ML Code

Zheng Pan

7/18/2021

Introduction

This is a sophisticated tutorial on Machine Learning using R programming language. The dependent value is continuous in this case.

Workflow

1. modeling
2. performance assessment
3. model optimization

code

prerequisites

- load required libraries

```
library('tidyverse')
library('mlr3')
library('mlr3verse')
library("mlr3viz")
library('reshape2')
library('GenSA')
```

If you get an error loading them, you should probably install or update some of required packages using :

```
install.packages("NAME OF PACKAGE")
```

- initialize file paths

```
# Change your path to your .csv
path_of_file <- '~/Desktop/code for ml/rawdata/train_and_test.csv'
st1 <- as.data.frame(read.csv(file = path_of_file))
# Change your path for prediction set
path_for_prediction <- '~/desktop/predions.csv'
```

THe path above is path on my laptop, users shall change them according to their own laptops.

- Create a function that generates pearson heat map

```
pearson_heat <- function(df, corm = -1){
  defaultW <- getOption("warn")
  options(warn = -1)
  if(corm == -1){
    corm = cor(df)
  }
  options(warn = defaultW)
  res <- melt(corm) %>%
    ggplot(aes(Var1,Var2,fill = value)) +
    geom_tile(color = "black",alpha = 1,
              show.legend = T) +
    theme(axis.text.x = element_text(angle = 90,
                                     hjust = 0.5,
                                     vjust = 0.5)) +
    scale_fill_gradient2(low = 'black', high = 'white', mid = "red", midpoint = 0,guide = "colorbar", li
    theme(panel.border = element_blank(),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          axis.line = element_line(colour = "black")) +
    xlab(NULL) +
```

```
    guides(fill = guide_colorbar(title = NULL)) +
    ylab(NULL)
  return(res)
}
```

- Create a function which selects feature by pearson correlation coefficient

```
fea_slc_bycor <- function(df, corr = 0.8){
  corm <- cor(df)
  name_of_features <- colnames(df)
  name_of_features_d <- name_of_features
  origin_fea_length <- length(name_of_features)
  for(q in 1:(origin_fea_length - 1)){
    fea_t <- name_of_features_d[q]
    other_fea_t <- name_of_features_d[(q+1):length(name_of_features_d)]
    de_fea <- names(corm[fea_t, other_fea_t][abs(corm[fea_t, other_fea_t]) >= corr])
    name_of_features <- name_of_features[!(name_of_features %in% de_fea)]
  }
  res <- df[, colnames(df) %in% name_of_features]
  return(res)
}
```

- import data
- prepare the prediction set

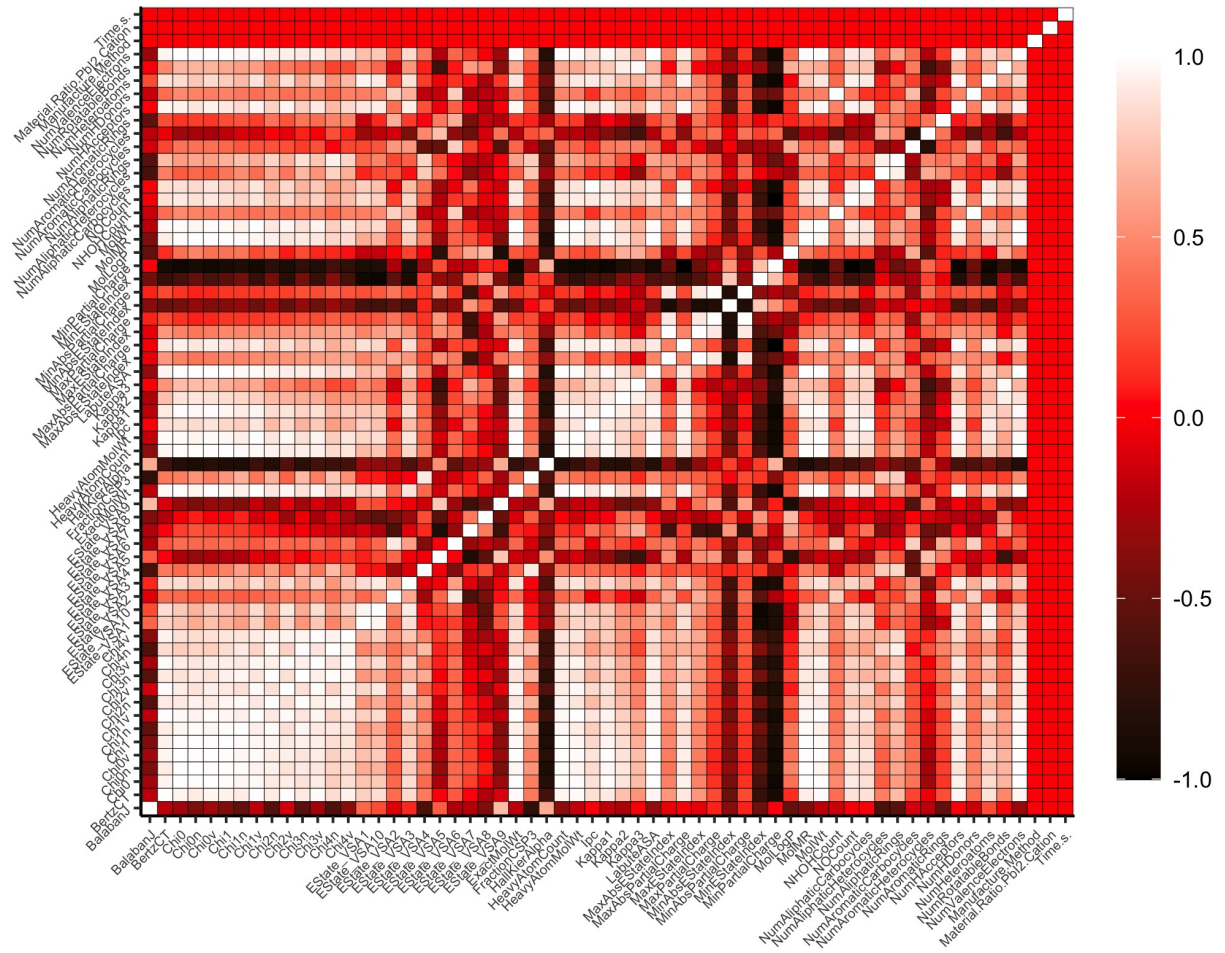
Code in this section generates a data.frame which is meant for prediction set. Note this, in this case, the number of descriptors which are suppuosed to change to generate a prediction set is three and in other cases the number might change and the code below shall change accordingly.
It's not the best code that generates prediction set(it might be slow for larger descriptor amount)
Users shall just ignore warnings.

```
tsk_df1 <- lapply(unique(st1$X), function(s){
  df1 <- st1[st1$X == s, ][1, ][, c(-length(st1[st1$X == s, ][1, ]),
      -length(st1[st1$X == s, ][1, ]) + 1,
      -length(st1[st1$X == s, ][1, ]) + 2, -length(st1[st1$X == s, ][1, ])
      + 3)]
  df2 <- cbind(df1, Manufacture.Method = c(1,2,3))
  res1 <- lapply(1:nrow(df2), function(q){
    df_t <- df2[q, ]
    (cbind(df_t, Material.Ratio.PbI2.Cation = seq(0,2,0.1)))
  })
  df3 <- do.call(rbind, res1)
  res2 <- lapply(1:nrow(df3), function(q){
    # q = 1
    df_t <- df3[q, ]
    (cbind(df_t, Time.s. = seq(0,500,5)))
  })
  df4 <- do.call(rbind, res2)
  df4 <- as.data.frame(df4)
```

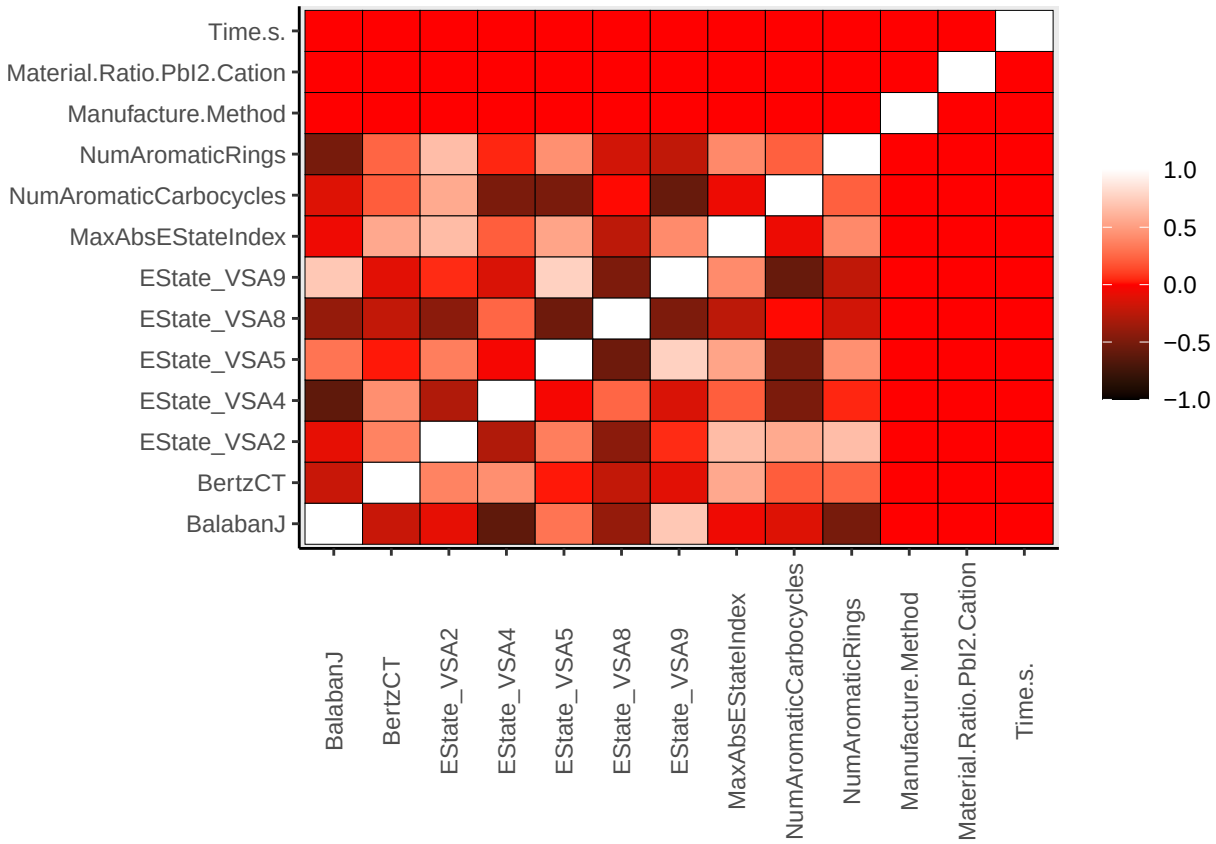
```
  df4$Current.A. <- NA
  df4
})
tsk_df1 <- do.call(rbind ,tsk_df1)
st2 <- rbind(st1, tsk_df1)
st_t <- st2[, -1]
```

- feature engineering

```
if(is.character(st1[, 1])){
  st <- st1[, -1]
}
NN <- nrow(st1)
features <- st[, -ncol(st)]
rawFeaPic <- pearson_heat(features, corm = cor(features)) +
  theme(legend.position = "right",
        legend.title = element_text(size = 10),
        legend.key.height = unit(2, "cm"),
        axis.text.x = element_text(size = 5.3,
                                   angle = 45,
                                   hjust = 1,
                                   vjust = 1),
        axis.text.y = element_text(size = 5.3,
                                     angle = 45)
        )
ggsave(filename = "Raw_pearson_heat.jpeg", plot = rawFeaPic, width = 17.24, height = 13.41, units = 'cm
```




```
fea_new <- fea_slc_bycor(features, corr = 0.8)
fea_new_names <- colnames(fea_new)
pearson_heat(fea_new)
```



machine learning

- Task construction

```
tsk_df <- cbind(st_t[, colnames(fea_new)], current = st_t[, ncol(st_t)])
task <- TaskRegr$new(id = "task", backend = tsk_df, target = "current")
```

- pick up a learner

```
cl <- mlr_learners$keys()[35:44]
qq = 9
learner <- mlr_learners$get(cl[qq])
```

In this case, the learner name is regr.svm

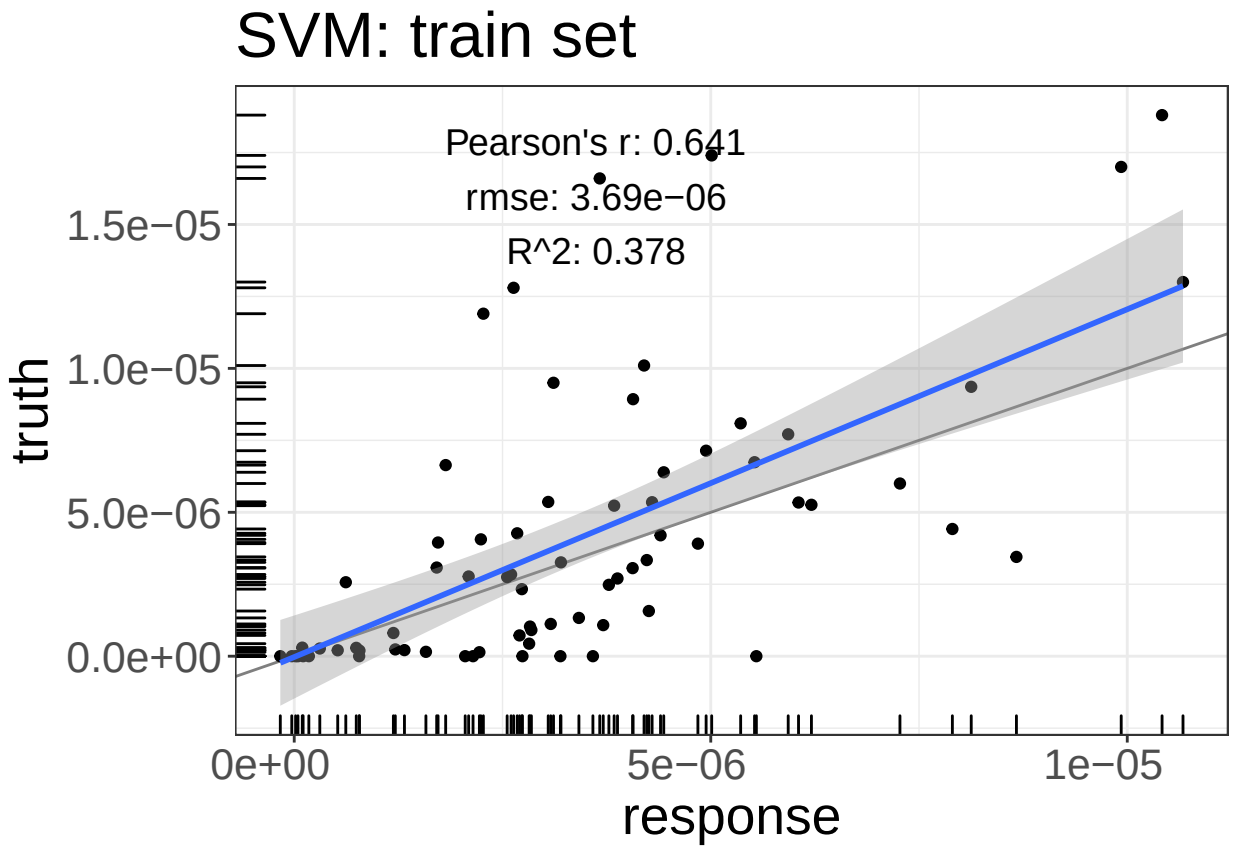
- train and test

```
train_set <- sample(NN, 0.8 * NN)
test_set <- setdiff(seq_len(NN), train_set)
write_csv(data.frame(train_set), "train_set.csv")
write_csv(data.frame(test_set), "test_set.csv")
learner$train(task, row_ids = train_set)
prediction <- learner$predict(task, row_ids = test_set)
```

- visualization of train and test set(performance assessment)

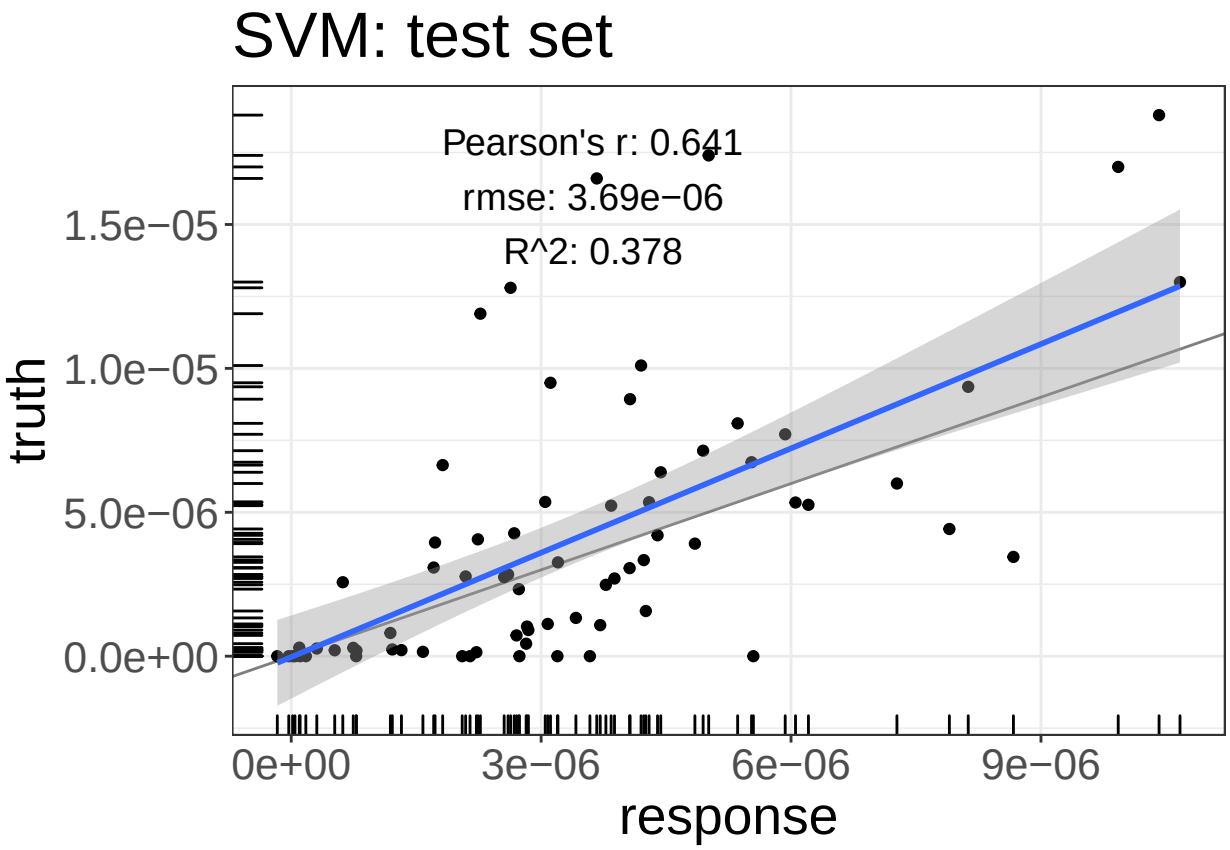
```
measure <- msr("regr.mse")
mse <- format(prediction$score(measure), digits = 5)
measure <- msr("regr.rmse")
rmse <- format(prediction$score(measure), digits = 3)
preds <- prediction$response
actual <- prediction$truth
pr <- round(cor(cbind(preds, actual))[1,2],3)
rss <- sum((preds - actual) ^ 2)  ## residual sum of squares
tss <- sum((actual - mean(actual)) ^ 2)  ## total sum of squares
rsq <- round(1 - rss/tss, 3)
train_set_pic_SVM <- autoplot(prediction) +
  theme_bw() +
  annotate("text",
           x = 0.35 * (range(prediction$response)[2] - range(prediction$response)[1])
           + range(prediction$response)[1],
           y = 0.85 * (range(prediction$truth)[2] - range(prediction$truth)[1])
           + range(prediction$truth)[1],
           label = paste0("Pearson's r: ",pr, "\n", "rmse: ",rmse, "\n", "R^2: ",rsq),
           size = 5) +
  theme(text = element_text(size = 20),axis.text.x = element_text(hjust = .7)) +
  scale_x_continuous(n.breaks = 3) +
  labs(title = "SVM: train set")
prediction <- learner$predict(task, row_ids = test_set)
measure <- msr("regr.mse")
mse <- format(prediction$score(measure), digits = 5)
measure <- msr("regr.rmse")
rmse <- format(prediction$score(measure), digits = 3)
preds <- prediction$response
actual <- prediction$truth
pr <- round(cor(cbind(preds, actual))[1,2],3)
rss <- sum((preds - actual) ^ 2)  ## residual sum of squares
tss <- sum((actual - mean(actual)) ^ 2)  ## total sum of squares
rsq <- round(1 - rss/tss, 3)
test_set_pic_SVM <- autoplot(prediction) +
  theme_bw() +
  annotate("text",
           x = 0.35 * (range(prediction$response)[2] - range(prediction$response)[1])
           + range(prediction$response)[1],
           y = 0.85 * (range(prediction$truth)[2] - range(prediction$truth)[1])
           + range(prediction$truth)[1],
           label = paste0("Pearson's r: ",pr, "\n", "rmse: ",rmse, "\n", "R^2: ",rsq),
           size = 5) +   theme(text = element_text(size = 20)) +   scale_x_continuous(minor_breaks = 2)
  labs(title = "SVM: test set")
print(train_set_pic_SVM)
```

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




```
print(test_set_pic_SVM)
```

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



- output predictions

```
ind_seq <- c(seq(NN + 1, nrow(st_t), 500000), nrow(st_t))
ind_seq_l <- lapply(1:(length(ind_seq) - 1), function(q){
  if(q != 1){
    return((ind_seq[q] + 1):ind_seq[q+1])
  }else{
    ind_seq[q]:ind_seq[q+1]
  }
})
pred <- lapply(1:length(ind_seq_l), function(q){
  prediction2 <- learner$predict(task, row_ids = ind_seq_l[[q]])
  data.frame(prediction2$response)
})
preds <- do.call(rbind, pred)
to_w <- st2[(NN+1): nrow(st2),]
to_w[, ncol(to_w)] <- preds
write_csv(to_w, file = path_for_prediction)
```

- Prediction set visualization

```
st_t <- read_csv(file = path_for_prediction)
```

```
##
## -- Column specification --------------------------------------------------------
```

```
## cols(
##   .default = col_double(),
##   X = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
st_t <- as.data.frame(st_t)
st_t_2 <- read_csv(file = path_of_file)
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   X1 = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
st_t_2 <- as.data.frame(st_t_2)
colnames(st_t_2) <- colnames(st_t)
st_t <- rbind(st_t_2, st_t)
st_t$lab <- "a"
st_t$lab[(NN + 1): nrow(st_t)] <- "prediction set"
km_test_set <- c(unlist(as.data.frame(read_csv(file = "test_set.csv"))))
```

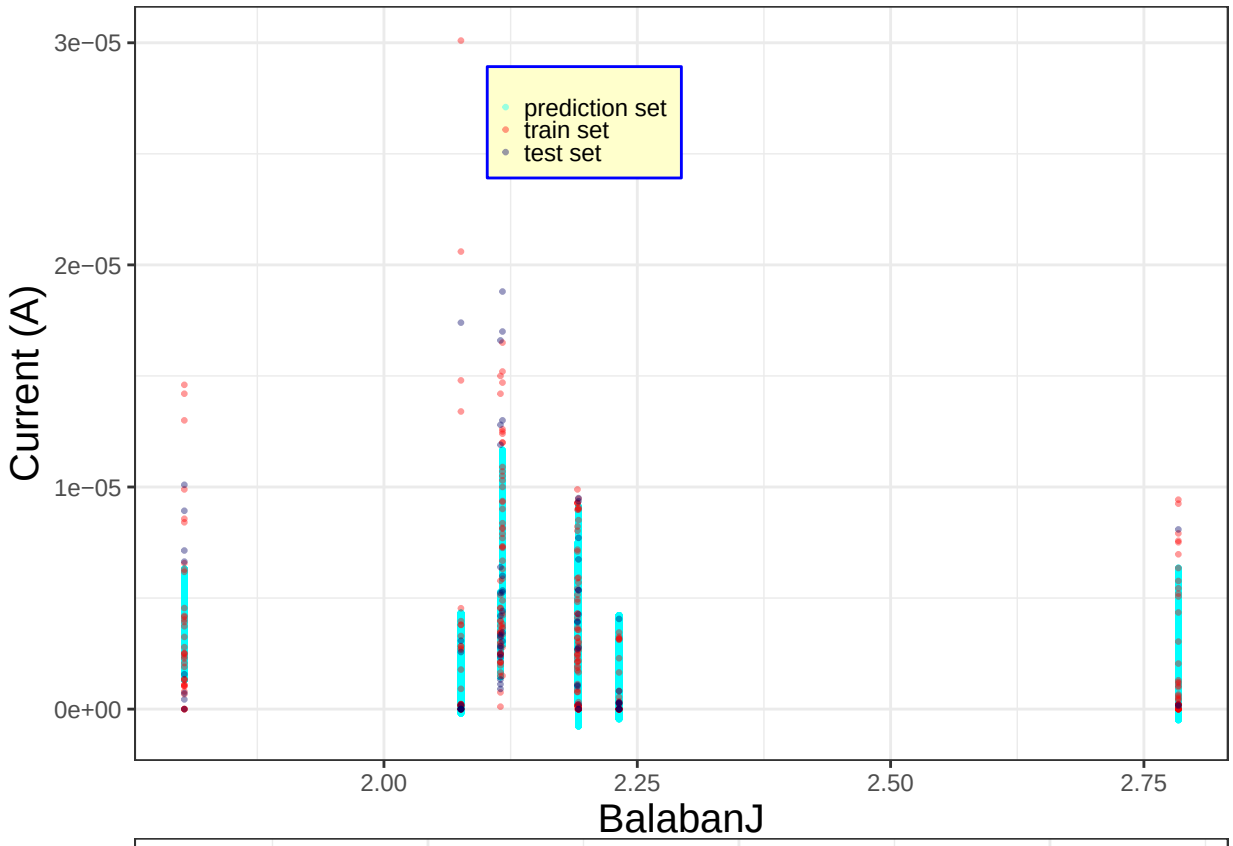
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   test_set = col_double()
## )
```

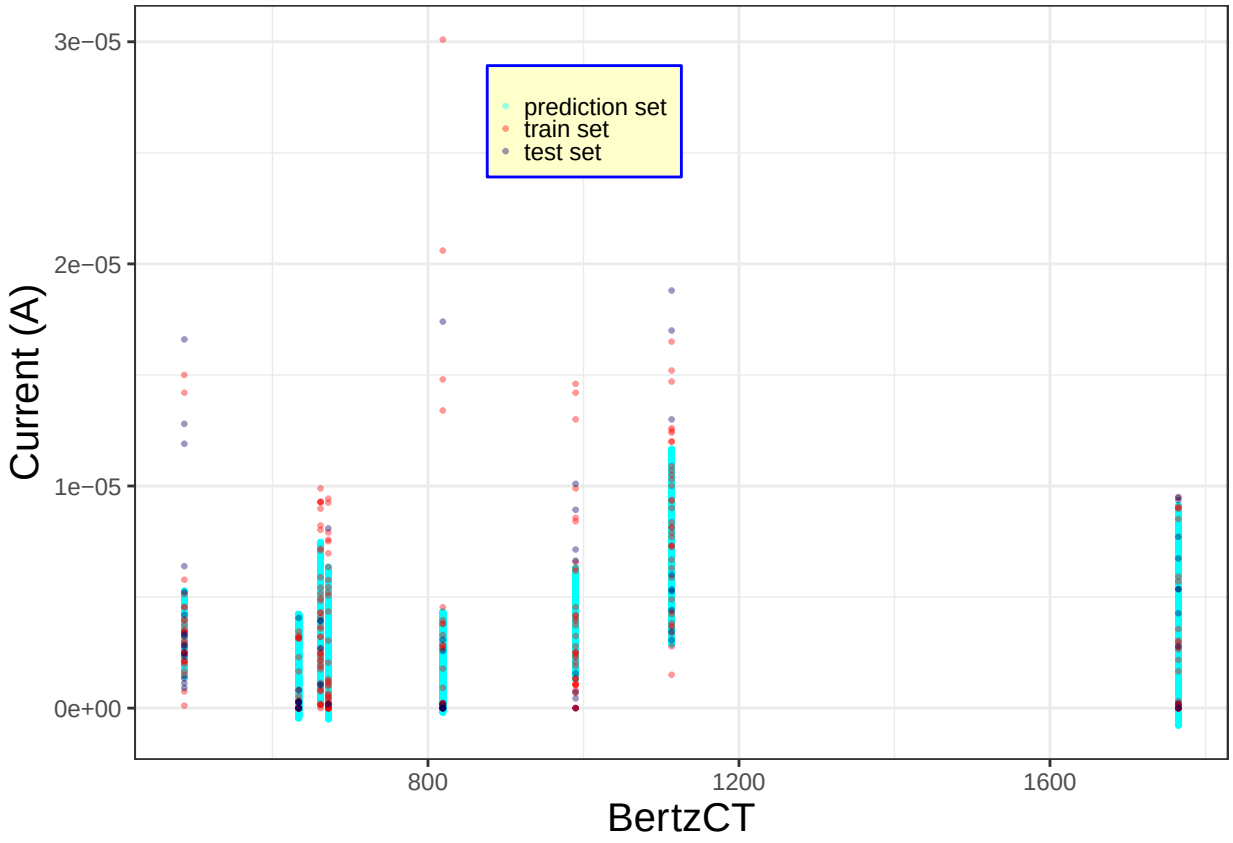
```
names(km_test_set) <- NULL
km_train_set <- unlist(as.data.frame(read_csv(file = "train_set.csv")))
```

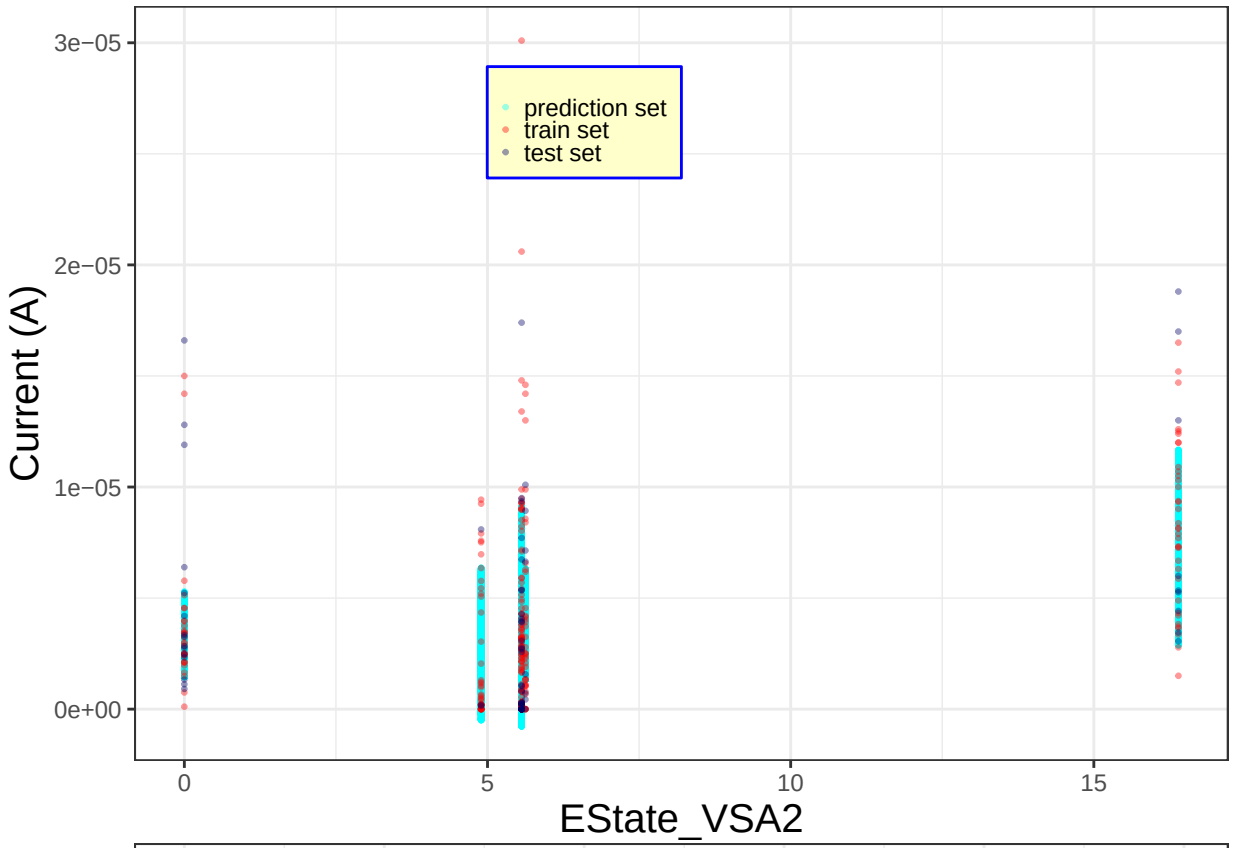
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   train_set = col_double()
## )
```

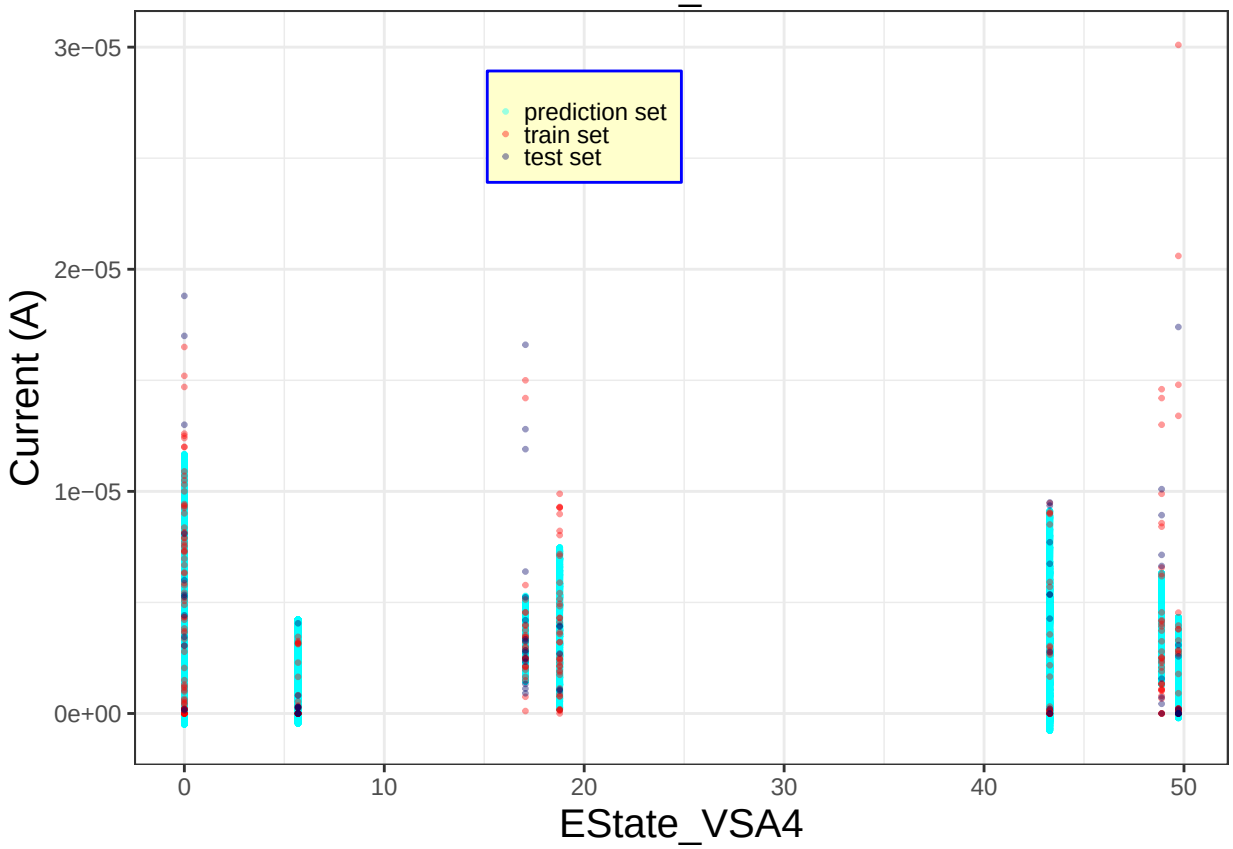
```
names(km_train_set) <- NULL
st_t$lab[km_test_set] <- "test set"
st_t$lab[km_train_set] <- "train set"
st_t_1 <- rbind(st_t[st_t$lab == "prediction set", ],
                st_t[st_t$lab == "train set", ], st_t[st_t$lab == "test set", ])

#### plot for 12 features ####

plot_of_prediation <- function(f_df, f_xn, f_yn){
  res_p <-
```

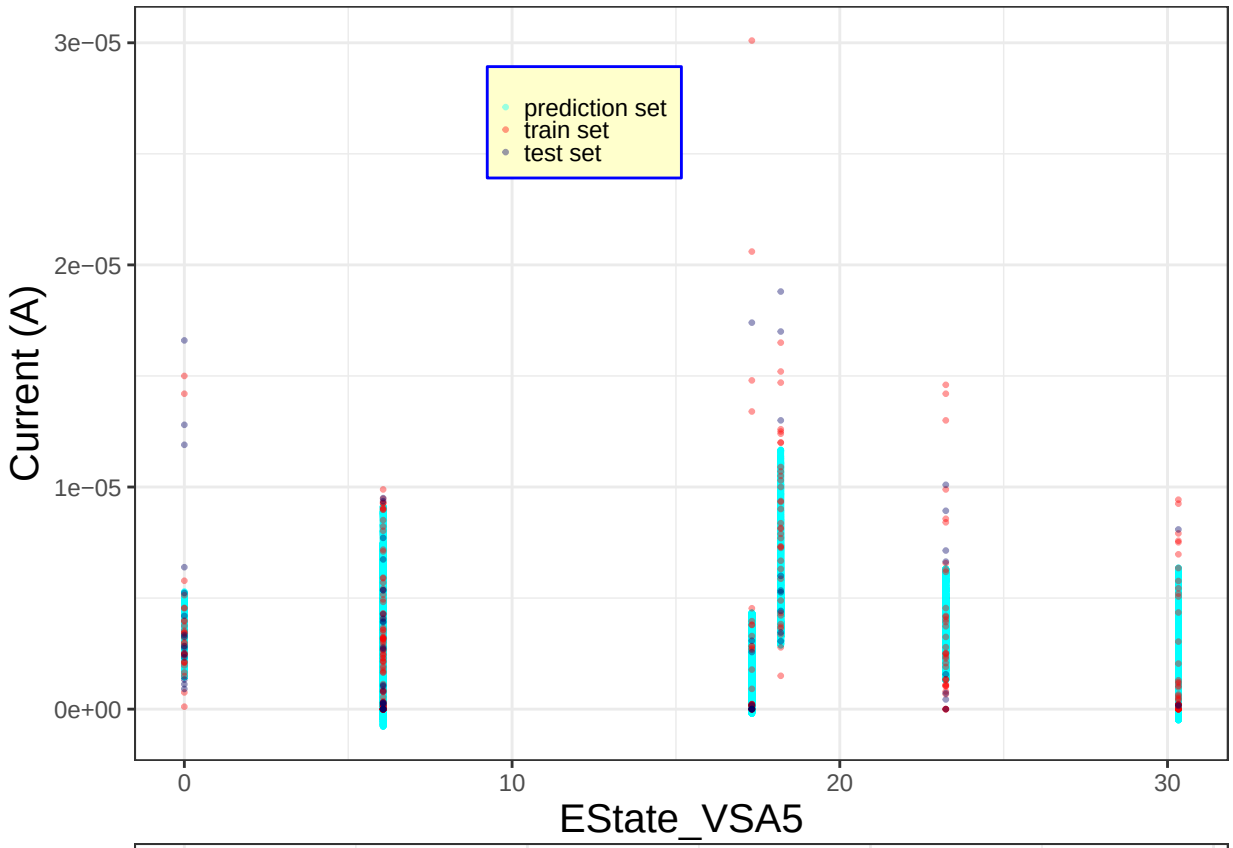
```
    ggplot(data = f_df, mapping = aes(y = f_df[, f_yn])) +
    geom_point(aes(x = f_df[, fea_new_names[f_xn]], color = factor(lab)),
               alpha = 0.4, size = .5) +
    theme_bw() +
    scale_color_manual(
      name = NULL,
      breaks = c("prediction set", "train set", 'test set'),
      values = c(
        'prediction set' = '#00FFFF',
        'train set' = '#FF0000',
        'test set' = '#000066'
      )
    ) +
    xlab(fea_new_names[f_xn]) +
    ylab('Current (A)') +
    guides(color = guide_legend(title=NULL)) +
    theme(
      legend.position = c(.92, .92),
      legend.justification = c(1,1),
      legend.background = element_rect(colour = 'blue', fill = '#FFFFCC'),
      legend.key.size = unit(.1, 'cm'),
      legend.key = element_rect(colour = NULL, fill = '#FFFFCC'),
      axis.title = element_text(size = 15)
    )
  return(res_p)
}


for (t in 1:length(fea_new_names)) {
  t_pic <- plot_of_prediation(st_t_1, t, ncol(st_t_1) - 1)
  temp_pic <- t_pic +
    theme(legend.position = c(.5, .92))
  print(temp_pic)
}
```

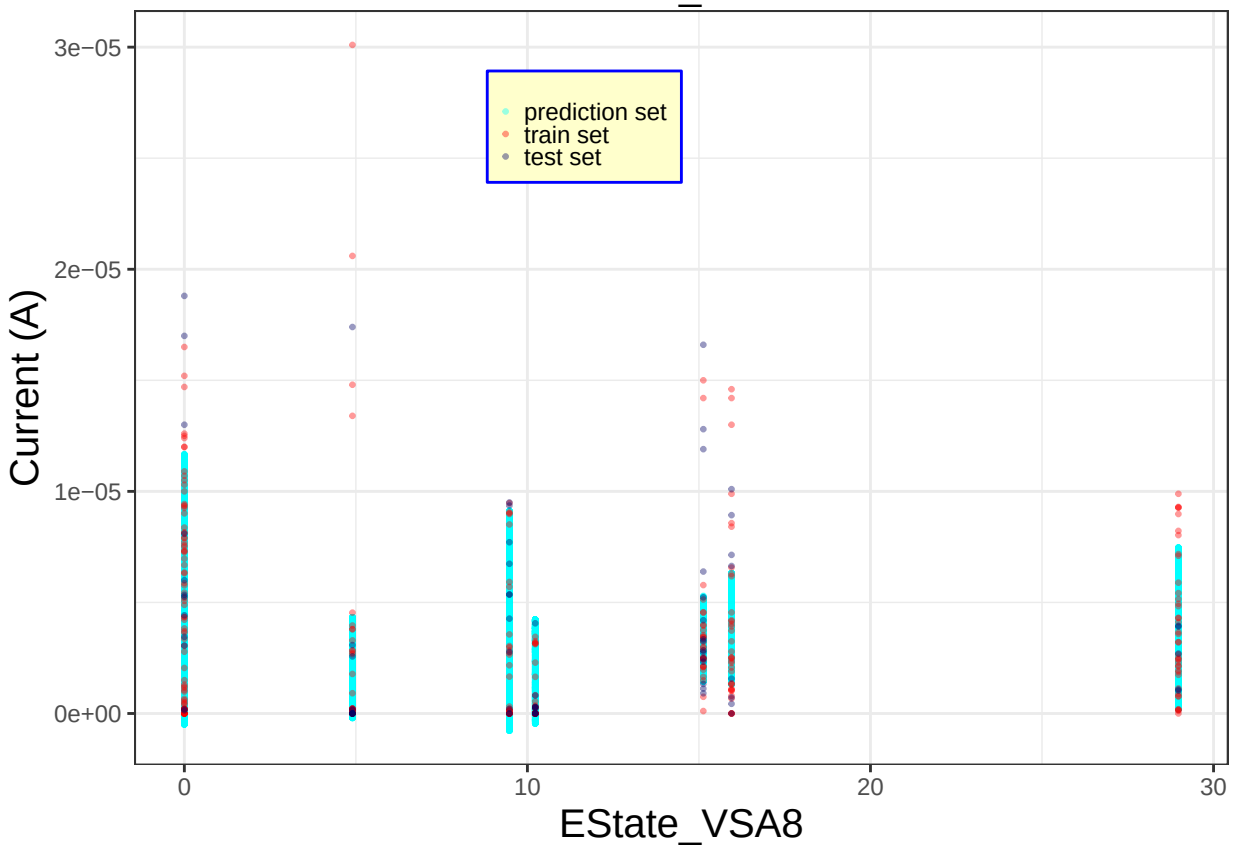

prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
2.00
2.25
2.50
2.75
BalabanJ

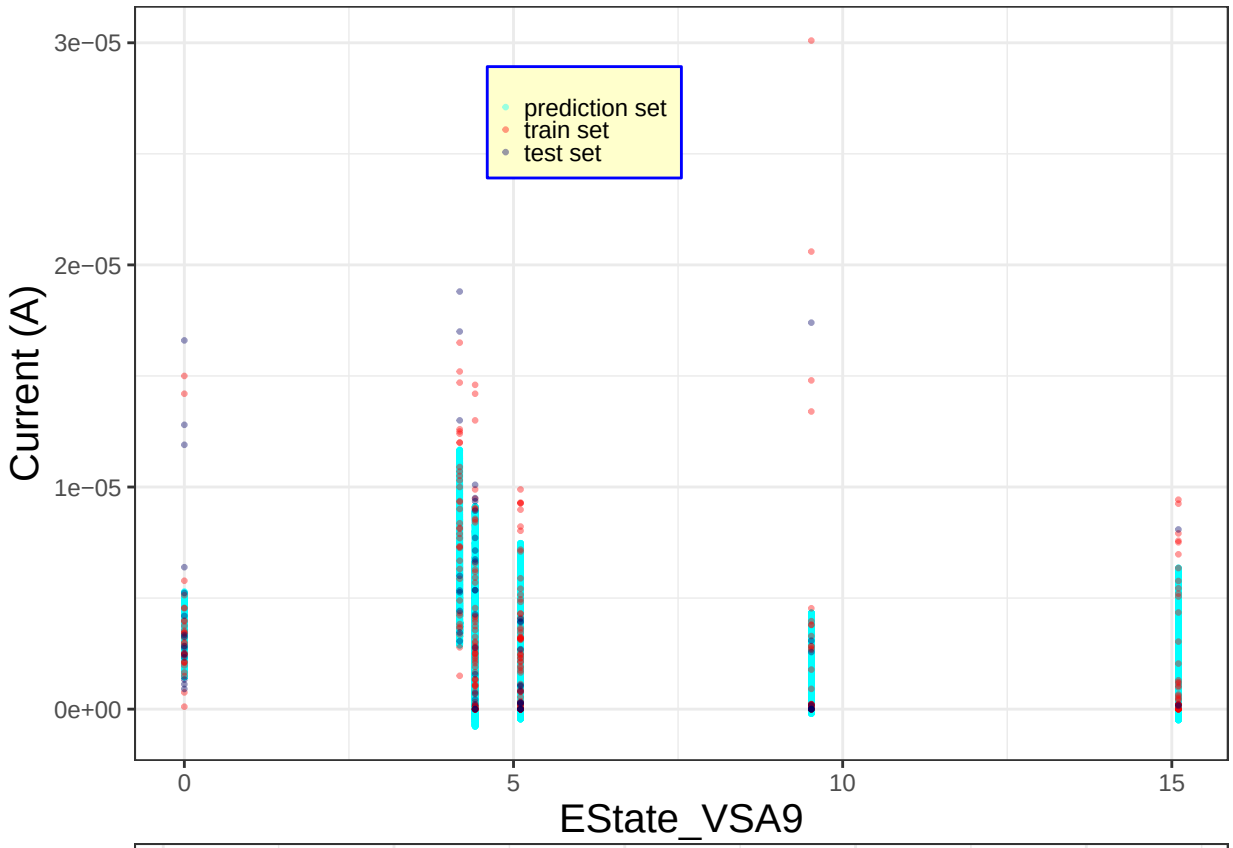

prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
800
1200
1600
BertzCT

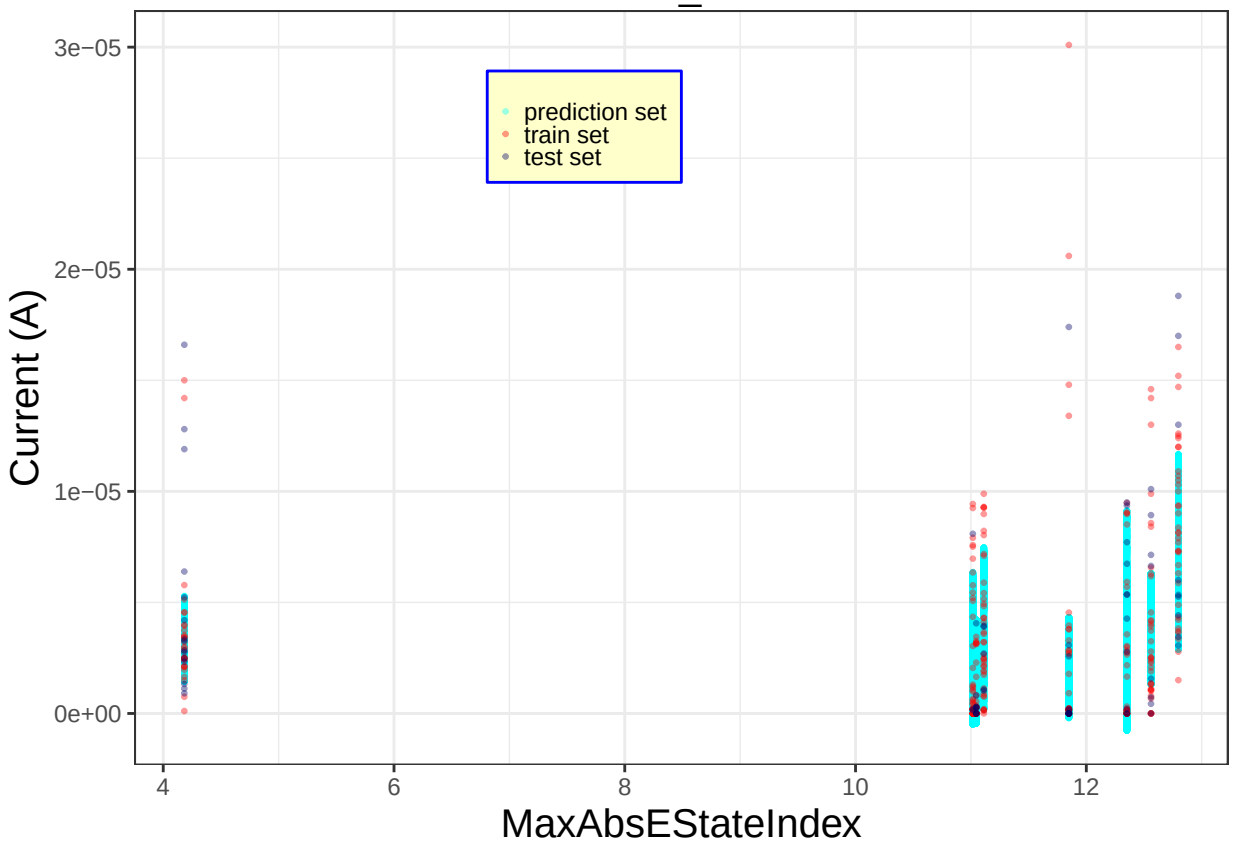

prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
0
5
10
15
EState_VSA2

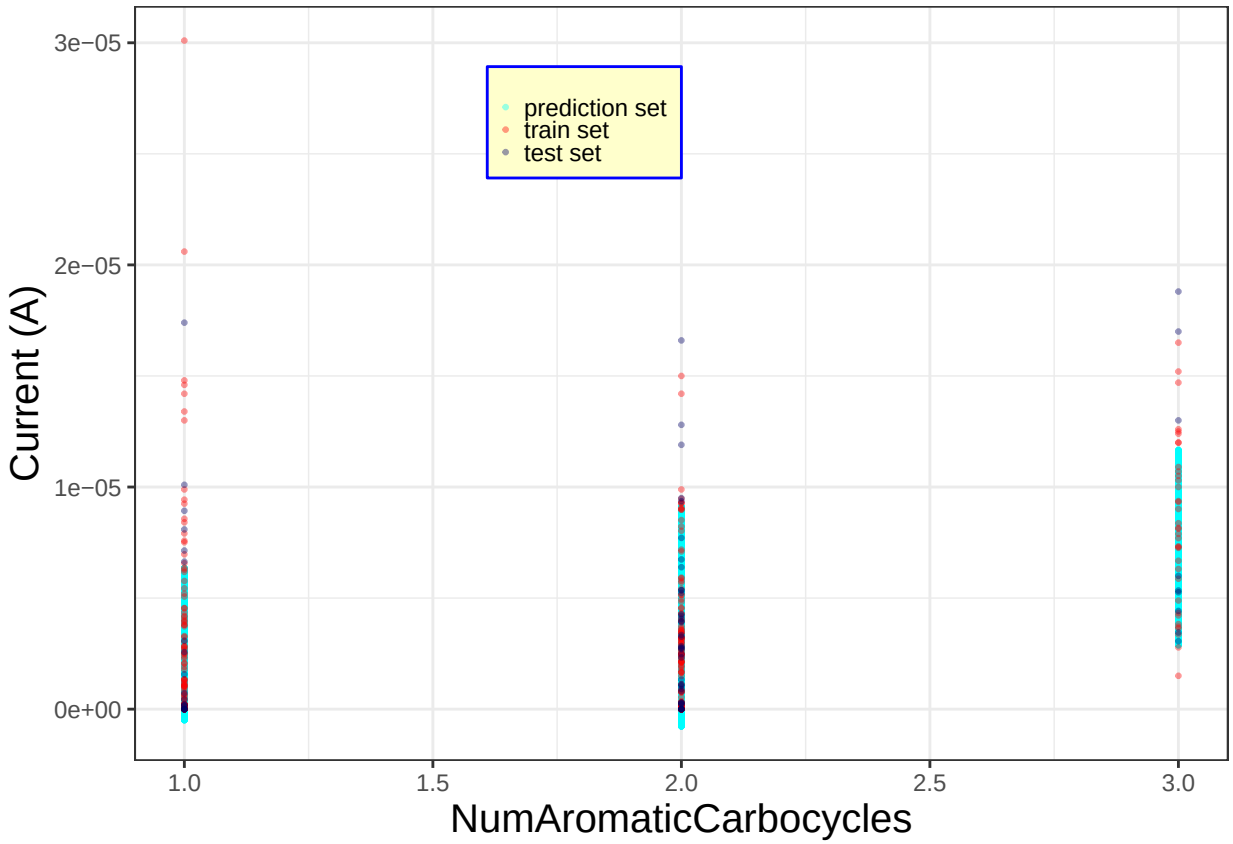

prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
0
10
20
30
40
50
EState_VSA4

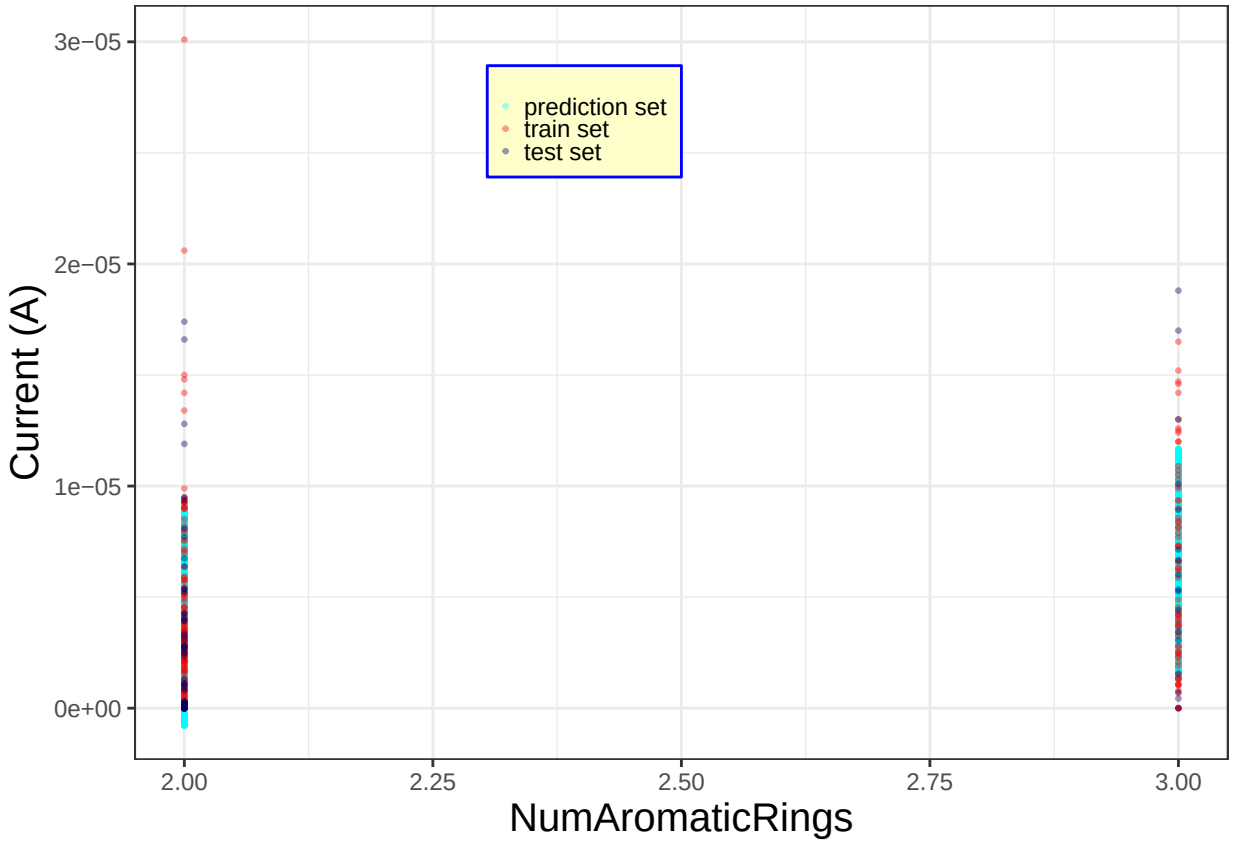

prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
0
10
20
30
EState_VSA5

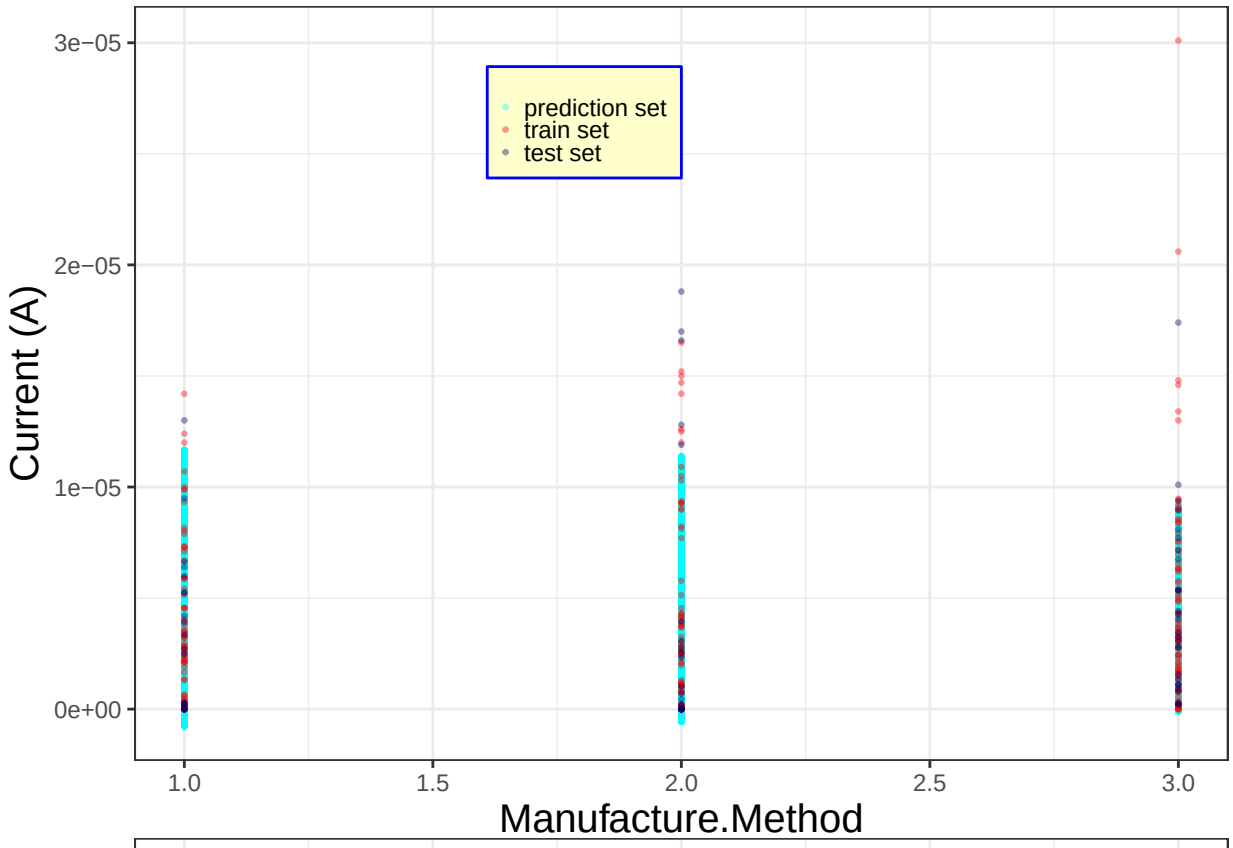

prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
0
10
20
30
EState_VSA8

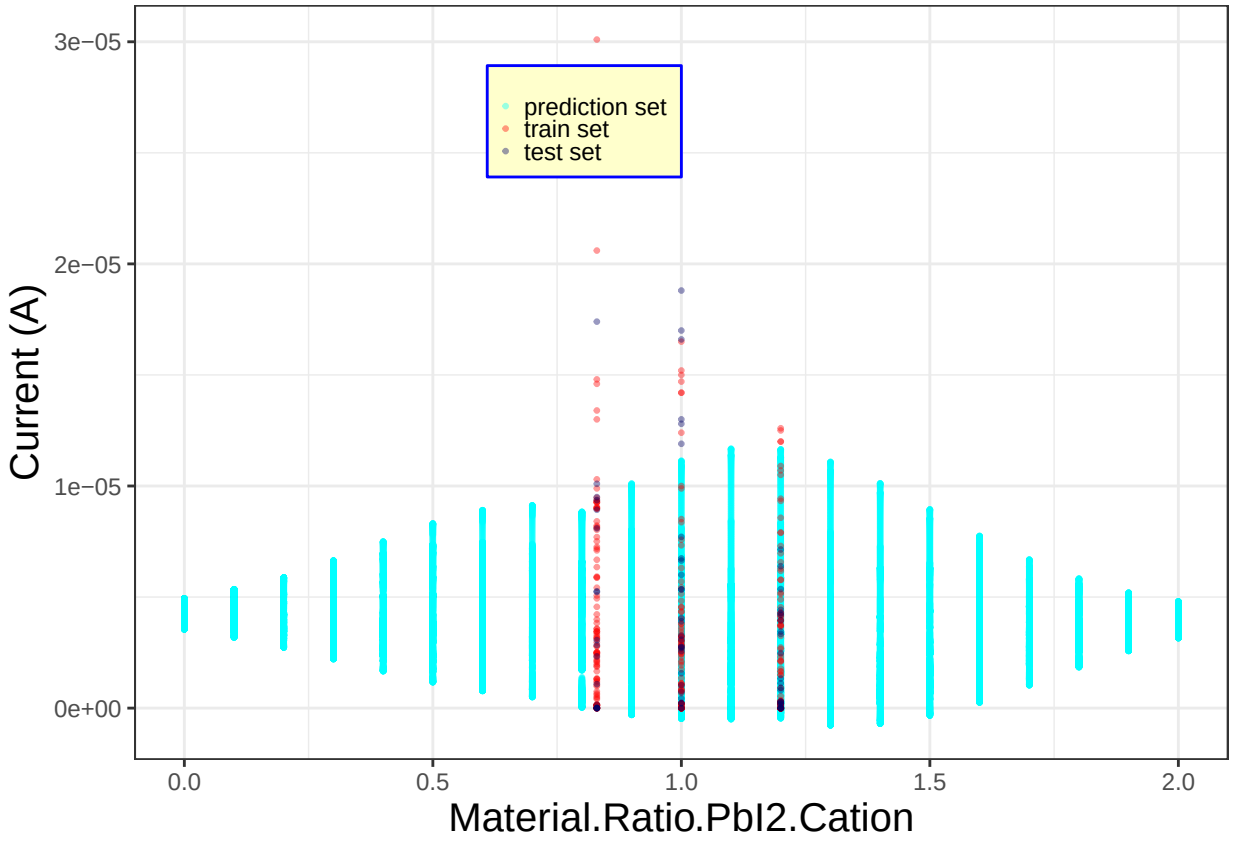

prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
0
5
10
15
EState_VSA9

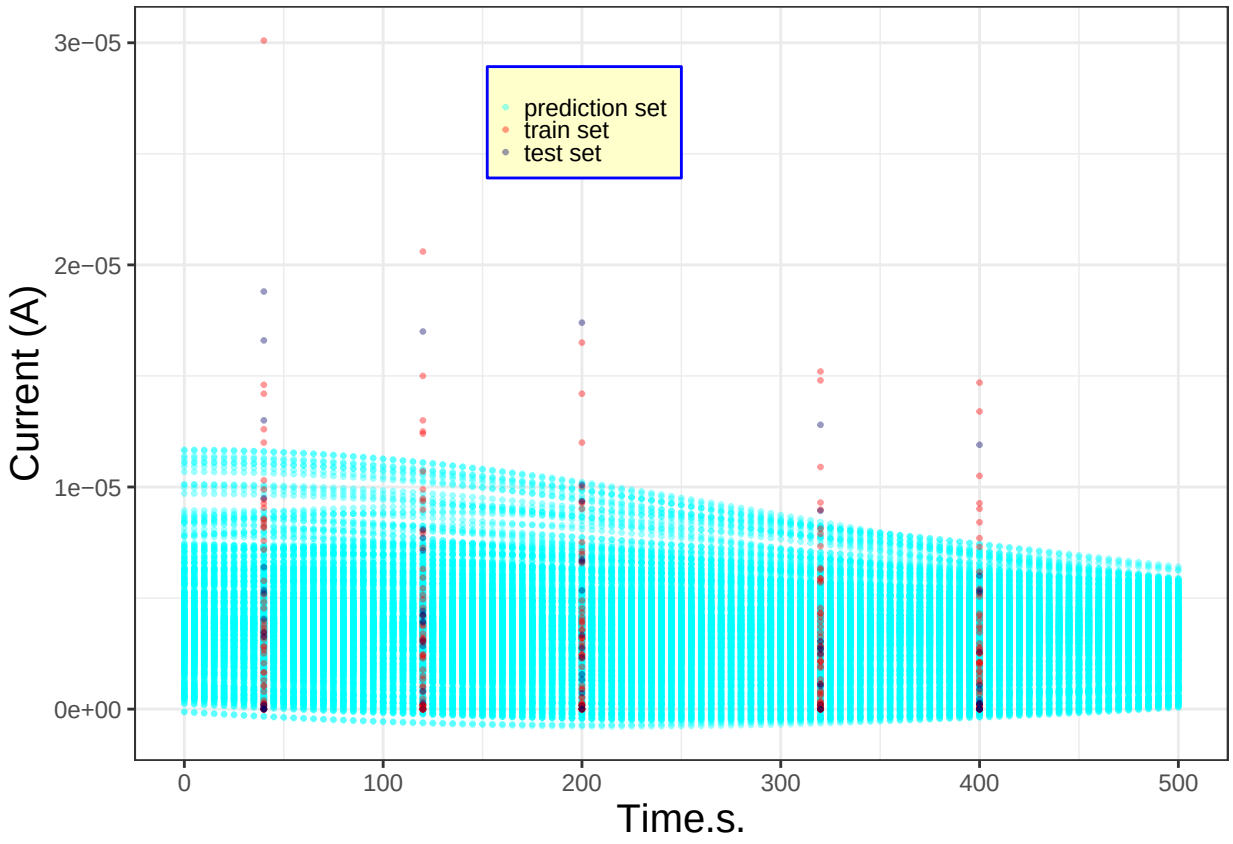

prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
4
6
8
10
12
MaxAbsEStateIndex


prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
1.0
1.5
2.0
2.5
3.0
NumAromaticCarbocycles


prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
2.00
2.25
2.50
2.75
3.00
NumAromaticRings


prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
1.0
1.5
2.0
2.5
3.0
Manufacture.Method


prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
0.0
0.5
1.0
1.5
2.0
Material.Ratio.PbI2.Cation


prediction set
train set
test set
Current (A)
3e−05
2e−05
1e−05
0e+00
0
100
200
300
400
500
Time.s.