Examining nonlinear methods for toxicity prediction of carboxylic acids

Abdelsalam A

The findings of *Improved QSAR Analysis of the Toxicity of Aliphatic Carboxylic Acids* found improvements in predicting aquatic toxicity of carboxylic acids by using a particular set of molecular and topological parameters. The statistical methods employed were linear, and polynomial in nature which relied upon removing predictors. We attempt to establish a stronger predictive relationship using purely nonlinear methods while retaining all predictors.

The dataset can be found at: *Toxicity of Carboxylic Acids Data*

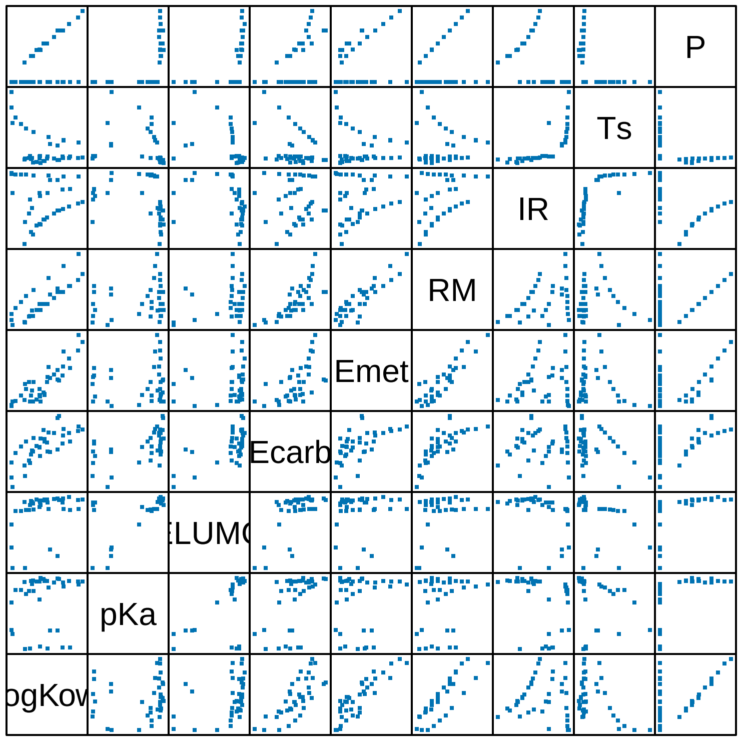
Firstly, we take a peak at the dataset:

```r
head(toxicity)
```

```
##   toxicity logKow  pKa ELUMO   Ecarb   Emet    RM    IR   Ts     P
## 1    -0.15   1.68 1.00  4.81 17.8635 1.4838 31.36 1.425 31.3 12.43
## 2    -0.33   0.94 0.98  4.68 16.9491 0.0000 22.10 1.408 30.4  8.76
## 3    -0.34   1.16 0.96  4.86 17.1806 0.2778 26.73 1.418 30.9 10.59
## 4     0.03   2.75 1.00  4.83 18.4794 3.5836 40.63 1.435 31.8 16.10
## 5    -0.57   0.79 0.97  4.80 16.8022 1.0232 22.14 1.411 32.5  8.77
## 6     0.08   2.64 1.01  4.90 18.3937 3.7145 40.63 1.435 31.8 16.10
```

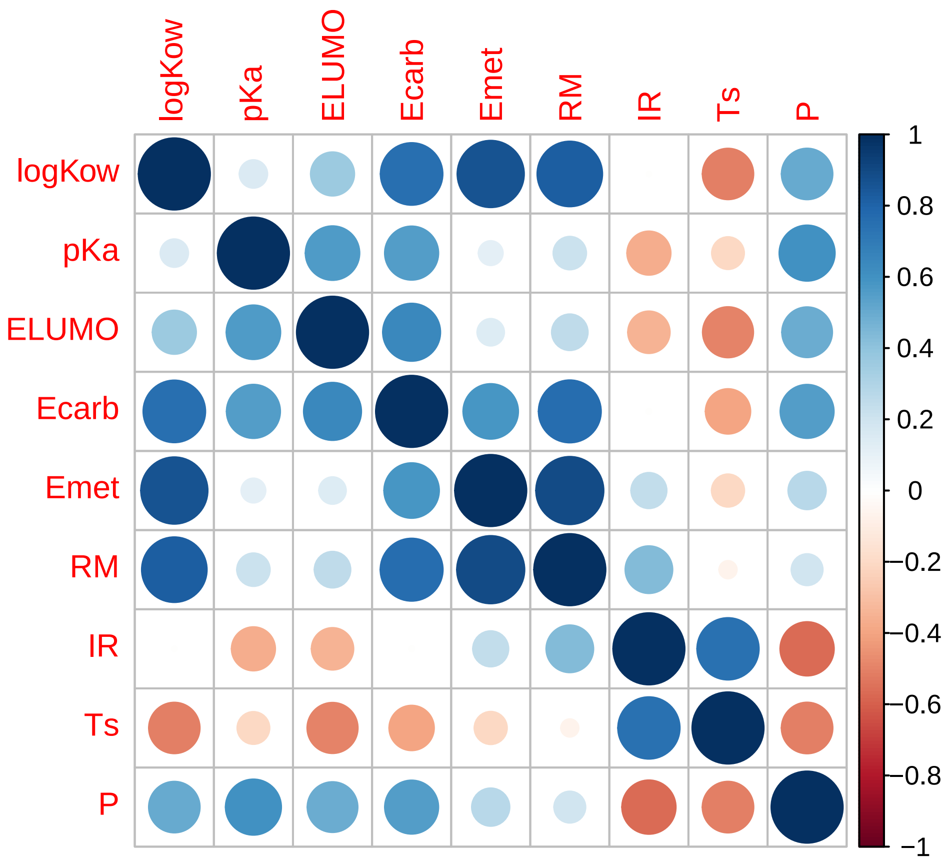
We want to examine the pairwise relationships of features, so we pre-process via centering, scaling, followed by plotting the scatter-plot matrix, and correlation matrix

```r
# Centering & Scaling
scaledToxicity <- preProcess(toxicity[, -1], method = c("center", "scale"))
csData <- predict(scaledToxicity, newdata = toxicity[,-1])
# Scatterplot Matrix
splom(~csData, pch = 15, cex = .25, pscales = 0)
```



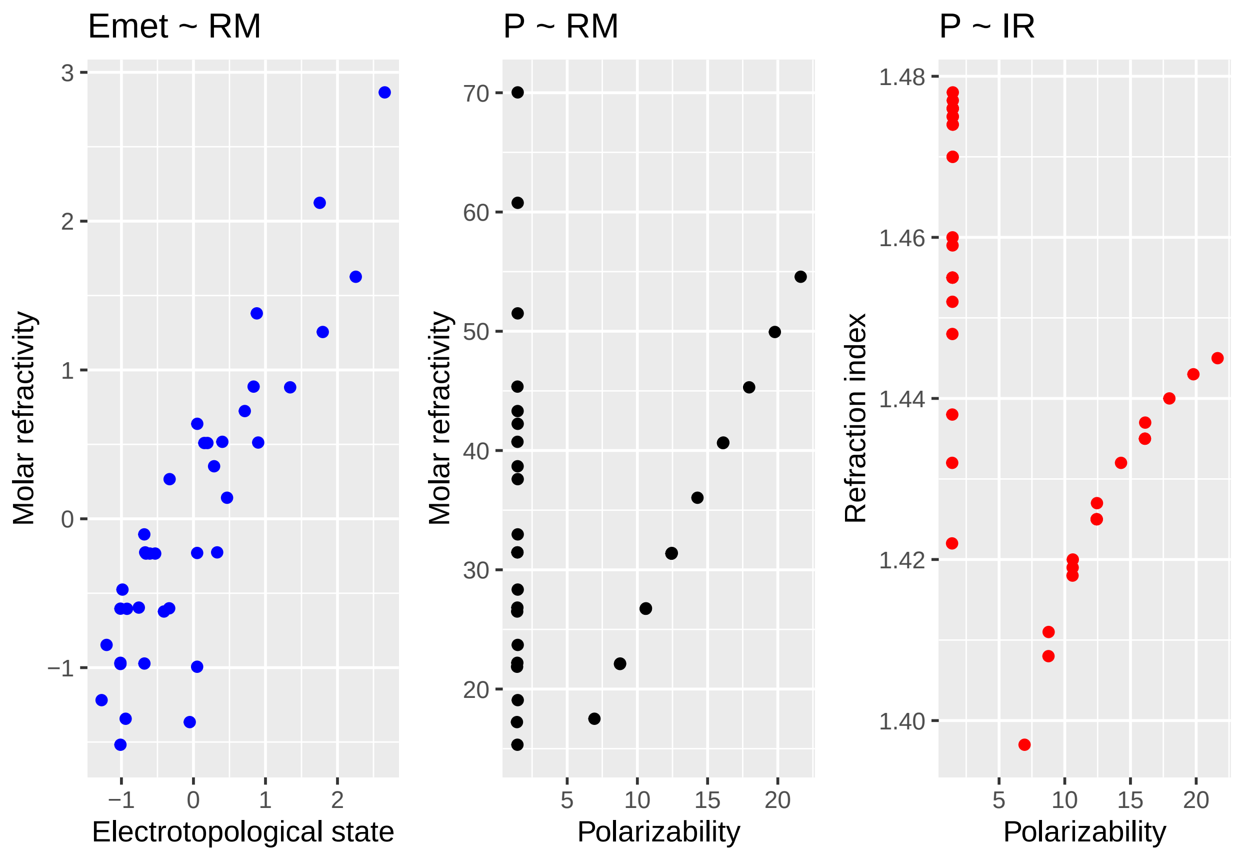
Scatter Plot Matrix

```
# Correlation Matrix
corrplot(cor(csData))
```



We observe a few select heavily-correlated non-linear features, such as **RM** ~ **Emet**, some with a mix of both linear and uncorrelated points, like **P** ~ **RM**, and some with non-linear and uncorrelated points, like **P** ~ **IR**

```r
csData <- as.data.frame(csData)
plt1 <- ggplot(csData, aes(x=Emet, y = RM)) + geom_point(color = "blue") + labs(
    title = "Emet ~ RM",
    x = "Electrotopological state", y = "Molar refractivity",
  )
plt2 <- ggplot(toxicity, aes(x=P, y = RM)) + geom_point() + labs(
    title = "P ~ RM",
    x = "Polarizability", y = "Molar refractivity",
  )
plt3 <- ggplot(toxicity, aes(x=P, y = IR)) + geom_point(color = "red") + labs(
    title = "P ~ IR",
    x = "Polarizability", y = "Refraction index",
  )
ggarrange(plt1, plt2, plt3, ncol = 3)
```

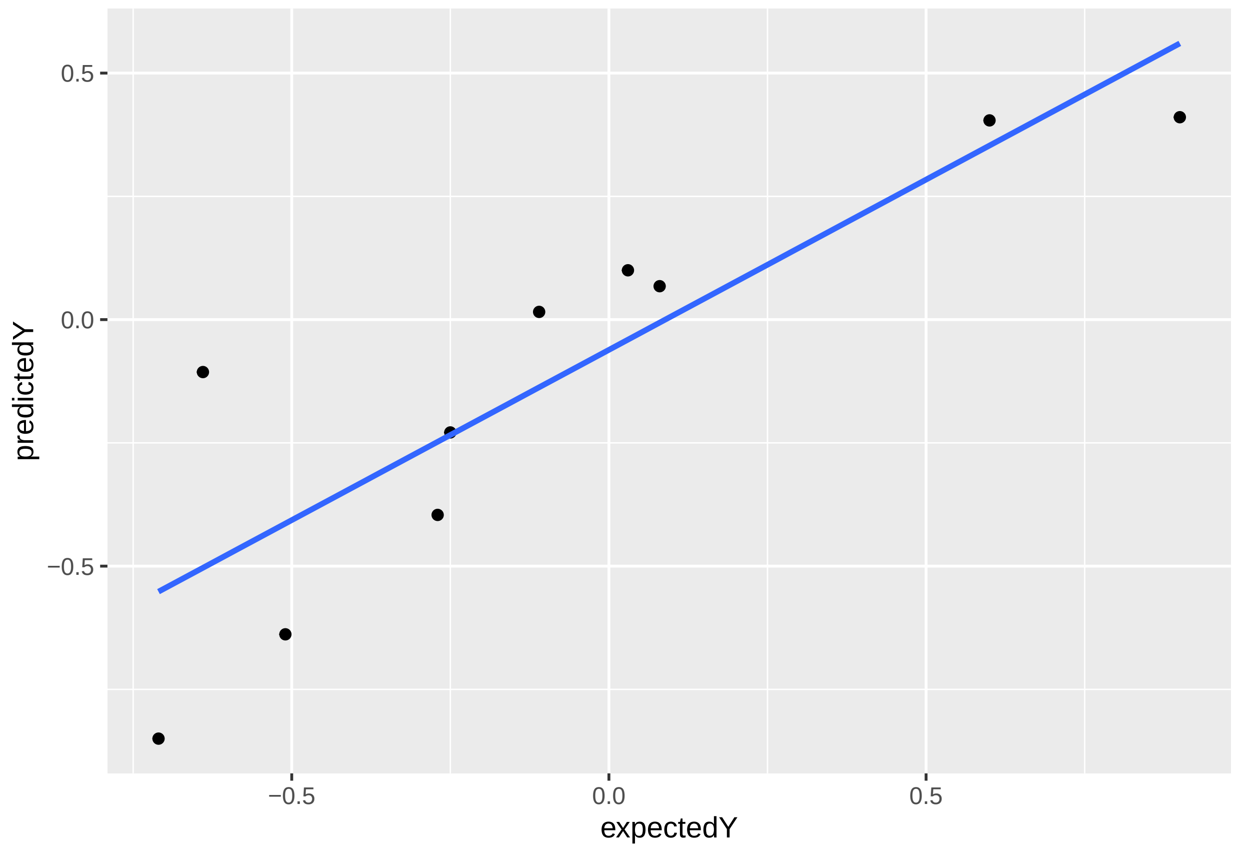


We see that some feature-feature relationships occur only after a certain threshold. For example, as polarizability increases after ~6.5, molar refractivity increases with it linearly. A model that captures these sudden pattern changes is MARS.

Multivariate Adaptive Regression Splines (MARS)

```
set.seed(1)
# Make training and test sets
trainingRows <- createDataPartition(toxicity[,1], p = 0.7, list = FALSE)
# Training set
trainSet <- toxicity[trainingRows,]
# Testing set
testSet <- toxicity[-trainingRows,]
# 1-3 degree interactions
hyper_grid <- expand.grid(
  degree = 1:3,
  nprune = seq(2, 100, length.out = 10) %>% floor()
)

# MARS with 5-fold CV
marsModel <- train(
  toxicity ~ .,
  data = trainSet,
  method = "earth",
  metric = "RMSE",
  preProcess = c("BoxCox", "center", "scale"),
  trControl = trainControl(method = "cv", number = 5),
  tuneGrid = hyper_grid
)
# Plotting results
predictedValuesMars <- predict(marsModel, testSet)
predictedDataMars <- cbind(testSet[,1], predictedValuesMars)
colnames(predictedDataMars) <- c("expectedY", "predictedY")
ggplot(predictedDataMars, aes(x = expectedY, y = predictedY)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
# R2 value
caret::R2(predictedValuesMars, testSet[,1])
```

```
##              y
## [1,] 0.7462671
```

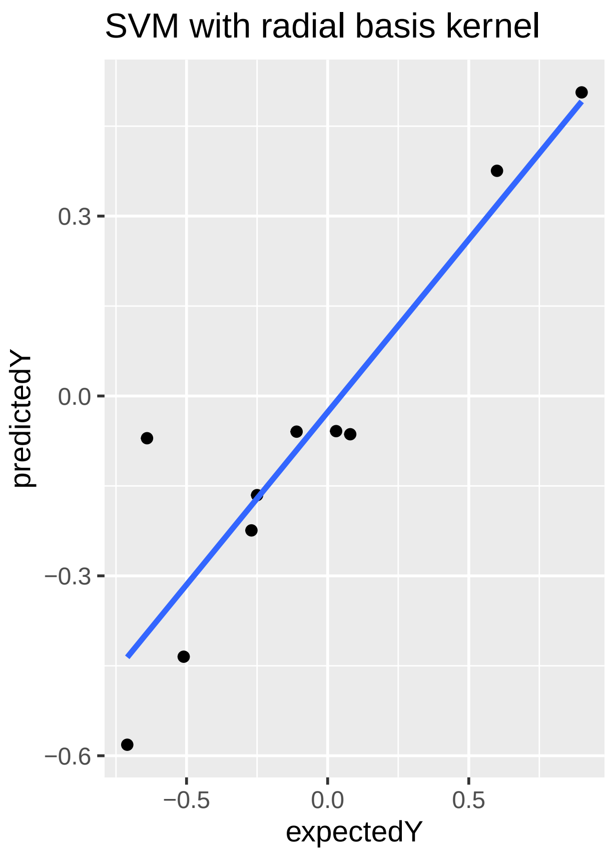
Unfortunately, MARS didn't perform particularly well with an R^2 of 0.7463.

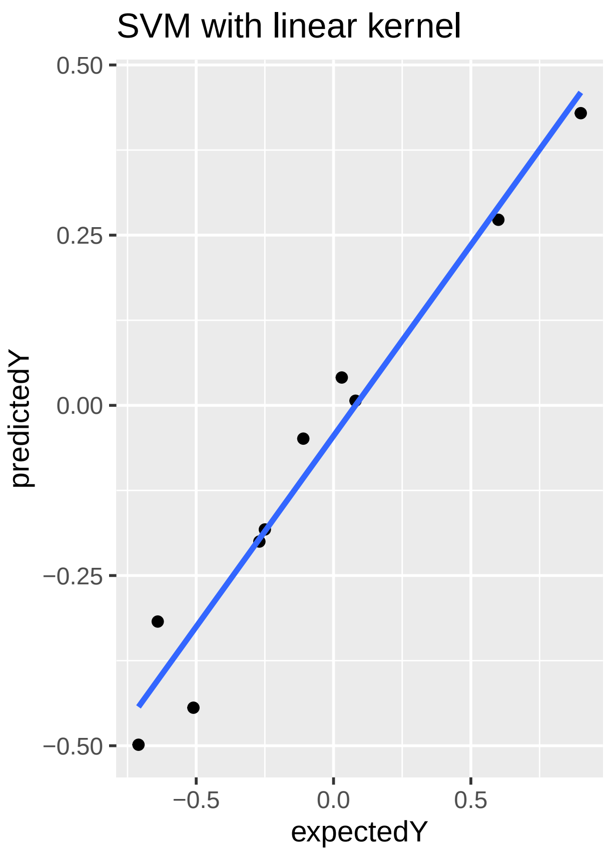
We follow with SVM; in particular, we try a radial basis, and linear kernel.

```
set.seed(1)
# Radial Basis kernel
svmRadialModel <- train(
  toxicity ~ .,
  data = trainSet,
```

```r
  method = "svmRadial",
  preProc = c("BoxCox", "center", "scale"),
  tuneLength = 14,
  trControl = trainControl(method = "cv"))
# Linear kernel
svmLinearModel <- train(
  toxicity ~ .,
  data = trainSet,
  method = "svmLinear",
  preProc = c("BoxCox", "center", "scale"),
  tuneLength = 14,
  trControl = trainControl(method = "cv"))

# Plotting results
predictedValuesRadial <- predict(svmRadialModel, testSet)
predictedDataRadial <- cbind(testSet[,1], predictedValuesRadial)
colnames(predictedDataRadial) <- c("expectedY", "predictedY")
pltSVMRadial <- ggplot(predictedDataRadial, aes(x = expectedY, y = predictedY)) +
     geom_point() +
     geom_smooth(method = "lm", se = FALSE) +
     labs(title = "SVM with radial basis kernel")

predictedValuesLinear <- predict(svmLinearModel, testSet)
predictedDataLinear <- cbind(testSet[,1], predictedValuesLinear)
colnames(predictedDataLinear) <- c("expectedY", "predictedY")
pltSVMLinear <-  ggplot(predictedDataLinear, aes(x = expectedY, y = predictedY)) +
     geom_point() +
     geom_smooth(method = "lm", se = FALSE) +
     labs(title = "SVM with linear kernel")

ggarrange(pltSVMRadial, pltSVMLinear, ncol = 2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```





```
# R2 value of SVM Radial
caret::R2(predictedValuesRadial, testSet[,1])
```

```
## [1] 0.8377675
```

```
# R2 value of SVM Linear
caret::R2(predictedValuesLinear, testSet[,1])
```

```
## [1] 0.9586954
```

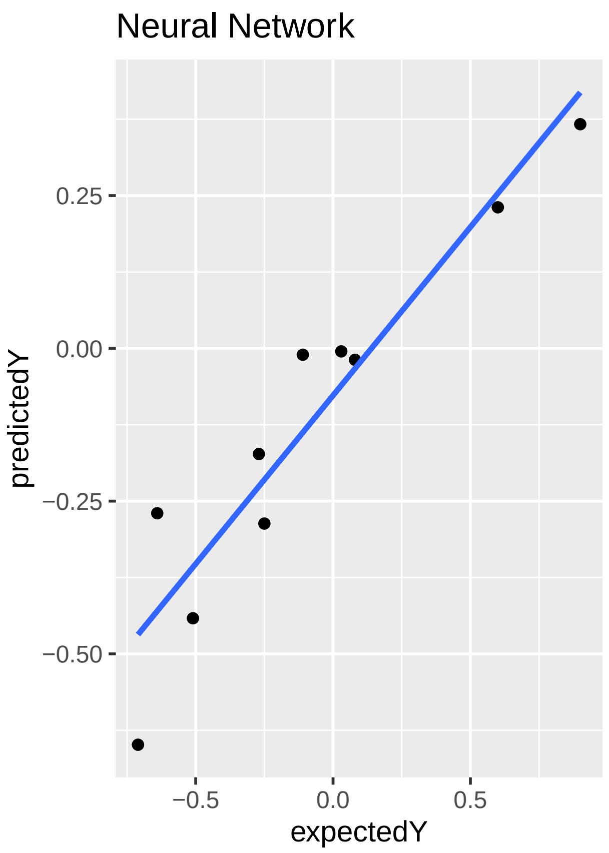
The radial basis kernel performed mediocre at $R^2 = 0.8377675$, while the linear kernel performed fairly well at $R^2 = 0.9586954$ which beats all but two polynomial models in the original study.

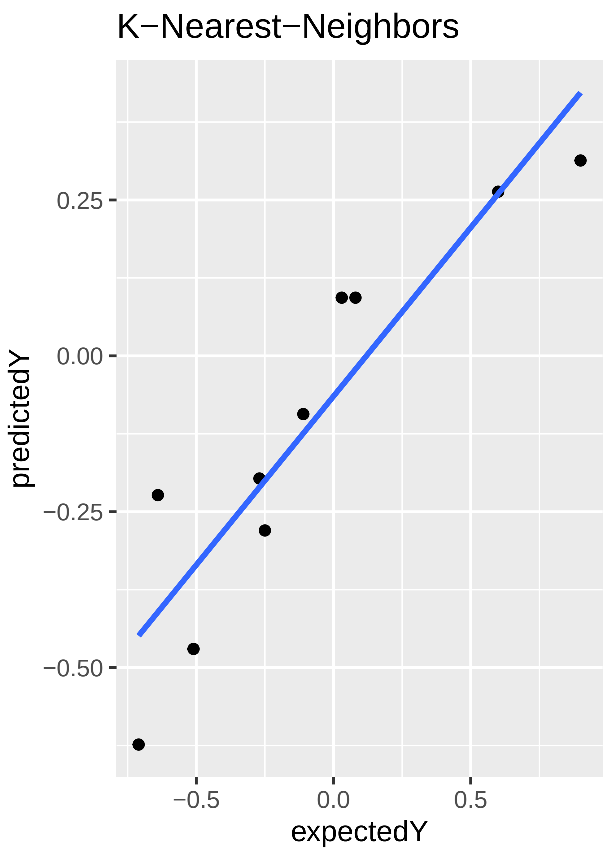
Finally, we attempt a neural network, and K-Nearest-neighbours.

For NN, we can't have our correlated features crippling the model, so we apply PCA for pre-processing.

```
set.seed(1)
# Neural Network with PCA
nnetGrid <- expand.grid(.decay = c(0, 0.01, .1),
    .size = c(1:10),
    .bag = FALSE)
nnetModel <- train(toxicity ~ .,
    data = trainSet,
    method = "nnet",
    tuningGrid = nnetGrid,
    trControl = trainControl(method = "cv"),
    preProc = c("center", "scale", "pca"),
    linout = TRUE,
    trace = FALSE,
    MaxNWts = 10 * (ncol(toxicity[,-1]) + 1) + 10 + 1,
    maxit = 500)

#K-Nearest-Neighbors

 knnModel <- train(toxicity ~ .,
    data = trainSet,
    method = "knn",
    preProc = c("center", "scale"),
    tuneGrid = data.frame(.k = 1:20),
    trControl = trainControl(method = "cv"))

# Generating predictions - NN
predictedValuesNNet <- predict(nnetModel, testSet)
predictedDataNNet <- cbind(testSet[,1], predictedValuesNNet)
colnames(predictedDataNNet) <- c("expectedY", "predictedY")
pltNNet <-  ggplot(predictedDataNNet, aes(x = expectedY, y = predictedY)) +
     geom_point() +
     geom_smooth(method = "lm", se = FALSE) +
     labs(title = "Neural Network")

# Generating predictions - KNN
predictedValuesKNN <- predict(knnModel, testSet)
predictedDataKNN <- cbind(testSet[,1], predictedValuesKNN)
colnames(predictedDataKNN) <- c("expectedY", "predictedY")
pltKNN <-  ggplot(predictedDataKNN, aes(x = expectedY, y = predictedY)) +
     geom_point() +
     geom_smooth(method = "lm", se = FALSE) +
     labs(title = "K-Nearest-Neighbors")

# Plotting results
ggarrange(pltNNet, pltKNN, ncol = 2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```





```
# R2 value of NNet
caret::R2(predictedValuesNNet, testSet[,1])
```

```
## [1] 0.8847985
```

```
# R2 value of KNN
caret::R2(predictedValuesKNN, testSet[,1])
```

```
## [1] 0.8399075
```

We observe that both NNet, and KNN perform fairly medicore with R^2 values of 0.8848 and 0.8399.

We conclude that the nonlinear methods examined in this analysis do not perform well enough to compete with the simpler, and better-performing linear methods described in the original study.