Final project Group 7

Bharath Reddy Madi
Prasanna Krishna Reddy Jeedipally
Divya Sharvani Kandukuri

2022-12-04

Problem: Analysis of Employee Attrition(Employee Churn Prediction)

Abstract:

A corporation may suffer from voluntary employee attrition in a number of ways, including increased labor costs, lowered employee morale, the loss of intellectual property and talent to rivals, etc. Therefore, it is crucial to spot each employee who has a propensity to leave the organization in order to prevent a future loss. Conventional procedures rely on qualitative evaluation of variables that may suggest an employee's propensity to leave a company. For instance, research has shown that staff turnover is related to both demographic data and behavioral activities, satisfaction, etc. Data-driven approaches that are based on statistical learning techniques show more accurate prediction of employee attrition because, by their very nature, they mathematically model the relationship between factors and attrition outcome and maximize the probability of predicting the right group of people using a properly trained machine learning model.

1. Introduction:

Employee churn is described as a decision made voluntarily by an employee to leave their employment or retire, necessitating the hiring of a new applicant. People frequently leave their jobs for a variety of reasons, including a feeling of lacking coaching and feedback, a lack of growth, commuting time, an unsatisfactory pay scale, a sense of devaluation, work stress, a lack of balance between work and life, a lack of trust in their boss, etc. According to a survey released by LinkedIn Talent Solutions in 2018, the IT sector leads all other sectors with a 13.2% employee turnover rate. We created four different models, one using Boosted Logistic Regression, Support Vector Machine, Random Forest, and the stack of all the above models, to optimize the performance of the models we used to predict whether an employee will leave or not in order to address the employee turnover issue. We explored and cleaned the IBM HR dataset that we downloaded from www.kaagle.com. We perfrmed feature selection in order to use only attributes that helps in model prediction. Employees are traditionally seen as important company assets. The best performers and especially those who have worked for a longer period of time are regarded as special employees. Compared to regular employees, businesses suffer more loss when exceptional employees opt to leave. When a senior and knowledgeable person leaves, it can have a psychological impact on the team, which lowers team morale.

2. Data Description:

Dataset:

Source-> https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset

This is a fictional data set created by IBM data scientists.

Attributes in the dataset: There are total 35 attributes in the dataset The dimensions of the data set is 1470 * 35

The attributes are:

```
1 "Age"
2 "Attrition"
3 "BusinessTravel"
4 "DailyRate"
5 "Department"
6 "DistanceFromHome"
7 "Education"
8 "EducationField"
9 "EmployeeCount"
10 "EmployeeNumber"
11 "EnvironmentSatisfaction"
12 "Gender"
13 "HourlyRate"
14 "JobInvolvement"
15 "JobLevel"
16 "JobRole"
17 "JobSatisfaction"
18 "MaritalStatus"
19 "MonthlyIncome"
20 "MonthlyRate"
21 "NumCompaniesWorked"
22 "Over18"
23 "OverTime"
24 "PercentSalaryHike"
25 "PerformanceRating"
26 "RelationshipSatisfaction"
27 "StandardHours"
28 "StockOptionLevel"
29 "TotalWorkingYears"
30 "TrainingTimesLastYear"
31 "WorkLifeBalance"
32 "YearsAtCompany"
33 "YearsInCurrentRole"
34 "YearsSinceLastPromotion"
35 "YearsWithCurrManager"
```

1. Data-driven analytics for HR attrition prediction

```
# importing all the required libraries required for our project

library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(magrittr)
library(stringr)
library(stringi)
library(readr)
library(DMwR2)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(caretEnsemble)
```

```
##
## Attaching package: 'caretEnsemble'
```

```
## The following object is masked from 'package:ggplot2':
##
##     autoplot
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(httr)
```

```
##
## Attaching package: 'httr'
```

```
## The following object is masked from 'package:caret':
##
##     progress
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(ggplot2)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
#reading the dataset
DATA1 <- "/Employee_attrition.csv"
```

```r
#storing the data in the dataframe
df <- read_csv("Employee_attrition.csv")
```

```
## Rows: 1470 Columns: 35
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (9): Attrition, BusinessTravel, Department, EducationField, Gender, Job...
## dbl (26): Age, DailyRate, DistanceFromHome, Education, EmployeeCount, Employ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#Displaying top 5 rows of the dataframe
head(df)
```

```
## # A tibble: 6 x 35
##     Age Attrit~1 Busin~2 Daily~3 Depar~4 Dista~5 Educa~6 Educa~7 Emplo~8 Emplo~9
##   <dbl> <chr>    <chr>     <dbl> <chr>     <dbl>   <dbl> <chr>     <dbl>   <dbl>
## 1    41 Yes      Travel~    1102 Sales         1       2 Life S~       1       1
## 2    49 No       Travel~     279 Resear~       8       1 Life S~       1       2
## 3    37 Yes      Travel~    1373 Resear~       2       2 Other         1       4
## 4    33 No       Travel~    1392 Resear~       3       4 Life S~       1       5
## 5    27 No       Travel~     591 Resear~       2       1 Medical       1       7
## 6    32 No       Travel~    1005 Resear~       2       2 Life S~       1       8
## # ... with 25 more variables: EnvironmentSatisfaction <dbl>, Gender <chr>,
## #   HourlyRate <dbl>, JobInvolvement <dbl>, JobLevel <dbl>, JobRole <chr>,
## #   JobSatisfaction <dbl>, MaritalStatus <chr>, MonthlyIncome <dbl>,
## #   MonthlyRate <dbl>, NumCompaniesWorked <dbl>, Over18 <chr>, OverTime <chr>,
## #   PercentSalaryHike <dbl>, PerformanceRating <dbl>,
## #   RelationshipSatisfaction <dbl>, StandardHours <dbl>,
## #   StockOptionLevel <dbl>, TotalWorkingYears <dbl>, ...
```

```r
#Displaying the dimensions of the data frame
dim(df)
```

```
## [1] 1470   35
```

```r
#Displaying the column names of the data frame
names(df)
```

```
##  [1] "Age"                      "Attrition"
##  [3] "BusinessTravel"           "DailyRate"
##  [5] "Department"               "DistanceFromHome"
##  [7] "Education"                "EducationField"
##  [9] "EmployeeCount"            "EmployeeNumber"
## [11] "EnvironmentSatisfaction"  "Gender"
## [13] "HourlyRate"               "JobInvolvement"
## [15] "JobLevel"                 "JobRole"
## [17] "JobSatisfaction"          "MaritalStatus"
## [19] "MonthlyIncome"            "MonthlyRate"
## [21] "NumCompaniesWorked"       "Over18"
## [23] "OverTime"                 "PercentSalaryHike"
## [25] "PerformanceRating"        "RelationshipSatisfaction"
## [27] "StandardHours"            "StockOptionLevel"
## [29] "TotalWorkingYears"        "TrainingTimesLastYear"
## [31] "WorkLifeBalance"          "YearsAtCompany"
```

```
## [33] "YearsInCurrentRole"       "YearsSinceLastPromotion"
## [35] "YearsWithCurrManager"
```

```
str(df)
```

```
## spc_tbl_ [1,470 x 35] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Age                     : num [1:1470] 41 49 37 33 27 32 59 30 38 36 ...
##  $ Attrition               : chr [1:1470] "Yes" "No" "Yes" "No" ...
##  $ BusinessTravel          : chr [1:1470] "Travel_Rarely" "Travel_Frequently" "Travel_Rarely" "Travel
##  $ DailyRate               : num [1:1470] 1102 279 1373 1392 591 ...
##  $ Department              : chr [1:1470] "Sales" "Research & Development" "Research & Development"
##  $ DistanceFromHome        : num [1:1470] 1 8 2 3 2 2 3 24 23 27 ...
##  $ Education               : num [1:1470] 2 1 2 4 1 2 3 1 3 3 ...
##  $ EducationField          : chr [1:1470] "Life Sciences" "Life Sciences" "Other" "Life Sciences" ..
##  $ EmployeeCount           : num [1:1470] 1 1 1 1 1 1 1 1 1 1 ...
##  $ EmployeeNumber          : num [1:1470] 1 2 4 5 7 8 10 11 12 13 ...
##  $ EnvironmentSatisfaction : num [1:1470] 2 3 4 4 1 4 3 4 4 3 ...
##  $ Gender                  : chr [1:1470] "Female" "Male" "Male" "Female" ...
##  $ HourlyRate              : num [1:1470] 94 61 92 56 40 79 81 67 44 94 ...
##  $ JobInvolvement          : num [1:1470] 3 2 2 3 3 3 4 3 2 3 ...
##  $ JobLevel                : num [1:1470] 2 2 1 1 1 1 1 1 3 2 ...
##  $ JobRole                 : chr [1:1470] "Sales Executive" "Research Scientist" "Laboratory Technici
##  $ JobSatisfaction         : num [1:1470] 4 2 3 3 2 4 1 3 3 3 ...
##  $ MaritalStatus           : chr [1:1470] "Single" "Married" "Single" "Married" ...
##  $ MonthlyIncome           : num [1:1470] 5993 5130 2090 2909 3468 ...
##  $ MonthlyRate             : num [1:1470] 19479 24907 2396 23159 16632 ...
##  $ NumCompaniesWorked      : num [1:1470] 8 1 6 1 9 0 4 1 0 6 ...
##  $ Over18                  : chr [1:1470] "Y" "Y" "Y" "Y" ...
##  $ OverTime                : chr [1:1470] "Yes" "No" "Yes" "Yes" ...
##  $ PercentSalaryHike       : num [1:1470] 11 23 15 11 12 13 20 22 21 13 ...
##  $ PerformanceRating       : num [1:1470] 3 4 3 3 3 3 4 4 4 3 ...
##  $ RelationshipSatisfaction: num [1:1470] 1 4 2 3 4 3 1 2 2 2 ...
##  $ StandardHours           : num [1:1470] 80 80 80 80 80 80 80 80 80 80 ...
##  $ StockOptionLevel        : num [1:1470] 0 1 0 0 1 0 3 1 0 2 ...
##  $ TotalWorkingYears       : num [1:1470] 8 10 7 8 6 8 12 1 10 17 ...
##  $ TrainingTimesLastYear   : num [1:1470] 0 3 3 3 3 2 3 2 2 3 ...
##  $ WorkLifeBalance         : num [1:1470] 1 3 3 3 3 2 2 3 3 2 ...
##  $ YearsAtCompany          : num [1:1470] 6 10 0 8 2 7 1 1 9 7 ...
##  $ YearsInCurrentRole      : num [1:1470] 4 7 0 7 2 7 0 0 7 7 ...
##  $ YearsSinceLastPromotion : num [1:1470] 0 1 0 3 2 3 0 0 1 7 ...
##  $ YearsWithCurrManager    : num [1:1470] 5 7 0 0 2 6 0 0 8 7 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Age = col_double(),
##   ..   Attrition = col_character(),
##   ..   BusinessTravel = col_character(),
##   ..   DailyRate = col_double(),
##   ..   Department = col_character(),
##   ..   DistanceFromHome = col_double(),
##   ..   Education = col_double(),
##   ..   EducationField = col_character(),
##   ..   EmployeeCount = col_double(),
##   ..   EmployeeNumber = col_double(),
##   ..   EnvironmentSatisfaction = col_double(),
##   ..   Gender = col_character(),
```

```
##   ..   HourlyRate = col_double(),
##   ..   JobInvolvement = col_double(),
##   ..   JobLevel = col_double(),
##   ..   JobRole = col_character(),
##   ..   JobSatisfaction = col_double(),
##   ..   MaritalStatus = col_character(),
##   ..   MonthlyIncome = col_double(),
##   ..   MonthlyRate = col_double(),
##   ..   NumCompaniesWorked = col_double(),
##   ..   Over18 = col_character(),
##   ..   OverTime = col_character(),
##   ..   PercentSalaryHike = col_double(),
##   ..   PerformanceRating = col_double(),
##   ..   RelationshipSatisfaction = col_double(),
##   ..   StandardHours = col_double(),
##   ..   StockOptionLevel = col_double(),
##   ..   TotalWorkingYears = col_double(),
##   ..   TrainingTimesLastYear = col_double(),
##   ..   WorkLifeBalance = col_double(),
##   ..   YearsAtCompany = col_double(),
##   ..   YearsInCurrentRole = col_double(),
##   ..   YearsSinceLastPromotion = col_double(),
##   ..   YearsWithCurrManager = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

3. Data Visualization:

The basic objectives of data visualization are to extract and transform huge volumes of complex information into a visual environment, such as a graph or chart, and to facilitate understanding or interpretation of this information. When presented as figures or spreadsheets, data can be challenging to analyze, especially when presented in huge volumes. Complex datasets are converted into a combination of understandable graphics and information via data visualization. The capacity to quickly convey information to an audience and enable them to determine outcomes or develop predictions in light of the interpreted facts. The use of effective data visualizations allows marketing teams to assess their results swiftly and move on to other campaigns or ideas.

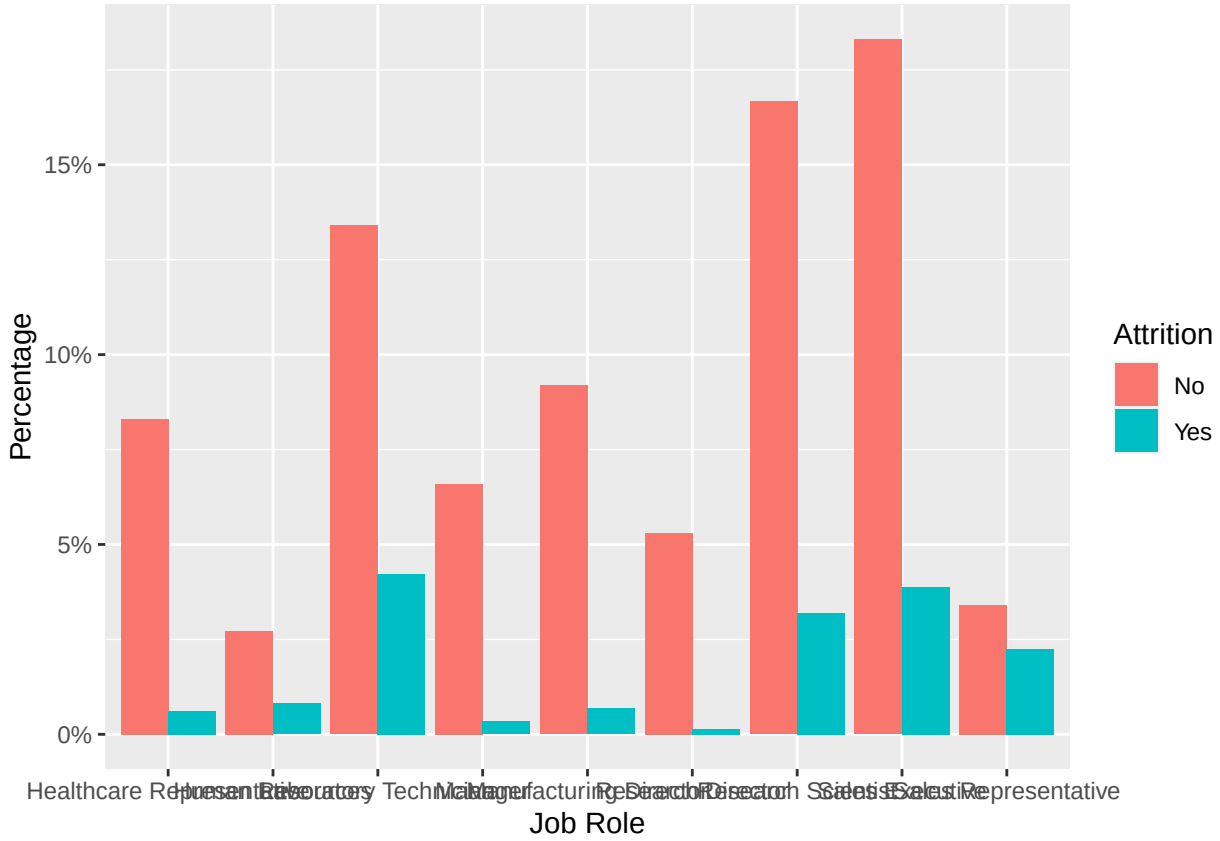
The capacity to concisely and clearly illustrate complicated data linkages. As a result, graphs and charts feature distinct headings and descriptions that provide a streamlined study of vast volumes of complex data.

Initial exploratory analysis can be performed to understand the data set. For example,

1. the proportion of employees with different job titles (or any other possible factor) for status of "attrition" and "non-attrition" may vary, and this can be plotted as follows. People titled "Laboratory Technician", "Sales Executive", and "Research Scientist" are among the top 3 groups that exhibit highest attrition rate.

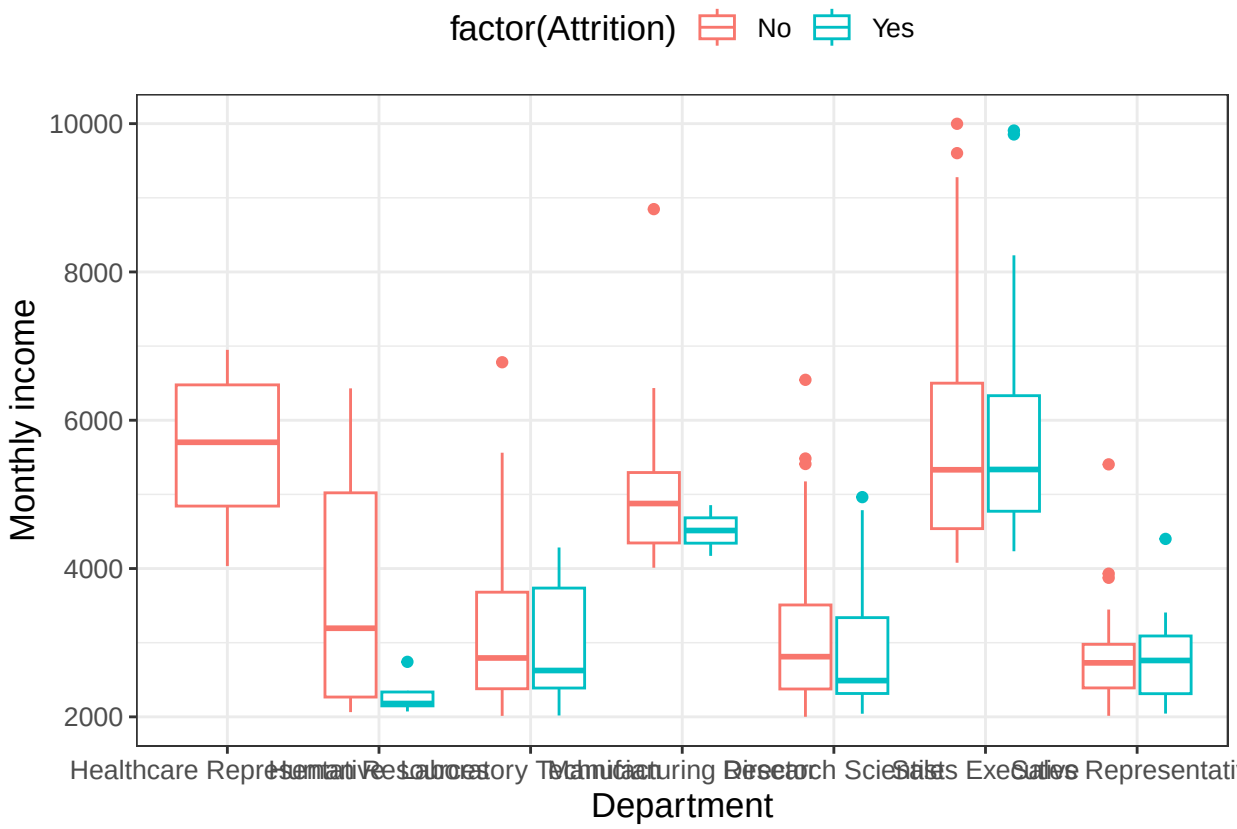
```
#Visualization of the data
#ggplot is based on the grammar of Graphics

ggplot(df, aes(JobRole, fill=Attrition)) +
  geom_bar(aes(y=(..count..)/sum(..count..)), position="dodge") +
  scale_y_continuous(labels=percent) +
  xlab("Job Role") +
  ylab("Percentage")
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
```



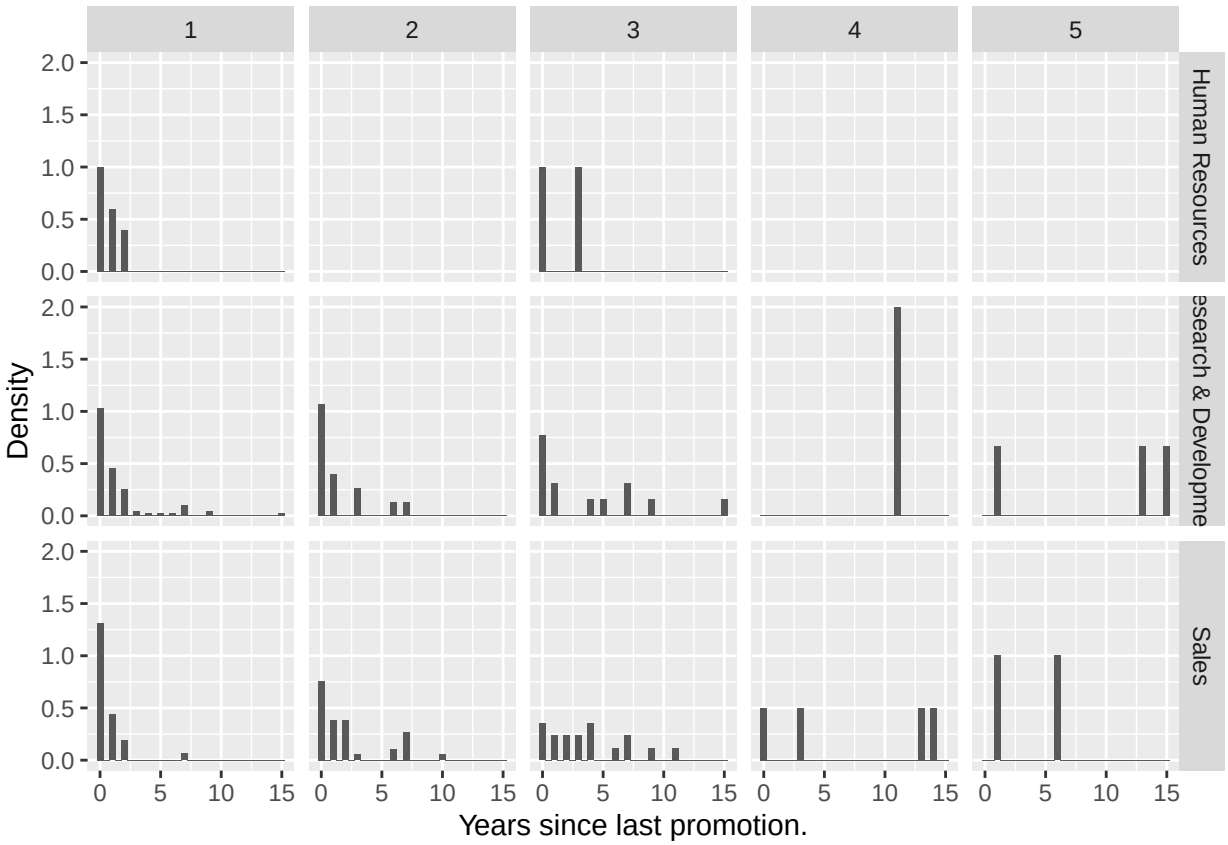
2. monthly income, job level, and service year may affect decision of leaving for employees in different departments. For example, junior staffs with lower pay will be more likely to leave compared to those who are paid higher.

```
ggplot(filter(df, (YearsAtCompany >= 2) & (YearsAtCompany <= 5) & (JobLevel < 3)),
       aes(x=factor(JobRole), y=MonthlyIncome, color=factor(Attrition))) +
  geom_boxplot() +
  xlab("Department") +
  ylab("Monthly income") +
  scale_fill_discrete(guide=guide_legend(title="Attrition")) +
  theme_bw() +
  theme(text=element_text(size=13), legend.position="top")
```



3. Promotion is a commonly adopted HR strategy for employee retention. It can be observed in the following plot that for a certain department, e.g., Research & Development, employees with higher job level is more likely to leave if there are years since their last promotion.

```
ggplot(filter(df, as.character(Attrition) == "Yes"), aes(x=YearsSinceLastPromotion)) +
  geom_histogram(binwidth=0.5) +
  aes(y=..density..) +
  xlab("Years since last promotion.") +
  ylab("Density") +
  # scale_fill_discrete(guide=guide_legend(title="Attrition")) +
  facet_grid(Department ~ JobLevel)
```



Density graphs displaying the department wise promotion year wise. There are three departments, sales, Research and development, and Human resources.

4. Data pre-processing

Data preprocessing is the process of removing unwanted data from the dataset, making the data balance, removing the noise and outliers and handling the missing data.

```
# obtaining predictors that has no variation

pred_no_var <- names(df[, nearZeroVar(df)]) %T>% print()
```

```
## [1] "EmployeeCount" "Over18"        "StandardHours"
```

```
# removing the predictor columns with zero variation
df %<>% select(-one_of(pred_no_var))
```

Integer types of predictors which are nominal are converted to categorical type.

```
# converting integer variable to factor

int_2_ftr_vars <- c("Education", "EnvironmentSatisfaction", "JobInvolvement", "JobLevel", "JobSatisfacti

df[, int_2_ftr_vars] <- lapply((df[, int_2_ftr_vars]), as.factor)
```

The variables of character type are converted to categorical type.

```
df %<>% mutate_if(is.character, as.factor)
```

```
str(df)
```

```
## tibble [1,470 x 32] (S3: tbl_df/tbl/data.frame)
##  $ Age                     : num [1:1470] 41 49 37 33 27 32 59 30 38 36 ...
##  $ Attrition               : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 1 1 1 1 ...
##  $ BusinessTravel          : Factor w/ 3 levels "Non-Travel","Travel_Frequently",..: 3 2 3 2 3 2 3 3
##  $ DailyRate               : num [1:1470] 1102 279 1373 1392 591 ...
##  $ Department              : Factor w/ 3 levels "Human Resources",..: 3 2 2 2 2 2 2 2 2 2 ...
##  $ DistanceFromHome        : num [1:1470] 1 8 2 3 2 2 3 24 23 27 ...
##  $ Education               : Factor w/ 5 levels "1","2","3","4",..: 2 1 2 4 1 2 3 1 3 3 ...
##  $ EducationField          : Factor w/ 6 levels "Human Resources",..: 2 2 5 2 4 2 4 2 2 4 ...
##  $ EmployeeNumber          : num [1:1470] 1 2 4 5 7 8 10 11 12 13 ...
##  $ EnvironmentSatisfaction : Factor w/ 4 levels "1","2","3","4": 2 3 4 4 1 4 3 4 4 3 ...
##  $ Gender                  : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 1 2 2 2 ...
##  $ HourlyRate              : num [1:1470] 94 61 92 56 40 79 81 67 44 94 ...
##  $ JobInvolvement          : Factor w/ 4 levels "1","2","3","4": 3 2 2 3 3 3 4 3 2 3 ...
##  $ JobLevel                : Factor w/ 5 levels "1","2","3","4",..: 2 2 1 1 1 1 1 1 3 2 ...
##  $ JobRole                 : Factor w/ 9 levels "Healthcare Representative",..: 8 7 3 7 3 3 3 3 5 1
##  $ JobSatisfaction         : Factor w/ 4 levels "1","2","3","4": 4 2 3 3 2 4 1 3 3 3 ...
##  $ MaritalStatus           : Factor w/ 3 levels "Divorced","Married",..: 3 2 3 2 2 3 2 1 3 2 ...
##  $ MonthlyIncome           : num [1:1470] 5993 5130 2090 2909 3468 ...
##  $ MonthlyRate             : num [1:1470] 19479 24907 2396 23159 16632 ...
##  $ NumCompaniesWorked      : Factor w/ 10 levels "0","1","2","3",..: 9 2 7 2 10 1 5 2 1 7 ...
##  $ OverTime                : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 1 2 1 1 1 ...
##  $ PercentSalaryHike       : num [1:1470] 11 23 15 11 12 13 20 22 21 13 ...
##  $ PerformanceRating       : Factor w/ 2 levels "3","4": 1 2 1 1 1 1 2 2 2 1 ...
##  $ RelationshipSatisfaction: Factor w/ 4 levels "1","2","3","4": 1 4 2 3 4 3 1 2 2 2 ...
##  $ StockOptionLevel        : Factor w/ 4 levels "0","1","2","3": 1 2 1 1 2 1 4 2 1 3 ...
##  $ TotalWorkingYears       : num [1:1470] 8 10 7 8 6 8 12 1 10 17 ...
##  $ TrainingTimesLastYear   : num [1:1470] 0 3 3 3 3 2 3 2 2 3 ...
##  $ WorkLifeBalance         : num [1:1470] 1 3 3 3 3 2 2 3 3 2 ...
##  $ YearsAtCompany          : num [1:1470] 6 10 0 8 2 7 1 1 9 7 ...
##  $ YearsInCurrentRole      : num [1:1470] 4 7 0 7 2 7 0 0 7 7 ...
##  $ YearsSinceLastPromotion : num [1:1470] 0 1 0 3 2 3 0 0 1 7 ...
##  $ YearsWithCurrManager    : num [1:1470] 5 7 0 0 2 6 0 0 8 7 ...
```

5. Materials and Methods

5.1 Problem formalization

A model for predicting attrition can be built once the data is well-prepared. To predict whether or not an employee will depart is known as a binary classification problem, which is how employee attrition prediction is typically classed. Person status, labeled as Attrition in the data set, is used in this research case to predict outcomes. It has two levels, Yes and No, depending on whether the employee has departed or not.

Check the label column to make sure it is a factor type, as the model to be built is a classifier.

```
#Checks for categorical variable
is.factor(df$Attrition)
```

```
## [1] TRUE
```

5.2 Feature Selection

Given that not all variables are necessarily connected with the label, feature selection is done to eliminate those that are most important.Since the data set contains both discrete and numerical variables, some correlation analyses, such Pearson correlation, is not appropriate. One option is to train a model, rank the variables' importance, and then choose the most important ones.

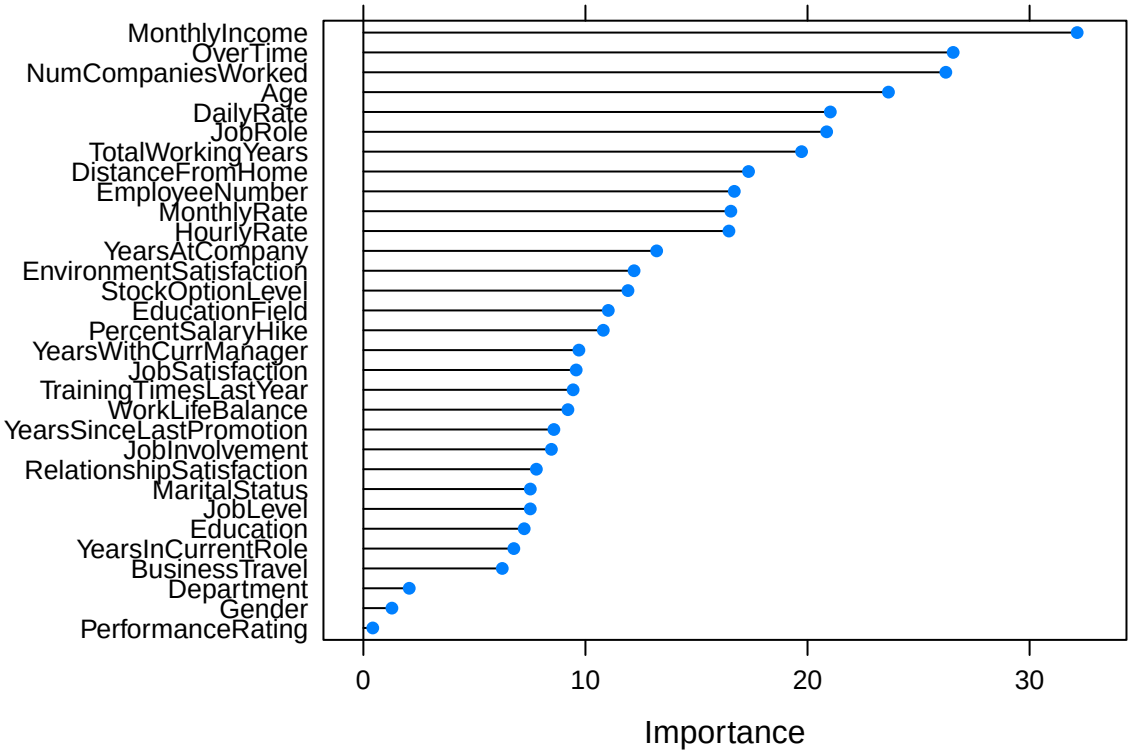
The following shows how to achieve variable importance ranking with a random forest model.

```
# set up the training control.

control <- trainControl(method="repeatedcv", number=3, repeats=1)

# train the model

model <- train(dplyr::select(df, -Attrition),
               df$Attrition,
               data=df,
               method="rf",
               preProcess="scale",
               trControl=control)
```

```
## Warning: Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
```

Now lets plot the importance graph from which we select our top features and drop the remaining from the dataset.

```
# estimate variable importance

imp <- varImp(model, scale=FALSE)

# plot

plot(imp)
```



Here below we dropped the last 3 low ranking variables and we worked on remaining 29 features and the the target variable "Attrition".

```
# select the top-ranking variables.

imp_list <- rownames(imp$importance)[order(imp$importance$Overall, decreasing=TRUE)]

# drop the low ranking variables. Here the last 3 variables are dropped.

top_var <-
  imp_list[1:(ncol(df) - 3)] %>%
  as.character()

top_var
```

```
##  [1] "MonthlyIncome"            "OverTime"
##  [3] "NumCompaniesWorked"       "Age"
##  [5] "DailyRate"                "JobRole"
##  [7] "TotalWorkingYears"        "DistanceFromHome"
##  [9] "EmployeeNumber"           "MonthlyRate"
## [11] "HourlyRate"               "YearsAtCompany"
## [13] "EnvironmentSatisfaction"  "StockOptionLevel"
## [15] "EducationField"           "PercentSalaryHike"
## [17] "YearsWithCurrManager"     "JobSatisfaction"
## [19] "TrainingTimesLastYear"    "WorkLifeBalance"
## [21] "YearsSinceLastPromotion"  "JobInvolvement"
## [23] "RelationshipSatisfaction" "MaritalStatus"
## [25] "JobLevel"                 "Education"
## [27] "YearsInCurrentRole"       "BusinessTravel"
## [29] "Department"
```

```
# select the top ranking variables

df %<>% select(., one_of(c(top_var, "Attrition")))
```

5.3 Splitting the data into train and test sets:

A prediction model can be then created for predictive analysis. The whole data is split into training and testing sets. The former is used for model creation while the latter for verification. We have split the data in 70:30 ratio.

Train data set consists of 70% of the dataset.

Test data set contains 30% of the dataset.

```
train_index <-
  createDataPartition(df$Attrition,
                      times=1,
                      p=.7) %>%
  unlist()

df_train <- df[train_index, ]
df_test <- df[-train_index, ]
```

One thing worthnoting is that the training set is not balanced, which may deteriorate the performance in training a model.

```
table(df_train$Attrition)
```

```
##
##  No Yes
## 864 166
```

There are more active employees (864) than terminated employees (166). The problem of data imbalance can be resolved in a number of ways:

Re-sampling the data, either to increase the minority class's representation or to decrease the majority class's.

Use cost sensitive learning method.

In the context of Imbalanced classes, the ROSE package contains routines to address binary classification issues. A binary classifier's estimation and accuracy evaluation phases can be aided in the presence of an uncommon class by using artificial balanced samples created using a smoothed bootstrap technique. There are further functions that implement more conventional solutions to the class imbalance as well as several metrics for accuracy evaluation. These are calculated using cross-validation, bootstrap, or holdout techniques.

```
# note DMwR::SMOTE does not handle well with tbl_df. Need to convert to data frame.
library("smotefamily")
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
df_train %<>% as.data.frame()
#ROSE(admit~., data = train, N = 500, seed=111)$data
df_train <- ROSE(Attrition ~ .,
                 data=df_train,
                  N=1030,
                  seed=111)$data
```

```
table(df_train$Attrition)
```

```
##
##  No Yes
## 507 523
```

5.4 Models

After balancing the training set, a model can be created for prediction. For comparison purpose, different individual models, as well as ensemble of them, are trained on the data set.

1. **Individual models.**

   1.1 Support vector machine with radial basis function kernel

   1.2 Random forest

   1.3 Boosted Logistic Regression

2. **Stack of Models**

5.4.1.1 Support vector machine with radial basis function kernel

Support vector machines are a famous and a very strong classification technique which does not use any sort of probabilistic model like any other classifier but simply generates hyperplanes or simply putting lines, to separate and classify the data in some feature space into different regions.

```
# initialize training control.

tc <- trainControl(method="boot",
                   number=3,
                   repeats=3,
                   search="grid",
                   classProbs=TRUE,
                   savePredictions="final",
                   summaryFunction=twoClassSummary)
```

```
## Warning: `repeats` has no meaning for this resampling method.
```

```
# SVM model.

time_svm <- system.time(
  model_svm <- train(Attrition ~ .,
                     df_train,
                     method="svmRadial",
                     trainControl=tc)
)
```

5.4.1.2 Random forest

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction committee is more accurate than that of any individual tree

```
# random forest model

time_rf <- system.time(
  model_rf <- train(Attrition ~ .,
                     df_train,
                     method="rf",
```

```
                      trainControl=tc)
)
```

5.4.2.2 Boosted Logistic Regression

Boosting the logistic regression model is a way to convert a set of weak learners to a strong model. The weak learners specialize on different subsets of data. The subsequent models will do the classification task on the misclassified data. The final model can be a weighted sum of your weak models. With boosting, you can get better results since it can reduce bias as well as variance.

```
#Boosted Logistic Regression
ctrl <- trainControl(method = "repeatedcv",
                         number = 4,
                         savePredictions = TRUE,
                         verboseIter = T,

                                          returnResamp = "all")
time_logit <- system.time(
  model_logit <- train(Attrition ~ .,
                       df_train,
                       method="LogitBoost",
                       family="binomial",
                       trainControl=ctrl))
```

5.4.2 Stack of Models

Model Stacking is a way to improve model predictions by combining the outputs of multiple models that we modeled above and running them through as another machine learning model called a meta-learner. Thismodel has given the highest accuracy amongst the other three models. This is a kind of ensembling technique. We used the ROC metric for evaluating the models in the stack of models meta model.

```
time_ensemble <- system.time(
  model_list <- caretList(Attrition ~ .,
                          data=df_train,
                          trControl=tc,
                          methodList=c("svmRadial", "rf","LogitBoost"
                                       ))
)
```

```
## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
# stack of models. Use glm for meta model.

model_stack <- caretStack(model_list,metric="ROC",method="glm",
  trControl=trainControl(method="boot",number=10, savePredictions="final",classProbs=TRUE,summaryFunctio
```

5.5 Model Validations

The trained models are applied on testing data sets for model evaluation. Confusion matrices are shown below for each model.

```
models <- list(model_svm, model_rf, model_stack,model_logit)

predictions <-lapply(models, predict, newdata=select(df_test, -Attrition))
```

```
#confusion matrix
cm_metrics <- lapply(predictions,confusionMatrix, reference=df_test$Attrition, positive="Yes")
cm_metrics
```

```
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  310  25
##        Yes  59  46
##
##                Accuracy : 0.8091
##                  95% CI : (0.7692, 0.8448)
##     No Information Rate : 0.8386
##     P-Value [Acc > NIR] : 0.9574879
##
##                   Kappa : 0.4089
##
##  Mcnemar's Test P-Value : 0.0003175
##
##             Sensitivity : 0.6479
##             Specificity : 0.8401
##          Pos Pred Value : 0.4381
##          Neg Pred Value : 0.9254
##              Prevalence : 0.1614
##          Detection Rate : 0.1045
##    Detection Prevalence : 0.2386
##       Balanced Accuracy : 0.7440
##
##        'Positive' Class : Yes
##
##
## [[2]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  309  27
##        Yes  60  44
##
##                Accuracy : 0.8023
##                  95% CI : (0.7619, 0.8385)
##     No Information Rate : 0.8386
##     P-Value [Acc > NIR] : 0.9817612
##
```

```
##                  Kappa : 0.3849
##
##  Mcnemar's Test P-Value : 0.0006019
##
##             Sensitivity : 0.6197
##             Specificity : 0.8374
##          Pos Pred Value : 0.4231
##          Neg Pred Value : 0.9196
##              Prevalence : 0.1614
##          Detection Rate : 0.1000
##    Detection Prevalence : 0.2364
##       Balanced Accuracy : 0.7286
##
##        'Positive' Class : Yes
##
##
## [[3]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  315  31
##        Yes  54  40
##
##                Accuracy : 0.8068
##                  95% CI : (0.7668, 0.8427)
##     No Information Rate : 0.8386
##     P-Value [Acc > NIR] : 0.96753
##
##                  Kappa : 0.3688
##
##  Mcnemar's Test P-Value : 0.01702
##
##             Sensitivity : 0.56338
##             Specificity : 0.85366
##          Pos Pred Value : 0.42553
##          Neg Pred Value : 0.91040
##              Prevalence : 0.16136
##          Detection Rate : 0.09091
##    Detection Prevalence : 0.21364
##       Balanced Accuracy : 0.70852
##
##        'Positive' Class : Yes
##
##
## [[4]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  266  21
##        Yes 103  50
##
##                Accuracy : 0.7182
```

```
##                  95% CI : (0.6736, 0.7598)
##     No Information Rate : 0.8386
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2899
##
##  Mcnemar's Test P-Value : 3.49e-13
##
##             Sensitivity : 0.7042
##             Specificity : 0.7209
##          Pos Pred Value : 0.3268
##          Neg Pred Value : 0.9268
##              Prevalence : 0.1614
##          Detection Rate : 0.1136
##    Detection Prevalence : 0.3477
##       Balanced Accuracy : 0.7125
##
##        'Positive' Class : Yes
##
```

6. Results

Here we have calculated accuracy, recall and precision for all our models and mapped them into a data frame for easier understanding at a glance.

```
# accuracy
acc_metrics <- lapply(cm_metrics, `[[`, "overall") %>%lapply(`[`, 1) %>% unlist()
# recall
rec_metrics <- lapply(cm_metrics, `[[`, "byClass") %>%lapply(`[`, 1) %>%unlist()
# precision
pre_metrics <- lapply(cm_metrics, `[[`, "byClass") %>%lapply(`[`, 3) %>%unlist()
algo_list <- c("SVM RBF", "Random Forest", "Stacking"," Boosted Logistic Regression")
time_consumption <- c(time_svm[3], time_rf[3], time_ensemble[3],time_logit[3])

df_comp <- data.frame(Models=algo_list,  Accuracy=acc_metrics, Recall=rec_metrics,Precision=pre_metrics
```

```
##                        Models  Accuracy    Recall Precision    Time
## 1                     SVM RBF 0.8090909 0.6478873 0.4380952  13.372
## 2               Random Forest 0.8022727 0.6197183 0.4230769 147.857
## 3                    Stacking 0.8068182 0.5633803 0.4255319  23.850
## 4  Boosted Logistic Regression 0.7181818 0.7042254 0.3267974   4.494
```

7. Conclusion

Employee attrition is one of the major issues faced by firms of small, large and medium scale. Losing valuable employees because of the lack of insights into their satisfaction and factors which contribute to the resignation/retirement can be a big loss to the firms. Through our project, we tried to address this issue by leveraging data and statistical models. We used advanced statistical models like SVM, Random Forest, Boosted Logistic Regression and Stacking to try to predict with higher accuracy , which employees are more likely to leave the firm. We also observed that all models performed well, but Stacking gave the best results on our data. Due to lack of volume of data, we ran into a number of problems which we tried to address through resampling. Future prospects of this project would include acquiring larger volumes of data so that we can achieve higher accuracy and precision in our predictions.

8. Author contributions:

Prasanna Krishna Reddy Jeedipally: Data Cleaning, preprocessing and visualization

Bharath Reddy Madi: Implemented Boosted logistic regression, random forest and stack of models

Divya Sharvani Kandukuri: Implemented SVM with radial kernel and performed model validations.

GIT HUB REPOSITORY

We have pushed our project into Github repository. Repository Link: https://github.com/bharathrmadi/SDM1.git