Signature Assignment: Predicting whether an employee will leave or stay

Janice Tjeng

2017-12-09

Business Understanding

Problem

High employee turnover can have a severe impact on businesses in terms of productivity, time, and money. When an employee leaves, productivity decreases and companies have to find a replacement, which costs time and money due to advertising, interviewing, training, and hiring a new employee. It takes more resources for a company to return to the same level of productivity it had before (Business Fillings 2017).

Analyses

Analytics can be applied to understand why employees leave and predict whether an employee will leave the company. Understanding why employees leave will help organizations come up with specific strategies to retain employees. Predicting which employees will leave allows the company to either focus on these employees and try to retain them, or invest less efforts on these employees since they will be leaving. Therefore the analyses has two goals to solve the problem of a high turnover rate:

1. Understand why employees leave
2. Predict which employees will leave

Data Understanding

The data set comes from kaggle, does not contain any missing values, and is simulated. The outcome is a categorical variable indicating whether an employee will leave or stay (left: 1 or 0 respectively). Other variables include:

- Satisfaction level (numeric, ranges from 0-1)
- Time since last performance evaluation (numeric, ranges from 0-1, measured in fraction of years)
- Number of projects completed (numeric)
- Average monthly hours (numeric)
- Number of years in the company (numeric)
- Work place accident (categorical, 0/1)
- Promotion last 5 years (categorical, 0/1)
- Department worked for (categorical, multiple levels)

- Salary level (categorical, low/medium/high)

Data Acquisition

```
library(tidyverse)
hr <- read_csv("../data/employee_prediction.csv")
```

Data Exploration

```
str(hr)

## Classes 'tbl_df', 'tbl' and '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                : chr  "sales" "sales" "sales" "sales" ...
##  $ salary               : chr  "low" "medium" "medium" "low" ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 10
##   .. ..$ satisfaction_level   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ last_evaluation      : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ number_project       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ average_montly_hours : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ time_spend_company   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Work_accident        : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ left                 : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ promotion_last_5years: list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ sales                : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ salary               : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"

summary(hr) # No missing values in data
```

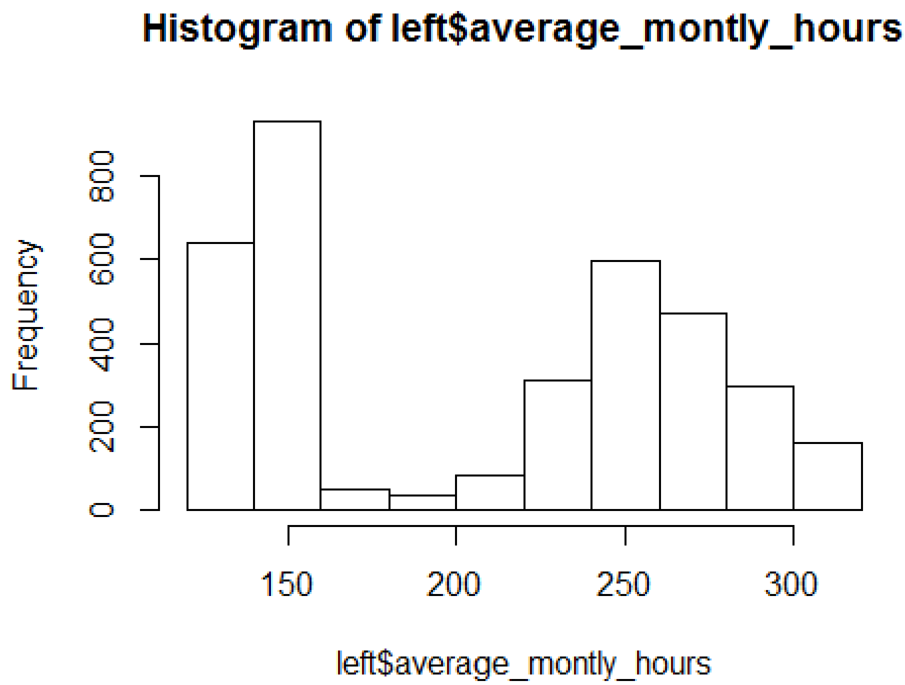
```
##  satisfaction_level last_evaluation  number_project  average_montly_hours
##  Min.   :0.0900     Min.   :0.3600   Min.   :2.000   Min.   : 96.0
##  1st Qu.:0.4400     1st Qu.:0.5600   1st Qu.:3.000   1st Qu.:156.0
##  Median :0.6400     Median :0.7200   Median :4.000   Median :200.0
##  Mean   :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       Length:14999       Length:14999
##  1st Qu.:0.00000       Class :character   Class :character
##  Median :0.00000       Mode  :character   Mode  :character
##  Mean   :0.02127
##  3rd Qu.:0.00000
##  Max.   :1.00000
```

Based on the summary statistics, values of different variables seem to make sense as there are no negative values for numerical variables. I will examine the distribution in greater detail in the section below.

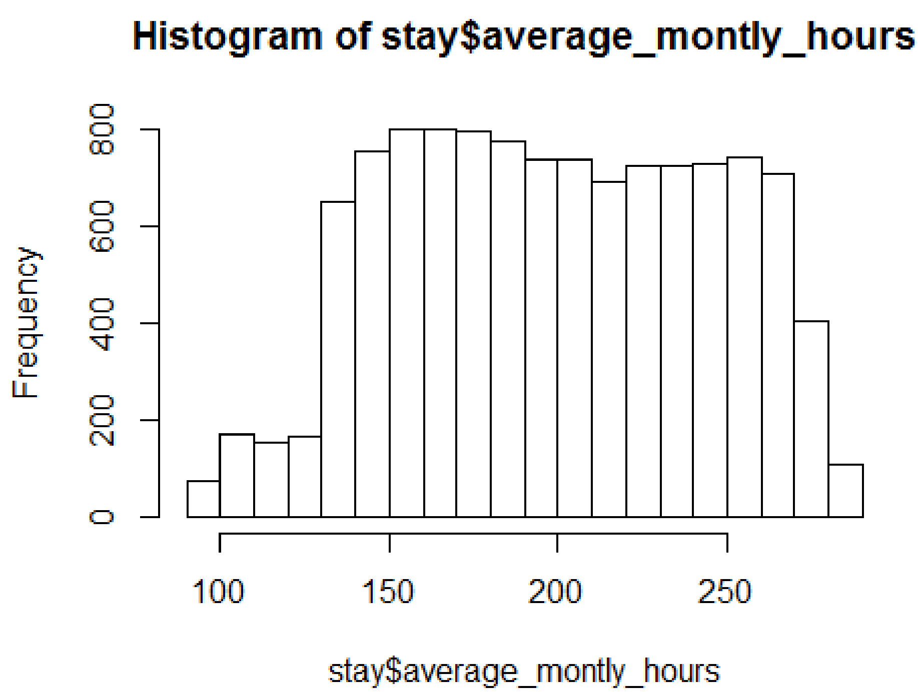
Exploratory data plots

The amount of time employees work, their satisfaction levels, and salaries, affect the company's turnover rate (Business Fillings 2017). For exploration, I am going to look at the distribution of hours and satisfaction levels for those who left vs those who stayed to understand how the distributions differ. I will also look at the proportion of people in each salary category and compare it between those who left and those who stayed.

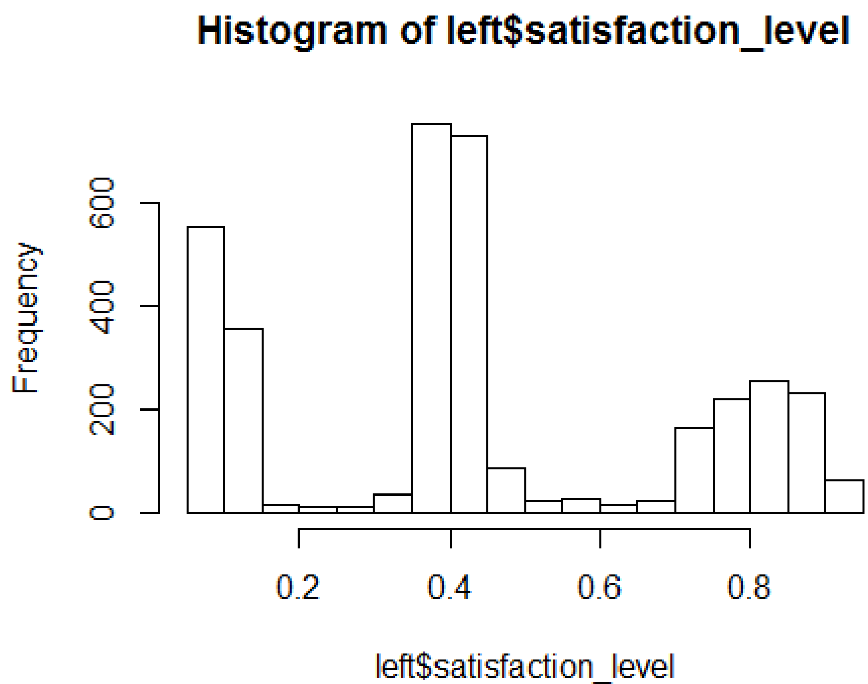
```r
left <- hr %>%
  filter(left==1)
stay <- hr %>%
  filter(left==0)

# Average monthly hours for people who left vs those who stayed
hist(left$average_montly_hours)
```



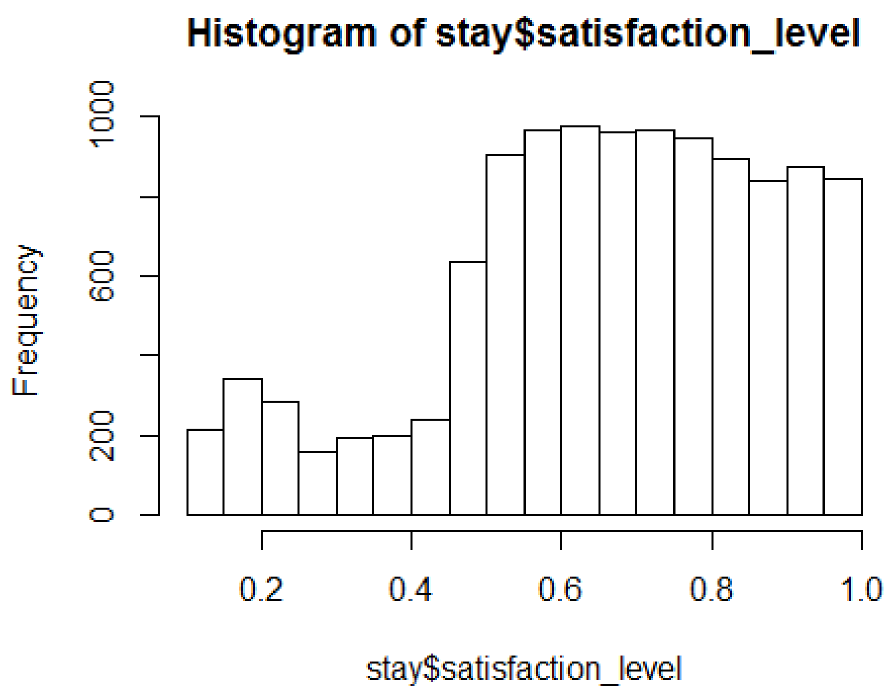

```r
hist(stay$average_montly_hours)
```




```r
# Satisfaction level for people who left vs people who stayed
hist(left$satisfaction_level)
```

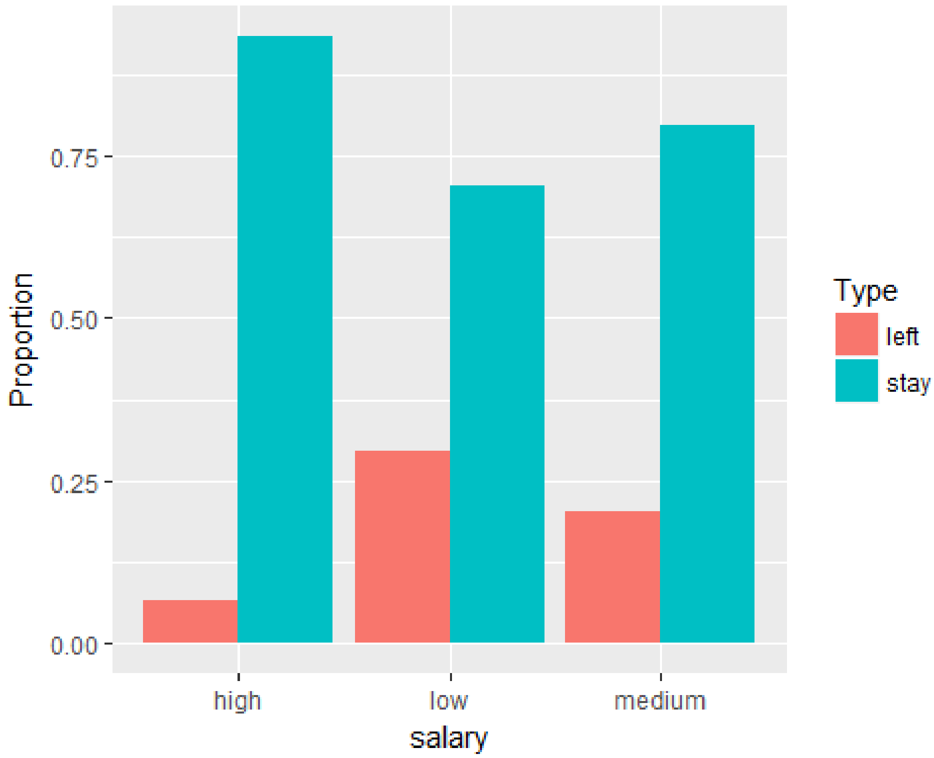



```r
hist(stay$satisfaction_level)
```




```r
# Salary of people who leave vs those who stayed
hr %>%
  group_by(salary) %>%
  summarize(left=mean(left),
            stay=1-left) %>%
  gather(`left`,`stay`,key="Type",value="Proportion") %>%
```

```r
  ggplot(aes(x=salary,y=Proportion)) +
  geom_col(aes(fill=Type), position="dodge")
```



People who stay mostly work between 130-270 hours while there is a greater variation in the number of hours worked for people who left. Similarly, there is a greater variation in satisfaction level for people who left, compared to that for people who stayed, and most people who stayed have satisfaction level above 0.5.

From the salary distribution, here is a higher proportion of people who stay than those who left. Most people who left have low and medium salary.

Outlier detection

Outliers are detected based on z-score standardization for numeric features.

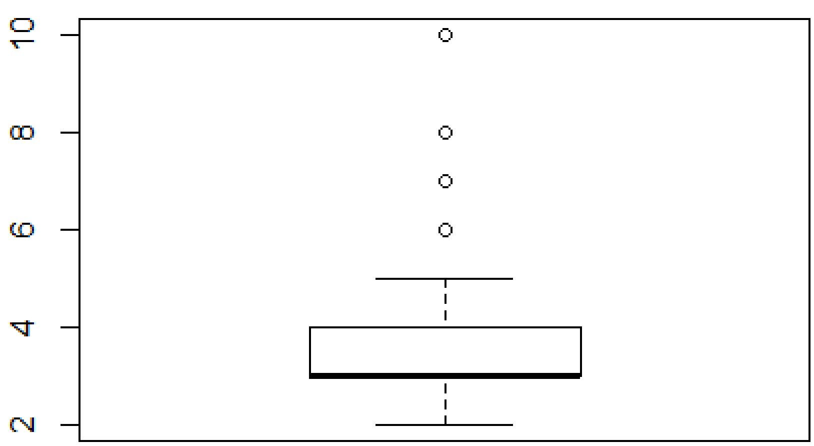
```r
# z-score standardization
standardize <- function(x){
  return ((x-mean(x))/sd(x))
}

hr.standardized <- hr %>%
  mutate_at(vars(satisfaction_level, last_evaluation, number_project, average
_monthly_hours, time_spend_company), standardize)

summary(hr.standardized)

##  satisfaction_level last_evaluation    number_project
##  Min.   :-2.1029    Min.   :-2.08041   Min.   :-1.4628
```

```
##   1st Qu.:-0.6951    1st Qu.:-0.91197   1st Qu.:-0.6515  
##   Median : 0.1093    Median : 0.02277   Median : 0.1598  
##   Mean   : 0.0000    Mean   : 0.00000   Mean   : 0.0000  
##   3rd Qu.: 0.8332    3rd Qu.: 0.89910   3rd Qu.: 0.9711  
##   Max.   : 1.5572    Max.   : 1.65858   Max.   : 2.5937  
##   average_montly_hours time_spend_company Work_accident         left       
##   Min.   :-2.10340     Min.   :-1.0261    Min.   :0.0000   Min.   :0.0000  
##   1st Qu.:-0.90203     1st Qu.:-0.3412    1st Qu.:0.0000   1st Qu.:0.0000  
##   Median :-0.02103     Median :-0.3412    Median :0.0000   Median :0.0000  
##   Mean   : 0.00000     Mean   : 0.0000    Mean   :0.1446   Mean   :0.2381  
##   3rd Qu.: 0.87999     3rd Qu.: 0.3436    3rd Qu.:0.0000   3rd Qu.:0.0000  
##   Max.   : 2.18148     Max.   : 4.4528    Max.   :1.0000   Max.   :1.0000  
##   promotion_last_5years    sales              salary         
##   Min.   :0.00000       Length:14999       Length:14999      
##   1st Qu.:0.00000       Class :character   Class :character  
##   Median :0.00000       Mode  :character   Mode  :character  
##   Mean   :0.02127                                            
##   3rd Qu.:0.00000                                            
##   Max.   :1.00000
```

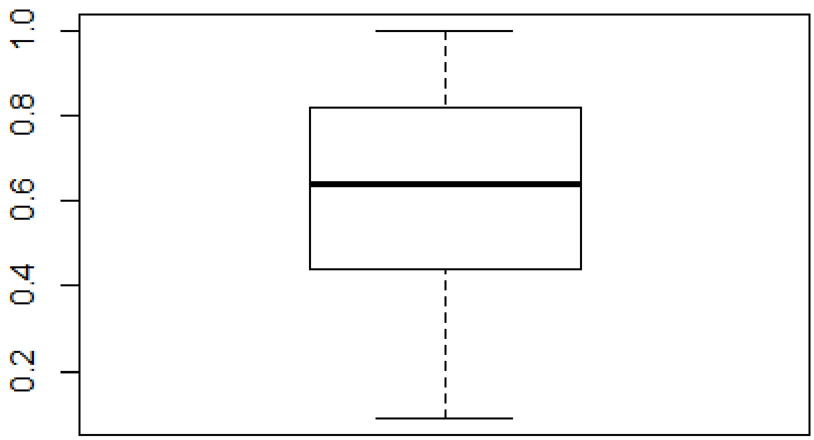
With a maximum z-score of 4, it looks like time spend in the company contain outliers.

Let's look at boxplots to visualize outliers

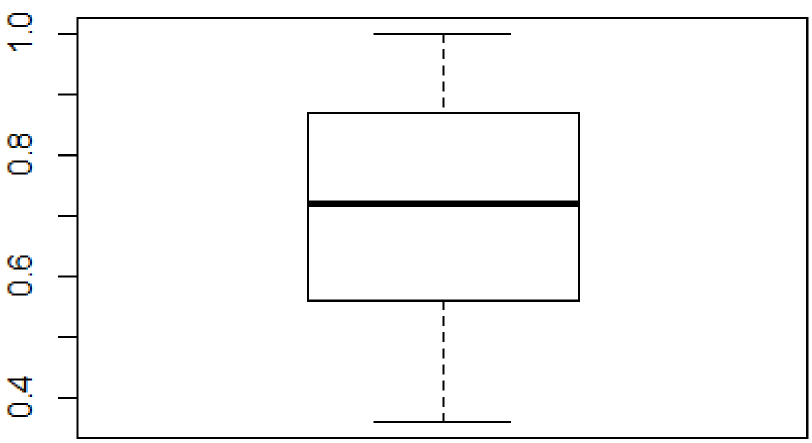
```
boxplot(hr$time_spend_company) # Outliers with values above 5. The distributi
on looks right skewed because the median is towards the lower quartile (more
observations with lower values).
```



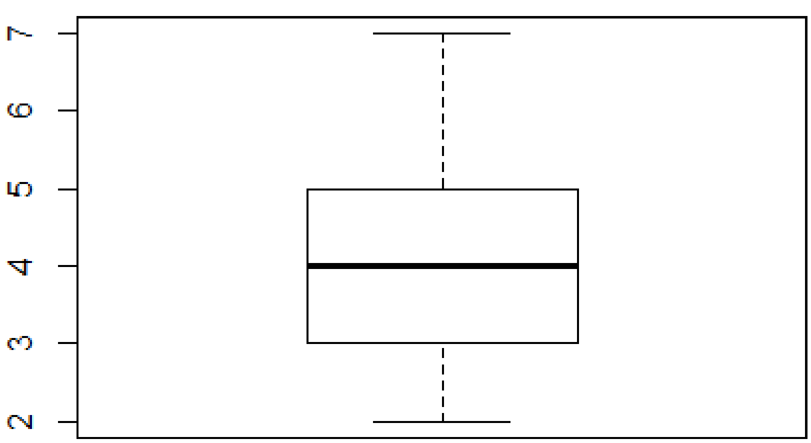

```
# Boxplox did not show outliers for other numeric features. Abolsute z-scores
for these features are also within 3 standard deviations from the mean.
boxplot(hr$satisfaction_level)
```



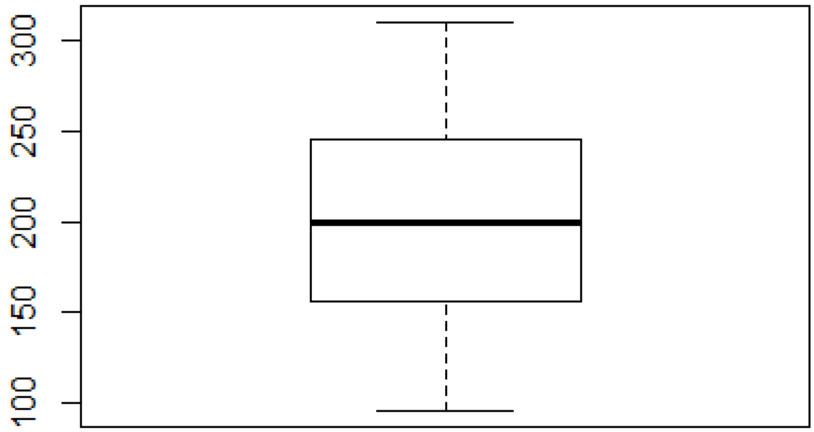

```
boxplot(hr$last_evaluation)
```




```
boxplot(hr$number_project)
```



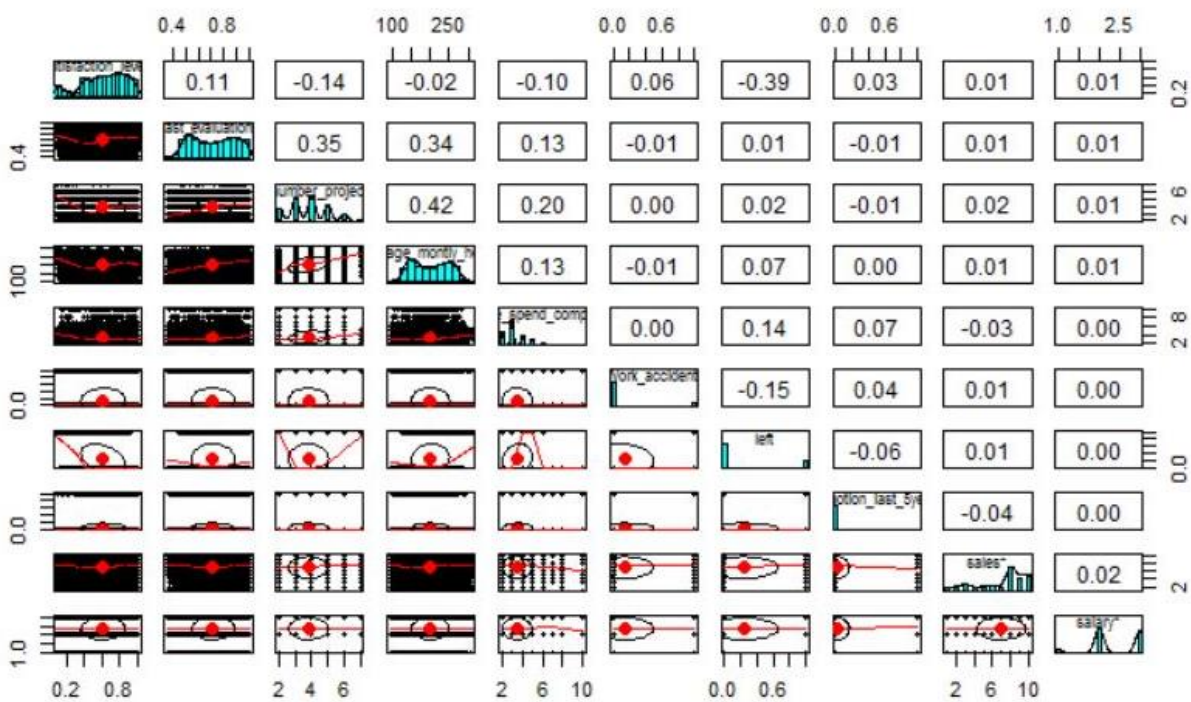

```
boxplot(hr$average_montly_hours)
```



Correlation/collinearity analysis

pairs.panels() will be used for collinearity analyses as well as assess distributional skew.

```
library(psych)
pairs.panels(hr)
```



There are few significant collinearities (correlations > 0.3), and multiple weak ones, but none of them are very strong (correlations < 0.5). Satisfaction level is negatively correlated (-0.39) with left. People who are satisfied with their job are less likely to leave. Last evaluation, number of projects, and average monthly hours are positively correlated with one another. As number of projects increase, so does average monthly hours and time since last evaluation, and vice-versa.

Regression assumes that features are normally distributed. For numeric features, time spend in the company is right skewed and has to be transformed (transformation shown in data shaping). Other numeric features are roughly normally distributed. For nominal features, it does not make much sense to transform to a normal distribution since the counts are based on categories.

Data Preparation

Data Cleaning

Dealing with outliers

Strategies dealing with outliers involve capping and prediction (Prabhakaran, 2017). Capping replaces outliers that are lower or above the 1.5*inter-quartile range (IQR) with the 5th and 95th percentile respectively. Prediction treats outliers as missing values, which are imputed via the mean, median, or mode, or machine learning techniques that consider the column with missing values as the response variable.

Since the project involves imputing missing values, I will treat outliers as missing values and impute them via a machine learning technique. I will also compare this approach with capping by comparing model performances after applying them on each of the data set (data set with capping technique applied vs data set with missing values imputed).

Capping

```
out <- hr$time_spend_company
qnt <- quantile(out, probs=c(0.25,0.75)) # values of 1st and 3rd quarter
caps <- quantile(out, probs=c(0.05,0.95)) # values of 5th and 95th percentile
H <- 1.5*IQR(out)
out[out>qnt[2]+H] <- caps[2]

hr.capping <- hr %>%
  mutate(time_spend_company=ifelse(time_spend_company>(qnt[2]+H),caps[2], tim
e_spend_company)) %>% # only consider upper limit since boxplot shows outlier
s in the upper limit
  mutate_at(vars(Work_accident,promotion_last_5years,sales, salary, left), as
.factor)
```

Imputing missing values

```
# Since features are a mix of categorical and numeric types, and the missing
values to impute are of numeric type, I will do a regression tree to impute t
hose values
library(rpart)
hr.impute <- hr %>%
  mutate(time_spend_company=ifelse(time_spend_company>(qnt[2]+H),NA, time_spe
nd_company)) %>% # Replace outliers with NA
  mutate_at(vars(Work_accident,promotion_last_5years,sales, salary, left), as
.factor)
model_tree <- rpart(time_spend_company~., data=hr.impute)
predict_tree <- predict(model_tree,hr.impute[-5])
```

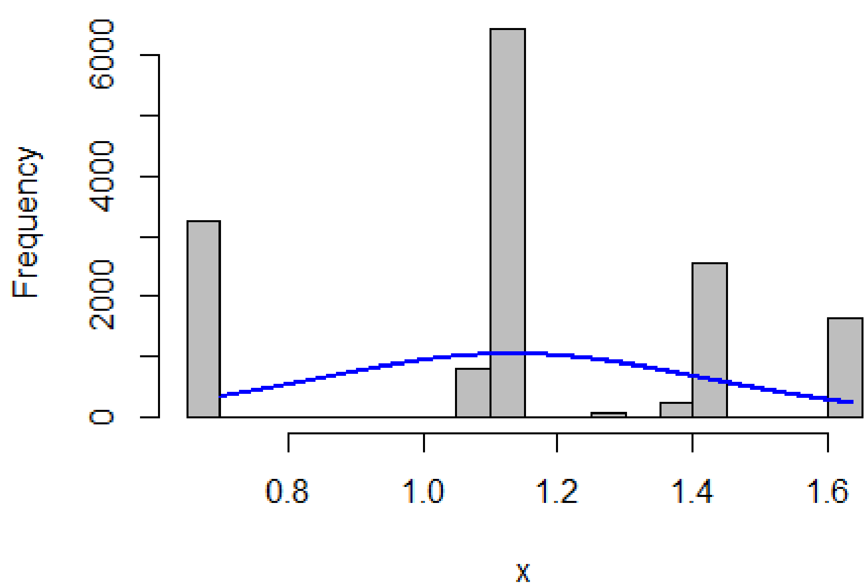
```
hr.impute1 <- hr.impute %>%
  mutate(time_spend_company=ifelse(is.na(time_spend_company),predict_tree, ti
me_spend_company))
```

Data Shaping

Dealing with distributional skew

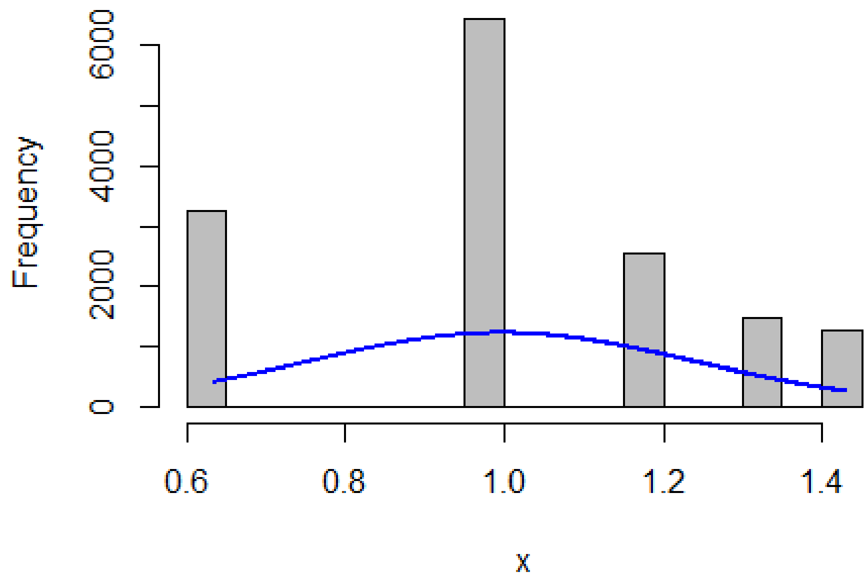
Box-cox transformation applied, which generates the lambda value required to transform the variable to a normal distribution. Such is required for regression models which assume that features are normally distributed.

```
library(MASS)
library(rcompanion)
# Imputed data
box <- boxcox(hr.impute1$time_spend_company~1)

cox <- data.frame(box$x, box$y)
cox2 <- cox[with(cox, order(-cox$box.y)),]
lambda <- cox2[1,"box.x"]
trans.hr <- (hr.impute1$time_spend_company ^ lambda - 1)/lambda
plotNormalHistogram(trans.hr)
```




```
hr.impute2 <- hr.impute1 %>%
  mutate(time_spend_company=trans.hr)
# Capped data
box1 <- boxcox(hr.capping$time_spend_company~1)
```

```
cox1 <- data.frame(box1$x, box1$y)
cox3 <- cox1[with(cox1, order(-cox1$box1.y)),]
lambda1 <- cox3[1,"box1.x"]
trans.hr1 <- (hr.capping$time_spend_company ^ lambda1 - 1)/lambda1
plotNormalHistogram(trans.hr1)
```




```
hr.capping1 <- hr.capping %>%
  mutate(time_spend_company=trans.hr1)
```

Normalization

I did standardization for outlier detection in the data exploration stage. Since standardization for scaling the data to a small interval works best for normally distributed features and not all features are exactly normally distributed, I will apply normalization instead. Such is necessary for models that involve distance measures such as kNN, neural networks, and SVM.

```
normalize <- function(x){
  return((x-min(x))/(max(x)-min(x)))
}

hr.imp.norm <- hr.impute2 %>%
  mutate_at(vars(satisfaction_level, last_evaluation, number_project, average
_montly_hours, time_spend_company), normalize)

hr.cap.norm <- hr.capping1 %>%
```

```
  mutate_at(vars(satisfaction_level, last_evaluation, number_project, average
_montly_hours, time_spend_company), normalize)
```

Feature Engineering

Dummy codes

Dummy coding converts categorical into numeric features, which is necessary for models that work best with numeric features such as kNN, regression, and SVM. Categorical variables n levels will be dummy coded to have n-1 columns. For a variable with 2 levels, additional transformation is not necessary (just need to convert categorical variable into numeric) because there is only one variable (2-1). For a variable with at least 3 levels, additional tranformation for each column will be done via the following function:

```
dummy <- function(data, column){
  binom <- data.frame(y=runif(nrow(data)), x=runif(nrow(data)), col=column)
  dummy <- as.data.frame(model.matrix(y~x+col, binom))
  return (dummy %>%
            dplyr::select (-c(1,2)))
}

df_sales <- dummy(hr.imp.norm, hr.imp.norm$sales)
df_salary <- dummy(hr.cap.norm, hr.cap.norm$salary)
hr.imp.n.d <- hr.imp.norm %>%
  dplyr::select(-c(sales,salary)) %>%
  cbind(df_sales,df_salary) %>%
  mutate(Work_accident=as.numeric(levels(Work_accident)[Work_accident]),
promotion_last_5years=as.numeric(levels(promotion_last_5years)[promotion_last
_5years]))
hr.cap.n.d <- hr.cap.norm %>%
  dplyr::select(-c(sales,salary)) %>%
  cbind(df_sales,df_salary) %>%
  mutate(Work_accident=as.numeric(levels(Work_accident)[Work_accident]),
promotion_last_5years=as.numeric(levels(promotion_last_5years)[promotion_last
_5years]))
```

New derived features

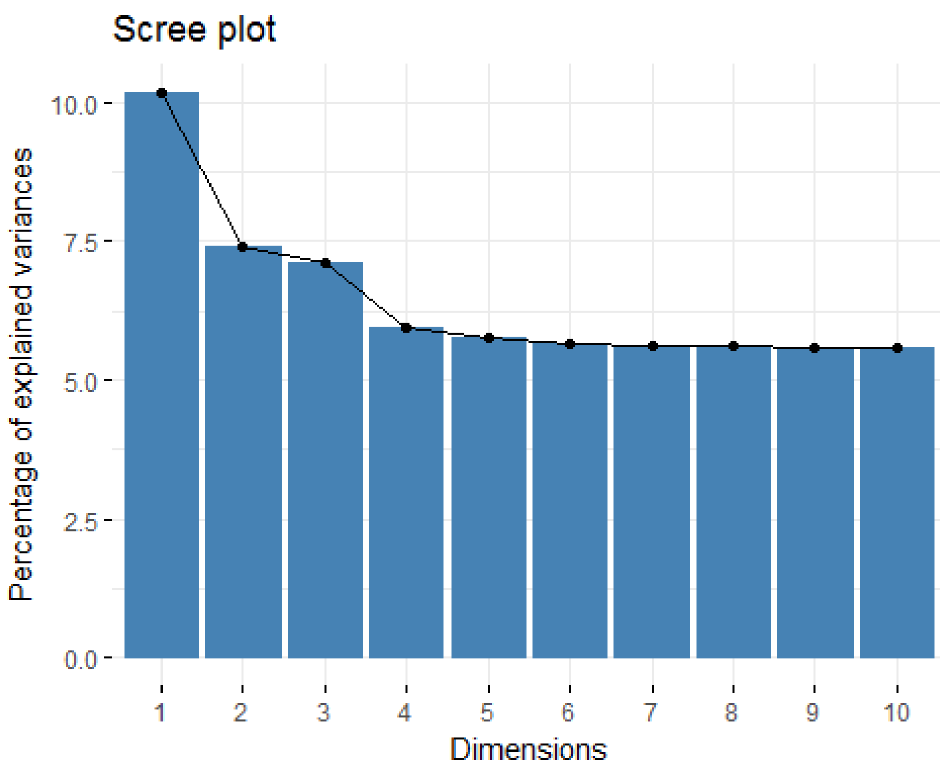
There are no newly derived features necessary in this data set.

PCA

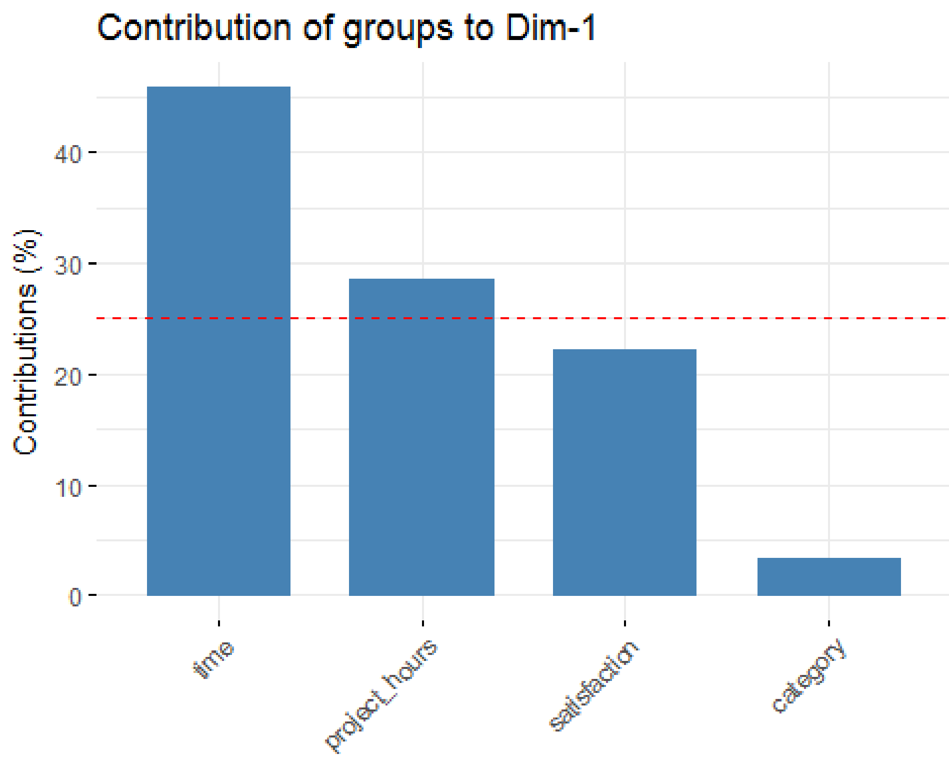
A multi-factor analysis (MFA) that takes into account categorical and numeric features would be applied to determine which features explain most of the variation in the data set. Although a regular principal component analysis can be done after transforming categorical features into numeric (through dummy coding), it is difficult to interpret which features are important when there are dummy coded features that are not actual distinct features.

MFA

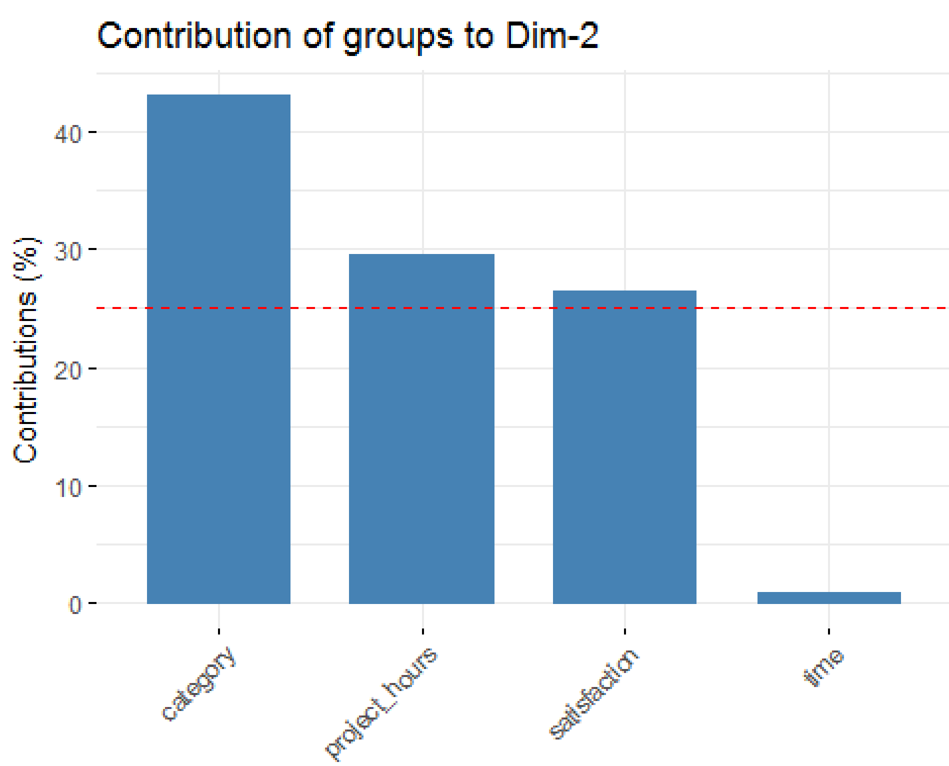
```r
library(FactoMineR)
library(factoextra)
hr.impute3 <- hr.impute2 %>%
  dplyr::select(1:6,9,8,10) # reorder columns
hr.mfa <- MFA(hr.impute3,
              group=c(1,3,1,4), # group columns together
              type=c("s","s","s","n"),
              name.group = c("satisfaction","project_hours","time","category"
),
              graph=F)
fviz_screeplot(hr.mfa) # First 3 components explain most variance in data
```



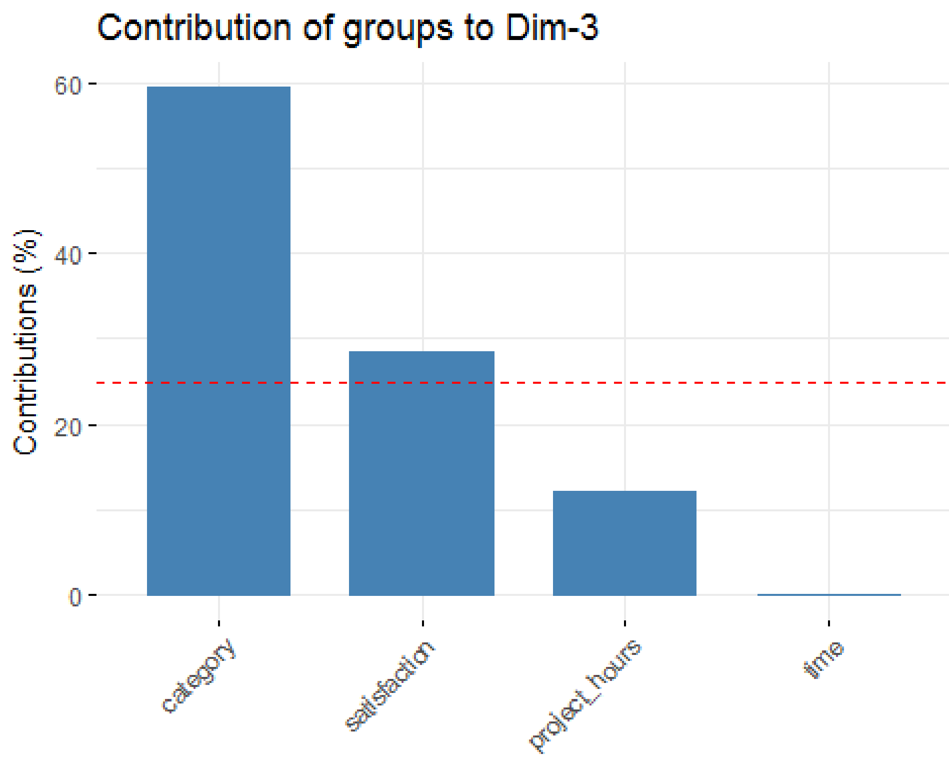

```r
# Contribution in each component.dimension
fviz_contrib(hr.mfa, "group", axes = 1)
```




```r
fviz_contrib(hr.mfa, "group", axes = 2)
```




```r
fviz_contrib(hr.mfa, "group", axes = 3)
```




```
group <- get_mfa_var(hr.mfa,"group")
group$correlation
```

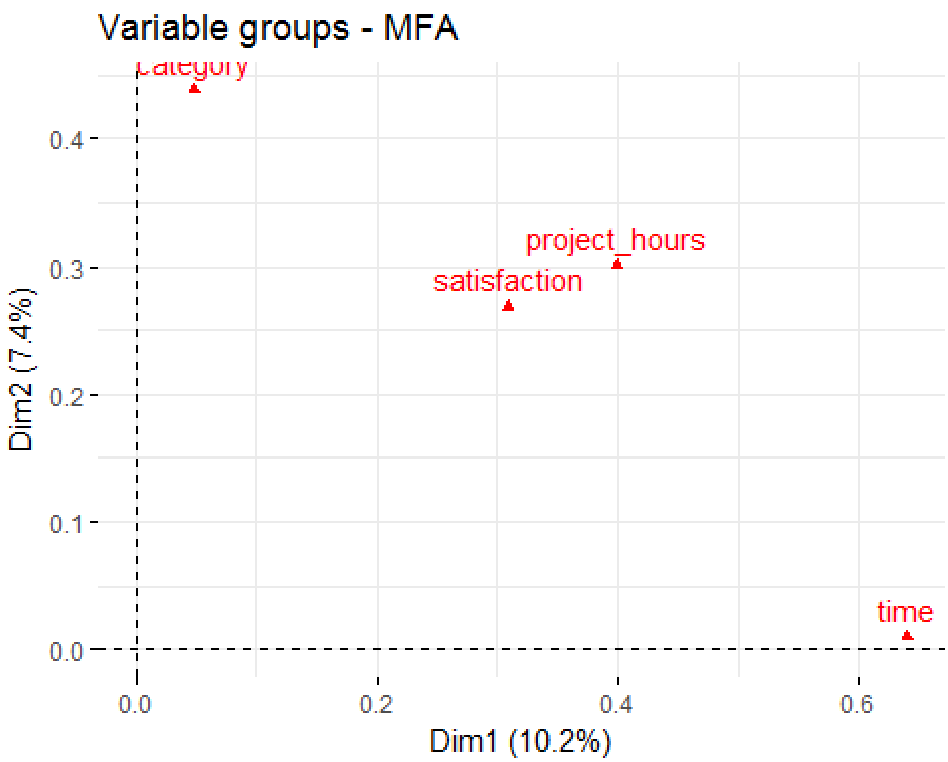
```
##                    Dim.1      Dim.2        Dim.3      Dim.4      Dim.5
## satisfaction  0.5558045 0.51805540 0.5272296868 0.04060731 0.07991676
## project_hours 0.6376133 0.55350512 0.3578415466 0.04068992 0.04614246
## time          0.8000531 0.09976582 0.0003856959 0.06176481 0.01298789
## category      0.2301783 0.66455225 0.7658578875 0.99490189 0.99345626
```

```
group$contrib
```

```
##                    Dim.1      Dim.2        Dim.3      Dim.4       Dim.5
## satisfaction  22.136192 26.4036303 2.847722e+01  0.2018486  0.80965700
## project_hours 28.596080 29.5673868 1.212328e+01  0.1879086  0.16134833
## time          45.866586  0.9792075 1.524010e-05  0.4669812  0.02138466
## category       3.401142 43.0497754 5.939949e+01 99.1432615 99.00761001
```

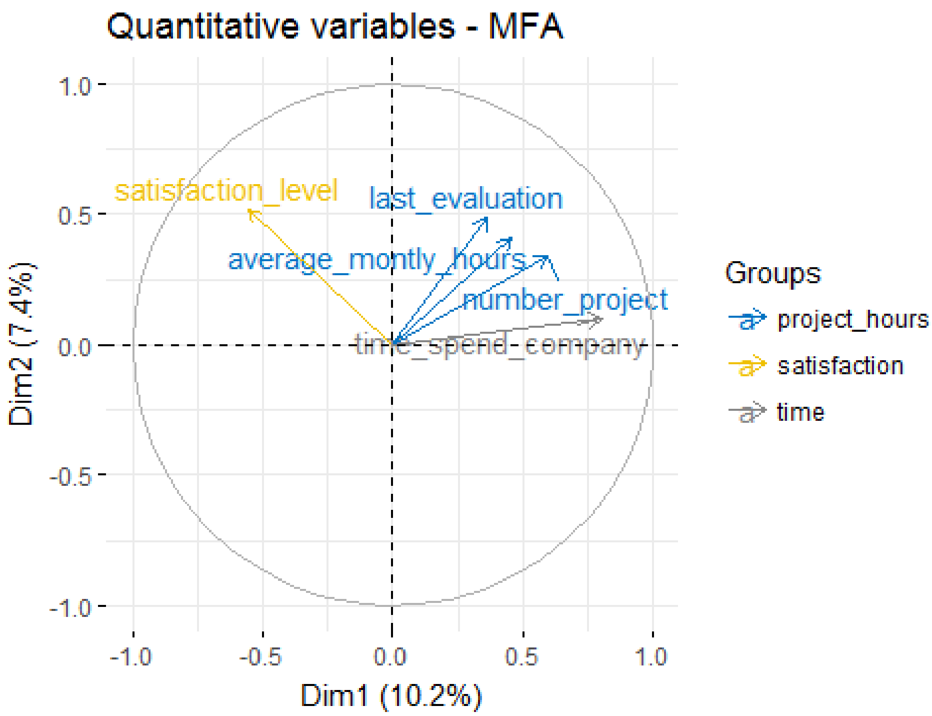
According to the screeplot, the first 3 components explain most of the variation in the data set. Time contributes most to the first dimension, and has the highest correlation in the first dimension. Category group (includes all categorical features except left (whether an employee left)) contributes most to the second through fifth dimensions and has the highest correlations in these dimensions.

```
# Plot groups of variables
fviz_mfa_var(hr.mfa, "group")
```



Time spend in the company contributes most to the first dimension while categorical features contribute most to the second dimension. Satisfaction and project hours have roughly equal contributions in both dimensions. Therefore, it appears that all features explain variation in the data set and should be kept.

```
# Visualize correlation of quantitative variables
fviz_mfa_var(hr.mfa, "quanti.var", palette = "jco",
             col.var.sup = "violet", repel = TRUE)
```



Variables that point in the same direction are positively correlated with one another while those that point in the opposite direction are negatively correlated with one another. According to the pairs.panels() plot shown in the data exploration stage, time spend in the company, last evaluation, average monthly hours, and number of projects are positively correlated with one another because they point in the same direction in one quadrant as shown in the plot above. On the other hand, satisfaction level is negatively correlated with other variables because it points in a different direction in another quadrant. However, the correlations are not very strong because the direction of satisfaction level is not completely opposite that of time spend company, and variables that positively correlate with each other do not completely align with one another. The correlations from pairs.panels also show that correlations are not very strong (<0.5).

Modeling & Evaluation

Creation of training and validation subsets

Training and testing sets would be split into 80% and 20% respectively. Random sampling will be applied to make sure that the training and validation data set each contains an even split of employees who left and employees who stayed.

```
set.seed(577)
n <- nrow(hr.imp.n.d)
size <- n*0.8
train_sample <- sample(n, size)
# Imputed data set
```

```
hr_train <- hr.imp.n.d[train_sample,]
hr_test <- hr.imp.n.d[-train_sample,]
prop.table(table(hr_train$left))

##
##         0         1
## 0.7614801 0.2385199

prop.table(table(hr_test$left))

##
##         0         1
## 0.7636667 0.2363333

# Capped data set
hr_train_cap <- hr.cap.n.d[train_sample,]
hr_test_cap <- hr.cap.n.d[-train_sample,]
prop.table(table(hr_train_cap$left))

##
##         0         1
## 0.7614801 0.2385199

prop.table(table(hr_test_cap$left))

##
##         0         1
## 0.7636667 0.2363333
```

There is a fairly even split of employees in each category in the training and testing data sets.

Model construction & evaluation

Since I am predicting a binary outcome, here are the models that work:

- kNN
- Naive Bayes
- Decision Tree
- Logistic Regression
- SVM
- Neural network
- Random forest

Since the variables in the data set are a combination of categorical and numeric types, decision tree and its ensemble, random forest, are the only models among those listed above, that do not require any data processing as they accept numerical and categorical features. Neural network also accepts both types of features but numerical features need to be scaled to a small interval. I wanted to explore how model performance would differ among those that require extensive data transformation in comparison to those that do not require extensive data transformation. I also wanted to see the difference in model performance across black box and white box models. Therefore, I decided to apply the following models:

- SVM (black box model, requires dummy coding as well as normalization)
- Logistic regression (straightforward commonly used model, determine whether most important features are the same as those determined by MFA- investigate feature selection techniques. Also used to understand why employees leave)
- Decision Tree (white box model, does not require data processing)
- Random Forest (ensemble of decision trees, but more of a black box model)

After training models on the training data set, I will apply the trained models on the testing data set , and compare predicted against actual values. Model performance will be evaluated and compared with one another via percentage accuracy, kappa statistic, false negative rate, and AUC.

1. SVM

```r
set.seed(577)
library(caret)
library(kernlab)
# Imputed data set
m_svm <- ksvm(left~., data=hr_train, kernel="rbfdot")
p_svm <- predict(m_svm, hr_test, type="response")
confusionMatrix(p_svm, hr_test$left, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2223   65
##          1   68  644
##
##                Accuracy : 0.9557
##                  95% CI : (0.9477, 0.9628)
##     No Information Rate : 0.7637
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8774
##  Mcnemar's Test P-Value : 0.8623
##
##             Sensitivity : 0.9083
##             Specificity : 0.9703
##          Pos Pred Value : 0.9045
##          Neg Pred Value : 0.9716
##              Prevalence : 0.2363
##          Detection Rate : 0.2147
##    Detection Prevalence : 0.2373
##       Balanced Accuracy : 0.9393
##
```

```
##        'Positive' Class : 1
##

cat("% of false positives (FP) is \n",round((68/3000)*100,2), "% of false neg
atives (FN) is \n",round((65/3000)*100,2))

## % of false positives (FP) is
##  2.27, % of false negatives (FN) is
##  2.17

# Capped data set
m_svm_cap <- ksvm(left~., data=hr_train_cap, kernel="rbfdot")
p_svm_cap <- predict(m_svm_cap, hr_test_cap, type="response")
confusionMatrix(p_svm_cap, hr_test_cap$left, positive = "1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2226   76
##          1   65  633
##
##                Accuracy : 0.953
##                  95% CI : (0.9448, 0.9603)
##     No Information Rate : 0.7637
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8691
##  Mcnemar's Test P-Value : 0.3997
##
##             Sensitivity : 0.8928
##             Specificity : 0.9716
##          Pos Pred Value : 0.9069
##          Neg Pred Value : 0.9670
##              Prevalence : 0.2363
##          Detection Rate : 0.2110
##    Detection Prevalence : 0.2327
##       Balanced Accuracy : 0.9322
##
##        'Positive' Class : 1
##

cat("% of false positives (FP) is \n",round((65/3000)*100,2), "% of false neg
atives (FN) is \n",round((76/3000)*100,2))

## % of false positives (FP) is
##  2.17, % of false negatives (FN) is
##  2.53
```

Evaluating model performance

There is a slight difference in model performance between the imputed and capped data set. The imputed data set performs slightly better in terms of higher accuracy, kappa value, and lower % of FN.

The SVM model for the imputed data set has a performance accuracy of 95.6% and a kappa statistic of 0.88 (very good agreement betweem model's prediction and true values), as well as a low FN rate of 2.17%.

2. Logistic regression

```
# Imputed data
train <- hr.imp.norm[train_sample,] # Use data set without dummy coding for e
asier handling of variable names as regression will automatically create dumm
y variables
test <- hr.imp.norm[-train_sample,]
m_lr <- glm(left~., data=train, family="binomial")
summary(m_lr) # Sales marketing has the highest p-value

##
## Call:
## glm(formula = left ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7948  -0.5550  -0.2667  -0.0554   3.4723
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -2.606205   0.199365 -13.073  < 2e-16 ***
## satisfaction_level     -3.589727   0.105859 -33.911  < 2e-16 ***
## last_evaluation         0.061139   0.118817   0.515 0.606857
## number_project         -2.604699   0.136217 -19.122  < 2e-16 ***
## average_montly_hours    0.840783   0.136864   6.143 8.09e-10 ***
## time_spend_company      4.283175   0.126014  33.990  < 2e-16 ***
## Work_accident1         -1.585639   0.106274 -14.920  < 2e-16 ***
## promotion_last_5years1 -0.981067   0.283308  -3.463 0.000534 ***
## saleshr                 0.127839   0.158232   0.808 0.419137
## salesIT                -0.202095   0.145520  -1.389 0.164901
## salesmanagement        -0.201849   0.190736  -1.058 0.289935
## salesmarketing          0.003272   0.156819   0.021 0.983354
## salesproduct_mng       -0.157394   0.154514  -1.019 0.308374
## salesRandD             -0.646737   0.170718  -3.788 0.000152 ***
## salessales             -0.025431   0.121863  -0.209 0.834691
## salessupport            0.086249   0.129512   0.666 0.505440
## salestechnical          0.035637   0.127229   0.280 0.779399
## salarylow               1.894318   0.150193  12.613  < 2e-16 ***
## salarymedium            1.388924   0.151498   9.168  < 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: 13183.7  on 11998  degrees of freedom
## Residual deviance:  8970.6  on 11980  degrees of freedom
## AIC: 9008.6
##
## Number of Fisher Scoring iterations: 6
```

```
# Remove sales variable and conduct a likelihood ratio test to determine if t
he variable should be included in the model
train1 <- train %>%
  dplyr::select(-sales)
m_lr1 <- glm(left~.,data=train1, family="binomial")
anova(m_lr,m_lr1,test="LRT") # Because there is a significant difference (p-v
alue<0.05) when sales is included vs when it is not included, sales should be
included in the model even though some levels might not be significant
```

```
## Analysis of Deviance Table
##
## Model 1: left ~ satisfaction_level + last_evaluation + number_project +
##     average_montly_hours + time_spend_company + Work_accident +
##     promotion_last_5years + sales + salary
## Model 2: left ~ satisfaction_level + last_evaluation + number_project +
##     average_montly_hours + time_spend_company + Work_accident +
##     promotion_last_5years + salary
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1     11980     8970.6
## 2     11989     9007.2 -9  -36.577 3.131e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
train2 <- train %>%
  dplyr::select(-last_evaluation) # Remove next highest p-value
m_lr2 <- glm(left~., data=train2, family="binomial")
summary(m_lr2) # Now everything else is significant
```

```
##
## Call:
## glm(formula = left ~ ., family = "binomial", data = train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7939  -0.5566  -0.2665  -0.0557   3.4635
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -2.60129    0.19912 -13.064  < 2e-16 ***
## satisfaction_level     -3.57660    0.10263 -34.850  < 2e-16 ***
## number_project         -2.58429    0.13020 -19.849  < 2e-16 ***
## average_montly_hours    0.85857    0.13243   6.483 8.98e-11 ***
```

```
## time_spend_company      4.29295    0.12469  34.430  < 2e-16 ***
## Work_accident1         -1.58532    0.10626 -14.919  < 2e-16 ***
## promotion_last_5years1 -0.98428    0.28326  -3.475 0.000511 ***
## saleshr                 0.12826    0.15824   0.811 0.417642
## salesIT                -0.20105    0.14551  -1.382 0.167066
## salesmanagement        -0.19944    0.19069  -1.046 0.295629
## salesmarketing          0.00349    0.15683   0.022 0.982249
## salesproduct_mng       -0.15684    0.15449  -1.015 0.310012
## salesRandD             -0.64638    0.17071  -3.786 0.000153 ***
## salessales             -0.02552    0.12187  -0.209 0.834129
## salessupport            0.08676    0.12952   0.670 0.502949
## salestechnical          0.03604    0.12723   0.283 0.776954
## salarylow               1.89480    0.15014  12.620  < 2e-16 ***
## salarymedium            1.38900    0.15144   9.172  < 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: 13183.7  on 11998  degrees of freedom
## Residual deviance:  8970.8  on 11981  degrees of freedom
## AIC: 9006.8
##
## Number of Fisher Scoring iterations: 6
```

```
p_lr <- predict(m_lr2, test, type="response")
p_lr1 <- ifelse(p_lr>0.5,1,0)
confusionMatrix(p_lr1, test$left, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2106  328
##          1  185  381
##
##                Accuracy : 0.829
##                  95% CI : (0.815, 0.8423)
##     No Information Rate : 0.7637
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4908
##  Mcnemar's Test P-Value : 3.623e-10
##
##             Sensitivity : 0.5374
##             Specificity : 0.9192
##          Pos Pred Value : 0.6731
##          Neg Pred Value : 0.8652
##              Prevalence : 0.2363
##          Detection Rate : 0.1270
```

```
##    Detection Prevalence : 0.1887
##       Balanced Accuracy : 0.7283
##
##        'Positive' Class : 1
##
```

```
cat("% of false positives (FP) is \n",round((185/3000)*100,2), "% of false ne
gatives (FN) is \n",round((328/3000)*100,2))
```

```
## % of false positives (FP) is
##  6.17, % of false negatives (FN) is
##  10.93
```

```
# Capped data
train_cap <- hr.cap.norm[train_sample,]
test_cap <- hr.cap.norm[-train_sample,]
c_lr <- glm(left~.,data=train_cap,family="binomial")
summary(c_lr)
```

```
##
## Call:
## glm(formula = left ~ ., family = "binomial", data = train_cap)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4141  -0.6348  -0.3550  -0.0907   3.1058
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -2.04159    0.19318 -10.568  < 2e-16 ***
## satisfaction_level      -3.55929    0.10043 -35.439  < 2e-16 ***
## last_evaluation          0.36828    0.11002   3.347 0.000816 ***
## number_project          -1.90872    0.12435 -15.349  < 2e-16 ***
## average_montly_hours     0.96033    0.12757   7.528 5.16e-14 ***
## time_spend_company       2.51393    0.09986  25.175  < 2e-16 ***
## Work_accident1          -1.54761    0.10145 -15.255  < 2e-16 ***
## promotion_last_5years1 -1.34922    0.27253  -4.951 7.39e-07 ***
## saleshr                  0.11782    0.15050   0.783 0.433706
## salesIT                 -0.24495    0.13853  -1.768 0.077032 .
## salesmanagement         -0.41801    0.17890  -2.337 0.019464 *
## salesmarketing          -0.07994    0.15018  -0.532 0.594554
## salesproduct_mng        -0.21623    0.14757  -1.465 0.142834
## salesRandD              -0.63840    0.16432  -3.885 0.000102 ***
## salessales              -0.11141    0.11641  -0.957 0.338547
## salessupport             0.05462    0.12367   0.442 0.658714
## salestechnical          -0.01173    0.12168  -0.096 0.923227
## salarylow                2.02054    0.14681  13.763  < 2e-16 ***
## salarymedium             1.45017    0.14770   9.819  < 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: 13183.7  on 11998  degrees of freedom
## Residual deviance:  9875.6  on 11980  degrees of freedom
## AIC: 9913.6
##
## Number of Fisher Scoring iterations: 5
```

```
train_cap1 <- train_cap %>%
  dplyr::select(-sales)
c_lr1 <- glm(left~., data=train_cap1, family="binomial")
anova(c_lr1, c_lr, test="LRT") # sales should be included
```

```
## Analysis of Deviance Table
##
## Model 1: left ~ satisfaction_level + last_evaluation + number_project +
##     average_montly_hours + time_spend_company + Work_accident +
##     promotion_last_5years + salary
## Model 2: left ~ satisfaction_level + last_evaluation + number_project +
##     average_montly_hours + time_spend_company + Work_accident +
##     promotion_last_5years + sales + salary
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1     11989     9917.6
## 2     11980     9875.6  9    42.09 3.164e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
p_c_lr <- predict(c_lr, test_cap, type="response")
p_c_lr1 <- ifelse(p_c_lr>0.5,1,0)
confusionMatrix(p_c_lr1,test_cap$left, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2109  432
##          1  182  277
##
##                Accuracy : 0.7953
##                  95% CI : (0.7804, 0.8096)
##     No Information Rate : 0.7637
##     P-Value [Acc > NIR] : 1.861e-05
##
##                   Kappa : 0.3544
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.39069
##             Specificity : 0.92056
##          Pos Pred Value : 0.60349
##          Neg Pred Value : 0.82999
##              Prevalence : 0.23633
```

```
##          Detection Rate : 0.09233
##    Detection Prevalence : 0.15300
##       Balanced Accuracy : 0.65562
##
##        'Positive' Class : 1
##
```

```
cat("% of false positives (FP) is \n",round((182/3000)*100,2), "% of false ne
gatives (FN) is \n",round((432/3000)*100,2))
```

```
## % of false positives (FP) is
##  6.07, % of false negatives (FN) is
##  14.4
```

Evaluating model performance

There is a better model performance in the imputed data set than the capped data set. The former has a higher accuracy, kappa value, and lower false negative rate. The difference in model performance between the two data sets is greater than that of SVM. Because SVM is not sensitive to outliers while logistic regression is, different methods of dealing with outliers will affect logsitic regression models more than they affect SVM.

The logistic regression model for the imputed data set has an accuracy of 82.9%, kappa statistic of 0.49 (moderate agreement between predicted and true values), and an FN rate of 10.9%. The SVM model performs better than the logisitc regression model in these 3 aspects (accuracy, kappa value, and FN rate).

Feature selection

Backfitting is applied where features with the highest p-value are removed each time, until each of the remaining features have a p-value<0.05. For categorical features that have been dummy coded, certain levels are not statistically significant. The significance of the categorical variable as a whole is tested using a likelihood ratio test where the model with that categorical variable is compared to a model without that variable. If the test shows significant difference (p<0.05), then the model with that variable is significantly different from a model without that variable and the variable should be included in the model (StackExchange 2013). In this case, sales is a significant variable as a whole because the likelihood ratio test shows a significant difference.

The features selected from the logistic regression model (for the imputed data set) aligns with that of the MFA in the following ways:

- Time spend in the company has the highest absolute coefficient (coefficients can be compared since data set is normalized), which coincides with the MFA that time contributes most to the first dimension, that explains most variation in the data.
- All categorical features are included in the logistic regression model, which coincides with the MFA that categorical features contribute most to the second dimension.
- The MFA analyses indicate that all features should be kept as they contribute to the first two dimensions. The logisitc regression model includes all features except for last

evaluation, which is grouped with monthly hours and number of project in the MFA, meaning that the group as a whole should be kept.

Understanding why employees leave

Based on the logistic regression model, employees are more likely to leave if they:

- Have lower satisfaction level and number of projects (negatively correlated with the chance of leaving)
- Spend more years and time per month working in the company (positively correlated with the chance of leaving)
- Do not have work accident (switching from no work accident to having work accident decreases chance of leaving)
- Are not promoted within last 5 years (switching from no promotion to having a promotion decreases the chance of leaving)
- Have low or medium salary (low and medium salary are positively correlated with chance of leaving, while high salary decreases chance of leaving)
- Work in hr, marketing, support, and technical departments (these categories are positively correlated with the chance of leaving)

3. Decision Tree vs Random Forest

Decision Tree

```
set.seed(577)
library(RWeka)
# Imputed data set
m_d <- J48(left~.,data=hr_train)
p_d <- predict(m_d, hr_test)
confusionMatrix(p_d, hr_test$left, positive="1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2280   48
##          1   11  661
##
##                Accuracy : 0.9803
##                  95% CI : (0.9747, 0.985)
##     No Information Rate : 0.7637
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9445
##  Mcnemar's Test P-Value : 2.775e-06
##
##             Sensitivity : 0.9323
##             Specificity : 0.9952
##          Pos Pred Value : 0.9836
```

```
##          Neg Pred Value : 0.9794
##              Prevalence : 0.2363
##          Detection Rate : 0.2203
##    Detection Prevalence : 0.2240
##       Balanced Accuracy : 0.9637
##
##        'Positive' Class : 1
##
```

```
cat("% of false positives (FP) is \n",round((11/3000)*100,2), "% of false neg
atives (FN) is \n",round((48/3000)*100,2))
```

```
## % of false positives (FP) is
##  0.37, % of false negatives (FN) is
##  1.6
```

```
# Capped data set
m_d_cap <- J48(left~.,data=hr_train_cap)
p_d_cap <- predict(m_d_cap,hr_test_cap)
confusionMatrix(p_d_cap, hr_test_cap$left, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2277   51
##          1   14  658
##
##                Accuracy : 0.9783
##                  95% CI : (0.9725, 0.9832)
##     No Information Rate : 0.7637
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9389
##  Mcnemar's Test P-Value : 7.998e-06
##
##             Sensitivity : 0.9281
##             Specificity : 0.9939
##          Pos Pred Value : 0.9792
##          Neg Pred Value : 0.9781
##              Prevalence : 0.2363
##          Detection Rate : 0.2193
##    Detection Prevalence : 0.2240
##       Balanced Accuracy : 0.9610
##
##        'Positive' Class : 1
##
```

```
cat("% of false positives (FP) is \n",round((14/3000)*100,2), "% of false neg
atives (FN) is \n",round((51/3000)*100,2))
```

```
## % of false positives (FP) is
##  0.47, % of false negatives (FN) is
##  1.7
```

Evaluate model performance

In the case of the decision tree model, there is a very slight difference in model performance between the capped and imputed data set. The imputed data set perform slightly better in terms of slightly higher kappa value, accuracy, and slightly lower FN rate. Similar to SVM, decision trees are not sensitive to outliers, and the presence of outliers would not greatly affect model performance.

The decision tree model performs better than SVM.

- Decision tree: accuracy of 98%, kappa value of 0.94, and FN rate of 1.6%
- SVM: accuracy of 95.6%, kappa value of 0.88, and FN rate of 2.17%

Random Forest

```
library(randomForest)
# Imputed data set
rf <- randomForest(left~., data=hr_train)
rf_p <- predict(rf, hr_test)
confusionMatrix(rf_p, hr_test$left, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2286   27
##          1    5  682
##
##                Accuracy : 0.9893
##                  95% CI : (0.985, 0.9927)
##     No Information Rate : 0.7637
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9701
##  Mcnemar's Test P-Value : 0.0002054
##
##             Sensitivity : 0.9619
##             Specificity : 0.9978
##          Pos Pred Value : 0.9927
##          Neg Pred Value : 0.9883
##              Prevalence : 0.2363
##          Detection Rate : 0.2273
##    Detection Prevalence : 0.2290
##       Balanced Accuracy : 0.9799
##
##        'Positive' Class : 1
##
```

```
cat("% of false positives (FP) is \n",round((5/3000)*100,2), "% of false nega
tives (FN) is \n",round((27/3000)*100,2))
```

```
## % of false positives (FP) is
##  0.17, % of false negatives (FN) is
##  0.9
```

```
# Capped data set
rf_cap <- randomForest(left~., data=hr_train_cap)
rf_p_cap <- predict(rf, hr_test_cap)
confusionMatrix(rf_p_cap, hr_test_cap$left, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2287  359
##          1    4  350
##
##                Accuracy : 0.879
##                  95% CI : (0.8668, 0.8905)
##     No Information Rate : 0.7637
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5947
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4937
##             Specificity : 0.9983
##          Pos Pred Value : 0.9887
##          Neg Pred Value : 0.8643
##              Prevalence : 0.2363
##          Detection Rate : 0.1167
##    Detection Prevalence : 0.1180
##       Balanced Accuracy : 0.7460
##
##        'Positive' Class : 1
##
```

```
cat("% of false positives (FP) is \n",round((4/3000)*100,2), "% of false nega
tives (FN) is \n",round((359/3000)*100,2))
```

```
## % of false positives (FP) is
##  0.13, % of false negatives (FN) is
##  11.9
```

Evaluate model performance

Random forest model performs better on the imputed data set than the capped data set, and the difference is substantial. Even though random forests, SVM, and decision trees can handle missing or noisy data, methods to deal with outliers still affect model performance.

Across all models, the imputation strategy, which replaces outliers via a machine learning technique where the outcome variable is the column that contains missing values, produces higher performance than capping (simply replacing outliers with a certain value). This is because, the imputation technique (regression tree in the example dealing with outliers) produces more variation than the capping technique, and are more accurate representations of actual values (Prabhakaran 2017).

The random forest model on the imputed data set produces the best model performance amongst logistic regression, decision tree, and SVM. It results in an accuracy of 98.9%, kappa value of 0.97, and FN rate of 0.9%.

AUC

AUC calculated on the imputed data set with better performance for each model.

```
library(caTools)
# SVM
colAUC(as.numeric(p_svm), hr_test$left)

##              [,1]
## 0 vs. 1 0.9393201

# Logistic regression
colAUC(p_lr1, test$left)

##              [,1]
## 0 vs. 1 0.7283129

# Decision Tree
colAUC(as.numeric(p_d_cap), hr_test$left)

##              [,1]
## 0 vs. 1 0.9609784

# Random forest
colAUC(as.numeric(rf_p), hr_test$left)

##              [,1]
## 0 vs. 1 0.9798679
```

Based on the AUC, random forest has the best model performance, followed by decision tree, SVM, and lastly, logistic regression. Results from the AUC coincides with other metrics such as percentage accuracy and kappa statistic, which output the same rank in terms of model performance.

Stacked ensemble model

I will build a stacked emsemble model by first combining multiple models (SVM, logistic regression, and decision tree, not including random forest because that would take too long to run), then utilize another model (random forest) to learn a combination function from the combined models, via the caretEnsemble package.

I will apply 10-fold cross validation in each model (SVM, logistic regression, and decision tree) for evaluation of fit and select the simplest model (most parsimonous according to the oneSE function) to avoid overfitting.

Combining models

Imputed data set used since it performs better.

```
library(caretEnsemble)
set.seed(577)
hr_train1 <- hr_train %>%
  mutate(left=factor(left, labels=c("leave","stay"), levels=c(1,0)))
control <- trainControl(method="cv", selectionFunction = "oneSE", savePredict
ions=T, classProbs = T) # Conduct 10 fold cross validation for each model. Se
lect the simplest result according to oneSE rule
algorithms <- c("glm","J48","svmRadial") # models to combine
models <- caretList(left~., hr_train1, trControl=control,methodList=algorithm
s)
models
```

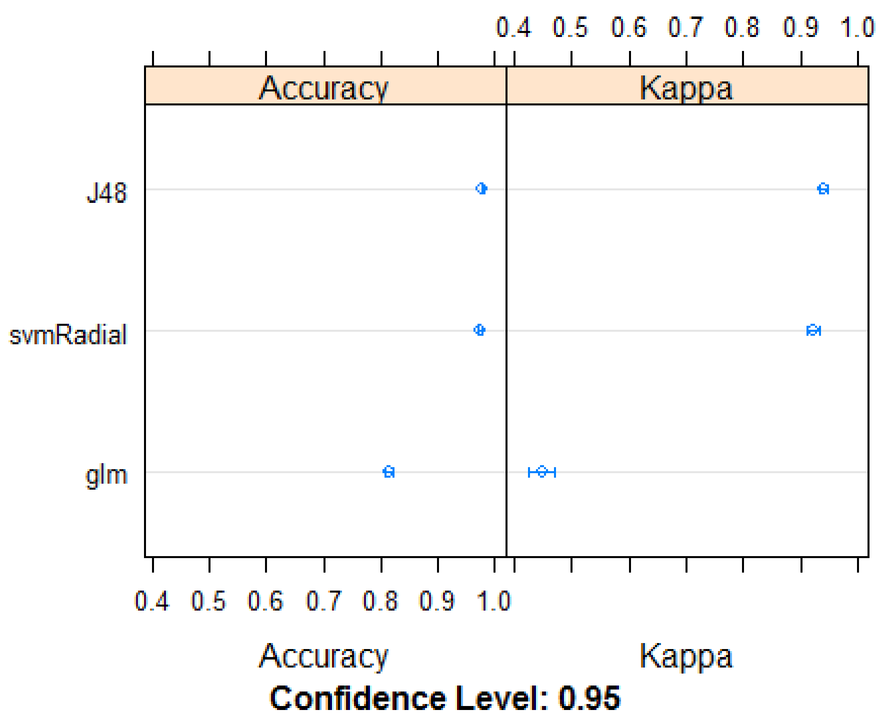
```
## $glm
## Generalized Linear Model
##
## 11999 samples
##    18 predictor
##     2 classes: 'leave', 'stay'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10800, 10799, 10799, 10799, 10798, 10798, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8137358  0.4479694
##
##
## $J48
## C4.5-like Trees
##
## 11999 samples
##    18 predictor
##     2 classes: 'leave', 'stay'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10800, 10799, 10799, 10799, 10798, 10798, ...
## Resampling results across tuning parameters:
##
##   C      M  Accuracy   Kappa
##   0.010  1  0.9771644  0.9357003
```

```
##   0.010  2  0.9770811  0.9354395
##   0.010  3  0.9771645  0.9356661
##   0.255  1  0.9782477  0.9391180
##   0.255  2  0.9779144  0.9381201
##   0.255  3  0.9787480  0.9403479
##   0.500  1  0.9787469  0.9411449
##   0.500  2  0.9779139  0.9388043
##   0.500  3  0.9767481  0.9354869
## 
## Accuracy was used to select the optimal model using  the one SE rule.
## The final values used for the model were C = 0.255 and M = 1.
## 
## $svmRadial
## Support Vector Machines with Radial Basis Function Kernel 
## 
## 11999 samples
##    18 predictor
##     2 classes: 'leave', 'stay' 
## 
## No pre-processing
## Resampling: Cross-Validated (10 fold) 
## Summary of sample sizes: 10800, 10799, 10799, 10799, 10798, 10798, ... 
## Resampling results across tuning parameters:
## 
##   C     Accuracy   Kappa    
##   0.25  0.9567450  0.8764258
##   0.50  0.9674957  0.9081175
##   1.00  0.9722463  0.9221519
## 
## Tuning parameter 'sigma' was held constant at a value of 0.0400196
## Accuracy was used to select the optimal model using  the one SE rule.
## The final values used for the model were sigma = 0.0400196 and C = 1.
## 
## attr(,"class")
## [1] "caretList"
```

```
results <- resamples(models)
summary(results)
```

```
## 
## Call:
## summary.resamples(object = results)
## 
## Models: glm, J48, svmRadial 
## Number of resamples: 10 
## 
## Accuracy 
##             Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## glm       0.7958  0.8097 0.8133 0.8137  0.8173 0.8367    0
## J48       0.9725  0.9769 0.9775 0.9782  0.9783 0.9842    0
```

```
## svmRadial 0.9650  0.9687 0.9708 0.9722  0.9771 0.9783    0
##
## Kappa
##                Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## glm       0.3900  0.4268 0.4551 0.4480  0.4667 0.5049    0
## J48       0.9227  0.9354 0.9372 0.9391  0.9394 0.9558    0
## svmRadial 0.9015  0.9120 0.9180 0.9222  0.9362 0.9403    0

dotplot(results)
```



The above plot shows that the decision tree model has the highest accuracy and kappa value on average, followed by SVM, and lastly logistic regression model.

```
modelCor(results) # Check that correlations are not >0.75, because then model
s would be making similar predictions most of the time, reducing the benefit
of combining predictions.

##                 glm       J48 svmRadial
## glm       1.0000000 0.2642496 0.1148707
## J48       0.2642496 1.0000000 0.6103675
## svmRadial 0.1148707 0.6103675 1.0000000
```

Although the decision tree model and SVM are correlated, the correlations are not very strong (<0.75).

Stacking

I will tune the model based on bootstrap sampling where 10 random training and testing data sets (with replacement) are selected and the model that results in the highest accuracy (after testing on the bootstrapped sample) is selected.

```
set.seed(577)
stack.rf <- caretStack(models, method="rf", metric="Accuracy", trControl=trai
nControl(method="boot", number=10, classProbs = T)) # tune model in trainCont
rol parameter

## note: only 2 unique complexity parameters in default grid. Truncating the
grid to 2 .

stack.rf

## A rf ensemble of 2 base models: glm, J48, svmRadial
##
## Ensemble results:
## Random Forest
##
## 11999 samples
##     3 predictor
##     2 classes: 'leave', 'stay'
##
## No pre-processing
## Resampling: Bootstrapped (10 reps)
## Summary of sample sizes: 11999, 11999, 11999, 11999, 11999, 11999, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9807002  0.9460688
##   3     0.9796844  0.9432275
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.

hr_test1 <- hr_test %>%
  mutate(left=factor(left, labels=c("leave","stay"), levels=c(1,0)))
p_models <- predict(stack.rf, hr_test1, type="prob") # Output probability of
positive class (leave)
colAUC(p_models, hr_test1$left)

##                      [,1]
## leave vs. stay 0.976839
```

Model performance of stacked ensemble has a higher accuracy (98.1%) and kappa value (0.95) than any individual model (SVM, decision tree or logistic regression). However, the ensemble model, random forest, has a slightly higher accuracy (98.9%) and kappa value (0.97) than the stacked ensemble model.

After predicting the testing data set, the stacked ensemble model yields an AUC of 0.977, which is higher than the AUC of individual model, but slightly lower than the AUC of random forest (0.980).

Conclusion

Comparison of models

On imputed data set that performs better than the capped data set.

	SVM	Logistic Regression	Decision Tree	Random Forest	Stacked Ensemble*
Accuracy	95.6%	82.9%	98.0%	98.9%	98.1%
Kappa	0.877	0.491	0.945	0.971	0.946
AUC	0.939	0.728	0.961	0.980	0.977

*Stacked ensemble is a combination of SVM, logistic regression, and decision tree models, with random forest as the combination function.

In terms of AUC, % accuracy, and kappa statistic, model ranking in ascending order:

1. Random Forest
2. Stacked Ensemble
3. Decision Tree
4. SVM
5. Logistic Regression

Random forest performs slightly better than the stacked ensemble. Given the long computation time to run the stacked ensemble model (~35 minutes) compared to that of random forest (~3 minutes), random forest model is preferred because it takes much less time to run but produces a better performance.

Models that are insensitive to noisy and missing data such as random forest, decision tree, and SVM, perform better than models that are sensitive to noisy and missing data (logistic regression). Decision tree, a white box model performs better than SVM, a black box model. In such case, decision tree is preferred because organizations can easily understand how the decision is made. Overall, ensemble models (random forest and stacked ensemble) perform better than individual models because multiple weaker learners are combined to form a stronger learner. If the organization wants to know how a decision is arrived, then the decision tree model is most suitable due to its high performance and transparency. Otherwise, a random forest model would be preferred due to its better performance.

Interpretation of results/prediction with interval

For prediction with interval, I will be using 95% confidence interval (CI) to output the probability of whether an employee will leave or stay given an unknown case, based on the random forest and stacked ensemble models.

95% CI formula

$$CI = x \pm 1.96se$$

$$se = \frac{s\,d(x)}{\sqrt{n-1}}$$

```
# Interval for random forest
rf_prob <- predict(rf, hr_test, type="prob")
se <- (sd(rf_prob)/(sqrt((nrow(rf_prob))-1)))
intervals <- 1.96*se
# Interval for stacked model
se_stacked <- (sd(p_models)/(sqrt((length(p_models))-1)))
intervals_stacked <- 1.96*se_stacked
```

Unknown case

- satisfaction level=0.1
- last evaluation=0.9
- number of projects =2
- average monthly hours=140
- time spend in company =2
- No work place accident
- Not promoted in last 5 years
- medium salary
- work in accounting department

```
hr_unknown <- rbind(hr_test, c(0.1,0.9,2,140,2,0,NA,0,0,0,0,0,0,0,0,0,0,0,1))
%>%
  slice(3001)
# Normalize numeric features as done for known cases
hr_unknown$satisfaction_level <- (hr_unknown$satisfaction_level-min(hr$satisf
action_level))/(max(hr$satisfaction_level)-min(hr$satisfaction_level))
hr_unknown$last_evaluation <- (hr_unknown$last_evaluation-min(hr$last_evaluat
ion))/(max(hr$last_evaluation)-min(hr$last_evaluation))
hr_unknown$number_project <- (hr_unknown$number_project-min(hr$number_project
))/(max(hr$number_project)-min(hr$number_project))
hr_unknown$average_montly_hours <- (hr_unknown$average_montly_hours-min(hr$av
erage_montly_hours))/(max(hr$average_montly_hours)-min(hr$average_montly_hour
s))hr_unknown$time_spend_company <- (hr_unknown$time_spend_company <-min(hr$t
ime_spend_company))/(max(hr$time_spend_company)-min(hr$time_spend_company))
```

Confidence interval using random forest

```
# Outputs probability of both classes
upp <- predict(rf, hr_unknown, type="prob")+intervals
low <- predict(rf, hr_unknown, type="prob")-intervals
cat("The 95% confidence interval of an employee staying ranges from \n", roun
d(low[1],2), "to \n", round(upp[1],2))
```

```
## The 95% confidence interval of an employee staying ranges from
##  0.61 to
##  0.64
```

```
cat("The 95% confidence interval of an employee leaving ranges from \n", roun
d(low[2],2), "to \n", round(upp[2],2))
```

```
## The 95% confidence interval of an employee leaving ranges from
##  0.36 to
##  0.39
```

```
cat("Since there is a higher probability that the employee will stay, the ran
dom forest model predicts that an employee with the following characteristics
will stay.")
```

```
## Since there is a higher probability that the employee will stay, the rando
m forest model predicts that an employee with the following characteristics w
ill stay.
```

Confidence interval using stacked model

```
# Outputs probability of positive class (leave=1)
upp_p <- predict(stack.rf, hr_unknown, type="prob") + intervals_stacked
low_p <- predict(stack.rf, hr_unknown, type="prob") - intervals_stacked
prob_stay <- 1- predict(stack.rf, hr_unknown, type="prob")
upp_n <- prob_stay+intervals_stacked
low_n <- prob_stay-intervals_stacked
cat("The 95% confidence interval of an employee leaving ranges from \n", roun
d(low_p,2), "to \n", round(upp_p,2))
```

```
## The 95% confidence interval of an employee leaving ranges from
##  0.1 to
##  0.13
```

```
cat("The 95% confidence interval of an employee staying ranges from \n", roun
d(low_n,2), "to \n", round(upp_n,2))
```

```
## The 95% confidence interval of an employee staying ranges from
##  0.87 to
##  0.9
```

```
cat("Since there is a higher probability that the employee will stay, the sta
cked ensemble model predicts that an employee with the following characterist
ics will stay.")
```

```
## Since there is a higher probability that the employee will stay, the stack
ed ensemble model predicts that an employee with the following characteristic
s will stay.
```

Both models result in the same prediction but the stacked ensemble model is more confident in predicting that an employee will stay.

References

Business Fillings. (2017). Identifying and addressing employee turnover issues. *BizFillings*. Accessed December 9, 2017 from https://www.bizfilings.com/toolkit/research-topics/office-hr/identifying-and-addressing-employee-turnover-issues

kaggle. (2017). HR analytics. *kaggle*. Accessed November 10, 2017 from https://www.kaggle.com/ludobenistant/hr-analytics

StackExchange. (2013). How to test for simultaneous equality of chosen coefficients in logit or probit model? *StackExchange*. Accessed December 8, 2017 from https://stats.stackexchange.com/questions/59085/how-to-test-for-simultaneous-equality-of-choosen-coefficients-in-logit-or-probit/59093#59093

Prabhakaran, S. (2017). Outlier treatment. *R-statistics.co*. Accessed December 7, 2017 from http://r-statistics.co/Outlier-Treatment-With-R.html