Project_SingRs

February 16, 2019

Introduction

- Business objective. Predicting movements of stock market is a big topic today within the financial industry. Banks, hedge funds and other organizations always in the constant research to improve accuracy of prediction and therefore to maximize return for investors.

Purpose of this project is make an attemp to predict the movement of S&P 500 index.

- Process The analysis is splitted into three parts:

1. Application of random forest on data about S&P 500 index
2. Time series analysis and prediction using ets and arima
3. Reproduction of Hidden Markov Model for Financial Time Series and Its Application to S&P 500 Index paper by Stephen H.-T. Lihn to illustrate Hidden Markov Model application

- Data Most of the financial data on the S&P 500 index is downloaded from CRSP data depository at Wharton Research Data Services Also data on the volatility index is being scrapped from CBOE using quantmod package and getSymbols function For Hidden Markov Model we use package ldhmm. All the data for S&P 500 and VIX index is predownloaded in the package.

- Results As we said in the beginning prediction of stock market movement is a complicated task that requires not only full and comprehensive understanding of stock market, macro environment, political situation etc but also a lot of computational power and knowledge of advanced math and statistical tools.

Our analysis showed that using random forest with limited data gives model with 53% accuracy, time series analysis model produces massive error and only HMM could predict volatility with certain degree of accuracy.

-Acknowledgement: Christos Oikonomou, Phd INSEAD, Frederico Belo, Associate Professor of Finance, Ahmed Guecioueur, Phd INSEAD
Paper by Stephen H.-T. Lihn “Hidden Markov Model for Financial Time Series and Its Application to S&P 500 Index (https://cran.r-project.org/web/packages/ldhmm/vignettes/ldhmm-spx.pdf (https://cran.r-project.org/web/packages/ldhmm/vignettes/ldhmm-spx.pdf))

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
## Loading required package: quantmod
```

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
##
## Attaching package: 'moments'
```

```
## The following objects are masked from 'package:e1071':
##
##     kurtosis, moment, skewness
```

```
## Warning: package 'forecast' was built under R version 3.5.2
```

```
## Loading required package: fma
```

```
##
## Attaching package: 'fma'
```

```
## The following objects are masked from 'package:MASS':
##
##     cement, housing, petrol
```

```
## Loading required package: expsmooth
```

```
## Loading required package: lmtest
```

```
## Loading required package: tseries
```

Part 1. Prediction of S&P 500 movement using Random Forest

In the first part of the analysis we process S&P 500 index data in random forest to indentify whether we can predict the next day movement.

Data is being taken from CRSP WRDS.

Load data

```
p1_data_general <- read.csv("sp500_master_woT0.csv")
data_working <- p1_data_general
```

For the model we've created a lagged variables for yesterday (T.1), the day before yesterday (T.2) and 30 days before yesterday (T.30).

- Data dictionary: caldt - Calendar date ewretd - Equal-Weighted Return (includes distributions)
  ewretx - Equal-Weighted Return (excluding dividends) spindx - Level on S&P Composite Index
  sprtrn - Return on S&P Composite Index
  totcnt - Total Market Count
  totval - Total Market Value
  usdcnt - Count of Securities Used
  usdval - Market Value of Securities Used vwretd - Value-Weighted Return (includes distributions)
  vwretx - Value-Weighted Return (excluding dividends) chusdval - Change in usdval vol - Volatility calculated as 30-days average volatility Movement - index movement (if 1 index went up, 0 - down)

```
data_working$caldt <- as.Date(as.character(data_working$caldt), "%Y%m%d")
data_working$Movement.T.0 <- as.factor(data_working$Movement.T.0)
str(data_working)
```

```
## 'data.frame':    4469 obs. of  32 variables:
##  $ caldt        : Date, format: "2001-02-13" "2001-02-14" ...
##  $ vwretd.T.30  : num  -0.00853 -0.00194 0.00839 -0.019 -0.01733 ...
##  $ vwretx.T.30  : num  -0.00861 -0.00213 0.00831 -0.01902 -0.01734 ...
##  $ ewretd.T.30  : num  -0.002075 0.000228 0.009689 -0.014006 -0.013838 ...
##  $ ewretx.T.30  : num  -0.00217 0.000009 0.009581 -0.014066 -0.01385 ...
##  $ totval.T.30  : num  1.16e+10 1.16e+10 1.17e+10 1.15e+10 1.13e+10 ...
##  $ usdval.T.30  : num  1.17e+10 1.16e+10 1.16e+10 1.17e+10 1.15e+10 ...
##  $ spindx.T.30  : num  1319 1316 1327 1302 1279 ...
##  $ sprtrn.T.30  : num  -0.00865 -0.00218 0.00812 -0.01891 -0.01736 ...
##  $ chusdval.T.30: num  1.012 0.991 0.999 1.009 0.981 ...
##  $ vol.T.30     : num  0.01413 0.01311 0.0094 0.00982 0.00914 ...
##  $ vwretd.T.2   : num  0.0199 0.0111 0.0255 -0.0243 -0.0047 ...
##  $ vwretx.T.2   : num  0.01988 0.01111 0.02546 -0.02436 -0.00472 ...
##  $ ewretd.T.2   : num  0.0183 0.01315 0.01438 -0.02091 -0.00304 ...
##  $ ewretx.T.2   : num  0.0183 0.01315 0.01438 -0.02106 -0.00309 ...
##  $ totval.T.2   : num  1.01e+10 1.02e+10 1.04e+10 1.02e+10 1.01e+10 ...
##  $ usdval.T.2   : num  9.87e+09 1.01e+10 1.02e+10 1.04e+10 1.02e+10 ...
##  $ spindx.T.2   : num  1140 1153 1182 1153 1148 ...
##  $ sprtrn.T.2   : num  0.01991 0.01128 0.02558 -0.02443 -0.00463 ...
##  $ chusdval.T.2 : num  0.996 1.02 1.011 1.025 0.976 ...
##  $ vol.T.2      : num  0.0154 0.0156 0.0163 0.0167 0.0167 ...
##  $ vwretd.T.1   : num  0.0111 0.0255 -0.0243 -0.0047 0.0108 ...
##  $ vwretx.T.1   : num  0.01111 0.02546 -0.02436 -0.00472 0.01081 ...
##  $ ewretd.T.1   : num  0.01315 0.01438 -0.02091 -0.00304 0.00975 ...
##  $ ewretx.T.1   : num  0.01315 0.01438 -0.02106 -0.00309 0.00975 ...
##  $ totval.T.1   : num  1.02e+10 1.04e+10 1.02e+10 1.01e+10 1.03e+10 ...
##  $ usdval.T.1   : num  1.01e+10 1.02e+10 1.04e+10 1.02e+10 1.01e+10 ...
##  $ spindx.T.1   : num  1153 1182 1153 1148 1160 ...
##  $ sprtrn.T.1   : num  0.01128 0.02558 -0.02443 -0.00463 0.01078 ...
##  $ chusdval.T.1 : num  1.02 1.011 1.025 0.976 0.995 ...
##  $ vol.T.1      : num  0.0156 0.0163 0.0167 0.0167 0.0168 ...
##  $ Movement.T.0 : Factor w/ 2 levels "0","1": 2 1 1 2 1 1 1 2 1 2 ...
```

Split the data into testing and training

```
train_general <- data_working[1:3500,]
test_general <- data_working[3501:4469,]
test_general_womov <- test_general
test_general_womov$MOVEMENT.T.0 <- NULL

set.seed(77300)
inTrain <- createDataPartition(y = train_general$Movement.T.0, p = 0.7, list = FAL
SE)
training <- train_general[ inTrain,]
testing <- train_general[ -inTrain,]
```

Random forest - only dynamic variables

Random forest analysis showed that total valuation return are the most significant variables in the model.

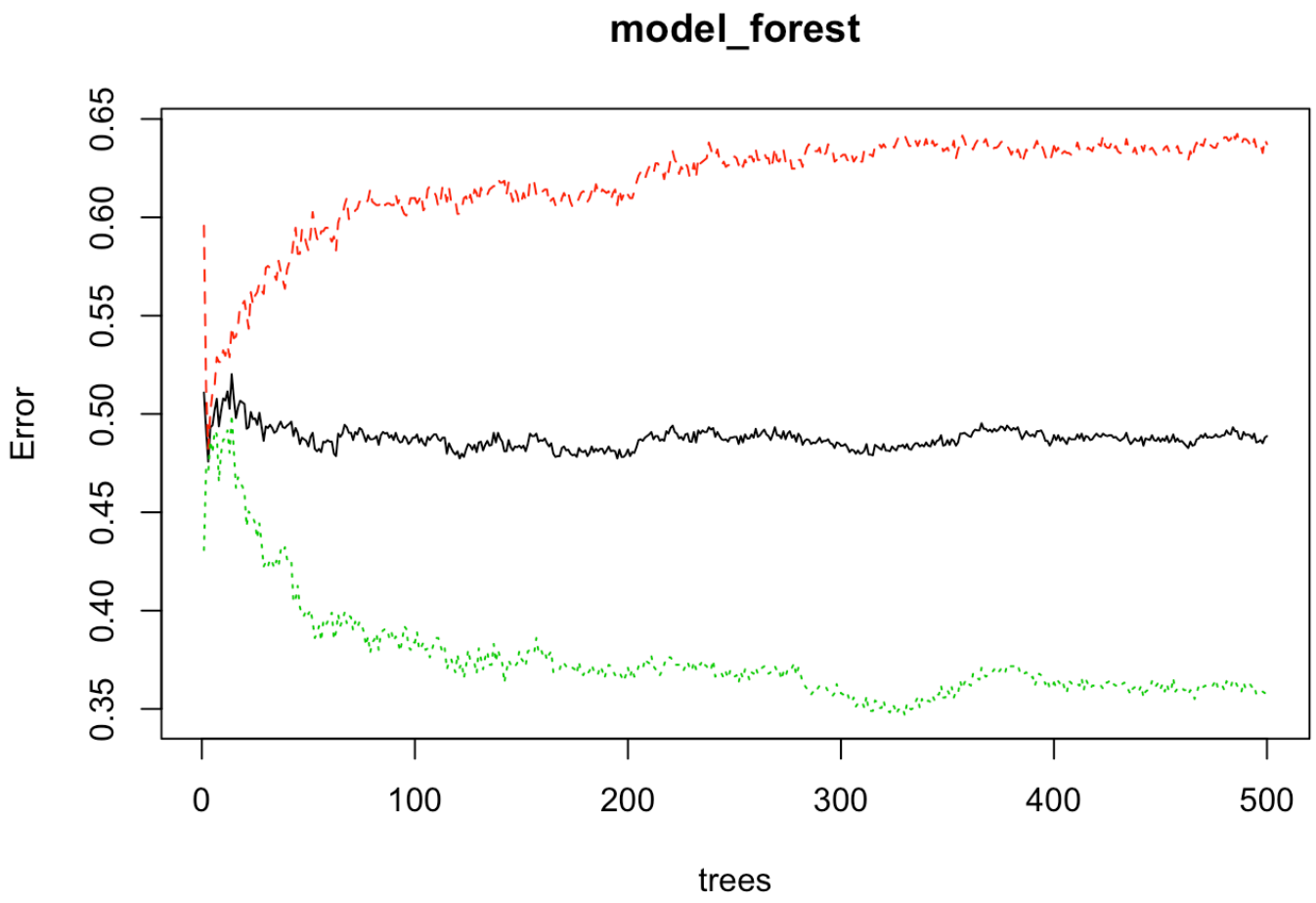
```
start_time <- Sys.time()
model_forest <- randomForest(Movement.T.0~., data=training, importance=TRUE, proxi
mity=TRUE,  type="classification")
print(model_forest)
```

```
##
## Call:
##  randomForest(formula = Movement.T.0 ~ ., data = training, importance = TRUE,
proximity = TRUE, type = "classification")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 48.88%
## Confusion matrix:
##     0   1 class.error
## 0 410 720   0.6371681
## 1 478 843   0.3618471
```

```
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 11.78921 secs
```

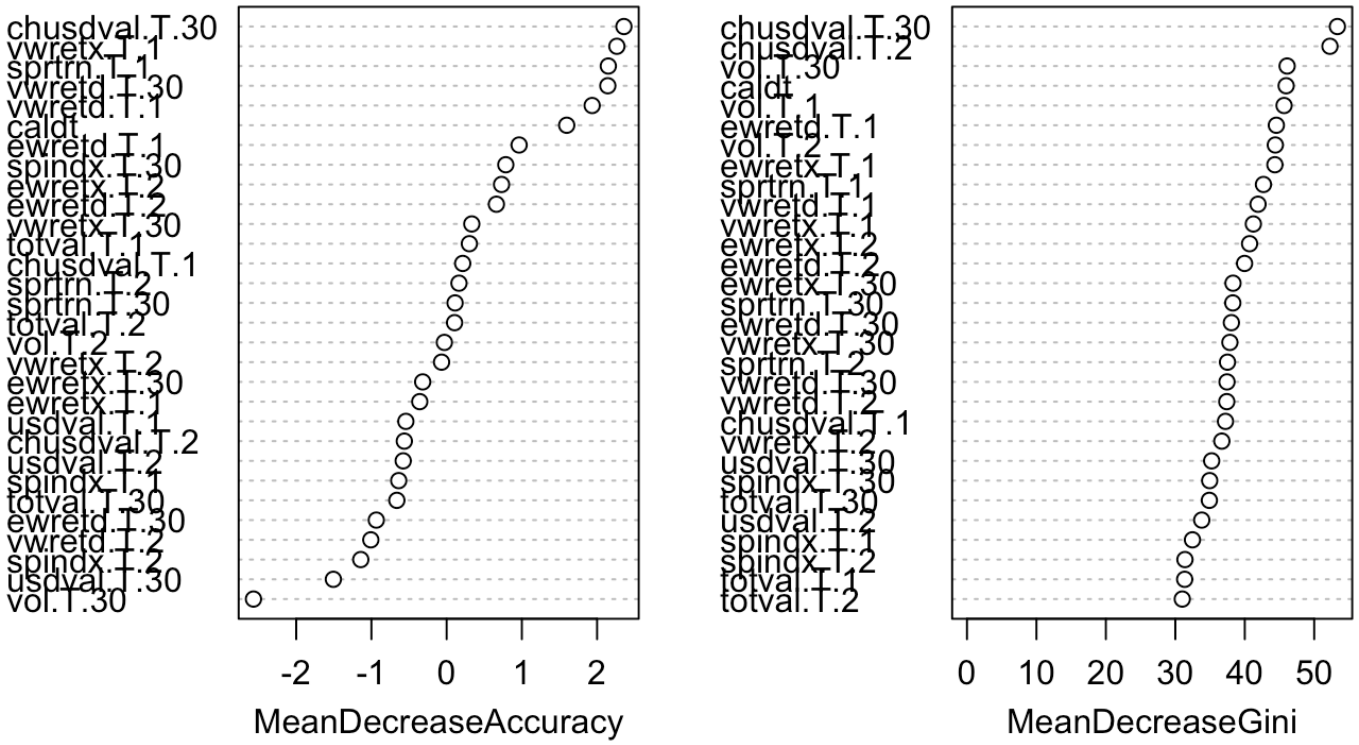
```
plot(model_forest)
```




```
importance(model_forest)
```

```
##                       0         1 MeanDecreaseAccuracy MeanDecreaseGini
## caldt         -1.351745 2.6976840           1.59627852         45.95973
## vwretd.T.30   -5.045713 6.6430205           2.14251643         37.44045
## vwretx.T.30   -5.017993 4.8904851           0.33415721         37.85538
## ewretd.T.30   -6.495307 4.9030606          -0.93500338         38.07333
## ewretx.T.30   -5.804591 4.9264694          -0.31849338         38.30711
## totval.T.30   -4.720884 3.8328239          -0.66194343         34.90097
## usdval.T.30   -7.695522 5.4307112          -1.50600609         35.23213
## spindx.T.30   -4.035256 4.2511138           0.78623132         34.94441
## sprtrn.T.30   -4.431928 4.1299831           0.11191548         38.27720
## chusdval.T.30  2.542415 0.6739320           2.35658587         53.33965
## vol.T.30      -3.798746 0.9420650          -2.56775059         46.09850
## vwretd.T.2    -4.787311 3.6766743          -1.00798917         37.40509
## vwretx.T.2    -4.295238 3.9447928          -0.06735237         36.69826
## ewretd.T.2    -6.599951 6.7609225           0.66160383         39.96983
## ewretx.T.2    -6.913611 6.6126879           0.72979924         40.69660
## totval.T.2    -3.827891 3.4258261           0.10536057         31.00209
## usdval.T.2    -3.840011 3.0168458          -0.57757769         33.80509
## spindx.T.2    -3.533409 2.2967705          -1.14139324         31.38332
## sprtrn.T.2    -5.596566 5.2550357           0.16397912         37.50983
## chusdval.T.2  -4.305597 3.2614064          -0.56260468         52.26525
## vol.T.2       -3.694305 3.2765180          -0.03009114         44.39859
## vwretd.T.1    -4.671976 5.2389610           1.93451119         41.90816
## vwretx.T.1    -3.191075 4.1032336           2.26317816         41.23638
## ewretd.T.1    -7.482329 7.7798505           0.96359221         44.54607
## ewretx.T.1    -6.473259 5.4651067          -0.35758210         44.33865
## totval.T.1    -4.423356 3.9878462           0.30206807         31.34807
## usdval.T.1    -3.397893 2.5855177          -0.54223585         30.42015
## spindx.T.1    -4.427593 3.4355230          -0.63711668         32.46576
## sprtrn.T.1    -4.919392 5.7382487           2.14913283         42.69093
## chusdval.T.1  -5.031924 4.5773288           0.21356972         37.22139
## vol.T.1       -3.968675 0.7953499          -2.86389347         45.65894
```

```
varImpPlot(model_forest)
```



Finding predicitons: probabilities and classification

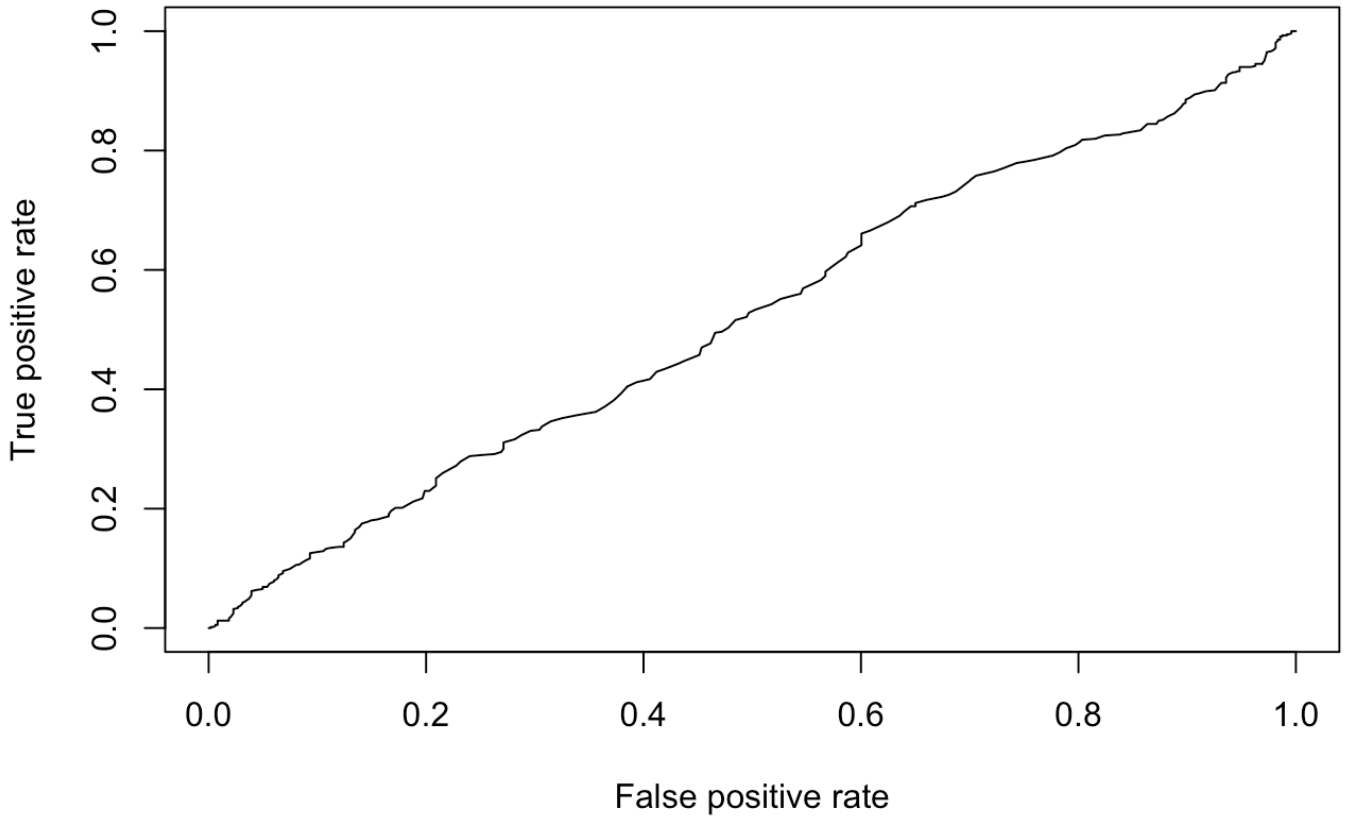
Confustion matrix analysis showed the accuracy of 53%, which is above flip-a-coin probability. Considering that stock market movements are

```
forest_probabilities<-predict(model_forest,newdata=testing,type="prob")
forest_classification<-rep("0",length(testing$Movement.T.0))
forest_classification[forest_probabilities[,2]>0.5]="1"
forest_classification<-as.factor(forest_classification)
confusionMatrix(forest_classification,testing$Movement.T.0, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 193 203
##          1 290 363
##
##                Accuracy : 0.53
##                  95% CI : (0.4993, 0.5606)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : 0.7424328
##
##                   Kappa : 0.0415
##  Mcnemar's Test P-Value : 0.0001074
##
##             Sensitivity : 0.6413
##             Specificity : 0.3996
##          Pos Pred Value : 0.5559
##          Neg Pred Value : 0.4874
##              Prevalence : 0.5396
##          Detection Rate : 0.3460
##    Detection Prevalence : 0.6225
##       Balanced Accuracy : 0.5205
##
##        'Positive' Class : 1
##
```

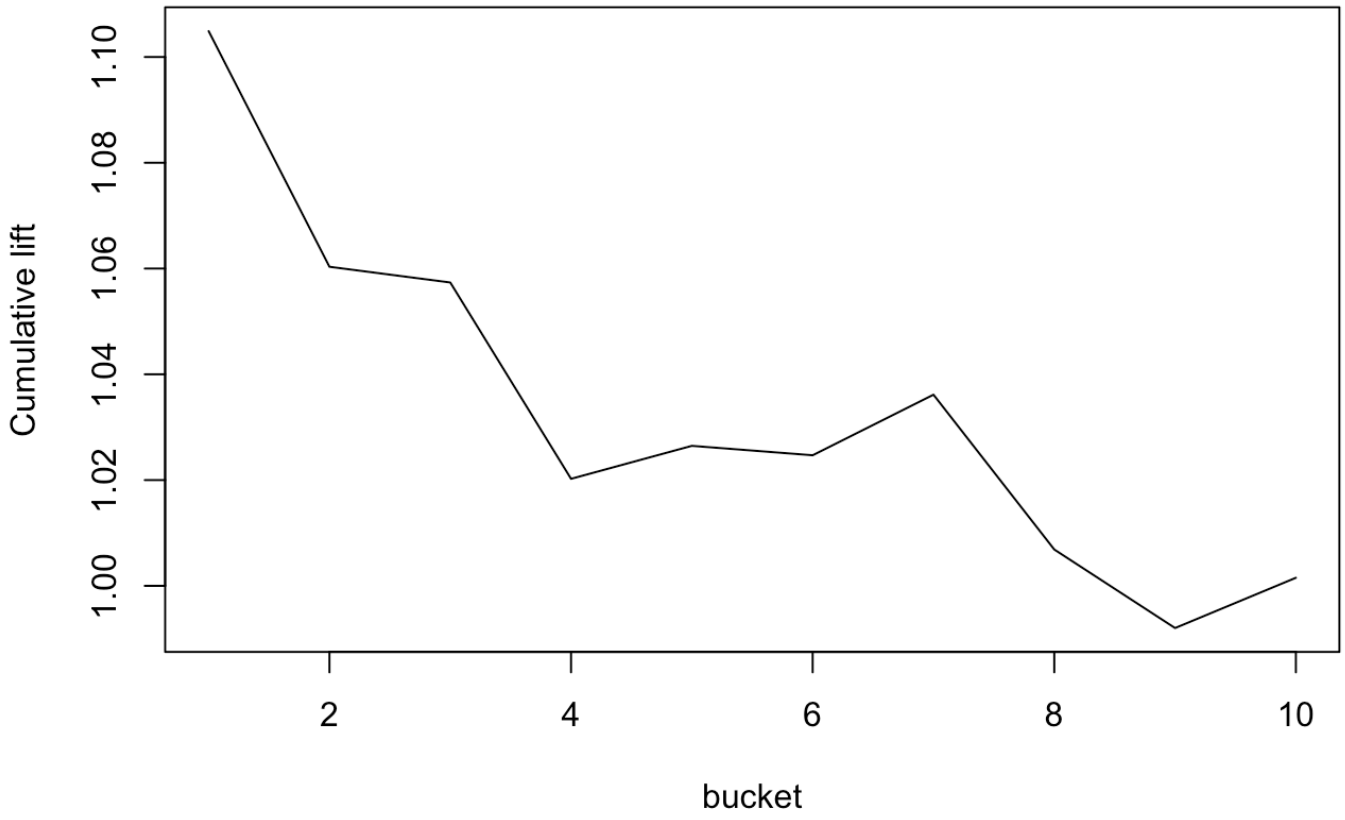
ROC Curve

```
forest_ROC_prediction <- prediction(forest_probabilities[,2], testing$Movement.T.0
)
forest_ROC_prediction <- performance(forest_ROC_prediction,"tpr","fpr")
plot(forest_ROC_prediction)
```



Lift chart

```
plotLift(forest_probabilities[,2],  testing$Movement.T.0, cumulative = TRUE, n.buc
kets = 10)
```



Part 2. Forecasting S&P500 movement using time series analysis

Description

We tried to predict the S&P Index's value for the next year using a set of time series forecasting methods - ETS, TBATS and auto arima.

For ETS, the model was unable to accomodate seasonality across a frequency of 252, hence we resolved to MAN parameters (multiplicative ERROR, additive TREND, no SEASONALITY).

TBATS and auto-arima do a better job at handling possible seasonality - a forecast for the next cycle is illustrated below.

ACF residuals appear to be within bounds - however the forecast does not appear very informative. The index is expected to remain flat albeit with room for volatility within 85% and 90% confidence bands.

Loading the data

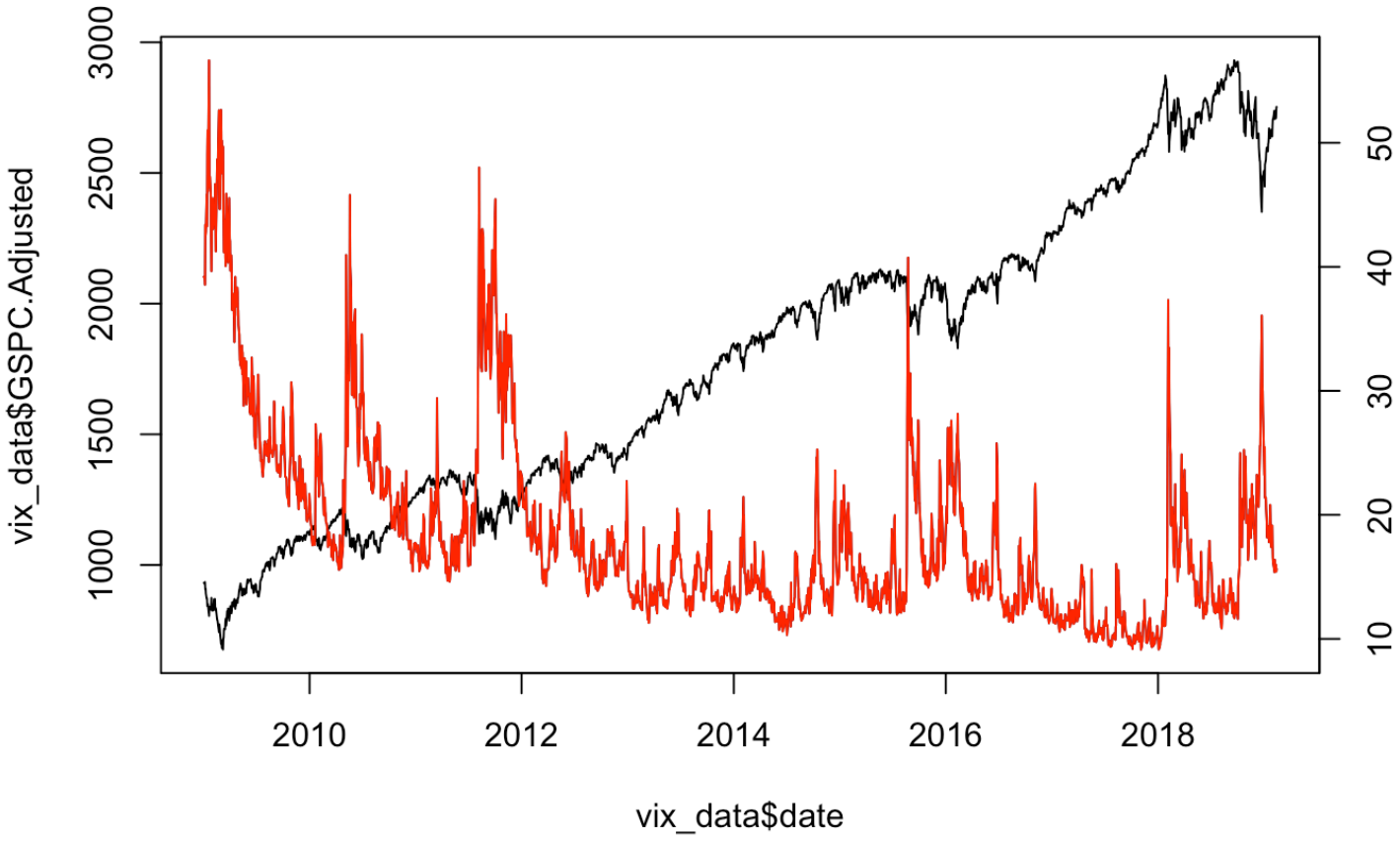
```
getSymbols(c("^GSPC","^VIX"),from = as.Date("2009-01-01"), to = as.Date("2019-02-1
4"))
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

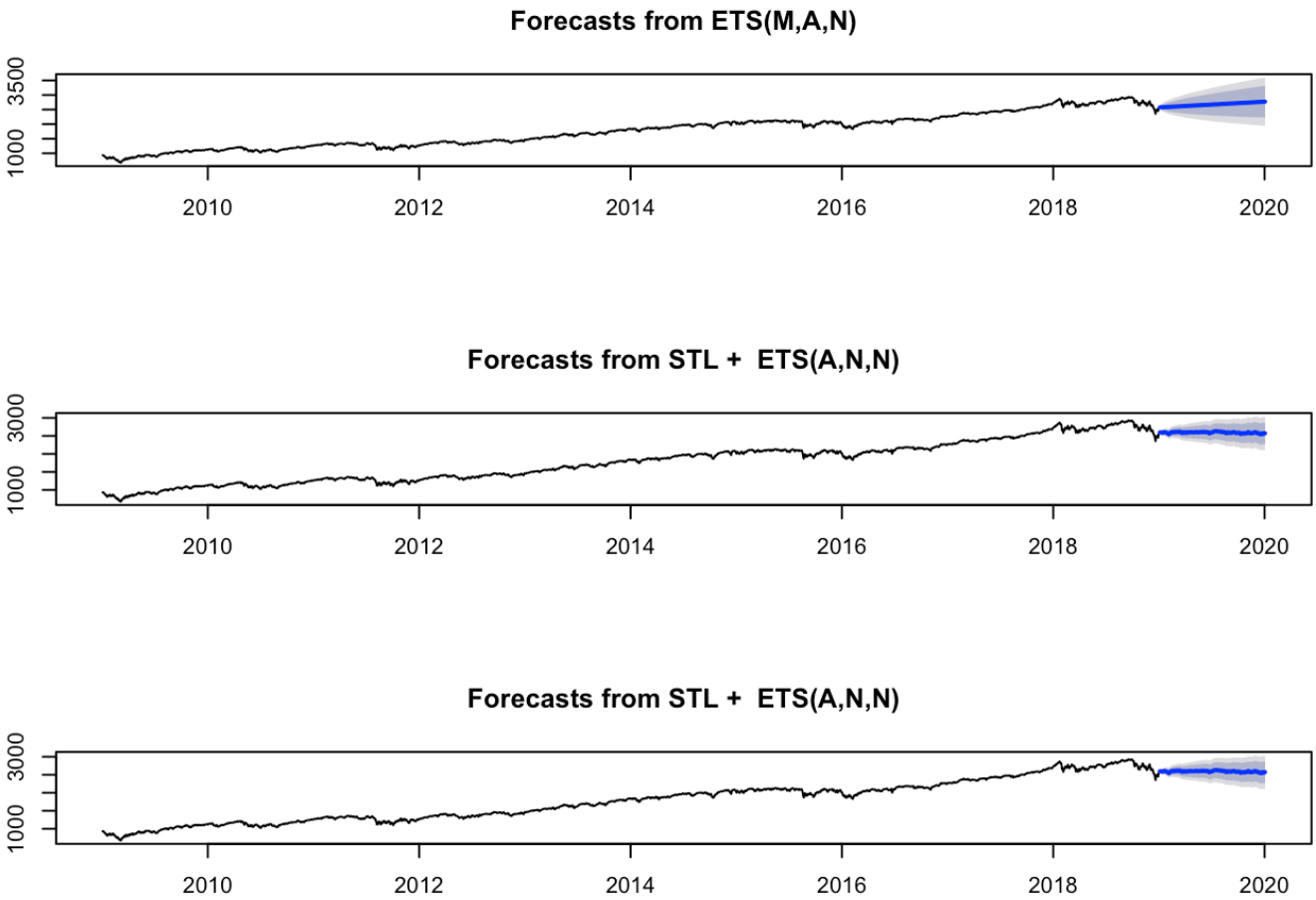
```
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

```
## [1] "GSPC" "VIX"
```

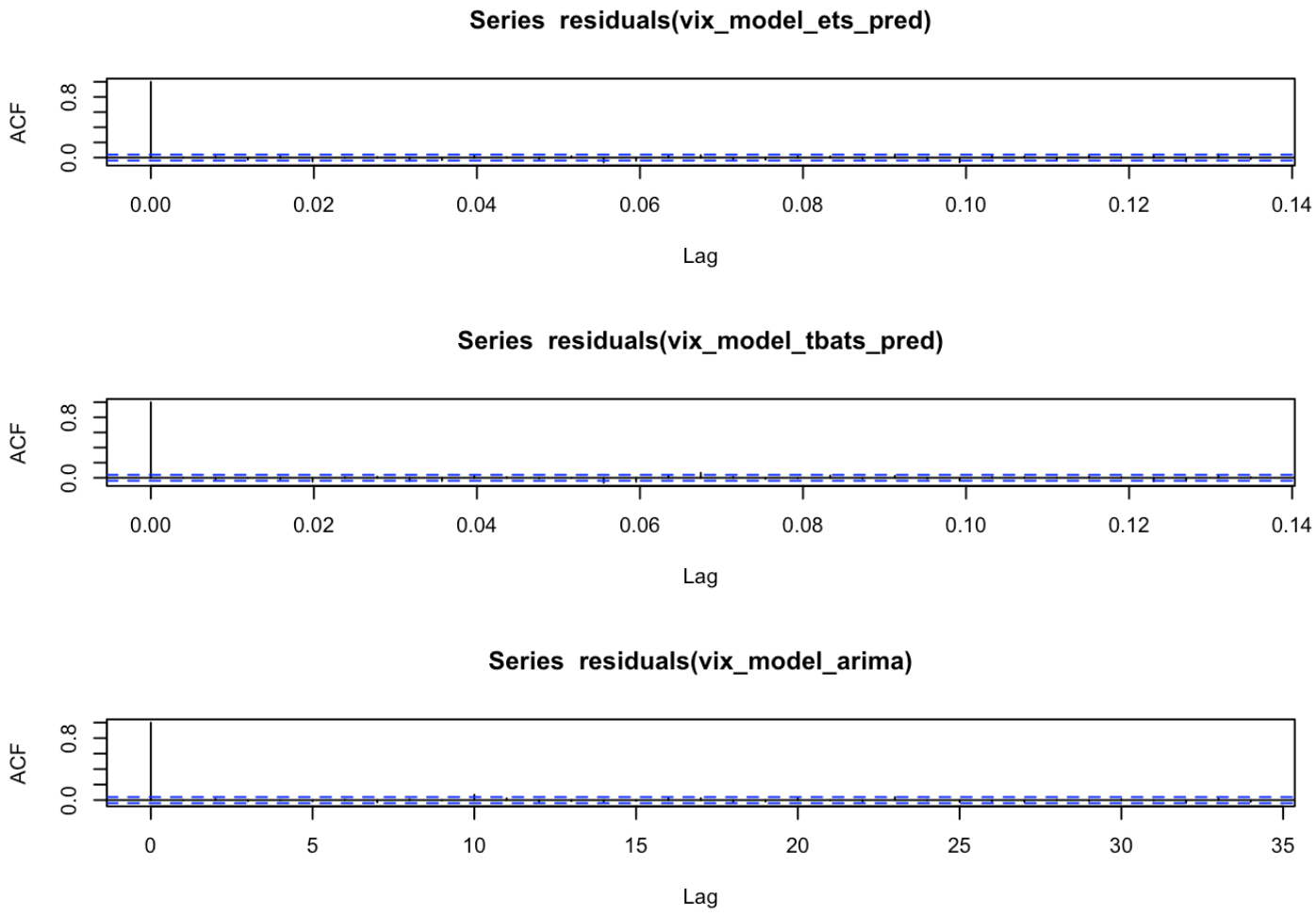
```
vix_data = as.data.frame(merge(GSPC,VIX))
vix_data <- data.frame(date = row.names(vix_data), vix_data, row.names = NULL)
vix_data$date <- as.Date(vix_data$date, "%Y-%m-%d")

vix_data_ts <- ts(vix_data$GSPC.Adjusted, start= c(2009, 02, 01), end=c(2019,2,13)
, frequency=252)

plot (vix_data$date,vix_data$GSPC.Adjusted, type = "l")

par(new = TRUE)
plot (vix_data$date,vix_data$VIX.Adjusted,type = "l", axes = FALSE, bty = "n", xla
b = "", ylab = "")
lines(vix_data$date,vix_data$VIX.Adjusted,col="red")
axis(side=4, at = pretty(range(vix_data$VIX.Adjusted)))
```




```
vix_model_ets <- ets(vix_data_ts, model = "MAN")
vix_model_ets_pred <- forecast(vix_model_ets, h=252,level=c(0.8, 0.95) )

vix_model <- tbats(vix_data_ts)
vix_model_tbats_pred <-forecast(vix_data_ts, h=252, level=c(0.8, 0.95))

vix_model_arima <- auto.arima(vix_data$VIX.Adjusted,seasonal=TRUE)
vix_model_arima_pred <-forecast(vix_data_ts, h=252, level=c(0.8, 0.95))

par(mfrow=c(3,1))

plot(vix_model_ets_pred)
plot(vix_model_tbats_pred)
plot(vix_model_arima_pred)
```




```
#automatically fits the ARIMA model (auto-regressive integrated moving average)
acf(residuals(vix_model_ets_pred))
acf(residuals(vix_model_tbats_pred))
acf(residuals(vix_model_arima))
```



Part 3. Forecasting S&P500 movement using Hidden Markov Model

Markov models System state is fully observable System state is partially observable System is autonomous Markov chain Hidden Markov model System is controlled Markov decision process Partially observable Markov decision process

```
##  [1] 30.279540 12.479094 11.631622 10.983566 10.505473 10.041342  9.595356
##  [8]  9.252240  8.903898  8.664750  8.518157
```

```
## [1] 0.2444656 0.2737506 0.2502788 0.2434303 0.2809230 0.2389925
```

```
## [1] 0.2271080 0.2461340 0.2290556 0.2375244 0.2520413 0.2273132
```

```
## iteration = 0
## Step:
## [1] 0 0 0 0 0 0
## Parameter:
## [1]   0.000600  -4.605170   0.000700  -3.912023  -4.553877 -13.774689
```

```
## Function Value
## [1] -55572.54
## Gradient:
## [1]  -778.16103   911.94801    39.34420 -1098.38325    12.95721   -26.51167
##
## iteration = 1
## Step:
## [1]  0.0525225928 -0.0615526505 -0.0026555680  0.0741362438 -0.0008745569
## [6]  0.0017894258
## Parameter:
## [1]   0.053122593  -4.666722836  -0.001955568  -3.837886762  -4.554751449
## [6] -13.772899138
## Function Value
## [1] -55676.55
## Gradient:
## [1] -263.187743 -245.449756   38.158527 -382.931311    9.850909  -30.991454
##
## iteration = 2
## Step:
## [1]  0.030117525  0.016777297 -0.003859146  0.043586633 -0.001015607
## [6]  0.003096717
## Parameter:
## [1]   0.083240117  -4.649945539  -0.005814714  -3.794300129  -4.555767055
## [6] -13.769802420
## Function Value
## [1] -55693.98
## Gradient:
## [1]  101.835358  -94.720358   36.755242 -172.375762    8.126384  -30.024710
##
## iteration = 3
## Step:
## [1] -0.003125684  0.010148646 -0.004506163  0.027917005 -0.001035427
## [6]  0.003656700
## Parameter:
## [1]   0.08011443  -4.63979689  -0.01032088  -3.76638312  -4.55680248
## [6] -13.76614572
## Function Value
## [1] -55698.39
## Gradient:
## [1]  66.843335 -29.785098  33.700613 -63.067473   7.116802 -29.297336
##
## iteration = 4
## Step:
## [1] -0.005284599  0.006038661 -0.005497836  0.017745730 -0.001197956
## [6]  0.004709980
## Parameter:
## [1]   0.07482983  -4.63375823  -0.01581871  -3.74863739  -4.55800044
## [6] -13.76143574
## Function Value
## [1] -55699.56
```

```
## Gradient:
## [1]   4.550391   5.390514  30.760741  -5.968085   6.104799 -29.154706
##
## iteration = 5
## Step:
## [1] -0.0009616589  0.0011371965 -0.0042101807  0.0052092220 -0.0008621060
## [6]  0.0038772757
## Parameter:
## [1]   0.07386818  -4.63262104  -0.02002889  -3.74342817  -4.55886254
## [6] -13.75755846
## Function Value
## [1] -55699.79
## Gradient:
## [1]  -7.136419   3.179874  28.697512  10.041162   5.705977 -29.211061
##
## iteration = 6
## Step:
## [1] -0.001025195  0.002309657 -0.013040068  0.008875144 -0.002649299
## [6]  0.012535500
## Parameter:
## [1]   0.07284298  -4.63031138  -0.03306896  -3.73455303  -4.56151184
## [6] -13.74502297
## Function Value
## [1] -55700.29
## Gradient:
## [1] -20.468819   2.477624  22.665568  31.917748   4.877851 -29.347847
##
## iteration = 7
## Step:
## [1] -0.0004119845  0.0028596590 -0.0252215255  0.0095976663 -0.0051672887
## [6]  0.0253749771
## Parameter:
## [1]   0.07243100  -4.62745172  -0.05829049  -3.72495536  -4.56667913
## [6] -13.71964799
## Function Value
## [1] -55701.08
## Gradient:
## [1] -27.904229   3.179065  11.655706  49.065902   3.806857 -29.521414
##
## iteration = 8
## Step:
## [1]  0.0005856842  0.0026576539 -0.0407499322  0.0071995198 -0.0085451017
## [6]  0.0433554642
## Parameter:
## [1]   0.07301668  -4.62479407  -0.09904042  -3.71775584  -4.57522423
## [6] -13.67629252
## Function Value
## [1] -55702.13
## Gradient:
## [1] -24.736328   9.345019  -5.653157  56.221656   2.801806 -29.639528
```

```
##
## iteration = 9
## Step:
## [1]  0.0012218762 -0.0002577695 -0.0389276242 -0.0022183263 -0.0086214773
## [6]  0.0454745938
## Parameter:
## [1]   0.07423856  -4.62505184  -0.13796804  -3.71997417  -4.58384571
## [6] -13.63081793
## Function Value
## [1] -55703.07
## Gradient:
## [1] -13.624289   9.122530 -23.383240  43.689313   2.507053 -29.698756
##
## iteration = 10
## Step:
## [1]  0.001223976 -0.002956941 -0.024136915 -0.010839336 -0.006276355
## [6]  0.035207194
## Parameter:
## [1]   0.07546253  -4.62800878  -0.16210496  -3.73081350  -4.59012207
## [6] -13.59561074
## Function Value
## [1] -55703.75
## Gradient:
## [1]  -1.699256   4.494334 -37.265876  14.755346   2.983401 -29.675274
##
## iteration = 11
## Step:
## [1]  0.0006610104 -0.0031947334 -0.0003959804 -0.0107741074 -0.0018182621
## [6]  0.0125632173
## Parameter:
## [1]   0.07612354  -4.63120351  -0.16250094  -3.74158761  -4.59194033
## [6] -13.58304752
## Function Value
## [1] -55704.16
## Gradient:
## [1]   5.7375364   0.9983633 -40.8597552  -9.2406037   3.7555711 -29.5807460
##
## iteration = 12
## Step:
## [1]  0.0004651716 -0.0038475602  0.0112853813 -0.0119021805 -0.0010578717
## [6]  0.0103319153
## Parameter:
## [1]   0.07658871  -4.63505107  -0.15121556  -3.75348979  -4.59299820
## [6] -13.57271560
## Function Value
## [1] -55704.63
## Gradient:
## [1]  12.139477  -9.014914 -38.995866 -37.713010   4.574477 -29.544780
##
## iteration = 13
```

```
## Step:
## [1]  0.0002523756 -0.0024087930  0.0129787364 -0.0074694877 -0.0029442748
## [6]  0.0218934037
## Parameter:
## [1]   0.07684109  -4.63745986  -0.13823682  -3.76095928  -4.59594247
## [6] -13.55082220
## Function Value
## [1] -55705.38
## Gradient:
## [1]  16.169608 -15.109048 -34.383003 -58.510778   4.890339 -29.619939
##
## iteration = 14
## Step:
## [1]  0.0008430834 -0.0065833899  0.0422417602 -0.0198080395 -0.0227878599
## [6]  0.1534287907
## Parameter:
## [1]   0.07768417  -4.64404325  -0.09599506  -3.78076732  -4.61873033
## [6] -13.39739341
## Function Value
## [1] -55709.21
## Gradient:
## [1]   28.098540  -27.808183  -15.517900 -123.271247    4.751881  -30.235218
##
## iteration = 15
## Step:
## [1]  0.001862897 -0.010223268  0.060983909 -0.026840067 -0.074417727
## [6]  0.482346431
## Parameter:
## [1]   0.07954707  -4.65426652  -0.03501115  -3.80760739  -4.69314806
## [6] -12.91504698
## Function Value
## [1] -55719.17
## Gradient:
## [1]   48.965230  -61.616489   18.884588 -232.209868    2.350102  -31.718766
##
## iteration = 16
## Step:
## [1]  0.005287797 -0.019178611  0.105558041 -0.044386500 -0.262226257
## [6]  1.675988119
## Parameter:
## [1]   0.08483487  -4.67344513   0.07054689  -3.85199389  -4.95537432
## [6] -11.23905886
## Function Value
## [1] -55751.54
## Gradient:
## [1]  111.455331 -129.090477   96.740412 -462.428799   -7.177335  -35.766259
##
## iteration = 17
## Step:
## [1]  0.01426805 -0.04515184  0.20173510 -0.08729830 -0.95583357  6.07524655
```

```
## Parameter:
## [1]  0.09910292 -4.71859697  0.27228199 -3.93929218 -5.91120789 -5.16381231
## Function Value
## [1] -55839.98
## Gradient:
## [1]   234.3296801   168.9355054   349.9575687 -1091.7350212   -53.4760028
## [6]     0.5532141
##
## iteration = 18
## Step:
## [1] -0.007881271  0.024558396 -0.192133450  0.086622211  0.428163023
## [6] -2.738474642
## Parameter:
## [1]  0.09122165 -4.69403857  0.08014854 -3.85266997 -5.48304486 -7.90228695
## Function Value
## [1] -55871.66
## Gradient:
## [1]  149.33566 -152.20860  113.12008 -432.79639  -31.55550  -45.11462
##
## iteration = 19
## Step:
## [1]  0.000543335 -0.009662008 -0.097981141  0.030460108 -0.297973058
## [6]  1.876537559
## Parameter:
## [1]  0.09176498 -4.70370058 -0.01783260 -3.82220986 -5.78101792 -6.02574939
## Function Value
## [1] -55954.7
## Gradient:
## [1]  105.78369 -183.24033   42.22114 -213.43895  -49.65745  -32.56562
##
## iteration = 20
## Step:
## [1] -0.0004600018 -0.0001549423 -0.0097088808  0.0053995274 -0.0548754612
## [6]  0.3523567625
## Parameter:
## [1]  0.09130498 -4.70385552 -0.02754148 -3.81681034 -5.83589338 -5.67339263
## Function Value
## [1] -55963.17
## Gradient:
## [1]   93.12078 -162.97157   36.04199 -182.81925  -53.12808  -22.43968
##
## iteration = 21
## Step:
## [1] -0.006024945  0.014153003 -0.028394603  0.033738172 -0.064678683
## [6]  0.449326133
## Parameter:
## [1]  0.08528003 -4.68970252 -0.05593609 -3.78307216 -5.90057207 -5.22406650
## Function Value
## [1] -55971.34
## Gradient:
```

```
## [1]  20.1885850 -24.5304969  18.4323799 -54.2154091 -57.1384707   0.4609492
##
## iteration = 22
## Step:
## [1] -0.002825754  0.008337191 -0.014931243  0.014960598  0.035737644
## [6] -0.195860733
## Parameter:
## [1]  0.08245428 -4.68136533 -0.07086733 -3.76811157 -5.86483442 -5.41992723
## Function Value
## [1] -55973.28
## Gradient:
## [1]  -4.407251  11.943091   9.696712 -15.333768 -54.891494  -8.874495
##
## iteration = 23
## Step:
## [1] -0.0002781127  0.0015811006 -0.0137132526  0.0058305746  0.0072038174
## [6] -0.0062660115
## Parameter:
## [1]  0.08217617 -4.67978423 -0.08458058 -3.76228099 -5.85763060 -5.42619324
## Function Value
## [1] -55973.71
## Gradient:
## [1]  -7.413008  11.076636   2.163528   4.944646 -54.886401  -9.033710
##
## iteration = 24
## Step:
## [1]  0.000259056  0.001316650 -0.022223864  0.007595994  0.019212552
## [6] -0.002642175
## Parameter:
## [1]  0.08243522 -4.67846758 -0.10680445 -3.75468500 -5.83841805 -5.42883542
## Function Value
## [1] -55974.51
## Gradient:
## [1]  -5.120375   0.328205  -9.561234  32.488380 -55.003120  -9.195545
##
## iteration = 25
## Step:
## [1]  0.0005846324  0.0013280387 -0.0289475954  0.0091954409  0.0444221699
## [6]  0.0029799853
## Parameter:
## [1]  0.08301986 -4.67713954 -0.13575204 -3.74548956 -5.79399588 -5.42585543
## Function Value
## [1] -55976.06
## Gradient:
## [1]   0.2186571 -14.1279838 -23.9553556  64.0883383 -55.1694654  -9.3178562
##
## iteration = 26
## Step:
## [1]  0.001149906  0.002078663 -0.051893197  0.016310559  0.132613233
## [6]  0.016527732
```

```
## Parameter:
## [1]  0.08416976 -4.67506088 -0.18764524 -3.72917900 -5.66138265 -5.40932770
## Function Value
## [1] -55980.26
## Gradient:
## [1]  10.367185 -37.841069 -47.389498 115.072887 -55.332488  -9.453464
##
## iteration = 27
## Step:
## [1]  0.001842878  0.002811058 -0.084658625  0.026512599  0.343671411
## [6]  0.051223601
## Parameter:
## [1]  0.08601264 -4.67224982 -0.27230386 -3.70266640 -5.31771124 -5.35810410
## Function Value
## [1] -55990.35
## Gradient:
## [1]  25.590409 -70.759570 -79.876962 186.711800 -54.672079  -9.610051
##
## iteration = 28
## Step:
## [1]  0.002660281  0.002875661 -0.133266112  0.040779541  0.827019861
## [6]  0.141358835
## Parameter:
## [1]  0.08867292 -4.66937416 -0.40556998 -3.66188686 -4.49069138 -5.21674526
## Function Value
## [1] -56011.95
## Gradient:
## [1]   45.52194 -111.69014 -118.88063  266.13473  -47.23011  -11.07652
##
## iteration = 29
## Step:
## [1]  0.0014944408 -0.0009829873 -0.0970534043  0.0296184474  1.1238489786
## [6]  0.3007350976
## Parameter:
## [1]  0.09016736 -4.67035715 -0.50262338 -3.63226841 -3.36684240 -4.91601017
## Function Value
## [1] -56032.71
## Gradient:
## [1]   39.46857 -109.96176 -134.90730  267.69959  -12.63824  -16.61332
##
## iteration = 30
## Step:
## [1] -0.0008227994 -0.0079656291  0.0951237842 -0.0272777056  0.0334060159
## [6]  0.2833830230
## Parameter:
## [1]  0.08934456 -4.67832278 -0.40749960 -3.65954611 -3.33343638 -4.63262714
## Function Value
## [1] -56053.62
## Gradient:
## [1]   15.399855  -31.473402 -113.135808  232.151208  -16.330861    1.176296
```

```
##
## iteration = 31
## Step:
## [1]  0.0004746136 -0.0442817179  0.4504829768 -0.1360354750  0.1812401123
## [6]  0.3699559891
## Parameter:
## [1]  0.08981918 -4.72260449  0.04298338 -3.79558159 -3.15219627 -4.26267115
## Function Value
## [1] -56075.79
## Gradient:
## [1]   24.720401   80.715848   93.082890 -173.717453    4.697227   10.613212
##
## iteration = 32
## Step:
## [1] -0.001689811  0.018397251 -0.205096461  0.060261771 -0.005245058
## [6] -0.208127108
## Parameter:
## [1]  0.08812937 -4.70420724 -0.16211308 -3.73531982 -3.15744133 -4.47079826
## Function Value
## [1] -56084.53
## Gradient:
## [1] -11.267650  -4.878811 -27.507034  59.587776   2.638315  -2.111155
##
## iteration = 33
## Step:
## [1]  0.0004060325 -0.0048960553  0.0510326195 -0.0176331401 -0.1140739994
## [6] -0.0032455133
## Parameter:
## [1]  0.0885354 -4.7091033 -0.1110805 -3.7529530 -3.2715153 -4.4740438
## Function Value
## [1] -56085.95
## Gradient:
## [1] -3.082590  2.176993 -3.355766 18.586650 -4.065379  0.245871
##
## iteration = 34
## Step:
## [1]  0.0005293077 -0.0033783932  0.0139924182 -0.0068552035  0.1006831864
## [6]  0.0523529332
## Parameter:
## [1]  0.08906471 -4.71248169 -0.09708804 -3.75980816 -3.17083214 -4.42169084
## Function Value
## [1] -56086
## Gradient:
## [1]   0.01141598   6.68986179   4.59686271 -15.64858315   3.67320385
## [6]  -0.52805227
##
## iteration = 35
## Step:
## [1] -0.0001032697  0.0011508441 -0.0076522041  0.0027958639 -0.0459055771
## [6] -0.0203597807
```

```
## Parameter:
## [1]  0.08896144 -4.71133085 -0.10474025 -3.75701230 -3.21673772 -4.44205062
## Function Value
## [1] -56086.11
## Gradient:
## [1] -0.46797504  1.96858823  0.15785918  0.26976730 -0.08943802  0.04147347
##
## iteration = 36
## Step:
## [1]  4.582253e-05 -1.729324e-04 -1.496214e-04 -1.586060e-04  1.191385e-03
## [6]  7.504390e-04
## Parameter:
## [1]  0.08900726 -4.71150378 -0.10488987 -3.75717090 -3.21554634 -4.44130018
## Function Value
## [1] -56086.11
## Gradient:
## [1] -1.275985e-01  5.557459e-01  5.839684e-02 -8.133519e-05  4.095566e-03
## [6] -7.055940e-03
##
## iteration = 37
## Step:
## [1]  1.774469e-05 -6.482211e-05 -8.291203e-05 -5.382548e-05  1.002612e-04
## [6]  3.848992e-04
## Parameter:
## [1]  0.0890250 -4.7115686 -0.1049728 -3.7572247 -3.2154461 -4.4409153
## Function Value
## [1] -56086.11
## Gradient:
## [1]  0.0031141099  0.0315263742  0.0047293724 -0.0153217816  0.0027606335
## [6]  0.0004636648
##
## iteration = 38
## Parameter:
## [1]  0.08902499 -4.71157013 -0.10498025 -3.75722400 -3.21548035 -4.44093321
## Function Value
## [1] -56086.11
## Gradient:
## [1]  0.0011641532 -0.0003459175  0.0003783498 -0.0009721355  0.0003054767
## [6] -0.0003145699
##
## Relative gradient close to zero.
## Current iterate is probably solution.
```

```
##                  mu      sigma
## [1,]  0.0005841545 0.00899065
## [2,] -0.0006888481 0.02334847
```

```
##            [,1]       [,2]
## [1,] 0.98835233 0.01164767
## [2,] 0.03858731 0.96141269
```

```
## [1] 0.7681363 0.2318637
```

```
##                mean         sd kurtosis
## [1,]  0.0005841545 0.00635735        3
## [2,] -0.0006888481 0.01650986        3
```

```
##                mean          sd  kurtosis    skewness length
## [1,]  0.0006007835 0.006333934  3.565389 -0.02491462  12868
## [2,] -0.0007847466 0.016749016 14.828955 -0.71360850   3737
```

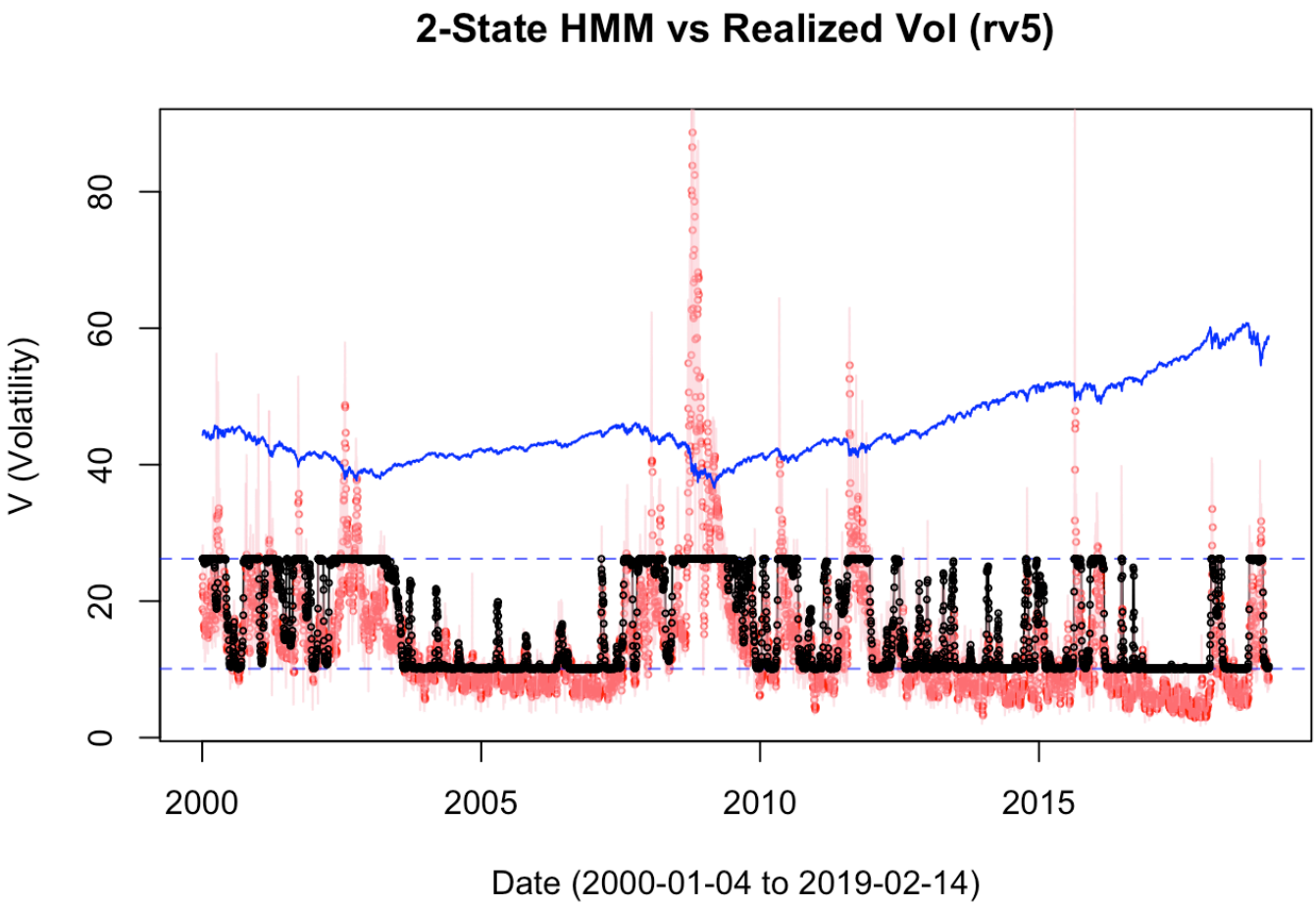
```
##                mean         sd kurtosis
## [1,]  0.0005841545 0.00635735        3
## [2,] -0.0006888481 0.01650986        3
```

```
ldhmm.calc_stats_from_obs(hd, drop=11)
```

```
##                mean         sd kurtosis     skewness length
## [1,]  0.0006027576 0.00629874 3.478128 -0.021274020  12857
## [2,] -0.0006481724 0.01568384 4.275335  0.003376325   3726
```

```
ldhmm.oxford_man_plot_obs(hd, insert.plot = FALSE)
```

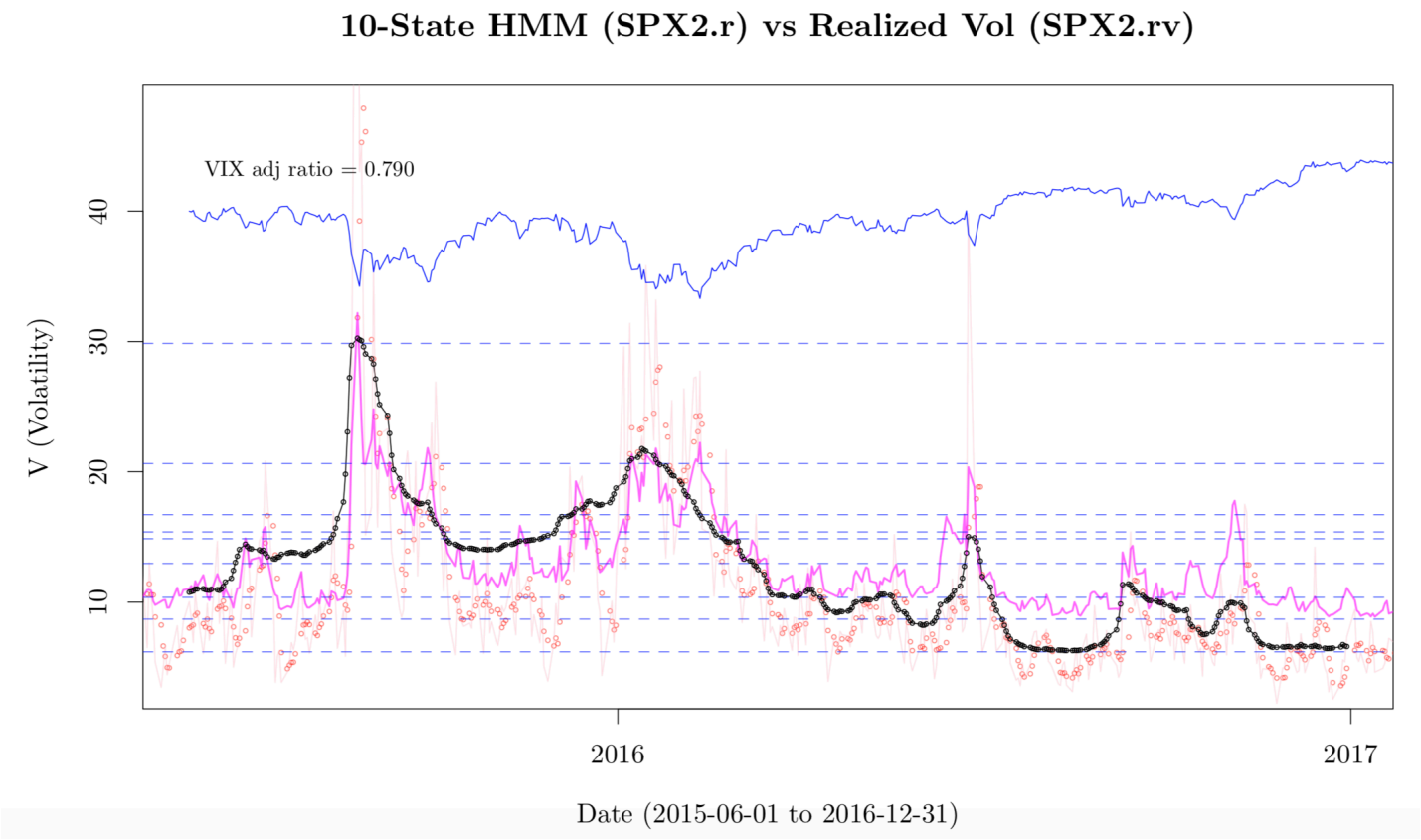
```
## [1] "Max date of VIX data is 2019-02-14"
## [1] "Max date of Oxford data is 2019-02-14"
## [1] "2000-01-04 2019-02-14"
```




```
knitr::include_graphics("Markov_Model_10MM.png")
```

	Countable state space	Continuous or general state space
Discrete-time	(discrete-time) Markov chain on a countable or finite state space	Harris chain (Markov chain on a general state space)
Continuous-time	Continuous-time Markov process or Markov jump process	Any continuous stochastic process with the Markov property, e.g., the Wiener process

```
knitr::include_graphics("Markov_Model_10MMchart.png")
```



Part 4. Next steps

Considering all the factors that impact the movement of stocks the key outcome is that predicting the movement is incredibly complicated task. Stock movements rely on a lot more than simply the factors captured in the data - sentiment analysis, market movements, political events - so our model was unlikely to be able to predict much of the variance.

As of next steps to furthe improve the model the work could be done in the following directions: - Data Data could be enriched with other variables such as sentiment analysis of news and social media, macroeconomics, bonds, geopolitical events etc. - Math and algorithms On the math and algorithms side the main idea to identify the most optimal and accurate algorithm - Computational power Within the computational power the idea is to process as much data as possible at cheapest price