Report-Data Mining

Tanushri Das

Table of contents:

Problem 1: Clustering

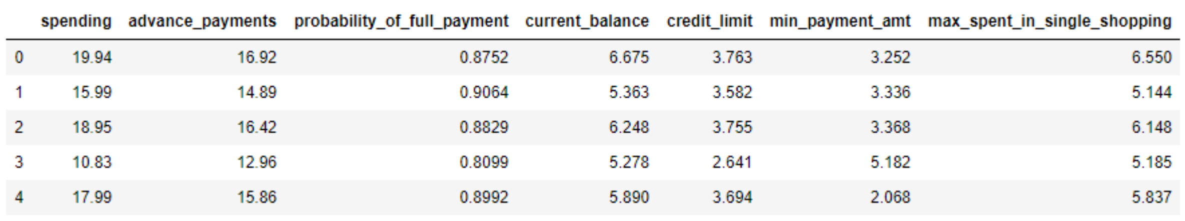
A leading bank wants to develop a customer segmentation to give promotional offers to its customers. They collected a sample that summarizes the activities of users during the past few months. You are given the task to identify the segments based on credit card usage.

1.1 Read the data and do exploratory data analysis. Describe the data briefly.

Ans)

The data consists of different credit card activities of customers of the given leading bank.

It has details of 210 customers with 7 attributes describing them. The first 5 entries of the dataset are as follows:

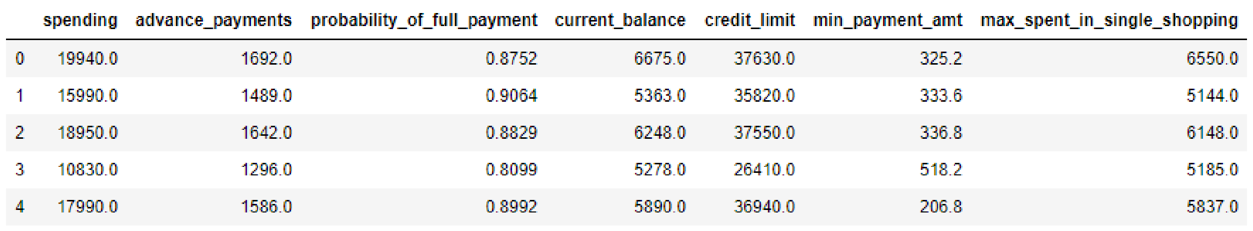


	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.550
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144
2	18.95	16.42	0.8829	6.248	3.755	3.368	6.148
3	10.83	12.96	0.8099	5.278	2.641	5.182	5.185
4	17.99	15.86	0.8992	5.890	3.694	2.068	5.837

The meaning of each attribute is as follows:

1. spending: Amount spent by the customer per month (in 1000s)
2. advance_payments: Amount paid by the customer in advance by cash (in 100s)
3. probability_of_full_payment: Probability of payment done in full by the customer to the bank
4. current_balance: Balance amount left in the account to make purchases (in 1000s)
5. credit_limit: Limit of the amount in credit card (10000s)
6. min_payment_amt : minimum paid by the customer while making payments for purchases made monthly (in 100s)
7. max_spent_in_single_shopping: Maximum amount spent in one purchase (in 1000s)

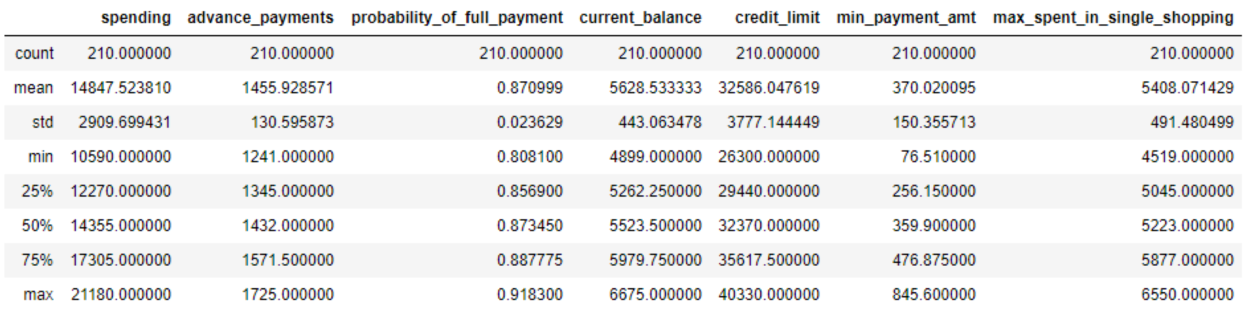
Bringing data to its original units and checking the head again:



	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
0	19940.0	1692.0	0.8752	6675.0	37630.0	325.2	6550.0
1	15990.0	1489.0	0.9064	5363.0	35820.0	333.6	5144.0
2	18950.0	1642.0	0.8829	6248.0	37550.0	336.8	6148.0
3	10830.0	1296.0	0.8099	5278.0	26410.0	518.2	5185.0
4	17990.0	1586.0	0.8992	5890.0	36940.0	206.8	5837.0

All the columns are of float datatype, and there are no null values and duplicate values present in the dataset.

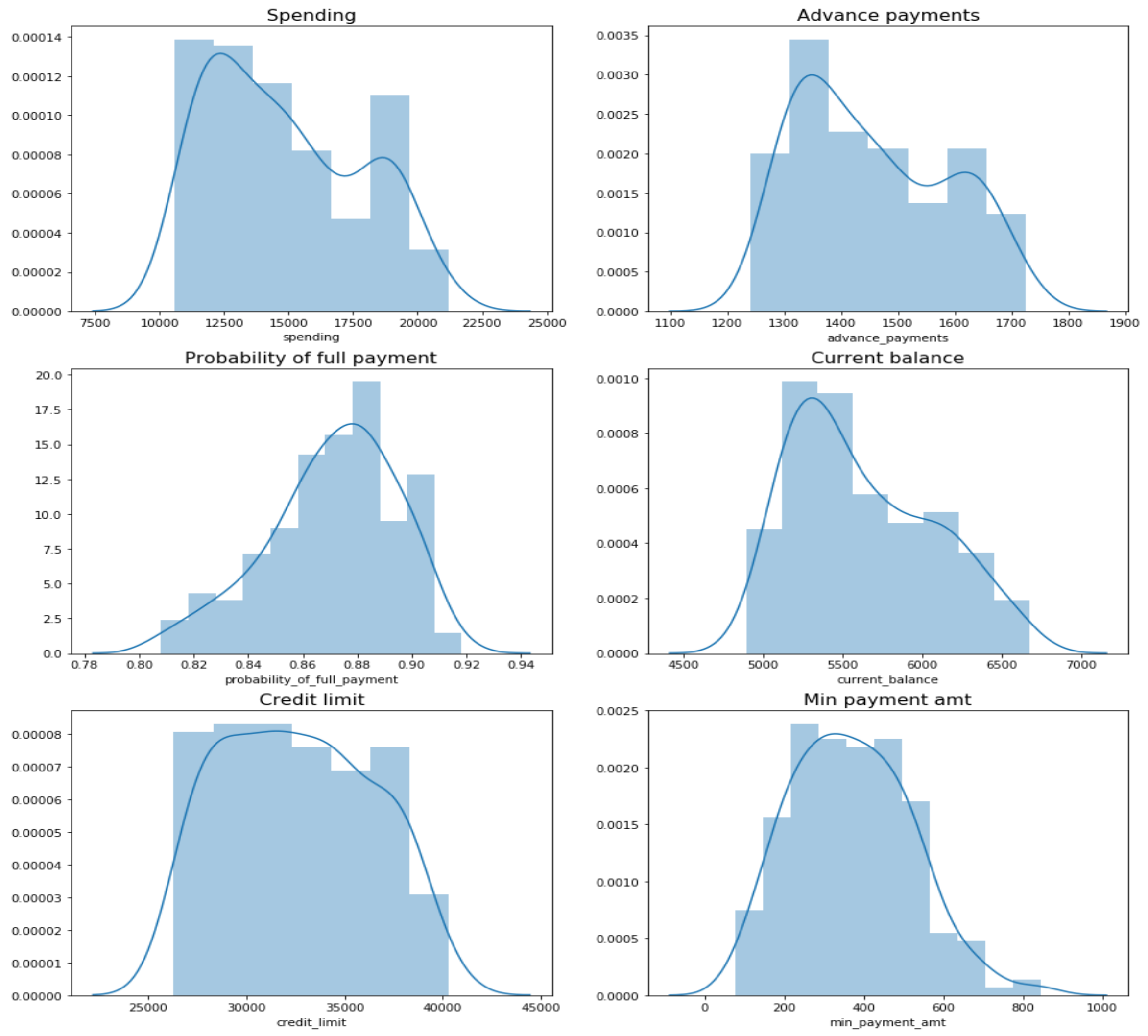
By checking the 5-point summary of the data, we get the following results:

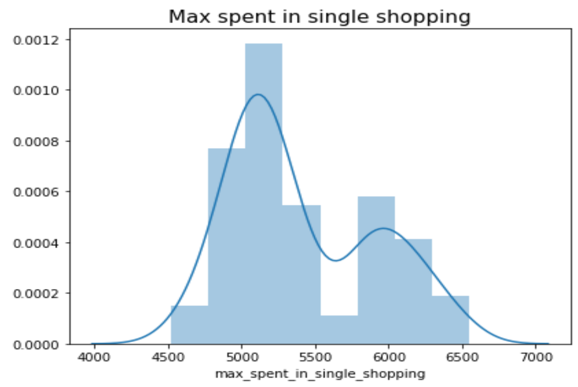


	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000
mean	14847.523810	1455.928571	0.870999	5628.533333	32586.047619	370.020095	5408.071429
std	2909.699431	130.595873	0.023629	443.063478	3777.144449	150.355713	491.480499
min	10590.000000	1241.000000	0.808100	4899.000000	26300.000000	76.510000	4519.000000
25%	12270.000000	1345.000000	0.856900	5262.250000	29440.000000	256.150000	5045.000000
50%	14355.000000	1432.000000	0.873450	5523.500000	32370.000000	359.900000	5223.000000
75%	17305.000000	1571.500000	0.887775	5979.750000	35617.500000	476.875000	5877.000000
max	21180.000000	1725.000000	0.918300	6675.000000	40330.000000	845.600000	6550.000000

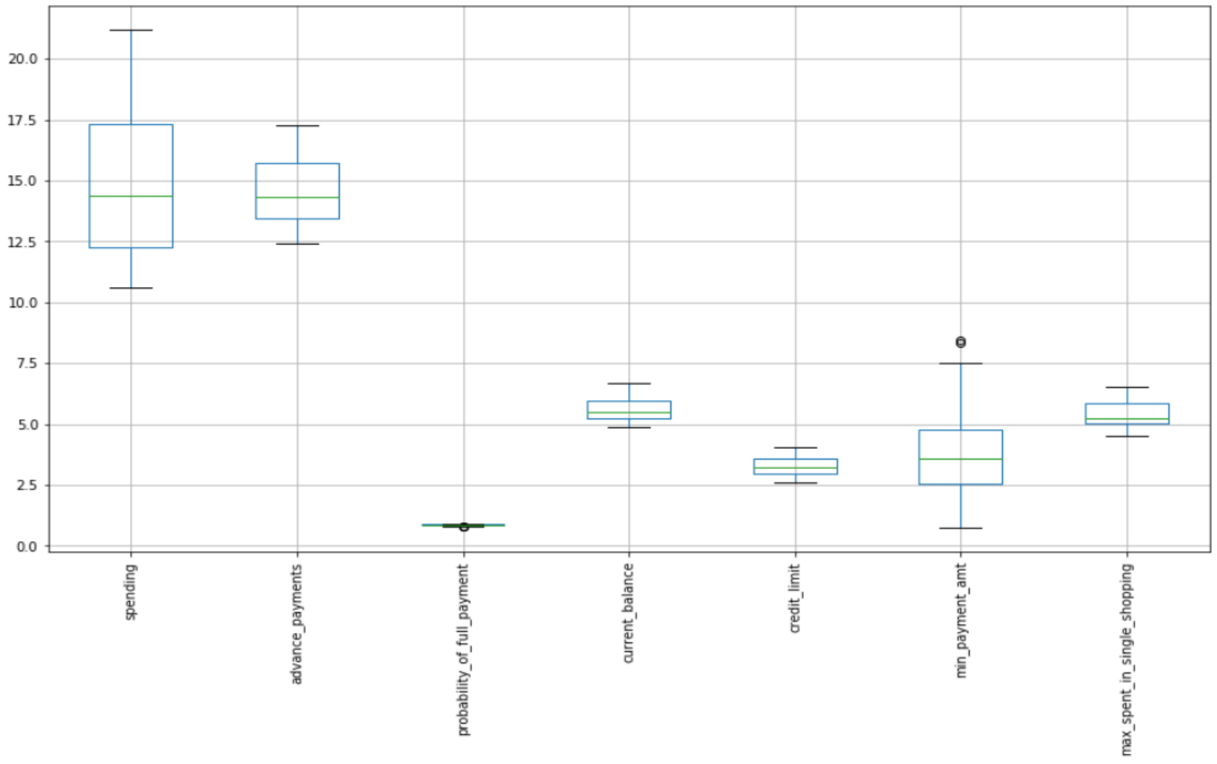
The probability of payment done in full by the customer to the bank lies between 0.8 to 0.9 approximately. This means there is a very high possibility that the customers will pay the bank in full.

We check the frequency distribution using histogram

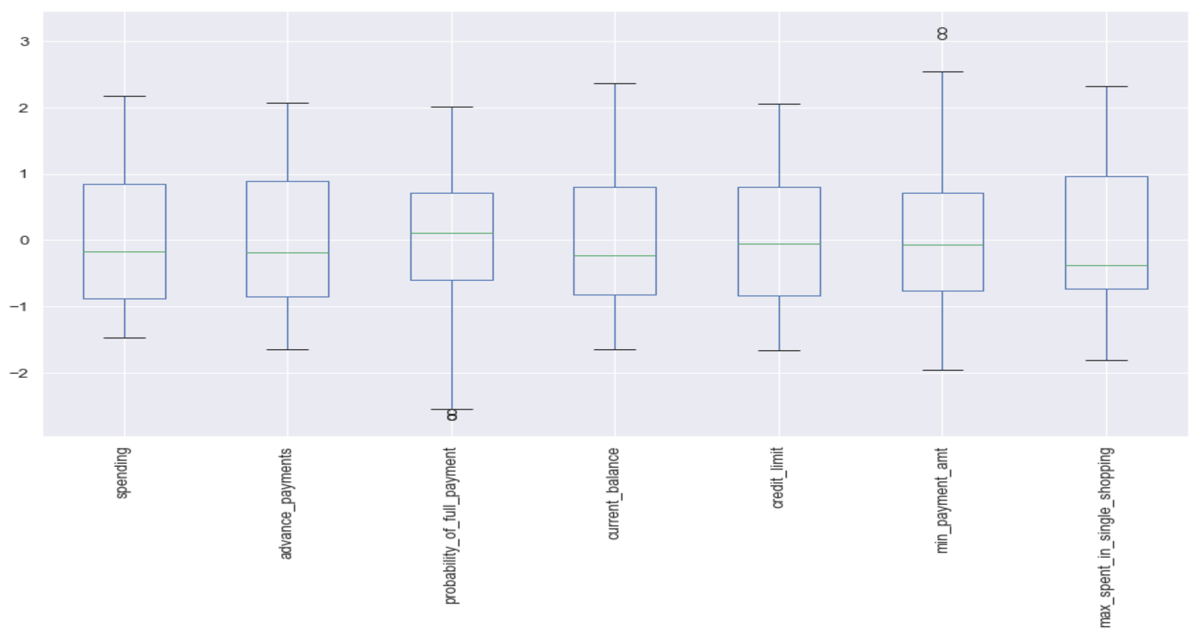




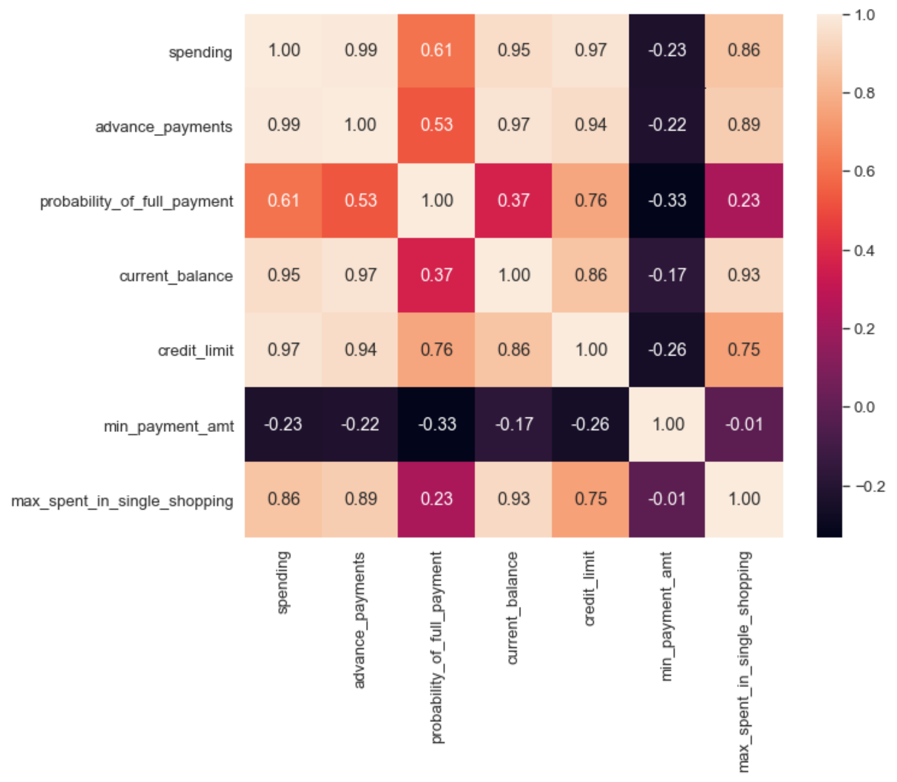
From the above plots the data seems to be normally distributed for each column, let’s check for outliers if any present.



There are practically minimal to no outliers present in the data. It appears to be pretty clean. But we can see that the columns are on different scales, and since clustering is a distance-based algorithm, it will require scaling in further steps. Scaled data looks as follows:

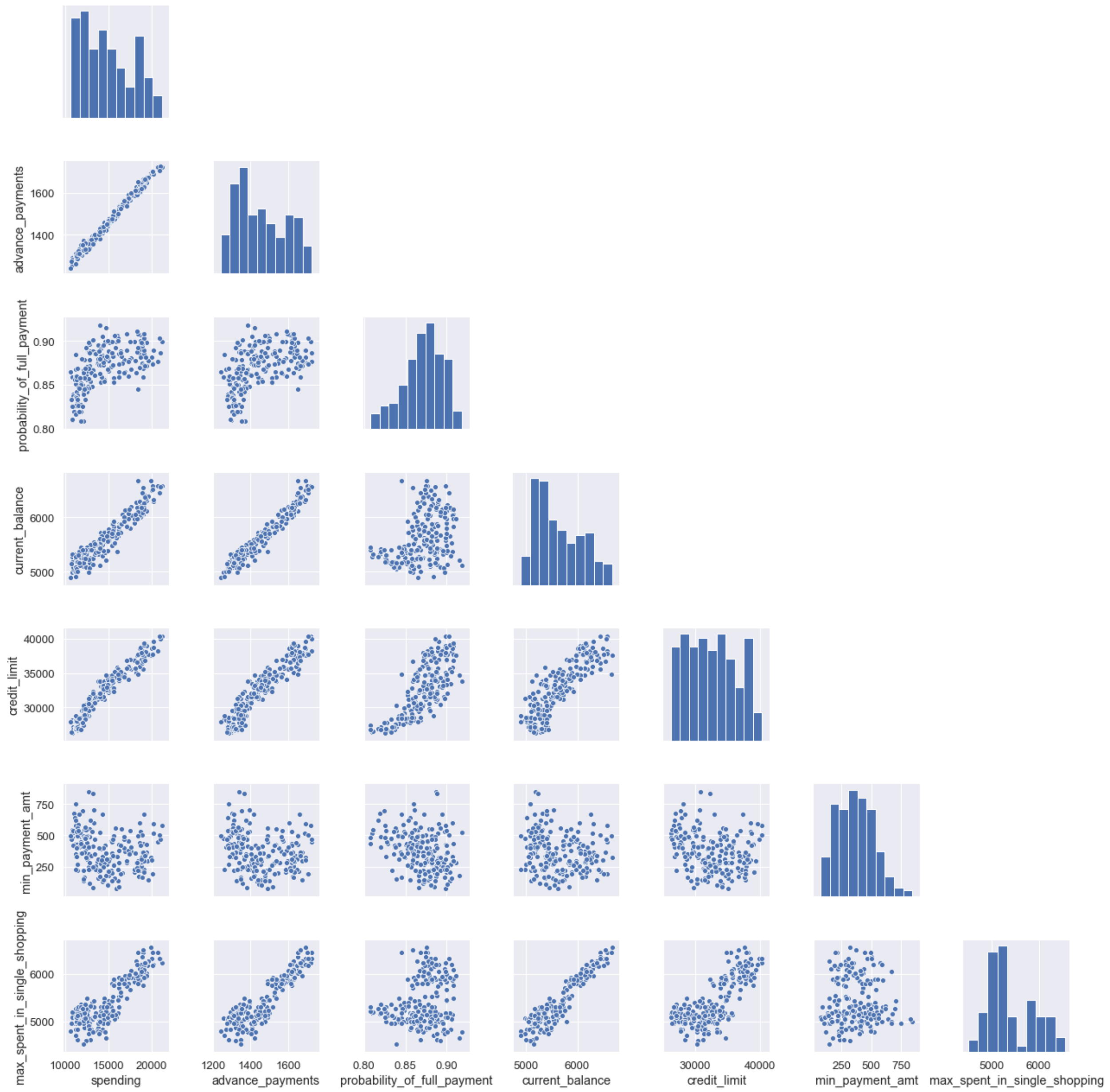


The correlation among the variables is shown in the following graph:



There seems to be a very high positive correlation between spending, advance_payments, current_balance, credit_limit and max_spent_in_single_shopping.

This can be verified using the pairplot as follows:



They appear to be positively linearly correlated. Infact apart from min_payment_amt, every other column appears to be positively correlated with others.

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

Ans)

By observing the boxplot from previous question, we can see that the variables are at different scales. Differences in the scales across input variables may increase the difficulty of the problem being modeled. Like in the given case study, large input values (i.e. a spread of hundreds or thousands of units) can result in a model that learns large weight values. A model with large weight values is often unstable, meaning that it may suffer from poor performance during learning and sensitivity to input values resulting in higher generalization error.

When input features are very different in scale / units, it is quite clear to see that classifiers / regressors which rely on Euclidean distance such as k-means will fail or be sub-optimal. Same goes for other regressors. Especially the ones that rely on gradient descent based optimization such as logistic regressions, Support Vector Machines and Neural networks. The only classifiers/regressors which are immune to impact of scale are the tree based regressors.

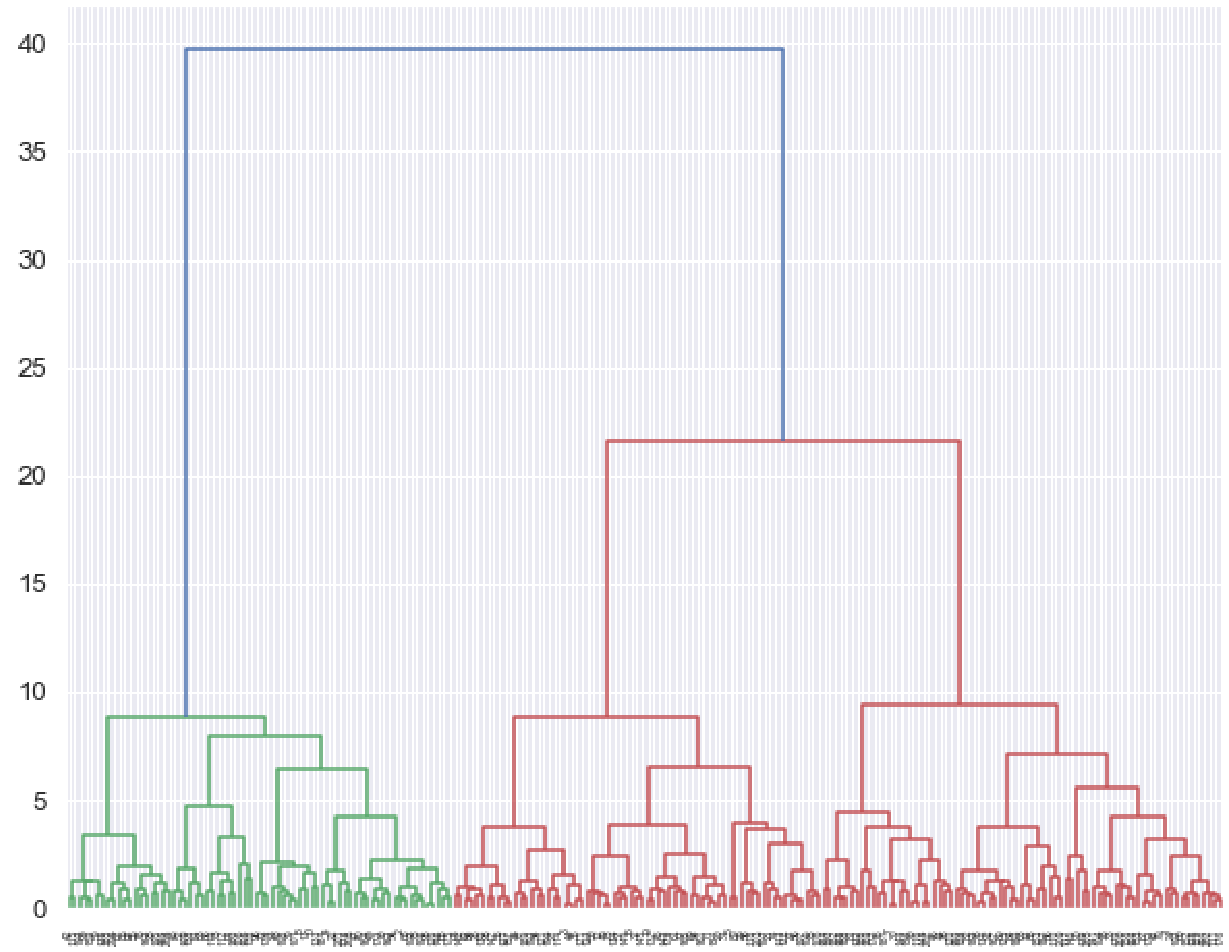
K-Means are distance-based algorithms. K-Means clusters the similar points together. The similarity here is defined by the distance between the points. Lesser the distance between the points, more is the similarity and vice versa. We do not want our algorithm to be affected by the magnitude of these variables. The algorithm should not be biased towards variables with higher magnitude. To overcome this problem, we can bring down all the variables to the same scale and thus scaling is necessary while clustering.

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

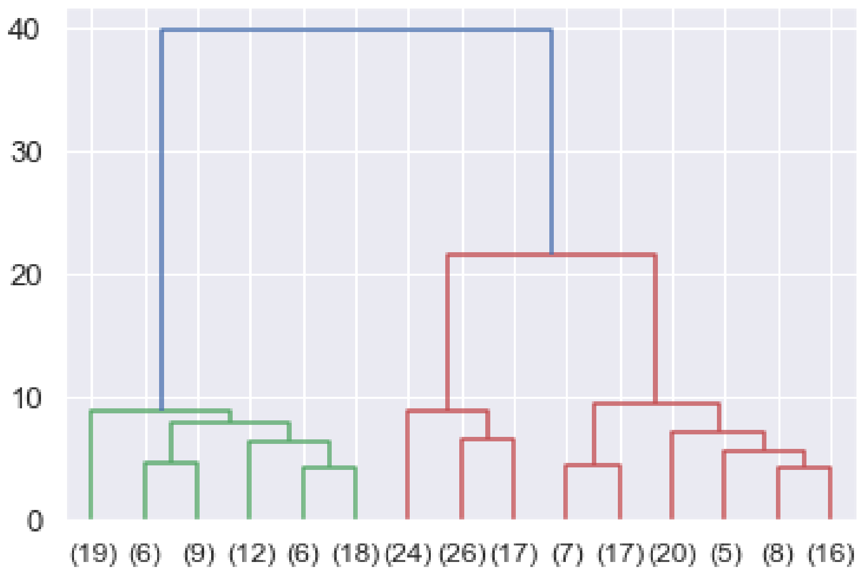
Ans)

For applying hierarchical clustering to scaled data, we will be using Ward's linkage method. **Ward's method** (a.k.a. *Minimum variance method* or *Ward's Minimum Variance Clustering Method*) is an alternative to single-link clustering because it usually creates compact, even-sized clusters. It starts with *n* clusters, each containing a single object. These *n* clusters are combined to make one cluster containing all objects. At each step, the process makes a new cluster that minimizes variance, measured by an index called the sum of squares index.

The Hierarchical clustering Technique can be visualized using a **Dendrogram.** A dendrogram is a visual representation of cluster-making. On the x-axis are the item names or item numbers. On the y-axis is the distance or height. The vertical straight lines denote the height where two items or two clusters combine. The higher the level of combining, the distant the individual items or clusters are.



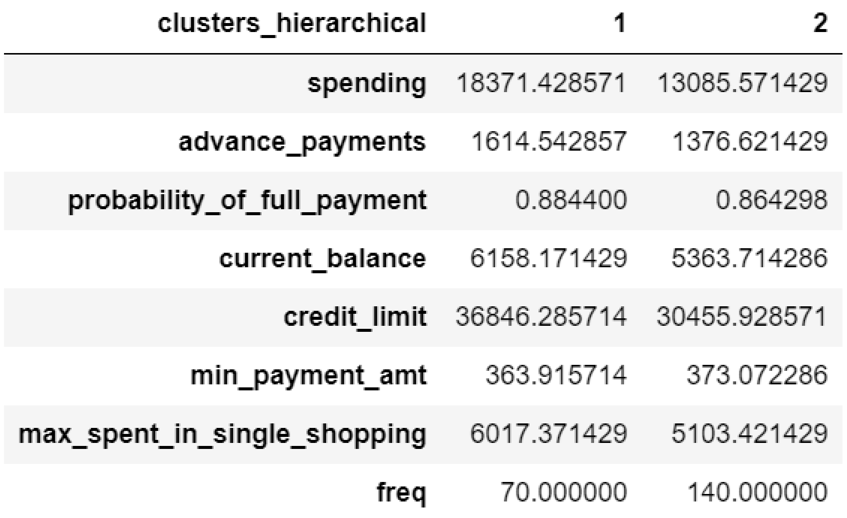
It appears to be quite complex, so we try to truncate the dendrogram to last 25 clusters.



In a dendrogram, each leaf corresponds to one observation. As we move up the tree, observations that are similar to each other are combined into branches, which are themselves fused at a higher height. The dendrogram can be cut where the difference is most significant.

The height of the fusion, provided on the vertical axis, indicates the (dis)similarity between two observations. The height of the cut to the dendrogram controls the number of clusters obtained. The higher the height of the fusion, the less similar the observations are. So by keeping these points in mind and observing the dendrogram, we can distinctly classify the data into majorly two clusters.

There are 70 customers in one cluster and 140 in the other. Their respective cluster profiles with mean used as estimator are as follows:



clusters_hierarchical	1	2
spending	18371.428571	13085.571429
advance_payments	1614.542857	1376.621429
probability_of_full_payment	0.884400	0.864298
current_balance	6158.171429	5363.714286
credit_limit	36846.285714	30455.928571
min_payment_amt	363.915714	373.072286
max_spent_in_single_shopping	6017.371429	5103.421429
freq	70.000000	140.000000

Observations:

- For cluster 1, with credit limit of around 37K, the mean spending is around 18K which is 50% of its limit whereas for cluster 2 the limit is around 30K and proportion of spending is also lesser as compared to cluster 1
- Cluster 1 has slightly higher probability of full payment than cluster 2
- There is not much significant difference in current_balance, min_payment_amt and max_spent_in_single_shopping amounts but the values are slighly higher for cluster 1.

1.4 Apply K-Means clustering on scaled data and determine optimum clusters. Apply elbow curve and silhouette score.

Ans)

K-means Clustering is a non-hierarchical approach to forming good clusters. It aims to partition n observations into *k* clusters in which each observation belongs to the cluster whose mean (centroid) is nearest to it, serving as a prototype of the cluster. It minimizes within-cluster variances (squared Euclidean distances). The number of clusters *k* needs to be pre-specified. Once that is determined, an arbitrary partition of data into *k* clusters is the starting point. Then sequential assignment and update will find the clusters that are most separated. The cardinal rule of cluster building is that, at every step within-cluster variance will reduce (or stay the same) but between-cluster variance will increase (or stay the same).

There are many methods that are recommended for determination of an optimal number of partitions. Unfortunately, however, there is no closed form solution to the problem of determining *k*. The choice is somewhat subjective and graphical methods are often employed.

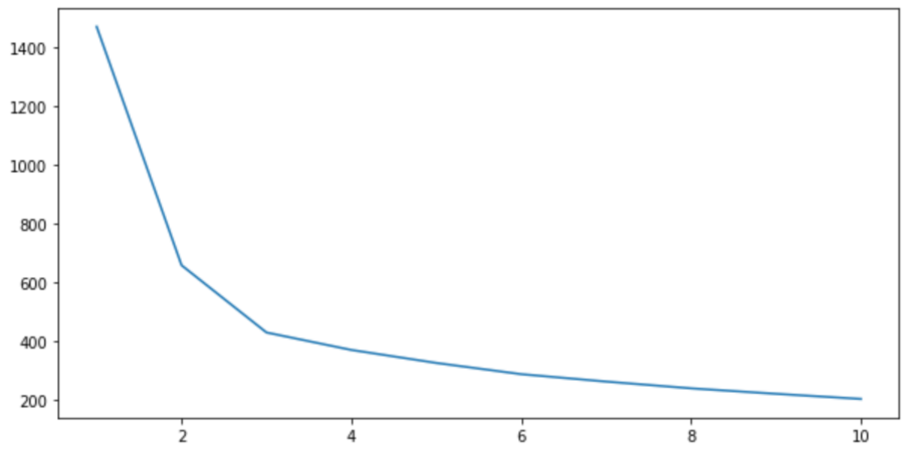
Objective of partitioning is to separate out the observations or units so that the 'most' similar items are put together. The singleton clusters will have the lowest value of WSS, but that is not useful. Hence finding *k* is striking a balance between WSS and cluster size.

Elbow method

The basic idea behind k-means clustering is to define clusters such that the total intra-cluster variation [or total within-cluster sum of square (WSS)] is minimized. The total WSS measures the compactness of the clustering and we want it to be as small as possible.

For a given number of clusters, the total within-cluster sum of squares (WCSS) is computed. That value of k is chosen to be optimum, where addition of one more cluster does not lower the value of total WCSS appreciably.

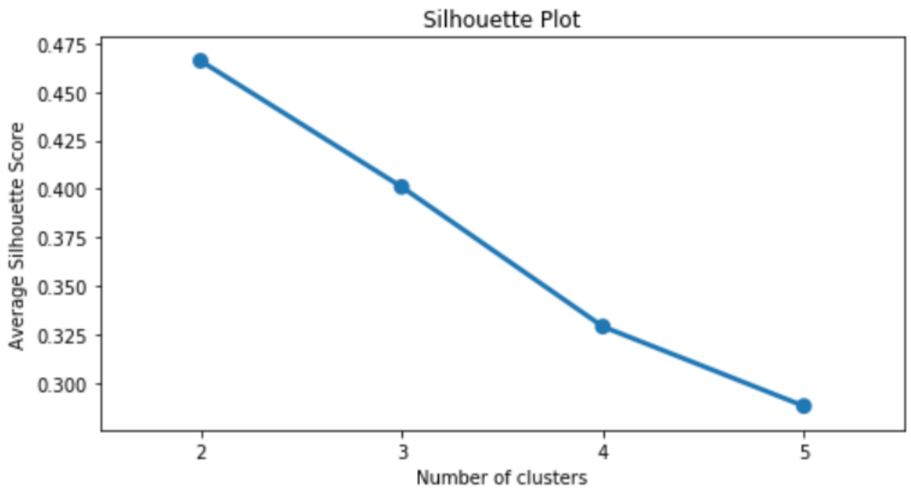
Following is the elbow plot for number of clusters in range from 1 to 10.



The Elbow method looks at the total WCSS as a function of the number of clusters. We select the value of k at the "elbow" i.e. the point after which the distortion/inertia start decreasing in a linear fashion. Above plot indicates a clear break in the elbow at k=2. Hence one option for optimum number of clusters is 2. Number of clusters equal to 3 can also be looked as a viable option, but the difference in WSS values from 2 to 3 is insignificant as compared from 1 to 2.

Silhouette Method

This method measures how tightly the observations are clustered and the average distance between clusters. For each observation a silhouette score is constructed which is a function of the average distance between the point and all other points in the cluster to which it belongs, and the distance between the point and all other points in all other clusters, that it does not belong to. The maximum value of the statistic indicates the optimum value of k.

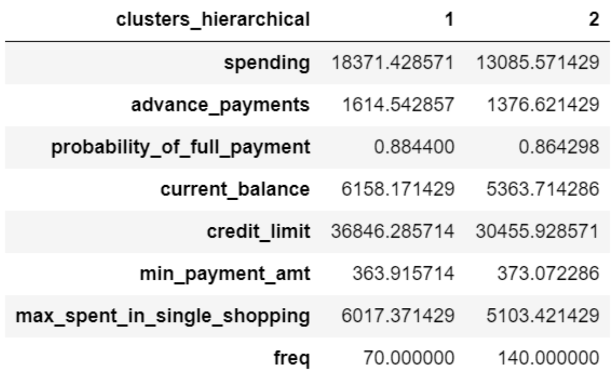


From the above plot, we see that the silhouette score is highest for 2 clusters, hence it can be considered as the optimal number of clusters in this case.

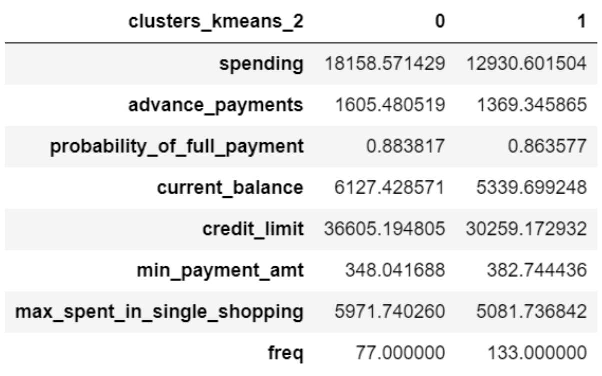
1.5 Describe cluster profiles for the clusters defined. Recommend different promotional strategies for different clusters.

Ans)

Below are the cluster profiles with both hierarchical and K-means clustering approaches with optimal number of clusters considered as 2.



clusters_hierarchical	1	2
spending	18371.428571	13085.571429
advance_payments	1614.542857	1376.621429
probability_of_full_payment	0.884400	0.864298
current_balance	6158.171429	5363.714286
credit_limit	36846.285714	30455.928571
min_payment_amt	363.915714	373.072286
max_spent_in_single_shopping	6017.371429	5103.421429
freq	70.000000	140.000000



clusters_kmeans_2	0	1
spending	18158.571429	12930.601504
advance_payments	1605.480519	1369.345865
probability_of_full_payment	0.883817	0.863577
current_balance	6127.428571	5339.699248
credit_limit	36605.194805	30259.172932
min_payment_amt	348.041688	382.744436
max_spent_in_single_shopping	5971.740260	5081.736842
freq	77.000000	133.000000

Hierarchical Clustering K-Means Clustering

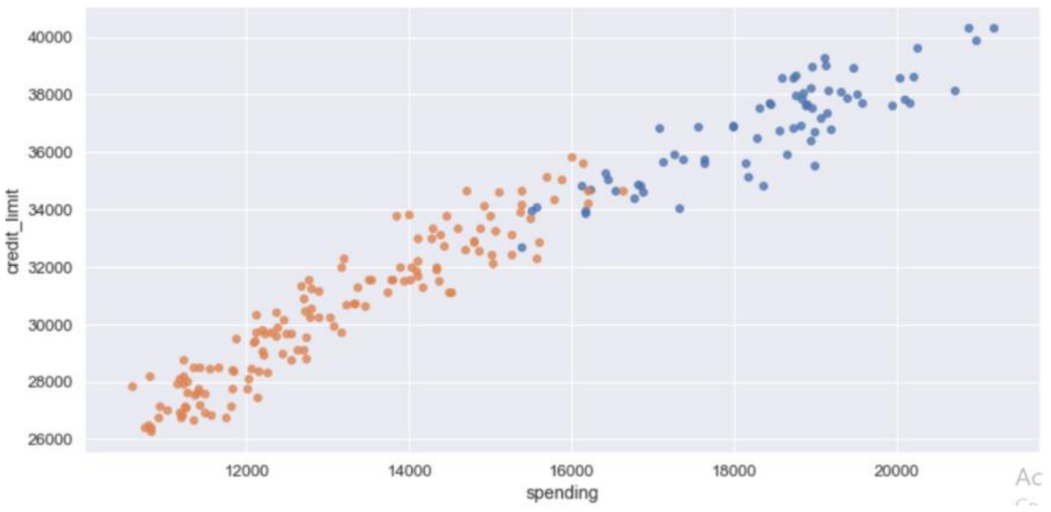
The values are pretty much the same; they have been clustered almost in the same manner. We will pick up the profile of hierarchical clustering for evaluation. To figure out the customer segments in each cluster, we first consider the mean profiles

- For cluster 1, mean credit limit is of around 37K, the mean spending is around 18K
- For cluster 2, mean credit limit is of around 30K, the mean spending is around 13K
- Cluster 1 has slightly higher probability of full payment than cluster 2
- There is not much significant difference in current_balance, min_payment_amt and max_spent_in_single_shopping amounts but the values are slighly higher for cluster 1.
- For cluster 1, with credit limit of around 37K, the mean spending is around 50% of its limit whereas for cluster 2 the limit is around 30K and proportion of spending is also lesser as compared to cluster 1
- There are 70 customers in cluster 1 and 140 customers in cluster 2

Detailed summary of clusters is described in the jupyter notebook attached at In[31] which helps us to derive useful insights for each cluster.

- Cluster 1 customers are higher spenders
- Cluster 2 customers have higher credit limits
- For both the current balances are in the range of approx. 5K to 7K
- The minimum - minimum payment amount for cluster 1 customers is higher
- Both minimum and maximum spent on single purchase amounts are higher for cluster 1 customers

From above observations, it is evident that the cluster 1 consists of higher earning bracket and cluster 2 consists of lower earning bracket.



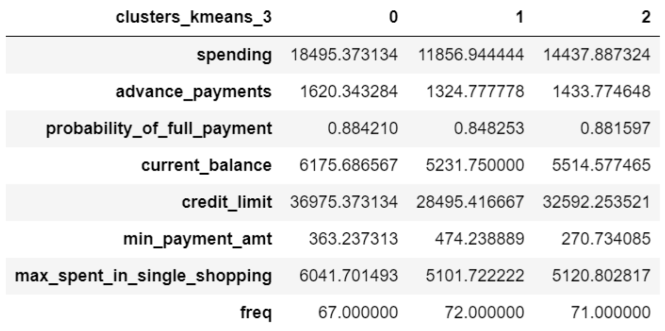
Looking at the values of every parameter it appears that the said bank has customers not very significantly different in their earning capabilities. They neither fall in the extremely rich bracket or extremely poor one. Most of the customers represent a mid-level earning group, where 1 group appears a bit more frugal in spending than the other. The high probability of paying the bank in full by both the groups suggest that they are aware and responsible customers who understand the importance of returning the amount in time and in full to the bank. This is also evident from the fact that they even pay a small amount in advance to the bank. Since their current balances are on the lower end, they might frequently use their credit cards for their day to day purchases.

Like any business, the bank’s ultimate goal is to make a profit. It can be increased by specific promotional strategies for each group. Charging fees is a well-known profit building technique, charging higher fees from the higher earning group can help in increase of profits.

It could be done in the form of marketing strategies too. Along with traditional advertising, the bank may offer special incentives to customers, such as a temporary low interest rate, frequent-flyer miles or free balance transfers. The cluster (1) with higher earning has lower number of people. As their spending is higher, they could be charged with higher interest rate than cluster 2, cluster 2 people will be motivated to shop more with lower interest rates which could increase their usage and as they are higher in population, the earnings through them would gradually increase. This can be carried out via "pre-approved" credit offers to entice them into filling out the card application.

These offers if targeted to the groups based on their maximum spent in single shopping would encourage them to buy more stuff in that range which would increase the profitability of the bank.

Since the customers are loyal, depending on their credit scores the credit limit could be increased. Banks should be attentive to the needs and performance of their customers in both the groups so that the loyal customers could have higher limits which increases their capability to spend.



clusters_kmeans_3	0	1	2
spending	18495.373134	11856.944444	14437.887324
advance_payments	1620.343284	1324.777778	1433.774648
probability_of_full_payment	0.884210	0.848253	0.881597
current_balance	6175.686567	5231.750000	5514.577465
credit_limit	36975.373134	28495.416667	32592.253521
min_payment_amt	363.237313	474.238889	270.734085
max_spent_in_single_shopping	6041.701493	5101.722222	5120.802817
freq	67.000000	72.000000	71.000000

For the particular case study, we could have considered 3 clusters too with their cluster profiles as above, but since as already mentioned the values are not significantly different from each other. So, to reduce the promotional cost and efforts and increase the profitability only two major customer segments were considered.

<u>Problem 2: CART-RF-ANN</u>

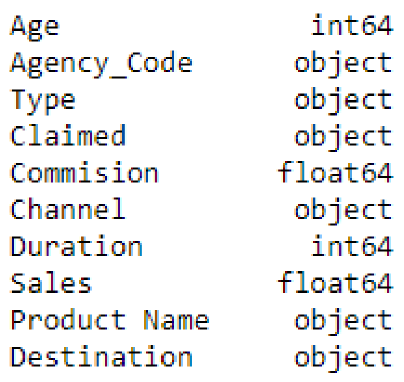
An Insurance firm providing tour insurance is facing higher claim frequency. The management decides to collect data from the past few years. You are assigned the task to make a model which predicts the claim status and provide recommendations to management. Use CART, RF & ANN and compare the models'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

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

Ans)

We are provided with the data of an Insurance firm over the past few years. The firm is facing higher claim frequency. Our task is to make a model using CART, RF & ANN to predict the claim status and provide recommendations to management.

The data contains information of about 3000 insured people with 10 attributes describing them. The attributes and their datatypes are as follows:



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

Attribute Information:

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) in days
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)

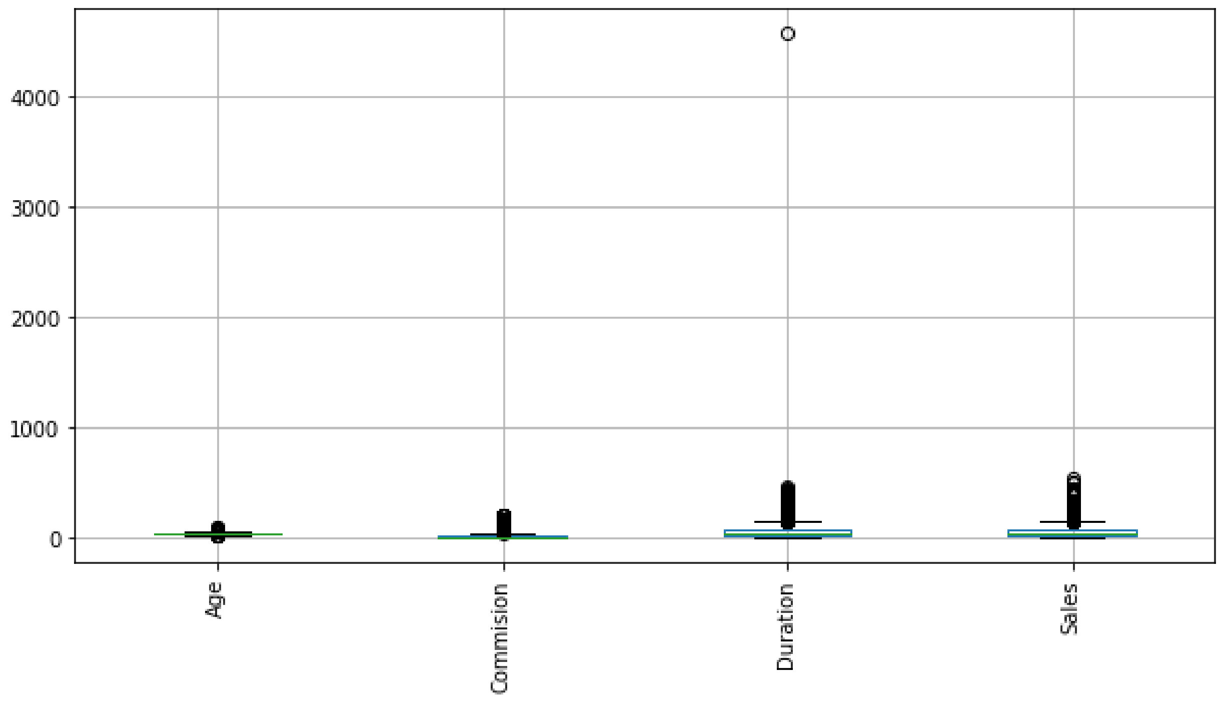
There are four numerical and 6 object datatypes. There are no null values in the dataset.

The five-point summary of data is as follows:

	Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
count	3000.000000	3000	3000	3000	3000.000000	3000	3000.000000	3000.000000	3000	3000
unique	NaN	4	2	2	NaN	2	NaN	NaN	5	3
top	NaN	EPX	Travel Agency	No	NaN	Online	NaN	NaN	Customised Plan	ASIA
freq	NaN	1365	1837	2076	NaN	2954	NaN	NaN	1136	2465
mean	38.091000	NaN	NaN	NaN	14.529203	NaN	70.001333	60.249913	NaN	NaN
std	10.463518	NaN	NaN	NaN	25.481455	NaN	134.053313	70.733954	NaN	NaN
min	8.000000	NaN	NaN	NaN	0.000000	NaN	-1.000000	0.000000	NaN	NaN
25%	32.000000	NaN	NaN	NaN	0.000000	NaN	11.000000	20.000000	NaN	NaN
50%	36.000000	NaN	NaN	NaN	4.630000	NaN	26.500000	33.000000	NaN	NaN
75%	42.000000	NaN	NaN	NaN	17.235000	NaN	63.000000	69.000000	NaN	NaN
max	84.000000	NaN	NaN	NaN	210.210000	NaN	4580.000000	539.000000	NaN	NaN

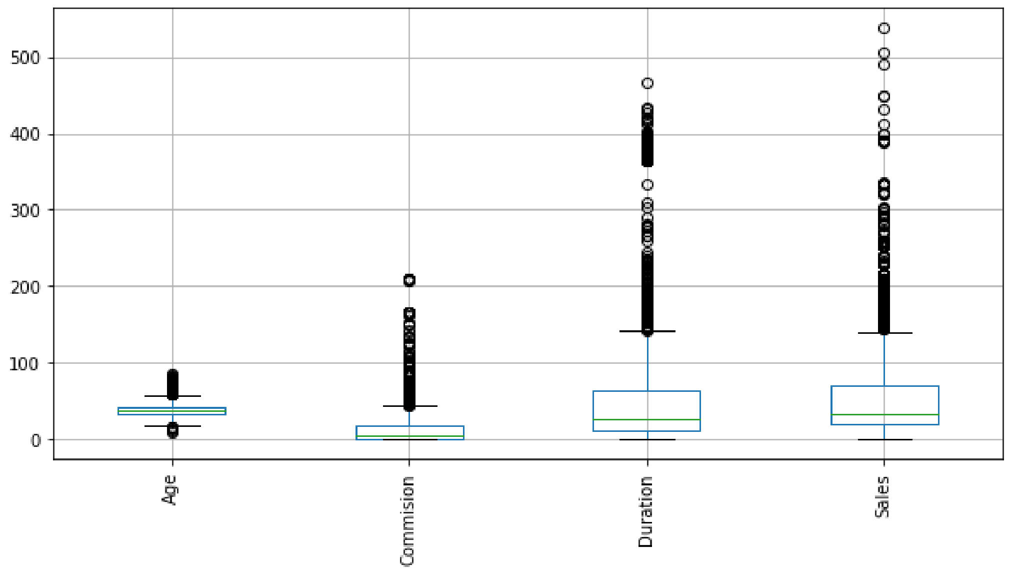
- We can see the unique counts for object datatype fields
- The numerical fields appear to be fine except the minimum and maximum values of duration column

Let's try to visualize distribution of continuous data using boxplot and pairplot.



From the box plot we can see that all observations appear closer to each other except one observation in the duration column. Also, the minimum value for duration present is -1 days, which could be an error since days cannot be negative. So, for the purpose of analysis we decide to drop the rows containing these two values as it would surely make a positive impact while building model.

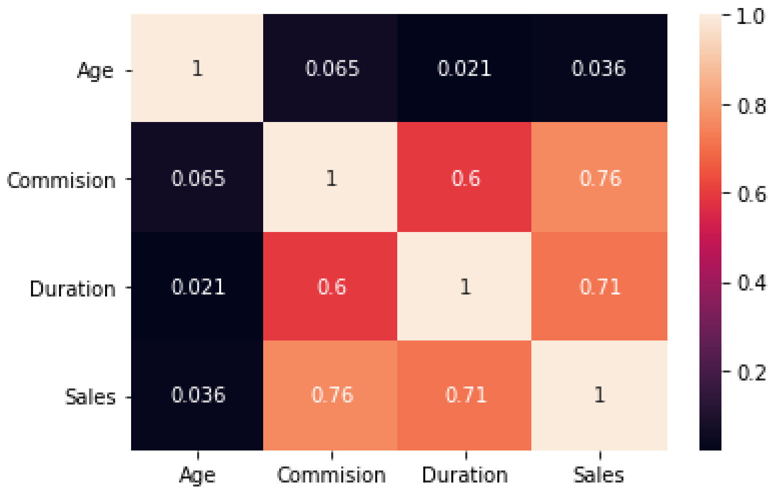
The boxplot after removing these two observations appears as follows:



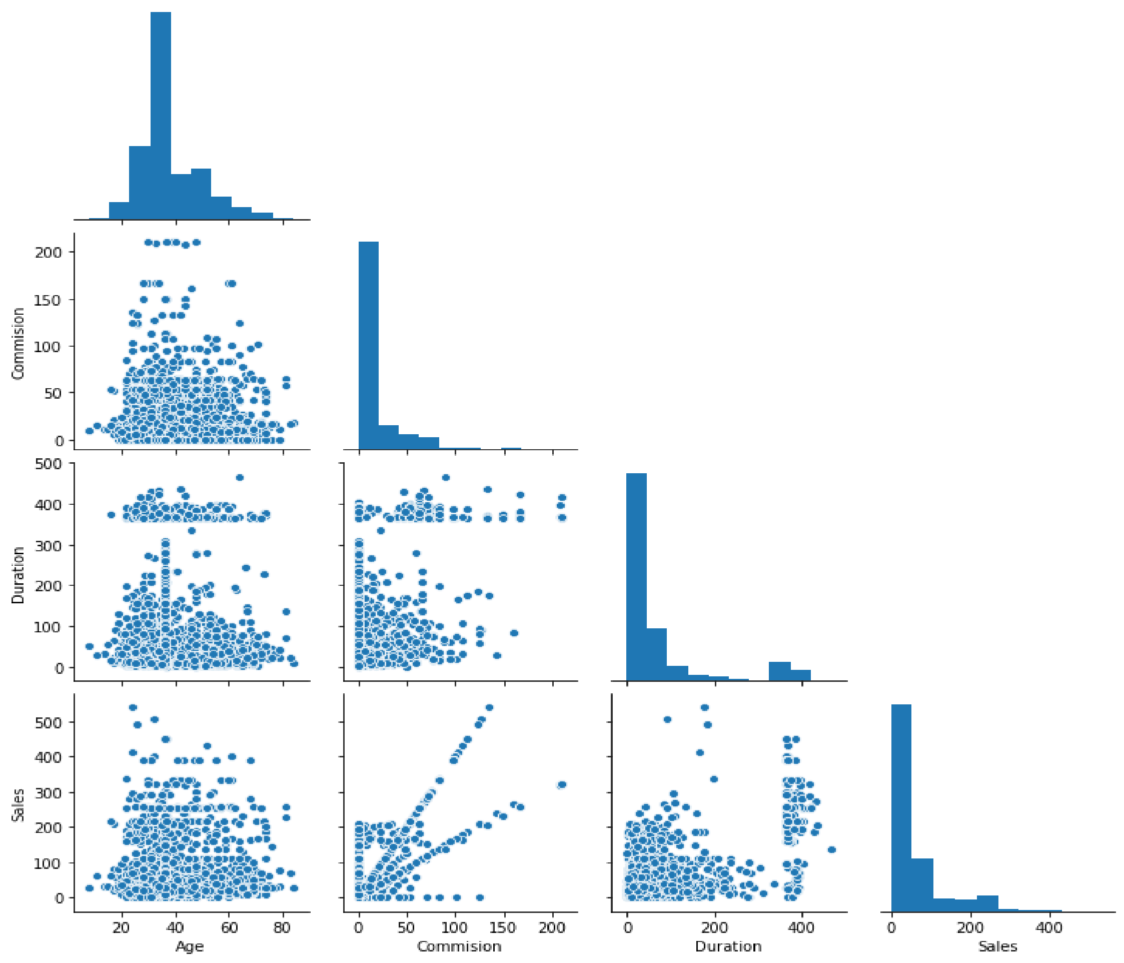
There are outliers in all the variables. Tree algorithms like CART and Random forest are robust to outliers. They split the data points on the basis of same value and so value of outlier won't affect that much to the split. Neural Networks can handle outliers if there are more hidden layers and if the number of outliers is lesser. For now, we will keep the data as it is, and if we find the performance of the neural network to be lesser, then we will treat the outliers and re-build neural network model.

There are 139 duplicate rows in the dataset which we will be removing as algorithm building cannot be performed with duplicate values and it doesn’t add any value. Thus we are left with 2859 observations which will be used further to build the models.

After making changes in data we can observe some positive correlation between sales and commission, duration and commission and duration.



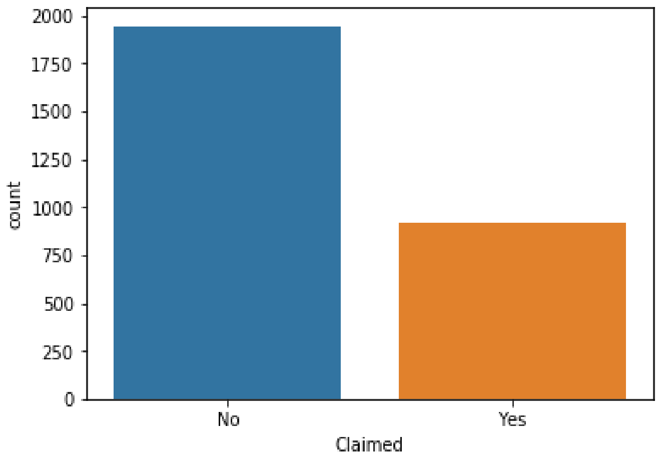
The pairplot for continuous variables is as follows:



From the frequency distribution we can see that apart from age column others appear to be highly skewed towards right, which indicates that the mean of them is greater than the median. And from the scatter plot there seems to be no significant correlation amongst the variables except commission and sales, which show a positive linear trend which is also evident from the correlation heatmap above.

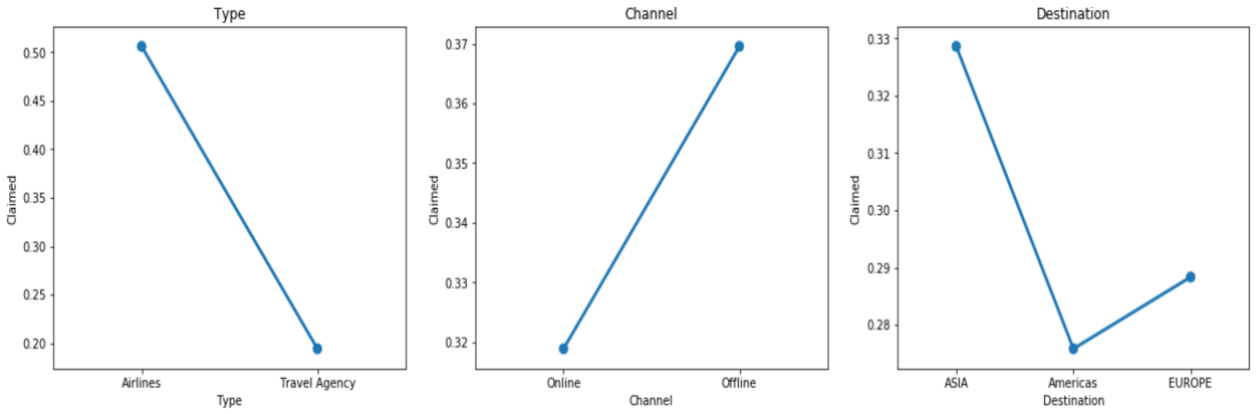
Exploratory Data Analysis

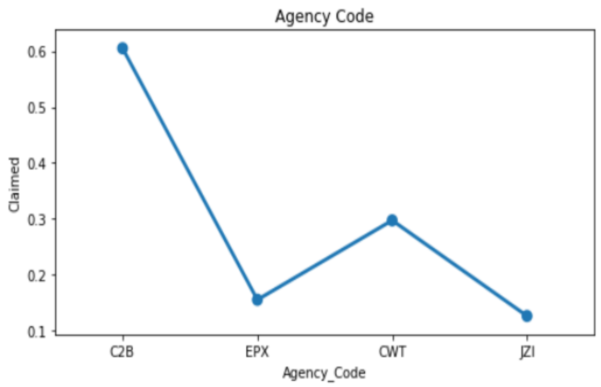
Let’s look at the number of claims in each category.

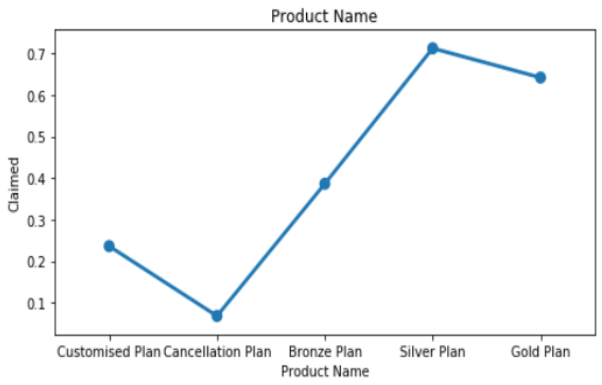


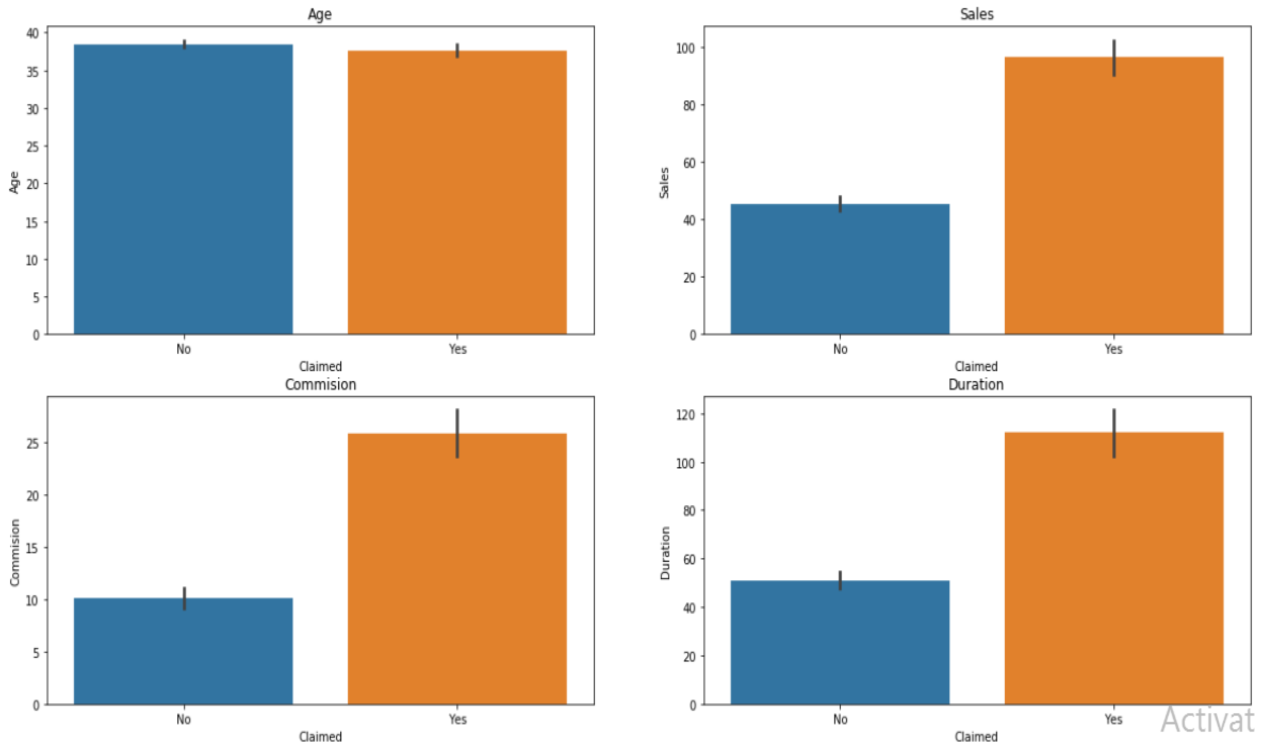
There are nearly 32% people who have claimed their policy and 68% people who have not claimed their policy. This seems to be a balanced dataset.

By observing the following point plots, we can check which categories have higher proportion of claims.



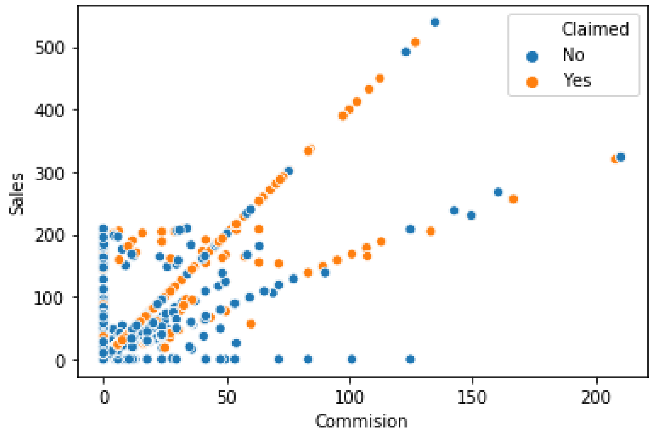






From the plots we can infer that

- Airlines have higher claim rates than travel agency
- Tour firm with agency code C2B has highest claim rates followed by CWT. EPX and JZI have very low claim rate
- Offline distribution channel of tour insurance agencies has higher claim rates
- Policies with destination as Asia have higher claims
- Silver plan and gold plan have significantly higher claim rates as compared to other products
- Age doesn't seem to be a defining factor when it comes to claims
- The claims are higher if the duration, sales and commission values are higher. It can be visualized from the following graph too.



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

Ans)

For building the above models in Python, we need to convert object type variables into numerical as python can only work with numerical / categorical columns. It cannot take string / object types for building models. The data has been split in train and test set in the ratio 7:3; 70% being the training data and 30% as testing data. We have also assigned a random state value for each model in order to have a synchronized output at all systems.

CART Model

Decision Trees are an important type of algorithm for predictive modeling machine learning. It is also known by its more modern name CART which stands for Classification and Regression Trees. The representation for the CART model is a binary tree. Each root node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric). The leaf nodes of the tree contain an output variable (y) which is used to make a prediction. Given a new input, the tree is traversed by evaluating the specific input started at the root node of the tree.

First, we create a basic model without specifying any other parameters except random state and try to examine its accuracy and classification report. We use the DecisionTreeClassifier() method from sklearn library to build the decision tree model.

Train data	Test data
Accuracy	
0.99	0.71
Classification report	

Train data:

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1353
           1       0.99      0.99      0.99       648

    accuracy                           0.99      2001
   macro avg       0.99      0.99      0.99      2001
weighted avg       0.99      0.99      0.99      2001
```

Test data:

```
              precision    recall  f1-score   support

           0       0.80      0.78      0.79       592
           1       0.53      0.56      0.55       266

    accuracy                           0.71       858
   macro avg       0.66      0.67      0.67       858
weighted avg       0.72      0.71      0.71       858
```

The accuracy, recall and precision values are 0.99 which suggest that model has been over-fitted which is not desirable. The difference in performances of the train and test set is also significant which is not desirable.

Creating a binary decision tree is actually a process of dividing up the input space. A greedy approach is used to divide the space called recursive binary splitting. The recursive binary splitting procedure needs to know when to stop splitting as it works its way down the tree with the training data. Hence pruning technique is applied to further lift performance of the model and get over the problem of over-fitting.

We prune and fine tune the model using the following parameters:

- **criterion**: This parameter determines how the impurity of a split will be measured. The default value is “gini”, but you can also use “entropy” as a metric for impurity.
- **max_depth:** This determines the maximum depth of the tree. The default value is set to none. This will often result in over-fitted decision trees. The depth parameter is one of the ways in which we can regularize the tree or limit the way it grows to prevent over-fitting.
- **min_samples_split:** The minimum number of samples a node must contain in order to consider splitting. The default value is two. This parameter is used to regularize the tree.
- **min_samples_leaf:** The minimum number of samples needed to be considered a leaf node. The default value is set to one. This parameter is used to limit the growth of the tree.

After fine tuning the model with the help of grid search, we got the best parameter values for max_depth, min_samples_leaf and min_samples_split as 4, 20 and 45 respectively. **We will look at the performances of each model in the next question.** The values used for tuning can be referred from the jupyter notebook attached.

Random Forest Model

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

It builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node.

First, we create a basic model without specifying any other parameters except random state and try to examine its accuracy and classification report. We use RandomForestClassifier() method from sklearn library to build the random forest model.

Train data	Test data
Accuracy	
0.99	0.75
Classification report	

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1353
           1       0.99      0.99      0.99       648

    accuracy                           0.99      2001
   macro avg       0.99      0.99      0.99      2001
weighted avg       0.99      0.99      0.99      2001
```

```
              precision    recall  f1-score   support

           0       0.82      0.82      0.82       592
           1       0.60      0.59      0.59       266

    accuracy                           0.75       858
   macro avg       0.71      0.71      0.71       858
weighted avg       0.75      0.75      0.75       858
```

The accuracy, recall and precision values are 0.99 which suggest that model has been over-fitted which is not desirable. The difference in performances of the train and test set is also significant which is not desirable. We can observe that random forest has performed slightly better than the basic CART model owing to the fact that is an ensemble-based model.

We will try to deal with the problem of over-fitting by adding certain parameters. They are used either to increase the predictive power of the model or to make it easier to train the model.

- **max_features**: These are the maximum number of features Random Forest is allowed to try in individual tree. Increasing max_features generally improves the performance of the model as at each node now we have a higher number of options to be considered. However, this is not necessarily true as this decreases the diversity of individual tree which is the USP of random forest. Hence, we need to strike the right balance and choose the optimal max_features.
- **n_estimators:** This is the number of trees you want to build before taking the maximum voting or averages of predictions. Higher number of trees gives better performance but makes the code slower. So, we must choose as high value as our processor can handle because this makes the predictions stronger and more stable.
- **max_depth:** This determines the maximum depth of the tree. The default value is set to none. This will often result in over-fitted decision trees. The depth parameter is one of the ways in which we can regularize the tree or limit the way it grows to prevent over-fitting.
- **min_samples_split:** The minimum number of samples a node must contain in order to consider splitting. The default value is two. This parameter is used to regularize the tree.
- **min_samples_leaf:** The minimum number of samples needed to be considered a leaf node. The default value is set to one. This parameter is used to limit the growth of the tree.

After fine tuning the model with the help of grid search, we got the best parameter values for max_depth, max_features, min_samples_leaf, min_samples_split and n_estimators as 10, 4, 5, 15 and 501 respectively. The values used for tuning can be referred from the jupyter notebook attached.

Artificial Neural Network Model

Artificial Neural Networks (ANN) is a supervised learning system built of a large number of simple elements, called neurons or perceptrons. Each neuron can make simple decisions, and feeds those decisions to other neurons, organized in interconnected layers. Each connection link is associated with a weight that has information about the input signal. This is the most useful information for neurons to solve a particular problem because the weight usually excites or inhibits the signal that is being communicated. Each neuron has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units. Together, the neural network can emulate almost any function, and answer practically any question, given enough training samples and computing power. A "shallow" neural network has only three layers of neurons:

- An input layer that accepts the independent variables or inputs of the model
- One hidden layer
- An output layer that generates predictions

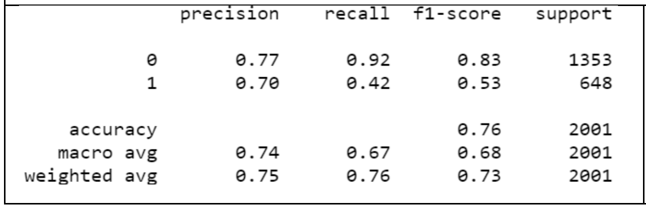
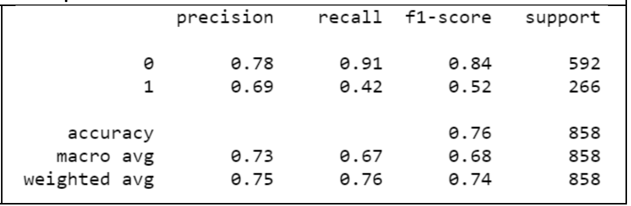
Neural network models learn mapping from input variables to an output variable. The scale and distribution of the data drawn from the domain may be different for each variable. Differences in the scales across input variables may increase the difficulty of the problem being modeled. Large input values can result in a model that learns large weight values. A model with large weight values is often unstable, meaning that it may suffer from poor performance during learning and sensitivity to input values resulting in higher generalization error.

So before building the model, we scale the training and test data. The following methods are used to center/feature scale of a given data. It basically helps to normalize the data within a particular range.

- fit (): Method calculates the parameters μ and σ and saves them as internal objects.
- transform (): Method using these calculated parameters apply the transformation to a particular dataset.
- Fit_transform (): joins the fit () and transform() method for transformation of dataset.

We apply the fit_transform() method to the train data, which calculates μ and σ of the training set and uses it to transform the test set, thus we just apply the transform() method on the test set. We use MLPClassifier() method from sklearn library to build the artificial neural network model.

First, we create a basic model without specifying any other parameters except random state and try to examine its accuracy and classification report.

Train data	Test data
Accuracy	
0.76	0.76
Classification report	
	

```
              precision    recall  f1-score   support

           0       0.77      0.92      0.83      1353
           1       0.70      0.42      0.53       648

    accuracy                           0.76      2001
   macro avg       0.74      0.67      0.68      2001
weighted avg       0.75      0.76      0.73      2001
```

```
              precision    recall  f1-score   support

           0       0.78      0.91      0.84       592
           1       0.69      0.42      0.52       266

    accuracy                           0.76       858
   macro avg       0.73      0.67      0.68       858
weighted avg       0.75      0.76      0.74       858
```

When compared to CART and random forest, we can infer from the accuracy values of train set that over-fitting has not occurred in this case. And the model has converged well as it has performed equally well as it did on its train set. But the recall values are low, we can try to fine tune the model by adding model parameters to increase its performance.

We use the following parameters to optimize the model and find the optimal parameter values to get the best model performance.

- **hidden_layer_sizes**: This parameter allows us to set the number of layers and the number of nodes we wish to have in the Neural Network Classifier. Each element in the tuple represents the number of nodes at the ith position where i is the index of the tuple. Thus, the length of tuple denotes the total number of hidden layers in the network
- **max_iter**: It denotes the number of epochs
- **activation**: The activation function for the hidden layers
- **solver**: This parameter specifies the algorithm for weight optimization across the nodes
- **tol:** It is the tolerance for the stopping criteria. This tells scikit to stop searching for a minimum (or maximum) once some tolerance is achieved, i.e. once you're close

After fine tuning the model with the help of grid search, we got the best parameter values for hidden_layer_sizes, max_iter, activation, solver and tol as 80, 500, relu, adam and 0.01 respectively. The values used for tuning can be referred from the jupyter notebook attached.

2.3 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

Ans)

After performing grid search on various parameter values, the optimized models were built. The optimized models have certainly performed better than their basic version when compared as mentioned in previous question. Their performance is evaluated based on the following evaluation metrics:

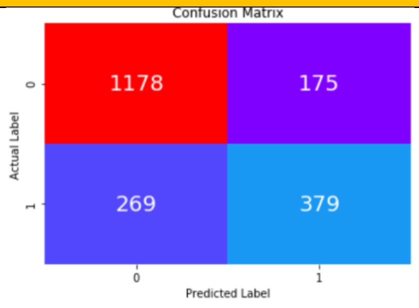
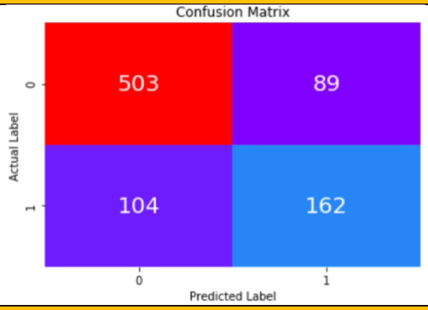
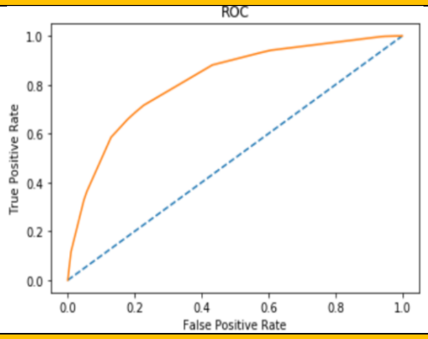
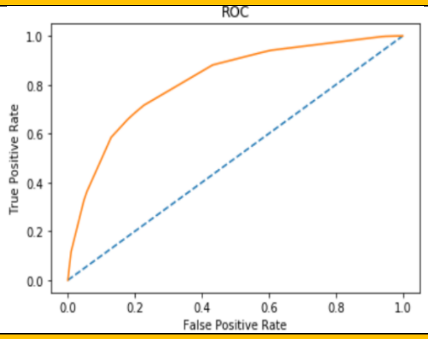
- **Accuracy**: Model accuracy in terms of classification models can be defined as the ratio of correctly classified samples to the total number of samples.
- **Precision**: The precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). High precision means that an algorithm returned substantially more relevant results than irrelevant ones.
- **Recall**: Recall is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been). High recall means that an algorithm returned most of the relevant results.
- **F1 score**: The F1 score is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.
- **ROC Curve**: A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true positive rate is also known as sensitivity, recall, or probability of detection in machine learning. The false-positive rate is also known as the fall-out, or probability of false alarm. The best possible prediction model would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0,1) point is also called a Perfect Classification.

- **AUC**: It is the area under the ROC curve. It measures how well predictions are ranked, rather than their absolute values. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.
- **Confusion matrix**: A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other. Most performance measures are computed from the confusion matrix.

Let's compare the metrics of each model for train and test data.

CART Model

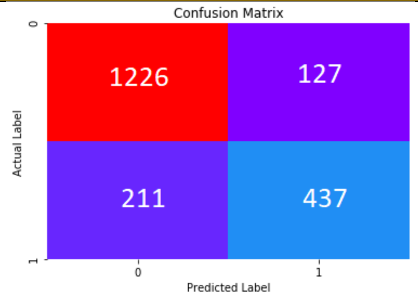
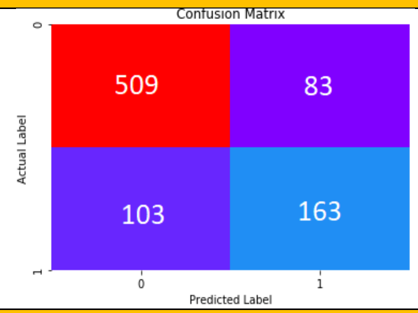
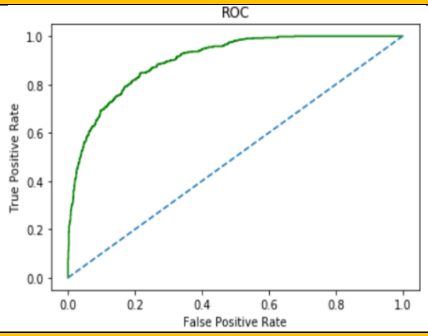
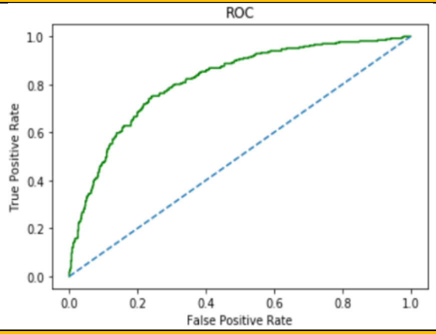
Training Data	Testing Data
Accuracy	
0.78	**0.78**
ROC_AUC score	
0.73	**0.73**
Confusion Matrix	
	
ROC curve	
	
Classification Report	

```
              precision    recall  f1-score   support

           0       0.81      0.87      0.84      1353
           1       0.68      0.58      0.63       648

    accuracy                           0.78      2001
   macro avg       0.75      0.73      0.74      2001
weighted avg       0.77      0.78      0.77      2001
```

```
              precision    recall  f1-score   support

           0       0.83      0.85      0.84       592
           1       0.65      0.61      0.63       266

    accuracy                           0.78       858
   macro avg       0.74      0.73      0.73       858
weighted avg       0.77      0.78      0.77       858
```

The optimized decision tree has performed very well on the train as well as test data. The accuracy and AUC values of train data are in line with the test data. So are the values of precision, recall and F1 – score.

Random Forest Model

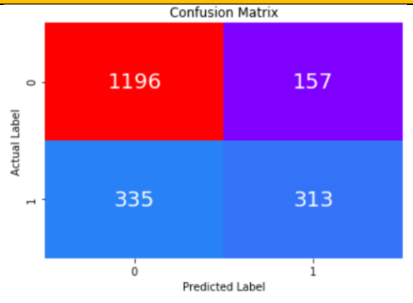
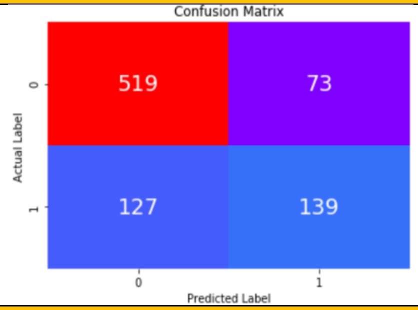
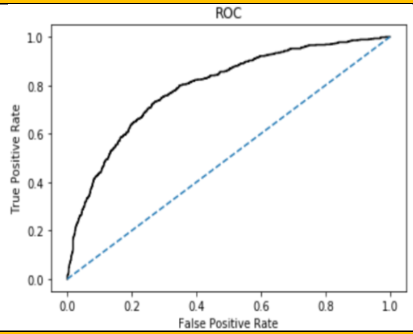
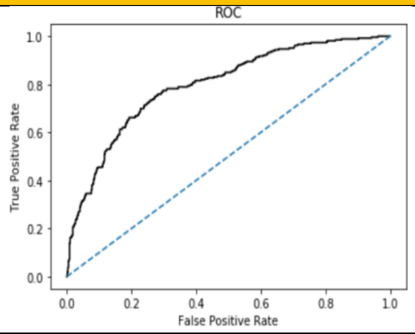
Training Data	**Testing Data**
Accuracy	
0.83	**0.77**
ROC_AUC score	
0.79	**0.74**
Confusion Matrix	
	
ROC curve	
	
Classification Report	

```
              precision    recall  f1-score   support

           0       0.85      0.91      0.88      1353
           1       0.77      0.67      0.72       648

    accuracy                           0.83      2001
   macro avg       0.81      0.79      0.80      2001
weighted avg       0.83      0.83      0.83      2001
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.85       592
           1       0.66      0.61      0.64       266

    accuracy                           0.78       858
   macro avg       0.75      0.74      0.74       858
weighted avg       0.78      0.78      0.78       858
```

The optimized RF model has performed very well on the train data. Test data has a bit lower performance than train data but overall has performed well. The accuracy and AUC values are pretty good. The recall and precision is almost similar to CART model, but F1-score is slightly higher for RF model. It has the highest number of true positives and lowest number of false negatives which is desirable in this problem statement, is discussed further in upcoming questions.

Artificial Neural Network Model

Training Data	Testing Data
Accuracy	
0.75	**0.77**
ROC_AUC score	
0.68	**0.70**
Confusion Matrix	
	
ROC curve	
	
Classification Report	

Training Data:

```
              precision    recall  f1-score   support

           0       0.78      0.88      0.83      1353
           1       0.67      0.48      0.56       648

    accuracy                           0.75      2001
   macro avg       0.72      0.68      0.69      2001
weighted avg       0.74      0.75      0.74      2001
```

Testing Data:

```
              precision    recall  f1-score   support

           0       0.80      0.88      0.84       592
           1       0.66      0.52      0.58       266

    accuracy                           0.77       858
   macro avg       0.73      0.70      0.71       858
weighted avg       0.76      0.77      0.76       858
```

Although the model had performed well in the basic model as far as over-fitting problem is concerned, it could not perform as well even after optimization as compared to CART and RF models. The accuracy and AUC score are good enough, but the recall value is not good enough to be used in the given problem statement, it is discussed further in the next question. For model computation part, please refer the Jupyter notebook attached.

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

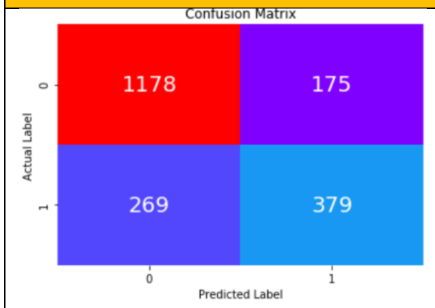
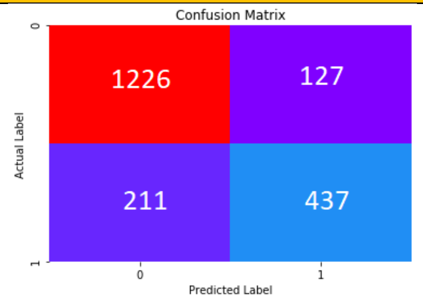
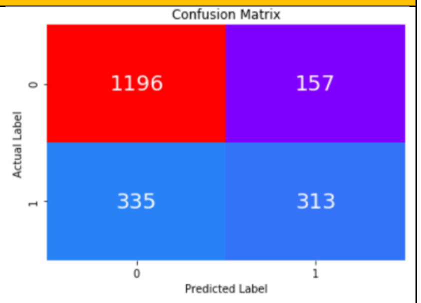
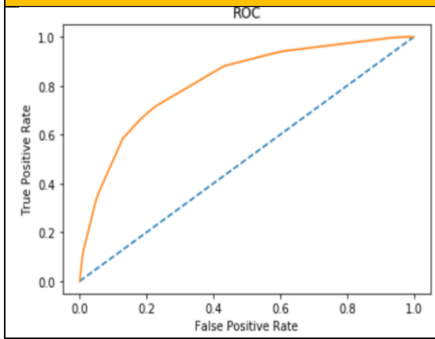
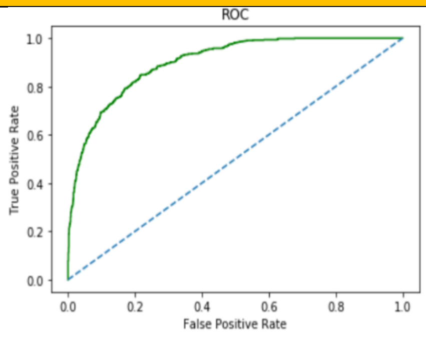
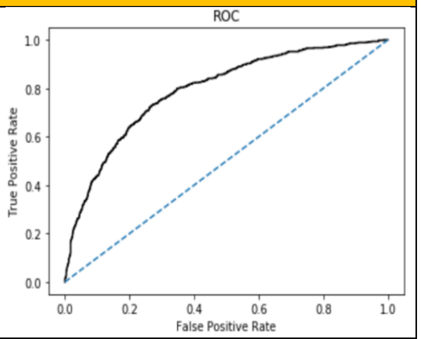
Ans)

From the previous question, we looked at the performances of base models, and by comparing their performance metrics we see that random forest has performed the best on the test set, followed by CART whereas ANN has performed the best at converging and overcoming the problem of over-fitting in train and test set. But since these models were over-fitted, they were optimized with certain parameters for higher performance and reliability.

We will now be comparing the optimized versions of all the models to get a better picture which model works best in this case study.

Training Data		
Decision Tree	**Random Forest**	**Artificial Neural Network**
Accuracy		
0.78	**0.83**	**0.75**
ROC_AUC score		
0.73	**0.79**	**0.68**
Classification Report		

`precision recall f1-score support`
`0 0.81 0.87 0.84 1353`
`1 0.68 0.58 0.63 648`
`accuracy 0.78 2001`
`macro avg 0.75 0.73 0.74 2001`
`weighted avg 0.77 0.78 0.77 2001`
	`precision recall f1-score support`
	`0 0.85 0.91 0.88 1353`
	`1 0.77 0.67 0.72 648`
	`accuracy 0.83 2001`
	`macro avg 0.81 0.79 0.80 2001`
	`weighted avg 0.83 0.83 0.83 2001`
		`precision recall f1-score support`
		`0 0.78 0.88 0.83 1353`
		`1 0.67 0.48 0.56 648`
		`accuracy 0.75 2001`
		`macro avg 0.72 0.68 0.69 2001`
		`weighted avg 0.74 0.75 0.74 2001`

Confusion Matrix		
		
ROC curve		
		

Testing Data		
Decision Tree	**Random Forest**	**Artificial Neural Network**
Accuracy		
0.78	0.77	0.77
ROC_AUC score		
0.73	0.74	0.70

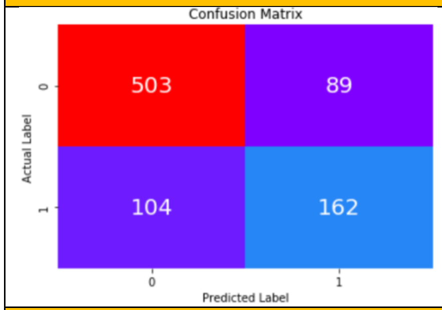
Classification Report

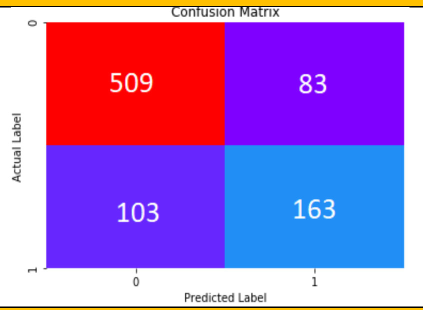
```
              precision    recall  f1-score   support

           0       0.83      0.85      0.84       592
           1       0.65      0.61      0.63       266

    accuracy                           0.78       858
   macro avg       0.74      0.73      0.73       858
weighted avg       0.77      0.78      0.77       858
```

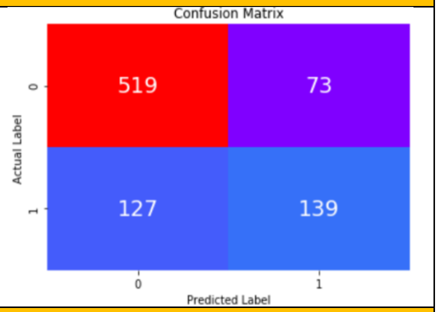
```
              precision    recall  f1-score   support

           0       0.83      0.86      0.85       592
           1       0.66      0.61      0.64       266

    accuracy                           0.78       858
   macro avg       0.75      0.74      0.74       858
weighted avg       0.78      0.78      0.78       858
```

```
              precision    recall  f1-score   support

           0       0.80      0.88      0.84       592
           1       0.66      0.52      0.58       266

    accuracy                           0.77       858
   macro avg       0.73      0.70      0.71       858
weighted avg       0.76      0.77      0.76       858
```

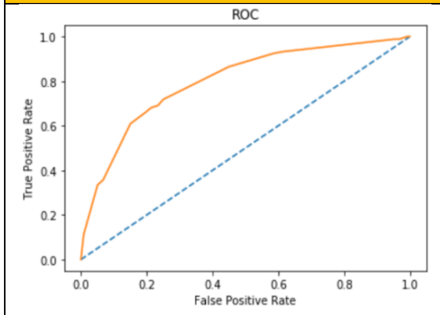
Confusion Matrix

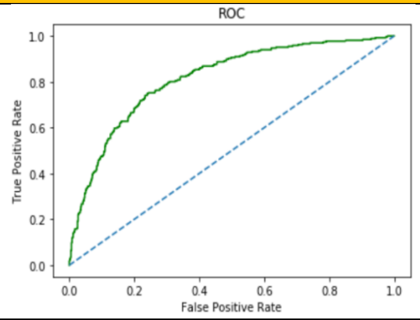


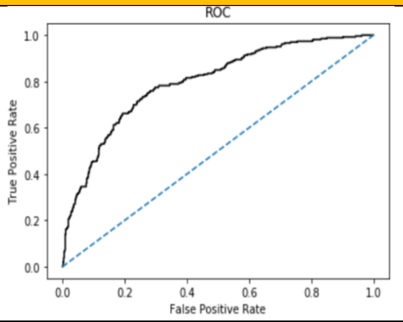




ROC curve





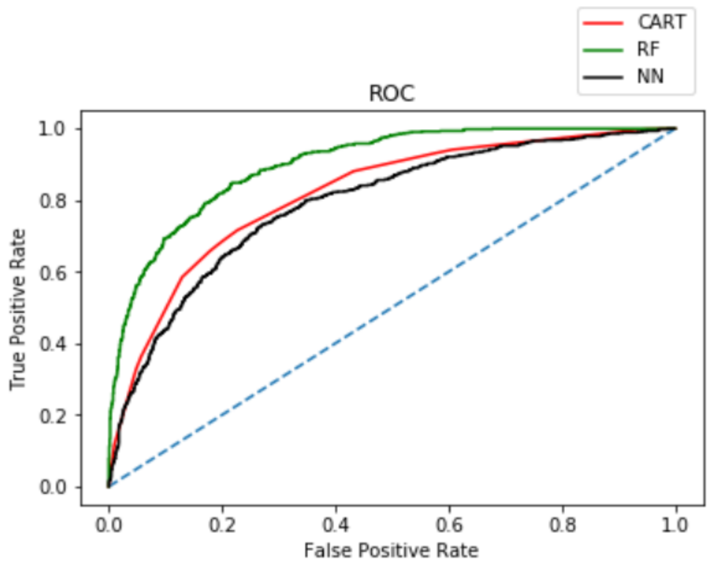


	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.78	0.78	0.78	0.78	0.75	0.77
AUC	0.82	0.80	0.79	0.74	0.68	0.70
Recall	0.58	0.61	0.67	0.61	0.48	0.52
Precision	0.68	0.65	0.77	0.66	0.67	0.66
F1 Score	0.63	0.63	0.72	0.64	0.56	0.58

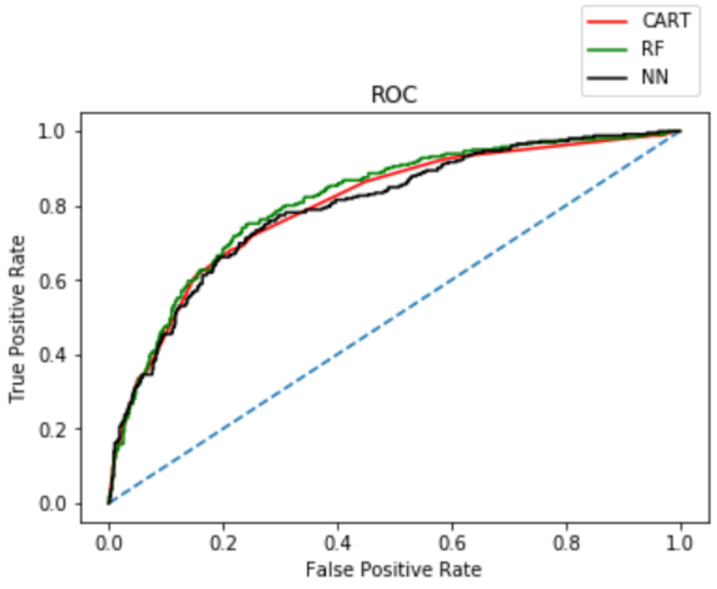
We can observe by looking at the comparison of model values that accuracy, AUC, Precision and Recall for test data is almost in line with training data. This proves no overfitting or underfitting has happened, and overall the models are good for classification.

Here the insurance firm in interested in correctly predicting/classifying if a new customer would claim his policy (1) rather than if he would not claim his policy (0), which implies that apart from accuracy, the recall value is also an important metric. We are able to get highest recall values from the random forest model, with 67% in train set and 61% in test set, decision tree following very close in performance. This is evident from the confusion matrix values, where the number of true positives is higher and false negatives is lower in both the cases, which is desired in this case study. The accuracy of train and test model is equally good for these models. The F1-score values are higher for the random forest model. The neural network model has underperformed in this case study. Let us observe their performances by comparing ROC curve for the 3 models:

ROC Curve for the 3 models on the Training data



ROC Curve for the 3 models on the Test data



The ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. As a baseline, a random classifier is expected to give points lying along the diagonal (FPR = TPR). The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

From both the comparisons we can see that random forest model is closest to the top-left corner, closely followed by the decision tree model. Out of the 3 models, random forest performs slightly better than decision tree on the train and test set and they both outperform ANN model.

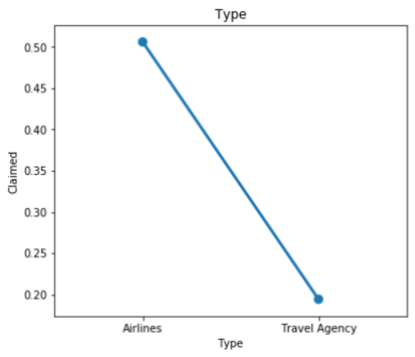
Overall all the 3 models are reasonably stable enough to be used for making any future predictions. From Cart and Random Forest Model, the variable change is found to be the most useful feature amongst all other features for predicting if a person has diabetes or not. If change is yes, then those patients have more chances of getting diabetes.

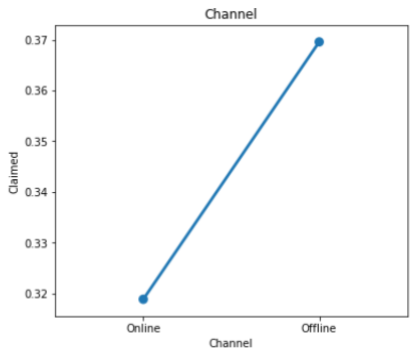
2.5 Inference: Basis on these predictions, what are the business insights and recommendations

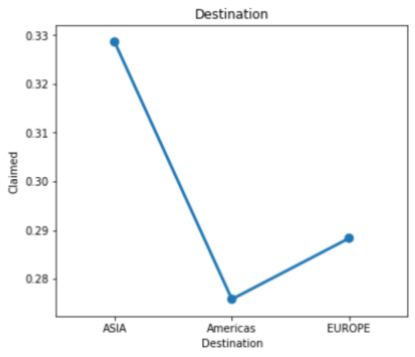
Ans)

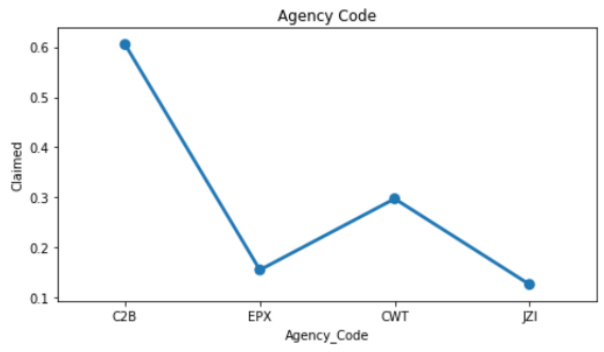
From the exploratory data analysis done in question 1, we were able to figure out following observations from the data by conducting the bivariate analysis.

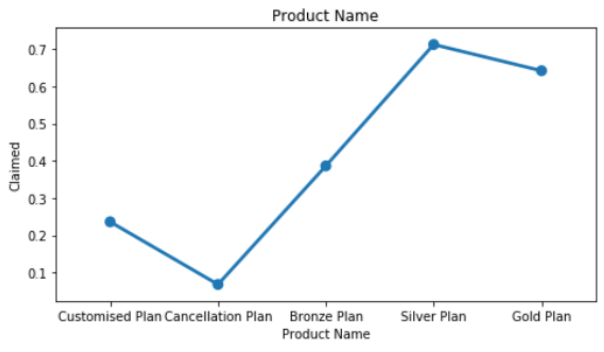
By observing the following point plots, we can check which categories have higher proportion of claims.

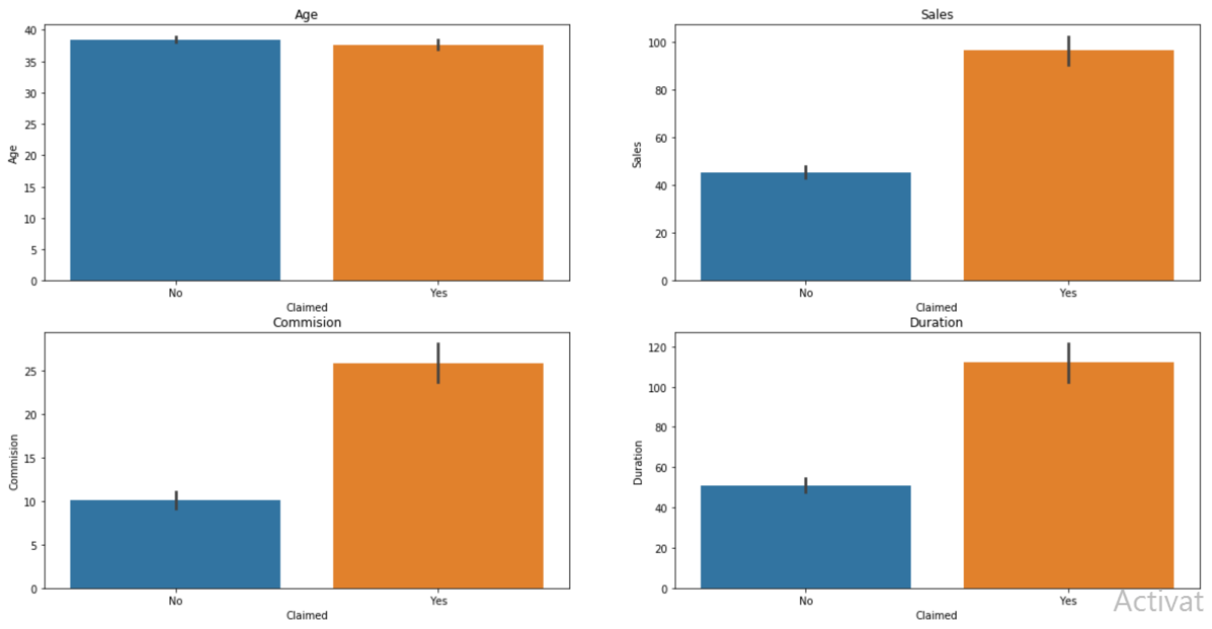




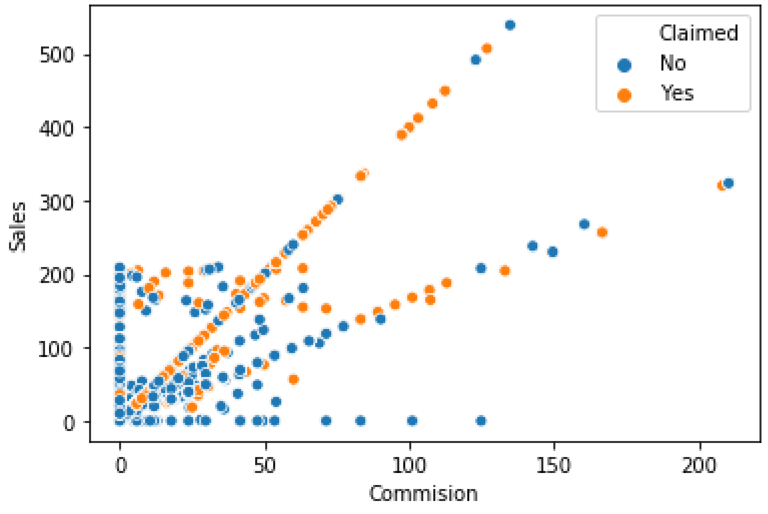




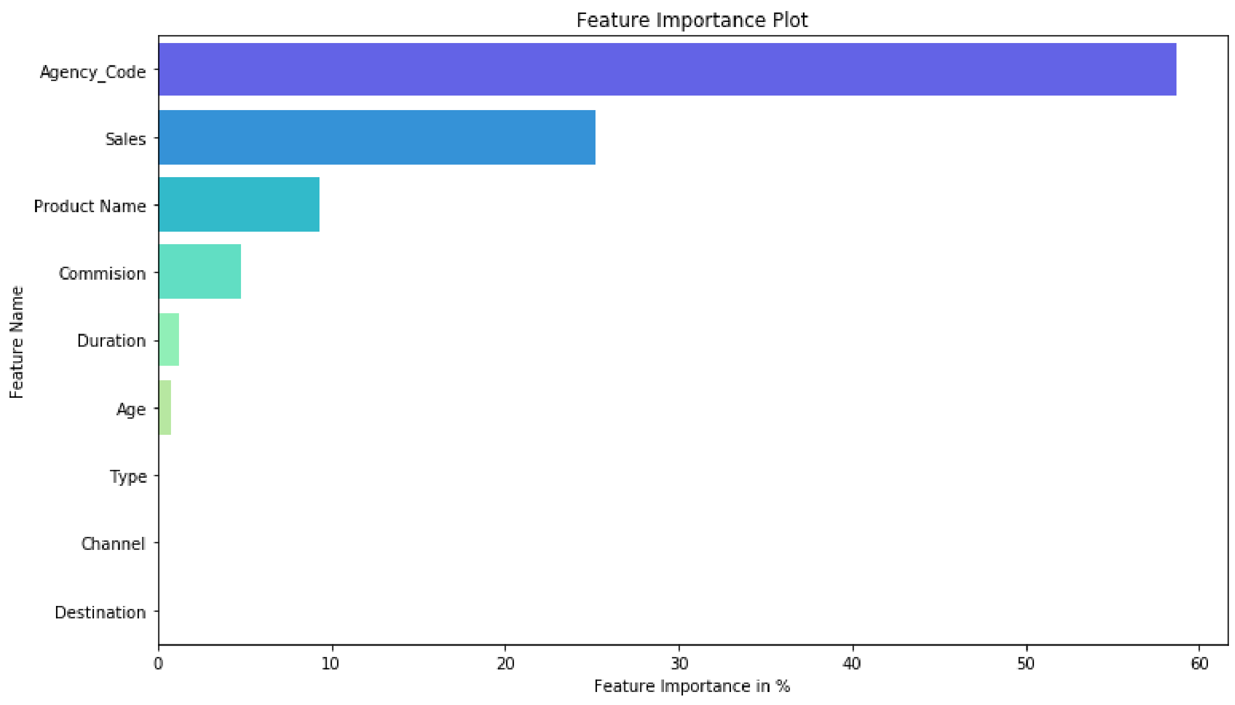




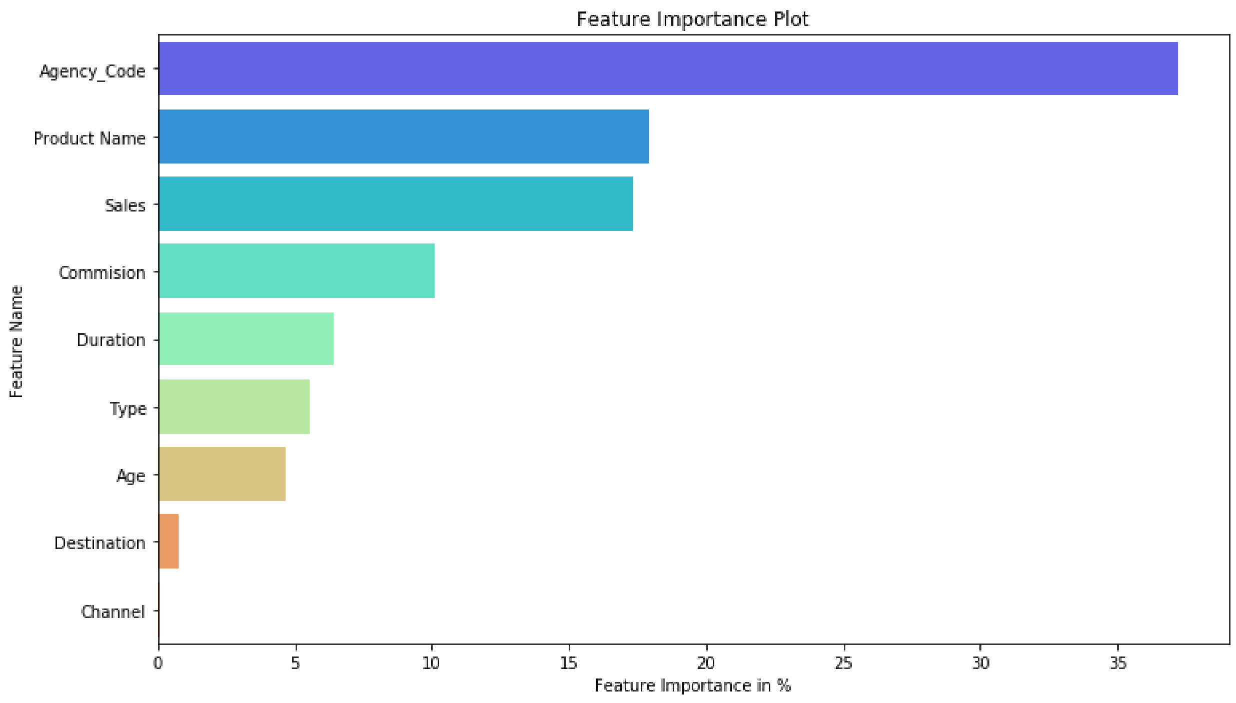
- Airlines have higher claim rates than travel agency
- Tour firm with agency code C2B has highest claim rates followed by CWT. EPX and JZI have very low claim rate
- Offline distribution channel of tour insurance agencies has higher claim rates
- Policies with destination as Asia have higher claims
- Silver plan and gold plan have significantly higher claim rates as compared to other products
- Age doesn’t seem to be a defining factor when it comes to claims
- The claims are higher if the duration, sales and commission values are higher. It can be visualized from the following graph too.



Feature importance in CART model

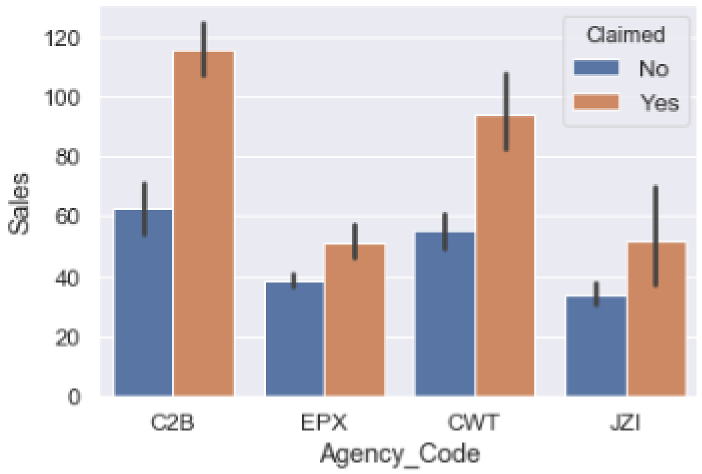


Feature importance in Random Forest model



Agency code plays the most important part in determining if a person will claim policy or not since it has highest feature importance in both the models.

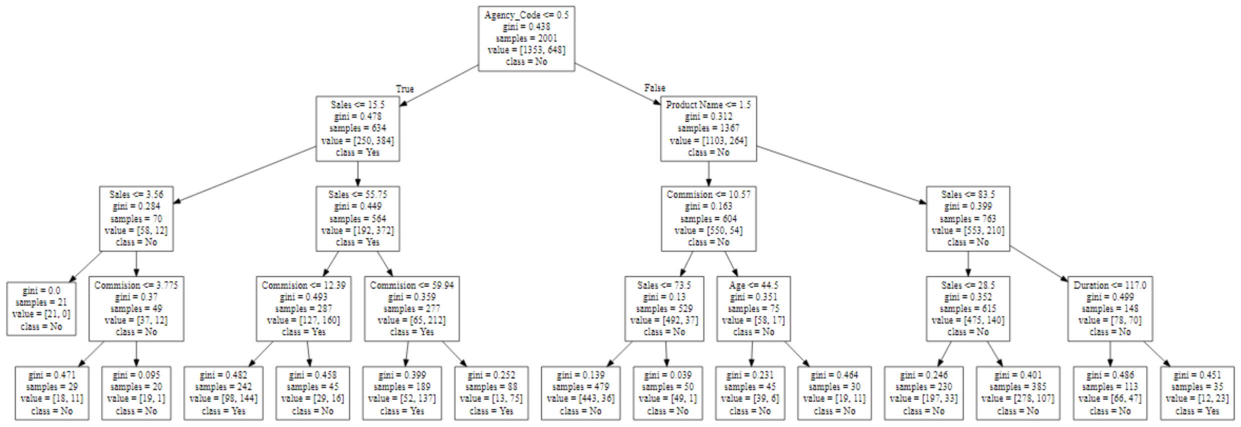
Product name and the amount of sales follow next as equally important factors as per the random forest model.



Channel will not be a deciding factor, so does the destination it seems as per the above graphs and can be omitted while consideration. The age has an equal impact on the probability that a customer will claim a policy or not, which will also be not useful for business to take into consideration.

Customers having gold plan and silver plan have more frequency claimed their policy, so the firm can look into this, what can be modified in these plans so that they bear minimum losses.

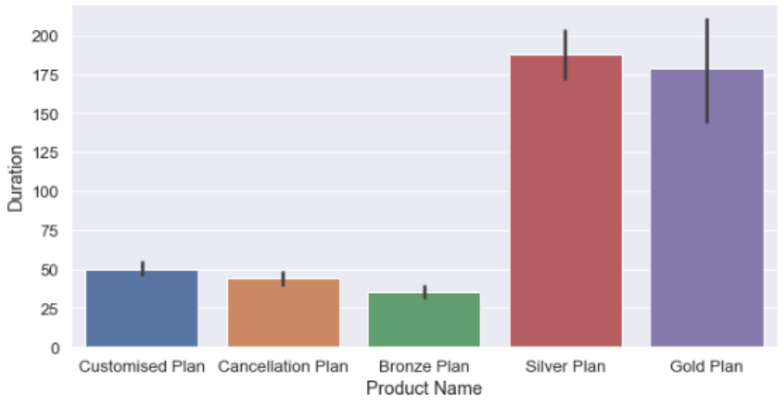
We will try to look at the optimized decision tree if we can get some useful information from that



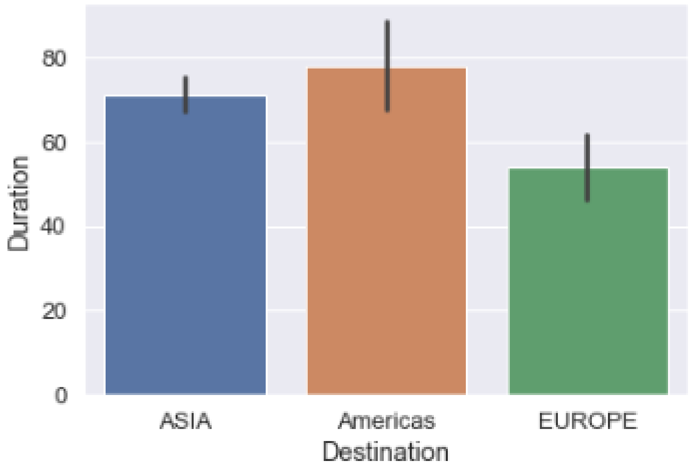
As the most important feature was considered as agency code, it is present at root node. If agency_code <= 0.5 means if agency is C2B, as it was assigned code 0, (line 45 jupyter notebook problem 2), it goes to the left and if its either of the other agency_codes then it traverses the right side. The majority samples in the root node are of no claim hence class is no.

If the agency_code is C2B and it has lower sales (<=15.5) then most likely the customer won't claim their policy as can be observed from above graph. And if the sales are higher then that then observe the commission received. If the commission is between 12.39 and 59.94 which can be considered high looking at the data statistics, then they there are high chances that the customers would claim the policy in future. Since more sales are desirable for the firm, the insurance firm could think of reducing the commission to curb the chances of claim.

From the extreme right-side leaf node, we can infer that higher duration increases the chances of claims, maybe try to decrease the duration of policies which have higher chances of claims. One interesting observation can be as follows



The products with highest claims have highest duration, so the firm could reduce the duration of these tour insurance products to reduce probability of claims. Same can be thought for destination.


Duration
ASIA
Americas
EUROPE
Destination