Protein Project

Bowen Zhang

Contents

1 Summary

Motivated by the data set of the COVID-19 protein 3D structure, I applied my knowledge of multi-linear regression model and fit the mathematical model.The data set contains 1946 samples of computer generated structures for the COVID-19 spike protein. Each sample has 685 characteristics (explanatory variables) about our protein structure. We will train our regression model on these 1946 samples to help us with our prediction on the accuracy of the protein structure given the corresponding data of our target protein structure.The response of our model is accuracy, which is a measure of how close that computer-generated protein structure is to a known benchmark structure. Removal of Multicollinearity, Model Selection, Prevention of Overfitting and many other techniques have been applied while training and cross validating the model. In the end, we chose the model acquired from backward selection since it gives us the smallest residual mean standard prediction error. The following sections will explain the process in more details.

2 Exploratory analysis of dataset

Before actually fitting the model, I tried to fit the model with all the 685 predictors. I noticed that in the estimates of our model coefficients, there are some coefficients hold the value NA(null). This means that there exists perfect multicolinearity. For example, suppose that initially, we have our full model:

$$Y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \epsilon_i$$

There may exists perfect multicollinearity with two explanatory variables:

$$X_{i1} = \alpha_0 + \alpha_1 X_{i2}$$

Hence, we need to remove such variables and keep removing the variables that might have serious multicollinearity.

3 Methods

In the following section, we will discuss the analysis and techniques I used to obtain my final model. After loading the data set into our model training schema, we need to perform different techniques to filter out some unnecessary predictors to remove multicolinearity or the possibility that certain predictors don't explain model very well. We will start with our first filter - Removal of Multicolinearity.

3.1 Removal of Multicollinearity

In this step, our goal is to remove the predictors from our model where these predictor have high Variance Inflation Factor (VIF). VIF for an variables explains how much can this predictor be explained by other predictos. For example, we have our full model:

$$Y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \epsilon_i$$

Then, VIF_1 corresponds to the following model:

$$X_{i1} = \alpha_0 + \alpha_2 x_{i2} + ... + \alpha_p x_p + \epsilon_i'$$

We have that $VIF_j = \frac{1}{1-R_j^2}$, where R_j^2 is the R^2 value for our new model (using other predictors to explain the current predictor). If the R_j^2 is high, then VIF_j is high. With high value of R_j means that other predictors explain our current predictor well, which is not what we want because this might cause serious multicollinearity. Hence, we can filter the predictors based on high value of VIF.

Note that in section 2, we discussed the existence of perfect multicollinearity. Hence, we need to remove those predictors which has value of NA for corresponding coefficients. In this step, we found scArgN_bbO_short and scArgN_bbC_medshort has the above behavior. We removed these two predictor here.

Then, we can just loop through all the variables and filter them based on their corresponding VIF values and our threshold is 10. If the $VIF > 10$, then we remove the predictor from our model to remove the possibility of the multicollinearity.

After removing the above predictors, we have 569 predictors left. Then, we can move on to preforming model selection.

3.2 Model Selection

Now the problem is given 569 predictors, we want to find the best models(select the best model based on our criteria). In the following subsections, we will use the following model selection techniques: Forward Selection, Backward Elimination, Forward-Backward Selection and Iterative Conditional Minimization (ICM). We will split the data in to two groups with 80% and 20% of the data each. We will use the 80% group to train our model and use the rest 20% group as our validation set. After executing the algorithm and we use the output model to fit the validation set. Then, we calculate the corresponding ******Prediction Error. Based on the ********prediction Error for each model, we will select model that has the best predictive performance.

3.2.1 Forward

For the Forward Selection, the idea is that we start with an empty model, then we keep adding appropriate variables. There are some questions need to be resolved:

1. What do we mean by adding "appropriate" variables?

2. What's the result and how do we interpret and use the result?

We will first answer question 1 by walking through our forward model selection algorithm. We chose Bayesian Information Criterion (BIC) as model selection criterion. We also executed our forward BIC selection algorithm with penalty doubled.

The algorithm starts with the empty model. Then, for ith iteration, we have $p+1-i$ predictors to select from. For example, in our first iteration, we have $p+1-1 = p$ predictors to choose from. We choose 1 predictor from these predictors and add it to our model such that this predictor gives us the best (lowest) BIC value. Then, we repeat the process, until we can't find any variable to improve our model performance in terms of the BIC criterion. The worst case runtime of our algorithm is $O(p^2)$.

After our algorithm terminates, our forward BIC selection algorithm gives us a model m_1 with 91 predictors. We will calculate the Root Mean Squared Error(RMSE) of our model m_1 using the validation set. Then, we can evaluate our model based on this predictive performance.(The resulting model m_1 is included in the Appendix because of the large size of the model and the limited space)

For model m_1, the RMSE for training set is: $RMSE_{train} = 0.4960$, the RMSE for validation set is: $RMSE_{valid} = 0.6601$. Note that the difference between the the validation set and the training set is relatively large in terms of the RMSE. Hence, our model might have overfitting problems.

3.2.2 Backward Elinimation

Similar to forward selection, backward selection procedure we start with our full model. Note that in this case, our full model refers to the model that has already removed the predictors with high VIFs. Then, for ith iteration, we have $p + 1 - i$ predictors to choose from to eliminate. For example, in the first iteration, we have to choose one predictor from the total p predictors based on the BIC criterion. In each iteration, we choose to eliminate the predictor which will help us improve the criterion after being removed. Note that the runtime for our backward elimination algorithm is also $O(p^2)$.

After our algorithm terminates, our backward BIC selection algorithm gives us a model m_3. We will calculate the Root Mean Squared Error(RMSE) of our model m_3 using the validation set. Then, we can evaluate our model based on this predictive performance.(The resulting model $m31$ is included in the Appendix because of the large size of the model and the limited space)

For model m_3, the RMSE for training set is: $RMSE_{train} = 0.4777$, the RMSE for validation set is: $RMSE_{valid} = 0.6588$.

3.2.3 Forward-Backward

For our third model, we perform the Forward-Backward Model Selection algorithm. This step we basically combined the techniques from Forward Selection and Backward Elimination.

In this model selection procedure, we start from forward selection. In each iteration, if we have already had k variables in the model, we have two phases:

1. Backward Elimination: Fit k models with $k - 1$ variables, then we check if removing any predictor helps us with improving the criterion, we then remove the predictor that improves our model the most.

2. Forward Selection: Fit $p-k$ models with $k+1$ variables, then we check by adding any predictor helps us with improving the criterion, we then add the predictor that improves our model the most.

Note that the runtime for this algorithm is also $O(p^2)$ since we are checking every possibilities at each iteration.

After our algorithm terminates, our Forward-Backward BIC selection algorithm gives us a model m_4 with 91 predictors. We will calculate the Root Mean Squared Error(RMSE) of our model m_4 using the validation set. Then, we can evaluate our model based on this predictive performance.(The resulting model $m4$ is included in the Appendix because of the large size of the model and the limited space)

For model m_4, the RMSE for training set is: $RMSE_{train} = 0.4919$, the RMSE for validation set is: $RMSE_{valid} = 0.6560$.

3.2.4 Iterative Conditional Minimization (ICM)

Now, we move on to the Iterative Conditional Minimization (ICM) model selection technique. Note that the special part of this algorithm is its stochasticity: We randomly shuffle the predictors: $x_1...x_p$ into $x_{(1)}...x_{(p)}$. Then, we execute the following algorithm.

We start with our model with just the intercept. Then, we loop for p iterations, then we let S denote the set of predictors currently in the model. We have the following two directions:

1. If $x_{(j)}$ is not in S, we fit the model with S and $x_{(j)}$. If the addition of $x_{(j)}$ improves our criterion, then we add our new predictor $x_{(j)}$ in our selected predictors set S.

2. If $x_{(j)}$ is in S, we git the model removing $x_{(j)}$. If by excluding the variable $x_{(j)}$, it improves the criterion, then we remove our predictor $x_{(j)}$ from our selected predictors set S.

We are checking each possibilities and for each predictor we have the choice to add it in or leave it out. We repeat the above for loop until no variables are added or removed from our model throughout the entire for loop.

After our algorithm terminates, our forward BIC selection algorithm gives us a model m_2 with 116 predictors. We will calculate the Root Mean Squared Error(RMSE) of our model m_2 using the validation set. Then, we can evaluate our model based on this predictive performance.(The resulting model m_2 is included in the Appendix because of the large size of the model and the limited space)

For model m_2, the RMSE for training set is: $RMSE_{train} = 0.4358$, the RMSE for validation set is: $RMSE_{valid} = 0.6341$. Note that the difference between the the validation set and the training set is relatively large in terms of the RMSE. Hence, our model might have overfitting problems.

3.3 Cross Validation K-fold Training

Note that in the past model selection sections, we performed model selection using Forward Selection, Backward Elimination and many other techniques. However, note that after getting the model as the output of our algorithm, we only used one single number $RMSE$ to evaluate our model's predictive performance, which could be improved. Another point we need to notice that, in the previous training techiniques, we didn't actually used all the data, we only used 80% of the data to train our model. To improve the sitation, we decide to use the cross validation k-fold to train and evaluate our model.

To address the above problem, we divide out data into k roughly equally sized sets randomly. Then we loop k times, in each iteration i, we use ith set as our cross validation set, and use the rest of the data to train our data. Then, after looping through k iteration, we will have k estimates of prediction error. Then, we take the average of them as our Root Mean Squared Prediction

Error($RMSPE$).

In our case, we choose $k = 5$. After performing the cross validation on 5 folds, we have:

Forward	Backward	Both Dirction	ICM
0.5325	0.5189	0.5286	0.5095

4 Results and Discussion

Let's take a look at the predictive evaluation on the selected models before we used the cross validation k-fold training techniques. We have:

	Forward	Backward	Both Dirction	ICM
train	0.4960	0.4777	0.4919	0.4358
valid	0.6601	0.6588	0.6560	0.6341

Hence, we have that ICM approach gives us the best predictive performance based on the RMSPE values shown in the above tables. Hence, our best model is m_2. (Check appendix for more details about model m_2). From model m_2, we listed the top 10 predictors that have the most effect on accuracy:

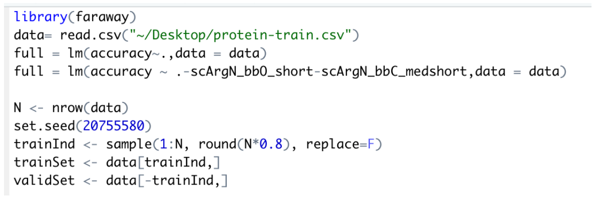
Predictor	Coefficients
scArgN_bbCA_medshort	3.3790668
scArgN_bbCA_medlong	2.2620308
scArgN_bbO_medshort	1.5213456
scArgN_bbO_medlong	1.2613332
scArgN_bbCA_long	0.3795392
aliph1HC_bbProN_medlong	0.3155181
aliph1HC_sulfur_short	0.2944100
scArgN_bbC_vlong	0.2540413
aromaticC_scArgN_medshort	0.2255396
aliph3HC_scArgN_vlong	0.2144764

Hence, we can see that atom type scArgN appeared a lot in the attributes that has large-valued coefficients. Note that when atom types are scArgN and bbCA, medium long and short, they affect the most. Hence, this could be of improtance to our prediction of COVID virus protein detection procedure.

After calculating the corresponding R^2 value, we have that our model $R^2 = 0.9136$, which is really good. Hence, I would use model m_2 as our output of ICM approach. (See more details about m_2 in appendix). If we use our model to fit the new data, we will have the $RSME$ roughly equal to 0.5095. Since, our cross validation is a simulation of our test data set.

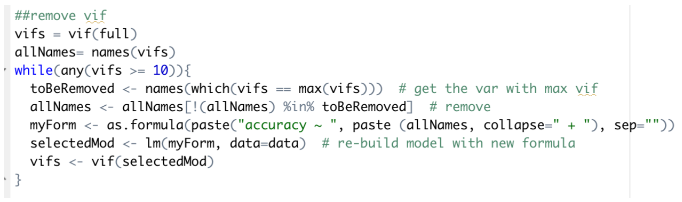
5 Appendix

5.1 Code for Data Prepartion



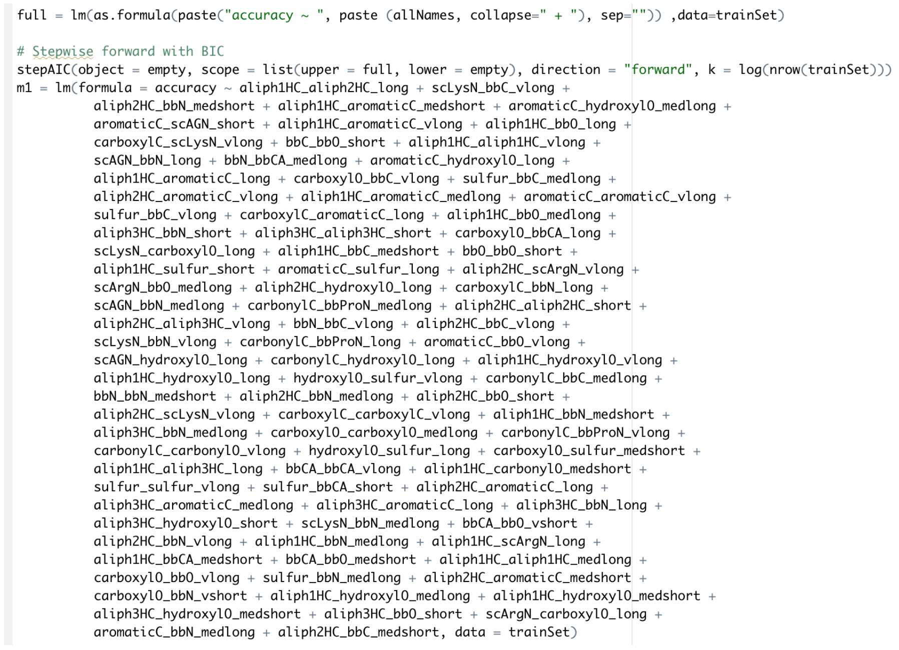
```
library(faraway)
data= read.csv("~/Desktop/protein-train.csv")
full = lm(accuracy~.,data = data)
full = lm(accuracy ~ .-scArgN_bbO_short-scArgN_bbC_medshort,data = data)

N <- nrow(data)
set.seed(20755580)
trainInd <- sample(1:N, round(N*0.8), replace=F)
trainSet <- data[trainInd,]
validSet <- data[-trainInd,]
```

5.2 Code for Removal of Multicollinearity



```
##remove vif
vifs = vif(full)
allNames= names(vifs)
while(any(vifs >= 10)){
  toBeRemoved <- names(which(vifs == max(vifs)))  # get the var with max vif
  allNames <- allNames[!(allNames) %in% toBeRemoved]  # remove
  myForm <- as.formula(paste("accuracy ~ ", paste (allNames, collapse=" + "), sep=""))
  selectedMod <- lm(myForm, data=data)  # re-build model with new formula
  vifs <- vif(selectedMod)
}
```

5.3 Code for Forward Selection m_1



```
full = lm(as.formula(paste("accuracy ~ ", paste (allNames, collapse=" + "), sep="")) ,data=trainSet)

# Stepwise forward with BIC
stepAIC(object = empty, scope = list(upper = full, lower = empty), direction = "forward", k = log(nrow(trainSet)))
m1 = lm(formula = accuracy ~ aliph1HC_aliph2HC_long + scLysN_bbC_vlong +
          aliph2HC_bbN_medshort + aliph1HC_aromaticC_medshort + aromaticC_hydroxylO_medlong +
          aromaticC_scAGN_short + aliph1HC_aromaticC_vlong + aliph1HC_bbO_long +
          carboxylC_scLysN_vlong + bbC_bbO_short + aliph1HC_aliph1HC_vlong +
          scAGN_bbN_long + bbN_bbCA_medlong + aromaticC_hydroxylO_long +
          aliph1HC_aromaticC_long + carboxylO_bbC_vlong + sulfur_bbC_medlong +
          aliph2HC_aromaticC_vlong + aliph1HC_aromaticC_medlong + aromaticC_aromaticC_vlong +
          sulfur_bbC_vlong + carboxylC_aromaticC_long + aliph1HC_bbO_medlong +
          aliph3HC_bbN_short + aliph3HC_aliph3HC_short + carboxylO_bbCA_long +
          scLysN_carboxylO_long + aliph1HC_bbC_medshort + bbO_bbO_short +
          aliph1HC_sulfur_short + aromaticC_sulfur_long + aliph2HC_scArgN_vlong +
          scArgN_bbO_medlong + aliph2HC_hydroxylO_long + carboxylC_bbN_long +
          scAGN_bbN_medlong + carbonylC_bbProN_medlong + aliph2HC_aliph2HC_short +
          aliph2HC_aliph3HC_vlong + bbN_bbC_vlong + aliph2HC_bbC_vlong +
          scLysN_bbN_vlong + carbonylC_bbProN_long + aromaticC_bbO_vlong +
          scAGN_hydroxylO_long + carbonylC_hydroxylO_long + aliph1HC_hydroxylO_vlong +
          aliph1HC_hydroxylO_long + hydroxylO_sulfur_vlong + carbonylC_bbC_medlong +
          bbN_bbN_medshort + aliph2HC_bbN_medlong + aliph2HC_bbO_short +
          aliph2HC_scLysN_vlong + carboxylC_carboxylC_vlong + aliph1HC_bbN_medshort +
          aliph3HC_bbN_medlong + carboxylO_carboxylO_medlong + carbonylC_bbProN_vlong +
          carbonylC_carbonylO_vlong + hydroxylO_sulfur_long + carboxylO_sulfur_medshort +
          aliph1HC_aliph3HC_long + bbCA_bbCA_vlong + aliph1HC_carbonylO_medshort +
          sulfur_sulfur_vlong + sulfur_bbCA_short + aliph2HC_aromaticC_long +
          aliph3HC_aromaticC_medlong + aliph3HC_aromaticC_long + aliph3HC_bbN_long +
          aliph3HC_hydroxylO_short + scLysN_bbN_medlong + bbCA_bbO_vshort +
          aliph2HC_bbN_vlong + aliph1HC_bbN_medlong + aliph1HC_scArgN_long +
          aliph1HC_bbCA_medshort + bbCA_bbO_medshort + aliph1HC_aliph1HC_medlong +
          carboxylO_bbO_vlong + sulfur_bbN_medlong + aliph2HC_aromaticC_medshort +
          carboxylO_bbN_vshort + aliph1HC_hydroxylO_medlong + aliph1HC_hydroxylO_medshort +
          aliph3HC_hydroxylO_medshort + aliph3HC_bbO_short + scArgN_carboxylO_long +
          aromaticC_bbN_medlong + aliph2HC_bbC_medshort, data = trainSet)
```

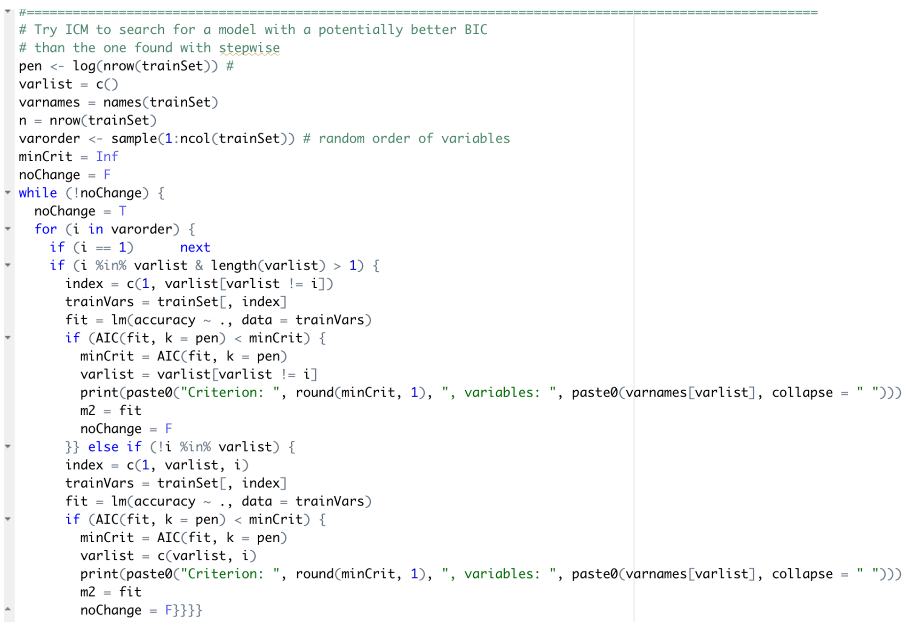
5.4 Code for Backward Elimination m_3

```
""""
# Backward Elimination
stepAIC(object =full, scope = list(upper = full, lower = empty), direction = "backward", k = log(nrow(trainSet)))
m3 = lm(formula = accuracy ~ carbonylC_aromaticC_short + carbonylC_scLysN_vlong +
          carbonylC_bbProN_medlong + carbonylC_bbProN_long + carbonylC_bbProN_vlong +
          carbonylC_hydroxylO_long + carbonylC_bbC_medlong + carboxylC_carboxylC_vlong +
          carboxylC_hydroxylO_short + carboxylC_bbN_long + carboxylC_bbC_medlong +
          aliph1HC_aliph1HC_long + aliph1HC_aliph1HC_vlong + aliph1HC_aliph2HC_long +
          aliph1HC_aliph3HC_short + aliph1HC_aliph3HC_medlong + aliph1HC_aliph3HC_vlong +
          aliph1HC_aromaticC_medshort + aliph1HC_aromaticC_medlong +
          aliph1HC_aromaticC_long + aliph1HC_aromaticC_vlong + aliph1HC_scArgN_long +
          aliph1HC_bbProN_medlong + aliph1HC_hydroxylO_medshort + aliph1HC_hydroxylO_medlong +
          aliph1HC_hydroxylO_long + aliph1HC_hydroxylO_vlong + aliph1HC_carbonylO_medshort +
          aliph1HC_carboxylO_vlong + aliph1HC_sulfur_short + aliph1HC_bbN_medshort +
          aliph1HC_bbCA_medshort + aliph1HC_bbC_medshort + aliph1HC_bbO_short +
          aliph1HC_bbO_medshort + aliph1HC_bbO_medlong + aliph1HC_bbO_long +
          aliph2HC_aromaticC_medshort + aliph2HC_aromaticC_vlong +
          aliph2HC_scArgN_vlong + aliph2HC_carbonylO_vlong + aliph2HC_sulfur_short +
          aliph2HC_bbN_medshort + aliph2HC_bbN_medlong + aliph2HC_bbC_long +
          aliph2HC_bbC_vlong + aliph2HC_bbO_vlong + aliph3HC_aliph3HC_short +
          aliph3HC_aliph3HC_long + aliph3HC_aliph3HC_vlong + aliph3HC_aromaticC_medlong +
          aliph3HC_aromaticC_long + aliph3HC_hydroxylO_short + aliph3HC_hydroxylO_medshort +
          aliph3HC_bbN_short + aliph3HC_bbN_medlong + aliph3HC_bbN_long +
          aliph3HC_bbC_vlong + aliph3HC_bbO_medshort + aliph3HC_bbO_medlong +
          aromaticC_aromaticC_vlong + aromaticC_hydroxylO_medlong +
          aromaticC_hydroxylO_long + aromaticC_sulfur_medlong + aromaticC_bbC_short +
          aromaticC_bbO_vshort + aromaticC_bbO_vlong + scAGN_bbProN_medshort +
          scAGN_hydroxylO_long + scAGN_carbonylO_medshort + scAGN_bbN_medlong +
          scAGN_bbN_long + scLysN_carboxylO_long + scLysN_bbN_medlong +
          scLysN_bbN_long + scLysN_bbN_vlong + scLysN_bbC_vlong + scArgN_carboxylO_long +
          scArgN_bbO_medlong + bbProN_bbN_long + bbProN_bbCA_medshort +
          hydroxylO_sulfur_long + hydroxylO_bbCA_medlong + carbonylO_bbN_medlong +
          carbonylO_bbCA_short + carbonylO_bbCA_medshort + carboxylO_carboxylO_long +
          carboxylO_bbN_vlong + carboxylO_bbO_vlong + sulfur_bbCA_short +
          sulfur_bbC_medlong + sulfur_bbC_vlong + sulfur_bbO_long +
          bbN_bbN_medshort + bbN_bbCA_medlong + bbN_bbC_vshort + bbN_bbC_vlong +
          bbN_bbO_medshort + bbCA_bbCA_vlong + bbCA_bbO_vshort + bbCA_bbO_medshort +
          bbC_bbO_short + bbO_bbO_vshort, data = trainSet)
```

5.5 Code for Forward-Backward Selection m_4

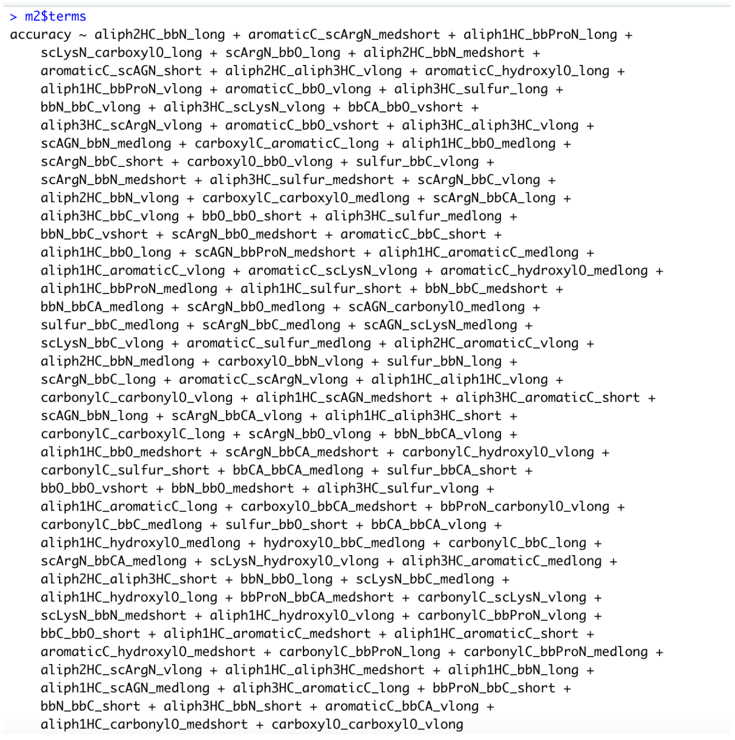
```
########### Forward-Backward BIC
stepAIC(object = empty, scope = list(upper = full, lower = empty), direction = "both", k = log(nrow(trainSet)))
m4  = lm(formula = accuracy ~ aliph1HC_aliph2HC_long + scLysN_bbC_vlong +
           aliph2HC_bbN_medshort + aliph1HC_aromaticC_medshort + aromaticC_hydroxylO_medlong +
           aromaticC_scAGN_short + aliph1HC_aromaticC_vlong + aliph1HC_bbO_long +
           bbC_bbO_short + aliph1HC_aliph1HC_vlong + scAGN_bbN_long +
           bbN_bbCA_medlong + aromaticC_hydroxylO_long + aliph1HC_aromaticC_long +
           sulfur_bbC_medlong + aliph2HC_aromaticC_vlong + aliph1HC_aromaticC_medlong +
           aromaticC_aromaticC_vlong + sulfur_bbC_vlong + carboxylC_aromaticC_long +
           aliph1HC_bbO_medlong + aliph3HC_bbN_short + aliph3HC_aliph3HC_short +
           carboxylO_bbCA_long + scLysN_carboxylO_long + bbO_bbO_short +
           aliph1HC_sulfur_short + aromaticC_sulfur_long + aliph2HC_scArgN_vlong +
           scArgN_bbO_medlong + carboxylC_bbN_long + scAGN_bbN_medlong +
           carbonylC_bbProN_medlong + aliph2HC_aliph2HC_short + aliph2HC_aliph3HC_vlong +
           bbN_bbC_vlong + aliph2HC_bbC_vlong + scLysN_bbN_vlong + carbonylC_bbProN_long +
           aromaticC_bbO_vlong + scAGN_hydroxylO_long + carbonylC_hydroxylO_long +
           aliph1HC_hydroxylO_vlong + aliph1HC_hydroxylO_long + hydroxylO_sulfur_vlong +
           carbonylC_bbC_medlong + bbN_bbN_medshort + aliph2HC_bbN_medlong +
           aliph2HC_scLysN_vlong + carboxylC_carboxylC_vlong + aliph1HC_bbN_medshort +
           aliph3HC_bbN_medlong + carbonylC_bbProN_vlong + carbonylC_carbonylO_vlong +
           hydroxylO_sulfur_long + carboxylO_sulfur_medshort + aliph1HC_scArgN_long +
           aliph1HC_bbCA_medshort + aliph1HC_carbonylO_medshort + carboxylO_bbO_vlong +
           bbCA_bbCA_vlong + sulfur_bbCA_short + sulfur_sulfur_vlong +
           bbCA_bbO_vshort + bbProN_bbCA_medshort + bbProN_bbN_long +
           aliph1HC_bbProN_vlong + carbonylC_scLysN_long + aliph3HC_bbN_long +
           aliph2HC_bbN_vlong + aliph3HC_hydroxylO_short + aliph3HC_bbO_short +
           carboxylO_carboxylO_long + aliph1HC_aliph3HC_short + bbProN_carboxylO_vlong +
           carboxylC_hydroxylO_short + aliph2HC_scLysN_long + aliph1HC_hydroxylO_medlong +
           aliph1HC_hydroxylO_medshort + aliph3HC_hydroxylO_medshort +
           aliph3HC_aliph3HC_vlong + aliph1HC_bbN_medlong + carbonylC_sulfur_short +
           scAGN_bbProN_medshort + aliph2HC_aromaticC_medshort + carboxylC_aliph2HC_long +
           carbonylC_carboxylC_long + carboxylO_sulfur_medlong + aliph3HC_aliph3HC_long +
           scAGN_scLysN_medlong + bbN_bbC_vshort, data = trainSet)
```

5.6 Code for ICM



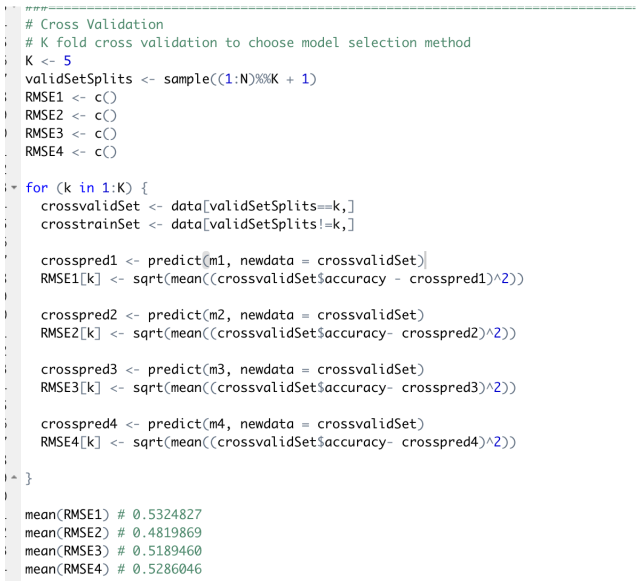
```r
#==============================================================================================
# Try ICM to search for a model with a potentially better BIC
# than the one found with stepwise
pen <- log(nrow(trainSet)) #
varlist = c()
varnames = names(trainSet)
n = nrow(trainSet)
varorder <- sample(1:ncol(trainSet)) # random order of variables
minCrit = Inf
noChange = F
while (!noChange) {
  noChange = T
  for (i in varorder) {
    if (i == 1)      next
    if (i %in% varlist & length(varlist) > 1) {
      index = c(1, varlist[varlist != i])
      trainVars = trainSet[, index]
      fit = lm(accuracy ~ ., data = trainVars)
      if (AIC(fit, k = pen) < minCrit) {
        minCrit = AIC(fit, k = pen)
        varlist = varlist[varlist != i]
        print(paste0("Criterion: ", round(minCrit, 1), ", variables: ", paste0(varnames[varlist], collapse = " ")))
        m2 = fit
        noChange = F
      }} else if (!i %in% varlist) {
      index = c(1, varlist, i)
      trainVars = trainSet[, index]
      fit = lm(accuracy ~ ., data = trainVars)
      if (AIC(fit, k = pen) < minCrit) {
        minCrit = AIC(fit, k = pen)
        varlist = c(varlist, i)
        print(paste0("Criterion: ", round(minCrit, 1), ", variables: ", paste0(varnames[varlist], collapse = " ")))
        m2 = fit
        noChange = F}}}}
```

After the above code, we have our model:



```
> m2$terms
accuracy ~ aliph2HC_bbN_long + aromaticC_scArgN_medshort + aliph1HC_bbProN_long +
    scLysN_carboxylO_long + scArgN_bbO_long + aliph2HC_bbN_medshort +
    aromaticC_scAGN_short + aliph2HC_aliph3HC_vlong + aromaticC_hydroxylO_long +
    aliph1HC_bbProN_vlong + aromaticC_bbO_vlong + aliph3HC_sulfur_long +
    bbN_bbC_vlong + aliph3HC_scLysN_vlong + bbCA_bbO_vshort +
    aliph3HC_scArgN_vlong + aromaticC_bbO_vshort + aliph3HC_aliph3HC_vlong +
    scAGN_bbN_medlong + carboxylC_aromaticC_long + aliph1HC_bbO_medlong +
    scArgN_bbC_short + carboxylO_bbO_vlong + sulfur_bbC_vlong +
    scArgN_bbN_medshort + aliph3HC_sulfur_medshort + scArgN_bbC_vlong +
    aliph2HC_bbN_vlong + carboxylC_carboxylO_medlong + scArgN_bbCA_long +
    aliph3HC_bbC_vlong + bbO_bbO_short + aliph3HC_sulfur_medlong +
    bbN_bbC_vshort + scArgN_bbO_medshort + aromaticC_bbC_short +
    aliph1HC_bbO_long + scAGN_bbProN_medshort + aliph1HC_aromaticC_medlong +
    aliph1HC_aromaticC_vlong + aromaticC_scLysN_vlong + aromaticC_hydroxylO_medlong +
    aliph1HC_bbProN_medlong + aliph1HC_sulfur_short + bbN_bbC_medshort +
    bbN_bbCA_medlong + scArgN_bbO_medlong + scAGN_carbonylO_medlong +
    sulfur_bbC_medlong + scArgN_bbC_medlong + scAGN_scLysN_medlong +
    scLysN_bbC_vlong + aromaticC_sulfur_medlong + aliph2HC_aromaticC_vlong +
    aliph2HC_bbN_medlong + carboxylO_bbN_vlong + sulfur_bbN_long +
    scArgN_bbC_long + aromaticC_scArgN_vlong + aliph1HC_aliph1HC_vlong +
    carbonylC_carbonylO_vlong + aliph1HC_scAGN_medshort + aliph3HC_aromaticC_short +
    scAGN_bbN_long + scArgN_bbCA_vlong + aliph1HC_aliph3HC_short +
    carbonylC_carboxylC_long + scArgN_bbO_vlong + bbN_bbCA_vlong +
    aliph1HC_bbO_medshort + scArgN_bbCA_medshort + carbonylC_hydroxylO_vlong +
    carbonylC_sulfur_short + bbCA_bbCA_medlong + sulfur_bbCA_short +
    bbO_bbO_vshort + bbN_bbO_medshort + aliph3HC_sulfur_vlong +
    aliph1HC_aromaticC_long + carboxylO_bbCA_medshort + bbProN_carbonylO_vlong +
    carbonylC_bbC_medlong + sulfur_bbO_short + bbCA_bbCA_vlong +
    aliph1HC_hydroxylO_medlong + hydroxylO_bbC_medlong + carbonylC_bbC_long +
    scArgN_bbCA_medlong + scLysN_hydroxylO_vlong + aliph3HC_aromaticC_medlong +
    aliph2HC_aliph3HC_short + bbN_bbO_long + scLysN_bbC_medlong +
    aliph1HC_hydroxylO_long + bbProN_bbCA_medshort + carbonylC_scLysN_vlong +
    scLysN_bbN_medshort + aliph1HC_hydroxylO_vlong + carbonylC_bbProN_vlong +
    bbC_bbO_short + aliph1HC_aromaticC_medshort + aliph1HC_aromaticC_short +
    aromaticC_hydroxylO_medshort + carbonylC_bbProN_long + carbonylC_bbProN_medlong +
    aliph2HC_scArgN_vlong + aliph1HC_aliph3HC_medshort + aliph1HC_bbN_long +
    aliph1HC_scAGN_medlong + aliph3HC_aromaticC_long + bbProN_bbC_short +
    bbN_bbC_short + aliph3HC_bbN_short + aromaticC_bbCA_vlong +
    aliph1HC_carbonylO_medshort + carboxylO_carboxylO_vlong
```

5.7 Code for Cross Validation k-fold



```
###==========================================================================
# Cross Validation
# K fold cross validation to choose model selection method
K <- 5
validSetSplits <- sample((1:N)%%K + 1)
RMSE1 <- c()
RMSE2 <- c()
RMSE3 <- c()
RMSE4 <- c()

for (k in 1:K) {
  crossvalidSet <- data[validSetSplits==k,]
  crosstrainSet <- data[validSetSplits!=k,]

  crosspred1 <- predict(m1, newdata = crossvalidSet)
  RMSE1[k] <- sqrt(mean((crossvalidSet$accuracy - crosspred1)^2))

  crosspred2 <- predict(m2, newdata = crossvalidSet)
  RMSE2[k] <- sqrt(mean((crossvalidSet$accuracy- crosspred2)^2))

  crosspred3 <- predict(m3, newdata = crossvalidSet)
  RMSE3[k] <- sqrt(mean((crossvalidSet$accuracy- crosspred3)^2))

  crosspred4 <- predict(m4, newdata = crossvalidSet)
  RMSE4[k] <- sqrt(mean((crossvalidSet$accuracy- crosspred4)^2))

}

mean(RMSE1) # 0.5324827
mean(RMSE2) # 0.4819869
mean(RMSE3) # 0.5189460
mean(RMSE4) # 0.5286046
```