HR Analytics: Predictive analysis of employee promotion status

Abstract

The size of companies has seen an exponential growth over the years. Corporations recruit anywhere from a few hundred to a few thousand employees every year. With such rates, HR management in companies is proving to be more and more significant everyday. Done manually, managing human resources is a laborious task, given the sheer quantity of employees. Luckily, over the years, data analytics in HR is emerging as an integral part in corporate operation. Yet, there remain a few problems that involve human involvement, one of them being selecting candidates that are eligible for a promotion. This paper proposes a solution involving machine learning techniques such as the Random Forest and XGBoost algorithms to learn from past employee records to aid this decision making process.

1. Introduction

HR analytics in companies plays a major role in restructuring the operanda of their HR department. A company of sizeable proportions deals with hundreds of employee records every day. Although HR analytics has been in operation in companies for years, some of these operations are still done manually. Automating such processes will aid in saving valuable time and increasing overall efficiency in the operation of the company.

Eligibility for promotions depends on many different criteria. These criteria vary between companies and even between departments within a company. Currently, there is no definitive method to determine why an employee receives a promotion since such decisions require logical reasoning and understanding the current environmental factors that computers are just not capable of. Problems such as this are where we utilize machine learning to our advantage.

Machine learning has been used to solve similar problems in different domains for several years now. In areas where some kind of human intervention is necessary, a well trained machine learning algorithm has been proven to be an acceptable substitute, if not an ideal solution. Here, we will be using machine learning techniques to not only predict the promotion status of future employees, but to also determine which of the provided attributes from the employees' data is most relevant to making this prediction.

2. Literature survey

Machine learning, as the name suggests, is the field of computing that deals with algorithms and concepts that attempt to emulate specified human behaviour. First pioneered in the late 1950s, machine learning has come a long way over the years to the point where we can now depend on them to perform reliably even in production level environments. The proposed solution to the promotion problem here is to train a machine learning algorithms to analyze past employee records of employees who were shortlisted for a promotion.

These records include information about the employee, along with their promotion status, that describes whether they were promoted or not. This method of training a machine learning algorithm is called Supervised Learning, where the final result expected from the algorithm is provided. This is the approach that is going to be used here. Supervised learning algorithms generally deal with problems that have definitive solutions to them, such as classifying an item as belonging to one of multiple categories.

The specified algorithms being proposed in this paper are namely, the Random Forest and XGBoost algorithms. Both of these algorithms operate on the principle of a much simpler algorithm, the Decision Tree.

2.1 Decision Tree

A decision tree, as the name suggests, is a graphical representation of different attributes of any given data in a tree like structures. It attempts to map out the correlation between the combination of values of each different attribute to the overall outcome.

A decision tree is represented as a flowchart, where each node represents an attribute of the data and the connections emerging from that node are the possibles values that can be assigned to that attribute. The edges present in the tree are seen as different pathways that can be taken to make a prediction, based on the what the provided values are.

2.2 Random Forest

As useful as decision trees are, they fall short as problems become more and more complex. This is attributed to the fact that they are quite unstable, as even small changes in the data can affect the structure of the tree profoundly. This causes overfitting, which means that the tree cannot perform as well on real world data as it does on it’s training data.

The Random Forest algorithm aims to overcome that shortcoming. Rather than have a single tree making a definitive decision, it utilizes multiple trees, each making a prediction to arrive at its final conclusion. This algorithm uses the drawback of the decision tree to its advantage to create diverse trees by altering the data little by little. This helps keep the overfitting problem in check. Finally, the predictions from all the decision trees are polled and the prediction with the highest frequency is taken as the final prediction.

2.3 XGBoost

The XGBoost(short for Extreme Gradient Boosting) algorithm is essentially a decision tree which utilizes the concept of gradient boosting to enhance the performance of the decision tree. This algorithm performs best when being used to model small to medium sized structured data.

The XGBoost algorithm is generally preferred over other conventional algorithms as it combines many different traits from other algorithms, such as bagging, gradient boosting, random forest and decision trees and makes improves upon them through system optimization and algorithmic enhancements such as regularization, sparsity awareness, weighted quantile sketch, and so on.

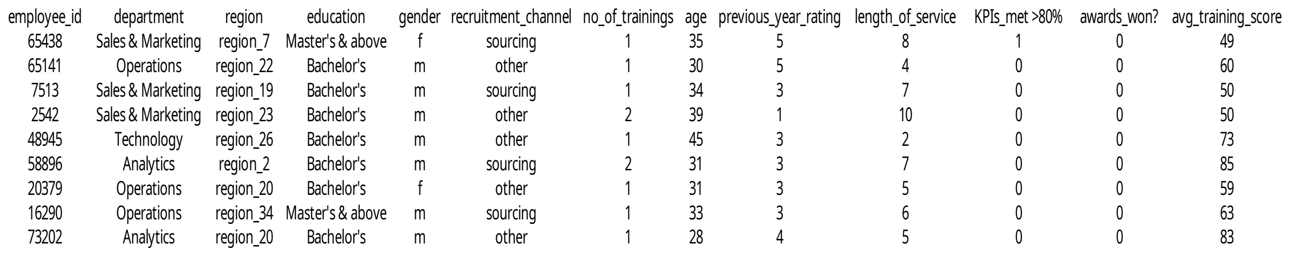
2.3.1 Gradient Boosting

Gradient boosting is a repetitive algorithm used to leverage the patterns of mistakes made by a model to strengthen the model using weak predictions. Basically, the data is modeled very simply and is analyzed for errors to identify data points that are difficult for the model to fit. The model is tweaked to fit better for those particular data points. Finally, this is combined with the original models for an optimal solution.

3. Methadology

3.1 Dataset

To fully explain the approach to solving this problem, we must first analyze and understand our dataset. Our dataset comprises of data collected by a company on previous candidates who were shorlisted for a promotion. Below, is a screenshot of the first 10 rows of our data. This dataset consists of 14 different attributes or columns, with 54808 total observations or rows.



employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	awards_won?	avg_training_score
65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35	5	8	1	0	49
65141	Operations	region_22	Bachelor's	m	other	1	30	5	4	0	0	60
7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34	3	7	0	0	50
2542	Sales & Marketing	region_23	Bachelor's	m	other	2	39	1	10	0	0	50
48945	Technology	region_26	Bachelor's	m	other	1	45	3	2	0	0	73
58896	Analytics	region_2	Bachelor's	m	sourcing	2	31	3	7	0	0	85
20379	Operations	region_20	Bachelor's	f	other	1	31	3	5	0	0	59
16290	Operations	region_34	Master's & above	m	sourcing	1	33	3	6	0	0	63
73202	Analytics	region_20	Bachelor's	m	other	1	28	4	5	0	0	83

Figure 1: Top 10 rows of the data

A breakdown of the attributes of this dataset is as follows:

employee_id (int): Unique id of the employee.

department (string): The department to which the employee belongs to. Possible values are: 'Analytics' .

'Finance' , 'HR' , 'Legal' , 'Operations', 'Procurement', 'R&D', 'Sales & Marketing', 'Technology'

region (string): Region of employment. Possible values are: 'region_10', 'region_11', 'region_12', 'region_13', 'region_14', 'region_15', 'region_16', 'region_17', 'region_18', 'region_19', 'region_2', 'region_20', 'region_21', 'region_22', 'region_23', 'region_24', 'region_25', 'region_26', 'region_27', 'region_28', 'region_29', 'region_3', 'region_30', 'region_31', 'region_32', 'region_33', 'region_34', 'region_4', 'region_5', 'region_6', 'region_7', 'region_8', 'region_9' .

education (string): Describes the level of education of the employee. Possible values are: 'Bachelor\'s', 'Below Secondary', 'Master\'s & above'.

gender (string): Gender of the employee. Possible values are: ‘f’, ‘m’.

recruitment_channel (string): Channel of recruitment of the employee. Possible values are: ‘referred’, ‘sourcing’, ‘other’.

no_of trainings (int): Describes the number of training programs completed by the employee. Range: from 1 to 10.

age (int): Describes the age of the employee.

previous_year_rating (int): Employee rating from previous year. Range from 1 to 5.

length_of_service (int): The service length of the employee in years.

KPIs_met>80% (int): Describes whether the employee’s Key Performance Indicators score are greater than 80%. Value is 1 if yes, else 0.

awards_won? (int): Whether the employee won any awards last year. Value is 1 if yes, else 0.

avg_training_score (int): Employee’s average training score in training evaluations.

is_promoted (int): Whether the employee was promoted or not. Value is 1 if yes, else 0.

Here, the is promoted variable is our target variable. This is what we have to predict to produce our final result.

3.2 Feature engineering

First off, we remove the employee_id column as it is just a column to distinguish records by and will not realistically impact the decision of whether a employee is to be promoted or not. This brings down our variable count to 13.

3.2.1 Duplicate values

Even though the size of our dataset is huge, we cannot be sure that all the observations in the data are unique. Looking at our data, we find that there are 118 duplicate records that are present in our dataset.

Even though it does not seem like a significantly large number, duplicate records will negatively impact the training process of machine learning algorithms and may cause them to overfit.

After the removal of the duplicate values, we have 54690 observations left.

3.2.2 Null values

Looking at our dataset, we see there are missing values. Observations with such values are not suitable for machine learning algorithms.

Luckily, out of 54690 observations, we have about 4042 observations with missing values. That accounts to about 7% of our total data.

A plot of this can be seen in figure 2.

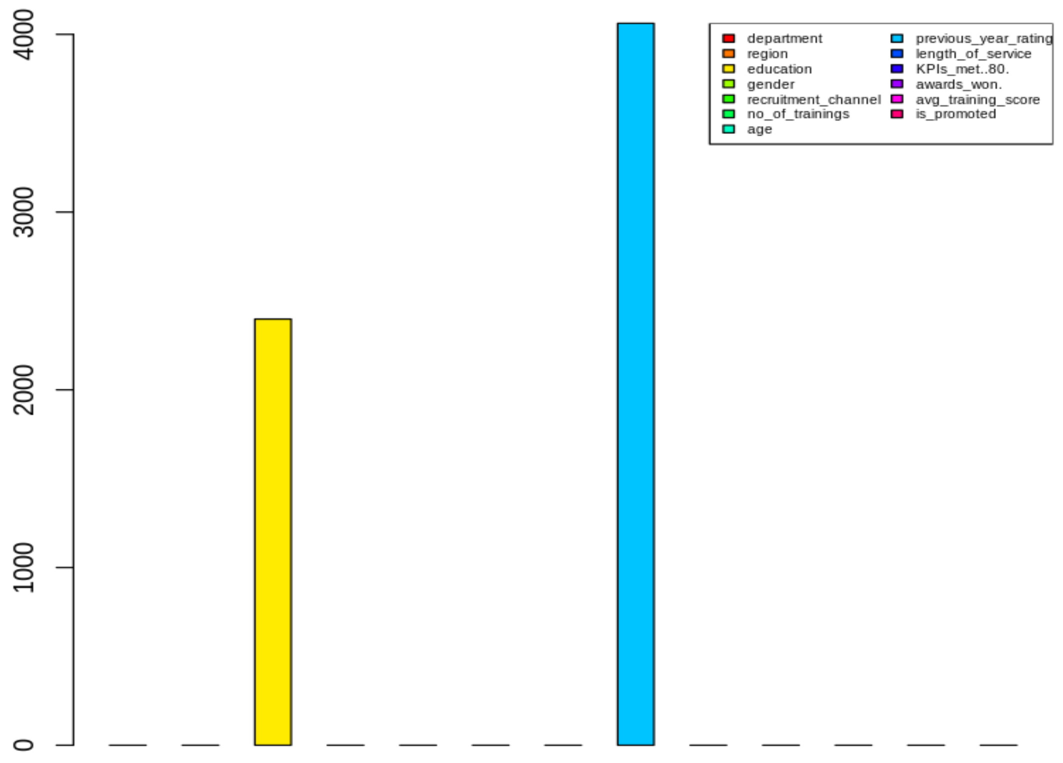


Figure 2: Barplot of columns along with the frequency of null values in them

This plot shows the frequency of null values in each column. There are only two columns with null values in our data, education and previous_year_rating, with 2398 and 4062 records with missing values respectively. These can be dealt with in one of two ways.

Either, these observations could be completely removed or they could be systematically generated using existing data. We choose the latter, as with that approach, we still maintain the size of our dataset.

Since both of these attributes are categorical, we can fill in missing values by finding the mode of the values that belong to that attribute. We also take into account the value of the target class for each attribute.

3.3 Pre-processing

The data is pre-processed to make it suitable for our algorithms. First, the target column values are renamed from “0” and “1” to “no” and “yes” so that the algorithm recognizes as categorical variables. Then, dummy variables are generated from the values of all the categorical attributes, or performs one-hot encoding for all the attributes.

3.4 Class imbalance problem

In this dataset, there is a clear imbalance between the values of our target variables. Out of 54690 observations, only 4665 observations are of class “no”, while 50025 observations are of class “yes”. This is a significant issue, as the difference exceeds more than 50% of the data.

For a machine learning algorithm to be properly able to parse and understand the given data, there should ideally be an equal distribution of the number of examples with the different classes it is meant to classify other data into. When one class takes precedence over the other class in the dataset, the algorithm is less likely to learn what the properties of each class are and tends to forget the less frequent class’s properties all together during training.

To ensure that this does not happen, we must make sure that there are equal number of examples for both the cases. Here, we simply randomly sample a subset of the data where the class is “yes”, as it is the class with higher frequency and append it to the observations with the class “no” to generate a new, minified training set with equal number of “yes” and “no” observations. This process is called undersampling.

This brings down the size of our dataset to 9330 observations total. Even though it is only a fraction of the original 54690, it is still a significant amount and should be enough to train our algorithms along with being balanced.

Figure 3(a): Distribution of class percentage before undersampling

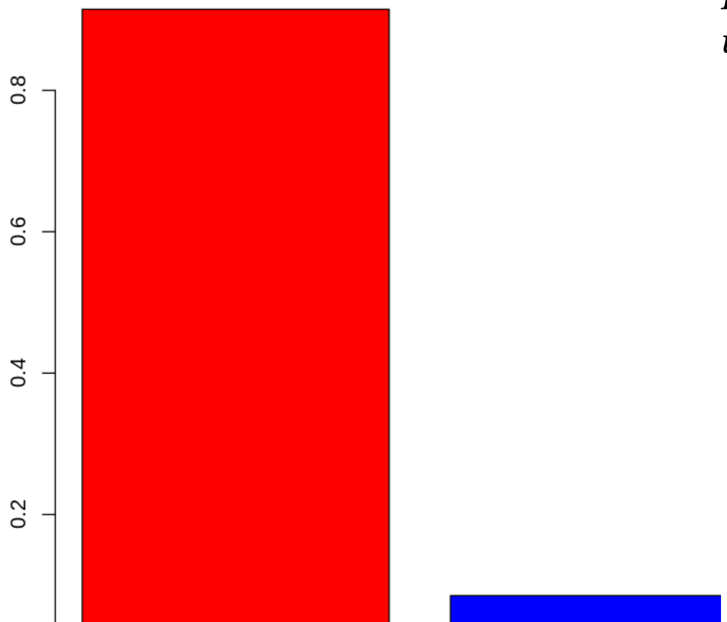
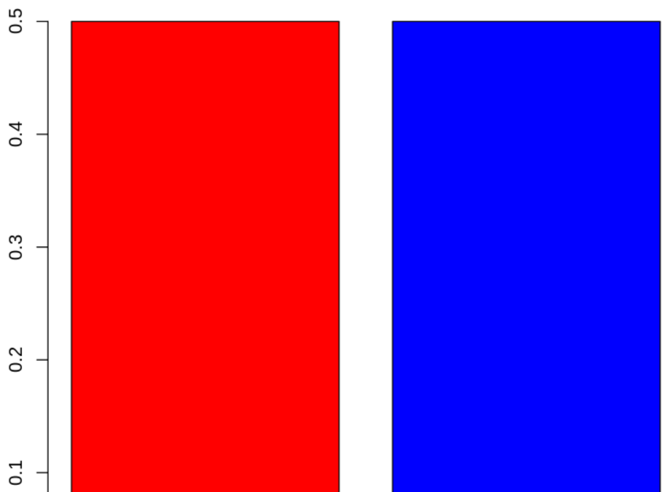


Figure 3(b): Distribution of class percentage after undersampling



3.5 Training and Evaluation

The dataset is split into a training set and a testing set where the testing set contains 1/5ths of the records in the dataset and the rest belongs to the training set.

The chosen method of evaulation to measure the performance of our models is K-fold cross validation, where K is taken to be 5.

4. Experimental results

The training of both the models reulted the following:

4.1 Random forest

The consfusion matrix of the model evaluated on the training set, with 5 fold cross validation is as follows:

```
Cross-Validated (5 fold) Confusion Matrix
          Reference
Prediction   no  yes
       no  37.5  5.8
       yes 12.5 44.2
```

```
 Accuracy (average) : 0.8162
```

And the confusion matrix of the model evaluated on the test set:

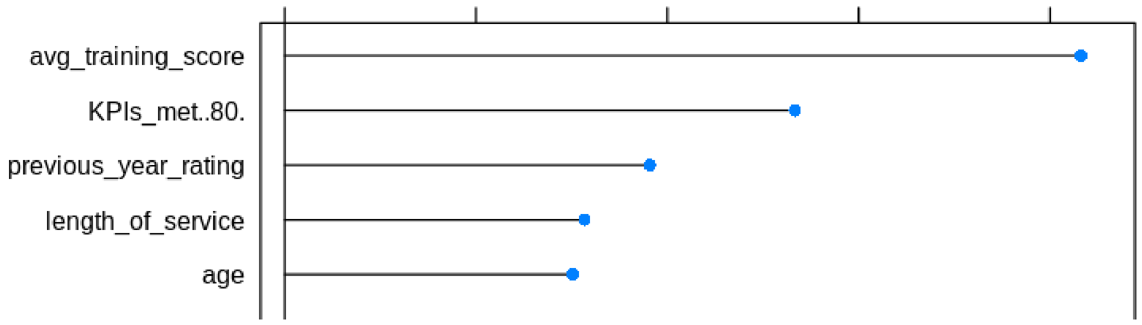
```
Confusion Matrix and statistics

rfpreds  no yes

    no  719 122
    yes 214 811

               Accuracy : 0.8199
                 95% CI : (0.8017, 0.8371)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6399

 Mcnemar's Test P-Value : 6.889e-07

            Sensitivity : 0.7706
            Specificity : 0.8692
         Pos Pred Value : 0.8549
         Neg Pred Value : 0.7912
             Prevalence : 0.5000
         Detection Rate : 0.3853
   Detection Prevalence : 0.4507
      Balanced Accuracy : 0.8199

       'Positive' Class : no
```

We can see that the performance of the model is consistent over both the training and testing set. So, we can ensure that our model has not overfit.

The random forest model also helps us calculate the importance of each attribute. The attribute importance scores, as predicted by our trained random forest model is as follows:

Attribute	Overall Score
avg_training_score	832.70
KPIs_met..80.	533.48
previous_year_rating	381.64
length_of_service	313.44
age	301.19
departmentSales & Marketing	186.40
awards_won.	114.01
departmentOperations	102.15
departmentProcurement	67.42
no_of_trainings	67.28
recruitment_channelsourcing	60.31
genderm	55.79
regionregion_2	41.94
departmentTechnology	39.70
departmentFinance	38.99
regionregion_22	38.76
regionregion_7	36.84
educationMaster's & above	31.27
departmentHR	30.92
educationBachelor's	30.88

Table 1: Top 20 most important features by Random Forest with no scaling of scores.



We can see here that the model has rated the average_training_score, KPI_mets>80 and the previous_year_rating as the 3 most important attributes for this decision.

4.2 XGBoost

Confusion matrix for the training set, evaluated with 5 fold cross validation is as follows:

```
Cross-Validated (5 fold) Confusion Matrix


          Reference
Prediction   no  yes
       no  38.1  5.1
       yes 11.9 44.9

 Accuracy (average) : 0.8301
```

And the confusion matrix for the test set:

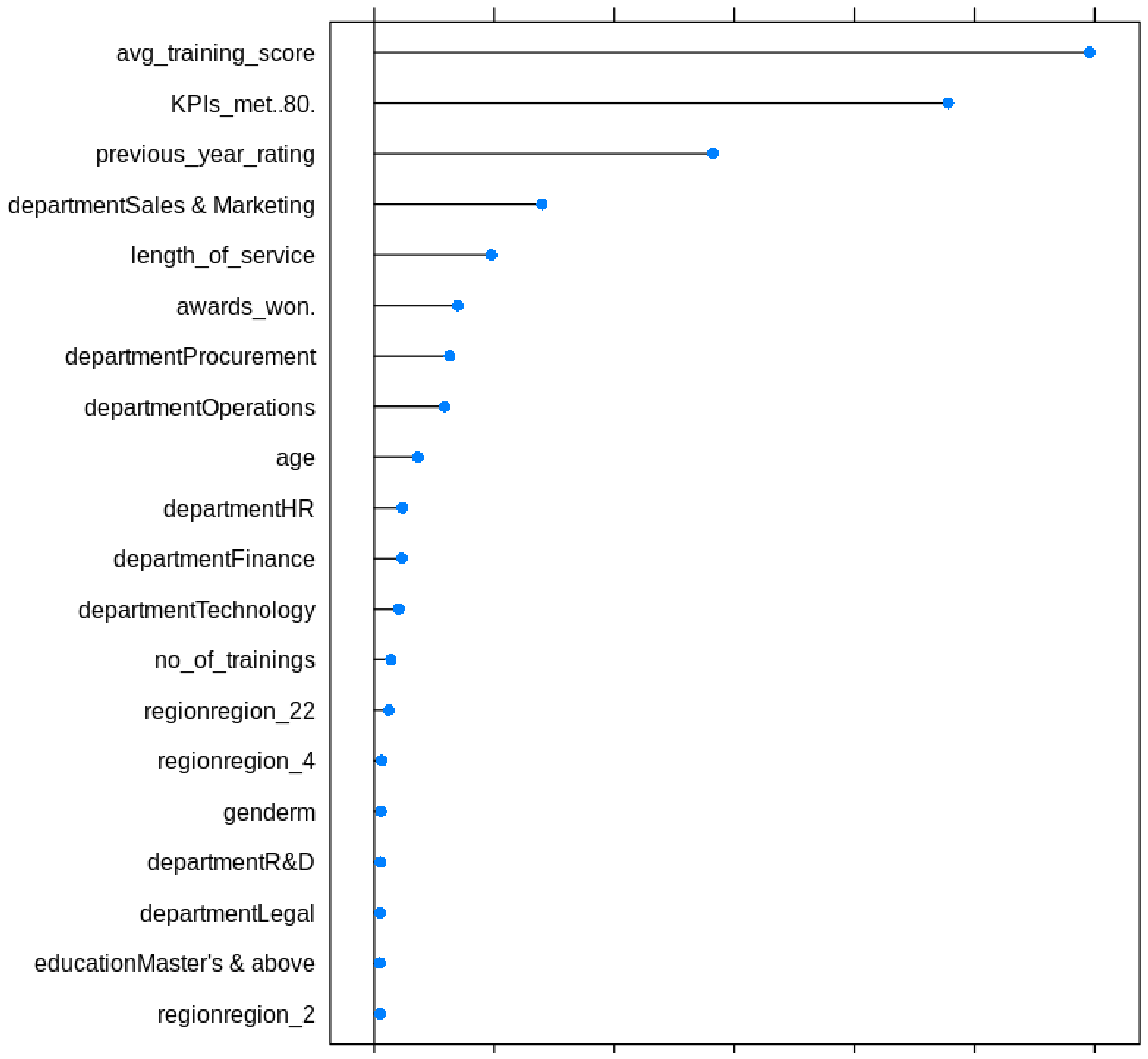
```
Confusion Matrix and Statistics


xgbpreds  no yes
     no  731 108
     yes 202 825

               Accuracy : 0.8339
                 95% CI : (0.8162, 0.8505)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6677

 Mcnemar's Test P-Value : 1.277e-07

            Sensitivity : 0.7835
            Specificity : 0.8842
         Pos Pred Value : 0.8713
         Neg Pred Value : 0.8033
             Prevalence : 0.5000
         Detection Rate : 0.3917
   Detection Prevalence : 0.4496
      Balanced Accuracy : 0.8339

       'Positive' Class : no
```

The XGBoost model gives us a final validation accuracy of 0.8339(or 83.38%), which is marginally higher when compared to the random forest. Other statistics such as the specificity and the sensitivity are also higher, which tells us that the XGBoost model will give us a better overall performance.

The feature importance table as calculated by the XGBoost tree is as follows:

Attribute	Overall
avg_training_score	0.298036
KPIs_met..80.	0.239061
previous_year_rating	0.140940
departmentSales & Marketing	0.069869
length_of_service	0.048764
awards_won.	0.034820
departmentProcurement	0.031456
departmentOperations	0.029417
age	0.018234
departmentHR	0.011814
departmentFinance	0.011585
departmentTechnology	0.010275
no_of_trainings	0.007006
regionregion_22	0.006087
regionregion_4	0.003136
genderm	0.002850
departmentR&D	0.002684
departmentLegal	0.002543
educationMaster's & above	0.002407
regionregion_2	0.002385

Table 2: Top 20 most important features by XGBoost model without scaling scores



We notice that the list of the 10 most important features according to both the Random Forest and XGBoost models are nearly identical with the exception of no_of_trainings and departmentHR. This tells us that the other attributes can be discarded to get improved performance from the models.

5. Conclusion

As we have demonstrated above, the use of machine learning as a predictive decision making tool or at least a suggestive tool is a completely viable solution for the presented problem. With fairly limited data, we were able to train algorithms to perform with significantly good accuracy. We can improve upon this with more data and more optimized solutions. Thus, we conclude that machine learning in the field of HR analytics by reducing the amount of time that goes into decision making, thereby increasing work efficiency.

6. References

- Dr. Abdul Quddus Mohammed, HR Analytics: A Modern Tool in HR for Predictive Decision Making, Journal of Management, 6(3), 2019, pp. 51-63. http://www.iaeme.com/jom/issues.asp?JType=JOM&VType=6&IType=3
- Tianqi Chen, Carlos Guestrin, 2016, XGBoost: A Scalable Tree Boosting System, https://arxiv.org/pdf/1603.02754v3.pdf
- Breiman, L. Machine Learning (2001) 45: 5. https://doi.org/10.1023/A:1010933404324