Final Project

Abhishek Bhatnagar

Data –

In order to prevent over-fitting I gathered more data from the S&P 500 list and cleaned in order to use the results appropriately.

Predictors –

1. Prevvol1 - 1 month previous monthly volatility
2. Prevvol2 - 2 months previous monthly volatility
3. Hilo – Previous months intraday movements measured by mean(intraday high – intraday low/intraday adjusted stock Price)
4. Above_avg – A factor form predictor which is 1 if the stock has a volatility greater than the entire index and 0 otherwise
5. Logvol – logarithm of the mean of the volumes of stocks traded in the past month
6. Logmarcap – logarithm of the market capitalization of the stock
7. Logshortratio – logarithm of the short ratio for the stock which is measured by the volumes shorted divided by the volumes traded and is reported as a percentage by yahoo
8. Change from 50 day moving average – measures the change from the 50 day momentum of the stock
9. Change from 200 day moving average – measures the change from the 200 day momentum of the stock
10. Earnings per share as reported one month ago

See the appendix for the set up for the predictors and response and the full listing of the r code to produce all the models

In order to determine the volatility, I used 2 time delayed previous months volatilities (one month and 2 months previous volatilities)

I also used some technical indicators such as the change from 50 day moving average and change from 200 day moving average to capture the momentum of the stock and how it factors into the volatility. I also employed the short-ratio which measures the number of market participants who are willing to short the stock at a given time. Similarly I employed the intra-day movements mean for the stock in the previous month.

I also used fundamental indicators such as EPS and Market Cap

In order to avoid forecasting the past I made sure I was taking the predictors which were more than 30 days before and modelling the realized volatility of the past 30 days

Summary of Findings

General Additive Model has provided me the best model for fitting the volatilities for the next month.

Below are the findings of the various models I employed

Models	Degrees of Freedom	Deviance	AIC
Robust Model	5	0.8027	-1768
Linear Model	11	0.7843	-1767
Exhaustive Model	5	0.7903	-1776
GAM	36.31	0.4282	-2015
PPR	63.91	0.4387	
Optimal Tree	18	0.407	
Pruned Tree	81	0.129	

Based on the findings I have decided to choose GAM as the best model.

I could have used the Optimal Tree as the best model because of the numerical results, but since the response is not a factor type I felt that GAM was more suited for the job

In the final GAM model the predictors and response are –
gammod3 = gam(transresp~s(prevvol1)+s(prevvol2)+hilo+s(ch200ma)+s(ch50ma)+s(logshortratio))

Factors which avoid overfitting –

- Gathering more data – I took the list from S&P 500 which allowed me to get a lot more data which allows better fitting and lesser risk of overfitting
- With around 500 data points I have chosen a model with degrees of freedom are 36.31
- Also the AIC shows that the increase in fit is more than compensates the increase in complexity
- In the GAM model, I removed the factors which weren't significant in order to lower the degrees of freedom while minimally impacting the fit
- After plotting the GAM model I realized that the hilo factor was being modelled mostly as a linear predictor and therefore I removed the splines from there to lower the degrees of freedom further

R-squared adjusted = 0.692

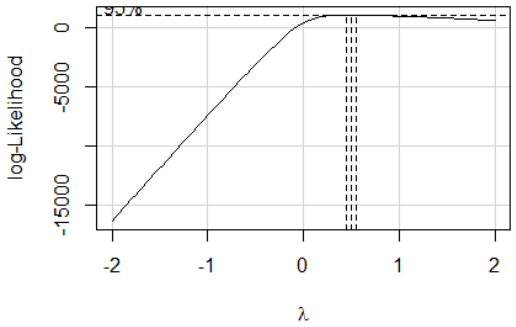
Diagnostic Plots for the Final Model

Please see them in the GAM section of the report

Linear Models

In order to understand how the response should be transformed I used the boxCox transformation and found the optimal transformation point to be Lambda = 0.5

BoxCox Transformation




```
lm(formula = transresp ~ prevvol1 + prevvol2 + hilo + logvol +
    factor(above_avg) + ch200ma + ch50ma + eps + logmarcap +
    logshortratio)

Residuals:
     Min       1Q   Median       3Q      Max
-0.13024 -0.02382 -0.00508  0.02039  0.27664

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       -1.920818   0.025756  -74.58  < 2e-16 ***
prevvol1          -1.692721   0.693791   -2.44    0.015 *
prevvol2          -0.108895   0.436020   -0.25    0.803
hilo               6.108500   0.694357    8.80  < 2e-16 ***
logvol            -0.000999   0.001989   -0.50    0.616
factor(above_avg)1 -0.005971   0.005710   -1.05    0.296
ch200ma           -0.000411   0.000283   -1.45    0.147
ch50ma             0.000730   0.000543    1.34    0.179
eps               -0.000261   0.000506   -0.52    0.605
logmarcap          0.005997   0.001424    4.21  3.0e-05 ***
logshortratio      0.015873   0.003725    4.26  2.4e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0402 on 485 degrees of freedom
Multiple R-squared:  0.476,    Adjusted R-squared:  0.465
F-statistic:   44 on 10 and 485 DF,  p-value: <2e-16
```

Using the exhaustive model since there are many predictors here which seem to be insignificant in predicting the response

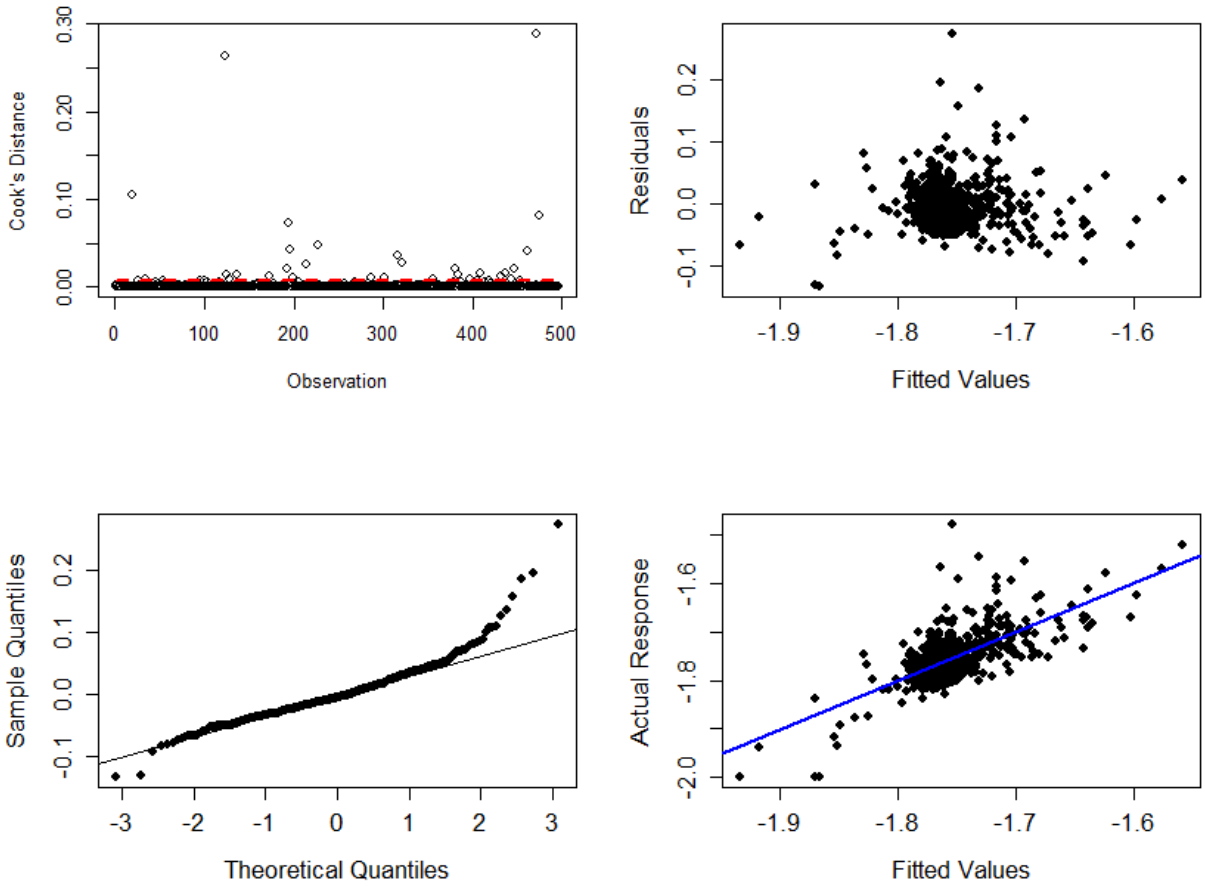
```
lm(formula = y ~ ., data = data.frame(Xy[, c(bestset[-1], FALSE),
    drop = FALSE], y = y))

Residuals:
```

```
     Min       1Q   Median       3Q      Max
-0.13425 -0.02455 -0.00495  0.01916  0.27534

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.93380    0.01374 -140.71  < 2e-16 ***
prevvol1      -1.91093    0.59830   -3.19   0.0015 **
hilo           6.20352    0.51358   12.08  < 2e-16 ***
logmarcap      0.00539    0.00117    4.60  5.4e-06 ***
logshortratio  0.01657    0.00356    4.66  4.1e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0401 on 491 degrees of freedom
Multiple R-squared:  0.472,    Adjusted R-squared:  0.467
F-statistic:  110 on 4 and 491 DF,  p-value: <2e-16
```

Performing an AIC based step procedure also provides the same model –

```
> aicmod = stepAIC(mod1)
transresp ~ prevvol1 + hilo + logmarcap + logshortratio
```

Diagnostic Plots – Linear Model (exhaustivemod – resulting model from exhaustive search in the full linear model)



Clearly there are not many influential observations as can be seen from the scale on the cooks distance

After running AIC on the both robust model and the exhaustive model we see low difference between the respective AICs. Thus there is not much need for the robust regression.

```
> AIC(robmod)
[1] -1768
> AIC(exhaustivemod)
[1] -1776
```

From the QQ line plot we can clearly see that the residuals are heavy tailed distributions and we should try non-parametric models to capture this distribution of volatilities.

From the fitted values and residuals plot and the fitted values and actual response plot we see that we need better fitting models.

General Additive Models

I first used all of the predictors available at my disposal to see which predictors are best suited

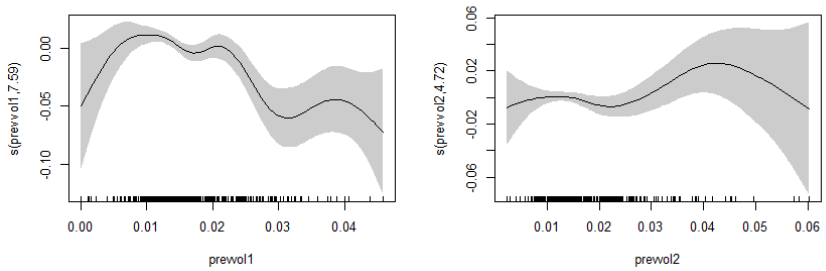
```
> gammod = gam(transresp~s(prevvol1)+s(prevvol2)+s(hilo)+s(logvol)+factor(abo
ve_avg)+s(ch200ma)+s(ch50ma)+s(eps)+s(logmarcap)+s(logshortratio))
> summary(gammod)

Family: gaussian
Link function: identity

Formula:
transresp ~ s(prevvol1) + s(prevvol2) + s(hilo) + s(logvol) +
    factor(above_avg) + s(ch200ma) + s(ch50ma) + s(eps) + s(logmarcap) +
    s(logshortratio)

Parametric coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        -1.74868    0.00349    -502   <2e-16 ***
factor(above_avg)1 -0.00835    0.00838      -1     0.32
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                  edf Ref.df     F p-value
s(prevvol1)      7.49   8.41  5.83 2.6e-07 ***
s(prevvol2)      4.94   6.07  1.94  0.0721 .
s(hilo)          1.31   1.54 40.63 7.2e-13 ***
s(logvol)        7.28   8.24  1.41  0.1885
s(ch200ma)       6.72   7.86  8.47 1.3e-10 ***
s(ch50ma)        8.03   8.75 16.01 < 2e-16 ***
s(eps)           1.00   1.00  2.19  0.1396
s(logmarcap)     1.00   1.00  1.09  0.2975
s(logshortratio) 6.52   7.61  3.12  0.0024 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.697   Deviance explained = 72.5%
GCV = 0.0010102  Scale est. = 0.00091586  n = 496
```

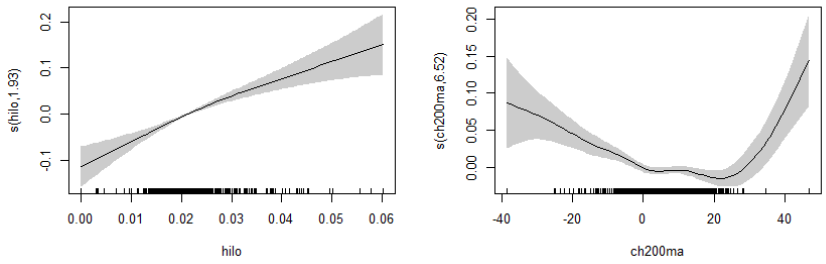
As clear from the summary, predictors above_avg, logvol, eps and logmarcap are not significant

After removing the above mentioned predictors and then re-running the General Additive model we get a better performance on the AIC

```
> gammod2 = gam(transresp~s(prevvol1)+s(prevvol2)+s(hilo)+s(ch200ma)+s(ch50ma
)+s(logshortratio))
> AIC(gammod2)
[1] -2014
> AIC(gammod)
[1] -2013
```

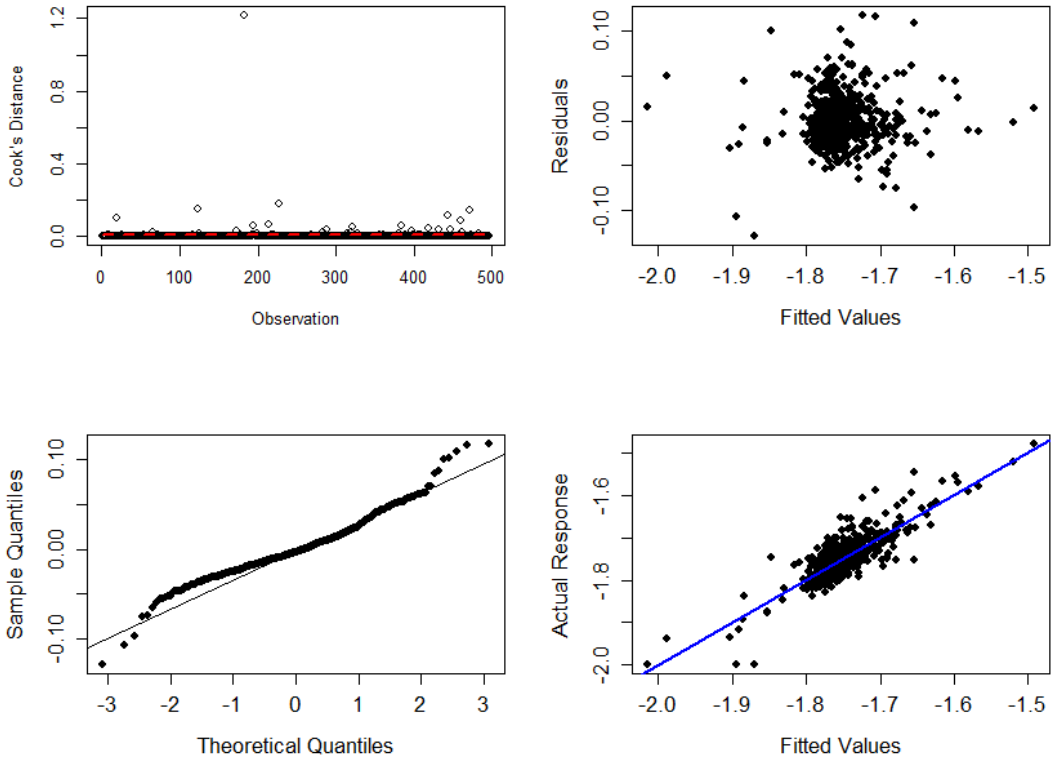
Below are some plots for the GAM model. Here we clearly see that Hilo has a linear relationship with the response. Thus we can just use the linear addition for Hilo and it may save us some degrees of freedom






```
> gammod3 = gam(transresp~s(prevvol1)+s(prevvol2)+hilo+s(ch200ma)+s(ch50ma)+s
(logshortratio))
> AIC(gammod3)
[1] -2015


> summary(gammod3)
Formula:
transresp ~ s(prevvol1) + s(prevvol2) + hilo + s(ch200ma) + s(ch50ma) +
    s(logshortratio)

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.8506     0.0131 -141.31  < 2e-16 ***
hilo          4.5376     0.5987    7.58  1.9e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                  edf Ref.df     F p-value
s(prevvol1)      7.93   8.69  7.71 2.8e-10 ***
s(prevvol2)      4.98   6.13  1.96   0.068 .
s(ch200ma)       6.67   7.81  8.18 3.8e-10 ***
s(ch50ma)        8.17   8.81 15.96 < 2e-16 ***
s(logshortratio) 6.57   7.62  7.20 1.2e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.692   Deviance explained = 71.4%
GCV = 0.001005  Scale est. = 0.00093145  n = 496
```

Plotting the diagnostics for the best GAM model



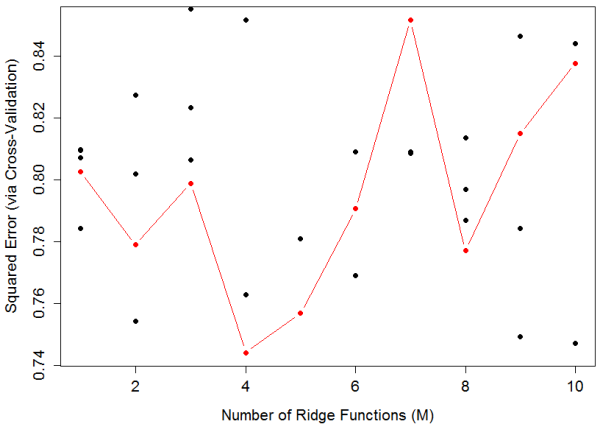
As indicated by the AIC, we see a much better performance for the fit and at the same time we have reduced the AIC from -1776 to -2015

- The cooks distance shows that there might be one influential observation, but based on the scale we can interpret that to be inconsequential
- The fit between the actual response and the fitted values is getting truer. The fit line is coming closer to 45 degrees in angle and towards the origin
- The residual value are showing little pattern which is also a better sign. Though some pattern does come out if we observe the outwards points

Project Pursuit Regression model

Running cross validation for obtaining the optimal parameters for running the PPR

```
source("http://www.stat.cmu.edu/~cschafer/MSCF/CVforppr.R")
modelformula = transresp ~ prevvol1 + prevvol2 + hilo + ch200ma + ch50ma + logshortratio
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(modelformula, nterms=i, numfolds=10,
                  sm.method="gcvspline")   }}
plot(1:nrow(pprCV),apply(pprCV,1,mean),type="b",pch=16,col=2,cex.axis=1.3,
    cex.lab=1.3,xlab="Number of Ridge Functions (M)",
    ylab="Squared Error (via Cross-Validation)")
for(j in 1:ncol(pprCV))
{  points(1:nrow(pprCV),pprCV[,j],pch=16) }
```

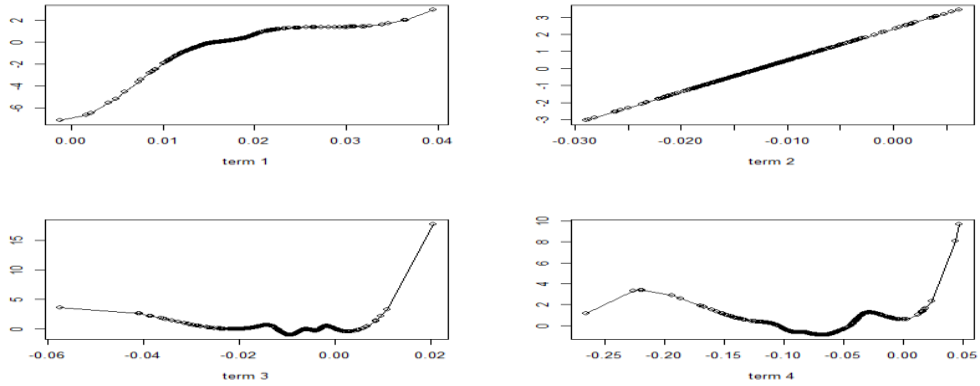


Clearly the most optimal choice for the number of ridge functions is 4

Running the PPR regression

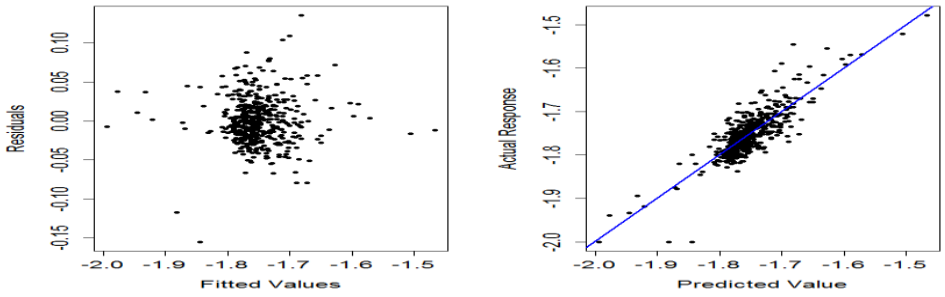
```
pprmod =
ppr(transresp~prevvol1+prevvol2+hilo+ch200ma+ch50ma+logshortratio,nterms=4,sm.method="gcvspli
ne")
par(mfrow=c(2,2))
plot(pprmod)
```

Please see below for the ridge functions



Finding the complexity of the PPR model
complexity_ppr = sum(pprmod$edf) + length(pprmod$beta) + length(pprmod$alpha)+1
63.91

<u>Plotting the diagnostic plots for PPR</u>



We are clearly seeing that the fit has not improved drastically and the fit between the actual response and predicted values is still not at 45 degrees.

Comparing the GAM model and the PPR model
GAM

```
> deviance(gammod3)
[1] 0.4282
> sum(gammod3$edf)
[1] 36.31
```

PPR

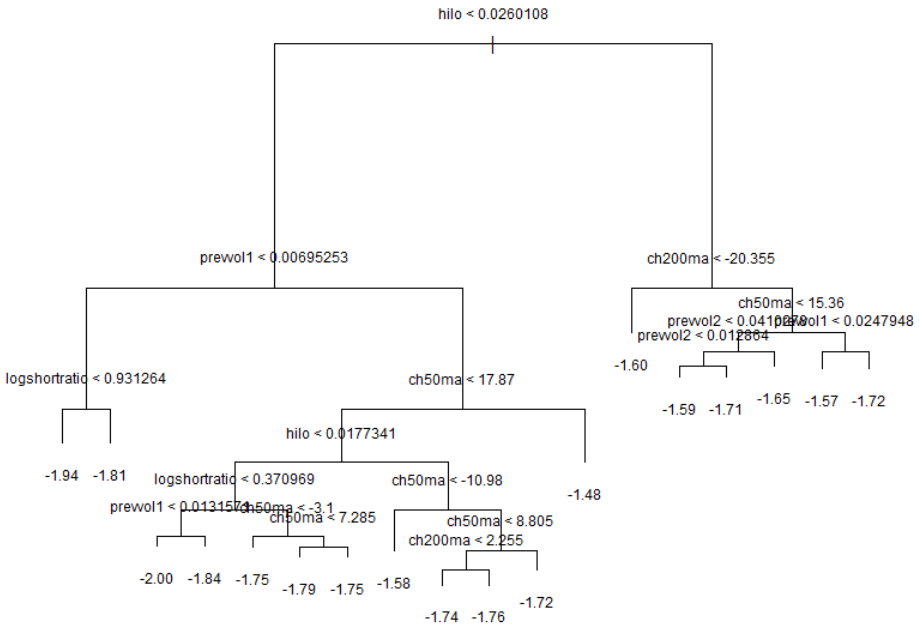
```
> complexity_ppr = sum(pprmod$edf) + length(pprmod$beta) + length(pprmod$alpha)+1
> pprmod$gof
[1] 0.4387
> complexity_ppr
[1] 63.91
```

From visual inspection and from comparing the deviance and the complexity we see that the GAM model fits much better. If we increase the nterms in PPR then we can easily get deviance lower than GAM but we already see that the complexity has gone up a lot. Thus we can easily choose GAM over PPR

Tree Based Model

Using the cross validation to find the optimal K such that the adjusted complexity value for the tree which includes the deviance plus K*number of terminal nodes,

We get the following tree



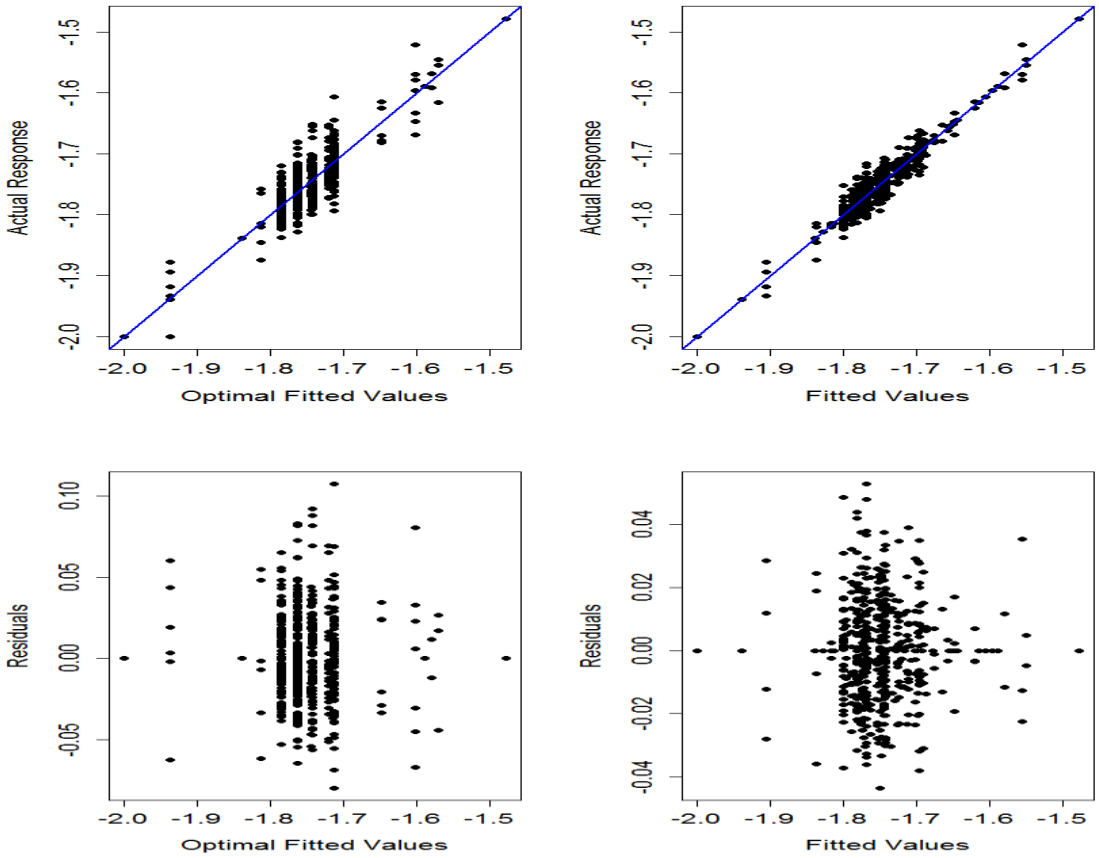

```
#Tree Based Model
fulltree = tree(transresp~prevvol1+prevvol2+hilo+ch200ma+ch50ma+logshortratio, mindev=0, minsize=2)
cvout = cv.tree.full(fulltree)
plot(cvout)
optalpha = cvout$k[which.min(cvout$dev)]
opttree = prune.tree(fulltree, k=optalpha)
plot(opttree)
text(opttree, cex=0.75, digits = 3)
```

Optimal K was 0.01258

If we decrease the alpha so as to increase the complexity of the model and predict more reliably, we get a better response. Decreasing alpha reduces the penalty for increasing the number of terminal nodes and hence allows more complexity

```
prunedtree = prune.tree(fulltree, k=0.002)
plot(as.numeric(predict(prunedtree)),as.numeric(transresp), pch=16,xlab="Fitted Values", ylab="Actual
Response",cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=4)
```

Plotting the diagnostic plots for both optimal tree and pruned tree



As we can see from the plots, the optimal fitted tree does provides a very discrete model which may not suit our purposes, thus a more complex model is fit. This model has 81 terminal nodes and thus 81 degrees of freedom are used in modelling. The optimal tree has only 18 degrees of freedom.

```
> summary(prunedtree)

Regression tree:
snip.tree(tree = fulltree, nodes = c(19L, 58L, 59L, 61L, 670L,
671L, 1329L, 1382L, 1834L, 5326L, 5512L, 5546L, 42L, 690L, 1392L,
1833L, 2776L, 175L, 912L, 2669L, 2774L, 11028L, 24L, 5547L, 334L,
2783L, 5336L, 5513L, 697L, 1328L, 459L, 913L, 5327L, 5550L, 5559L,
14685L, 349L, 2777L, 11117L, 1335L, 5515L, 5518L, 115L, 1330L,
2758L, 2772L, 17L, 692L, 11029L, 5337L, 13L, 18L, 2662L, 22204L,
82L, 666L, 2782L, 7343L))
Number of terminal nodes:  81
Residual mean deviance:  0.000312 = 0.129 / 415
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.0434 -0.0108  0.0000  0.0000  0.0108  0.0526
```

We can see that the residual is 0.407 which is lower than the GAM model and also the number of terminal nodes is 18 which is less than the complexity in the GAM model of 36. But due to discreteness of the tree model I am inclined to choose the GAM model over the tree model

```
> summary(opttree)
Regression tree:
snip.tree(tree = fulltree, nodes = c(42L, 82L, 31L, 30L, 29L,
166L, 87L, 167L, 173L, 9L, 57L, 8L, 6L, 172L))
Number of terminal nodes:  18
Residual mean deviance:  0.000852 = 0.407 / 478
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.07950 -0.01960 -0.00288  0.00000  0.01870  0.10700
```

Appendix – R code for setup

```
library(quantmod)
library(tseries)
library(stringi)
library(MASS)
library(car)
library(mgcv)
library(bestglm)
library(tree)
SymbolList = read.table("http://www.stat.cmu.edu/~cschafer/MSCF/Project/ChallengeSymbols2015.txt")
Longlistall = read.csv("https://s3.amazonaws.com/quandl-static-content/Ticker+CSV%27s/Indicies/SP500.csv")
Longlist = Longlistall[,1]
symbols = Longlist
symbols2 = data.frame()
# Code for modelling
SymbolList2 = data.frame(SymbolList)
SymbolList2 = SymbolList2[-5,]


prevvol1 = numeric(length(symbols))
prevvol2 = numeric(length(symbols))
logvol = numeric(length(symbols))
hilo = numeric(length(symbols))
above_avg = numeric(length(symbols))

resp = numeric(length(symbols))

#Getting the Response
for(i in 1:length(symbols)){
 tryCatch({holddata1 = getSymbols(as.character(symbols[i]), from=Sys.Date()-31, to=Sys.Date()-1, auto.assign=F)
 resp[i] = sqrt(mean(dailyReturn(Ad(holddata1),type="log")^2))
 },error=function(e){resp[i]=0})
}
resp

for(i in 1:length(symbols)){
 tryCatch({
   if(resp[i]==0){resp[i]=0.0000000001}
 },error=function(e){resp[i]=0})
}

#The Predictors
for(i in 1:length(symbols)){
 tryCatch({holddata = getSymbols(as.character(symbols[i]), from=(Sys.Date()-61), to=Sys.Date()-31, auto.assign=F)
 prevvol1[i] = sqrt(mean(dailyReturn(Ad(holddata),type="log")^2))
 logvol[i] = log(mean(as.numeric(holddata[,c(5)])))
 hilo[i] = mean((as.numeric(holddata[,c(2)]) - as.numeric(holddata[,c(3)]))/as.numeric(holddata[,c(6)]))
 },error=function(e){})
}
prevvol1;
```

```
for(i in 1:length(symbols)){
  tryCatch({
    if(is.na(prevvol1[i])){prevvol1[i]=0}
    if(is.na(logvol[i])){logvol[i]=0}
    if(is.na(hilo[i])){hilo[i]=0}
    },error=function(e){prevvol1[i]=0})
}

stockvolavg = mean(prevvol1);
for(i in 1:length(symbols)){
  tryCatch({
  holddata2 = getSymbols(as.character(symbols[i]), from=(Sys.Date()-91), to=Sys.Date()-61, auto.assign=F);
  prevvol2[i] = sqrt(mean(dailyReturn(Ad(holddata2),type="log")^2));
  if(prevvol1[i]>stockvolavg) above_avg[i] = 1 else above_avg[i] = 0;
  },error=function(e){})
}
prevvol2

main_metrics <- yahooQF(c(
  "Short Ratio",
  "Percent Change From 50-day Moving Average",
  "Percent Change From 200-day Moving Average",
  "Earnings/Share",
  "Market Capitalization"
  ))
metrics <- getQuote(paste(symbols, sep="", collapse=";"), what=main_metrics, start=Sys.Date()-30)
metrics = data.frame(metrics[,-1])
marcapmetrics = data.frame(metrics[,5])
names(metrics[5])
MarCap=numeric(length(symbols))
for(i in 1:length(symbols))
{ tryCatch({
  string_val = marcapmetrics[i,1]
  num_val = as.numeric(substr(string_val,1,stri_length(string_val)-1));
  if (substr(string_val,stri_length(string_val),stri_length(string_val)) == "B"){
    num_val = num_val * 1000
  } else if (substr(string_val,stri_length(string_val),stri_length(string_val)) == "M"){
    num_val = num_val
  } else {num_val = 0}

  MarCap[i]=num_val
 }, error=function(e){num_val =0
                      MarCap[i]=num_val    })
}
logmarcap= log(MarCap+1)

shortratiometrics = data.frame(metrics[,1])
Shortratio=numeric(length(symbols))
for(i in 1:length(symbols))
{ string_val = shortratiometrics[i,1]
  if(string_val=="N/A"){num_val=0
  } else{num_val = as.numeric(substr(string_val,1,stri_length(string_val)-1))}
  Shortratio[i] = num_val
```

```
}
logshortratio= log(Shortratio+1)

ch50mametrics = data.frame(metrics[,2])
ch50ma=numeric(length(symbols))
for(i in 1:length(symbols))
{ string_val = ch50mametrics[i,1]
  if(string_val=="N/A"){num_val=0
  } else{num_val = as.numeric(substr(string_val,1,stri_length(string_val)-1))}
  ch50ma[i] = num_val
}

ch200mametrics = data.frame(metrics[,3])
ch200ma=numeric(length(symbols))
for(i in 1:length(symbols))
{ string_val = ch200mametrics[i,1]
  if(string_val=="N/A"){num_val=0
  } else{num_val = as.numeric(substr(string_val,1,stri_length(string_val)-1))}
  ch200ma[i] = num_val
}

epsmetrics = data.frame(metrics[,4])
eps=numeric(length(symbols))
for(i in 1:length(symbols))
{ string_val = epsmetrics[i,1]
  if(string_val=="N/A"){num_val=0
  } else{num_val = as.numeric(substr(string_val,1,stri_length(string_val)-1))}
  eps[i] = num_val
}
```

#Models

#Linear Model

```
boxCox(resp~prevvol1+prevvol2+hilo+logvol+factor(above_avg)+ch200ma+ch50ma+eps+logmarcap+logshortratio)
transresp = bcPower(resp, 0.5)
mod1 =
lm(transresp~prevvol1+prevvol2+hilo+logvol+factor(above_avg)+ch200ma+ch50ma+eps+logmarcap+logshortratio
)
summary(mod1)
AIC(mod1)
#Exhaustive Mod
XYFrame =
data.frame(prevvol1,prevvol2,hilo,logvol,factor(above_avg),ch200ma,ch50ma,eps,logmarcap,logshortratio,transre
sp)
bestglmresult = bestglm(XYFrame, IC="AIC", method = "exhaustive")
exhaustivemod = bestglmresult$BestModel
summary(exhaustivemod)
plot_diagnostics(transresp,exhaustivemod)
#AIC Based Stepmod
aicmod = stepAIC(mod1)
#Robust Regression
robmod = rlm(transresp~prevvol1+hilo+logmarcap+logshortratio)
summary(robmod)
```

```
AIC(robmod)
AIC(exhaustivemod)
```

#General Additive Models

```
gammod =
gam(transresp~s(prevvol1)+s(prevvol2)+s(hilo)+s(logvol)+factor(above_avg)+s(ch200ma)+s(ch50ma)+s(eps)+s(log
marcap)+s(logshortratio))
summary(gammod)
gammod2 = gam(transresp~s(prevvol1)+s(prevvol2)+s(hilo)+s(ch200ma)+s(ch50ma)+s(logshortratio))
summary(gammod2)
AIC(gammod)
AIC(gammod2)
plot(gammod2, pages =2, scale=0,scheme=1)
gammod3 = gam(transresp~s(prevvol1)+s(prevvol2)+hilo+s(ch200ma)+s(ch50ma)+s(logshortratio))
plot(gammod3, pages =2, scale=0,scheme=1)
AIC(gammod3)
summary(gammod3)
deviance(gammod3)
sum(gammod3$edf)
plot_diagnostics(transresp, gammod3)
```

#PPR Model and crossvalidation

```
source("http://www.stat.cmu.edu/~cschafer/MSCF/CVforppr.R")
modelformula = transresp ~ prevvol1 + prevvol2 + hilo + ch200ma + ch50ma + logshortratio
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(modelformula, nterms=i, numfolds=10,
                    sm.method="gcvspline")
 }
}
plot(1:nrow(pprCV),apply(pprCV,1,mean),type="b",pch=16,col=2,cex.axis=1.3,
    cex.lab=1.3,xlab="Number of Ridge Functions (M)",
    ylab="Squared Error (via Cross-Validation)")

for(j in 1:ncol(pprCV))
{
 points(1:nrow(pprCV),pprCV[,j],pch=16)
}

pprmod =
ppr(transresp~prevvol1+prevvol2+hilo+ch200ma+ch50ma+logshortratio,nterms=4,sm.method="gcvspline")
par(mfrow=c(1,2))
plot(pprmod)
summary(pprmod)
complexity_ppr = sum(pprmod$edf) + length(pprmod$beta) + length(pprmod$alpha)+1
pprmod$
plot(pprmod$fit, pprmod$residuals, xlab="Fitted Values",
ylab="Residuals",cex.axis=1.3,cex.lab=1.3,pch=16,cex=0.7)
```

```
plot(predict(pprmod),transresp,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=4)
```

#Tree Based Model

```
fulltree = tree(transresp~prevvol1+prevvol2+hilo+ch200ma+ch50ma+logshortratio, mindev=0, minsize=2)
prunedtree = prune.tree(fulltree, k=0.002)
plot(prunedtree)
text(prunedtree, cex=0.75,digits=4)
cvout = cv.tree.full(fulltree)
plot(cvout)
optalpha = cvout$k[which.min(cvout$dev)]
opttree = prune.tree(fulltree, k=optalpha)
plot(opttree)
text(opttree, cex=0.75, digits = 3)
summary(opttree)
summary(prunedtree)
opttree$
par(mfrow=c(1,1))
plot(as.numeric(predict(prunedtree)),as.numeric(transresp), pch=16,xlab="Fitted Values", ylab="Actual
Response",cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=4)
plot(as.numeric(predict(opttree)),as.numeric(residuals(opttree)), pch=16,xlab="Optimal Fitted Values",
ylab="Residuals",cex.axis=1.3,cex.lab=1.3)
plot(as.numeric(predict(prunedtree)),as.numeric(residuals(prunedtree)), pch=16,xlab="Fitted Values",
ylab="Residuals",cex.axis=1.3,cex.lab=1.3)
```

#Cross Validation Code for regression trees

```
cv.tree.full <- function (object, rand, FUN = prune.tree, K = 10, mindev=0, mincut=1, minsize=2,...)
{
 require(tree)
 if (!inherits(object, "tree"))
  stop("Not legitimate tree")
 m <- model.frame(object)
 extras <- match.call(expand.dots = FALSE)$...
 FUN <- deparse(substitute(FUN))
 init <- do.call(FUN, c(list(object), extras))
 if (missing(rand))
  rand <- sample(K, length(m[[1]]), replace = TRUE)
 cvdev <- 0
 for (i in unique(rand)) {
  tlearn <- tree(model = m[rand != i, , drop = FALSE],mindev=mindev,mincut=mincut,minsize=minsize)
  plearn <- do.call(FUN, c(list(tlearn, newdata = m[rand ==
                                    i, , drop = FALSE], k = init$k), extras))
  cvdev <- cvdev + plearn$dev
 }
 init$dev <- cvdev
 init
}
```

#Diagnostics Plot

```
plot_diagnostics(transresp,exhaustivemod)

plot_diagnostics <- function(resp, curmod, model_name = "", file_name = ""){
par(mfrow=c(2,2))
cookd = as.numeric(cooks.distance(curmod))
plot(cookd, xlab= "Observation", ylab = "Cook's Distance")
lines(c(1, length(cookd)),c(4/length(cookd), 4/length(cookd)),lwd =2, col = 2, lty =2)
plot(as.numeric(curmod$fitted.values),as.numeric(curmod$residuals), pch=16,xlab="Fitted Values",
ylab="Residuals",cex.axis=1.3,cex.lab=1.3)
qqnorm(as.numeric(curmod$residuals),cex.axis=1.3,cex.lab=1.3,pch=16,main="")
qqline(as.numeric(exhaustivemod$residuals))
plot(as.numeric(curmod$fitted.values),as.numeric(resp), pch=16,xlab="Fitted Values", ylab="Actual
Response",cex.axis=1.3,cex.lab=1.3)
abline(0,1,lwd=2,col=4)
}

#Extra

summary(mod1)
summary(gammod3)
```

#Comparisons

```
sum(robmod$residuals^2)
sum(exhaustivemod$residuals^2)
sum(mod1$residuals^2)
sum(gammod3$residuals^2)
sum(pprmod$residuals^2)
AIC(robmod)
AIC(mod1)
AIC(exhaustivemod)
AIC(gammod3)

#Finishing up
print("Saving best model");
finalmodel = gammod3
save(finalmodel, file="Abhatna1_Models.Robj");
```