ReadMe

Step 1: Data Preprocessing & Feature Engineering

First, I deleted useless features, namely “id” and “random_state”. Then I changed “penalty” into dummy variables. After that, I changed “n_jobs = -1” into “16” because -1 means the program use all the CPUs. What’s more, due to the value of “alpha” is too small, I took log10 for it. Besides, I also added three new variables called new1, new2 and new3. The details are as following:

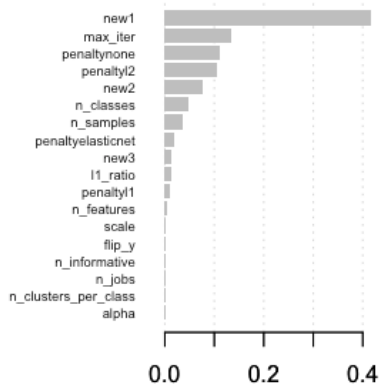
new1 <- (train1$n_samples*train1$n_features)/train1$n_jobs

new2 <- (train1$n_samples*train1$n_features)

new3 <- (train1$n_classes*train1$n_clusters_per_class)

Step 2: Model Building – XGBoost + Lasso

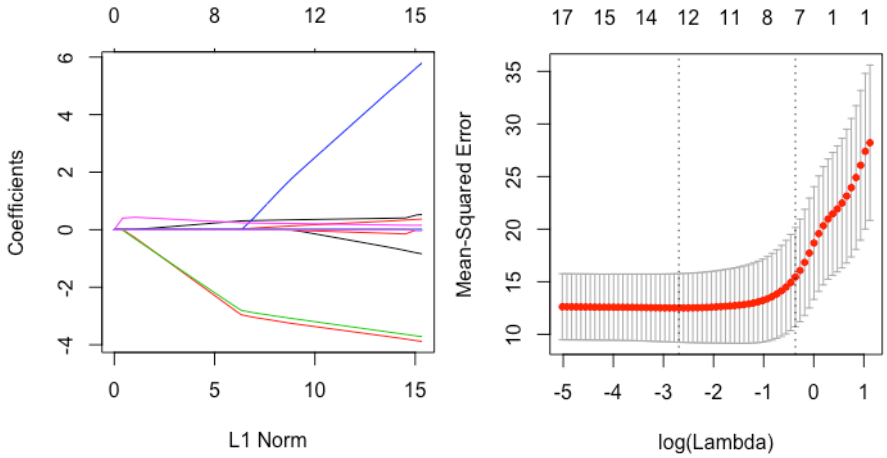
I divided the dataset into training data and validation data two parts with percentage 0.95. First, build the XGBoost model. I used default parameters to fit model for variable selection based on their contribution. We see after “n_feature”, the remaining variables all have less contributions. However, if I delete all of them, may will cause underfitting. So, I printed their importance, and chose a suitable value “0.003”. If their importance larger than 0.003, I kept them as the final feature.



Besides, I build a XGBoost model 1 depended on above features. I also used the

cross validation to find the best parameters which can bring a small “rmse”. After getting parameters, we used them to build the other XGBoost model as the final. However, we also need to change the parameters of model 1 as the same, because we used the prediction from model 1 to find weight with the prediction with Lasso model (details showed in Result part).

Then I build a Lasso model which also used cv to find a best “lambda”.



Step 3: Combine XGBoost & Lasso to Get the Final Result

I used a loop to find the best weight to combine these two models:

```
for (i in 1:length(w)){
   pred <- w[i] * preds.xgb1 + (1 - w[i]) * lasso.pred
   rmse <- mean((subset(train1, select = time)[381:400, ] - pred) ^ 2)
   if (rmse < best_rmse){
      best_rmse <- rmse
      best_rmse_index <- i
   }
}
w_best <- c(w[best_rmse_index], 1 - w[best_rmse_index])
```

And the final result was as **w*preds.xgb2 + (1-w)*lasso.pred1**. If the result less than zero, I used value got from XGBoost model.

Note: Packages used in R

install.packages("caTools")

install.packages("glmnet")

install.packages("caret")

install.packages("gbm")

library(caTools)

library(glmnet)

library(caret)

library(gbm)

library(readr)