Problem Statement:

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.

Data Dictionary for Market Segmentation:

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)
7. Destination of the tour (Destination)
8. Amount of sales of tour insurance policies (Sales)
9. The commission received for tour insurance firm (Commission)
10. Age of insured (Age)

Data Ingestion: Read the dataset. Do the descriptive statistics and do null value condition check, write an inference on it. Perform Exploratory Data Analysis.

```
df = pd.read_csv('insurance_part2_data.csv')
df.head()
```

	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

- The given dataset has 3000 rows and 10columns.
- The columns are namely:
    i) Age
    ii) Agency code
    iii) Insurance type
    iv) Claimed
    v) Commission
    vi) Channel
    vii) Duration
    viii) Sales
    ix) Product Name
    x) Destination

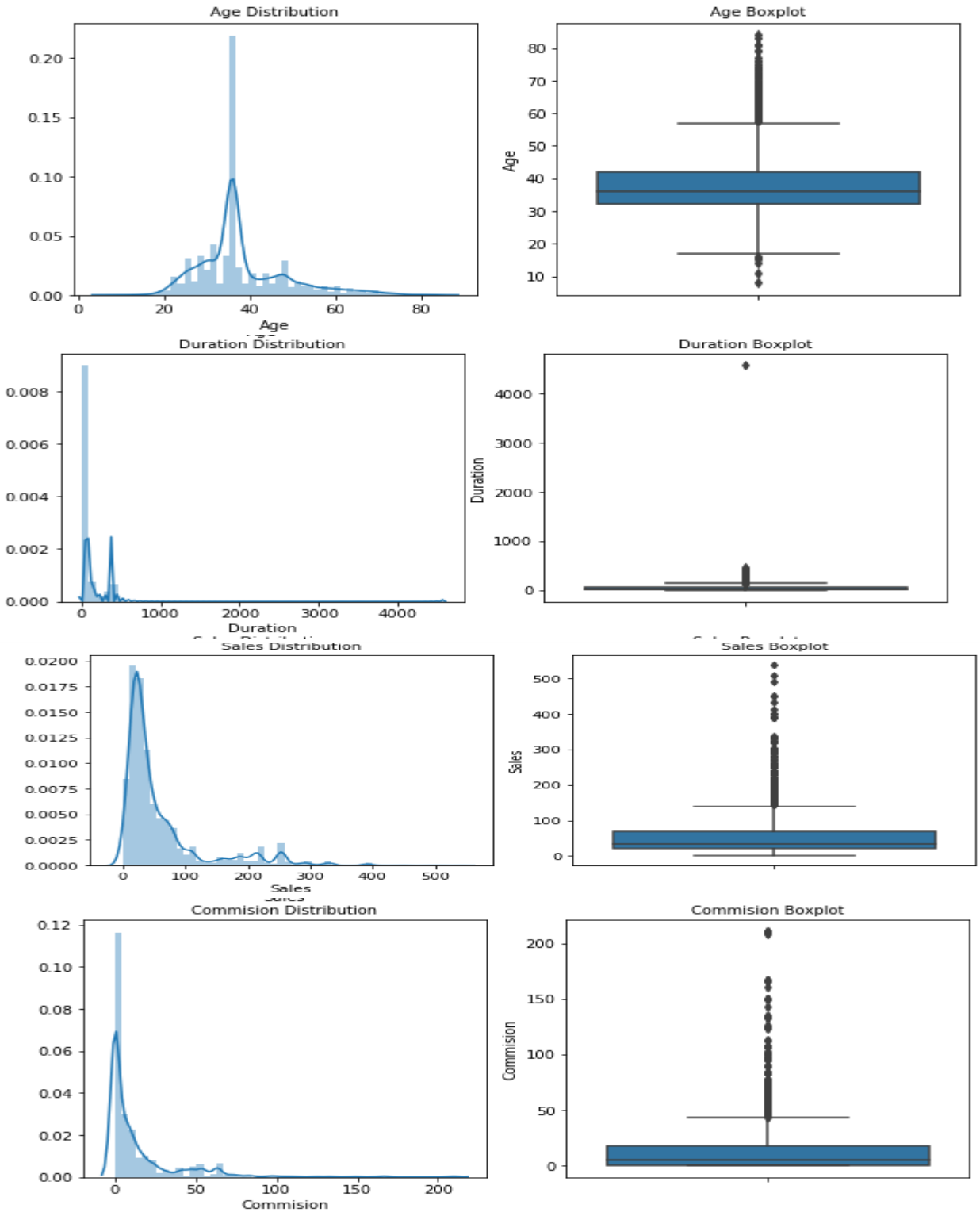
- The variables data type is 2 float, 2 integer and 6 object data types
- There are no null values
- There are no missing values
- There are 139 duplicated values
- Visible outliers are present

Describing all D-types Variables

```
df.describe(include='all').T
```

	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
Commision	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	Customised Plan	1136	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Destination	3000	3	ASIA	2465	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Univariate Analysis



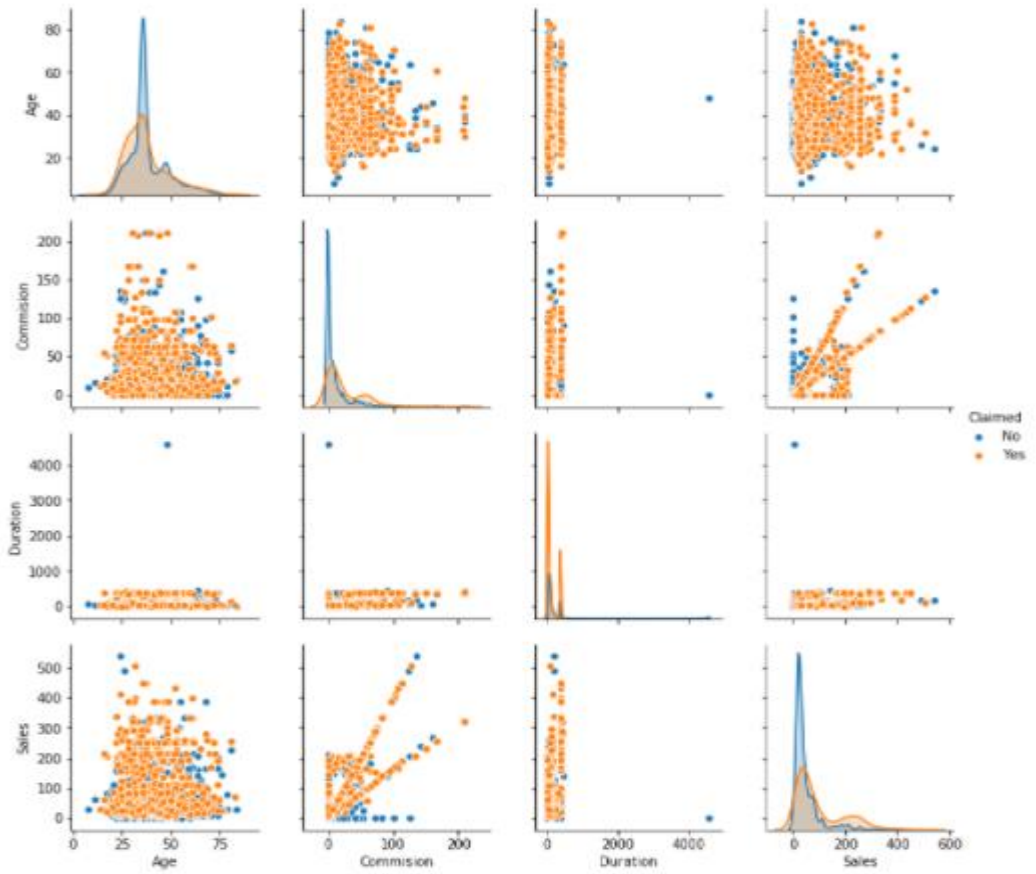
Inference:

The box plots indicate that all the continuous variables have outliers.
The box plot of 'Duration' has an extreme value of above 4000.

Multivariate Analysis

```
1 sns.pairplot(df, hue = 'Claimed' )
```

```
<seaborn.axisgrid.PairGrid at 0x3855b15ba8>
```



Inference: We can infer that the dataset is unbalanced. The class imbalance problem is evident in the plots above.

Data Pre-processing

```
def remove_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
```

```
lr,ur=remove_outlier(df["Age"])
df["Age"]=np.where(df["Age"]>ur,ur,df["Age"])
df["Age"]=np.where(df["Age"]<lr,lr,df["Age"])
lr,ur=remove_outlier(df["Commision"])
df["Commision"]=np.where(df["Commision"]>ur,ur,df["Commision"])
df["Commision"]=np.where(df["Commision"]<lr,lr,df["Commision"])
lr,ur=remove_outlier(df["Duration"])
df["Duration"]=np.where(df["Duration"]>ur,ur,df["Duration"])
df["Duration"]=np.where(df["Duration"]<lr,lr,df["Duration"])
lr,ur=remove_outlier(df["Sales"])
df["Sales"]=np.where(df["Sales"]>ur,ur,df["Sales"])
df["Sales"]=np.where(df["Sales"]<lr,lr,df["Sales"])
```

Inferences:

As from the Univariate analysis we found Outliers in all the Continuous variables so from the above code we have treated the Outliers, by capping the less than lower range values(Q1-(1.5 * IQR)) as lower range itself and greater than upper range values(Q3+(1.5 * IQR)) as upper range. So instead of deleting the outlier we are saving the information by this method.

Categorical Variables Value Count

AGENCY_CODE		
Description	Codes	Value Counts
JZI	3	239
CWT	1	471
C2B	0	913
EPX	2	1238

TYPE		
Description	Codes	Value Counts
Airlines	0	1152
Travel Agency	1	1709

CHANNEL		
Description	Codes	Value Counts
Offline	0	46
Online	1	2815

PRODUCT NAME		
Description	Codes	Value Counts
Gold Plan	3	109
Silver Plan	4	421
Cancellation Plan	1	615
Bronze Plan	0	645
Customised Plan	2	1071

DESTINATION		
Description	Codes	Value Counts
EUROPE	2	215
Americas	1	319
ASIA	0	2327

Conversion of Object type data into integers:

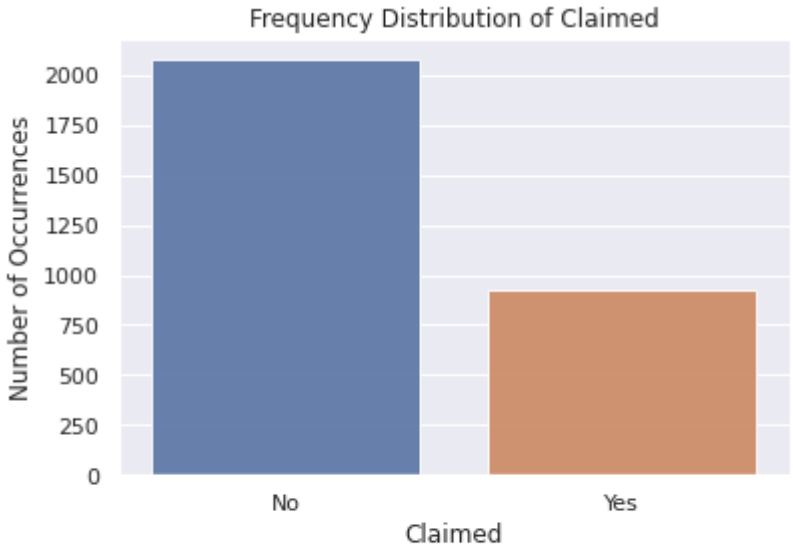
Data mining algorithms in Python can take only numerical columns. It cannot take string / object types. All columns with data type object will need to be changed for both Independent Variable (IV) and Dependent Variable (DV) into integer.

```
df['Product Name']=df['Product Name'].astype('category')
df['Claimed']=df['Claimed'].astype('category')
df['Product Name_cat']=df['Product Name'].cat.codes
df['Claimed_cat']=df['Claimed'].cat.codes
df=df.drop('Claimed',axis=1)
```

```
df= pd.get_dummies(df,drop_first=True)
df.head(2)
```

Claimed and Product name features are encode with **Label Encoding** method
Other Categorical variables are handled using **One Hot Encoding** method

Data Split: Split the data into test (30% of the data) and train (70% of the data), build classification model CART, Random Forest, Artificial Neural Network




```
No     69.2
Yes    30.8
Name: Claimed, dtype: float64
```

As per the problem statement 'CLAIMED' feature is our dependent variable. From the above Bar Graph we can find that the proportion of the two classes in the claimed feature, so there is small amount of class imbalance. For handling the imbalance we can implement some techniques like SMOTE, etc. But here we are not treating the class imbalance and we are proceeding further for the model building part.

Splitting the data into Training Set and Test Set

```
X = df1.drop("Claimed_cat" , axis=1)

y = df1.Claimed_cat
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30 , random_state=10,stratify = y)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2100, 16)
(900, 16)
(2100,)
(900,)
```

For model building we need to split the data into Train and Test.Before splitting the data into Train and Test we need to separate Dependent variable and Independent variable and store in y and x data frames respectively, these steps remain common for all three models

For our model we are splitting the data into 70 % Train and 30 %Test.by providing Stratify as yes so that same proportions in the train and test data will be maintained. Random state provides you the freedom to work on different system while using the same sample data from the population. The Random state value chosen as 10

CART Model

```
dt_model = DecisionTreeClassifier(criterion = 'gini',random_state=1)
```

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

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=1, splitter='best')
```

```
y_predict = dt_model.predict(X_test)
```

We Choose Decision tree classifier as our first model with criterion as ‘Gini’. CART uses Gini Impurity as the criterion to choose the root node and for further node splits. So now we have fitted and predicted our base model, after building the base model we need to perform Hyper parameter tuning is done for getting better accuracy and we use Gridsearchcv for performing hyprerparameter tuning.

Hyper parameter Tuning – CART

From the hyper parameter tuning we got below best parameters

```
grid_search.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 6,
 'max_features': 4,
 'min_samples_leaf': 8,
 'min_samples_split': 7}
```

By substituting the best parameters, again we predict the test data, then we compare check the accuracy of the base model and tuned model.

Random Forest Model

```
from sklearn.ensemble import RandomForestClassifier
rfcl = RandomForestClassifier(n_estimators = 501)
rfcl = rfcl.fit(X_train,y_train)
```

We Choose Random Forest classifier as our second model which is a Bagging (Bootstrap) technique in ensemble method with n_estimators(number of decision trees) as 501.

Random forest also uses Gini Impurity as the criterion for internal decision tree algorithm to choose the root node and for further node splits. So now we have fitted and predicted our base model, after building the base model we need to perform Hyper parameter tuning is done for getting better accuracy and we use Gridsearchcv for performing hyprerparameter tuning.

Hyper parameter Tuning- Random Forest

```
grid_search2.best_params_
```

```
{'max_depth': 10,
 'max_features': 4,
 'min_samples_leaf': 50,
 'min_samples_split': 150,
 'n_estimators': 501}
```

```
rfcl_model = RandomForestClassifier(max_depth= 10,min_samples_leaf=50,min_samples_split=150,max_features=4,n_estimators=501)
rfcl_model.fit(X_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=10, max_features=4, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=50, min_samples_split=150,
                       min_weight_fraction_leaf=0.0, n_estimators=501,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
ytrain_predict2 = rfcl_model.predict(X_train)
ytest_predict2 = rfcl_model.predict(X_test)
```

By substituting the best parameters, again we predict the test data, then we compare check the accuracy of the base model and tuned model.

ANN Model

Scaling of data is a critical step in ANN models, because as ANN works on the distance based algorithm technique scaling is always advantageous as the model understands the pattern very well.

```
sc = StandardScaler()
X_train_scaled = sc.fit_transform(X_train)
X_test_scaled = sc.transform(X_test)
```

```
clf = MLPClassifier(hidden_layer_sizes=10, max_iter=5000,
                    solver='sgd', verbose=True,  random_state=21,tol=0.01)
```

After scaling the train and test data we Choose Artificial neural network classifier as our third model which is a neural network technique which works on the weights estimations. Generally Ann is also known as Feed forward neural network, so now we fitted our base model

ANN calculates the log loss after completing total EPOCH,s and uses optimization technique like Adam, Stochastic gradient,etc to reduce the cost function. Hyper parameter tuning is done for getting better accuracy and we use Gridsearchcv for performing hyprerparameter tuning.

Hyper parameter Tuning – ANN Model

```
grid_search3.best_params_
```

```
{'activation': 'logistic',
 'hidden_layer_sizes': 7,
 'max_iter': 200,
 'solver': 'adam',
 'tol': 0.0001}
```

```
clf_model = MLPClassifier(activation= 'logistic',hidden_layer_sizes=7,max_iter=200,solver= 'adam',tol= 0.0001)
```

```
ytrain_predict3 = clf_model.predict(X_train_scaled)
ytest_predict3 = clf_model.predict(X_test_scaled)
```

By substituting the best parameters, again we predict the test data, then we compare check the accuracy of the base model and tuned model.

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

a. CART MODEL (Train and Test)
b. RNN Model (Train and Test)
c. ANN Model (Train and Test)

Performance metrics: used for calculating the model performance in classification problems are

Accuracy Score- Companies use machine learning models to make practical business decisions.

Confusion Matrix - Confusion matrices make it easy to tell how accurate a model's outcomes are.

ROC (**Receiver Operating Characteristic Curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

AUC score (Area under the curve): The AUC value lies between 0.5 to 1 where 0.5 denotes a bad classifier and 1 denotes an excellent classifier.

RECALL = TP/TP+FN

PRECISION = TP/TP+FP

F1 Score = 2(SENSITIVITY * PRECISION)/(SENSITIVITY + PRECISION).

.

CART Model: (Train and Test)

```
## ACCURACY SCORE
print(reg_dt_model.score(X_train,y_train))
print(reg_dt_model.score(X_test,y_test))
```

```
0.7933333333333333
0.7522222222222222
```

```
##Confusion Matrix Decision Tree
print(metrics.confusion_matrix(y_test, ytest_predict))
print(metrics.confusion_matrix(y_train, ytrain_predict))
```

```
[[551  72]
 [151 126]]
[[1338  115]
 [ 319  328]]
```

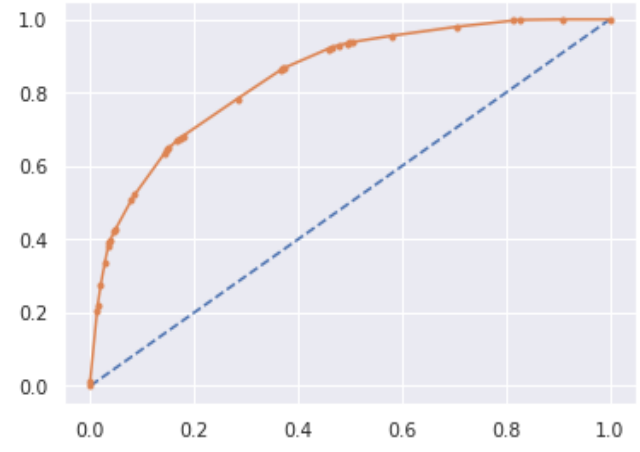
```
## Classifcation report train
print(metrics.classification_report(y_train, ytrain_predict))
```

	precision	recall	f1-score	support
0	0.81	0.92	0.86	1453
1	0.74	0.51	0.60	647
accuracy			0.79	2100
macro avg	0.77	0.71	0.73	2100
weighted avg	0.79	0.79	0.78	2100

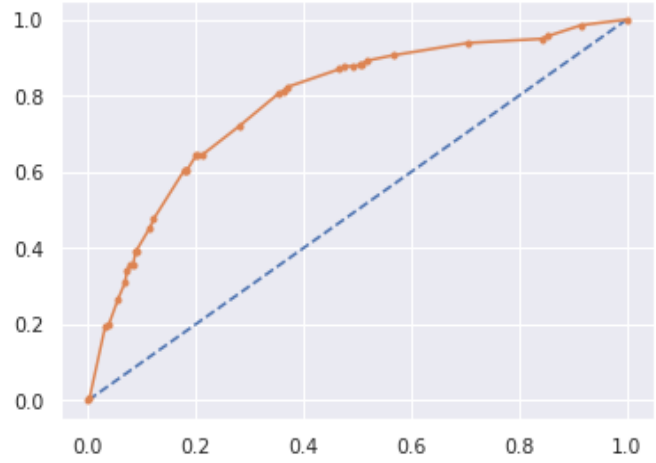
```
##Classification report test
print(metrics.classification_report(y_test, ytest_predict))
```

	precision	recall	f1-score	support
0	0.78	0.88	0.83	623
1	0.64	0.45	0.53	277
accuracy			0.75	900
macro avg	0.71	0.67	0.68	900
weighted avg	0.74	0.75	0.74	900

AUC and ROC Train
AUC: 0.844



AUC and ROC Test
AUC: 0.783



Inference:

In the initial CART model the Accuracy scores was very high indicating it is an over fitted model. This called for additional pruning parameters; the above classification report has been generated after using best hyper parameters provided by the Grid search CV function which provides better results.

RNN Model: (Train and Test)

```
## Accuracy score train and test)
print(rfcl_model.score(X_test,y_test))
print(rfcl_model.score(X_train,y_train))
```

```
0.7633333333333333
0.7876190476190477
```

```
## Confusion Matrix
print(metrics.confusion_matrix(y_test, ytest_predict2))
print(metrics.confusion_matrix(y_train, ytrain_predict2))
```

```
[[548  75]
 [138 139]]
[[1327  126]
 [ 320  327]]
```

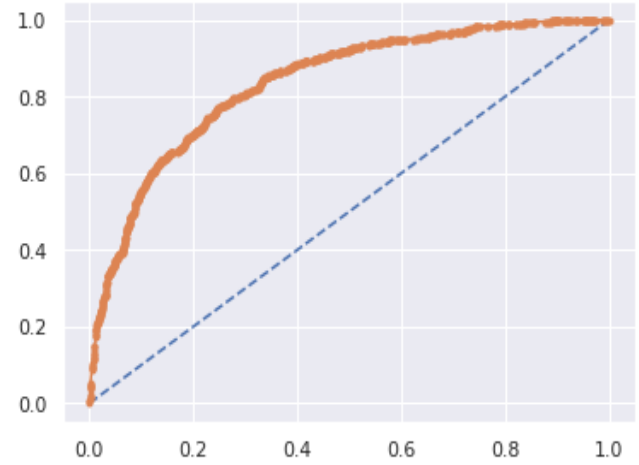
```
## Classificationn report train
print(metrics.classification_report(y_train, ytrain_predict2))
```

	precision	recall	f1-score	support
0	0.81	0.91	0.86	1453
1	0.72	0.51	0.59	647
accuracy			0.79	2100
macro avg	0.76	0.71	0.73	2100
weighted avg	0.78	0.79	0.78	2100

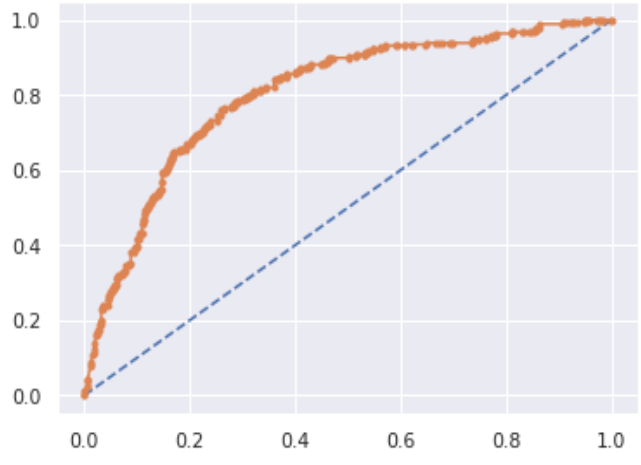
```
## Classification report test
print(metrics.classification_report(y_test, ytest_predict2))
```

	precision	recall	f1-score	support
0	0.80	0.88	0.84	623
1	0.65	0.50	0.57	277
accuracy			0.76	900
macro avg	0.72	0.69	0.70	900
weighted avg	0.75	0.76	0.75	900

AUC and ROC Train
AUC: 0.837



AUC and ROC Test
AUC: 0.804



Inference:

Based on the classification report, the performance of dependent variable (Claimed – Yes) model in train and test data is well within the threshold of 10%..

ANN Model: (Train and Test)

```
##Accuracy score
print(clf_model.score(X_test_scaled,y_test))
print(clf_model.score(X_train_scaled,y_train))
```

```
0.6622222222222223
0.6971428571428572
```

```
## Confusion Matrix
print(metrics.confusion_matrix(y_test, ytest_predict3))
print(metrics.confusion_matrix(y_train, ytrain_predict3))
```

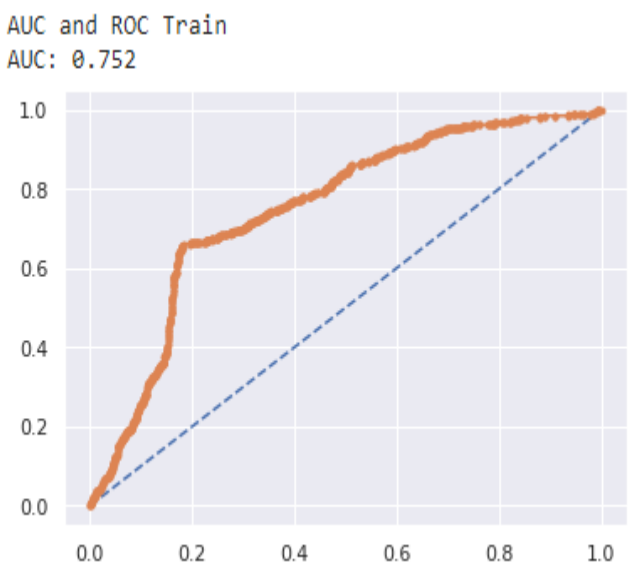
```
[[405 218]
 [ 86 191]]
[[1008  445]
 [ 191  456]]
```

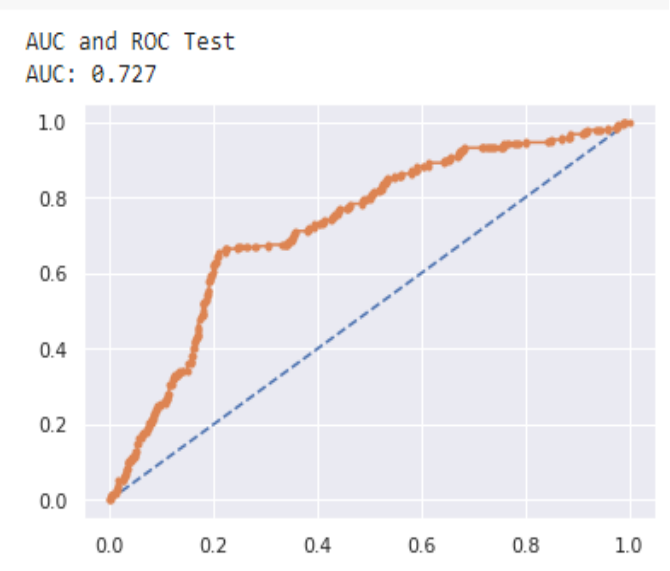
```
## Classification report train
print(metrics.classification_report(y_train, ytrain_predict3))
```

	precision	recall	f1-score	support
0	0.84	0.69	0.76	1453
1	0.51	0.70	0.59	647
accuracy			0.70	2100
macro avg	0.67	0.70	0.67	2100
weighted avg	0.74	0.70	0.71	2100

```
## Classification report  test
print(metrics.classification_report(y_test, ytest_predict3))
```

	precision	recall	f1-score	support
0	0.82	0.65	0.73	623
1	0.47	0.69	0.56	277
accuracy			0.66	900
macro avg	0.65	0.67	0.64	900
weighted avg	0.71	0.66	0.67	900





Inference:

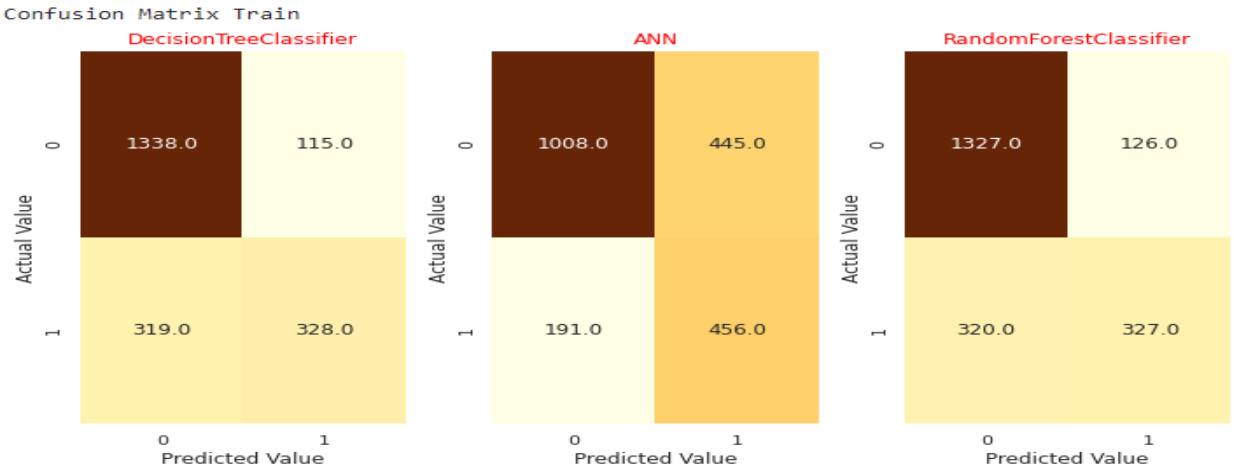
Based on the classification report, the performance of dependent variable (Claimed – Yes) model in train and test data is well within the threshold of 10%.

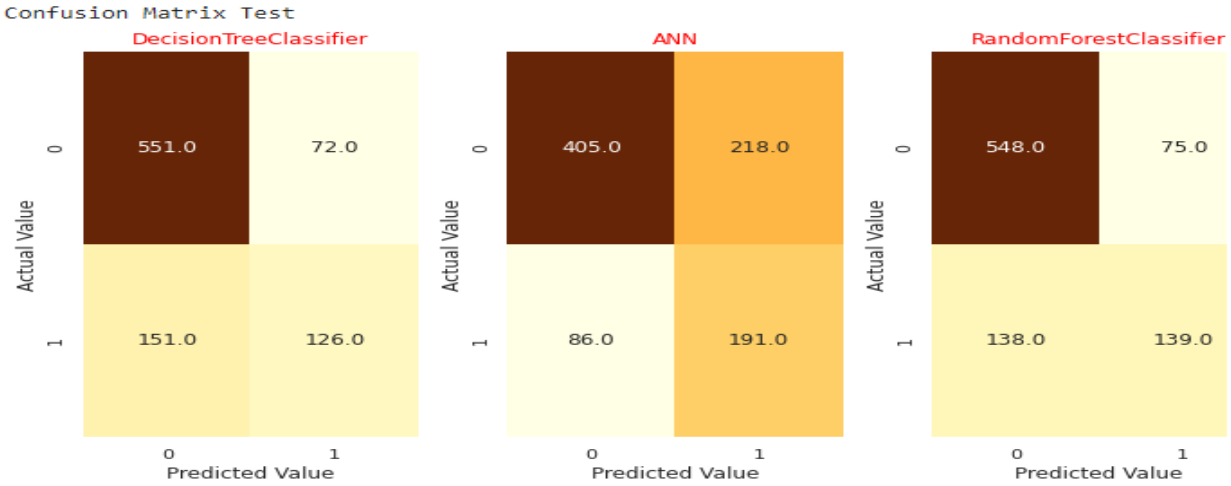
Final Model: Compare all the model and write an inference which model is best/optimized.

Accuracy score: (Train and Test)

```
Accuracy Score for Train set for DecisionTreeClassifier is 0.79
Accuracy Score for Test set for DecisionTreeClassifier is 0.76
Accuracy Score for Train set for ANN is 0.8
Accuracy Score for Test set for ANN is 0.7
Accuracy Score for Train set for RandomForestClassifier is 0.79
Accuracy Score for Test set for RandomForestClassifier is 0.76
```

Confusion Matrix: (Train and Test)



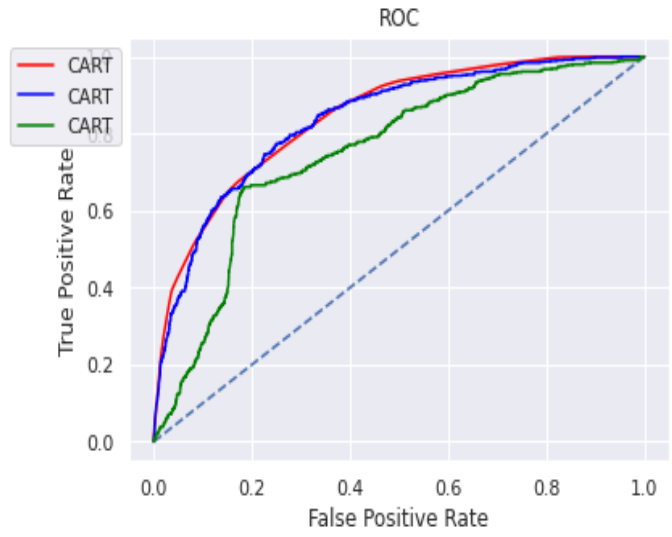


Classification Report: (Train and Test)

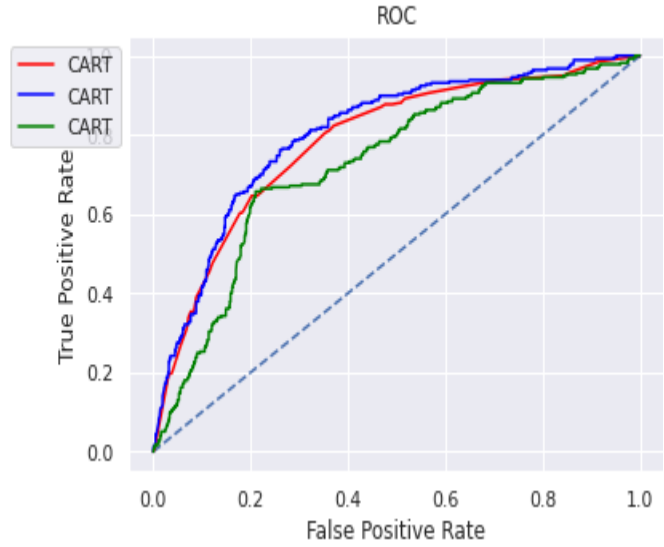
	ANN Test	ANN Train	RNN Test	RNN Train	CART Test	CART Train
Accuracy	0.66	0.70	0.76	0.79	0.75	0.79
AUC	0.73	0.75	0.80	0.84	0.78	0.84
Precision	0.65	0.67	0.65	0.67	0.71	0.77
Recall	0.67	0.70	0.69	0.71	0.67	0.71
F1 Score	0.64	0.67	0.70	0.73	0.68	0.73

ROC Curve: (Train and Test)

AUC and ROC Train
<matplotlib.legend.Legend at 0x7fd865124710>



AUC and ROC Test
<matplotlib.legend.Legend at 0x7fd86510e250>



Inference on the model performance metrics:

a. Random Forest model has a difference of only 1% to 3% performance score in all types of performance metrics. So the model is performing good with both training and test data
b. Also if you see in the confusion matrix Random Forest model predicts only less amount of false positive with test data, so this indicates that the claim frequency deviation will not be so high from the predicted model.
c. So from the above inferences we came to a conclusion for the best model as Random Forest Classifier model

Inference: Basis on these predictions, what are the business insights and recommendations

a. The below features or information's are important to be considered for processing the claim request
    1. Age
    2. Duration
    3. Product name
    4. Sales
b. The customers having age between 34 to 36, with 8 hours of travel duration, having 20 dollars of sales value and the customization plan has a higher claim frequency, so our recommendation is company can expect more claim request frequency from this bucket range
c. The highest claim frequency is through Airline type in the destination Asia
d. Though the 'not claimed' ratio is high, the commission provided for Bronze plan in Asia is remarkably high compared to other ' Not Claimed' Category.
e. The Majority of these claims are processed through the C2B agency.
f. 90% of insurance is done by an online channel .
g. Higher the commission the duration for processing also increases.
h. The currency for the commission is not mentioned, assuming that the commission was provided in dollars. Majority of commission above $100 has been processed by the travel agency.
i. All the offline insurance is through agency only.
j. From the built prediction model (Random Forest) using the historical data provided by the company we can able to predict around 75% of the future claim frequency of the customers. So the company can able to keep the funds prepared accordingly and avoid last minute fund mismanagement due to sudden increase in claim frequency,
k. Also, if we can able to discuss more with concern domain person of the company we can able to collect more insights on the same problem statement and build a better accurate model for predicting the future claim frequency.