Human Resources Analytics

Name - Manasi Sheth Class - Stats 6620 Section - 03

Project Implementation

Step - 1 - Collecting the Data

The Human Resources Analytics Dataset is collected from Kaggle at https://www.kaggle.com/c/sm/data. This data was donated by Mr. Ludovic Benistant and contains following fields :

- Employee satisfaction level - Satisfaction Level of the Employees in the company which can be between 0 to 1.
- Last evaluation - The score which Employees received in their last evaluation
- Number of projects - The number of projects employees has received
- Average monthly hours - The average monthly hours which employees work
- Time spent at the company - Total years spend by a employee at a company
- Whether they have had a work accident - This field would have answer yes or no for question whether an employee had an accident at work or not.
- Whether they have had a promotion in the last 5 years
- Department - Department in which employee is working
- Salary - If the salary of the employee is "Low", "Medium" and "High"
- Whether the employee has left - This field is answer to the question - if the employee is still working or not for the company.

The outcome variable is "Left" which has values 0 and 1. Hence the models used will be initially Logistic Regression, then the model is improved by Decision Trees and Random Forests.

Step - 2 - Exploring and Preparing the Data

The project begins by importing the CSV data file - "HR_comma_sep.csv".

After the file is exported, First, we begin with exploring data on broader sense and obtaining basic information.

```
## [1] 14999    10
```

We can see that our data set comprises of 14999 rows and 10 columns.

Next, we take a look at high-level, non-statistical summary of entire data frame i.e. we look at the structure of the data.

```
## 'data.frame':    14999 obs. of  10 variables:
##  $ satisfaction_level   : num  0.38 0.8 0.11 0.72 0.37 0.41 0.1 0.92 0.89
0.42 ...
##  $ last_evaluation      : num  0.53 0.86 0.88 0.87 0.52 0.5 0.77 0.85 1 0.
```

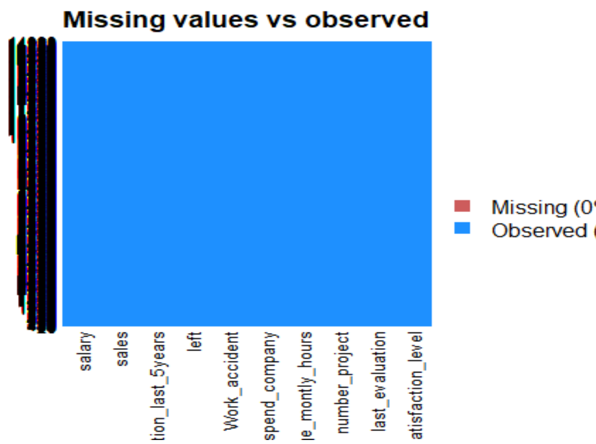
```
53 ...
##  $ number_project       : int  2 5 7 5 2 2 6 5 5 2 ...
##  $ average_montly_hours : int  157 262 272 223 159 153 247 259 224 142 ...
##  $ time_spend_company   : int  3 6 4 5 3 3 4 5 5 3 ...
##  $ Work_accident        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ left                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ promotion_last_5years: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ sales                : Factor w/ 10 levels "accounting","hr",..: 8 8 8
8 8 8 8 8 8 8 ...
##  $ salary               : Factor w/ 3 levels "high","low","medium": 2 3 3
2 2 2 2 2 2 2 ...
```

From the above results, we can see that we have 2 variables - satisfaction_level and last_evaluation of data type number, Then the variables - number_project, average_monthly_hours, time_spend_company, work_accident, left are of datatype integer and sales and salary are of type factor. Next we look at the statistical summary of the data set.

```
##  satisfaction_level last_evaluation  number_project  average_montly_hours
##  Min.   :0.0900     Min.   :0.3600   Min.   :2.000   Min.   : 96.0
##  1st Qu.:0.4400     1st Qu.:0.5600   1st Qu.:3.000   1st Qu.:156.0
##  Median :0.6400     Median :0.7200   Median :4.000   Median :200.0
##  Mean   :0.6128     Mean   :0.7161   Mean   :3.803   Mean   :201.1
##  3rd Qu.:0.8200     3rd Qu.:0.8700   3rd Qu.:5.000   3rd Qu.:245.0
##  Max.   :1.0000     Max.   :1.0000   Max.   :7.000   Max.   :310.0
##
##  time_spend_company Work_accident         left
##  Min.   : 2.000     Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 3.000     1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 3.000     Median :0.0000   Median :0.0000
##  Mean   : 3.498     Mean   :0.1446   Mean   :0.2381
##  3rd Qu.: 4.000     3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :10.000     Max.   :1.0000   Max.   :1.0000
##
##  promotion_last_5years         sales         salary
##  Min.   :0.00000       sales      :4140   high  :1237
##  1st Qu.:0.00000       technical  :2720   low   :7316
##  Median :0.00000       support    :2229   medium:6446
##  Mean   :0.02127       IT         :1227
##  3rd Qu.:0.00000       product_mng: 902
##  Max.   :1.00000       marketing  : 858
##                        (Other)    :2923
```

We can see distribution of variables in the above output. We can see that there are no NA's present in the data, hence we can say that there is no missing data in our dataset.

To confirm if there is no missing data in the dataset, Amelia package is used which has a special plotting function missmap() that will plot hr_data dataset and highlight missing values. We also confirm from sapply function that there are no missing values.




```
##    satisfaction_level       last_evaluation         number_project
##                     0                     0                      0
##  average_montly_hours    time_spend_company          Work_accident
##                     0                     0                      0
##                  left promotion_last_5years                  sales
##                     0                     0                      0
##                salary
##                     0
```

We can see that there are no missing values in the datasest from the missmap function and sapply function. Next, we find out how many values are unique in the dataset.

```
##    satisfaction_level       last_evaluation         number_project
##                    92                    65                      6
##  average_montly_hours    time_spend_company          Work_accident
##                   215                     8                      2
##                  left promotion_last_5years                  sales
##                     2                     2                     10
##                salary
##                     3
```

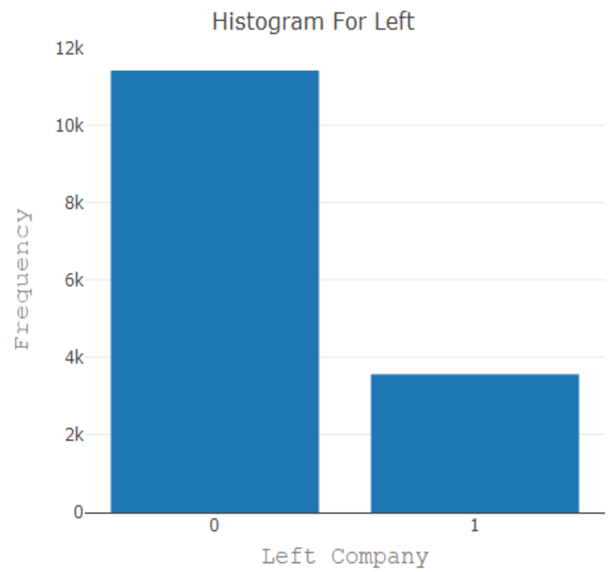
We can see that there are very less distinct values in the dataset. To model the output variable “left”, the variable is converted into factor.

Exploratory Data Aanalysis

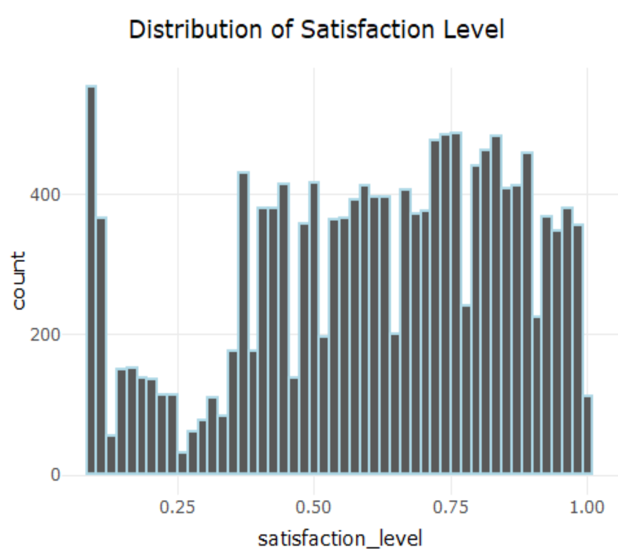
We start the Exploratory Data Analysis by seeing the exploring the categorical variable “Left”. In contrast to data, categorical data is typically examined using tables rather numeric than summary statistics. With the help of “table()” function, a one-way table is generated for “left” variable.

```
##
##     0     1
## 11428  3571
```

We can see that the number of employees working in the company are 11428 and number of employees who left the company are 3571. We can visualize the same data with histogram as follows -

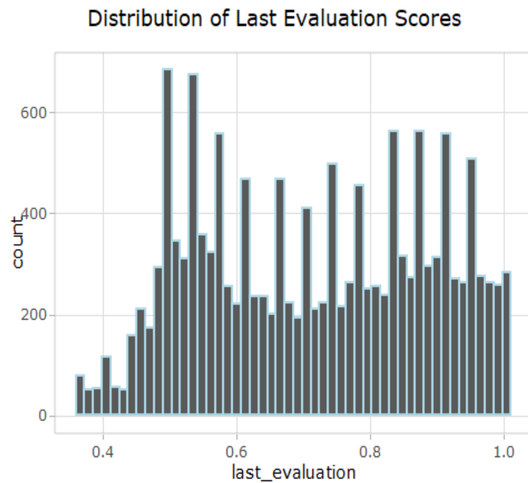


Next, we distribution of the variable “Satisfaction Level”.



From the histogram, we can see that the maximum counts of satisfaction level are for values approximately equal to 0.09. The minimum counts of satisfaction level are for values approximately equal to 0.25. There are less than 200 records with value 1.

Next, we see distribution of variable last_evaluation_score.



We can see from the histogram that the minimum value of last evaluation score is 0.36 and 77 employeees had that score. The maximum number of records are for value 0.497. The maximum last evaluation score is 1.005.

Then we answer the questions if the numeric variables are correlated.

Next, we see what is the relation between employees leaving and getting promoted in last 5 years.

```
##
##          0     1
##   0 11128   300
##   1  3552    19
```

From the tabular output and mosaic plot we can see that if the majority of the employees that are not promoted in last 5 years tend to not leave the job. However there are 3352 employees who left the job even if they were not promoted. Out of the total employees who did not leave the job after getting promoted were 300 and 19 employees who were promoted in last 5 years left the job. We can say from the graphs that promotion in last 5 years does not play mamjor role in determining if the employees would leave the company or not.

Then we explore the relationship between employees leaving and having work accident.

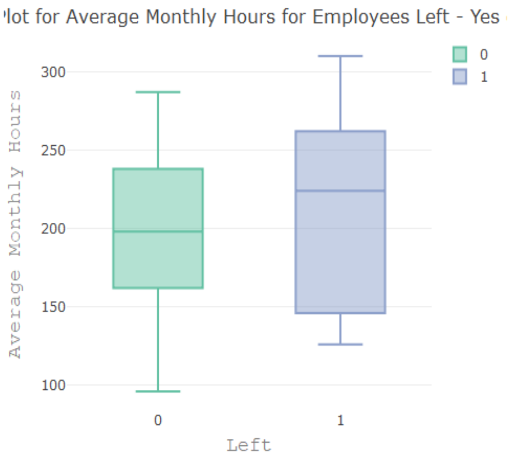
```
##
##         0    1
##   0 9428 2000
##   1 3402  169
```

We can see from the mosaic plot and tabular output that the majority of the workers who have work accident tend to not leave the job. We can see there are 2000 such records, also there are 169 records of employees who had work accident and left the job. The next question would be to see, generally after how many projects employees tend to leave the company?

```
##
##         2    3    4    5    6    7
##   0  821 3983 3956 2149  519    0
##   1 1567   72  409  612  655  256
```

We can see from the mosaic plot and the tabular output, that the majority of the employees who left the company left it after doing 2 projects. Minimum number of employees left the company after doing three projects. We can also see that there are no employees who did not left but also worked on 7 projects. There are however 256 employees that worked on 7 projects and left the company. So we can safely say that if the employees work on 7 projects, then they tend to leave the company.

We have compared numereic variables with our outcome variable using box plots.



Following are the observations from the box plot. From the first plot we can see that if the satisfaction level is low then the employees have left the job. From the second box plot we can that the median value of the last evaluation score is high for the people who have left the job. From the third box plot we can see that people how have left the job tend to put in more hours. From the fourth plot we can see that there is no specific trend with respect to time spent in the company.

Next, the data is split into test and training dataset to build the logistic regression model and to evaluate the performance of the model on new data. The data is randomized, and the first 90% is used for training and the rest of the data is used for testing.

Step - 3 - Training a logistic regression model on the data

In this section we begin by training the logistic regression model using glm function.

The logistic regression model looks as follows:

```
##
## Call:  glm(formula = left ~ ., family = binomial(link = "logit"), data = h
r_train)
##
## Coefficients:
##          (Intercept)     satisfaction_level        last_evaluation
##            -1.485290              -4.116651               0.666263
##       number_project   average_montly_hours     time_spend_company
##            -0.304008               0.004732               0.263469
##        Work_accident  promotion_last_5years                saleshr
##            -1.499329              -1.359880               0.263300
##              salesIT        salesmanagement         salesmarketing
##            -0.207982              -0.535651              -0.055261
##     salesproduct_mng             salesRandD             salessales
##            -0.221588              -0.534263              -0.025812
##         salessupport         salestechnical              salarylow
##             0.041641               0.100681               1.890103
```

```
##          salarymedium
##              1.383615
##
## Degrees of Freedom: 13498 Total (i.e. Null);  13480 Residual
## Null Deviance:      14820
## Residual Deviance: 11590    AIC: 11620
```

We can see the intercept values and the values of slopes for different variables in the data set. The model has 13498 degress of freedom and the AIC value is 11620.

By using function summary(), we obtain the results of the model.

```
##
## Call:
## glm(formula = left ~ ., family = binomial(link = "logit"), data = hr_train
)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2313  -0.6663  -0.4037  -0.1189   3.0298
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -1.4852896  0.2027110  -7.327 2.35e-13 ***
## satisfaction_level    -4.1166506  0.1030029 -39.966  < 2e-16 ***
## last_evaluation        0.6662632  0.1568365   4.248 2.16e-05 ***
## number_project        -0.3040083  0.0224704 -13.529  < 2e-16 ***
## average_montly_hours   0.0047321  0.0005449   8.684  < 2e-16 ***
## time_spend_company     0.2634692  0.0163591  16.105  < 2e-16 ***
## Work_accident         -1.4993285  0.0933577 -16.060  < 2e-16 ***
## promotion_last_5years -1.3598804  0.2658466  -5.115 3.13e-07 ***
## saleshr                0.2633004  0.1374765   1.915 0.055462 .
## salesIT               -0.2079819  0.1290126  -1.612 0.106939
## salesmanagement       -0.5356511  0.1703439  -3.145 0.001664 **
## salesmarketing        -0.0552614  0.1399913  -0.395 0.693028
## salesproduct_mng      -0.2215877  0.1383656  -1.601 0.109274
## salesRandD            -0.5342629  0.1515505  -3.525 0.000423 ***
## salessales            -0.0258120  0.1081589  -0.239 0.811378
## salessupport           0.0416408  0.1152865   0.361 0.717954
## salestechnical         0.1006805  0.1123383   0.896 0.370132
## salarylow              1.8901026  0.1327844  14.234  < 2e-16 ***
## salarymedium           1.3836153  0.1335604  10.359  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 14823  on 13498  degrees of freedom
## Residual deviance: 11585  on 13480  degrees of freedom
## AIC: 11623
```

```
##
## Number of Fisher Scoring iterations: 5
```

Now we have run the ANOVA function on the model to analyze the table of deviance.

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: left
##
## Terms added sequentially (first to last)
##
##
##                       Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                 13498      14823
## satisfaction_level     1  2051.38     13497      12772 < 2.2e-16 ***
## last_evaluation        1    18.11     13496      12754 2.084e-05 ***
## number_project         1    89.79     13495      12664 < 2.2e-16 ***
## average_montly_hours   1    82.71     13494      12581 < 2.2e-16 ***
## time_spend_company     1   164.76     13493      12416 < 2.2e-16 ***
## Work_accident          1   344.24     13492      12072 < 2.2e-16 ***
## promotion_last_5years  1    60.79     13491      12011 6.363e-15 ***
## sales                  9    91.91     13482      11919 6.723e-16 ***
## salary                 2   334.14     13480      11585 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

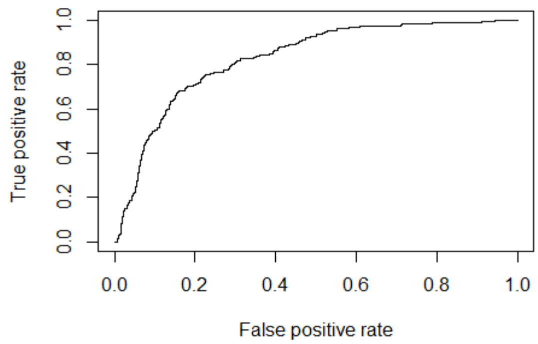
From the anova output we can see that all the variables are significant. Since all the variables are significant, we will not remove any variables and this will be the final model. The difference between the null deviance and the residual deviance shows how our model is doing against the null model. The wider this gap, the better. Analyzing the ANOVA output we can see that as we add each variable one at a time the residual deviance has dropped.

Step - 4 - Evaluating Model Performance

Now, we would like to see the model performance for predicting with new data set. By setting the parameter type='reponse', R will output probabilities in form of P(y=1|X). Our decision boundary is 0.5. If P(y=1|X)>0.5 then y=1 otherwise y=0.

```
## [1] "Accuracy 0.796666666666667"
```

The accuracy of the model is 0.79 which is a good result. We are going to plot the ROC curve and calculate the AUC (area under the curve), which are typical performance measurements for the binary classifiers. As a rule of thumb, a model with good predictive ability should have an AUC closer to 1, than to 0.5.



A popular way for summarizing the discrimination ability of the model is to report the area under the ROC curve. In a model with good discrimination ability the ROC curve will go closer to the left corner. We have calculated the AUC to estimate the model's predictive ability.

```
## [1] 0.8288136
```

Since the AUC is 0.82 we can say that model has good predictive abilities.

However this is result is somewhat dependent on the manual split of the data we did earlier. So we will be looking at improving the model performance.

Step 5 - Improving Model Performance

To improve the model performance we have first constructed the decision trees, then boosted them and then created a random forest model.

Training and Testing Data for Decision Trees

The same training and testing dataset is used for decision tree.

create a model for decision trees

The decision tree is build using c5.0 algorithm. The model is created by excluding the 'left' class variable from the training data set. The 'left' variable is set as target factor vector for classification.

The basic data about the tree is as follows:

```
##
## Call:
## C5.0.default(x = hr_c50_train[-7], y = hr_c50_train$left)
##
## Classification Tree
## Number of samples: 13499
## Number of predictors: 9
##
## Tree size: 41
```

```
##
## Non-standard options: attempt to group attributes
```

From the output it can be seen that the tree size 41, which means that the tree is 41 decisions deep. The confusion matrix from the training dataset is as follows:

```
Evaluation on training data (13499 cases):

	    Decision Tree
	  ----------------
	  Size      Errors

	    41  246( 1.8%)   <<


	   (a)   (b)    <-classified as
	  ----  ----
	 10255    28    (a): class 0
	   218  2998    (b): class 1


	Attribute usage:

	100.00% average_montly_hours
	 97.76% satisfaction_level
	 72.15% time_spend_company
	 34.98% last_evaluation
	 32.43% number_project
	  2.31% Work_accident
	  1.67% sales
	  0.38% salary


Time: 0.2 secs
```

The confusion matrix has displayed the incorrectly classified records. It can be seen that out of 13499 records, 246 records are incorrectly classified giving error rate of 1.8%. 28 values which were actually employees in the company were wrongly classified as left and 218 employees who left were misclassified as working.

Decision Trees prediction

Using the predict function the decision tree is applied to test data set.

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1500
##
##
##                | predicted default
## actual default |         0 |         1 | Row Total |
## ---------------|-----------|-----------|-----------|
```

```
##               0 |      1142 |         3 |      1145 |
##                 |     0.761 |     0.002 |           |
## ---------------|-----------|-----------|-----------|
##               1 |        19 |       336 |       355 |
##                 |     0.013 |     0.224 |           |
## ---------------|-----------|-----------|-----------|
##    Column Total |      1161 |       339 |      1500 |
## ---------------|-----------|-----------|-----------|
## 
## 
```

From the confusion matrix it can be seen that out of 1500 records 22 records were misclassified. This resulted in an accuracy of 98.5 and an error rate of 1.5%. 3 values which were actually employees in the company were wrongly classified as left and 19 employees who left were misclassified as working.

boosted decision trees

Next we have used boosted decision trees to improve the model performance of decision trees. The number of trials are set to 10.

```
## 
## Call:
## C5.0.default(x = hr_c50_train[-7], y = hr_c50_train$left, trials = 10)
## 
## Classification Tree
## Number of samples: 13499 
## Number of predictors: 9 
## 
## Number of boosting iterations: 10 
## Average tree size: 45.1 
## 
## Non-standard options: attempt to group attributes
```

We can see that the average tree size has increased from 41 to 45.1 by using boosted decision trees. The confusion matrix for the boosted decision tree’s training dataset is as follows –

```
Evaluation on training data (13499 cases):

Trial	    Decision Tree
-----	  ----------------
	  Size      Errors

   0	    41  246( 1.8%)
   1	    44 1311( 9.7%)
   2	    38  800( 5.9%)
   3	    40  883( 6.5%)
   4	    55 1272( 9.4%)
   5	    55  674( 5.0%)
   6	    45  880( 6.5%)
   7	    43  462( 3.4%)
   8	    40 1931(14.3%)
   9	    50  433( 3.2%)
boost	        176( 1.3%)   <<


	   (a)   (b)    <-classified as
	  ----  ----
	 10255    28    (a): class 0
	   148  3068    (b): class 1


	Attribute usage:

	100.00% satisfaction_level
	100.00% last_evaluation
	100.00% number_project
	100.00% average_montly_hours
	 99.33% time_spend_company
	 85.69% Work_accident
	 83.78% sales
	 81.02% salary
	 57.80% promotion_last_5years


Time: 0.9 secs
```

The confusion matrix has displayed the incorrectly classified records. It can be seen that out of 13499 records, 176 records are incorrectly classified giving error rate of 1.3%. 28 values which were actually employees in the company were wrongly classified as left and 148 employees who left were misclassified as working.

Using the predict function the boosted decision tree is applied to test data set.

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  1500 
## 
## 
##                | predicted default 
## actual default |         0 |         1 | Row Total | 
## ---------------|-----------|-----------|-----------|
##              0 |      1141 |         4 |      1145 | 
##                |     0.761 |     0.003 |           | 
## ---------------|-----------|-----------|-----------|
```

```
##              1 |        14 |       341 |       355 |
##                |     0.009 |     0.227 |           |
## ---------------|-----------|-----------|-----------|
##   Column Total |      1155 |       345 |      1500 |
## ---------------|-----------|-----------|-----------|
##
##
```

From the confusion matrix it can be seen that out of 1500 records 18 records were misclassified. This resulted in an accuracy of 98.8 and an error rate of 1.2%. 4 values which were actually employees in the company were wrongly classified as left and 14 employees who left were misclassified as working.

create a model for random forests

Next we evaluate random forsests to improve the model. We keep the same training and test dataset.

Training and Testing Data For Random Forests

The random forest model is fitted using randomForest() function in the randomForest package.

```
## 11.02 sec elapsed

##
## Call:
##  randomForest(formula = left ~ ., data = hr_rf_train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0.81%
## Confusion matrix:
##       0    1 class.error
## 0 10265   18 0.001750462
## 1    91 3125 0.028296020
```

The output shows that the random forest included 500 trees and tried 3 variables at each split. The out-of-bag error rate is 0.81%, which is an unbiased estimate of the test set error. The error rate in the confusion matrix is same 0.8%.

Prediction of data using Random Forests

The random forest performance is evaluated using the predict() function.

```
## [1] "Accuracy  0.994666666666667"
```

The accuracy of the random forest model is 99.4%, which is higher than logistic regression, decision tree and boosted decision tree.

Conclusion

- The accuracy of the logistic regression is 0.8288 or approximately 83%
- The accuracy of the decision tree is 98.5% which is significantly higher than the logistic regression.
- The accuracy of the boosted decision tree with trials = 10 is 98.8%
- The accuracy of the random forest is 99.4% which is highest among all the models. The error rate of the confusion matrix is also the lowest for the random forest both for training as well as testing dataset
- The future work would involve trying the dataset on SVM and neural networks

Appendix

The R Code for the project is as follows –

Step - 2 - Exploring and Preparing the Data

```
suppressWarnings(library(ggplot2))
suppressWarnings(library(gmodels))
suppressWarnings(library(caret))
suppressWarnings(library(C50))
suppressWarnings(library(tidyverse,quietly = TRUE))
suppressWarnings(library(corrplot,quietly = TRUE))
suppressWarnings(library(stringr,quietly = TRUE))
suppressWarnings(library(Hmisc,quietly = TRUE))
```

Read csv file

```
hr_data <- read.csv("HR_comma_sep.csv")
```

shape of dataframe

```
#shape of DF
dim(hr_data)
nrow(hr_data)
ncol(hr_data)
```

names of columns

```
names(hr_data)
```

structure of the data

```
str(hr_data)
```

summary of the data

```
summary(hr_data)
```

missmap function

```r
library(Amelia)

missmap(hr_data, main = "Missing values vs observed")
```

sapply function

```r
sapply(hr_data,function(x) sum(is.na(x)))
```

unique values for each function

```r
sapply(hr_data, function(x) length(unique(x)))
```

convert "left" to factor

```r
hr_data$left <- factor(hr_data$left)
```

Exploratory Data Aanalysis

one-way table for left

```r
table(hr_data$left)
```

histogram for left

```r
library(plotly)
f <- list(
  family = "Courier New, monospace",
  size = 18,
  color = "#7f7f7f"
)

plot_ly(x = hr_data$left, type = "histogram") %>%
  layout(xaxis = list(title = "Left Company", titlefont = f),
         yaxis = list(title = "Frequency", titlefont = f),
         title = "Histogram For Left")
```

Distribution of satisfaction level

```r
p <- ggplot(hr_data, aes(satisfaction_level)) +
  geom_histogram(bins = 50, color = "lightblue") +
  ggtitle("Distribution of Satisfaction Level") +
    theme_minimal()
ggplotly(p)
```

Distribution of last evaluation score

```r
p <- ggplot(hr_data, aes(last_evaluation)) +
  geom_histogram(bins = 50, color = "lightblue") +
  ggtitle("Distribution of Last Evaluation Scores") +
  theme_light()

ggplotly(p)
```

correlation matrix

```r
cor_matrix <- cor(select_if(hr_data,is.numeric))
corrplot(cor_matrix,method = "number",mar = c(3,3,3,3))
```

mosaic map for left and promotion in last 5 years

```
test_data <-table(hr_data$left,hr_data$promotion_last_5years)
test_data

mosaicplot(test_data, xlab = "Employees Left - Yes or No", ylab = "Promoted i
n last 5 years", color = TRUE)
```

mosaic map for left and work accident

```
test_data <- table(hr_data$left, hr_data$Work_accident)

test_data

mosaicplot(test_data, xlab = "Employees Left - Yes or No", ylab = "Had Work A
ccident - Yes or No", color = TRUE)
```

mosaic map for left and number of projects

```
test_data <- table(hr_data$left, hr_data$number_project)

test_data

mosaicplot(test_data, xlab = "Employees Left - Yes or No", ylab = "Number of
Projects",color = TRUE)
```

boxplots for numeric variables with respect to left variable

```
par(mfrow=c(2,3))

plot_ly(hr_data, y = ~satisfaction_level, color = ~left,
        type = "box")%>%
       layout(xaxis = list(title = "Left", titlefont = f),
              yaxis = list(title = "Satisfaction Level", titlefont = f),
              title = "Box Plot for Satisfaction Level for Employees Left - Y
es or No")

plot_ly(hr_data, y = ~last_evaluation, color = ~left,
        type = "box")%>%
       layout(xaxis = list(title = "Left", titlefont = f),
              yaxis = list(title = "Last Evaluation", titlefont = f),
              title = "Box Plot for Last Evaluation for Employees Left - Yes
or No")
plot_ly(hr_data, y = ~average_montly_hours, color = ~left,
        type = "box")%>%
       layout(xaxis = list(title = "Left", titlefont = f),
              yaxis = list(title = "Average Monthly Hours", titlefont = f),
              title = "Box Plot for Average Monthly Hours for Employees Left
- Yes or No")

plot_ly(hr_data, y = ~time_spend_company, color = ~left,
        type = "box")%>%
       layout(xaxis = list(title = "Left", titlefont = f),
              yaxis = list(title = "Time Spend In Company", titlefont = f),
```

```
                title = "Box Plot for Time Spend in Company for Employees that have Left - Yes or No")
```

scatterplot of satisfaction levels versus last evaluation

```
p <-ggplot(aes(y = satisfaction_level, x = last_evaluation), data = hr_data) +
    geom_point(aes(color = left,
                   alpha = 0.05), size = 1.0) +
    ggtitle("Satisfaction Levels versus Last Evaluation")
ggplotly(p)
```

scatterplot of satisfaction levels versus average monthly hours

```
p <-ggplot(aes(y = satisfaction_level, x = average_montly_hours), data = hr_data) +
    geom_point(aes(color = left,
                   alpha = 0.05), size = 1.0) +
    ggtitle("Satisfaction Levels versus average monthly hours")
ggplotly(p)
```

get 90% of random sample of data

```
set.seed(300)

indx <- sample(1:nrow(hr_data), as.integer(0.9*nrow(hr_data)))
```

split the data into training and testing

```
hr_train <- hr_data[indx,]
hr_test <- hr_data[-indx,]
```

Step - 3 - Training a logistic regression model on the data

train a logistic model on the data

```
model <- glm(left ~.,family=binomial(link='logit'),data=hr_train)
```

look at the model

```
model
```

summary of the model

```
summary(model)
```

anova of the model

```
anova(model, test="Chisq")
```

Step - 4 - Evaluating Model Performance

Logistic Regression

```
fitted.results <- predict(model,newdata=hr_test,type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)
misClasificError <- mean(fitted.results != hr_test$left)
print(paste('Accuracy',1-misClasificError))
```

Draw ROC and AUC curve

```
library(ROCR)

p <- predict(model, newdata=hr_test, type="response")
pr <- prediction(p, hr_test$left)
prf.glm <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf.glm)

auc <- performance(pr, measure = "auc")
auc.glm <- auc@y.values[[1]]
auc.glm
```

Step 5 - Improving Model Performance

Training and Testing Data for Decision Trees

```
hr_c50_train = hr_data[indx,]
hr_c50_test = hr_data[-indx,]
```

create a model for decision trees

```
hr_c50_model <- C5.0(hr_c50_train[-7], hr_c50_train$left)
```

take a look at decision tree model

```
hr_c50_model
```

summary of the model

```
summary(hr_c50_model)
```

Decision Trees prediction

```
hr_c50_pred <- predict(hr_c50_model, hr_c50_test)

CrossTable(hr_c50_test$left, hr_c50_pred,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

boosted decision trees

```
hr_c50_boost10 <- C5.0(hr_c50_train[-7], hr_c50_train$left,
                       trials = 10)
hr_c50_boost10
```

summary of the boosted decision tree

```
summary(hr_c50_boost10)
```

prediction of boosted decision tree

```
hr_c50_boost_pred10 <- predict(hr_c50_boost10, hr_c50_test)
CrossTable(hr_c50_test$left, hr_c50_boost_pred10,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

create a model for random forests

```
hr_rf_train_labels = hr_data[indx,7]
hr_rf_test_labels = hr_data[-indx,7]
```

Training and Testing Data For Random Forests

```
hr_rf_train = hr_data[indx,]
hr_rf_test = hr_data[-indx,]
```

train the model on the dataset

```
library(randomForest)
library(tictoc)  # A nice package for measuring run times in R.

# fit the random forest model, with all predictor variables
tic()
set.seed(300)
rf <- randomForest(left ~ . , data = hr_rf_train)
toc()
## 0.59 sec elapsed
rf
```

predictions for random forest

```
# predicted model
pred <- predict(rf, newdata = hr_rf_test)

#Accuracy
acc <- sum(pred==hr_rf_test$left) / nrow(hr_rf_test)

print(paste('Accuracy ',acc))
```