Business Report
on a case study of

Insurance Firm using the following Models:

Classification and Regression Trees

+

Random Forest

+

Artificial Neural Network

Submitted by: Group 5

Table of Contents

Topic **Page no.**

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

1.1. Understanding the case study:

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

Following columns are given in the data set and the meaning of each column is interpreted as follows:

- **Claimed:** This column is the target variable representing weather the passenger has claimed the insurance amount. It is a Boolean variable with values as 'Yes' and 'No'.
- **Agency_Code:** This is the code of the tour firm representing the agency with 3 letters. Here we have 4 agencies with codes such as 'JZI', 'CWT', 'C2B' and 'EPX'.
- **Type:** This variable refers to the type of tour insurance firms being 'Airlines' or 'Travel Agency'.
- **Channel:** This represents the distribution channel of tour insurance agencies as 'Online' and 'Offline'.
- **Product Name:** This column is the representation for the name of the tour insurance products. The 5 types of products used by Passengers are: 'Gold Plan', 'Silver Plan', 'Bronze Plan', 'Customized Plan' and 'Cancellation Plan'.
- **Duration:** This is the duration of the tour for each passenger.
- **Destination:** This variable determines the destination of the tour. This dataset has 3 destinations value as 'EUROPE', 'Americas' and 'ASIA'.
- **Sales:** This is an important variable signifying the amount of sales made by the tour insurance policies.
- **Commission:** This is the commission received for tour insurance firm on the amount of 'Sales'.
- **Age:** This is the age in years of the passenger insured.

1.2. Exploratory Data Analysis:

1.2.1. Checking the data type for each column in the data.

```
#Checking the data types of each variables
df.dtypes
```

```
Age               int64
Agency_Code      object
Type             object
Claimed          object
Commision       float64
Channel          object
Duration          int64
Sales           float64
Product Name     object
Destination      object
```

We have checked that there are **10 variables** with different data types like integer, float, and object.

1.2.2. Checking the null values in the data

```
#Checking the null value in each columns
df.isnull().sum()
```

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

It is checked that there are **no null values** present in any of the columns. Hence no null value treatment is needed over this dataset.

1.2.3. Checking the duplicated values in the data

```
#Checking for the duplicate records in the data
df.duplicated().sum()
```

```
139
```

There are 139 duplicate rows in the data.

```
#Dropping the duplicate records from the data
df.drop_duplicates(keep='first', inplace=True)
```

```
df.shape
```

```
(2861, 10)
```

Here, we observe that there are**139 duplicated rows** in the data set. Duplicates are dropped with the help of drop_duplicates(). Finally we get 2861 unique records.

1.2.4. Checking the descriptive statistics of the data.

```
# checking dataset decription for all numerical variables
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	2861.0	38.204124	10.678106	8.0	31.0	36.00	43.00	84.00
Commision	2861.0	15.080996	25.826834	0.0	0.0	5.63	17.82	210.21
Duration	2861.0	72.120238	135.977200	-1.0	12.0	28.00	66.00	4580.00
Sales	2861.0	61.757878	71.399740	0.0	20.0	33.50	69.30	539.00

After removing missing values, final dataset has 2816 records. 'Duration' has the largest standard deviation of 135.977 in comparison to others. 'Sales' is on 2nd number for standard deviation, but it is 71.399. Data points of 'Duration' are scattered far from the mean in comparison to others.

'Duration' has mean of 72.12 followed by 'Sales' which has mean of 61.75 and 'Commission' has mean of 15.08 which is lowest amongst all.

'Age' has mean of 38.20 and minimum age is 8 and max of 84.'Age' range is large but still its median is 36 although max is 84. Standard deviation of 'Age' is 10.678, it is lowest among other variables which means the values are clustered closely.

```
# Check description for object columns
df[cat].describe().T
```

	count	unique	top	freq
Agency_Code	2861	4	EPX	1238
Type	2861	2	Travel Agency	1709
Channel	2861	2	Online	2815
Product Name	2861	5	Customised Plan	1071
Destination	2861	3	ASIA	2327

Now, we have checked the descriptive stat for categorical value. 'Product Name' has the largest number of unique items that is 5 followed by 'Agency_Code' which has 4 and 'Destination' has 3 unique items, 'Type' and 'Channel' has 2 unique items in each.

In ‘Channel’ variable, online item is in majority with 2815 frequency. In ‘Destination’, Asia is in majority with 2327 frequency and other categories totals to only 534.

In ‘Type’, Agency’ and ‘Product Name’, Travel Agency top with the frequency of 1709, EPX is in majority with 1238 frequency and Customised Plan has frequency of 1071, respectively.

1.2.5. Checking the target column for the biasness.

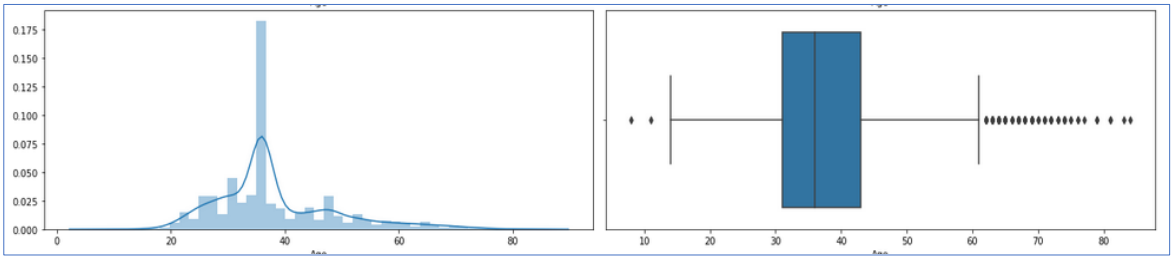
```
Cases in no:
 No     1947
Yes     914
Name: Claimed, dtype: int64

Cases in %:
 No     68.053128
Yes    31.946872
Name: Claimed, dtype: float64
```

It is checked that the data is moderately balanced with approximately 30-60 proportion of Claimed vs Non claimed records.

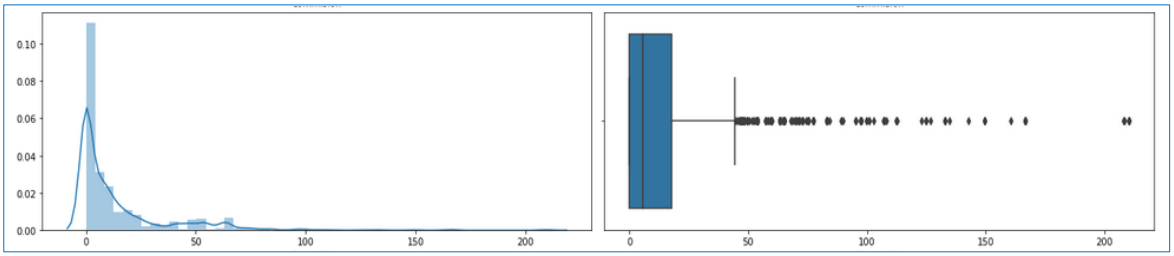
1.2.6. Checking the outliers and overall distribution of numerical columns.

Age (fig1):



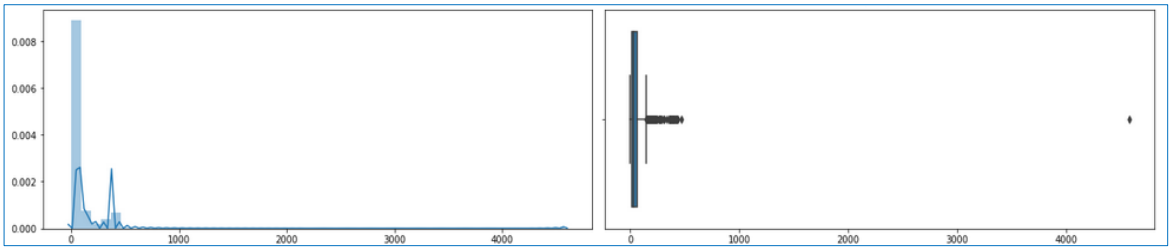
As per distplot, we can conclude that data is skewed a little towards right side. Age also has outliers and that too on both bounds.

Commission(fig2):



Commission is highly skewed towards right and has large number of outliers on upper bound side.

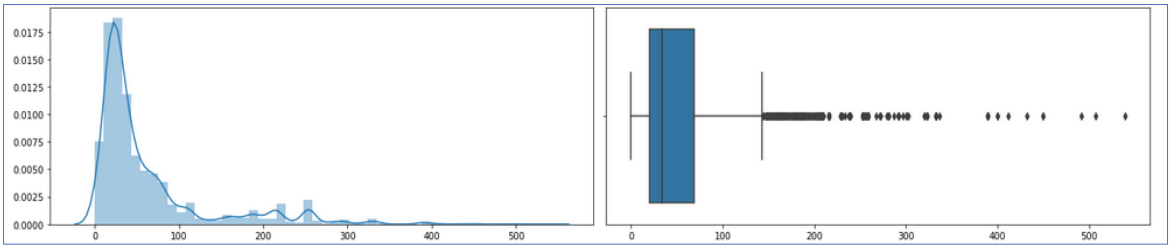
Duration(fig3):



It is showing two spikes in the distplot at very early stage of graph and can also be seen that it is highly skewed, but the range is also big. This defines the standard deviation calculated above which states that data is scattered far away from mean.

Duration also has outlier but one of data point is very far from the outer bound.

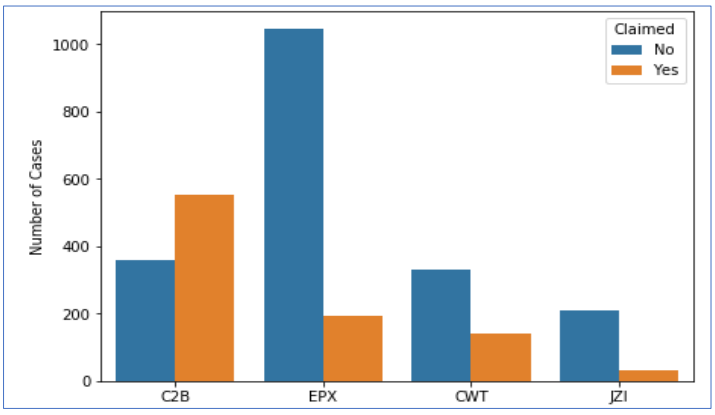
Sales(fig4):



Data points of Sales is also skewed towards right. It also has outliers which are spread far from Upper bound limit.

*1.2.7. Checking the distribution of **categorical columns** w.r.t. **target column**.*

Agency_Code(fig5):

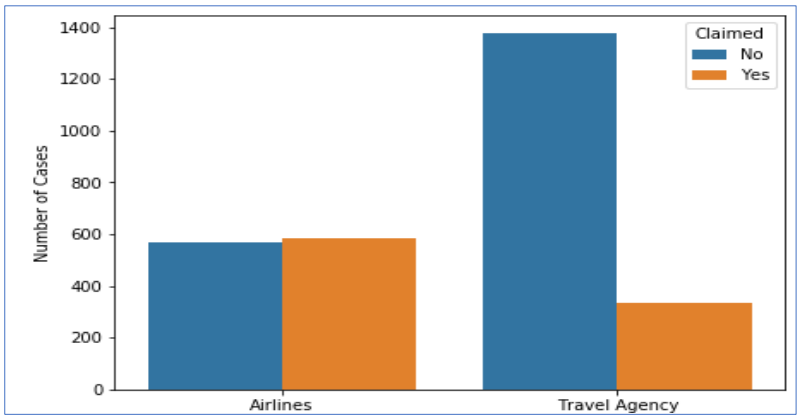


Here, we may observe that C2B agency not only has the highest claiming records, but its ratio of claimed to non-claimed is higher than 1 i.e., its claiming customers are significantly higher than non-claiming customers.

Rest all three have claimed to not-claimed ratio less than 1 which is expected in the insurance business.

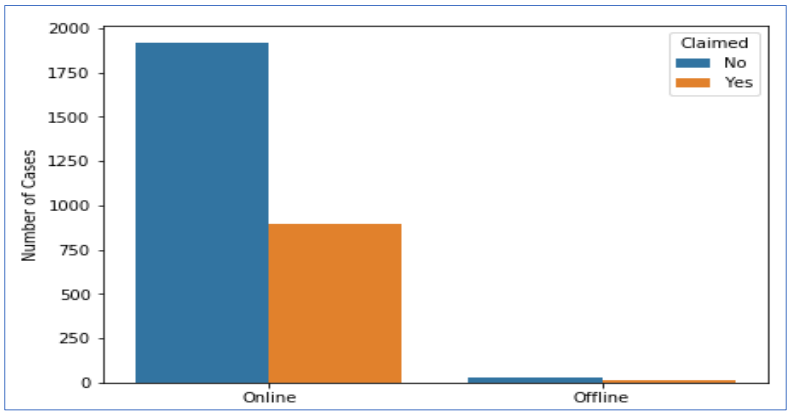
Interestingly, EPX is selling the more than double the number of policies than C2B, but the number of claimed records of EPX are less than half the number of claimed records of C2B. Hence, this feature may have important role in prediction.

Type(fig6):



Total number of policies sold by Airlines is significantly lower than the total number of policies sold by Travel Agency, and still, the number of total claims is approximately double in Airlines. Hence, this feature may have important role in prediction.

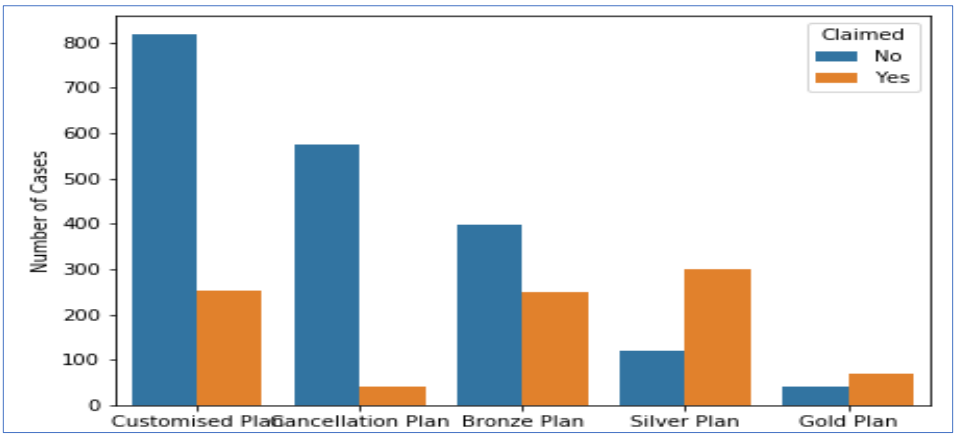
Channel(fig6):



Offline has very low contribution towards overall number of records.

However, for both online and offline, number of claimed records are lower than not claimed. Hence, this feature seems not to important for prediction.

Product Name(fig7):



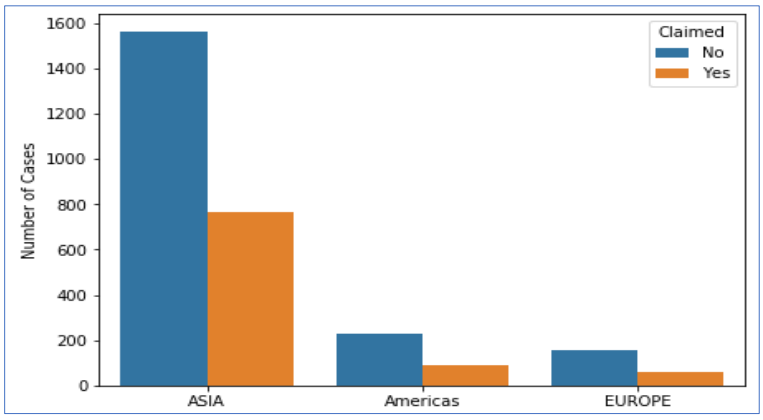
Here, Customized, Cancellation and Bronze plan have claimed to not-claimed ratio less than 1, but for Silver and Gold plan this ratio is more than 1.

We may also observe that Silver has the highest claimed to not-claimed ratio of number of records.

Another observation is for Bronze plan too, this ratio is more than 0.5 which means more than 50% customers have claimed

Hence, Product Name may also be a useful variable for the model.

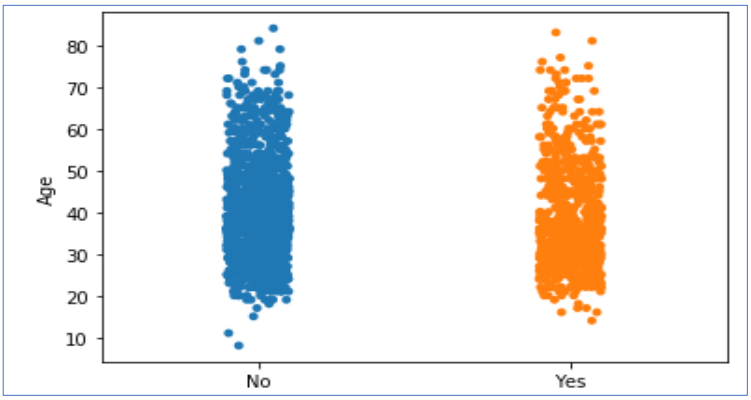
Destination(fig8):



Destination, may show the most number of customers travelled but it does not give any clear picture to claim status as for all destinations claimed records are reasonable less than not-claimed records.

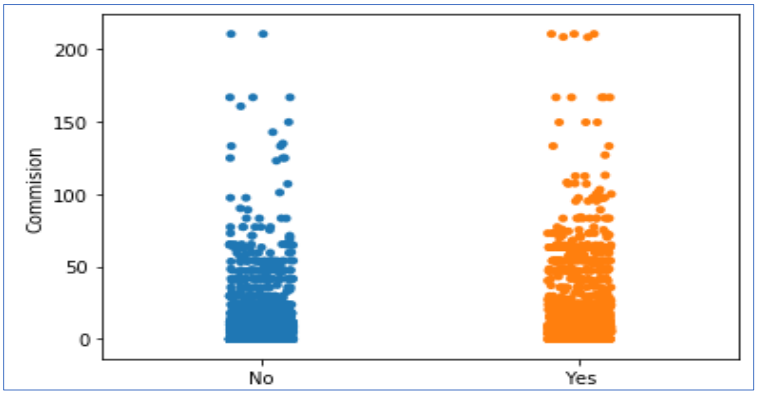
1.2.8. Checking the relation of numerical columns w.r.t. target column.

Age(fig9):



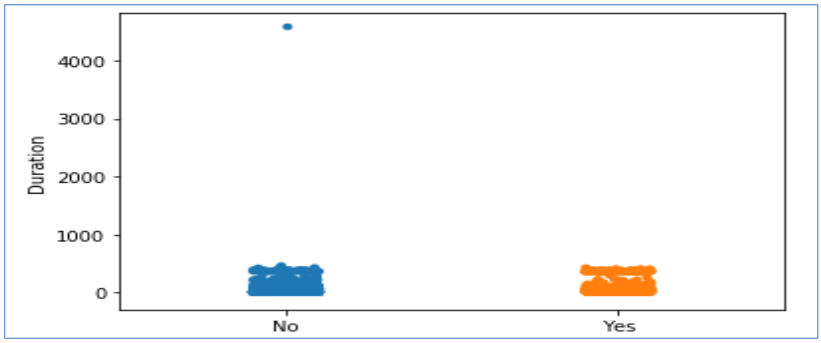
The distribution of both claimed and not-claimed is almost same across age of customers.

Commission(fig10):



Here, for claimed records the distribution is slightly denser on higher commission range specially above 50 bar.

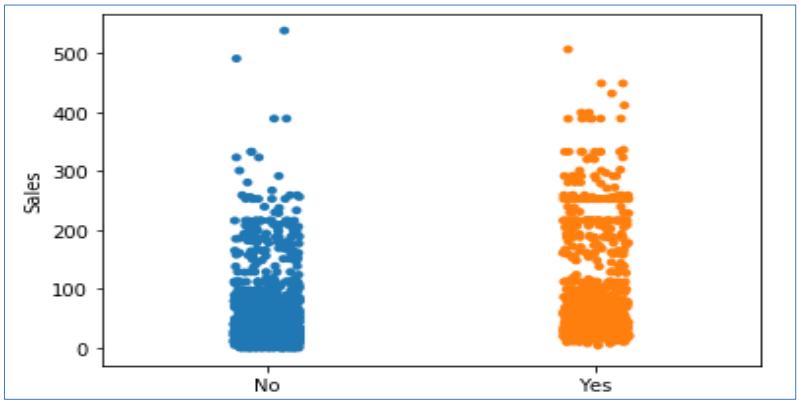
Duration(fig11):



In Duration, the distribution is more or less same for both claimed and not claimed with slightly extra density for not-claimes

Also, Not claimed has one data point which is above 4000 which is not realistic as tour duration of above 4000 does not make sense and may be checked from the data source.

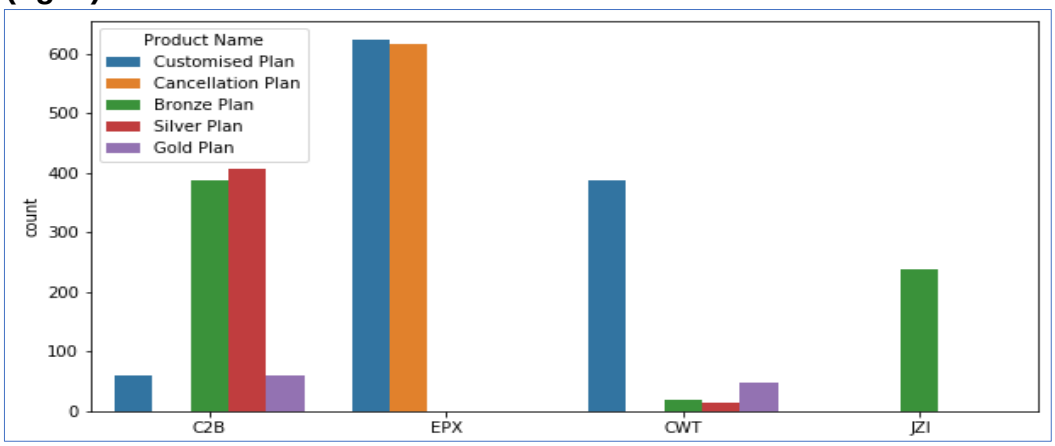
Sales(fig12)



Policies with lower sales value have similar distribution for both claimed and not-claimed distribution, but insurance policies with higher sales value (specially above 200 bar) show distinctly more records for claimed side. Hence, this feature may also play important role in the prediction.

*1.2.9. Checking **multivariate analysis** to dig deep on our earlier findings*

(fig13)



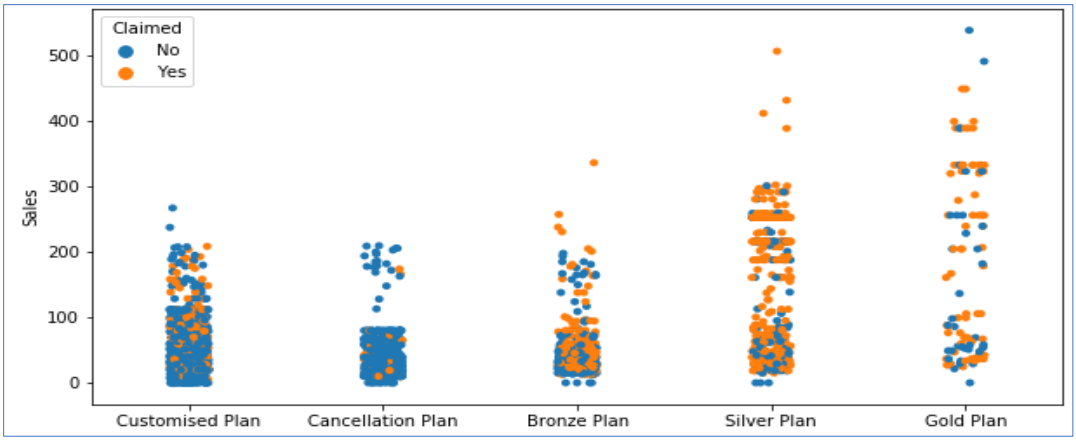
As we noticed in our EDA before that C2B Agency type and Silver Plan stood out highest in terms of claimed case, we may notice a connection between the two features in this graph.

Silver Plan has been sold by only two agencies C2B and CWT, where CWT has a very negligible share of silver customer.

Also, second highest claimed plan type is Gold which is again being sold by C2B and CWT, but from previous graph (fig5) we have already seen that CWT has a lower claimed ratio, so our red flag may be C2B.

Here, we may also notice that C2B is selling four out of five plans of which Bronze and Silver are significantly higher in number than others. As we saw before(fig7) that claimed to not-claimed ratio of Silver and Gold was higher than 1 and for Bronze it was approximately 0.5 (i.e., 50% claim ratio), C2B is selling all plans with higher claim ratio.

(fig14)



Another feature we found from fig12 as important was sales on higher value had more claims. So now we may observe that sales values of Silver, Bronze and Gold is on higher side as compared to Customized and Cancellation plans and these are the same plans with high claimed ratio too.

Also, for Silver plan, the claimed records are most dense of all specially at 200 bar above of sales value which is in line with our observation from previous analysis (fig12).

Observation:

Overall, we may have some important features which raise red flag : Product name:Silver Plan, Agency type: C2B, Sales: with high value records

2. 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

2.1. Split Data:

In order to build the models, we need to divide the data into train data set and test data set, so that we can measure the performance metrics of both the data sets and avoid problems like overfitting, etc.

Before splitting the data into train and test we need to perform couple of data pre-processing steps:

1. Separating out the target and independent variables into X and y.

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

y = data.pop("Claimed")
```

2. Converting all the columns with object data types to numeric ones because these models cannot process string values in python.

```
# Decision tree in Python can take only numerical / categorical colums. It cannot take string / object types.
X = pd.get_dummies(X)
X.head()
```

Now, the data set is ready to be split into train and test data sets. We have followed the splitting criteria of 70% data in train and 30% data in test with the random_state set to 1 to get same output on each run.

```
# splitting data into training and test set for independent attributes
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)
```

The next step is to build the models with selection of hyperparameters to get the most optimum model. A hyperparameter is a parameter whose value is used to control the learning process, few of which we have used in the model building are studies as follows with the associated models.

2.2. Build Models - Decision Tree Or CART

The decision tree method is a powerful and popular predictive machine learning technique that is used for both classification and regression. So, it is also known as Classification and Regression Trees (CART).

A *Classification And Regression Tree* (CART), is a predictive model, which explains how an outcome variable's values can be predicted based on other values. A CART output is a decision tree where each fork is a split in a predictor variable and each end node contains a prediction for the outcome variable.

Parameters:

criterion{“mse”, “friedman_mse”, “mae”, “poisson”}, default=”mse”

The function to measure the quality of a split. Supported criteria are “mse” for the mean squared error, which is equal to variance reduction as feature selection criterion and minimizes the L2 loss using the mean of each terminal node, “friedman_mse”, which uses mean squared error with Friedman’s improvement score for potential splits, “mae” for the mean absolute error, which minimizes the L1 loss using the median of each terminal node, and “poisson” which uses reduction in Poisson deviance to find splits.

splitter{“best”, “random”}, default=”best”

The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

max_depth : int, default=None

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : int or float, default=2

The minimum number of samples required to split an internal node:

• If int, then consider min_samples_split as the minimum number.

• If float, then min_samples_split is a fraction and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.

min_samples_leaf : int or float, default=1

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider min_samples_leaf as the minimum number.
- If float, then min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each node.
- Feature Importance: In decision tree, we can know the order of feature importance for building the tree from root to the leaf with help of feature_importances_. It provides the best features in data to help determining the tree split. Here, we may observe that the three out of first four features stood out as red flags in the EDA.

```
# importance of features in the tree building ( The importance of a feature is computed as the
#(normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance )

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

	imp
Sales	0.254349
Duration	0.251523
Age	0.172918
Agency_Code_C2B	0.164117
Commision	0.071581
Destination_EUROPE	0.013830
Product Name_Cancellation Plan	0.013030
Product Name_Silver Plan	0.010117
Product Name_Bronze Plan	0.008055
Destination_ASIA	0.007807
Product Name_Customised Plan	0.007314
Product Name_Gold Plan	0.007038
Destination_Americas	0.003535
Channel_Offline	0.003368
Agency_Code_CWT	0.003087
Agency_Code_EPX	0.002734
Type_Airlines	0.002011
Channel_Online	0.001852
Agency_Code_JZI	0.001736
Type_Travel Agency	0.000000

- Grid Search and Best Grid parameters for model training: This is how we have built the model for CART in python.

```
# fourth parameters

params = {'max_depth':[2,3,4,5,6,7],
          'min_samples_leaf':[40,50,70,90,110],
          'min_samples_split':[120,150,210,270,330]}

# trial with first set of params
dt_model = DecisionTreeClassifier()
grid_search = GridSearchCV(estimator = dt_model,
                           param_grid = params,
                           cv = 3)
print(grid_search.fit(X_train, train_labels))

print('\nBest params:' ,grid_search.best_params_)
```

2.3. Random forest classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

Parameters:

n_estimators : int, default=100

The number of trees in the forest.

criterion{“gini”, “entropy”}, default=”gini”

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain. Note: this parameter is tree-specific.

max_depth :int, default=None

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split :int or float, default=2

The minimum number of samples required to split an internal node:

- If int, then consider min_samples_split as the minimum number
- If float, then min_samples_split is a fraction and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.

min_samples_leaf: int or float, default=1

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider min_samples_leaf as the minimum number.
- If float, then min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each node.

```
# second set of params

params = {
    'max_depth':[8,10],
    'max_features':[5,7],
    'min_samples_leaf':[6,12],
    'min_samples_split':[18,36],
    'n_estimators':[75,150],
}

rfcl = RandomForestClassifier()
grid_search = GridSearchCV(estimator=rfcl, param_grid=params, cv=3)

grid_search.fit(X_train, train_labels)

print('\nBest Params:',grid_search.best_params_)
```

This is how we have built the model for Random Forest in python. The optimum model selection will be done in later sections.

2.4. Artificial Neural Network or Multi-layer Perceptron classifier (ANN)

This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

Parameters:

hidden_layer_sizes : tuple, length = n_layers - 2, default=(100,)

The ith element represents the number of neurons in the ith hidden layer.

Solver {'lbfgs', 'sgd', 'adam'}, default='adam'

The solver for weight optimization.

- 'lbfgs' is an optimizer in the family of quasi-Newton methods.
- 'sgd' refers to stochastic gradient descent.
- adam' refers to a stochastic gradient-based optimizer

random_state : int, RandomState instance, default=None

Determines random number generation for weights and bias initialization, train-test split if early stopping is used, and batch sampling when solver='sgd' or 'adam'. Pass an int for reproducible results across multiple function calls.

max_iter : int, default=200

Maximum number of iterations. The solver iterates until convergence (determined by 'tol') or this number of iterations. For stochastic solvers ('sgd', 'adam'), note that this determines the number of epochs (how many times each data point will be used), not the number of gradient steps.

verbosebool, default=False

Whether to print progress messages to stdout.

For this model, we need to scale the data before building the model because in input layer the multiplied value of weight and input variable will be used. Hence it is important to bring all the inputs to the same scale in order to avoid the biasness in the variables.

```
# Sixth run

params = {'hidden_layer_sizes':[(500,500,500)],
          'max_iter':[5000],
          'solver':['sgd','adam'],
          'activation':['relu'],
          'tol':[0.01]}

mlp = MLPClassifier()

grid_search = GridSearchCV(estimator=mlp, param_grid=params, cv=3)
grid_search.fit(X_train, train_labels)

print('\nBest params:',grid_search.best_params_)

grid1 = grid_search.best_estimator_

ytrain_predict = grid1.predict(X_train)
ytest_predict = grid1.predict(X_test)
```

This is how we have built the model for Multi-layer perceptron classifier (ANN) in python. The optimum model selection will be done in later sections.

3. 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

3.1. Performance Metrics – Business Perspective

To compare our model, we need to determine which parameters are important from our business problem:

As we know that we need to build a model which may address the issues of high claim frequency, our focus of prediction is to have correct numbers of claimed status.

3.1.1. Hence, the confusion metrics for the model can be as below:

Claimed Status	Predicted Yes	Predicted No
Actual Yes	TP	FN
Actual No	FP	TN

Here, the brief definition of Confusion metrics is as below

- ✓ True Positives (TP): True positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1(True)
- ✓ True Negatives (TN): True negatives are the cases when the actual class of the data point was 0(False) and the predicted is also 0(False)
- ✓ False Positives (FP): False positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one. (1)
- ✓ False Negatives (FN): False negatives are the cases when the actual class of the data point was 1(True) and the predicted is 0(False). False is because the model has predicted incorrectly and negative because the class predicted was a negative one. (0)

*3.1.2. From the above, we know that our interest is in knowing **correct number of Claimed Yes**, which means TP is more important for us.*

*Now every wrong prediction of 'Claimed No' will mean another policy sold in loss which defeats the purpose of our model. Hence, **minimizing False Negatives** is our focus for the model.*

3.1.3. Before analyzing our model, we need to check the list of metrics that are important for our model and can be derived from Confusion metrics. Based on TP, TN, FP and FN, we may further find model evaluation metrics as follows:

- ✓ **Accuracy**: Accuracy is the quintessential classification metric. It is easy to understand. And easily suited for binary as well as a multiclass classification problem. Accuracy is the proportion of true results among the total number of cases examined.
  Accuracy = (TP+TN)/(TP+FP+FN+TN)
  When to use?
  Accuracy is a valid choice of evaluation for classification problems which are well balanced and not skewed or No class imbalance.

- ✓ **Precision**: Precision is a measure that tells us what proportion of predicted Positives is truly Positive.
  Precision = (TP)/(TP+FP)
  When to use?
  Precision is a valid choice of evaluation metric when we want to be very sure of our prediction. For example: If we are building a system to predict if we should decrease the credit limit on a particular account, we want to be very sure about our prediction or it may result in customer dissatisfaction.
- ✓ **Recall or Sensitivity:** Another very useful measure is recall, which answers a different question: what proportion of actual Positives is correctly classified?

Recall = (TP)/(TP+FN)

When to use?

Recall is a valid choice of evaluation metric when we want to capture as many positives as possible. For example: If we are building a system to predict if a person has cancer or not, we want to capture the disease even if we are not very sure.

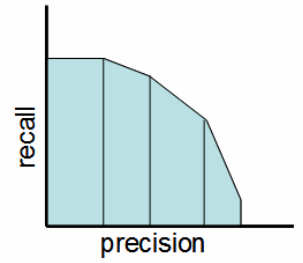
- ✓ **Specificity:** Specificity is the exact opposite of Recall. It answers a different question: what proportion of actual Negatives is correctly classified??

  Specificity = (TN)/(TN+FP)

  When to use?

  Specificity is a valid choice of evaluation metric when we want to capture as many negatives as possible.
- ✓ **F1 Score**: This is an important evaluation metric and it tends to be used a lot in classification projects. The F1 score is a number between 0 and 1 and is the harmonic mean of precision and recall.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$



When to use?

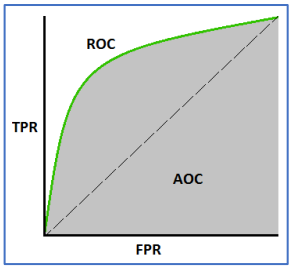
When we want to have a model with both good precision and recall. Simply stated the F1 score sort of maintains a balance between the precision and recall for your classifier. If your precision is low, the F1 is low and if the recall is low again your F1 score is low.

- ✓ **AUC-ROC:** AUC ROC basically indicates how well the probabilities from the positive classes are separated from the negative classes. AUC is the area under the ROC curve.

  What is the ROC curve?

  We have got the probabilities from our classifier. We can use various threshold values to plot our sensitivity (TPR) and (1-specificity)(FPR) on the cure and we will have a ROC curve.

  Where True positive rate or TPR is just the proportion of true we are capturing using our algorithm.
- ✓ TPR (True Positive Rate) = Sensitivity = Recall = TP/(TP+FN) and False positive rate or FPR is just the proportion of false we are capturing using our algorithm. Fpr (False Positive Rate)= 1- Specificity = FP/(TN+FP)



3.1.4. Hence, as the Claimed Yes status is our focus, we may conclude that the following metrics are more important for our model in the same order as below*:*

True Positives – as it gives the count of all correctly claimed Yes cases

False Negatives – as this gives us the count for case which are claimed Yes but classified No

Recall – as this will give us the proportion of correct Yes cases from all the actual Yes cases

Accuracy – since our dataset is only moderately biased (aprox. 'No':'Yes' is 2:1), we may use accuracy too to evaluate our model

AUC Curve – this metrics can always be helpful as it gives an indication on how the data is classified between True Positives and False Positives

F1 score –this metrics is mostly important where both Type I error (FP) and Type II error (FN) have same significance in prediction; hence, we may just consider this metrics for information.

3.2. Performance of prediction for Decision Tree / CART model

Below are the two best models for CART after few iterations.

We have produced the *train and test sets of parameters for our value of interest which is '1' meaning Claimed-Yes*

CART Model	Grid_params	Best_Params	Data	Precision	Recall	f1_score	AUC	Accuracy _score
Model 1	params = {'max_depth':[2,3,4,5,6,7], 'min_samples_leaf':[50,60,70,90,110], 'min_samples_split':[150,180,210,270,330]}	Best params: {'max_depth': 3, 'min_samples_leaf': 50, 'min_samples_split': 150}	Train	0.64	0.63	0.64	0.79	0.77
			Test	0.65	0.62	0.63	0.78	0.77
Model 2	params = {'max_depth':[2,3,4,5,6,7], 'min_samples_leaf':[40,50,70,90,110], 'min_samples_split':[120,150,210,270,330]}	Best params: {'max_depth': 3, 'min_samples_leaf': 40, 'min_samples_split': 120}	Train	0.65	0.62	0.63	0.79	0.78
			Test	0.66	0.63	0.64	0.79	0.77

	precision	recall	f1-score	support
0	0.83	0.84	0.83	1359
1	0.64	0.63	0.64	643
accuracy			0.77	2002
macro avg	0.73	0.73	0.73	2002
weighted avg	0.77	0.77	0.77	2002

	precision	recall	f1-score	support
0	0.83	0.85	0.84	588
1	0.65	0.62	0.63	271
accuracy			0.77	859
macro avg	0.74	0.73	0.74	859
weighted avg	0.77	0.77	0.77	859

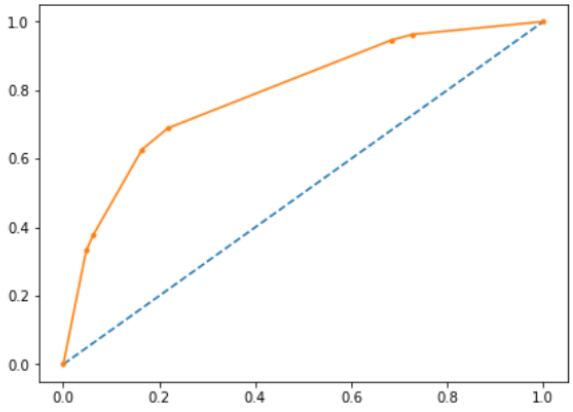
Model1 Classification Report

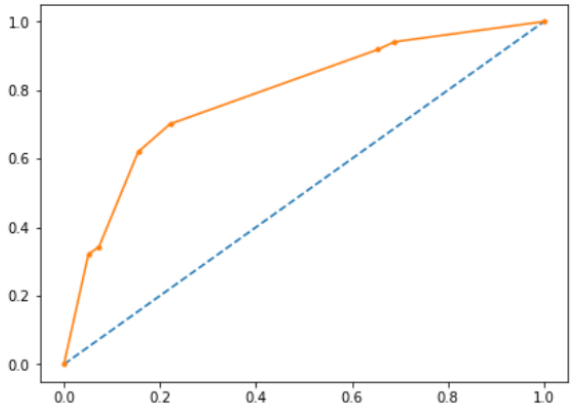
	precision	recall	f1-score	support
0	0.82	0.84	0.83	1359
1	0.65	0.62	0.63	643
accuracy			0.77	2002
macro avg	0.74	0.73	0.73	2002
weighted avg	0.77	0.77	0.77	2002

	precision	recall	f1-score	support
0	0.83	0.85	0.84	588
1	0.66	0.63	0.64	271
accuracy			0.78	859
macro avg	0.75	0.74	0.74	859
weighted avg	0.78	0.78	0.78	859

Model 2 Classification Report

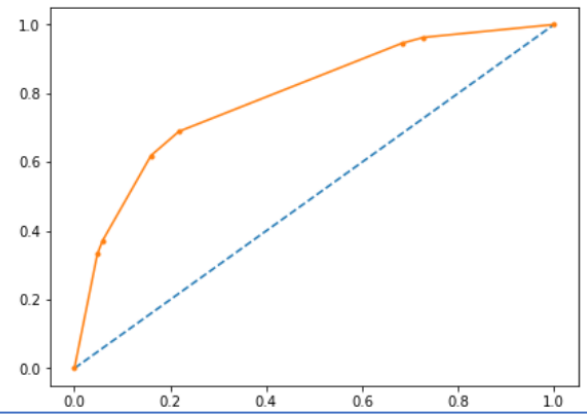
```
auc_train: 0.7895889050246213 , auc_test: 0.78357745312147
accuracy_test: 0.7741559953434226 , accuracy_train: (0.7687312687312687,)
```

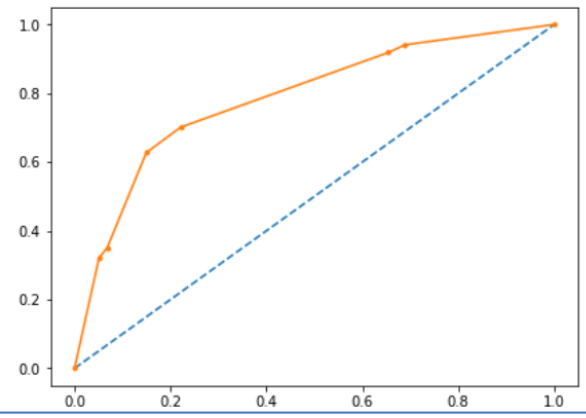




Model 1 AUC-ROC Curve

```
auc_train: 0.7898681333017484 , auc_test: 0.7861818158997916
accuracy_test: 0.779976717112922 , accuracy_train: (0.7692307692307693,)
```





Model 2 AUC-ROC Curve

Observations:

From the above two models, we may consider model 2 as better because:

Recall for train and test is comparable (difference by .01) in both models, although it is higher for train data in model 1 and higher for test data in model 2.

Precision and f1 score is also very close in both models for both train and test.

AUC and Accuracy score is slightly higher in model 2 for both test and train data.

Hence, we may go with model 2.

3.3. Performance of prediction for Random Forest Classifier model

CART Model	Grid_params	Best_Params	Data	Precision	Recall	f1_score	AUC	Accuracy _score

Model 1	params = {
	'max_depth':[8,10],
	'max_features':[5,7],
	'min_samples_leaf':[6,12],
	'min_samples_split':[18,36],
	'n_estimators':[75,150],
	}
		Best Params: {'max_depth': 8, 'max_features': 7, 'min_samples_leaf': 6, 'min_samples_split': 36, 'n_estimators': 75}	Train	0.73	0.6	0.66	0.86	0.78

			Test	0.69	0.57	0.63	0.82	0.80

Model 2	params = {
	'max_depth':[8,10],
	'max_features':[5,6,7],
	'min_samples_leaf':[4,6],
	'min_samples_split':[35,40],
	'n_estimators':[120,150],
	}
		Best Params: {'max_depth': 8, 'max_features': 6, 'min_samples_leaf': 4, 'min_samples_split': 35, 'n_estimators': 120}	Train	0.73	0.6	0.66	0.86	0.78

			Test	0.68	0.57	0.62	0.82	0.80

	precision	recall	f1-score	support
0	0.82	0.90	0.86	1359
1	0.73	0.60	0.66	643
accuracy			0.80	2002
macro avg	0.78	0.75	0.76	2002
weighted avg	0.79	0.80	0.79	2002

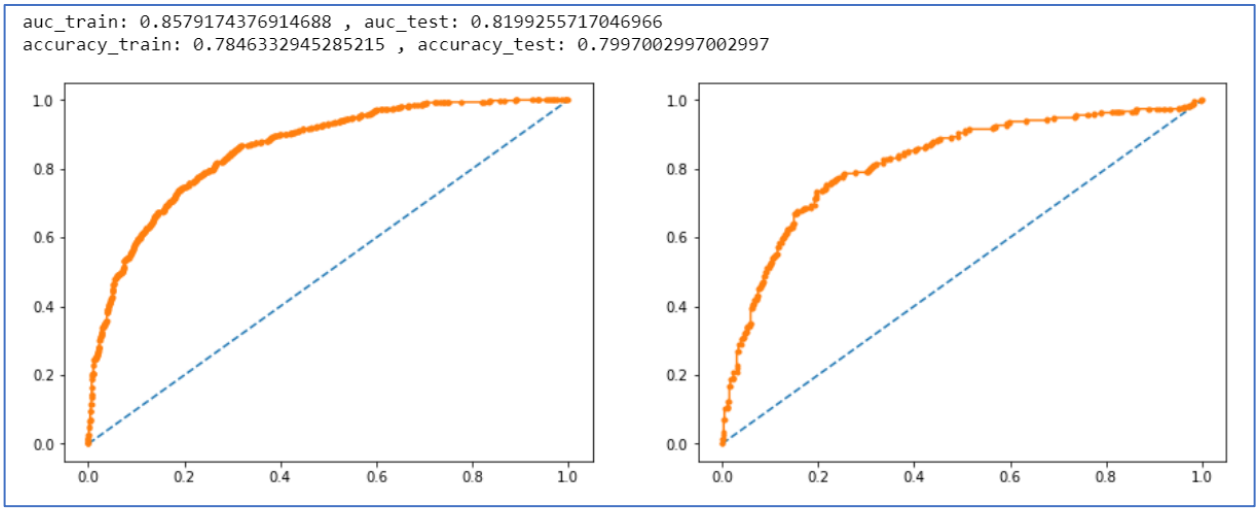
	precision	recall	f1-score	support
0	0.82	0.88	0.85	588
1	0.69	0.57	0.63	271
accuracy			0.78	859
macro avg	0.75	0.73	0.74	859
weighted avg	0.78	0.78	0.78	859

Model1 Classification Report

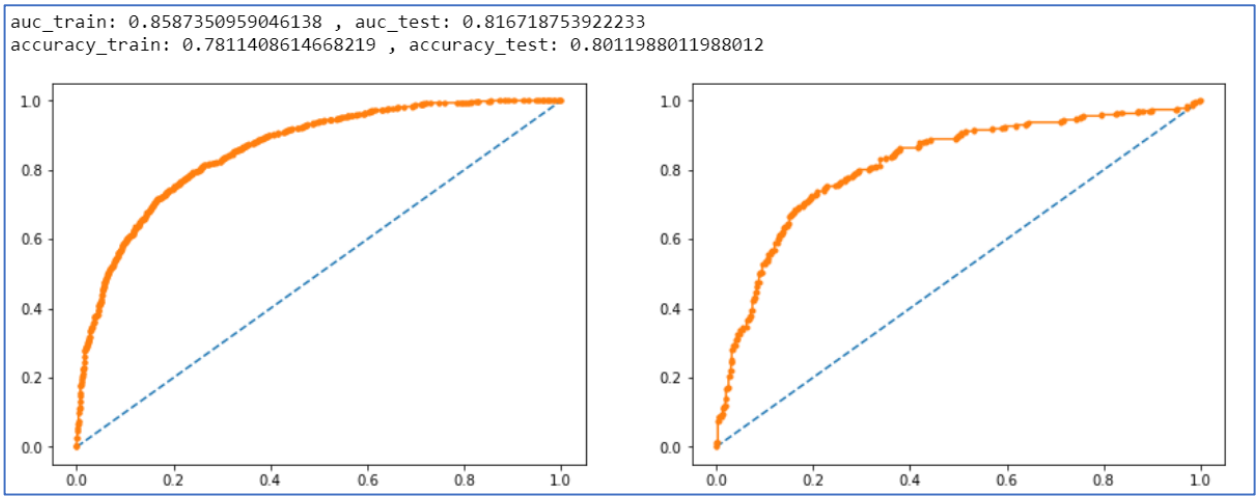
	precision	recall	f1-score	support
0	0.83	0.90	0.86	1359
1	0.73	0.60	0.66	643
accuracy			0.80	2002
macro avg	0.78	0.75	0.76	2002
weighted avg	0.80	0.80	0.80	2002

	precision	recall	f1-score	support
0	0.82	0.88	0.85	588
1	0.68	0.57	0.62	271
accuracy			0.78	859
macro avg	0.75	0.72	0.73	859
weighted avg	0.77	0.78	0.78	859

Model 2 Classification Report



Model 1 AUC-ROC Curve



Model 2 AUC-ROC Curve

Observations:

From the above two models, we may consider model 1 as better because:

Recall, AUC, and accuracy score is same for both model per train and test data.

F1 score – Since above 3 important metrics are same for both models, we can decide on f1 score which is slightly higher for model 1 than model 2

3.4. Performance of prediction for ANN model

CART Model	Grid_params	Best_Params	Data	Precision	Recall	f1_score	AUC	Accuracy _score
Model 1	params = {'hidden_layer_sizes':[(500,500,500)], 'max_iter':[5000], 'solver':['sgd','adam'], 'activation':['relu'], 'tol':[0.01]}	Best params: {'activation': 'relu', 'hidden_layer_sizes': (500, 500, 500), 'max_iter': 5000, 'solver': 'adam', 'tol': 0.01}	Train	0.63	0.69	0.66	0.82	0.77
			Test	0.62	0.73	0.67	0.81	0.77
Model 2	params = {'hidden_layer_sizes':[(400,400,400)], 'max_iter':[5000], 'solver':['sgd','adam'], 'activation':['relu'], 'tol':[0.01]}	Best params: {'activation': 'relu', 'hidden_layer_sizes': (400, 400, 400), 'max_iter': 5000, 'solver': 'adam', 'tol': 0.01}	Train	0.64	0.67	0.66	0.82	0.77
			Test	0.63	0.69	0.66	0.81	0.77

	precision	recall	f1-score	support
0	0.85	0.81	0.83	1359
1	0.63	0.69	0.66	643
accuracy			0.77	2002
macro avg	0.74	0.75	0.74	2002
weighted avg	0.78	0.77	0.77	2002

	precision	recall	f1-score	support
0	0.86	0.79	0.83	588
1	0.62	0.73	0.67	271
accuracy			0.77	859
macro avg	0.74	0.76	0.75	859
weighted avg	0.79	0.77	0.78	859

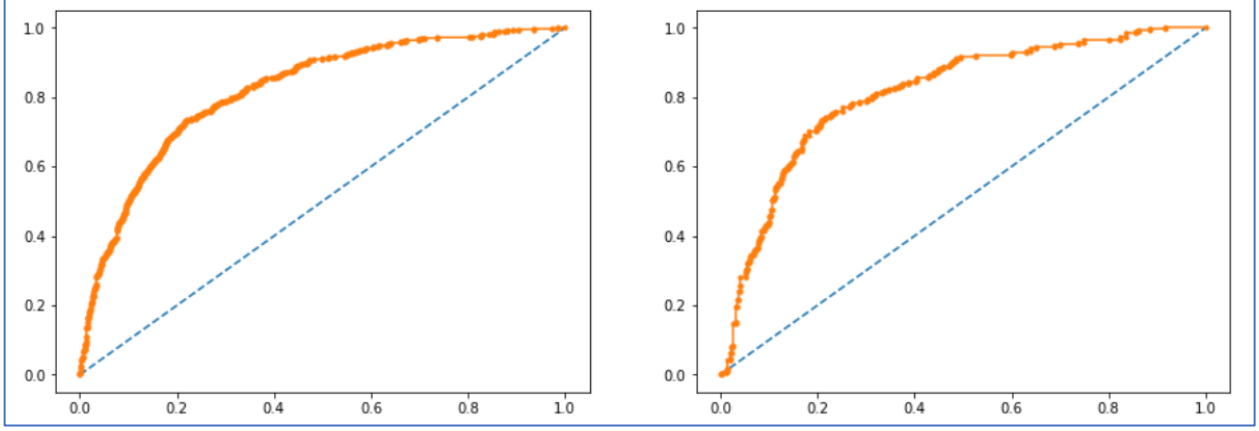
Model1 Classification Report

	precision	recall	f1-score	support
0	0.84	0.82	0.83	1359
1	0.64	0.67	0.66	643
accuracy			0.77	2002
macro avg	0.74	0.75	0.74	2002
weighted avg	0.78	0.77	0.78	2002

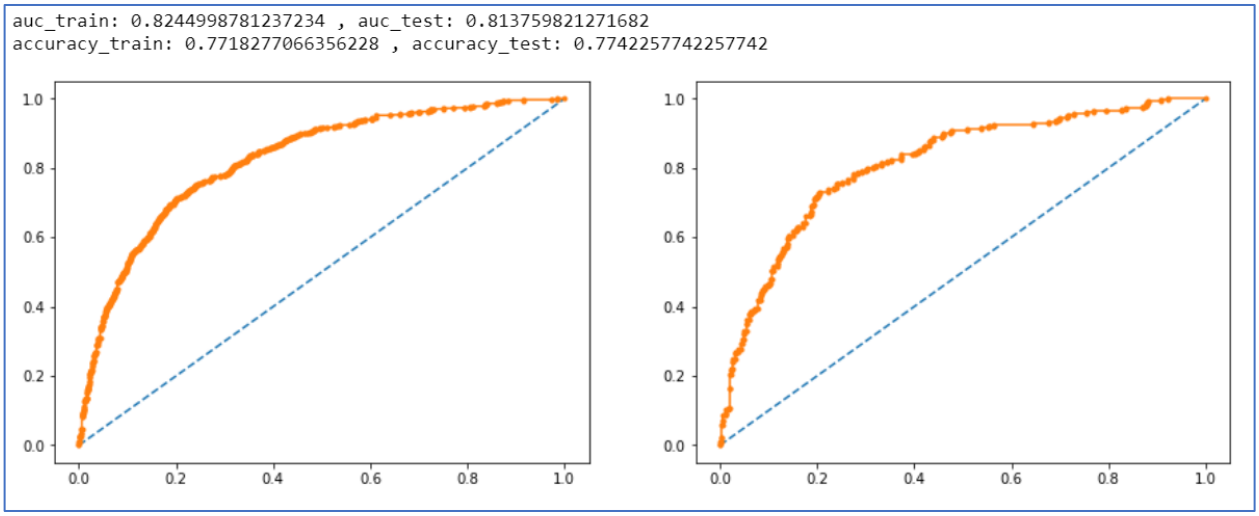
	precision	recall	f1-score	support
0	0.85	0.81	0.83	588
1	0.63	0.69	0.66	271
accuracy			0.77	859
macro avg	0.74	0.75	0.74	859
weighted avg	0.78	0.77	0.77	859

Model 2 Classification Report

```
auc_train: 0.8207508951898352 , auc_test: 0.8123791952205237
accuracy_train: 0.7718277066356228 , accuracy_test: 0.7697302697302697
```



Model 1 AUC-ROC Curve



Model 2 AUC-ROC Curve

From the above two models, we may easily distinguish model 1 as better because:

Recall for model 1 for both train and test data is higher by few points than model 2.

In this case, we do not have to look into other metrics as our most important metrics is recall / sensitivity which is making clear difference in two models.

Besides, we may observe that even the precision values are also comparable to recall values, meaning the errors are probably balanced in this case.

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

4.1. Model Comparison:

Below are the best three models from each technique:

CART Model	Grid_params	Best_Params	Data	Precision	Recall	f1_score	AUC	Accuracy _score
Model 2 CART	params = {'max_depth':[2,3,4,5,6,7], 'min_samples_leaf':[40,50,70,90,110], 'min_samples_split':[120,150,210,270,330] }	Best params: {'max_depth': 3, 'min_samples_leaf': 40, 'min_samples_split': 120}	Train	0.65	0.62	0.63	0.79	0.78
			Test	0.66	0.63	0.64	0.79	0.77
Model 1 RFC	params = { 'max_depth':[8,10], 'max_features':[5,7], 'min_samples_leaf':[6,12], 'min_samples_split':[18,36], 'n_estimators':[75,150], }	Best Params: {'max_depth': 8, 'max_features': 7, 'min_samples_leaf': 6, 'min_samples_split': 36, 'n_estimators': 75}	Train	0.73	0.6	0.66	0.86	0.78
			Test	0.69	0.57	0.63	0.82	0.80
Model 1 ANN	params = {'hidden_layer_sizes':[(500,500,500)], 'max_iter':[5000], 'solver':['sgd','adam'], 'activation':['relu'], 'tol':[0.01]}	Best params: {'activation': 'relu', 'hidden_layer_sizes': (500, 500, 500), 'max_iter': 5000, 'solver': 'adam', 'tol': 0.01}	Train	0.63	0.69	0.66	0.82	0.77
			Test	0.62	0.73	0.67	0.81	0.77

From the above three model, as we may see the **Recall** for **ANN** is highest for both train and test data, followed by CART and least in case of RFC, **we would recommend the ANN model for the prediction.**

Also, the accuracy score is same for both test and train data for this model, which proves that the model is balance on both.

Besides, AUC curve value is also .82 for train and .81 for test which is good as tpr and fpr are balanced in the model.

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

The objective of this model is to predict customers who might claim in future and the predictors here are all sales and channel based. Thus, this model can help in controlling the issues better by making changes in policy applications / regulation.

Referring below, a few flags which appeared both in Eda and in feature importance parameters, which we may analyze one by one to improve the business problem:

	imp
Sales	0.254349
Duration	0.251523
Age	0.172918
Agency_Code_C2B	0.164117
Commision	0.071581
Destination_EUROPE	0.013830
Product Name_Cancellation Plan	0.013030
Product Name_Silver Plan	0.010117
Product Name_Bronze Plan	0.008055
Destination_ASIA	0.007807
Product Name_Customised Plan	0.007314
Product Name_Gold Plan	0.007038
Destination_Americas	0.003535
Channel_Offline	0.003368
Agency_Code_CWT	0.003087
Agency_Code_EPX	0.002734
Type_Airlines	0.002011
Channel_Online	0.001852
Agency_Code_JZI	0.001736
Type_Travel Agency	0.000000

5.1. Sales:

As Sales is the most important variable and we also know that sales with higher value are claimed more which in turn means Product name: Bronze, Silver and Gold Plans. Hence, the policies with higher values may be made more regulated by increasing number of checks before passing them

5.2. Duration:

For higher durations, it may be checked if policies are being taken for the reason that they are cheaper in home country and customers are going with an intention to use these policies while they stay away. E.g. people may decide to go abroad for longer duration possibly for some kind of treatment, in which case, some of their health expenses will be covered by the insurance taken in home country too.

5.3. Age

Similarly, more claims are for higher age groups indicates that people with growing age have more health support and financial security which they may get partially covered by investing in good insurance policy.

5.4. Agency Code C2B;

Agency CodeC2B: The agency C2B is a big red flag and it has appeared in the EDA that this agency sells only to Airlines, it deals more with high value policies, hence, it is advisable to investigate into the operations of this agency closely if there are any underhand processes going on with them.

5.5. Commission:

As commission in insurance industry is directly related to the Sales value, it can easily be controlled if Sales value is controlled

5.6. Product Name:

As the feature importance suggests, Cancellation, Bronze and Silver plans are red flags. These plans should be looked into from the aspect of their cost versus benefits to the customer. Also they may be looked into if these plans have too much of flexibility to encourage claims by the customer

Observations:

In general, changes are recommended at:

Procedural level so that sales channels and sales policy may be investigated e.g., who approves the policy and how the commission is assigned etc.
Regulatory level for inside policy lapses e.g., the clauses of a particular policy etc.