

Final Project

Pittsburgh Campus
Jingyi Guo (jingyig1)

Model Description

1. Considered predictors:

Ret_2 ~ Ret_120:

Obtained by excluding rows containing NA data in columns Ret_2 ~ Ret_120 in the trainset, stored in RetsOnly.

My models only consider Ret_2 ~ Ret_120 because the final test set only provides these data for prediction.

V1 ~ V42 :

Obtained in the following three steps:

Step 1: Conduct lasso and select 42 variables out of Ret_2 ~ Ret_120.

Step 2: Scale the 42 new variables to have mean 0 and variance 1.

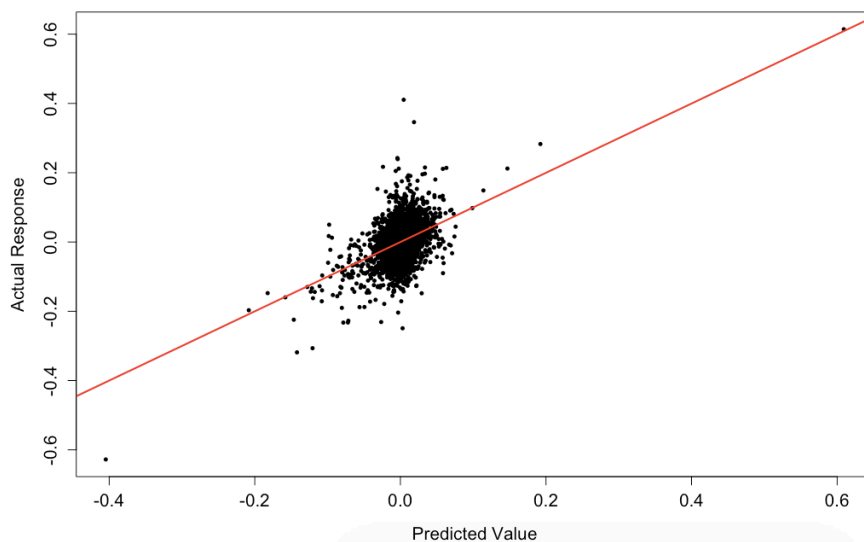
Step 3: Use one-dimensional nonparametric regression model to smooth the remaining 42 variables, stored in nonpara_fitted.

2. Three Models as Required

Model 1: Fully Linear Model

Model: $Ret_PlusOne = \beta_0 + \beta_1 Ret_2 + \dots + \beta_{119} Ret_120 + \epsilon$

Plot of Predictions vs. Response:



We can see that the fit is terrible.

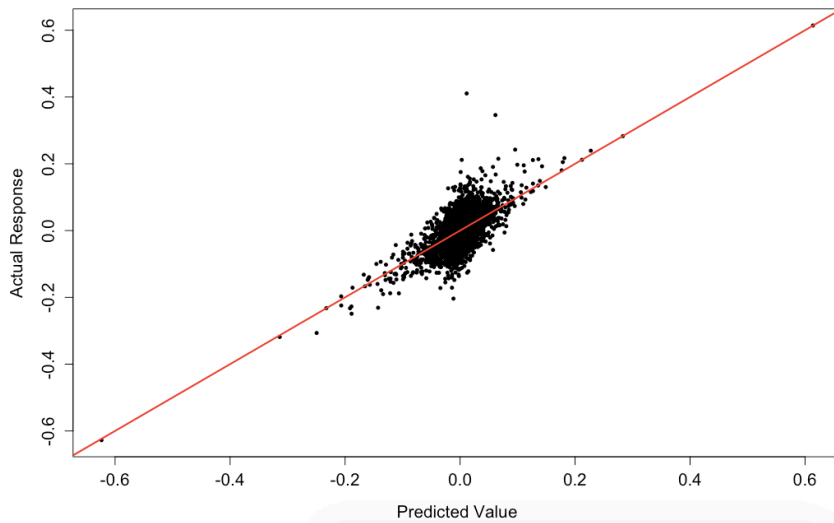
Model 2: Additive Nonparametric Model

Model: $Ret_PlusOne = f(Ret_2, \dots, Ret_120) + \epsilon$

I use General Additive Model, and use smoothing splines for the individual nonparametric fits

here.

Plot of Predictions vs. Response:



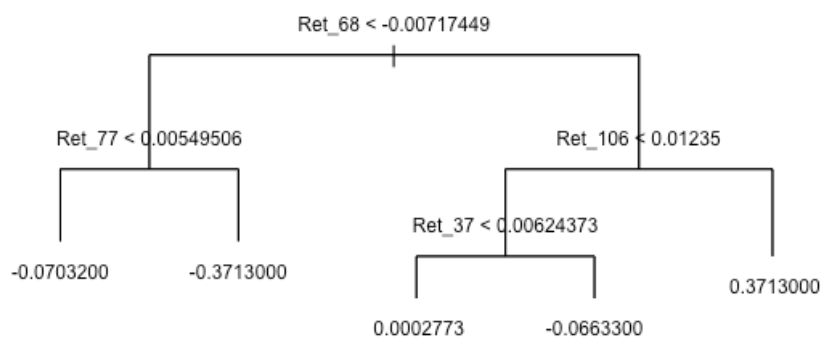
We can see that this model fits better than the first one.

Model 3: Tree Based Model

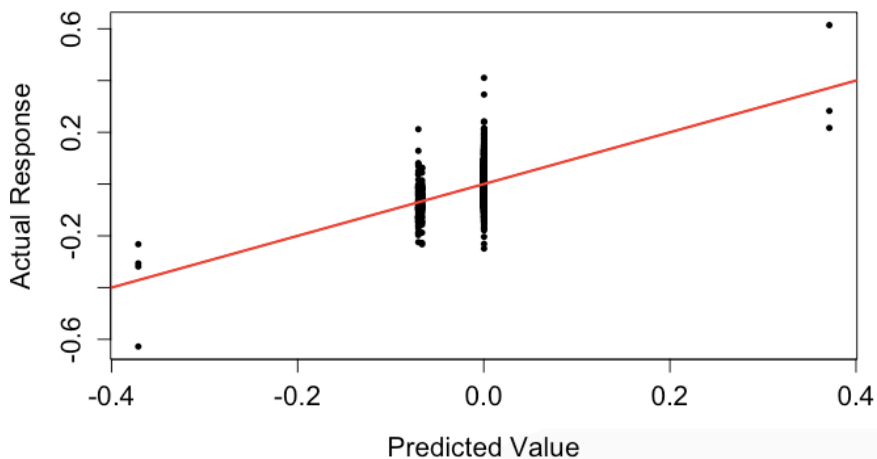
Model: Regression Tree.

This model take Ret_2 ~ Ret_120 predictors into consideration.

Plot of optimal tree:



Plot of Predictions vs. Response:



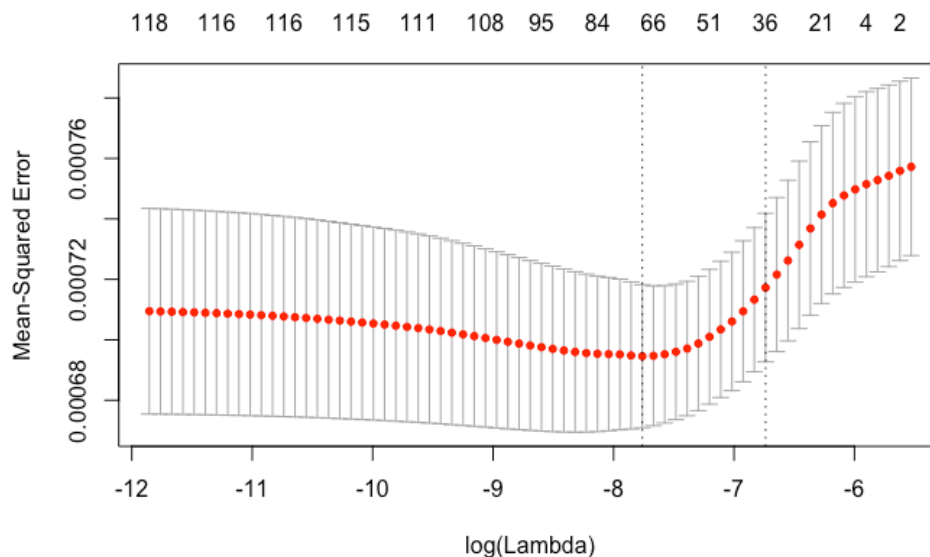
There are too few branches in the optimal tree, and the model fits poorly.

3. Developing My Prediction Model

First select predictors to avoid overfitting.

Variable Selection

In order to prevent multicollinearity and reduce overestimating, I decide to use lasso to reduce variables.



The conclusion is that the optimal model has 42 nonzero $\hat{\beta}_j$. After variable selection, the predictors include:

Ret_4, Ret_9, Ret_10, Ret_22, Ret_23, Ret_27, Ret_29, Ret_34, Ret_36, Ret_37, Ret_42, Ret_44, Ret_49, Ret_50, Ret_53, Ret_57, Ret_62, Ret_64, Ret_66, Ret_68, Ret_72, Ret_74, Ret_75, Ret_80, Ret_85, Ret_86, Ret_89, Ret_90, Ret_100, Ret_101, Ret_102, Ret_103, Ret_104, Ret_105, Ret_107, Ret_109, Ret_113, Ret_115, Ret_116, Ret_117, Ret_119, Ret_120

Our analysis and prediction would base on those variables.

Smoothing the Return Series

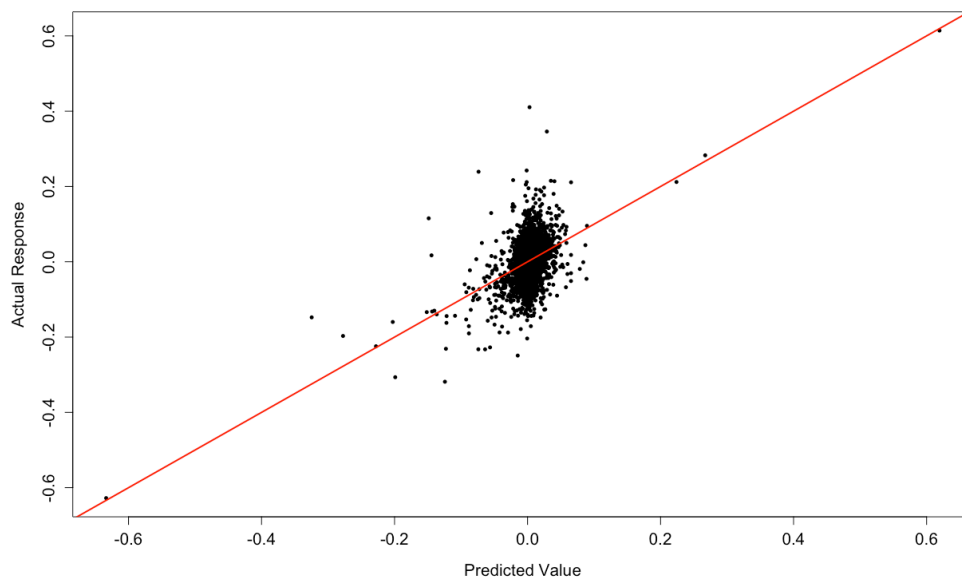
Uses one-dimensional nonparametric regression model to smooth the remaining 42 return series and reduce “noise”.

By plotting the histograms of those smoothed variables, we can see that most of the variables are skewed, so we might consider a log transformation.

Model 4: Robust Regression

First, consider using the robust linear regression model to fit since the score on leaderboard is ranked with regard to weighted mean absolute error.

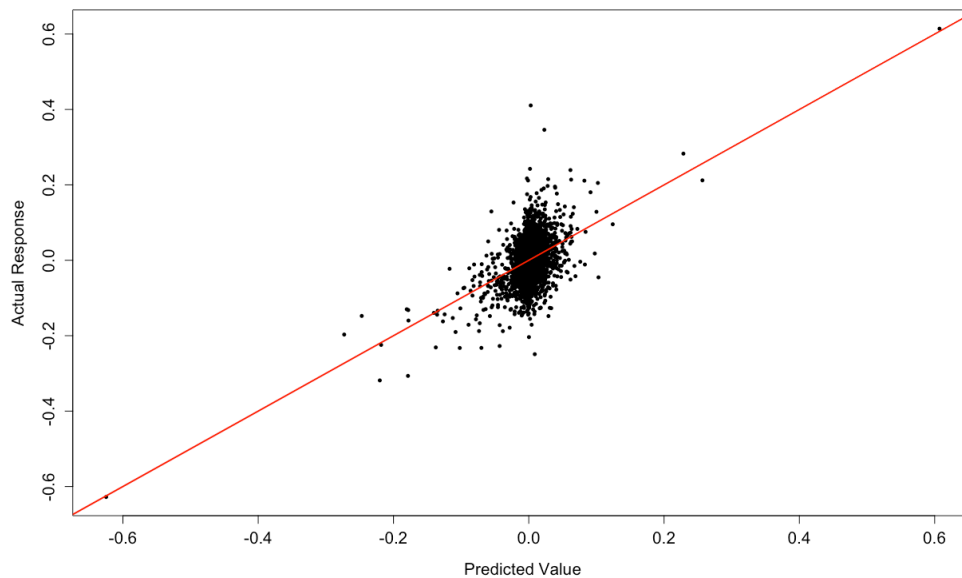
Plot of Predictions vs. Response:



From the plot above, we can see that most of the points do not align with the 45 degree line, indicating that the robust linear regression fits poorly.

Model 5: Projection Pursuit

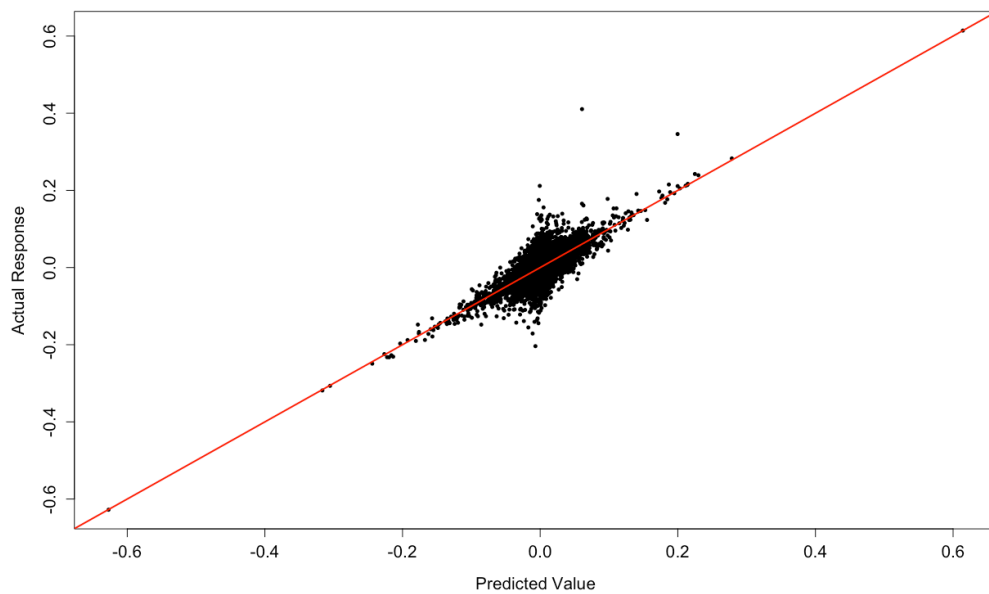
Since projection pursuit has the best performance in the last homework, I decide to try projection pursuit first.



From the plot above, we can see that the projection pursuit model still lack of fit.

Model 6: Neural Network

Since neural network has the best fitting performance in the last homework, I decide to try neural network first.

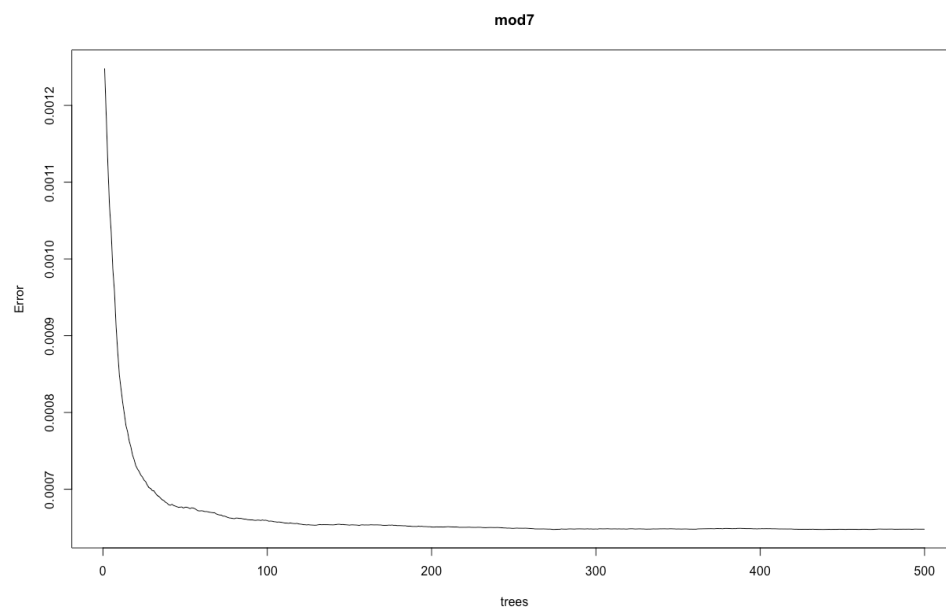


In the predicted value vs. response plot, the points scatter closer to the 45 degree line, indicating that this might be a better fit compared to the previous models, but further improvements are needed.

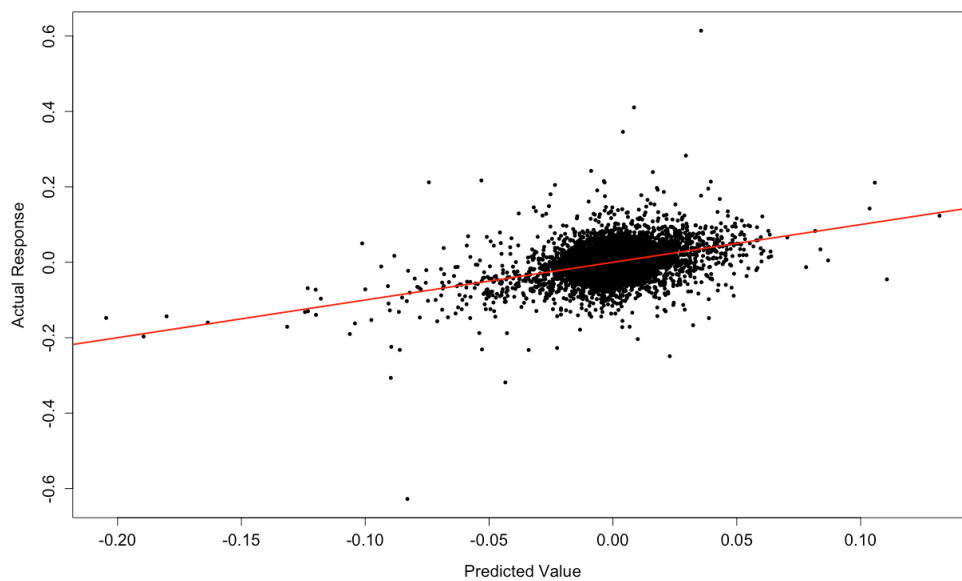
Model 7: Random Forest [Final Model]

First, use `tuneRF()` to search for the optimal choice of `mtry`. The output shows that the OOB error is minimized when `mtry=14`.

Then, try fitting the random forest model with the `mtry=14` and other parameters being the default value. Plot the model (as shown below) and We found that the OOB error has leveled off by $B \approx 50$, so using the default 500 trees was likely excessive.



At last, try again fitting the random forest model with `mtry=14` and `B=ntree=50`.

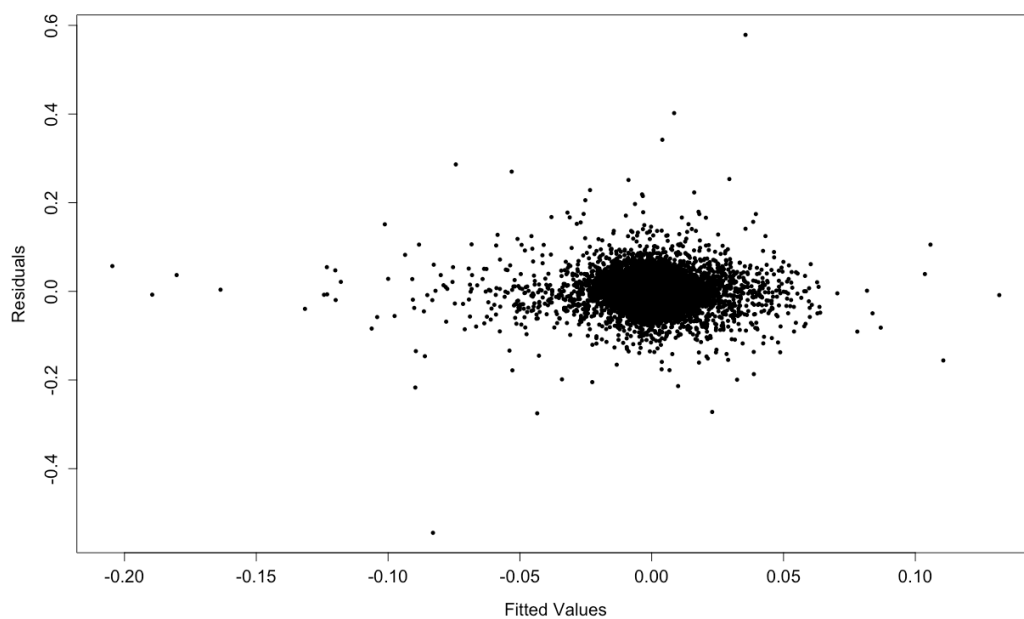


In the predicted value vs. response plot above, the points scatter around to the 45 degree line, indicating that this might be a better fit compared to the previous models.

So far Model 7 seems to be the best-fitting model. So we draw diagnostic plots the Model 7 to check its quality of fit.

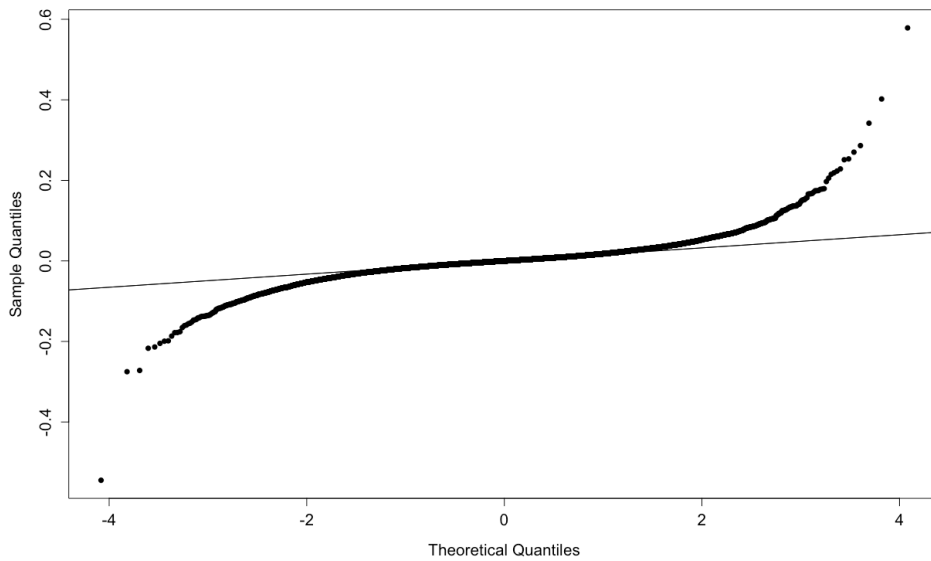
Model Diagnostic

1. Plot of Residuals versus Fitted Values



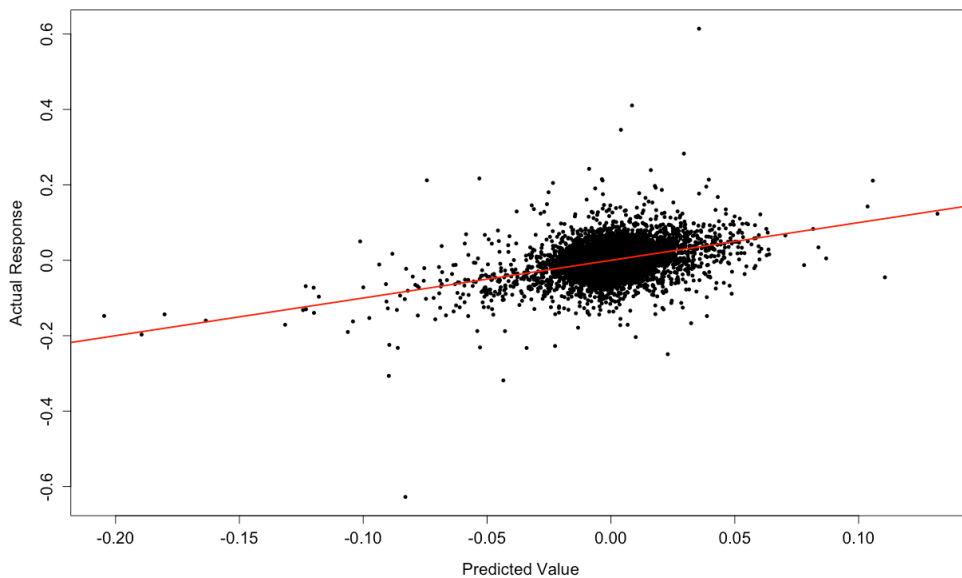
There is no prevalent pattern in the plot of residuals versus fitted values. Model 7 seems to be an acceptable fit.

2. Normal Probability Plot



Normal probability plot suggests that errors have distribution with tails heavier than normal distribution. However, since random forest model do not require the normal assumption, this is acceptable.

3. Plot of Predicted Value versus Response



The points scatter around the 45 degree line rather evenly, and they show a trend along the 45 degree line. This indicates that this model is a good fit.

* Cook's Distance is not applicable to random forest model, and residual vs. time plot doesn't make sense here, so I omitted those two plots.

4. Overfitting


In the plot of predicted value versus response, there is enough scatter around the 45 degree line and not all the points on strictly on the line. So there is no noticeable sign of overfitting. In the modeling process, I took several measures to avoid overfitting.

First, I conduct lasso to select variables and reduce dimension in the first step. By reducing predictors and simplifying the model, I reduce overestimating the absolute coefficients and overfitting the model to some extent. I also used cross validation when applying lasso.

Second, I smooth the predictors with nonparametric regression and reduce their “noise”, which might contribute to overfitting.

Third, when selecting the smoothing parameters, I search for the optimal choice of smoothing parameter $mtry$ and simplify the random forest model by reducing B from 500 to 50 according to the model plot.

Test Set Performance



The Winton Stock Market Challenge

Join a multi-disciplinary team of research scientists
\$50,000 · 832 teams · a year ago

[Overview](#) [Data](#) [Discussion](#) [Leaderboard](#) [More](#) [My Submissions](#) [Submit Predictions](#)

5 submissions for [JingyiGuo](#)

You can select up to 2 submissions to be used to calculate your final leaderboard score. If 2 submissions are not selected, they will be chosen based on your best submission scores on the public leaderboard.

Your final score will not be based on the same exact subset of data as the public leaderboard, but rather a different private data subset of your full submission —your public score is only a rough indication of what your final score is.

You should thus choose submissions that will most likely be best overall, and not necessarily on the public subset.

| Submission and Description | Private Score | Public Score | Use for Final Score |
|--------------------------------------------------------------------------------------------------------------------------|---------------|--------------|--------------------------|
| mysubmission.txt 3 minutes ago by JingyiGuo add submission details | 1730.54800 | 1772.58231 | <input type="checkbox"/> |

R Code

```
#library
library(mgcv)
library(tree)
source("http://www.stat.cmu.edu/~cschafer/MSCF/cv.tree.full.txt")
library(glmnet)
library(fANCOVA)
library(MASS)
library(nnet)
library(car)
source("http://www.stat.cmu.edu/~cschafer/MSCF/CVforppr.R")
source("http://www.stat.cmu.edu/~cschafer/MSCF/CVfornnet.R")
library(randomForest)

#data
trainset = read.table("train.csv", sep=",", header=T)
fullrow = rep(FALSE,nrow(trainset))
for(i in 1:nrow(trainset))
{
  fullrow[i] = !any(is.na(trainset[i,29:147]))
}
RetsOnly = trainset[fullrow, 29:147]
varnames = c(paste("Ret_", 2:120, sep=""))
fulltrainset=trainset[fullrow,]
Ret_PlusOne=fulltrainset$Ret_PlusOne

# First 3 models
# Mode 1 - fully linear model containing only the parametric predictors
fullform = as.formula(paste("Ret_PlusOne ~ ",paste(varnames,collapse="+")))
mod1 = lm(fullform, data=trainset, subset=fullrow)
summary(mod1)
AIC1=AIC(mod1)
plot(predict(mod1),Ret_PlusOne,pch=16,cex=0.7,xlab="Predicted Value",ylab="Actual
Response", cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=2)

# Model 2 - additive nonparametric model
# use gam model here
gamfullform = as.formula(paste("Ret_PlusOne ~
",paste(paste("s(",varnames,")"),collapse="+")))
mod2 = gam(gamfullform, data=RetsOnly)
```

```

summary(mod2)
AIC2=AIC(mod2)
plot(predict(mod2),Ret_PlusOne,pch=16,cex=0.7,xlab="Predicted Value",
      ylab="Actual Response", cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=2)

# Model 3 - tree-based model
# use regression tree model here
mod3=tree(fullform,data=RetsOnly,mindev=0.01,minsize=2)
cvout=cv.tree.full(mod3,mindev=0.002)
plot(cvout)
optalpha=cvout$k[which.min(cvout$dev)]
opttree=prune.tree(mod3,k=optalpha)
plot(opttree)
text(opttree,cex=0.75)
summary(opttree)
plot(predict(opttree),Ret_PlusOne,pch=16,cex=0.7,xlab="Predicted Value",
      ylab="Actual Response", cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=2)

# Develop my best model
# Step 1 Variable Select
xmatrix=as.matrix(RetsOnly)
glmnetout=glmnet(xmatrix,Ret_PlusOne)
# cross validation to find lambda
cvglmout=cv.glmnet(xmatrix,Ret_PlusOne)
plot(cvglmout)
optlambda=cvglmout$lambda.1se
optilambdapos=which(cvglmout$glmnet.fit$lambda==optlambda)
glmnetout$beta[glmnetout$beta[,optilambdapos]!=0,optilambdapos]
# selected predictor data matrix
selectedx=as.data.frame(cbind(RetsOnly$Ret_4,RetsOnly$Ret_9,RetsOnly$Ret_10,RetsOnly$Ret_22,RetsOnly$Ret_23,RetsOnly$Ret_27,RetsOnly$Ret_29,RetsOnly$Ret_34,RetsOnly$Ret_36,RetsOnly$Ret_37,RetsOnly$Ret_42,RetsOnly$Ret_44,RetsOnly$Ret_49,RetsOnly$Ret_50,RetsOnly$Ret_53,RetsOnly$Ret_57,RetsOnly$Ret_62,RetsOnly$Ret_64,RetsOnly$Ret_66,RetsOnly$Ret_68,RetsOnly$Ret_72,RetsOnly$Ret_74,RetsOnly$Ret_75,RetsOnly$Ret_80,RetsOnly$Ret_85,RetsOnly$Ret_86,RetsOnly$Ret_89,RetsOnly$Ret_90,RetsOnly$Ret_100,RetsOnly$Ret_101,RetsOnly$Ret_102,RetsOnly$Ret_103,RetsOnly$Ret_104,RetsOnly$Ret_105,RetsOnly$Ret_107,RetsOnly$Ret_109,RetsOnly$Ret_113,RetsOnly$Ret_115,RetsOnly$Ret_116,RetsOnly$Ret_117,RetsOnly$Ret_119,RetsOnly$Ret_120))
# scale the data
selectedx=as.data.frame(scale(selectedx))

```

```

# Step 2 Smoothing the Variables
nonpara_fitted = matrix(nrow = 22390, ncol = 42)
newvarnames= c(paste("V", 1:42, sep=""))
for(i in 1:42)
{
  nonpara_fitted[,i] = loess(Ret_PlusOne ~ get(newvarnames[i]), data = selectedx,
span = 0.5, degree = 1,parametric = FALSE,family="symmetric")$fitted
}
nonpara_fitted=as.data.frame(nonpara_fitted)
newform=as.formula(paste("Ret_PlusOne ~ ",paste(newvarnames,collapse = "+"))))

# Step 3 Trying models
# Model 4 - Robust Regression
mod4 = rlm(newform,data=nonpara_fitted)
plot(predict(mod4),Ret_PlusOne,pch=16,cex=0.7,xlab="Predicted Value",
      ylab="Actual Response", cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=2)

# Model 5 - Projection Pursuit
pprCV = matrix(0,nrow=10,ncol=4)
for(j in 1:ncol(pprCV))
{
  set.seed(j)
  for(i in 1:nrow(pprCV))
  {
    pprCV[i,j] = CVforppr(newform, nterms=i, numfolds=10, data=nonpara_fitted,
                          sm.method="gcv spline")
  }
}
# decide to choose M=nterms=2
mod5 = ppr(newform,nterms=2,data=nonpara_fitted)
summary(mod5)
plot(predict(mod5),Ret_PlusOne,pch=16,cex=0.7,xlab="Predicted Value",
      ylab="Actual Response", cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=2)

# Model 6 - Neural Network
nnetfullform = as.formula(paste("Ret_PlusOne ~
",paste(paste("scale(",newvarnames,")"),collapse="+"))))
nnetCV = array(0,dim=c(5,5,4))
decaylist = c(0,0.0001,0.001, 0.01, 0.02)

```

```

Mlist = c(5,10,15,25,50)
for(k in 1:dim(nnetCV)[[3]])
{
  set.seed(k)
  for(i in 1:length(decaylist))
  {
    for(j in 1:length(Mlist))
    {
      nnetCV[i,j,k] = CVformnnet(nnetfullform, size=Mlist[j],
                                decay=decaylist[i], numfolds=10, data=selectedx)
    }
  }
}
# decide to choose M=size=10, lambda=decay=0.0001
mod6 =
nnet(nnetfullform,data=nonpara_fitted,size=25,linout=TRUE,decay=0.001,maxit=000,M
axNWts=8000)
plot(predict(mod6),Ret_PlusOne,pch=16,cex=0.7,xlab="Predicted Value",
      ylab="Actual Response", cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=2)

# Model 7 - Random Forest
tuneRF(selectedx,Ret_PlusOne)
# decide to choose mtry=14
# mod7 = randomForest(newform,data=selectedx,ntree=500,nodesize=5,mtry=14)
# after plotting mod7 above, I found that the OOB error leveled off by B=50, so
# using 500 trees was likely excessive
mod7 = randomForest(newform,data=selectedx,ntree=50,nodesize=5,mtry=14)
plot(predict(mod7),Ret_PlusOne,pch=16,cex=0.7,xlab="Predicted Value",
      ylab="Actual Response", cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=2)

# Diagnostic Plots
# plot of residuals vs. fitted values
residsmod7 = Ret_PlusOne - predict(mod7)
plot(predict(mod7), residsmod7, xlab="Fitted Values", ylab="Residuals",
      cex.axis=1.3, cex.lab=1.3, pch=16, cex=0.7)

# normal probability plot
qqnorm(as.numeric(residsmod7),cex.axis=1.3,cex.lab=1.3,pch=16,main="")
qqline(as.numeric(residsmod7))

```

```

# Prediction using test set
testsetsub=read.table("testsetsub.csv", sep=",", header=T)
# conduct the same data processing on testset
# select the 42 variabls out of testset
test_selectedx=as.data.frame(cbind(testsetsub$Ret_4,testsetsub$Ret_9,testsetsub$R
et_10,testsetsub$Ret_22,testsetsub$Ret_23,testsetsub$Ret_27,testsetsub$Ret_29,tes
tsetsub$Ret_34,testsetsub$Ret_36,testsetsub$Ret_37,testsetsub$Ret_42,testsetsub$R
et_44,testsetsub$Ret_49,testsetsub$Ret_50,testsetsub$Ret_53,testsetsub$Ret_57,tes
tsetsub$Ret_62,testsetsub$Ret_64,testsetsub$Ret_66,testsetsub$Ret_68,testsetsub$R
et_72,testsetsub$Ret_74,testsetsub$Ret_75,testsetsub$Ret_80,testsetsub$Ret_85,tes
tsetsub$Ret_86,testsetsub$Ret_89,testsetsub$Ret_90,testsetsub$Ret_100,testsetsub$
Ret_101,testsetsub$Ret_102,testsetsub$Ret_103,testsetsub$Ret_104,testsetsub$Ret_1
05,testsetsub$Ret_107,testsetsub$Ret_109,testsetsub$Ret_113,testsetsub$Ret_115,te
stsetsub$Ret_116,testsetsub$Ret_117,testsetsub$Ret_119,testsetsub$Ret_120))
test_selectedx=as.data.frame(scale(test_selectedx))

# prediction
mypreds=predict(mod7,newdata=test_selectedx)
ConvertSubmission(mypreds)

# save file
write(mypreds, file = "jingyig1.txt", sep = "\n")

```