Employee ATTRITION –MODELing

Data Mining – Group Assignment

**Data Mining Group 12**

AKSHAY ARORA

BHOMIK CHOPRA

MANISHA DAWAR

SHWETA GUPTA

VARUN GAUTAM

Table of Contents

* Data Import (Target variable is "Attrition" column)
* Split the data in Dev & Hold Out sample (70:30)
* Perform Exploratory Data Analysis
* Identify columns which are of no use. drop those columns
* Write Hypothesis and validate the Hypothesis
* Build Neural Network Model (Development sample)
* Validate NN model on Hold Out. If need be improvise
* Build Random Forest Model
* Validate RF Model
* Compare NN, RF and CART model
* Combine NN, RF and CART into Ensemble Model
* Check whether Ensemble Model Performance outperforms the individual models

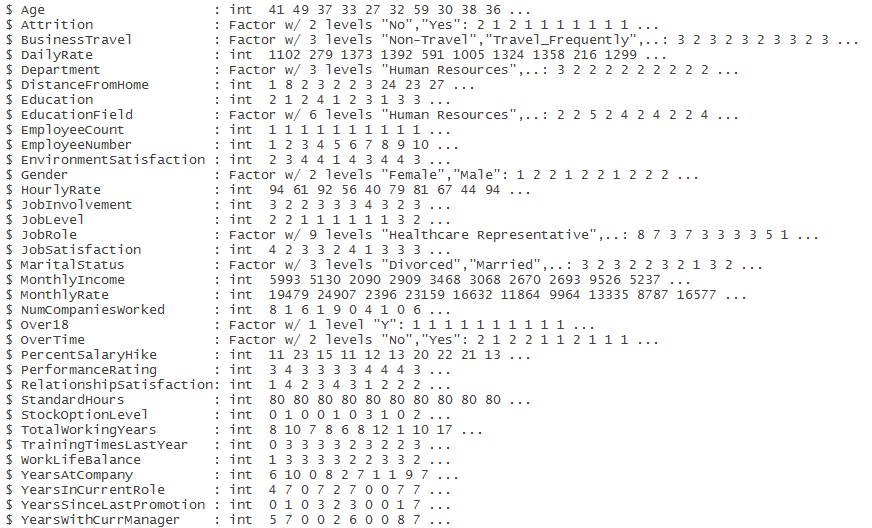
# **Dataset**

### 

### **Import the HR\_Employee\_Attrition\_Data.csv file in R**

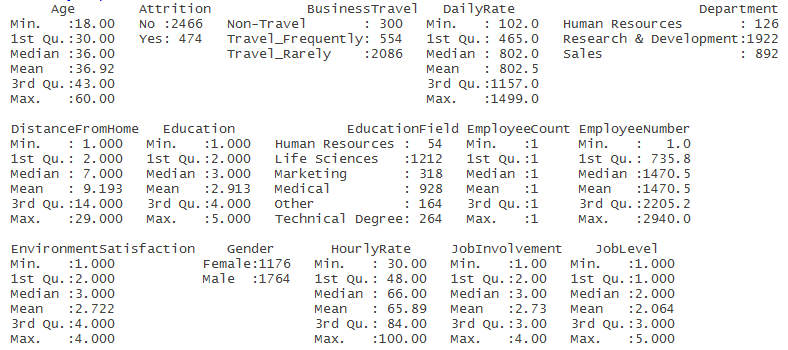
* The Employee Attrition Dataset has 2940 observations and 35 variables.

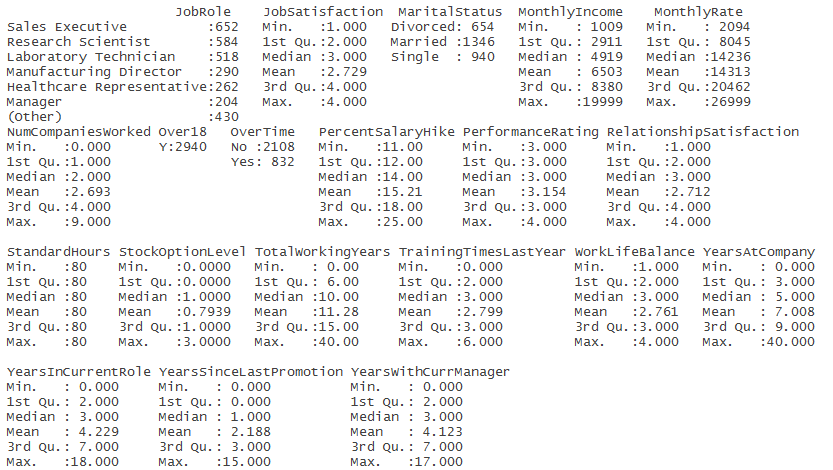
### **See the structure of the file using str() function in R**



* Attrition, Business Travel, Department, Education Field, Gender, Job Role, Marital Status, Over 18, Over Time are Factor variables.
* All the other variables are integer variables.
* Target variable – Attrition has two levels – Yes, No.

### **Perform EDA of the data using summary() function**



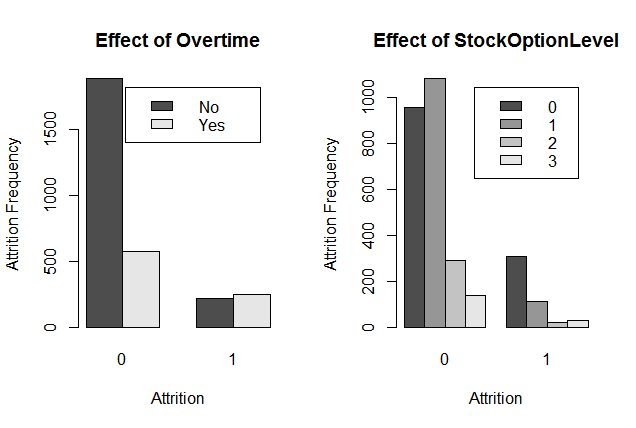


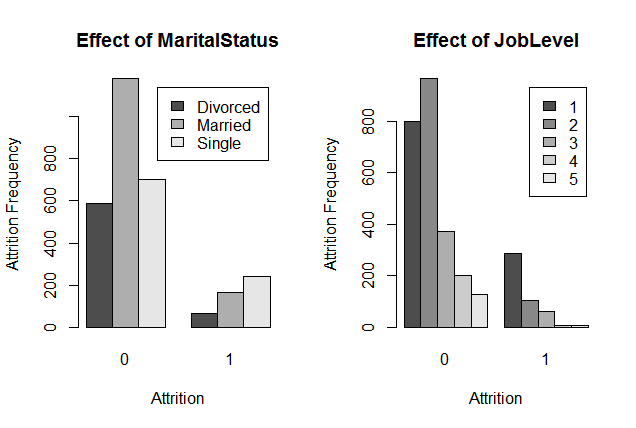
* Target variable – Attrition has 84% observations as No and 16% as Yes.
* Employee Count variable has a value of 1 for all the observations.
* Over 18 variable has a value Y for all of the 2940 observations.
* Standard Hours variable has a value of 80 for all of the 2940 observations.

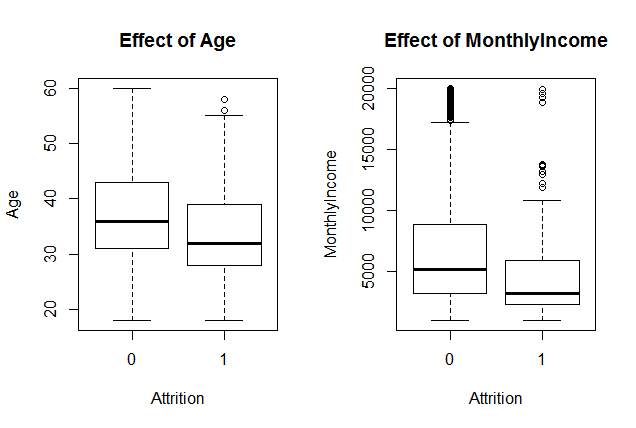
**Understanding Distribution of Variables**



**Bi-Variate Analysis:-**







----------------------------------------------------------------------------------------------------------------

Checking for Correlation in data   
----------------------------------------------------------------------------------------------------------------

cor(data[,unlist(lapply(data, is.numeric))])

Observations:

1. Job level has high correlation with monthly income and total working years
2. Monthly Income & Total working years are highly correlated
3. Percent Salary hike is highly correlated with Performance rating
4. Years at company is correlated with years in role & years with current manager

These correlations however don’t give a ready insight to attrition rate.

----------------------------------------------------------------------------------------------------------------

Other general observations from data  
----------------------------------------------------------------------------------------------------------------

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attrition Analysis with Gender and Marital Status | | | | |  |
| Category | No | Yes | Grand Total | % Attrition | Attrition weightage |
| Female | 1002 | 174 | 1176 | 14.80 | 36.71 |
| Divorced | 216 | 18 | 234 | 7.69 | 10.34 |
| Married | 482 | 62 | 544 | 11.40 | 35.63 |
| Single | 304 | 94 | 398 | 23.62 | 54.02 |
| Male | 1464 | 300 | 1764 | 17.01 | 63.29 |
| Divorced | 372 | 48 | 420 | 11.43 | 16.00 |
| Married | 696 | 106 | 802 | 13.22 | 35.33 |
| Single | 396 | 146 | 542 | 26.94 | 48.67 |
| Grand Total | 2466 | 474 | 2940 | 16.12 |  |

High attrition rates of more than 20 % observed at following ages. Which also accounts for approximately 53% of total attrition.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Age | No | Yes | Grand Total | % Attrition |
| 18 | 8 | 8 | 16 | 50.00 |
| 19 | 6 | 12 | 18 | 66.67 |
| 20 | 10 | 12 | 22 | 54.55 |
| 21 | 14 | 12 | 26 | 46.15 |
| 22 | 22 | 10 | 32 | 31.25 |
| 23 | 20 | 8 | 28 | 28.57 |
| 24 | 38 | 14 | 52 | 26.92 |
| 25 | 40 | 12 | 52 | 23.08 |
| 26 | 54 | 24 | 78 | 30.77 |
| 28 | 68 | 28 | 96 | 29.17 |
| 29 | 100 | 36 | 136 | 26.47 |
| 31 | 102 | 36 | 138 | 26.09 |
| 33 | 92 | 24 | 116 | 20.69 |
| 56 | 22 | 6 | 28 | 21.43 |
| 58 | 18 | 10 | 28 | 35.71 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Department Name | No | Yes | Grand Total | %Attrition | Attrition weightage |
| Human Resources | 102 | 24 | 126 | 19.05 | 5.06 |
| Research & Development | 1656 | 266 | 1922 | 13.84 | 56.12 |
| Sales | 708 | 184 | 892 | 20.63 | 38.82 |
| Grand Total | 2466 | 474 | 2940 | 16.12 | 100.00 |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Business Travel | No | Yes | Grand Total | %Attrition | Attrition weightage |
| Non-Travel | 276 | 24 | 300 | 8.00 | 5.06 |
| Travel\_Frequently | 416 | 138 | 554 | 24.91 | 29.11 |
| Travel\_Rarely | 1774 | 312 | 2086 | 14.96 | 65.82 |
| Grand Total | 2466 | 474 | 2940 | 16.12 | 100.00 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Job Role | No | Yes | Grand Total | %Attrition | Attrition weightage |
| Healthcare Representative | 244 | 18 | 262 | 6.87 | 3.80 |
| Human Resources | 80 | 24 | 104 | 23.08 | 5.06 |
| Laboratory Technician | 394 | 124 | 518 | 23.94 | 26.16 |
| Manager | 194 | 10 | 204 | 4.90 | 2.11 |
| Manufacturing Director | 270 | 20 | 290 | 6.90 | 4.22 |
| Research Director | 156 | 4 | 160 | 2.50 | 0.84 |
| Research Scientist | 490 | 94 | 584 | 16.10 | 19.83 |
| Sales Executive | 538 | 114 | 652 | 17.48 | 24.05 |
| Sales Representative | 100 | 66 | 166 | 39.76 | 13.92 |
| Grand Total | 2466 | 474 | 2940 | 16.12 | 100.00 |

# **Hypothesis Testing:-**

By performing Hypothesis testing, we found that Age, Monthly Income, Overtime, Marital Status, Job level, job role, Total working Years, Years at Company have significant impact on Attrition.

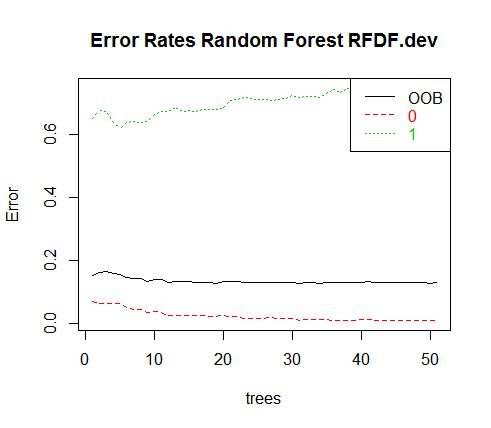
# **Model Creation**

### **Create Training (Development) Sample and Testing (Holdout) Sample**

* Stratified sampling of the Employee Attrition Dataset into Train and Test sets with 70% and 30% of the total observations respectively.
* Each of Train and Test samples have 16% Attrition rate.

 **Error! Not a valid embedded object.**

### **Build Random Forest model**

* Random Forest model is built with Attrition as the target variable.
* All the other variables are used as independent variables except Employee Number (identifier variable), Over 18, Employee Count and Standard Hours (same values for all observations).
* Control Parameters used for model creation:
  + ntree = 51
  + mtry= 9
  + nodesize = 60
* Choosing the number of trees as **51** for Random Forest:-
* 
* Output of Random Forest Model by selecting the best model using tuneRF:-

Type of random forest: classification

Number of trees: **51**

No. of variables tried at each split: **9**

**OOB estimate of error rate: 13.14%**

**Confusion matrix**:

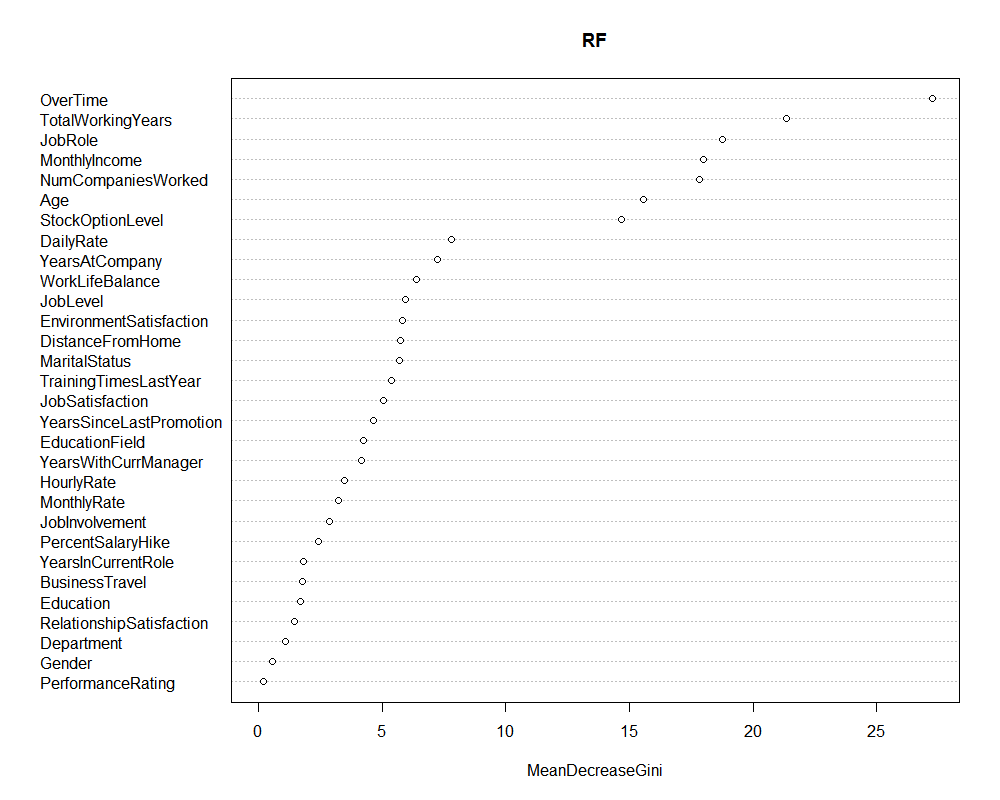
0 1 class.error

0 **1731** **20** 0.01142204455

1 **255** **87** 0.74561403509

**Training error- 13%**

**Important Variables:-**

****

Note:- Now, we can rebuild our model with less variables based on above mentioned importance of variables.

### **Building Neural Network Model**

* First step is to convert categorical variables to dummy variables. We get 94 variables to work with.
* Next , scale the variables.
* Control parameters for NN model:-
* hidden = c(10,3) # 2 layers with 10 and 3 neurons respectively as we have around 94 variables
* err.fct = "sse",
* linear.output = FALSE,
* lifesign = "full",
* lifesign.step = 1,
* threshold = 0.1,
* stepmax = 2000
* **Confusion Matrix:-**

Attrition **0 1**

**0 1743 8**

**1 17 325**

* **Training Error:- 2%**

# 

# **Model Validation**

### **Random Forest Model Evaluation**

### **Test the model on Training and Testing Sample**

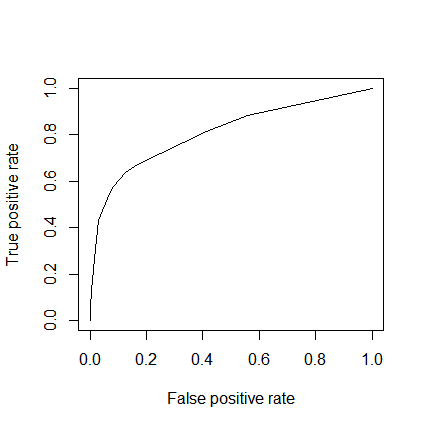
* Rank ordering on Development sample:-

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **deciles** | **cnt** | **cnt\_resp** | **cnt\_non\_resp** | **rrate** | **cum\_resp** | **cum\_non\_resp** | **cum\_rel\_resp** | **cum\_rel\_non\_resp** | **ks** |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 210 | 184 | 26 | 88% | 184 | 26 | 54% | 1% | 0.53 |
| 9 | 213 | 107 | 106 | 50% | 291 | 132 | 85% | 8% | 0.75 |
| 8 | 205 | 26 | 179 | 13% | 317 | 311 | 93% | 18% | 0.77 |
| 7 | 209 | 12 | 197 | 6% | 329 | 508 | 96% | 29% | 0.67 |
| 6 | 221 | 5 | 216 | 2% | 334 | 724 | 98% | 41% | 0.57 |
| 5 | 231 | 7 | 224 | 3% | 341 | 948 | 100% | 54% | 0.46 |
| 4 | 217 | 1 | 216 | 0% | 342 | 1164 | 100% | 66% | 0.34 |
| 3 | 273 | 0 | 273 | 0% | 342 | 1437 | 100% | 82% | 0.18 |
| 2 | 209 | 0 | 209 | 0% | 342 | 1646 | 100% | 94% | 0.06 |
| 1 | 105 | 0 | 105 | 0% | 342 | 1751 | 100% | 100% | 0 |

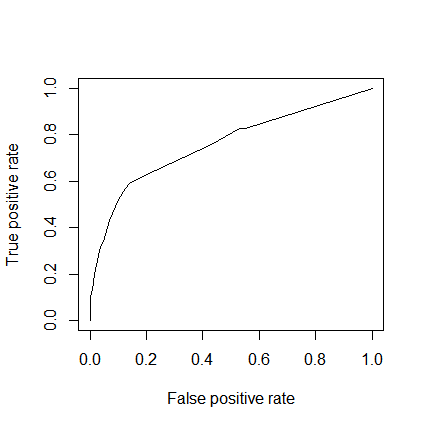
* Rank ordering on Holdout sample:-

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **deciles** | **cnt** | **cnt\_resp** | **cnt\_non\_resp** | **rrate** | **cum\_resp** | **cum\_non\_resp** | **cum\_rel\_resp** | **cum\_rel\_non\_resp** | **ks** |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 86 | 59 | 27 | 69% | 59 | 27 | 45% | 4% | 0.41 |
| 9 | 84 | 34 | 50 | 40% | 93 | 77 | 70% | 11% | 0.59 |
| 8 | 84 | 15 | 69 | 18% | 108 | 146 | 82% | 20% | 0.62 |
| 7 | 88 | 11 | 77 | 12% | 119 | 223 | 90% | 31% | 0.59 |
| 6 | 85 | 4 | 81 | 5% | 123 | 304 | 93% | 43% | 0.5 |
| 5 | 98 | 6 | 92 | 6% | 129 | 396 | 98% | 55% | 0.43 |
| 4 | 89 | 2 | 87 | 2% | 131 | 483 | 99% | 68% | 0.31 |
| 3 | 73 | 1 | 72 | 1% | 132 | 555 | 100% | 78% | 0.22 |
| 2 | 97 | 0 | 97 | 0% | 132 | 652 | 100% | 91% | 0.09 |
| 1 | 63 | 0 | 63 | 0% | 132 | 715 | 100% | 100% | 0 |

* KS value for Development sample is **0.77**
* KS value for Holdout sample is **0.62**
* AUC-ROC for Development Sample:-



* AUC value is **0.80**
* AUC-ROC for Holdout Sample:-



* AUC value is **0.76**
* Error matrix on Development sample:-

|  |  |  |
| --- | --- | --- |
| Actual Attrition/Predicted | 0 | 1 |
| 0 | 1731 | 20 |
| 1 | 225 | 87 |

* Error matrix on Holdout sample:-

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | 0 | 1 |
| 0 | 706 | 9 |
| 1 | 94 | 38 |

* Classification Accuracy on holdout sample = 87%

### **Neural Network Model Evaluation**

### **Test the model on Training and Testing Sample**

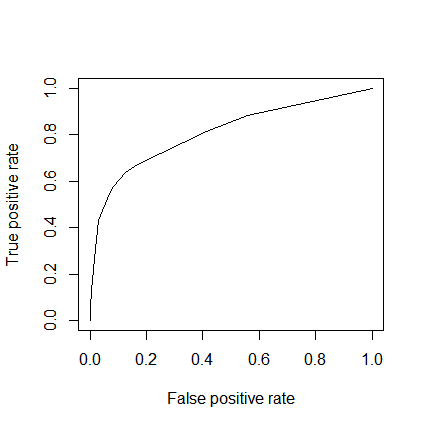
* Rank ordering on Development sample:-

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **deciles** | **cnt** | **cnt\_resp** | **cnt\_non\_resp** | **rrate** | **cum\_resp** | **cum\_non\_resp** | **cum\_rel\_resp** | **cum\_rel\_non\_resp** | **ks** |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 210 | 209 | 1 | 100% | 209 | 1 | 61% | 0% | 0.61 |
| 9 | 210 | 118 | 92 | 56% | 327 | 93 | 96% | 5% | 0.91 |
| 8 | 208 | 0 | 208 | 0% | 327 | 301 | 96% | 17% | 0.79 |
| 7 | 209 | 2 | 207 | 1% | 329 | 508 | 96% | 29% | 0.67 |
| 6 | 211 | 1 | 210 | 0% | 330 | 718 | 96% | 41% | 0.55 |
| 5 | 208 | 2 | 206 | 1% | 332 | 924 | 97% | 53% | 0.44 |
| 4 | 210 | 2 | 208 | 1% | 334 | 1132 | 98% | 65% | 0.33 |
| 3 | 208 | 1 | 207 | 0% | 335 | 1339 | 98% | 76% | 0.22 |
| 2 | 210 | 2 | 208 | 1% | 337 | 1547 | 99% | 88% | 0.11 |
| 1 | 209 | 5 | 204 | 2% | 342 | 1751 | 100% | 100% | 0 |

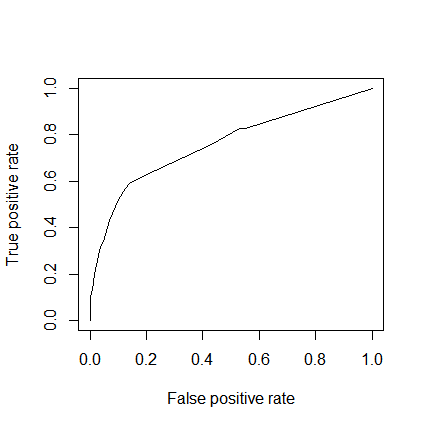
* Rank ordering on Holdout sample:-

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **deciles** | **cnt** | **cnt\_resp** | **cnt\_non\_resp** | **rrate** | **cum\_resp** | **cum\_non\_resp** | **cum\_rel\_resp** | **cum\_rel\_non\_resp** | **ks** |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 85 | 78 | 7 | 92% | 78 | 7 | 59% | 1% | 0.58 |
| 9 | 85 | 41 | 44 | 48% | 119 | 51 | 90% | 7% | 0.83 |
| 8 | 84 | 2 | 82 | 2% | 121 | 133 | 92% | 19% | 0.73 |
| 7 | 86 | 2 | 84 | 2% | 123 | 217 | 93% | 30% | 0.63 |
| 6 | 84 | 3 | 81 | 4% | 126 | 298 | 95% | 42% | 0.53 |
| 5 | 84 | 0 | 84 | 0% | 126 | 382 | 95% | 53% | 0.42 |
| 4 | 85 | 2 | 83 | 2% | 128 | 465 | 97% | 65% | 0.32 |
| 3 | 84 | 1 | 83 | 1% | 129 | 548 | 98% | 77% | 0.21 |
| 2 | 85 | 0 | 85 | 0% | 129 | 633 | 98% | 89% | 0.09 |
| 1 | 85 | 3 | 82 | 4% | 132 | 715 | 100% | 100% | 0 |

* KS value for Development sample is **0.91**
* KS value for Holdout sample is **0.83**
* AUC-ROC for Development Sample:-



* AUC value is **0.90**
* AUC-ROC for Holdout Sample:-



* AUC value is **0.88**
* Error matrix on Development sample:-

|  |  |  |
| --- | --- | --- |
| Actual Attrition/Predicted | 0 | 1 |
| 0 | 1743 | 8 |
| 1 | 17 | 325 |

* Error matrix on Holdout sample:-

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | 0 | 1 |
| 0 | 668 | 47 |
| 1 | 13 | 119 |

* Classification Accuracy on holdout sample = 93%

# **Ensemble Modeling**

* Probability scores of Random Forest Model and Neural Network Model are Combined.
* Development Sample Output:-

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **predict.class.RF** | **predict.score.RF** | **deciles** | **Class.nn** | **deciles.1** | **prob.score.nn** | **avg.score** | **deciles.final** | **final.class** |
|  |  |  |  |  |  |  |  |  |
| 0 | 0.994 | 3 | 0 | 2 | 1 | 0.997 | 9 | 0 |
| 1 | 0.334 | 10 | 1 | 9 | 0.016 | 0.175 | 1 | 0 |
| 0 | 0.816 | 9 | 0 | 8 | 0.9966 | 0.906 | 3 | 0 |
| 0 | 0.976 | 5 | 0 | 8 | 0.9946 | 0.985 | 6 | 0 |
| 0 | 0.812 | 9 | 0 | 9 | 0.99 | 0.901 | 3 | 0 |
| 0 | 0.84 | 8 | 0 | 6 | 1 | 0.92 | 3 | 0 |
| 0 | 0.996 | 2 | 0 | 2 | 1 | 0.998 | 9 | 0 |
| 0 | 0.994 | 3 | 0 | 1 | 1 | 0.997 | 9 | 0 |
| 0 | 0.782 | 9 | 0 | 4 | 1 | 0.891 | 3 | 0 |
| 0 | 0.994 | 3 | 0 | 2 | 1 | 0.997 | 9 | 0 |
| 1 | 0.19 | 10 | 1 | 10 | 0.0109 | 0.1 | 1 | 0 |
| 0 | 0.894 | 8 | 0 | 5 | 1 | 0.947 | 3 | 0 |

* Hold out sample Output:-

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **predict.class.RF** | **predict.score.RF** | **deciles** | **Predict.score.nn** | **Class.nn** | **deciles.1** | **prob.score.nn** | **avg.score** | **deciles.final** | **final.class** |
|  |  |  |  |  |  |  |  |  |  |
| 0 | 0.6 | 10 | 0.9928 | 1 | 10 | 0.0072 | 0.3026 | 1 | 0 |
| 0 | 0.99 | 3 | 1E-07 | 0 | 2 | 1 | 0.994 | 8 | 0 |
| 0 | 1 | 1 | 3E-07 | 0 | 3 | 1 | 0.999 | 10 | 0 |
| 0 | 0.98 | 4 | 4E-06 | 0 | 5 | 1 | 0.991 | 7 | 0 |
| 0 | 0.99 | 3 | 0.0003 | 0 | 7 | 0.9997 | 0.9939 | 8 | 0 |
| 0 | 0.82 | 9 | 0.0007 | 0 | 7 | 0.9993 | 0.9097 | 3 | 0 |
| 0 | 0.99 | 3 | 2E-07 | 0 | 3 | 1 | 0.993 | 8 | 0 |
| 1 | 0.46 | 10 | 5E-06 | 0 | 5 | 1 | 0.73 | 2 | 0 |
| 0 | 1 | 2 | 1E-07 | 0 | 2 | 1 | 0.998 | 10 | 0 |
| 0 | 0.96 | 5 | 8E-08 | 0 | 1 | 1 | 0.979 | 6 | 0 |
| 0 | 0.98 | 4 | 2E-07 | 0 | 3 | 1 | 0.99 | 7 | 0 |

* Error matrix on Holdout sample on Ensemble Model:-

|  |  |  |
| --- | --- | --- |
| Actual/Predicted | 0 | 1 |
| 0 | 715 | 0 |
| 1 | 125 | 7 |

* Classification Accuracy on holdout sample = 86%

# **Inferences**

### **Draw inferences from the models**

* Ensemble Model does not outperform the Individual Random Forest and Neural Network Models.
* Neural Network gives the best model and predicts with 93% accuracy that which employee is going to attrite.
* So, if we want to find out in near future which employee is going to churn, we can use this model to find out his/her probability score based on the model and if it lies in top three deciles, then he/she is most likely to churn.
* Based on above calculated Model Evaluation measures , we can say, the model is able to classify Employee Attrition with 93% accuracy which is 7% % better than random guessing as the proportion of Employee Attrition(positive) class is only 16%.

### **Recommendation:-**

* Since the performance of the Neural Network model is consistent across the training and test data sets, the results can be trusted.
* Based on the final model, the recommendation for the HR is to counsel all the employees working overtime and try to understand the reason for the same. Based the discussions’ outcome, appropriate action can be taken and proper strategy can be made.

# **Appendix (Code)**

library(ineq)

library(car)

library(caret)

library(rpart)

library(randomForest)

library(RColorBrewer)

library(rpart.plot)

library(rattle)

library(caTools)

library(gplots)

library(ROCR)

setwd ("E:/Akxay/GLIM/Data Mining/Assignment")

getwd()

## Data Import

data <- read.table("HR\_Employee\_Attrition\_Data.csv", sep = ",", header = T)

dim(data)

str(data)

summary(data)

## Converting Data to variables for further EDA

data$Attrition <- as.factor(data$Attrition)

data$BusinessTravel <- as.factor(data$BusinessTravel)

data$Department <- as.factor(data$Department)

data$Education <- as.factor(data$Education)

data$EducationField <- as.factor(data$EducationField)

data$EnvironmentSatisfaction <- as.factor(data$EnvironmentSatisfaction)

data$Gender <- as.factor(data$Gender)

data$JobInvolvement <- as.factor(data$JobInvolvement)

data$JobLevel <- as.factor(data$JobLevel)

data$JobRole <- as.factor(data$JobRole)

data$JobSatisfaction <- as.factor(data$JobSatisfaction)

data$MaritalStatus <- as.factor(data$MaritalStatus)

data$NumCompaniesWorked <- as.factor(data$NumCompaniesWorked)

data$OverTime <- as.factor(data$OverTime)

data$PerformanceRating <- as.factor(data$PerformanceRating)

data$RelationshipSatisfaction <- as.factor(data$RelationshipSatisfaction)

data$StockOptionLevel <- as.factor(data$StockOptionLevel)

data$TrainingTimesLastYear <- as.factor(data$TrainingTimesLastYear)

data$WorkLifeBalance <- as.factor(data$WorkLifeBalance)

data$Flag[data$Attrition == "Yes"] <- 1

data$Flag[data$Attrition == "No"] <- 0

data$Flag <- as.factor(data$Flag)

# Plotting univariate summaries in the data

par(mfrow= c(1,2))

par(mfrow= c(1,2))

barplot(table(data$OverTime,data$Flag),main="Effect of Overtime",

xlab="Attrition", ylab="Attrition Frequency",

legend = rownames(table(data$OverTime,data$Flag)),beside=TRUE)

barplot(table(data$StockOptionLevel,data$Flag),main="Effect of StockOptionLevel",

xlab="Attrition", ylab="Attrition Frequency",

legend = rownames(table(data$StockOptionLevel,data$Flag)),beside=TRUE)

par(mfrow= c(1,2))

barplot(table(data$MaritalStatus,data$Flag),main="Effect of MaritalStatus",

xlab="Attrition", ylab="Attrition Frequency",

legend = rownames(table(data$MaritalStatus,data$Flag)),beside=TRUE)

barplot(table(data$JobLevel,data$Flag),main="Effect of JobLevel",

xlab="Attrition", ylab="Attrition Frequency",

legend = rownames(table(data$JobLevel,data$Flag)),beside=TRUE)

par(mfrow= c(1,2))

boxplot(data$Age~data$Flag,data=data, main="Effect of Age",

xlab="Attrition", ylab="Age")

boxplot(data$MonthlyIncome ~data$Flag,data=data, main="Effect of MonthlyIncome",

xlab="Attrition", ylab="MonthlyIncome")

# split data into dev and holdout

data$Attrition = factor(data$Attrition, levels = c("No","Yes"), labels=c(0,1))

data$Attrition = as.numeric(as.character(data$Attrition))

set.seed(1001)

data$random = runif(nrow(data),0,1)

RFDF.dev = data[which(data$random <= 0.7),]

RFDF.holdout = data[which(data$random > 0.7),]

c(nrow(RFDF.dev), nrow(RFDF.holdout))

## Target Rate

sum(RFDF.dev$Attrition)/nrow(RFDF.dev)

sum(RFDF.holdout$Attrition)/nrow(RFDF.holdout)

# Building Random Forest

library(randomForest)

?randomForest

View(RFDF.dev)

## Calling syntax to build the Random Forest

RF <- randomForest(as.factor(RFDF.dev$Attrition) ~ ., data = RFDF.dev[,-c(2,9,10,22,27,36,37)],

ntree=51, mtry = 9, nodesize = 60,

importance=TRUE, set.seed(1))

print(RF)

plot(RF, main="")

legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)

title(main="Error Rates Random Forest RFDF.dev")

RF$err.rate

## List the importance of the variables.

impVar <- round(randomForest::importance(RF), 2)

impVar[order(impVar[,3], decreasing=TRUE),]

?tuneRF

## Tuning Random Forest

tRF <- tuneRF(x = RFDF.dev[,-c(2,9,10,22,27,36,37)],

y=as.factor(RFDF.dev$Attrition),

mtryStart = 2,

ntreeTry=51,

stepFactor = 1.5,

improve = 0.001,

trace=TRUE,

plot = FALSE,

doBest = TRUE,

nodesize = 60,

importance=TRUE,

set.seed(1)

)

(tRF$importance)

(VI\_F=importance(RF))

varImpPlot(RF,type=2)

View(RFDF.dev)

## Scoring syntax

RFDF.dev$predict.class.RF <- predict(tRF, RFDF.dev, type="class")

RFDF.dev$predict.score.RF <- predict(tRF, RFDF.dev, type="prob")

head(RFDF.dev)

class(RFDF.dev$predict.score.RF)

## deciling

## deciling code

decile <- function(x){

deciles <- vector(length=10)

for (i in seq(0.1,1,.1)){

deciles[i\*10] <- quantile(x, i, na.rm=T)

}

return (

ifelse(x<deciles[1], 1,

ifelse(x<deciles[2], 2,

ifelse(x<deciles[3], 3,

ifelse(x<deciles[4], 4,

ifelse(x<deciles[5], 5,

ifelse(x<deciles[6], 6,

ifelse(x<deciles[7], 7,

ifelse(x<deciles[8], 8,

ifelse(x<deciles[9], 9, 10

))))))))))

}

RFDF.dev$deciles <- decile(RFDF.dev$predict.score.RF[,2])

library(data.table)

tmp\_DT = data.table(RFDF.dev)

rank <- tmp\_DT[, list(

cnt = length(Attrition),

cnt\_resp = sum(Attrition),

cnt\_non\_resp = sum(Attrition == 0)) ,

by=deciles][order(-deciles)]

rank$rrate <- round (rank$cnt\_resp / rank$cnt,2);

rank$cum\_resp <- cumsum(rank$cnt\_resp)

rank$cum\_non\_resp <- cumsum(rank$cnt\_non\_resp)

rank$cum\_rel\_resp <- round(rank$cum\_resp / sum(rank$cnt\_resp),2);

rank$cum\_rel\_non\_resp <- round(rank$cum\_non\_resp / sum(rank$cnt\_non\_resp),2);

rank$ks <- abs(rank$cum\_rel\_resp - rank$cum\_rel\_non\_resp);

library(scales)

rank$rrate <- percent(rank$rrate)

rank$cum\_rel\_resp <- percent(rank$cum\_rel\_resp)

rank$cum\_rel\_non\_resp <- percent(rank$cum\_rel\_non\_resp)

View(rank)

sum(RFDF.dev$Attrition) / nrow(RFDF.dev)

library(ROCR)

pred <- prediction(RFDF.dev$predict.score[,2], RFDF.dev$Attrition)

perf <- performance(pred, "tpr", "fpr")

plot(perf)

KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])

KS

## Area Under Curve

auc <- performance(pred,"auc");

auc <- as.numeric(auc@y.values)

auc

## Gini Coefficient

library(ineq)

gini = ineq(RFDF.dev$predict.score.RF[,2], type="Gini")

gini

## Classification Error

with(RFDF.dev, table(Attrition, predict.class.RF))

## Scoring syntax

RFDF.holdout$predict.class.RF <- predict(tRF, RFDF.holdout, type="class")

RFDF.holdout$predict.score.RF <- predict(tRF, RFDF.holdout, type="prob")

RFDF.holdout$deciles <- decile(RFDF.holdout$predict.score.RF[,2])

tmp\_DT = data.table(RFDF.holdout)

h\_rank <- tmp\_DT[, list(

cnt = length(Attrition),

cnt\_resp = sum(Attrition),

cnt\_non\_resp = sum(Attrition == 0)) ,

by=deciles][order(-deciles)]

h\_rank$rrate <- round (h\_rank$cnt\_resp / h\_rank$cnt,2);

h\_rank$cum\_resp <- cumsum(h\_rank$cnt\_resp)

h\_rank$cum\_non\_resp <- cumsum(h\_rank$cnt\_non\_resp)

h\_rank$cum\_rel\_resp <- round(h\_rank$cum\_resp / sum(h\_rank$cnt\_resp),2);

h\_rank$cum\_rel\_non\_resp <- round(h\_rank$cum\_non\_resp / sum(h\_rank$cnt\_non\_resp),2);

h\_rank$ks <- abs(h\_rank$cum\_rel\_resp - h\_rank$cum\_rel\_non\_resp);

library(scales)

h\_rank$rrate <- percent(h\_rank$rrate)

h\_rank$cum\_rel\_resp <- percent(h\_rank$cum\_rel\_resp)

h\_rank$cum\_rel\_non\_resp <- percent(h\_rank$cum\_rel\_non\_resp)

View(h\_rank)

# AUC and gini for test sample

library(ROCR)

pred <- prediction(RFDF.holdout$predict.score.RF[,2], RFDF.holdout$Attrition)

perf <- performance(pred, "tpr", "fpr")

plot(perf)

KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])

auc <- performance(pred,"auc");

auc <- as.numeric(auc@y.values)

##install.packages("ineq")

library(ineq)

gini = ineq(RFDF.holdout$predict.score.RF[,2], type="Gini")

with(RFDF.holdout, table(Attrition, predict.class.RF))

auc

KS

gini

with(RFDF.holdout, table(Attrition, predict.class.RF))

#-----------------------------------------------------#

# Building Neural Network

# split data into dev and holdout

#get categorical variables

factors.df = (data[, sapply(data, is.factor)])

#get levels of each category

lapply(factors.df, levels)

#creating dummy variables of categorical variables for NN model

#occ.matrix <- model.matrix(~ Occupation - 1, data = nn.dev)

#nn.dev <- data.frame(nn.dev, occ.matrix)

library(dummies)

data.new <- dummy.data.frame(data[,-c(9,10,22,27,36)], sep = ".")

colnames(data.new) = make.names(colnames(data.new), unique=TRUE)

View(data.new)

set.seed(1001)

data.new$random = runif(nrow(data.new),0,1)

nn.dev = data.new[which(data.new$random <= 0.7),]

nn.holdout = data.new[which(data.new$random > 0.7),]

c(nrow(nn.dev), nrow(nn.holdout))

## Target Rate

sum(nn.dev$Attrition)/nrow(nn.dev)

sum(nn.holdout$Attrition)/nrow(nn.holdout)

library(neuralnet)

library(nnet)

?"neuralnet"

#scaled data

nn.devscaled <- scale(nn.dev[,-c(2,95)])

nn.devscaled <- cbind(nn.dev[2], nn.devscaled)

View(nn.devscaled)

set.seed(1111)

names <- c(colnames(nn.devscaled)) #choose the names you want

a <- as.formula(paste('nn.devscaled$Attrition ~ ' ,paste(names[-1],collapse='+')))

nn2 <- neuralnet(formula = a,

data = nn.devscaled[,-1],

hidden = c(10,3),

err.fct = "sse",

linear.output = FALSE,

lifesign = "full",

lifesign.step = 1,

threshold = 0.1,

stepmax = 2000

##startweights = startweightsObj

)

plot(nn2)

## Assigning the Probabilities to Dev Sample

nn.dev$Prob.nn = nn2$net.result[[1]]

## The distribution of the estimated probabilities

quantile(nn.dev$Prob.nn, c(0,1,5,10,25,50,75,85,90,95,99,100)/100)

hist(nn.dev$Prob.nn)

## deciling

nn.dev$deciles <- decile(nn.dev$Prob.nn)

## Ranking code

tmp\_DT = data.table(nn.dev)

rank <- tmp\_DT[, list(

cnt = length(Attrition),

cnt\_resp = sum(Attrition),

cnt\_non\_resp = sum(Attrition == 0)) ,

by=deciles][order(-deciles)]

rank$rrate <- round (rank$cnt\_resp / rank$cnt,2);

rank$cum\_resp <- cumsum(rank$cnt\_resp)

rank$cum\_non\_resp <- cumsum(rank$cnt\_non\_resp)

rank$cum\_rel\_resp <- round(rank$cum\_resp / sum(rank$cnt\_resp),2);

rank$cum\_rel\_non\_resp <- round(rank$cum\_non\_resp / sum(rank$cnt\_non\_resp),2);

rank$ks <- abs(rank$cum\_rel\_resp - rank$cum\_rel\_non\_resp);

rank$rrate <- percent(rank$rrate)

rank$cum\_rel\_resp <- percent(rank$cum\_rel\_resp)

rank$cum\_rel\_non\_resp <- percent(rank$cum\_rel\_non\_resp)

View(rank)

?read.table

## Assgining 0 / 1 class based on certain threshold

nn.dev$Class.nn = ifelse(nn.dev$Prob.nn>0.1,1,0)

with( nn.dev, table(Attrition, as.factor(Class.nn) ))

## We can use the confusionMatrix function of the caret package

##install.packages("caret")

library(caret)

confusionMatrix(nn.dev$Attrition, nn.dev$Class.nn)

## Error Computation

sum((nn.dev$Attrition - nn.dev$Prob.nn)^2)/2

## Scoring another dataset using the Neural Net Model Object

## To score we will use the compute function

?compute

#scaled data

x.scaled <- scale(nn.holdout[,-c(2,95)])

View(x.scaled)

compute.output = compute(nn2, x.scaled)

nn.holdout$Predict.score.nn = compute.output$net.result

View(nn.holdout)

quantile(nn.holdout$Predict.score.nn, c(0,1,5,10,25,50,75,85,90,95,99,100)/100)

nn.holdout$deciles <- decile(nn.holdout$Predict.score.nn)

library(data.table)

tmp\_DT = data.table(nn.holdout)

h\_rank <- tmp\_DT[, list(

cnt = length(Attrition),

cnt\_resp = sum(Attrition),

cnt\_non\_resp = sum(Attrition == 0)) ,

by=deciles][order(-deciles)]

h\_rank$rrate <- round (h\_rank$cnt\_resp / h\_rank$cnt,2);

h\_rank$cum\_resp <- cumsum(h\_rank$cnt\_resp)

h\_rank$cum\_non\_resp <- cumsum(h\_rank$cnt\_non\_resp)

h\_rank$cum\_rel\_resp <- round(h\_rank$cum\_resp / sum(h\_rank$cnt\_resp),2);

h\_rank$cum\_rel\_non\_resp <- round(h\_rank$cum\_non\_resp / sum(h\_rank$cnt\_non\_resp),2);

h\_rank$ks <- abs(h\_rank$cum\_rel\_resp - h\_rank$cum\_rel\_non\_resp);

library(scales)

h\_rank$rrate <- percent(h\_rank$rrate)

h\_rank$cum\_rel\_resp <- percent(h\_rank$cum\_rel\_resp)

h\_rank$cum\_rel\_non\_resp <- percent(h\_rank$cum\_rel\_non\_resp)

View(h\_rank)

## Assgining 0 / 1 class based on certain threshold

nn.holdout$Class.nn = ifelse(nn.holdout$Predict.score.nn>0.1,1,0)

with( nn.holdout, table(Attrition, as.factor(Class.nn) ))

#--------------------------------------------------------------------------

**# Ensemble Modeling- Combining Random Forest and Neural Network**

final.df = cbind(RFDF.dev,nn.dev[,c(96,98,97)])

final.df$prob.score.nn = 1-final.df$Prob.nn[,1]

final.df$avg.score = (final.df$prob.score.nn + final.df$predict.score.RF[,1])/2

View(final.df)

quantile(final.df$avg.score, c(0,1,5,10,25,50,75,85,90,95,99,100)/100)

final.df$deciles.final <- decile(final.df$avg.score)

## Assgining 0 / 1 class based on certain threshold

final.df$final.class = ifelse(final.df$avg.score<0.1,1,0)

with( final.df, table(Attrition, as.factor(final.class) ))

#holdout sample

final.df.holdout = cbind(RFDF.holdout,nn.holdout[,c(96,98,97)])

View(final.df.holdout)

final.df.holdout$prob.score.nn = 1-final.df.holdout$Predict.score.nn[,1]

final.df.holdout$avg.score = (final.df.holdout$prob.score.nn + final.df.holdout$predict.score.RF[,1])/2

View(final.df.holdout)

quantile(final.df.holdout$avg.score, c(0,1,5,10,25,50,75,85,90,95,99,100)/100)

final.df.holdout$deciles.final <- decile(final.df.holdout$avg.score)

## Assgining 0 / 1 class based on certain threshold

final.df.holdout$final.class = ifelse(final.df.holdout$avg.score<0.1,1,0)

with( final.df.holdout, table(Attrition, as.factor(final.class) ))

# ensemble output does not outperform the Individual Neaural Network Model.