Predict Filtration Time

Ki Jun, Choi

2019-06-27

Introduction

AMPAC is a consigned pharmaceutical developer and manufacturer, located in Rancho Cordoba, California, USA. In the case of pharmaceutical production, yield, quality, and the reduction of production time are important issues as a batch process. Among these, in this project, the production time was progressed as a target. The process that can be reduced during production is filtration time, and the purpose of this task is to estimate the influencing factors affecting filtration time and their influence.

That is, finding an operation method that reduces the filtration Time rather than accurately predicting the filtration Time is the most important part from the user's point of view. For this reason, models such as regression, which have good explanatory power, are the first to be considered. However, the assumption of a linear relationship between variables was determined to be too strong an assumption to be applied in a chemical process. Also, from the standpoint of the field, the reliability of models with good predictive power is inevitably high.

Therefore, after estimating a model with good predictive power, the analysis was conducted by estimating the relationship between variables in the model.

```
# library
library(dplyr)
library(tidyr)
library(ggplot2)
library(caret)
library(ranger)
library(knitr)
library(kableExtra)

# source
source(file = "./src/data_loader.R")
source(file = "./src/utils.R")

# graph option (white space & grid for ggplot)
p <- theme_bw() +
  theme(  axis.line = element_line(colour = "black")
        , panel.grid.minor = element_blank()
        , panel.background = element_blank()
        )
```

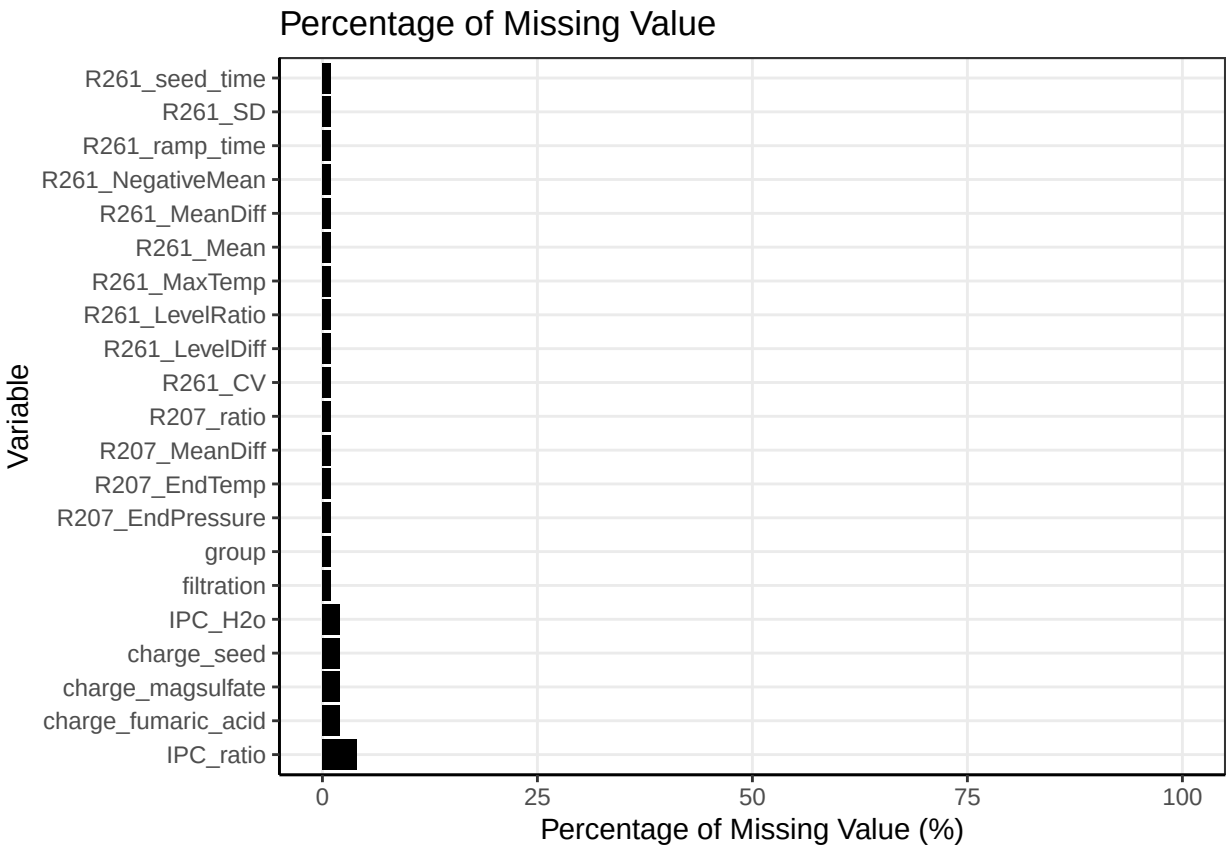
loading data and preprocessing

There are a total of three derived variable data created. Each data set was created based on EDA and field knowledge of the original data, and knowledge of chemical engineering. For more detailed EDA results, refer to the appendix.

```r
# read data (information of charging) and select variable
charge <- read_df(data_dir = "./data/charge_0508.csv")
charge <- charge %>%
  select( -charge_IPAc )

R207 <- read_df(data_dir = "./data/R207_Var_0507.csv", indicator = "R207")
R207 <- R207 %>%
  select(-filtration, -group)

R261 <- read_df(data_dir = "./data/R261_Var_0508.csv", indicator = "R261")
R261 <- R261 %>%
  select(-R261_LevelStart, -R261_LevelEnd)

# merge data using batch id
raw_df <- R207[R261, on = .(batch == batch)][charge, on = .(batch == batch)]
```

After merging into one data, a simple pre-processing was performed. Since pre-treatment was carried out to some extent in advance, only the missing values were processed.

```r
# missing value
raw_df %>%
  select(-batch) %>%
  summarise_all(.funs = function(x) round(100*sum(is.na(x))/length(x))) %>%
  gather(key = "Variable", value = "Missing") %>%
  ggplot(aes(x = reorder(Variable, -Missing), y = Missing)) +
  geom_bar(stat = 'identity', fill = "black")+
  labs(y = "Percentage of Missing Value (%)", x = "Variable") +
  ggtitle(label = "Percentage of Missing Value") +
  scale_y_continuous(limits = c(0, 100)) +
  coord_flip() + p
```




```
# remove missing row
raw_df <- na.omit(object = raw_df)
```

Model

Modeling proceeded in three steps.

1. validation

- Since time order is important due to the nature of the domain, the past 80% was divided into training data and the rest into test data in index order. In addition, using only training data, hyperparameters were tuned through the cross-validation method.

2. model

- After that, I tried to fit various models. The considered models are Linear Model (Regression, Lasso, etc.), SVM, Ensemble (RandomForest, Boosting). However, in this report, the results of Random Forest are described.

3. assessment & interpretation

- The final model was selected based on predictive power. Interpretation for the model is after estimating the Greedy Approximate Function. This is expressed visually using partial dependence plot.

```r
# validation
n <- nrow(x = raw_df)
train_idx <- 1:round(n * .8)
train_df  <- raw_df[train_idx]
test_df   <- raw_df[-train_idx]
```

Hyperparameter optimization is essential to improve the model's performance. This process was carried out through repetitive cross-validation using training data. 10 folds-cross validation was repeated a total of 5 times. The random forest model was trained with the hyperparameters optimized in this way. At the same time, the importance of each variable was also estimated using the permutation method.

```r
# tuning hyperparameters
fitControl <- caret::trainControl(  method = "repeatedcv"
                                  , number = 2
                                  , repeats = 2)
fit_rf <- caret::train(  filtration ~ . -batch -group
                       , data = train_df
                       , method = "ranger"
                       , trControl = fitControl
                       , verbose = TRUE
                       )

bestTune <- fit_rf$bestTune

# fit the model
fit_rf <- ranger::ranger(  formula = filtration ~ .-batch -group
                         , data = train_df
                         , num.trees = 1000
                         , importance = "permutation"
                         , mtry = bestTune$mtry
                         , splitrule = bestTune$splitrule
                         , min.node.size = bestTune$min.node.size
                         )

# variable importance
imp_rf <- fit_rf$variable.importance
imp_rf <- imp_rf + abs(x = min(imp_rf))
imp_rf <- data.frame(t(x = imp_rf))
```

The model was evaluated quantitatively and qualitatively. Quantitative evaluation was performed using evaluation indicators such as RMSE, MAE and MAPE, and qualitative evaluation was confirmed through graphs of actual and predicted values.

The quantitative evaluation results are shown in Table 1 below.

```r
pred_rf <- predict(object = fit_rf, data = raw_df )$predictions

eval_whole <- eval_quantitative(  actual = raw_df$filtration
                                , predict = pred_rf
                                , digits = 2
                                )
eval_test  <- eval_quantitative(  actual = raw_df$filtration[-train_idx]
                                , predict = pred_rf[-train_idx]
```

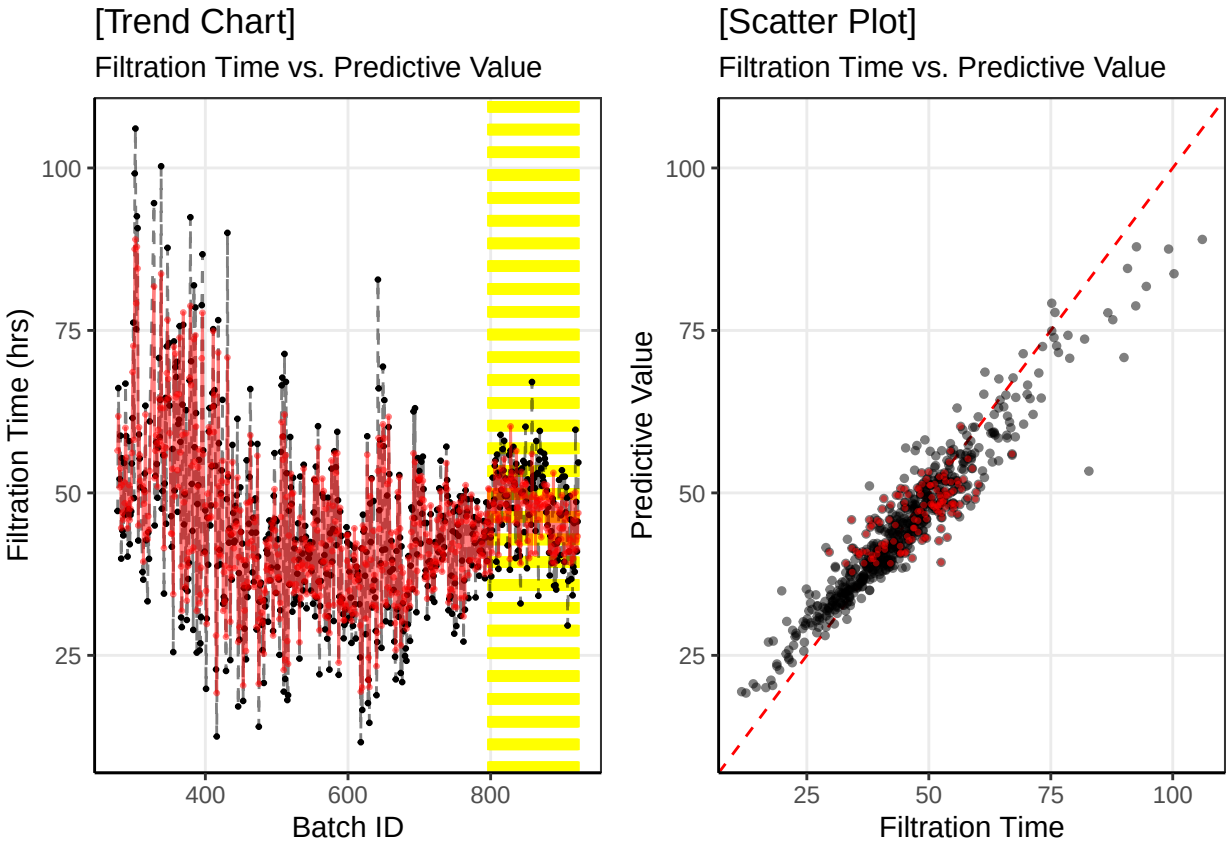
```
                                  , digits = 2
                                  )
pred_df <- data.table::data.table(  batch = raw_df$batch
                                  , filtration = raw_df$filtration
                                  , pred = pred_rf
                                  )
```

Table 1: quantitative evaluation

	MAPE	RMSE	MAE
whole data	8.04	4.53	3.32
test data	9.23	5.15	4.32

Qualitative results are shown in the graph below. These are the trend chart according to the batch and the scatter plot between the actual and predicted values.

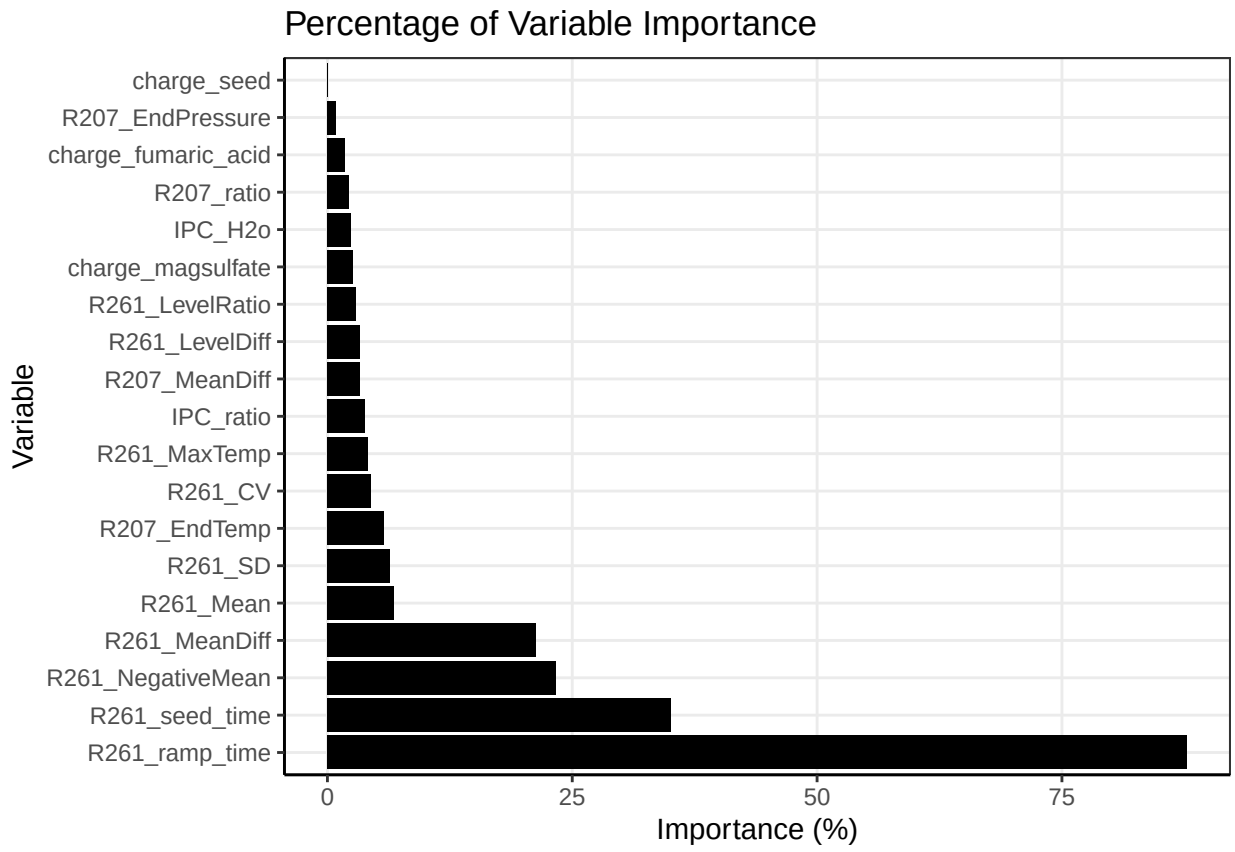
```
p_trend <- pred_df %>%
  ggplot(mapping = aes(x = as.numeric(batch), y = filtration)) +
  geom_vline(xintercept = seq(  from = min(test_df$batch)
                              , to = max(test_df$batch)),
             linetype = "dashed", col = "yellow") +
  geom_point(size = .5) + geom_line(linetype = "dashed", alpha = .5) +
  xlab(label = "Batch ID") + ylab(label = "Filtration Time (hrs)") +
  ggtitle(  label = "[Trend Chart]"
          , subtitle = "Filtration Time vs. Predictive Value") + p
p_trend <- p_trend + geom_point(  mapping = aes(x = as.numeric(batch), y = pred)
                                , col = "red", size = .5, alpha = .5) +
  geom_line(aes(x = as.numeric(batch), y = pred), col = "red", alpha = .5)


p_scatter <- pred_df %>%
  ggplot(mapping = aes(x = filtration, y = pred)) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", col = "red") +
  geom_point(size = 1, alpha = .5) +
  xlab(label = "Filtration Time") + ylab(label = "Predictive Value") +
  ggtitle(  label = "[Scatter Plot]"
          , subtitle = "Filtration Time vs. Predictive Value") +
  scale_x_continuous(limits = range(c(pred_df$filtration, pred_df$pred))) +
  scale_y_continuous(limits = range(c(pred_df$filtration, pred_df$pred))) +
  p
p_scatter <- p_scatter + geom_point(  data = pred_df[-train_idx, ]
                                    , mapping = aes(x = filtration, y = pred)
                                    , col = "red", size = .5, alpha = .5)
gridExtra::grid.arrange(p_trend, p_scatter, nrow = 1)
```



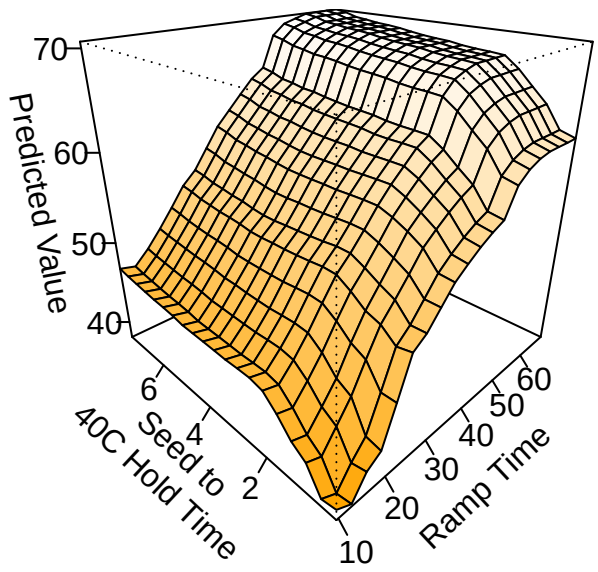
The yellow background in the trend chart is where the test data is located. Also, the black point is the actual value, and the red color is the predicted value. On the other hand, the red dot in the scatter plot is the test data. The red dashed line in the scatter plot corresponds to the case where the predicted value and the actual value are exactly the same.

In this task, it is important to search for factors that affect filtration time rather than predictive power, and to estimate the relationship between the factor and filtration time. First, we looked at the importance of variables based on the model. After that, the influence of the important variables was visualized through partial dependence plot.

```
# variable importance
imp_rf %>%
  gather(key = "Variable", value = "Imp") %>%
  ggplot(aes(x = reorder(Variable, -Imp), y = Imp)) +
  geom_bar(stat = "identity", fill = "black")+
  labs(y = "Importance (%)", x = "Variable") +
  ggtitle(label = "Percentage of Variable Importance") +
  coord_flip() + p
```




```
# partial dependence plot
partial_plot_3d(  object = fit_rf, data = train_df
                , var1 = "R261_ramp_time", var2 = "R261_seed_time"
                , xlab = "Ramp Time"
                , ylab = paste0("Seed to", "\n", "40C Hold Time"))
```



Reference

Q. Zhao and T. Hastie. Causal interpretations of black-box models. Journal of Business & Economic Statistics, pages 1–10, 2019.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. Annals of statistics, pages 1189–1232, 2001.