Machine Learning Engineer Nanodegree

Capstone Project

Arshad Khurshid
11/11/2017

Definition

Project Overview

Machine Learning Engineer Nanodegree course has emphasized on different type of Supervised and Unsupervised Learning. Supervised Learning deals with Regression and Classification. Classification is being implemented across all the fields for different applications and are very prominent in Cyber Security and Medical diagnosis. With boom in data, Classification model has gained momentum to understand the cause of change in behavior in consumer. This helps business to work on data and not to take decision just on intuition.
A company invest lot on employee to train them and make them ready for next generation business. Once you invest in skill enhancement of an employee you need to use it for benefit of business. Employee may be agitated even if they are being paid well as human have aspiration and if aspiration is fulfilled then they perform to their maximum capability. I was looking for dataset which has features that can be reason for employee to quit and Kaggle gave me one. Retaining an employee means retaining knowledge and they are the one who grooms the people working in one level down thus helping to increase and knowledge base for all.

Dataset Link: https://www.kaggle.com/ludobenistant/hr-analytics

Problem Statement

Managing people in a company is challenging job. When it comes to attrition HR has a tough job to identify the root cause of attrition. Most of the time it happens that diagnosis based on intuition is wrong and hence the policy implemented has more damaging effect and situation worsen. What if HR knows in that particular employee is more probable of leaving and could take damage control action. What is they know why employee are leaving and features that is having more damaging effect on them. Since I need to emphasize on 2 groups Classification of employee into Left and Stayed and I treat it as Classification problem. The other is the reason why people are leaving. For this i will drop the decision feature (left) and will group features in different nodes and find the features that has most impact.

Features and description

• satisfaction_level :	Level of Satisfaction
• last_evaluation :	Time since Last performance Evaluation
• number_project :	Number of Project completed while at work
• average_montly_hours :	Average monthly hours at workplace
• time_spend_company :	Number of years spent in the company
• Work_accident :	Whether the employee had a workplace accident
• left :	Whether employee left the workplace or not
• promotion_last_5years:	Whether employee was promoted in last 5 years
• sales :	Department they work for
• Salary :	Relative level of Salary(high)

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. Some employee may have worked on big projects continued for long time whereas some may work for high number of small projects. Average monthly hours plays important role as employee putting too much time for work may have no time to relax and hence will burn out and would like to move on. Work accident plays important role as it concern with security of people and that may require people to quit. Left is the decision column which will used to predict. Promotion in last 5 years and salary should have same impact and hence I believe salary should be used and promotion to be dropped. Sales is department value and will be interesting to see which department is more impacted. But I do not think it impacts on decision. Salary an important column for employee to quit.

The dataset has total of 14999 rows and 10 columns. I will read the csv file using pandas library. Will analyze the data to check for any null values in any of columns and impact of each column on output. With this model i hope to analyze the employees who are more likely to exit and could do damage control exercise to retain them thus retaining talent and boosting company performance. Target variable (left) is imbalance dataset. It contains 3751 records of employee who have left the company. 11428 records of employee who stayed in the company. Here we are trying to predict employee who can quit. Dataset has been extracted from Kaggle.

Dataset Link: https://www.kaggle.com/ludobenistant/hr-analytics

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. Machine Learning model will help them to identify the people who could quit in future and the reason for quitting the jobs. So that a preventive measure is taken and employee could be retained. I plan to use Classification model of Supervised Learning techniques to learn from history data and when existing employee is pass to the model it could predict if employee will be moving on and damage control action will be taken. Couple of step will be taken for preprocessing of data. Will check for null value for feature. If available will fill with mean for numerical column and mode value for categorical column. Will do one hot encoding for categorical column. 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. I will use Ensemble machine learning model (AdaBoost /Gradient Boosting) to understand the employee moving on. Will finalize any one based on performance. I will also use Stochastic Gradient Descent Classifier (SGDC) and SVM to check their f score and if performing better than Ensemble then will use it in fine tune and create model. Since the goal of model is to understand why employees are moving on I will consider more interpretable model than highest prediction accuracy.

Metrics

Classification model will be evaluated on F Beta score and Accuracy, precision and recall. We can use F-beta score as a metric that considers both precision and recall:

$$F_\beta = (1+\beta_2) \cdot precision \cdot recall / ((\beta_2 \cdot precision) + recall)$$

In particular, when $\beta=0.5$, more emphasis is placed on precision. This is called the $F_{0.5}$ score (or F-score for simplicity).

Accuracy measures how often the classifier makes the correct prediction. It is the ratio of the number of correct predictions to the total number of predictions (the number of test data points).

Precision tells us what proportion of messages we classified as left, actually left. It is a ratio of true positives(predicted value for person left and actually person left) to all positives (all the person left company irrespective of true or false), in other words it is the ratio of
[True Positives/(True Positives + False Positives)]

Recall (sensitivity) tells us what proportion of employee that left were classified by us as left. It is a ratio of true positives (employee that left, and employee actually left) to all the employee left, in other words it is the ratio of [True Positives/ (True Positives + False Negatives)]

Analysis

Data Exploration

I have calculated total number of records available in dataset. Number of people moved out and number of people stayed. Percent of employees moved on. Total number of features available. To find null values in dataset and datatype of imported data.
By looking at the data type we can understand that there are two columns sales and Salary which is categorical column and rest other are numerical columns. Categorical column need to be one hot encoder.

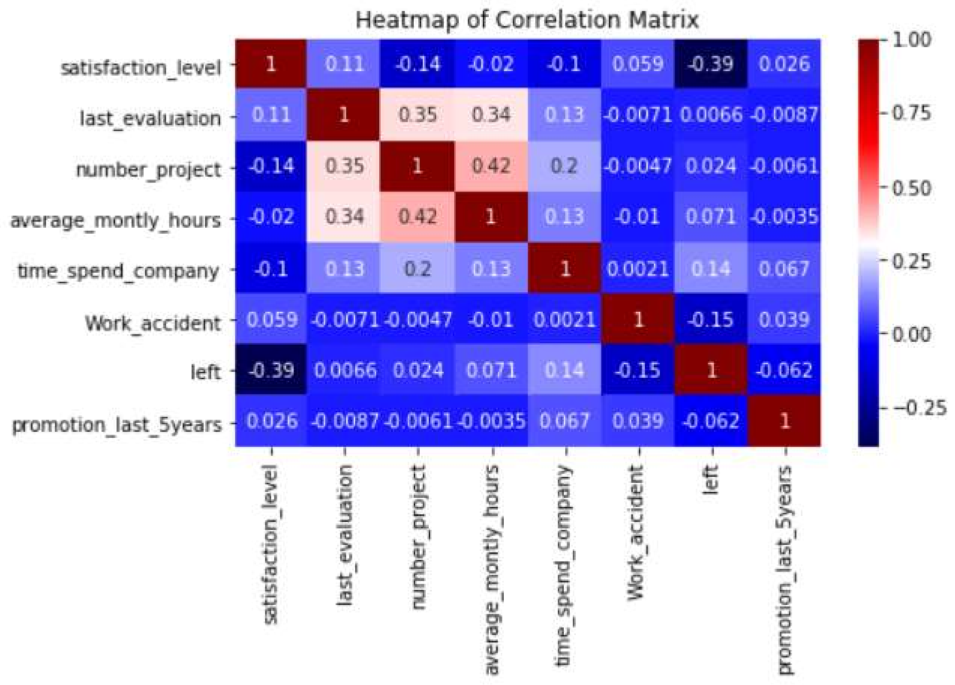
```
Total number of records: 14999
Total number of employee who have moved on: 3571
Total number of employee who have stayed: 11428
Percent of employee who have moved on: 23.81%
Total number of null rows in Dataframe: 0

 Datatype as below
satisfaction_level       float64
last_evaluation          float64
number_project             int64
average_montly_hours       int64
time_spend_company         int64
Work_accident              int64
left                       int64
promotion_last_5years      int64
sales                     object
salary                    object
dtype: object
```

Exploratory Visualization

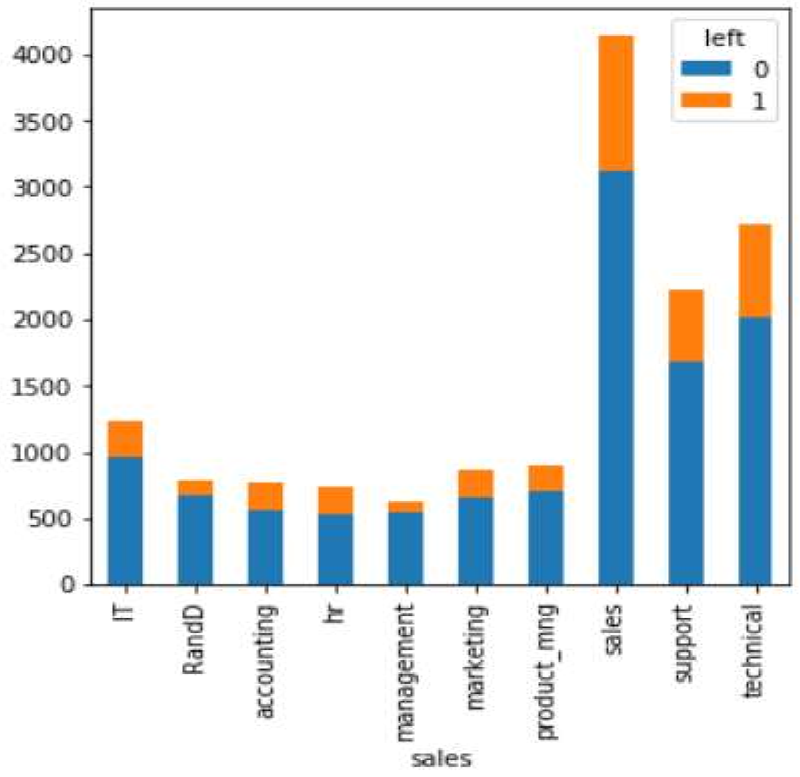
I have first generated Heatmap of correlation matrix understand the relation of each column on the other column. It states that average monthly hours and number of projects are highly correlated and hence using any one column for machine learning model should good.

```
Text(0.5,1,'Heatmap of Correlation Matrix')
```

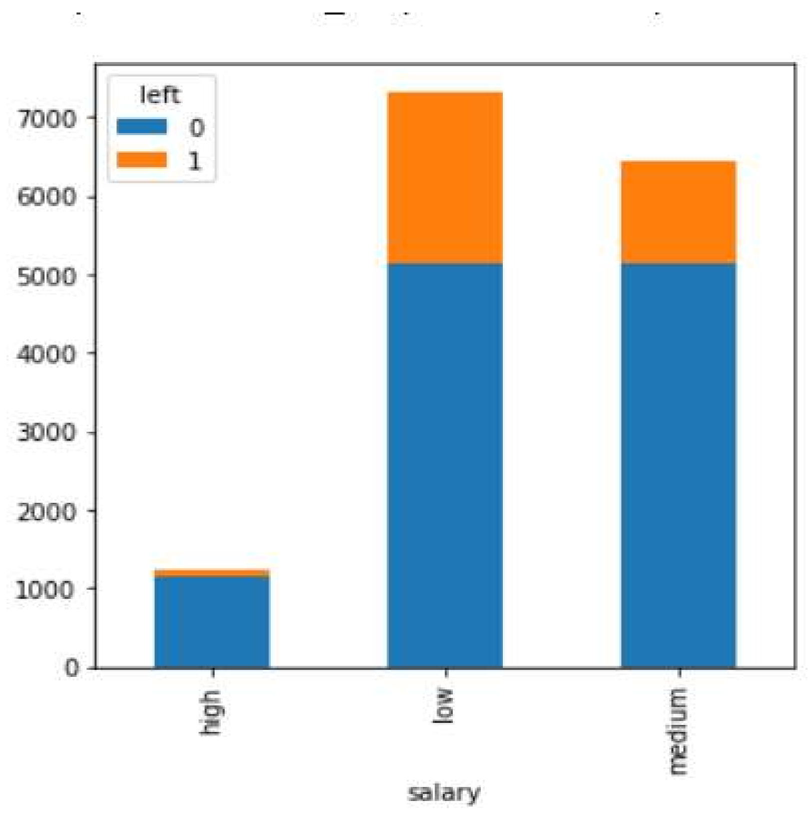


	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
satisfaction_level	1	0.11	-0.14	-0.02	-0.1	0.059	-0.39	0.026
last_evaluation	0.11	1	0.35	0.34	0.13	-0.0071	0.0066	-0.0087
number_project	-0.14	0.35	1	0.42	0.2	-0.0047	0.024	-0.0061
average_montly_hours	-0.02	0.34	0.42	1	0.13	-0.01	0.071	-0.0035
time_spend_company	-0.1	0.13	0.2	0.13	1	0.0021	0.14	0.067
Work_accident	0.059	-0.0071	-0.0047	-0.01	0.0021	1	-0.15	0.039
left	-0.39	0.0066	0.024	0.071	0.14	-0.15	1	-0.062
promotion_last_5years	0.026	-0.0087	-0.0061	-0.0035	0.067	0.039	-0.062	1

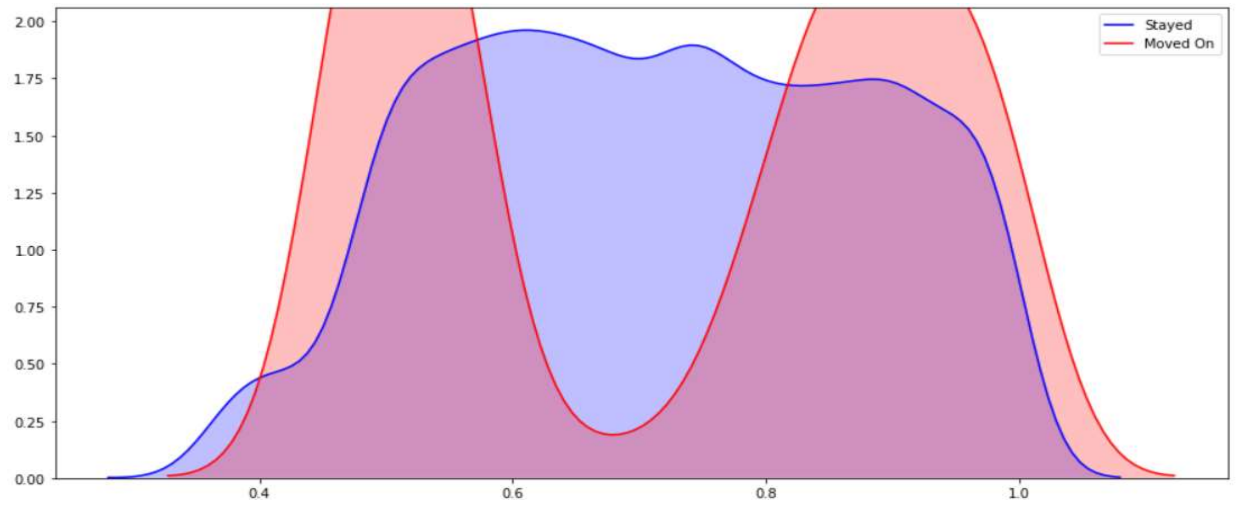
I would like to find out which department are most affected to understand if it is departmental problem or attrition is across the board. Below graphs confirms which department is most impacted. One of features to look into it.



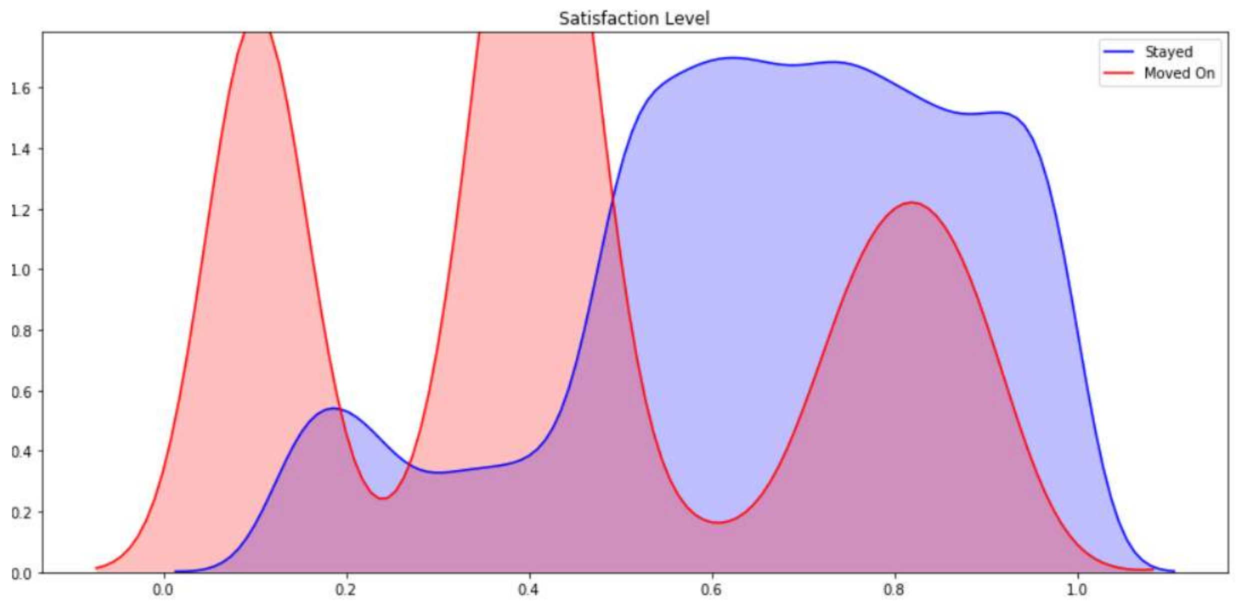
As per my intuition I want to check if people are low paid is impacted and will like to move on for better opportunity and below graphs confirms that most people with low salary are one to quit. Moderate number of median salary have quit job and very few with high salary.



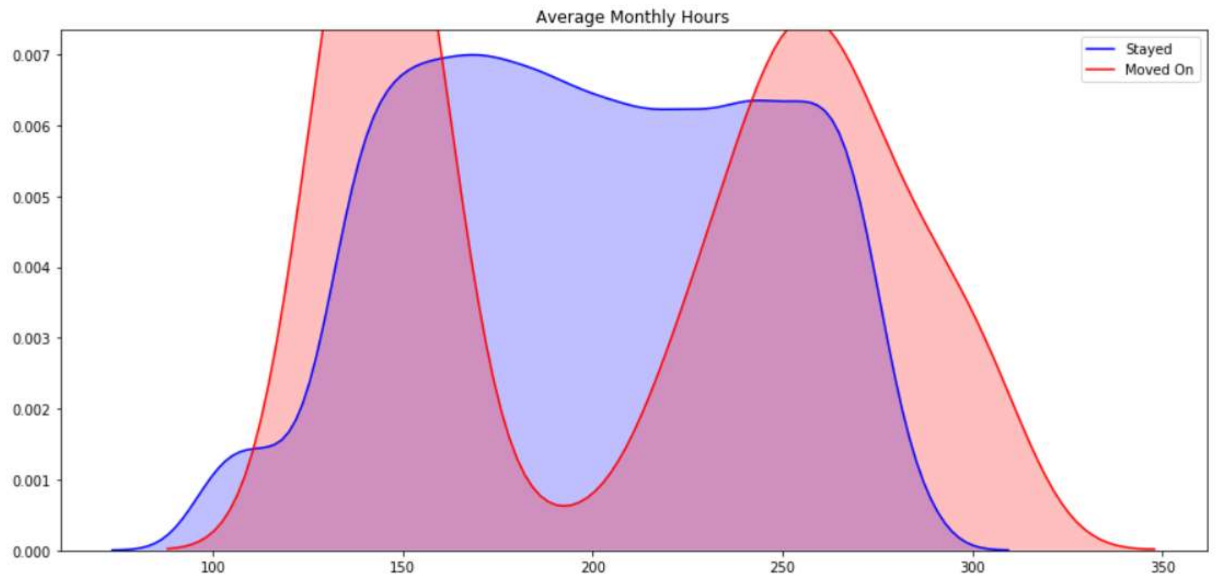
I tried to plot graph on last evaluation and left column to understand the impact of last evaluation on decision to stay in a company and it was found that evaluation score between 0.4 to 0.6 and 0.8 to 1.0 were the one leave company. Giving us view that one with mid-level of evaluation usually stays.



I tried to plot graph with satisfaction level and left column to check what is people are not satisfied. Do they make decision to move on and graphs prove that indeed they are one to move out.



Another column which could impact on people was average monthly hours in company and guess was that if a person spends too much time he may be burn out and would quit. This is what shown in below graph.



Algorithm and Techniques

One of challenge for machine learning is to choose which technique will work for the problem dataset. 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 classifiers 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. 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. These are models that achieve accuracy just above random chance on a classification problem.

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:

$$\text{weight}(xi) = 1/n$$

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:

$$\text{error} = (\text{correct} - N) / N$$

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:

$$\text{error} = \text{sum}(w(i) * \text{terror}(i)) / \text{sum}(w)$$

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

As a benchmark model I have used Logistic Regression classification model and Confusion matrix, accuracy and F Beta score is calculated so that any improvement will add to the performance of model. F Beta will be used for measuring performance, Accuracy is for support purpose only.

Logistic Regression

Confusion Matrix

	ACTUAL	
PREDICTED	Left	Stayed
Left	227	474
Stayed	165	2134

Accuracy: 0.787
F Beta Score: 0.500

Methodology

Data Preprocessing

It was observed that different numerical columns in in data frame needs to be brought down to scale. In our case we used Logarithmic scale where minimum value for is) and maximum value is 1. It is done using sklearn preprocessing library MinMaxScaler.

We had two categorical column. Sales and Salary. Sales had different value such as IT, R&D, HR, sales etc which are different department name. slary has categories divided into low, medium and High. These values were converted to different column with value 0 and 1. So total feature where increased from 10 to twenty.

Data is now split in 80 20 ratio using random state 0 and shuffle = true so that we get good mix of data.
For training set we have variable as X_train and y_train.
For testing set we have data X_test and y_test.
Final step is to print out number of rows for each training and test set to know how much data contained in each training and test dataset.

Implementation

I tried different method to check which model gives better prediction on basis of F Beta and accuracy.

Adaboost Classifier

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


EAB_Classifier = AdaBoostClassifier(base_estimator=None, n_estimators=50, learning_rate=1.0,
random_state=0)
EAB_Classifier.fit(X_train, y_train)

EAB_Predict = EAB_Classifier.predict(X_test)

nm='AdaBoost Classifier'
calculate_performance( y_test, EAB_Predict, nm)
```

Output:

Confusion Matrix

	ACTUAL	
PREDICTED	Left	Stayed
Left	642	52
Stayed	59	2247

Accuracy: 0.963
F Beta Score: 0.923

Gradient Boosting Classifier

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions.
Import Gradient Boosting from sklearn.ensemble library. Parmaeter used is
Random state = 0

Code Snippet:

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import fbeta_score, accuracy_score, confusion_matrix


GBC_Classifier = GradientBoostingClassifier(random_state=0)
GBC_Classifier.fit(X_train, y_train,)

GBC_Predict = GBC_Classifier.predict(X_test)

nm='Gradient Boosting Classifier'
calculate_performance( y_test, GBC_Predict, nm)
```

Output:

Confusion Matrix

	ACTUAL	
PREDICTED	Left	Stayed
Left	656	20
Stayed	45	2279

Accuracy: 0.978
F Beta Score: 0.963

Stochastic Gradient Descent 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)
SGD_Classifier.fit(X_train, y_train)

SGD_Predict = SGD_Classifier.predict(X_test)

nm='Stochastic Gradient Descent Classifier'
calculate_performance( y_test, SGD_Predict, nm)
```

Output:

Confusion Matrix

	ACTUAL	
PREDICTED	Left	Stayed
Left	656	20
Stayed	45	2279

Accuracy: 0.775
F Beta Score: 0.297

Support Vector Machine

SVM are memory effective and are also effective in high dimension space. Uses subset of training points in decision functions and performs well even if number of dimension are greater than number of samples. SVM does not directly provide probability estimates. These are calculated using expensive five-fold cross validation.

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


SVC_Classifier = SVC()
SVC_Classifier.fit(X_train, y_train)

SVC_Predict = SVC_Classifier.predict(X_test)

nm='Support Vector Machine Classifier'
calculate_performance( y_test, SVC_Predict, nm)
```

Output:

Confusion Matrix

	ACTUAL	
PREDICTED	Left	Stayed
Left	354	347
Stayed	54	2245

Accuracy: 0.866
F Beta Score: 0.759

Refinement

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. Adaboost and Gradient Boosting classifier both performs almost at the same level. I choose Adaboost classifier for final model.

We have training set and test set for training and testing model. To avoid overfitting I have decided to use validation set as well. In this case I use KFold validation, I have divided the training set into10 block of dataset and job will run for 10 iteration 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.

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.

Accuracy and Score as below.

Model	Benchmark model	Un optimized model	Optimized model
Accuracy	.787	.963	.965
F Score	.500	.923	.926

Results

Model Evaluation and Validation

K Fold validation has been used for model validation. It has 10 iteration and with 10 subset of training set known as validation set. Accuracy and FBeta score of it is provided as below.

Model	Benchmark model	Un optimized model	Optimized model
Accuracy	.787	.963	.965
F Score	.500	.923	.926

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 as below:

```
'split0_train_score': array([ 0.9210218 ,  0.92028364,  0.92278866,  0.92233388,  0.92
008371,
        0.92186162,  0.92309492,  0.92267681,  0.92223877,  0.9221487 ,
        0.92087844,  0.92093967,  0.92058893,  0.91993119,  0.92155488,
        0.92276012]),
'split1_train_score': array([ 0.91850289,  0.91725215,  0.91852661,  0.91712062,  0.92
084701,
        0.91969721,  0.91839925,  0.91774622,  0.91850289,  0.91714821,
        0.91951334,  0.91761319,  0.92177457,  0.9194102 ,  0.92103206,
        0.92064113]), 'split2_train_score': array([ 0.91701343,  0.91936877,  0.917427
61,  0.91993119,  0.9162798 ,
        0.91714353,  0.91776856,  0.91782326,  0.91965758,  0.9188956 ,
        0.91951773,  0.92031778,  0.91920696,  0.91782326,  0.91876817,
        0.91656874]),
'split3_train_score': array([ 0.91656828,  0.91722244,  0.9181911 ,  0.92079518,  0.91
753547,
        0.91763513,  0.91836735,  0.91734893,  0.91897621,  0.91902263,
        0.91766455,  0.91952681,  0.91890614,  0.91959007,  0.92015388,
        0.92035678]),
'split4_train_score': array([ 0.91998439,  0.91847064,  0.9204776 ,  0.9196164 ,  0.91
751368,
        0.91803279,  0.91797788,  0.91766724,  0.91681014,  0.91873343,
        0.92088756,  0.92029325,  0.91834347,  0.92014704,  0.91990329,
        0.91881343]),
'split5_train_score': array([ 0.91658849,  0.91830371,  0.91692668,  0.91752417,  0.91
83114 ,
        0.91627543,  0.91814504,  0.91627726,  0.91687422,  0.91962065,
        0.91869348,  0.91801117,  0.91785937,  0.91840711,  0.91963449,
        0.92017134]),
'split6_train_score': array([ 0.9175104 ,  0.91883854,  0.92007216,  0.91795193,  0.91
990234,
        0.9199906 ,  0.92038242,  0.91852722,  0.91879168,  0.92033938,
        0.92073219,  0.91967997,  0.91983556,  0.92060744,  0.92056771,
        0.92102161]),
'split7_train_score': array([ 0.91613456,  0.91584929,  0.91473831,  0.91414141,  0.91
600374,
        0.91473831,  0.91880342,  0.91768718,  0.91321545,  0.913678  ,
        0.91409521,  0.914037  ,  0.91479473,  0.91652457,  0.91577644,
        0.91758412]),
'split8_train_score': array([ 0.91960784,  0.91876663,  0.9178744 ,  0.91897388,  0.91
988131,
        0.92006564,  0.9200874 ,  0.9200874 ,  0.92171125,  0.92364918,
        0.92146311,  0.92239086,  0.91954023,  0.92119693,  0.92039216,
        0.92138266]),
'split9_train_score': array([ 0.91864433,  0.91856501,  0.91678311,  0.91974155,  0.92
076503,
        0.91657604,  0.92133872,  0.91946464,  0.91965264,  0.91781993,
        0.92256809,  0.92097051,  0.91849432,  0.91951334,  0.92166797,
        0.91847064]),
'mean_train_score': array([ 0.91815764,  0.91829208,  0.91838062,  0.91881302,  0.9187
1235,
        0.91820163,  0.9194365 ,  0.91853062,  0.91864308,  0.91910557,
        0.91960137,  0.91937802,  0.91893443,  0.91931511,  0.9199451 ,
        0.91977706]),
'std_train_score': array([ 0.00157508,  0.00118105,  0.00214111,  0.00214942,  0.00172
249,
        0.0020524 ,  0.00165886,  0.00171853,  0.0024595 ,  0.00258505,
        0.00227463,  0.00222251,  0.00175766,  0.00131317,  0.00161967,
        0.00178889]
```

Since best estimator has been used and validated using K Force validation it is trusted with results.

Justification

I have created Logistic Regression as benchmark model which has F Beta Score of 0.5. Since accuracy is not a good measure to measure performance of model as it can trick to be wrong landing us into Accuracy paradox.

With choice of different classifier in hand such as Stochastic Gradient Boosting, SVM, Ensemble learning. I decided to go with ensemble learning as it is good with small amount of data. 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.

Decision between Gradient Boosting and Adaboost classifier was tough to make with Gradient Boosting performing slightly better than Adaboost. I did not want model to fall in trap of overfitting and consider it safe to go with Adaboost classifier.

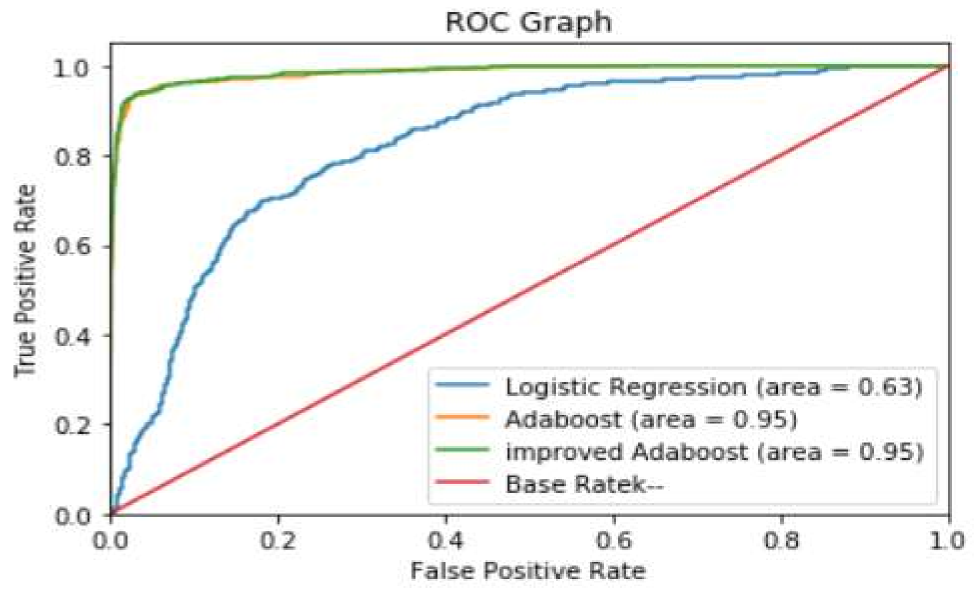
Further using K Fold validation and Grid Search I avoided overfitting and further tune Adaboost hyper parameter to optimize the model.

Model	Benchmark model	Un optimized model	Optimized model
Accuracy	.787	.963	.965
F Score	.500	.923	.926

Conclusion

Free Form Visualization

I plotted ROC graph to visually identify how model has been improved and fine-tuned to predict value.



Reflection

First Step to start of any machine learning model design is to analyze data and explore the possible combination which could impact decision. In this process we identified number of rows and features available in dataset. What percentage of people quit company? Data type of all features available. This help us to identify numerical and categorical columns. Are there any null columns if se we need to address it? We did not have any null column.

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.

Before proceeding to train model it was required that we scale all the numerical column. In our case I thought to scale it between 0 and 1. It was done using MinMaxScaler. Next Categorical column needs to be converted into n -1 number of column of values present in each column. So that it can converted to numerical column with values either 0 or 1. 1 for features having this value and 0 for absent of this value. This is called onehotcoding.

Now data is ready as input for model to learn. Dataset is split into 2 parts in ratio of 8:20. 80 parts for training and 20 parts for testing the model once it has been trained. One should use option of shuffle as True so that data is well shuffled and model can learn on random set of data.

As data is ready one should train it using the simplest classifier available. In our case Logistic regression was used so as calculate f Score and accuracy for benchmark. Now we have to improve our model to perform better than benchmark model.

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. Grid Search mechanism is used to fine tune the model by using hyper parameter.

In final step roc curve is used to show the performance improvement from base to benchmark model to Adboost and Optimized Adaboost classifier.