we can promote the purchasing power benefit by increasing credit limit and reduce interest rate.

Group 3: Low Spending Group

- Should do the promotional activities to increase the payment rate.
- Provide the reminder for payment before due date
- Increase the spending habits by tie up with Utilities and shopping marts.

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.

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

As an initial steps, all the necessary libraries have been imported. Data based on insurance domain has read successfully. Data contains 3000 rows with 10 variables. Within these 10 variables, 6 has object dtype and rest are int or float.

	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

FIGURE 15 INSURANCE DATASET

Below figures shows the value count of each categorical variables. Based on the data, most preferred channel is Online and destination is Asia.

```
<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
```

```
AGENCY_CODE :  4
JZI      239
CWT      472
C2B      924
EPX     1365
Name: Agency_Code, dtype: int64

TYPE :  2
Airlines          1163
Travel Agency     1837
Name: Type, dtype: int64

CLAIMED :  2
Yes      924
No      2076
Name: Claimed, dtype: int64

CHANNEL :  2
Offline       46
Online      2954
Name: Channel, dtype: int64

PRODUCT NAME :  5
Gold Plan              109
Silver Plan            427
Bronze Plan            650
Cancellation Plan      678
Customised Plan       1136
Name: Product Name, dtype: int64

DESTINATION :  3
EUROPE        215
Americas      320
ASIA         2465
Name: Destination, dtype: int64
```

Data distribution of variable such as age, commission, duration and sales are positive skewed. For Age, most of the data lies in range of 25 to 50. Commission data lies in range of 0 to 100. Similarly, duration distribution in 0 to 800 approx. Sales distribution is in range of 0 to 300.

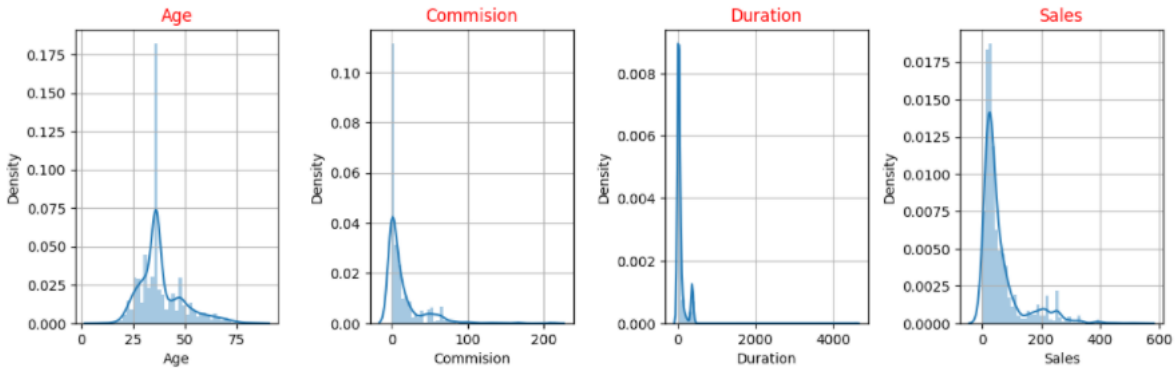


FIGURE 16 DISTPLOT FOR INSURANCE DATASET

Based on count plots, LPX is most preferred Agency Code. Similarly, travel agency is more used type. In many cases, insurance claim are not done.

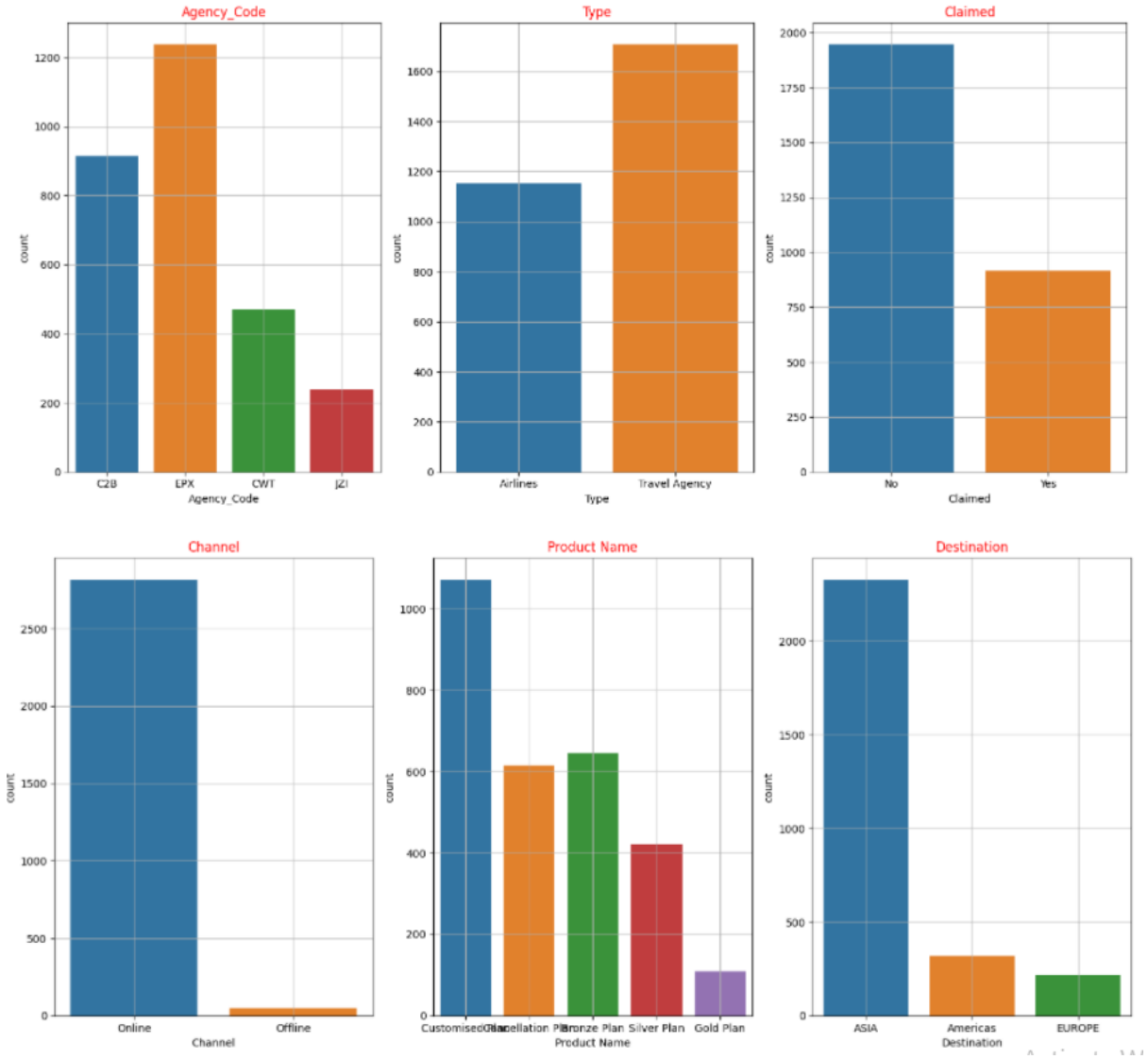
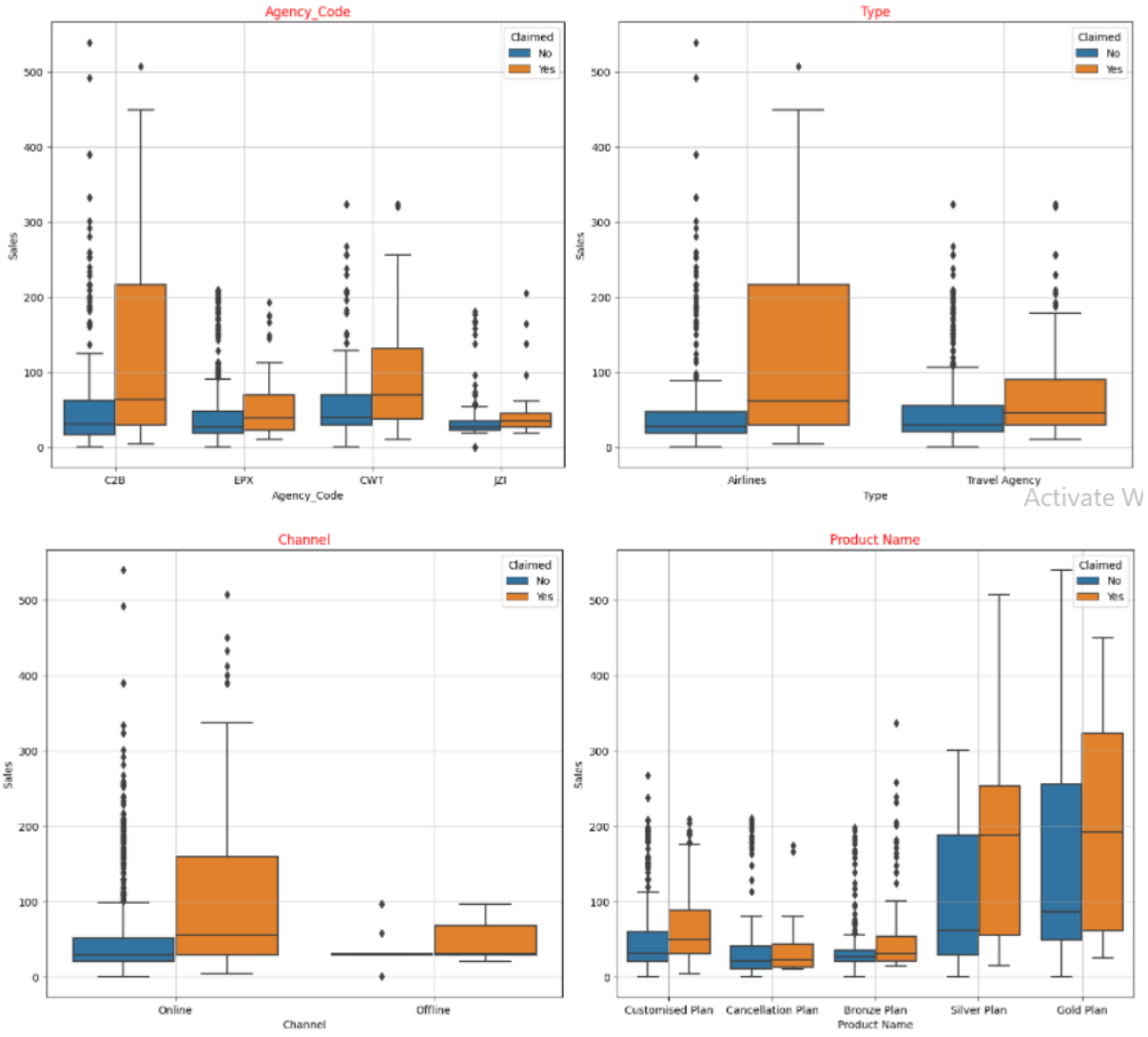


FIGURE 17 COUNT PLOT FOR INSURANCE DATASET

Online is most suitable channel. Asia is preferred destination. Customized plan is used in many of the cases.

Below figure shows the boxplot representation of Agency Code, type, destination, channel and product name with respect to sales points. Here, we have considered claimed reference hue.



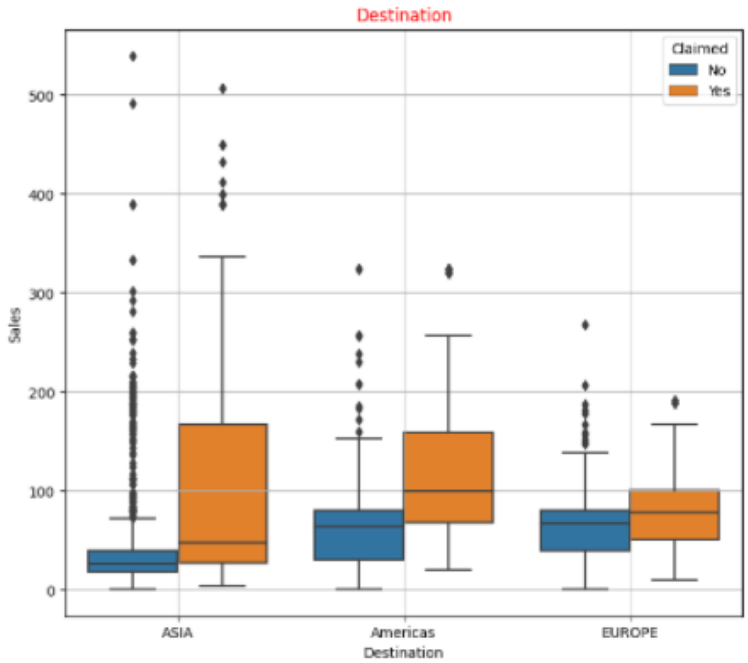


FIGURE 18 BOXPLOT FOR INSURANCE DATASET

Based on heatmap, Sales and Duration are highly correlated.

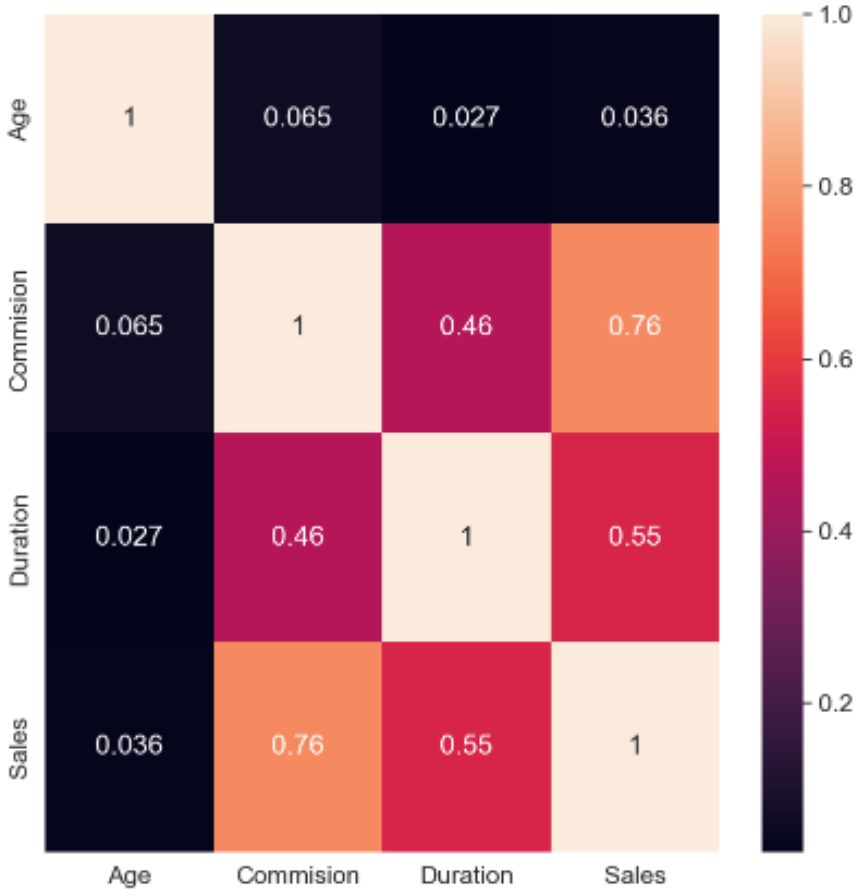


FIGURE 19 HEATMAP FOR INSURANCE DATASET

Below pair plot shows the correlations between the variables. Plot reflects weak correlation among the variables.

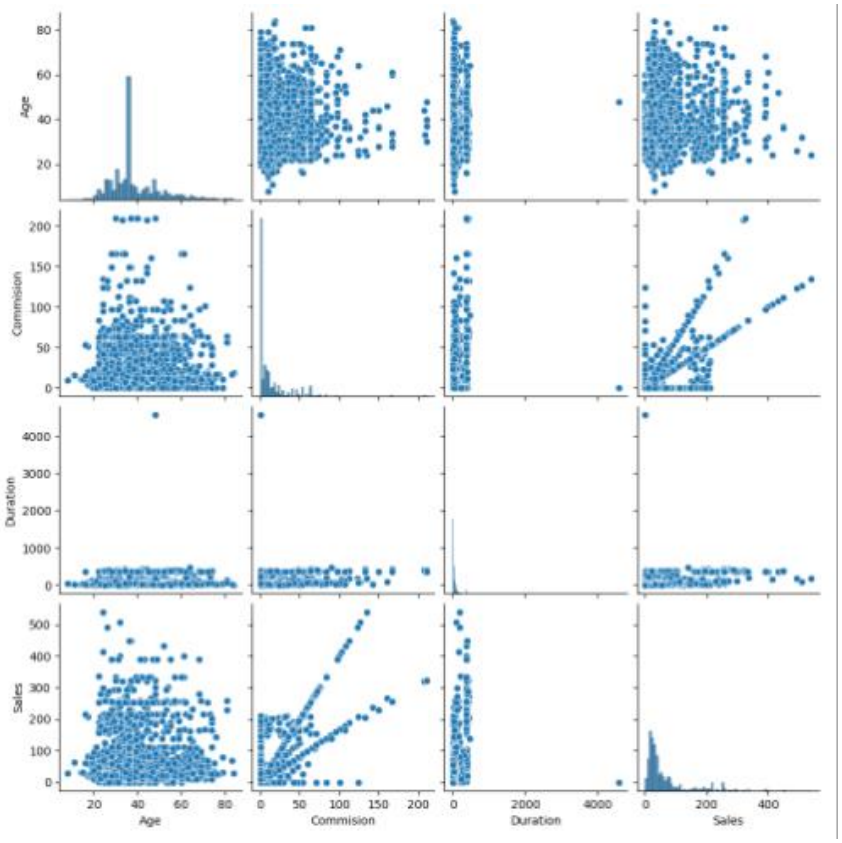


FIGURE 20 PAIR PLOT FOR INSURANCE DATASET

Based on analysis, total 139 rows come duplicate. We can treat the duplicate rows before move further and reshape the data.

Check for duplicate data

```
# Are there any duplicates ?
dups = df.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
```

```
Number of duplicate rows = 139
```

FIGURE 21 DUPLICATE DATA

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

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

```
X = df.drop("Claimed", axis=1)

y = df.pop("Claimed")

X.head()
```

	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

Splitting data into training and test set

```
from sklearn.model_selection import train_test_split

X_train, X_test, train_labels, test_labels = train_test_split(X, y, test_size=.30, random_state=1)
```

Checking the dimensions of the training and test data

```
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 (2002, 9)
X_test (859, 9)
train_labels (2002,)
test_labels (859,)
```

FIGURE 22 TRAIN AND TEST SPLIT FROM DATASET

Dataset is split into 70:30 wise as train and test data. We have hyper-tuned the parameters with GridSearchCV for building decision tree classifier.

Building a Decision Tree Classifier

```
param_grid = {
    '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 = GridSearchCV(estimator = dtcl, param_grid = param_grid, cv = 10)
```

```
grid_search.fit(X_train, train_labels)
print(grid_search.best_params_)
best_grid = grid_search.best_estimator_
best_grid
#{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 450}
```

```
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 300}

DecisionTreeClassifier(max_depth=10, min_samples_leaf=50, min_samples_split=300,
                       random_state=1)
```

FIGURE 23 DECISION TREE CLASSIFIER MODEL BUILDING

Based on importance of variable, Agency Code is most important variable. Type, channel, and destination are less important variables.

Variable Importance

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

	Imp
Agency_Code	0.600450
Sales	0.304966
Product Name	0.047357
Duration	0.018764
Commision	0.014732
Age	0.013731
Type	0.000000
Channel	0.000000
Destination	0.000000

FIGURE 24 VARIABLE IMPORTANCE BASED ON DECISION TREE CLASSIFIER MODEL BUILDING

Model 2:

Hyper-tuned the parameters with GridSearchCV for building Random Forest classifier.

Building a Random Forest Classifier

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

```
param_grid = {
    'max_depth': [10],## 20,30,40
    'max_features': [6],## 7,8,9
    'min_samples_leaf': [10],## 50,100
    'min_samples_split': [50], ## 60,70
    'n_estimators': [300] ## 100,200
}

rfcl = RandomForestClassifier(random_state=1)

grid_search = GridSearchCV(estimator = rfcl, param_grid = param_grid, cv = 5)
```

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

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=1),
             param_grid={'max_depth': [10], 'max_features': [6],
                         'min_samples_leaf': [10], 'min_samples_split': [50],
                         'n_estimators': [300]})
```

```
grid_search.best_params_
```

```
{'max_depth': 10,
 'max_features': 6,
 'min_samples_leaf': 10,
 'min_samples_split': 50,
 'n_estimators': 300}
```

```
best_grid = grid_search.best_estimator_
```

```
best_grid
```

```
RandomForestClassifier(max_depth=10, max_features=6, min_samples_leaf=10,
                       min_samples_split=50, n_estimators=300, random_state=1)
```

FIGURE 25 RANDOM FOREST CLASSIFIER MODEL BUILDING

Based on importance of variable, Agency Code is most important variable. Channel is less important variables.

```
# Variable Importance
print (pd.DataFrame(best_grid.feature_importances_, columns = ["Imp"], index = X_train.columns).sort_values('Imp',ascending=Fals
```

	Imp
Agency_Code	0.325075
Sales	0.207197
Product Name	0.169962
Duration	0.101869
Commision	0.095711
Age	0.069549
Type	0.015744
Destination	0.013474
Channel	0.001419

FIGURE 26 VARIABLE IMPORTANCE FOR RF CLASSIFIER MODEL BUILDING

Model 3:

Hyper-tuned the parameters with GridSearchCV for building a Neural Network classifier.

Building a Neural Network Classifier

```
param_grid = {
    'hidden_layer_sizes': [50,100], # 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)
```

```
grid_search.fit(X_train, train_labels)
grid_search.best_params_
#{'hidden_layer_sizes': 100, 'max_iter': 2500, 'solver': 'adam', 'tol': 0.01}
```

```
{'hidden_layer_sizes': 100, 'max_iter': 2500, 'solver': 'adam', 'tol': 0.01}
```

```
best_grid = grid_search.best_estimator_
best_grid
```

```
MLPClassifier(hidden_layer_sizes=100, max_iter=2500, random_state=1, tol=0.01)
```

Predicting the Training and Testing data

```
ytrain_predict = best_grid.predict(X_train)
ytest_predict = best_grid.predict(X_test)
```

FIGURE 27 NN CLASSIFIER MODEL BUILDING

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.

For CART Model:

Train Data:

Confusion Matrix for the training data

```
confusion_matrix(train_labels, ytrain_predict)
```

```
array([[1157,  202],
       [ 270,  373]], dtype=int64)
```

```
#Train Data Accuracy
cart_train_acc=best_grid.score(X_train,train_labels)
cart_train_acc
```

```
0.7642357642357642
```

```
print(classification_report(train_labels, ytrain_predict))
```

```
              precision    recall  f1-score   support

           0       0.81      0.85      0.83      1359
           1       0.65      0.58      0.61       643

    accuracy                           0.76      2002
   macro avg       0.73      0.72      0.72      2002
weighted avg       0.76      0.76      0.76      2002
```

```
cart_metrics=classification_report(train_labels, ytrain_predict,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  0.65
cart_train_recall  0.58
cart_train_f1  0.61
```

AUC and ROC for the training data

```
# predict probabilities
probs = best_grid.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)
plt.show()
```

```
AUC: 0.810
```

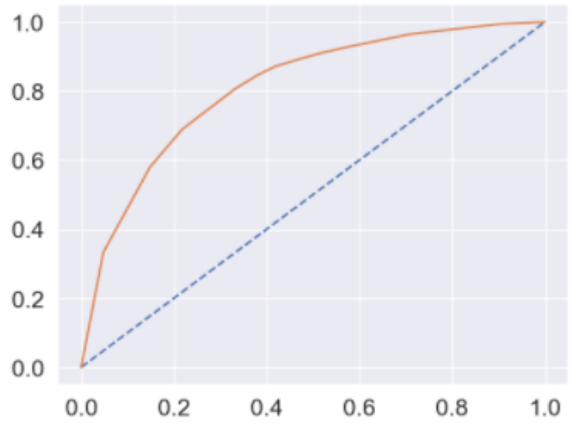


FIGURE 28 CONFUSION MATRIX, CLASSIFICATION, AUC AND ROC CURVE DATA FOR TRAINSET IN CART MODEL

Test Data:

Confusion Matrix for test data

```
confusion_matrix(test_labels, ytest_predict)
```

```
array([[510,  78],
       [109, 162]], dtype=int64)
```

```
#Test Data Accuracy
cart_test_acc=best_grid.score(X_test,test_labels)
cart_test_acc
```

```
0.7823050058207218
```

```
print(classification_report(test_labels, ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.82      0.87      0.85       588
           1       0.68      0.60      0.63       271

    accuracy                           0.78       859
   macro avg       0.75      0.73      0.74       859
weighted avg       0.78      0.78      0.78       859
```

```
cart_metrics=classification_report(test_labels, ytest_predict,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  0.68
cart_test_recall  0.6
cart_test_f1  0.63
```

AUC and ROC for the test data

```
# predict probabilities
probs = best_grid.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)
plt.show()
```

```
AUC: 0.792
```

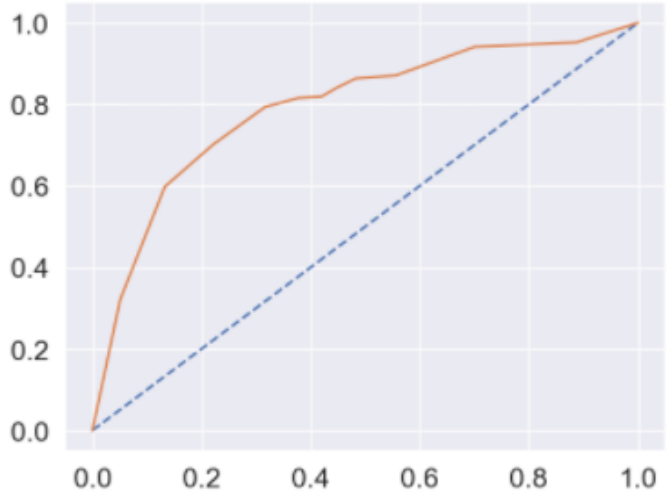


FIGURE 29 CONFUSION, CLASSIFICATION MATRIX, AUC AND ROC CURVE DATA FOR TEST SET IN CART MODEL

For Random Forest Model:

Train Data:

RF Model Performance Evaluation on Training data

```
confusion_matrix(train_labels,ytrain_predict)
```

```
array([[1228,  131],
       [ 258,  385]], dtype=int64)
```

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

```
0.8056943056943057
```

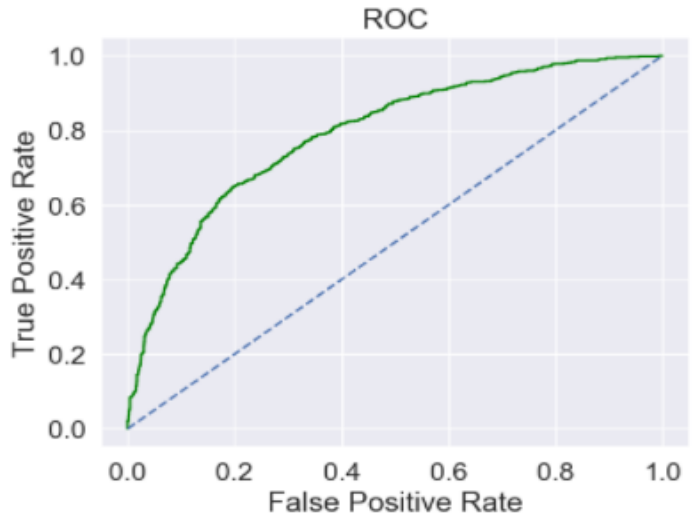
```
print(classification_report(train_labels,ytrain_predict))
```

```
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1359
           1       0.75      0.60      0.66       643

    accuracy                           0.81      2002
   macro avg       0.79      0.75      0.76      2002
weighted avg       0.80      0.81      0.80      2002
```

```
rf_metrics=classification_report(train_labels, ytrain_predict,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  0.75
rf_train_recall  0.6
rf_train_f1  0.66
```

```
rf_train_fpr, rf_train_tpr,_=roc_curve(train_labels,best_grid.predict_proba(X_train)[:,1])
plt.plot(rf_train_fpr,rf_train_tpr,color='green')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
plt.show()
rf_train_auc=roc_auc_score(train_labels,best_grid.predict_proba(X_train)[:,1])
print('Area under Curve is', rf_train_auc)
```




```
Area under Curve is 0.7920464571767961
```

FIGURE 30 CONFUSION MATRIX, CLASSIFICATION MATRIX, AUC AND ROC CURVE DATA FOR TRAINSET IN RF MODEL

Test Data:

RF Model Performance Evaluation on Test data

```
confusion_matrix(test_labels,ytest_predict)
```

```
array([[522,  66],
       [118, 153]], dtype=int64)
```

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

```
0.7857974388824214
```

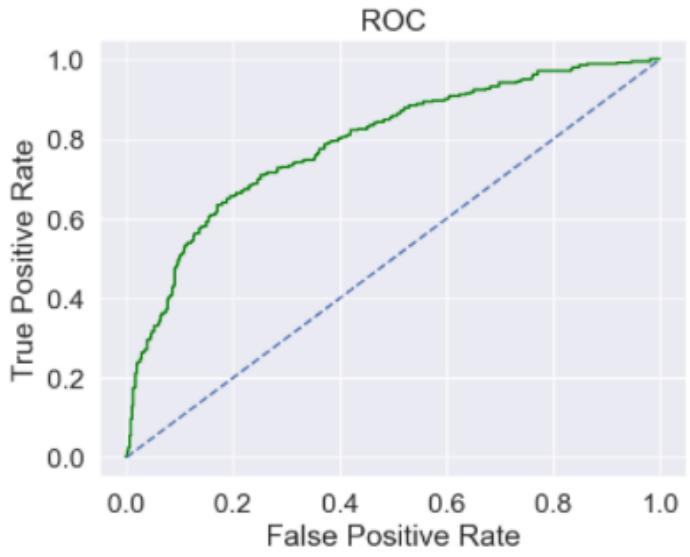
```
print(classification_report(test_labels,ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.82      0.89      0.85       588
           1       0.70      0.56      0.62       271

    accuracy                           0.79       859
   macro avg       0.76      0.73      0.74       859
weighted avg       0.78      0.79      0.78       859
```

```
rf_metrics=classification_report(test_labels, ytest_predict,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  0.7
rf_test_recall  0.56
rf_test_f1  0.62
```

```
rf_test_fpr, rf_test_tpr,_=roc_curve(test_labels,best_grid.predict_proba(X_test)[:,1])
plt.plot(rf_test_fpr,rf_test_tpr,color='green')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
plt.show()
rf_test_auc=roc_auc_score(test_labels,best_grid.predict_proba(X_test)[:,1])
print('Area under Curve is', rf_test_auc)
```




```
Area under Curve is 0.7911175540326832
```

FIGURE 31 CONFUSION MATRIX, CLASSIFICATION MATRIX, AUC AND ROC CURVE DATA FOR TEST SET IN RF MODEL

For Neural Network Model:

Train Data:

NN Model Performance Evaluation on Training data

```
confusion_matrix(train_labels,ytrain_predict)
```

```
array([[1163,  196],
       [ 280,  363]], dtype=int64)
```

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

```
0.7622377622377622
```

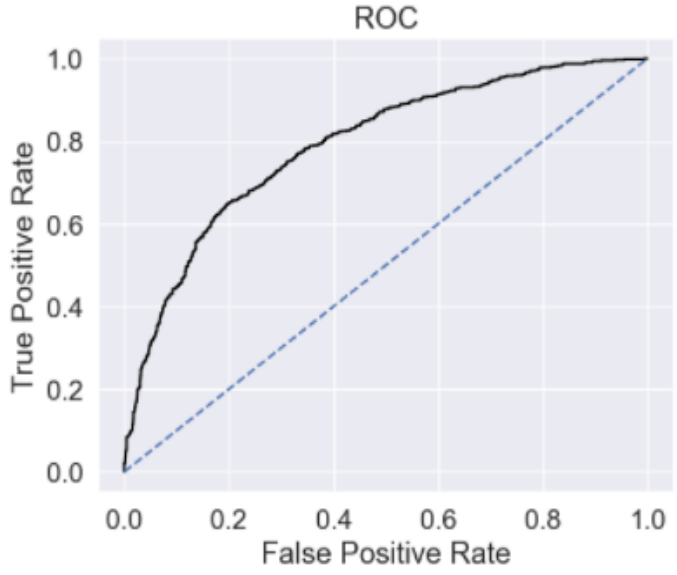
```
print(classification_report(train_labels,ytrain_predict))
```

```
              precision    recall  f1-score   support

           0       0.81      0.86      0.83      1359
           1       0.65      0.56      0.60       643

    accuracy                           0.76      2002
   macro avg       0.73      0.71      0.72      2002
weighted avg       0.76      0.76      0.76      2002
```

```
nn_metrics=classification_report(train_labels, ytrain_predict,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  0.65
nn_train_recall  0.56
nn_train_f1  0.6
```

```
nn_train_fpr, nn_train_tpr,_=roc_curve(train_labels,best_grid.predict_proba(X_train)[:,1])
plt.plot(nn_train_fpr,nn_train_tpr,color='black')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
plt.show()
nn_train_auc=roc_auc_score(train_labels,best_grid.predict_proba(X_train)[:,1])
print('Area under Curve is', nn_train_auc)
```




```
Area under Curve is 0.7920464571767961
```

FIGURE 32 CONFUSION MATRIX, CLASSIFICATION MATRIX, AUC AND ROC CURVE DATA FOR TRAIN SET IN NN MODEL

Test Data:

NN Model Performance Evaluation on Test data

```
confusion_matrix(test_labels,ytest_predict)
```

```
array([[510,  78],
       [118, 153]], dtype=int64)
```

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

```
0.7718277066356228
```

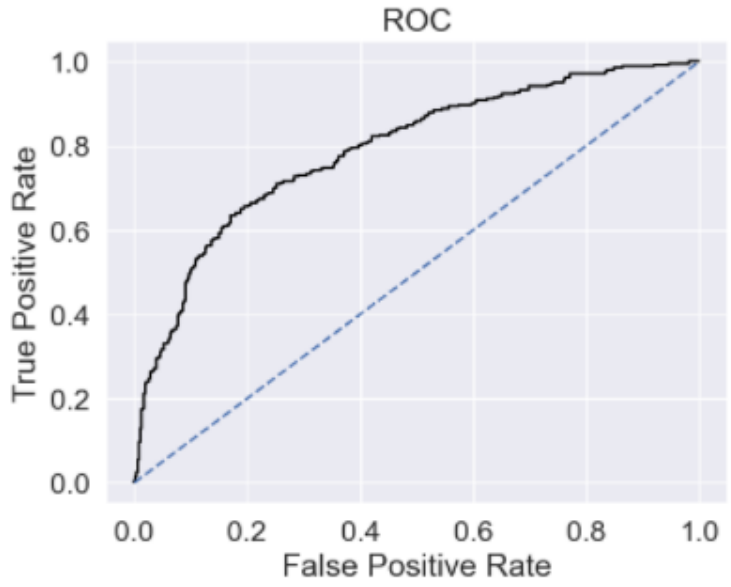
```
print(classification_report(test_labels,ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.81      0.87      0.84       588
           1       0.66      0.56      0.61       271

    accuracy                           0.77       859
   macro avg       0.74      0.72      0.72       859
weighted avg       0.76      0.77      0.77       859
```

```
nn_metrics=classification_report(test_labels, ytest_predict,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  0.66
nn_test_recall  0.56
nn_test_f1  0.61
```

```
nn_test_fpr, nn_test_tpr,_=roc_curve(test_labels,best_grid.predict_proba(X_test)[:,1])
plt.plot(nn_test_fpr,nn_test_tpr,color='black')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
plt.show()
nn_test_auc=roc_auc_score(test_labels,best_grid.predict_proba(X_test)[:,1])
print('Area under Curve is', nn_test_auc)
```




```
Area under Curve is 0.7911175540326832
```

FIGURE 33 CONFUSION MATRIX, CLASSIFICATION MATRIX, AUC AND ROC CURVE DATA FOR TEST SET IN NN MODEL

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

	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.76	0.78	0.81	0.79	0.76	0.77
AUC	0.81	0.79	0.86	0.81	0.79	0.79
Recall	0.58	0.60	0.60	0.56	0.56	0.56
Precision	0.65	0.68	0.75	0.70	0.65	0.66
F1 Score	0.61	0.63	0.66	0.62	0.60	0.61

FIGURE 34 MODEL COMPARISON TABLE

ROC Curve for the 3 models on the Training data

```
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(cart_train_fpr, cart_train_tpr,color='red',label="CART")
plt.plot(rf_train_fpr,rf_train_tpr,color='green',label="RF")
plt.plot(nn_train_fpr,nn_train_tpr,color='black',label="NN")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
plt.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc='lower right')
plt.show()
```

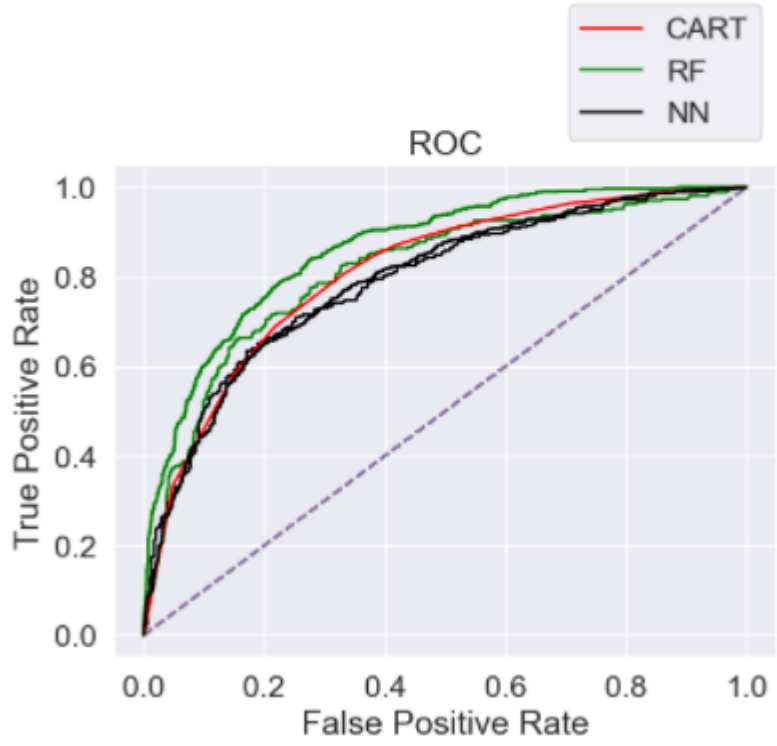


FIGURE 35 ROC CURVE FOR 3 MODELS ON TRAINING DATA

ROC Curve for the 3 models on the Test data

```
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(cart_test_fpr, cart_test_tpr,color='red',label="CART")
plt.plot(rf_test_fpr,rf_test_tpr,color='green',label="RF")
plt.plot(nn_test_fpr,nn_test_tpr,color='black',label="NN")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
plt.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc='lower right')
plt.show()
```

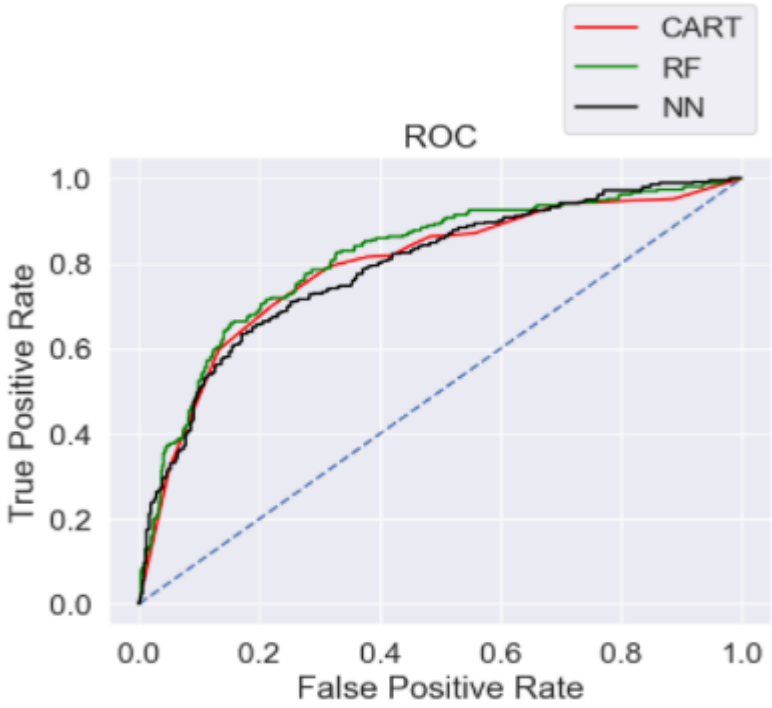


FIGURE 36 ROC CURVE FOR 3 MODELS ON TEST DATA

Based on Analysis of all the building model, we have concluded that RF model has better accuracy, precision, recall and f1 score than other two model such as CART and ANN.

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

Based on recent analysis on the model, we can suggest that more data will contribute better to predict the model further. Data suggests 90% of insurance claimed using online channel. We have to train JZI agency resources to pick up sales as per there recent status is low and need to run promotional marketing campaign. Model are able to achieve 80% accuracy so customer travel info, and tickets patterns can helpful for insurance provide business. Sales happens more with Agency than Airlines, but more claims are registered through Airline. Some important point can be consider such as deduction in claim handling costs, more align towards customer satisfaction, and avoid fraudulent transactions at earliest.