CAPSTONE PROJECT

Social Media Tourism

Final Report

Submitted by,

Sindhu R Udupa.

BATCH: PGPDSBA.O. NOV22.B

Contents

List of Figures

List of Tables

1. Introduction.

In embarking on this project, our central focus was the formulation of a targeted digital advertising strategy tailored to the unique dynamics of an aviation company engaged in both domestic and international travel. Departing from conventional outreach approaches, we sought to leverage the power of a collaborative effort with a social networking platform. The crux of our endeavor was a comprehensive analysis of customers' digital and social behavior, aimed at optimizing the precision and efficiency of digital advertisements. The ultimate goal was clear: reach users most likely to engage with and make purchases from the diverse range of services offered by the aviation company. To achieve this, we strategically segmented our approach based on the distinct characteristics of customers' login devices, leading us to create two separate models—one for laptops and another for mobile devices. This meticulous segmentation was imperative to ensure the utmost accuracy in our digital advertising endeavors, given the inherent cost considerations associated with the chosen platform.

Our project was spurred by a crucial need within the aviation company's marketing strategy—an evolution from traditional tele-calling methods to the precision and efficiency offered by targeted digital advertising. This transition not only yielded cost-effective benefits but also empowered the company to allocate advertising budgets with heightened efficiency. Importantly, the project facilitated personalized advertising grounded in a profound understanding of customers' digital and social behavior, culminating in a tangible improvement in conversion rates. The nuanced approach of creating distinct models for laptops and mobile devices was driven by the recognition that buying propensity varies based on the device used. In the fiercely competitive aviation industry, this strategic pivot not only positioned the company for a competitive edge but also significantly heightened customer engagement, thereby bolstering overall sales.

Beyond its immediate business implications, our project unfolded as a multidimensional opportunity. It seamlessly aligned with the company's broader objectives, promising increased sales, cost savings, and a more formidable position in the highly competitive aviation industry. Simultaneously, from a societal perspective, the project held considerable significance. By enhancing the customer experience through personalized and non-intrusive advertising, it addressed a social need—reducing the burden of irrelevant content for users. The long-term impact of the project extended beyond mere business growth; it promised positive societal outcomes. The endeavor harmonized with contemporary industry trends, showcasing the aviation company as an innovator and a socially responsible entity. In this light, the project emerged not merely as a tactical marketing initiative but as a strategic endeavor with dual implications for both business success and the well-being of society at large.

The data was collected from 11760 users with various characteristics. It includes information on whether they have taken a product (e.g., a travel package), their yearly average views on travel pages, preferred devices, total likes given and received on outstation check-ins, yearly average outstation check-ins, family membership status, preferred location type, yearly average comments on travel pages, and more. The data appears to encompass both personal and behavioral aspects, allowing for in-depth analysis to understand user preferences and behaviors related to travel and online activities. From the visual inspection of the data, we understand the following.

i. The data contains 11760 rows and 17 columns.
ii. There are no duplicates present in the data.
iii. There are missing values in few columns. Details of which are given below:

UserID	0
Taken_product	0
Yearly_avg_view_on_travel_page	581
preferred_device	53
total_likes_on_outstation_checkin_given	381
yearly_avg_Outstation_checkins	75
member_in_family	0
preferred_location_type	31
Yearly_avg_comment_on_travel_page	206
total_likes_on_outofstation_checkin_received	0
week_since_last_outstation_checkin	0
following_company_page	103
montly_avg_comment_on_company_page	0
working_flag	0
travelling_network_rating	0
Adult_flag	0
Daily_Avg_mins_spend_on_traveling_page	0

Table 1: Missing values in the data.

iv. Snippets of first and last five rows of the data are given below:

	UserID	Taken_product	Yearly_avg_view_on_travel_page	preferred_device	total_likes_on_outstation_checkin_given	yearly_avg_Outstation_checkins	member_
0	1000001	Yes	307.0	iOS and Android	38570.0	1	
1	1000002	No	367.0	iOS	9765.0	1	
2	1000003	Yes	277.0	iOS and Android	48055.0	1	
3	1000004	No	247.0	iOS	48720.0	1	
4	1000005	No	202.0	iOS and Android	20685.0	1	

Table 2: First five records of the data.

	UserID	Taken_product	Yearly_avg_view_on_travel_page	preferred_device	total_likes_on_outstation_checkin_given	yearly_avg_Outstation_checkins	mem
11755	1011756	No	279.0	Laptop	30987.0	23	
11756	1011757	No	305.0	Tab	21510.0	6	
11757	1011758	No	214.0	Tab	5478.0	4	
11758	1011759	No	382.0	Laptop	35851.0	2	
11759	1011760	No	270.0	Tab	22025.0	8	

Table 3: Last five records of the data.

v. There are 10 numerical and 7 categorical columns in the data.

```
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   UserID                                  11760 non-null  int64
 1   Taken_product                           11760 non-null  object
 2   Yearly_avg_view_on_travel_page          11179 non-null  float64
 3   preferred_device                        11707 non-null  object
 4   total_likes_on_outstation_checkin_given 11379 non-null  float64
 5   yearly_avg_Outstation_checkins          11685 non-null  object
 6   member_in_family                        11760 non-null  object
 7   preferred_location_type                 11729 non-null  object
 8   Yearly_avg_comment_on_travel_page       11554 non-null  float64
 9   total_likes_on_outofstation_checkin_received 11760 non-null  int64
 10  week_since_last_outstation_checkin      11760 non-null  int64
 11  following_company_page                  11657 non-null  object
 12  montly_avg_comment_on_company_page      11760 non-null  int64
 13  working_flag                            11760 non-null  object
 14  travelling_network_rating               11760 non-null  int64
 15  Adult_flag                              11760 non-null  int64
 16  Daily_Avg_mins_spend_on_traveling_page  11760 non-null  int64
dtypes: float64(3), int64(7), object(7)
memory usage: 1.5+ MB
```

Table 4: Information about the data.

The table provided highlights an issue in the data structure where two columns, 'yearly average outstation check-ins' and 'members in the family,' were intended to be numerical variables but are erroneously categorized as object data types. Additionally, the 'adult flag' and 'travel network rating' columns, which were meant to be categorical, have been incorrectly identified as numerical variables. This discrepancy underscores the necessity for data cleaning and standardization to ensure the dataset's accuracy and adherence to the defined data types, a crucial step in maintaining data integrity.

vi. Statistical description of the data is given below.

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
UserID	11760.0	NaN	NaN	NaN	1005880.5	3394.963917	1000001.0	1002940.75	1005880.5	1008820.25	1011760.0
Taken_product	11760	2	No	9864	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Yearly_avg_view_on_travel_page	11179.0	NaN	NaN	NaN	280.830844	68.182958	35.0	232.0	271.0	324.0	464.0
preferred_device	11707	10	Tab	4172	NaN	NaN	NaN	NaN	NaN	NaN	NaN
total_likes_on_outstation_checkin_given	11379.0	NaN	NaN	NaN	28170.481765	14385.032134	3570.0	16380.0	28076.0	40525.0	252430.0
yearly_avg_Outstation_checkins	11685	30	1	4543	NaN	NaN	NaN	NaN	NaN	NaN	NaN
member_in_family	11760	7	3	4561	NaN	NaN	NaN	NaN	NaN	NaN	NaN
preferred_location_type	11729	15	Beach	2424	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Yearly_avg_comment_on_travel_page	11554.0	NaN	NaN	NaN	74.790029	24.02665	3.0	57.0	75.0	92.0	815.0
total_likes_on_outofstation_checkin_received	11760.0	NaN	NaN	NaN	6531.699065	4706.613785	1009.0	2940.75	4948.0	8393.25	20065.0
week_since_last_outstation_checkin	11760.0	NaN	NaN	NaN	3.203571	2.616365	0.0	1.0	3.0	5.0	11.0
following_company_page	11657	4	No	8355	NaN	NaN	NaN	NaN	NaN	NaN	NaN
montly_avg_comment_on_company_page	11760.0	NaN	NaN	NaN	28.661565	48.660504	11.0	17.0	22.0	27.0	500.0
working_flag	11760	2	No	9952	NaN	NaN	NaN	NaN	NaN	NaN	NaN
travelling_network_rating	11760.0	NaN	NaN	NaN	2.712245	1.080887	1.0	2.0	3.0	4.0	4.0
Adult_flag	11760.0	NaN	NaN	NaN	0.793878	0.851823	0.0	0.0	1.0	1.0	3.0
Daily_Avg_mins_spend_on_traveling_page	11760.0	NaN	NaN	NaN	13.817432	9.070657	0.0	8.0	12.0	18.0	270.0

Table 5: Statistical Information of the data.

The data dictionary is given below. We are operating under the assumption that the 'buy_ticket' mentioned in the data dictionary corresponds to the 'taken_product' column in the dataset. This assumption forms the basis for our data analysis and interpretation.

Given that all variable names within the dataset are readily comprehensible and aligned with their respective attributes, we do not intend to alter or replace any of these variable names. This decision is based on the clarity and relevance of the existing nomenclature, which facilitates transparent data interpretation and analysis.

Variable	Description
UserID	Unique ID of user
Buy_ticket	Buy ticket in next month
Yearly_avg_view_on_travel_page	Average yearly views on any travel related page by user
preferred_device	Through which device user preferred to do login
total_likes_on_outstation_checkin_given	Total number of likes given by a user on out of station checkings in last year
yearly_avg_Outstation_checkins	Average number of out of station check-in done by user
member_in_family	Total number of relationship mentioned by user in the account
preferred_location_type	Preferred type of the location for travelling of user
Yearly_avg_comment_on_travel_page	Average yearly comments on any travel related page by user
total_likes_on_outofstation_checkin_received	Total number of likes received by a user on out of station checkings in last year
week_since_last_outstation_checkin	Number of weeks since last out of station check-in update by user
following_company_page	Weather the customer is following company page (Yes or No)
montly_avg_comment_on_company_page	Average monthly comments on company page by user
working_flag	Weather the customer is working or not
travelling_network_rating	Does user have close friends who also like travelling. 1 is highs and 4 is lowest
Adult_flag	Weather the customer is adult or not
Daily_Avg_mins_spend_on_traveling_page	Average time spend on the company page by user on daily basis

Table 6: Data dictionary.

The categorical variables contain the following values:

```
Value counts for Taken_product:
No     9864
Yes    1896
Name: Taken_product, dtype: int64
```

```
Value counts for preferred_device:
Tab                4172
iOS and Android    4134
Laptop             1108
iOS                1095
Mobile              600
Android             315
Android OS          145
ANDROID             134
Other                 2
Others                2
Name: preferred_device, dtype: int64
```

```
Value counts for preferred_location_type:
Beach              2424
Financial          2409
Historical site    1856
Medical            1845
Other               643
Big Cities          636
Social media        633
Trekking            528
Entertainment       516
Hill Stations       108
Tour  Travel         60
Tour and Travel      47
Game                 12
OTT                   7
Movie                 5
Name: preferred_location_type, dtype: int64
```

```
Value counts for following_company_page:
No     8355
Yes    3285
1        12
0         5
Name: following_company_page, dtype: int64
```

```
Value counts for travelling_network_rating:
3    3672
4    3456
2    2424
1    2208
Name: travelling_network_rating, dtype: int64
```

```
Value counts for working_flag:
No     9952
Yes    1808
Name: working_flag, dtype: int64
```

```
Value counts for Adult_flag:
0    5048
1    4768
2    1264
3     680
Name: Adult_flag, dtype: int64
```

Table 7: Value counts of different categorical variables.

2. Exploratory Data Analysis

Univariate Analysis:

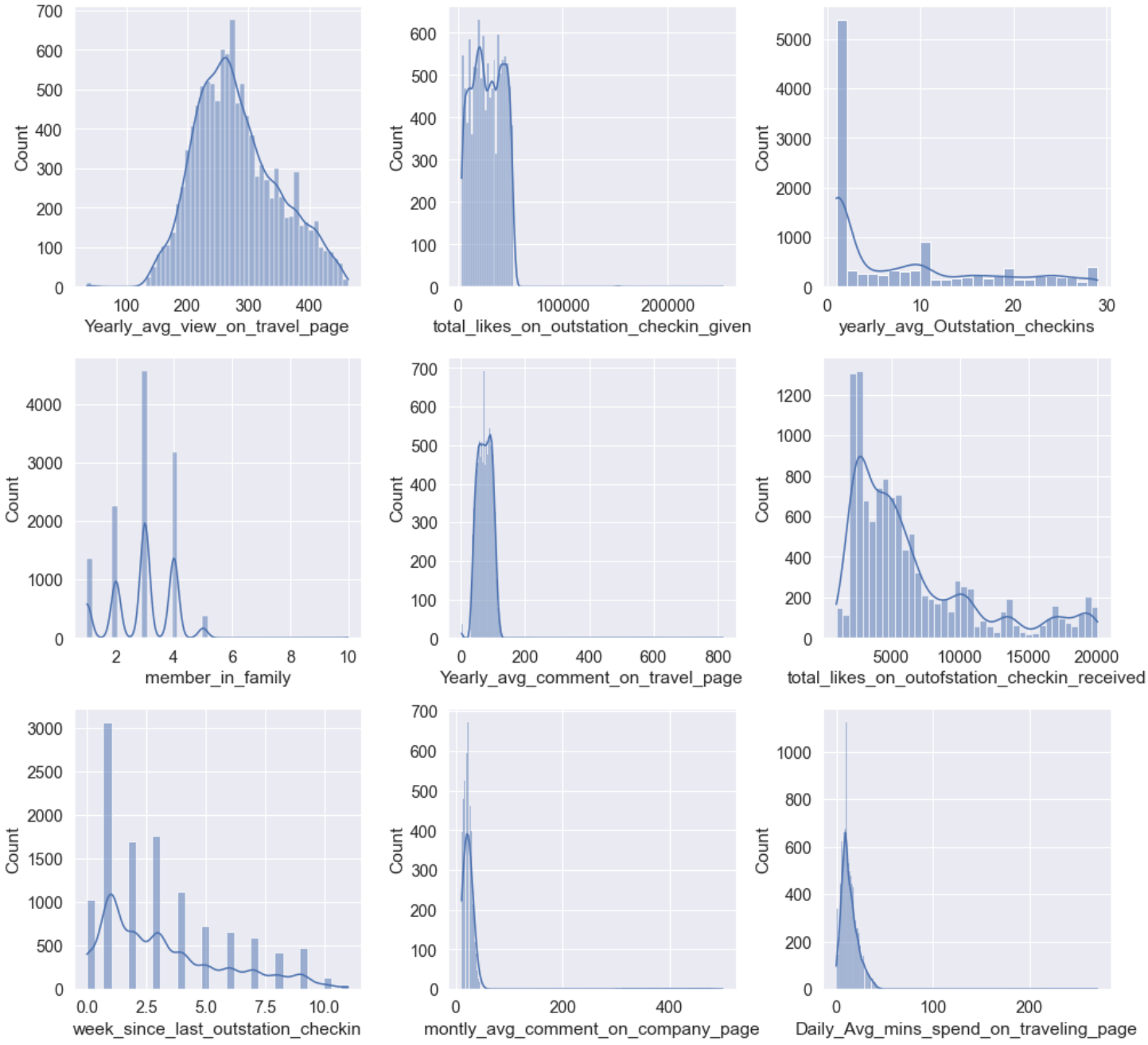

Fig 1: Histograms of all continuous variables.

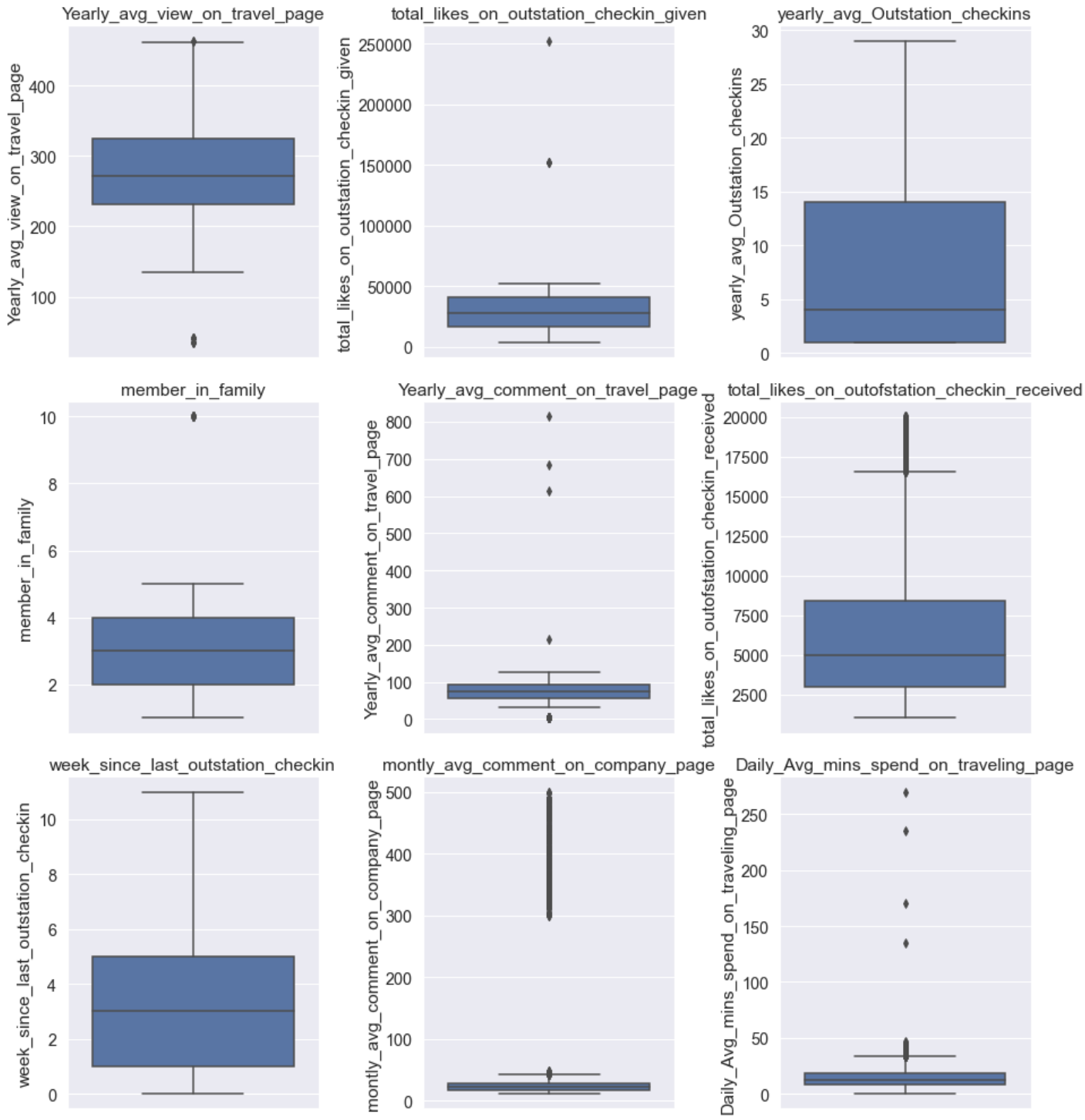

Fig 2: Box plots of all continuous variables.

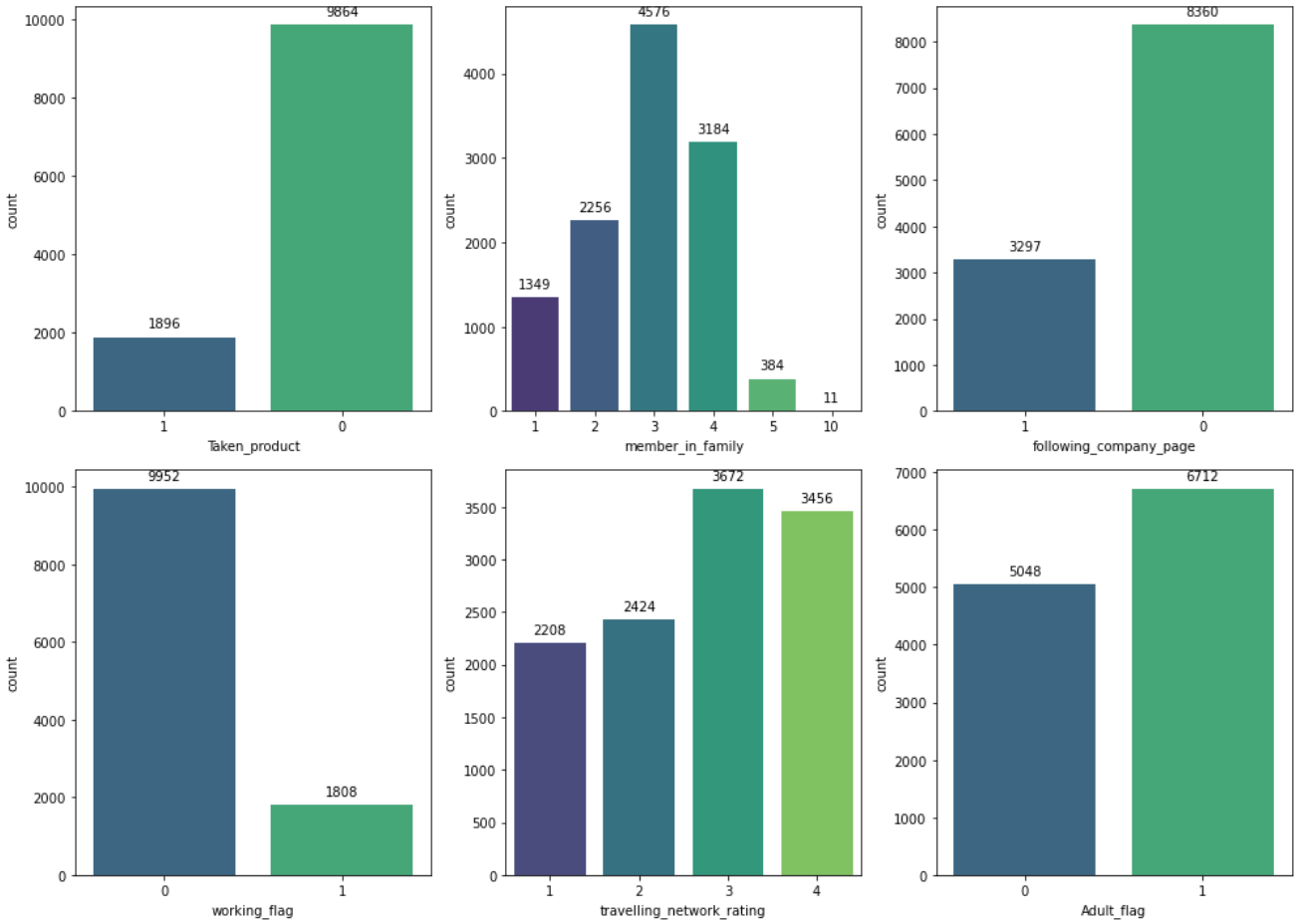

Fig 3: Count plots of categorical variables.

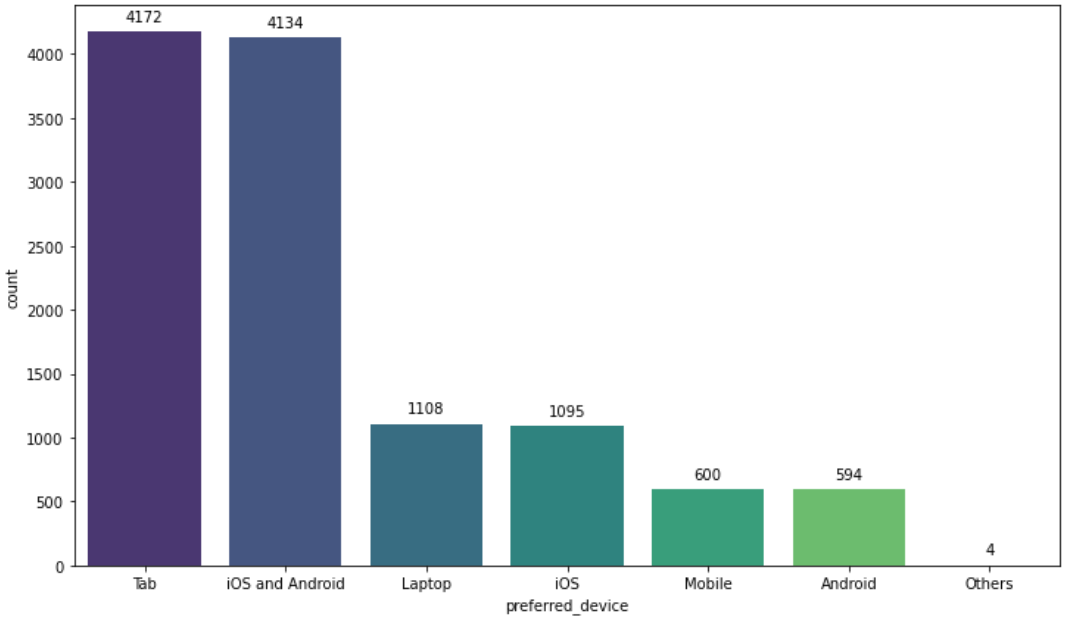

Fig 4: Count plot of preferred devices.

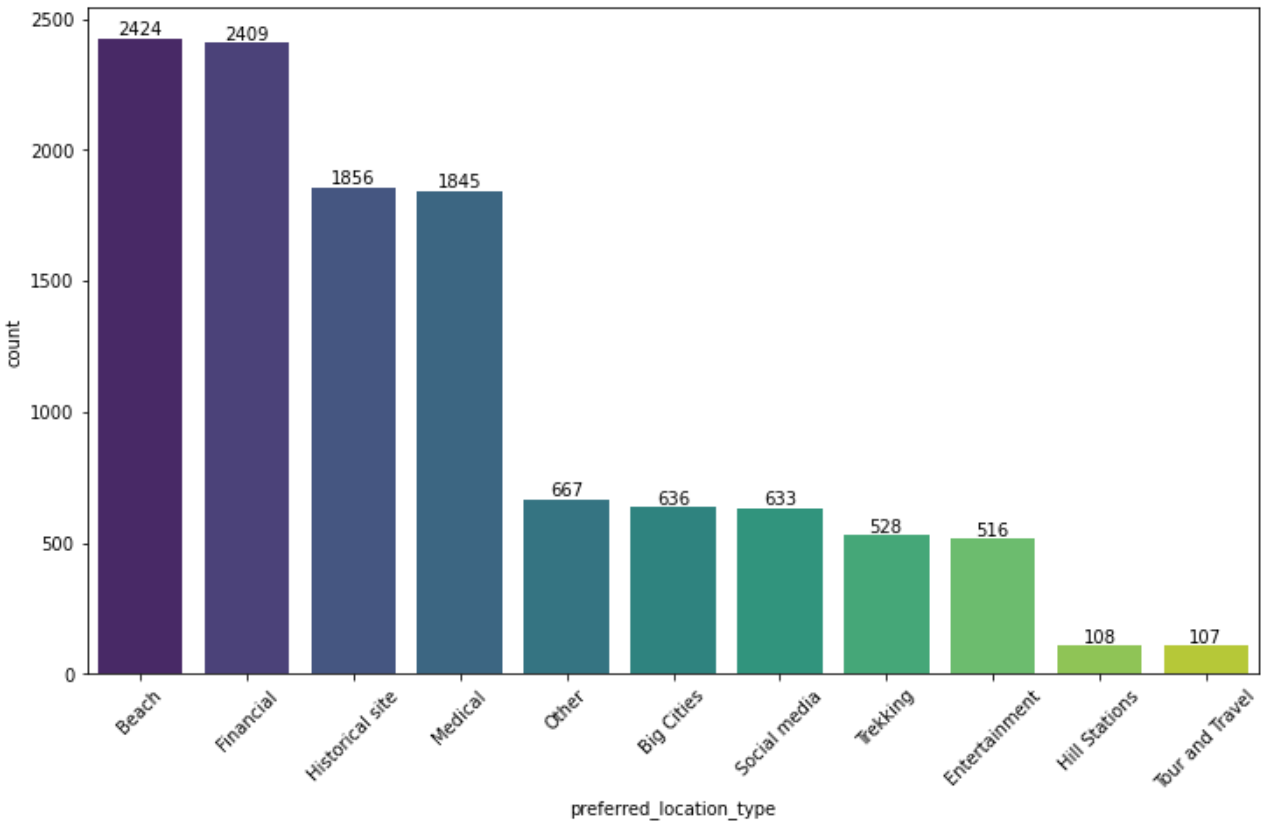

Fig. 5: Count plot of Preferred location types.

Inference:

In the descriptive analysis of the dataset, it is observed that the 'yearly average view on travel page' exhibits a slight normal distribution, indicating a relatively balanced distribution of data. On the other hand, 'yearly average outstation check-ins' appears to be left-skewed, suggesting a concentration of lower values. For 'members in the family,' as it represents a discrete variable, the application of kernel density estimation (KDE) is not suitable.

In the boxplots, several variables exhibit outliers. 'Yearly average view on travel page' displays lower outliers, while 'total likes on outstation check-in given,' 'members in the family,' 'total likes on outstation check-in received,' 'monthly average comment on company page,' and 'daily average minutes spent on traveling page' show upper outliers. 'Week since last outstation check-in' and 'yearly average outstation check-ins' do not exhibit any outliers.

In the categorical variables, it is noteworthy that a relatively small proportion of customers have purchased the product, with only 1896 individuals having done so, as opposed to 9864 individuals who have not made a purchase. 'Members in the family' predominantly contains the value '3.' A significant number of customers do not follow the company page. Furthermore, there is a notable disparity in the usage of laptops compared to mobile devices, with mobile usage far surpassing laptop usage. Regarding location types, 'beaches' rank as the most popular, followed by 'financial locations' and 'historical sites,' while 'game,' 'OTT,' and 'movies' represent the least popular choices among customers.

Bivariate Analysis:

In the presented graph, we are observing the behavior of our target variable, ‘taken_product’, concerning the independent continuous variables. This visual representation serves to elucidate the relationship between the target variable and various continuous predictors, aiding in the exploration of patterns and correlations within the dataset.

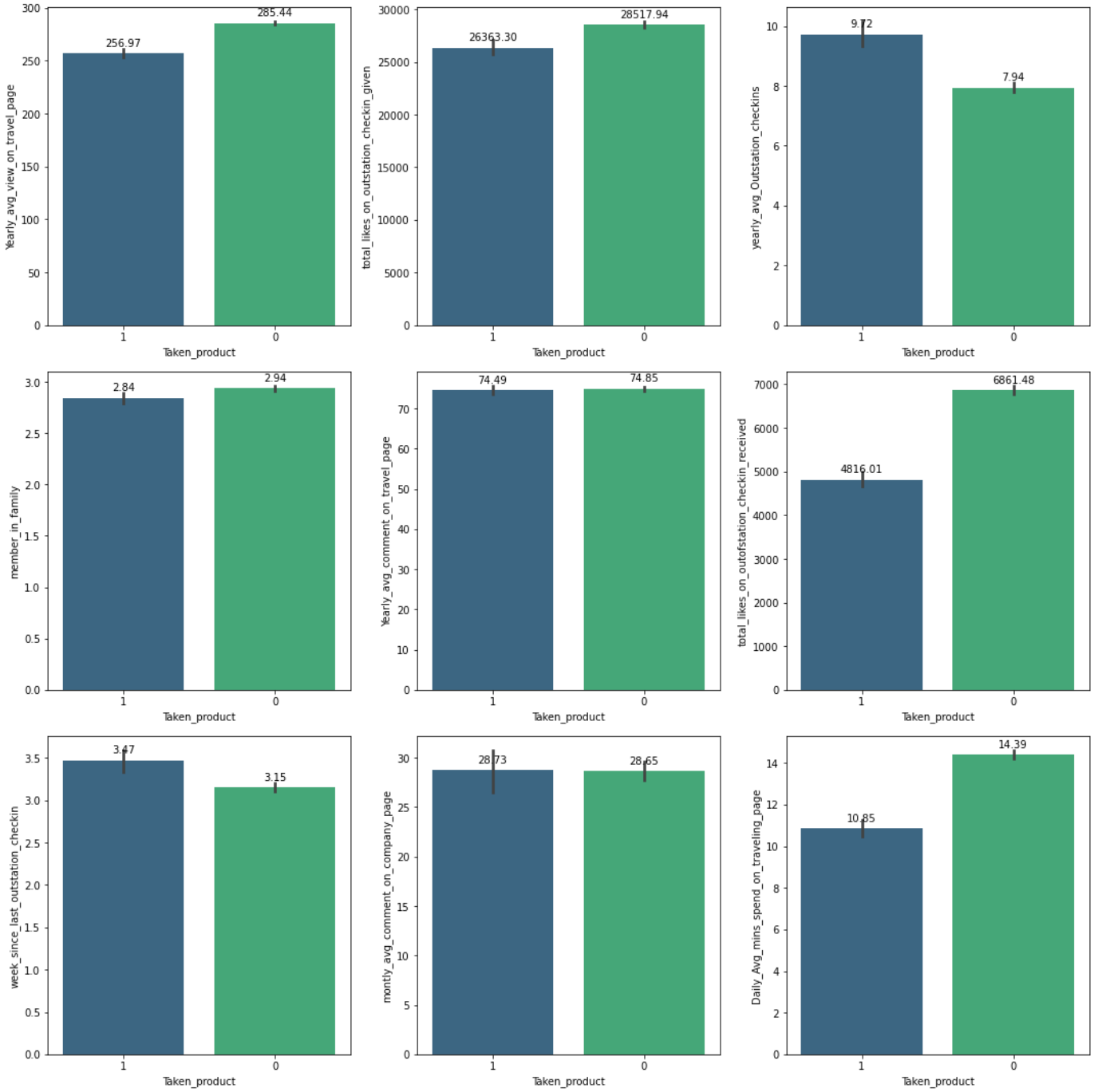

Fig. 6: Behavior of the target variable with respect to the continuous variables.

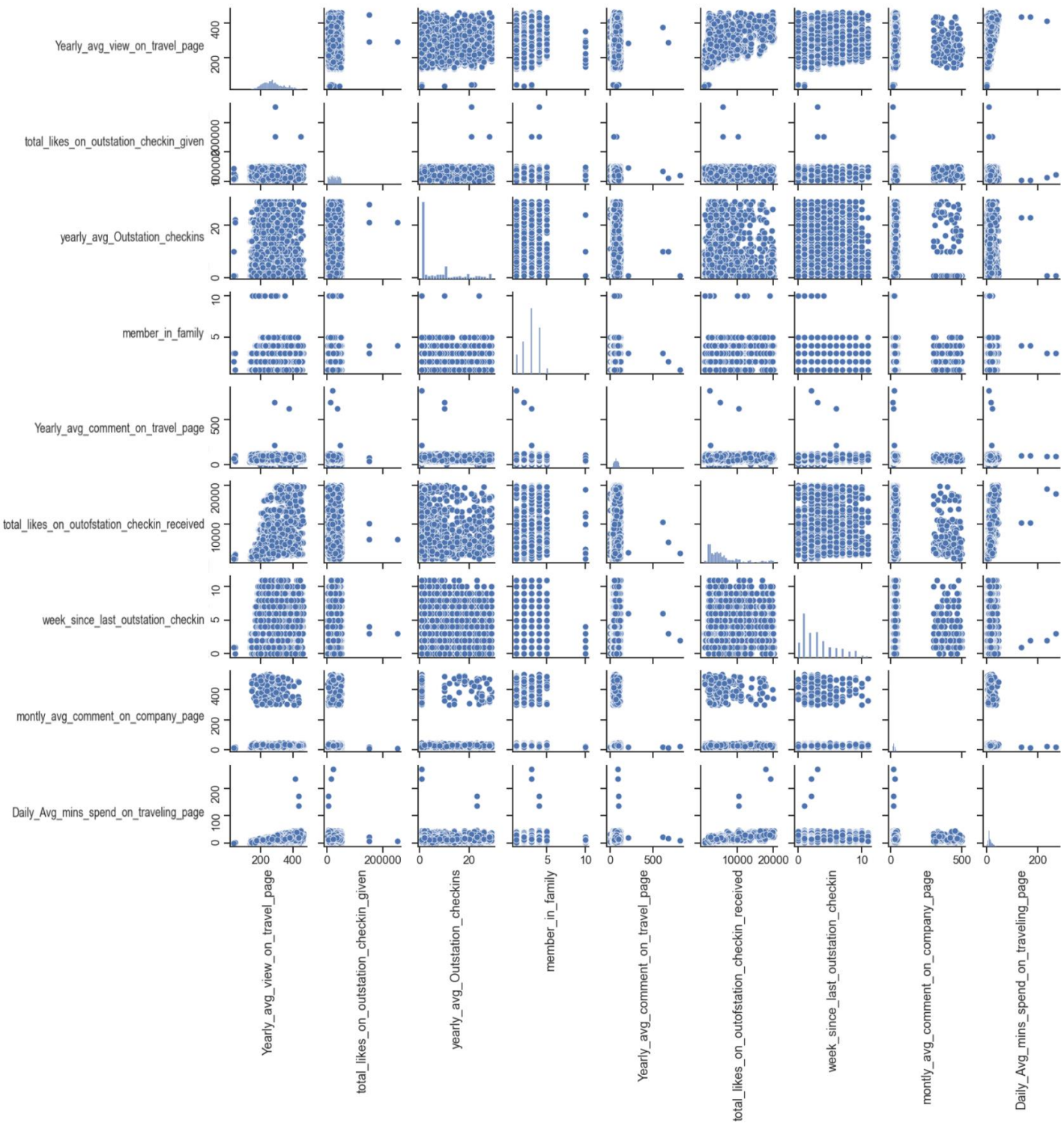

Fig. 7: Pair plot of all numerical variables.

Fig. 8: Heat map of correlation matrix.

Inference:

During the bivariate analysis, the effectiveness of different variables in distinguishing between '0s' and '1s' in the target column was examined through bar plots. Notably, 'total likes on outstation check-in received,' 'yearly average outstation check-ins,' and 'daily average minutes spent on the traveling page' stood out as strong indicators, effectively differentiating the two classes. In particular, it was evident that individuals spending more time on the travel page were less likely to purchase holiday packages. Conversely, the count of comments posted on travel or company pages appeared to be a weak predictor, as it showed similar patterns for both classes. Additionally, those receiving fewer likes on outstation check-ins were more inclined to buy tickets, while individuals with higher yearly average check-ins demonstrated a higher likelihood of making a purchase.

In contrast, the pair plots presented a complex and somewhat unclear picture, suggesting a lack of obvious correlations among the variables. To gain more clarity, a correlation matrix was visualized using a heatmap. Within this heatmap, a notable positive correlation was observed between 'total likes on outstation check-in received' and 'yearly average view on travel page.' Similarly, 'daily average minutes spent on the traveling page' exhibited correlations with 'yearly average view on travel page' and 'total likes received on outstation check-ins.' In contrast, the majority of other variables displayed limited correlations with each other.

3. Data Cleaning and pre-processing.

In this phase, several transformations and corrections were applied to the dataset to enhance its quality and usability:

- The 'yearly_avg_Outstation_checkins' column originally contained a special character, denoted by an asterisk (*). This special character has been addressed and replaced with 'NaN' (Not a Number) to signify missing or undefined data.
- 'member_in_family' Column: The values 'Three' were replaced with '3' to ensure numerical consistency. Subsequently, the 'member_in_family' column was converted to a numerical data type for analytical purposes.
- 'following_company_page' Column: In the 'following_company_page' column, 'No' was replaced with '0,' and 'Yes' was replaced with '1,' effectively converting it into a binary representation to reflect whether customers follow the company page.
- 'Adult_flag' Column: In the 'Adult_flag' column, values other than '0' were uniformly replaced with '1,' essentially indicating the presence of an adult customer.
- 'preferred_device' Column: In the 'preferred_device' column, variations in device labels such as 'Android OS,' 'ANDROID,' and 'Other' were standardized to 'Android' and 'Others' for consistency and clarity.
- 'preferred_location_type' Column: The value 'Tour Travel' in the 'preferred_location_type' column was corrected to 'Tour and Travel' for accuracy.
- 'working_flag' Column: Similar to 'following_company_page,' 'No' was replaced with '0,' and 'Yes' was replaced with '1' in the 'working_flag' column, signifying whether a customer is employed.
- 'Taken_product' Column: In the 'Taken_product' column, 'No' was replaced with '0,' and 'Yes' was replaced with '1,' denoting whether a customer has purchased the product.

These data preprocessing steps were essential to ensure uniformity and accuracy in the dataset, facilitating meaningful analysis and interpretation.

Removal of unwanted variables:

The 'User ID' column was removed from the table as it was found to be non-contributory and lacked any meaningful purpose in the analysis.

Missing Value Treatment:

As indicated in Table 1, our dataset initially contained 1430 missing values. Subsequently, special characters were replaced with null values, resulting in 1431 null values within the dataset. The number of missing values in each column is provided in the table below.

Taken_product	0
Yearly_avg_view_on_travel_page	581
preferred_device	53
total_likes_on_outstation_checkin_given	381
yearly_avg_Outstation_checkins	76
member_in_family	0
preferred_location_type	31
Yearly_avg_comment_on_travel_page	206
total_likes_on_outofstation_checkin_received	0
week_since_last_outstation_checkin	0
following_company_page	103
montly_avg_comment_on_company_page	0
working_flag	0
travelling_network_rating	0
Adult_flag	0
Daily_Avg_mins_spend_on_traveling_page	0

dtype: int64

Table 8: Missing values in the data after special character treatment.

It's important to note the distribution of missing values across columns, and in the context of our dataset, which comprises 188,160 entries, the proportion of missing data stands at a mere 0.76%. Given this minimal percentage of missing values, the choice of imputation method becomes less critical. Therefore, we opt for a straightforward approach, replacing null values in numerical columns with the median to account for potential outliers. In categorical columns, null values are replaced with the mode. This pragmatic imputation strategy aligns with the dataset's characteristics and the limited impact of missing values, ensuring data completeness and analytical integrity.

Outlier Treatment.

As evident from the boxplot visualization, seven columns in the dataset exhibit outliers. To enhance the robustness of our data while preserving its original integrity to a significant extent, we have chosen to cap the values at the 95th and 5th percentiles. This approach ensures that extreme values are adjusted, making the data more representative without compromising the overall dataset structure. Addressing outliers is crucial when using models like Logistic Regression, Decision Trees, etc., due to their sensitivity. Careful outlier treatment ensures the robustness and accuracy of these models, enhancing overall analysis reliability. Fig. 2 shows the visualization of variables before the outlier treatment. The visualization of the variables after the outlier treatment is given below.

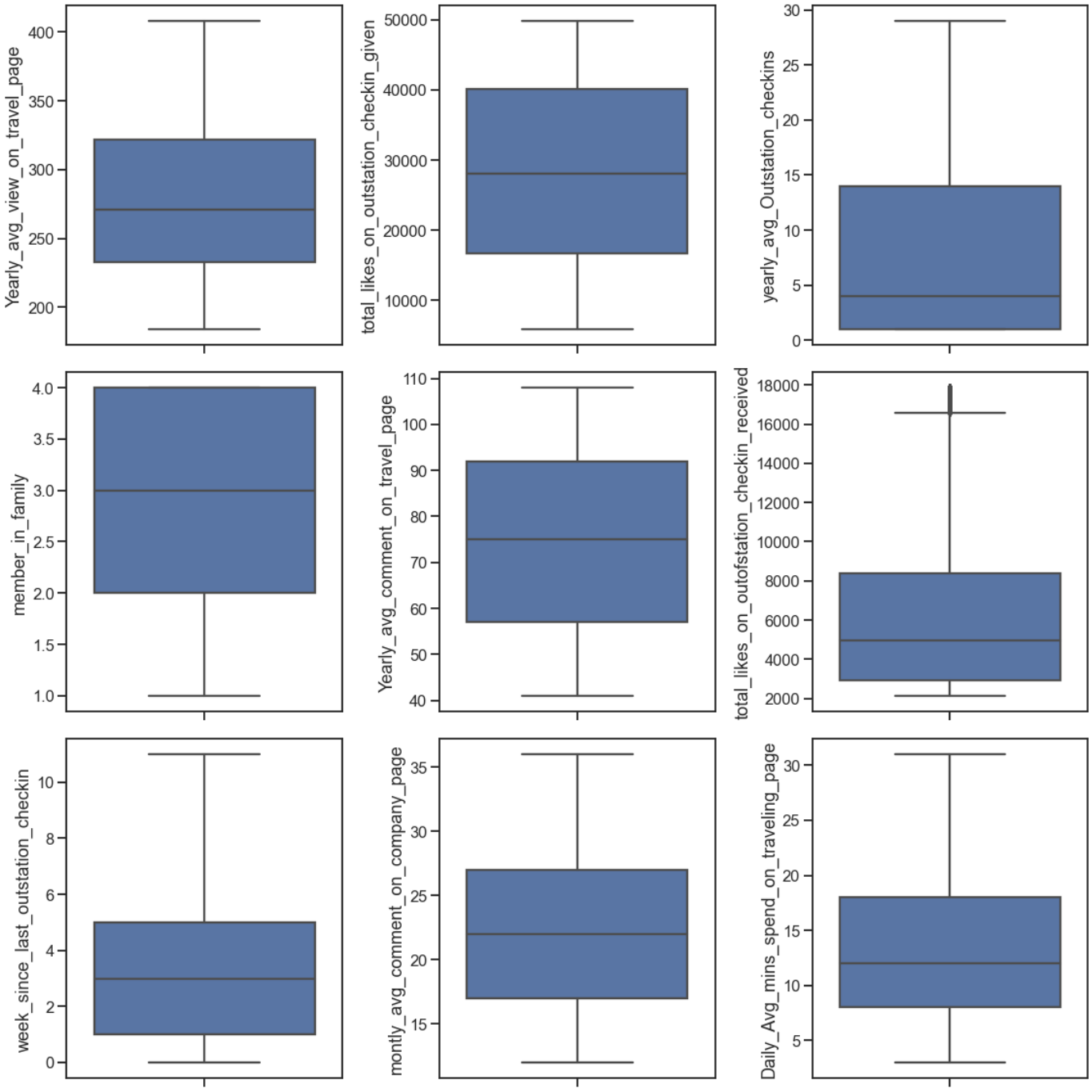

Fig. 10: Boxplots after outlier treatment.

Even following the treatment of outliers in our data, one particular boxplot continues to exhibit outliers. This occurrence can be attributed to our chosen approach of capping values at the 95th percentile rather than adhering strictly to the whiskers, which are typically used in boxplots as the boundary for identifying outliers. The decision to cap values at the 95th percentile was made to mitigate the impact of extreme values while still retaining the original data structure. Consequently, this approach may still reveal some values beyond the whiskers, which are visually interpreted as outliers in the boxplot.

Variable Transformation:

First, we standardized the numerical columns in the dataset using the StandardScaler from scikit-learn. Scaling plays a vital role, since we are employing distance-based machine learning models such as K-Nearest Neighbors (KNN). It ensures fair consideration of features by preventing undue influence based on varying scales. Within the 'preferred_device', values that are not equal to 'Laptop' are replaced with 'Mobile' This simplifies the 'preferred_device' column, categorizing it into 'Laptop' and 'Mobile' as asked in the problem statement. Then, we performed one-hot encoding for the 'preferred_device' and 'preferred_location_type' columns. This generates new binary columns for each category within these columns, allowing for categorical data to be represented numerically.

The resulting data now contains the scaled, transformed, and one-hot encoded data, making it suitable for analytical and modeling purposes. The snippet of the data is given below.

	Taken_product	Yearly_avg_view_on_travel_page	total_likes_on_outstation_checkin_given	yearly_avg_Outstation_checkins	member_in_family	Yearly_avg_c
0	1	0.427409	0.766137	-0.831881	-0.910174	
1	0	1.390536	-1.344218	-0.831881	-1.940858	
2	1	-0.054154	1.461041	-0.831881	-0.910174	
3	0	-0.535717	1.509761	-0.831881	1.151193	
4	0	-1.258062	-0.544181	-0.831881	-1.940858	
...	...	...	...	...	...	
11755	0	-0.022050	0.210580	1.711300	-0.910174	
11756	0	0.395305	-0.483738	-0.253886	-1.940858	
11757	0	-1.065437	-1.626004	-0.485084	0.120509	
11758	0	1.631318	0.566933	-0.716282	0.120509	
11759	0	-0.166519	-0.446008	-0.022687	0.120509	

11760 rows × 28 columns

Table 9: Scaled and encoded data.

4. Model Building.

Before commencing the model-building process, it's important to note that we've initially divided the dataset into two distinct subsets. One subset encompasses data where the customer's preferred device is a mobile device, while the other subset comprises data from customers who prefer using laptops.

In the initial stages of model development, we perform key preparatory steps. This includes the segregation of the dataset into predictors and the target variable ('taken_product'). Subsequently, utilizing sklearn's model selection tools, we conduct a train-test split, dividing the predictors and the target into training and testing sets with a 70:30 ratio. To maintain a balanced representation of customers who have taken the product or not, we implement the 'stratify' function, ensuring an equitable distribution in both the training and testing datasets.

The details of the train and test data are given below:

	Yearly_avg_view_on_travel_page	total_likes_on_outstation_checkin_given	yearly_avg_Outstation_checkins	member_in_family	Yearly_avg_comment_on_travel
1533	-0.150467	1.589549	1.942498	0.120509	1.1
6304	-1.450688	-0.282264	-0.600683	-0.910174	-0.2
7575	-0.214675	1.384920	2.173697	-1.940858	-0.9
7307	-0.969124	0.805333	-0.022687	-0.910174	-0.1
4061	-0.134415	0.209774	-0.831881	1.151193	0.5

Table 10: Snippet of train data of predictors (Mobile)

	Yearly_avg_view_on_travel_page	total_likes_on_outstation_checkin_given	yearly_avg_Outstation_checkins	member_in_family	Yearly_avg_comment_on_travel
2472	-0.294936	-1.546792	1.133304	-0.910174	-0.7
5604	1.165806	1.084760	-0.831881	1.151193	-1.0
382	-1.370427	-0.585208	-0.831881	-1.940858	-0.0
9532	-0.696238	-1.317184	-0.716282	0.120509	1.6
9492	-0.086258	0.128964	-0.369485	1.151193	1.2

Table 11: Snippet of test data of predictors (Mobile)

```
2472    0
5604    0
382     1
9532    0
9492    0
Name: Taken_product, dtype: int64
```

Table 12: Train data of target (Mobile)

```
2472    0
5604    0
382     1
9532    0
9492    0
Name: Taken_product, dtype: int64
```

Table 13: Test data of target (Mobile)

The data subsets exhibit the following dimensions: X_train consists of 7,456 rows and 23 columns, X_test comprises 3,196 rows and 23 columns, y_train includes 7,456 rows, and y_test encompasses 3,196 rows. These specifications delineate the size and structure of the training and testing datasets, essential for subsequent stages in the model development process.

The same procedure is followed for the other dataset, where the login device is laptop.

	Yearly_avg_view_on_travel_page	total_likes_on_outstation_checkin_given	yearly_avg_Outstation_checkins	member_in_family	Yearly_avg_comment_on_travel_
564	-0.102310	0.131089	1.133304	1.151193	0.89
634	0.218732	1.589549	-0.716282	-0.910174	-0.67
193	1.599213	1.589549	2.289296	-0.910174	1.38
203	-1.033333	1.589549	-0.831881	0.120509	1.28
690	0.298992	0.366338	1.595701	0.120509	1.33

Table 14: Snippet of train data of predictors (Laptop)

	Yearly_avg_view_on_travel_page	total_likes_on_outstation_checkin_given	yearly_avg_Outstation_checkins	member_in_family	Yearly_avg_comment_on_travel
588	-0.278884	0.685474	-0.716282	-1.940858	-0.4
798	-0.198623	-1.456165	0.092912	1.151193	1.5
1041	2.048672	0.635655	0.324110	1.151193	1.6
514	-0.712291	1.305063	-0.716282	1.151193	0.6
477	0.668191	-0.141012	0.092912	-0.910174	0.2

Table 15: Snippet of test data of predictors (Laptop)

```
564    0
634    0
193    1
203    1
690    0
Name: Taken_product, dtype: int64
```

Table 16: Train data of target (Laptop)

```
588     0
798     1
1041    0
514     0
477     0
Name: Taken_product, dtype: int64
```

Table 17: Test data of target (Laptop)

The data subsets exhibit the following dimensions: X_train comprises 775 rows and 23 columns, while X_test consists of 333 rows and 23 columns. Additionally, y_train contains 775 rows, and y_test comprises 333 rows. These dimensions provide a snapshot of the dataset's structure, essential for model training and evaluation. Various classification models, including Linear Regression, LDA, Naive Bayes, KNN, Decision Trees, Random Forest, Bagging, AdaBoost, and Gradient Boosting, were constructed and fine-tuned using grid search cross-validation*.

*Refer to Appendix for more details on all the models.

Model	Train Accuracy	Test Accuracy	Train Precision	Test Precision	Train Recall	Test Recall	Train F1 Score	Test F1 Score	Train AUC	Test AUC
Logistic Regression	87.06%	86.92%	76.66%	73.61%	21.43%	21.81%	33.49%	33.65%	78.99%	79.85%
LDA	86.86%	87.02%	69.85%	70.06%	23.90%	25.51%	35.61%	37.41%	78.52%	79.50%
KNN	97.61%	95.18%	96.32%	92.56%	87.65%	74.28%	91.78%	82.42%	99.58%	98.09%
Naive Bayes	78.02%	77.94%	28.10%	28.14%	28.57%	29.01%	28.33%	28.57%	71.87%	71.87%
Decision Tree	100.00%	97.43%	100.00%	92.08%	100.00%	90.95%	100.00%	91.51%	100.00%	94.77%
Random Forest	100.00%	97.31%	100.00%	99.75%	100.00%	82.51%	100.00%	90.32%	100.00%	99.83%
Bagging for Random Forest	99.49%	95.24%	99.91%	99.70%	96.74%	68.93%	98.30%	81.51%	100.00%	99.49%
Ada Boost	87.86%	86.76%	71.89%	64.93%	33.16%	28.19%	45.38%	39.31%	85.20%	82.29%
Gradient Boost	91.04%	89.89%	93.31%	87.56%	44.27%	39.09%	60.05%	54.05%	94.33%	91.07%
Tuned Logistic Regression	87.15%	86.89%	77.16%	73.10%	22.05%	21.81%	34.29%	33.60%	78.97%	79.85%
Tuned LDA	86.86%	87.02%	69.85%	70.06%	23.90%	25.51%	35.61%	37.41%	78.52%	79.50%
Tuned KNN	100.00%	98.75%	100.00%	96.07%	100.00%	95.68%	100.00%	95.88%	100.00%	97.49%
Tuned Random Forest	88.60%	87.33%	95.51%	90.10%	26.28%	18.72%	41.22%	31.01%	96.65%	92.69%
Tuned Bagging	87.33%	86.42%	93.15%	85.14%	17.99%	12.96%	30.16%	22.50%	94.73%	90.84%
Tuned Ada Boost	87.93%	86.80%	72.24%	64.81%	33.51%	28.81%	45.78%	39.89%	85.60%	82.64%
Tuned Gradient Boost	98.07%	94.59%	99.40%	94.84%	87.83%	68.11%	93.26%	79.28%	99.82%	97.78%

Table 18: Performance of different models on mobile data.

Model	Train Accuracy	Test Accuracy	Train Precision	Test Precision	Train Recall	Test Recall	Train F1 Score	Test F1 Score	Train AUC	Test AUC
Logistic Regression	81.68%	84.68%	71.07%	77.59%	44.56%	54.22%	54.78%	63.83%	82.19%	89.29%
LDA	81.94%	85.89%	71.90%	77.27%	45.08%	61.45%	55.41%	68.46%	81.53%	90.08%
KNN	96.26%	89.49%	98.81%	88.71%	86.01%	66.27%	91.97%	75.86%	99.04%	94.77%
Naive Bayes	77.55%	77.18%	57.36%	54.93%	38.34%	46.99%	45.96%	50.65%	76.61%	81.86%
Decision Tree	100.00%	92.79%	100.00%	79.80%	100.00%	95.18%	100.00%	86.81%	100.00%	93.59%
Random Forest	100.00%	97.00%	100.00%	93.98%	100.00%	93.98%	100.00%	93.98%	100.00%	99.68%
Bagging for Random Forest	99.74%	93.39%	100.00%	92.96%	98.96%	79.52%	99.48%	85.71%	100.00%	99.14%
Ada Boost	91.35%	86.49%	89.87%	75.00%	73.58%	68.67%	80.91%	71.70%	96.02%	91.17%
Gradient Boost	98.71%	95.80%	100.00%	97.26%	94.82%	85.54%	97.34%	91.03%	99.96%	99.30%
Tuned Logistic Regression	81.81%	84.68%	71.67%	78.57%	44.56%	53.01%	54.95%	63.31%	82.14%	89.46%
Tuned LDA	81.94%	85.89%	71.90%	77.27%	45.08%	61.45%	55.41%	68.46%	81.53%	90.08%
Tuned KNN	100.00%	99.10%	100.00%	97.62%	100.00%	98.80%	100.00%	98.20%	100.00%	99.00%
Tuned Random Forest	84.39%	83.48%	96.15%	88.89%	38.86%	38.55%	55.35%	53.78%	96.14%	91.78%
Tuned Bagging	81.03%	81.08%	96.00%	95.45%	24.87%	25.30%	39.51%	40.00%	93.51%	90.74%
Tuned Ada Boost	91.61%	86.79%	89.51%	74.68%	75.13%	71.08%	81.69%	72.84%	96.64%	91.02%
Tuned Gradient Boost	100.00%	97.30%	100.00%	91.11%	100.00%	98.80%	100.00%	94.80%	100.00%	99.79%

Table 19: Performance of different models on laptop data.

The most effective model across various metrics, including precision and recall, among the models considered was the tuned KNN model. It demonstrated superior performance across the board, making it the standout choice in terms of overall effectiveness.

5. Model Validation.

Tuned KNN Model

The KNN model was tuned using grid search cross-validation with parameters including the number of neighbors, algorithm (auto, ball_tree, kd_tree, brute), and weights (uniform, distance). After an extensive search, the optimal configuration was identified as having 1 neighbor, utilizing the 'auto' algorithm, and employing 'uniform' weights.

Classification Report on Train Data

	precision	recall	f1-score	support
0	1.00	1.00	1.00	6322
1	1.00	1.00	1.00	1134
accuracy			1.00	7456
macro avg	1.00	1.00	1.00	7456
weighted avg	1.00	1.00	1.00	7456

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2710
1	0.96	0.96	0.96	486
accuracy			0.99	3196
macro avg	0.98	0.97	0.98	3196
weighted avg	0.99	0.99	0.99	3196

Confusion matrix on train data

```
[[6322    0]
 [   0 1134]]
```

Confusion matrix on test data

```
[[2691   19]
 [  21  465]]
```

Classification Report on Train Data

	precision	recall	f1-score	support
0	1.00	1.00	1.00	582
1	1.00	1.00	1.00	193
accuracy			1.00	775
macro avg	1.00	1.00	1.00	775
weighted avg	1.00	1.00	1.00	775

Classification Report on Test Data

	precision	recall	f1-score	support
0	1.00	0.99	0.99	250
1	0.98	0.99	0.98	83
accuracy			0.99	333
macro avg	0.99	0.99	0.99	333
weighted avg	0.99	0.99	0.99	333

Confusion matrix on train data

```
[[582   0]
 [  0 193]]
```

Confusion matrix on test data

```
[[248   2]
 [  1  82]]
```

Table 20: Model validation on Mobile Data and Laptop Data respectively.

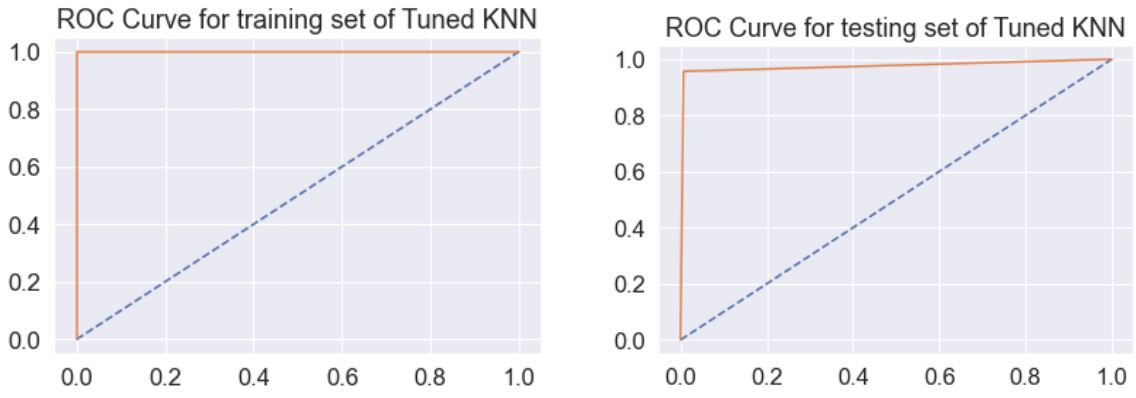

Fig. 10: ROC Curves of Tuned KNN Model on Mobile Data

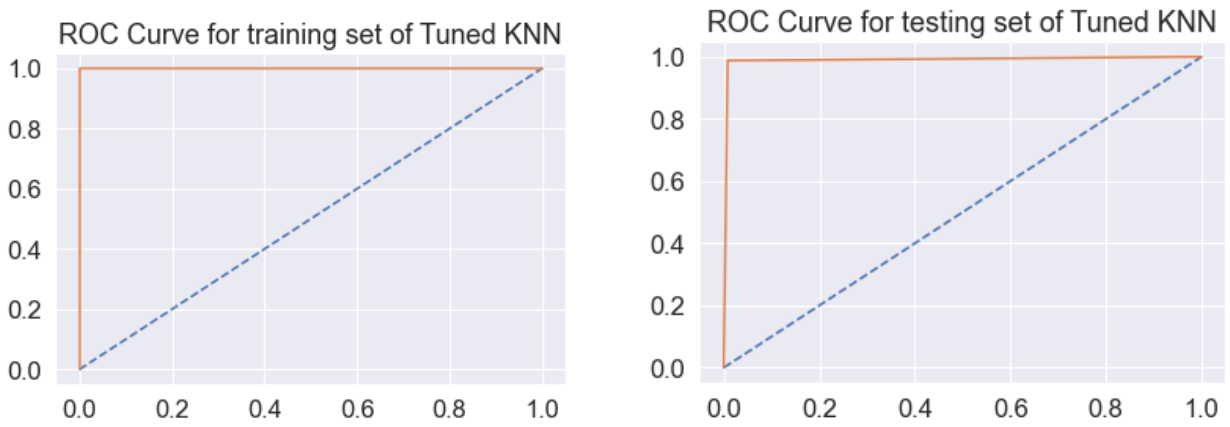

Fig. 11: ROC Curves of Tuned KNN Model on Laptop Data.

In the evaluation of our classification models, we prioritized precision and recall over accuracy due to the imbalanced nature of the data. Recognizing the equal weightage of Type 1 and Type 2 errors in our context—where both misallocating resources to the wrong customers (false positives) and failing to contact interested customers (false negatives) are of significant concern—we focused on achieving a balance between precision and recall.

This model demonstrates commendable precision and recall scores for both mobile and laptop data. For the mobile data, the model achieves a precision of 0.96, signifying the accuracy of positive predictions among those predicted as interested customers. The recall score of 0.96 emphasizes the model's effectiveness in capturing the majority of actual interested customers. Similarly, in the laptop data classification report, class 1 exhibits a precision of 0.98, indicating a high level of accuracy in positive predictions, and a recall score of 0.99, showcasing the model's proficiency in identifying a substantial portion of actual interested customers. These outcomes underscore the models' ability to discern and accurately predict the presence of interested customers, crucial for resource allocation and engagement strategies.

6. Final Interpretation and Recommendation.

Interpretation:

1. Nuclear Families and Travel Inclination:

Nuclear families show a higher likelihood of ticket purchases, indicating a market segment inclined towards travel. Recognize and target nuclear families in marketing campaigns, tailoring promotions to cater to their travel preferences.

2. Non-working Individuals and Ticket Purchase Rate:

Non-working individuals demonstrate a higher ticket purchase rate compared to the working class. Create campaigns specifically targeting non-working individuals, emphasizing flexible schedules and mid-week getaways to capture this market segment.

3. Outstation Check-ins and Customer Satisfaction:

Customers with a greater number of outstation check-ins are more likely to buy tickets, suggesting satisfaction with the company's services. Implement strategies to retain and engage satisfied customers, potentially through loyalty programs or targeted promotions.

4. Preferential Travel Patterns and Beach Vacations:

Preferential customer travel patterns suggest a notable inclination towards beach vacations. Expand beach vacation packages, showcasing unique features and experiences for heightened promotion to cater to this specific preference.

Business Recommendations:

1. Strategic Marketing:

- Expand Beach Vacation Packages: Capitalize on the inclination towards beach vacations by expanding promotional efforts, showcasing unique features and experiences to attract this market segment.
- Target Non-working Individuals: Craft campaigns for non-working individuals, emphasizing flexible schedules and mid-week getaways to tap into the higher ticket purchase rate observed in this group.

2. Customer Engagement:

- Loyalty Programs: Introduce a loyalty program with perks for frequent outstation check-ins, enhancing customer retention by rewarding and acknowledging their loyalty.
- Tailored Aviation Bundles: Craft aviation bundles catering to distinct customer needs, offering all-inclusive options for leisure travelers and flexible ticket choices for business travelers.

3. Business Optimization:

- Tailored Marketing Campaigns: Utilize predictive models to create highly targeted marketing campaigns, optimizing resource allocation and offering tailored promotions to boost sales.
- Pricing Strategies: Adjust pricing based on model insights, considering competitive pricing for price-sensitive customers and exploring premium options for less price-sensitive segments.

4. Customer-Centric Approach:

- Personalized Customer Engagement: Leverage predictive models to identify high-potential leads and provide personalized experiences, enhancing customer satisfaction and loyalty.
- Product Bundling: Create enticing product bundles aligned with customer preferences, simplifying decision-making and increasing the likelihood of purchase.

5. Continuous Model Refinement:

- Regular Updates: Continually monitor and refine predictive models, incorporating new data and adjusting hyperparameters to ensure accuracy and relevance in a dynamic aviation industry.
- Market Adaptation: Stay ahead of market shifts and customer trends by adapting predictive capabilities to evolving industry dynamics.

APPENDIX

Models built on the data where the login device is a Mobile

Logistic Regression

We are constructing a linear regression model using the linear model library provided by scikit-learn The classification report, confusion matrix and the ROC curves of train and test data are as follows.

```
--------------------------------------------------

Classification Report on Train Data

              precision    recall  f1-score   support

           0       0.88      0.99      0.93      6322
           1       0.77      0.21      0.33      1134

    accuracy                           0.87      7456
   macro avg       0.82      0.60      0.63      7456
weighted avg       0.86      0.87      0.84      7456

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.88      0.99      0.93      2710
           1       0.74      0.22      0.34       486

    accuracy                           0.87      3196
   macro avg       0.81      0.60      0.63      3196
weighted avg       0.85      0.87      0.84      3196

--------------------------------------------------

Confusion matrix on train data
[[6248   74]
 [ 891  243]]

Confusion matrix on test data
[[2672   38]
 [ 380  106]]
--------------------------------------------------
```

Table: Evaluation Metrics of Logistic Regression.

The AUC of train data is 0.79 and that of test data is 0.799.

Figure: ROC of Logistic Regression on train data.

Figure: ROC of Logistic Regression on test data.

Linear Discriminant Analysis

The LDA model is constructed using the scikit-learn Discriminant Analysis library. Here are the key characteristics and details of the model:

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.88	0.98	0.93	6322
1	0.70	0.24	0.36	1134
accuracy			0.87	7456
macro avg	0.79	0.61	0.64	7456
weighted avg	0.85	0.87	0.84	7456

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.88	0.98	0.93	2710
1	0.70	0.26	0.37	486
accuracy			0.87	3196
macro avg	0.79	0.62	0.65	3196
weighted avg	0.85	0.87	0.84	3196

Confusion matrix on train data

```
[[6205  117]
 [ 863  271]]
```

Confusion matrix on test data

```
[[2657   53]
 [ 362  124]]
```

Table: Evaluation Metric of LDA Model.

The AUC of train data is 0.785 and that of test data is 0.795.

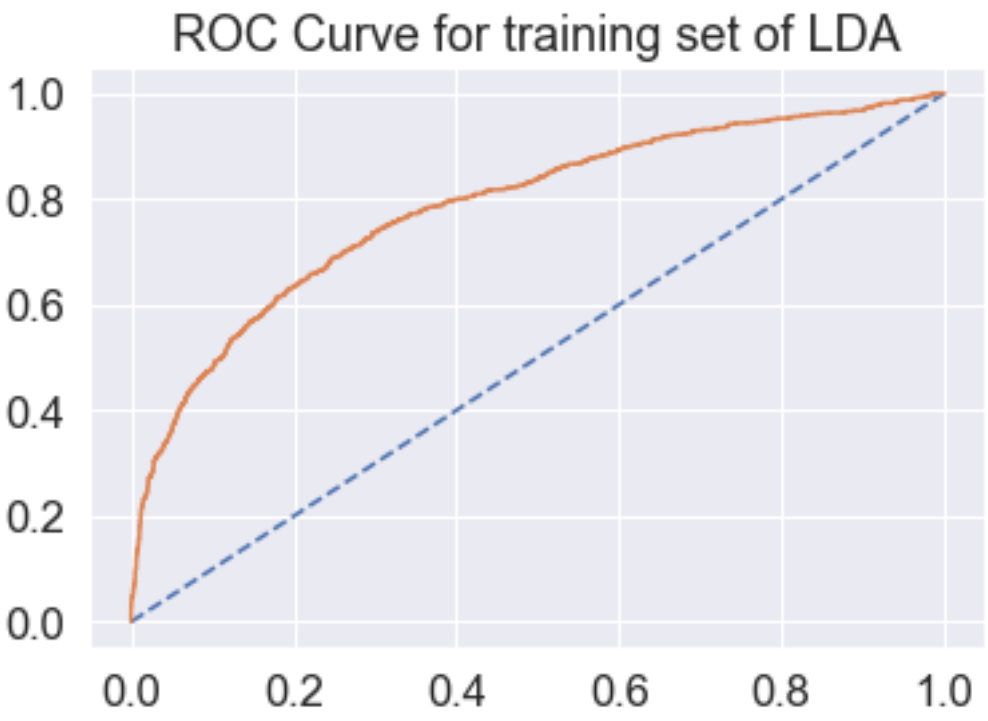

Figure: ROC of train data on LDA model.

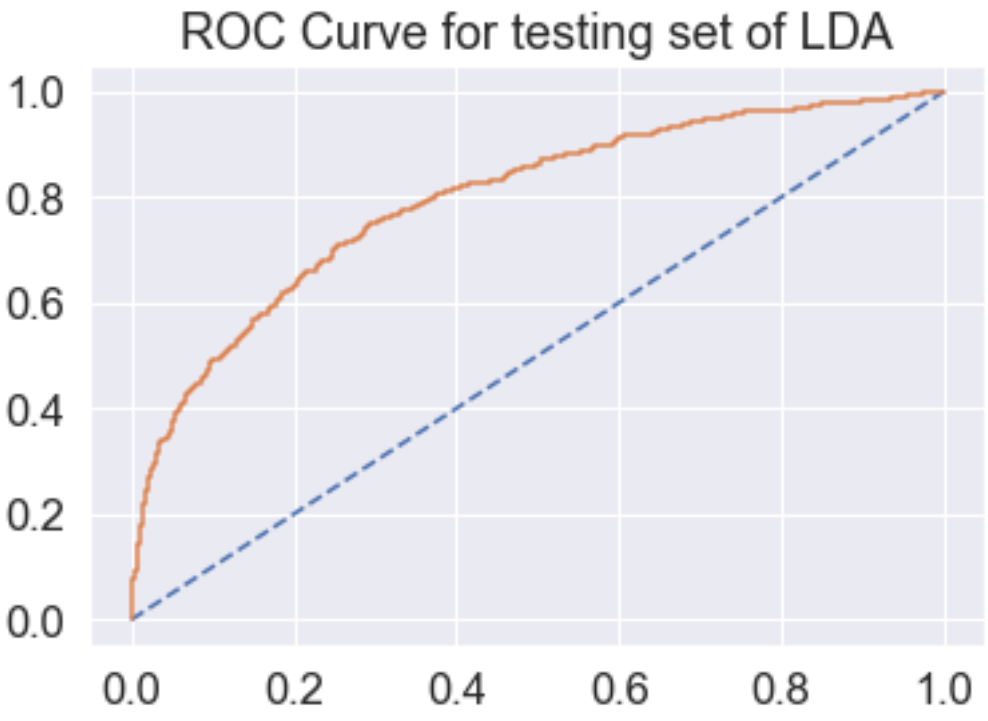

Figure : ROC of test data on LDA model.

Naïve Bayes

We are using Gaussian Naïve Bayesfrom from sklearn library to build this model. The details are as follows.

```
------------------------------------------------------

Classification Report on Train Data

              precision    recall  f1-score   support

           0       0.87      0.87      0.87      6322
           1       0.28      0.29      0.28      1134

    accuracy                           0.78      7456
   macro avg       0.58      0.58      0.58      7456
weighted avg       0.78      0.78      0.78      7456

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.87      0.87      0.87      2710
           1       0.28      0.29      0.29       486

    accuracy                           0.78      3196
   macro avg       0.58      0.58      0.58      3196
weighted avg       0.78      0.78      0.78      3196

------------------------------------------------------

Confusion matrix on train data
[[5493  829]
 [ 810  324]]

Confusion matrix on test data
[[2350  360]
 [ 345  141]]
------------------------------------------------------
```

Table: Evaluation Metric of Naïve Bayes Model.

The AUC of train data and test data are both 0.719

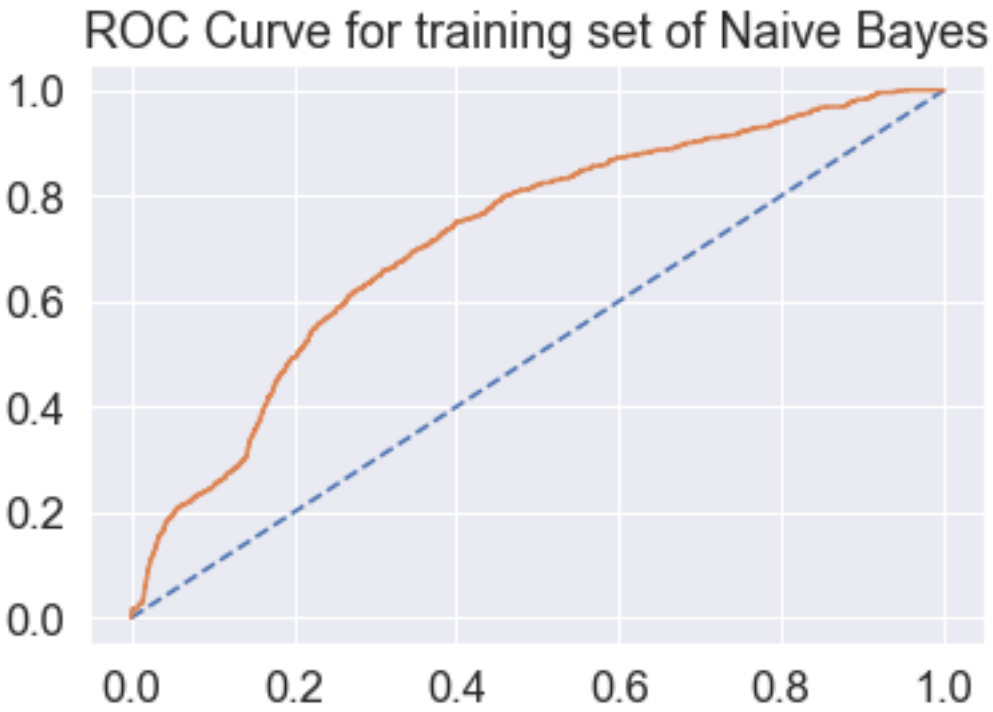

Figure: ROC of train data on Naïve Bayes model.

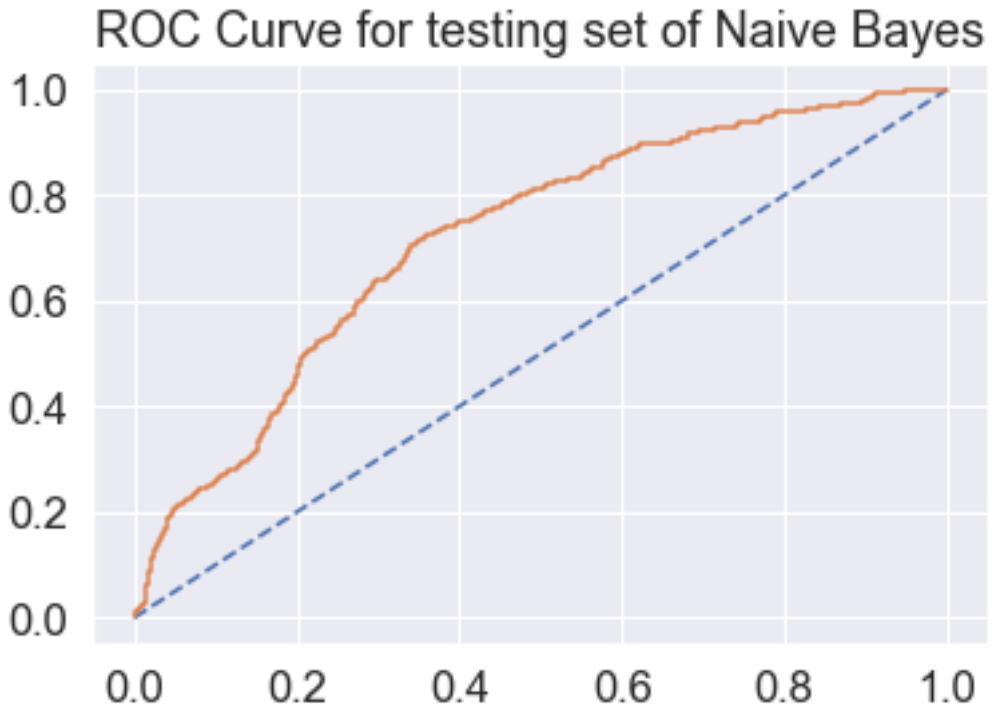

Figure : ROC of test data on Naïve Bayes model.

K-Nearest Neighbors

This model is also built from sklearn neighbors library. The details are given below.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.98	0.99	0.99	6322
1	0.96	0.88	0.92	1134
accuracy			0.98	7456
macro avg	0.97	0.94	0.95	7456
weighted avg	0.98	0.98	0.98	7456

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.96	0.99	0.97	2710
1	0.93	0.74	0.82	486
accuracy			0.95	3196
macro avg	0.94	0.87	0.90	3196
weighted avg	0.95	0.95	0.95	3196

Confusion matrix on train data

```
[[6284   38]
 [ 140  994]]
```

Confusion matrix on test data

```
[[2681   29]
 [ 125  361]]
```

Table: Evaluation Metric of KNN Model.

The AUC of the train data is 0.996 and that of test is 0.981.

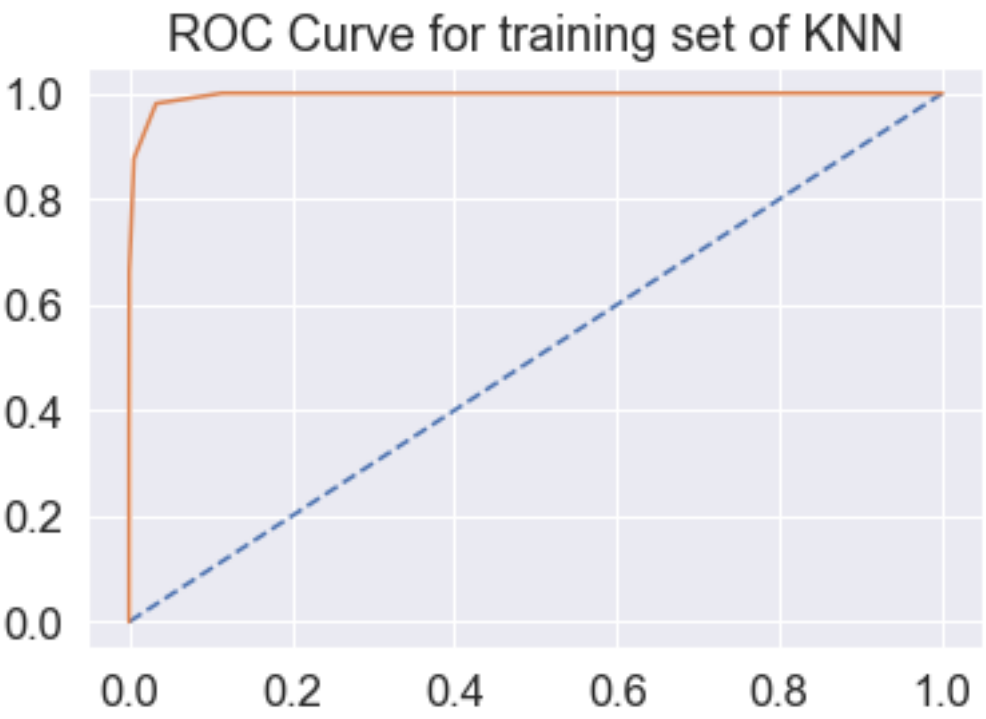

Figure: ROC of train data on KNN model.

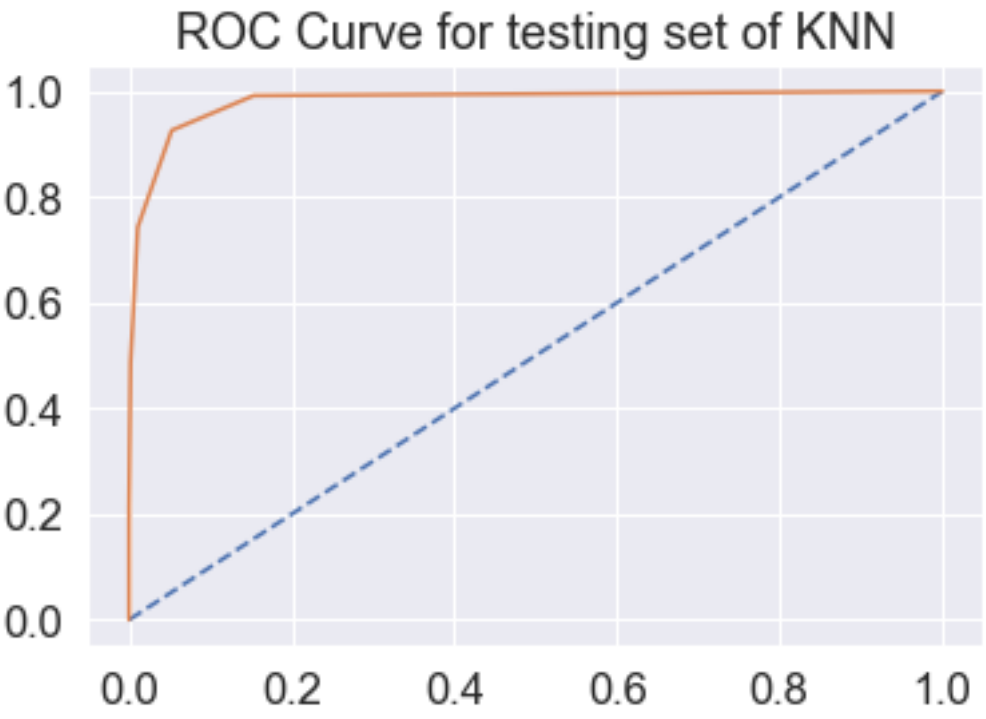

Figure: ROC of test data on KNN model.

Decision Tree Model

This model is built using the decision tree classifier from sklearn tree library. The details are given below.

```
-----------------------------------------------------

Classification Report on Train Data

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      6322
           1       1.00      1.00      1.00      1134

    accuracy                           1.00      7456
   macro avg       1.00      1.00      1.00      7456
weighted avg       1.00      1.00      1.00      7456

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.98      0.99      0.99      2710
           1       0.92      0.91      0.92       486

    accuracy                           0.97      3196
   macro avg       0.95      0.95      0.95      3196
weighted avg       0.97      0.97      0.97      3196

-----------------------------------------------------

Confusion matrix on train data
[[6322    0]
 [   0 1134]]

Confusion matrix on test data
[[2674   36]
 [  44  442]]
-----------------------------------------------------
```

Table: Evaluation Metric of Decision Tree Model.

The AUC of train data is 1.0 and that of test data is 0.948

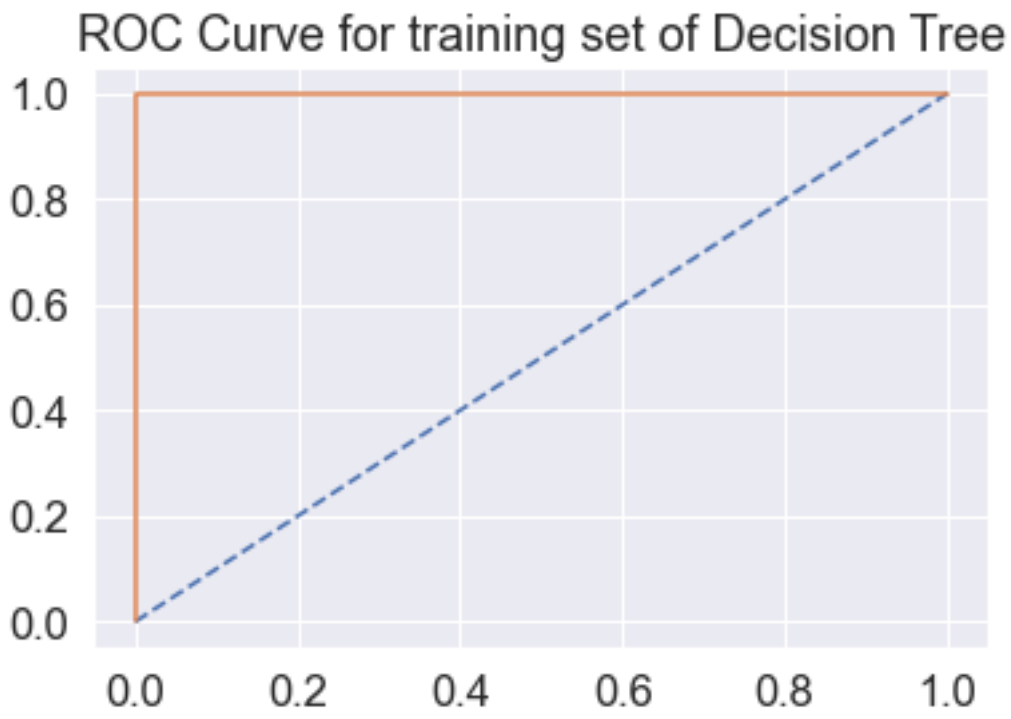

Figure: ROC of train data on Decision Tree model.

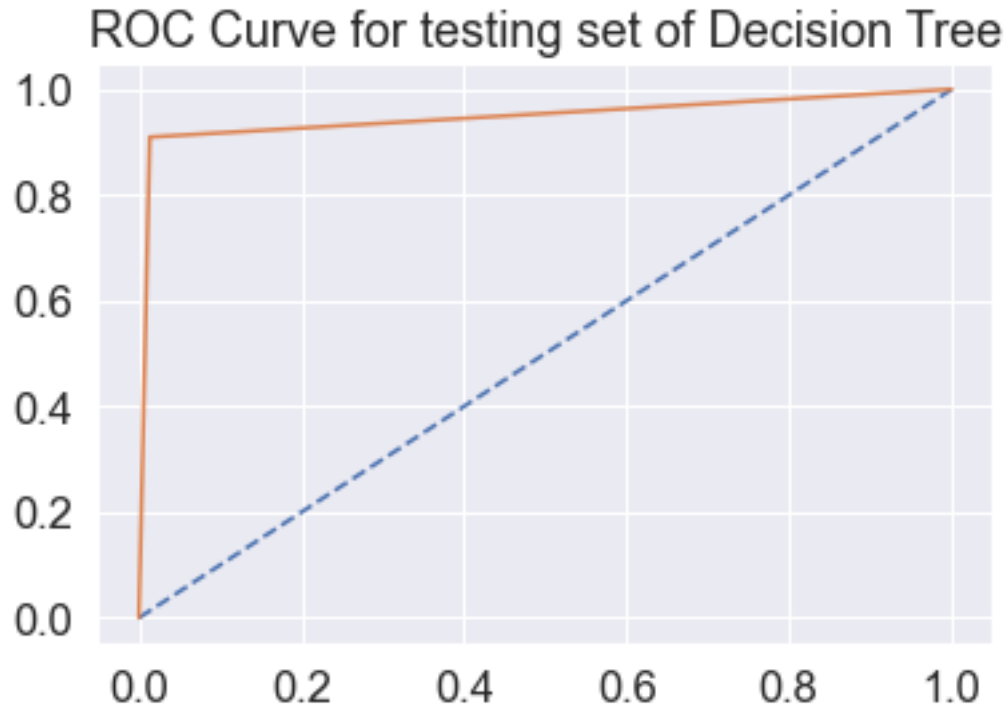

Figure: ROC of test data on Decision Tree Model.

a. Interpretation of the models:

Logistic Regression:

- Train and test accuracy are similar, suggesting that the model is not overfitting.
- Train and test precision are higher than recall, suggesting that the model is better at identifying positive cases than negative cases.
- F1 score is moderate, suggesting that the model has a good balance between precision and recall.
- AUC is high, suggesting that the model is good at ranking positive cases higher than negative cases.

Linear Discriminant Analysis:

- Train and test accuracy are similar, suggesting that the model is not overfitting.
- Train and test precision are lower than recall, suggesting that the model is better at identifying negative cases than positive cases.
- F1 score is moderate, suggesting that the model has a good balance between precision and recall.
- AUC is slightly lower than logistic regression, suggesting that the model is not as good at ranking positive cases higher than negative cases.

K-Nearest Neighbors:

- Train accuracy is very high, but test accuracy is lower, suggesting that the model is overfitting the training data.
- Train and test precision are high, suggesting that the model is good at identifying both positive and negative cases.
- Train and test recall are high, suggesting that the model is good at identifying all positive cases, even if it means identifying some false positives.
- F1 score is very high, suggesting that the model has a very good balance between precision and recall.
- AUC is very high, suggesting that the model is very good at ranking positive cases higher than negative cases.

Naive Bayes:

- Train and test accuracy are similar, but relatively low, suggesting that the model is not very good at predicting the outcome.
- Train and test precision are low, suggesting that the model is not very good at identifying either positive or negative cases.
- Train and test recall are low, suggesting that the model is not very good at identifying all positive cases, even if it means identifying some false positives.
- F1 score is very low, suggesting that the model has a very poor balance between precision and recall.
- AUC is slightly higher than 0.5, suggesting that the model is slightly better at ranking positive cases higher than negative cases than random guessing.

Decision Tree:

- Train accuracy is perfect, but test accuracy is very high, suggesting that the model is overfitting the training data to a very small extent.
- Train and test precision are high, suggesting that the model is very good at identifying both positive and negative cases.
- Train and test recall are high, suggesting that the model is very good at identifying all positive cases, even if it means identifying some false positives.
- F1 score is very high, suggesting that the model has a very good balance between precision and recall.
- AUC is perfect, suggesting that the model is perfect at ranking positive cases higher than negative cases.

Model Tuning and Business Implication.

a. Ensemble Modelling:

Ensemble techniques are widely used in machine learning to improve model performance. They involve combining multiple models to make predictions more accurate and robust. By leveraging the diversity of different models, ensembles can reduce overfitting and enhance overall predictive power, making them a valuable tool for achieving high-quality results in various applications. We are building the following ensemble techniques.

Random Forest

To build the random forest model we are using the Random Forest Classifier from sklearn ensemble library. The details are as follows.

```
Classification Report on Train Data

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      6322
           1       1.00      1.00      1.00      1134

    accuracy                           1.00      7456
   macro avg       1.00      1.00      1.00      7456
weighted avg       1.00      1.00      1.00      7456

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.97      1.00      0.98      2710
           1       1.00      0.83      0.90       486

    accuracy                           0.97      3196
   macro avg       0.98      0.91      0.94      3196
weighted avg       0.97      0.97      0.97      3196

-----------------------------------------------------

Confusion matrix on train data
[[6322    0]
 [   0 1134]]

Confusion matrix on test data
[[2709    1]
 [  85  401]]
-----------------------------------------------------
```

Table: Evaluation Metric of Random Forest.

The AUC of train data is 1 and that of test is 0.998. The ROC Curve are given below.

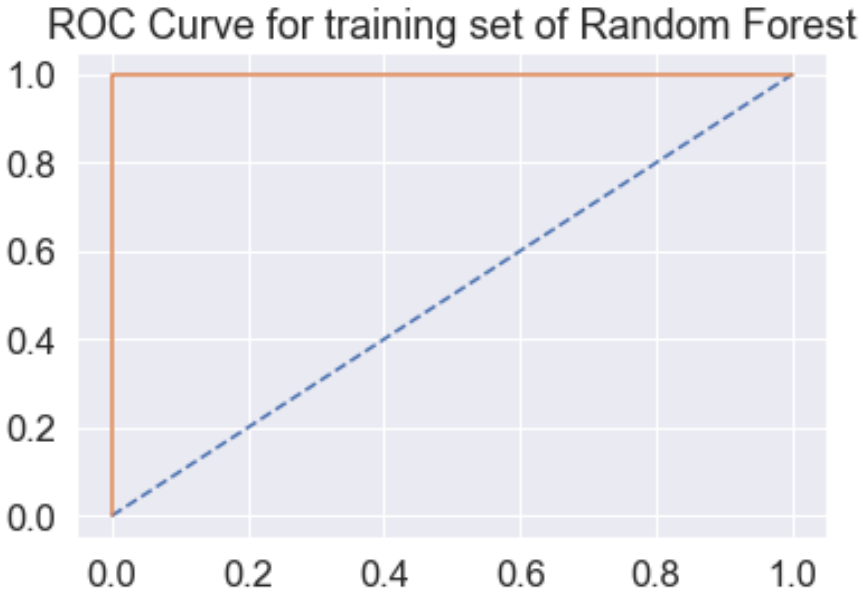

Figure: ROC of train data on Random Forest

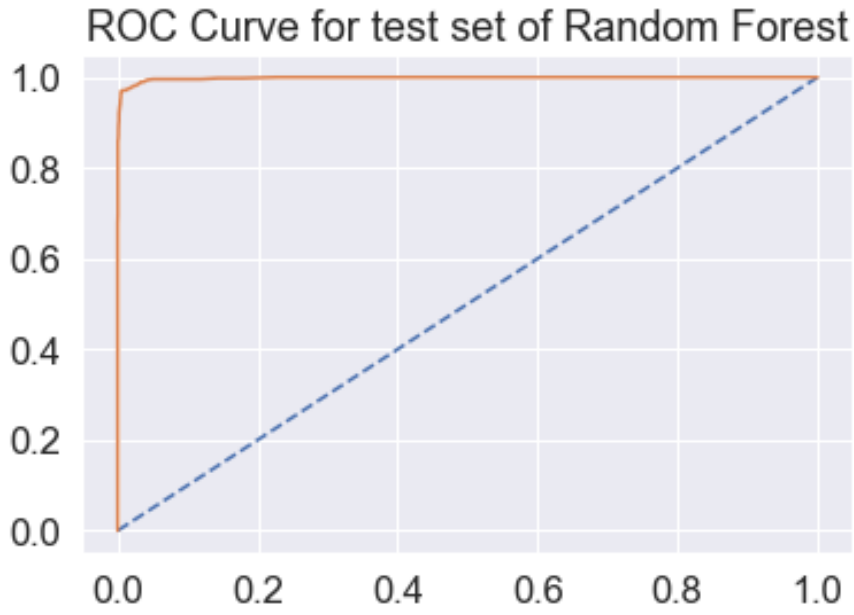

Figure: ROC of train data on Random Forest

The Random Forest model performs exceptionally well on the training data with perfect accuracy and precision. On the test data, it maintains high accuracy and precision, indicating good generalization. However, its recall on the test data is slightly lower, suggesting it may miss some positive instances. Overall, it's a robust model with high AUC scores on both training and test data.

Ada Boosting

We are using AdaBoost classifier from the sklearn ensemble library to build this mode. The details are as follows.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.89	0.98	0.93	6322
1	0.72	0.33	0.45	1134
accuracy			0.88	7456
macro avg	0.80	0.65	0.69	7456
weighted avg	0.86	0.88	0.86	7456

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.88	0.97	0.93	2710
1	0.65	0.28	0.39	486
accuracy			0.87	3196
macro avg	0.77	0.63	0.66	3196
weighted avg	0.85	0.87	0.84	3196

Confusion matrix on train data

```
[[6175  147]
 [ 758  376]]
```

Confusion matrix on test data

```
[[2636   74]
 [ 349  137]]
```

Table: Evaluation Metric of Ada Boost.

The AUC of train data is 0.852 and that of test is 0.823. The ROC curves are given below.

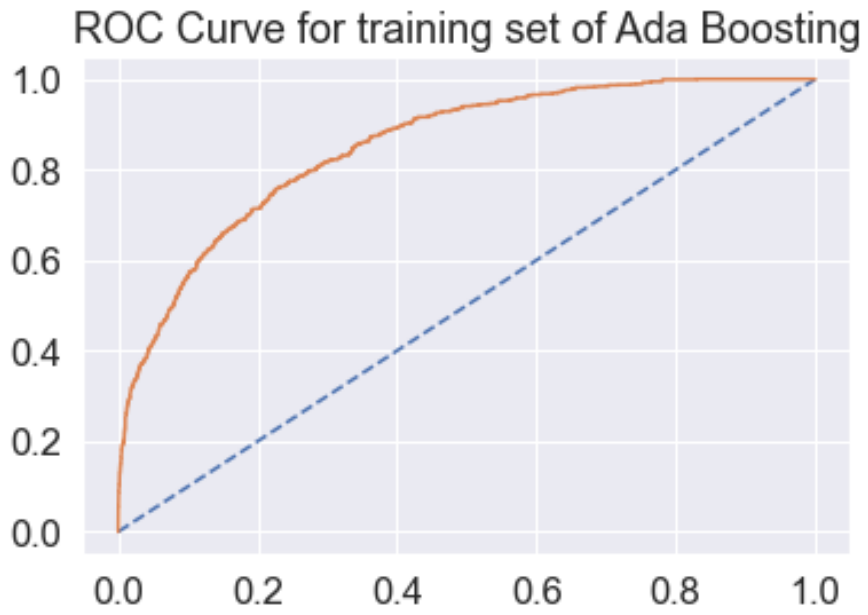

Figure: ROC of train on Ada boost model

Figure: ROC of test on Ada boost model

The Ada Boost model demonstrates good overall performance, with a test accuracy of 86.77%. It achieves a reasonable balance between precision and recall on both training and testing data, indicating it can make accurate predictions while capturing a substantial portion of relevant instances. The model's F1 scores are also relatively high, suggesting its ability to provide a harmonious blend of precision and recall. Additionally, the AUC values for both training and testing sets are robust, showing its effectiveness in distinguishing between positive and negative cases.

Gradient Boosting

We are using Gradient Boost classifier from the sklearn ensemble library to build this model. The details are as follows.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.91	0.99	0.95	6322
1	0.93	0.44	0.60	1134
accuracy			0.91	7456
macro avg	0.92	0.72	0.78	7456
weighted avg	0.91	0.91	0.90	7456

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.90	0.99	0.94	2710
1	0.88	0.39	0.54	486
accuracy			0.90	3196
macro avg	0.89	0.69	0.74	3196
weighted avg	0.90	0.90	0.88	3196

Confusion matrix on train data

```
[[6286   36]
 [ 632  502]]
```

Confusion matrix on test data

```
[[2683   27]
 [ 296  190]]
```

Table: Evaluation Metric of Gradient Boost.

The AUC of train data is 0.943 and that of test is 0.911. The ROC curves are given below.

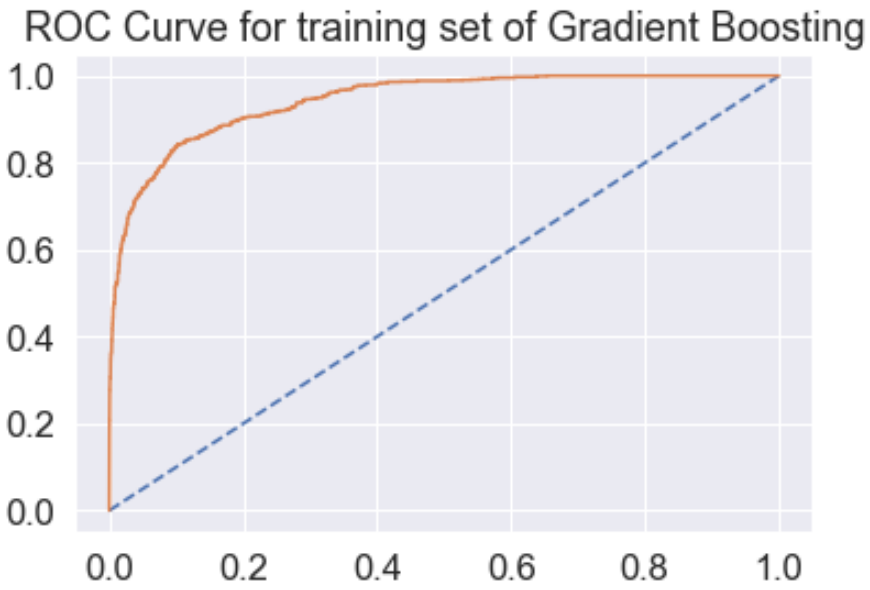

Figure: ROC of train on Gradient boost model

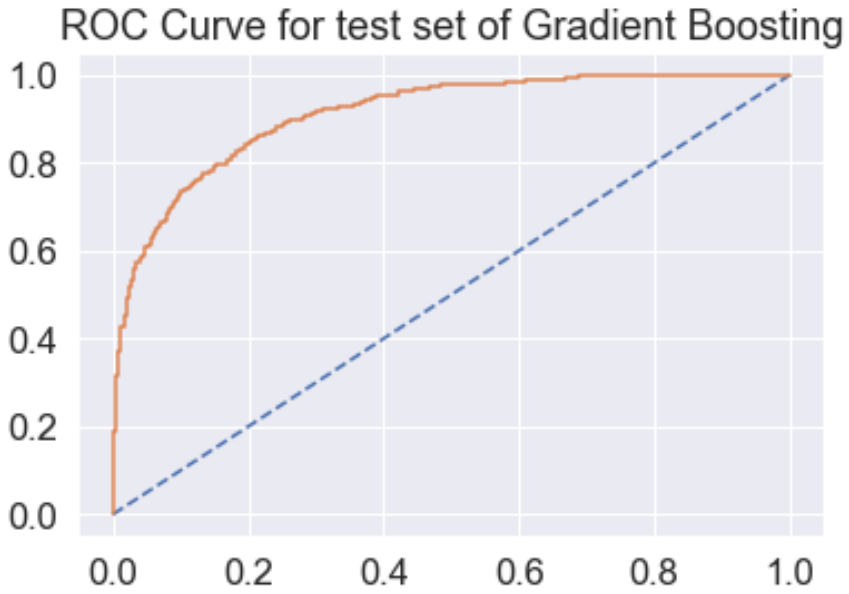

Figure: ROC of test on Gradient boost model

The Gradient Boost model shows strong performance with a high training accuracy of 91.04% and a test accuracy of 89.89%. It excels in precision, with 93.31% precision on the training data and 87.56% on the test data, indicating its ability to make accurate positive predictions. However, it has relatively lower recall, especially on the test set, which means it may miss some positive cases. The F1 scores indicate a balance between precision and recall. Moreover, the model exhibits excellent discriminative power, as evidenced by high AUC scores of 94.33% on training and 91.07% on the test data.

Bagging

We are using Bagging classifier from the sklearn ensemble library to build this model. The details are as follows.

```
Classification Report on Train Data

              precision    recall  f1-score   support

           0       0.99      1.00      1.00      6322
           1       1.00      0.97      0.98      1134

    accuracy                           0.99      7456
   macro avg       1.00      0.98      0.99      7456
weighted avg       0.99      0.99      0.99      7456

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.95      1.00      0.97      2710
           1       1.00      0.69      0.82       486

    accuracy                           0.95      3196
   macro avg       0.97      0.84      0.89      3196
weighted avg       0.95      0.95      0.95      3196

-----------------------------------------------------

Confusion matrix on train data
[[6321    1]
 [  37 1097]]

Confusion matrix on test data
[[2709    1]
 [ 151  335]]
-----------------------------------------------------
```

Table: Evaluation Metric of Bagging

The AUC of train data is 1 and that of test is 0.995. The ROC curves are given below.

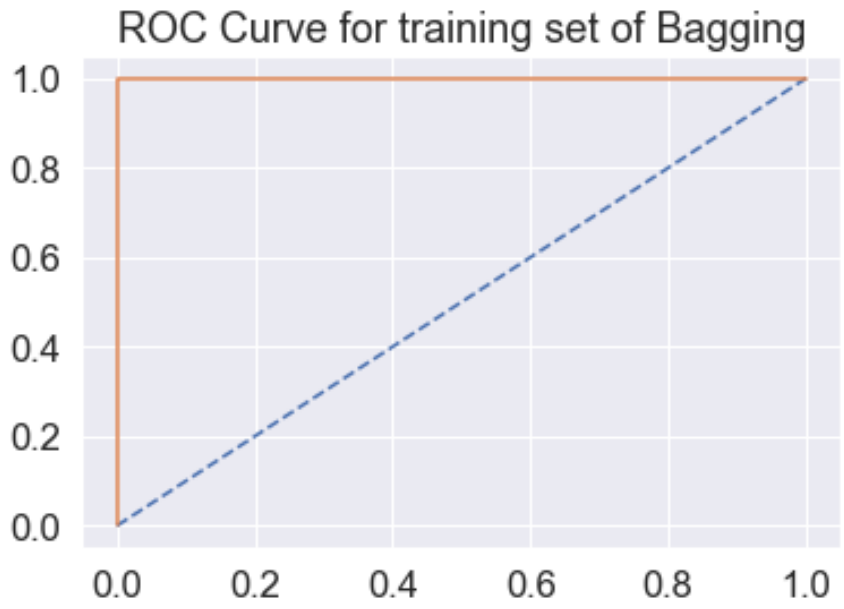

Figure: ROC of train on bagging model

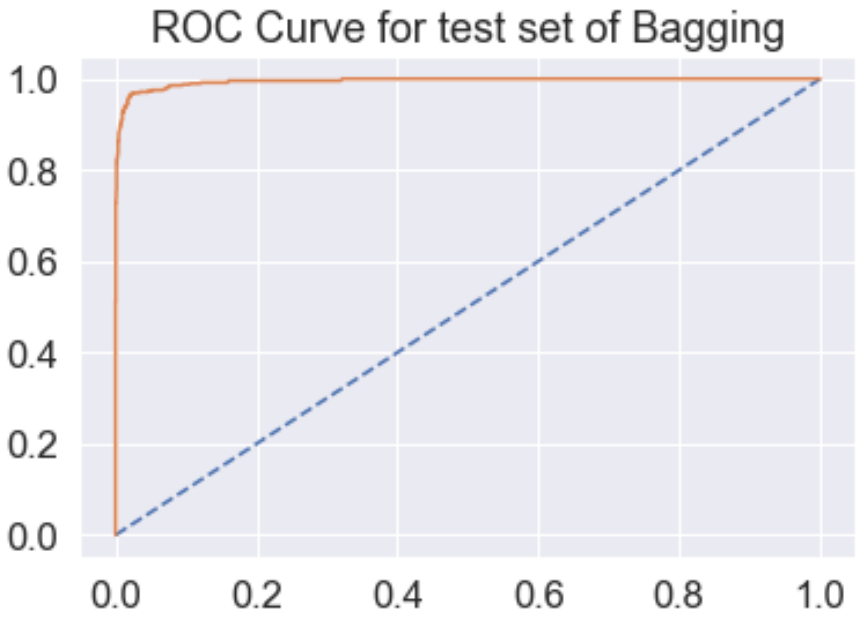

Figure: ROC of test on bagging model

The Bagging for Random Forest model demonstrates high train accuracy (99.49%) and good test accuracy (95.24%). It achieves excellent precision in both training and testing (99.91% and 99.70%), indicating its ability to make accurate positive predictions. However, the model's recall on the test set (68.93%) is comparatively lower, suggesting it may struggle to identify all positive cases. The F1 scores and AUC values indicate strong overall performance, with room for improvement in recall to enhance its predictive capabilities.

b. Model Tuning.

In our model tuning efforts, we applied Grid Search Cross-Validation to optimize all the models. Below is an in-depth exposition of the K-Nearest Neighbors (KNN) model.

The hyperparameters under consideration were 'n_neighbors,' 'algorithm,' and 'weights.' After an exhaustive search, the best configuration was found to be 'n_neighbors' set to 1, 'algorithm' as 'auto,' and 'weights' set to 'uniform.' This configuration emphasizes the nearest neighbor, using the 'auto' algorithm to select the most suitable approach and giving equal importance to all neighbors ('uniform' weights). Except the n_neighbors, the remaining two were the default values.

The impact of this tuning was significant. The original KNN model exhibited commendable performance, with a training accuracy of 97.61%, test accuracy of 95.18%, and other metrics reflecting a strong balance between precision, recall, and F1 score. However, after tuning, the KNN model reached exceptional levels, achieving a flawless 100% training accuracy, and an outstanding 98.75% test accuracy. The precision, recall, and F1 score on the test data also saw substantial improvements, highlighting the ability of the tuned KNN model to make highly accurate predictions. Additionally, the AUC (Area Under the Curve) values further demonstrate the enhanced discriminative power of the tuned model, with the test AUC reaching 97.49%. This exemplifies how meticulous hyperparameter tuning can significantly elevate the performance of machine learning models, making them more adept at capturing patterns in the data and providing more accurate predictions.

For Logistic Regression, we discovered that setting 'C' to 100, 'penalty' to 'none,' and 'solver' to 'lbfgs' yielded the best results, striking a balance between accuracy and model complexity. Linear Discriminant Analysis (LDA) performed best with 'solver' set to 'lsqr,' ensuring robust performance across different dataset sizes. In the case of Random Forest, we selected a 'max_depth' of 10, 'min_samples_leaf' of 15, 'min_samples_split' of 15, and 'n_estimators' of 50 to achieve a reliable trade-off between model accuracy and complexity. Bagging was enhanced with 'n_estimators' set at 30 for increased model diversity. For AdaBoost, the choice of 'algorithm' as 'SAMME.R' and 'n_estimators' as 60 led to good convergence and boosting strength. Finally, Gradient Boosting with 'learning_rate' at 0.1, 'max_depth' at 4, and 'n_estimators' at 200 struck the right balance between accuracy and training speed. These parameter choices were guided by the specific strengths of each model and the need to achieve a balanced performance.

Models built on the data where the login device is a laptop

Logistic Regression:

We are constructing a linear regression model using the linear model library provided by scikit-learn.

The classification report, confusion matrix and the ROC curves of train and test data are as follows.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.84	0.94	0.89	582
1	0.71	0.45	0.55	193
accuracy			0.82	775
macro avg	0.77	0.69	0.72	775
weighted avg	0.81	0.82	0.80	775

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.86	0.95	0.90	250
1	0.78	0.54	0.64	83
accuracy			0.85	333
macro avg	0.82	0.75	0.77	333
weighted avg	0.84	0.85	0.84	333

Confusion matrix on train data

```
[[547  35]
 [107  86]]
```

Confusion matrix on test data

```
[[237  13]
 [ 38  45]]
```

Table: Evaluation Metric of Logistic Regression.

The AUC of train data is 0.822 and that of test data is 0.893.

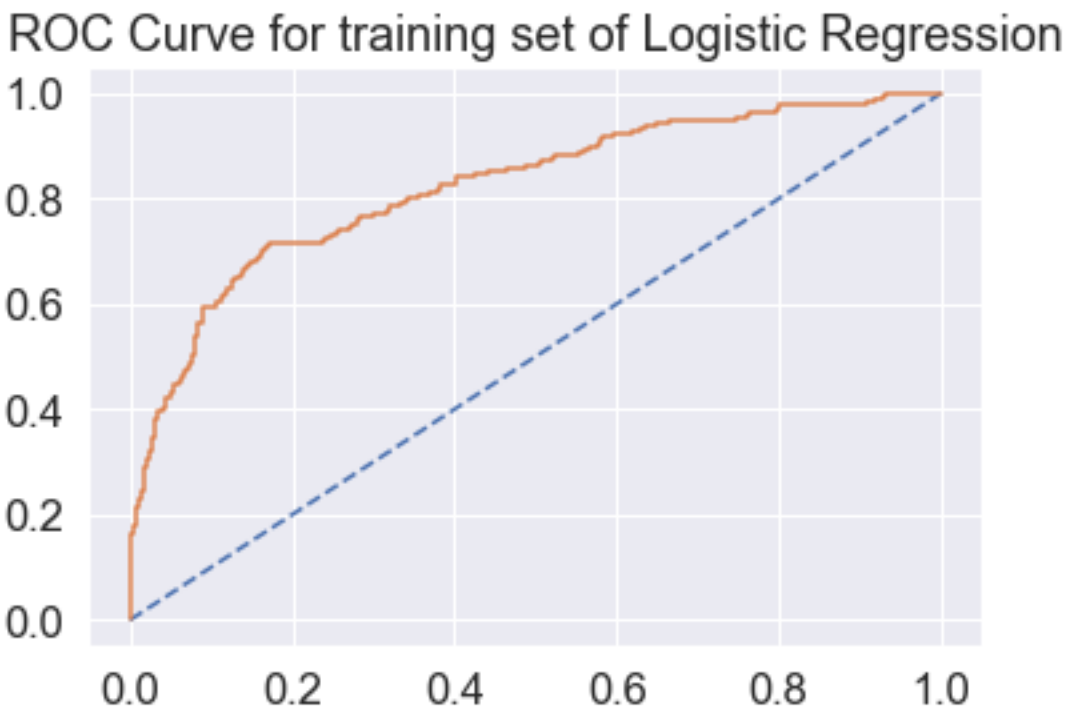

Figure: ROC Curve of Logistic Regression on train data

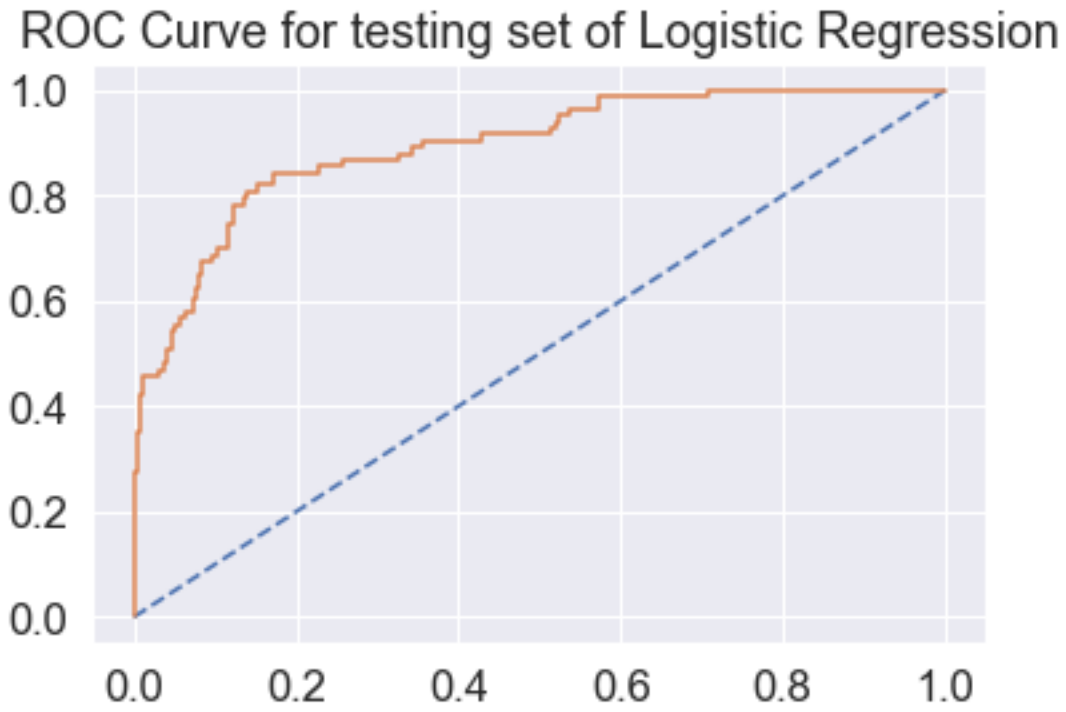

Figure : ROC Curve of Logistic Regression on test data

Linear Discriminant Analysis

The LDA model is constructed using the scikit-learn Discriminant Analysis library. Here are the key characteristics and details of the model:

```
-----------------------------------------------------

Classification Report on Train Data

              precision    recall  f1-score   support

           0       0.84      0.94      0.89       582
           1       0.72      0.45      0.55       193

    accuracy                           0.82       775
   macro avg       0.78      0.70      0.72       775
weighted avg       0.81      0.82      0.80       775

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.88      0.94      0.91       250
           1       0.77      0.61      0.68        83

    accuracy                           0.86       333
   macro avg       0.83      0.78      0.80       333
weighted avg       0.85      0.86      0.85       333

-----------------------------------------------------

Confusion matrix on train data
[[548  34]
 [106  87]]

Confusion matrix on test data
[[235  15]
 [ 32  51]]
-----------------------------------------------------
```

Table: Evaluation Metric of LDA Model.

The AUC of train data is 0.815 and that of test data is 0.901.

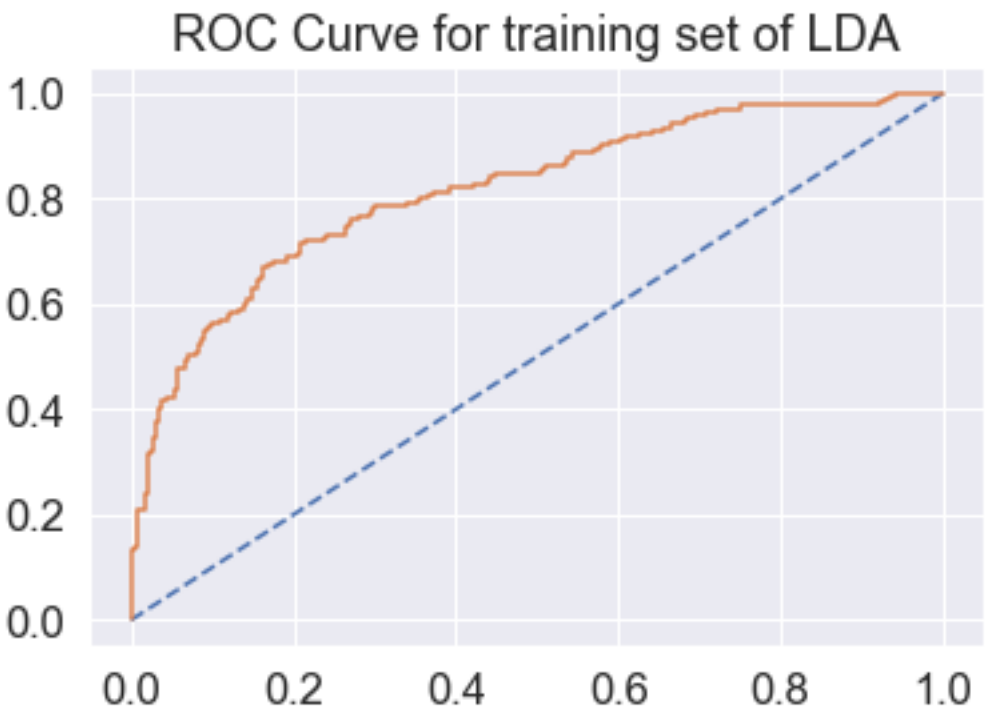

Figure: ROC of LDA on train

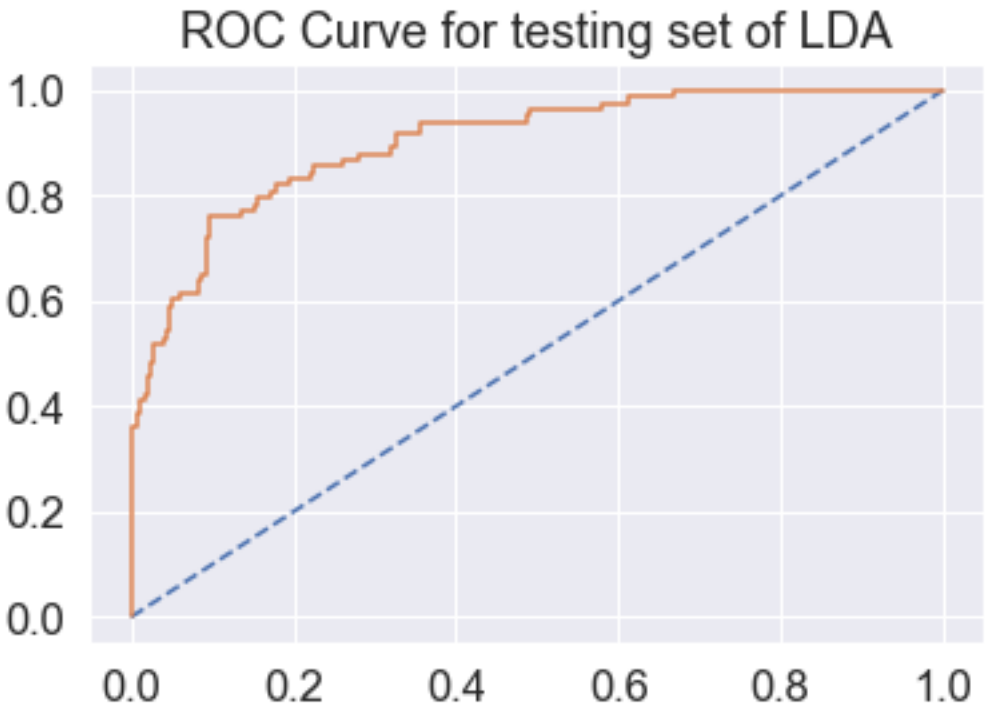

Figure : ROC of LDA on test

Naïve Bayes

We are using Gaussian Naïve Bayesfrom from sklearn library to build this model. The details are as follows.

```
-----------------------------------------------------

Classification Report on Train Data

              precision    recall  f1-score   support

           0       0.82      0.91      0.86       582
           1       0.57      0.38      0.46       193

    accuracy                           0.78       775
   macro avg       0.69      0.64      0.66       775
weighted avg       0.76      0.78      0.76       775

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.83      0.87      0.85       250
           1       0.55      0.47      0.51        83

    accuracy                           0.77       333
   macro avg       0.69      0.67      0.68       333
weighted avg       0.76      0.77      0.77       333

-----------------------------------------------------

Confusion matrix on train data
[[527  55]
 [119  74]]

Confusion matrix on test data
[[218  32]
 [ 44  39]]
-----------------------------------------------------
```

Table: Evaluation Metric of Naïve Bayes Model.

The AUC of train data 0.766 and test data is 0.819

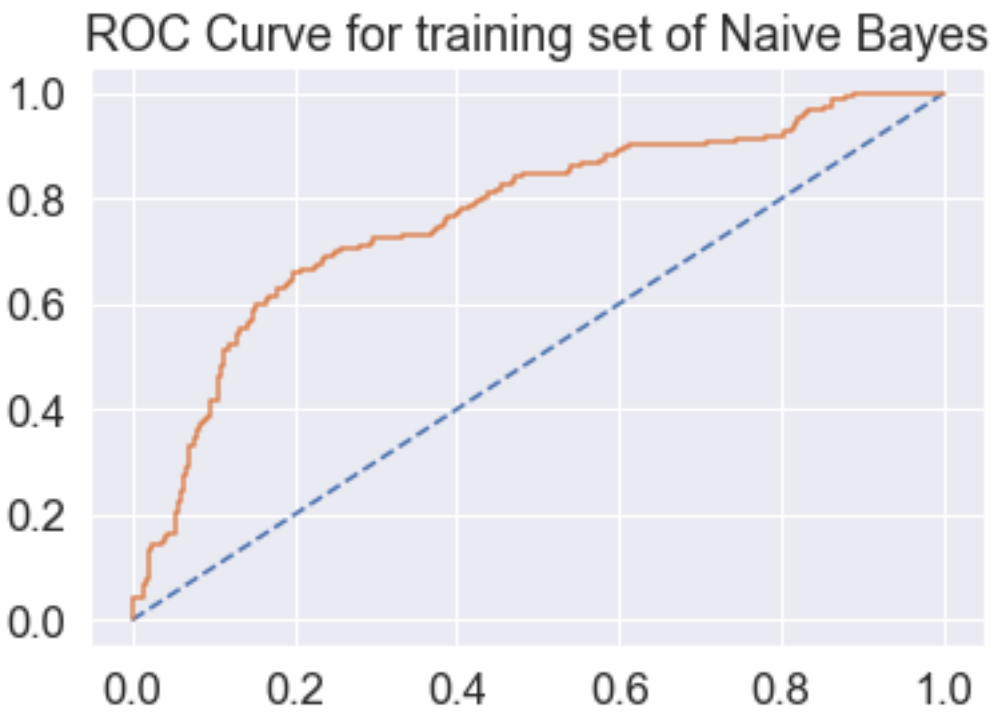

Figure : ROC of train data on Naïve Bayes model.

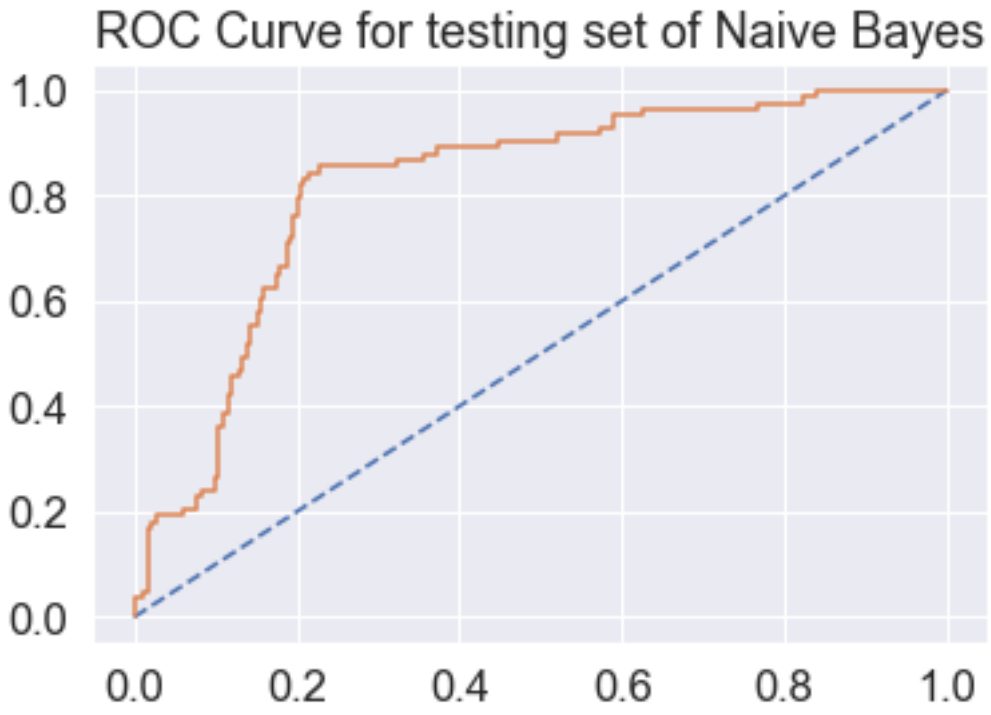

Figure: ROC of test data on Naïve Bayes model.

K-Nearest Neighbors

This model is also built from sklearn neighbors library. The details are given below.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.96	1.00	0.98	582
1	0.99	0.86	0.92	193
accuracy			0.96	775
macro avg	0.97	0.93	0.95	775
weighted avg	0.96	0.96	0.96	775

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.90	0.97	0.93	250
1	0.89	0.66	0.76	83
accuracy			0.89	333
macro avg	0.89	0.82	0.85	333
weighted avg	0.89	0.89	0.89	333

Confusion matrix on train data

```
[[580   2]
 [ 27 166]]
```

Confusion matrix on test data

```
[[243   7]
 [ 28  55]]
```

Table : Evaluation Metric of KNN Model.

The AUC of the train data is 0.990 and that of test is 0.948.

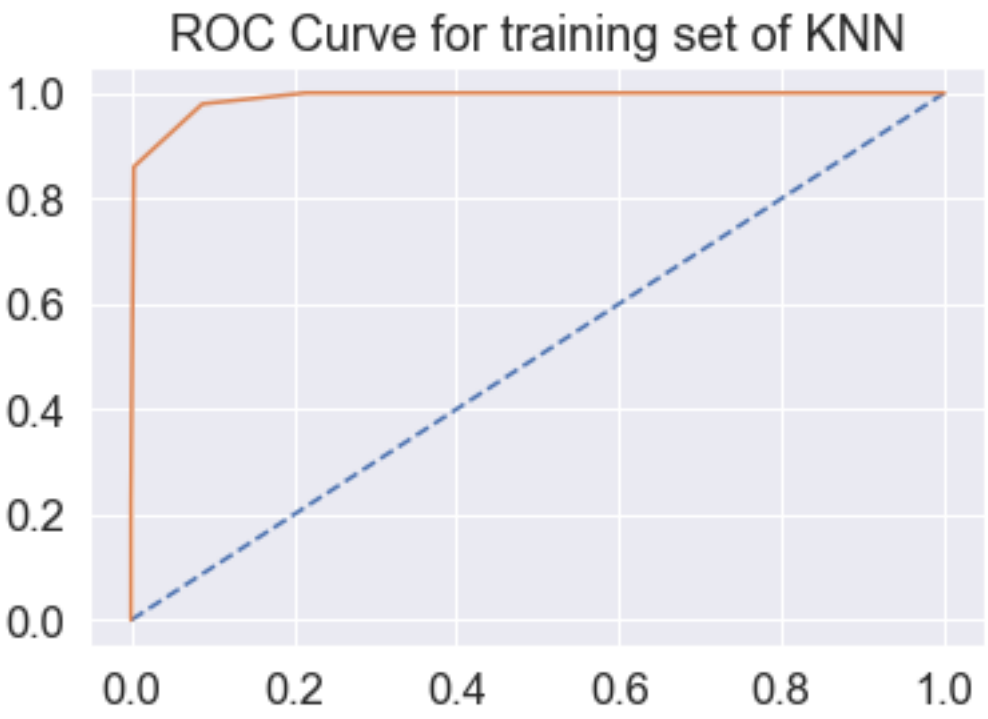

Figure: ROC of train data on KNN model.

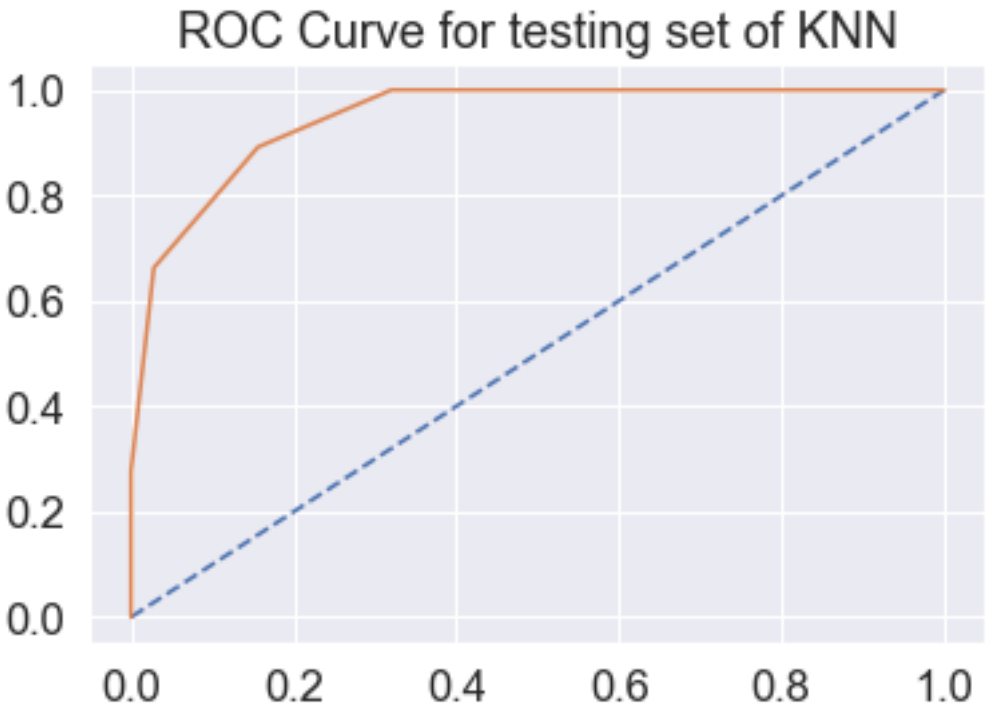

Figure: ROC of test data on KNN model.

Decision Tree Model

This model is built using the decision tree classifier from sklearn tree library. The details are given below.

Classification Report on Train Data

	precision	recall	f1-score	support
0	1.00	1.00	1.00	582
1	1.00	1.00	1.00	193
accuracy			1.00	775
macro avg	1.00	1.00	1.00	775
weighted avg	1.00	1.00	1.00	775

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.99	0.92	0.95	250
1	0.79	0.96	0.87	83
accuracy			0.93	333
macro avg	0.89	0.94	0.91	333
weighted avg	0.94	0.93	0.93	333

Confusion matrix on train data

```
[[582   0]
 [  0 193]]
```

Confusion matrix on test data

```
[[229  21]
 [  3  80]]
```

Table : Evaluation Metric of Decision Tree Model.

The AUC of train data is 1.0 and that of test data is 0.940

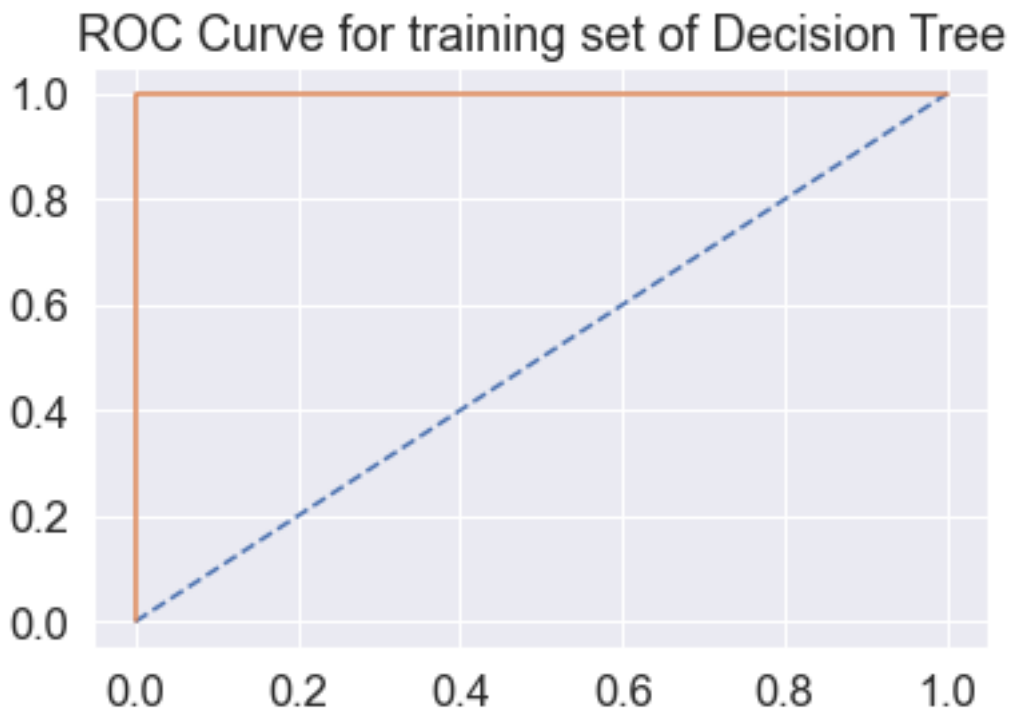

Figure: ROC of train data on Decision Tree model.

Figure: ROC of test data on Decision Tree Model.

Interpretation of the models:

Logistic Regression:

- Train and test accuracy are similar, suggesting that the model is not overfitting.
- Train and test precision are higher than recall, suggesting that the model is better at identifying positive cases than negative cases.
- F1 score is moderate, suggesting that the model has a good balance between precision and recall.
- AUC is high, suggesting that the model is good at ranking positive cases higher than negative cases.

Linear Discriminant Analysis:

- Train and test accuracy are similar, suggesting that the model is not overfitting.
- Train and test precision are lower than recall, suggesting that the model is better at identifying negative cases than positive cases.
- F1 score is moderate, suggesting that the model has a good balance between precision and recall.
- AUC is slightly lower than logistic regression, suggesting that the model is not as good at ranking positive cases higher than negative cases.

K-Nearest Neighbors:

- Train accuracy is very high, but test accuracy is lower, suggesting that the model is overfitting the training data.
- Train and test precision are high, suggesting that the model is good at identifying both positive and negative cases.
- Train and test recall are high, suggesting that the model is good at identifying all positive cases, even if it means identifying some false positives.
- F1 score is very high, suggesting that the model has a very good balance between precision and recall.
- AUC is very high, suggesting that the model is very good at ranking positive cases higher than negative cases.

Naive Bayes:

- Train and test accuracy are similar, but relatively low, suggesting that the model is not very good at predicting the outcome.
- Train and test precision are low, suggesting that the model is not very good at identifying either positive or negative cases.
- Train and test recall are low, suggesting that the model is not very good at identifying all positive cases, even if it means identifying some false positives.
- F1 score is very low, suggesting that the model has a very poor balance between precision and recall.
- AUC is slightly higher than 0.5, suggesting that the model is slightly better at ranking positive cases higher than negative cases than random guessing.

Decision Tree:

- Train accuracy is perfect, but test accuracy is0.92, suggesting that the model is overfitting the training data to a very small extent.
- Train and test precision are high, suggesting that the model is very good at identifying both positive and negative cases.
- Train and test recall are high, suggesting that the model is very good at identifying all positive cases, even if it means identifying some false positives.
- F1 score is very high, suggesting that the model has a very good balance between precision and recall.
- AUC is perfect, suggesting that the model is perfect at ranking positive cases higher than negative cases.

Model Tuning and Business Implication.

a. Ensemble Modelling:

Ensemble techniques are widely used in machine learning to improve model performance. They involve combining multiple models to make predictions more accurate and robust. By leveraging the diversity of different models, ensembles can reduce overfitting and enhance overall predictive power, making them a valuable tool for achieving high-quality results in various applications. We are building the following ensemble techniques.

Random Forest

To build the random forest model we are using the Random Forest Classifier from sklearn ensemble library. The details are as follows.

```
Classification Report on Train Data

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       582
           1       1.00      1.00      1.00       193

    accuracy                           1.00       775
   macro avg       1.00      1.00      1.00       775
weighted avg       1.00      1.00      1.00       775

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.98      0.98      0.98       250
           1       0.94      0.94      0.94        83

    accuracy                           0.97       333
   macro avg       0.96      0.96      0.96       333
weighted avg       0.97      0.97      0.97       333

-----------------------------------------------------

Confusion matrix on train data
[[582   0]
 [  0 193]]

Confusion matrix on test data
[[245   5]
 [  5  78]]
-----------------------------------------------------
```

Table: Evaluation Metric of Random Forest.

The AUC of train data is 1 and that of test is 0.997. The ROC Curve are given below.

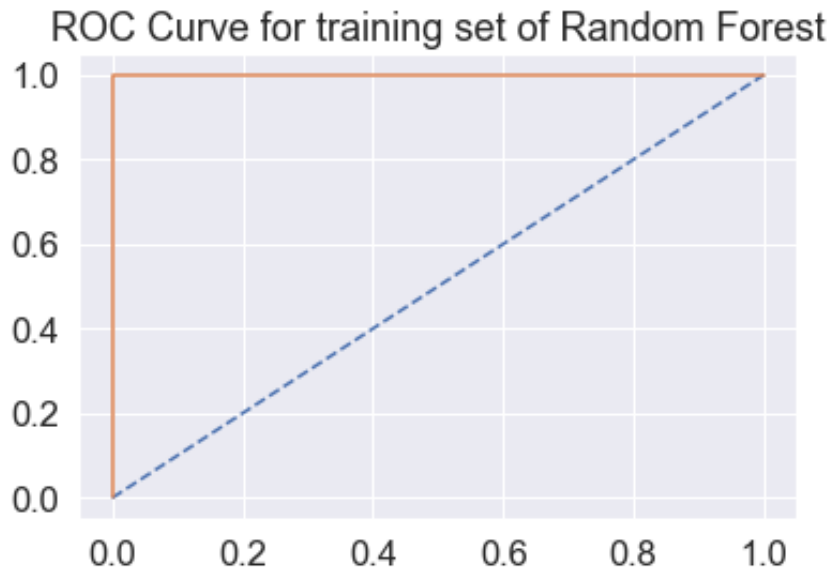

Figure: ROC of train data on Random Forest

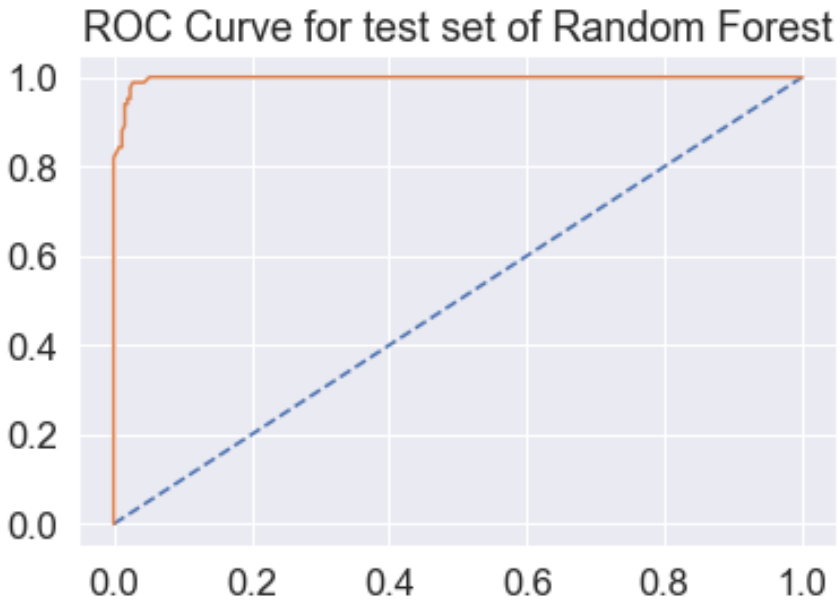

Figure: ROC of train data on Random Forest

The Random Forest model demonstrates impressive performance on the training data, achieving perfect accuracy, precision, recall, F1 score, and AUC. However, on the test data, it maintains a high accuracy of 96.997% but slightly lower precision, recall, F1 score, and AUC, indicating some overfitting. Overall, it's a robust model, although there's a small drop in performance when applied to unseen data.

Ada Boosting

We are using AdaBoost classifier from the sklearn ensemble library to build this mode. The details are as follows.

```
Classification Report on Train Data

              precision    recall  f1-score   support

           0       0.92      0.97      0.94       582
           1       0.90      0.74      0.81       193

    accuracy                           0.91       775
   macro avg       0.91      0.85      0.88       775
weighted avg       0.91      0.91      0.91       775

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.90      0.92      0.91       250
           1       0.75      0.69      0.72        83

    accuracy                           0.86       333
   macro avg       0.82      0.81      0.81       333
weighted avg       0.86      0.86      0.86       333

-----------------------------------------------------

Confusion matrix on train data
[[566  16]
 [ 51 142]]

Confusion matrix on test data
[[231  19]
 [ 26  57]]
-----------------------------------------------------
```

Table: Evaluation Metric of Ada Boost.

The AUC of train data is 0.96 and that of test is 0.912. The ROC curves are given below.

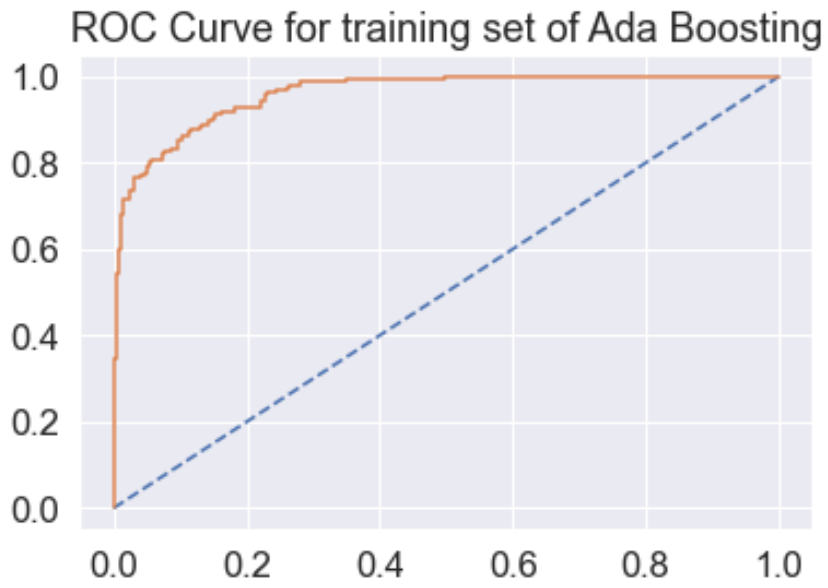

Figure: ROC of train on Ada boost model

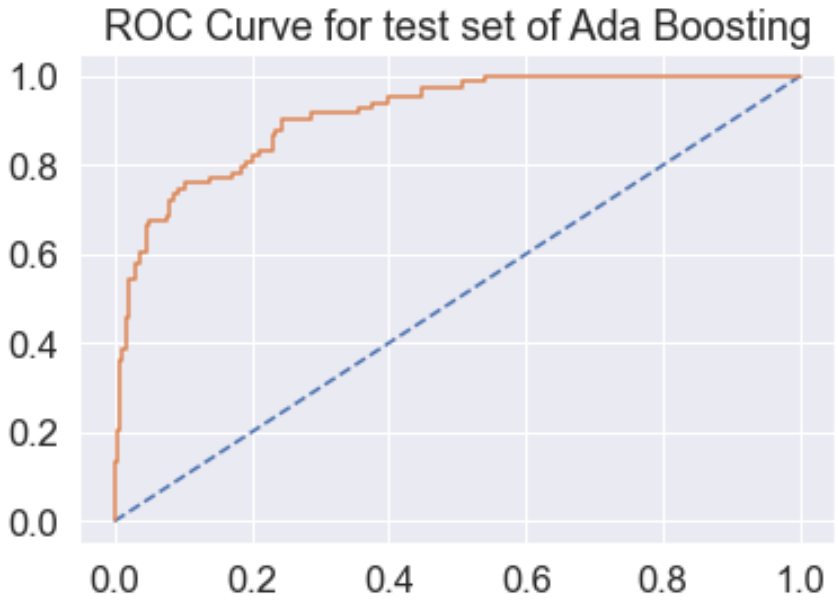

Figure: ROC of test on Ada boost model

The Ada Boost model demonstrates good overall performance, with a training accuracy of 91.35% and a testing accuracy of 86.49%. It shows strong precision in both training (89.87%) and testing (75%) and reasonable recall rates in training (73.58%) and testing (68.67%). The F1 score on training data is 0.809, indicating a good balance between precision and recall, while the test F1 score is 0.716. Additionally, the model exhibits excellent discrimination ability with an AUC of 0.960 for training and 0.911 for testing, suggesting it can effectively distinguish between positive and negative cases.

Gradient Boosting

We are using Gradient Boost classifier from the sklearn ensemble library to build this model. The details are as follows.

```
Classification Report on Train Data

              precision    recall  f1-score   support

           0       0.98      1.00      0.99       582
           1       1.00      0.95      0.97       193

    accuracy                           0.99       775
   macro avg       0.99      0.97      0.98       775
weighted avg       0.99      0.99      0.99       775

Classification Report on Test Data

              precision    recall  f1-score   support

           0       0.95      0.99      0.97       250
           1       0.97      0.86      0.91        83

    accuracy                           0.96       333
   macro avg       0.96      0.92      0.94       333
weighted avg       0.96      0.96      0.96       333

-----------------------------------------------------

Confusion matrix on train data
[[582   0]
 [ 10 183]]

Confusion matrix on test data
[[248   2]
 [ 12  71]]
-----------------------------------------------------
```

Table: Evaluation Metric of Gradient Boost.

The AUC of train data is 1 and that of test is 0.993. The ROC curves are given below.

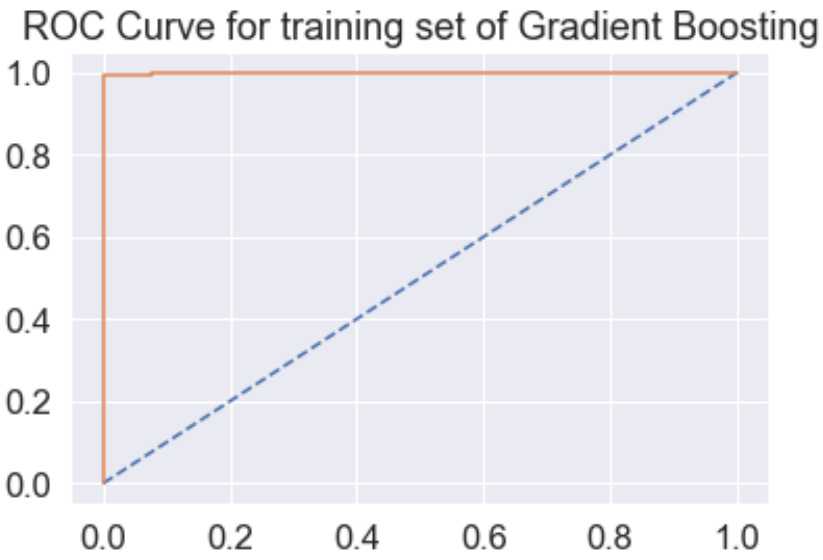

Figure: ROC of train on Gradient boost model

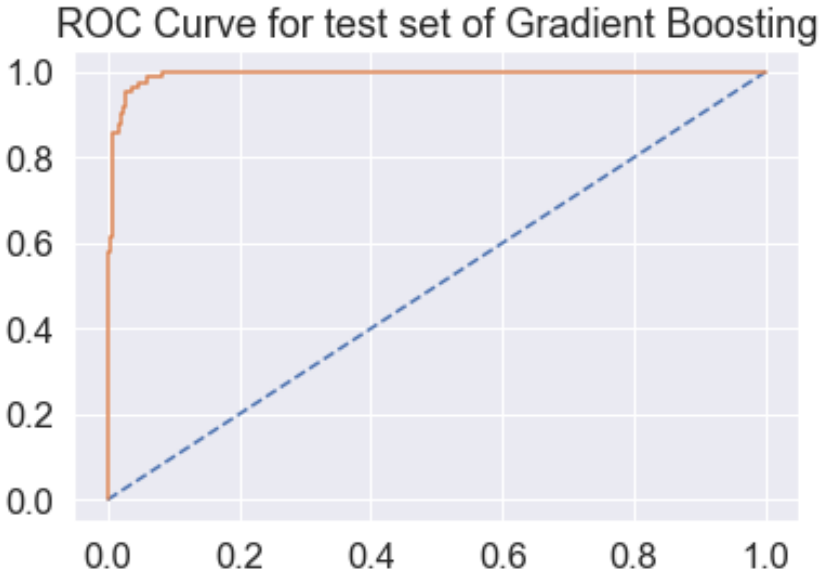

Figure: ROC of test on Gradient boost model

The model is Gradient Boost, and it exhibits high accuracy on both the training and test datasets, with values of approximately 98.7% and 95.8%, respectively. It demonstrates excellent precision on the training data (100%) and good precision on the test data (97.3%). The model shows strong recall on the training set (94.8%), although it could be improved on the test set (85.5%). The F1 scores are high for both training (97.3%) and test (91.0%), indicating a well-balanced trade-off between precision and recall. Additionally, the AUC values suggest excellent discriminative power, particularly on the test dataset.

Bagging

We are using Bagging classifier from the sklearn ensemble library to build this model. The details are as follows.

Classification Report on Train Data

	precision	recall	f1-score	support
0	1.00	1.00	1.00	582
1	1.00	0.99	0.99	193
accuracy			1.00	775
macro avg	1.00	0.99	1.00	775
weighted avg	1.00	1.00	1.00	775

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.94	0.98	0.96	250
1	0.93	0.80	0.86	83
accuracy			0.93	333
macro avg	0.93	0.89	0.91	333
weighted avg	0.93	0.93	0.93	333

Confusion matrix on train data

```
[[582   0]
 [  2 191]]
```

Confusion matrix on test data

```
[[245   5]
 [ 17  66]]
```

Table: Evaluation Metric of Bagging

The AUC of train data is 1 and that of test is 0.991. The ROC curves are given below.

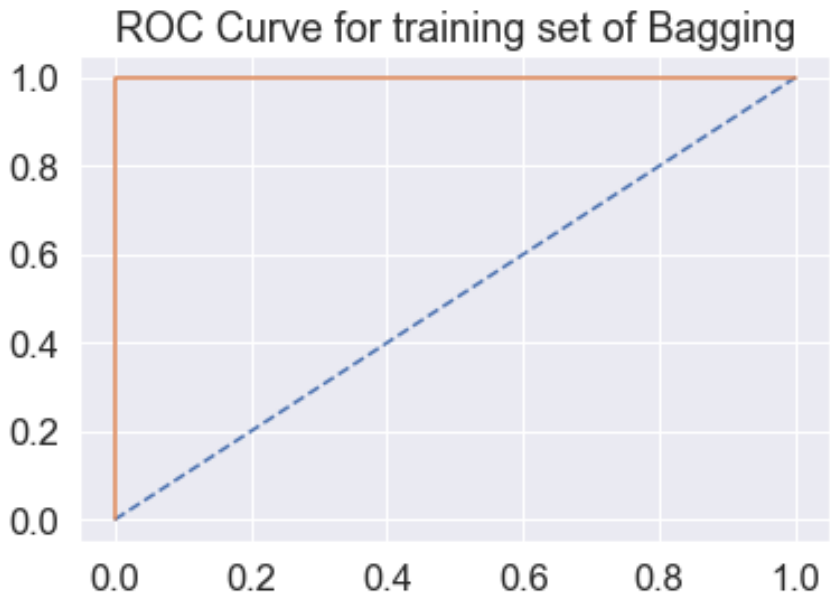

Figure: ROC of train on bagging model

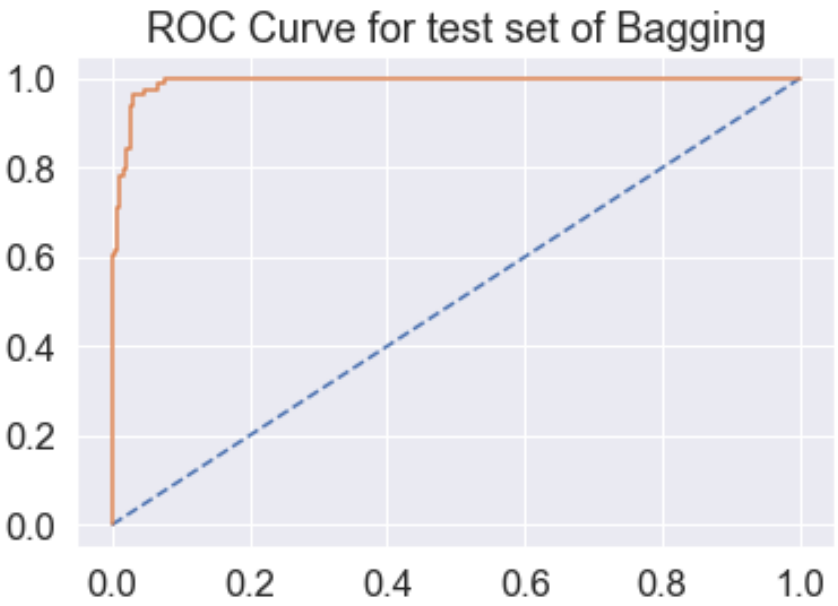

Figure: ROC of test on bagging model

The model shows strong performance in terms of accuracy, precision, recall, F1 score, and AUC. It achieves a high training accuracy of 99.74% and a test accuracy of 93.39%, indicating good generalization. Precision and recall are also impressive, with 100% precision on the training data and 92.96% on the test data. The model excels in capturing positive instances with a high recall, although there is some room for improvement in test recall. Overall, this model is promising, but some fine-tuning may be needed to enhance its performance further.

b. **Model Tuning:**

The K-nearest neighbors (KNN) model was fine-tuned with parameters, and the optimal configuration was found to be 'n_neighbors' = 1. The other parameters being 'algorithm' = 'auto' and 'weights' = 'uniform', which were already the default values. A comparison of the original KNN model and the tuned KNN model revealed that the tuned KNN model exhibited remarkable improvements in accuracy, precision, recall, F1 score, and AUC, achieving perfect training accuracy and precision. On the test data, the tuned KNN model demonstrated a significantly higher accuracy of 99.10%, precise identification of positive cases (97.62%), and improved recall (98.80%). This suggests that the tuned KNN model is more adept at generalization and overall superior in classification performance compared to the original KNN model, making it the recommended choice for this specific task.

Similarly, all the other models were also tuned. For Logistic Regression, we adjusted parameters related to how the model learns and found that it works best when we use 'C' at 1.0, 'penalty' as 'l1,' and 'solver' set to 'liblinear.' In the case of Linear Discriminant Analysis (LDA), we selected 'solver' as 'svd' to optimize its performance. The Random Forest model was fine-tuned to have 'n_estimators' set to 50, 'max_depth' at 10, 'min_samples_split' at 15, and 'min_samples_leaf' at 15. For Bagging, 'n_estimators' was set to 30 for optimal results. Our AdaBoost model performed best with 'algorithm' set to 'SAMME.R' and 'n_estimators' at 60. Lastly, the Gradient Boost model excelled with 'n_estimators' set to 200, 'learning_rate' at 0.1, and 'max_depth' set to 4. These adjustments help each model perform its best on our specific task, ensuring more accurate and efficient predictions.