Machine Learning Engineer Nanodegree

Capstone Project Report

Udaykiran Dakavarapu

25/06/2018

Definition

Project Overview

Employee retention is of significant importance to companies. It should come as no surprise that companies that lead in employee engagement are often some of the most successful and profitable companies. Poor employee engagement means turnover, low morale, poor customer service, and a general blow to your bottom line. However, this issue become ultra-costly when high-performing employees start leaving because of low engagement. Supervised learning deals with Regression and Classification. Classification is being implemented in different fields Ex: Medical diagnosis and this model has gained a momentum to understand the change in behaviour of the consumer, so this helps business works on the data. Every company invest a lot on employee to train them and makes employee to work in any kind of situation and in any kind of project. And there are many reasons employees to quit job. Some of them are rude behaviour, work imbalance, Feeling undervalued, Employment misalignment etc. If a company Invest in skill enhancement of an employee he should use it for the growth of the business, not for his personal profit. And I was searching for the dataset which has features that can be the reasons for the employee to quit job.

Link: https://www.kaggle.com/giripujar/hr-analytics/data#HR_comma_sep.csv

Link: https://peoriamagazines.com/ibi/2009/dec/12-reasons-employees-leave-organizations

Link: https://www.visier.com/clarity/reduce-employee-turnover-with-workforce-analytics/

Problem Statement

HR managers have many responsibilities within an organisation as they protect the interests of both the employer and the employee. For employers, they manage employee relations and identify ways to cut labour costs. For employees, they protect their rights ensuring that employers operate within the scope of employment and labour law. HR has a tough job to find the root cause of attrition. For suppose if HR knows that a particular employee is more probable of leaving and could take damage control action. So, HR needs to find what features are making employees to leave. So I classified the data into sets, the employees who stayed in the company and who left the company and I treated it as classification problem. Model will be predicting if person is potential employee who will stay or leave the company. Satisfaction level is important attribute and a person not satisfied will probably leave the organization. Last evaluation means that performing person is not getting enough feedback and hence enthusiasm to continue work is lost. Number of project completed does not seem important attribute as project can be small or big.

Features and Description:

- satisfaction_level: It is the level of satisfaction of the employee.
- last_evaluation: It is the time since last performance is evaluation.
- number_project: It is the number of projects completed while at work.
- average_montly_hours : It is the average monthly hours at workplace.
- time_spend_company : It is number of years spent in the company.
- Work_accident: whether the employee had a workplace accident or not.
- Left: Whether the employee left the work place or not.
- Promotion_last_5years: Whether employee promoted in last five years or not.
- Department: Department they work for.
- Salary: It is relative level of the salary.

Company is in a situation where its talented and experience employee are quitting jobs and there might be n number of reasons. HR job is to identify the root cause and take a remedy action. So that a preventive measure is taken, and employee could be retained. I plan to use Classification model of Supervised Learning techniques. There are total of 10 features available. Since

all of them will not have equal impact on prediction I will use PCA to identify the most predictive feature and use those. By considering the above features, satisfaction level is very important feature for the employee to stay in the company, because if the work what he is doing doesn't give him satisfaction means he will definitely leave the company. last_evaluation is also an important feature which will give confidence to the employee. I think number_project is not that much important because if an employee engaged in big project means it will take time to complete the project. Average_monthly_hours also we have to consider because if he is putting his entire time in doing work he/she will not get enough time to relax so he may eventually quit the job. time_spend_company is high means he will be paid good. left feature is the decision column which we have to predict. And the main feature is salary of the employee. If he is satisfied with the salary definitely he will continue in the company. Promotion feature will give employee satisfaction about his work. Department is not that much important. I have taken the Human resource analytics dataset for the project and will predict how many employees will left the company and why are they leaving the company and what features are helping them to leave the company. The dataset consists of 14999 rows and 10 columns. And the number of employees left are 3571. And the remaining employees are 11425 and the target set is left feature. In project we are predicting the employees who are going to left. And the data set is taken from the Kaggle

Link: https://www.kaggle.com/giripujar/hr-analytics/data#HR_comma_sep.csv

Metrics

Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.

$$\text{Accuracy} = (TP+TN)/(TP+TN+FP+FN)$$

In the Numerator, are our correct predictions (True positives and True Negatives) (Marked as red in the fig above) and in the denominator, are the kind of all predictions made by the algorithm (Right as well as wrong ones).

Precision is a measure that tells us what proportion of patients that we diagnosed as having cancer, actually had cancer. The predicted positives (People predicted as cancerous are TP and FP) and the people actually having a cancer are TP.

$$\text{Precision} = TP/(TP+FP)$$

Recall is a measure that tells us what proportion of patients that actually had cancer was diagnosed by the algorithm as having cancer. The actual positives (People having cancer are TP and FN) and the people diagnosed by the model having a cancer are TP.

(Note: FN is included because the Person actually had a cancer even though the model predicted otherwise).

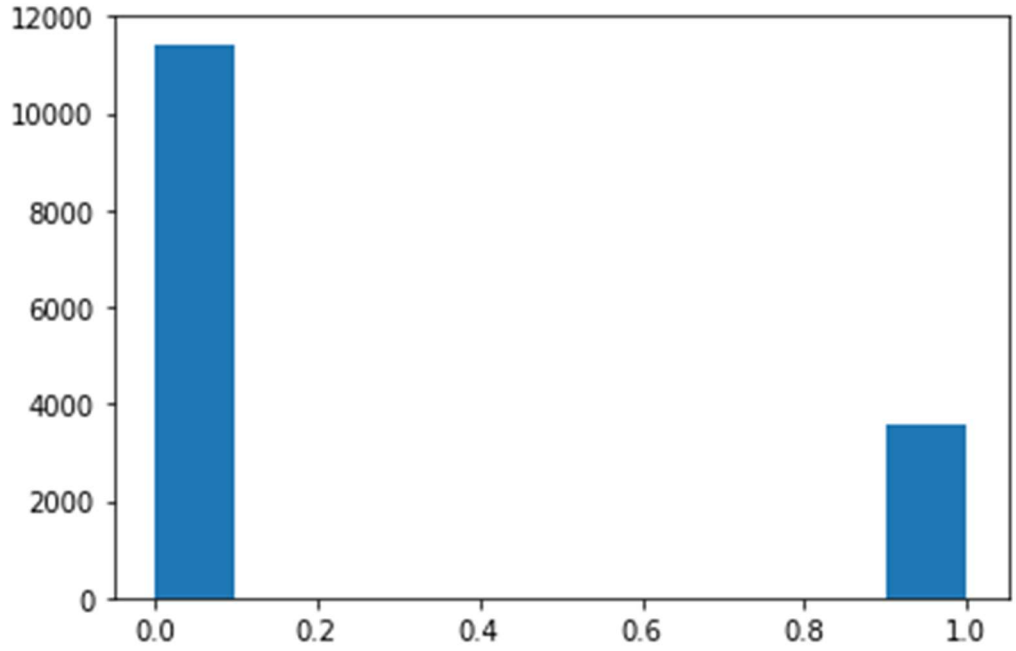
$$\text{Recall} = TP/(TP+FN)$$

Fβ score – measures the effectiveness of retrieval with respect to a user who attaches beta times as much importance to recall as precision.

$$F\beta=(1+\beta 2) \cdot \text{precision} \cdot \text{recall} / (\beta 2 \cdot \text{precision}) + \text{recall}$$

When beta=0.5 more emphasis is placed on precision. This is called f 0.5 score.

I choose fbeta score as a metric because my data is imbalanced. Accuracy will not give better results when the data is imbalanced. The histogram for the target variable left is shown below. Accuracy is not a good metric when the data is imbalanced.



Analysis

Data Exploration

In this data exploration section, I calculated the total number of records and the number of employees who are currently working in the company and the number of employees left and percentage of employees left and checked for null values in the data set. So, I predicted that there are no null values in the data set.

```
number of records: 14999
number of employee left: 3571
number of employee currently working: 11428
Percent of employee left: 23.81%
number of null rows: 0
```

data.describe()

```
In [31]: data.describe()
```

Out[31]:

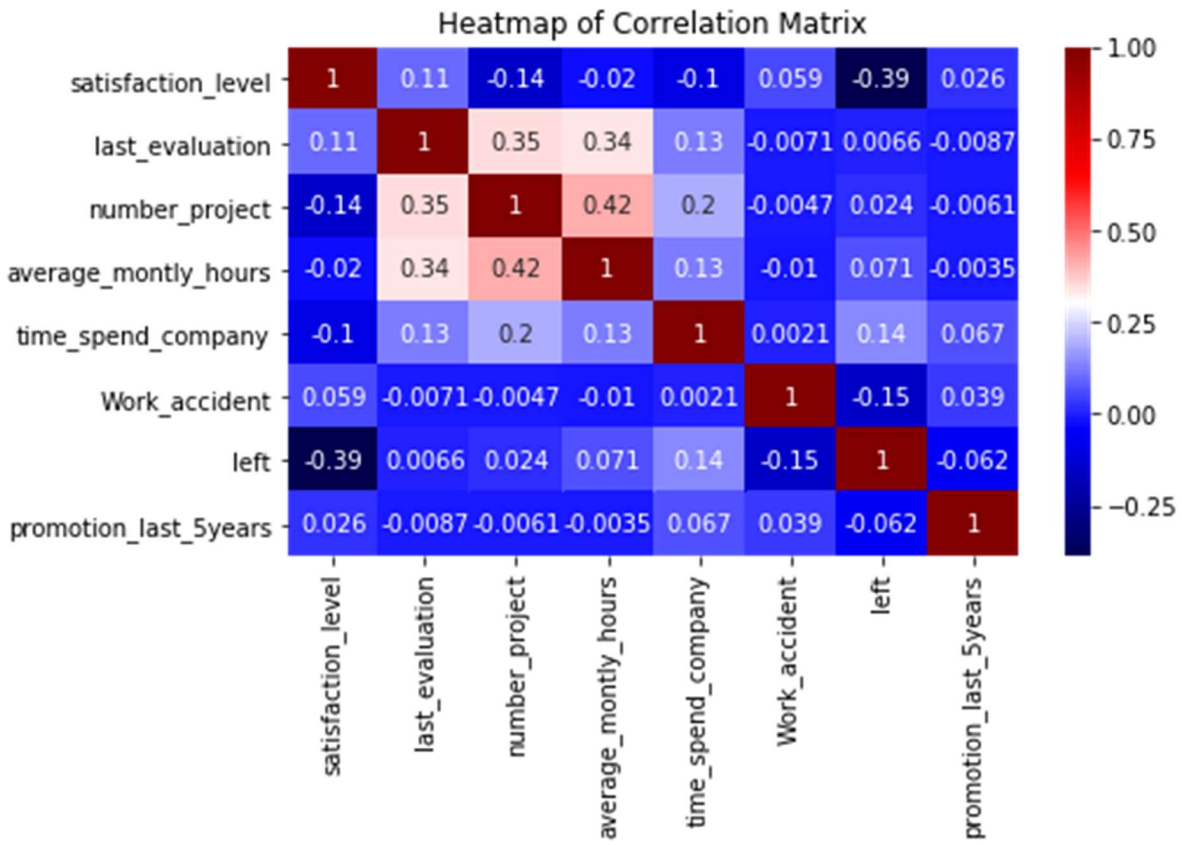
	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	0.144281
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	1.000000

data.head()

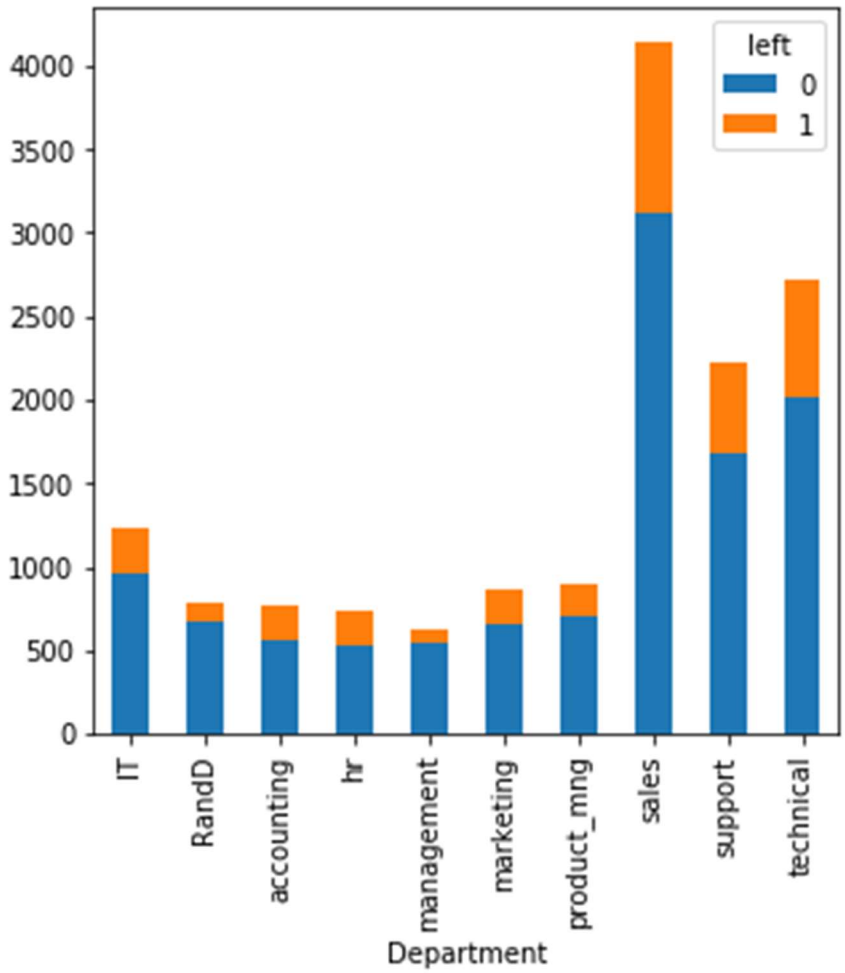
satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Department	salary
0.38	0.53	2	157	3	0	1	0	sales	low
0.80	0.86	5	262	6	0	1	0	sales	medium
0.11	0.88	7	272	4	0	1	0	sales	medium
0.72	0.87	5	223	5	0	1	0	sales	low
0.37	0.52	2	159	3	0	1	0	sales	low

Visualization

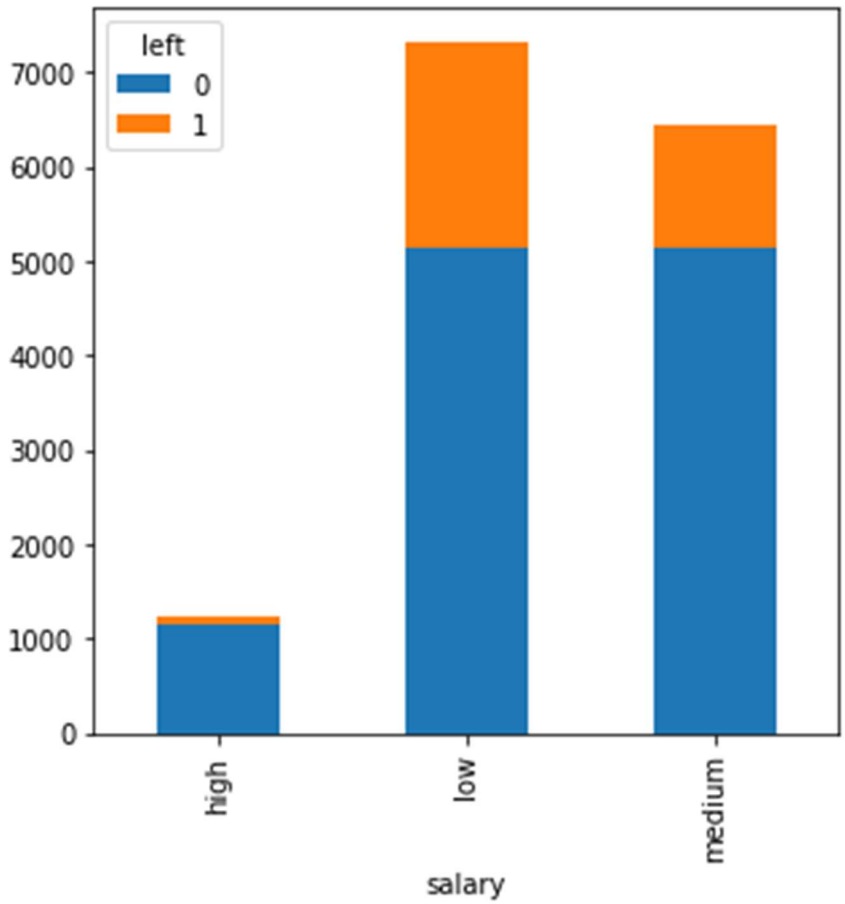
In this first I plot a heatmap of all the features to know how the features are correlated. It states that average monthly hours and number of projects are highly correlated and hence using any one column for machine learning model should good. In the below graph average monthly hours and number of projects has highest value in the below graph.



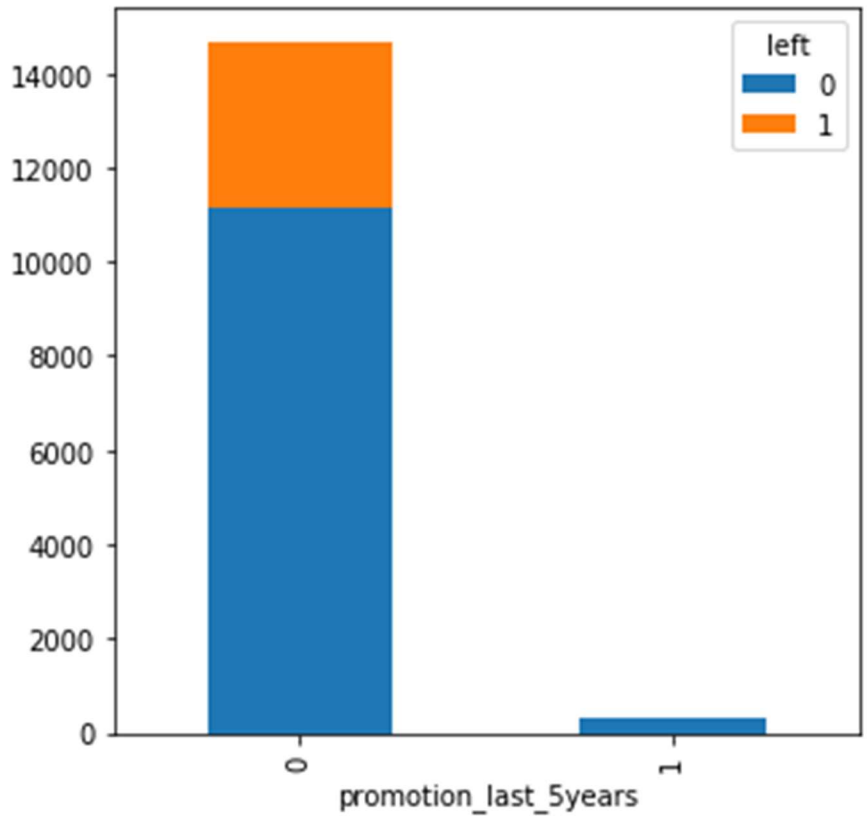
I am interested in finding which department are most affected. I plotted the graph between department and the left (target variable). From the below graph we can clearly say that the department is not a problem.



I would also like to plot the graph between salary and the target variable. If an employee is willing to move from the company the main reason is based on the salary. If the salary of an employee is low he would like to look for better opportunity. So he shows more interest in leaving the job.



I also plot the graph between promotion_last_5years and salary



Algorithm and Techniques

In machine learning the biggest problem is which technique we have to choose. Logistic regression was simple linear classification of data and it is good to act as a Benchmark model and not a model for practical use with multiple features. SVM was good for binary classification but when we have multiple features then Kernel SVM can be used. Challenge is to identify kernel and it leads to error if correct kernel is not identified. Stochastic Gradient descent requires a lot more tuning than other model and descent is sometime slow and sometime very steep. Ensemble Learning was the choice for me. Ensemble Learning makes a strong classifier from number of weak classifiers. This is done by building a model from training data, then creating a second model which corrects the error from the first model. SVM was also good for binary classification. For suppose if the data has multiple features we can use kernel based SVM. Ensemble learning methods are used to increase the accuracy of the model. So, this can be done by building a model from training data, and again creating a second model which corrects the error from the first model. The Models are added until training data is predicted perfectly, or maximum number of model are added. AdaBoost can be used to boost the

performance of any machine learning algorithm. It is best used with weak learners. AdaBoost was the first really successful boosting algorithm developed for binary classification. It is the best starting point for understanding boosting. The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. Because these trees are so short and only contain one decision for classification, they are often called decision stumps. Each instance in the training dataset is weighted. The initial weight is set to :

$$weight(xi) = 1/n$$

Where xi is the i'th training instance and n is the number of training instances.

A weak classifier (decision stump) is prepared on the training data using the weighted samples. Only binary (two-class) classification problems are supported, so each decision stump makes one decision on one input variable and outputs a +1.0 or -1.0 value for the first or second-class value.

The misclassification rate is calculated for the trained model. Traditionally, this is calculated as:

$$error = (correct – N) / N$$

Where error is the misclassification rate, correct are the number of training instance predicted correctly by the model and N is the total number of training instances. For example, if the model predicted 78 of 100 training instances correctly the error or misclassification rate would be (78-100)/100 or 0.22.

This is modified to use the weighting of the training instances:

$$error = sum(w(i) * terror(i)) / sum(w)$$

Which is the weighted sum of the misclassification rate, where w is the weight for training instance i and terror is the prediction error for training instance i which is 1 if misclassified and 0 if correctly classified.

For example, if we had 3 training instances with the weights 0.01, 0.5 and 0.2. The predicted values were -1, -1 and -1, and the actual output variables in the instances were -1, 1 and -1, then the terrors would be 0, 1, and 0. The misclassification rate would be calculated as:

$$error = (0.01*0 + 0.5*1 + 0.2*0) / (0.01 + 0.5 + 0.2)$$

or

$$error = 0.704$$

A stage value is calculated for the trained model which provides a weighting for any predictions that the model makes. The stage value for a trained model is calculated as follows:

$$stage = \ln((1-error) / error)$$

Where stage is the stage value used to weight predictions from the model, ln() is the natural logarithm and error is the misclassification error for the model. The effect of the stage weight is that more accurate models have more weight or contribution to the final prediction.

The training weights are updated giving more weight to incorrectly predicted instances, and less weight to correctly predicted instances.

For example, the weight of one training instance (w) is updated using:

$$w = w * \exp(stage * terror)$$

Where w is the weight for a specific training instance, exp() is the numerical constant e or Euler's number raised to a power, stage is the misclassification rate for the weak classifier and terror is the error the weak classifier made predicting the output variable for the training instance, evaluated as:

$$terror = 0$$

if(y == p), otherwise 1

Where y is the output variable for the training instance and p is the prediction from the weak learner.

This has the effect of not changing the weight if the training instance was classified correctly and making the weight slightly larger if the weak learner misclassified the instance.

Benchmark Model

Logistic regression is used as benchmark model. FBeta score of benchmark model is reference and other model will be judging to perform better if their fbeta score will be greater than Logistic regression model. Accuracy, f-beta score and confusion matrix and will try to get better results in the ensemble learning models.

```
In LogisticRegression

True Negative :2134
False Positive :165
False Negative :474
True Positive :227
Model has 0.787 accuracy
Model has 0.500 fbeta Score
```

Confusion Matrix

ACTUAL	PREDICTED	
	Left	Stayed
Left	227	165
Stayed	474	2134

Methodology

Data Pre-processing

In the data many values are categorical values, so we need to bring them to a scale. I will graphically represent each feature with the target variable (left) and with the help of minmaxscaler will apply logarithmic transformations of the features and will perform one hot encoding to change categorial values into numerical data. As all the categorical column has been converted to numerical column and all numerical column has been scaled. In the below section will split data (both features and label) into training and test sets. 80% will be used for training and 20% will be used for test.

Implementation

I implemented different methods on the model to get better results. I used fbeta score and accuracy along with roc curve means area under the curve.

Logistic regression

Logistic regression is another technique borrowed by machine learning from the field of statistics.

It is the go-to method for binary classification problems (problems with two class values).

I will import logistic regression from sklearn.linear_model. And will implement the classifier. In this no parameter is passed into the classifier.

Code snippet:

```
from  sklearn.linear_model import LogisticRegression

from sklearn.metrics import fbeta_score, accuracy_score, confusion_matrix

LRClassifier = LogisticRegression()

LRClassifier.fit(X_train, y_train)

LRPredict = LRClassifier.predict(X_test)

tn, fp, fn, tp = confusion_matrix(y_test, LRPredict).ravel()

print("In LogisticRegression\n")

print("True Negative :{}  " .format(tn))

print("False Positive :{}" .format(fp))

print("False Negative :{}" .format(fn))

print("True Positive :{}" .format(tp))

values( y_test,LRPredict)
```

output:

```
In LogisticRegression

True Negative :2134
False Positive :165
False Negative :474
True Positive :227
```

```
Model has 0.787 accuracy
Model has 0.500 fbeta Score
```

Adaboost:

AdaBoostClassifier is best used to boost the performance of decision trees on binary classification problems.

AdaBoost was originally called AdaBoost.More recently it may be referred to as discrete AdaBoost because it is used for classification rather than regression.

AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem.

Adaboost classifier imported from sklearn.ensemble library and following parameter were used. base_estimator=None n_estimators=50, learning_rate=1.0 random_state=0

Code snippet:

```
from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import fbeta_score, accuracy_score, confusion_matrix

ABClassifier = AdaBoostClassifier(base_estimator=None, n_estimators=50, learning_rate=1.0, random_state=0)

ABClassifier.fit(X_train, y_train)

adaboostPredict = ABClassifier.predict(X_test)

tn, fp, fn, tp = confusion_matrix(y_test, adaboostPredict).ravel()

print("In AdaBoost\n")

print("True Negative :{}  " .format(tn))

print("False Positive :{}" .format(fp))

print("False Negative :{}" .format(fn))

print("True Positive :{}" .format(tp))


values(y_test,adaboostPredict)
```

output:

```
In AdaBoost

True Negative :2247
False Positive :52
False Negative :59
True Positive :642
Model has 0.963 accuracy
Model has 0.923 fbeta Score
```

Gradient Boost:

Gradient boosting is a machine learning technique for regression and classificati on problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It allows for the optimization of arbitrary differentiable loss functions.

Import Gradient Boosting from sklearn.ensemble library.
Parameter used is Random state = 0

Code snippet:

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import fbeta_score, accuracy_score, confusion_matrix
GBClassifier = GradientBoostingClassifier(random_state=0)
GBClassifier.fit(X_train, y_train)
GBPredict = GBClassifier.predict(X_test)
tn, fp, fn, tp = confusion_matrix(y_test, GBPredict).ravel()
print("In gradientBoosting\n")
print("True Negative :{}  " .format(tn))
print("False Positive :{}" .format(fp))
print("False Negative :{}" .format(fn))
print("True Positive :{}" .format(tp))

values( y_test,GBPredict)
```

output:

```
In gradientBoosting

True Negative :2279
False Positive :20
False Negative :45
True Positive :656
```

```
Model has 0.978 accuracy
Model has 0.963 fbeta Score
```

SGD classifier:

Stochastic Gradient Classifier is known for efficiency and ease of implementation. Since it is sensitive to feature scaling and requires lot of parameter tuning we will like to check fbeta and accuracy to make judgment to use for fine tuning.

Will import SGDClassifier from sklearn.linear_model.

Model is trained with below parameter. Shuffle = True

Code snippet:

```
from sklearn.linear_model import SGDClassifier

from sklearn.metrics import fbeta_score, accuracy_score, confusion_matrix

SGD_Classifier = SGDClassifier(shuffle = True,loss='log')

SGD_Classifier.fit(X_train, y_train)

SGD_Predict = SGD_Classifier.predict(X_test)

print("In SGDclassifier\n")

tn, fp, fn, tp = confusion_matrix(y_test, SGD_Predict).ravel()

print("In AdaBoost\n")

print("True Negative :{}  " .format(tn))

print("False Positive :{}" .format(fp))

print("False Negative :{}" .format(fn))

print("True Positive :{}" .format(tp))

values( y_test, SGD_Predict)
```

output:

```
In SGDclassifier

True Negative :2013
False Positive :286
False Negative :388
```

```
True Positive :313
Model has 0.775 accuracy
Model has 0.505 fbeta Score
```

SVC :

“Support Vector Machine” (SVM) is a supervised machine learning algorithm w hich can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data ite m as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Supp ort Vectors are simply the co-ordinates of individual observation. Support Vect or Machine is a frontier which best segregates the two classes (hyper-plane/ lin e). SVM does not directly provide probability estimates. Uses subset of training points in decision functions and performs well even if number of dimension are greater than number of samples.

Model is fetched from library sklearn.svm
Mo parameter is used for classifier.

Code snippet:

```
from sklearn.svm import SVC
from sklearn.metrics import fbeta_score, accuracy_score, confusion_matrix


SVClassifier = SVC(random_state=0,probability=True)
SVClassifier.fit(X_train, y_train,)

SVPredict = SVClassifier.predict(X_test)
print("In SVC")
tn, fp, fn, tp = confusion_matrix(y_test, SVPredict).ravel()
print("True Negative :{}  " .format(tn))
print("False Positive :{}" .format(fp))
print("False Negative :{}" .format(fn))
print("True Positive :{}" .format(tp))

values(y_test,SVPredict)
```

output:

```
In SVC
True Negative :2245
False Positive :54
False Negative :347
True Positive :354
Model has 0.866 accuracy
Model has 0.759 fbeta Score
```

Refinement:

Adaboost classifier has number of hyper parameters on which model can be trained. To find the optimized parameter I used Grid Search method where we provide list of parameters in dictionary and model runs and score the model on different parameters combination and optimized the model.

In the above section we have accuracy and fbeta scores for all the techniques and the scores of SVM and SGD are lower than adaboost.

We have accuracy score and F Beta Score for all the Classification model on which we train the dataset. Given that F Beta Score and accuracy for SVM and Stochastic Gradient Classifier is lower than that of ensemble Learning. I decided to go with the ensemble learning method.

To avoid overfitting, I have decided to use validation set. In this case I use KFold validation, I have divided the training set into 10 blocks of dataset and job will run for 10 iterations where at each iteration one set will be validation set and other nine data set will be used for training till we have validated with all the 10 blocks of data set.

As there is never enough data to train your model, *removing a part of it for validation poses a problem of underfitting.* ***By reducing the training data***, ***we risk losing important patterns/ trends in data set, which in turn increases error induced by bias.*** So, what we require is a method that provides ample data for training the model and also leaves ample data for validation. K Fold cross validation does exactly that. In **K Fold cross validation**, the data is divided into k subsets. Now the holdout method is repeated k times, such that ***each time, one of the k subsets is used as the test set/ validation set and the other k-1 subsets are put together to form a training set***. The *error estimation is averaged over all k trials to get total effectiveness of our model*. As can be seen, every data point gets to be in a validation set exactly once, and gets to be in a

training set *k-1*times. ***This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in validation set.*** Interchanging the training and test sets also adds to the effectiveness of this method. **As a general rule and empirical evidence, K = 5 or 10 is generally preferred**, but nothing's fixed and it can take any value.

Adaboost classifier has number of hyper parameters on which model can be trained. To find the optimized parameter I used Grid Search method where we provide list of parameters in dictionary and model runs and score the model on different parameters combination and optimized the model.

```
Best classifier is: AdaBoostClassifier(algorithm='SAMME.R', base_estima
tor=None,
          learning_rate=0.7, n_estimators=90, random_state=0)
```

Accuracy score for benchmark model : 0.787

Fbeta score for benchmark model : 0.500

Unoptimized model

```
Accuracy score on testing data: 0.9630
F-score on testing data: 0.9232
```

Optimized Model

```
Final accuracy score on the testing data: 0.9657
Final F-score on the testing data: 0.9276
```

Results

Model Evaluation and Validation

As we know that the accuracy score on testing data for unoptimized model is 0.9630 and the fbeta score on testing data is 0.9232 and for optimised model. Final accuracy score on the testing data is 0.9657 and fbeta score is 0.9276.

Adaboost classifier has been chosen for prediction using Grid Search method. It has below parameter as best estimator.

AdaBoostClassifier (algorithm='SAMME.R', base_estimator=None, learning_rate=0.95, n_estimators=125, random_state=0)

K Fold mechanism has made model to test for unseen data in different iteration which makes it more robust and will perform well with unseen data.

Data:

```
'split0_train_score': array ([ 0.91825613, 0.92292721,  0.91954916,  0.
91965188,  0.91887482,
        0.92186162, 0.92267681, 0.92399346,  0.92079518,  0.9221487 ,
        0.92093967, 0.92135007,  0.92009401,  0.91993119,  0.92276012,
        0.92032647]),
 'split1_train_score': array([ 0.91957167,  0.92016872,  0.91831954,  0
.91860647,  0.92076503,
        0.91969721,  0.91774622,  0.91920375,  0.91678365,  0.91714821,
        0.91761319,  0.91922657,  0.91897388,  0.9194102 ,  0.92064113,
        0.92016872]),
'split2_train_score': array([ 0.92054644,  0.91922987,  0.91768936,  0.
9180315 ,  0.91800706,
        0.91714353,  0.91782326,  0.91847104,  0.91876817,  0.9188956 ,
        0.92031778,  0.91947257,  0.91939152,  0.91782326,  0.91656874,
        0.91743119]),
 'split3_train_score': array([ 0.91872096,  0.91826359,  0.91897621,  0
.91782326,  0.91701343,
        0.91763513,  0.91734893,  0.91573517,  0.91916097,  0.91902263,
        0.91952681,  0.91771411,  0.91861656,  0.91959007,  0.92035678,
        0.92079364]),
 'split4_train_score': array([ 0.9151751 ,  0.9173528 ,  0.91831954,  0
.91906615,  0.91722683,
        0.91803279,  0.91766724,  0.91691325,  0.91891681,  0.91873343,
        0.92029325,  0.91803279,  0.91902013,  0.92014704,  0.91881343,
        0.92064113]),
'split5_train_score': array([ 0.91552048,  0.91731669,  0.9159919 ,  0.
91536661,  0.91627543,
        0.91627543,  0.91627726,  0.91609589,  0.91521333,  0.91962065,
        0.91801117,  0.91770671,  0.91832762,  0.91840711,  0.92017134,
        0.92037269]),
'split6_train_score': array([ 0.92003439,  0.92001252,  0.92002831,  0.
91881514,  0.92046082,
        0.9199906 ,  0.91852722,  0.91850341,  0.9210423 ,  0.92033938,
        0.91967997,  0.919865  ,  0.91867351,  0.92060744,  0.92102161,
        0.92108372]),
'split7_train_score': array([ 0.91375291,  0.91495443,  0.91484242,  0.
91505043,  0.91419935,
        0.91473831,  0.91768718,  0.9189294 ,  0.91466253,  0.913678  ,
        0.914037  ,  0.91409521,  0.91311094,  0.91652457,  0.91758412,
        0.91729964]),
 'split8_train_score': array([ 0.91816758,  0.91920375,  0.91967495,  0
.91727741,  0.91899703,
        0.92006564,  0.9200874 ,  0.92066219,  0.92198026,  0.92364918,
        0.92239086,  0.9234622 ,  0.92011974,  0.92119693,  0.92138266,
        0.92257711]),
```

```
'split9_train_score': array([ 0.91631767,  0.91828794,  0.91704245,  0.
91673152,  0.91870045,
        0.91657604, 0.91946464,  0.9200124 ,  0.92070416,  0.91781993,
        0.92097051, 0.92010912,  0.91857388,  0.91951334,  0.91847064,
        0.92062257]),
 'mean_train_score': array ([ 0.91760633,  0.91877175,  0.91804338,  0.
91764204,  0.91805202,
        0.91820163,  0.91853062,  0.918852  ,  0.91880274,  0.91910557,
        0.91937802,  0.91910343,  0.91849018,  0.91931511,  0.91977706,
        0.92013169]),
'std_train_score': array([ 0.00217148,  0.00200874,  0.00159606,  0.001
46336,  0.00186854,
        0.0020524 ,  0.00171853,  0.00229181,  0.00238625,  0.00258505,
        0.00222251,  0.00236429,  0.00188611,  0.00131317,  0.00178889,
        0.00152432])}
```

Justification

The Logistic regression is used as benchmark model. FBeta score of benchmark model is reference and other model will be judging to perform better if their fbeta score will be greater than Logistic regression model. Accuracy, f-beta score and confusion matrix and will try to get better results in the ensemble learning models. With choice of different classifier in hand such as Stochastic Gradient Boosting, SVM, Ensemble learning. I decided to go with ensemble learning because this method increase the accuracy score of the models. Performance of ensemble learning will not be slow and it uses weak learner to identify the features and label it with result this was best choice.

I felt difficult in choosing between Gradient boosting and adaboost because both give good performance and scores. And if I choose Gradient boosting I may fell into overfitting, so I decided to choose adaboot classifier. And with the help of kfold cross validation and grid search I avoided overfitting

Accuracy score for benchmark model: 0.787

Fbeta score for benchmark model: 0.500

Unoptimized model

Accuracy score on testing data: 0.9630
F-score on testing data: 0.9232

Optimized Model

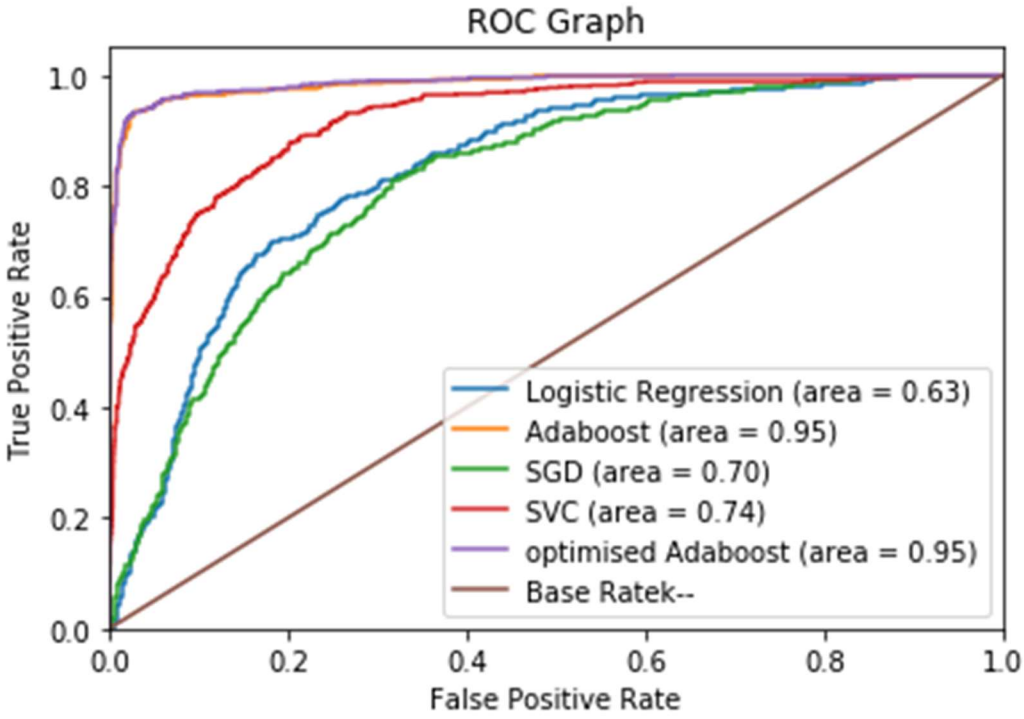
Final accuracy score on the testing data: 0.9657
Final F-score on the testing data: 0.9276

Conclusion

Free form visualization

The diagnostic performance of a test, or the accuracy of a test to discriminate diseased cases from normal cases is evaluated using Receiver Operating Characteristic (ROC) curve analysis.

In a Receiver Operating Characteristic (ROC) curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. A test with perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore, the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test. In the below graph optimised Adaboost shows better performance compared to others.



Reflection:

First of all, I load the csv into a data frame with the help of pandas library and after loading it with the help of NumPy library I calculate the number of employees who had already left and the length of the data set and the total number of employees who stayed and I find the null values in the data frame. After that I plot a heat map of all the features and come to know how the features are correlating with each other with the help of mathplotlib.pyplot library and after that I graphically represent each feature with the target variable (left) and with the help of minmaxscaler I apply logarithmic transformations of the features and I perform one hot encoding to change categorial values into numerical data. As all the categorical column has been converted to numerical column and all numerical column has been scaled. I split data (both features and label) into training and test sets. 80% will be used for training and 20% will be used for test. And classification model is evaluated on the basis of accuracy, precision, and recall and roc curve. Logistic regression

is used as benchmark model. FBeta score of benchmark model is reference and other model will be judging to perform better if their fbeta score will be greater than Logistic regression model. And with the help of ensemble learning methods I find the accuracy scores of the model and I also perform SGDC and GradientBoosting Classifier and find the accuracy of the models. After that I apply grid search and to tune the parameters and increase the performance. Once the model is trained will compare with the actual values with predicted values. HR Analytics data has various data which as per intuition has some features that could easily identified as decision-making columns such as satisfaction level, salary, number of hours spent. When these data were plotted on graph intuition was confirmed and it was seen that people with low salary were most to quit and people with salary prefer to stay. People having low satisfaction level score did quit the company. Those people who are spending more than 300 hours did quit the company. Interestingly I thought that people must be eager for promotion and those not promoted in last 5 years will quit but this was not the case and this feature seems to have almost no impact on decision. In my data AdaBoost classifier of ensemble learning use to train model and it performed better than the Benchmark model (Logistic Regression). Further to avoid overfitting a K Fold mechanism was used to train model so that it can have variety of data and will know to make decision based on features.

Improvement:

Since the given task is a Binary Classification problem, it is very wise enough to apply Logistic Regression. But there are also certain algorithms like Decision Trees and Support Vector Machines(SVM) that can also be applied that could potentially show better performances upon the raw data we process.

References:

- https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/

- https://machinelearningmastery.com/logistic-regression-for-machine-learning/
- https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/
- https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f
- http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
- http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html
- https://machinelearningmastery.com/assessing-comparing-classifier-performance-roc-curves-2/
- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
- http://scikit-learn.org/stable/auto_examples/model_selection/plot_multi_metric_evaluation.html
- http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
- https://www.techopedia.com/definition/28334/human-resources-analytics-hr-analytics
- https://www.medcalc.org/manual/roc-curves.php