Final project Group 7

Bharath Reddy Madi

2022-12-04

1. Data-driven analytics for HR attrition prediction

```
# data wrangling

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)
```

```
## Warning: package 'readr' was built under R version 4.2.2
```

```
# machine learning and advanced analytics

library(DMwR2)
```

```
## Warning: package 'DMwR2' was built under R version 4.2.2

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

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.2

## Loading required package: lattice
```

```r
library(caretEnsemble)
```

```
## Warning: package 'caretEnsemble' was built under R version 4.2.2

##
## Attaching package: 'caretEnsemble'

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

```r
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.2.2

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

##
## Attaching package: 'pROC'

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

```r
# natural language processing

#library(msLanguageR)
#library(tm)
#library(jiebaR)

# tools

library(httr)
```

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

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

```r
#library(XML)
#library(jsonlite)

# data visualization

library(scales)
```

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

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

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

```
## Warning: package 'wordcloud' was built under R version 4.2.2

## Loading required package: RColorBrewer
```

```r
DATA1 <- "/Employee_attrition.csv"
```

```r
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
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>, ...
```

```
dim(df)
```

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

```
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)
```

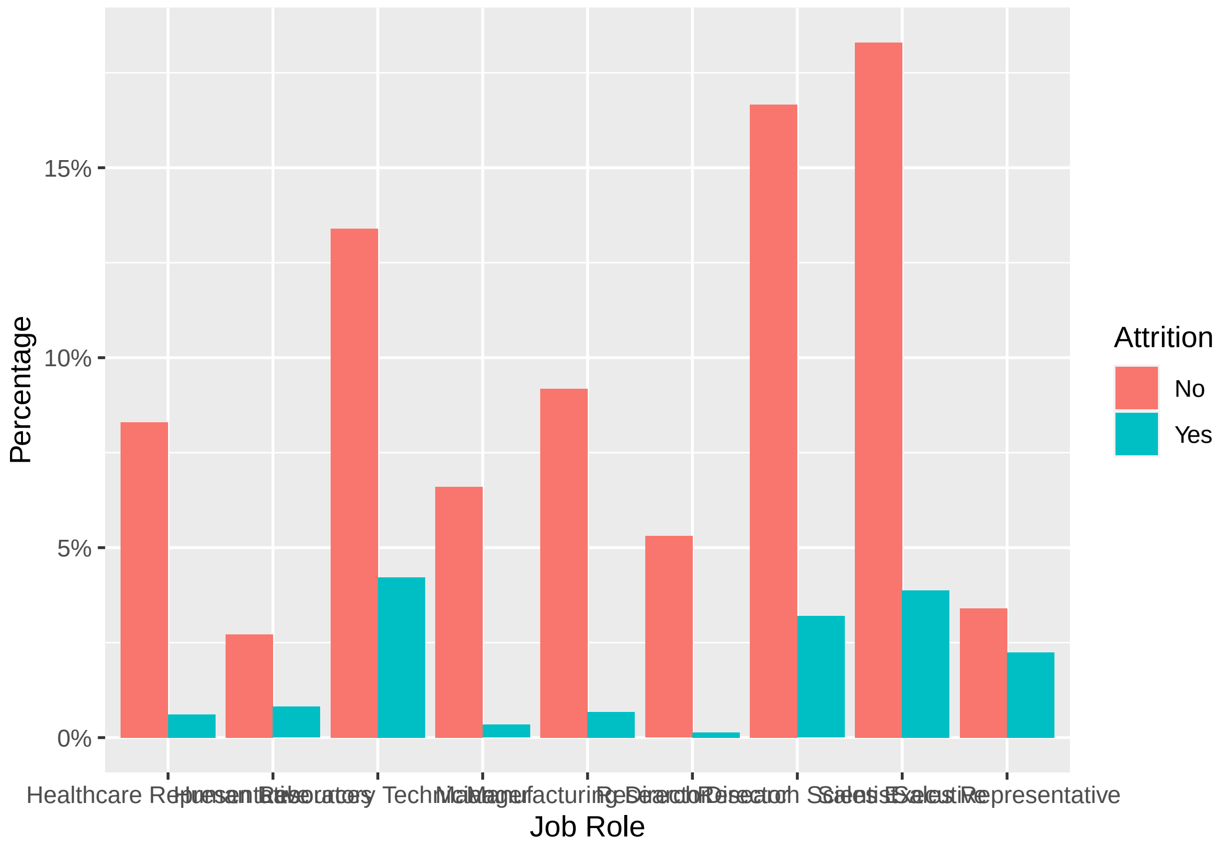
```
## spc_tbl_ [1,470 x 35] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Age                     : num [1:1470] 41 49 37 33 27 32 59 30 38 36 ...
##  $ Attrition               : chr [1:1470] "Yes" "No" "Yes" "No" ...
##  $ BusinessTravel          : chr [1:1470] "Travel_Rarely" "Travel_Frequently" "Travel_Rarely" "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 Technic
##  $ 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>
```

2 Visualization of 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.

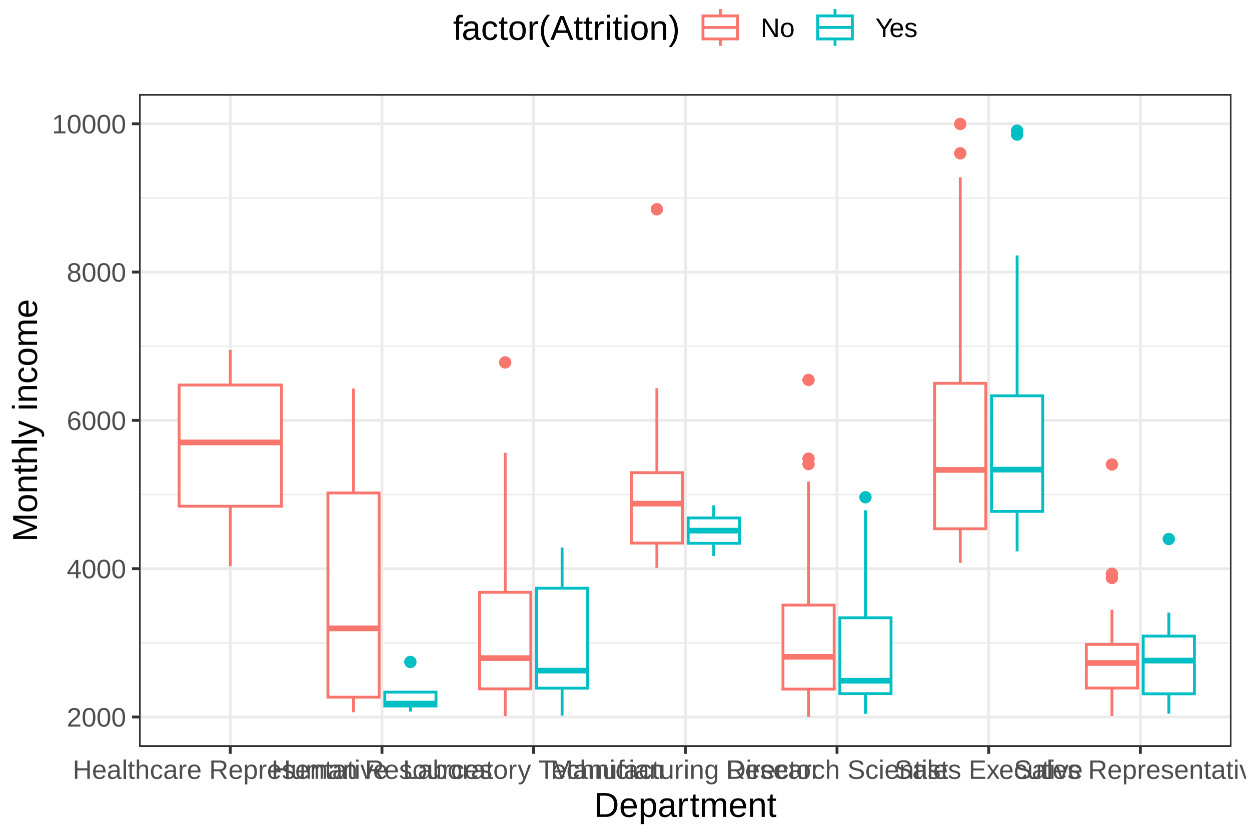
```r
ggplot(df, aes(JobRole, fill=Attrition)) +
  geom_bar(aes(y=(..count..)/sum(..count..)), position="dodge") +
  scale_y_continuous(labels=percent) +
  xlab("Job Role") +
  ylab("Percentage")
```

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



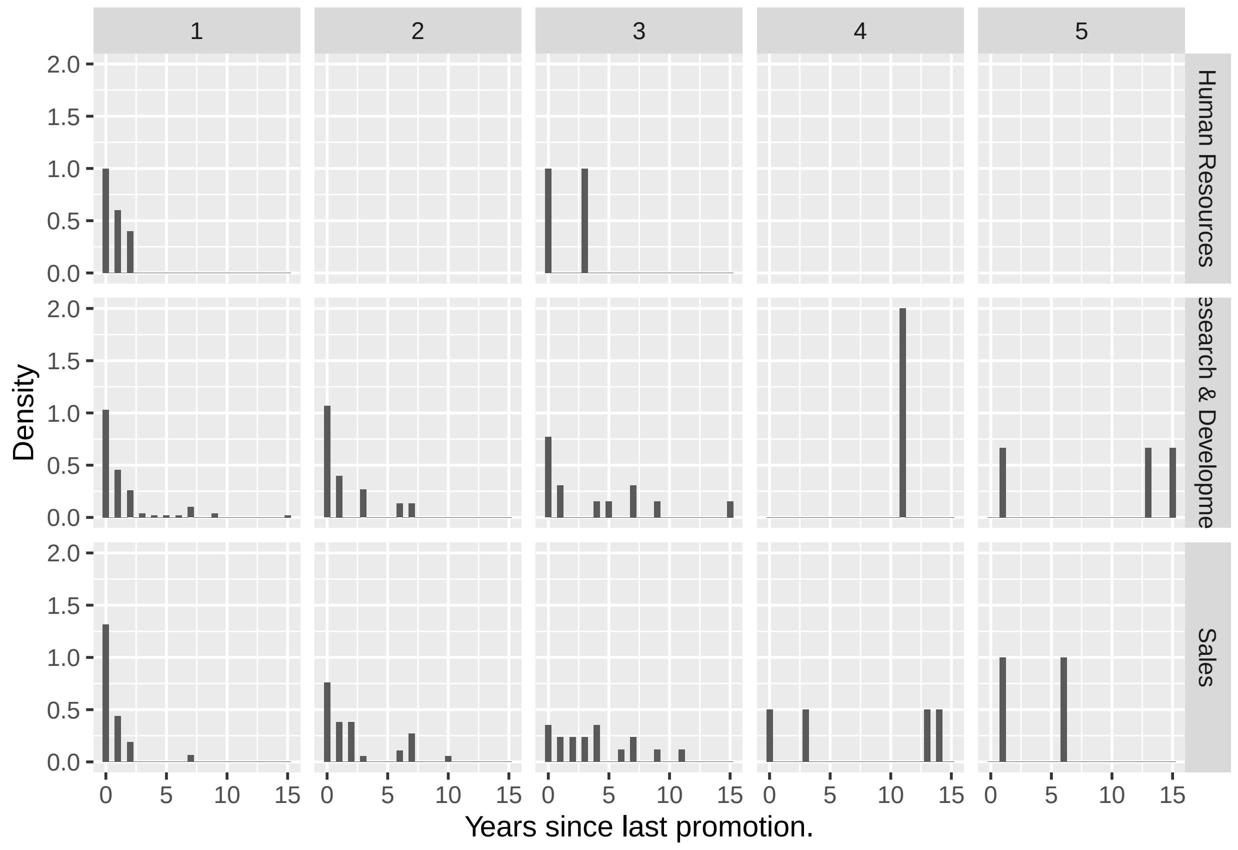
2. monthly income, job level, and service year may affect decision of leaving for employees in different departments. For example, junior staffs with lower pay will be more likely to leave compared to those who are paid higher.

```r
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)
```



```r
# get predictors that has no variation.

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

Data pre-processing

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

```r
# remove the zero variation predictor columns.

df %<>% select(-one_of(pred_no_var))
```

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

```r
# convert certain integer variable to factor variable.

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

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 ...
```

Problem formalization After the data is well prepared, a model can be constructed for attrition prediction. Normally employee attrition prediction is categorized as a binary classification problem, i.e., to predict *whether or not an employee will leave.*

In this study case, the label for prediction is employee status, named as `Attrition` in the data set, which has two levels, `Yes` and `No`, indicating that the employee has left or stayed.

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

```
is.factor(df$Attrition)
```

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

It is possible that not all variables are correlated with the label, feature selection is therefore performed to filter out the most relevant ones.

As the data set is a blend of both numerical and discrete variables, certain correlation analysis (e.g., Pearson correlation) is not applicable. One alternative is to train a model and then rank the variable importance so as to select the most salient 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)
```

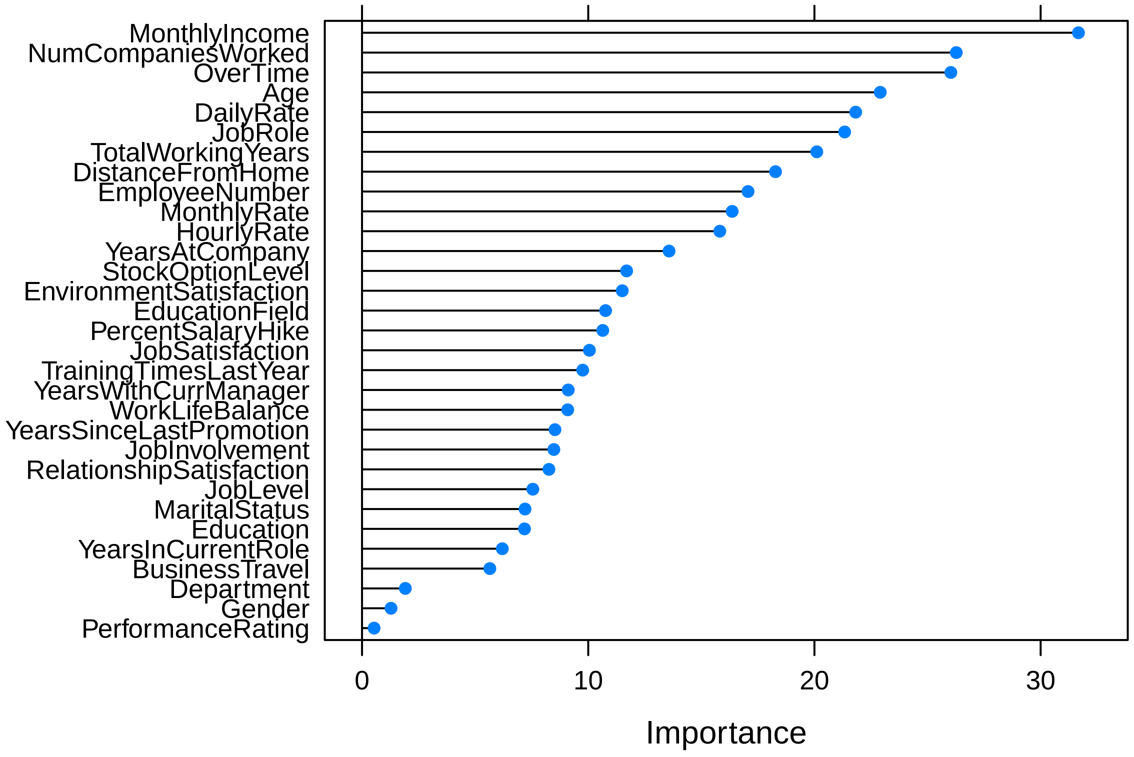
```
## Warning: Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.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

```
# estimate variable importance

imp <- varImp(model, scale=FALSE)

# plot

plot(imp)
```



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

```
# select the top ranking variables

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

2.1.6 Resampling 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.

```
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
```

Active employees (864) are more than terminated employees (166). There are several ways to deal with data imbalance issue:

1. Resampling the data - either upsampling the minority class or downsampling the majority class.
2. Use cost sensitive learning method.

In this case the first method is used. SMOTE is a commonly adopted method for synthetically upsampling minority class in an imbalanced data set. Package `DMwR` provides methods that apply SMOTE methods on training data set.

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

```
## Warning: package 'smotefamily' was built under R version 4.2.2
```

```
library(ROSE)
```

```
## Warning: package 'ROSE' was built under R version 4.2.2
```

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

```r
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
```

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

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

Model building 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. `caret` and `caretEnsemble` packages are used for training models.

1. Individual models.

Three algorithms, support vector machine with radial basis function kernel, random forest, and extreme gradient boosting (xgboost), are used for model building.

```r
# 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.
```

```r
# SVM model.

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

# random forest model

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

```r
#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))
```

```r
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.
```

```r
# 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,
    summaryFunction=twoClassSummary
  )
)
```

2.1.8 Model validation The trained models are applied on testing data sets for model evaluation.

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

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

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

```
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  305  24
##        Yes  64  47
##
##                Accuracy : 0.8
##                  95% CI : (0.7595, 0.8364)
##     No Information Rate : 0.8386
##     P-Value [Acc > NIR] : 0.9866
##
##                   Kappa : 0.398
##
##  Mcnemar's Test P-Value : 3.219e-05
##
##             Sensitivity : 0.6620
##             Specificity : 0.8266
##          Pos Pred Value : 0.4234
##          Neg Pred Value : 0.9271
##              Prevalence : 0.1614
##          Detection Rate : 0.1068
##    Detection Prevalence : 0.2523
##       Balanced Accuracy : 0.7443
##
##        'Positive' Class : Yes
##
##
## [[2]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  309  26
##        Yes  60  45
##
##                Accuracy : 0.8045
##                  95% CI : (0.7644, 0.8406)
##     No Information Rate : 0.8386
##     P-Value [Acc > NIR] : 0.975514
##
##                   Kappa : 0.3949
##
##  Mcnemar's Test P-Value : 0.000373
##
##             Sensitivity : 0.6338
##             Specificity : 0.8374
##          Pos Pred Value : 0.4286
##          Neg Pred Value : 0.9224
##              Prevalence : 0.1614
##          Detection Rate : 0.1023
##    Detection Prevalence : 0.2386
##       Balanced Accuracy : 0.7356
##
##        'Positive' Class : Yes
##
##
## [[3]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  324  30
##        Yes  45  41
##
##                Accuracy : 0.8295
##                  95% CI : (0.7911, 0.8635)
##     No Information Rate : 0.8386
##     P-Value [Acc > NIR] : 0.7233
##
##                   Kappa : 0.4197
##
##  Mcnemar's Test P-Value : 0.1060
##
##             Sensitivity : 0.57746
##             Specificity : 0.87805
##          Pos Pred Value : 0.47674
##          Neg Pred Value : 0.91525
##              Prevalence : 0.16136
##          Detection Rate : 0.09318
##    Detection Prevalence : 0.19545
##       Balanced Accuracy : 0.72776
##
##        'Positive' Class : Yes
##
##
## [[4]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  231  17
##        Yes 138  54
##
##                Accuracy : 0.6477
##                  95% CI : (0.6011, 0.6924)
##     No Information Rate : 0.8386
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.229
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7606
##             Specificity : 0.6260
##          Pos Pred Value : 0.2812
##          Neg Pred Value : 0.9315
##              Prevalence : 0.1614
##          Detection Rate : 0.1227
##    Detection Prevalence : 0.4364
##       Balanced Accuracy : 0.6933
##
##        'Positive' Class : Yes
##
```

```
# 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,
             Time=time_consumption) %T>%
             {head(.) %>% print()}
```

```
##                         Models  Accuracy    Recall Precision   Time
## 1                      SVM RBF 0.8000000 0.6619718 0.4234234  28.20
## 2                Random Forest 0.8045455 0.6338028 0.4285714 143.73
## 3                     Stacking 0.8295455 0.5774648 0.4767442  21.56
## 4  Boosted Logistic Regression 0.6477273 0.7605634 0.2812500   3.63
```