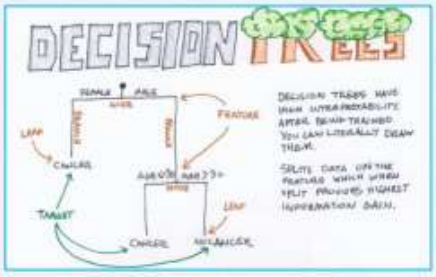
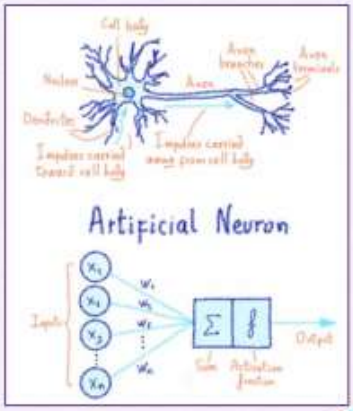
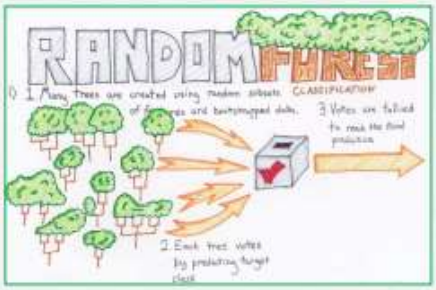
DATA MINING

Group Assignment

(Decision Tree, Random Forest & ANN)







Lavanya

Nikhil

Rekha

Rajiv

Submission Dt:

Feb 14th, 2021

GREAT LAKES
INSTITUTE OF MANAGEMENT

greatlearning
Power Ahead

Table of Contents

About Us

This section will cover brief introduction about us. We are seasoned professionals with diverse experience.

Rajiv

Am a CA with almost 20 years of experience of strengthening the governance environment of different multinational companies. Am currently working as **Vice President & Group Head of Internal Audits** for Flipkart, Myntra, PhonePe, e-Kart and Walmart India.

Rekha

Overall **20+ years** of corporate and management experience. Currently (from 2018) working as **Strategy Consultant (Independently)** for firms and companies that range from hospitality, manufacturing, and retail industries. Before moving to an independent role was with **KPMG Global Services for 12 years**. Hands-on manager with expertise in accounting systems development, fiscal management. and financial reporting. Proven record of developing and implementing financial and operational controls that improve P&L scenario and competitively position firms.

Nikhil

Overall **13+ years of cross functional experience** in FMCG and Alcobev industry. Working with **Diageo PLC** since 2017, looking after **Data Analytics CoE for Global Audit & Risk department**. Before to that worked with **Coca-Cola for 10 years** and completed stint in operations, supply chain (direct & indirect), Master data maintenance (SAP cross functional role) and Data analytics (Internal Audit).

Lavanya

Working with **Ernst & Young** as a Manager, having **13 years of techno-functional experience** in SAP Fiori, Embedded Analytics,SCP, ABAP on HANA and also having SAP Consulting experience on UX design strategy, RFP's/Pursuits and solutioning. During this tenure, worked in **various SAP ERP implementation projects spanning across Manufacturing, Healthcare, Retail and FMCG domains**. Prior to this, I have worked with **Accenture, Caterpillar and Wipro**.

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.

Note: All the analysis and workings of questions in the problem statement is detailed in the Jupyter Notebook.

Data Dictionary:

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)

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

Exploratory Data Analysis (EDA)

EDA is performed to understand the data first and try to gather as many insights from it. In short EDA is all about checking the dataset before the same is used to make any predictive model. EDA is a critical process for performing initial investigations on data so as to discover patterns, to spot anomalies and to check assumptions with the help of summary statistics and graphical representations.

Below is the Exploratory Data Analysis of the data set insurance_part2_data.csv of our problem statement .

Variable Summary and Status:

Steps and Results:

- Import libraries: as first step, we imported all possible libraries which may be required for the entire project, viz:
    - Numpy, pandas, seaborn
    - Scipy, pyplot
    - Sklearn- train test split, standard scaler, decision tree classifier, RF classifier, Grid Search CV, MLP classifier, tree, auc, roc curve and confusion matrix.

- Load and read the data set “insurance_part2_data.csv”.

```
: # load the dataset
  df = pd.read_csv('insurance_part2_data.csv')
```

- Understand the dataset by displaying the first few rows *(code: 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

- Analysing the dimension of the dataset and the data type of variables in the dataset *(code: df.shape() & df.info())*

```
The number of columns (variables) in the dataset is 10
The number of rows (observations per variable) in the dataset is 3000
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 10 columns):
Age             3000 non-null int64
Agency_Code     3000 non-null object
Type            3000 non-null object
Claimed         3000 non-null object
Commision       3000 non-null float64
Channel         3000 non-null object
Duration        3000 non-null int64
Sales           3000 non-null float64
Product Name    3000 non-null object
Destination     3000 non-null object
dtypes: float64(2), int64(2), object(6)
memory usage: 234.5+ KB
```

- Checking for null value in the dataset *(code: 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
dtype: int64
```

- Checking for duplicates in the dataset *(code: df.duplicated())*

```
There are 139 records in the dataset which are duplicates
```

Insights from the above Results:

- There are 139 records in the dataset that are potential duplicates. These observations have been retained in the dataset as there is no unique identifier available in the dataset.
- No Null values in the dataset
- There are Six object datatype variables which has to be encoded to integer which is a pre-requisite for model building.
- The target (dependent) variable is **"Claimed"**
- The predictor (Independent) variables are:
    - Numerical - Age, Commission, Duration, Sales and
    - Categorical – Agency Code, Type, Claimed, Channel, Product Name, Destination.

Descriptive Analysis:

Steps for analysis:

- Generating the descriptive statistics summary to include both numerical and categorical. *(code : df.describe(include='all').T)*

Results :

	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

Insights from the above Results:

- There is a significant gap between mean and median for commission, duration and sales which gives the inference that there are outliers in the data.
- Unique values in categorical variables are:
    - Product Name - 5,
    - Agency Code - 4,
    - Destination - 3,
    - Type - 2 and
    - Channel - 2

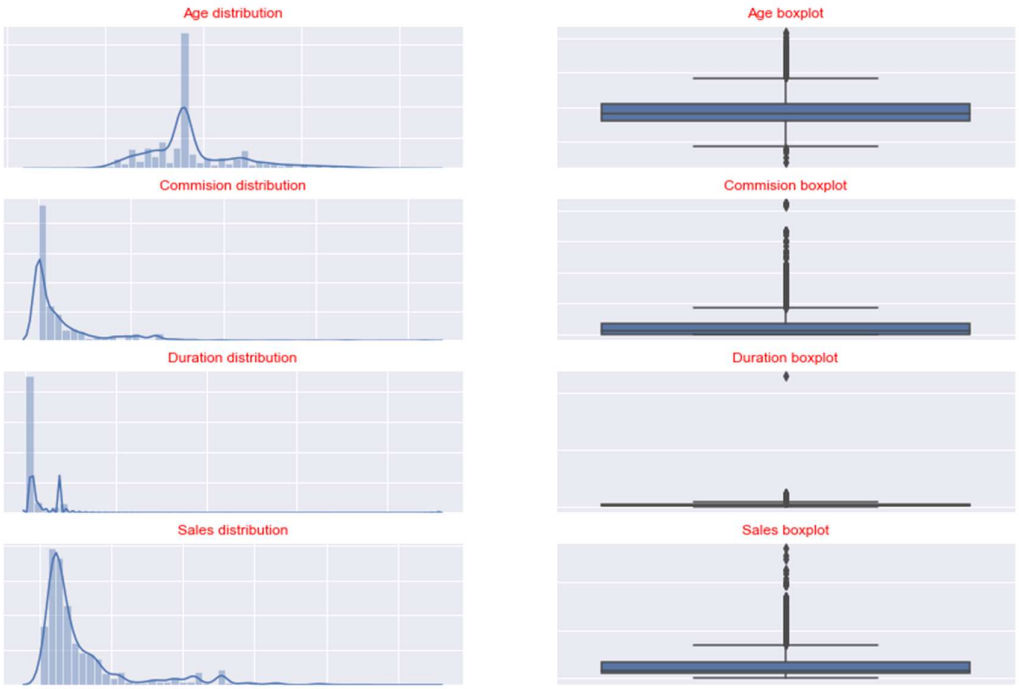
Univariate Analysis (Numerical Variables)

Steps for analysis:

- Generating distribution plot and boxplot for four numerical variables.

- Analysis of the skewness of the four numerical variables. *(code: df.skew())*

Results:



Insights from the above Results:

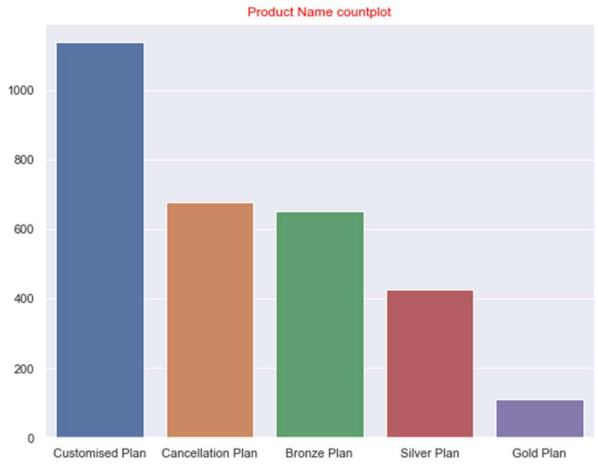
- Commission, Sales and Duration variables are not normally distributed; however, variable Age seems to be closer to normal distribution.
- Box plot implies that all the four variables have outliers.

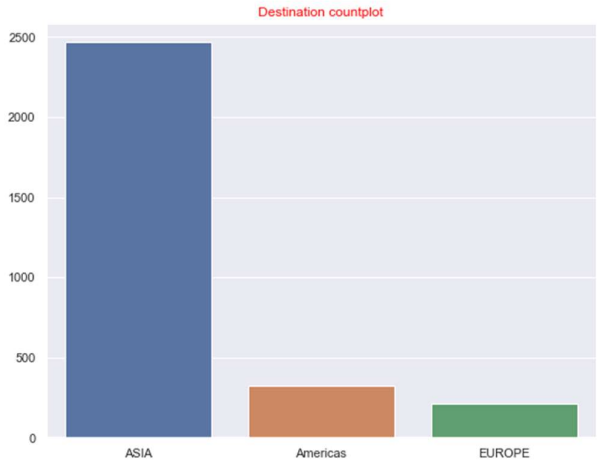
Univariate Analysis (Categorical Variables)

Steps for analysis:

- Generating count plot for five categorical variables.

Results:





<u>Insights from the above Results:</u>

- **EPX Agency** has the highest count followed by C2B, CWT and JZI.
- **Travel Agency** type of insurance firm has the highest count followed by airlines.
- **Online** is the distribution channel having the more contribution followed by offline.
- **Customised Plan** seems the in-demand plan followed by Cancellation plan, Bronze plan, Silver plan & Gold plan
- **Asia** is the destination which has more contribution followed by Americas & Europe.

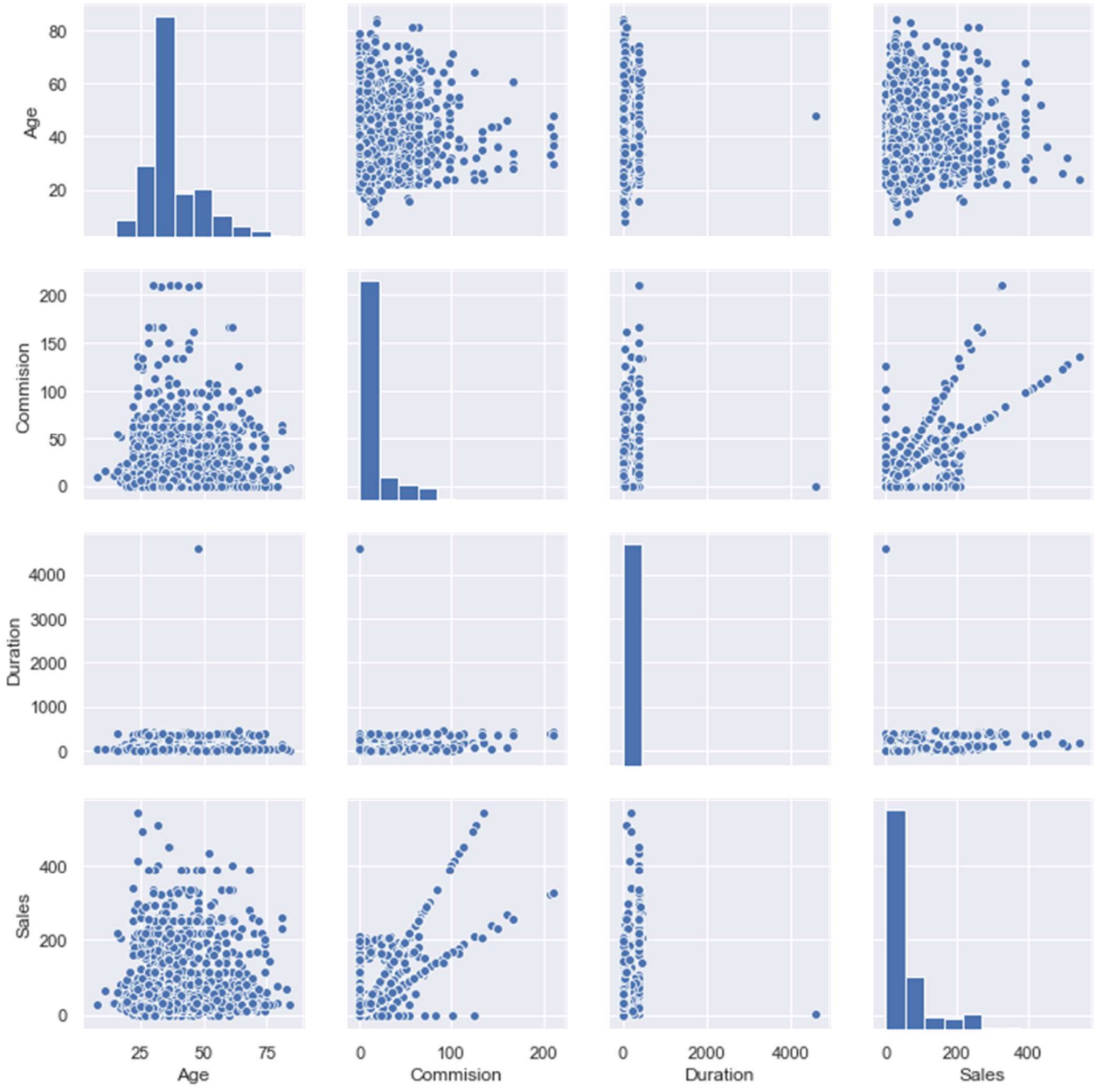
Bivariate Analysis (Numerical variables Vs Numerical Variables)

Steps for analysis:

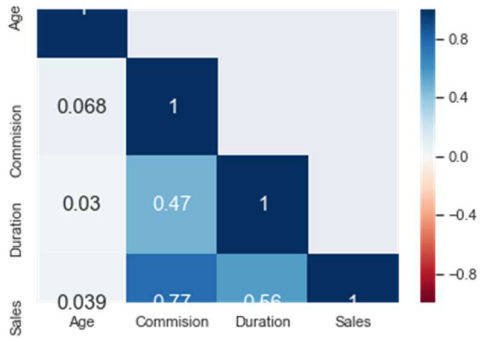
- Generating scatter plot. *(code : sns.pairplot())*
- Generating heat map. *(code: sns.heatmap())*

Results:

➢ Scatter Plot



- Heatmap



Insights from the above Results:

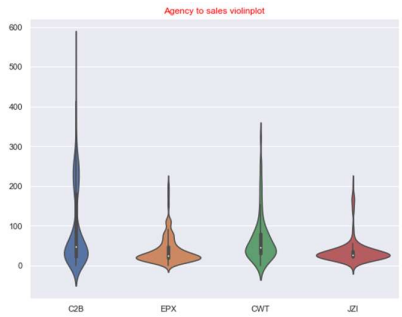
- Both scatter plot and heat map clearly indicate that Sales & Commission variables have a strong positive correlation (0.77). i.e., when commission increases the Sales increases.
- Other variables do not show a strong correlation.

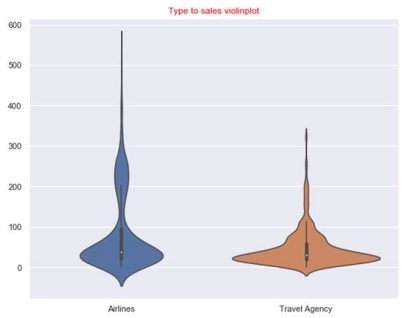
Bivariate Analysis (Categorical variables Vs Numerical Variables)

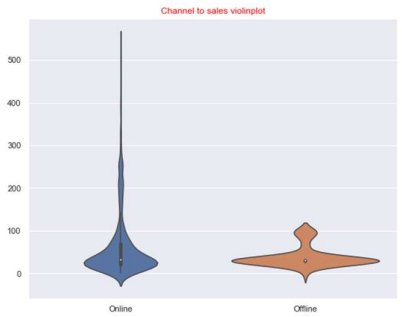
Steps for analysis:

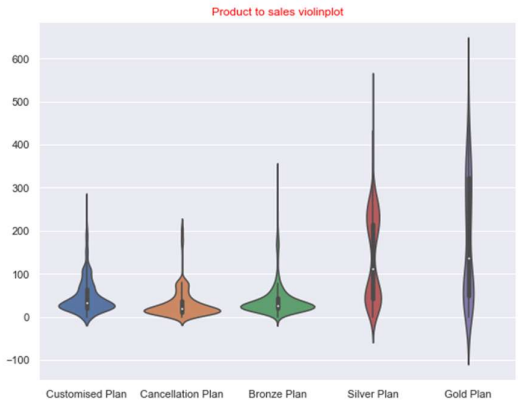
- Generating violin plot. (code : sns.violinplot())

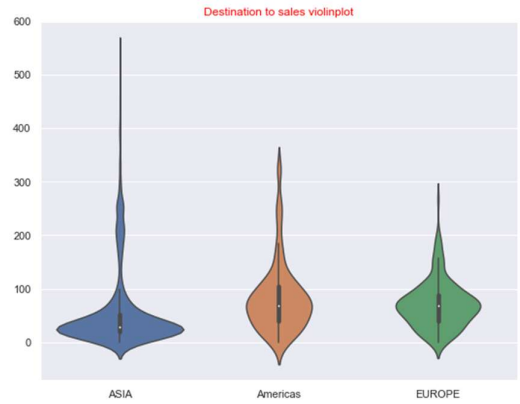
Results on all variables to “Sales”:









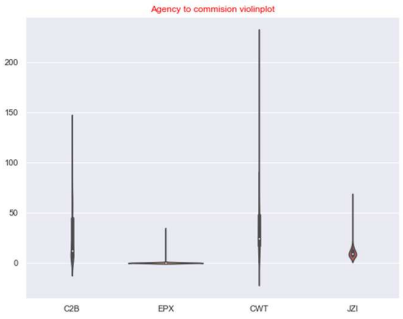


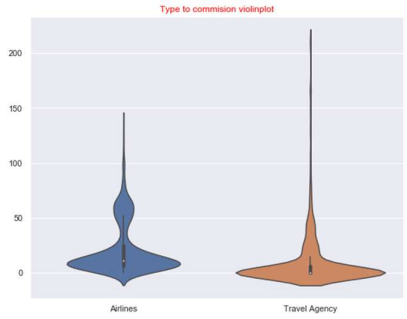
Insights from the above Results:

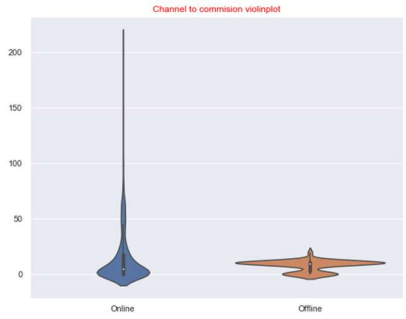
- **Median sales across agencies** are **not same**. Further, *C2B is the agency which has highest sales amongst others.*
- **Airlines** type of insurance firm has the **highest sales** but *median of both are almost similar.*
- **Online** distribution channel is the **quite preferred one**.
- **Gold & Silver plan** are the **key contributors to the sales.**
- **Asia** is the destination from **where sales are coming from**. Below is the summary of the total sales value from each destination that agrees with the graph results.

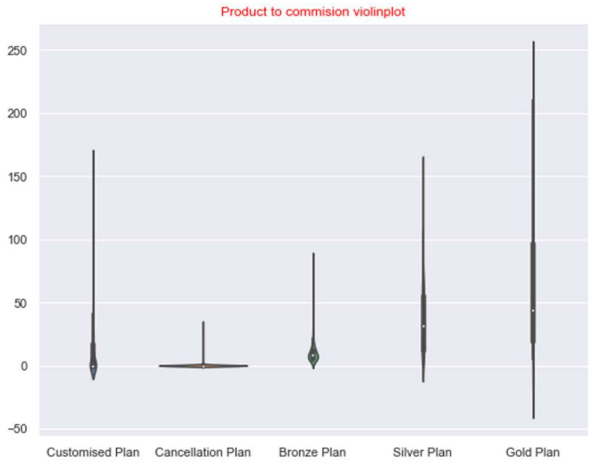
Destination	*Sales*
Asia	*1,39,192*
America	*26,423*
Europe	*15,133*

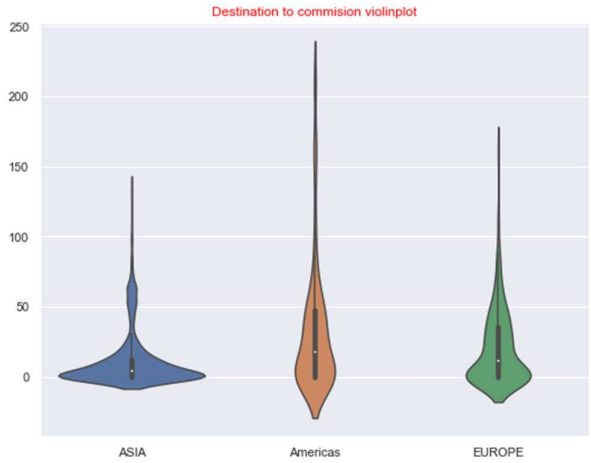
Results on all variables to “Commission”:







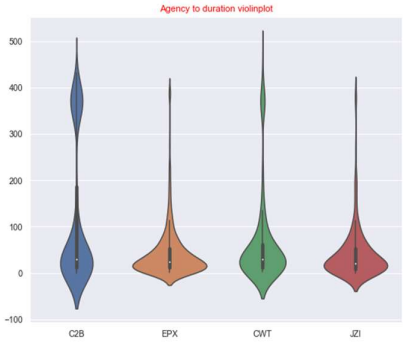


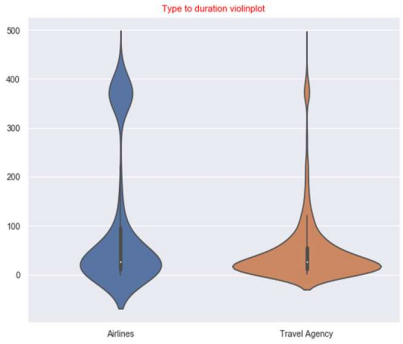


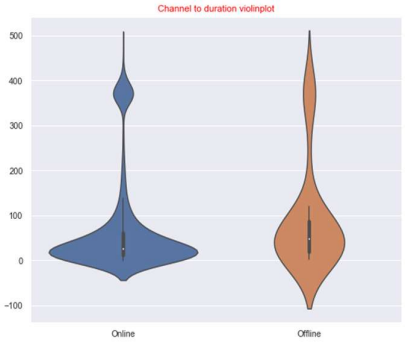
<u>Insights from the above Results:</u>

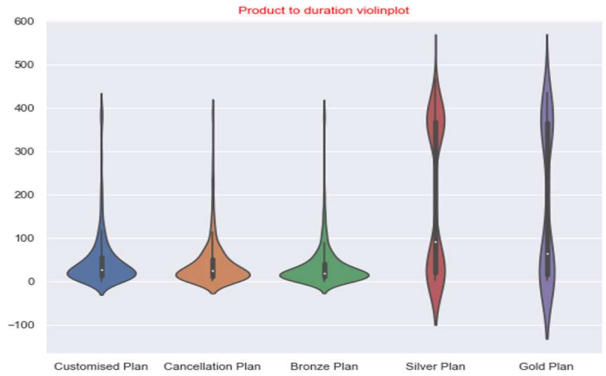
- **Median across agencies** are **not same**. Further, *CWT and C2B are the agencies have highest commissions amongst others.*
- **Airlines** type of insurance firm has the **highest commission**
- **Online** distribution channel is the **quite preferred one**.
- **Gold & Silver plan** are the **key contributors.**
- **Asia** is the destination from **where highest commissions is coming**.

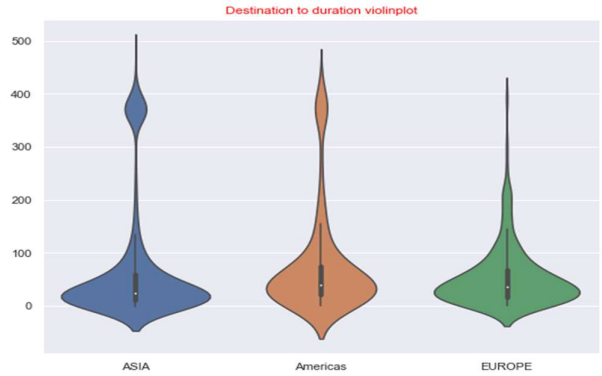
Results on all variables to “Duration”:











<u>Insights from the above Results:</u>

- There is **one record where duration is "4580" days** which has been **excluding for this visualisation** for a better clarity.
- **Median across agencies** are **almost same**.
- **Median across insurance firm type** are **almost same**. This infers that there is *no preferences for customer on duration days while choosing insurance firm type*
- **Online** distribution channel is the **quite preferred one**; however, it could be seen that the *median of online is lesser then offline*
- **Gold & Silver plan** are the products people preferred while **going for higher duration days**
- There is **a slight change in medians across destination.**

Target Variable Analysis – "Claimed"

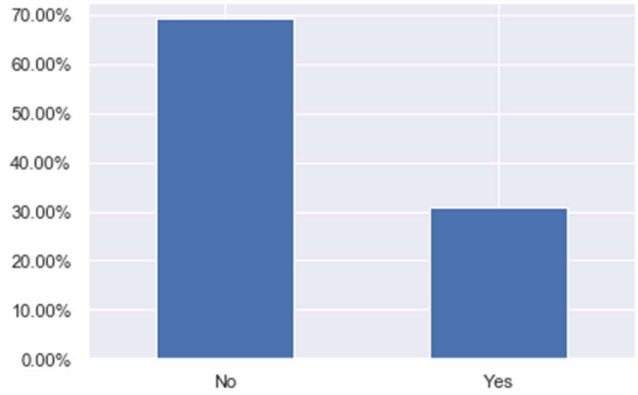
Steps for analysis:

- Computing the number of records of 'Yes' and 'No' claims.(code : df['Claimed'].value_counts())
- Computing the % of records of 'Yes' and 'No' claims.(code : df['Claimed'].value_counts(normalize=True))
- Generating bar chart for visualisation.

Results:

```
No     2076
Yes     924
Name: Claimed, dtype: int64
```

```
No     0.692
Yes    0.308
Name: Claimed, dtype: float64
```



Insights from the above Results:

- There is no issue of class imbalance in the dataset provided. We have reasonable proportions in both the classes for better analysis of model.

Actions & Assumptions

Based on the EDA the below are certain actions and assumptions that has been taken as we move towards building MODELS.

- We have Six object datatype variables which has to be encoded to Integer which is a pre-requisite for model building (Refer Variable Summary and Status EDA above).

- The descriptive summary analysis and the univariate analysis (boxplot) indicates that Age, Commission, Duration and Sales have outliers. In this case we have ignored the treatment of outliers for model building. However, in real scenario the outlier treatment needs to be taken care for better results after model building.

Q 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.

Step 1: Convert object variable types to categorical:

As firs step, we need to convert all the object variables into categorical. For that, we have used ***for loop*** with ***conditional action***.

Result:

```
feature: Agency_Code
[C2B, EPX, CWT, JZI]
Categories (4, object): [C2B, CWT, EPX, JZI]
[0 2 1 3]


feature: Type
[Airlines, Travel Agency]
Categories (2, object): [Airlines, Travel Agency]
[0 1]


feature: Claimed
[No, Yes]
Categories (2, object): [No, Yes]
[0 1]


feature: Channel
[Online, Offline]
Categories (2, object): [Offline, Online]
[1 0]


feature: Product Name
[Customised Plan, Cancellation Plan, Bronze Plan, Silver Plan, Gold Plan]
Categories (5, object): [Bronze Plan, Cancellation Plan, Customised Plan, Gold Plan, Silver Plan]
[2 1 0 4 3]


feature: Destination
[ASIA, Americas, EUROPE]
Categories (3, object): [ASIA, Americas, EUROPE]
[0 1 2]
```

And then we used.info function to check the data types after conversion:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 10 columns):
Age             3000 non-null int64
Agency_Code     3000 non-null int8
Type            3000 non-null int8
Claimed         3000 non-null int8
Commision       3000 non-null float64
Channel         3000 non-null int8
Duration        3000 non-null int64
Sales           3000 non-null float64
Product Name    3000 non-null int8
Destination     3000 non-null int8
dtypes: float64(2), int64(2), int8(6)
memory usage: 111.5 KB
```

Step 2: Data Split and Scaling

Dependent Variables and Independent Variables are put into Y and X dataset respectively.

Results:

X dataset – first 5 records

	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

Y dataset – top 5 rows:

```
0    0
1    0
2    0
3    0
4    0
Name: Claimed, dtype: int8
```

Step 3: Then we scaled the data using zscore function and looked at first few records using .head function.

Note*: Data Scaling is not a pre-requisite for CART and Random Forest; however, is a pre-requisite for Artificial Neural Network. In this case study, as we are building models using CART, RF and ANN, we have used scaled data for consistency.*

First 5 records from the scaled Data Set

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

Step 4: Then we split the data between train and test

This is done so that 70% of the dataset is used for building the model and remaining 30% used to test the model. We used the **train_test_split** function and then we used the **print** function to see if the split has happened as desired:

Dimension of the training and test data

```
X_train: (2100, 9)
Y_train: (2100,)
X_test: (900, 9)
Y_test: (900,)
Total Count: 3000
```

Note: *Data Split into Test and Train is consistent for all the recommended models build and therefore the step has been performed only once in this report.*

Model Building using Decision Tree (CART)

Step I: Define Decision Tree using CART algorithm using **decisiontreeclassifier** function with the splitting criteria for each node as 'gini' and 'random_state=1' as the parameter.

```
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')
```

Step II: Now to visualise the Decision Tree, we created a dot file which contains all the instructions on how to build this graphical visualization of the Classification Tree that we had built.

Then we used the browser → opened www.webgraphviz.com → pasted the content of the dot file and generated the graph. By seeing the visualisation, below are the key observations:

- Max depth of the tree is ~25
- Many of the leaf samples have their sample size as "1"
- Most of the samples split have their split after depth 6 is between 2 to 10

Step III: Now we will check the accuracy of our model on training and test data using ***.score*** function on both train and test datasets

For train dataset → score is `0.9947619047619047`

For test dataset → score is `0.7177777777777777`

Since there is significant difference in the accuracy score between the train and test data, this is a classic case of overfitting of the model. And hence tree pruning is required.

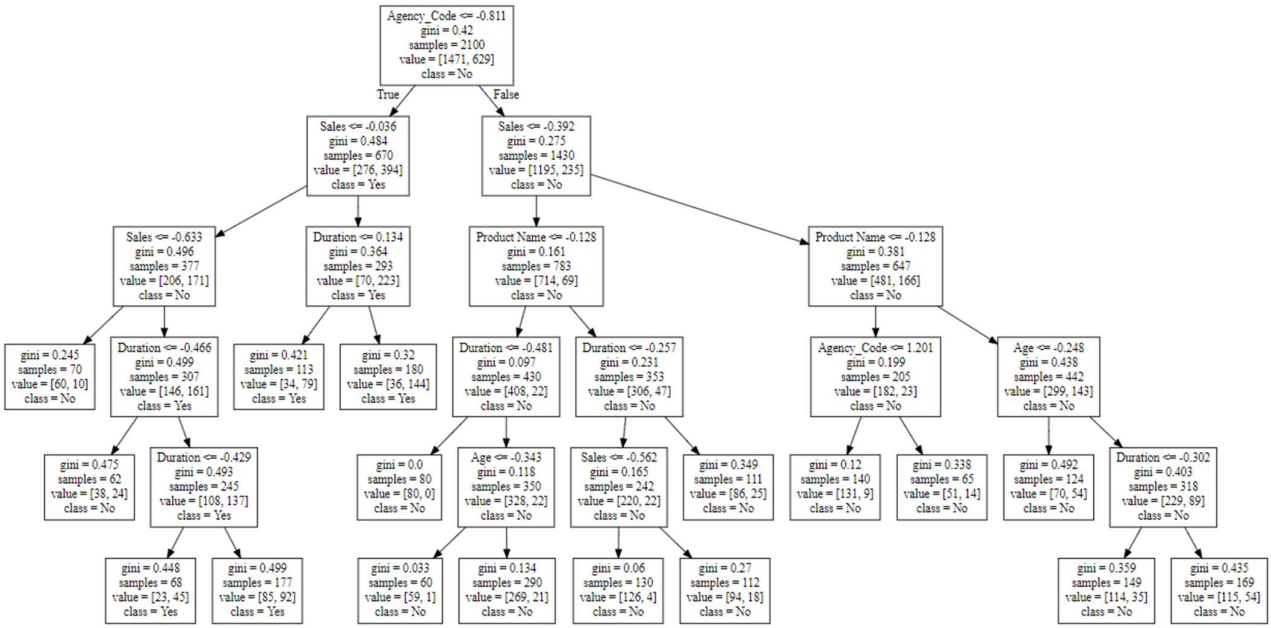
Step IV: Using **GridSearch** CV we can arrive at the best parameter combinations for Pruning the tree. This takes a few iterations. It took us 2 iterations to arrive at the best combination. And then we used the best grid function to know the best combination, as below:

	DT		
	initial	**iteration1**	**iteration2**
Train	99.47	78.7	78.7
Test	71.77	77.1	77.1

Since the difference between the accuracy of train and test dataset has significantly reduced in 1st and 2nd iterations, we have used the 2nd as the best model

Best Parameter for the model : **{'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 60, 'min_samples_split': 200}**

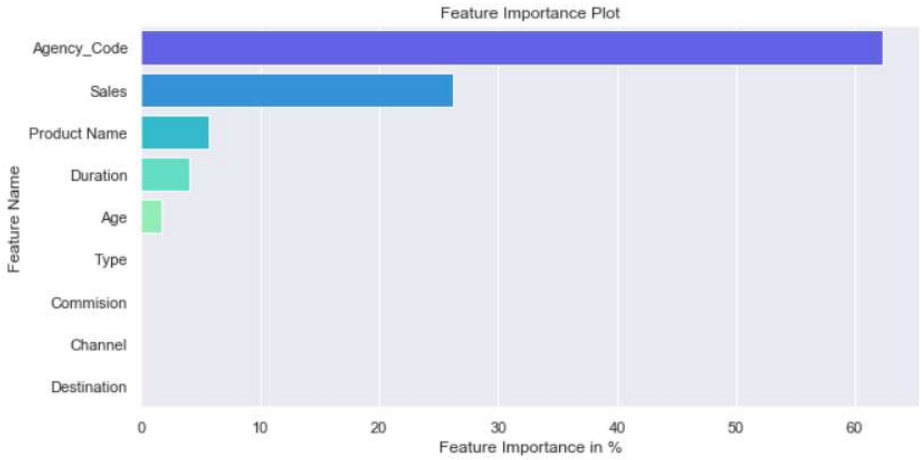
Step V: Visualization of the Decision Tree (using the best parameter determined above i.e. after pruning)



Step VI : Now check the importance of each feature in building the model using ***feature_importances_ function***. We then sorted the values in descending order:

	Importance
Agency_Code	0.622966
Sales	0.262609
Product Name	0.056775
Duration	0.040261
Age	0.017389
Type	0.000000
Commision	0.000000
Channel	0.000000
Destination	0.000000

And also prepared a **barplot** for better visualisation.



Step VII: Now we will make the predictions for both the train and test data and the probability values using the Pruned/Regularized Decision Tree.

- Predicting on Training and Test dataset and getting predicted classes

Getting the Predicted Classes

```
Y_train_predict_dt
```

```
array([0, 0, 1, ..., 0, 0, 1], dtype=int8)
```

```
Y_test_predict_dt
```

```
array([0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
```

And then to get predicted probabilities, we used .predict_proba function on both train and test datasets

Train dataset

	0	1
0	0.969231	0.030769
1	0.680473	0.319527

Test dataset

	0	1
0	0.935714	0.064286
1	0.480226	0.519774

Model Building using Random Forest (RF)

Step I : Build Random Forest model using RandomForestClassifier function and then fit the model using .fit function

It gives the following output:

```
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=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=501,
                       n_jobs=None, oob_score=True, random_state=1, verbose=0,
                       warm_start=False)
```

Step 2: We then checked the OOB score and OOB error rate

OOB score → 78.28571428571428

OOB error rate → 21.71428571428572

Step 3: We then checked the accuracy of the model on the train and test datasets

Accuracy on TRAIN data → 0.8947619047619048

Accuracy on TEST data → 0.7711111111111111

We see that the accuracy on the Training data is 0.89 and the accuracy on the test data is 0.77. This shows ~12% of gap between train and test data which is a case of ***overfitting***. We have allowed the Decision Trees inside the Random Forest algorithm to grow to their fullest. We need to regularize this particular Random Forest model to achieve comparative accuracy on both the Training and Test data.

We regularized the Decision Tree by looking at a largely overgrown tree. Here, in the case of Random Forest it is not possible to look at every tree and then go ahead and prune the trees. To make this job simpler we are going to use a command called GridSearchCV.

Step 4: Prune Random Forest using GridSearchCV function

We ran multiple iterations of GridSearch CV an checked the accuracy score of Train and Test data at each stage. The scores were as below:

	RF			
	initial	**iteration1**	**iteration2**	**iteration3**
Train	89.47	78.7	81	81.5
Test	77.11	75.3	77.1	77.5

Now, max_depth, max_features, min_samples_leaf, min_samples_split and n_estimators are looks like well frozen. So, we considered 3rd iteration as the best parameter for the Random Forest model.
We used best_estimator_ function to get the best parameters:

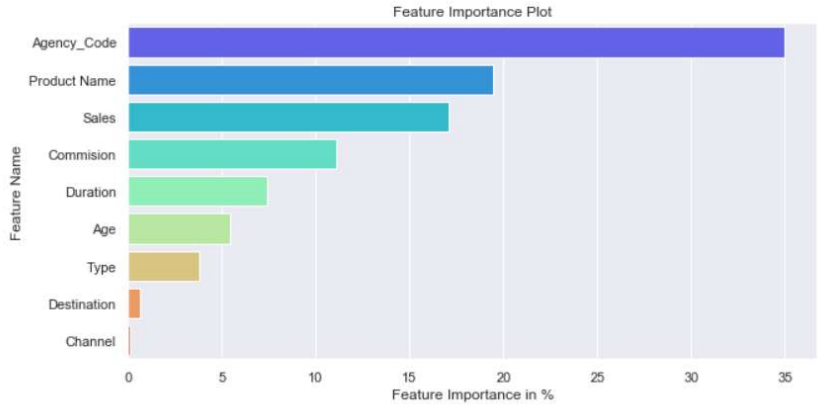
```
{'max_depth': 6, 'max_features': 5, 'min_samples_leaf': 7, 'min_samples_split': 40, 'n_estimators': 301}
```

Step 5: *Feature Importance: Training Data*

Now check the importance of each feature in building the model using ***feature importance function***. We then sorted the values in descending order:

```
              Importance
Agency_Code     0.349620
Product Name    0.194576
Sales           0.170558
Commision       0.111002
Duration        0.074119
Age             0.054613
Type            0.037946
Destination     0.006161
Channel         0.001405
```

And prepared a ***barplot*** for better visualisation.



Step 6 : Predicting on Training and Test dataset and getting predicted classes

Getting the Predicted Classes

```
Y_train_predict_rf
```

```
array([0, 0, 1, ..., 0, 0, 0], dtype=int8)
```

```
Y_test_predict_rf
```

```
array([0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
```

And then to get predicted probabilities, we used .predict_proba function on both train and test datasets

Train dataset

	0	1
0	0.884733	0.115267
1	0.733226	0.266774

Test dataset

	0	1
0	0.760362	0.239638
1	0.461013	0.538987

Model Building Artificial Neural Network (ANN)

Steps along with results:

Step I: Scaling the data is pre-requisite for building Neural Network. This step has already been performed and so not repeated.

Step 2: We defined ANN model using MLPClassifier function and then fitted the model using .fit function

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=100, learning_rate='constant',
              learning_rate_init=0.001, max_iter=500, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=1, shuffle=True, solver='sgd', tol=0.001,
              validation_fraction=0.1, verbose=True, warm_start=False)
```

Step 3: We then checked the accuracy of the model on Train and Test data using .score function

Accuracy on Train data → 0.7704761904761904

Accuracy on Test data → 0.7422222222222222

We see that the training and test data gives comparable results but the training data gives slightly better results than the test data.

Step 4: We then looked for better parameters using GridSearch CV function

And it gave the following parameters

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=50, learning_rate='constant',
              learning_rate_init=0.001, max_iter=2500, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=1, shuffle=True, solver='adam', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

<u>Step 5: We re-checked the accuracy of the model on Train and Test data using .score function</u>

Accuracy on Train data → 0.7961904761904762

Accuracy on Test data → 0.7711111111111111

As the accuracy of the model on both the train and test data has come significantly closer, we have considered the best parameters from GridSearch for prediction.

<u>Step 6: Predicting on Training and Test dataset and getting predicted classes.</u>

Getting the Predicted Classes

```
Y_train_predict_nn
```

```
array([0, 0, 1, ..., 0, 0, 1], dtype=int8)
```

```
Y_test_predict_nn
```

```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
```

Step 7: And then to get predicted probabilities, we used .predict_proba function on both train and test datasets

Train dataset

	0	1
0	0.869161	0.130839
1	0.589695	0.410305

Test dataset

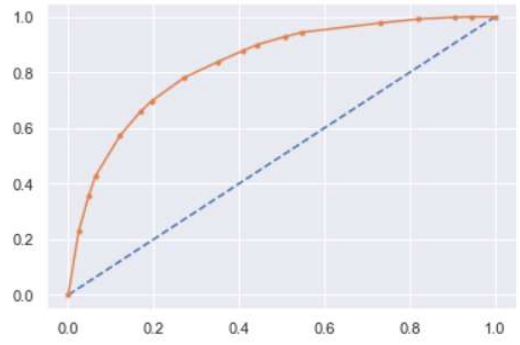
	0	1
0	0.921467	0.078533
1	0.529627	0.470373

Q. 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.

Performance Metrics – CART

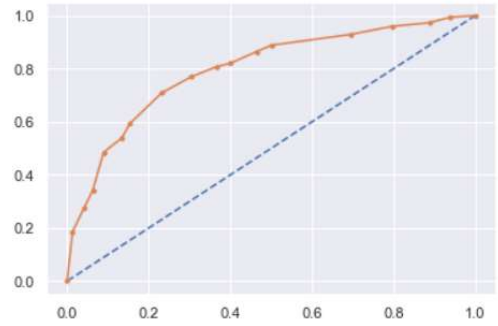
ROC_AUC score – Training Data

```
AUC Training Data for Decision Tree: 0.833
```



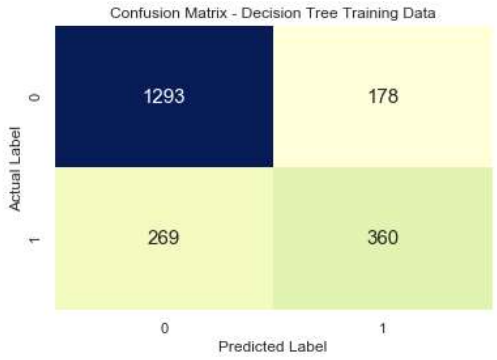
ROC_AUC score – Test Data

```
AUC Test Data for Decision Tree: 0.798
```



Confusion_Classification Report – Training Data

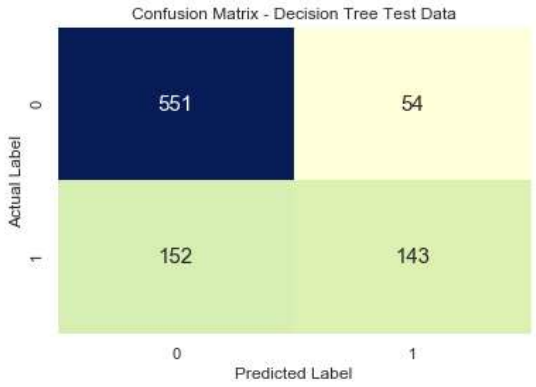
```
array([[1293,  178],
       [ 269,  360]], dtype=int64)
```



	precision	recall	f1-score	support
0	0.83	0.88	0.85	1471
1	0.67	0.57	0.62	629
accuracy			0.79	2100
macro avg	0.75	0.73	0.73	2100
weighted avg	0.78	0.79	0.78	2100

Confusion_Classification Report – Test Data

```
array([[551,  54],
       [152, 143]], dtype=int64)
```



	precision	recall	f1-score	support
0	0.78	0.91	0.84	605
1	0.73	0.48	0.58	295
accuracy			0.77	900
macro avg	0.75	0.70	0.71	900
weighted avg	0.76	0.77	0.76	900

Summary

Training Data:

- AUC : **83%**
- Accuracy : **79%**
- Recall : **88%**
- Precision : **83%**
- F1 score : **85%**

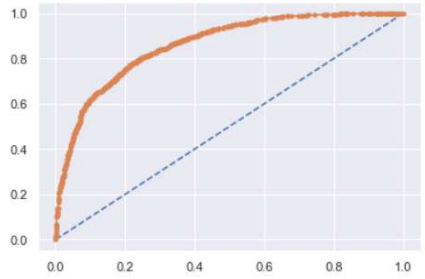
Test Data:

- AUC : **80%**
- Accuracy : **77%**
- Recall : **91%**
- Precision : **78%**
- F1 score : **84%**

Performance Metrics – Random Forest

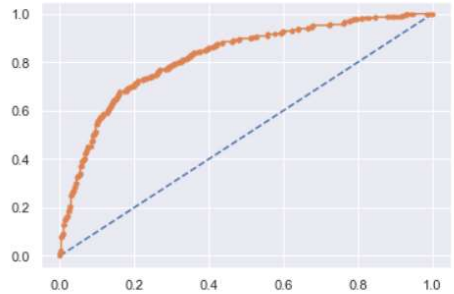
ROC_AUC score – Training Data

```
AUC Training Data for Random Forest: 0.863
```



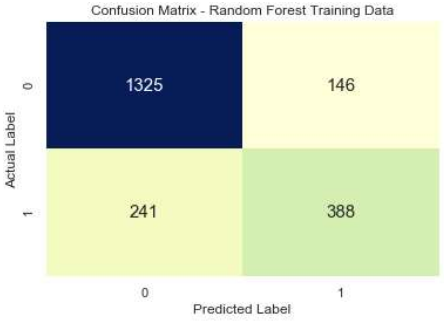
ROC_AUC score – Test Data

```
AUC Test Data for Random Forest: 0.821
```



Confusion_Classification Report – Training Data

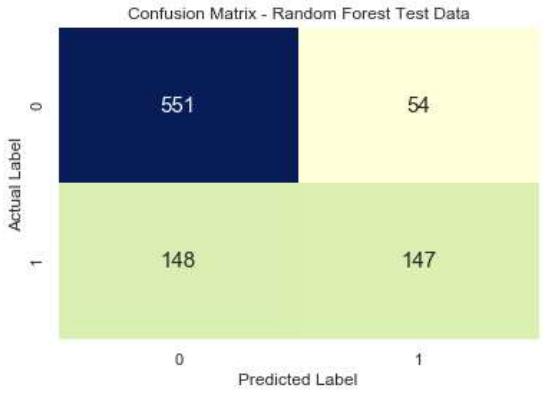
```
array([[1325,  146],
       [ 241,  388]], dtype=int64)
```



	precision	recall	f1-score	support
0	0.85	0.90	0.87	1471
1	0.73	0.62	0.67	629
accuracy			0.82	2100
macro avg	0.79	0.76	0.77	2100
weighted avg	0.81	0.82	0.81	2100

Confusion_Classification Report – Test Data

```
array([[551,  54],
       [148, 147]], dtype=int64)
```



	precision	recall	f1-score	support
0	0.79	0.91	0.85	605
1	0.73	0.50	0.59	295
accuracy			0.78	900
macro avg	0.76	0.70	0.72	900
weighted avg	0.77	0.78	0.76	900

Summary

Training Data:

- AUC : **86%**
- Accuracy : **82%**
- Recall : **90%**
- Precision : **85%**
- F1 score : **87%**

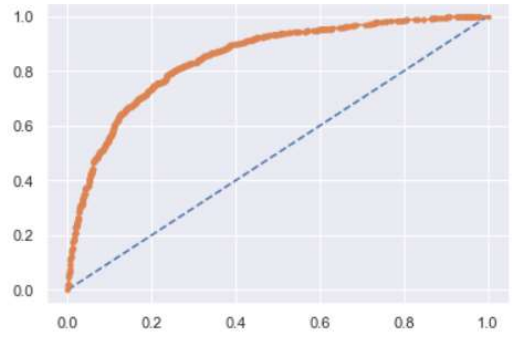
Test Data:

- AUC : **82%**
- Accuracy : **78%**
- Recall : **91%**
- Precision : **79%**
- F1 score : **85%**

Performance Metrics – Artificial Neural Network

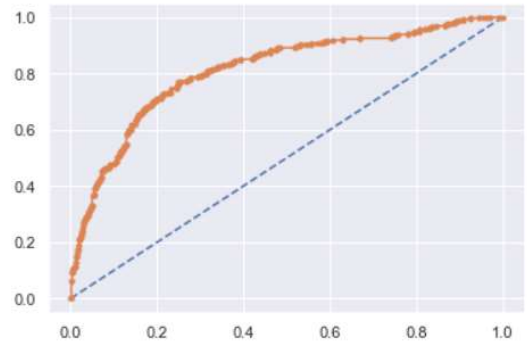
ROC_AUC score – Training Data

```
AUC Training Data for ANN: 0.848
```



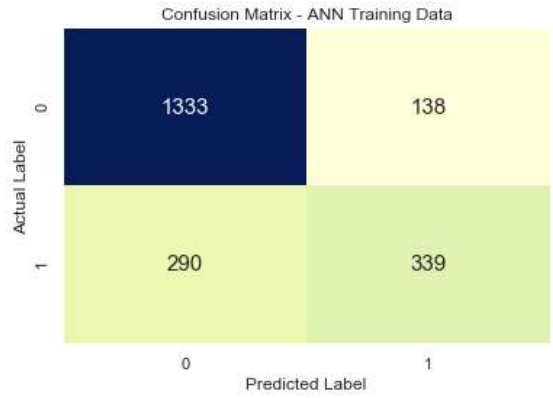
ROC_AUC score – Test Data

```
AUC Test Data for ANN: 0.812
```



Confusion_Classification Report – Training Data

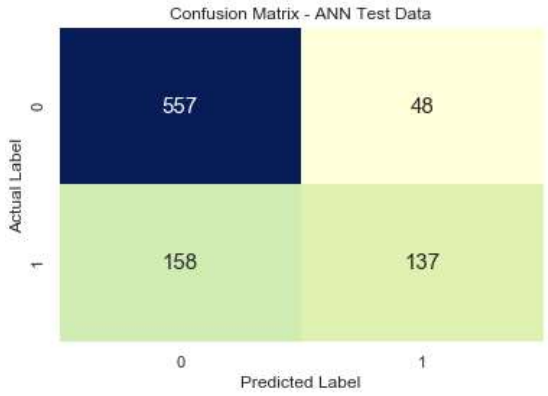
```
array([[1333,  138],
       [ 290,  339]], dtype=int64)
```



	precision	recall	f1-score	support
0	0.82	0.91	0.86	1471
1	0.71	0.54	0.61	629
accuracy			0.80	2100
macro avg	0.77	0.72	0.74	2100
weighted avg	0.79	0.80	0.79	2100

Confusion_Classification Report – Test Data

```
array([[557,  48],
       [158, 137]], dtype=int64)
```



	precision	recall	f1-score	support
0	0.78	0.92	0.84	605
1	0.74	0.46	0.57	295
accuracy			0.77	900
macro avg	0.76	0.69	0.71	900
weighted avg	0.77	0.77	0.75	900

Summary

Training Data:

- AUC : **85%**
- Accuracy : **80%**
- Recall : **91%**
- Precision : **82%**
- F1 score : **86%**

Test Data:

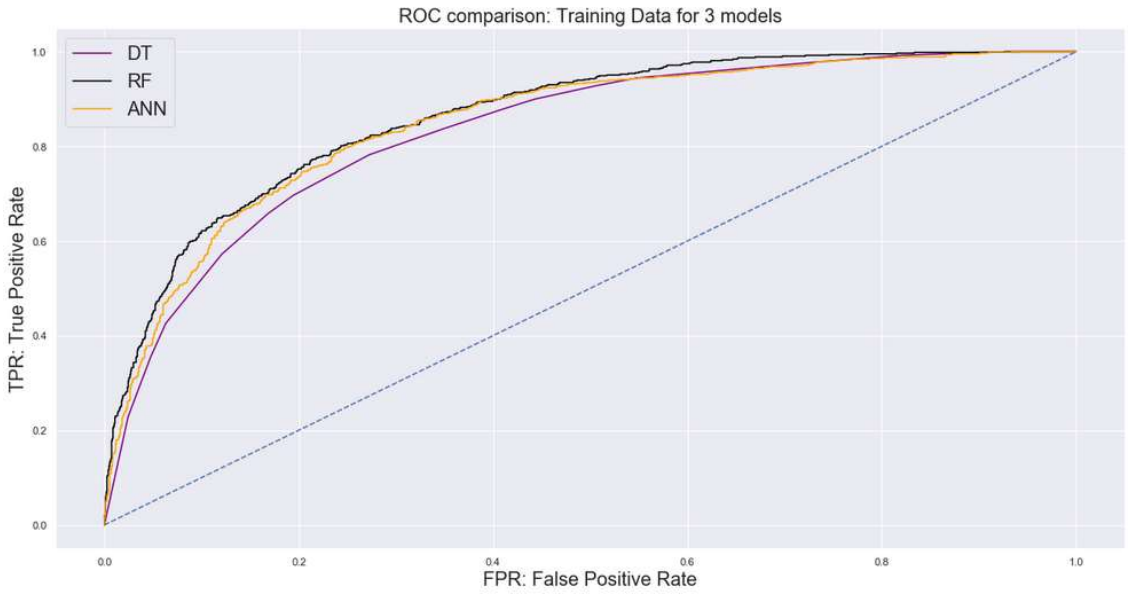
- AUC : **81%**
- Accuracy : **77%**
- Recall : **92%**
- Precision : **78%**
- F1 score : **84%**

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

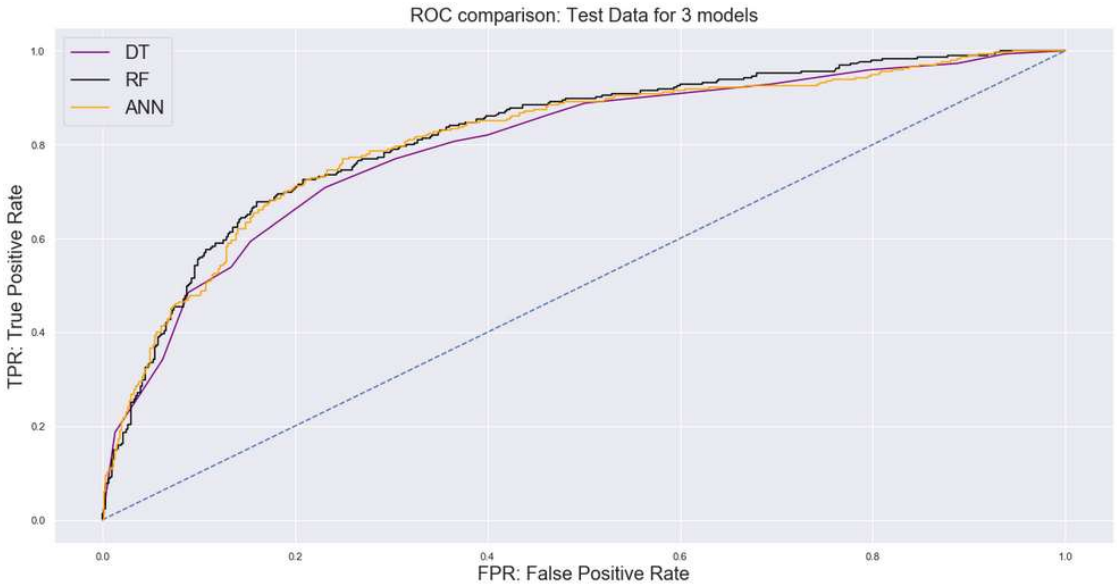
Based on above working let's compare 3 models. For that let's combine all key performance metrics into one dataframe and do the inferences

	DT Train	DT Test	RF Train	RF Test	ANN Train	ANN Test
AUC	0.83	0.80	0.86	0.82	0.85	0.81
Accuracy	0.79	0.77	0.82	0.78	0.80	0.77
Recall	0.88	0.91	0.90	0.91	0.91	0.92
Precision	0.83	0.78	0.85	0.79	0.82	0.78
F1 Score	0.85	0.84	0.87	0.85	0.86	0.84

We also made a comparison of ROC curve of all 3 models, train and test data, using pyplot:




```
AUC Training Data for DT: 0.833
AUC Training Data for RF: 0.863
AUC Training Data for ANN: 0.848
```




```
AUC Test Data for DT: 0.798
AUC Test Data for RF: 0.821
AUC Test Data for ANN: 0.812
```

Reviewing the scores (AUC and Accuracy) of all 3 models, it seems that the model built using Random Forest gives the best score.

- All the three models have a good AUC score; however, Random Forest score is high.
- The test score has a variation from the trained scores but is within the range of 5 % which is an acceptable variance considering the risk involved in insurance business.

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

RANDOM FOREST - SUMMARY SCORE				
	CLAIM	Precision	Recall	F1-score
TRAINED	AUC	0.86		
	0(NO)	0.85	0.90	0.87
	1(YES)	0.73	0.62	0.67
	Accuracy	0.82		
TEST	AUC	0.82		
	0(NO)	0.79	0.91	0.85
	1(YES)	0.73	0.50	0.59
	Accuracy	0.78		

In this assignment, we have used insurance data and based on that we built 3 models. After performing EDA and using all three models (CART, RF and ANN), mentioned below are the key insights and recommendations.

Business Insights:

- There is a significant gap between mean and median of numerical variables. Hence the data is skewed.
- Customised Plan seems the in-demand plan followed by Cancellation plan, Bronze plan, Silver plan & Gold plan.
- Gold & Silver plan are the key contributors to the sales.
- Duration and Sales were identified as the most important features followed by Age and Agency Code.
- Also, for the higher value of the duration chances for claims is also high.
- Maximum number of claims are from C2B Agency.
- Age of the insurer does not show any significant difference between claimed value ("Yes" or "No")
- As per the data 90% of insurance is done by online channel.
- In Sales variable, there are 53 observations which are Zero – means insurance product is sold for no cost, which is not an ideally situation but leaving untreated as of now since it has no impact in the performance of the model.
- There is no overfitting instance in any of the models since the test perform almost near to train dataset.
- Considering the business of travel insurance, predicting no claim when actually claim is raised is a serious issue to the insurance company and hence Type 2 error is costlier. Taking this into consideration, Recall is the important metric since it is inversely proportion to Type 2 error. Hence Random forest is the best model compared to others since Recall is high.
- AUC score gives us a positive indication that the claim predicted by the model is relevant/accurate by 86 % which helps the company to assess their claim liability based on the different variables.

Recommendations :

- For all the insurance policies wherein the duration of the tour is more than 350 or higher, business should analyse the reason for high claims and what are the changes to be made in the policy in order to reduce the claim frequency.
- C2B is the agency which is having more claims. A further analysis needs to be carried with C2B agency for finding the reasons for more claims.
- Need to check the JZI agency resources to raise up the sales as they are in bottom, need to run promotional marketing campaign.
- Age does not show any impact on the claim frequency. However, business should analyse this feature using more observations (data set) to understand the impact on claim frequency.
- As we are getting ~80% accuracy for selected model, so we need customer book airline tickets or plans and cross sell the insurance based on the claimed data pattern.
- Precision score also is a positive indicator and this could be improved by looking into the feature importance of the independent variable. The business would need to work on the features based on the importance and ignore irrelevant features. The below table indicates that Agency and the product has great relevance in the prediction. The firm need to therefore do an in-depth analysis based on the agency and product name.
- Based on the model we have built and EDA we performed, we recommend business should implement KPIs. (Key performance indicators) and mentioned below are the key criteria's:
    - Client Satisfaction - Client satisfaction is probably the best represented in client retention and policy renewal.
    - Reduce claims cycle time
    - Introduce more of flexi / customised packages
    - Optimize claims recovery
    - Reduce claim handling costs

With this we could conclude that insights gained from the dataset and data mining techniques could expand the boundaries of insurability, extend existing products, and give rise to new risk transfer solutions in areas like a non-damage business interruption and reputational damage.