IBM HR ANALYTICS EMPLOYEE ATTRITION & PERFORMANCE

ABSTRACT

A comprehensive report on IBM HR Analytics employee attrition and performance attrition which exhaustively uses data visualization and an array of machine learning algorithms to predict primary reasons for attrition.

SHIVAM NEGI

Graduate Student and Data Scientist, Northeastern University

Abstract

This report comprises of an exhaustive research on Kaggle's "IBM HR Analytics Employee Attrition & Performance" data. The study consists of identifying the root cause of the problem of attrition, finding several indicators and features which are leading to such issues and some very robust and expansive visualizations which adhere to this problem faced by IBM Human Resource team. The study also entails Keras deep learning algorithm and Decision trees which points towards some of the leading features which contribute towards attrition. In the end the report concludes with some of the recommendations regarding features which need special attention by the department and practices which can be adopted for better performance. Business is not just doing deals; business is having great products, doing great engineering, and providing tremendous service to customers. Finally, business is a cobweb of human relationships!

Introduction

Employee attrition is one of the biggest issue which has gripped some of the major top firms in United States. It's an issue that has been puzzling the Human Resource Managers of various companies for a long time now. In this project, it is tried to analyse what factors lead to employee retention in companies, and what factors influence them the most. It uses a dataset that is published by the Human Resource department of IBM.

From the obtained data of 1470 employees, it is examined that how many of the employees are retained by the company. It is observed that 237 employees have been retained whereas 1233 employees have been let go of.

IBM Attrition Data

The key to success in any organization is attracting and retaining top talent. As an analyst, one of the tasks is to determine which factors keep employees at IBM and which prompt others to leave.

Here are some of the salient features of the data:

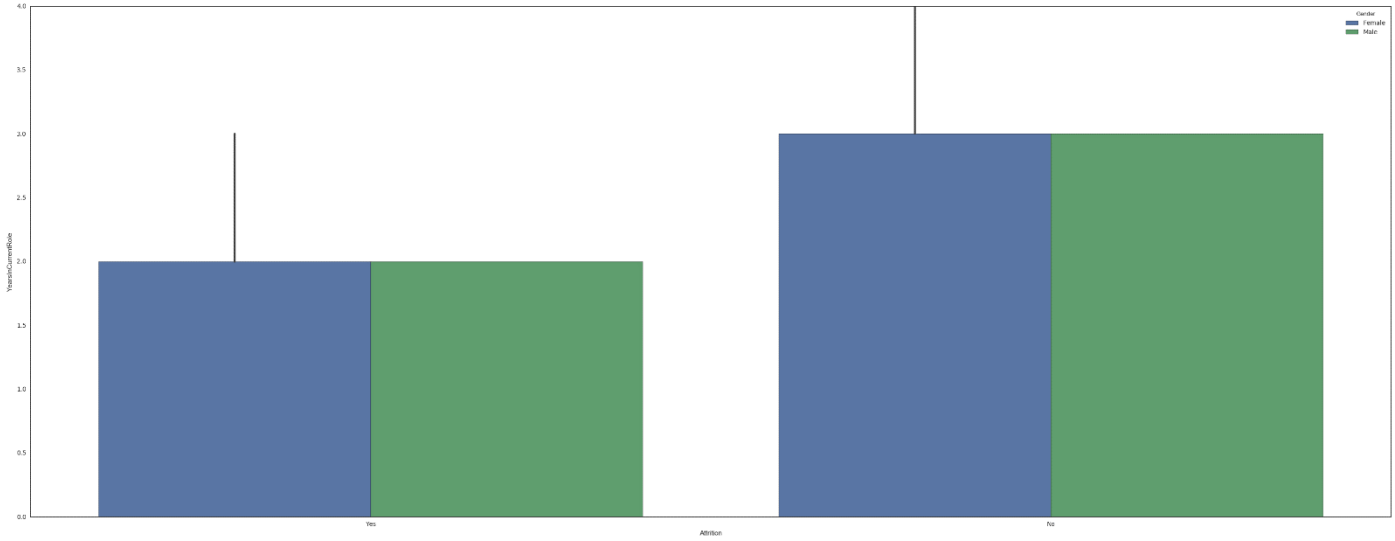
- Total Records: 1470 employee's record
- We can see that 237 employees have been retained whereas 1233 employees have been let go of.
- The average age of the retained employee is around 33 years of age, which is quite young and early in the employee's career. While the average age for all the employees irrespective of retention is 36.923810.
- On a scale of 1 to 5, where: 1-below college, 2-college, 3-bachelors, 4-masters, 5-doctorate; mean is 2.912925, which means that they have attended college, and most of them have bachelors' degrees.

Comprehensive Analysis and Visualization

1. Attrition Vs Years in Current Role

x='Attrition', y= Years in Current Role’

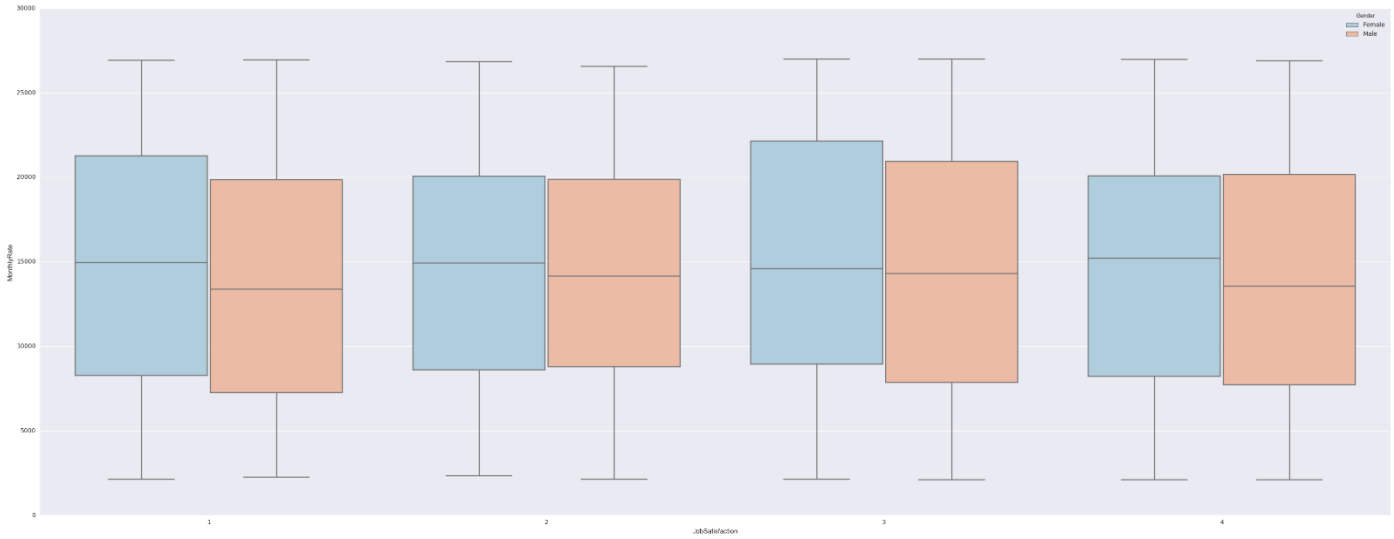
hue= 'Gender'



It can be seen and analyzed that, those who have left the company have surely had worked at their current position for a long time compared to those who have retained. Though there is no disparity on the basis of gender in each subset which is formed.

2. Job Satisfaction vs Monthly Rate

x='Job Satisfaction', y='Monthly Rate'



Job Satisfaction

1 'Low'

2 'Medium'

3 'High'

4 'Very High'

Here we can see that Job Satisfation of female counterpart is relative higher irrespective of the Monthly rate. This also says that, females who are paid well are having relatively higher job satisfaction.

3. Job Satisfaction vs Distance From Home

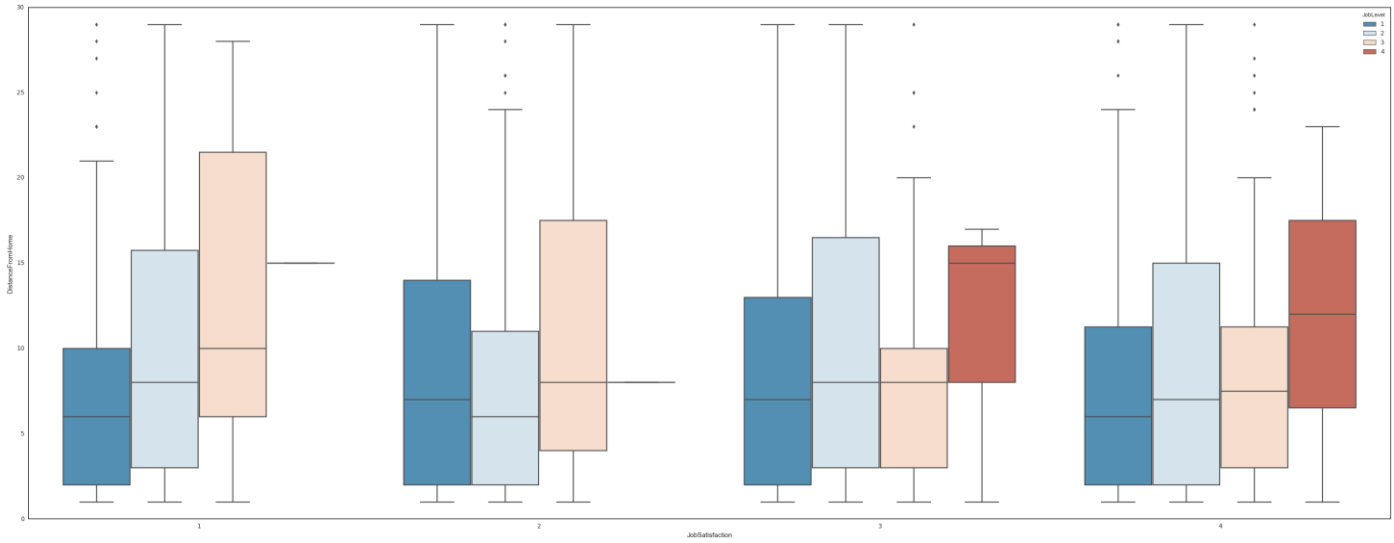
Job Satisfaction

1 'Low'

2 'Medium'

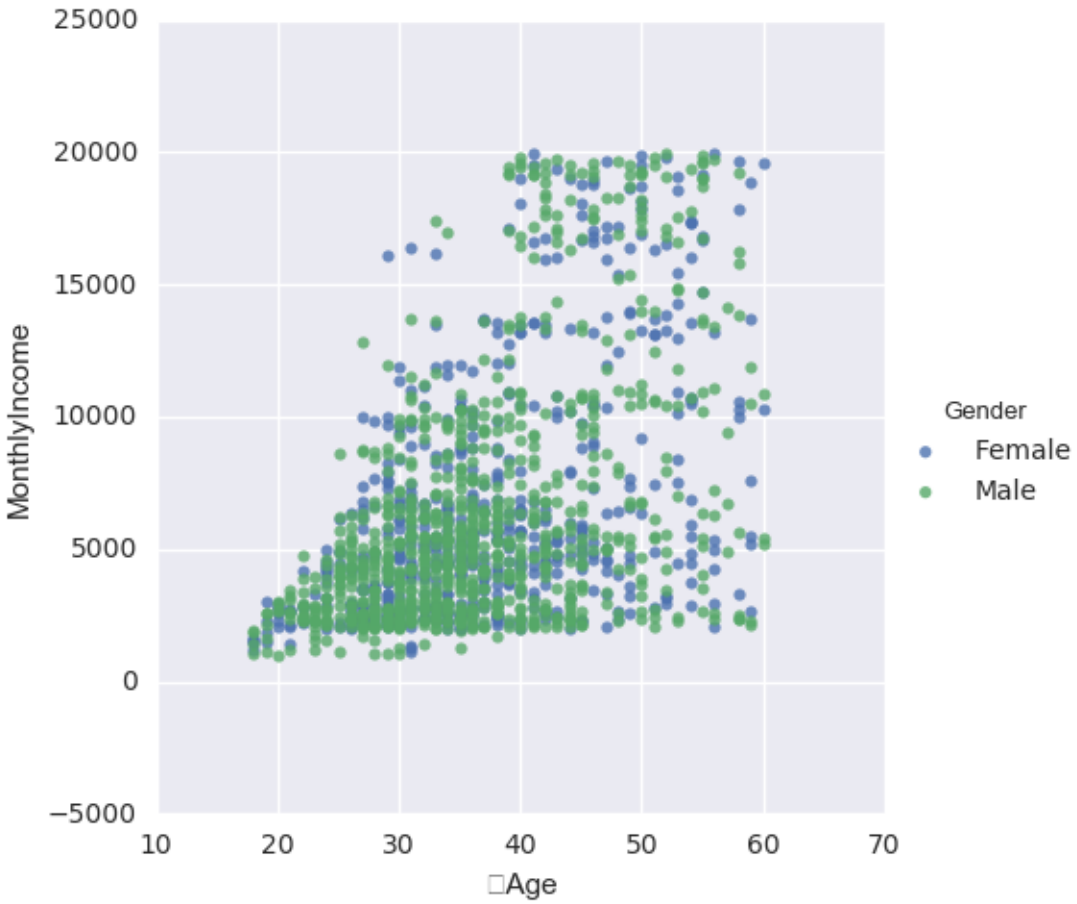
3 'High'

4 'Very High'



It can be analyzed that, as the distance increase, the job satisfaction is decreased. While when we have 4 different positions, several scenarios can be deciphered from examining the plots.

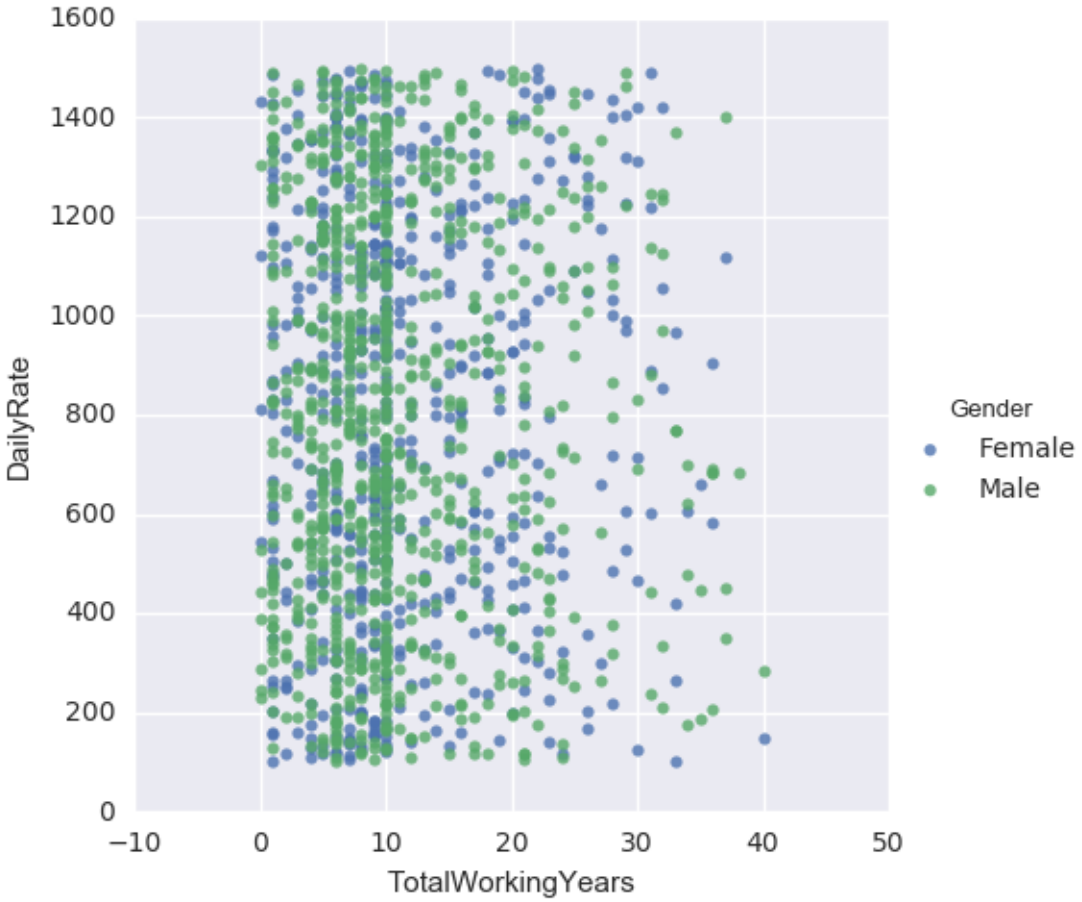
4. **Scatter Plot OF Age of Employees vs Monthly Income, grouping it by gender**



Following things can be analyzed:

- More men are working in the company as there are more green data points
- From the scatter spread getting less dense in above region, it can be seen that, a huge chunk of young population (below 50 years) are working in low paying jobs (below 10000$ per month)

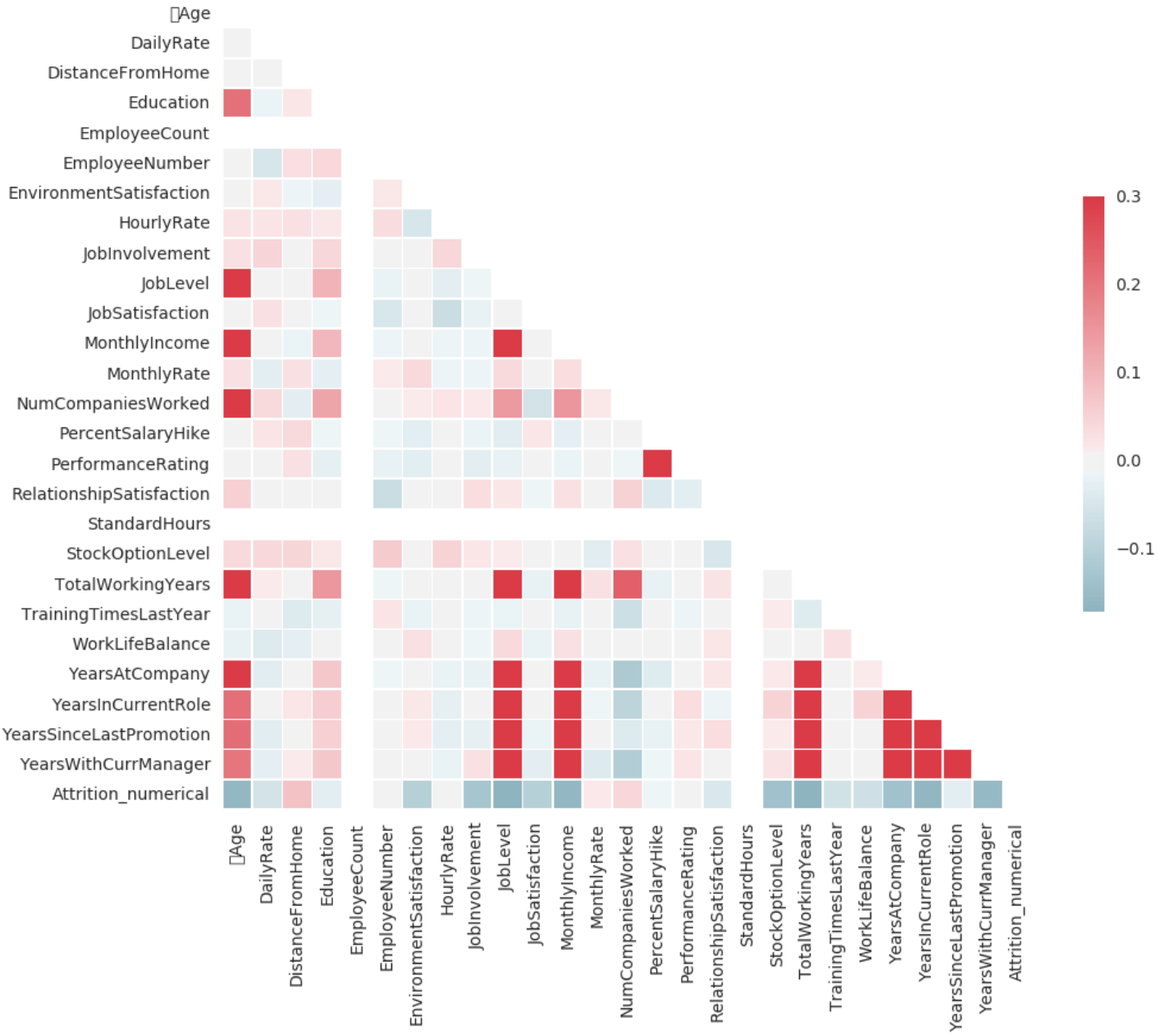
5. TotalWorkingYears vs DailyRate on Gender

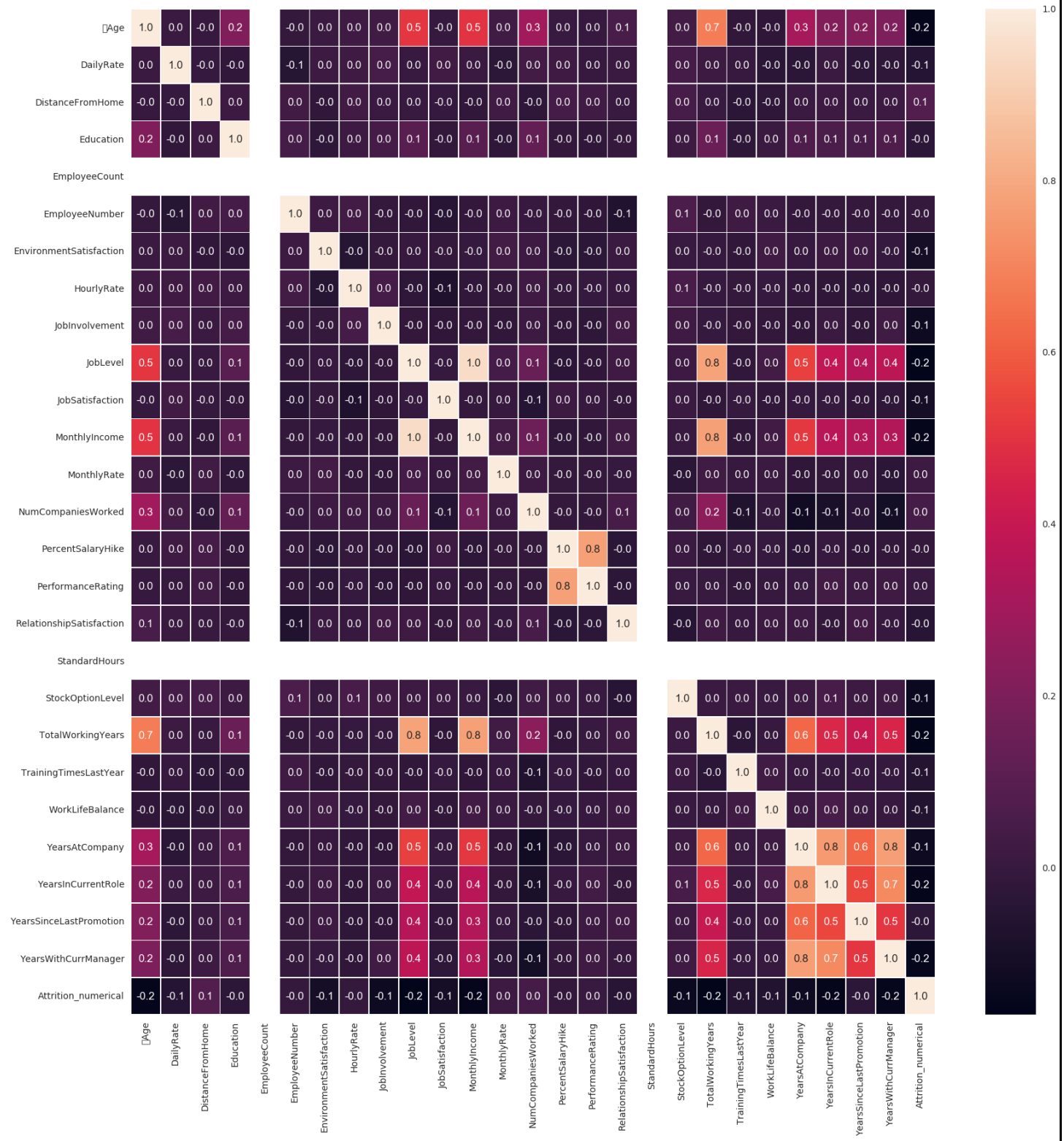


Following observations are being made:

- Most of the people are having a total experience of 10 years.

6. Correlation Plot- Matrix for feature selection




Age 1.0 0.0 -0.0 0.2 -0.0 0.0 0.0 0.0 0.5 -0.0 0.5 0.0 0.3 0.0 0.0 0.1 0.0 0.7 -0.0 -0.0 0.3 0.2 0.2 0.2 -0.2
DailyRate 0.0 1.0 -0.0 -0.0 -0.1 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.1
DistanceFromHome -0.0 -0.0 1.0 0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.1
Education 0.2 -0.0 0.0 1.0 0.0 -0.0 0.0 0.0 0.1 -0.0 0.1 -0.0 0.1 -0.0 -0.0 -0.0 0.0 0.1 -0.0 0.0 0.1 0.1 0.1 0.1 -0.0
EmployeeCount
EmployeeNumber -0.0 -0.1 0.0 0.0 1.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.0 -0.1 0.1 -0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 -0.0
EnvironmentSatisfaction 0.0 0.0 -0.0 -0.0 0.0 1.0 -0.0 -0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 -0.0 -0.1
HourlyRate 0.0 0.0 0.0 0.0 0.0 -0.0 1.0 0.0 -0.0 -0.1 -0.0 -0.0 0.0 -0.0 -0.0 0.0 0.1 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0
JobInvolvement 0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 1.0 -0.0 -0.0 -0.0 -0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 -0.0 0.0 -0.1
JobLevel 0.5 0.0 0.0 0.1 -0.0 0.0 -0.0 -0.0 1.0 -0.0 1.0 0.0 0.1 -0.0 -0.0 0.0 0.0 0.8 -0.0 0.0 0.5 0.4 0.4 0.4 -0.2
JobSatisfaction -0.0 0.0 -0.0 -0.0 -0.0 -0.0 -0.1 -0.0 -0.0 1.0 -0.0 0.0 -0.1 0.0 0.0 -0.0 0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.1
MonthlyIncome 0.5 0.0 -0.0 0.1 -0.0 -0.0 -0.0 -0.0 1.0 -0.0 1.0 0.0 0.1 -0.0 -0.0 0.0 0.0 0.8 -0.0 0.0 0.5 0.4 0.3 0.3 -0.2
MonthlyRate 0.0 -0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 1.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 0.0
NumCompaniesWorked 0.3 0.0 -0.0 0.1 -0.0 0.0 0.0 0.0 0.1 -0.1 0.1 0.0 1.0 -0.0 -0.0 0.1 0.0 0.2 -0.1 -0.0 -0.1 -0.1 -0.0 -0.1 0.0
PercentSalaryHike 0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.0 1.0 0.8 -0.0 0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0
PerformanceRating 0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.0 0.8 1.0 -0.0 0.0 0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0
RelationshipSatisfaction 0.1 0.0 0.0 -0.0 -0.1 0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 0.1 -0.0 -0.0 1.0 -0.0 0.0 0.0 0.0 0.0 -0.0 0.0 -0.0 -0.0
StandardHours
StockOptionLevel 0.0 0.0 0.0 0.0 0.1 0.0 0.1 0.0 0.0 0.0 0.0 -0.0 0.0 0.0 0.0 -0.0 1.0 0.0 0.0 0.0 0.0 0.1 0.0 0.0 -0.1
TotalWorkingYears 0.7 0.0 0.0 0.1 -0.0 -0.0 -0.0 -0.0 0.8 -0.0 0.8 0.0 0.2 -0.0 0.0 0.0 0.0 1.0 -0.0 0.0 0.6 0.5 0.4 0.5 -0.2
TrainingTimesLastYear -0.0 0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 0.0 -0.1 -0.0 -0.0 0.0 0.0 -0.0 1.0 0.0 0.0 -0.0 -0.0 -0.0 -0.1
WorkLifeBalance -0.0 -0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 0.0 0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 -0.1
YearsAtCompany 0.3 -0.0 0.0 0.1 -0.0 0.0 -0.0 -0.0 0.5 -0.0 0.5 -0.0 -0.1 -0.0 0.0 0.0 0.0 0.6 0.0 0.0 1.0 0.8 0.6 0.8 -0.1
YearsInCurrentRole 0.2 0.0 0.0 0.1 -0.0 0.0 -0.0 0.0 0.4 -0.0 0.4 -0.0 -0.1 -0.0 0.0 -0.0 0.0 0.5 -0.0 0.0 0.8 1.0 0.5 0.7 -0.2
YearsSinceLastPromotion 0.2 -0.0 0.0 0.1 -0.0 0.0 -0.0 -0.0 0.4 -0.0 0.3 0.0 -0.0 -0.0 0.0 0.0 0.0 0.4 -0.0 0.0 0.6 0.5 1.0 0.5 -0.0
YearsWithCurrManager 0.2 -0.0 0.0 0.1 -0.0 0.0 -0.0 0.0 0.4 -0.0 0.3 -0.0 -0.1 -0.0 0.0 -0.0 0.0 0.5 -0.0 0.0 0.8 0.7 0.5 1.0 -0.2
Attrition_numerical -0.2 -0.1 0.1 -0.0 -0.0 -0.1 -0.0 -0.1 -0.2 -0.1 -0.2 0.0 0.0 -0.0 0.0 -0.0 -0.1 -0.2 -0.1 -0.1 -0.1 -0.2 -0.0 -0.2 1.0
Age
DailyRate
DistanceFromHome
Education
EmployeeCount
EmployeeNumber
EnvironmentSatisfaction
HourlyRate
JobInvolvement
JobLevel
JobSatisfaction
MonthlyIncome
MonthlyRate
NumCompaniesWorked
PercentSalaryHike
PerformanceRating
RelationshipSatisfaction
StandardHours
StockOptionLevel
TotalWorkingYears
TrainingTimesLastYear
WorkLifeBalance
YearsAtCompany
YearsInCurrentRole
YearsSinceLastPromotion
YearsWithCurrManager
Attrition_numerical
1.0
0.8
0.6
0.4
0.2
0.0

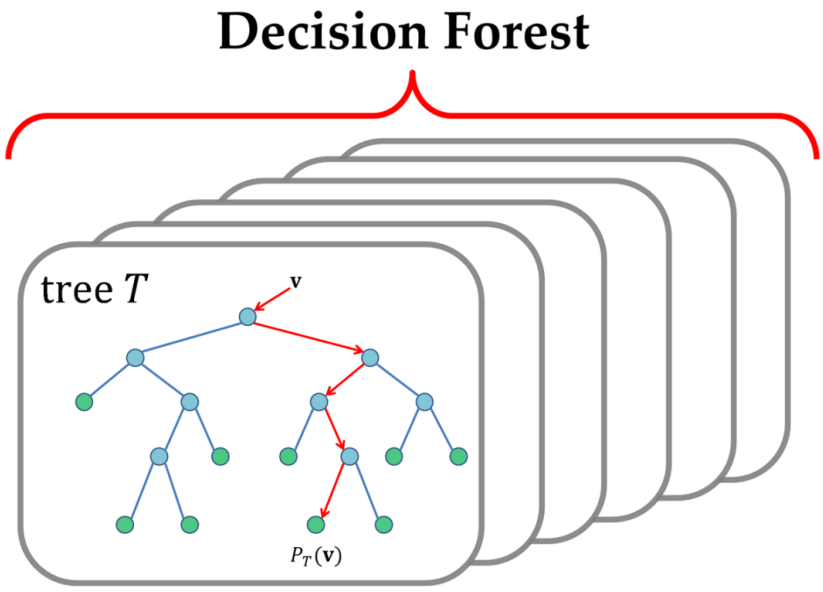
Technical Approach

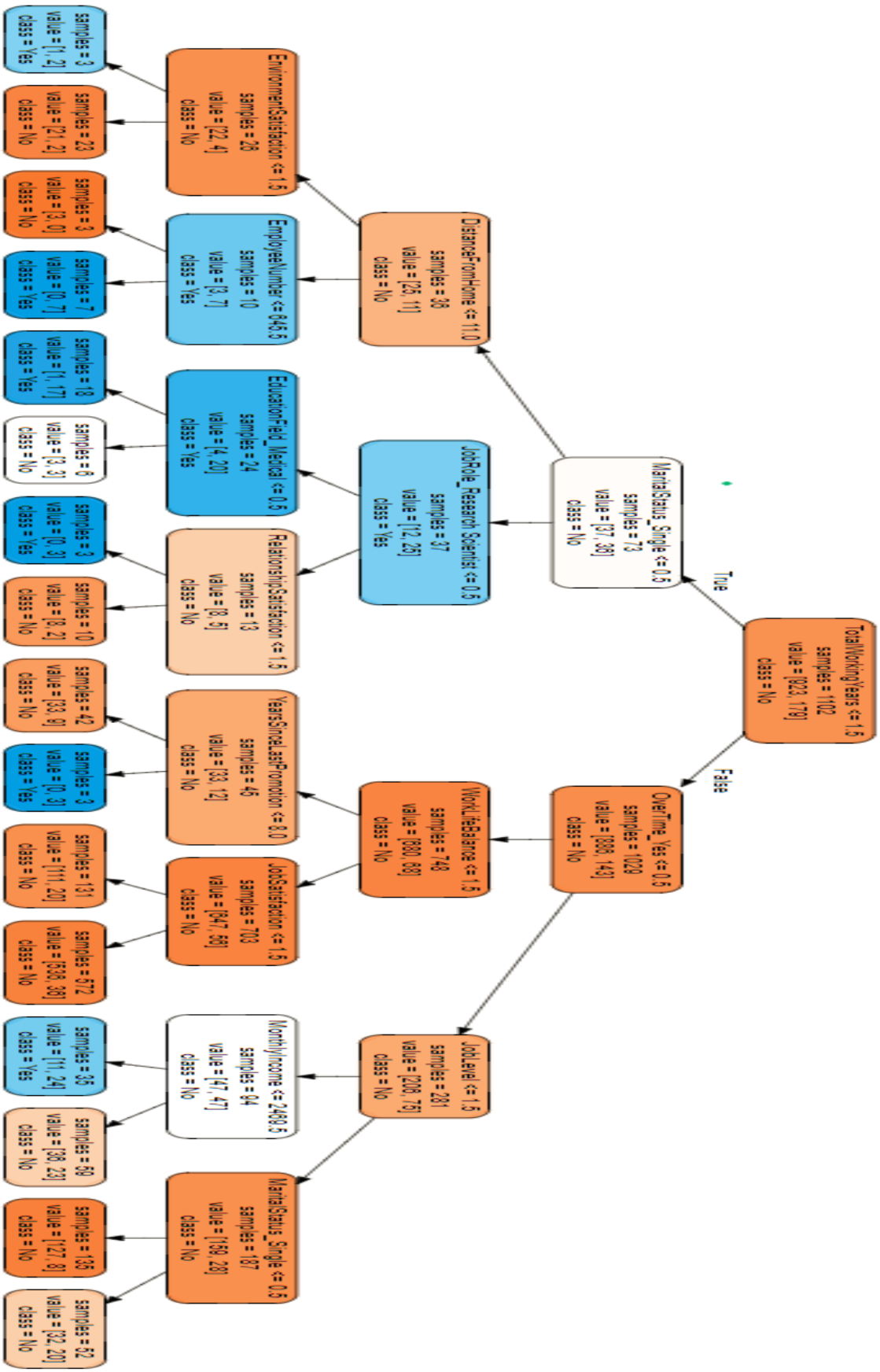
Following are the two Machine Learning model approach which has been adopted to get the required result with improvised accuracy:

1.Random Forest

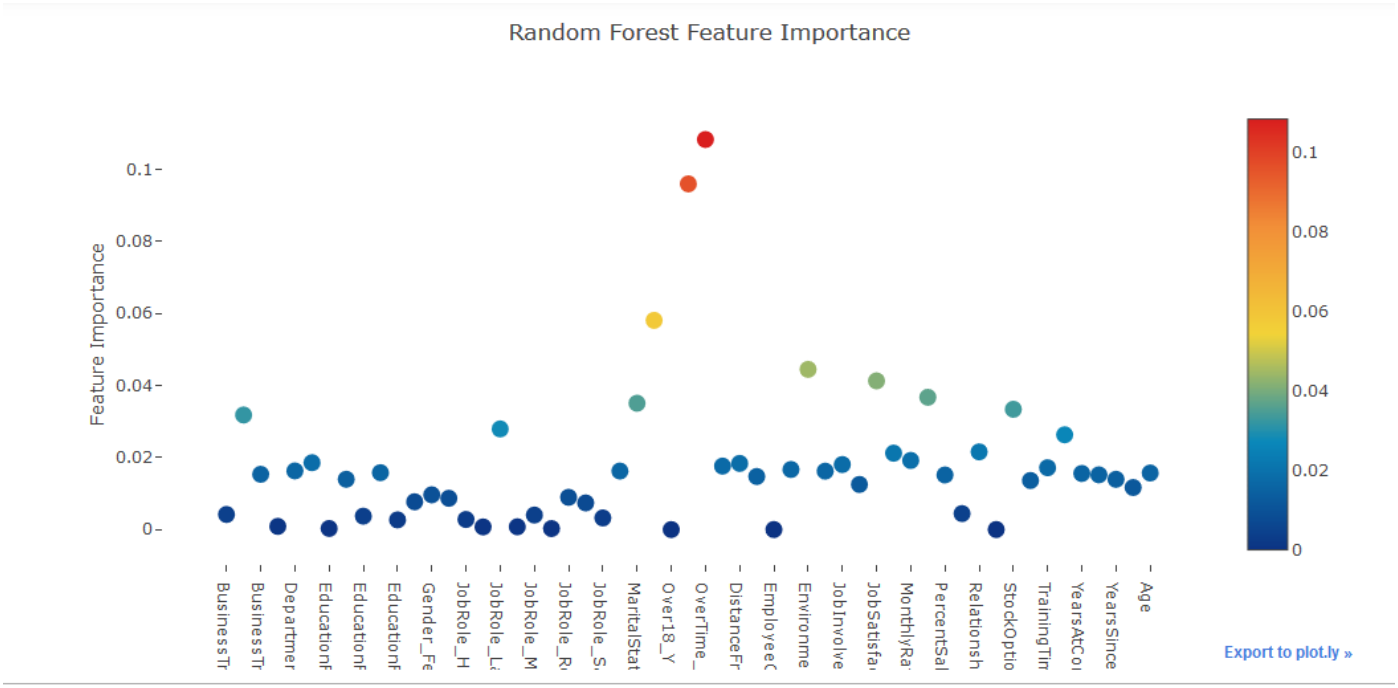
2.Keras Python Library

1. Random Forest data construct is applied to machine learning that develops large numbers of **random** decision trees analyzing sets of variables provided by the dataset. This algorithm helps to enhance the ways that technologies analyze **complex** data. The categorical data has been transformed to numerical by doing one hot encoding.
2. This is a regression problem, so Regression based Random forest is being used.




TotalWorkingYears <= 1.5
samples = 1102
value = [923, 179]
class = No
True
False
MaritalStatus_Single <= 0.5
samples = 73
value = [37, 36]
class = No
OverTime_Yes <= 0.5
samples = 1029
value = [886, 143]
class = No
DistanceFromHome <= 11.0
samples = 36
value = [25, 11]
class = No
JobRole_Research Scientist <= 0.5
samples = 37
value = [12, 25]
class = Yes
WorkLifeBalance <= 1.5
samples = 748
value = [680, 68]
class = No
JobLevel <= 1.5
samples = 281
value = [206, 75]
class = No
EnvironmentSatisfaction <= 1.5
samples = 26
value = [22, 4]
class = No
EmployeeNumber <= 849.5
samples = 10
value = [3, 7]
class = Yes
EducationField_Medical <= 0.5
samples = 24
value = [4, 20]
class = Yes
RelationshipSatisfaction <= 1.5
samples = 13
value = [8, 5]
class = No
YearsSinceLastPromotion <= 8.0
samples = 45
value = [33, 12]
class = No
JobSatisfaction <= 1.5
samples = 703
value = [647, 56]
class = No
MonthlyIncome <= 2469.5
samples = 94
value = [47, 47]
class = No
MaritalStatus_Single <= 0.5
samples = 187
value = [159, 28]
class = No
samples = 3
value = [1, 2]
class = Yes
samples = 23
value = [21, 2]
class = No
samples = 3
value = [3, 0]
class = No
samples = 7
value = [0, 7]
class = Yes
samples = 18
value = [1, 17]
class = Yes
samples = 6
value = [3, 3]
class = No
samples = 3
value = [0, 3]
class = Yes
samples = 10
value = [8, 2]
class = No
samples = 42
value = [33, 9]
class = No
samples = 3
value = [0, 3]
class = Yes
samples = 131
value = [111, 20]
class = No
samples = 572
value = [536, 36]
class = No
samples = 35
value = [11, 24]
class = Yes
samples = 59
value = [36, 23]
class = No
samples = 135
value = [127, 8]
class = No
samples = 52
value = [32, 20]
class = No

Feature Selection by Random Forest



It can be observed that, the overtime, marital status and Job roles are one of the biggest contributing factor.

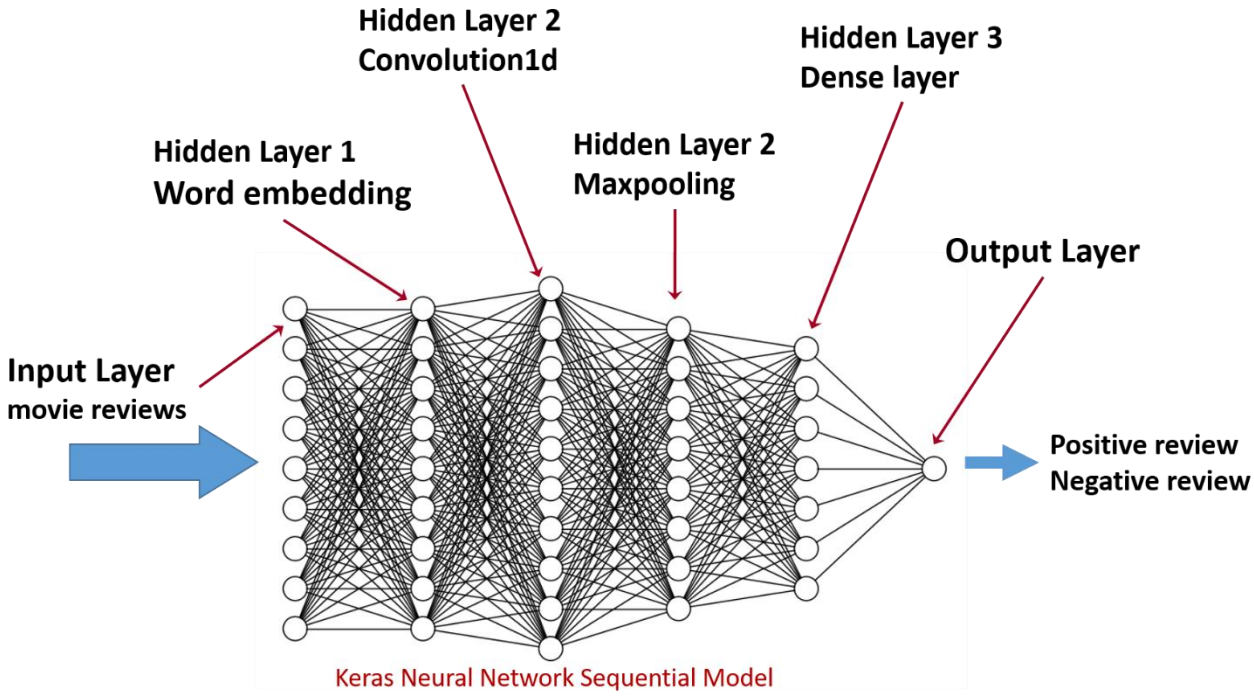
Feature selection will be done on the basis of the random forest.

Python Library used:

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Keras advantages:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.



Here is one of the use case of the library where it is being used for developing deep neural net for movie review sentimental analysis.

Methodology

Code Implementation

1. Configuring Random Forest

In [56]:
```
seed = 0
# Random Forest parameters
rf_params = {
    'n_jobs': -1,
    'n_estimators': 900,
    'warm_start': True,
    'max_features': 0.4,
    'max_depth':12,
    'min_samples_leaf': 2,
    'max_features' : 'sqrt',
    'random_state' : seed,
    'verbose': 0
}
```

In [103]:
```
rf = RandomForestClassifier(**rf_params)
rf.fit(smote_train, smote_target)
```

Out[103]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=12, max_features='sqrt', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=2, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=900, n_jobs=-1,
            oob_score=False, random_state=0, verbose=0, warm_start=True)
```

In [58]:
```
rf_predictions = rf.predict(test)
print("Predictions finished")
```

```
Predictions finished
```

In [59]:
```
accuracy_score(target_val, rf_predictions)
```

Out[59]:
```
0.8600682593856656
```

Here the accuracy for the predicted output, which is the possibility of employee leaving the company is obtained. The accuracy is 86%.

2. Preparing data for Artificial Neural Network

a. Doing One Hot Encoding using

i. Label Encoder

ii. One HotEncoder

In [76]:
```
X = myDataset.iloc[:, 1:].values
y = myDataset.iloc[:, 0].values
```

In [77]:
```
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_3 = LabelEncoder()
X[:, 3] = labelencoder_X_3.fit_transform(X[:, 3])
labelencoder_X_6= LabelEncoder()
X[:, 6] = labelencoder_X_6.fit_transform(X[:, 6])
labelencoder_X_9= LabelEncoder()
X[:, 9] = labelencoder_X_9.fit_transform(X[:, 9])
labelencoder_X_13= LabelEncoder()
X[:, 13] = labelencoder_X_13.fit_transform(X[:, 13])
labelencoder_X_15= LabelEncoder()
X[:, 15] = labelencoder_X_15.fit_transform(X[:, 15])
labelencoder_X_19= LabelEncoder()
X[:, 19] = labelencoder_X_19.fit_transform(X[:, 19])
X = X.astype(float)
labelencoder_y= LabelEncoder()
y = labelencoder_y.fit_transform(y)
```

In [78]:
```
#no dummy trap
onehotencoder1 = OneHotEncoder(categorical_features = [1])
X = onehotencoder1.fit_transform(X).toarray()
X = X[:,1:]
onehotencoder3 = OneHotEncoder(categorical_features = [4])
X = onehotencoder3.fit_transform(X).toarray()
X = X[:,1:]
onehotencoder6 = OneHotEncoder(categorical_features = [8])
X = onehotencoder6.fit_transform(X).toarray()
X = X[:,1:]
onehotencoder13 = OneHotEncoder(categorical_features = [19])
X = onehotencoder13.fit_transform(X).toarray()
X = X[:,1:]
onehotencoder15 = OneHotEncoder(categorical_features = [28])
X = onehotencoder15.fit_transform(X).toarray()
X = X[:,1:]
```

Implementing Keras:

Why Keras?

1.Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).

2.Supports both convolutional networks and recurrent networks, as well as combinations of the two.

3.Runs seamlessly on CPU and GPU.

In [81]:
```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
```

```
Using TensorFlow backend.
```

In [88]:
```
dropout = 0.1
epochs = 10
batch_size = 10
optimizer = 'adam'
k = 20
```

1. **Dropout**: Drops the hidden unit randomly from different layers with probability 0 to 0.5 so that Hidden units in NN don't find their own suitable pattern which leads to over-fitting

2. **Optimizer** = 'adam'--has little memory requirements

3. **Loss needs two elements:**

1 Optimizer

2. Loss Function= binary_crossentropy (helps in optimization)

Configuring Keras layers

In [84]:
```
# Evaluating the Keras ANN using Keras Sequential model
def build_classifier():
    classifier = Sequential()

    classifier.add(Dense(16, kernel_initializer="truncated_normal", activation = 'relu', input_shape = (X.shape[1],)))

    classifier.add(Dropout(dropout))

    classifier.add(Dense(1, kernel_initializer="truncated_normal", activation = 'sigmoid', )) #outputlayer

    classifier.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = ["accuracy"])

    return classifier
```

In [89]:
```
classifier = KerasClassifier(build_fn = build_classifier, batch_size = batch_size, epochs = epochs, verbose=0)
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 30)
max = accuracies.max()
```

In [90]:
```
print("Best accuracy: ",max)
```

```
Best accuracy:  0.96551723315798
```

The accuracy obtained is better than the accuracy obtained through the Random Forest.

Methods adopted to tweak an existing machine learning algorithm:

1. Change Dropouts
2. Adjust Layers
3. Modify epoch

Activators for Keras:

1. **Relu** : rectifier is an activation function defined as the positive part of its argument x+
2. **Sigmoid** : used to introduce nonlinearity in the model

In [96]:
```
dropout = 0.2
epochs = 50
batch_size = 10
optimizer = 'adam'
k = 20
```

In [ ]:
```
```

In [ ]:
```
```

In [97]:
```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
def build_classifier():
    classifier = Sequential()
    classifier.add(Dense(16, kernel_initializer="truncated_normal", activation = 'relu', input_shape = (X.shape[1],)))
    classifier.add(Dropout(dropout))
    classifier.add(Dense(1, kernel_initializer="truncated_normal", activation = 'sigmoid', )) #outputlayer
    classifier.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = ["accuracy"])
    return classifier
classifier = KerasClassifier(build_fn = build_classifier, batch_size = batch_size, epochs = epochs, verbose=0)
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 30)
max = accuracies.max()
```

In [193]:
```
print("Best accuracy: ",max)
```

```
Best accuracy:  0.9743589682456775
```

The tweaking of the dropout and epochs led to better accuracy which is very satisfactory. If it is more than 98 then the model would be over fit and may lead to inaccuracies.

Conclusion

In this project, it is being derived that, marital status and Job roles are one of the biggest contributing factor for the attrition status. Apart from that an efficient and dynamic model is being formulated which predicts the possibilities of attrition of an employee based on the given employee information. The accuracy of the Decision Tree model has been found to be around 87 % while the Keras Deep learning model is being trained to the improvised accuracy of around 97%.

Acknowledgement

I would like to acknowledge Professor Nick Brown for his constant motivation and knowledge. I would also like to thank the teaching assistant for help and guidance with the code implementation.

References

1. https://keras.io/
2. https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset/data
3. http://scikit-learn.org/stable/modules/tree.html
4. https://seaborn.pydata.org/
5. https://seaborn.pydata.org/tutorial.html
6. https://www.tensorflow.org/api_docs/python/tf/keras/backend/stack
7. https://stackoverflow.com/questions/38714959/understanding-keras-lstms