UDACITY

Machine Learning Engineer Nanodegree 2020

Capstone Project

Predicting Employee Attrition in the Dawn of Recession

Chetan S Rane

October 2020

1. Definition

- **Project Overview**
  As the COVID-19 keeps unleashing its havoc, the world continues to get pushed into the crisis of the great economic recession, more and more companies start to cut down their underperforming employees. Companies firing hundreds and thousands of Employees is a typical headline today. Cutting down employees or reducing an employee salary is a tough decision to take. It needs to be taken with utmost care as imprecision in the identification of employees whose performance is attriting may lead to sabotaging of both employees' career and the company's reputation in the market.

  The goal of this project is to predict the employee attrition by the given data about his/her past history.

- **Problem statement**
  The goal is to predict the employee attrition by the given data about his/her past history. From the given data we have to select best features using statistical and visual analysis which help in better classification of a given record. Experimenting feature engineering can be very useful in strong feature creation which help in better classification of employee attrition.

  The challenging part about dataset is we have a good variety of categorical features. Also considering the number of feature and records we have, one more challenge here is of curse of dimensionality.

- **Metrics**
  The evaluation metric for this competition is Confusion Matrix, Classification report and AUC score under ROC.

  Confusion Matrix:

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known.

Let's now define the most basic terms, which are whole numbers (not rates):

- **true positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
- **true negatives (TN):** We predicted no, and they don't have the disease.
- **false positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **false negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Classification Report:
The classification report visualizer displays the precision, recall, F1, and support scores for the model.

AUC-ROC curve:

AUC–ROC curve is the model selection metric for bi–multi class classification problem. ROC is a probability curve for different classes. ROC tells us how good the model is for distinguishing the given classes, in terms of the predicted probability.

A typical ROC curve has False Positive Rate (FPR) on the X-axis and True Positive Rate (TPR) on the Y-axis.

The bigger the area covered, the better the machine learning models is at distinguishing the given classes. Ideal value for AUC is 1.

When to Use ROC vs. Precision-Recall Curves?

Generally, the use of ROC curves and precision-recall curves are as follows:

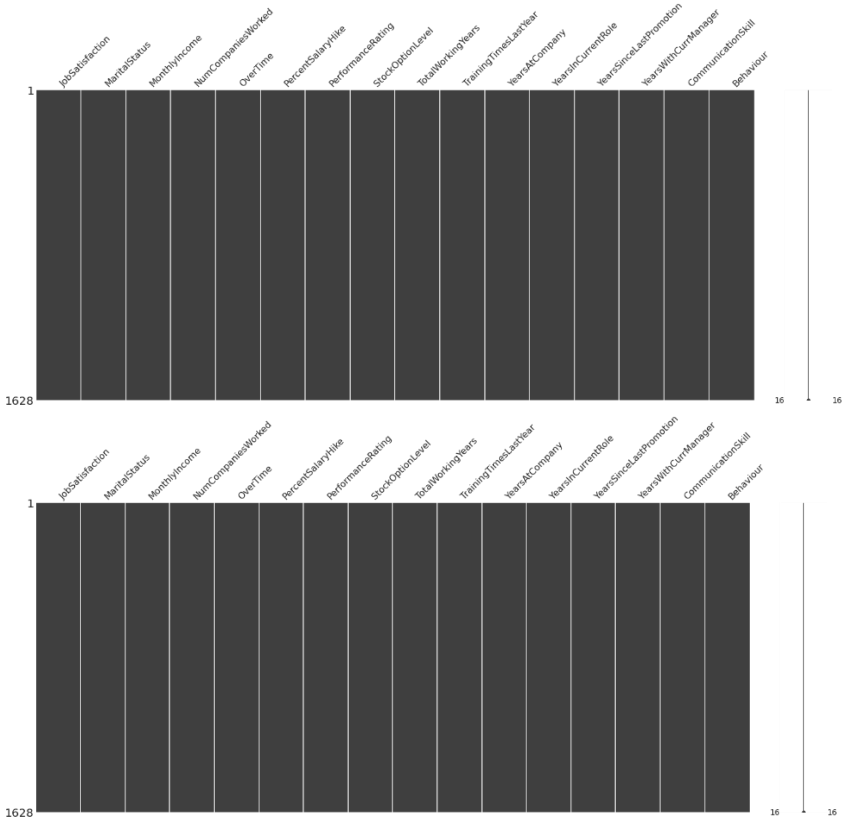
- ROC curves should be used when there are roughly equal numbers of observations for each class.

- Precision-Recall curves should be used when there is a moderate to large class imbalance.

In our case we need to predict the attrition of employee which is our positive class. So, our main goal will be to reduce False negatives i.e. actual is attrition but predicted as no attrition.

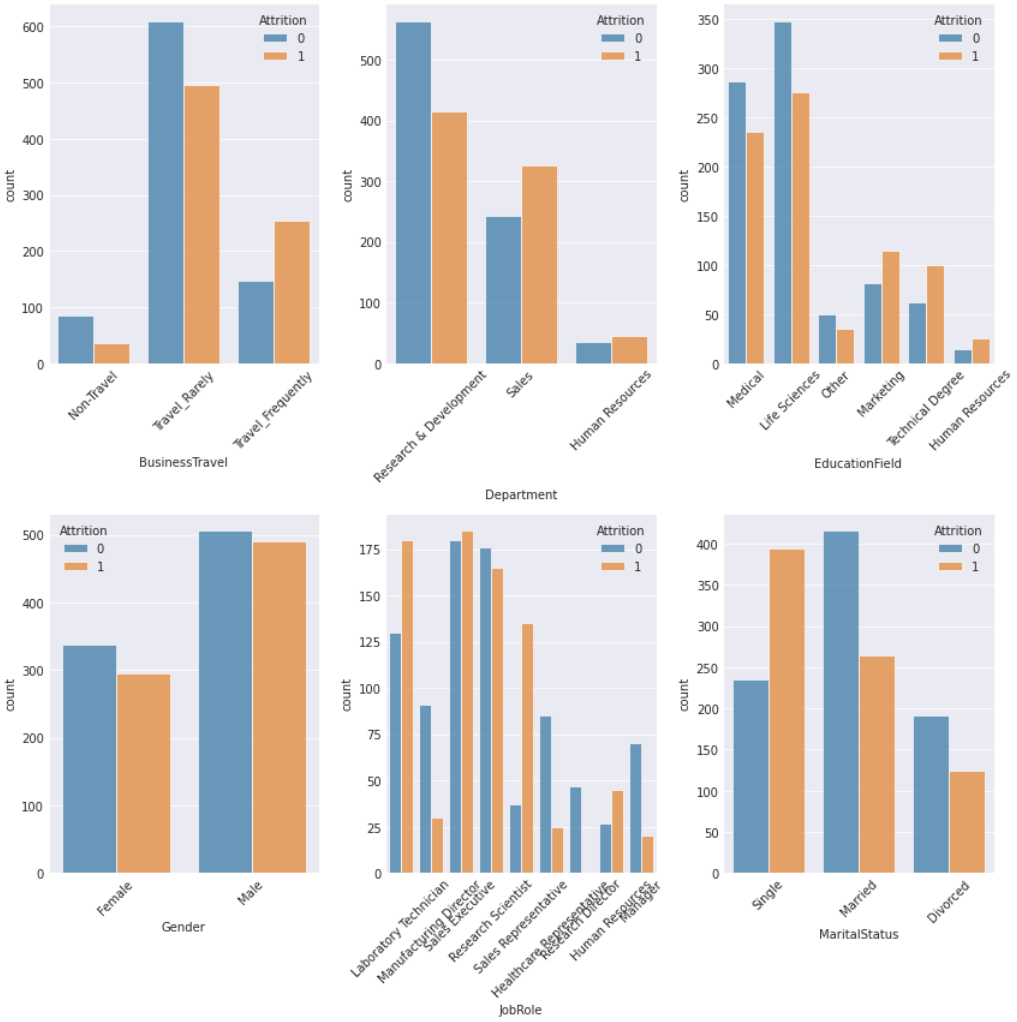
2. Analysis

- **Data Exploration and Visualization**
  - Employee Attrition dataset contains total 1628 records and 29 columns.
  - Column ID is unique for each and every row.
  - Columns which are close to normal distribution and have no skewness: Age, EnvironmentSatisfaction, JobInvolvement, JobSatisfaction, NumCompaniesWorked, PerformanceRating,TrainingTimesLastYear,CommunicationSkill.
  - No missing values in data. Confirmed using **missingno** package

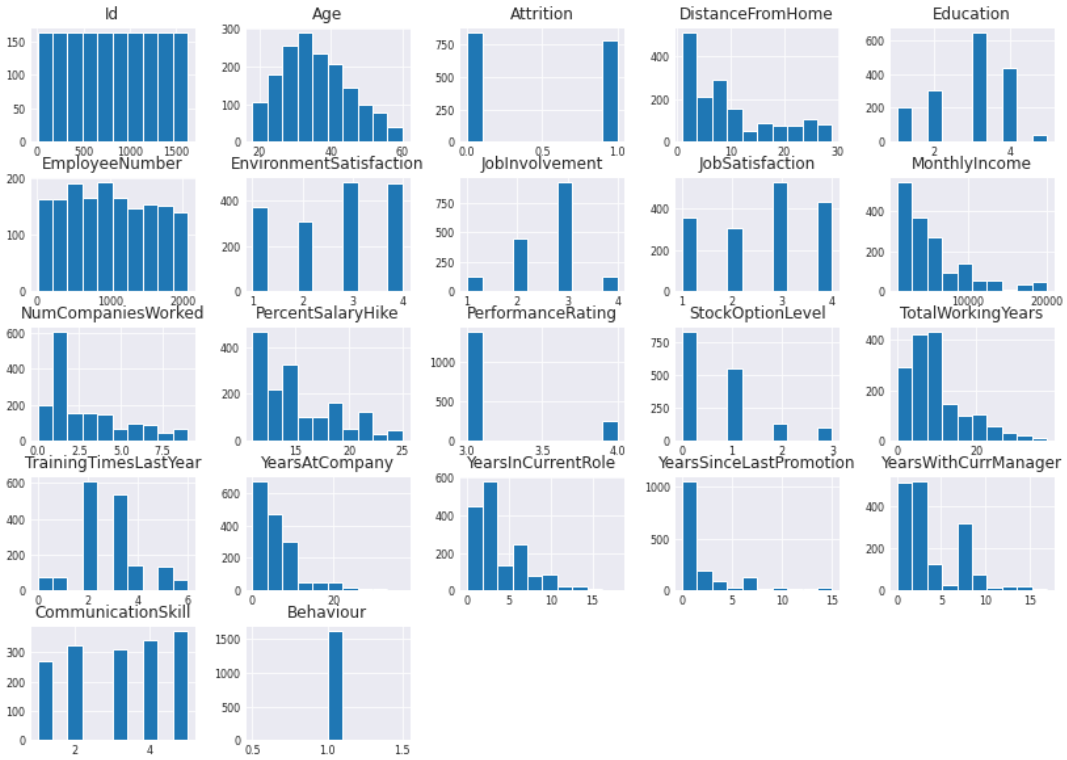


  - Dataset contains 628 duplicate records.

- Each categorical column has different proportion of categorical values.

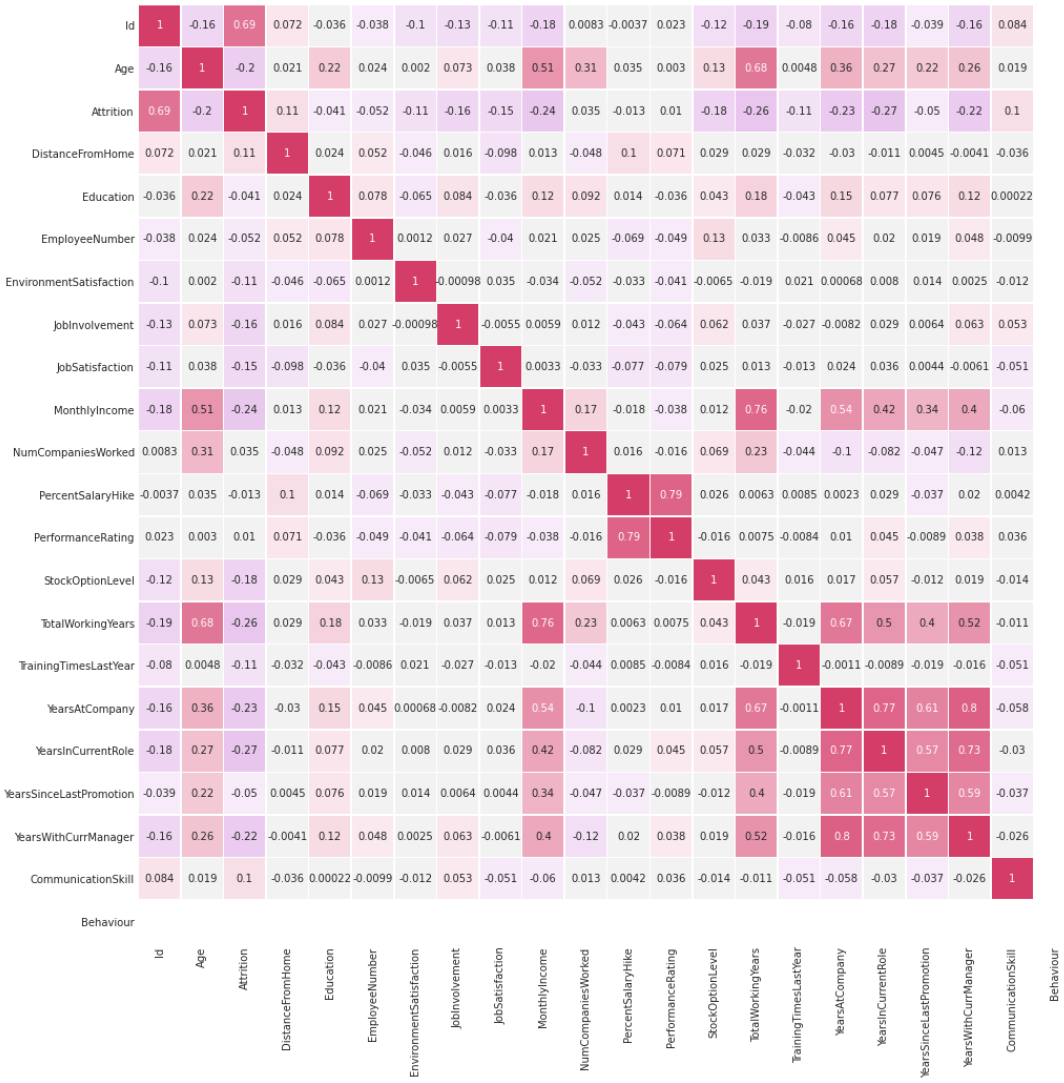


- No normal distribution for numerical data



- Features like MonthlyIncome, TotalWorkingYears , YearsAtCompany, YearsInCurrentRole and YearsWithCurrentManager have a rough threshold which divides the two target categories. Rest of the features don't help in clear classification of target variable.

- Features have high corelation with other features and hardly any feature is linearly corelated with target. So, we have issue of multicollinearity.
  Below is the heatmap for corelation visualization. More darker the block is higher is the corelation.



- Pairplot analysis conveys that many features have strong trend with each other. Normal distribution for features. Also, each feature has multiple peaks with in them.

- **Inferences and Conclusion**
  Some interesting observations from the visualizations

  - The proportion of values in categorical variables are not equal.
  - Single Employees have higher attrition value.
  - Employees in the age group of 32-45 have higher attrition probability
  - The Employees who experience business travel and have less job satisfaction pointer are more likely to leave company
  - The employee from Sales, Laboratory and Research department have higher attrition probability

- Hardly any feature strongly classifies the Target variable
- Employees subjected to overtime have higher attrition value.
- Percentage salary hike is also an important factor. Less the hike higher is the probability of employee attrition.

- **Algorithm and Technique**

Starting with basic classification model, will move to tree-based model and then ensemble models if performance found unsatisfactory.
As we don't know which algorithm and combination of hyperparameters will be best suited for dataset to give best generalized results, we have shortlisted below model.

1) Logistic Regression
2) Support Vector Machine
3) Gaussian Naïve Bayes
4) Decision Tree
5) Random Forest
6) Gradient Boosting
7) Bagging Classifier

Challenging part in our problem is class imbalance issue. Very few records belong to class of interest to us.

3. Methodology

- **Data Processing and Splitting**

Duplicate removal:
Dataset contains duplicate records. Below is the code to remove duplicates.

```
[7]:
train_data.drop_duplicates(inplace=True)
```

Feature Engineering:
As observed from visualization, we need to do feature engineering and removal of certain redundant feature to reduce multicollinearity. Multicollinearity effects the model performance. Hence, this step is must.

Below is the code to for feature engineering

```
def feature_engineering(data):
    data['JobInvolment_On_Salary']= data['JobInvolvement'] / data['MonthlyIncome'] * 1000
    data['DistanceFromHome_rootedTo_JobSatisfaction'] = data['DistanceFromHome']**(1/data['JobSatisfaction'])
    data['Mothers'] = np.where((data['Gender']=='Female') & (data['Age']>=36), 1,0)
    data['RateExtended'] = data['MonthlyIncome'] * (8 - data['JobSatisfaction'] - data['EnvironmentSatisfaction'])
    data['TotalJobSatisfaction'] = data['EnvironmentSatisfaction'] + data['JobSatisfaction']
    data['Sal_Rating']=data['PercentSalaryHike'] + data['PerformanceRating']
    data['Years_in_non_current_role']=data.YearsAtCompany-data.YearsInCurrentRole
    data.drop(['EnvironmentSatisfaction','YearsInCurrentRole','JobSatisfaction','PercentSalaryHike','MonthlyIncome','JobRole
    return data
```

Multicollinearity before feature engineering:

2]:

	VIF Factor	features
10	100.544367	PerformanceRating
9	42.125665	PercentSalaryHike
0	32.847989	Age
5	14.584347	JobInvolvement
12	13.615016	TotalWorkingYears
14	10.229251	YearsAtCompany
2	9.358355	Education
6	7.101860	JobSatisfaction
7	7.025604	MonthlyIncome
4	6.905842	EnvironmentSatisfaction
17	6.439047	YearsWithCurrManager
15	6.351248	YearsInCurrentRole
18	5.670901	CommunicationSkill
13	5.639723	TrainingTimesLastYear
3	3.842176	EmployeeNumber
8	2.737087	NumCompaniesWorked
16	2.360346	YearsSinceLastPromotion
1	2.308548	DistanceFromHome
11	1.907105	StockOptionLevel

Multicollinearity after feature engineering:

	VIF Factor	features
4	22.132607	JobInvolvement
17	18.729697	TotalJobSatisfaction
9	18.153850	YearsAtCompany
18	17.932026	Sal_Rating
7	9.048925	TotalWorkingYears
2	8.848843	Education
13	7.068094	JobInvolment_On_Salary
11	6.448851	YearsWithCurrManager
12	5.529290	CommunicationSkill
8	5.489200	TrainingTimesLastYear
19	5.175066	Years_in_non_current_role
16	5.031948	RateExtended
3	3.824892	EmployeeNumber
1	3.089843	DistanceFromHome
5	2.716761	NumCompaniesWorked
10	2.387032	YearsSinceLastPromotion
14	2.279467	DistanceFromHome_rootedTo_JobSatisfaction
6	1.931258	StockOptionLevel
15	1.430564	Mothers

Data Scaling:

Used MinMaxScaler to bring all numerical features on the same scale

```
#Scale the values
rb=MinMaxScaler()
new_data[num_var]=rb.fit_transform(new_data[num_var])
```

Convert categorical data to numeric:

We used get_dummies function to convert categorical data to numeric values.

In our dataset, categorical features were not ordinal, so we opted for one hot encoding of values.

```
21]: final_df=new_data.copy(deep=True)
     #encode categorical variables
     final_df=pd.get_dummies(final_df,cat_variables,drop_first=True)
     final_df.shape

21]: (1000, 33)
```

Data Splitting:

We used sklearn train_test_split to split data into train and test datasets.

From entire dataset we used 20% of data as test dataset.

```
:  #split train test data
   y=final_df['Attrition']
   X=final_df.drop(columns=['Attrition'],axis=1)
   X_train, X_test, y_train, y_test=train_test_split(X,y,random_state=42,test_size=0.2,shuffle=True)

:  print(X_train.shape,y_train.shape)
   print(X_test.shape,y_test.shape)

(800, 32) (800,)
(200, 32) (200,)
```

- **Implementation**

  I have trained various models, by tunning hyper parameters of the model.
  For each model I checked the performance using confusion matrix, classification report (for precision recall values) and check the area under curve.

  Below is the example of training, tunning and testing of RandomForest Classifier model on our dataset and check the model performance.

  RandomForest Classifier

```
RF_model=RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [20,30,40,50,100,125,150,175,200,225,250,275],
    'max_depth' : range(2,20),
    'criterion':["gini","entropy"],
    "min_samples_leaf": range(1,20),
    "max_leaf_nodes": range(2,20),
    'max_features':["auto", "sqrt", "log2"]
}

CV_rfc = RandomizedSearchCV(estimator=RF_model, param_distributions=param_grid, cv= 6,)
CV_rfc.fit(X_train, y_train)

print("Best Parameters:")
print(CV_rfc.best_params_)
print(CV_rfc.best_score_)
```

```
Best Parameters:
{'n_estimators': 200, 'min_samples_leaf': 5, 'max_leaf_nodes': 15, 'max_features': 'auto', 'max_depth': 10, 'criterion': 'gin
i'}
0.8512419107470167
```

```
rc_predictions=CV_rfc.predict(X_test)
```

```
print(confusion_matrix(y_test,rc_predictions))

print(classification_report(y_test,rc_predictions))
```

```
[[167   0]
 [ 33   0]]
              precision    recall  f1-score   support

         0.0       0.83      1.00      0.91       167
         1.0       0.00      0.00      0.00        33

    accuracy                           0.83       200
   macro avg       0.42      0.50      0.46       200
weighted avg       0.70      0.83      0.76       200
```

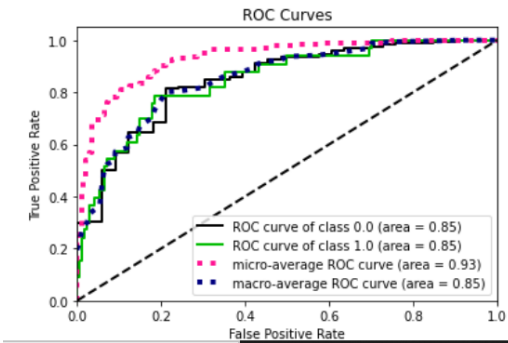
```
metrics_value=precision_recall_fscore_support(y_test, rc_predictions, average=None,labels=[0,1])
recall_for_class_1_RandomForest=round(metrics_value[1][1],2)


auc_randomforest=round(roc_auc_score(y_test,CV_rfc.predict_log_proba(X_test)[:,1]),2)
print('AUC score {}'.format(auc_randomforest))
skplt.metrics.plot_roc_curve(y_test, CV_rfc.predict_proba(X_test))
```

```
AUC score 0.85
```

```
C:\Users\lenovo\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:86: FutureWar
ot_roc_curve is deprecated; This will be removed in v0.5.0. Please use scikitplot.metrics.plot_roc instead.
  warnings.warn(msg, category=FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x193a1b20cc8>
```



4. Results

- **Model Evaluation**

  As mentioned, we will be using confusion matrix, classification report (for precision recall values) and AUC-ROC curve for model evaluation.

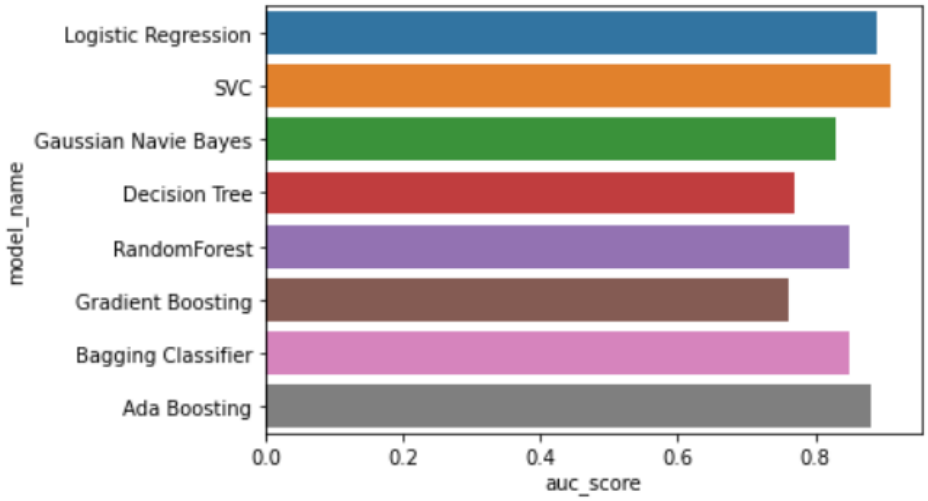
  We have calculated recall value for class label 1 and also calculated area under the curve for class label 1. This value I have compared in tabular format and as a barplot.

	model_name	auc_score	recall_score
0	Logistic Regression	0.89	0.484848
1	SVC	0.91	0.520000
2	Gaussian Navie Bayes	0.83	0.730000
3	Decision Tree	0.77	0.090000
4	RandomForest	0.85	0.000000
5	Gradient Boosting	0.76	0.270000
6	Bagging Classifier	0.85	0.730000
7	Ada Boosting	0.88	0.480000

  Comparing AUC score for various models.

```
1 sns.barplot(data=results_df,x='auc_score',y='model_name')
```

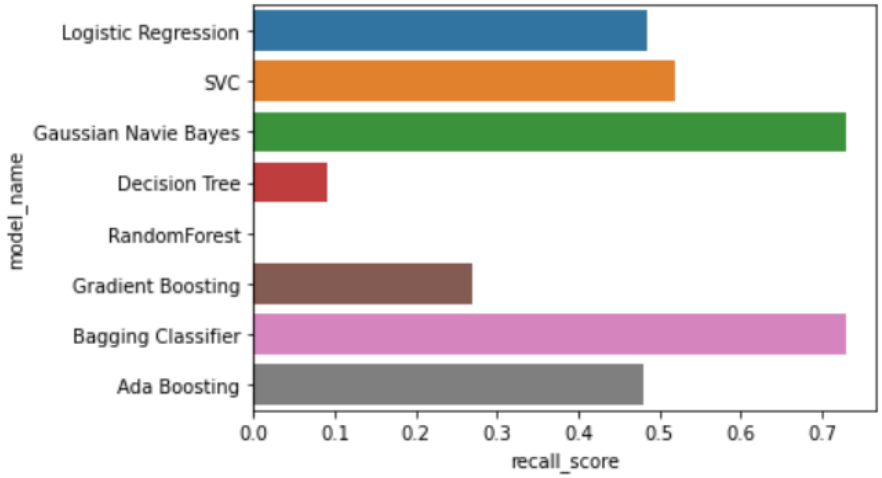
```
<matplotlib.axes._subplots.AxesSubplot at 0x193ac19e348>
```



Comparing Recall score for various models

```
1 sns.barplot(data=results_df,x='recall_score',y='model_name')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x193ab15f588>
```



- **Justification**

Considering the number of records which we have, number of features and amount of class imbalance in our dataset, we will use recall values to select the best model.

From the above models we observed Navie Bayes to be the best helps in identifying True positive. So, we have created BaggingClassifier using Naive Bayes as the base classifier, it helps in best reduction of False values for both the classes.

BaggingClassifier, has best balance between Recall score and AUC value for class 1

- **Saving model**
  Serialize the model using pickle python package.
  This serialized model can be easily consumed by code to do prediction of employee attrition.

5. Improvements:

Model performance can be improved by gathering more data.
More the data, better is the model training and better is the model performance on unseen data.
Considering the amount of features we have; we should try to collect more data to cover all possible combinations of features.

In case of dataset imbalance, we can also upsample minority class records.