Salifort Capstone Project

ng khuang yang

2024-06-21

R Markdown

The HR department at Salifort Motors wants to take some initiatives to improve employee satisfaction levels at the company. Because it is time-consuming and expensive to find, interview, and hire new employees, increasing employee retention will be beneficial to the company.

Your goals in this project are to analyze the data collected by the HR department and to build a model that predicts whether or not an employee will leave the company.

Understand your dataset

The dataset that you'll be using in this lab contains 15,000 rows and 10 columns for the variables listed below.

For more information about the data, refer to its source on Kaggle link.

```
# import dataset
df <- read.csv("C:/Users/khuan/Downloads/salifort capstone/HR_capstone_dataset.csv", header = TRUE)
```

```
# display colnames
colnames(df)
```

```
##  [1] "satisfaction_level"    "last_evaluation"       "number_project"
##  [4] "average_montly_hours"  "time_spend_company"    "Work_accident"
##  [7] "left"                  "promotion_last_5years" "Department"
## [10] "salary"
```

Variable	Description
satisfaction_level	Employee-reported job satisfaction level [0–1]
last_evaluation	Score of employee's last performance review [0–1]
number_project	Number of projects employee contributes to
average_monthly_hours	Average number of hours employee worked per month
time_spend_company	How long the employee has been with the company (years)
Work_accident	Whether or not the employee experienced an accident while at work
left	Whether or not the employee left the company
promotion_last_5years	Whether or not the employee was promoted in the last 5 years
Department	The employee's department
salary	The employee's salary (U.S. dollars)

```
# install and activate libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
```

```
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ggplot2)
library(dplyr)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library('fastDummies')
```

```
## Thank you for using fastDummies!
## To acknowledge our work, please cite the package:
## Kaplan, J. & Schlegel, B. (2023). fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(pscl)
```

```
## Classes and Methods for R originally developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University (2002-2015),
## by and under the direction of Simon Jackman.
## hurdle and zeroinfl functions by Achim Zeileis.
```

```
library(e1071)
library(ISLR)
library(rpart)
library(rpart.plot)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:gridExtra':
##
##     combine
##
## The following object is masked from 'package:psych':
##
##     outlier
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(vip)
```

```
##
## Attaching package: 'vip'
##
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
library(tinytex)
```

```
# rename col names
setnames(df,  old=c("average_montly_hours"), new=c("average_monthly_hours"))
setnames(df,  old=c("time_spend_company"), new=c("tenure"))
# set col to lowercase
df <-rename_with(df,tolower)
```

```
# determine NA
sum(is.na(df))
```

```
## [1] 0
```

```
# determine duplicated
sum(duplicated(df))
```

```
## [1] 3008
```

```
# filter out duplicated col and evaluated it
df_duplicated <- df %>% mutate(duplicate = duplicated(df))
df_duplicated <- df_duplicated %>% filter(duplicate=='TRUE')
df_duplicated <- sort_by(df_duplicated, list(df_duplicated$satisfaction_level,df_duplicated$last_evalua
```

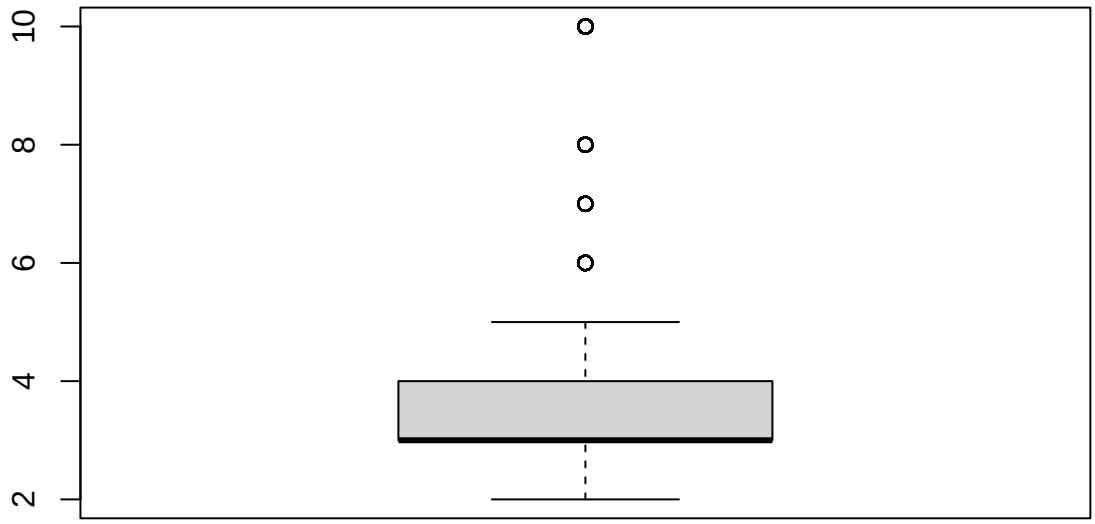
```
# show duplicated dataset
head(df_duplicated, n = c(6, 10))
```

```
##      satisfaction_level last_evaluation number_project average_monthly_hours
## 40                 0.09            0.62              6                   294
## 2251               0.09            0.62              6                   294
## 81                 0.09            0.77              5                   275
## 2292               0.09            0.77              5                   275
## 662                0.09            0.77              6                   290
## 2873               0.09            0.77              6                   290
##      tenure work_accident left promotion_last_5years  department salary
## 40        4             0    1                     0  accounting    low
## 2251      4             0    1                     0  accounting    low
## 81        4             0    1                     0 product_mng medium
## 2292      4             0    1                     0 product_mng medium
## 662       4             0    1                     0   technical medium
## 2873      4             0    1                     0   technical medium
```

```
# remove duplicated
df1 <- distinct(df)
# confirm no duplicated
sum(duplicated(df1))
```

```
## [1] 0
```

```r
# plot tenure boxplot
boxplot(df1$tenure)
```




```r
# determine upper & lower limit
q1 <- quantile(df1$tenure,probs = 0.25)
q3 <- quantile(df1$tenure,probs = 0.75)
iqr <- q3-q1
upper_limit <- q3+(iqr*1.5)
lower_limit <- q1-(iqr*1.5)
print(upper_limit)
```

```
## 75%
## 5.5
```

```r
print(lower_limit)
```

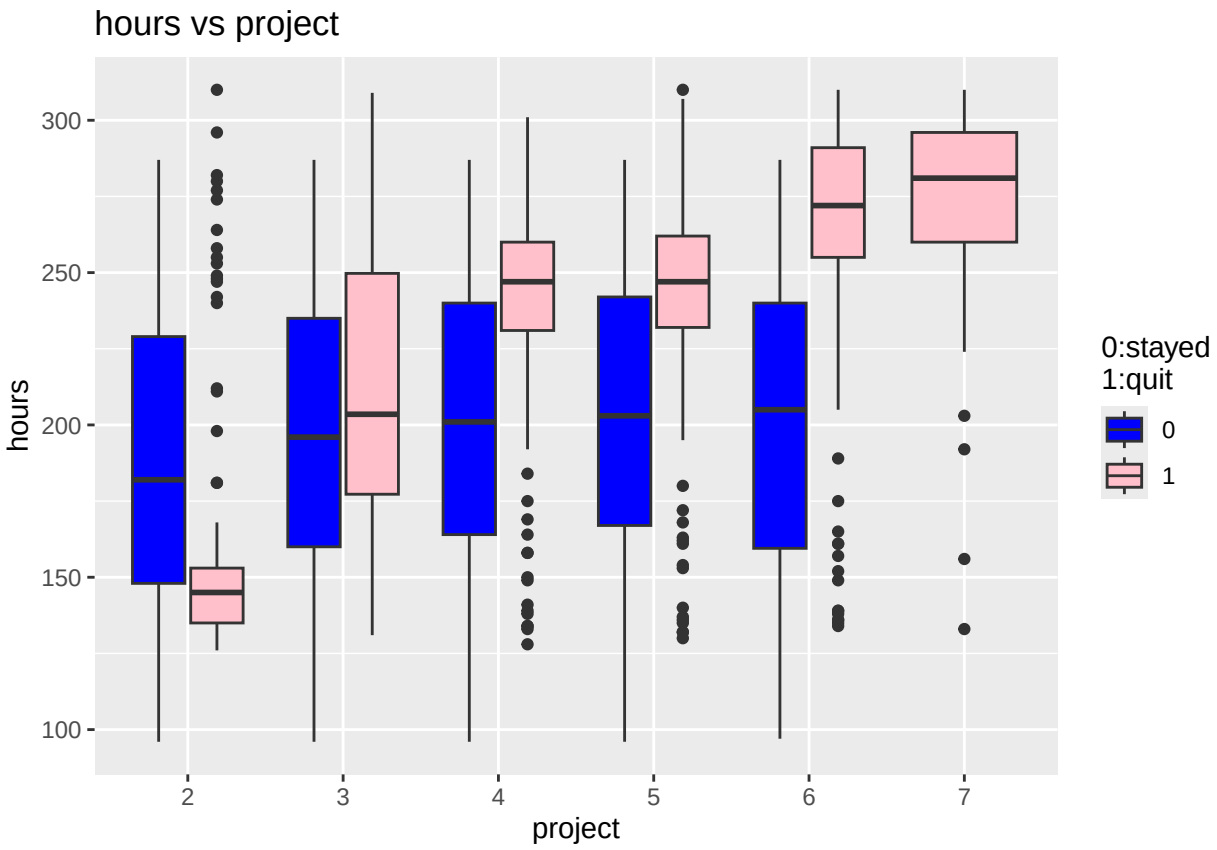
```
## 25%
## 1.5
```

```r
# draw left table
df_left <- table(df1$left)
df_left <- data.frame(df_left)
df_left <- df_left %>% mutate(percent = Freq/sum(Freq))
df_left
```

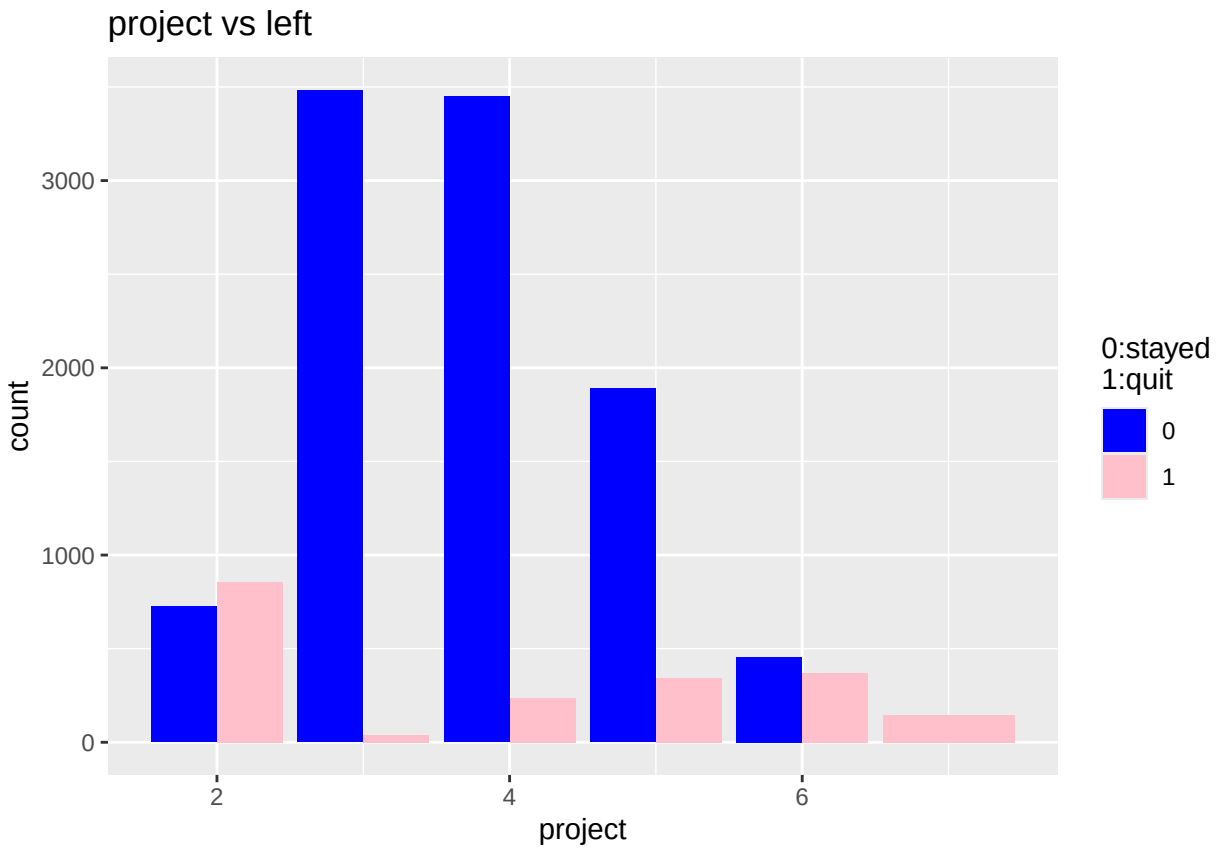
```
##   Var1  Freq   percent
## 1    0 10000 0.8339588
## 2    1  1991 0.1660412
```

Data vitualisation

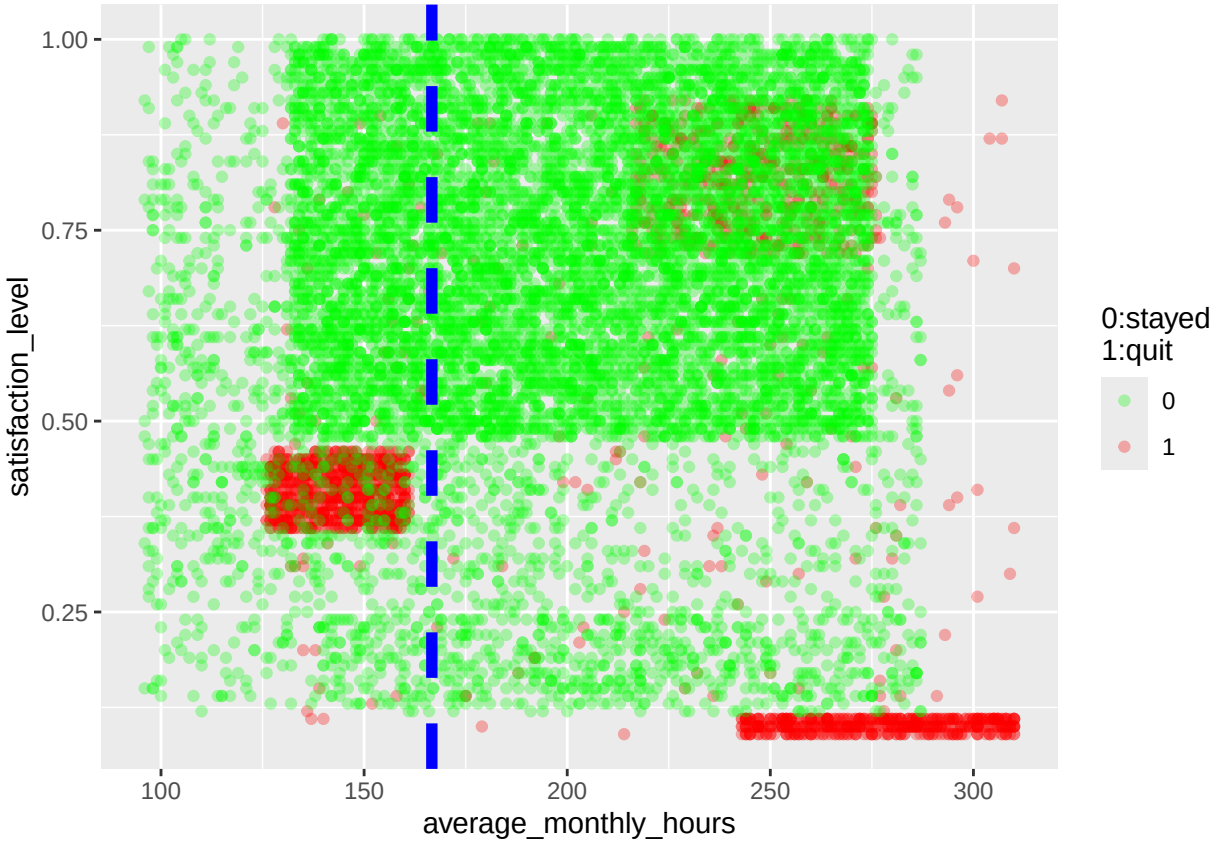
```
# boxplot : hours vs projects (left)
ggplot() + geom_boxplot(data= df1, mapping= aes(x=as.character(number_project), y=average_monthly_hours
labs(title="hours vs project", x="project", y="hours")+ scale_fill_manual('0:stayed\n1:quit', values=c(
```



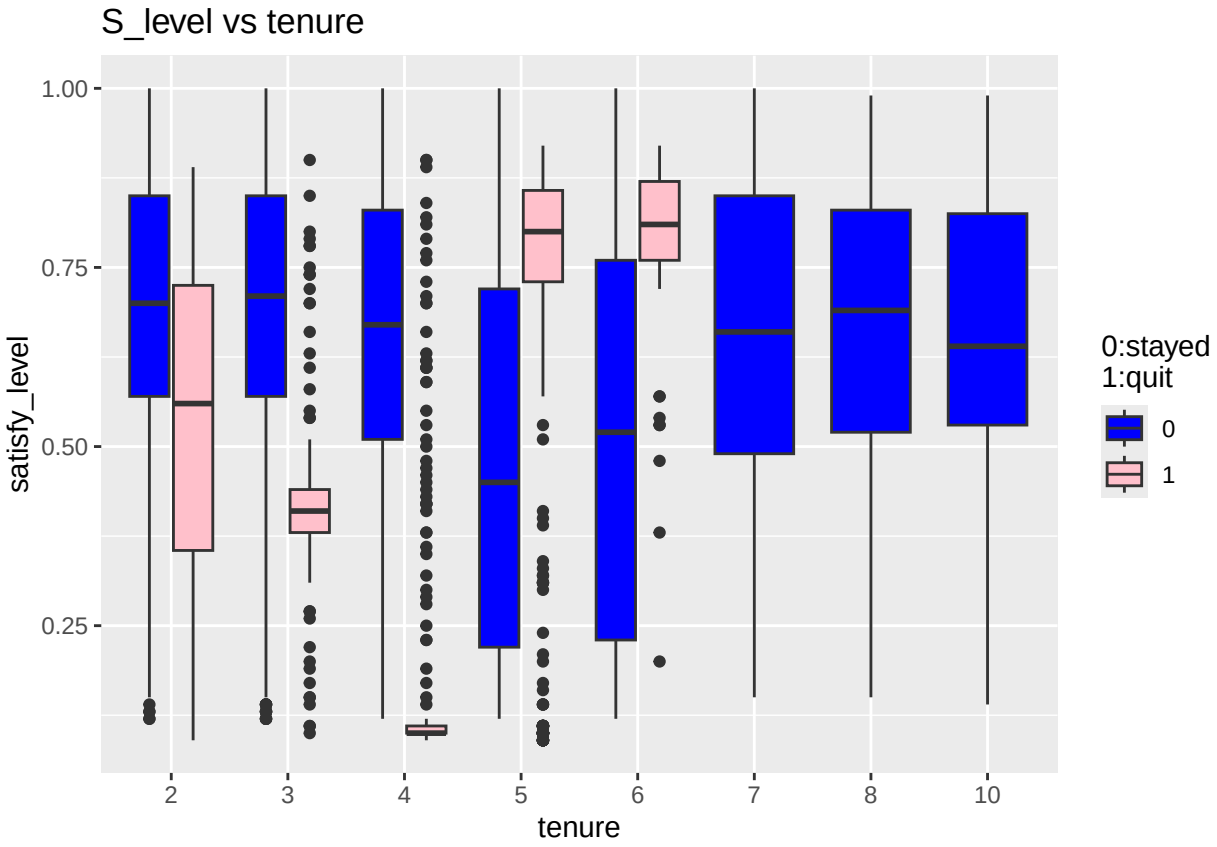

```
# barplot : projects vs left
ggplot() + geom_bar(position='dodge',data= df1, mapping= aes(x=number_project, fill = as.factor(left)))
labs(title="project vs left", x="project", y="count")+scale_fill_manual('0:stayed\n1:quit', values=c('b
```



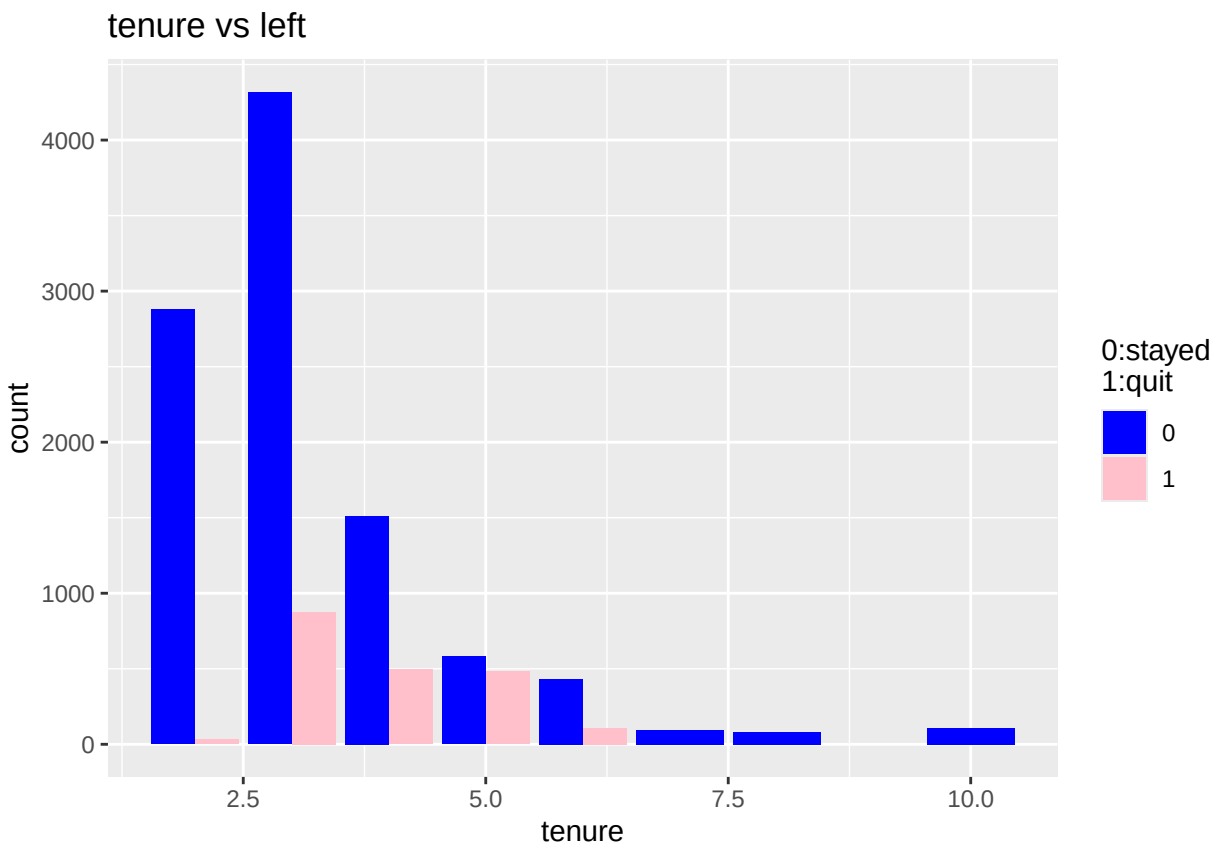

```
# scatterplot : hours vs sat_level (left)
scatter1 <- ggplot(df1, aes(x = average_monthly_hours, y = satisfaction_level)) +
  geom_point(aes(color = as.factor(left)),alpha=0.3)+scale_color_manual('0:stayed\n1:quit', values=c('g
  geom_vline(xintercept=166.67, linetype='dashed', color='blue', linewidth=2)
scatter1
```



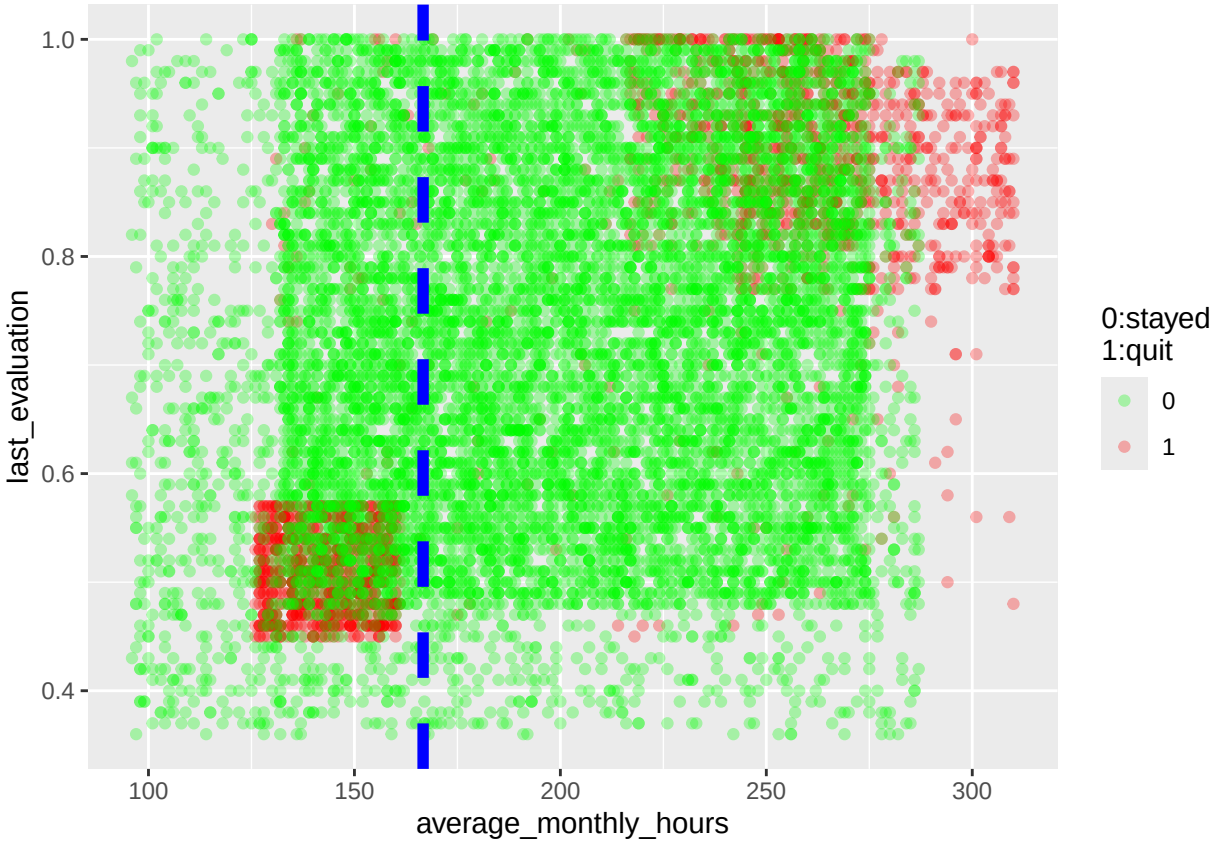

```
# boxplot : satis_level vs tenure (left)
ggplot() + geom_boxplot(data= df1, mapping= aes(x=as.factor(tenure), y=satisfaction_level,fill=as.facto
  labs(title="S_level vs tenure", x="tenure", y="satisfy_level")+ scale_fill_manual('0:stayed\n1:quit',
```




```
# barplot : tenure vs left
ggplot() + geom_bar(position='dodge',data= df1, mapping= aes(x=tenure, fill = as.factor(left)))+
  labs(title="tenure vs left", x="tenure", y="count")+scale_fill_manual('0:stayed\n1:quit', values=c('bl
```




```
# scatterplot : hours vs eval (left)
scatter2 <- ggplot(df1, aes(x = average_monthly_hours, y = last_evaluation)) +
  geom_point(aes(color = as.factor(left)),alpha=0.3)+scale_color_manual('0:stayed\n1:quit', values=c('g
  geom_vline(xintercept=166.67, linetype='dashed', color='blue', linewidth=2)
scatter2
```



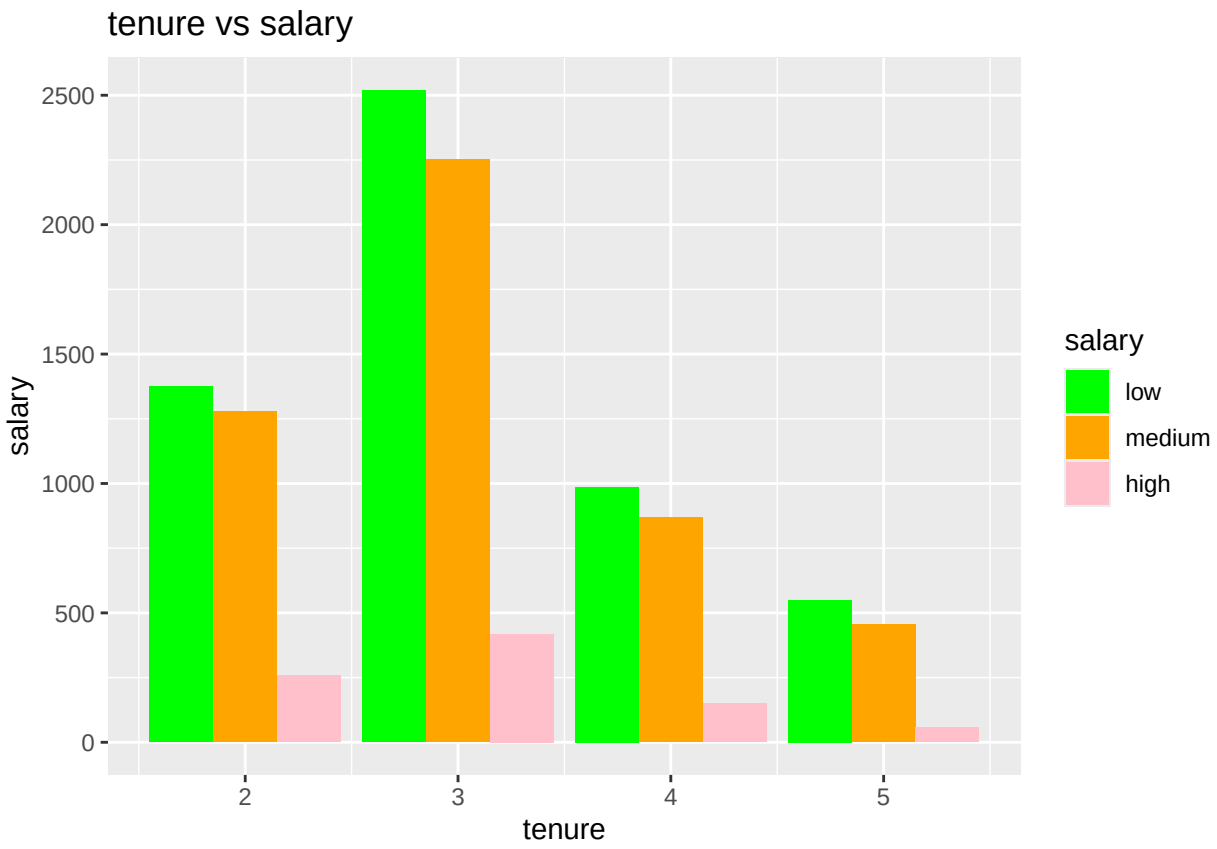

```
# table : mean & median satisf_level
table_mean_median_SLevel <- df1 %>% group_by(df1$left) %>% select(satisfaction_level) %>% summarize(mea
```

```
## Adding missing grouping variables: `df1$left`
```

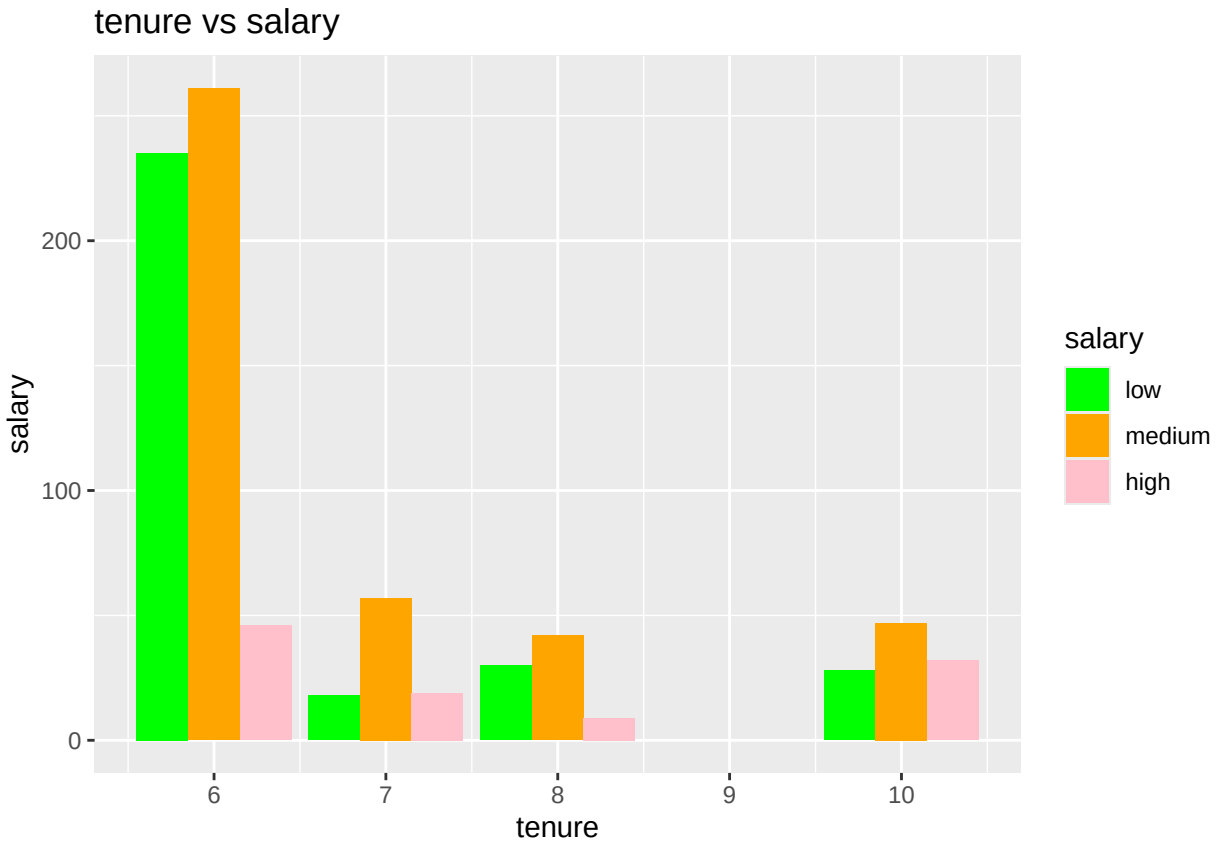
```
table_mean_median_SLevel
```

```
## # A tibble: 2 x 3
##   `df1$left`  mean `median(satisfaction_level)`
##        <int> <dbl>                         <dbl>
## 1          0 0.667                          0.69
## 2          1 0.440                          0.41
```

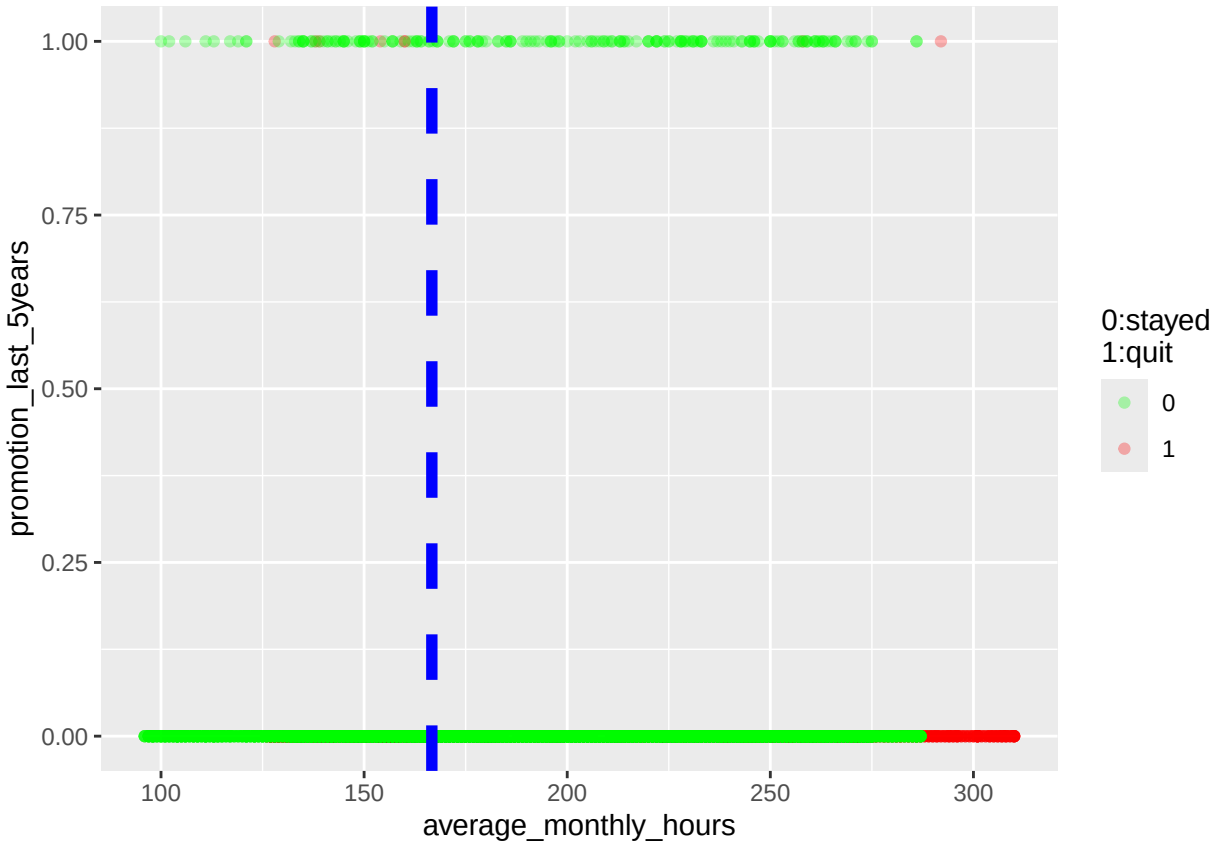
```
# split tenure into 2 groups(1-5,6-10)
salary_5 <- df1 %>% filter(tenure<=5)
salary_10 <- df1 %>% filter(tenure>5)
# barplot : salary vs tenure 1-5
ggplot() + geom_bar(position='dodge',data= salary_5, mapping= aes(x=tenure, fill = factor(salary,levels=
  labs(title="tenure vs salary", x="tenure", y="salary")+scale_fill_manual('salary', values=c('green','
```



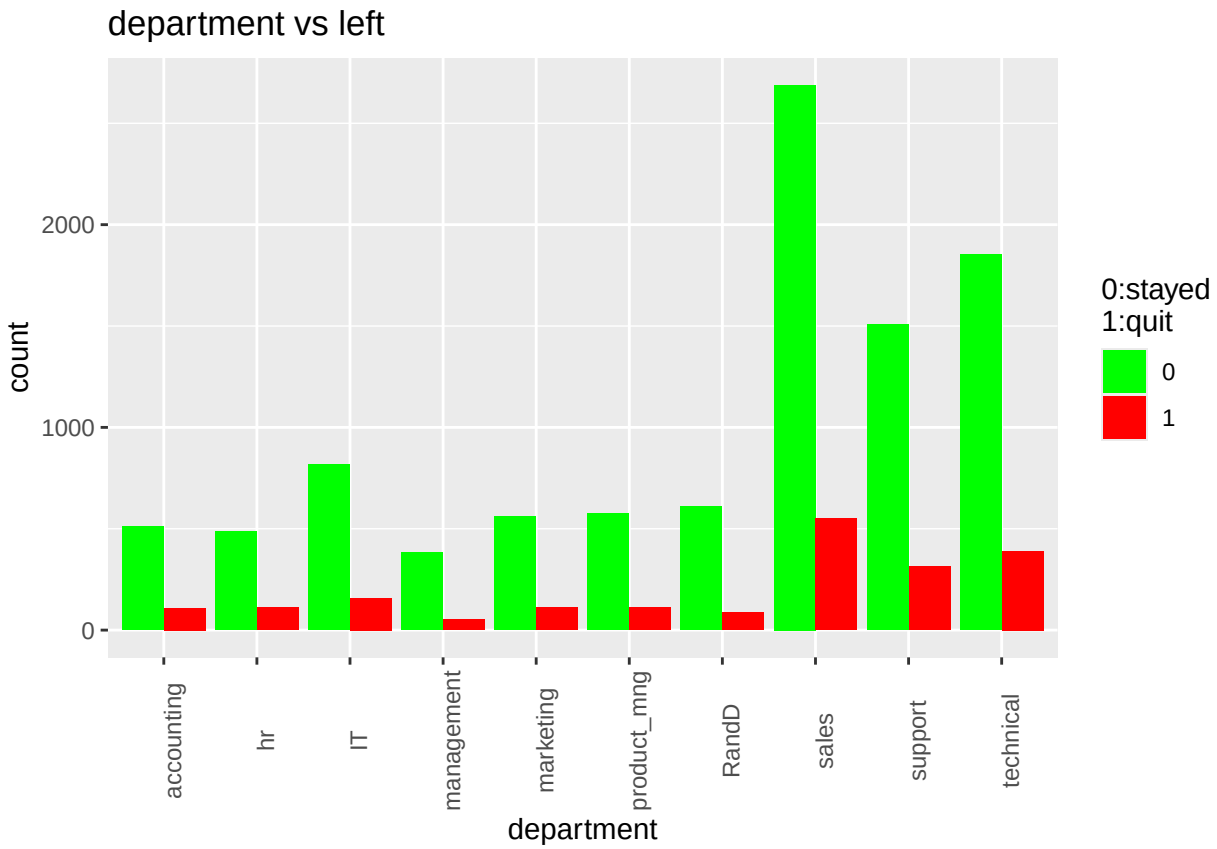

```
# barplot : salary vs tenure 6-10
ggplot() + geom_bar(position='dodge',data= salary_10, mapping= aes(x=tenure, fill = factor(salary,levels
  labs(title="tenure vs salary", x="tenure", y="salary")+scale_fill_manual('salary', values=c('green','
```



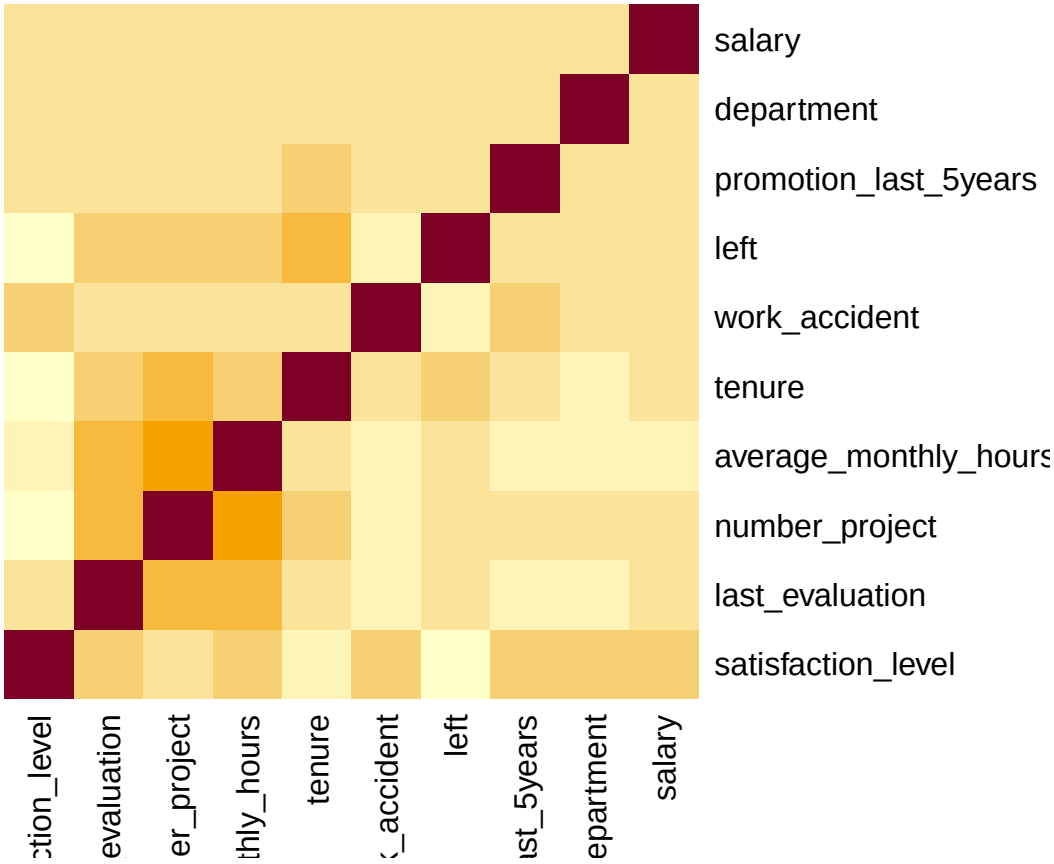

```
# scatterplot : hours vs promotion (left)
ggplot(df1, aes(x = average_monthly_hours, y = promotion_last_5years)) +
  geom_point(aes(color = as.factor(left)),alpha=0.3)+scale_color_manual('0:stayed\n1:quit', values=c('g
  geom_vline(xintercept=166.67, linetype='dashed', color='blue', linewidth=2,show.legend = TRUE)
```



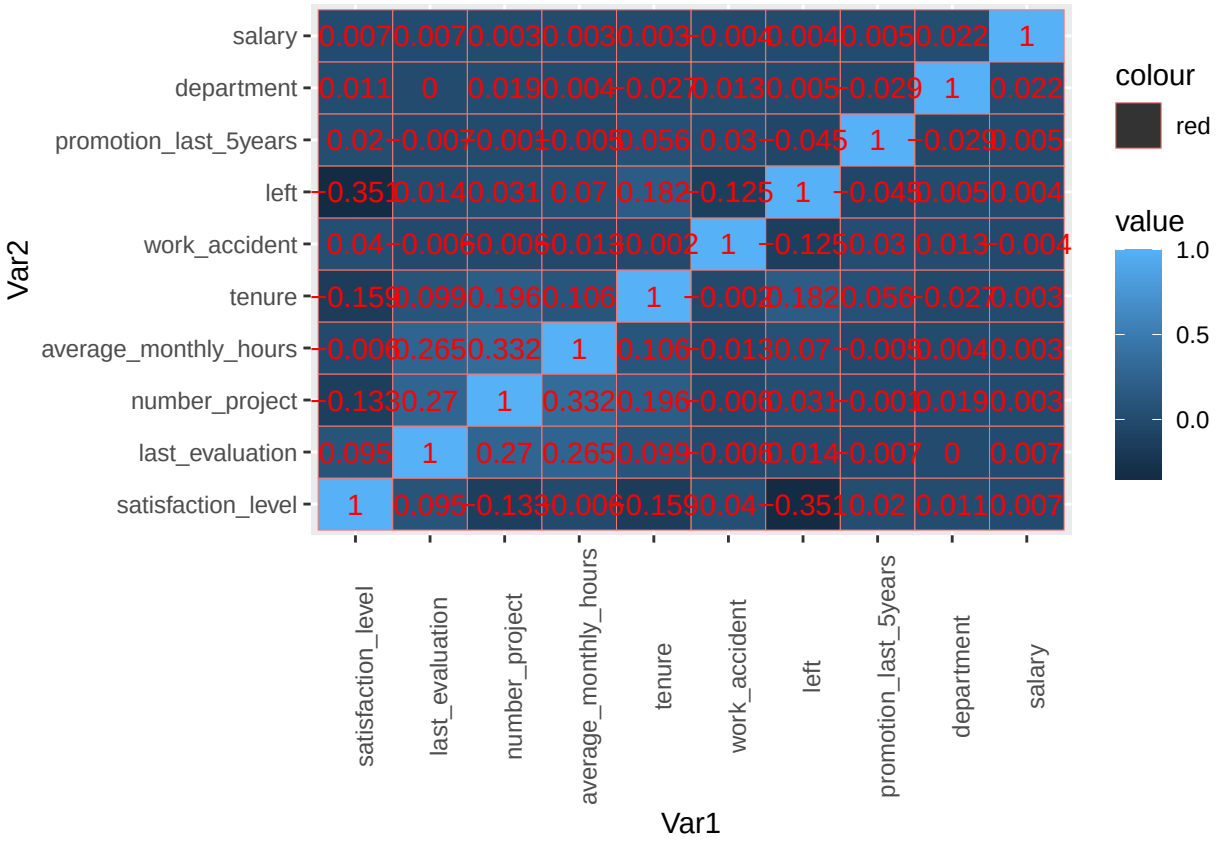

```
# barplot : departments vs left
ggplot() + geom_bar(position='dodge',data= df1, mapping= aes(x=department, fill = as.factor(left)))+
  labs(title="department vs left", x="department", y="count")+theme(axis.text.x = element_text(angle = 9
  scale_fill_manual('0:stayed\n1:quit', values=c('green','red'))
```




```
# heatmap
df2 <- mutate_all(df1, function(x) as.numeric(as.factor(x)))
heatmap(cor(df2),Rowv = NA, Colv = NA)
```



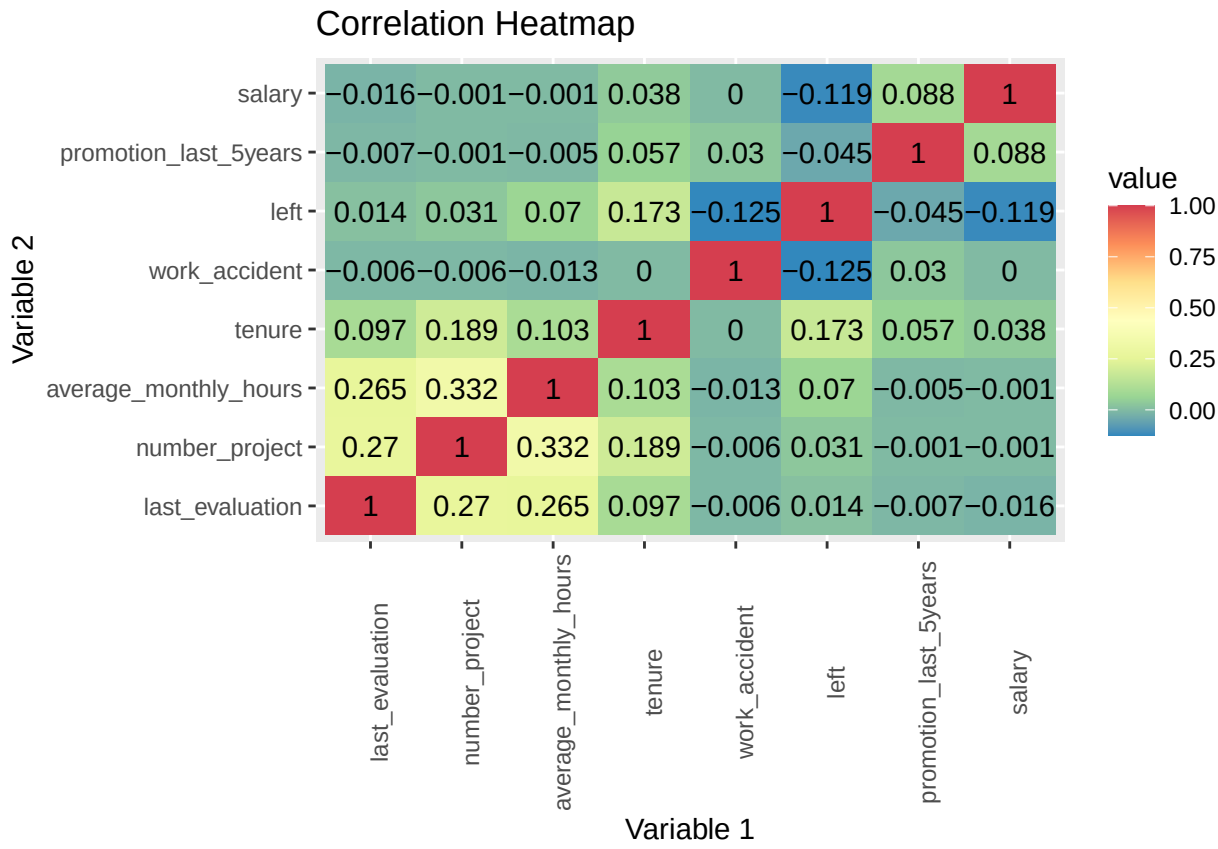

```
# creating correlation matrix
corr_mat <- round(cor(df2),3)
melted_corr_mat <- melt(corr_mat)
heatmap2 <- ggplot(data = melted_corr_mat, aes(x=Var1, y=Var2, fill=value, colour = 'red')) +
  geom_tile() + geom_text(aes(Var2, Var1, label = value),
            color = "red", size = 4)+theme(axis.text.x = element_text(angle = 90))
heatmap2
```



Building binary logistic regression

```
# dummy variables
df_enco <- df1
df_enco$left <- as.numeric(df_enco$left)
df_enco$salary <- factor(df_enco$salary,levels=c('low','medium','high'))
df_enco$salary <- as.numeric(df_enco$salary)
df_enco <- dummy_cols(df_enco, select_columns = c('department'),
                      remove_selected_columns = TRUE)
```

```
# filter out departments
df_enco_flt <- df_enco %>% select(last_evaluation,number_project,average_monthly_hours,tenure,work_acci
# plot heatmap
cor_df2 <- round(cor(df_enco_flt),3)
melt_df <- melt(cor_df2)
heatmap3 <- ggplot(melt_df,aes(x = Var1, y = Var2,fill = value))+
  geom_tile() + scale_fill_distiller(palette = "Spectral")+
  geom_tile() +
  labs(title = "Correlation Heatmap",
       x = "Variable 1",
       y = "Variable 2") + geom_text(aes(Var2, Var1, label = value),
                                     color = "black", size = 4)+theme(axis.text.x = element_text(angle =
heatmap3
```




```
# filtered out outliers
df_logreg <- df_enco %>% filter(tenure>=1.5&tenure<=5.5)
```

```
set.seed(42)
# training dataset
indexset <- createDataPartition(df_logreg$left,p = 0.75,list = F)
train <- df_logreg[indexset,]
test <- df_logreg[-indexset,]
```

```
# fit logistic regression model
model <- glm(left~., family="binomial", data=train)

#disable scientific notation for model summary
options(scipen=999)

#view model summary
model
```

```
##
## Call:  glm(formula = left ~ ., family = "binomial", data = train)
##
## Coefficients:
##            (Intercept)      satisfaction_level         last_evaluation
##              -0.645877               -4.560496               -0.019529
##         number_project   average_monthly_hours                  tenure
```

```
##              -0.473278               0.003766               1.068130
##          work_accident   promotion_last_5years                 salary
##              -1.488957              -1.232367              -0.537264
##  department_accounting          department_hr          department_IT
##              -0.052130               0.011768              -0.105508
##  department_management    department_marketing  department_product_mng
##               0.012644               0.078349              -0.274936
##       department_RandD        department_sales      department_support
##              -0.373843               0.112025               0.026937
##   department_technical
##                     NA
##
## Degrees of Freedom: 8375 Total (i.e. Null);  8358 Residual
## Null Deviance:       7672
## Residual Deviance: 5376  AIC: 5412
```

```
# r2
r2 <- pR2(model)["McFadden"]
```

```
## fitting null model for pseudo-r2
```

```
r2
```

```
##  McFadden
## 0.2993338
```

```
# calculate probability of default for each individual in test dataset
predicted <- predict(model, test, type="response")

predicted_test <- ifelse(predicted > 0.50, 1,0)

# convert it into a table
table_predicted_test <- table(Predicted = predicted_test, Actual = test$left)
table_predicted_test
```

```
##          Actual
## Predicted    0    1
##         0 2202  332
##         1  142  115
```

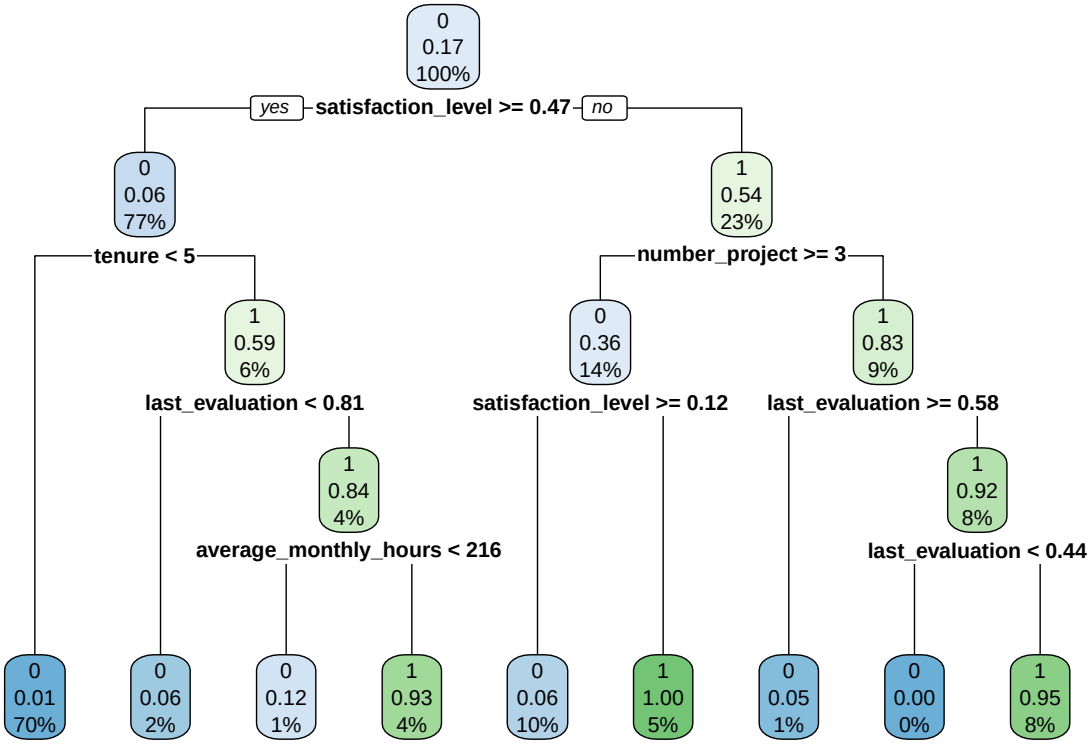
```
# cm for logistic regression
confusionMatrix(table_predicted_test, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted    0    1
##         0 2202  332
##         1  142  115
##
##                Accuracy : 0.8302
```

```
##                  95% CI : (0.8157, 0.8439)
##     No Information Rate : 0.8398
##     P-Value [Acc > NIR] : 0.9213
##
##                   Kappa : 0.2375
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.9394
##             Specificity : 0.2573
##          Pos Pred Value : 0.8690
##          Neg Pred Value : 0.4475
##               Precision : 0.8690
##                  Recall : 0.9394
##                      F1 : 0.9028
##              Prevalence : 0.8398
##          Detection Rate : 0.7890
##    Detection Prevalence : 0.9079
##       Balanced Accuracy : 0.5983
##
##        'Positive' Class : 0
##
```

decision tree model

```
# initiate decision tree model
fit <- rpart(left~., data = train, method = 'class')
rpart.plot(fit, extra = 106)
```




```
# compute probility for tree model
tree_prob <-predict(fit, test, type = 'prob')


# compute AUC for tree model
tree_auc <- auc(test$left, tree_prob[,2])
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
tree_auc
```

```
## Area under the curve: 0.9772
```

```
# predict tree model using test dataset
tree_predict <- predict(fit,test,type='class')

# cm for tree model
tree_cm <- confusionMatrix(as.factor(test$left),tree_predict, mode = "everything")
tree_cm
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    0    1
##          0 2325   19
##          1   37  410
##
##                Accuracy : 0.9799
##                  95% CI : (0.974, 0.9848)
##     No Information Rate : 0.8463
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.9242
##
##  Mcnemar's Test P-Value : 0.0231
##
##             Sensitivity : 0.9843
##             Specificity : 0.9557
##          Pos Pred Value : 0.9919
##          Neg Pred Value : 0.9172
##               Precision : 0.9919
##                  Recall : 0.9843
##                      F1 : 0.9881
##              Prevalence : 0.8463
##          Detection Rate : 0.8330
##    Detection Prevalence : 0.8398
##       Balanced Accuracy : 0.9700
##
##        'Positive' Class : 0
##
```

```
# improve the model by tuning it
control <- rpart.control(xval=4, minsplit = 2,
                         minbucket = round(6 / 3),
                         maxdepth = 4,
                         cp = 0)
tune_fit <- rpart(left~., data = train, method="class", control = control)

# compute auc for tunned tree model on test dataset
tree_prob2 <-predict(tune_fit, test, type = 'prob')
tree_auc2 <- auc(test$left, tree_prob2[,2])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
tree_auc2
```

```
## Area under the curve: 0.9776
```

```
# compute cm for tunned tree model
tree_predict2 <-predict(tune_fit, test, type = 'class')
tree_cm2 <- confusionMatrix(as.factor(test$left),tree_predict2, mode = "everything")
tree_cm2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2325   19
##          1   34  413
##
##                Accuracy : 0.981
##                  95% CI : (0.9752, 0.9857)
##     No Information Rate : 0.8452
##     P-Value [Acc > NIR] : < 0.0000000000000002
##
##                   Kappa : 0.9284
##
##  Mcnemar's Test P-Value : 0.05447
##
##             Sensitivity : 0.9856
##             Specificity : 0.9560
##          Pos Pred Value : 0.9919
##          Neg Pred Value : 0.9239
##               Precision : 0.9919
##                  Recall : 0.9856
##                      F1 : 0.9887
##              Prevalence : 0.8452
##          Detection Rate : 0.8330
##    Detection Prevalence : 0.8398
##       Balanced Accuracy : 0.9708
##
##        'Positive' Class : 0
##
```

random forest model

```
## Set seed for reproducibility
set.seed(42)

## Split the data so that we use 75% of it for training
train_index <- createDataPartition(y=df_logreg$left, p=0.75, list=FALSE)
repeat_cv <- trainControl(method='repeatedcv', number=4,repeats=4,classProbs=T)

## Subset the data
training_set <- df_logreg[train_index, ]
testing_set <- df_logreg[-train_index, ]

training_set$left <- as.factor(training_set$left)
training_set$left <- ifelse(training_set$left=="1","yes","no")

## Train a random forest model
forest <- train(left~.,
  data=training_set,
  method='rf',
```

```
  trControl=repeat_cv,
  metric='AUC')
```

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

```
## Print out the details about the model
forest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 1.53%
## Confusion matrix:
##       no  yes class.error
## no  6929   12 0.001728858
## yes  116 1319 0.080836237
```

```
# compute roc score
forest_prob <- predict(forest, testing_set, type = "prob")
head(forest_prob)
```

```
##       no   yes
## 6  0.000 1.000
## 14 0.000 1.000
## 17 0.016 0.984
## 20 0.000 1.000
## 21 0.000 1.000
## 28 0.002 0.998
```

```
forest_auc <- auc(testing_set$left,forest_prob[,2])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
forest_auc
```

```
## Area under the curve: 0.9824
```

```
## Generate predictions for cm
y_hats <- predict(object=forest,  newdata=testing_set[, -7])
head(y_hats)
```

```
## [1] yes yes yes yes yes yes
## Levels: no yes
```

```
testing_set$left <- ifelse(testing_set$left=="1","yes","no")
forest_cm <- confusionMatrix(as.factor(testing_set$left),y_hats,mode='everything')
forest_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  2341    3
##        yes   33  414
##
##                Accuracy : 0.9871
##                  95% CI : (0.9822, 0.991)
##     No Information Rate : 0.8506
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.9507
##
##  Mcnemar's Test P-Value : 0.000001343
##
##             Sensitivity : 0.9861
##             Specificity : 0.9928
##          Pos Pred Value : 0.9987
##          Neg Pred Value : 0.9262
##               Precision : 0.9987
##                  Recall : 0.9861
##                      F1 : 0.9924
##              Prevalence : 0.8506
##          Detection Rate : 0.8388
##    Detection Prevalence : 0.8398
##       Balanced Accuracy : 0.9895
##
##        'Positive' Class : no
##
```

Feature Engineering

```
# create new feature (overworked)
df3 <- df_enco
df3 <- df3 %>% mutate(overworked = df3$average_monthly_hours)
df3$overworked <- ifelse(df3$overworked>175,1,0)

# drop unuse columns
df3$average_monthly_hours <- NULL
df3$satisfaction_level<- NULL
```

```
# round 2 tree base model
set.seed(42)

indexset2 <- createDataPartition(df3$left,p = 0.75,list = F)
```

```
train2 <- df3[indexset2,]
test2 <- df3[-indexset2,]


tune_fit2 <- rpart(left~., data = train2, method="class", control = control)

tree_prob3 <-predict(tune_fit2, test2, type = 'prob')
tree_auc3 <- auc(test2$left, tree_prob3[,2])
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
tree_auc3
```

```
## Area under the curve: 0.9535
```

```
tree_predict3 <-predict(tune_fit2, test2, type = 'class')
cm_tree3 <- confusionMatrix(as.factor(test2$left),tree_predict3, mode = "everything")
cm_tree3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2403  106
##          1   81  407
##
##                Accuracy : 0.9376
##                  95% CI : (0.9283, 0.946)
##     No Information Rate : 0.8288
##     P-Value [Acc > NIR] : < 0.0000000000000002
##
##                   Kappa : 0.7758
##
##  Mcnemar's Test P-Value : 0.07925
##
##             Sensitivity : 0.9674
##             Specificity : 0.7934
##          Pos Pred Value : 0.9578
##          Neg Pred Value : 0.8340
##               Precision : 0.9578
##                  Recall : 0.9674
##                      F1 : 0.9625
##              Prevalence : 0.8288
##          Detection Rate : 0.8018
##    Detection Prevalence : 0.8372
##       Balanced Accuracy : 0.8804
##
##        'Positive' Class : 0
##
```

```r
# random forest round 2
## Set seed for reproducibility
set.seed(42)

## Split the data so that we use 75% of it for training
train_index2 <- createDataPartition(y=df3$left, p=0.75, list=FALSE)
repeat_cv <- trainControl(method='repeatedcv', number=4,repeats=4,classProbs=T)

## Subset the data
training_set2 <- df3[train_index2, ]
testing_set2 <- df3[-train_index2, ]

training_set2$left <- as.factor(training_set2$left)
training_set2$left <- ifelse(training_set2$left=="1","yes","no")

## Train a random forest model
forest2 <- train(left~.,
  data=training_set2,
  method='rf',
  trControl=repeat_cv,
  metric='AUC')
```

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

```r
## Print out the details about the model
forest2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 3.87%
## Confusion matrix:
##       no  yes class.error
## no  7339  152  0.02029102
## yes  196 1307  0.13040585
```

```r
# compute roc score
forest_prob2 <- predict(forest2, testing_set2, type = "prob")
head(forest_prob2)
```

```
##       no   yes
## 6  0.012 0.988
## 10 0.000 1.000
## 12 0.000 1.000
## 17 0.000 1.000
## 20 0.046 0.954
## 23 0.054 0.946
```

```r
forest_auc2 <- auc(testing_set2$left,forest_prob2[,2])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
forest_auc2
```

```
## Area under the curve: 0.9654
```

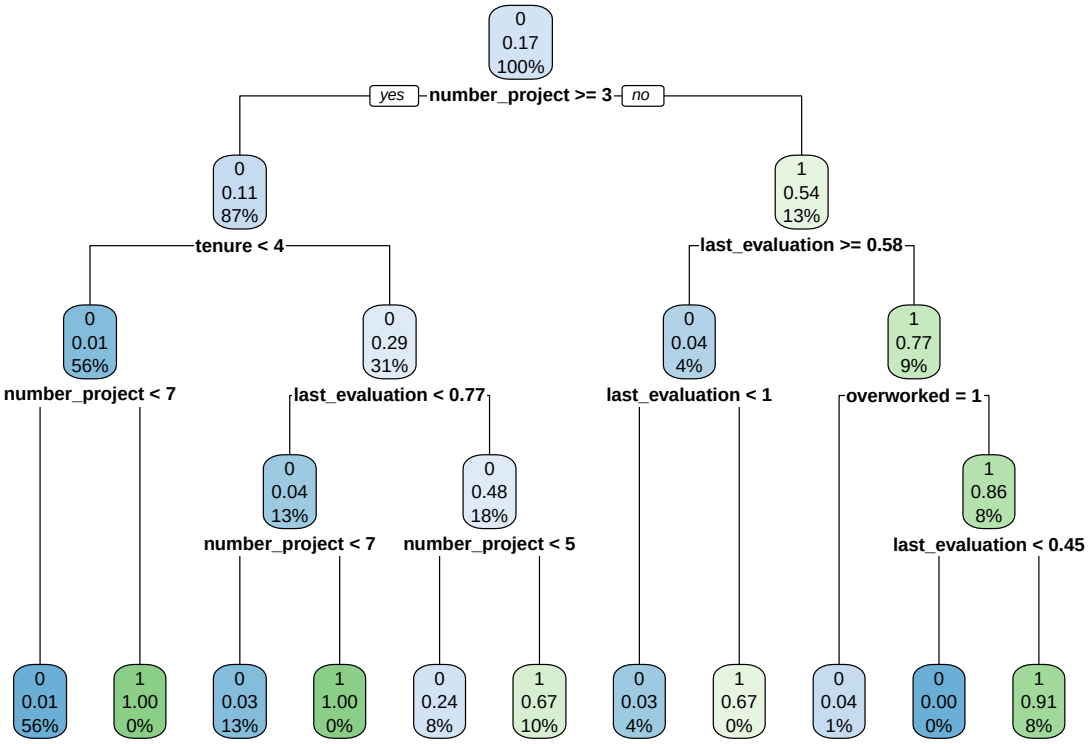
```r
## Generate predictions for cm
y_hats2 <- predict(object=forest2,  newdata=testing_set2[, -5])
head(y_hats2)
```

```
## [1] yes yes yes yes yes yes
## Levels: no yes
```

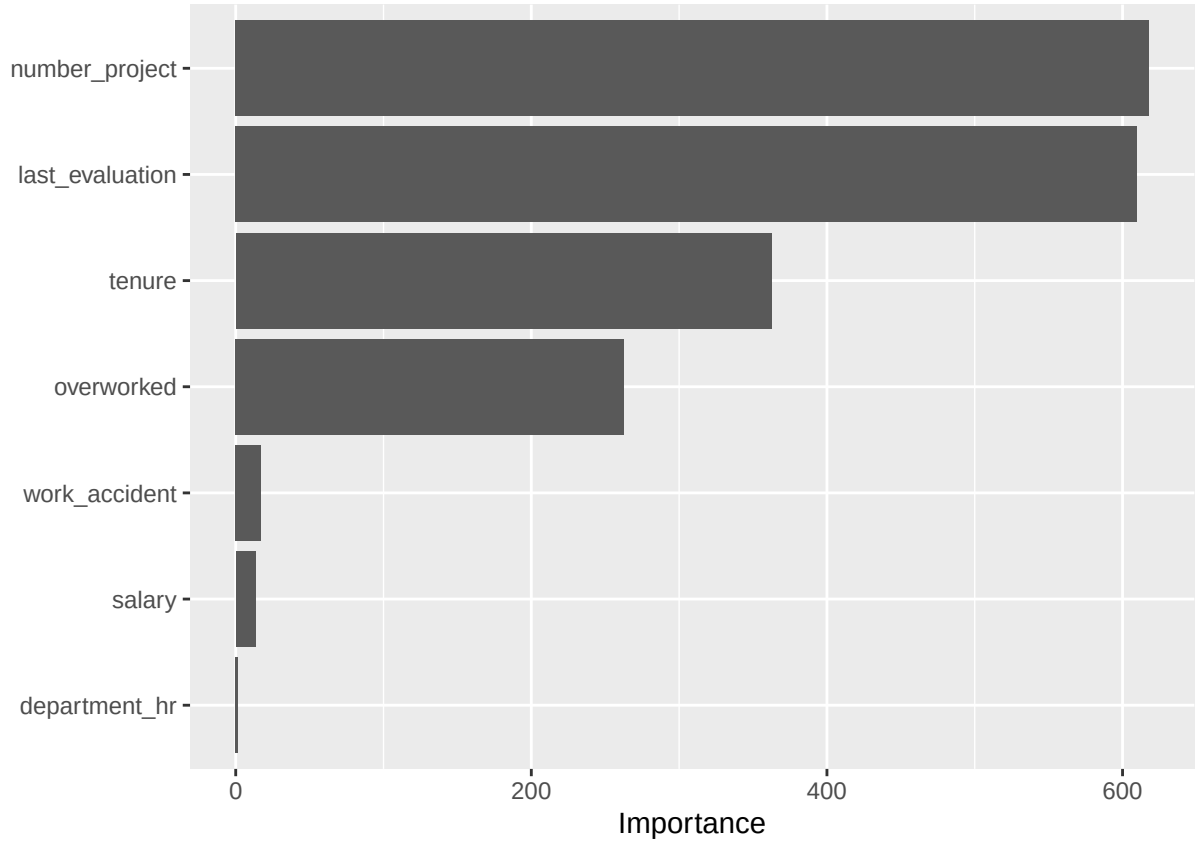
```r
testing_set2$left <- ifelse(testing_set2$left=="1","yes","no")
forest_cm2 <- confusionMatrix(as.factor(testing_set2$left),y_hats2,mode='everything')
forest_cm2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  2472   37
##        yes   53  435
##
##                Accuracy : 0.97
##                  95% CI : (0.9632, 0.9758)
##     No Information Rate : 0.8425
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.8884
##
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.9790
##             Specificity : 0.9216
##          Pos Pred Value : 0.9853
##          Neg Pred Value : 0.8914
##               Precision : 0.9853
##                  Recall : 0.9790
##                      F1 : 0.9821
##              Prevalence : 0.8425
##          Detection Rate : 0.8248
##    Detection Prevalence : 0.8372
##       Balanced Accuracy : 0.9503
##
##        'Positive' Class : no
##
```

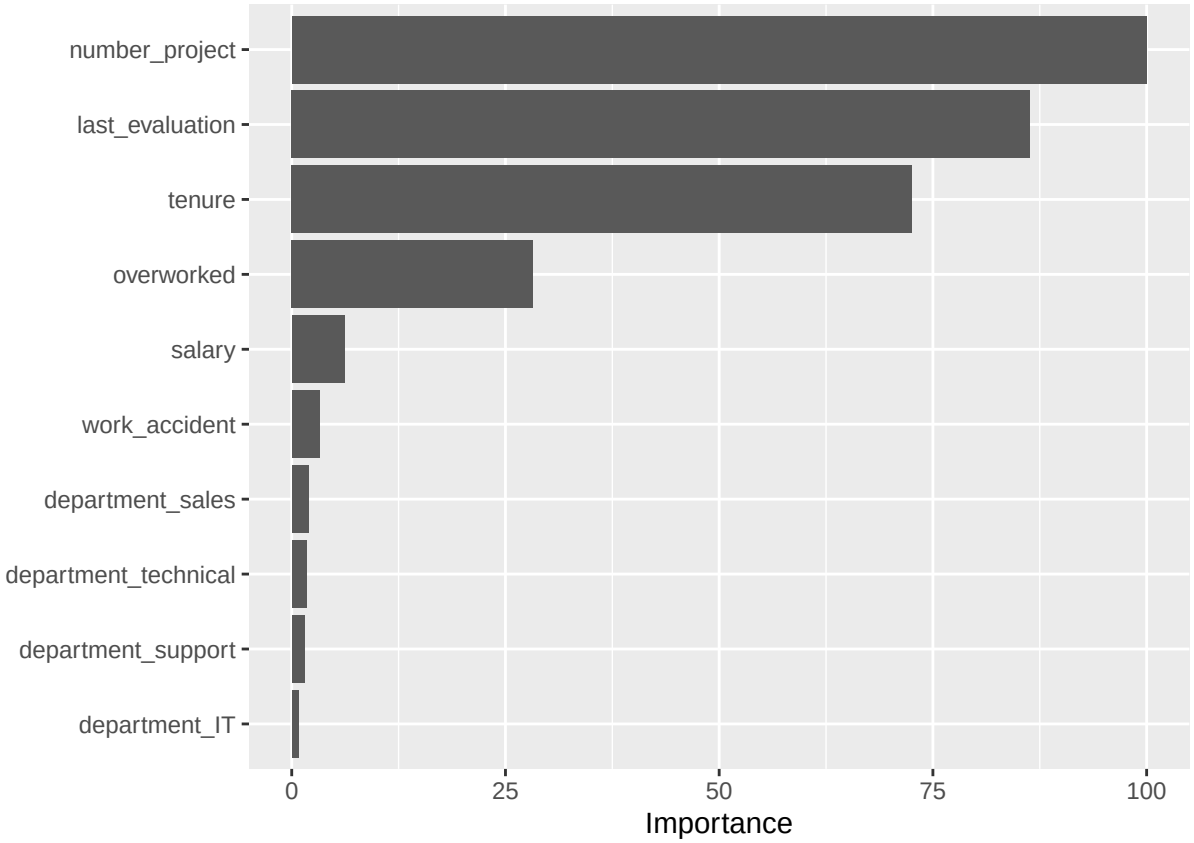
```
# plot tree model
rpart.plot(tune_fit2, extra = 106)
```




```
tree2_vip <- vip(tune_fit2)
tree2_vip
```




```
forest2_vip <- vip(forest2)
forest2_vip
```



summary

Logistic Regression

The logistic regression model achieved precision of 87%, recall of 94%, f1-score of 90% (all weighted averages), and accuracy of 83%, on the test set.

Tree-based Machine Learning

After conducting feature engineering, the decision tree model achieved AUC of 95.4%, precision of 95.8%, recall of 96.7%, f1-score of 96.3%, and accuracy of 93.7%, on the test set. the random forest model achieved AUC of 96.5%, precision of 98.5%, recall of 97.9%, f1-score of 98.2%, and accuracy of 97.0%, on the test set.The random forest modestly outperformed the decision tree model.

Conclusion, Recommendations, Next Steps

The models and the feature importances extracted from the models confirm that employees at the company are overworked.

To retain employees, the following recommendations could be presented to the stakeholders:

- Cap the number of projects that employees can work on.
- Consider promoting employees who have been with the company for atleast four years, or conduct further investigation about why four-year tenured employees are so dissatisfied.

- Either reward employees for working longer hours, or don't require them to do so.
- If employees aren't familiar with the company's overtime pay policies, inform them about this. If the expectations around workload and time off aren't explicit, make them clear.
- Hold company-wide and within-team discussions to understand and address the company work culture, across the board and in specific contexts.
- High evaluation scores should not be reserved for employees who work 200+ hours per month. Consider a proportionate scale for rewarding employees who contribute more/put in more effort.

Next Steps

It may be justified to still have some concern about data leakage. It could be prudent to consider how predictions change when `last_evaluation` is removed from the data. It's possible that evaluations aren't performed very frequently, in which case it would be useful to be able to predict employee retention without this feature. It's also possible that the evaluation score determines whether an employee leaves or stays, in which case it could be useful to pivot and try to predict performance score. The same could be said for satisfaction score.

For another project, you could try building a K-means model on this data and analyzing the clusters. This may yield valuable insight.