Data Analysis Report

(EDA, Preprocess, Model Training and Tuning)

Name: Shi Chen
E-mail: yuejianyingmm@icloud.com

This report is discussing the findings during EDA stage and the strategies using in dealing with the missing values, and regeneration performance by using “glmnet” method under different assumptions during dealing with Ass2Data.

Part 1 Data cleaning

The first thing is to clean the data. In this stage, unifying the missing value placeholders and identifying the missingness of each variable are “Not Applicable” or “Not Available” are the most important things. There are different missing value placeholders, such as “-99”, “--”, “NA”, “”. They should be unified as “NA”. As to the “Not Applicable” missingness, there is an interesting pattern, most “NA” in the “HEALTHCARE_COST” occur in the same observations whose “HEALTHCARE_BASIS” are “FREE”. In fact, it makes sense. Because the “HEALTHCARE_BASIS” is “FREE”, so there is no cost of “HEALTHCARE”. The missing value which caused by “HEALTHCARE_BASIS” is “FREE” of the variable “HEALTHCARE_COST” should be classified as “Not Applicable”, and it is suitable to replace these “NA” by number 0. Others, should keep as “NA”.

Then browsing the entire dataset to know some detail about missingness by using the preliminary cleaning dataset. As shown in figure 1, in order to retain as many observations as possible, it is better to remove excessively missing variables before evaluating the excessive missingness of observations. From the summary, it shows that each column has a different number of missing values except outcome variable “DEATHRATE” and id column “COUNTRY”. Most worth mentioning, the column “AGEMEDIAN” contains 113 missing values and the missingness ratio is 59% (> 50%). Whether to remove it, depends on the importance of this variable. Random Forest can be used to select the very important predictors. The outputs of Random Forest are slightly different, especially the rank of un very important variables. But as can be seen from figure 2, the first two "GOVERNMENT" and "POPDENSITY" never change, they are the very import variables to predict the “DEATHRATE”, which should not be deleted. While, the variable "AGEMEDIAN" should be deleted, because it is not the very important and its missingness are expired 50%.

Figure 1 Missingness distribution

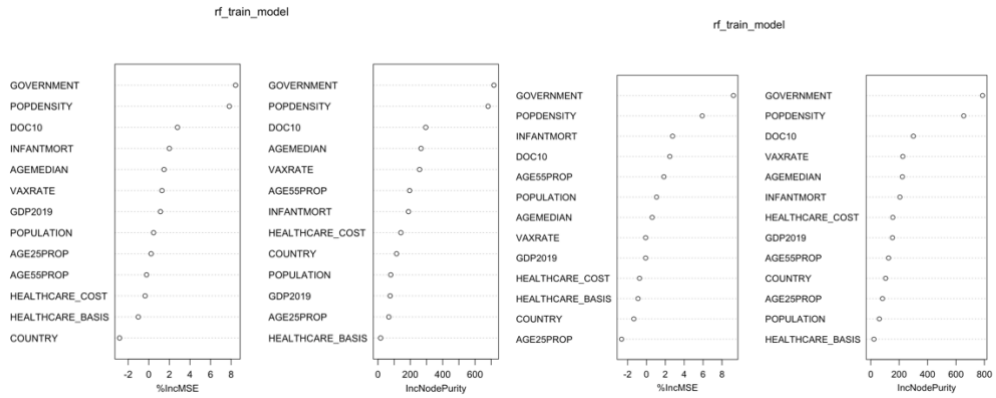

Figure 2 Important predictors (Random Forest output)

From figure 2, it suggests that “GOVERNMENT” and “POPDENSITY” are two very import predictors to predict “DEATHRATE”, their missingness pattern are very important, which may suggest unseen missingness in the future. This part will discuss more after removing the excessively missing observation.

After removing the excessively missing variable "AGEMEDIAN", the observations with missing values in the data set were significantly reduced. By using the 50% threshold, the excessively missing observations are row 28,76,118, and 168. When filter the threshold to 45, and 30, there are some patterns shown in figure 3, the excessive missing observations are grouped around row 100 and row 200.

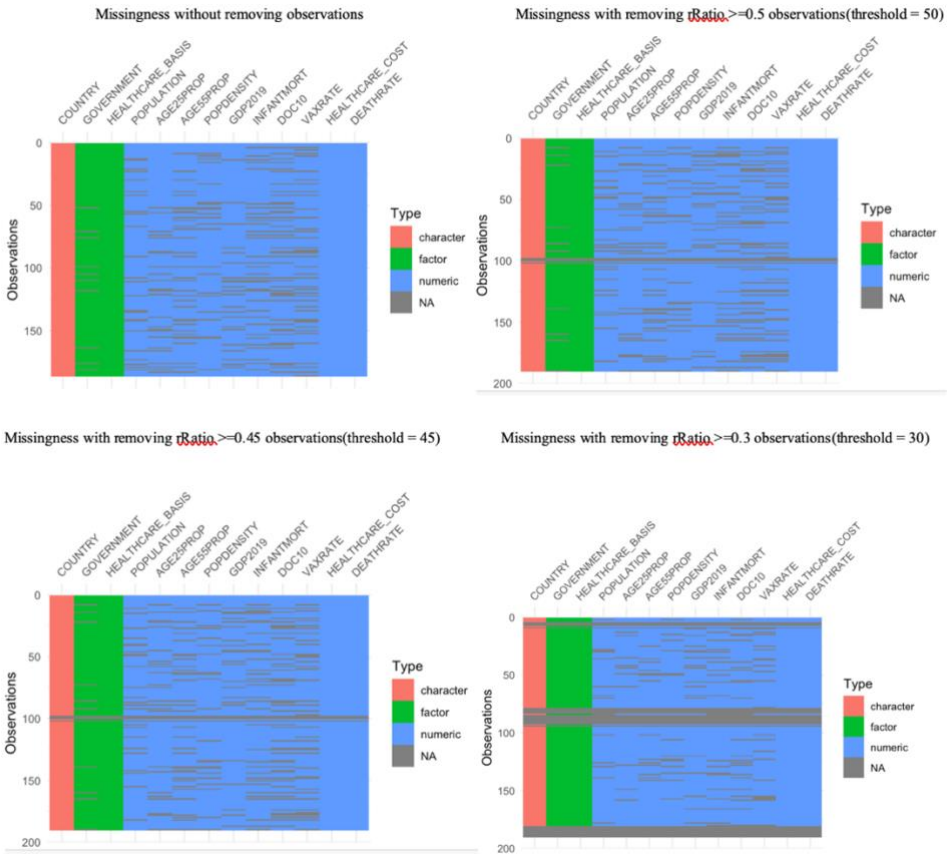

Figure 3 The comparation of the removing observations pattern in different threshold

Part 2 Exploratory Data Analysis (EDA)

In order to remain as more features of the dataset, we use threshold = 50 to remove the four excessive missing observations(row 28,76,118, and 168), the cleaning dataset now have 13 columns variables and 186 rows observations. It is time to do the EDA. In this part, the corrgram matrix, gg_miss, boxplot has been used to show the missingness patterns.

From figure 4, there are five pairs variable missing value correlation. The first one is variable “HEALTHCARE_BASIS” and variable “HEALTHCARE_COST”. The second one is variable “AGE25PRO” and variable “GOVERNMENT”. The third one is variable “INFANTMORT” and variable “GOVERNMENT”. The last two pairs are slightly corelated to each other, which is pair “DOC10” and “AGE55PRO”, and pair “AGE55PRO” and “VAXRATE”.

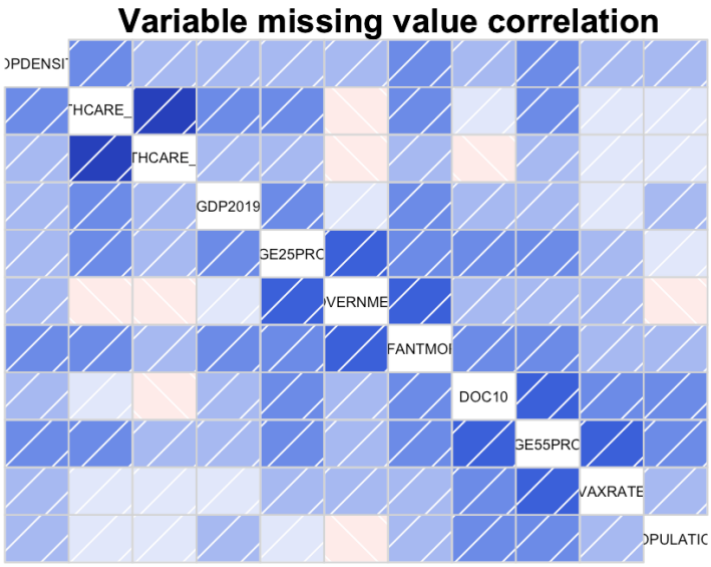

Figure 4 The variable missing value correlation

From figure 5 gg_miss comparation graph, before removing “AGEMEDIAN”, the pattern shows that the excessive missing of “AGEMEDIAN” does not correlated missing with other variables. After removing “AGEMEDIAN” and the four excessive missing observations (row 28,76,118, and 168), and the missing of “DOC10” correspond with other variables missingness more than 21 forms.

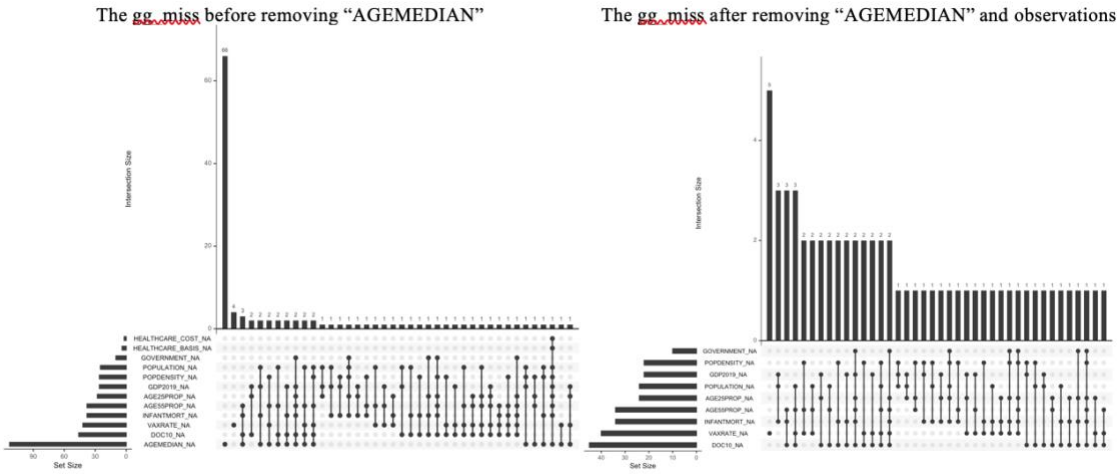

figure 5 gg_miss comparation graph

From figure 6 boxplot, there are plenty of outliers in variable “HEALTHCARE_COST” when using the clean dataset (removing excessive missing variable and observations). Also, the chart shows that there is a big gap among the range of values of different variables.

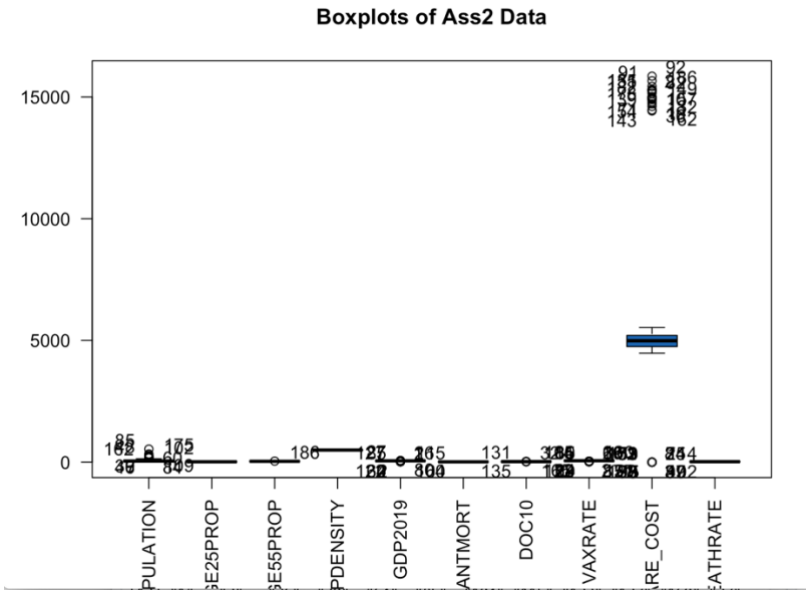

Figure 6 Boxplot of the outliers

Part 3 Strategy and Model

In this part, three different strategies and their performance will be discussed. They are three kinds of preprocessing model pipeline used to imputation missing values and variables standardization.

From the EDA part, it seems that there are some patterns in the variable missingness, to confirm this point After that, it is necessary to think about the unseen missingness in the future, the rpart tree is used to find out the variables which may have effect on other variables’ missingness. From figure 7, it suggests that there is a pattern in the observation’s missing value of variables, and this pattern will still occur in the unseen data. And from the comparation, this pattern becomes weaker after cleaning the data. Take the uncleaning dataset as example, the pattern is that if the “POPULATION” of an observation is larger than 17 million, the missingness will have 76% of 1.3 missingness variables. On the other hand, if the “POPULATION” of an observation is less than 17 million, the missingness will have 25% of 4.6 missingness variables. The total missingness of an observation will be 2.1.

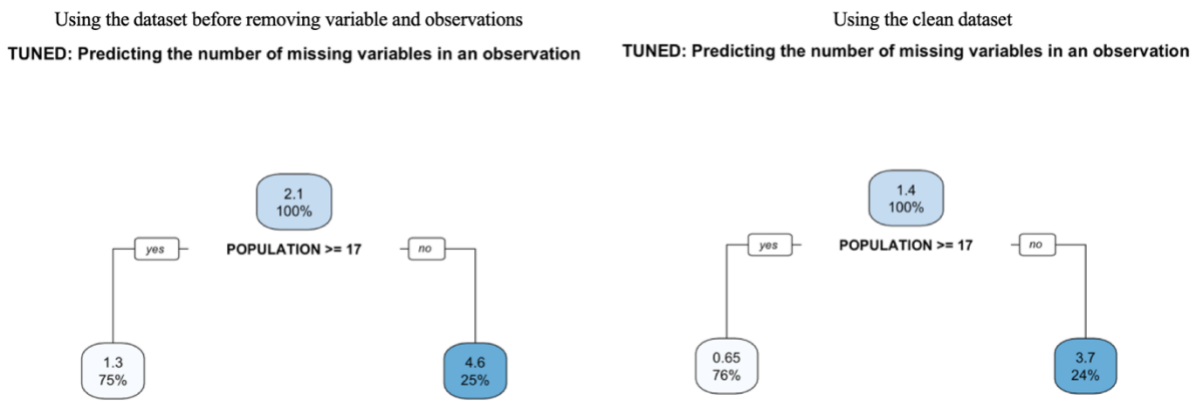

Figure 7 The comparation of TUNED result

From this point of view, in this case, the missingness type of variables is not "Missing Completely At Random" (not MCAR), and after confined with the data collector, the missingness type of variables are not "Missing Not At Random" (not MNAR) either. So, the missingness type of variables should be "Missing At Random" (MAR) with some pattern. Based on this, **the strategy "imputation"** can be used to deal with the missingness before training the data. While, the stratagem "partial delete" is not suitable because it will increase the bias for MAR type variables.

There are many ways to do the imputation, KNN is better in maintain the feature distribution and value relationship. And also, it can do both numeric and mode imputation. The disadvantage of KNN is velocity. It may be not as far as the linear method, especially when dealing with big data. In this case, the dataset is not too big, KNN imputation is suitable.

To fairly compare the performance of models using different strategies, it is need to set.seed() before doing the train_test dataset split and training model. To compare the effect on model performance of train_test dataset split. Two set.seed() groups has been used. Group one is set.seed(1) for the train_test dataset split and set.seed(10) for models training. Group two is set.seed(2) for the train_test dataset split and set.seed(11) for models training.

The strategies discuss as follows are using the set.seed() group one.

Strategy 1:

"Model_0" Only using KNN to do the preprocessing

In this set.seed(2), The dataset has been split into 70% as train data and 30% as test data. And by using set.seed(10) to control the model function. The model_0 comes out in 13 seconds by running in my laptop. And its performance of train data and test data prediction are shown in figure 8. On the left side, the best pair of hyperparameter of glmnet model_0 is when regularization parameter $\lambda = 0.07678$, mixing percentage $\alpha = 1.0$. In this situation, the model RMSE of train data is approximately 3.64. And the RMSE of test target prediction is 2.67.

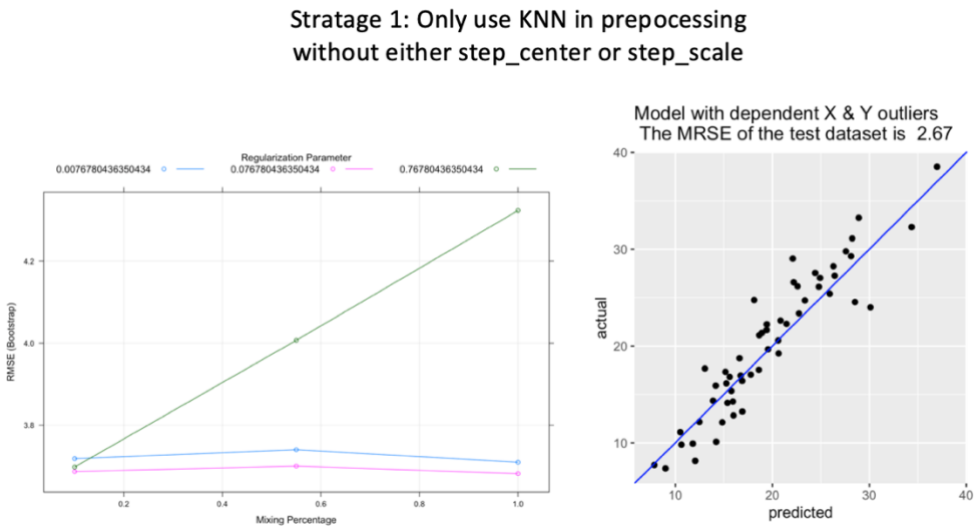

Figure 8 The performance of strategy 1

As mentioned before, in this case the variables are measured at different scales, for example the values of "HEALTHCARE_COST" are much higher than the values of other variables. The output coefficients of the training model may mislead the contribution of each predictor. So variable standardization can be applicated before training the model. Strategy 2 and strategy 3 will add "Scale" and "Center".

Strategy 2:

"Model" use KNN, step_center and step_scale with the order that step_center and step_scale follow KNN

Still using set.seed (2) for 70% train data and 30% test data split. And set.seed (10) to fair control the model function. The model comes out in 23 seconds by running in the same environment. Its performance of train data and test data prediction are shown in figure 9. By accident, the best pair of hyperparameter of glmnet model is also when regularization parameter $\lambda = 0.07678$, mixing percentage $\alpha = 1.0$. In this situation, the model RMSE of train data is approximately 3.64. And the RMSE of test target prediction is also 2.67. (By accident)

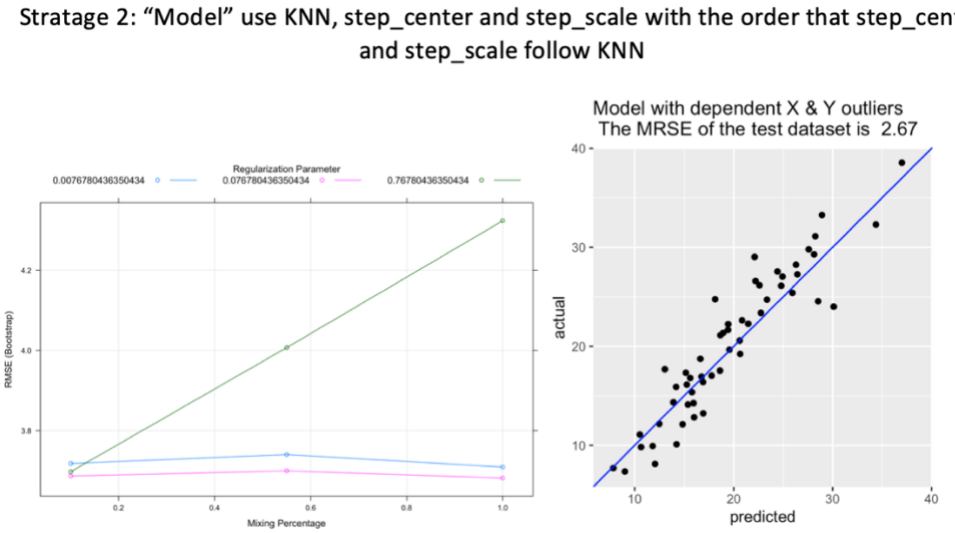

Figure 9 The performance of strategy 2

Strategy 3:

“Model_1” use KNN, step_center and step_scale, but shift the order, KNN comes after step_center and step_scale

Still using set.seed (2) for 70% train data and 30% test data split. And set.seed (10) to fair control the model function. The model_1 comes out in 17 seconds by running in the same environment. Its performance of train data and test data prediction are shown in figure 10. By accident, the best pair of hyperparameter of glmnet model is also when regularization parameter $\lambda = 0.07678$, mixing percentage $\alpha = 1.0$. In this situation, the model RMSE of train data is approximately 3.64 (as same as the last two in this particular case). And the RMSE of test target prediction is 2.82.

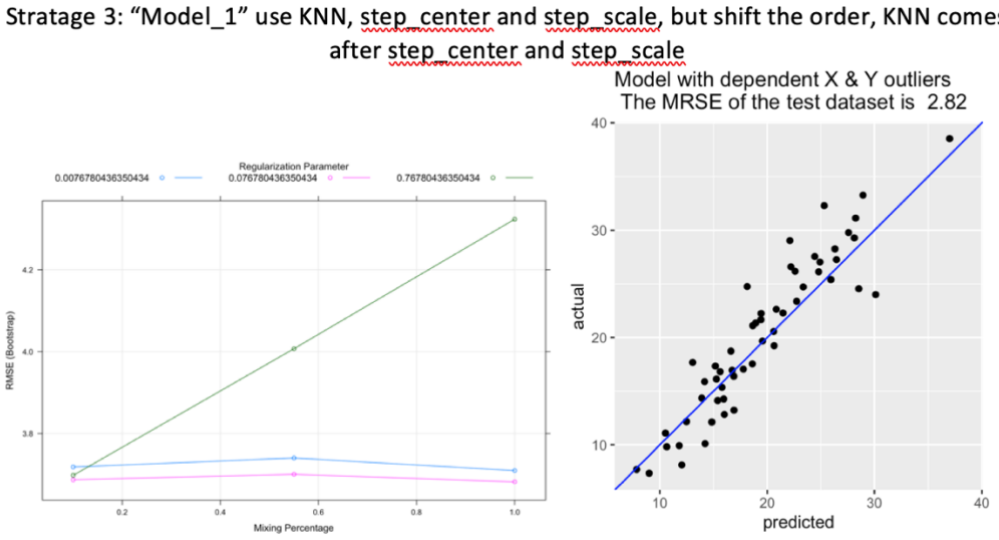

Figure 10 The performance of strategy 3

In this particular set.seed (2) and set.seed(10) case, the performances of different models are quite similar. In this particular case, variable standardization did not improve the model performance significantly. The performance rank is Strategy 2 = Strategy 1 > Strategy 3.

When changing the set.seed () group into group two which is set.seed(2) for the train_test dataset split and set.seed (11) for models training. There are some different conclutions. As shown in figure11, the best pairs of hyperparameter of glmnet models are the same even by using the different strategies, the RMSE of test target predictions by using different strategies are different this time, they are $\mathrm{RMSE}_{test} = 2.9$ for preprocessing strategy 1(only KNN), $\mathrm{RMSE}_{test} = 3.76$ for preprocessing strategy 2(KNN before standardization), and $\mathrm{RMSE}_{test} = 2.89$ for preprocessing strategy 3(standardization before KNN). In this case the performance rank is Strategy 3 > Strategy 1 > Strategy 2.

Each time, changing the train_test group and changing the model set.seed, it will give different conclusion. It needs to study more deeply.

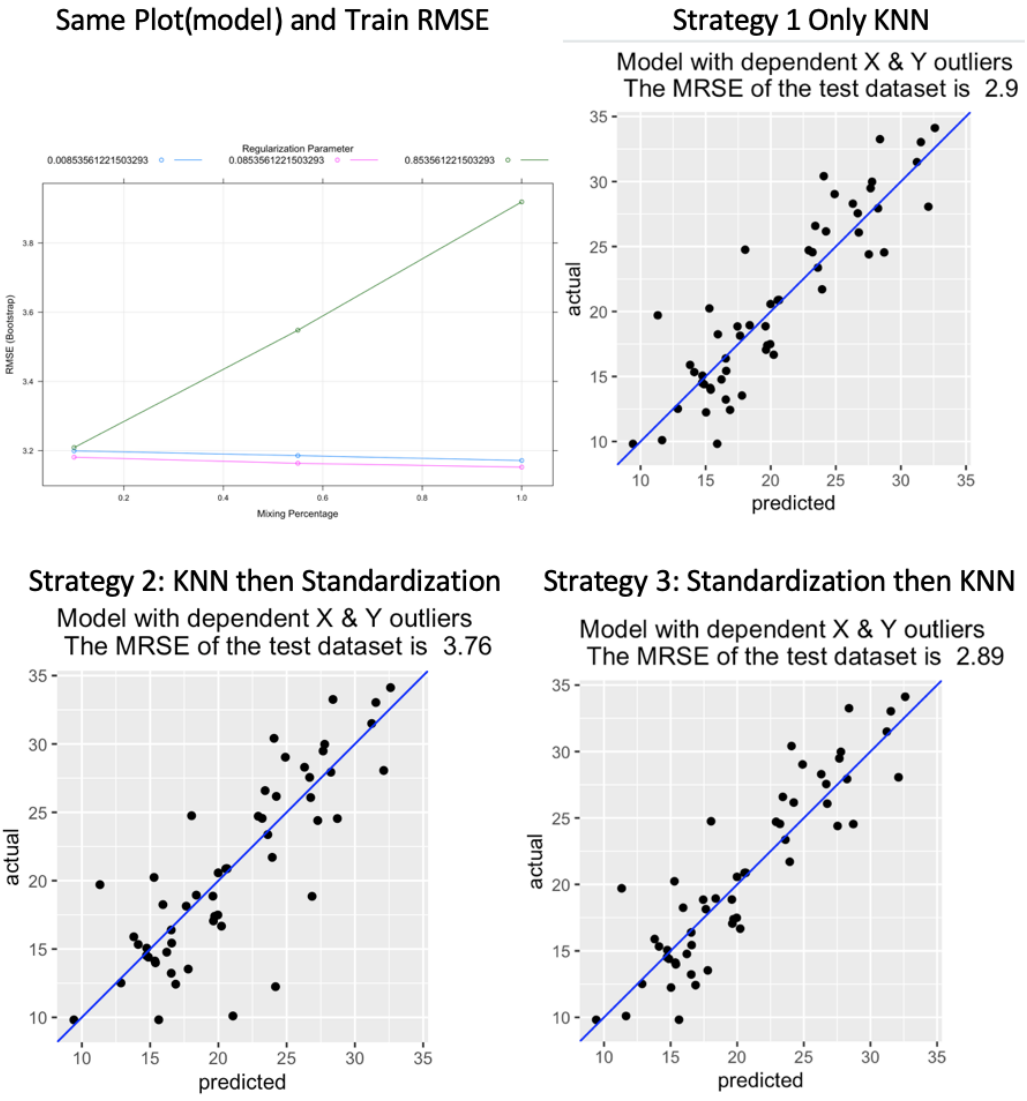

Figure 11 The performance of another train_test dataset group

According to above, the best performance is using group 1 set.seed() and Strategy 2 to train the model. Using this model, the "modelbased outliers" plots at different IQR multiplier is shown in Figure 12. When the coef = 1.5, there are some outliers very close to the tail which may be misclassified as outliers. To remove theses spots by increasing the coef to 2, there are

only three outliers left. From this point of view, using coef = 2 (still reasonable), the modelbased outliers are not too much. The outliers of dataset (as shown in figure 7) can follow the model's pattern, they share the features of the whole dataset. It confirms that outliers are not need to be removed in this case.

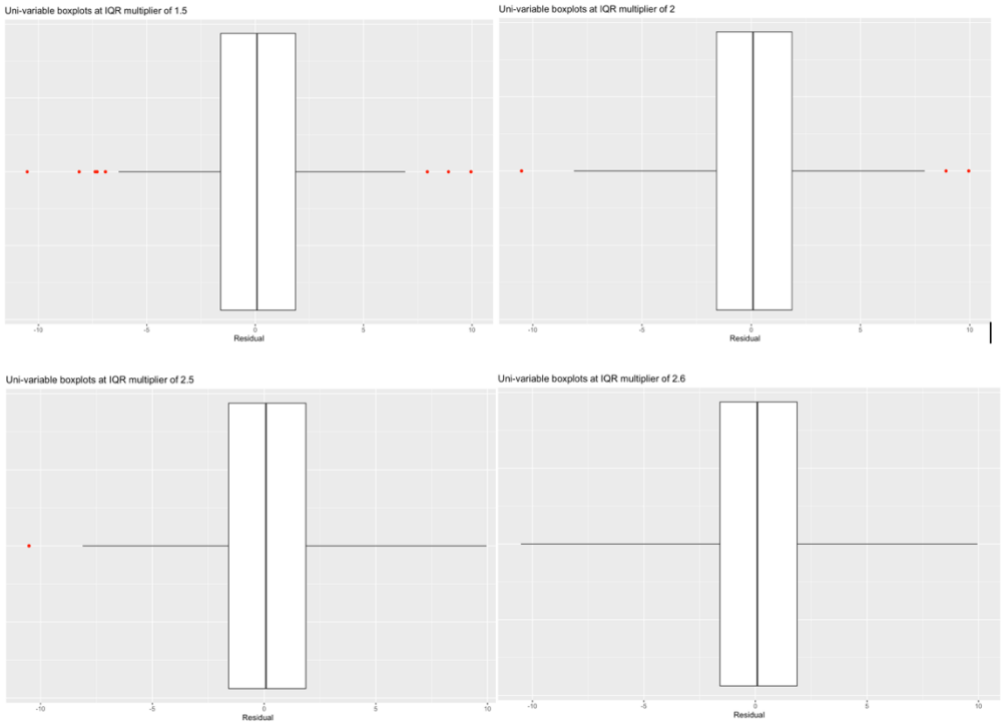

Figure 12 Boxplots of the modelbased outliers change

To sum up, before starting to deal with the data, it is important to browse the data at first. Checking the data type, the missing value, the outliers then choose the right order and the reasonable way to do the data cleaning (replace the missing value placeholder and find the pattern between variables missingness, remove excessive missing variables and observations but not outliers).

Then using the clean data to do the EDA and pay attention to the missing pattern and outliers. Keep thinking about the missingness type and whether the missingness will occur in the unseen data.

After that, choosing the right way to do the preprocessing, imputation or partly delete. If it is suitable to do imputation, still thinking about which method is suitable for which kind of missing values. This will keep the model from misleading. The Normalization and Scaling is very useful to deal with the ununiform-scale variables. It is suggested to be used when there are variables of different magnitudes appear in predictors. There are many factors influence the performance of a model. For example, the train_test data split, the methods or models and their orders used in pre-processing. After training the model, it is necessary to review the modelbased outliers, it will help to find out whether the outliers feature is caught by the model.