Business Report

Capstone Project on Tourism Package Adoption

Submitted By: Preeti Singh

Batch Name: PGPDSBA online_Feb20A

Table of Content

Problem: Tourism

A reputed tourism company is planning to launch a long term travel package. The Product Manager has access to the existing customers' data and information. He wishes to analyse the trend of existing customers to figure out which customer is going to purchase the long term travel package.

Problem Understanding:

Defining Problem Statement:

This data is basically about of tourism based company. It's objective is to launch long term travel package and offered the product to customers which belongs to probably subscription based customers (the customers who had paid money to get membership of the organisation to buy a product).There is total 4888 rows and 20 columns in the data set. To check the viability of market they have gone out to certain no of customers and calculated all features of the data that is mentioned in the data .On the behalf of this we have to predict whether a customer is taken a long term travel product or not.

Need of study/Project:

Tourism is a favorite leisure activity. The motivation which causes someone to choose certain activities and a destination for vacation is an interesting issue, which allows for a better understanding of people's behavior in the area of leisure spending.

Understanding Business and Social Opportunity:

Social tourism improves the well-being of people and reduces stress, improves physical and mental health, increases self-esteem and confidence, enables families to develop positive relationships, provides new skills, and even helps increase employment **opportunities**.

Data Report:

First we will import all necessary libraries. Then load, view and get high level understanding of data set.

Checking the quantum of data:

```
the number of rows 4888
the number of columns 20
```

Checking the data types:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4888 entries, 0 to 4887
Data columns (total 20 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   CustomerID               4888 non-null   int64
 1   ProdTaken                4888 non-null   int64
 2   Age                      4662 non-null   float64
 3   PreferredLoginDevice     4863 non-null   object
 4   CityTier                 4888 non-null   int64
 5   DurationOfPitch          4637 non-null   float64
 6   Occupation               4888 non-null   object
 7   Gender                   4888 non-null   object
 8   NumberOfPersonVisited    4888 non-null   int64
 9   NumberOfFollowups        4843 non-null   float64
 10  ProductPitched           4888 non-null   object
 11  PreferredPropertyStar    4862 non-null   float64
 12  MaritalStatus            4888 non-null   object
 13  NumberOfTrips            4748 non-null   float64
 14  Passport                 4888 non-null   int64
 15  PitchSatisfactionScore   4888 non-null   int64
 16  OwnCar                   4888 non-null   int64
 17  NumberOfChildrenVisited  4822 non-null   float64
 18  Designation              4888 non-null   object
 19  MonthlyIncome            4655 non-null   float64
dtypes: float64(7), int64(7), object(6)
memory usage: 763.9+ KB
```

Observations:

Here, the features ProdTaken(target variable),CityTier,OwnCar and Passport ,PreferredPropertyStar are actually categorical in nature but in the data set all are in numerical (int/flaot) type. We need to convert these into object type for further analysis. ProdTaken is target variable and rest of all are predictor(input variables).

Checking the descriptive statistics of data:

	count	mean	std	min	25%	50%	75%	max
CustomerID	4888.0	202443.500000	1411.188388	200000.0	201221.75	202443.5	203665.25	204887.0
ProdTaken	4888.0	0.188216	0.390925	0.0	0.00	0.0	0.00	1.0
Age	4662.0	37.622265	9.316387	18.0	31.00	36.0	44.00	61.0
CityTier	4888.0	1.654255	0.916583	1.0	1.00	1.0	3.00	3.0
DurationOfPitch	4637.0	15.490835	8.519643	5.0	9.00	13.0	20.00	127.0
NumberOfPersonVisited	4888.0	2.905074	0.724891	1.0	2.00	3.0	3.00	5.0
NumberOfFollowups	4843.0	3.708445	1.002509	1.0	3.00	4.0	4.00	6.0
PreferredPropertyStar	4862.0	3.581037	0.798009	3.0	3.00	3.0	4.00	5.0
NumberOfTrips	4748.0	3.236521	1.849019	1.0	2.00	3.0	4.00	22.0
Passport	4888.0	0.290917	0.454232	0.0	0.00	0.0	1.00	1.0
PitchSatisfactionScore	4888.0	3.078151	1.365792	1.0	2.00	3.0	4.00	5.0
OwnCar	4888.0	0.620295	0.485363	0.0	0.00	1.0	1.00	1.0
NumberOfChildrenVisited	4822.0	1.187267	0.857861	0.0	1.00	1.0	2.00	3.0
MonthlyIncome	4655.0	23619.853491	5380.698361	1000.0	20346.00	22347.0	25571.00	98678.0

Observations:

- At least 50% customers are in age of 35 to 36(younger age grup) that is closer to average age of customers also.
- At least 50% customers belong to Tier-1 city. It means 50% customers belong to metropolitan city.
- At least 75% customers come up with 3 persons to visit the company.
- At least 50% customers preferred to stay in 3 star hotels.
- At least 50% customers are having no passport. It means they are local traveller.
- At least 50% customers are having own car, may be they use their own car for travelling.
- At least 50% customers are done total no of trips 3.It means these customers can do travelling most frequently.
- An average monthly income of customers is 23619.
- Out of 4888 customers on an average total 920(18 %) customers are taken long term travel package.

Let's convert the features that are actually in categorical nature but in data set in numerical nature, into appropriate data type for further analysis.

The features ProdTaken,Passport,OwnCar are having binary values. We will convert all these variables into object type by assigning 1 == Yes and 0==No with the lambda( ) and the features CityTier and preferredPropertyStar are having ordered values, so we will labelled different name for each different values. For CityTier feature we will assign 1==Tier-1,2===Tier-2 and 3==Tier-3 and for feature PreferredPropertyStar, we will replace all nan values==Unknown,4==4 Star,3==3 Star,2==2 Star,1==1Star.

Checking the info of data set df_tourism1:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4888 entries, 0 to 4887
Data columns (total 20 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   CustomerID               4888 non-null   int64
 1   ProdTaken                4888 non-null   object
 2   Age                      4662 non-null   float64
 3   PreferredLoginDevice     4863 non-null   object
 4   CityTier                 4888 non-null   object
 5   DurationOfPitch          4637 non-null   float64
 6   Occupation               4888 non-null   object
```

```
 7   Gender                   4888 non-null   object
 8   NumberOfPersonVisited    4888 non-null   int64
 9   NumberOfFollowups        4843 non-null   float64
 10  ProductPitched           4888 non-null   object
 11  PreferredPropertyStar    4862 non-null   object
 12  MaritalStatus            4888 non-null   object
 13  NumberOfTrips            4748 non-null   float64
 14  Passport                 4888 non-null   object
 15  PitchSatisfactionScore   4888 non-null   int64
 16  OwnCar                   4888 non-null   object
 17  NumberOfChildrenVisited  4822 non-null   float64
 18  Designation              4888 non-null   object
 19  MonthlyIncome            4655 non-null   float64
dtypes: float64(6), int64(3), object(11)
memory usage: 763.9+ KB
```

Now, all int/float type categorical variables are in object type.

Exploratory Data Analysis :

Checking missing values: We can check missing values by using df_tourism1.isnull().sum().

```
DurationOfPitch            251
MonthlyIncome              233
Age                        226
NumberOfTrips              140
NumberOfChildrenVisited     66
NumberOfFollowups           45
PreferredPropertyStar       26
PreferredLoginDevice        25
Passport                     0
MaritalStatus                0
ProductPitched               0
Designation                  0
NumberOfPersonVisited        0
Gender                       0
Occupation                   0
PitchSatisfactionScore       0
CityTier                     0
OwnCar                       0
ProdTaken                    0
CustomerID                   0
dtype: int64
```

There are so many missing values present in data. We need to take care of this for future analysis.

DurationOfPitch,MonthlyIncome,Age,NumberOfTrips,NumberOfChildrenVisited,NumberOfFollwups are numerical variable wherein missing values are present.

PreferredPropertyStar,PreferredLoginDevice are categorical variable wherein missing values are present.

Checking of total no of missing values:

df_tourism1.isnull().sum().sum()

```
1012
```

Let's see count plot of missing values:

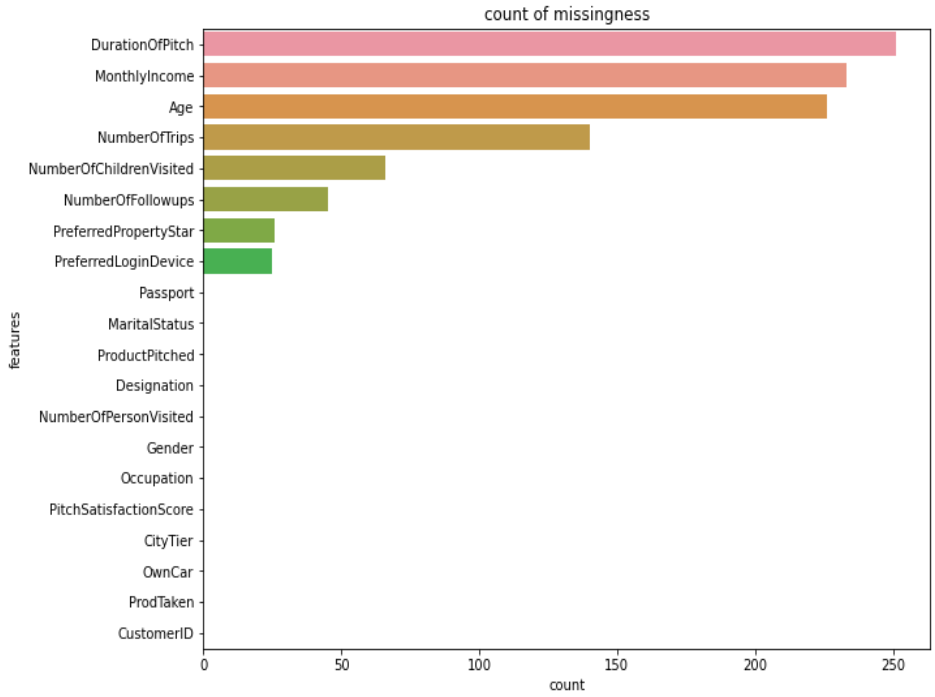

Fig-1

From the above plot we can see feature DurationOfPitch has highest count of missing values.

Calculating propensity of missing values:

```
DurationOfPitch            0.051350
MonthlyIncome              0.047668
Age                        0.046236
NumberOfTrips              0.028642
NumberOfChildrenVisited    0.013502
NumberOfFollowups          0.009206
PreferredPropertyStar      0.005319
PreferredLoginDevice       0.005115
```

```
Passport                  0.000000
MaritalStatus             0.000000
ProductPitched            0.000000
Designation               0.000000
NumberOfPersonVisited     0.000000
Gender                    0.000000
Occupation                0.000000
PitchSatisfactionScore    0.000000
CityTier                  0.000000
OwnCar                    0.000000
ProdTaken                 0.000000
CustomerID                0.000000
dtype: float64
```

Observations:

We can observe from the above output, there are some missing values present in numerical variable and categorical variable as well and the extent of missing ness is not so high. It is varying from 0.5% to 5.1%.We can opt removing these observation because variation of missing ness is not high but we will try and impute these missing values to best extent as possible.

Treating of Missing values by median and mode:

Let's separate the numerical and categorical variable first. We can treat the missing values that are present in numerical variable by median().and we can use mode() for categorical variable. We are referring df_tourism2_num data frame for numerical variable and df_tourism2_cat for categorical variable.

Missing values imputation for numerical variable by using median().

```
df_tourism2_num["Age"]=df_tourism2_num["Age"].fillna(df_tourism2_num["Age"].median())

df_tourism2_num["DurationOfPitch"]=df_tourism2_num["DurationOfPitch"].fillna(df_tourism2_num["DurationOfPitch"].median())

df_tourism2_num["NumberOfFollowups"]=df_tourism2_num["NumberOfFollowups"].fillna(df_tourism2_num["NumberOfFollowups"].median())

df_tourism2_num["NumberOfTrips"]=df_tourism2_num["NumberOfTrips"].fillna(df_tourism2_num["NumberOfTrips"].median())

df_tourism2_num["NumberOfChildrenVisited"]=df_tourism2_num["NumberOfChildrenVisited"].fillna(df_tourism2_num["NumberOfChildrenVisited"].median())

df_tourism2_num["MonthlyIncome"]=df_tourism2_num["MonthlyIncome"].fillna(df_tourism2_num["MonthlyIncome"].median())
```

Now we are going to check missing values only for numerical variable by df_tourism2_num.isnull().sum()

```
CustomerID                0
Age                       0
DurationOfPitch           0
NumberOfPersonVisited     0
NumberOfFollowups         0
NumberOfTrips             0
PitchSatisfactionScore    0
NumberOfChildrenVisited   0
MonthlyIncome             0
dtype: int64
```

There are no missing values present in data after imputation.

Missing values imputation for categorical variable by mode().

```
df_mode=df_tourism2_cat["PreferredLoginDevice"].mode()[0]

df_mode

'Self Enquiry'
df_mode1=df_tourism2_cat["PreferredPropertyStar"].mode()[0]

df_mode1

'3 Star'
df_tourism2_cat["PreferredLoginDevice"]=df_tourism2_cat["PreferredLoginDevice"].replace(np.nan,df_mode)

df_tourism2_cat["PreferredPropertyStar"]=df_tourism2_cat["PreferredPropertyStar"].replace(np.nan,df_mode1)
```

Let’s check missing values for categorical variable after imputation:

```
ProdTaken               0
PreferredLoginDevice    0
CityTier                0
Occupation              0
Gender                  0
ProductPitched          0
PreferredPropertyStar   0
MaritalStatus           0
Passport                0
OwnCar                  0
Designation             0
dtype: int64
```

There are no missing values present in the data after imputation.

After treating the missing values we will concate the numerical and categorical variables and create a new data frame df_tourism2.

```
df_tourism2 = pd.concat([df_tourism2_cat,df_tourism2_num],axis=1)
```

Checking of unique values present in categorical columns:

```
ProdTaken
No     3968
Yes     920
Name: ProdTaken, dtype: int64

PreferredLoginDevice
Self-Enquiry       3469
Company Invited    1419
Name: PreferredLoginDevice, dtype: int64

CityTier
Tier-1    3190
Tier-3    1500
Tier-2     198
Name: CityTier, dtype: int64

Occupation
Salaried          2368
Small Business    2084
Large Business     434
Free Lancer          2
Name: Occupation, dtype: int64

Gender
Male       2916
Female     1817
Fe Male     155
Name: Gender, dtype: int64

ProductPitched
Multi           1842
Super Deluxe    1732
Standard         742
Deluxe           342
King             230
Name: ProductPitched, dtype: int64

PreferredPropertyStar
3 Star    3019
5 Star     956
4 Star     913
Name: PreferredPropertyStar, dtype: int64

MaritalStatus
Married      2340
Divorced      950
Single        916
Unmarried     682
Name: MaritalStatus, dtype: int64

Passport
No     3466
Yes    1422
Name: Passport, dtype: int64

OwnCar
Yes    3032
```

```
No     1856
Name: OwnCar, dtype: int64

Designation
Executive         1842
Manager           1732
Senior Manager     742
AVP                342
VP                 230
Name: Designation, dtype: int64
```

Observations:

Here we can see, total count of each labelled categorical variable. **Something th at we found here, there is unstructured label Fe Male in Gender column see ms like bad data with 155 records**. We need to take care of this. We should re place Fe Male by Female for further analysis. Only 2 records of Free Lancer. Th ey are
100 % probability that they sell their product.

```
df_tourism2['Gender'] = df_tourism2['Gender'].apply(lambda x: 'Female' if x == 'Fe Male' else x)
```

Checking of propensity in target variable:

```
No     0.811784
Yes    0.188216
Name: ProdTaken, dtype: float64
```

Only 18% customers is going to opt long term travel package. Also, this is an i m-balance data set because no of 1's is more than 0's.

Checking of duplicates rows: checking of duplicates rows by df_tourism2.du plicated().

```
total no of duplicates rows 0
```

Removal of unwanted variable: Here , no need of CustomerID column for a n-alysis.

```
df_tourism2=df_tourism2.drop(["CustomerID"],axis=1)
```

Also features NumberOfTrips,NumberOfChildrenVisited,NumberOfPersonVisit ed,NumberOfFollowUps are having fraction values. So we should take it as rou nd number before doing visualization for better understanding.

Let's separate out all the numerical variables and categorical variabl es
from df_tourism2 data set.
df_tourism2_num

```
Index(['Age', 'DurationOfPitch', 'NumberOfPersonVisited',
       'NumberOfFollowups', 'NumberOfTrips', 'PitchSatisfactionScore',
       'NumberOfChildrenVisited', 'MonthlyIncome'],
      dtype='object')
```

df_tourism2_cat

```
Index(['ProdTaken', 'PreferredLoginDevice', 'CityTier', 'Occupation', '
Gender',
       'ProductPitched', 'PreferredPropertyStar', 'MaritalStatus', 'Pas
sport','OwnCar', 'Designation'], dtype='object')
```

Let's do univariate analysis for all numerical and categorical variables. The dat a

set that are used in below univariate analysis is df_tourism2.

Univarate analysis for Numerical Variable:

Univariate analysis by using distplot() and boxplot() for each and every numeric al variables.

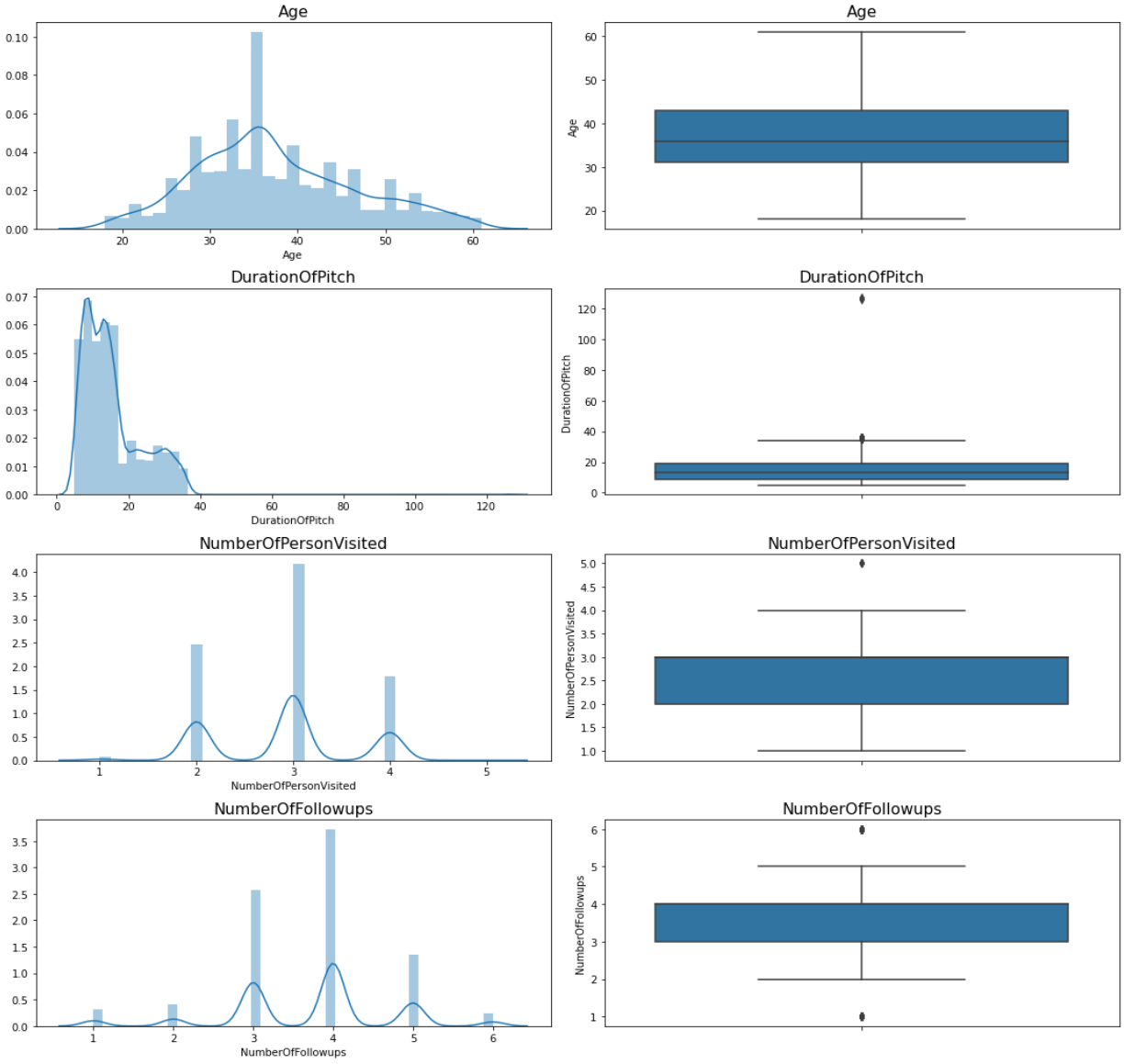

Fig-2

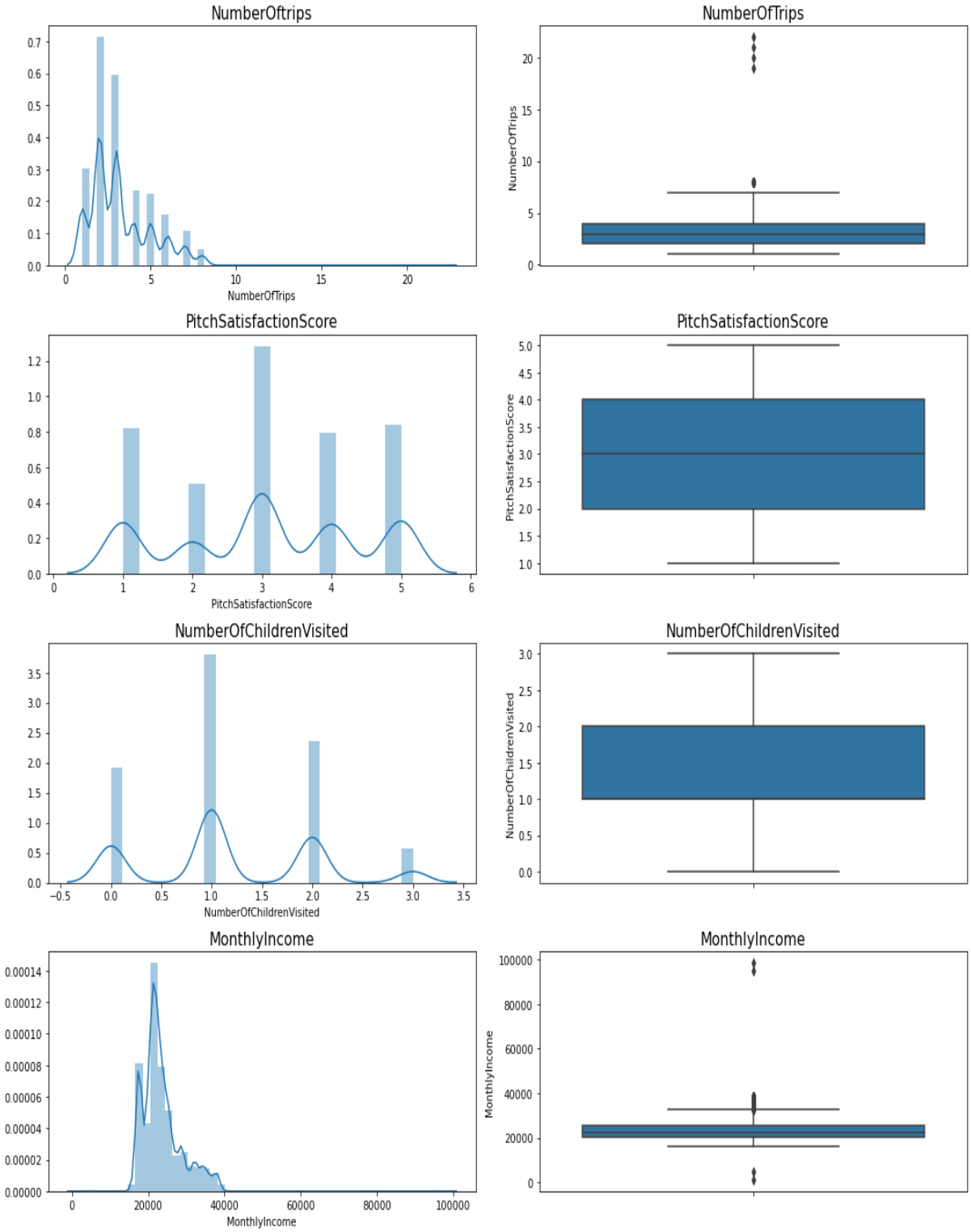

Fig-3

Observations:

- Among all the above numeric variables, only Age is having unimodal dist r-ibution (single peak) .So we can say Age is normally distrusted. Others

numerical variables are having multimodal distribution(Multiple peak). Si nce
this is a classification problem, we can choose to leave such variables as t hey are. To get rid of such multimodal distribution, we can use Binning a pproach wherein we can create buckets.

- Also there are outliers present in variables NumberOfFollowUps,Number OfPersonVisited,NumberOfTrips,MonthlyIncome.There is different appr oaches to handle outlier. We can remove outlier, retain outlier and can do imputation also. This totally depends upon business problem that we are d ealing. We will do it later for further analysis.
- 50% customers are in age 35-36(younger age group) and their monthly in come in rage of 21000 to 23000.

Univariate Analysis of all Categorical Variables:

Univariate analysis of all categorical variable by countplot().

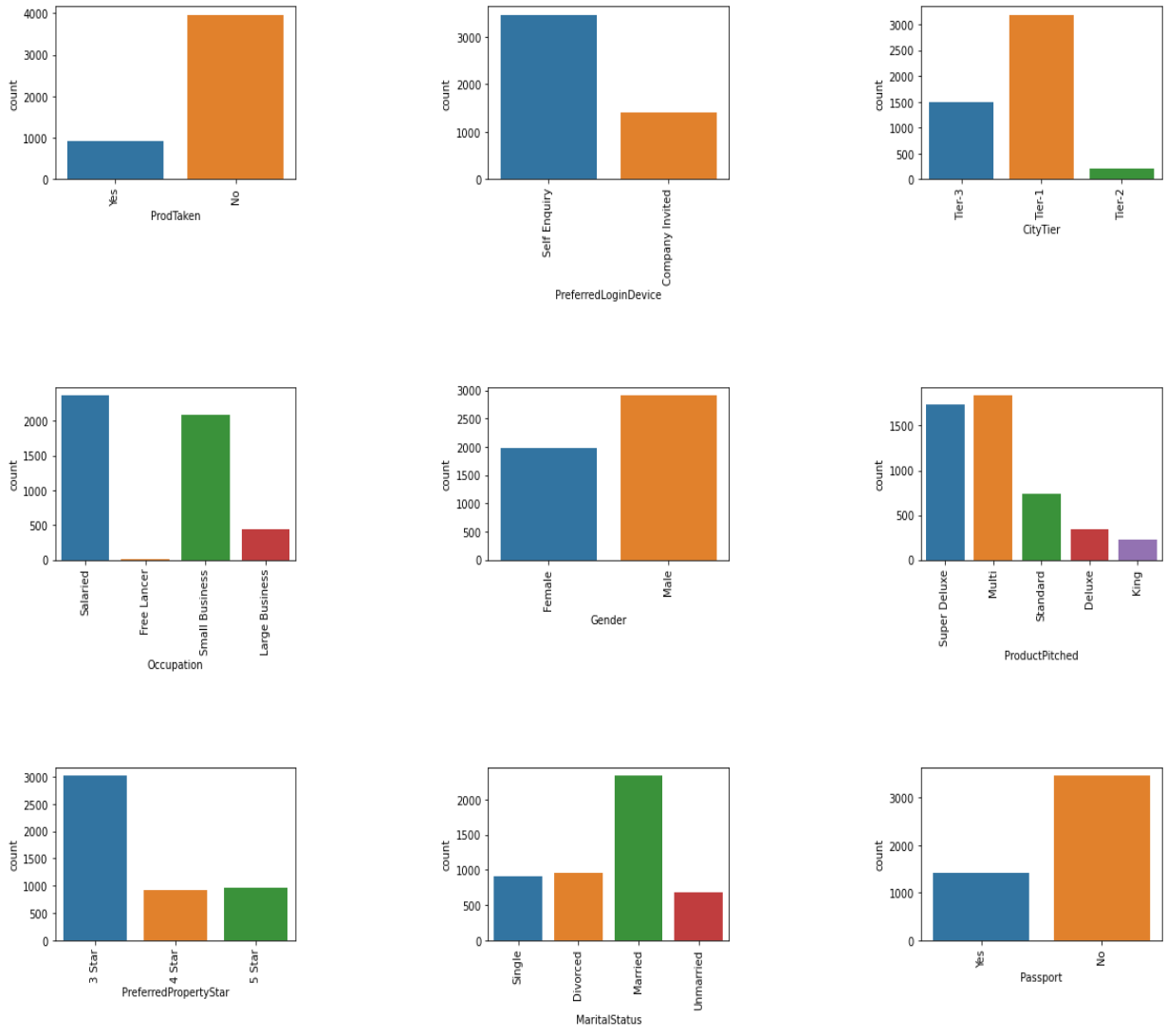

Fig-4

Observations:

- Most of the customers are not taken product.
- Most of the customers come up by themselves.
- Most of the customers do not have passport.
- Most of the customers are gender.
- Most of the customers are taken super deluxe and multi package.
- Most of the customers prefer to stay in 3-Star.
- Most of the customers belong to Tier-1.
- Most of the customer's occupations are salaried and small business.

Conclusion:

- From the above inferences of the categorical variable, we can conclude that most of customers live in metropolitan city.
- Since the customers belong to small occupation (salaried and small busi nes), hence we can conclude that, they have small monthly income, the y cannot afford more no of trips, may be they will buy cheaper product and Super Deluxe and multi product.
- Since some of the customers do not have passport but they are taking th e product, so we can conclude these customers are domestic traveller.

Feature Engineering:

To get rid from multimodal distribution from that is present in numerical variables in df_tourism2 data set, we are going to use **Binning.** For the sake of further analysis we have taken df_tourism3 data set. This comes under **feature engineering and it is itself divide into two parts:**

1. Variable Transformation

2. Variable Creations

There are many approaches that are used in variable transformation and variable creation. Binning is one of the approach that I have used in variable transformation. See Appendix A.

Binning using quartiles: durationOfPitch:

Let's check descriptive statistics of variable DurationOfPitch:

```
count    4888.000000
mean       15.362930
std         8.316166
min         5.000000
25%         9.000000
```

```
50%         13.000000
75%         19.000000
max        127.000000
Name: DurationOfPitch, dtype: float64
```

After that we will create binning variable durationOfPitch_bins:

```
Really Low    1471
High          1199
Low           1118
Medium        1100
Name: DurationOfPitch_bins, dtype: int64
```

Here we have done labelling on the behalf of quartiles like from range min to25 % named as Really Low, from range 25% to 50% named as Low, from 50% to 75% named as Medium and from 75% up to max named as High.

Binning using quartile:NumberOfFollowups:

```
Medium    2081
Low       1903
High       904
Name: NumberOfFollowups_bins, dtype: int64
```

Binning using quartiles: NumberOfTrips:

```
Low          2084
Very High    1114
Medium       1081
High          609
Name: NumberOfTrips_bins, dtype: int64
```

Binning using map function: PitchSatisfactionScore:

```
Good         1478
Excellent     970
Bad           942
Very Good     912
OK            586
Name: PitchSatisfactionScore_bins, dtype: int64
```

Binning using lambda function: NumberOfPersonVisited

```
Three and above    3431
One or Two         1457
Name: NumberOfPersonVisited_bins, dtype: int64
```

Checking of data types after binning:

After bucketing, we have to drop all the variables that are used for binning. No w, we will check info() of data to check the new variables that are created in bin ning and also data types of each new variables.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4888 entries, 0 to 4887
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   ProdTaken                       4888 non-null   object
 1   PreferredLoginDevice            4888 non-null   object
 2   CityTier                        4888 non-null   object
 3   Occupation                      4888 non-null   object
 4   Gender                          4888 non-null   object
 5   ProductPitched                  4888 non-null   object
 6   PreferredPropertyStar           4888 non-null   object
 7   MaritalStatus                   4888 non-null   object
 8   Passport                        4888 non-null   object
 9   OwnCar                          4888 non-null   object
 10  Designation                     4888 non-null   object
 11  Age                             4888 non-null   float64
 12  MonthlyIncome                   4888 non-null   float64
 13  DurationOfPitch_bins            4888 non-null   object
 14  NumberOfPersonVisited_bins      4888 non-null   object
 15  NumberOfFollowups_bins          4888 non-null   object
 16  NumberOfTrips_bins              4888 non-null   object
 17  PitchSatisfactionScore_bins     4888 non-null   object
 18  NumberOfChildrenVisited_bins    4888 non-null   object
dtypes: float64(2), object(17)
memory usage: 725.7+ KB
```

Observations:

- Here, we can see, DurationOfPitch_bins,NumberOfPersonVisited_bins,NumberOfFollowups_bins,PitchSatisfactionScore_bins,NumberOfChildrenVisited_bins are the new variables names that we have created while binning and also all these variables are object type.

- We are left with two numerical variables Age and MonthlyIncome.

Univariate analysis for all Numerical Variables:

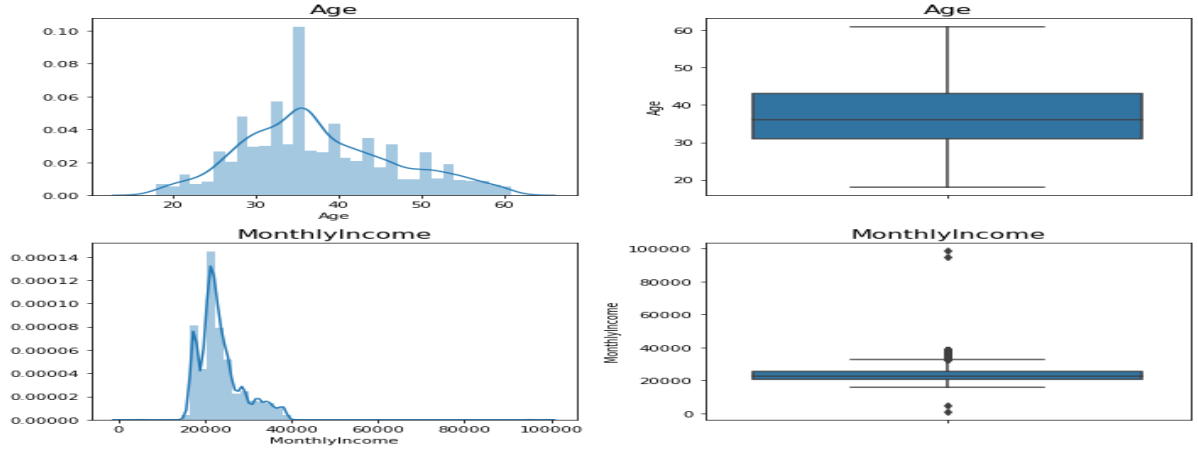

Fig-5

- Age is normally distributed and 50% of customers are in age group35-36(young age group).
- MonthlyIncome is normally distributed.50 % customers are having Mo nthlyIncome range between 2000-2200.There are some outliers also in MonthlyIncome variables that demonstrate that some customers are hav ing very high MonthlyIncome , they might be those customers whose d esignation is high and some of customers have very low MonthlyIncom e , they might be those customers whose belong to small occupation.

Univariate Analysis of all categorical variables:

Here is Univariate analysis for all categorical variables by using count plot.

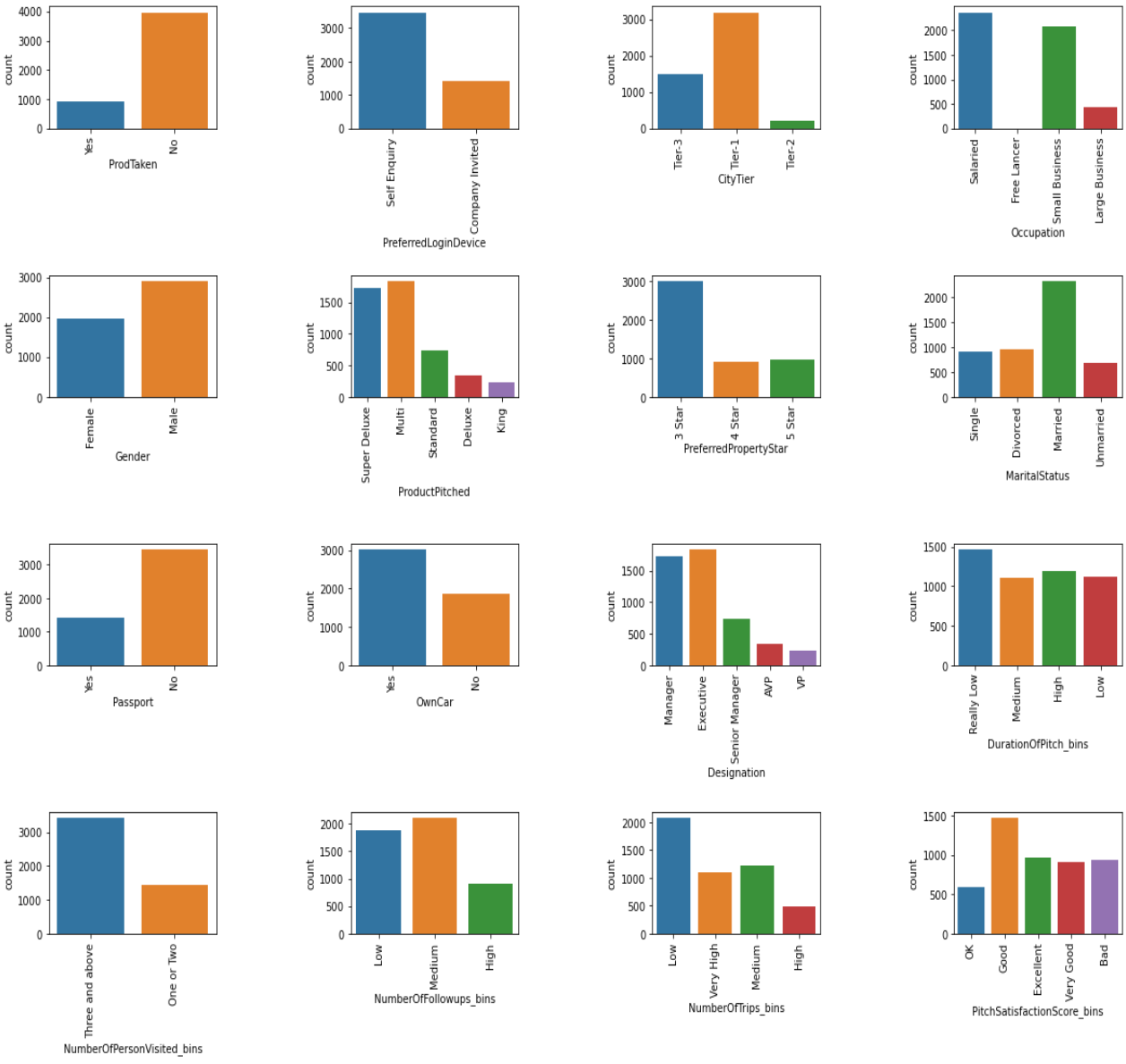

Fig-6

Observations:

- Most of the customers are not taken product.
- Most of the customers come up by themselves.
- Most of the customers belong to city Tier-1(metropolitan city).
- Most of the customers are salaried and have small business.
- Most of the customers are Male.
- Most of the customers are opted multi and Super Deluxe package.
- Most of the customers are preferred to stay in 3-Star.
- Most of the customers are married.
- Most of the customers do not have passport.
- Most of the customers have own car.
- Most of the customer's designations are Executive, Manager.
- Duration of pitch by salesman to customers is really low.
- Most of the customers bring two or three children along with.
- No of follow up is done by sales persons, are medium.
- Most of the customers are done less no of trips in a year.
- Pitch satisfactory score is given by most of the customers, are good.

Conclusion:

- From the above inferences of the categorical variable, we can conclude t hat most of customers live metropolitan city and they belong to middle / upper middle family and they have probably kids and family and own c ar as well.
- Since the customers belong to small occupation (salaried and small busin es), hence we can conclude that, they have small monthly income, they c annot afford more no of trips, may be they will buy cheaper product and Super Deluxe and multi product.
- Most of the customer do not have passport, so we can conclude they all ar e domestic traveller.

Looking at proportion of labelled categorical variable:

```
Proportion of Customers as per ProdTaken

No     0.811784
Yes    0.188216
Name: ProdTaken, dtype: float64

Proportion of Customers as per PreferredLoginDevice

Self Enquiry       0.709697
Company Invited    0.290303
Name: PreferredLoginDevice, dtype: float64
```

```
Proportion of Customers as per CityTier

Tier-1    0.652619
Tier-3    0.306874
Tier-2    0.040507
Name: CityTier, dtype: float64

Proportion of Customers as per Occupation

Salaried          0.484452
Small Business    0.426350
Large Business    0.088789
Free Lancer       0.000409
Name: Occupation, dtype: float64

Proportion of Customers as per Gender

Male      0.596563
Female    0.403437
Name: Gender, dtype: float64

Proportion of Customers as per ProductPitched

Multi           0.376841
Super Deluxe    0.354337
Standard        0.151800
Deluxe          0.069967
King            0.047054
Name: ProductPitched, dtype: float64

Proportion of Customers as per PreferredPropertyStar

3 Star    0.617635
5 Star    0.195581
4 Star    0.186784
Name: PreferredPropertyStar, dtype: float64

Proportion of Customers as per MaritalStatus

Married      0.478723
Divorced     0.194354
Single       0.187398
Unmarried    0.139525
Name: MaritalStatus, dtype: float64

Proportion of Customers as per Passport
```

```
No     0.709083
Yes    0.290917
Name: Passport, dtype: float64

Proportion of Customers as per OwnCar

Yes    0.620295
No     0.379705
Name: OwnCar, dtype: float64

Proportion of Customers as per Designation

Executive         0.376841
Manager           0.354337
Senior Manager    0.151800
AVP               0.069967
VP                0.047054
Name: Designation, dtype: float64

Proportion of Customers as per DurationOfPitch_bins

Really Low    0.300941
High          0.245295
Low           0.228723
Medium        0.225041
Name: DurationOfPitch_bins, dtype: float64

Proportion of Customers as per NumberOfPersonVisited_bins

Three and above    0.701923
One or Two         0.298077
Name: NumberOfPersonVisited_bins, dtype: float64

Proportion of Customers as per NumberOfFollowups_bins

Medium    0.432283
Low       0.382774
High      0.184943
Name: NumberOfFollowups_bins, dtype: float64

Proportion of Customers as per NumberOfTrips_bins

Low          0.426350
Medium       0.249386
Very High    0.226473
High         0.097791
Name: NumberOfTrips_bins, dtype: float64
```

```
Proportion of Customers as per PitchSatisfactionScore_bins

Good          0.302373
Excellent     0.198445
Bad           0.192717
Very Good     0.186579
OK            0.119885
Name: PitchSatisfactionScore_bins, dtype: float64

Proportion of Customers as per NumberOfChildrenVisited_bins

One           0.660393
2 or more     0.339607
Name: NumberOfChildrenVisited_bins, dtype: float64
```

Observations:

- Only 18% customers are taken product.
- 70% customers come up themselves.
- 65% customers belong to Tier-1 (Urban city).
- 33% customers have 2 or more children.
- 70% customers don't have passport.
- 37% customers are working as executive.

Bi-Variate Analysis and Multivariate Analysis:

This is the bivariate analysis across all numerical variables by pairplot() and heatmap().

Fig-7

Observations:

There is hardly any correlation.
See scatter plot between numerical variable in Appendix B.

Bivariate analysis by heatmap():

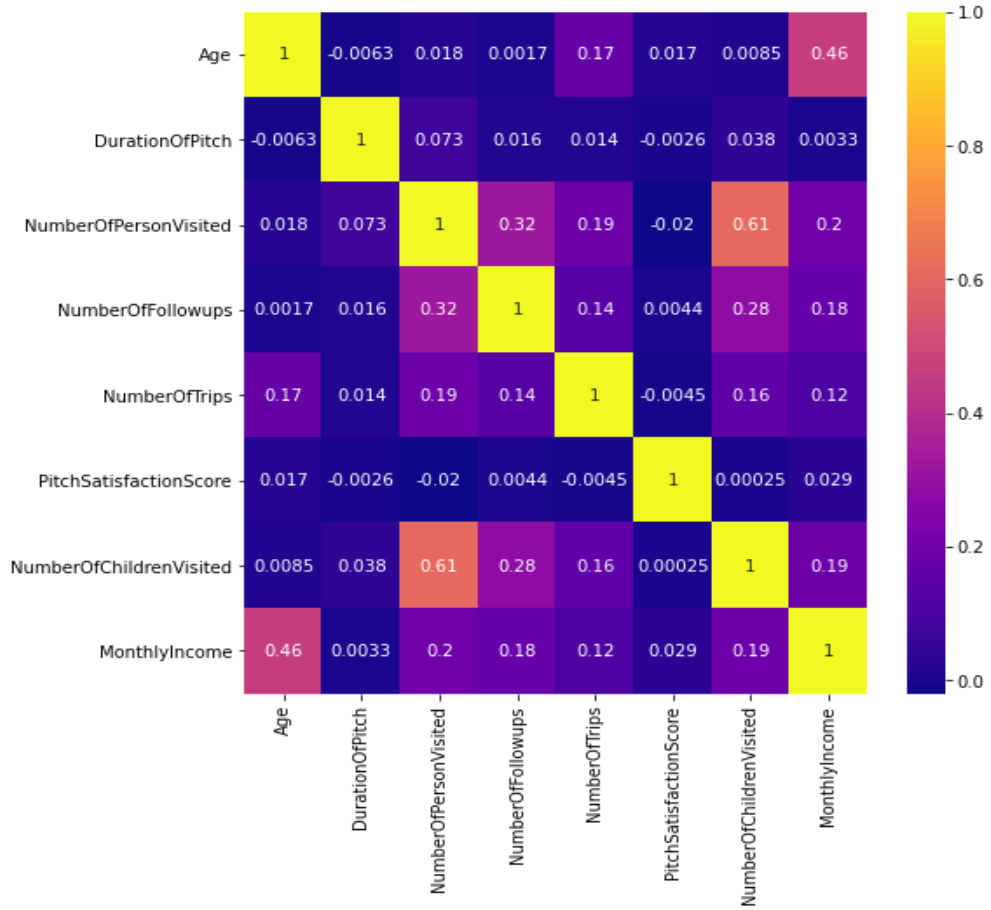

Fig: 8

Observations:

There is only correlation between NumberOfChildrenVisited and Num berOfPersonVisited. As the Number of children visited increases num ber of person visited also increases.

Distribution of age across all categorical variable and binned variable:

Let's see distribution of age across categorical and binned variable by boxplot().

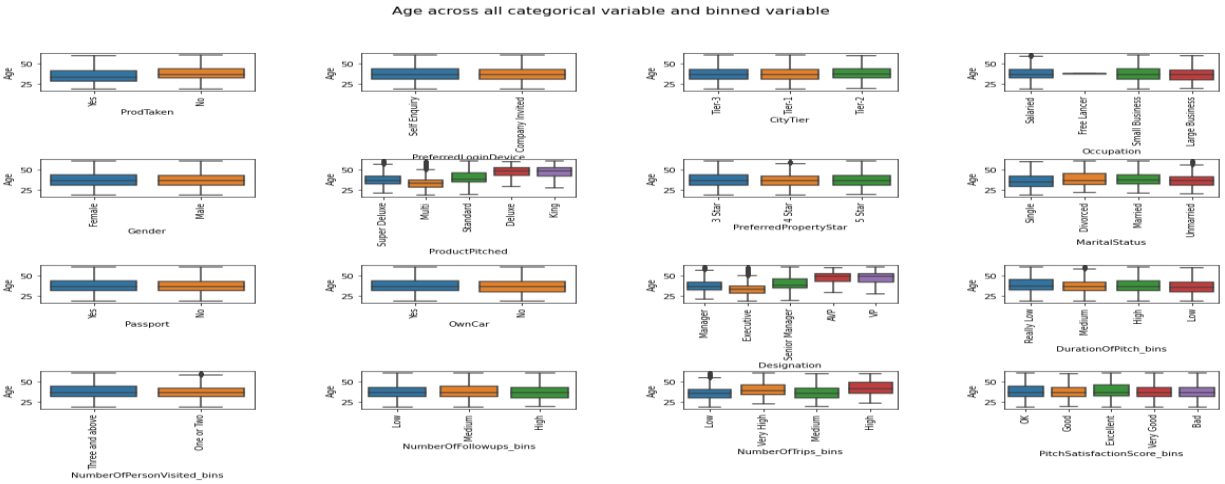

Fig: 9

Observations:

- Median age of the people who has taken product is lesser than who has no taken. It means, the people, who has taken product are lower age group.
- Customers, who are younger group and middle age group, both com e up by themselves and by company invitation.
- 50% of customers whose age above 36+ belong to Tier-2(urban city).
- The customers those occupations are salaried, small business and lar ge business, belong to almost same age group that is middle age gro up.
- The people who are of higher age group have little more income an d their designation is also high. They are working as AVP, VP. That ’s why they are pitching the product Deluxe and king.
- The people who are of higher age group, their number of trips are m ore because their monthly income and designation is high.
- 50% of customers who belong to middle age group are married.
- The people, who belong to old age group and middle age group they are buying expensive product and average range of product like Sta ndard, Deluxe and King. Also there are some outliers are also prese nt in SuperDeluxe and multi product that depicts that some of the ol der age customers are also buying cheaper product.

Distribution of monthly income across categorical and binned variables:

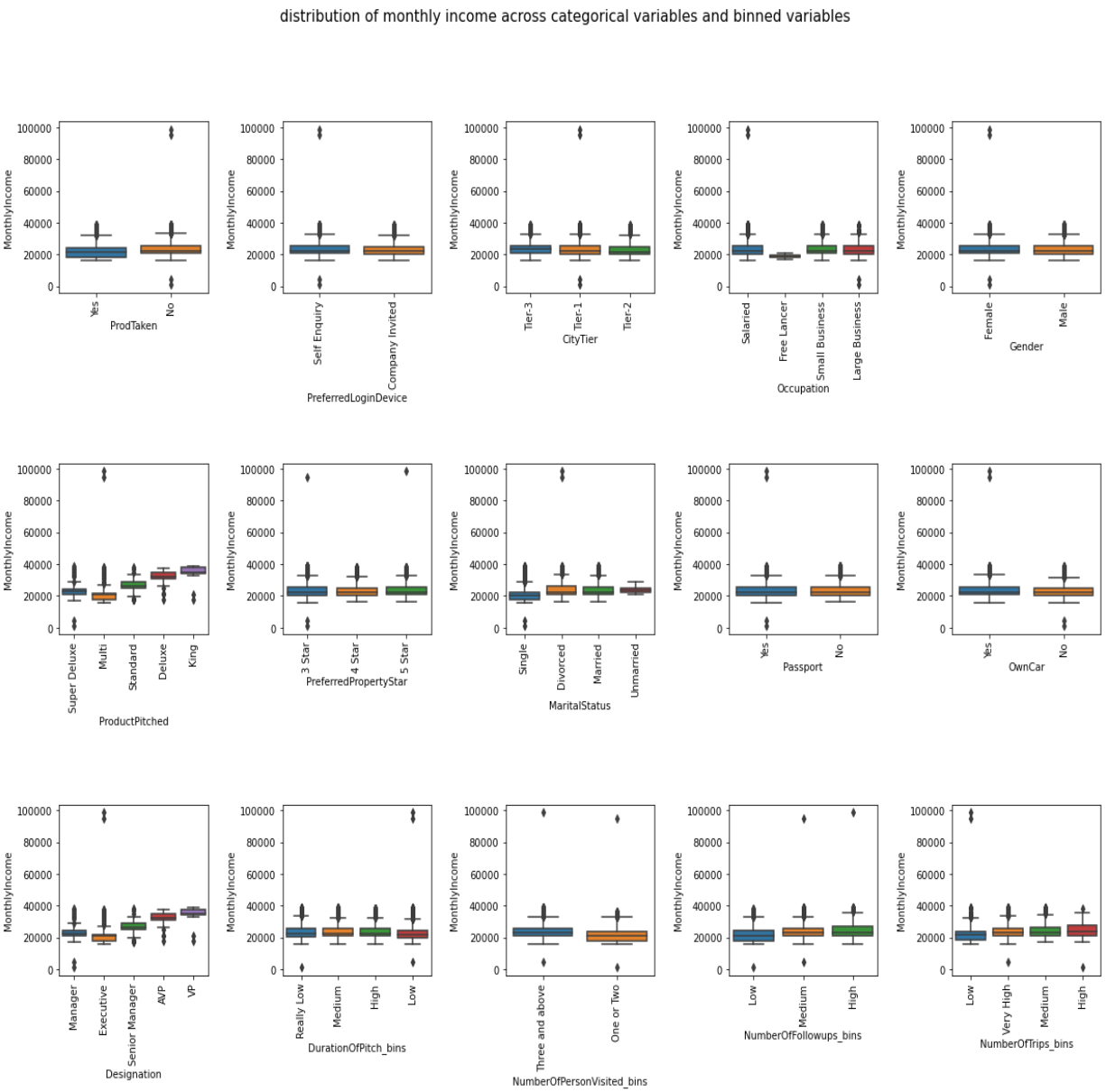

Fig: 10

Observations:

- Monthly income is higher of those customers who have taken prod uct Deluxe and King.
- Monthly income is higher of those customers who are working as AVP and VP.
- Monthly income is higher of those customers who are doing more no of trips or more travel.
- Monthly income is slightly lower of the customers who have taken product than the customers who have not taken.

Distribution of ProdTaken across all categorical and binned variables:

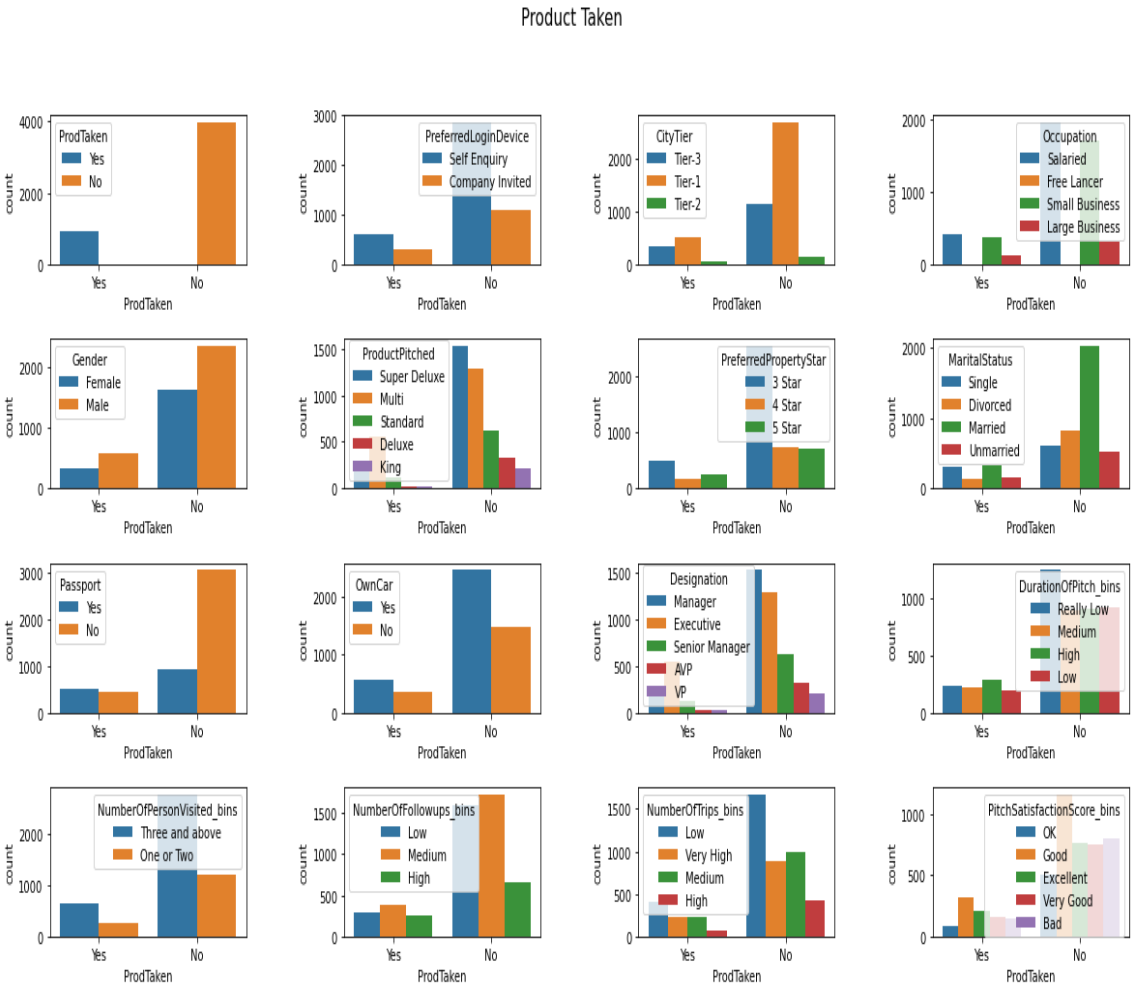

Fig: 11

Observations:

- Product taken is more by the customers who have passport, may be they can travel outside of the country. The customers who are taken product and do not have passport, are local traveller.
- Product taken is more by those customer who has own car
- Product taken is more by those customers who are working as exec utive.
- Product taken is more by the customers who live in Tier-1 city(Met ropolitan city)
- Product taken is more by the customers who is visiting with three a nd above three people.
- The customers who have taken product, their DurationOfPitch_bins is high.
- The product taken is more by the customers who stay in 3-Star hote l.

Distribution of Age and ProdTaken across different categorical variables:

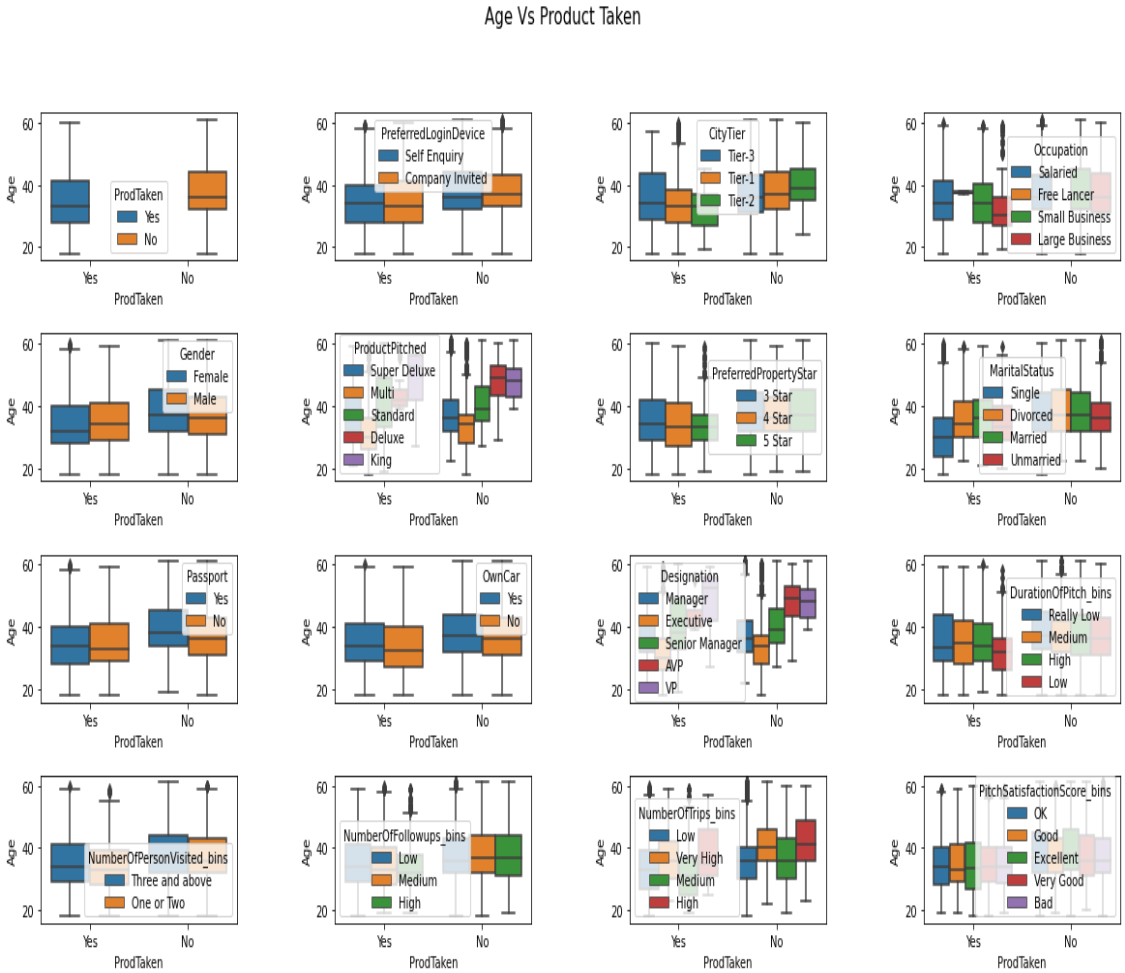

Fig: 12

Observations:

- The customers who are single and belong to age rage 50 to 60 have high income. They can spend money to buy Deluxe and King type of product.
- The product taken by the customers is more who preferred to stay i n 3 star hotel and their average age is 35 to 36.
- The most of the product is taken by younger and middle age group of customers who belong to small occupation.
- They are 2 Free Lancer and they are taken product also. They will definitely sell their product to customers.
- The most of the product is taken by those middle age group custom er whose travelling is more.

- The product taken is more of the customers who are younger and h ave passport.
- The product taken is more, of the customers who belong to middle age group and married.

In next step, we are going to do variable cluster analysis. In which, we will do feature creation also. This is also a part of feature engineering.

.

Cluster Analysis: See Appendix C

For cluster analysis I have used k-means algorithm and optimal value of k=5 for this case.

Univariate analysis for all clusters across numerical variable:

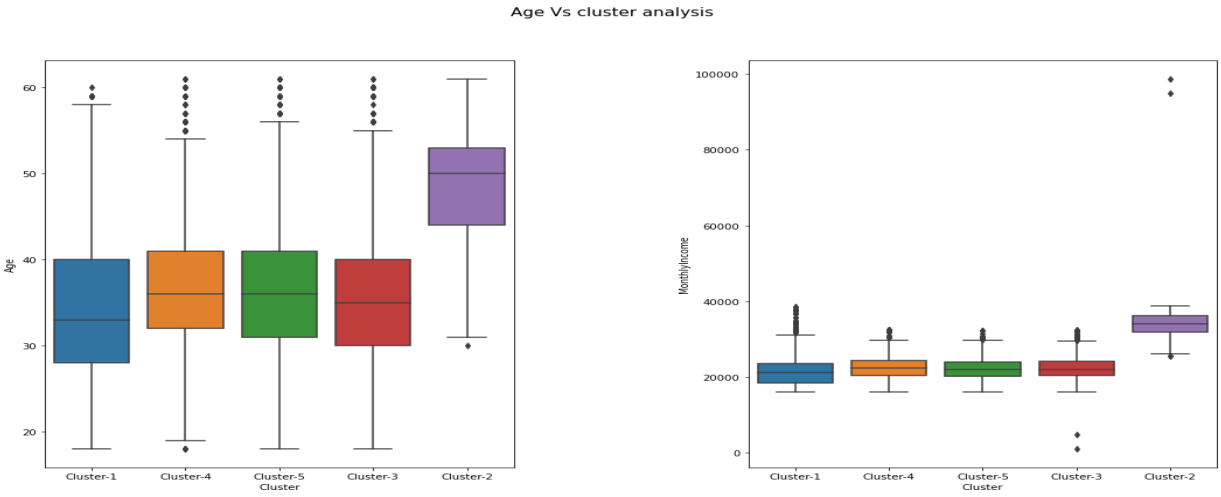

Fig: 13

Observations:

- Cluster-2 is the group of those customers who belong to age 50+(old er age group people)
- Cluster-4 is the group of those customers who belong to age group 3 6 to 40(middle people)
- Cluster-1 is the group of younger to middle age group of customers.
- Cluster-5 is group of younger and middle age group of customers.
- Cluster-2 is the group of those customers whose monthly income is high.

- Cluster-1, Cluster-3, Cluster-4 and cluster-5 are group of those customers whose monthly income is in range of 21000-23000.

.

Univariate analysis of clusters across all categorical variable:

See appendix D

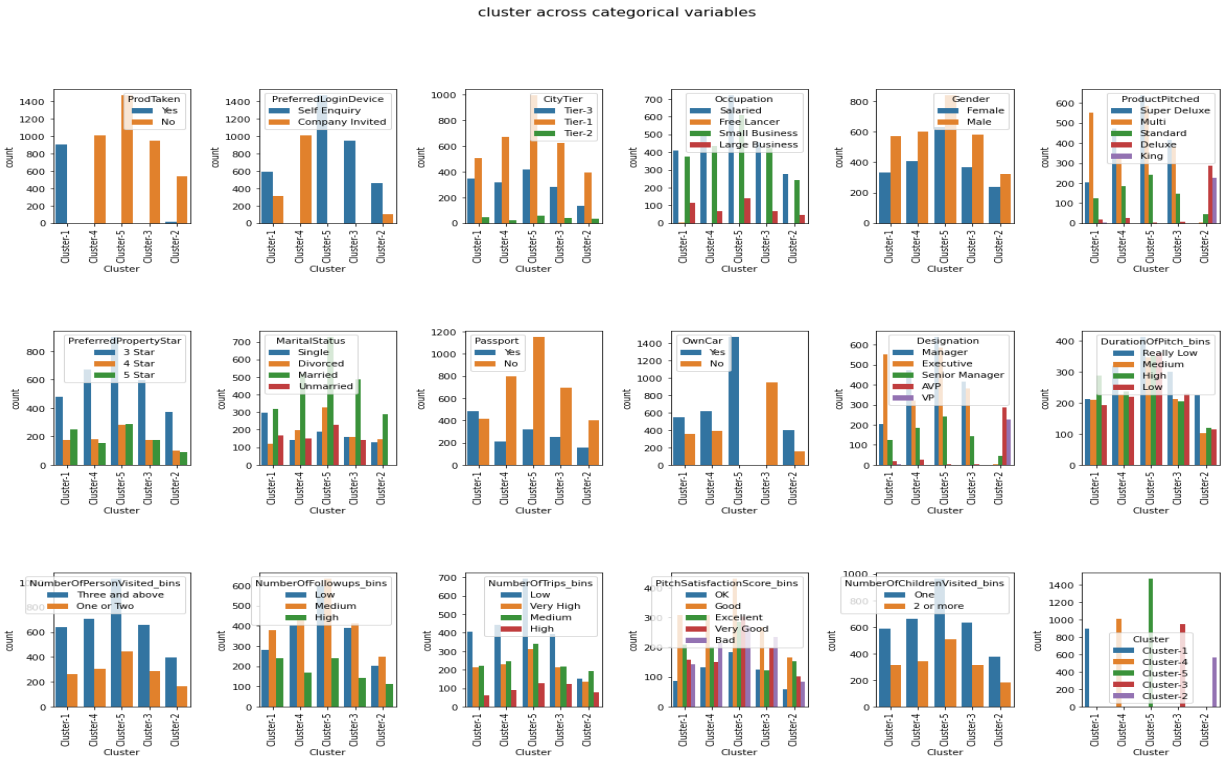

Fig: 14

Observations:

- Cluster-1 is group of those customers who have taken product .
- Maximum numbers of customers who have passport are in Cluster-1.
- Maximum numbers of customers, who are working as executive, belong to Cluster-1.
- Maximum number of customers belong to Tier-1(metropolitan city) are in Cluster-1.
- The product that are pitched maximum number of times, are multi(cheaper product)
- Most of the customers have travelled very less number of trips in Cluster-1.
- Most of the customers come by themselves in Cluster-1.
- Very few customers are taken product in Cluster-2.

- Very few customers have passport that belong to Cluster-2.

Conclusion:

- Cluster-1 is the group of younger people who have passport. So , their propensity of travel will be more.
- Most of the customers in Cluster-1 are working as executive. So their monthly income will be low, most probably they will buy cheaper product like Multi or Super Deluxe.
- Cluser-2 is the group of older people and very few people have passport. Hence, their propensity of buying product is very low even though all they have high monthly income.
- Cluster-4, Cluster-3 and Cluster-5 are the group of younger and middle age group and some of them have passport and most of t hem are working as manager. So their monthly income is low. Hence, propensity of buying product is very- very low.

Checking Outliers:

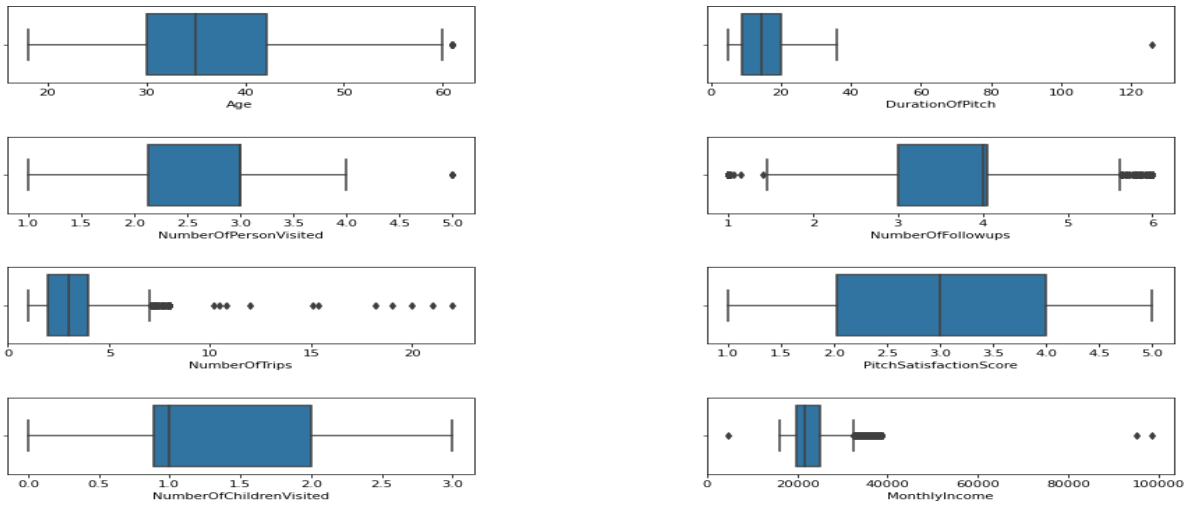

Fig.15

There are some outliers. We need to treat it.

Treating Outliers by Winsorization:

Winsorization, or winsorizing, is the process of transforming the data by limiting the extreme values, that is, the outliers, to a certain arbitrary value, closer to the mean of the distribution. Winsorizing is different from trimming because the extreme values are not removed, but are instead replaced by other values. A typical strategy involves setting outliers to a specified percentile.

In this case we used 95% winsorization, we set all data below the 5th percentile to the value at the 5th percentile and all data above the 95th percentile to the value at the 95th percentile.The purpose of using winsorization to avoid imposing bias into the data.

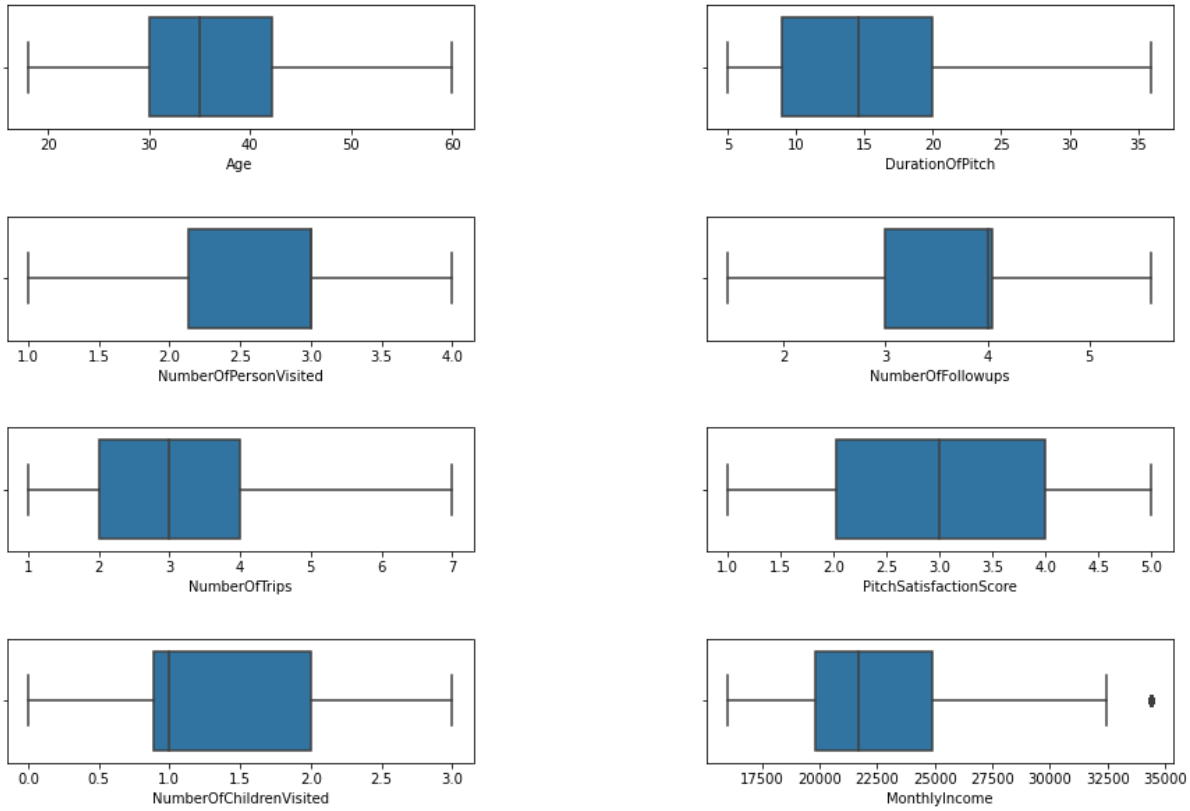

Fig: 16

Almost all outliers have removed from all features except MonthlyIncome.

Model Building and Interpretation and Model Validation:

Let's check a data info() for df_tourism2.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4888 entries, 0 to 4887
Data columns (total 19 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ProdTaken                4888 non-null   object
 1   PreferredLoginDevice     4888 non-null   object
 2   CityTier                 4888 non-null   object
 3   Occupation               4888 non-null   object
 4   Gender                   4888 non-null   object
 5   ProductPitched           4888 non-null   object
 6   PreferredPropertyStar    4888 non-null   object
```

```
 7   MaritalStatus             4888 non-null   object
 8   Passport                  4888 non-null   object
 9   OwnCar                    4888 non-null   object
 10  Designation               4888 non-null   object
 11  Age                       4888 non-null   float64
 12  DurationOfPitch           4888 non-null   float64
 13  NumberOfPersonVisited     4888 non-null   float64
 14  NumberOfFollowups         4888 non-null   float64
 15  NumberOfTrips             4888 non-null   float64
 16  PitchSatisfactionScore    4888 non-null   float64
 17  NumberOfChildrenVisited   4888 non-null   float64
 18  MonthlyIncome             4888 non-null   float64
dtypes: float64(8), object(11)
memory usage: 725.7+ KB
```

There are some categorical and numerical variable as well. ProdTaken is target variable. For model building all feature should be in numerical nature, so first we need to convert all categorical variable into numerical variable.

Check the imbalance level in target variable.

```
No     0.811784
Yes    0.188216
Name: ProdTaken, dtype: float64
```

We can see from the above output there is a good amount of class imbalance in t he data w.r.t the target variable i.e. ProdTaken. To take care of this imbalance w e will have to apply **SMOTE.** Before applying SMOTE we will split the data in to training and testing sets to avoid introducing bias in the test data set.

We can do model building without SMOTE also because data is not highly imbalance.

But even before that we need to convert all the categorical variables into numeri cal form so that it is conducive to modelling.

There are two types of categorical variable in the data set wherein some are or o rdinal like ProductPitched,PreferredPropertyStar, Designation which is ranked b ased and rest of all are categorical where weightage are equal for all different la bel . For ordinal categorical variable we will use map and lambda function or Categorical().code and for other categorical variable we will use one hot encodi ng and or dummy variable creation.

Let's check the info() of df_tourism2_dummified.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4888 entries, 0 to 4887
Data columns (total 24 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Age                               4888 non-null   float64
```

```
 1   DurationOfPitch                    4888 non-null   float64
 2   NumberOfPersonVisited              4888 non-null   float64
 3   NumberOfFollowups                  4888 non-null   float64
 4   NumberOfTrips                      4888 non-null   float64
 5   PitchSatisfactionScore             4888 non-null   float64
 6   NumberOfChildrenVisited            4888 non-null   float64
 7   MonthlyIncome                      4888 non-null   float64
 8   ProductPitched_codes               4888 non-null   int64
 9   PreferredPropertyStar_codes        4888 non-null   int64
 10  Designation_codes                  4888 non-null   int64
 11  ProdTaken_Yes                      4888 non-null   uint8
 12  PreferredLoginDevice_Self Enquiry  4888 non-null   uint8
 13  CityTier_Tier-2                    4888 non-null   uint8
 14  CityTier_Tier-3                    4888 non-null   uint8
 15  Occupation_Large Business          4888 non-null   uint8
 16  Occupation_Salaried                4888 non-null   uint8
 17  Occupation_Small Business          4888 non-null   uint8
 18  Gender_Male                        4888 non-null   uint8
 19  MaritalStatus_Married              4888 non-null   uint8
 20  MaritalStatus_Single               4888 non-null   uint8
 21  MaritalStatus_Unmarried            4888 non-null   uint8
 22  Passport_Yes                       4888 non-null   uint8
 23  OwnCar_Yes                         4888 non-null   uint8
dtypes: float64(8), int64(3), uint8(13)
memory usage: 482.2 KB
```

Now all features are numeric and data is ready for modelling.

A very first step of model building is that we have to divide the data into predictor (independent set) and target (dependent set) set. After that we have to split the data into train and test set. In next step, we will apply SMOTE on train set to fix the problem of imbalance problem. Here comes some basic steps that we have to follow at early stage of model building.

1. Dividing the data set into predictor and target variable

2. Splitting the data into (70%-30% ratio) train and test set:

3. Applying SMOTE on training data.

Checking the propensity of target variables after SMOTE:

```
ProdTaken_Yes
1               0.5
0               0.5
dtype: float64
```

We can see from above that the proportion of the minority class i.e. ***ProdTaken_Yes*** has been increased from approximately 19% to 50%.

Next, we are going to create base line model. The purpose of building this baseline model to get minimum level of performance from the data and to get some interpretation.

Baseline Logistic regression Model:

Here, our purpose is to figure out what all are the features significant and minimum baseline accuracy.

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 8
```

	Coef	Pvalue
Age	-0.005357	2.469125e-01
DurationOfPitch	0.039303	1.692078e-16
NumberOfPersonVisited	0.086194	2.199815e-01
NumberOfFollowups	0.546081	8.423108e-30
NumberOfTrips	0.058873	1.295002e-02
PitchSatisfactionScore	0.172784	1.139421e-09
NumberOfChildrenVisited	-0.273611	1.897338e-06
MonthlyIncome	0.000126	8.395948e-16
ProductPitched_codes	-0.295464	1.897178e-15
PreferredPropertyStar_codes	0.323875	8.803698e-12
Designation_codes	-0.799169	3.914291e-25
PreferredLoginDevice_Self Enquiry	-0.632843	4.300468e-16
CityTier_Tier-2	-0.128176	5.499689e-01
CityTier_Tier-3	0.501573	4.687505e-08
Occupation_Large Business	-3.998666	8.323858e-72
Occupation_Salaried	-4.210610	6.813429e-108
Occupation_Small Business	-4.128698	1.274674e-101
Gender_Male	-0.105839	1.573813e-01
MaritalStatus_Married	-1.451994	4.931178e-53
MaritalStatus_Single	0.028861	7.797252e-01
MaritalStatus_Unmarried	-0.615253	2.630744e-07

	Coef	Pvalue
Passport_Yes	1.144677	9.651458e-46
OwnCar_Yes	-0.534470	1.460727e-12

So, to figure out significant feature we have to do hypothesis testing here.

Null Hypothesis: Feature has significant impact on dependent variable.

Alternate Hypothesis: Feature has not significant impact on dependent variable.

If p-vale > 0.05, then we have to reject null hypothesis and If p_value < 0.05, then we fail to reject null hypothesis at 95% confidence interval.

Now, we are going to **filter out those features whose p-value > 0.05**.These are the non- significant features. Later on we will drop these features while model building.

	Coef	Pvalue
Age	-0.005357	0.246912
NumberOfPersonVisited	0.086194	0.219982
CityTier_Tier-2	-0.128176	0.549969
Gender_Male	-0.105839	0.157381
MaritalStatus_Single	0.028861	0.779725

Observations:

From above output we can conclude that Age, NumberOfPersonVisited, CityTier_Tier_2, Gender_Male and MaritalStatus_Single are non-significant features. But as we have been seen in our EDA part, Age has significant impact on ProdTaken(dependent variable).

So we will consider Age variable also in model building and will leave the **non-Signinficant feature NumberOfPersonVisited, CityTier_Tier_2, Gender_Male and MaritalStatus_Single.**

Building a Logistic regression model considering based on significant variable and calculating coefficients and P-value:

Logistic regression **is** the appropriate regression analysis to conduct when the dependent variable **is** dichotomous (binary).Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 8
```

	Coef	Pvalue
Age	-0.005428	2.404370e-01
DurationOfPitch	0.039259	1.780121e-16
NumberOfPersonVisited	0.088844	2.053901e-01
NumberOfFollowups	0.542924	9.091607e-30
NumberOfTrips	0.058852	1.298536e-02
PitchSatisfactionScore	0.172472	1.098447e-09
NumberOfChildrenVisited	-0.275016	1.573669e-06
MonthlyIncome	0.000125	9.999423e-16
ProductPitched_codes	-0.297130	1.184042e-15
PreferredPropertyStar_codes	0.322188	1.090916e-11
Designation_codes	-0.793425	6.443302e-25

	Coef	Pvalue
PreferredLoginDevice_Self Enquiry	-0.641294	1.400271e-16
CityTier_Tier-3	0.508851	2.447838e-08
Occupation_Large Business	-4.032260	2.473993e-76
Occupation_Salaried	-4.237081	9.327497e-114
Occupation_Small Business	-4.155929	6.263443e-107
MaritalStatus_Married	-1.472798	2.462015e-66
MaritalStatus_Unmarried	-0.625723	2.837693e-08
Passport_Yes	1.149069	3.429055e-46
OwnCar_Yes	-0.534483	1.423924e-12

Observations:

Now, all the above features have P_value<0.05.Hence, we can conclude these features has significant impact on dependent variable.

Interpreting the coefficient, odds and probability:

	Coef	Pvalue	Odds	Prob
Age	-0.005428	2.404370e-01	0.994587	0.498643
DurationOfPitch	0.039259	1.780121e-16	1.040039	0.509813
NumberOfPersonVisited	0.088844	2.053901e-01	1.092910	0.522196
NumberOfFollowups	0.542924	9.091607e-30	1.721032	0.632492
NumberOfTrips	0.058852	1.298536e-02	1.060619	0.514709

	Coef	Pvalue	Odds	Prob
PitchSatisfactionScore	0.172472	1.098447e-09	1.188239	0.543011
NumberOfChildrenVisited	-0.275016	1.573669e-06	0.759560	0.431676
MonthlyIncome	0.000125	9.999423e-16	1.000125	0.500031
ProductPitched_codes	-0.297130	1.184042e-15	0.742947	0.426259
PreferredPropertyStar_codes	0.322188	1.090916e-11	1.380144	0.579857
Designation_codes	-0.793425	6.443302e-25	0.452293	0.311434
PreferredLoginDevice_Self Enquiry	-0.641294	1.400271e-16	0.526611	0.344954
CityTier_Tier-3	0.508851	2.447838e-08	1.663379	0.624537
Occupation_Large Business	-4.032260	2.473993e-76	0.017734	0.017425
Occupation_Salaried	-4.237081	9.327497e-114	0.014450	0.014244
Occupation_Small Business	-4.155929	6.263443e-107	0.015671	0.015429
MaritalStatus_Married	-1.472798	2.462015e-66	0.229283	0.186518
MaritalStatus_Unmarried	-0.625723	2.837693e-08	0.534874	0.348481
Passport_Yes	1.149069	3.429055e-46	3.155254	0.759341
OwnCar_Yes	-0.534483	1.423924e-12	0.585972	0.369472

Observations:

- A customer having passport, increases the probability of taken prod by 76%.
- If the customer age increases, the probability of taken product decreases by 50%.
- A customer belongs to Tier-3, increases the probability of prod taken increases by 61%.

- If the monthly income of customer increases, the probability of prod taken increases by 50%.
- If the number of trips increases, probability of taking prod decreases by 51%.
- If the customer having small business and salaried, the probability of taking product decreases by 1%.
- A person having own car, decreases the probability of product taken by 36%.
- A customer, who has married, decreases the probability of product taken by 19%.
- A customer, whose designation is high (work as VP,AVP),decreases the probability of product taken by 31%.

Looking at baseline accuracy score:

```
Training Accuracy of Logistic Model: 0.809
Test Accuracy of Logistic Model: 0.773
```

Now, I have stored smote train independent and dependent set into new variable X_train and Y_train as per my convenience.

X_train = os_data_X

Y_train = os_data_Y

Let's build different models. ***All models are built with hyper parameter tuning. Hyper parameter tuning is the powerful tool to enhance supervised learning model- improving accuracy, recall and other important metrics by searching the optimal parameter based on different scoring methods. For hyper parameter tuning we have to use GridSearchCV() that are available in sklearn.model_selection.***

Model1: Logistic Regression:

1. These are the hyper parameters,that has been used for model tuning.

```
grid = {'solver':['newton-cg','lbfgs','liblinear','sag','saga'],
    'max_iter':[1000,2000],
     'n_jobs':[2,3]
    }
```

2. After model tuning, we got these parameters are as a best parameter for model.

```
{'max_iter': 1000, 'n_jobs': 2, 'solver': 'liblinear'}
```

3. Predicting the probability and probability class for train set.

4. Calculating model performance matrices for train and test set.

Accuracy Score:

For train set:

```
0.8131294964028777
```

For test set:

```
0.7757327880027266
```

Confusion Matrix:

For train set:

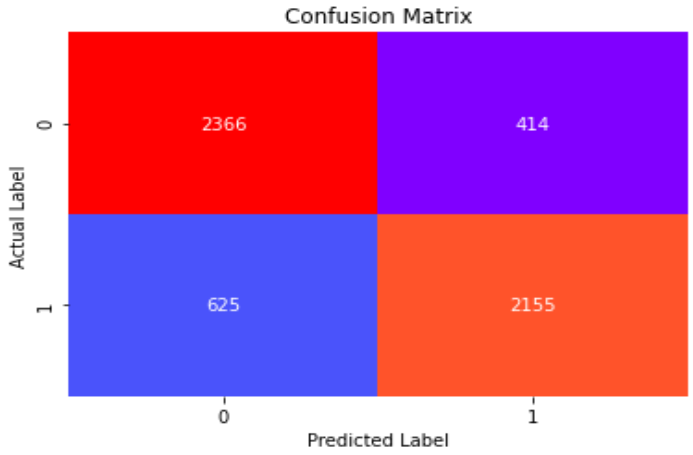

Fig.17

Classification Report:

	precision	recall	f1-score	support
0	0.85	0.79	0.82	2991
1	0.78	0.84	0.81	2569
accuracy			0.81	5560
macro avg	0.81	0.81	0.81	5560
weighted avg	0.82	0.81	0.81	5560

For test set:

Confusion matrix:

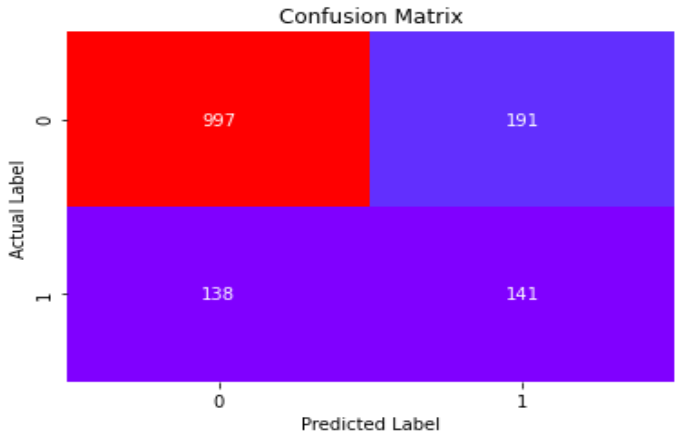

Fig.18

- Total no of correct prediction=141+997
- Total no of incorrect prediction=191+138

Classification Report:

	precision	recall	f1-score	support
0	0.84	0.88	0.86	1135
1	0.51	0.42	0.46	332
accuracy			0.78	1467
macro avg	0.67	0.65	0.66	1467
weighted avg	0.76	0.78	0.77	1467

- 42 % customers are correctly identified as the customers who have taken product.

AUC and ROC Curve:

For train set:

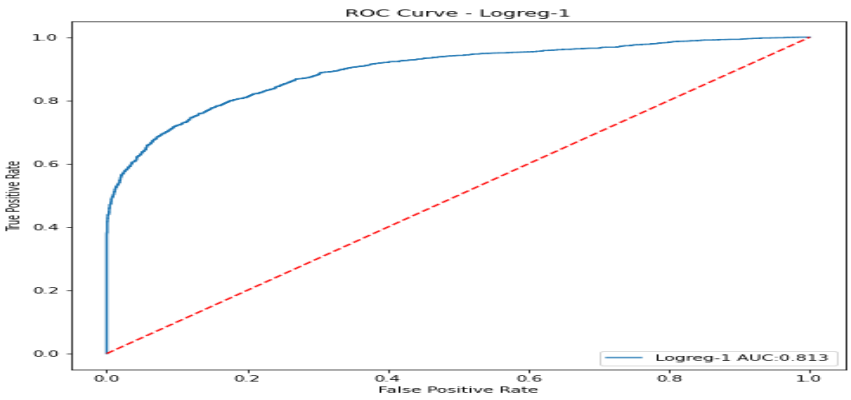

Fig.19

For test set:

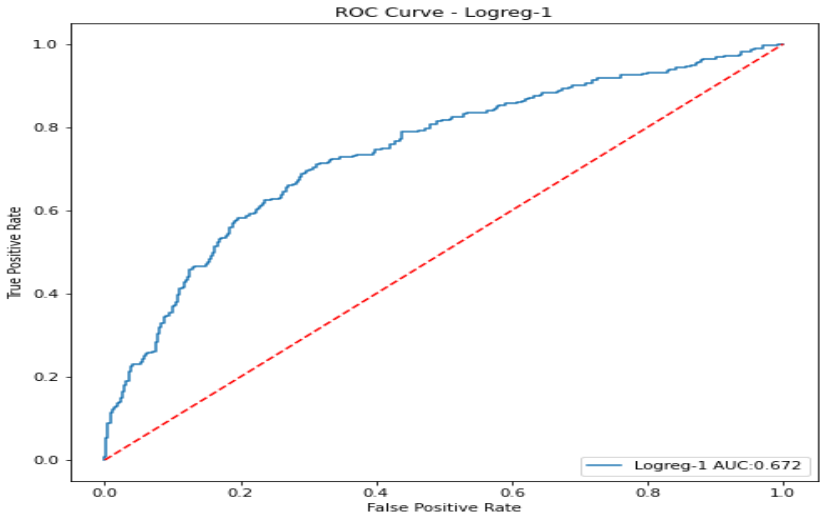

Fig.19

- For test set AUC score is less than train set. Model performs over fitting.

Model2: Logistic regression model with Recursive Feature Elimination:

RFE (Recursive Feature Elimination): RFE is feature selection algorithm that selecting those features (columns) in training data set that are more or most relevant in predicting target variable.

1. First, we have to pass total no. of parameters that has significant impact on dependent variable and fit the model for train set.
2. After filtering out significant feature, we will divide the data into train and test set for independent set. Next, we will build logistic regression model for X_train_FS and X_test_FS.
3. Predicting the probability and probability for train and test set.
4. Calculating performance metrics for train and test set.

Accuracy score:

For train set

```
0.810611510791367
```

For test set:

```
0.7716428084526245
```

Confusion Matrix:

For train set:

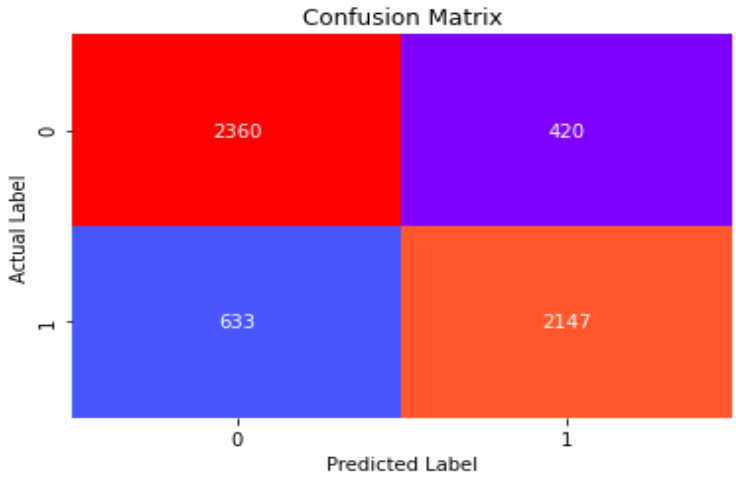

Fig.20

For test set:

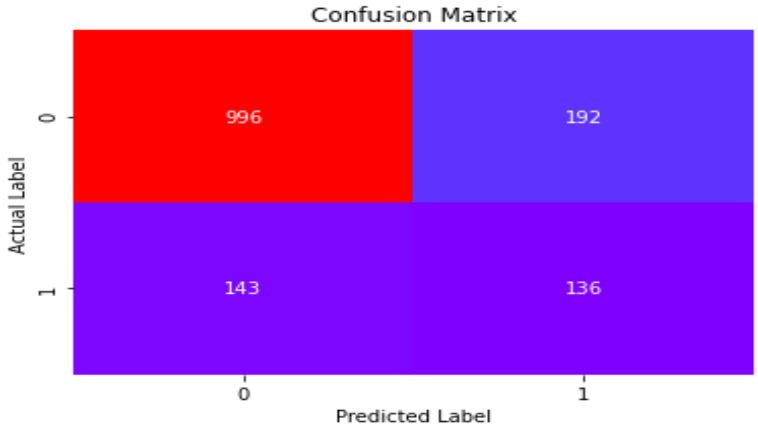

Fig.21

- Total no of correct prediction=136+996
- Total no of incorrect prediction=192+143

Classification Report:

For train set:

	precision	recall	f1-score	support
0	0.79	0.85	0.82	2780
1	0.84	0.77	0.80	2780
accuracy			0.81	5560
macro avg	0.81	0.81	0.81	5560
weighted avg	0.81	0.81	0.81	5560

For test set:

	precision	recall	f1-score	support
0	0.87	0.84	0.86	1188
1	0.41	0.49	0.45	279
accuracy			0.77	1467
macro avg	0.64	0.66	0.65	1467
weighted avg	0.79	0.77	0.78	1467

- 49% of customers are correctly identified as those customers who have taken product.

AUC and ROC-curve:

For train set:

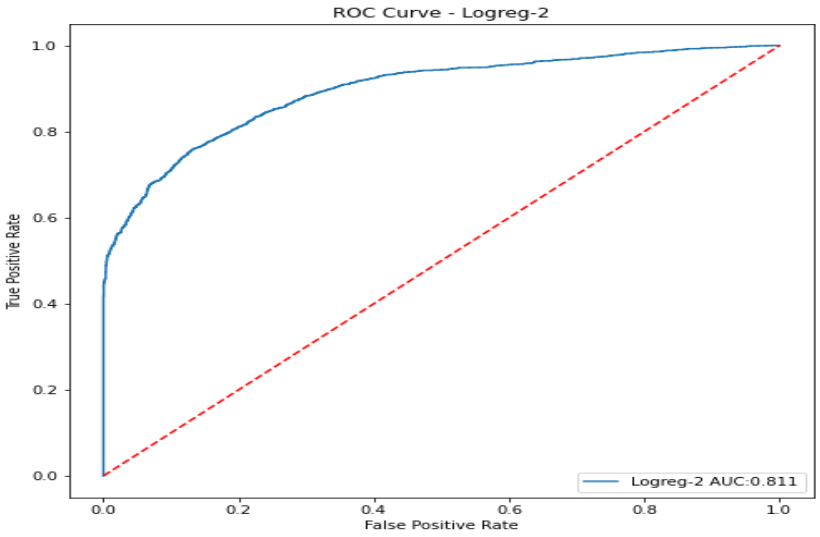

Fig.22

For test set:

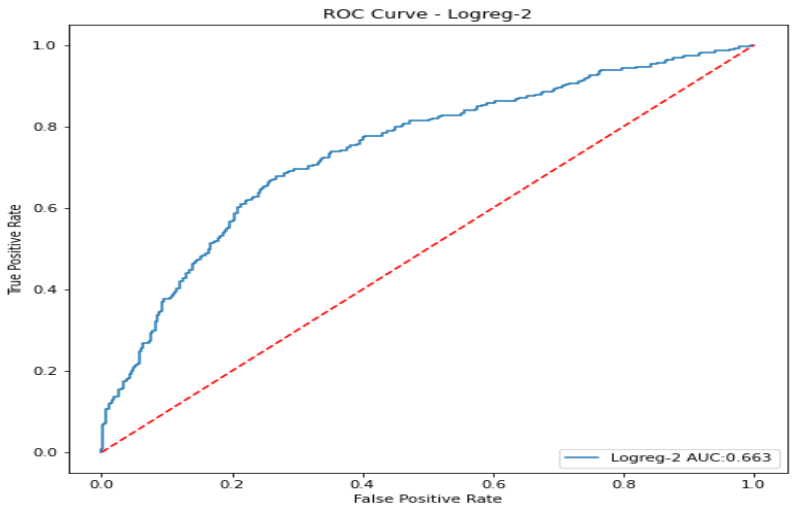

Fig.23

- Poor performance on test set: over fitting problem.

Model 3: Decision Tree with hyper parameter tuning:

1. These are the hyper parameters that are used for model tunning.

```
parameters = {'criterion':['gini','entropy'], 'max_depth':[2,5,10,15],
    'min_samples_split':[2,10,15,20,25,30,60,80,100],
    'min_samples_leaf':[1,7,10,15,20,33],
     'min_impurity_decrease':[0.0001,0.001]}
```

2. After that we have to build logistic regression model and we have to pass the parameter grid into GridSearchCV. Next, we have to do model fitting.

```
grid = GridSearchCV(DT1,param_grid = parameters,cv=10,verbose=1,n_jobs=-1)
DT1=grid.fit(X_train,Y_train.values.ravel())
```

3. Next, we can figure out best parameter for model.
4. Predicting probabilities and probability classes for train and test set.
5. Calculating performance matrices.

Accuracy score:

For train set:

```
0.9820863309352518
```

For test set:

```
0.8336741649625086
```

Confusion Matrix:

For train set:

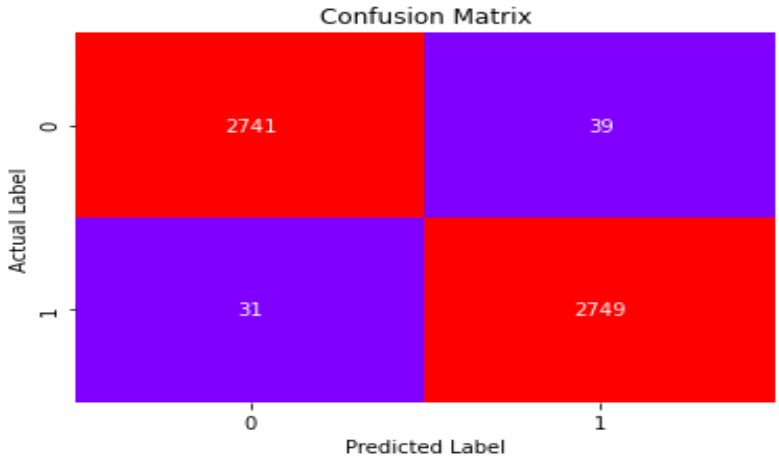

Fig.24

For test set:

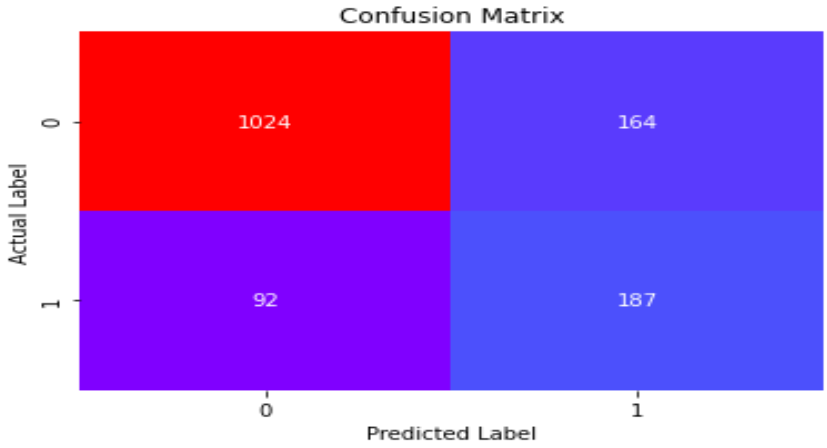

Fig.25

- Total no of correct prediction=187+1024
- Total no of incorrect prediction=164+92

Classification Report:

For train set:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2780
1	0.99	0.99	0.99	2780
accuracy			0.99	5560

macro avg	0.99	0.99	0.99	5560
weighted avg	0.99	0.99	0.99	5560

For test set:

	precision	recall	f1-score	support
0	0.92	0.86	0.89	1188
1	0.53	0.67	0.59	279
accuracy			0.83	1467
macro avg	0.73	0.77	0.74	1467
weighted avg	0.84	0.83	0.83	1467

- 67 % customers are correctly identified as those customers who have Taken product.

AUC and ROC Curve:

For train set:

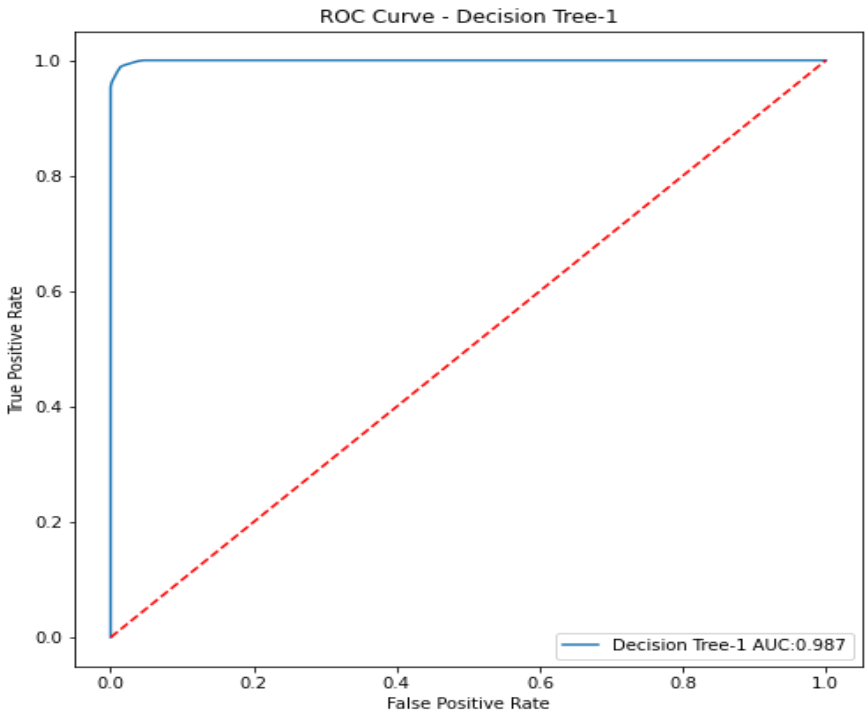

Fig.26

For test set:

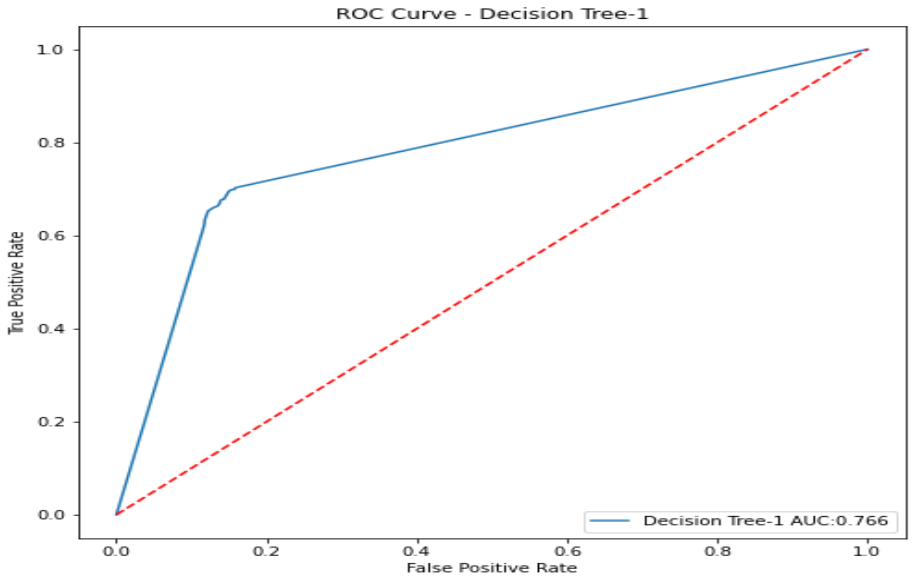

Fig.27

- AUC score is poor on test set: over fitting problem.

Model4: Random Forest with hyper parameter tuning:

1. These are the hyper parameters that are used for model tuning.

```
parameters = {'n_estimators':[60,80,100,200,300],

              'min_samples_split':[1,2,3],

              'min_samples_leaf':[1,5,10],

              'max_features': [2,3,4],

              'min_impurity_decrease':[0.00001,0.0001,0.001]}
```

2. After model fitting, we figured out the below parameters are as best parameters for model.

```
{'max_features': 4,
 'min_impurity_decrease': 1e-05,
 'min_samples_leaf': 1,'min_samples_split': 2,
 'n_estimators': 300}
```

3. Predicting probabilities and probability classes for train and test set.

4. Calculating performance matrices.

Accuracy score:

For train set:

```
1.0
```

For test set:

```
0.8657123381049762
```

Confusion Matrix:

For train set:

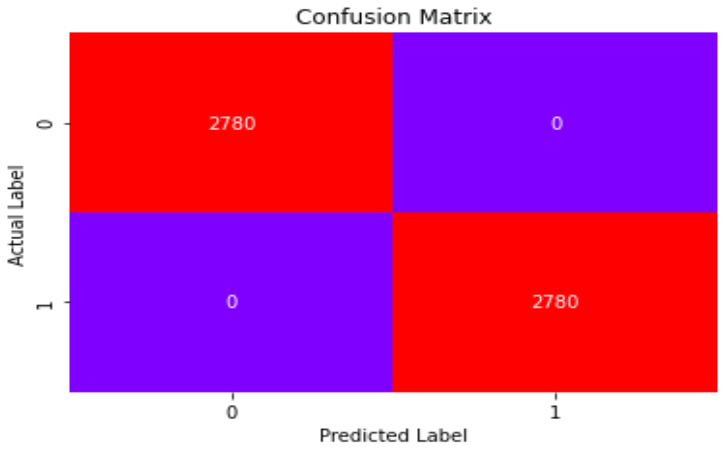

Fig.28

For test set:

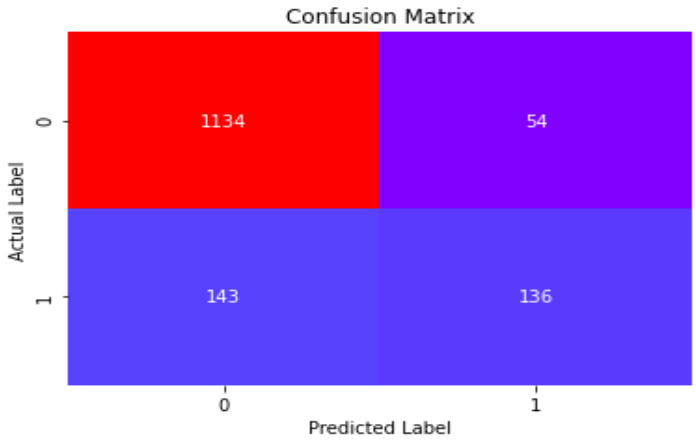

Fig.29

- Total no of correct prediction=136+1134
- Total no of incorrect prediction=143+54

Classification Report:

For train set:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2780
1	1.00	1.00	1.00	2780
accuracy			1.00	5560
macro avg	1.00	1.00	1.00	5560
weighted avg	1.00	1.00	1.00	5560

For test set:

	precision	recall	f1-score	support
0	0.89	0.95	0.92	1188
1	0.70	0.49	0.58	279
accuracy			0.87	1467
macro avg	0.79	0.72	0.75	1467
weighted avg	0.85	0.86	0.85	1467

- 49 % of customers are correctly identified as the customers who have taken product.

AUC and ROC-Curve:

For train set:

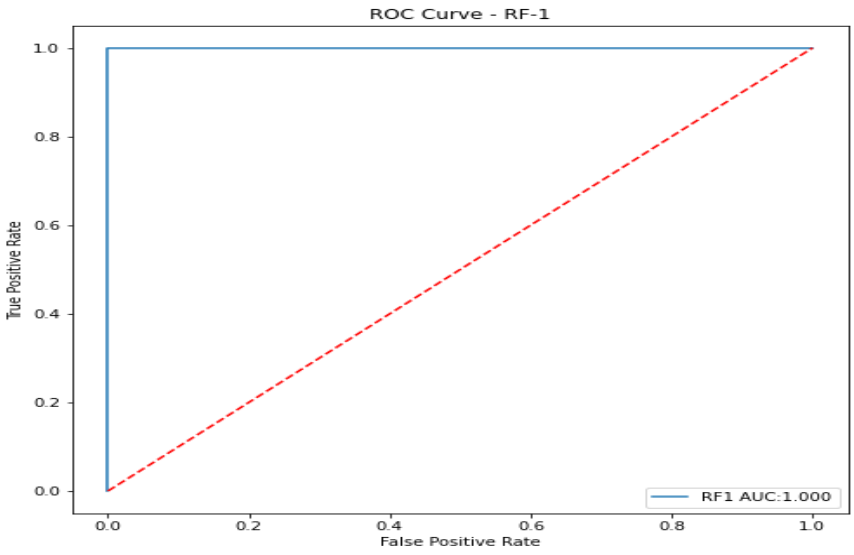

Fig. 30

For test set:

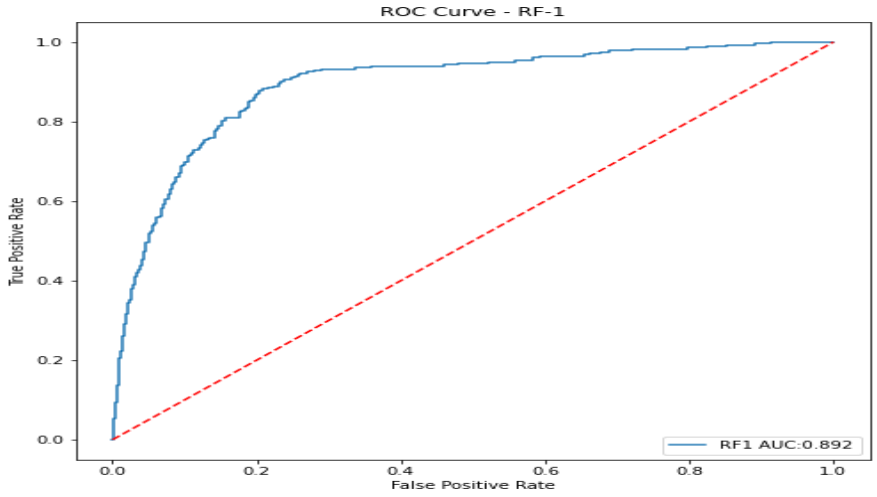

Fig.31

Looking at the Features Important:

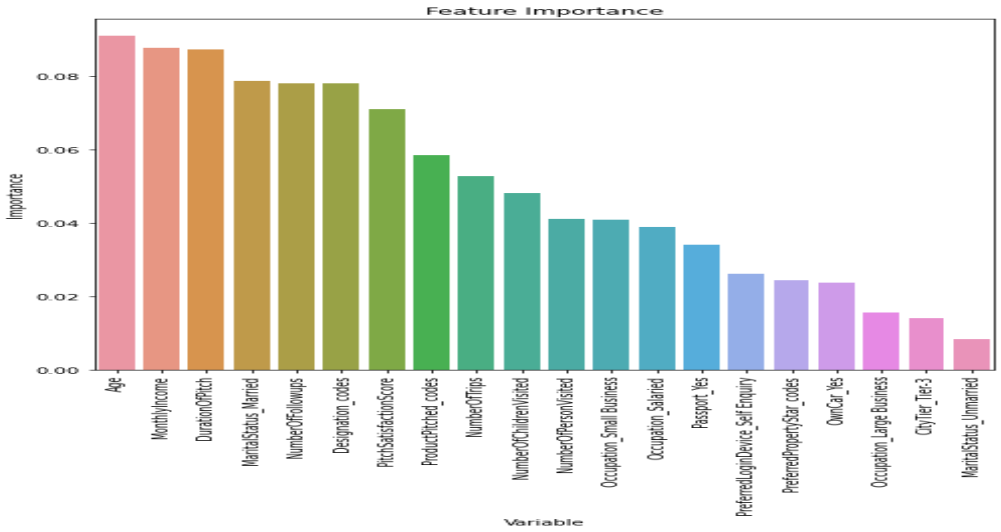

Fig.32

Observations:

Features which have longer bar, are most significant features. Age, MonthlyIncome,DurationOfpitch,Marital_status_married,NumberOfFollowups, Designation_code has significant impact on ProdTaken(dependent variable).The most important features among all is Age and MonthlyIncome.

Model5: Gradient Boosting Model:

1. These are the hyper parameters that has been used for model tuning.

```
params = {'loss':['deviance','exponential'],

    'learning_rate':[0.15,0.17,0.20],
```

'n_estimators':[300,500,700]}

2. After that, we have to fit the model for train set.

3. Next, we can figure out best parameter for model.

4. Predicting the probability and probability class for train set.

5. Calculating model performance matrices for train and test set.

Accuracy Score:

For train set:

```
0.9985611510791367
```

For test set:

```
0.896387184730743
```

Confusion Matrix:

For train set:

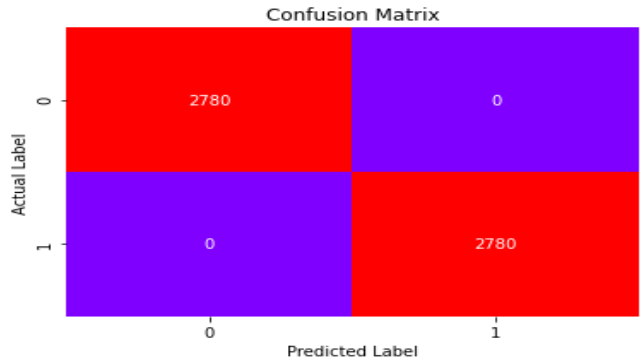

Fig.33

For test set:

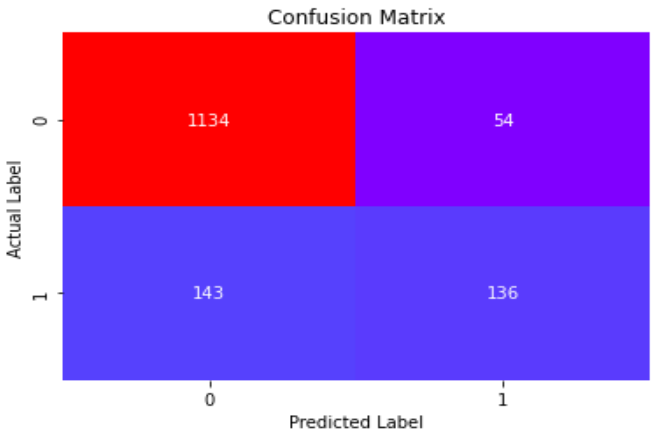

Fig.34

- Total no of correct prediction=136+1134
- Total no of incorrect prediction=143+54

Classification Report:

For train set:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2780
1	1.00	1.00	1.00	2780
accuracy			1.00	5560
macro avg	1.00	1.00	1.00	5560
weighted avg	1.00	1.00	1.00	5560

For test set:

	precision	recall	f1-score	support
0	0.89	0.95	0.92	1188
1	0.72	0.49	0.58	279
accuracy			0.87	1467
macro avg	0.80	0.72	0.75	1467
weighted avg	0.86	0.87	0.86	1467

- 49 % of customers are correctly identified as the customers who have tak en product.

AUC and ROC Curve:

For train set:

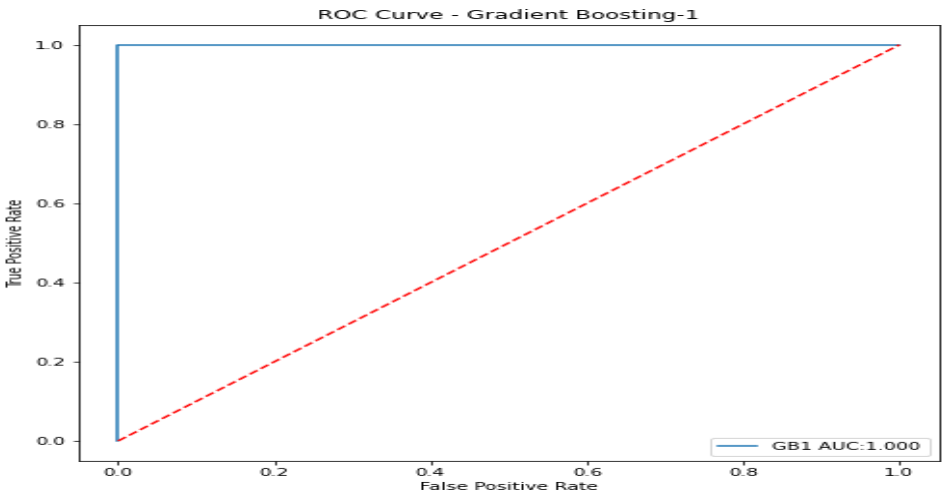

Fig.35

For test set:

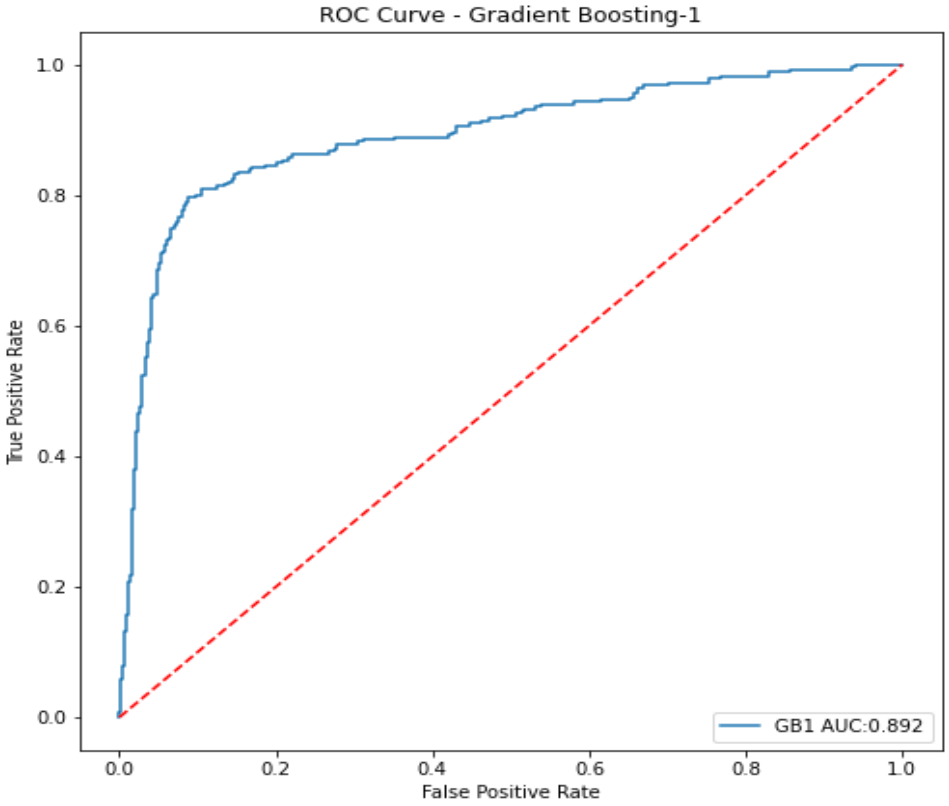

Fig.36

Looking at the features Importance:

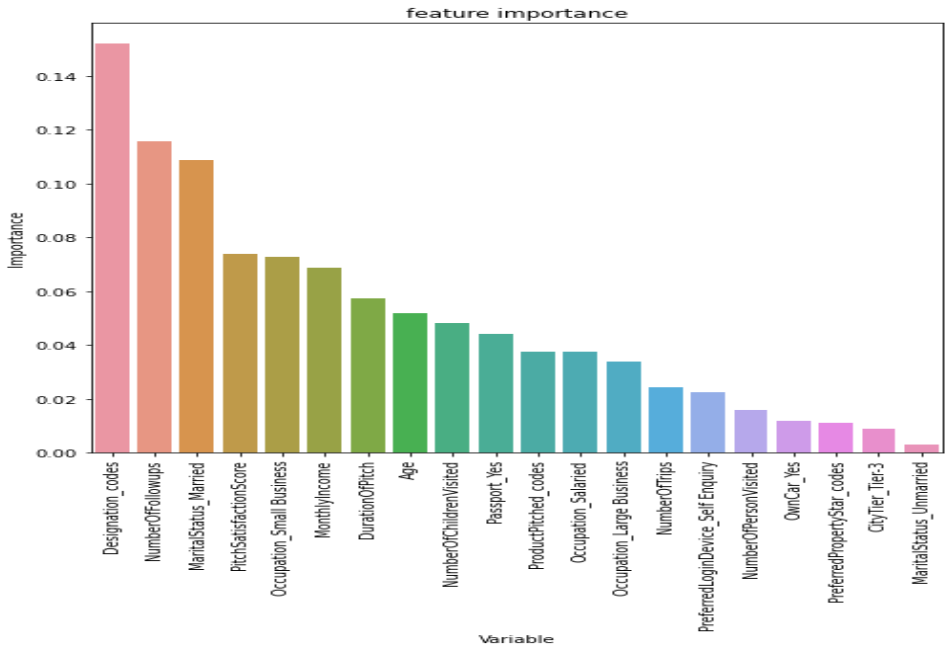

Fig.37

Observation:

In gradient boosting model the most significant feature is Designation_code followed by NumberOfFollowUps followed by Martial_Staues_Married.

Model6: KNN(K Nearest Neighbour) :

1. For building KNN model we have to scale train and test independent set.

2. The below is the hyper parameters that has been used for model tuning.

```
params = {'n_neighbors':range(2,11),
        'p':[2,3],
        'metric':['manhattan','chebyshev','minkowski']}
```

3. After that we have to do model fitting.

4. These are best parametrs that has been used for model tuning.

```
{'metric': 'manhattan', 'n_neighbors': 2, 'p': 2}
```

5. Predicting the probability and probability class for train set.

6. Calculating model performance matrices for train and test set.

Accuracy Score:

For train set:

```
0.9739208633093526
```

For test set:

```
0.8657123381049762
```

Confusion Matrix:

For train set:

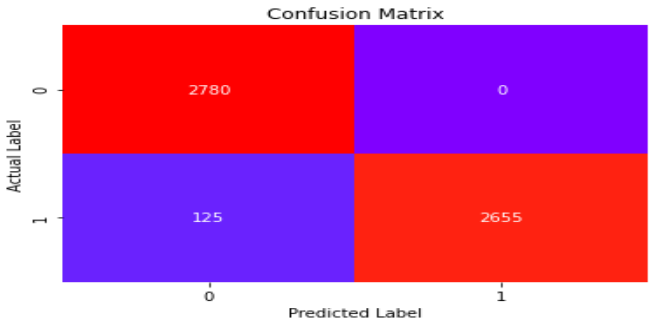

Fig: 38

For test set:

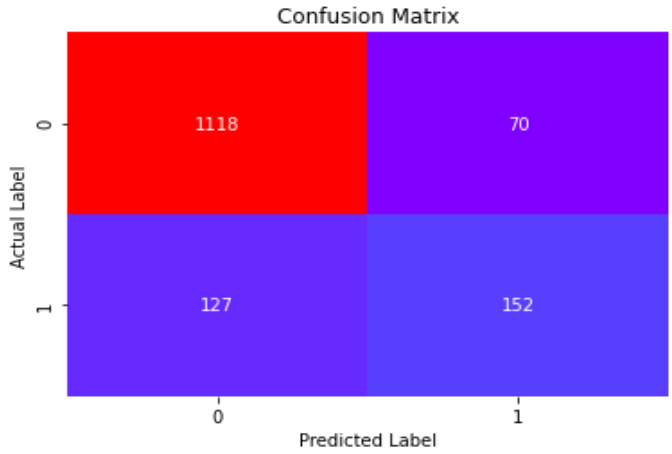

Fig.39

- Total no of correct prediction=201+1001
- Total no of correct prediction=78+187

Classification Report:

For train set:

	precision	recall	f1-score	support
0	0.98	0.97	0.97	2780
1	0.97	0.98	0.97	2780
accuracy			0.97	5560
macro avg	0.97	0.97	0.97	5560
weighted avg	0.97	0.97	0.97	5560

For test set:

	precision	recall	f1-score	support
0	0.90	0.94	0.92	1188
1	0.68	0.54	0.61	279
accuracy			0.87	1467
macro avg	0.79	0.74	0.76	1467
weighted avg	0.86	0.87	0.86	1467

- 54 % customers are correctly identified as the customers who have taken product.

AUC and ROC Curve:

For train set:

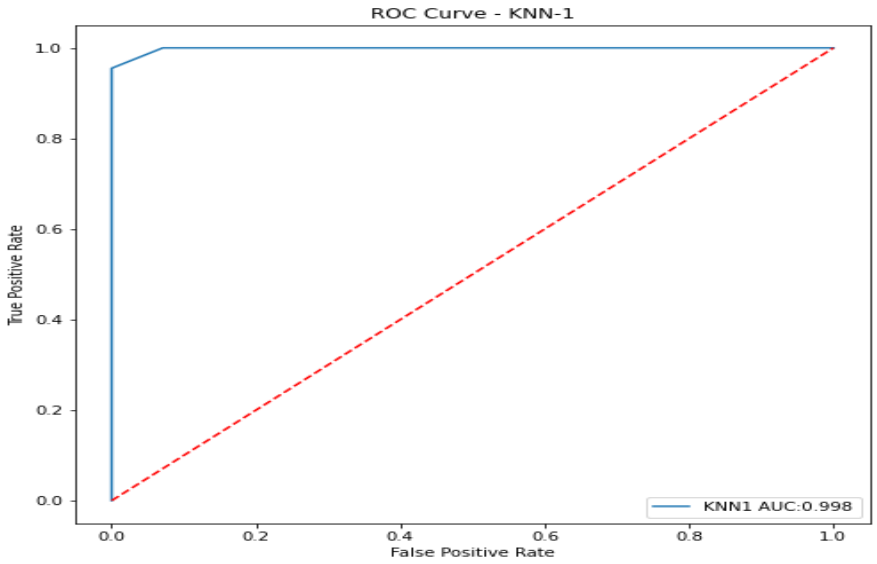

Fig.40

For test set:

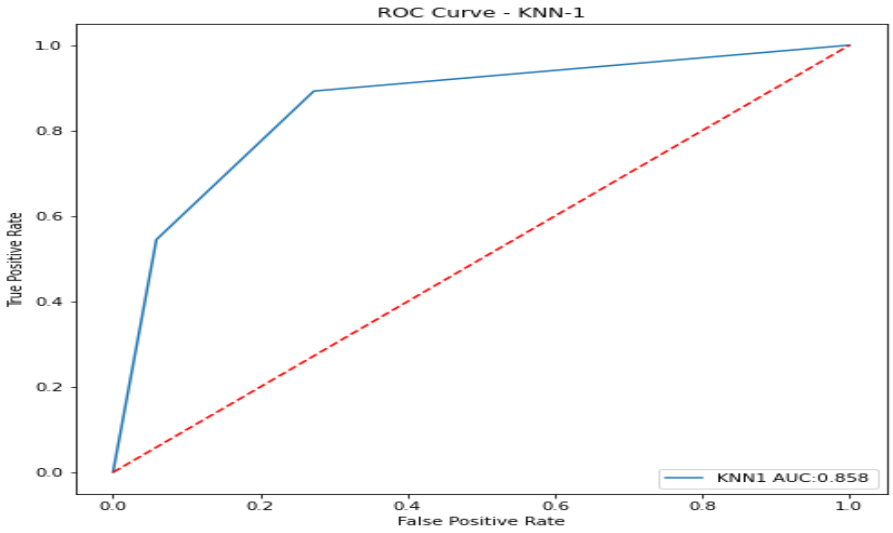

Fig.41

Model7: Neural Network:

1. For building NN model we have to scale train and test independent set.

2.These are the hyper parameters that has been used for model tuning.

```
param_grid = {
    'activation':['logistic', 'tanh', 'relu'],
    'hidden_layer_sizes': [100,200,300,500],
```

```
    'max_iter': [5000],
    'solver': ['sgd','adam'],
    'tol': [0.001],
}
```

3. After that we have to build logistic regression model and we have to pass the parameter grid into GridSearchCV. Next, we have to do model fitting.

4. Next, we can figure out best parameter for model.

```
{'activation': 'tanh',
 'hidden_layer_sizes': 300,
 'max_iter': 5000,
 'solver': 'adam',
 'tol': 0.001}
```

5. Predicting the probability and probability class for train set.

6. Calculating model performance matrices for train and test set.

Accuracy Score:

For train set:

```
0.9973021582733813
```

For test set:

```
0.6046353101567825
```

Confusion Matrix:

For train set:

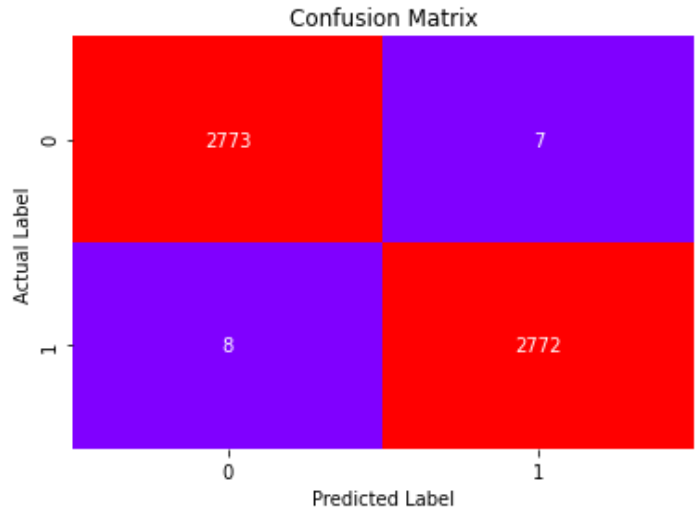

Fig.42

For test set:

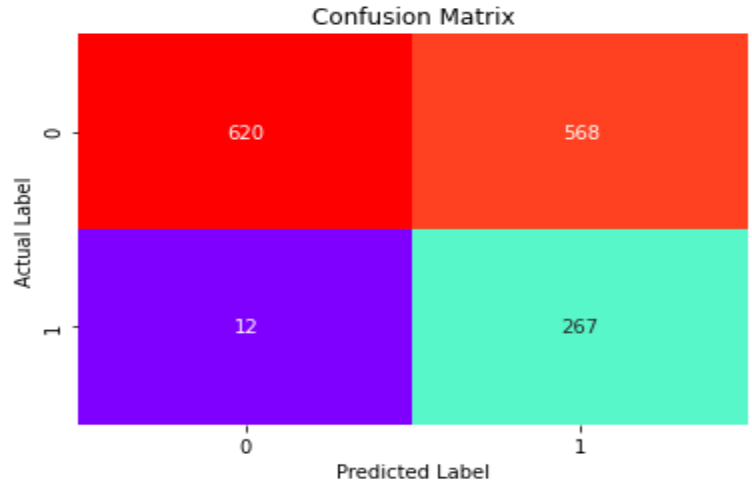

Fig.43

Classification Report:

For train set:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2780
1	1.00	1.00	1.00	2780
accuracy			1.00	5560
macro avg	1.00	1.00	1.00	5560
weighted avg	1.00	1.00	1.00	5560

For test set:

	precision	recall	f1-score	support
0	0.98	0.52	0.68	1188
1	0.32	0.96	0.48	279
accuracy			0.60	1467
macro avg	0.65	0.74	0.58	1467
weighted avg	0.86	0.60	0.64	1467

- 96 % customers are correctly identified as the customers who have taken product (good recall).

AUC and ROC Curve:

For train set:

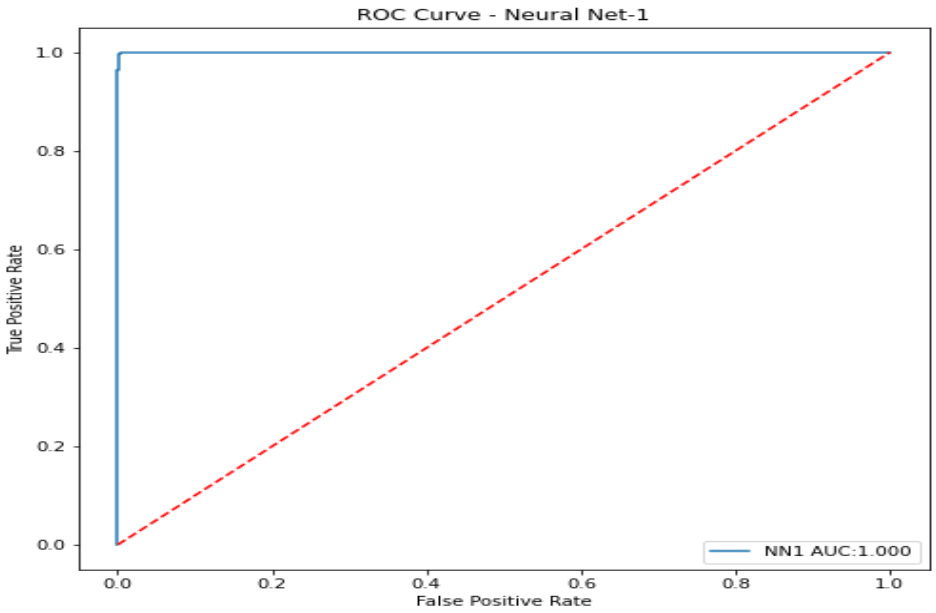

Fig.44

For test set:

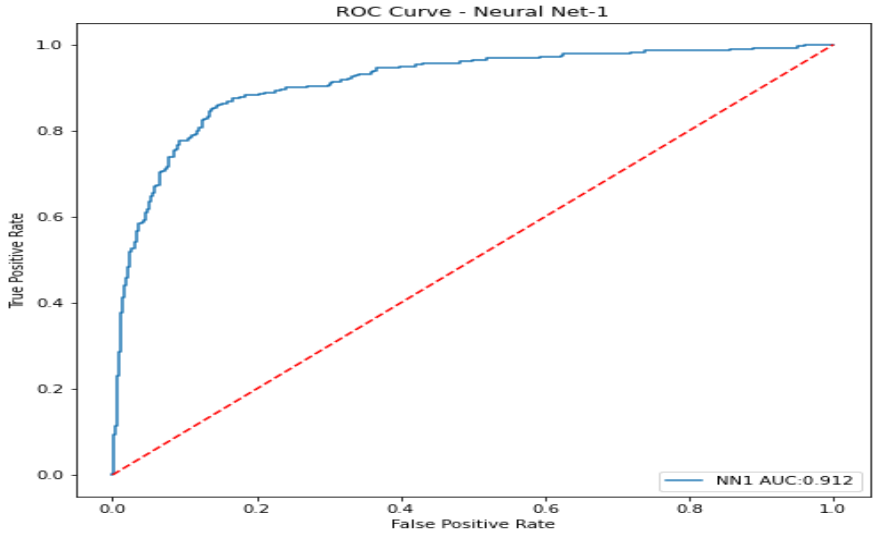

Fig.45

Model8: SVM(Support Vector Machine):

1. For building SVM model we have to scale train and test independent set.

2. These are the parameters that has been used for model tuning.

```
params = {'C':[0.01,1,10,20],
    'kernel':['linear','poly','rbf','sigmoid'],
   'probability':[True]}
```

3. After that we have to build logistic regression model and we have to pass the parameter grid into GridSearchCV. Next, we have to do model fitting.

4. Next, we can figure out best parameter for model.

```
{'C': 20, 'kernel': 'rbf', 'probability': True}
```

5. Predicting the probability and probability class for train set.

6. Calculating model performance matrices for train and test set.

Accuracy Score:

For train set:

```
0.9920863309352518
```

For test set:

```
0.7668711656441718
```

Confusion Matrix:

For train set:

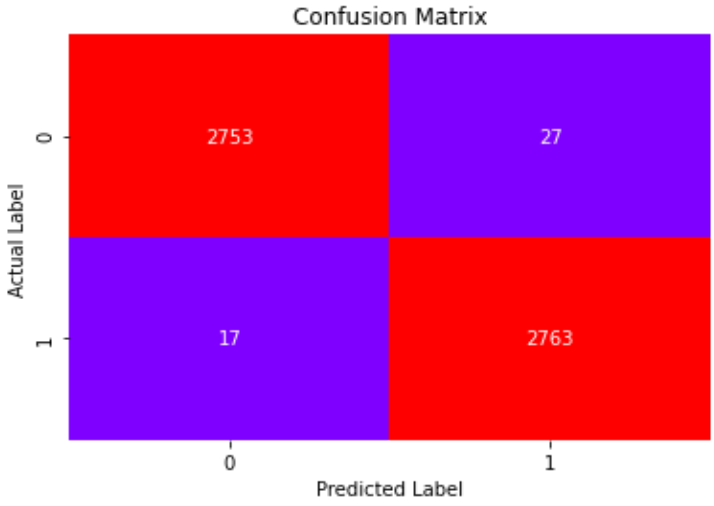

Fig.46

For test set:

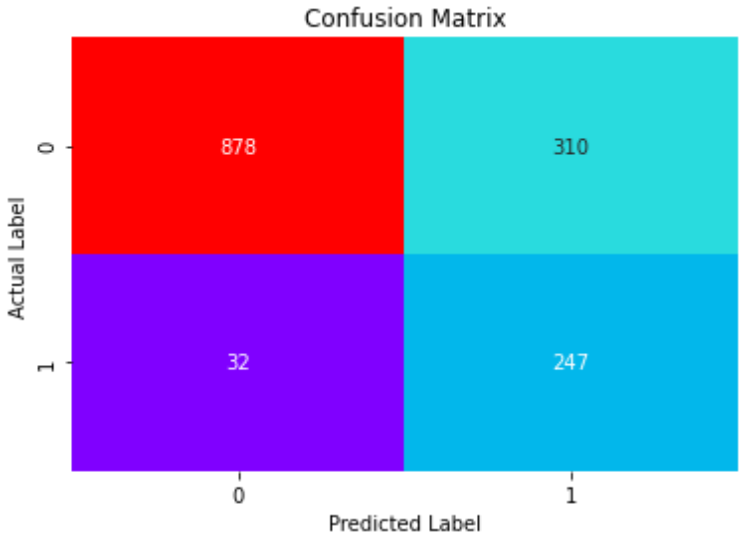

Fig.47

Classification Report:

For train set:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2780
1	0.99	0.99	0.99	2780
accuracy			0.99	5560
macro avg	0.99	0.99	0.99	5560
weighted avg	0.99	0.99	0.99	5560

For test set:

	precision	recall	f1-score	support
0	0.96	0.74	0.84	1188
1	0.44	0.89	0.59	279
accuracy			0.77	1467
macro avg	0.70	0.81	0.71	1467
weighted avg	0.87	0.77	0.79	1467

AUC and ROC Curve:

For train set:

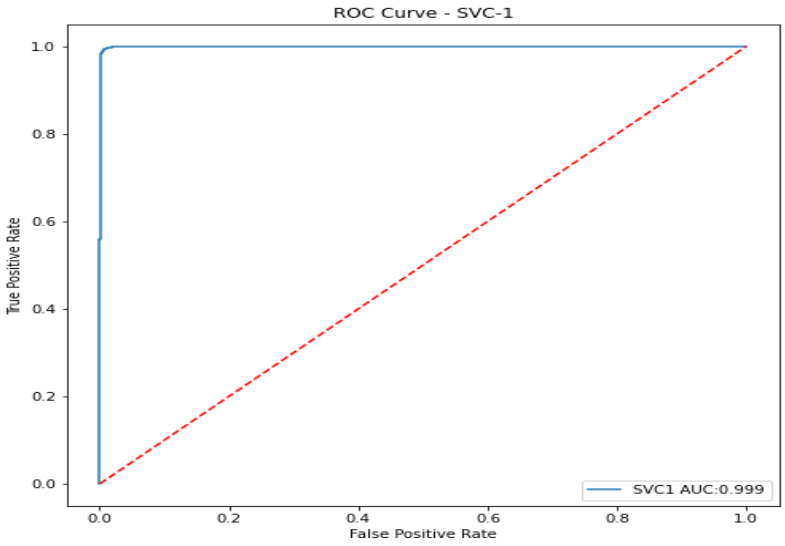

Fig.48

For test set:

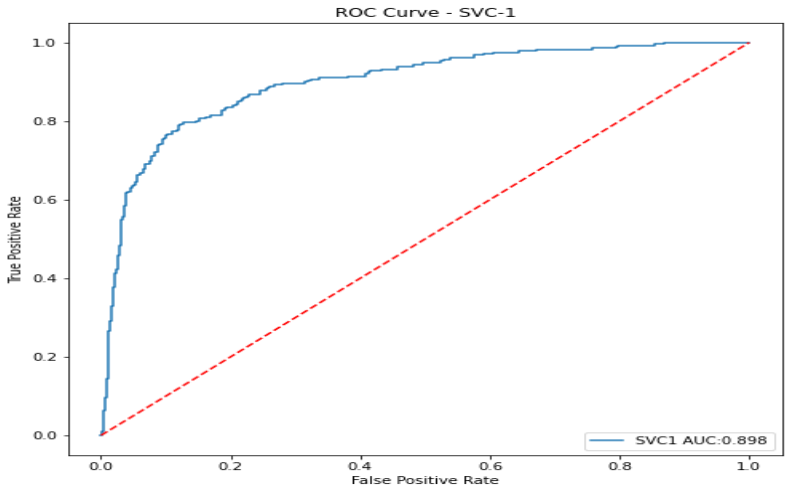

Fig.49

- AUC score is not good at test set. Hence model is performing overfitting.

Model10: Naive Bayes Classifier:

1. First we have to import GaussianNB from sklearn.naive_bayes. In naïve bayes classifier we can't do hyper parameter tuning.

2. After that we have to build model and fit the model.

```
NB1 = GaussianNB()

NB1.fit(X_train,Y_train.values.ravel())
```

4. Predicting the probability and probability class for train set.

5. Calculating model performance matrices for train and test set.

Accuracy Score:

For train set:

```
0.7528776978417266
```

For test set:

```
0.7007498295841854
```

Confusion Matrix:

For train set:

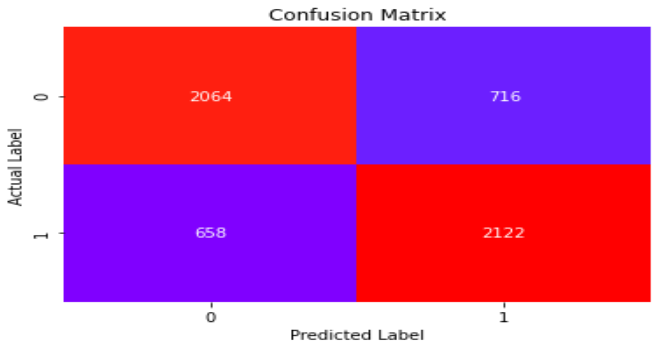

Fig.50

For test set:

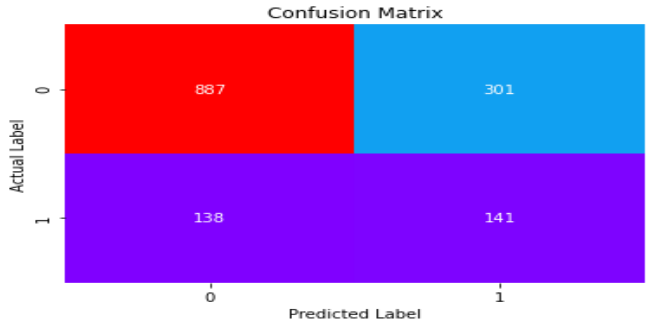

Fig.51

Classification Report:

For train set:

	precision	recall	f1-score	support
0	0.76	0.74	0.75	2780
1	0.75	0.76	0.76	2780
accuracy			0.75	5560
macro avg	0.75	0.75	0.75	5560
weighted avg	0.75	0.75	0.75	5560

For test set:

	precision	recall	f1-score	support
0	0.87	0.75	0.80	1188
1	0.32	0.51	0.39	279
accuracy			0.70	1467
macro avg	0.59	0.63	0.60	1467
weighted avg	0.76	0.70	0.72	1467

AUC and ROC Curve:

For train set:

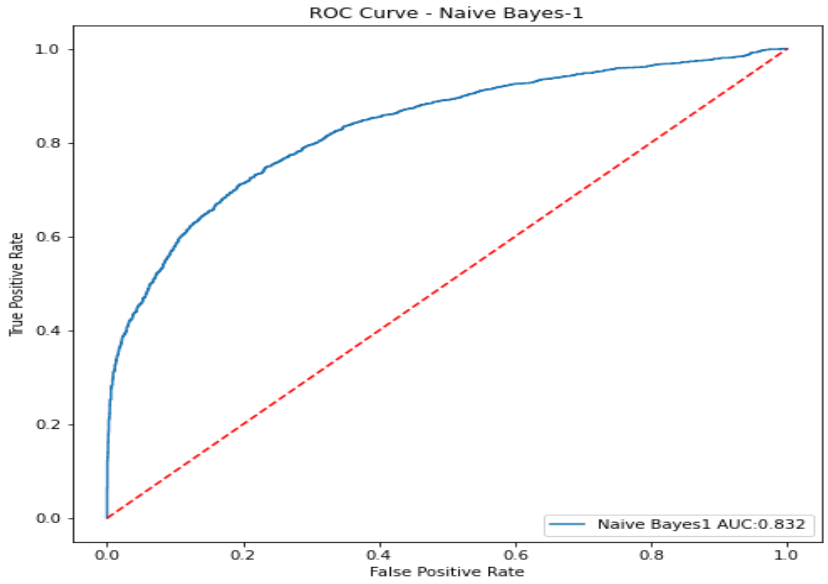

Fig.52

For test set:

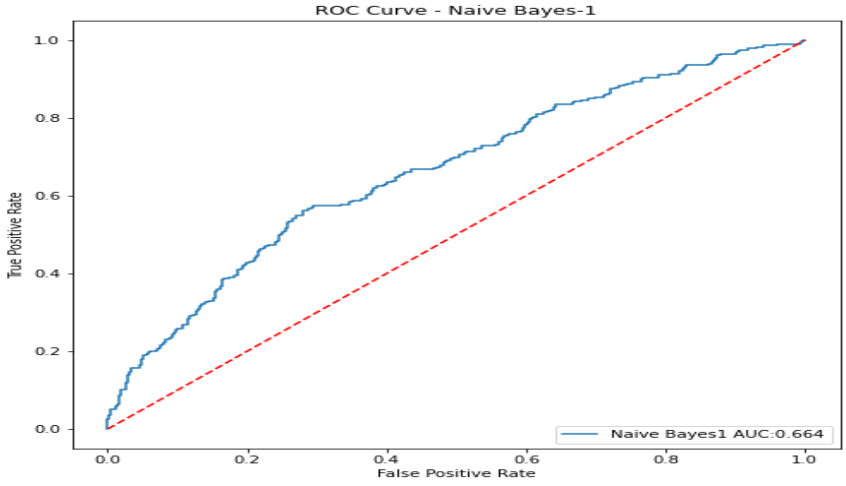

Fig.53

- AUC score is not good at test set. Hence model is performing overfitting.

Model 11: Bagging Classifier Using KNN as base model:

1. First we have to import BaggingClassifier from sklearn.ensemble and KneighborsClassifier from skearn.neighbors.
2. After that we have to build Bagging classifier model by BaggingClassifier() and pass the KNN model as base model. We can also pass parameter like max_samples,max_features,n_estimators. In next step, we have to do model fitting.

```
BaggingClassifier(base_estimator=KNeighborsClassifier(metric='manhattan',
                                                      n_neighbors=2),
                  max_features=15, max_samples=1000, n_estimators=500)
```

3. Predicting the probability and probability class fir train set.
4. Calculating the model performance metrics.

Accuracy Score:

For train set:

```
0.9336330935251799
```

For test set:

```
0.7989093387866394
```

Confusion Matrix:

For train set:

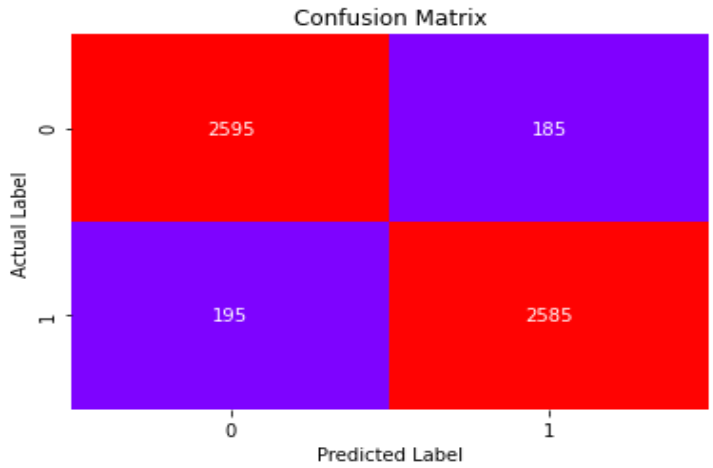

Fig.54

For test set:

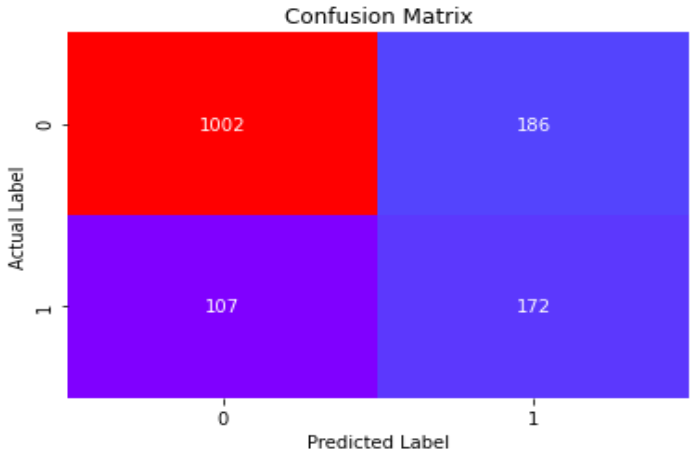

Fig.55

Classification Report:

For train set:

	precision	recall	f1-score	support
0	0.93	0.93	0.93	2780
1	0.93	0.93	0.93	2780
accuracy			0.93	5560
macro avg	0.93	0.93	0.93	5560
weighted avg	0.93	0.93	0.93	5560

For test set:

	precision	recall	f1-score	support
0	0.91	0.84	0.87	1188
1	0.48	0.62	0.54	279
accuracy			0.80	1467
macro avg	0.69	0.73	0.71	1467
weighted avg	0.82	0.80	0.81	1467

AUC and ROC Curve:

For train set:

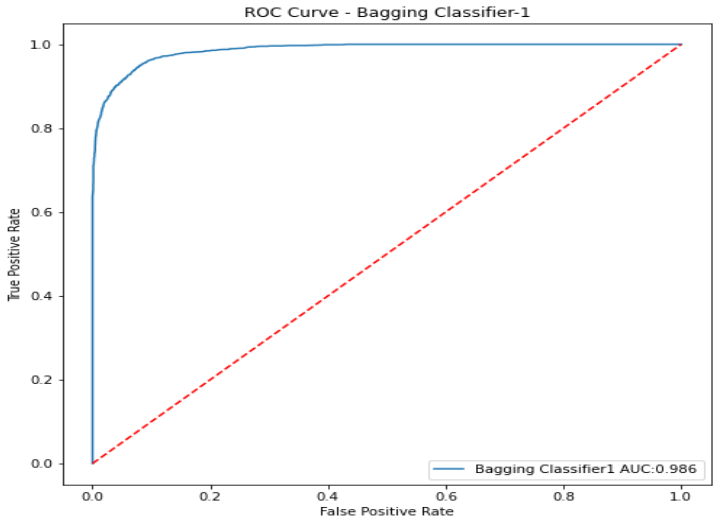

Fig.56

For test set:

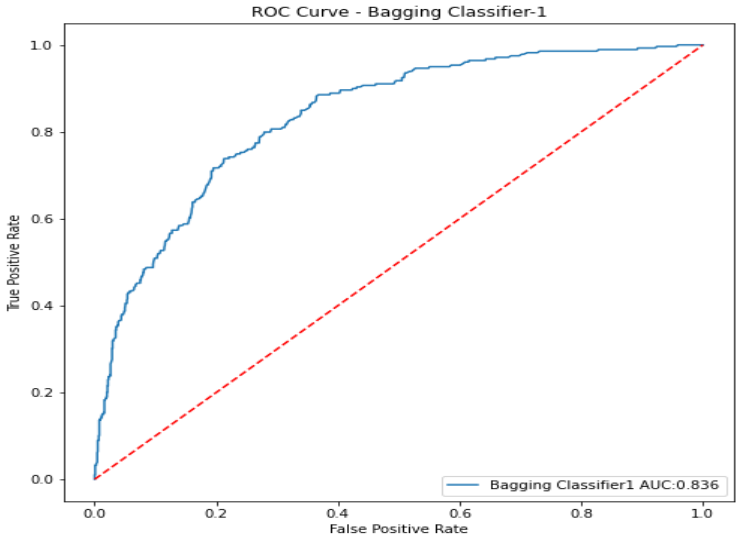

Fig.57

- AUC Score is not good at test side. Model is performing overfitting.

Model 12: Bagging Classifier with Decision tree as base model:

1. First we have to import BaggingClassifier from sklearn.ensemble and DecisionTreeClassifier from skearn.tree.
2. After that we have to build Bagging classifier model by BaggingClassifier() and pass the decision tree as base model. We can also pass parameter like max_samples,max_features,n_estimators. In next step, we have to do model fitting.

```
cart = DecisionTreeClassifier()
```

```
Bagging_model=BaggingClassifier(base_estimator=cart,n_estimators=100,max_samples=1000,
                max_features=15,random_state=1)
BC2=Bagging_model.fit(X_train, Y_train.values.ravel())
```

3. Predicting the probability and probability class fir train set.
4. Calculating the model performance metrics.

Accuracy Score:

For train set:

```
0.956294964028777
```

For test set:

```
0.8498091342876619
```

Confusion Matrix:

For train set:

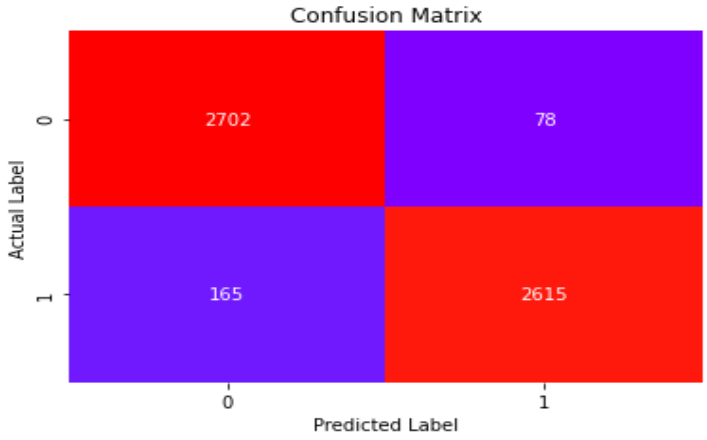

Fig.58

For test set:

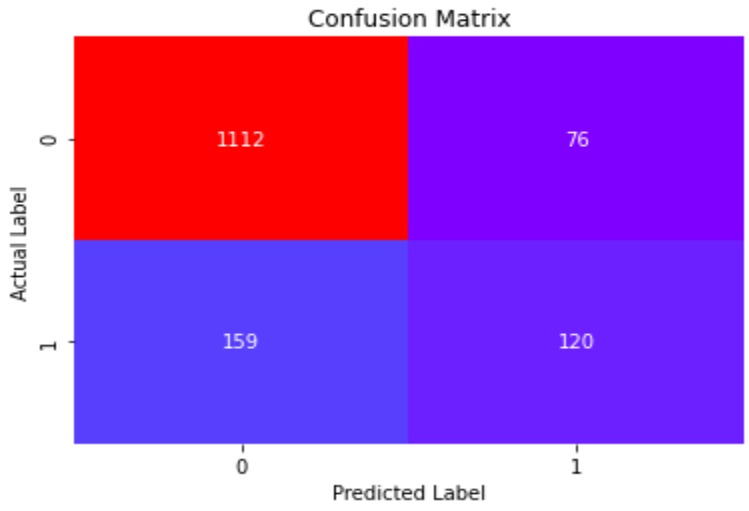

Fig.59

Classification Report:

For train set:

	precision	recall	f1-score	support
0	0.94	0.97	0.96	2780
1	0.97	0.94	0.96	2780
accuracy			0.96	5560
macro avg	0.96	0.96	0.96	5560
weighted avg	0.96	0.96	0.96	5560

For test set:

	precision	recall	f1-score	support
0	0.87	0.94	0.90	1188
1	0.61	0.43	0.51	279
accuracy			0.84	1467
macro avg	0.74	0.68	0.70	1467
weighted avg	0.82	0.84	0.83	1467

AUC and ROC Curve:

For train set:

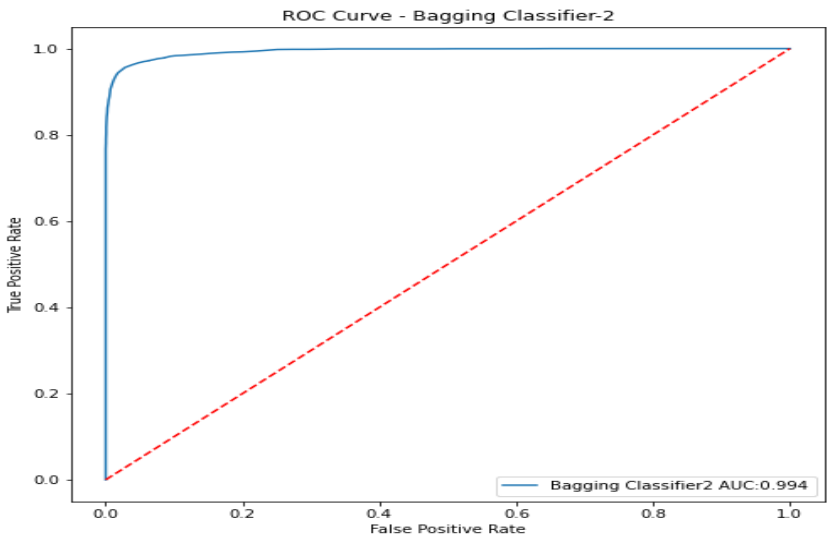

Fig.60

For test set:

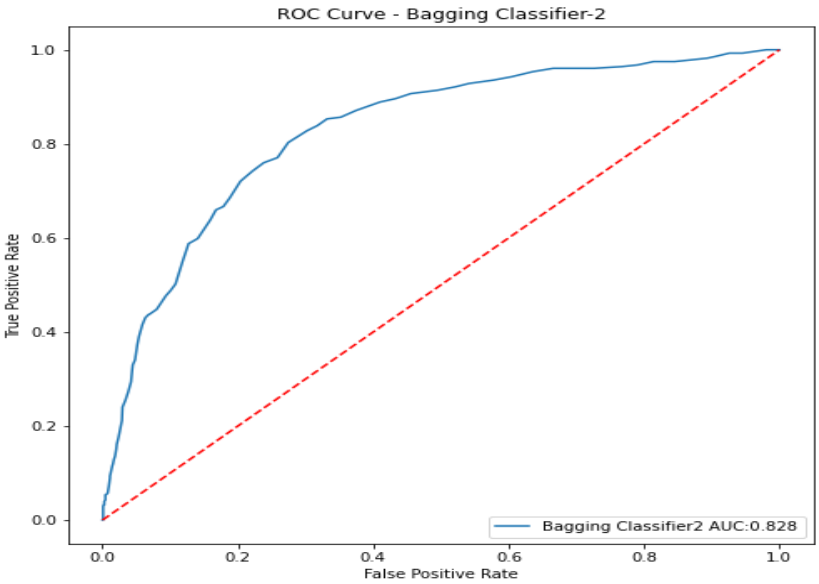

Fig.61

- AUC score is not good at test set. Hence we can say model is performing overfiiting.

Now, I am going to use voting classifier to further improve the performance of model. See Appendix E.

Model 14: Voting Classifier1:

1. First we have to import VotingClassifer from sklearn.ensemble.
2. Next, we will create list of different models like logistic regression, decision tree, random forest, gradient boosting,naïve bayes, bagging with decision tree that we have to pass as base model into VotingClassifier().
   estim = [('LR1',LR1),('LR2',LR2),('DT1',DT1),('RF1',RF1),('GB1',GB1),('NB1',NB1),('BC2',BC2)]
3. Fit the model for train set.
4. Predicting the probability and probability class for train and test set.
5. Calculating performance metrics.

Accuracy score:

For train set:

```
0.9751798561151079
```

For test set:

```
0.8452624403544649
```

Confusion Matrix:

For train set:

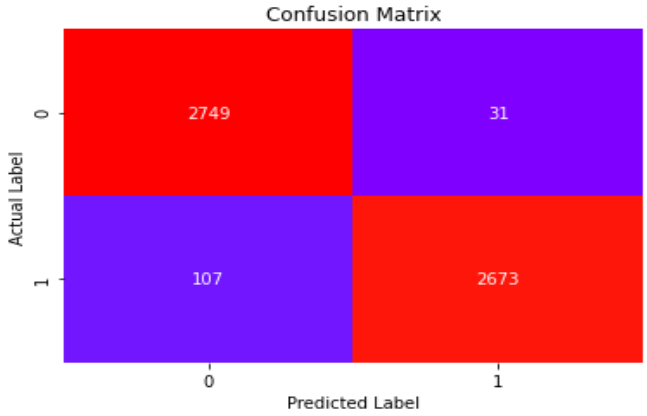

Fig.62

For test set:

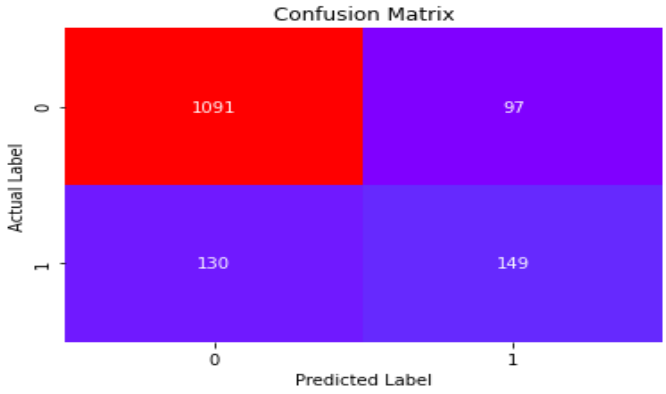

Fig.63

- Total no of correct prediction=149+1091
- Total no of incorrect prediction=130+97

Classification Report:

For train set:

	precision	recall	f1-score	support
0	0.96	0.99	0.98	2780
1	0.99	0.96	0.97	2780
accuracy			0.98	5560
macro avg	0.98	0.98	0.98	5560
weighted avg	0.98	0.98	0.98	5560

For test set:

	precision	recall	f1-score	support
0	0.89	0.92	0.91	1188

1	0.61	0.53	0.57	279
accuracy			0.85	1467
macro avg	0.75	0.73	0.74	1467
weighted avg	0.84	0.85	0.84	1467

- Only 53 % customers are correctly identified as the customers who have t aken product by Voting classifier1(VO1).

AUC and ROC curve:

For train set:

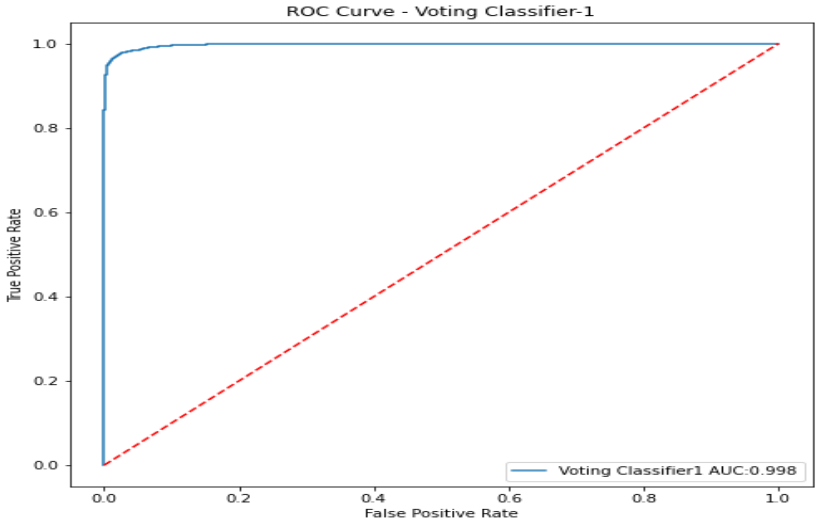

Fig.64

For test set:

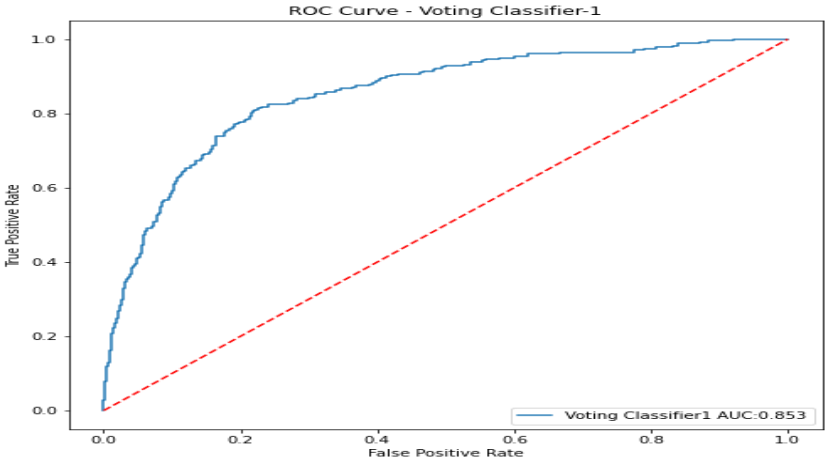

Fig.65

- AUC score for test set is not as good as train set. It means model performance is over fitting.

Model 15:VotingClassifer2:

1. First we have to import VotingClassifer from sklearn.ensemble.
2. Next, we will create list of different model likeKNN,SVC1,NN1,Bagging with decision tree(BC1) that we have to pass as base model into VotingClassifier().
   estim = [('KNN1',KNN1),('NN1',NN1),('SVC1',SVC1),('BC1',BC1)]
3. Fit the model for train set.
4. Predicting the probability and probability class for train and test set.
5. Calculating performance metrics.

Accuracy Score:

For train set:

```
0.9940647482014389
```

For test set:

```
0.8118609406952966
```

Confusion Matrix:

For train set:

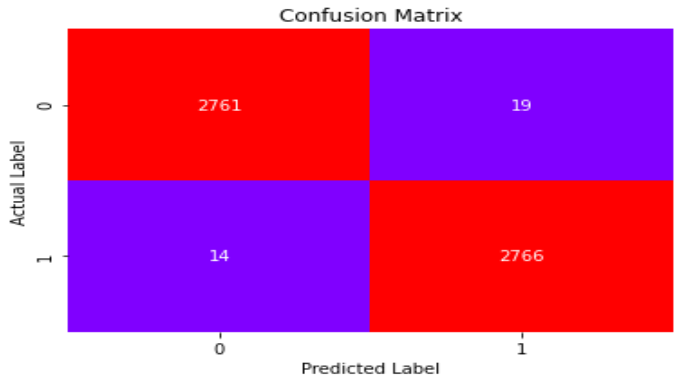

Fig.66

For test set:

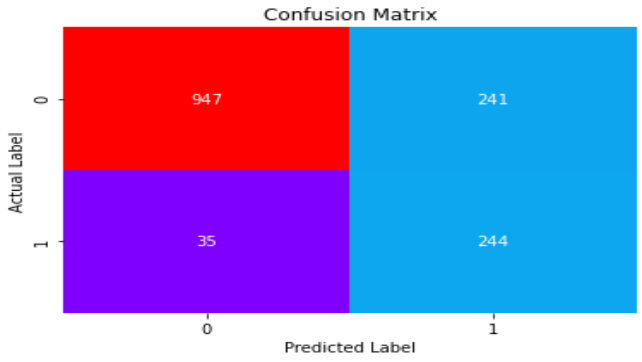

Fig.67

Classification Report:

For train set:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2780
1	0.99	0.99	0.99	2780
accuracy			0.99	5560
macro avg	0.99	0.99	0.99	5560
weighted avg	0.99	0.99	0.99	5560

For test set:

	precision	recall	f1-score	support
0	0.96	0.80	0.87	1188
1	0.50	0.87	0.64	279
accuracy			0.81	1467
macro avg	0.73	0.84	0.76	1467
weighted avg	0.88	0.81	0.83	1467

- 87 % customers are correctly identified , the customers who have taken pr oduct.

AUC and ROC Curve:

For train set:

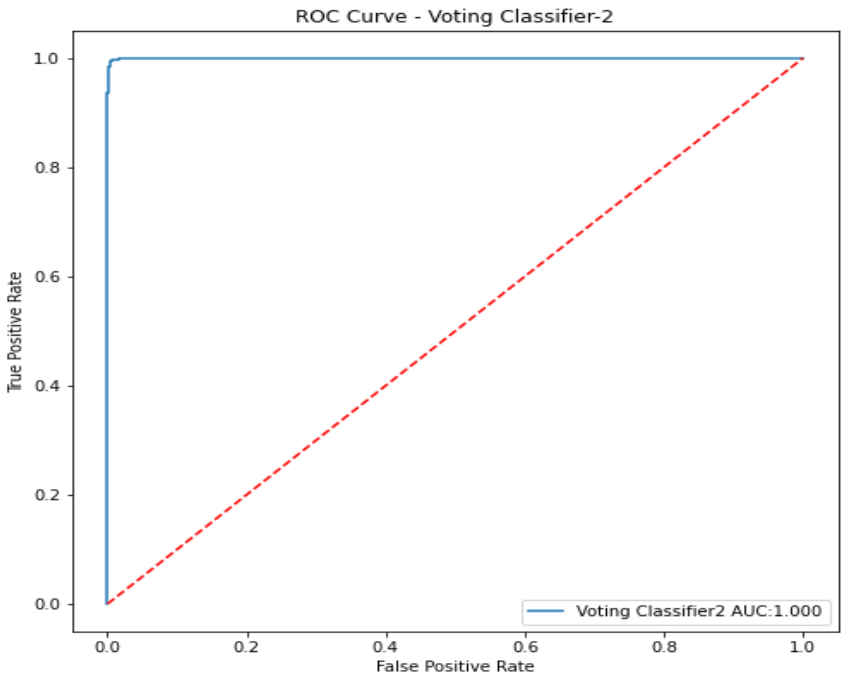

Fig.68

For test set:

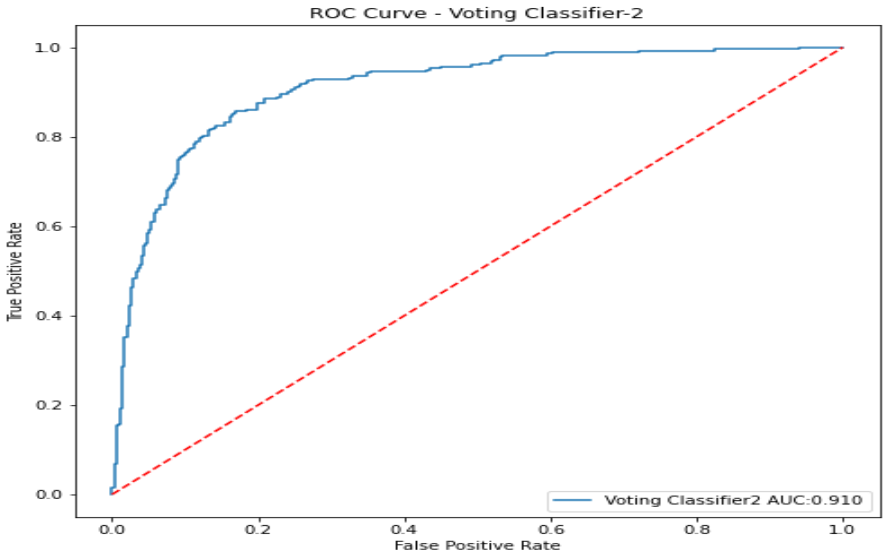

Fig.69

- AUC score is also good at test set as well as train set. Hence, model is valid.

Now, we are going to try another classifier which is known as Stacking classifier to further improve the performance of model. See Appendix F.

Model 16:StackingClassifier1:

1. For stacking classifier, first I have taken logistic regression model as meta classifier and LR1,LR2,DT1,GB1,NB1,BC1 as base model and then pass the parameter estimator and final estimator into StackingClassifer and fit the model on train set.

2. Predicting the probabilities and probability class.
3. Calculating the performance metrics.

Accuracy Score:

For train set:

```
1.0
```

For test set:

```
0.885480572597137
```

Confusion matrix:

For train set:

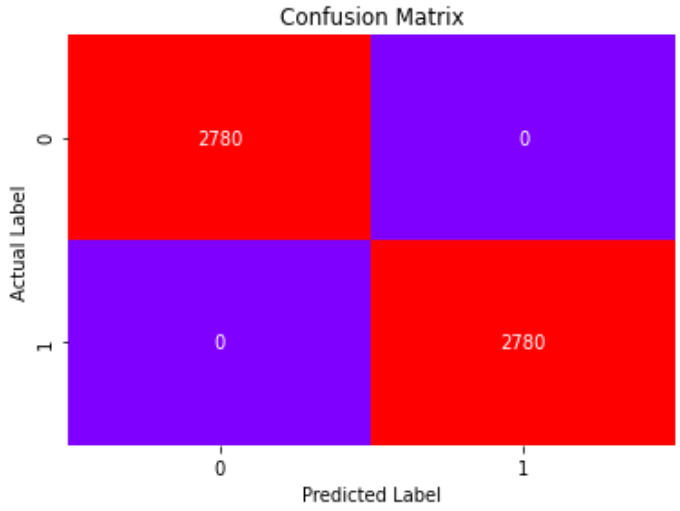

Fig.70

For test set:

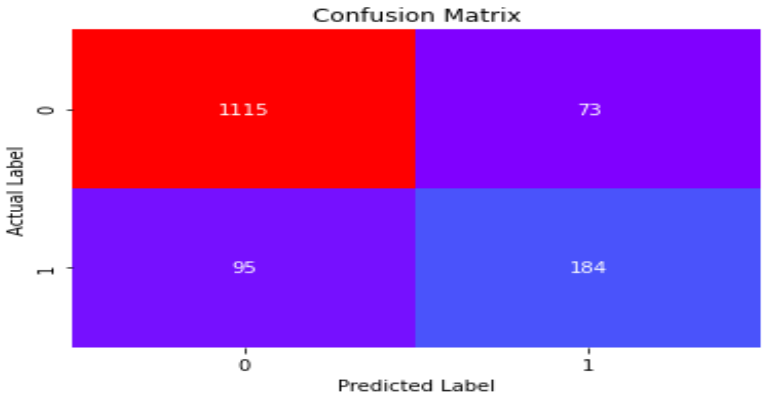

Fig.71

Classification Report:

For train set:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2780
1	1.00	1.00	1.00	2780
accuracy			1.00	5560
macro avg	1.00	1.00	1.00	5560
weighted avg	1.00	1.00	1.00	5560

For test set:

	precision	recall	f1-score	support
0	0.92	0.94	0.93	1188
1	0.72	0.66	0.69	279

accuracy			0.89	1467
macro avg	0.82	0.80	0.81	1467
weighted avg	0.88	0.89	0.88	1467

AUC and ROC Curve:

For train set:

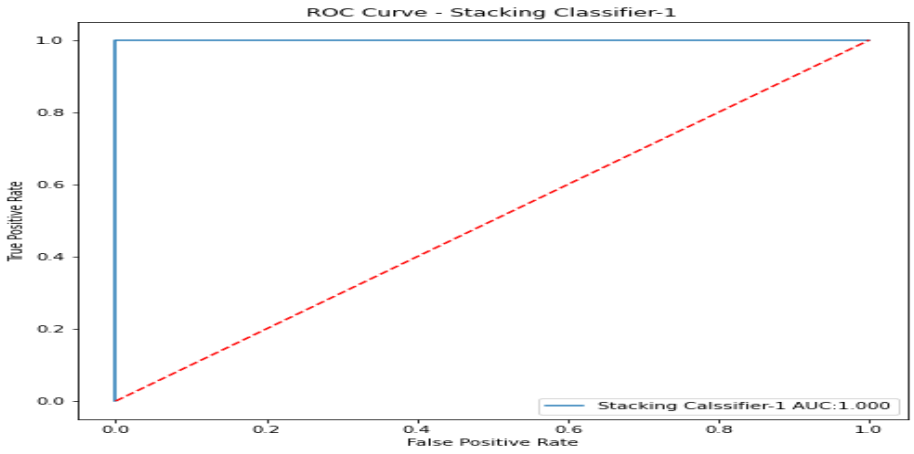

Fig.72

For test set:

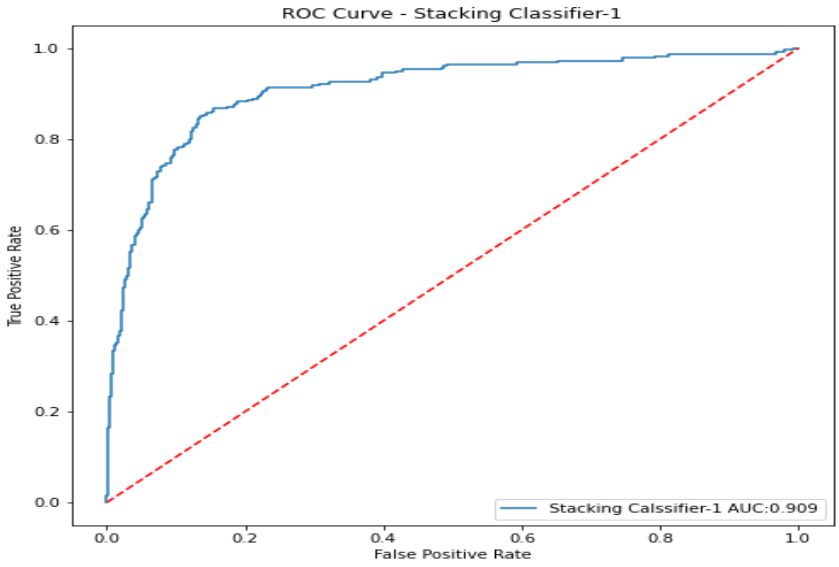

Fig.73

- There is only ±10% difference between AUC score of train and test set. Hence, we can say model is valid.

Model 17:StackingClassifier2:

1. For this case, I have taken logistic regression model as meta classifier and KNN1,NN1,SVM1 and Bagging Classifier with BC1 as base model and then pass the parameter estimator and final estimator into StackingClassifer and fit the model on train set. Train and test set (only independent set) should be scaled in this case because scaling is necessary for all model that we have to ensemble.
2. Predicting the probabilities and probability class.
3. Calculating performance metrics.

Accuracy Score:

For train set:

```
0.9951438848920864
```

For test set:

```
0.7934560327198364
```

Confusion Matrix:

For train set:

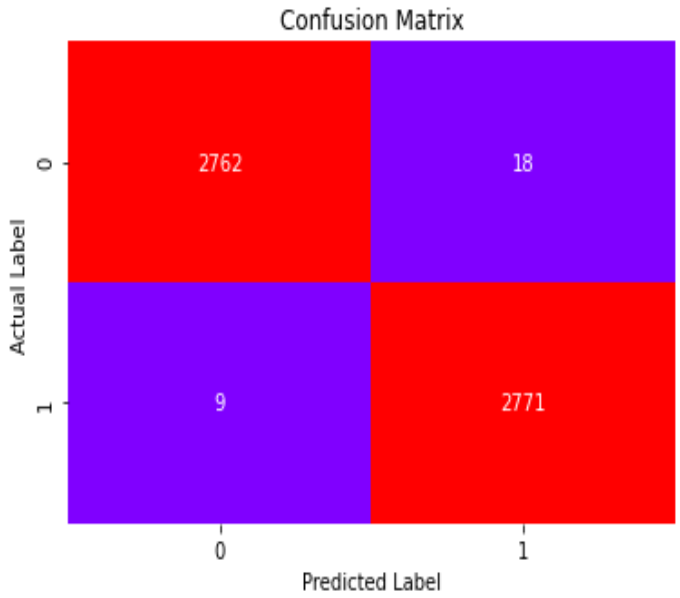

Fig.74

For test set:

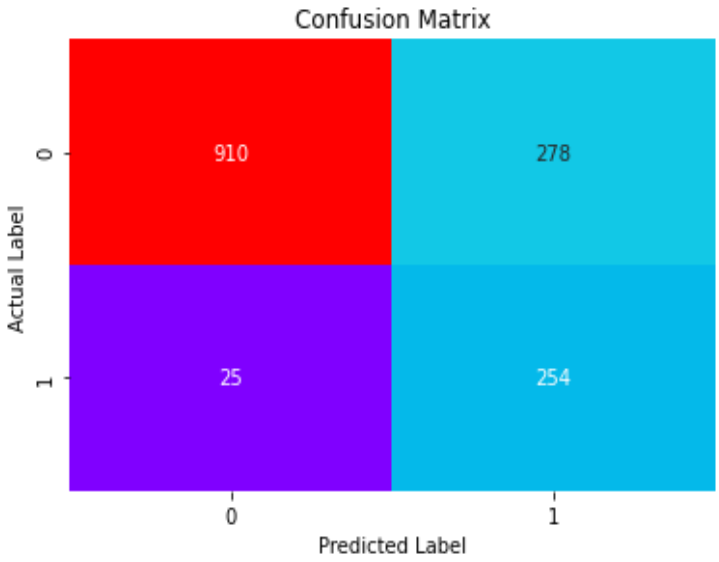

Fig.75

- For test set, total no correct prediction=254+910
- For test set, total no of incorrect prediction=25+278

Classification Report:

For train set:

	precision	recall	f1-score	support
0	1.00	0.99	1.00	2780
1	0.99	1.00	1.00	2780
accuracy			1.00	5560
macro avg	1.00	1.00	1.00	5560
weighted avg	1.00	1.00	1.00	5560

For test set:

	precision	recall	f1-score	support
0	0.97	0.77	0.86	1188
1	0.48	0.91	0.63	279
accuracy			0.79	1467
macro avg	0.73	0.84	0.74	1467
weighted avg	0.88	0.79	0.81	1467

- 91 % customers are correctly identified as the customers who have taken product.

AUC and ROC Curve:

For train set:

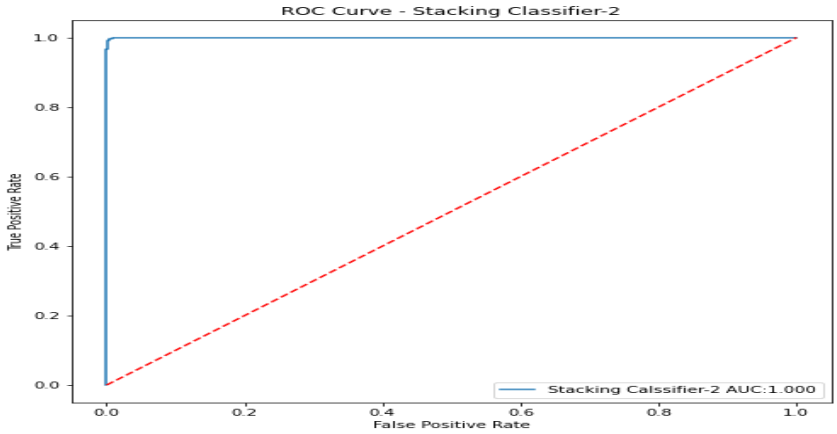

Fig.76

For test set:

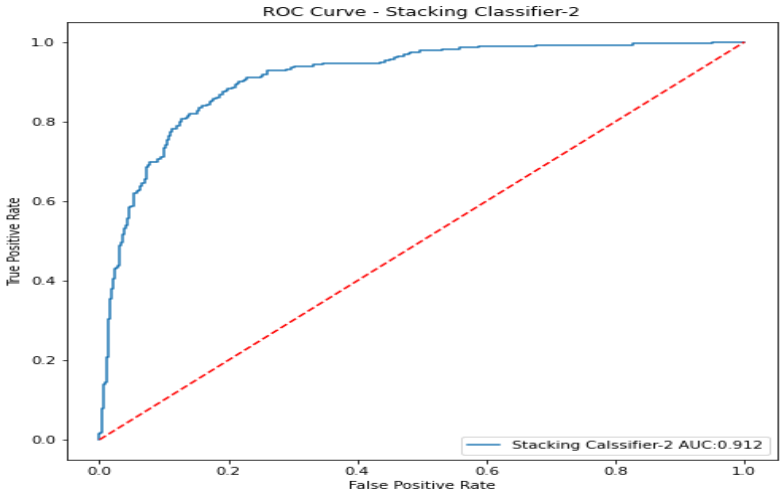

Fig.77

- There is only ±10% difference between AUC score of train and test set. Hence, we can say, model is valid.

Comparing the model performance:

As we have to know, we have to predict, whether a customer will opt long term tourism package or not. Hence, for model selection criterion would be the model which has more accurately predicted class 1.Therefore, we have to check recall that predict how many true data points identified as true.

Recall for Neural Network (NN1) for test set is (0.96) that is more than all other model's test set that we have done so far.

Model	Accuracy		Precision		Recall		F1-score		AUC	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Logistic regression (LR1)	0.81	0.77	0.78	0.51	0.84	0.42	0.81	0.46	0.81	0.67
Logistic Regression with Recursive Feature Elimination (LR2)	0.81	0.77	0.84	0.41	0.77	0.49	0.80	0.45	0.81	0.66
Decision Tree (DT1)	0.99	0.83	0.99	0.53	0.99	0.67	0.99	0.59	0.99	0.76
Random Forest (RF1)	1	0.87	1	0.72	1	0.49	1	0.58	1	0.89
Gradient Boosting(GB1)	1	0.89	1		1		1		1	
K-Nearest Neighbours (KNN1)	0.98	0.87	1	0.68	0.96	0.54	0.98	0.61	0.99	0.85
Neural Network (NN1)	1	0.60	1	0.32	1	**0.96**	1	0.48	1	**0.91**
Support Vector Classifier(SVC1)	0.99	0.77	0.99	0.44	0.99	0.89	0.99	0.59	0.99	0.90
Naïve Bayes (NB1)	0.75	0.70	0.75	0.32	0.76	0.51	0.76	0.39	0.83	0.66
Bagging with K-Nearest Neighbours (BC1)	0.93	0.80	0.93	0.48	0.93	0.62	0.93	0.54	0.99	0.83
Bagging with Decision tree (BC2)	0.96	0.84	0.98	0.62	0.94	0.42	0.96	0.50	0.99	0.83
Voting Classifier1 (VO1)	0.98	0.85	0.99	0.61	0.96	0.53	0.97	0.57	0.99	0.85
Voting Classifier2 (VO2)	0.99	0.81	0.99	0.50	0.99	0.87	0.99	0.64	1	0.91
Stacking Classifier1 (STA1)	1	0.89	1	0.72	1	0.66	1	0.69	1	0.91
Stacking Classifier2 (STA2)	1	0.79	0.99	0.48	1	0.91	1	0.63	1	0.91

AUC score is also good (0.91) for Neural Network1 (NN1) and (STA1 and STA2) Stacking Classifier1 (0.91) is more than others model. As we know, higher the AUC score, better the model is. But recall is poor for model STA1 (0.62) and not so poor for STA2 (0.91) but less than NN1.So we conclude here, NN1 is best model. Although accuracy is not so good for NN1, we will consider it as best model because accuracy is not measure concern to decide the model is good or not.

"Neural Network (NN1) is the best model."

Visualization the model performance:

This depicts, how the models are performing with respect to different metrics.

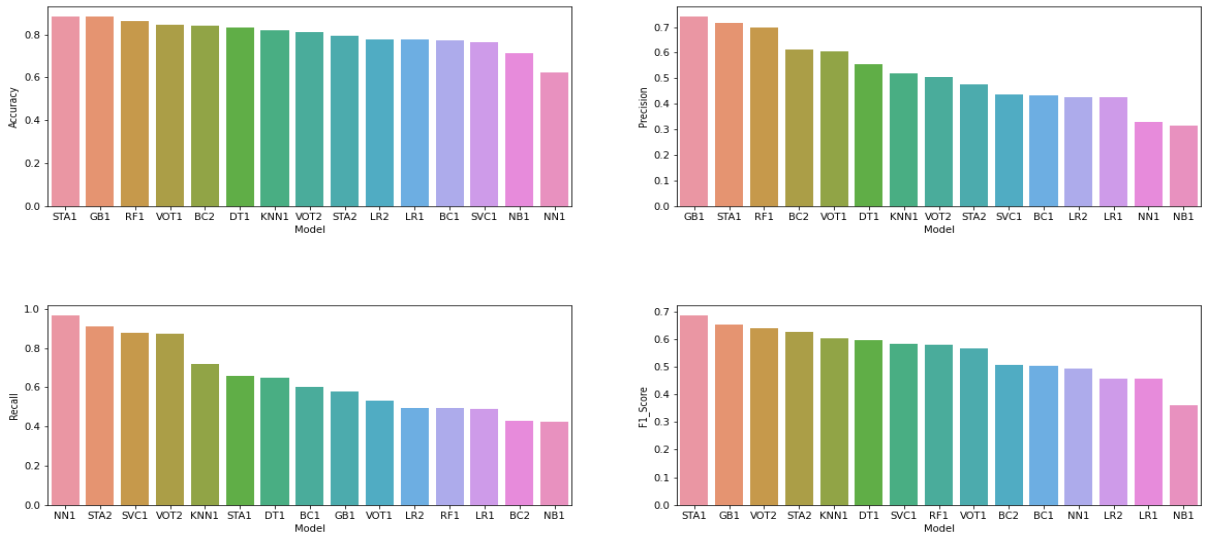

Fig.78

Observations:

- Recall is the best for Neural network model(NN1).
- AUC is the best for Stacking classifier1,2(STA1,2)andNN1).
- Precision is the best for gradient boosting (GB1).
- F1-score is the best for Stacking Classifier1(STA1).

Business Insights:

- 1. A customer having passport, increases the probability of taken prod by 76%.
- If the customer age increases, the probability of taken product decreases by 50%.

- A customer belongs to Tier-3, increases the probability of prod taken increases by 61%.
- If the monthly income of customer increases, the probability of prod taken increases by 50%.
- If the number of trips increases, probability of taking prod decreases by 51%.
- If the customer having small business and salaried, the probability of taking product decreases by 1%.
- A person having own car, decreases the probability of product taken by 36%.
- A customer, who has married, decreases the probability of product taken by 19%.
- A customer, whose designation is high (work as VP, AVP), decreases the probability of product taken by 31%.

Recommendations:

- Cluster-1 is the group of younger people who have passport also. For business perspective Travel Company should target these people for selling the product. Also thought of some strategy so that non passport holder can get passport so that propensity of product could increase.

- Cluster-3, Cluster-4 and cluster-5 are the largest chunk of the customers who are not taking product. So for business perspective, company should organize some campaign to increase the purchasing of product and business.

- How frequently a salesperson is following up with customers, increases the probability of product taken. So, for business perspective we need a very proactive and aggressive sales team.

- Most of the customers who have passport, their probability of product taken is very high. Hence for business perspective, travel company should plan to provide a passport and visa services to non- passport holder.

- The another largest chunk of customers in tier3 city .Hence, the travel company should organize some campaign to increase the purchasing of product and business.

- PitchSatisfactionScore also increases the buying of the product. Therefore, travel company should provide some training to the salesperson so that salesperson could convince the customers easily.
- Most of the customers are married and have children 2 or more and they don’t have passport. Hence for business perspective we should provide some offers and passport service to increase the purchasing of product taken.

Appendix

A. Binning Approach: Binning method is used to smoothing data or to handle noisy data. In this method, the data is first sorted and then the sorted values are distributed into a number of buckets or bins. As binning methods consult the nei-ghborhood of values, they perform local smoothing.

When dealing with continuous numeric data, it is often helpfu l to bin the data into multiple buckets for further analysis. There are several diff erent terms for binning including bucketing, discrete binning, discretization or quantization. Pandas supports these approaches using the cut and qcut functions.

Binning using quartiles: durationOfPitch:

This approach describes as a "Quantile-based discretization function." This basi cally means that qcut tries to divide up the underlying data into equal sized bins. The function defines the bins using percentiles based on the distribution of the d ata, not the actual numeric edges of the bins

B. Scatterplot between Age and NumberOfTrips regarding ProdTaken:

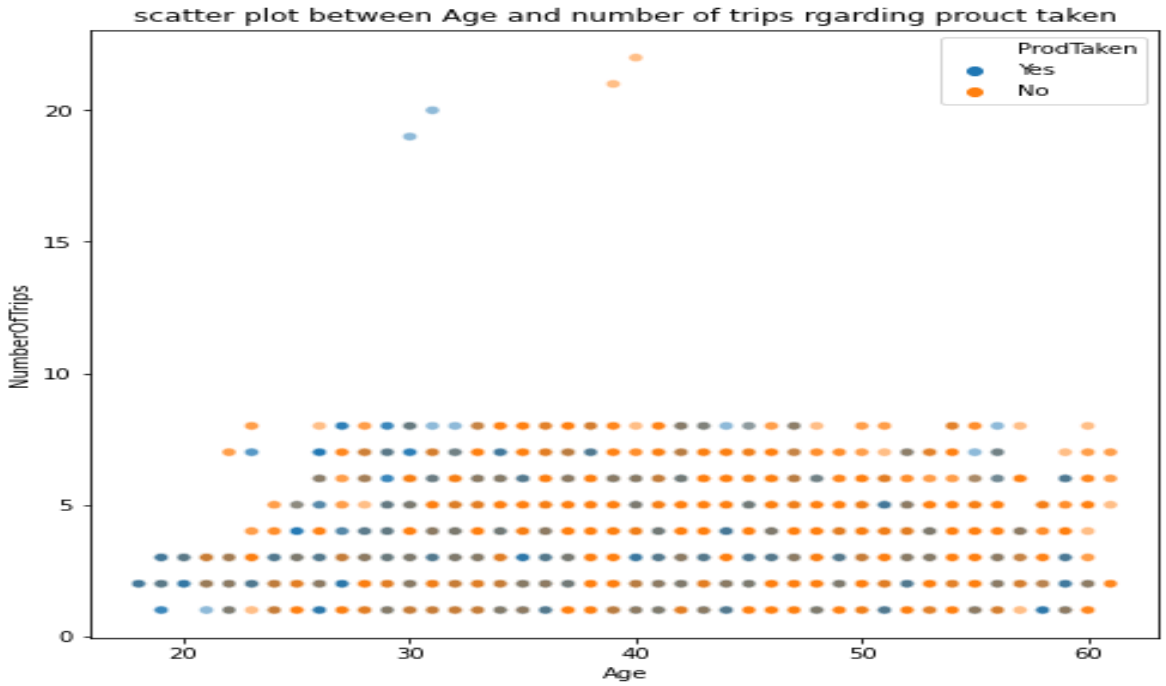

Fig: 11

- We cannot find any correlation.

Scatterplot between MonthlyIncome and Age regarding ProdTaken:

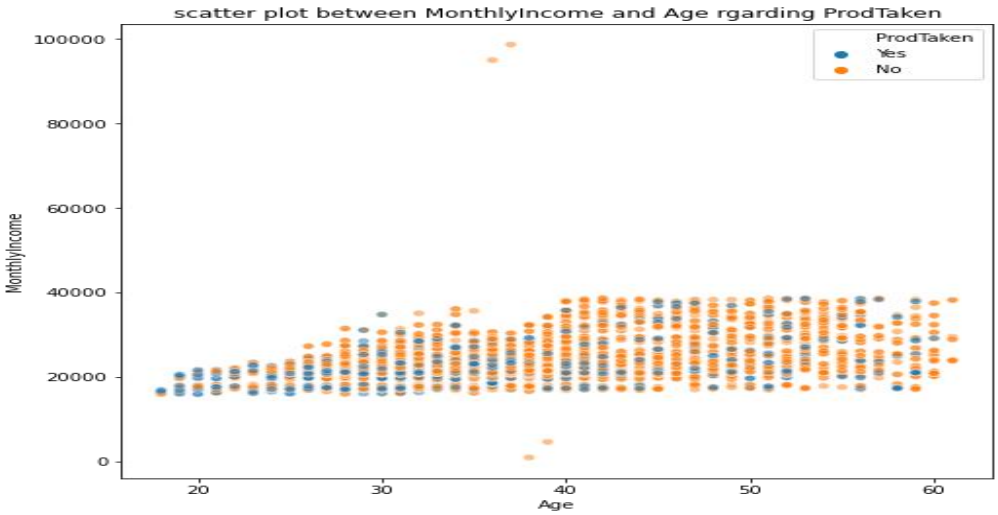

Fig: 12

Observations:

- Concentration of blue dots are high at lower age group people and inco me range 20000-40000.
- Most of the customers that are taken product belong to young age and middle age group and they have monthly income 2000 to 40000. After 40 years of age monthly income of the customers are saturated. We ca n conclude that they all are the customers who belong to higher design ation and their monthly income is high as well.

Scatterplot between NumberOfPersonVisited and Age regarding ProdTaken:

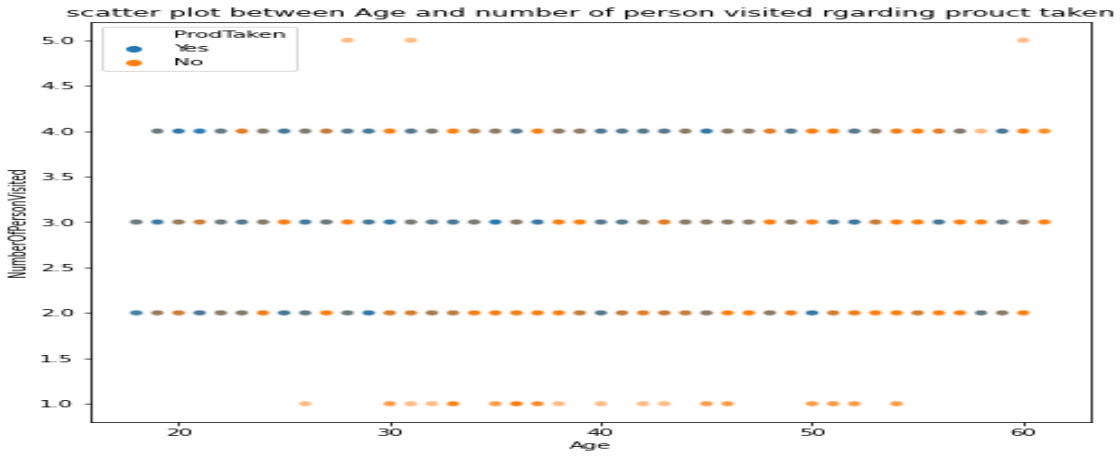

Fig: 13

Observations:

Most of the customers who are taken product belong to younger and middle age group and they belong to small and middle family as well.

C:Cluster Analysis:

1.First take the df_tourism4 data set .This is the copy of df_tourism2 data set.

2. Let's check data types and variables by info().

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4888 entries, 0 to 4887
Data columns (total 19 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ProdTaken                4888 non-null   object
 1   PreferredLoginDevice     4888 non-null   object
 2   CityTier                 4888 non-null   object
 3   Occupation               4888 non-null   object
 4   Gender                   4888 non-null   object
 5   ProductPitched           4888 non-null   object
 6   PreferredPropertyStar    4888 non-null   object
 7   MaritalStatus            4888 non-null   object
 8   Passport                 4888 non-null   object
 9   OwnCar                   4888 non-null   object
 10  Designation              4888 non-null   object
 11  Age                      4888 non-null   float64
 12  DurationOfPitch          4888 non-null   float64
 13  NumberOfPersonVisited    4888 non-null   int64
 14  NumberOfFollowups        4888 non-null   float64
 15  NumberOfTrips            4888 non-null   float64
 16  PitchSatisfactionScore   4888 non-null   int64
 17  NumberOfChildrenVisited  4888 non-null   float64
 18  MonthlyIncome            4888 non-null   float64
dtypes: float64(6), int64(2), object(11)
memory usage: 725.7+ KB
```

There are two types of categorical variable in the data set wherein some are or o rdinal like ProductPitched,PreferredPropertyStar, Designation which is ranked b ased and rest of all are categorical where weightage are equal for all different la bel .

3. For sake of clustering, we need to convert all categorical variables into numer ic al. For ordinal categorical variable we will use map and lambda function or Categorical().code and other categorical variable we will use one hot encoding a nd or dummy variable creation.

```
df_tourism4['ProductPitched_codes'] = df_tourism4['ProductPitched'].map({'Multi':1,'Standard':2,'Deluxe':3,'Su
per Deluxe':4,'King':5})
df_tourism4.drop('ProductPitched',inplace=True,axis=1)
df_tourism4['PreferredPropertyStar_codes'] = df_tourism4['PreferredPropertyStar'].map({'3 Star':1,'4 Star':2,'5 S
tar':3})
```

```
df_tourism4.drop('PreferredPropertyStar',inplace=True,axis=1)
df_tourism4['Designation_codes'] = df_tourism4['Designation'].map({'Executive':1,'Manager':2,'Senior Manager
':3,'AVP':4,'VP':5})
df_tourism4.drop('Designation',inplace=True,axis=1)

df_tourism4_cat = df_tourism4[categorical3]
df_tourism4_dummies = pd.get_dummies(df_tourism4_cat)
```

Let's check info() of data:

```
0   Age                                    4888 non-null   float64
 1   DurationOfPitch                        4888 non-null   float64
 2   NumberOfPersonVisited                  4888 non-null   int64
 3   NumberOfFollowups                      4888 non-null   float64
 4   NumberOfTrips                          4888 non-null   float64
 5   PitchSatisfactionScore                 4888 non-null   int64
 6   NumberOfChildrenVisited                4888 non-null   float64
 7   MonthlyIncome                          4888 non-null   float64
 8   ProductPitched_codes                   4888 non-null   int64
 9   PreferredPropertyStar_codes            4888 non-null   int64
 10  Designation_codes                      4888 non-null   int64
 11  ProdTaken_No                           4888 non-null   uint8
 12  ProdTaken_Yes                          4888 non-null   uint8
 13  PreferredLoginDevice_Company Invited   4888 non-null   uint8
 14  PreferredLoginDevice_Self Enquiry      4888 non-null   uint8
 15  CityTier_Tier-1                        4888 non-null   uint8
 16  CityTier_Tier-2                        4888 non-null   uint8
 17  CityTier_Tier-3                        4888 non-null   uint8
 18  Occupation_Free Lancer                 4888 non-null   uint8
 19  Occupation_Large Business              4888 non-null   uint8
 20  Occupation_Salaried                    4888 non-null   uint8
 21  Occupation_Small Business              4888 non-null   uint8
 22  Gender_Female                          4888 non-null   uint8
 23  Gender_Male                            4888 non-null   uint8
 24  MaritalStatus_Divorced                 4888 non-null   uint8
 25  MaritalStatus_Married                  4888 non-null   uint8
 26  MaritalStatus_Single                   4888 non-null   uint8
 27  MaritalStatus_Unmarried                4888 non-null   uint8
 28  Passport_No                            4888 non-null   uint8
 29  Passport_Yes                           4888 non-null   uint8
 30  OwnCar_No                              4888 non-null   uint8
 31  OwnCar_Yes                             4888 non-null   uint8
dtypes: float64(6), int64(5), uint8(21)
memory usage: 520.4 KB
```

Here we can see there are lot of new variables.

4. For clustering analysis we need to scale data because features are in different scale, is not allowed in clustering. So, here we will do scaling by StandarScaler( )
that are available in scikitlearn library.

5. Next, we will apply k_means algorithm for clustering. K-Means clustering is non-hierarchical clustering wherein initially we have to pre specified how many clusters we require before the model run.

6. Now, we will calculate WSS (within sum of square) for n number of clusters. Here we define range of clusters from 1 to 31.Then calculate inertia for each n n umber of clusters.

7. Let's see elbow curve: WSS plot for n number of clusters.

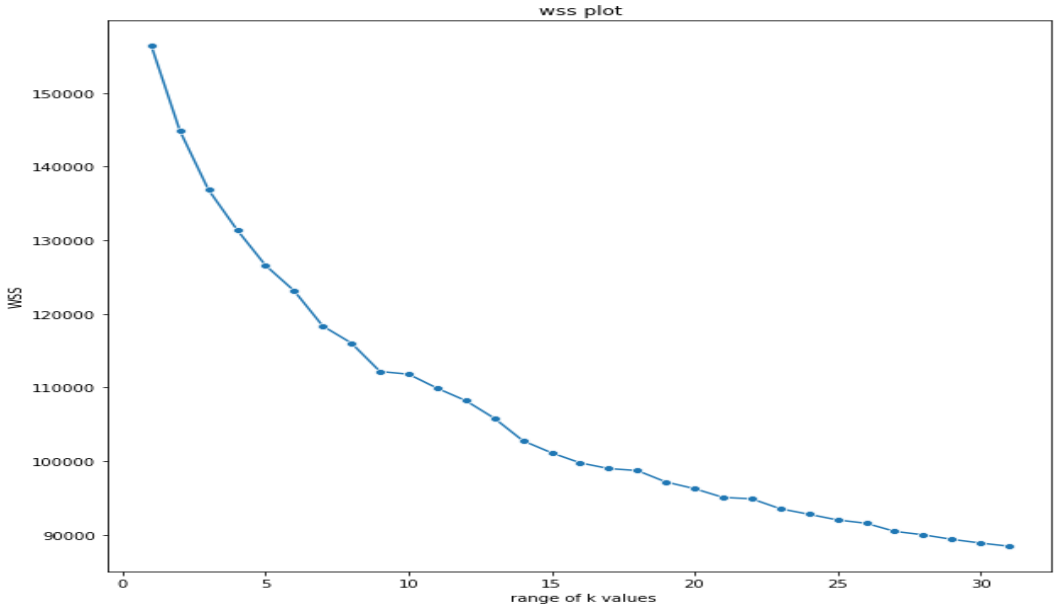

Fig: 18

Here we can see there is significant drop from 1 to 2, 2to3, 3 to4 and 4to5.After 5 very less. We can conclude, 4 and 5 could be optimal number of clusters.

8. Now, we will check silhouette score for each number of clusters. This is an in direct model evaluation technique that helps us to analyse whether each and eve ry observation that is mapped to cluster1,clauster2 and cluster3 is actually corre ct or not based on the distance criteria. Now we will check for what number of c lusters silhouette score is better. Is it 3 or 4? For which we will get silhouette sc ore is better , consider as an optimal number of cluster.

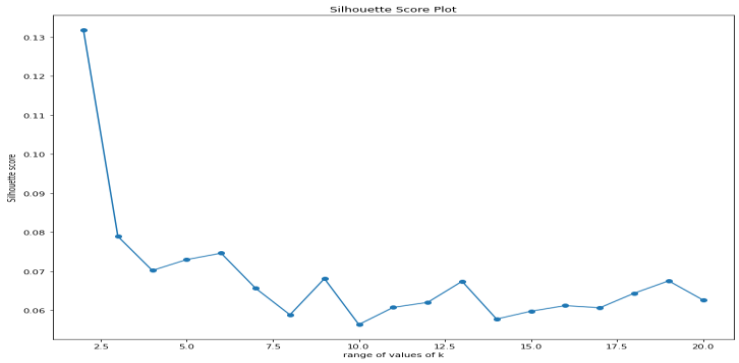

Fig: 1

From the above plot we can say silhoutte score for k=4 is0.07 (approximate) wh ich is better than k= 5 is (0.075).Hence k=5 is optimal number of clusters.

9.After getting the optimal number of clusters, we will append the clusters into df_tourism2 and df_tourism3 data set.

D:Let's see the plots of cluster vs Product taken and cluster vs Passport.

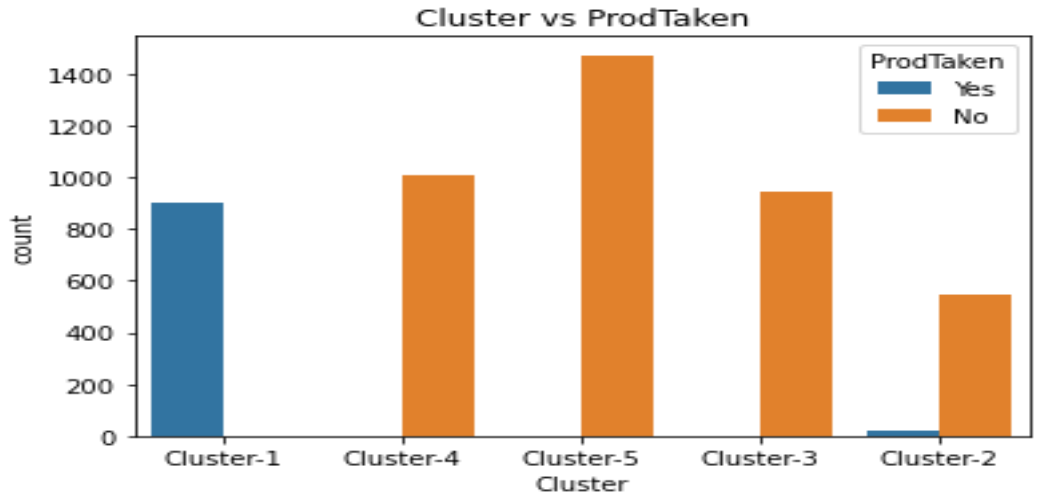

Fig: 22

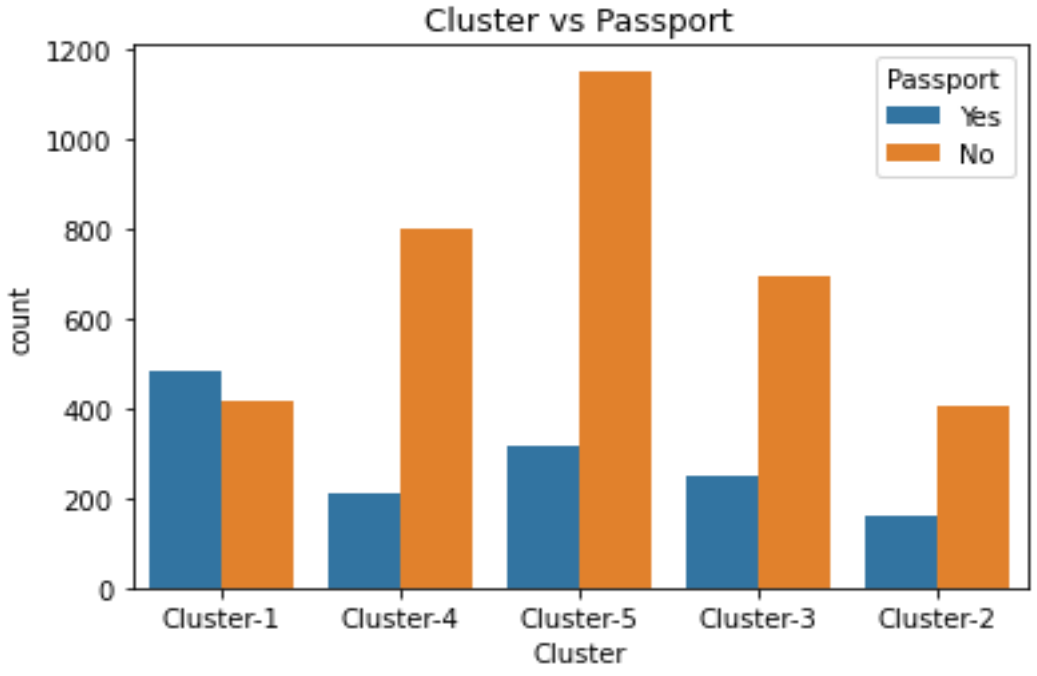

Fig: 23

E. Voting Classifier:

Voting classifier is a machine learning model that trains on ensemble of numerous models and predict an output (class) based on their highest probability of chosen class as the output.

It simply aggregates the findings of each classifier passed into voting classifier and predict the output class based on the highest majority of voting.

F. Stacking Classifier:

Stacking classifier is an ensemble learning technique to combine multiple classification models via meta-classifier. The individual classification models are trained on the complete training set, then the meta- classifier is fitted based on the outputs-meta-features-of the individual classification models in the ensemble. The meta- classifier can either be trained on the predicted class labels or probabilities from the ensemble.