EMPLOYEE ATTRITION PREDICTION

CAPSTONE PROJECT

Submitted in partial fulfilment of the requirements of thePost Graduate Certification Program
in
Artificial Intelligence and Machine Learning

By

Name	Bits ID
KIRTI RANI	2020AIML518
VIJAY KUMAR	2020AIML525
ABHISHEK SHUKLA	2020AIML528
NIMANSHA MALIK	2020AIML531
K RAMAKRISHNAN	2020AIML591

Under the supervision of Mr. Satyaki Dasgupta

Project work carried out at

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) INDIA

(April, 2022)

PCAM ZC321 CAPSTONE PROJECT

EMPLOYEE ATTRITION PREDICTION

Submitted in partial fulfilment of the requirements of thePGP - Artificial Intelligence and Machine Learning

By

Name	Bits ID
KIRTI RANI	2020AIML518
VIJAY KUMAR	2020AIML525
ABHISHEK SHUKLA	2020AIML528
NIMANSHA MALIK	2020AIML531
K RAMAKRISHNAN	2020AIML591

Under the supervision of Mr. Satyaki Dasgupta

Project work carried out at

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCEPILANI (RAJASTHAN)

April, 2022

ACKNOWLEDGEMENTS

A project like Employee Attrition Prediction 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 Prof. N L Bhanu Murthy for being a pillar support for the AIML program and laying the foundation required through the basic mathematics and Regression Videos
Thank you to Prof. Y V K Ravi Kumar for clearly covering the topics related to Regression concepts.
Thank you to Dr. Chetana Gavankar for covering the core concepts of Classification and some of the 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. Srikanth Gunturu for covering the classification concepts clearly through the videos, 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. Pravin Pawar, for covering feature engineering through in depth technical sessions and practical applications.
Thank you to Prof. S.P. Vimal for laying the foundations through elaborate introductory Python videos and for covering core concepts of Unsupervised Learning.
Thank you to Prof. Raja Vadhana P for clearly covering the topics related to Association rule mining through videos and 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 Neural networks in depth with elaborate sessions and helping with multitude sources of information/study material

Thank you to TAs – for supporting at each stage with additional reference material, clarifying doubts through additional sessions and assignments
Thank you to Mr. Rahul Mohandasan for anchoring the entire cohort and being the support for all the technical and academic queries and issues.

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 timeseries.

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 **Employee Attrition Prediction**

and submitted by **Mr./Ms. KIRTI RANI, VIJAY KUMAR. ABHISHEK SHUKLA, NIMANSHA MALIK, K RAMAKRISHNAN ID No. 2020AIML518, 2020AIML525, 2020AIML528, 2020AIML531, 2020AIML591**

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

Place: Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date: 09/Apr/2022

Name: Mr. Satyaki Dasgupta

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Capstone Project entitled **Employee Attrition Prediction**

and submitted by **Ms. KIRTI RANI** ID No. **2020AIML518**

in partial fulfilment of the requirements of PCAM ZC321 Capstone Project, embodies the workdone by her under my supervision.

Place: Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date: 09/Apr/2022

Name: Mr. Satyaki Dasgupta

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Capstone Project entitled **Employee Attrition Prediction**

and submitted by **Mr. VIJAY KUMAR** ID No. **2020AIML525**

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

Place: Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date: 09/Nov/2022

Name: Mr. Satyaki Dasgupta

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Capstone Project entitled **Employee Attrition Prediction**

and submitted by **Mr. ABHISHEK SHUKLA** ID No. **2020AIML528**

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

Place: Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date: 09/Apr/2022

Name: Mr. Satyaki Dasgupta

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Capstone Project entitled **Employee Attrition Prediction**

and submitted by **Ms. NIMANSHA MALIK** ID No. **2020AIML531**

in partial fulfilment of the requirements of PCAM ZC321 Capstone Project, embodies the workdone by her under my supervision.

Place: Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date: 09/Apr/2022

Name: Mr. Satyaki Dasgupta

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Capstone Project entitled **Employee Attrition Prediction**

and submitted by **Mr. K RAMAKRISHNAN** ID No. **2020AIML591**

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

Place: Kolkata

Signature of the Mentor

Satyaki Dasgupta

Date: 09/Apr/2022

Name: Mr. Satyaki Dasgupta

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI SECOND SEMESTER2021-22

PCAM ZC321 CAPSTONE PROJECT

Project Title	EMPLOYEE ATTRITION PREDICTION
Name of Mentor	Mr. SATYAKI DASGUPTA
Names of Students	KIRTI RANI VIJAY KUMAR ABHISHEK SHUKLA NIMANSHA MALIK K RAMAKRISHNAN
ID No. of Students	2020AIML518 2020AIML525 2020AIML528 2020AIML531 2020AIML591

Abstract

Employees are the pillars of the organization and workforce retention is one of the **key factors** in measuring the **organizational growth**. But swelling attrition levels across the businesses are strait-jacketing the distraught HR practitioner's ability to come out with good retention strategies. Objective is to build a model to **predict** employee attrition which could help human resource department **strategize hiring** and **retention** process.

This project follows a CRISP-DM (Cross Industry Standard Process for Data Mining) approach which comprises of 6 phases namely, Business understanding, Data understanding, Data preparation, Modelling, Evaluation and Deployment. Enough care and attention have been put to understand the data set with different analytics like univariate, bivariate, multivariate analysis, and several numerical and categorical plots for better visualization. Several hidden information is extracted for a deeper analysis to understand the behavior of employees and practices in the organization to come up with recommendations for better employee retention.

Project also aims in building a predictive **model** with good accuracy to predict if an employee is likely to leave or not with different classification techniques like Logistic Regression, Decision Tree, KNN, Naïve Bayes, Random Forest, etc. The dataset was split, using 80% for training the algorithm and 20% for testing achieving an accuracy of more than 85% in both train and test data set. Further enhancement and optimization have been made through Randomized search cross validation, 5-fold cross validation, boosting techniques, Random Forest algorithms and recursive feature elimination technique to find the best parameters for predicting if an employee is likely to leave or not which can help the HR team in strategizing hiring and retention process.

Need of Employee Attrition Prediction:

Managing workforce: If the supervisors or HR came to know about some employees that they will be planning to leave the company then they could get in touch with those employees and can convince them to stay back through various negotiations.

Smooth pipeline: If all the employees in the current project are working continuously on a project, then the pipeline of that project will be smooth but if one efficient asset of the project(employee) suddenly leaves that company, then the workflow will be disturbed and won't be so smooth any longer.

Hiring Management: If HR came to know about the employee of a particular project is willing to leave the company, then he/she can manage the number of hiring and they can get the valuable asset whenever they need for the efficient flow of work.

Importance:

- Organization success hinges much on the retention of employees
- Frequent attrition impacts the morale of the organization
- People emerge key competitive differentiators
- Hiring new employees is more expensive than retaining existing ones
- Losing knowledgeable and trained employees can cause serious damage to the company's **progress and performance** in the market

How can we predict the employee attrition?

- Collect the past employee data and other data related to the impacting factors like Salary, Total Business Value, Quarterly Rating, etc.
- Apply Pre-Processing and feature engineering techniques
    - Understanding Data, fields involved, Visualize and Impute Missing Values
    - New Feature creation – Promotion, Salary Change
    - Data Normalization and Scaling, Data Reshape, Data Encoding
    - Outlier Analysis and PCA
- Visualization of the data in various plots – Univariate and Bi-variate analysis, SNS/panda, Strip, Swarm, Box, Point, Bar, Count, Joinplot, Pairplot, Line plot, Numerical scatterplot, Heatmap correlation
- Finalize the Model metrics and Measures
- Finalize the Model Tuning Techniques
- Finalize the Model Parameters
- Prepare Datasets for Models

- Apply Models
- Logistic Regression, Decision Tree, KNN, Naïve Bayes, Random Forest,
    - Decision Tree Models
    - Logistic Regression Models
    - K Neighbors Classifier Models
    - Bernoulli Naïve Bayes Models
    - Gaussian Naïve Bayes Models
    - Random Forest Models
    - Gradient Boosting Models
    - AdaBoost Models
    - Voting Classifier Models
- Analyze Model Metrics

Key Words:	Employee attrition prediction, PCA, Feature Importance, Outlier Detection, DBCSAN, LOF, Imputation, Logistic Regression, SVM, Ensemble, KNN, AdaBoost, Random Forest, Decision Tree, Naïve Bayes

List of Abbreviations used

Abbreviation	Meaning/Usage
AUC	Area Under Curve
CRISP-DM	Cross Industry Standard Process for Data Mining
CV	Cross Validation
DBSCAN	Density Based Spatial Clustering of Applications with Noise
DT	Decision Tree
GBM	Gradient Boosting Model
Inv_rnk	Inverse Rank
KNN	K Nearest Neighbor
LOF	Local Outlier Factor
LR	Logistic Regression
MAE	Mean Absolute Error
ME	Mean Error
MSE	Mean Squared Error
PCA	Principal Component Analysis
RF	Random Forest
RFE	Recursive Feature Elimination
RFECV	Recursive Feature Elimination with Cross Validation
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
RSCV	Randomized Search Cross Validation

Table 1:List of Abbreviations

List of Figures

Contents

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

Employees are the pillars of organization and workforce retention is one of the key factor in measuring the **organizational growth**. But swelling attrition levels across the businesses are strait-jacketing the distraught HR practitioner's ability to come out with good retention strategies. Employee attrition must be **decreased** for a firm as it increases the high training cost and the crucial business time of an organization.

2. Objective of the project

After proving its mettle in sales and marketing, **artificial intelligence** is also becoming central to employee-related decisions within HR management.
Therefore, the objective is to build a model to **predict** employee attrition which could help human resource department **strategize hiring** and **retention** process.

- The objective is also to thoroughly understand the data in hand and derive business insights, come up with **recommendations** for better employee retention and build a **Predictive model** that would help predict the required information in an efficient manner.

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

Predictive Attrition Model: Using Analytics to predict Employee Attrition:
:https://analyticsindiamag.com/predictive-attrition-model

Employee Attrition Prediction – A Comprehensive Guide
https://zdataset.com/learn-ml/employee-attrition-prediction-a-comprehensive-guide

Cotton, J.L. and Tuttle, J.M., 1986. "Employee turnover: A metaanalysis and review with implications for research" Academy of management review, pp.55-70.

Liu, D., Mitchell, T.R., Lee, T.W., Holtom, B.C. and Hinkin, T.R., 2012. "When employees are out of step with coworkers: How job satisfaction trajectory and dispersion influence individual-and unit-level voluntary turnover". Academy of Management Journal, pp.1360-1380.

Employee attrition refers to the gradual loss of employees over time. Most literature on employee attrition categorizes it as either voluntary or involuntary. Involuntary attrition is thought of as the mistake of the employee and refers to the organization firing the employee for various reasons. Voluntary attrition is when the employee leaves the organization by his own will. This paper focuses on voluntary attrition. A meta-analytic review of voluntary attrition (Predictive Attrition Model: Using Analytics to predict Employee Attrition, n.d.) found that the strongest predictors of voluntary attrition included age, pay, and job satisfaction. Other studies showed that several other features, such as working conditions, job satisfaction, and growth potential also contributed to voluntary attrition (Cotton & Tuttle). Organizations try to prevent employee attrition by using machine learning algorithms to predict the risk of an employee leaving, and then take pro-active steps for preventing such an incident.

4. Machine Learning process flow (Consolidated Approach / Solution Architecture)

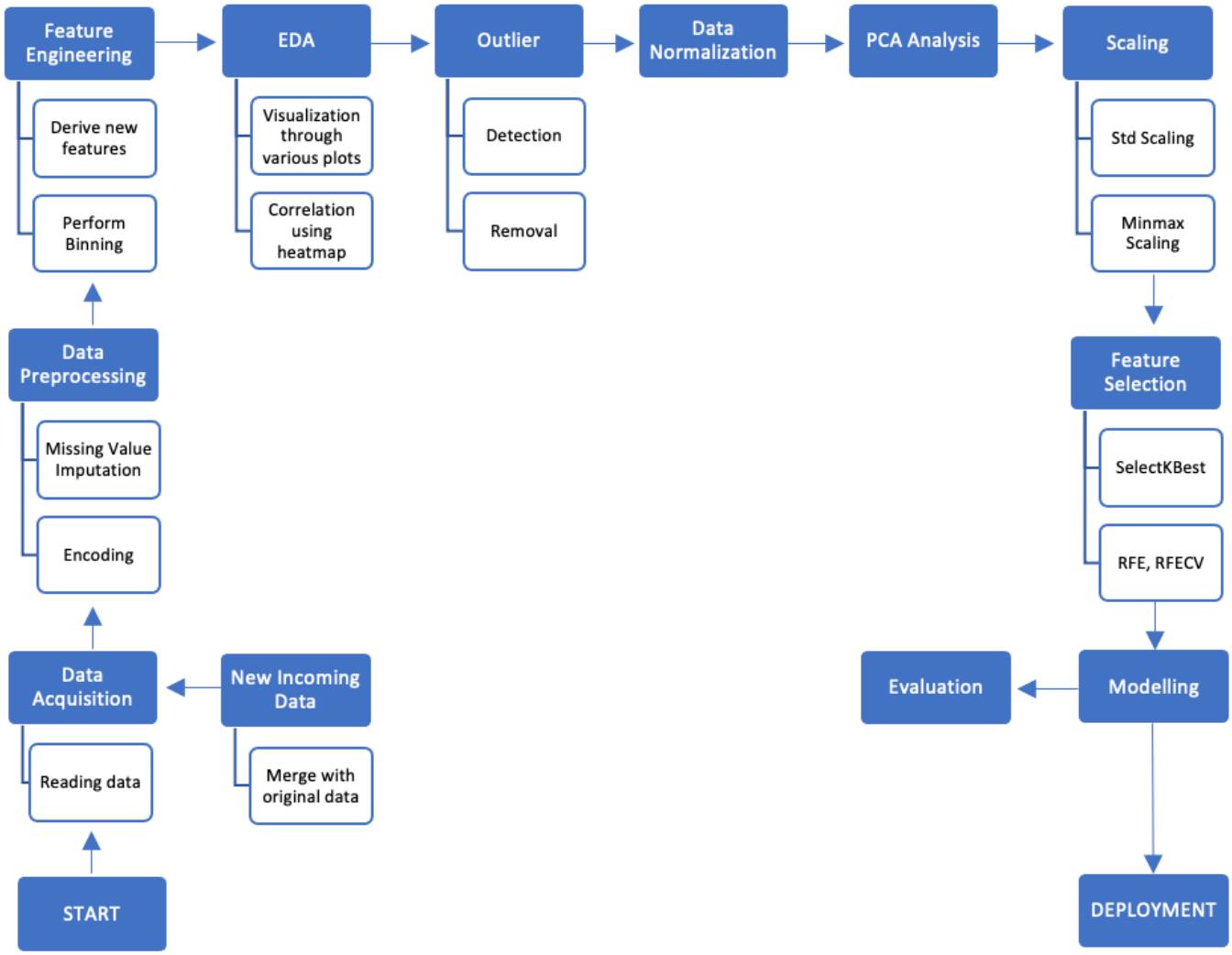


Figure 1: Process workflow

Pre-Processing	Feature Engineering	Visualization	Modeling
•Data understanding •Basic statistics •Merging of new dataset •Data clean -up •Understanding patterns •Data categorisation •Extraction of hidden information •Outliers detection for single variate •Outlier detection for multivariate using DBSCAN •Outlier correction.	•Data derivation like resignation, promotion, retention years, business value and rating trend •Categoric to Numerical conversion •Data binning •Missing data imputation using MICE technique •Normalisation of data using std scalar •PCA analysis	•Univariate ad multivariate plots •Swam plot •Strip plot •Box plot and hist plot •Count plots •Pair plots with scatter and KDE plots •Estimate plot •Joint plot •Correlation plots •Heat maps •Pie chart	•Model preparation •Normalisation •Standardisation •Feature ranking and recursive future elimination •Classification model techniques •Hyper parameter tuning. •Model evaluation •Model prediction •Validation.

Figure 2: Detailed Workflow

5. Classes and functions

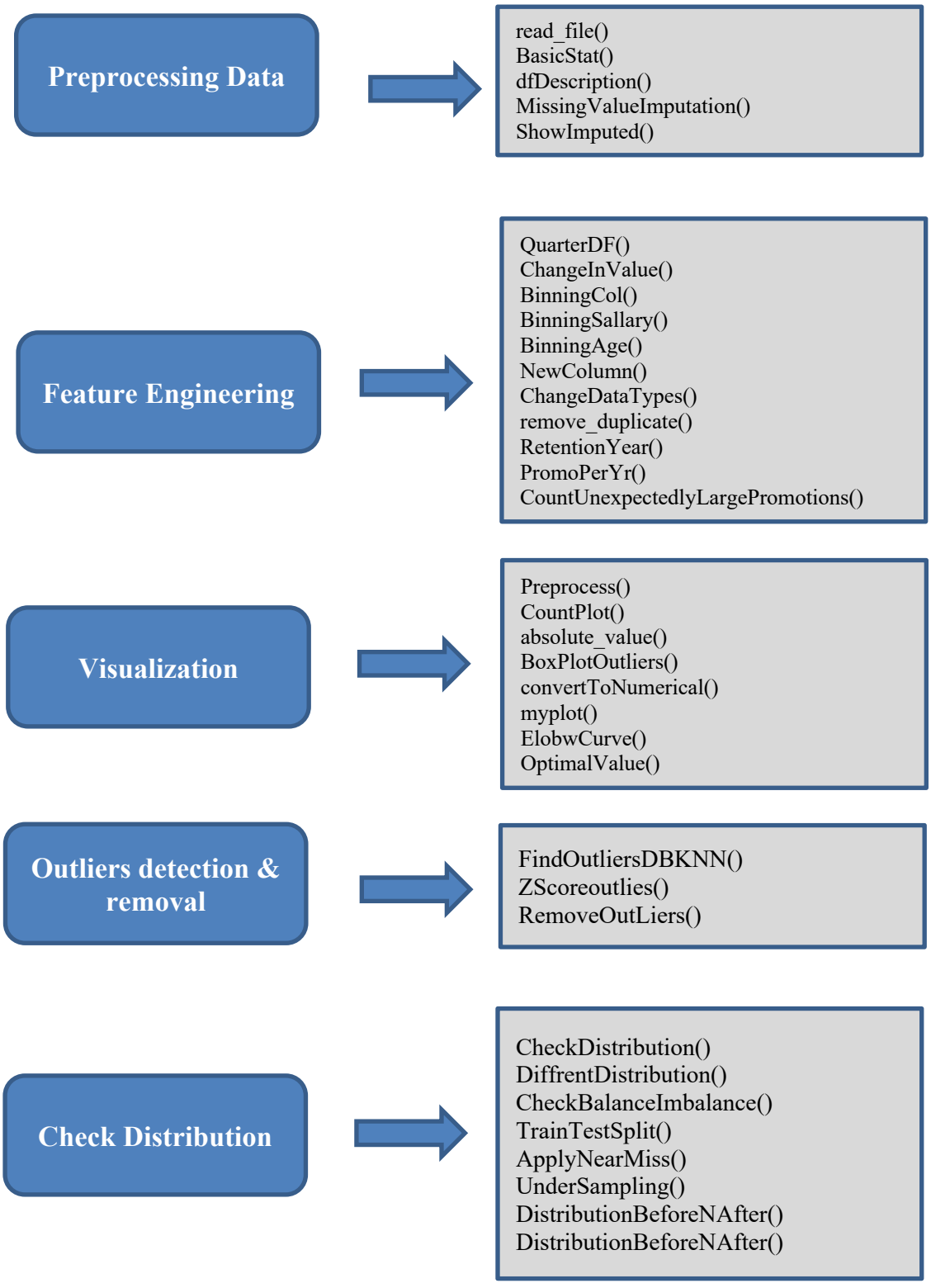
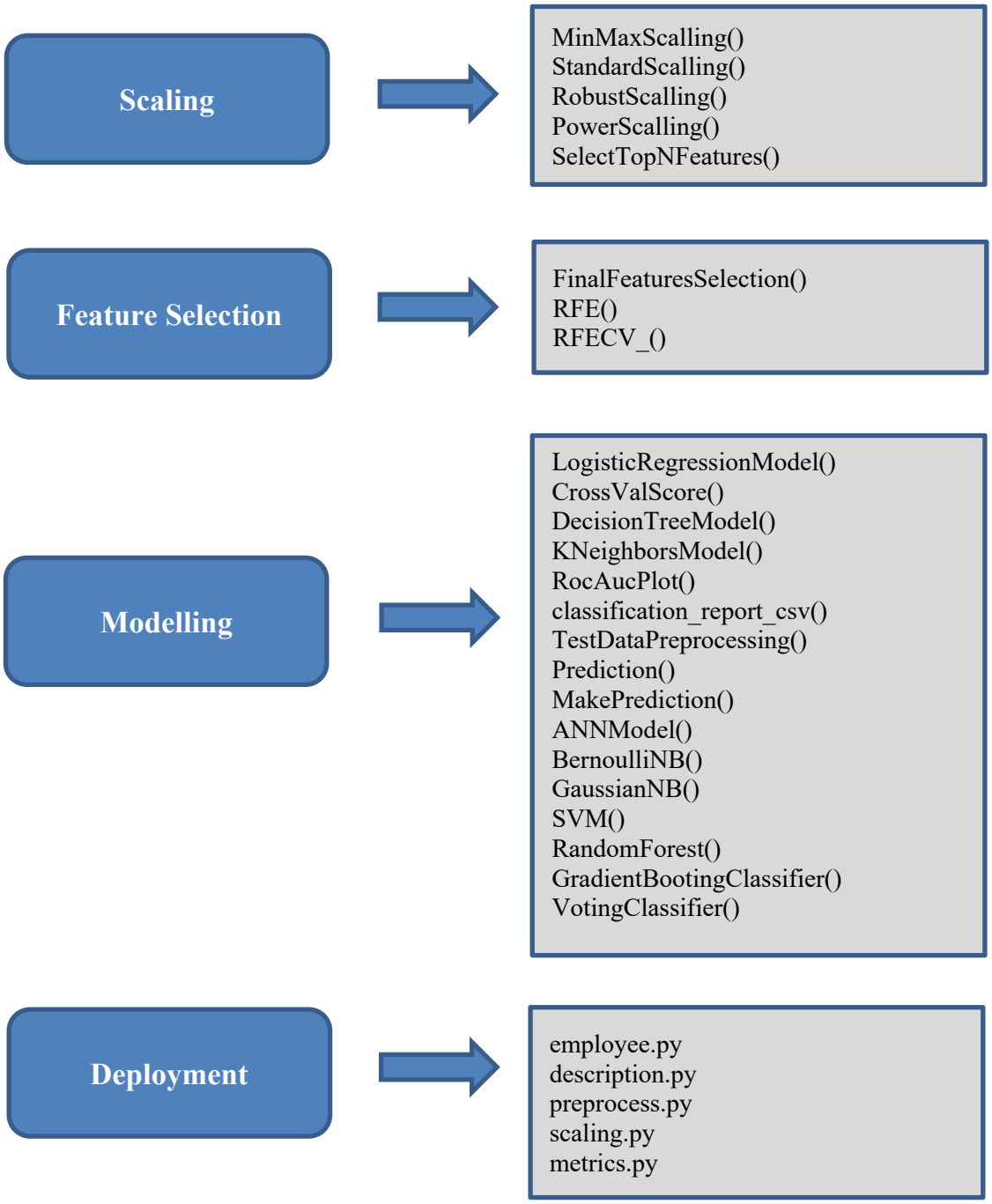


Figure 3: Classes and functions


Scaling
MinMaxScalling()
StandardScalling()
RobustScalling()
PowerScalling()
SelectTopNFeatures()
Feature Selection
FinalFeaturesSelection()
RFE()
RFECV_()
Modelling
LogisticRegressionModel()
CrossValScore()
DecisionTreeModel()
KNeighborsModel()
RocAucPlot()
classification_report_csv()
TestDataPreprocessing()
Prediction()
MakePrediction()
ANNModel()
BernoulliNB()
GaussianNB()
SVM()
RandomForest()
GradientBootingClassifier()
VotingClassifier()
Deployment
employee.py
description.py
preprocess.py
scaling.py
metrics.py

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

6.1.Information

6.1.1. 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

6.1.2. Business knowledge of what all could impact the in count of employee attrition into the organization atvarious times and the degree of impact.

6.1.3. Knowledge of various tools and machine learning techniques that can be applied tobe 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

6.2.Technology

6.2.1. Data storage at a common location accessible to all the team members

6.2.2. Python run time environments – Google Colab /Jupyter/Spyder

6.3.Process

6.3.1. Business Understanding

6.3.2. Project Planning and Execution

6.3.3. Resource management

6.3.3.1. Version Management of the code

6.3.3.2. Work distribution and collaboration

6.3.4. Data Collection

6.3.5. Data Pre-processing

6.3.5.1. Data Acquisition

6.3.5.2. Data Normalization

6.3.5.3. Outlier Analysis

6.3.5.4. Missing Values Imputation

6.3.6. Data Inference

6.3.6.1. Uni Variate Analysis

6.3.6.2. Multi Variate Analysis

6.3.6.3. Feature Creation

6.3.6.4. Data Correlation Analysis

6.4.People

6.4.1. A guide to direct the team in right direction so that learnt knowledge can be appliedin an efficient

6.4.2. 4-5 People with good understanding of Machine learning concepts and Python hands-on experience – preferably with Object Oriented Programming experience

- Python Developer – Developer who codes in Python
- DBA – Database Analyst
- Tester – One who tests the functionality
- Data Scientist – One who is well verse with the Data Science algori

7. Potential data challenges & risks in doing the project

6.1. Data Challenges

- Last Date of Working vs Date of Joining
  - Ideally the Last date of Working for an Employee should be later than the Date of Joining
  - Found around 70 instances where Date of Joining is later than Last Date of Working
  - How to deal?
    - Assumption: The 70 cases correspond to the Employees who have re-joined the organization after resigning

- Joining Designation vs Latest Designation
  - Ideally, we know that employees may get a 0 or 1 promotion per year
  - In the given that dataset, there are 44 cases where employees have got more than 1 promotion per year
  - A maximum of 50 promotions per year is observed

6.2. Risks

Intermediate data set of 2016-2017 and for only limited 2371 employees are available out of which 67% have already left the organization.

Quarterly rating not available for all the quarters.

8. Detailed Plan of Work

	Week 1 and 2	Week 3 and 4
Pre-Processing and Feature Engineering		
Understanding Data and the fields involved		
Data Acquisition		
Unique value understanding		
Checking and Impute Missing Values		
Hidden Data Extraction		
Data description		
Data Encoding		
Outlier Analysis		
Project Documentation		
Exploratory Data Analysis		
New feature extraction		
Binning		
Visualization of missing values		
One hot encoding, target encoding and label encoding		
Data anomalies understanding, outlier corrections		
Visualization		
SNS/Univariate/Bi-variate/panda		
Categorical scatter plot with hue		
Strip		
Swarm		
Categorical distribution plots		
Box		
Pie		
Bar		
Categorical estimate plots		
Point		
Bar		
Count		
Joinplot		
Displot		
Pairplot		
KDEplot		
Line plot		
Numerical scatterplot		
Heatmap Correlation		
Descriptive Statistics		
No of people who left based on Hike%		
No of people who left based on Quarterly Rating		
No of people who left based on Promotion		
No of people who left based on Designation		
No of people who left based on Total Business Value		
Correlation		
PCA		

Figure 4: Execution Plan week 1 to 4

Prepare for Modelling	Week 5	Week 6	Week 7 and 8
Finalize the Metrics and Measures			
Finalize the Model Tuning Techniques, Normalisation and Test train split			
Finalize the Model Parameters			
Prepare Datasets for Models			
Apply Models			
Models- KNN, NB, DT, LR			
Model evaluation and hyper parameter tuning			
Model predictions			
Additional Models			
Random forest, Ensemble techniques			
Gradient Booting, AdaBoost, Voting			
Recursive elimination, Pruning, Cross validations			
Model performance analysis & metrics			
Summary and recommendations			
Project Documentation			

Figure 5: Execution Plan week 6 to 8

9. Pre-Processing steps

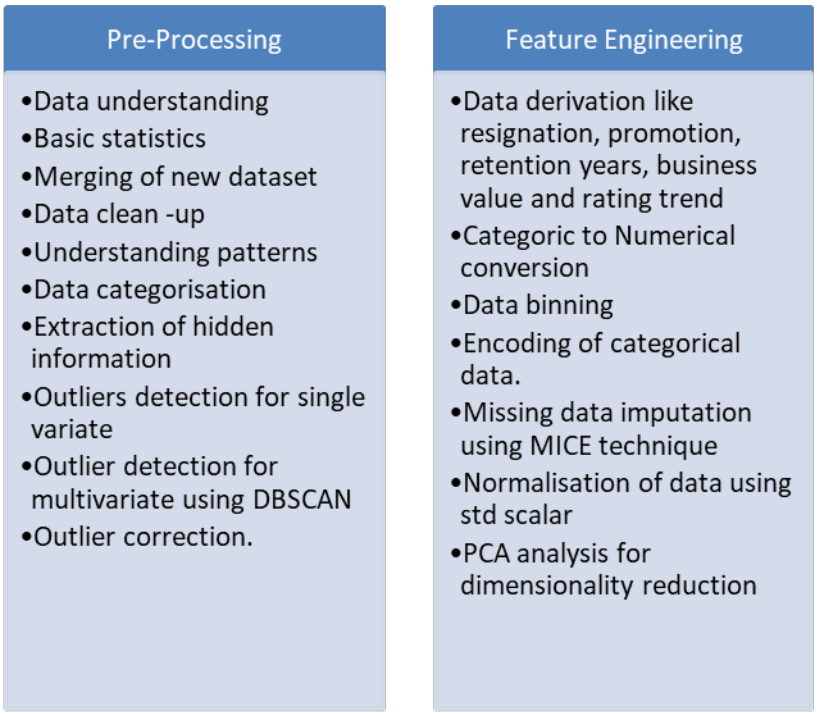


Figure 6: Pre-processing and Feature Engineering flow

9.1 Data Understanding

Reading data set: Given employee data set is in CSV format. Function is created to read data set of any file format like csv, .xlsx , txt or .json.

```
In [162]:
#function to read the file
def read_file(path,file_name):
    import glob
    import os
    file_patteren=file_name
    print(f'File: {file_patteren}')
    files = glob.glob(path +file_patteren, recursive = True)
    if(len(files)==0):
        print('File does not exits')
        return 0
    else:
        for file in files:
            file_text,file_extension=os.path.splitext(file)
            if(file_extension=='.csv'):
                print('Loading CSV file...')
                df=pd.read_csv(file)
            elif(file_extension=='.xlsx'):
                print('Loading excel file...')
                df=pd.read_excel(file)
            elif(file_extension=='.txt'):
                print('Loading text file')
                df=pd.read_csv(file)
            elif(file_extension=='.json'):
                print('Loading the JSON file')
                df=pd.read_json(file)
            else:
                df=pd.read_csv(file)
    return df
```

Figure 7: Data Understanding

Also before reading the dataset, function checks if there is new data frame df1 available in order to combine the new data with the test data set.

Reading the new incoming data and merging with the original one

```
df1=read_file('./','test.csv')
```

```
File: test.csv
File does not exits
```

```
if(df1!=0):
    frames = [df, df1]
    result = pd.concat(frames)
    df = result.sort_values(by=['EmpID', 'MMM-YY'], ascending=True)
    df.to_csv('train.csv')

    file = 'test.csv'
    if(os.path.exists(file) and os.path.isfile(file)):
        os.remove(file)
        print("file deleted")
    else:
        print("file not found")
```

Figure 8: Merging data

Given employee dataset has 13 attributes as follows:

```
#reading the file
df=read_file('./','train.csv')
df.head()
```

```
File: train.csv
Loading CSV file...
```

MMM-YY	EmpID	Age	Gender	City	EducationLevel	Salary	DateofJoining	LastWorkingDate	JoiningDesignation	Designation	TotalBusinessValue	QuarterlyRating
01-01-2016	1	28	Male	C23	Master	57387	24-12-2015	NaN	1	1	2381060	2
01-02-2016	1	28	Male	C23	Master	57387	24-12-2015	NaN	1	1	-665480	2
01-03-2016	1	28	Male	C23	Master	57387	24-12-2015	11-03-2016	1	1	0	2
01-11-2017	2	31	Male	C7	Master	67016	06-11-2017	NaN	2	2	0	1

Figure 9: Code Snippet - Reading the data to the Data Frame

Here are the details of 13 attributes in the given data set.

MMMM - YY

- Reporting Date (Monthly) – When the employee data is updated

Emp_ID

- A unique ID that helps to identify each employee distinctly

Age

- Age of the employee on the Reporting Date

Gender

- Gender of the Employee

City

- City code of the Employee

Education Level

- Education Level of the Employee : Bachelor / Master / College

Salary

- Salary of the Employee on the Reporting Date

Dateof joining

- The date when the Employee joined the Organization

LastWorkingDate

- Last date of working for the employee –
    - Null for the employees still working in the Organization
    - Last date of working for the employees who left the Organization

Joining Designation

- Designation at which the Employee was hired

Designation

- Designation of the Employee on the Reporting Date

TotalBusinessValue

- Total business value acquired by the Employee in a month –
    - Positive value corresponds to successful sold of insurance policy
    - Negative value corresponds to cancellation / refund of the sold insurance policy
    - Zero value corresponds to the case when the Employee couldn't sell any insurance policy in the given month

Quarterly Rating

- Quarterly rating of the Employee as evaluated based on Employee's performance in the given quarter
    - Possible values: 1, 2, 3, 4
    - Higher value indicates better performance and better rating

9.2 Basic statistics and understanding the data

Data set has no null values except for the "Last working day" column which indicates that people with date on "last working day" has left the organization.

Basic statistics:

```
def BasicStat(dataframe):
    print('--------------Basic Description of Data Frame--------------')
    print()
    #total count
    print('Total records in data frame: {}\nTotal Columns:{}'.format(dataframe.shape[0],dataframe.shape[1]))
    print()
    #missing values
    print('Total Missing value in each Column:\n{}\n'.format(dataframe.isnull().sum()))
    print()
    #duplicate count
    print('Total Duplicate Count:{}'.format(dataframe.duplicated().sum()))
    print()
    #data types
    print('Data Types of each column:\n{}'.format(dataframe.dtypes))
```

Figure 10: Basic statistic function

```
BasicStat(df)
```

```
--------------Basic Description of Data Frame--------------

Total records in data frame: 19104
Total Columns:13

Total Missing value in each Column:
MMM-YY                     0
EmpID                      0
Age                        0
Gender                     0
City                       0
EducationLevel             0
Salary                     0
DateofJoining              0
LastWorkingDate        17488
JoiningDesignation         0
Designation                0
TotalBusinessValue         0
QuarterlyRating            0
dtype: int64


Total Duplicate Count:0

Total Duplicate Count:0

Data Types of each column:
MMM-YY                object
EmpID                  int64
Age                    int64
Gender                object
City                  object
EducationLevel        object
Salary                 int64
DateofJoining         object
LastWorkingDate       object
JoiningDesignation     int64
Designation            int64
TotalBusinessValue     int64
QuarterlyRating        int64
dtype: object
```

Figure 11: Basic description of data frame

Data set has 19104 records and the "Last working Date" column has 17488 missing values.

Data description:

```
def dfDescription(dataframe):
    #Discreption of Data Frame
    print(f'Statistics of Data Frame:\n')
    result=dataframe.describe().apply(lambda s: s.apply('{0:.5f}'.format)).T
    result['median']=dataframe.median()
    return result
```

Figure 12: Data description

```
dfDescription(df)
```

```
Statistics of Data Frame:
```

	count	mean	std	min	25%	50%	75%	max	median
EmpID	19104.00000	1415.59113	810.70532	1.00000	710.00000	1417.00000	2137.00000	2788.00000	1417.0
Age	19104.00000	34.65028	6.26447	21.00000	30.00000	34.00000	39.00000	58.00000	34.0
Salary	19104.00000	65652.02513	30914.51534	10747.00000	42383.00000	60087.00000	83969.00000	188418.00000	60087.0
JoiningDesignation	19104.00000	1.69054	0.83698	1.00000	1.00000	1.00000	2.00000	5.00000	1.0
Designation	19104.00000	2.25267	1.02651	1.00000	1.00000	2.00000	3.00000	5.00000	2.0
TotalBusinessValue	19104.00000	571662.07496	1128312.21846	-6000000.00000	0.00000	250000.00000	699700.00000	33747720.00000	250000.0
QuarterlyRating	19104.00000	2.00890	1.00983	1.00000	1.00000	2.00000	3.00000	4.00000	2.0

Figure 8: Data description

From the above description we could understand that “Age” attribute varies from 21 to 58 years There is huge standard deviation for “salary” and “TotalBusinessValue” attributes. Designation varies from 1 to 5 and the “Quarterly Rating” varies from 1 to 4.

When we try to filter the data set with unique employee ID to find the number of employees we found

- Out of the 19104 records the data has 16733 duplicate entries when filtered with EMP_ID
- After filtering by employee ID we have data of 2371 Employees for analysis

```
df=remove_duplicate(df,'EmpID')
df.head()
```

```
Before Removing the duplicates:18916
Removing the Duplicate
After Removing the duplicates:2371
```

Figure 9:Duplicate data based on “EmpID”

9.3 About dataset

```
In [11]: pd.to_datetime(df['MMM-YY']).min()
Out[11]: Timestamp('2016-01-01 00:00:00')

In [12]: pd.to_datetime(df['MMM-YY']).max()
Out[12]: Timestamp('2017-01-12 00:00:00')

In [13]: pd.to_datetime(df['DateofJoining']).min()
Out[13]: Timestamp('2010-01-04 00:00:00')

In [14]: pd.to_datetime(df['DateofJoining']).max()
Out[14]: Timestamp('2017-12-28 00:00:00')

In [15]: df['EmpID'].min()
Out[15]: 1

In [16]: df['EmpID'].max()
Out[16]: 2788

In [17]: len(df['EmpID'])
Out[17]: 19104

In [50]: len(pre_df[pre_df['PromoPerYr']>1])
Out[50]: 44

In [51]: pre_df['PromoPerYr'].max()
Out[51]: 50.0

In [52]: pre_df['Gender'].value_counts()
Out[52]: Male      1404
         Female     977
         Name: Gender, dtype: int64

In [53]: mask = pre_df[pre_df['MMM-YY'] < '2017-1-1']
         mask['Promotion'].value_counts()
Out[53]: No     684
         Yes    146
         Name: Promotion, dtype: int64

In [54]: mask = pre_df[pre_df['MMM-YY'] >= '2017-1-1']
         mask['Promotion'].value_counts()
Out[54]: No     1290
         Yes     261
         Name: Promotion, dtype: int64
```

- Data set has 2016 and 2017 employee data (intermediate data set)
- We could also see designation and salary has changed only for 43 employees in 1 year data set given for employees (1 appraisal cycle data)
- Quarterly rating is available for each employee (maximum of 8 quarterly rating for employee who is in system for 2016 and 2017 full year)
- Emp ID is from 1 to 2788, but the data available only for 2371 employees (417 employees have left before 2016)
- Out of total Employees , the percentage of Female Employees is 41% and that of Male Employees is 59%

Figure 11: Aggregated Data Snapshot

9.4 Preprocessing:

Function is created to have preprocessing to convert categorical column for Gender, Education level, quarterly rating and resignation. Also the date related attributes are converted to date , month, year for extraction of useful information.

On the numerical columns like salary, designation , Total business value and quarterly rating analysis has been done by using below code snipped to understand the changes and trend

happening per employee to see the overall impact during 2016-2017

```
# Single function that include all the function to process the file
def Preprocess(path,file):

    # read the file
    df=read_file(path,file)
    print()

    # Change in value
    columns=['Salary','Designation','TotalBusinessValue','QuarterlyRating']
    for c in columns:
        df=ChangeInValue(df,c)
    print()

    # Adding columns
    df=NewColumn(df)
    print()

    #changing the data types into category
    cat_column=['Gender','EducationLevel','QuarterlyRating','Resign']
    df=ChangeDataTypes(df,cat_column,'category')
    print()

    ##changing the data types into date
    column=['MMM-YY','DateofJoining','LastWorkingDate']
    for col in column:
        df=ChangeDataTypes(df,col,'date')
    print()

     # Calling function to Extract year,month and quarter
#      df=ExtractDate(df)
#      print()


     # Remove the duplicate
     df=remove_duplicate(df,'EmpID')
     print()

      #Add the retention year column
     df=RetentionYear(df)

     #Add the PromoPerYr year column
     df=PromoPerYr(df)

     count = CountUnexpectedlyLargePromotions(df)

     print('Max no. of promotions per year: ',df['PromoPerYr'].max())

     cat_column=['JoiningDesignation','Designation',]
     df=ChangeDataTypes(df,cat_column,'category')
     print()

     df['YearsOfExperience'] = round(df['Retention_years'])

     df.drop(['bins_Age', 'bins_Salary'],axis=1,inplace=True)

     return df
```

Figure 13: Preprocessing

9.5 Data quality issues

```
pre_df=Preprocess('../','train.csv')
pre_df.head()

File: train.csv
Loading CSV file...

Change in Value Column added..
Change in Value Column added..
Change in Value Column added..
Change in Value Column added..

Resign,Hike and Promotion column added,..
intervals in  Salary
: [(10569.329, 69970.667], (69970.667, 129194.333], (129194.333, 188418.0]]
Categories (3, interval[float64]): [(10569.329, 69970.667] < (69970.667, 129194.333] < (129194.333, 188418.0]]
BinnedSalary column added,..
intervals in  Age
: [(20.963, 33.333], (33.333, 45.667], (45.667, 58.0]]
Categories (3, interval[float64]): [(20.963, 33.333] < (33.333, 45.667] < (45.667, 58.0]]
BinnedAge column added,..

Data types have been changed to category..

Data types have been changed to date..
Data types have been changed to date..
Data types have been changed to date..

Before Removing the duplicates:19104
Removing the Duplicate
After Removing the duplicates:2381

No. of Cases where Date of Joining is later than Last Working Date: 71

No. of cases where employees have got more than 1 promotion per year:  44

Max no. of promotions per year:  50.0

Data types have been changed to category..
```

- **Last Date of Working vs Date of Joining**

  Ideally the Last date of Working for an Employee should be later than the Date of Joining

  Found around 71 instances where Date of Joining is later than Last Date of Working

  How to deal?

  Assumption: The 71 cases corresponds to the Employees who have re joined the organisation after resigning

- **Joining Designation vs Latest Designation**

  Ideally, we know that employees may get a **0 or 1 promotion per year**

  In the given that dataset, there are 44 cases where employees have got more than 1 promotion per year

  In the given dataset we noticed 50 promotions cases during 2016-2017 out of 2371 employees.(2%)

Figure 14: Data quality issues

9.6 Data Extraction

Following data are extracted from the given dataset by using the attached code snippet

1. **Resign (Target Variable)**: Derived from the Last Working Date
2. **Retention Years**: Derived from features – Date of Joining, Last Working Date and Latest Reporting Date
3. **Promotion**: Derived from the difference in the Joining Designation and the Designation at the latest Reporting Date
4. **Hike%:** Derived from the salary changes across all the reporting dates corresponding to each Employee
5. For each employee it is possible to derive the **change in designation** and salary in 1 appraisal cycle by comparing 2016 and 2017 data (around 12 months from date of joining)
6. **Quarter:** As quarterly rating is changing for each employee, we have created 8 quarters per employee (4 quarters in 2016 and 4 quarters in 2017) to understand the rating change trend.
7. Total business value also varies with each quarter. We have extracted the change in total business value for each employees to understand their **sales performance**.
8. New features added ***Salary_change, Designation_change, TotalBusinessValue_change, QuarterlyRating_change,***

9. ***Year & Month columns*** are derived corresponding to the Date columns
10. **BinnedSalary and BinnedAge** columns are derived from binning the Salary and Age columns.

Creating New relevant Features from Existing Features

```
def NewColumn(dataframe):
    # adding target column
    dataframe['Resign']=np.where(dataframe['LastWorkingDate'].isnull()==True,0,1)

    # Hike Column
    dataframe['SalaryChangeAmount']=dataframe.groupby(['EmpID'])['Salary'].transform(lambda y:y-y.shift(1)).apply(lambda x: x
    dataframe['Hike%']=(dataframe['SalaryChangeAmount'])*100//dataframe['Salary']
    dataframe.drop('SalaryChangeAmount',axis=1,inplace=True)

   # Promotion Column (Yes/No)
    dataframe['Promotion']=dataframe['Designation']-dataframe['JoiningDesignation']

    # Replacing Promotion values to Yes and No
    dataframe['Promotion']=dataframe['Promotion'].replace(to_replace =[1,2,3,4], value ="Yes").replace(to_replace =[0], value
    print('Resign,Hike and Promotion column added,..')

    # Sallary Binning
    dataframe['BinnedSalary']= BinningCol(dataframe, 'Salary') #applying the function
    print('BinnedSalary column added,..')

    # Age binning
    dataframe['BinnedAge']= BinningCol(dataframe, 'Age') #applying the function
    print('BinnedAge column added,..')


    return dataframe
```

Figure 15:Creating New Features

Adding Feature for Quarters to see the trend in each Quarter for 2016 and 2017

```
# new dataframe to see the trend in each quarter for 2016 and 2017
def QuarterDF():
    df['MMM-YY']=pd.to_datetime(df['MMM-YY'])
    #making a seperate dataframe by unique employee id and Q1---Q8 empty column
    emp=df['EmpID'].unique().tolist()
    quarter_df=pd.DataFrame(emp)
    quarter_df.columns=['EmpID']
    q=['Q1','Q2','Q3','Q4','Q5','Q6','Q7','Q8']
    for i in q:
        quarter_df[i]= np.nan # filling with 0 instead of NaN (np.nan)

    # make a seperate dataframe for each employee and take required index value for that emplyoyee
    # iterate through unique year and get the Quarter
    # for 2016 quarter value will go from Q1 to Q4 and for 2017 Q5 to Q8
    for e in emp:
        IDF=df[df['EmpID']==e]
        idx=quarter_df[quarter_df['EmpID']==e].index.tolist()[0]
        for index,y in enumerate(IDF['MMM-YY'].dt.year.tolist()):
            if(y==2016):
                quarter_numer=pd.to_datetime(IDF['MMM-YY'].iloc[index]).quarter
                if(quarter_numer==1):
                    quarter_df['Q1'].loc[idx]=IDF['QuarterlyRating'].iloc[index]
                elif(quarter_numer==2):
                    quarter_df['Q2'].loc[idx]=IDF['QuarterlyRating'].iloc[index]
                elif(quarter_numer==3):
                    quarter_df['Q3'].loc[idx]=IDF['QuarterlyRating'].iloc[index]
                else:
                    quarter_df['Q4'].loc[idx]=IDF['QuarterlyRating'].iloc[index]
            elif(y==2017):
                quarter_numer=pd.to_datetime(IDF['MMM-YY'].iloc[index]).quarter
                if(quarter_numer==1):
                    quarter_df['Q5'].loc[idx]=IDF['QuarterlyRating'].iloc[index]
                elif(quarter_numer==2):
                    quarter_df['Q6'].loc[idx]=IDF['QuarterlyRating'].iloc[index]
                elif(quarter_numer==3):
                    quarter_df['Q7'].loc[idx]=IDF['QuarterlyRating'].iloc[index]
                else:
                    quarter_df['Q8'].loc[idx]=IDF['QuarterlyRating'].iloc[index]
            else:
                pass
    return quarter_df
```

Figure 16:Adding Features

Below function is added to remove any duplicate items based on the column name.

```
def remove_duplicate(dataframe,column_name):
    #Aggregating the TotalBusinessValue before removing the duplicates
    total_tvb=dataframe[['EmpID','TotalBusinessValue']].groupby('EmpID').sum()
    total_tvb.reset_index(drop = True,inplace = True)

    #Cumulative sum of TotalBusinessValue_change before removing the duplicates
    # total_tvbc=dataframe[['EmpID','TotalBusinessValue_change']].groupby('EmpID').sum()
    # total_tvbc.reset_index(drop = True,inplace = True)

    #remove duplicate
    print(f'Before Removing the duplicates:{len(dataframe)}')
    print('Removing the Duplicate')
    dataframe=dataframe.drop_duplicates([column_name],keep='last')
    dataframe.reset_index(drop = True,inplace = True)
    print(f'After Removing the duplicates:{len(dataframe)}')

    #replacing the TotalBusinessValue value by Aggrgating values
    dataframe['TotalBusinessValue']=total_tvb['TotalBusinessValue']
    #dataframe['TotalBusinessValue_change']=total_tvbc['TotalBusinessValue_change']
    return dataframe
```

Figure 17:Remove duplicate items based on the column name

Extraction of retention years from date of joining and last working day

```
def RetentionYear(dataframe):
    cnt = 0
    dataframe['DateofJoining'] = pd.to_datetime(dataframe['DateofJoining'])
    dataframe['LastWorkingDate'] = pd.to_datetime(dataframe['LastWorkingDate'])
    dataframe['MMM-YY'] = pd.to_datetime(dataframe['MMM-YY'])
    for j, val in enumerate(dataframe.columns.values):
        k = j
    dataframe.insert(k+1, "Retention_years", np.nan)
    for i in range(0,len(dataframe['LastWorkingDate'])):
        f1 = pd.isnull(df['LastWorkingDate'].loc[i])
        if(f1 == False):
            dataframe["Retention_years"].loc[i]=(dataframe['LastWorkingDate'].loc[i]- dataframe['DateofJoining'].loc[i])/np.t
            if(pd.isnull(dataframe["Retention_years"].loc[i])):
                dataframe["Retention_years"].loc[i]=(dataframe["MMM-YY"].loc[i]-dataframe["DateofJoining"].loc[i])/np.timedel
        else:
            dataframe["Retention_years"].loc[i]=(dataframe["MMM-YY"].loc[i]-dataframe["DateofJoining"].loc[i])/np.timedelta64
            if(dataframe["MMM-YY"].loc[i] == dataframe["DateofJoining"].loc[i]):
                dataframe["Retention_years"].loc[i] = 0

        if(dataframe["Retention_years"].loc[i] < 0):
            cnt = cnt + 1
            dataframe["Retention_years"].loc[i] = 0

    print("No. of Cases where Date of Joining is later than Last Working Date:",cnt)
    dataframe["Retention_years"]=dataframe['Retention_years'].round(decimals=2)
    return dataframe
```

```
def PromoPerYr(dataframe):
    for j, val in enumerate(dataframe.columns.values):
        k = j
    dataframe.insert(k+1, "PromoPerYr", np.nan)
    for i in range(0,len(dataframe['Retention_years'])):
        if(dataframe['Retention_years'].loc[i] != 0):
            dataframe['PromoPerYr'].loc[i] = (dataframe['Designation'].loc[i] -dataframe['JoiningDesignation'].loc[i])
            /dataframe['Retention_years'].loc[i]

    return dataframe
```

Figure 18:Extraction of retention years

```
# Single function that include all the function to process the file
def Preprocess(path,file):

    # read the file
    df=read_file(path,file)
    print()

    # Change in value
    columns=['Salary','Designation','TotalBusinessValue','QuarterlyRating']
    for c in columns:
        df=ChangeInValue(df,c)
    print()

    # Adding columns
    df=NewColumn(df)
    print()

    #changing the data types into category
    cat_column=['Gender','EducationLevel','QuarterlyRating','Resign']
    df=ChangeDataTypes(df,cat_column,'category')
    print()

    ##changing the data types into date
    column=['MMM-YY','DateofJoining','LastWorkingDate']
    for col in column:
        df=ChangeDataTypes(df,col,'date')
    print()

    # Calling function to Extract year,month and quarter
#     df=ExtractDate(df)
#     print()


    # Remove the duplicate
    df=remove_duplicate(df,'EmpID')
    print()

     #Add the retention year column
    df=RetentionYear(df)

    #Add the PromoPerYr year column
    df=PromoPerYr(df)

    count = CountUnexpectedlyLargePromotions(df)

    print('Max no. of promotions per year: ',df['PromoPerYr'].max())

    cat_column=['JoiningDesignation','Designation',]
    df=ChangeDataTypes(df,cat_column,'category')
    print()

    df['YearsOfExperience'] = round(df['Retention_years'])

    df.drop(['bins_Age', 'bins_Salary'],axis=1,inplace=True)

    return df
```

Figure 19:Extraction of retention years

```
pre_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381 entries, 0 to 2380
Data columns (total 25 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   MMM-YY                     2381 non-null   datetime64[ns]
 1   EmpID                      2381 non-null   int64
 2   Age                        2381 non-null   int64
 3   Gender                     2381 non-null   category
 4   City                       2381 non-null   object
 5   EducationLevel             2381 non-null   category
 6   Salary                     2381 non-null   int64
 7   DateofJoining              2381 non-null   datetime64[ns]
 8   LastWorkingDate            1616 non-null   datetime64[ns]
 9   JoiningDesignation         2381 non-null   category
 10  Designation                2381 non-null   category
 11  TotalBusinessValue         2381 non-null   int64
 12  QuarterlyRating            2381 non-null   category
 13  Salary_change              2381 non-null   int64
 14  Designation_change         2381 non-null   int64
 15  TotalBusinessValue_change  2381 non-null   int64
 16  QuarterlyRating_change     2381 non-null   int64
 17  Resign                     2381 non-null   category
 18  Hike%                      2381 non-null   float64
 19  Promotion                  2381 non-null   category
 20  BinnedSalary               2381 non-null   category
 21  BinnedAge                  2381 non-null   category
 22  Retention_years            2381 non-null   float64
 23  PromoPerYr                 2304 non-null   float64
 24  YearsOfExperience          2381 non-null   float64
dtypes: category(9), datetime64[ns](3), float64(4), int64(8), object(1)
memory usage: 319.8+ KB
```

Figure 20: Feature information

10. Visualization Plots

Various graphs have been plotted on the given dataset to gain more insights from the data.

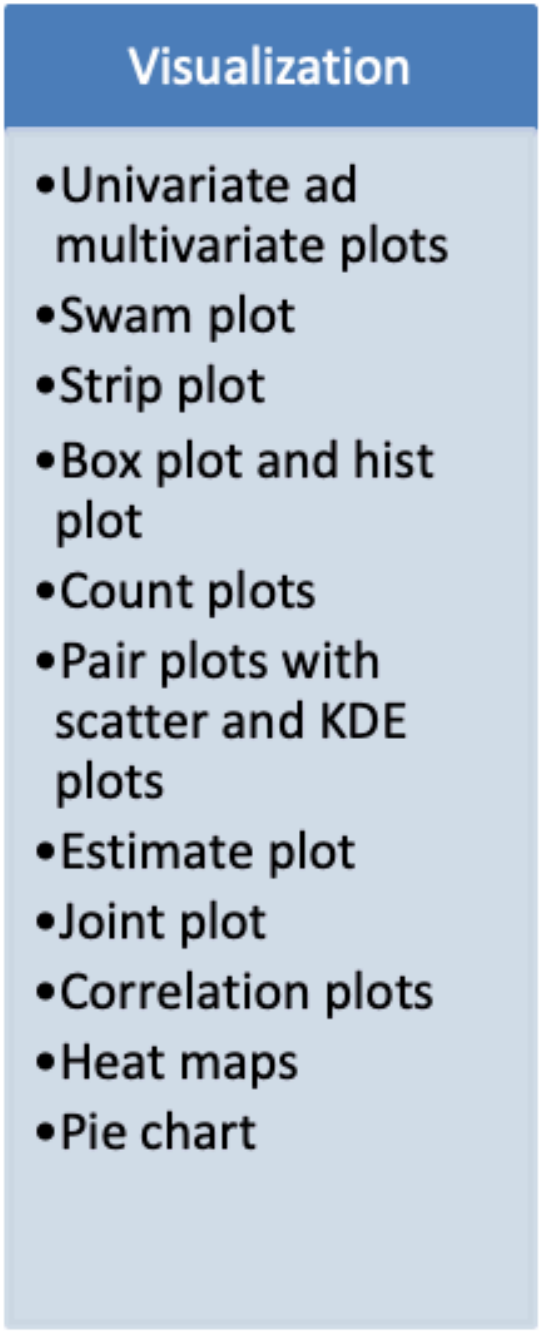
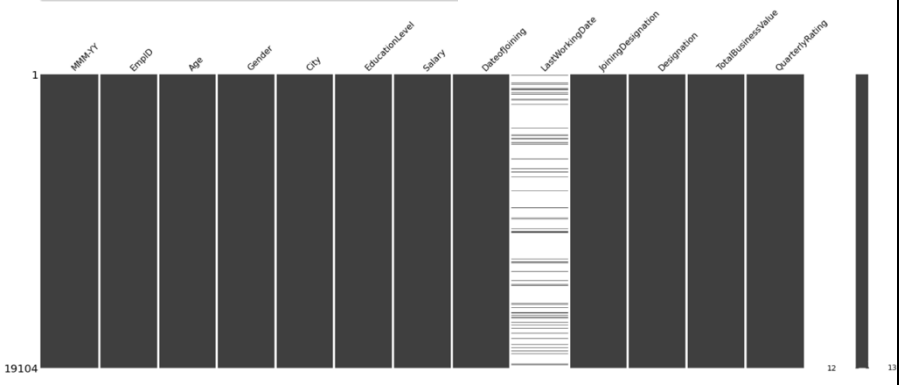
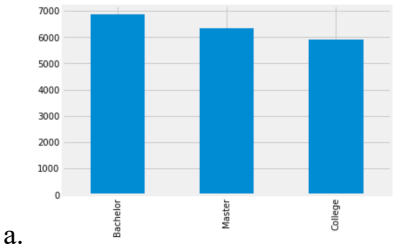


Figure 21: Visualization Plots

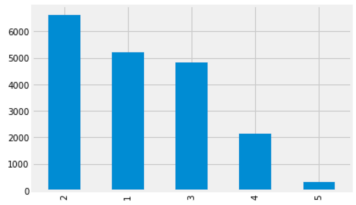
```
#vsualizing Missing value
msno.matrix(df_with_mv)
```



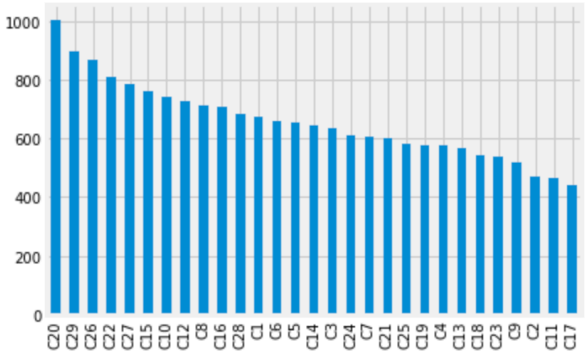
Technique: msno plot
Stage Used: before pre-processing **Purpose**: To find the missing value in the dataset and visualize it
Data: Visualizationdata
Inference:
Missing value is found only on the "Last working date" column which indicates the employees have left the organisation. For Other employees with "Nan" value indicates the employees are staying back in the org.



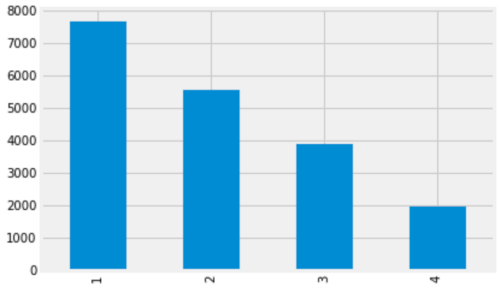
a. Education Level



b. Designation



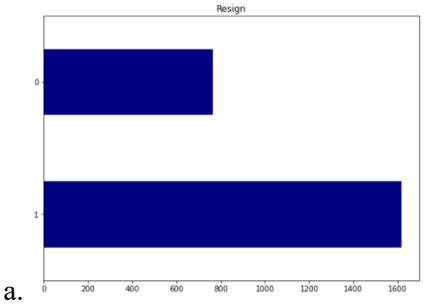
c. City



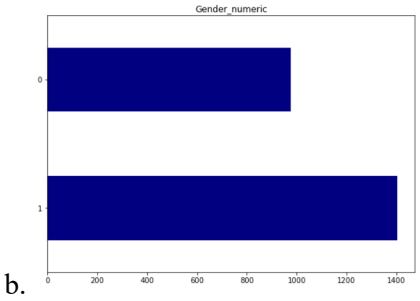
d. Quarterly Rating

Technique: Count-Bar plots for Education level, Designation level and City
Stage Used: before pre-processing
Purpose: To understand the categories
Data: Visualization data
Inference:
Figure a: There are 3 levels of education in the given data set with fairly equal distribution of count
Figure b: Designation varies from 1 to 5 in ordinal format. Designation 2 being the maximum.
Figure c: employees come from 29 different cities with C-20 being the maximum count.
Figure d: There are 4 quaterly ratings varying from 1 to 4 , with '1' being the lowest rating. Also max. employees are with rating '1'.

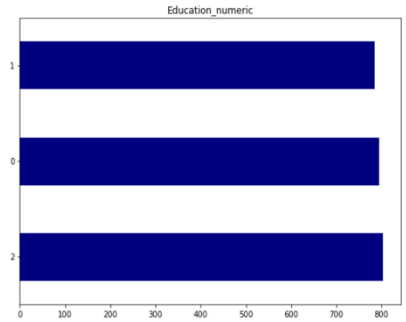
```
# Set the color coding for the visualizations to be plotted
df_bar_plot = df1.drop(labels=['Age','Salary', 'TotalBusinessValue','LastWorkingDate','Retention_years'],
                       axis=1, index=None, columns=None, level=None, inplace=False)

from matplotlib import cm
cmap = cm.get_cmap('jet')
cmap1 = cm.get_cmap('Spectral')

plt.figure(figsize=(20,30))
pos = 1
for i in df_bar_plot.columns:
    plt.subplot(4, 2, pos)
    #sns.boxplot(df[i])
    df_bar_plot[i].value_counts().plot.barh(title= i,cmap=cmap)
    pos += 1
```

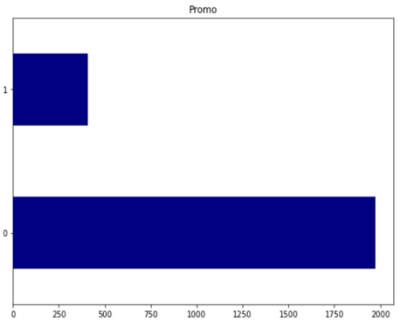


a.

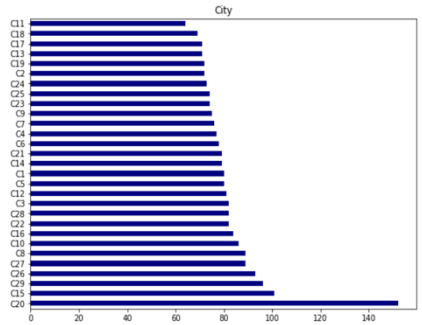


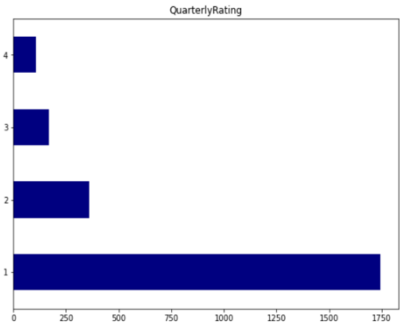
b.



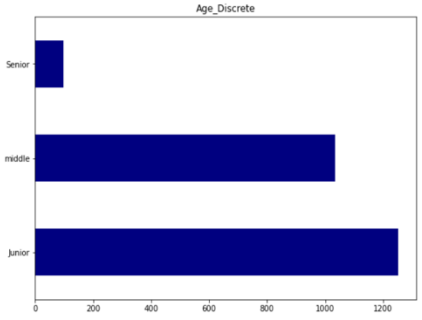


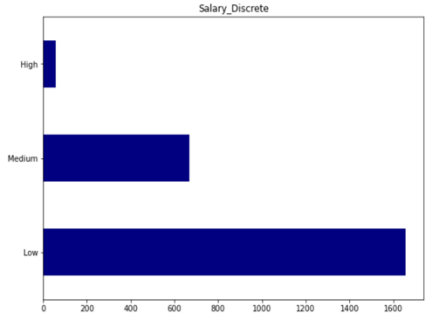
c.





d.





e.

Technique: Bar Plots
plot Stage Used: After dropping duplicates by Emp-ID **Purpose:** To understand the count of each category
Data: Bar visualization
Inference:

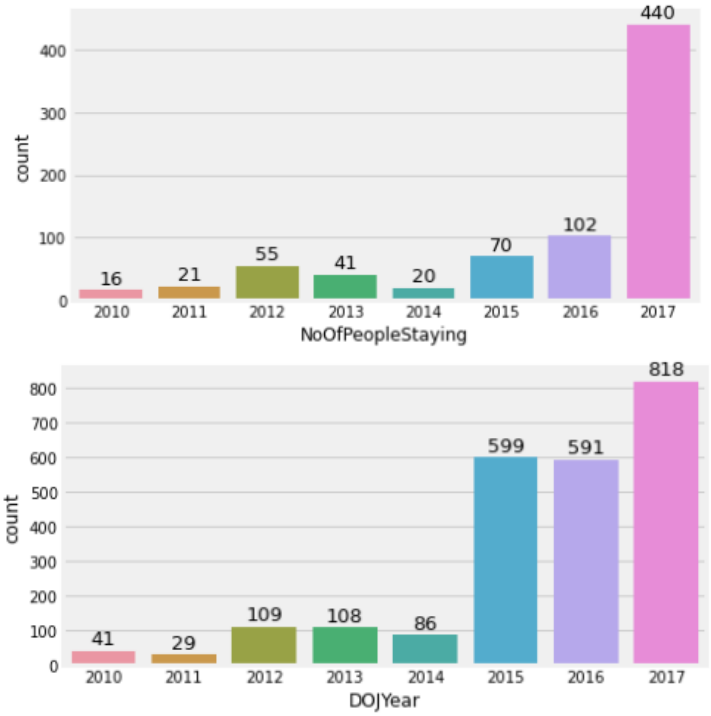
Figure a: From the resignation count we can find 1616 employees have resigned and 765 are staying back with an attrition rate of 67%

Figure b: Gender ratio: Male : female counts stands at 1404 : 977

Figure c: Education-> all three categories of education are well distributed with equal counts. Promotion-> Maximum employees did not receive any promotion

Figure d: City-> people come from various cities out of which City code 'C-20' seems to be the Maximum. Quaterly rating-> Maximum employees are rated with the lowest rating of '1'. This seems to be consistent with 'promo' graph as the reason for no promotion.

Figure e: Age and Salary-> Maximum group of people are from Junior and middle level. This is well correlating with Promotion and salary bands as Juniors receive lesser salary.



a. Date of joining Vs number of people staying in the organisation

Inference:

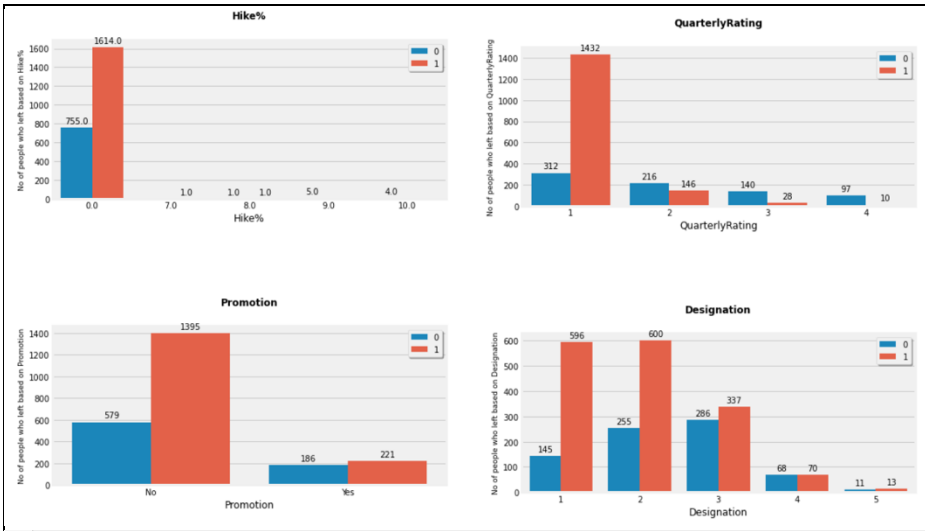
- *Figure a*: By comparing DOJ and no. of people who are in the organisation, we could see huge attrition with people joined in 2015 to 2017.
- This could be a typical scenario in IPO/Call Center organisation where the attrition is so high and employee hardly stay for 1 year.

Bi-variate analysis Plots

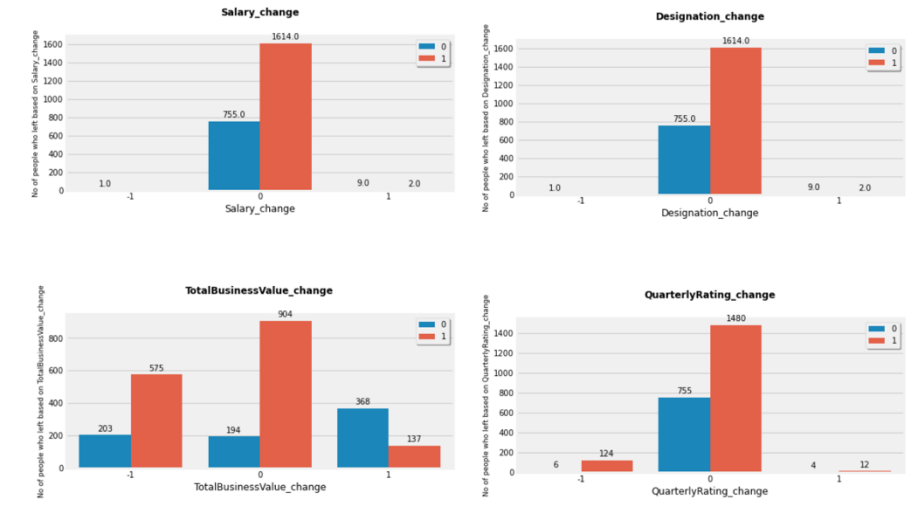
```
# function for count plot
def CountPlot(dataframe,column,target_column):
    count=len(column)
    fig = plt.figure(figsize=(15,10))
    a = round((count/2)+0.5)  # number of rows
    b = 2  # number of columns
    c = 1  # initialize plot counter

    #iterating each columns and creating a count plot
    for i in column:
        plt.subplot(a, b, c)
        plt.title('{}'.format(i),fontsize=12,fontweight='bold',y=1.1)
        plt.xlabel(i)
        plots=sns.countplot(dataframe[i],hue=dataframe[target_column])
        plt.legend(loc='upper right',shadow=True)
        plt.ylabel(f"No of people who left based on {i}",fontsize=9)
        #below code is to print count on each bar
        for bar in plots.patches:
            plots.annotate(format(bar.get_height(), ''),
                           (bar.get_x() + bar.get_width() / 2,
                            bar.get_height()),ha='center', va='center',
                           size=10, xytext=(0, 8),
                           textcoords='offset points')
        c = c + 1
    plt.subplots_adjust(right=1.2,wspace=0.5,hspace=0.8)
    plt.show()
```

```
#column=['Designation','QuarterlyRating','LastWorkingDate_year','LastWorkingDate_month']
column=['Hike%','QuarterlyRating','Promotion','Designation']
CountPlot(pre_df,column,'Resign')
```

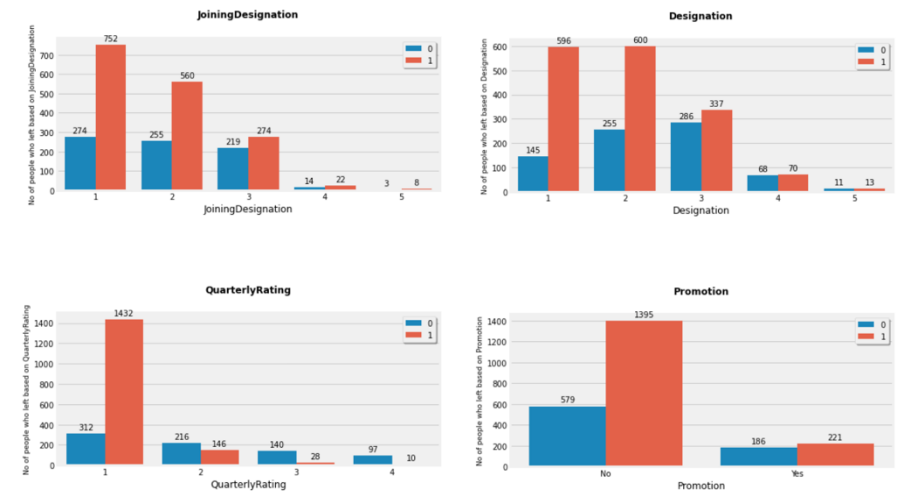
Code snippet for Bi-variate plot




```
column=['Salary_change','Designation_change','TotalBusinessValue_change','QuarterlyRating_change']
CountPlot(pre_df,column,'Resign')
```




```
column=['JoiningDesignation','Designation','QuarterlyRating','Promotion']
CountPlot(pre_df,column,'Resign')
```



Technique: Count plots with comparison of 2 variables
Stage Used: Before Pre processing data **Purpose**: To understand the relation of 1 variable in comparison to other
Data: Visualizationdata
Inference:

1) People are more likely to resign if they haven't recieved any hike.

2) People are more likely to resign if they have 1 rating.

3) People are more likely to resign if they did not get any promotion.

4) People who are on the junior position are likely to resign.

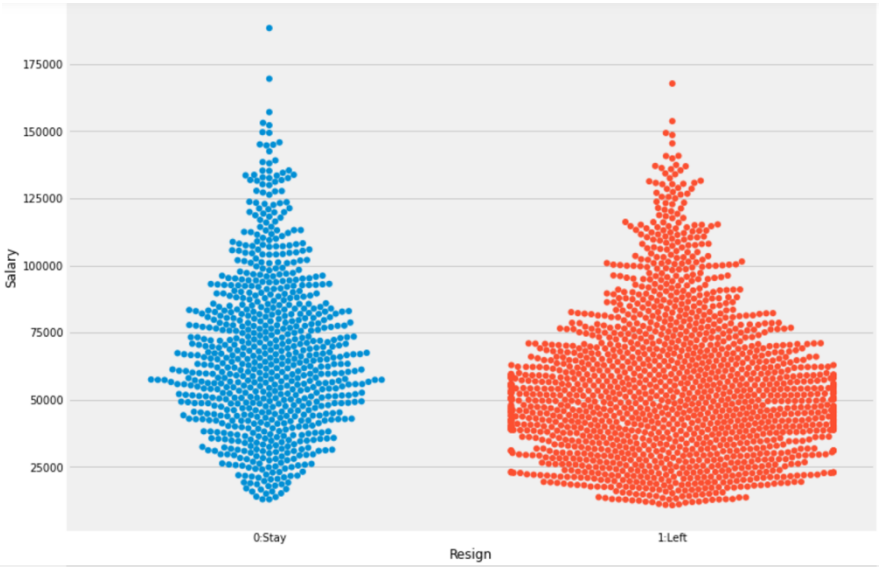
5)If there is no change in salary and designation then they are more likely to resign.

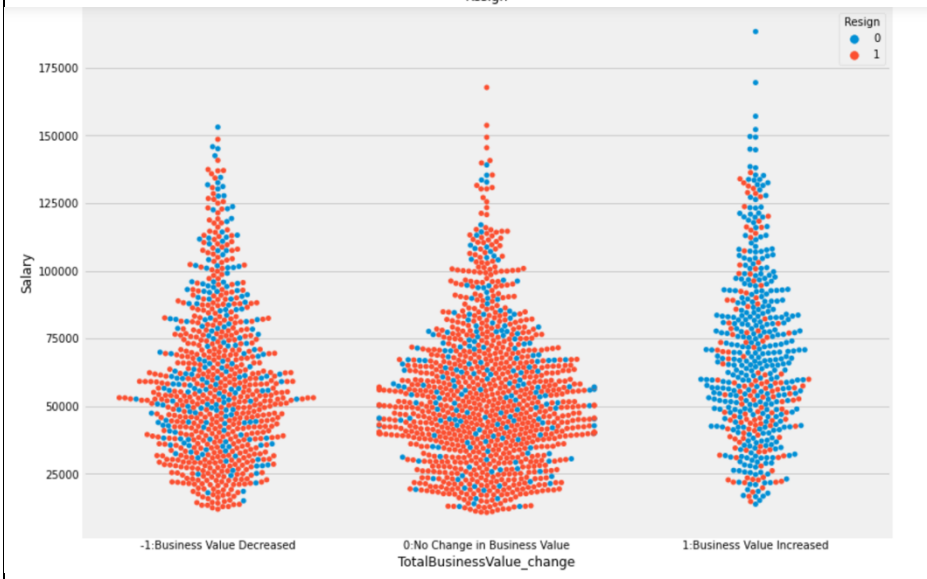
6) If there is no change in business value or if the business value decreases then they are more likely to resign.

7) If there is no change Quarterly rating then also they are more likely to resign.

```
plt.figure(figsize=(12,9))
plots=sns.swarmplot(y = pre_df['Salary'], x = pre_df['Resign'],size=6)

old_ticks =plots.get_xticks().tolist()
new_ticks = ['0:Stay','1:Left']

#Add new ticks to x axis
plt.xticks(range(len(old_ticks)), new_ticks)
plt.show()
```

```
plt.figure(figsize=(12,9))
plots=sns.swarmplot(y = pre_df['Salary'], x = pre_df['TotalBusinessValue_change'],hue=pre_df['Resign'])
old_ticks =plots.get_xticks().tolist()
new_ticks = ['-1:Business Value Decreased','0:No Change in Business Value','1:Business Value Increased']

#Add new ticks to x axis
plt.xticks(range(len(old_ticks)), new_ticks)
plt.show()
```





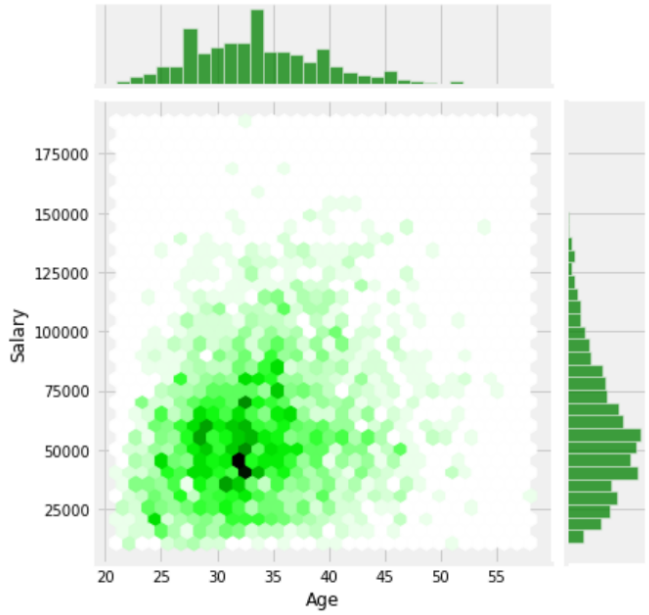
Technique: Swarn plot
Stage Used: After Pre processing data
Purpose: To see the relation between continuous and categorical variable
Data:Visualizationdata
Inference:
1)From this graph also we can confirm people who have resigned have the salary between 25000 to 75000.

2)From the second graph, we can clearly see maximum people who have resigned is either from negative or zero business value.

3)People who have shown positive business turnover are more likely to stay back as it is also directly linked to quarterly rating, promotion and salary.

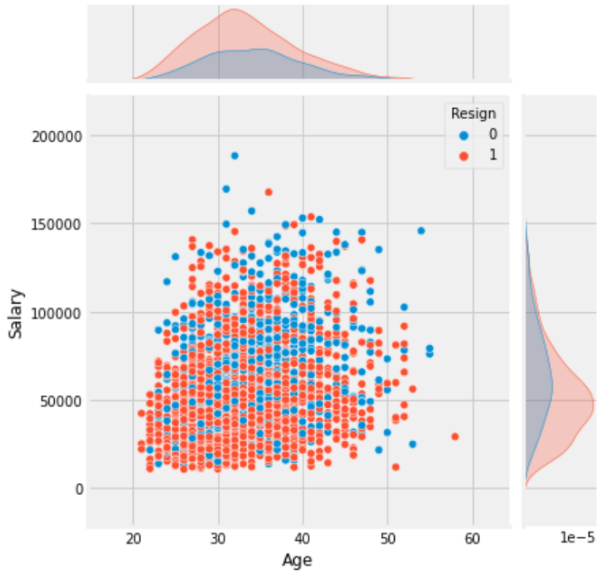
```
#joint plot between Age and Salary
sns.jointplot(x=pre_df['Age'],y=pre_df['Salary'],color='green',kind='hex')
```

```
3]: <seaborn.axisgrid.JointGrid at 0x18a00947160>
```



Code snippet for Joint plot with Hex bin

```
sns.jointplot(x=pre_df['Age'],y=pre_df['Salary'],hue=pre_df['Resign'],color='green')
plt.show()
```



Technique: seaborn Joint plots
Stage Used: After Preprocessing data
Purpose: To understand the relation between 2 numerical data
Data: Salary Vs Age
Inference:
1)In the first graph data is plotted as hex bin to know the data concentration.
From the Hex bin joint plot, majority of the data is concentrated between age of 30 to 35 and salary rate of 30K to 50K.
Mean of salary is concentrated around 50K and mean of age around 35.

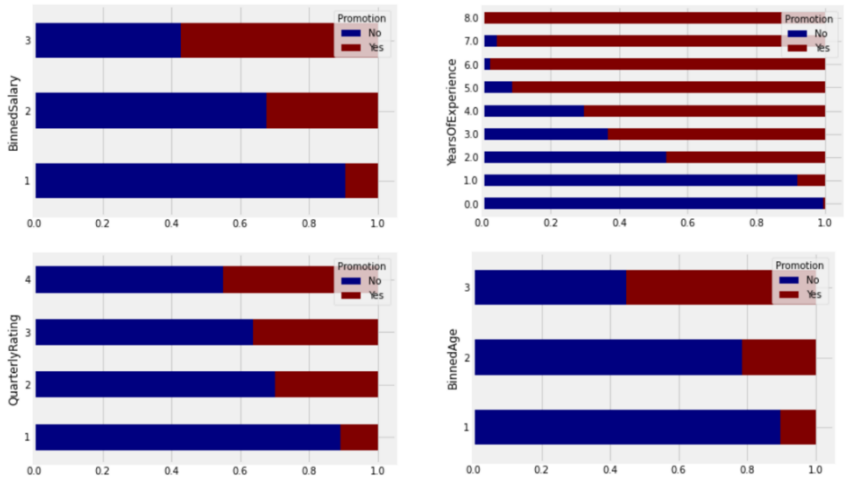
2)In the second joint plot with Hue as 'Resign' we can clearly see there is shift in mean between who resigned and who are staying back.
Salary of people who are resigned mean is at 50K and mean of people who are staying back are around 60K. Similarly for age group mean of people who resigned tends towards age of 30 Vs average age of people who are staying back tends towards 40 years.

3)To summarize people with lower salary and junior people tend to leave the organisation.

```
from matplotlib import cm
plt.figure(figsize=(14,6))
cmap = cm.get_cmap('jet')
cmap1 = cm.get_cmap('Spectral')

columns=['BinnedSalary','QuarterlyRating','Promotion','YearsOfExperience','BinnedAge']
df_bar_plot=pre_df[columns]

for i in df_bar_plot.columns:
    if(i!='Promotion'):
        status=pd.crosstab(df_bar_plot[i],df_bar_plot['Promotion'])
        status.div(status.sum(1).astype(float), axis=0).plot(kind="barh", cmap=cmap, stacked=True)
    plt.show()
```

```
<Figure size 1008x432 with 0 Axes>
```

Code snipper for Bi-variate analysis



Bi-variate Bar-h graph using pandas cross tab

Technique: Pandas cross tab
Stage Used: After Pre processing data **Purpose**: To understand relation between 2 variables
Data:Visualization data
Inference:

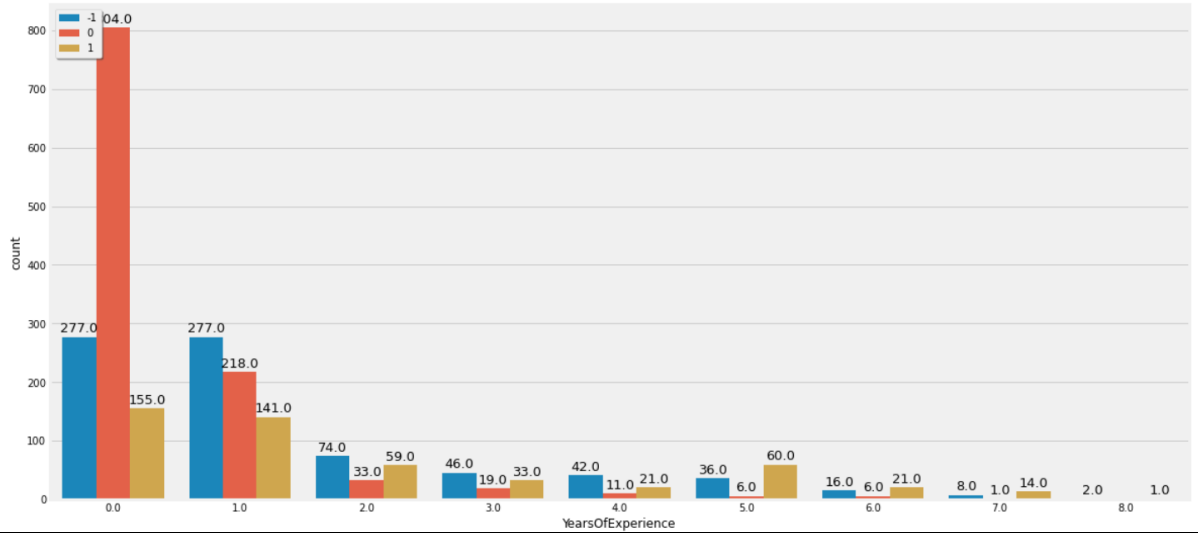
1)Promotion has direct relation with higher salary. For lower salary promotion is quite less.

2)Promotion is also quite consistent with years of experience. We could see 50% promotion after 2 years of experience and much higher for 5 years and above.

3)Promotion has got good relation with quarterly rating which is quite consistent.

4)Also promotion is high for middle and senior level people which is consistent with years of experience.

```
#Years of experience
plt.figure(figsize=(18,9))
plots=sns.countplot(pre_df['YearsOfExperience'],hue=pre_df['TotalBusinessValue_change'])
plt.legend(loc='upper left',shadow=True)

for bar in plots.patches:
    plots.annotate(format(bar.get_height(), ''),
                           (bar.get_x() + bar.get_width() / 2,
                            bar.get_height()),ha='center', va='center',
                           size=13, xytext=(0, 8),
                           textcoords='offset points')
plt.show()
#pre_df.drop('YearsOfExperience',axis=1,inplace=True)
```



Technique: Bar graph

Stage Used: After pre-processing

Purpose: to understand the relation between 2 variables: Business value and years of experience

Data: dataframe – Business value change Vs Years of experience

Inference:

1. With lesser years of experience Business value change is either 0 or negative.
2. Business value change is positive with higher years of experience.
3. Also majority of the data corresponds to 0 to 1 year experience.

Figure 22:Bar Graph

```
# #pair plot
g.map_diag(sns.kdeplot)
g.map_upper(plt.scatter)
g.map_lower(sns.histplot)
plt.show()
```

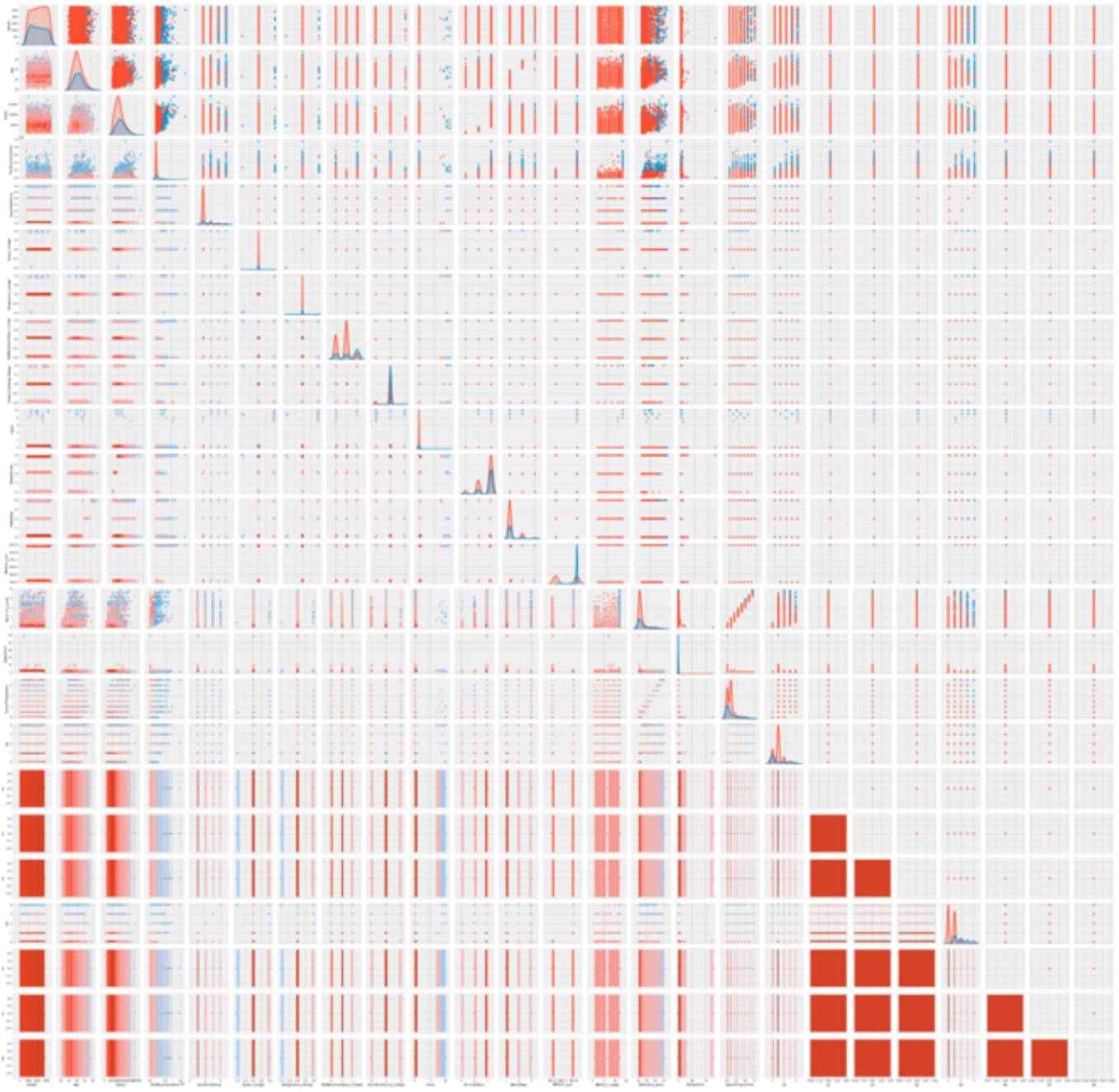
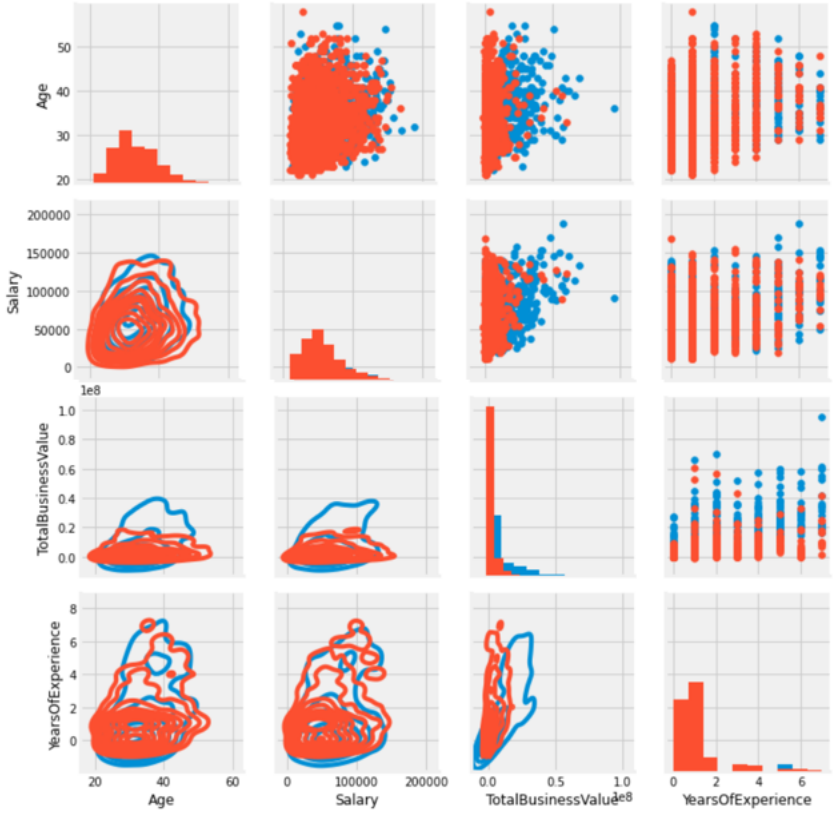


Figure 23:Pair Plot

```
import warnings
warnings.simplefilter(action='ignore')
plt.figure(figsize=(14,7))
#Let us see the pairplot w.r.t to class label and the correlation of different fearures.
column=['Age','Salary','TotalBusinessValue','YearsOfExperience','Resign']
g = sns.PairGrid(pre_df[column], hue="Resign")
colors=['red','yellow']
g.map_diag(plt.hist)
g.map_upper(plt.scatter)
g.map_lower(sns.kdeplot)
plt.show()
```

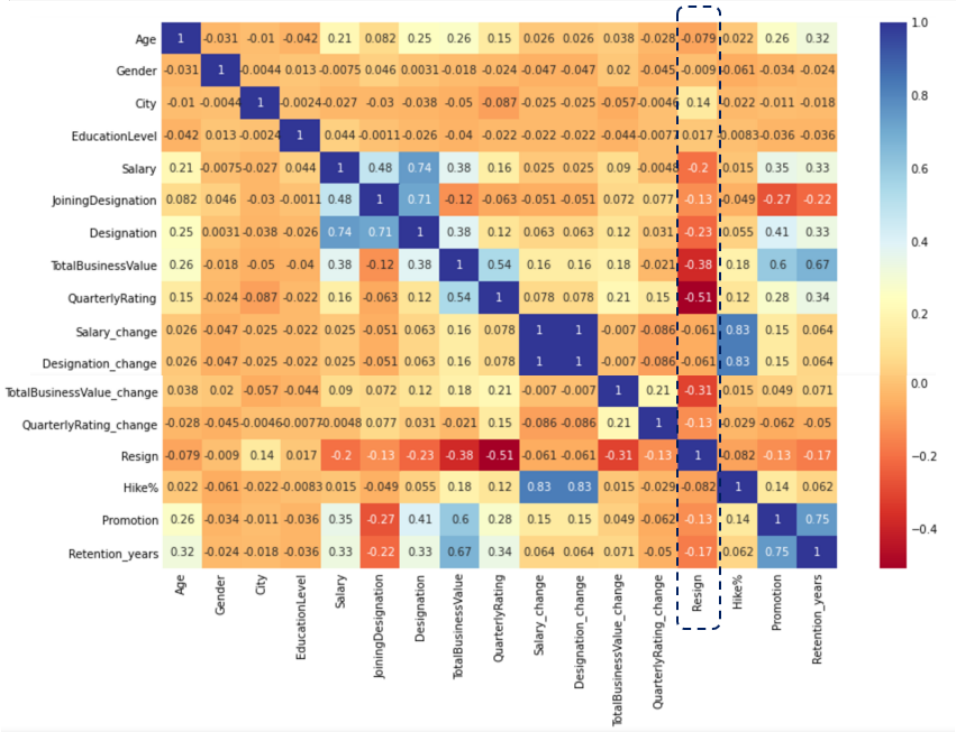
```
<Figure size 1008x504 with 0 Axes>
```



Technique: Pair plot with KDE on one side diagonal and Scatter on other side (Hue as "Resign")
Stage Used: After pre-processing
Purpose: To understand co-relation among input variables
Data: All numerical input variables
Inference:

1. From Age and Salary KDE relation we can understand concentration of people who have resigned fall on the lower salary category and lower age.
2. From the relation between Total business value and Resignation, we can clearly see most of the people who have resigned have a lower business turnover.
3. Years of experience and Age are well correlated. Also maximum concentration of people who have resigned fall in lower age and lower years of experience.
4. Total business value also has a good relation with Salary.

```
#heatmap
pc_df=ChangeDataTypes(pc_df,['JoiningDesignation','Designation','Resign'],'int64')
plt.figure(figsize=(12,9))
sns.heatmap(pc_df.corr(),annot=True,cmap='RdYlBu')
```



Technique: Correlationplot/Heat map

Stage Used: After pre-processing

Purpose: To understand co-relation among categorical/numerical variables

Data: Numerical variables

Inference:

1)Quaterly rating has a strong influence on resignation.

2) Age, gender and Education have least relation with Resignation.

3)Designation change and Salary change are strongly related. Hence it is enough if we retain 1 attribute while modelling

4)Promotion and Retention years are also strongly related to total business value change

Figure 24:Correlation among numerical variables

Summary of top observations from the visualization plots:

1. Given 2381 employee data 1616 employees have resigned and 765 are continuing in the same organisation. This is a huge attrition rate in the organisation which is at 67%. Also, the data set is imbalanced given 2 category of people who have left and who are staying back.

2. Data set has employees who have joined from 2010 to 2017, however the status of employees and several attributes like salary, promotion, Business value and quarterly ratings corresponds to the year 2016 to 2017. This is an intermediate data set for the year 2016-2017 covering 4 quarters of changes per employee.

3. 84% of the employees are below 40 years of age with mean age at 34 years. Maximum group of people are in the lower age group.

4. Out of given dataset for 2381 employees, 2008 employees have joined the organisation only from 2015 to 2017 (84%). This is consistent with the Age group as we have maximum set of people in Junior age group and with lower salary.

5. Total business value generated by each employee has a strong correlation with quarterly rating, salary, grade change and resignation. Since Maximum group of people are less than 3 years of experience (joined from 2015 to 2017), Age and salary scale are also concentrated on the lower side.

6. Maximum employees are rated as '1' (lowest performance rating) which can be related to huge attrition.

7. From Sworn plot, maximum employees who have left the organisation have either 0 or negative business turnover. This can be well related to the lower performance rating, lower salary and no designation change for those employees.

8. 99% (2369 out of 2381) of the employees have no change in salary or promotion during 2016-2017. Which can also be related to high attrition of 67%.

11. Data Encoding:

Different encoding technique are used for the numerical data conversion as most of the modelling techniques works only with numerical data.

- **One-Hot Encoding**
  - In this method, we map each category to a vector that contains 1 and 0 denoting the presence of the feature or not. The number of vectors depends on the categories which we want to keep. This is typically used for categorical data with 2 or more status.

    Attributes used: Gender, Promotion, Resignation
- **Target Encoding**
  - Target Encoding is similar to label encoding, except here labels are correlated directly with the target. This technique is used when there are more than 5 categories.

    Attributes used: City
- **Label Encoding**
  - In label encoding, we map each category to a number or a label.

    Attributes used: Binned Age, Education level, Binned salary

```
# Target Encoding
df_test = df_t1.copy()
columns = ['City']
df_test=convertToNumerical(df_test,columns,"target")
df_test.head()
```

Age	Gender	City	EducationLevel	Salary	JoiningDesignation	Designation	TotalBusinessValue	QuarterlyRating	Salary_change	Designation_change	TotalBusinessValue_change	QuarterlyRating_change	Resign	Hike%	Promotion	BinnedSalary	BinnedAge	Retention_years
28	1	0.770270	2	57387	1	1	1715580	2	0	0	1	0	1	0.0	0	1	1	0.21
31	1	0.684211	2	67016	2	2	0	1	0	0	0	0	0	0.0	0	1	1	0.07
43	1	0.816901	2	65603	2	2	350000	1	0	0	-1	0	1	0.0	0	1	2	0.39
29	1	0.706667	1	46368	1	1	120360	1	0	0	-1	0	1	0.0	0	1	1	0.16
31	0	0.703125	0	78728	3	3	1265000	2	0	0	-1	0	0	0.0	0	2	1	0.34

Figure 25:Encoded data set

12. Outlier detection:

Box plot and histogram technique is used detect the outlier for the numerical data.

```
#function to detect outliers
def BoxPlotOutliers(dataframe,col):
    fig = plt.figure(figsize=(14,6))

    ax=plt.subplot(1,2, 1)
    sns.boxplot(x=dataframe[col],ax=ax)
    plt.xlabel(col)
    plt.title(f'Outlier in {col}',fontsize=12,fontweight='bold',y=1.1)
    #sns.boxplot(x=dataframe[i],hue=dataframe[target_column])

    ax=plt.subplot(1,2, 2)
    sns.distplot(x=dataframe[col],ax=ax,color='red')
    plt.xlabel(col)
    plt.title('Distribution before any outliers handling',fontsize=12,fontweight='bold',y=1.1)
    plt.subplots_adjust(right=1.2,wspace=0.5,hspace=0.8)
    plt.show()
```

```
columns=['Age','Salary']
for c in columns:
    BoxPlotOutliers(pre_df,c)
```

Figure 26:Code Snippet – Outlier detection

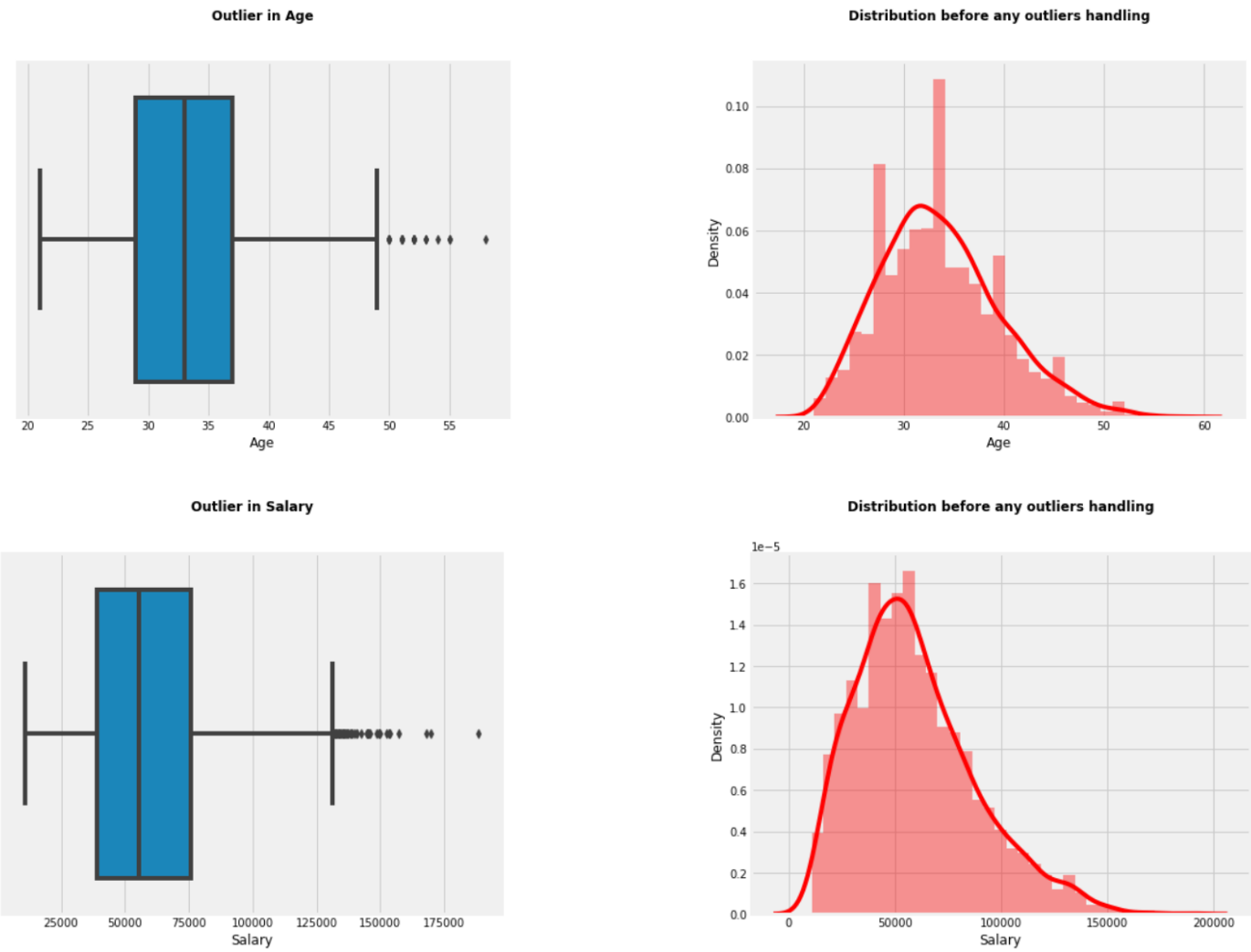


Figure 27: Outliers in Features: Age and Salary

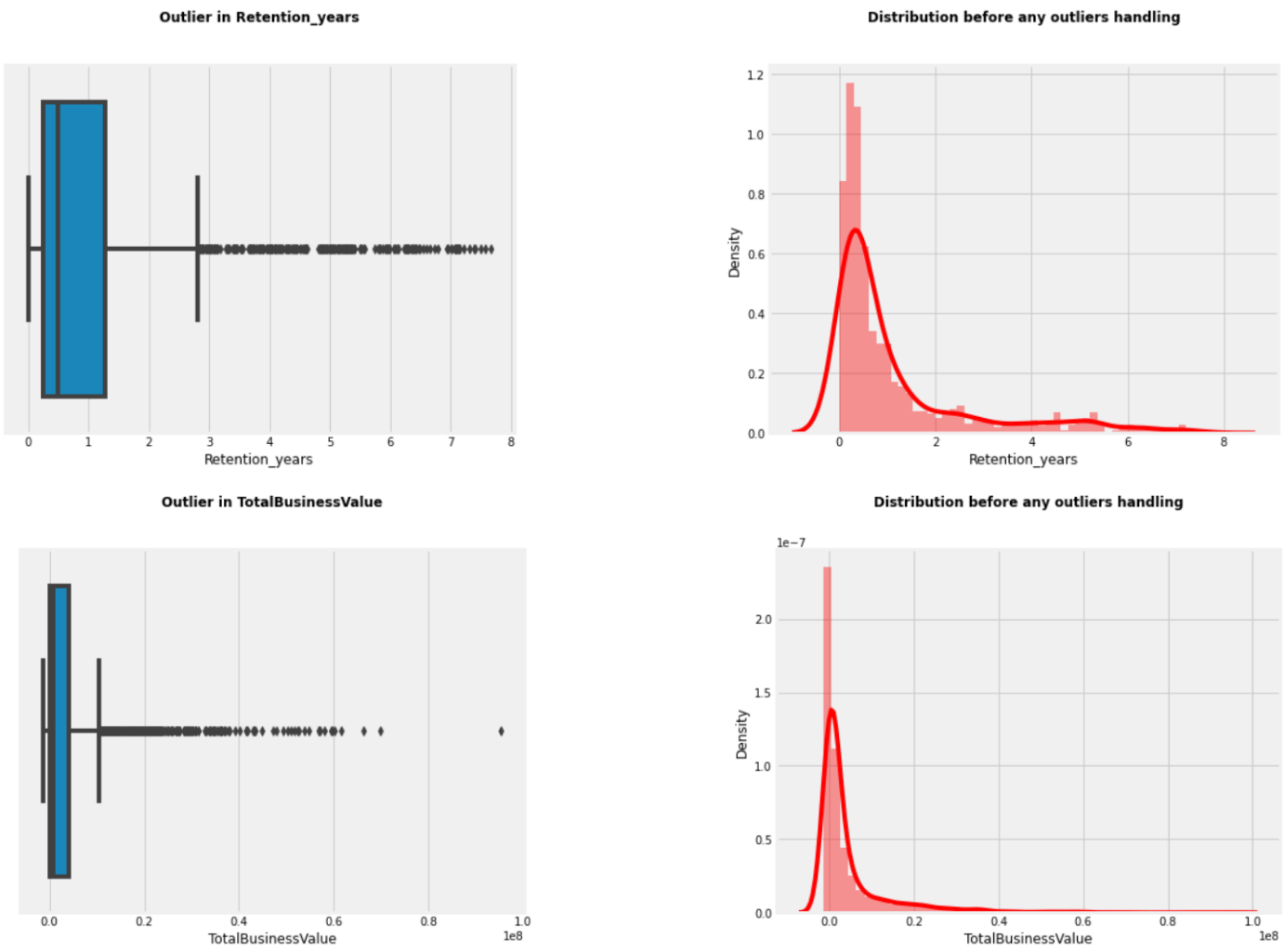


Figure 28: Outliers in Features: Retention_years and TotalBusinessValue

13. Multivariate outlier detection using DB SCAN and K-Means

Clustering techniques are used to find the outliers in the overall data set using DB SCAN and K-mean techniques. K-means however didn't detect any outlier as it is based on hard clustering and all points are taken in to clusters.

DB-Scan technique: for DB scan we need 2 hyperparameters Eps and min points. Best Eps value is determined by using iterative method using best Silhouette score and min points is taken as 5 for our visualization.

DB scan is able to detect 13 outliers in the given data set with 3 Clusters.

```
#outliers using knn
data=FindOutliersDBKNN(df_pca,"dbscan")
```

```
DBSCAN Outliers detection
Best EPS value is 0.9599999999999995 and silhouette score is 0.6525663220156511
```

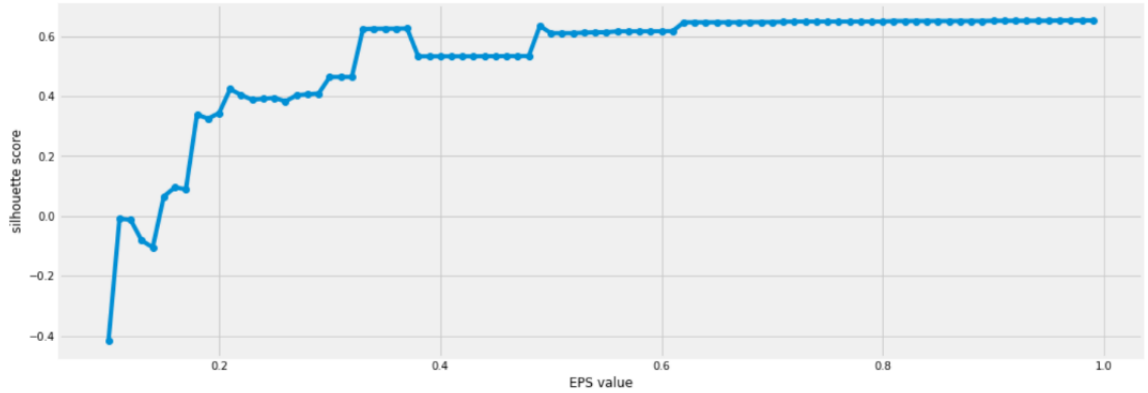


Figure 29:Code Snippet – Clustering technique

```
Identified Outliers: 13
Identified Clusters: 2
```

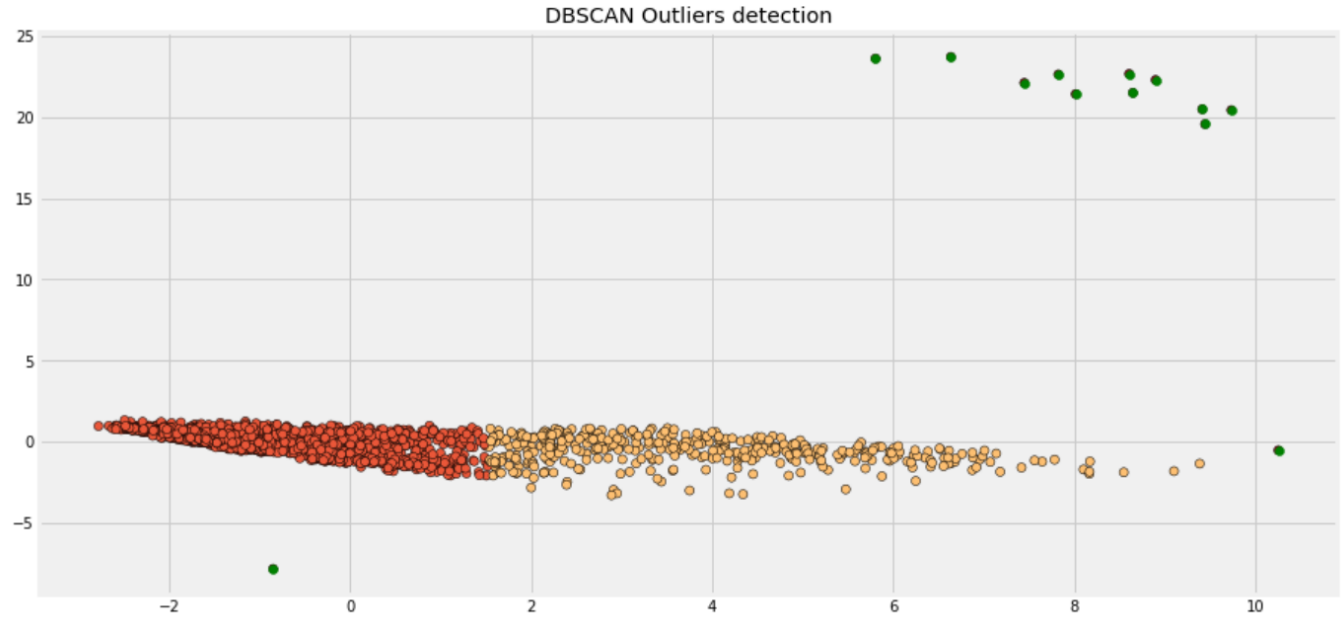


Figure 30:Outliers visualization in 2D

K-means technique need hyperparameter of n-cluster before we could visualize the outliers.

- Hence elbow method and Silhouette methods were used to find the optimum clusters to visualize the outliers.
- K-means technique could not detect outliers as all the points are added to the clusters.

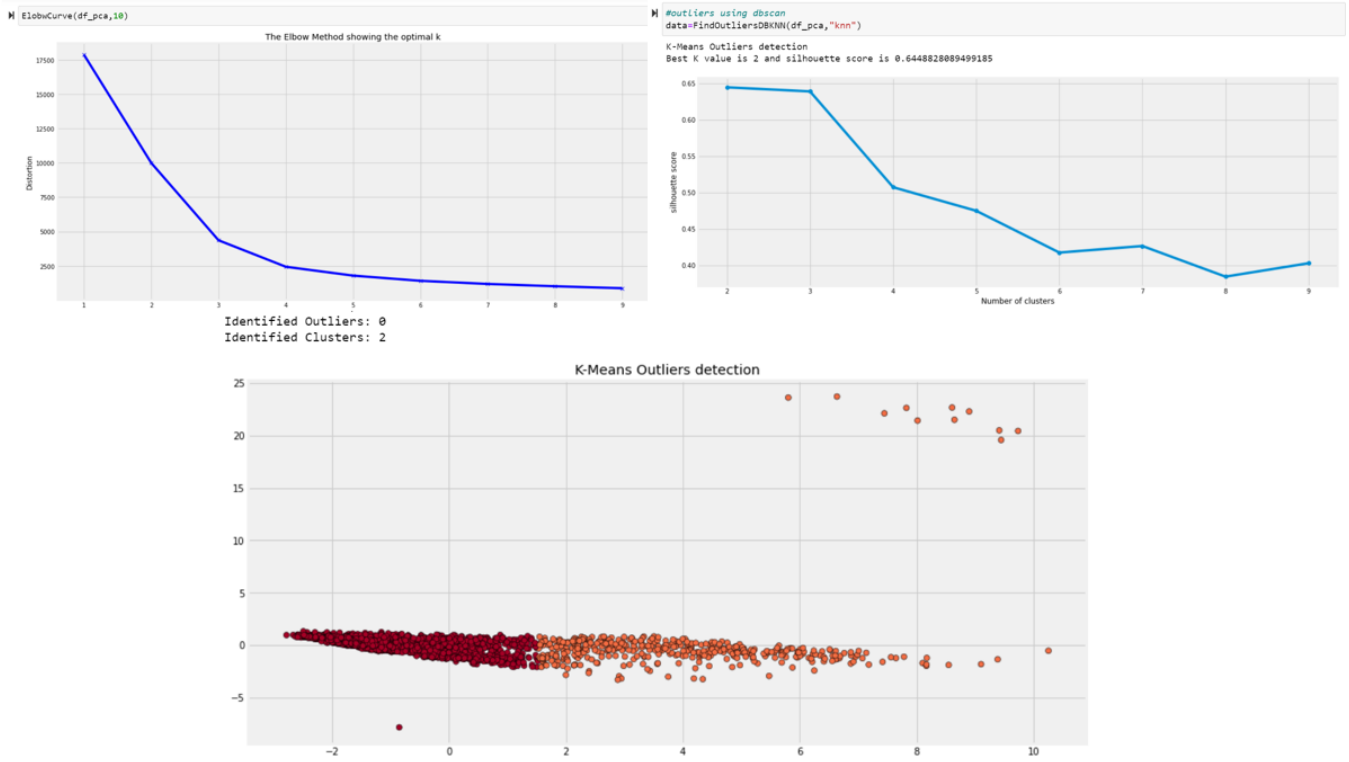


Figure 31:Code Snippet – K-means technique

14. Outliers' removal

Outliers in the numerical data is removed by using inter -quartile technique

```
columns=['Age','Salary']
df_test=RemoveOutLiers(df_test,columns,"")
```

```
Outliers using interquartile range..
Age
Percentiles: 25th=29.000, 75th=37.000, IQR=8.000
Identified outliers: 25
Outliers removed..

Salary
Percentiles: 25th=39104.000, 75th=75986.000, IQR=36882.000
Identified outliers: 48
Outliers removed..
```

Figure 32:Code Snippet – outlier removal

15. Feature Distribution and Normalisation

- Feature Distribution

```
CheckDistribution(train_df,'Salary')
```

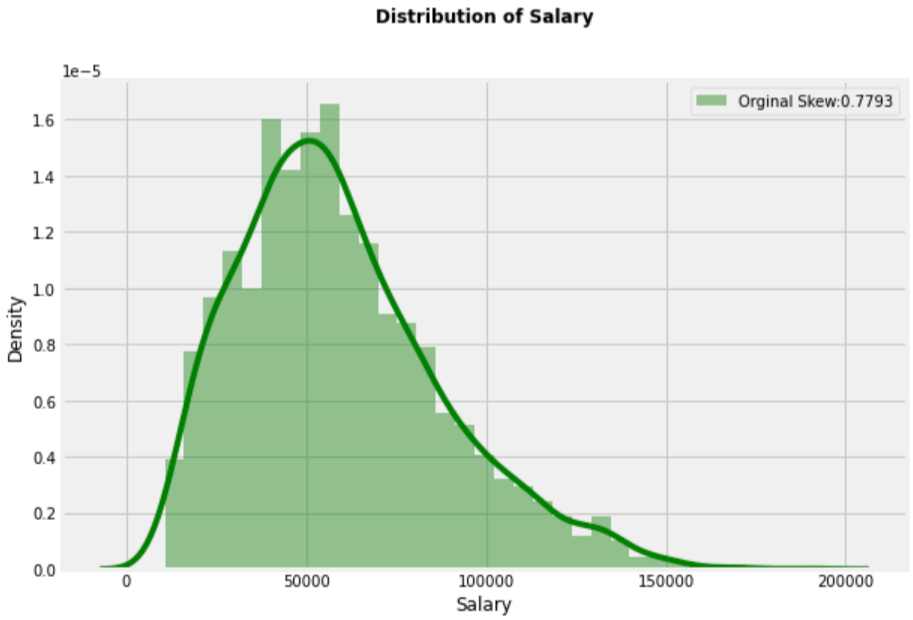


Figure 33: Salary Distribution

```
CheckDistribution(train_df,'TotalBusinessValue')
```

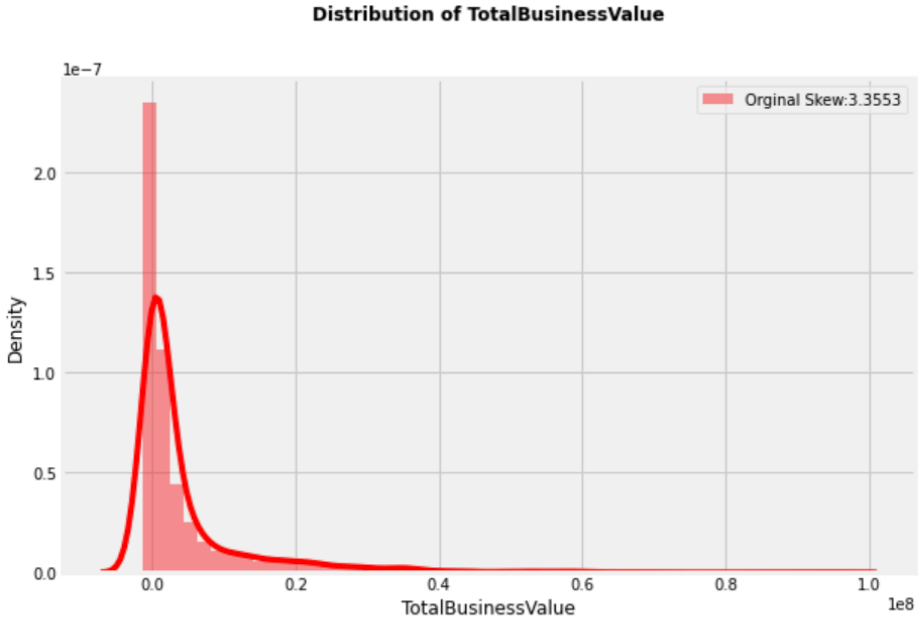


Figure 34:TotalBusinessValue Distribution

Skewness is a measure of asymmetry of a distribution.
In a normal distribution, the mean divides the curve symmetrically into two equal parts at the median and the value of skewness is zero. When a distribution is asymmetrical the tail of the distribution is skewed to one side-to the right or to the left.
When the value of the skewness is negative, the tail of the distribution is longer towards the left side

of the curve. When the value of the skewness is positive, the tail of the distribution is longer towards the right side of the curve.

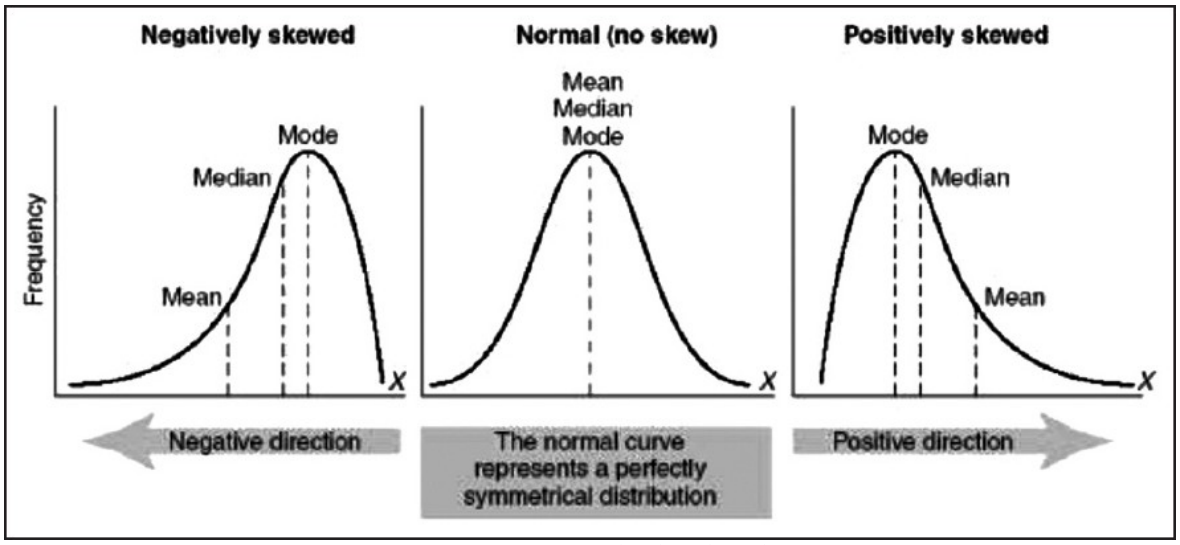


Figure 35: Normalized vs Skewed Data

- If the skewness is between -0.5 and 0.5, the data are fairly symmetrical.
- If the skewness is between -1 and — 0.5 or between 0.5 and 1, the data are moderately skewed.
- If the skewness is less than -1 or greater than 1, the data are highly skewed

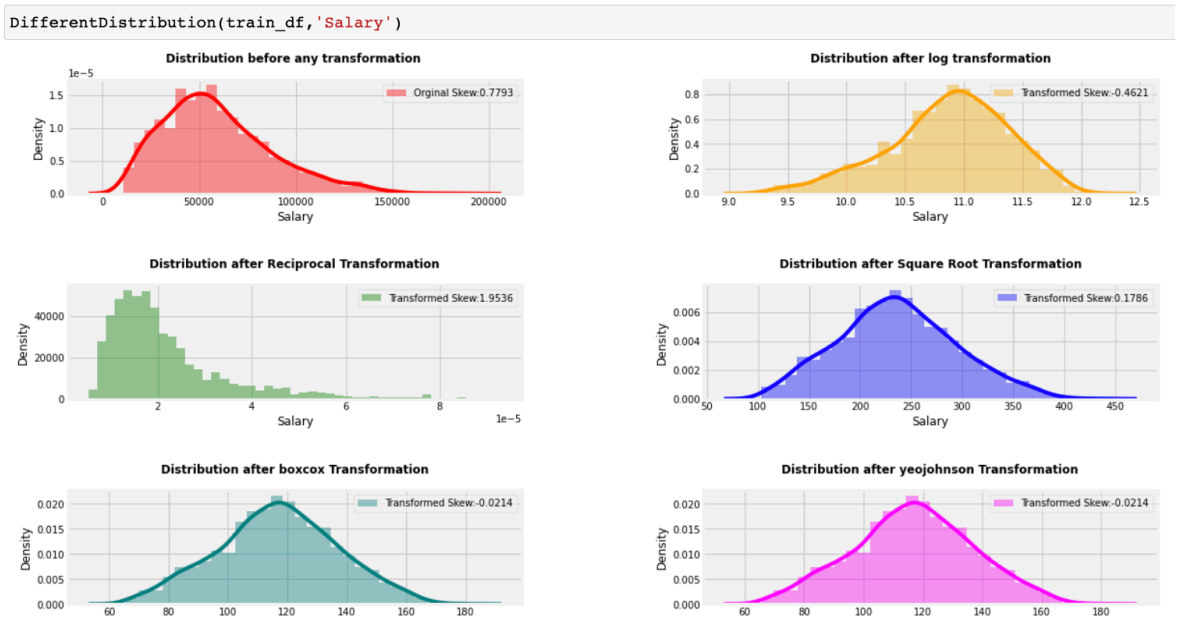


Figure 36:Before and after different feature transformation techniques

16. PCA Analysis

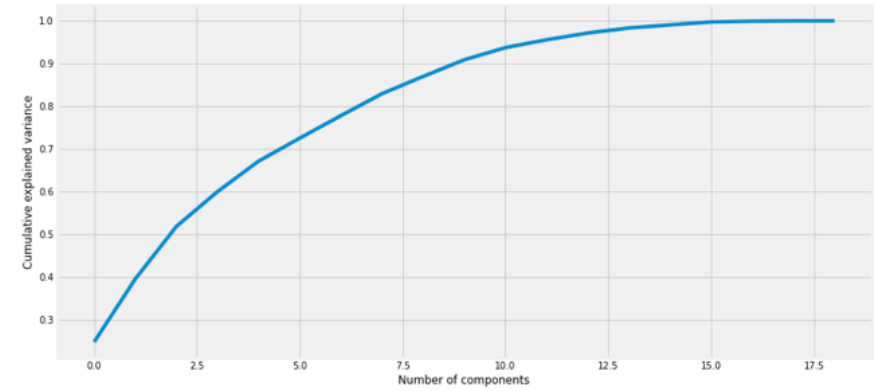


Figure 37 Feature Contribution to PCA components

Technique: PCA

Stage Used: PCA wasdone towards the end of pre-processing step.

Purpose: To understand how many principal components give maximum variance

Data: All the data variables after normalization

Inference: When applied on normalized data, 95% variance can be preserved with 10 principal components.

Principal component Analysis of all the features is done. When applied on normalized data, 95% variance is preserved in 10 principal components. A detailed Analysis of contribution of each feature to the 10 Principal components were done.

	Age	Gender	City	EducationLevel	Salary	JoiningDesignation	Designation	TotalBusinessValue	QuarterlyRating	Salary_change	Designation_change	TotalBusinessValue_change	QuarterlyRating_change	Hike%	Promotion	BinnedSalary	BinnedAge	Retention_years	YearsOfExperience
0	0.231407	0.019728	0.026085	0.019755	0.307478	0.050187	0.306991	0.360914	0.212477	0.115013	0.115013	0.078339	0.010308	0.111134	0.354027	0.286340	0.214418	0.376157	0.372069
1	0.075115	0.046413	0.008654	0.018349	0.208628	0.244507	0.195107	0.050422	0.045144	0.524838	0.524838	0.042073	0.071921	0.492495	0.051603	0.192475	0.075171	0.006093	0.005832
2	0.061926	0.009704	0.022928	0.030421	0.318744	0.522616	0.355242	0.150256	0.132906	0.230581	0.230581	0.022636	0.025233	0.206586	0.206442	0.305205	0.068063	0.283222	0.287542
3	0.644125	0.042437	0.050168	0.062737	0.118825	0.055983	0.056738	0.155979	0.147762	0.047594	0.047594	0.118598	0.087694	0.024014	0.124910	0.114764	0.657024	0.111184	0.102640
4	0.079969	0.053755	0.187292	0.121451	0.107338	0.063186	0.046168	0.097225	0.415523	0.004381	0.004381	0.576624	0.576933	0.052475	0.154533	0.112139	0.102308	0.111583	0.116693
5	0.043652	0.740165	0.575384	0.185142	0.030168	0.023272	0.021806	0.045281	0.077259	0.011254	0.011254	0.061884	0.253558	0.011805	0.045389	0.048184	0.039244	0.016833	0.015486
6	0.045007	0.352751	0.208021	0.886879	0.047899	0.060508	0.069434	0.005050	0.086238	0.011802	0.011802	0.091718	0.101346	0.017946	0.026157	0.041332	0.052531	0.023280	0.022278
7	0.024174	0.518448	0.731635	0.366002	0.003040	0.010416	0.019955	0.014502	0.019490	0.028734	0.028734	0.150482	0.180155	0.040797	0.004437	0.012593	0.039847	0.004000	0.008630
8	0.005185	0.206374	0.187578	0.056519	0.022061	0.000679	0.067685	0.154049	0.238508	0.029244	0.029244	0.505494	0.726018	0.039239	0.135432	0.030539	0.001944	0.118405	0.118925
9	0.006864	0.046810	0.122646	0.086551	0.019913	0.106912	0.044300	0.220676	0.696170	0.038992	0.038992	0.592128	0.038307	0.009608	0.194815	0.003810	0.009066	0.132548	0.131353

Figure 38:Feature contribution to 10 Principal Component

When sorted in increasing order for each Principal component, the feature importance obtained is as below.

	0	1	2	3	4	5	6	7	8	9
Retention_years	0.376157	0.006093	0.283222	0.111184	0.111583	0.016833	0.023280	0.004000	0.118405	0.132548
YearsOfExperience	0.372069	0.005832	0.287542	0.102640	0.116693	0.015486	0.022278	0.008630	0.118925	0.131353
TotalBusinessValue	0.360914	0.050422	0.150256	0.155979	0.097225	0.045281	0.005050	0.014502	0.154049	0.220676
Promotion	0.354027	0.051603	0.206442	0.124910	0.154533	0.045389	0.026157	0.004437	0.135432	0.194815
Salary	0.307478	0.208628	0.318744	0.118825	0.107338	0.030168	0.047899	0.003040	0.022061	0.019913
Designation	0.306991	0.195107	0.355242	0.056738	0.046168	0.021806	0.069434	0.019955	0.067685	0.044300
BinnedSalary	0.286340	0.192475	0.305205	0.114764	0.112139	0.048184	0.041332	0.012593	0.030539	0.003810
Age	0.231407	0.075115	0.061926	0.644125	0.079969	0.043652	0.045007	0.024174	0.005185	0.006864
BinnedAge	0.214418	0.075171	0.068063	0.657024	0.102308	0.039244	0.052531	0.039847	0.001944	0.009066
QuarterlyRating	0.212477	0.045144	0.132906	0.147762	0.415523	0.077259	0.086238	0.019490	0.238508	0.696170
Salary_change	0.115013	0.524838	0.230581	0.047594	0.004381	0.011254	0.011802	0.028734	0.029244	0.038992
Designation_change	0.115013	0.524838	0.230581	0.047594	0.004381	0.011254	0.011802	0.028734	0.029244	0.038992
Hike%	0.111134	0.492495	0.206586	0.024014	0.052475	0.011805	0.017946	0.040797	0.039239	0.009608
TotalBusinessValue_change	0.078339	0.042073	0.022636	0.118598	0.576624	0.061884	0.091718	0.150482	0.505494	0.592128
JoiningDesignation	0.050187	0.244507	0.522616	0.055983	0.063186	0.023272	0.060508	0.010416	0.000679	0.106912
City	0.026085	0.008654	0.022928	0.050168	0.187292	0.575384	0.208021	0.731635	0.187578	0.122646
EducationLevel	0.019755	0.018349	0.030421	0.062737	0.121451	0.185142	0.886879	0.366002	0.056519	0.086551
Gender	0.019728	0.046413	0.009704	0.042437	0.053755	0.740165	0.352751	0.518448	0.206374	0.046810
QuarterlyRating_change	0.010308	0.071921	0.025233	0.087694	0.576933	0.253558	0.101346	0.180155	0.726018	0.038307

Figure 39:Feature contribution sorted w.r.t zeroth component

From the PCA analysis, it is seen that Retention_years has the highest co-efficient value for principal component 0 (which is the most important principal component). Therefore, we can deduce that Retention_years is the most important feature from variance perspective. In addition, since the dataset is not linear, dataset after PCA may not work well for modelling.

17. Class Balance in Data

- Following is the distribution between two classes (of target variable) in the given training data.

```
CheckBalanceImbalance(train_df)
```

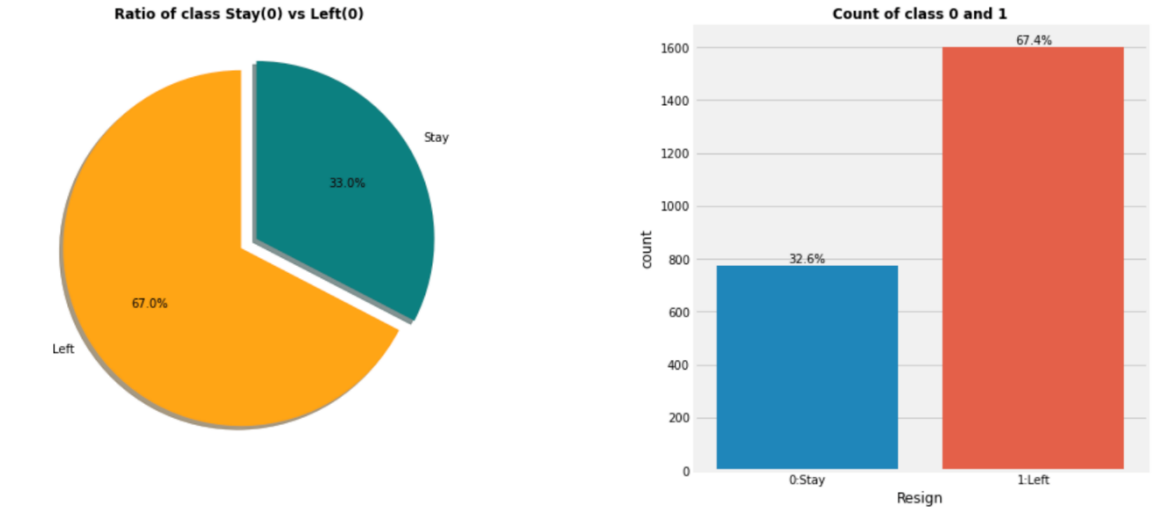


Figure 40:Class balance in the given data

- For creating a better ratio in the two classes, undersampling is done on the majority class. Following is the distribution between two classes after implementing undersampling on majority class

```
CheckBalanceImbalance(balance_df)
```

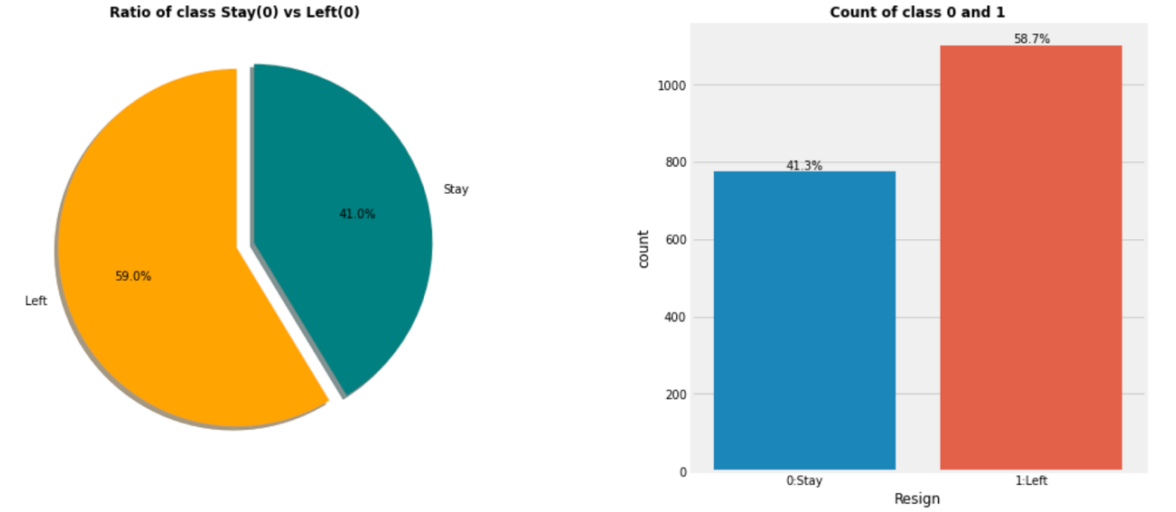


Figure 41:Class balance after undersampling

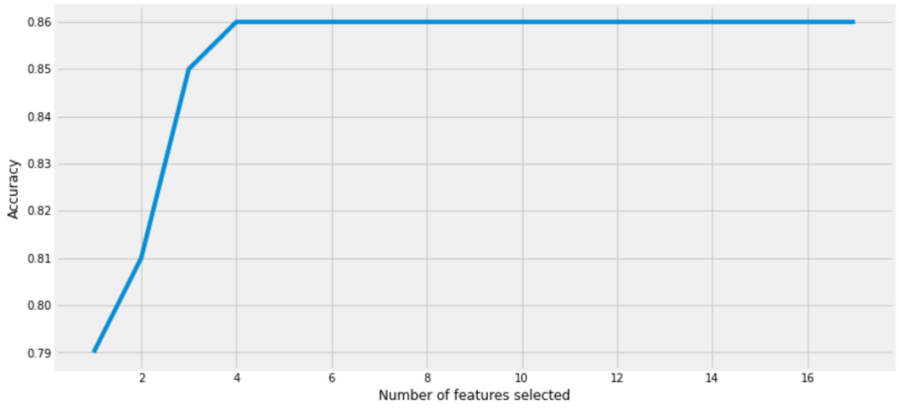
- Thus, a ratio of 58.7% (Left) – 41.3% (Stayed) from the dataset has been achieved.

18. Feature Selection

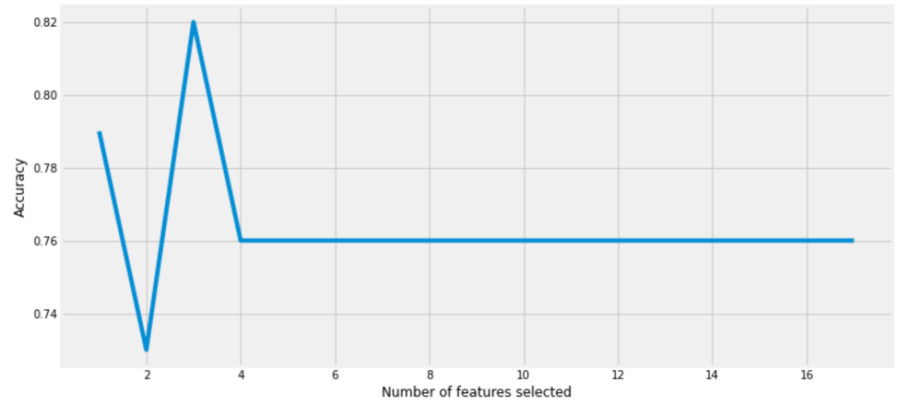
Feature selection is the process of extracting the most consistent, non-redundant, and relevant features to use in the model building to achieve the best model accuracy.

18.1. Recursive Feature Elimination (RFE)

Recursive Feature Elimination is a feature selection technique that fits a model and removes the weakest feature (or features) until the specified number of features is reached. (Machine Learning Mastery Recursive Feature Elimination (RFE) for Feature Selection in Python, n.d.) (Scikit Learn sklearn.feature_selection.RFE, n.d.)
It has been implemented on both the balanced and unbalance dataset using Decision Tree classifier.



RFE with Decision Tree on unbalanced data



RFE with Decision Tree on balanced data

Figure 42 Number of features selected vs Accuracy for RFE

Technique: RFE
Stage Used: Feature Selection
Purpose: To conclude the number of most relevant features to be extracted for model building
Data: Applied on Balanced and Unbalanced data
Inference:

1. Better accuracy is observed when implemented on unbalanced data.
2. Accuracy is consistent when 4+ most relevant features are selected.

18.2. Recursive Feature Elimination Cross Validated (RFECV)

RFECV uses cross validation with RFE to score different feature subset and select the best scoring collection of features. (Scikit Learn sklearn.feature_selection.RFECV, n.d.)
It has been implemented on unbalanced dataset using Decision Tree Classifier.

feature	rank	inv_rank
QuarterlyRating	1	12
TotalBusinessValue_change	1	12
TotalBusinessValue	1	12
Salary	1	12
Retention_years	1	12
City	1	12
Age	2	11

Figure 43:Top 7 features from RFECV with Decision Tree Classifier

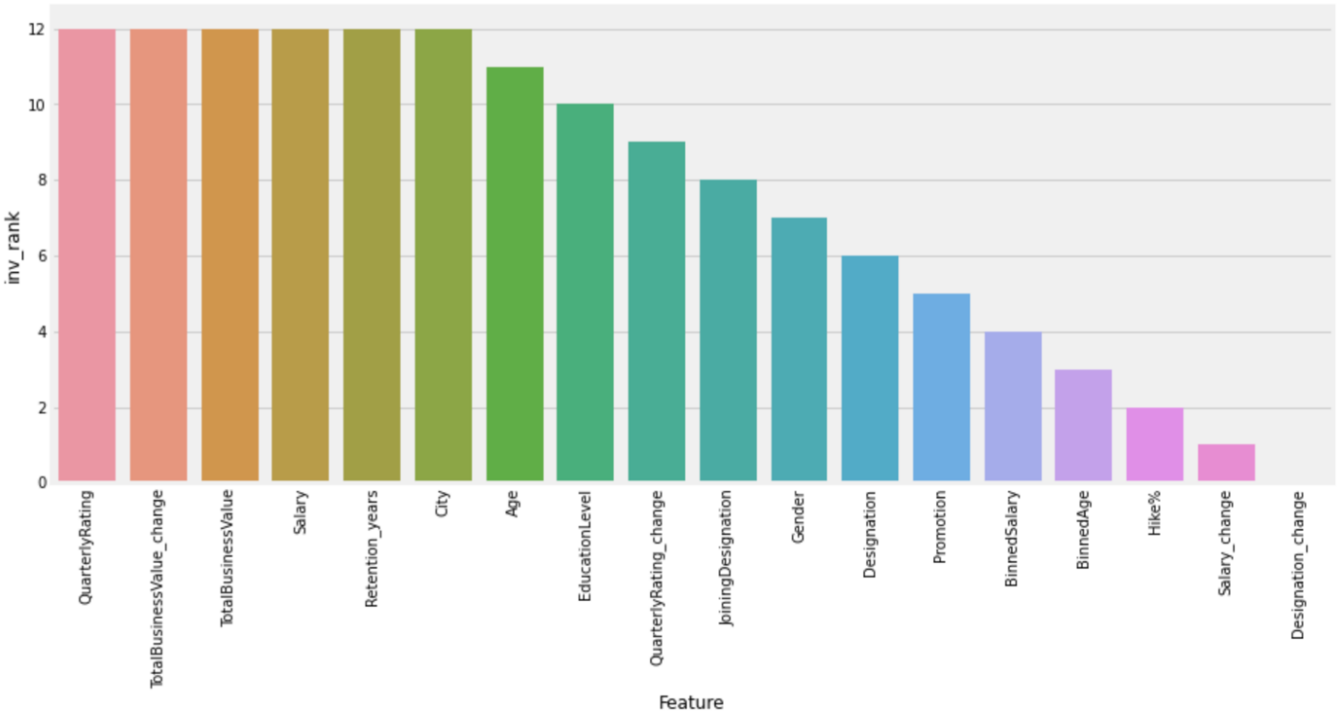


Figure 44:Features vs Inverse Rank from RFECV implementation

18.3. **SelectKBest**

SelectKBest is a feature selection technique that selects the K most explicative features out of the original data set. Different scoring function allows it to be used for both Regression and Classification problem. (Scikit Learn sklearn.feature_selection.SelectKBest, n.d.)

Score functions used for Classification are :

- f_classif – computes the ANNOVA F-value for the provided sample
- mutual_info_classif - estimates mutual information for a discrete target variable
- It has been implemented on unbalanced dataset.

```
-------Top 7 features----------
                              Score
Feature_Name
QuarterlyRating            538.902359
TotalBusinessValue         254.123971
TotalBusinessValue_change  180.975144
Designation                 90.408826
Salary                      73.325148
BinnedSalary                60.845766
Retention_years             53.223743
```

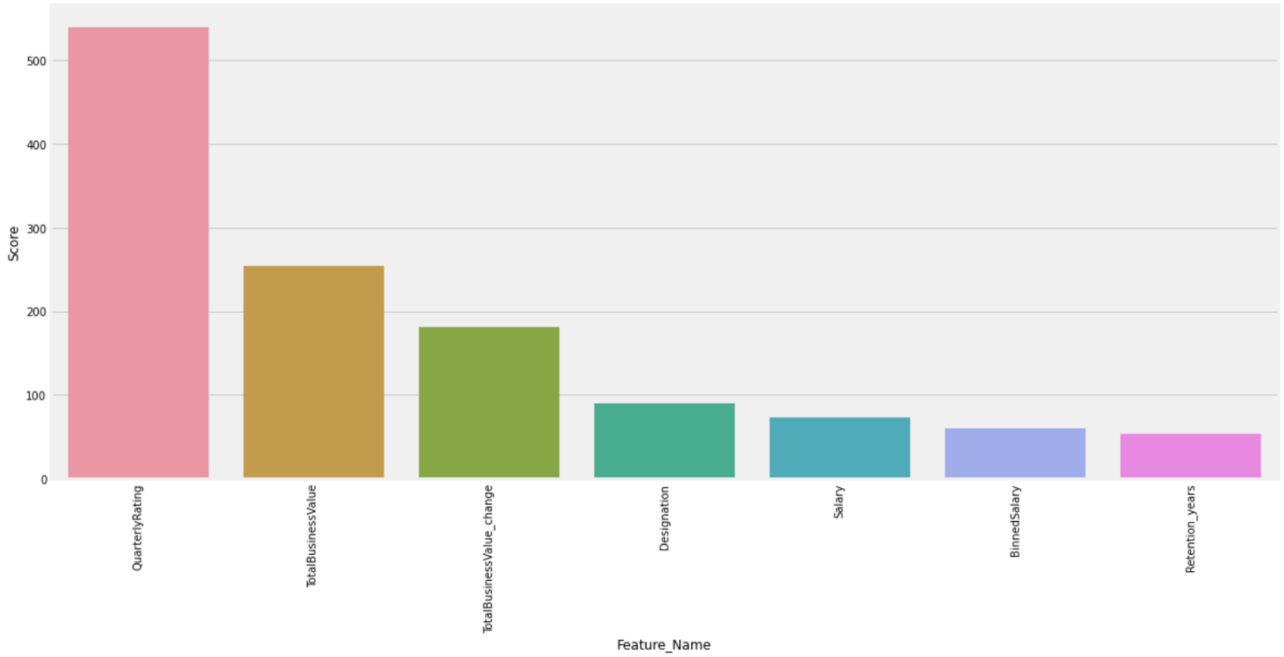


Figure 45:Top 7 Features from SelectKBest using f_classif

```
-------Top 7 features----------
                            Score
Feature_Name
QuarterlyRating            0.128559
Retention_years            0.102490
TotalBusinessValue_change  0.099625
TotalBusinessValue         0.067584
Designation                0.040167
Salary                     0.026506
City                       0.019006
```

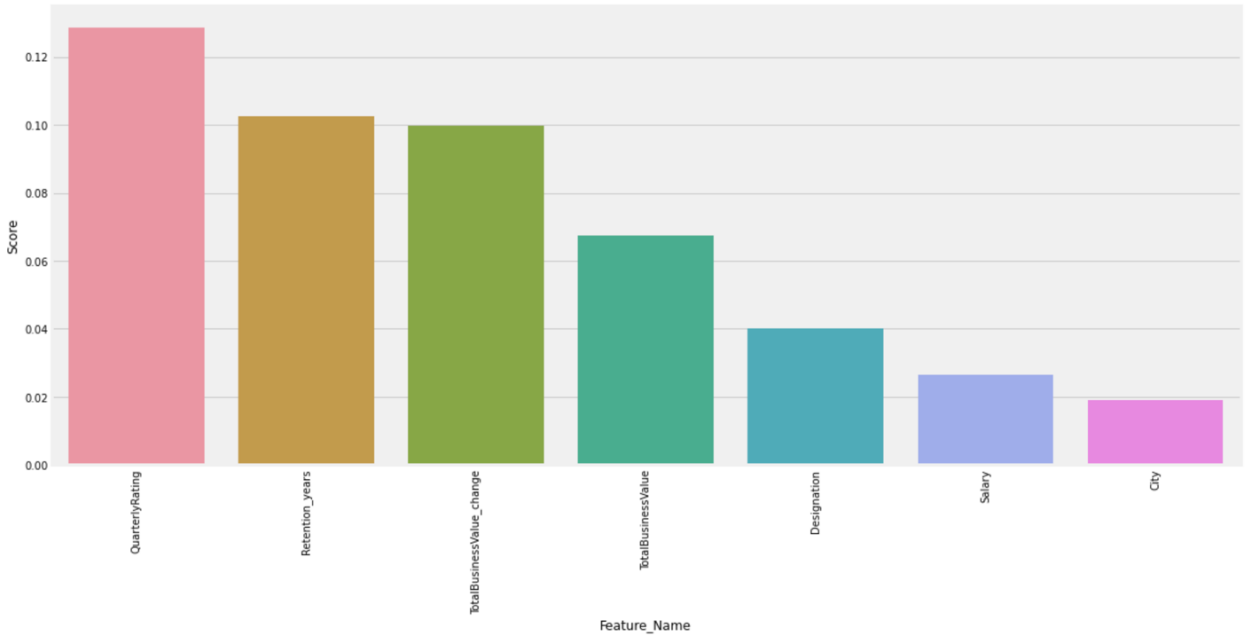


Figure 46:Top 7 Features from SelectKBest using mutual_info_classif

19. Dataset for Modelling

Multiple datasets have been created for training the models to compare the results on different datasets. Datasets created are listed below -

- Balanced Derived Dataset – the dataset is balanced and includes all the original and derived features excluding the features whose information has been extracted into derived features.
- Balanced Binned Derived Dataset – the features are replaced with binned features in Balanced Derived Dataset
- Stdscaled Balanced Binned Derived Dataset – the Balanced Binned Derived Dataset is scaled using standard scaling technique
- Minmaxscaled Balanced Binned Derived Dataset – the Balanced Binned Derived Dataset is scaled using minmax scaling technique

20. Machine Learning Modelling & Techniques Applied

Model training flow is depicted in the below diagram.

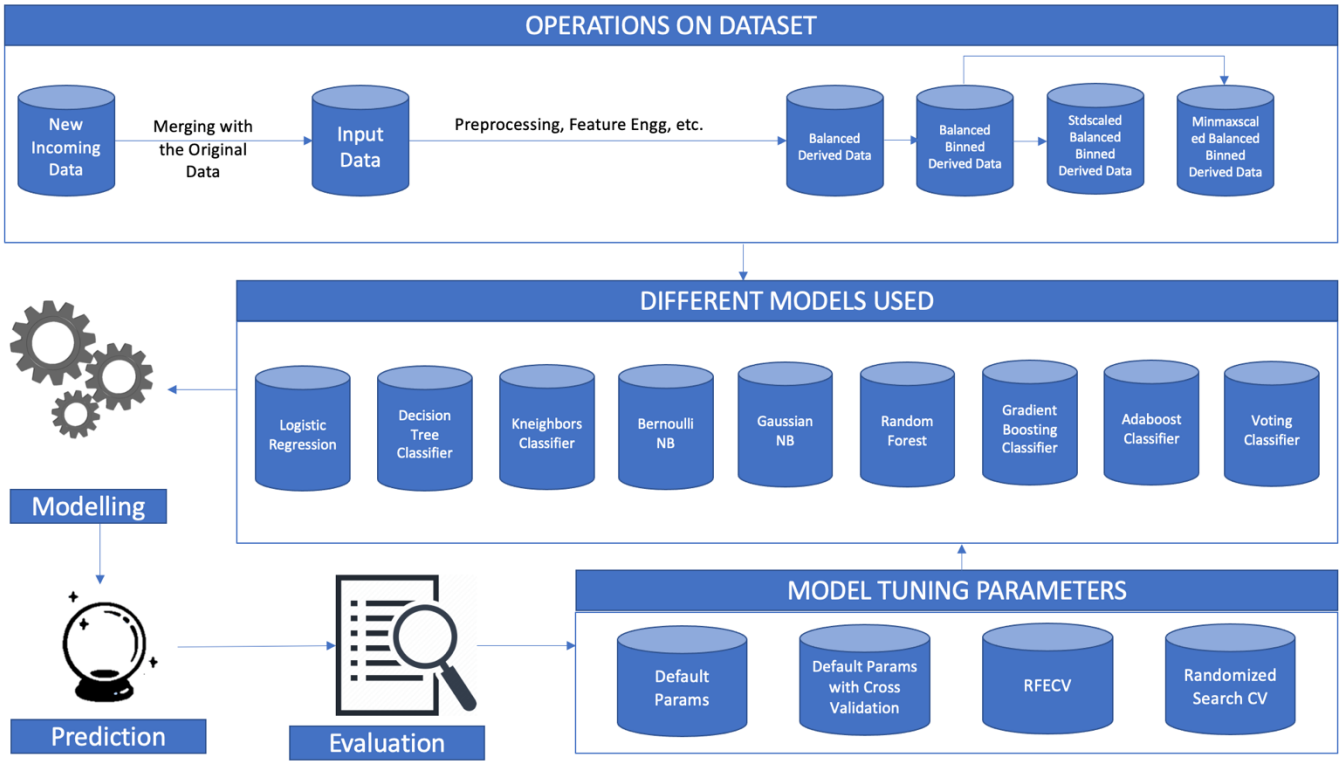


Figure 47:Model training and evaluation flow

Different types of models were created and applied on different dataset to compare their predictions and achieve the most accurate prediction among them. All the models used are discussed below.

20.1. Base Models

Base models without any Hyperparameter tuning were built on different datasets. Following table lists the accuracy, precision, recall and f1-score for each base model built.

	Model	Test Accuracy	Training Accuracy	Precision	Recall	0/1 Precision	0 Precision	1 Precision	0/1 Recall	0 Recall	1 Recall	F1-Score	Dataset
	GradientBoostingClassifier	0.87	0.90	0.87	0.91	[0.87, 0.87]	0.87	0.87	[0.80, 0.91]	0.80	0.91	0.89	Minmax scaled Balanced Binned Derived Data
	AdaBoostClassifier	0.87	0.87	0.86	0.93	[0.88, 0.86]	0.88	0.86	[0.79, 0.93]	0.79	0.93	0.90	Balanced Binned Derived Data
	GradientBoostingClassifier	0.87	0.90	0.87	0.91	[0.87, 0.87]	0.87	0.87	[0.80, 0.91]	0.80	0.91	0.89	Stdscaled Balanced Binned Derived Data
	AdaBoostClassifier	0.87	0.87	0.86	0.93	[0.88, 0.86]	0.88	0.86	[0.79, 0.93]	0.79	0.93	0.90	Stdscaled Balanced Binned Derived Data
	AdaBoostClassifier	0.87	0.87	0.86	0.93	[0.88, 0.86]	0.88	0.86	[0.79, 0.93]	0.79	0.93	0.90	Minmax scaled Balanced Binned Derived Data
	GradientBoostingClassifier	0.87	0.90	0.87	0.91	[0.87, 0.87]	0.87	0.87	[0.80, 0.91]	0.80	0.91	0.89	Balanced Binned Derived Data
	GradientBoostingClassifier	0.87	0.90	0.86	0.93	[0.88, 0.86]	0.88	0.86	[0.79, 0.93]	0.79	0.93	0.89	Balanced Derived Data
	RandomForestClassifier	0.86	1.00	0.86	0.90	[0.84, 0.86]	0.84	0.86	[0.80, 0.90]	0.80	0.90	0.88	Balanced Binned Derived Data
	RandomForestClassifier	0.86	1.00	0.86	0.90	[0.84, 0.86]	0.84	0.86	[0.80, 0.90]	0.80	0.90	0.88	Balanced Derived Data
	RandomForestClassifier	0.85	1.00	0.86	0.89	[0.83, 0.86]	0.83	0.86	[0.79, 0.89]	0.79	0.89	0.87	Stdscaled Balanced Binned Derived Data
	AdaBoostClassifier	0.85	0.86	0.85	0.90	[0.85, 0.85]	0.85	0.85	[0.78, 0.90]	0.78	0.90	0.88	Balanced Derived Data
	RandomForestClassifier	0.84	1.00	0.85	0.87	[0.81, 0.85]	0.81	0.85	[0.78, 0.87]	0.78	0.87	0.86	Minmax scaled Balanced Binned Derived Data

Figure 48 Base Models on different datasets with Accuracy >=0.8

Model	Test Accuracy	Training Accuracy	Precision	Recall	0/1 Precision	0 Precision	1 Precision	0/1 Recall	0 Recall	1 Recall	F1-Score	Dataset
Logistic Regression	0.79	0.78	0.78	0.90	[0.82, 0.78]	0.82	0.78	[0.63, 0.90]	0.63	0.90	0.84	Stdscaled Balanced Binned Derived Data
Logistic Regression	0.79	0.78	0.78	0.90	[0.82, 0.78]	0.82	0.78	[0.63, 0.90]	0.63	0.90	0.84	Minmax scaled Balanced Binned Derived Data
SVM	0.79	0.80	0.79	0.88	[0.79, 0.79]	0.79	0.79	[0.67, 0.88]	0.67	0.88	0.83	Balanced Binned Derived Data
SVM	0.79	0.80	0.79	0.88	[0.79, 0.79]	0.79	0.79	[0.67, 0.88]	0.67	0.88	0.83	Stdscaled Balanced Binned Derived Data
SVM	0.79	0.80	0.79	0.88	[0.79, 0.79]	0.79	0.79	[0.67, 0.88]	0.67	0.88	0.83	Minmax scaled Balanced Binned Derived Data
Logistic Regression	0.79	0.78	0.78	0.90	[0.82, 0.78]	0.82	0.78	[0.63, 0.90]	0.63	0.90	0.84	Balanced Binned Derived Data
Decision Tree Classifier	0.78	1.00	0.82	0.80	[0.72, 0.82]	0.72	0.82	[0.75, 0.80]	0.75	0.80	0.81	Balanced Derived Data
Decision Tree Classifier	0.78	1.00	0.81	0.81	[0.73, 0.81]	0.73	0.81	[0.74, 0.81]	0.74	0.81	0.81	Balanced Binned Derived Data

Figure 49 Base Models on different datasets with Accuracy >0.76

Model	Test Accuracy	Training Accuracy	Precision	Recall	0/1 Precision	0 Precision	1 Precision	0/1 Recall	0 Recall	1 Recall	F1-Score	Dataset	
KNN	0.76	0.82	0.78	0.84	[0.74, 0.78]	0.74	0.78	[0.65, 0.84]	0.65	0.84	0.81	Balanced Binned Derived Data	
KNN	0.76	0.82	0.78	0.84	[0.74, 0.78]	0.74	0.78	[0.65, 0.84]	0.65	0.84	0.81	Stdscaled Balanced Binned Derived Data	
KNN	0.76	0.82	0.78	0.84	[0.74, 0.78]	0.74	0.78	[0.65, 0.84]	0.65	0.84	0.81	Minmax scaled Balanced Binned Derived Data	
Decision Tree Classifier	0.76	1.00	0.82	0.76	[0.69, 0.82]	0.69	0.82	[0.75, 0.76]	0.75	0.76	0.79	Stdscaled Balanced Binned Derived Data	
Decision Tree Classifier	0.76	1.00	0.81	0.78	[0.70, 0.81]	0.70	0.81	[0.74, 0.78]	0.74	0.78	0.79	Minmax scaled Balanced Binned Derived Data	
BernoulliNB	0.76	0.77	0.74	0.90	[0.80, 0.74]	0.80	0.74	[0.55, 0.90]	0.55	0.90	0.81	Balanced Derived Data	Be
BernoulliNB	0.76	0.76	0.79	0.80	[0.71, 0.79]	0.71	0.79	[0.70, 0.80]	0.70	0.80	0.79	Balanced Binned Derived Data	Be
BernoulliNB	0.76	0.76	0.79	0.80	[0.71, 0.79]	0.71	0.79	[0.70, 0.80]	0.70	0.80	0.79	Stdscaled Balanced Binned Derived Data	Be
BernoulliNB	0.76	0.76	0.79	0.80	[0.71, 0.79]	0.71	0.79	[0.70, 0.80]	0.70	0.80	0.79	Minmax scaled Balanced Binned Derived Data	Be
Gaussian NB	0.74	0.72	0.71	0.94	[0.85, 0.71]	0.85	0.71	[0.46, 0.94]	0.46	0.94	0.81	Balanced Binned Derived Data	
Gaussian NB	0.74	0.72	0.71	0.94	[0.85, 0.71]	0.85	0.71	[0.46, 0.94]	0.46	0.94	0.81	Stdscaled Balanced Binned Derived Data	
Gaussian NB	0.74	0.72	0.71	0.94	[0.85, 0.71]	0.85	0.71	[0.46, 0.94]	0.46	0.94	0.81	Minmax scaled Balanced Binned Derived Data	
Gaussian NB	0.71	0.69	0.69	0.93	[0.81, 0.69]	0.81	0.69	[0.40, 0.93]	0.40	0.93	0.79	Balanced Derived Data	
Logistic Regression	0.70	0.68	0.69	0.88	[0.72, 0.69]	0.72	0.69	[0.44, 0.88]	0.44	0.88	0.77	Balanced Derived Data	
SVM	0.70	0.69	0.68	0.94	[0.81, 0.68]	0.81	0.68	[0.37, 0.94]	0.37	0.94	0.79	Balanced Derived Data	
KNN	0.64	0.76	0.68	0.71	[0.57, 0.68]	0.57	0.68	[0.53, 0.71]	0.53	0.71	0.70	Balanced Derived Data	

Figure 50 Base Models on different datasets with Accuracy <= 0.76

Their performance was also compared based on cross validation score using accuracy and ROC AUC as scoring techniques.

	Algorithm	Dataset	ROC AUC Mean	Accuracy Mean
35	GradientBoostingClassifier	Minmax scaled Balanced Binned Derived Data	91.13	85.13
34	GradientBoostingClassifier	Stdscaled Balanced Binned Derived Data	91.12	85.20
33	GradientBoostingClassifier	Balanced Binned Derived Data	91.10	85.33
32	GradientBoostingClassifier	Balanced Derived Data	90.53	84.40
29	AdaBoostClassifier	Balanced Binned Derived Data	90.19	84.47
30	AdaBoostClassifier	Stdscaled Balanced Binned Derived Data	90.19	84.47
31	AdaBoostClassifier	Minmax scaled Balanced Binned Derived Data	90.19	84.47
26	RandomForestClassifier	Stdscaled Balanced Binned Derived Data	89.77	84.47
25	RandomForestClassifier	Balanced Binned Derived Data	89.72	83.93
28	AdaBoostClassifier	Balanced Derived Data	89.62	84.20
24	RandomForestClassifier	Balanced Derived Data	89.39	84.13
27	RandomForestClassifier	Minmax scaled Balanced Binned Derived Data	89.38	84.07
7	SVM	Minmax scaled Balanced Binned Derived Data	85.08	77.67
6	SVM	Stdscaled Balanced Binned Derived Data	85.08	77.67
5	SVM	Balanced Binned Derived Data	85.08	77.67
3	Logistic Regression	Minmax scaled Balanced Binned Derived Data	83.61	77.53
2	Logistic Regression	Stdscaled Balanced Binned Derived Data	83.61	77.53
1	Logistic Regression	Balanced Binned Derived Data	83.61	77.53
21	BernoulliNB	Balanced Binned Derived Data	82.07	75.20
23	BernoulliNB	Minmax scaled Balanced Binned Derived Data	82.07	75.20
22	BernoulliNB	Stdscaled Balanced Binned Derived Data	82.07	75.20
18	Gaussian NB	Stdscaled Balanced Binned Derived Data	80.59	70.67
19	Gaussian NB	Minmax scaled Balanced Binned Derived Data	80.59	70.67
17	Gaussian NB	Balanced Binned Derived Data	80.59	70.67
11	KNN	Minmax scaled Balanced Binned Derived Data	79.57	76.33
10	KNN	Stdscaled Balanced Binned Derived Data	79.57	76.33
9	KNN	Balanced Binned Derived Data	79.57	76.33
15	Decision Tree Classifier	Minmax scaled Balanced Binned Derived Data	77.71	78.13
14	Decision Tree Classifier	Stdscaled Balanced Binned Derived Data	77.18	78.00
13	Decision Tree Classifier	Balanced Binned Derived Data	76.86	77.27
12	Decision Tree Classifier	Balanced Derived Data	76.50	77.27
20	BernoulliNB	Balanced Derived Data	75.48	76.53
16	Gaussian NB	Balanced Derived Data	71.08	68.87
8	KNN	Balanced Derived Data	68.59	64.93
4	SVM	Balanced Derived Data	65.48	69.40
0	Logistic Regression	Balanced Derived Data	61.04	67.67

Figure 51:Base Models with cross validation on different Datasets

Mean accuracy has been compared for different models on each dataset.

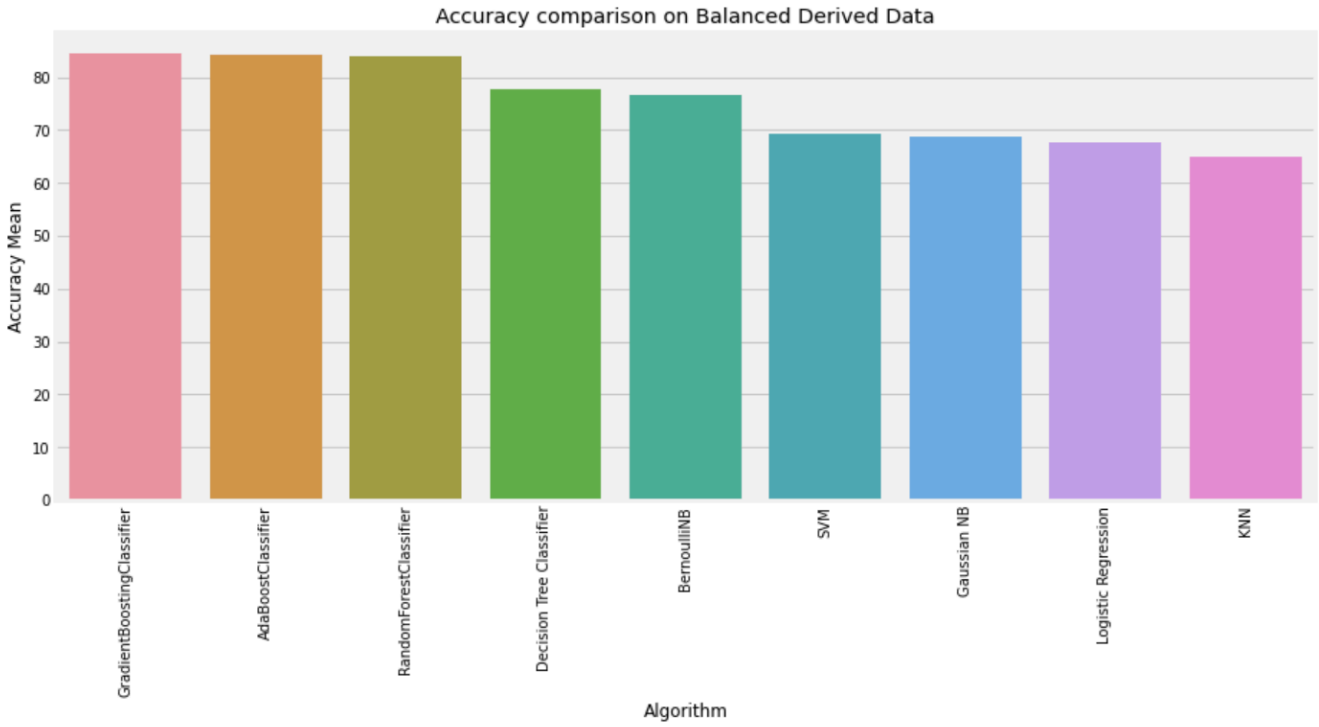


Figure 52:Accuracy comparison on Balanced Derived Data

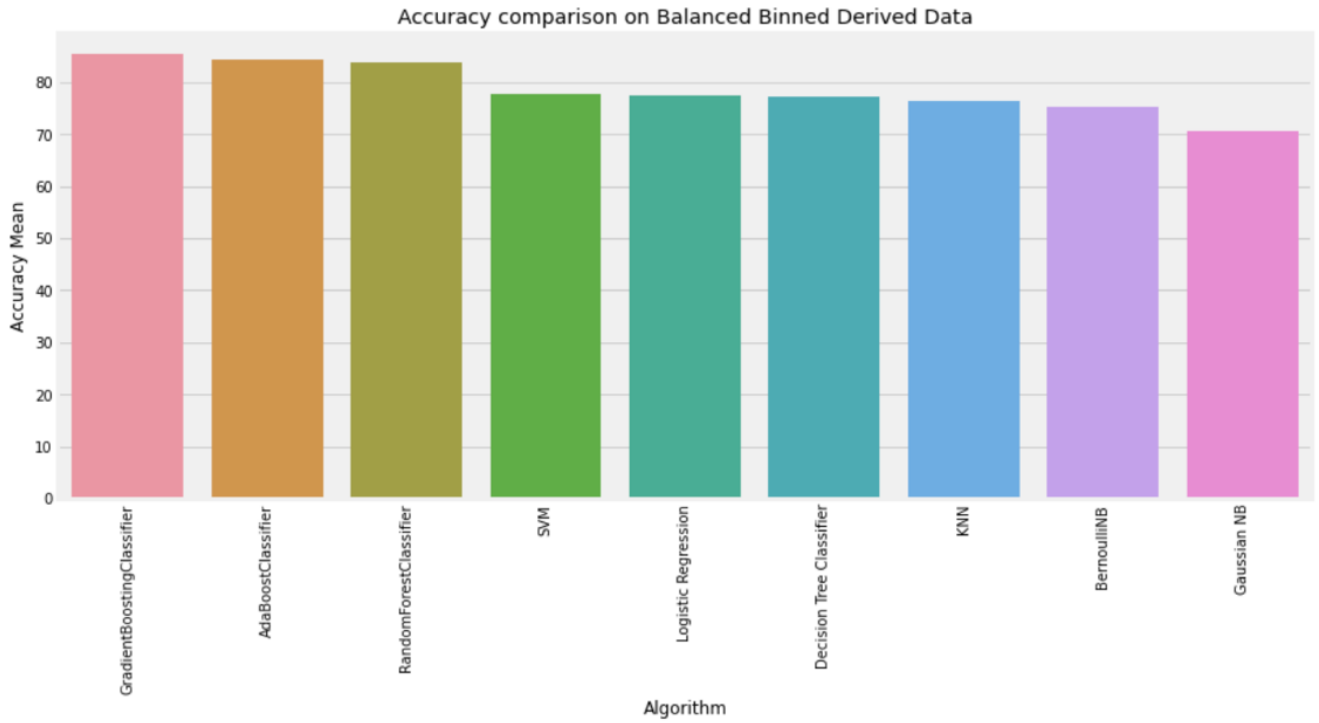


Figure 53:Accuracy comparison on Balanced Binned Derived Data

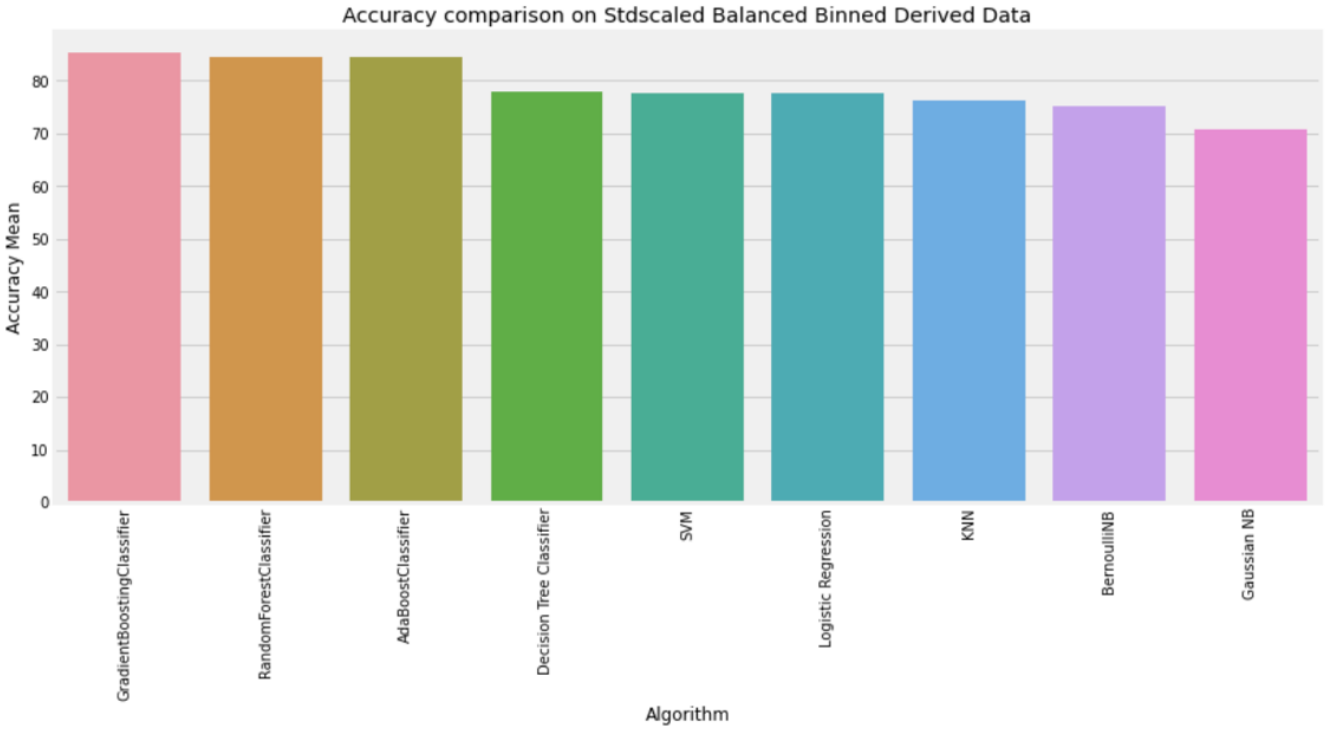


Figure 54:Accuracy comparison on Stdscaled Balanced Binned Derived Data

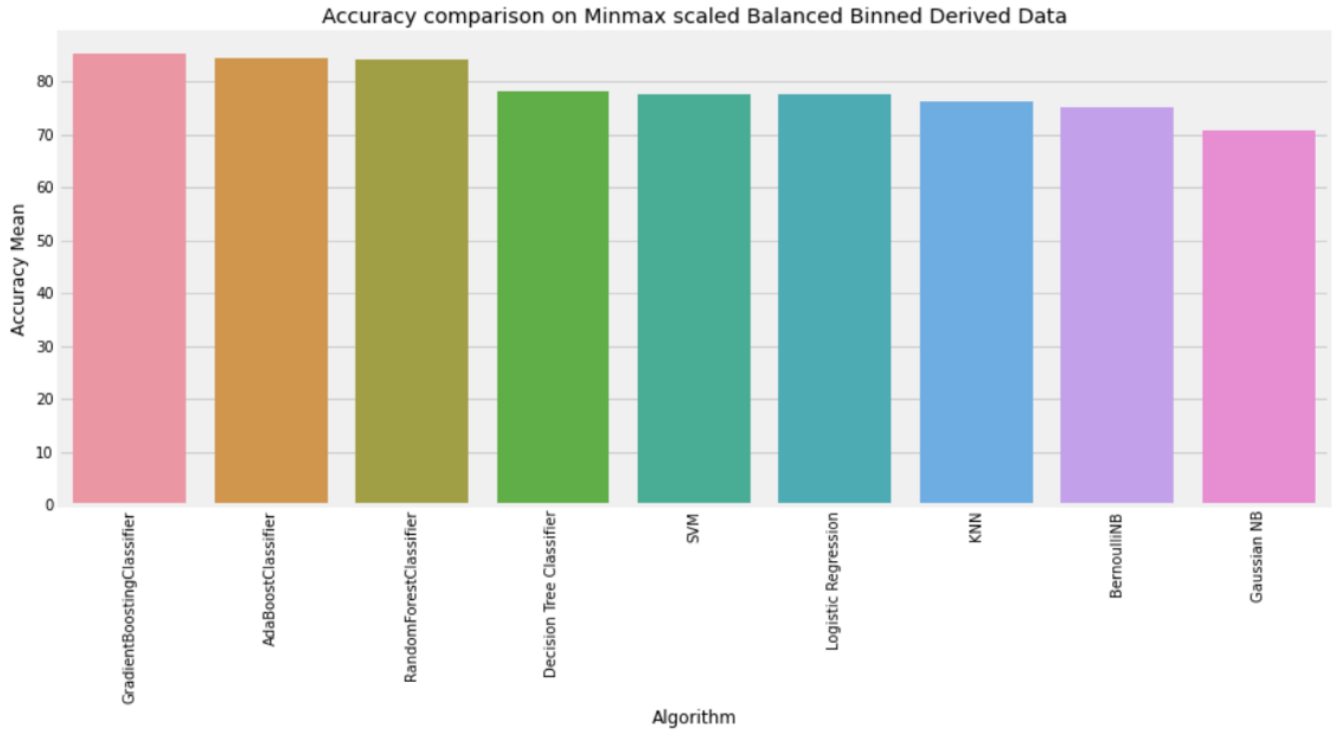


Figure 55:Accuracy comparison on Minmaxscaled Balanced Binned Derived Data

20.2. Model 1: Logistic Regression

Logistic Regression is a "Supervised Machine Learning" algorithm that can be used to build model and apply on a linearly separable dataset. By default, logistic regression can only be used for binary classifications. Since, the target variable (Resign) is binary in nature, we don't need to modify the default algorithm. (Scikit Learn sklearn.linear_model.LogisticRegression, n.d.)

To conclude the model parameters corresponding to the best model performance Randomized Search CV was implemented on different datasets.

```
lr_rand_params_ = {}
for name, data in datasets:
    print('Dataset:',name)
    param = RSCV_LR(data)
    lr_rand_params_[name] = param
```

```
Dataset: Balanced Derived Data
Best Solver: liblinear
Best C: 0.16099999999999998
Best penalty: l1
Best max_iter: 79
Best tol: 0.05099999999999999
Best Accuracy Score: 0.764

{'C': 0.16099999999999998, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ra
tio': None, 'max_iter': 79, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l1', 'random_state': 12, 'solver': 'li
blinear', 'tol': 0.05099999999999999, 'verbose': 0, 'warm_start': False}
-----
Dataset: Balanced Binned Derived Data
Best Solver: lbfgs
Best C: 1.3809999999999998
Best penalty: l2
Best max_iter: 11
Best tol: 0.9909999999999999
Best Accuracy Score: 0.7753333333333334

{'C': 1.3809999999999998, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_rat
io': None, 'max_iter': 11, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l2', 'random_state': 12, 'solver': 'lbf
gs', 'tol': 0.9909999999999999, 'verbose': 0, 'warm_start': False}
-----
Dataset: Stdscaled Balanced Binned Derived Data
Best Solver: liblinear
Best C: 1.1309999999999998
Best penalty: l1
Best max_iter: 80
Best tol: 0.030999999999999996
Best Accuracy Score: 0.776

{'C': 1.1309999999999998, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_rat
io': None, 'max_iter': 80, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l1', 'random_state': 12, 'solver': 'lib
linear', 'tol': 0.030999999999999996, 'verbose': 0, 'warm_start': False}
-----
Dataset: Minmax scaled Balanced Binned Derived Data
Best Solver: liblinear
Best C: 1.6909999999999996
Best penalty: l1
Best max_iter: 42
Best tol: 0.16099999999999998
Best Accuracy Score: 0.7753333333333333

{'C': 1.6909999999999996, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_rat
io': None, 'max_iter': 42, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l1', 'random_state': 12, 'solver': 'lib
linear', 'tol': 0.16099999999999998, 'verbose': 0, 'warm_start': False}
```

Figure 56:Logistic Regression hyperparamter with RandomizedSearchCV

20.2.1. Logistic Regression Model Parameters

- Solver – This parameter represents which algorithm to use in the optimization problem.
    - **liblinear** (A library for large linear classification) – It's a linear classification

that supports logistic regression and linear support vector machines.

- **sag (Stochastic Average Gradient)** – It optimizes the sum of finite number of smooth convex functions.
- **lbfgs (Limited memory Broyden Fletcher Goldfarb Shanno)** – It approximates the second derivative matrix updates with gradient evaluations and stores only the last few updates and thus saves memory.

- C – It represents the inverse of regularization strength, which must always be a positive float.
- Penalty – This parameter is used to specify the norm (L1 or L2) used in penalization (regularization).

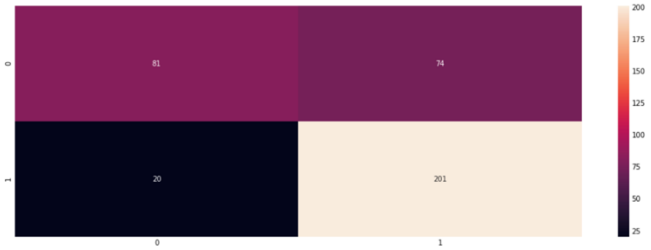
20.2.2. Logistic Regression plots for various datasets

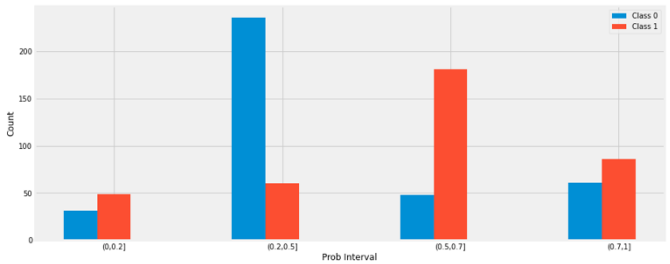
Best parameters concluded from Randomised Search CV on Logistic regression were used for model building. (Scikit Learn sklearn.linear_model.LogisticRegression, n.d.) (Data Analystics Learning Curves Explained with Python Sklearn Example, n.d.)

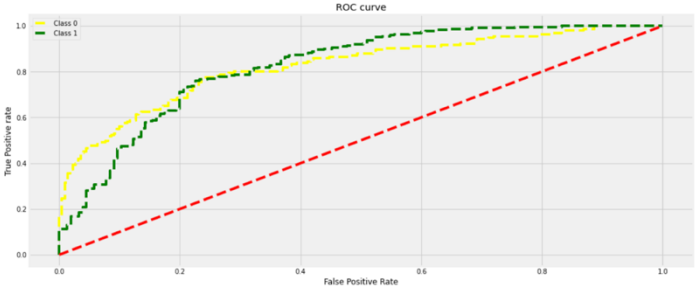
Dataset: Balanced Derived Data

	precision	recall	f1-score	support
0	0.80	0.52	0.63	155
1	0.73	0.91	0.81	221
accuracy			0.75	376
macro avg	0.77	0.72	0.72	376
weighted avg	0.76	0.75	0.74	376

Accuracy:0.75
Precision:0.73
Recall:0.75
F1-Score:0.74







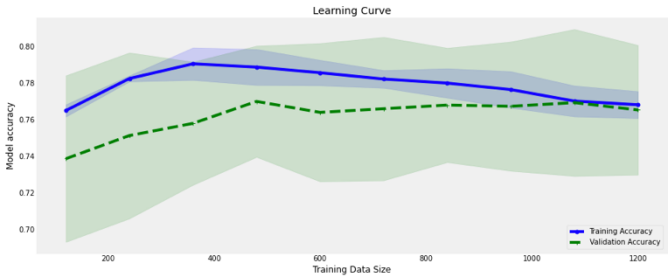


Figure 57:Logistic Regression model on Balanced Derived Dataset

Dataset: Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.83	0.63	0.71	155
1	0.78	0.91	0.84	221
accuracy			0.79	376
macro avg	0.80	0.77	0.78	376
weighted avg	0.80	0.79	0.79	376

Accuracy:0.79
Precision:0.78
Recall:0.79
F1-Score:0.79

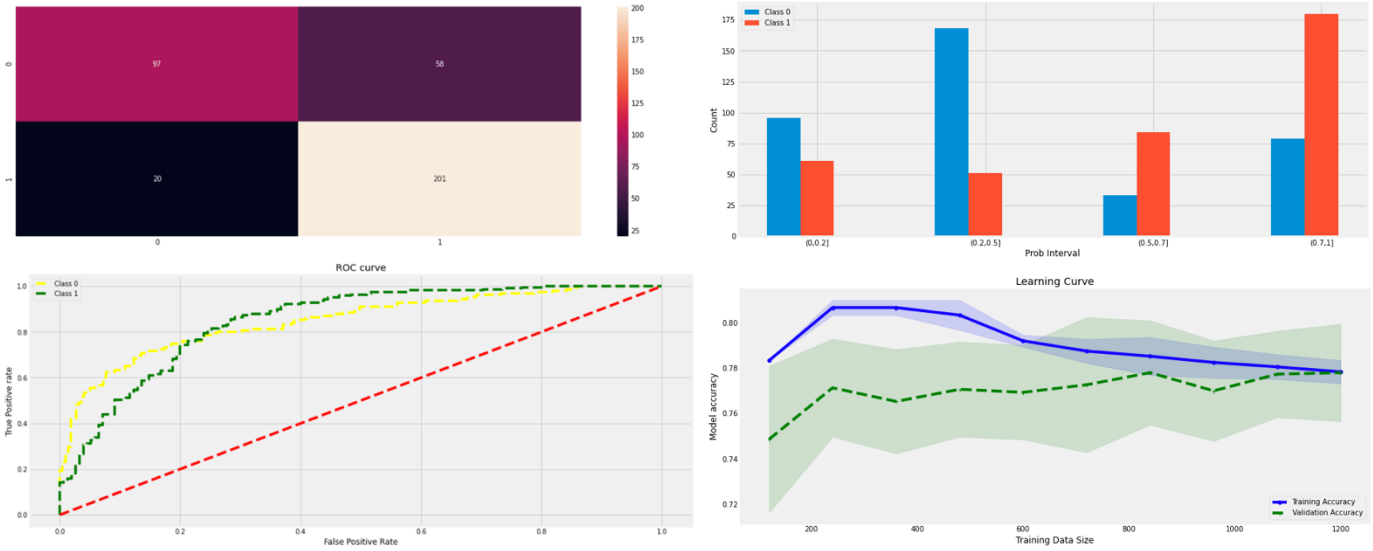


Figure 58: Logistic Regression model on Balanced Binned Derived Dataset

Dataset: Stdscaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.86	0.62	0.72	155
1	0.78	0.93	0.85	221
accuracy			0.80	376
macro avg	0.82	0.77	0.78	376
weighted avg	0.81	0.80	0.79	376

Accuracy:0.8
Precision:0.78
Recall:0.80
F1-Score:0.79

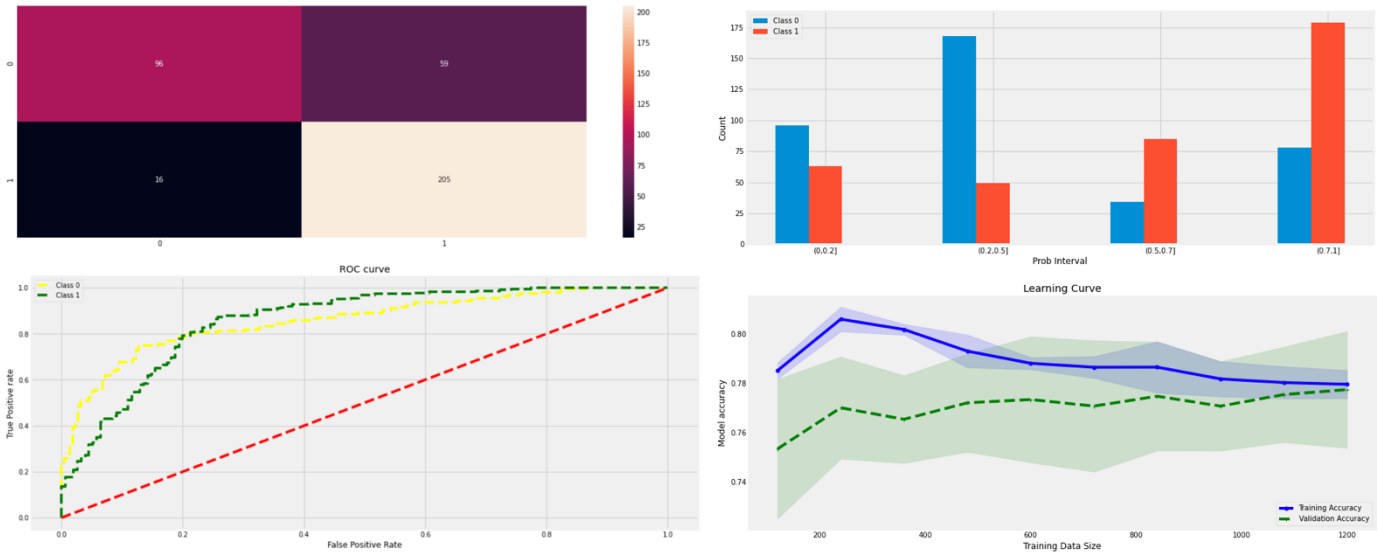


Figure 59:Logistic Regression model on Stdscaled Balanced Binned Derived Dataset

Dataset: Minmax scaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.86	0.62	0.72	155
1	0.78	0.93	0.85	221
accuracy			0.80	376
macro avg	0.82	0.77	0.78	376
weighted avg	0.81	0.80	0.79	376

Accuracy:0.8
Precision:0.78
Recall:0.80
F1-Score:0.79

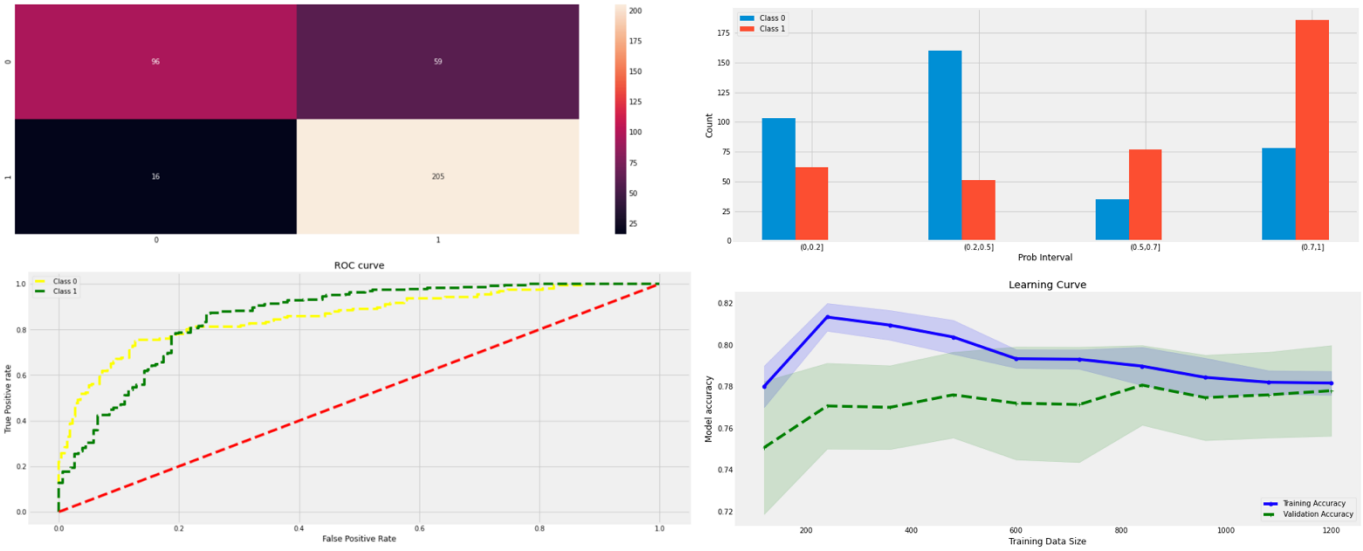


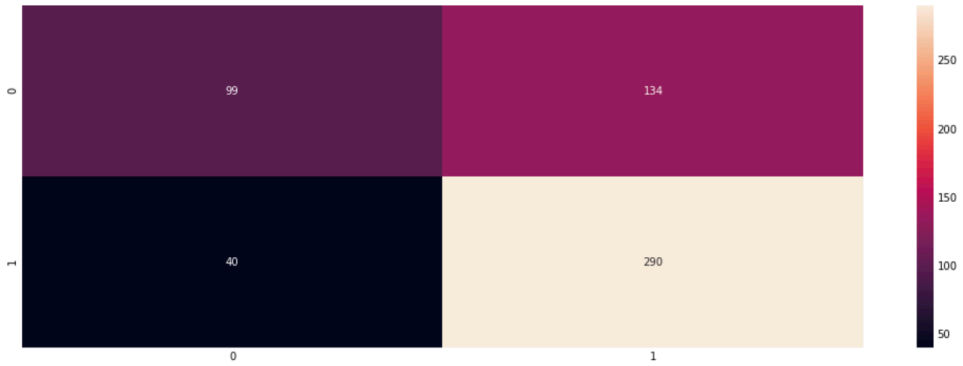
Figure 60:Logistic Regression model on Minmaxscaled Balanced Binned Derived Dataset

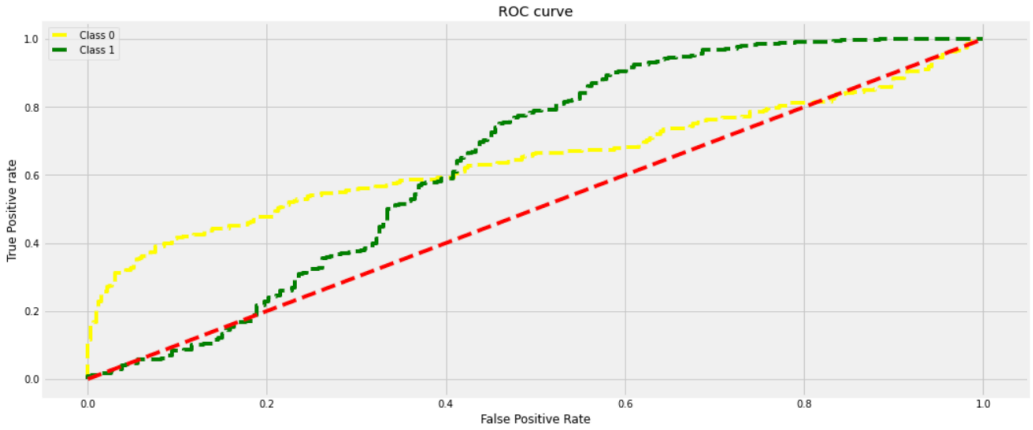
20.2.3. Logistic Regression with RFECV

(Scikit Learn sklearn.feature_selection.RFECV, n.d.)

Dataset: Balanced Derived Data

	precision	recall	f1-score	support
0	0.71	0.42	0.53	233
1	0.68	0.88	0.77	330
accuracy			0.69	563
macro avg	0.70	0.65	0.65	563
weighted avg	0.70	0.69	0.67	563





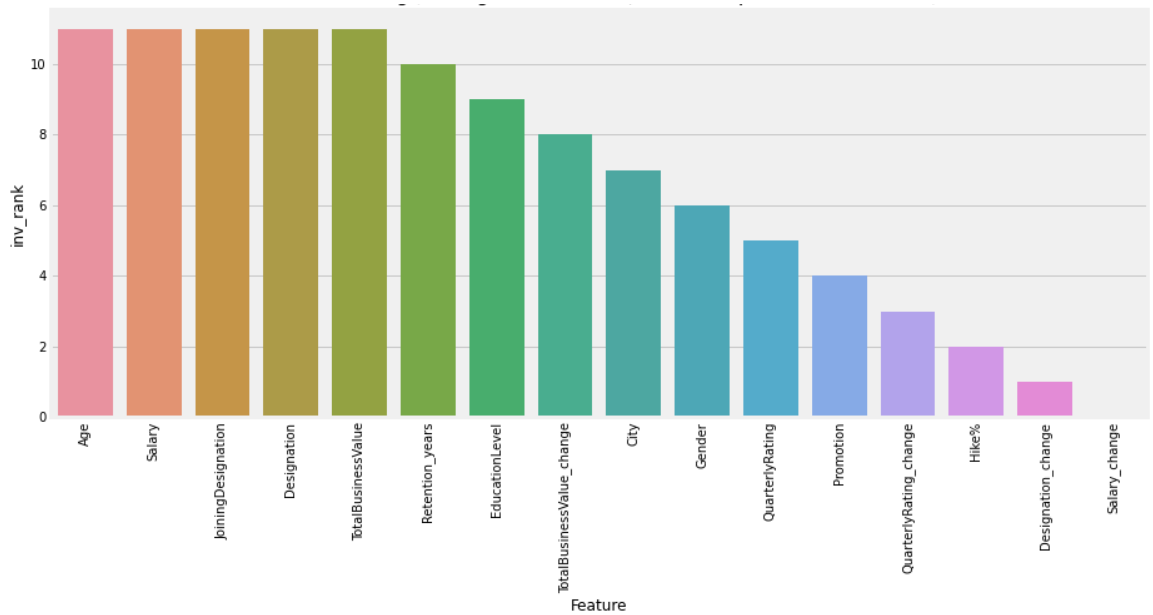
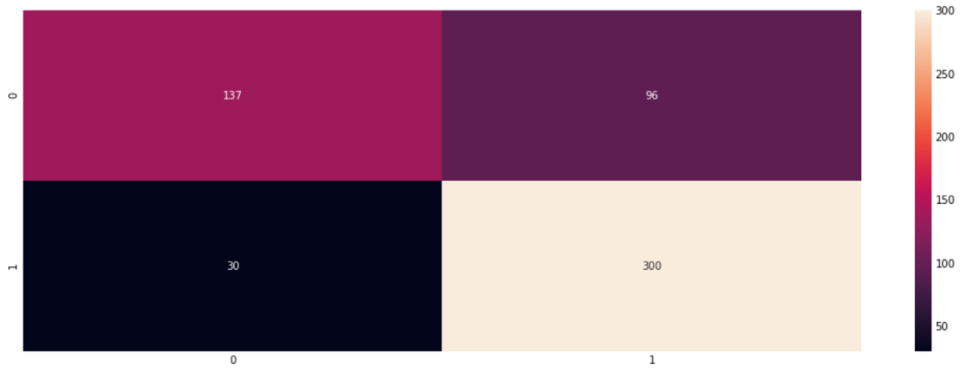
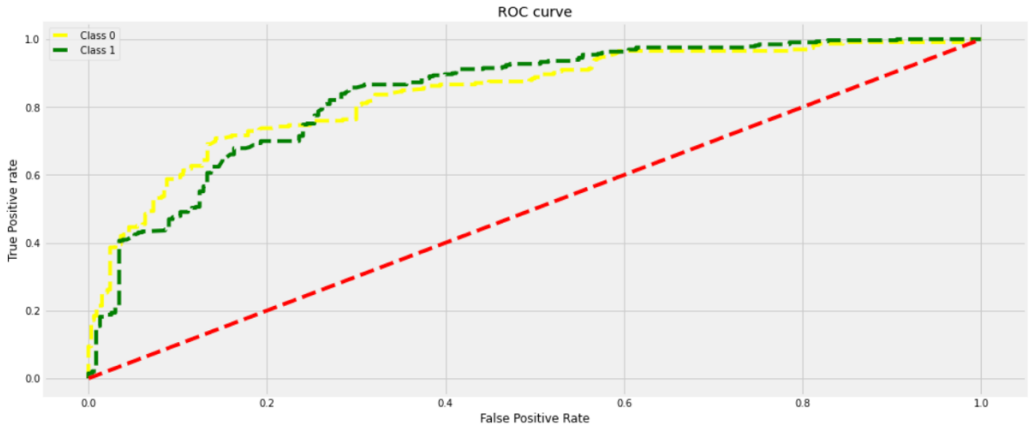


Figure 61: Logistic Regression model with RFECV on Balanced Derived Dataset

Dataset: Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.82	0.59	0.68	233
1	0.76	0.91	0.83	330
accuracy			0.78	563
macro avg	0.79	0.75	0.76	563
weighted avg	0.78	0.78	0.77	563





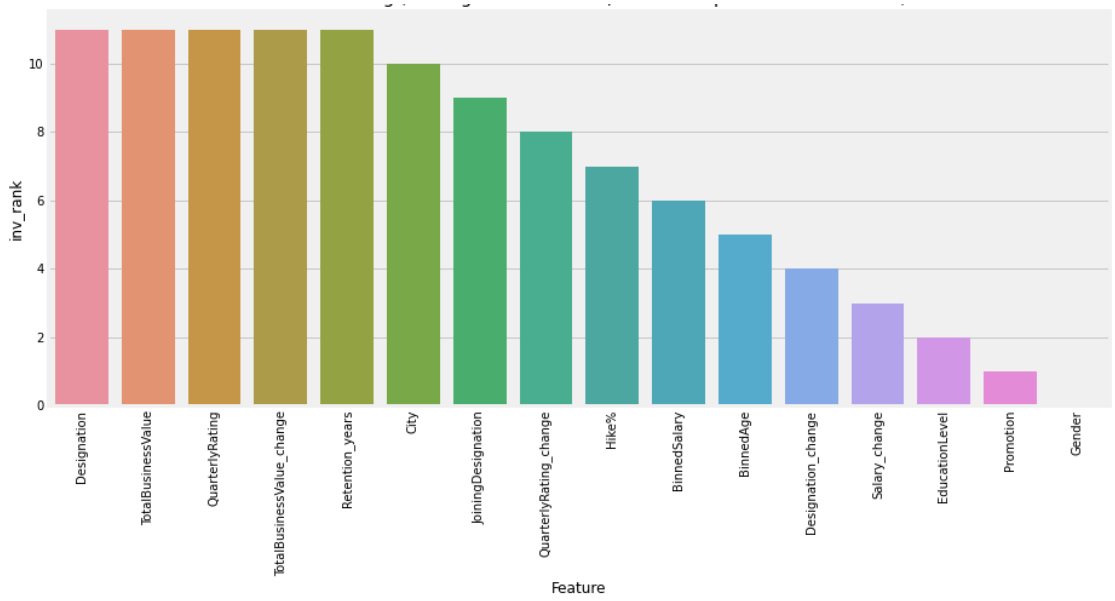
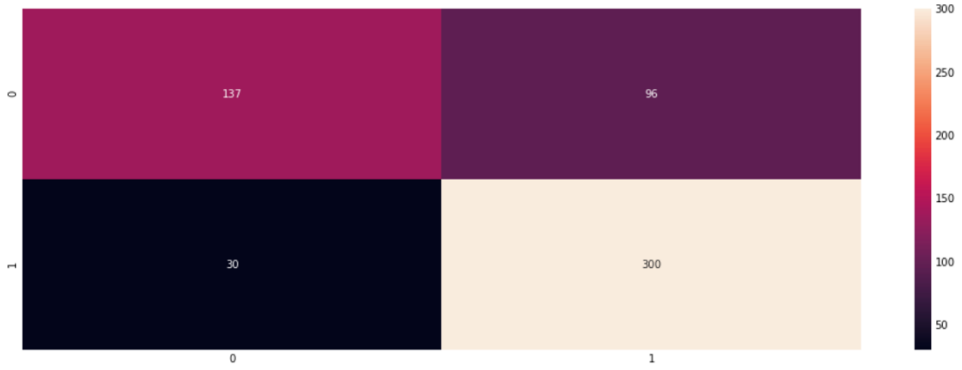
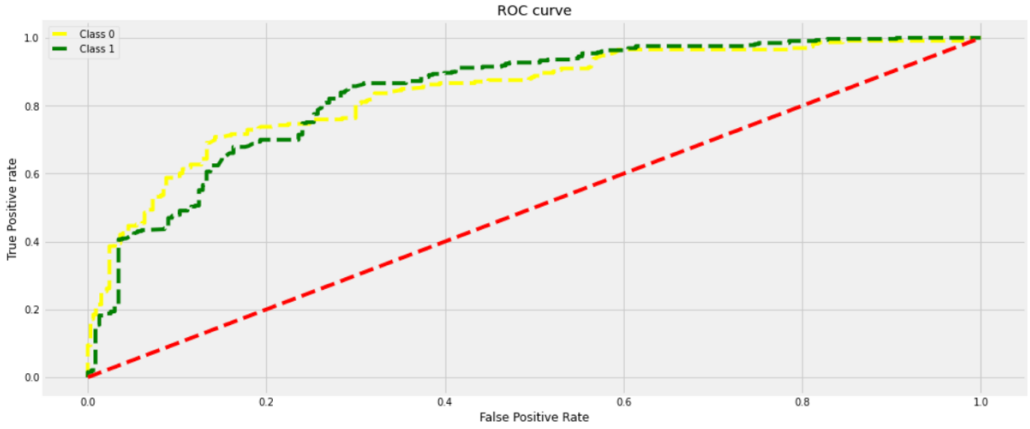


Figure 62:Logistic Regression model with RFECV on Balanced Binned Derived Dataset

Dataset: Stdscaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.82	0.59	0.68	233
1	0.76	0.91	0.83	330
accuracy			0.78	563
macro avg	0.79	0.75	0.76	563
weighted avg	0.78	0.78	0.77	563





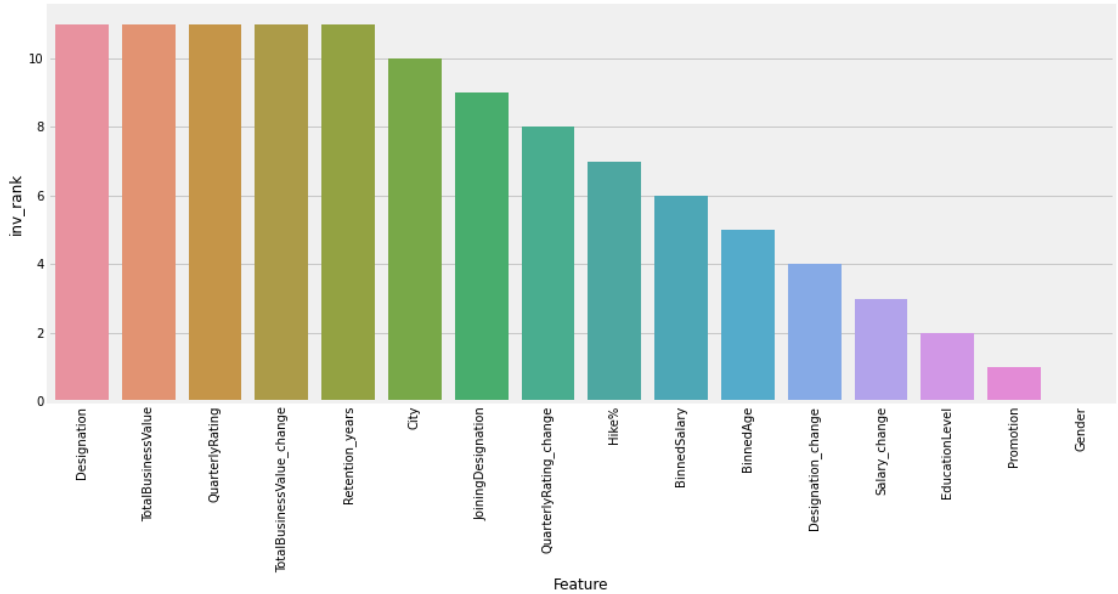
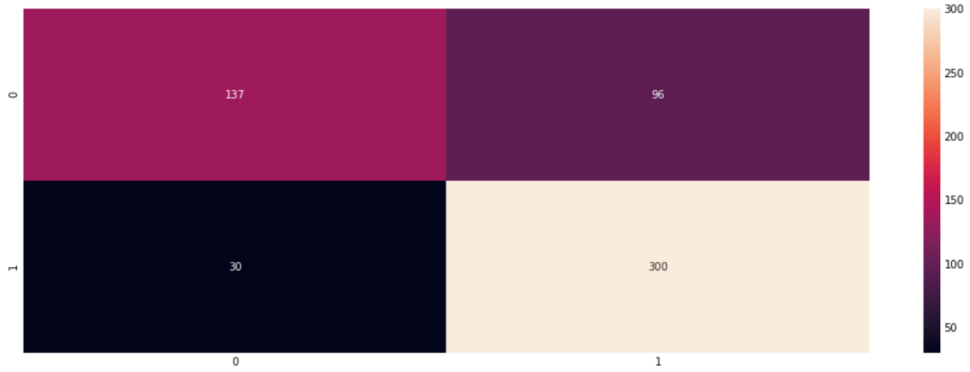
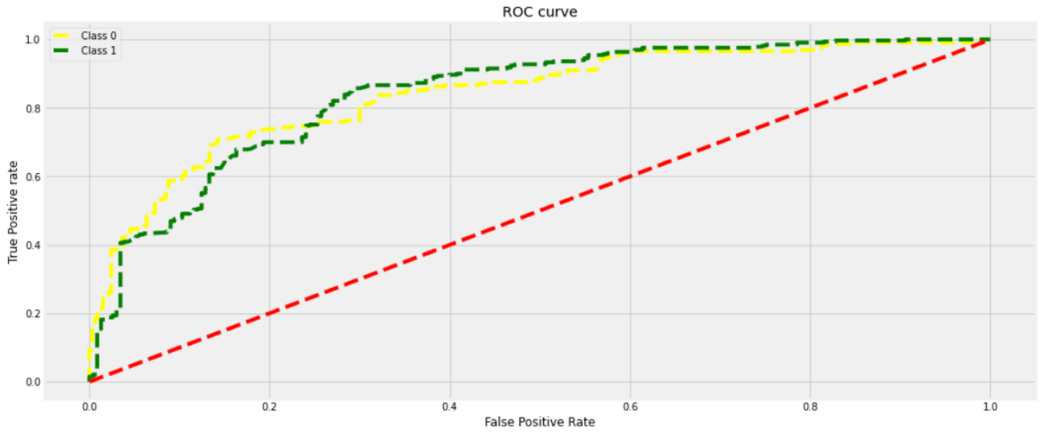


Figure 63:Logistic Regression model with RFECV on Stdscaled Balanced Binned Derived Dataset

```
Dataset: Minmax scaled Balanced Binned Derived Data
              precision    recall  f1-score   support

           0       0.82      0.59      0.68       233
           1       0.76      0.91      0.83       330

    accuracy                           0.78       563
   macro avg       0.79      0.75      0.76       563
weighted avg       0.78      0.78      0.77       563
```





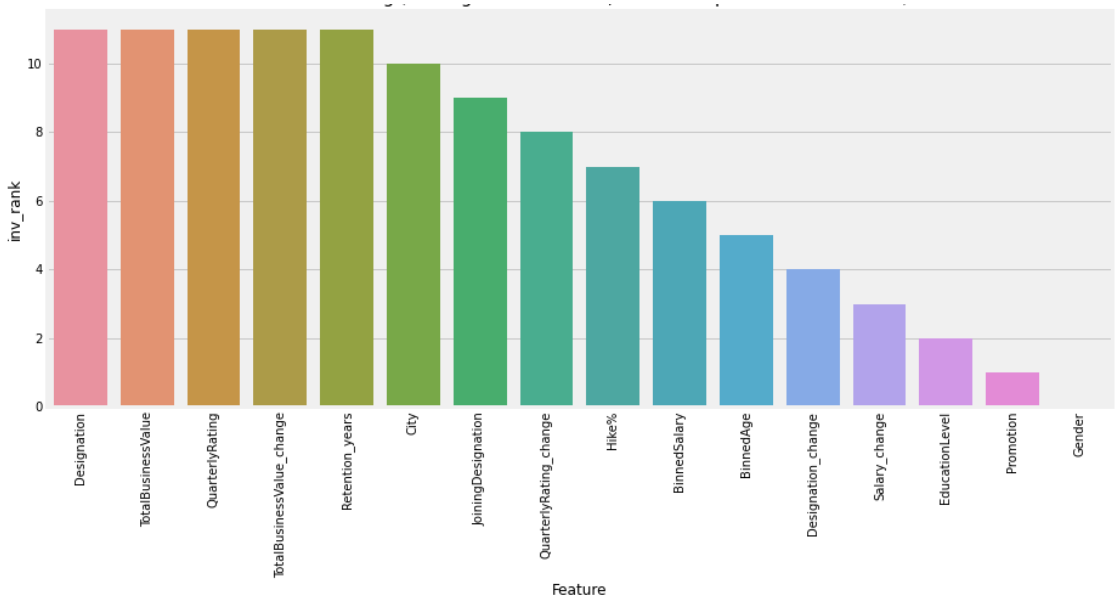


Figure 64:Logistic Regression model with RFECV on Minmaxscaled Balanced Binned Derived Dataset

20.3. **Model 2: Decision Tree**

Decision Trees are a type of "Supervised Machine Learning" algorithm where the data is continuously split according to certain parameters. A Decision Tree is a flowchart like structure where each internal node is indicates a test on a feature and all the leaf nodes are class labels (resulted from decision making on each internal node) and branches represents combination of features that lead to those class labels. (Scikit Learn sklearn.tree.DecisionTreeClassifier, n.d.)

To conclude the model parameters corresponding to the best model performance Randomized Search CV was implemented.

```
dt_rand_params_={}
for name, data in datasets:
    print('Dataset:',name)
    param = RSCV_DT(data)
    dt_rand_params_[name] = param
```

```
Dataset: Balanced Derived Data
Best criterion: entropy
Best max_depth: 2
Best max_features: log2
Best splitter: best
Best min_samples_leaf: 19
Best random_state: 12
Best max_leaf_nodes: 9
Best Accuracy Score: 0.7486666666666666

{'ccp_alpha': 0.040999999999999995, 'class_weight': None, 'criterion': 'entropy', 'max_depth': 2, 'max_features': 'lo
g2', 'max_leaf_nodes': 9, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 19, 'min_samples_split': 2, 'min_weight_f
raction_leaf': 0.0, 'random_state': 12, 'splitter': 'best'}
-----
Dataset: Balanced Binned Derived Data
Best criterion: gini
Best max_depth: 10
Best max_features: log2
Best splitter: best
Best min_samples_leaf: 1
Best random_state: 12
Best max_leaf_nodes: 4
Best Accuracy Score: 0.7193333333333334

{'ccp_alpha': 0.011, 'class_weight': None, 'criterion': 'gini', 'max_depth': 10, 'max_features': 'log2', 'max_leaf_no
des': 4, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.
0, 'random_state': 12, 'splitter': 'best'}
-----
Dataset: Stdscaled Balanced Binned Derived Data
Best criterion: entropy
Best max_depth: 2
Best max_features: auto
Best splitter: random
Best min_samples_leaf: 8
Best random_state: 12
Best max_leaf_nodes: 16
Best Accuracy Score: 0.5866666666666667

{'ccp_alpha': 1.2509999999999997, 'class_weight': None, 'criterion': 'entropy', 'max_depth': 2, 'max_features': 'aut
o', 'max_leaf_nodes': 16, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 8, 'min_samples_split': 2, 'min_weight_fr
action_leaf': 0.0, 'random_state': 12, 'splitter': 'random'}
-----
Dataset: Minmax scaled Balanced Binned Derived Data
Best criterion: gini
Best max_depth: 7
Best max_features: log2
Best splitter: random
Best min_samples_leaf: 6
Best random_state: 12
Best max_leaf_nodes: 11
Best Accuracy Score: 0.6326666666666667

{'ccp_alpha': 0.020999999999999998, 'class_weight': None, 'criterion': 'gini', 'max_depth': 7, 'max_features': 'log
2', 'max_leaf_nodes': 11, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 6, 'min_samples_split': 2, 'min_weight_fr
action_leaf': 0.0, 'random_state': 12, 'splitter': 'random'}
-----
```

Figure 65: Decision Tree hyperparameter with RandomizedSearchCV

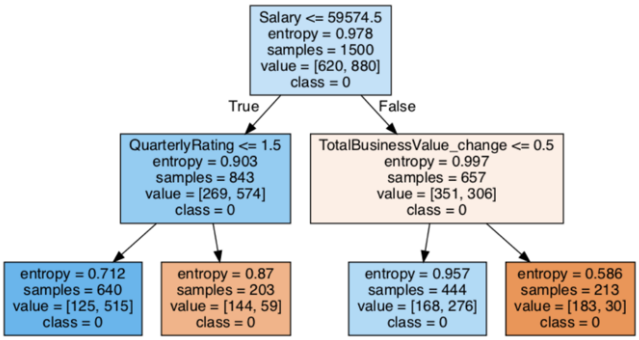
20.3.1. Decision Tree Model Parameters

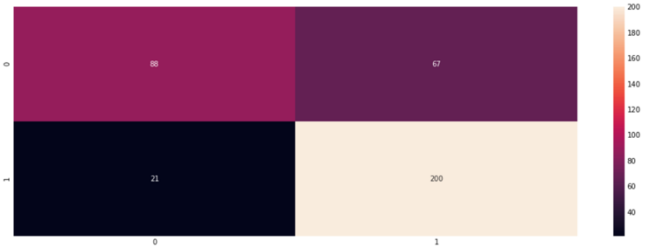
- **Criterion** – The function to measure the quality of the split.
- **Splitter** – The strategy used to choose the split at each node.
- **Max Depth** – The maximum depth of the tree.
- **Max Features** – The number of features to be considered when looking for the best split. If "auto" then max_features = sqrt(n_features) and if "sqrt" then max_features = sqrt(n_features)

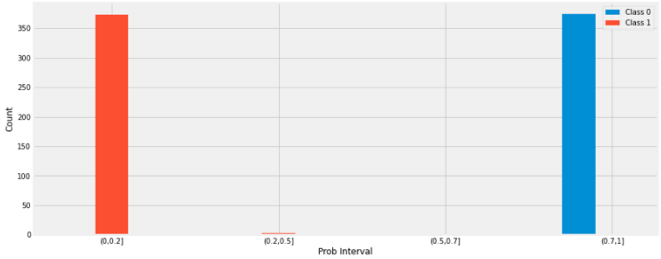
20.3.2. Decision Tree plots for various datasets

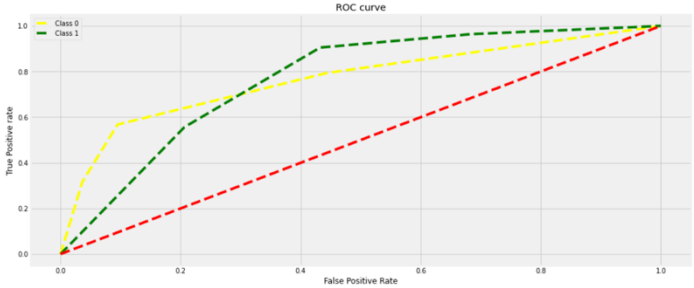
Best parameters concluded from Randomised Search CV on Decision Tree were used for model building. (Scikit Learn sklearn.tree.DecisionTreeClassifier, n.d.) (Data Analystics Learning Curves Explained with Python Sklearn Example, n.d.)

```
Dataset:  Balanced Derived Data
Training done..

Calculating Accuracy..
Accuracy:0.77
Precision:0.75
Recall:0.77
F1-Score:0.76
              precision    recall  f1-score   support

           0       0.81      0.57      0.67       155
           1       0.75      0.90      0.82       221

    accuracy                           0.77       376
   macro avg       0.78      0.74      0.74       376
weighted avg       0.77      0.77      0.76       376
```









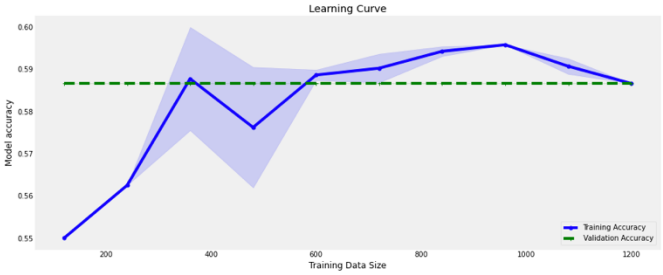
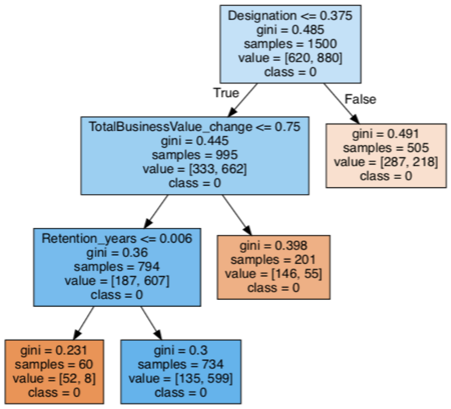
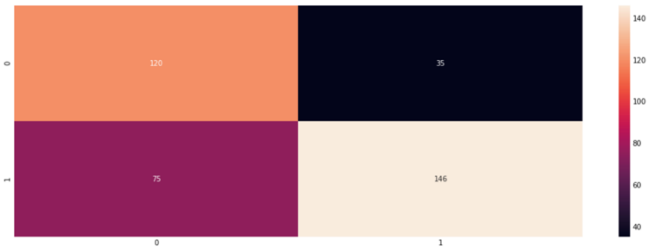
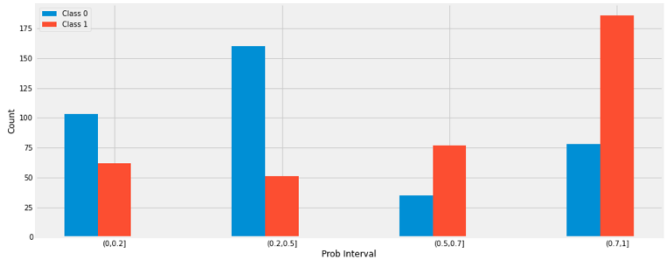


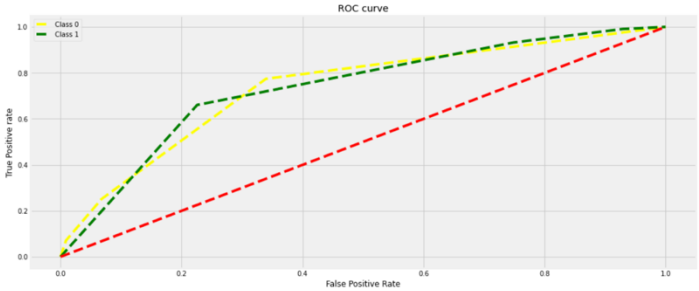
Figure 66: Decision Tree model on Balanced Derived Data

```
Dataset:  Balanced Binned Derived Data
Training done..

Calculating Accuracy..
Accuracy:0.71
Precision:0.81
Recall:0.71
F1-Score:0.71
              precision    recall  f1-score   support

           0       0.62      0.77      0.69       155
           1       0.81      0.66      0.73       221

    accuracy                           0.71       376
   macro avg       0.71      0.72      0.71       376
weighted avg       0.73      0.71      0.71       376
```









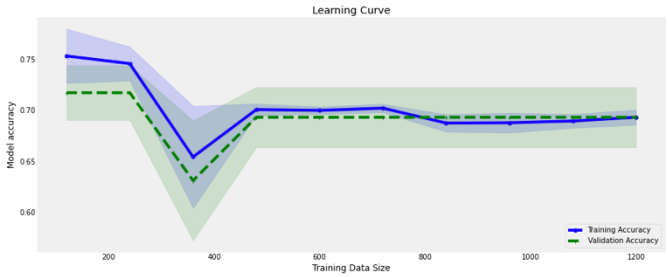
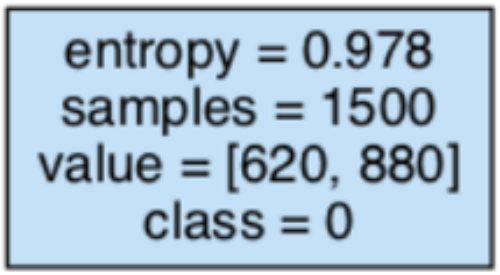


Figure 67:Decision Tree model on Balanced Binned Derived Data

```
Dataset:  Stdscaled Balanced Binned Derived Data
Training done..

Calculating Accuracy..
Accuracy:0.59
Precision:0.59
Recall:0.59
F1-Score:0.44
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       155
           1       0.59      1.00      0.74       221

    accuracy                           0.59       376
   macro avg       0.29      0.50      0.37       376
weighted avg       0.35      0.59      0.44       376
```



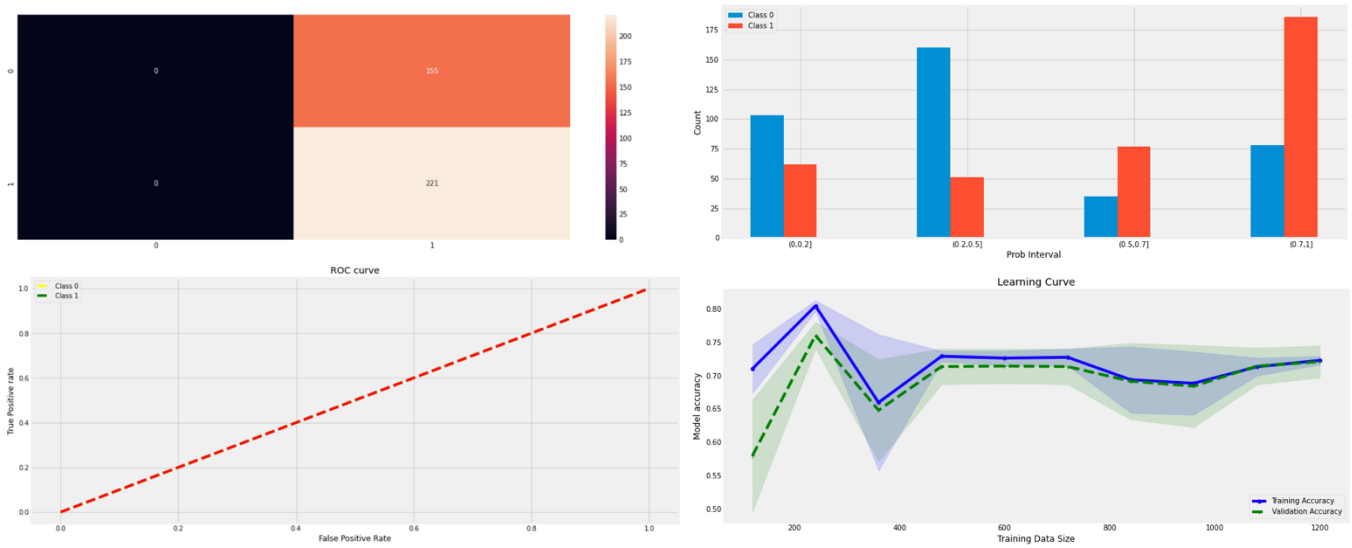
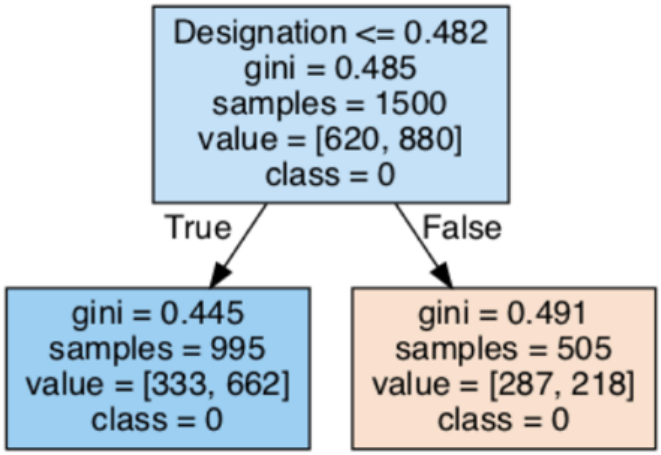


Figure 68:Decision Tree model on Stdscaled Balanced Binned Derived Data

```
Dataset:  Minmax scaled Balanced Binned Derived Data
Training done..

Calculating Accuracy..
Accuracy:0.64
Precision:0.69
Recall:0.64
F1-Score:0.64
              precision    recall  f1-score   support

           0       0.57      0.52      0.55       155
           1       0.69      0.73      0.71       221

    accuracy                           0.64       376
   macro avg       0.63      0.63      0.63       376
weighted avg       0.64      0.64      0.64       376
```



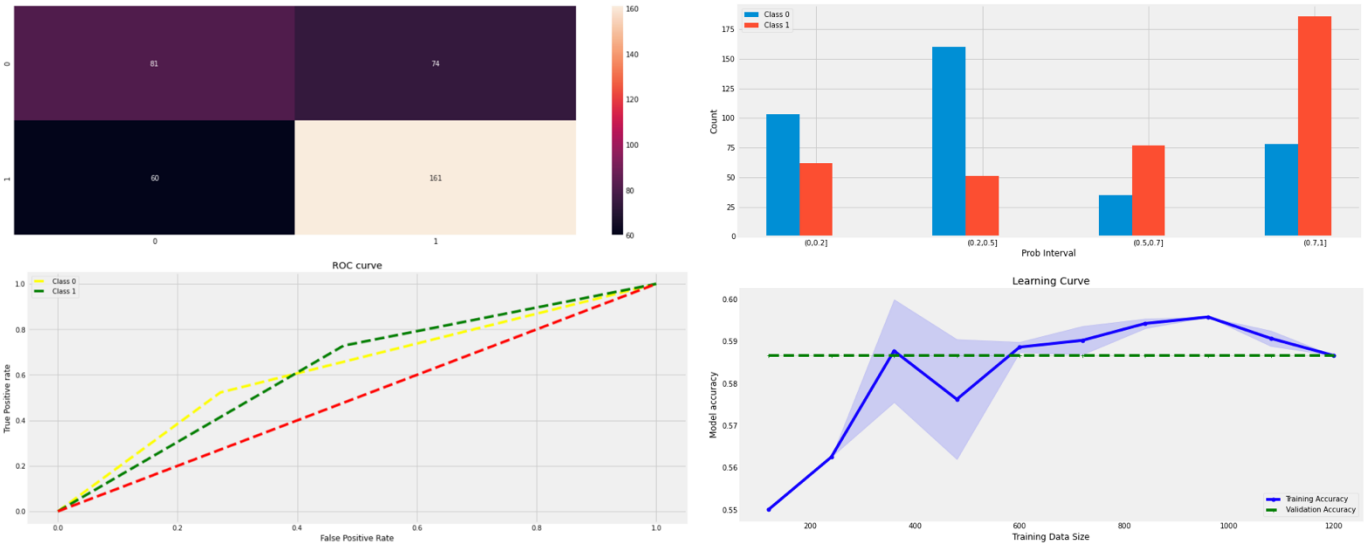
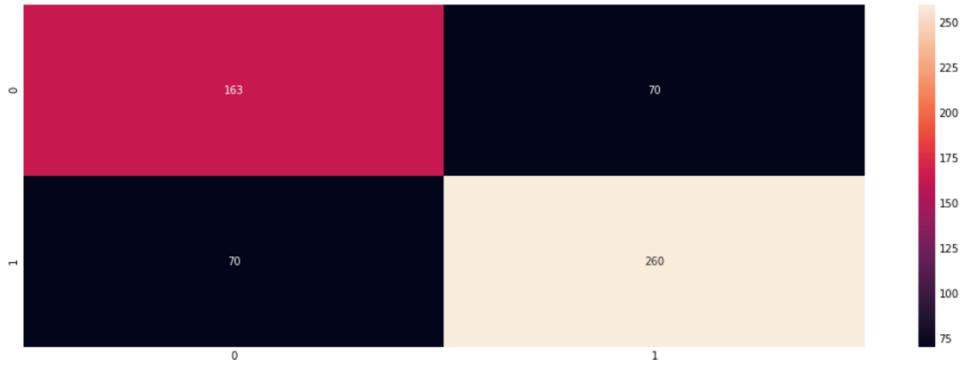


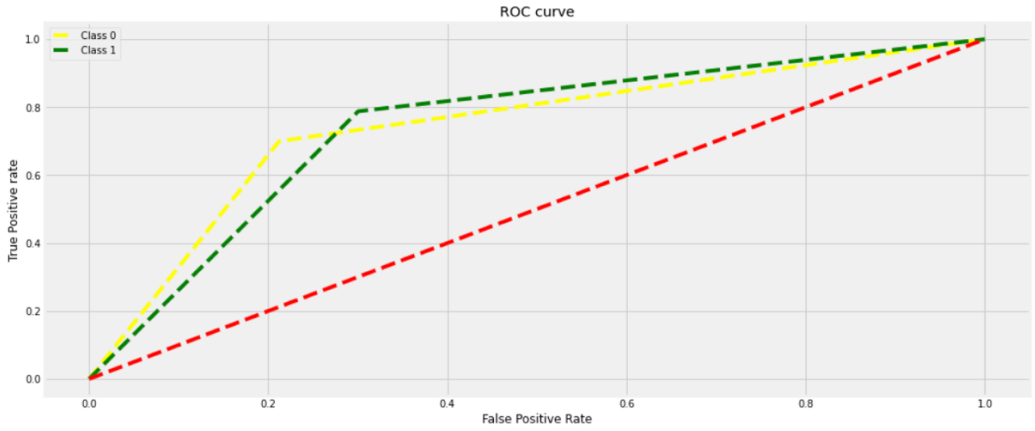
Figure 69:Decision Tree model on Minmaxscale Balanced Binned Derived Data

20.3.3. Decision Tree with RFECV
(Scikit Learn sklearn.feature_selection.RFECV, n.d.)

Dataset: Balanced Derived Data

	precision	recall	f1-score	support
0	0.70	0.70	0.70	233
1	0.79	0.79	0.79	330
accuracy			0.75	563
macro avg	0.74	0.74	0.74	563
weighted avg	0.75	0.75	0.75	563





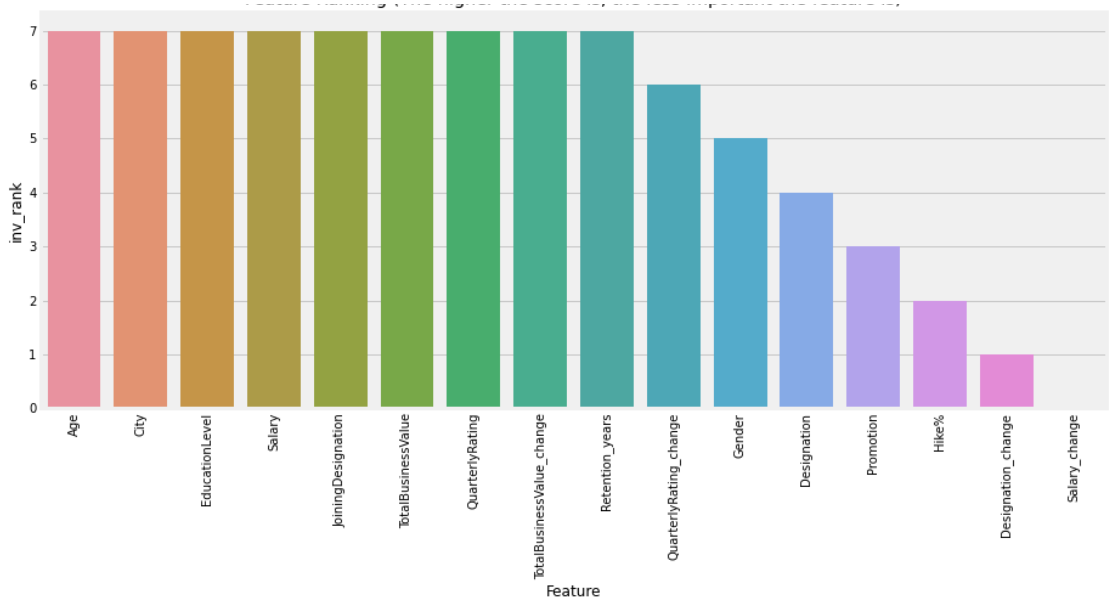
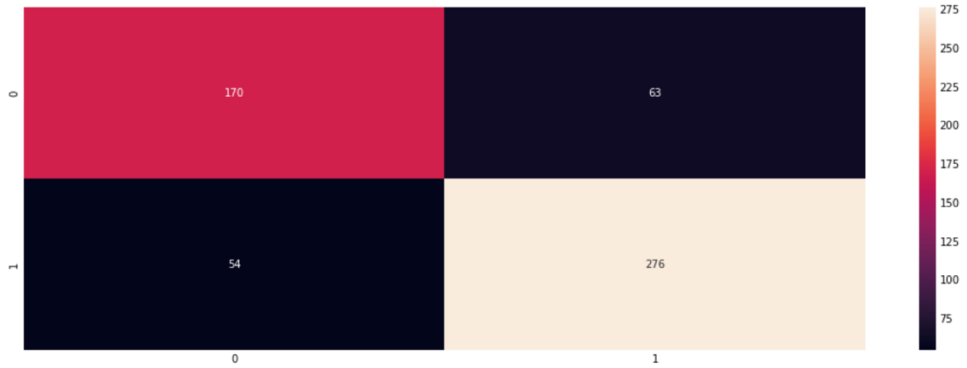
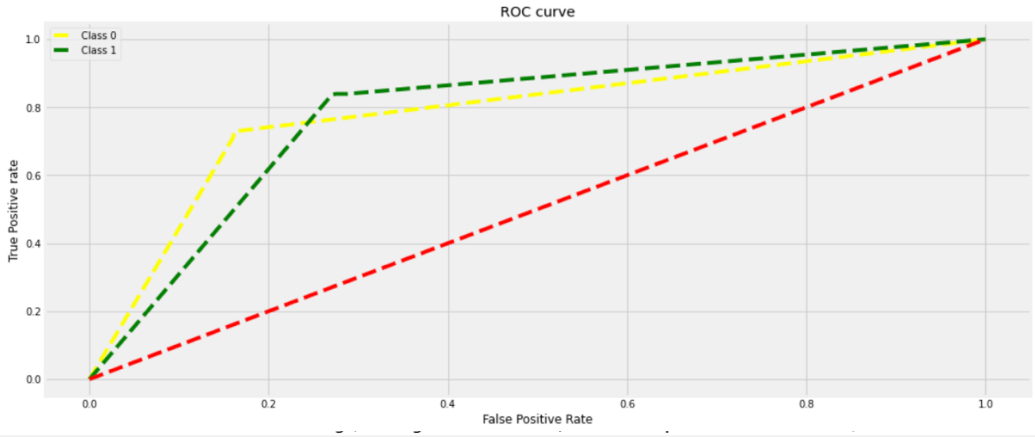


Figure 70:Decision Tree model with RFECV on Balanced Derived Dataset

Dataset: Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.76	0.73	0.74	233
1	0.81	0.84	0.83	330
accuracy			0.79	563
macro avg	0.79	0.78	0.78	563
weighted avg	0.79	0.79	0.79	563





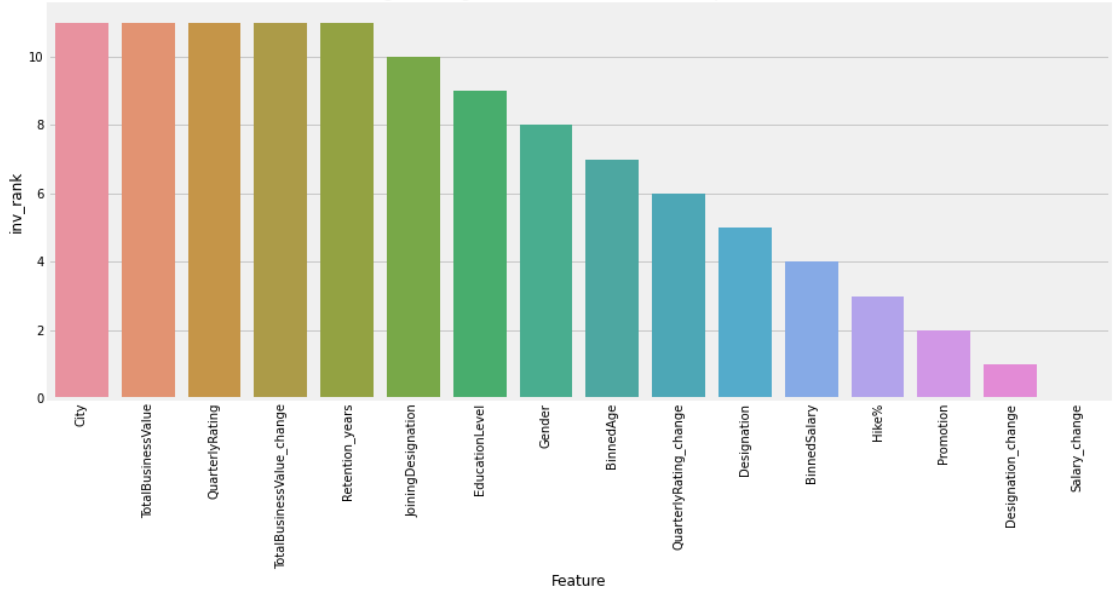
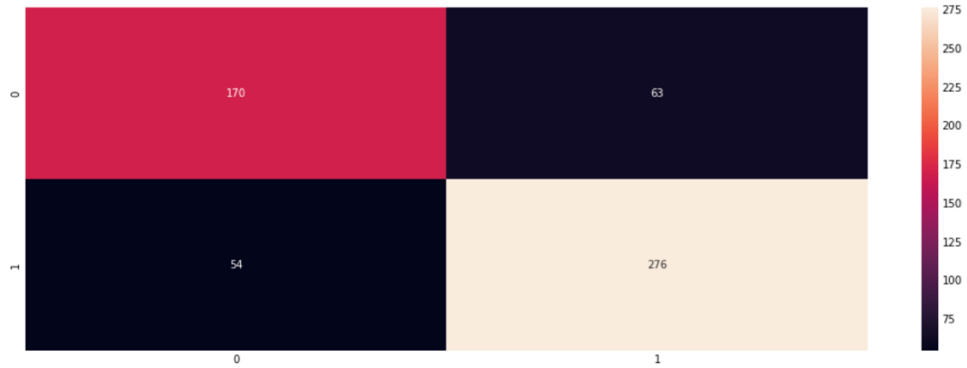
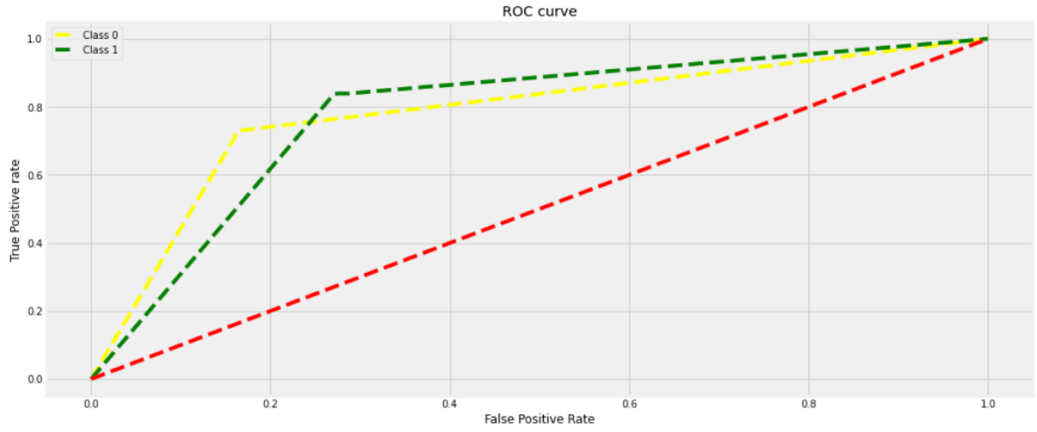


Figure 71:Decision Tree model with RFECV on Balanced Binned Derived Dataset

```
Dataset: Stdscaled Balanced Binned Derived Data
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       233
           1       0.81      0.84      0.83       330

    accuracy                           0.79       563
   macro avg       0.79      0.78      0.78       563
weighted avg       0.79      0.79      0.79       563
```





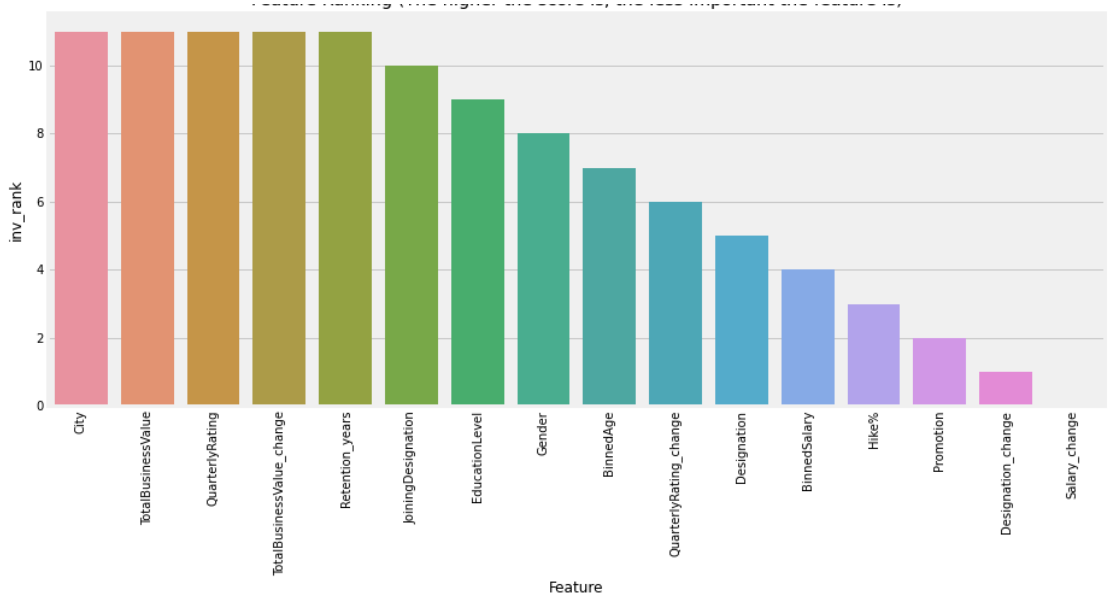
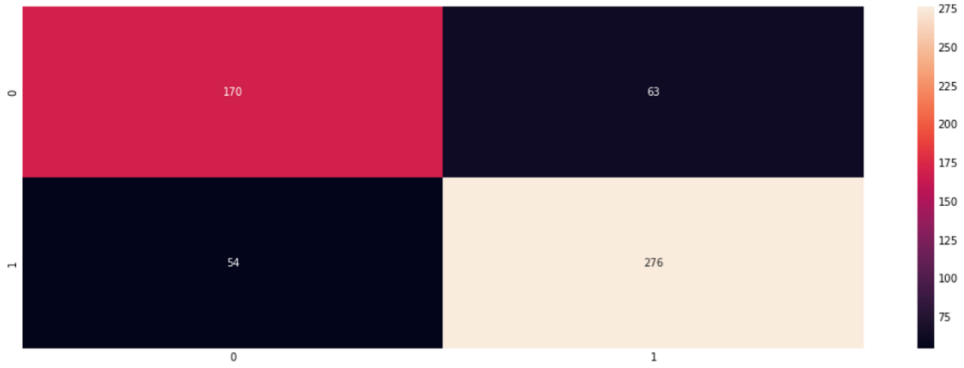
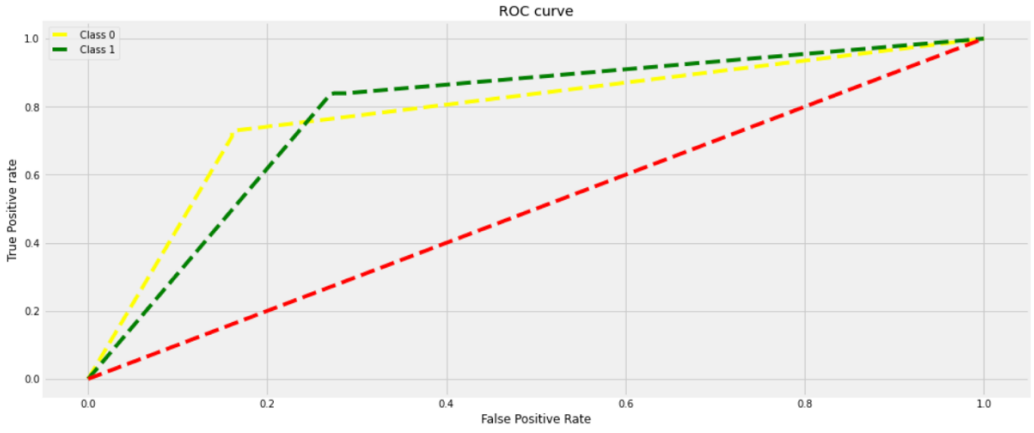


Figure 72: Decision Tree model with RFECV on Stdscaled Balanced Binned Derived Dataset

Dataset: Minmax scaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.76	0.73	0.74	233
1	0.81	0.84	0.83	330
accuracy			0.79	563
macro avg	0.79	0.78	0.78	563
weighted avg	0.79	0.79	0.79	563





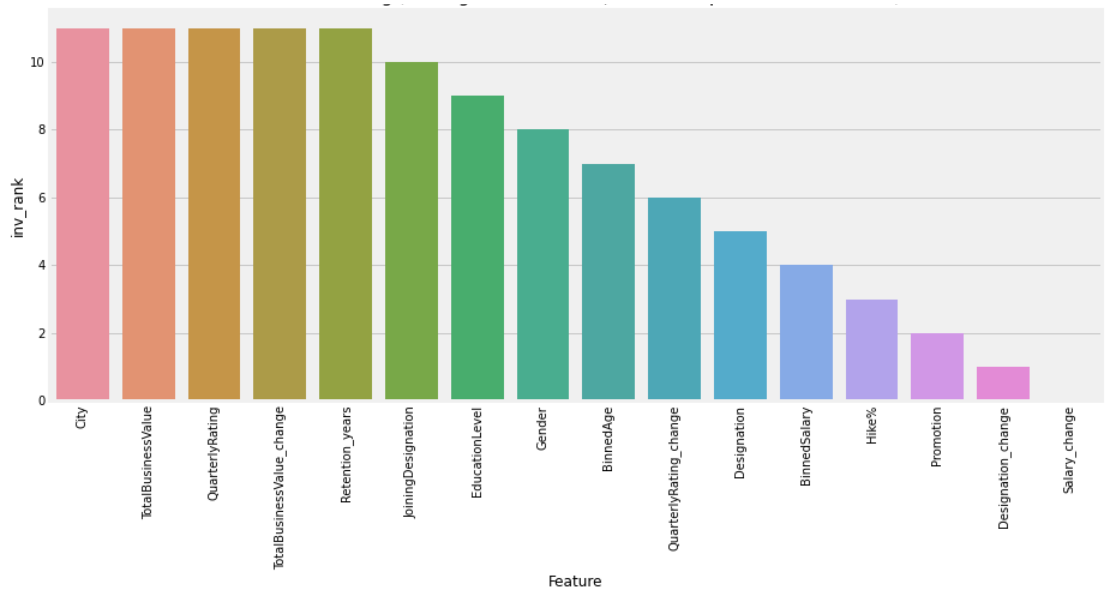


Figure 73:Decision Tree model with RFECV on Minmaxscaled Balanced Binned Derived Dataset

20.4. **Model 3: KNeighbors Classifier**

KNeighbors Classifier is a "Supervised Machine Learning" algorithm that implements classification based on the voting by the k nearest data points wrt the target data point. (Scikit Learn sklearn.neighbors.KNeighborsClassifier, n.d.)

To conclude the model parameters corresponding to the best model performance Randomized Search CV was implemented on different datasets.

```
knn_rand_params_={}
for name, data in datasets:
    print('Dataset:',name)
    param = RSCV_KNN(data)
    knn_rand_params_[name] = param
```

```
Dataset: Balanced Derived Data
Best n_neighbors: 49
Best weights: uniform
Best metric: manhattan
Best algorithm: kd_tree
Best p: 13
Best Accuracy Score: 0.7

{'algorithm': 'kd_tree', 'leaf_size': 30, 'metric': 'manhattan', 'metric_params': None, 'n_jobs': None, 'n_neighbor
s': 49, 'p': 13, 'weights': 'uniform'}
-----
Dataset: Balanced Binned Derived Data
Best n_neighbors: 17
Best weights: uniform
Best metric: minkowski
Best algorithm: auto
Best p: 11
Best Accuracy Score: 0.7566666666666666

{'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 1
7, 'p': 11, 'weights': 'uniform'}
-----
Dataset: Stdscaled Balanced Binned Derived Data
Best n_neighbors: 9
Best weights: uniform
Best metric: manhattan
Best algorithm: brute
Best p: 6
Best Accuracy Score: 0.7593333333333332

{'algorithm': 'brute', 'leaf_size': 30, 'metric': 'manhattan', 'metric_params': None, 'n_jobs': None, 'n_neighbors':
9, 'p': 6, 'weights': 'uniform'}
-----
Dataset: Minmax scaled Balanced Binned Derived Data
Best n_neighbors: 10
Best weights: distance
Best metric: manhattan
Best algorithm: kd_tree
Best p: 46
Best Accuracy Score: 0.756

{'algorithm': 'kd_tree', 'leaf_size': 30, 'metric': 'manhattan', 'metric_params': None, 'n_jobs': None, 'n_neighbor
s': 10, 'p': 46, 'weights': 'distance'}
-----
```

Figure 74:KNeighborsClassifier hyperparamter with RandomizedSearchCV

20.4.1. KNeighbors Model Parameters

- n_neighbors – number of neighbors
- weights – weight function used in prediction
- uniform – all points in neighborhood are weighted equally
- distance – weight points by the inverse of their distance
- metric – distance metric used for the tree
- algorithm – algorithm used to compute the nearest neighbors
- p – power parameter for the Minkowski metric

20.4.2. KNeighbors plots for various datasets

Best parameters concluded from Randomised Search CV on KNeighbors Classifier were used for model building. (Scikit Learn sklearn.neighbors.KNeighborsClassifier, n.d.) (Data Analystics Learning Curves Explained with Python Sklearn Example, n.d.)

```
Dataset:  Balanced Derived Data
Accuracy:0.68
Precision:0.67
Recall:0.87
F1-Score:0.76
              precision    recall  f1-score   support

           0       0.68      0.40      0.50       155
           1       0.67      0.87      0.76       221

    accuracy                           0.68       376
   macro avg       0.68      0.63      0.63       376
weighted avg       0.68      0.68      0.65       376
```

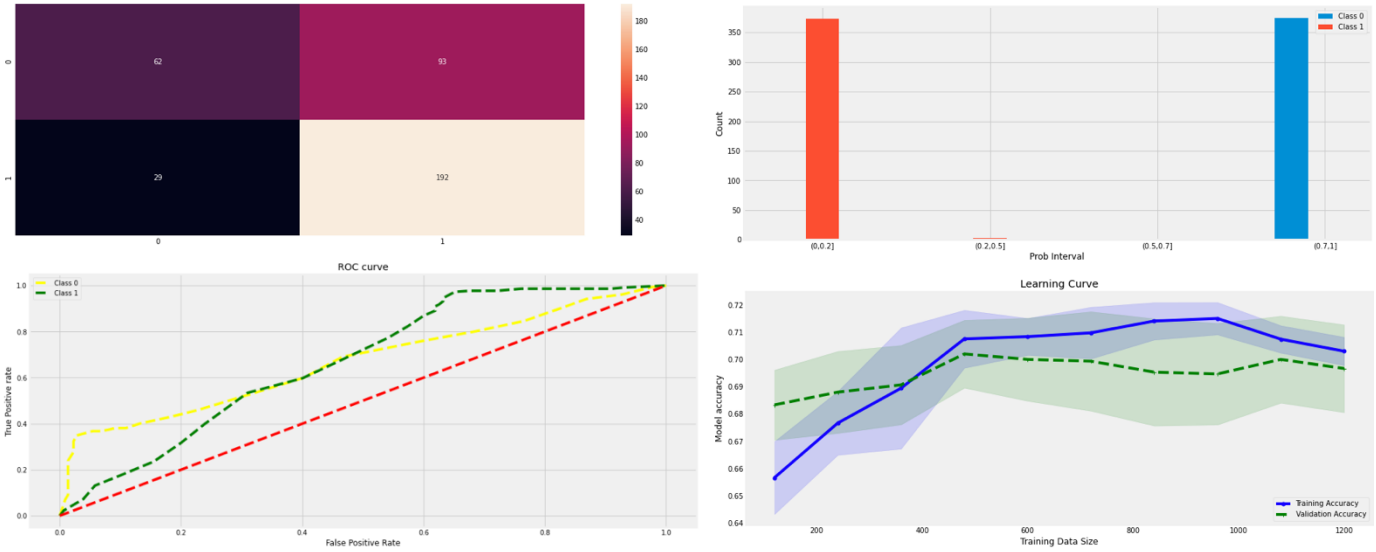


Figure 75: KNeighborsClassifier model on Balanced Derived Data

Dataset: Balanced Binned Derived Data
Accuracy:0.76
Precision:0.77
Recall:0.86
F1-Score:0.81

	precision	recall	f1-score	support
0	0.75	0.63	0.69	155
1	0.77	0.86	0.81	221
accuracy			0.76	376
macro avg	0.76	0.74	0.75	376
weighted avg	0.76	0.76	0.76	376

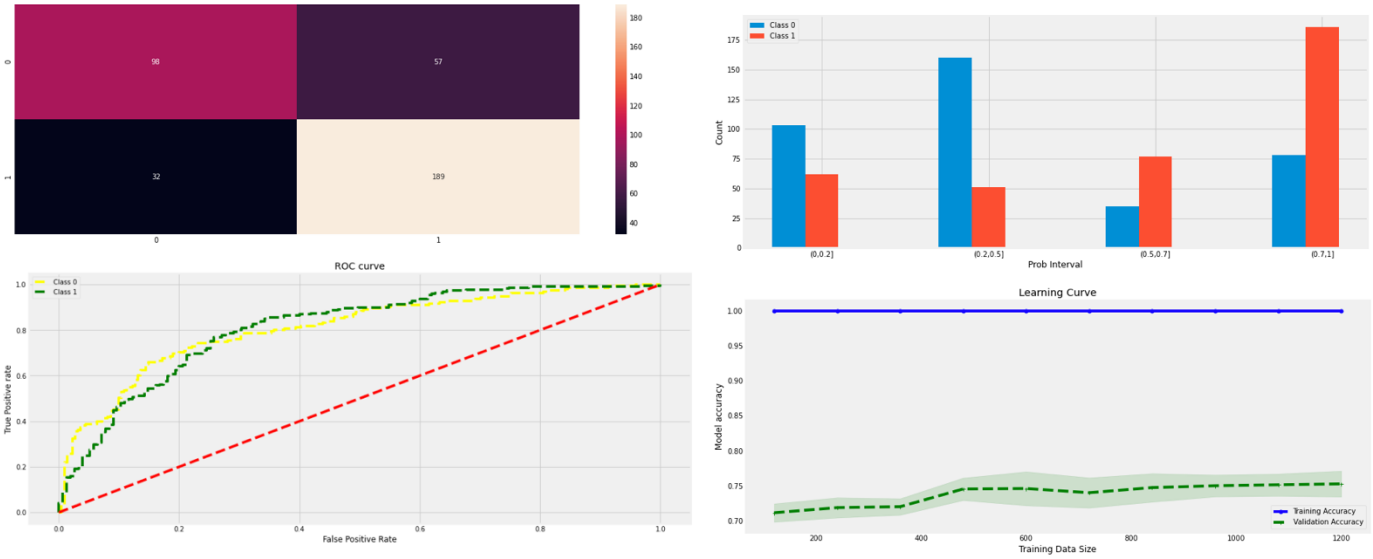


Figure 76:KNeighborsClassifier model on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data
Accuracy:0.77
Precision:0.77
Recall:0.87
F1-Score:0.82

	precision	recall	f1-score	support
0	0.77	0.63	0.70	155
1	0.77	0.87	0.82	221
accuracy			0.77	376
macro avg	0.77	0.75	0.76	376
weighted avg	0.77	0.77	0.77	376

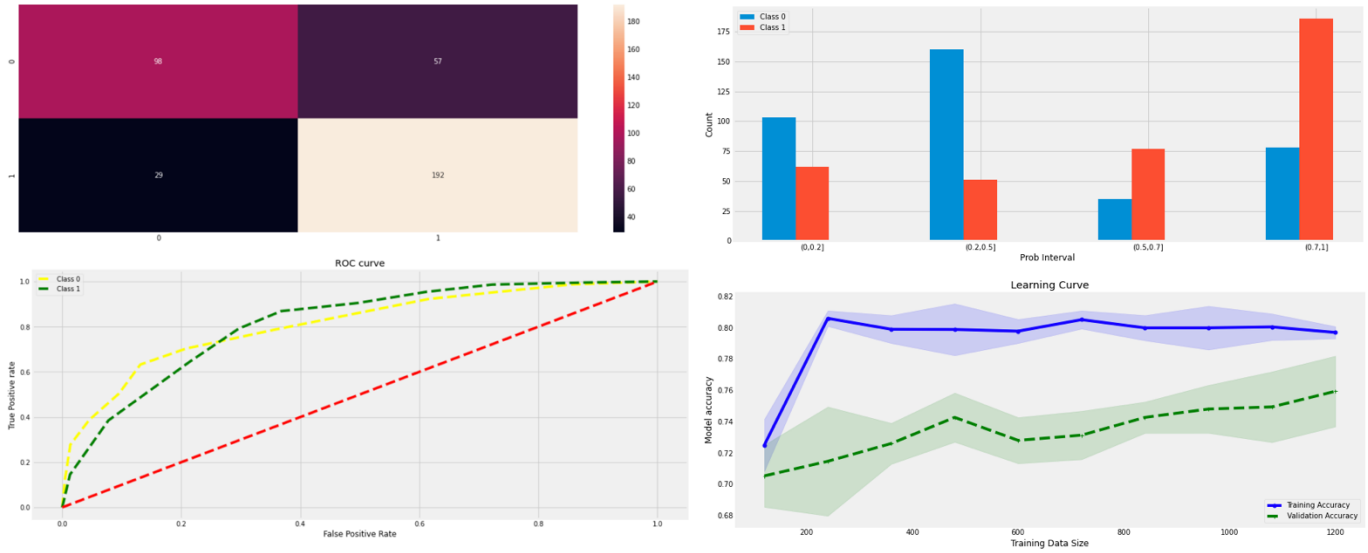


Figure 77:KNeighborsClassifier model on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data

Accuracy:0.77

Precision:0.78

Recall:0.85

F1-Score:0.81

	precision	recall	f1-score	support
0	0.75	0.65	0.70	155
1	0.78	0.85	0.81	221
accuracy			0.77	376
macro avg	0.77	0.75	0.76	376
weighted avg	0.77	0.77	0.77	376

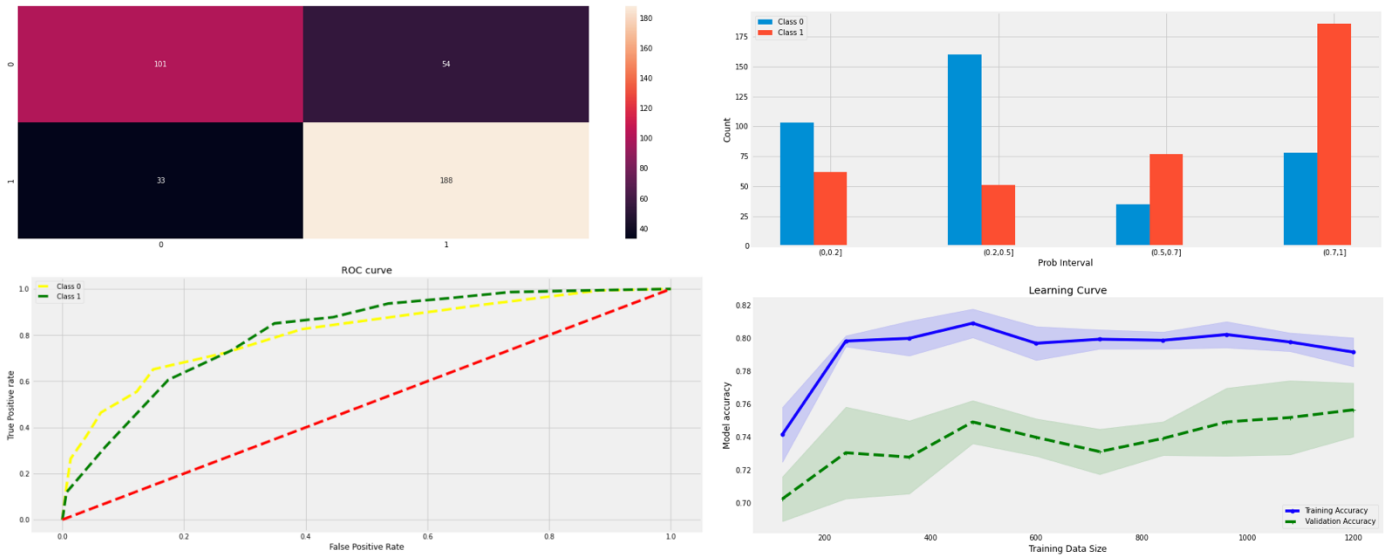


Figure 78:KNeighborsClassifier model on Minmaxscale Balanced Binned Derived Data

20.5. Model 4: Bernoulli NB

Bernoulli Naïve Bayes is a variant of Naïve Bayes algorithm in Machine Learning. It assumes that each feature is a binary valued independent feature. (Scikit Learn sklearn.naive_bayes.BernoulliNB, n.d.)

To conclude the model parameters corresponding to the best model performance Randomized Search CV was implemented on different datasets.

```
bnb_rand_params_={}
for name, data in datasets:
    print('Dataset:',name)
    param = RSCV_BNB(data)
    bnb_rand_params_[name] = param
```

```
Dataset: Balanced Derived Data
Best Accuracy Score: 0.7673333333333334

{'alpha': 0.001, 'binarize': 0.0, 'class_prior': None, 'fit_prior': True}
-----
Dataset: Balanced Binned Derived Data
Best Accuracy Score: 0.7553333333333333

{'alpha': 0.001, 'binarize': 0.0, 'class_prior': None, 'fit_prior': True}
-----
Dataset: Stdscaled Balanced Binned Derived Data
Best Accuracy Score: 0.7553333333333333

{'alpha': 0.001, 'binarize': 0.0, 'class_prior': None, 'fit_prior': True}
-----
Dataset: Minmax scaled Balanced Binned Derived Data
Best Accuracy Score: 0.7553333333333333

{'alpha': 0.001, 'binarize': 0.0, 'class_prior': None, 'fit_prior': True}
-----
```

Figure 79: BernoulliNB hyperparamter with RandomizedSearchCV

20.5.1. Bernoulli NB Model Parameters

- **alpha** – additive (laplace/lidstone) smoothing parameter
- **fit_prior** – whether to learn class prior probabilities or not

20.5.2. Bernoulli NB plots for various datasets

Best parameters concluded from Randomised Search CV on BernoulliNB were used for model building.

Dataset: Balanced Derived Data
Accuracy:0.76
Precision:0.74
Recall:0.76
F1-Score:0.75

	precision	recall	f1-score	support
0	0.80	0.55	0.65	155
1	0.74	0.90	0.81	221
accuracy			0.76	376
macro avg	0.77	0.73	0.73	376
weighted avg	0.76	0.76	0.75	376

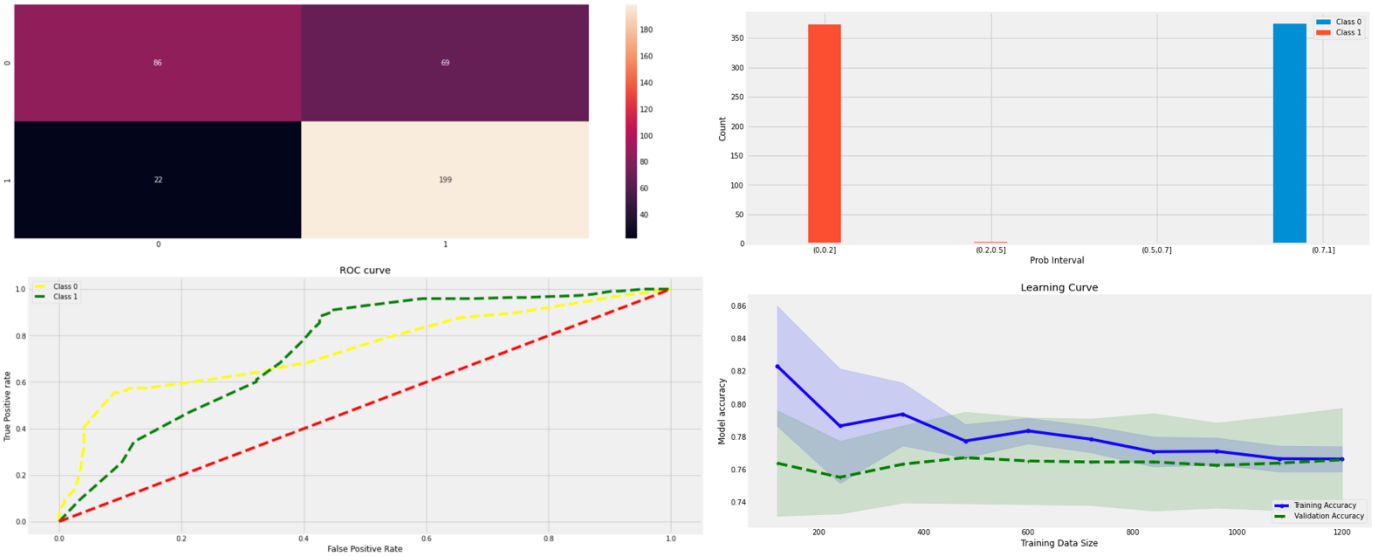


Figure 80:BernoulliNB model on Balanced Derived Data

Dataset: Balanced Binned Derived Data
Accuracy:0.76
Precision:0.79
Recall:0.76
F1-Score:0.76

	precision	recall	f1-score	support
0	0.71	0.70	0.70	155
1	0.79	0.80	0.79	221
accuracy			0.76	376
macro avg	0.75	0.75	0.75	376
weighted avg	0.75	0.76	0.76	376

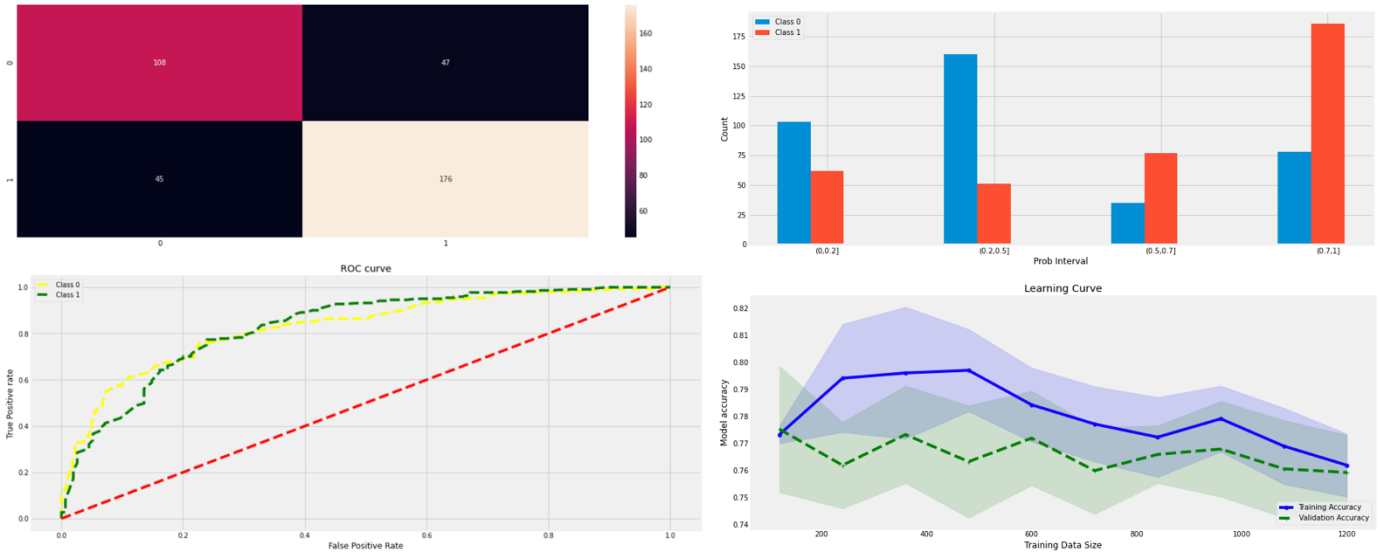


Figure 81:BernoulliNB model on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data
Accuracy:0.76
Precision:0.79
Recall:0.76
F1-Score:0.76

	precision	recall	f1-score	support
0	0.71	0.70	0.70	155
1	0.79	0.80	0.79	221
accuracy			0.76	376
macro avg	0.75	0.75	0.75	376
weighted avg	0.75	0.76	0.76	376

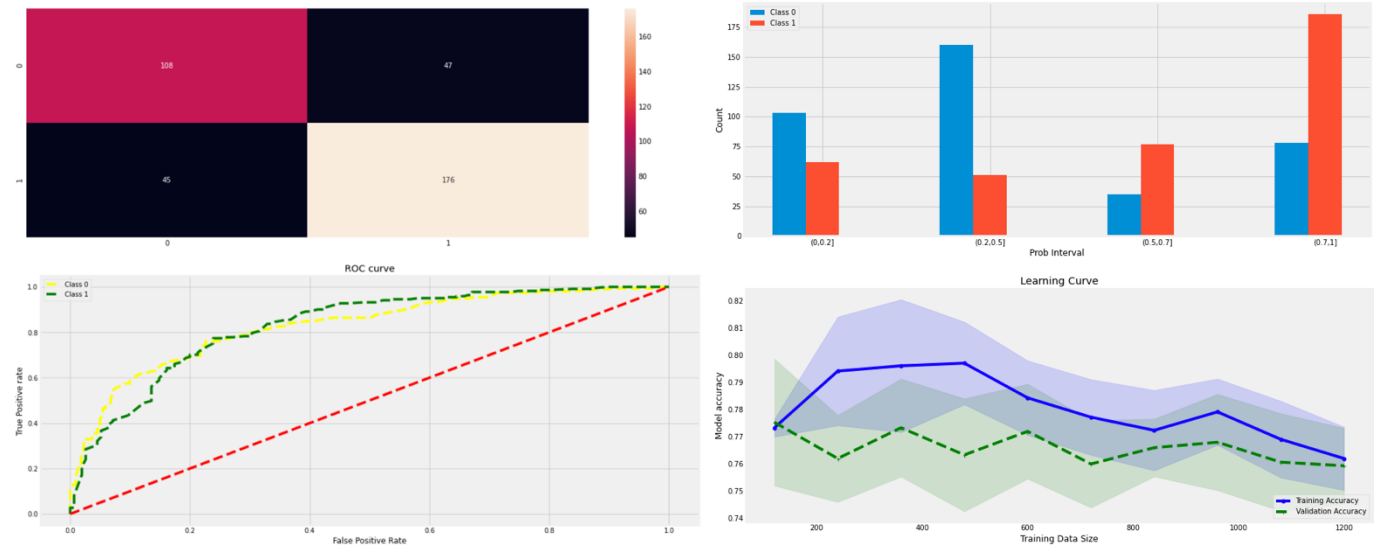


Figure 82:BernoulliNB model on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data
Accuracy:0.76
Precision:0.79
Recall:0.76
F1-Score:0.76

	precision	recall	f1-score	support
0	0.71	0.70	0.70	155
1	0.79	0.80	0.79	221
accuracy			0.76	376
macro avg	0.75	0.75	0.75	376
weighted avg	0.75	0.76	0.76	376

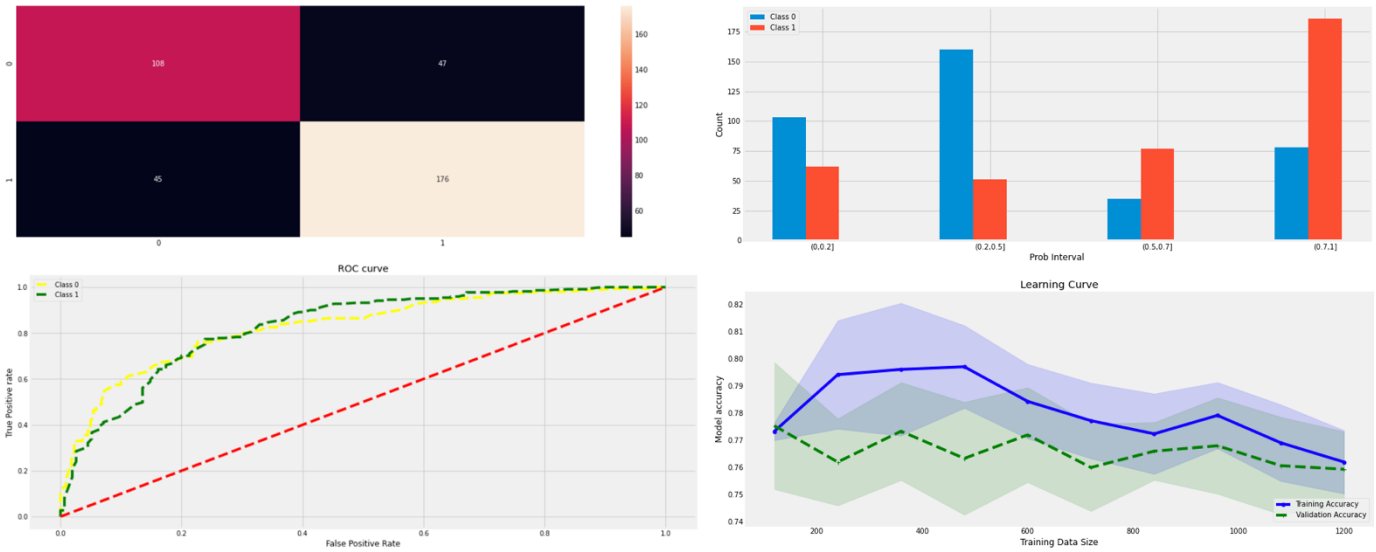


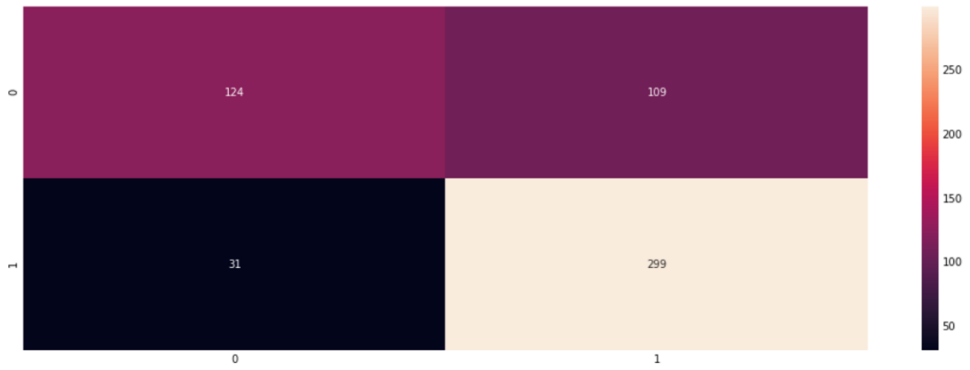
Figure 83: BernoulliNB model on Minmaxscaled Balanced Binned Data

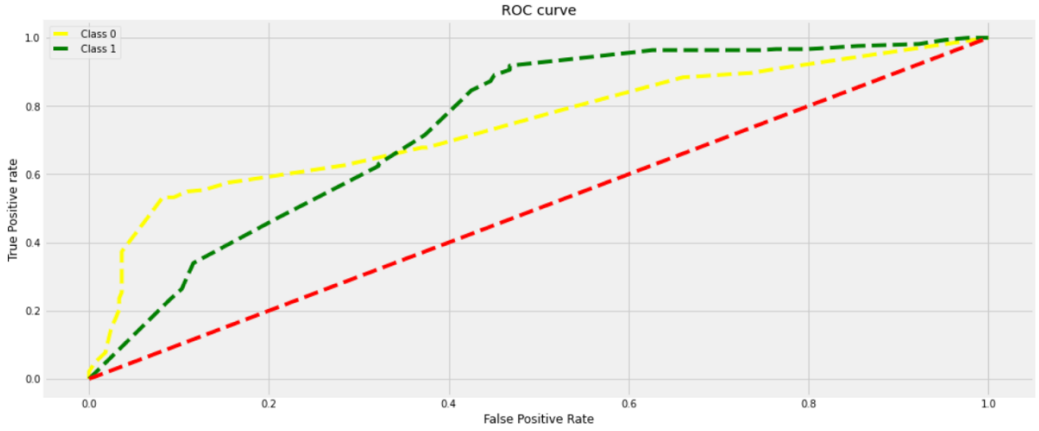
20.5.3. Bernoulli NB with RFECV

(Scikit Learn sklearn.feature_selection.RFECV, n.d.)

Dataset: Balanced Derived Data

	precision	recall	f1-score	support
0	0.80	0.53	0.64	233
1	0.73	0.91	0.81	330
accuracy			0.75	563
macro avg	0.77	0.72	0.72	563
weighted avg	0.76	0.75	0.74	563





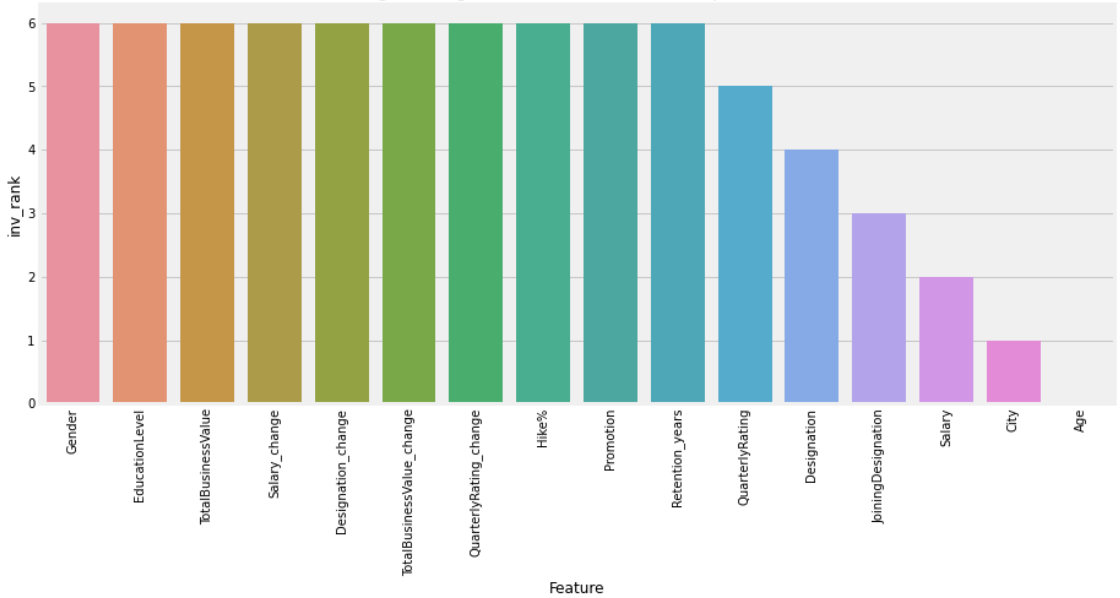
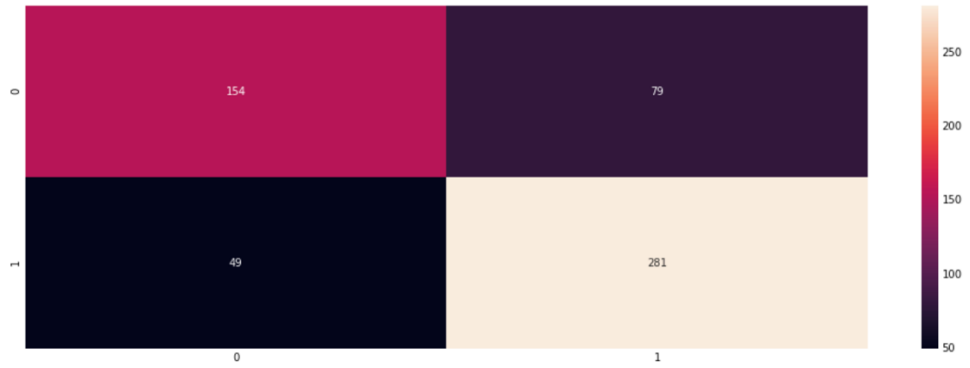
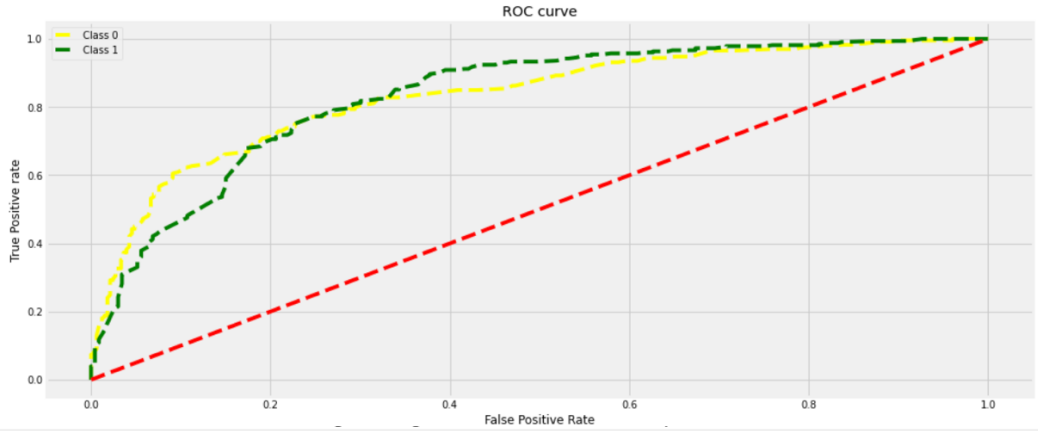


Figure 84:BernoulliNB model with RFECV on Balanced Derived Data

Dataset: Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.76	0.66	0.71	233
1	0.78	0.85	0.81	330
accuracy			0.77	563
macro avg	0.77	0.76	0.76	563
weighted avg	0.77	0.77	0.77	563





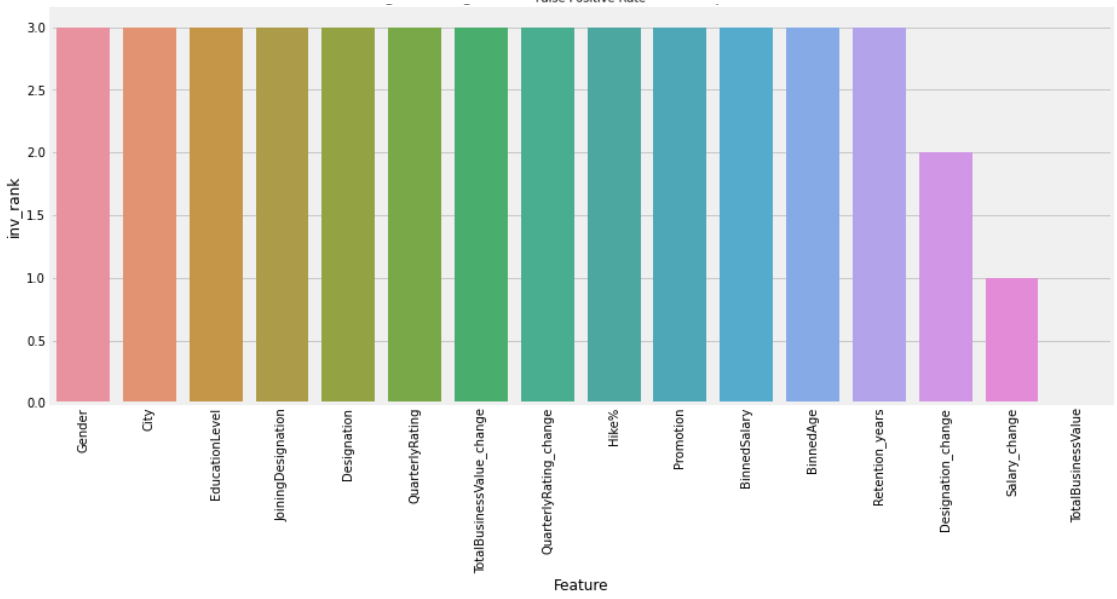
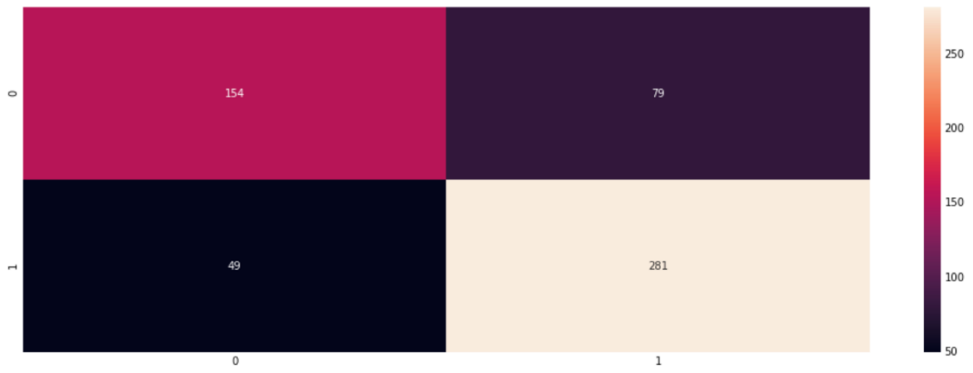
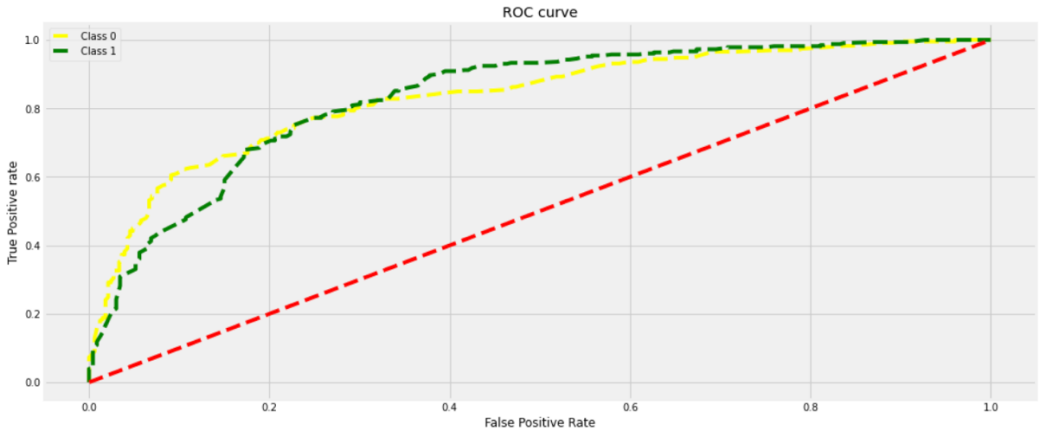


Figure 85:BernoulliNB model with RFECV on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.76	0.66	0.71	233
1	0.78	0.85	0.81	330
accuracy			0.77	563
macro avg	0.77	0.76	0.76	563
weighted avg	0.77	0.77	0.77	563





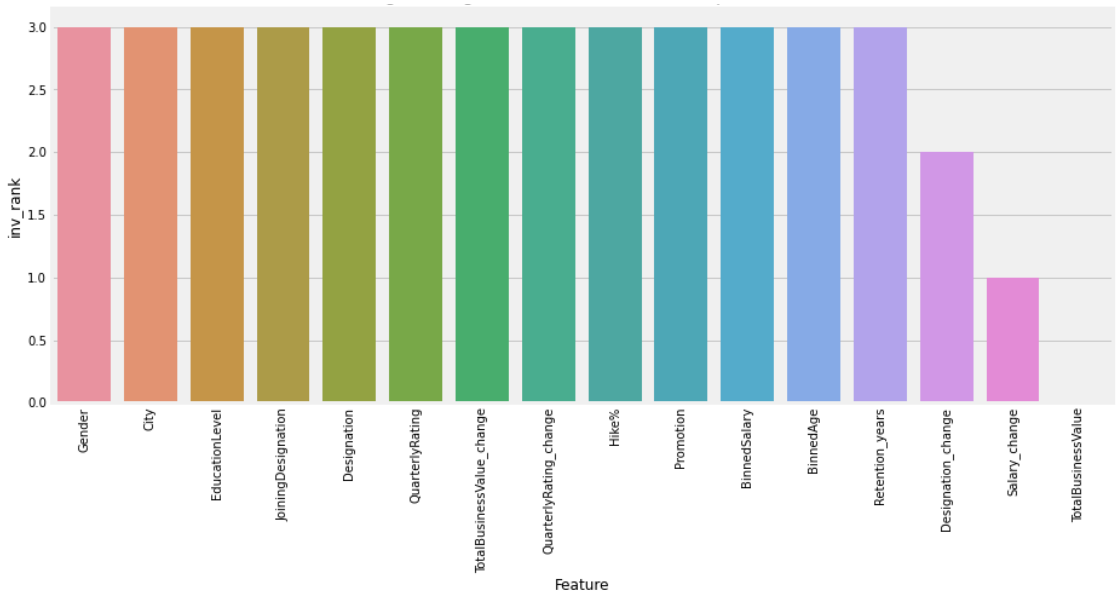
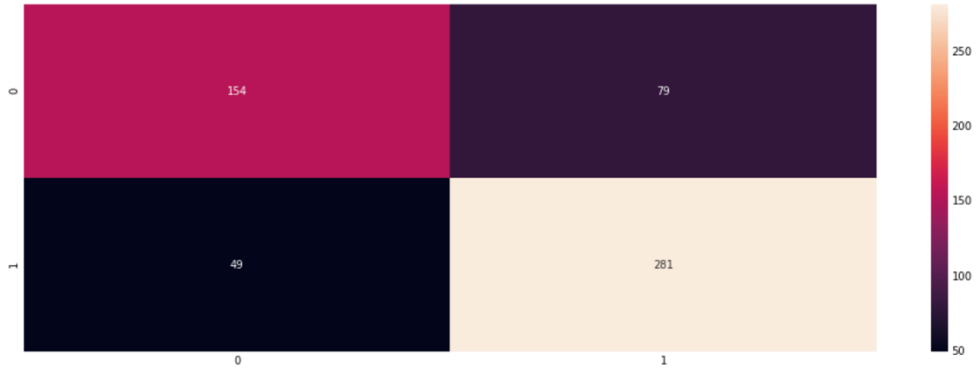
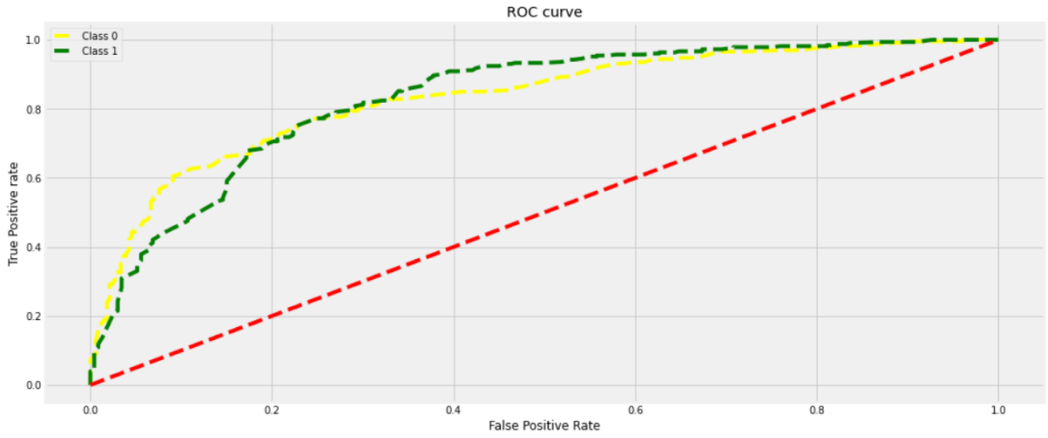


Figure 86:BernoulliNB model with RFECV on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.76	0.66	0.71	233
1	0.78	0.85	0.81	330
accuracy			0.77	563
macro avg	0.77	0.76	0.76	563
weighted avg	0.77	0.77	0.77	563





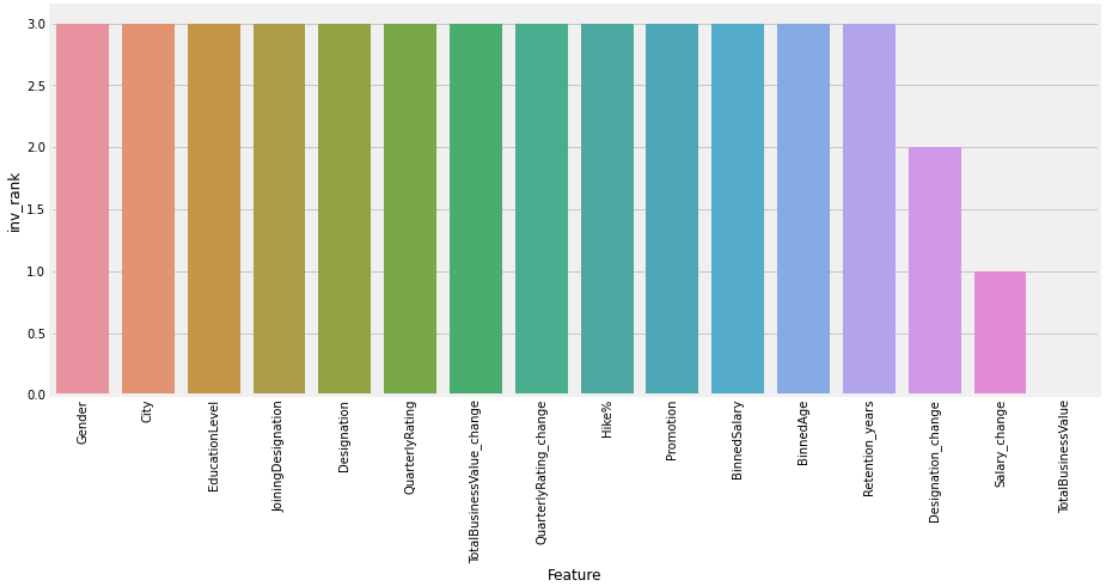


Figure 87:BernoulliNB model with RFECV on Minmaxscale Balanced Binned Derived Data

20.6. Model 5: Gaussian NB

Gaussian Naïve Bayes is a special type of Naïve Bayes algorithm in Machine Learning. It assumes that the features follow a gaussian (or normal) distribution. (Scikit Learn sklearn.naive_bayes.GaussianNB, n.d.)

To conclude the model parameters corresponding to the best model performance Randomized Search CV was implemented on different datasets.

```
gnb_rand_params_={}
for name, data in datasets:
    print('Dataset:',name)
    param = RSCV_GNB(data)
    gnb_rand_params_[name] = param
```

```
Dataset: Balanced Derived Data
Best Accuracy Score: 0.6953333333333334

{'priors': None, 'var_smoothing': 0.001}
-----
Dataset: Balanced Binned Derived Data
Best Accuracy Score: 0.7193333333333334

{'priors': None, 'var_smoothing': 0.001}
-----
Dataset: Stdscaled Balanced Binned Derived Data
Best Accuracy Score: 0.7193333333333334

{'priors': None, 'var_smoothing': 0.001}
-----
Dataset: Minmax scaled Balanced Binned Derived Data
Best Accuracy Score: 0.7193333333333334

{'priors': None, 'var_smoothing': 0.001}
-----
```

Figure 88:GaussianNB hyperparameter with RandomizedSearchCV

20.6.1. Gaussian NB Model Parameters

- **prior** – prior proabilities of the classes. If specified, the priors are not adjusted according to the data
- **var_smoothing** – portion of the largest variance of all features added to variances for calculation stability

20.6.2. Gaussian NB plots for various datasets

Best parameters concluded from Randomised Search CV on GaussianNB were used for model building. (Scikit Learn sklearn.naive_bayes.GaussianNB, n.d.) (Data Analystics Learning Curves Explained with Python Sklearn Example, n.d.)

Dataset: Balanced Derived Data
Accuracy:0.7
Precision:0.68
Recall:0.70
F1-Score:0.67

	precision	recall	f1-score	support
0	0.81	0.37	0.51	155
1	0.68	0.94	0.79	221
accuracy			0.70	376
macro avg	0.74	0.66	0.65	376
weighted avg	0.73	0.70	0.67	376

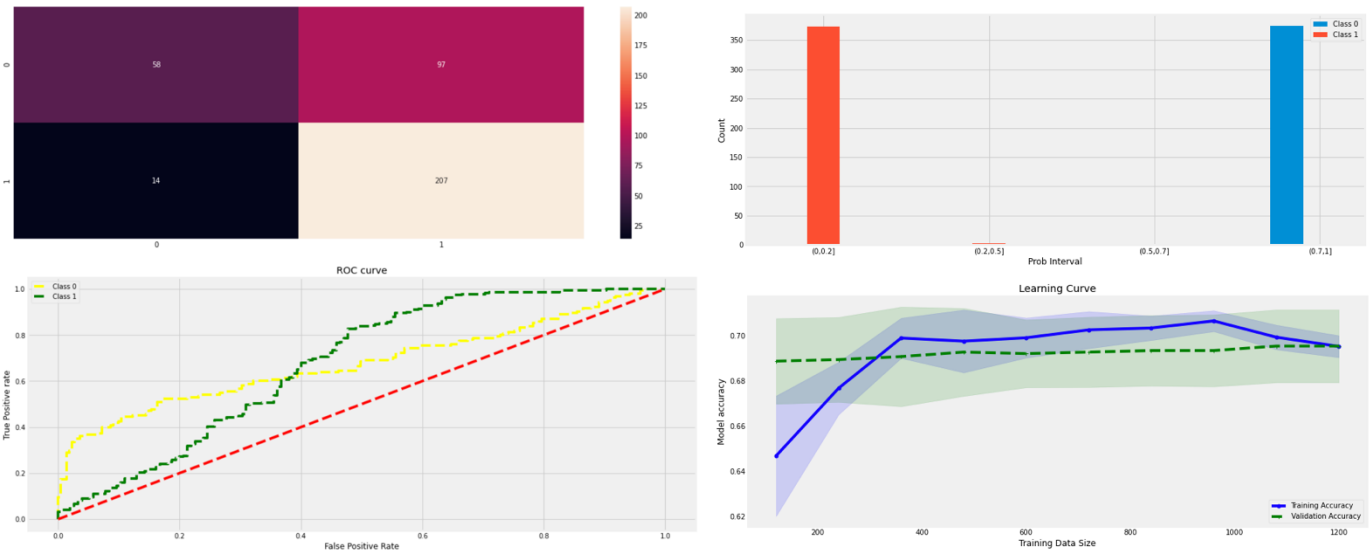


Figure 89:GaussianiNB model on Balanced Derived Data

Dataset: Balanced Binned Derived Data
Accuracy:0.74
Precision:0.72
Recall:0.74
F1-Score:0.72

	precision	recall	f1-score	support
0	0.82	0.48	0.60	155
1	0.72	0.93	0.81	221
accuracy			0.74	376
macro avg	0.77	0.70	0.71	376
weighted avg	0.76	0.74	0.72	376

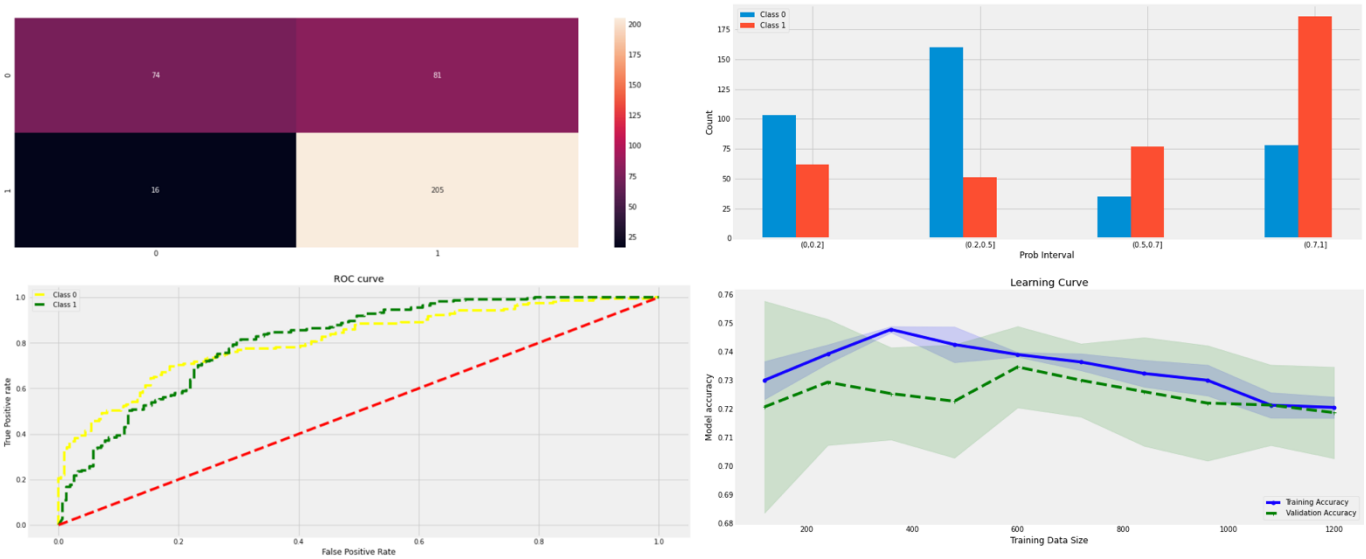


Figure 90:GaussianNB model on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data

Accuracy:0.74

Precision:0.72

Recall:0.74

F1-Score:0.72

	precision	recall	f1-score	support
0	0.82	0.48	0.60	155
1	0.72	0.93	0.81	221
accuracy			0.74	376
macro avg	0.77	0.70	0.71	376
weighted avg	0.76	0.74	0.72	376

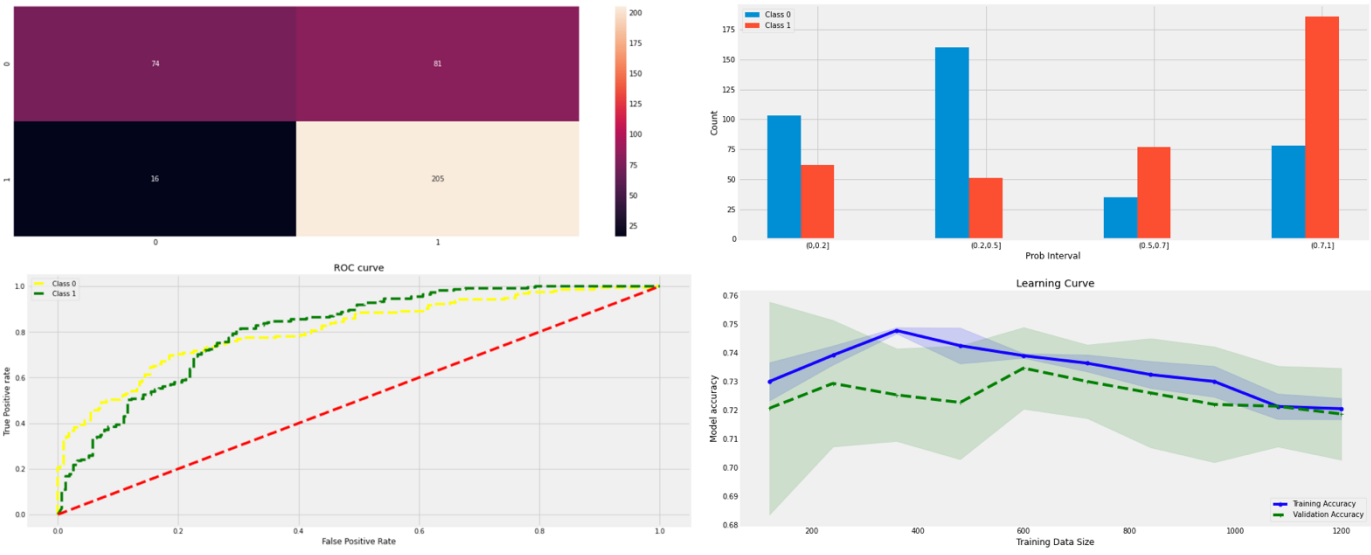


Figure 91: GaussianNB model on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data
Accuracy:0.74
Precision:0.72
Recall:0.74
F1-Score:0.72

	precision	recall	f1-score	support
0	0.82	0.48	0.60	155
1	0.72	0.93	0.81	221
accuracy			0.74	376
macro avg	0.77	0.70	0.71	376
weighted avg	0.76	0.74	0.72	376

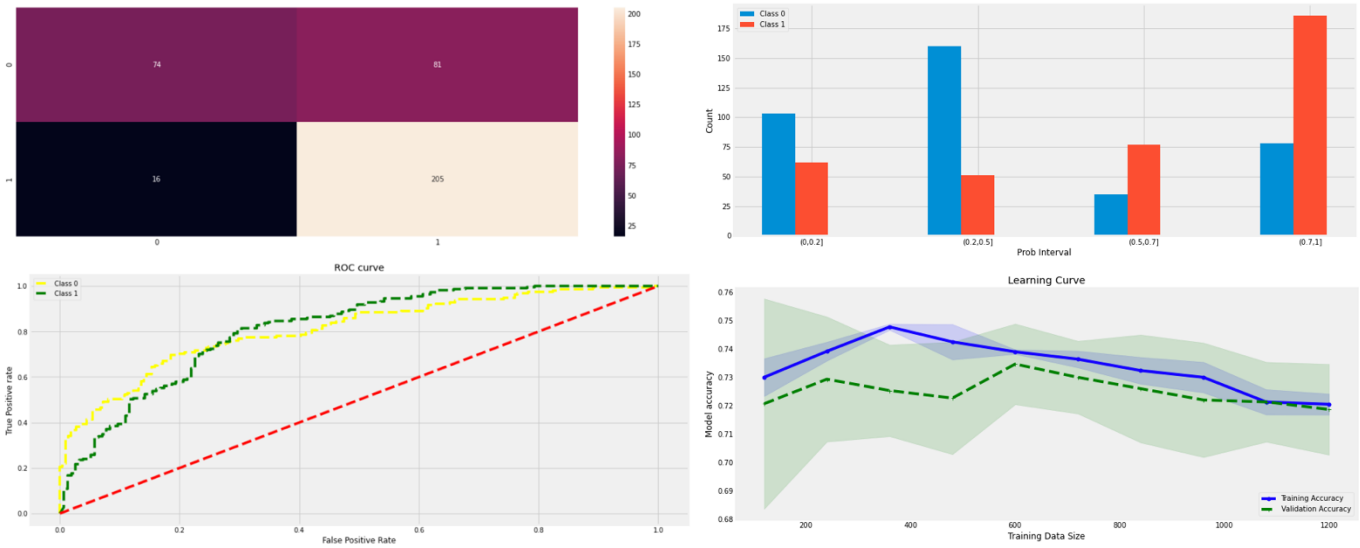


Figure 92: GaussiaNB model on Minmaxscale Balanced Binned Derived Data

20.7. **Model 6: Random Forest**

Random forest is a “Supervised Machine Learning” algorithm widely used in Classification and Regression problems. It works by building decision tree on different samples and use the majority vote for classification. (Scikit Learn sklearn.ensemble.RandomForestClassifier, n.d.)

To conclude the model parameters corresponding to the best model performance Randomized Search CV was implemented on different datasets.

```
rf_rand_params_={}
for name, data in datasets:
    print('Dataset:',name)
    param = RSCV_RF(data)
    rf_rand_params_[name] = param
```

```
Dataset: Balanced Derived Data
Best criterion: entropy
Best max_depth: 3
Best max_features: sqrt
Best min_samples_split: 2
Best min_samples_leaf: 14
Best random_state: 12
Best max_leaf_nodes: 19
Best Accuracy Score: 0.5866666666666667

{'bootstrap': True, 'ccp_alpha': 0.72099999999999999, 'class_weight': None, 'criterion': 'entropy', 'max_depth': 3, 'm
ax_features': 'sqrt', 'max_leaf_nodes': 19, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1
4, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False,
'random_state': 12, 'verbose': 0, 'warm_start': False}
-----
Dataset: Balanced Binned Derived Data
Best criterion: entropy
Best max_depth: 15
Best max_features: sqrt
Best min_samples_split: 2
Best min_samples_leaf: 17
Best random_state: 12
Best max_leaf_nodes: 19
Best Accuracy Score: 0.5866666666666667

{'bootstrap': True, 'ccp_alpha': 0.23099999999999996, 'class_weight': None, 'criterion': 'entropy', 'max_depth': 15,
'max_features': 'sqrt', 'max_leaf_nodes': 19, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf':
17, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False,
'random_state': 12, 'verbose': 0, 'warm_start': False}
-----
Dataset: Stdscaled Balanced Binned Derived Data
Best criterion: gini
Best max_depth: 13
Best max_features: log2
Best min_samples_split: 2
Best min_samples_leaf: 13
Best random_state: 12
Best max_leaf_nodes: 17
Best Accuracy Score: 0.6300000000000001

{'bootstrap': True, 'ccp_alpha': 0.11099999999999999, 'class_weight': None, 'criterion': 'gini', 'max_depth': 13, 'ma
x_features': 'log2', 'max_leaf_nodes': 17, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 13,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False, 'ra
ndom_state': 12, 'verbose': 0, 'warm_start': False}
-----
Dataset: Minmax scaled Balanced Binned Derived Data
Best criterion: entropy
Best max_depth: 13
Best max_features: sqrt
Best min_samples_split: 2
Best min_samples_leaf: 13
Best random_state: 12
Best max_leaf_nodes: 18
Best Accuracy Score: 0.7646666666666666

{'bootstrap': True, 'ccp_alpha': 0.071, 'class_weight': None, 'criterion': 'entropy', 'max_depth': 13, 'max_feature
s': 'sqrt', 'max_leaf_nodes': 18, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 13, 'min_sam
ples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False, 'random_sta
te': 12, 'verbose': 0, 'warm_start': False}
-----
```

Figure 93: Random Forest hyperparamter with RandomizedSearchCV

20.7.1. Random Forest Model Parameters

- **criteria** – the function to measure the quality of split
- **n_estimator** – the number of trees in the forest
- **max_depth** – the maximum depth of the tree
- **min_samples_split** – the minimum number of samples required to split an internal node
- **min_sample_leaf** – the minimum number of samples required to be at the leaf node
- **max_features** – the number of features considered when looking for the best split
- **max_leaf_nodes** - Grow trees with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurities
- **bootstrap** – Whether bootstrap samples are used when tree building
- **ccp_alpha** – complexity parameter used for minimum cost complexity pruning

20.7.2. Random Forest plots for various datasets

Best parameters concluded from Randomised Search CV on Random Forest were used for model building. (Scikit Learn sklearn.ensemble.RandomForestClassifier, n.d.) (Data Analystics Learning Curves Explained with Python Sklearn Example, n.d.)

```
Dataset: Balanced Derived Data
Accuracy:0.73
Precision:0.71
Recall:0.92
F1-Score:0.80
              precision    recall  f1-score   support

           0       0.81      0.46      0.59       155
           1       0.71      0.92      0.80       221

    accuracy                           0.73       376
   macro avg       0.76      0.69      0.70       376
weighted avg       0.75      0.73      0.72       376
```

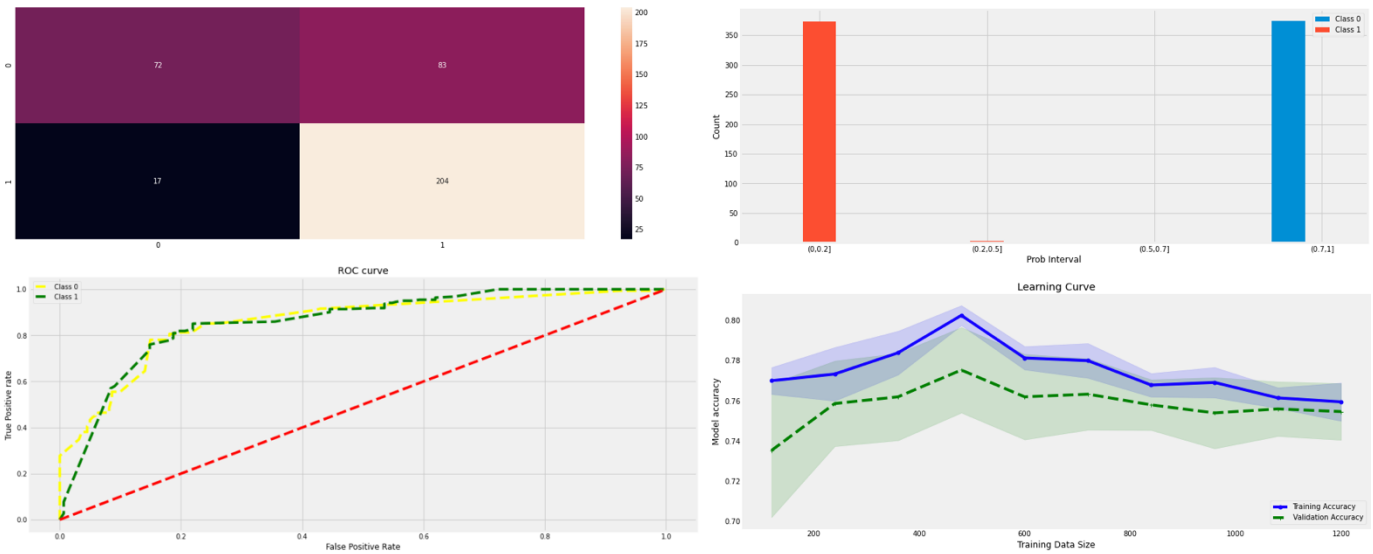


Figure 94:Random Forest model on Balanced Derived Data

Dataset: Balanced Binned Derived Data
Accuracy:0.77
Precision:0.77
Recall:0.87
F1-Score:0.82

	precision	recall	f1-score	support
0	0.77	0.62	0.69	155
1	0.77	0.87	0.82	221
accuracy			0.77	376
macro avg	0.77	0.75	0.75	376
weighted avg	0.77	0.77	0.76	376

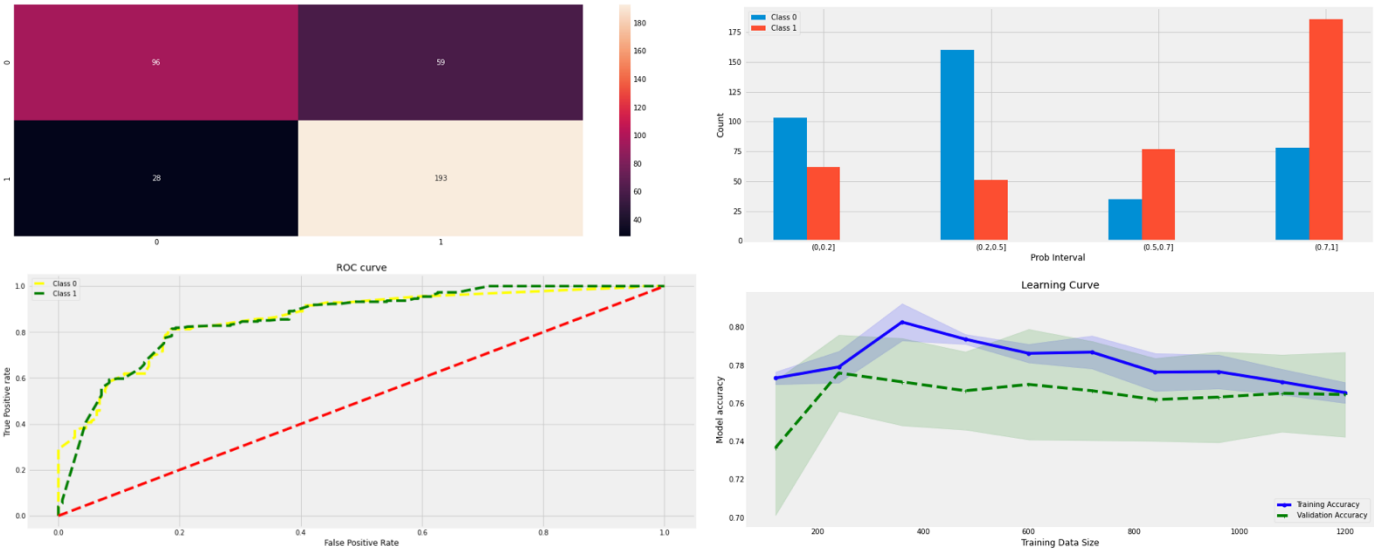


Figure 95:Random Forest model on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data
Accuracy:0.59
Precision:0.59
Recall:1.00
F1-Score:0.74

	precision	recall	f1-score	support
0	0.00	0.00	0.00	155
1	0.59	1.00	0.74	221
accuracy			0.59	376
macro avg	0.29	0.50	0.37	376
weighted avg	0.35	0.59	0.44	376

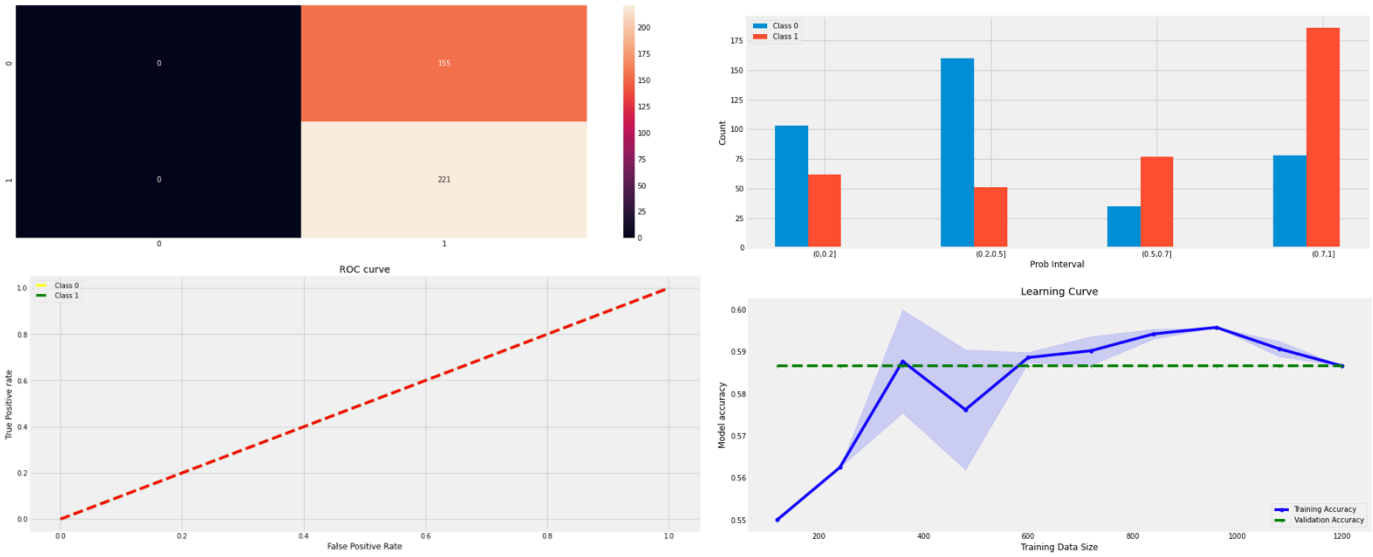


Figure 96:Random Forest model on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data
Accuracy:0.8
Precision:0.81
Recall:0.87
F1-Score:0.84

	precision	recall	f1-score	support
0	0.79	0.71	0.75	155
1	0.81	0.87	0.84	221
accuracy			0.80	376
macro avg	0.80	0.79	0.79	376
weighted avg	0.80	0.80	0.80	376

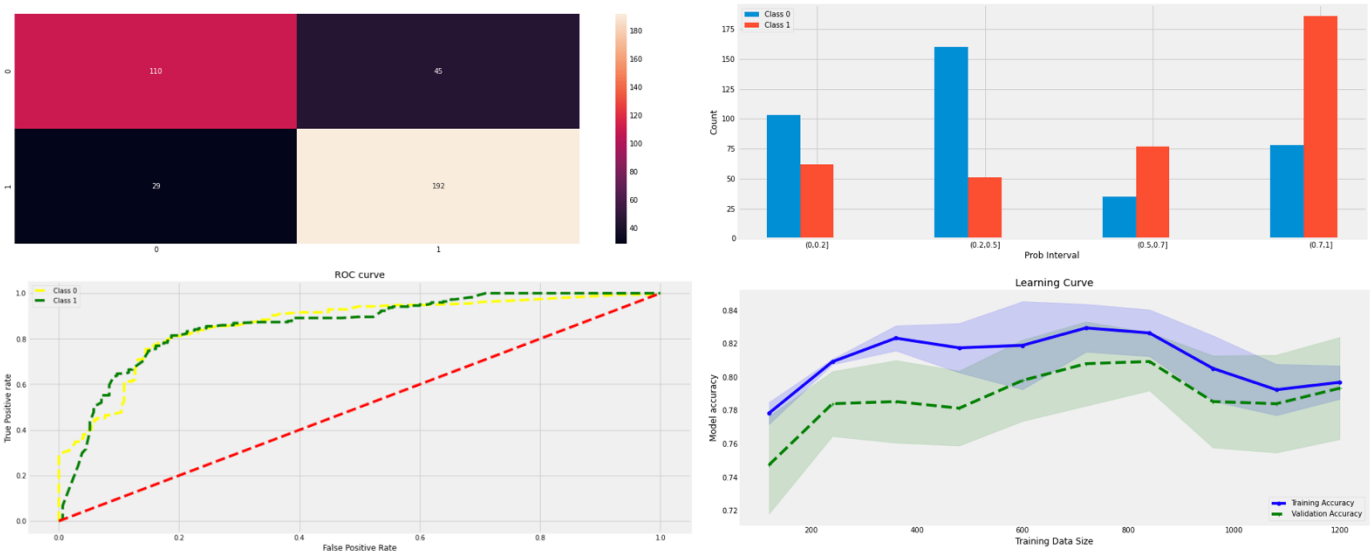
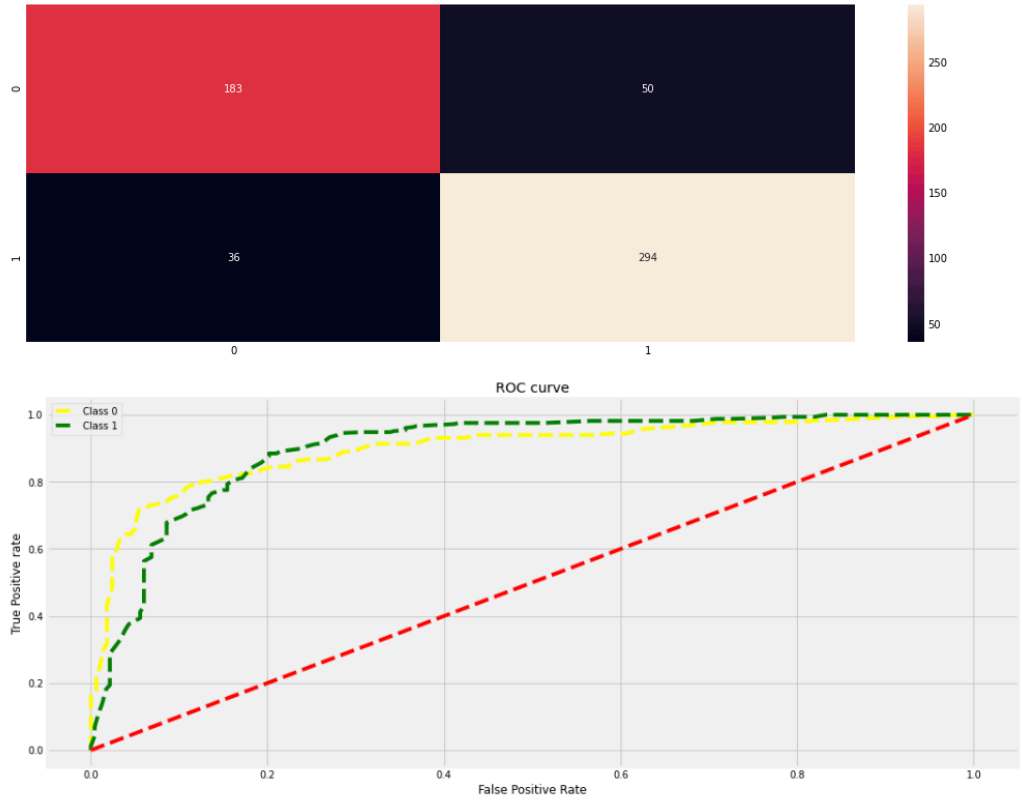


Figure 97: Random Forest model on Minmaxscaled Balanced Binned Derived Data

20.7.3. Random Forest with RFECV
(Scikit Learn sklearn.feature_selection.RFECV, n.d.)

Dataset: Balanced Derived Data

	precision	recall	f1-score	support
0	0.84	0.79	0.81	233
1	0.85	0.89	0.87	330
accuracy			0.85	563
macro avg	0.85	0.84	0.84	563
weighted avg	0.85	0.85	0.85	563



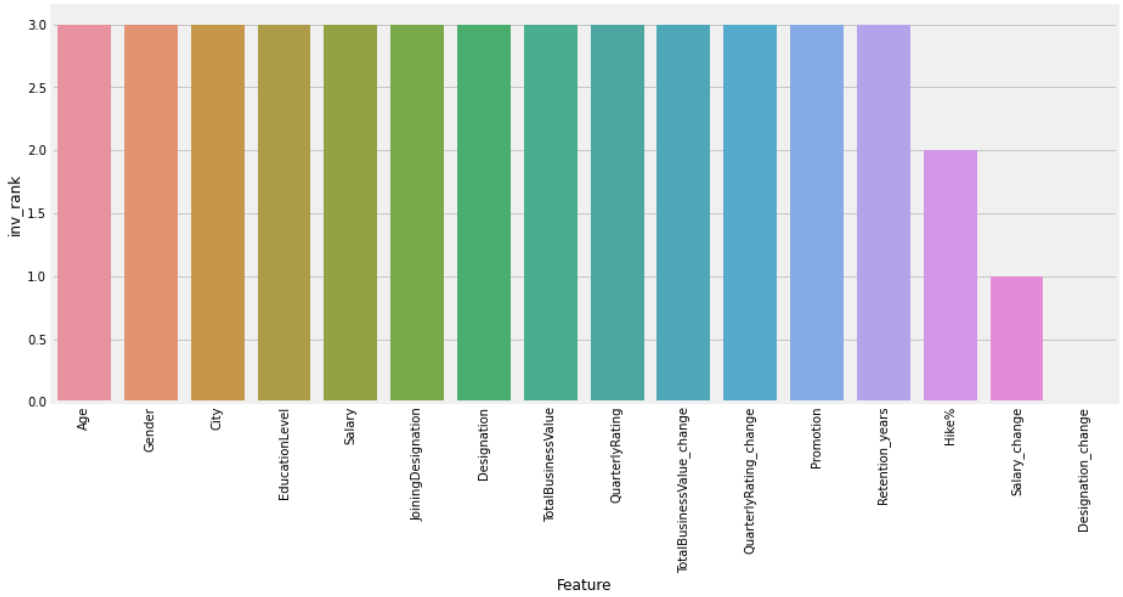
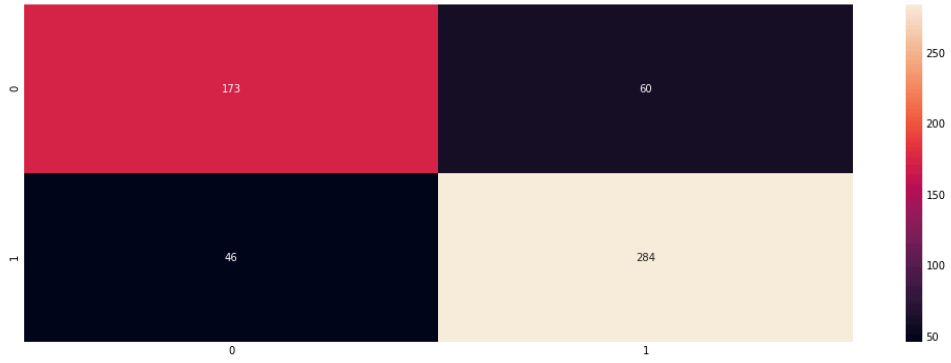
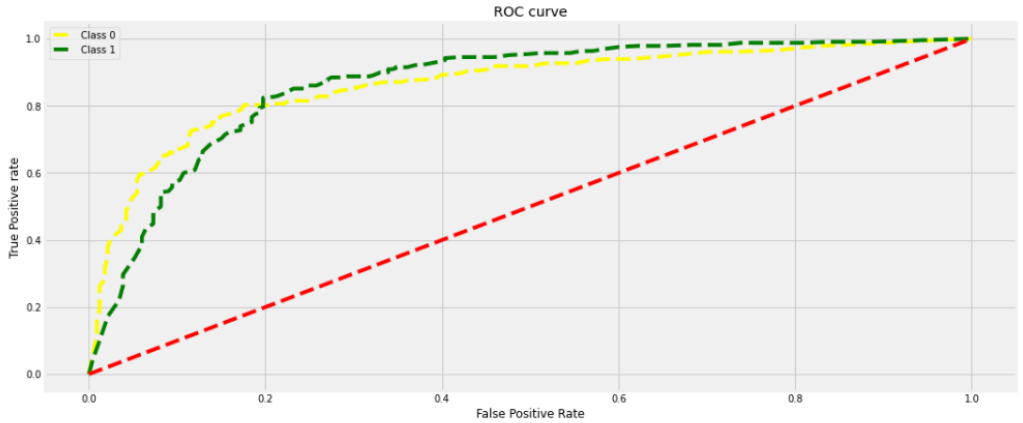


Figure 98: Random Forest model on Balanced Derived Data

Dataset: Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.79	0.74	0.77	233
1	0.83	0.86	0.84	330
accuracy			0.81	563
macro avg	0.81	0.80	0.80	563
weighted avg	0.81	0.81	0.81	563





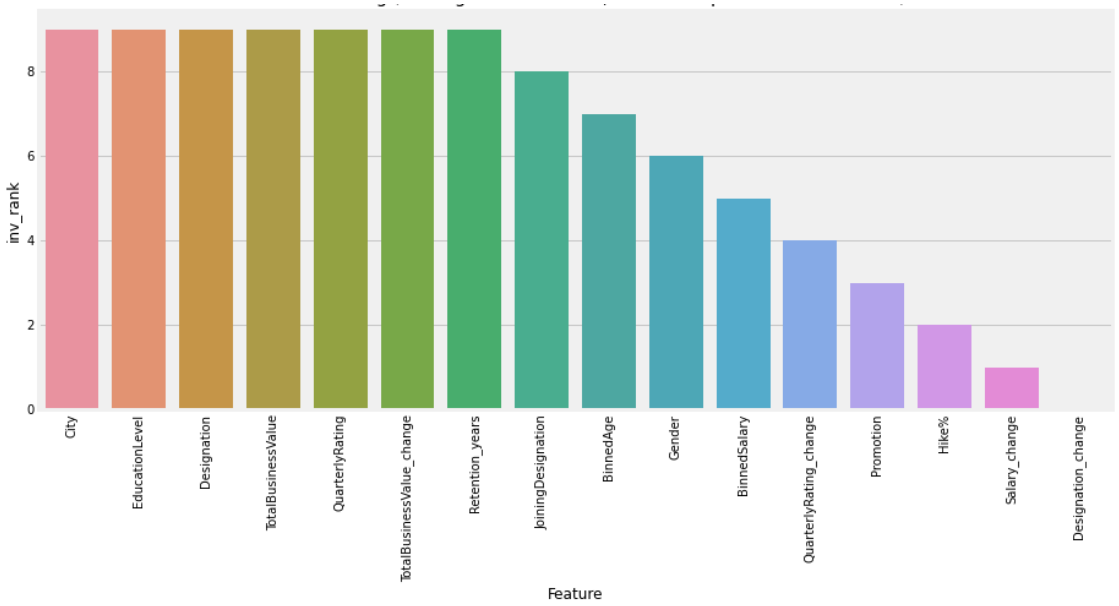
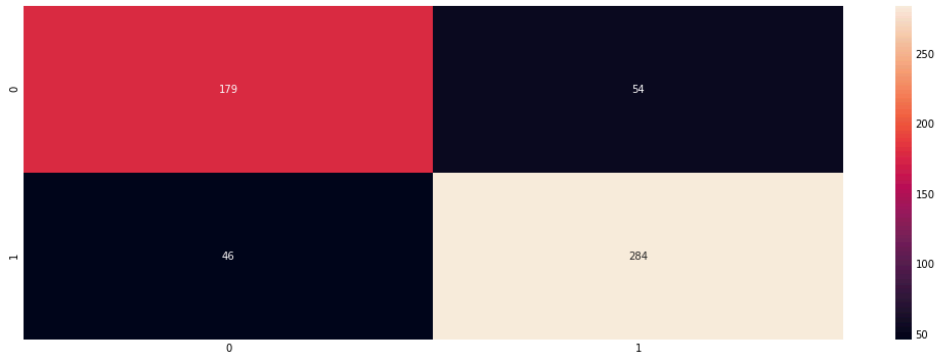
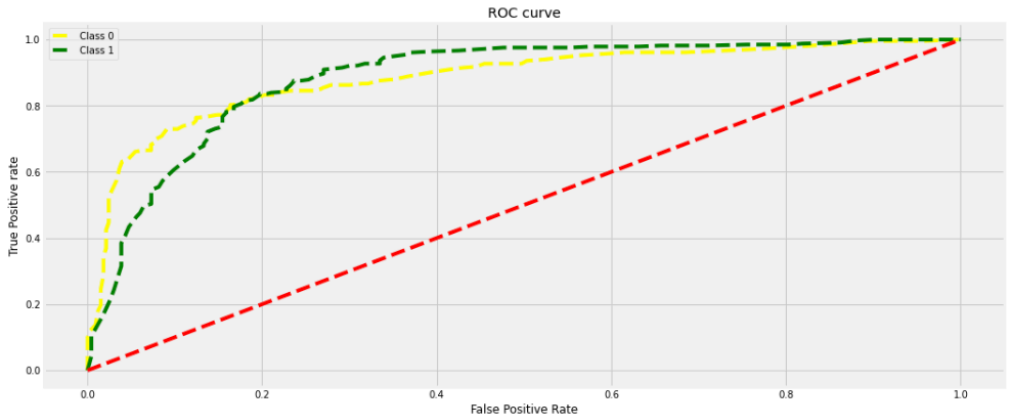


Figure 99:Random Forest model on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.80	0.77	0.78	233
1	0.84	0.86	0.85	330
accuracy			0.82	563
macro avg	0.82	0.81	0.82	563
weighted avg	0.82	0.82	0.82	563





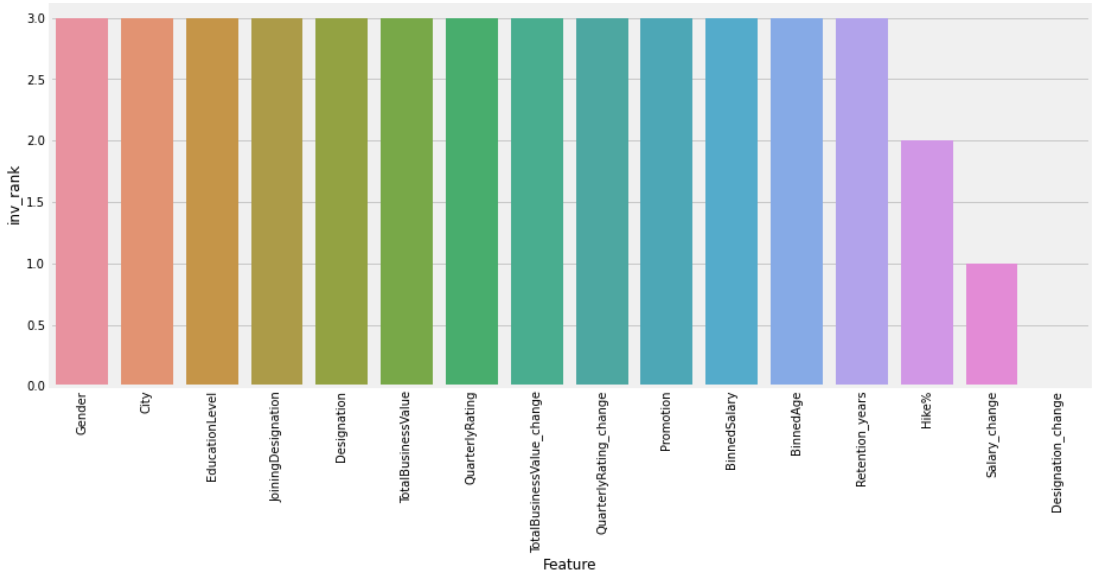
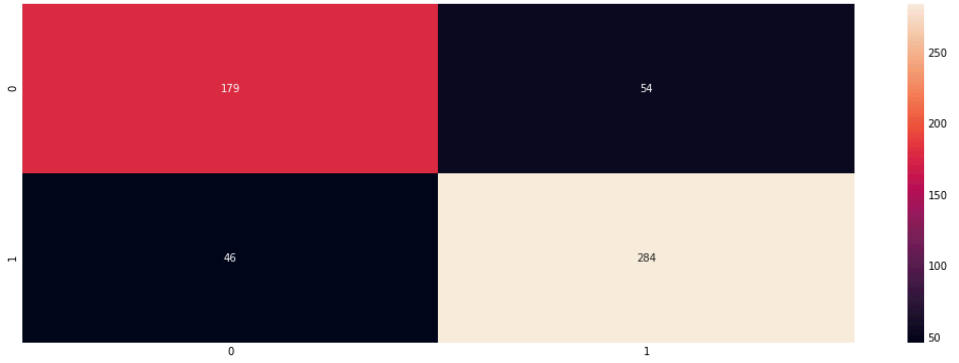
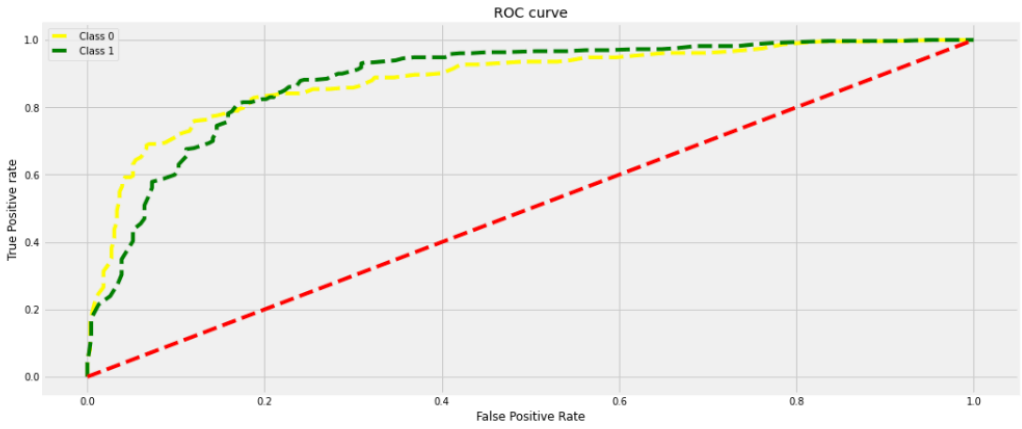


Figure 100: Random Forest model on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.80	0.77	0.78	233
1	0.84	0.86	0.85	330
accuracy			0.82	563
macro avg	0.82	0.81	0.82	563
weighted avg	0.82	0.82	0.82	563





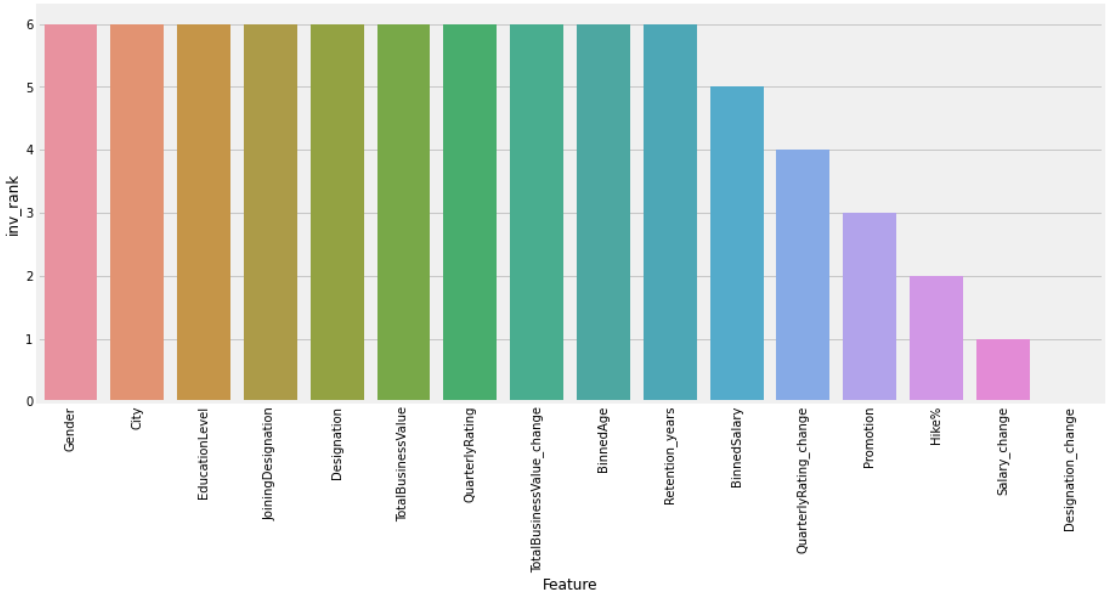


Figure 101: Random Forest model on Minmaxscaled Balanced Binned Derived Data

20.8. Model 7: Gradient Boosting Classifier

Gradient Boosting is a very powerful Ensemble Machine Learning algorithm. It works to combine many weak learning models together resulting in a strong predictive model. Usually, Decision trees are used when implementing gradient boosting. (Scikit Learn sklearn.ensemble.GradientBoostingClassifier, n.d.)

To conclude the model parameters corresponding to the best model performance Randomized Search CV was implemented on different datasets.

```
gbc_rand_params_={}
for name, data in datasets:
    print('Dataset:',name)
    param = RSCV_GBC(data)
    gbc_rand_params_[name] = param
```

```
Dataset: Balanced Derived Data
Best learning_rate: 0.06099999999999999
Best max_depth: 10
Best loss: deviance
Best criterion: mse
Best min_samples_leaf: 8
Best random_state: None
Best max_leaf_nodes: 6
Best Accuracy Score: 0.852

{'ccp_alpha': 0.0, 'criterion': 'mse', 'init': None, 'learning_rate': 0.06099999999999999, 'loss': 'deviance', 'max_d
epth': 10, 'max_features': None, 'max_leaf_nodes': 6, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 8, 'min_sampl
es_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_iter_no_change': None, 'random_state': None,
'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False}
-----
Dataset: Balanced Binned Derived Data
Best learning_rate: 0.08099999999999999
Best max_depth: 12
Best loss: deviance
Best criterion: mse
Best min_samples_leaf: 6
Best random_state: None
Best max_leaf_nodes: 12
Best Accuracy Score: 0.8560000000000001

{'ccp_alpha': 0.0, 'criterion': 'mse', 'init': None, 'learning_rate': 0.08099999999999999, 'loss': 'deviance', 'max_d
epth': 12, 'max_features': None, 'max_leaf_nodes': 12, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 6, 'min_samp
les_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_iter_no_change': None, 'random_state': None,
'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False}
-----
Dataset: Stdscaled Balanced Binned Derived Data
Best learning_rate: 0.05099999999999999
Best max_depth: 17
Best loss: deviance
Best criterion: squared_error
Best min_samples_leaf: 13
Best random_state: None
Best max_leaf_nodes: 12
Best Accuracy Score: 0.852

{'ccp_alpha': 0.0, 'criterion': 'squared_error', 'init': None, 'learning_rate': 0.05099999999999999, 'loss': 'devianc
e', 'max_depth': 17, 'max_features': None, 'max_leaf_nodes': 12, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1
3, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_iter_no_change': None, 'random_st
ate': None, 'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False}
-----
Dataset: Minmax scaled Balanced Binned Derived Data
Best learning_rate: 0.16099999999999998
Best max_depth: 17
Best loss: deviance
Best criterion: mse
Best min_samples_leaf: 6
Best random_state: None
Best max_leaf_nodes: 17
Best Accuracy Score: 0.8506666666666668

{'ccp_alpha': 0.0, 'criterion': 'mse', 'init': None, 'learning_rate': 0.16099999999999998, 'loss': 'deviance', 'max_d
epth': 17, 'max_features': None, 'max_leaf_nodes': 17, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 6, 'min_samp
les_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_iter_no_change': None, 'random_state': None,
'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False}
```

Figure 102: Gradient Boosting hyperparamter with RandomizedSearchCV

20.8.1. Gradient Boosting Model Parameters

- **criterion** – the function to measure the quality of the split
- **loss** – the loss function to be optimized
- **max_depth** – the maximum depth limits the number of nodes in the tree
- **n_estimators** – the number of boosting stages to perform
- **min_samples_leaf** – minimum number of samples required to be at the leaf node
- **max_leaf_nodes** – growing tree with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity.
- **learning_rate** – learning_rate shrinks the contribution of each tree by learning_rate

20.8.2. Gradient Boosting plots for various datasets

Best parameters concluded from Randomised Search CV on Gradient Boosting were used for model building. (Scikit Learn sklearn.ensemble.GradientBoostingClassifier, n.d.) (Data Analystics Learning Curves Explained with Python Sklearn Example, n.d.)

```
Dataset: Balanced Derived Data
Accuracy:0.86
Precision:0.86
Recall:0.91
F1-Score:0.88
```

	precision	recall	f1-score	support
0	0.86	0.78	0.82	155
1	0.86	0.91	0.88	221
accuracy			0.86	376
macro avg	0.86	0.85	0.85	376
weighted avg	0.86	0.86	0.86	376

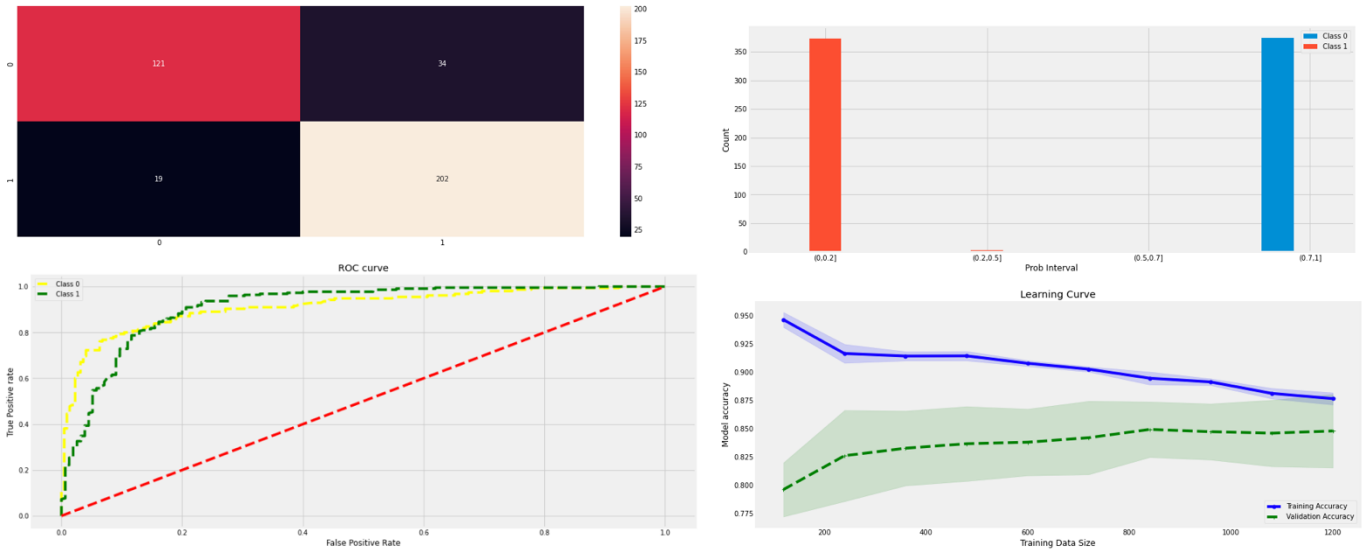


Figure 103:Gradient Boosting model on Balanced Derived Data

Dataset: Balanced Binned Derived Data
Accuracy:0.88
Precision:0.88
Recall:0.93
F1-Score:0.90

	precision	recall	f1-score	support
0	0.89	0.81	0.85	155
1	0.88	0.93	0.90	221
accuracy			0.88	376
macro avg	0.89	0.87	0.88	376
weighted avg	0.88	0.88	0.88	376

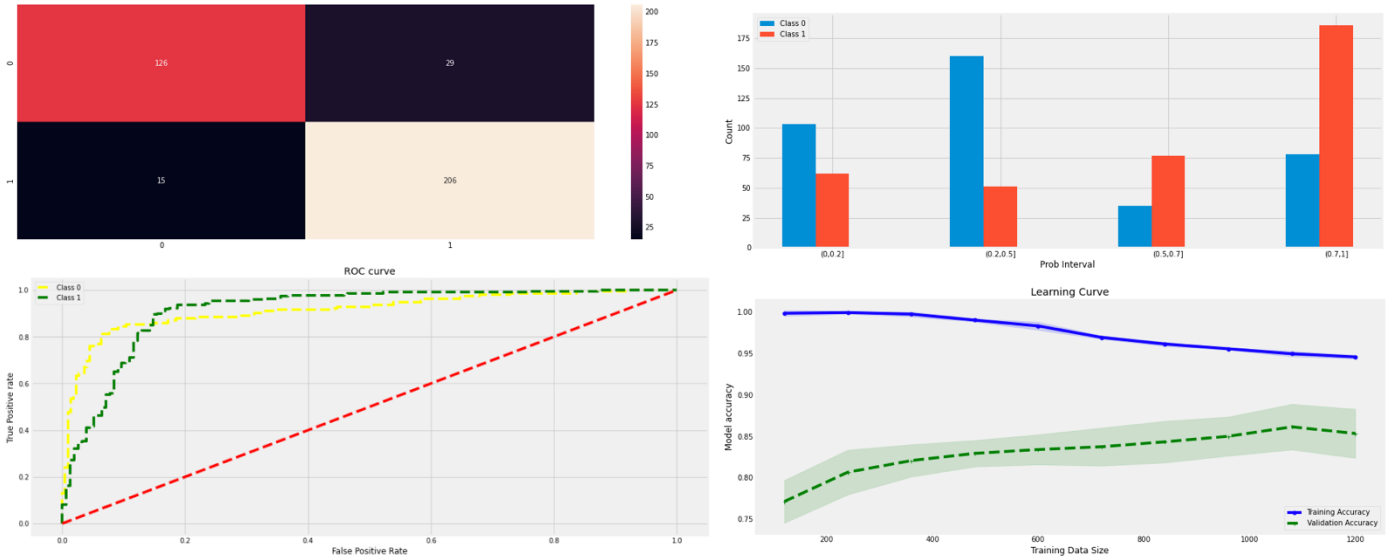


Figure 104:Gradient Boosting model on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data
Accuracy:0.86
Precision:0.85
Recall:0.92
F1-Score:0.88

	precision	recall	f1-score	support
0	0.87	0.77	0.82	155
1	0.85	0.92	0.88	221
accuracy			0.86	376
macro avg	0.86	0.84	0.85	376
weighted avg	0.86	0.86	0.85	376

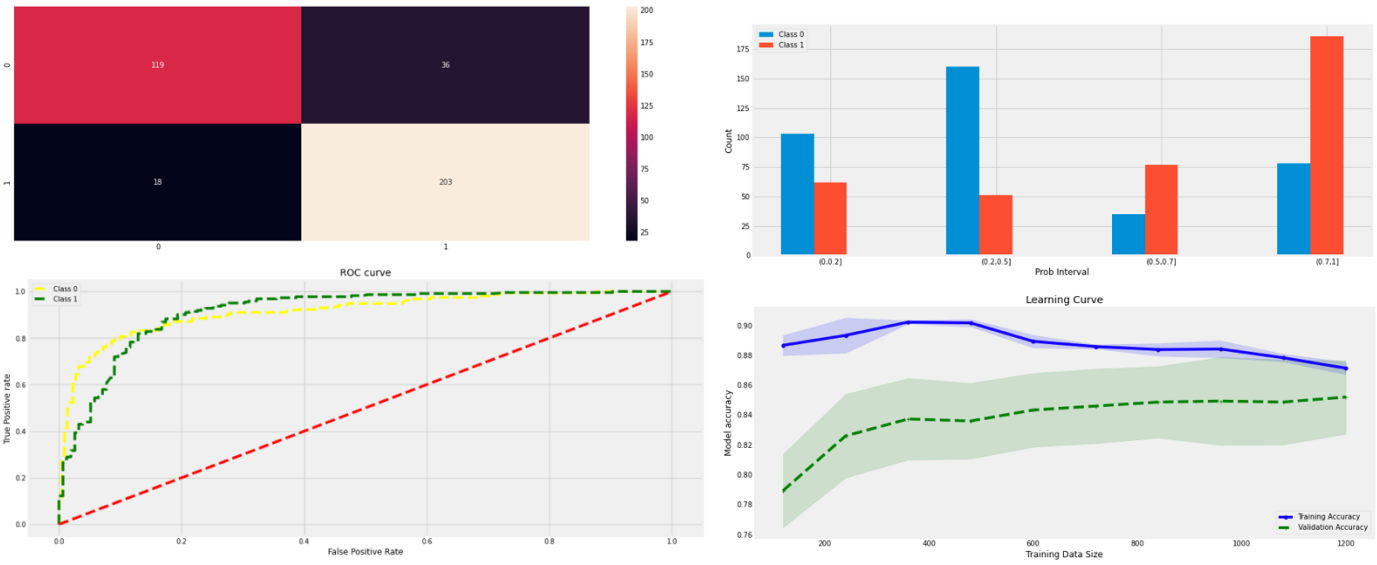


Figure 105:Gradient Boosting model on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data
Accuracy:0.87
Precision:0.87
Recall:0.92
F1-Score:0.89

	precision	recall	f1-score	support
0	0.87	0.80	0.84	155
1	0.87	0.92	0.89	221
accuracy			0.87	376
macro avg	0.87	0.86	0.86	376
weighted avg	0.87	0.87	0.87	376

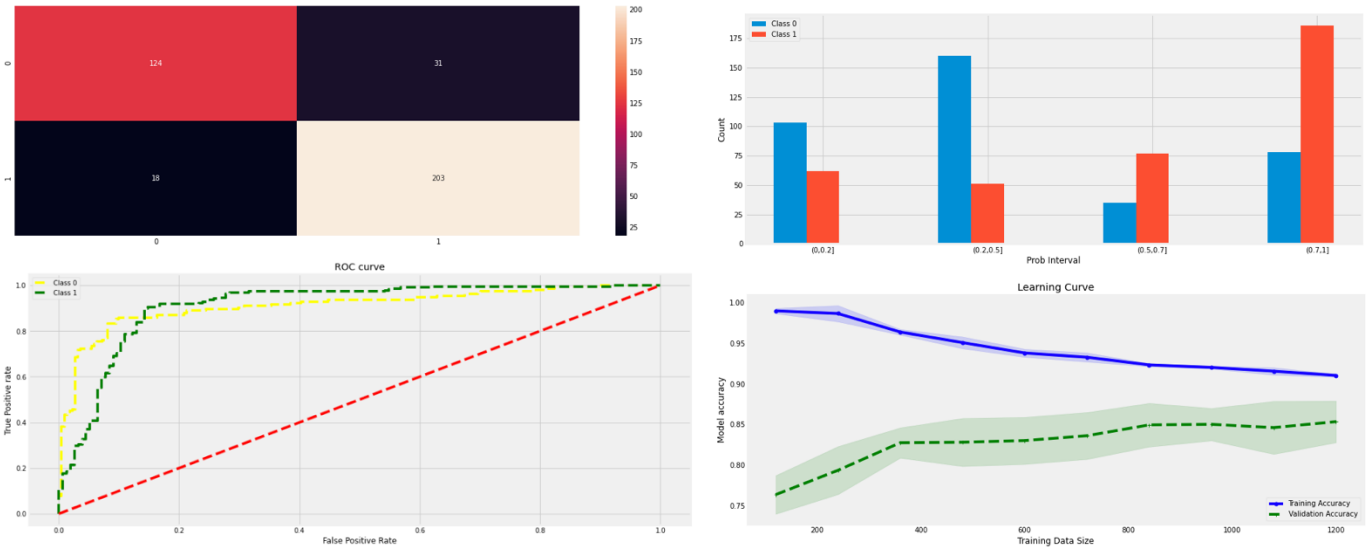
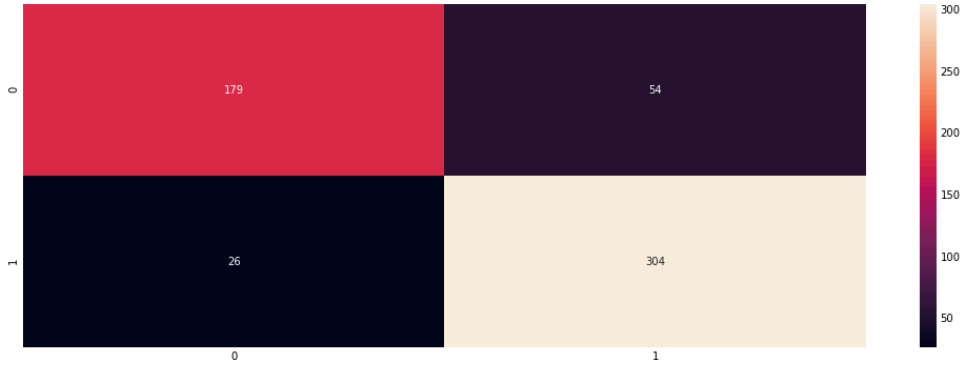


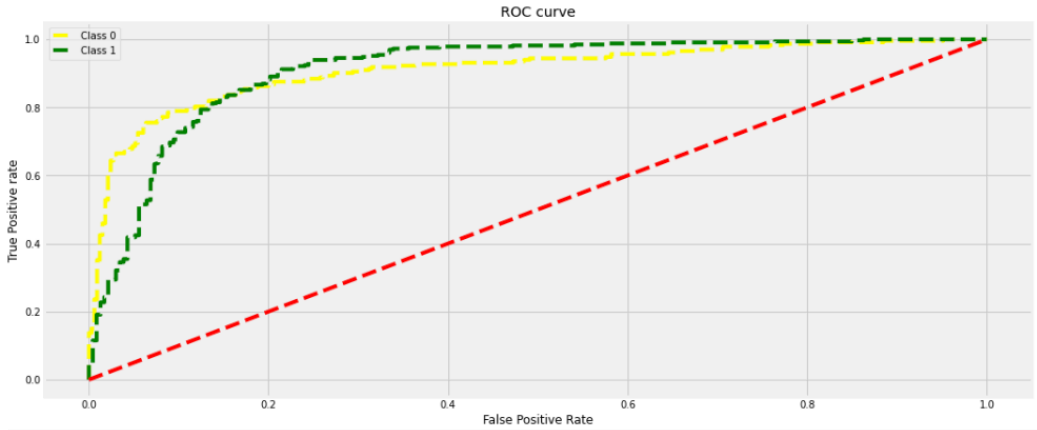
Figure 106:Gradient Boosting model on Minmaxscaled Balanced Binned Derived Data

20.8.3. Gradient Boosting with RFECV

Dataset: Balanced Derived Data

	precision	recall	f1-score	support
0	0.87	0.77	0.82	233
1	0.85	0.92	0.88	330
accuracy			0.86	563
macro avg	0.86	0.84	0.85	563
weighted avg	0.86	0.86	0.86	563





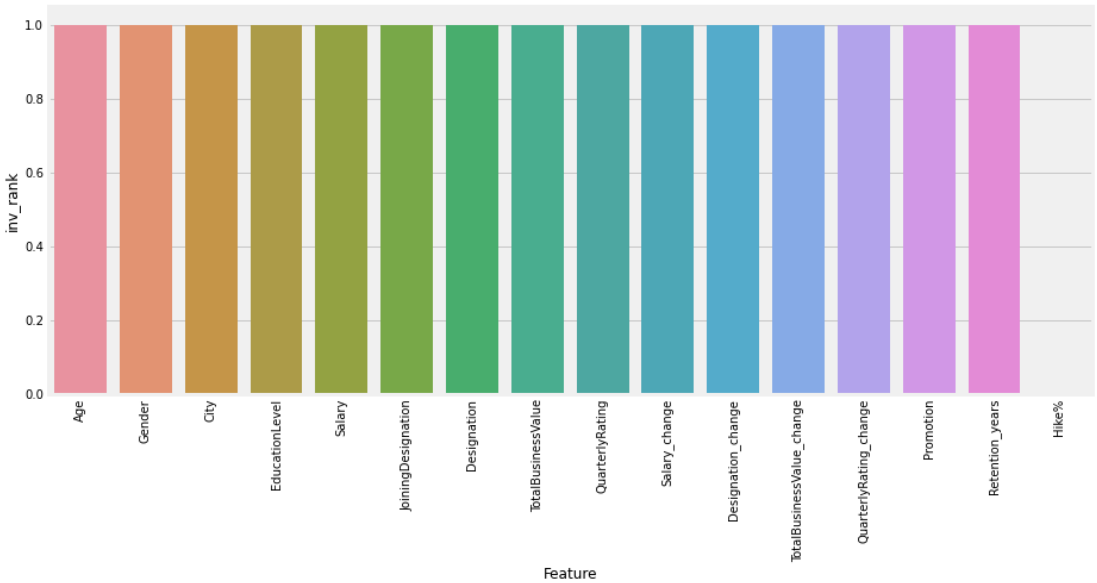
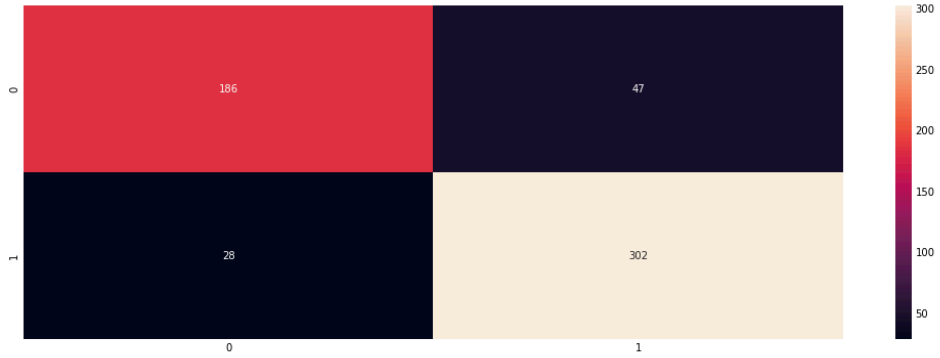
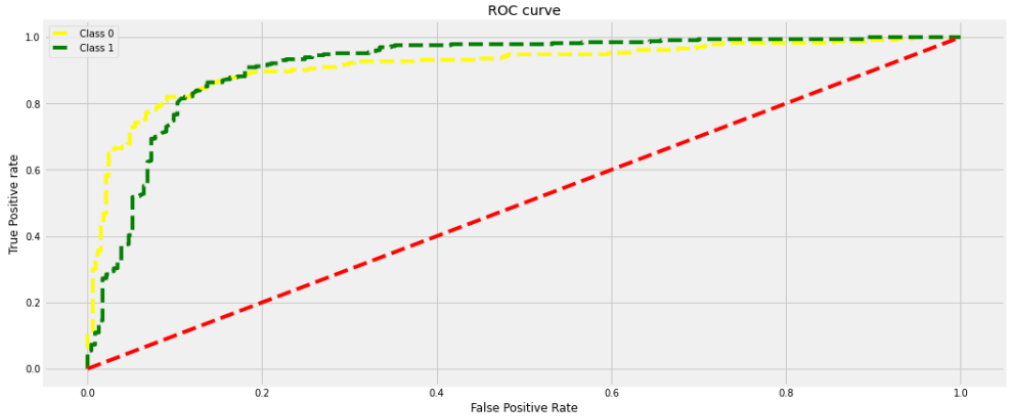


Figure 107:GradientBoosting model with RFECV on Balanced Derived Data

Dataset: Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.87	0.80	0.83	233
1	0.87	0.92	0.89	330
accuracy			0.87	563
macro avg	0.87	0.86	0.86	563
weighted avg	0.87	0.87	0.87	563





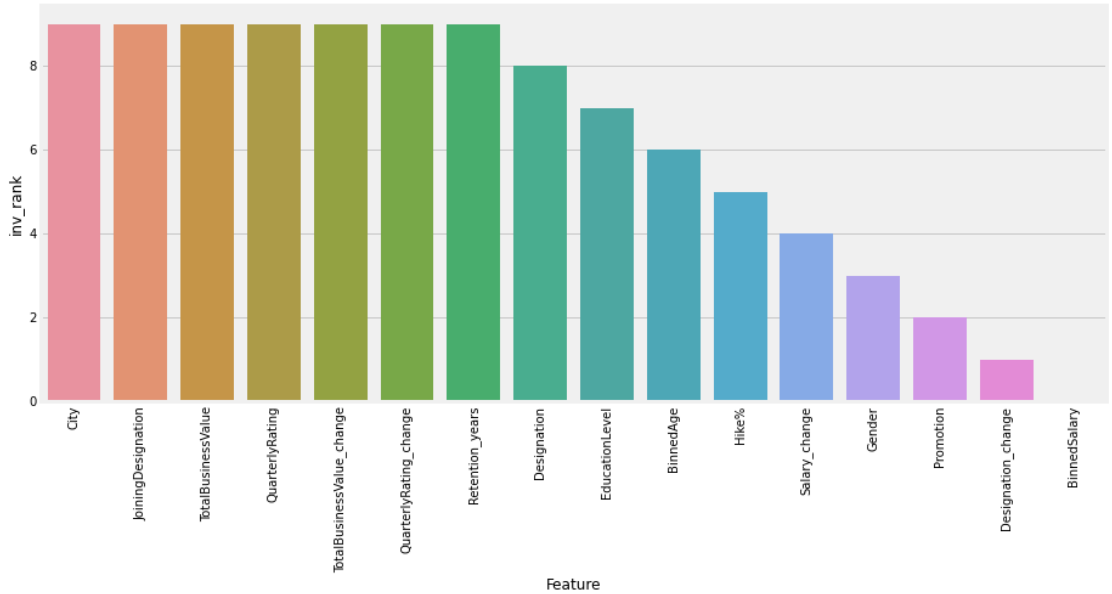
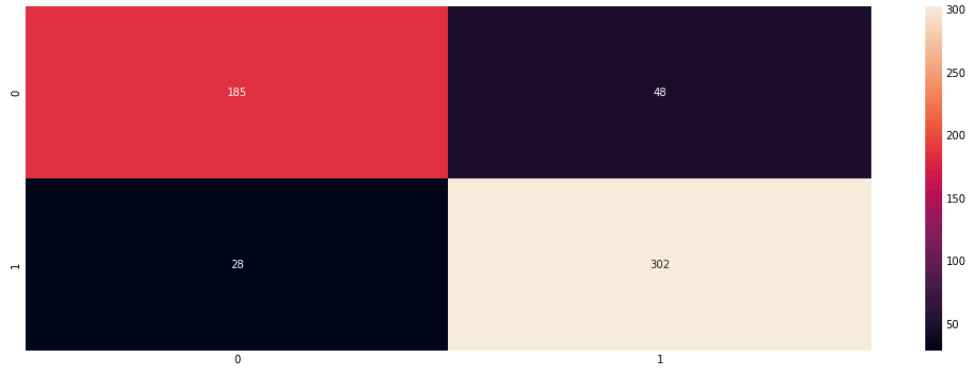
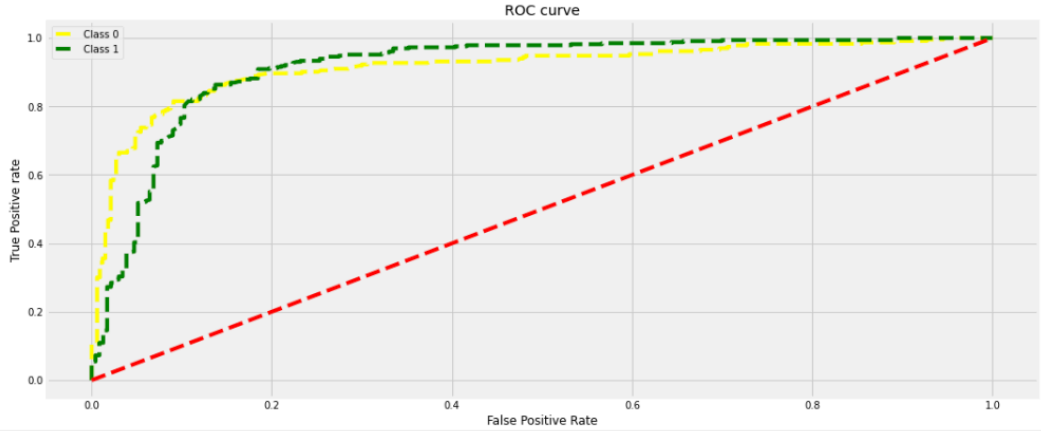


Figure 108:GradientBoosting model with RFECV on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.87	0.79	0.83	233
1	0.86	0.92	0.89	330
accuracy			0.87	563
macro avg	0.87	0.85	0.86	563
weighted avg	0.87	0.87	0.86	563





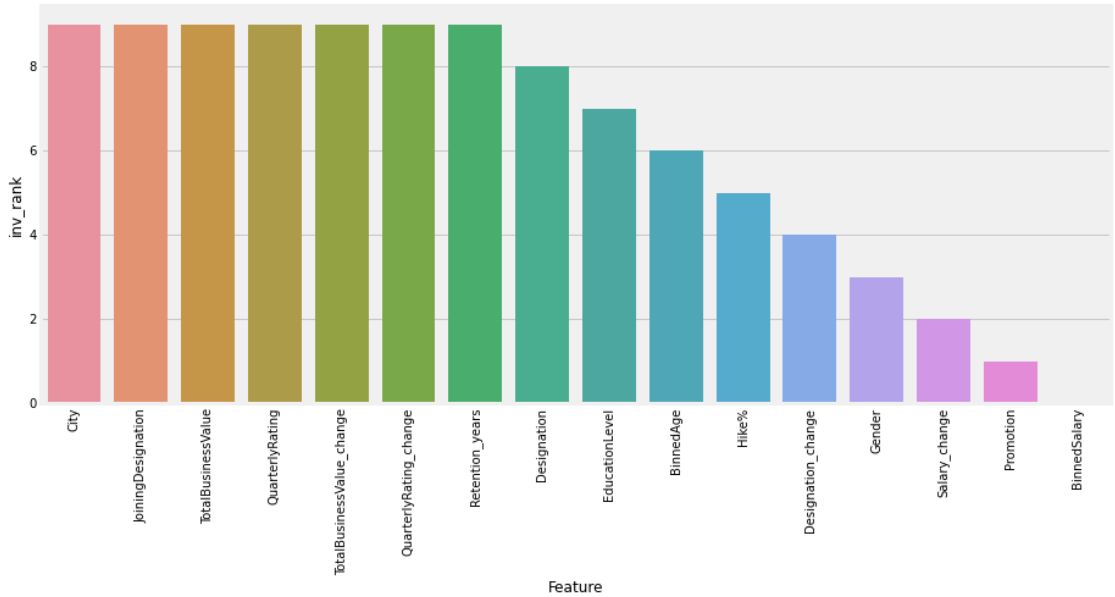
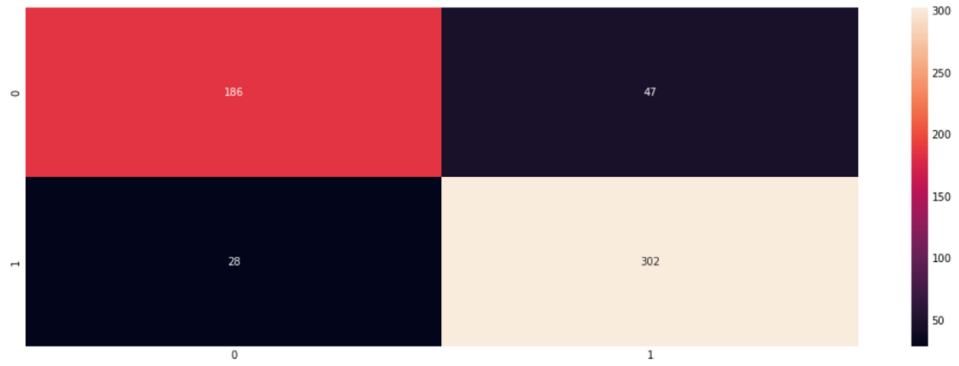
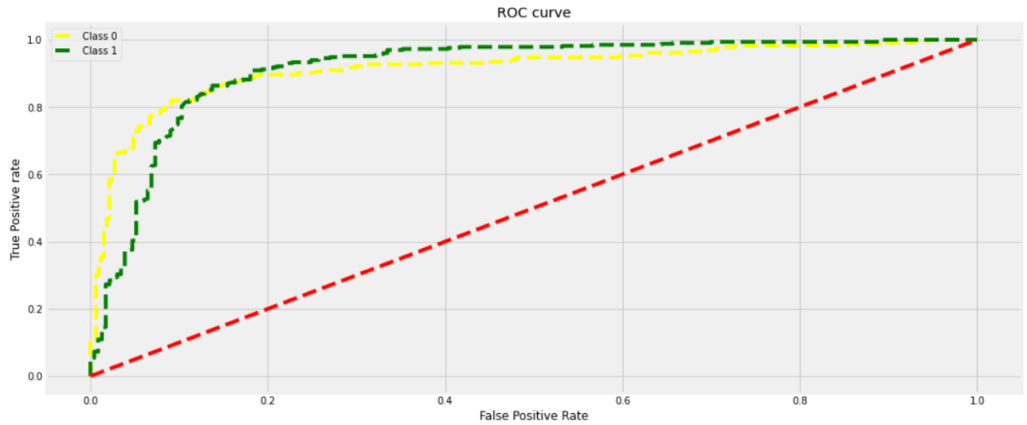


Figure 109:GradientBoosting model with RFECV on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.87	0.80	0.83	233
1	0.87	0.92	0.89	330
accuracy			0.87	563
macro avg	0.87	0.86	0.86	563
weighted avg	0.87	0.87	0.87	563





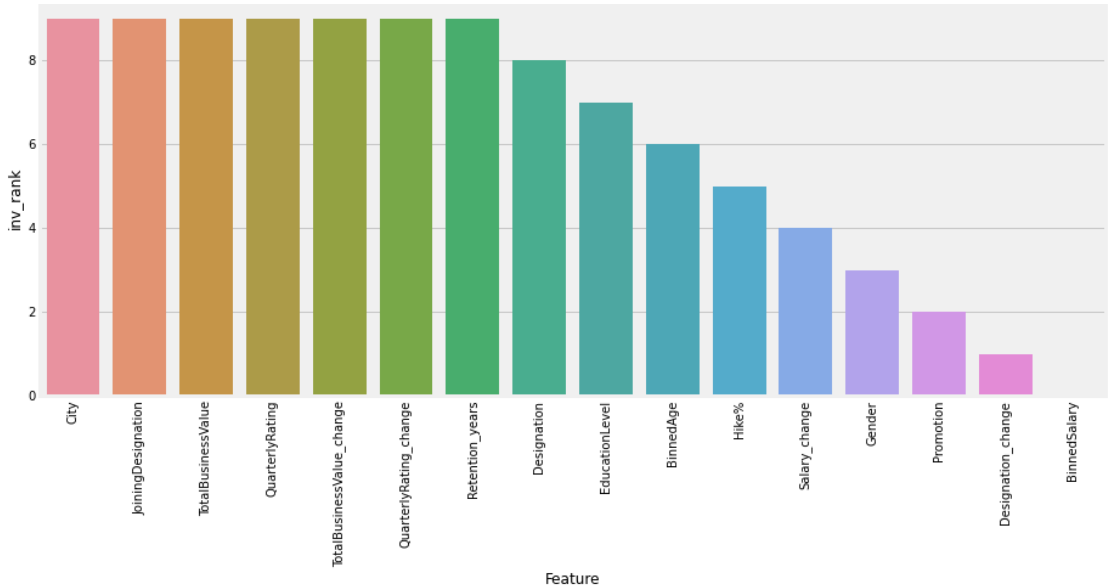


Figure 110: GradientBoosting model with RFECV on Minmaxscaled Balanced Binned Derived Data

20.9. Model 8: AdaBoost Classifier

AdaBoost or Adaptive Boosting is a Boosting technique used as an Ensemble method in Machine Learning. It can be used for both Regression and Classification problems. It works by re-assigning weights to each instance in successive iterations such that the incorrectly classified instances are assigned higher weights in next iteration. This weight adjustment helps dealing with the difficult instances thus creating a strong predictive model. (Scikit Learn sklearn.ensemble.AdaBoostClassifier, n.d.)

To conclude the model parameters corresponding to the best model performance Randomized Search CV was implemented on different datasets.

```
abc_rand_params_={}
for name, data in datasets:
    print('Dataset:',name)
    param = RSCV_ABC(data)
    abc_rand_params_[name] = param
```

```
Dataset: Balanced Derived Data
Best algorithm: SAMME
Best n_estimators: 667
Best random_state: 10
Best learning_rate: 0.36099999999999993
Best Accuracy Score: 0.8513333333333334

{'algorithm': 'SAMME', 'base_estimator': None, 'learning_rate': 0.36099999999999993, 'n_estimators': 667, 'random_sta
te': 10}
-----
Dataset: Balanced Binned Derived Data
Best algorithm: SAMME.R
Best n_estimators: 118
Best random_state: 13
Best learning_rate: 0.4709999999999999
Best Accuracy Score: 0.8533333333333333

{'algorithm': 'SAMME.R', 'base_estimator': None, 'learning_rate': 0.4709999999999999, 'n_estimators': 118, 'random_st
ate': 13}
-----
Dataset: Stdscaled Balanced Binned Derived Data
Best algorithm: SAMME.R
Best n_estimators: 299
Best random_state: 17
Best learning_rate: 0.141
Best Accuracy Score: 0.8533333333333333

{'algorithm': 'SAMME.R', 'base_estimator': None, 'learning_rate': 0.141, 'n_estimators': 299, 'random_state': 17}
-----
Dataset: Minmax scaled Balanced Binned Derived Data
Best algorithm: SAMME.R
Best n_estimators: 131
Best random_state: 4
Best learning_rate: 0.5909999999999999
Best Accuracy Score: 0.8539999999999999

{'algorithm': 'SAMME.R', 'base_estimator': None, 'learning_rate': 0.5909999999999999, 'n_estimators': 131, 'random_st
ate': 4}
-----
```

Figure 111:AdaBoost hyperparamter with RandomizedSearchCV

20.9.1. AdaBoost Model Paramters

- **base_estimator** – the base estimator from which the boosting ensemble is built. If none, then base_estimator is initialized with Decision Tree Classifier with max_depth = 1
- **algorithm** – if 'SAMME.R' then use SAMME.R real boosting algorithm, if 'SAMME' then use SAMME discrete boosting algorithm. SAMME.R algorithm converges faster than SAMME.
- **n_estimator** – maximum number of estimators at which boosting is terminated. In case of perfect fit, learning procedure is stopped early
- **random_state** – controls the random seeds given at each base_estimator at each

boosting iteration

- **learning_rate** – weight applied to each classifier at each boosting iteration

20.9.2. AdaBoost plots for various datasets

Best parameters concluded from Randomised Search CV on Gradient Boosting were used for model building. (Scikit Learn sklearn.ensemble.AdaBoostClassifier, n.d.) (Data Analystics Learning Curves Explained with Python Sklearn Example, n.d.)

```
Dataset: Balanced Derived Data
Accuracy:0.85
Precision:0.85
Recall:0.90
F1-Score:0.87
```

	precision	recall	f1-score	support
0	0.84	0.77	0.80	155
1	0.85	0.90	0.87	221
accuracy			0.85	376
macro avg	0.85	0.83	0.84	376
weighted avg	0.85	0.85	0.84	376

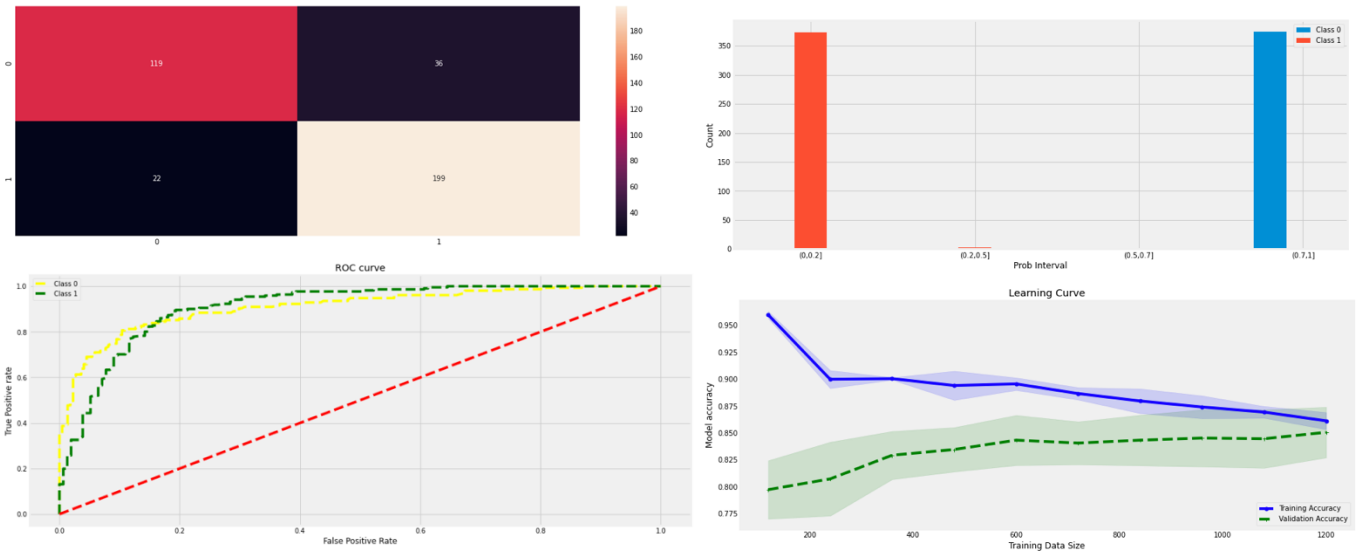


Figure 112:Adaboost model on Balanced Derived Data

Dataset: Balanced Binned Derived Data
Accuracy:0.85
Precision:0.85
Recall:0.91
F1-Score:0.88

	precision	recall	f1-score	support
0	0.86	0.77	0.81	155
1	0.85	0.91	0.88	221
accuracy			0.85	376
macro avg	0.85	0.84	0.85	376
weighted avg	0.85	0.85	0.85	376

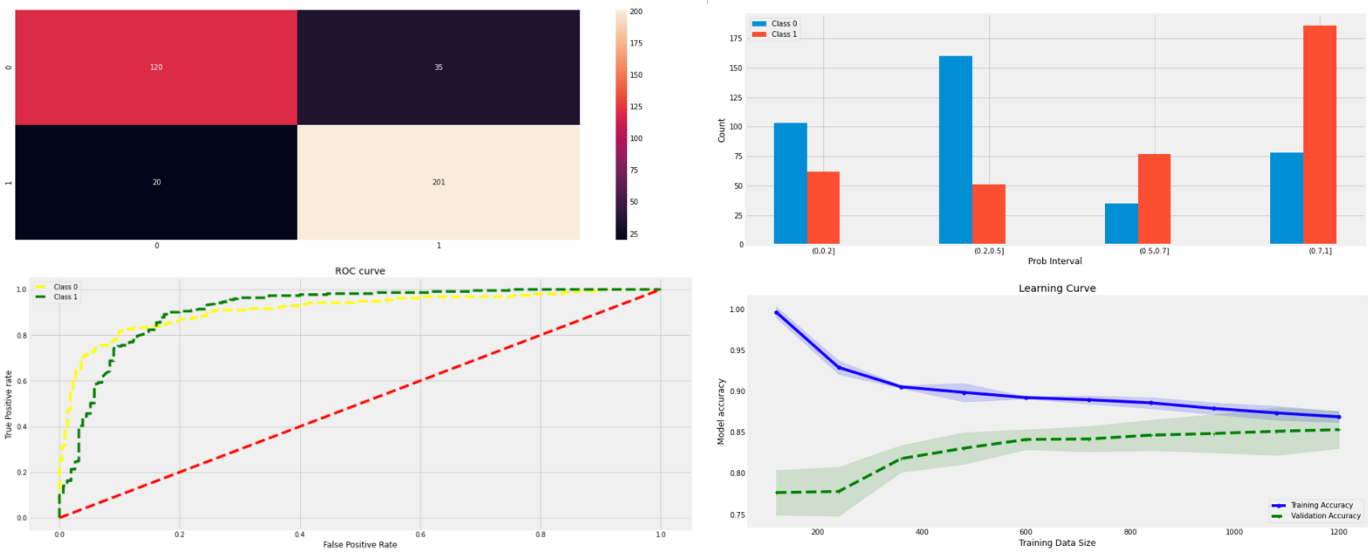


Figure 113:Adaboost model on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data
Accuracy:0.87
Precision:0.87
Recall:0.92
F1-Score:0.89

	precision	recall	f1-score	support
0	0.88	0.80	0.84	155
1	0.87	0.92	0.89	221
accuracy			0.87	376
macro avg	0.87	0.86	0.87	376
weighted avg	0.87	0.87	0.87	376

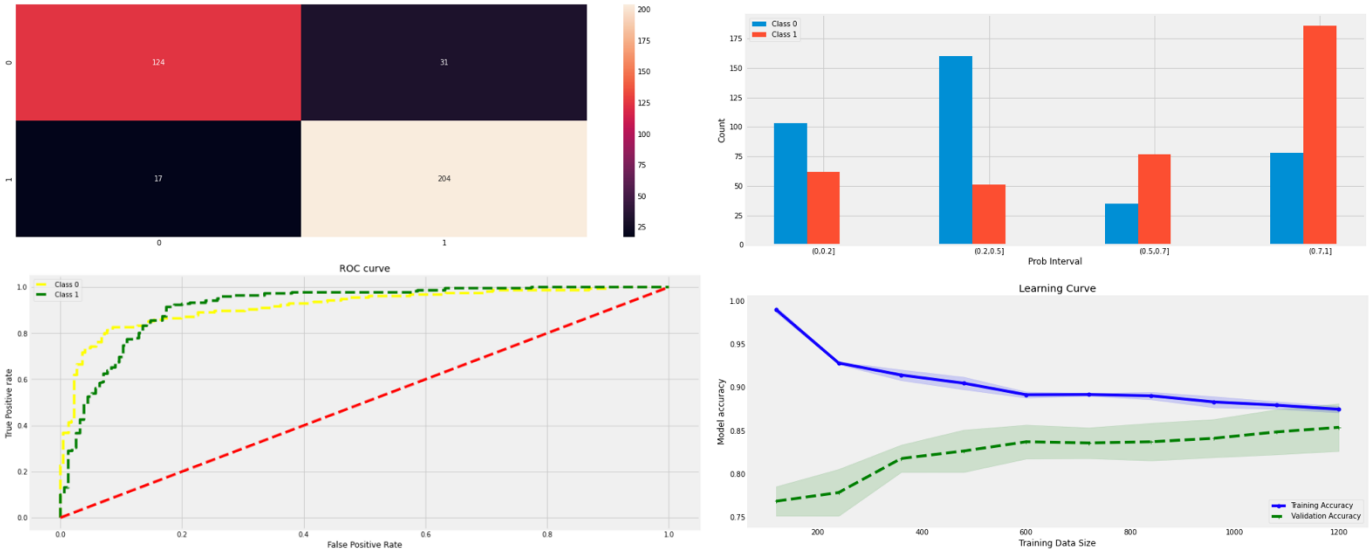


Figure 114:AdaBoost model on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data
Accuracy:0.86
Precision:0.85
Recall:0.92
F1-Score:0.89

	precision	recall	f1-score	support
0	0.88	0.77	0.82	155
1	0.85	0.92	0.89	221
accuracy			0.86	376
macro avg	0.86	0.85	0.85	376
weighted avg	0.86	0.86	0.86	376

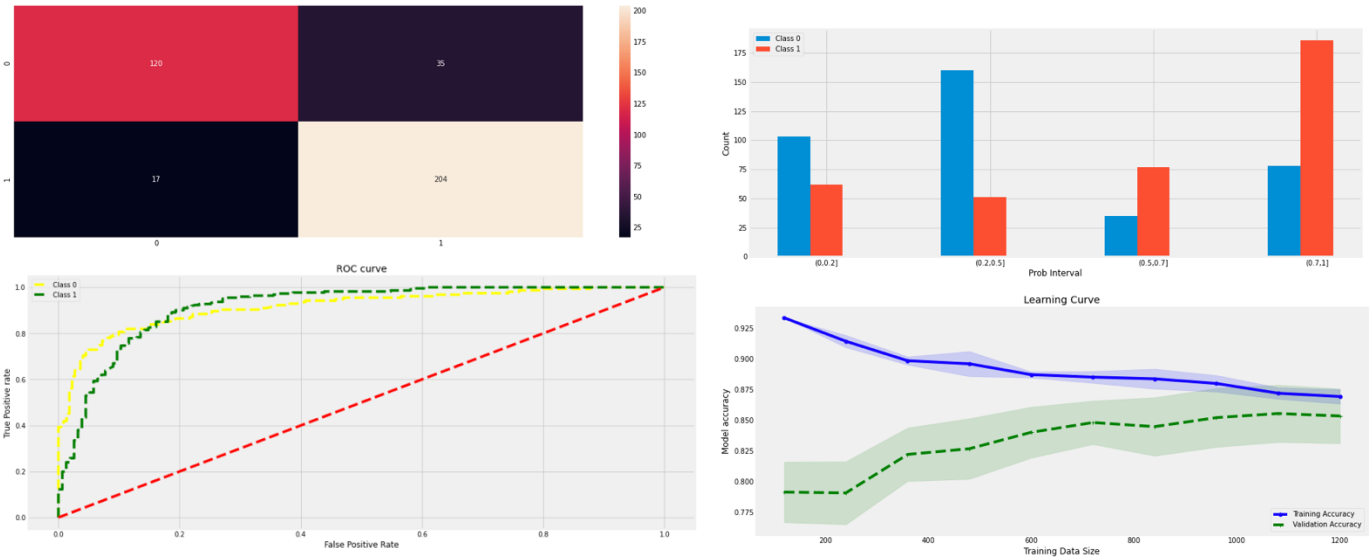
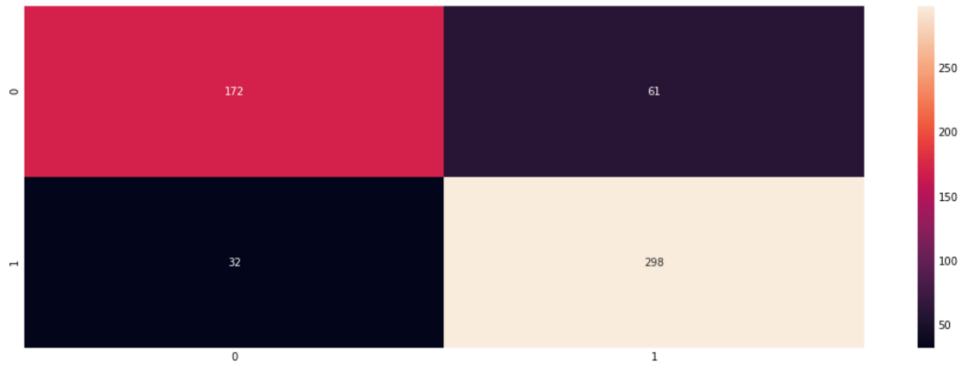


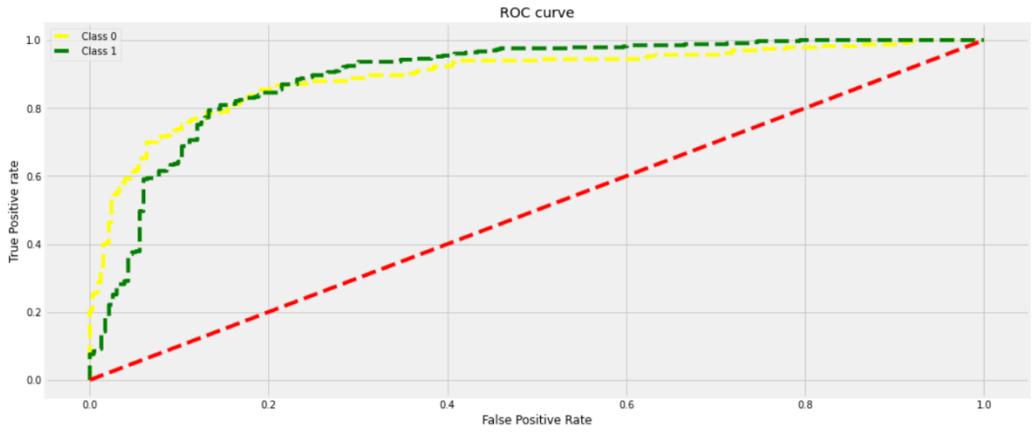
Figure 115: AdaBoost model on Minmaxscaled Balanced Binned Derived Data

20.9.3. AdaBoost with RFECV
(Scikit Learn sklearn.feature_selection.RFECV, n.d.)

Dataset: Balanced Derived Data

	precision	recall	f1-score	support
0	0.84	0.74	0.79	233
1	0.83	0.90	0.87	330
accuracy			0.83	563
macro avg	0.84	0.82	0.83	563
weighted avg	0.84	0.83	0.83	563





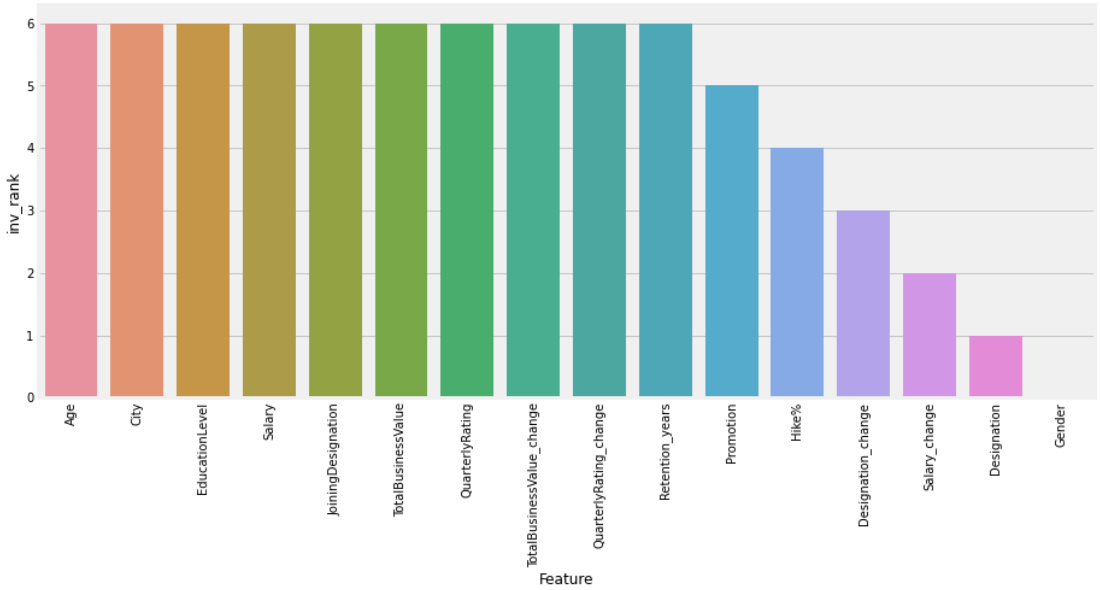
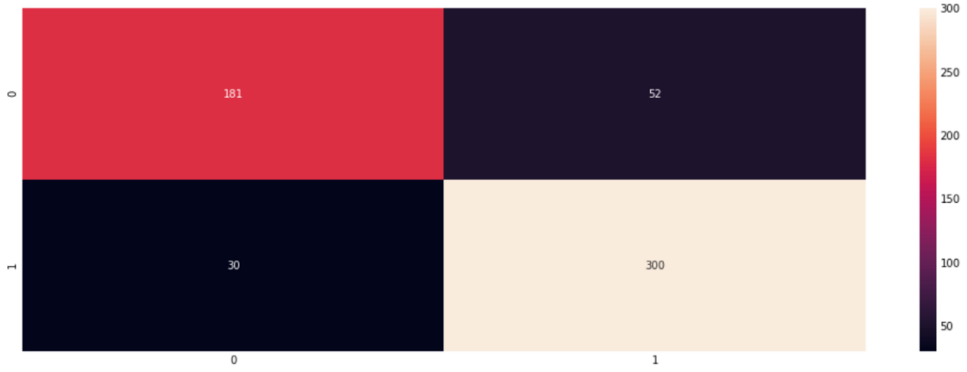
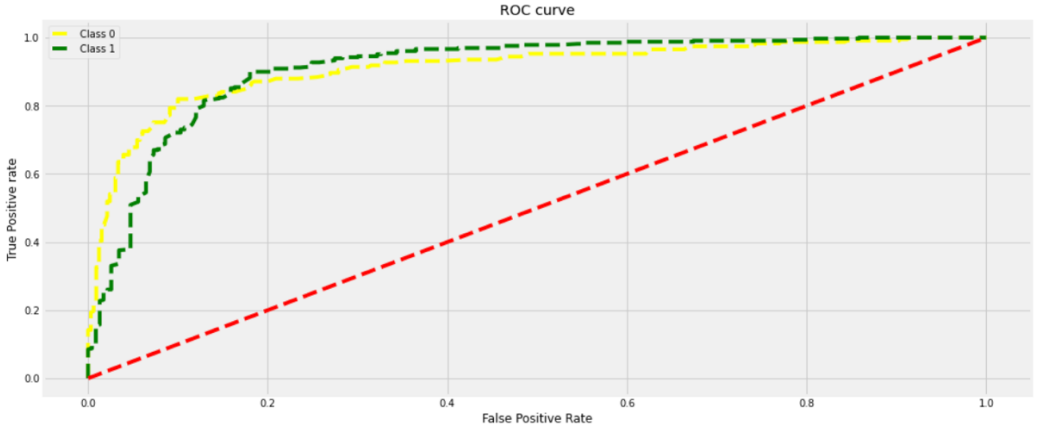


Figure 116: AdaBoost mode with RFECV on Balanced Derived Data

Dataset: Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.86	0.78	0.82	233
1	0.85	0.91	0.88	330
accuracy			0.85	563
macro avg	0.86	0.84	0.85	563
weighted avg	0.85	0.85	0.85	563





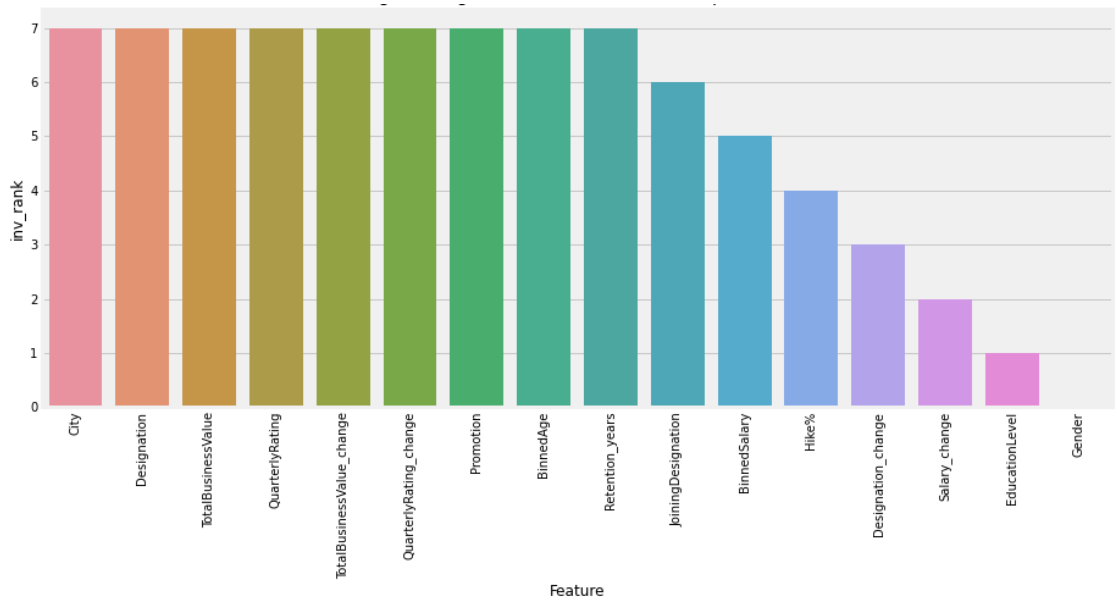
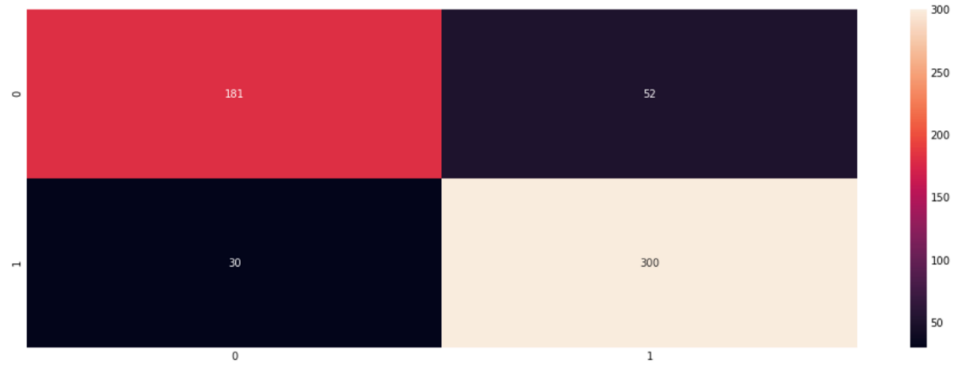
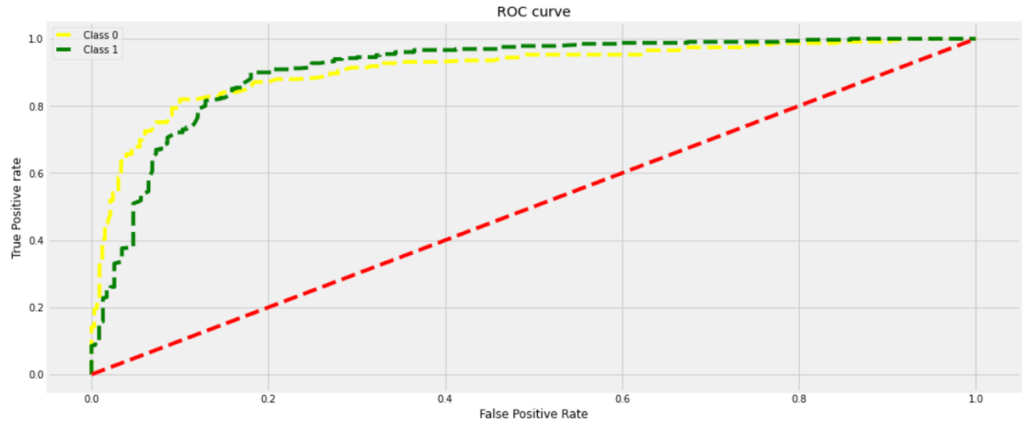


Figure 117:AdaBoost model with RFECV on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.86	0.78	0.82	233
1	0.85	0.91	0.88	330
accuracy			0.85	563
macro avg	0.86	0.84	0.85	563
weighted avg	0.85	0.85	0.85	563





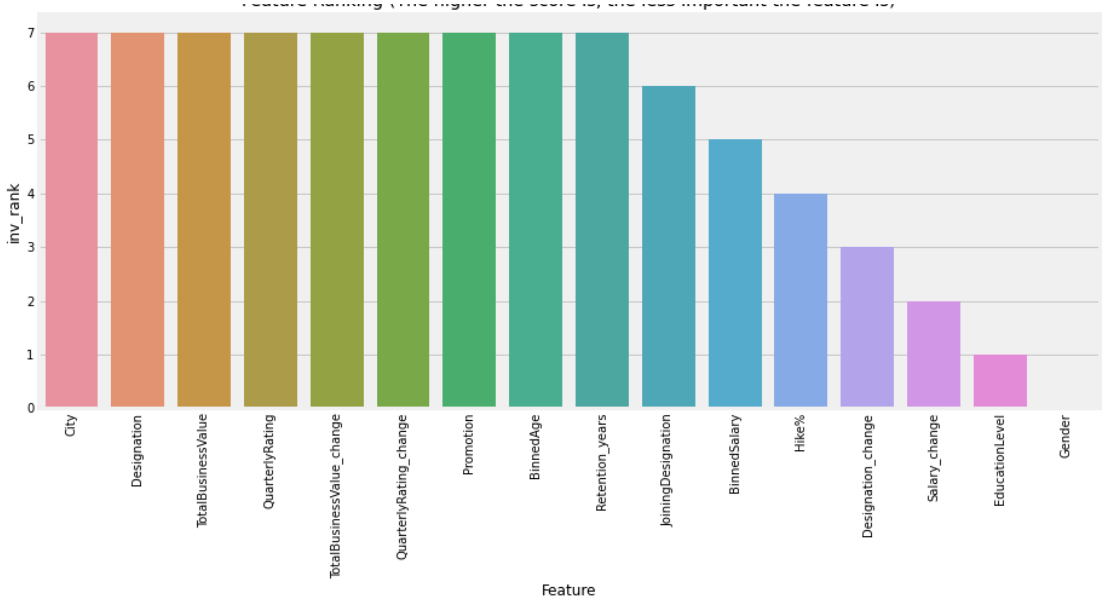
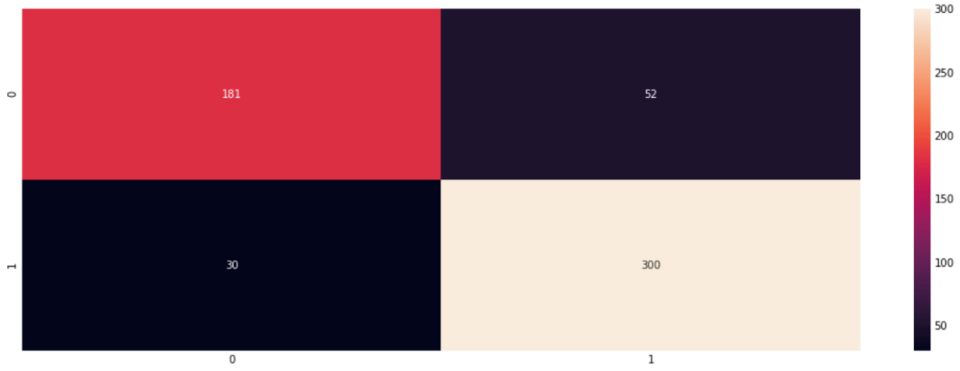
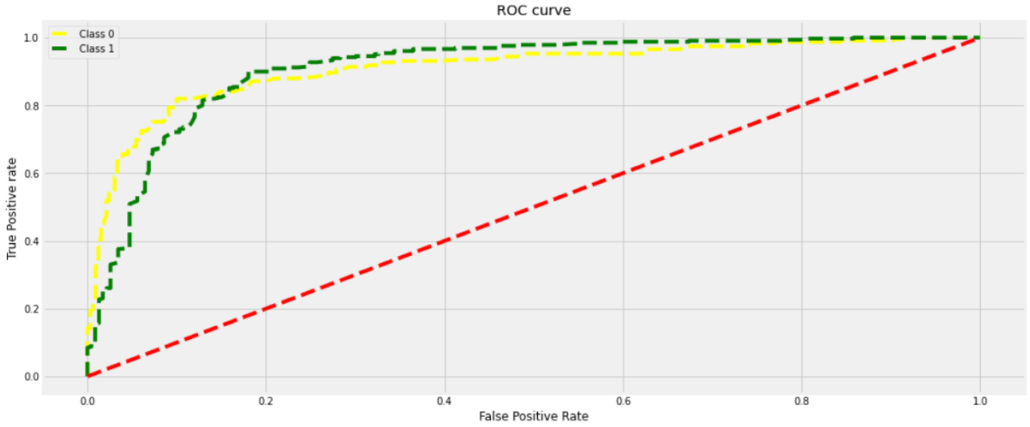


Figure 118:AdaBoost model with RFECV on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data

	precision	recall	f1-score	support
0	0.86	0.78	0.82	233
1	0.85	0.91	0.88	330
accuracy			0.85	563
macro avg	0.86	0.84	0.85	563
weighted avg	0.85	0.85	0.85	563





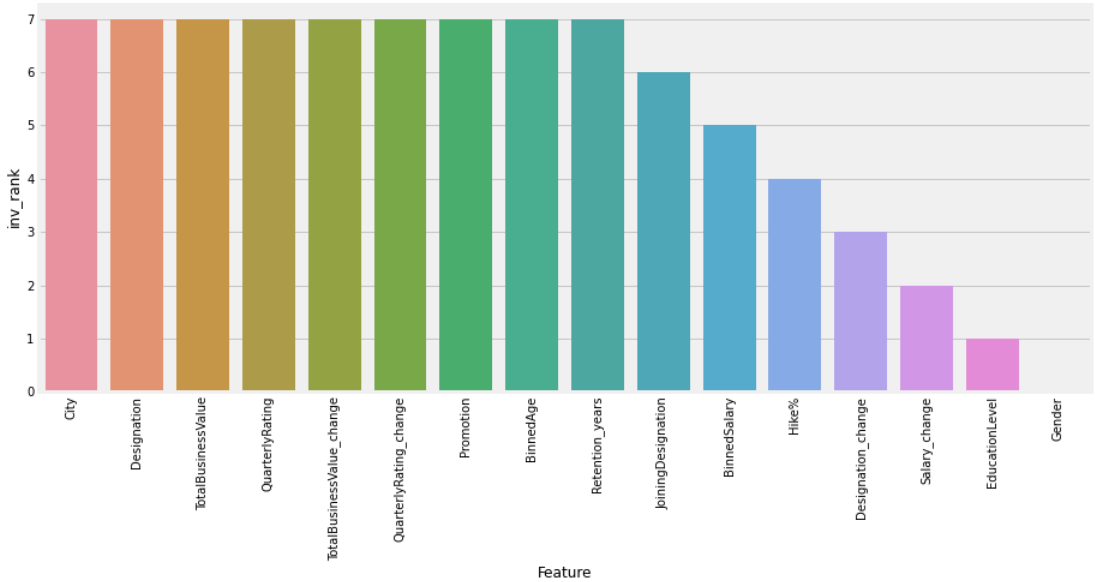


Figure 119:AdaBoost model with RFECV on Minmaxscaled Balanced Binned Derived Data

20.10. **Model 9: Voting Classifier**

Voting Classifier is a Machine Learning model that trains on an ensemble of multiple base models or estimators and makes prediction based on aggregating the findings of each base model. The aggregating criteria can be combined decision of voting for each estimator output. (Scikit Learn sklearn.ensemble.VotingClassifier, n.d.)

20.10.1. Voting Classifier Model Parameters

- **estimators** – list of (str, estimator) tuples that stores the models or estimators to be used in modelling
- **voting** – if 'hard' uses predicted class label for majority rule voting, else if 'soft' predicts the class label based on the argmax of sum of predicted probabilities

20.10.2. Voting Classifier plots for various datasets

Used a combination of following all models built with the best parameters concluded from Randomised Search CV for Voting Classifier –

- Logistic Regression
- Decision Tree
- KNeighborsClassifier
- Bernoulli NB
- Gaussian NB
- Random Forest
- Gradient Boosting
- Adaboost

Model was built on various datasets with the following results. (Scikit Learn sklearn.ensemble.VotingClassifier, n.d.) (Data Analystics Learning Curves Explained with Python Sklearn Example, n.d.)

```
Dataset: Balanced Derived Data
Accuracy:0.77
Precision:0.74
Recall:0.92
F1-Score:0.82
              precision    recall  f1-score   support

           0       0.83      0.54      0.66       155
           1       0.74      0.92      0.82       221

    accuracy                           0.77       376
   macro avg       0.79      0.73      0.74       376
weighted avg       0.78      0.77      0.75       376
```

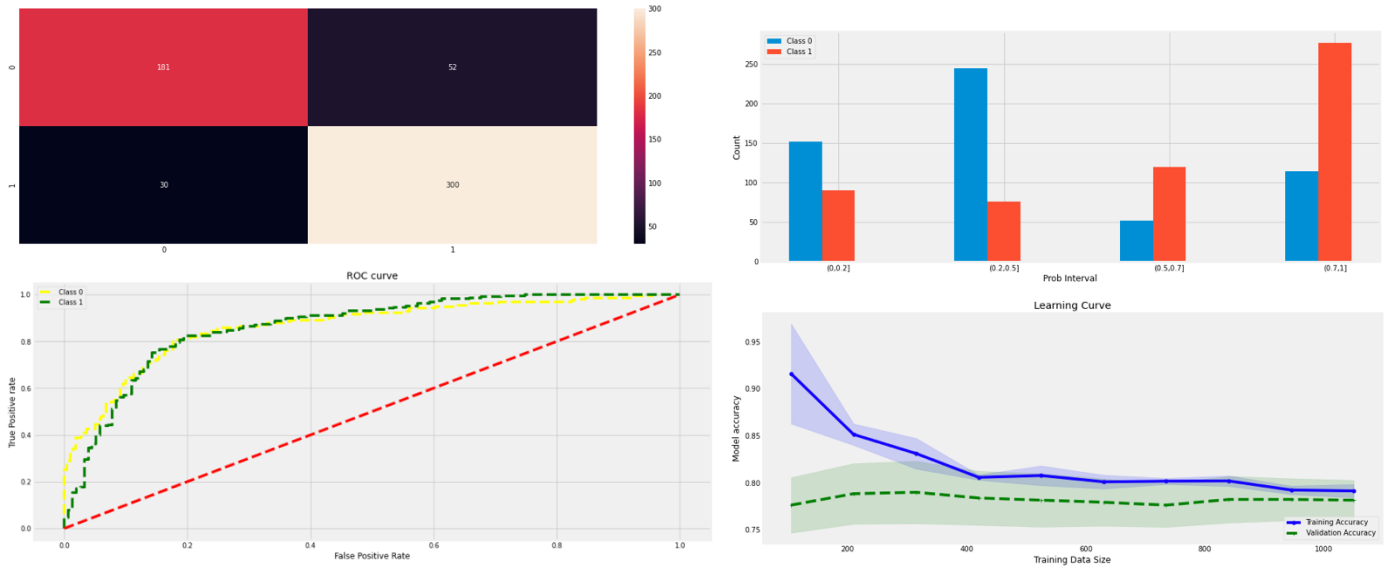


Figure 120:Voting Classifier model on Balanced Derived Data

Dataset: Balanced Binned Derived Data
Accuracy:0.8
Precision:0.79
Recall:0.91
F1-Score:0.84

	precision	recall	f1-score	support
0	0.83	0.65	0.73	155
1	0.79	0.91	0.84	221
accuracy			0.80	376
macro avg	0.81	0.78	0.79	376
weighted avg	0.81	0.80	0.80	376

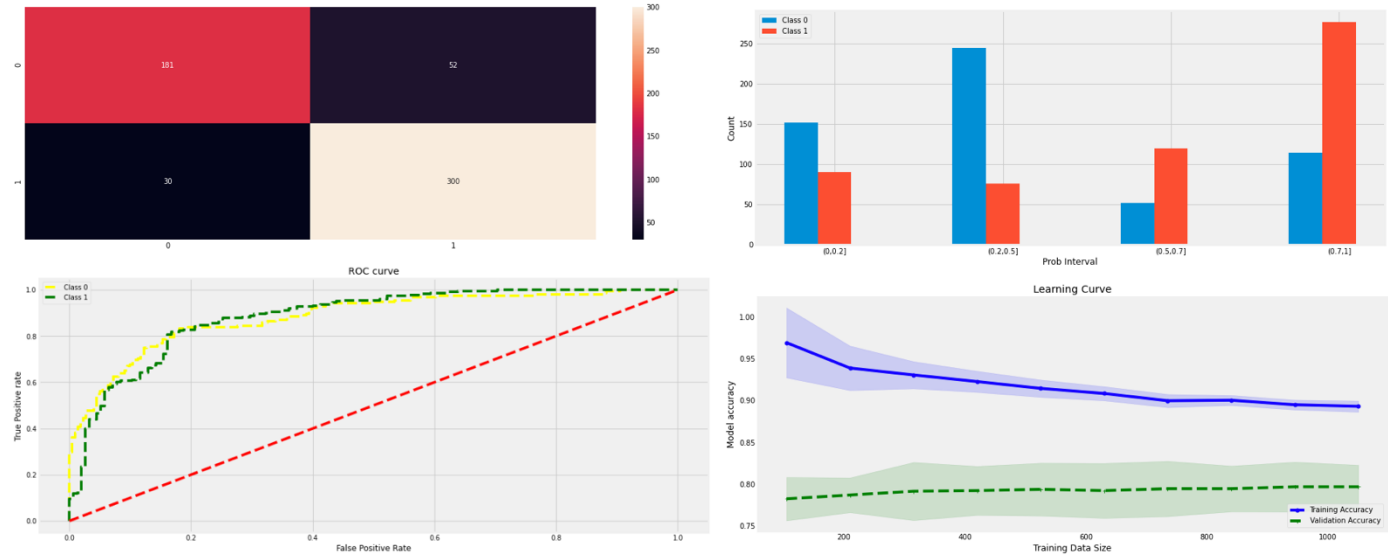


Figure 121:Voting Classifier model on Balanced Binned Derived Data

Dataset: Stdscaled Balanced Binned Derived Data
Accuracy:0.8
Precision:0.79
Recall:0.91
F1-Score:0.85

	precision	recall	f1-score	support
0	0.84	0.65	0.73	155
1	0.79	0.91	0.85	221
accuracy			0.80	376
macro avg	0.81	0.78	0.79	376
weighted avg	0.81	0.80	0.80	376

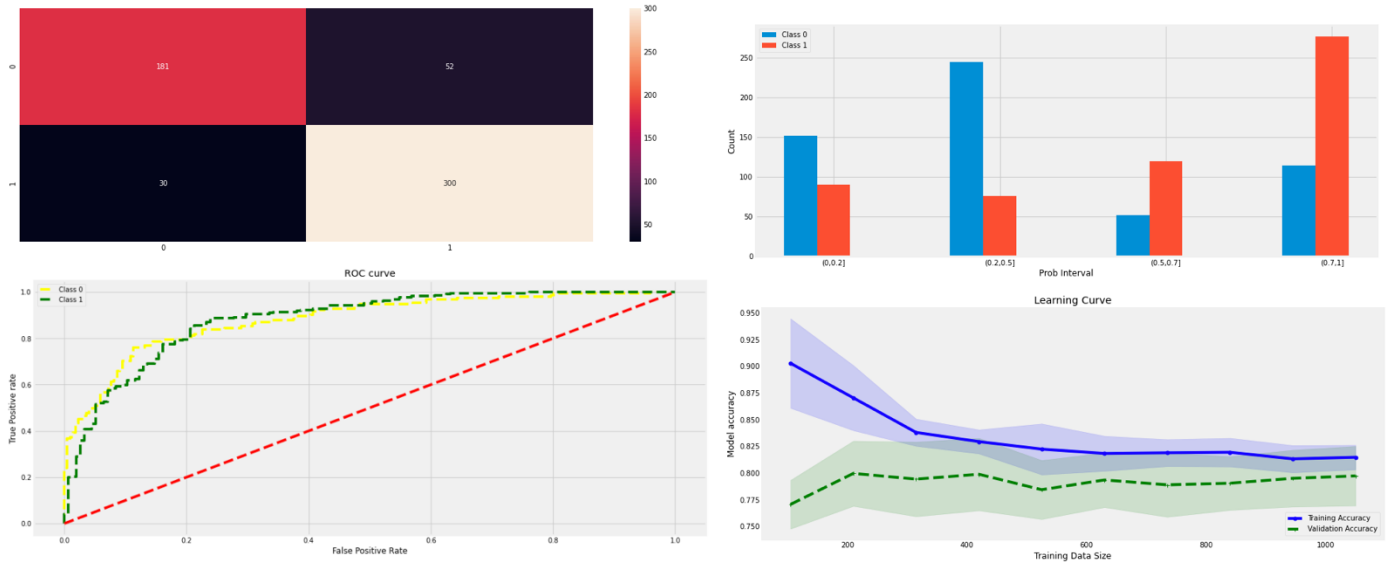


Figure 122:Voting Classifier model on Stdscaled Balanced Binned Derived Data

Dataset: Minmax scaled Balanced Binned Derived Data
Accuracy:0.81
Precision:0.79
Recall:0.92
F1-Score:0.85

	precision	recall	f1-score	support
0	0.85	0.65	0.73	155
1	0.79	0.92	0.85	221
accuracy			0.81	376
macro avg	0.82	0.78	0.79	376
weighted avg	0.81	0.81	0.80	376

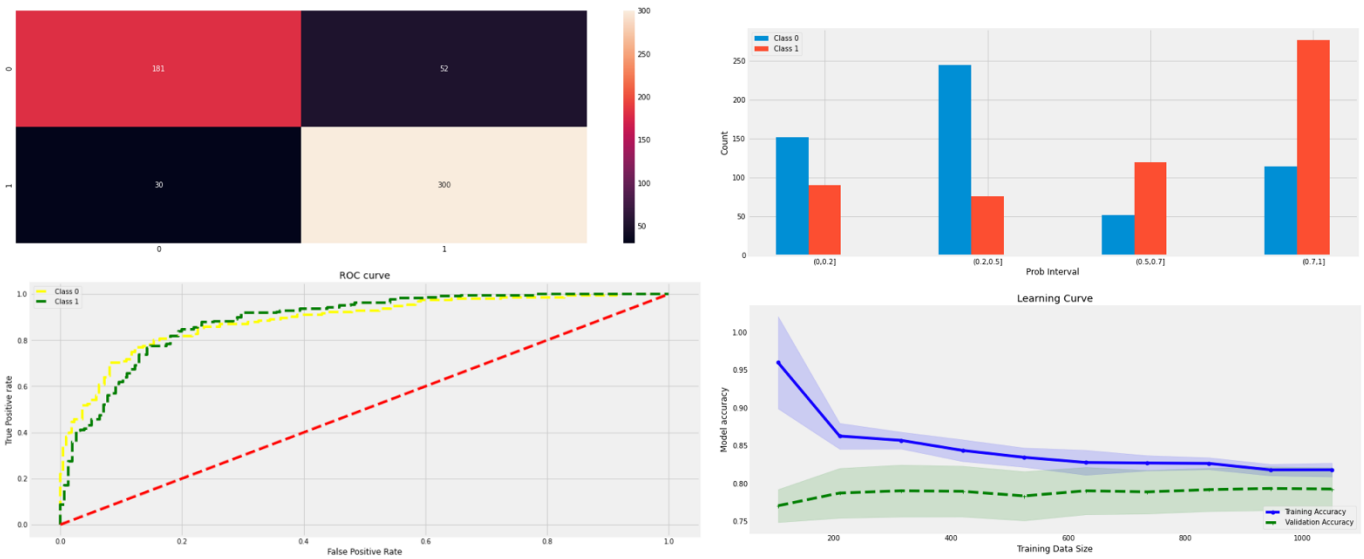


Figure 123:Voting Classifier model on Minmaxscaled Balanced Binned Derived Data

21. Modelling Summary

- The results from all the models (using the best parameters concluded from Randomized Search CV) built on various databases are summarized in the table below.

Model	Data	Test Accuracy	Training Accuracy	Precision	Recall	0/1 Precision	0/1 Recall	F1-Score	Trained Model
GradientBoostingClassifier	Balanced Binned Derived Data	0.88	0.93	0.88	0.93	[0.89, 0.88]	[0.81, 0.93]	0.90	GradientBoostingClassifier_Balanced Binned Der...
AdaBoostClassifier	Stdscaled Balanced Binned Derived Data	0.87	0.87	0.87	0.92	[0.88, 0.87]	[0.80, 0.92]	0.89	AdaBoostClassifier_Stdscaled Balanced Binned D...
GradientBoostingClassifier	Minmax scaled Balanced Binned Derived Data	0.87	0.90	0.87	0.92	[0.87, 0.87]	[0.80, 0.92]	0.89	GradientBoostingClassifier_Minmax scaled Balan...
GradientBoostingClassifier	Balanced Derived Data	0.86	0.86	0.86	0.91	[0.86, 0.86]	[0.78, 0.91]	0.88	GradientBoostingClassifier_Balanced Derived Da...
GradientBoostingClassifier	Stdscaled Balanced Binned Derived Data	0.86	0.86	0.85	0.92	[0.87, 0.85]	[0.77, 0.92]	0.88	GradientBoostingClassifier_Stdscaled Balanced ...
AdaBoostClassifier	Minmax scaled Balanced Binned Derived Data	0.86	0.87	0.85	0.92	[0.88, 0.85]	[0.77, 0.92]	0.89	AdaBoostClassifier_Minmax scaled Balanced Binn...
AdaBoostClassifier	Balanced Derived Data	0.85	0.86	0.85	0.90	[0.84, 0.85]	[0.77, 0.90]	0.87	AdaBoostClassifier_Balanced Derived Data_Acc_0...
AdaBoostClassifier	Balanced Binned Derived Data	0.85	0.87	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88	AdaBoostClassifier_Balanced Binned Derived Dat...
VotingClassifier	Balanced Binned Derived Data	0.81	0.81	0.79	0.92	[0.85, 0.79]	[0.65, 0.92]	0.85	VotingClassifier_Balanced Binned Derived Data_...
VotingClassifier	Stdscaled Balanced Binned Derived Data	0.81	0.81	0.79	0.92	[0.85, 0.79]	[0.65, 0.92]	0.85	VotingClassifier_Stdscaled Balanced Binned Der...
VotingClassifier	Minmax scaled Balanced Binned Derived Data	0.81	0.81	0.79	0.92	[0.85, 0.79]	[0.65, 0.92]	0.85	VotingClassifier_Minmax scaled Balanced Binned...
LogisticRegression	Balanced Binned Derived Data	0.80	0.78	0.78	0.93	[0.86, 0.78]	[0.62, 0.93]	0.85	LogisticRegression_Balanced Binned Derived Dat...
LogisticRegression	Stdscaled Balanced Binned Derived Data	0.80	0.78	0.78	0.93	[0.86, 0.78]	[0.62, 0.93]	0.85	LogisticRegression_Stdscaled Balanced Binned D...
LogisticRegression	Minmax scaled Balanced Binned Derived Data	0.80	0.78	0.78	0.92	[0.85, 0.78]	[0.62, 0.92]	0.84	LogisticRegression_Minmax scaled Balanced Binn...
RandomForestClassifier	Minmax scaled Balanced Binned Derived Data	0.80	0.81	0.81	0.87	[0.79, 0.81]	[0.71, 0.87]	0.84	RandomForestClassifier_Minmax scaled Balanced ...

Figure 124:Models with Test Accuracy greater than 0.8

Model	Data	Test Accuracy	Training Accuracy	Precision	Recall	0/1 Precision	0/1 Recall	F1-Score	Trained Model
VotingClassifier	Balanced Derived Data	0.79	0.83	0.77	0.93	[0.86, 0.77]	[0.59, 0.93]	0.84	VotingClassifier_Balanced Derived Data_Acc_0.7...
KNeighborsClassifier	Minmax scaled Balanced Binned Derived Data	0.77	0.79	0.78	0.85	[0.75, 0.78]	[0.65, 0.85]	0.81	KNeighborsClassifier_Minmax scaled Balanced Bi...
KNeighborsClassifier	Stdscaled Balanced Binned Derived Data	0.77	0.80	0.77	0.87	[0.77, 0.77]	[0.63, 0.87]	0.82	KNeighborsClassifier_Stdscaled Balanced Binned...
RandomForestClassifier	Balanced Binned Derived Data	0.77	0.76	0.77	0.87	[0.77, 0.77]	[0.62, 0.87]	0.82	RandomForestClassifier_Balanced Binned Derived...
LogisticRegression	Balanced Derived Data	0.76	0.77	0.75	0.90	[0.80, 0.75]	[0.56, 0.90]	0.82	LogisticRegression_Balanced Derived Data_Acc_0...
KNeighborsClassifier	Balanced Binned Derived Data	0.76	1.00	0.77	0.86	[0.75, 0.77]	[0.63, 0.86]	0.81	KNeighborsClassifier_Balanced Binned Derived D...
BernoulliNB	Minmax scaled Balanced Binned Derived Data	0.76	0.76	0.79	0.80	[0.71, 0.79]	[0.70, 0.80]	0.79	BernoulliNB_Minmax scaled Balanced Binned Deri...
BernoulliNB	Stdscaled Balanced Binned Derived Data	0.76	0.76	0.79	0.80	[0.71, 0.79]	[0.70, 0.80]	0.79	BernoulliNB_Stdscaled Balanced Binned Derived ...
BernoulliNB	Balanced Binned Derived Data	0.76	0.76	0.79	0.80	[0.71, 0.79]	[0.70, 0.80]	0.79	BernoulliNB_Balanced Binned Derived Data_Acc_0...
BernoulliNB	Balanced Derived Data	0.76	0.77	0.74	0.90	[0.80, 0.74]	[0.55, 0.90]	0.81	BernoulliNB_Balanced Derived Data_Acc_0.76_f1_...
GaussianNB	Balanced Binned Derived Data	0.74	0.72	0.72	0.93	[0.82, 0.72]	[0.48, 0.93]	0.81	GaussianNB_Balanced Binned Derived Data_Acc_0....
GaussianNB	Stdscaled Balanced Binned Derived Data	0.74	0.72	0.72	0.93	[0.82, 0.72]	[0.48, 0.93]	0.81	GaussianNB_Stdscaled Balanced Binned Derived D...
GaussianNB	Minmax scaled Balanced Binned Derived Data	0.74	0.72	0.72	0.93	[0.82, 0.72]	[0.48, 0.93]	0.81	GaussianNB_Minmax scaled Balanced Binned Deriv...
RandomForestClassifier	Balanced Derived Data	0.73	0.76	0.71	0.92	[0.81, 0.71]	[0.46, 0.92]	0.80	RandomForestClassifier_Balanced Derived Data_A...
DecisionTreeClassifier	Stdscaled Balanced Binned Derived Data	0.71	0.72	0.81	0.66	[0.62, 0.81]	[0.77, 0.66]	0.73	DecisionTreeClassifier_Stdscaled Balanced Binn...
GaussianNB	Balanced Derived Data	0.70	0.70	0.68	0.94	[0.81, 0.68]	[0.37, 0.94]	0.79	GaussianNB_Balanced Derived Data_Acc_0.7_f1_0.79
DecisionTreeClassifier	Balanced Binned Derived Data	0.68	0.69	0.76	0.67	[0.60, 0.76]	[0.70, 0.67]	0.71	DecisionTreeClassifier_Balanced Binned Derived...
KNeighborsClassifier	Balanced Derived Data	0.68	0.71	0.67	0.87	[0.68, 0.67]	[0.40, 0.87]	0.76	KNeighborsClassifier_Balanced Derived Data_Acc...
DecisionTreeClassifier	Balanced Derived Data	0.59	0.59	0.59	1.00	[0.00, 0.59]	[0.00, 1.00]	0.74	DecisionTreeClassifier_Balanced Derived Data_A...
RandomForestClassifier	Stdscaled Balanced Binned Derived Data	0.59	0.59	0.59	1.00	[0.00, 0.59]	[0.00, 1.00]	0.74	RandomForestClassifier_Stdscaled Balanced Binn...
DecisionTreeClassifier	Minmax scaled Balanced Binned Derived Data	0.59	0.59	0.59	1.00	[0.00, 0.59]	[0.00, 1.00]	0.74	DecisionTreeClassifier_Minmax scaled Balanced ...

Figure 125:Models with Test Accuracy lesser than 0.8

- The results from all the models (with RFECV) built on various databases are summarized in the table below.

Model	Data	Features	Test Accuracy	Training Accuracy	Precision	Recall	0/1 Precision	0/1 Recall	F1-Score
GradientBoostingClassifier	Balanced Derived Data	[Age, Gender, City, EducationLevel, Salary, Jo...	0.87	0.89	0.86	0.92	[0.88, 0.86]	[0.79, 0.92]	0.89
GradientBoostingClassifier	Balanced Binned Derived Data	[City, JoiningDesignation, TotalBusinessValue,...	0.87	0.90	0.87	0.92	[0.87, 0.87]	[0.80, 0.92]	0.89
GradientBoostingClassifier	Stdscaled Balanced Binned Derived Data	[City, JoiningDesignation, TotalBusinessValue,...	0.87	0.90	0.87	0.92	[0.87, 0.87]	[0.81, 0.92]	0.89
GradientBoostingClassifier	Minmax scaled Balanced Binned Derived Data	[City, JoiningDesignation, TotalBusinessValue,...	0.87	0.90	0.87	0.92	[0.87, 0.87]	[0.80, 0.92]	0.89
AdaBoostClassifier	Balanced Binned Derived Data	[City, Designation, TotalBusinessValue, Quarte...	0.86	0.86	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88
AdaBoostClassifier	Stdscaled Balanced Binned Derived Data	[City, Designation, TotalBusinessValue, Quarte...	0.86	0.86	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88
AdaBoostClassifier	Minmax scaled Balanced Binned Derived Data	[City, Designation, TotalBusinessValue, Quarte...	0.86	0.86	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88
RandomForestClassifier	Balanced Derived Data	[Age, City, Salary, TotalBusinessValue, Quarte...	0.85	0.98	0.86	0.90	[0.85, 0.86]	[0.79, 0.90]	0.88
AdaBoostClassifier	Balanced Derived Data	[Age, City, EducationLevel, Salary, JoiningDes...	0.85	0.86	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88
RandomForestClassifier	Balanced Binned Derived Data	[Gender, City, EducationLevel, JoiningDesignat...	0.84	0.98	0.85	0.88	[0.82, 0.85]	[0.77, 0.88]	0.86
RandomForestClassifier	Minmax scaled Balanced Binned Derived Data	[Gender, City, EducationLevel, JoiningDesignat...	0.84	0.98	0.86	0.86	[0.80, 0.86]	[0.80, 0.86]	0.86
RandomForestClassifier	Stdscaled Balanced Binned Derived Data	[City, EducationLevel, Designation, TotalBusin...	0.82	0.98	0.83	0.86	[0.79, 0.83]	[0.75, 0.86]	0.85
DecisionTreeClassifier	Balanced Derived Data	[Age, City, EducationLevel, Salary, JoiningDes...	0.80	0.96	0.83	0.82	[0.75, 0.83]	[0.76, 0.82]	0.82
DecisionTreeClassifier	Stdscaled Balanced Binned Derived Data	[City, TotalBusinessValue, QuarterlyRating, To...	0.80	0.96	0.81	0.86	[0.78, 0.81]	[0.72, 0.86]	0.83
DecisionTreeClassifier	Minmax scaled Balanced Binned Derived Data	[Gender, City, EducationLevel, JoiningDesignat...	0.80	0.97	0.83	0.81	[0.74, 0.83]	[0.77, 0.81]	0.82

Figure 126: Models with RFECV with Test Accuracy >= 0.8

Model	Data	Features	Test Accuracy	Training Accuracy	Precision	Recall	0/1 Precision	0/1 Recall	F1-Score
LogisticRegression	Balanced Binned Derived Data	[Designation, TotalBusinessValue, QuarterlyRat...	0.79	0.78	0.77	0.91	[0.83, 0.77]	[0.62, 0.91]	0.84
DecisionTreeClassifier	Balanced Binned Derived Data	[Gender, City, EducationLevel, JoiningDesignat...	0.79	0.97	0.83	0.81	[0.74, 0.83]	[0.76, 0.81]	0.82
LogisticRegression	Stdscaled Balanced Binned Derived Data	[Designation, TotalBusinessValue, QuarterlyRat...	0.79	0.78	0.77	0.91	[0.83, 0.77]	[0.62, 0.91]	0.84
LogisticRegression	Minmax scaled Balanced Binned Derived Data	[Designation, TotalBusinessValue, QuarterlyRat...	0.79	0.78	0.77	0.91	[0.83, 0.77]	[0.62, 0.91]	0.84
BernoulliNB	Balanced Binned Derived Data	[Gender, City, EducationLevel, JoiningDesignat...	0.77	0.76	0.78	0.84	[0.74, 0.78]	[0.67, 0.84]	0.81
BernoulliNB	Stdscaled Balanced Binned Derived Data	[Gender, City, EducationLevel, JoiningDesignat...	0.77	0.76	0.78	0.84	[0.74, 0.78]	[0.67, 0.84]	0.81
BernoulliNB	Minmax scaled Balanced Binned Derived Data	[Gender, City, EducationLevel, JoiningDesignat...	0.77	0.76	0.78	0.84	[0.74, 0.78]	[0.67, 0.84]	0.81
BernoulliNB	Balanced Derived Data	[Gender, EducationLevel, TotalBusinessValue, S...	0.76	0.77	0.74	0.90	[0.80, 0.74]	[0.55, 0.90]	0.81
LogisticRegression	Balanced Derived Data	[Age, Salary, Designation, TotalBusinessValue]	0.70	0.68	0.69	0.88	[0.72, 0.69]	[0.44, 0.88]	0.77

Figure 127: Models with RFECV with Test Accuracy < 0.8

- The results from all the models (using the best parameters concluded from Randomized Search CV and with RFECV) built on various databases combined are summarized in the table below.

Model	Data	Approach	Features	Test Accuracy	Training Accuracy	Precision	Recall	0/1 Precision	0/1 Recall	F1-Score
GradientBoostingClassifier	Balanced Binned Derived Data	RSCV	NA	0.88	0.93	0.88	0.93	[0.89, 0.88]	[0.81, 0.93]	0.90
AdaBoostClassifier	Stdscaled Balanced Binned Derived Data	RSCV	NA	0.87	0.87	0.87	0.92	[0.88, 0.87]	[0.80, 0.92]	0.89
GradientBoostingClassifier	Stdscaled Balanced Binned Derived Data	RFECV	[City, JoiningDesignation, TotalBusinessValue,...	0.87	0.90	0.87	0.92	[0.87, 0.87]	[0.81, 0.92]	0.89
GradientBoostingClassifier	Balanced Derived Data	RFECV	[Age, Gender, City, EducationLevel, Salary, Jo...	0.87	0.89	0.86	0.92	[0.88, 0.86]	[0.79, 0.92]	0.89
GradientBoostingClassifier	Balanced Binned Derived Data	RFECV	[City, JoiningDesignation, TotalBusinessValue,...	0.87	0.90	0.87	0.92	[0.87, 0.87]	[0.80, 0.92]	0.89
GradientBoostingClassifier	Minmax scaled Balanced Binned Derived Data	RFECV	[City, JoiningDesignation, TotalBusinessValue,...	0.87	0.90	0.87	0.92	[0.87, 0.87]	[0.80, 0.92]	0.89
GradientBoostingClassifier	Minmax scaled Balanced Binned Derived Data	RSCV	NA	0.87	0.90	0.87	0.92	[0.87, 0.87]	[0.80, 0.92]	0.89
AdaBoostClassifier	Minmax scaled Balanced Binned Derived Data	RFECV	[City, Designation, TotalBusinessValue, Quarte...	0.86	0.86	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88
GradientBoostingClassifier	Stdscaled Balanced Binned Derived Data	RSCV	NA	0.86	0.86	0.85	0.92	[0.87, 0.85]	[0.77, 0.92]	0.88
GradientBoostingClassifier	Balanced Derived Data	RSCV	NA	0.86	0.86	0.86	0.91	[0.86, 0.86]	[0.78, 0.91]	0.88
AdaBoostClassifier	Minmax scaled Balanced Binned Derived Data	RSCV	NA	0.86	0.87	0.85	0.92	[0.88, 0.85]	[0.77, 0.92]	0.89
AdaBoostClassifier	Balanced Binned Derived Data	RFECV	[City, Designation, TotalBusinessValue, Quarte...	0.86	0.86	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88
AdaBoostClassifier	Stdscaled Balanced Binned Derived Data	RFECV	[City, Designation, TotalBusinessValue, Quarte...	0.86	0.86	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88
RandomForestClassifier	Balanced Derived Data	RFECV	[Age, City, Salary, TotalBusinessValue, Quarte...	0.85	0.98	0.86	0.90	[0.85, 0.86]	[0.79, 0.90]	0.88
AdaBoostClassifier	Balanced Derived Data	RSCV	NA	0.85	0.86	0.85	0.90	[0.84, 0.85]	[0.77, 0.90]	0.87
AdaBoostClassifier	Balanced Derived Data	RFECV	[Age, City, EducationLevel, Salary, JoiningDes...	0.85	0.86	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88
AdaBoostClassifier	Balanced Binned Derived Data	RSCV	NA	0.85	0.87	0.85	0.91	[0.86, 0.85]	[0.77, 0.91]	0.88
RandomForestClassifier	Minmax scaled Balanced Binned Derived Data	RFECV	[Gender, City, EducationLevel, JoiningDesignat...	0.84	0.98	0.86	0.86	[0.80, 0.86]	[0.80, 0.86]	0.86
RandomForestClassifier	Balanced Binned Derived Data	RFECV	[Gender, City, EducationLevel, JoiningDesignat...	0.84	0.98	0.85	0.88	[0.82, 0.85]	[0.77, 0.88]	0.86
RandomForestClassifier	Stdscaled Balanced Binned Derived Data	RFECV	[City, EducationLevel, Designation, TotalBusin...	0.82	0.98	0.83	0.86	[0.79, 0.83]	[0.75, 0.86]	0.85
VotingClassifier	Stdscaled Balanced Binned Derived Data	RSCV	NA	0.81	0.81	0.79	0.92	[0.85, 0.79]	[0.65, 0.92]	0.85
VotingClassifier	Minmax scaled Balanced Binned Derived Data	RSCV	NA	0.81	0.81	0.79	0.92	[0.85, 0.79]	[0.65, 0.92]	0.85
VotingClassifier	Balanced Binned Derived Data	RSCV	NA	0.81	0.81	0.79	0.92	[0.85, 0.79]	[0.65, 0.92]	0.85
LogisticRegression	Stdscaled Balanced Binned Derived Data	RSCV	NA	0.80	0.78	0.78	0.93	[0.86, 0.78]	[0.62, 0.93]	0.85
LogisticRegression	Balanced Binned Derived Data	RSCV	NA	0.80	0.78	0.78	0.93	[0.86, 0.78]	[0.62, 0.93]	0.85
LogisticRegression	Minmax scaled Balanced Binned Derived Data	RSCV	NA	0.80	0.78	0.78	0.92	[0.85, 0.78]	[0.62, 0.92]	0.84
RandomForestClassifier	Minmax scaled Balanced Binned Derived Data	RSCV	NA	0.80	0.81	0.81	0.87	[0.79, 0.81]	[0.71, 0.87]	0.84
DecisionTreeClassifier	Minmax scaled Balanced Binned Derived Data	RFECV	[Gender, City, EducationLevel, JoiningDesignat...	0.80	0.97	0.83	0.81	[0.74, 0.83]	[0.77, 0.81]	0.82
DecisionTreeClassifier	Stdscaled Balanced Binned Derived Data	RFECV	[City, TotalBusinessValue, QuarterlyRating, To...	0.80	0.96	0.81	0.86	[0.78, 0.81]	[0.72, 0.86]	0.83
DecisionTreeClassifier	Balanced Derived Data	RFECV	[Age, City, EducationLevel, Salary, JoiningDes...	0.80	0.96	0.83	0.82	[0.75, 0.83]	[0.76, 0.82]	0.82

Figure 128: Models with Accuracy >=0.8

Model	Data	Approach	Features	Test Accuracy	Training Accuracy	Precision	Recall	0/1 Precision	0/1 Recall	F1-Score
VotingClassifier	Balanced Derived Data	RSCV	NA	0.79	0.83	0.77	0.93	[0.86, 0.77]	[0.59, 0.93]	0.84
LogisticRegression	Balanced Binned Derived Data	RFECV	[Designation, TotalBusinessValue, QuarterlyRat...	0.79	0.78	0.77	0.91	[0.83, 0.77]	[0.62, 0.91]	0.84
DecisionTreeClassifier	Balanced Binned Derived Data	RFECV	[Gender, City, EducationLevel, JoiningDesignat...	0.79	0.97	0.83	0.81	[0.74, 0.83]	[0.76, 0.81]	0.82
LogisticRegression	Stdscaled Balanced Binned Derived Data	RFECV	[Designation, TotalBusinessValue, QuarterlyRat...	0.79	0.78	0.77	0.91	[0.83, 0.77]	[0.62, 0.91]	0.84
LogisticRegression	Minmax scaled Balanced Binned Derived Data	RFECV	[Designation, TotalBusinessValue, QuarterlyRat...	0.79	0.78	0.77	0.91	[0.83, 0.77]	[0.62, 0.91]	0.84
BernoulliNB	Balanced Binned Derived Data	RFECV	[Gender, City, EducationLevel, JoiningDesignat...	0.77	0.76	0.78	0.84	[0.74, 0.78]	[0.67, 0.84]	0.81
BernoulliNB	Stdscaled Balanced Binned Derived Data	RFECV	[Gender, City, EducationLevel, JoiningDesignat...	0.77	0.76	0.78	0.84	[0.74, 0.78]	[0.67, 0.84]	0.81
BernoulliNB	Minmax scaled Balanced Binned Derived Data	RFECV	[Gender, City, EducationLevel, JoiningDesignat...	0.77	0.76	0.78	0.84	[0.74, 0.78]	[0.67, 0.84]	0.81
RandomForestClassifier	Balanced Binned Derived Data	RSCV	NA	0.77	0.76	0.77	0.87	[0.77, 0.77]	[0.62, 0.87]	0.82
KNeighborsClassifier	Minmax scaled Balanced Binned Derived Data	RSCV	NA	0.77	0.79	0.78	0.85	[0.75, 0.78]	[0.65, 0.85]	0.81
KNeighborsClassifier	Stdscaled Balanced Binned Derived Data	RSCV	NA	0.77	0.80	0.77	0.87	[0.77, 0.77]	[0.63, 0.87]	0.82
BernoulliNB	Stdscaled Balanced Binned Derived Data	RSCV	NA	0.76	0.76	0.79	0.80	[0.71, 0.79]	[0.70, 0.80]	0.79
KNeighborsClassifier	Balanced Binned Derived Data	RSCV	NA	0.76	1.00	0.77	0.86	[0.75, 0.77]	[0.63, 0.86]	0.81
BernoulliNB	Balanced Derived Data	RFECV	[Gender, EducationLevel, TotalBusinessValue, S...	0.76	0.77	0.74	0.90	[0.80, 0.74]	[0.55, 0.90]	0.81
BernoulliNB	Minmax scaled Balanced Binned Derived Data	RSCV	NA	0.76	0.76	0.79	0.80	[0.71, 0.79]	[0.70, 0.80]	0.79
BernoulliNB	Balanced Derived Data	RSCV	NA	0.76	0.77	0.74	0.90	[0.80, 0.74]	[0.55, 0.90]	0.81
LogisticRegression	Balanced Derived Data	RSCV	NA	0.76	0.77	0.75	0.90	[0.80, 0.75]	[0.56, 0.90]	0.82
BernoulliNB	Balanced Binned Derived Data	RSCV	NA	0.76	0.76	0.79	0.80	[0.71, 0.79]	[0.70, 0.80]	0.79
GaussianNB	Stdscaled Balanced Binned Derived Data	RSCV	NA	0.74	0.72	0.72	0.93	[0.82, 0.72]	[0.48, 0.93]	0.81
GaussianNB	Balanced Binned Derived Data	RSCV	NA	0.74	0.72	0.72	0.93	[0.82, 0.72]	[0.48, 0.93]	0.81
GaussianNB	Minmax scaled Balanced Binned Derived Data	RSCV	NA	0.74	0.72	0.72	0.93	[0.82, 0.72]	[0.48, 0.93]	0.81
RandomForestClassifier	Balanced Derived Data	RSCV	NA	0.73	0.76	0.71	0.92	[0.81, 0.71]	[0.46, 0.92]	0.80
DecisionTreeClassifier	Stdscaled Balanced Binned Derived Data	RSCV	NA	0.71	0.72	0.81	0.66	[0.62, 0.81]	[0.77, 0.66]	0.73
GaussianNB	Balanced Derived Data	RSCV	NA	0.70	0.70	0.68	0.94	[0.81, 0.68]	[0.37, 0.94]	0.79
LogisticRegression	Balanced Derived Data	RFECV	[Age, Salary, Designation, TotalBusinessValue]	0.70	0.68	0.69	0.88	[0.72, 0.69]	[0.44, 0.88]	0.77
DecisionTreeClassifier	Balanced Binned Derived Data	RSCV	NA	0.68	0.69	0.76	0.67	[0.60, 0.76]	[0.70, 0.67]	0.71
KNeighborsClassifier	Balanced Derived Data	RSCV	NA	0.68	0.71	0.67	0.87	[0.68, 0.67]	[0.40, 0.87]	0.76
RandomForestClassifier	Stdscaled Balanced Binned Derived Data	RSCV	NA	0.59	0.59	0.59	1.00	[0.00, 0.59]	[0.00, 1.00]	0.74
DecisionTreeClassifier	Minmax scaled Balanced Binned Derived Data	RSCV	NA	0.59	0.59	0.59	1.00	[0.00, 0.59]	[0.00, 1.00]	0.74
DecisionTreeClassifier	Balanced Derived Data	RSCV	NA	0.59	0.59	0.59	1.00	[0.00, 0.59]	[0.00, 1.00]	0.74

Figure 129: Models with Accuracy <0.8

22. Conclusion / Recommendations

A total of 60 models have been built using 9 different algorithm and trained on 4 different datasets. Out of which 36 models were built using the best parameter concluded from the Randomized Search CV applied on the 9 algorithms and 24 models were built using the RFECV technique on the models.

The **maximum test accuracy** achieved among all the models **is 0.88** with the **train accuracy of 0.93, precision of {0: 0.89, 1: 0.88} and recall of {0: 0.81, 1:0.93}.** These results are achieved by **Gradient Boosting Classifier** with the best parameters concluded from **Randomized Search CV** implementation on **Balanced Binned Derived dataset.**

Out of the top 10 models in terms of test accuracy, **7 are Gradient Boosting Classifier** models trained on different datasets using different approach (RSCV or RFECV). It has been observed that the results from 2 different datasets created from scaling by 2 techniques StdScaling and Minmax Scaling are quite similar with slight variation.

After Gradient Boosting Classifier**, AdaBoost** with best parameters concluded from RSCV when trained on Stdscaled Balanced Binned Data is giving the best results **– test accuracy: 0.87, train accuracy: 0.87, precision: {0: 0.88, 1: 0.87} and recall {0: 0.80, 1: 0.92}.**

Another Classifier named **Random Forest** when implemented with **RFECV** and trained on **Balanced Derived Data** reaches the bracket of **test accuracy >=0.85.** It resulted in **test accuracy: 0.85, train accuracy: 0.98, precision: {0: 0.85, 1: 0.86} and recall {0: 0.79, 1: 0.90}.**

Thus, the top 3 algorithm implementations concluded are –

- Gradient Boosting Classifier
- AdaBoost Classifier
- Random Forest Classifier

The above mentioned top 3 models are recommended for Employee Attrition Prediction for the given feature set and data.

Since the given data spans over 2 years of reporting only. It is recommended to acquire data that covers records from larger duration of reporting for model building to achieve best results.

23. Future Work & Extension or Scope of improvements

- Several other models like Support Vector Machine, Neural Network models, Stacking Classifier, etc. could be used for Modelling. Comparing new models could result in better results.
- Other feature selection techniques could be explored to achieve feature subsets.
- More datasets could be created using the feature subsets extracted from feature selection techniques. This could help reach better results.
- As mentioned earlier, the given dataset spans over 2 yrs of reporting. Acquiring the data over larger duration could result in further improvement.

24. Deployment

The project has been deployed using Streamlit.
Three sections have been created namely –

- Batch Prediction – For predicting attrition in a set of records
- Single Prediction – For predicting if an employee will stay or resign for a single record
- Train Prediction – For training the model in case of new data arrival

Following are the screenshots from the Deployment

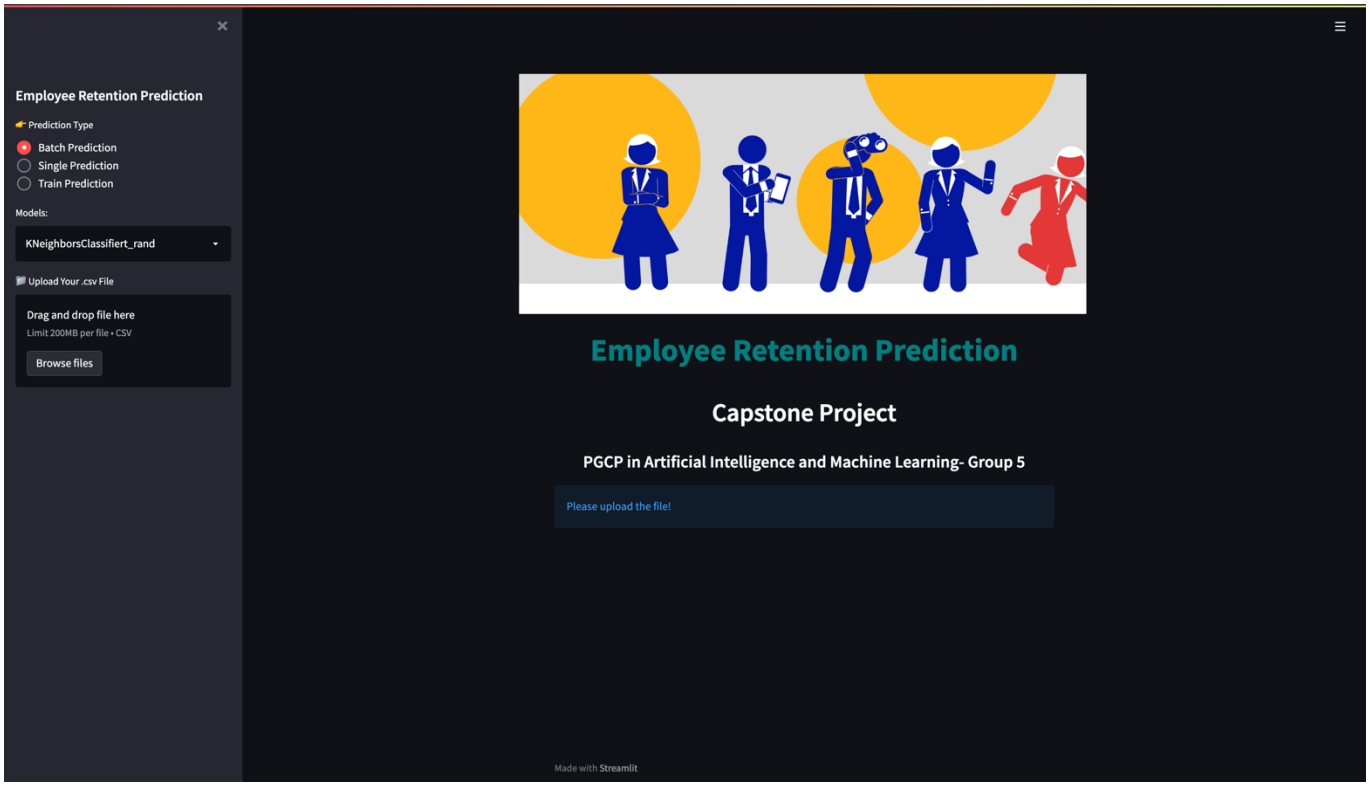


Figure 130: Home Page for the application

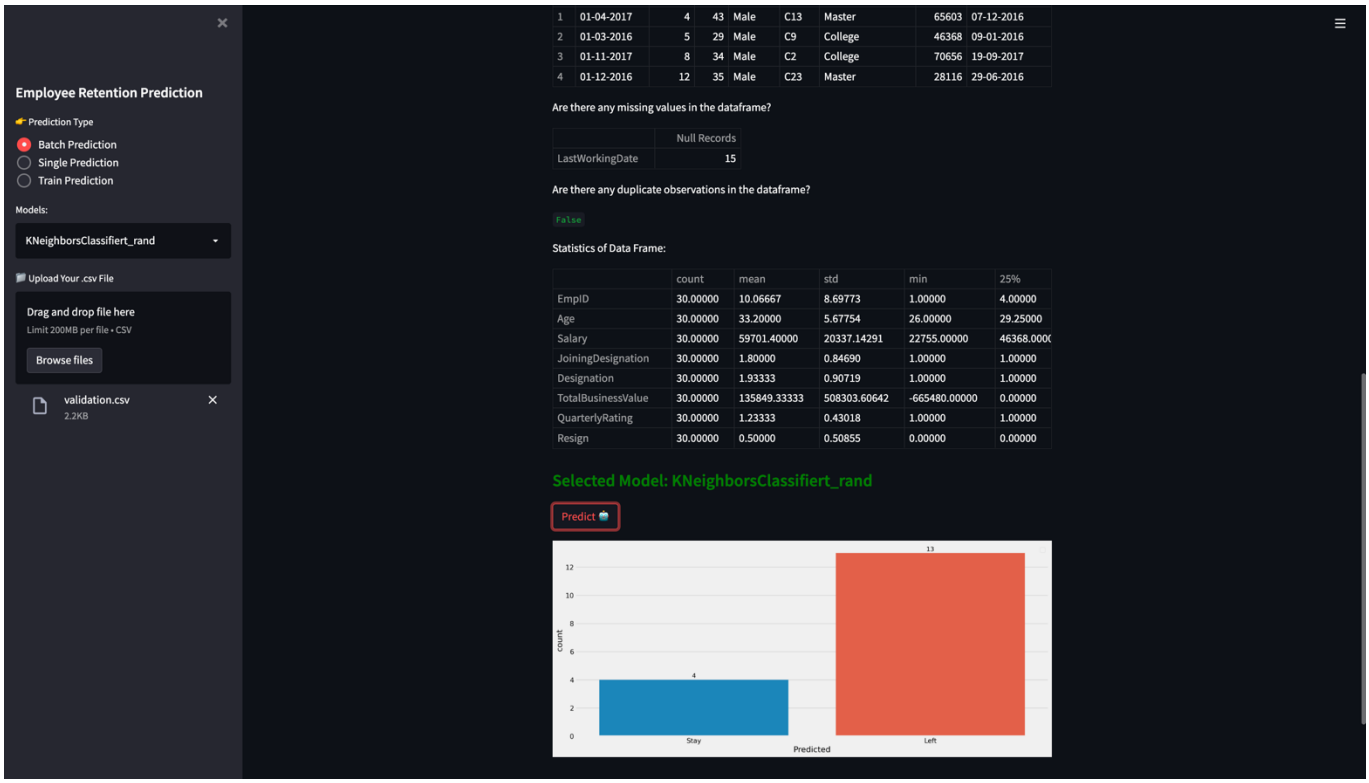


Figure 131: Attrition ratio predicted for a batch of records

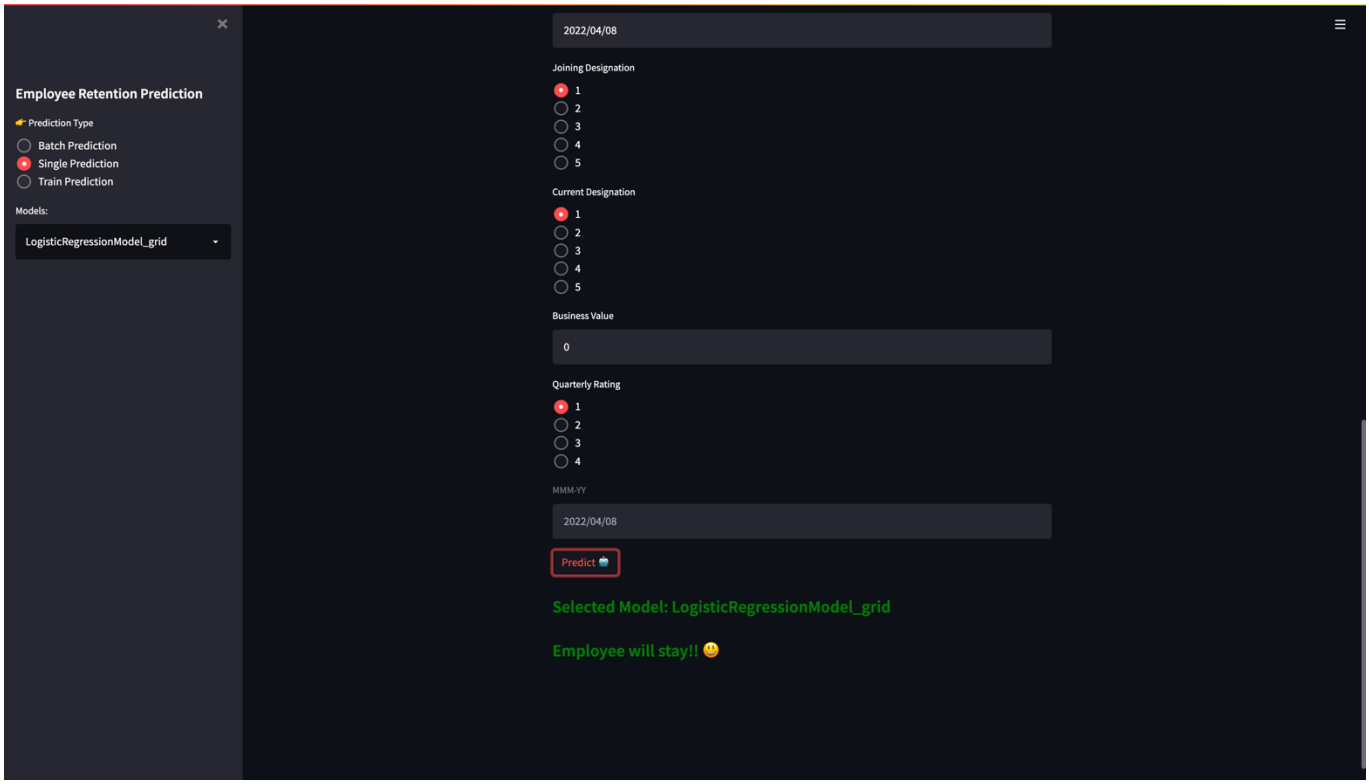


Figure 132: Prediction for a particular employee

25. References / Bibliography

Predictive Attrition Model: Using Analytics to predict Employee Attrition. (n.d.). Retrieved from https://analyticsindiamag.com/predictive-attrition-model/

Cotton, J. L., & Tuttle, J. M. (n.d.). Employee turnover: A metaanalysis and review with implications for research.

Liu, D., Mitchell, T. R., Lee, T. W., Holton, B. C., & Hinkin, T. R. (n.d.).

Scikit Learn sklearn.ensemble.AdaBoostClassifier. (n.d.). Retrieved from Scikit Learn : https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

(n.d.).

(n.d.). Retrieved from Scikit Learn sklearn.feature_selection.SelectKBest: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

(n.d.). Retrieved from Scikit Learn sklearn.feature_selection.RFE: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

(n.d.). Retrieved from Scikit Learn sklearn.feature_selection.RFECV: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html

(n.d.). Retrieved from Machine Learning Mastery Recursive Feature Elimination (RFE) for Feature Selection in Python: https://machinelearningmastery.com/rfe-feature-selection-in-python/

(n.d.). Retrieved from Data Analytics Learning Curves Explained with Python Sklearn Example: https://vitalflux.com/learning-curves-explained-python-sklearn-example/

(n.d.). Retrieved from Scikit Learn sklearn.linear_model.LogisticRegression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

(n.d.). Retrieved from Scikit Learn sklearn.tree.DecisionTreeClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

(n.d.). Retrieved from Scikit Learn sklearn.neighbors.KNeighborsClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

(n.d.). Retrieved from Scikit Learn sklearn.ensemble.RandomForestClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

(n.d.). Retrieved from Scikit Learn sklearn.naive_bayes.BernoulliNB: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html

(n.d.). Retrieved from Scikit Learn sklearn.naive_bayes.GaussianNB: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

(n.d.). Retrieved from Scikit Learn sklearn.ensemble.GradientBoostingClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

(n.d.). Retrieved from Scikit Learn sklearn.ensemble.VotingClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

Appendix A: Principal Component Analysis

This appendix consists of the description of PCA Analysis and reason for scaling the data before implementing PCA.

Principal Component Analysis is a dimensionality reduction method that is usually used to reduce the dimensions of a large dataset. It transforms the dataset into n components as specified in user input. These n components are each combination of original features. The contribution of each feature to these components can be extracted. Using n=2 makes it easier to visualize the dataset as it can be plotted in graphs.

The data needs to be scaled before implementing the Principal Component Analysis. If not the contribution of small-valued features will be suppressed by the contribution of large-valued features. This could also be seen in the jupyter snippet below

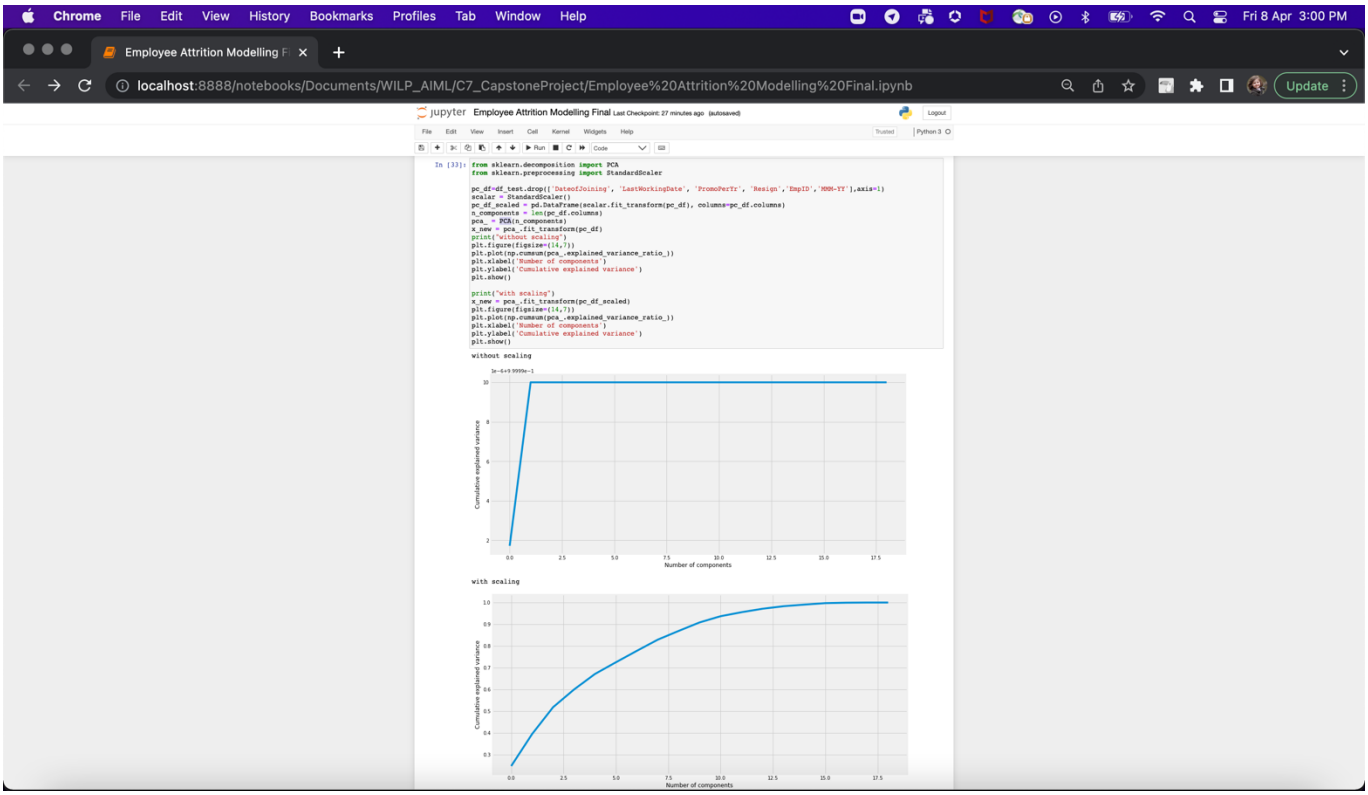


Figure 133:Jupyter snippet comparing PCA with and w/o scaling

For the given dataset, the data needs to be transformed into 10 components to preserve 95% of variance in the original data.

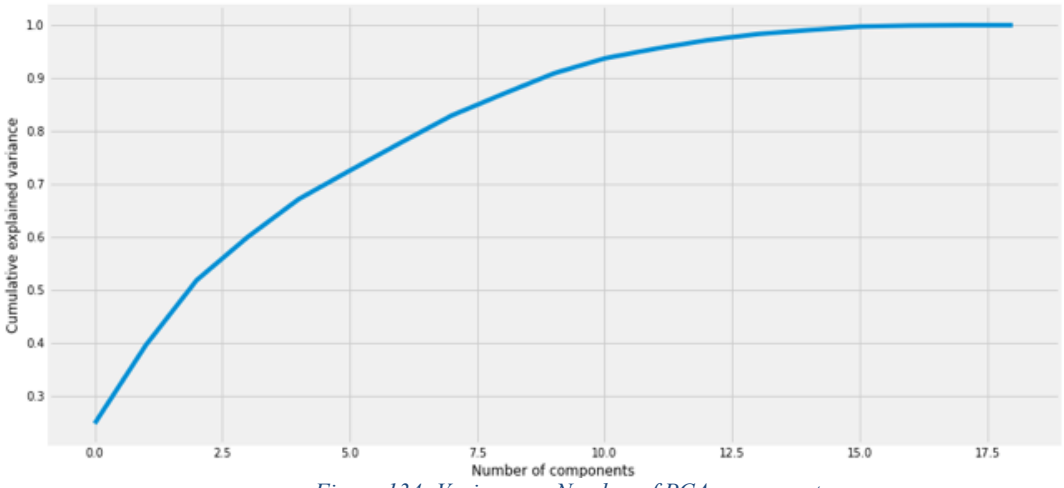


Figure 134: Variance vs Number of PCA components

The original features cannot be extracted by from the principal components. Therefore, there is no way to achieve the most relevant features after PCA. Thus, PCA data is not used for modelling in this project.

Appendix B: Feature Selection

This appendix consists of the description of code snippets in Feature Selection section.

There are various methods available for shortlisting the most relevant features for modelling. In this project few methods like SelectKBest, RFE and RFECV are implemented.

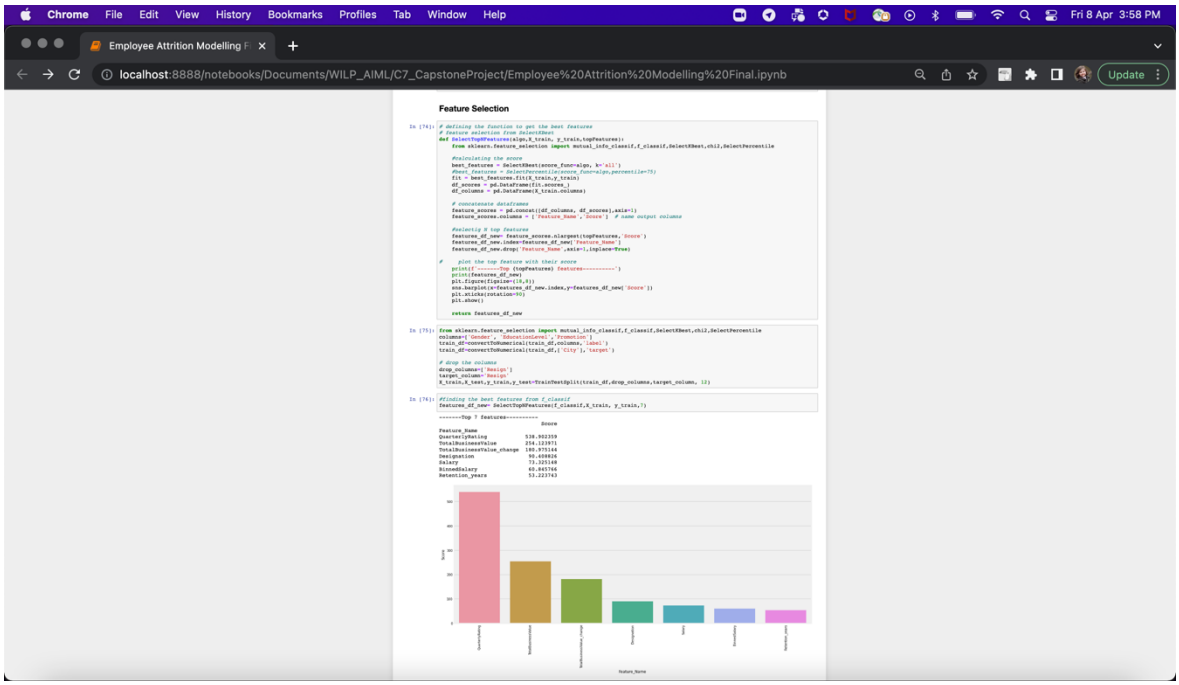


Figure 135:Code snippet: Top7 feature from SelectKBest using f_classif

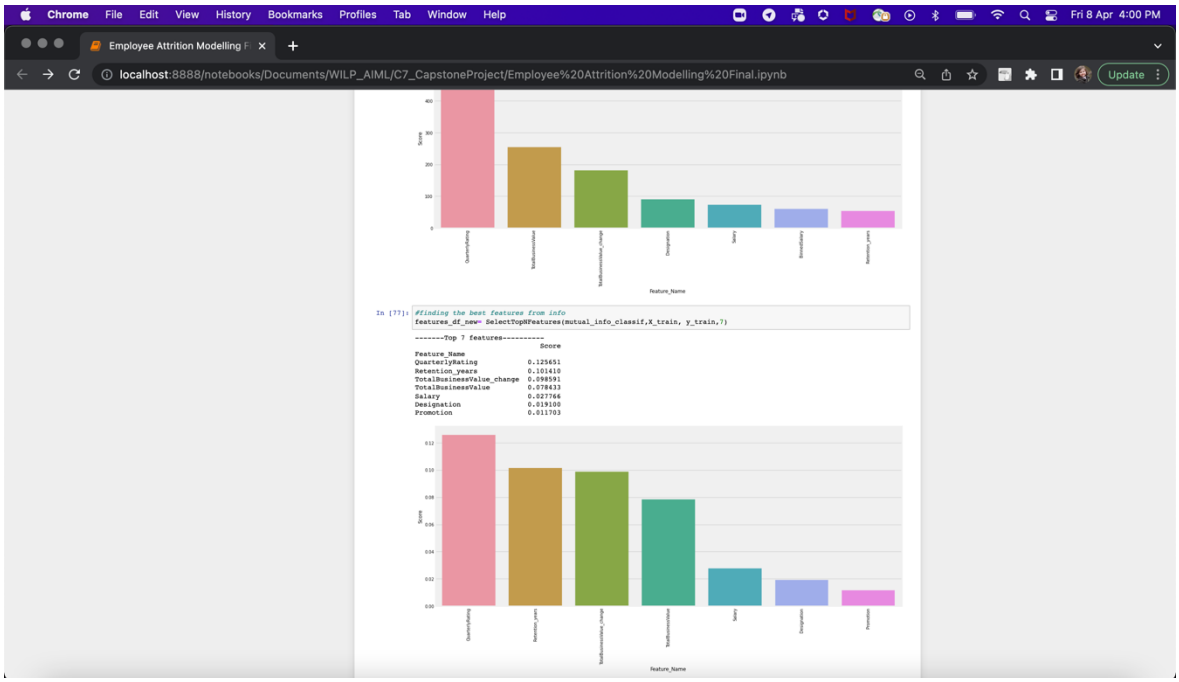


Figure 136: Code snippet: Top7 feature from SelectKBest using mutual_info_classif

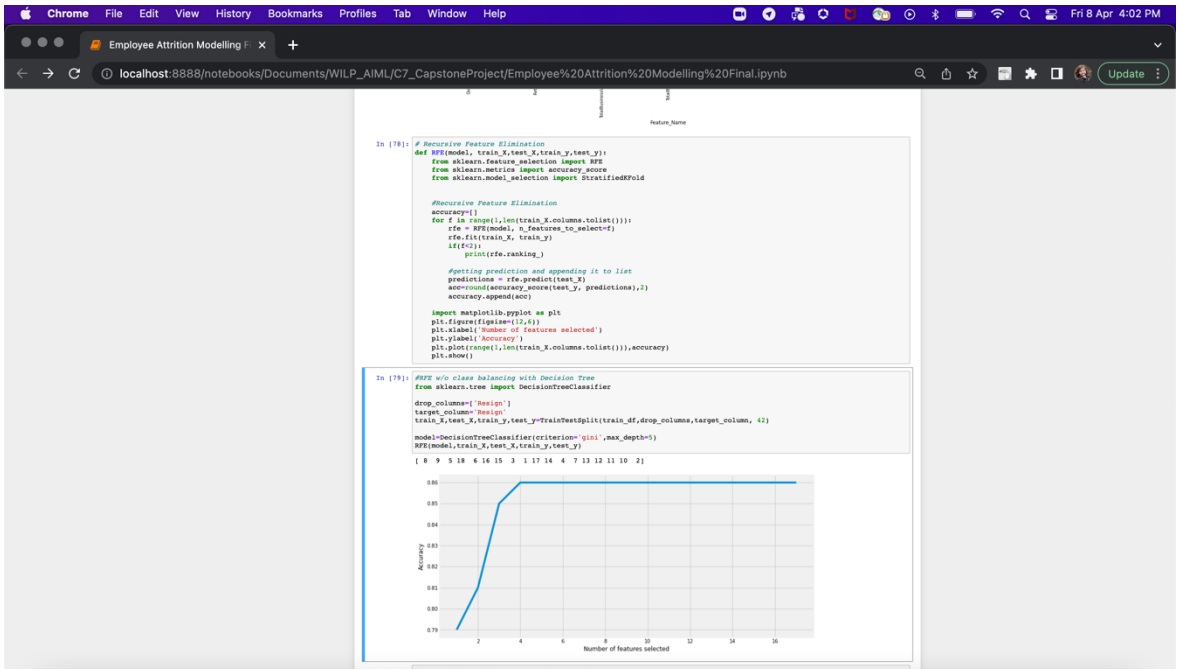


Figure 137: Code snippet: Accuracy vs feature count from RFE with Decision Tree

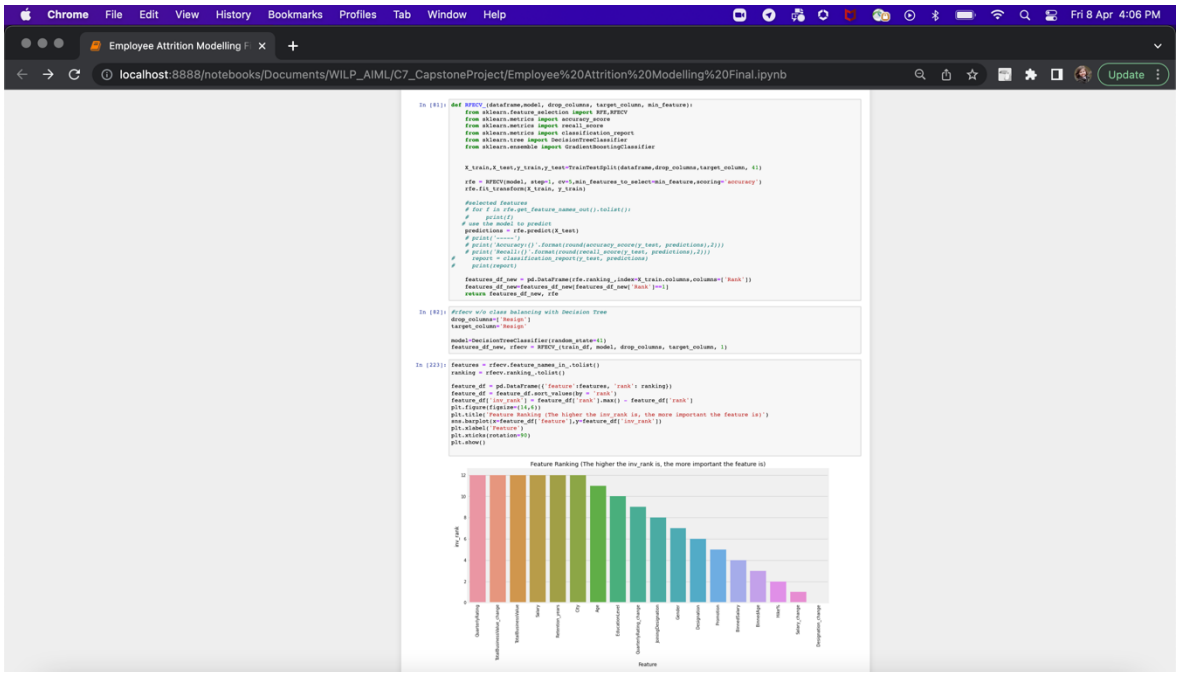


Figure 138: Code snippet: Feature ranking from RFECV with Decision Tree

Appendix C: Modelling

This appendix consists of the description of code snippets in Modelling section.

Multiple datasets have been created for modelling as displayed in the code snippet (Figure 135) above.

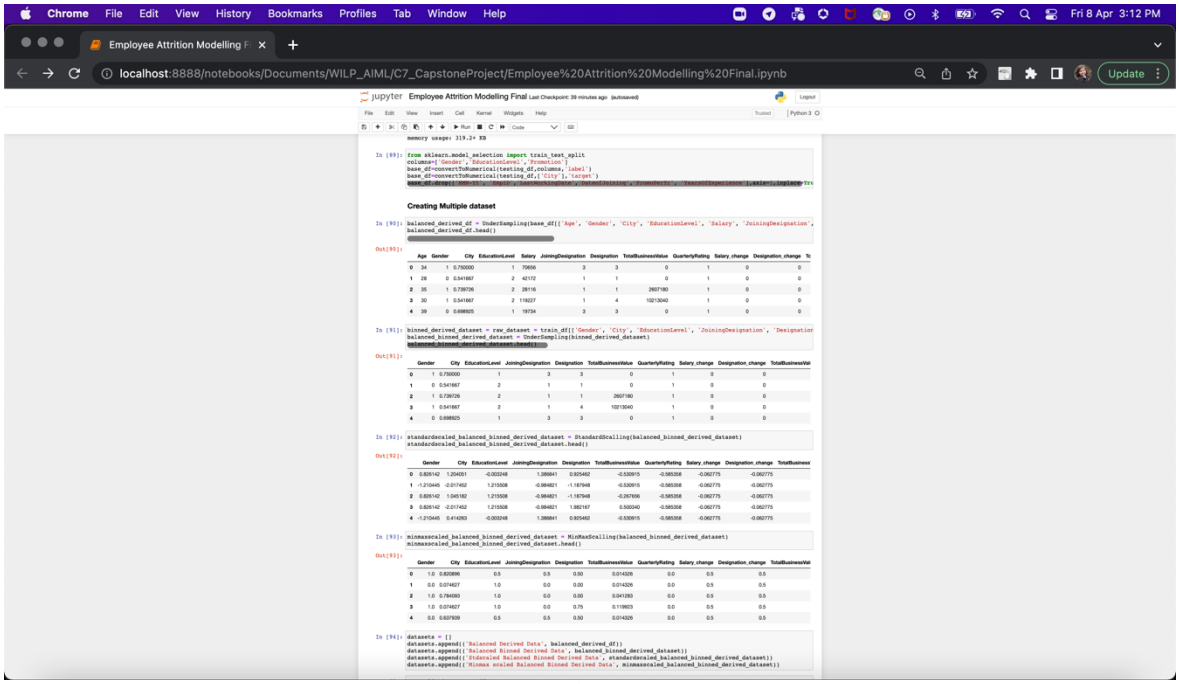


Figure 139: Code snippet: Dataset creation

Base models are created (i.e. using default parameters) and iterated over the datasets to achieve the

results before any hyper parameter tuning.

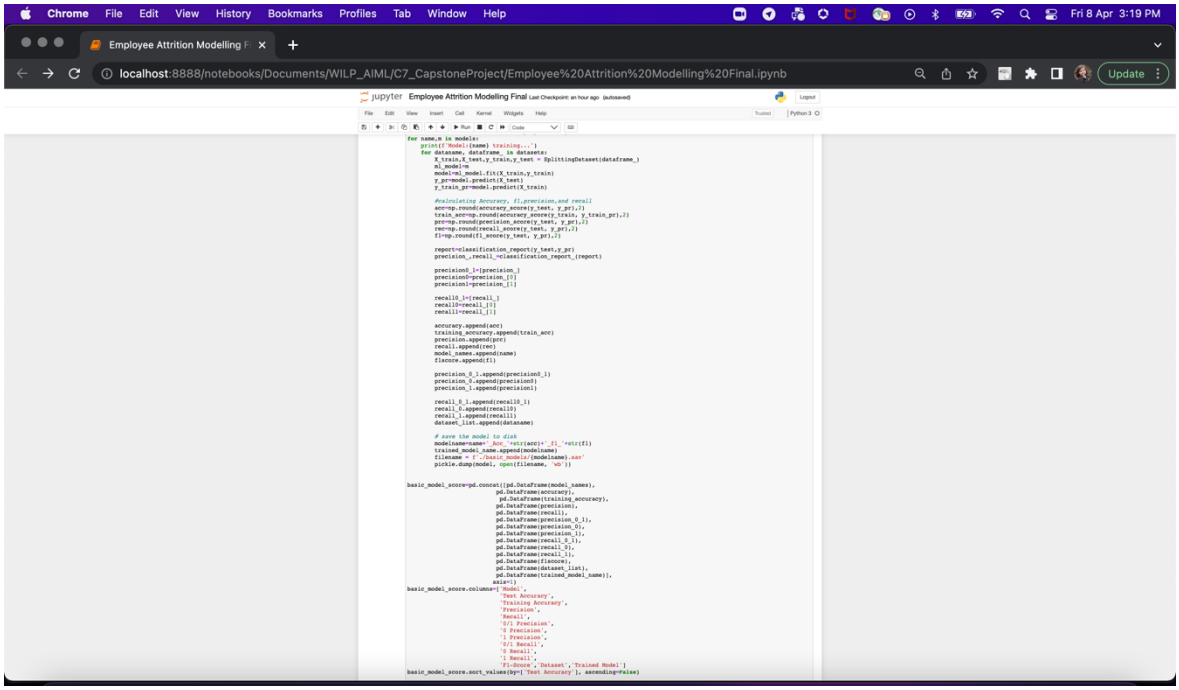


Figure 140: Code snippet: Base Models

Base models are again iterated over the datasets to conclude the cross-validation score.

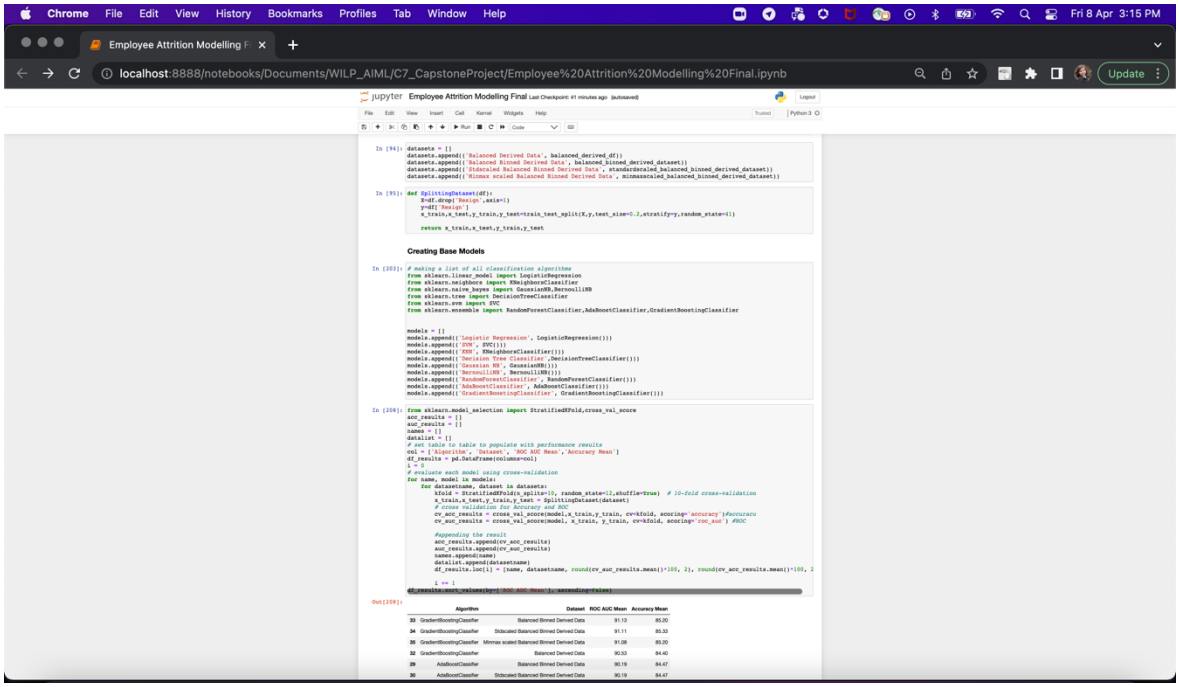


Figure 141: Code snippet: Base Models with cross-validation score

Functions have been created to plot the Learning curve and the count vs predicted probability intervals.

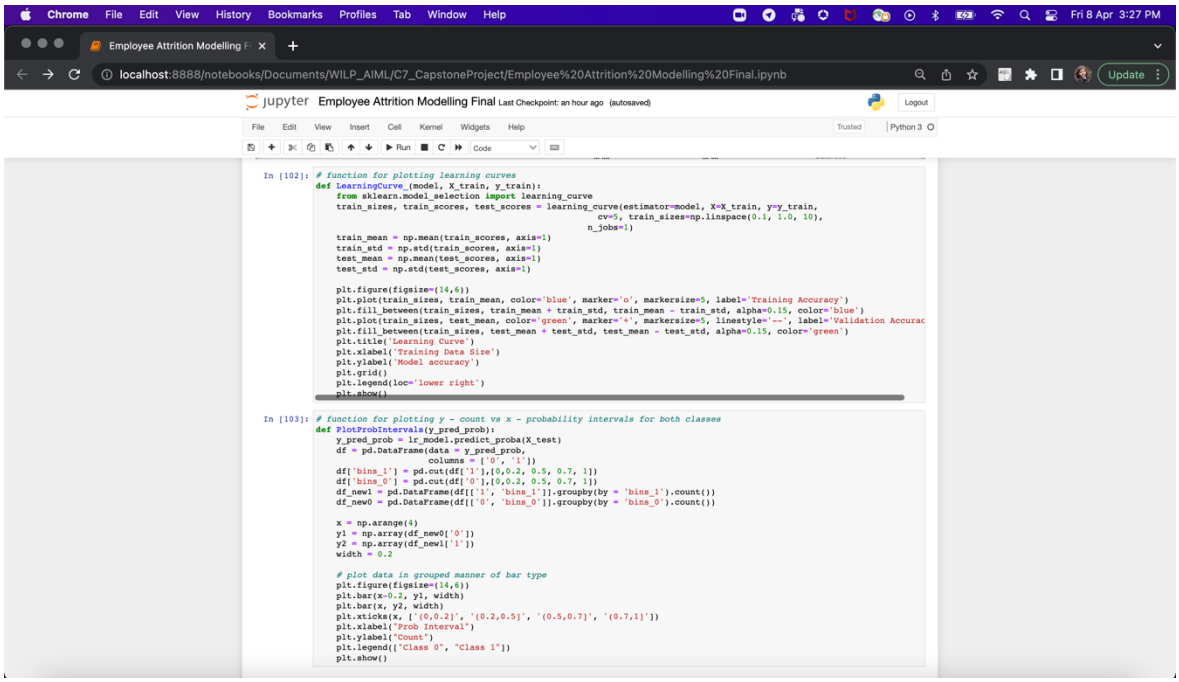


Figure 142: Code snippet: Graphs for Model results

Best parameters collected from implementing Randomized Search CV on each model over each dataset has been stored. After visualizing results from all the Modelling algorithms, all possible models built were summarized using the stored parameters.

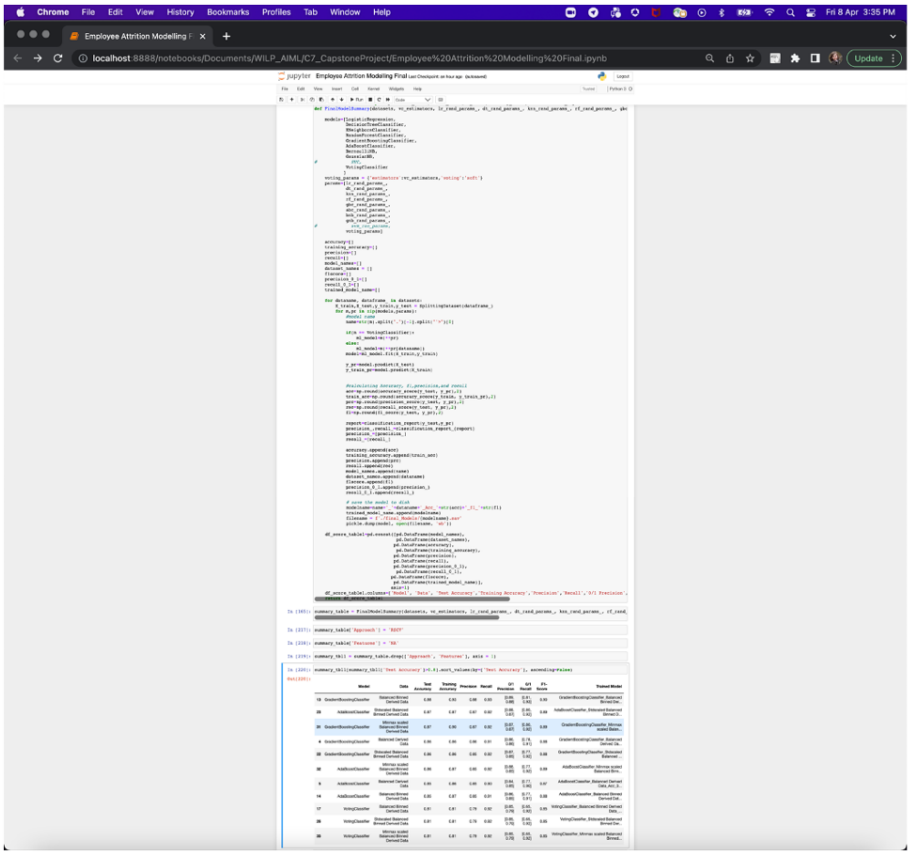


Figure 143: Code snippet: Summarizing RSCV models

After visualizing results from all the possible Modelling algorithms implemented with Recursive

Feature Elimination with cross-validation, all these models built were summarized using the stored parameters.

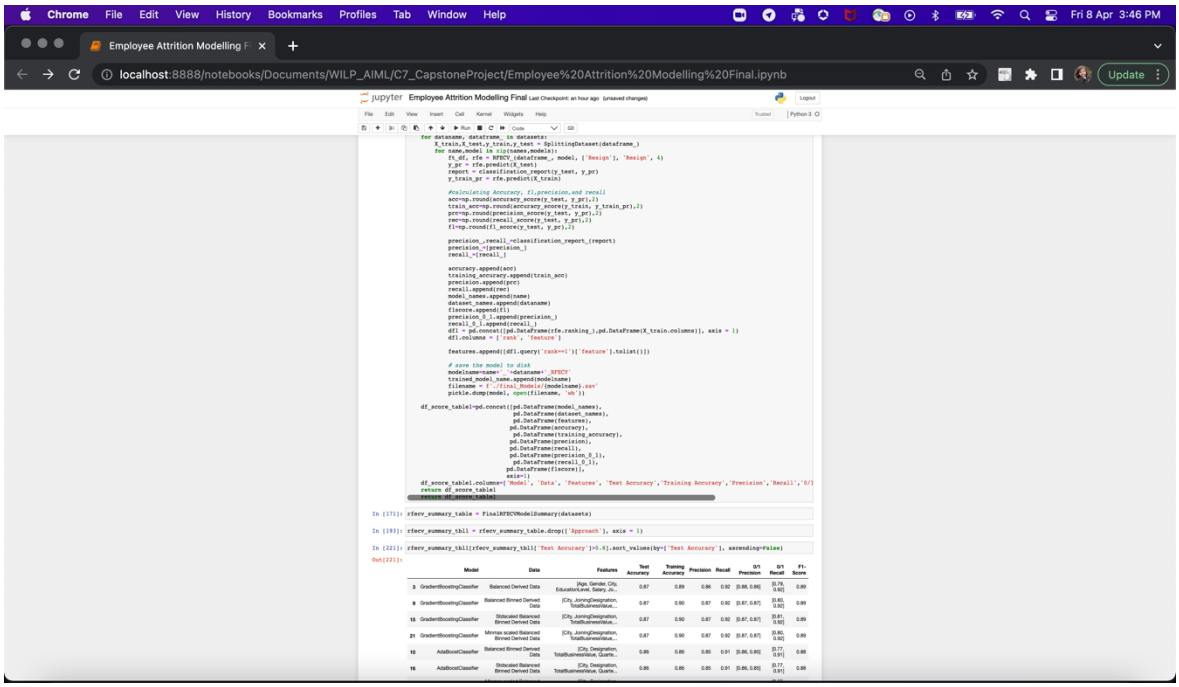


Figure 144: Code snippet: Summarizing RFECV models

Check list of items for the Final report

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? Y
    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

Student ID	Student/ Professor Name	Signature
2020AIML518	KIRTI RANI	Kirti
2020AIML525	VIJAY KUMAR	Vijay
2020AIML528	ABHISHEK SHUKLA	Abhishek
2020AIML531	NIMANSHA MALIK	Nimansha
2020AIML591	K RAMAKRISHNAN	K. Ramakrishnan
	SATYAKI DASGUPTA	