Predicting Employee Attrition

Capstone Project

Authored by

Akhila Sri Medarametla

811189960

Contents

Introduction:

In business, employee attrition is when employees leave the company for whatever reason, either they've found a new job or retired, and haven't been replaced immediately. For a company to be successful, it needs not only to attract top talent but also needs to retain these talents. For this reason, I investigated a dataset containing information regarding a certain company's employee list to try to find patterns that may provide useful information in understanding why employees leave. How can we make the work environment better? What kind of changes must be done? These are important questions to be asked. Thus, I have performed EDA on the model to analyze the importance of the features in the model and have performed Classification using 5 Classification Models. The data contains 19,104 instances (employees)

Problem statement:

To predict the factors that affect an employee to leave the organization in and to find the model that can help the company to predict whether the employee attrition at the joining based on attributes like Demographics of the employee (city, age, gender etc.), Tenure information (joining date, Last Date), Historical data regarding the performance of the employee (Quarterly rating, Monthly business acquired, designation, salary).To uncover the factors that lead to employee attrition and explore important questions such as ‘show me a breakdown of distance from home by job role and attrition’ or ‘compare average monthly income by education and attrition’.

Overview of the dataset:

The following are the columns in our dataset:

-employee id

-age

-city

-gender

-education level

-salary

-date of joining

-last working date

-joining designation

-designation

-total business value

-quarterly rating

Literature:

Introduction-

Machine learning is one of the artificial intelligence technologies that provide systems with the ability to automatically learn and improve from experience or gain human-like intelligence without explicit programming. Employee attrition causes the organization to bear the cost of business disruption, hiring and training new staff. As a result, prediction on employee attrition and identifying the major contributing factors that lead to attrition becomes an important objective of an organization in order to enhance its human resource strategy. IBM created a fictional dataset by its scientist. The dataset contains several attributes influencing the predicted variable named "Attrition" which signifies whether an employee left the company or not from 1,470 instances and 35 attributes. The purpose of this study is to conduct a comparative study to develop machine learning models, i.e., Decision Tree, Support Vector Machine, and Artificial Neural Networks, for predicting probable employee attrition and compare between the algorithms in terms of their accuracy and efficiencies.

Characteristics and implementation-

To detect outliers and any unusual pattern about the dataset using statistical methods data pre-processing is done and also Data visualization to understand each attribute pattern. Any irrelevant attributes that are not contributing to the objectives of this study are removed.

Features –

Feature selection is a process of data reduction that helps to improve accuracy, reduce overfitting, reduce training time and identify the fields that are most important and predictive for a given analysis and also to reduce the dimensionality of the dataset. The k-fold cross validation is applied to the training set in view of its simplicity. Generally, it results in a less biased or less optimistic estimate of the model trained as compared to the other methods, such as the simple train/test split. The data quality report indicated an imbalance in the class distribution, with 237 tuples predicted as "Yes" and 1233 tuples predicted as "No".

Evaluation—

There are three algorithms that are used in this study:

1) Decision Tree that classifies instances by sorting them based on feature values. The basic algorithm for decision tree induction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner .The decision tree is induced by various algorithms. However, as it grows deeper, it happens that sometimes it generates unwanted and meaningless, and this is called overfitting. Therefore, pruning is needed to reduce the size of the tree that is too large and deep. In this study, pruning parameters such as the confidence factor and the number of objects (at the leaf node) were tuned to improve the DT classifier's performance. The basic idea of SVM is to separate classes with maximum margin created by hyper planes.

SVM is known as a popular supervised algorithm in machine learning. Also, based on literature, SVM is also commonly used for employee attrition dataset. SVM acts as a classifier that categorizes the data into different "classes" or as a regression function to estimate the numerical value of the desired output based on a linear combination of features for both linear and non-linear data. In relation to his study, the SVM model which is based on the training dataset, will try to generalize the input data based on their features and make a prediction. SVM machine learning will then produce a model that predicts the test data's target values. The basic idea of SVM is to separate classes with maximum margin created by hyper planes. The tuning parameter in SVM includes the kernel, regularization parameter (C parameter), and gamma. Polynomial and exponential kernels calculate separation lines in a higher dimension called kernel tricks.

ANN is a machine learning technique that acquires knowledge through learning and is used to solve classification problems. The ANN can be organized in different topologies/architectures. During the training of the dataset, the function f is optimized, where the network output for the input vectors in X is as close as possible to the target values in Y. Matrices X and Y represent the training data.

Results:

Four measures are used to compare the performance of the three classifiers being studied i.e., J48, SVM, and ANN are accuracy rate, error rate, root mean square error (RMSE), receiver operating characteristic (ROC), and the time taken or speed to build a model. The RMSE indicates an absolute measure of the fitness of the training dataset. ANN had the highest accuracy result at 86.76% while SVM showed the lowest at 81.97%. ANN showed the best RMSE with the lowest value of 0.3359 and also best ROC at the highest value of 0.922. J48 achieved the best time to build a model at 0.02 sec. The Effect of Feature Selection on Classification Accuracies is that 10-fold cross-validation test option enables the accuracy improvement of 15 attributes in comparison to 30 attributes. Based on the table, the results indicated that the use of top 15 attributes through feature selection has very much reduced the time taken to build the model from 330.23sec to 28.01sec without affecting the accuracy much where there is only a slight change from 85.96% to 85.13%. The result in the training dataset represents the best result for each classifier after applying parameter tuning and regularization. The results were then be compared with the unseen/test data. From the results, SVM is revealed to be the best model that separates the class that can later be used to decide the class of a new set of data in predicting attrition. DT showed the lowest accuracy rate of 84.40%

Conclusion—

The comparative study on IBM Human Resource Analytic Employee Attrition and Performance was conducted to evaluate the classification models, i.e., J48, SVM, and ANN. Each of the three classifiers used in this study has advantages and limitations; thus, evaluation is required to determine its suitability to solve the problem in relation to the dataset being studied. Also data pre-processing is an important stage to ensure only relevant features are selected for the training set. However, SVM showed the highest accuracy after the parameter tuning due to its capacity to handle high-dimensional data with the use of different kernel functions. Hence, future work may look into identifying the key features that lead to employee attrition.

Data preparation and data exploration steps

This dataset is about employee attrition prediction. The data contains 19,104 instances (employees) with other features such as Age, gender, city, Date of joining, Last working date, Designation etc.

Importing the required libraries:

We have imported the following libraries which are required to implement machine learning models

```
import numpy as np
import pandas as pd
import warnings
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline


from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import (RandomForestClassifier)

from sklearn.model_selection import (train_test_split,GridSearchCV)
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import (FunctionTransformer, StandardScaler)

from sklearn.metrics import(classification_report, roc_auc_score)
from sklearn.inspection import plot_partial_dependence

from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler


plt.style.use('ggplot')
```

The data is read in the Google colab to check what it consists of.

```
df.head()
```

]:

	MMM-YY	Emp_ID	Age	Gender	City	Education_Level	Salary	Dateofjoining	LastWorkingDate	Joining Designation	Designation	Total Business Value	Quarterly Rating
0	2016-01-01	1	28	Male	C23	Master	57387	2015-12-24	NaN	1	1	2381060	2
1	2016-02-01	1	28	Male	C23	Master	57387	2015-12-24	NaN	1	1	-665480	2
2	2016-03-01	1	28	Male	C23	Master	57387	2015-12-24	2016-03-11	1	1	0	2
3	2017-11-01	2	31	Male	C7	Master	67016	2017-11-06	NaN	2	2	0	1
4	2017-12-01	2	31	Male	C7	Master	67016	2017-11-06	NaN	2	2	0	1

```
# we read the data and shown the top 5 observations
```

```
# give information of the data set
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
```

Exploratory Data Analysis is being done for data preprocessing:

EDA

```
In [7]: df.isnull().sum()
```

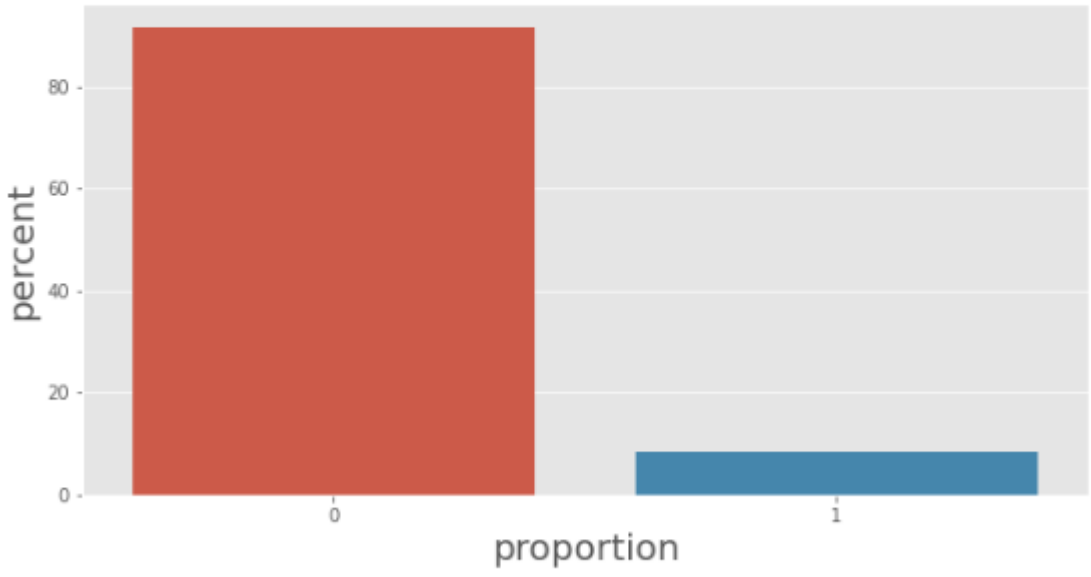
```
Out[7]: MMM-YY                     0
        Emp_ID                     0
        Age                        0
        Gender                     0
        City                       0
        Education_Level            0
        Salary                     0
        Dateofjoining              0
        LastWorkingDate        17488
        Joining Designation        0
        Designation                0
        Total Business Value       0
        Quarterly Rating           0
        dtype: int64
```

we can observe that we have null values in lastworking date column as lastworking date is a dependent variable,creating one more variable considering all not null values

We have created a dependent variable i.e. “main” with last working days values

```
df['main'] = df.LastWorkingDate.notnull().astype(int)
df_plot = df['main'].value_counts(normalize=True) * 100
plt.figure(figsize=(10, 5))
sns.barplot(df_plot.index, df_plot);
plt.xlabel('proportion', fontsize=20);
plt.ylabel('percent', fontsize=20)
```

```
3]: Text(0, 0.5, 'percent')
```



we created a dependent variable main without null values plotted graph gives us a proportion of 9:1

Dropping the duplicate values

```
df.drop_duplicates(inplace=True)
df
```

[9]:

	MMM-YY	Emp_ID	Age	Gender	City	Education_Level	Salary	Dateofjoining	LastWorkingDate	Joining Designation	Designation	Total Business Value	Quarterly Rating	main
0	2016-01-01	1	28	Male	C23	Master	57387	2015-12-24	NaN	1	1	2381060	2	0
1	2016-02-01	1	28	Male	C23	Master	57387	2015-12-24	NaN	1	1	-665480	2	0
2	2016-03-01	1	28	Male	C23	Master	57387	2015-12-24	2016-03-11	1	1	0	2	1
3	2017-11-01	2	31	Male	C7	Master	67016	2017-11-06	NaN	2	2	0	1	0
4	2017-12-01	2	31	Male	C7	Master	67016	2017-11-06	NaN	2	2	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19099	2017-08-01	2788	30	Male	C27	Master	70254	2017-06-08	NaN	2	2	740280	3	0
19100	2017-09-01	2788	30	Male	C27	Master	70254	2017-06-08	NaN	2	2	448370	3	0
19101	2017-10-01	2788	30	Male	C27	Master	70254	2017-06-08	NaN	2	2	0	2	0

We have 2381 employees' data

```
In [10]: df.Emp_ID.nunique(), df['MMM-YY'].min(),df['MMM-YY'].max()
Out[10]: (2381, '2016-01-01', '2017-12-01')
```

we have data of 2381 employees from starting date 2016-01-01 to end date of 2017-12-01

As a part of the process we have dropped couple of columns which are not useful.

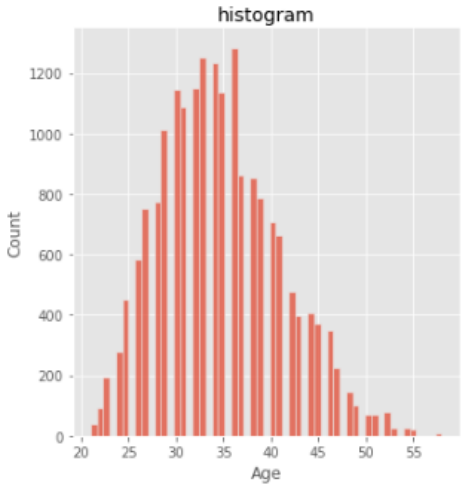
Dropping unimportant columns

```
.]: df.drop(['Joining Designation','Designation'],axis=1, inplace=True)
```

We have taken salary, education level, and age histograms and observed the following:

AGE HISTOGRAM

In [12]:
```
sns.displot(df.Age, kind='hist')
plt.title('histogram');
```



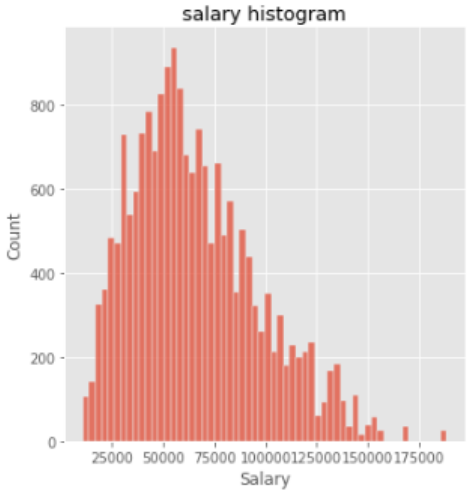
histogram is almost similar to a normal distribution

In [13]:
```
print('The mean of the age is: %.2f and the median is: %i' %(df.Age.mean(),df.Age.median()))
```

```
The mean of the age is: 34.65 and the median is: 34
```

SALARY HISTOGRAM

In [14]:
```
sns.displot(df.Salary, kind='hist')
plt.title('salary histogram');
```



salary histogram is skewed to the right

so lets check the mean and median for salary as it is positively skewed

In [15]:
```
print('The mean of the salary is: %i and the median is: %i' %(df.Salary.mean(),df.Salary.median()))
```

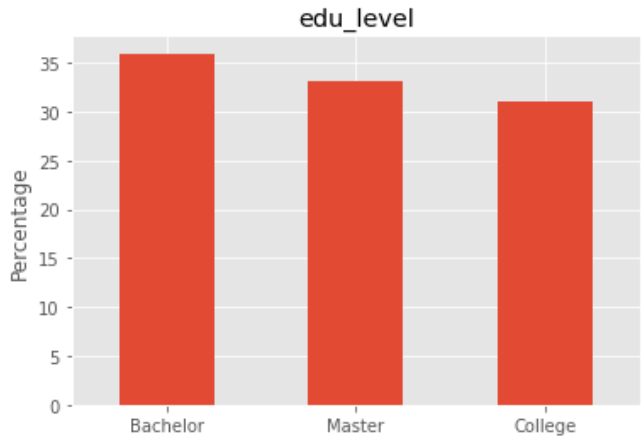
```
The mean of the salary is: 65652 and the median is: 60087
```

Here we can observe that the graph is skewed to the right i.e. it is a distribution where the mean median and the mode are positive.

The difference between the mean and the median is 5000USD

EDUCATION HISTOGRAM

```
#lets observe how education column is affecting the attrition
(df.Education_Level.value_counts(normalize=True)*100).plot.bar(rot=0, title='edu_level')
plt.ylabel('Percentage');
```

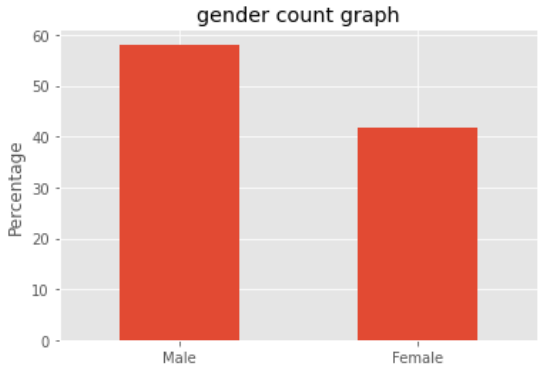


They have almost similar proportion at the education level

From the bar plot we can say that the proportions are same

GENDER GRAPH

```
In [16]: (df.Gender.value_counts(normalize=True)*100).plot.bar(rot=0, title='gender count graph')
plt.ylabel('Percentage');
```



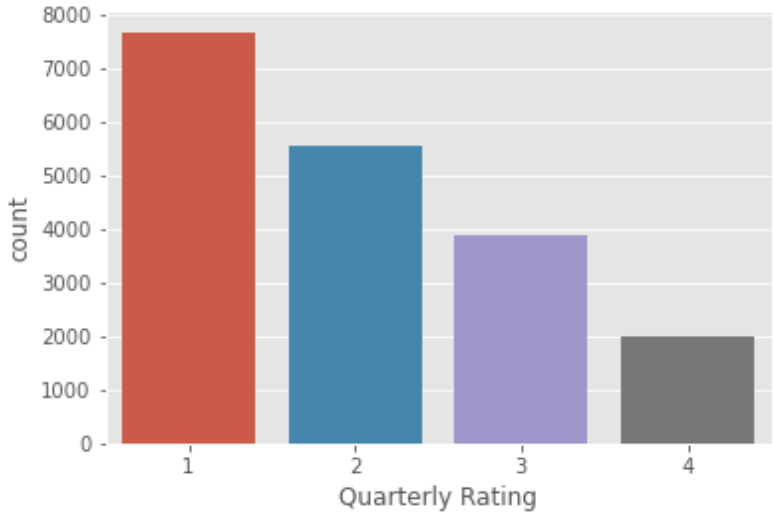
we can observe that there are more males than females

There is almost a difference of 16%

QUARTERLY RATINGS

```
In [21]: sns.countplot(data=df,x='Quarterly Rating')
```

```
Out[21]: <AxesSubplot:xlabel='Quarterly Rating', ylabel='count'>
```



Decrease in the ratings can be observed from the above graph

Comparing the quarterly ratings for the employees present and left can be done using the attribute "main"

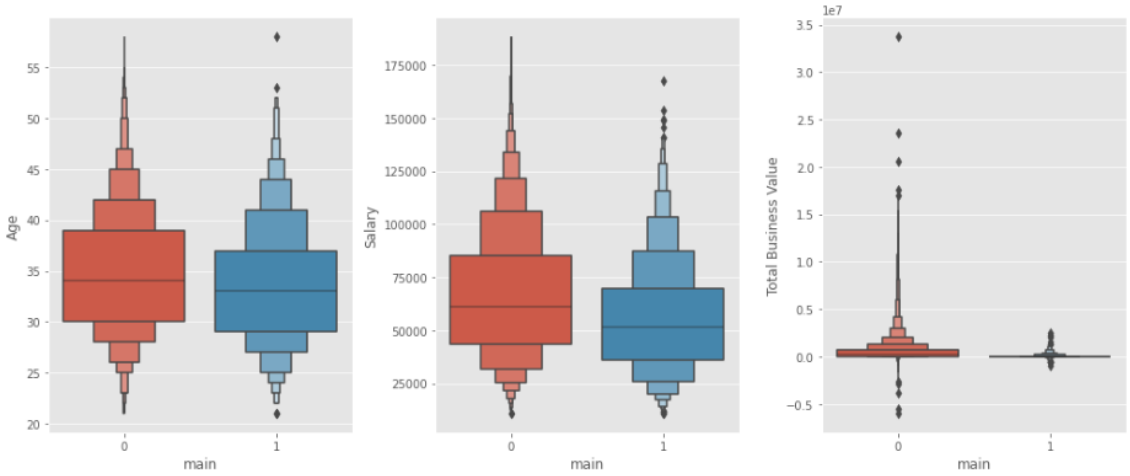
```
In [23]: df.groupby('main')['Quarterly Rating'].value_counts().unstack()
```

Out[23]:

Quarterly Rating	1	2	3	4
main				
0	6247	5407	3867	1967
1	1432	146	28	10

Now we know how our data is, our next job is to describe the relationship with the main variable

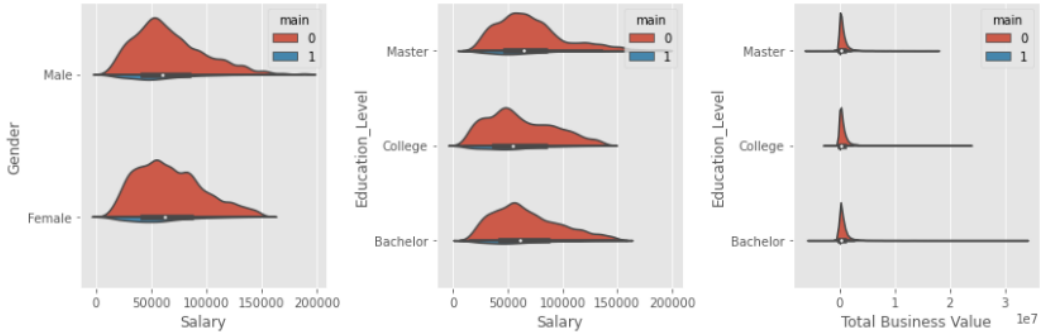
In [26]:
```
plt.figure(figsize=(14,6))
for idx,col in enumerate(num_cols):
    df_temp = df[['main']+[col]]
    plt.subplot(1,3,idx+1)
    sns.boxenplot(data=df_temp,x='main',y=col)
    plt.tight_layout()
```



These are the observations made from the graph above

- for the total business value, we see a lot of outliers
- We can observe that young employees have a little more chance to quit their job
- we can observe that low salary could be one of the reasons

In [28]:
```
plt.figure(figsize=(12,4))
plt.subplot(1,3,1)
sns.violinplot(data=df, y='Gender', x='Salary', hue='main', split=True, scale='count')
plt.subplot(1,3,2)
sns.violinplot(data=df, y='Education_Level', x='Salary', hue='main', split=True, scale='count')
plt.subplot(1,3,3)
sns.violinplot(data=df, y='Education_Level', x='Total Business Value', hue='main', split=True, scale='count')
plt.tight_layout()
```



We have plotted our graphs in such a way that we have compared them with our main value and the proportions of the positive case are low, we cannot see a great difference. An important observation is that the employees who did master's degrees have more salary. More salary then fewer people quitting the job

Let's create some features that we think would be helpful to our machine learning model

Visualization

```
In [31]: df['main'].value_counts()

Out[31]: 0    17488
         1     1616
         Name: main, dtype: int64
```

```
In [32]: from sklearn.utils import resample
         df_1 = df[df['main']==0]
         df_2 = df[df['main']==1]

         df_2_upsampled = resample(df_2,
                                   replace=True,     # sample with replacement
                                   n_samples=17488,    # to match majority class
                                   random_state=123) # reproducible results

         # Combine majority class with upsampled minority class
         df_upsampled = pd.concat([df_1, df_2_upsampled])

         # Display new class counts
         df_upsampled.main.value_counts()

Out[32]: 0    17488
         1    17488
         Name: main, dtype: int64
```

The data before unsampling is –

```
[33]: df_upsampled
```

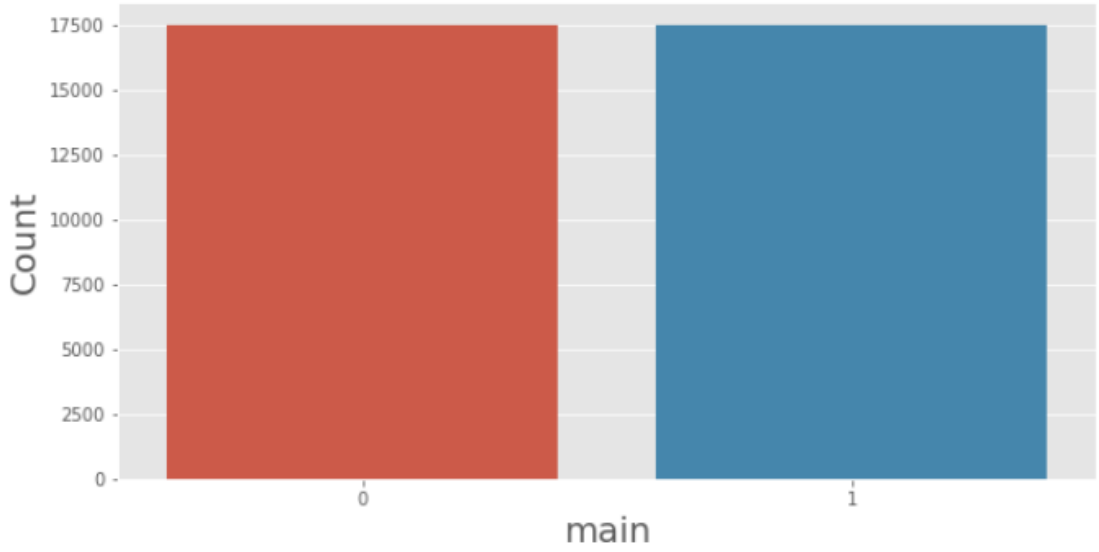
Out[33]:

	MMM-YY	Emp_ID	Age	Gender	City	Education_Level	Salary	Dateofjoining	LastWorkingDate	Total Business Value	Quarterly Rating	main
0	2016-01-01	1	28	Male	C23	Master	57387	2015-12-24	NaN	2381060	2	0
1	2016-02-01	1	28	Male	C23	Master	57387	2015-12-24	NaN	-665480	2	0
3	2017-11-01	2	31	Male	C7	Master	67016	2017-11-06	NaN	0	1	0
4	2017-12-01	2	31	Male	C7	Master	67016	2017-11-06	NaN	0	1	0
5	2016-12-01	4	43	Male	C13	Master	65603	2016-12-07	NaN	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
4551	2016-09-01	680	37	Female	C3	Master	127380	2011-07-28	2016-09-22	331570	1	1
11203	2017-11-01	1665	35	Male	C12	Bachelor	52175	2017-04-26	2017-11-21	255480	2	1
16115	2017-10-01	2393	30	Female	C27	College	87010	2017-08-08	2017-10-06	0	1	1
2856	2016-04-01	430	32	Male	C28	College	14489	2016-01-06	2016-03-29	0	1	1
10307	2016-09-01	1536	27	Male	C15	Master	39731	2016-07-19	2016-08-30	0	1	1

34976 rows × 12 columns

```
status_label = df_upsampled.main.value_counts()
plt.figure(figsize=(10, 5))
sns.barplot(status_label.index, status_label);
plt.xlabel('main', fontsize=20);
plt.ylabel('Count', fontsize=20)
```

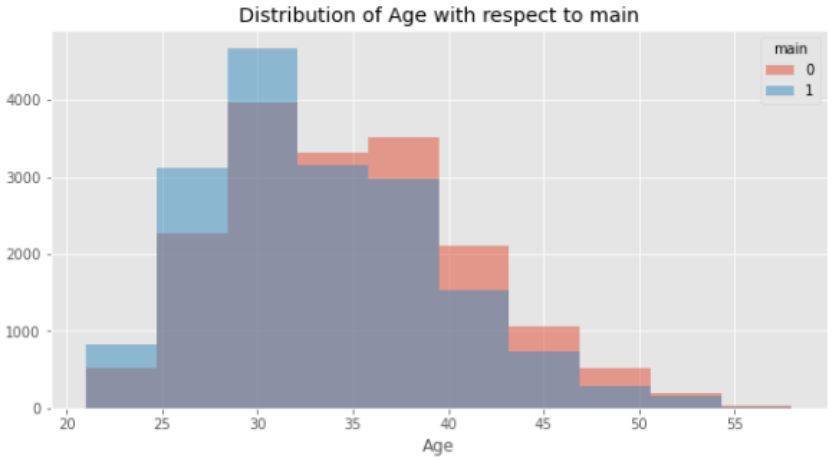
[90]: Text(0, 0.5, 'Count')



From the above graph we can observe that the classes are balanced

Distribution of age with respect to main

In [39]:
```
plt.figure(figsize=(10, 5))
plt.hist("Age", data = df_upsampled[df_upsampled["main"] == 0], alpha = 0.5, label = "0")
plt.hist("Age", data = df_upsampled[df_upsampled["main"] == 1], alpha = 0.5, label = "1")
plt.title("Distribution of Age with respect to main")
plt.xlabel("Age")
plt.legend(title = "main")
plt.show()
```



Here we can observe that young people are quitting more than the elder ones

In [41]:
```
df_upsampled.head()
```

Out[41]:

	MMM-YY	Emp_ID	Age	Gender	City	Education_Level	Salary	Dateofjoining	LastWorkingDate	Total Business Value	Quarterly Rating	main
0	2016-01-01	1	28	Male	C23	Master	57387	2015-12-24	NaN	2381060	2	0
1	2016-02-01	1	28	Male	C23	Master	57387	2015-12-24	NaN	-665480	2	0
3	2017-11-01	2	31	Male	C7	Master	67016	2017-11-06	NaN	0	1	0
4	2017-12-01	2	31	Male	C7	Master	67016	2017-11-06	NaN	0	1	0
5	2016-12-01	4	43	Male	C13	Master	65603	2016-12-07	NaN	0	1	0

City

In [43]:
```
city_name_label = {value: key for key, value in enumerate(df_upsampled['City'].unique())}
df_upsampled['City'] = df_upsampled['City'].map(city_name_label)
```

education_level

In [45]:
```
Education_Level_label = {value: key for key, value in enumerate(df_upsampled['Education_Level'].unique())}
df_upsampled['Education_Level'] = df_upsampled['Education_Level'].map(Education_Level_label)
```

salary

In [47]:
```
Salary_label = {value: key for key, value in enumerate(df_upsampled['Salary'].unique())}
df_upsampled['Salary'] = df_upsampled['Salary'].map(Salary_label)
```

Gender

In [49]:
```
Gender_label = {value: key for key, value in enumerate(df_upsampled['Gender'].unique())}
df_upsampled['Gender'] = df_upsampled['Gender'].map(Gender_label)
```

total business value

In [51]:
```
BusinessValue_label = {value: key for key, value in enumerate(df_upsampled['Total Business Value'].unique())}
df_upsampled['Total Business Value'] = df_upsampled['Total Business Value'].map(BusinessValue_label)
```

quarterly rating

In [53]:
```
QuarterlyRating_label = {value: key for key, value in enumerate(df_upsampled['Quarterly Rating'].unique())}
df_upsampled['Quarterly Rating'] = df_upsampled['Quarterly Rating'].map(QuarterlyRating_label)
```

main

In [55]:
```
main_label = {value: key for key, value in enumerate(df_upsampled['main'].unique())}
df_upsampled['main'] = df_upsampled['main'].map(main_label)
```

In [56]:
```
df_upsampled.head()
```

Out[56]:

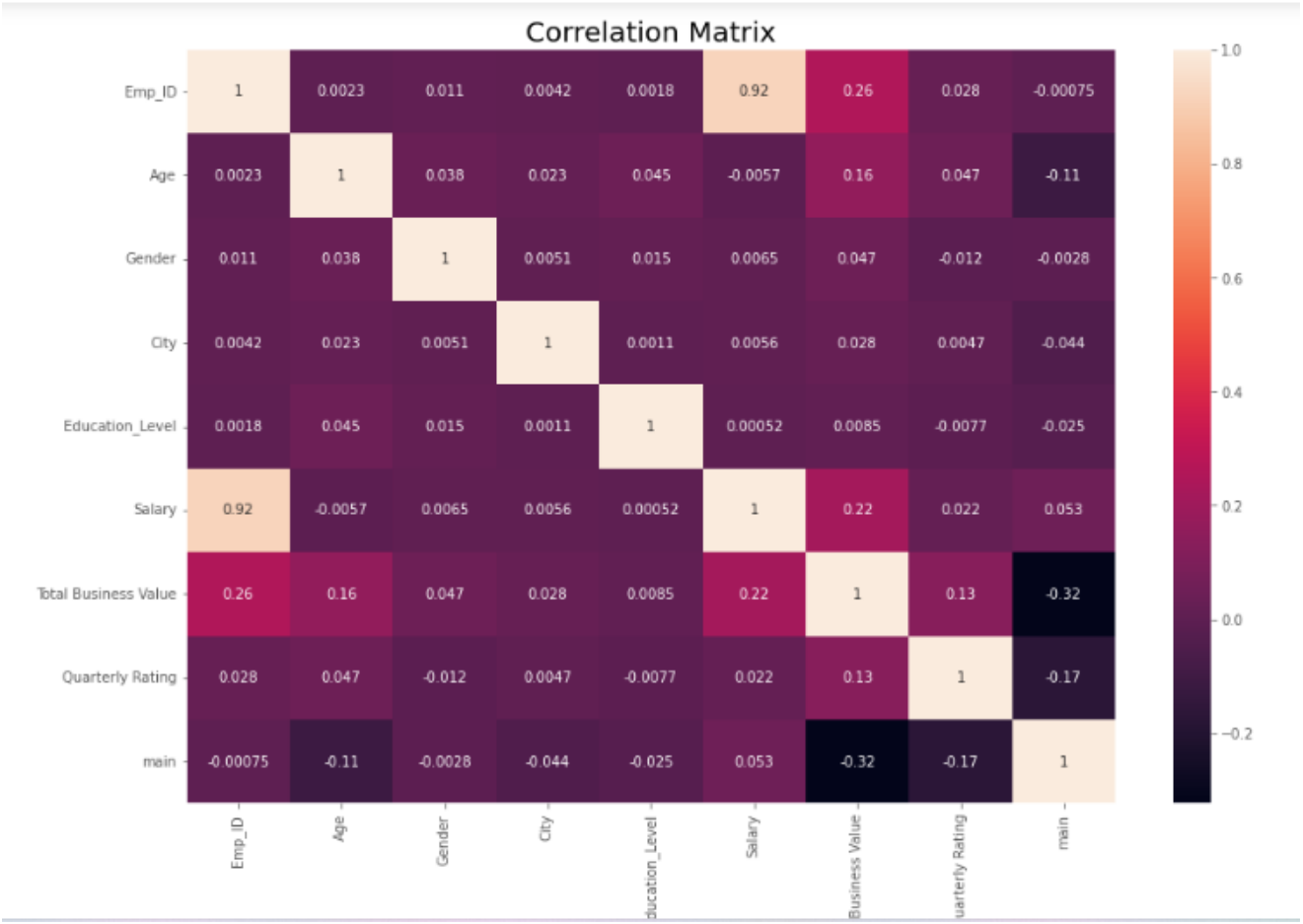
	MMM-YY	Emp_ID	Age	Gender	City	Education_Level	Salary	Dateofjoining	LastWorkingDate	Total Business Value	Quarterly Rating	main
0	2016-01-01	1	28	0	0	0	0	2015-12-24	NaN	0	0	0
1	2016-02-01	1	28	0	0	0	0	2015-12-24	NaN	1	0	0
3	2017-11-01	2	31	0	1	0	1	2017-11-06	NaN	2	1	0
4	2017-12-01	2	31	0	1	0	1	2017-11-06	NaN	2	1	0
5	2016-12-01	4	43	0	2	0	2	2016-12-07	NaN	2	1	0

We can observe that after data preprocessing there is a change in the unsampled data

The correlation matrix for the unsampled data is

Correlation Matrix

:
```
plt.figure(figsize=(15, 10))
sns.heatmap(df_upsampled.corr(), annot=True);
plt.title('Correlation Matrix', fontsize=20);
```



Checking the Correlation of main with repect to other features

In [59]:
```
df_upsampled.corr()['main'].sort_values(ascending=False)[1:]
```

Out[59]:
```
Salary                 0.052602
Emp_ID                -0.000748
Gender                -0.002840
Education_Level       -0.024872
City                  -0.044360
Age                   -0.113846
Quarterly Rating      -0.173231
Total Business Value  -0.323274
Name: main, dtype: float64
```

In [60]:
```
df_upsampled.head()
```

Out[60]:

	MMM-YY	Emp_ID	Age	Gender	City	Education_Level	Salary	Dateofjoining	LastWorkingDate	Total Business Value	Quarterly Rating	main
0	2016-01-01	1	28	0	0	0	0	2015-12-24	NaN	0	0	0
1	2016-02-01	1	28	0	0	0	0	2015-12-24	NaN	1	0	0
3	2017-11-01	2	31	0	1	0	1	2017-11-06	NaN	2	1	0
4	2017-12-01	2	31	0	1	0	1	2017-11-06	NaN	2	1	0
5	2016-12-01	4	43	0	2	0	2	2016-12-07	NaN	2	1	0

MODEL BUILDING:

For model building, we are going to build a supervised learning models
The models we use are:

- Logistic Regression
- K-Nearest Neighbors Classifier
- Gaussian Naive Bayes Classifier
- Decision Tree
- Random Forest
-

are compared in order to decide which classification model is the best.

In [64]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, classification_report, roc_auc_score,roc_curve,auc
```

In [65]:
```
accuracy_list = []
f1_list = []
roc_auc_list = []
```

Train Test Evaluation Function

```
def result(X, y, ts, rs, model):

    #train test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=ts, random_state=rs)

    #scaling
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)

    #fit on data
    model.fit(X_train, y_train)

    #prediction
    pred = model.predict(X_test)

    #performance of model
    print("Classification Report: \n", classification_report(y_test, pred))
    print("-" * 100)
    print()

    #accuracy of model
    acc = accuracy_score(y_test, pred)
    accuracy_list.append(acc)
    print("Accuracy Score: ", acc)
    print("-" * 100)
    print()
```

```
#f1-score of model
f1 = f1_score(y_test, pred)
f1_list.append(f1)
print("F1 Score: ", f1)
print("-" * 100)
print()

#roc-auc curve of model
fpr,tpr,threshold = roc_curve(y_test,pred)
auc_value = auc(fpr,tpr)
rocauc_score = roc_auc_score(y_test, pred)
roc_auc_list.append(rocauc_score)
plt.figure(figsize=(5,5),dpi=100)
print("ROC-AUC Score: ", f1)
print("-" * 100)
print()
plt.plot(fpr,tpr,linestyle='-',label = "(auc_value = %0.3f)" % auc_value)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
print()

#confusion matrix for model
print("Confusion Matrix: ")
plt.figure(figsize=(10, 5))
sns.heatmap(confusion_matrix(y_test, pred), annot=True, fmt='g');
plt.title('Confusion Matrix', fontsize=20)
```

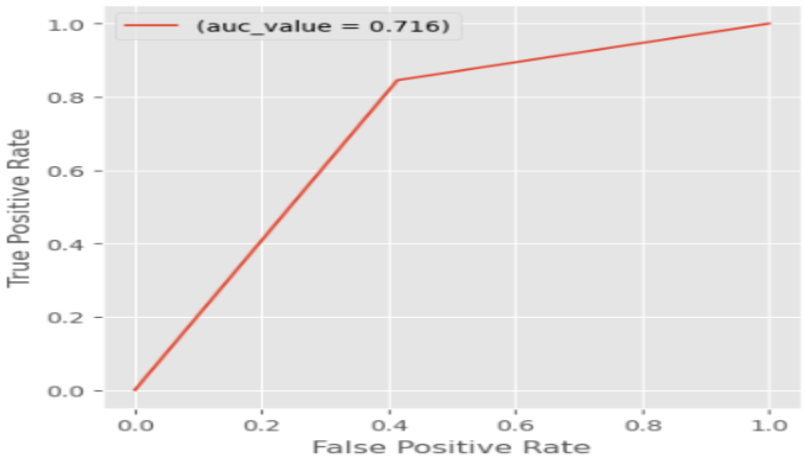
In the above code what we did was that the model that can calculate its accuracy, f1 score and Roc-Auc scores. And at the end confusion matrix is done for all the classifiers

Logistic Regression

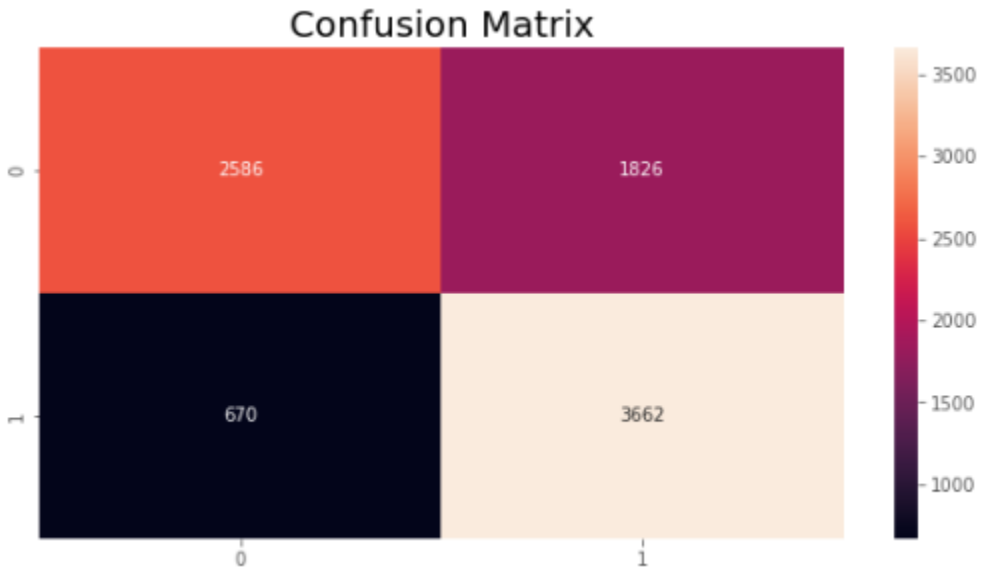
Performance of logistic regression model

```
model = LogisticRegression()
result(x, y, 0.25, 42, model)
```

```
Classification Report:
                precision    recall  f1-score   support

           0       0.79      0.59      0.67      4412
           1       0.67      0.85      0.75      4332

    accuracy                           0.71      8744
   macro avg       0.73      0.72      0.71      8744
weighted avg       0.73      0.71      0.71      8744

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

Accuracy Score:  0.7145471180237878
----------------------------------------------------------------------------------------------------

recall Score:  0.84533702677747
----------------------------------------------------------------------------------------------------

F1 Score:  0.7458248472505092
----------------------------------------------------------------------------------------------------

ROC-AUC Score:  0.7458248472505092
----------------------------------------------------------------------------------------------------
```



Confusion matrix



The harmonic mean of precision and recall i.e. f1 score for logistic regression is 0.7458

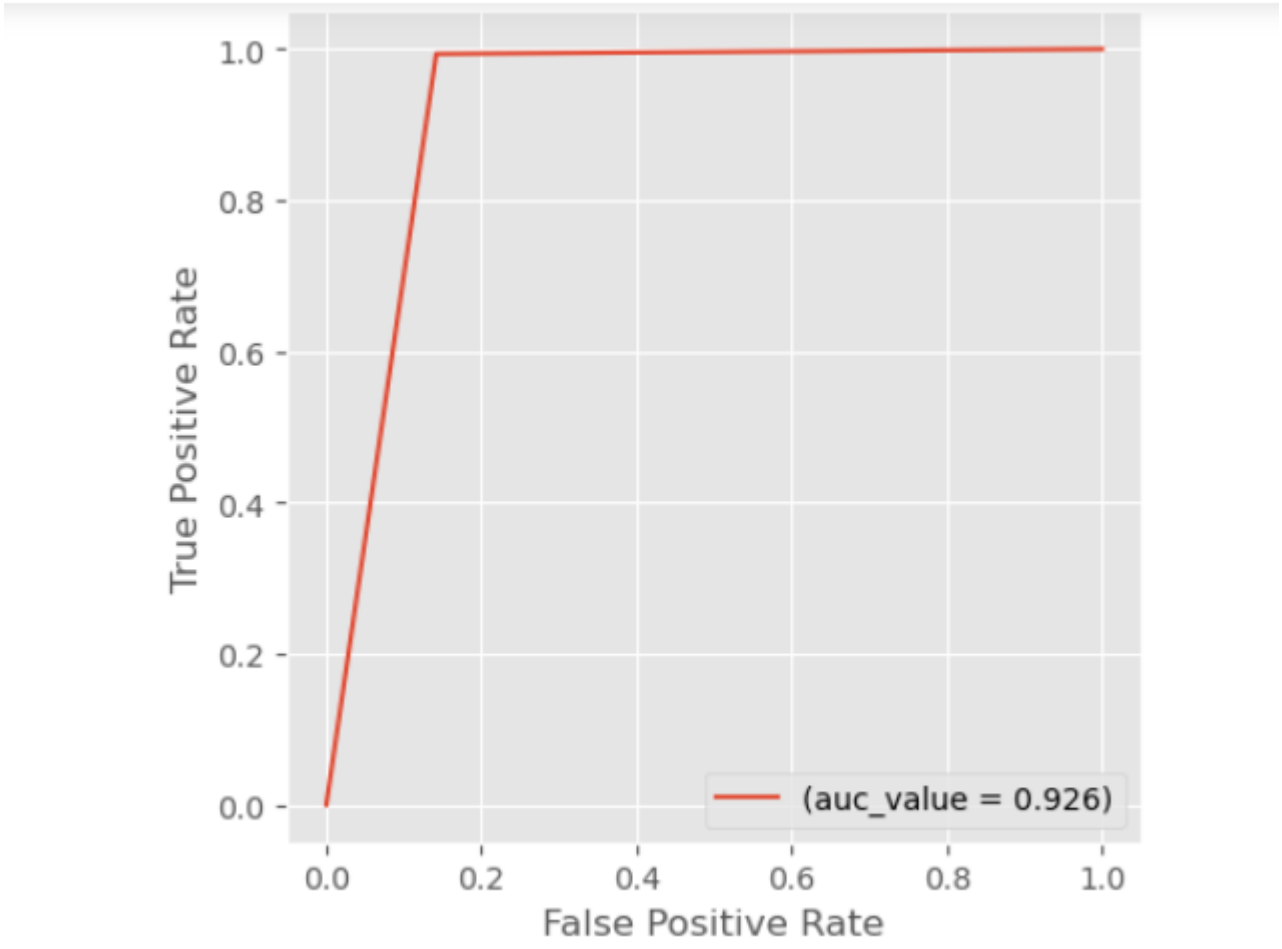
Accuracy score for this model is 0.714

RANDOM FOREST

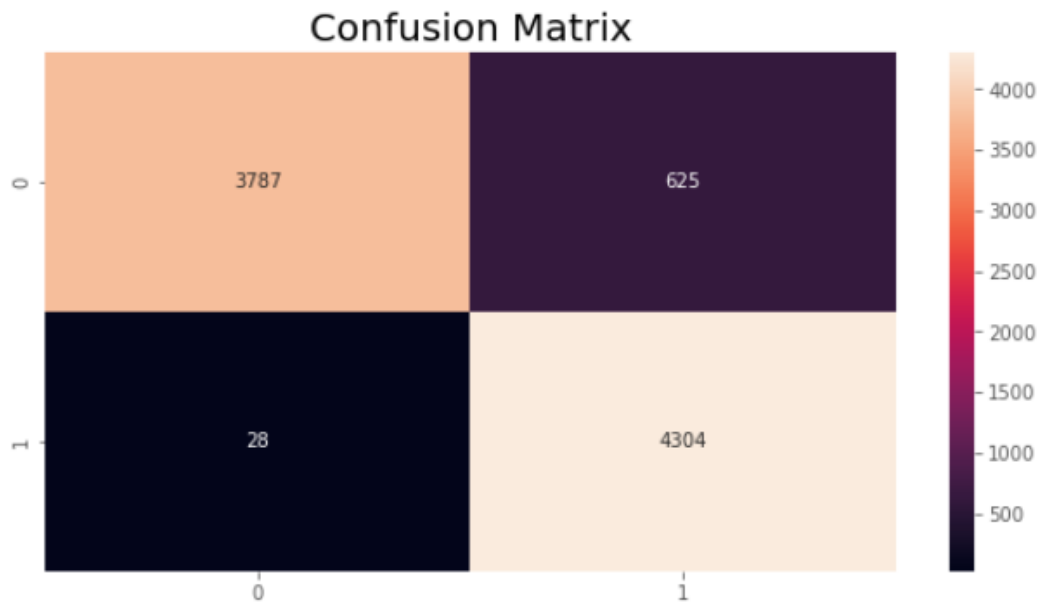
Performance of random forest model

```
rf = RandomForestClassifier()
result(x, y, 0.25, 42, rf)
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.86      0.92      4412
           1       0.88      1.00      0.93      4332

    accuracy                           0.93      8744
   macro avg       0.94      0.93      0.93      8744
weighted avg       0.94      0.93      0.93      8744

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

Accuracy Score:  0.9274931381518756
----------------------------------------------------------------------------------------------------

recall Score:  0.9953831948291783
----------------------------------------------------------------------------------------------------

F1 Score:  0.9315186865413696
----------------------------------------------------------------------------------------------------

ROC-AUC Score:  0.9315186865413696
----------------------------------------------------------------------------------------------------
```



Confusion Matrix:



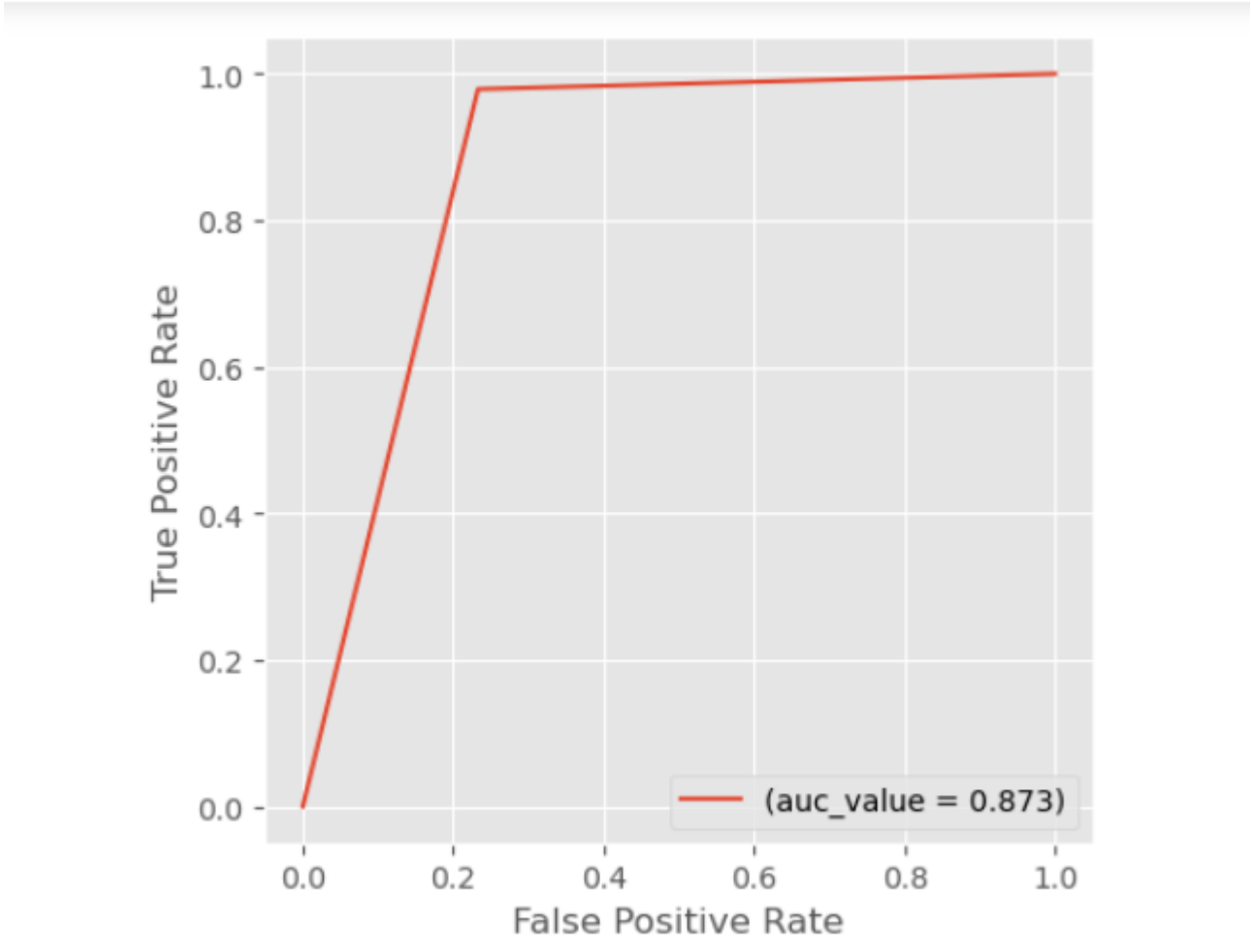
For random forest classifier Accuracy Score: 0.926 F1 Score: 0.93

KNN Classifier

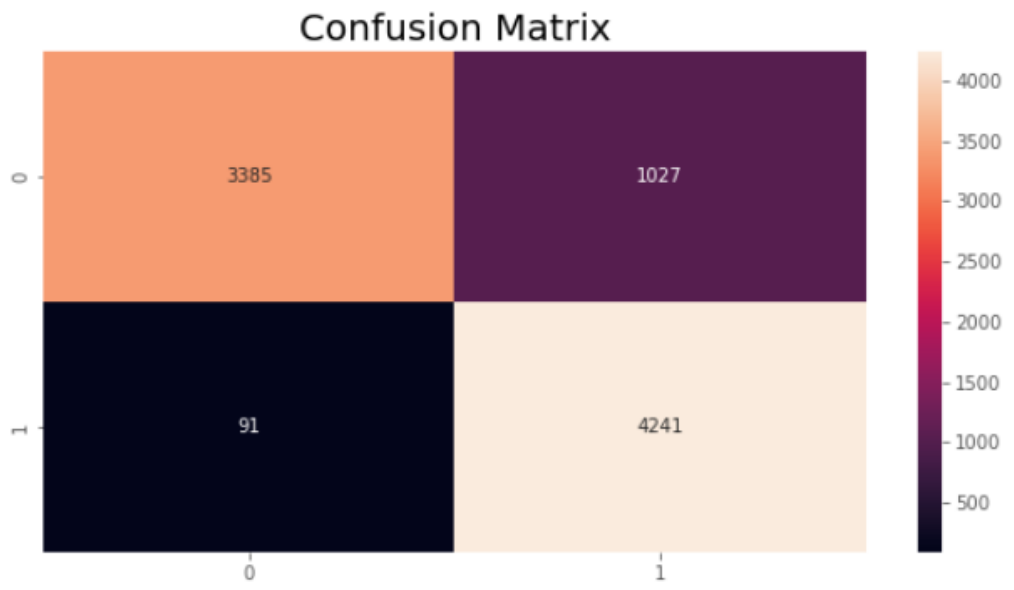
Performance of knn classifier model

```
knn = KNeighborsClassifier()
result(x, y, 0.25, 42, knn)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.77      0.86      4412
           1       0.81      0.98      0.88      4332

    accuracy                           0.87      8744
   macro avg       0.89      0.87      0.87      8744
weighted avg       0.89      0.87      0.87      8744

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

Accuracy Score:  0.8721408966148216
----------------------------------------------------------------------------------------------------

recall Score:  0.9789935364727609
----------------------------------------------------------------------------------------------------

F1 Score:  0.8835416666666666
----------------------------------------------------------------------------------------------------

ROC-AUC Score:  0.8835416666666666
----------------------------------------------------------------------------------------------------
```



Confusion Matrix:



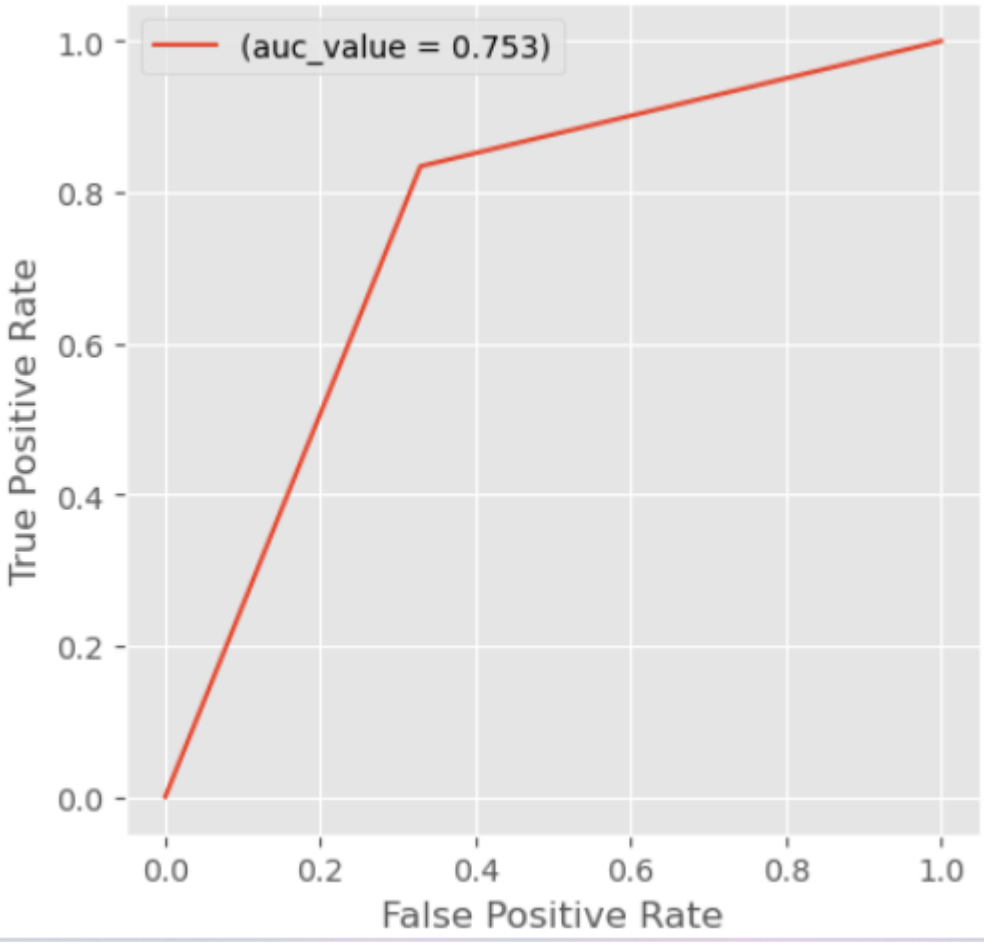
Knn classifier has an accuracy of 0.87 and f1 score is 0.88

Gaussian Naive Bayes

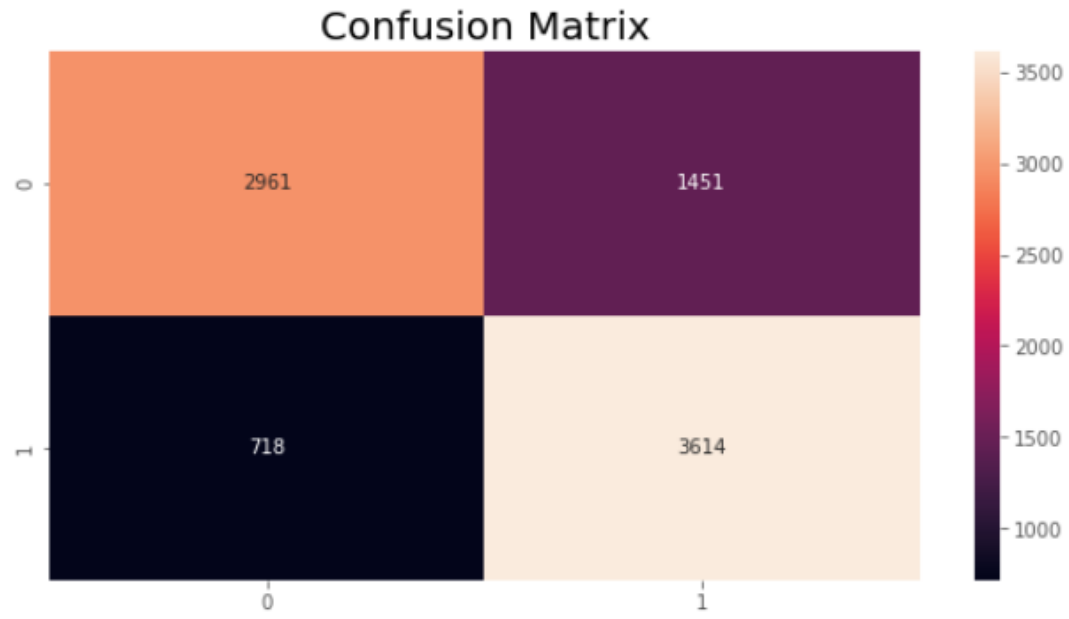
Performance of Gaussian naïve Bayes model

```
nb = GaussianNB()
result(x, y, 0.25, 42, nb)
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.67      0.73      4412
           1       0.71      0.83      0.77      4332

    accuracy                           0.75      8744
   macro avg       0.76      0.75      0.75      8744
weighted avg       0.76      0.75      0.75      8744

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

Accuracy Score:  0.7519441903019213
--------------------------------------------------------------------------------

recall Score:  0.8342566943674977
--------------------------------------------------------------------------------

F1 Score:  0.7691816537192722
--------------------------------------------------------------------------------

ROC-AUC Score:  0.7691816537192722
--------------------------------------------------------------------------------
```



Confusion Matrix:



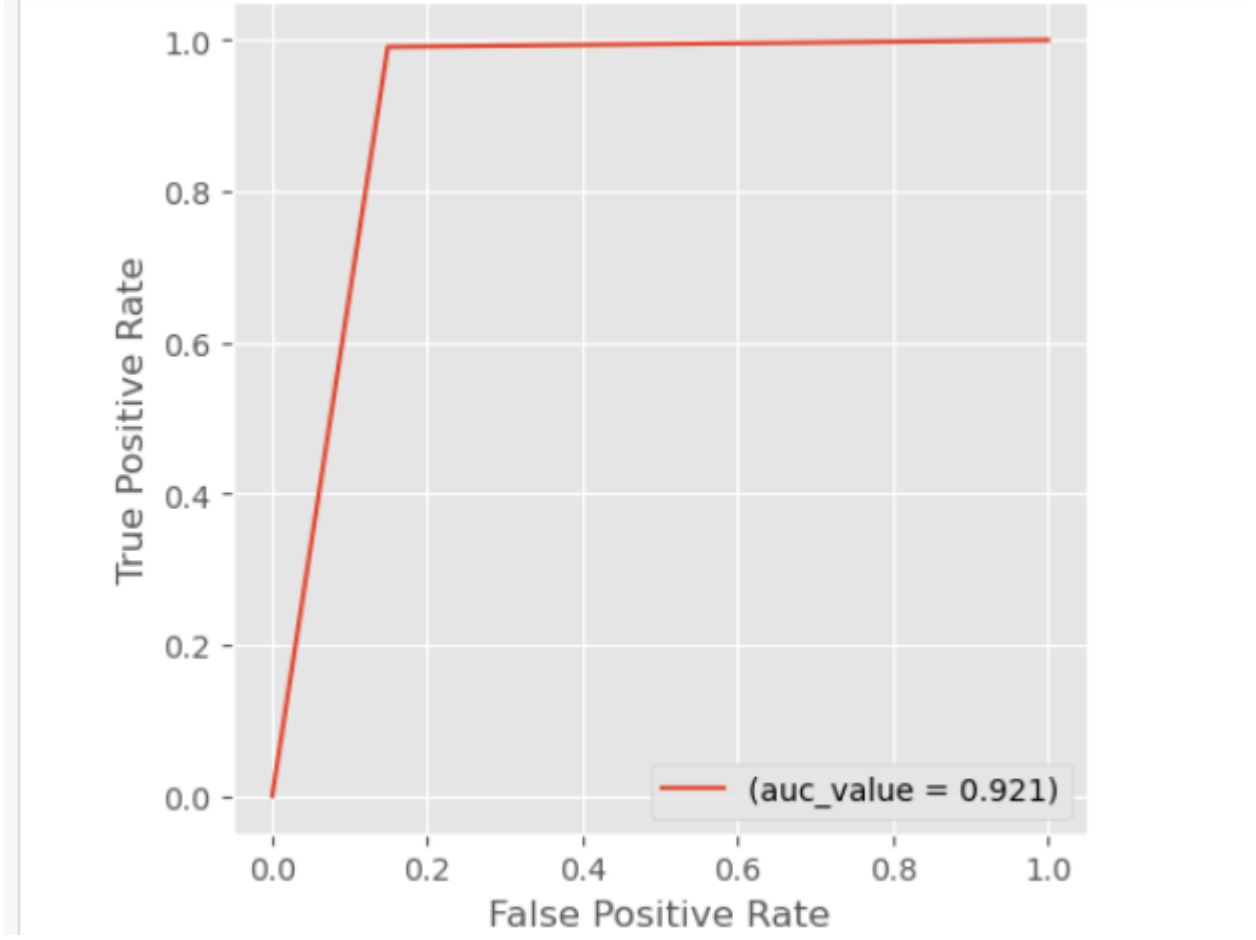
Gaussian Naive Bayes model has an Accuracy Score: 0.75 and F1 Score: 0.76

DECISION TREE CLASSIFIER

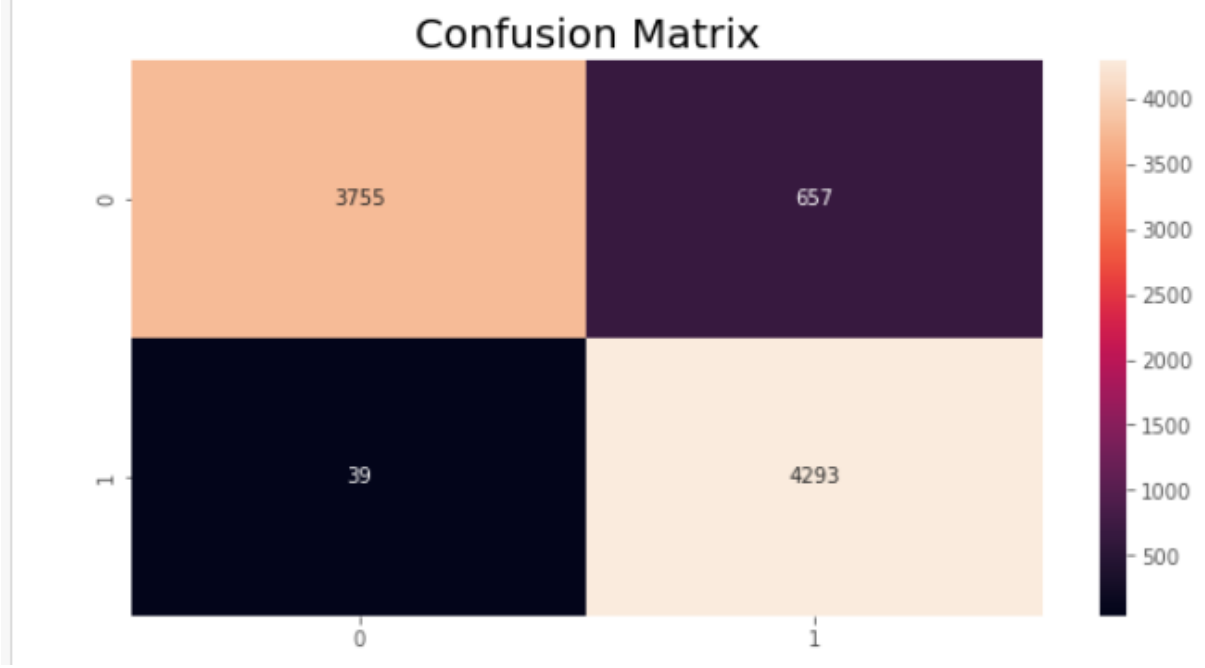
Performance of decision tree model

```
dt = DecisionTreeClassifier()
result(x, y, 0.25, 42, dt)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.85      0.92      4412
           1       0.87      0.99      0.93      4332

    accuracy                           0.92      8744
   macro avg       0.93      0.92      0.92      8744
weighted avg       0.93      0.92      0.92      8744

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

Accuracy Score:  0.9213174748398902
--------------------------------------------------------------------------

recall Score:  0.9909972299168975
--------------------------------------------------------------------------

F1 Score:  0.9258141039465171
--------------------------------------------------------------------------

ROC-AUC Score:  0.9258141039465171
--------------------------------------------------------------------------
```



Confusion Matrix:



DECISION TREE model has an Accuracy Score: 0.92 and F1 Score: 0.925

Now that we have accuracy and recall of the models then we can pick out the best model.

Classifier Comparison

```
classifier_list = ["Logistic Regression", "Random Forest", "KNN", "Naive Bayes", "Decision Tree"]
list_class = []
for i in range(0, len(classifier_list)):
  listclass = [classifier_list[i], accuracy_list[i],recall_list[i], f1_list[i], roc_auc_list[i]]
  list_class.append(listclass)

list_class
```

```
 0.9274931381518756,
 0.9953831948291783,
 0.9315186865413696,
 0.9281086418388865],
['KNN',
 0.8721408966148216,
 0.9789935364727609,
 0.8835416666666666,
 0.8731096422164348],
['Naive Bayes',
 0.7519441903019213,
 0.8342566943674977,
 0.7691816537192722,
 0.7526904505382366],
['Decision Tree',
 0.9213174748398902,
 0.9909972299168975,
 0.9258141039465171,
 0.9219492042603527]]
```

```
cc_table = pd.DataFrame(list_class, columns = ["Classifier", "Accuracy","recall", "F1 Score", "ROC-AUC Score"])
cc_table.sort_values(ascending = False, by = "Accuracy")
```

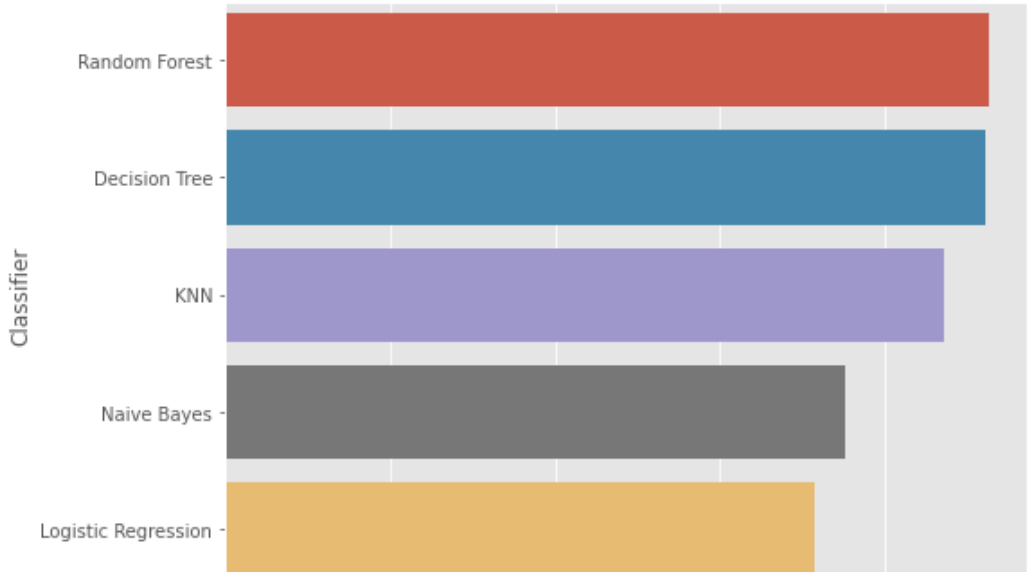
l27]:

	Classifier	Accuracy	recall	F1 Score	ROC-AUC Score
1	Random Forest	0.927493	0.995383	0.931519	0.928109
4	Decision Tree	0.921317	0.990997	0.925814	0.921949
2	KNN	0.872141	0.978994	0.883542	0.873110
3	Naive Bayes	0.751944	0.834257	0.769182	0.752690
0	Logistic Regression	0.714547	0.845337	0.745825	0.715733

Plotting the Comparison Table as a Bar Plot For Accuracy

In [82]:
```
plt.figure(figsize = (8,6))
sns.barplot(x = cc_table["Accuracy"]*100,
            y = cc_table["Classifier"],
            data = cc_table,
            order = cc_table.sort_values("Accuracy", ascending = False).Classifi
```

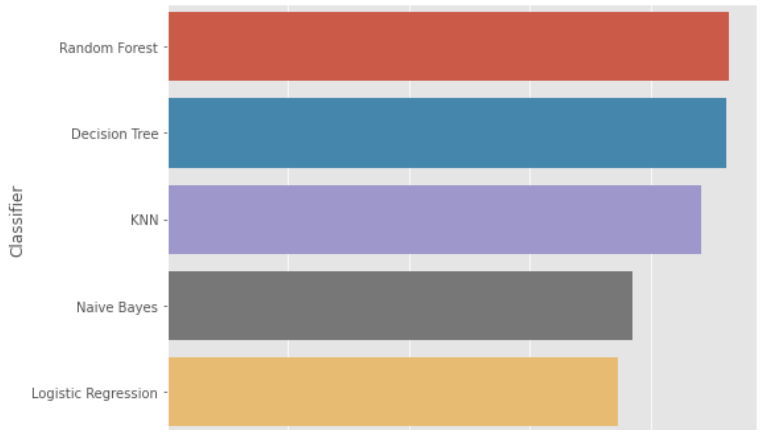
Out[82]: <AxesSubplot:xlabel='Accuracy', ylabel='Classifier'>



For F1 Score

```
In [83]: plt.figure(figsize = (8,6))
         sns.barplot(x = cc_table["F1 Score"]*100,
                     y = cc_table["Classifier"],
                     data = cc_table,
                     order = cc_table.sort_values("F1 Score", ascending = False).Classifier)
```

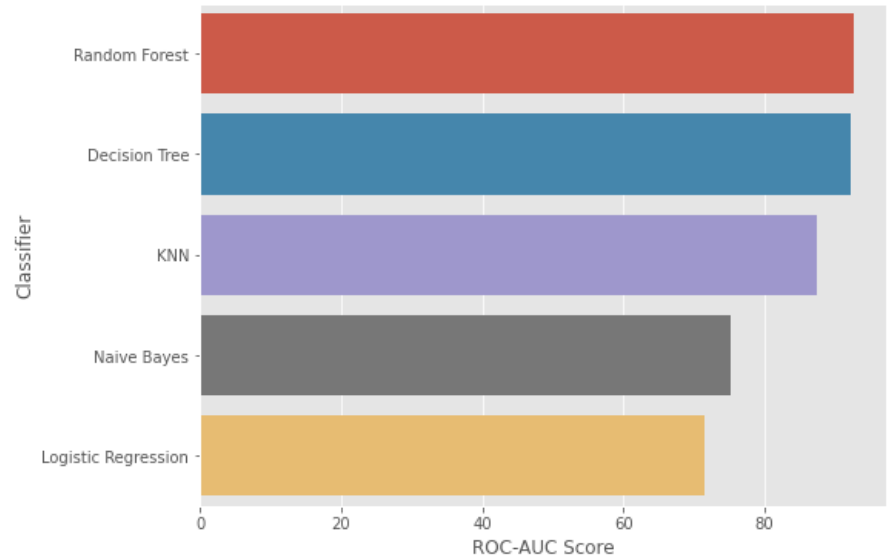
```
Out[83]: <AxesSubplot:xlabel='F1 Score', ylabel='Classifier'>
```



For ROC-AUC Score

```
In [84]: plt.figure(figsize = (8,6))
         sns.barplot(x = cc_table["ROC-AUC Score"]*100,
                     y = cc_table["Classifier"],
                     data = cc_table,
                     order = cc_table.sort_values("ROC-AUC Score", ascending = False).Classifier)
```

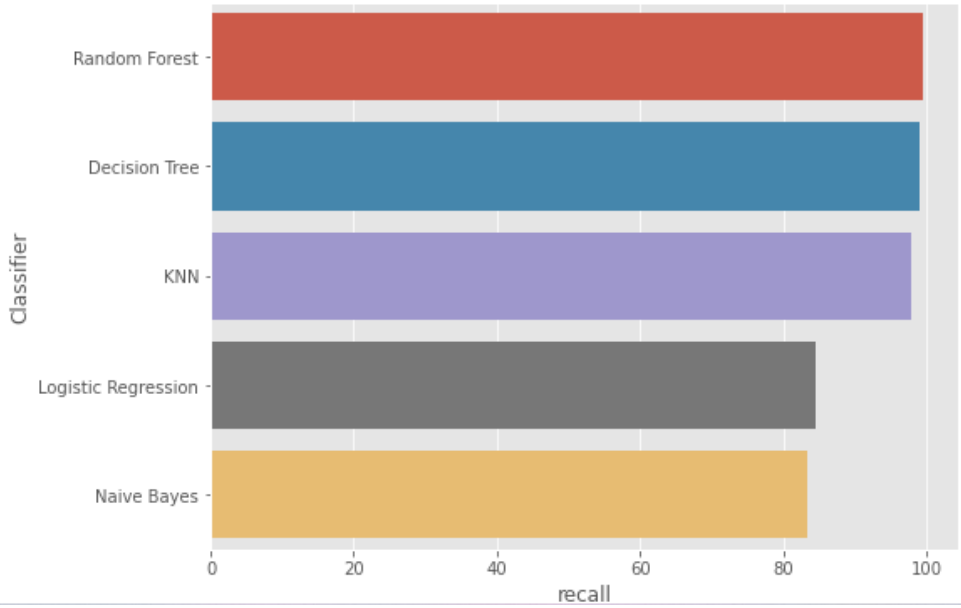
```
Out[84]: <AxesSubplot:xlabel='ROC-AUC Score', ylabel='Classifier'>
```



Recall

In [131]:
```
plt.figure(figsize = (8,6))
sns.barplot(x = cc_table["recall"]*100,
            y = cc_table["Classifier"],
            data = cc_table,
            order = cc_table.sort_values("recall", ascending = False).Classifier)
```

Out[131]: <AxesSubplot:xlabel='recall', ylabel='Classifier'>



Conclusion

In this report, we have studied different models that can be used to predict employee attrition. From the exploratory data analysis i.e. the critical process of performing investigations on data to discover patterns and so, we have found that the employees are resigning mainly of the less pay And So mostly, employees who leave tend to be young**,** with less time working in the company and at the beginning of their career in general**,** since most of these employees were working for less than 10 years in total. Here with the help of summary statistics, we can overcome that kind of attrition by hiring experienced candidates from graphical representations, we have found that people with master's degree have higher pay than the other degree employees.

We have trained and tested 5 models against our dataset. Considering the main goal is to identify employees that are more likely to leave the company, the recall score is the one that I'm focusing on here. I believe that in this case, false positives wouldn't be much more expensive than false negatives, so I'm more interested in the model that predicted the largest amount of employees to leave, which was the random forest. After comparing the evaluation metrics of each model we can conclude that the best classifier for our dataset is random forest. And decision tree classifier would be the next best option.

References

https://thesai.org/Downloads/Volume12No11/Paper_49-Machine_Learning_for_Predicting_Employee_Attrition.pdf

International Journal of Machine Learning and Computing, Vol. 11, No. 2, March 2021
Aseel Qutub, Asmaa Al-Mehmadi, Munirah Al-Hssan, Ruyan Aljohani, and Hanan S. Alghamdi

Computing, Information Systems & Development Informatics Vol. 4 No. 1 March, 2013
ANALYZING EMPLOYEE ATTRITION USING DECISION TREE ALGORITHMS

Wie-Chiang H., Ruey-Ming C. (2007); A Comparative Test of Two Employee Turnover Prediction Models. International Journal of Management; June 2007; Vol.24 No.2; pp. 216 – 229.

S. Das, A. Dey, A. Pal and N. Roy, "Applications of artificial intelligence in machine learning: Review and prospect," Int. J. Comp. Appl., vol. 115, pp. 31–41, January 2015.

A. Abu, R. Hamdan, R. and N.S. Sani, "Ensemble Learning for Multidimensional Poverty Classification," Sains Malaysiana," vol. 49(2), pp.447-459 2020.

Nor Samsiah Sani, Abdul Hadi Abd Rahman, Afzan Adam, Israa Shlash and Mohd Aliff, "Ensemble Learning for Rainfall Prediction" International Journal of Advanced Computer Science and Applications, vol. 11(11), pp. 153-162, 2020.