CYO_Project

#1-Introduction

As I work with environmental issues, I have chosen a dataset in Kaggle to study the water quality of a river and I try a different question. The dataset was extracted from: River Water Quality EDA and Forecasting (kaggle.com), with is a river in Uckraine. Reference: https://www.kaggle.com/datasets/vbmokin/wq-southern-bug-river-0105202 Dataset formats: two csv files. There were called PB_All_2000_2021 and PB_stations. #Define the variables - PB_stations-> selected: id and the length (Distance from the mouth of the river, km). The name of the station was removed, as it was needed. - PB_All_2000_2021 Water Quality Parameters : O2 ,CL ,SO4, PO4, BSK5 ,Suspended, NO2, NH4 The main goal of the project: Develop some numeric Regression Models to calculate the target column (NH4).The question is can we substitute the weekly measurement of NH4 for this Regression Model?

##1.1-Loading Libraries

1.2-Data Preprocessing

###Data Loading

###Join, Selecting Columns

```
stations <- PB_stations%>%select(id,length)
All <-PB_All_2000_2021%>%mutate(date=str_replace_all(date,"[.]","-"))%>%mutate(date=dmy(date))%>%mutate
#Add the column: length
df <- left_join(All, stations, by = "id")
df_total <- df
#Filter: select stations(rows). Considering the measures realized, the similar stations are:14, 15 and
df <-df%>%filter(id==c(14,15,16))
# Remove Columns
df <- df %>% select(-one_of('id', 'date'))
```

#2-Analysis

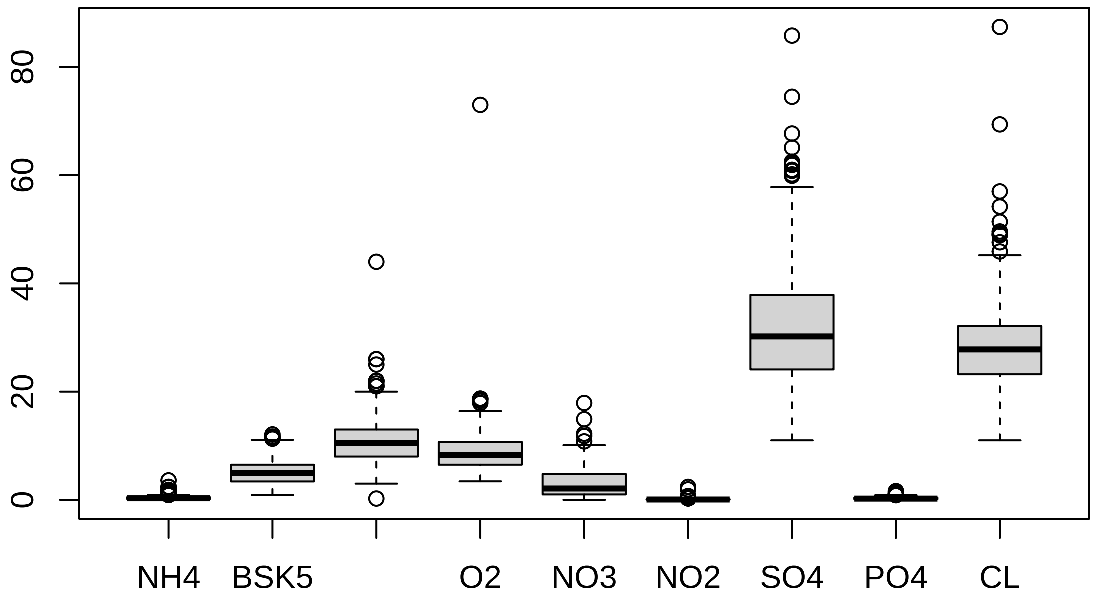
a methods/analysis section that explains the process and techniques used, including data cleaning, data exploration and visualization, insights gained, and your modeling approaches (you must use at least two different models or algorithms);

##2.1 Preprocessing - Handling Missing Data The option applied here was to remove the entire rows that contain any NA, with the function drop_na(). # 2.2 - Data Normalization The option applied here was to standardization mean centering and scaling in train function: preProcess = c("center","scale") Reference:https://www.rdocumentation.org/packages/caret/versions/6.0-92/topics/preProcess

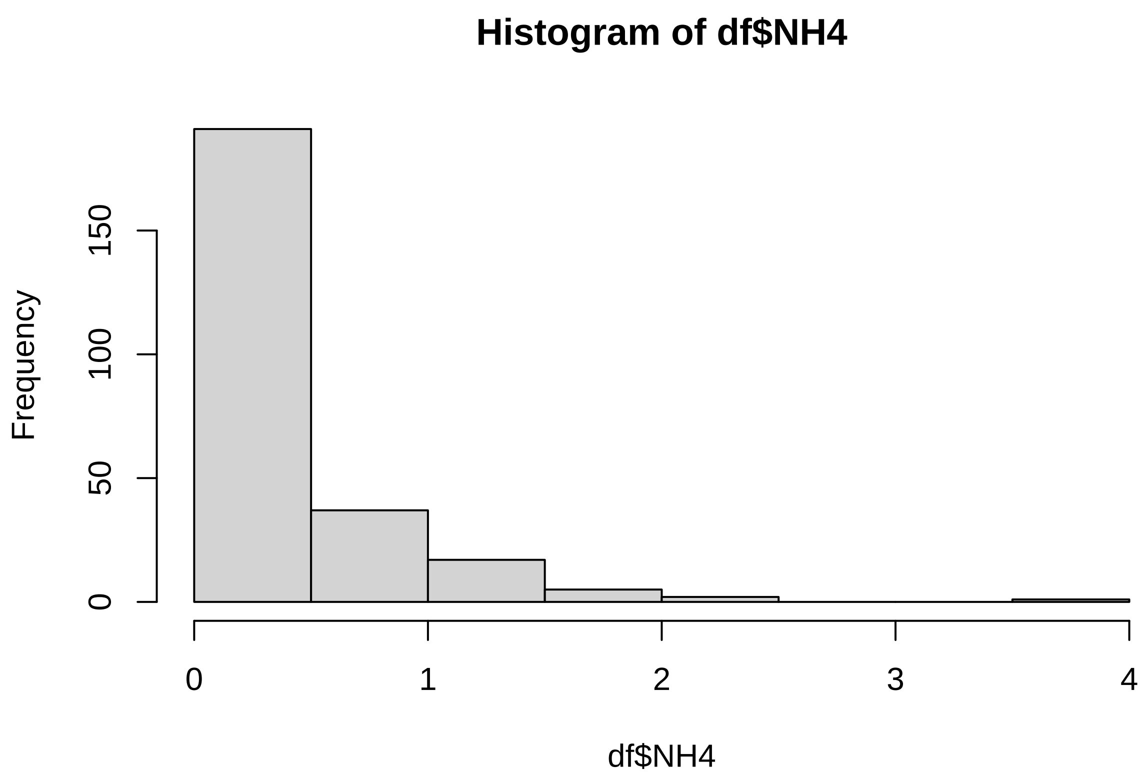
```
##Handling Missing Data -
df <- df %>% drop_na()
```

2.2 Dataviz - Exploratory Data Analysis (EDA)

```
# Box Plot of the water quality parameters
df %>% select(-one_of('length')) %>% boxplot(df)
```

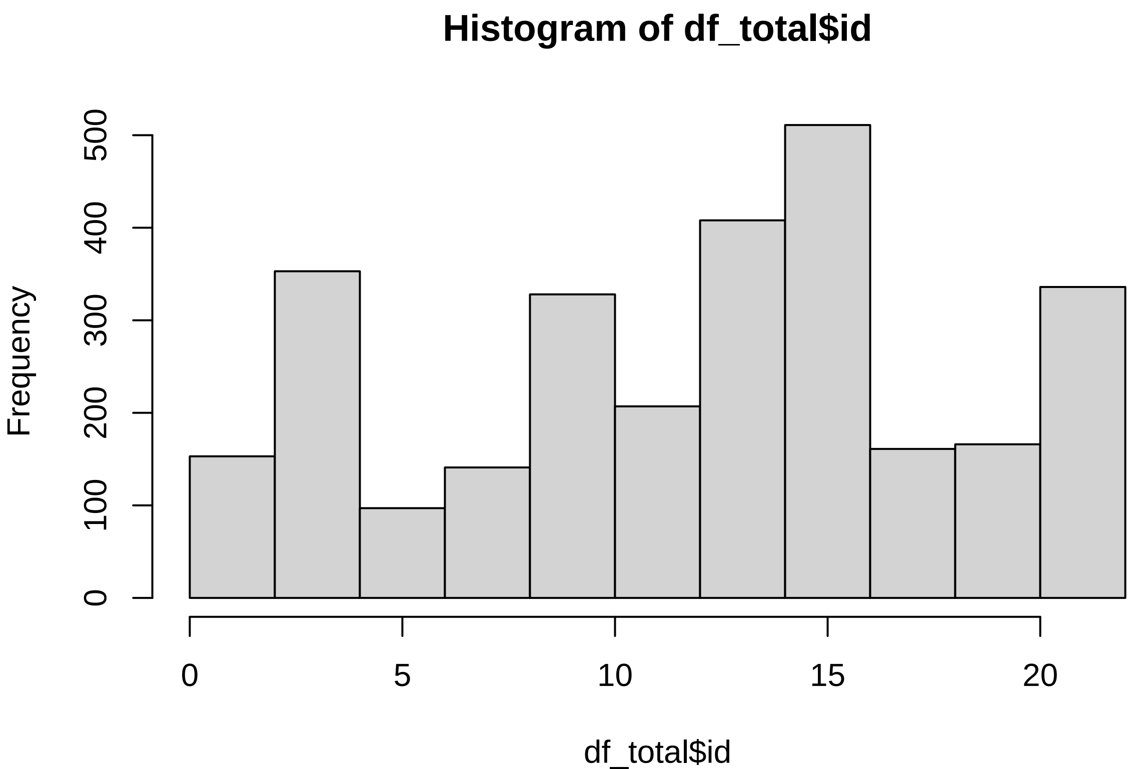



```r
# Histogram of the water Target
hist(df$NH4)
```




```r
#Ask if there is an near Zero variance
nzv <- nearZeroVar(df) #OK

#See all the stations
hist(df_total$id)
```



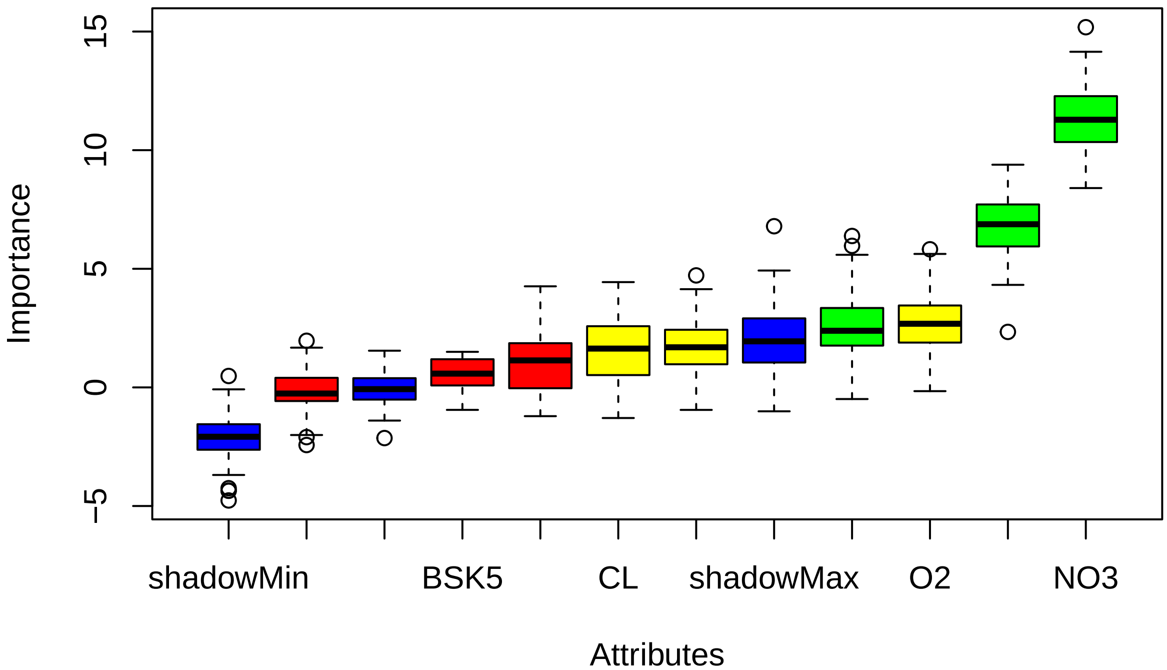
###

2.2.1 Unimportant variables

```
#Boruta function helps to identify unimportant variables
boruta_results <- Boruta(NH4~., df)
boruta_results
```

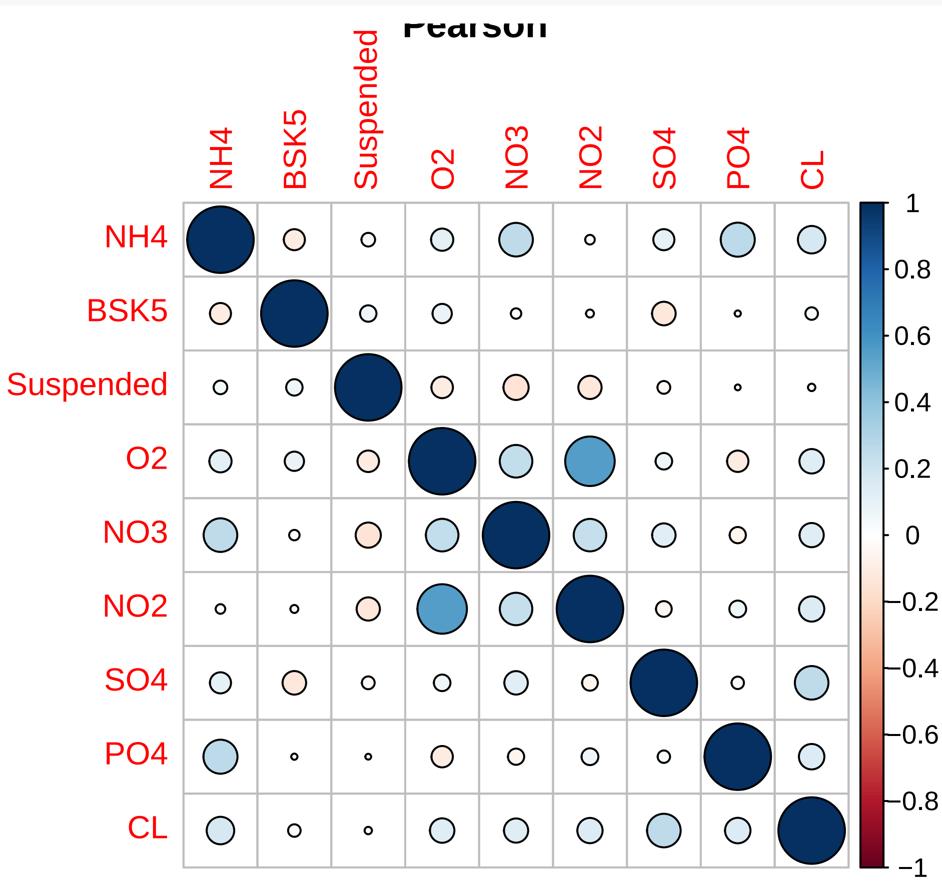
```
## Boruta performed 99 iterations in 2.254217 secs.
##  3 attributes confirmed important: NO2, NO3, PO4;
##  3 attributes confirmed unimportant: BSK5, length, Suspended;
##  3 tentative attributes left: CL, O2, SO4;
```

```
plot(boruta_results)
```

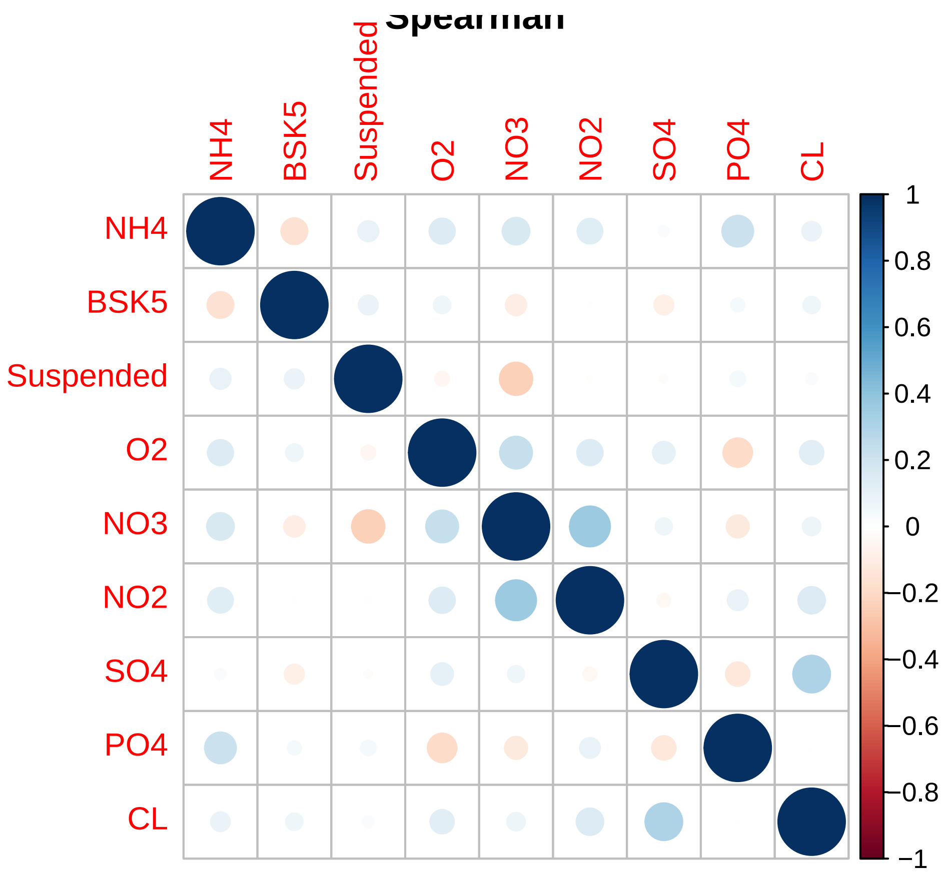


###
2.2.2 Correlograms (Pearson and Spearman)

```
CorPearson <-cor(df[1:9], method = "pearson")
corrplot(CorPearson, method="circle",title = "Pearson", outline = TRUE, addCoefasPercent = TRUE)
```



```r
CorSpearman <-cor(df[1:9], method = "spearman")
corrplot(CorSpearman, method="circle",title ="Spearman")
```



2.3 - Data Preparation to Machine Learning, Create Partition: train and test sets. - The train(70%) and the test(30%) sets were created to run the models. - Data Normalization or Standardization

```r
#Define the target

df <- df %>% rename(y=NH4)

# Remove Columns
#df <- df %>% select(-one_of('BSK5'))

# Split the dataset: train and test sets.
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(df$y, times = 1, p = 0.7, list = FALSE)
train_set <- df%>%slice(-test_index)
test_set <- df%>%slice(test_index)
```

#3-Results

a results section that presents the modeling results and discusses the model performance ##3.1 - method: Generalized Linear Model(glm)

```r
set.seed(1, sample.kind="Rounding")
```

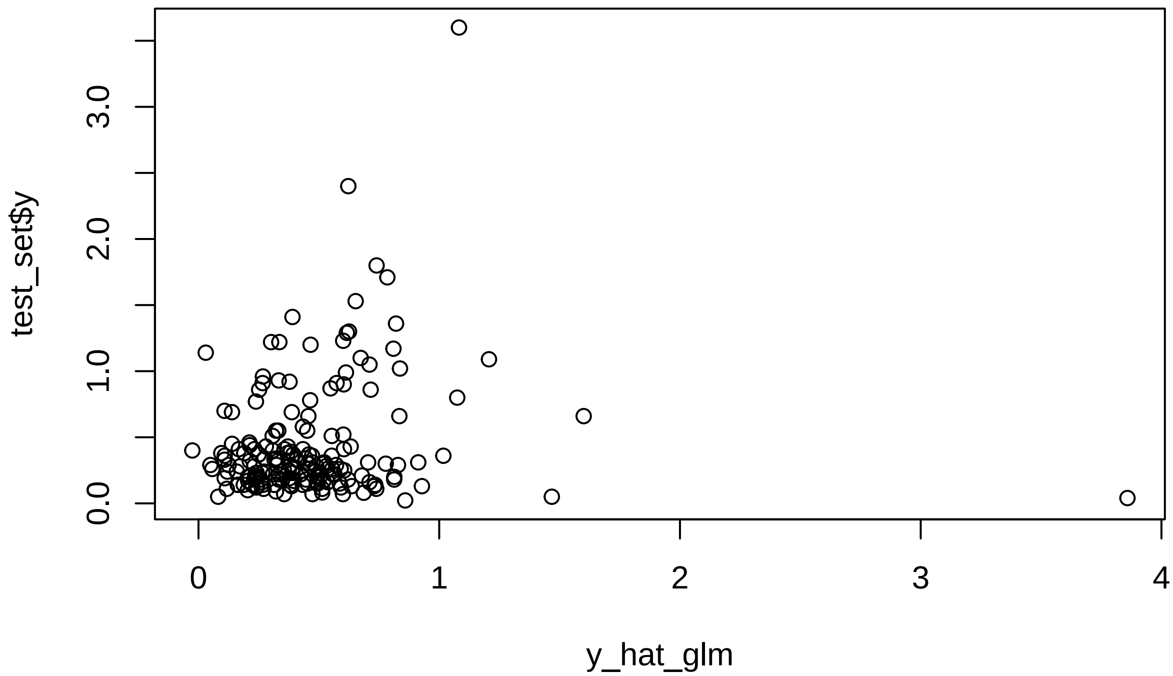
```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
#getModelInfo("glm")
#modelLookup("glm")

train_glm <- train(y~.,method="glm",data=train_set)
y_hat_glm <- predict(train_glm,test_set)
head(train_glm$results)
```

```
##   parameter      RMSE   Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1      none 0.4601786 0.08367184 0.3335552 0.07203233  0.0725752 0.04812494
```

```r
plot(y_hat_glm,test_set$y)
```



##3.2

- method: k-Nearest Neighbors(knn)

```r
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn <- train(y~.,method = "knn",trControl = control,data=train_set,preProc = c("center","scale"),

train_knn$results
```

```
##     k      RMSE  Rsquared       MAE    RMSESD RsquaredSD      MAESD
## 1   1 0.5139515 0.1291402 0.3762464 0.2107932  0.1477006 0.14615637
## 2   2 0.4510650 0.1887239 0.3191506 0.1919110  0.1481463 0.10515372
## 3   3 0.4346265 0.1541407 0.3138653 0.1791915  0.1496252 0.09913606
## 4   4 0.3907682 0.1886804 0.2877467 0.1906384  0.2271254 0.10477881
## 5   5 0.4015222 0.1625321 0.2912057 0.1853998  0.2445458 0.10520973
## 6   6 0.3898073 0.1798333 0.2783723 0.1906333  0.2522767 0.10294725
## 7   7 0.3960519 0.2026196 0.2832980 0.1891231  0.2800732 0.10713501
## 8   8 0.4008202 0.2226985 0.2857981 0.1823675  0.2644773 0.09886177
## 9   9 0.4006081 0.2251680 0.2848890 0.1862114  0.2775629 0.10243767
## 10 10 0.3921413 0.2325514 0.2779074 0.1765200  0.2718874 0.09490737
```

```
## 11 11 0.3890848 0.2401319 0.2765410 0.1747885  0.2400981 0.09654846
## 12 12 0.3899013 0.2557143 0.2762030 0.1745297  0.2273728 0.09069050
## 13 13 0.3904820 0.2453290 0.2794841 0.1773921  0.2118422 0.09504339
## 14 14 0.3872160 0.2820119 0.2756169 0.1731553  0.2490372 0.09478019
## 15 15 0.3872999 0.2719817 0.2719115 0.1774495  0.2270844 0.09543371
## 16 16 0.3871674 0.2740952 0.2739744 0.1774207  0.2186906 0.09223066
## 17 17 0.3879407 0.2571307 0.2732276 0.1794440  0.2033807 0.09145566
## 18 18 0.3814620 0.3060629 0.2694402 0.1745175  0.2077750 0.08898454
## 19 19 0.3798760 0.3208540 0.2677024 0.1754064  0.2442531 0.08859884
## 20 20 0.3795820 0.3232172 0.2665096 0.1775565  0.2517326 0.08859970
```

```r
y_hat_knn <- predict(train_knn, test_set, type = "raw")
```

##3.3 - method: Random Forest(rf)

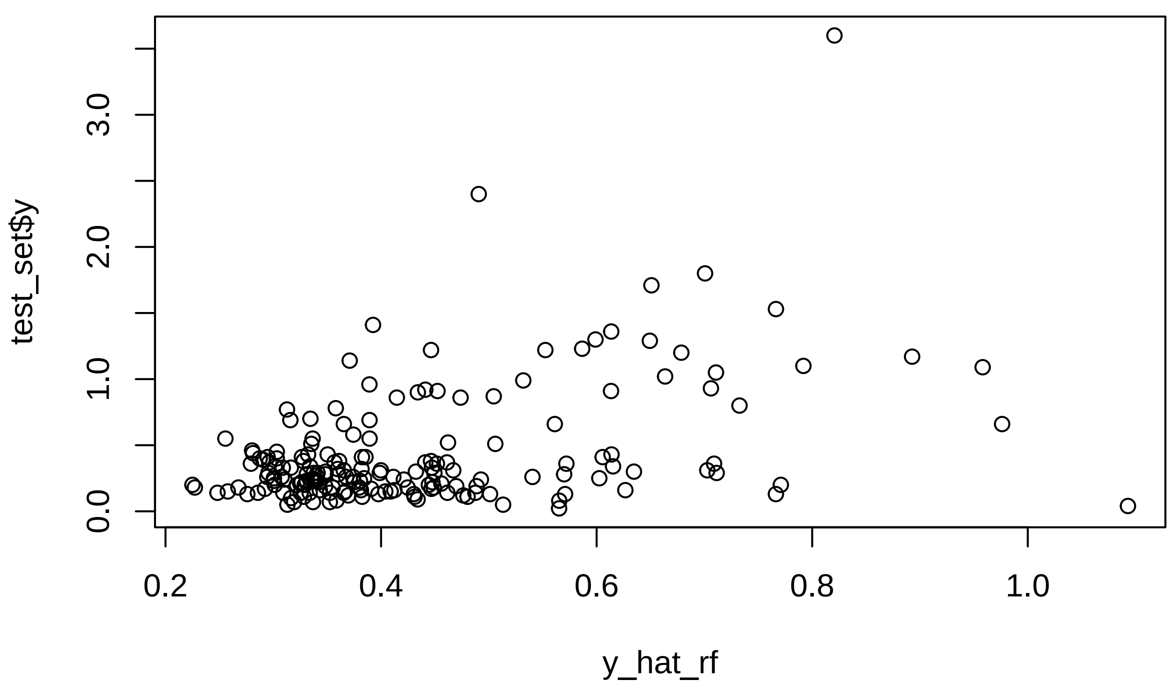
```r
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

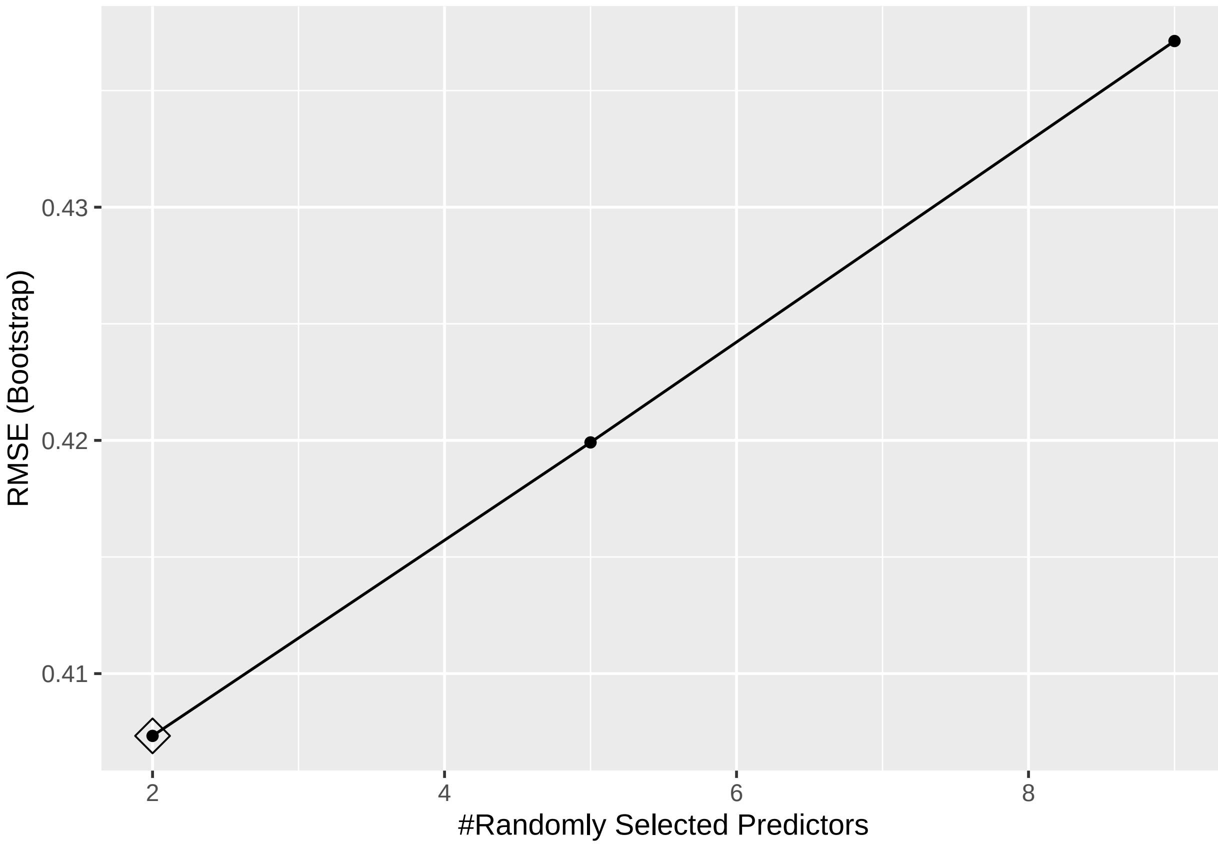
```r
train_rf<- train(y~.,method="rf",data=train_set,preProc = c("center","scale"))
y_hat_rf <- predict(train_rf, test_set, type = "raw")
train_rf$results
```

```
##   mtry      RMSE   Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1    2 0.4073299 0.08257347 0.2793530 0.08673883 0.07864375 0.03997598
## 2    5 0.4199147 0.08966032 0.2843959 0.08599512 0.09159172 0.04378686
## 3    9 0.4371253 0.08280651 0.2929305 0.08768655 0.08866133 0.04614684
```

```r
plot(y_hat_rf,test_set$y)
```



```r
ggplot(train_rf, highlight = TRUE)
```




```r
train_rf$results
```

```
##   mtry      RMSE   Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1    2 0.4073299 0.08257347 0.2793530 0.08673883 0.07864375 0.03997598
## 2    5 0.4199147 0.08966032 0.2843959 0.08599512 0.09159172 0.04378686
## 3    9 0.4371253 0.08280651 0.2929305 0.08768655 0.08866133 0.04614684
```

```r
#fit our final model
mtry = train_rf$bestTune$mtry
fit_rf <- randomForest(y~., mtry = train_rf$bestTune$mtry, data=train_set)
```

3.4- Results - Comparing Models

"To compare different models or to see how well we're doing compared to a baseline, we will use root mean squared error (RMSE) as our loss function."

```r
glm <- train_glm$results$RMSE
knn <- train_knn$results$RMSE
rf <- fit_rf$results$RMSE


methods <- c("glm", "knn","rf")
RMSE_results <- c(glm,knn,rf)

Comparing_Models_RMSE <- as.data.frame(cbind(methods,RMSE_results))
Comparing_Models_RMSE <- Comparing_Models_RMSE %>% arrange(RMSE_results)
```

```r
Comparing_Models_RMSE
```

```
##    methods      RMSE_results
## 1       rf 0.379581960443479
## 2      knn 0.379876048345998
## 3      glm 0.381462018970422
## 4      knn 0.387167350248044
## 5       rf 0.387216029378152
## 6      glm 0.387299876846025
## 7       rf  0.38794066453678
## 8       rf 0.389084841110567
## 9      glm 0.389807325548943
## 10     glm 0.389901282043031
## 11     knn 0.390482031412535
## 12     knn 0.390768179365463
## 13     knn 0.392141321108666
## 14     knn 0.396051893212936
## 15     glm  0.40060805282552
## 16      rf  0.40082022809039
## 17      rf 0.401522156770642
## 18     glm 0.434626483275032
## 19      rf 0.451064985254846
## 20     glm 0.460178560888662
## 21     knn 0.513951546650521
```

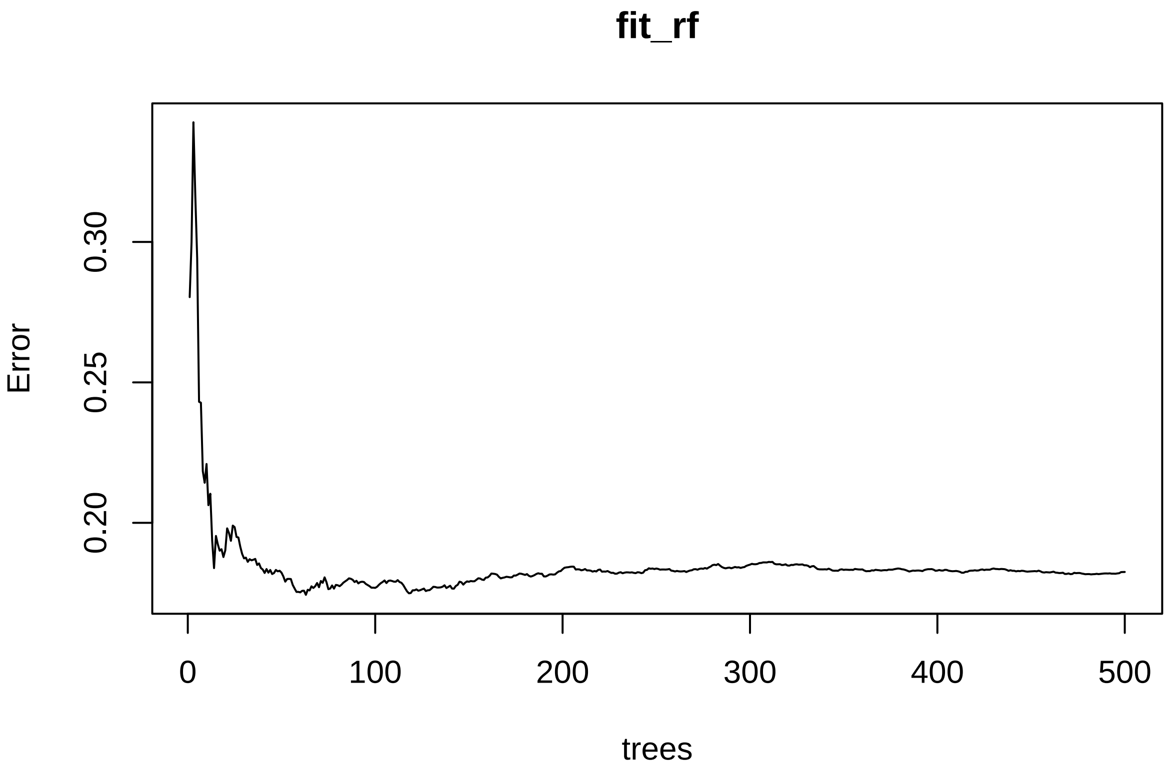
```r
selected_model <- Comparing_Models_RMSE[1,1]

selected_model
```

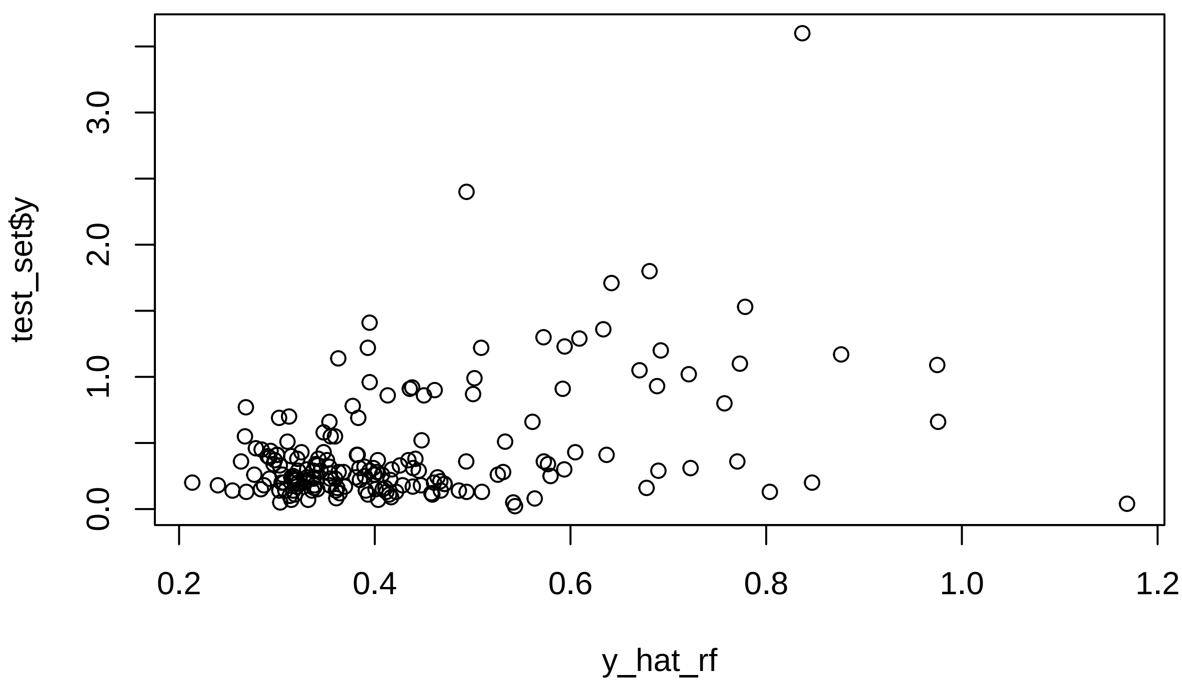
```
## [1] "rf"
```

```r
 #The lower RMSE (0.379581960443479) was from rf


# Do we have enough trees?
plot(fit_rf)
```




```
#Results
y_hat_rf <- predict(fit_rf, test_set)
plot(y_hat_rf,test_set$y)
```



#4-Conclusion

After have compared the 3 models and have optimizated the model (rf) and calculate the Importance(IncNodePurity) of each attribute to the target (NH4) in this model. We could remove from "df" the 3 less important attributes e: BSK5,Suspended,length. #4.1- Future works In this chunk, called "Future works", I try to run all the code for rf again and compare or just the selected model as showed in and see if we are going to find a lower RMSE. After removing and running, I found a higher RMSE. For future, I can also optimizated more, try another methods for models and remove 1 column at a time.

```
#We can calculate the Importance of the variables
imp <- as.data.frame(importance(fit_rf))
imp%>%arrange(desc(IncNodePurity))
```

```
##           IncNodePurity
## CL            2.8556147
## NO3           1.7456642
## O2            1.3930205
## NO2           1.3260303
## SO4           1.2743238
## PO4           1.1678887
## BSK5          0.7905756
## Suspended     0.5960588
## length        0.3042787
```

```
#REMOVE the 3 less importants
df <- df %>% select(-one_of('BSK5','Suspended','length'))
# Split the dataset: train and test sets.
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

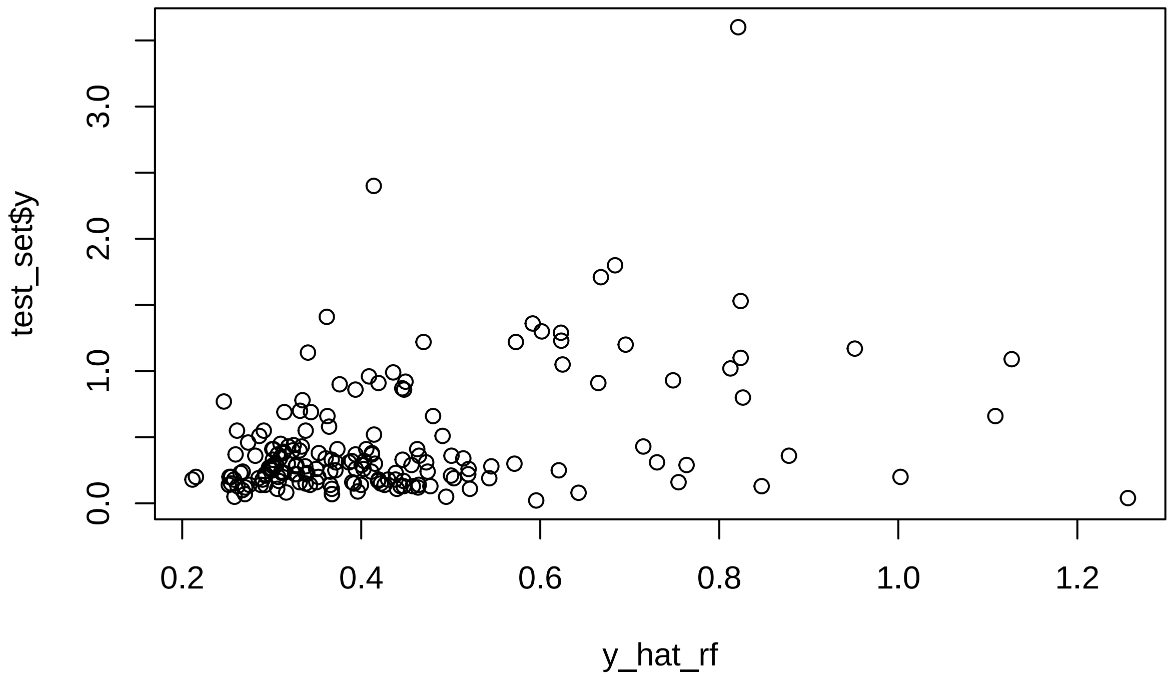
```
test_index <- createDataPartition(df$y, times = 1, p = 0.7, list = FALSE)
train_set <- df%>%slice(-test_index)
test_set <- df%>%slice(test_index)
#Random Forest(rf)
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

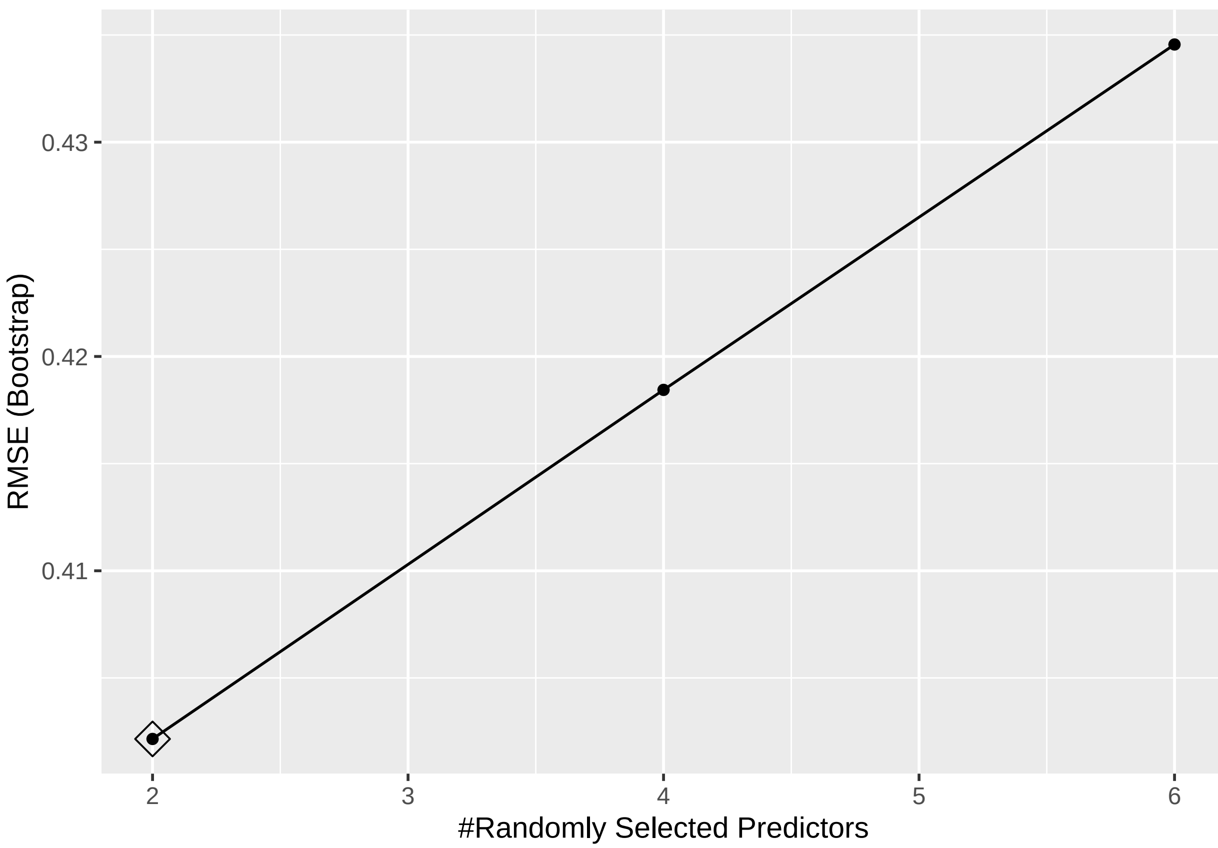
```
train_rf<- train(y~.,method="rf",data=train_set,preProc = c("center","scale"))
y_hat_rf <- predict(train_rf, test_set, type = "raw")
train_rf$results
```

```
##   mtry      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1    2 0.4021534 0.1270502 0.2714992 0.09020316  0.1238934 0.04402098
## 2    4 0.4184410 0.1058008 0.2816563 0.08991175  0.1255156 0.04731326
## 3    6 0.4345579 0.0963078 0.2917334 0.09413615  0.1196816 0.05374767
```

```
plot(y_hat_rf,test_set$y)
```




```r
ggplot(train_rf, highlight = TRUE)
```




```r
train_rf$results
```

```
##   mtry      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1    2 0.4021534 0.1270502 0.2714992 0.09020316  0.1238934 0.04402098
```

```
## 2    4 0.4184410 0.1058008 0.2816563 0.08991175  0.1255156 0.04731326
## 3    6 0.4345579 0.0963078 0.2917334 0.09413615  0.1196816 0.05374767
```

```
#fit our final model
mtry = train_rf$bestTune$mtry
fit_rf <- randomForest(y~., mtry = train_rf$bestTune$mtry, data=train_set)
fit_rf$results$RMSE
```

```
## NULL
```