R Notebook

```
library(keras)
```

```
## Warning: package 'keras' was built under R version 3.5.3
```

```
use_python("D:/anaconda3/python.exe")
```

Identify the catrgorical variables. One-hot encode your lists to turn them into vectors of 0s and 1s.

```
library(MASS)
data(Boston)

#create new dummy variables
head(Boston)
```

	crim <dbl>	zn <dbl>	indus <dbl>	chas <int>	nox <dbl>	rm <dbl>	age <dbl>	dis <dbl>	rad <int>	▶
1	0.00632	18	2.31	0	0.538	6.575	65.2	4.0900	1	
2	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	
3	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	
4	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	
5	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	
6	0.02985	0	2.18	0	0.458	6.430	58.7	6.0622	3	

6 rows | 1-10 of 15 columns

```
#chase and rad
Boston$chas<- factor(Boston$chas)
Boston$chas<- factor(Boston$rad)
dummy<- model.matrix(~.,data=Boston)
data1<- dummy[,-1] #21 vairables
variables<- names(data.frame(data1))

xvariables<- dummy[,-c(1,21)]
yvariable<- dummy[,21]
```

Use the caret package to tune the parameters. one-hidden-layer neural network. Number of nodes in the hidden layer (size), the dropout rate (dropout), the training batch size (batch_size), the learning rate (lr) and the activation function.

The tuning results are listed below: number of nodes: 4 activation function: tanh batch size: 128 learning rate: 1e-06 dropout rate: 1

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

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

```
## Loading required package: ggplot2
```

```
#10-fold CV
set.seed(8888)
## 10-fold CV
caret_control <- trainControl(
                           method = "cv",
                           number = 10
                           )


#tunegrid,fix some of the tuning parameter while changing the others and choose the best

#tune number of nodes in the hidden layer and activation function at the same time
size_grid <- expand.grid(batch_size=64,
                          dropout=0.1,
                          size=1:20,
                          lr=0.00001,
                          rho=1,
                          decay=0,
                          activation = c("relu","sigmoid","tanh")
                          )


size_select <- train(medv ~., data = data1,
                 method = "mlpKerasDropout",
                 trControl = caret_control,
                 tuneGrid = size_grid,
                 verbose = FALSE,
                 metric="MSE"
                 )
```

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

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

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

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

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

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

```
#tune batch size & learning rate
batch_grid <- expand.grid(batch_size=c(32,64,128),
                          dropout=0.1,
                          size=4,
                          lr=c(0.000001,0.00001,0.0001, 0.001, 0.01, 0.1, 0.2, 0.3),
                          rho=1,
                          decay=0,
                          activation = "tanh"
                          )

batch_select <- train(medv ~ ., data = data1,
                 method = "mlpKerasDropout",
                 trControl = caret_control,
                 tuneGrid = batch_grid,
                 verbose = FALSE,
                 metric="MSE"
                 )
```

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

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

```
#tune dropout rate
dropout_grid <- expand.grid(batch_size=128,
                          dropout=seq(0,1,0.1),
                          size=4,
                          lr=1e-06,
                          rho=1,
                          decay=0,
                          activation = "tanh"
                          )

dropout_select <- train(medv ~ ., data = data1,
                 method = "mlpKerasDropout",
                 trControl = caret_control,
                 tuneGrid = dropout_grid,
                 verbose = FALSE,
                 metric="MSE"
                 )
```

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

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

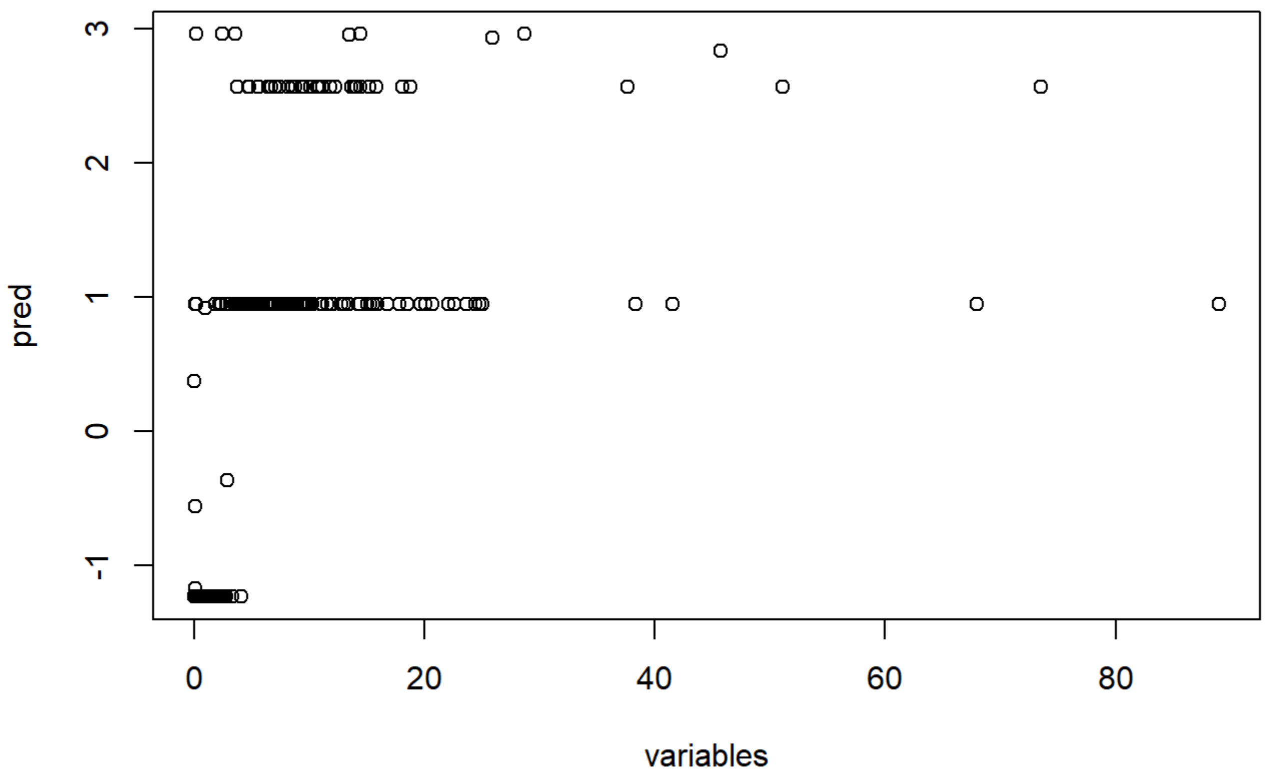
Fit the model with the best parameters using Keras.

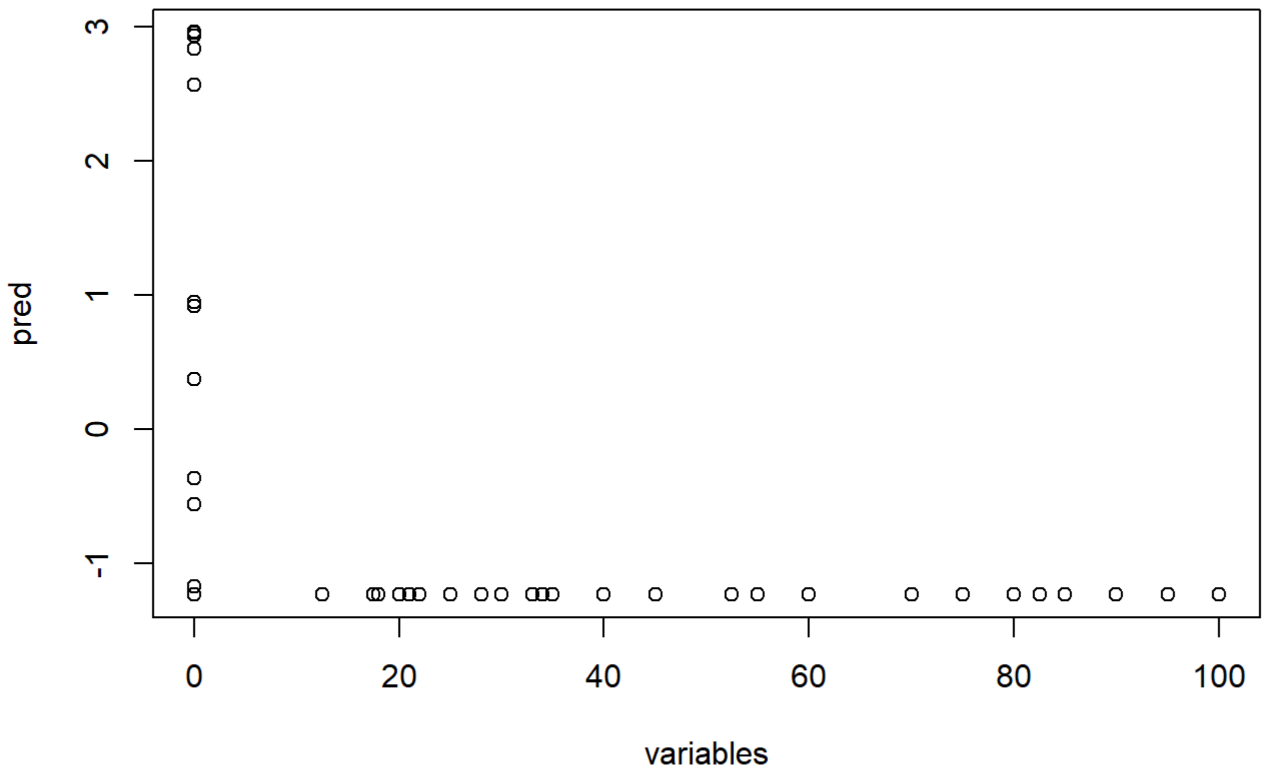
number of nodesin the layer: 4 activation function: tanh batch size: 128 learning rate: 1e-06 dropout rate: 1

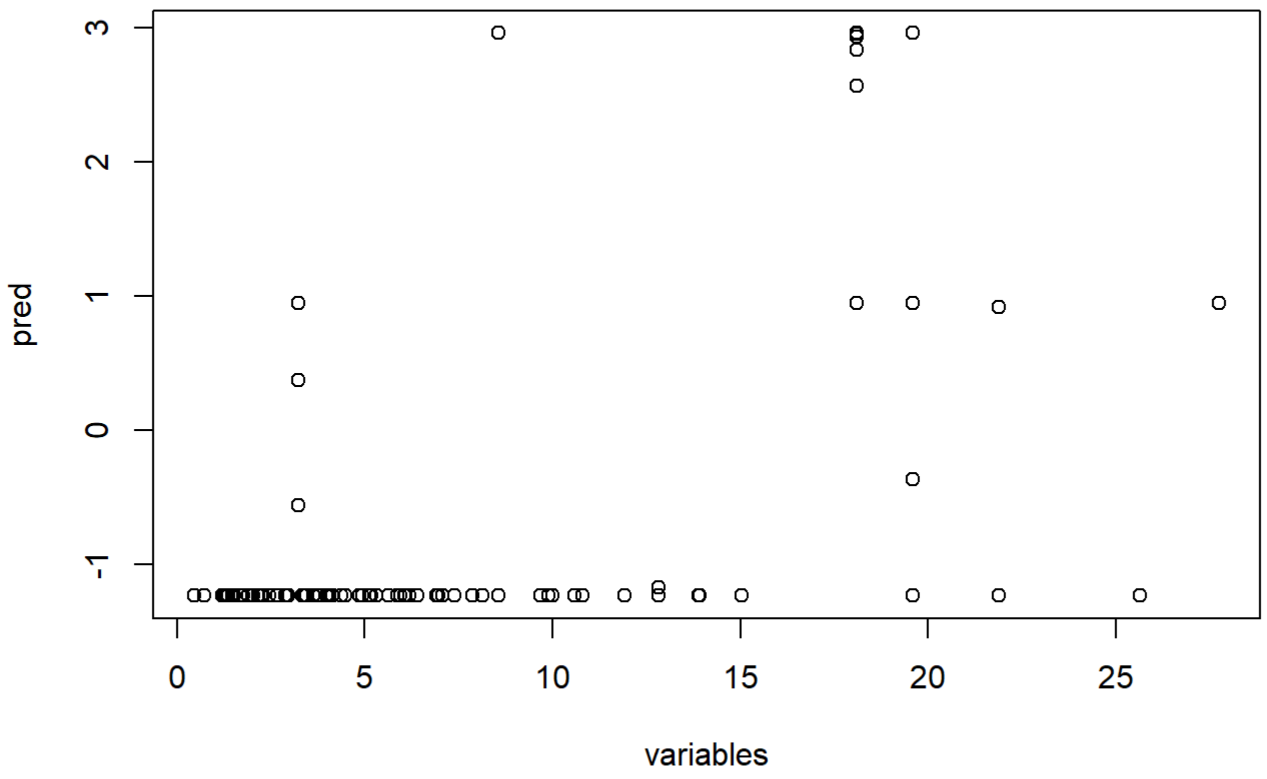
```
model <- keras_model_sequential() %>%
layer_dense(units = 4, activation = "tanh",                 input_shape = c(20)) %>%
  layer_dropout(rate = 1) %>%
  layer_dense(units = 1)
  model %>% compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(lr = 1e-06),
    metrics = list("mean_absolute_error")
    )

model %>% fit(
  xvariables,
  yvariable,
  epochs = 100,
  batch_size = 128,
  validation_split = 0.2,
  verbose = 0 )
```

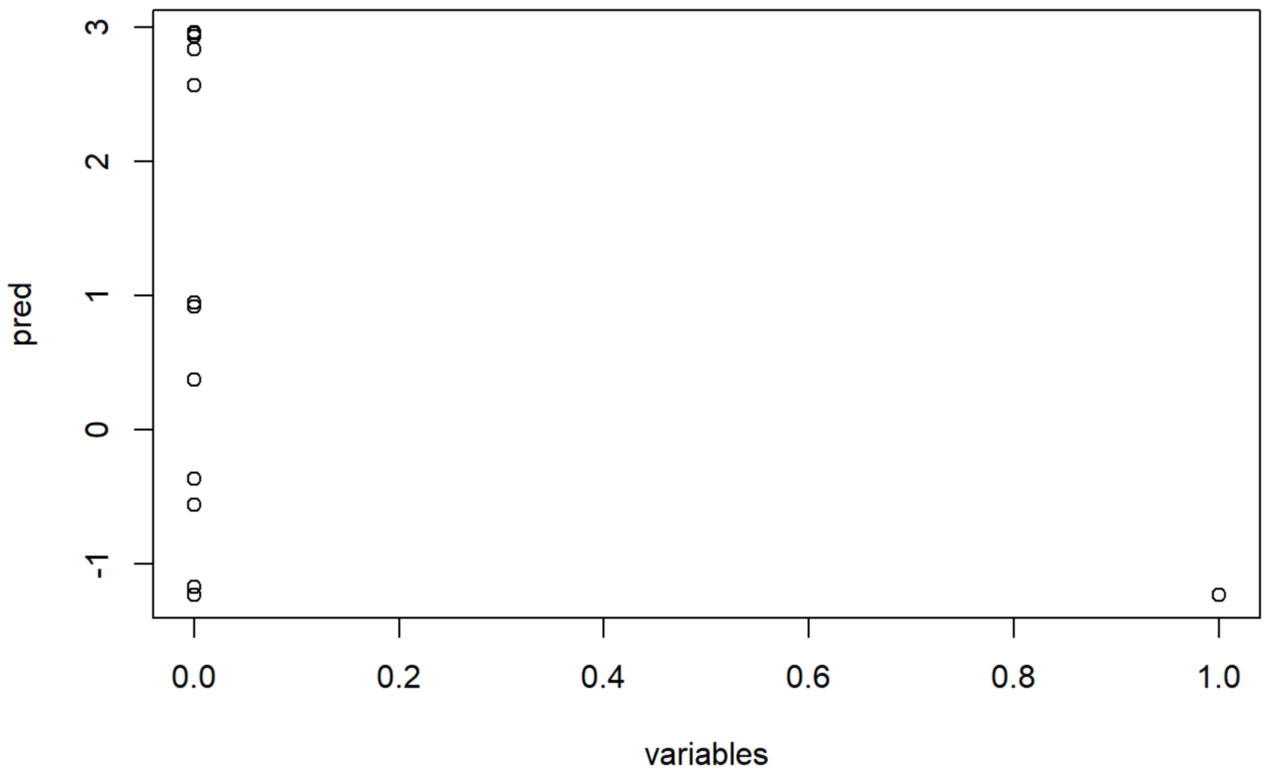
Obtain the predictions.Plot the prediction with respect to each variable. My preditction is kind of wierd here since the value is negative. Which variables seem to have on-linear effects? The graph shows that most of the variables have non-linear effetcs.

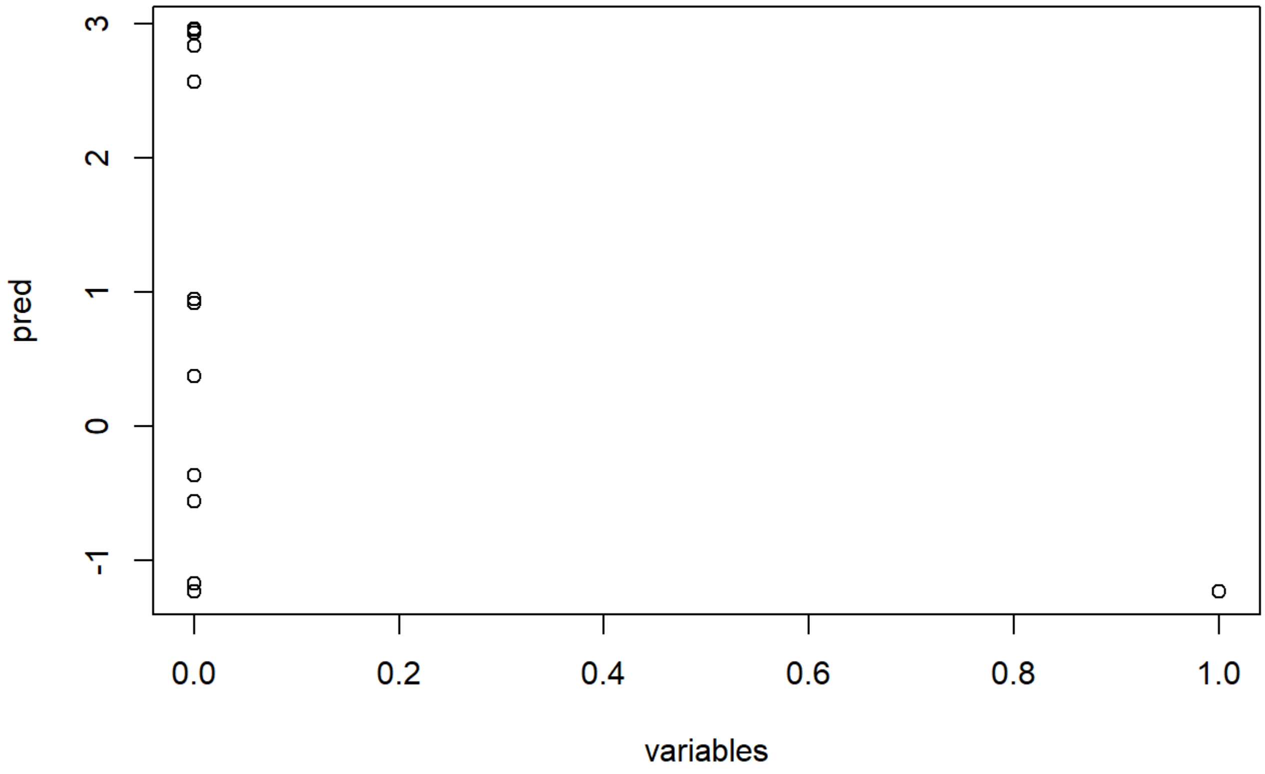
```r
predictions <- model %>% predict(xvariables)
pred<- predictions[,1]
count=0
for( variables in data.frame(data1)){
  names<- names(data)
  count<- count+1
  variables_name=names[count]
  plot(variables,pred,xlab=variables_name)
  }
```

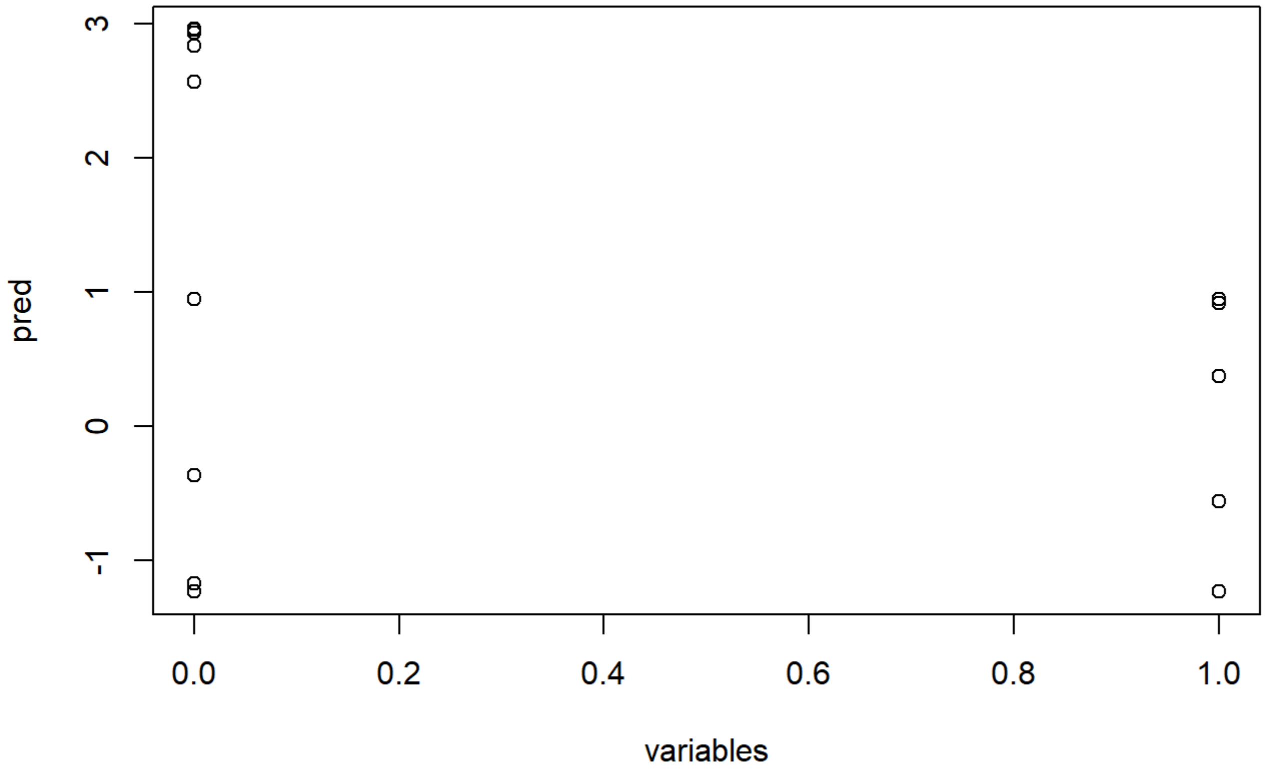

pred
-1
0
1
2
3
0
20
40
60
80
variables

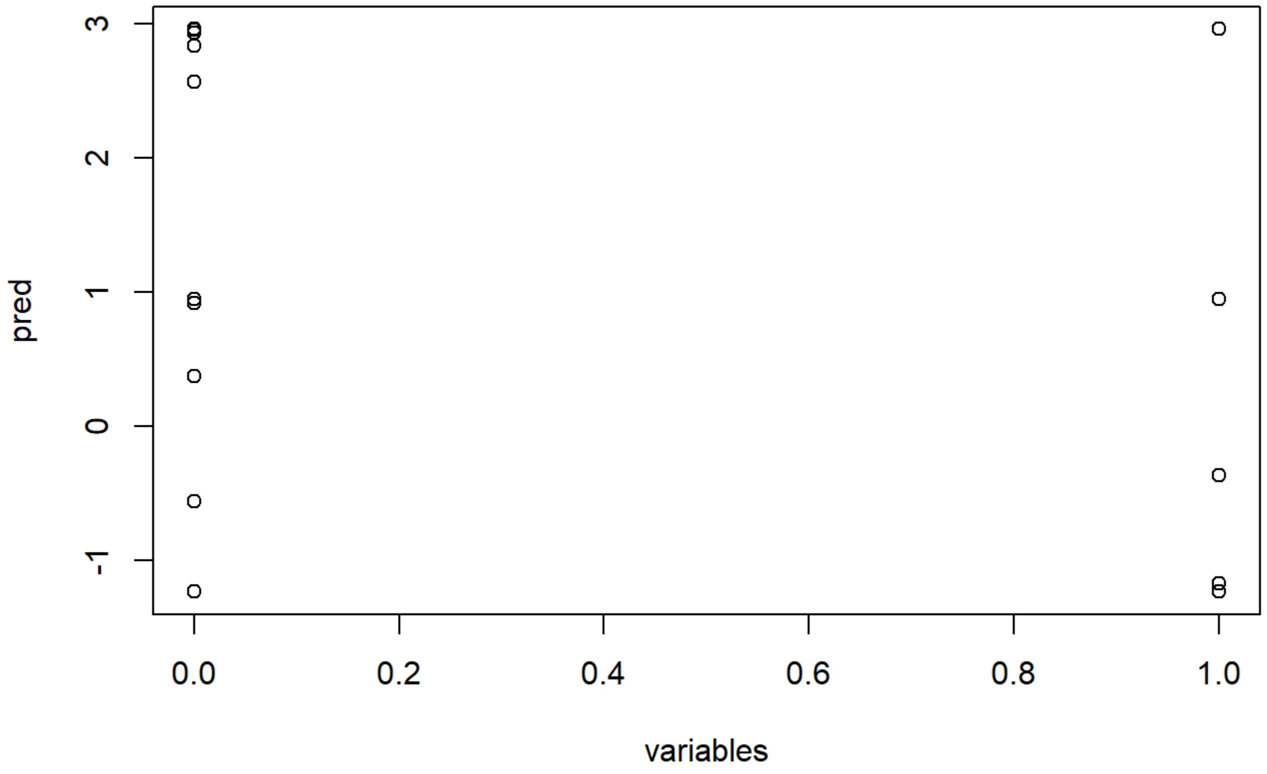

pred
-1
0
1
2
3
0
20
40
60
80
100
variables

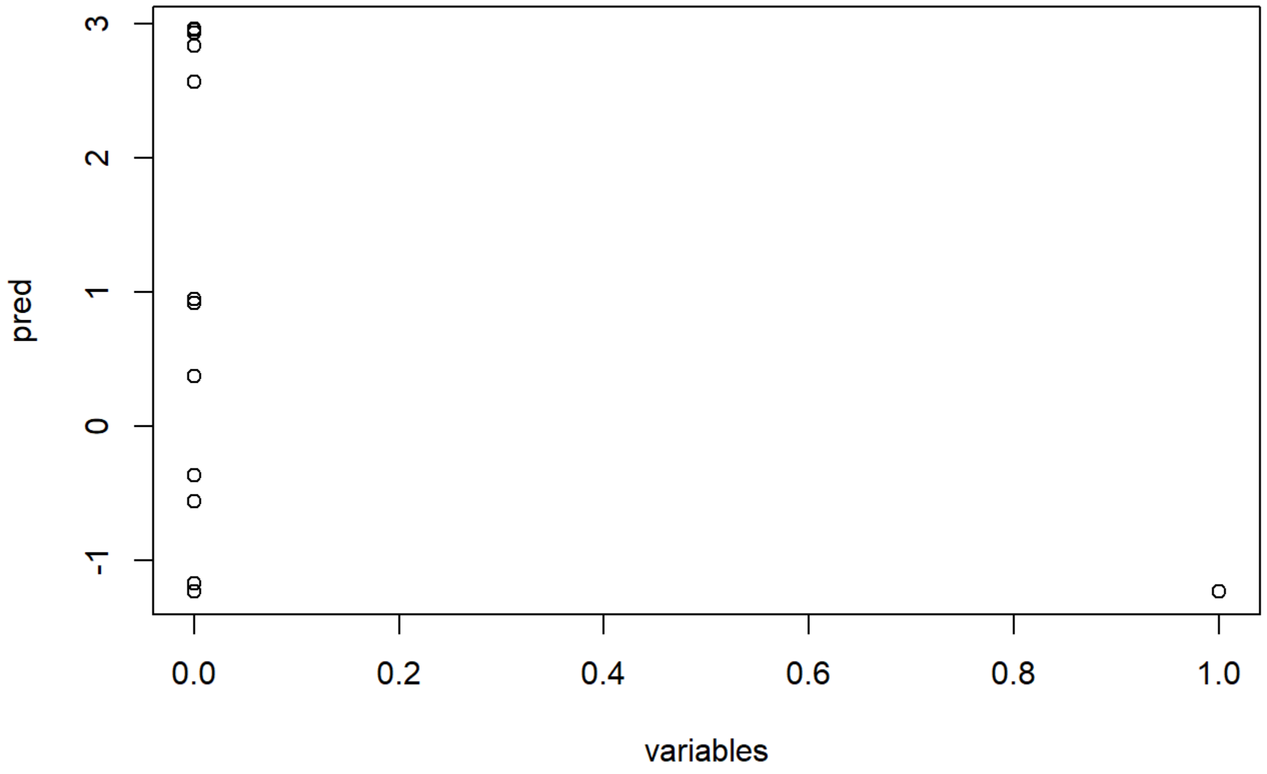

pred
-1
0
1
2
3
0
5
10
15
20
25
variables

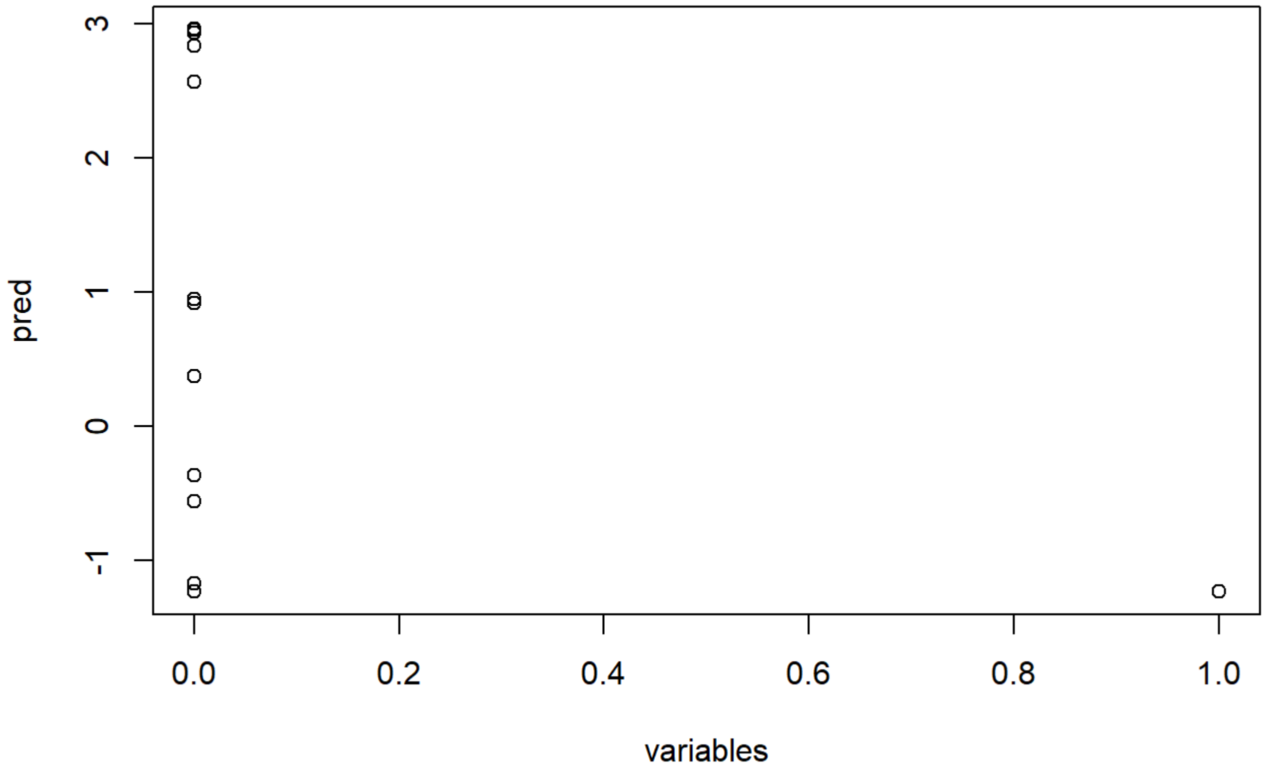

pred
-1
0
1
2
3
0.0
0.2
0.4
0.6
0.8
1.0
variables

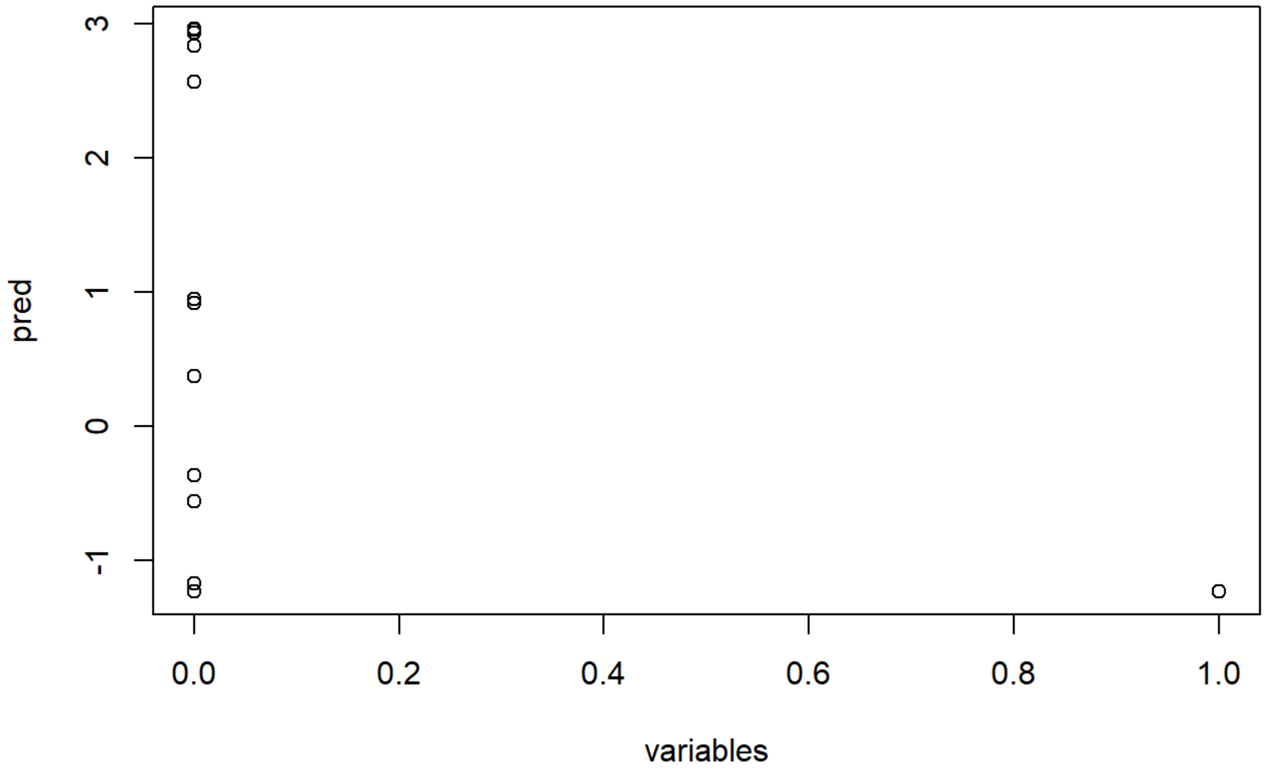

pred
variables
3
2
1
0
-1
0.0
0.2
0.4
0.6
0.8
1.0

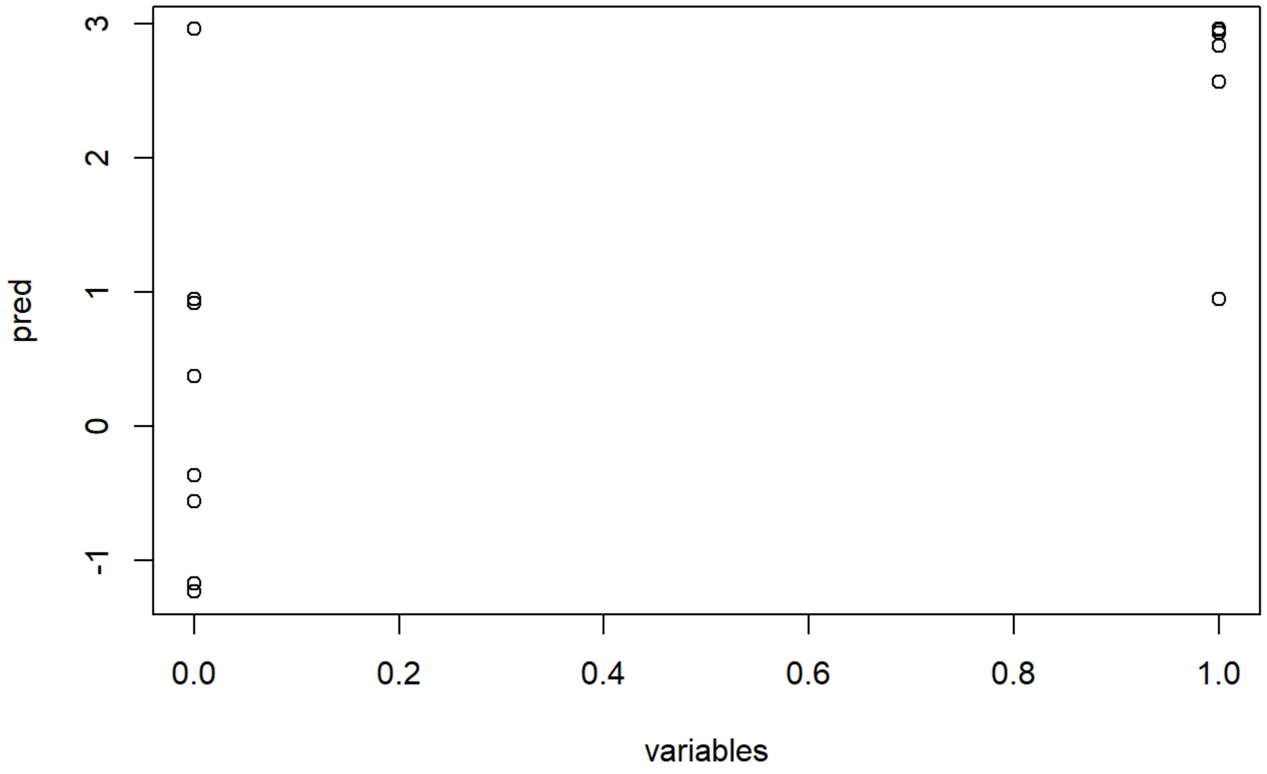

pred
variables
3
2
1
0
-1
0.0
0.2
0.4
0.6
0.8
1.0

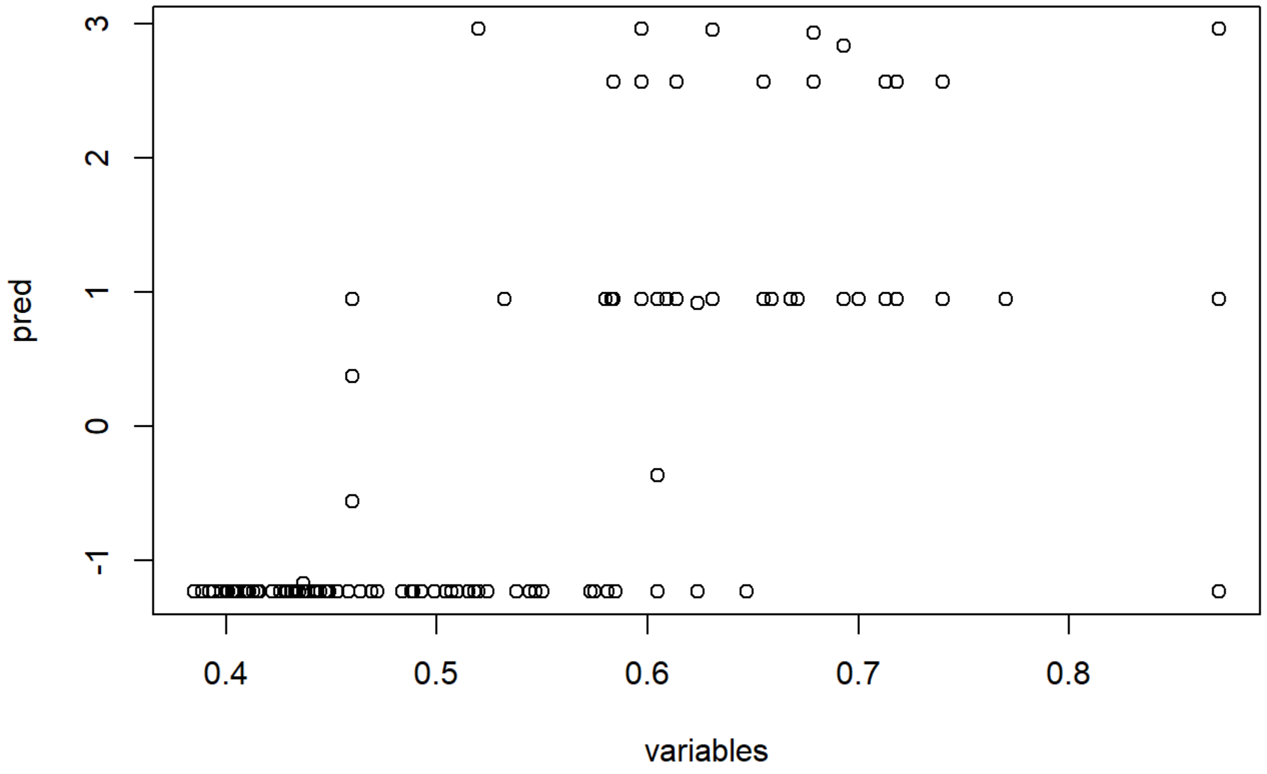

pred
-1
0
1
2
3
0.0
0.2
0.4
0.6
0.8
1.0
variables

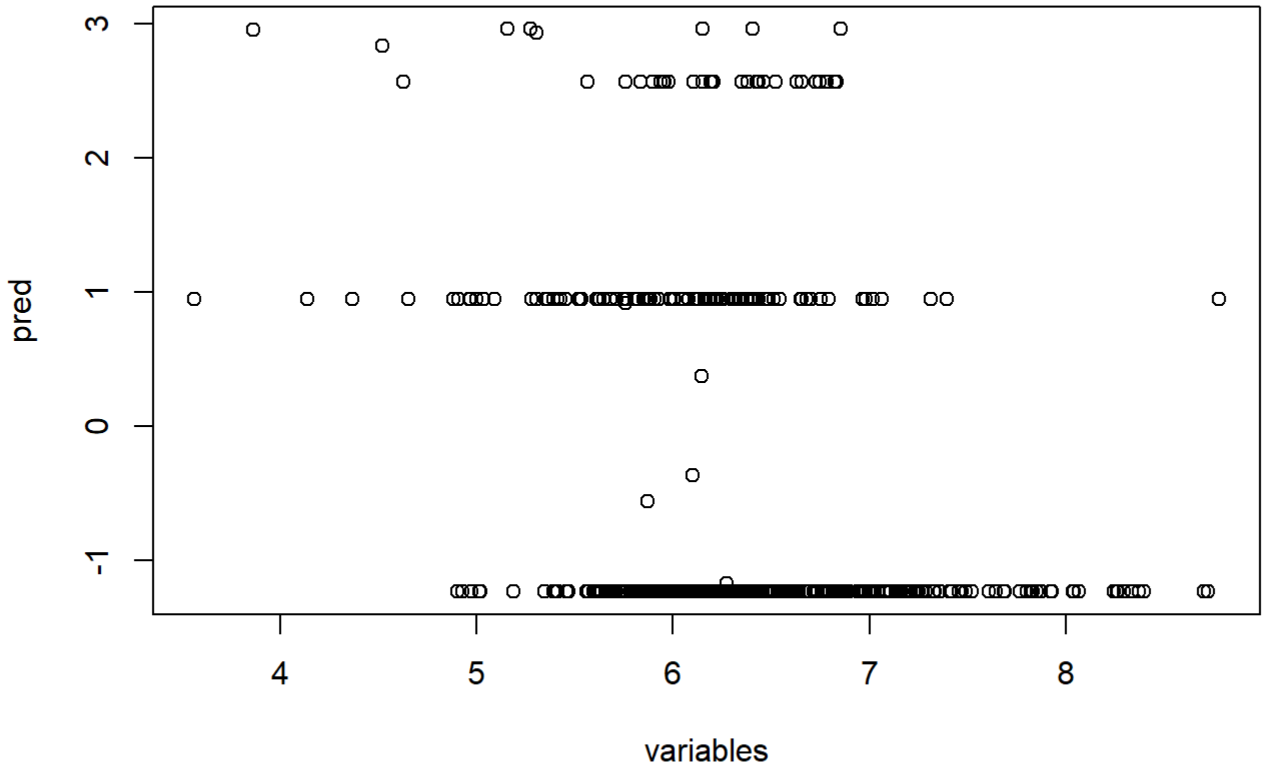

pred
-1
0
1
2
3
0.0
0.2
0.4
0.6
0.8
1.0
variables

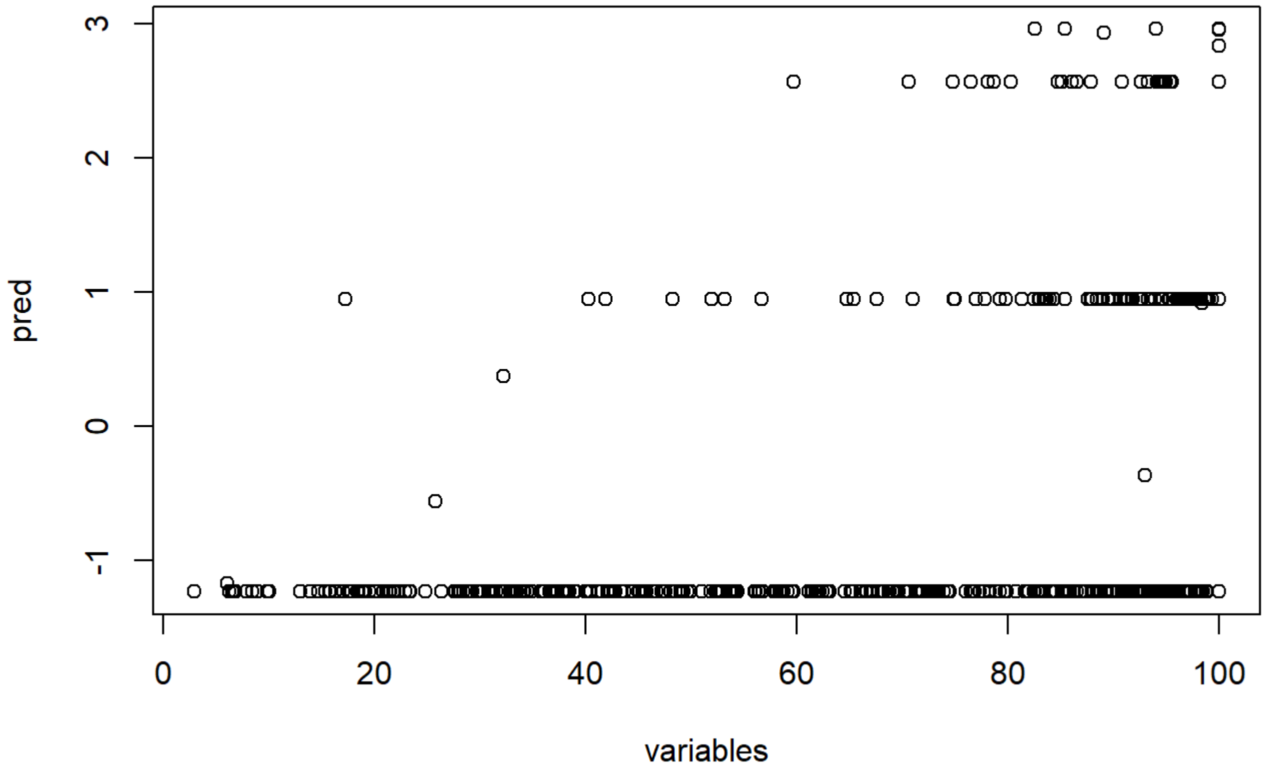

pred
3
2
1
0
-1
0.0
0.2
0.4
0.6
0.8
1.0
variables

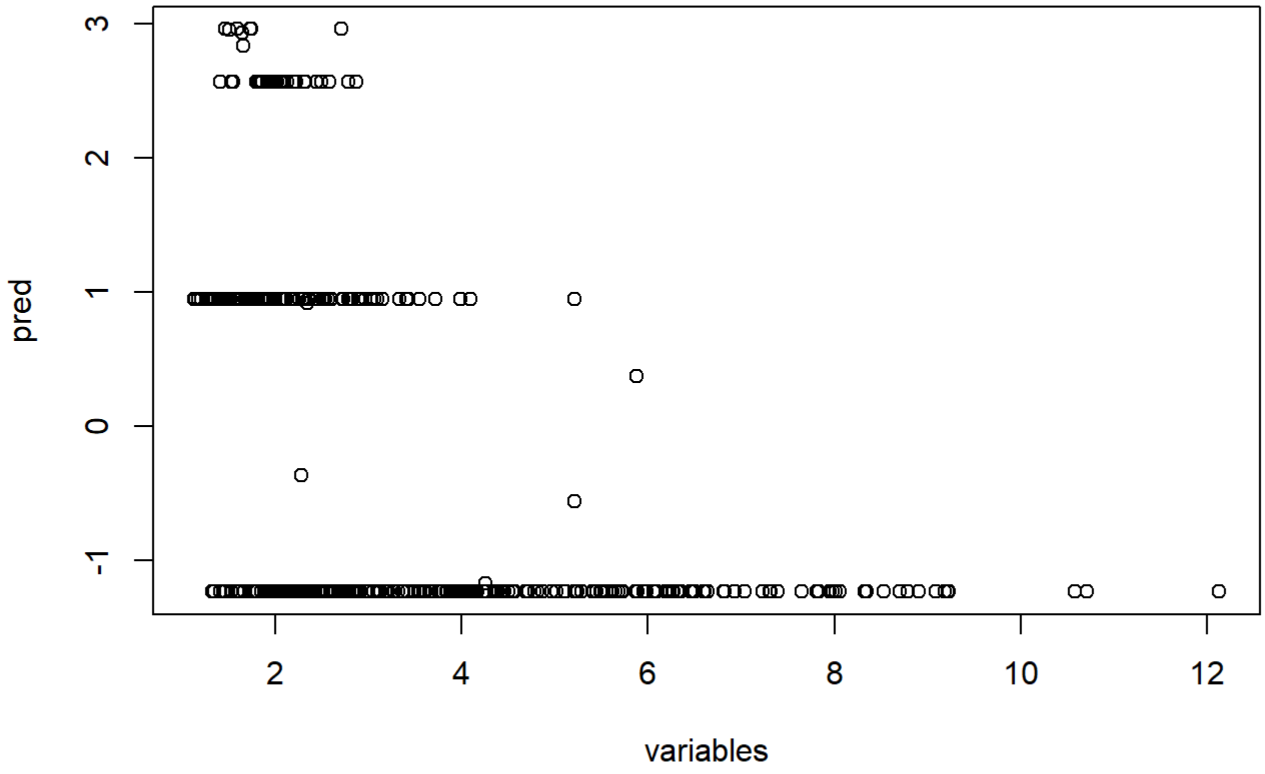

pred
3
2
1
0
-1
0.0
0.2
0.4
0.6
0.8
1.0
variables

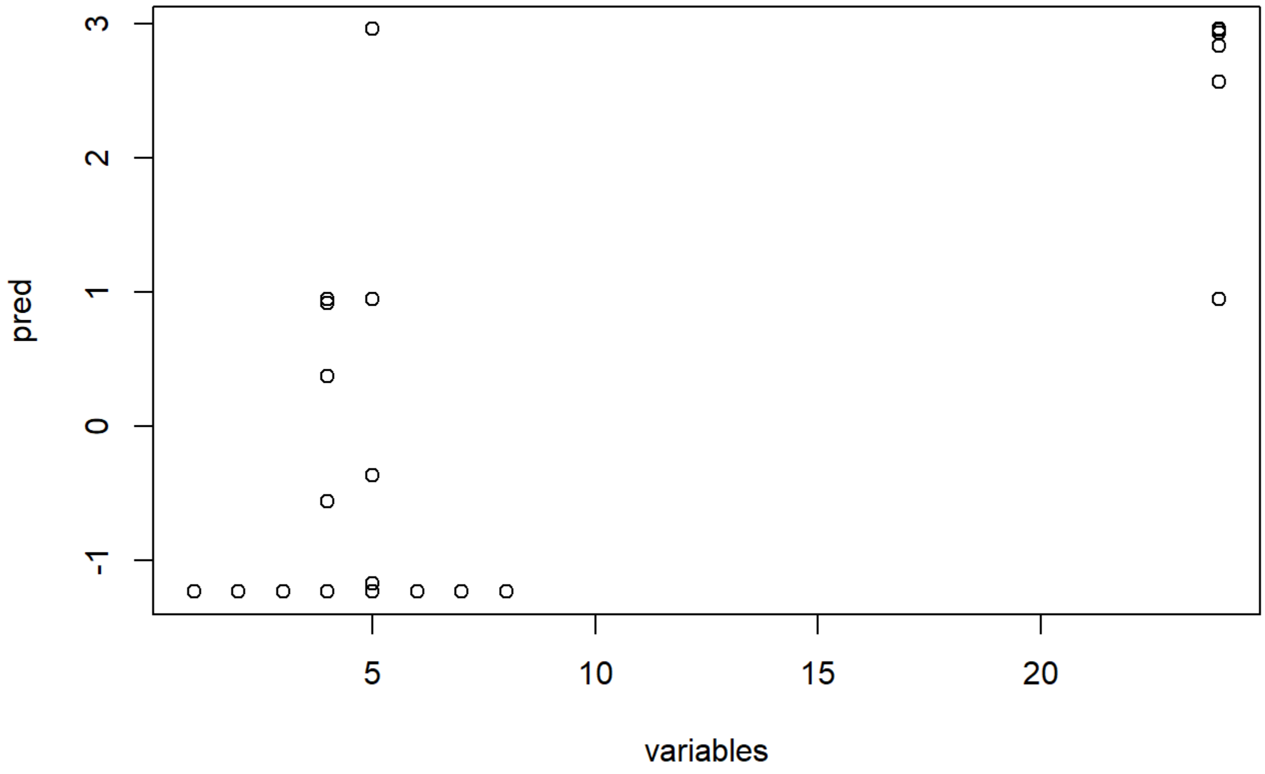

pred
-1
0
1
2
3
0.0
0.2
0.4
0.6
0.8
1.0
variables

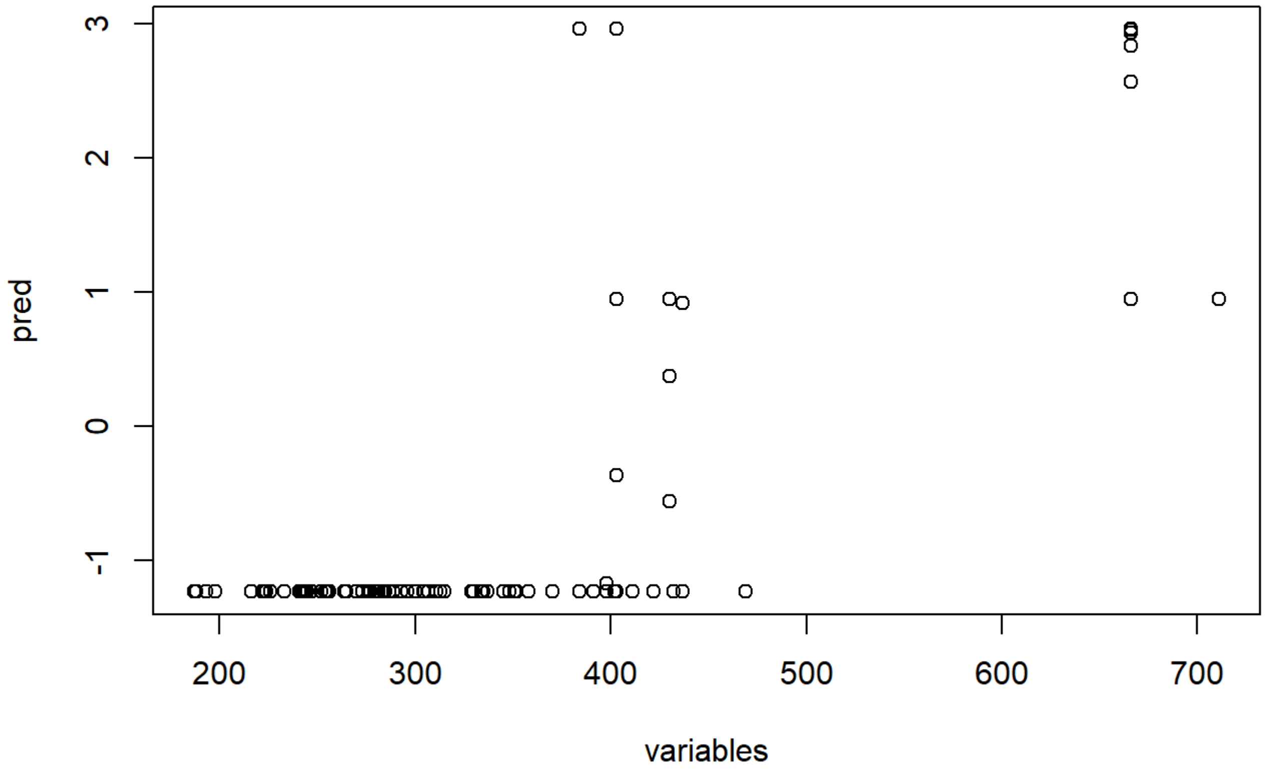

pred
-1
0
1
2
3
0.4
0.5
0.6
0.7
0.8
variables

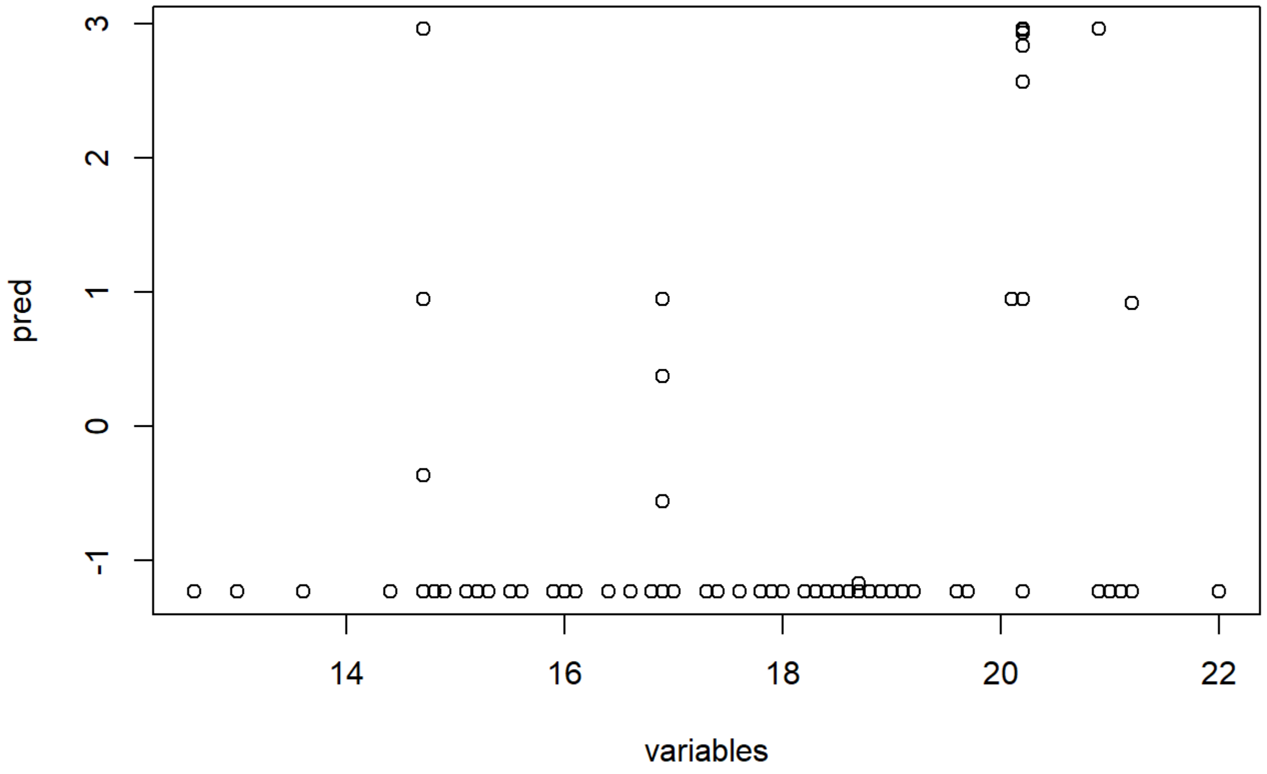

pred
3
2
1
0
-1
4
5
6
7
8
variables

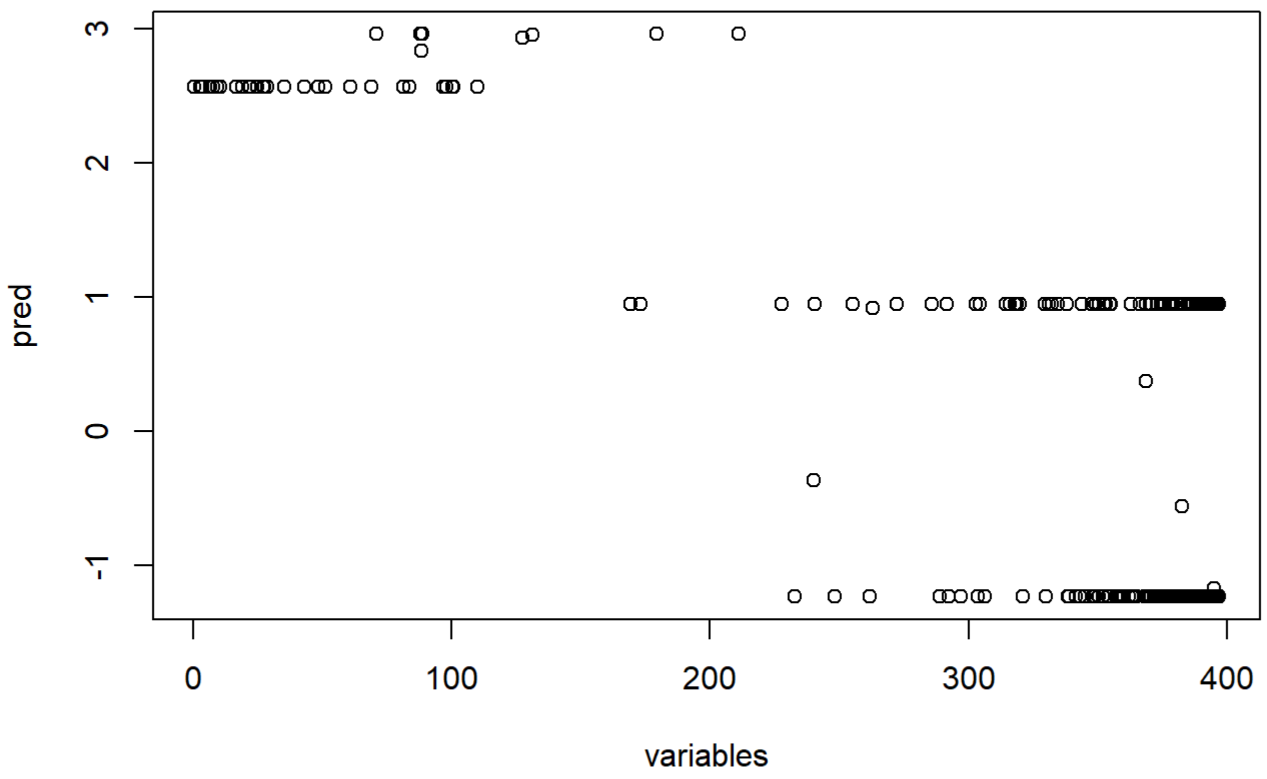

pred
3
2
1
0
-1
0
20
40
60
80
100
variables

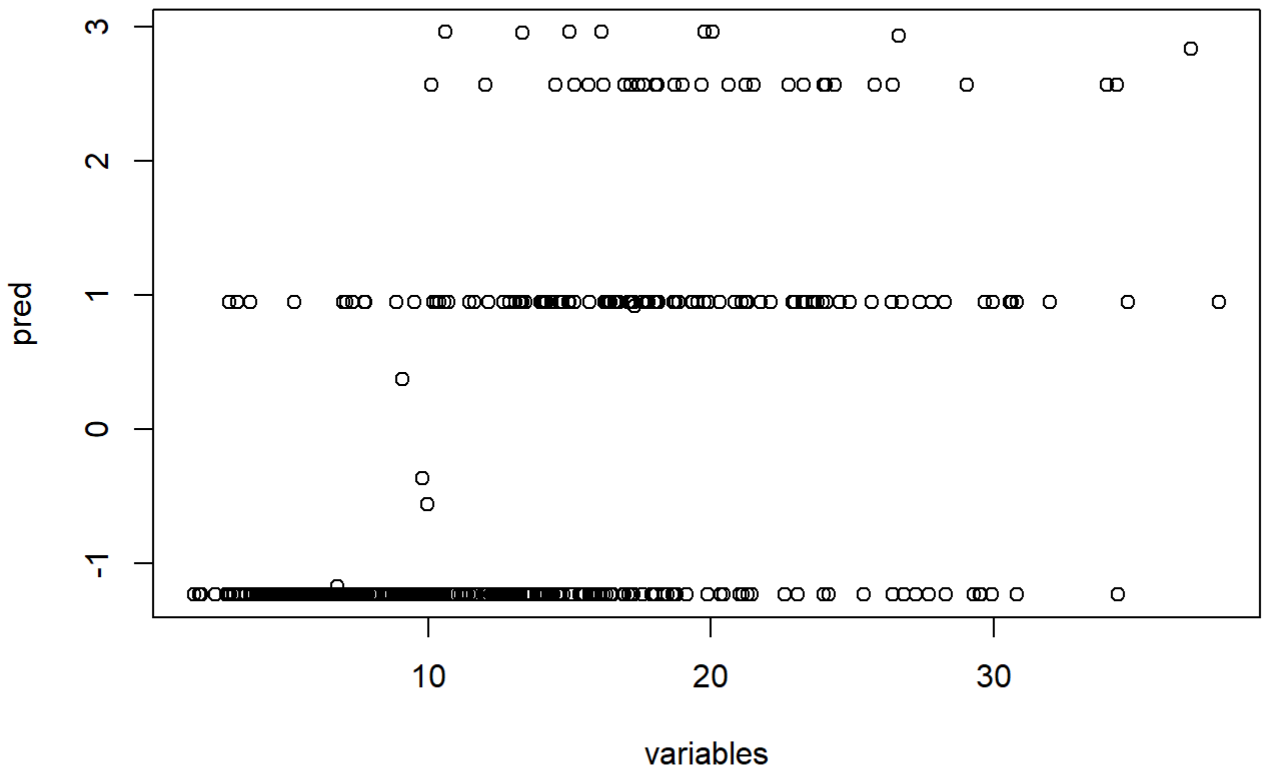

pred
3
2
1
0
-1
2
4
6
8
10
12
variables

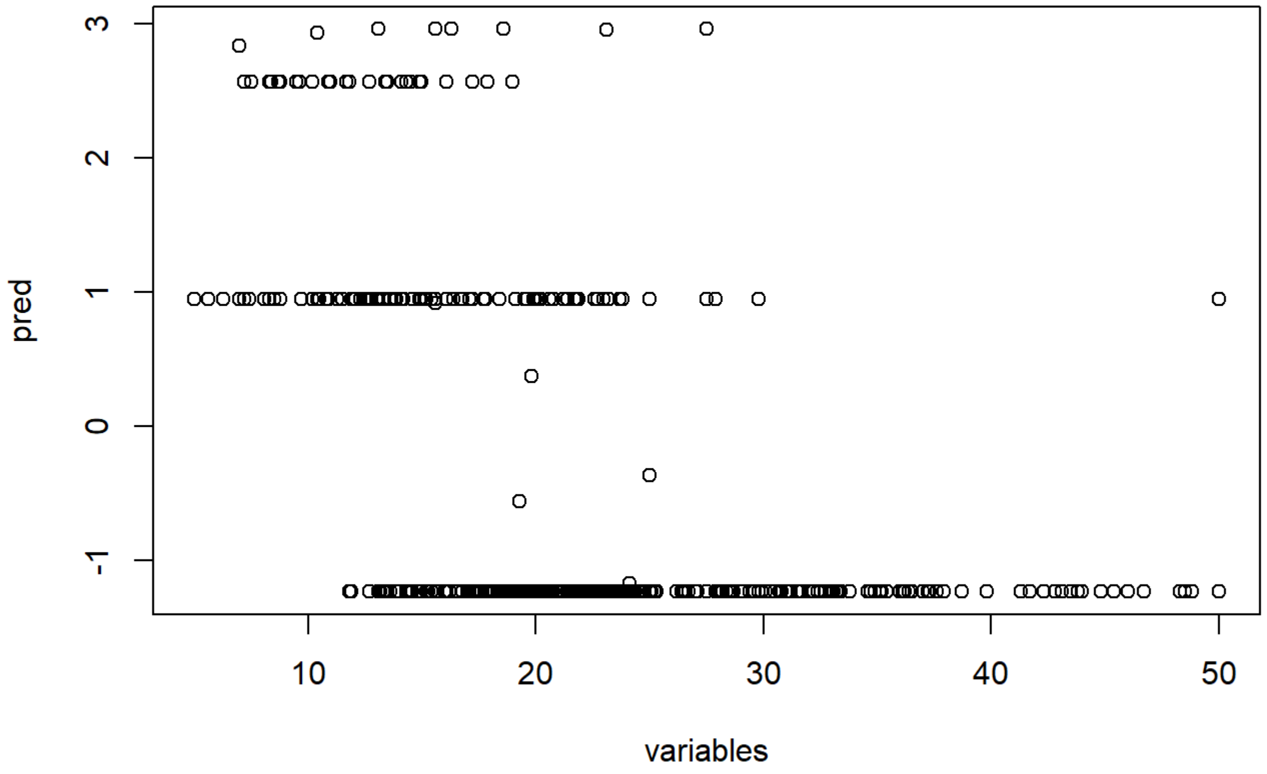

pred
3
2
1
0
-1
5
10
15
20
variables


pred
-1
0
1
2
3
200
300
400
500
600
700
variables


pred
-1
0
1
2
3
14
16
18
20
22
variables


pred
variables
3
2
1
0
-1
0
100
200
300
400


pred
variables
3
2
1
0
-1
10
20
30


pred
variables
3
2
1
0
-1
10
20
30
40
50