Project 2 Report

Junheng Zhang

301306955

November 28, 2021

Library and settings

Library used:

library(ggplot2)

library(dplyr)

library(MASS)

library(glmnet)

library(pls)

library(mgcv)

library(kableExtra)

library(nnet)

library(rpart)

library(rpart.plot)

library(randomForest)

library(gbm)

Setting before fitting model:

Random seed = 1 (for better reproduction)

Data cleaning

The data cleaning is to clean out the data records that should not be used in the analysis.

Check NA: no NA exist in the given data file

Check outliers: There are some outliers for some of the predictive variables. However, by given situation where we do not have any background information about the dataset, we cannot certainly indicate why the outliers exist and we do not have any good reason to exclude the outliers from dataset. I decided to keep all data as original in this analysis.

Model fitting

This step is to check every model and the fitting result visually to see how each model works with the given dataset. The models I first attempted to fit and the code (marked in blue) are the following:

- Model 1: Least square regression
  mod.fit1 = lm(Y~., data = df)

- Model 2: Stepwise regression (bidirectional)
  mod.fit2.start = lm(Y~., data = df)
  mod.fit2 <- stepAIC(mod.fit2.start, direction = "both", trace = FALSE)

- Model 3: Ridge regression

```
lambda.vals = seq(from = 0, to = 100, by = 0.05)
mod.fit3 = lm.ridge(Y~., lambda = lambda.vals,data = df)
```

- Model 4: LASSO regression with λ_{min}

```
data.mat.raw = model.matrix(Y ~ ., data = df)
data.mat = data.mat.raw[,-1]
mod.fit4_5 = cv.glmnet(data.mat, df$Y)
lambda.min = mod.fit4_5$lambda.min
```

- Model 5: LASSO regression with λ_{1se}

```
mod.fit4_5 = cv.glmnet(data.mat, df$Y)
lambda.1se = mod.fit4_5$lambda.1se
```

- Model 6: Partial Least Squares

```
mod.fit6 = plsr(Y ~ ., data = df, validation = "CV", segments = 10)
```

- Model 7: Generalized additive models

```
mod.fit7 = gam(Y ~ s(X1) + s(X2) + s(X3) + s(X4) + s(X5) + s(X6) + s(X7) + s(X8) + s(X9) + s(X10) +
s(X11) + s(X12) + s(X13) + s(X14) + s(X15), data = df)
```

- Model 8: Neural net

```
#rescale is a user define function copied from lecture code
X.raw = dplyr::select(df, -Y)
X.nnet = rescale(X.raw, X.raw)
mod.fit8 = nnet(x=X.nnet,y=df$Y,size = 2,trace = FALSE)
```

- Model 9: Full tree model

```
mod.fit9 = rpart(Y ~ ., data = df, cp = 0)
```

- Model 10: Minimum CV error tree

```
#CP.best is the tunning parameter found by the min cv error in full tree
mod.fit10 = prune(mod.fit9, cp = CP.best)
```

- Model 11: 1 SE rule CV tree

```
#CP.1se is the tunning parameter found by 1se rule
mod.fit11 = prune(mod.fit9, cp = CP.1se)
```

- Model 12: Random Forest

```
mod.fit12 = randomForest(Y ~ ., data = df, importance = T)
```

- Model 13: Boosting

```
mod.fit13 <- gbm(data=df, Y~., distribution="gaussian")
```

The implementation detail and finding for each model are the following:

Least square regression: in the summary, it shows many of the predictive variables have large p-values, which means many of the variables are not good factors and should not be include in the model.

Stepwise regression: the variables selected by bidirectional stepwise regression has relatively smaller p-values, they are: X1, X2, X6, X9, X11

Ridge regression: the λ_{min} chose by the ridge regression is 100, and the X1 has a very large coefficient while the others are small

LASSO regression with λ_{min}: the λ_{min} chose by the LASSO regression is 0.108, and it shows similar coefficient with ridge regression where X1 has a very large coefficient while the others are small.

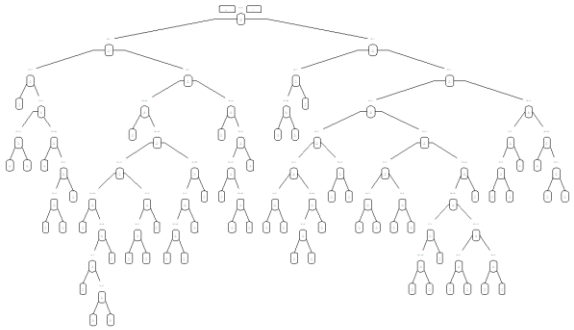
Model 5: LASSO regression with λ_{1se}: the λ_{1se} chose by the LASSO regression is 0.208, the coefficients are similar to LASSO regression with λ_{min}

Partial Least Squares: the optimal number of folds chose by partial least square with cross validation is 3

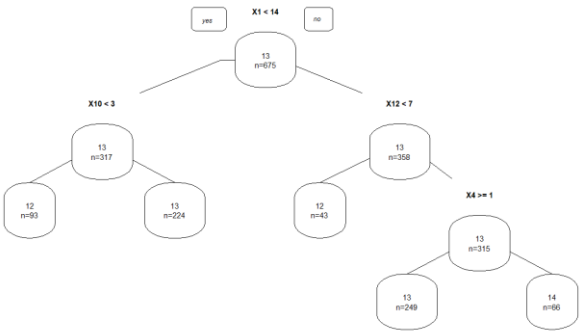
Generalized additive models: adjusted R-square is 0.201, and Deviance explained = 24.9%

Neural net: fitted with default settings and size =2, the optimal tunning parameters will be calculated in the model evaluation part later.

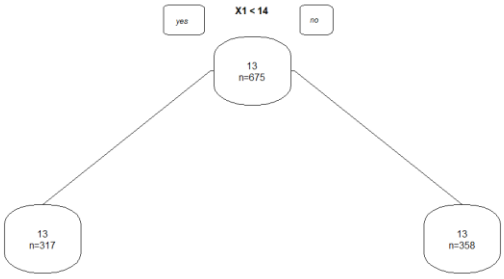
Full tree model: The tree contains many leaves, and it may be overfitted to the data



Minimum CV error tree: the minimum cp is calculated to be 0.0167, and the tree can be visualized as follow:



1 SE rule CV tree: the minimum cp is calculated to be 0.042, and the tree can be visualized as follow:



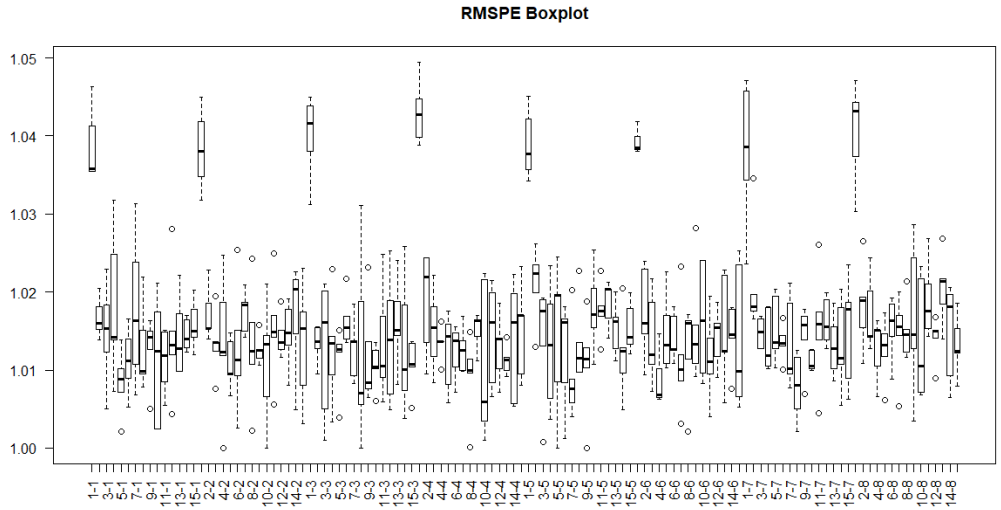
Random Forest: start with default setting for all tuning parameters, perform 5 times replicate process for finding the tuning parameter, detail of tuning parameter will be explained later

Boosting: start with default setting for all tuning parameters, perform 2 reps of 5-fold CV to find best tunning parameter, detail of tuning parameter will be explained later

Tuning parameter

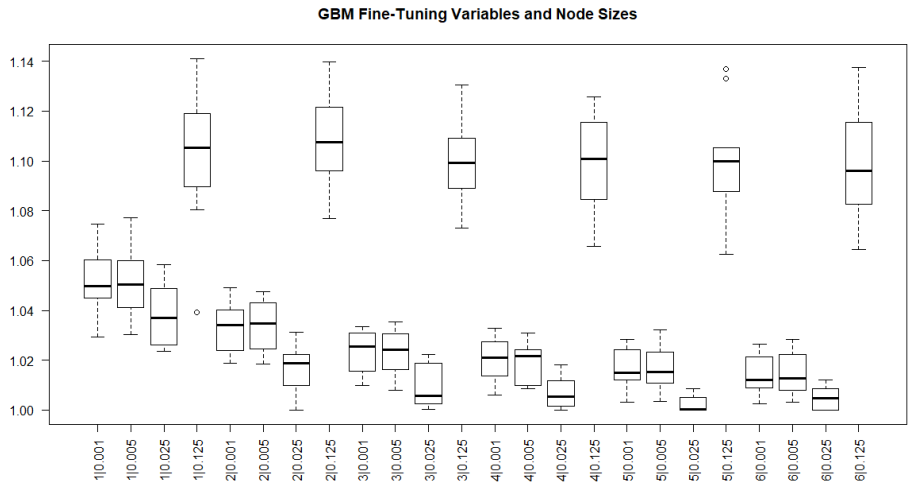
There are 3 models can be improved by using different tuning parameters:

- Neural net: starting with setting the range of nodes = c(1, 3, 5, 7, 9), shrinkage = c(0.001, 0.1, 0.5, 1, 2). Perform 5-fold CV with 10 times re-fit for each combination of tuning parameter within the model evaluation part, that is perform inner CV within the large CV. After the inner CV for nnet model, find the best tuning MSPEs by minimizing average and choose the best tuning parameter automatically.
- Random Forest: starting with setting the range of mtry = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15), nodesize = (1, 2, 3, 4, 5, 6, 7, 8), and I used 5 times replicate process for calculating OOB errors based on bootstrapping. The resulting RMSPE plot is following:



Based on the RMSPE boxplot, the model with mtry=7 and nodesize=5 looks best to me, so I set the mtry=7, nodesize=5 for my best random forest model, as well as in the model evaluation later.

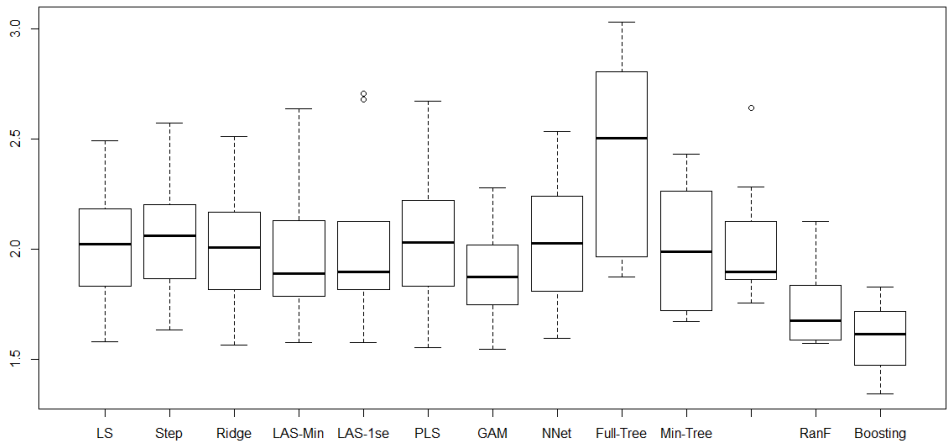
- Boosting: starting with setting the range of shr = (.001,.005,.025,.125), dep = (1, 2, 3, 4, 5, 6), and I used 2 reps of 5-fold CV to find best tunning parameter. The resulting RMSPE plot is following:



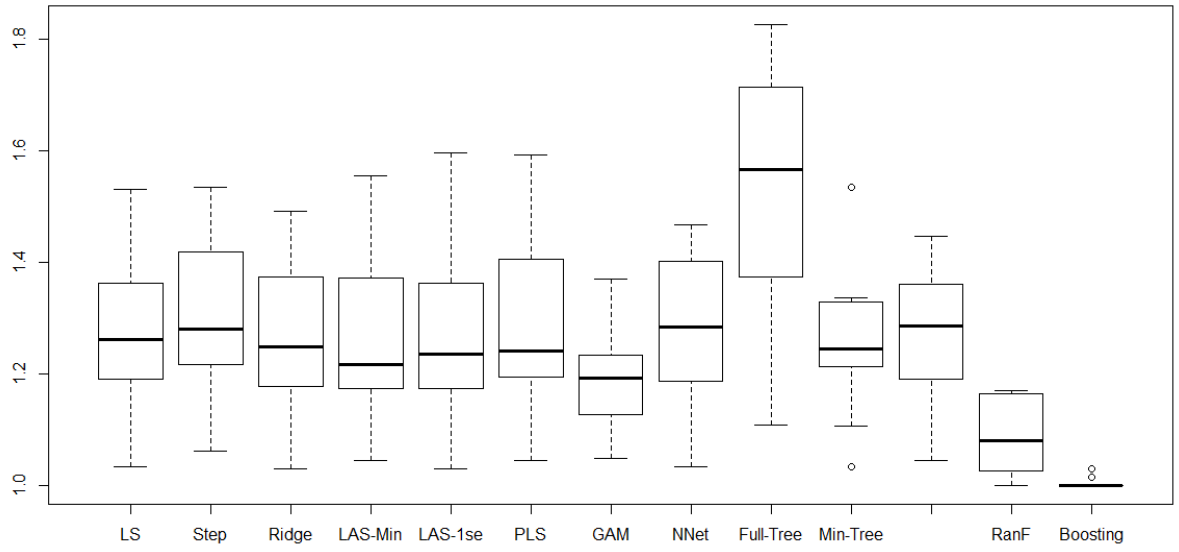
Based on the RMSPE boxplot, the model with shr = 0.025 and dep = 5 looks best to me, so I set the interaction.depth=5, shrinkage=0.025 for my best boosting model, as well as in the model evaluation later.

Model evaluation

The evaluation method used for all models is 10-fold cross validation, fit all models to each resample. All the min, 1se and tunning parameter for models are set as same as above. The tuning parameter for nnet model is automatically calculated and determined in the inner loop. The resulting MSPE boxplot is following:



And the RMSPE plot is following:



The full MSPE are the following:

```
      LS      Step     Ridge   LAS-Min   LAS-1se       PLS       GAM
2.042891  2.091839  2.030508  2.023017  2.040493  2.065157  1.882738
    NNet Full-Tree  Min-Tree  1SE-Tree      RanF  Boosting
2.048430  2.441928  2.002384  2.019718  1.733795  1.601871
```

Based on the RMSPE plot and full MSPE values, the Boosting model with tunning parameter interaction.depth=5, shrinkage=0.025 is the best model to me. Thus I choose Boosting model as my best model.

Prediction

The code of generating the prediction result is the following (marked in blue):

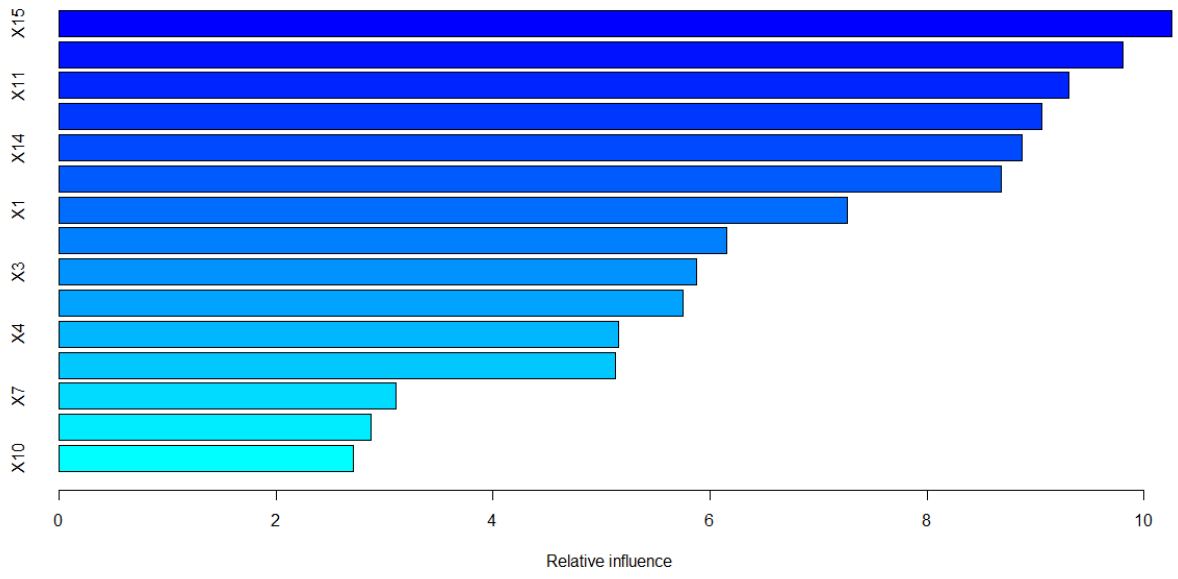
```
#please set working dirctory before reading the data

df.train <- read.csv("Data2021_final.csv")

df.predict <- read.csv("Data2021test_final_noY.csv")

set.seed(1)


#fit best model

mod.best <- gbm(data=df.train, Y~., distribution="gaussian",

          n.trees=10000, interaction.depth=5, shrinkage=0.025)
```

```
#make prediction

n.tree = gbm.perf(mod.best, plot.it = F)*2

predictions = predict(mod.best, df.predict, n.tree)


#output result

write.table(predictions, "test.csv", sep = ",", row.names = F, col.names =F)
```

After running the code above, you should get the file named test.csv like following:

1	12.16776
2	12.87984
3	13.39832
4	14.42036
5	12.31898
6	12.89988
7	13.22823
8	12.23495
9	14.1389
10	12.80421
11	13.14165
12	12.63368
13	12.94118
14	13.74934
15	13.04761

Importance of variables



```
> summary(mod.best)
    var   rel.inf
X15 X15 10.257022
X8   X8  9.807440
X11 X11  9.303626
X13 X13  9.061613
X14 X14  8.874111
X5   X5  8.679031
X1   X1  7.267443
X6   X6  6.157283
X3   X3  5.874675
X2   X2  5.748729
X4   X4  5.154139
X12 X12  5.128232
X7   X7  3.104565
X9   X9  2.871631
X10 X10  2.710460
```

As we can see for the best model, the top 3 important variables are X15, X8, X11, and the X15 is determined to be the most important variable.