IBM HR ATTRITION RATE ANALYTICS

Attrition is an issue that every company deals with. It impacts all the businesses, irrespective of geography, industry, and size of the company. Attrition in human resources refers to the gradual loss of employee's overtime. A major problem in high employee attrition is its cost to an organization. Job postings, hiring processes, paperwork, and new hire training are some of the common expenses of losing employees and replacing them.

Every year a lot of companies hire several employees. The companies that invest it's time and money in training employees, there are also conducting training programs within the companies for their existing employees. HR analytics, aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve these processes.

Attrition Rate (%) =
(Number of separations/ Average Number of employees) * 100

Some of the factors contribute to the attrition rate:

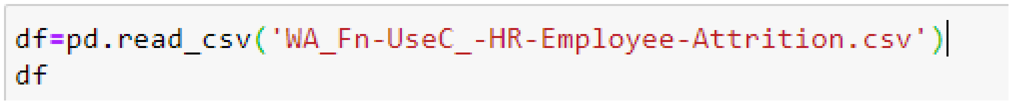
- Improper work-life balance
- Better job opportunities
- Commendable pay hike
- Lack of growth or work recognition
- Absence of proper working environment and infrastructure
- The untimely death of employees or retirement
- Unhealthy relations with managers or other members of the organization.

With advances in machine learning and data science, it's possible to predict the employee attrition. The IBM HR Attrition Case Study is a dataset which aims to identify which contributions that might be influential in predicting which employee might leave the firm and who may not.

Table of content:

1.Data Analysis:

The Data set is published by IBM Human Resource Department is available at Kaggle. To get the analysis of this data set let's look into the basic features available in the dataset and also the information present in that dataset. Here, once we upload the data into our Python script or any python notebook as shown in the figure below , we get to know:



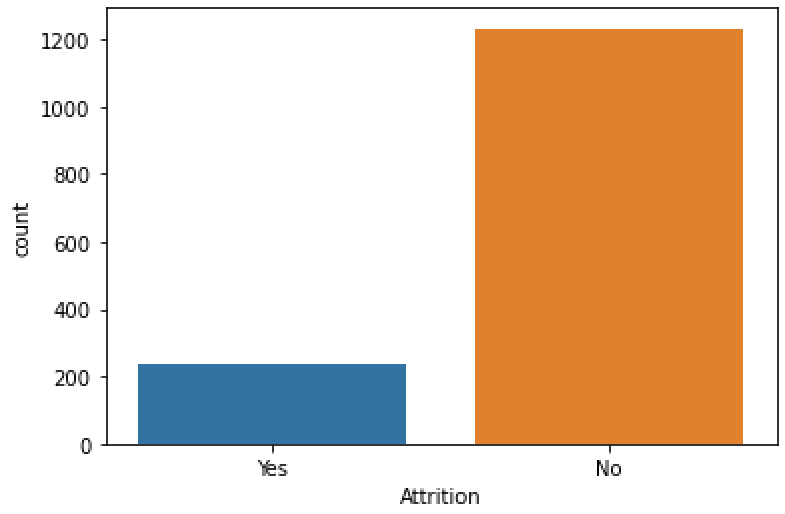
```
df=pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
df
```

The shape of the data as (1470, 35), which means there are 1470 rows
and 35 columns, and all the columns contains non null values which
mean there are no columns of null values. we are having columns lik
e age, business travel, daily rate, Department, Distance From Home,
Education, Education-Field, Employee Number, ..etc of an organisat
ion employee details. We need to predict how, does attrition effect th
e company.

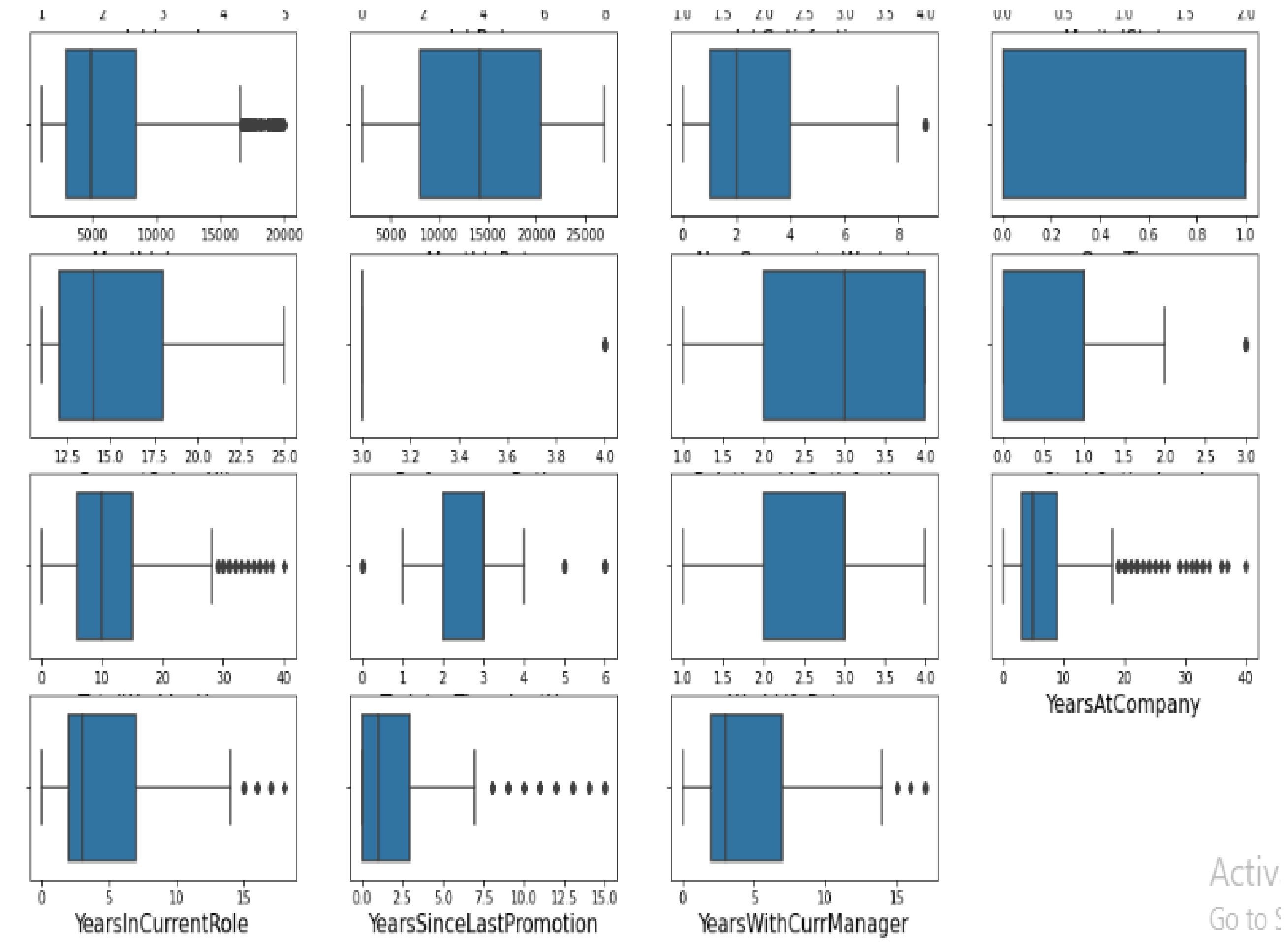
The target variable is Attrition and other columns are referred as feature
with which we can analyse the data. The 'attrition' variable is having a bi
nary categorical data and hence the model which we can build is of classi
fication type. This target has value count as:

No 1233
Yes 237
And datatype: int64.

```
sns.countplot(x='Attrition',data=df)
plt.show()
```



The above graph is a count plot which shows the attrition variable data. It shows, around 230 'Yes' data and around 1200 'No' data. Which clearl y define the target variable exist with imbalance of data which probably l eads to over fitting of data while model building. Since, it has many colu mns the minimum, maximum, standard deviation and mean of each colu mn is difficult to, measure. Checking of outliers is possible by visualising the data using boxplot. In the below graph there exists some outliers whi ch can be processed using techniques.

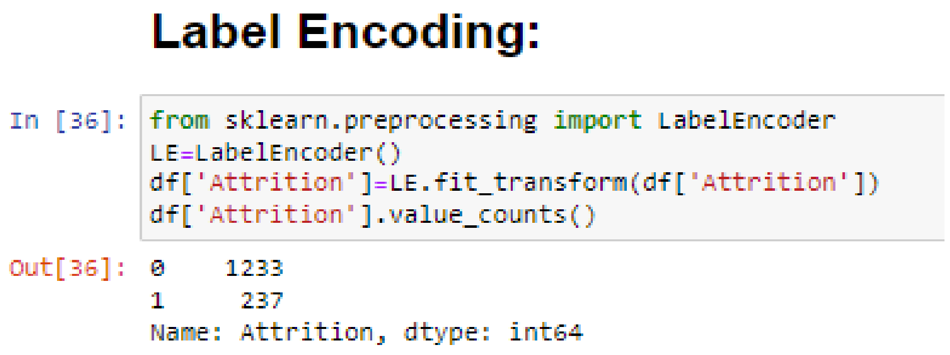


2.EDA Conclusion:

The given Data set consists of more than 30 columns which is having Attrition as target variable and all the other columns are employee detail s with which we can predict the data. Here in this target variable there is some imbalance of data exists. The feature variable consists of some outli ers and skewness which need to be further processed. All the columns ar e existing with 0 null count. Based on the type of the data, we can Predict the attrition using any binary Classification Model.

3.Pre-Processing Pipeline:

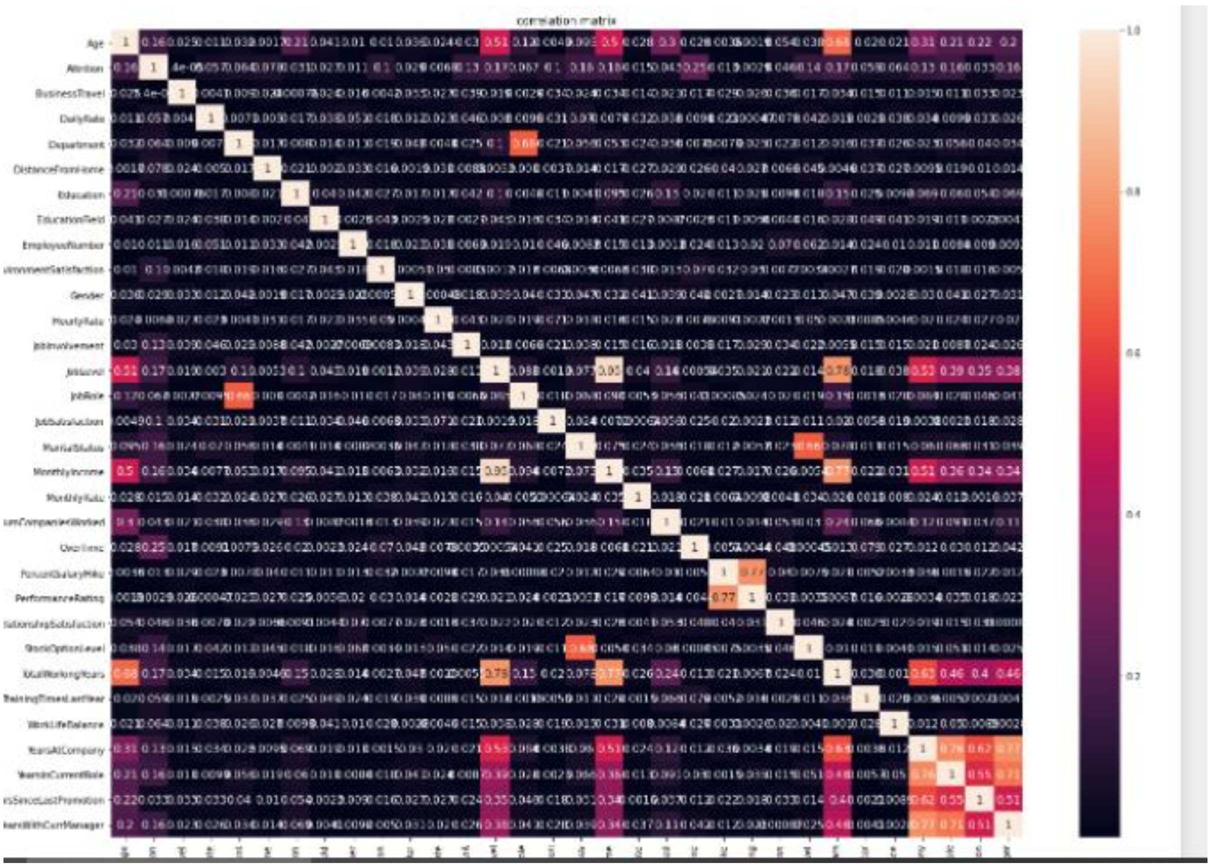
Firstly we need to encode the data for further processing the data. We use label encoding for the Attrition variable and also we use ordinal encoder for object type columns like department, gender ..etc as shown below:

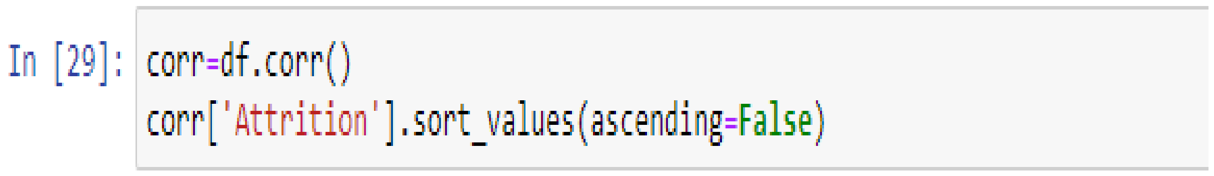

Label Encoding:

```
In [36]: from sklearn.preprocessing import LabelEncoder
         LE=LabelEncoder()
         df['Attrition']=LE.fit_transform(df['Attrition'])
         df['Attrition'].value_counts()

Out[36]: 0    1233
         1     237
         Name: Attrition, dtype: int64
```

Now after looking into the columns like Employee Count, Standard Hours, Over18 are containing only one value through out the rows which is not necessary, so columns above are dropped of the dataset.

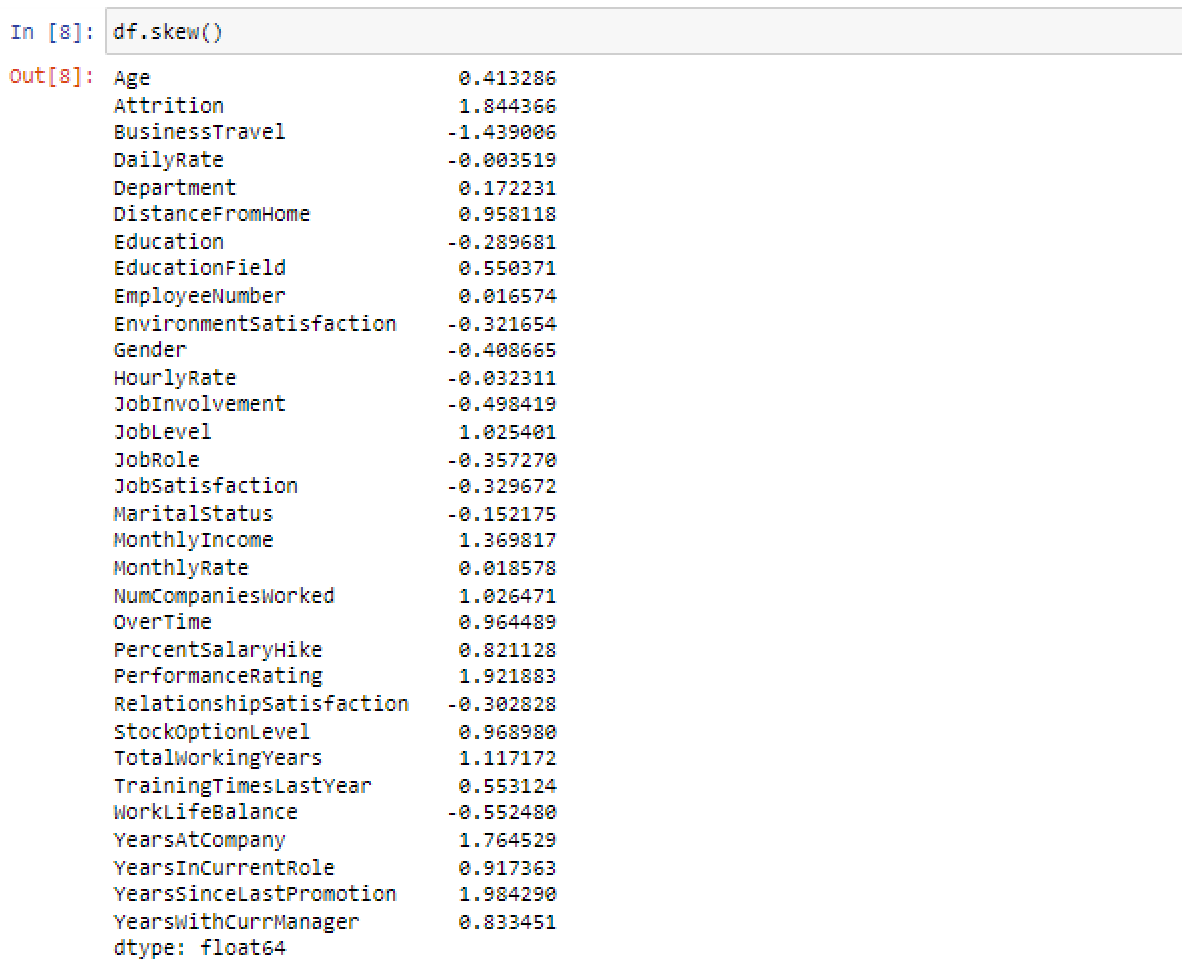
Here,let’s check the correlation of the ‘Attrtion’ column with respect to all the other features and also the correlation between all the columns exists using the heat map and descriptively also.But in the heat map hown below there is no cear information so we use corr() method to find the correlation.





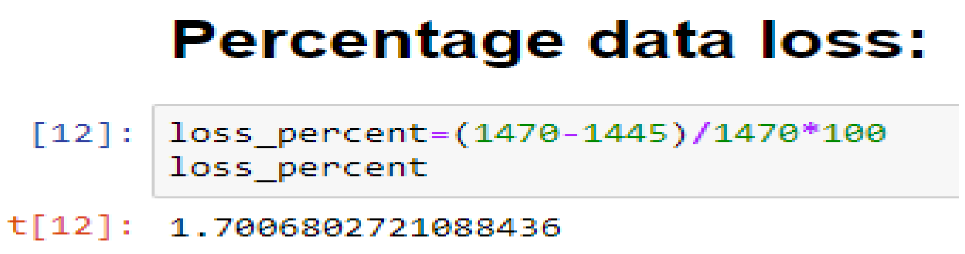
```
In [29]: corr=df.corr()
         corr['Attrition'].sort_values(ascending=False)
```

Skewness of the data can be found using skew( ) method and the Skewness threshold is taken as +/-0.5.
Columns which are having skewness which effects the model are: Distance-From-Home, Job-Level, Monthly Income, No Companies Worked, Performance Rating , Total Working Years , Years At Company, Years Since Last Promotion which are dropped for further processing.



```
In [8]: df.skew()

Out[8]: Age                          0.413286
        Attrition                    1.844366
        BusinessTravel              -1.439006
        DailyRate                   -0.003519
        Department                   0.172231
        DistanceFromHome             0.958118
        Education                   -0.289681
        EducationField               0.550371
        EmployeeNumber               0.016574
        EnvironmentSatisfaction     -0.321654
        Gender                      -0.408665
        HourlyRate                  -0.032311
        JobInvolvement              -0.498419
        JobLevel                     1.025401
        JobRole                     -0.357270
        JobSatisfaction             -0.329672
        MaritalStatus               -0.152175
        MonthlyIncome                1.369817
        MonthlyRate                  0.018578
        NumCompaniesWorked           1.026471
        OverTime                     0.964489
        PercentSalaryHike            0.821128
        PerformanceRating            1.921883
        RelationshipSatisfaction    -0.302828
        StockOptionLevel             0.968980
        TotalWorkingYears            1.117172
        TrainingTimesLastYear        0.553124
        WorkLifeBalance             -0.552480
        YearsAtCompany               1.764529
        YearsInCurrentRole           0.917363
        YearsSinceLastPromotion      1.984290
        YearsWithCurrManager         0.833451
        dtype: float64
```

Now using Zscore technique we can minimise the effect of outliers and the percentage loss of data after removing outliers is considerable



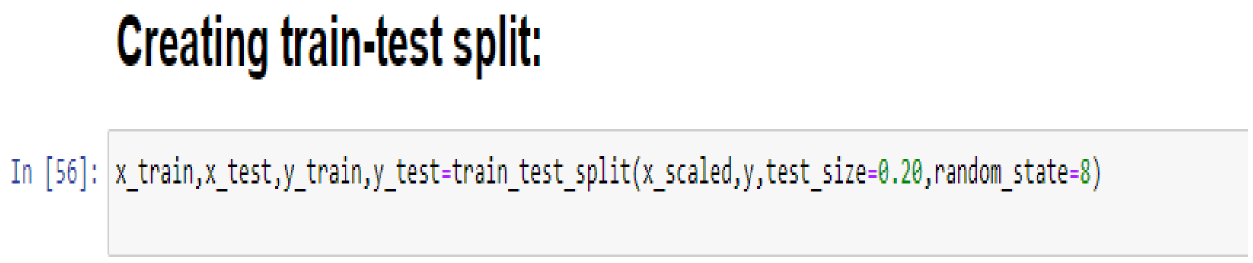
Percentage data loss:

```
[12]: loss_percent=(1470-1445)/1470*100
      loss_percent
t[12]: 1.7006802721088436
```

Now that the Data is pre-processed and ready to build models. We can use this data for model building using classification techniques.

4.Building Machine Learning Models:

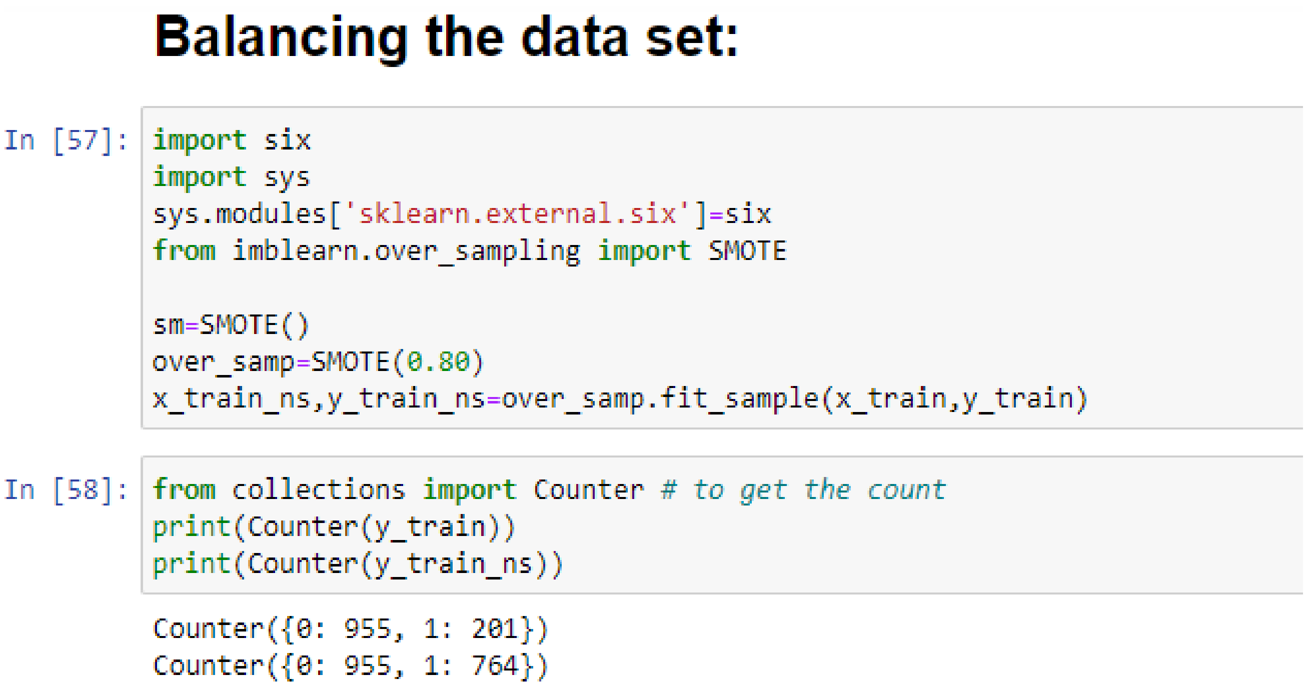
First step of this process is to split the target variable and all the other feature into x and y and scaling of x. Next finding the best random state for modelling, here the output for the best random state is '8'.Using this we use Train-Test Split to divide the training and testing data.



Creating train-test split:

```
In [56]: x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.20,random_state=8)
```

Once the training and testing data is divided, we use 'SMOTE' technique to remove overfitting and balancing the data. Since the yes count is less than no count of the target variable we used oversampling technique in order to modify and balance the data using SMOTE technique.



Balancing the data set:

```
In [57]: import six
         import sys
         sys.modules['sklearn.external.six']=six
         from imblearn.over_sampling import SMOTE

         sm=SMOTE()
         over_samp=SMOTE(0.80)
         x_train_ns,y_train_ns=over_samp.fit_sample(x_train,y_train)
```

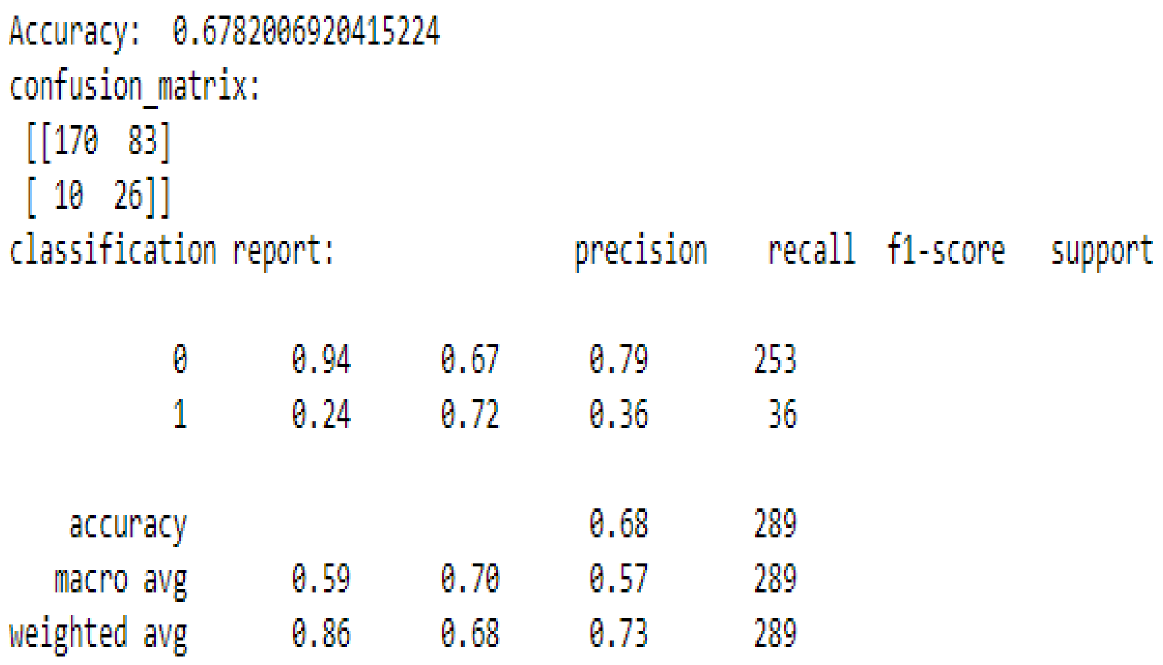
```
In [58]: from collections import Counter # to get the count
         print(Counter(y_train))
         print(Counter(y_train_ns))
```

```
Counter({0: 955, 1: 201})
Counter({0: 955, 1: 764})
```

As shown in the above figure, the target variable count of 0 and 1 in training data is 955 and 201 respectively , which is showing inappropriate proportions of 0 and 1.after sampling the count of 0 and 1 of target Attrition variable is 955 and 764 respectively, which is considerably balanced we can use this data for further modelling.

Model1: K-Neighbors Classification Model:

In this model, firstly we need to import the KNeighborsClassifier and we need to fit the (x_train_ns & y_train_ns) training data and then predict the target variable test data using the test data available.

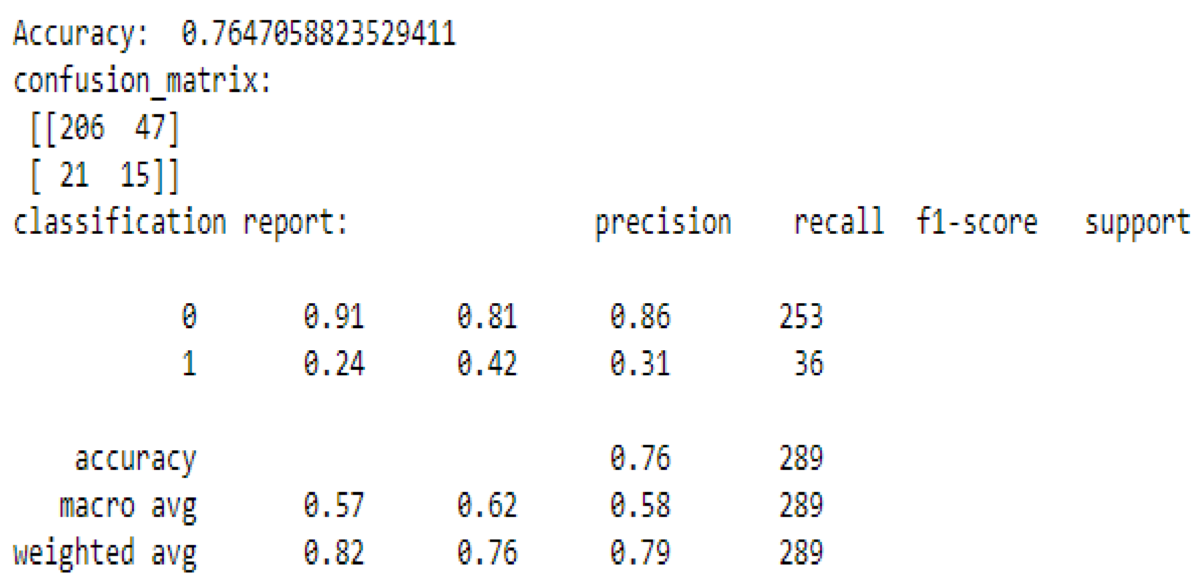


```
Accuracy:  0.6782006920415224
confusion_matrix:
 [[170  83]
 [ 10  26]]
classification report:               precision    recall  f1-score   support

           0       0.94      0.67      0.79       253
           1       0.24      0.72      0.36        36

    accuracy                           0.68       289
   macro avg       0.59      0.70      0.57       289
weighted avg       0.86      0.68      0.73       289
```

Here in this model we get the accuracy around 68% and in the confusion matrix the True Positive rate (TP)of the model is given as 170 and False Positive rate (FP)as 83 ,False negative rate(FN) is given as 10 where as True negative(TN) rate is 26.Since the FP <TP and FN<TN the model is performing well .In the classification report , the precision is given as 79% and recall is around 250.

Model2: Decision-Tree Classification Model:

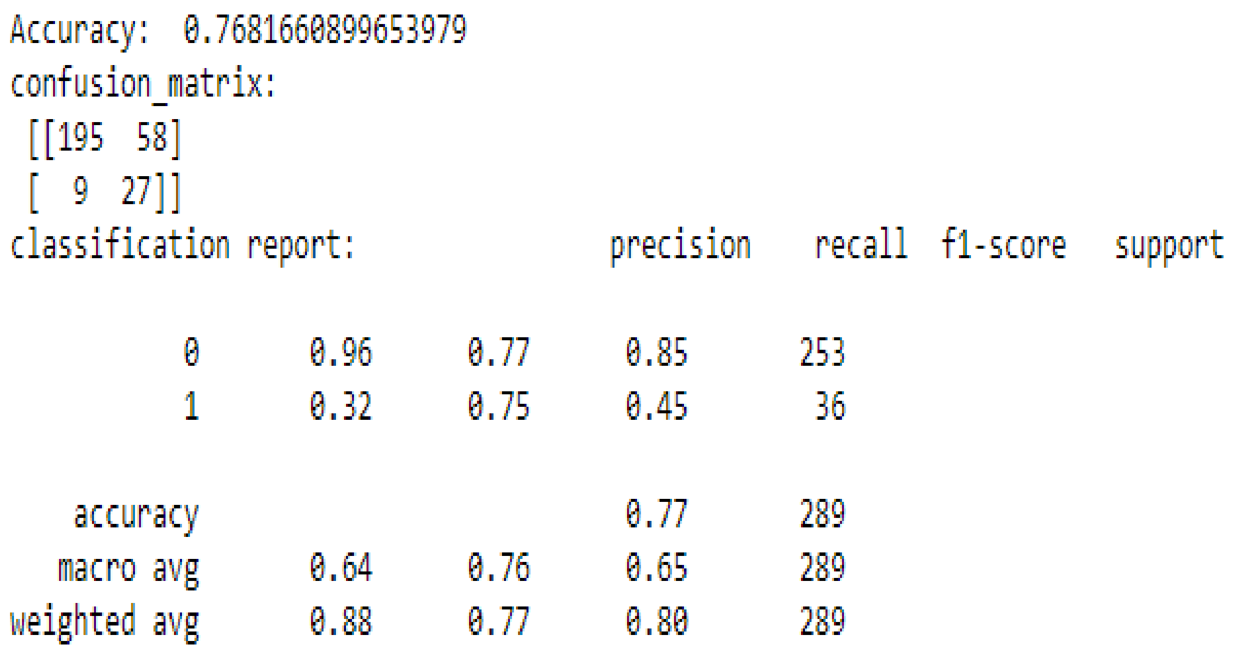
In this model, firstly we need to import the DecisionTreeClassifier and we need to fit the (x_train_ns & y_train_ns) training data and then predict the target variable using the test data available.


```
Accuracy:  0.7647058823529411
confusion_matrix:
 [[206  47]
 [ 21  15]]
classification report:               precision    recall  f1-score   support

           0       0.91      0.81      0.86       253
           1       0.24      0.42      0.31        36

    accuracy                           0.76       289
   macro avg       0.57      0.62      0.58       289
weighted avg       0.82      0.76      0.79       289
```

Here in this model we get the accuracy around 76% and in the confusion matrix the True Positive rate (TP)of the model is given as 206 and False Positive rate (FP)as 47, False negative rate(FN) is given as 21 where as True negative(TN) rate is 15.Since the FP <TP and FN>TN the model is performing well .In the classification report , the precision is given as 86% and recall is around 250.

Model3: Logistic Regression Model:

In this model, firstly we need to import the LogisticRegression and we need to fit the (x_train_ns & y_train_ns) training data and then predict the target variable using the test data available.

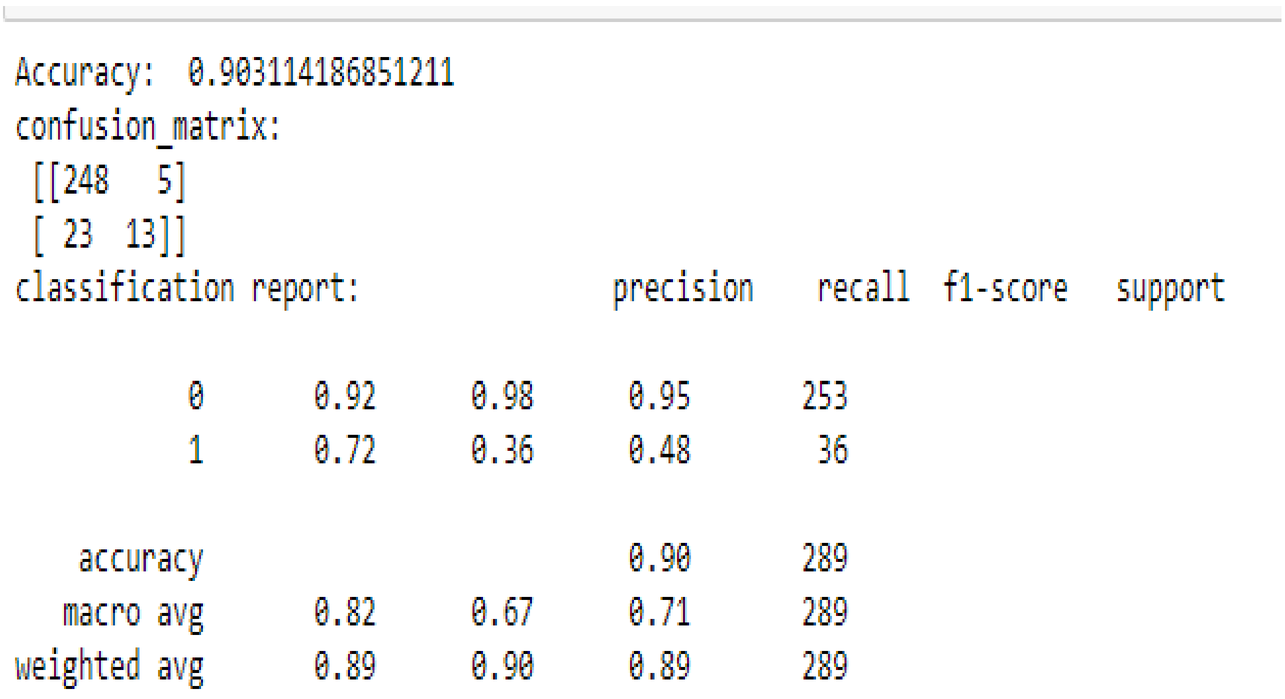


```
Accuracy:  0.7681660899653979
confusion_matrix:
 [[195  58]
 [  9  27]]
classification report:               precision    recall  f1-score   support

           0       0.96      0.77      0.85       253
           1       0.32      0.75      0.45        36

    accuracy                           0.77       289
   macro avg       0.64      0.76      0.65       289
weighted avg       0.88      0.77      0.80       289
```

Here in this model we get the accuracy around 77% and in the confusion matrix the True Positive rate (TP)of the model is given as 195 and False Positive rate (FP)as 58, False negative rate (FN) is given as 9 where as True negative(TN) rate is 27.Since the FP <TP and FN<TN the model is performing well .In the classification report , the precision is given as 85% and recall is around 250.

Model4: Random Forest Classification Model:

In this model, firstly we need to import the RandomForestClassifier and we need to fit the (x_train_ns & y_train_ns) training data and then predict the target variable using the test data available.



```
Accuracy:  0.903114186851211
confusion_matrix:
 [[248   5]
 [ 23  13]]
classification report:               precision    recall  f1-score   support

           0       0.92      0.98      0.95       253
           1       0.72      0.36      0.48        36

    accuracy                           0.90       289
   macro avg       0.82      0.67      0.71       289
weighted avg       0.89      0.90      0.89       289
```

Here in this model we get the accuracy around 90% and in the confusion matrix the True Positive rate (TP)of the model is given as 248 and False Positive rate (FP)as 5, False negative rate (FN) is given as 23 where as True negative (TN) rate is 13. This model is performing well, in the classification report , the precision is given as 95% and recall is around 250.

K-Neighbors (KNN)Classification Model, Decision-Tree Classification Model, Logistic regression, and Random Forest Classifier are producing good accuracy. Based on the above analysis we can proceed the modelling using Random forest classification model which is having good accuracy around 90% ,precision of 95% and recall 253.

Now we will check cross validation score as well as overfitting if exists.

The cross validation score of the above 4 models at cv=5 is taken.

cross validation score of KNN model is: 0.8380622837370242, cross validation score of Decision tree model is: 0.7598615916955017,

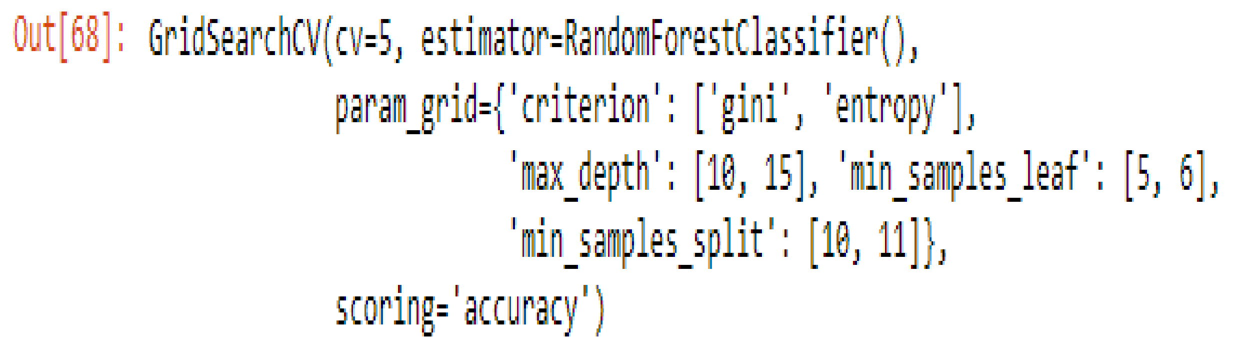
cross validation score of Logistic regression model is: 0.8574394463667
82

cross validation score of Random Forest classifier model is: 0.84844290
65743946,

The best model is selected based on both model accuracy and its cross-validation score. If the difference between 'cv' and model 'accuracy' is less, then that model is treated as best model. And so, by doing this analysis we considered Random Forest classification model with accuracy 90.31% and cv score as 84.84% is performing well.

Hyper-Parameter Tuning:

Hyper parameter tuning is a technique used to get the maximum best possible accuracy of the desired model. Here, in this model we tune the model to get the best parameters of the listed one's. In this model we opted the parameters as 'criterion', 'maximum depth', 'minimum sample split' and 'min sample leaf' with cv=5 and estimator as Random Forest Classifier( ).

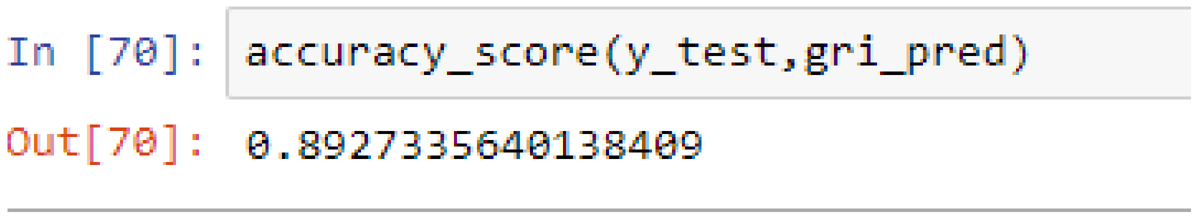


```
Out[68]: GridSearchCV(cv=5, estimator=RandomForestClassifier(),
                      param_grid={'criterion': ['gini', 'entropy'],
                                  'max_depth': [10, 15], 'min_samples_leaf': [5, 6],
                                  'min_samples_split': [10, 11]},
                      scoring='accuracy')
```

We used best parameters to predict the model. The model is predicted
Using the following parameters:

'criterion': 'entropy', 'max_depth': 15, 'min_samples_leaf': 5, 'min_samp les_split': 10.

Now for this trained and hyper parameter tunned model, the accuracy fo r the predicted model is given as around 89% and the figure is shown bel ow.

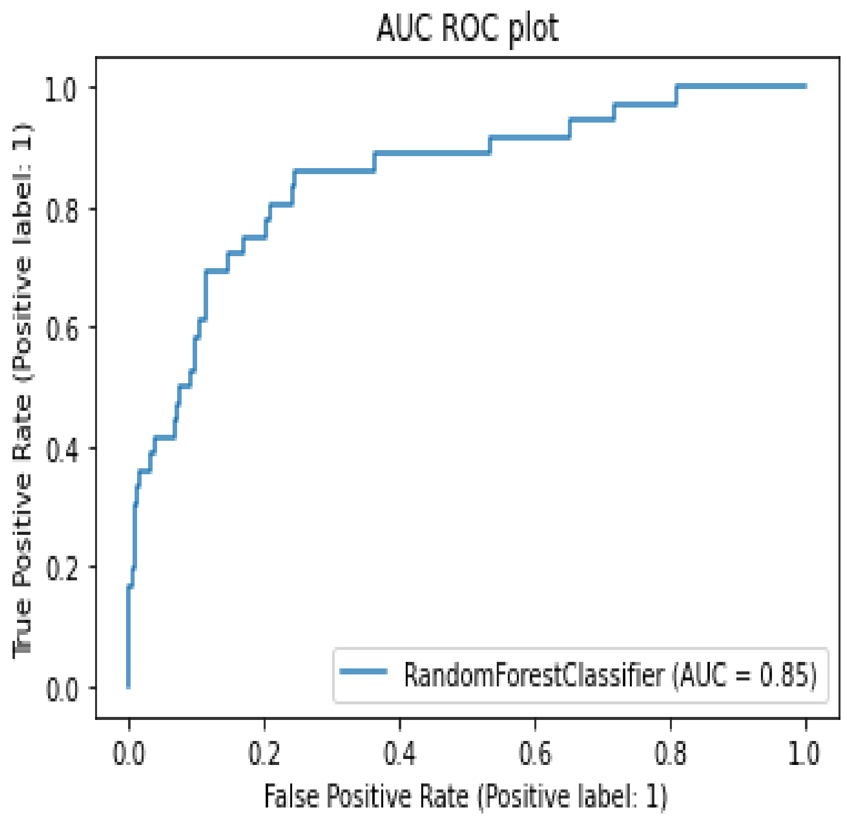


```
In [70]: accuracy_score(y_test,gri_pred)

Out[70]: 0.8927335640138409
```

ROC AUC PLOT:

ROC AUC PLOT is a mat plot function plot where the plot is drawn between the False positive rate and true positive rate where for plot_roc_curve we give the best parameters, tested data i.e., X_ test and Y_test so that we get the graph with AUC Score of 85% which is good.

```
Out[71]: <function matplotlib.pyplot.show(close=None, block=None)>
```



Final accuracy is 89% and AUC score is 85% for the Random Forest Classification model which is decently good.

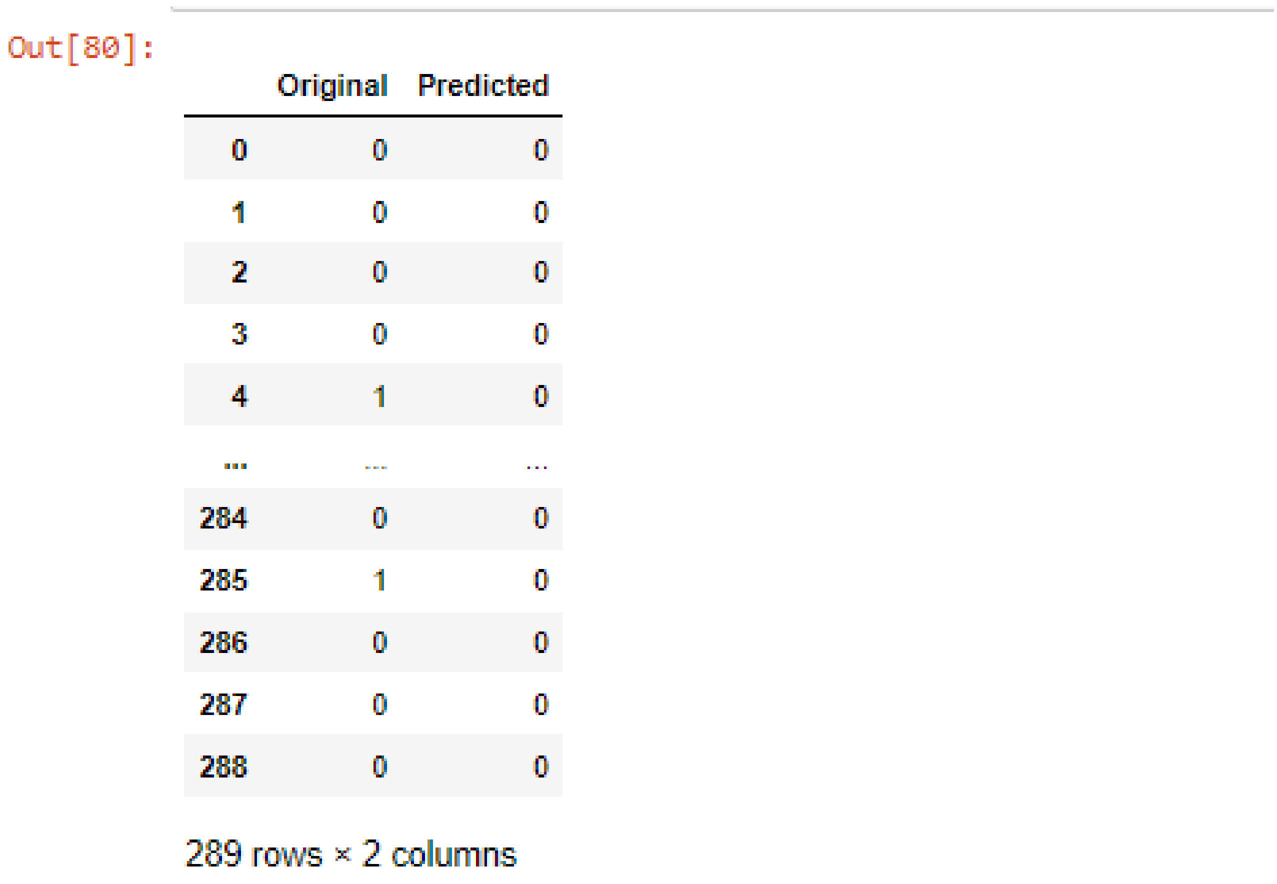
Now that the model is build, we can save this model in pickle format and can load and test the data whenever we need.

Saving the model:

```
In [78]: import pickle
         filename='HR_analytics.pkl'
         pickle.dump(gri.best_estimator_,open(filename,'wb'))
```

5. Concluding& Remarks:

The difference between the original data and the predicted data is shown by creating a data frame below:



Out[80]:

	Original	Predicted
0	0	0
1	0	0
2	0	0
3	0	0
4	1	0
...	...	...
284	0	0
285	1	0
286	0	0
287	0	0
288	0	0

289 rows × 2 columns

Here the model having the accuracy around 89% which is decently good score to get even more best accuracy we should try other classification models such as XG Boosting classifier, Supply Vector Machine, Gradient boosted classifier, and bagging classifier and many more to check and get the best model with better accuracy. The next aspect that can be modified is tunning parameters in hyper parameter tunning technique, we can modify by giving varied lists in each parameter that can really help in increasing accuracy and getting the best output.

Likewise, ML models can be used to solve complex business problems across Industry verticals/Business functions such as Banking, Insurance, Aviation, Public sector, R&D, Sales and Marketing, Food & Beverages and so on, would have been almost impossible to even think as recently as a decade ago, and yet the pace at which scientists and researchers are advancing is nothing short of amazing. In the coming future, we'll see that machine learning can recognize, alter, and improve upon their own internal architecture with minimal human intervention.