Predicting Employee Retention using HR Data Set using the Non parametric and parametric method

2023-04-21

- The goal of this project is to predict employee retention. The data set using is HR Data for Analytics. It can be found in Kaggle.com.
- In order to achieve this objective, first we need to explore and compare which model gives the accurate and better results.
- We will be examining parametric method such as linear regression, as well as non parametric method such as KNN, to determine the most effective approach.

load the required library

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

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

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

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
## Loaded glmnet 4.1-7
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
## corrplot 0.92 loaded
```

```
##
## Attaching package: 'corrplot'
```

```
## The following object is masked from 'package:pls':
##
##     corrplot
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.22-2
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## Loaded gbm 2.1.8.1
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:pls':
##
##     R2
```

```
##
## Attaching package: 'FNN'
```

```
## The following objects are masked from 'package:class':
##
##     knn, knn.cv
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

load the dataset

```
HREmployee=read.csv("~/Desktop/SFSU/personalMachineLearningProject/HR_comma_sep.csv",
header=TRUE)
```

quick look at the dataset

```
str(HREmployee)
```

```
## '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" ...
```

Convert the numeric binary variable into factor variable. (This will make the variable easier to understand and interpret)

```
HREmployee$Work_accident=factor(HREmployee$Work_accident,levels=c(0,1),labels=c("no w
ork accident","work accident"))
HREmployee$left=factor(HREmployee$left,levels=c(0,1),labels=c("stay with the company"
,"left the company"))
HREmployee$promotion_last_5years=factor(HREmployee$promotion_last_5years,levels=c(0,1
),labels=c("no promotion","promotion"))
```

Change the column name sales into department in R. (This will make the variable easier to understand and interpret)

```
names(HREmployee)[names(HREmployee)=="sales"]="department"
```

Perform feature selection using randomForest

To identify the most important predictors that contribute to a given outcome variable).

By selecting only the most relevant predictors, you can improve the accuracy and interpretability of predictive models.

```
set.seed(456)
rf <- randomForest(time_spend_company~ ., data = HREmployee,ntree=500, importance = T
RUE)
importance <- varImp(rf)
selected_features_rf = rownames(importance)[importance$Overall > mean(importance$Over
all)]
selected_features <- c(selected_features_rf, "time_spend_company")
print(selected_features_rf)
```

```
## [1] "satisfaction_level"   "last_evaluation"      "number_project"
## [4] "average_montly_hours" "left"
```

```
HREmployee <- HREmployee%>%
  dplyr::select(one_of(selected_features))
```

From the features selection using random Forest, I found out that variables “satisfaction_level”, “last_evaluation”,“number_project”,“average_montly_hours”,“left” are selected features.

Check missing values

```
sapply(HREmployee,function(x) sum(is.na(x)))
```

```
##   satisfaction_level      last_evaluation       number_project
##                    0                    0                    0
## average_montly_hours                 left   time_spend_company
##                    0                    0                    0
```

No missing values detect from the data

Check null values

```
sum(sapply(HREmployee,is.null))
```

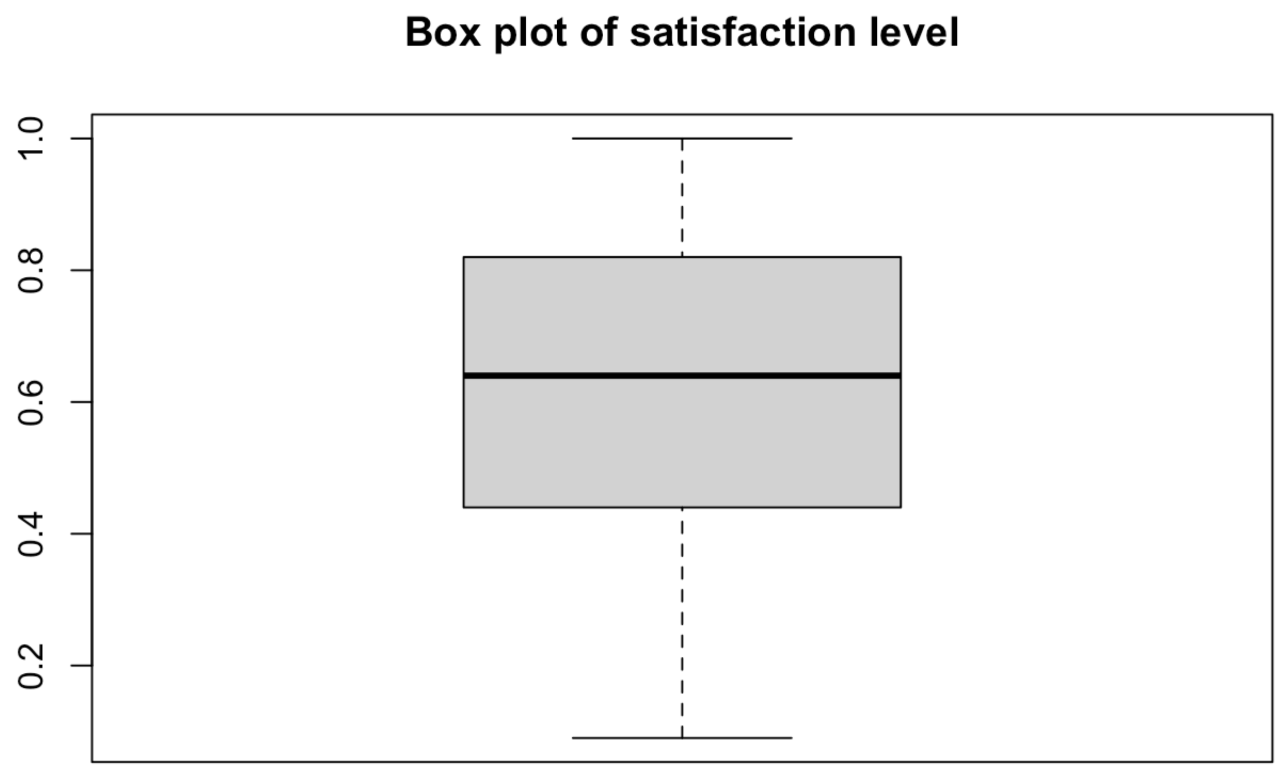
```
## [1] 0
```

no null values detect from the data

Check outlers of selected features using boxplot

boxplot of satisfaction level

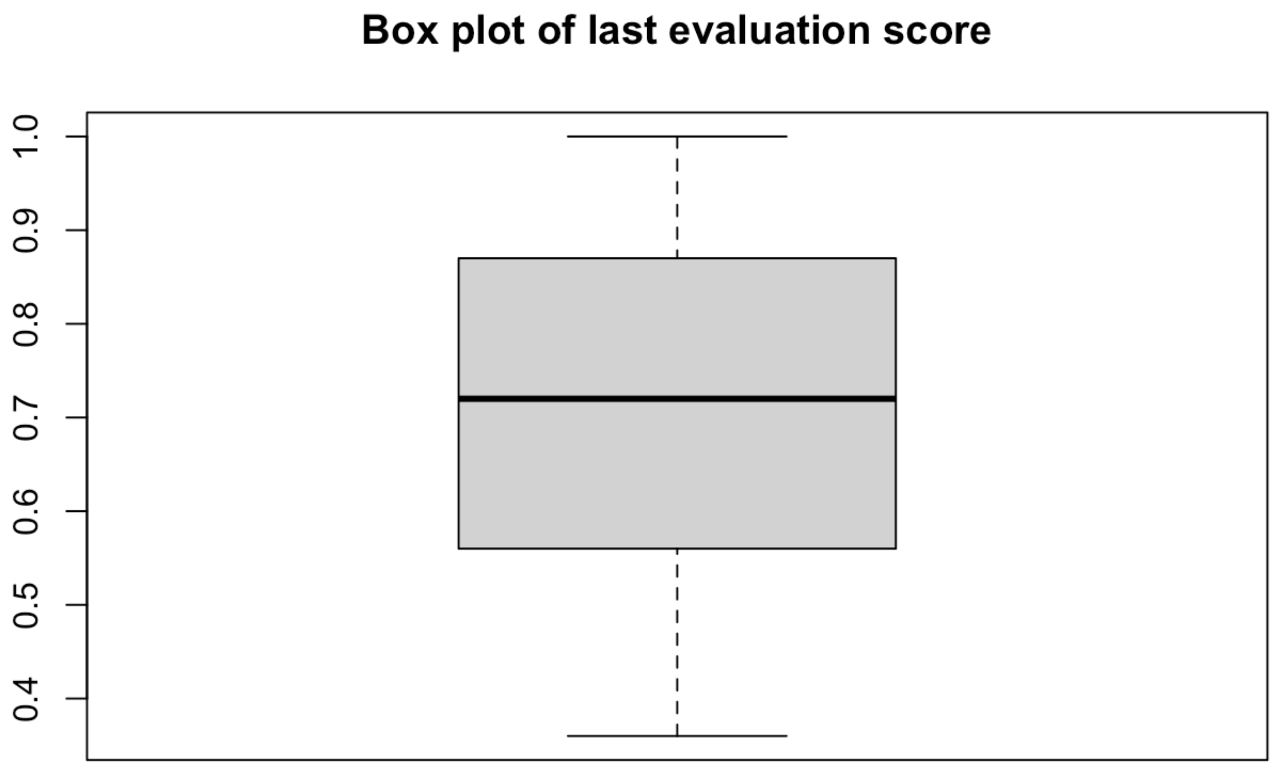
```
boxplot(HREmployee$satisfaction_level,main="Box plot of satisfaction level")
```



From the above graph, no outliers detect form the satisfaction level variables.

boxplot of last evaluation

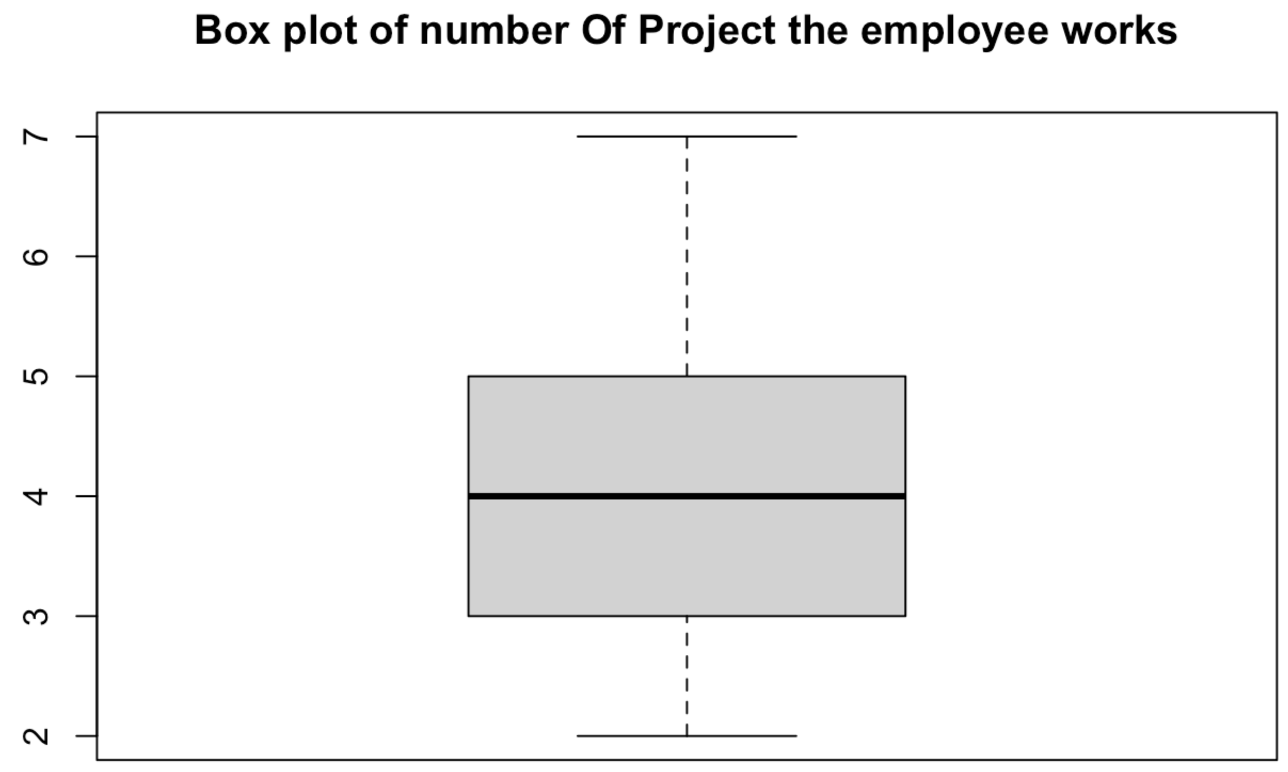
```
boxplot(HREmployee$last_evaluation,main="Box plot of last evaluation score")
```



From the above graph, no outliers detect form the last evaluation variables.

boxplot of number Of Project the employee works

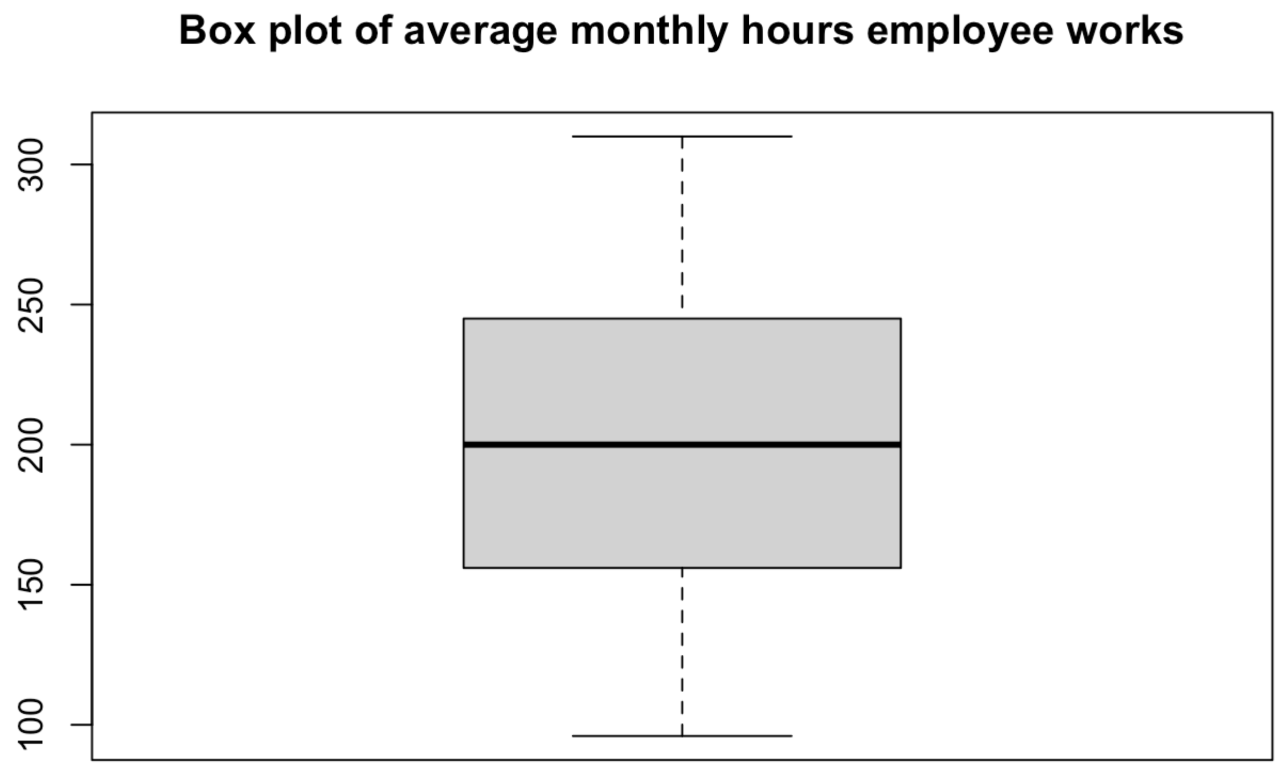
```
boxplot(HREmployee$number_project,main="Box plot of number Of Project the employee wo
rks")
```



From the above graph, no outliers detect form the number Of Project the employee works variables.

boxplot of average monthly hours employee works

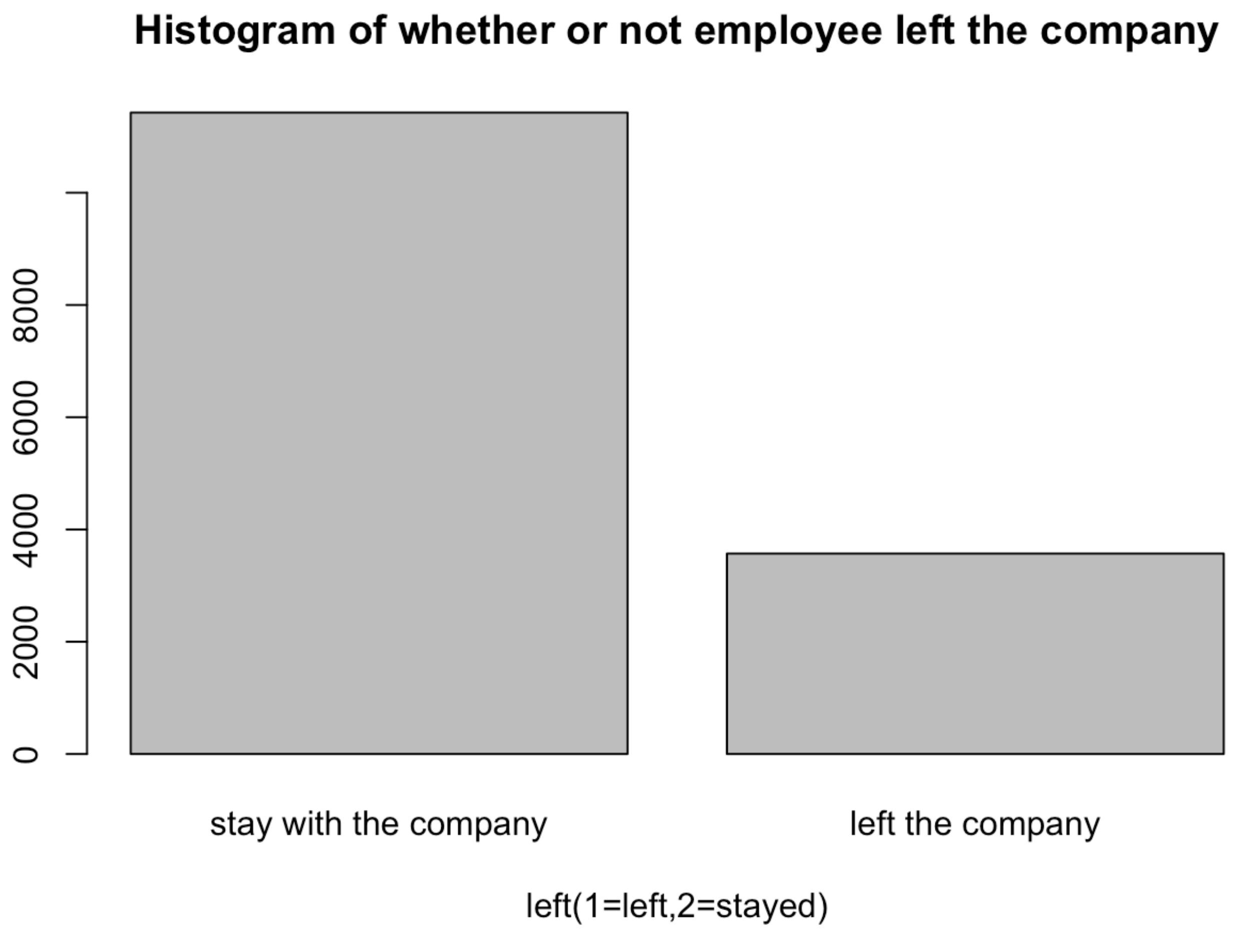
```
boxplot(HREmployee$average_montly_hours,main="Box plot of average monthly hours emplo
yee works")
```



From the above graph, no outliers detect form the average monthly hours employee works variables.

cannot use boxplot to check the “left” column since it is a non-numeric column. Need to use a barplot to visualize the frequncy of the variable

```
barplot(table(HREmployee$left),main="Histogram of whether or not employee left the co
mpany",xlab="left(1=left,2=stayed)")
```



From the above graph, we detect the imbalanced data set. It can cause problems when we tried to use “left” as our response variable.

Creating training and testing data

```
train = sample(dim(HREmployee)[1], dim(HREmployee)[1]*0.8)
test=-train
HREmployee.test=HREmployee[test,]
HREmployee.train=HREmployee[train,]
```

Check correlation between selected features() since multicollinearity can make the problem difficult to see the true

relationship between the predictor and response variables.

```
numericVariable=sapply(HREmployee,is.numeric)
numericData=HREmployee[,numericVariable]
predictorsNumericData=dplyr::select(numericData,-time_spend_company)
corMatrix=cor(predictorsNumericData)
corMatrix
```

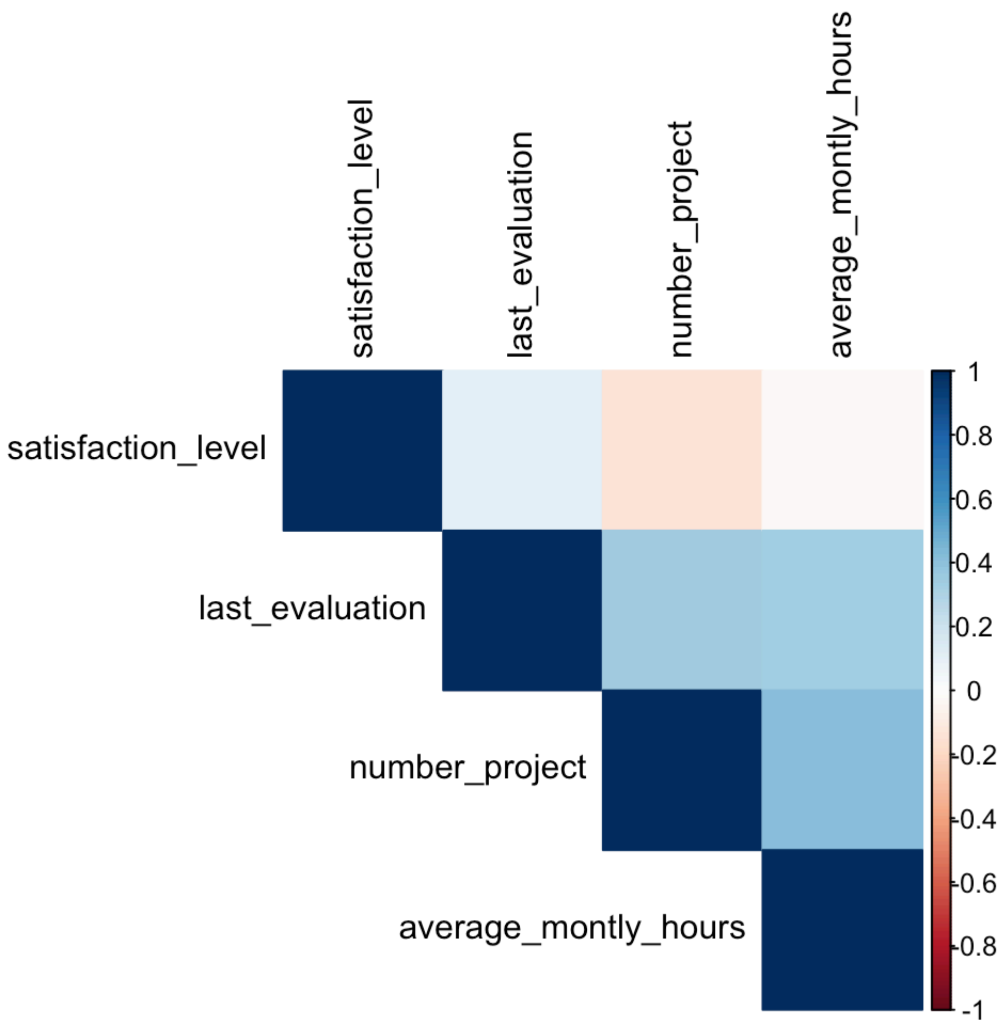
```
##                      satisfaction_level last_evaluation number_project
## satisfaction_level           1.00000000       0.1050212     -0.1429696
## last_evaluation              0.10502121       1.0000000      0.3493326
## number_project              -0.14296959       0.3493326      1.0000000
## average_montly_hours        -0.02004811       0.3397418      0.4172106
##                      average_montly_hours
## satisfaction_level            -0.02004811
## last_evaluation                0.33974180
## number_project                 0.41721063
## average_montly_hours           1.00000000
```

```
corrplot(corMatrix,type="upper",method="color",tl.col="black",t1.srt=45)
```

```
## Warning in text.default(pos.xlabel[, 1], pos.xlabel[, 2], newcolnames, srt =
## tl.srt, : "t1.srt" is not a graphical parameter
```

```
## Warning in text.default(pos.ylabel[, 1], pos.ylabel[, 2], newrownames, col =
## tl.col, : "t1.srt" is not a graphical parameter
```

```
## Warning in title(title, ...): "t1.srt" is not a graphical parameter
```



The correlation coefficients between all pairs of predictor variables in the model are less than 0.5. Thus, we don’t need to worry about multicollinearity in this problem.

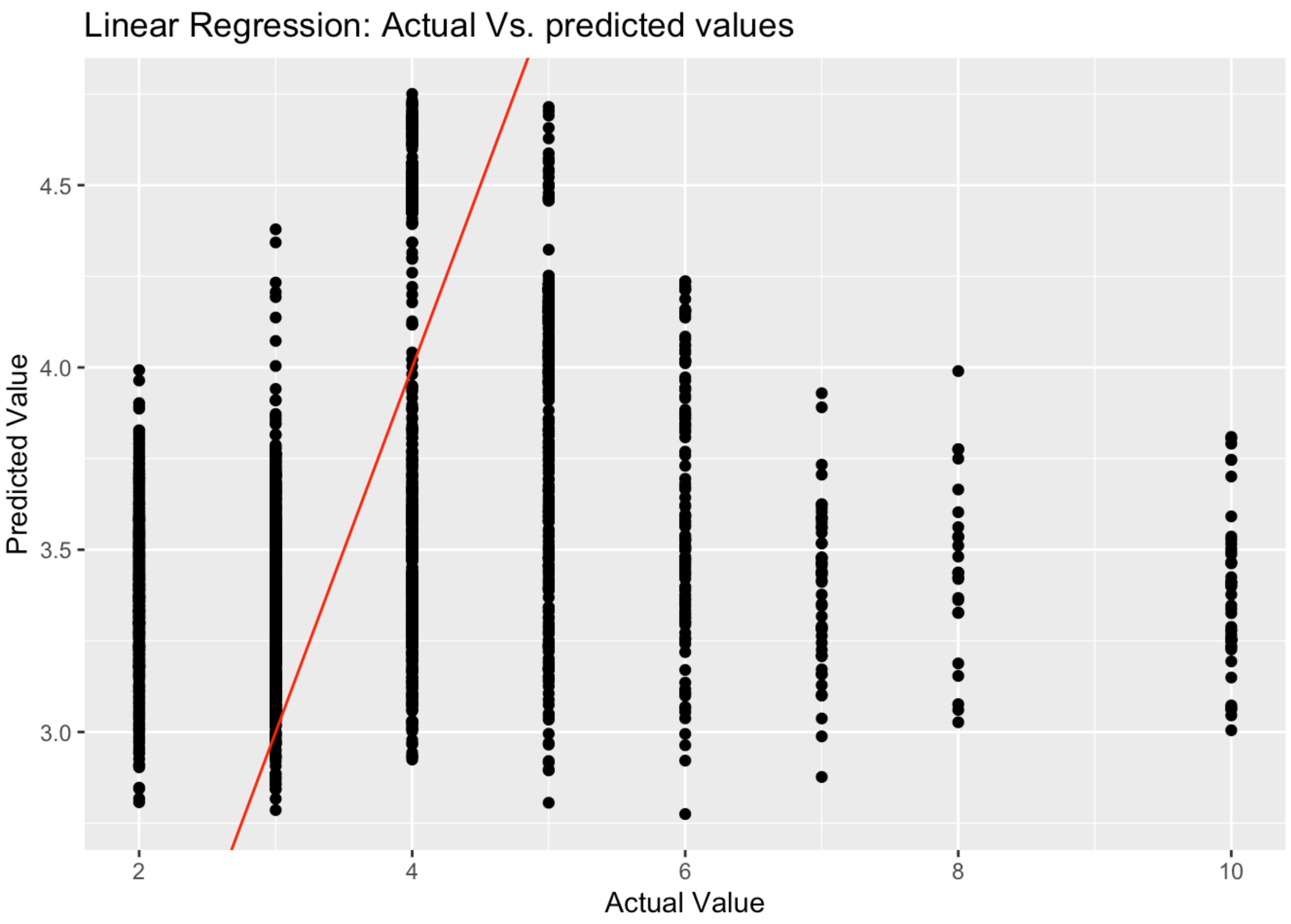
Fit the linear regression model with selected features using cross-validation.

Since this is a linear regression model, we will use MSE and RMSE. It will give an idea of how well the fitted model is predicting the response variable

```
fitControl <- trainControl(method = "cv", number = 10)
lm.fit <- train(time_spend_company ~ ., data = HREmployee.train , method = "glm", trC
ontrol = fitControl,metric="RMSE")
lm.pred=predict(lm.fit,HREmployee.test)
mse=mean((lm.pred-HREmployee.test$time_spend_company)^2) #test MSE
linearRegRMSE=sqrt(mean((lm.pred-HREmployee.test$time_spend_company)^2))
linearRegRMSE
```

```
## [1] 1.423136
```

```
ggplot(data=data.frame(actual=HREmployee.test$time_spend_company,predicted=lm.pred))+
  geom_point(aes(x=actual,y=predicted))+
  geom_abline(slope=1,intercept=0,color="red")+
  labs(x="Actual Value",y="Predicted Value",title="Linear Regression: Actual Vs. pred
icted values")
```



- MSE and RMSE tell us that the model is making accurate predictions. However, the scatter plot of predicted values vs. actual values shows that the points are far from diagonal line.
- This suggests that the linear regression model is not good for making predictions. The linear regression model is not capturing all the important patterns in the data.
- Therefore, I will use a different machine learning model to see which model will make accurate predictions.
- We will fit the model using K-Nearest neighbor. It can handle non linear relationships between predictor and response variables.

- However, in KNN model, the predictor variable,x, must be numeric.

converting non numeric predictors to numeric value since KNN requires to calculate the distance between test data(aka.real data) and all training points.

```
#first we need to convert non numeric predictors varabile into numeric variable by re
placing the factor levels with integers
#this is for training data
HREmployee.train$left=as.numeric(HREmployee.train$left)
#this is for testing data
HREmployee.test$left=as.numeric(HREmployee.test$left)
```

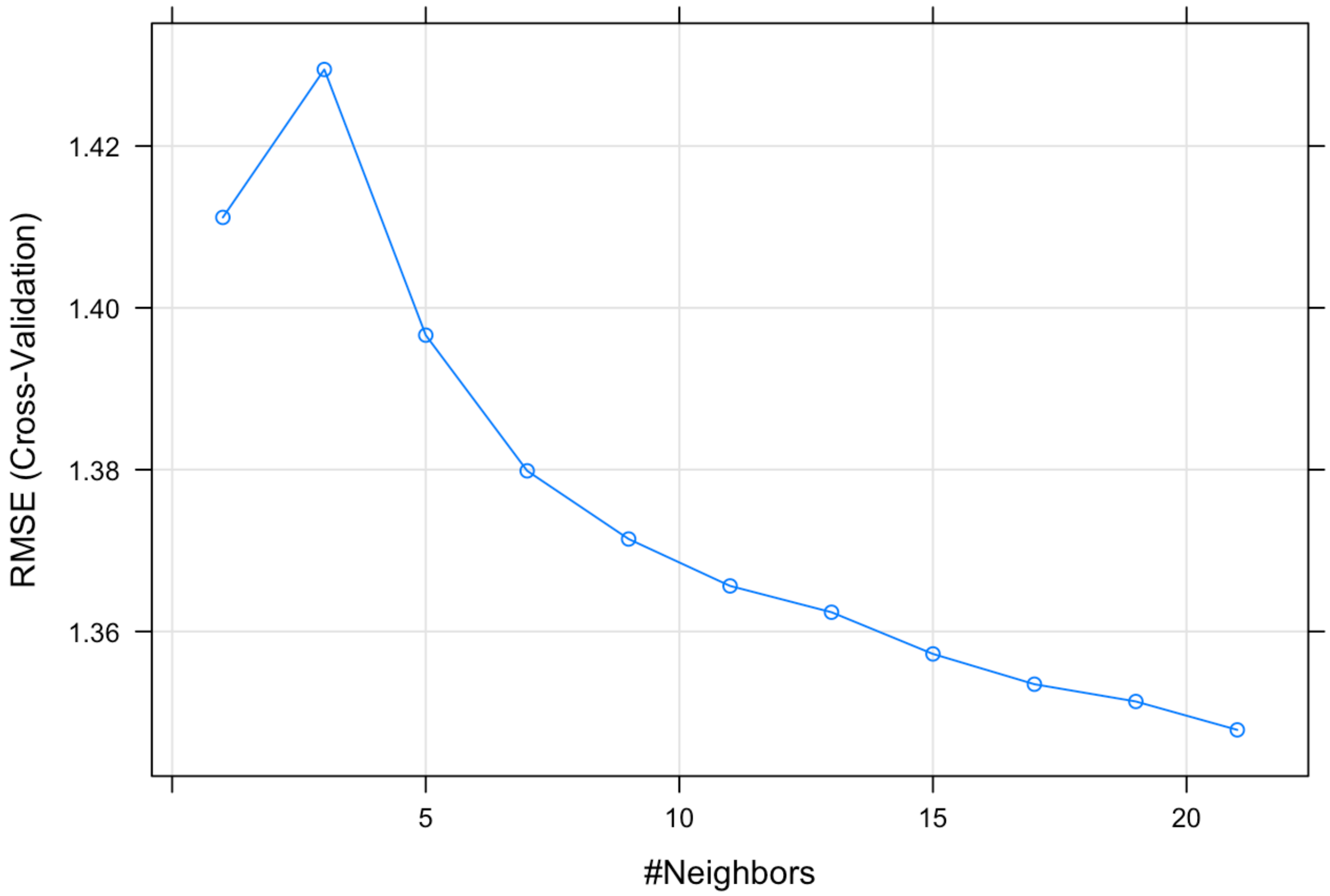
Chose the best Kvalue using the cross validation

```
kRange=seq(1,20,by=1)
fitControl2 <- trainControl(method = "cv", number = 10)
knnGrid=expand.grid(k=kRange)
knnFit=train(time_spend_company ~ ., data=HREmployee.train,method="knn",trControl=fit
Control2,tuneGrid=knnGrid,preProcess = c("center", "scale"), metric = "RMSE")
knnOptimal=knnFit$bestTune$k
print(paste("K value",knnOptimal,"will give the best balance between bias and varianc
e of the model to make predictions on a new data point. "))
```

```
## [1] "K value 20 will give the best balance between bias and variance of the model
to make predictions on a new data point. "
```

Use a graph to select the optimal value of k in KNN.

```
kValue=seq(1,22,by=2)
knnFit2=train(time_spend_company ~ ., data=HREmployee.train,method="knn",trControl=fitControl2,tuneGrid=data.frame(k=kValue),preProcess = c("center", "scale"), metric = "RMSE")
plot(knnFit2)
```



From the above graph, k=20 nearest neighbors is also good for predicting the new data point. It will give the best balance between bias and variance of the model.

fit the KNN model using the K value from cross validation

```
library(FNN)
library(MASS)
library(class)

knnRegressionFit=knn.reg(train=HREmployee.train[,-6],test=HREmployee.test[,-6],y=HREm
ployee.train[,6],k=knnOptimal)

HREmployee_knn_rmse <- sqrt(mean((HREmployee.test$time_spend_company - knnRegressionF
it$pred)^2))
print(paste0("RMSE: ", round(HREmployee_knn_rmse, 2)))
```

```
## [1] "RMSE: 1.4"
```

Let’s continue examining using the different model. This time we will be using decision trees machine learning model to see whether or not this model is the best fit for this model.

Decision trees.

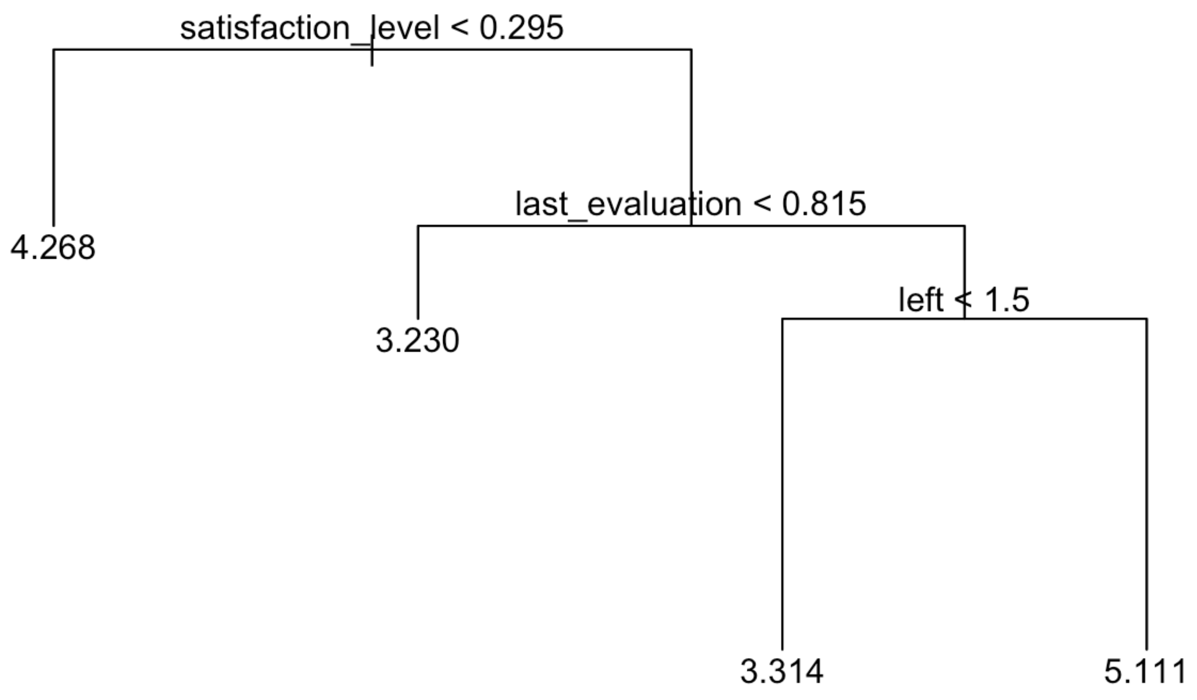
unlike linear regression, decision trees machine learning model doesn’t assume that there is a underlying distribution of the data. However, it has some disadvantages. For example, it can cause ovefitting if we don’t handle careful.

fit the decision trees model on training data

```
set.seed(1)
tree.HR=tree(time_spend_company~satisfaction_level+last_evaluation+number_project+ave
rage_montly_hours+left,data=HREmployee.train)
```

plots the decision tree model and #add text labels to the decision tree plot

```
plot(tree.HR)
text(tree.HR, pretty=0)
```



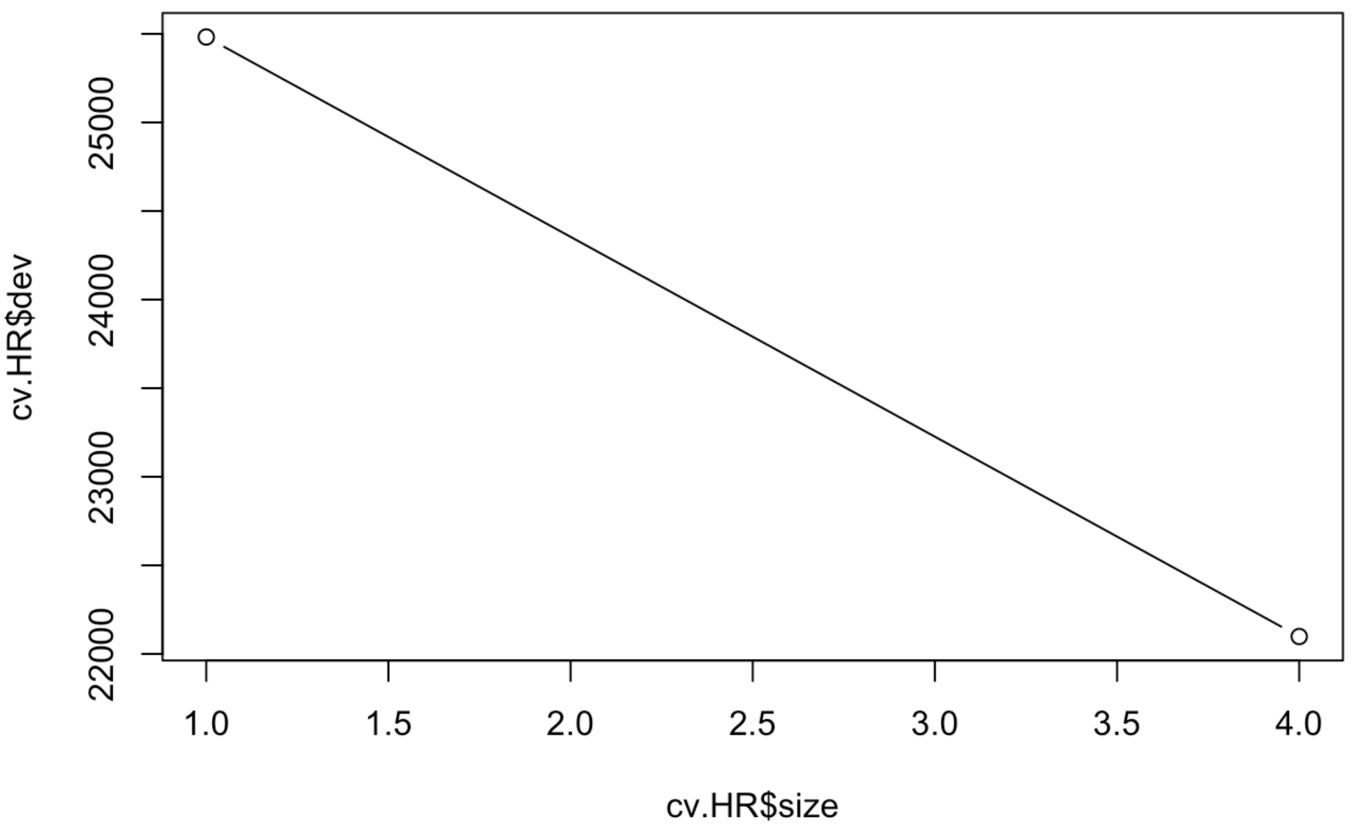
```
summary(tree.HR) #displays a summary of the decision tree model, including the number
of terminal nodes and overall error rate.
```

```
##
## Regression tree:
## tree(formula = time_spend_company ~ satisfaction_level + last_evaluation +
##     number_project + average_montly_hours + left, data = HREmployee.train)
## Variables actually used in tree construction:
## [1] "satisfaction_level" "last_evaluation"    "left"
## Number of terminal nodes:  4
## Residual mean deviance:  1.839 = 22060 / 12000
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.1110 -0.3143 -0.2305  0.0000 -0.1108  6.7700
```

performs cross validation on the decision

tree model. This function generates a sequence of the models with different numbers of terminal nodes, and computes the cross validation error rate for each model.

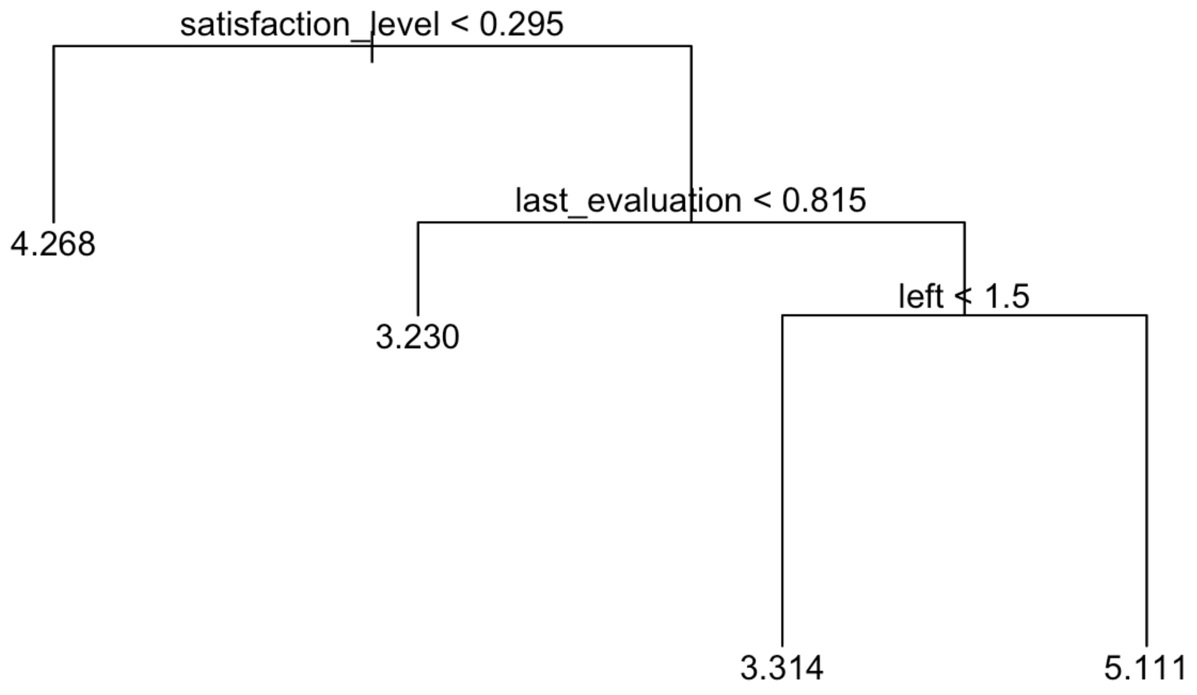
```
cv.HR=cv.tree(tree.HR)
plot(cv.HR$size,cv.HR$dev,type='b') #plot the cross validation error rate against the
number of terminal nodes
```



```
#choose terminal nodes 4 because it gives the lowest cross validation error rate in g
raph
```

prunes the decision tree model to the optimal number of terminal nodes.

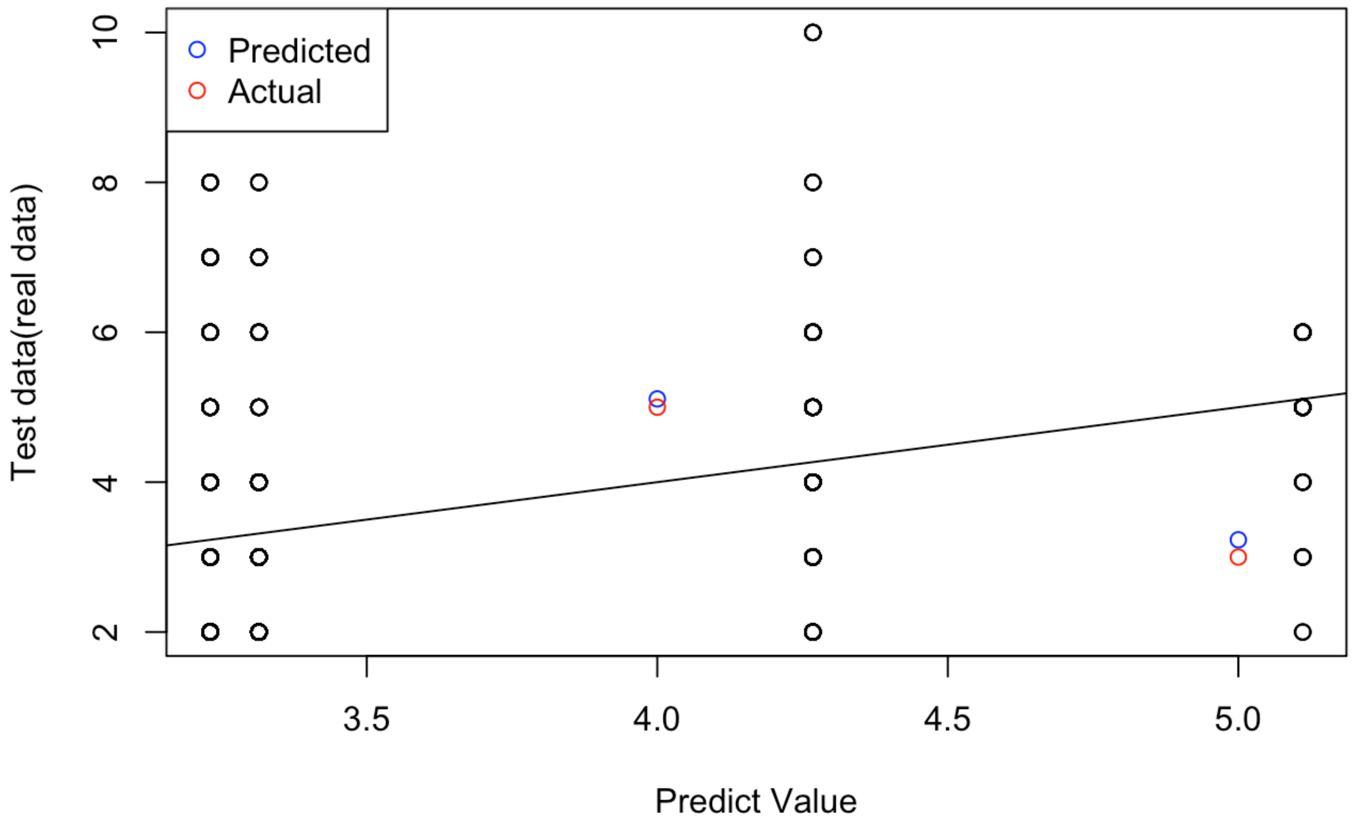
```
prune.HR=prune.tree(tree.HR,best=4) #prunes the decision tree model to the optimal nu
mber of terminal nodes. The best parameter is set to 4 to select the model with the l
owest cross validation error rate.
plot(prune.HR) #plots the pruned decision tree model
text(prune.HR,pretty=0) #adds text labels to the pruned decision tree plot
```



#

test to see the performance of the model on the testing data or new data

```
yhat=predict(tree.HR,newdata=HREmployee.test) #predict
plot(yhat,HREmployee.test$time_spend_company,xlab="Predict Value",ylab="Test data(rea
l data)")
points(yhat,col="blue")
points(HREmployee.test$time_spend_company,col="red")
legend("topleft", legend=c("Predicted", "Actual"), col=c("blue", "red"), pch=1)
abline(0,1)
```



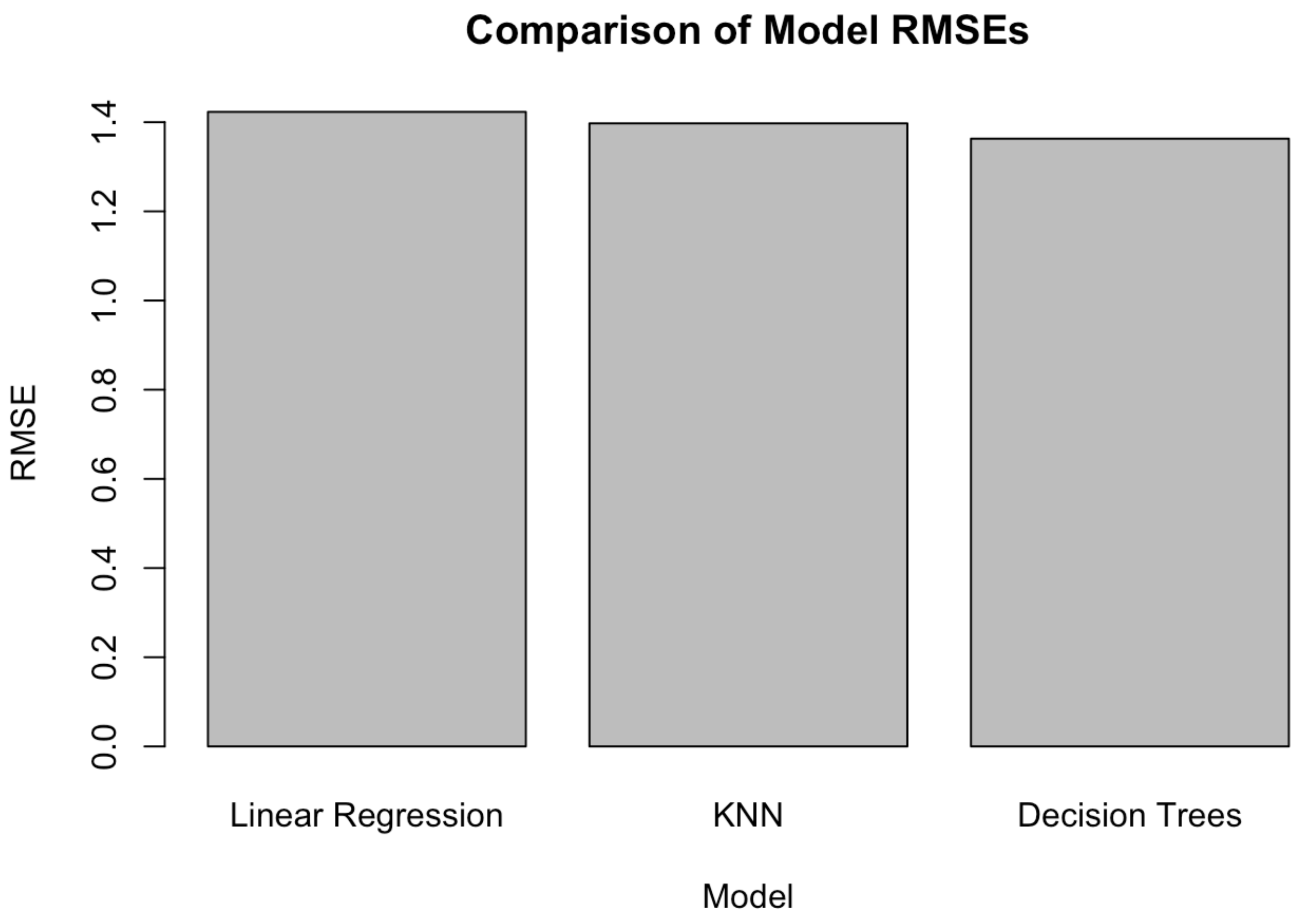
```
decisionTreesMSe=mean((yhat-HREmployee.test$time_spend_company)^2) #MSE
decisionTreesRMSE=sqrt(mean((yhat-HREmployee.test$time_spend_company)^2)) #RMSE
print(paste("RMSE value using decision trees method is",round(decisionTreesRMSE,2)))
```

```
## [1] "RMSE value using decision trees method is 1.36"
```

Graph to show RMSE values for linear

regression, KNN, and decision trees.

```
RMSE=c(linearRegRMSE,HREmployee_knn_rmse,decisionTreesRMSE)
barplot(RMSE,names.arg = c("Linear Regression", "KNN", "Decision Trees"), xlab = "Mod
el", ylab = "RMSE", main = "Comparison of Model RMSEs")
```



Based on the above graph, RMSE value from linear regression, KNN, and Decision trees are close to each other. It is hard to say which model is the best fit without farther examine using the different machine learning model.

In the future, I need to use different metrics to evaluate the performance of these three models. For example, we could use adjusted R-squared or AIC or BIC. In practice, many Data Scientist use multiple metrics to evaluate the performance of the mode.