-

# PREDICTION OF AIRLINE PASSENGER SATISFACTION

CAPSTONE PROJECT

Submitted in partial fulfilment of the requirements of the

Post Graduate Certification Program
in
Artificial Intelligence and Machine Learning

By


| Name | Bits ID |
|---|---|
| Jagdish Yalla | 2021aiml064 |
| Robin Mathew | 2021aiml003 |
| Deepa Krishnaswami | 2021aiml049 |
| Abhishek Agarwal | 2021aiml045 |


Under the supervision of

Mr. Satyaki Dasgupta

Project work carried out at


BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) INDIA


August, 2022

## ACKNOWLEDGEMENTS

A project like **PREDICTION OF AIRLINE PASSENGER SATISFACTION** or any such project needs in depth understanding of Data Analytics technics, mathematics behind it, various machine learning models available and the way to implement them, reasoning ability to choose a right technique for a given problem statement, ability to apply all the techniques through a language like Python.

Thank you to **Dr. N.L.Bhanu Murthy** for being a pillar support for the AIML program and layingthe foundation required through the basic mathematics and Regression Videos
Thank you to **Prof**. **Yvk Ravi Kumar** for being a pillar support for the AIML program and layingthe foundation required through the basic mathematics and Regression lectures
Thank you to **Prof**. **Ramakrishna Dantu** for being a pillar support for the AIML program and laying thefoundation required through the Feature Engineering lectures
Thank you to **Dr. Chetana Gavankar**, for covering the core concepts of Classification and Text mining topics in depth, taking time to clear all the clarifications and ensuring that every student understood the concepts to the core.
Thank you to **Prof. Swarna Choudhary**, for covering the core concepts of Classification and Unsupervised Learning topics in depth, taking time to clear all the clarifications and ensuring that every student understood the concepts to the core.
Thank you to **Prof. Srikanth Gunturu** for covering the classification concepts clearly through thevideos, making them as elaborate and simple as possible.
Thank you to **Prof. Aruna Malapati** for covering the core concepts of Feature Engineering and Text Mining through Videos, making the videos extensive and easy to understand.
Thank you to **Prof. S.P. Vimal** for laying the foundations through elaborate introductory Pythonvideos and for covering core concepts of Unsupervised Learning.
Thank you to **Prof. Raja Vadhana P** for clearly covering the topics related to Unsupervised Learning, Association rulemining through sessions.
Thank you to **Prof. Kamlesh Tiwari** for covering with detailed videos on Deep learning and Artificial Neural networks.
Thank you to **Dr. Sugata Ghosal** for further covering Deep learning and Artificial Neuralnetworks in depth with elaborate sessions and helping with multitude sources of information/study material
Thank you to **Prof**. **Seetha Parameswaran** for further covering Deep learning and Artificial Neuralnetworks in depth with elaborate sessions and helping with multitude sources of information/study material

Thank you to TAs – **Murtuza Dahodwala, Bulla Radhika, Shristy Kapoor, Chinesh Doshi and Partha Sarathy PD** for supporting at each stage with additional reference material, clarifyingdoubts through additional sessions and assignments

Thank you to **Mr. Satyaki Das Gupta** for guiding us throughout the project in step by step manner, helping us to unearth many new dimensions in data analytics and especially in solving the problems involving prediction of airline passenger satisfaction.

Thank you to all fellow students who were available for various technical discussions, helping us explore and learn the machine learning concepts even better.

A big Thank you to the Organizations we work for and our family members for bearing with us and being the support in the backend for all these months allowing us to spend extended time on the learning, assignments and the final project.

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**CERTIFICATE**

This is to certify that the Capstone Project entitled

**PREDICTION OF AIRLINE PASSENGER SATISFACTION**

and submitted by Mr./Ms. **Jagdish Yalla, Robin Mathew, Deepa Krishnaswami, Abhishek Agarwal** ID No. **2021aiml064, 2021aiml003, 2021aiml049, 2021aiml045**

in partial fulfilment of the requirements of PCAM ZC321 Capstone Project, embodies the work done by him/her under my supervision.

Place : Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date : 08/Oct/2022

Name : Mr. Satyaki Dasgupta

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

## CERTIFICATE

This is to certify that the Capstone Project entitled **PREDICTION OF AIRLINE PASSENGER SATISFACTION**

and submitted by **Mr. Jagdish Yalla** ID No. **2021aiml064**

in partial fulfilment of the requirements of PCAM ZC321 Capstone Project, embodies the work done by him under my supervision.

Place : Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date : 08/Oct/2022

Name : Mr. Satyaki Dasgupta

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**CERTIFICATE**

This is to certify that the Capstone Project entitled **PREDICTION OF AIRLINE PASSENGER SATISFACTION**

and submitted by **Mr. Robin Mathew** ID No. **2021aiml003**

in partial fulfilment of the requirements of PCAM ZC321 Capstone Project, embodies the work done by him under my supervision.

Place : Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date : 08/Oct/2022

Name : Mr. Satyaki Dasgupta

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

## CERTIFICATE

This is to certify that the Capstone Project entitled **PREDICTION OF AIRLINE PASSENGER SATISFACTION**

and submitted by **Mrs. Deepa Krishnaswami** ID No. **2021aiml049**

in partial fulfilment of the requirements of PCAM ZC321 Capstone Project, embodies the work done by him under my supervision.

Place : Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date : 08/Oct/2022

Name : Mr. Satyaki Dasgupta

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**CERTIFICATE**

This is to certify that the Capstone Project entitled **PREDICTION OF AIRLINE PASSENGER SATISFACTION**

and submitted by **Mr. Abhishek Agarwal** ID No. **2021aiml045**

in partial fulfilment of the requirements of PCAM ZC321 Capstone Project, embodies the work done by him under my supervision.

Place : Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date : 08/Oct/2022

Name : Mr. Satyaki Dasgupta

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI SECOND SEMESTER 2021-22**

**PCAM ZC321 CAPSTONE PROJECT**

| | |
|---|---|
| Project Title | **PREDICTION OF AIRLINE PASSENGER SATISFACTION** |
| Name of Mentor | Mr. SATYAKI DASGUPTA |
| Name of Student | Jagdish Yalla, Robin Mathew, Deepa Krishnaswami, Abhishek Agarwal |
| ID No. of Student | 2021aiml064, 2021aiml003, 2021aiml049, 2021aiml045 |

**Abstract**

- **Problem Statement**: Measuring customer satisfaction is a key element for modern businesses as it can significantly contribute to a continuing effort of service quality improvement. In order to meet customer expectations and achieve higher quality levels, airlines need to develop a specific mechanism of passenger satisfaction measurement.
- Customer satisfaction is always top of mind for airlines, unhappy or disengaged customers naturally mean fewer passengers and less revenue.
- On-time flights, good in-flight entertainment, more (and better) snacks, and more legroom are the obvious and traditional contributors to a good experience and more loyalty.
- The advent of digital technologies and never than before emphasis on customer experience, its no more about just the flight itself, its everything from purchasing the ticket on the company's website or mobile app to checking bags in at the airport or via a mobile app to waiting in the terminal.
- Self-service has been top-of-mind for airlines since the introduction of airport kiosks that enable passengers to check-in, upgrade their seats, and even make flight changes. This mindset has been, and continues to be, adapted to the post-security, onboard, and post-flight experience.

Timely Departure and Arrival

Customer Service at Airport and Inflight Entertainment

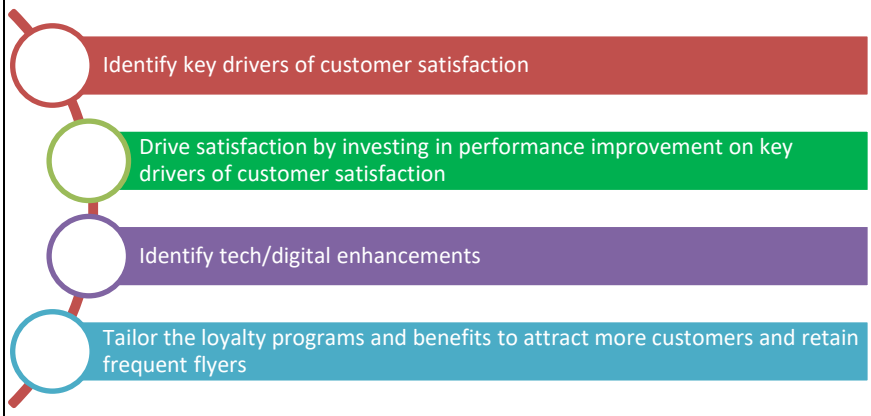
Seat Comfort, Leg Room, Food & Beverages

Baggage Handling, Arrival , Fees

Ticket Prices, Loyalty Programs, Upgrades

Priority Boarding fees,Early Check in fees, Pet Charge

## Objective




- Predict
  - Whether passenger will be satisfied or not
- Data : Publicly sourced data
  - Dataset consists of 100,000+ rows and 25 columns

| | |
|---|---|
| **Key Words:** | Airline Satisfaction prediction, Regression, Lag variables, Data Cleansing, Exploratory Analysis, Correlation, IVF, PCA, Feature Importance, Outlier Detection, KNN, DBCSAN, LOF, Imputation, Linear Regression, Logistic Regression, H2O, Neural Networks, Ensemble, Naive Bayes, KNN for Modeling, AdaBoost, XG Boost, Bagging, Random Forest |

## List of Abbreviations used

| Abbreviation | Meaning/Usage |
|---|---|
| VIF | Variance Inflation Factor |
| NN | Neural Network |
| DBSCAN | Density Based Spatial Clustering of Applications with Noise |
| GBM | Gradient Boosting Model |
| KNN | K Nearest Neighbor |
| LOF | Local Outlier Factor |
| MLP | Multi Layer Perceptron |
| PCA | Principal Component Analysis |
| RMSE | Root Mean Square Error |
| SVM | Support Vector Machine |
| XGBoost | eXtreme Gradient Boosting |

*Table 1:List of Abbreviations*

**Table Of Contents:**

## 1. Problem statement (what is the problem being addressed)

Measuring customer satisfaction is a key element for modern businesses as it can significantly contribute to a continuing effort of service quality improvement. In order to meet customer expectations and achieve higher quality levels, airlines need to develop a specific mechanism of passenger satisfaction measurement.

## 2. Objective of the project

- Predict
  - Whether passenger will be satisfied or not
- Data: Publicly sourced data
  - Dataset consists of 100,000+ rows and 25 columns

- Identify key drivers of customer satisfaction
- Drive satisfaction by investing in performance improvement on key drivers of customer satisfaction
- Identify tech/digital enhancements
- Tailor the loyalty programs and benefits to attract more customers and retain frequent flyers

## 3. Background of previous work done in the chosen area (Literature Review)

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0224-1

https://www.researchgate.net/publication/350552031_Predicting_Airline_Passenger_Satisfaction_with_Classification_Algorithms

s41598-022-14566-3.pdf

https://www.nature.com/articles/s41598-022-14566-3

https://www.rcs.cic.ipn.mx/rcs/2019_148_6/Predicting%20Airline%20Customer%20Satisfaction%20using%20k-nn%20Ensemble%20Regression%20Models.pdf

http://proteusresearch.org/gallery/4-june2022.pdf

80-221-4-PB.pdf

## 4. Workflow

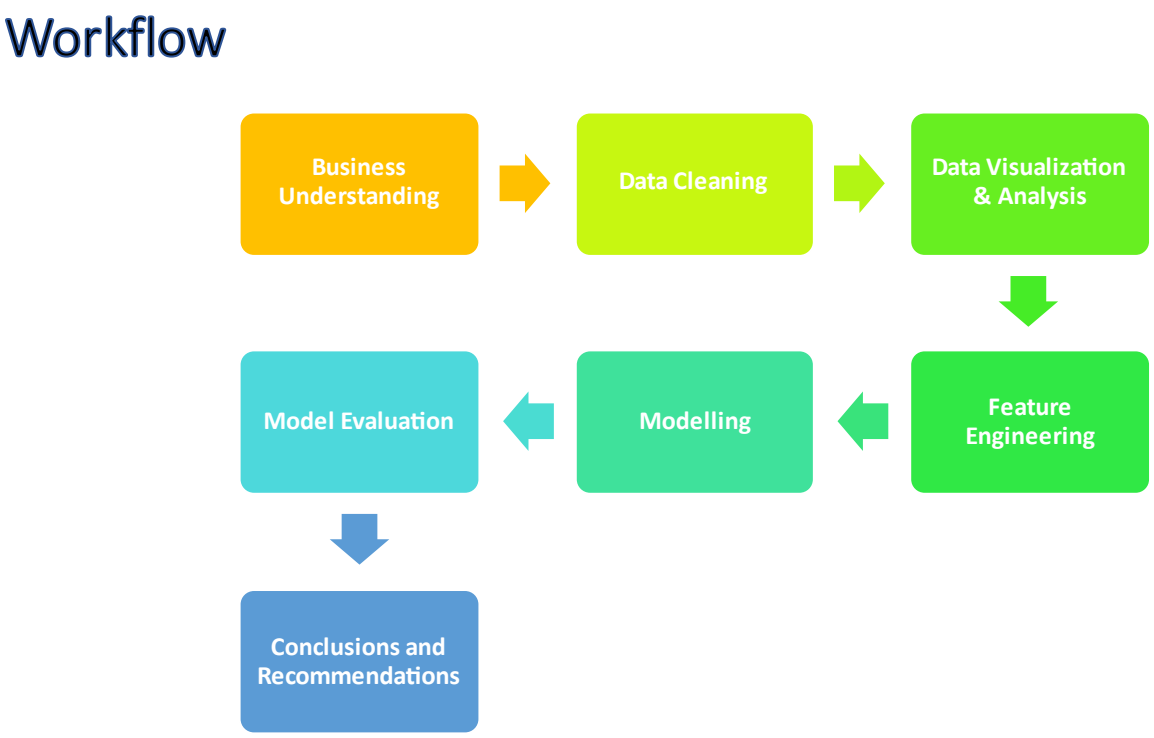



*Figure 1: Process flow*

## 5. Data Overview

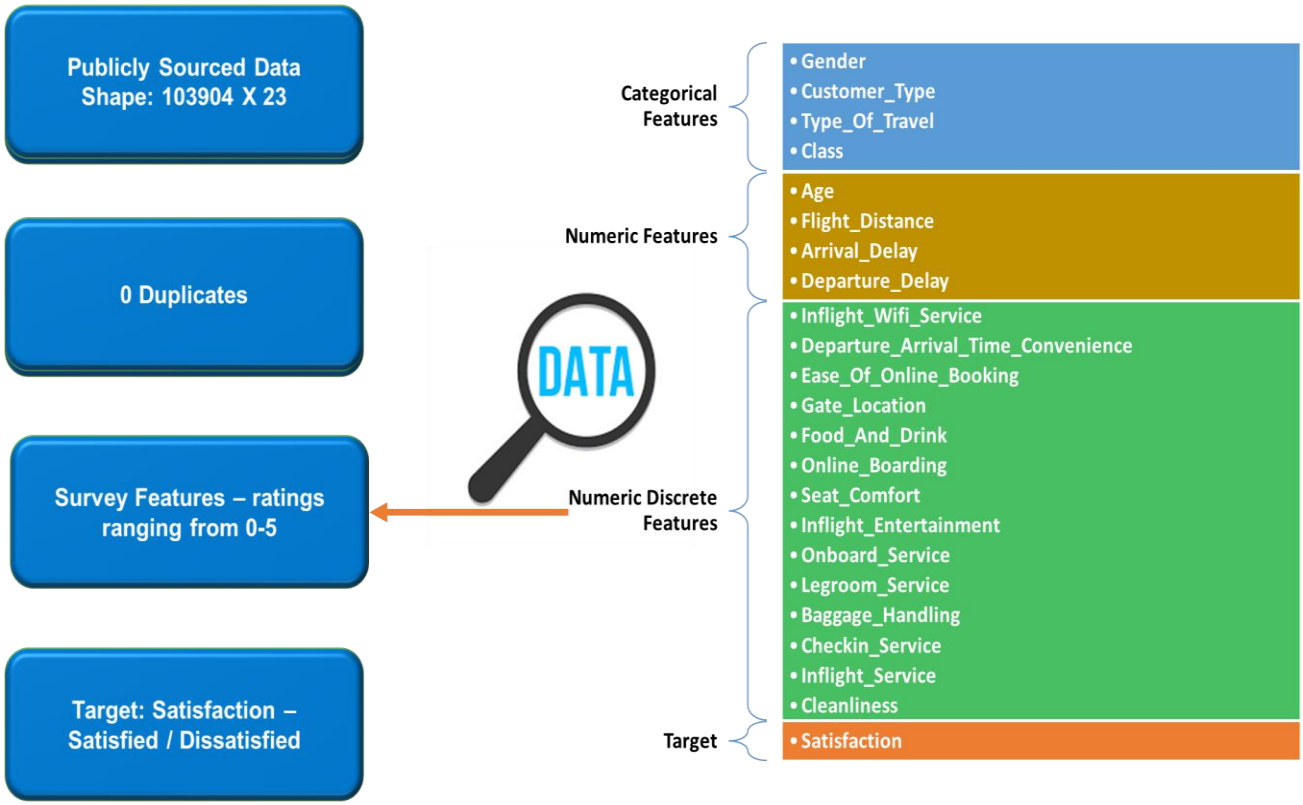



*Figure 2: Data Overview 1*

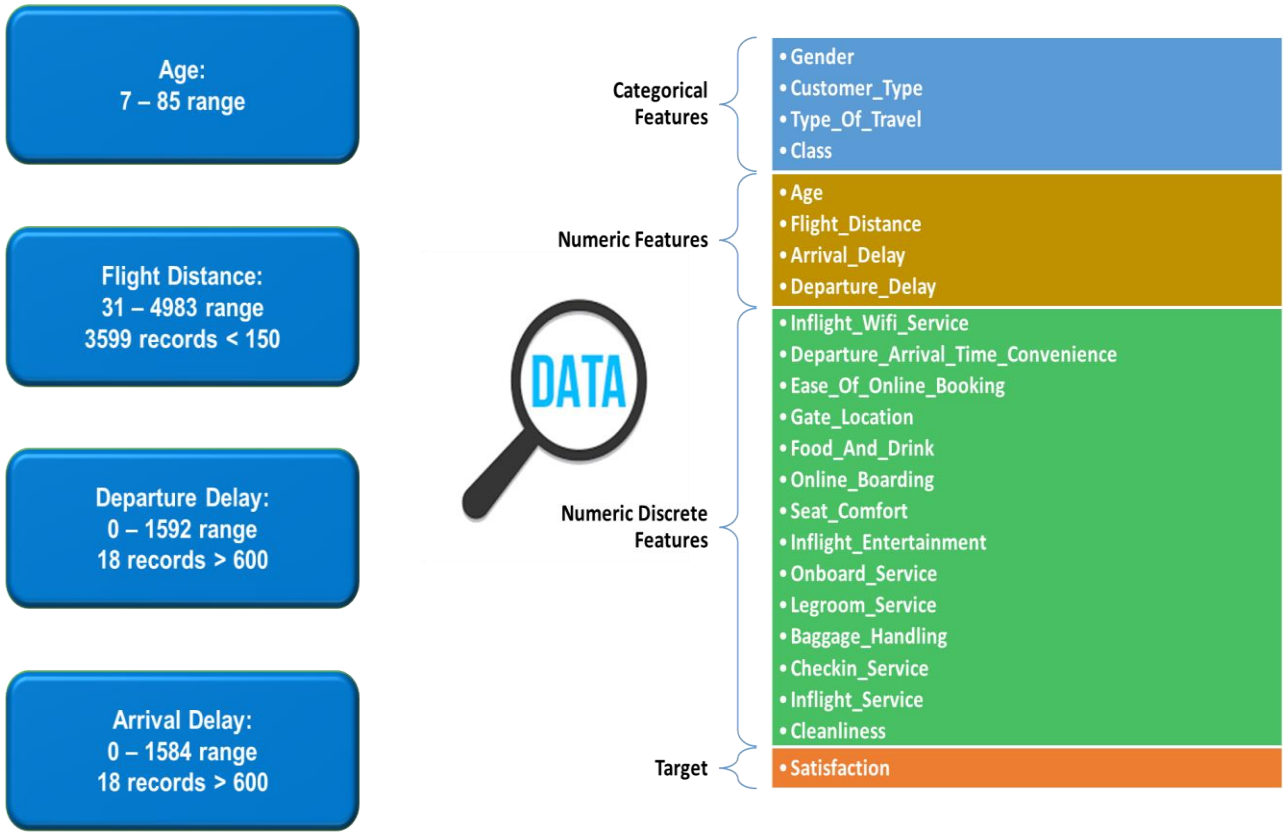


Figure 3: Data Overview Continued

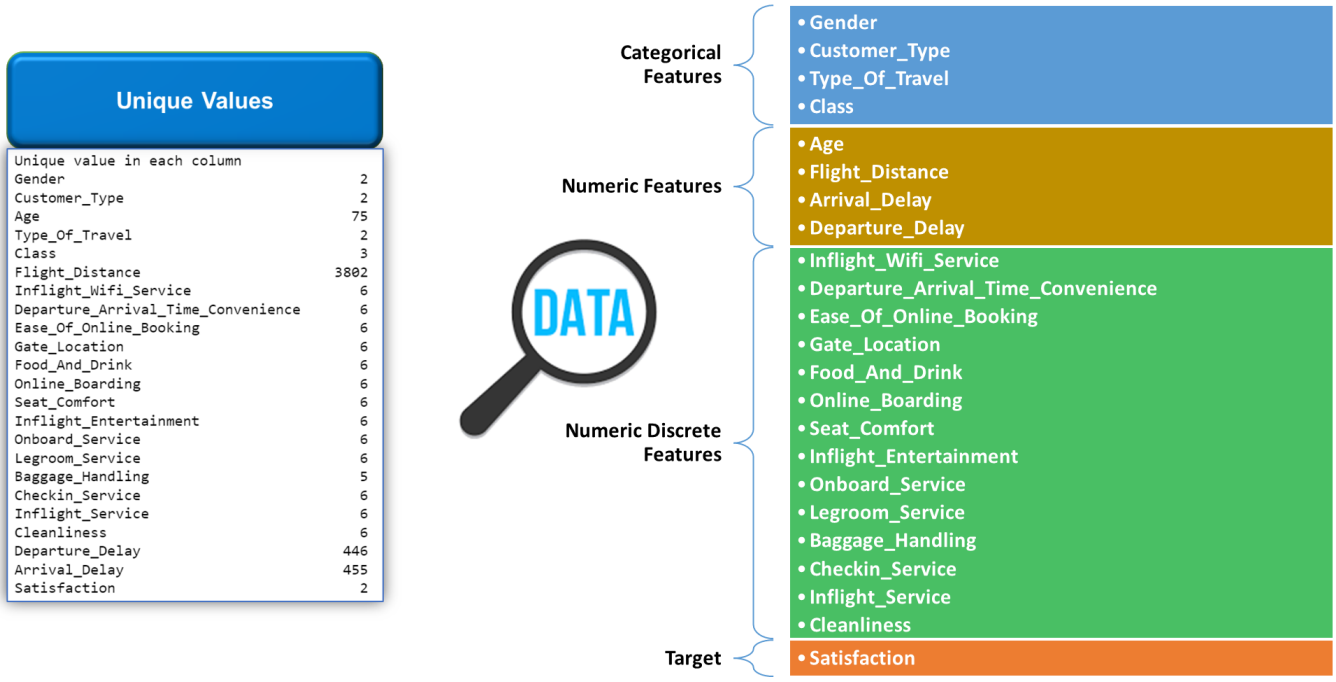



Figure 4: Data Overview Conti

## 6. Resources needed for the project, including people, hardware, software, etc.

The resources needed for the project can be classified in to four broad categories – Information,Technology, Process and People

- Information
  - The past data that can be used as an input for various machine learning models, after thorough pre-processing, inference and visualization of the underlying information
  - Business knowledge of what all could impact Airline Satisfaction
  - Knowledge of various tools and machine learning techniques that can be applied to be able to understand the data available better, zero down on the right machine learning models that can be applied and how to tune them to be able to predict the required parameters with the at most accuracy

- Technology
  - Data storage at a common location accessible to all the team members
  - Python run time environments – Google Colab /Jupyter/Spyder

- Process
  - Business Understanding
  - Project Planning and Execution
  - Resource management
    - Version Management of the code
    - Work distribution and collaboration
  - Data Collection
  - Data Pre-processing
    - Data Merger from various sources
    - Data Normalization
    - Outlier Analysis
    - Missing Values Imputation
  - Data Inference
    - Uni Variate Analysis
    - Multi Variate Analysis
    - Trend Analysis
    - Data Correlation Analysis

- People
  - A guide/Mentor to direct the team in right direction so that learnt knowledge can be applied in an efficient
  - A team of members with skillsets: SME, Data Scientist, Python Developer, SQL Developer, AI/ML Expert and Tester

## 7. Potential data challenges & risks in doing the project

### 7.1. Data Challenges

Possible missing critical data or noise example: an unplanned event during flying time or at the airport or national or international emergency situations not captured there by resulting in outliers/noisy data

### 7.2. Risks

Un availability of data related to special situations as described above. New aviation standards or potential disruptions by enhanced or degraded situations – example Covid 19 pandemic created a special situation for the airline industry and this is no captured in the data available for modeling.

## 8. Detailed Plan of Work

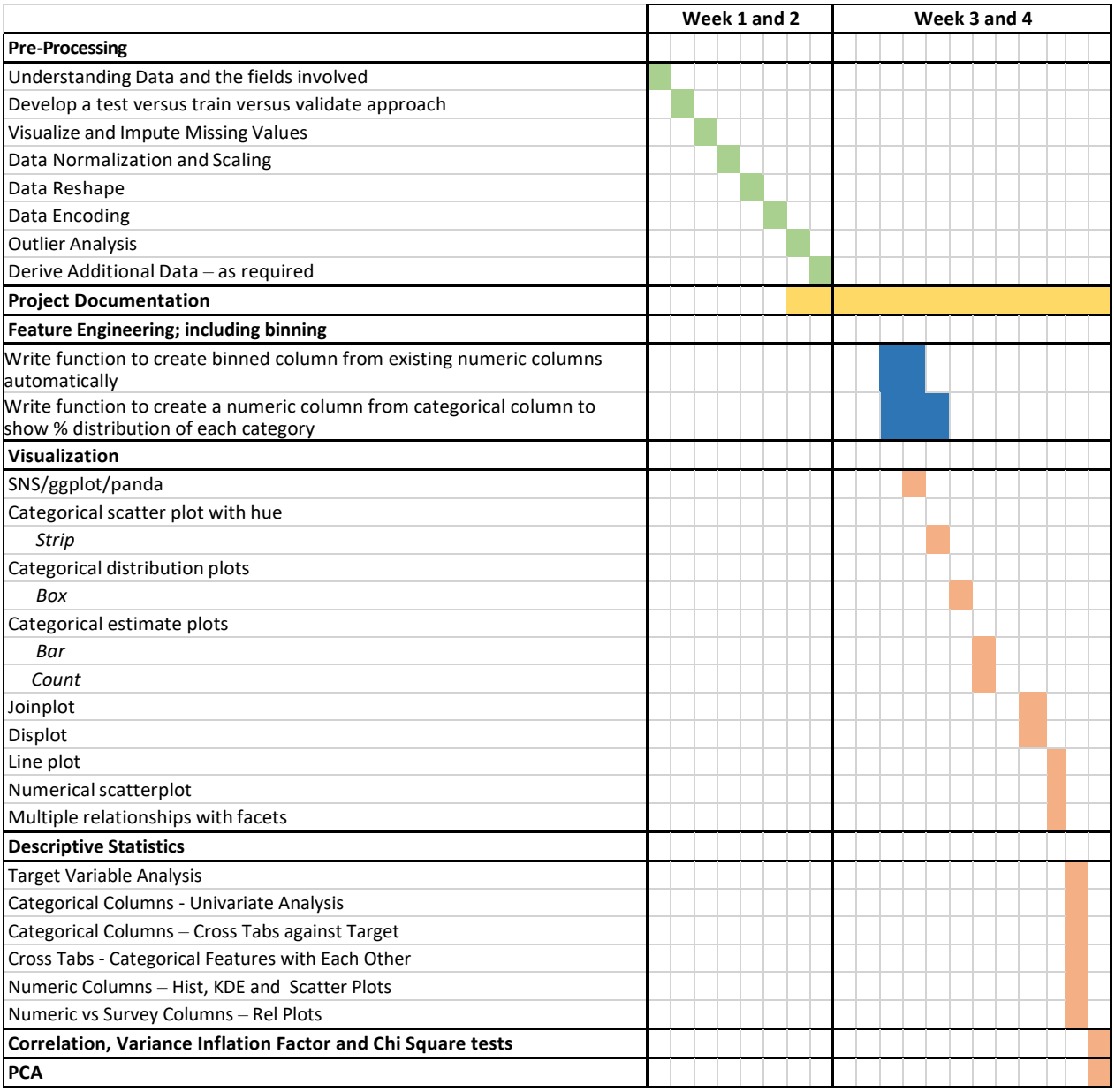



*Figure 3: Execution Plan week 1 to 4*

| Prepare for Modelling | Week 5 | Week 6 | Week 7 and 8 |
|---|---|---|---|
| Identify and prepare train, test and validate data | | | |
| Finalize the Model Tuning Techniques | | | |
| Finalize the Model Parameters | | | |
| Prepare Datasets for Models | | | |
| **Apply Models** | | | |
| Standard Classification Models | | | |
| Standard Ensemble Models | | | |
| Advance NN Models | | | |
| **Analyze Model Metrics** | | | |
| **Project Deployment** | | | |
| **Project Documentation** | | | |

*Figure 4: Execution Plan week 6 to 8*

## 9 Data Acquisition

Data Acquisition will be useful to train correctly your training model. While acquiring the data be sure to have enough features.

The given csv file datasets are read into multiple Panda data frames.
Overall Approach –

- Use training data to examine variable patterns, draw inferences and do feature selection and engineering.
- Train data is available as a part of initial files.
- Test data is reserved for use later.
- The preprocessing done on train will be replicated on test, while feature selection/engineering/modeling iterations will be limited to train only.

# Load Data

```
# Read Training Data
airlineTrainingData = readDataFile_CSV("train.csv")
airlineData=airlineTrainingData.copy()
#airlineTrainingData=addColumn(airlineTrainingData,'FileType','train')

# Read Testing Data
#airlineTestingData = readDataFile_CSV("test.csv")
#airlineTestingData=addColumn(airlineTestingData,'FileType','test')

# Combine Training and Testing Data
#airlineData = mergeDataFrames([airlineTrainingData,airlineTestingData])
```

*Figure 6: Code Snippet - Reading the data to the Data Frame*

### 9.1 Examine Data and Drop unwanted columns, and rename as required

We have a total of 103904 rows in our training data set.
We are trying to predict "Airline Satisfaction" = "TargetColumn", which is a binary variable.
Several predictor variables are available: Survey columns (numeric discrete), other numeric columns and categorical variables like gender etc.
Following columns are dropped : 'Unnamed: 0', 'id'

**Original Airline Data:**

```
airlineData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 25 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   Unnamed: 0                        103904 non-null  int64
 1   id                                103904 non-null  int64
 2   Gender                            103904 non-null  object
 3   Customer Type                     103904 non-null  object
 4   Age                               103904 non-null  int64
 5   Type of Travel                    103904 non-null  object
 6   Class                             103904 non-null  object
 7   Flight Distance                   103904 non-null  int64
 8   Inflight wifi service             103904 non-null  int64
 9   Departure/Arrival time convenient 103904 non-null  int64
 10  Ease of Online booking            103904 non-null  int64
 11  Gate location                     103904 non-null  int64
 12  Food and drink                    103904 non-null  int64
 13  Online boarding                   103904 non-null  int64
 14  Seat comfort                      103904 non-null  int64
 15  Inflight entertainment            103904 non-null  int64
 16  On-board service                  103904 non-null  int64
 17  Leg room service                  103904 non-null  int64
 18  Baggage handling                  103904 non-null  int64
 19  Checkin service                   103904 non-null  int64
 20  Inflight service                  103904 non-null  int64
 21  Cleanliness                       103904 non-null  int64
 22  Departure Delay in Minutes        103904 non-null  int64
 23  Arrival Delay in Minutes          103594 non-null  float64
 24  satisfaction                      103904 non-null  object
dtypes: float64(1), int64(19), object(5)
memory usage: 19.8+ MB
```

*Figure 7: Original data snapshot*

Make Basic Observations about available data

- Gender, Customer_Type, Type_Of_Travel, Class are categorical features
- Satisfaction is the target label which holds 2 target values - binary classification
- Following features represent ratings [0-5] given by passengers as feedback for various flight amenities/services.
  Inflight_Wifi_Service,Departure_Arrival_Time_Convenience,Ease_Of_Online_Booking,Gate_Location,Inflight_Entertainment,Onboard_Service,Legroom_Service,Baggage_Handling,Checkin_Service Inflight_Service,Cleanliness
- Age, Flight_Distance, Arrival and Departure Delay are numerical features.
- Age ranges between 7 to 85 which is a valid age group for flight passengers.
- For Flight_Distance, Departure_Delay and Arrival_Delay, we see discrepancies in min-max values - outliers or noise in data.

## 9.2 Outlier Analysis

Assuming an airline company flight would be for distance > 150 (less than this would be a private flight).

Assuming Departure Delays would not be greater than 10 hours.

For the above assumptions, we see a minor % of data as noise. These records can be further compared against outliers during outlier detection and we can decide whether to drop these or consider as extreme condition scenarios

## 9.3 Duplicate Data Analysis

There are no duplicates found from dataset. No further analysis on duplicates required.

## 9.4 Target Variable Analysis


This is a  binary class prediction problem. We want to check if we have adequate representation of each class in the data set. Here, the 2 class comprise 57% and 43% of the train data set hence we can see there is no class imbalance problem. No further analysis or sampling required at this point.

### 9.5 Missing Value Analysis

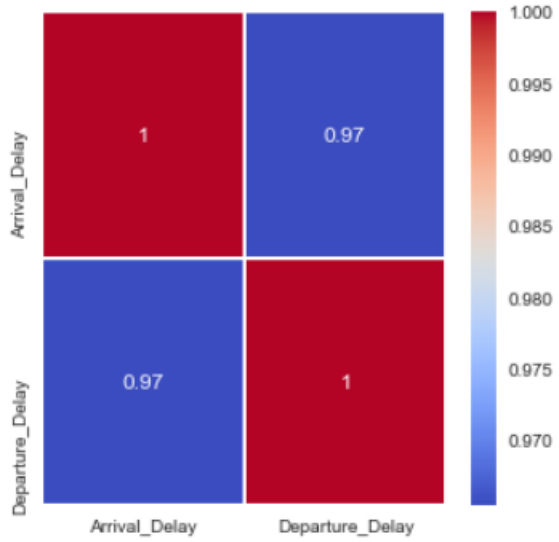
```
displayMissingValues(airlineData)
```

```
Missing Values:
-----------------
Gender                                  0
Customer_Type                           0
Age                                     0
Type_Of_Travel                          0
Class                                   0
Flight_Distance                         0
Inflight_Wifi_Service                   0
Departure_Arrival_Time_Convenience      0
Ease_Of_Online_Booking                  0
Gate_Location                           0
Food_And_Drink                          0
Online_Boarding                         0
Seat_Comfort                            0
Inflight_Entertainment                  0
Onboard_Service                         0
Legroom_Service                         0
Baggage_Handling                        0
Checkin_Service                         0
Inflight_Service                        0
Cleanliness                             0
Departure_Delay                         0
Arrival_Delay                         310
Satisfaction                            0
dtype: int64
```

- Arrival Delay has 393 values missing from dataset. We can try with different imputation methods after analysing the data.
- Since arrival delay has missing values, we need to check for values present in arrival delay column to understand this data further.
- Average delay is about 15 mins.
- Maximum delay is 1584 mins which is 26 hrs.
- We need to look at outlier analysis (covered later) to understand how to treat this data point.
- We will remove this record if it found as outlier.

*Figure 14: Missing Value Analysis*

```
displayHeatMap(5,5,airlineData[['Arrival_Delay','Departure_Delay']])
```




Clearly we can see arrival delay and departure delay are highly correlated with correlation coefficient of 0.97.
We will drop arrival delay from data set which also solve the problem of missing values imputation.

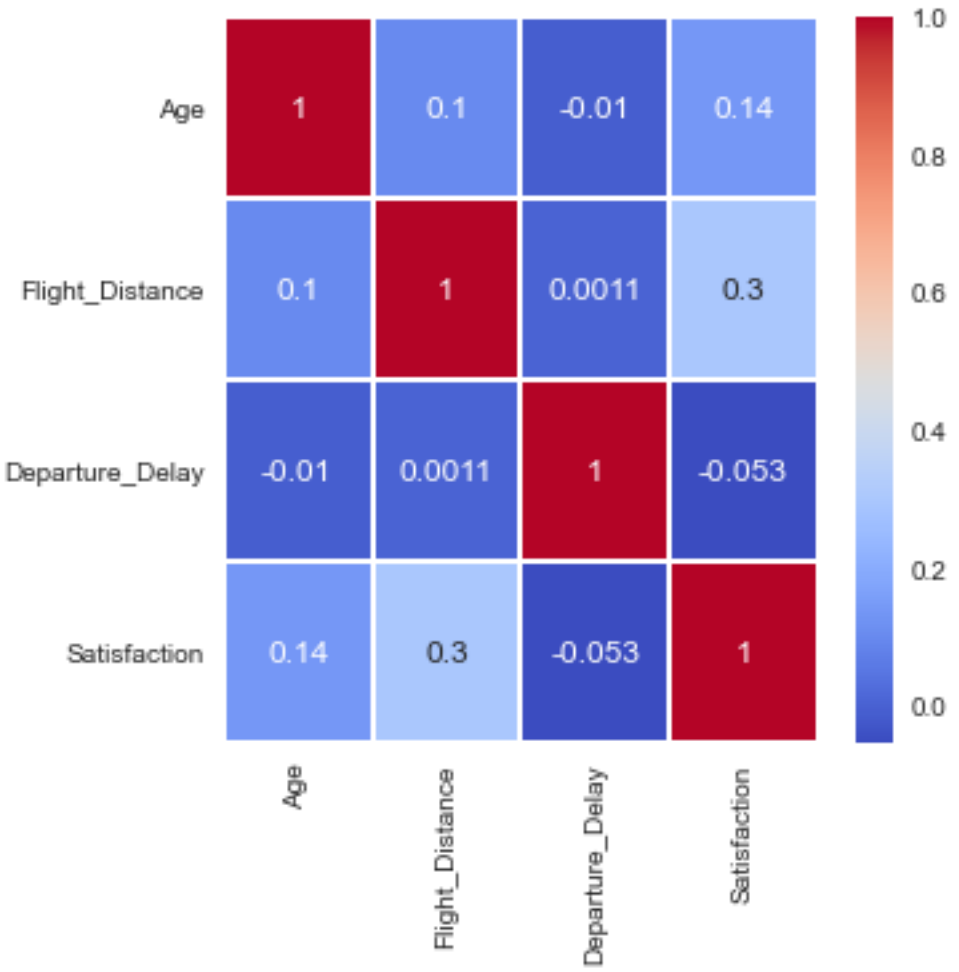
## 9.6 Feature Selection

### 9.6.1 Correlation among continuous variables and dependent

```
temp=airlineData.copy()
columnsToEncode=airlineData_NumericColumns.copy()
columnsToEncode.append(targetColumn)
temp=performLabelEncoding(temp, columnsToEncode, False, False)

displayHeatMap(5,5,temp[columnsToEncode])
temp=displayCorrelationInfo(temp[columnsToEncode])
if temp.shape[0]!=0:
    display(temp)
else:
    print('No  of features have correlation > 0.8')
```

```
No  of features have correlation > 0.8
```

**Correlation using HeatMap (Numerical Predictors)**




- Observations from heatmap: Only a few limited set of variables are showing correlation among themselves.
- However, we need to look at multicollinearity in a multidimensional way - for which we will use VIF
- VIF or Variance Inflation Factor will guide feature selection among numeric variables

*Figure 16: Normalize Data set*

### 9.6.2 Multicollinearity Check using Variance Inflation Factor (VIF)

Variance inflation factor measures how much the behaviour (variance) of an independent variable is influenced, or inflated, by its interaction/correlation with the other **independent variables. It is a measure of multivariate corelations.**.

```
temp=airlineData.copy()
tempNumericColumns=airlineData_NumericColumns.copy()
tempNumericDiscreteColumns=airlineData_NumericDiscreteColumns.copy()

numericVIF=displayVIF(temp, tempNumericColumns, 'Numeric')
discreteVIF=displayVIF(temp, tempNumericDiscreteColumns, 'Numeric Discrete')
VIF=mergeDataFrames([numericVIF,discreteVIF])
i=1
print('----------------------Iteration ',i,'----------------------')
display(VIF.sort_values("VIF",ascending=False))

while VIF['VIF'].max()>10:
    i+=1
    feature=VIF[VIF['VIF']==VIF['VIF'].max()]['Feature']
    feature=feature.values[0]
    #print(feature)
    #break
    temp=dropColumns(temp,[feature])
    try:
        tempNumericColumns.remove(feature)
    except ValueError:
        pass  # do nothing!
    try:
        tempNumericDiscreteColumns.remove(feature)
    except ValueError:
        pass  # do nothing!

    numericVIF=displayVIF(temp, tempNumericColumns, 'Numeric')
    discreteVIF=displayVIF(temp, tempNumericDiscreteColumns, 'Numeric Discrete')
    VIF=mergeDataFrames([numericVIF,discreteVIF])
    print('----------------------Iteration ',i,'----------------------')
    display(VIF.sort_values("VIF",ascending=False))
```

**Approach towards VIF Analysis: As a rule of thumb, a VIF of more than 10 indicates highly dependent predictor variables. This results in poor predictions or poor results on test data set. We want to drop variables with the most VIF and re-run the VIF on the remaining set till all variables end up having VIF less than 10 Alternatively we can leverage PCA to identify the best predicting factors.**

**Conclusions from VIF Analysis:**

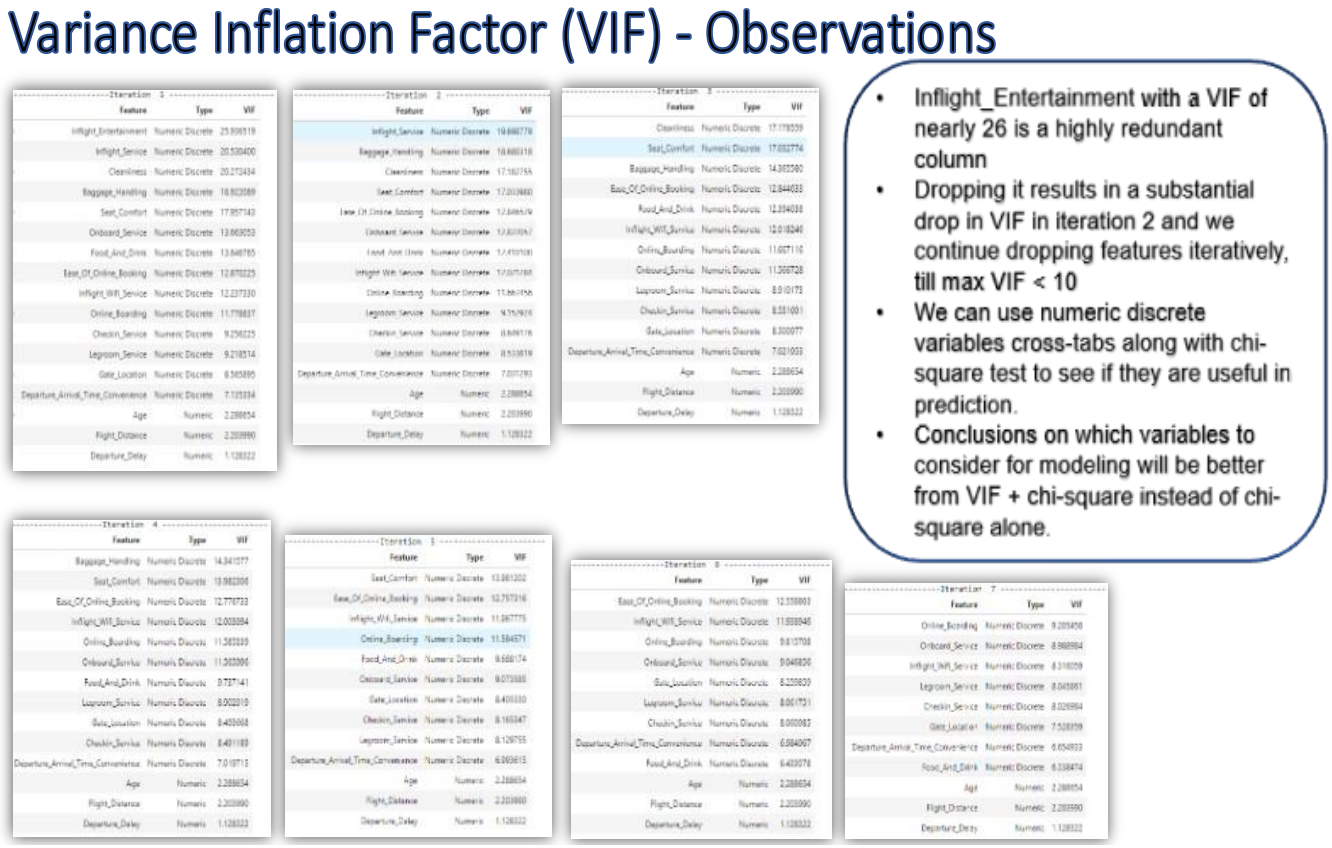
**Inflight_Entertainment with a VIF of nearly 26 is a highly redundant column**

**Dropping it results in a substantial drop in VIF in iteration 2**

**We carry on dropping till all remaining variables do not show multicollinearity**

**We can use numeric discrete variables cross-tabs along with chisquare testing to see if they are useful in prediction.**

**Conclusions on which variables to take into modeling will be more powerful from VIF + chisquare instead of chisquare alone.**




### 9.6.3 Chi Square Test - Categorical Features vs Target

| Feature | Chi-Squared Statistic | p-value | Degrees of Freedom | Impact | Retain | Drop |
|---|---|---|---|---|---|---|
| Online_Boarding | 39751.00 | 0.000 | 5 | Dependent on Target | Y | N |
| Inflight_Wifi_Service | 28696.41 | 0.000 | 5 | Dependent on Target | Y | N |
| Class | 26471.86 | 0.000 | 2 | Dependent on Target | Y | N |
| Type_Of_Travel | 20945.23 | 0.000 | 1 | Dependent on Target | Y | N |
| Inflight_Entertainment | 18508.07 | 0.000 | 5 | Dependent on Target | Y | N |
| Seat_Comfort | 15756.13 | 0.000 | 5 | Dependent on Target | Y | N |
| Legroom_Service | 12271.37 | 0.000 | 5 | Dependent on Target | Y | N |
| Onboard_Service | 11508.56 | 0.000 | 5 | Dependent on Target | Y | N |
| Ease_Of_Online_Booking | 10407.61 | 0.000 | 5 | Dependent on Target | Y | N |
| Cleanliness | 10245.16 | 0.000 | 5 | Dependent on Target | Y | N |
| Baggage_Handling | 8610.54 | 0.000 | 4 | Dependent on Target | Y | N |
| Inflight_Service | 8283.07 | 0.000 | 5 | Dependent on Target | Y | N |
| Checkin_Service | 6422.32 | 0.000 | 5 | Dependent on Target | Y | N |
| Food_And_Drink | 5203.24 | 0.000 | 5 | Dependent on Target | Y | N |
| Customer_Type | 3657.28 | 0.000 | 1 | Dependent on Target | Y | N |
| Gate_Location | 2493.44 | 0.000 | 5 | Dependent on Target | Y | N |
| Departure_Arrival_Time_Convenience | 451.06 | 0.000 | 5 | Dependent on Target | Y | N |
| Gender | 15.44 | 0.000 | 1 | Dependent on Target | Y | N |

**Conclusions from chisquare. All categorical predictor variables have significant chisquare statistic, hence all are relevant in the prediction of airline satisfaction. No categorical columns are dropped based on the chisquare test.** **Top categorical columns relevant in predicting satisfaction are Satisfaction Online_Boarding, Inflight_Wifi_Service, Class, and Type_Of_Travel.

### 9.7 Variable Encoding

```
# Encode usign label encoder
columnsToEncode=['Gender','Customer_Type','Type_Of_Travel','Class', targetColumn]
airlineData=performLabelEncoding(airlineData, columnsToEncode)
```

| | Gender | Gender_Encoded |
|---|---|---|
| **0** | Male | 1 |
| **1** | Female | 0 |

| | Customer_Type | Customer_Type_Encoded |
|---|---|---|
| **0** | Loyal Customer | 0 |
| **1** | disloyal Customer | 1 |

| | Type_Of_Travel | Type_Of_Travel_Encoded |
|---|---|---|
| **0** | Personal Travel | 1 |
| **1** | Business travel | 0 |

| | Class | Class_Encoded |
|---|---|---|
| **0** | Eco Plus | 2 |
| **1** | Business | 0 |
| **2** | Eco | 1 |

| | Satisfaction | Satisfaction_Encoded |
|---|---|---|
| **0** | neutral or dissatisfied | 0 |
| **1** | satisfied | 1 |

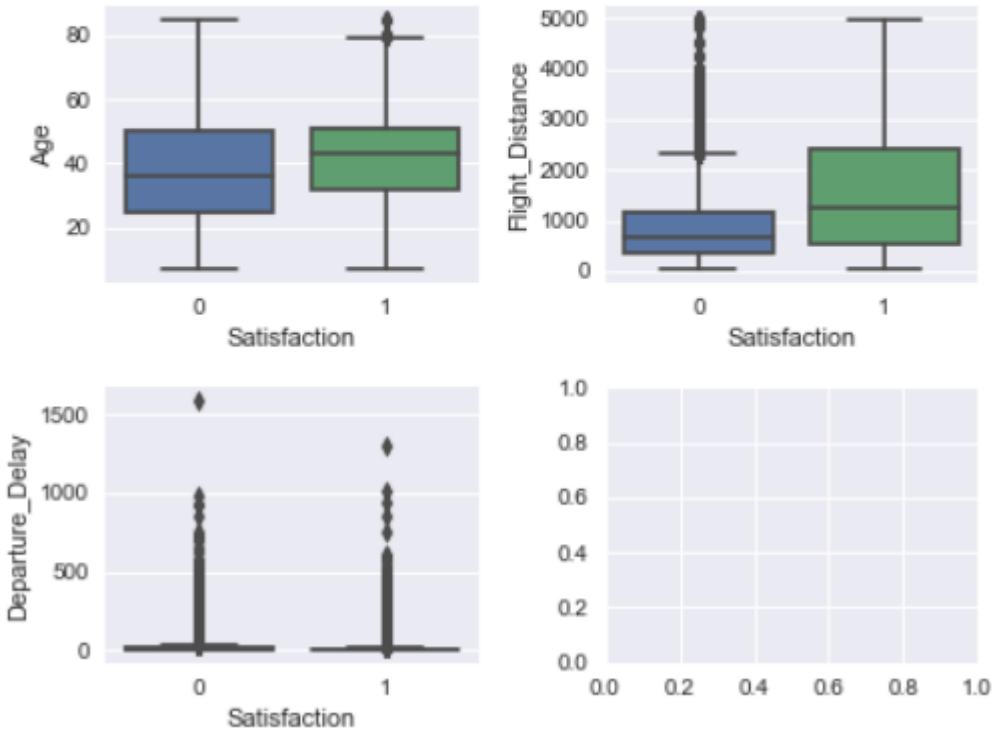
### 9.8 Outlier

**Four different techniques were used for outlier detection- box plots, Zscore and IQR which are univariate approaches and DBScan and KMeans clustering wihich are multivariate approaches.**

**Code, output and conclusion of each of these methods follow:**

## Box Plot Outlier Detection

```
temp=airlineData_NumericColumns.copy()
temp.append(targetColumn)
boxPlot(airlineData[temp], targetColumn)
```




*Based on boxplots, univariate outliers are seen in Age, Flight_Distance and Departure_Delay*

### zScore Outlier Detection

```
outlierCountDF = pd.DataFrame(columns=['Model', 'TargetValue', 'OutlierCount'])
outlierZscoreData, zScorelevel0Outlier, zScorelevel1Outlier = zScoreOutlier(airlineData.loc[:, airlineData.columns != 'FileType']
outlierCountDF = outlierCountDF.append({'Model':'zScore', 'TargetValue':0, 'OutlierCount':zScorelevel0Outlier}, ignore_index=True
outlierCountDF = outlierCountDF.append({'Model':'zScore', 'TargetValue':1, 'OutlierCount':zScorelevel1Outlier}, ignore_index=True
display(outlierCountDF)
```

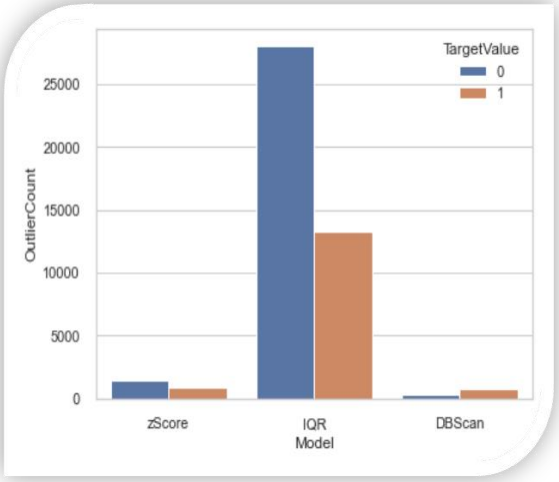
| | Model | TargetValue | OutlierCount |
|---|---|---|---|
| 0 | zScore | 0 | 1457 |
| 1 | zScore | 1 | 843 |

## DB Scan Outlier Detection

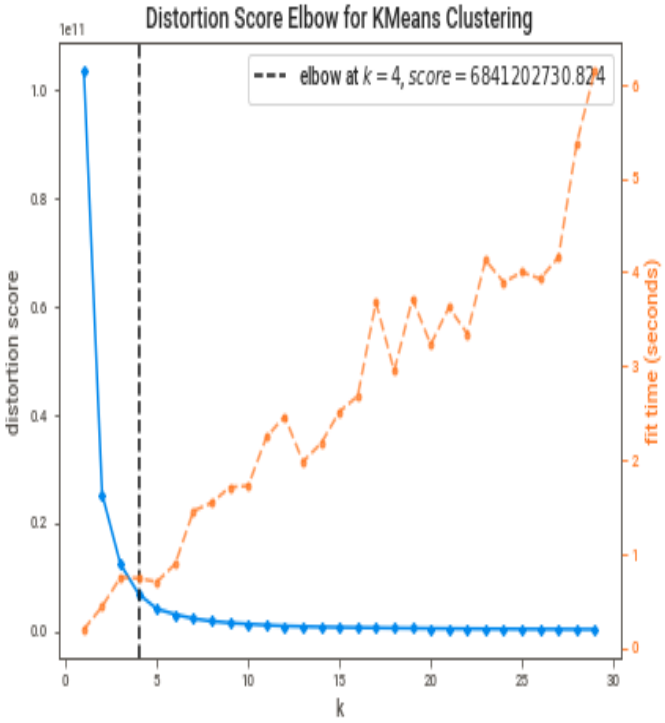
```
data=airlineData.loc[:, airlineData.columns != 'FileType']
outliersRowDBS, DBSlevel0Outlier, DBSlevel1Outlier = DBScanOutlier(data, 'Departure_Delay', 'Flight_Distance', targetColumn, data.shape[1]+1)
outlierCountDF = outlierCountDF.append({'Model':'DBScan', 'TargetValue':0, 'OutlierCount':DBSlevel0Outlier}, ignore_index=True)
outlierCountDF = outlierCountDF.append({'Model':'DBScan', 'TargetValue':1, 'OutlierCount':DBSlevel1Outlier}, ignore_index=True)
outlierCountDF
```

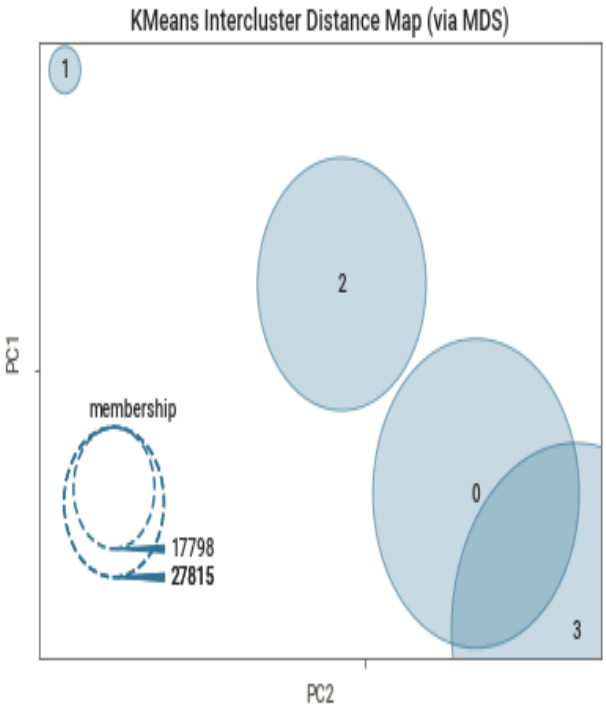
- DBScan method yields a multivariate outlier count of about 1.1K
- IQR based approach yields about 41K outliers since IQR is at variable level and not multidimensional approach - this results in dropping a large proportion of the data set and is not recommended
- Zscore based approach gives around 3K outliers and is a univariate approach

| Model | TargetValue | OutlierCount |
|---|---|---|
| zScore | 0 | 1457 |
| zScore | 1 | 843 |
| IQR | 0 | 28014 |
| IQR | 1 | 13236 |
| DBScan | 0 | 303 |
| DBScan | 1 | 800 |



**Based on DB Scan, Cluster 1 seems to be the farthest from other clusters and indicates the presence of outliers**



### 9.9 Scaling of Data

Prior to PCA, data scaling is done using min-max scaler

```
airlineDataFinal_Scaled=scaleData(airlineDataFinal)
airlineDataFinal_Scaled.head()
```

| | Satisfaction | Inflight_Wifi_Service | Departure_Arrival_Time_Convenience | Ease_Of_Online_Booking | Gate_Location | Food_And_Drink | Online_Boarding |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 0.6 | 0.8 | 0.6 | 0.2 | 1.0 | 0.6 |
| **1** | 0 | 0.6 | 0.4 | 0.6 | 0.6 | 0.2 | 0.6 |
| **2** | 1 | 0.4 | 0.4 | 0.4 | 0.4 | 1.0 | 1.0 |
| **3** | 0 | 0.4 | 1.0 | 1.0 | 1.0 | 0.4 | 0.4 |
| **4** | 1 | 0.6 | 0.6 | 0.6 | 0.6 | 0.8 | 1.0 |

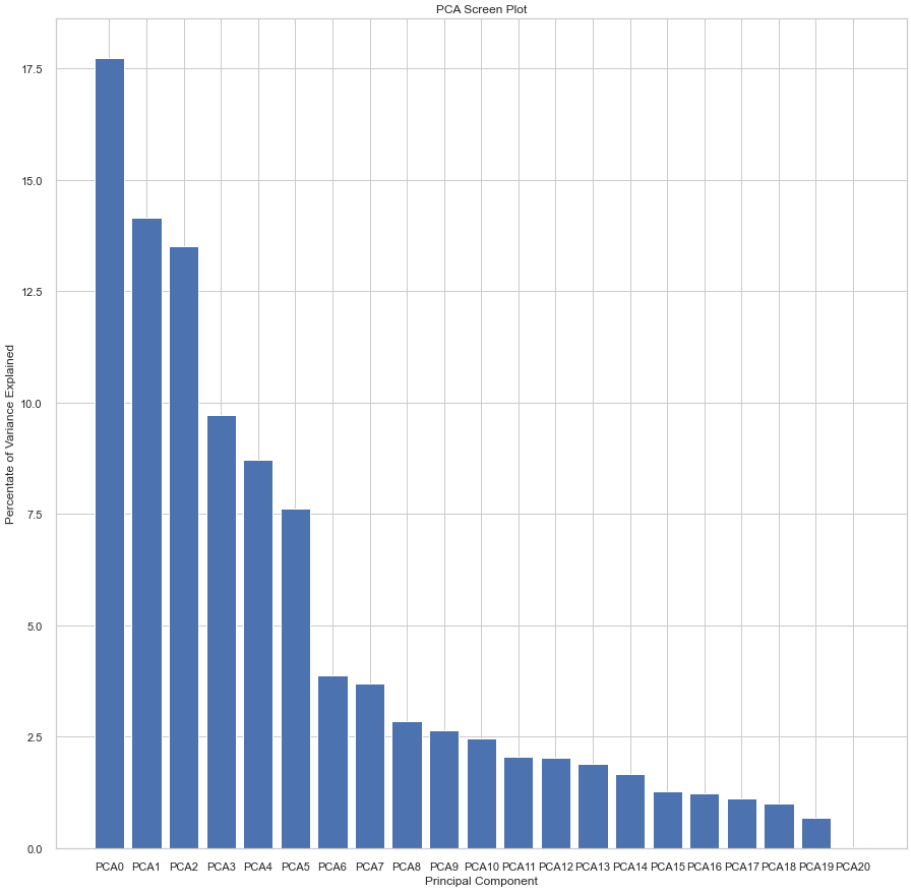
5 rows × 22 columns

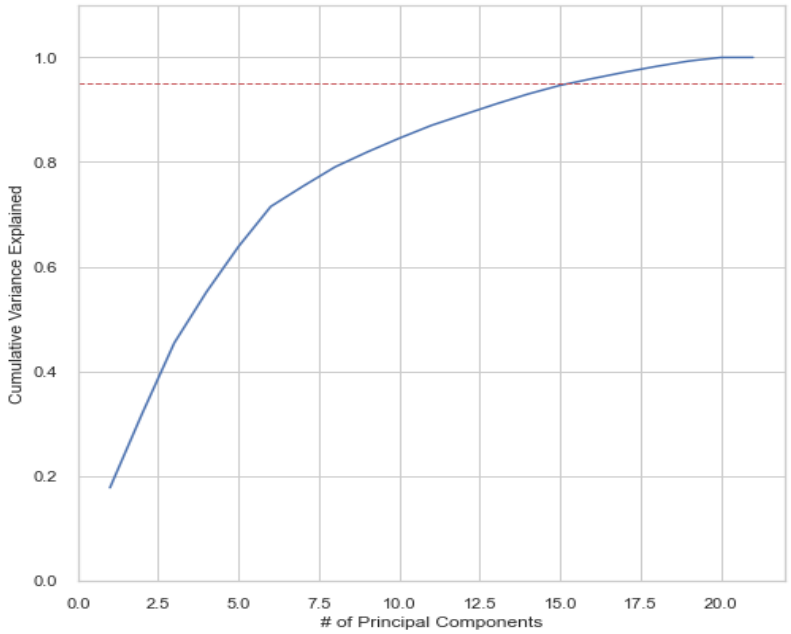
### 9.10 PCA

## We find that 15 principal Components are required to explain 95% of variance in data.

```
pcaObj = pca()
pcaData = pcaObj.createPCA(airlineDataFinal_Scaled, targetColumn)

pcaData = pd.concat([pcaData.iloc[:, 0:15], pcaData[targetColumn]], axis=1)
```

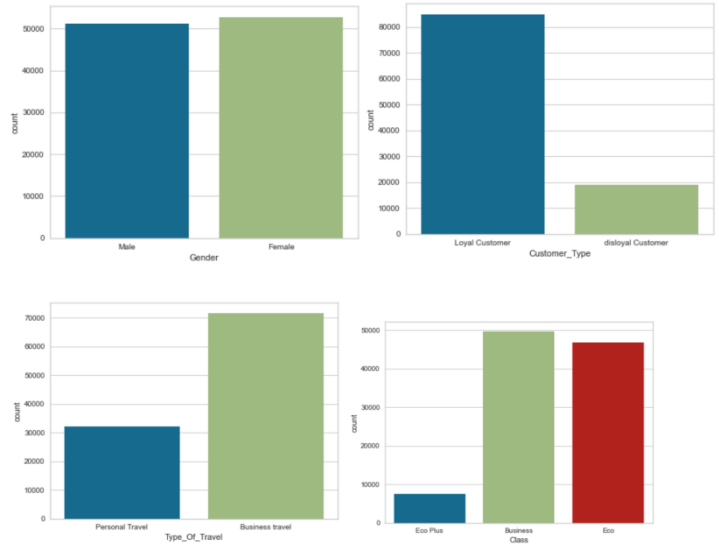
```
       PCA0      PCA1      PCA2      PCA3      PCA4      PCA5      PCA6  \
0  0.311914 -1.001735 -0.156314 -0.416334 -0.204536  0.688131 -0.199073
1  0.244700  0.604401  1.140787  0.503926 -0.290849 -0.041171 -0.182339
2 -0.689349  0.214959 -0.487480 -0.539570  0.210242  0.055010 -0.025009
3  0.115425  0.495264 -0.197199  0.748074  0.002030 -0.400625  0.459316
4 -0.457754 -0.332117  0.352987  0.061300  0.366152 -0.137917 -0.182226
5  1.003513  0.170917 -0.405782  0.076431 -0.493697 -0.535816 -0.222390
6  0.823062 -0.571864  0.255701  0.001800 -0.275076 -0.290117  0.080131
7 -1.012095 -0.059425 -0.670259  0.043906 -0.136296 -0.136109 -0.041773
8  0.368239  0.844156 -0.008085 -0.100528  0.870136 -0.228658  0.030260
9  0.105020  0.313895  0.874290  0.427819 -0.316785  0.403846  0.015159

       PCA7      PCA8      PCA9  ...     PCA11     PCA12     PCA13     PCA14  \
0  0.075551  0.105424  0.020204  ...  0.273883 -0.114055  0.040879  0.010130
1  0.089039  0.346797 -0.281227  ... -0.129405 -0.224064 -0.241496  0.096725
2 -0.112277  0.146055  0.133700  ... -0.076695 -0.176228  0.038590  0.062439
3  0.403514  0.147082 -0.287837  ... -0.232997 -0.397475 -0.086343  0.065681
4 -0.007538 -0.264565  0.040058  ... -0.187932 -0.199119 -0.225989  0.047804
5 -0.155922  0.178351 -0.180654  ...  0.203746 -0.122624 -0.128230 -0.146241
```


PCA Screen Plot
Percentate of Variance Explained
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
PCA0 PCA1 PCA2 PCA3 PCA4 PCA5 PCA6 PCA7 PCA8 PCA9 PCA10 PCA11 PCA12 PCA13 PCA14 PCA15 PCA16 PCA17 PCA18 PCA19 PCA20
Principal Component

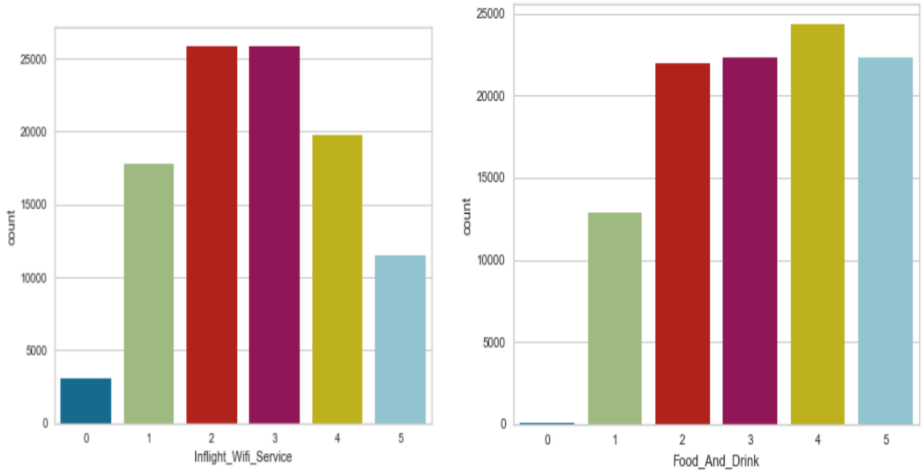

Cumulative Variance Explained
1.0
0.8
0.6
0.4
0.2
0.0
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
# of Principal Components

## 10 Visualization for Summarization

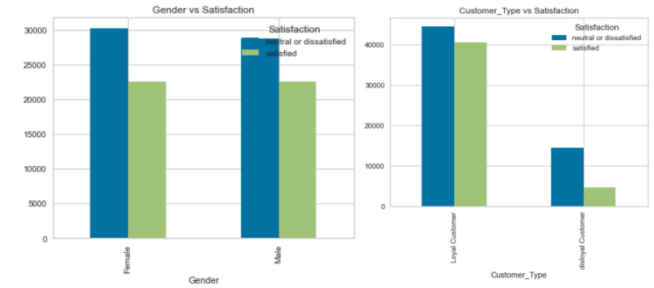
The visualization of inter-relationships among predictor variables, their relationship with dependent variables, variable density distribution, visualization of PCA are done usingvarious plots.

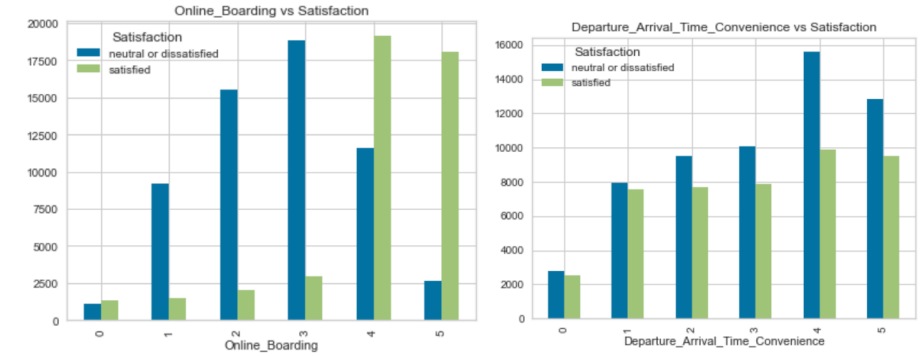



**Technique:** Frequency Charts
**Stage Used**: Applied on raw train data
**Purpose**: To understand the how values of categorical variables behave
**Data**: Train data
**Inference**: 1) several inter-relationships are clearly visible
**Action Taken**: Used observations made to guide modeling




**Inferences:**
Survey ratings for all services have been rated average or above by >75% of passengers.
0 ratings are in single digit % for all services, mostly <1%
Inflight Service, Baggage Handling, Legroom Service, Onboard Service, Inflight Entertainment, Seat Comfort, Online Boarding services have high ratings (4,5) from more than 50% of the passenger




**Technique:** Cross Tabs against Target

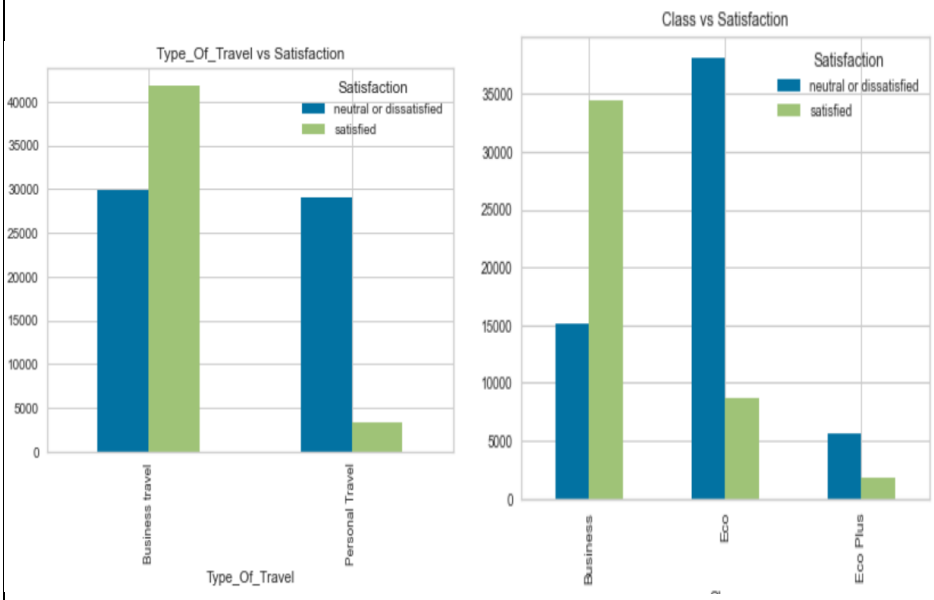
**Stage Used**: Applied on raw train data

**Purpose**: To understand which categorical variable drive satisfaction

**Data**: Train Data

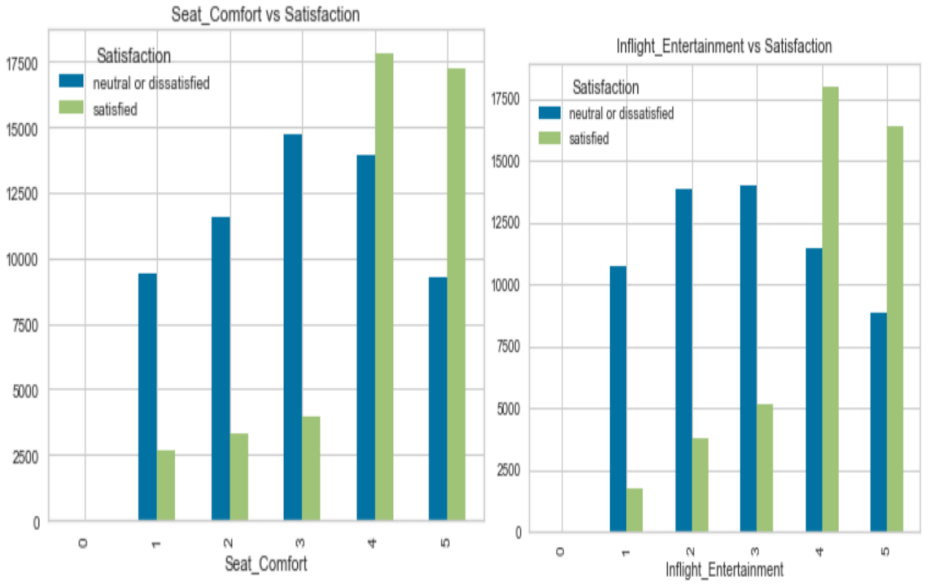
**Inference**: Several Interrelations are called out

**Action Taken**:Used observations made to guide modeling



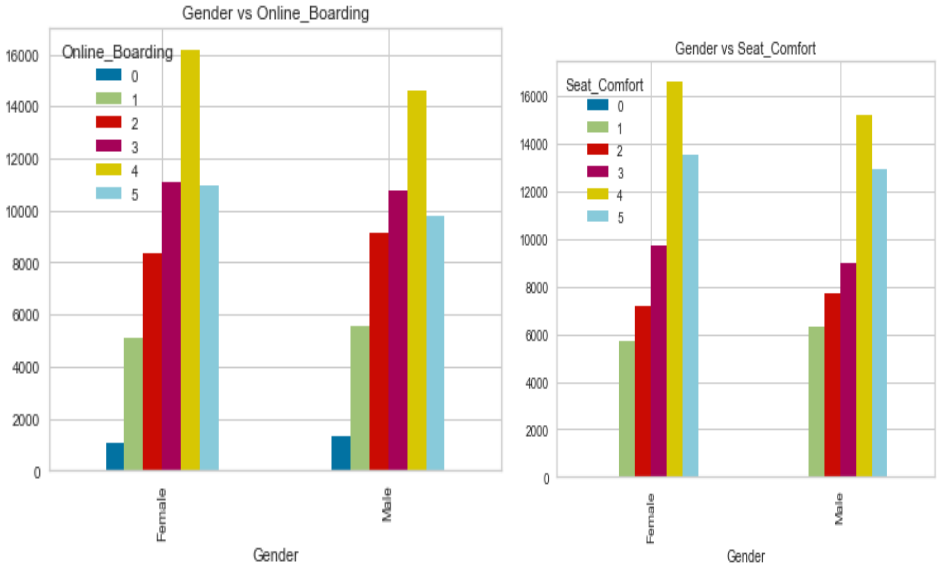

**Inferences:**

- Personal travel customers have greater proportion of dis-satisfied fliers compared to business travelers
- Business class has largest proportion of satisfied fliers while eco class has it the other way around



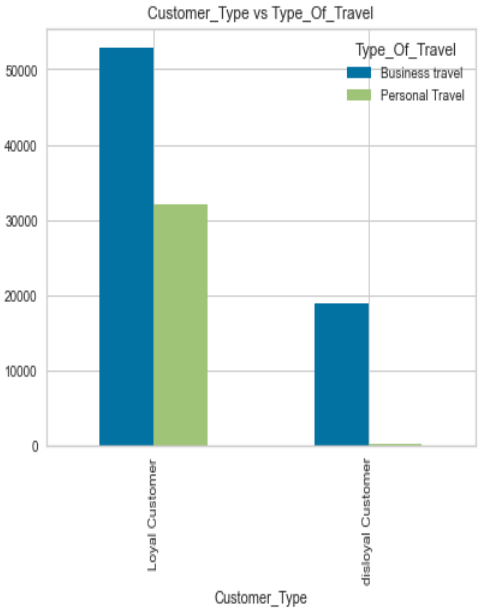

**Inferences:**

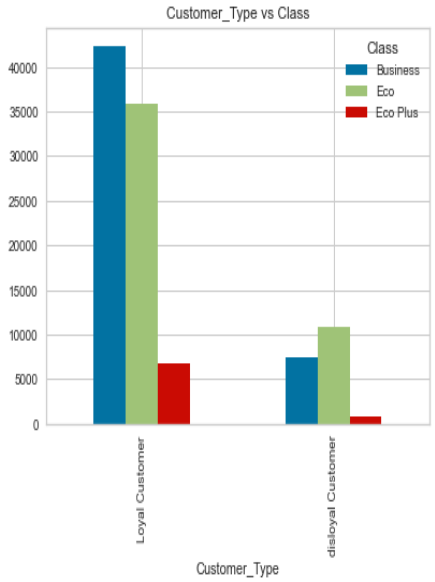
- Seat comfort is also an important indicator of overall satisfaction
- Inflight entertainment satisfaction is important for overall satisfaction




**Inferences:**

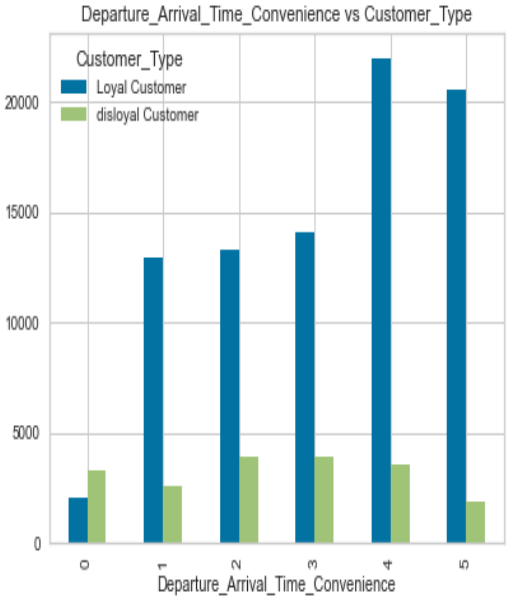
- Female fliers rated online booking good as compared to male fliers
- Female fliers rated seat comfort good as compared to male fliers

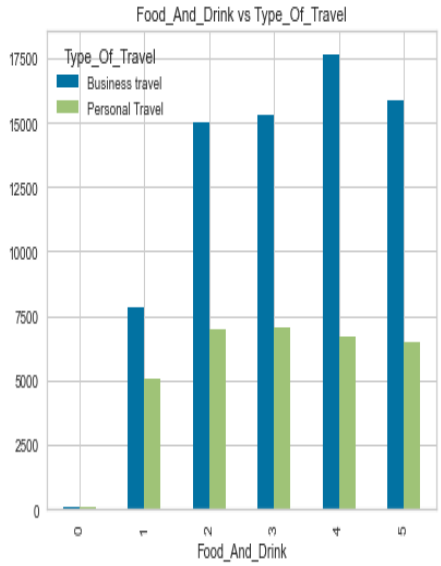



**Inferences:**

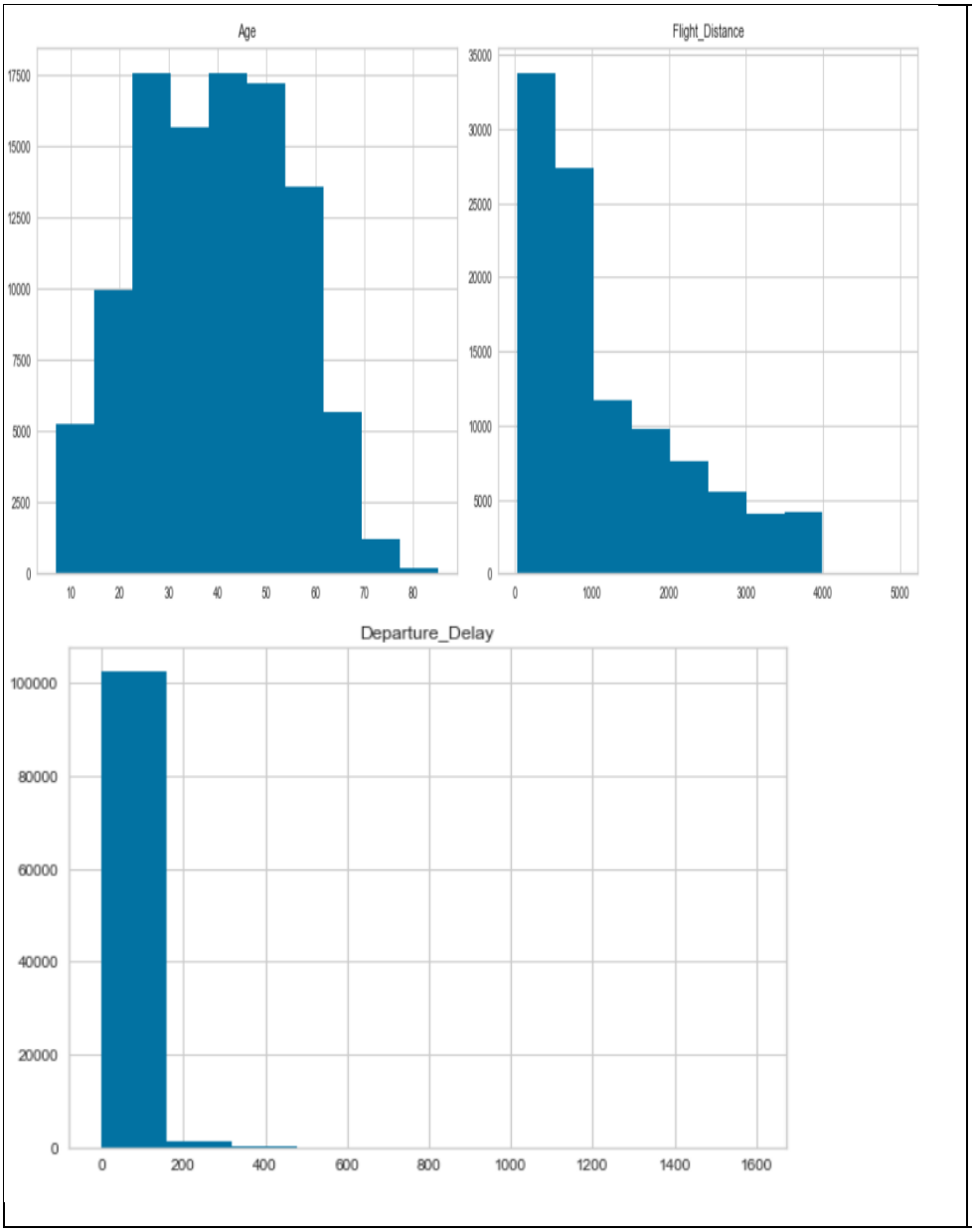
- Business travelers are mostly loyal customers.
- Most of the business class passengers are loyal customers



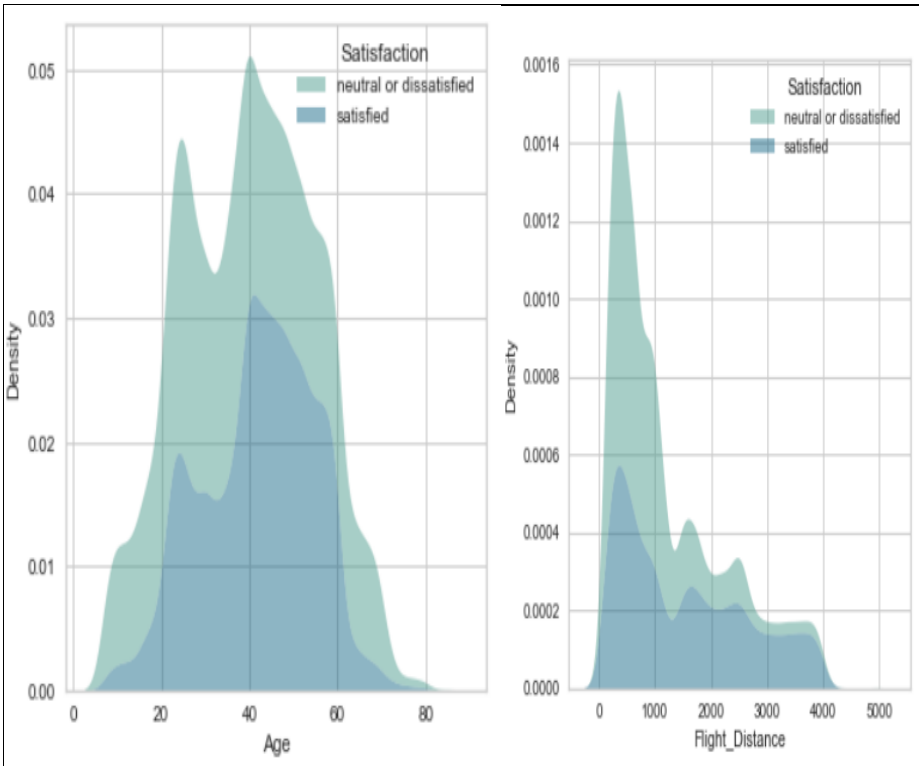

**Inferences:**

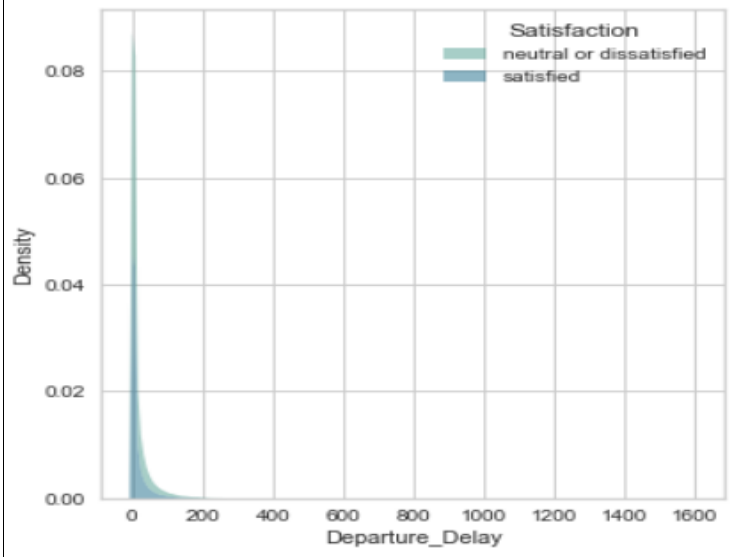
- Loyal customers are happy with departure time convenience
- Business Travelers are happy with food and drink services.




**Inferences:**

- **None of the numeric variables are normally distributed**
- **Age is bimodal with a large proportion of fliers in the range 22 to 50.**
- **A large proportion of the flights are less than 1500 kms in the data set**
- **Delayed departures are a very small proportion of our data set.**




**Technique:** Cross Tabs against Target

**Stage Used**: Applied on raw train data

**Purpose**: To understand which categorical variable drive satisfaction

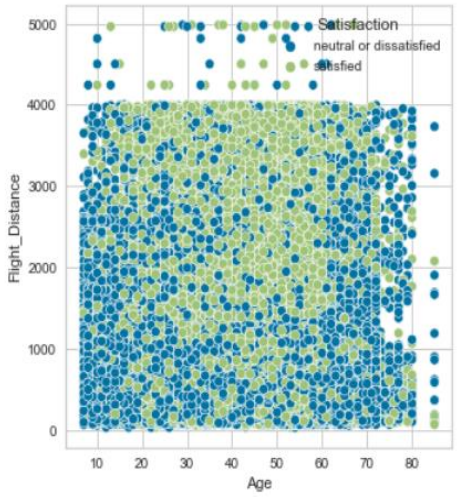
**Data**: Train Data

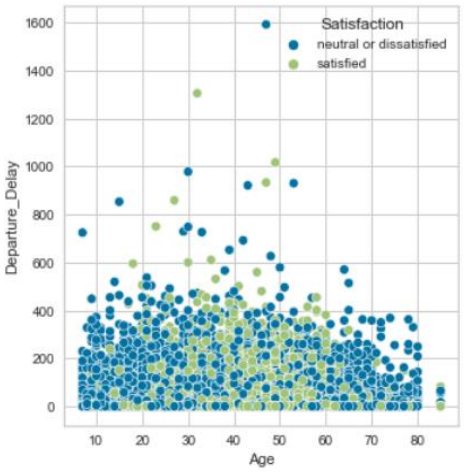
**Inference**: Several Interrelations are called out

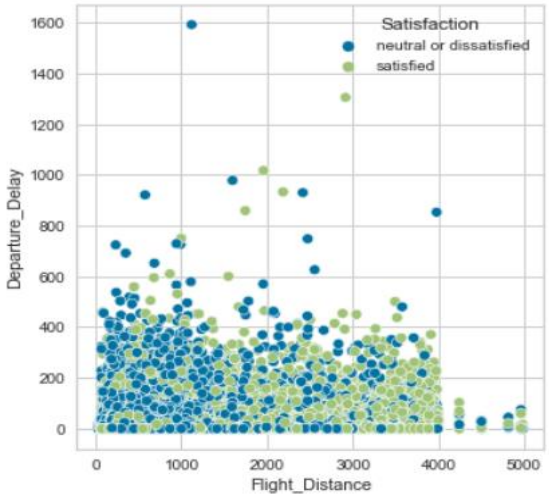
**Action Taken**:Used observations made to guide modeling

**Inferences:**

- Even when we look at satisfied and dissatisfied as different distributions, none of the numeric variables are normally distributed in our data set
- The distribution of the three continuous variables - age, flight distance and delayed departures within the binary class of the target variable are similar



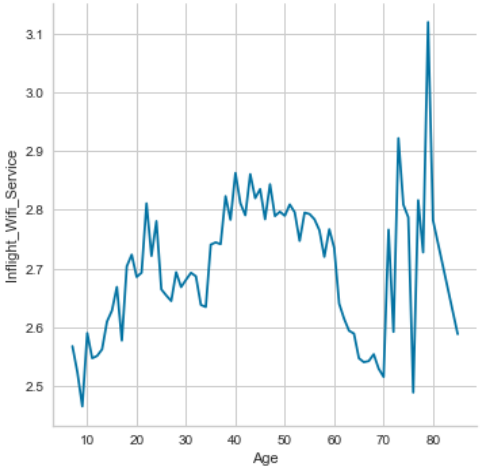



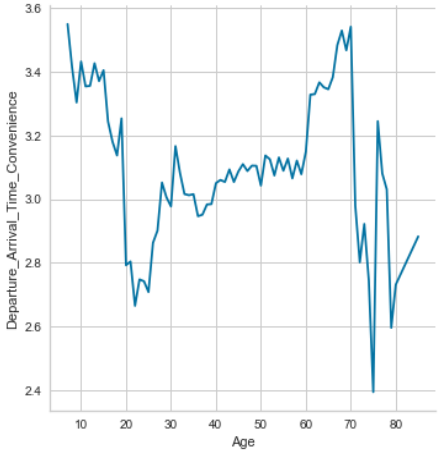




**Numeric Columns – Scatter Plots**

**Inferences:**

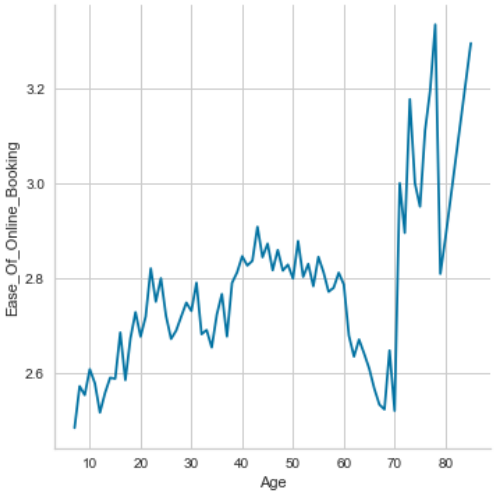
- Fliers in the age range of 30 to 50 flying in the range of 2000 to 4000 kms rate overall satisfaction positively compared to others.
- Passengers aged under 20 and over 60, facing departure delays, rate satisfaction neutral or poor generally
- For long distance flights, departure delays do not seem to impact satisfaction as much

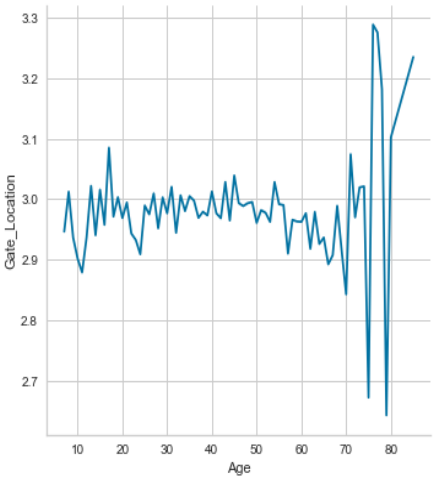



**Inferences:**

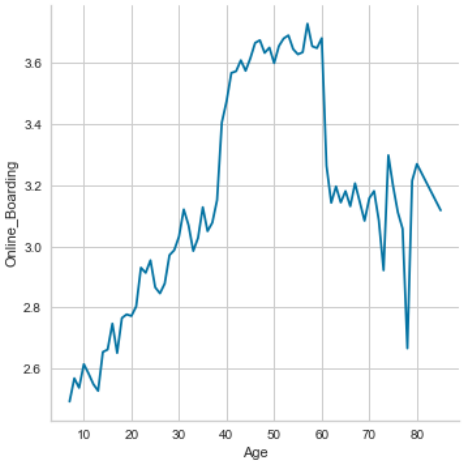
- Rating of inflight services with respect to age is jagged -it improves as age increases up to 23, then shows dip in the range of 25 to 35
- Departure-Arrival timing is rated low by those in mid-20s and lowest by those in mid-70s.

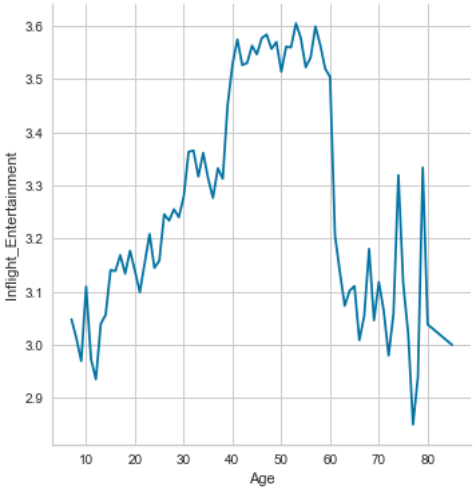



**Inferences:**

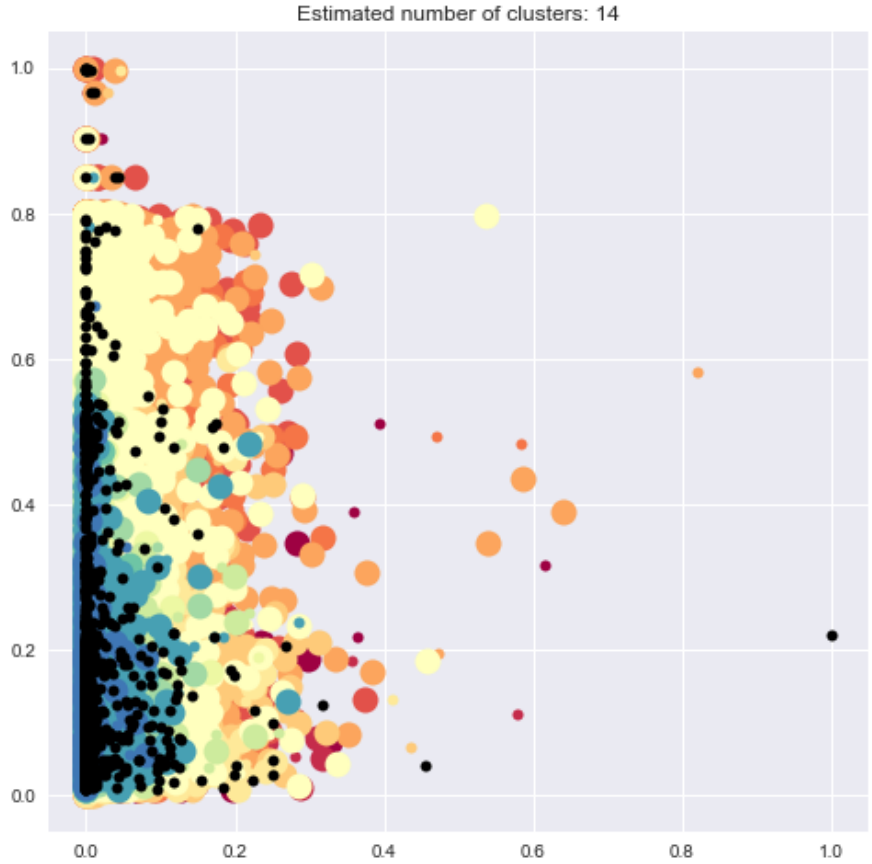
- Ease of online booking is rated higher by fliers over 70 years of age
- Gate location rating is average for most of the fliers, except it tends to vary widely in the 70+ age group



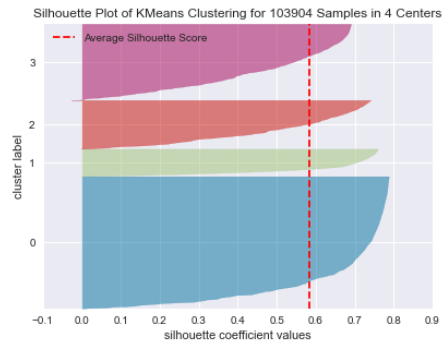



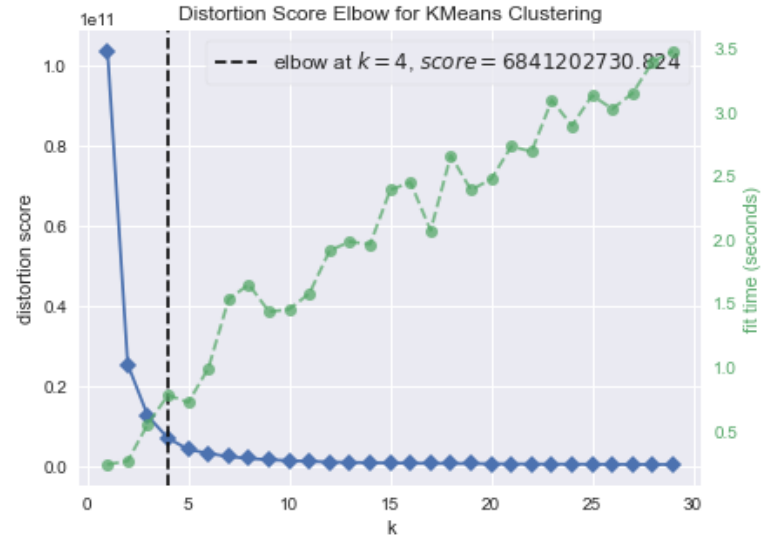

**Inferences:**

- Online boarding is rated higher by fliers aged 40-60 years
- Seat comfort is also rated higher by fliers aged 40 to 60 years as is in-flight entertainment

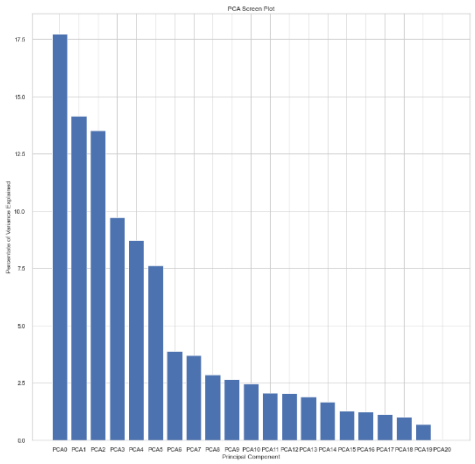


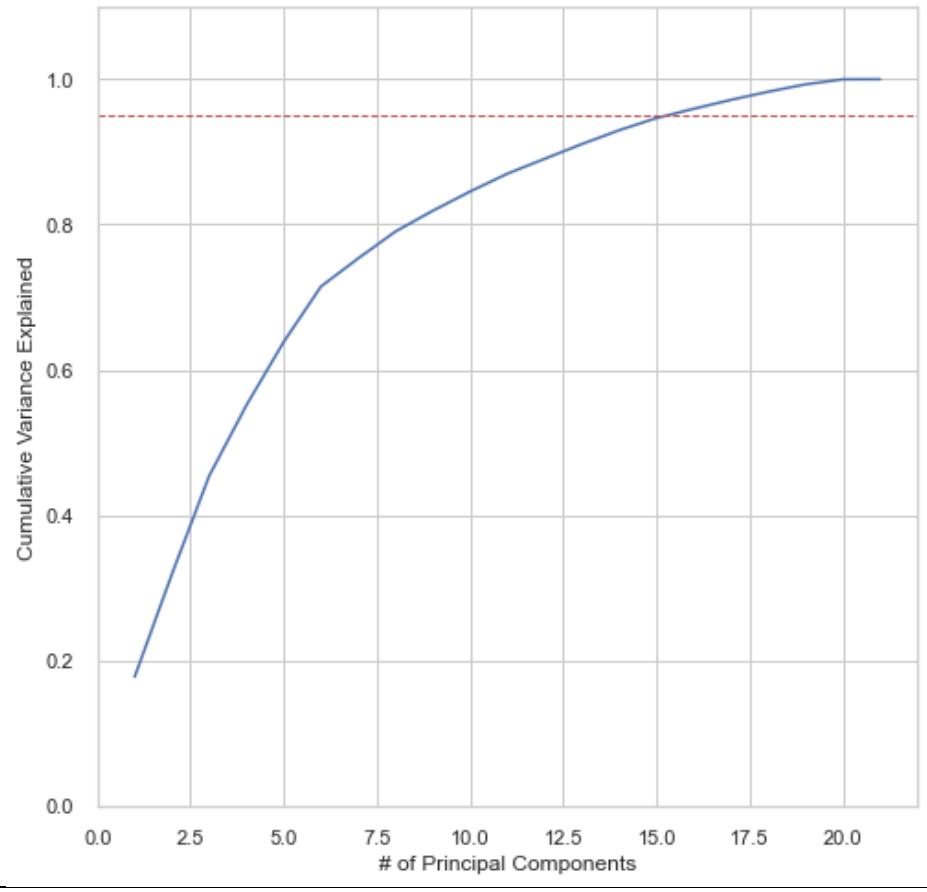

**Technique:** DBSCAN
**Stage Used**: After Normalising the data, applied on the entire dataset.
**Purpose**: To understand outliers in the data
**Data**: Train data
**Inference**: outlier cluster clearly flagged
**Action Taken**: Outlier removed




**Technique:** KMeans
**Stage Used**: After Normalising the data, applied on the train.
**Purpose**: To understand outliers in the data
**Data**: all the data variables
**Inference**: many points are shown as outliers,
**Action Taken**: Outlier removed






**Technique:** PCA

**Stage Used**: PCA was done towards the end of pre-processing step.

**Purpose**: To understand how many principal components give maximum variance

**Data**: all the data variables in train

**Inference**: 95% variance can be preserved with 14 principal components

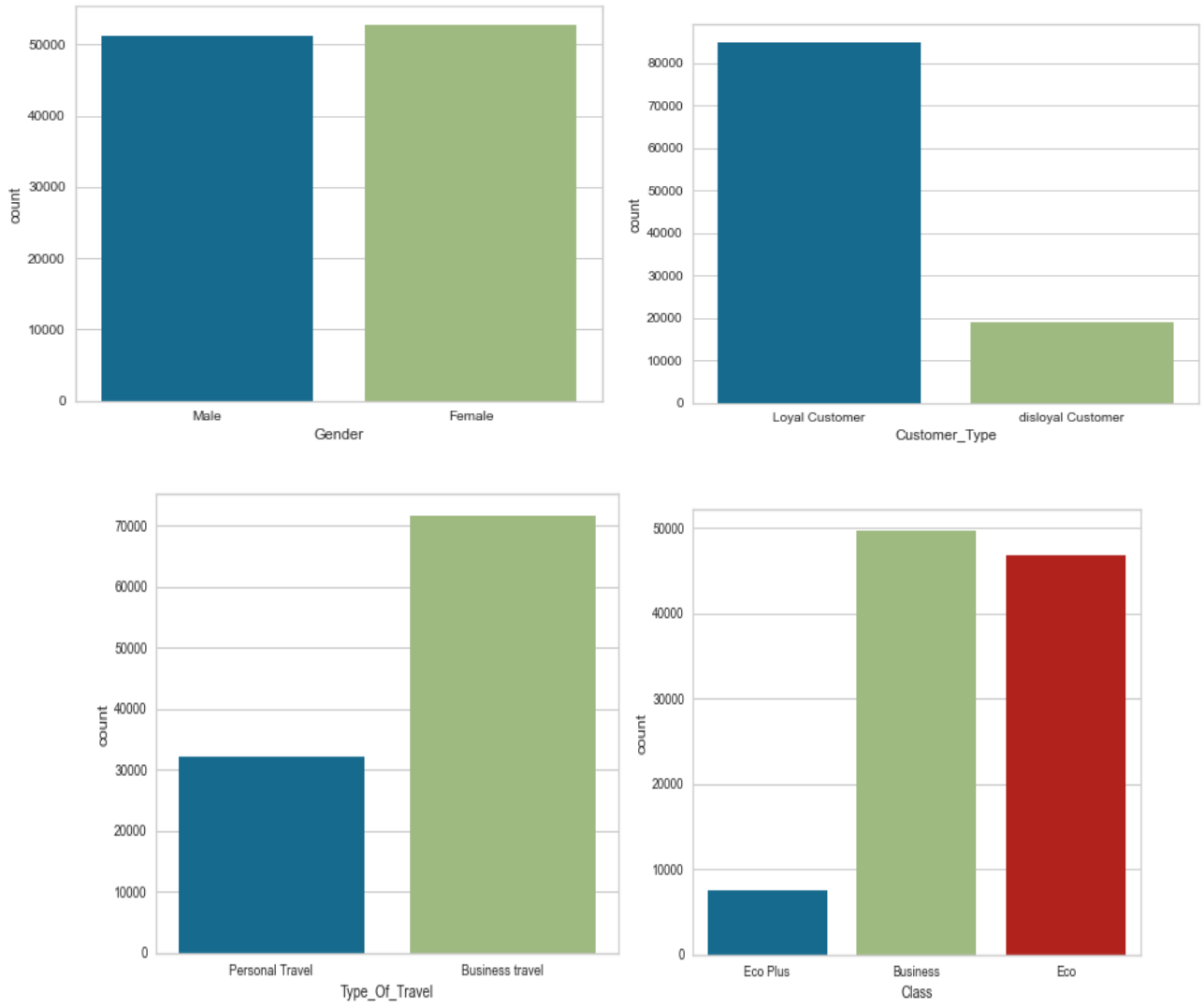
**Action Taken**: Since we don't have too many number of variables, we don't intend to use PCA to reduce the dimensions

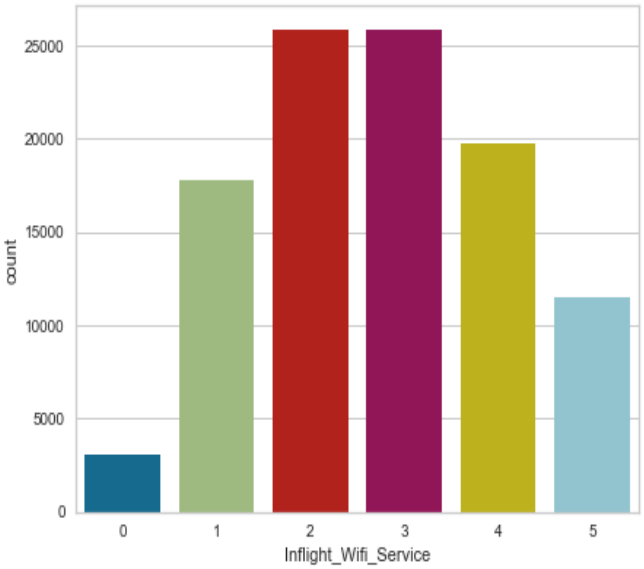
## 10.1 Visualizations from Code

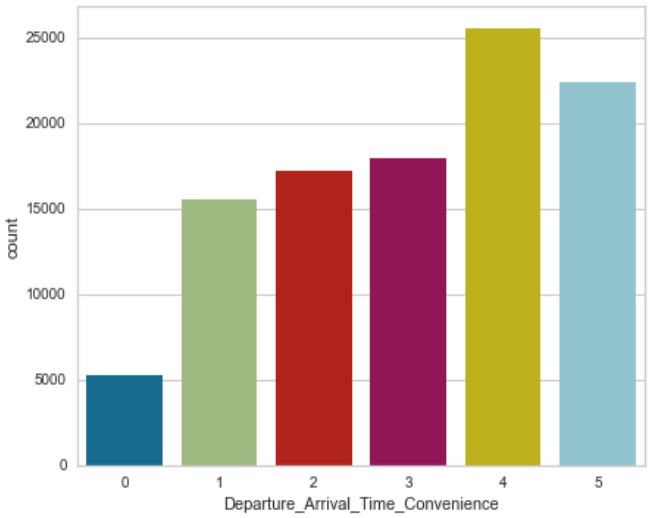
# Count Plots

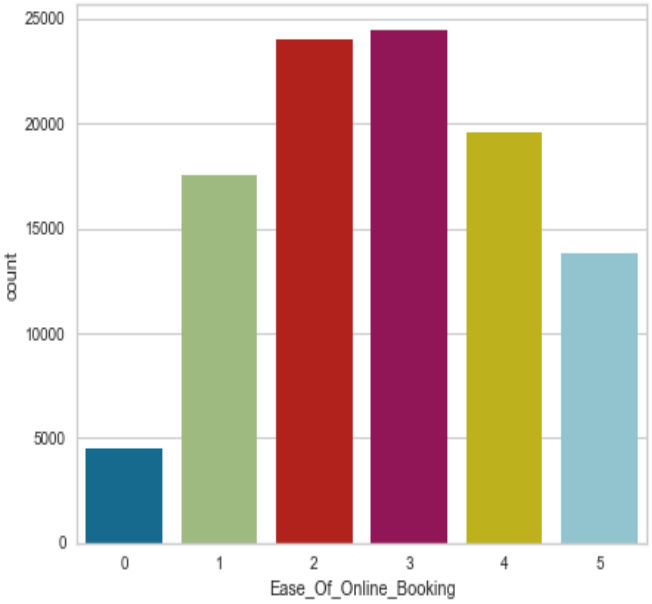
Show the counts of observations in each categorical bin using **bars**.

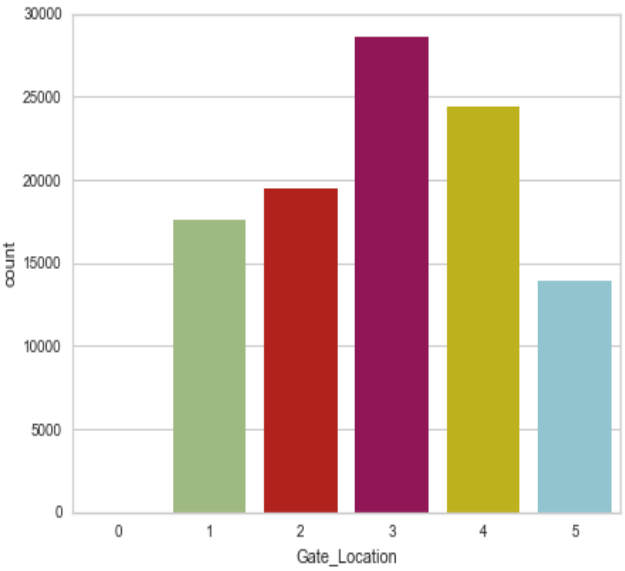
```
]: displayCountPlots(airlineData,airlineData_CategoricalColumns)
```

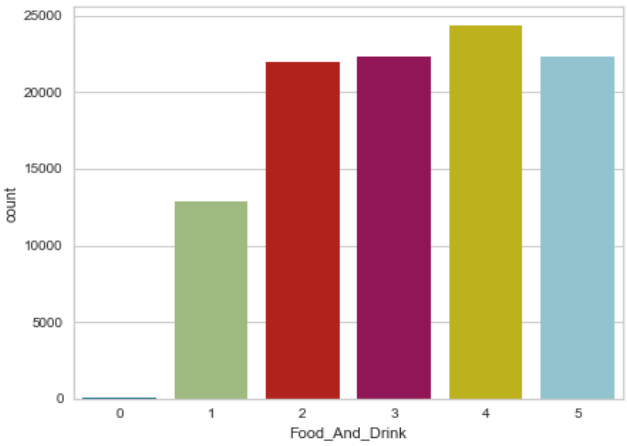


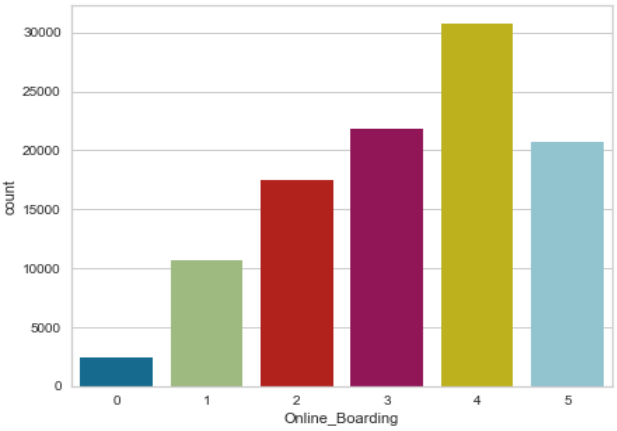

count
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_Wifi_Service

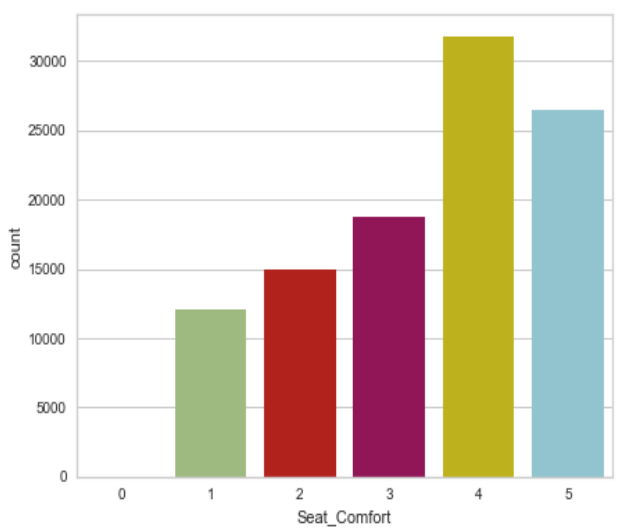

count
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Departure_Arrival_Time_Convenience

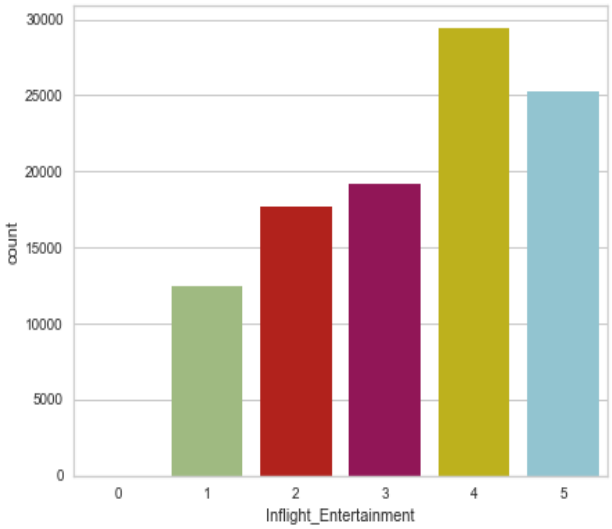

count
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Ease_Of_Online_Booking

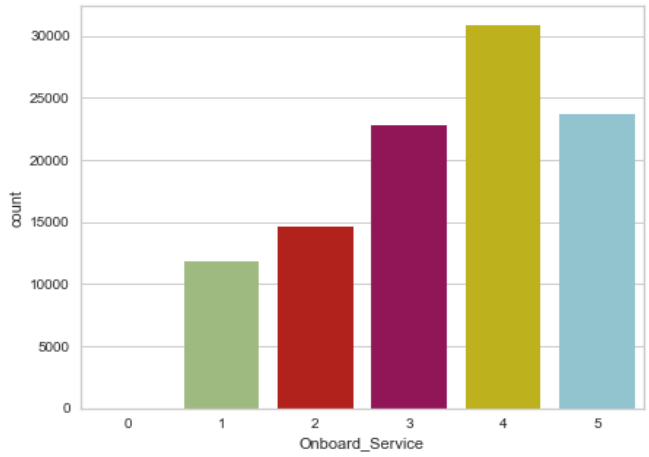

count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Gate_Location

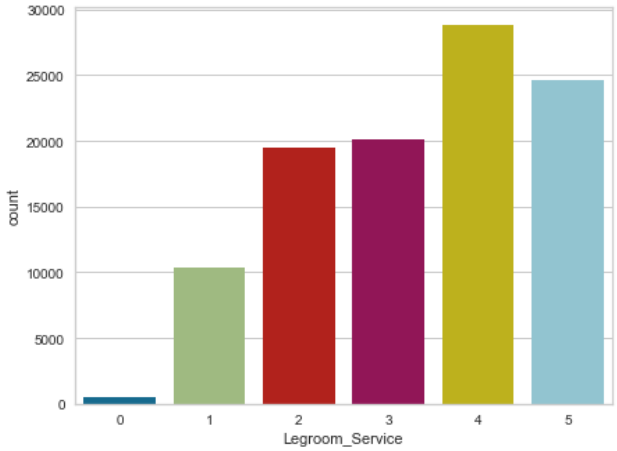

count
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Food_And_Drink

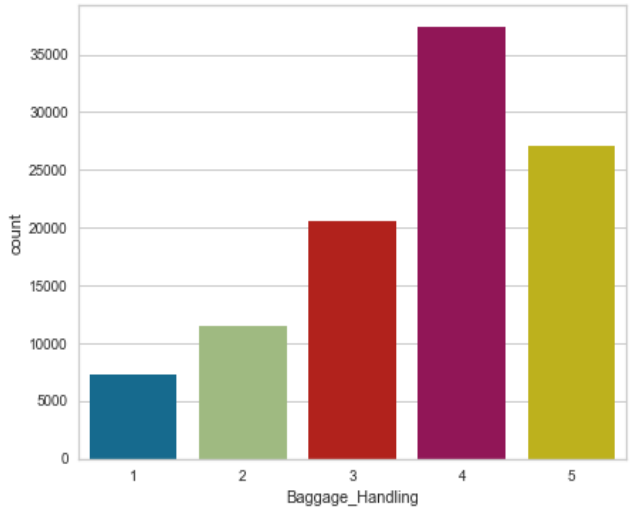

count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Online_Boarding

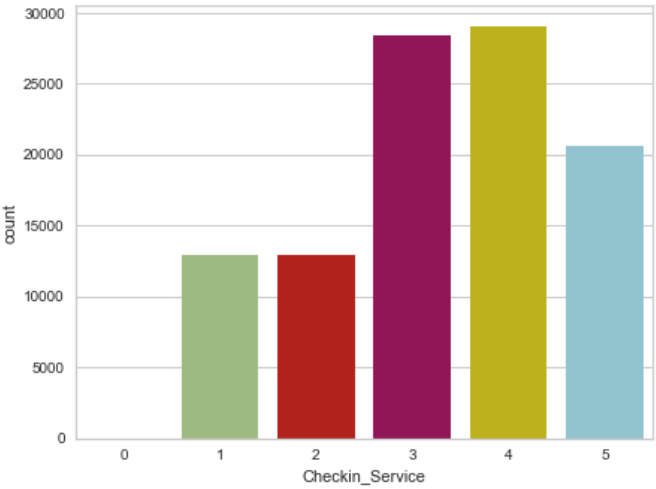

count
Seat_Comfort

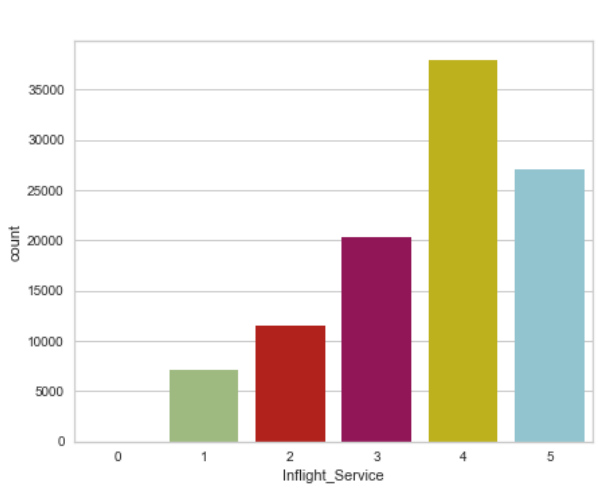

count
Inflight_Entertainment

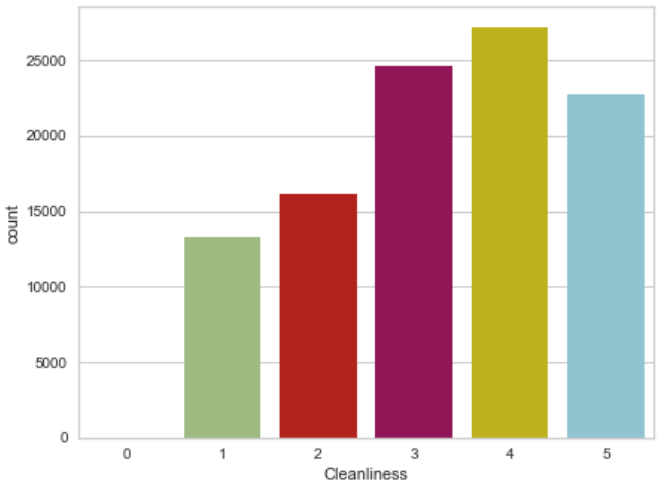

count
Onboard_Service


count
Legroom_Service


count
Baggage_Handling


count
Checkin_Service



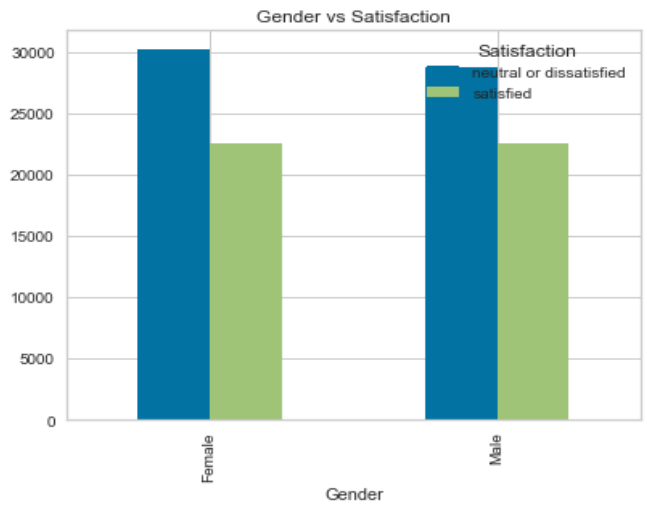



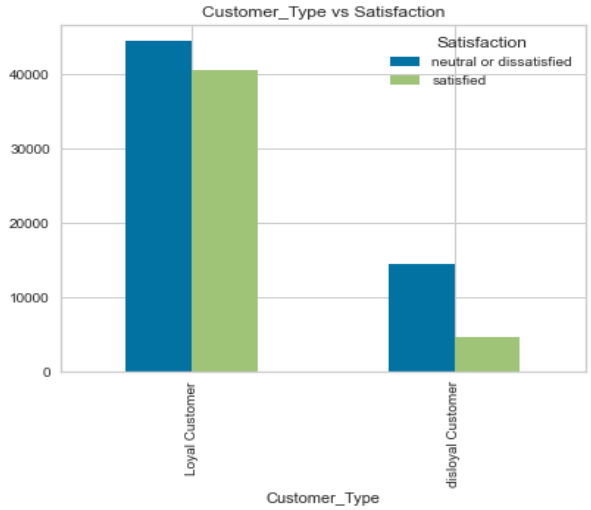

### 10.1.1 Univariate Analysis on categorical columns gives us following insights

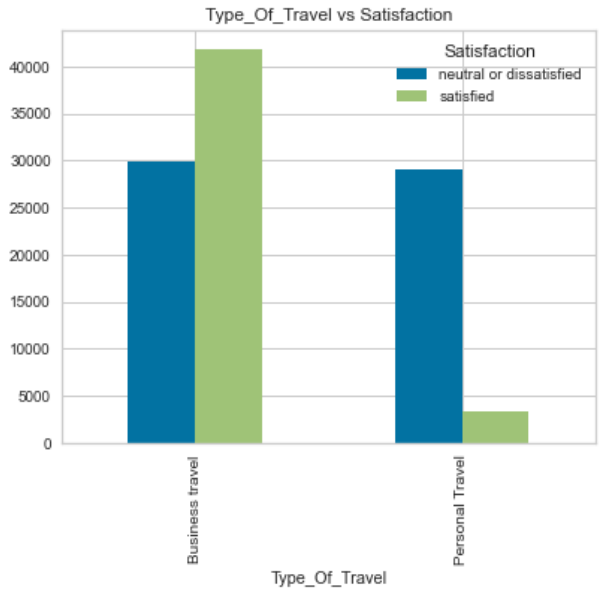
1.Proportion of female passengers is nearly the same as male passengers in the train data.
2.Data set has substantially higher representation of loyal customers (82%) versus disloyal customer (18%).
3.Busines travelers are more than personal travelers (69% versus 71%).
4.Business class passengers and eco class are nearly balanced in the data while eco plus class is a very small 7% .
5.Largest proportion of passengers rated the inflight wifi service as average.
6.Departure-Arrival time convivence is rating is good.
7.Ease of Online booking rated as average.
8.Gate location rated at average.
9.Food and drink services are rated as average to good.
10.Online Boarding is rated as good.
11.Seat comfort is rated good.
12.All other services are rated from average to good.

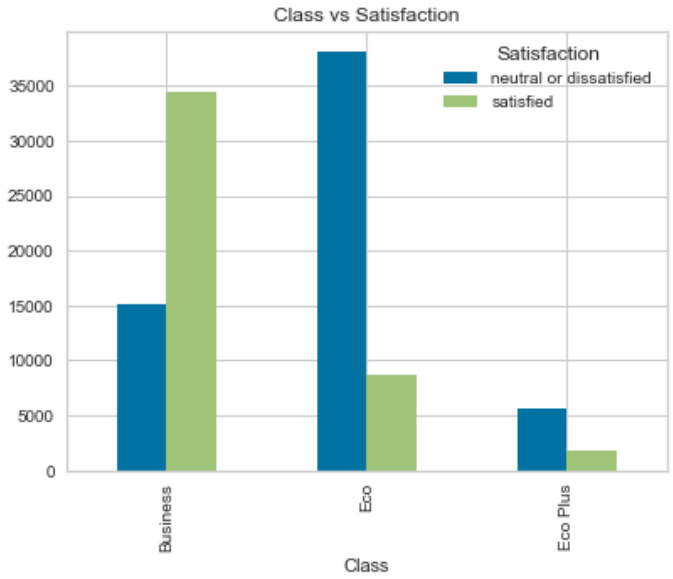
### 10.1.2 Cross Tabs - Categorical Features vs Target

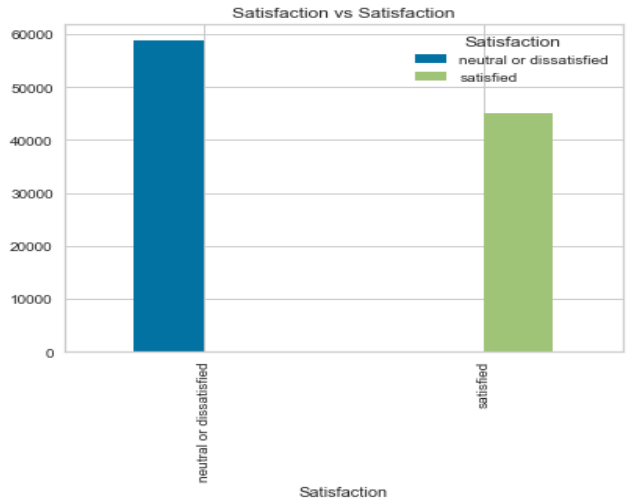
displayCrossTabs(airlineData, airlineData_CategoricalColumns, targetColumn)

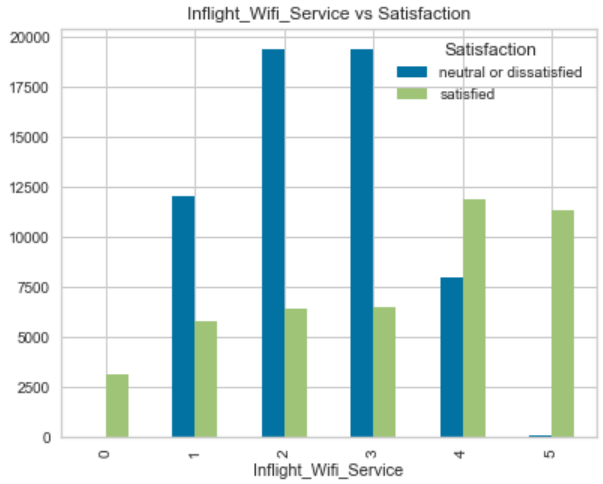


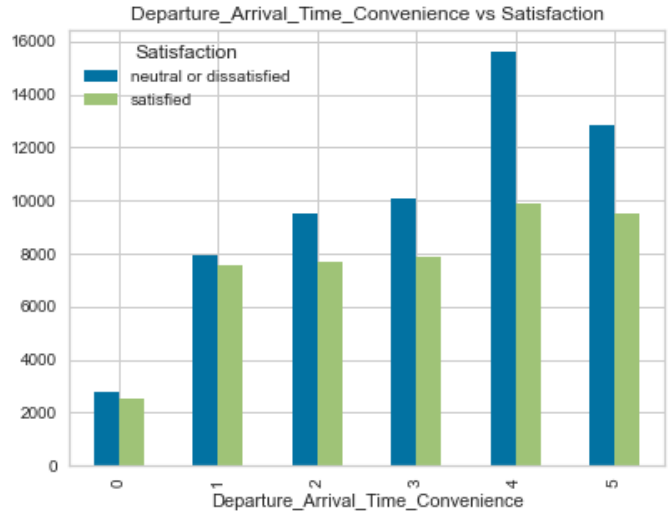



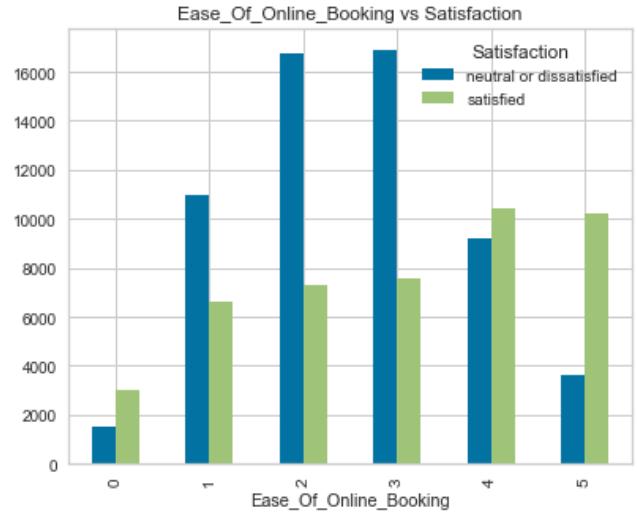



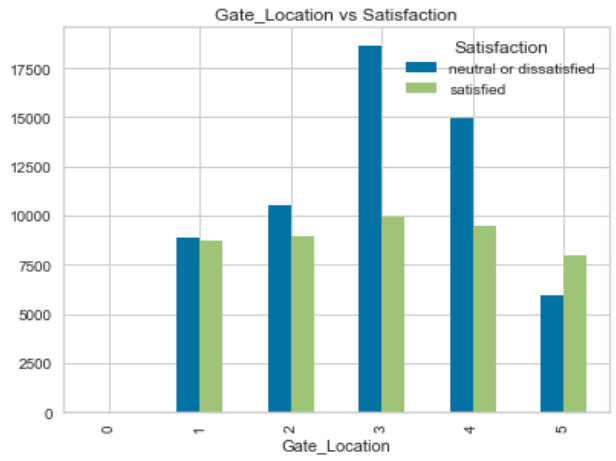



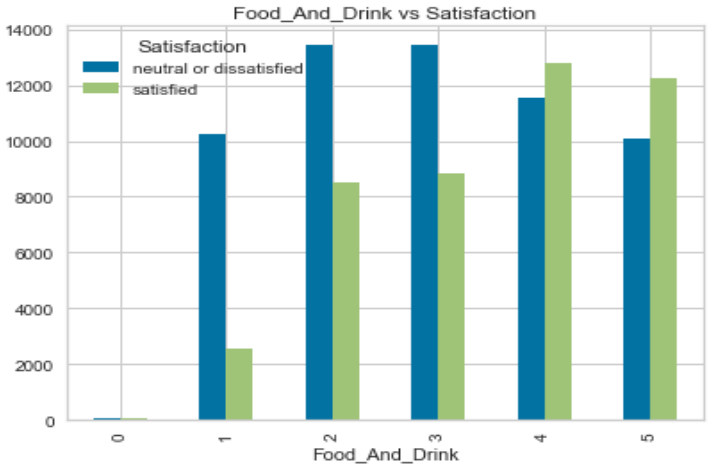

Satisfaction vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
60000
50000
40000
30000
20000
10000
0
neutral or dissatisfied
satisfied
Satisfaction

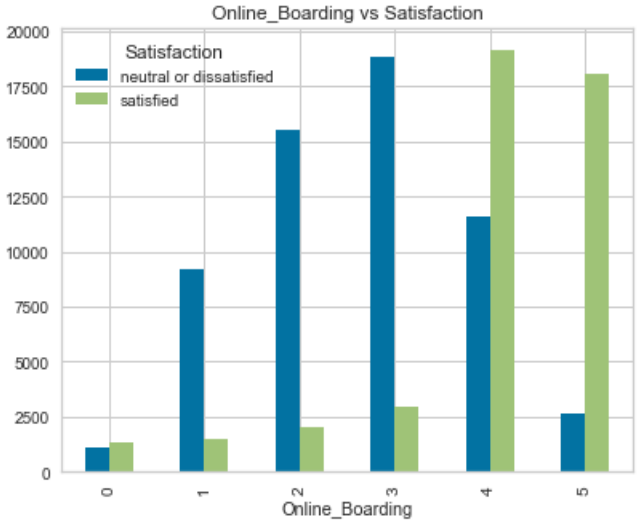

Inflight_Wifi_Service vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Inflight_Wifi_Service

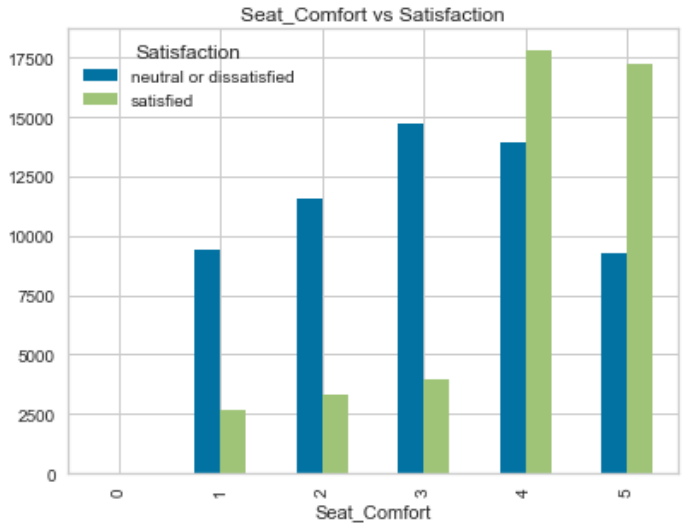

Departure_Arrival_Time_Convenience vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Departure_Arrival_Time_Convenience

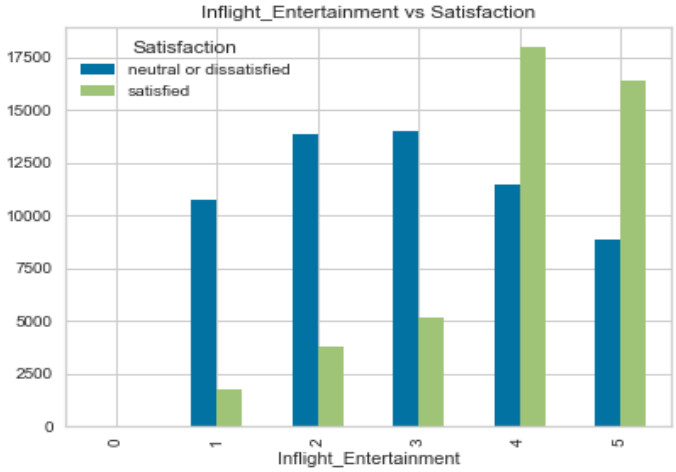

Ease_Of_Online_Booking vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Ease_Of_Online_Booking

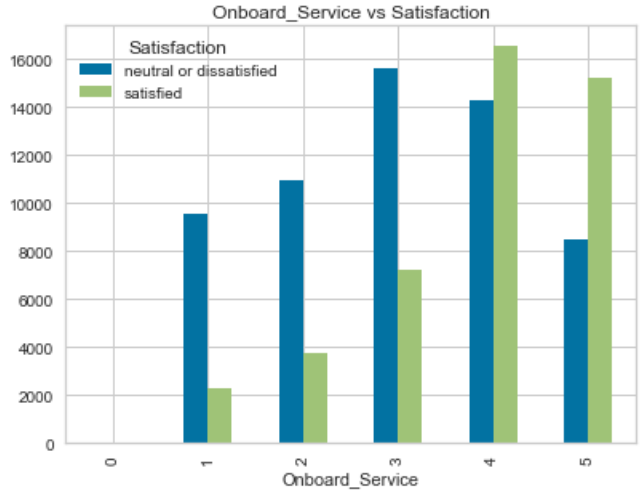

Gate_Location vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Gate_Location

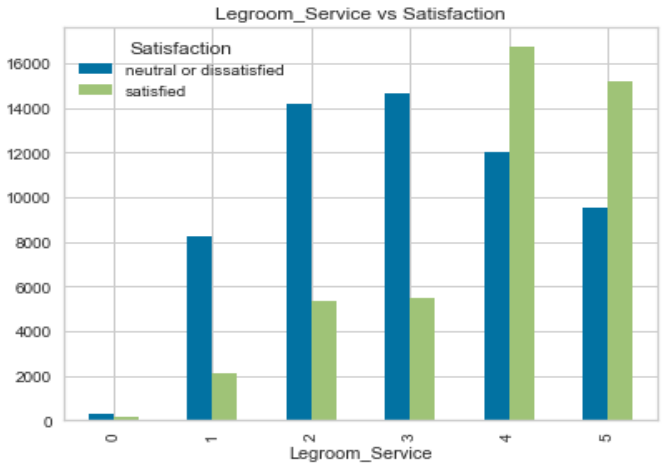

Food_And_Drink vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Food_And_Drink

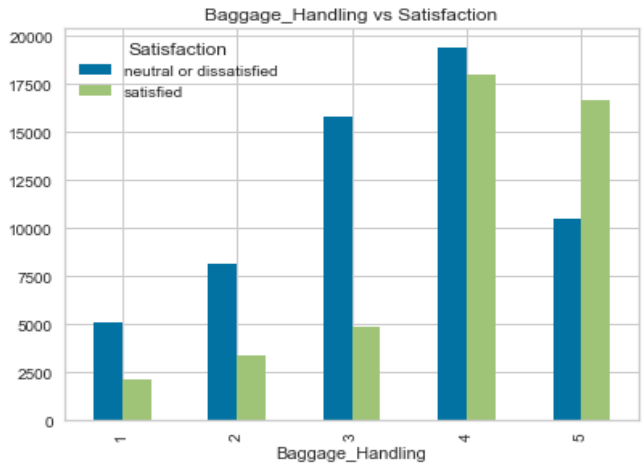

Online_Boarding vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
Online_Boarding

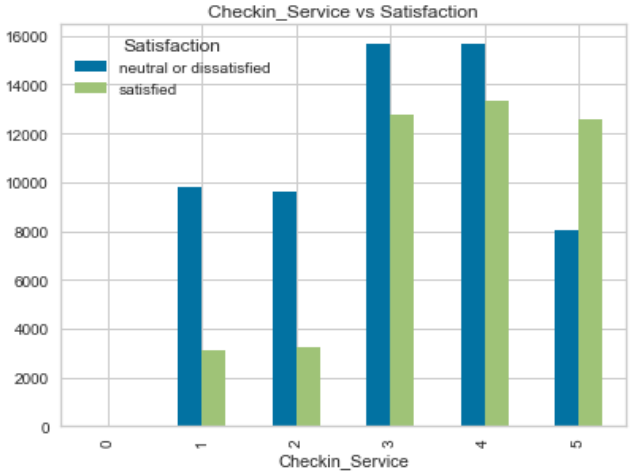

Seat_Comfort vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
Seat_Comfort

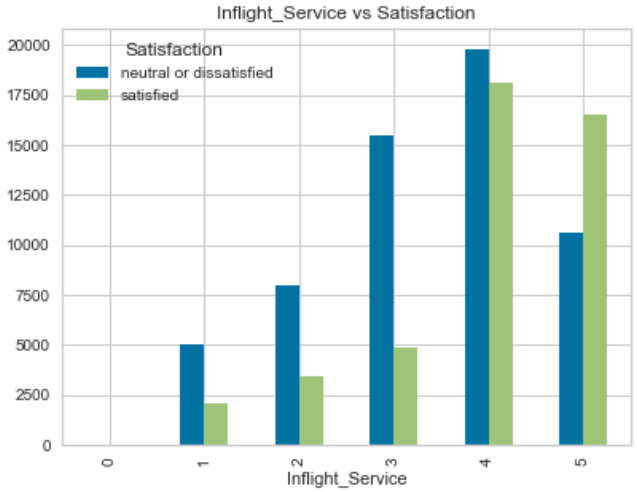

Inflight_Entertainment vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
Inflight_Entertainment

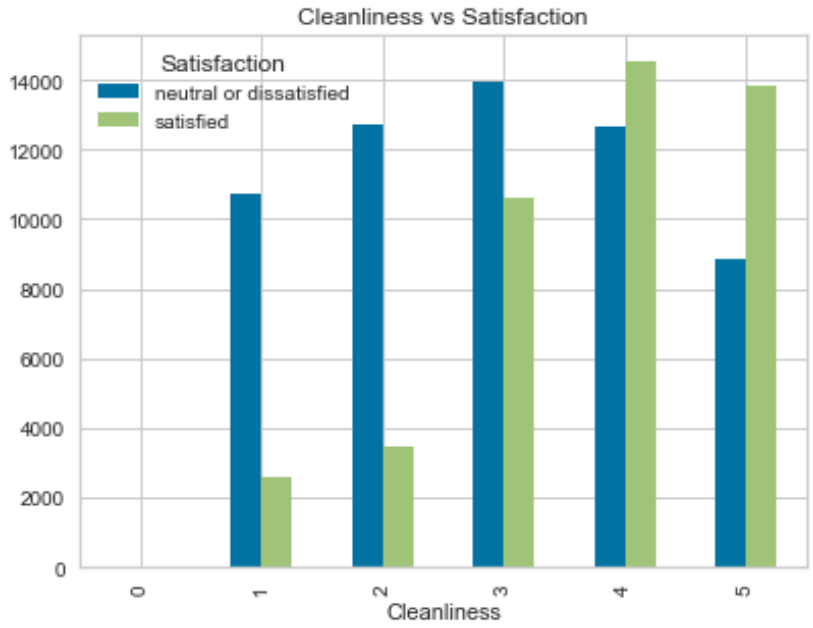

Onboard_Service vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
Onboard_Service


Legroom_Service vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
Legroom_Service


Baggage_Handling vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
Baggage_Handling

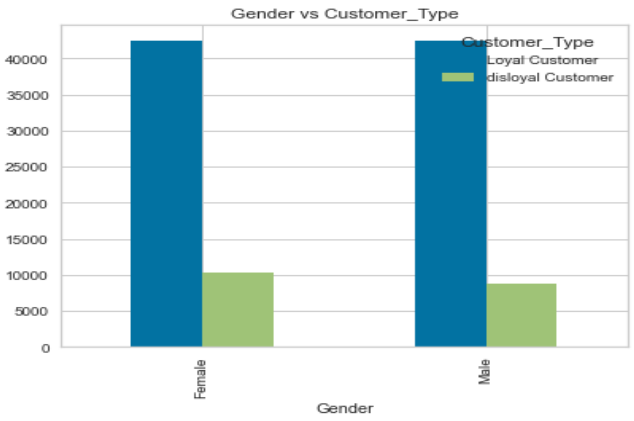

Checkin_Service vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Checkin_Service

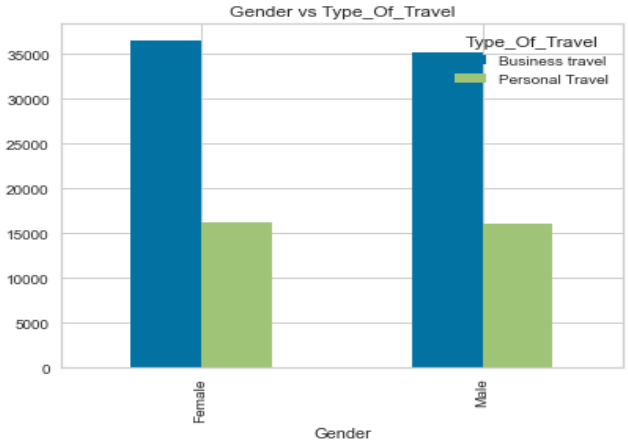

Inflight_Service vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Inflight_Service

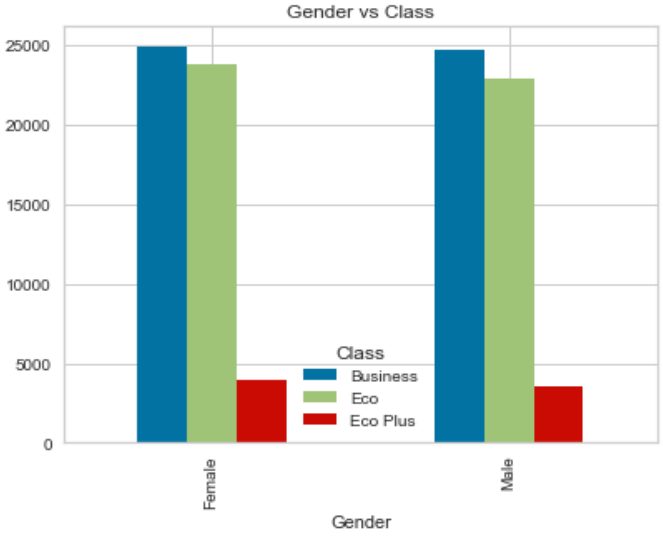

Cleanliness vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Cleanliness

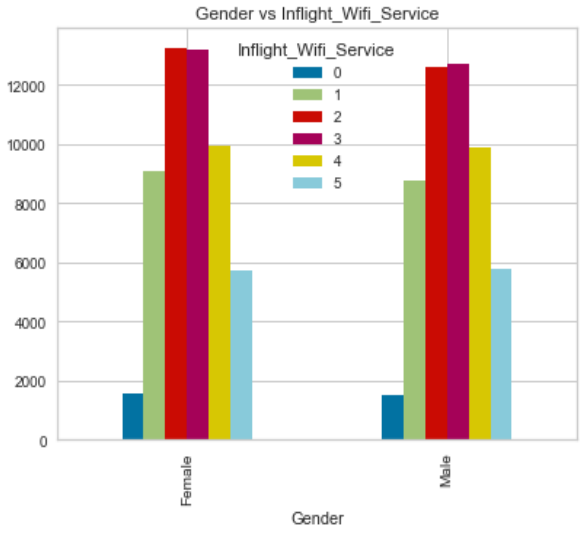
### 10.1.3 Cross Tabs - Categorical Features with Each Other

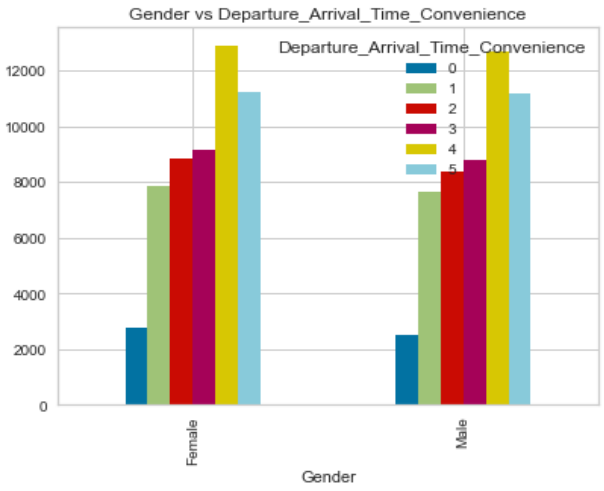
```
for col in airlineData_CategoricalColumns:
    for col2 in airlineData_CategoricalColumns:
        if col!=col2:
            displayCrossTabs(airlineData, [col], col2)
```



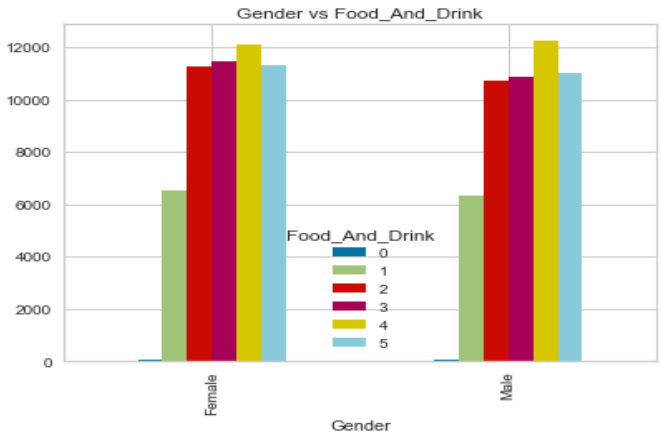



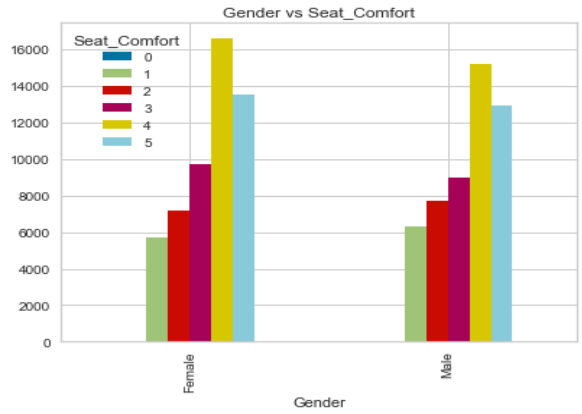



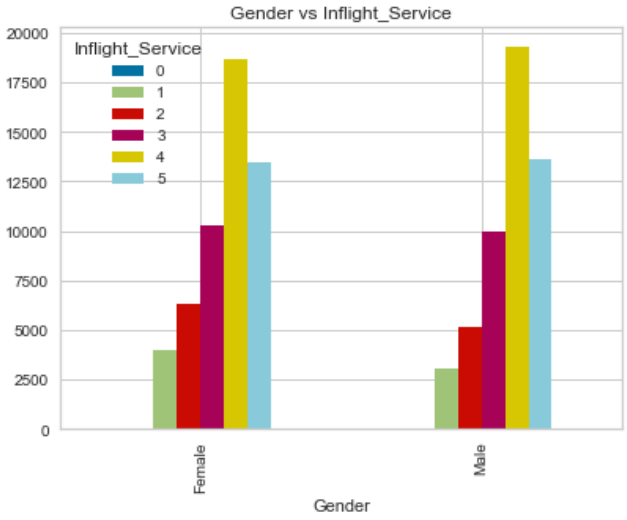



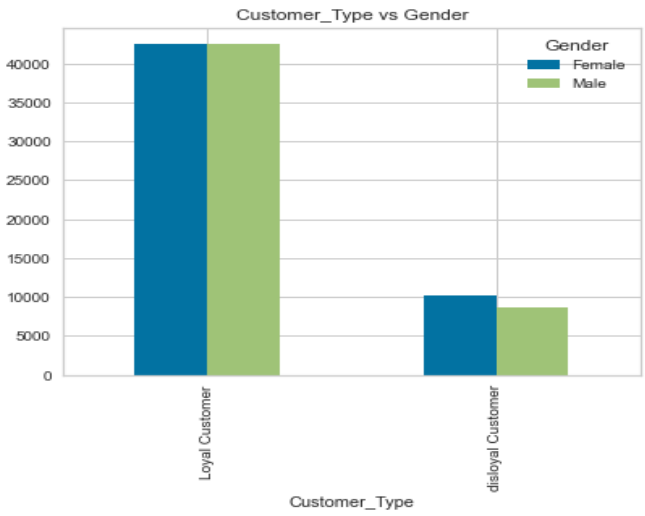

Gender vs Departure_Arrival_Time_Convenience
Departure_Arrival_Time_Convenience
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
Female
Male
Gender

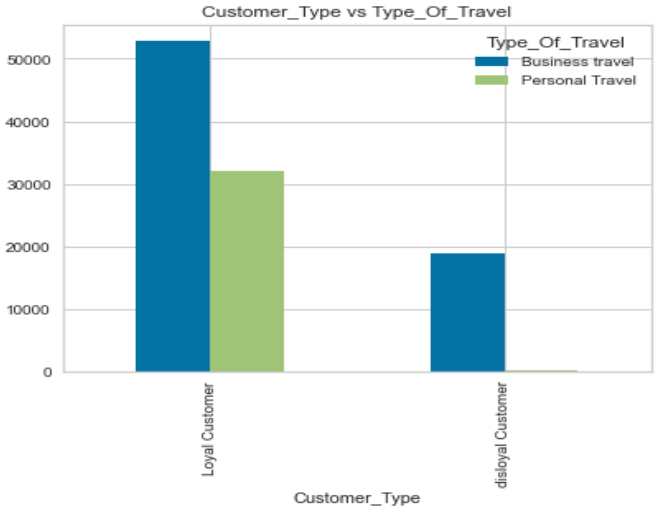

Gender vs Ease_Of_Online_Booking
Ease_Of_Online_Booking
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
Female
Male
Gender


Gender vs Food_And_Drink
Food_And_Drink
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
Female
Male
Gender


Gender vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
16000
14000
12000
10000
8000
6000
4000
2000
0
Female
Male
Gender

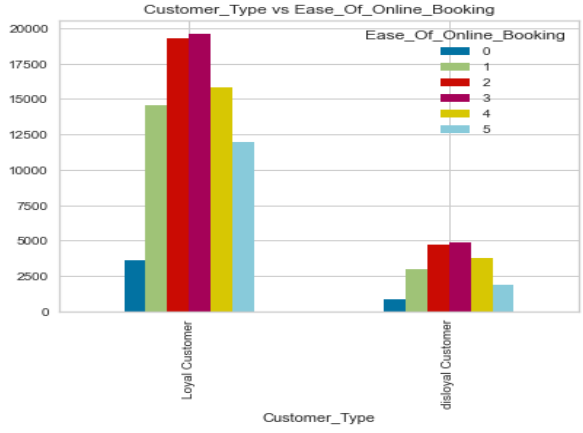

Gender vs Inflight_Service
Inflight_Service
0
1
2
3
4
5
20000
17500
15000
12500
10000
7500
5000
2500
0
Female
Male
Gender

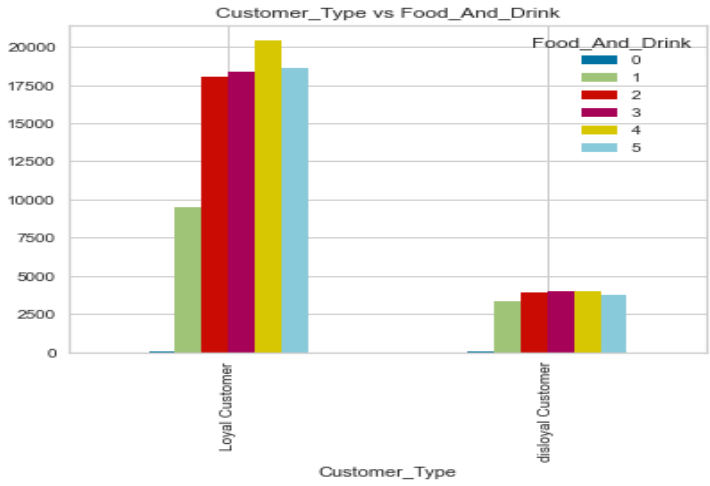

Customer_Type vs Gender
Gender
Female
Male
40000
35000
30000
25000
20000
15000
10000
5000
0
Loyal Customer
disloyal Customer
Customer_Type

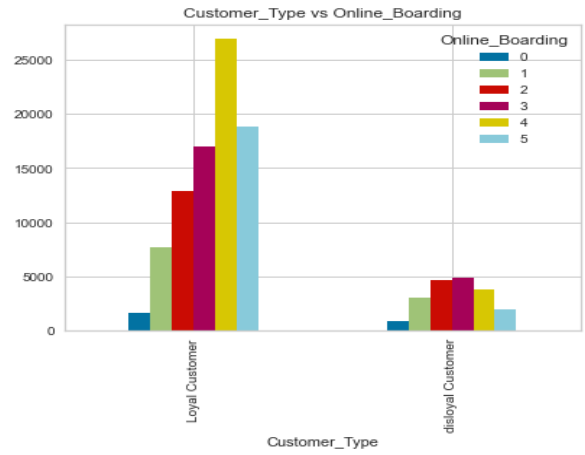

Customer_Type vs Type_Of_Travel
Type_Of_Travel
Business travel
Personal Travel
50000
40000
30000
20000
10000
0
Loyal Customer
disloyal Customer
Customer_Type

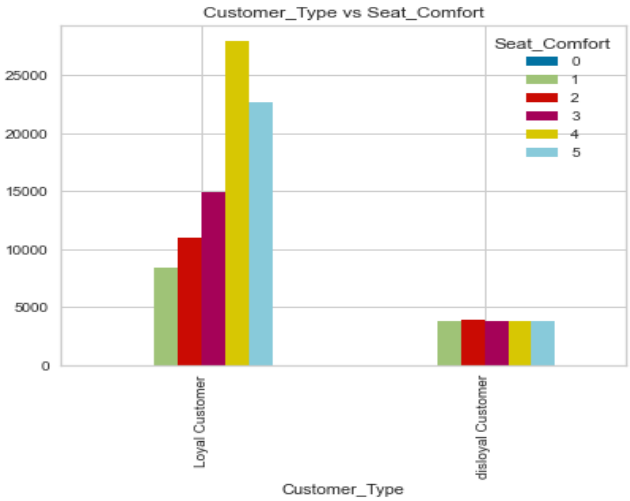

Customer_Type vs Class
Class
Business
Eco
Eco Plus
40000
35000
30000
25000
20000
15000
10000
5000
0
Loyal Customer
disloyal Customer
Customer_Type

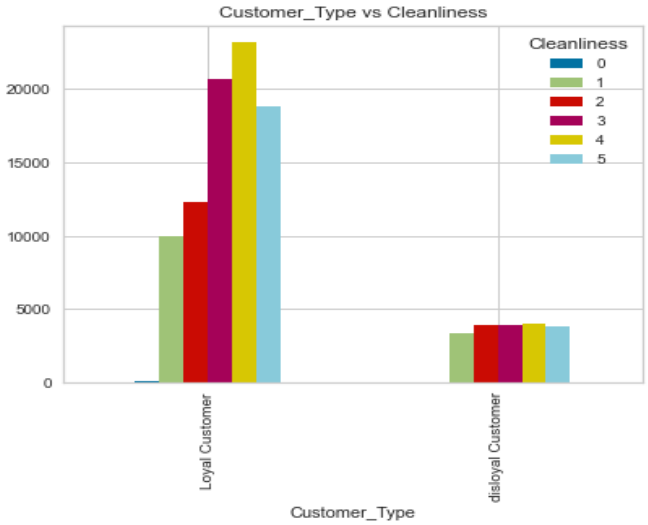

Customer_Type vs Inflight_Wifi_Service
Inflight_Wifi_Service
0
1
2
3
4
5
20000
17500
15000
12500
10000
7500
5000
2500
0
Loyal Customer
disloyal Customer
Customer_Type

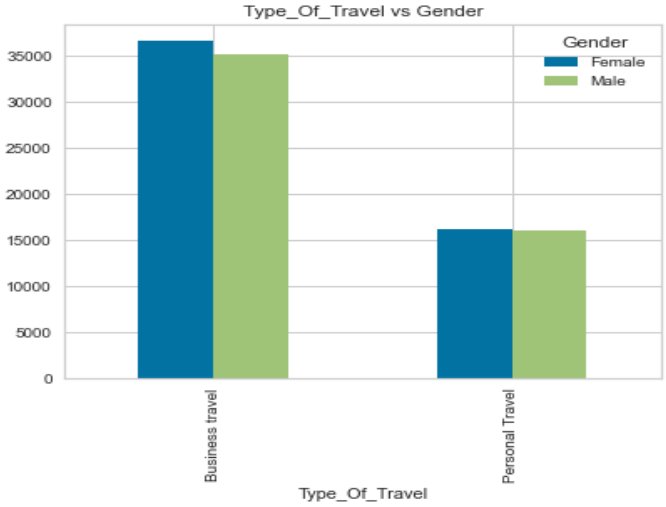

Customer_Type vs Ease_Of_Online_Booking
Ease_Of_Online_Booking
0
1
2
3
4
5
20000
17500
15000
12500
10000
7500
5000
2500
0
Loyal Customer
disloyal Customer
Customer_Type

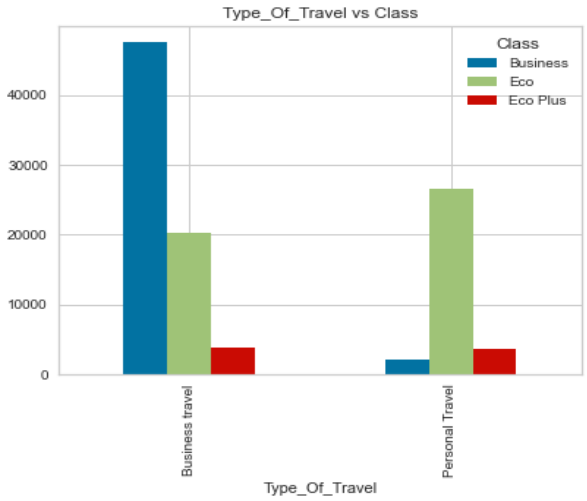

Customer_Type vs Food_And_Drink
Food_And_Drink
0
1
2
3
4
5
20000
17500
15000
12500
10000
7500
5000
2500
0
Loyal Customer
disloyal Customer
Customer_Type

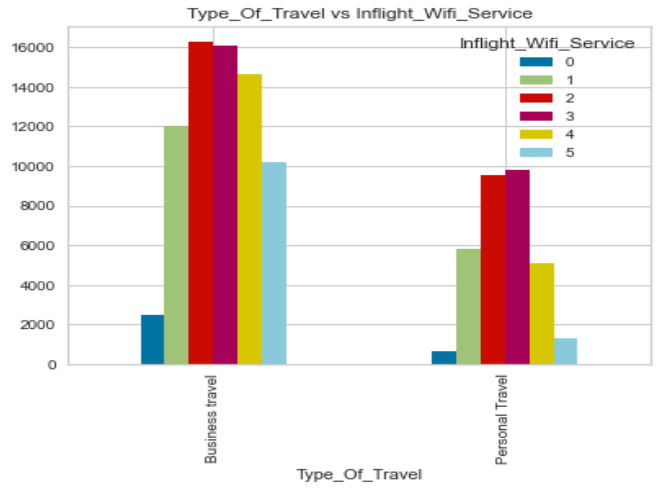

Customer_Type vs Online_Boarding
Online_Boarding
0
1
2
3
4
5
25000
20000
15000
10000
5000
0
Loyal Customer
disloyal Customer
Customer_Type

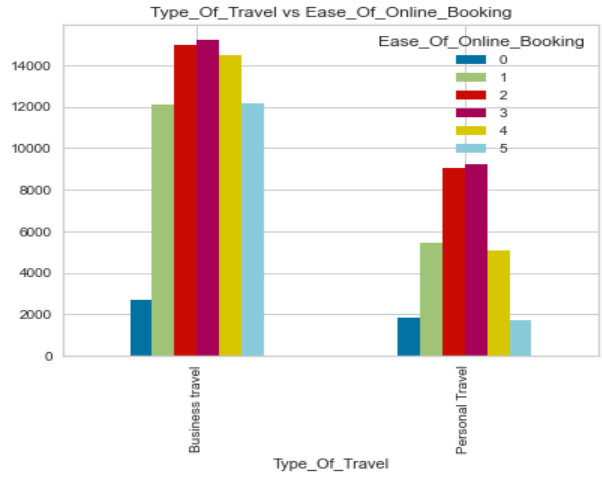

Customer_Type vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
25000
20000
15000
10000
5000
0
Loyal Customer
disloyal Customer
Customer_Type

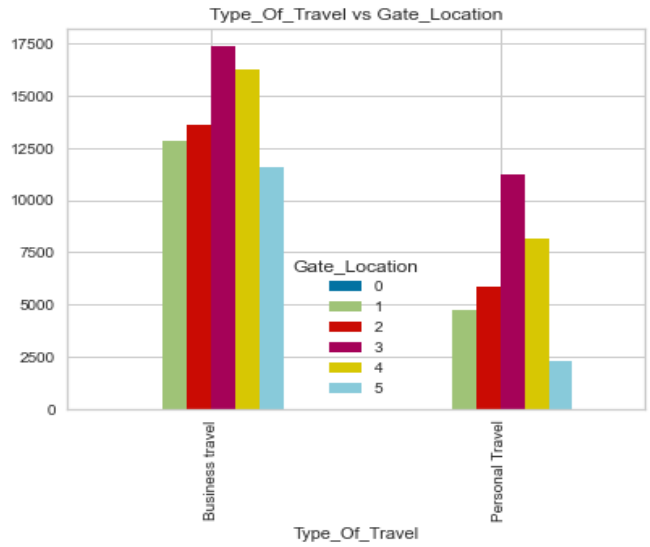

Customer_Type vs Cleanliness
Cleanliness
0
1
2
3
4
5
20000
15000
10000
5000
0
Loyal Customer
disloyal Customer
Customer_Type

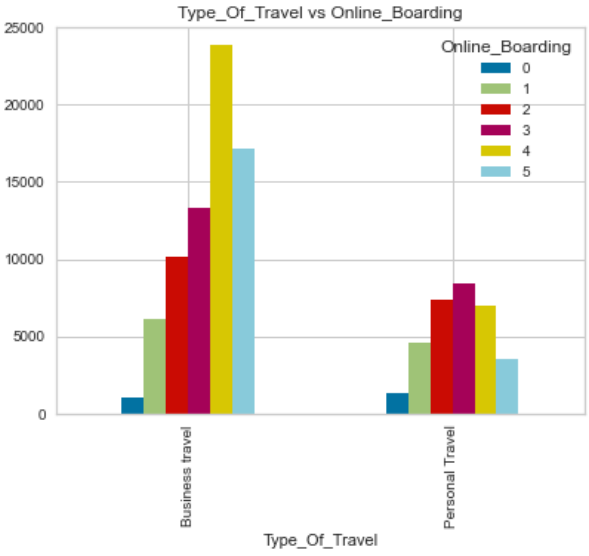

Type_Of_Travel vs Gender
Gender
Female
Male
35000
30000
25000
20000
15000
10000
5000
0
Business travel
Personal Travel
Type_Of_Travel

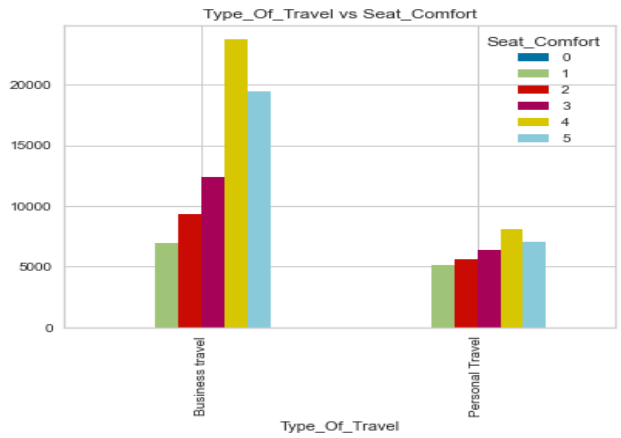

Type_Of_Travel vs Class
Class
Business
Eco
Eco Plus
40000
30000
20000
10000
0
Business travel
Personal Travel
Type_Of_Travel

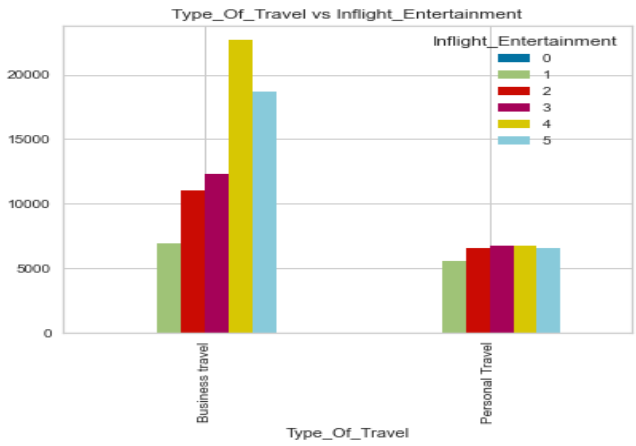

Type_Of_Travel vs Inflight_Wifi_Service
Inflight_Wifi_Service
0
1
2
3
4
5
16000
14000
12000
10000
8000
6000
4000
2000
0
Business travel
Personal Travel
Type_Of_Travel

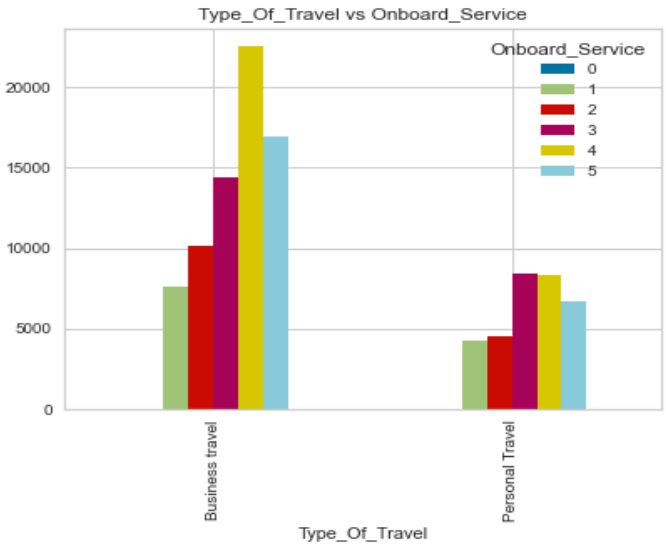

Type_Of_Travel vs Ease_Of_Online_Booking
Ease_Of_Online_Booking
0
1
2
3
4
5
14000
12000
10000
8000
6000
4000
2000
0
Business travel
Personal Travel
Type_Of_Travel

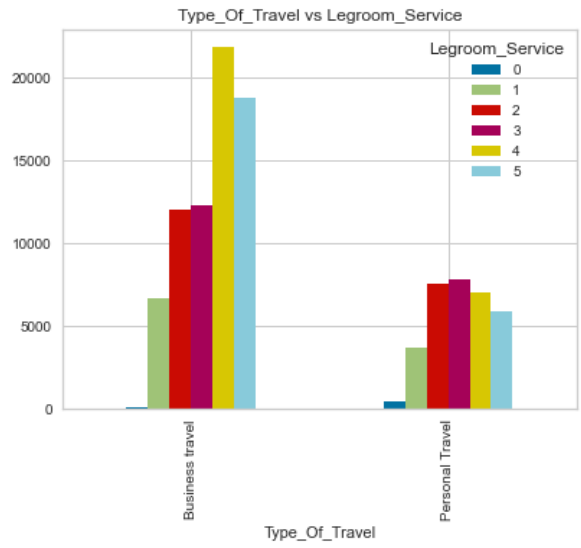

Type_Of_Travel vs Gate_Location
Gate_Location
0
1
2
3
4
5
17500
15000
12500
10000
7500
5000
2500
0
Business travel
Personal Travel
Type_Of_Travel

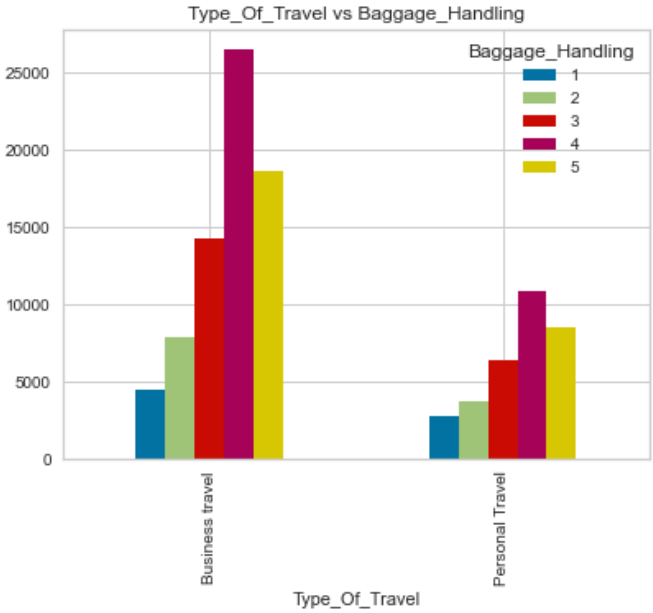

Type_Of_Travel vs Online_Boarding
Online_Boarding
0
1
2
3
4
5
25000
20000
15000
10000
5000
0
Business travel
Personal Travel
Type_Of_Travel

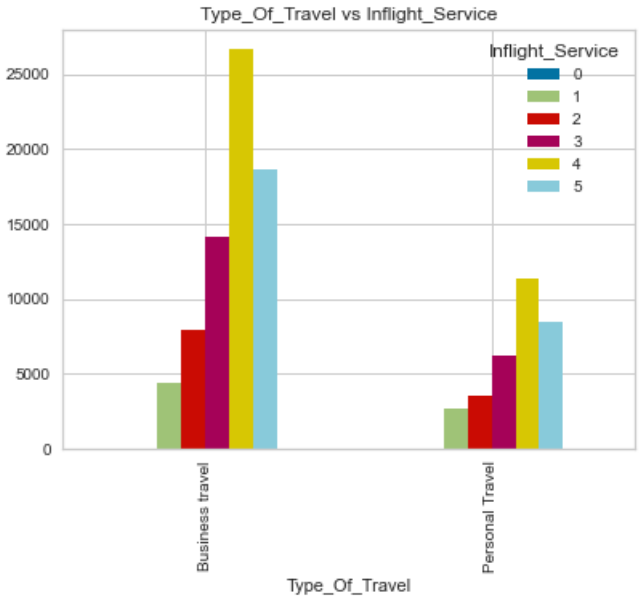

Type_Of_Travel vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
20000
15000
10000
5000
0
Business travel
Personal Travel
Type_Of_Travel


Type_Of_Travel vs Inflight_Entertainment
Inflight_Entertainment
0
1
2
3
4
5
20000
15000
10000
5000
0
Business travel
Personal Travel
Type_Of_Travel

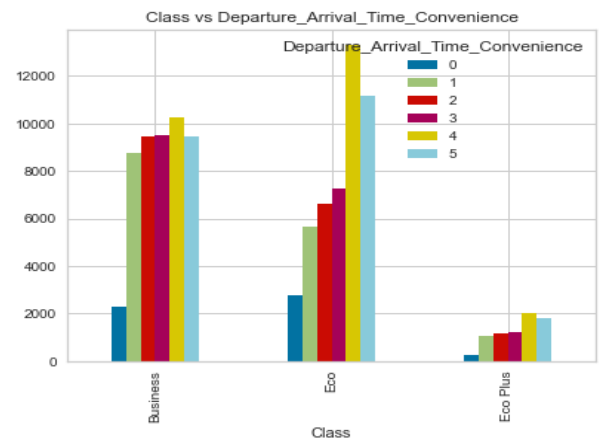

Type_Of_Travel vs Onboard_Service
Onboard_Service
0
1
2
3
4
5
20000
15000
10000
5000
0
Business travel
Personal Travel
Type_Of_Travel

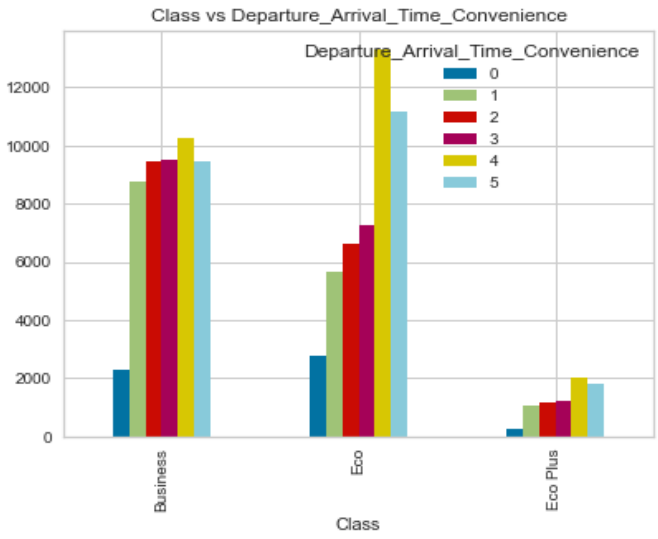

Type_Of_Travel vs Legroom_Service
Legroom_Service
0
1
2
3
4
5
20000
15000
10000
5000
0
Business travel
Personal Travel
Type_Of_Travel

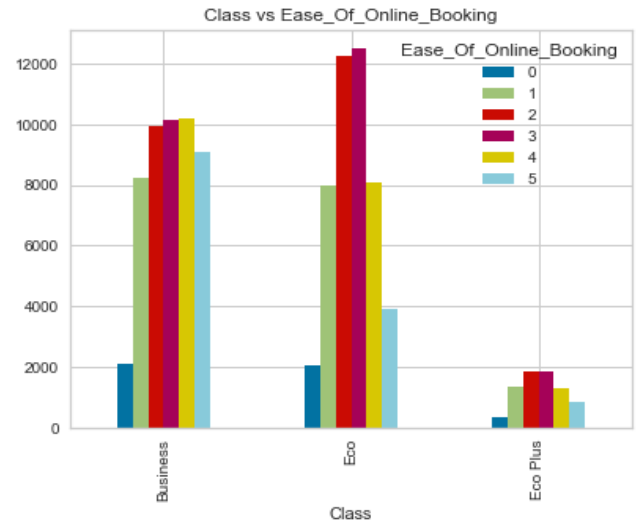

Type_Of_Travel vs Baggage_Handling
Baggage_Handling
1
2
3
4
5
25000
20000
15000
10000
5000
0
Business travel
Personal Travel
Type_Of_Travel

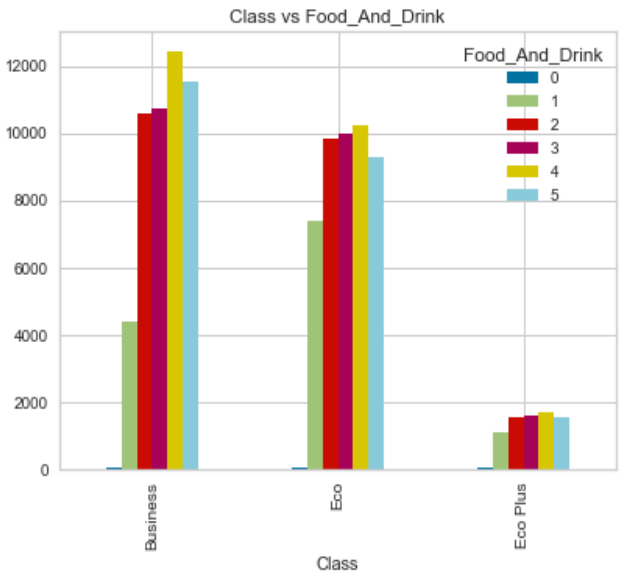

Type_Of_Travel vs Inflight_Service
Inflight_Service
0
1
2
3
4
5
25000
20000
15000
10000
5000
0
Business travel
Personal Travel
Type_Of_Travel

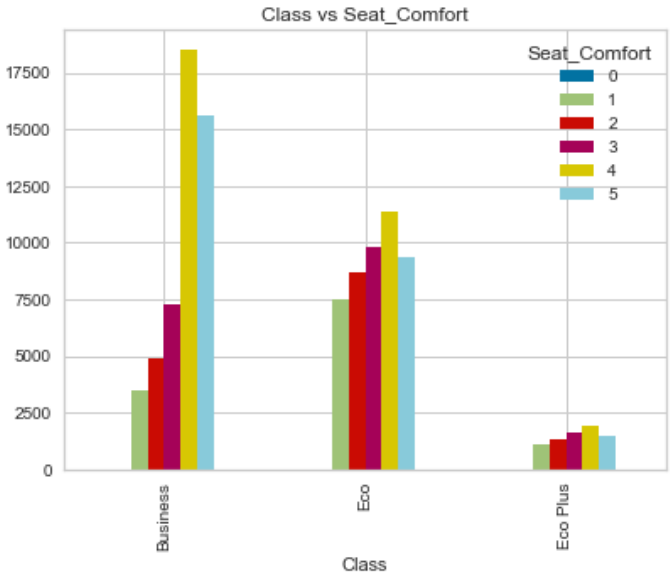

Class vs Inflight_Wifi_Service
Inflight_Wifi_Service
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
Business
Eco
Eco Plus
Class

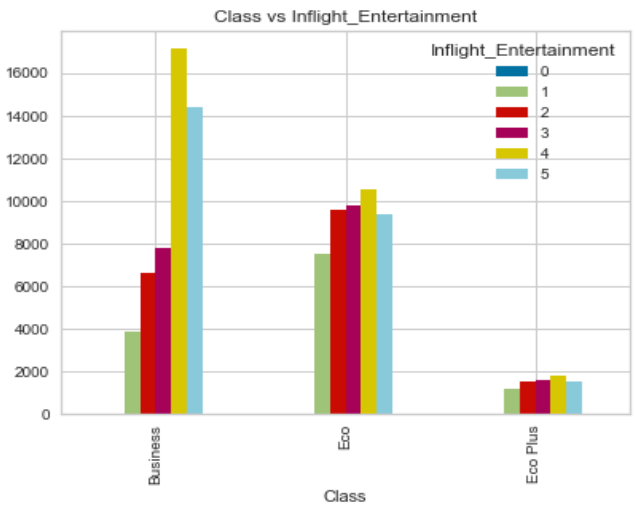

Class vs Departure_Arrival_Time_Convenience
Departure_Arrival_Time_Convenience
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
Business
Eco
Eco Plus
Class


Class vs Departure_Arrival_Time_Convenience
Departure_Arrival_Time_Convenience
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
Business
Eco
Eco Plus
Class

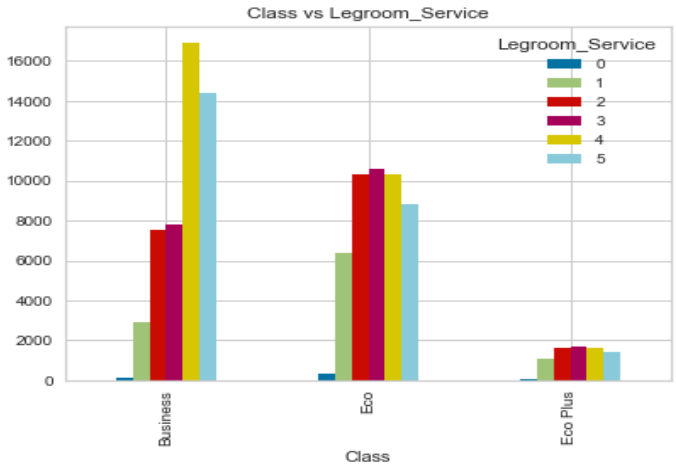

Class vs Ease_Of_Online_Booking
Ease_Of_Online_Booking
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
Business
Eco
Eco Plus
Class

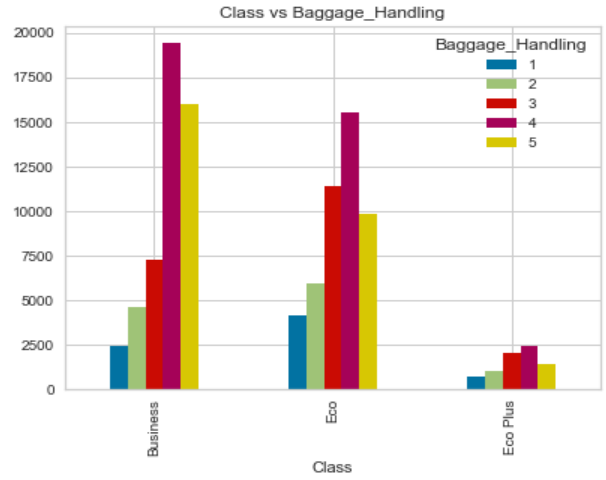

Class vs Food_And_Drink
Food_And_Drink
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
Business
Eco
Eco Plus
Class

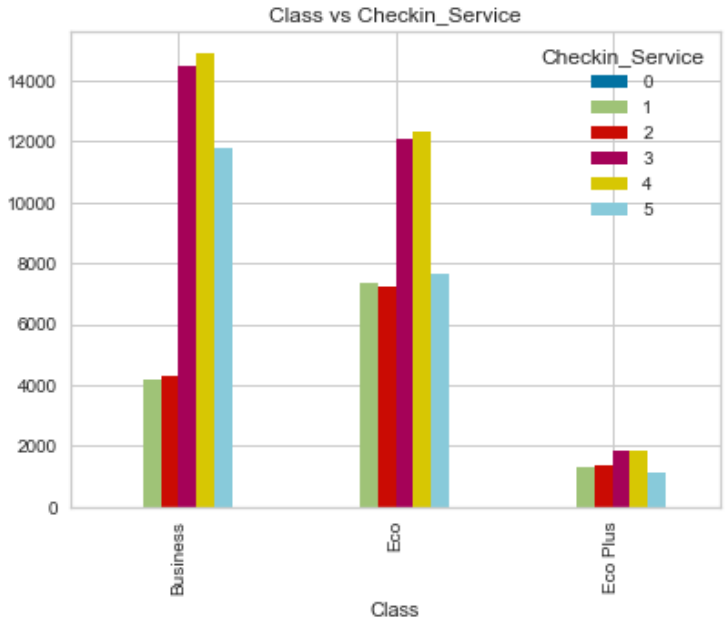

Class vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
17500
15000
12500
10000
7500
5000
2500
0
Business
Eco
Eco Plus
Class

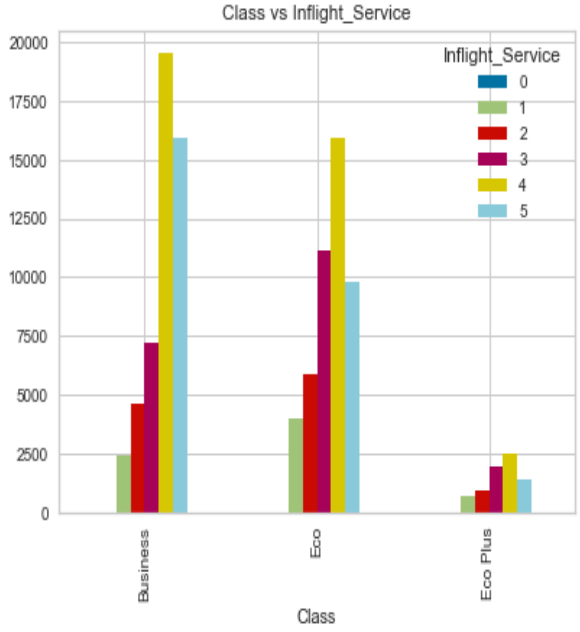

Class vs Inflight_Entertainment
Inflight_Entertainment
0
1
2
3
4
5
16000
14000
12000
10000
8000
6000
4000
2000
0
Business
Eco
Eco Plus
Class

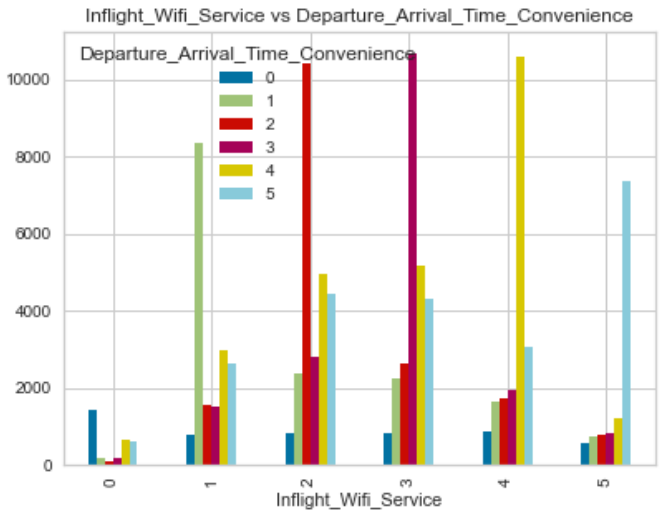

Class vs Onboard_Service
Onboard_Service
0
1
2
3
4
5
17500
15000
12500
10000
7500
5000
2500
0
Business
Eco
Eco Plus
Class

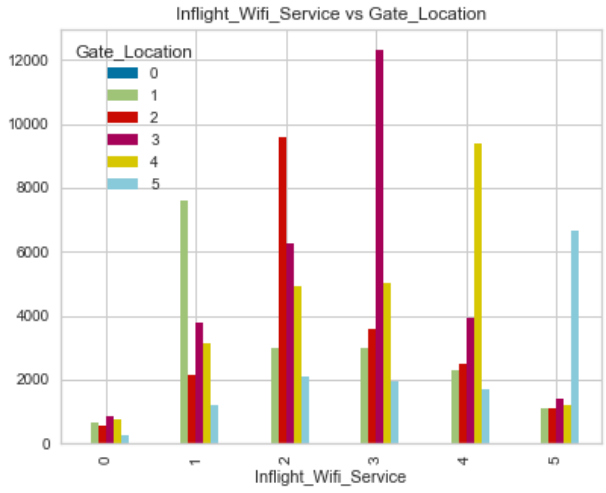

Class vs Legroom_Service
Legroom_Service
0
1
2
3
4
5
16000
14000
12000
10000
8000
6000
4000
2000
0
Business
Eco
Eco Plus
Class

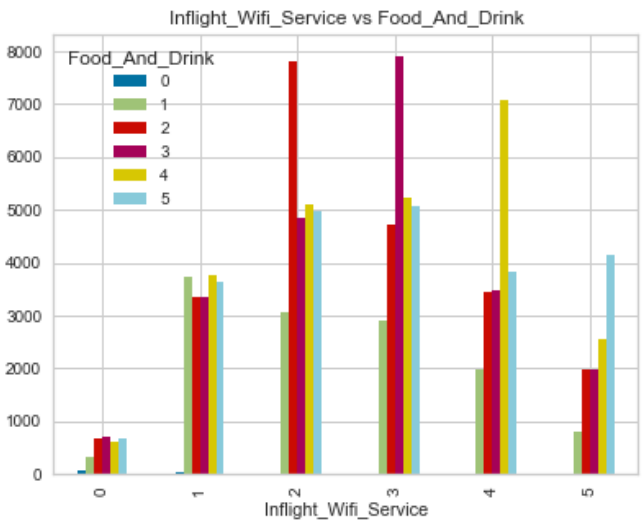

Class vs Baggage_Handling
Baggage_Handling
1
2
3
4
5
20000
17500
15000
12500
10000
7500
5000
2500
0
Business
Eco
Eco Plus
Class

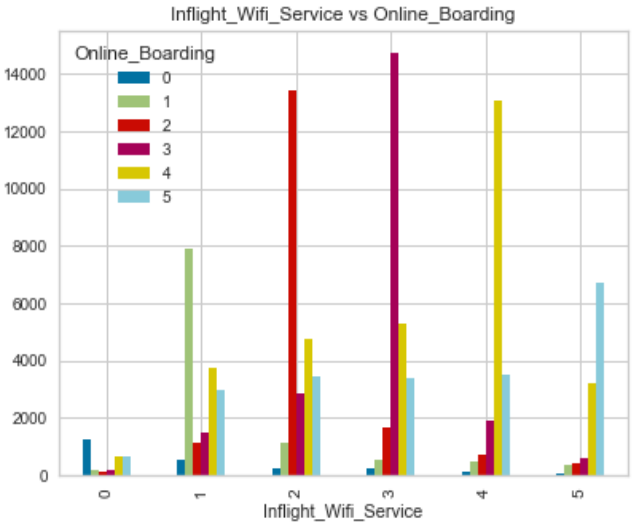

Class vs Checkin_Service
Checkin_Service
0
1
2
3
4
5
14000
12000
10000
8000
6000
4000
2000
0
Business
Eco
Eco Plus
Class

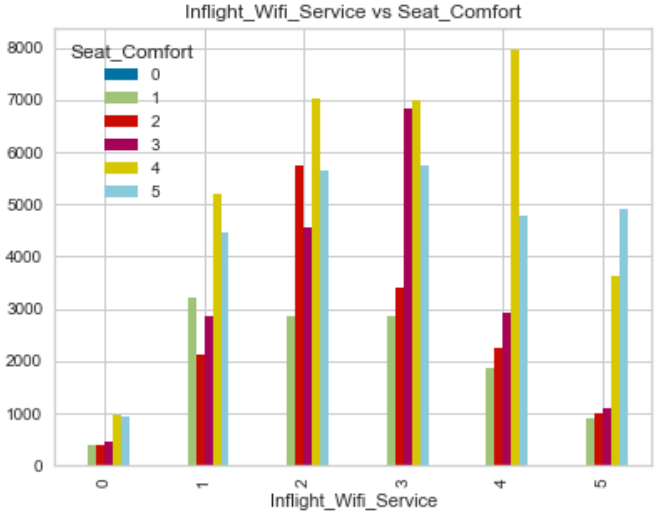

Class vs Inflight_Service
Inflight_Service
0
1
2
3
4
5
20000
17500
15000
12500
10000
7500
5000
2500
0
Business
Eco
Eco Plus
Class

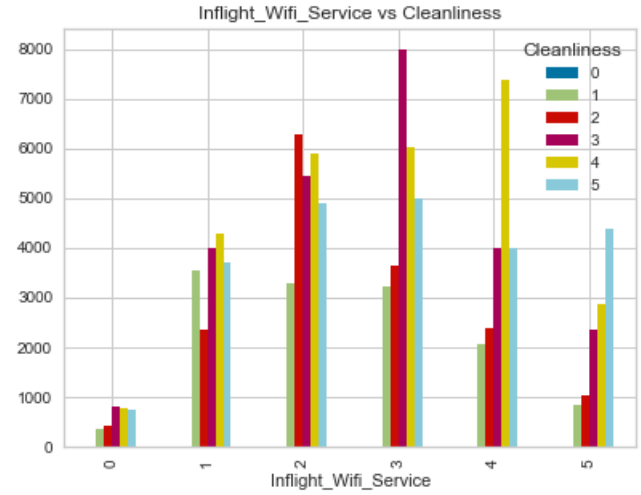

Inflight_Wifi_Service vs Departure_Arrival_Time_Convenience
Departure_Arrival_Time_Convenience
0
1
2
3
4
5
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Inflight_Wifi_Service

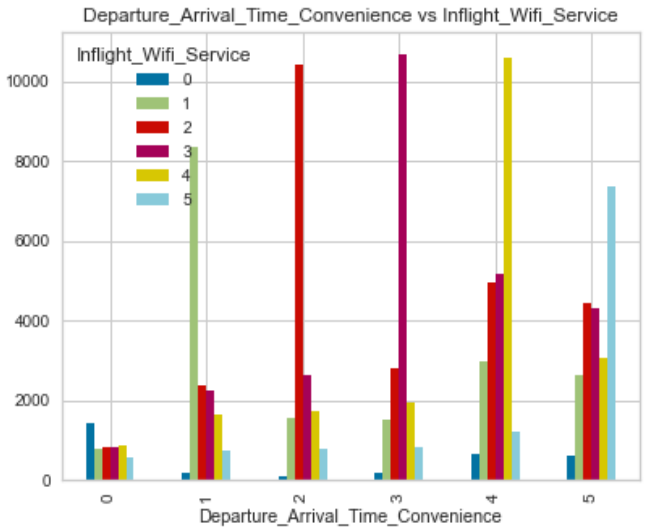

Inflight_Wifi_Service vs Gate_Location
Gate_Location
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Inflight_Wifi_Service

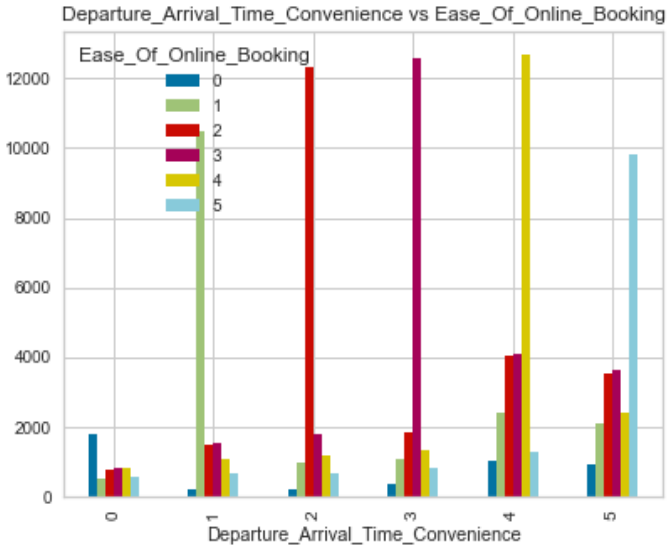

Inflight_Wifi_Service vs Food_And_Drink
Food_And_Drink
0
1
2
3
4
5
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Inflight_Wifi_Service

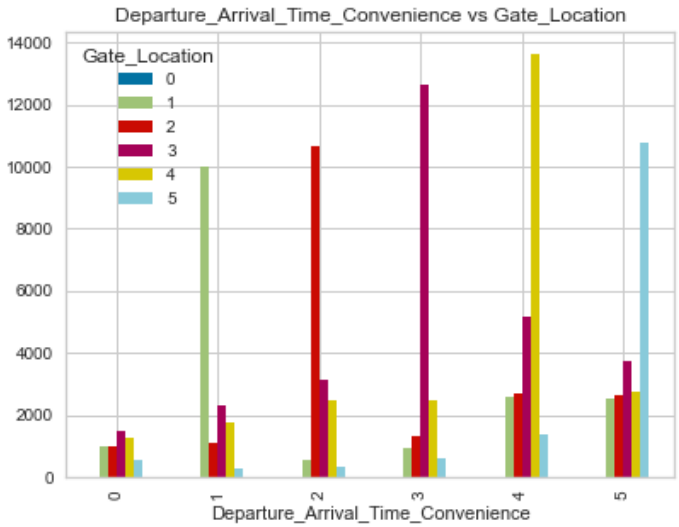

Inflight_Wifi_Service vs Online_Boarding
Online_Boarding
0
1
2
3
4
5
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Inflight_Wifi_Service

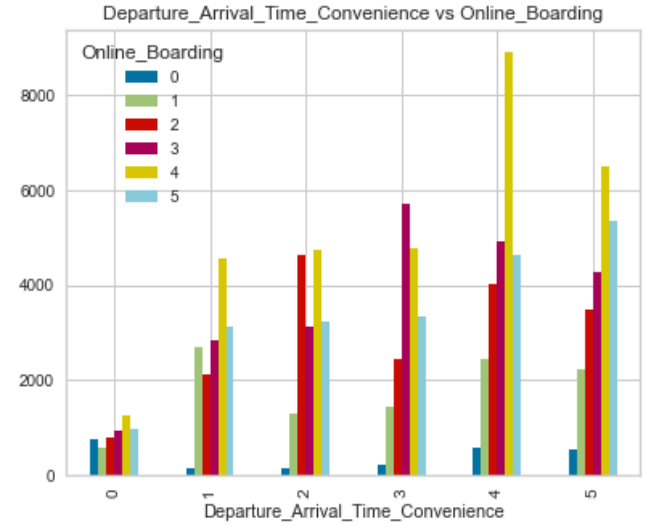

Inflight_Wifi_Service vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Inflight_Wifi_Service

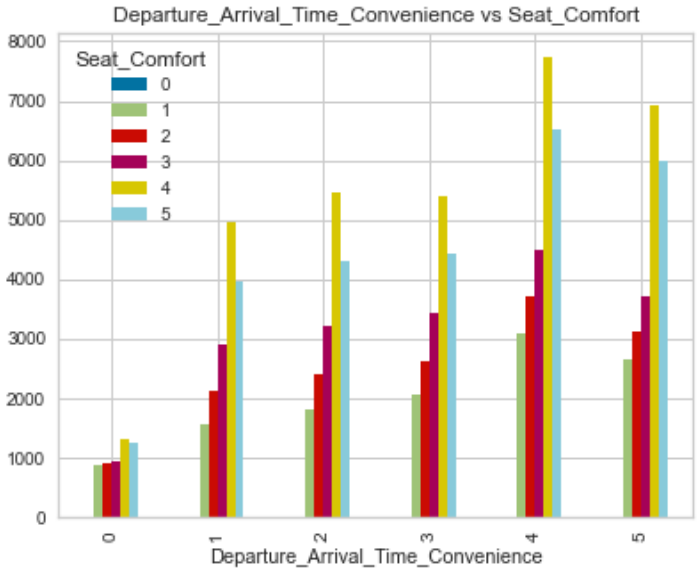

Inflight_Wifi_Service vs Cleanliness
Cleanliness
0
1
2
3
4
5
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Inflight_Wifi_Service

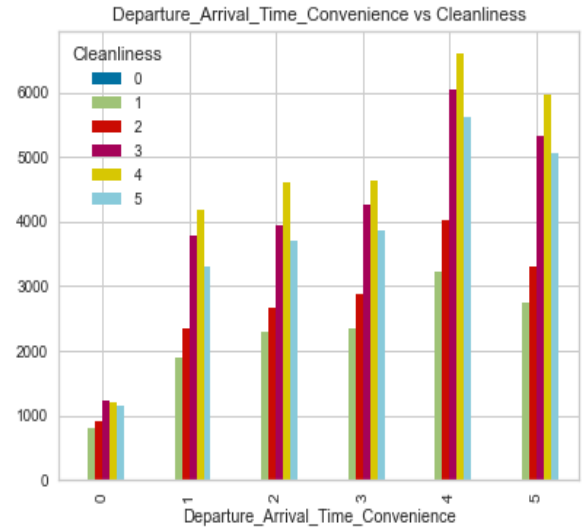

Departure_Arrival_Time_Convenience vs Inflight_Wifi_Service
Inflight_Wifi_Service
0
1
2
3
4
5
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Departure_Arrival_Time_Convenience

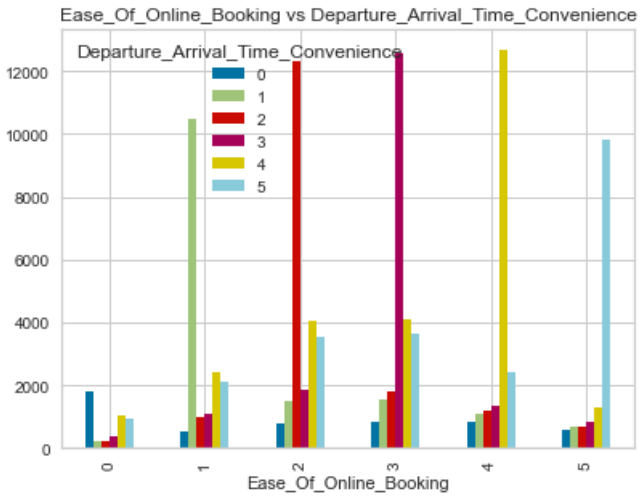

Departure_Arrival_Time_Convenience vs Ease_Of_Online_Booking
Ease_Of_Online_Booking
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Departure_Arrival_Time_Convenience


Departure_Arrival_Time_Convenience vs Gate_Location
Gate_Location
0
1
2
3
4
5
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Departure_Arrival_Time_Convenience


Departure_Arrival_Time_Convenience vs Online_Boarding
Online_Boarding
0
1
2
3
4
5
8000
6000
4000
2000
0
0
1
2
3
4
5
Departure_Arrival_Time_Convenience

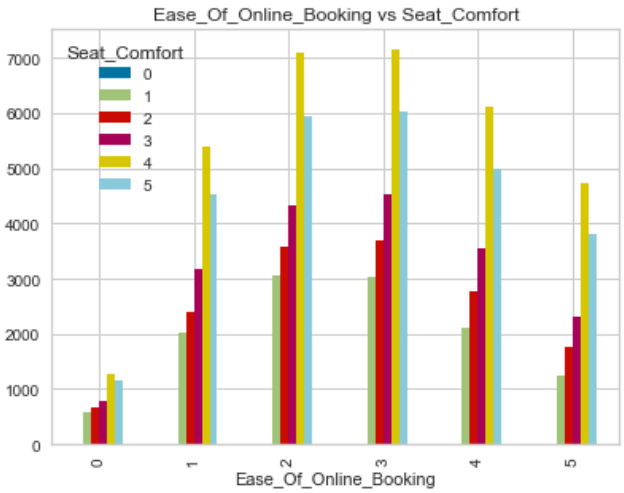

Departure_Arrival_Time_Convenience vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Departure_Arrival_Time_Convenience

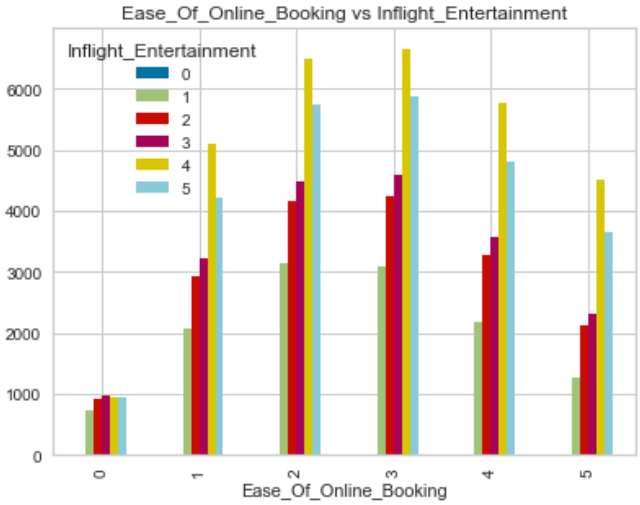

Departure_Arrival_Time_Convenience vs Cleanliness
Cleanliness
0
1
2
3
4
5
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Departure_Arrival_Time_Convenience

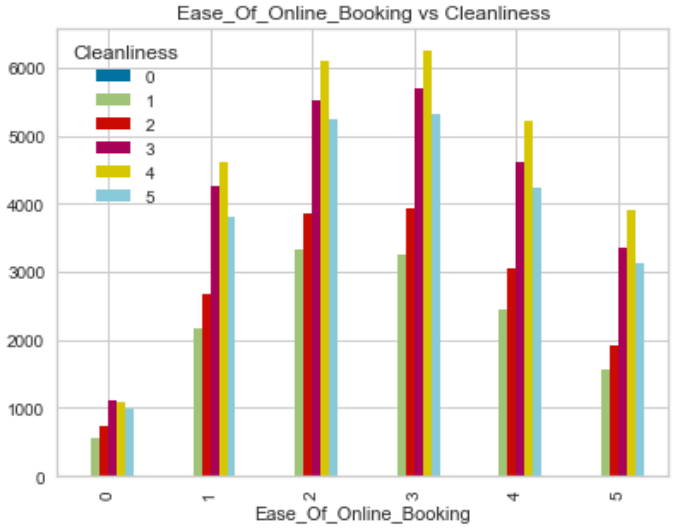

Ease_Of_Online_Booking vs Departure_Arrival_Time_Convenience
Departure_Arrival_Time_Convenience
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Ease_Of_Online_Booking

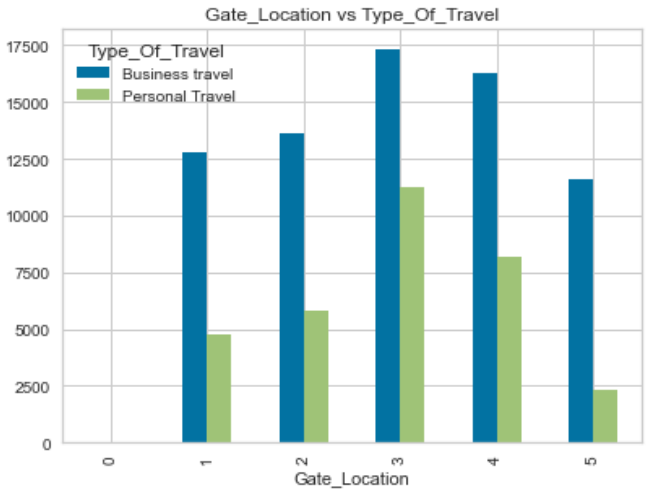

Ease_Of_Online_Booking vs Departure_Arrival_Time_Convenience
Departure_Arrival_Time_Convenience
0
1
2
3
4
5
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Ease_Of_Online_Booking

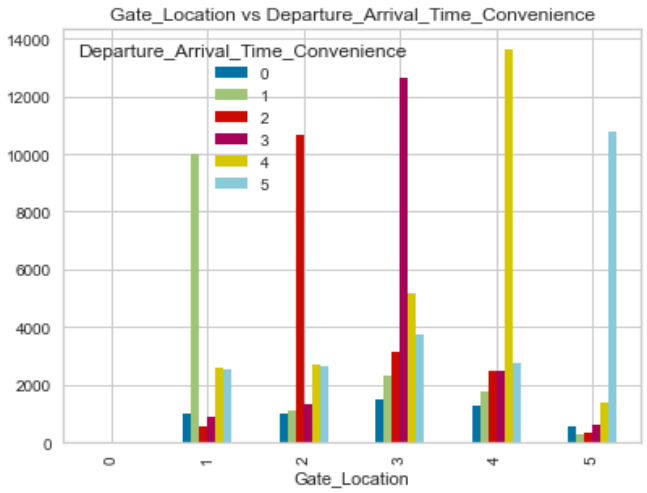

Ease_Of_Online_Booking vs Food_And_Drink
Food_And_Drink
0
1
2
3
4
5
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Ease_Of_Online_Booking

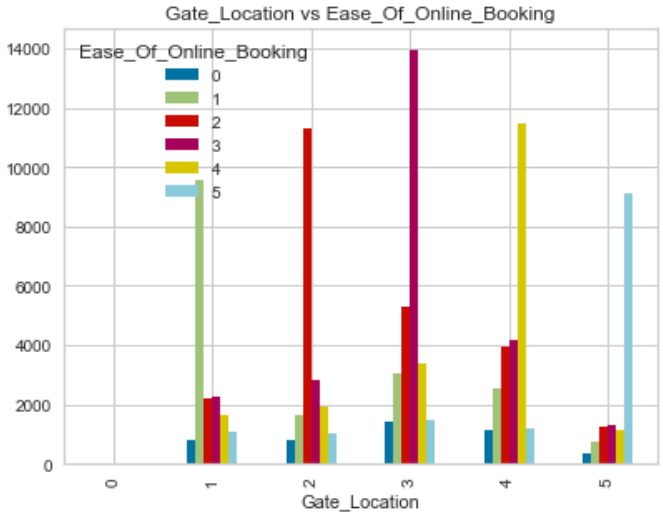

Ease_Of_Online_Booking vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Ease_Of_Online_Booking

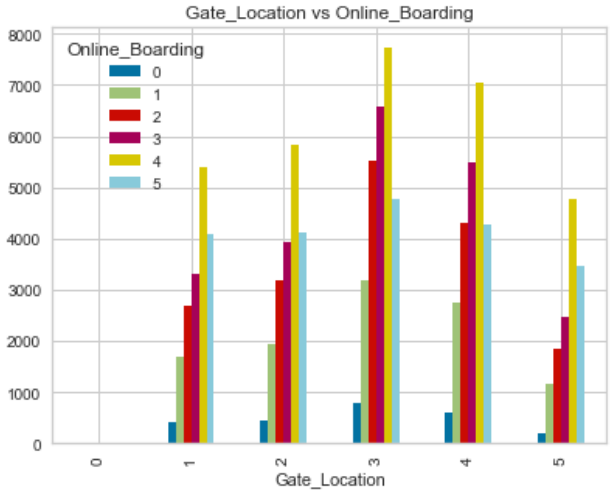

Ease_Of_Online_Booking vs Inflight_Entertainment
Inflight_Entertainment
0
1
2
3
4
5
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Ease_Of_Online_Booking

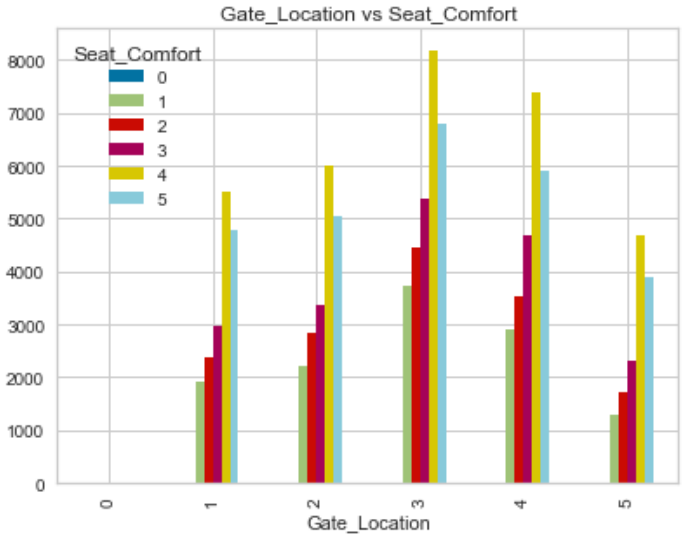

Ease_Of_Online_Booking vs Cleanliness
Cleanliness
0
1
2
3
4
5
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Ease_Of_Online_Booking

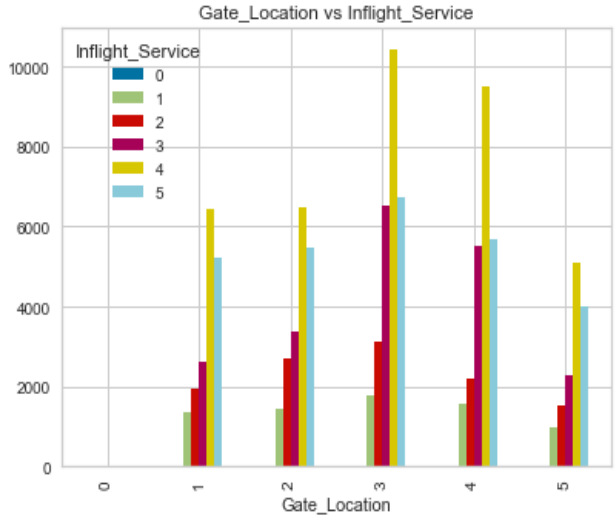

Gate_Location vs Type_Of_Travel
Type_Of_Travel
Business travel
Personal Travel
Gate_Location

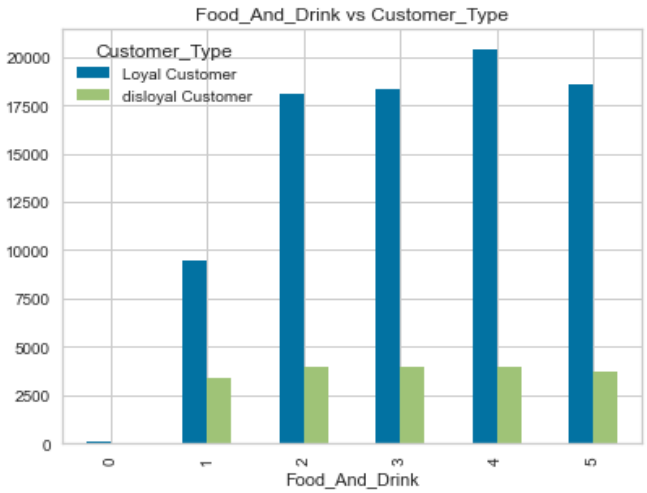

Gate_Location vs Departure_Arrival_Time_Convenience
Departure_Arrival_Time_Convenience
Gate_Location

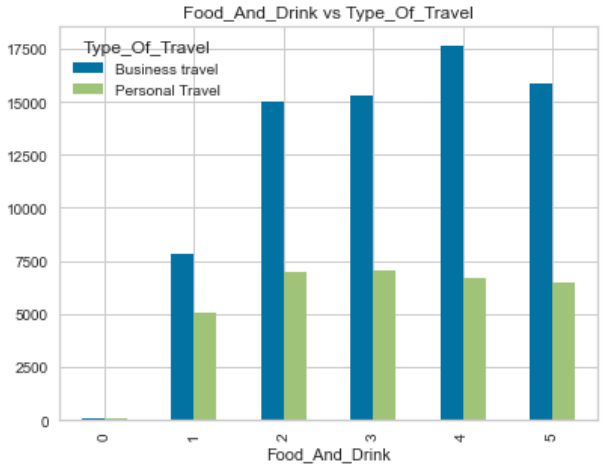

Gate_Location vs Ease_Of_Online_Booking
Ease_Of_Online_Booking
Gate_Location

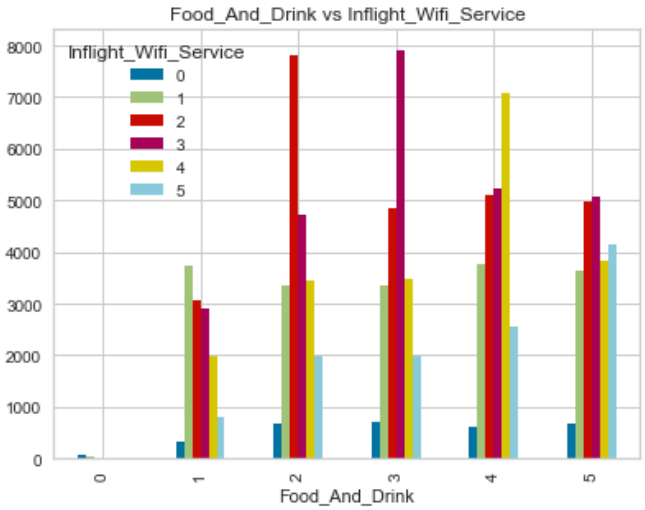

Gate_Location vs Online_Boarding
Online_Boarding
Gate_Location

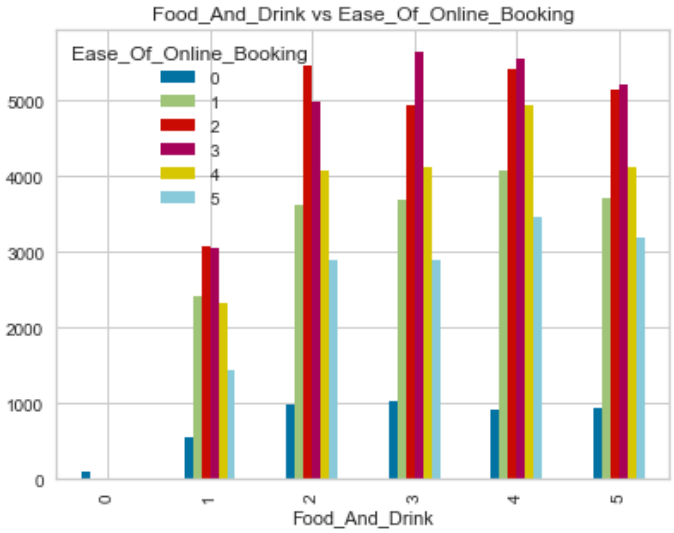

Gate_Location vs Seat_Comfort
Seat_Comfort
Gate_Location

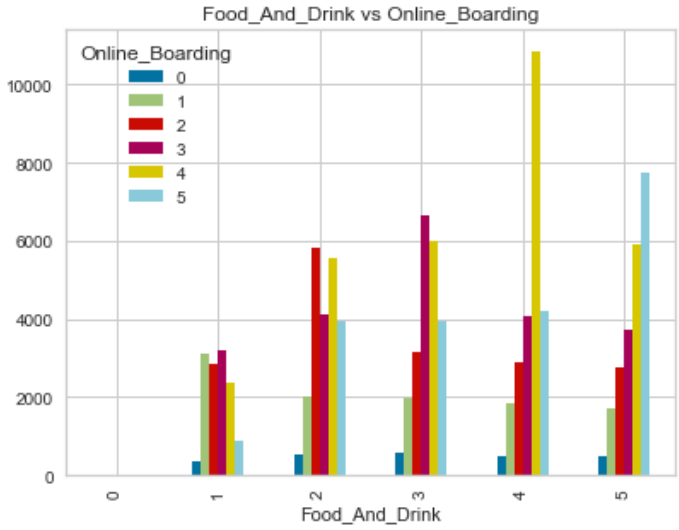

Gate_Location vs Inflight_Service
Inflight_Service
Gate_Location

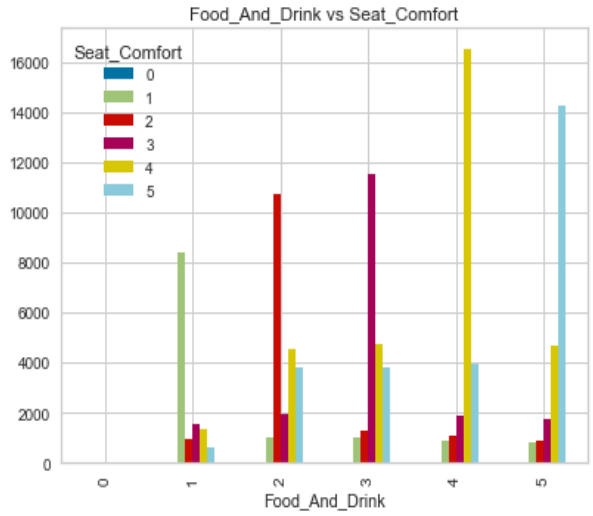

Food_And_Drink vs Customer_Type
Customer_Type
Loyal Customer
disloyal Customer
Food_And_Drink

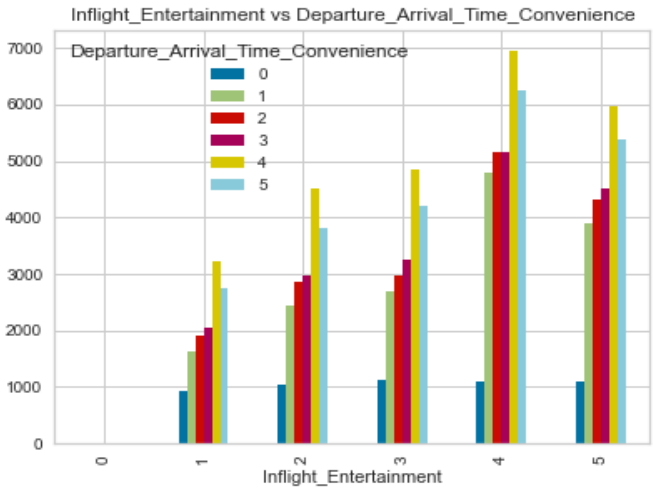

Food_And_Drink vs Type_Of_Travel
Type_Of_Travel
Business travel
Personal Travel
Food_And_Drink

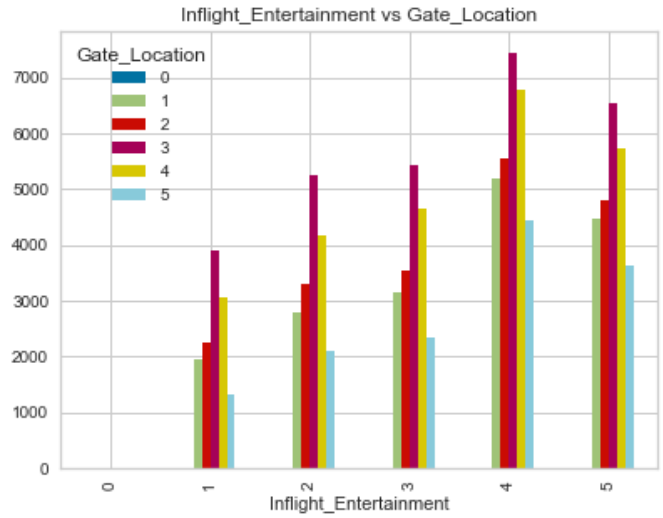

Food_And_Drink vs Inflight_Wifi_Service
Inflight_Wifi_Service
Food_And_Drink

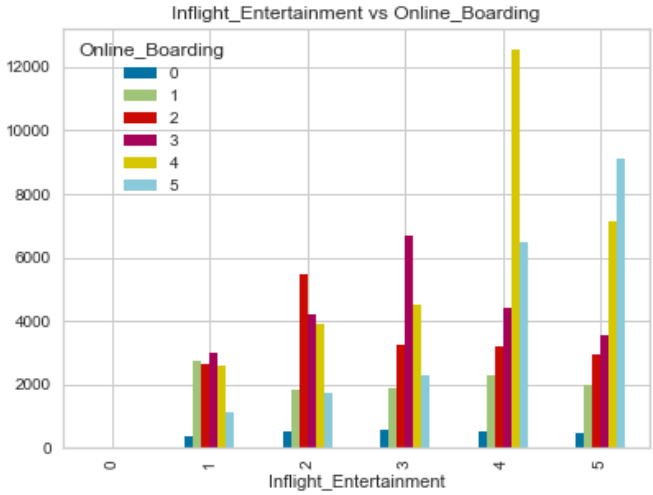

Food_And_Drink vs Ease_Of_Online_Booking
Ease_Of_Online_Booking
Food_And_Drink

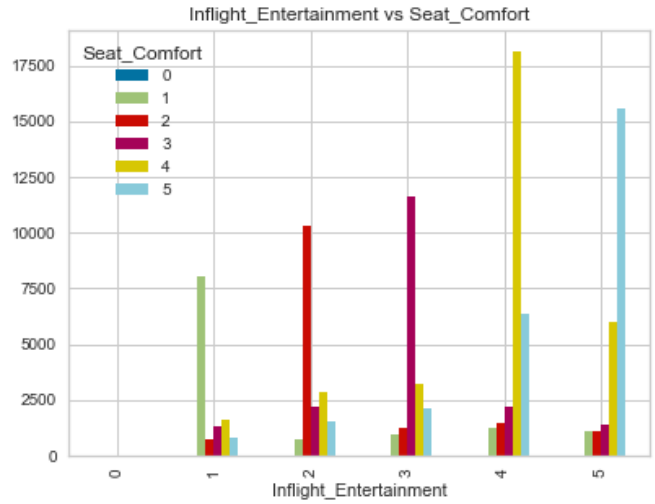

Food_And_Drink vs Online_Boarding
Online_Boarding
Food_And_Drink

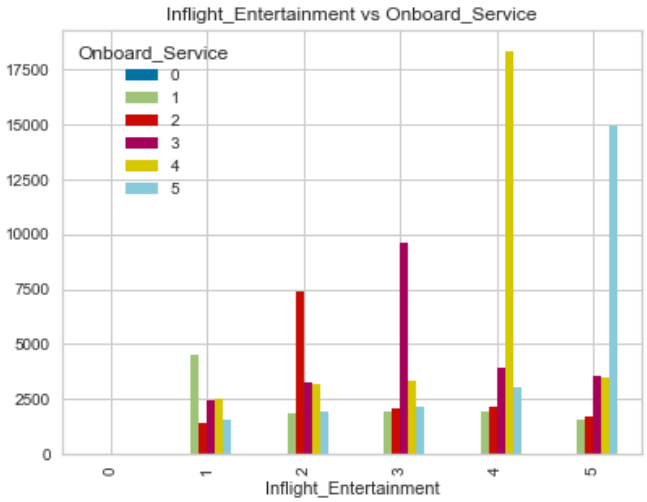

Food_And_Drink vs Seat_Comfort
Seat_Comfort
Food_And_Drink

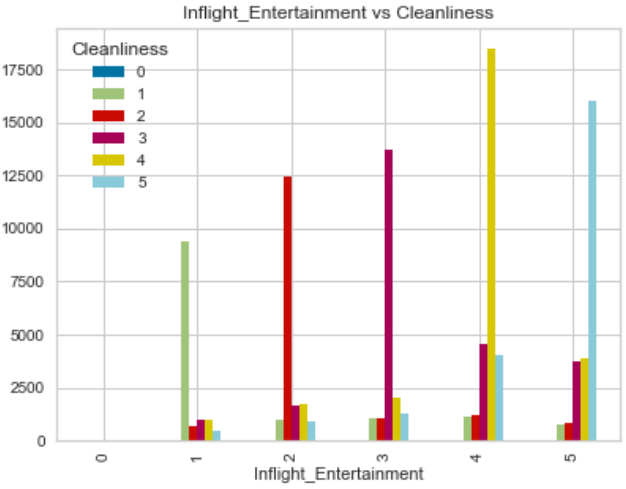

Inflight_Entertainment vs Departure_Arrival_Time_Convenience
Departure_Arrival_Time_Convenience
0
1
2
3
4
5
Inflight_Entertainment


Inflight_Entertainment vs Gate_Location
Gate_Location
0
1
2
3
4
5
Inflight_Entertainment

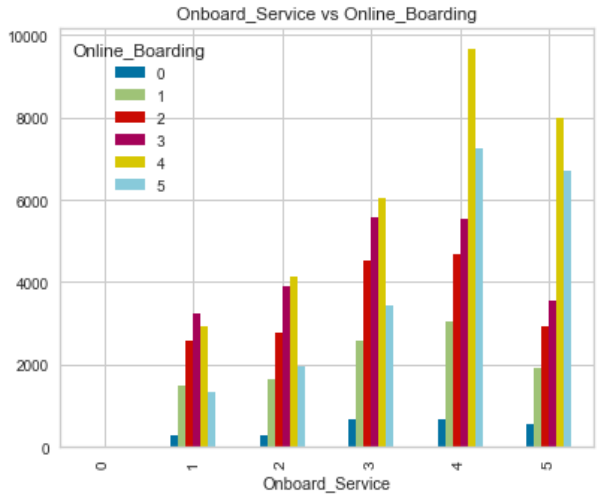

Inflight_Entertainment vs Online_Boarding
Online_Boarding
0
1
2
3
4
5
Inflight_Entertainment

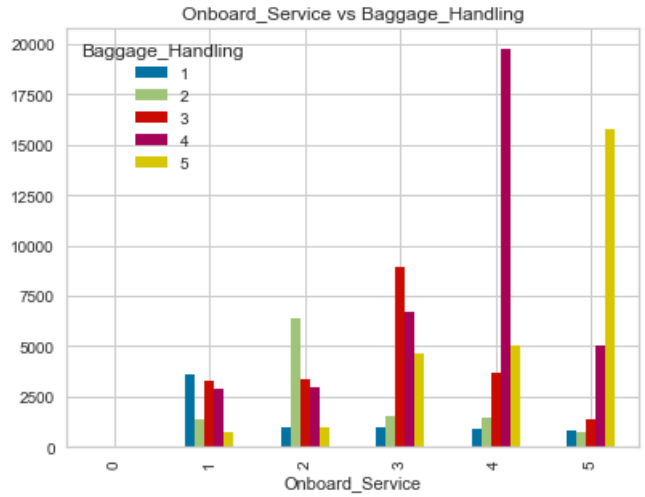

Inflight_Entertainment vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
Inflight_Entertainment

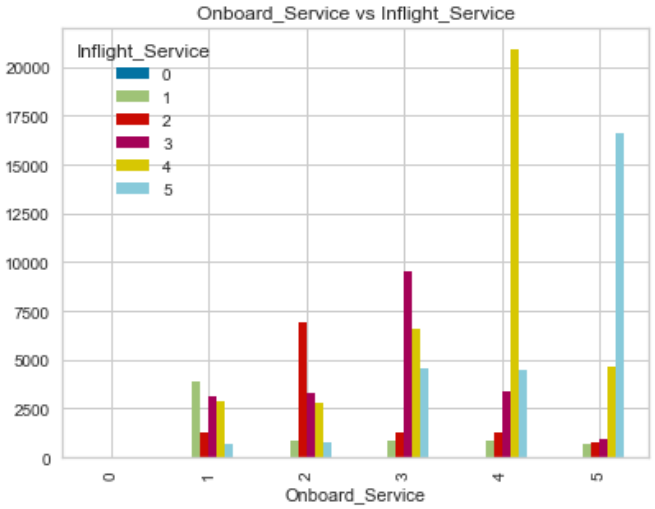

Inflight_Entertainment vs Onboard_Service
Onboard_Service
0
1
2
3
4
5
Inflight_Entertainment

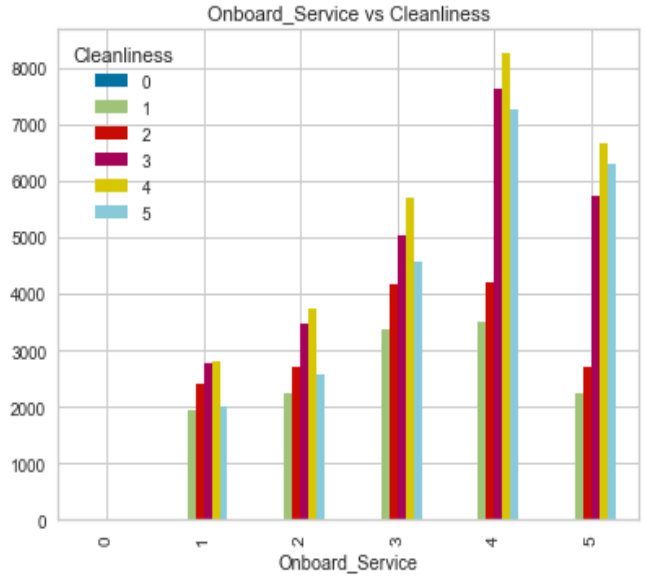

Inflight_Entertainment vs Cleanliness
Cleanliness
0
1
2
3
4
5
Inflight_Entertainment

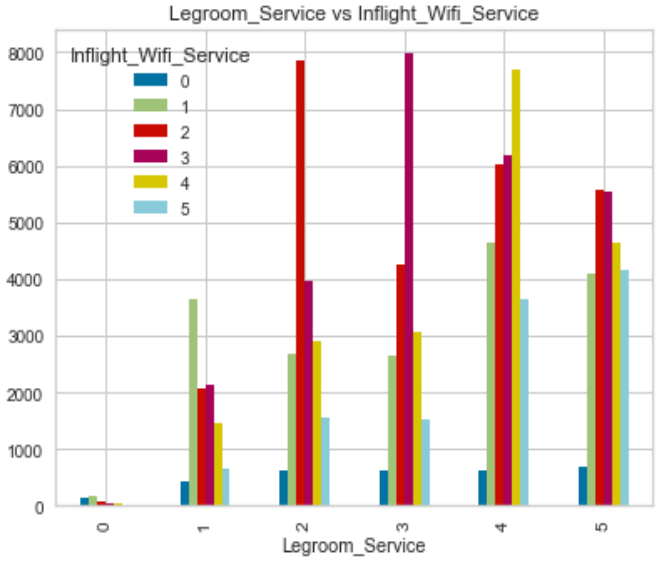

Onboard_Service vs Inflight_Wifi_Service
Inflight_Wifi_Service
0
1
2
3
4
5
Onboard_Service

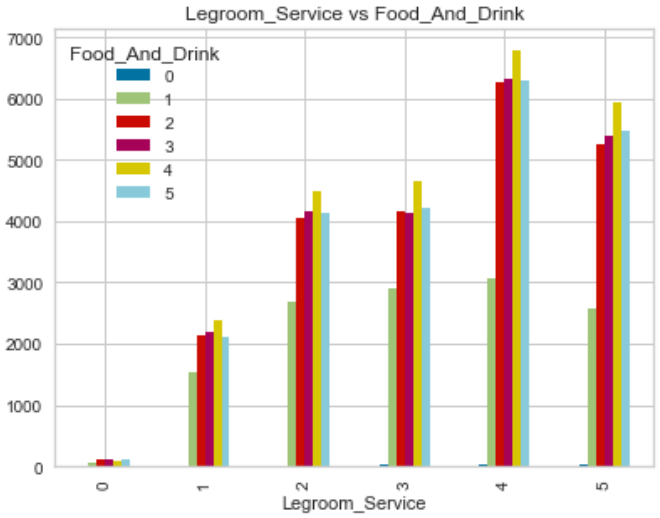

Onboard_Service vs Online_Boarding
Online_Boarding
0
1
2
3
4
5
Onboard_Service

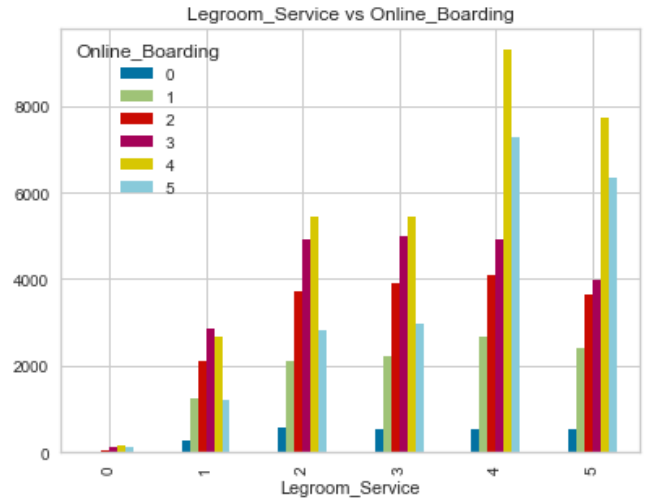

Onboard_Service vs Baggage_Handling
Baggage_Handling
1
2
3
4
5
Onboard_Service

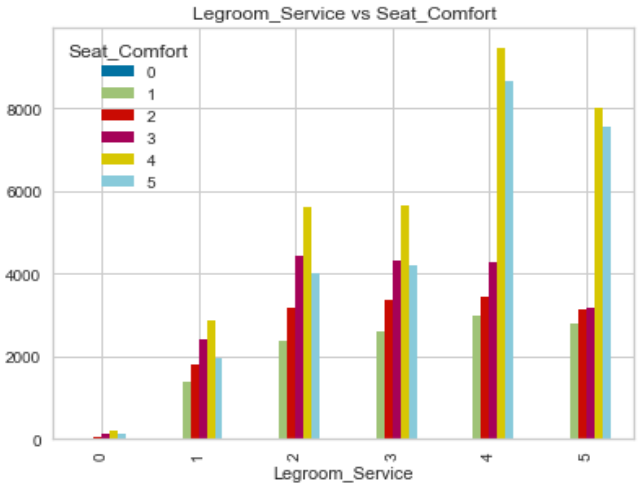

Onboard_Service vs Inflight_Service
Inflight_Service
0
1
2
3
4
5
Onboard_Service

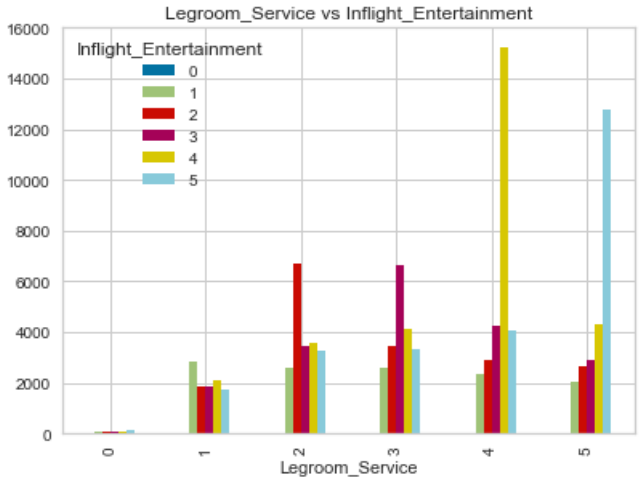

Onboard_Service vs Cleanliness
Cleanliness
0
1
2
3
4
5
Onboard_Service

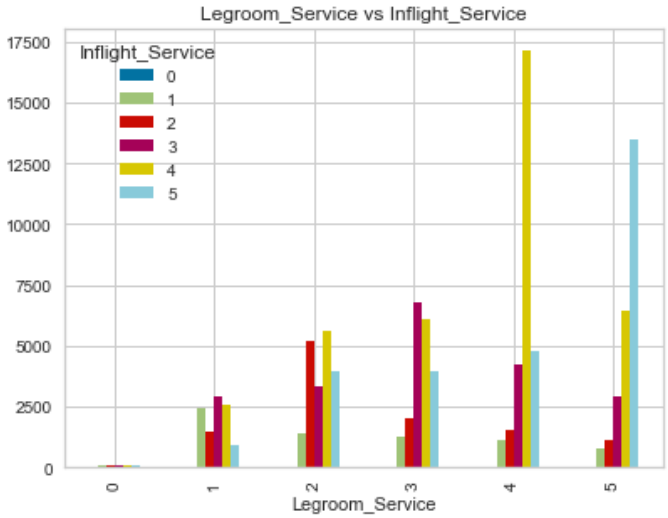

Legroom_Service vs Inflight_Wifi_Service
Inflight_Wifi_Service
0
1
2
3
4
5
Legroom_Service

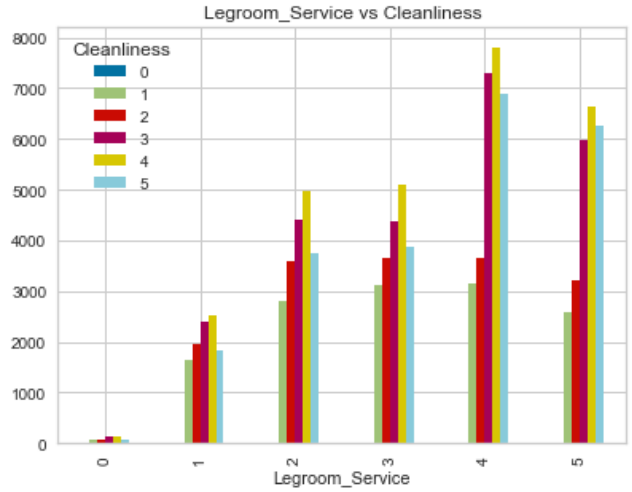

Legroom_Service vs Food_And_Drink
Food_And_Drink
0
1
2
3
4
5
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Legroom_Service


Legroom_Service vs Online_Boarding
Online_Boarding
0
1
2
3
4
5
8000
6000
4000
2000
0
0
1
2
3
4
5
Legroom_Service

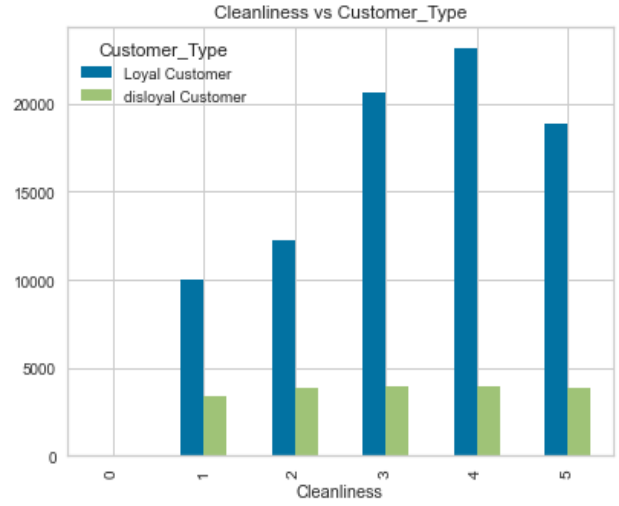

Legroom_Service vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
8000
6000
4000
2000
0
0
1
2
3
4
5
Legroom_Service

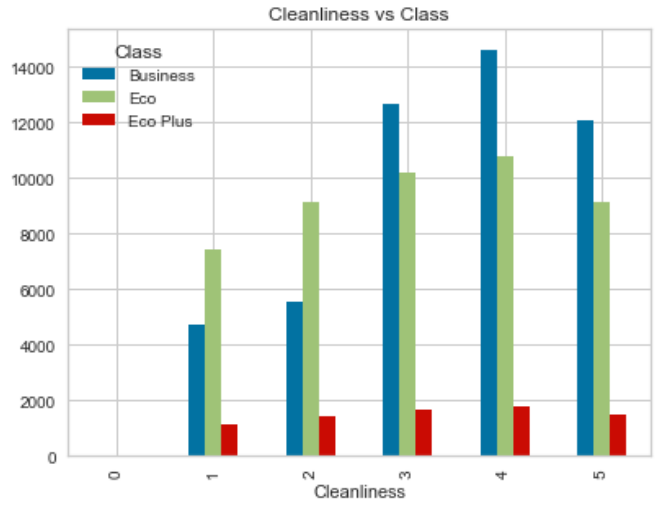

Legroom_Service vs Inflight_Entertainment
Inflight_Entertainment
0
1
2
3
4
5
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Legroom_Service

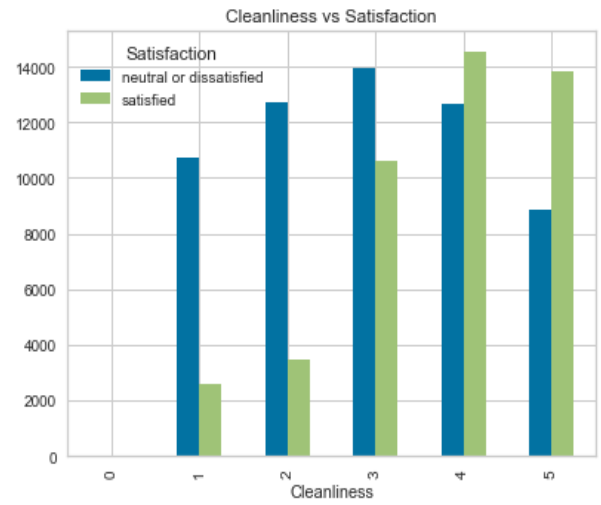

Legroom_Service vs Inflight_Service
Inflight_Service
0
1
2
3
4
5
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Legroom_Service

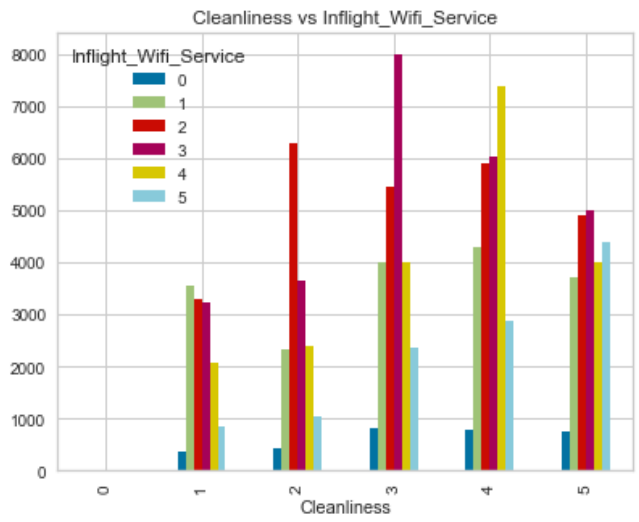

Legroom_Service vs Cleanliness
Cleanliness
0
1
2
3
4
5
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Legroom_Service

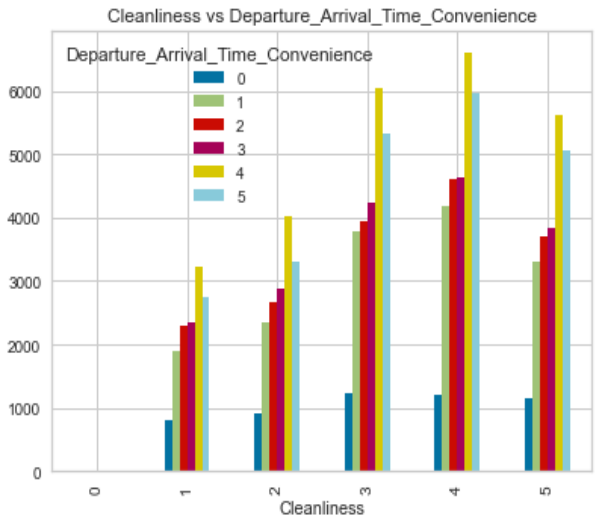

Cleanliness vs Customer_Type
Customer_Type
Loyal Customer
disloyal Customer
20000
15000
10000
5000
0
0
1
2
3
4
5
Cleanliness

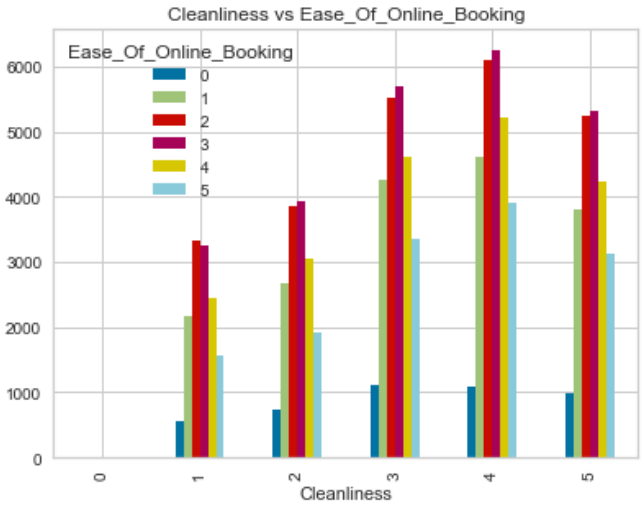

Cleanliness vs Customer_Type
Customer_Type
Loyal Customer
disloyal Customer
20000
15000
10000
5000
0
0
1
2
3
4
5
Cleanliness

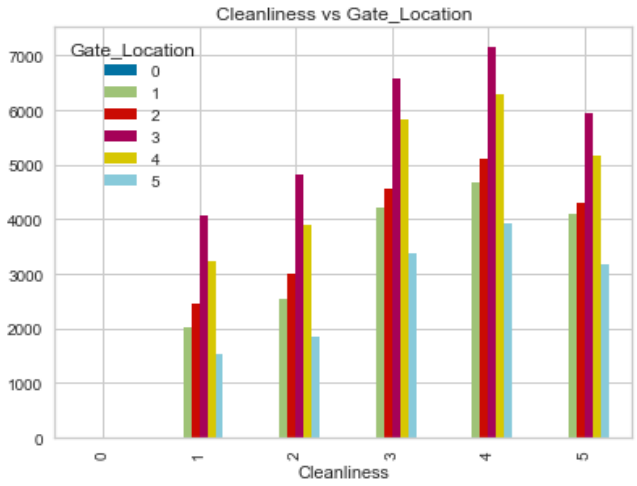

Cleanliness vs Class
Class
Business
Eco
Eco Plus
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Cleanliness

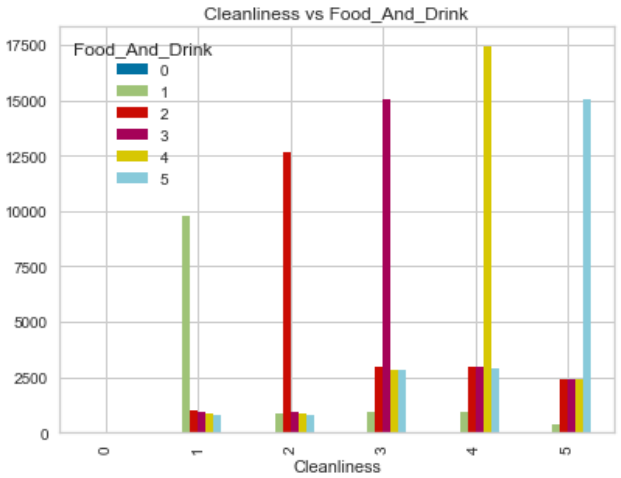

Cleanliness vs Satisfaction
Satisfaction
neutral or dissatisfied
satisfied
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Cleanliness

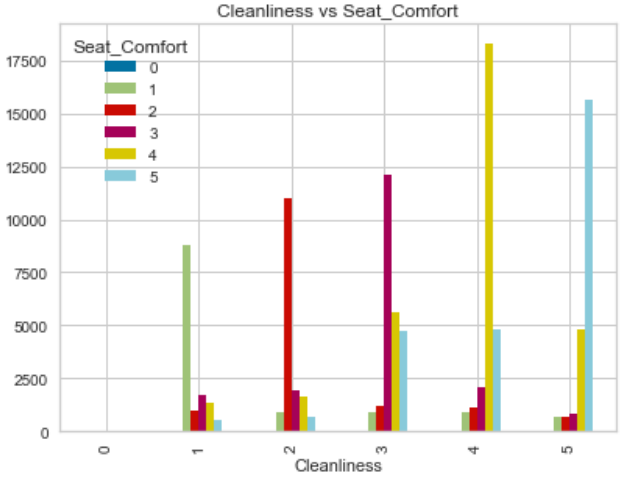

Cleanliness vs Inflight_Wifi_Service
Inflight_Wifi_Service
0
1
2
3
4
5
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Cleanliness

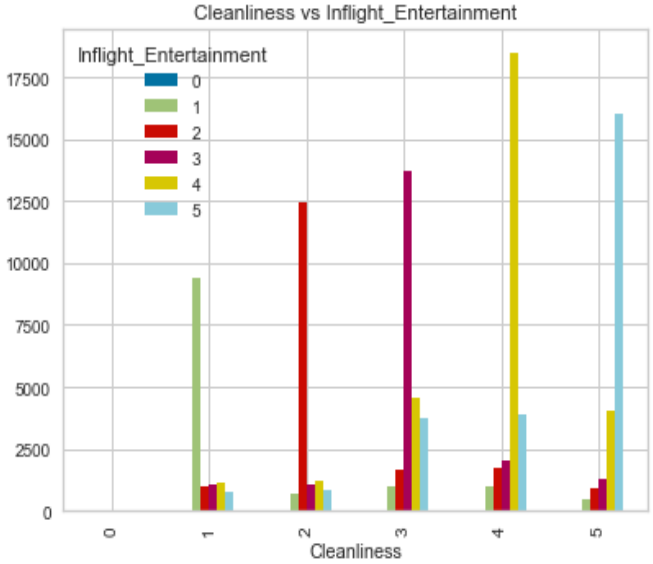

Cleanliness vs Departure_Arrival_Time_Convenience
Departure_Arrival_Time_Convenience
0
1
2
3
4
5
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Cleanliness

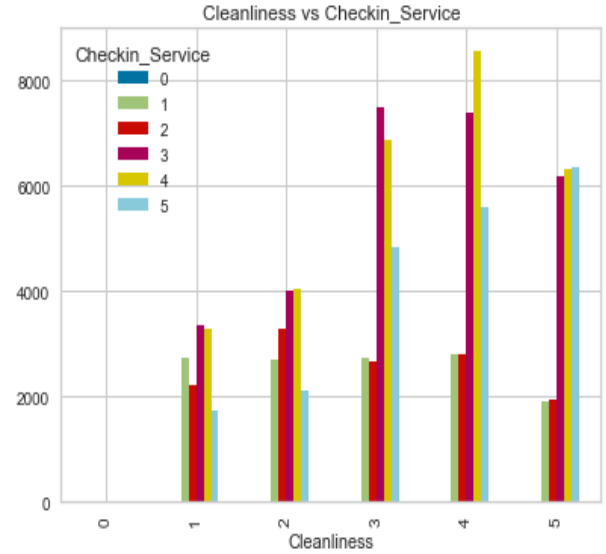

Cleanliness vs Ease_Of_Online_Booking
Ease_Of_Online_Booking
0
1
2
3
4
5
Cleanliness


Cleanliness vs Gate_Location
Gate_Location
0
1
2
3
4
5
Cleanliness


Cleanliness vs Food_And_Drink
Food_And_Drink
0
1
2
3
4
5
Cleanliness


Cleanliness vs Seat_Comfort
Seat_Comfort
0
1
2
3
4
5
Cleanliness


Cleanliness vs Inflight_Entertainment
Inflight_Entertainment
0
1
2
3
4
5
Cleanliness

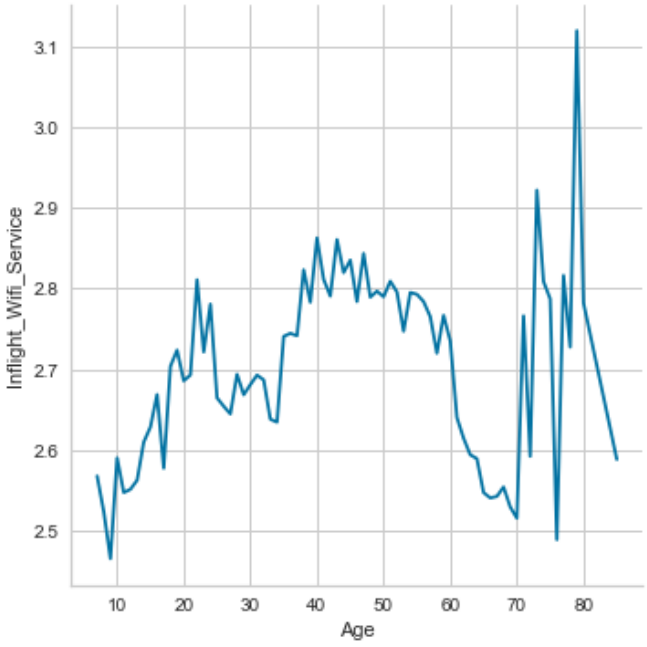

Cleanliness vs Checkin_Service
Checkin_Service
0
1
2
3
4
5
Cleanliness

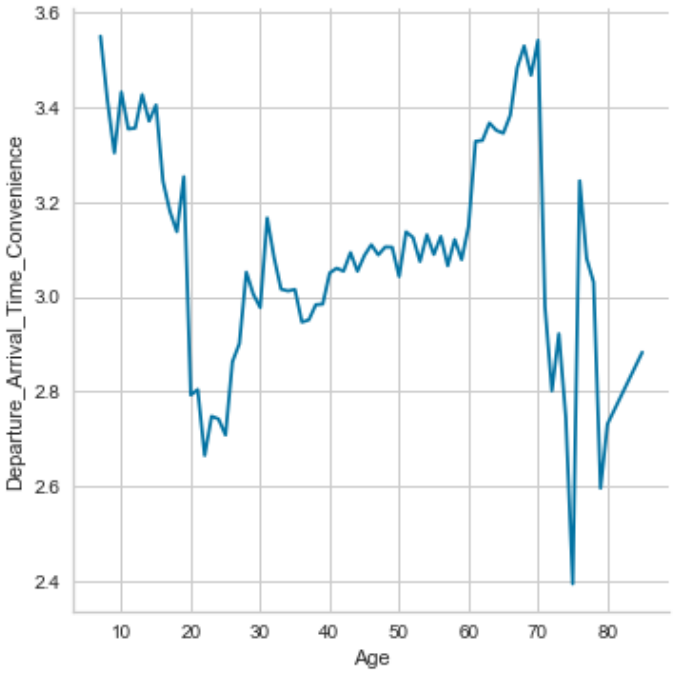
## Conclusions from Cross Tabulations

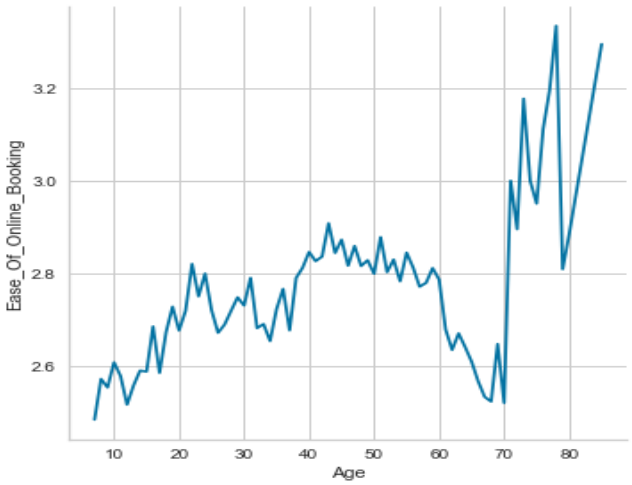
1. A slighly larger proportion of females are neutral or dis-satisfied as compared to males
2. Loyal customers have a greater proportion of satisfied fliers
3. Personal travel customer have greater proportion of dissatisfied fliers compared to business travellers
4. Business class has largest proportion of satisfied fliers while eco class has it the other way around
5. Satisfaction with inflight wifi implies satisfaction with flight overall
6. Satisfaction with departure arrival times does not result in certain satisfaction with overall flight
7. Greater ease of online booking implies more satisfaction
8. Only the maximum satisfaction with gate location results in a greater proportion of satisfied customers
9. Higher satisfaction with food and drink results in higher overall satisfaction
10. Online boarding satisfaction is clearly very important in determining overall satisfaction
11. Seat comfort is also an important indicator of overall satisfaction
12. Inflight entertainment satisfaction is important for overall satisfaction
13. Poor rating on onboard services implies less satisfaction

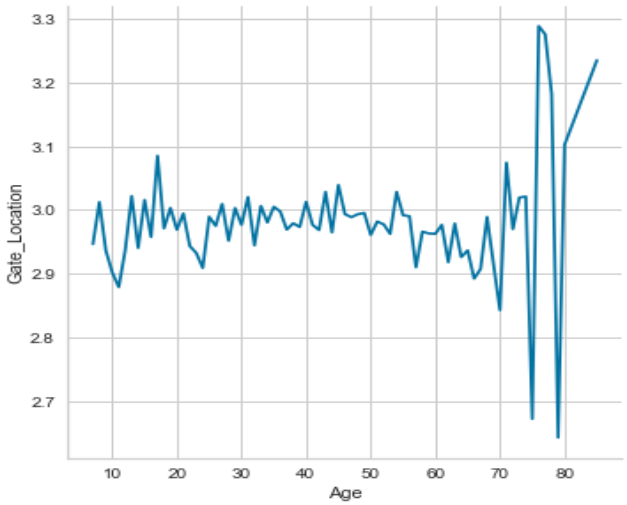
### 10.1.4 Rel Plots - Numeric vs Survery Features

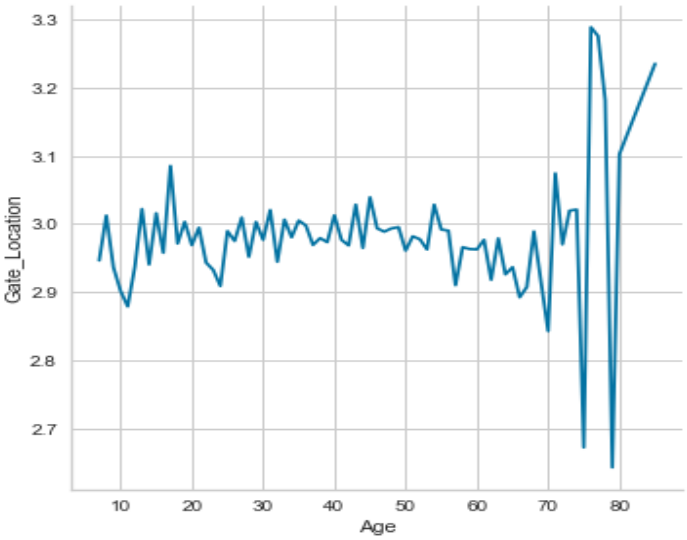
```
for col in airlineData_NumericColumns:
    for col1 in airlineData_NumericDiscreteColumns:
        meanFrame=airlineData.groupby(col)[col1].mean()
        meanFra me = pd.DataFrame(meanFrame)
        sns.relplot(data=meanFrame, x=col, y=col1,  kind='line')
        plt.show()
```

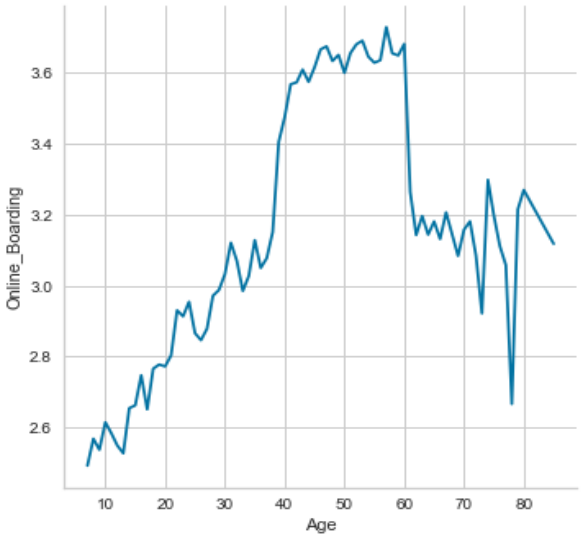


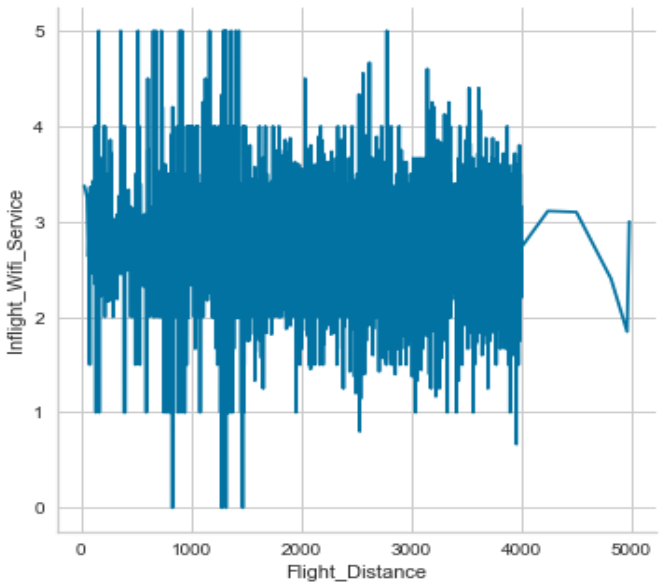



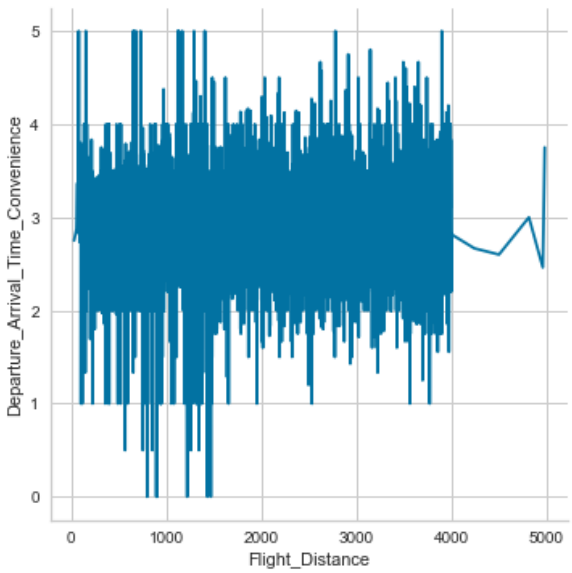

Ease_Of_Online_Booking
Age

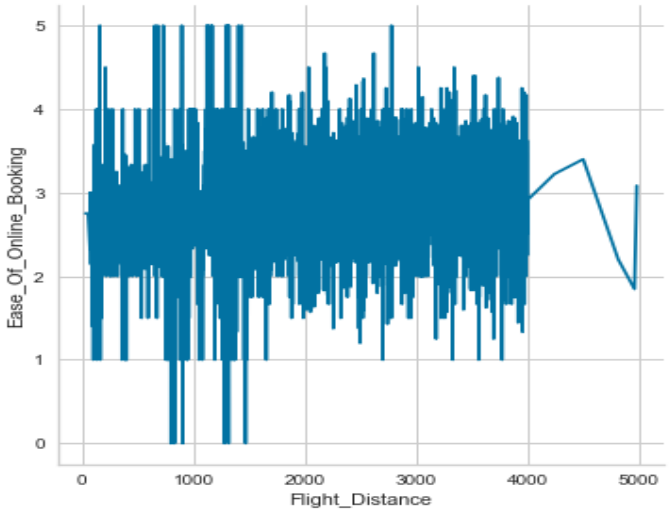

Gate_Location
Age

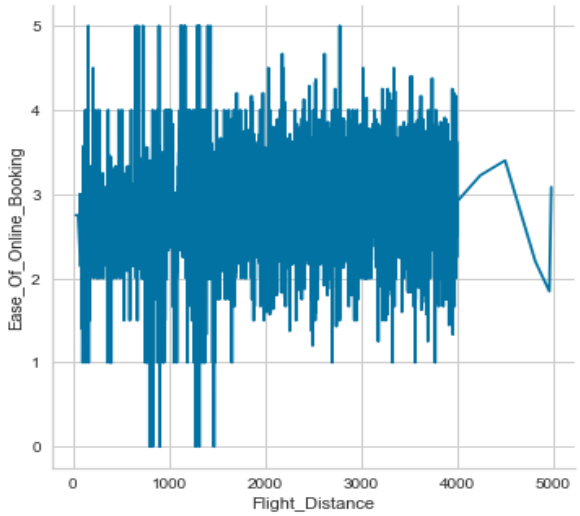

Gate_Location
Age

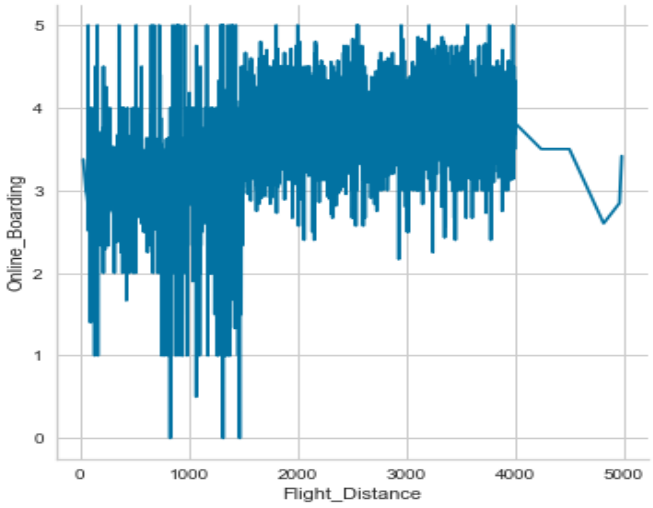

Online_Boarding
Age

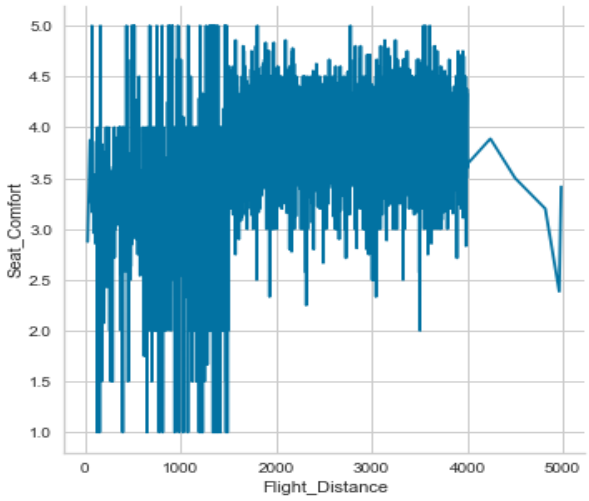

Inflight_Wifi_Service
Flight_Distance

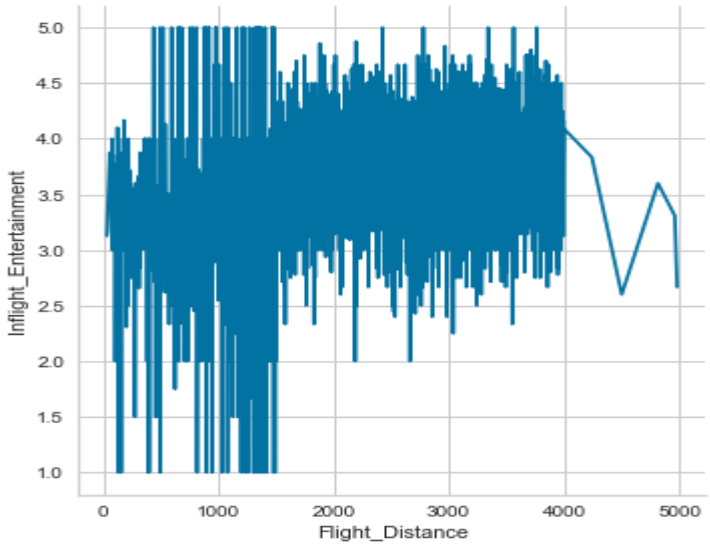

Departure_Arrival_Time_Convenience
Flight_Distance

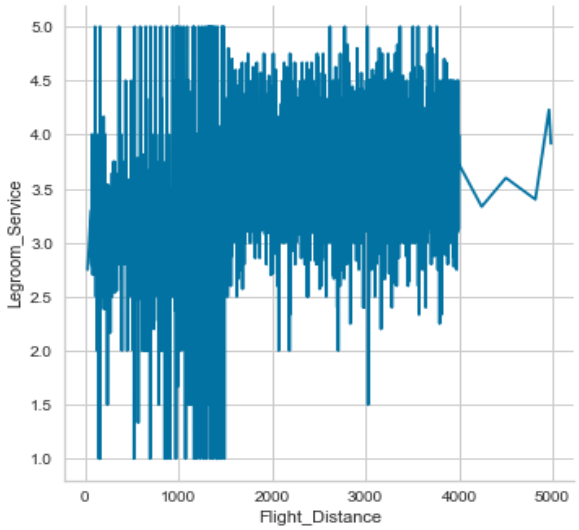

Ease_Of_Online_Booking
Flight_Distance

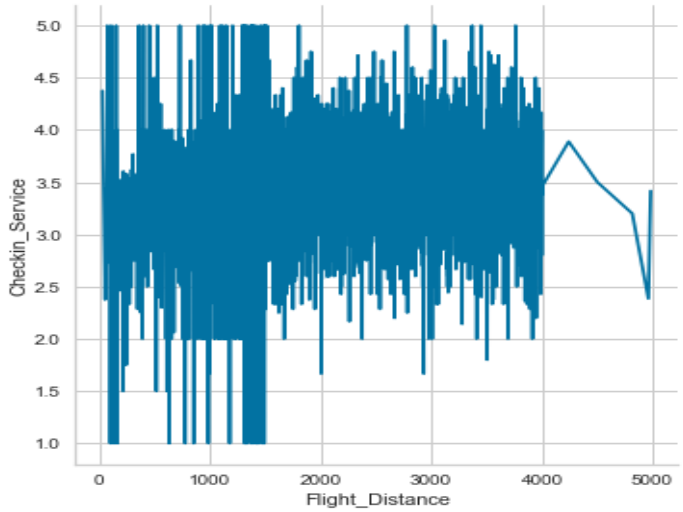

Ease_Of_Online_Booking
Flight_Distance

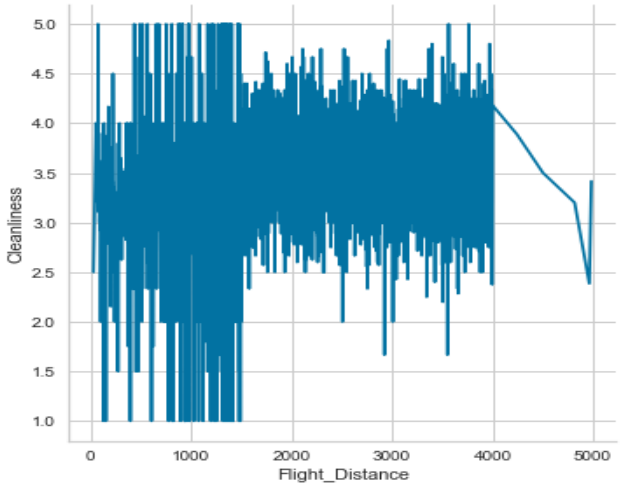

Online_Boarding
Flight_Distance

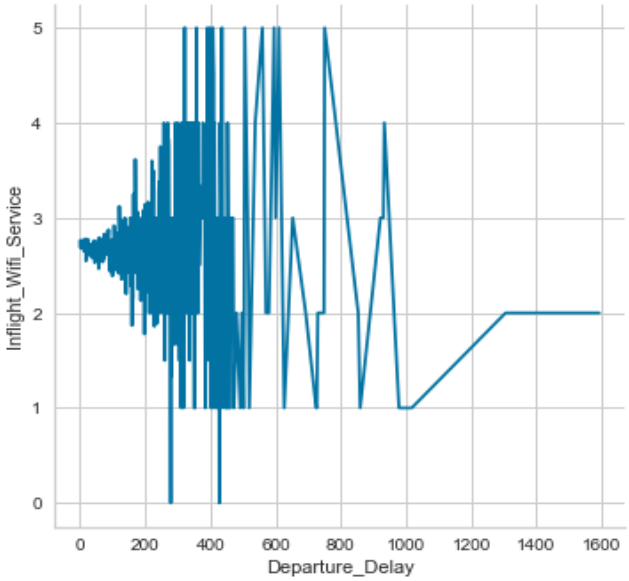

Seat_Comfort
Flight_Distance

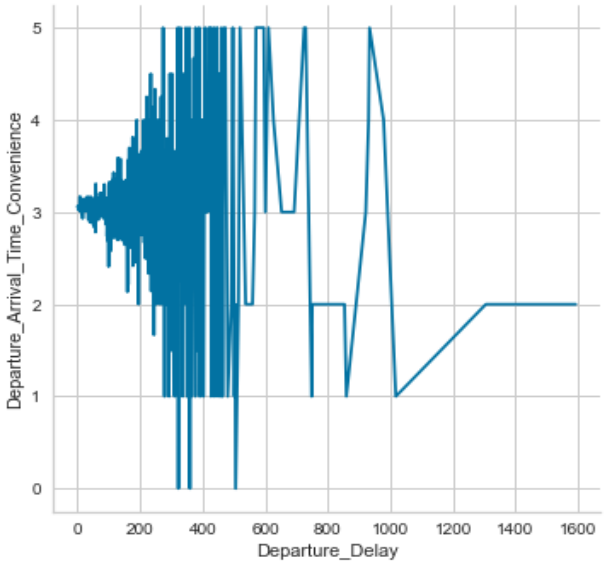

Inflight_Entertainment
Flight_Distance

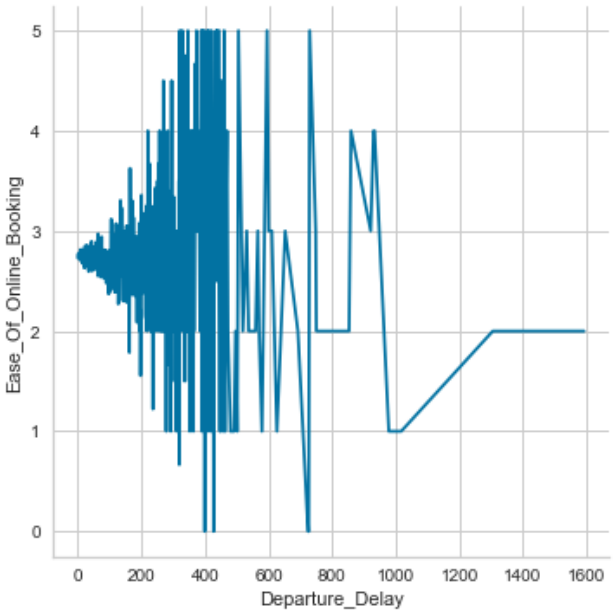

Legroom_Service
Flight_Distance

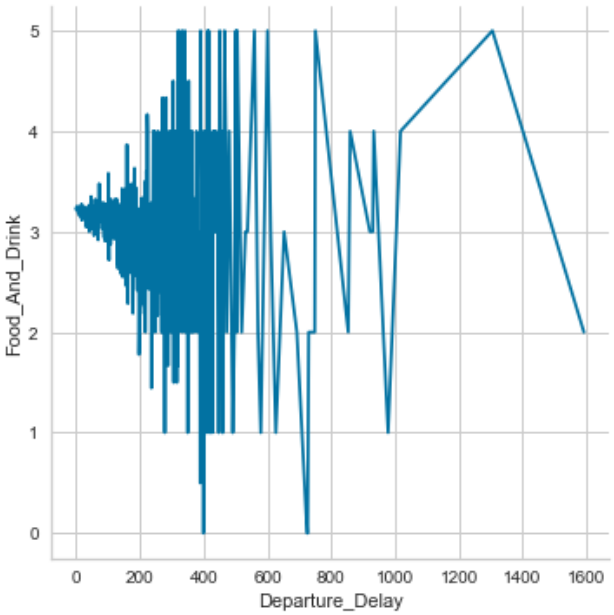

Checkin_Service
Flight_Distance

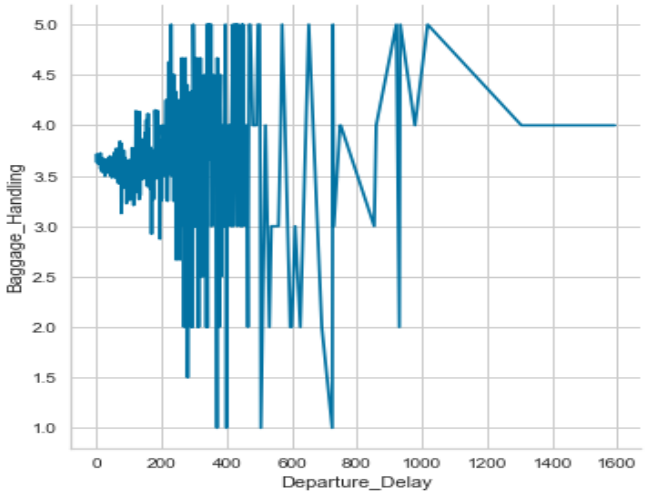

Cleanliness
Flight_Distance

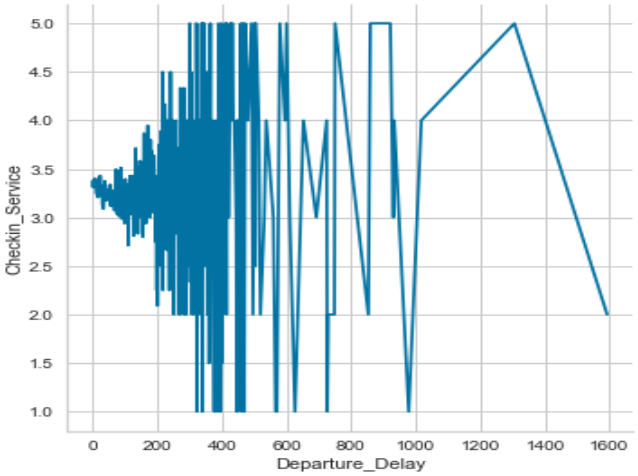

Inflight_Wifi_Service
Departure_Delay

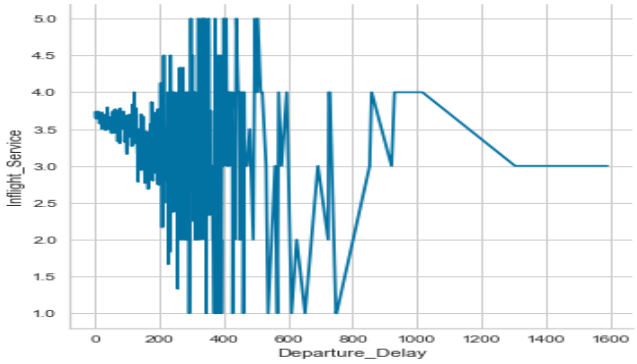

Departure_Arrival_Time_Convenience
Departure_Delay

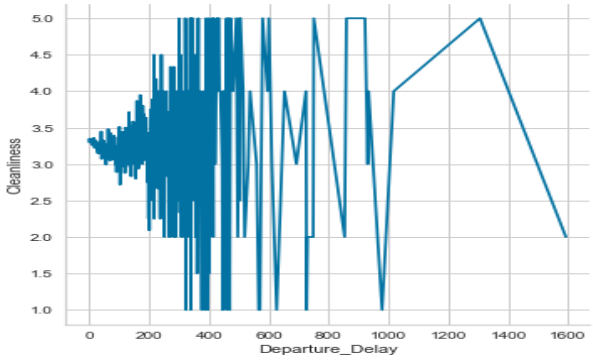

Ease_Of_Online_Booking
Departure_Delay


Food_And_Drink
Departure_Delay



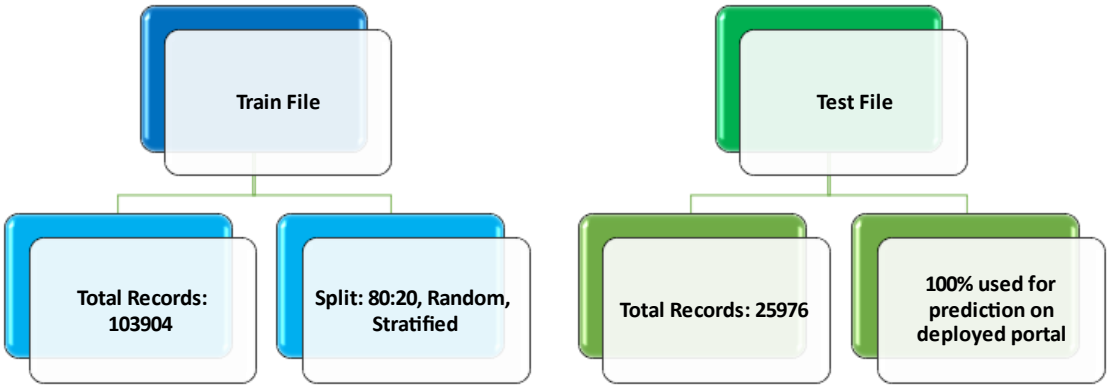



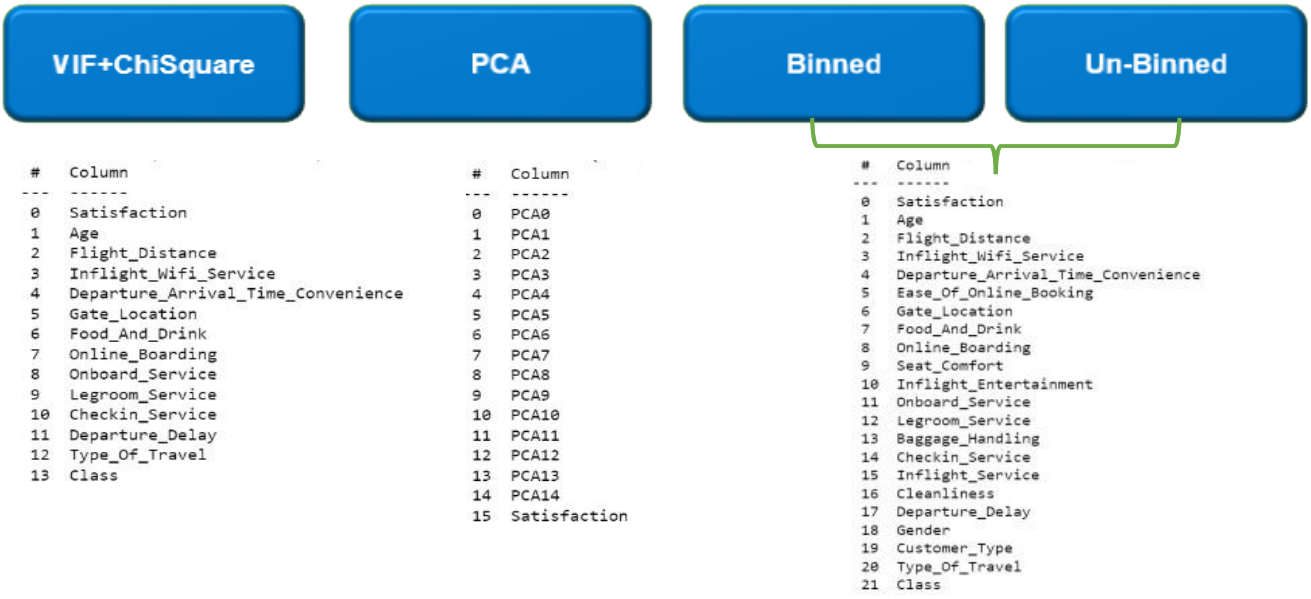

## CONCLUSIONS FROM REL PLOTS

1. Rating of inflight services with respect to age is jagged - it improves as age increases upto 23, then shows a dip in the range of 25 to 35.
2. Departure-Arrival timing convenience is rated low by those in the mid-20s and lowest by those in the mid-70s.
3. Ease of online booking is rated higher by fliers over 70 years of age
4. Gate location rating is average for most of the fliers, except it tends to vary widely in the 70+ age group
5. Online boarding is rated higher by fliers aged 40-60 years.
6. Seat comfort is also rated higher by fliers aged 40 to 60 years as is in-flight entertainment.
7. Onboard services and leg-room are rated highest by fliers in the range 40 to 60 years while lowest by those above 65.
8. Baggage handling, check-in, in-flight service and cleanliness satisfaction is lowest among senior citizens in the age range more than 65 years.
9. Satisfaction with most parameters dips with longer flight distances, is varible for short duration flights and is average-high for moderate flying distances.
10. Departure delays result in dis-satisfaction with most other parameters - can be interpreted as departure delays skews the overall perception of performance on all parameters towards the negative side

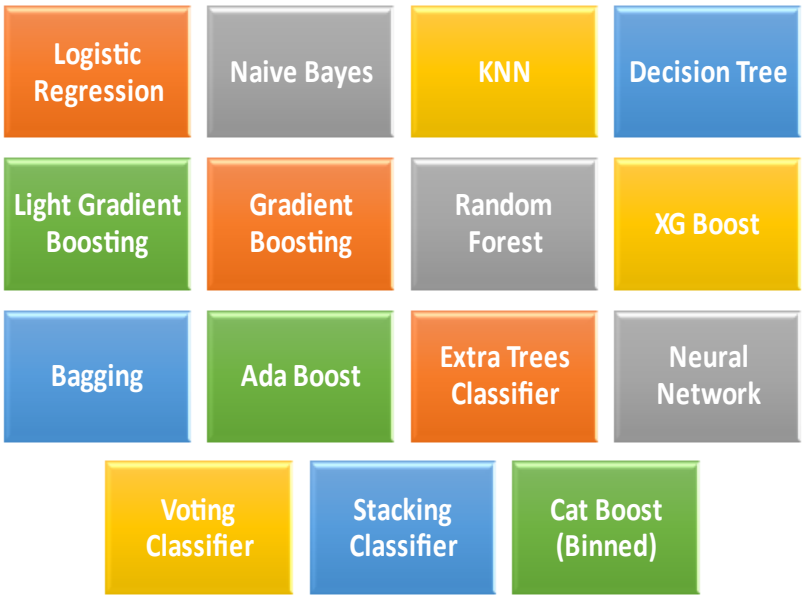
## 11 Train Test Split




## 12 Data Sets for Modelling



## 13 Models Implemented



## 14 Machine Learning Modelling & Techniques Applied

### 14.1 Standard Classifier models

The below classifier models were created during the modelling phase.

All the models were trained using the below techniques:

- Using default parameters for the models
- Hyperparameter tuning with RandomizedSearchCV

All the 21 models were trained with different combinations of hyperparameters
Example of hyperparameters:

**Random Forest Classifier**: Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output

```
rf = RandomForestClassifier(
    n_estimators=100, max_depth=5, max_features='auto', max_leaf_nodes=50, random_state=SEED,
    min_samples_split=10, bootstrap='True', criterion='gini', class_weight='balanced')
```

```
parameters.update({"Random Forest": {"class_weight": [None, "balanced"],
                                     "max_features": ["auto", "sqrt", "log2"],
                                     "max_depth": [3, 4, 5, 6, 7, 8],
                                     "min_samples_split": [0.005, 0.01, 0.05, 0.10],
                                     "min_samples_leaf": [0.005, 0.01, 0.05, 0.10],
                                     "criterion": ["gini", "entropy"]
                                     }})
```

**SVC**: Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.The advantages of support vector machines are, Effective in high dimensional spaces.Still effective in cases where number of dimensions is greater than the number of samples.Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

```
svc = SVC(C=100, probability=True)
```

```
parameters.update({"Support Vector Machine": {"kernel": ["linear", "rbf", "poly"],
                                              "gamma": ["auto"], "C": [0.1, 0.5, 1, 5, 10, 50, 100],
                                              "degree": [1, 2, 3, 4, 5, 6]
                                              }})
```

**KNN**: The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.
For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and and when it gets new data, then it classifies that data into a category that is much similar to the new data.

```
knn = KNeighborsClassifier(n_neighbors=3, leaf_size=30, p=2, weights='uniform',
                           algorithm='auto', n_jobs=-1, metric='minkowski')
```

```
parameters.update({"KNN": {"n_neighbors": [3, 5, 11, 19],
                           "p": [1, 2, 3, 4, 5],
                           "leaf_size": [5, 10, 15, 20, 25, 30, 35, 40, 45, 50],
                           "n_jobs": [-1],
                           "weights": ['uniform', 'distance'],
                           "metric": ['euclidean', 'manhattan']
                           }})
```

**Logistic regression:** is a classification algorithm. It is used to predict a binary outcome based on a set of independent variables which can be continuous, Discrete, ordinal or Discrete, nominal. Logistic Regression assumes
The dependent variable is binary There should be no, or very little, multicollinearity between the predictor variables The independent variables should be linearly related to the log odds Logistic regression requires fairly large sample sizes.There are three types of logistic regression. 1. Binary logistic regression, 2. Multinomial logistic regression, 3. Ordinal logistic regression

```
lr = LogisticRegression(
    solver='liblinear', C=1.0, random_state=SEED)
```

```
parameters.update({"Logistic Regression": {"penalty": ['l1', 'l2'],
                                           "solver": ['liblinear', 'lbfgs'],

                                           "n_jobs": [-1]
                                           }})
```

**Gradient boosting:** is one of the most powerful techniques for building predictive models, and it is called a Generalization of AdaBoost. The main objective of Gradient Boost is to minimize the loss function by adding weak learners using a gradient descent optimization algorithm. The generalization allowed arbitrary differentiable loss functions to be used, expanding the technique beyond binary classification problems to support regression, multi-class classification and more.
Gradient Boost has three main components.
Loss Function: The role of the loss function is to estimate how best is the model in making predictions with the given data. This could vary depending on the type of the problem.
Weak Learner: Weak learner is one that classifies the data so poorly when compared to random guessing. The weak learners are mostly decision trees, but other models can be used in GBM.
Additive Model: It is an iterative and sequential process in adding the decision trees one step at a time. Each iteration should reduce the value of loss function. A fixed number of trees are added, or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

```
gb = GradientBoostingClassifier(
    n_estimators=100, random_state=SEED)
```

```
parameters.update({"Gradient Boosting": {"learning_rate": [0.15, 0.1, 0.05, 0.01, 0.005, 0.001],
                           "max_depth": [2, 3, 4, 5, 6],
                           "min_samples_split": [0.005, 0.01, 0.05, 0.10],
                           "min_samples_leaf": [0.005, 0.01, 0.05, 0.10],
                           "max_features": ["auto", "sqrt", "log2"],
                           "subsample": [0.8, 0.9, 1]
                           }})
```

**Extreme Gradient Boosting**: XGBoost is an implementation of Gradient Boosted decision trees. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Weight of variables predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

```
xgb = XGBClassifier(max_depth=3, learning_rate=0.1, n_estimators=150,
      booster='gbtree', n_jobs=1, nthread=None, gamma=0, min_child_weight=1, max_delta_step=0, verbosity = 0,
      subsample=1, colsample_bytree=1, colsample_bylevel=1, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
      base_score=0.5, random_state=SEED, seed=None, use_label_encoder=False, missing=1, silent=True)
```

```
parameters.update({"Gradient Boosting": {"learning_rate": [0.15, 0.1, 0.05, 0.01, 0.005, 0.001],
                           "max_depth": [2, 3, 4, 5, 6],
                           "min_samples_split": [0.005, 0.01, 0.05, 0.10],
                           "min_samples_leaf": [0.005, 0.01, 0.05, 0.10],
                           "max_features": ["auto", "sqrt", "log2"],
                           "subsample": [0.8, 0.9, 1]
                           }})
```

**Linear Discriminant Analysis :** A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule. The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix. The fitted model can also be used to reduce the dimensionality of the input by projecting it to the most discriminative directions, using the transform method.

```
lda = LinearDiscriminantAnalysis(solver='svd', tol=0.0001)
```

```
parameters.update({"lda": {"solver": ["svd"],
                           }})
```

**Quadratic Discriminant Analysis**: A classifier with a quadratic decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule. The model fits a Gaussian density to each class.

```
qda = QuadraticDiscriminantAnalysis(
    reg_param=0.0, store_covariance=False, tol=0.0001)
```

```
parameters.update({"qda": {"reg_param": [0.01*ii for ii in range(0, 101)],
                           }})
```

**Ada Boost**: AdaBoost is an ensemble method that trains and deploys trees in series. AdaBoost implements boosting, wherein a set of weak classifiers is connected in series such that each weak classifier tries to improve the classification of samples that were misclassified by the previous weak classifier. The decision trees used in boosting methods are called "stump" because each decision tree tends to be shallow models that do not overfit but can be biased. An individual tree is trained to pay specific attention to the weakness of only the previous tree. The classification accuracy increases when more weak classifiers are added in series to the model; however, this may lead to severe overfitting and drop in generalization capability. AdaBoost is suited for imbalanced datasets but under performs in the presence of noise. Steps are as follows

Step 1 A weak classifier (e.g. a decision stump) is made on top of the training data based on the weighted samples. Here, the weights of each sample indicate how important it is to be correctly classified. Initially, for the first stump, we give all the samples equal weights.

Step 2 We create a decision stump for each variable and see how well each stump classifies samples to their target classes.

Step 3 More weight is assigned to the incorrectly classified samples so that they're classified correctly in the next decision stump. Weight is also assigned to each classifier based on the accuracy of the classifier, which means high accuracy = high weight!

Step 4 Reiterate from Step 2 until all the data points have been correctly classified, or the maximum iteration level has been reached.

```
ada = AdaBoostClassifier(
    n_estimators=50, learning_rate=1.0, algorithm='SAMME.R')
```

```
parameters.update({"ada boost": {"base_estimator": [DecisionTreeClassifier(max_depth=ii) for ii in range(1, 6)],
                                 "learning_rate": [0.001, 0.01, 0.05, 0.1, 0.25, 0.50, 0.75, 1.0]
                                 }})
```

A **Bagging classifie**r is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

```
bagging = BaggingClassifier(base_estimator=None, n_estimators=10, max_samples=1.0,
                            max_features=1.0, bootstrap=True, bootstrap_features=False,
                            oob_score=False, warm_start=False, n_jobs=None)
```

```
parameters.update({"bagging": {"base_estimator": [DecisionTreeClassifier(max_depth=ii) for ii in range(1, 6)],
                               "n_estimators": [200],
                               "max_features": [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
                               "n_jobs": [-1]
                               }})
```

**Extra Trees Classifier**

Extremely Randomized Trees Classifier(Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output it's classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest. Each Decision Tree in the Extra Trees Forest is constructed from the original training

sample. Then, at each test node, Each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees.

```
etc = ExtraTreesClassifier(n_estimators=100, criterion='gini', max_depth=None,
                           min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
                           max_features='auto', max_leaf_nodes=None,
                           min_impurity_decrease=0.0,
                           bootstrap=False, oob_score=False, n_jobs=-1,)
```

```
parameters.update({"Extra Trees Classifier": {"class_weight": [None, "balanced"],
                                    "max_features": ["auto", "sqrt", "log2"],
                                    "max_depth": [3, 4, 5, 6, 7, 8],
                                    "min_samples_split": [0.005, 0.01, 0.05, 0.10],
                                    "min_samples_leaf": [0.005, 0.01, 0.05, 0.10],
                                    "criterion": ["gini", "entropy"],
                                    "n_jobs": [-1]
                                    }})
```

**Classifier using Ridge regression:**

This classifier first converts the target values into {-1, 1} and then treats the problem as a regression task (multi-output regression in the multiclass case).

```
ridge = RidgeClassifier(alpha=1.0, fit_intercept=True, normalize=False, copy_X=True,
                        max_iter=None, tol=0.001, class_weight=None, solver='auto')
```

```
parameters.update({"ridge": {"alpha": [1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 0.25, 0.50, 0.75, 1.0]
                             }})
```

**Linear classifiers (SVM, logistic regression, etc.) with SGD training**.

This estimator implements regularized linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). SGD allows minibatch (online/out-of-core) learning via the partial_fit method. For best results using the default learning rate schedule, the data should have zero mean and unit variance.

This implementation works with data represented as dense or sparse arrays of floating point values for the features. The model it fits can be controlled with the loss parameter; by default, it fits a linear support vector machine (SVM).

```
sgd = SGDClassifier(alpha=1.0, penalty='l2')
```

```
parameters.update({"sgd": {"alpha": [1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 0.25, 0.50, 0.75, 1.0],
                           "penalty": ["l1", "l2"],
                           "n_jobs": [-1]
                           }})
```

**Decision Trees**

A Decision Tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question;

edges represent the answers to the question, and the leaves represent the actual output or class label. They are used in non-linear decision making with a simple linear decision surface. In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:
Information Gain - When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy. Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.
Gini Index - Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower Gini index should be preferred.

```
dtc = DecisionTreeClassifier(criterion='gini', splitter='best')
```

```
parameters.update({"Decision Tree": {"criterion": ["gini", "entropy"],
                            "splitter": ["best", "random"],
                            "class_weight": [None, "balanced"],
                            "max_features": ["auto", "sqrt", "log2"],
                            "max_depth": [1, 2, 3, 4, 5, 6, 7, 8],
                            "min_samples_split": [0.005, 0.01, 0.05, 0.10],
                            "min_samples_leaf": [0.005, 0.01, 0.05, 0.10],
                            }})
```

**Light Gradient Boosting:** LightGBM is a gradient boosting framework based on decision trees to increases the efficiency of the model and reduces memory usage. It uses two novel techniques: Gradient-based One Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) which fulfills the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks. The two techniques of GOSS and EFB form the characteristics of LightGBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks
Gradient-based One Side Sampling Technique for LightGBM:
Different data instances have varied roles in the computation of information gain. The instances with larger gradients(i.e., under-trained instances) will contribute more to the information gain. GOSS keeps those instances with large gradients (e.g., larger than a predefined threshold, or among the top percentiles), and only randomly drop those instances with small gradients to retain the accuracy of information gain estimation. This treatment can lead to a more accurate gain estimation than uniformly random sampling, with the same target sampling rate, especially when the value of information gain has a large range. .
Exclusive Feature Bundling Technique for LightGBM:
High-dimensional data are usually very sparse which provides us a possibility of designing a nearly lossless approach to reduce the number of features. Specifically, in a sparse feature space, many features are mutually exclusive, i.e., they never take nonzero values simultaneously. The exclusive features can be safely bundled into a single feature (called an Exclusive Feature Bundle). Hence, the complexity of histogram building changes from O(#data × #feature) to O(#data × #bundle), while #bundle <<# feature . Hence, the speed for training framework is improved without hurting accuracy.

```
LGB = LGBMClassifier(boosting_type='gbdt', num_leaves=31, max_depth=- 1,
                     learning_rate=0.1, n_estimators=100, subsample_for_bin=200000,
                     objective=None, class_weight=None, min_split_gain=0.0,
                     min_child_weight=0.001, min_child_samples=20)
```

```
parameters.update({"LightGradientBoosting": {"max_depth": range(3, 10, 2), "min_child_weight": range(1, 6, 2),
                                             "subsample": [i/10.0 for i in range(6, 10)],
                                             "colsample_bytree": [i/10.0 for i in range(6, 10)],
                                             "reg_alpha": [1e-5, 1e-2, 0.1, 1, 100],
                                             "learning_rate": [0.001, 0.01, 0.1, 0.2, 0.3]
                                             }})
```

**Class MLPClassifier:** implements a multi-layer perceptron (MLP) algorithm that trains using Backpropagation. MLP trains on two arrays: array X of size (n_samples, n_features), which holds the training samples represented as floating point feature vectors; and array y of size (n_samples,), which holds the target values (class labels) for the training samples:

```
nn = MLPClassifier((80, 10), early_stopping=False, random_state=SEED)
```

```
parameters.update({"Neural Network": {"hidden_layer_sizes": [(5), (10), (5,5), (10,10), (5,5,5), (10,10,10)],
                                      "activation": ["identity", "logistic", "tanh", "relu"],
                                      "batch_size" : [16,32],
                                      "solver":['sgd','adam','lbfgs'],
                                      "learning_rate": ["constant", "invscaling", "adaptive"],
                                      "max_iter": [100, 200, 300, 500, 1000, 2000],
                                      "alpha": list(10.0 ** -np.arange(1, 10)),
                                      }})
```

**Naive Bayes** methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x1 through xn, :

$P(y|x1,\ldots,xn)=P(y)P(x1,\ldots,xn|y)P(x1,\ldots,xn)$

Using the naive conditional independence assumption that

$P(xi|y,x1,\ldots,xi-1,xi+1,\ldots,xn)=P(xi|y)$,

for all i, this relationship is simplified to

$P(y|x1,\ldots,xn)=P(y)\prod i=1nP(xi|y)P(x1,\ldots,xn)$

Since $P(x1,\ldots,xn)$ is constant given the input, we can use the following classification rule:

$P(y|x1,\ldots,xn)\propto P(y)\prod i=1nP(xi|y)\Downarrow y\hat{}=\arg f0\max yP(y)\prod i=1nP(xi|y)$,

and we can use Maximum A Posteriori (MAP) estimation to estimate P(y) and P(xi|y); the former is then the relative frequency of class y in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi|y). In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. (For theoretical reasons why naive Bayes works well, and on which types of data it does, see the references below.)

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling o

the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.
On the flip side, although naive Bayes is known as a decent classifier, it is known to be a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.

```
nb=GaussianNB()
```

```
parameters.update({"naive bayes": {'var_smoothing': np.logspace(0,-9, num=100)}})
```

**CatBoost** is based on gradient boosted decision trees. During training, a set of decision trees is built consecutively. Each successive tree is built with reduced loss compared to the previous trees. The number of trees is controlled by the starting parameters. Usually, we have to encode categorical features (if any) in the dataset before we use them in algorithms. But with CatBoost, you don't need to do encoding as the algorithm automatically encodes those categorical features into numeric values using one-hot encoding.

```
cb=CatBoostClassifier(iterations=5, learning_rate=0.1)#, loss_function='CrossEntropy')
```

```
parameters.update({"CatBoostClassifier": {"learning_rate": [0.001, 0.01, 0.1, 0.2, 0.3],"depth":[6,7,8,9,10]}})
```

A **Bagging classifier is an ensemble meta-estimator** that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.
This algorithm encompasses several works from the literature. When random subsets of the dataset are drawn as random subsets of the samples, then this algorithm is known as Pasting [1]. If samples are drawn with replacement, then the method is known as Bagging [2]. When random subsets of the dataset are drawn as random subsets of the features, then the method is known as Random Subspaces [3]. Finally, when base estimators are built on subsets of both samples and features, then the method is known as Random Patches [4].
A **Bagging classifier with additional balancing**.This implementation of Bagging is similar to the scikit-learn implementation. It includes an additional step to balance the training set at fit time using a given sampler.
This classifier can serves as a basis to implement various methods such as Exactly Balanced Bagging [6], Roughly Balanced Bagging [7], Over-Bagging [6], or SMOTE-Bagging [8].

```
bb = BalancedBaggingClassifier(base_estimator=LogisticRegression(
    penalty='l2', tol=0.01, class_weight='balanced', verbose=1, n_jobs=1))
```

**VotingClassifier** is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing models in order to balance out their individual weaknesses.
1.11.6.1. Majority Class Labels (Majority/Hard Voting)
In majority voting, the predicted class label for a particular sample is the class label that represents the majority (mode) of the class labels predicted by each individual classifier.
E.g., if the prediction for a given sample is classifier 1 -> class 1 classifier 2 -> class 1 classifier 3 -> class 2

the VotingClassifier (with voting='hard') would classify the sample as “class 1” based on the majority class label. In the cases of a tie, the VotingClassifier will select the class based on the ascending sort order. E.g., in the following scenario classifier 1 -> class 2 classifier 2 -> class 1 the class label 1 will be assigned to the sample.

```
votingCLF = VotingClassifier(
    estimators=ensembleEstimators, voting='hard', n_jobs=-1, verbose=True)
```

Stacked generalization is a method for combining estimators to reduce their biases . More precisely, the predictions of each individual estimator are stacked together and used as input to a final estimator to compute the prediction. This final estimator is trained through cross-validation.

The **StackingClassifier** and StackingRegressor provide such strategies which can be applied to classification and regression problems.The estimators parameter corresponds to the list of the estimators which are stacked together in parallel on the input data. It should be given as a list of names and estimators:

```
stackingCLF = StackingClassifier(
    estimators=ensembleEstimators, final_estimator=ensembleEstimators[0][1], passthrough=True, cv=3)
```

*Figure 43: Hyper parameter tuning*

### 14.2 Cross Validation technique

Cross validation and cross_val_predict were applied on all models to get the most generalized models.

```
def randomizedSearch(self, paramGrid, m, name):
    try:
        # train and predict each model with grid search CV
        kFold = StratifiedKFold(shuffle=True, random_state=50)
        #gscv = GridSearchCV(m, param_grid=paramGrid, cv = kFold,  n_jobs= -1, verbose = 1, scoring = "roc_auc")
        rscv = RandomizedSearchCV(m, param_distributions=paramGrid, n_jobs=-1, scoring='roc_auc', cv=kFold,
                                  n_iter=10, verbose=1, random_state=30)
        # Fit gscv
        #print(f"Now tuning {m}.")
        rscv.fit(self.predictorTrain, np.ravel(self.targetTrain))
        prediction = rscv.best_estimator_.predict(self.predictorTest)
        #rscvPredictions = rscvPredictions.rename(columns={i: name})

        auc = metrics.roc_auc_score(self.targetTest, prediction)

        #bestParams = rscv.best_params_

        # update score for train and test
        self.aucScores.update({"Model": [name] + self.aucScores["Model"],
                               "AUC": [auc] + self.aucScores["AUC"],
                               "AUC Type": ["Test"] + self.aucScores["AUC Type"]})

        self.aucScores.update({"Model": [name] + self.aucScores["Model"],
                               "AUC": [rscv.best_score_] + self.aucScores["AUC"],
                               "AUC Type": ["Train"] + self.aucScores["AUC Type"]})

        self.plotLearningCurve(rscv.best_estimator_, 'Learning Curves', self.predictorTrain, self.targetTrain,
                               cv=kFold, n_jobs=-1)

        return rscv.best_estimator_, rscv.best_params_, prediction
    except Exception as exp:
        self.errObj = ErrorHandler()
        err = self.errObj.handleErr(str(exp))
        print(str(err))
```

RFE (Recursive Feature Elimination) CV technique

RFE CV technique was used to eliminate the not important features and find the most optimal set of features

```
def rfeCV(self, model,X_train,y_train,X_test,y_test,X_columns, n_params,name):

  rfeCV = RFECV(estimator=model, step=1,cv= KFold(), scoring= 'r2')
  rfeCV.fit(X_train,y_train)
  prediction_rfecv = rfeCV.predict(X_test)
  if(self.dataset_type == 'detrend'):# inverse the detrend by adding trend and seasonal info
      # inverse detrend for the predictons
      prediction_rfecv = self.inverse_detrend(prediction_rfecv,self.trend,self.seasonal)

  loss_values = model_utility().calculate_metrics(y_test,prediction_rfecv,len(y_test),n_params,name)
  print("RFECV - optimal number of features ", rfeCV.n_features_)
  optimal_features = []
  for i in range(len(X_columns)):
    if rfeCV.ranking_[i]==1:
      optimal_features.append(X_columns[i])
  print("RFE CV - optimal features ", optimal_features )
  plt.figure(figsize=(16, 8))
  plt.title('Recursive Feature Elimination with CV - '+ name, fontsize=18, fontweight='bold', pad=20)
  plt.xlabel('Number of features selected', fontsize=14, labelpad=20)
  plt.ylabel('R2 score', fontsize=14, labelpad=20)
  plt.plot(range(1, len(rfeCV.grid_scores_) + 1), rfeCV.grid_scores_, color='#303F9F', linewidth=3)
  plt.show()
  return prediction_rfecv,loss_values
```

## 14.3 Example of feature elimination for XGB:

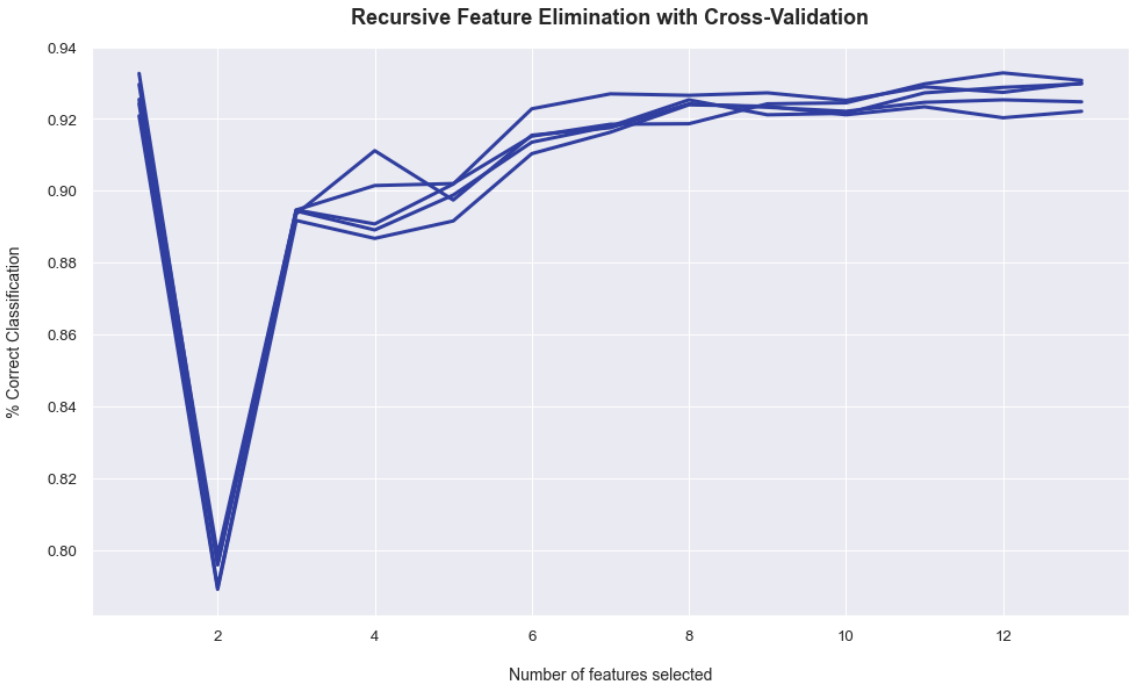



*Figure 44: Recursive feature elimination*

## 14.4 Feature Importance:

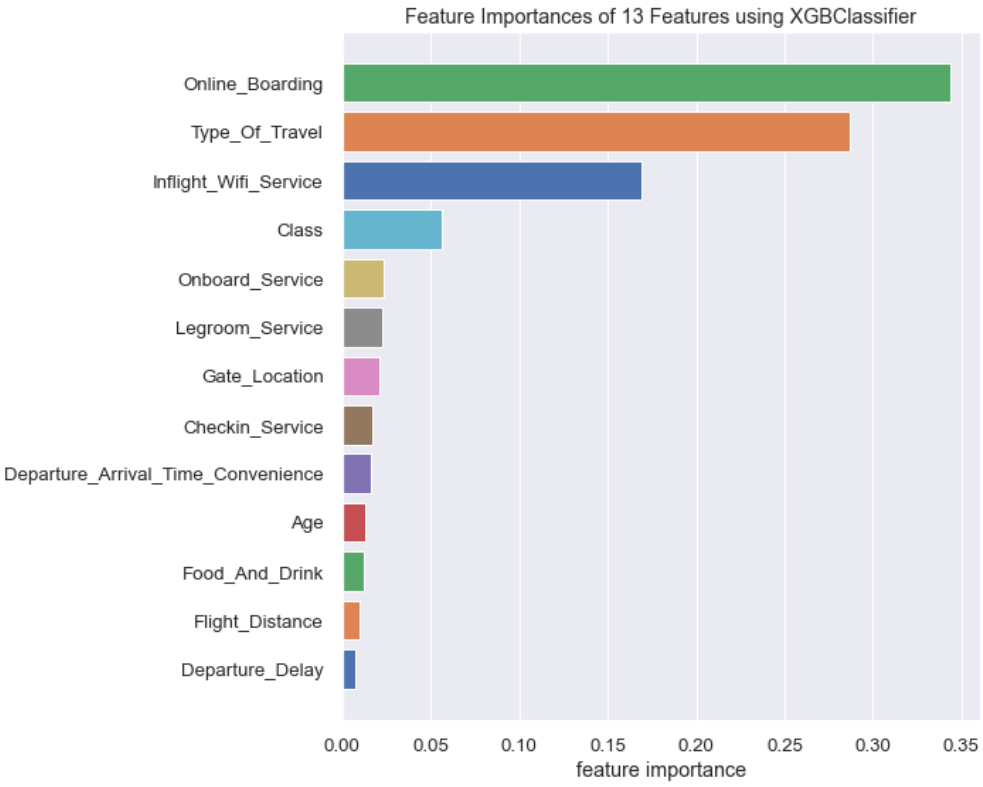
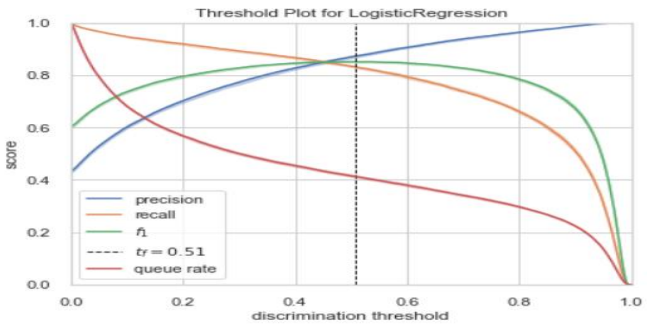


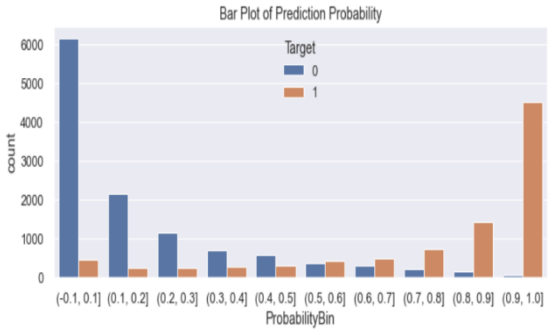

*Figure 45: Feature importance using normal dataset*

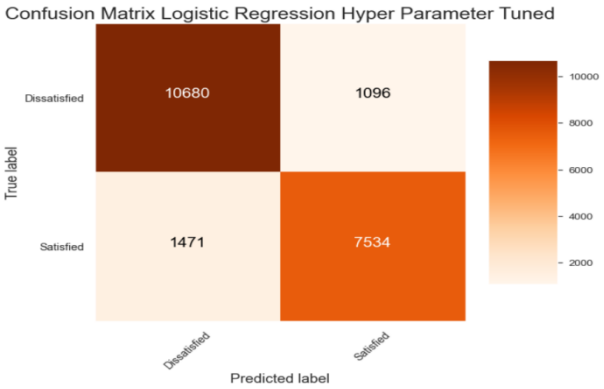
## 15 Modelling Output

The above classifier models were trained using Unbinned Data, Binned Data and data reduced in dimensions using PCA and VIF+ChiSquare analysis

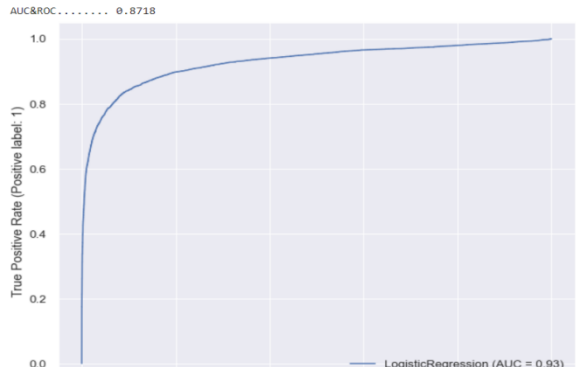
### Logistic Regression Model - UnBinned Data

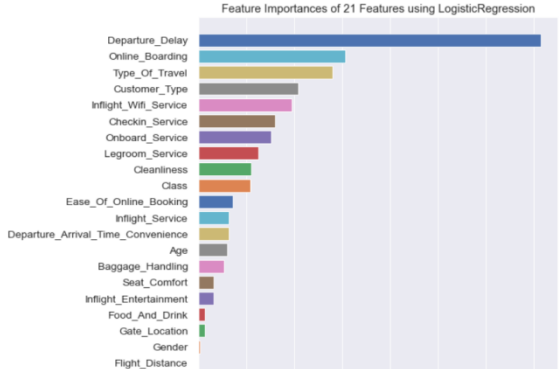



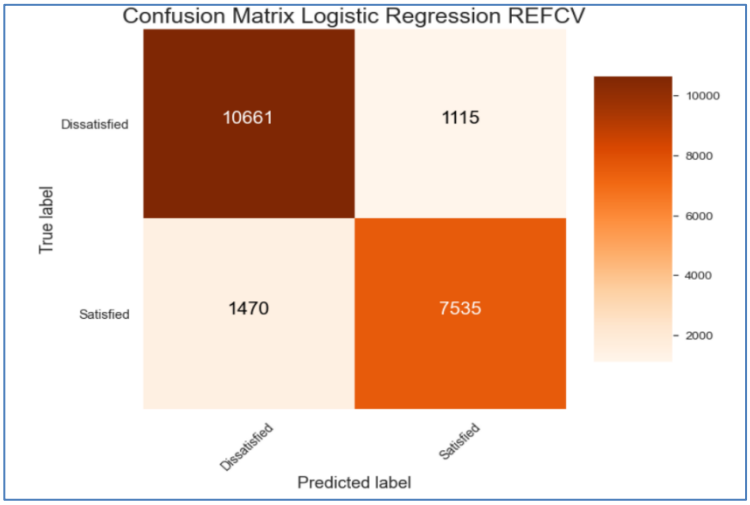




```
Misclassified Counts: 2567
accuracy........ 0.8765
precision score........ 0.8760
recall score........ 0.8718
classification_report
              precision    recall  f1-score   support

           0       0.88      0.91      0.89     11776
           1       0.87      0.84      0.85      9005

    accuracy                           0.88     20781
   macro avg       0.88      0.87      0.87     20781
weighted avg       0.88      0.88      0.88     20781
```

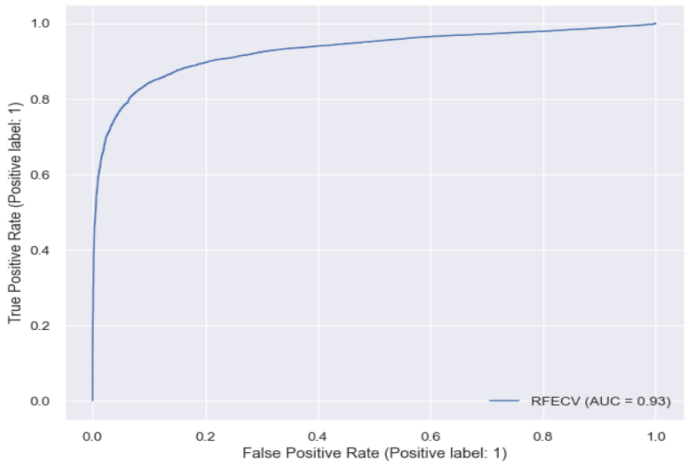


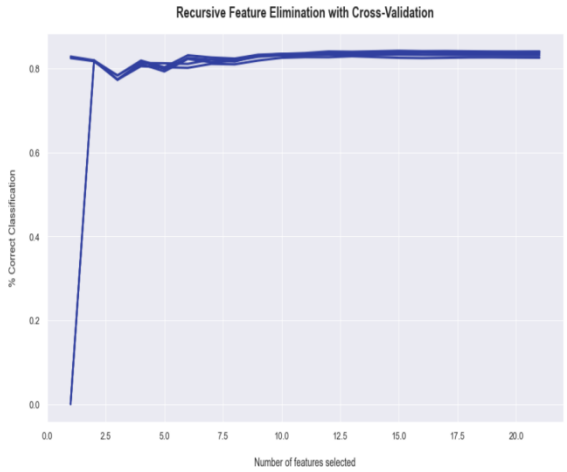





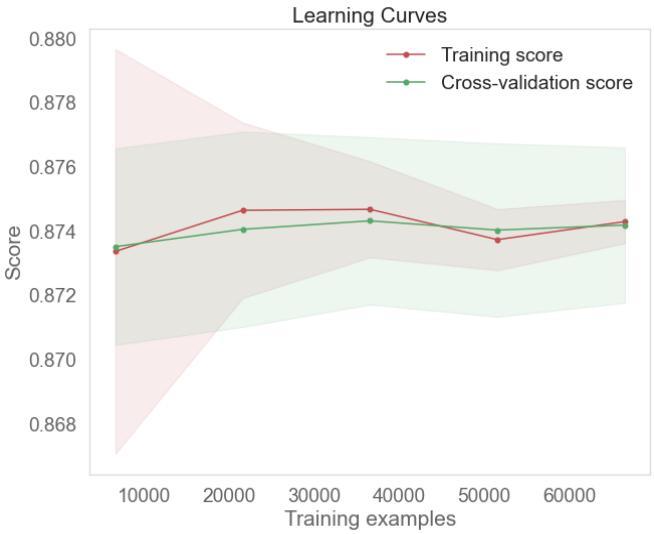

```
Misclassified Counts: 2585
accuracy........ 0.8756
precision score........ 0.8750
recall score........ 0.8710
classification_report
              precision    recall  f1-score   support

           0       0.88      0.91      0.89     11776
           1       0.87      0.84      0.85      9005

    accuracy                           0.88     20781
   macro avg       0.87      0.87      0.87     20781
weighted avg       0.88      0.88      0.88     20781
```

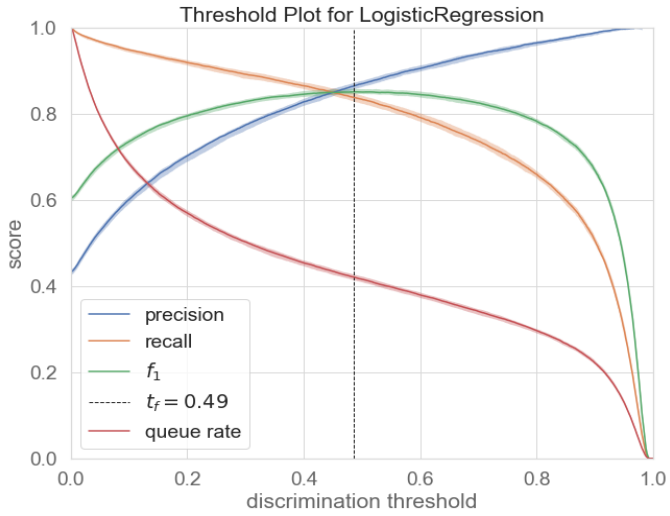
```
AUC&ROC........ 0.8710
```

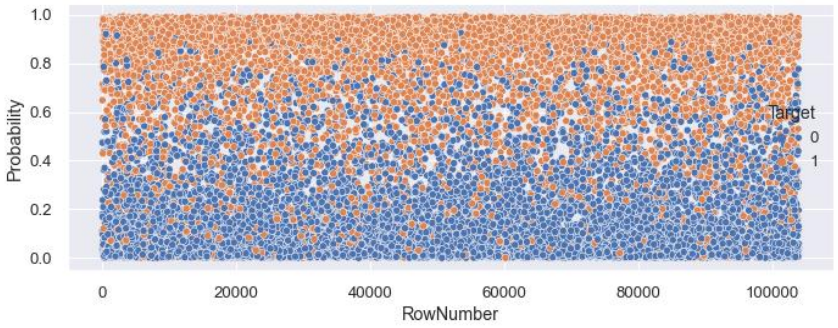


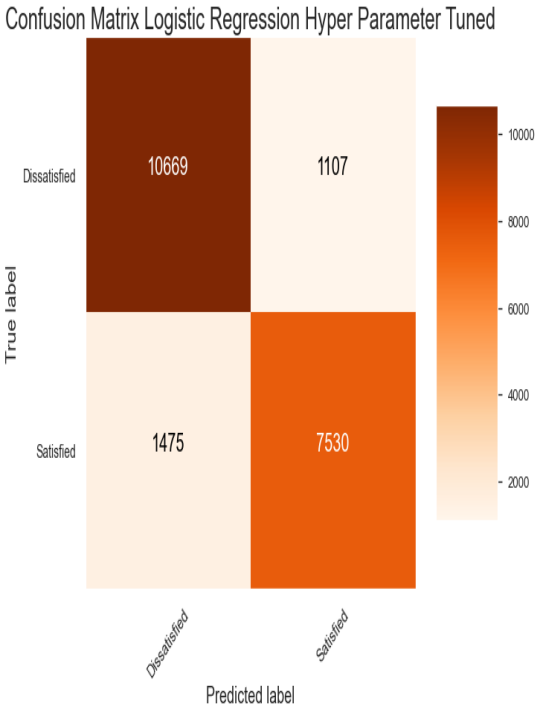



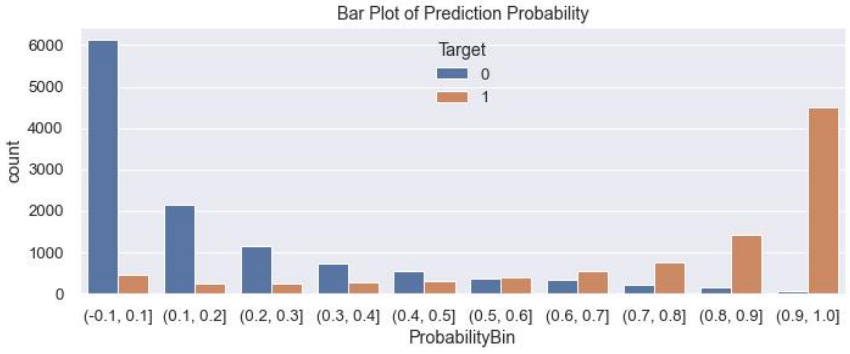
## Logistic Regression Model - Binned Data



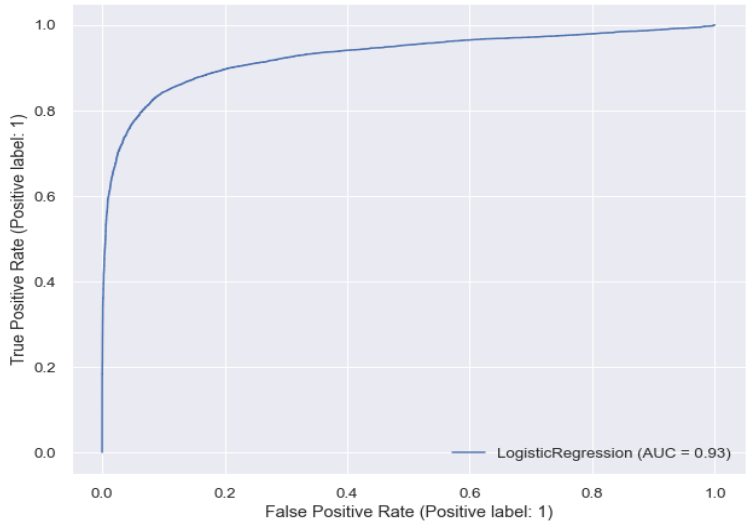



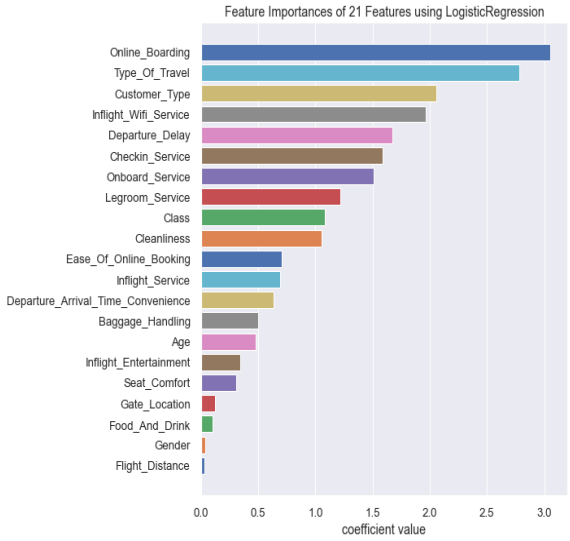



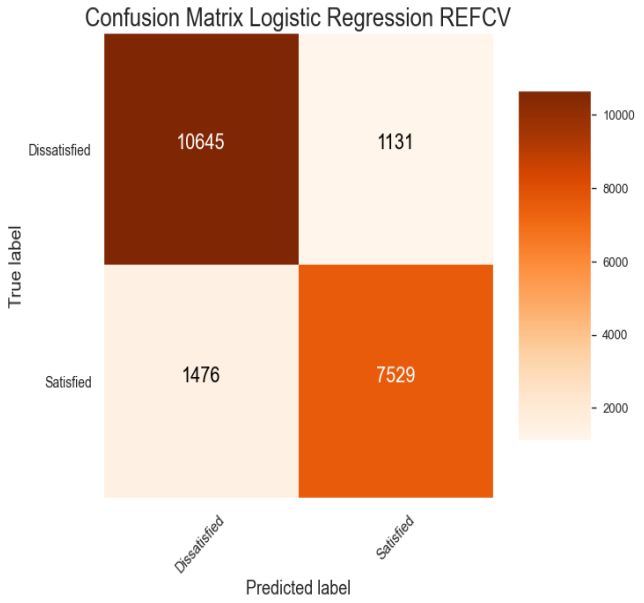



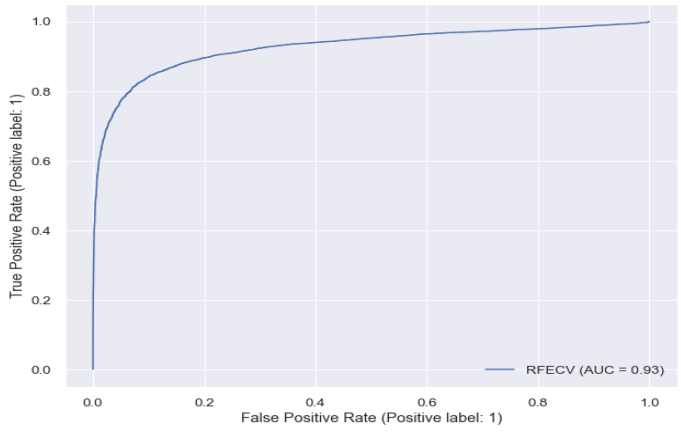

```
Misclassified Counts: 2582
accuracy........ 0.8758
precision score........ 0.8752
recall score........ 0.8711
classification_report
                precision    recall  f1-score   support

           0       0.88      0.91      0.89     11776
           1       0.87      0.84      0.85      9005

    accuracy                           0.88     20781
   macro avg       0.88      0.87      0.87     20781
weighted avg       0.88      0.88      0.88     20781
```

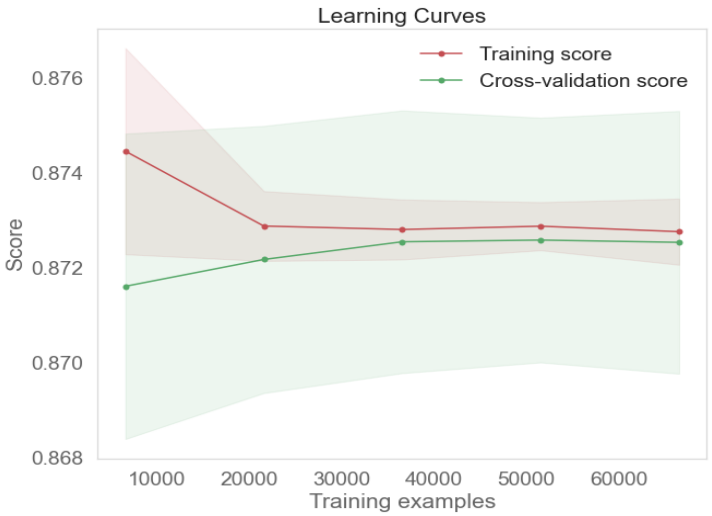


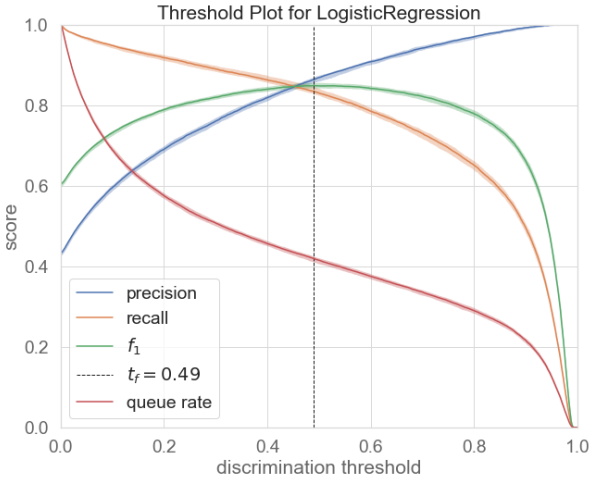



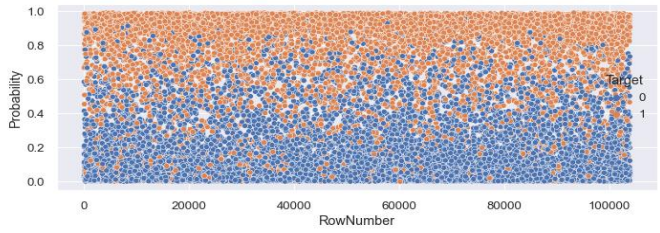



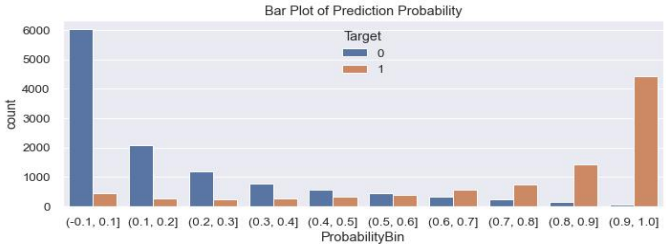

```
Misclassified Counts: 2607
accuracy........ 0.8745
precision score........ 0.8738
recall score........ 0.8700
classification_report
              precision    recall  f1-score   support

           0       0.88      0.90      0.89     11776
           1       0.87      0.84      0.85      9005

    accuracy                           0.87     20781
   macro avg       0.87      0.87      0.87     20781
weighted avg       0.87      0.87      0.87     20781
```

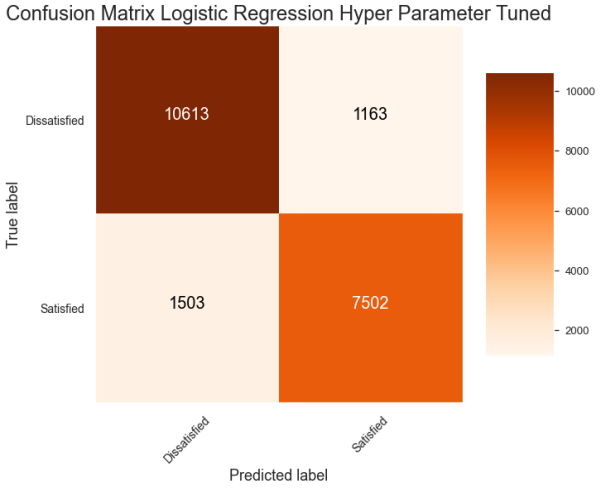


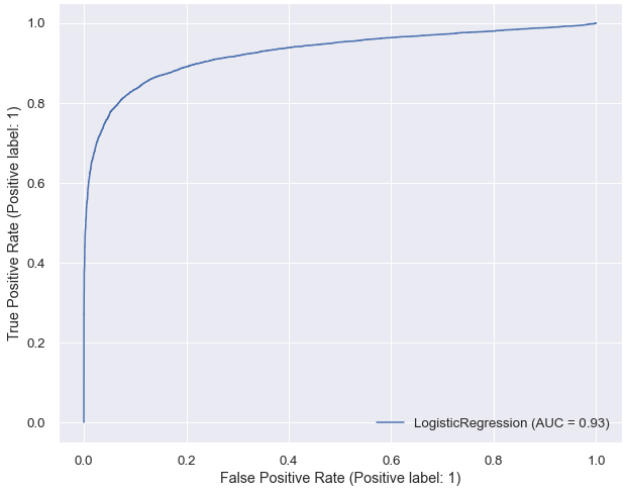



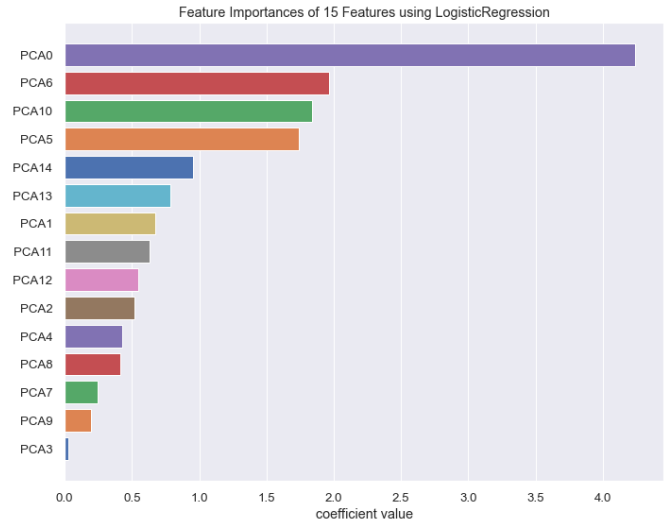



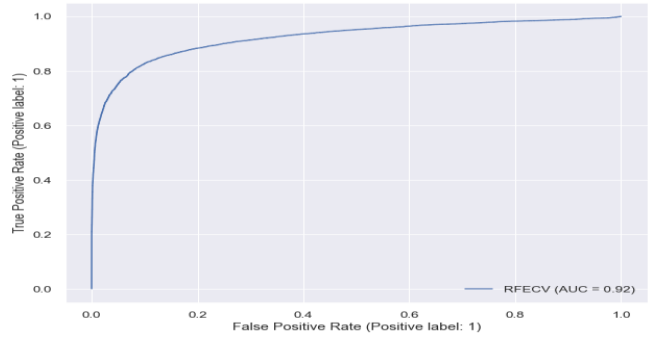



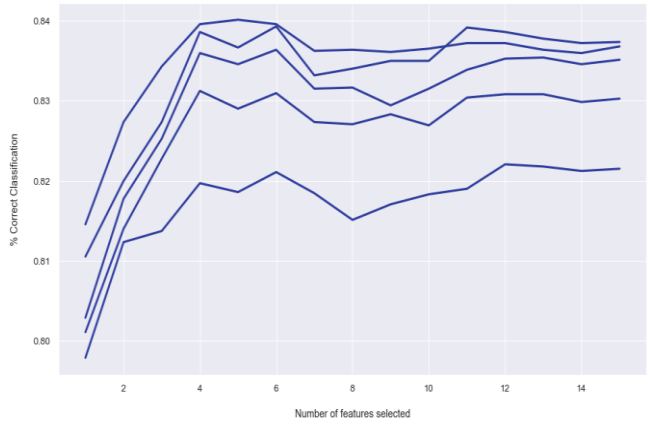



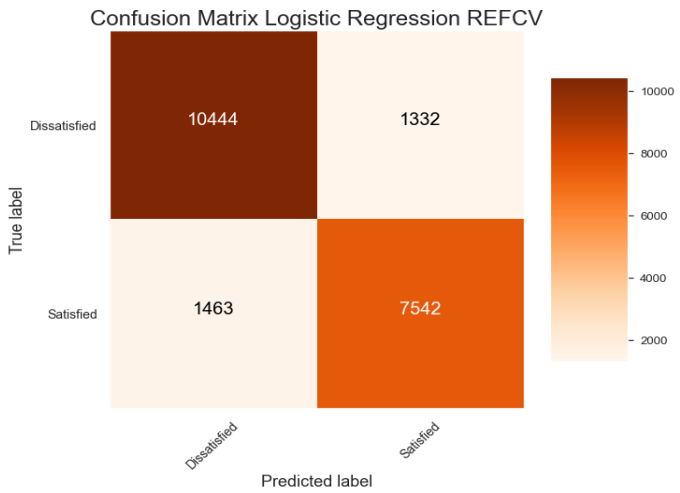

```
Misclassified Counts: 2666
accuracy........ 0.8717
precision score........ 0.8709
recall score........ 0.8672
classification_report
              precision    recall  f1-score   support

           0       0.88      0.90      0.89     11776
           1       0.87      0.83      0.85      9005

    accuracy                           0.87     20781
   macro avg       0.87      0.87      0.87     20781
weighted avg       0.87      0.87      0.87     20781
```



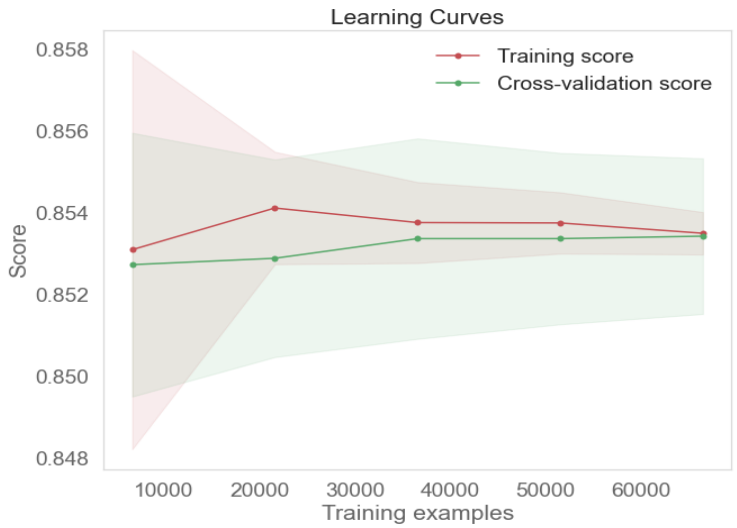


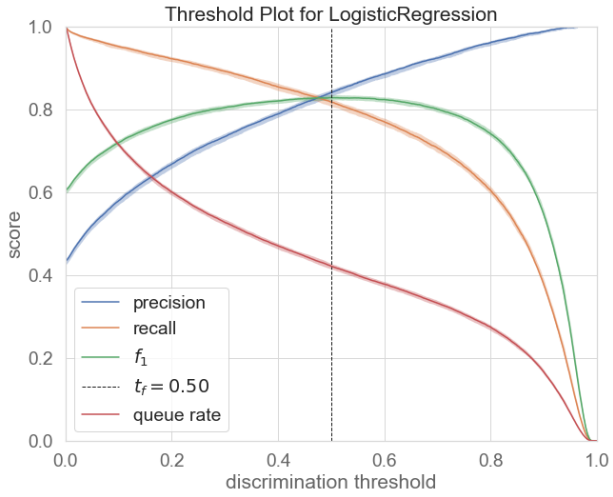



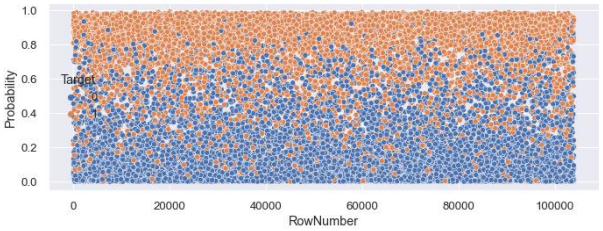



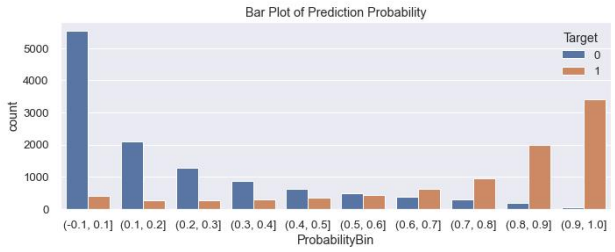



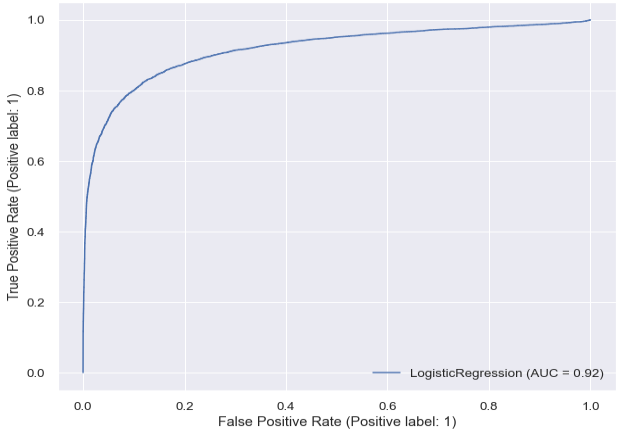

```
Misclassified Counts: 2795
accuracy........ 0.8655
precision score........ 0.8635
recall score........ 0.8622
classification_report
              precision    recall  f1-score   support

           0       0.88      0.89      0.88     11776
           1       0.85      0.84      0.84      9005

    accuracy                           0.87     20781
   macro avg       0.86      0.86      0.86     20781
weighted avg       0.87      0.87      0.87     20781
```

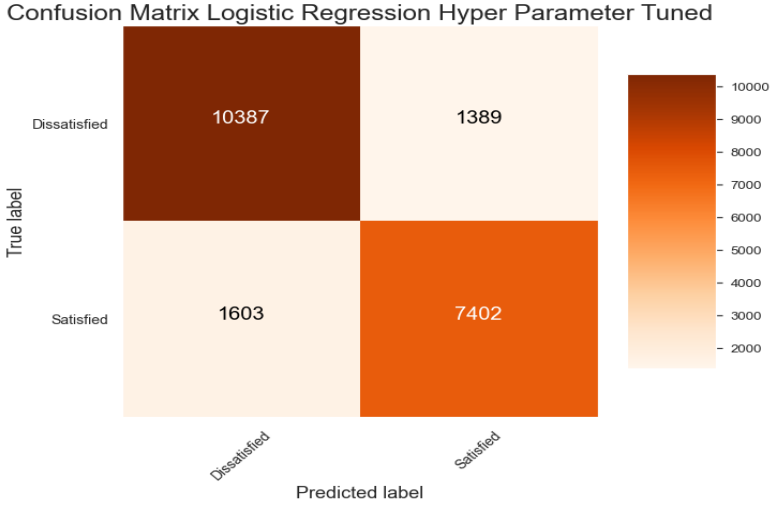


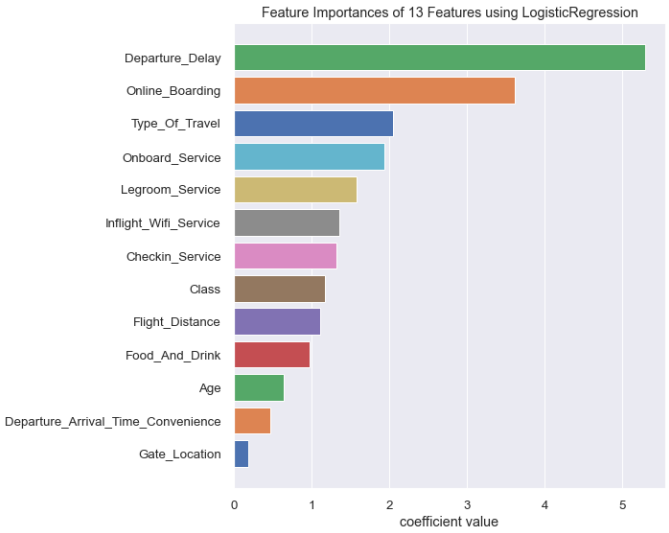



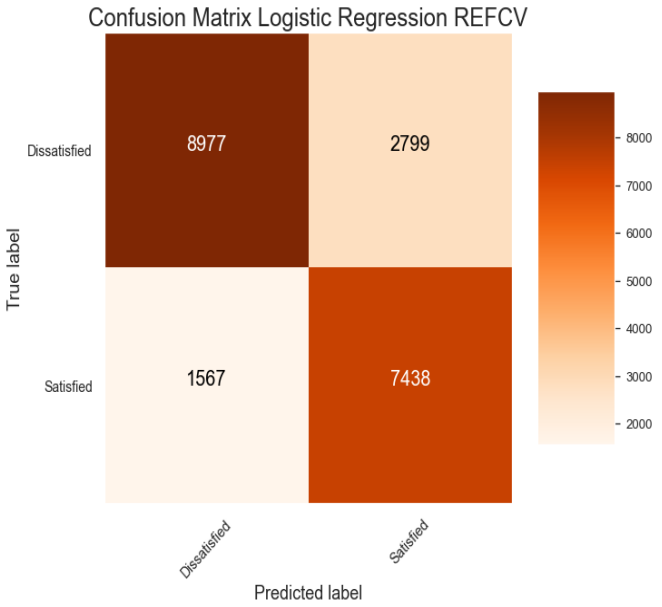



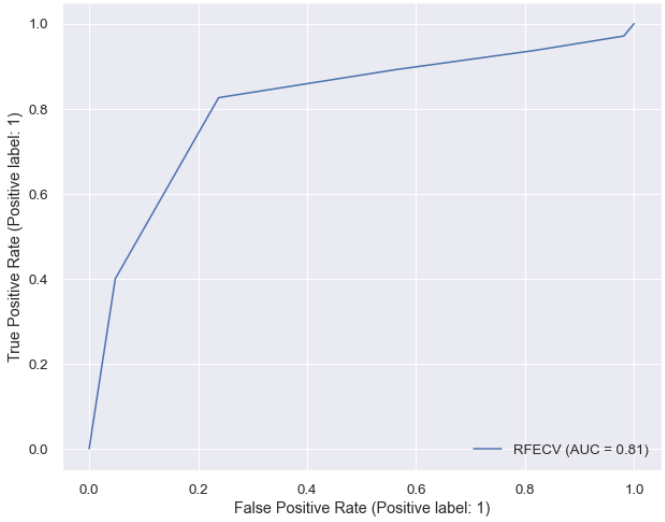

```
Misclassified Counts: 2992
accuracy........ 0.8560
precision score........ 0.8542
recall score........ 0.8520
classification_report
              precision    recall  f1-score   support

           0       0.87      0.88      0.87     11776
           1       0.84      0.82      0.83      9005

    accuracy                           0.86     20781
   macro avg       0.85      0.85      0.85     20781
weighted avg       0.86      0.86      0.86     20781
```

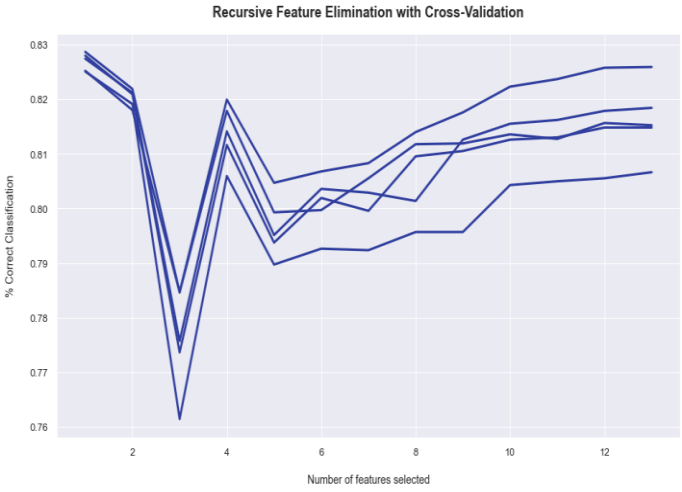
## Logistic Regression Model - VIF Contd..

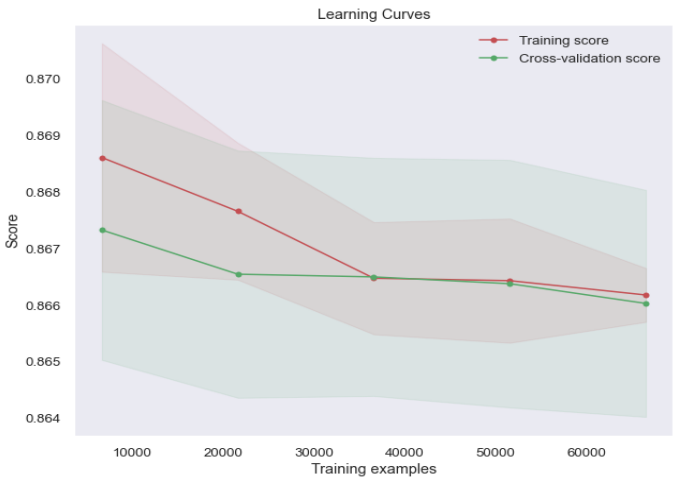


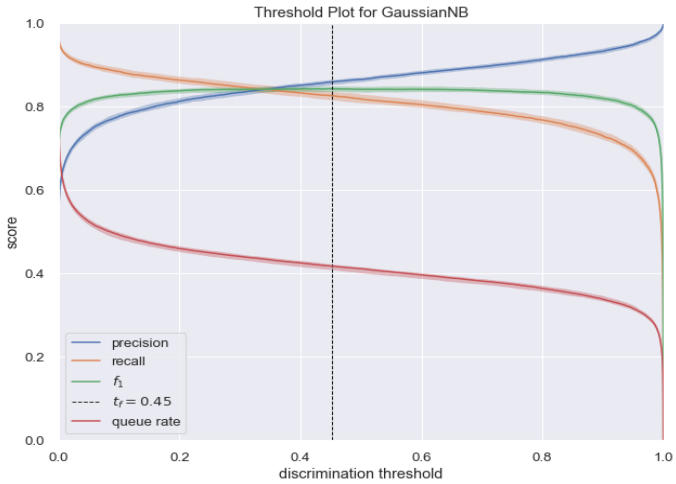



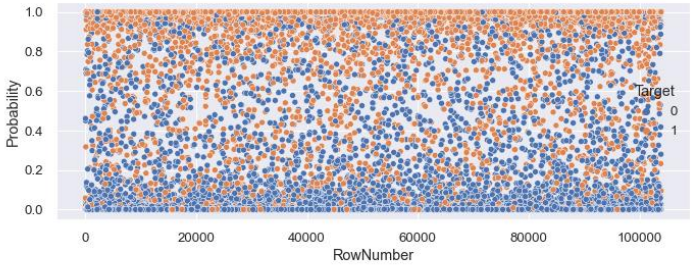

```
Misclassified Counts: 4366
accuracy........ 0.7899
precision score........ 0.7890
recall score........ 0.7941
classification_report
              precision    recall  f1-score   support

           0       0.85      0.76      0.80     11776
           1       0.73      0.83      0.77      9005

    accuracy                           0.79     20781
   macro avg       0.79      0.79      0.79     20781
weighted avg       0.80      0.79      0.79     20781
```

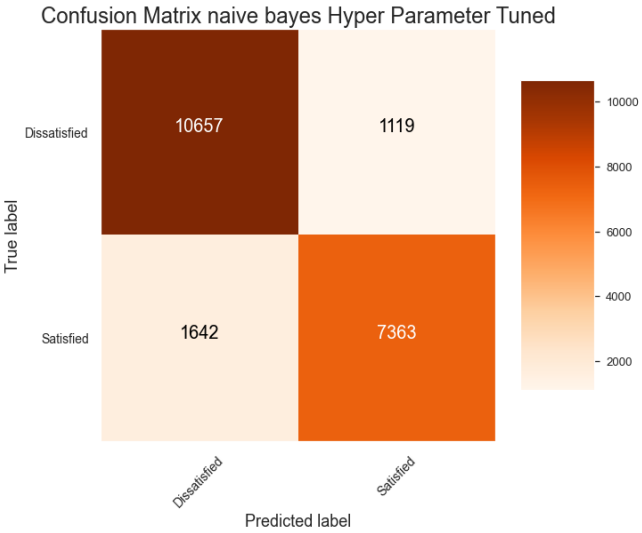


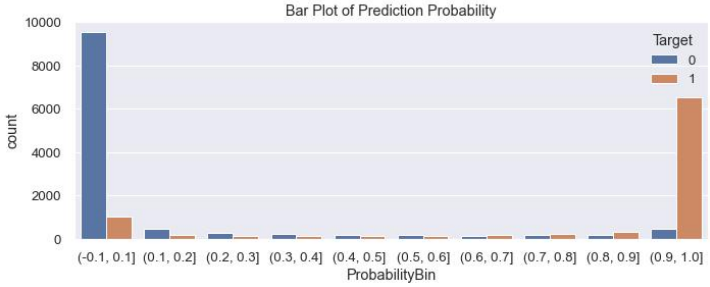



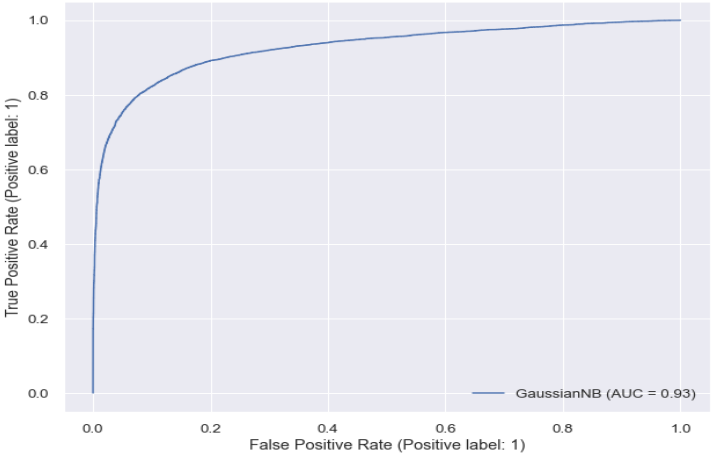


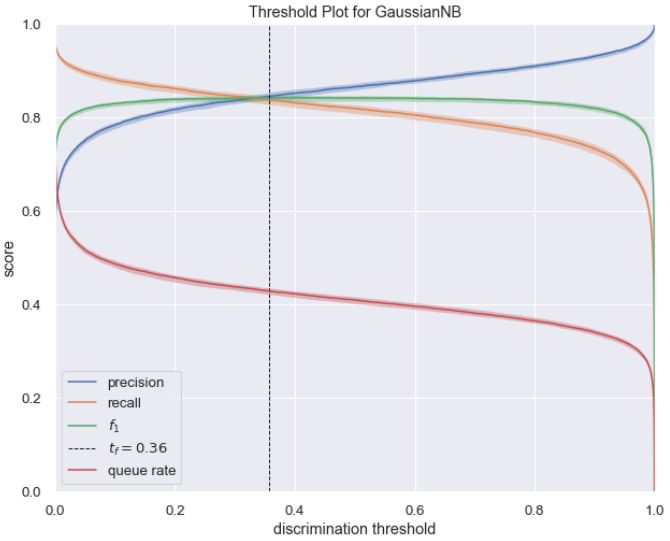



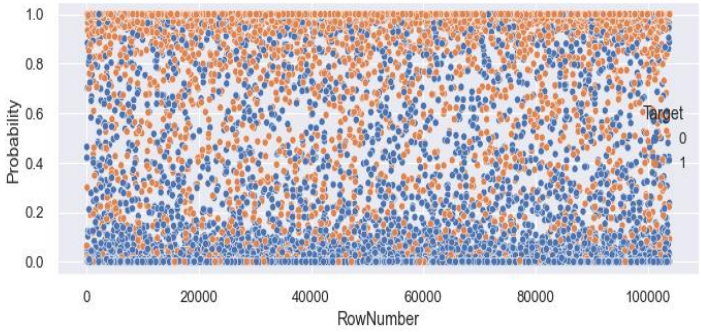



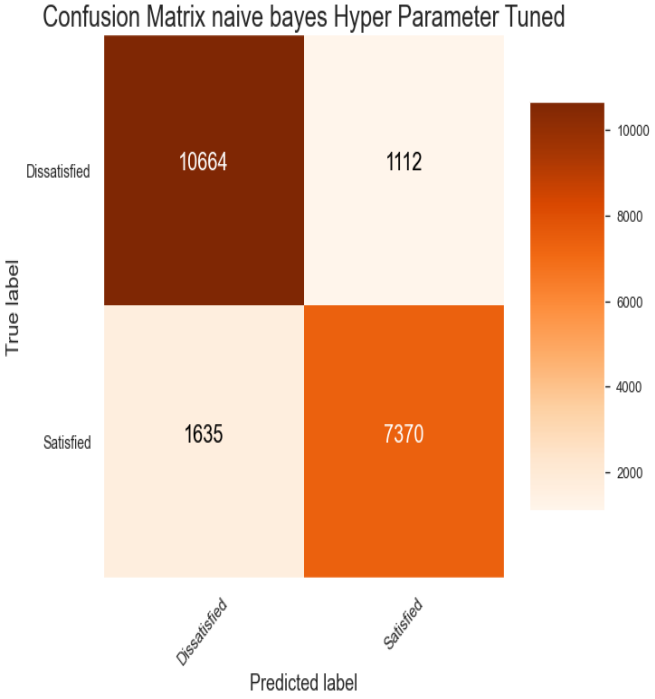



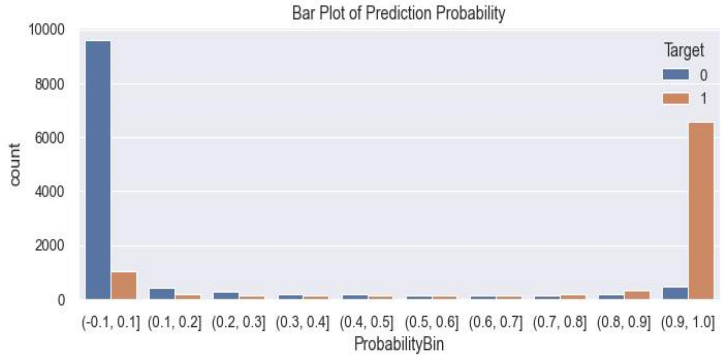



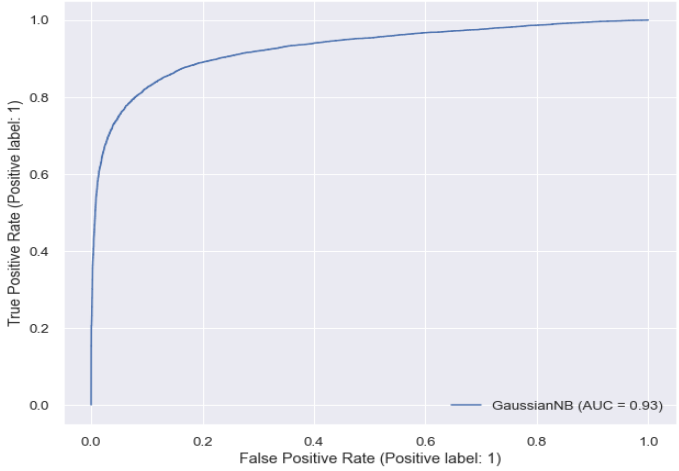

```
Misclassified Counts: 2761
accuracy........ 0.8671
precision score........ 0.8673
recall score........ 0.8613
classification_report
              precision    recall  f1-score   support

           0       0.87      0.90      0.89     11776
           1       0.87      0.82      0.84      9005

    accuracy                           0.87     20781
   macro avg       0.87      0.86      0.86     20781
weighted avg       0.87      0.87      0.87     20781
```

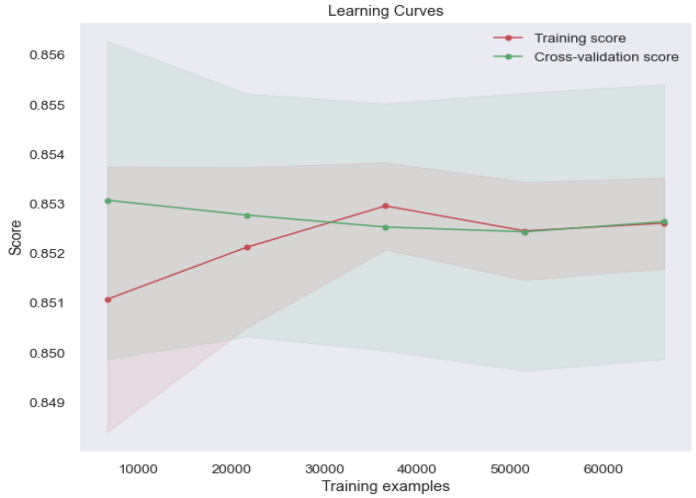



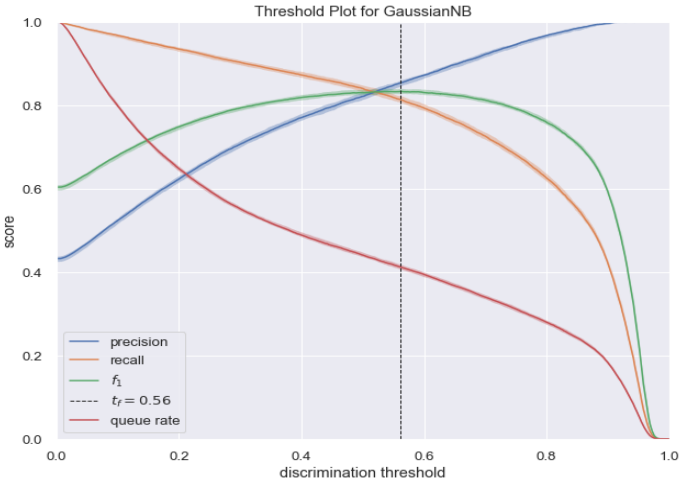



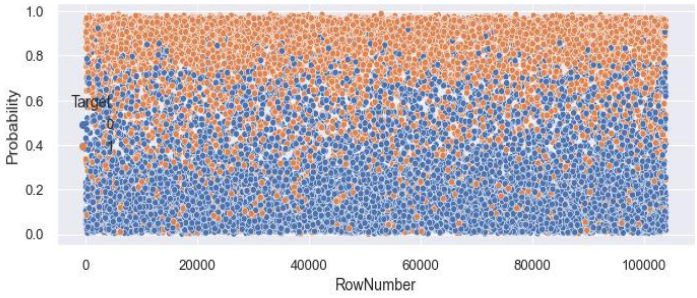



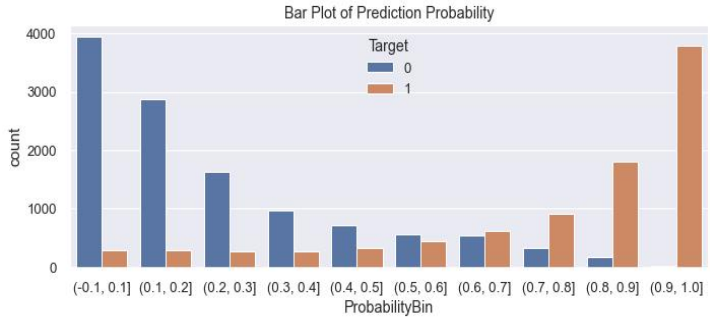



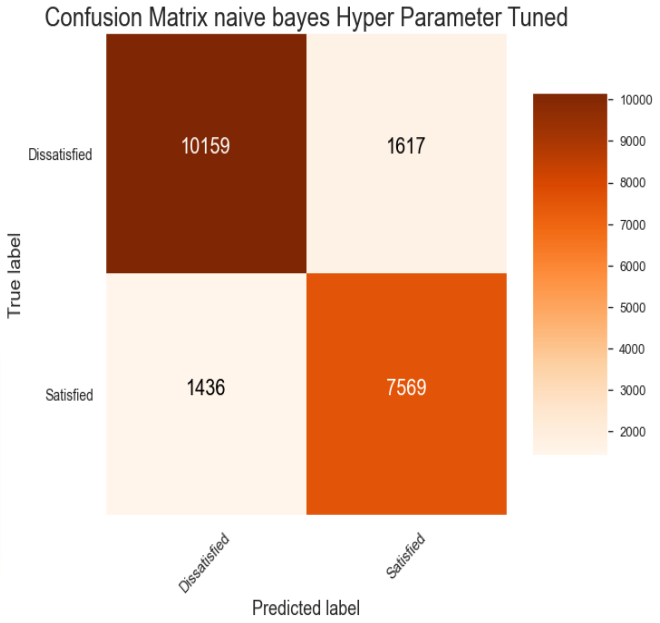

```
Misclassified Counts: 2747
accuracy........ 0.8678
precision score........ 0.8680
recall score........ 0.8620
classification_report
              precision    recall  f1-score   support

           0       0.87      0.91      0.89     11776
           1       0.87      0.82      0.84      9005

    accuracy                           0.87     20781
   macro avg       0.87      0.86      0.86     20781
weighted avg       0.87      0.87      0.87     20781
```

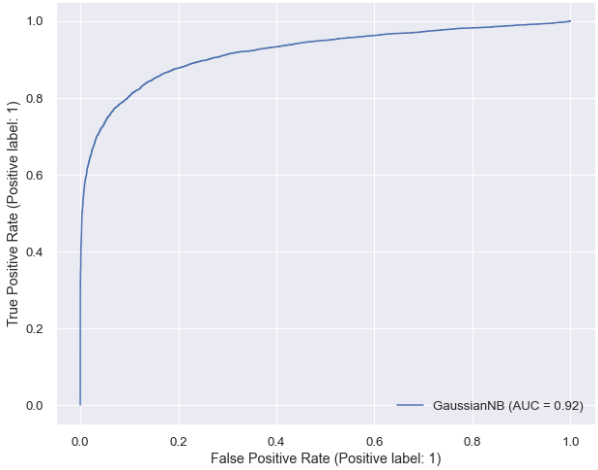


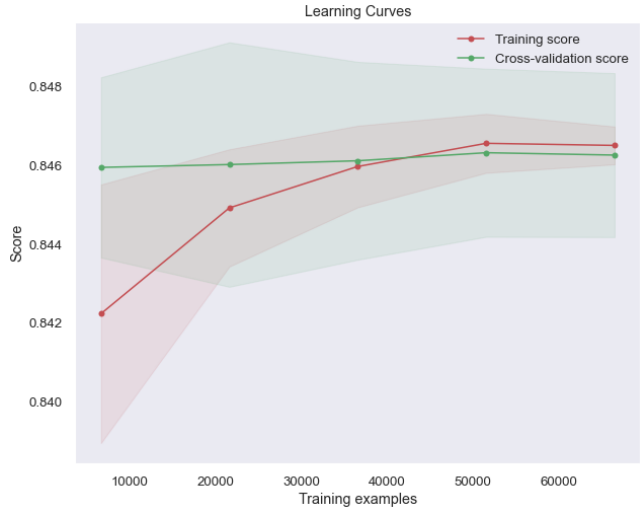



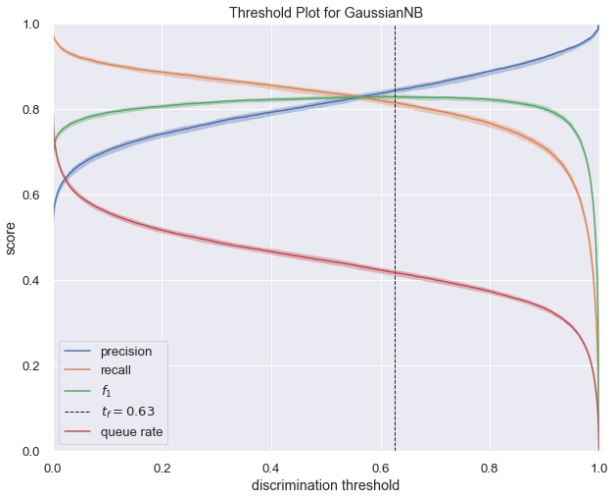



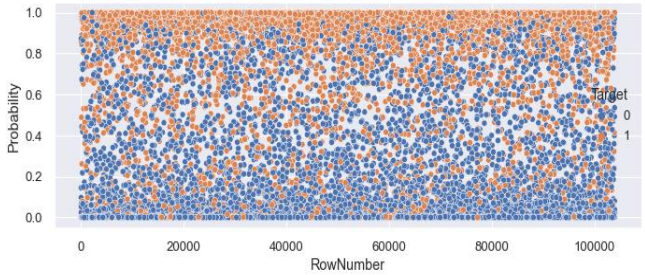



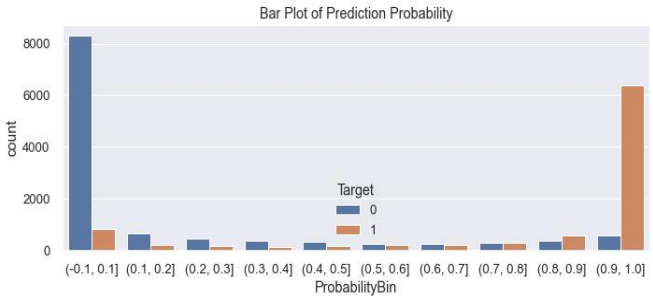



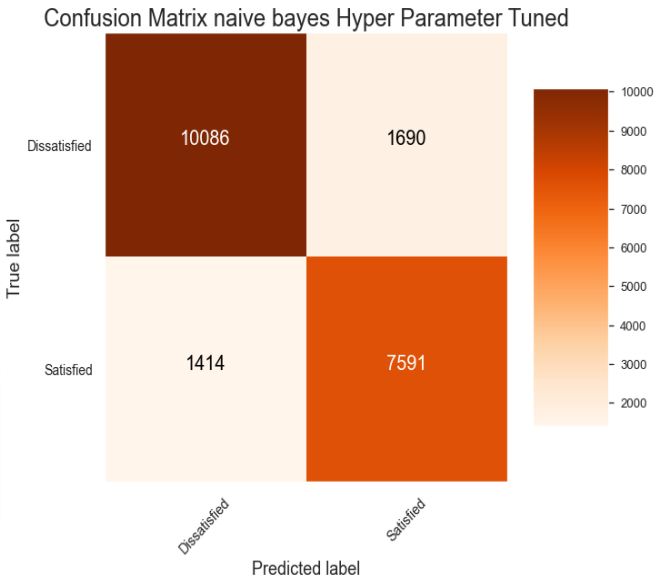

```
Misclassified Counts: 3053
accuracy........ 0.8531
precision score........ 0.8501
recall score........ 0.8516
classification_report
              precision    recall  f1-score   support

           0       0.88      0.86      0.87     11776
           1       0.82      0.84      0.83      9005

    accuracy                           0.85     20781
   macro avg       0.85      0.85      0.85     20781
weighted avg       0.85      0.85      0.85     20781
```

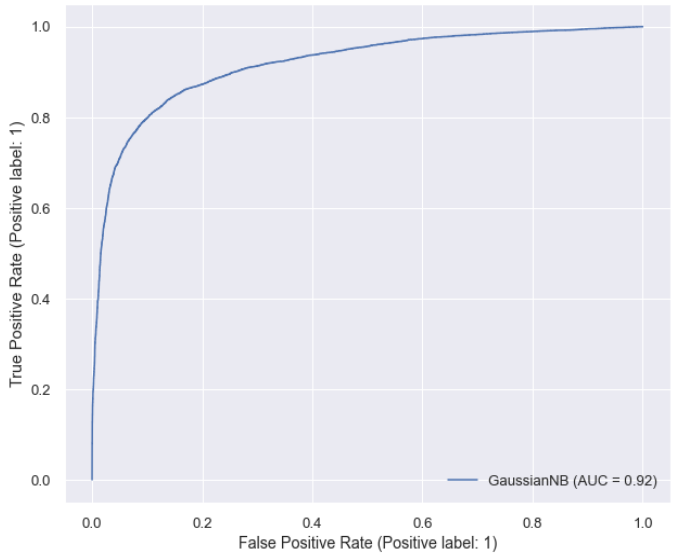


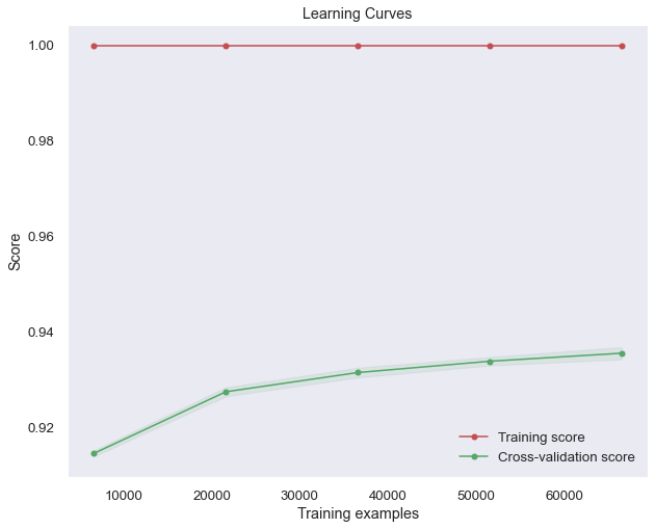



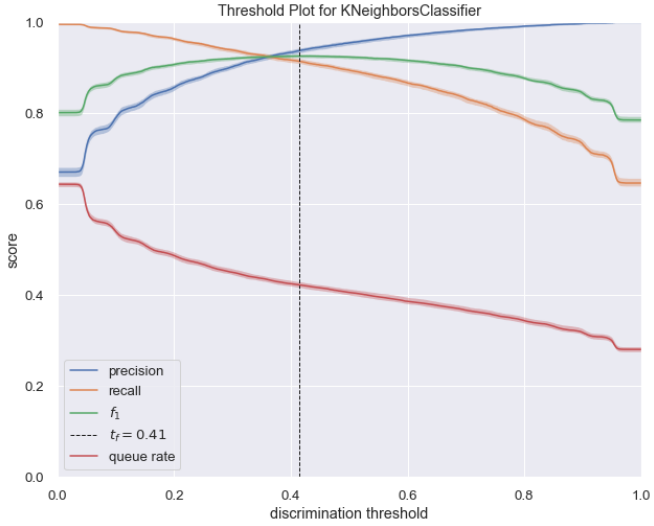



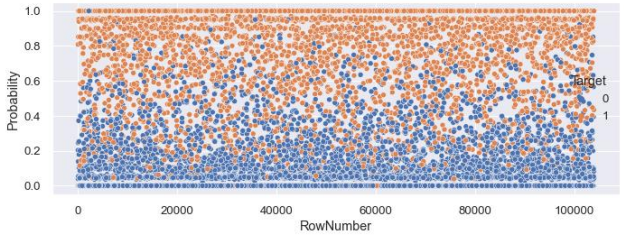



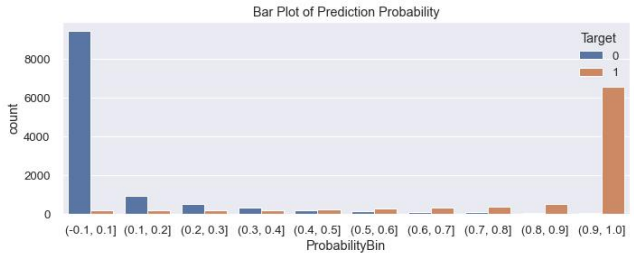



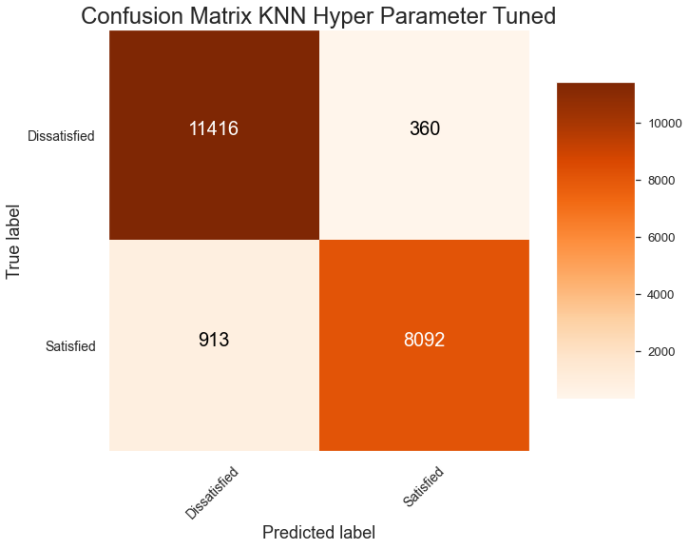

```
Misclassified Counts: 3104
accuracy........ 0.8506
precision score........ 0.8475
recall score........ 0.8497
classification_report
              precision    recall  f1-score   support

           0       0.88      0.86      0.87     11776
           1       0.82      0.84      0.83      9005

    accuracy                           0.85     20781
   macro avg       0.85      0.85      0.85     20781
weighted avg       0.85      0.85      0.85     20781
```

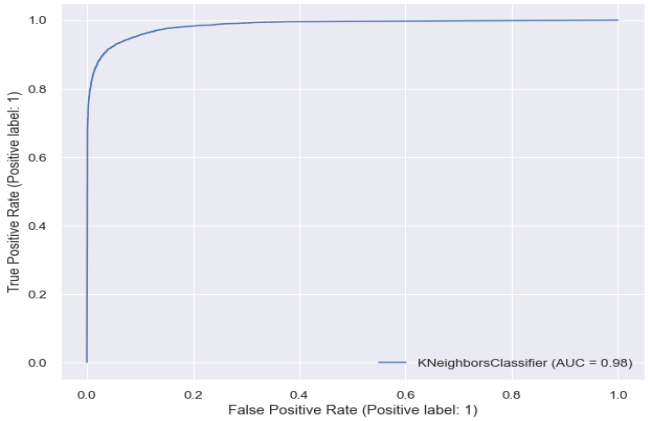


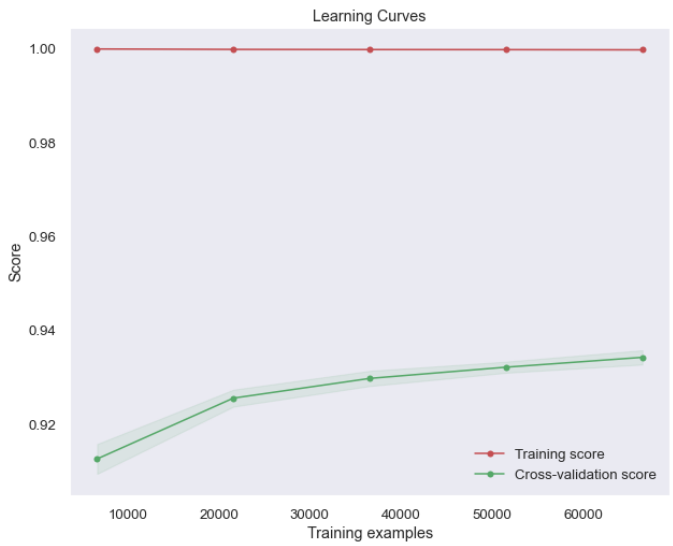



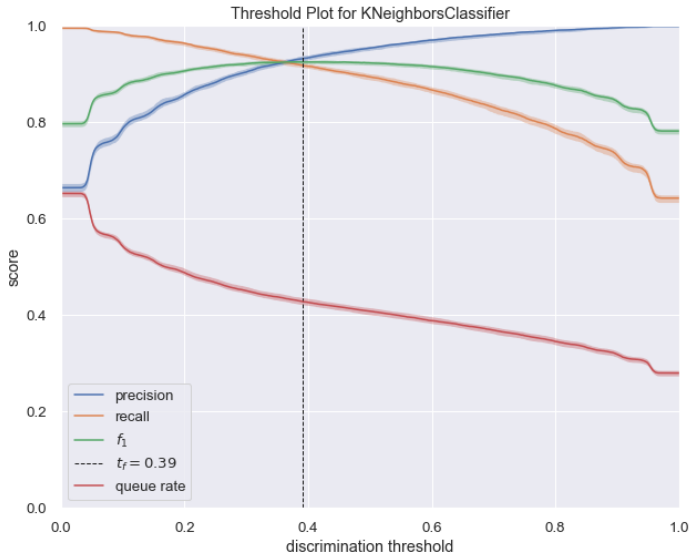



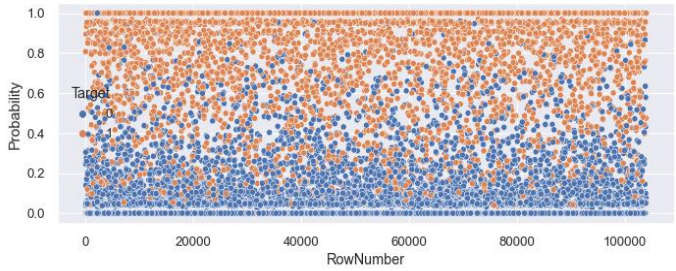



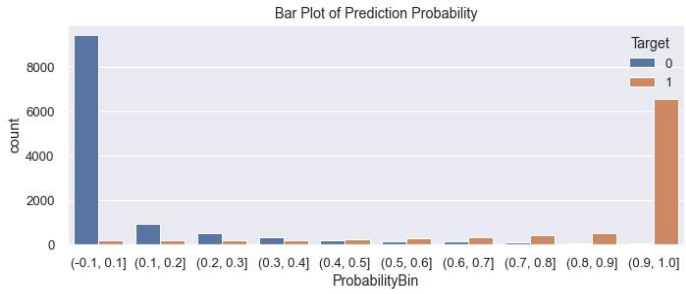



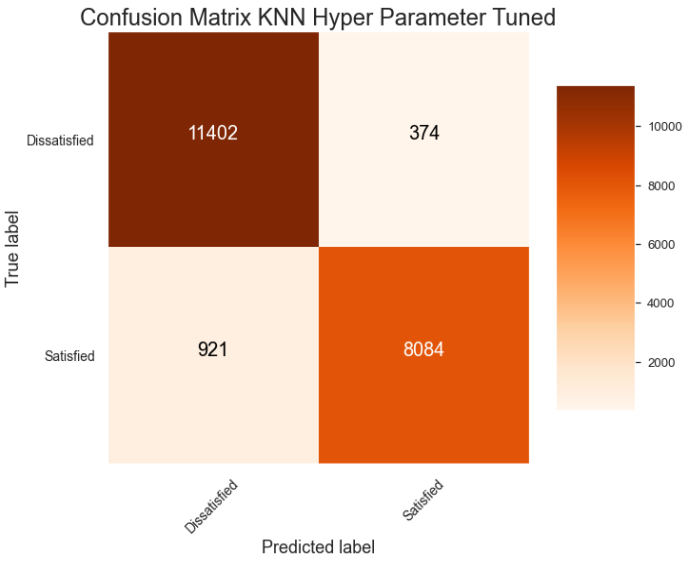

```
Misclassified Counts: 1273
accuracy........ 0.9387
precision score........ 0.9417
recall score........ 0.9340
classification_report
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     11776
           1       0.96      0.90      0.93      9005

    accuracy                           0.94     20781
   macro avg       0.94      0.93      0.94     20781
weighted avg       0.94      0.94      0.94     20781
```

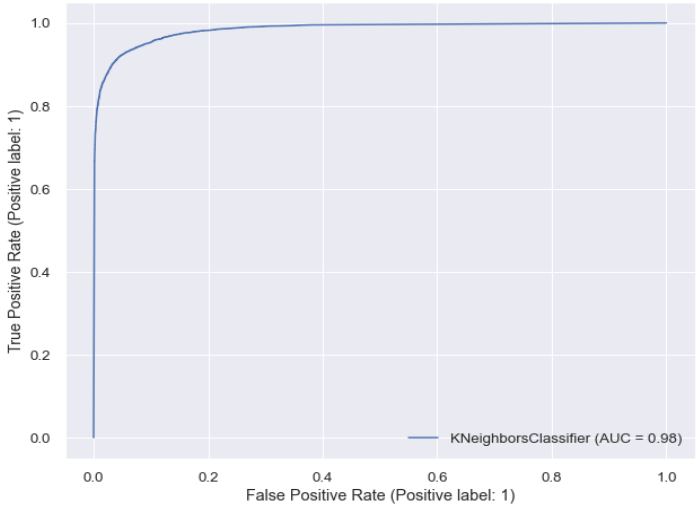


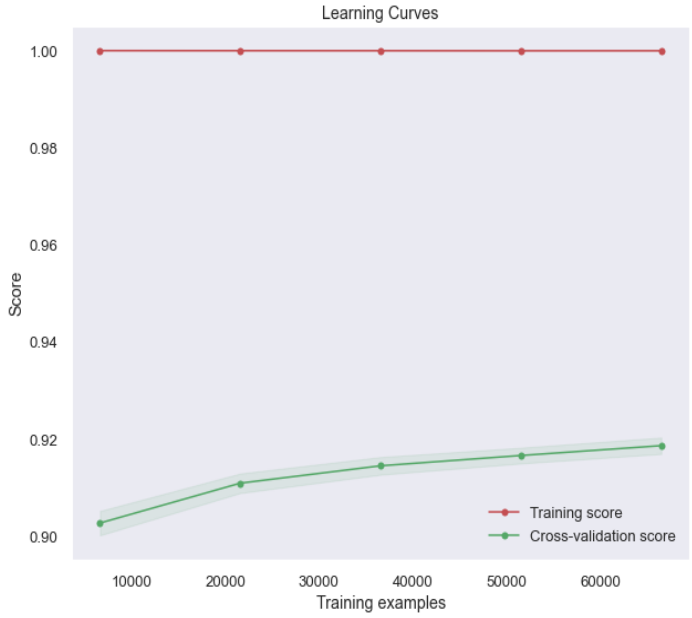



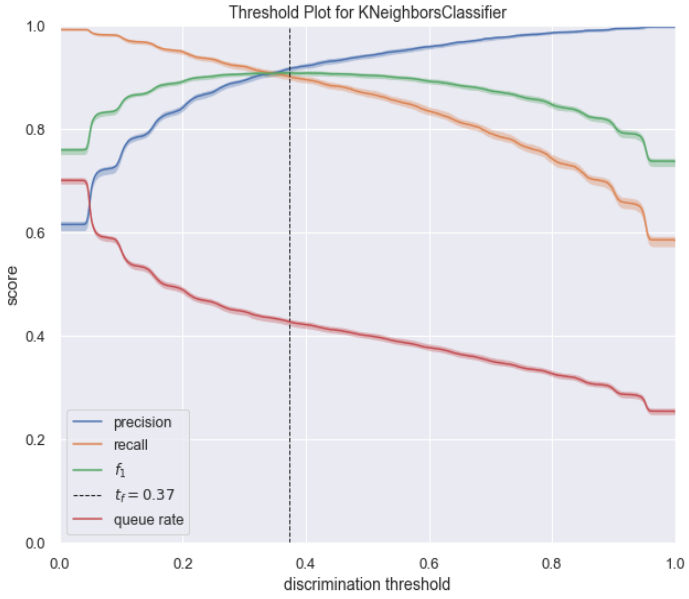



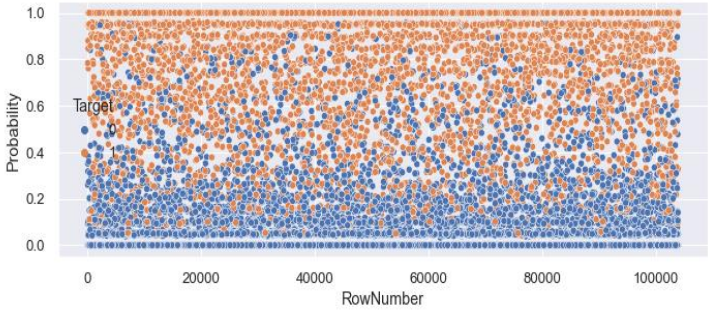



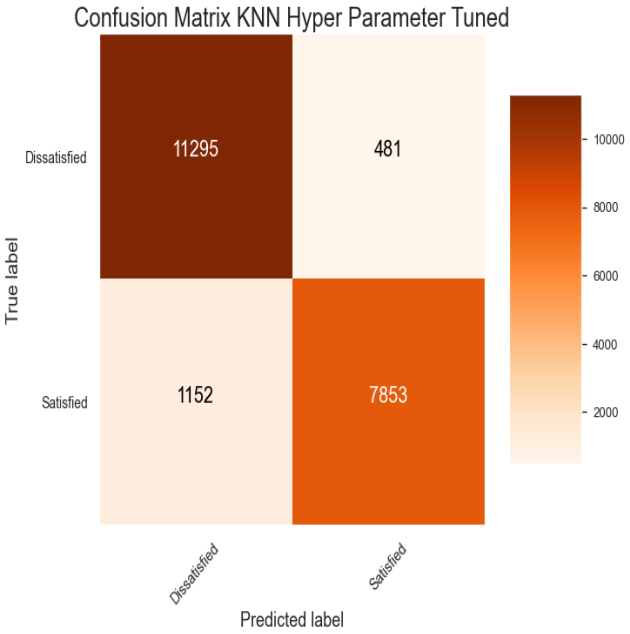

```
Misclassified Counts: 1295
accuracy........ 0.9377
precision score........ 0.9405
recall score........ 0.9330
classification_report
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     11776
           1       0.96      0.90      0.93      9005

    accuracy                           0.94     20781
   macro avg       0.94      0.93      0.94     20781
weighted avg       0.94      0.94      0.94     20781
```

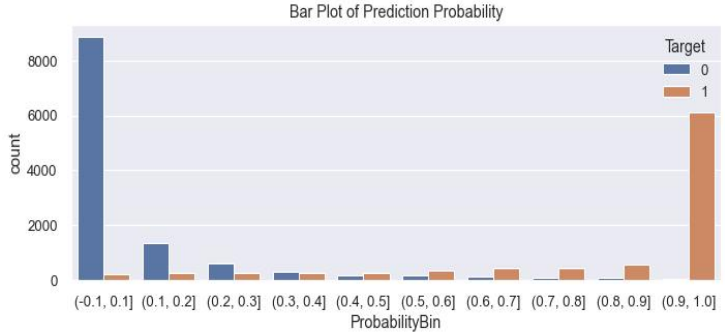
**KNN- PCA**

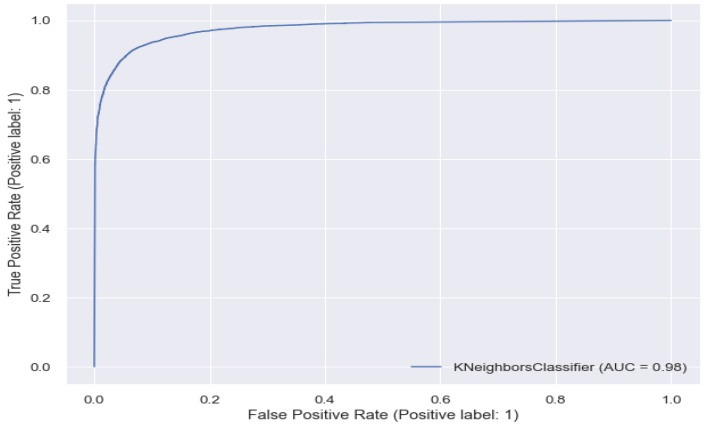


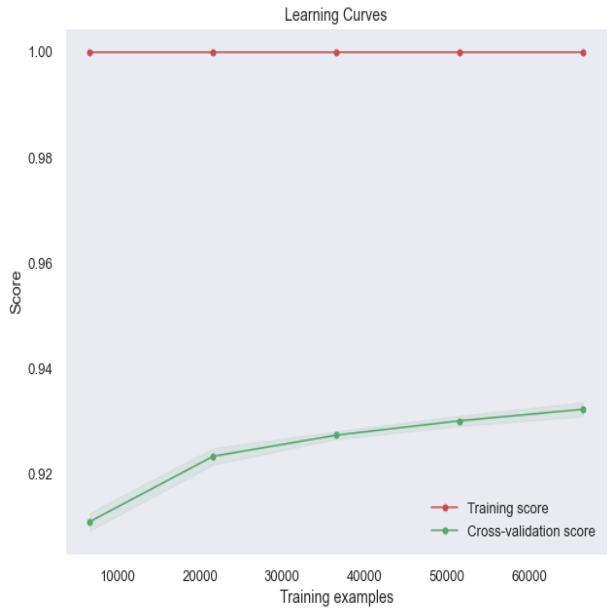



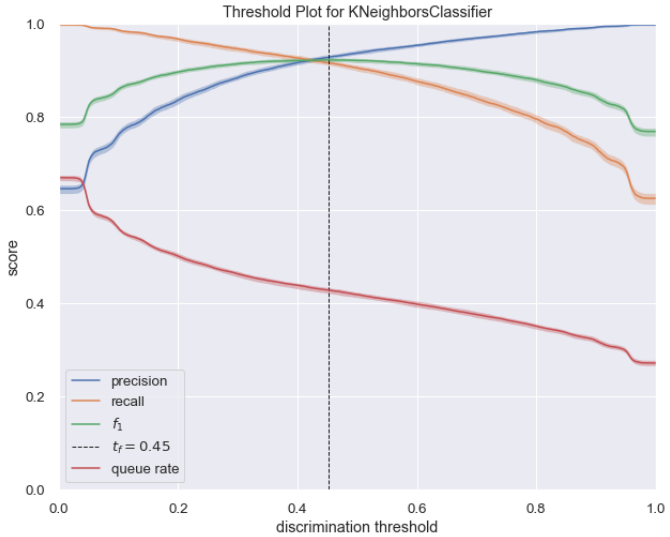



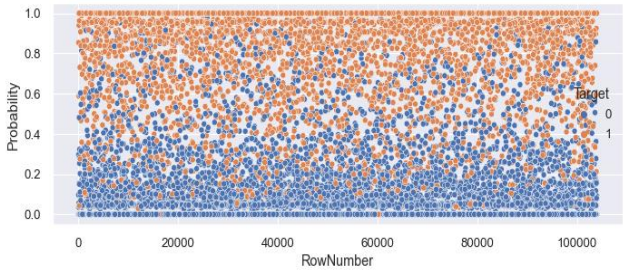



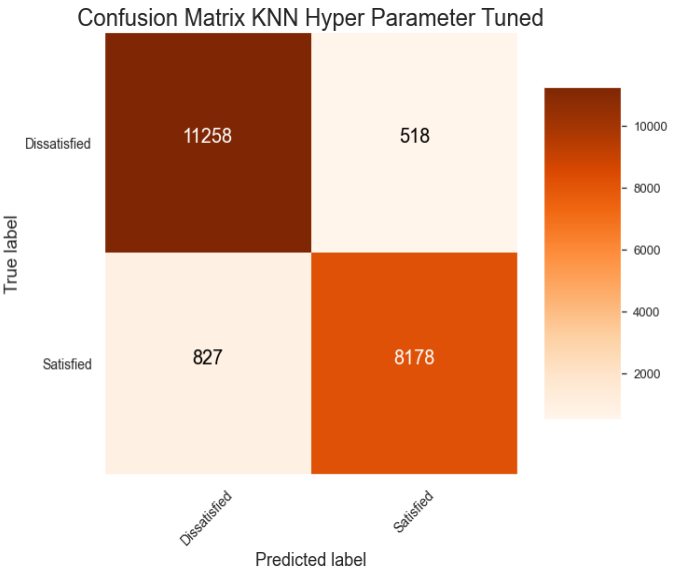

```
Misclassified Counts: 1633
accuracy........ 0.9214
precision score........ 0.9249
recall score........ 0.9156
classification_report
              precision    recall  f1-score   support

           0       0.91      0.96      0.93     11776
           1       0.94      0.87      0.91      9005

    accuracy                           0.92     20781
   macro avg       0.92      0.92      0.92     20781
weighted avg       0.92      0.92      0.92     20781
```

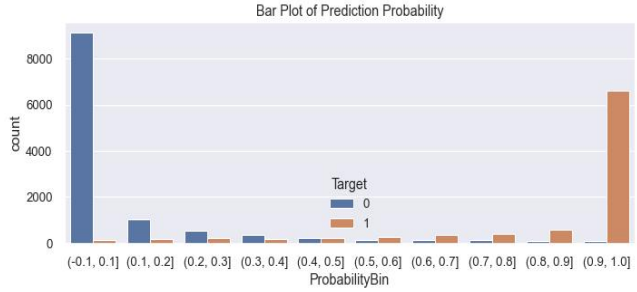
**KNN -VIF**

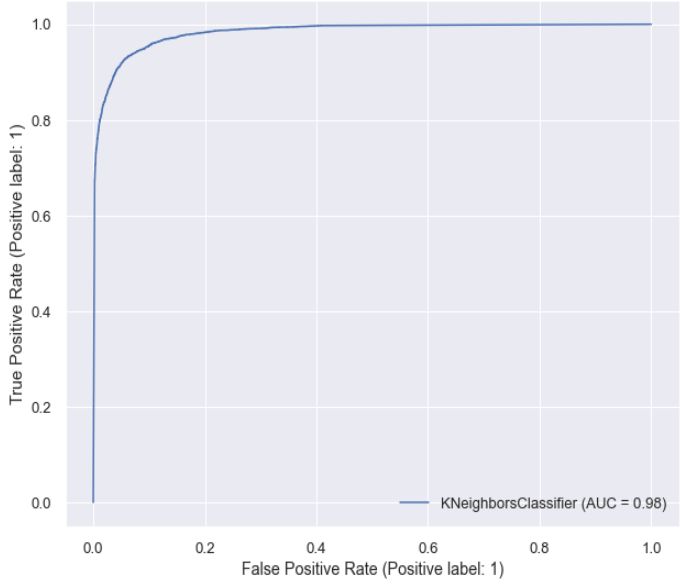


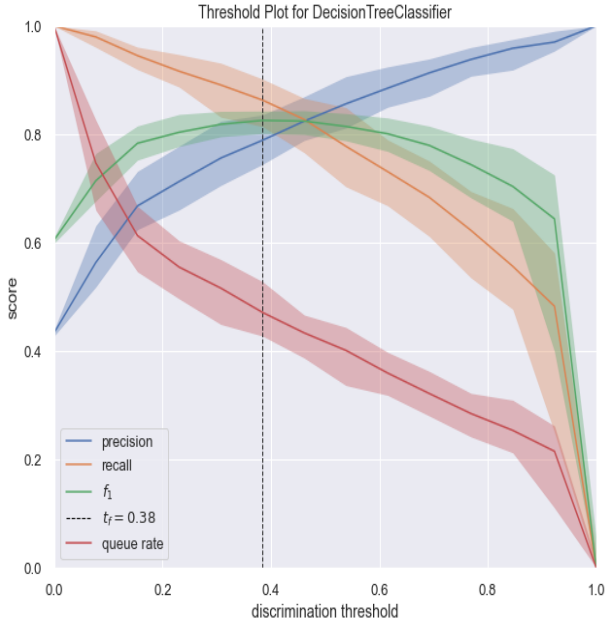



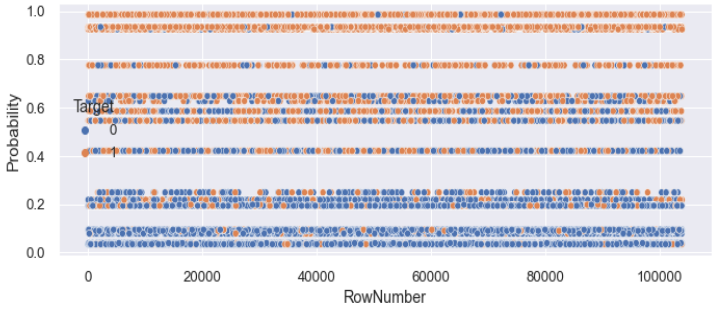



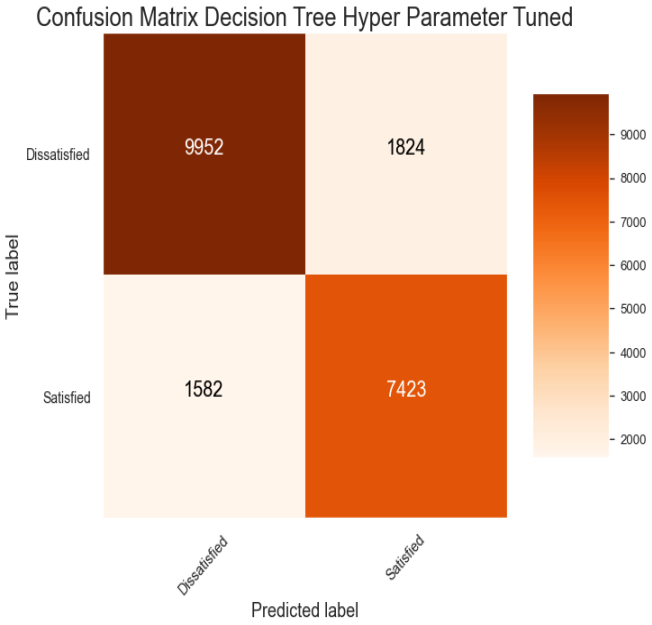



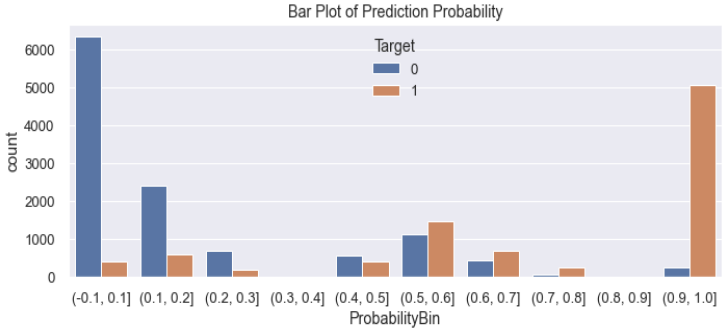

```
Misclassified Counts: 1345
accuracy........ 0.9353
precision score........ 0.9360
recall score........ 0.9321
classification_report
              precision    recall  f1-score   support

           0       0.93      0.96      0.94     11776
           1       0.94      0.91      0.92      9005

    accuracy                           0.94     20781
   macro avg       0.94      0.93      0.93     20781
weighted avg       0.94      0.94      0.94     20781
```

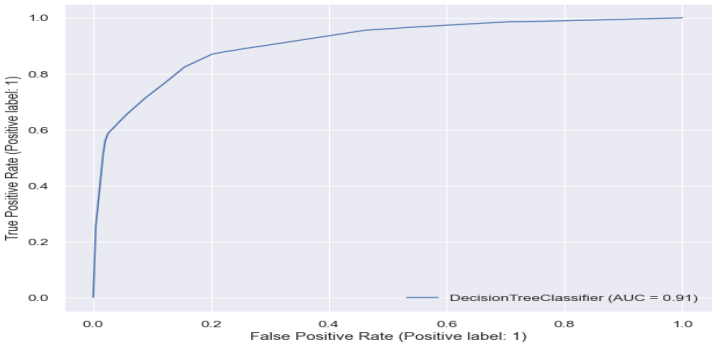
## Decision Tree - UnBinned Data

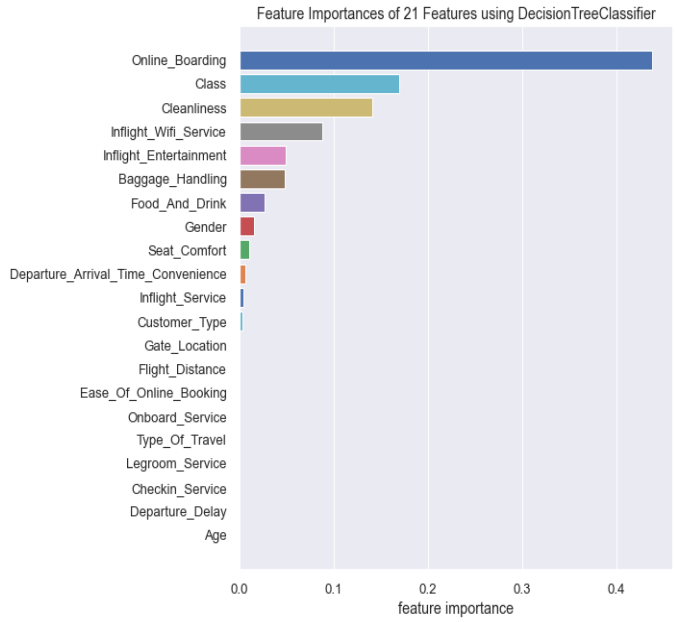



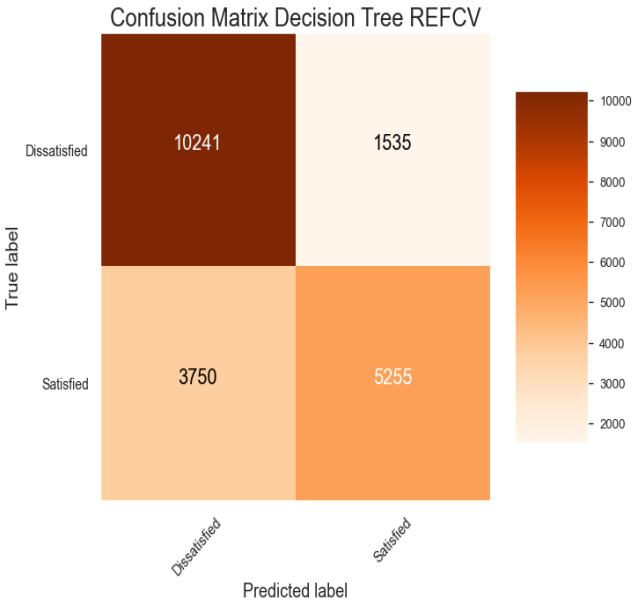



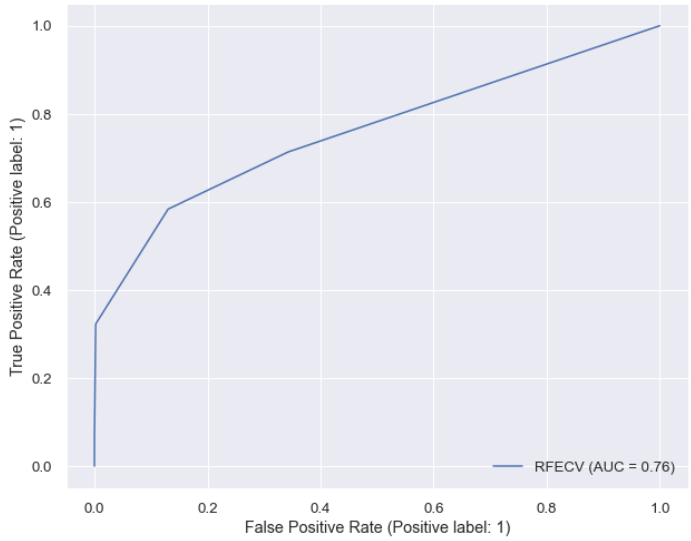




```
Misclassified Counts: 3406
accuracy........ 0.8361
precision score........ 0.8328
recall score........ 0.8347
classification_report
              precision    recall  f1-score   support

           0       0.86      0.85      0.85     11776
           1       0.80      0.82      0.81      9005

    accuracy                           0.84     20781
   macro avg       0.83      0.83      0.83     20781
weighted avg       0.84      0.84      0.84     20781
```

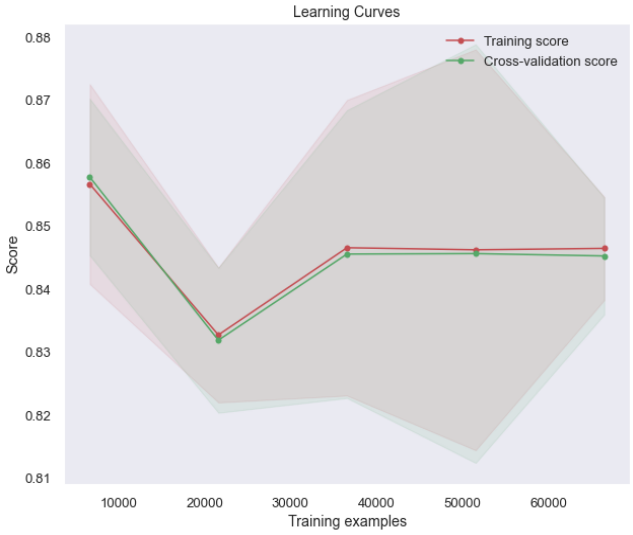
## Decision Tree - UnBinned Data Contd..

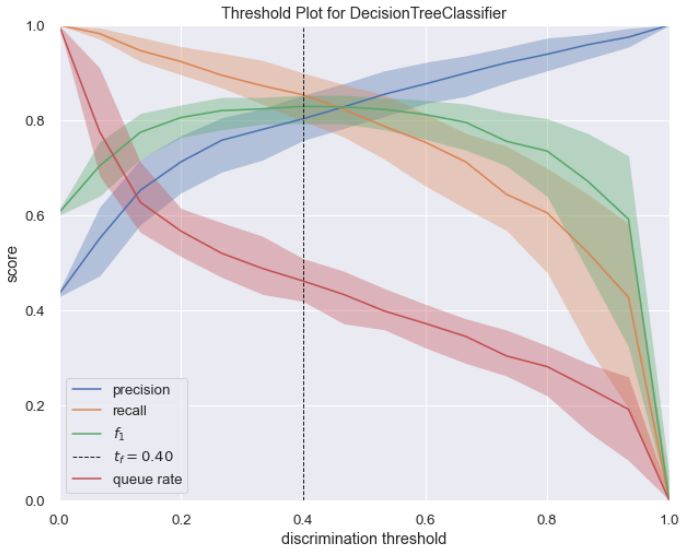


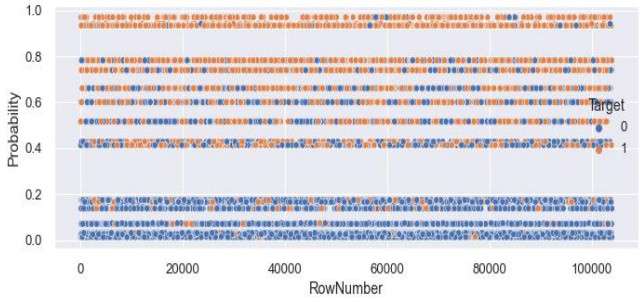



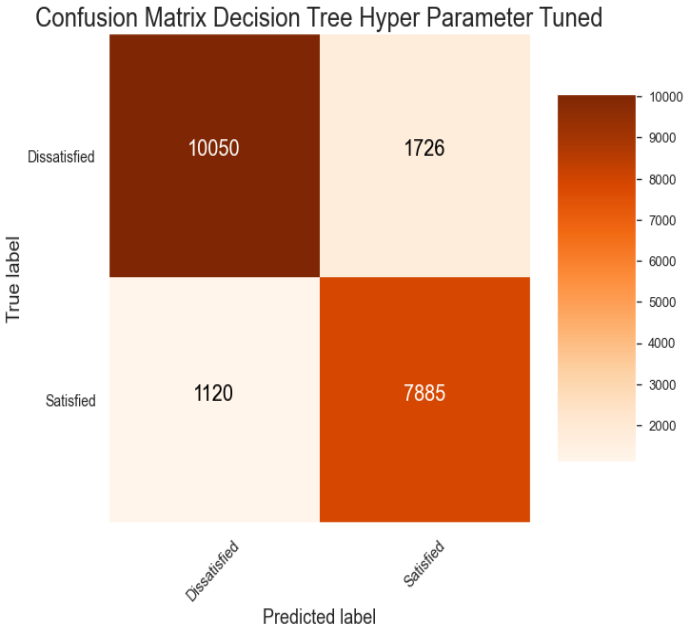



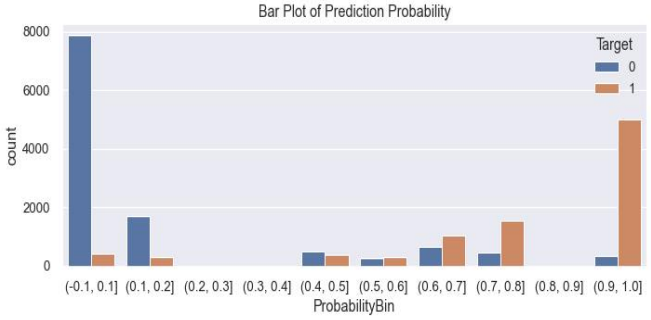

```
Misclassified Counts: 5285
accuracy........ 0.7457
precision score........ 0.7530
recall score........ 0.7266
classification_report
              precision    recall  f1-score   support

           0       0.73      0.87      0.79     11776
           1       0.77      0.58      0.67      9005

    accuracy                           0.75     20781
   macro avg       0.75      0.73      0.73     20781
weighted avg       0.75      0.75      0.74     20781
```

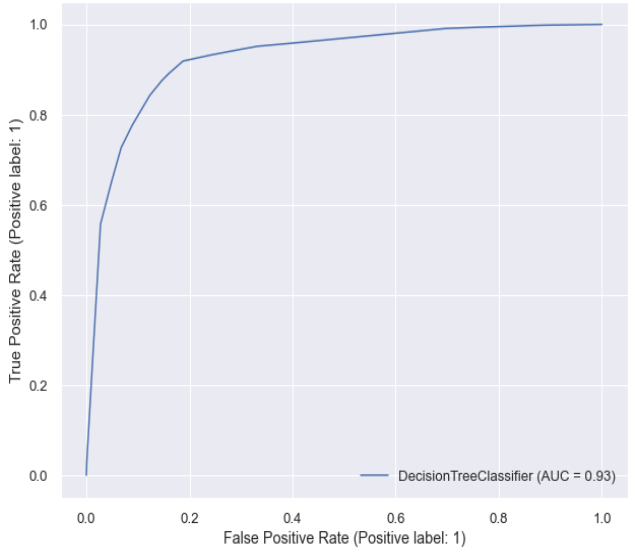


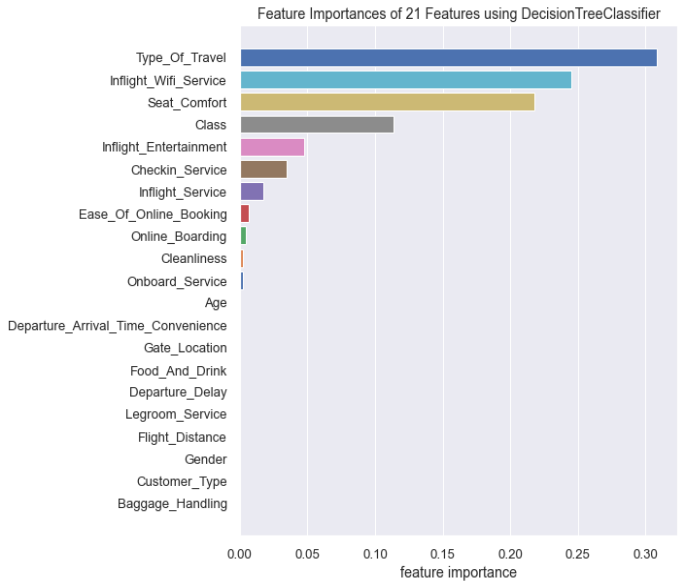



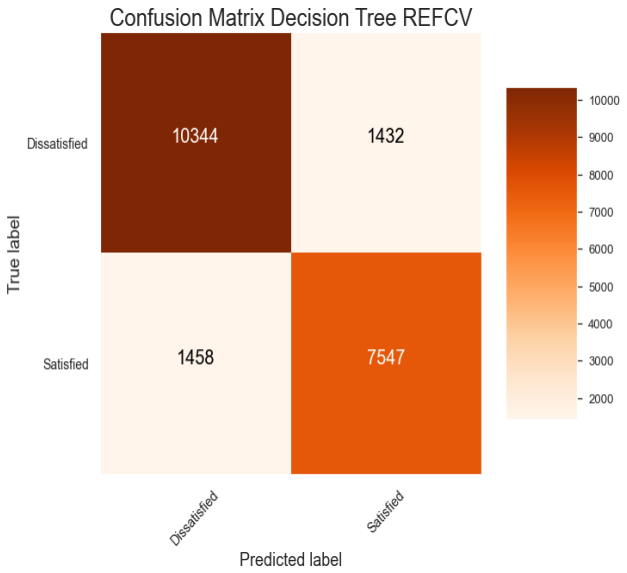



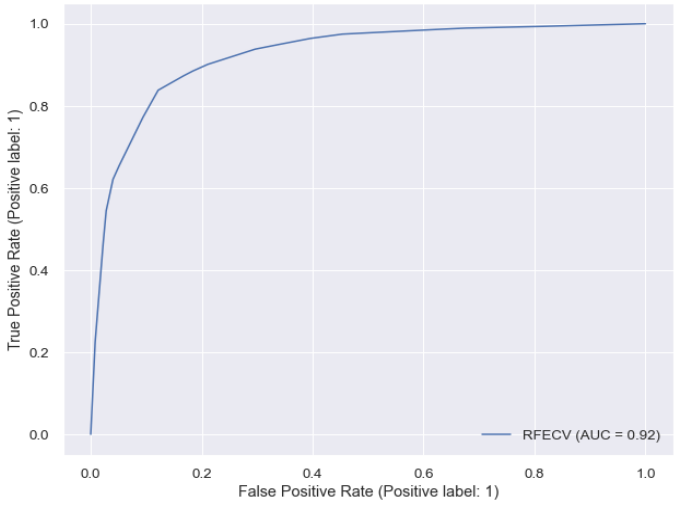



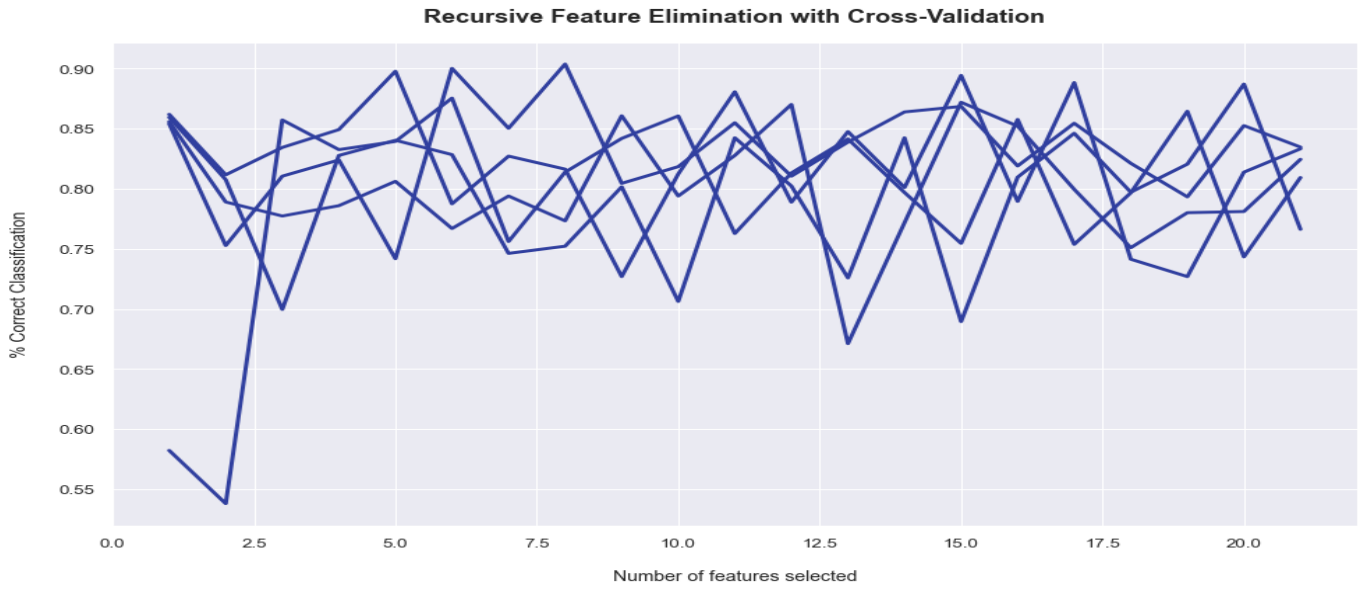

```
Misclassified Counts: 2846
accuracy........ 0.8630
precision score........ 0.8601
recall score........ 0.8645
classification_report
              precision    recall  f1-score   support

           0       0.90      0.85      0.88     11776
           1       0.82      0.88      0.85      9005

    accuracy                           0.86     20781
   macro avg       0.86      0.86      0.86     20781
weighted avg       0.87      0.86      0.86     20781
```

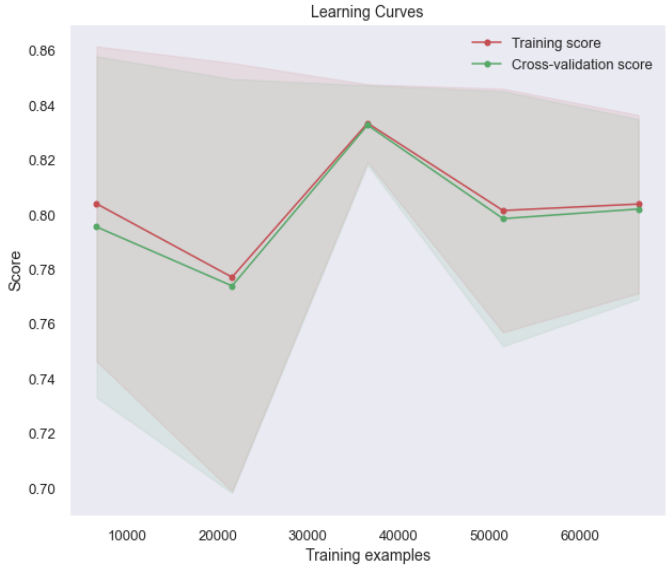


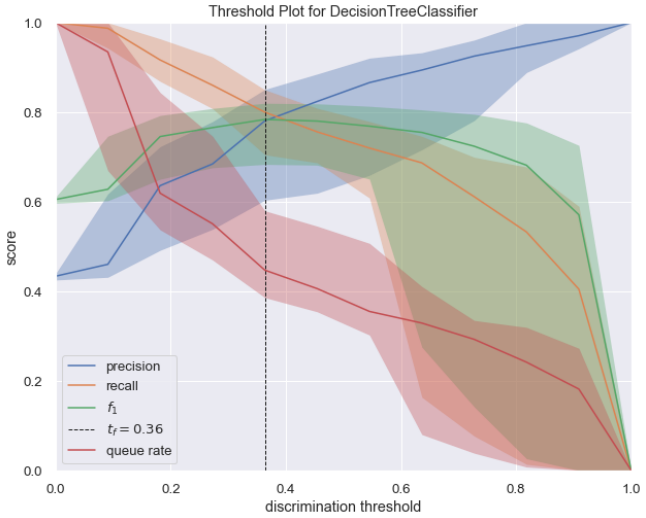



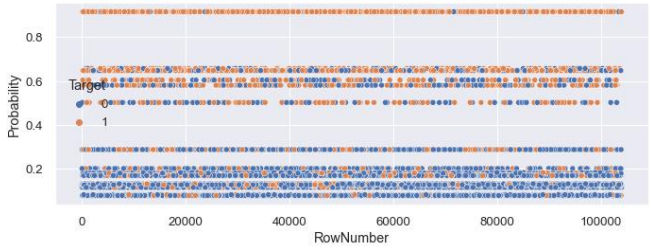



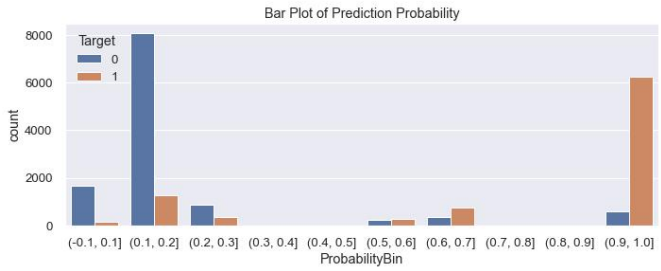

```
Misclassified Counts: 2890
accuracy........ 0.8609
precision score........ 0.8585
recall score........ 0.8582
classification_report
              precision    recall  f1-score   support

           0       0.88      0.88      0.88     11776
           1       0.84      0.84      0.84      9005

    accuracy                           0.86     20781
   macro avg       0.86      0.86      0.86     20781
weighted avg       0.86      0.86      0.86     20781
```

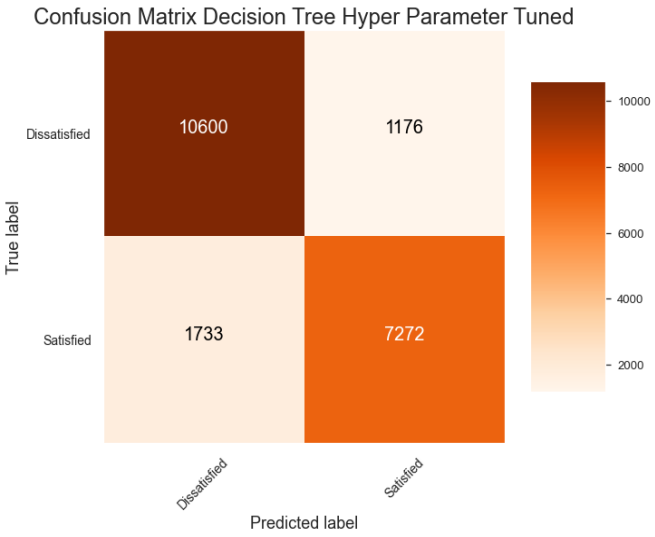


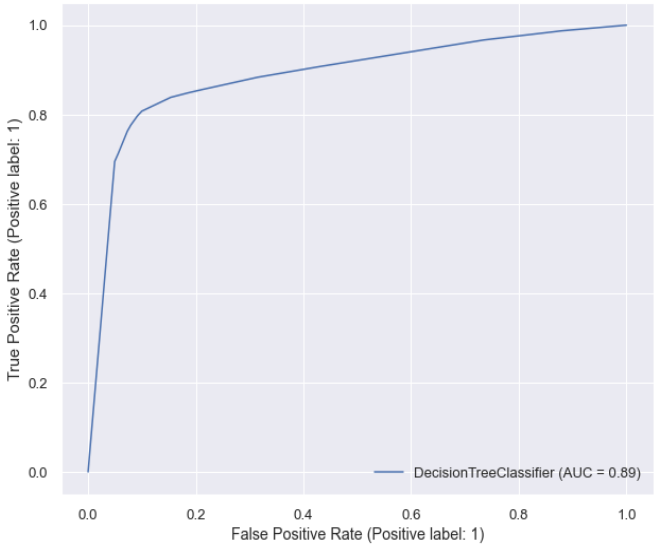


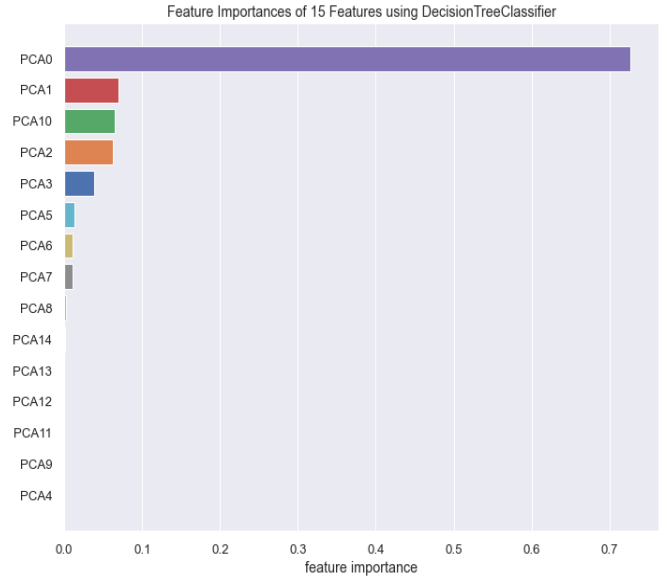



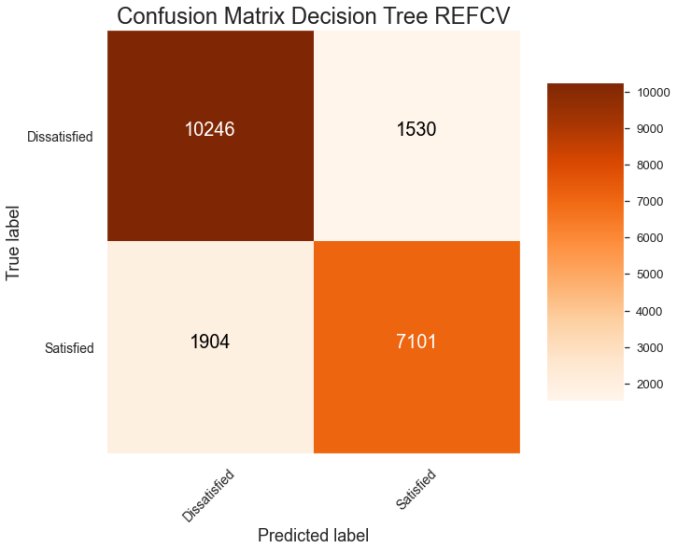



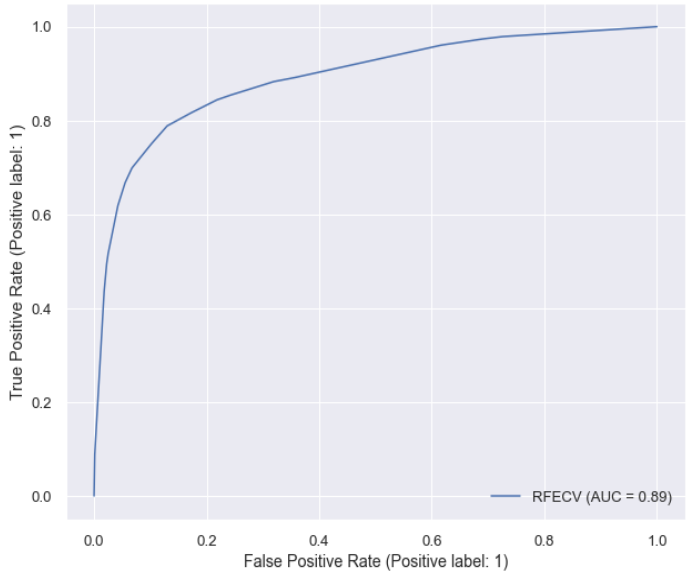



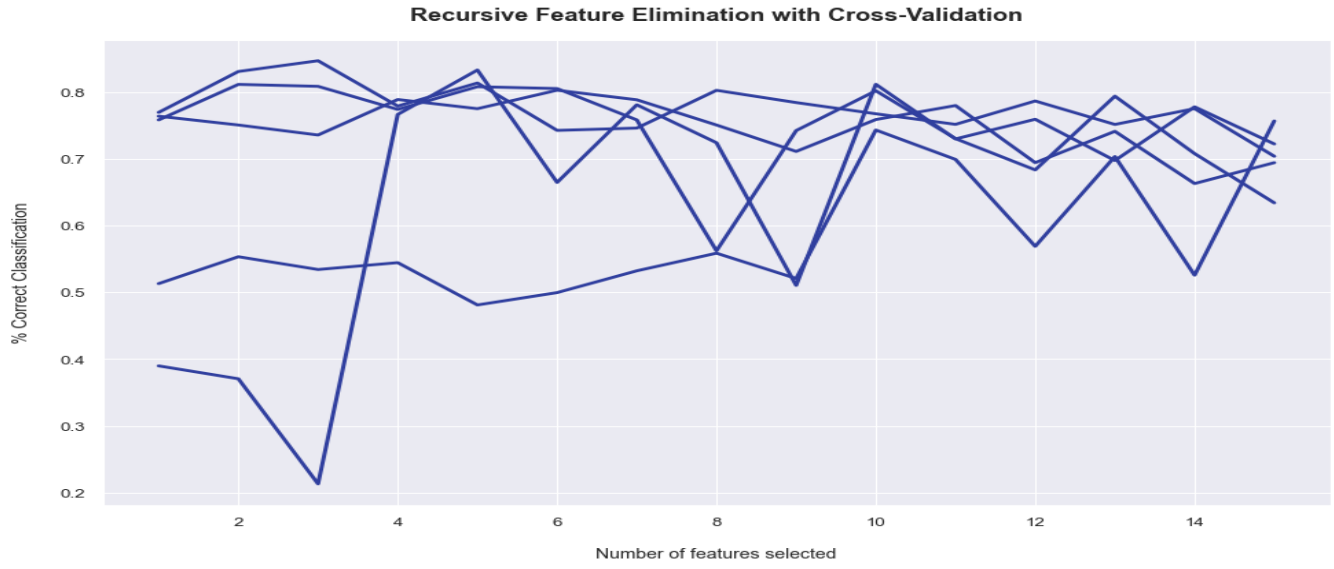

```
Misclassified Counts: 2909
accuracy........ 0.8600
precision score........ 0.8601
recall score........ 0.8538
classification_report
              precision    recall  f1-score   support

           0       0.86      0.90      0.88     11776
           1       0.86      0.81      0.83      9005

    accuracy                           0.86     20781
   macro avg       0.86      0.85      0.86     20781
weighted avg       0.86      0.86      0.86     20781
```

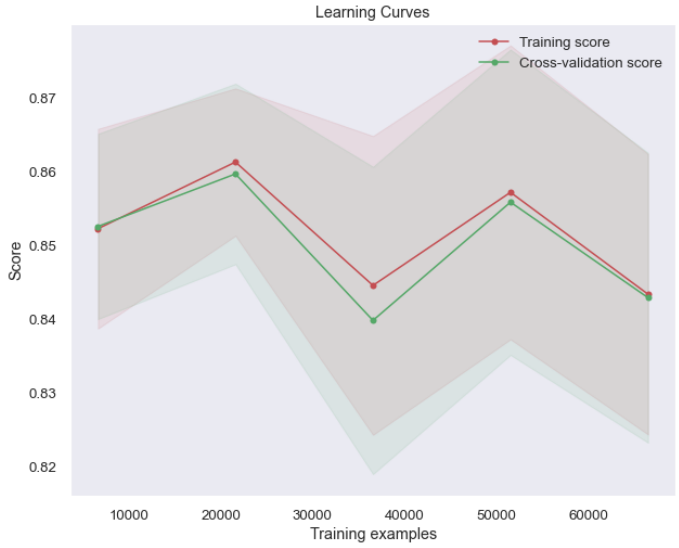


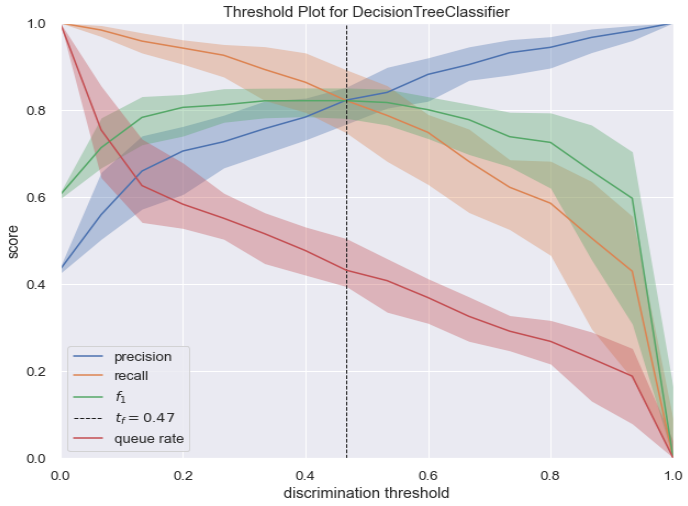



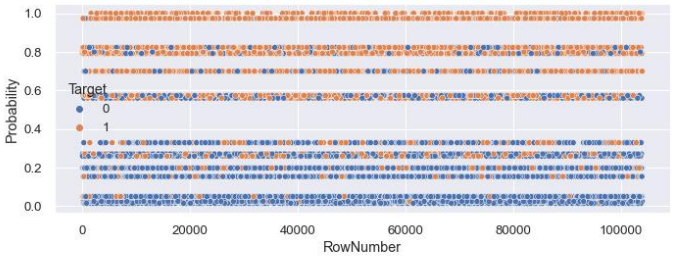



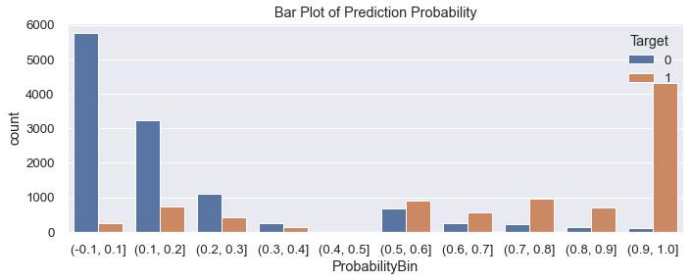

```
Misclassified Counts: 3434
accuracy........ 0.8348
precision score........ 0.8330
recall score........ 0.8293
classification_report
              precision    recall  f1-score   support

           0       0.84      0.87      0.86     11776
           1       0.82      0.79      0.81      9005

    accuracy                           0.83     20781
   macro avg       0.83      0.83      0.83     20781
weighted avg       0.83      0.83      0.83     20781
```

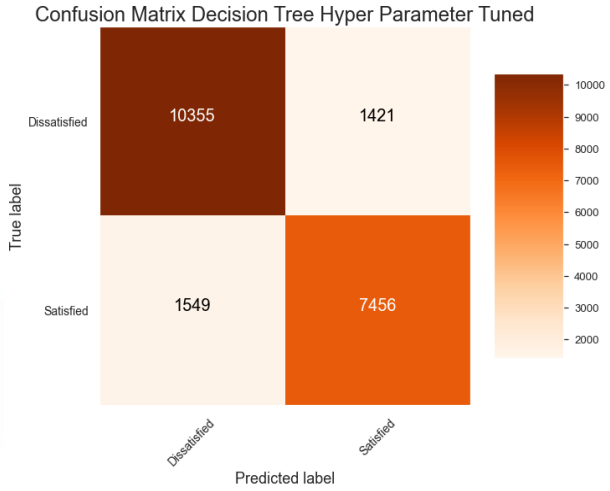


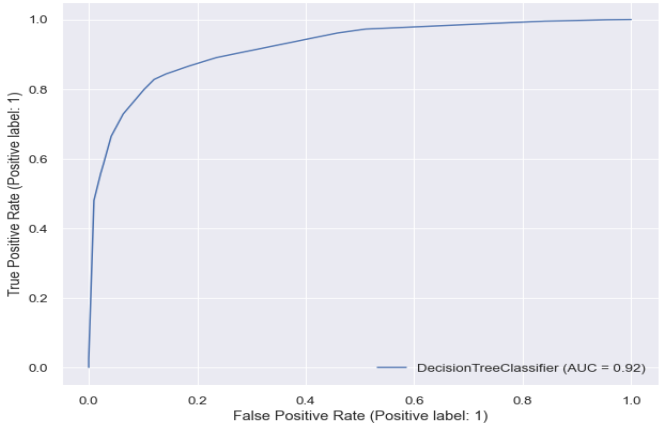


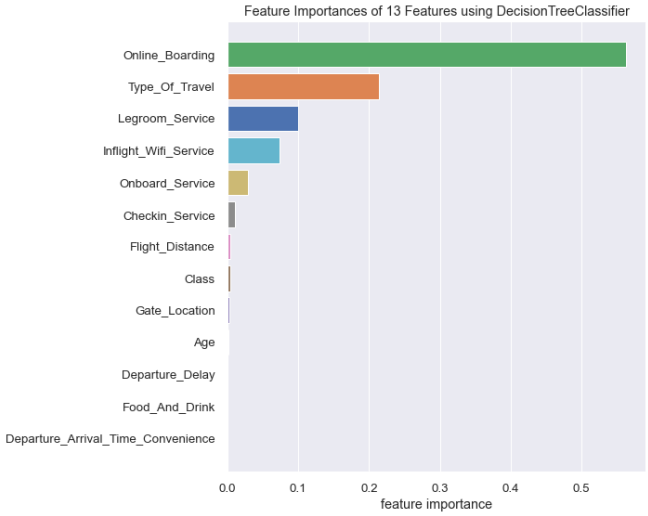



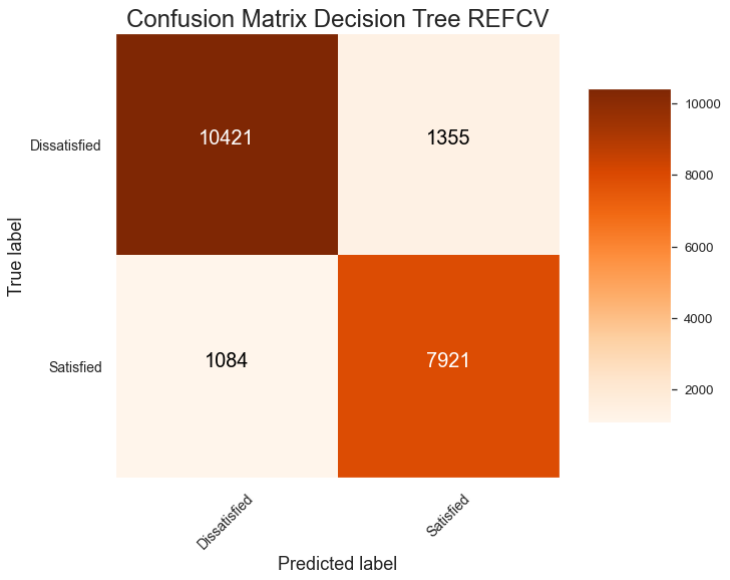



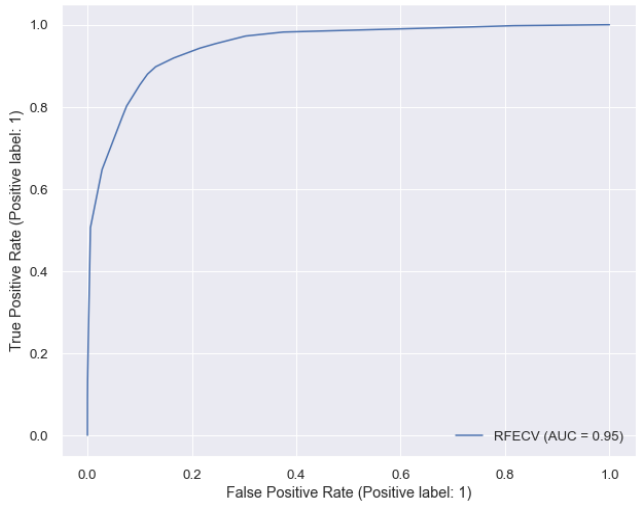




```
Misclassified Counts: 2970
accuracy........ 0.8571
precision score........ 0.8549
recall score........ 0.8537
classification_report
              precision    recall  f1-score   support

           0       0.87      0.88      0.87     11776
           1       0.84      0.83      0.83      9005

    accuracy                           0.86     20781
   macro avg       0.85      0.85      0.85     20781
weighted avg       0.86      0.86      0.86     20781
```

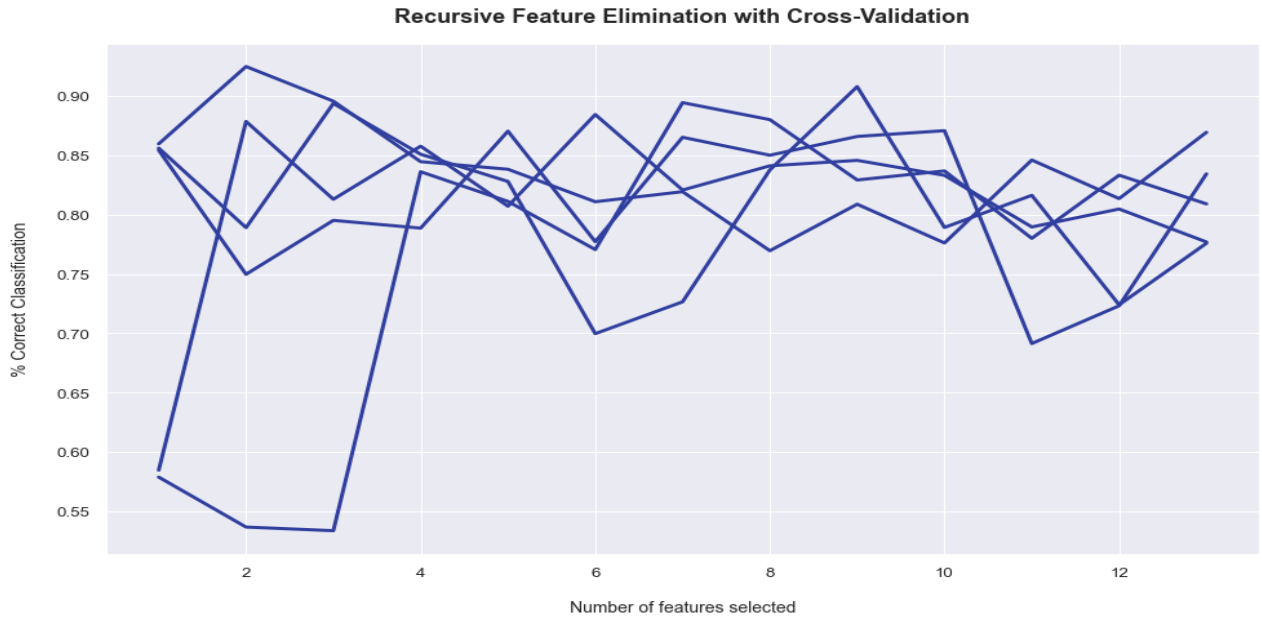
## Decision Tree - VIF Contd..



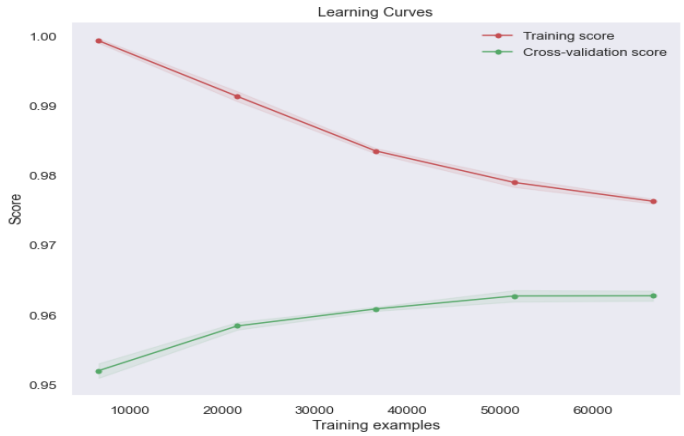



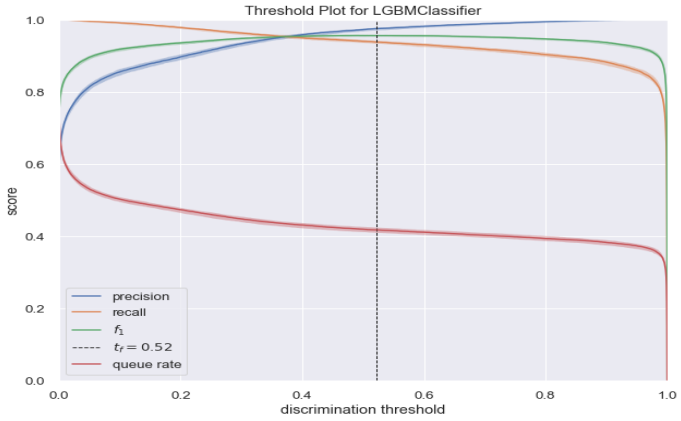



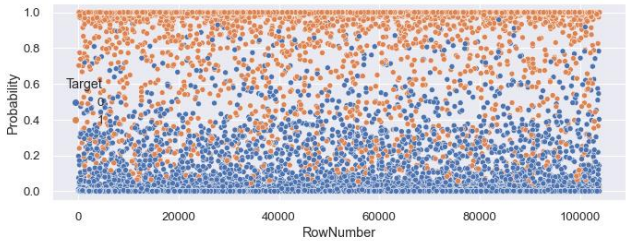

```
Misclassified Counts: 2439
accuracy........ 0.8826
precision score........ 0.8799
recall score........ 0.8823
classification_report
              precision    recall  f1-score   support

           0       0.91      0.88      0.90     11776
           1       0.85      0.88      0.87      9005

    accuracy                           0.88     20781
   macro avg       0.88      0.88      0.88     20781
weighted avg       0.88      0.88      0.88     20781
```

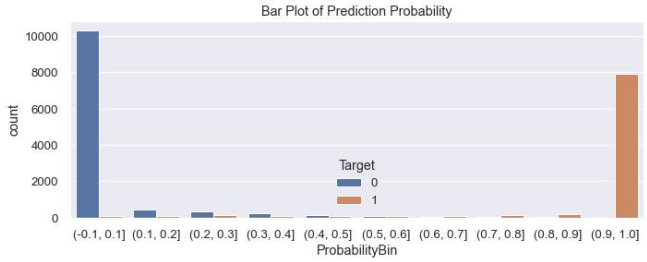


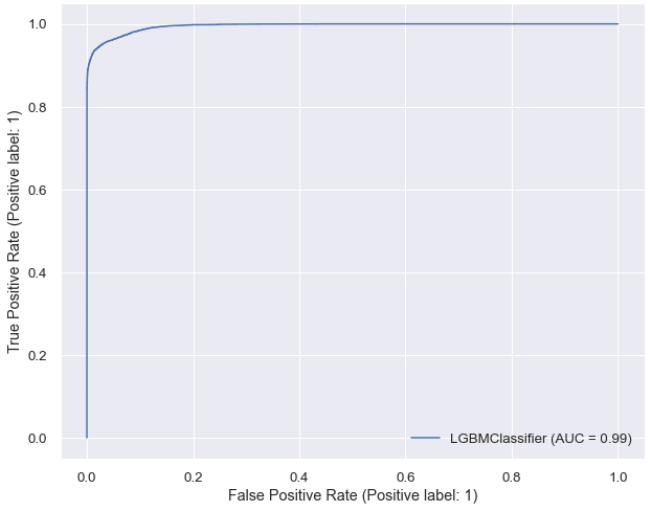
## LightGradientBoosting UnBinned Data

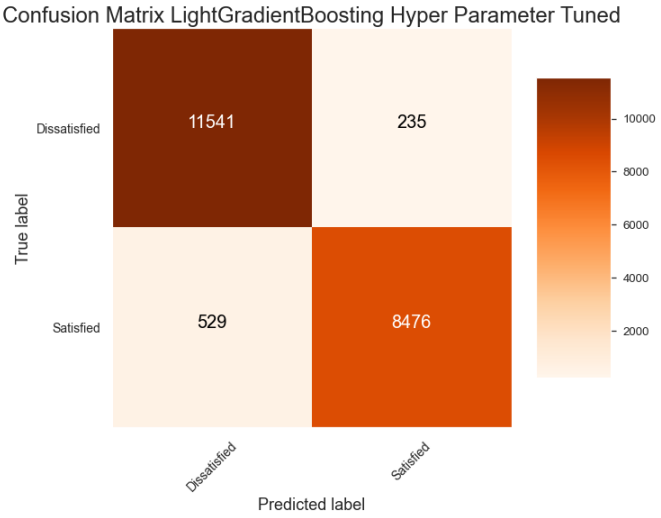


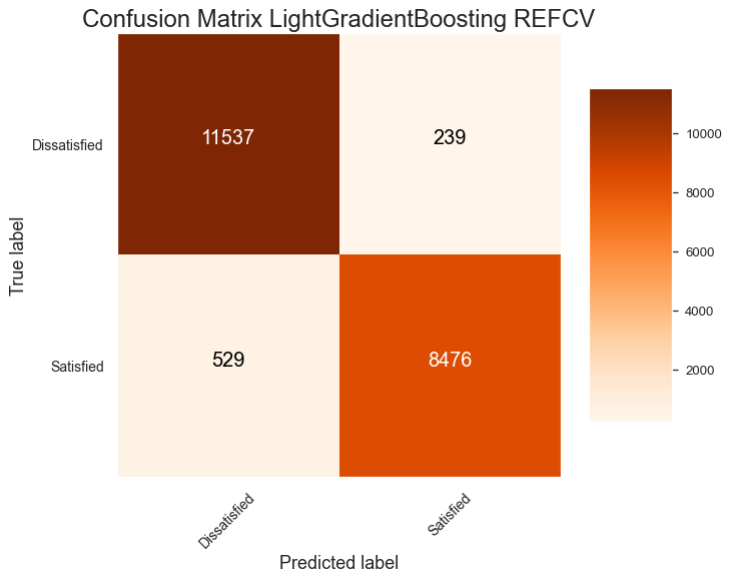



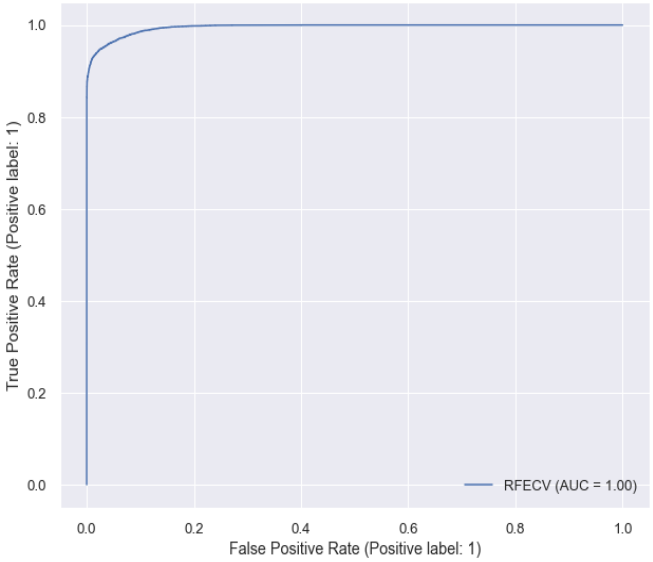



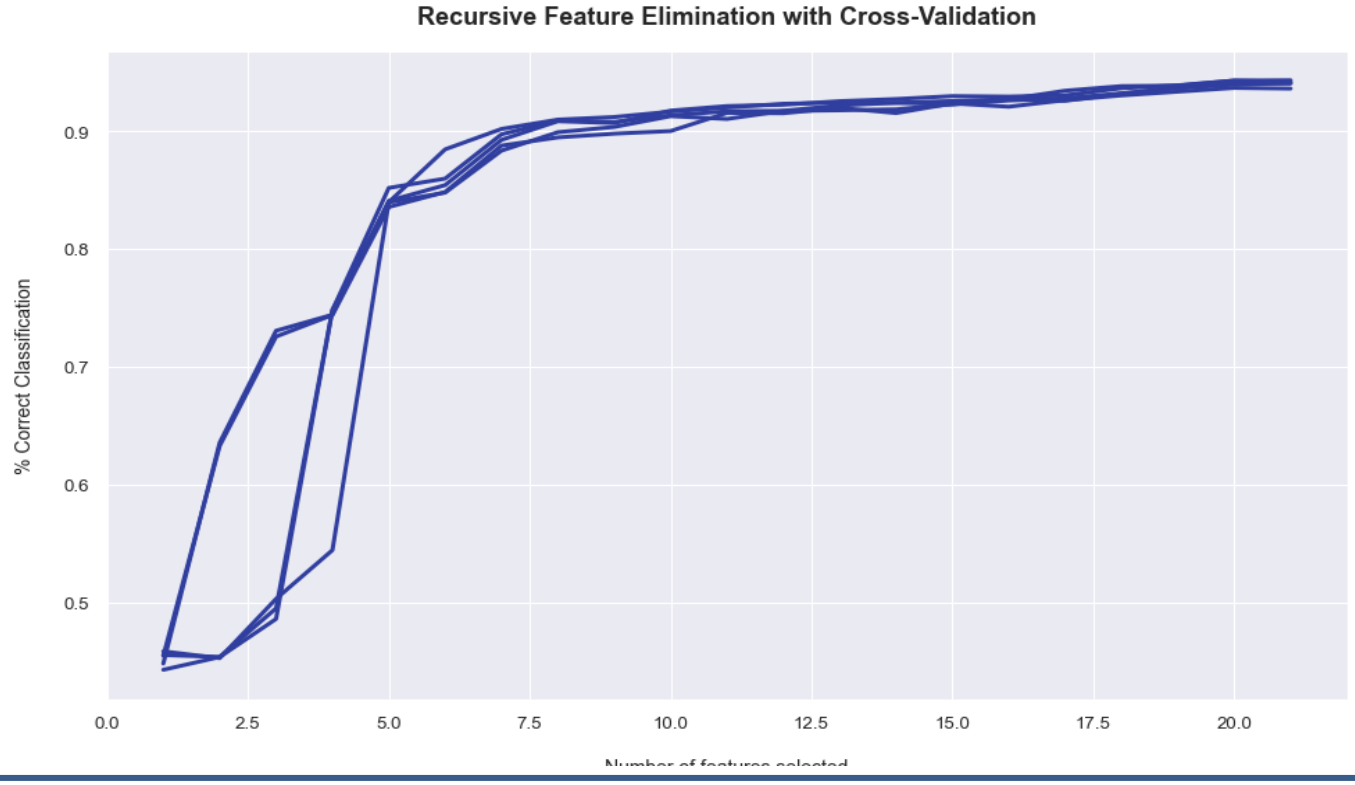



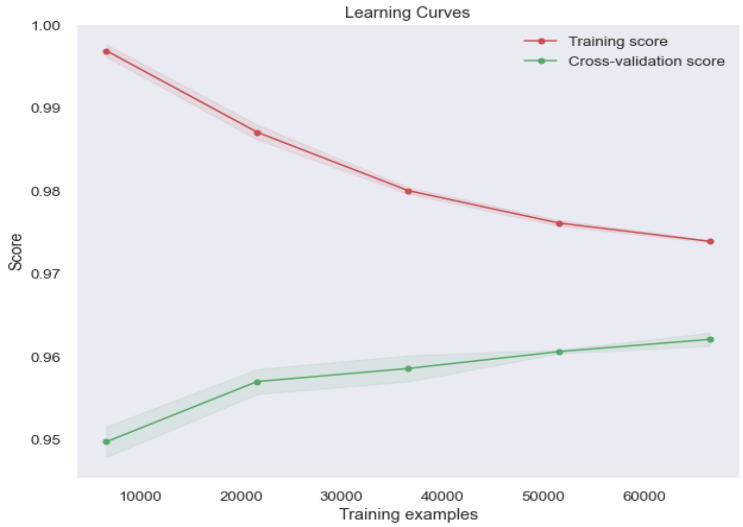

```
Misclassified Counts: 764
accuracy........ 0.9632
precision score........ 0.9646
recall score........ 0.9606
classification_report
              precision    recall  f1-score   support

           0       0.96      0.98      0.97     11776
           1       0.97      0.94      0.96      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

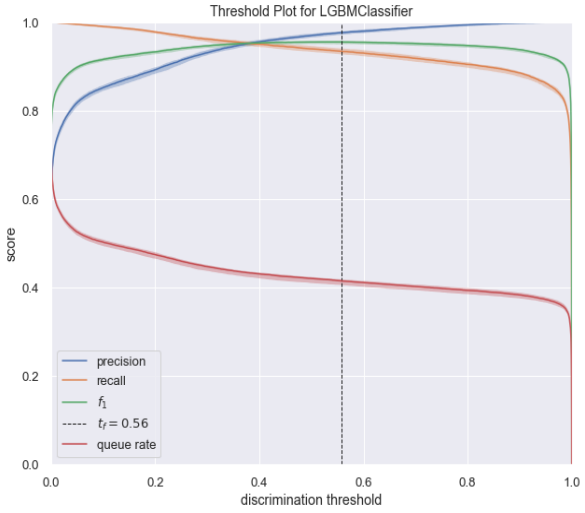


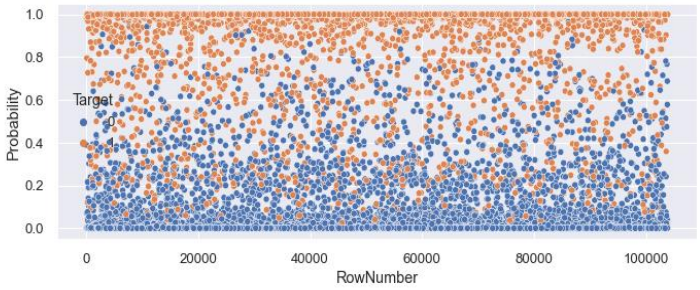



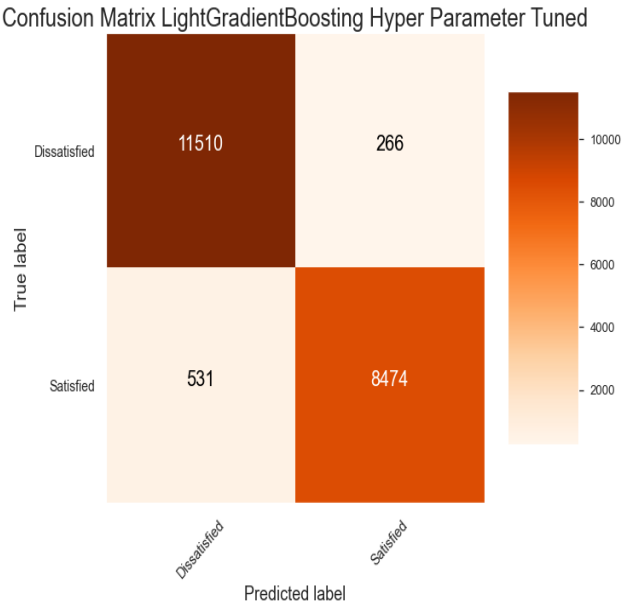

```
Misclassified Counts: 768
accuracy........ 0.9630
precision score........ 0.9644
recall score........ 0.9605
classification_report
                precision    recall  f1-score   support

           0       0.96      0.98      0.97     11776
           1       0.97      0.94      0.96      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

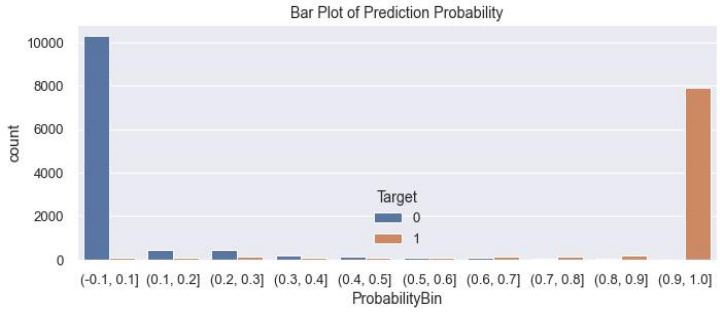


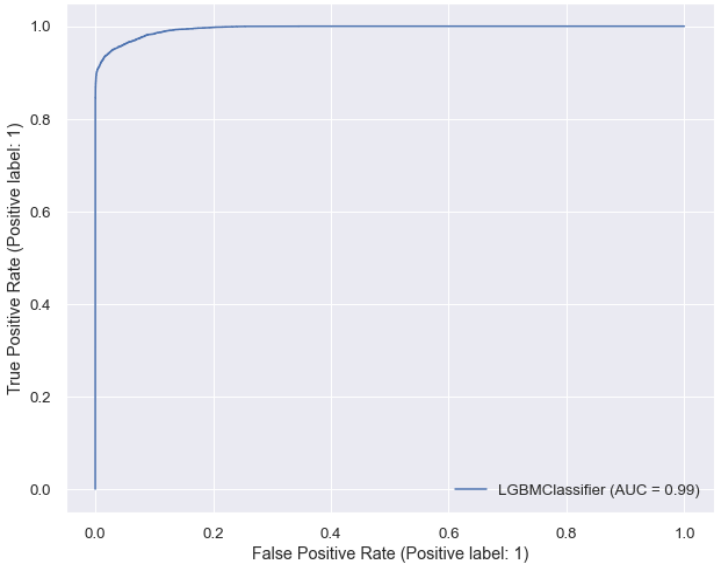


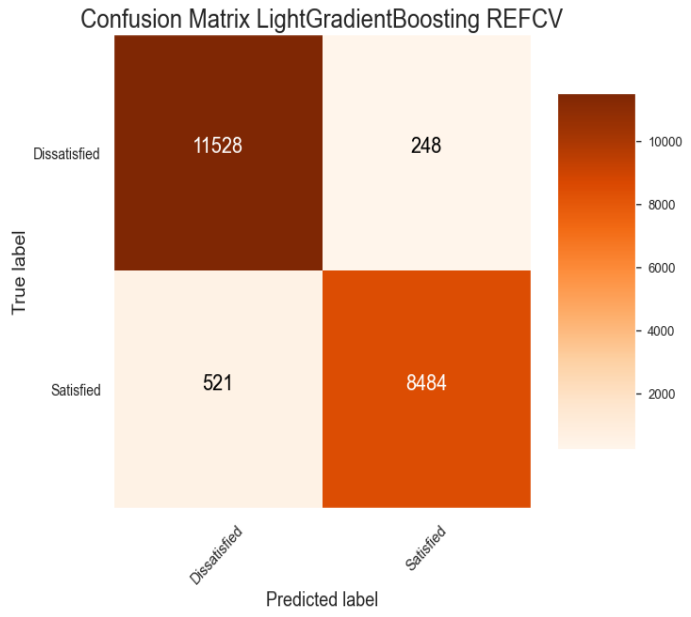



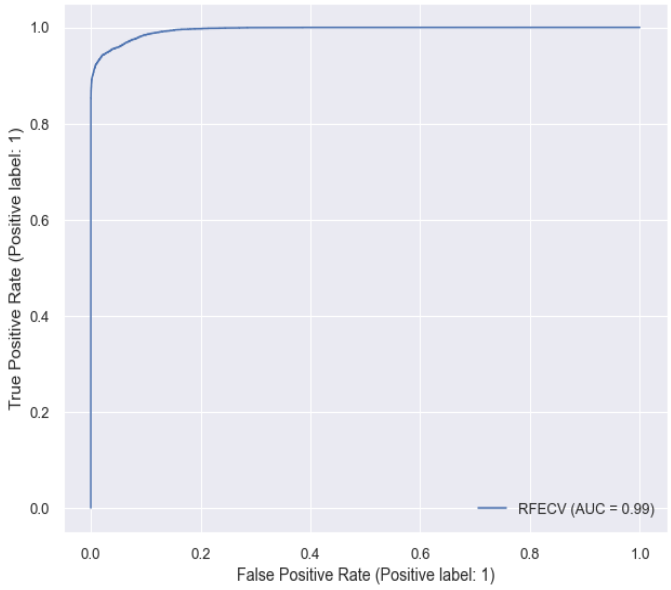




```
Misclassified Counts: 797
accuracy........ 0.9616
precision score........ 0.9627
recall score........ 0.9592
classification_report
              precision    recall  f1-score   support

           0       0.96      0.98      0.97     11776
           1       0.97      0.94      0.96      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

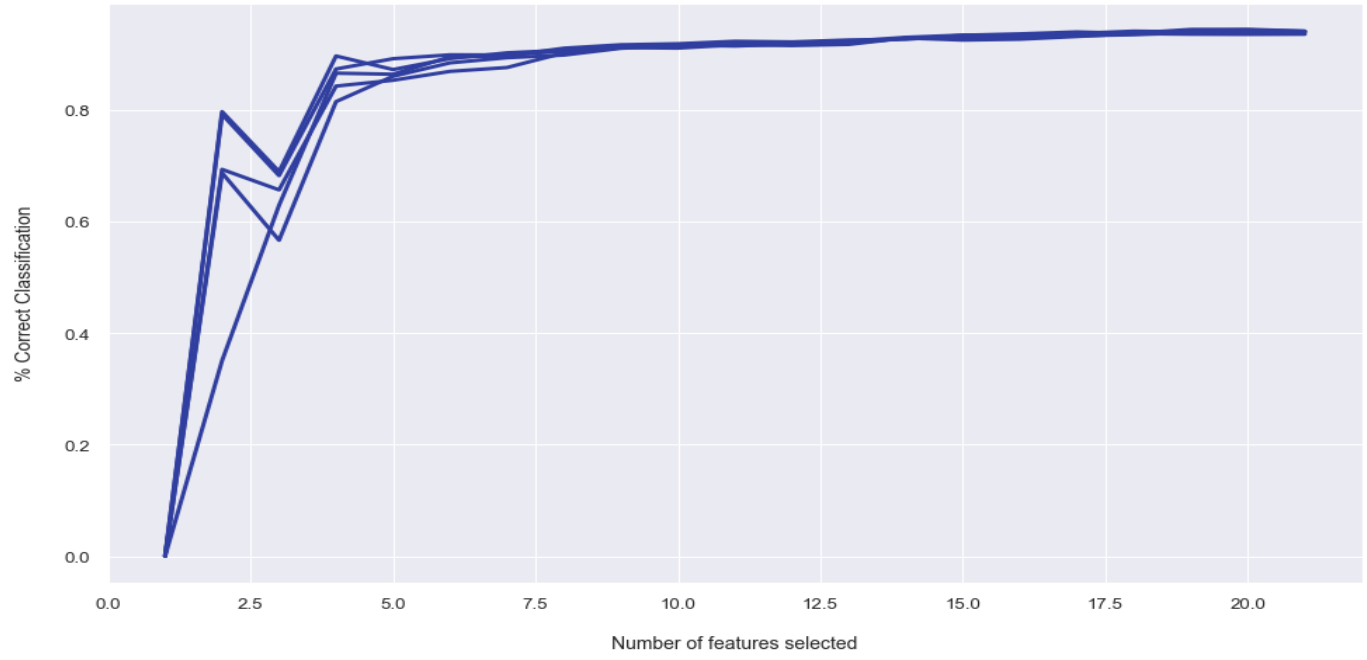
## Light Gradient Boosting- Binned Data Contd..

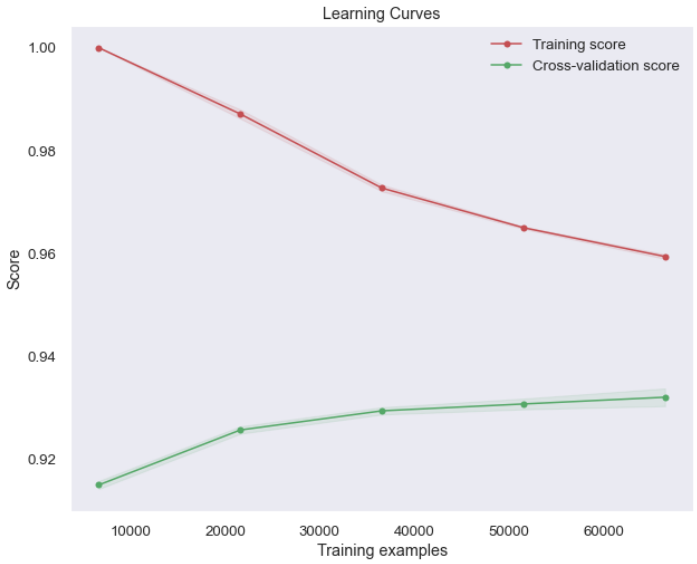


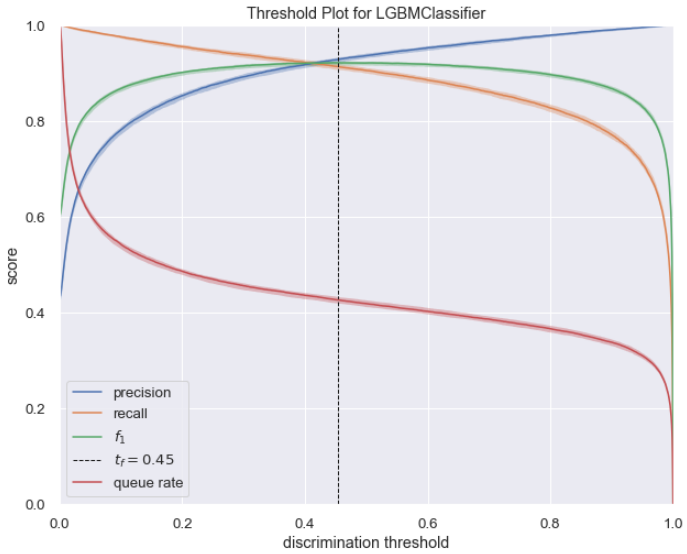



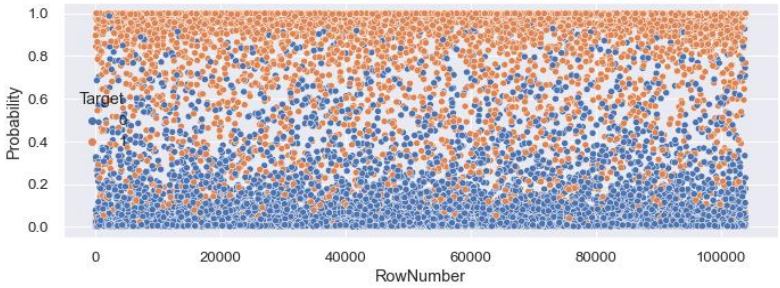

```
Misclassified Counts: 769
accuracy........ 0.9630
precision score........ 0.9642
recall score........ 0.9605
classification_report
                precision    recall  f1-score   support

           0       0.96      0.98      0.97     11776
           1       0.97      0.94      0.96      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

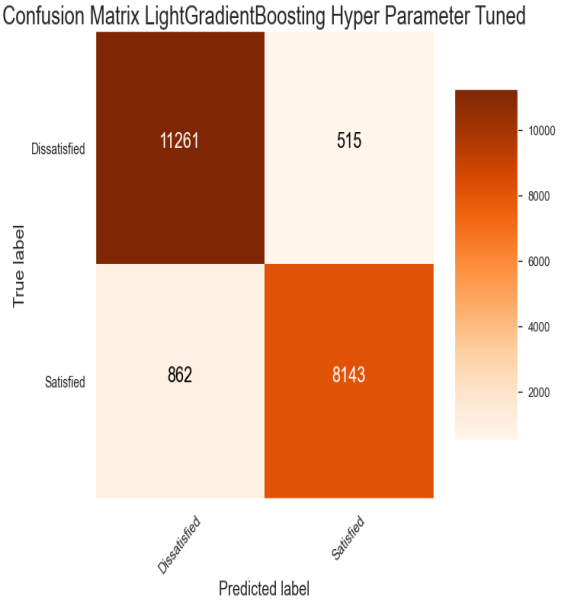
**Recursive Feature Elimination with Cross-Validation**

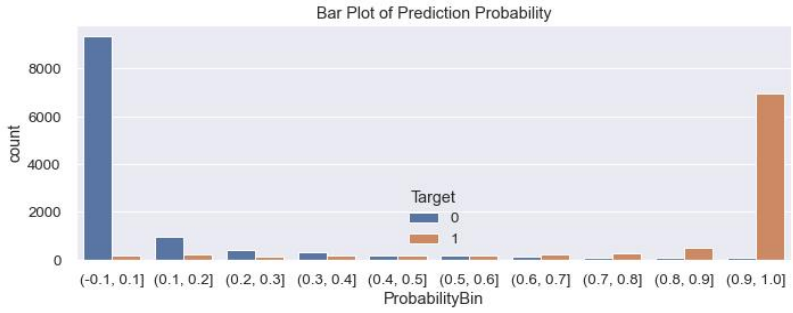


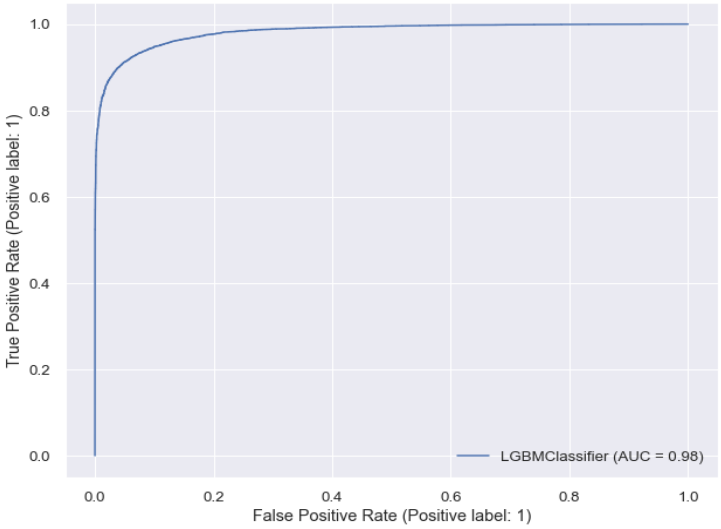


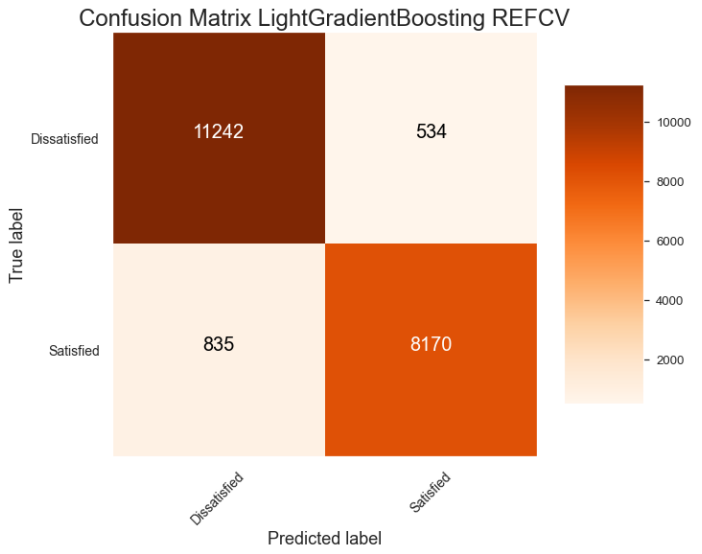



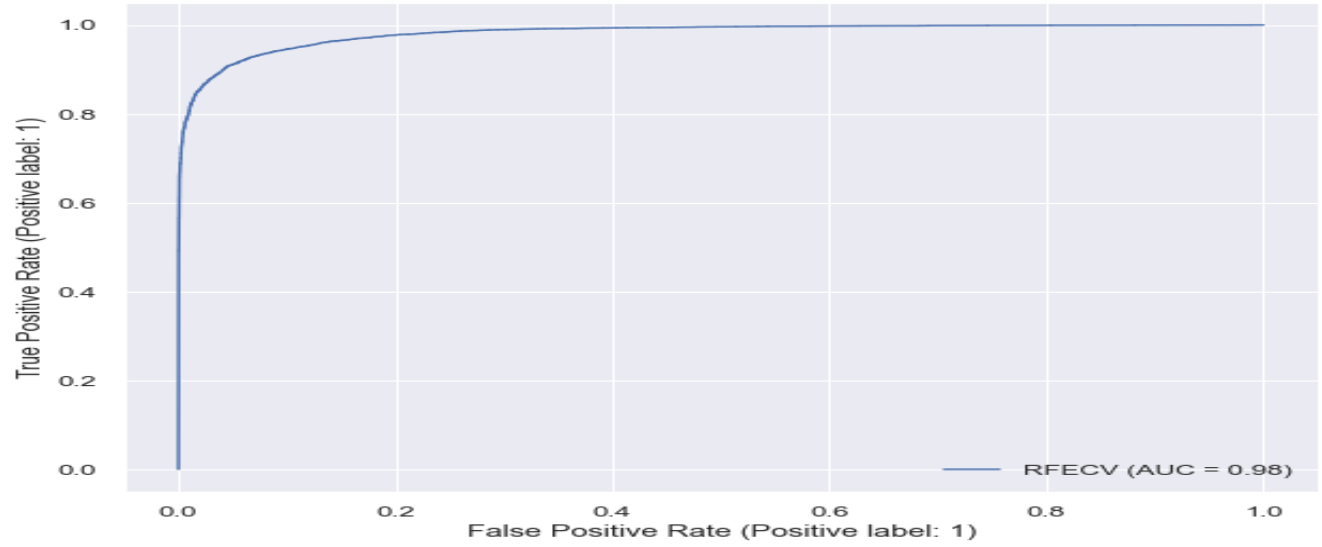



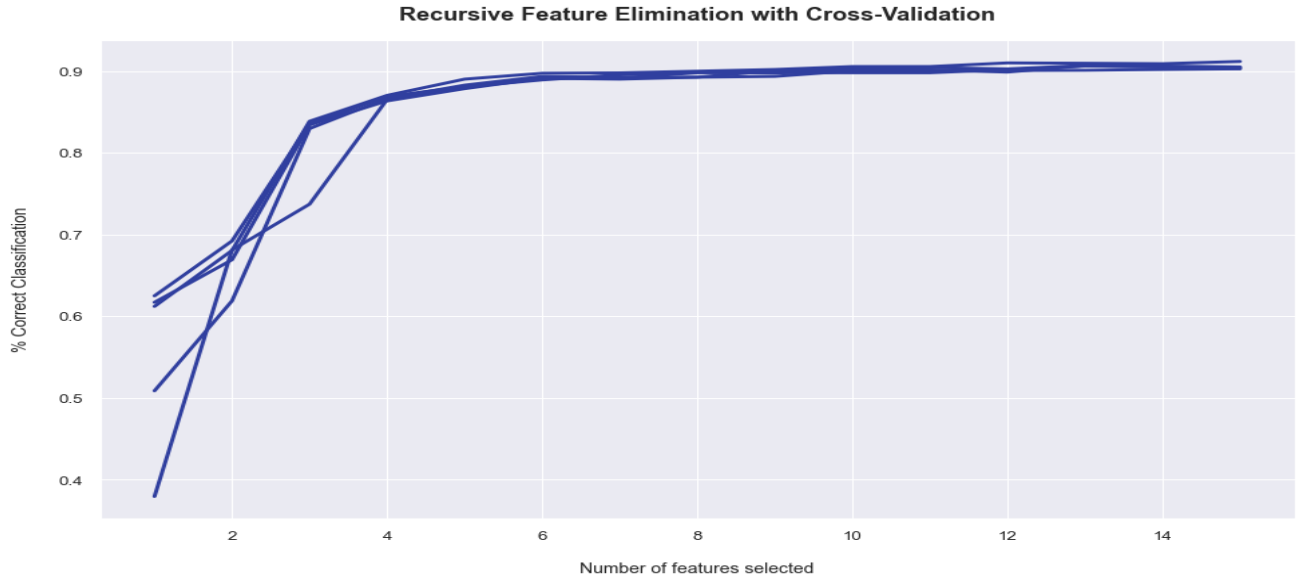



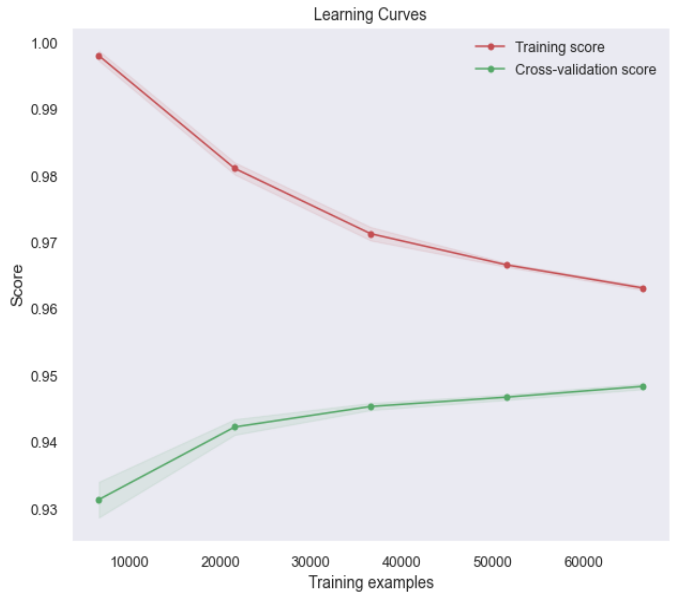

```
Misclassified Counts: 1377
accuracy........ 0.9337
precision score........ 0.9347
recall score........ 0.9303
classification_report
              precision    recall  f1-score   support

           0       0.93      0.96      0.94     11776
           1       0.94      0.90      0.92      9005

    accuracy                           0.93     20781
   macro avg       0.93      0.93      0.93     20781
weighted avg       0.93      0.93      0.93     20781
```

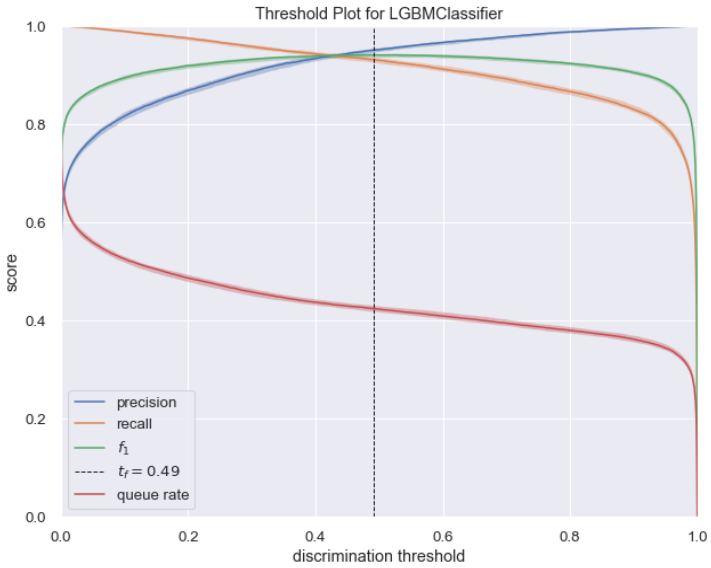


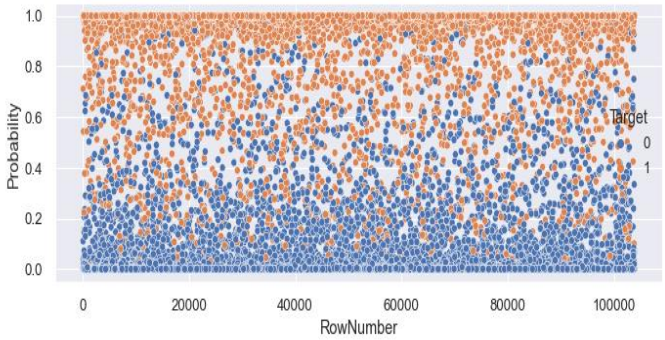

```
Misclassified Counts: 1369
accuracy........ 0.9341
precision score........ 0.9348
recall score........ 0.9310
classification_report
              precision    recall  f1-score   support

           0       0.93      0.95      0.94     11776
           1       0.94      0.91      0.92      9005

    accuracy                           0.93     20781
   macro avg       0.93      0.93      0.93     20781
weighted avg       0.93      0.93      0.93     20781
```

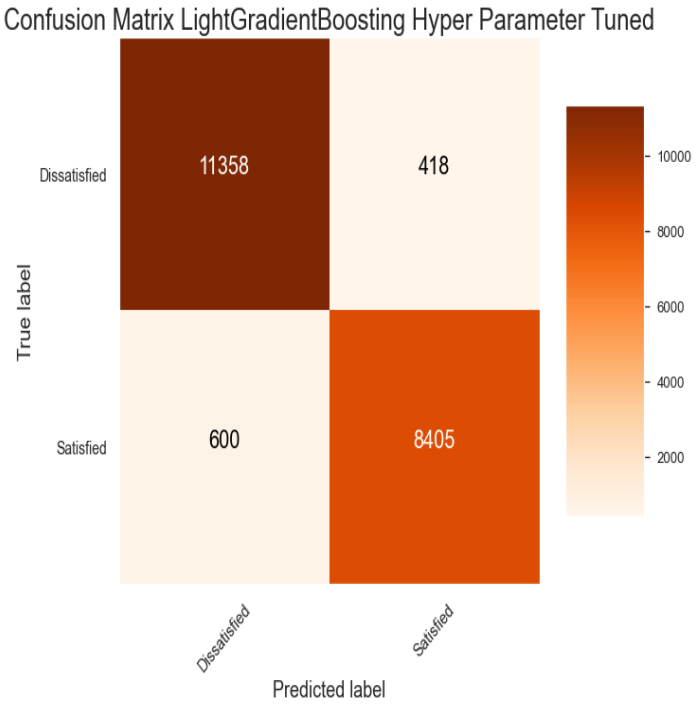


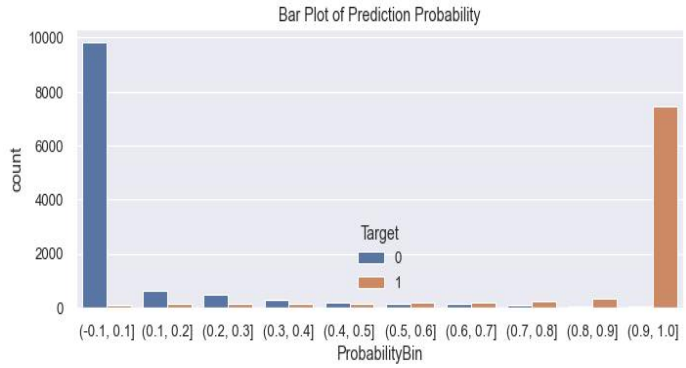



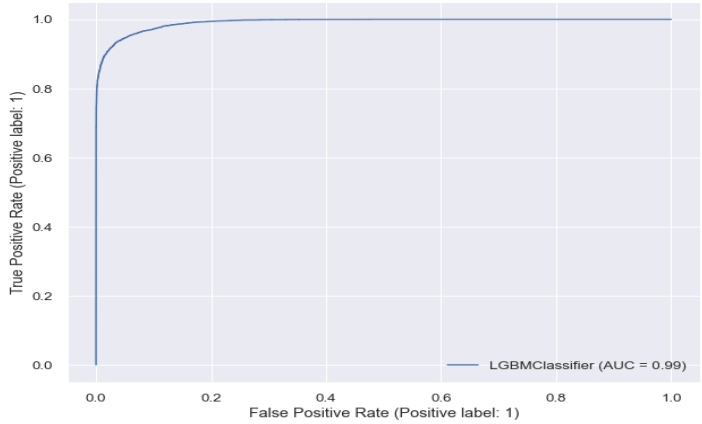


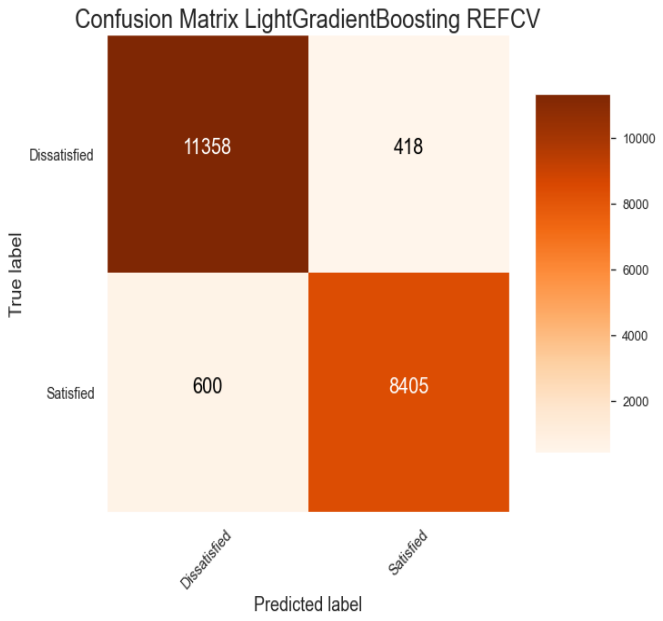



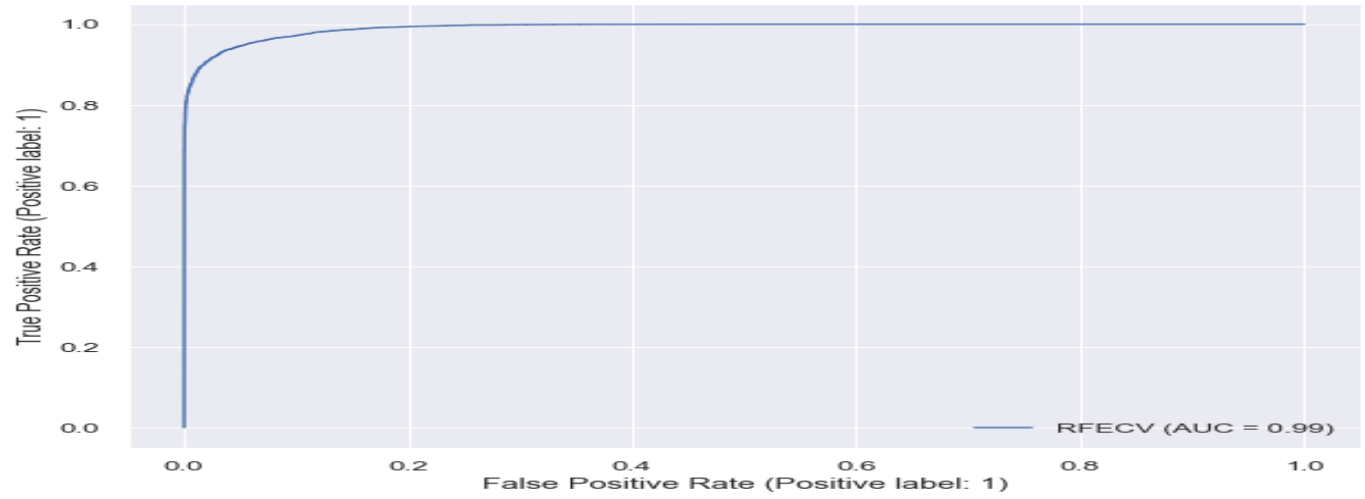



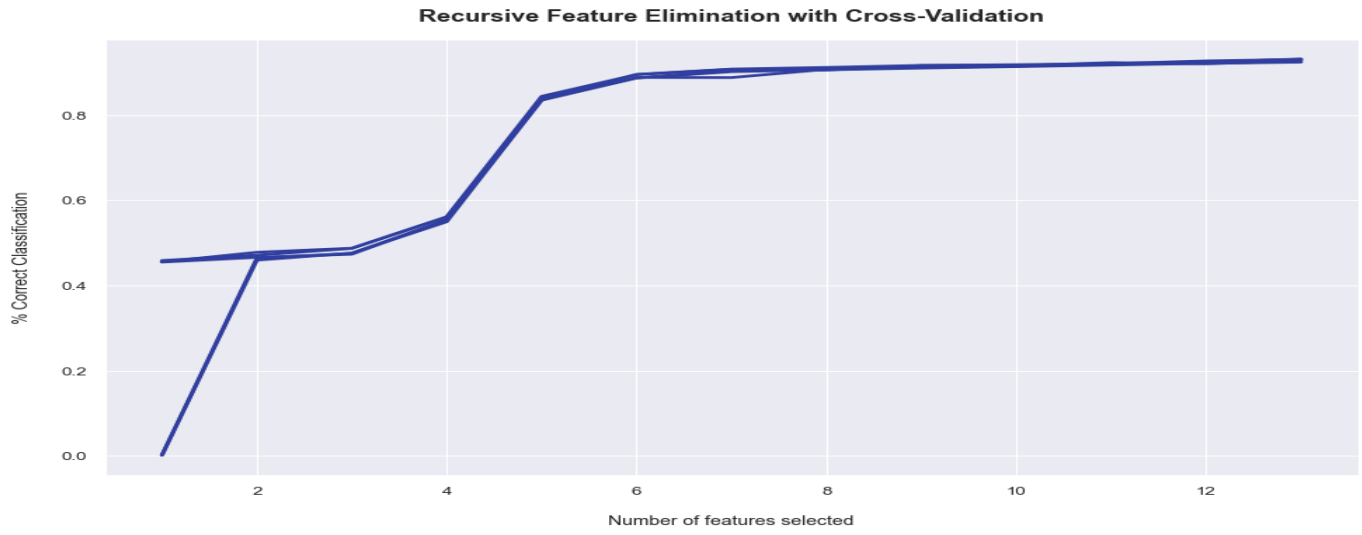

```
Misclassified Counts: 1018
accuracy........ 0.9510
precision score........ 0.9512
recall score........ 0.9489
classification_report
              precision    recall  f1-score   support

           0       0.95      0.96      0.96     11776
           1       0.95      0.93      0.94      9005

    accuracy                           0.95     20781
   macro avg       0.95      0.95      0.95     20781
weighted avg       0.95      0.95      0.95     20781
```

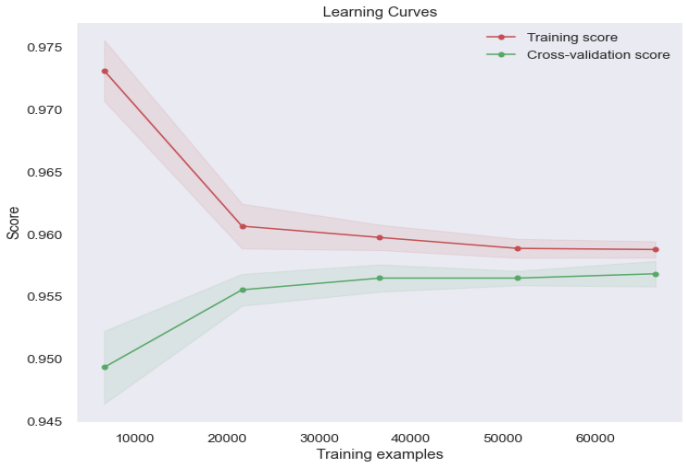
**LightGradientBoosting VIF Data Contd..**

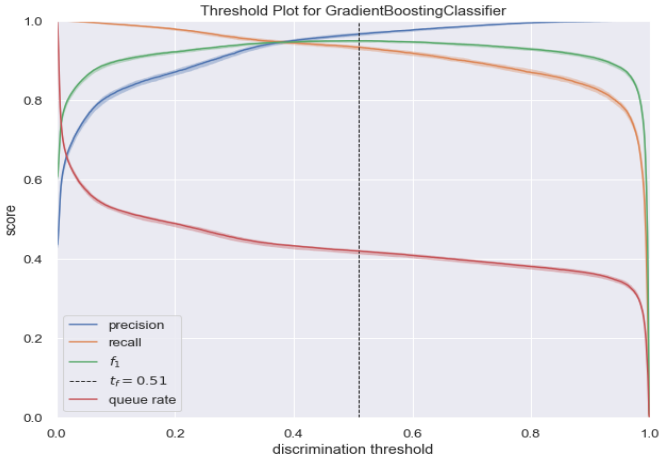


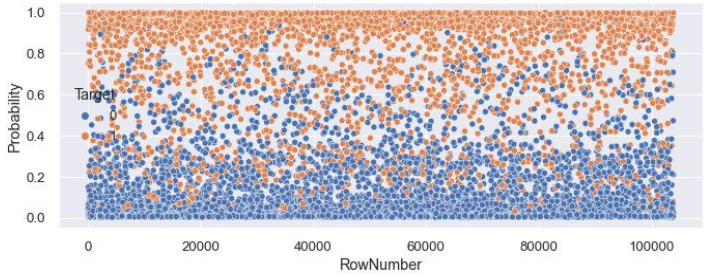

```
Misclassified Counts: 1018
accuracy........ 0.9510
precision score........ 0.9512
recall score........ 0.9489
classification_report
              precision    recall  f1-score   support

           0       0.95      0.96      0.96     11776
           1       0.95      0.93      0.94      9005

    accuracy                           0.95     20781
   macro avg       0.95      0.95      0.95     20781
weighted avg       0.95      0.95      0.95     20781
```

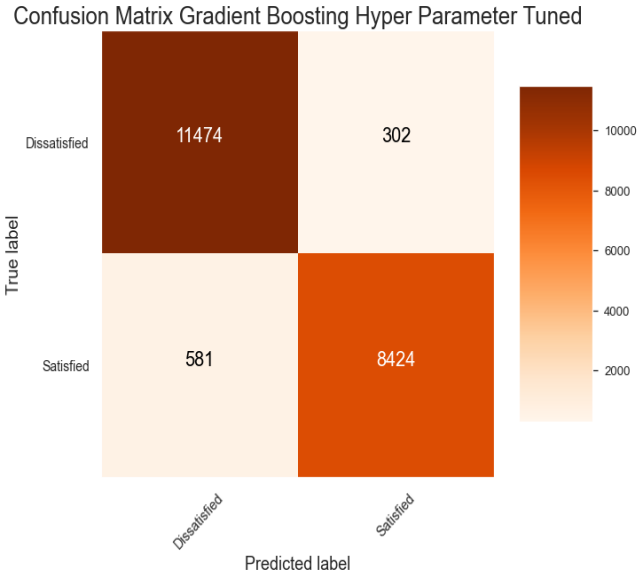


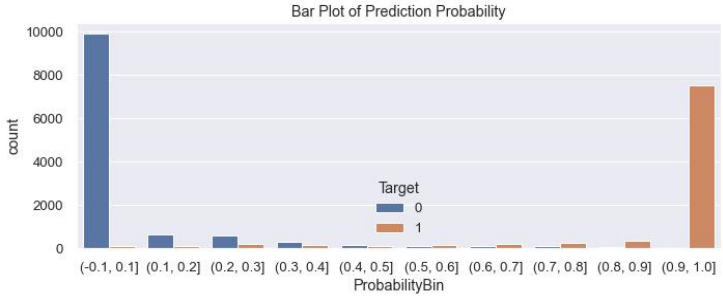



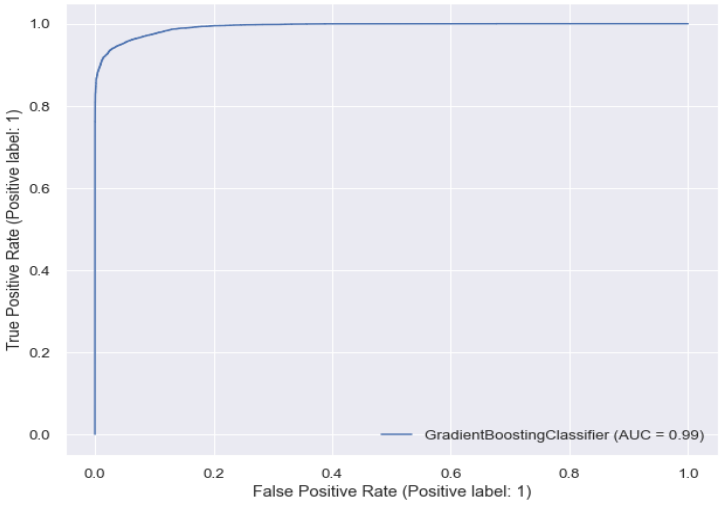


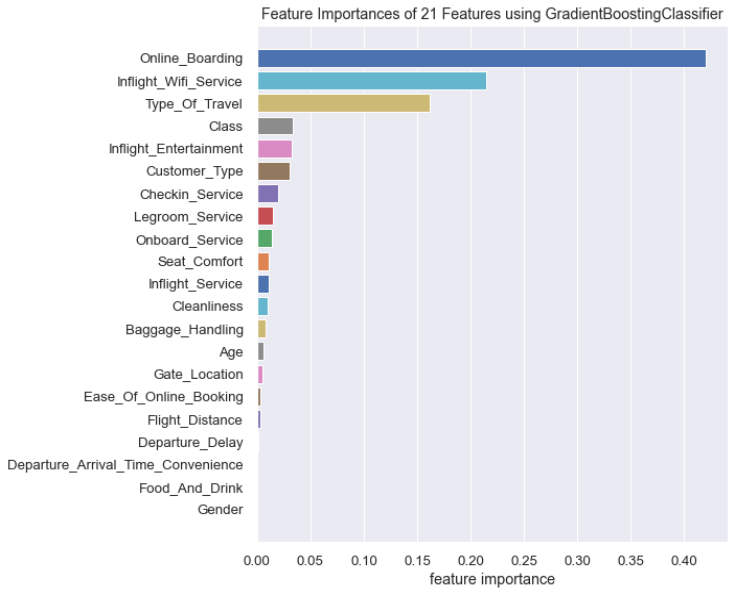



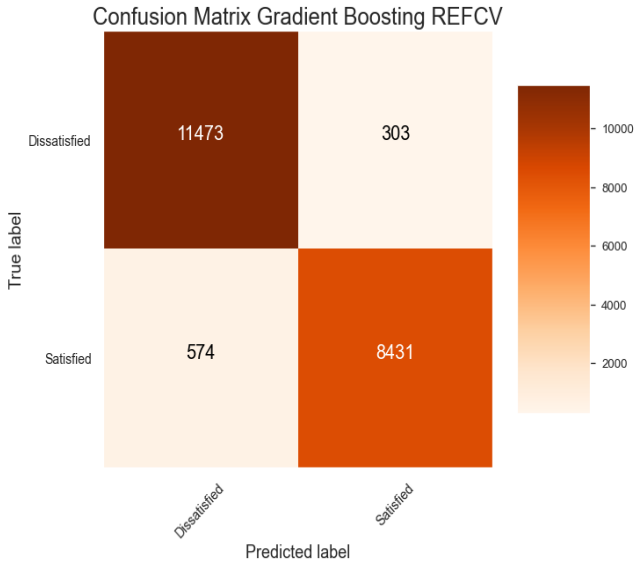



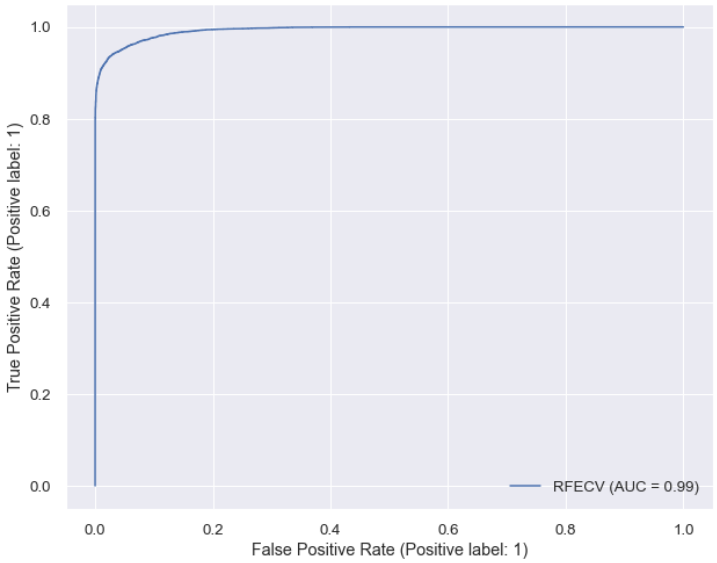




```
Misclassified Counts: 883
accuracy........ 0.9575
precision score........ 0.9586
recall score........ 0.9549
classification_report
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     11776
           1       0.97      0.94      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.95      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

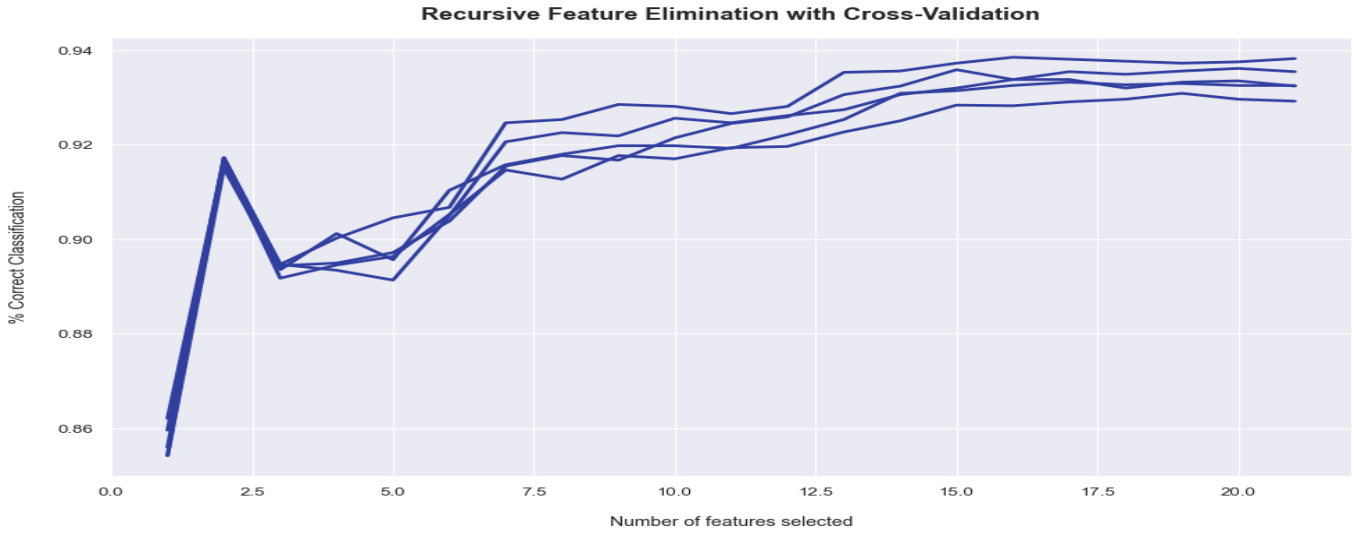
## Gradient Boosting UnBinned Data Contd..

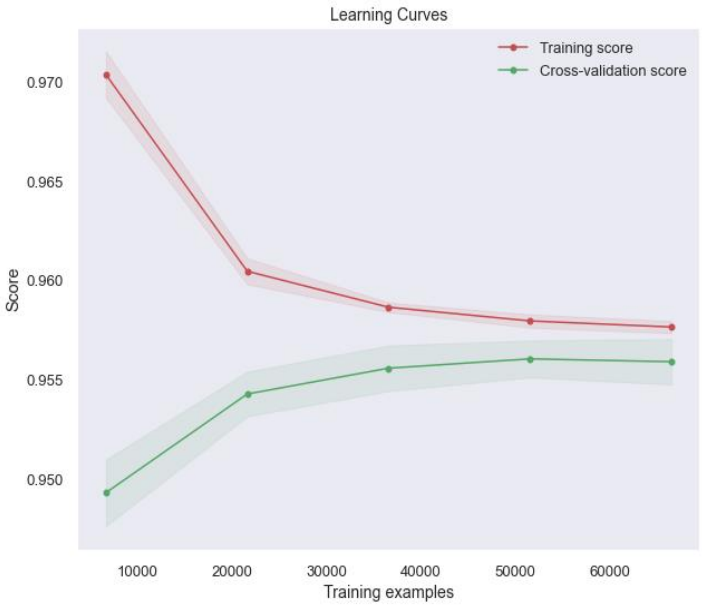


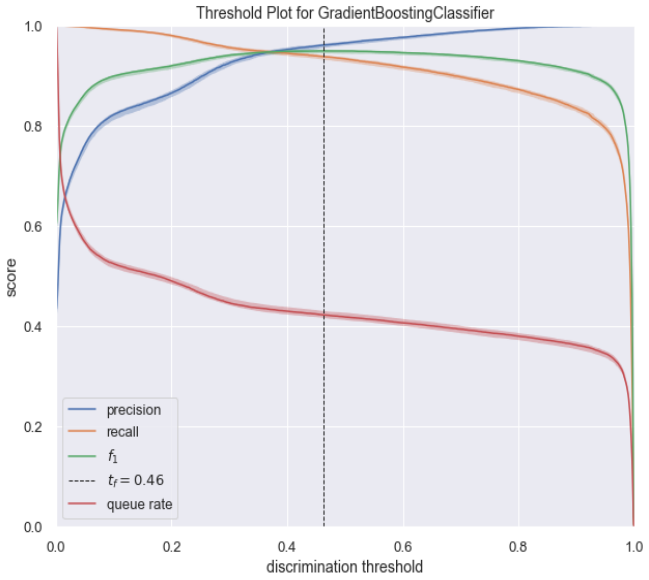



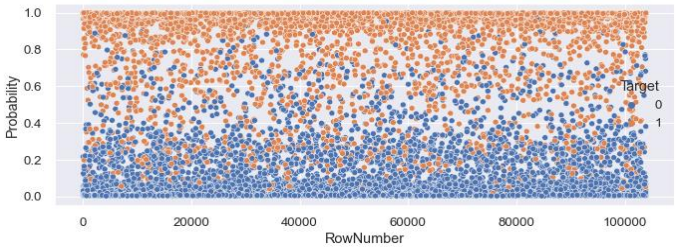



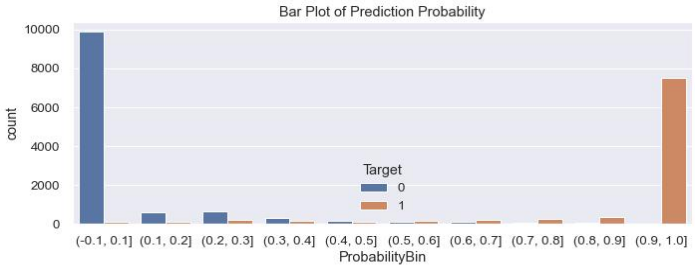

```
Misclassified Counts: 877
accuracy........ 0.9578
precision score........ 0.9588
recall score........ 0.9553
classification_report
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     11776
           1       0.97      0.94      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

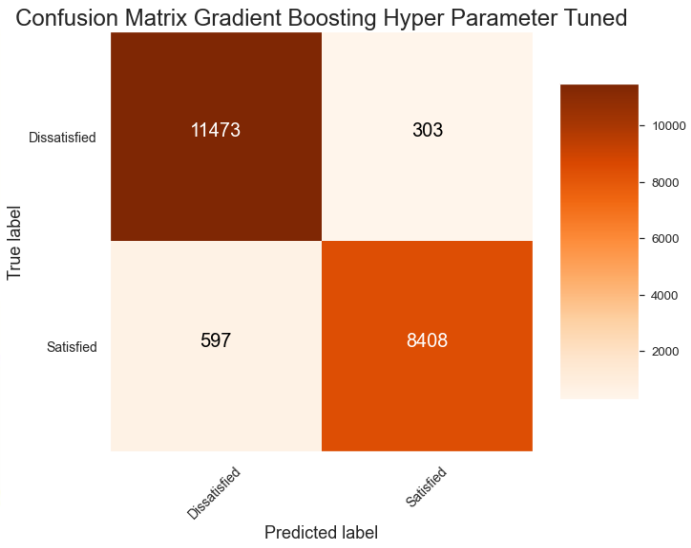


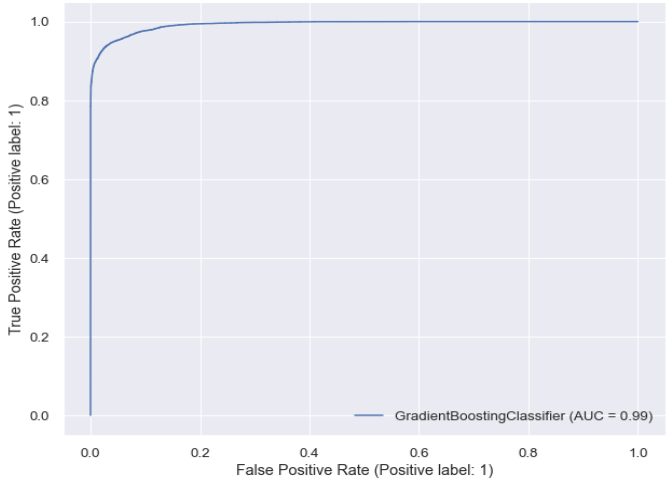


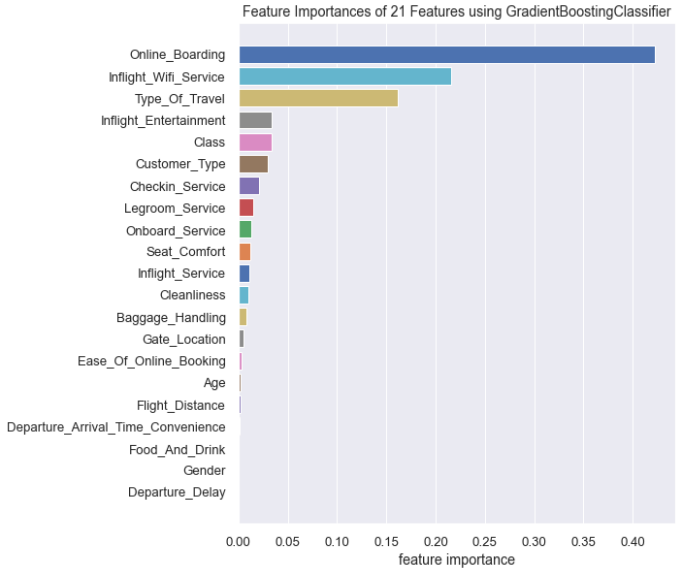



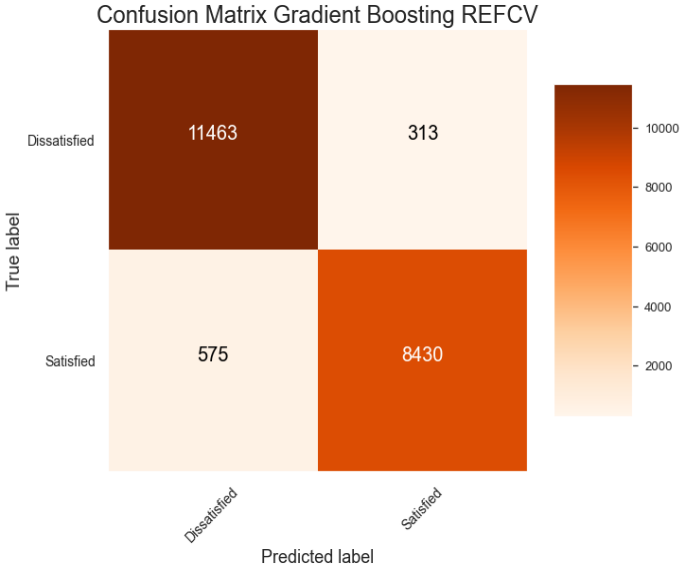



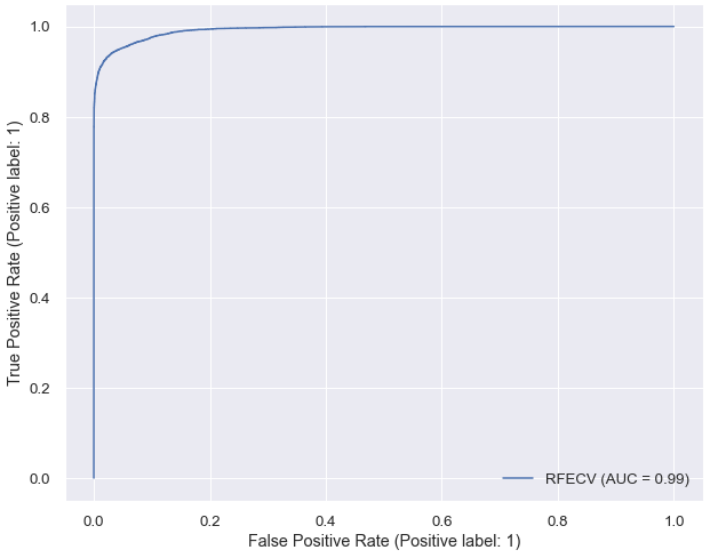



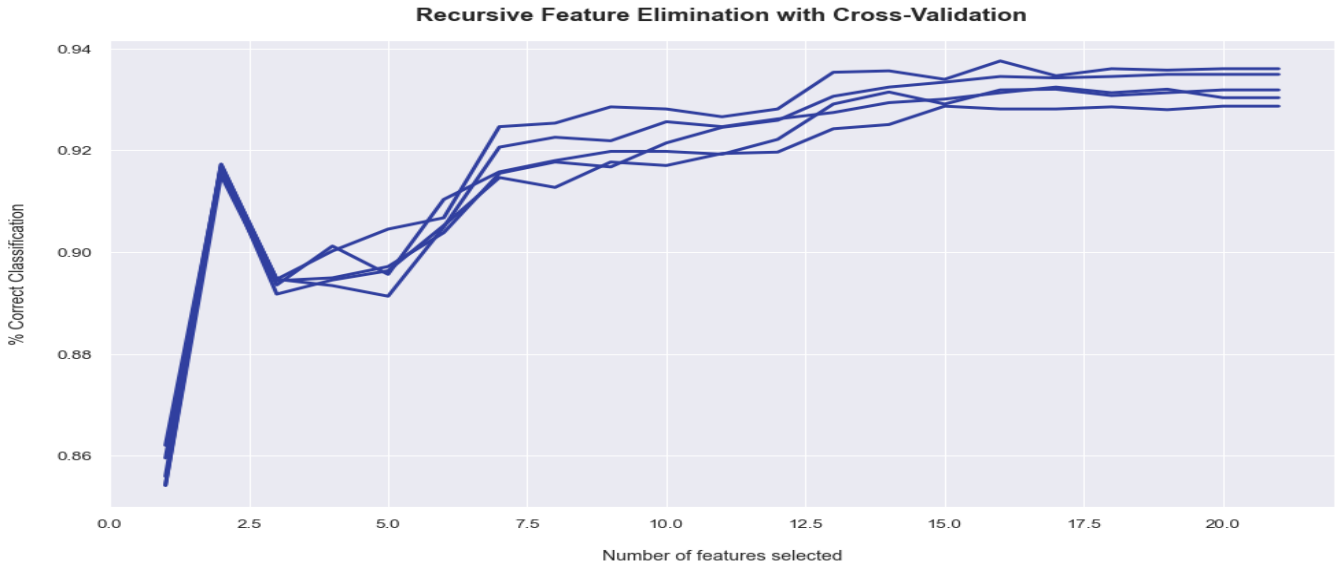

```
Misclassified Counts: 900
accuracy........ 0.9567
precision score........ 0.9579
recall score........ 0.9540
classification_report
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     11776
           1       0.97      0.93      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.95      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

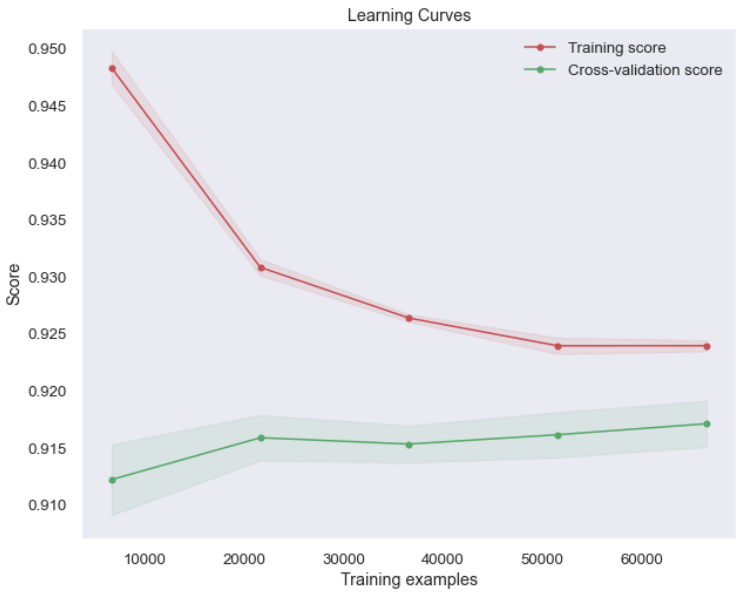


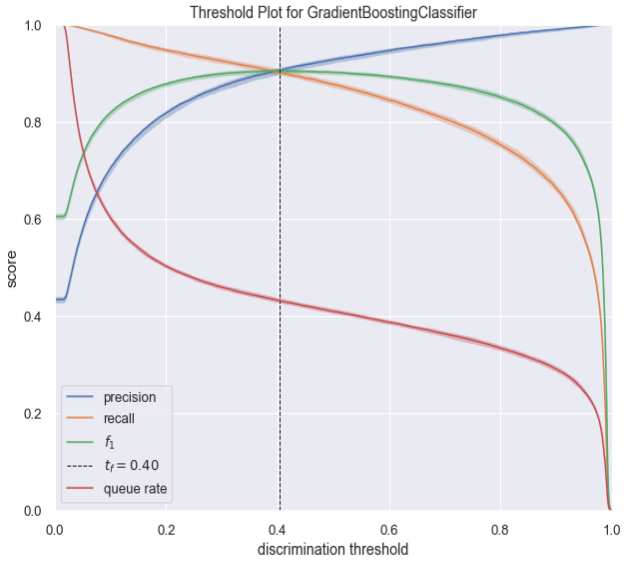



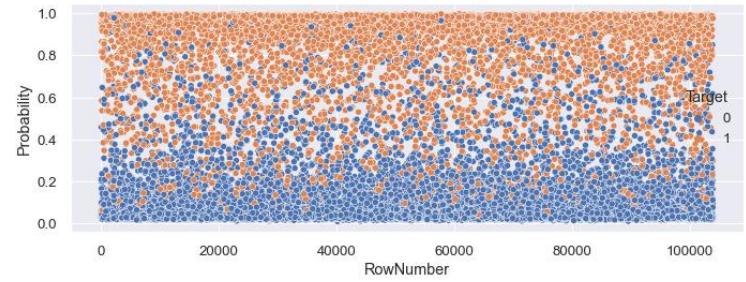



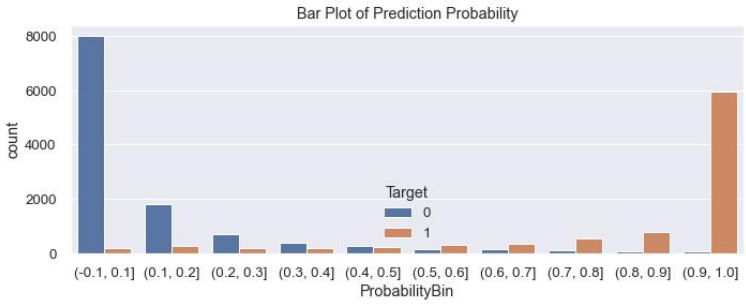

```
Misclassified Counts: 888
accuracy........ 0.9573
precision score........ 0.9582
recall score........ 0.9548
classification_report
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     11776
           1       0.96      0.94      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.95      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

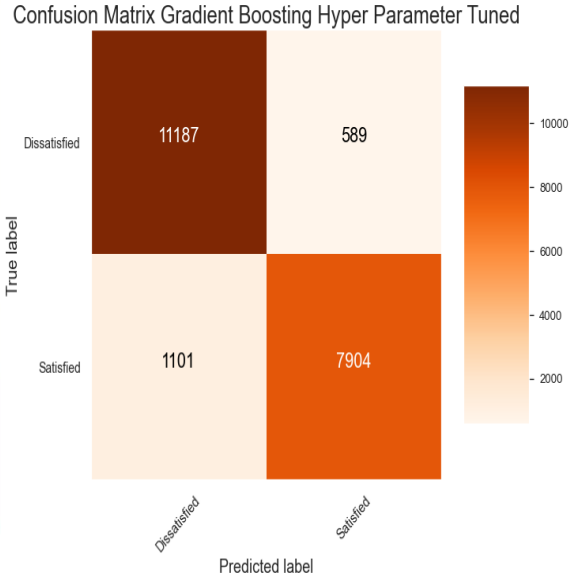


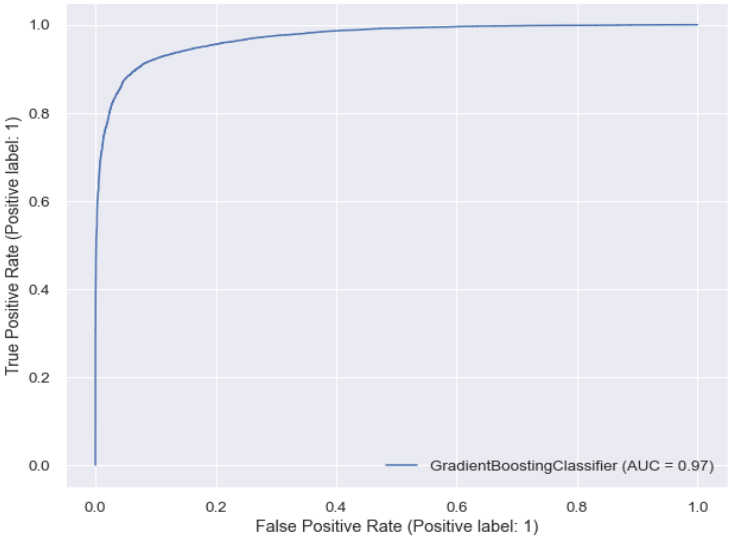


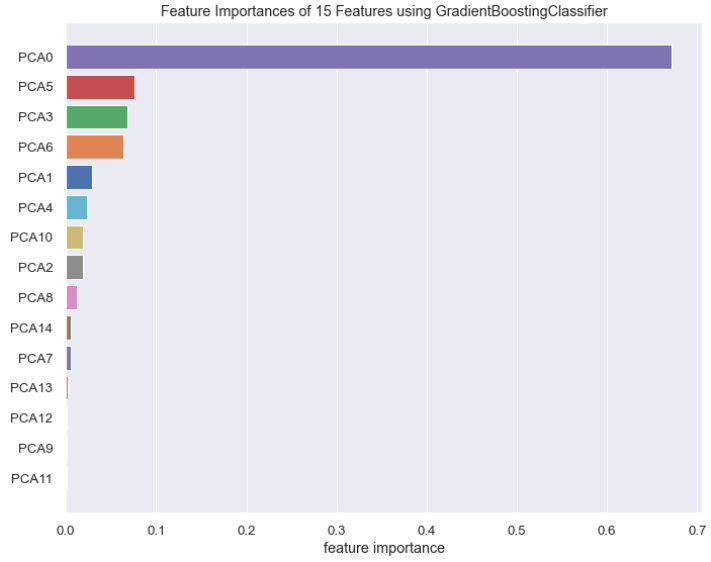



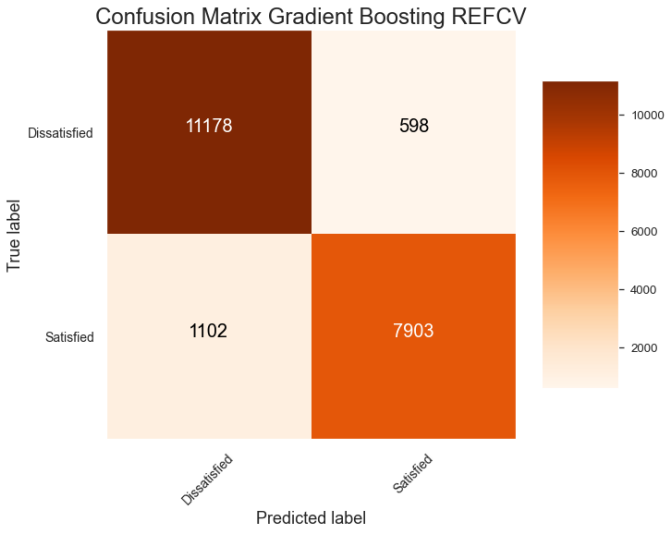



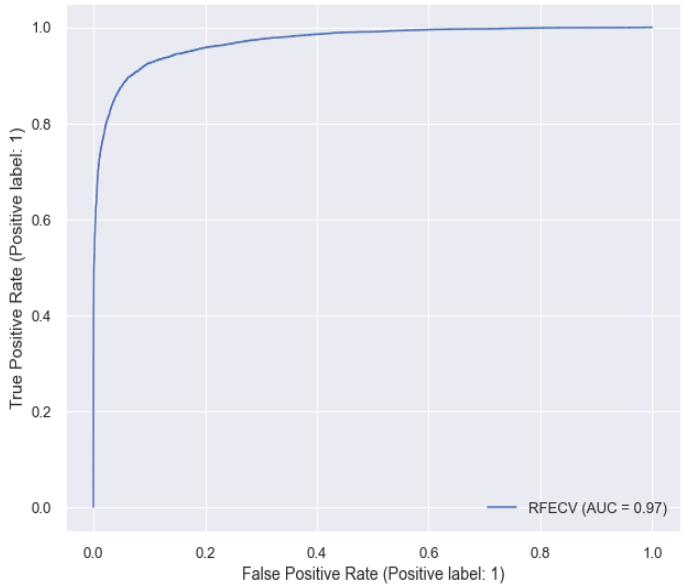



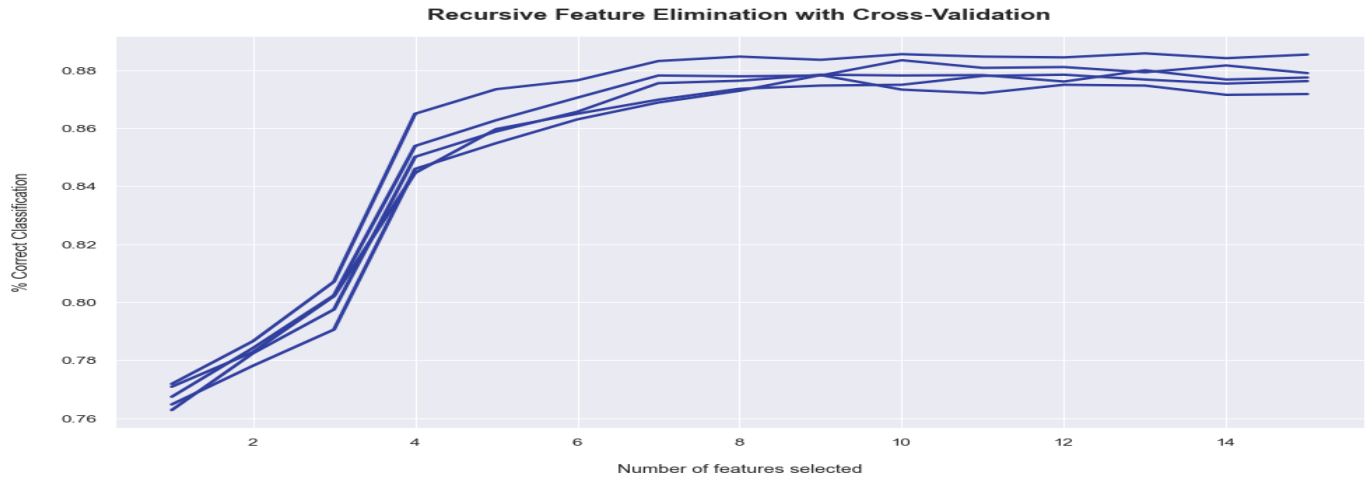

```
Misclassified Counts: 1690
accuracy........ 0.9187
precision score........ 0.9205
recall score........ 0.9139
classification_report
              precision    recall  f1-score   support

           0       0.91      0.95      0.93     11776
           1       0.93      0.88      0.90      9005

    accuracy                           0.92     20781
   macro avg       0.92      0.91      0.92     20781
weighted avg       0.92      0.92      0.92     20781
```

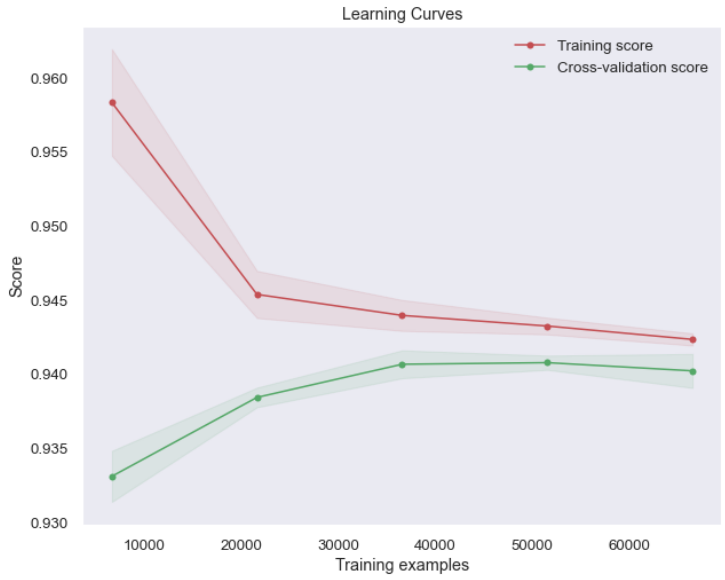


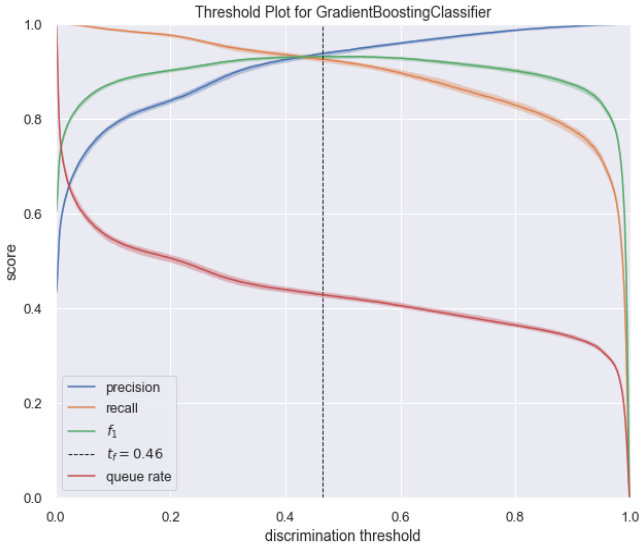



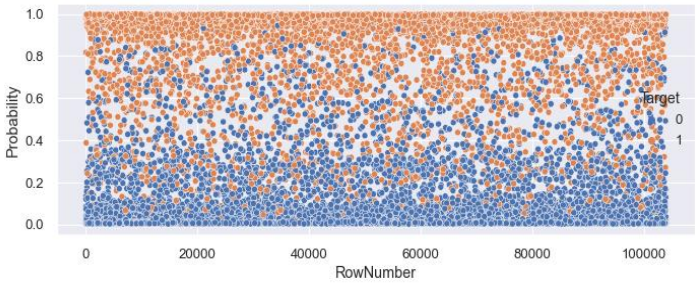



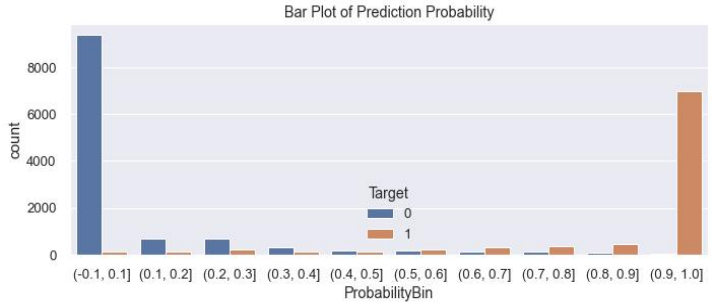

```
Misclassified Counts: 1700
accuracy..... 0.9182
precision score..... 0.9200
recall score..... 0.9134
classification_report
              precision    recall  f1-score   support

           0       0.91      0.95      0.93     11776
           1       0.93      0.88      0.90      9005

    accuracy                           0.92     20781
   macro avg       0.92      0.91      0.92     20781
weighted avg       0.92      0.92      0.92     20781
```

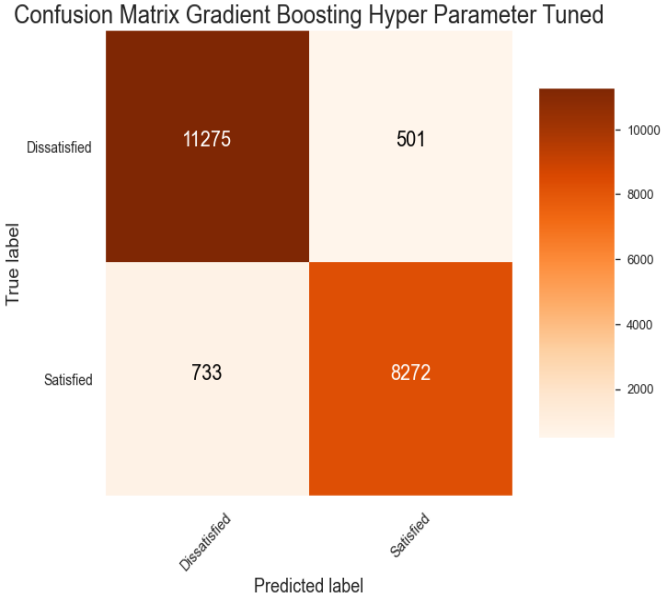


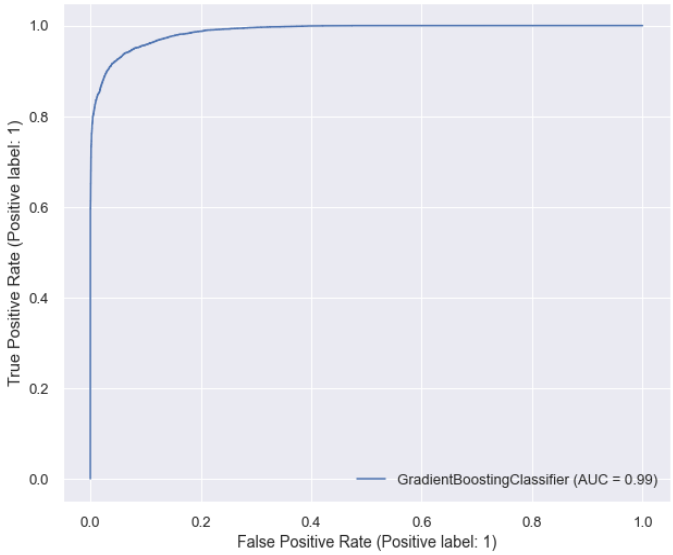


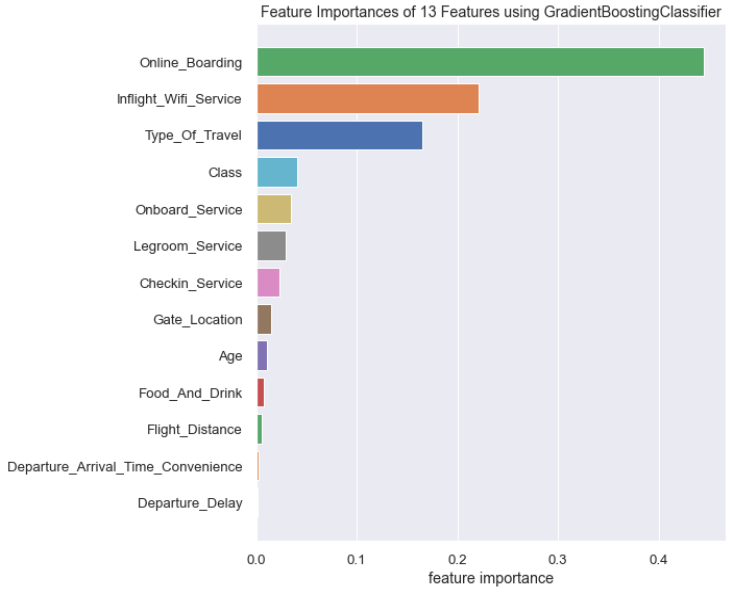



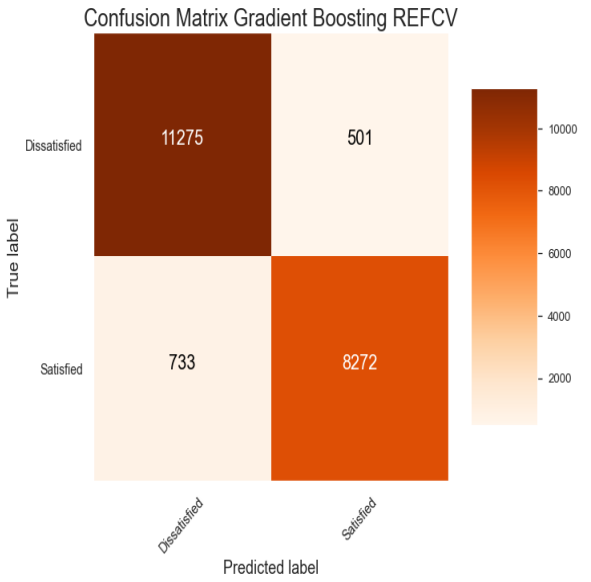



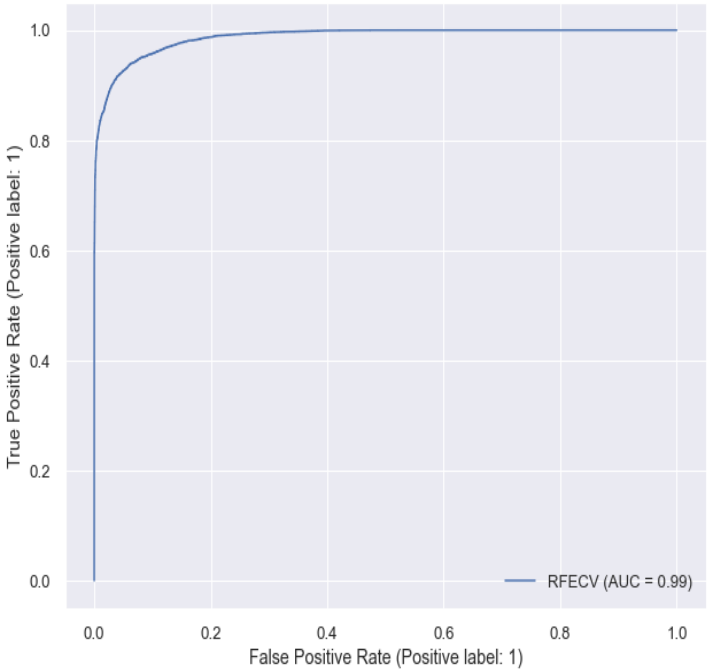




```
Misclassified Counts: 1234
accuracy........ 0.9406
precision score........ 0.9409
recall score........ 0.9380
classification_report
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     11776
           1       0.94      0.92      0.93      9005

    accuracy                           0.94     20781
   macro avg       0.94      0.94      0.94     20781
weighted avg       0.94      0.94      0.94     20781
```

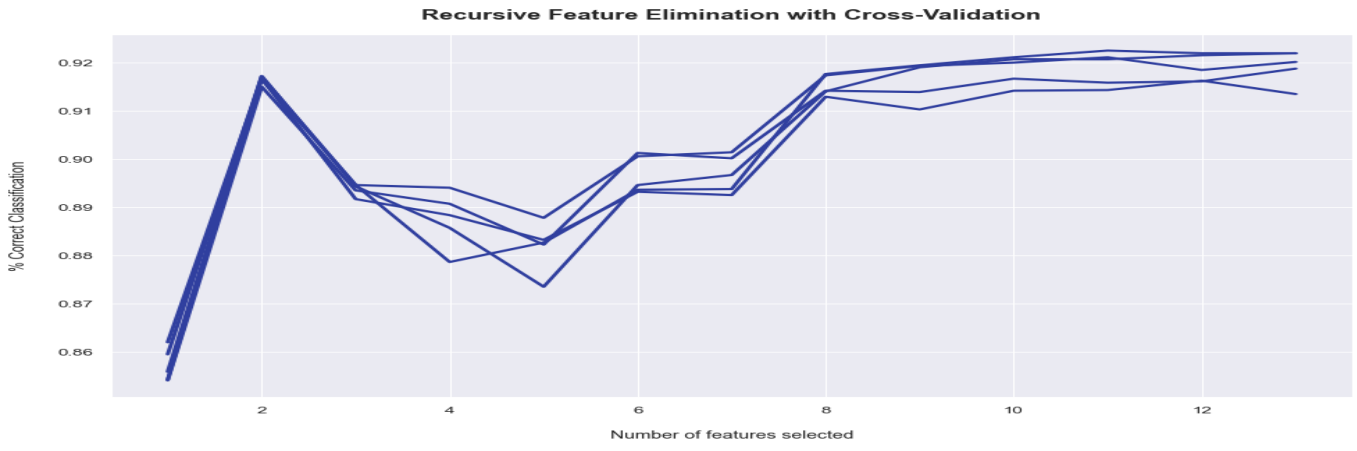


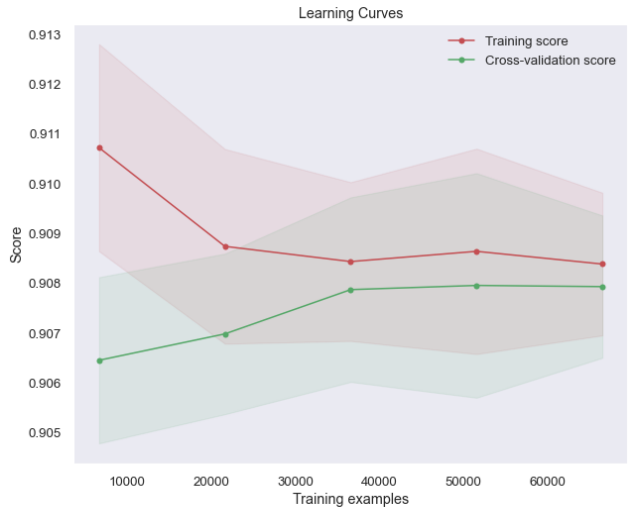



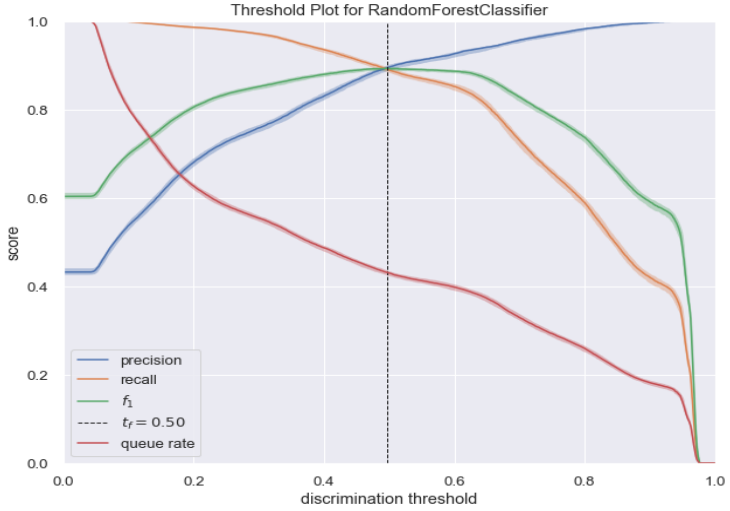

Misclassified Counts: 1234
accuracy..... 0.9406
precision score..... 0.9409
recall score..... 0.9380
classification_report

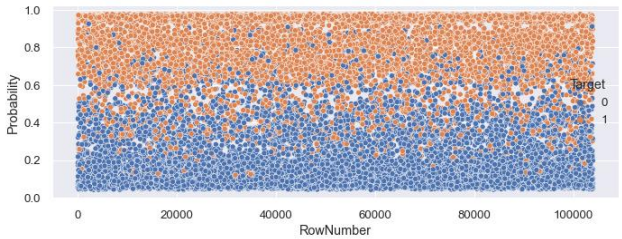
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.94 | 0.96 | 0.95 | 11776 |
| 1 | 0.94 | 0.92 | 0.93 | 9005 |
| accuracy | | | 0.94 | 20781 |
| macro avg | 0.94 | 0.94 | 0.94 | 20781 |
| weighted avg | 0.94 | 0.94 | 0.94 | 20781 |

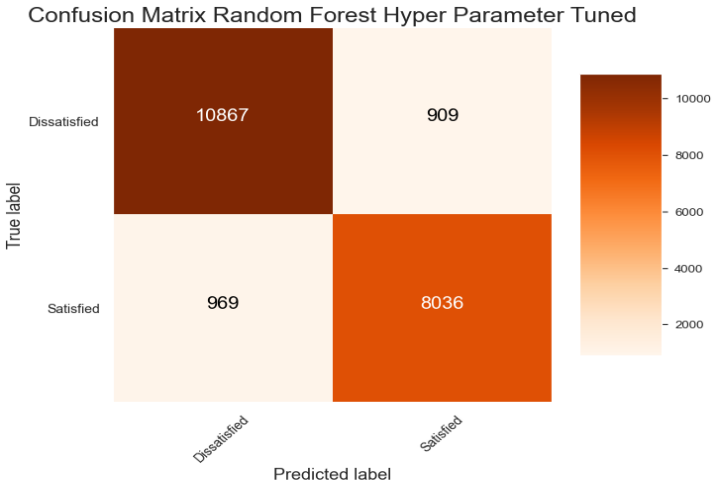


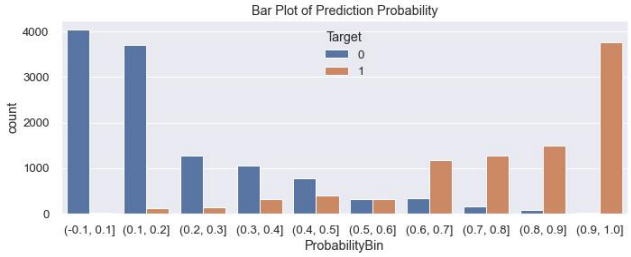
## Random Forest - Unbinned Data

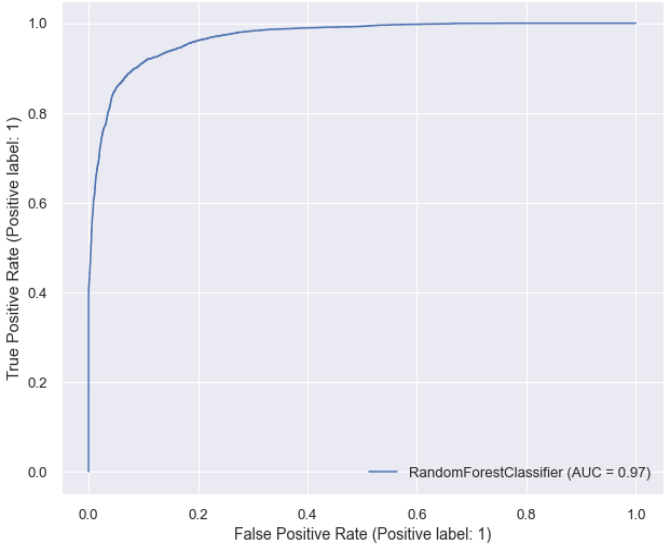


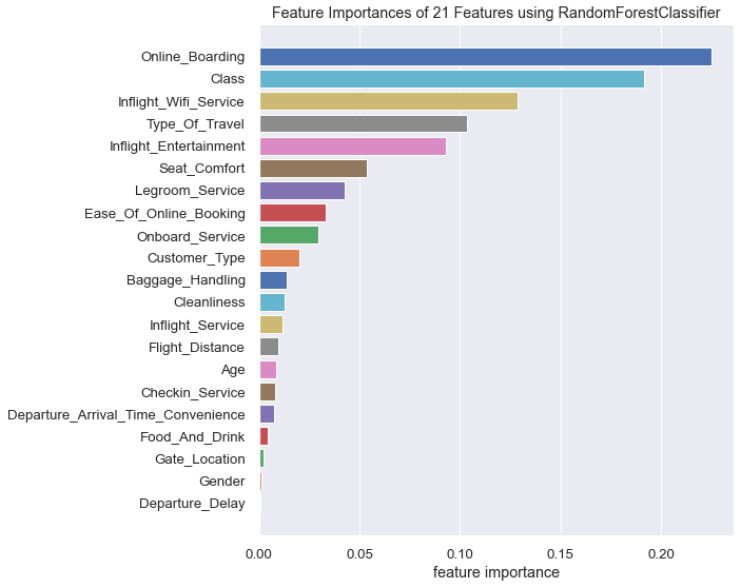



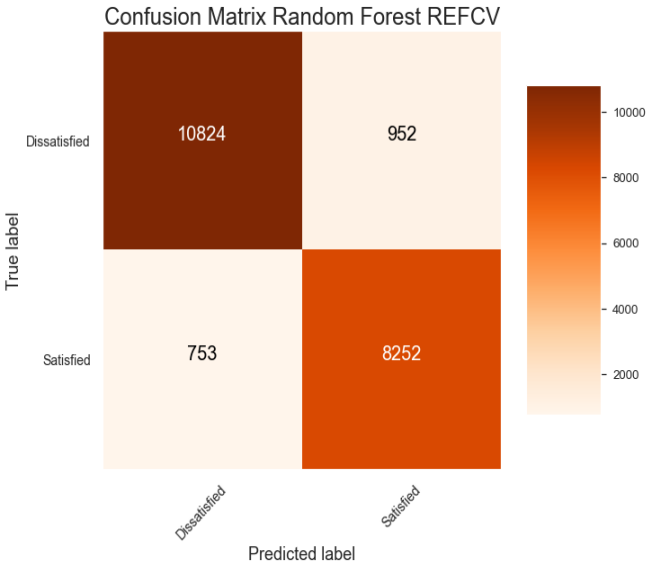



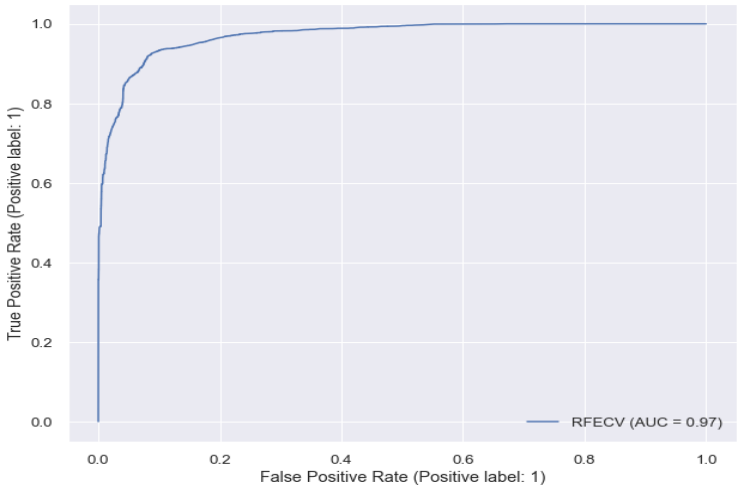




```
Misclassified Counts: 1878
accuracy........ 0.9096
precision score........ 0.9083
recall score........ 0.9076
classification_report
              precision    recall  f1-score   support

           0       0.92      0.92      0.92     11776
           1       0.90      0.89      0.90      9005

    accuracy                           0.91     20781
   macro avg       0.91      0.91      0.91     20781
weighted avg       0.91      0.91      0.91     20781
```

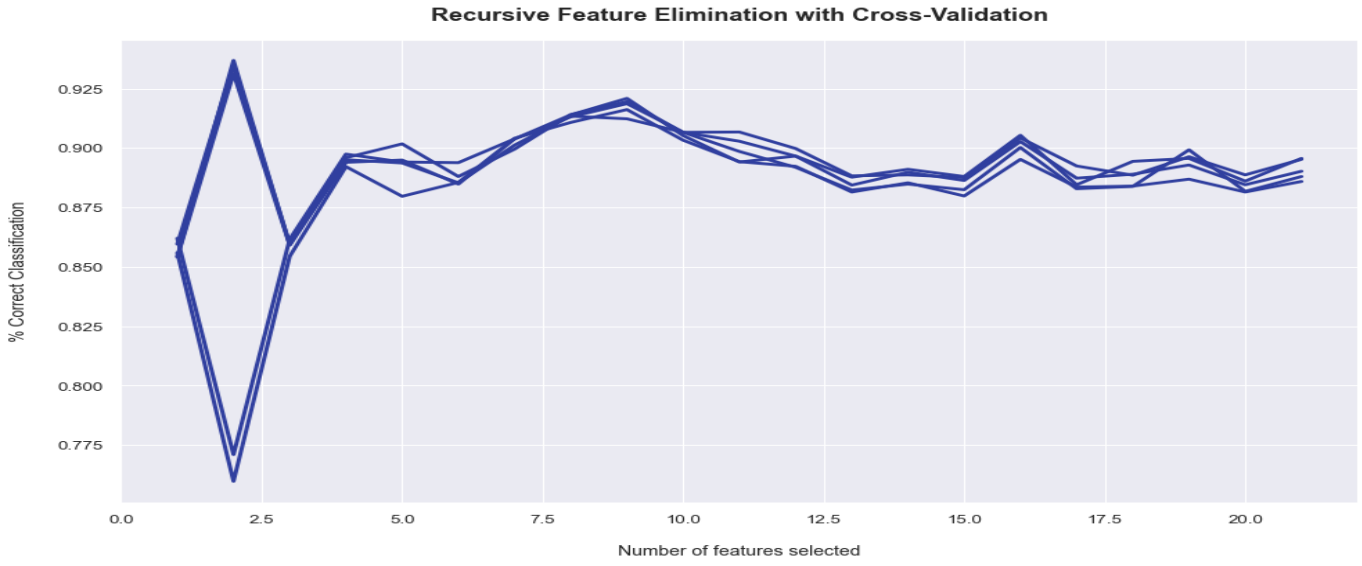
## Random Forest - Unbinned Data Contd..



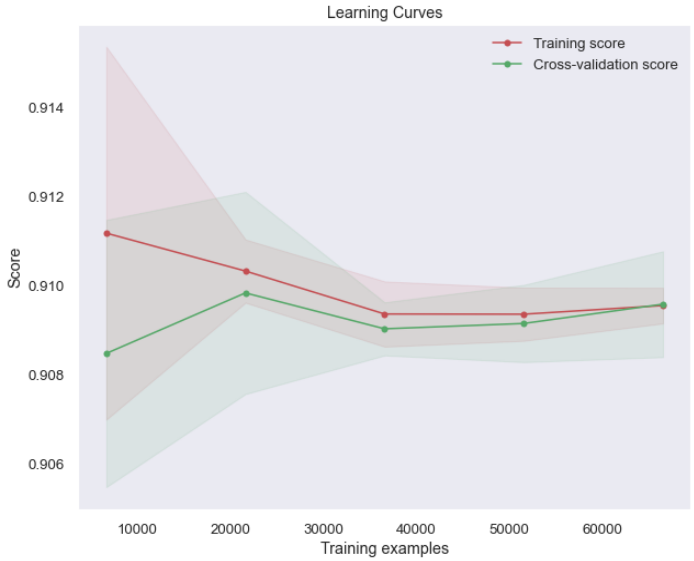



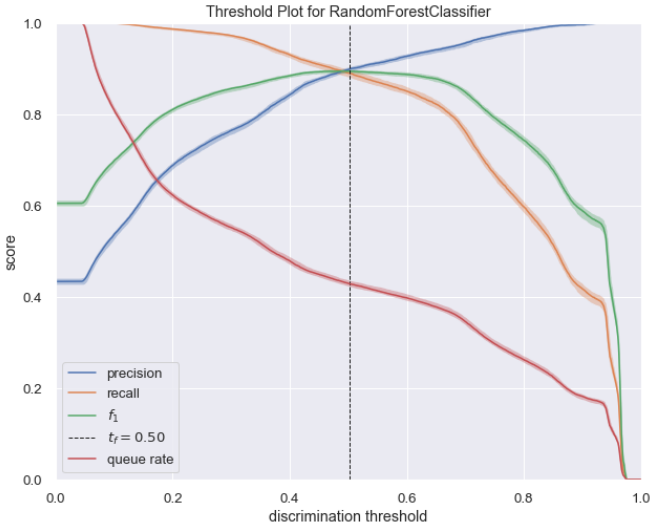



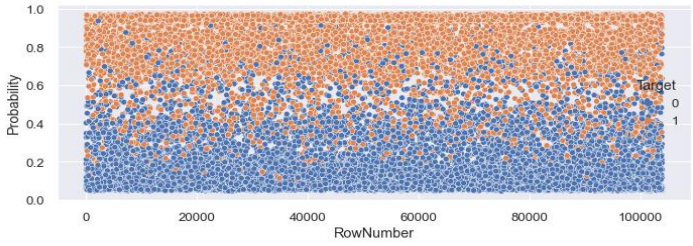

```
Misclassified Counts: 1705
accuracy........ 0.9180
precision score........ 0.9158
recall score........ 0.9178
classification_report
              precision    recall  f1-score   support

           0       0.93      0.92      0.93     11776
           1       0.90      0.92      0.91      9005

    accuracy                           0.92     20781
   macro avg       0.92      0.92      0.92     20781
weighted avg       0.92      0.92      0.92     20781
```

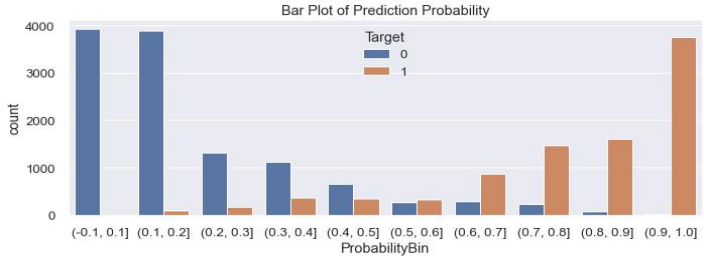


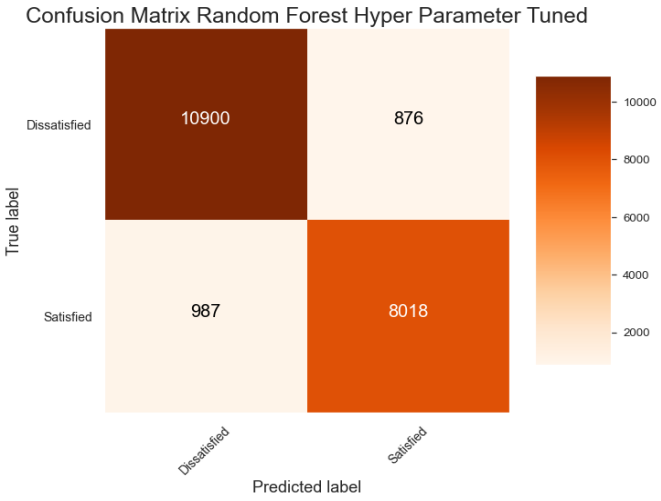
## Random Forest - Binned Data

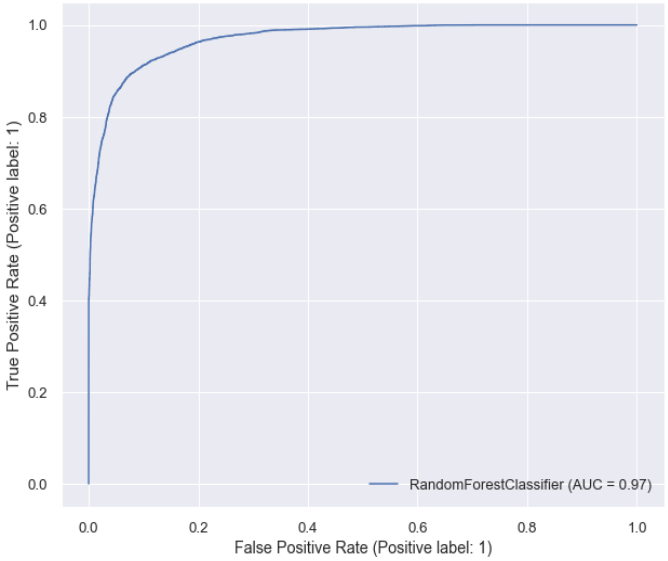


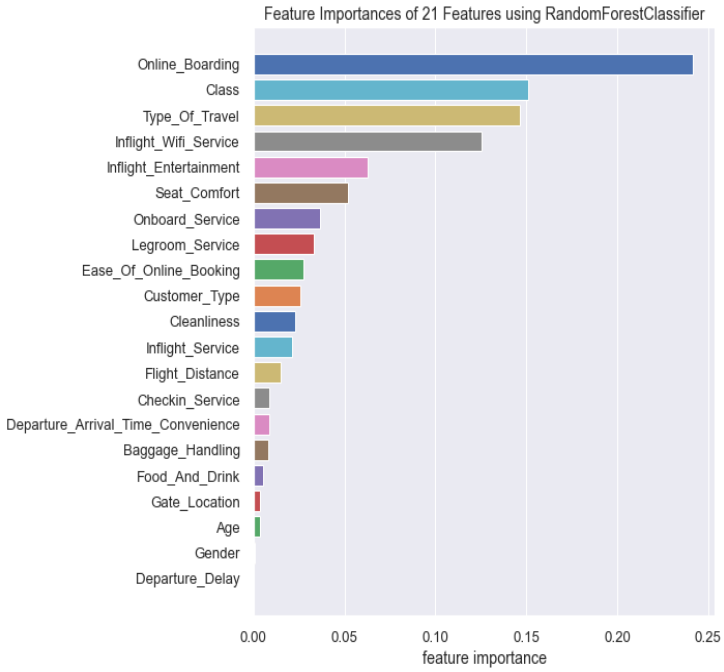



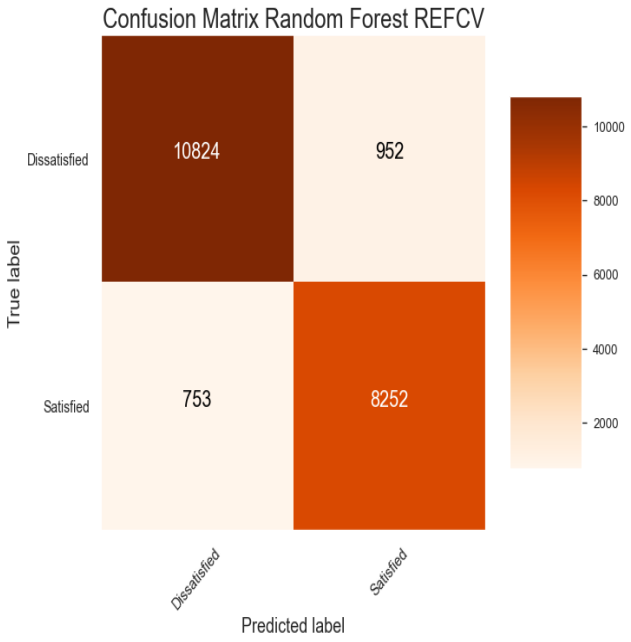



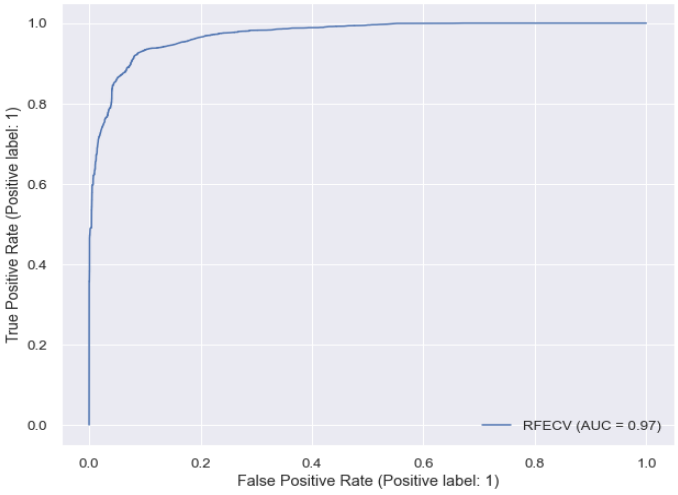




```
Misclassified Counts: 1863
accuracy..... 0.9104
precision score..... 0.9092
recall score..... 0.9080
classification_report
              precision    recall  f1-score   support

           0       0.92      0.93      0.92     11776
           1       0.90      0.89      0.90      9005

    accuracy                           0.91     20781
   macro avg       0.91      0.91      0.91     20781
weighted avg       0.91      0.91      0.91     20781
```

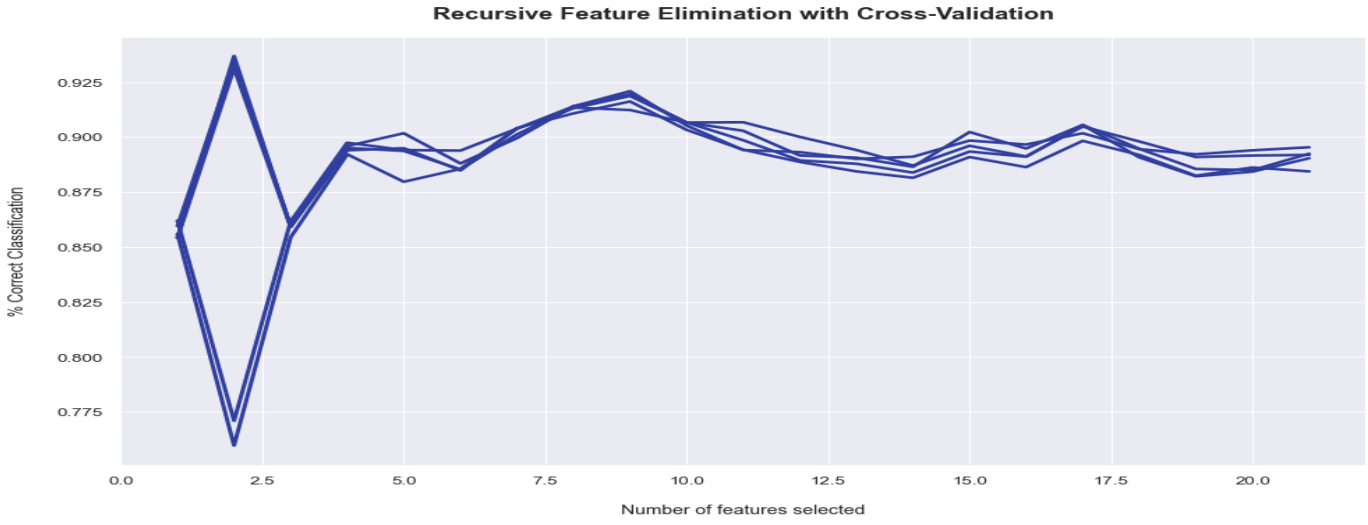
## Random Forest - Binned Data Contd..



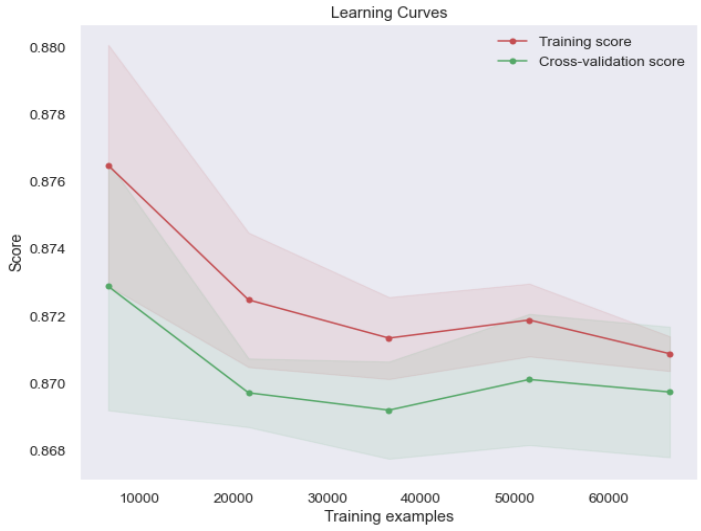



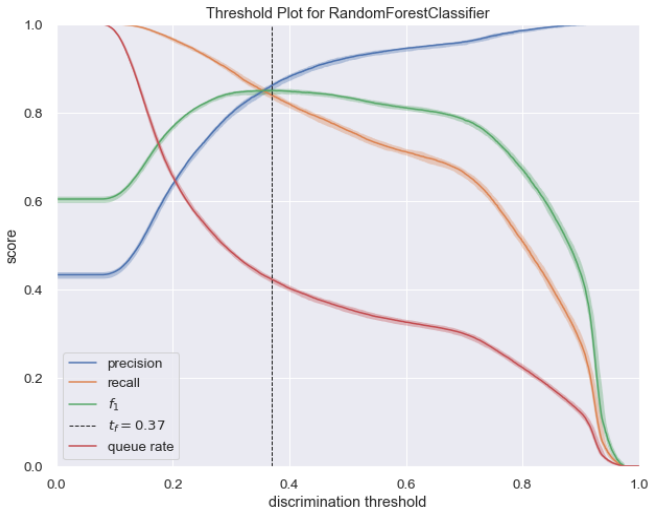



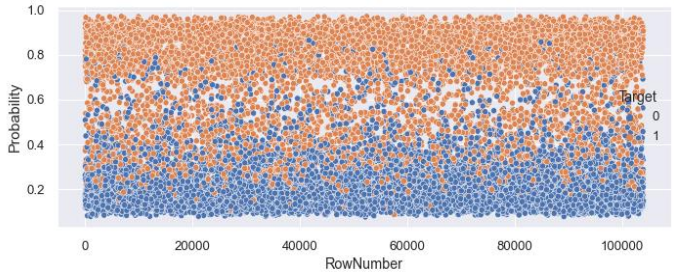

```
Misclassified Counts: 1705
accuracy........ 0.9180
precision score........ 0.9158
recall score........ 0.9178
classification_report
              precision    recall  f1-score   support

           0       0.93      0.92      0.93     11776
           1       0.90      0.92      0.91      9005

    accuracy                           0.92     20781
   macro avg       0.92      0.92      0.92     20781
weighted avg       0.92      0.92      0.92     20781
```

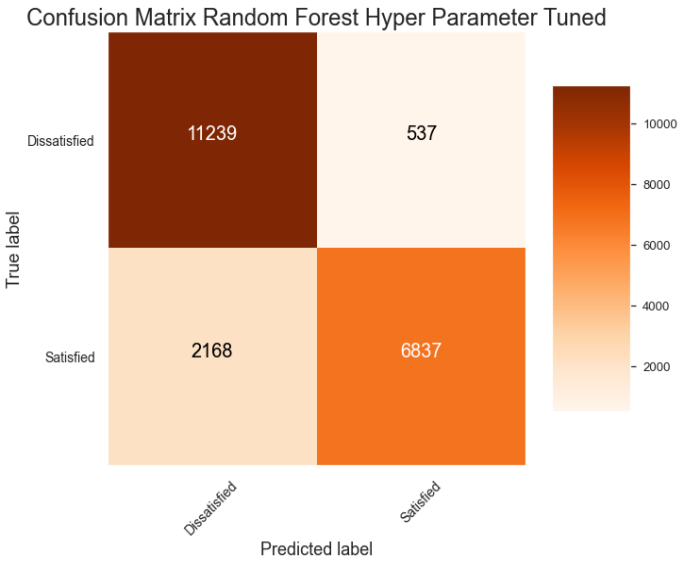


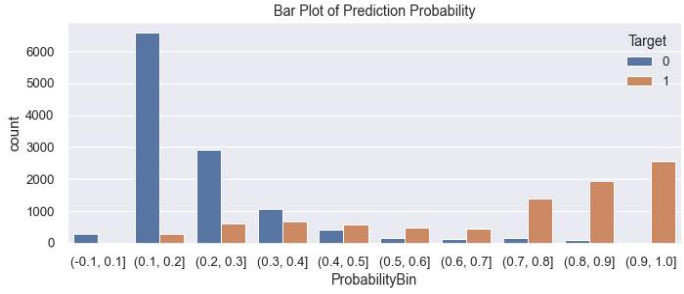
## Random Forest - PCA Data

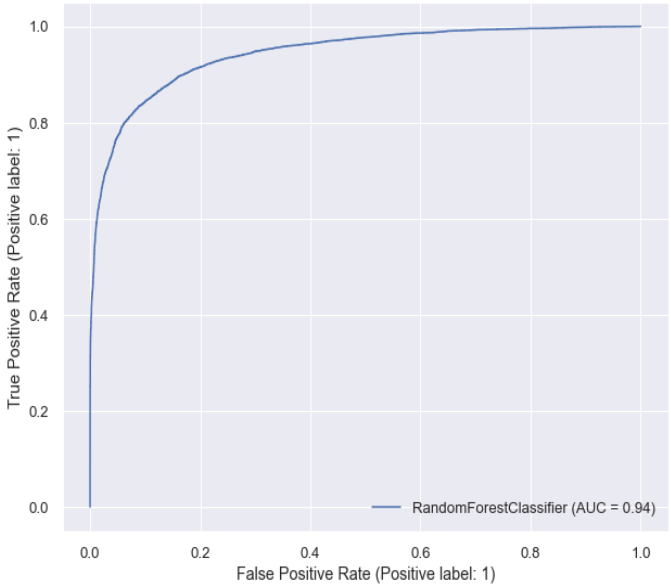


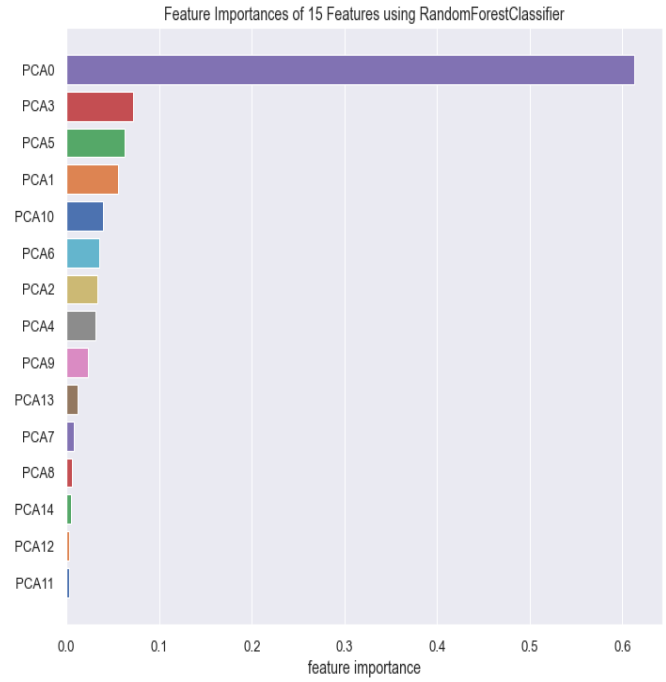



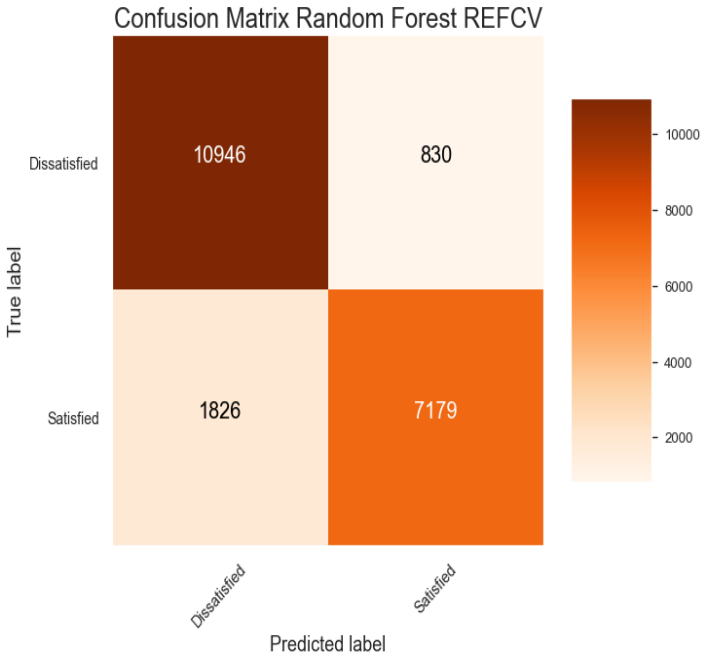



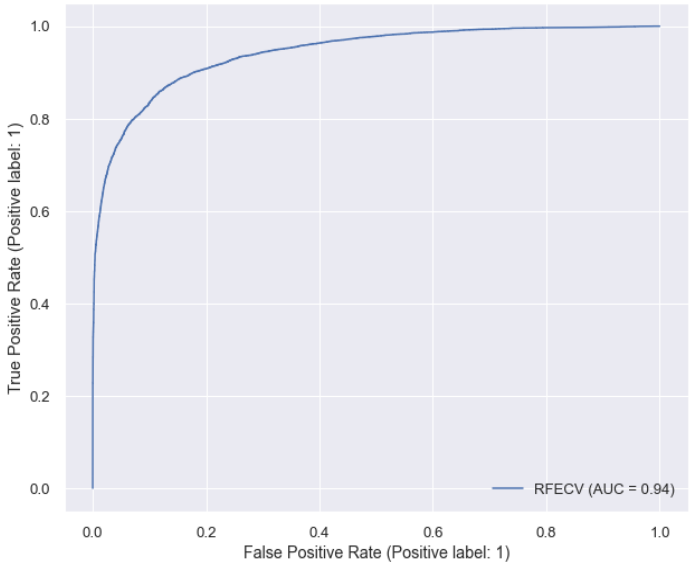




```
Misclassified Counts: 2705
accuracy........ 0.8698
precision score........ 0.8827
recall score........ 0.8568
classification_report
              precision    recall  f1-score   support

           0       0.84      0.95      0.89     11776
           1       0.93      0.76      0.83      9005

    accuracy                           0.87     20781
   macro avg       0.88      0.86      0.86     20781
weighted avg       0.88      0.87      0.87     20781
```

## Random Forest - PCA Data Contd..

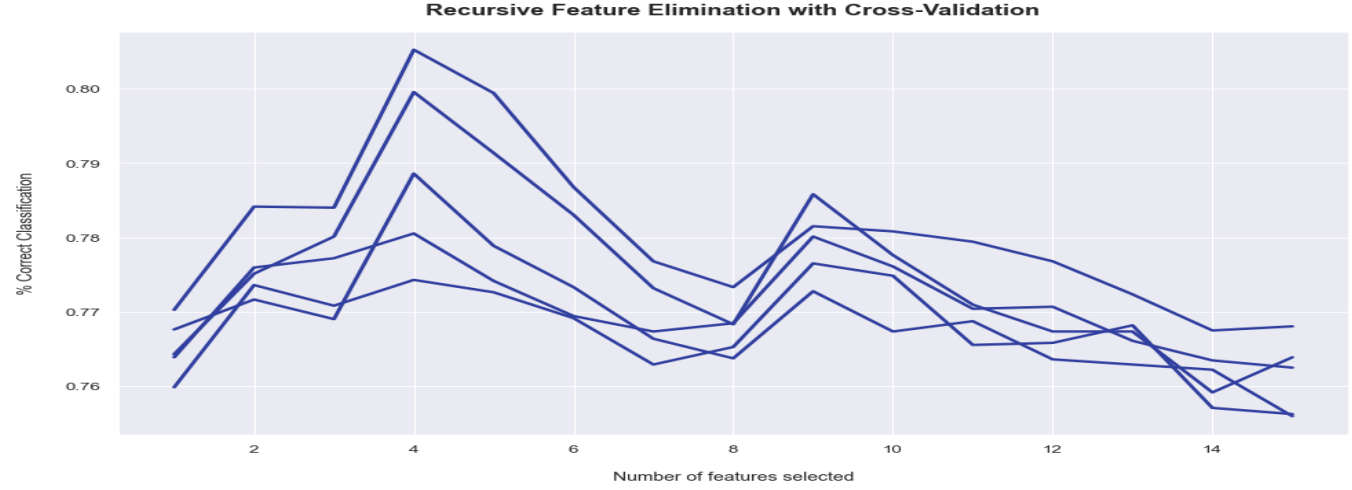


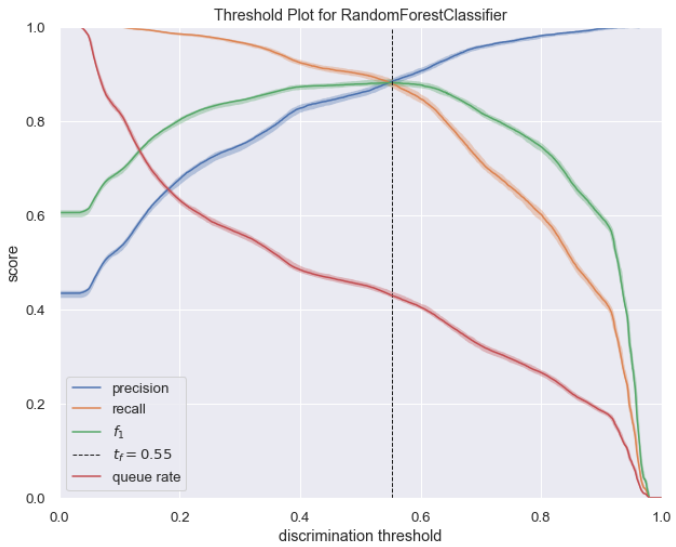



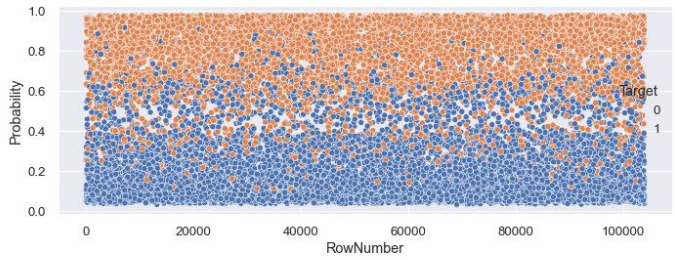

Misclassified Counts: 2656
accuracy..... 0.8722
precision score..... 0.8767
recall score..... 0.8634
classification_report

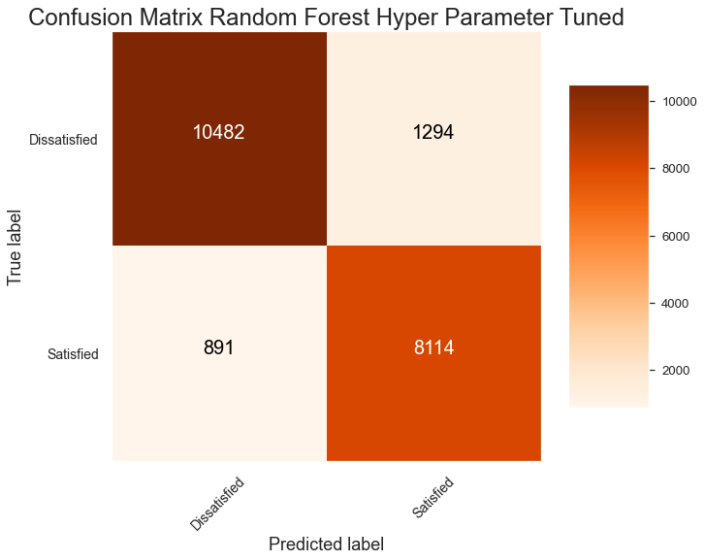
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.86 | 0.93 | 0.89 | 11776 |
| 1 | 0.90 | 0.80 | 0.84 | 9005 |
| accuracy | | | 0.87 | 20781 |
| macro avg | 0.88 | 0.86 | 0.87 | 20781 |
| weighted avg | 0.87 | 0.87 | 0.87 | 20781 |

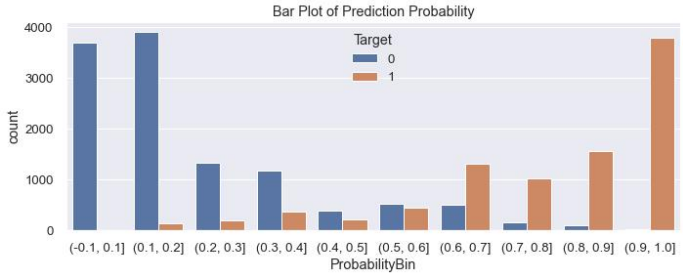


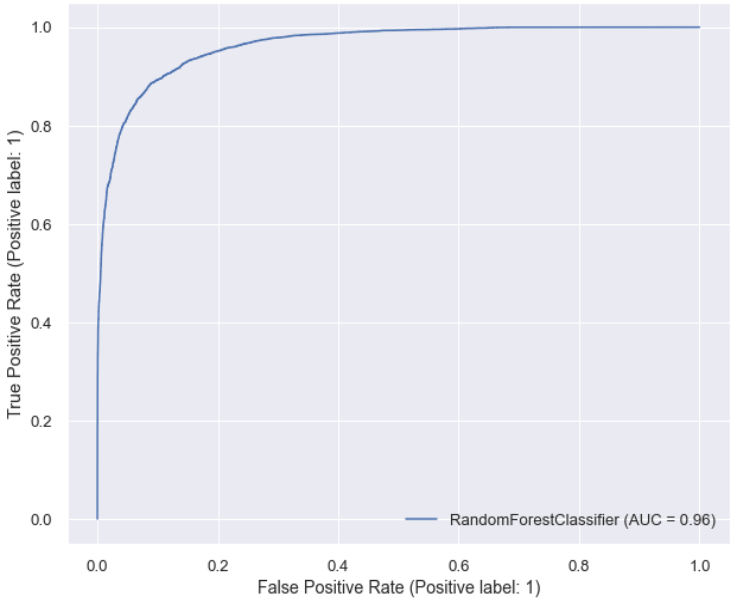
## Random Forest - VIF Data

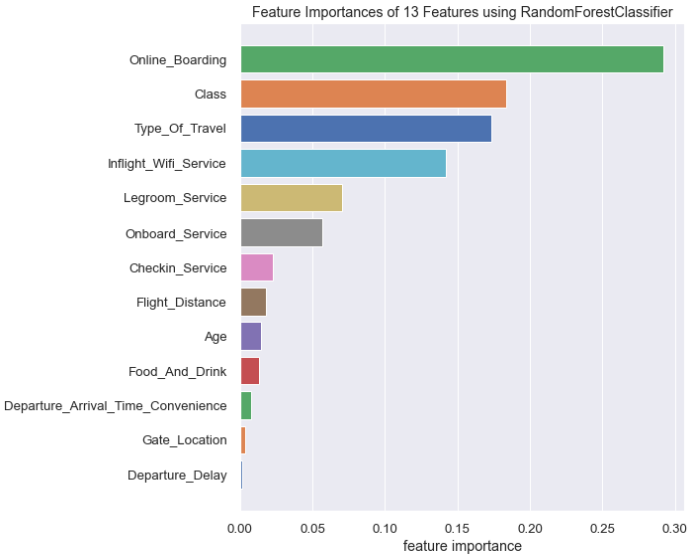



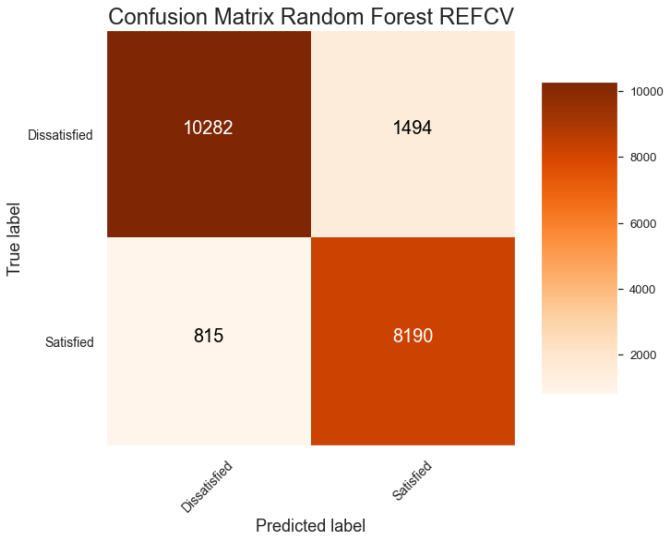



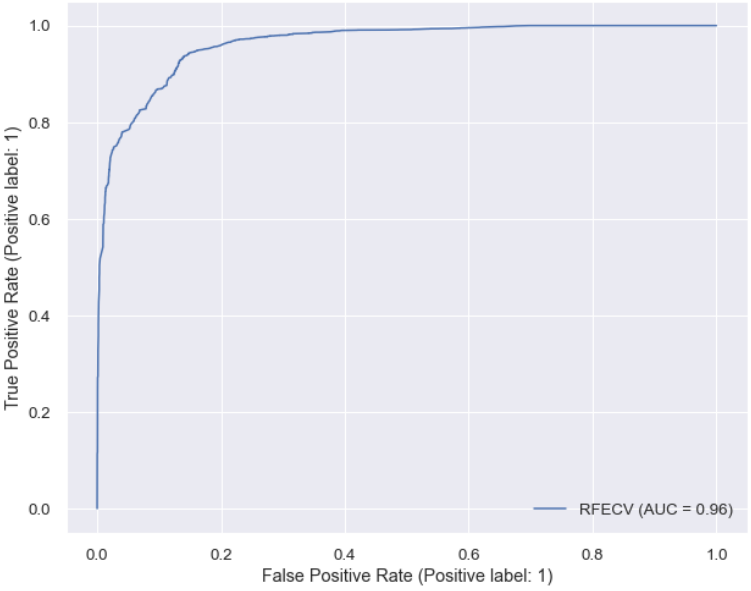




```
Misclassified Counts: 2185
accuracy........ 0.8949
precision score........ 0.8921
recall score........ 0.8956
classification_report
              precision    recall  f1-score   support

           0       0.92      0.89      0.91     11776
           1       0.86      0.90      0.88      9005

    accuracy                           0.89     20781
   macro avg       0.89      0.90      0.89     20781
weighted avg       0.90      0.89      0.90     20781
```

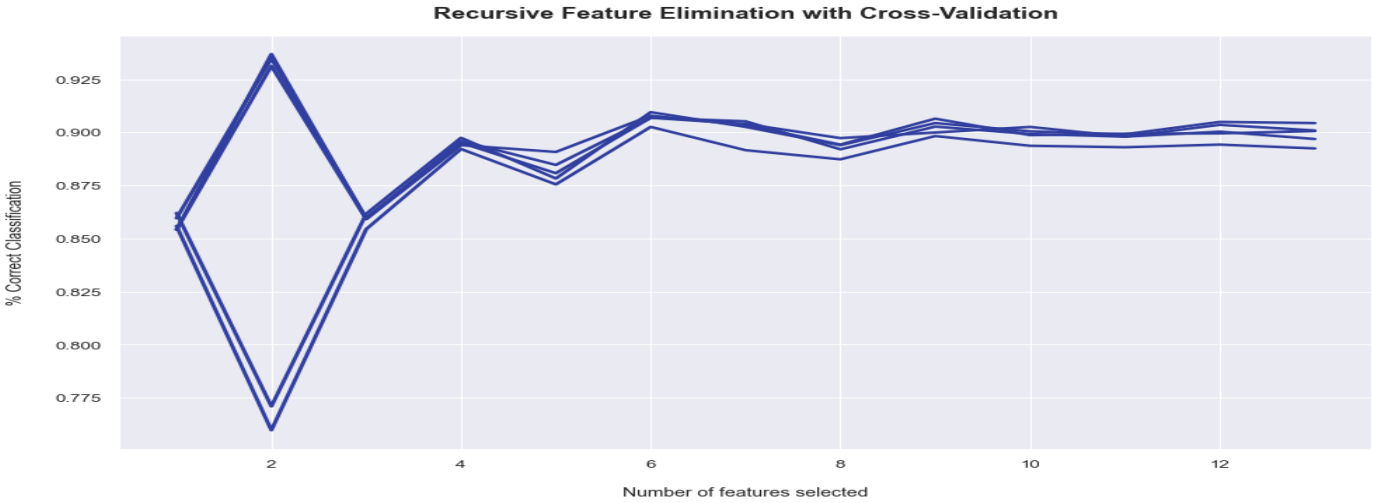
## Random Forest - VIF Data Contd..



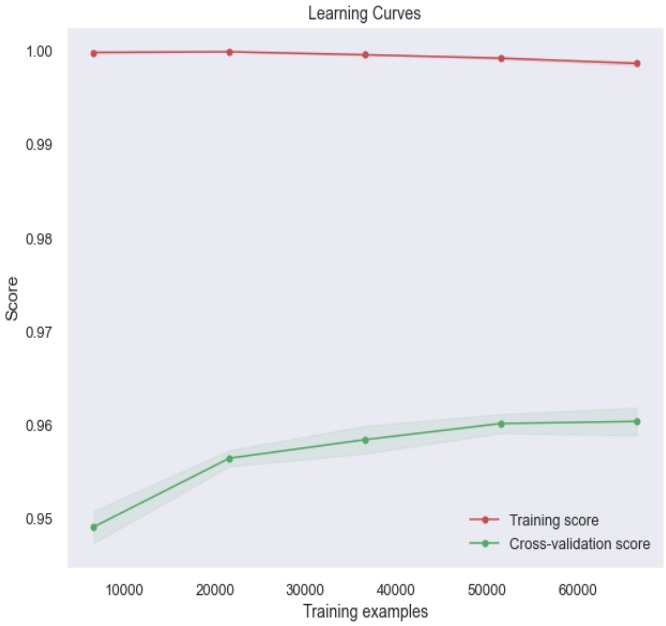



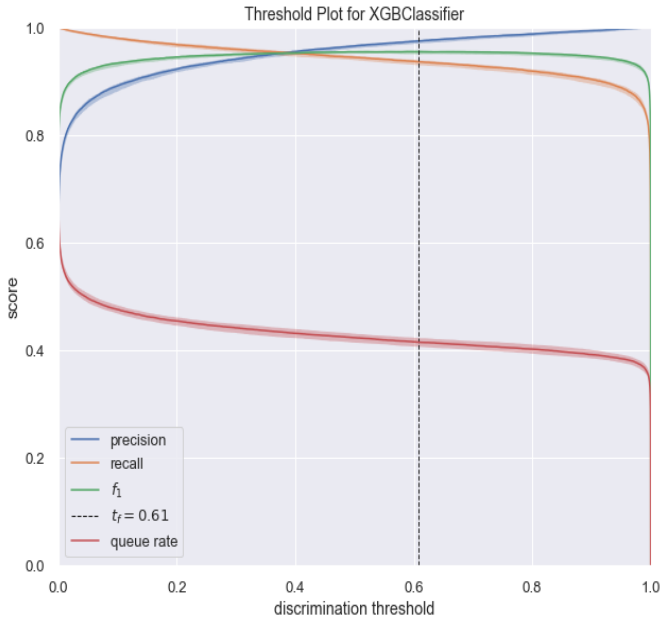



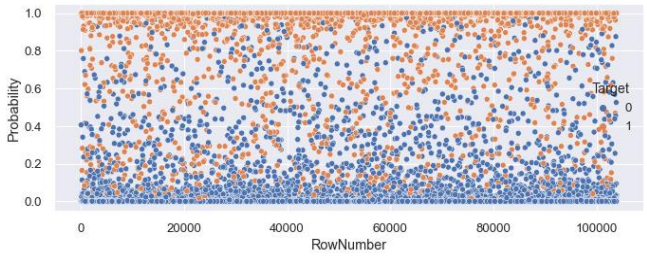

```
Misclassified Counts: 2309
accuracy..... 0.8889
precision score..... 0.8861
recall score..... 0.8913
classification_report
              precision    recall  f1-score   support

           0       0.93      0.87      0.90     11776
           1       0.85      0.91      0.88      9005

    accuracy                           0.89     20781
   macro avg       0.89      0.89      0.89     20781
weighted avg       0.89      0.89      0.89     20781
```

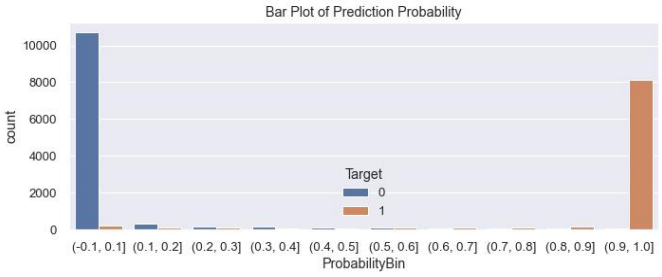


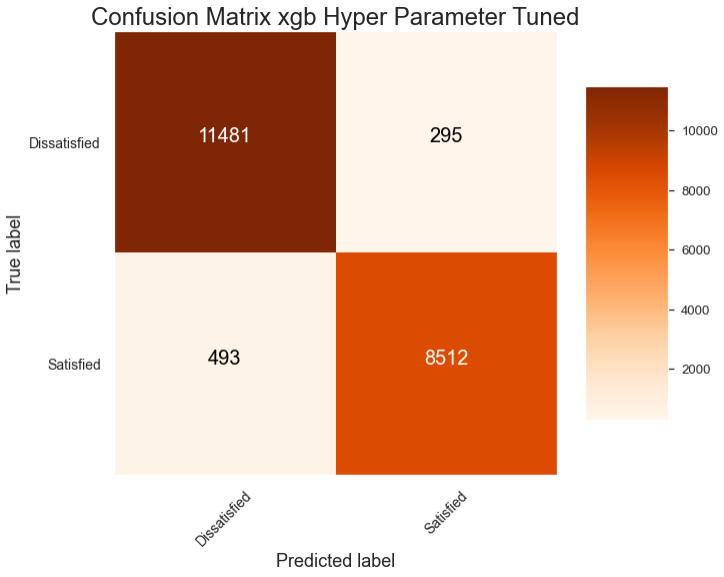
## Xgb - UnBinned Data

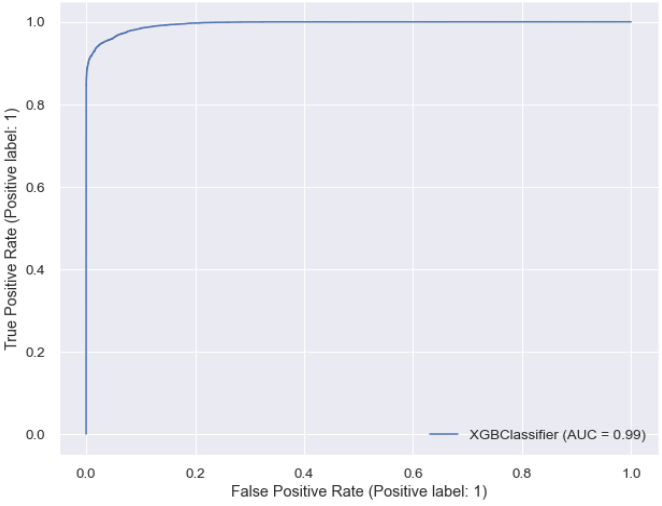


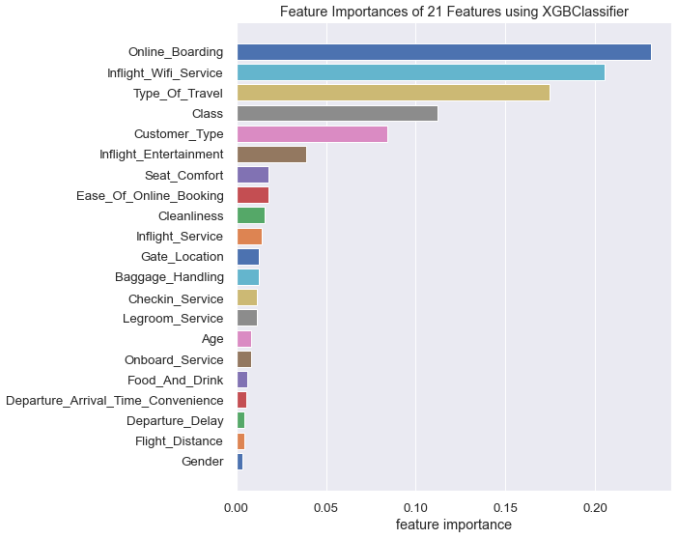



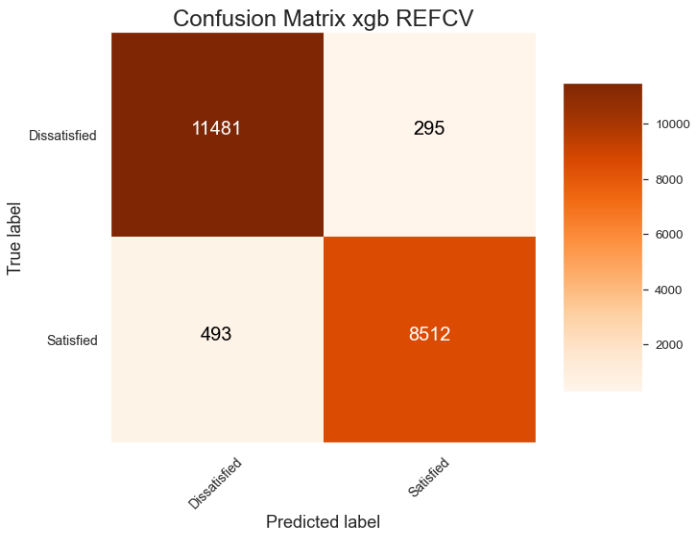



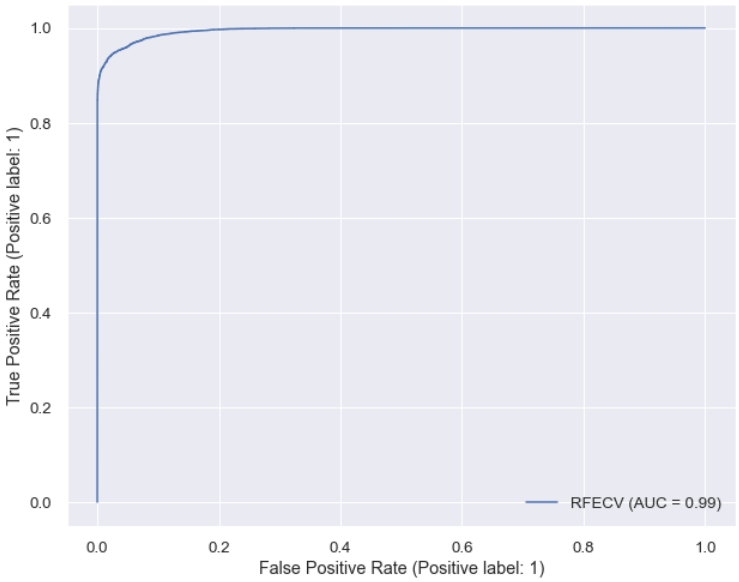




```
Misclassified Counts: 788
accuracy........ 0.9621
precision score........ 0.9627
recall score........ 0.9601
classification_report
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     11776
           1       0.97      0.95      0.96      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

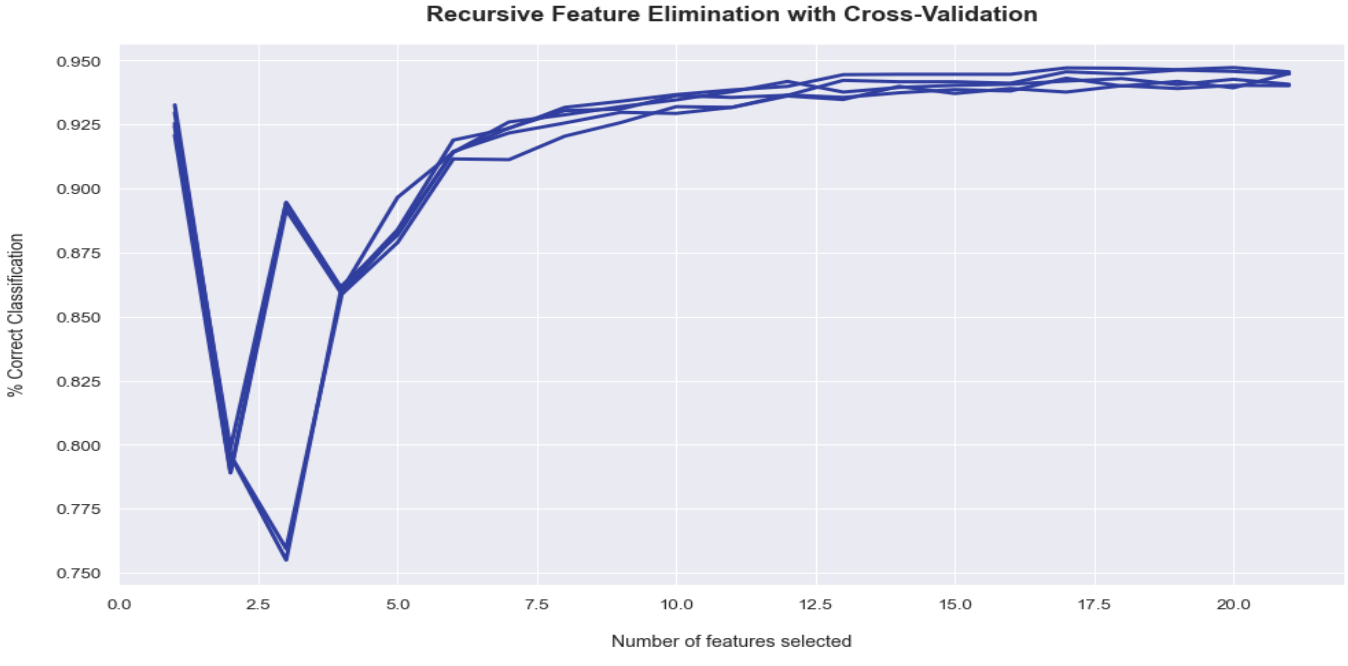
## Xgb - UnBinned Data Contd..



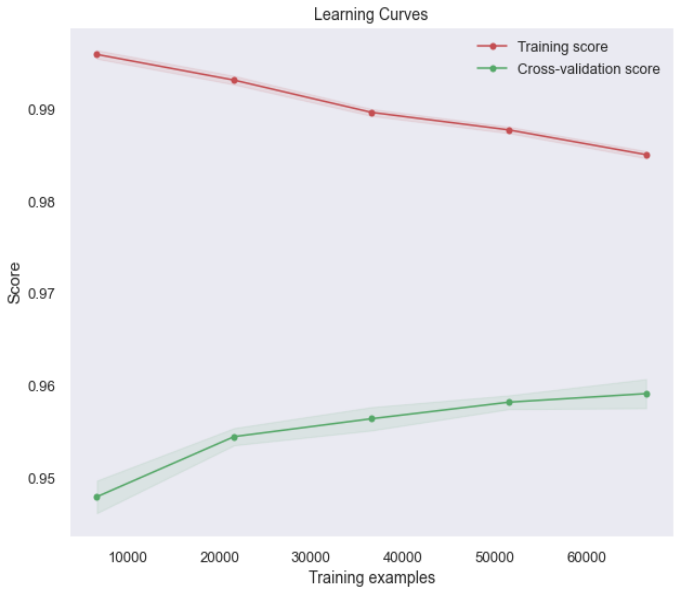



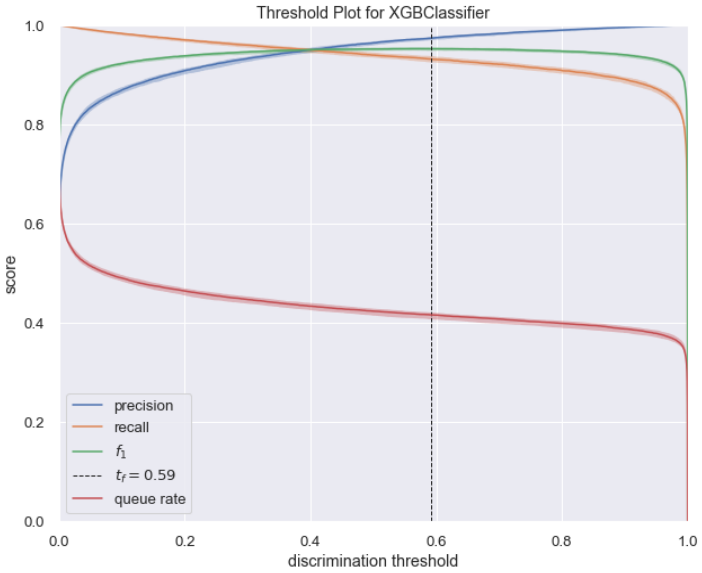



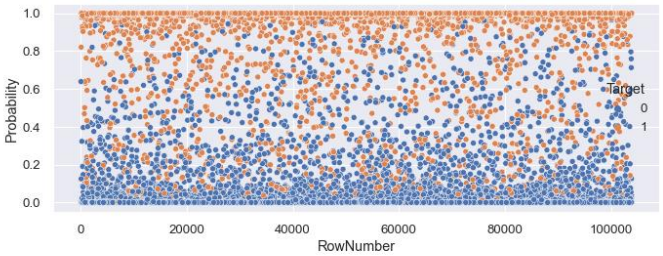

```
Misclassified Counts: 788
accuracy........ 0.9621
precision score........ 0.9627
recall score........ 0.9601
classification_report
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     11776
           1       0.97      0.95      0.96      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

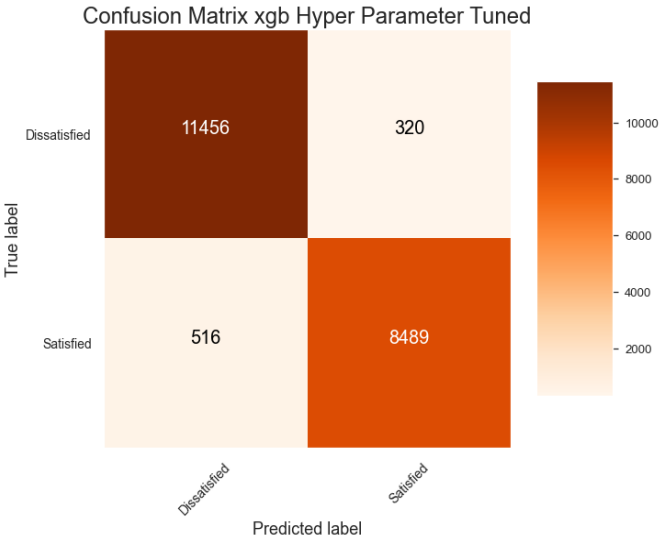


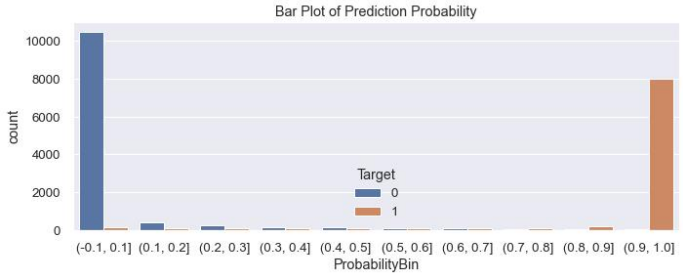
## XGB-Binned Data

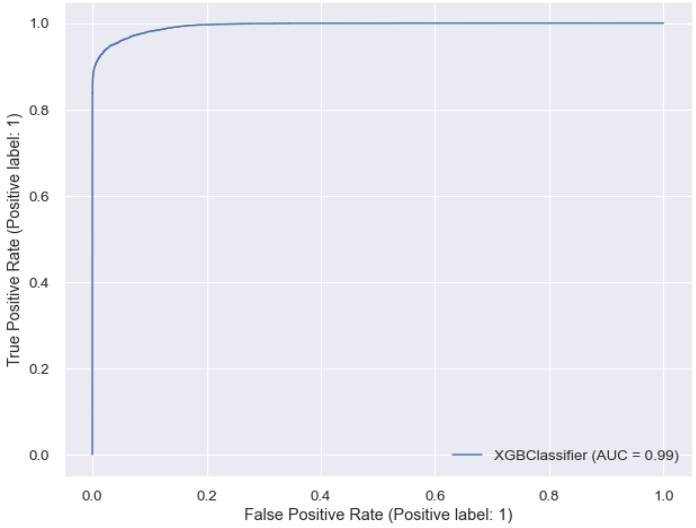


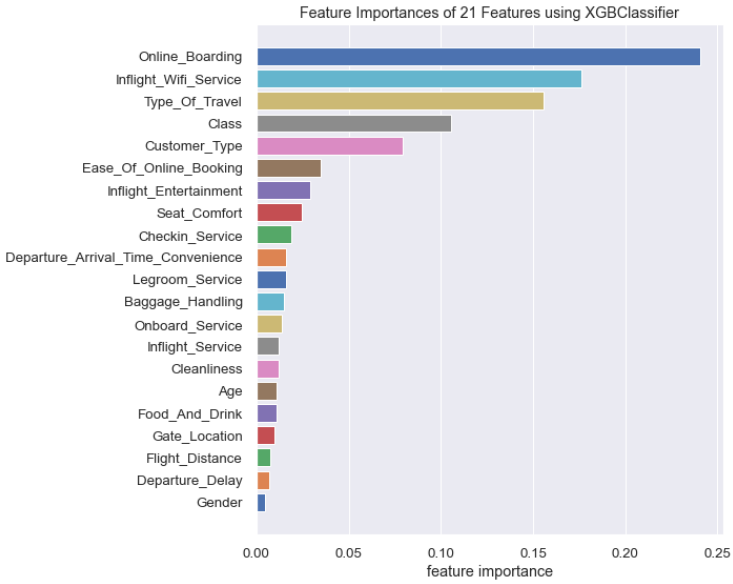



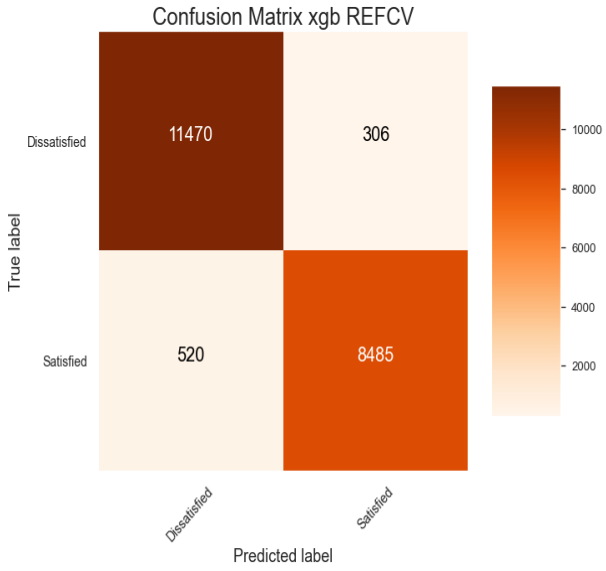



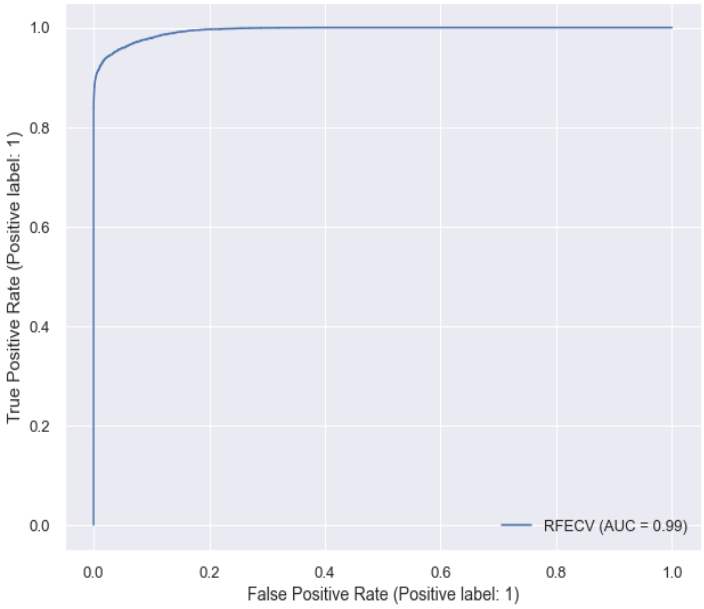




```
Misclassified Counts: 836
accuracy........ 0.9598
precision score........ 0.9603
recall score........ 0.9578
classification_report
              precision    recall  f1-score   support

           0       0.96      0.97      0.96     11776
           1       0.96      0.94      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

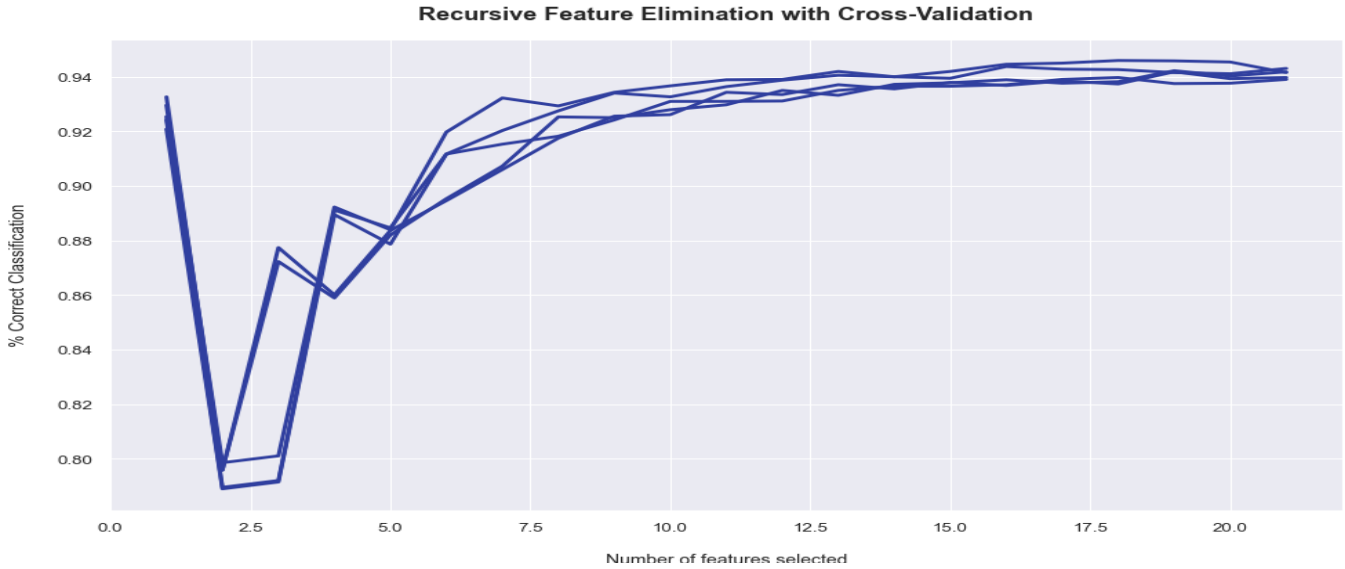
## XGB-Binned Data Contd..



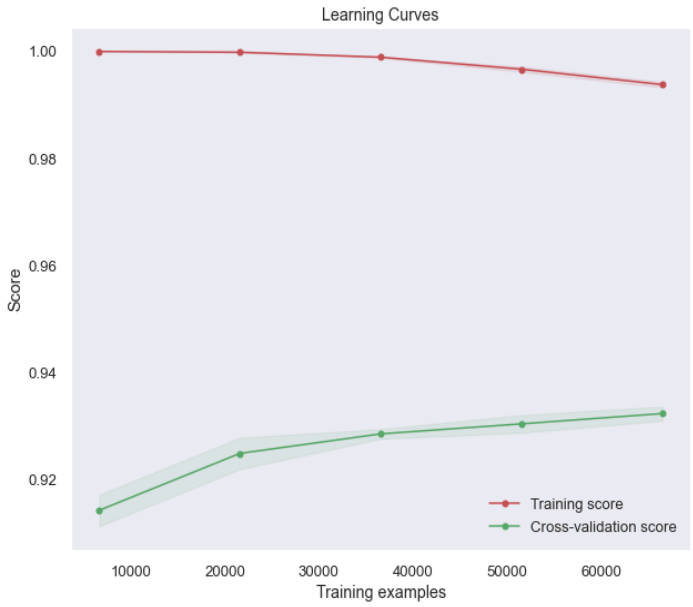



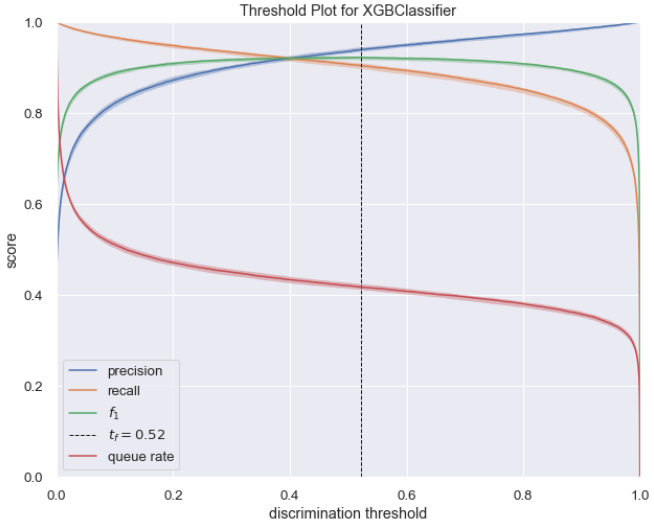



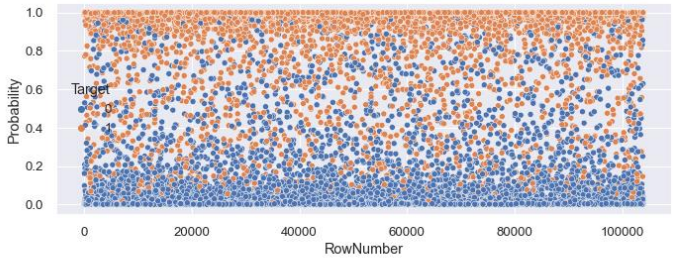

```
Misclassified Counts: 826
accuracy........ 0.9603
precision score........ 0.9609
recall score........ 0.9581
classification_report
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     11776
           1       0.97      0.94      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

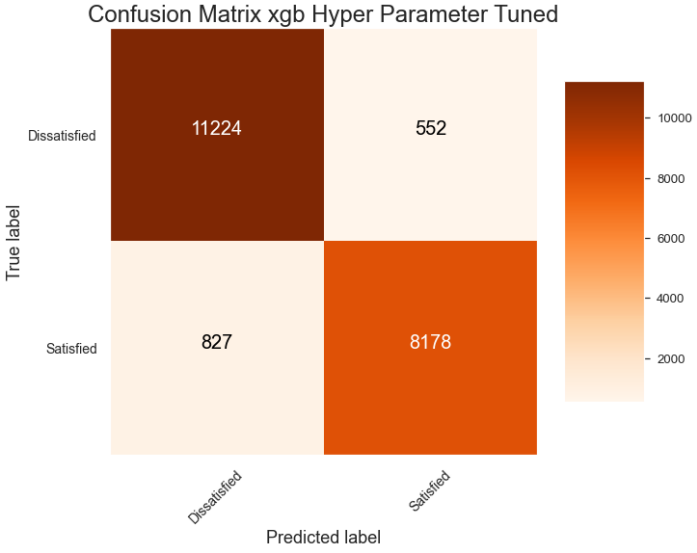


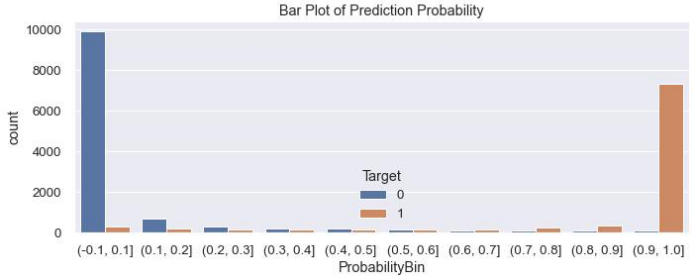
## XGB PCA Data

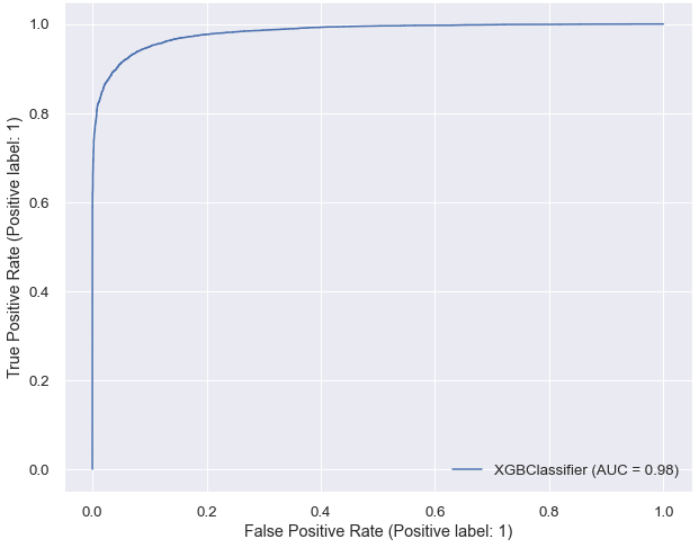


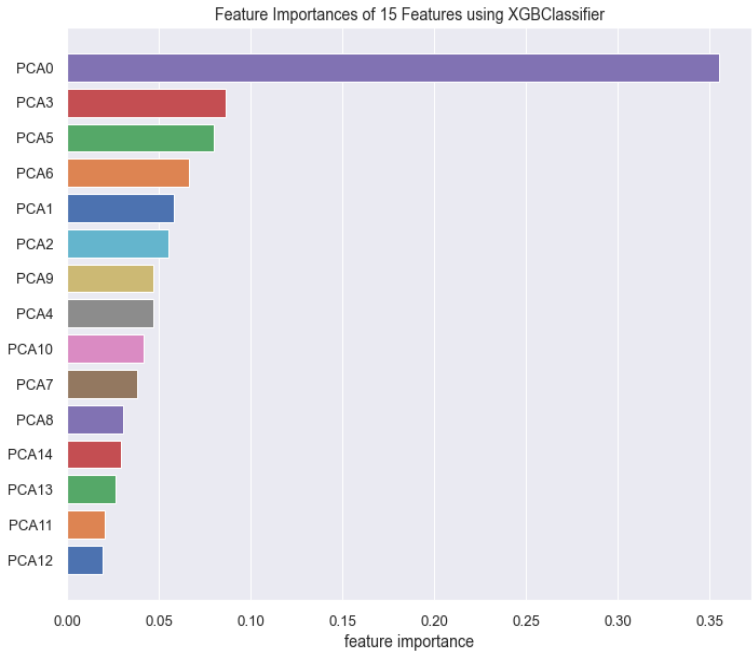



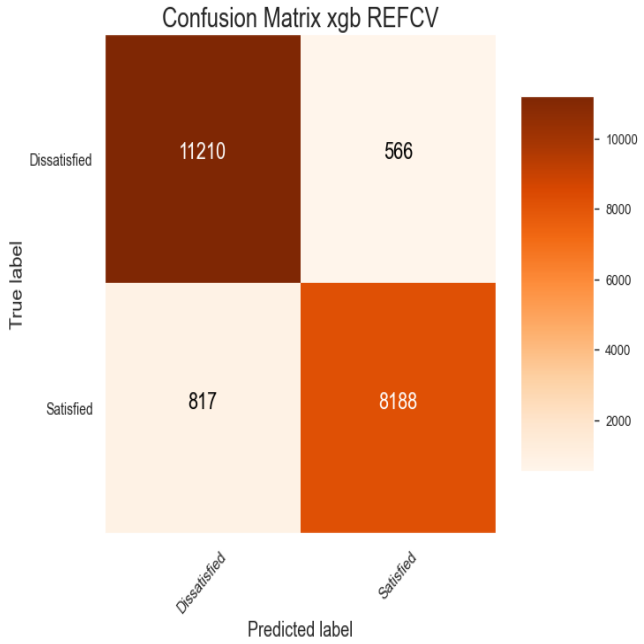



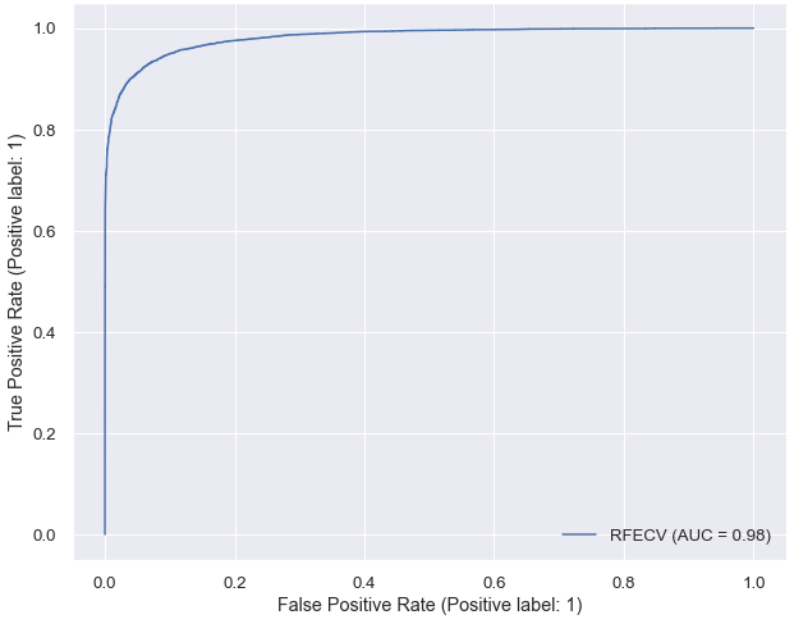




```
Misclassified Counts: 1379
accuracy........ 0.9336
precision score........ 0.9341
recall score........ 0.9306
classification_report
              precision    recall  f1-score   support

           0       0.93      0.95      0.94     11776
           1       0.94      0.91      0.92      9005

    accuracy                           0.93     20781
   macro avg       0.93      0.93      0.93     20781
weighted avg       0.93      0.93      0.93     20781
```

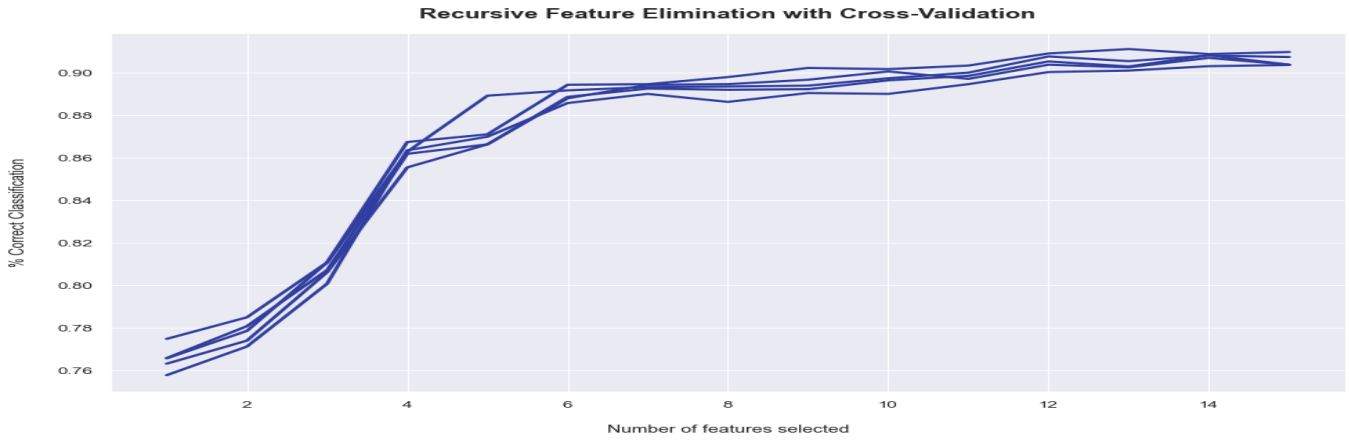
## XGB PCA Contd



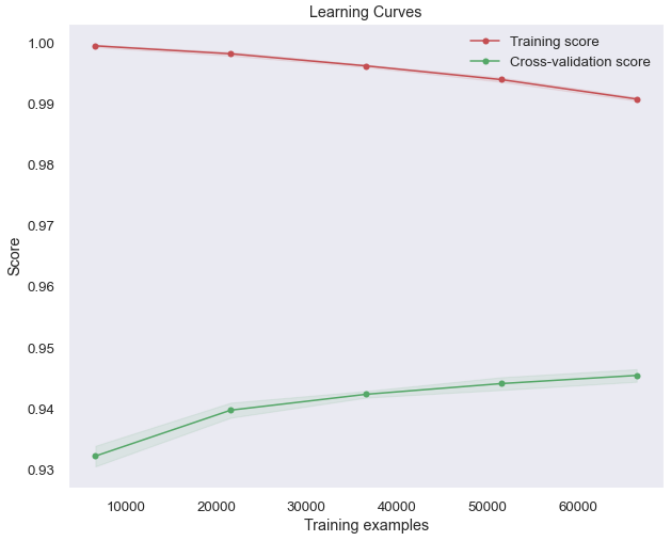



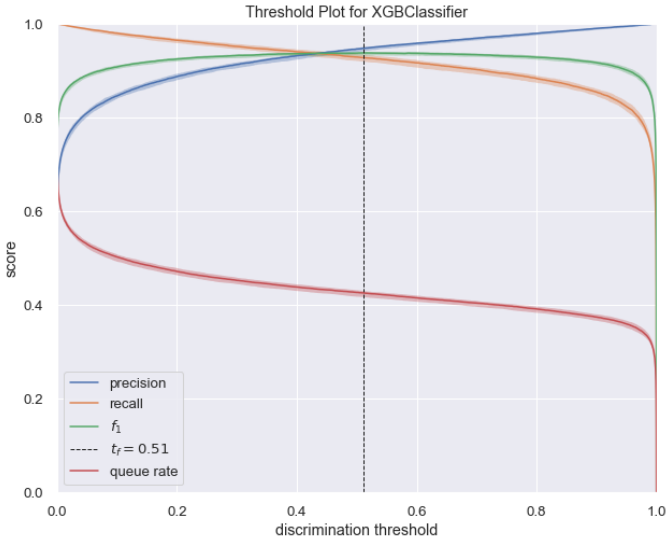



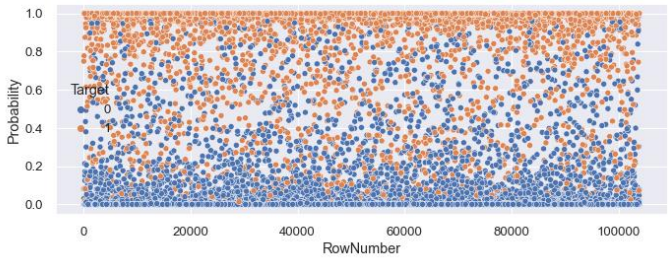

```
Misclassified Counts: 1383
accuracy..... 0.9334
precision score..... 0.9337
recall score..... 0.9306
classification_report
              precision    recall  f1-score   support

           0       0.93      0.95      0.94     11776
           1       0.94      0.91      0.92      9005

    accuracy                           0.93     20781
   macro avg       0.93      0.93      0.93     20781
weighted avg       0.93      0.93      0.93     20781
```

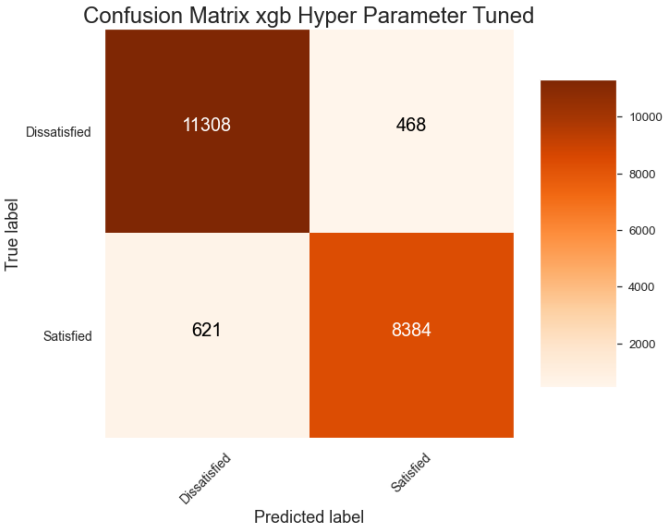


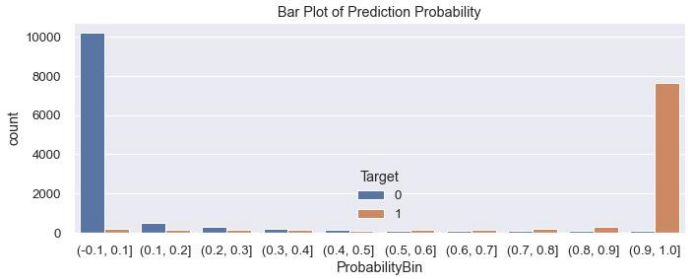
## XGB VIF Data

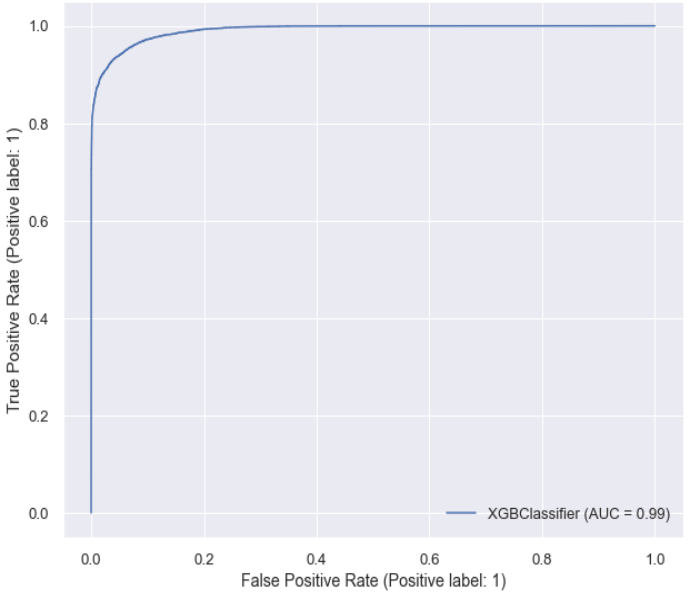


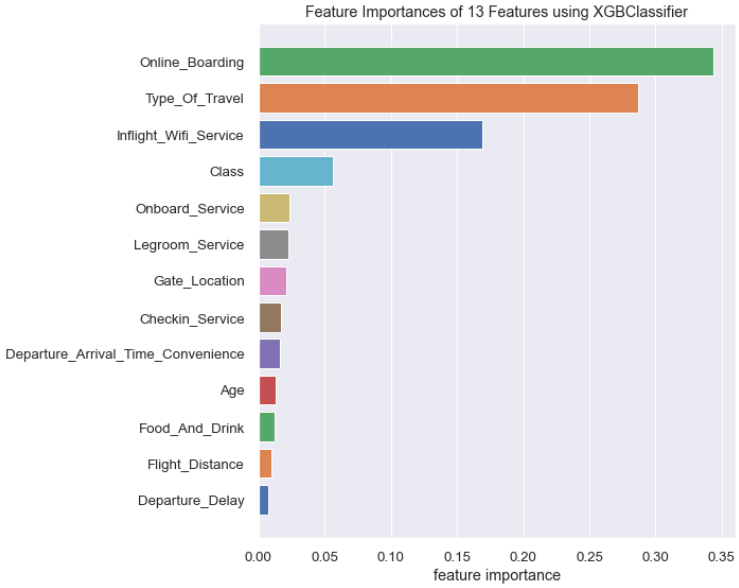



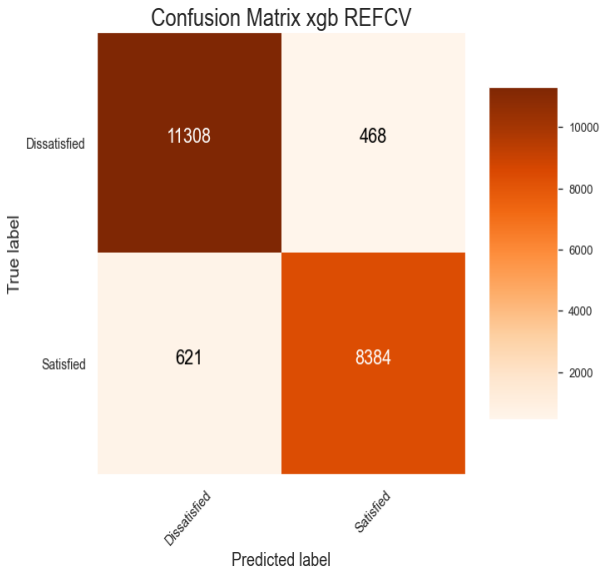



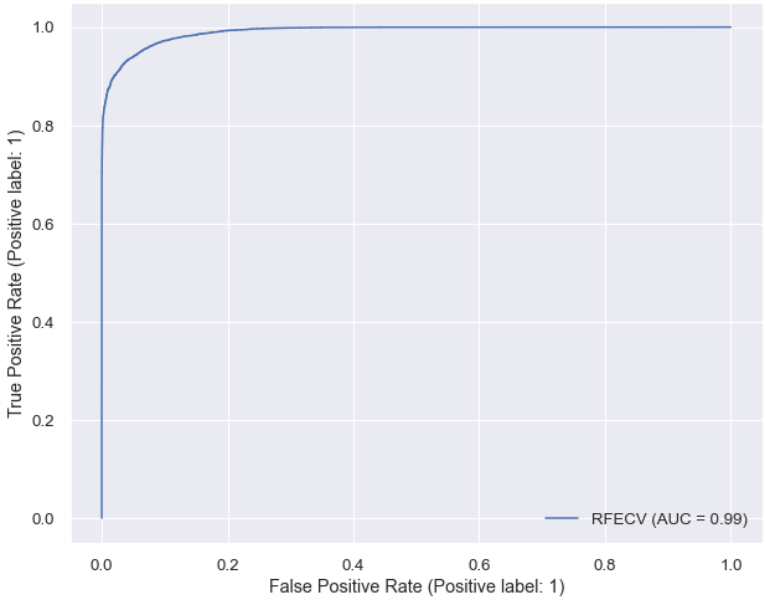




```
Misclassified Counts: 1089
accuracy........ 0.9476
precision score........ 0.9475
recall score........ 0.9456
classification_report
              precision    recall  f1-score   support

           0       0.95      0.96      0.95     11776
           1       0.95      0.93      0.94      9005

    accuracy                           0.95     20781
   macro avg       0.95      0.95      0.95     20781
weighted avg       0.95      0.95      0.95     20781
```

**XGB VIF Data Contd..**

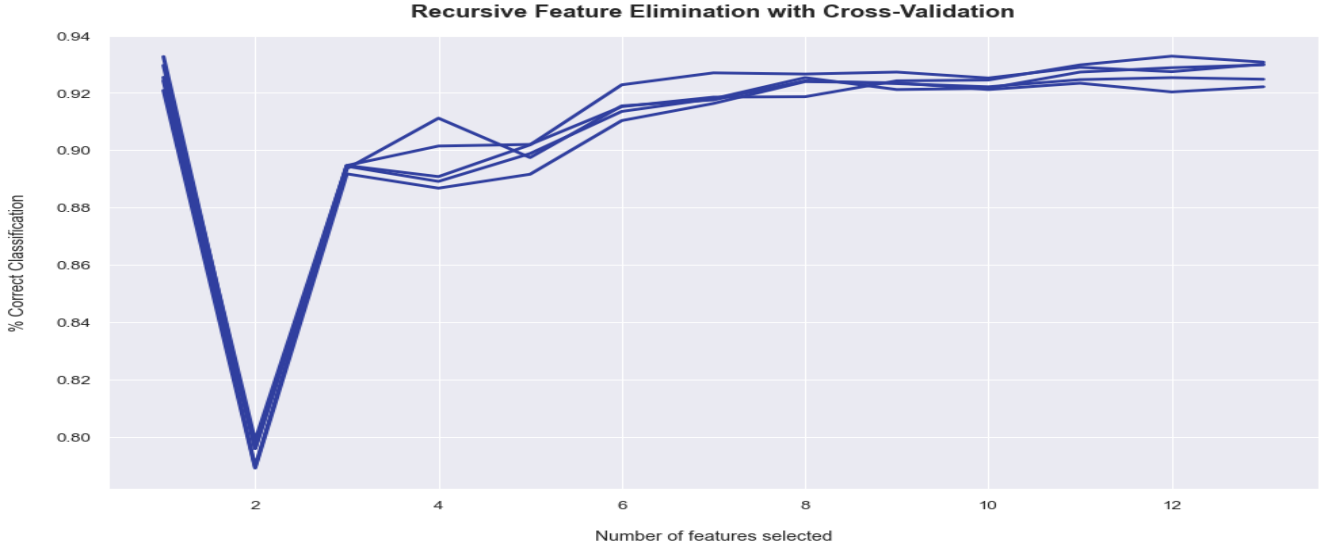


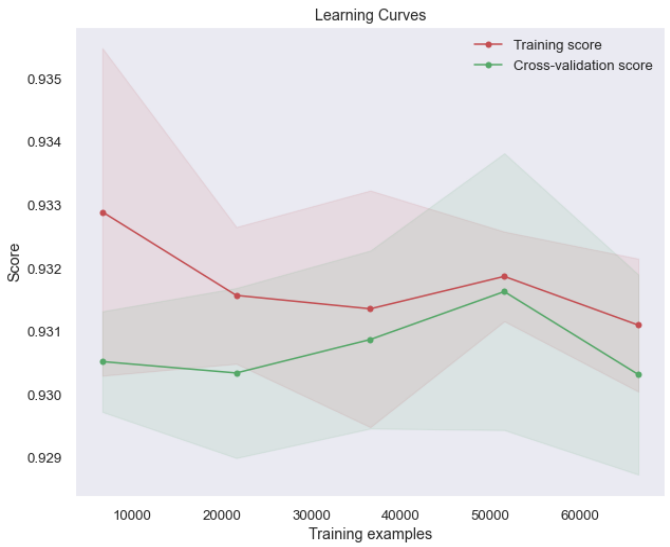



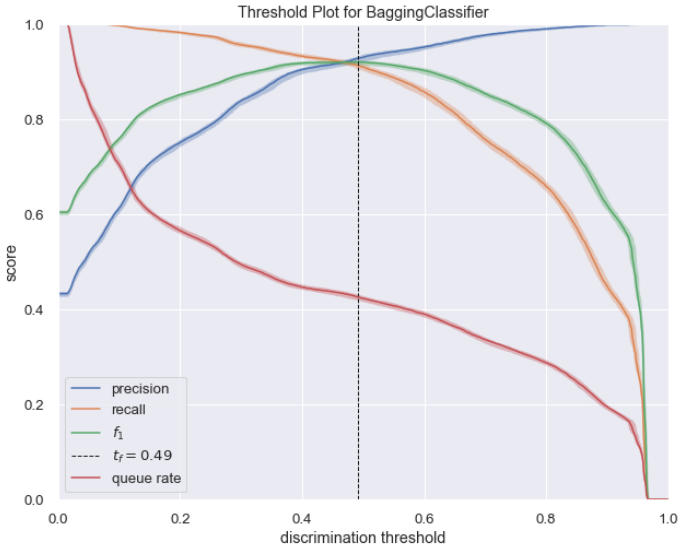

Misclassified Counts: 1089
accuracy..... 0.9476
precision score..... 0.9475
recall score..... 0.9456
classification_report

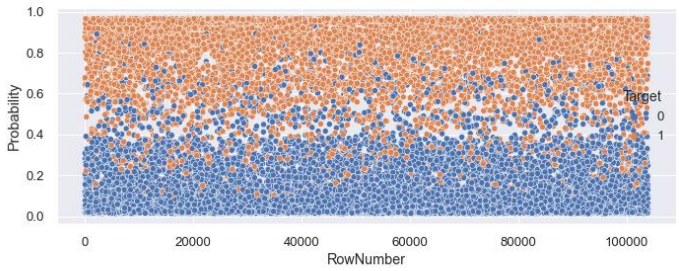
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.96 | 0.95 | 11776 |
| 1 | 0.95 | 0.93 | 0.94 | 9005 |
| accuracy | | | 0.95 | 20781 |
| macro avg | 0.95 | 0.95 | 0.95 | 20781 |
| weighted avg | 0.95 | 0.95 | 0.95 | 20781 |

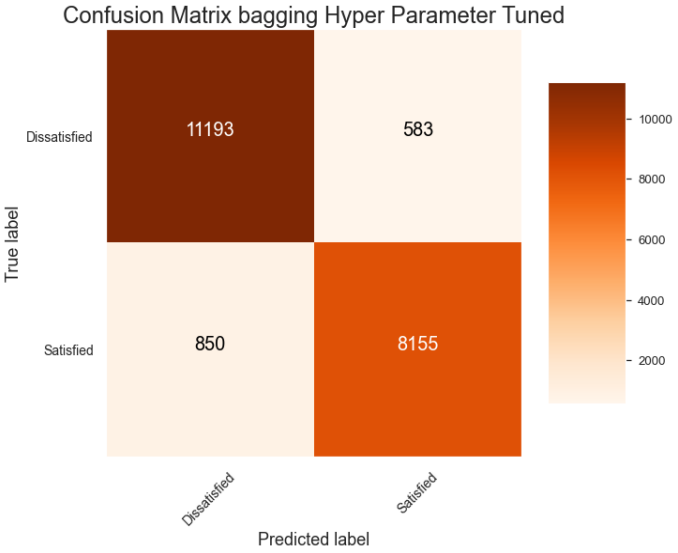


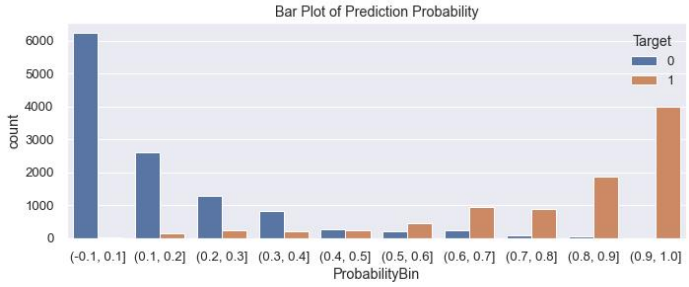
## Bagging UnBinned Data

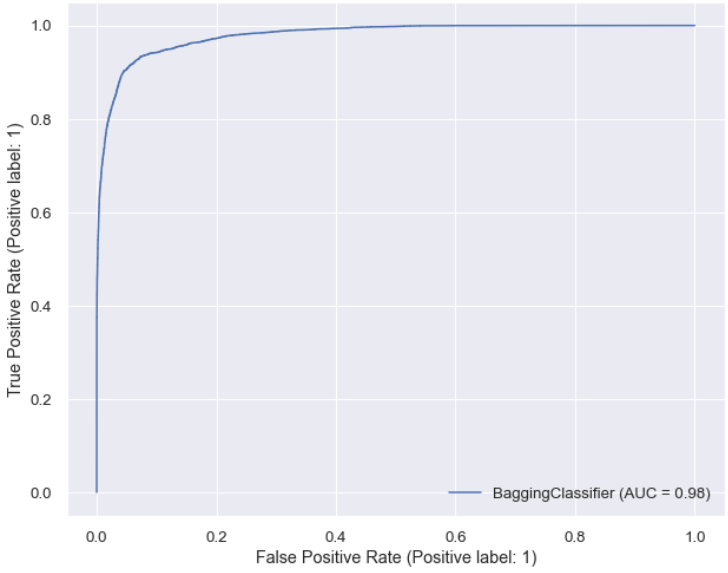


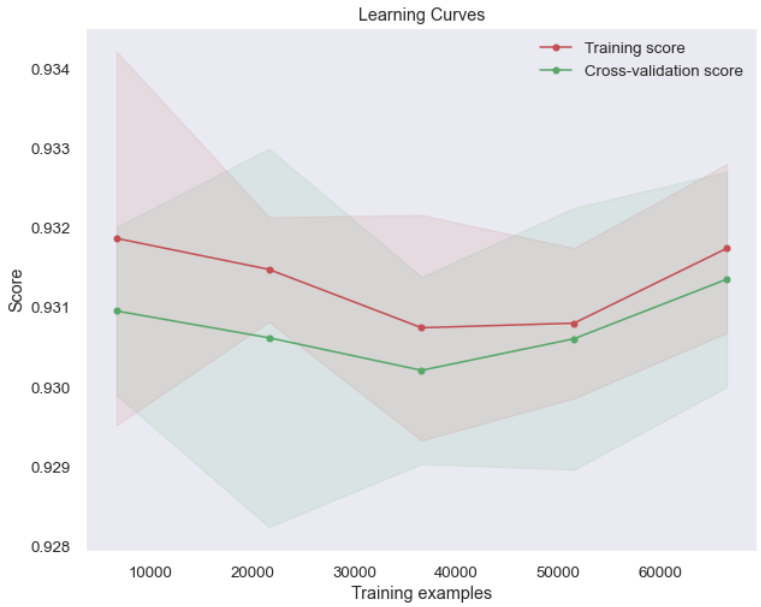



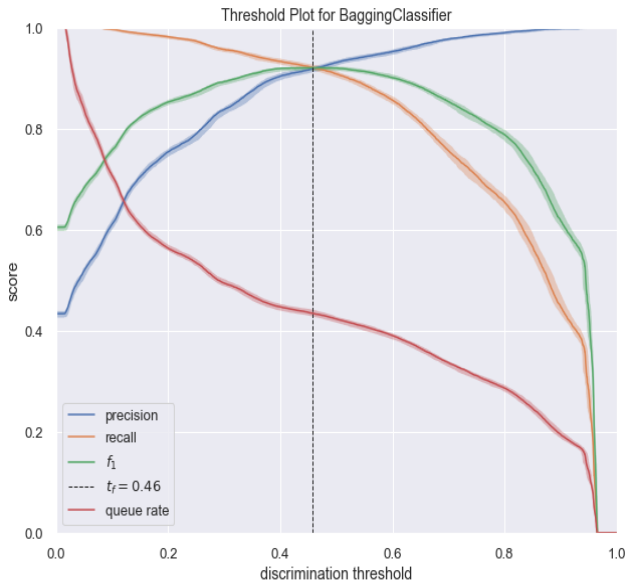



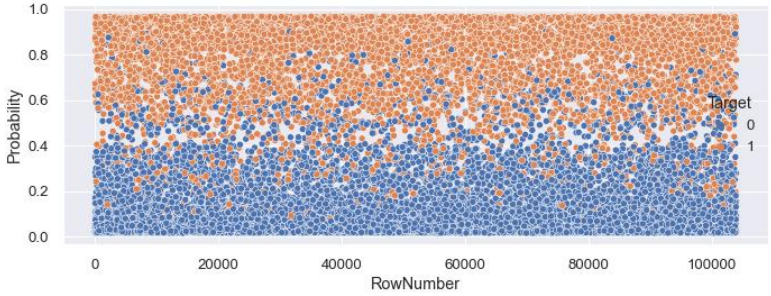



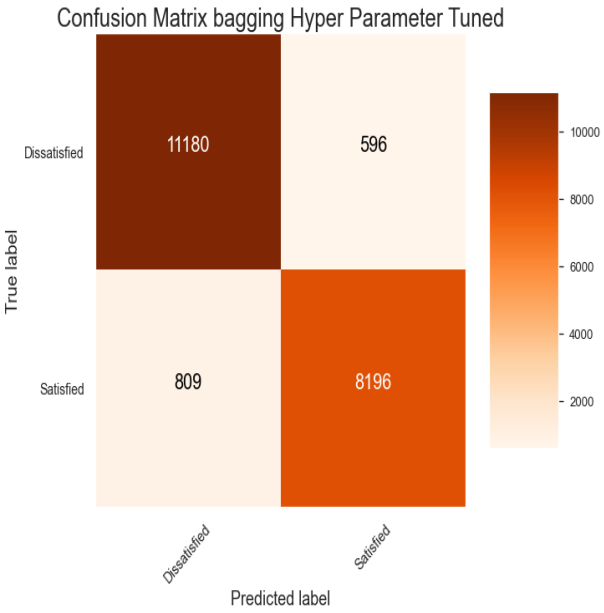

```
Misclassified Counts: 1433
accuracy........ 0.9310
precision score........ 0.9313
recall score........ 0.9281
classification_report
              precision    recall  f1-score   support

           0       0.93      0.95      0.94     11776
           1       0.93      0.91      0.92      9005

    accuracy                           0.93     20781
   macro avg       0.93      0.93      0.93     20781
weighted avg       0.93      0.93      0.93     20781
```

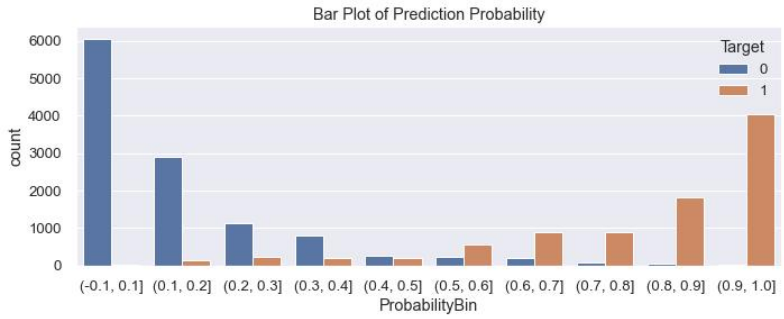
## Bagging Binned Data

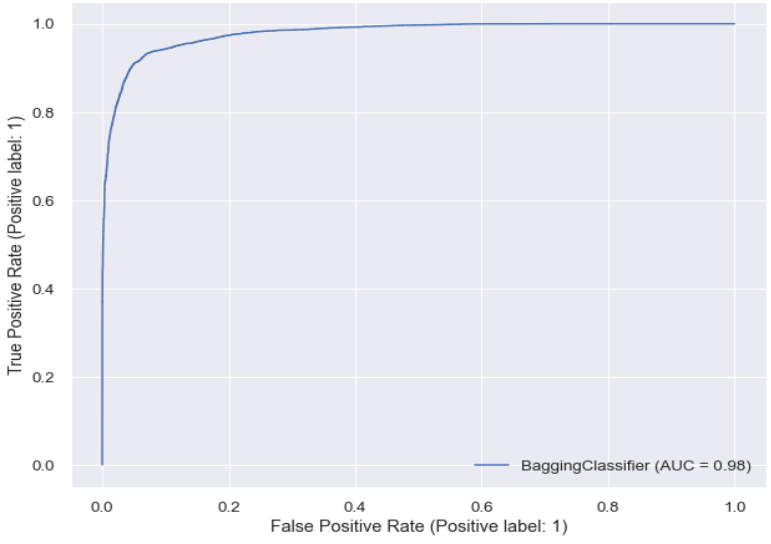


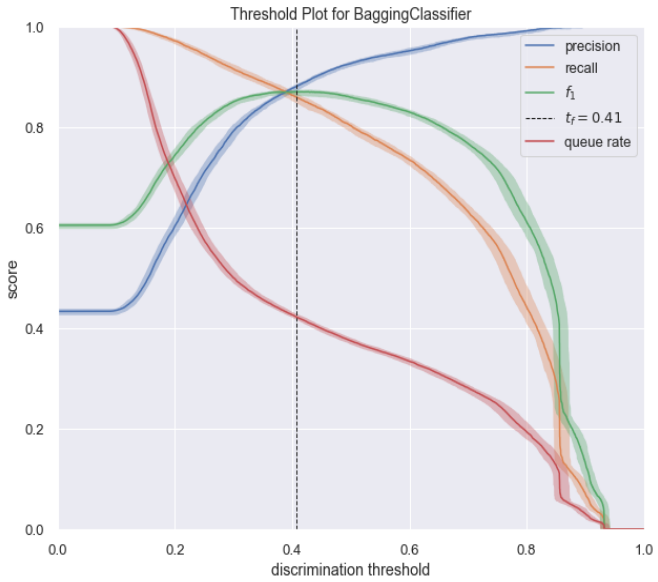



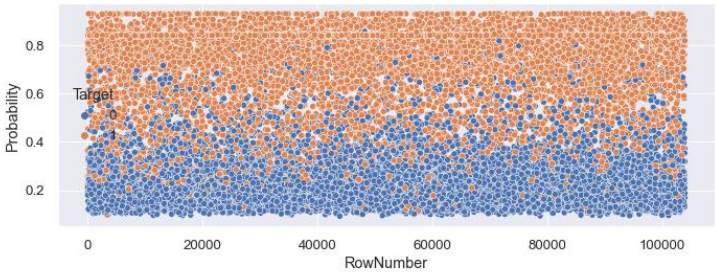



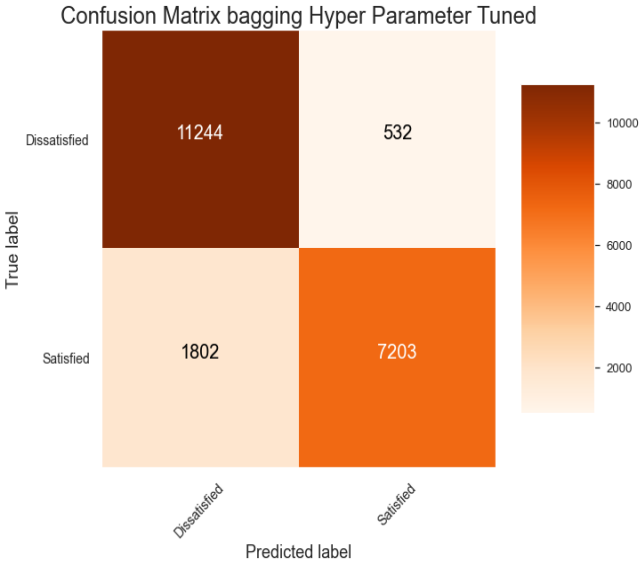



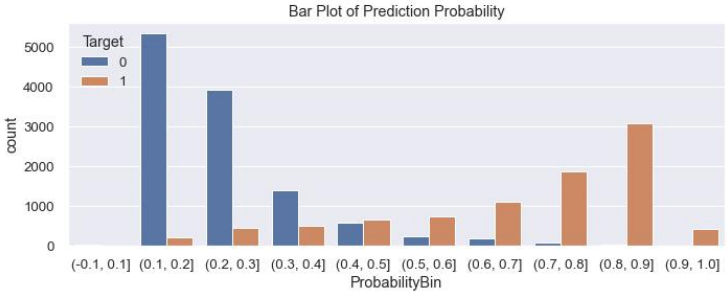

```
Misclassified Counts: 1405
accuracy........ 0.9324
precision score........ 0.9324
recall score........ 0.9298
classification_report
              precision    recall  f1-score   support

           0       0.93      0.95      0.94     11776
           1       0.93      0.91      0.92      9005

    accuracy                           0.93     20781
   macro avg       0.93      0.93      0.93     20781
weighted avg       0.93      0.93      0.93     20781
```

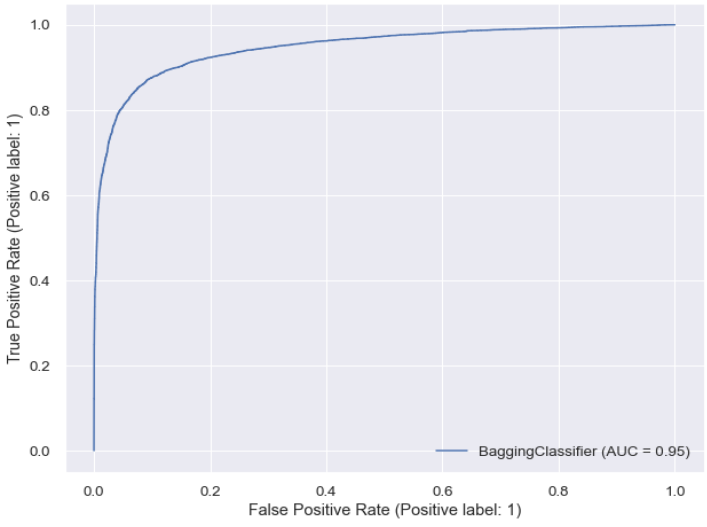
## Bagging PCA Data

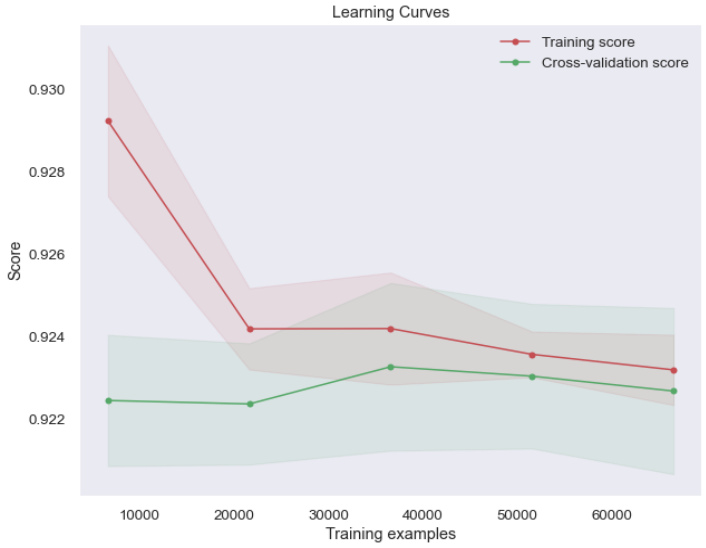



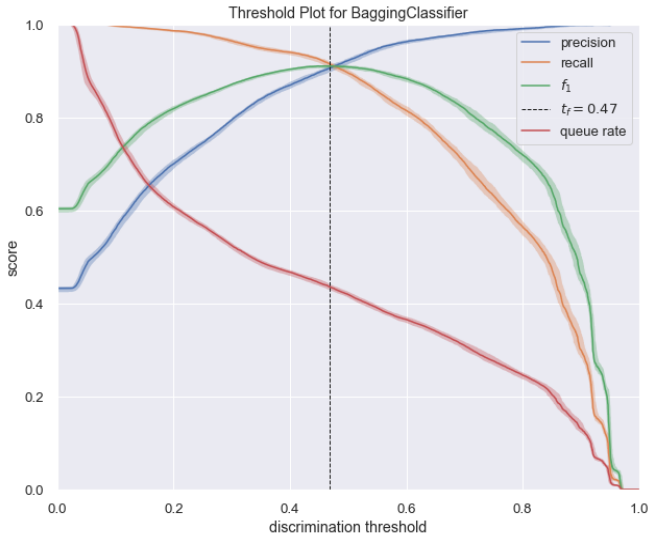



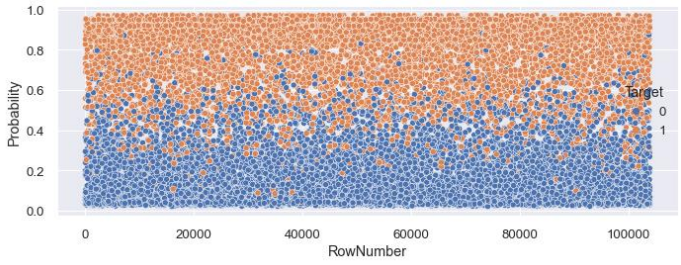



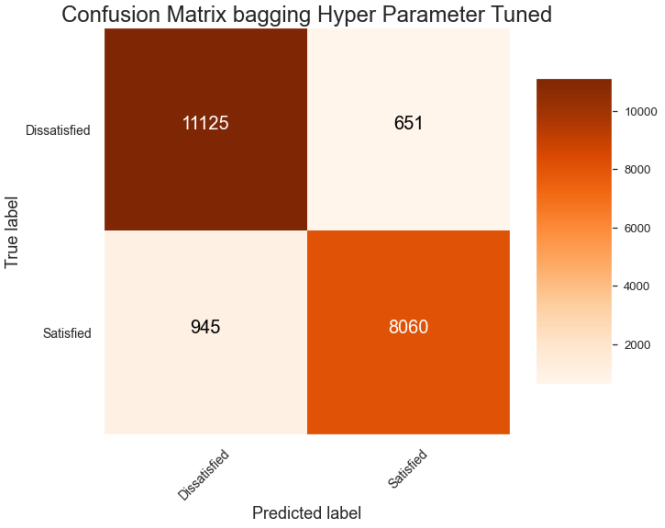

```
Misclassified Counts: 2334
accuracy........ 0.8877
precision score........ 0.8965
recall score........ 0.8774
classification_report
              precision    recall  f1-score   support

           0       0.86      0.95      0.91     11776
           1       0.93      0.80      0.86      9005

    accuracy                           0.89     20781
   macro avg       0.90      0.88      0.88     20781
weighted avg       0.89      0.89      0.89     20781
```

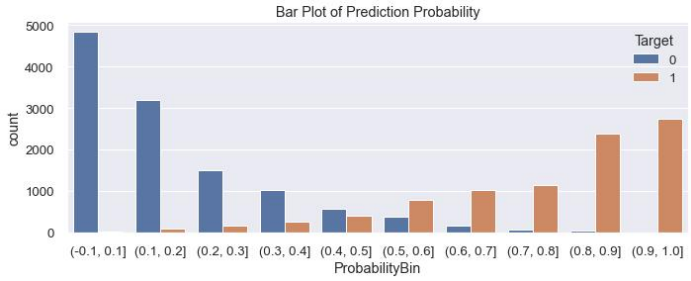
## Bagging VIF Data

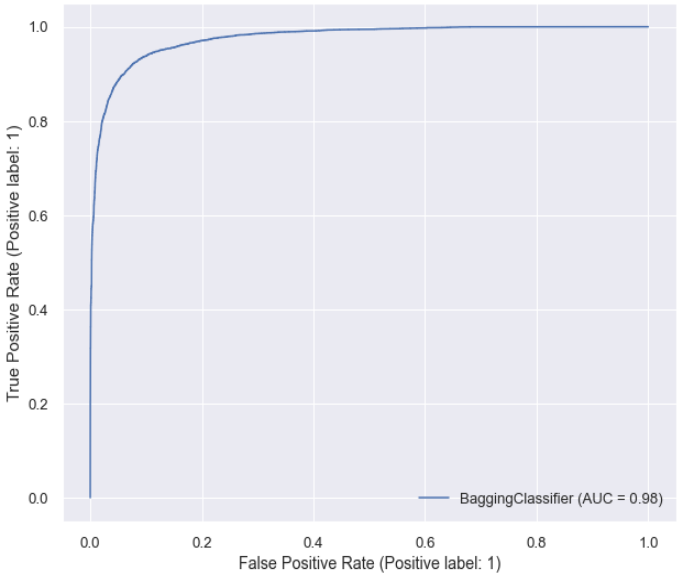


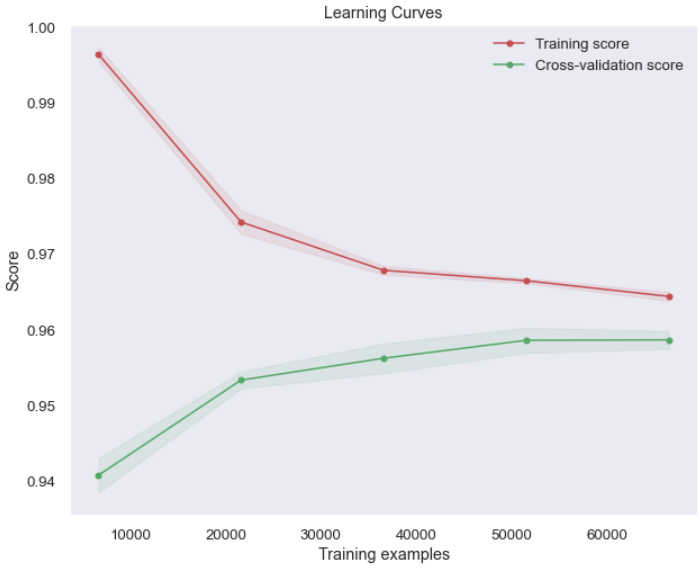



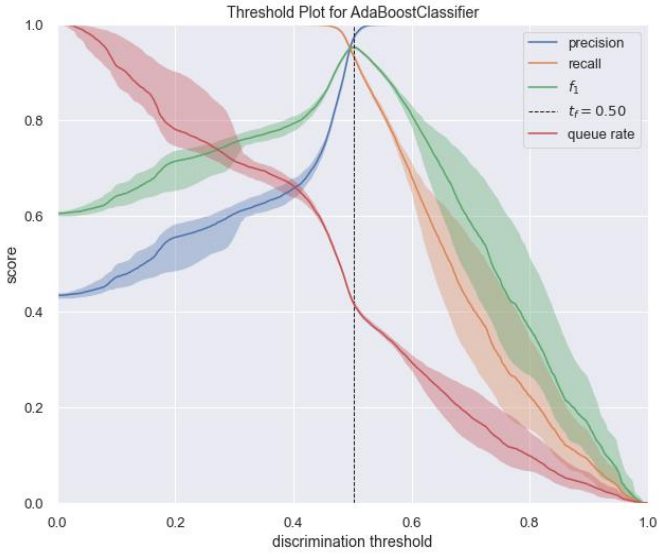



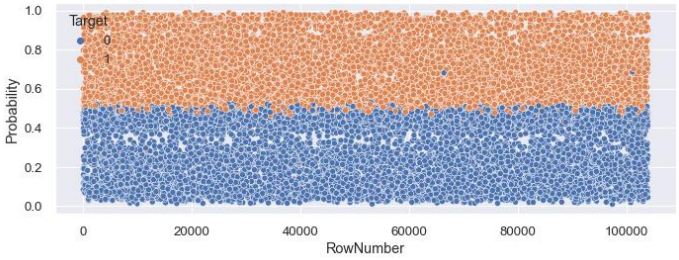



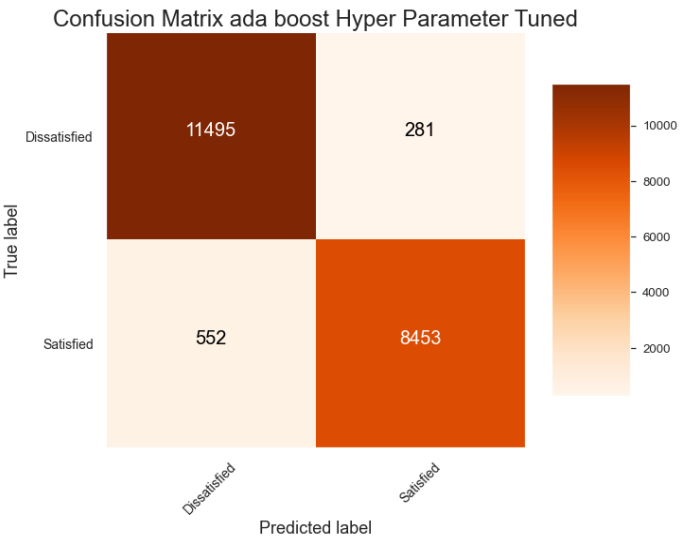

```
Misclassified Counts: 1596
accuracy........ 0.9232
precision score........ 0.9235
recall score........ 0.9199
classification_report
              precision    recall  f1-score   support

           0       0.92      0.94      0.93     11776
           1       0.93      0.90      0.91      9005

    accuracy                           0.92     20781
   macro avg       0.92      0.92      0.92     20781
weighted avg       0.92      0.92      0.92     20781
```

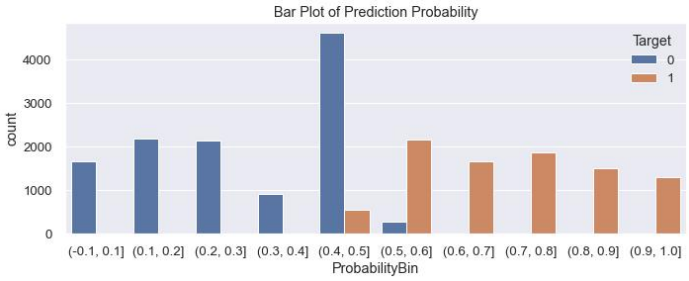
## ada boost-UnBinned Data

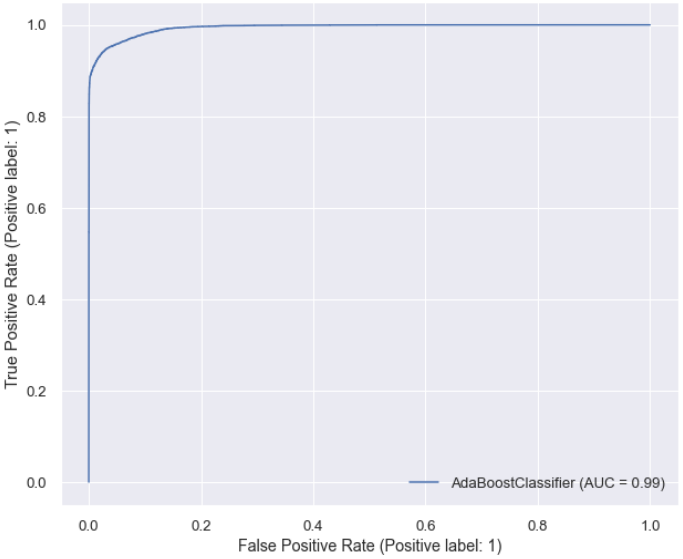


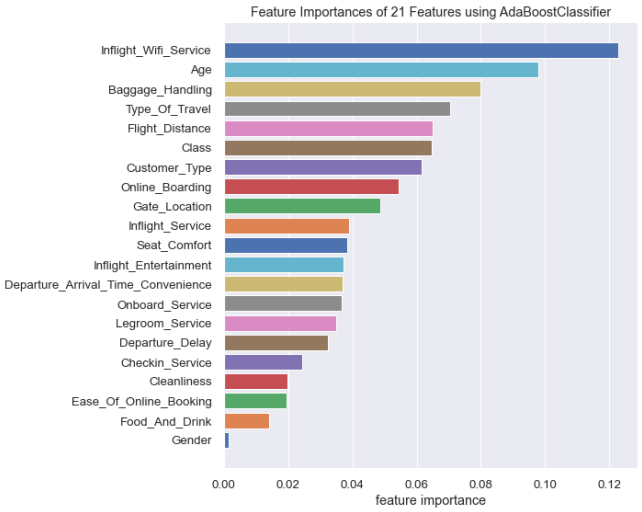



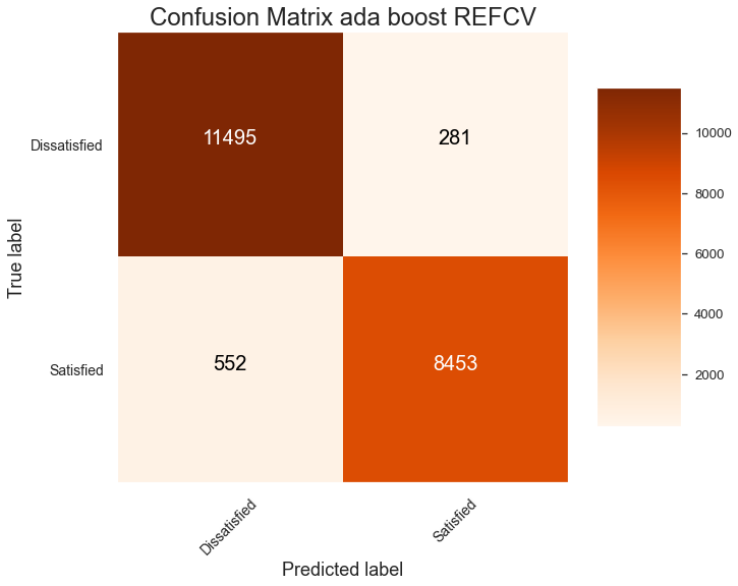



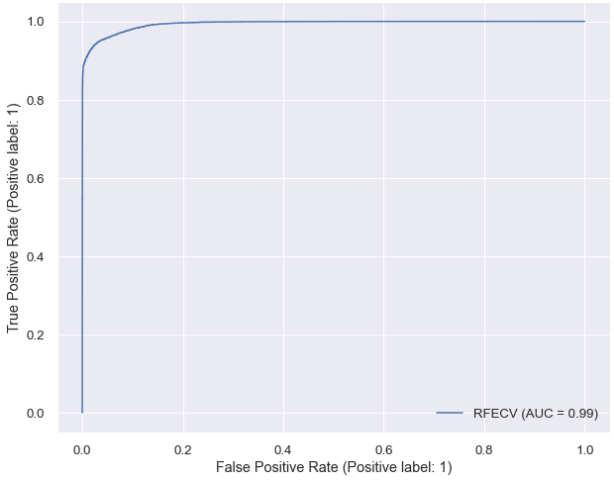




```
Misclassified Counts: 833
accuracy........ 0.9599
precision score........ 0.9610
recall score........ 0.9574
classification_report
              precision    recall  f1-score   support

           0       0.95      0.98      0.97     11776
           1       0.97      0.94      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

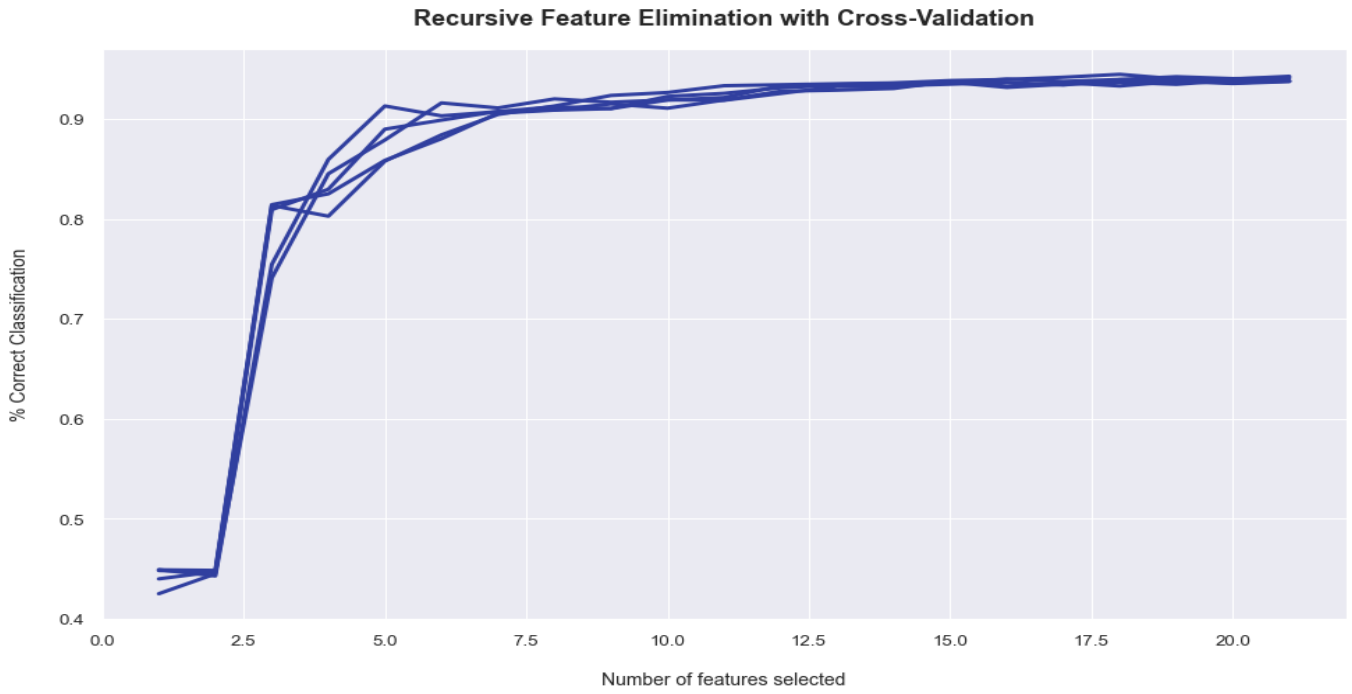
**ada boost-UnBinned Data Contd..**



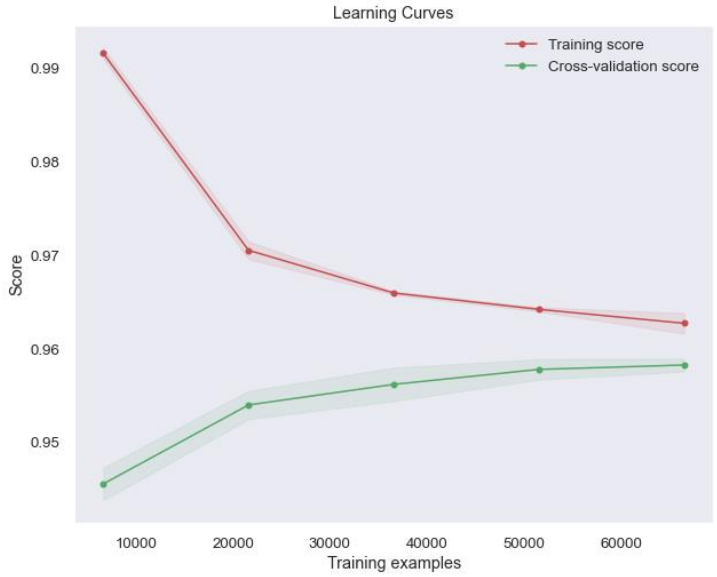



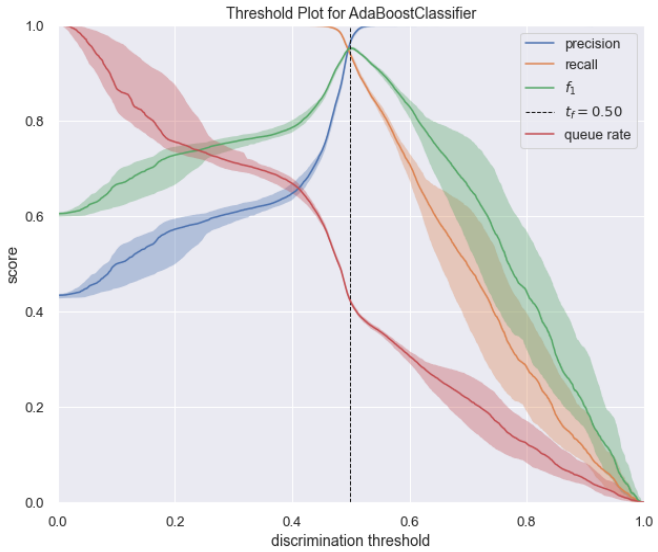



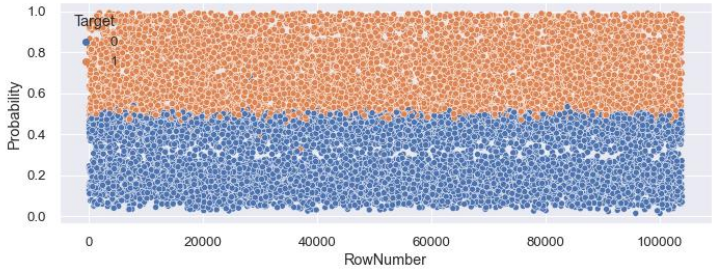

```
Misclassified Counts: 833
accuracy........ 0.9599
precision score........ 0.9610
recall score........ 0.9574
classification_report
              precision    recall  f1-score   support

           0       0.95      0.98      0.97     11776
           1       0.97      0.94      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

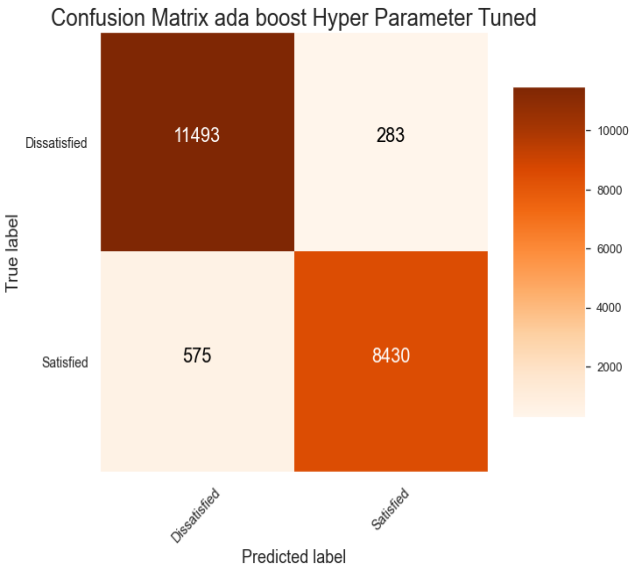


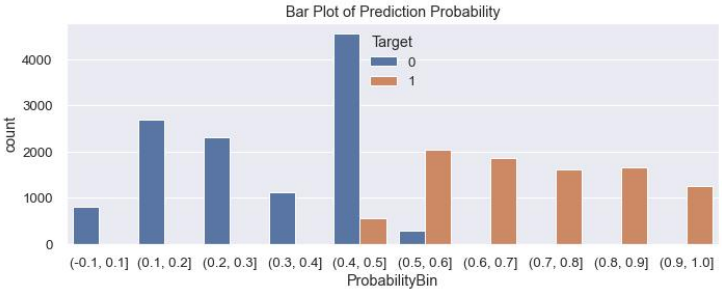
## ada boost-Binned Data

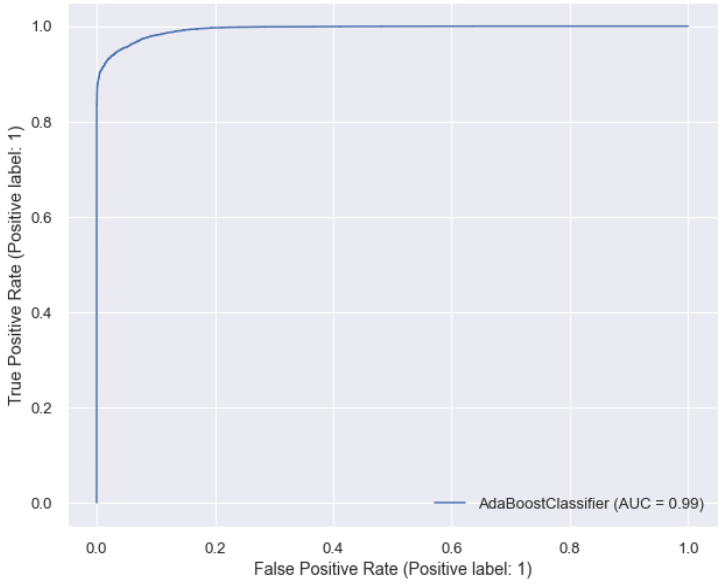


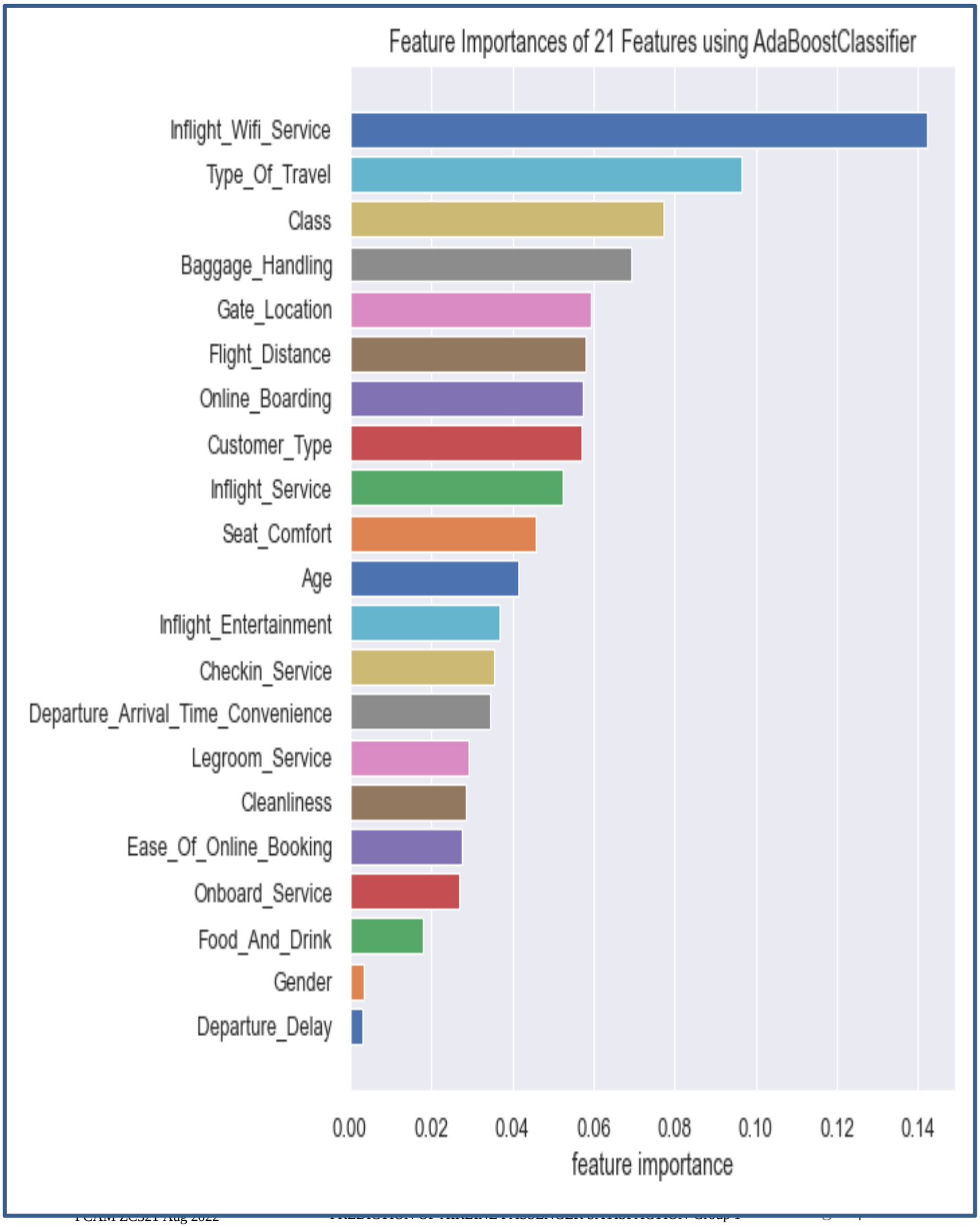



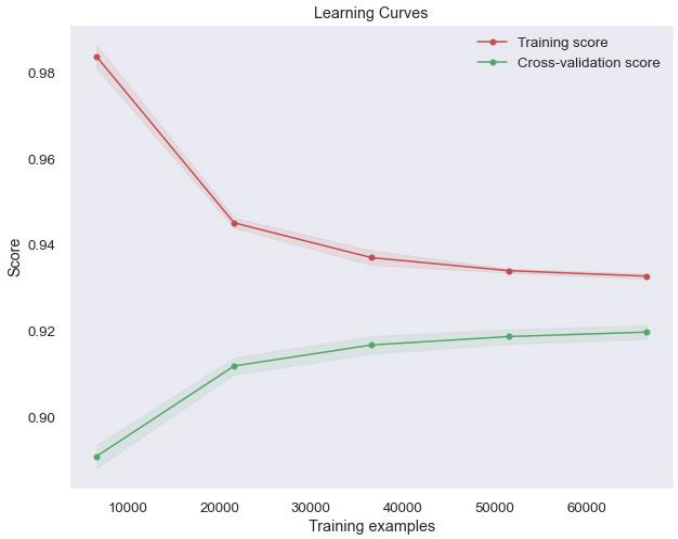



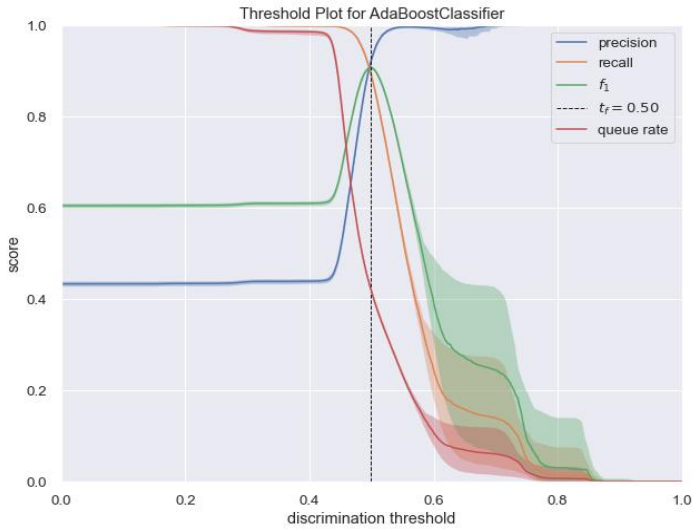

```
Misclassified Counts: 858
accuracy........ 0.9587
precision score........ 0.9599
recall score........ 0.9561
classification_report
              precision    recall  f1-score   support

           0       0.95      0.98      0.96     11776
           1       0.97      0.94      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

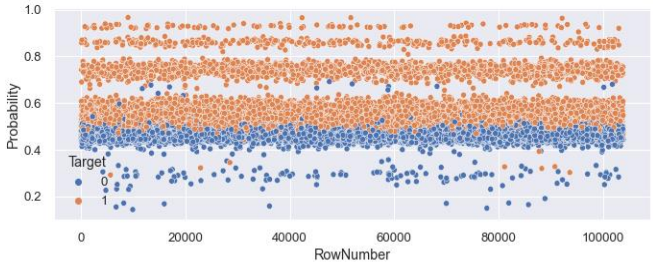
**ada boost-Binned Data Contd..**

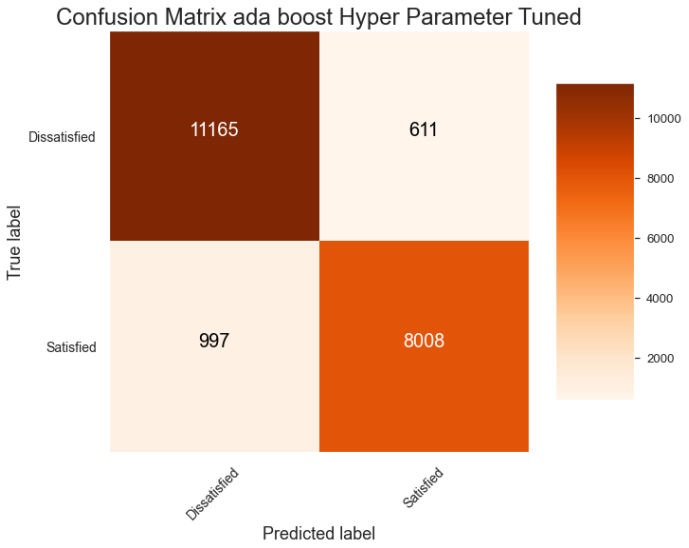


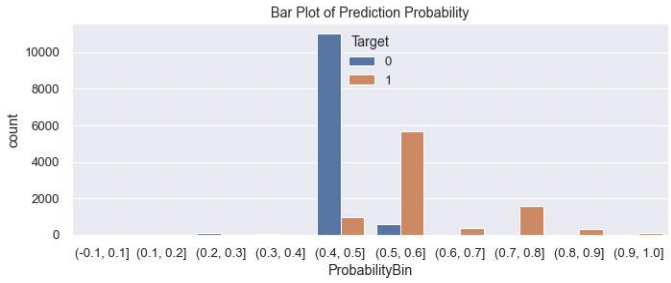
**ada boost-PCA Data**

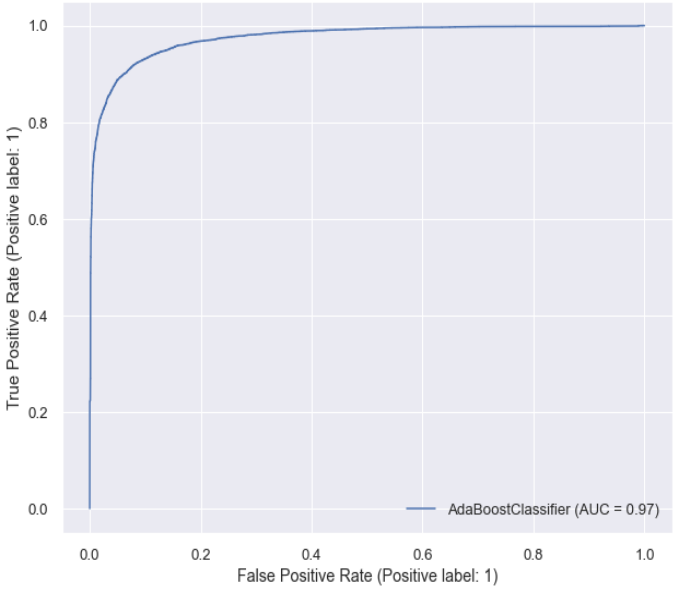


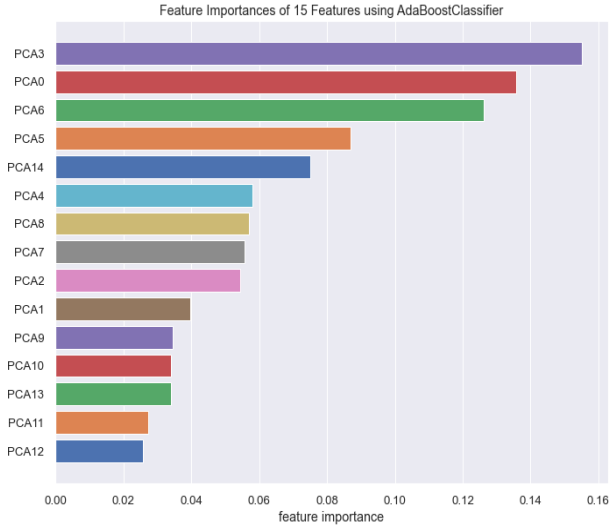



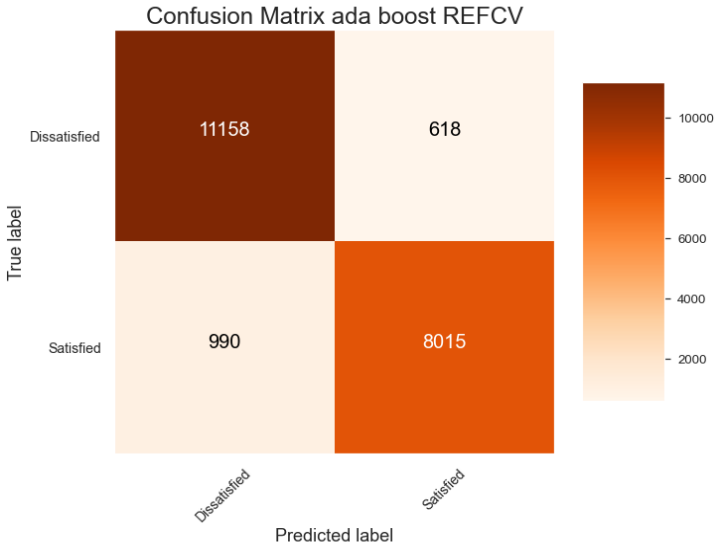



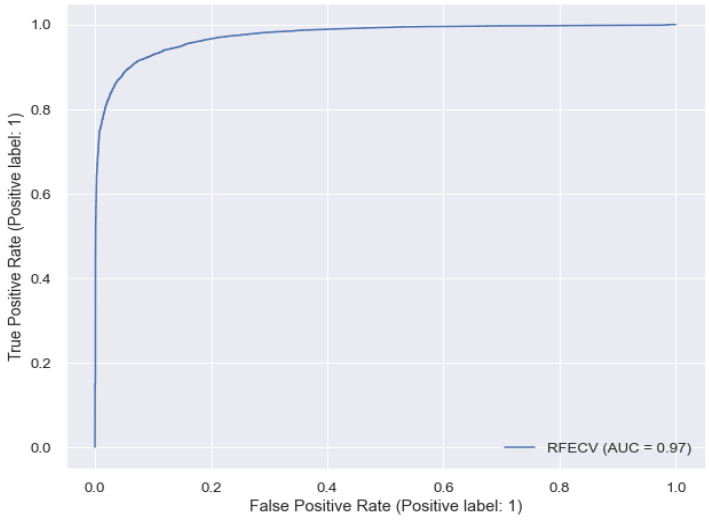




```
Misclassified Counts: 1608
accuracy..... 0.9226
precision score..... 0.9236
recall score..... 0.9187
classification_report
              precision    recall  f1-score   support

           0       0.92      0.95      0.93     11776
           1       0.93      0.89      0.91      9005

    accuracy                           0.92     20781
   macro avg       0.92      0.92      0.92     20781
weighted avg       0.92      0.92      0.92     20781
```

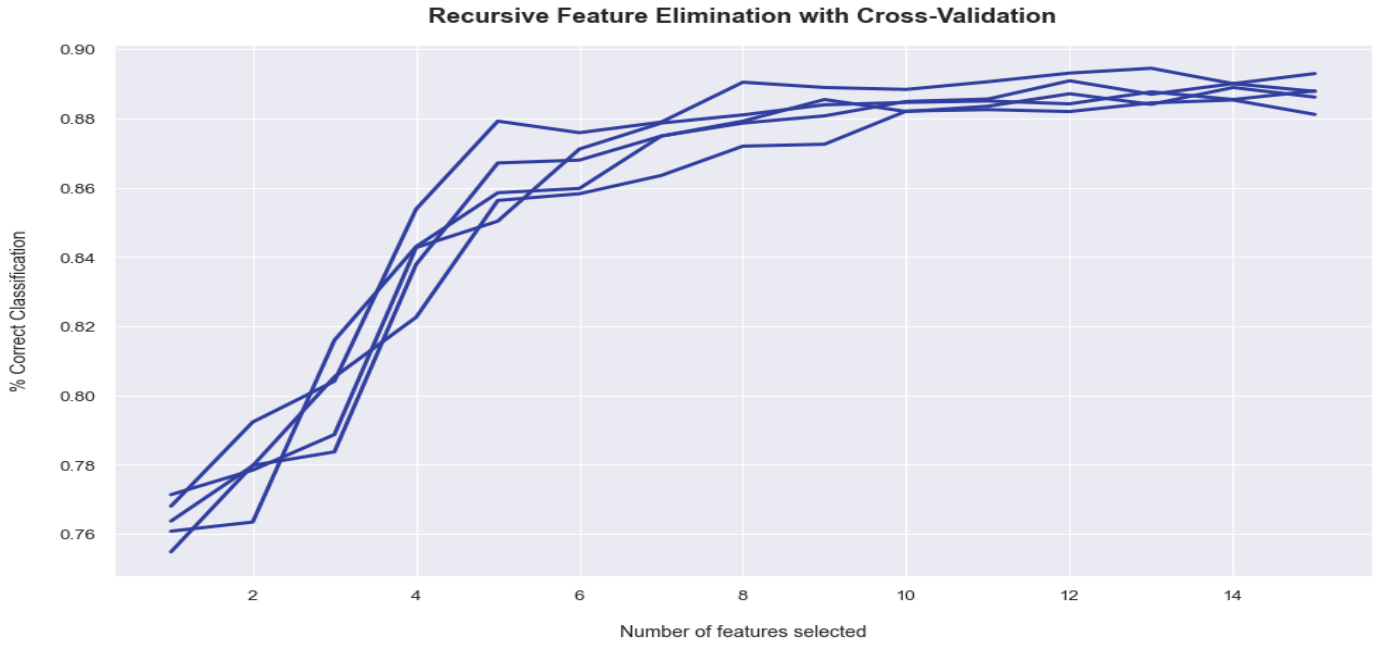
**ada boost-PCA Data Contd..**



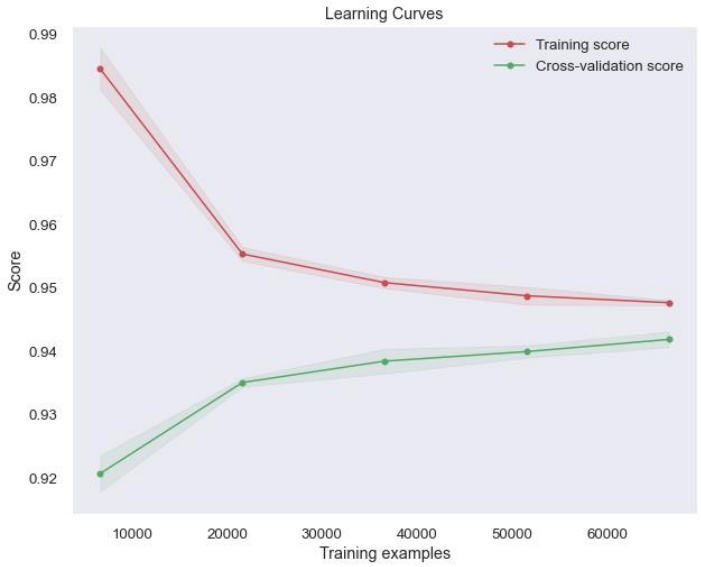



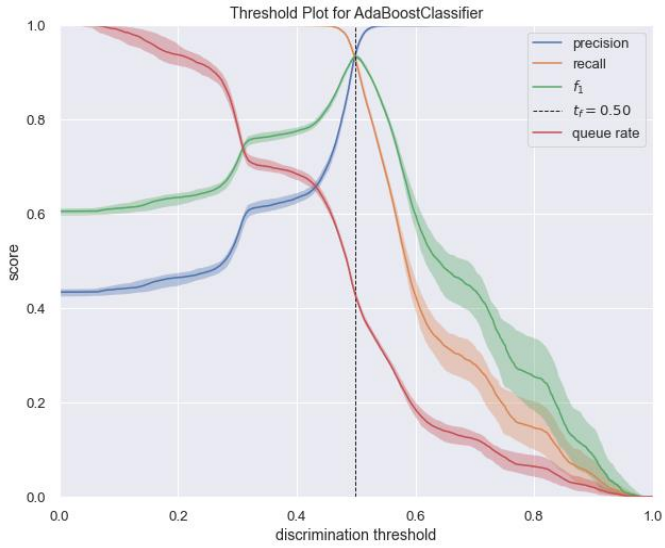



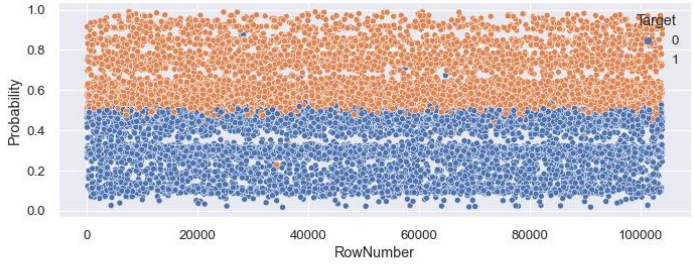

```
Misclassified Counts: 1608
accuracy........ 0.9226
precision score........ 0.9235
recall score........ 0.9188
classification_report
              precision    recall  f1-score   support

           0       0.92      0.95      0.93     11776
           1       0.93      0.89      0.91      9005

    accuracy                           0.92     20781
   macro avg       0.92      0.92      0.92     20781
weighted avg       0.92      0.92      0.92     20781
```

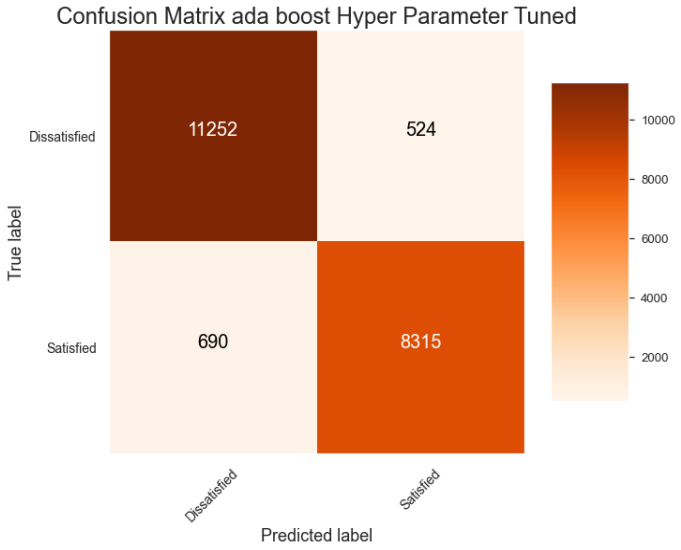


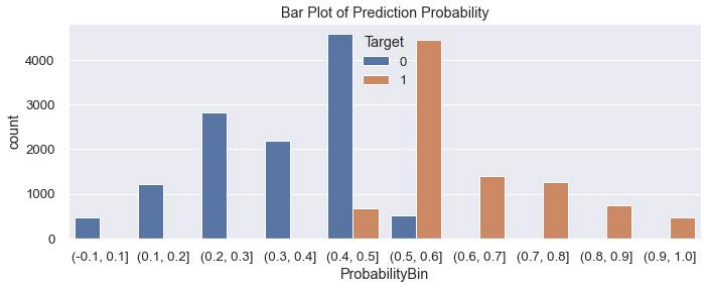
## ada Boost-VIF Data

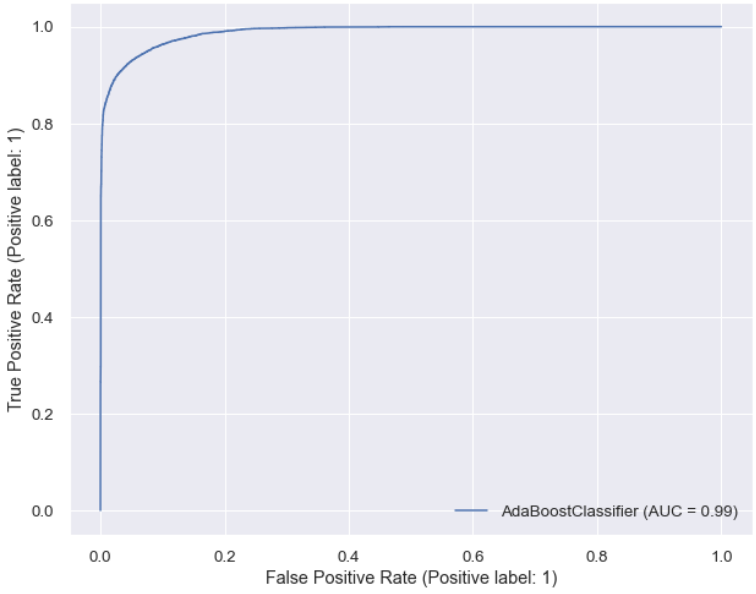


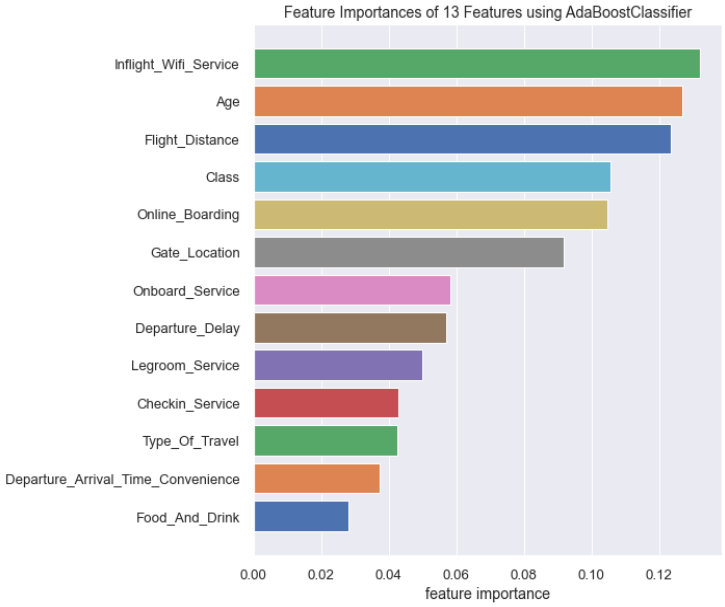



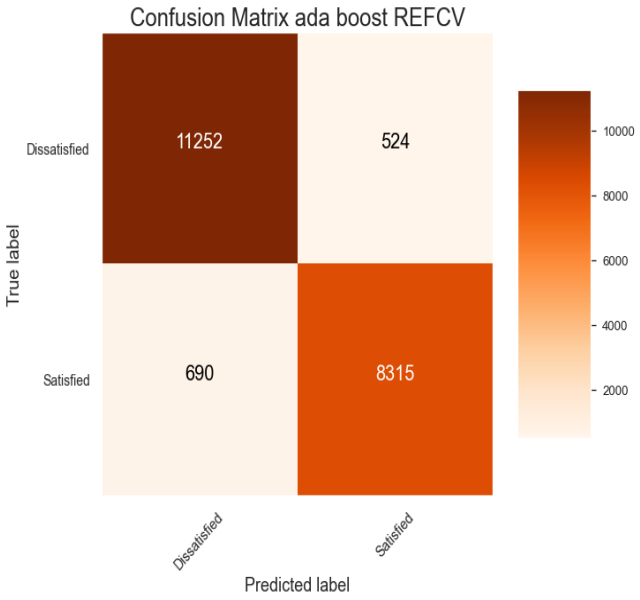



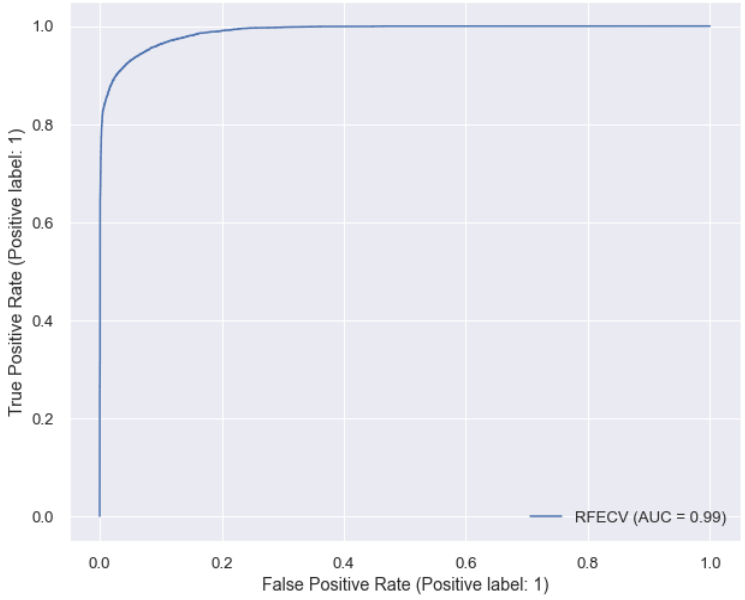




```
Misclassified Counts: 1214
accuracy..... 0.9416
precision score..... 0.9415
recall score..... 0.9394
classification_report
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     11776
           1       0.94      0.92      0.93      9005

    accuracy                           0.94     20781
   macro avg       0.94      0.94      0.94     20781
weighted avg       0.94      0.94      0.94     20781
```

## ada Boost-VIF Data Contd..

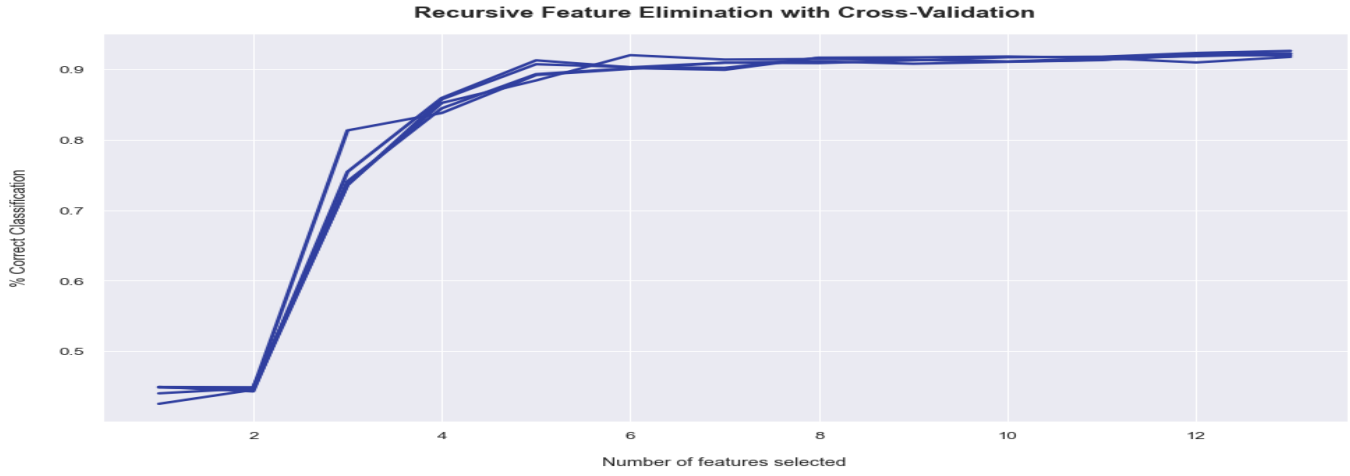


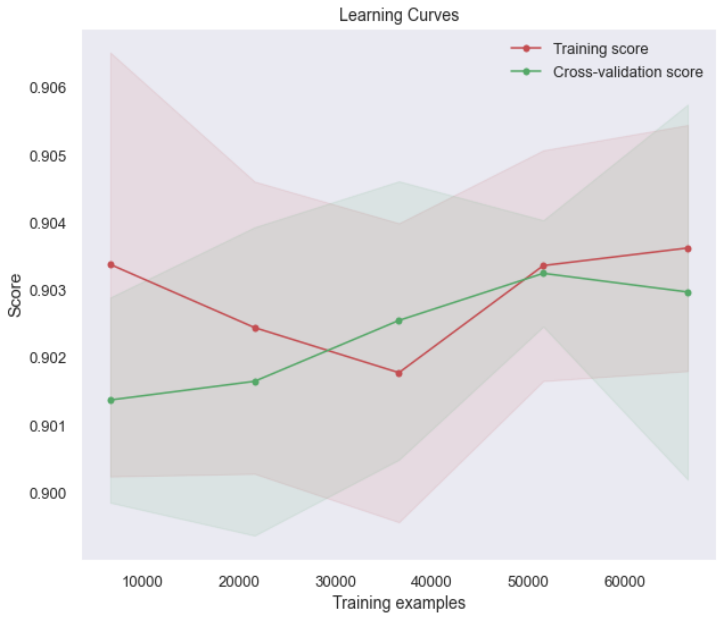



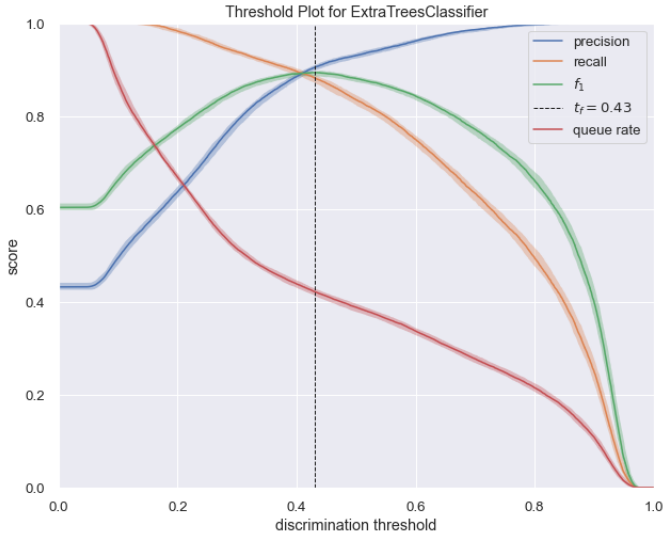

Misclassified Counts: 1214
accuracy..... 0.9416
precision score..... 0.9415
recall score..... 0.9394
classification_report

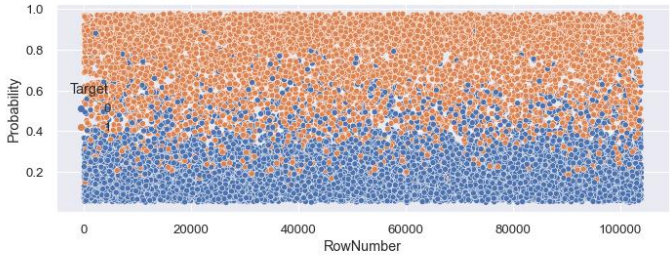
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.94 | 0.96 | 0.95 | 11776 |
| 1 | 0.94 | 0.92 | 0.93 | 9005 |
| accuracy | | | 0.94 | 20781 |
| macro avg | 0.94 | 0.94 | 0.94 | 20781 |
| weighted avg | 0.94 | 0.94 | 0.94 | 20781 |

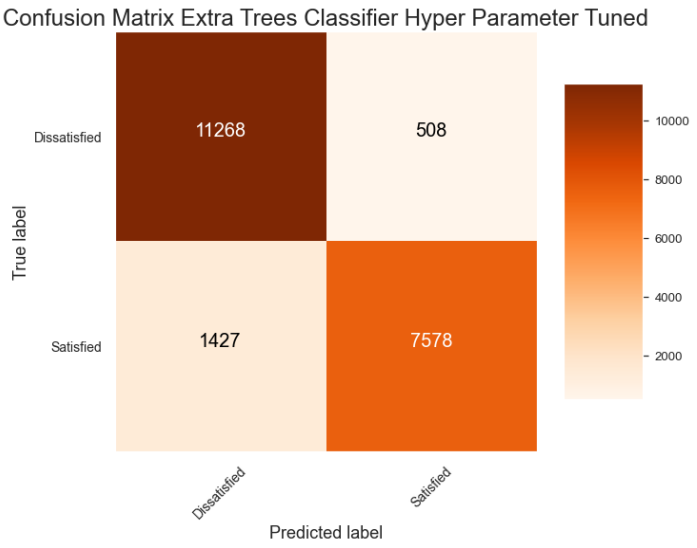


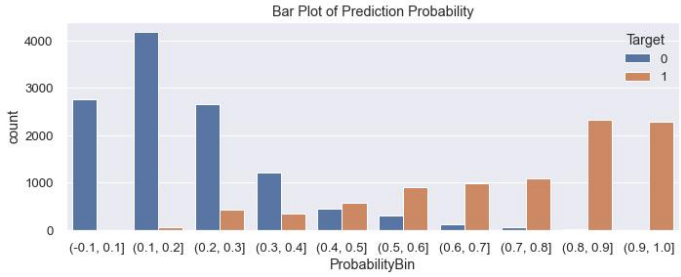
## Extra Trees Classifier - Un Binned Data

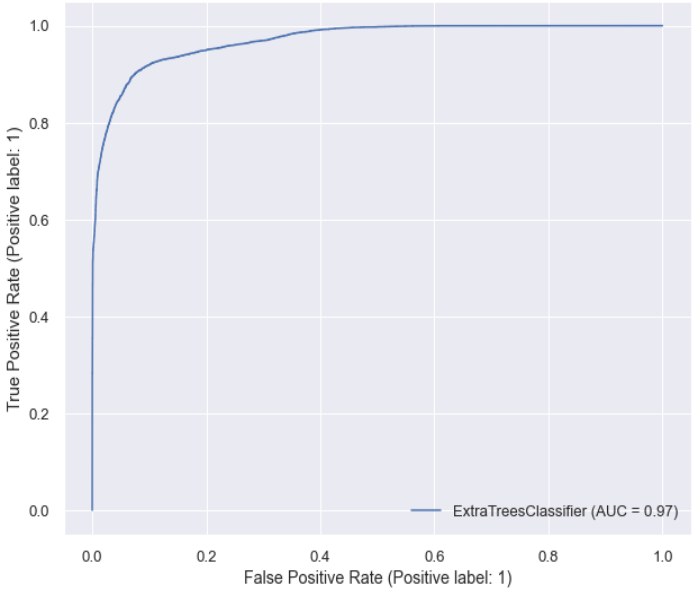


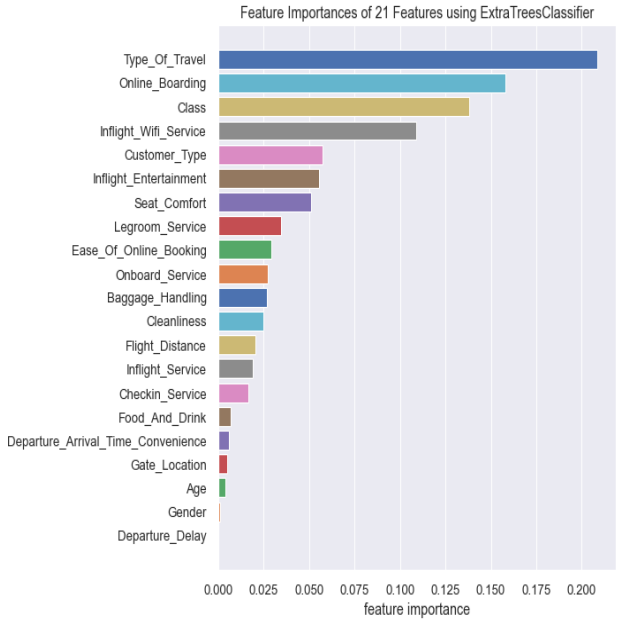



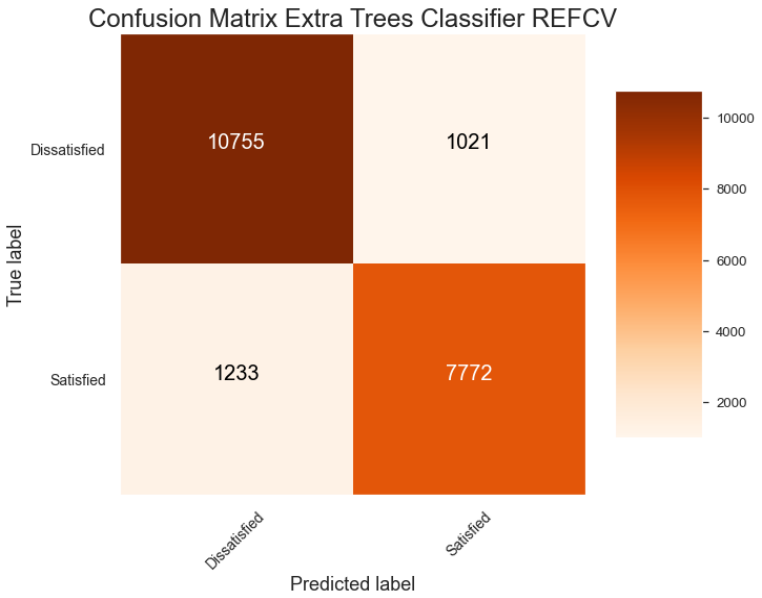



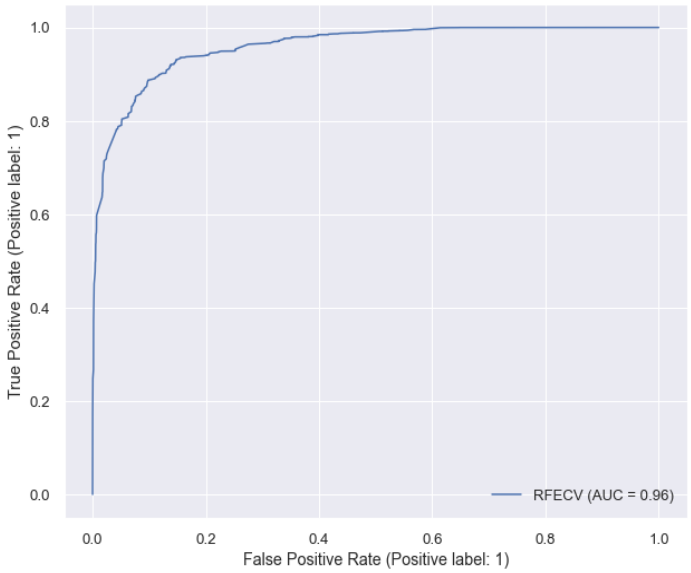




```
Misclassified Counts: 1935
accuracy........ 0.9069
precision score........ 0.9124
recall score........ 0.8992
classification_report
              precision    recall  f1-score   support

           0       0.89      0.96      0.92     11776
           1       0.94      0.84      0.89      9005

    accuracy                           0.91     20781
   macro avg       0.91      0.90      0.90     20781
weighted avg       0.91      0.91      0.91     20781
```

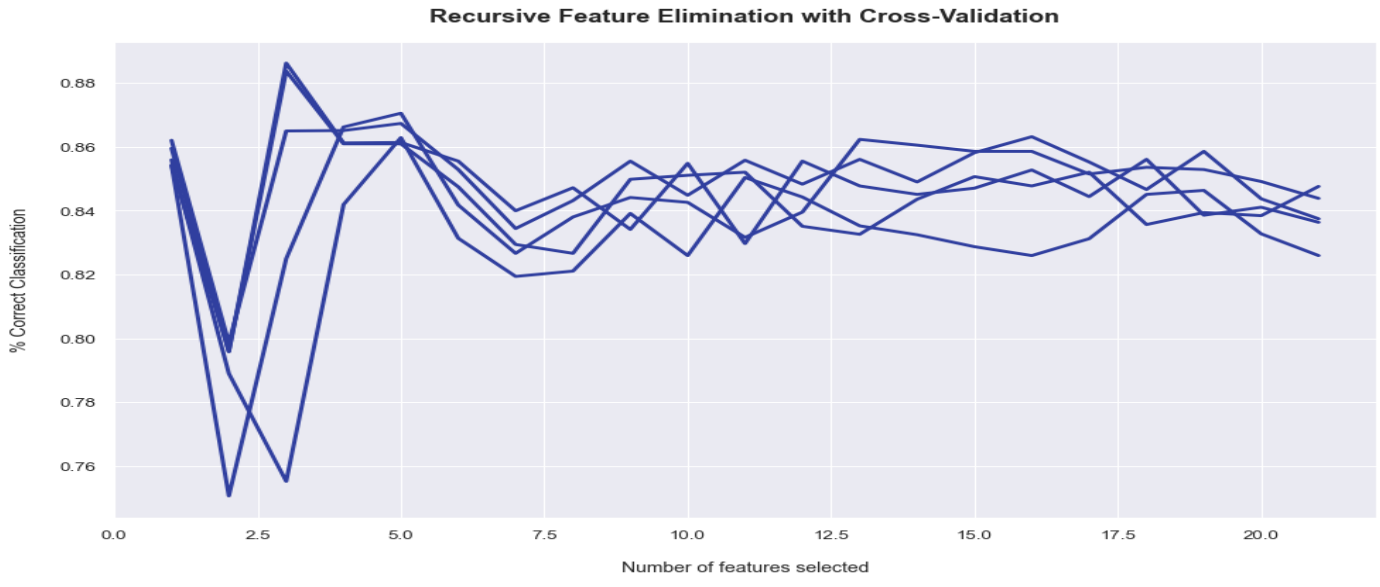
## Extra Trees Classifier - Un Binned Data Contd..



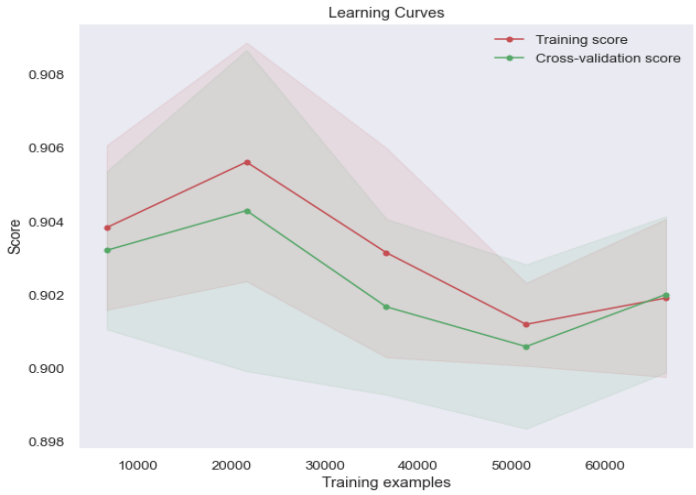



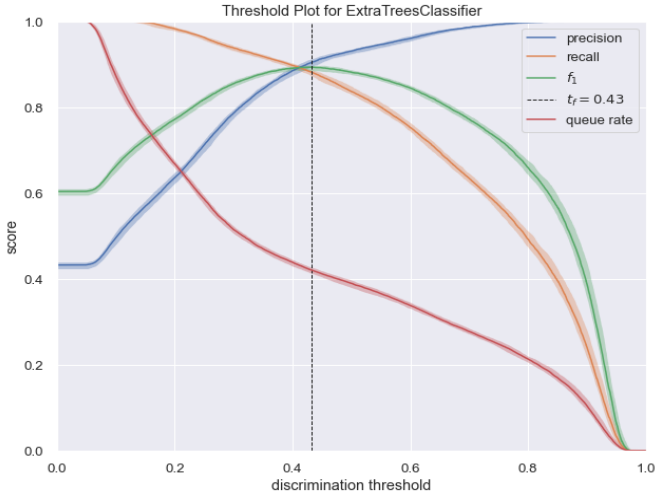



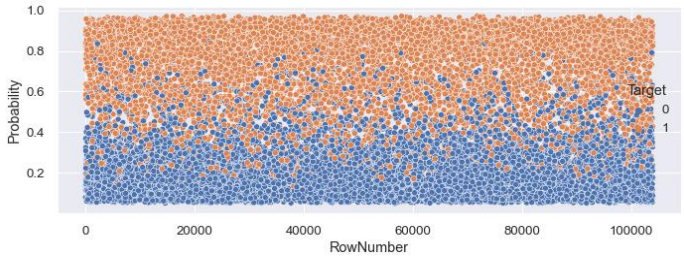

```
Misclassified Counts: 2254
accuracy........ 0.8915
precision score........ 0.8905
recall score........ 0.8882
classification_report
              precision    recall  f1-score   support

           0       0.90      0.91      0.91     11776
           1       0.88      0.86      0.87      9005

    accuracy                           0.89     20781
   macro avg       0.89      0.89      0.89     20781
weighted avg       0.89      0.89      0.89     20781
```

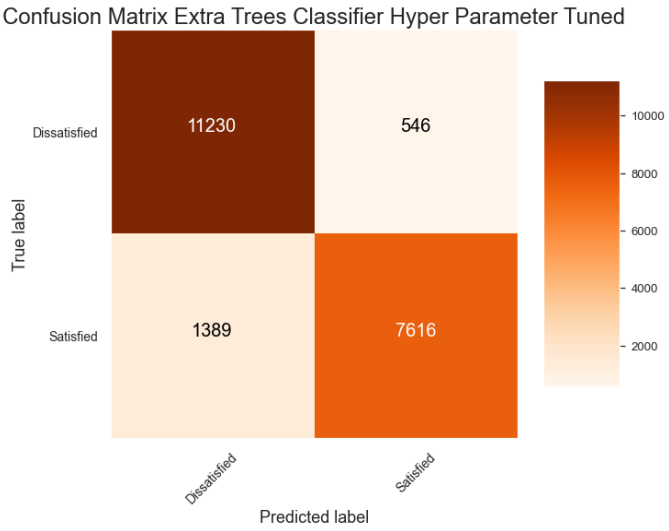


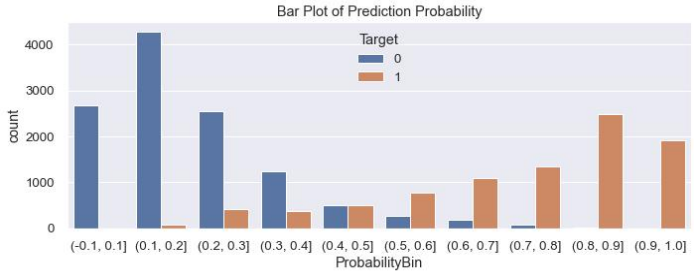
## Extra Trees Classifier - Binned Data

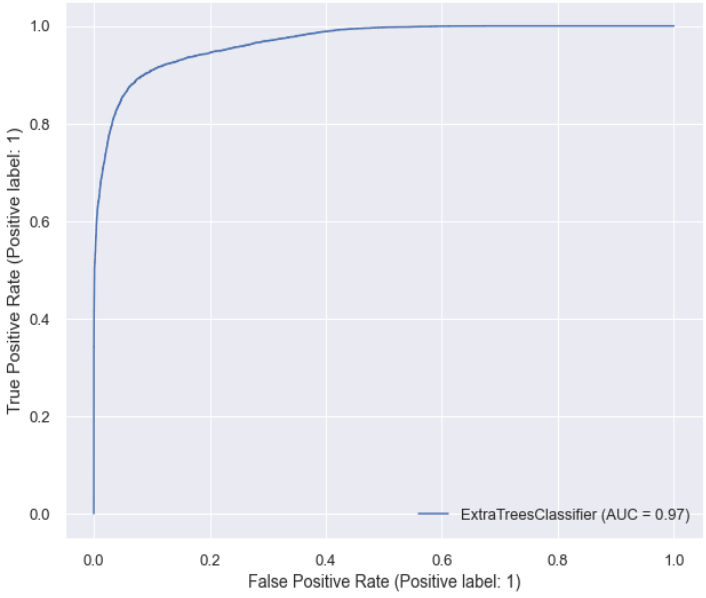


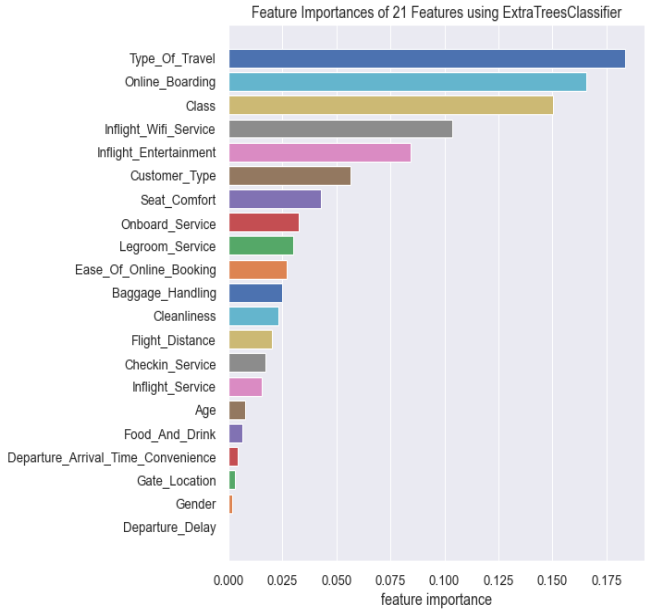



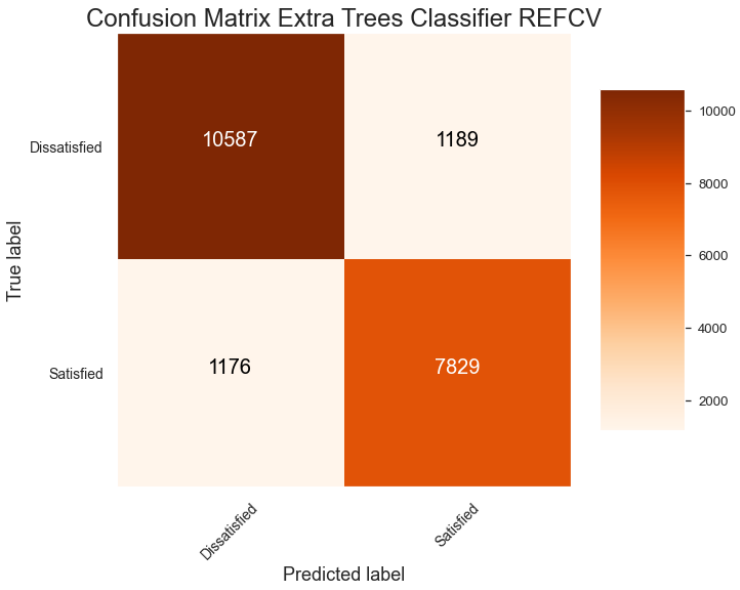



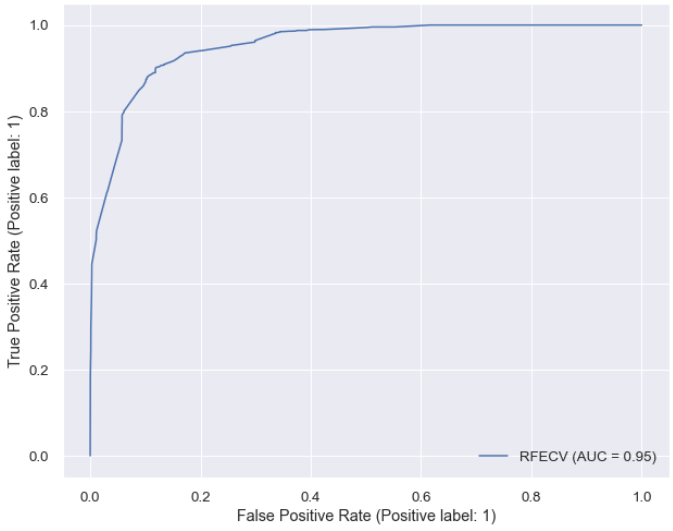




```
Misclassified Counts: 1935
accuracy........ 0.9069
precision score........ 0.9115
recall score........ 0.8997
classification_report
              precision    recall  f1-score   support

           0       0.89      0.95      0.92     11776
           1       0.93      0.85      0.89      9005

    accuracy                           0.91     20781
   macro avg       0.91      0.90      0.90     20781
weighted avg       0.91      0.91      0.91     20781
```

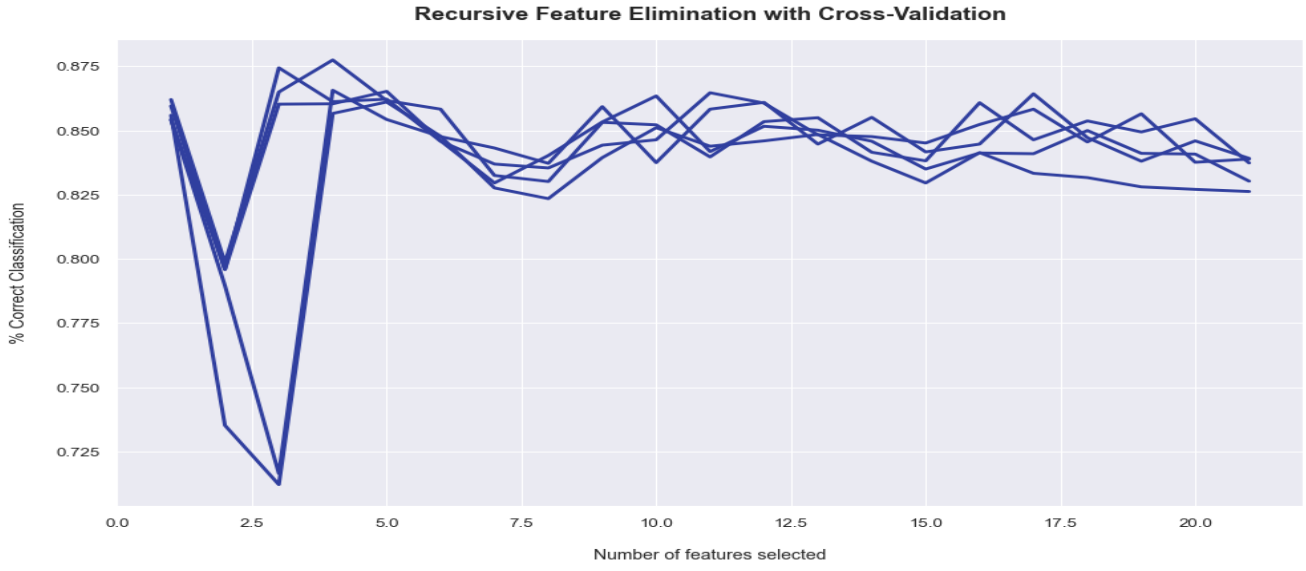
## Extra Trees Classifier - Binned Data Contd..

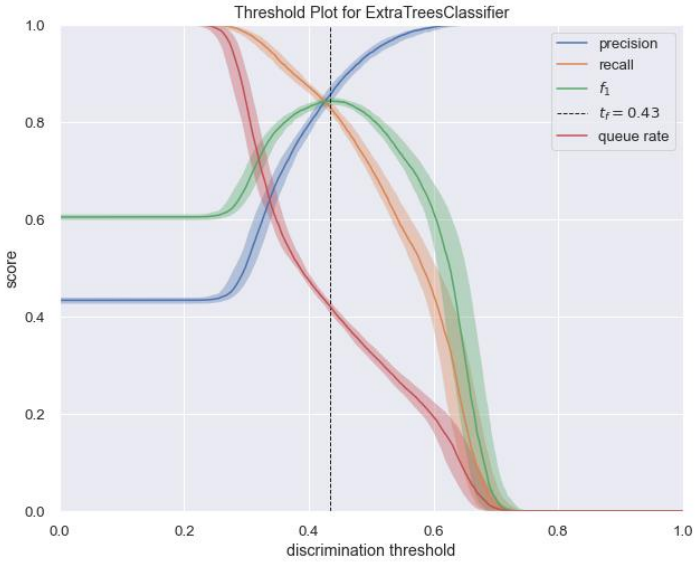


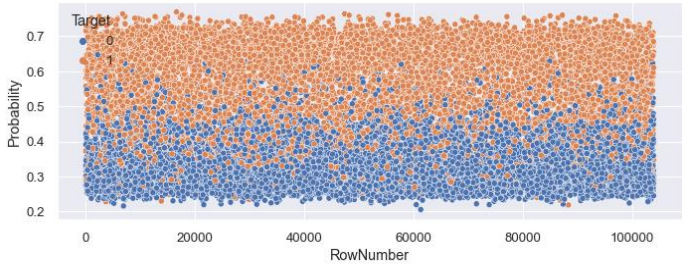



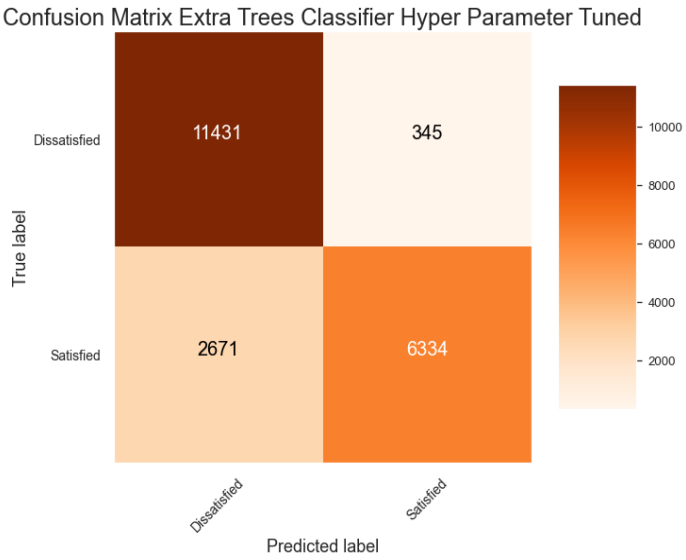



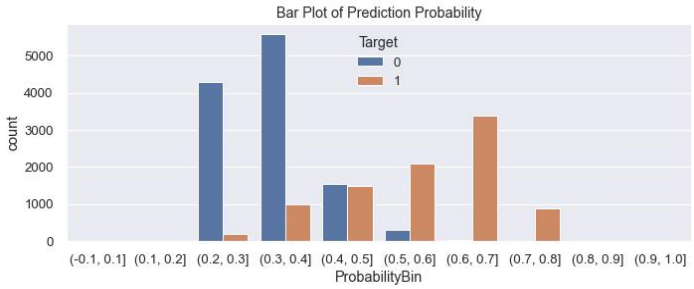

```
Misclassified Counts: 2365
accuracy........ 0.8862
precision score........ 0.8841
recall score........ 0.8842
classification_report
              precision    recall  f1-score   support

           0       0.90      0.90      0.90     11776
           1       0.87      0.87      0.87      9005

    accuracy                           0.89     20781
   macro avg       0.88      0.88      0.88     20781
weighted avg       0.89      0.89      0.89     20781
```

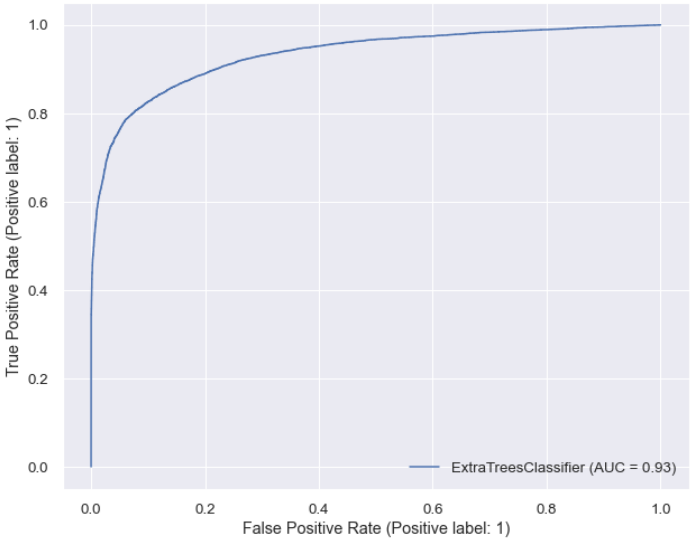


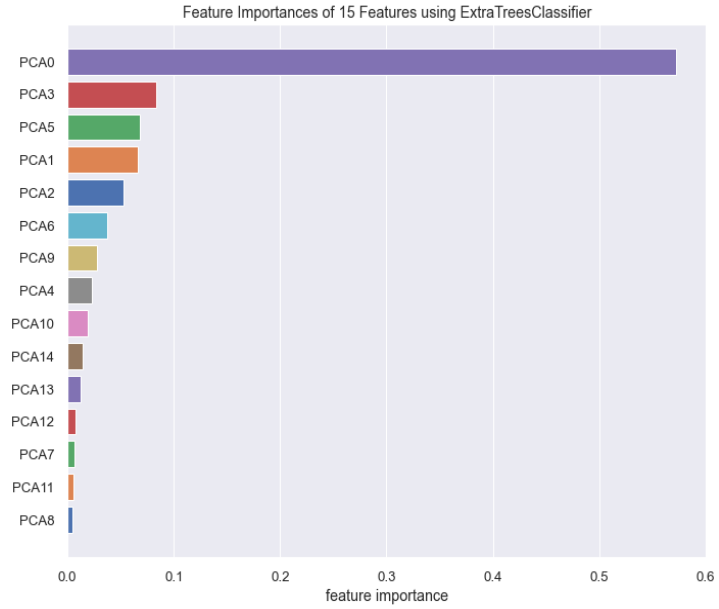



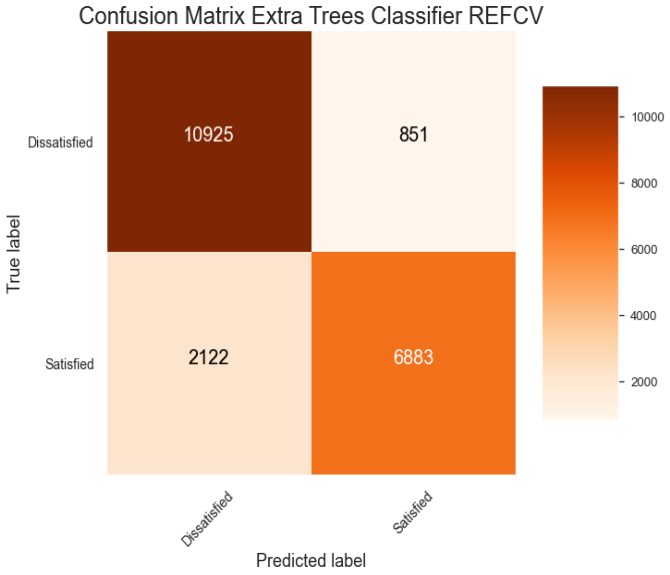



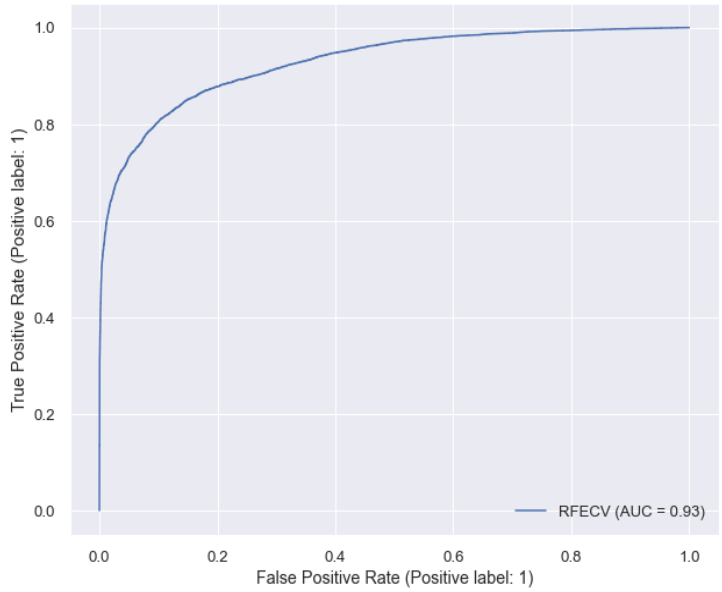



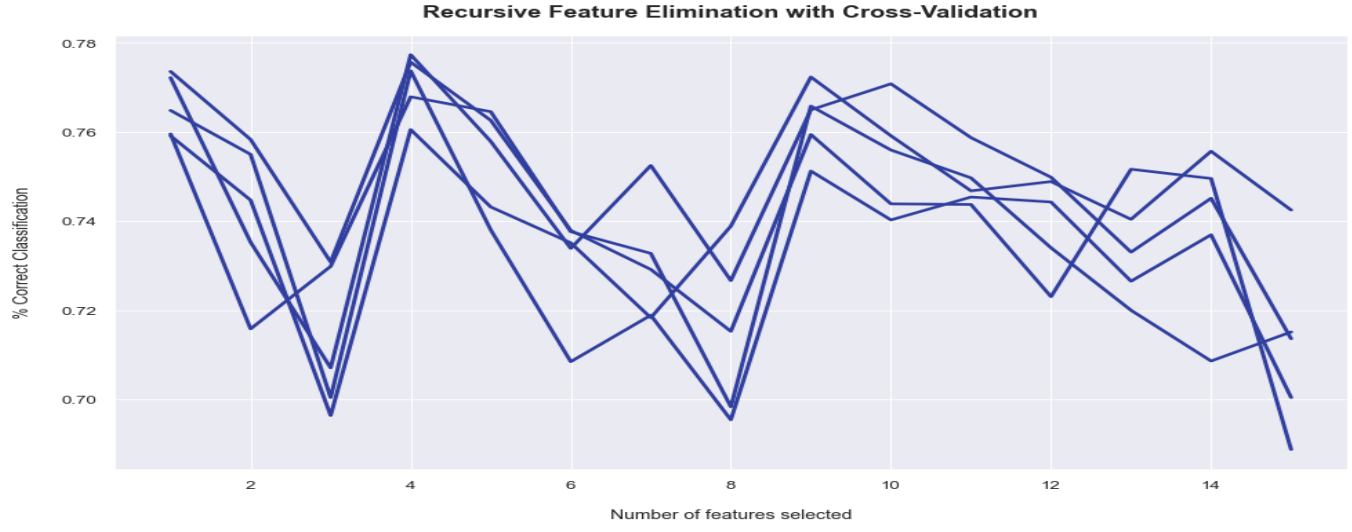

```
Misclassified Counts: 3016
accuracy..... 0.8549
precision score..... 0.8795
recall score..... 0.8370
classification_report
              precision    recall  f1-score   support

           0       0.81      0.97      0.88     11776
           1       0.95      0.70      0.81      9005

    accuracy                           0.85     20781
   macro avg       0.88      0.84      0.85     20781
weighted avg       0.87      0.85      0.85     20781
```



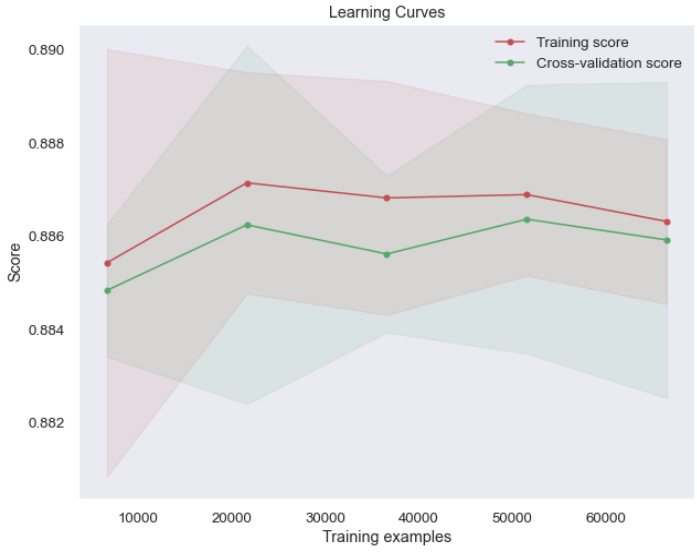



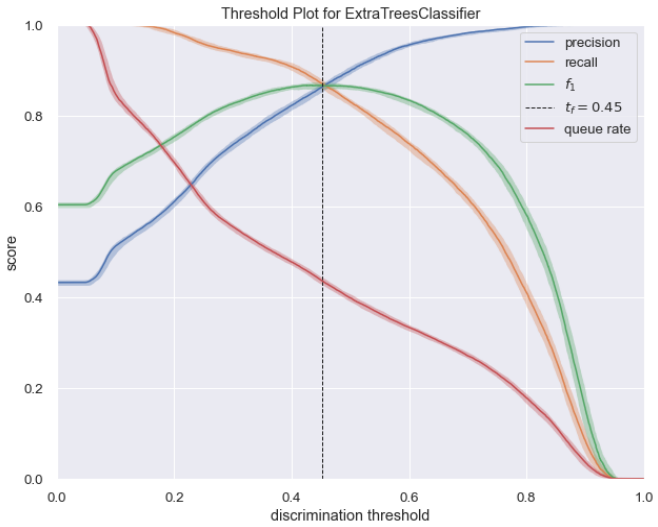



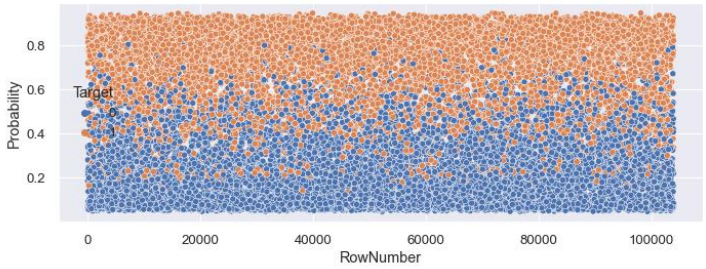

```
Misclassified Counts: 2973
accuracy........ 0.8569
precision score........ 0.8637
recall score........ 0.8460
classification_report
              precision    recall  f1-score   support

           0       0.84      0.93      0.88     11776
           1       0.89      0.76      0.82      9005

    accuracy                           0.86     20781
   macro avg       0.86      0.85      0.85     20781
weighted avg       0.86      0.86      0.86     20781
```

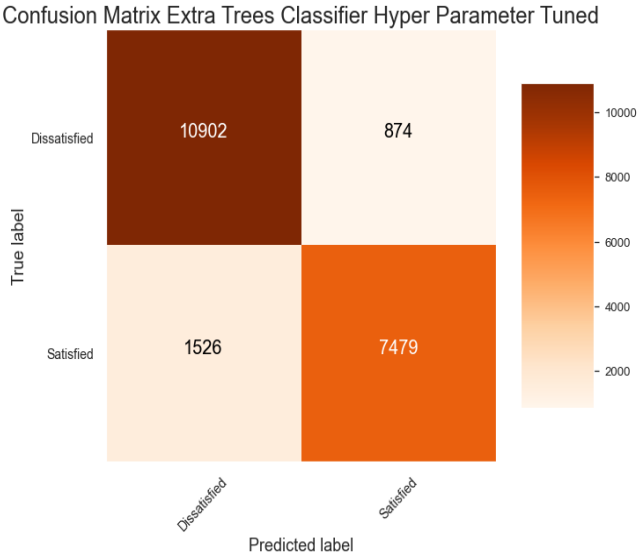


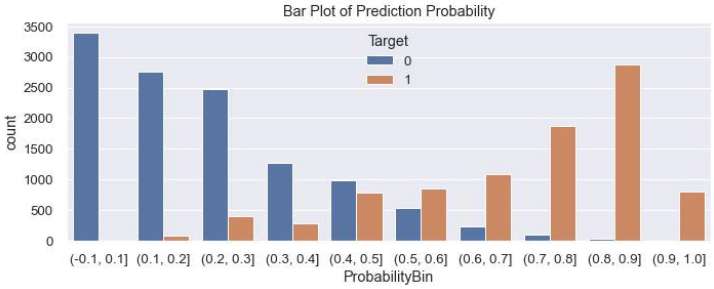
## Extra Trees Classifier - VIF Data

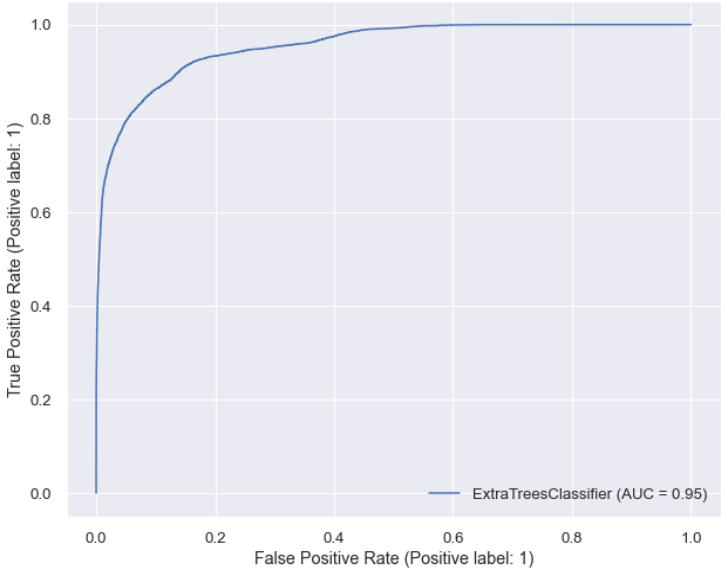


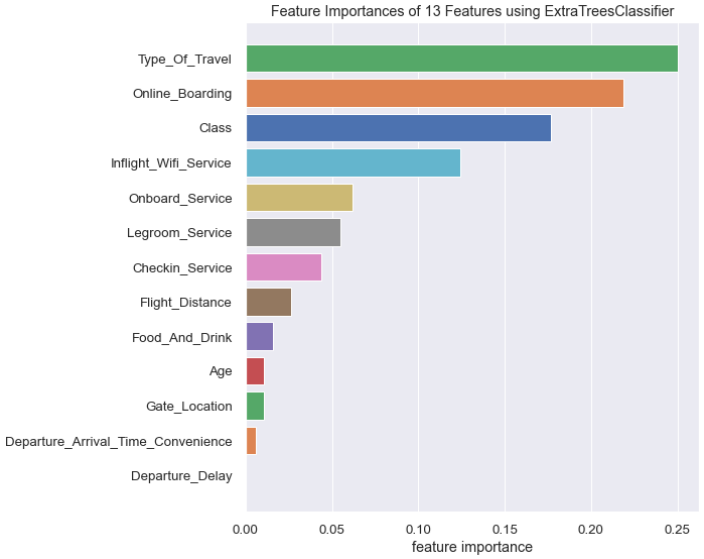



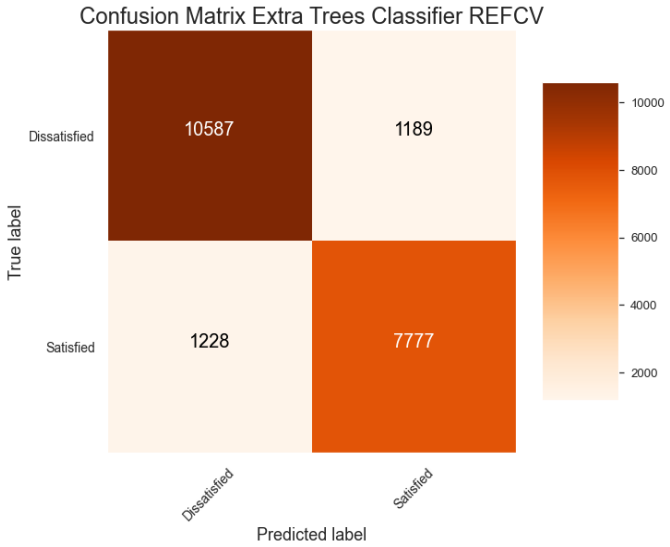



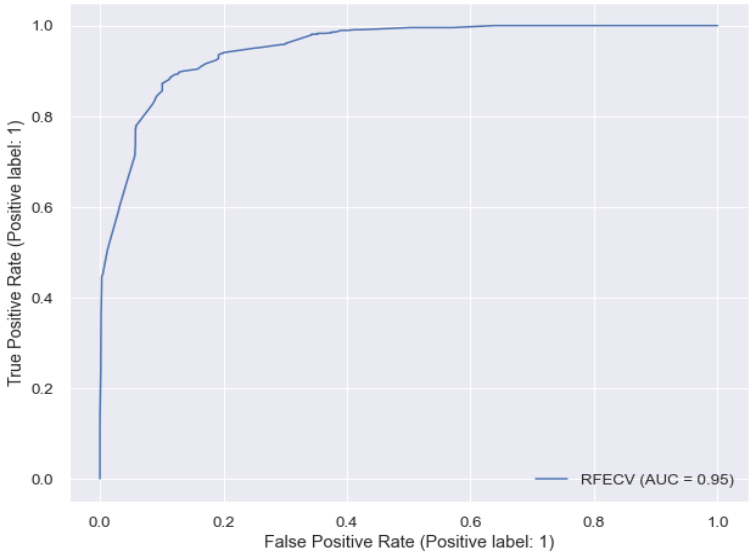




```
Misclassified Counts: 2400
accuracy........ 0.8845
precision score........ 0.8863
recall score........ 0.8782
classification_report
              precision    recall  f1-score   support

           0       0.88      0.93      0.90     11776
           1       0.90      0.83      0.86      9005

    accuracy                           0.88     20781
   macro avg       0.89      0.88      0.88     20781
weighted avg       0.89      0.88      0.88     20781
```

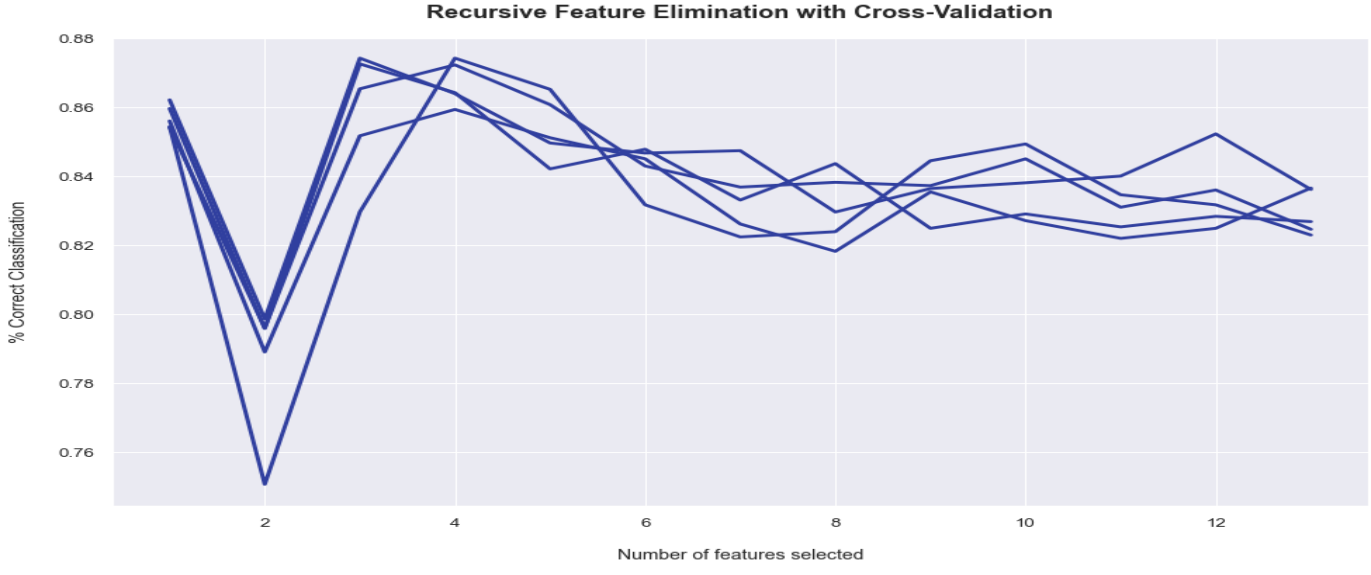
## Extra Trees Classifier - VIF Data Contd..



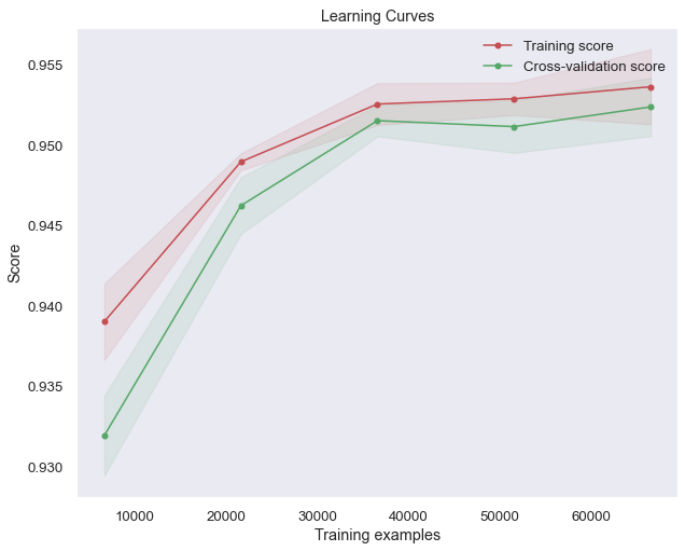



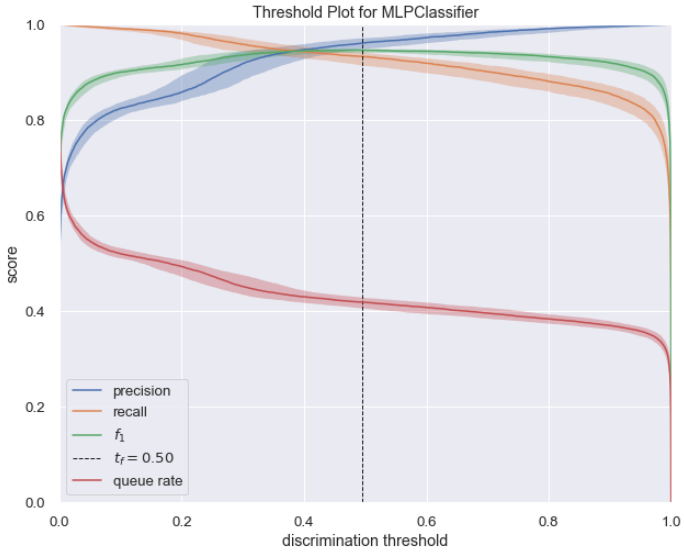



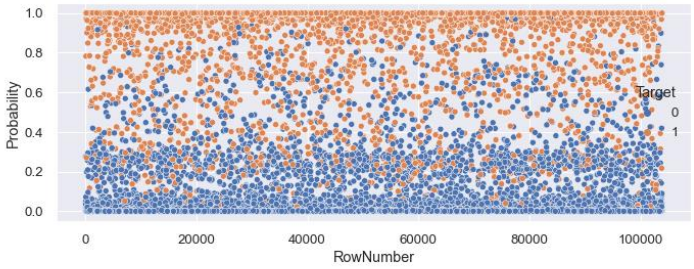

```
Misclassified Counts: 2417
accuracy........ 0.8837
precision score........ 0.8817
recall score........ 0.8813
classification_report
              precision    recall  f1-score   support

           0       0.90      0.90      0.90     11776
           1       0.87      0.86      0.87      9005

    accuracy                           0.88     20781
   macro avg       0.88      0.88      0.88     20781
weighted avg       0.88      0.88      0.88     20781
```

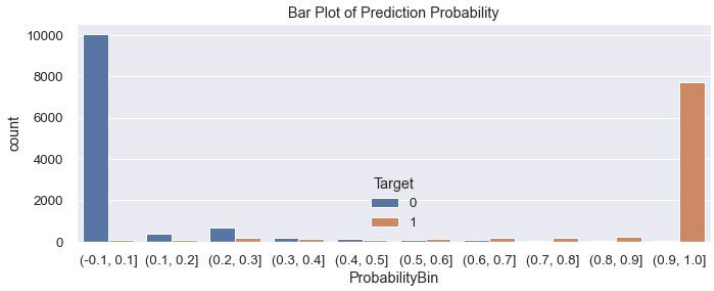


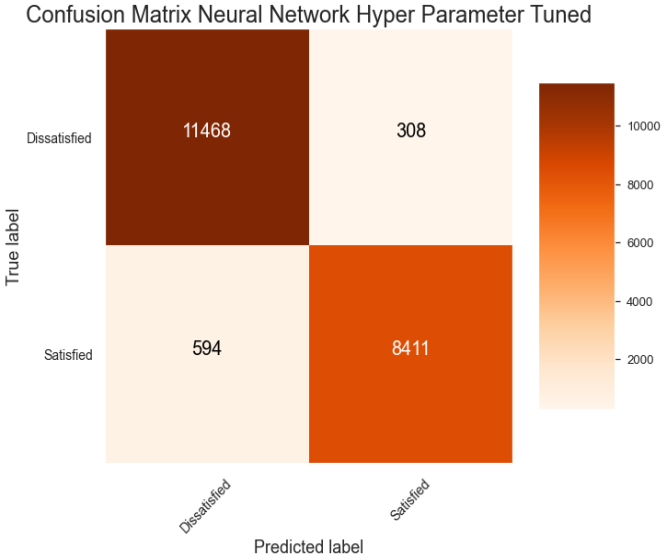
## Neural Network Binned Data

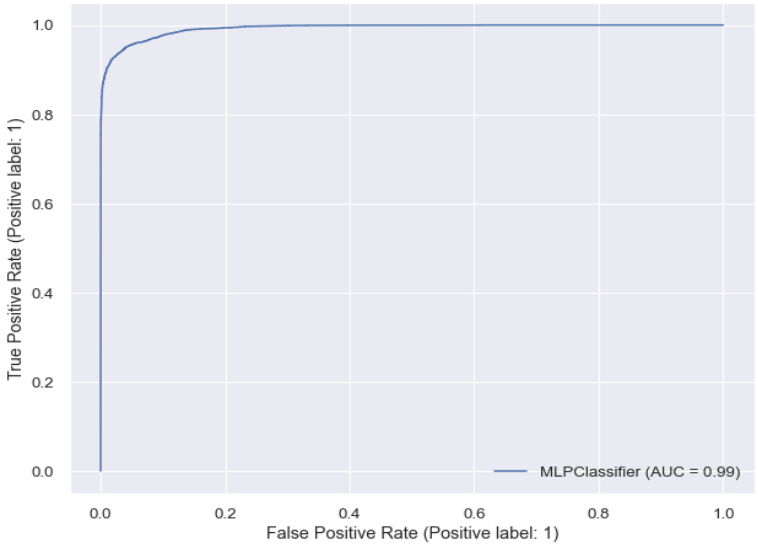


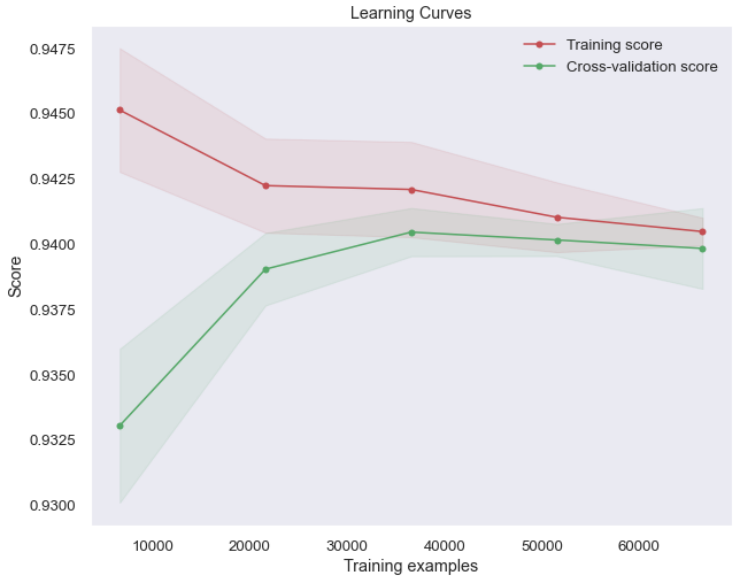



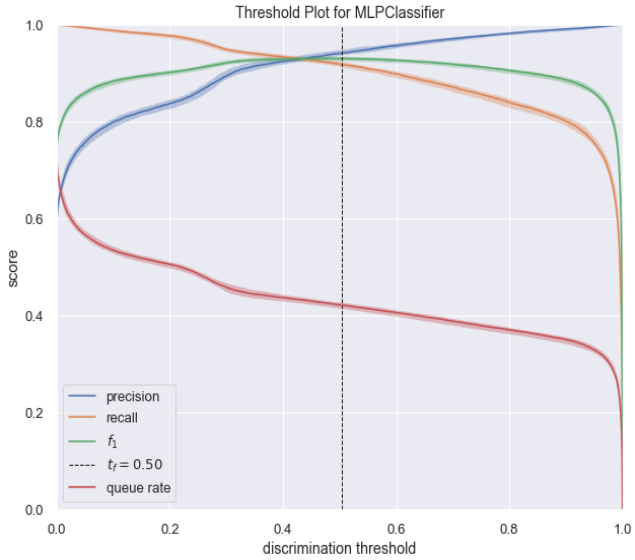



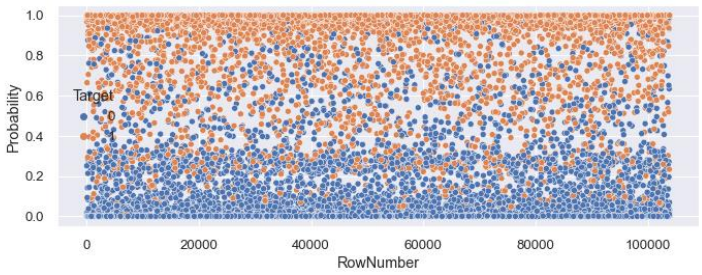



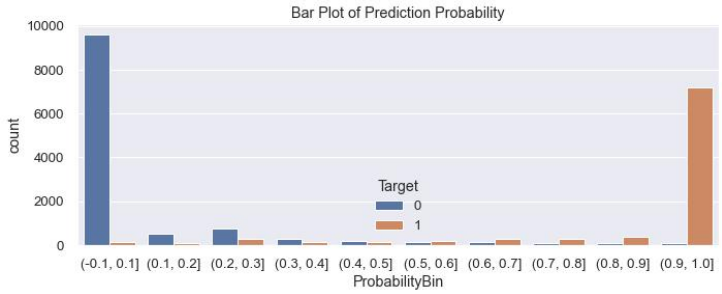

```
Misclassified Counts: 902
accuracy........ 0.9566
precision score........ 0.9577
recall score........ 0.9539
classification_report
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     11776
           1       0.96      0.93      0.95      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.95      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

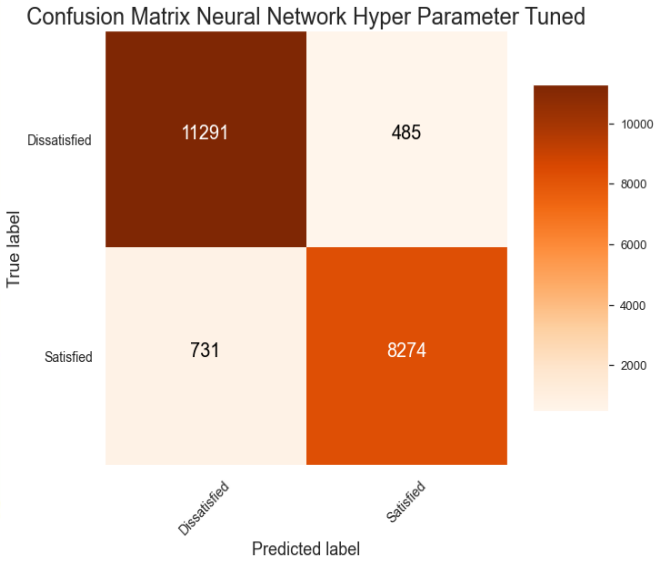
## Neural Network VIF Data

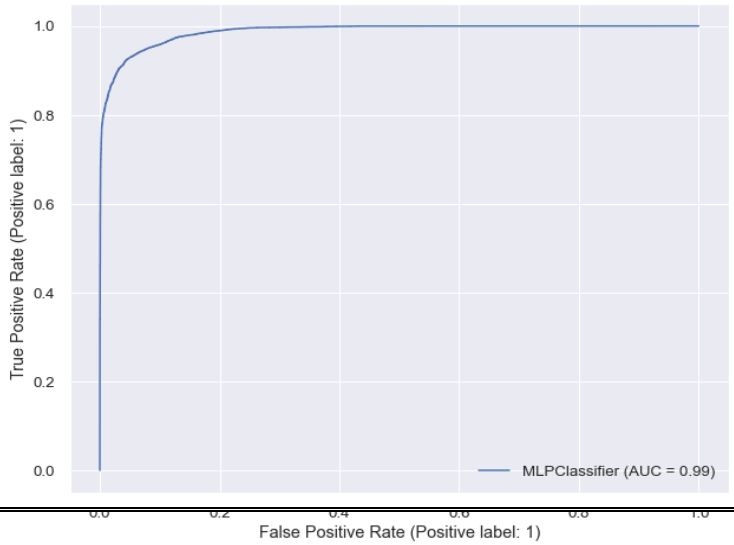


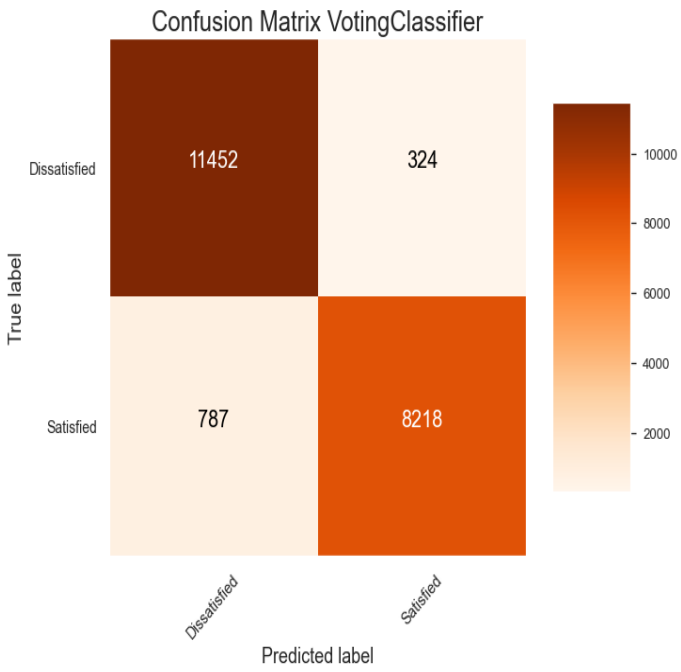



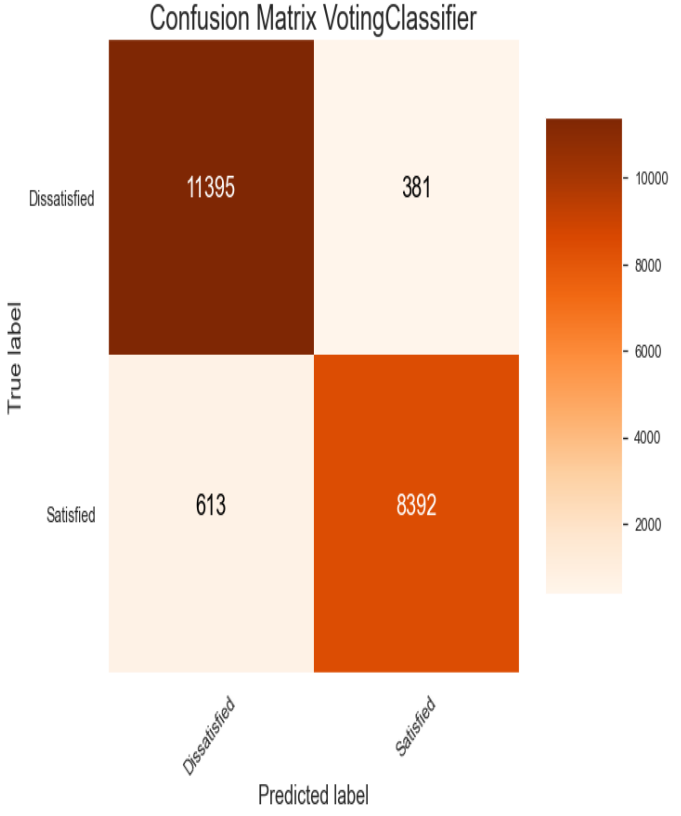

```
Misclassified Counts: 1216
accuracy........ 0.9415
precision score........ 0.9419
recall score........ 0.9388
classification_report
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     11776
           1       0.94      0.92      0.93      9005

    accuracy                           0.94     20781
   macro avg       0.94      0.94      0.94     20781
weighted avg       0.94      0.94      0.94     20781
```

**Un Binned Data**



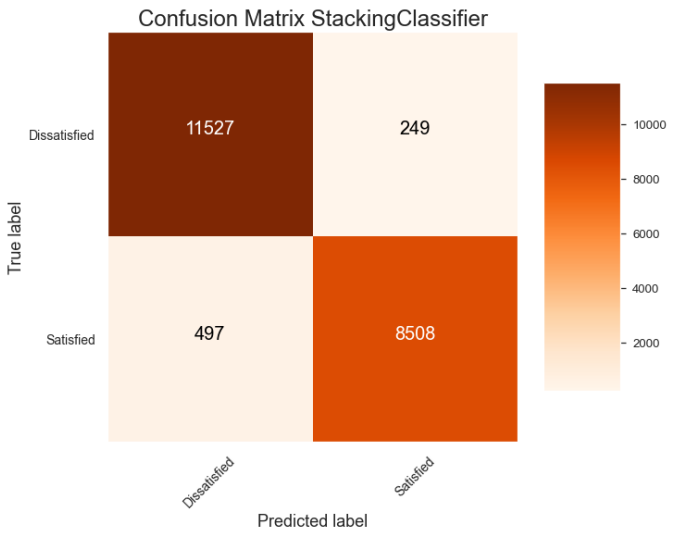

```
Misclassified Counts: 1111
accuracy........ 0.9465
precision score........ 0.9489
recall score........ 0.9425
classification_report
              precision    recall  f1-score   support

           0       0.94      0.97      0.95     11776
           1       0.96      0.91      0.94      9005

    accuracy                           0.95     20781
   macro avg       0.95      0.94      0.95     20781
weighted avg       0.95      0.95      0.95     20781
```

**Binned Data**



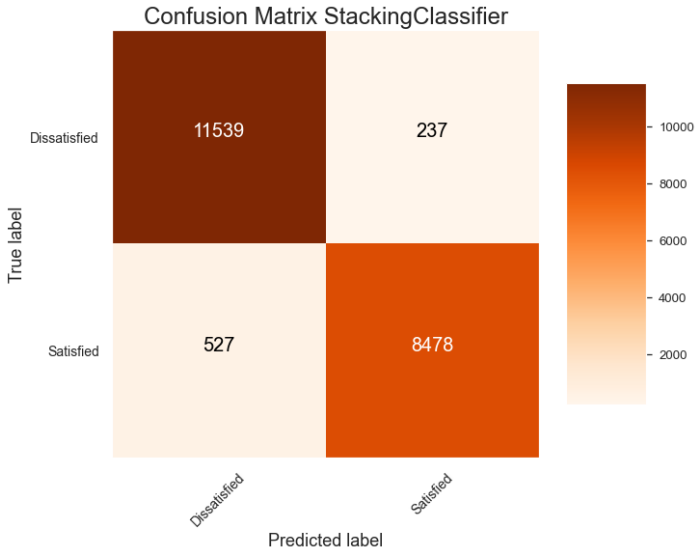

```
Misclassified Counts: 994
accuracy........ 0.9522
precision score........ 0.9528
recall score........ 0.9498
classification_report
              precision    recall  f1-score   support

           0       0.95      0.97      0.96     11776
           1       0.96      0.93      0.94      9005

    accuracy                           0.95     20781
   macro avg       0.95      0.95      0.95     20781
weighted avg       0.95      0.95      0.95     20781
```

## Un Binned Data



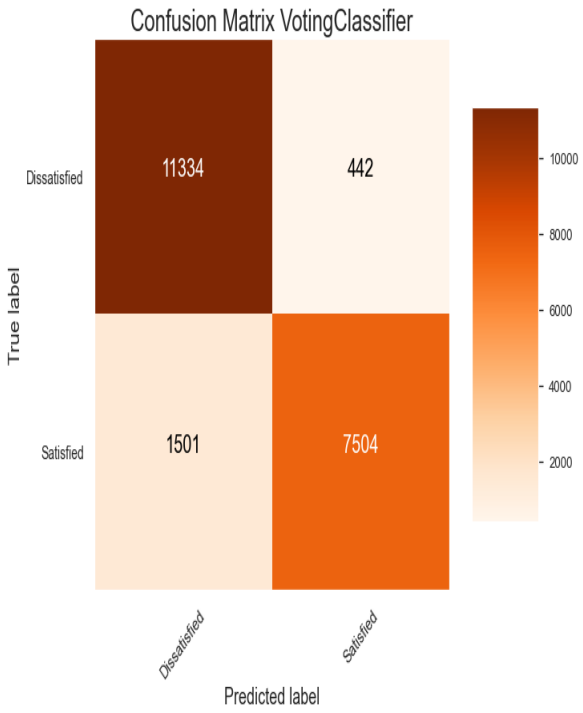

```
Misclassified Counts: 746
accuracy........ 0.9641
precision score........ 0.9651
recall score........ 0.9618
classification_report
              precision    recall  f1-score   support

           0       0.96      0.98      0.97     11776
           1       0.97      0.94      0.96      9005

    accuracy                           0.96     20781
   macro avg       0.97      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

## Binned Data

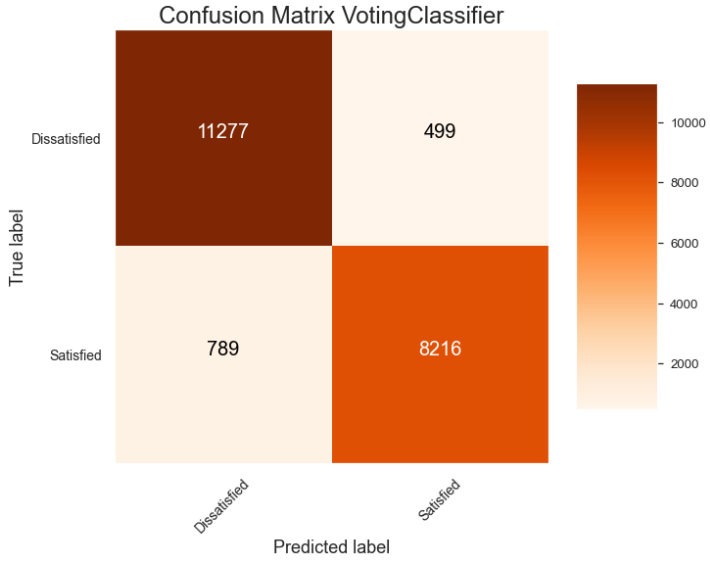



```
Misclassified Counts: 764
accuracy........ 0.9632
precision score........ 0.9646
recall score........ 0.9607
classification_report
              precision    recall  f1-score   support

           0       0.96      0.98      0.97     11776
           1       0.97      0.94      0.96      9005

    accuracy                           0.96     20781
   macro avg       0.96      0.96      0.96     20781
weighted avg       0.96      0.96      0.96     20781
```

## PCA Data

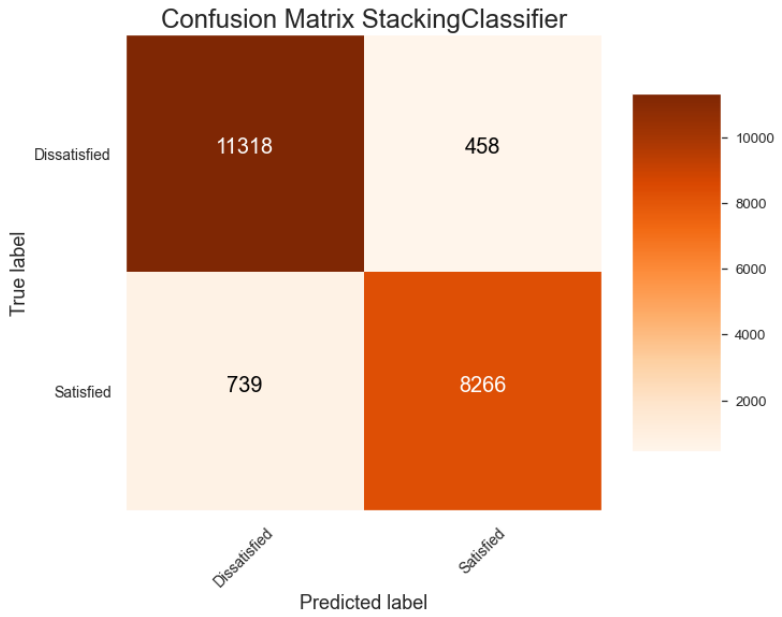



## VIF Data




```
Misclassified Counts: 1288
accuracy........ 0.9380
precision score........ 0.9387
recall score........ 0.9350
classification_report
              precision    recall  f1-score   support

           0       0.93      0.96      0.95     11776
           1       0.94      0.91      0.93      9005

    accuracy                           0.94     20781
   macro avg       0.94      0.94      0.94     20781
weighted avg       0.94      0.94      0.94     20781
```

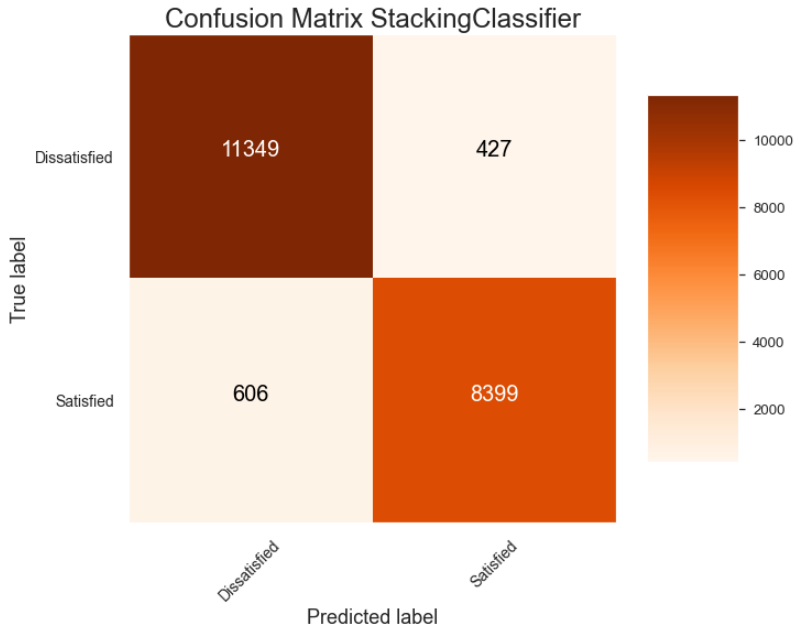
## PCA Data




Misclassified Counts: 1197
accuracy..... 0.9424
precision score..... 0.9431
recall score..... 0.9395
classification_report

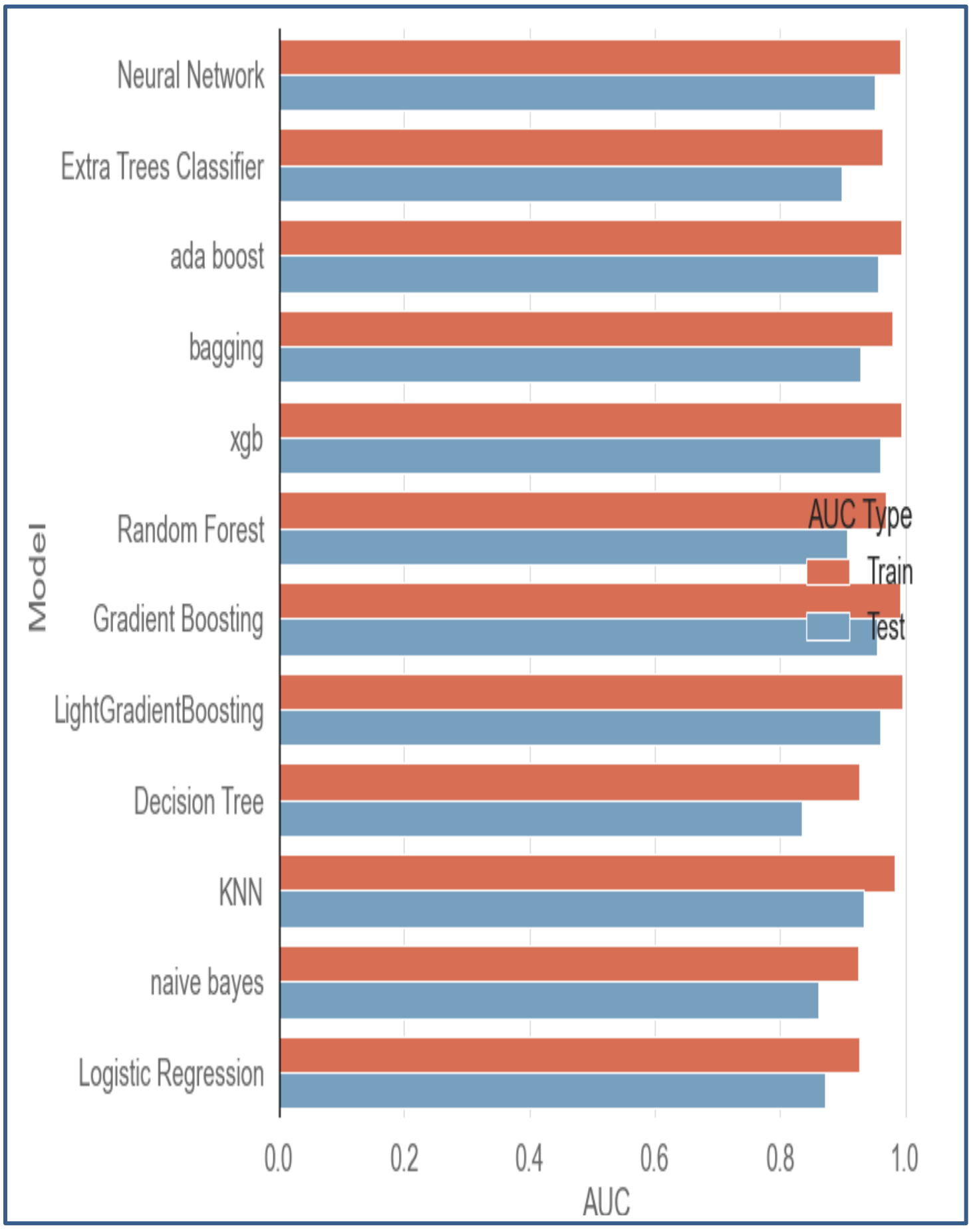
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.94 | 0.96 | 0.95 | 11776 |
| 1 | 0.95 | 0.92 | 0.93 | 9005 |
| accuracy | | | 0.94 | 20781 |
| macro avg | 0.94 | 0.94 | 0.94 | 20781 |
| weighted avg | 0.94 | 0.94 | 0.94 | 20781 |

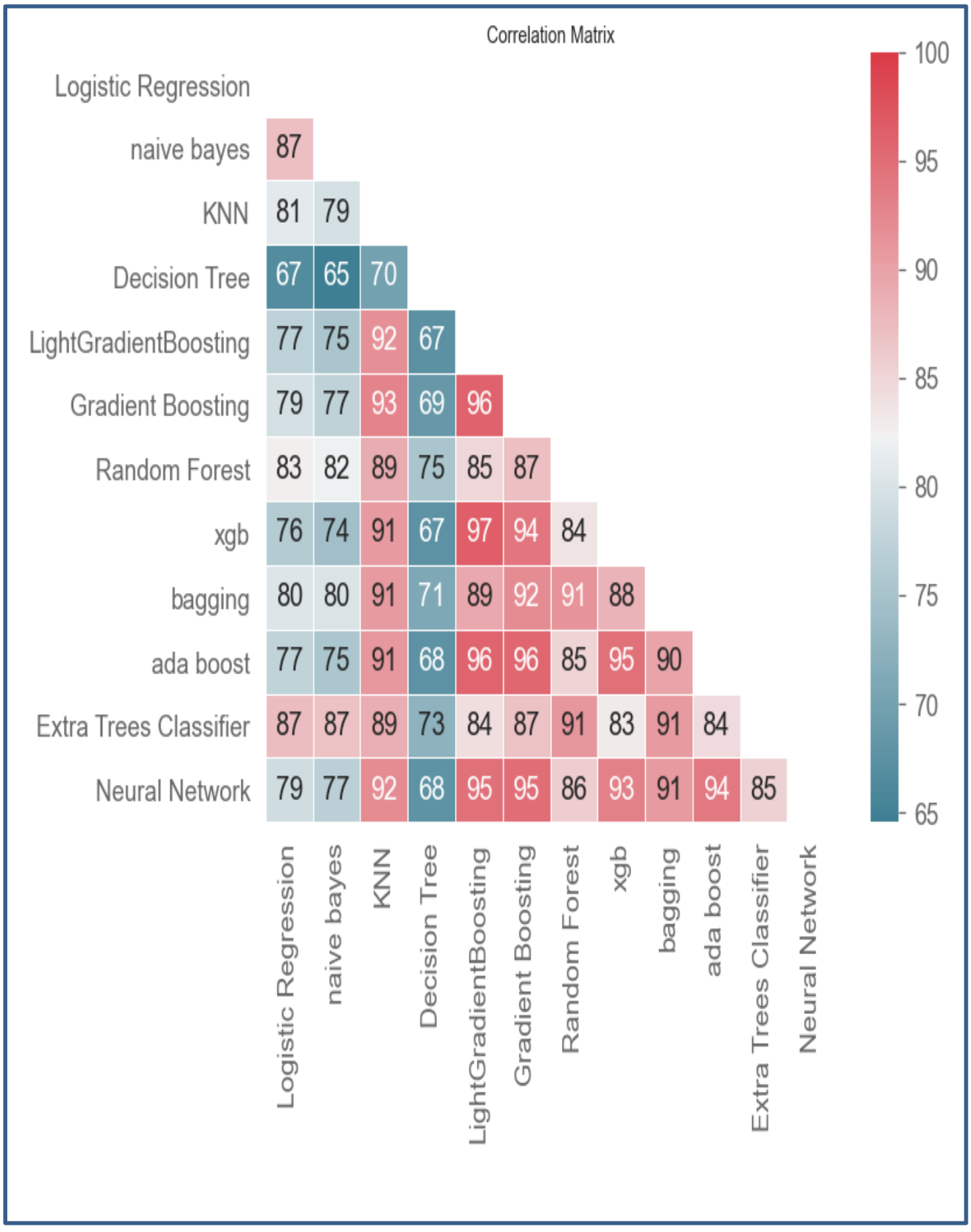
## VIF Data

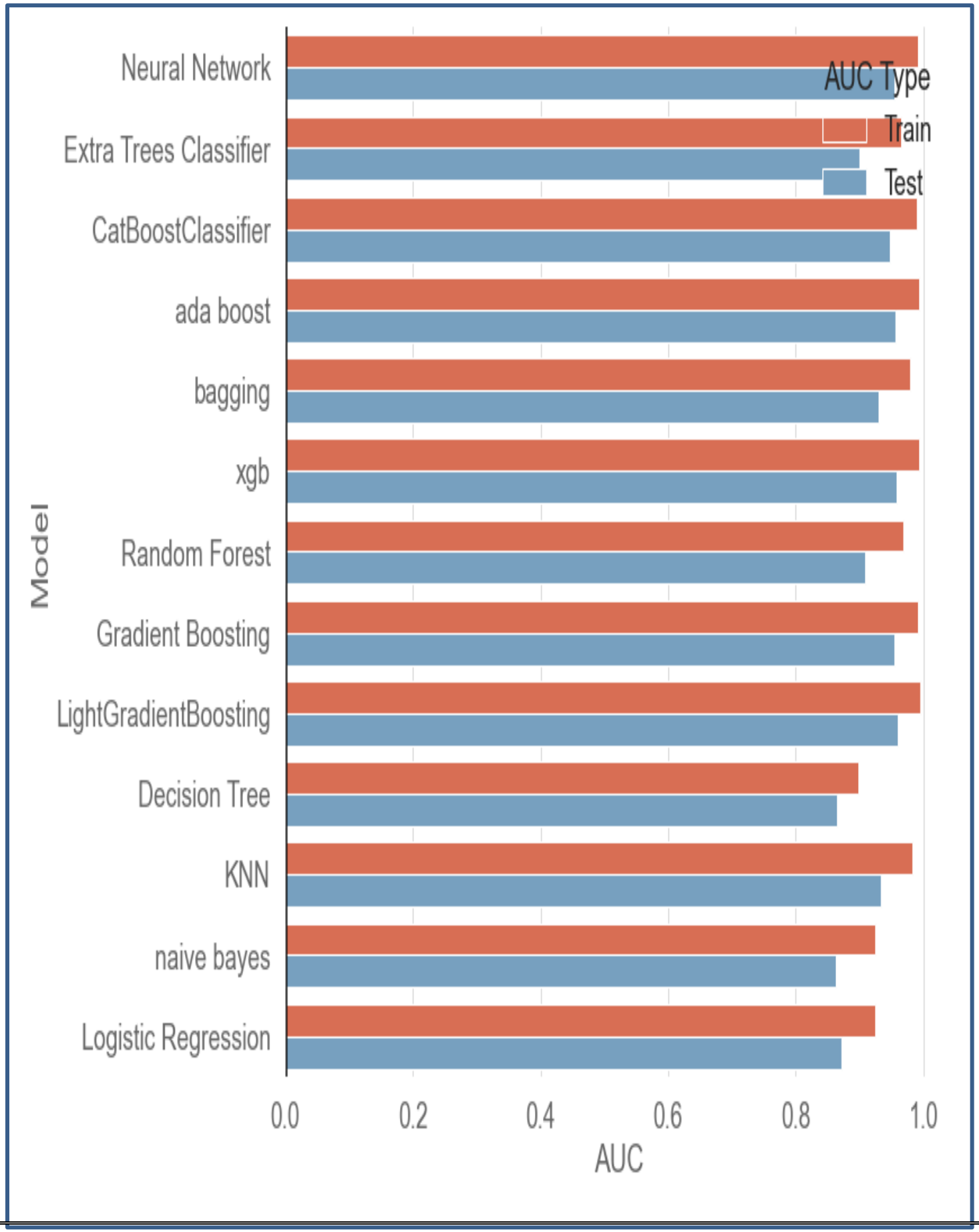


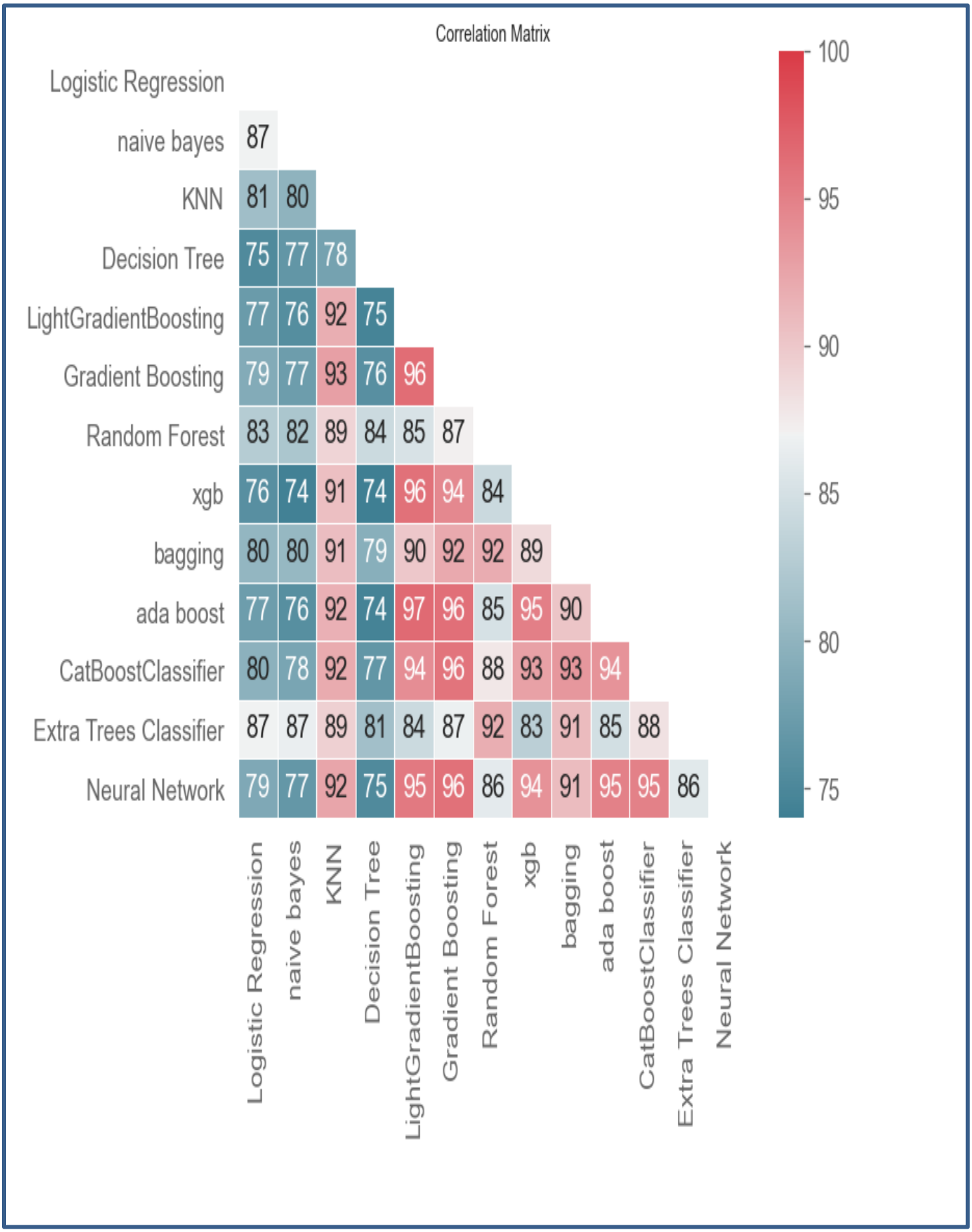

Misclassified Counts: 1033
accuracy..... 0.9503
precision score..... 0.9505
recall score..... 0.9482
classification_report

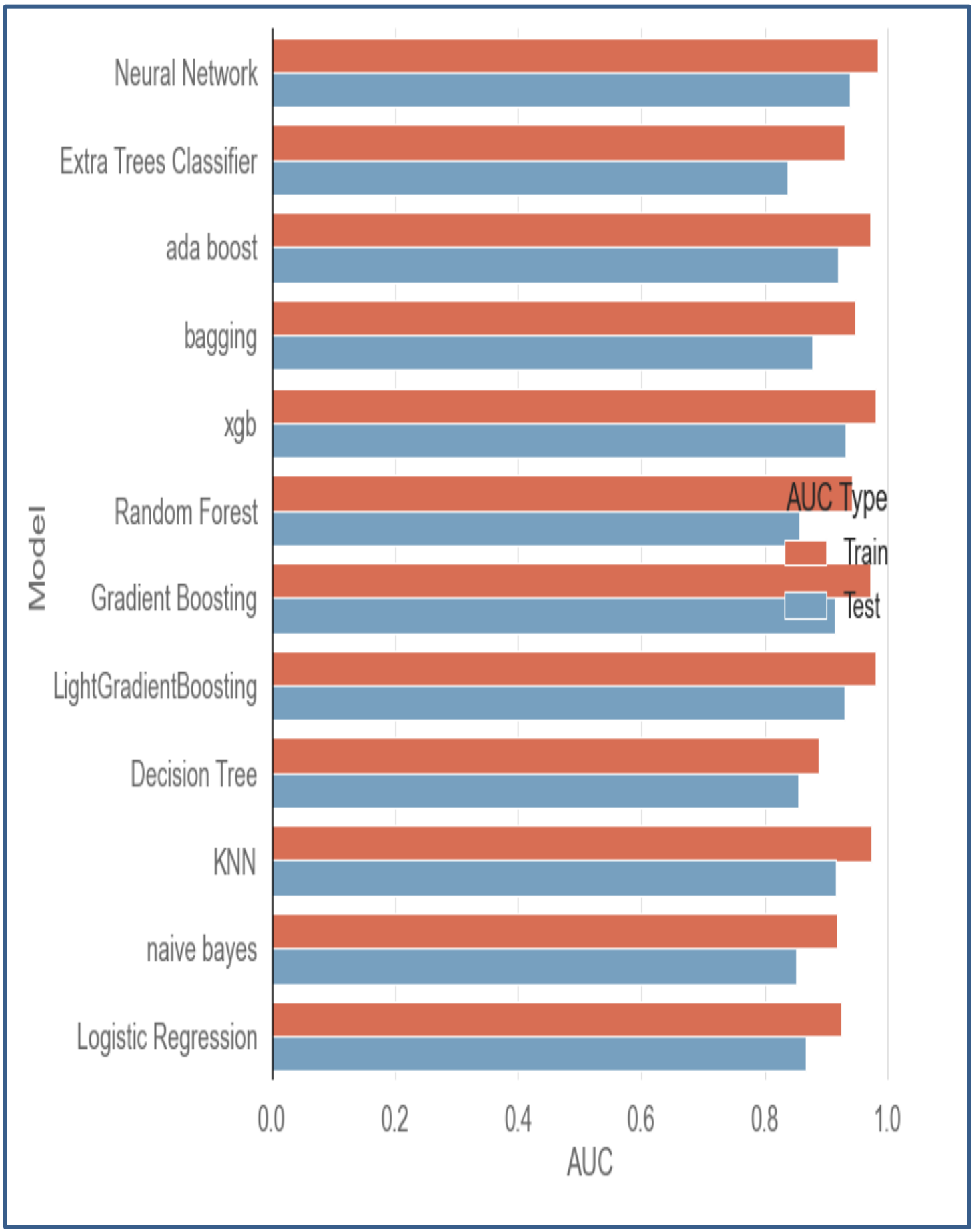
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.96 | 0.96 | 11776 |
| 1 | 0.95 | 0.93 | 0.94 | 9005 |
| accuracy | | | 0.95 | 20781 |
| macro avg | 0.95 | 0.95 | 0.95 | 20781 |
| weighted avg | 0.95 | 0.95 | 0.95 | 20781 |

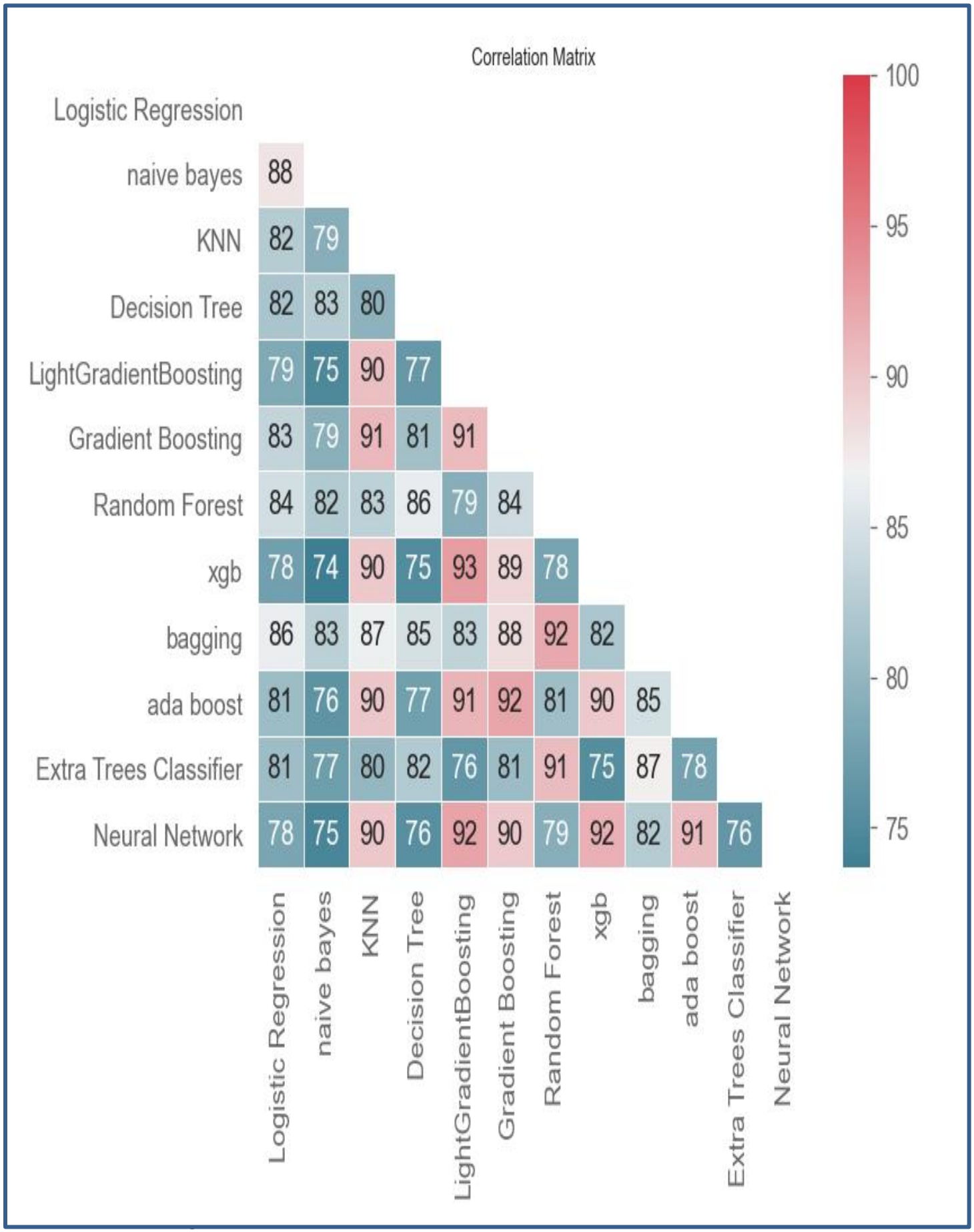

Neural Network
Extra Trees Classifier
ada boost
bagging
xgb
Random Forest
Gradient Boosting
LightGradientBoosting
Decision Tree
KNN
naive bayes
Logistic Regression
Model
AUC Type
Train
Test
0.0
0.2
0.4
0.6
0.8
1.0
AUC

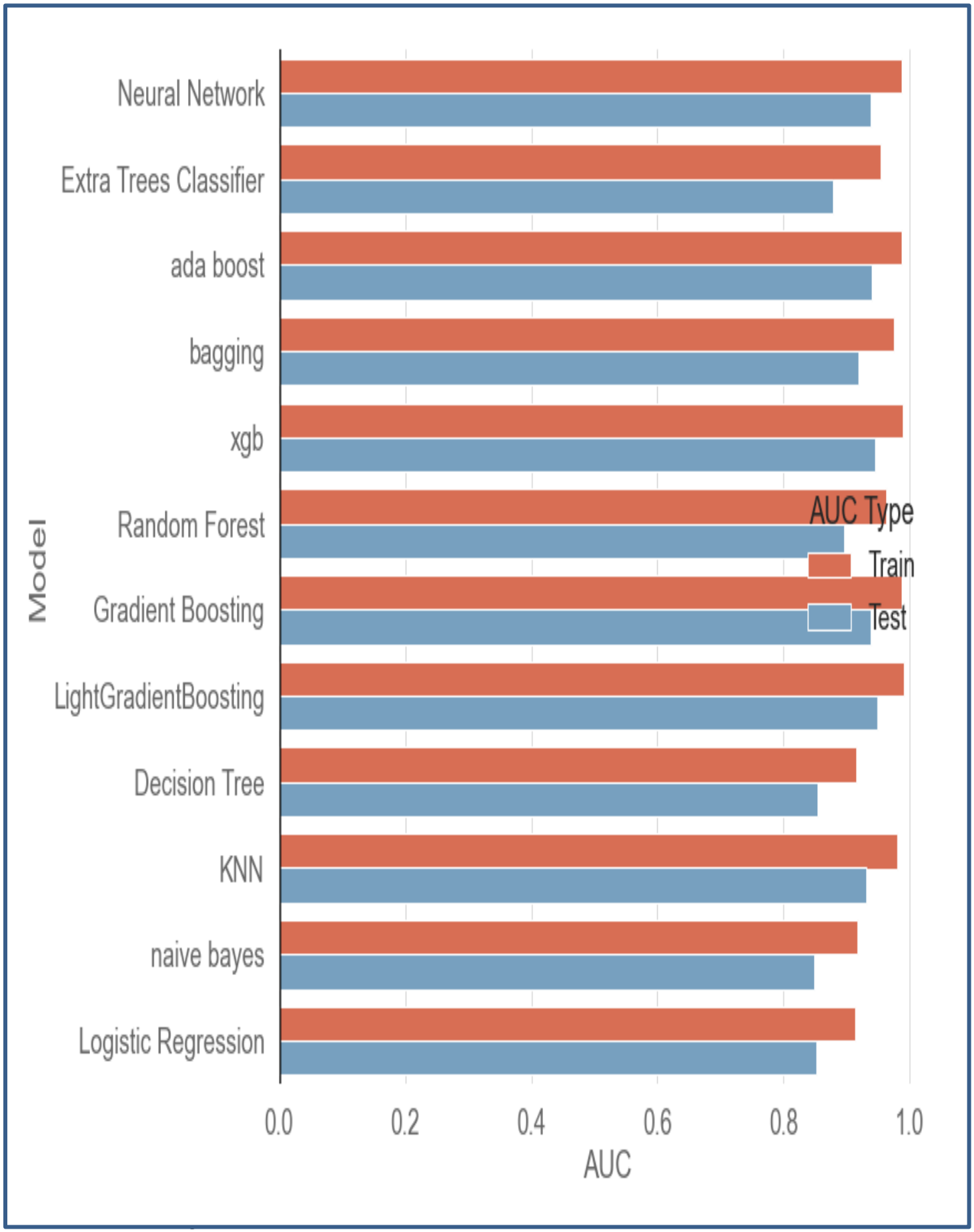

Correlation Matrix
Logistic Regression
naive bayes 87
KNN 81 79
Decision Tree 67 65 70
LightGradientBoosting 77 75 92 67
Gradient Boosting 79 77 93 69 96
Random Forest 83 82 89 75 85 87
xgb 76 74 91 67 97 94 84
bagging 80 80 91 71 89 92 91 88
ada boost 77 75 91 68 96 96 85 95 90
Extra Trees Classifier 87 87 89 73 84 87 91 83 91 84
Neural Network 79 77 92 68 95 95 86 93 91 94 85
Logistic Regression
naive bayes
KNN
Decision Tree
LightGradientBoosting
Gradient Boosting
Random Forest
xgb
bagging
ada boost
Extra Trees Classifier
Neural Network
100
95
90
85
80
75
70
65

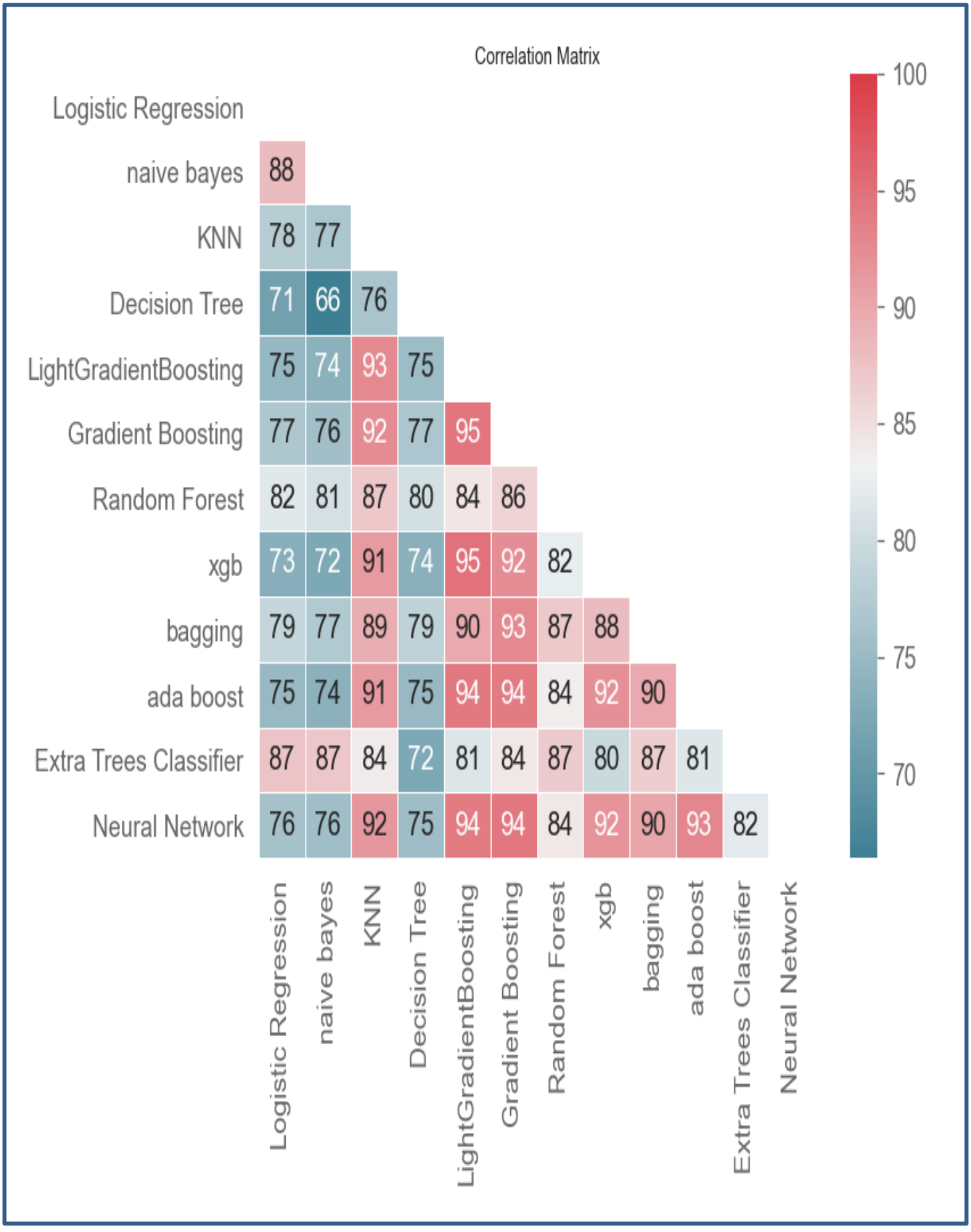

Neural Network
Extra Trees Classifier
CatBoostClassifier
ada boost
bagging
xgb
Random Forest
Gradient Boosting
LightGradientBoosting
Decision Tree
KNN
naive bayes
Logistic Regression
Model
AUC Type
Train
Test
0.0
0.2
0.4
0.6
0.8
1.0
AUC


Correlation Matrix
Logistic Regression
naive bayes 87
KNN 81 80
Decision Tree 75 77 78
LightGradientBoosting 77 76 92 75
Gradient Boosting 79 77 93 76 96
Random Forest 83 82 89 84 85 87
xgb 76 74 91 74 96 94 84
bagging 80 80 91 79 90 92 92 89
ada boost 77 76 92 74 97 96 85 95 90
CatBoostClassifier 80 78 92 77 94 96 88 93 93 94
Extra Trees Classifier 87 87 89 81 84 87 92 83 91 85 88
Neural Network 79 77 92 75 95 96 86 94 91 95 95 86
Logistic Regression
naive bayes
KNN
Decision Tree
LightGradientBoosting
Gradient Boosting
Random Forest
xgb
bagging
ada boost
CatBoostClassifier
Extra Trees Classifier
Neural Network
100
95
90
85
80
75


Neural Network
Extra Trees Classifier
ada boost
bagging
xgb
Random Forest
Gradient Boosting
LightGradientBoosting
Decision Tree
KNN
naive bayes
Logistic Regression
Model
AUC Type
Train
Test
0.0
0.2
0.4
0.6
0.8
1.0
AUC


Correlation Matrix
Logistic Regression
naive bayes 88
KNN 82 79
Decision Tree 82 83 80
LightGradientBoosting 79 75 90 77
Gradient Boosting 83 79 91 81 91
Random Forest 84 82 83 86 79 84
xgb 78 74 90 75 93 89 78
bagging 86 83 87 85 83 88 92 82
ada boost 81 76 90 77 91 92 81 90 85
Extra Trees Classifier 81 77 80 82 76 81 91 75 87 78
Neural Network 78 75 90 76 92 90 79 92 82 91 76
Logistic Regression
naive bayes
KNN
Decision Tree
LightGradientBoosting
Gradient Boosting
Random Forest
xgb
bagging
ada boost
Extra Trees Classifier
Neural Network
100
95
90
85
80
75


Neural Network
Extra Trees Classifier
ada boost
bagging
xgb
Random Forest
Gradient Boosting
LightGradientBoosting
Decision Tree
KNN
naive bayes
Logistic Regression
Model
AUC Type
Train
Test
0.0
0.2
0.4
0.6
0.8
1.0
AUC


Correlation Matrix
Logistic Regression
naive bayes 88
KNN 78 77
Decision Tree 71 66 76
LightGradientBoosting 75 74 93 75
Gradient Boosting 77 76 92 77 95
Random Forest 82 81 87 80 84 86
xgb 73 72 91 74 95 92 82
bagging 79 77 89 79 90 93 87 88
ada boost 75 74 91 75 94 94 84 92 90
Extra Trees Classifier 87 87 84 72 81 84 87 80 87 81
Neural Network 76 76 92 75 94 94 84 92 90 93 82
Logistic Regression
naive bayes
KNN
Decision Tree
LightGradientBoosting
Gradient Boosting
Random Forest
xgb
bagging
ada boost
Extra Trees Classifier
Neural Network
100
95
90
85
80
75
70

## 16 Model Best Params

| | Model | BestParam |
|---|---|---|
| 0 | Neural Network | {'solver': 'lbfgs', 'max_iter': 2000, 'learnin... |
| 1 | Extra Trees Classifier | {'n_jobs': -1, 'min_samples_split': 0.005, 'mi... |
| 2 | ada boost | {'learning_rate': 0.5, 'base_estimator': Decis... |
| 3 | bagging | {'n_jobs': -1, 'n_estimators': 200, 'max_featu... |
| 4 | xgb | {'subsample': 0.8, 'reg_alpha': 0.01, 'min_chi... |
| 5 | Random Forest | {'min_samples_split': 0.01, 'min_samples_leaf'... |
| 6 | Gradient Boosting | {'subsample': 0.8, 'min_samples_split': 0.01, ... |
| 7 | LightGradientBoosting | {'subsample': 0.9, 'reg_alpha': 1, 'min_child_... |
| 8 | Decision Tree | {'splitter': 'best', 'min_samples_split': 0.05... |
| 9 | KNN | {'weights': 'distance', 'p': 1, 'n_neighbors':... |
| 10 | naive bayes | {'var_smoothing': 0.012328467394420659} |
| 11 | Logistic Regression | {'solver': 'liblinear', 'penalty': 'l1', 'n_jo... |

**Unbinned**

| | Model | BestParam |
|---|---|---|
| 0 | Neural Network | {'solver': 'sgd', 'max_iter': 300, 'learning_r... |
| 1 | Extra Trees Classifier | {'n_jobs': -1, 'min_samples_split': 0.005, 'mi... |
| 2 | CatBoostClassifier | {'learning_rate': 0.3, 'depth': 9} |
| 3 | ada boost | {'learning_rate': 0.5, 'base_estimator': Decis... |
| 4 | bagging | {'n_jobs': -1, 'n_estimators': 200, 'max_featu... |
| 5 | xgb | {'subsample': 0.7, 'reg_alpha': 0.01, 'min_chi... |
| 6 | Random Forest | {'min_samples_split': 0.01, 'min_samples_leaf'... |
| 7 | Gradient Boosting | {'subsample': 0.8, 'min_samples_split': 0.01, ... |
| 8 | LightGradientBoosting | {'subsample': 0.7, 'reg_alpha': 0.01, 'min_chi... |
| 9 | Decision Tree | {'splitter': 'best', 'min_samples_split': 0.05... |
| 10 | KNN | {'weights': 'distance', 'p': 1, 'n_neighbors':... |
| 11 | naive bayes | {'var_smoothing': 0.0008111308307896872} |
| 12 | Logistic Regression | {'solver': 'liblinear', 'penalty': 'l1', 'n_jo... |

**Binned**

| | Model | BestParam |
|---|---|---|
| 0 | Neural Network | {'solver': 'lbfgs', 'max_iter': 2000, 'learnin... |
| 1 | Extra Trees Classifier | {'n_jobs': -1, 'min_samples_split': 0.005, 'mi... |
| 2 | ada boost | {'learning_rate': 0.5, 'base_estimator': Decis... |
| 3 | bagging | {'n_jobs': -1, 'n_estimators': 200, 'max_featu... |
| 4 | xgb | {'subsample': 0.8, 'reg_alpha': 0.01, 'min_chi... |
| 5 | Random Forest | {'min_samples_split': 0.01, 'min_samples_leaf'... |
| 6 | Gradient Boosting | {'subsample': 0.8, 'min_samples_split': 0.01, ... |
| 7 | LightGradientBoosting | {'subsample': 0.7, 'reg_alpha': 0.01, 'min_chi... |
| 8 | Decision Tree | {'splitter': 'best', 'min_samples_split': 0.05... |
| 9 | KNN | {'weights': 'distance', 'p': 1, 'n_neighbors':... |
| 10 | naive bayes | {'var_smoothing': 0.012328467394420659} |
| 11 | Logistic Regression | {'solver': 'liblinear', 'penalty': 'l2', 'n_jo... |

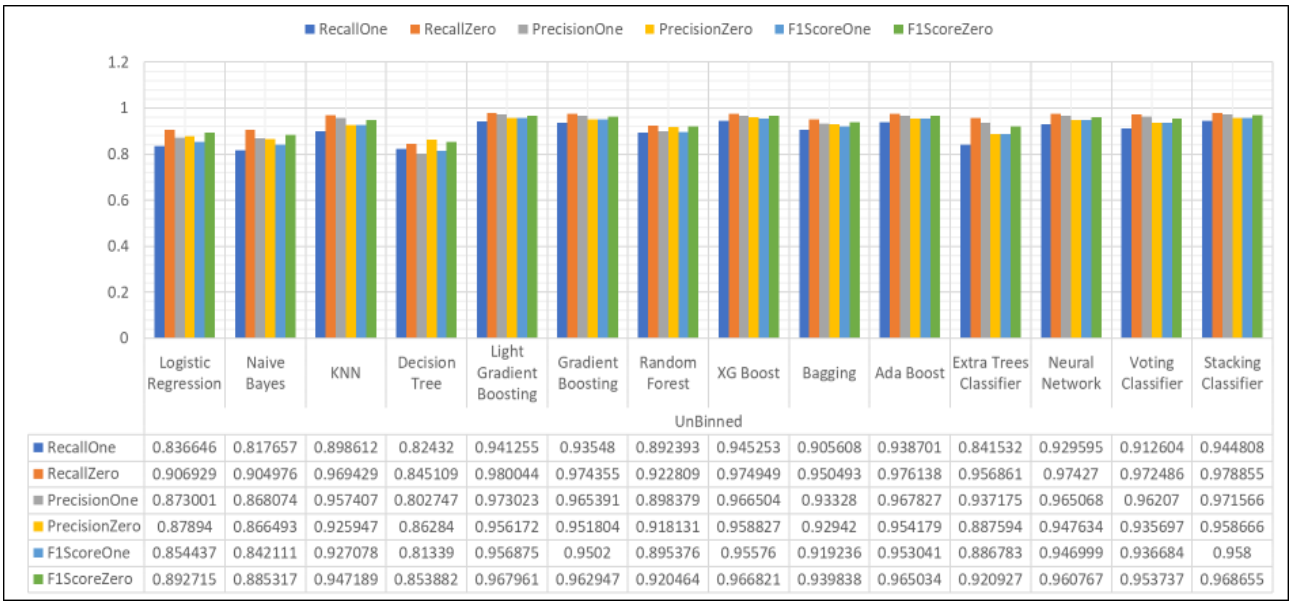
**VIF**

| | Model | BestParam |
|---|---|---|
| 0 | Neural Network | {'solver': 'lbfgs', 'max_iter': 2000, 'learnin... |
| 1 | Extra Trees Classifier | {'n_jobs': -1, 'min_samples_split': 0.005, 'mi... |
| 2 | ada boost | {'learning_rate': 0.5, 'base_estimator': Decis... |
| 3 | bagging | {'n_jobs': -1, 'n_estimators': 200, 'max_featu... |
| 4 | xgb | {'subsample': 0.8, 'reg_alpha': 0.01, 'min_chi... |
| 5 | Random Forest | {'min_samples_split': 0.005, 'min_samples_leaf... |
| 6 | Gradient Boosting | {'subsample': 0.8, 'min_samples_split': 0.01, ... |
| 7 | LightGradientBoosting | {'subsample': 0.7, 'reg_alpha': 0.01, 'min_chi... |
| 8 | Decision Tree | {'splitter': 'best', 'min_samples_split': 0.05... |
| 9 | KNN | {'weights': 'distance', 'p': 1, 'n_neighbors':... |
| 10 | naive bayes | {'var_smoothing': 0.1873817422860384} |
| 11 | Logistic Regression | {'solver': 'liblinear', 'penalty': 'l1', 'n_jo... |

**PCA**

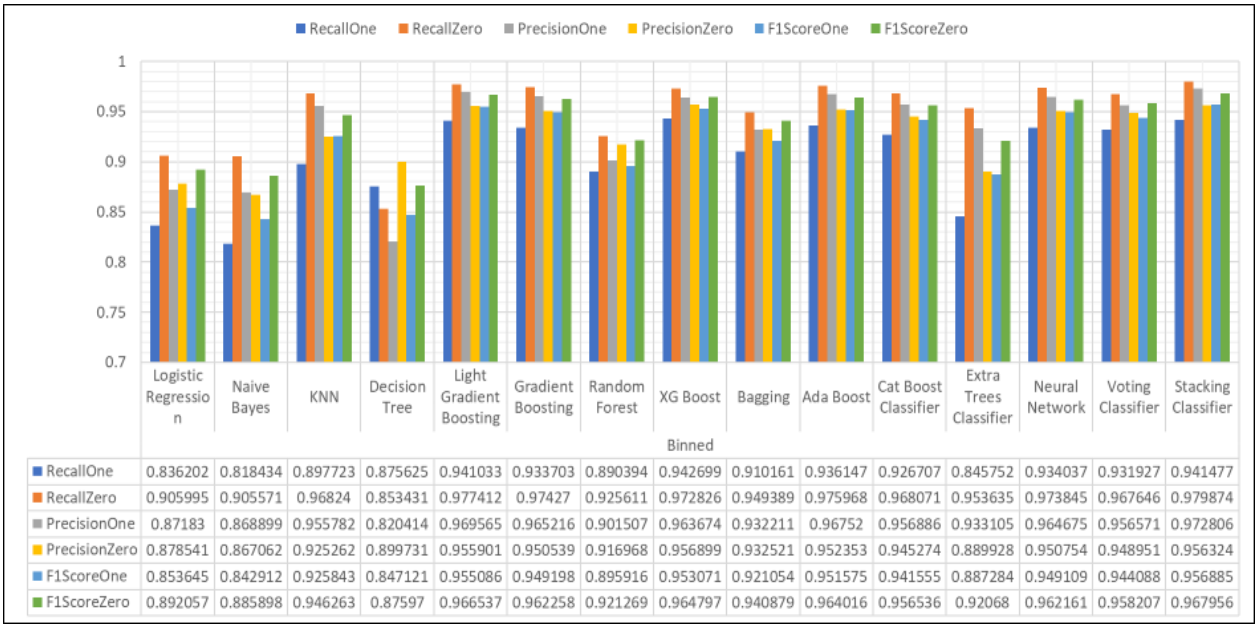
## 17 Modelling Results

# Modelling Results - UnBinned



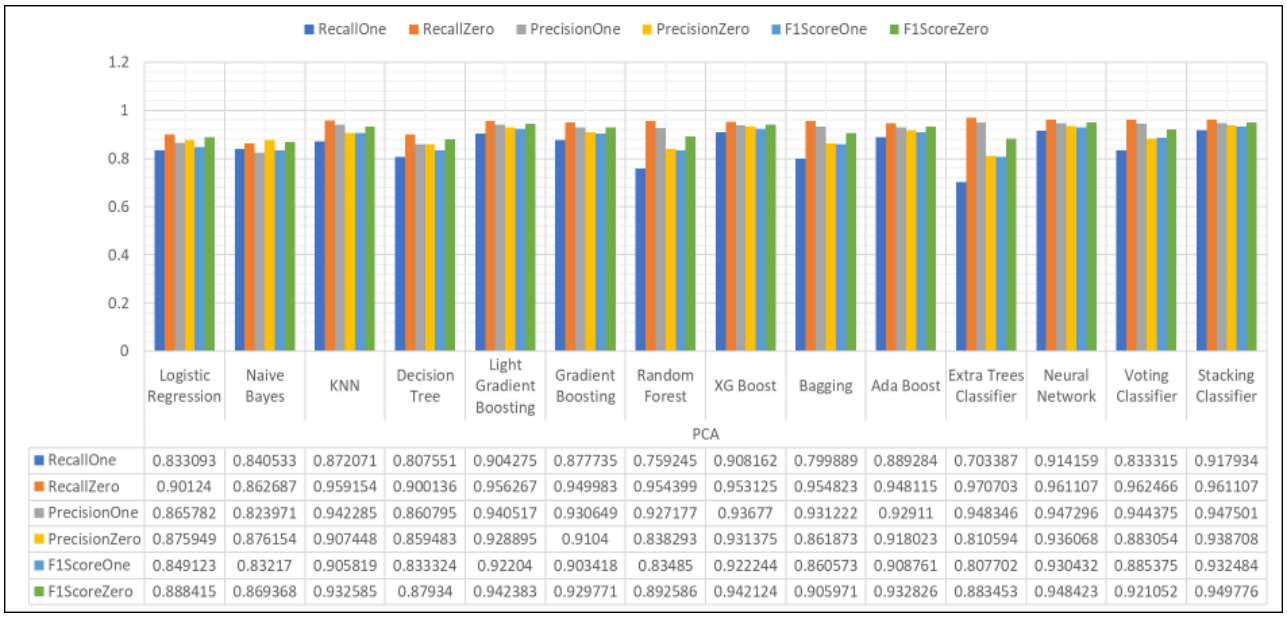

| | Logistic Regression | Naive Bayes | KNN | Decision Tree | Light Gradient Boosting | Gradient Boosting | Random Forest | XG Boost | Bagging | Ada Boost | Extra Trees Classifier | Neural Network | Voting Classifier | Stacking Classifier |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RecallOne | 0.836646 | 0.817657 | 0.898612 | 0.82432 | 0.941255 | 0.93548 | 0.892393 | 0.945253 | 0.905608 | 0.938701 | 0.841532 | 0.929595 | 0.912604 | 0.944808 |
| RecallZero | 0.906929 | 0.904976 | 0.969429 | 0.845109 | 0.980044 | 0.974355 | 0.922809 | 0.974949 | 0.950493 | 0.976138 | 0.956861 | 0.97427 | 0.972486 | 0.978855 |
| PrecisionOne | 0.873001 | 0.868074 | 0.957407 | 0.802747 | 0.973023 | 0.965391 | 0.898379 | 0.966504 | 0.93328 | 0.967827 | 0.937175 | 0.965068 | 0.96207 | 0.971566 |
| PrecisionZero | 0.87894 | 0.866493 | 0.925947 | 0.86284 | 0.956172 | 0.951804 | 0.918131 | 0.958827 | 0.92942 | 0.954179 | 0.887594 | 0.947634 | 0.935697 | 0.958666 |
| F1ScoreOne | 0.854437 | 0.842111 | 0.927078 | 0.81339 | 0.956875 | 0.9502 | 0.895376 | 0.95576 | 0.919236 | 0.953041 | 0.886783 | 0.946999 | 0.936684 | 0.958 |
| F1ScoreZero | 0.892715 | 0.885317 | 0.947189 | 0.853882 | 0.967961 | 0.962947 | 0.920464 | 0.966821 | 0.939838 | 0.965034 | 0.920927 | 0.960767 | 0.953737 | 0.968655 |

# Modelling Results - Binned




| | Logistic Regressio n | Naive Bayes | KNN | Decision Tree | Light Gradient Boosting | Gradient Boosting | Random Forest | XG Boost | Bagging | Ada Boost | Cat Boost Classifier | Extra Trees Classifier | Neural Network | Voting Classifier | Stacking Classifier |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RecallOne | 0.836202 | 0.818434 | 0.897723 | 0.875625 | 0.941033 | 0.933703 | 0.890394 | 0.942699 | 0.910161 | 0.936147 | 0.926707 | 0.845752 | 0.934037 | 0.931927 | 0.941477 |
| RecallZero | 0.905995 | 0.905571 | 0.96824 | 0.853431 | 0.977412 | 0.97427 | 0.925611 | 0.972826 | 0.949389 | 0.975968 | 0.968071 | 0.953635 | 0.973845 | 0.967646 | 0.979874 |
| PrecisionOne | 0.87183 | 0.868899 | 0.955782 | 0.820414 | 0.969565 | 0.965216 | 0.901507 | 0.963674 | 0.932211 | 0.96752 | 0.956886 | 0.933105 | 0.964675 | 0.956571 | 0.972806 |
| PrecisionZero | 0.878541 | 0.867062 | 0.925262 | 0.899731 | 0.955901 | 0.950539 | 0.916968 | 0.956899 | 0.932521 | 0.952353 | 0.945274 | 0.889928 | 0.950754 | 0.948951 | 0.956324 |
| F1ScoreOne | 0.853645 | 0.842912 | 0.925843 | 0.847121 | 0.955086 | 0.949198 | 0.895916 | 0.953071 | 0.921054 | 0.951575 | 0.941555 | 0.887284 | 0.949109 | 0.944088 | 0.956885 |
| F1ScoreZero | 0.892057 | 0.885898 | 0.946263 | 0.87597 | 0.966537 | 0.962258 | 0.921269 | 0.964797 | 0.940879 | 0.964016 | 0.956536 | 0.92068 | 0.962161 | 0.958207 | 0.967956 |

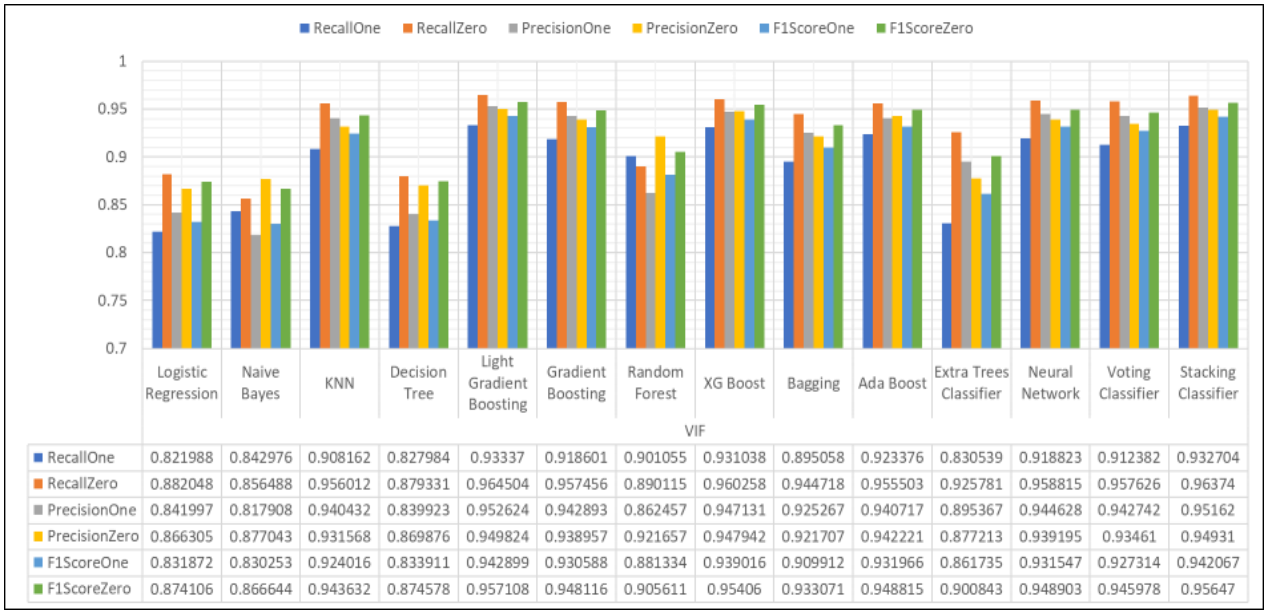
# Modelling Results - PCA




PCA

| | Logistic Regression | Naive Bayes | KNN | Decision Tree | Light Gradient Boosting | Gradient Boosting | Random Forest | XG Boost | Bagging | Ada Boost | Extra Trees Classifier | Neural Network | Voting Classifier | Stacking Classifier |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RecallOne | 0.833093 | 0.840533 | 0.872071 | 0.807551 | 0.904275 | 0.877735 | 0.759245 | 0.908162 | 0.799889 | 0.889284 | 0.703387 | 0.914159 | 0.833315 | 0.917934 |
| RecallZero | 0.90124 | 0.862687 | 0.959154 | 0.900136 | 0.956267 | 0.949983 | 0.954399 | 0.953125 | 0.954823 | 0.948115 | 0.970703 | 0.961107 | 0.962466 | 0.961107 |
| PrecisionOne | 0.865782 | 0.823971 | 0.942285 | 0.860795 | 0.940517 | 0.930649 | 0.927177 | 0.93677 | 0.931222 | 0.92911 | 0.948346 | 0.947296 | 0.944375 | 0.947501 |
| PrecisionZero | 0.875949 | 0.876154 | 0.907448 | 0.859483 | 0.928895 | 0.9104 | 0.838293 | 0.931375 | 0.861873 | 0.918023 | 0.810594 | 0.936068 | 0.883054 | 0.938708 |
| F1ScoreOne | 0.849123 | 0.83217 | 0.905819 | 0.833324 | 0.92204 | 0.903418 | 0.83485 | 0.922244 | 0.860573 | 0.908761 | 0.807702 | 0.930432 | 0.885375 | 0.932484 |
| F1ScoreZero | 0.888415 | 0.869368 | 0.932585 | 0.87934 | 0.942383 | 0.929771 | 0.892586 | 0.942124 | 0.905971 | 0.932826 | 0.883453 | 0.948423 | 0.921052 | 0.949776 |

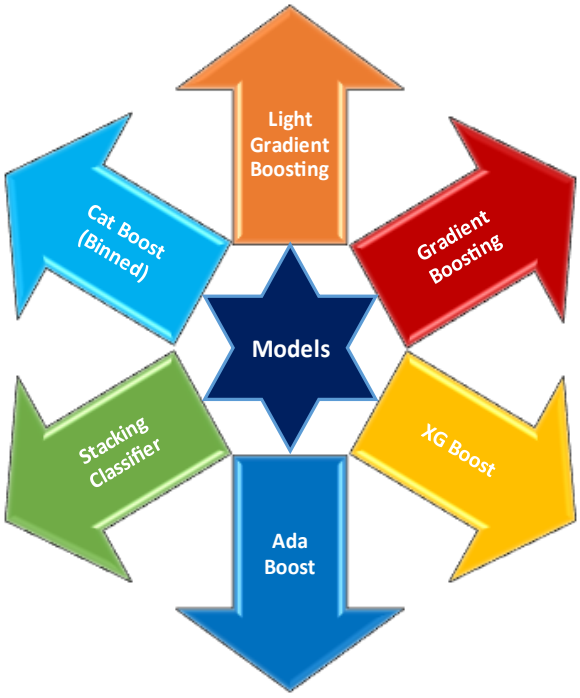
# Modelling Results – VIF + ChiSquare




VIF

| | Logistic Regression | Naive Bayes | KNN | Decision Tree | Light Gradient Boosting | Gradient Boosting | Random Forest | XG Boost | Bagging | Ada Boost | Extra Trees Classifier | Neural Network | Voting Classifier | Stacking Classifier |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RecallOne | 0.821988 | 0.842976 | 0.908162 | 0.827984 | 0.93337 | 0.918601 | 0.901055 | 0.931038 | 0.895058 | 0.923376 | 0.830539 | 0.918823 | 0.912382 | 0.932704 |
| RecallZero | 0.882048 | 0.856488 | 0.956012 | 0.879331 | 0.964504 | 0.957456 | 0.890115 | 0.960258 | 0.944718 | 0.955503 | 0.925781 | 0.958815 | 0.957626 | 0.96374 |
| PrecisionOne | 0.841997 | 0.817908 | 0.940432 | 0.839923 | 0.952624 | 0.942893 | 0.862457 | 0.947131 | 0.925267 | 0.940717 | 0.895367 | 0.944628 | 0.942742 | 0.95162 |
| PrecisionZero | 0.866305 | 0.877043 | 0.931568 | 0.869876 | 0.949824 | 0.938957 | 0.921657 | 0.947942 | 0.921707 | 0.942221 | 0.877213 | 0.939195 | 0.93461 | 0.94931 |
| F1ScoreOne | 0.831872 | 0.830253 | 0.924016 | 0.833911 | 0.942899 | 0.930588 | 0.881334 | 0.939016 | 0.909912 | 0.931966 | 0.861735 | 0.931547 | 0.927314 | 0.942067 |
| F1ScoreZero | 0.874106 | 0.866644 | 0.943632 | 0.874578 | 0.957108 | 0.948116 | 0.905611 | 0.95406 | 0.933071 | 0.948815 | 0.900843 | 0.948903 | 0.945978 | 0.95647 |

## 18 Models Chosen




## 19 Deployment

# Web Portal - Options

**Pre-Processing**

- Drop Unwanted Columns
- Rename Columns
- Remove Duplicates (Optional)
- Handle Missing Values (Optional)
- Encode
- Remove Outliers (Optional)
- Scale

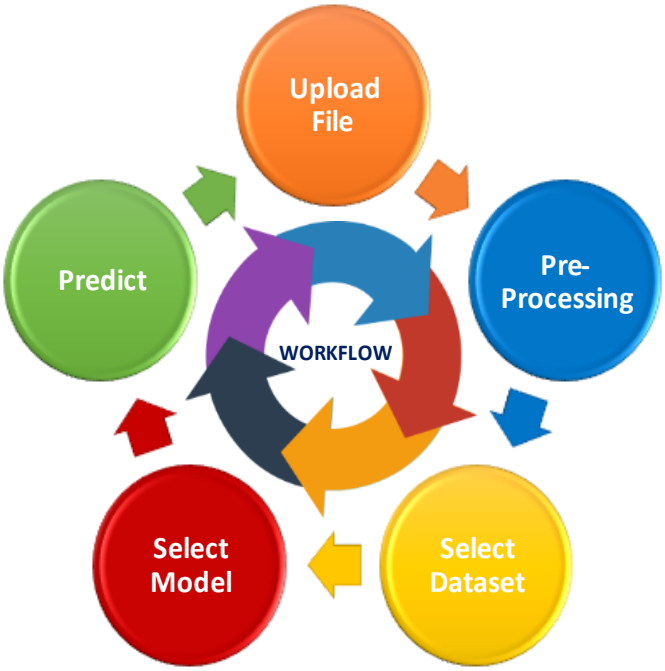
**Datasets**

- Un-Binned
- Binned

**Models**

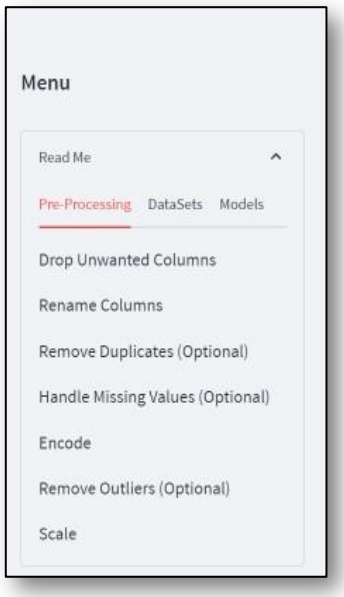
- Light Gradient Boosting
- Gradient Boosting
- XGB
- Ada Boost
- Stacking Classifier
- CatBoost Classifier (Binned)

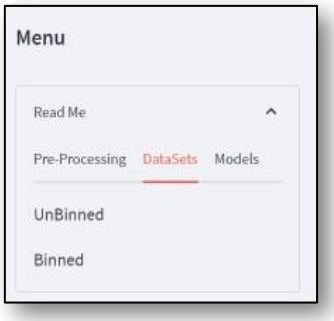
# Web Portal - Workflow

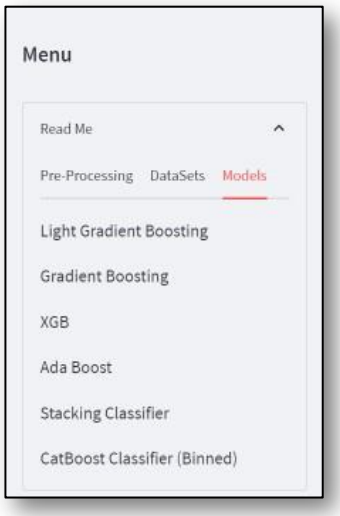



# Web Portal – Home

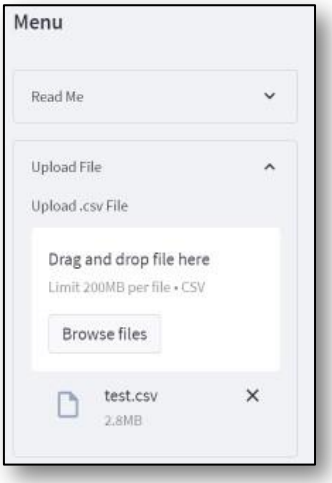
# Web Portal – Options

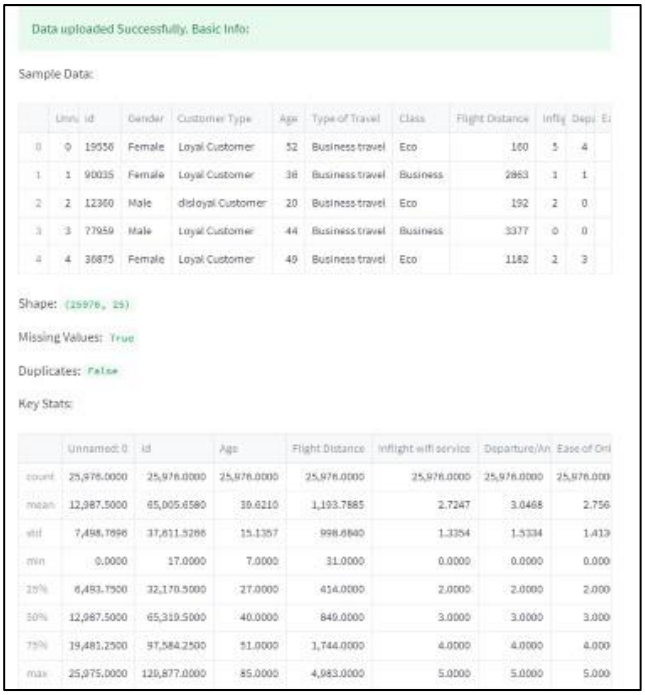


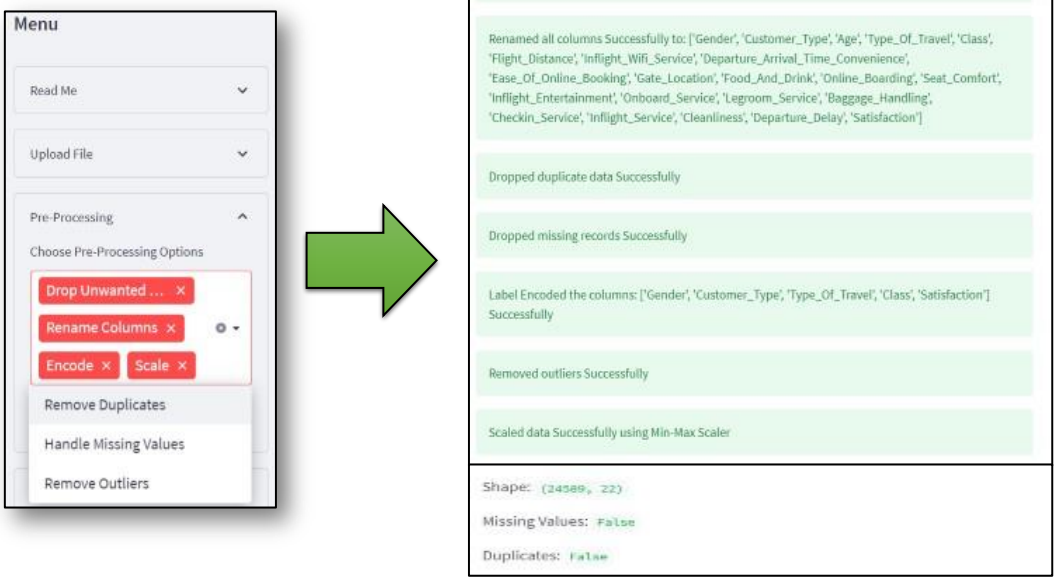




# Web Portal – Upload File



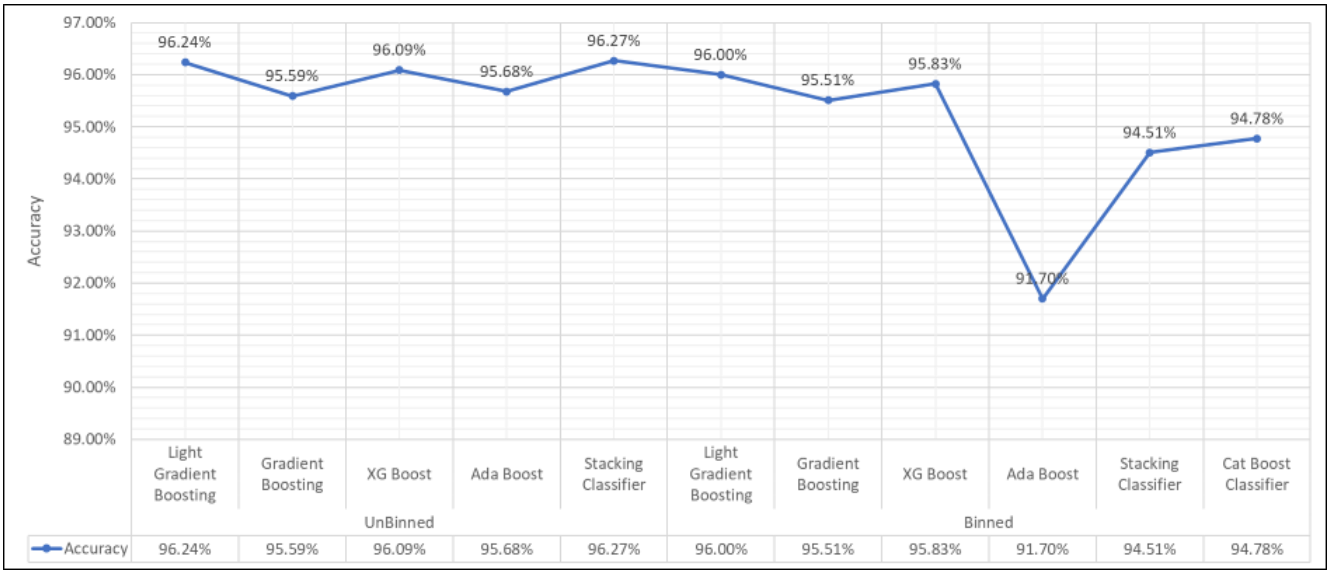
# Web Portal – PreProcessing




# Web Portal – Prediction

## 20 Prediction Results

# Web Portal – Prediction Results



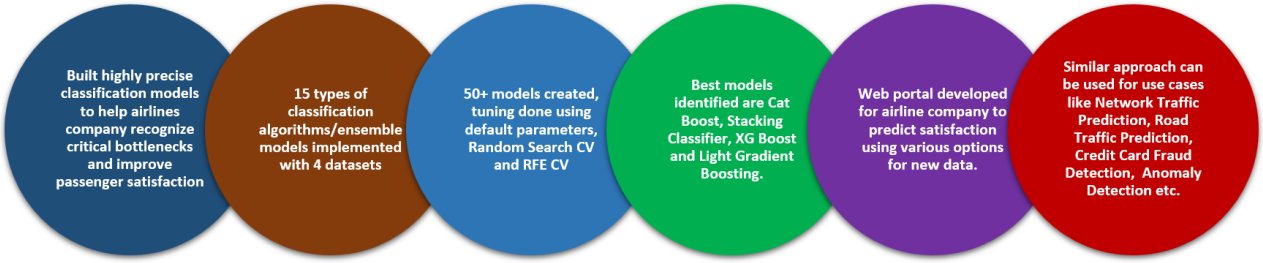
## 21 Future Work & Extension or Scope of improvements

The below items can be explored further as future work.

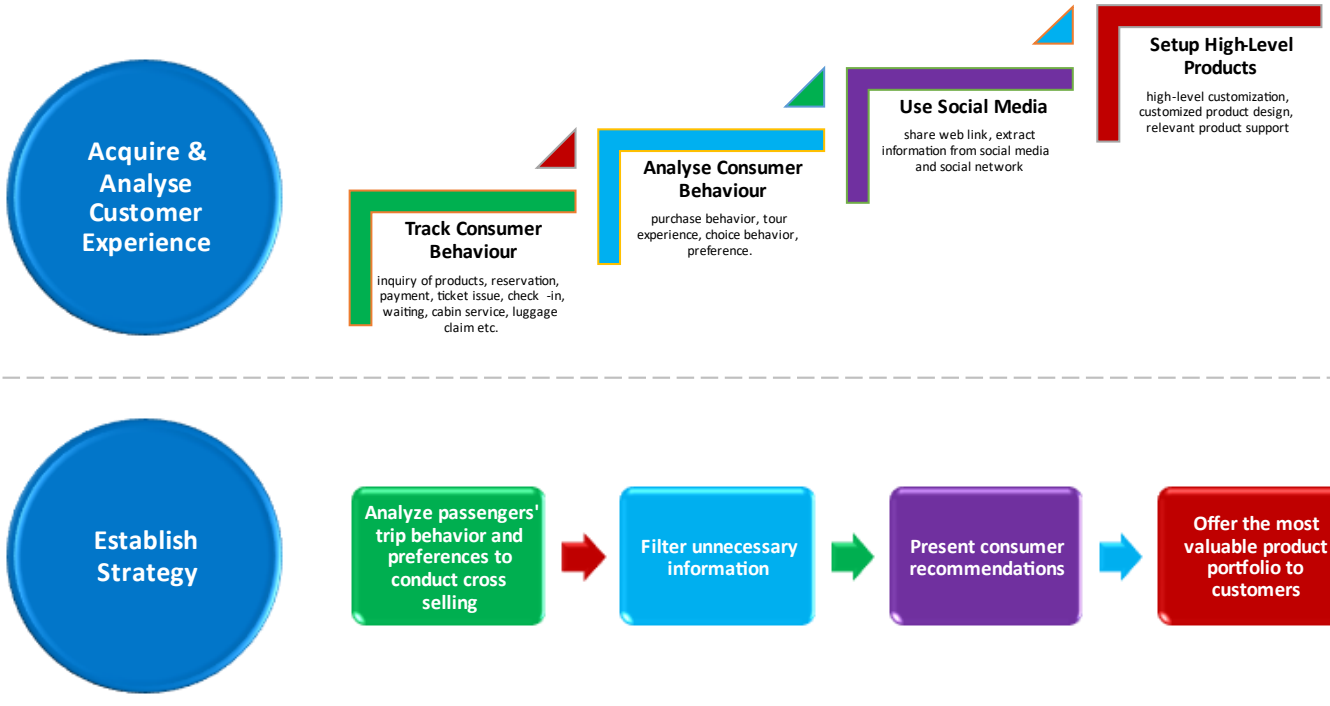
- Implement H20 model
- Implement Auto Encoders
- Implement FB Prophet models

## 22 Conclusions / Recommendations

# Conclusion




# Recommendations



## 23 Directions for future work

- If we can get the data from the all airline companies , we can enhance the current models with thoseextra features.
- For GAN, we can reuse the same model architecture and classes and enhance it from there. It requires a lot of hyper parameter tuning to arrive at the best model.

## 24 Bibliography / References


1. Sklearn : https://scikit-learn.org/stable/supervised_learning.html
2. https://machinelearningmastery.com/time-series-forecasting-performance-measures-with-python/
3. https://machinelearningmastery.com/bagging-ensemble-with-python/
4. https://mlwave.com/kaggle-ensembling-guide/
5. https://www.scikit-yb.org/en/latest/api/model_selection/importances.html
6. https://machinelearningmastery.com/calculate-feature-importance-with-python/
7. https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-lightgbm-and-catboost/
8. https://blog.cambridgespark.com/hyperparameter-tuning-in-xgboost-4ff9100a3b2f
9. https://info.cambridgespark.com/latest/getting-started-with-xgboost
10. Statsmodels : https://www.statsmodels.org/stable/index.html
11. Light Gradient Boosting : https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html
12. Categorical gradient boosting : https://arxiv.org/pdf/1810.11363.pdf
13. Tensor Flow : https://www.tensorflow.org/tutorials/structured_data/time_series
14. Plotly Graphs : https://plotly.com/python/horizontal-bar-charts/#horizontal-bar-chart-with-gobar
15. Machine Learning Visualization : https://www.scikit-yb.org/en/latest/
16. Keras : https://keras.io/guides/
17. Keras: Starting, stopping, and resuming training: https://www.pyimagesearch.com/2019/09/23/keras-starting-stopping-and-resuming-training/

## 25 Duly Completed Checklist

| | **Check list of items for the Final report** | **Y/N** |
|---|---|---|
| a) | Is the Cover page in proper format? | Y |
| b) | Is the Title page in proper format? | Y |
| c) | Is the Certificate from the Mentor in proper format? Has it been signed? | Y |
| d) | Is Abstract included in the Report? Is it properly written? | Y |
| e) | Does the Table of Contents page include chapter page numbers? | Y |
| f) | Does the Report contain a summary of the literature survey? | |
| | i. Are the Pages numbered properly? | Y |
| | ii. Are the Figures numbered properly? | Y |
| | iii. Are the Tables numbered properly? | Y |
| | iv. Are the Captions for the Figures and Tables proper? | Y |
| | v. Are the Appendices numbered? | Y |
| g) | Does the Report have Conclusion / Recommendations of the work? | Y |
| h) | Are References/Bibliography given in the Report? | Y |
| i) | Have the References been cited in the Report? | Y |
| j) | Is the citation of References / Bibliography in proper format? | Y |

*Table 3: Completion Checklist*