Student: Evgenia Trudova

Results

This project uses R and four libraries,

library(ggplot2), flexible library commonly used to visualize data in R.

library(caret), a library that is used for training and evaluation of machine learning models, and has a lot of functions that can be used in modeling, adjustment of models, evaluation of data. In this project, caret is picket to be able to calculate mean absolute error,MAE.

library(randomForest), runs randomForest algorithm on the data to show results as a model.

library(nnet), neutral network specifik library, this model is often compared to randomForest, as it outperforms it at certain conditions and sometimes randomForest outperform neutral network, depending on variation in the data.

The models that are plotted, are three linear regressions with response vs LogP, response vs TopoPSA and response vs MW. Scatter points represent data from the dataset. Regression line represents prediction of respective model and model is evaluated with three functions, mean Squared Error (RMSA)m Mean absolute Error (MAE) and variance evaluation R-squared coefficent (R^2).

Root Mean Squared Error (RMSA) metric that is used to keep the model sensetive, because it penalizes error, low RMSA is better model.

Mean absolute Error(MAE) metric checks model for prediction errors.

R-squared (R^2) method, is common linear regression evaluation of proportion of variance, in rande from 0 to 1, higher number indicates better model fil (example $R^2 = 0.7$ is equivalent of 70% data fit).

Intercept and slope are helping us build a linear regression graph, by calculating change between data points and predict a line from existing data points.

Purpose of this project was to visualise data and apply evaluation methods to data. This was done by interacting with exam.Rdata file by loading it into R, and further interacting with “dataset” inside of the file that contains data of intrest. Dataset is opened and visualised and appeared to contain a lot of empty data spots, which is the reason why variables with missing values was removed. Additionally values with variance below 0,5 was removed.

There are multiple strategies for how to “clean” data using R, one way is to use logic and missing values(na).

```
dataset_clean   <- dataset[ , colSums(is.na(dataset)) == 0]
```

Data is stored in columns.New intended folder with data, called (dataset_clean) will only contain information from the dataset, that passes a logical ==0 equation, to not contain missing values(NA). This effectivly removes columns with missing values.

Same logic is applied to removing columns with variance below 0.5. (>= 0.5).

Linear Regression models are build and evaluated with the same design, reaching in for different columns into dataset.It is difficult to identify which linear regression model is which, when overlooking the simple visualisation. It is also difficult to analyse evaluation work done on the models, that data can be left in the code and not show visualisation on the scatter plot. That is why results of evaluation by RMSA, MAE and R^2 are added as text under the title of the models.

Linear regressions are created with,

lm_model AB <- lm (variable A ~ variable B, data = train_data)

Predictions of data from the linear regression are created with predict function.

predictions_lm_AB <- predict(lm_model_AB, newdata = val_data)

predictions_lm_AB are then used in RMSE, MAE and R-squared to provide respective metric of evaluation of quality of linear regression that was created. This is done by respective function, just changing name of the variable.

rmse_lm_AB <- sqrt(mean((val_data$A - predictions_lm_AB)^2))

mae_lm_AB <- mean(abs(val_data$A - predictions_lm_AB))

r_squared_lm_AB <- cor(predictions_lm_AB, val_data$A)^2

coefficients_lm_AB <- summary(lm_model_AB)$coefficiaents

To add text to a metric data, one uses

cat(“text”, link to the evaluation method, “\n”)

Before making a plot, R reaches into the dataset, using a function to create a new folder plot_data containing only variables of interest and predictions of interest.

plot_data AB <- data.frame(variable A = val_data$B, variable B = val_data$A, predictions = predictions_lm_AB)

To write a subtitle, one creates a new folder metrics_text, and sprint method that contains text of the subtitle with space for numbers a separation by %.3f |, in that space that will be changeable data, data that is calculated in respective folder with for example, evaluation using RMSE is calculated and result is then moved to subtitle storage to be visualized as numbers on the graph.

```
metrics_text <- sprintf(
"RMSE: %.3f | MAE: %.3f ",
rmse_lm_AB, mae_lm_AB,
)
```

Plot is made using ggplot function.

```
ggplot(plot_data_AB, aes(x = A, y = B)) +
    geom_point() +
    geom_line(aes(y = predictions), color = "blue", size = 1) +
    labs(
      x = "A",
      y = "B",
      title = "Linear Regression: A vs. B",
      subtitle = metrics_text
    ) +
    theme_minimal() +
    theme(
      plot.subtitle = element_text(size = 10, hjust = 0.5)
    )
```

Finally, the data is analyzed using randomForest and neutralNetworks, this is done with functions and methods for training of a model. Logic is used in similar way to evaluation code, but Boolean variant with TRUE or FALSE is used.

A simple model training

```
set.seed(123) # is used to randomize the results.
model_rf <- randomForest(A ~ ., data = train_data, importance = TRUE)
```

A new folder model_rf is created in it, only data that goes through randomForest is stored, where variable of interest is data, that goes through training, this must be controlled.

randomForest can also be evaluated and has a prediction strategy, that looks similar to linear regression evaluation but uses folders with data instead of tables with data.

```
pred_rf <- predict(model_rf, val_data)
```

```
rmse_rf <- sqrt(mean((val_data$A - pred_rf)^2))
```

varImpPlot(model_rf) is used to visualize randomForest.

randomForest is also evaluated using cross-validation method, this method uses train() and sets specific demands on how data is evaluated, how many times and in how many folds. Cross valiadation method is used to control quality of randomForest model.

NeutralNetrwork is created using library(nnet) function and outlined conditions of how many time data is checked and when checking stops.R performs these checking for 5-10 minutes.

```
set.seed(123)
```

```
nn_model <- nnet(A ~ ., data = train_data, size = 5, linout = TRUE, maxit = 100)
```

Conclusion:

Data from RMSE, MAE and R^2 indicates that Random Forest RMSE is lowest, which means that it outperforms the linear regression models and neutral network.

Model	RMSE	MAE	R^2
Linear Regression (XLogP)	1.1360	0.9079	0.6582
Linear Regression (TopoPSA)	0.9120	0.7160	0.8188
Linear Regression (MW)	1.6665	1.1880	0.2885
Random Forest	**0.6724**	n/a	n/a
Neutral Network	1.4898	1.1568	0.4292

Location and Visualisation of the dataset.

Control of data in the file. Objective is to find the file and check on type of data in the file.

```
setwd("C:/Users/evgen/Downloads")

getwd()

load (”exam.Rdata”)

ls()

    ”dataset”

str(dataset)

     1142 obs of 454 variables

summary(dataset)

    Response / tpsaEfficiency / TopoPSA /nHbDon
```

# The fine contains data that needs to be cleaned and visualized, three libraries are suitable for this task (gglot2,randomForest and caret) the are downloaded in R.

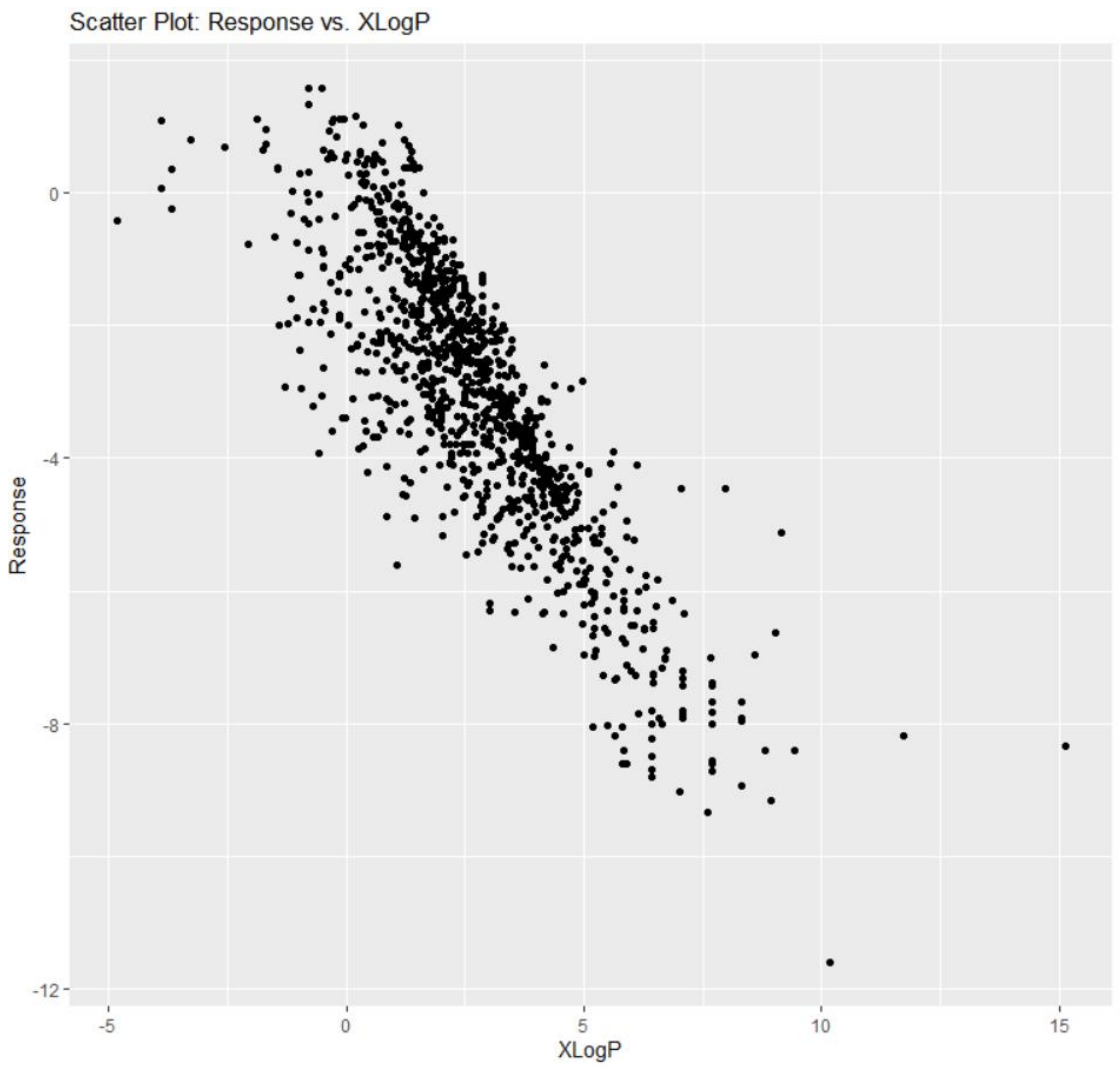
```
library (ggplot2)

library(randomForest)

library(caret)
```

# Control of data.

```
load("exam.Rdata")

ls()

[1] "dataset"

str(dataset)

     1142 obs of 454 variables

summary(dataset)
```

Visualisation of data with gglot2

```
ggplot(dataset, aes(x = XLogP, y = response)) +
    geom_point() +
    labs(x = "XLogP", y = "Response") +
    ggtitle("Scatter Plot: Response vs. XLogP")
```



Cleaning of data (remove data with missing values + remove data with variance under 0,5)

```
dataset_clean <- dataset[ , colSums(is.na(dataset)) == 0]

dataset_clean <- dataset_clean[, apply(dataset_clean, 2, var) >= 0.5]

summary(dataset_clean)
```

```
response          TopoPSA          nHBDon            nHBAcc
  bpol              apol          nSmallRings      nAromRings
nRingBlocks      nAromBlocks        nRings6          Zagreb
WPATH               WPOL            WTPT.1            WTPT.3
WTPT.4            WTPT.5          VAdjMat          MDEC.12
MDEC.13             MDEC.14         MDEC.22           MDEC.23
MDEC.24             MDEC.33         MDEC.34            khs.sCH3
khs.ssCH2          khs.aaCH        khs.sssCH          khs.dssC
khs.aasC          khs.aaaC        khs.ssssC         khs.sOH
  khs.dO           khs.ssO          khs.sCl           Kier1
Kier2             fragC             ECCEN             SP.0
  SP.1              SP.2            SP.3              SP.4
  SP.5              SP.6            SP.7              VP.0
VP.1              VP.2            VP.3              VP.4
VP.5                VP.6              VP.7              SPC.4
SPC.5               SPC.6             VPC.4               VPC.5
VPC.6                 SC.3            VC.3                C1SP2
C2SP2               C3SP2             C1SP3             C2SP3
C3SP3               ATSp1               ATSp2             ATSp3
  ATSp4               ATSp5           ATSm1               ATSm2
ATSm3               ATSm4             ATSm5                XLogP
  MW                 nRotB          nAtomLAC            nAtomP
  nAtomLC                 nB          nAtom             nAromBond
naAromAtom           ALogP              ALogp2                AMR
BCUTw.1h           BCUTp.1l          BCUTp.1h
```

Splitting data into training (80%) and validation(20%) sets.

```
set.seed(123)

train_index <- createDataPartition(dataset_clean$response, p = 0.8, list = FALSE)

train_data <- dataset_clean[train_index, ]

val_data <- dataset_clean[-train_index, ]
```

Simple Linear regression with LogP from the dataset and simple prediction method

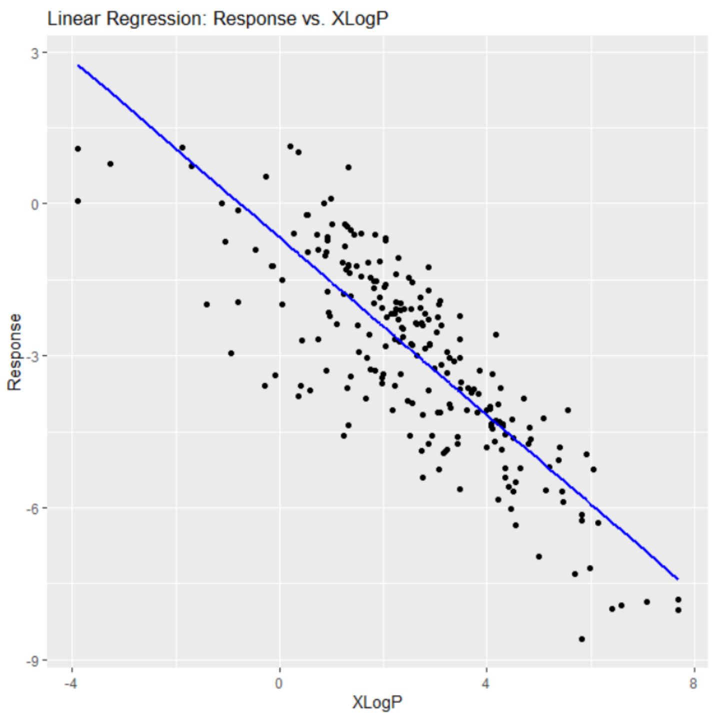
```
lm_model_XLogP <- lm(response ~ XLogP, data = train_data)

predictions_lm_XLogP <- predict(lm_model_XLogP, newdata = val_data)

rmse_lm_XLogP <- sqrt(mean((val_data$response - predictions_lm_XLogP)^2))
```

Evaluation of performance

```
rmse_lm_XLogP
```

```
[1] 1.136048
```

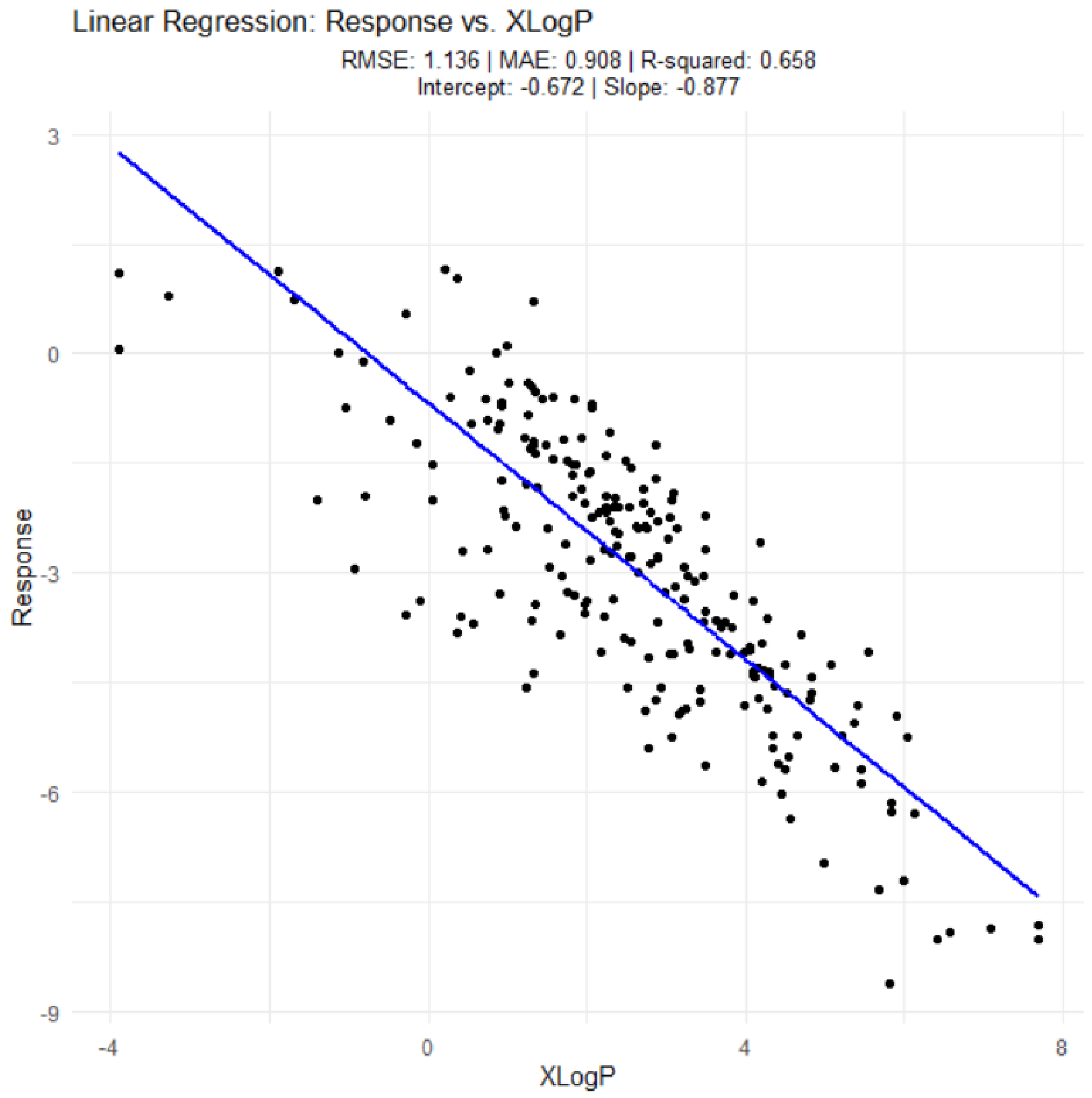
```
lm_LogP <- lm(response ~ XLogP, data = train_data)

predictions_LogP <- predict(lm_LogP, newdata = val_data)

plot_data_LogP <- data.frame(XLogP = val_data$XLogP, response = val_data$response,
predictions = predictions_LogP)

ggplot(plot_data_LogP, aes(x = XLogP, y = response)) +

   geom_point() +

   geom_line(aes(y = predictions), color = "blue", size = 1) +

   labs(x = "XLogP", y = "Response") +

   ggtitle("Linear Regression: Response vs. XLogP")
```



Linear regression with evaluation function.

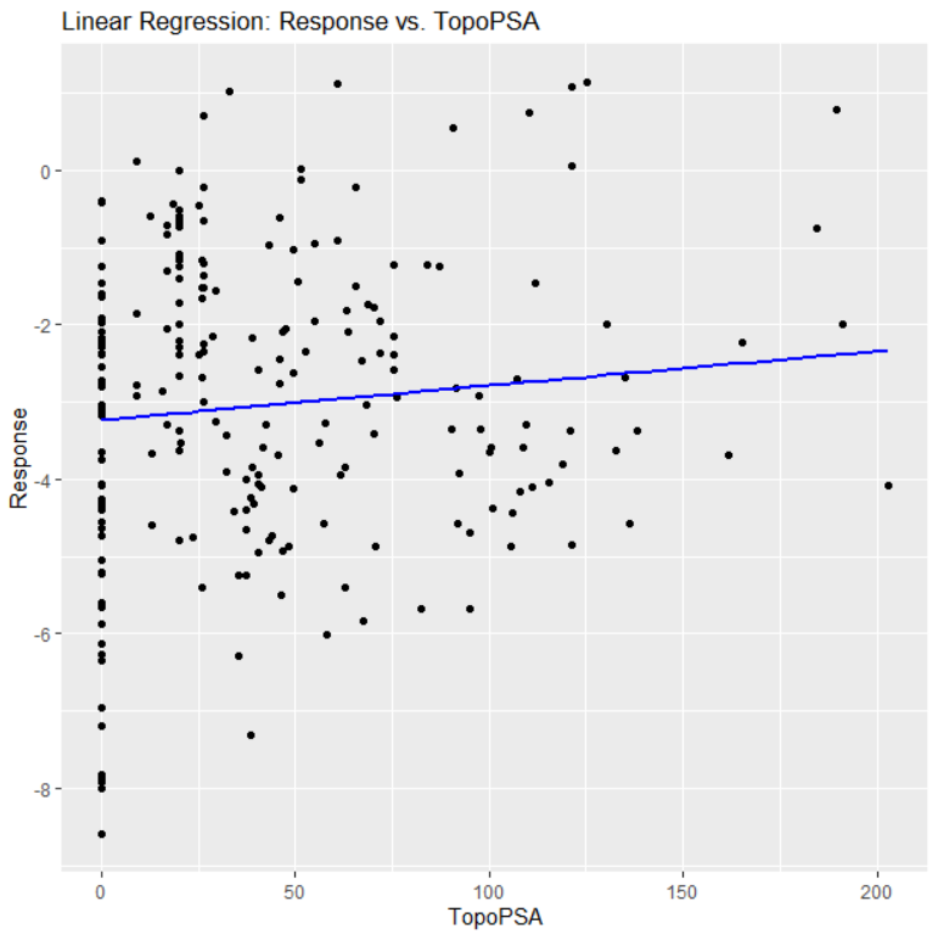
```
library(ggplot2)

library(caret)

set.seed(123)

lm_model_XLogP <- lm(response ~ XLogP, data = train_data)

predictions_lm_XLogP <- predict(lm_model_XLogP, newdata = val_data)

# RMSE

rmse_lm_XLogP <- sqrt(mean((val_data$response - predictions_lm_XLogP)^2))

# MAE

mae_lm_XLogP <- mean(abs(val_data$response - predictions_lm_XLogP))

# R2

r_squared_lm_XLogP <- cor(predictions_lm_XLogP, val_data$response)^2

# Coefficient

coefficients_lm_XLogP <- summary(lm_model_XLogP)$coefficients

# Metrics

cat("RMSE:", rmse_lm_XLogP, "\n")

RMSE: 1.136048

cat("MAE:", mae_lm_XLogP, "\n")

MAE: 0.9079298

cat("R-squared:", r_squared_lm_XLogP, "\n")

R-squared: 0.6582457

cat("Coefficients:\n")

Coefficients:

print(coefficients_lm_XLogP)

              Estimate Std. Error   t value      Pr(>|t|)

(Intercept) -0.6723153 0.06328441 -10.62371   6.181751e-25

XLogP       -0.8765902 0.01853446 -47.29515 1.251468e-247
```

```
> # Plot
plot_data_LogP <- data.frame(XLogP = val_data$XLogP, response = val_data$response,
predictions = predictions_lm_XLogP)
metrics_text <- sprintf(
    "RMSE: %.3f | MAE: %.3f | R-squared: %.3f\nIntercept: %.3f | Slope: %.3f",
    rmse_lm_XLogP, mae_lm_XLogP, r_squared_lm_XLogP,
    coefficients_lm_XLogP["(Intercept)", "Estimate"],
    coefficients_lm_XLogP["XLogP", "Estimate"]
 )
ggplot(plot_data_LogP, aes(x = XLogP, y = response)) +
    geom_point() +
    geom_line(aes(y = predictions), color = "blue", size = 1) +
    labs(
      x = "XLogP",
      y = "Response",
      title = "Linear Regression: Response vs. XLogP",
      subtitle = metrics_text
    )   +
   theme_minimal() +
   theme(
     plot.subtitle = element_text(size = 10, hjust = 0.5)
   )
```


Linear Regression: Response vs. XLogP
RMSE: 1.136 | MAE: 0.908 | R-squared: 0.658
Intercept: -0.672 | Slope: -0.877
3
0
-3
-6
-9
Response
-4
0
4
8
XLogP

Linear regression line for Response vs. TopoPSA, with predictions and dataframe preparation for scatter plot with regression line.

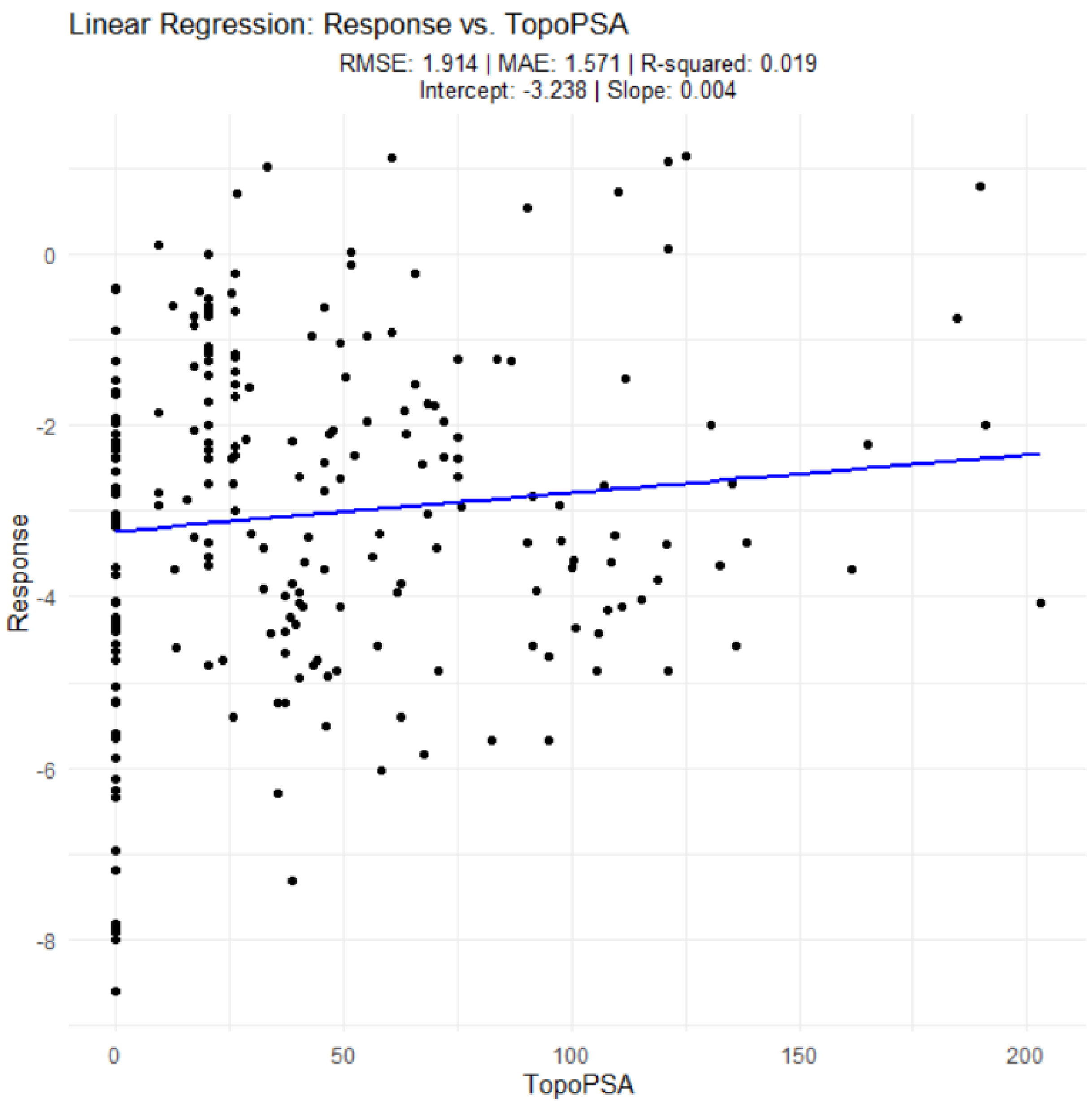
```
lm_TopoPSA <- lm(response ~ TopoPSA, data = train_data)

predictions_TopoPSA <- predict(lm_TopoPSA, newdata = val_data)

plot_data_TopoPSA <- data.frame(TopoPSA = val_data$TopoPSA, response =
val_data$response, predictions = predictions_TopoPSA)

ggplot(plot_data_TopoPSA, aes(x = TopoPSA, y = response)) +

    geom_point() +

    geom_line(aes(y = predictions), color = "blue", size = 1) +

    labs(x = "TopoPSA", y = "Response") +

    ggtitle("Linear Regression: Response vs. TopoPSA")
```



Linear regression line for Response vs. TopoPSA, with evaluation methods.

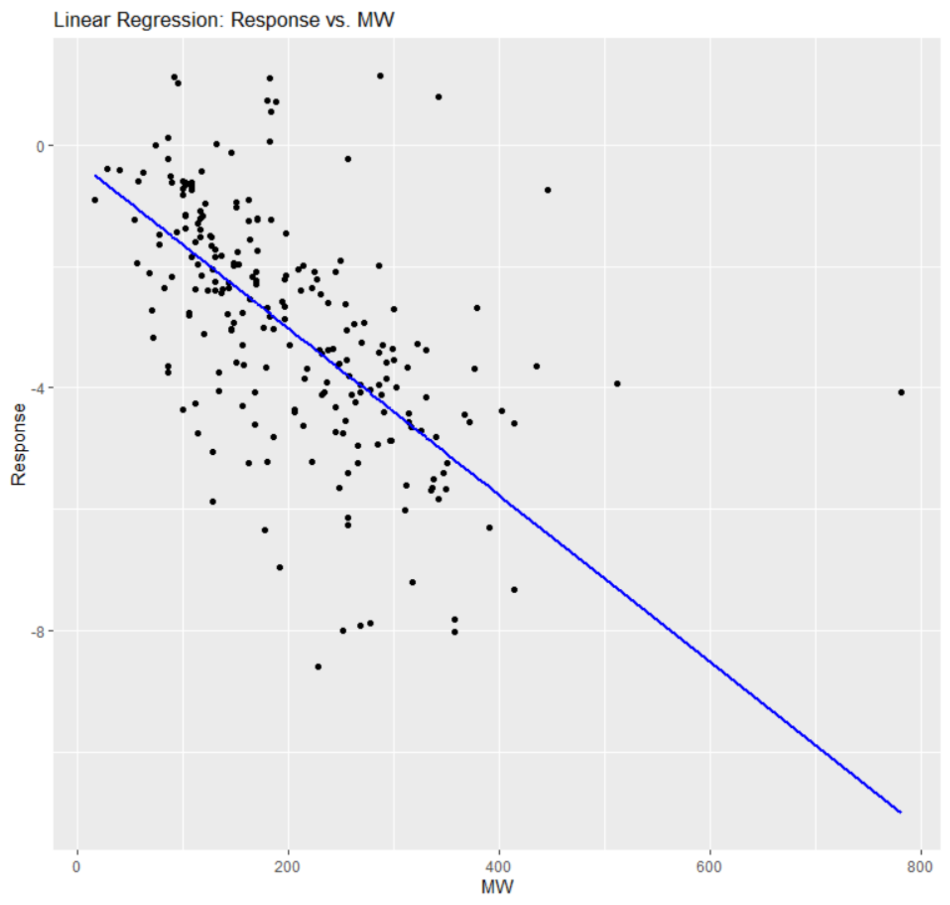
```
library(ggplot2)

library(caret)

set.seed(123)

lm_TopoPSA <- lm(response ~ TopoPSA, data = train_data)

predictions_TopoPSA <- predict(lm_TopoPSA, newdata = val_data)

# RMSE

rmse_lm_TopoPSA <- sqrt(mean((val_data$response - predictions_TopoPSA)^2))

# MAE

mae_lm_TopoPSA <- mean(abs(val_data$response - predictions_TopoPSA))

# R2

r_squared_lm_TopoPSA <- cor(predictions_TopoPSA, val_data$response)^2

# Coefficient

coefficients_lm_TopoPSA <- summary(lm_TopoPSA)$coefficients

# Metrics

cat("RMSE:", rmse_lm_TopoPSA, "\n")

cat("MAE:", mae_lm_TopoPSA, "\n")

cat("R-squared:", r_squared_lm_TopoPSA, "\n")

cat("Coefficients:\n")

print(coefficients_lm_TopoPSA)

# Plot

plot_data_TopoPSA <- data.frame(TopoPSA = val_data$TopoPSA, response =
val_data$response, predictions = predictions_TopoPSA)

metrics_text <- sprintf(

  "RMSE: %.3f | MAE: %.3f | R-squared: %.3f\nIntercept: %.3f | Slope: %.3f",

  rmse_lm_TopoPSA, mae_lm_TopoPSA, r_squared_lm_TopoPSA,

  coefficients_lm_TopoPSA["(Intercept)", "Estimate"],
```

```r
  coefficients_lm_TopoPSA["TopoPSA", "Estimate"]
)
ggplot(plot_data_TopoPSA, aes(x = TopoPSA, y = response)) +
  geom_point() +
  geom_line(aes(y = predictions), color = "blue", size = 1) +
  labs(
    x = "TopoPSA",
    y = "Response",
    title = "Linear Regression: Response vs. TopoPSA",
    subtitle = metrics_text
  ) +
  theme_minimal() +
  theme(
    plot.subtitle = element_text(size = 10, hjust = 0.5)
  )
```


Linear Regression: Response vs. TopoPSA
RMSE: 1.914 | MAE: 1.571 | R-squared: 0.019
Intercept: -3.238 | Slope: 0.004
Response
0
-2
-4
-6
-8
TopoPSA
0
50
100
150
200

Linear regression line Response vs. MW

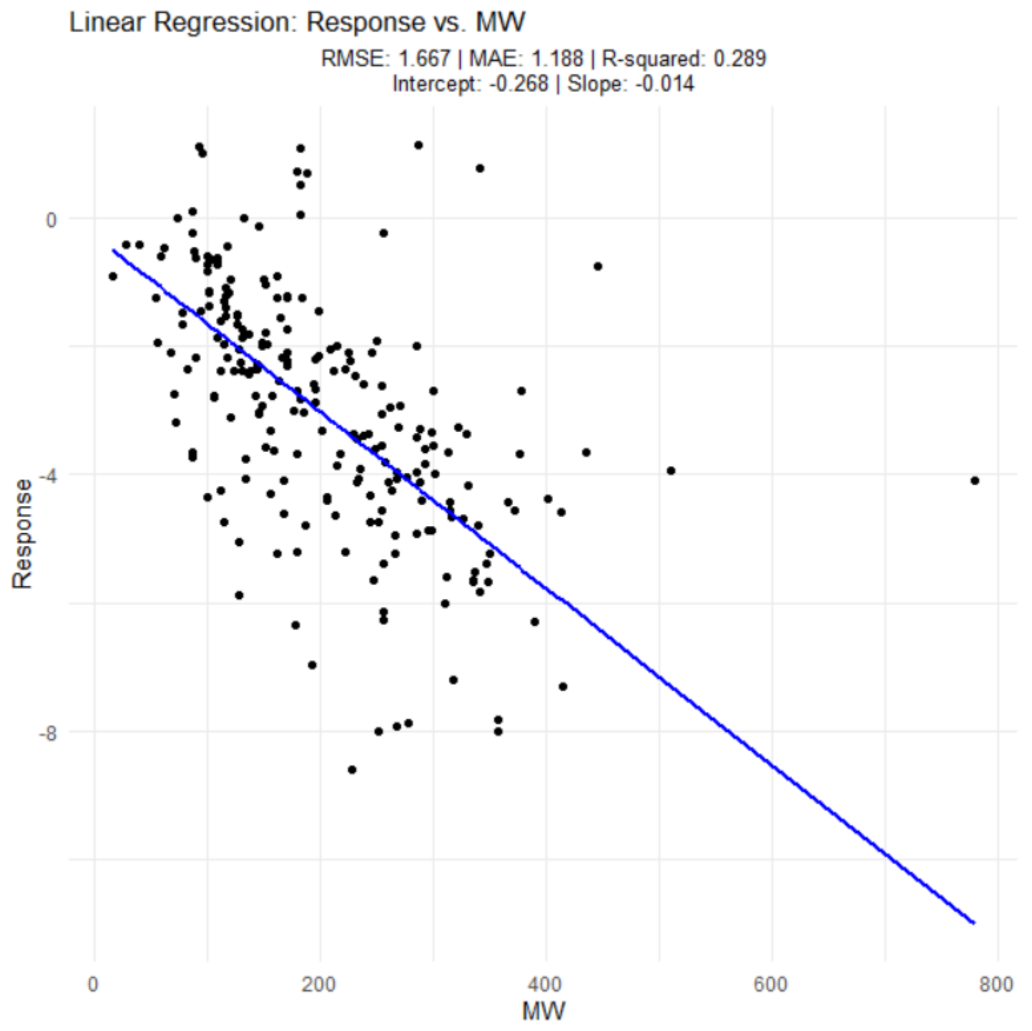
```
lm_MW <- lm(response ~ MW, data = train_data)

predictions_MW <- predict(lm_MW, newdata = val_data)

plot_data_MW <- data.frame(MW = val_data$MW, response = val_data$response, predictions =
          predictions_MW)

ggplot(plot_data_MW, aes(x = MW, y = response)) +
  geom_point() +
  geom_line(aes(y = predictions), color = "blue", size = 1) +
  labs(x = "MW", y = "Response") +
  ggtitle("Linear Regression: Response vs. MW")
```



Linear regression line Response vs. MW with evaluation methods.

```
library(ggplot2)

library(caret)

set.seed(123)

lm_MW <- lm(response ~ MW, data = train_data)

predictions_MW <- predict(lm_MW, newdata = val_data)

# RMSE

rmse_lm_MW <- sqrt(mean((val_data$response - predictions_MW)^2))

# MAE

mae_lm_MW <- mean(abs(val_data$response - predictions_MW))

# R2

r_squared_lm_MW <- cor(predictions_MW, val_data$response)^2

# Coefficient

coefficients_lm_MW <- summary(lm_MW)$coefficients

# Metrics

cat("RMSE:", rmse_lm_MW, "\n")

RMSE: 1.666544

> cat("MAE:", mae_lm_MW, "\n")

MAE: 1.188021

cat("R-squared:", r_squared_lm_MW, "\n")

R-squared: 0.2885051

cat("Coefficients:\n")

Coefficients:

print(coefficients_lm_MW)

               Estimate    Std. Error    t value       Pr(>|t|)

(Intercept) -0.26815274 0.1188425514   -2.25637   2.428356e-02

MW          -0.01376098 0.0005226545 -26.32901 4.343319e-114
```

```
> # Plot
plot_data_MW <- data.frame(MW = val_data$MW, response = val_data$response, predictions =
predictions_MW)
metrics_text_MW <- sprintf(
    "RMSE: %.3f | MAE: %.3f | R-squared: %.3f\nIntercept: %.3f | Slope: %.3f",
    rmse_lm_MW, mae_lm_MW, r_squared_lm_MW,
    coefficients_lm_MW["(Intercept)", "Estimate"],
    coefficients_lm_MW["MW", "Estimate"]
  )
ggplot(plot_data_MW, aes(x = MW, y = response)) +
    geom_point() +
    geom_line(aes(y = predictions), color = "blue", size = 1) +
    labs(
      x = "MW",
      y = "Response",
      title = "Linear Regression: Response vs. MW",
      subtitle = metrics_text_MW
    ) +
    theme_minimal() +
    theme(
      plot.subtitle = element_text(size = 10, hjust = 0.5)   # Adjust text size and position
    )
>
```


Linear Regression: Response vs. MW
RMSE: 1.667 | MAE: 1.188 | R-squared: 0.289
Intercept: -0.268 | Slope: -0.014
Response
MW
0
-4
-8
0
200
400
600
800

Simple Random Forest

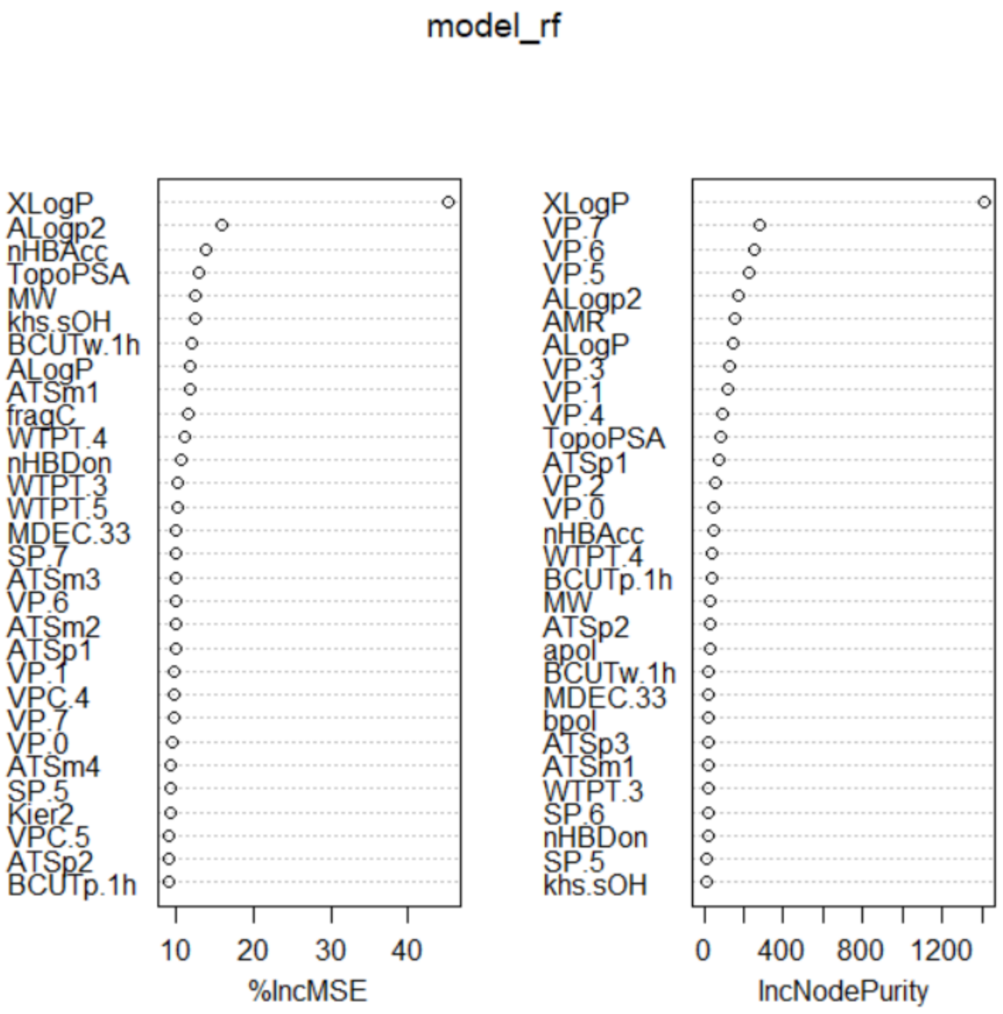
```
library(randomForest)
set.seed(123)
model_rf <- randomForest(response ~ ., data = train_data, importance = TRUE)
pred_rf <- predict(model_rf, val_data)
rmse_rf <- sqrt(mean((val_data$response - pred_rf)^2))
var_importance <- importance(model_rf)
summary(model_lm1)
```

```
Call:
lm(formula = response ~ XLogP, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.0054 -0.7250  0.1039  0.8316  5.6046 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.67232    0.06328  -10.62   <2e-16 ***
XLogP       -0.87659    0.01853  -47.30   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.149 on 912 degrees of freedom
Multiple R-squared:  0.7104,	Adjusted R-squared:  0.7101 
F-statistic:  2237 on 1 and 912 DF,  p-value: < 2.2e-16
```

```
summary(model_lm2)

Call:
lm(formula = response ~ XLogP + TopoPSA + MW, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5083 -0.5667 -0.0072  0.5697  4.8416

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.4349643  0.0716221   6.073 1.84e-09 ***
XLogP       -0.7137078  0.0245073 -29.122  < 2e-16 ***
TopoPSA      0.0001108  0.0013708   0.081    0.936
MW          -0.0076546  0.0005258 -14.559  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9096 on 910 degrees of freedom
Multiple R-squared:  0.8188,	Adjusted R-squared:  0.8182
F-statistic:  1370 on 3 and 910 DF,  p-value: < 2.2e-16
print(model_performance)
                                 Model      RMSE
1             Linear Regression (XLogP) 1.1360477
2 Linear Regression (XLogP, TopoPSA, MW) 0.9120217
3                         Random Forest 0.6723658
```

Random Forest with evaluation

```
install.packages("randomForest")

set.seed(123)

model_rf <- randomForest(response ~ ., data = train_data, importance = TRUE)

pred_rf <- predict(model_rf, val_data)

rmse_rf <- sqrt(mean((val_data$response - pred_rf)^2))

var_importance <- importance(model_rf)

varImpPlot(model_rf)
```



Validation of Random Forest

```
model_lm1 <- lm(response ~ XLogP, data = train_data)

model_lm2 <- lm(response ~ XLogP + TopoPSA + MW, data = train_data)


summary(model_lm1)

Call:

lm(formula = response ~ XLogP, data = train_data)


Residuals:

    Min      1Q  Median      3Q     Max

-4.0054 -0.7250  0.1039  0.8316  5.6046


Coefficients:

            Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.67232    0.06328  -10.62   <2e-16 ***

XLogP       -0.87659    0.01853  -47.30   <2e-16 ***

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 1.149 on 912 degrees of freedom

Multiple R-squared:  0.7104,    Adjusted R-squared:  0.7101

F-statistic:  2237 on 1 and 912 DF,  p-value: < 2.2e-16
```

```
summary(model_lm2)

Call:
lm(formula = response ~ XLogP + TopoPSA + MW, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5083 -0.5667 -0.0072  0.5697  4.8416

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.4349643  0.0716221   6.073 1.84e-09 ***
XLogP       -0.7137078  0.0245073 -29.122  < 2e-16 ***
TopoPSA      0.0001108  0.0013708   0.081    0.936
MW          -0.0076546  0.0005258 -14.559  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9096 on 910 degrees of freedom
Multiple R-squared:  0.8188,	Adjusted R-squared:  0.8182
F-statistic:  1370 on 3 and 910 DF,  p-value: < 2.2e-16
```

Cross Validation Train with Caret, data set is split into 10 folds, for each fold a Random Forest model is trained on 9 folds and validated on the remaining fold.

```
library(caret)

library(randomForest)

set.seed(123)

cv_control <- trainControl(

    method = "cv",

    number = 10,

    summaryFunction = defaultSummary,

    verboseIter = TRUE

 )

 rf_cv_model <- train(

    response ~ .,

    data = dataset_clean,

    method = "rf",

    trControl = cv_control,

    tuneLength = 3

 )
```

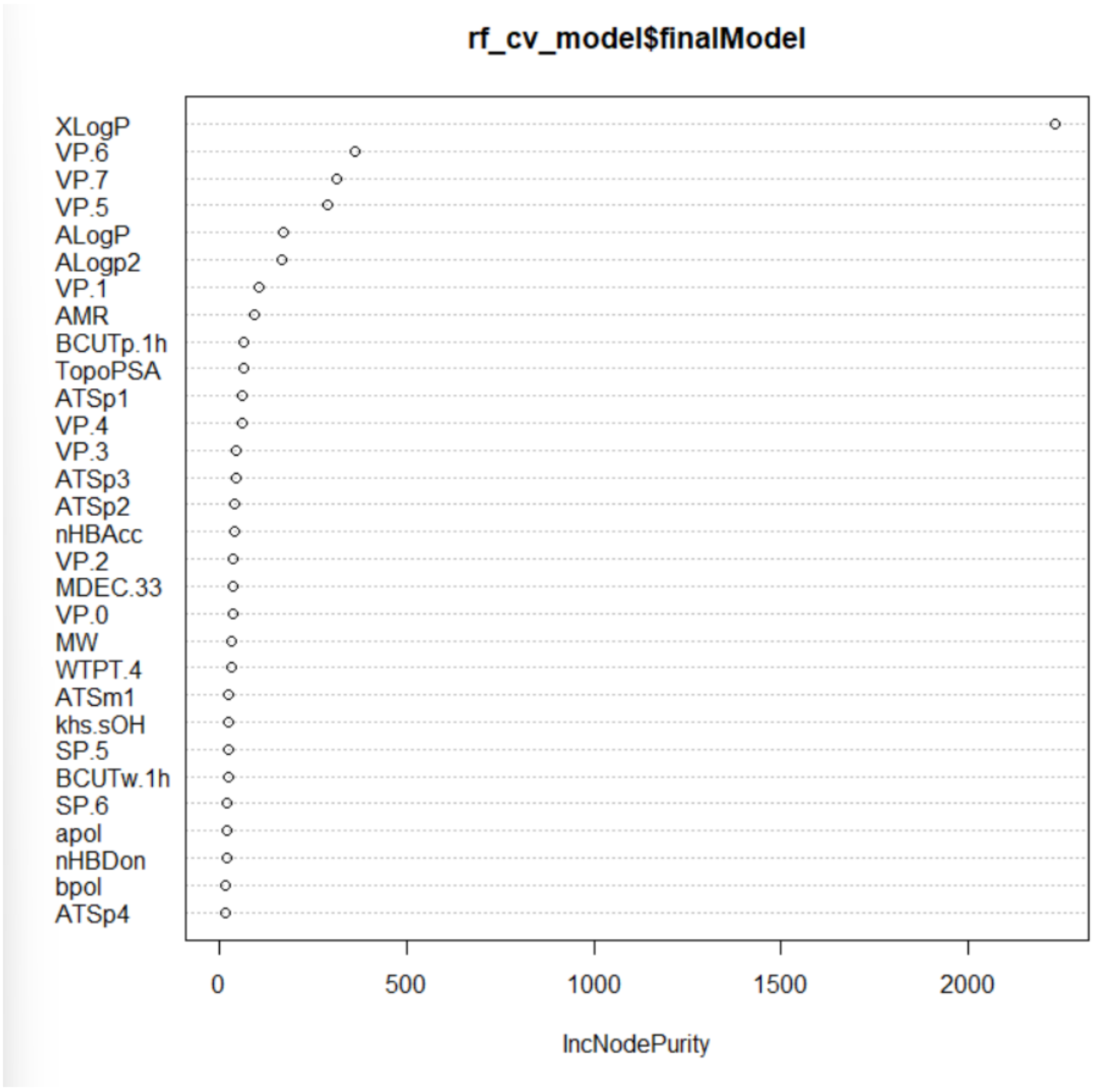
```
+ Fold01: mtry= 2

- Fold01: mtry= 2

+ Fold01: mtry=50

- Fold01: mtry=50

+ Fold01: mtry=98

- Fold01: mtry=98

+ Fold02: mtry= 2

- Fold02: mtry= 2

+ Fold02: mtry=50
```

```
- Fold02: mtry=50
+ Fold02: mtry=98
- Fold02: mtry=98
+ Fold03: mtry= 2
- Fold03: mtry= 2
+ Fold03: mtry=50
- Fold03: mtry=50
+ Fold03: mtry=98
- Fold03: mtry=98
+ Fold04: mtry= 2
- Fold04: mtry= 2
+ Fold04: mtry=50
- Fold04: mtry=50
+ Fold04: mtry=98
- Fold04: mtry=98
+ Fold05: mtry= 2
- Fold05: mtry= 2
+ Fold05: mtry=50
- Fold05: mtry=50
+ Fold05: mtry=98
- Fold05: mtry=98
+ Fold06: mtry= 2
- Fold06: mtry= 2
+ Fold06: mtry=50
- Fold06: mtry=50
+ Fold06: mtry=98
- Fold06: mtry=98
+ Fold07: mtry= 2
- Fold07: mtry= 2
+ Fold07: mtry=50
- Fold07: mtry=50
+ Fold07: mtry=98
- Fold07: mtry=98
```

```
+ Fold08: mtry= 2

- Fold08: mtry= 2

+ Fold08: mtry=50

- Fold08: mtry=50

+ Fold08: mtry=98

- Fold08: mtry=98

+ Fold09: mtry= 2

- Fold09: mtry= 2

+ Fold09: mtry=50

- Fold09: mtry=50

+ Fold09: mtry=98

- Fold09: mtry=98

+ Fold10: mtry= 2

- Fold10: mtry= 2

+ Fold10: mtry=50

- Fold10: mtry=50

+ Fold10: mtry=98

- Fold10: mtry=98

Aggregating results

Selecting tuning parameters

Fitting mtry = 50 on full training set
```

Plotting of Random Forest model, Response vs All Variables.

```
> print(rf_cv_model)

Random Forest

1142 samples

  98 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 1026, 1028, 1028, 1029, 1027, 1027, ...

Resampling results across tuning parameters:


  mtry   RMSE          Rsquared    MAE

    2      0.7599642   0.8769532   0.5599094

  50      0.6279393   0.9105673   0.4545395

  98      0.6363250   0.9081007   0.4599597


RMSE was used to select the optimal model using the smallest value.

The final value used for the model was mtry = 50.

>varImpPlot(rf_cv_model$finalModel)
```



```
> print(var_importance)

                %IncMSE IncNodePurity

TopoPSA       12.858620      82.2707101

nHBDon         10.614180      18.0078223

nHBAcc        13.725011      46.9622309

bpol          6.534381      22.7523190

apol          6.703213      29.2679526

nSmallRings   3.577169       0.9465039

nAromRings     3.174403       1.1815470

nRingBlocks   3.590537       1.2592731
```

nAromBlocks	2.156715	1.1647310
nRings6	2.230785	1.0056321
Zagreb	5.362165	3.9380934
WPATH	8.833718	13.4058869
WPOL	3.030717	3.0252645
WTPT.1	6.344454	9.4515172
WTPT.3	10.211059	20.3324599
WTPT.4	11.019605	41.7552285
WTPT.5	10.189428	9.3416319
VAdjMat	3.860414	2.8476809
MDEC.12	7.418983	7.5108602
MDEC.13	6.259188	5.4849144
MDEC.14	3.179977	1.2919394
MDEC.22	8.478058	12.6551182
MDEC.23	7.299897	12.0270155
MDEC.24	3.535487	0.9557685
MDEC.33	9.986211	25.2107945
MDEC.34	2.280937	1.0224433
khs.sCH3	4.385373	2.6125589
khs.ssCH2	4.007765	5.5469047
khs.aaCH	4.435462	2.3768254
khs.sssCH	4.227994	5.2802532
khs.dssC	3.822433	1.9058082
khs.aasC	8.809841	10.1654116
khs.aaaC	2.511473	1.0431349
khs.ssssC	1.719592	0.3992240
khs.sOH	12.329598	16.3701718
khs.dO	4.583525	2.8816238
khs.ssO	4.773136	2.6090049
khs.sCl	3.721777	4.6476335

Kier1	8.727650	10.7748399
Kier2	9.148477	13.9446697
fragC	11.540326	12.9090861
ECCEN	5.984523	10.8406251
SP.0	8.718336	9.5509980
SP.1	7.197196	10.3968875
SP.2	6.977543	9.0108094
SP.3	8.020935	6.0801276
SP.4	6.507917	5.8401663
SP.5	9.193149	16.5469901
SP.6	7.534453	18.1994268
SP.7	9.963231	12.4203874
VP.0	9.521502	52.2798387
VP.1	9.742398	116.3594634
VP.2	6.931378	55.1977338
VP.3	7.544800	129.1207623
VP.4	8.012910	97.0629591
VP.5	8.744393	230.6464134
VP.6	9.925704	249.6642763
VP.7	9.664156	284.4284084
SPC.4	7.568168	6.0780245
SPC.5	7.746402	5.8732915
SPC.6	6.497093	6.0845875
VPC.4	9.675612	7.0604132
VPC.5	9.031271	6.9274184
VPC.6	5.153514	8.1337306
SC.3	4.959978	5.3501479
VC.3	7.511479	6.4169240
C1SP2	6.647693	3.7891626
C2SP2	5.961012	11.8720679

C3SP2	5.191587	2.6508130
C1SP3	5.535018	2.5614221
C2SP3	4.048156	3.5173133
C3SP3	1.850553	0.8011127
ATSp1	9.829810	77.6642669
ATSp2	8.991917	32.1574560
ATSp3	8.938140	22.7368503
ATSp4	8.253439	13.7487466
ATSp5	6.809434	15.5723324
ATSm1	11.674847	22.2915045
ATSm2	9.830247	13.9109607
ATSm3	9.951704	9.1649665
ATSm4	9.232974	6.5587757
ATSm5	8.228755	11.2240921
XLogP	45.352201	1412.5450461
MW	12.379746	32.8400736
nRotB	7.132464	6.7951013
nAtomLAC	5.681571	7.4097657
nAtomP	7.968388	7.3893988
nAtomLC	4.884425	6.0807814
nB	1.925739	5.2622014
nAtom	7.523156	7.8637587
nAromBond	5.258834	2.9244147
naAromAtom	3.654752	4.3605863
ALogP	11.782556	148.2328989
ALogp2	15.891321	172.6117440
AMR	8.383585	155.7330857
BCUTw.1h	11.948901	26.1632375
BCUTp.1l	6.837615	13.8892340
BCUTp.1h	8.976050	35.3436566

Control

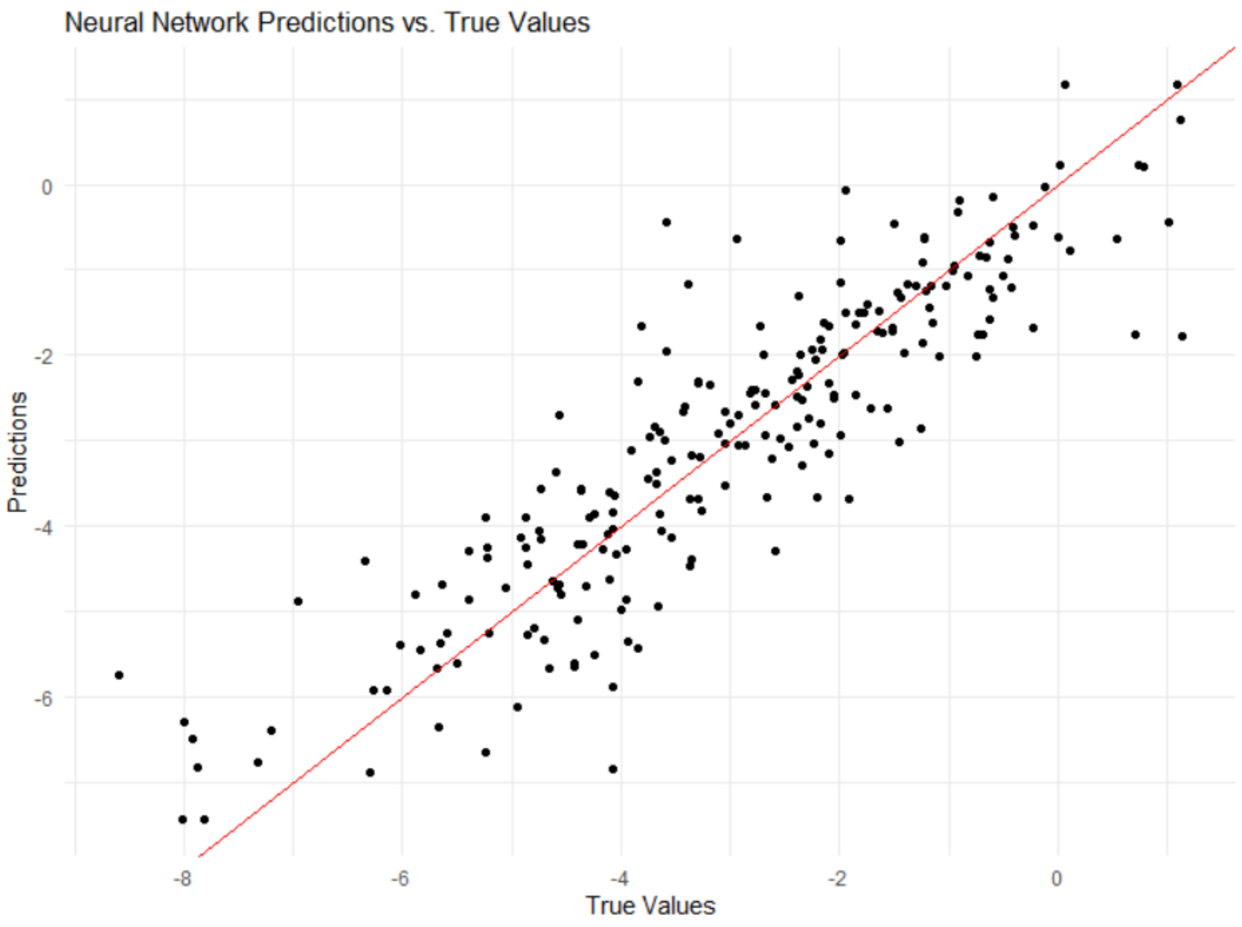
```
> ls()

 [1] "cv_control"          "dataset"             "dataset_clean"       "lm_LogP"
"lm_model_XLogP"       "lm_MW"               "lm_TopoPSA"

 [8] "model_lm1"           "model_lm2"           "model_performance"   "model_rf"
"plot_data_LogP"       "plot_data_MW"        "plot_data_TopoPSA"

[15] "pred_rf"             "predictions_lm_XLogP" "predictions_LogP"    "predictions_MW"
"predictions_TopoPSA"  "rf_cv_model"         "rmse_lm_XLogP"

[22] "rmse_rf"             "train_data"          "train_index"         "val_data"
```

Neutral Networks

```
library(nnet)

library(caret)

set.seed(123)

nn_model <- nnet(response ~ ., data = train_data, size = 5, linout = TRUE, maxit = 100)

print(nn_model)

a 98-5-1 network with 501 weights

inputs: TopoPSA nHBDon nHBAcc bpol apol nSmallRings nAromRings nRingBlocks nAromBlocks nRings6
Zagreb WPATH WPOL WTPT.1 WTPT.3 WTPT.4 WTPT.5 VAdjMat MDEC.12 MDEC.13 MDEC.14 MDEC.22
MDEC.23 MDEC.24 MDEC.33 MDEC.34 khs.sCH3 khs.ssCH2 khs.aaCH khs.sssCH khs.dssC khs.aasC khs.aaaC
khs.ssssC khs.sOH khs.dO khs.ssO khs.sCl Kier1 Kier2 fragC ECCEN SP.0 SP.1 SP.2 SP.3 SP.4 SP.5 SP.6 SP.7 VP.0
VP.1 VP.2 VP.3 VP.4 VP.5 VP.6 VP.7 SPC.4 SPC.5 SPC.6 VPC.4 VPC.5 VPC.6 SC.3 VC.3 C1SP2 C2SP2 C3SP2
C1SP3 C2SP3 C3SP3 ATSp1 ATSp2 ATSp3 ATSp4 ATSp5 ATSm1 ATSm2 ATSm3 ATSm4 ATSm5 XLogP MW
nRotB nAtomLAC nAtomP nAtomLC nB nAtom nAromBond naAromAtom ALogP ALogp2 AMR BCUTw.1h
BCUTp.1l BCUTp.1h

output(s): response

options were - linear output units

nn_predictions <- predict(nn_model, newdata = val_data)

nn_rmse <- sqrt(mean((val_data$response - nn_predictions)^2))

nn_mae <- mean(abs(val_data$response - nn_predictions))

nn_r_squared <- cor(nn_predictions, val_data$response)^2

cat("Neural Network RMSE:", nn_rmse, "\n")

Neural Network RMSE: 1.48982

cat("Neural Network MAE:", nn_mae, "\n")

Neural Network MAE: 1.156846

cat("Neural Network R-squared:", nn_r_squared, "\n")

Neural Network R-squared: 0.429185

cv_control <- trainControl(method = "cv", number = 5)

nn_cv_model <- train(response ~ ., data = train_data, method = "nnet",

                          trControl = cv_control, linout = TRUE, tuneLength = 5)
```

```
final  value 1918.040100
stopped after 100 iterations
print(nn_cv_model)
Neural Network
914 samples
 98 predictor
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 732, 731, 730, 731, 732
Resampling results across tuning parameters:

  size  decay   RMSE      Rsquared   MAE
  1     0e+00   1.941116  0.3874982  1.526077
  1     1e-04   1.946414  0.2710867  1.541985
  1     1e-03   1.986251  0.3124709  1.572986
  1     1e-02   1.963913  0.2385563  1.538849
  1     1e-01   1.728175  0.3399656  1.368041
  3     0e+00   1.910538  0.3106428  1.518703
  3     1e-04   1.771041  0.3834286  1.382590
  3     1e-03   1.729736  0.3490883  1.354240
  3     1e-02   1.620201  0.4208771  1.295966
  3     1e-01   1.589614  0.4396036  1.255601
  5     0e+00   1.824605  0.4287592  1.434948
  5     1e-04   1.667150  0.3820419  1.328670
  5     1e-03   1.600180  0.4378320  1.272597
  5     1e-02   1.614554  0.4167933  1.283874
  5     1e-01   1.581845  0.4537418  1.256152
  7     0e+00   1.559800  0.4642798  1.231736
  7     1e-04   1.734555  0.3397301  1.366796
  7     1e-03   1.667938  0.3972318  1.331742
  7      1e-02   1.679355   0.3964291   1.309352
  7      1e-01   1.594073   0.4431566   1.238746
  9      0e+00   1.526202   0.4828329   1.195854
  9      1e-04   1.567604   0.4666702   1.221465
  9      1e-03   1.472076   0.5333091   1.147110
  9      1e-02   1.637735   0.4476296   1.237748
  9      1e-01   1.518417   0.4968250   1.218054

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 9 and decay = 0.001.
```

```
# Plot
ggplot(results_df, aes(x = TrueValues, y = Predictions)) +
   geom_point() +
   geom_abline(intercept = 0, slope = 1, color = "red") +
   labs(
     title = "Neural Network Predictions vs. True Values",
     x = "True Values",
     y = "Predictions"
   ) +
   theme_minimal()
```


Neural Network Predictions vs. True Values
Predictions
True Values
0
-2
-4
-6
-8
-6
-4
-2
0