# STOCK MOVEMENT PREDICTION


**Group No**.: Group 21

**Student Names**: Nikilesh Kumar, Suman Kumar

**Northeastern University**
**IE 7275: Data Mining in Engineering**
**College of Engineering**

**Executive Summary:** The goal of this study is to predict movement in stock price based on various ETFs and let the user know if the price on Day N will be higher than Day N-1. A 2-class binary classification problem. Financial data for this project was obtained using the built in **quantmod** functions and sourced from Yahoo Finance. As part of the data processing, we utilized technical performance indicators to convert our data into stationary data and remove seasonality. We utilized techniques such as xtreme gradient boosting and KNN to classify our data with the end goal of achieving as high an AUC score as possible.

## I. Background and Introduction

This is an example of stock prediction with R using ETFs of which the stock is a composite. The premise for this is that, we can think of an ETF as a representative for the entire industry. Banking and financial firms are all pretty much correlated to each other as even a minor policy change could potentially affect all of them. Thus, by using the performance of the ETF to train our Machine Learning models, we can arrive at a healthy and reasonable prediction for target stock**.** To get rid of seasonality in the data, we used technical indicators like RSI, ADX and Parabolic SAR that more or less showed stationarity. The goal of the project is to predict if the stock price today will go higher or lower than yesterday.

The stock we are trying to predict is JP Morgan (NYSE: JPM) and the ETFs we are using are FNCL - Fidelity MSCI Financials Index, IYF - iShares US Financials ETF and XLF - Financial Select Sector SPDR Fund.

## II. Data Exploration and Visualization

We utilized the highcharter R package to render our visualizations of the stock and ETF prices for the last 5 years.


The data obtained from quantmod is in the form of XTS time-series objects with each day having Open, High, Low, Close and Volume. This is a candlestick chart with OHLC data.

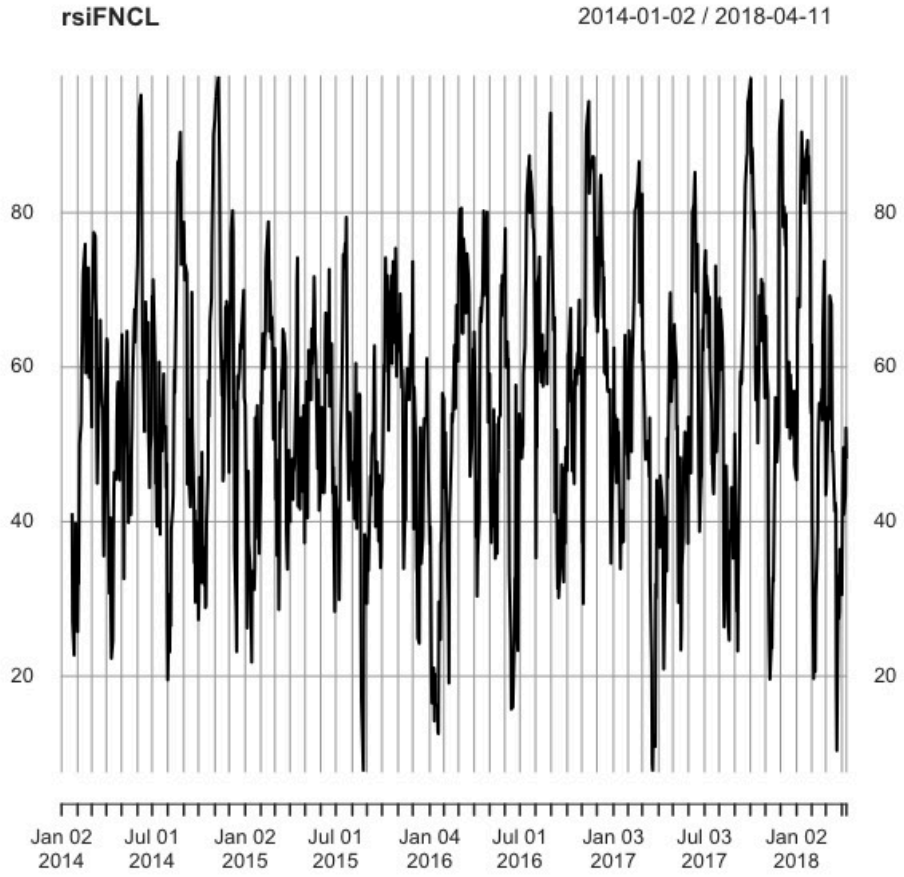
## III. Data Preparation and Preprocessing

One common mistake in using time series data is that the data tends to exhibit seasonality and to arrive at an accurate measure, we need to convert it into a stationary data

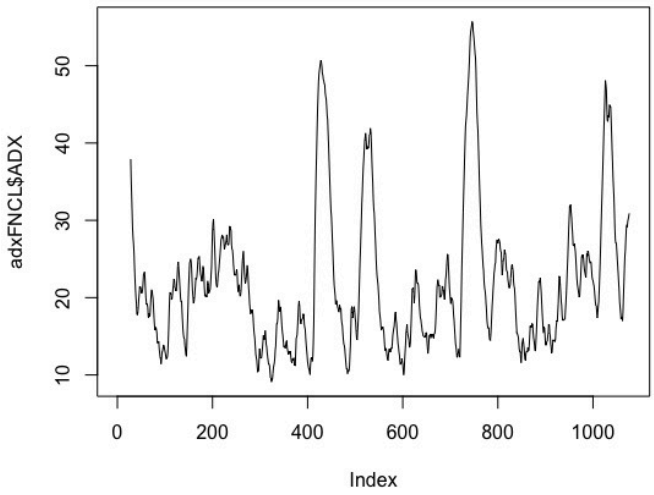
One way we can go about doing this is differencing the data. But since this is financial data, the **quantmod** package has a lot of technical indicator functions which we can use to generate indicator data that more or less gets rid of seasonality.

Some of the indicators, we have used in our model are:

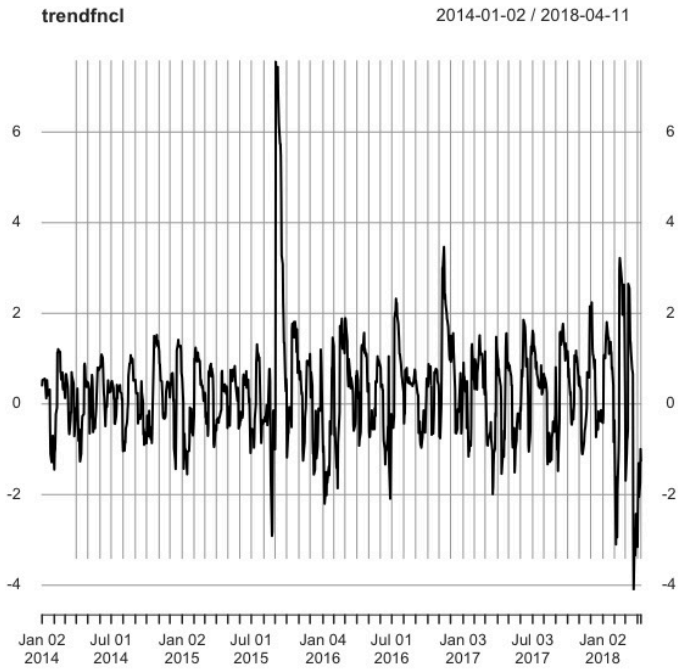
RSI - Relative Stregth Index (A measure of how the stock performed scaled to 0-100 w.r.t the Weighted Moving Average)




ADX - Average Directional Indicator




Parabolic SAR Trend- Stop and Reverse Indicator

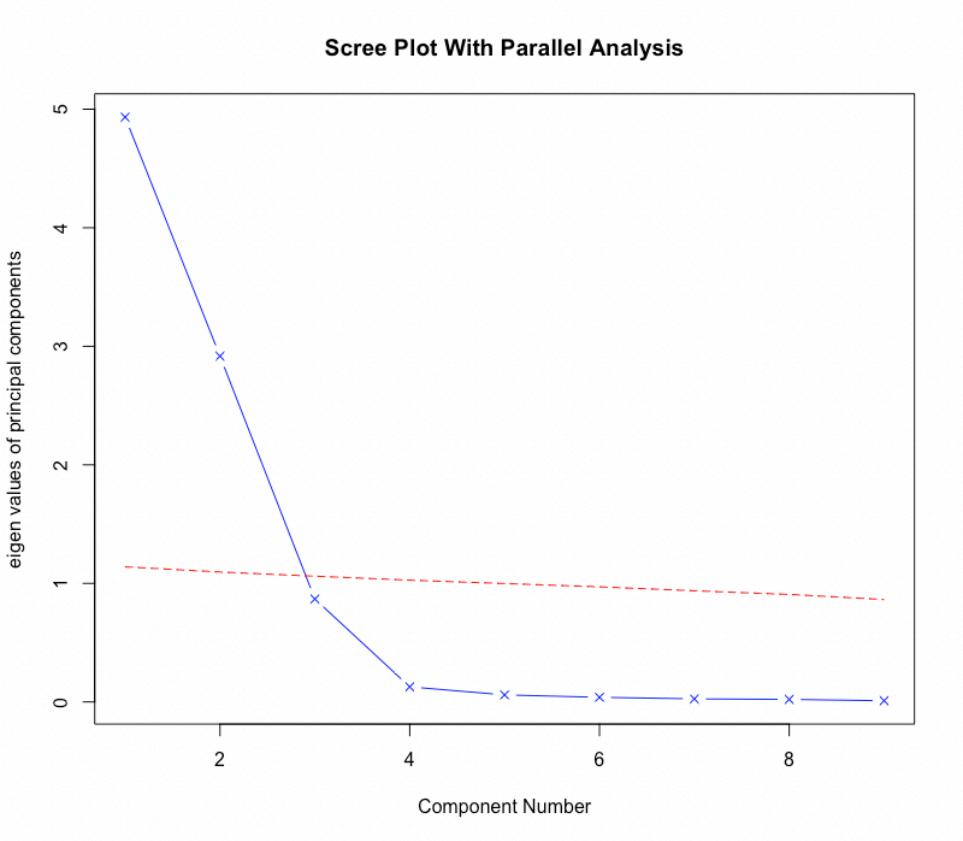



After munging out all the numbers for the indicators, we then feed it into our model. We also incorporate a lag of 1 day to avoid a look-ahead bias on the data.

## IV. Data Mining Techniques and Implementation

We will be using the xgboost algorithm with the goal of binary logistic regression. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

As part of preliminary analysis, we ran PCA on our predictor variables.

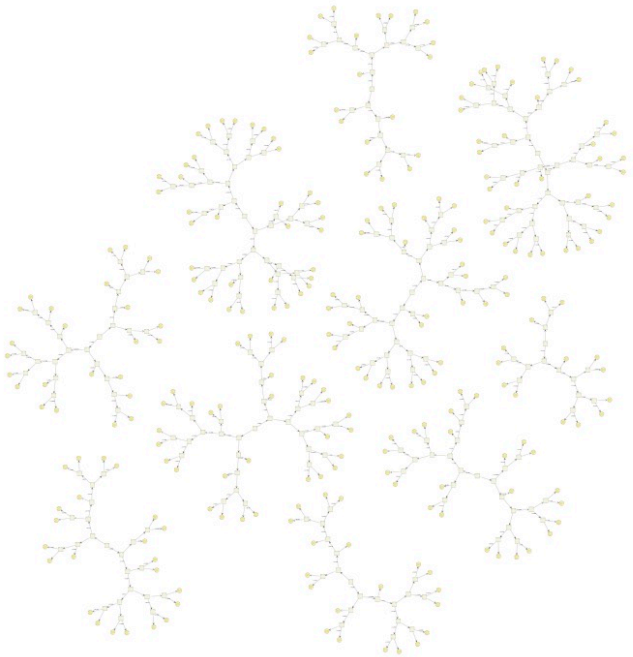



Parallel analysis suggests that 2 components suffice.

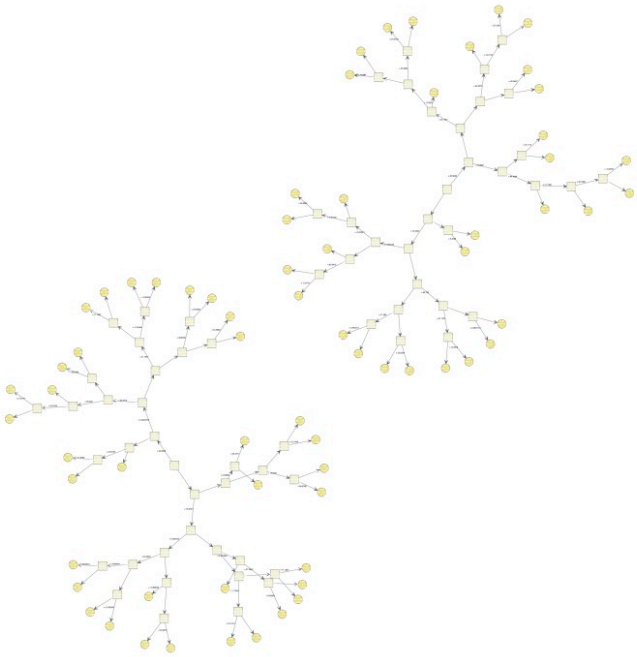
After data preparation into training (approx 70% )and test (approx 30%) sets, we then feed it to the algorithm.

We are used 10 rounds of trees in our model.

Using the **diagrammeR** R package we can visualize the tree structure.



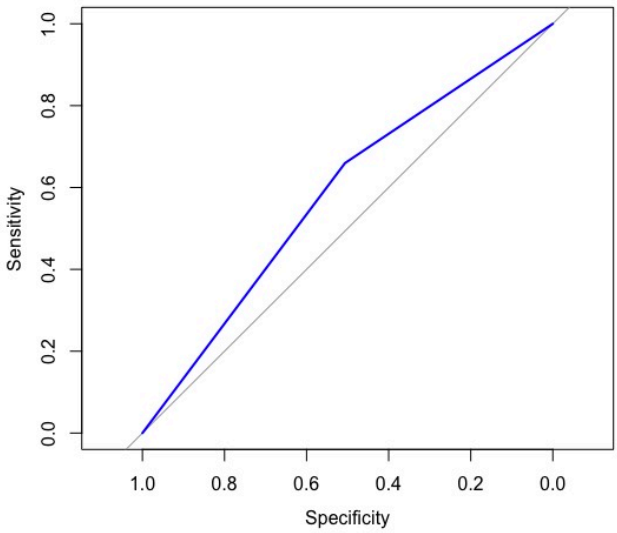
On zooming into one round, we get,



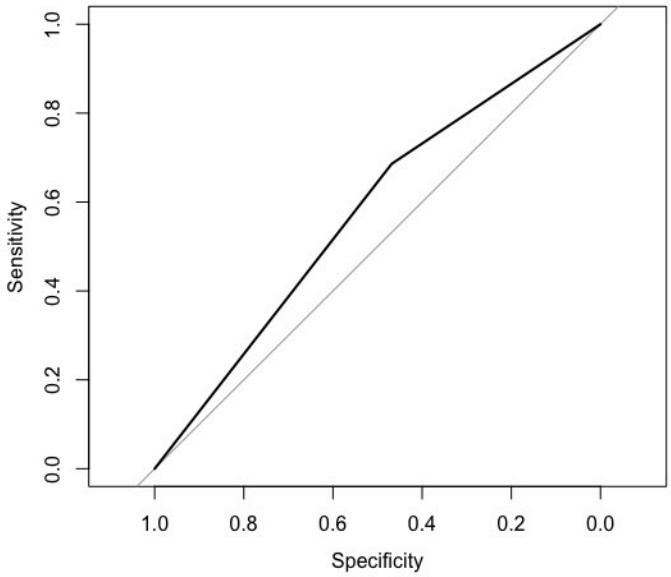
Additionally, we also ran KNN classification for 8 nearest neighbors and received similar AUC results.

## V. Performance Evaluation

For our first model based on xgboost, we achieved an AUC score of:
0.591939755047997




For our second model based on KNN classification, we achieved an AUC Score of:
0.5728



## VI. Discussion and Recommendation

Based on overall results, there clearly is room to improve the model through model tweaking and tuning. We could also have evaluated the model performance through other accuracy measures.

There is scope to build on this project and predict a price band with CI instead of a simple up or down signal.

## VII. Summary

Using the data gathered from various ETFs, we successfully built a machine learning model that was able to correlate the ETFs performance with the performance of one composite and predict if the composite was going to exhibit a change in value the next day. By running and validating against two Machine Learning models (KNN and XGBoost) we give further support to our study.

## VII. Acknowledgements


- R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.
- R Packages used : xgboost, quantmod, highcharter, psych, pROC


Github Repo Directory: https://github.com/niki864/Stock_Prediction_With_R

# Appendix: R Code for use case study

```
library(xgboost)
library(quantmod)
library(TTR)
stocksym <-c("XLF","JPM","IYF","FNCL")
#Get stock data using getSymbols call
getSymbols(stocksym)
#Dataframe for JP Morgan stock for last 5 years
JPM<-last(JPM,"5 years")
#Similarly for XLF, IYF and FNCL
XLF<-last(XLF,"5 years")
IYF<-last(IYF,"5 years")
FNCL<-last(FNCL,"5 years")
#Lets Visualise the data
library(highcharter)
#Highcharter charts are look really good on the eyes
highchart(type = "stock") %>% hc_add_series(JPM,type="ohlc") %>%
  hc_add_series(XLF,type="ohlc") %>% hc_add_series(FNCL, type="ohlc") %>%
  hc_add_series(IYF, type="ohlc") %>% hc_legend(enabled=TRUE) %>%
hc_title(text="OHLC Prices For The Last 5 Years")
#If you would like candlecharts, susbstitute type="line" with type="ohlc"
#Its not a good idea to build a model based on the prices themselves.
#Time series data tends to be correlated in time and in turn exhibits
significant autocorrelation
#Learnt that from this article here https://www.linkedin.com/pulse/how-use-
machine-learning-time-series-forecasting-vegard-flovik-phd/
#Bottom line, we need to use stationary data to build a model -> Feature
Engineering
#Let's define a few technical indicators to build a model
#Relative Strength Indicator
rsiFNCL <-RSI(FNCL$FNCL.Close, n=14, maType = "WMA")
plot(rsiFNCL, type = 'l')
rsiIYF <-RSI(IYF$IYF.Close, n=14, maType = "WMA")
plot(rsiIYF, type = 'l')
rsiXLF<-RSI(XLF$XLF.Close, n=14, maType = "WMA")
plot(rsiXLF, type = 'l')
#Average Directional Indicator
adxFNCL <- data.frame(ADX(FNCL[,c("FNCL.High","FNCL.Low","FNCL.Close")]))
plot(adxFNCL$ADX,type = 'l')
adxIYF <- data.frame(ADX(IYF[,c("IYF.High","IYF.Low","IYF.Close")]))
plot(adxIYF$ADX,type = 'l')
```

```
adxXLF <- data.frame(ADX(XLF[,c("XLF.High","XLF.Low","XLF.Close")]))
plot(adxXLF$ADX,type = 'l')
#Parabolic Stop and Reversal Indicator
sarFNCL <- SAR(FNCL[,c("FNCL.High","FNCL.Low")], accel = c(0.02, 0.2))
trendfncl=FNCL$FNCL.Close-sarFNCL
plot(trendfncl)
sarIYF <- SAR(IYF[,c("IYF.High","IYF.Low")], accel = c(0.02, 0.2))
trendiyf=IYF$IYF.Close-sarIYF
plot(trendiyf)
sarXLF <- SAR(XLF[,c("XLF.High","XLF.Low")], accel = c(0.02, 0.2))
trendxlf=XLF$XLF.Close-sarXLF
plot(trendxlf)
#Induce a lag to avoid look ahead bias
rsiFNCL=c(NA,head(rsiFNCL,-1))
rsiIYF=c(NA,head(rsiIYF,-1))
rsiXLF=c(NA,head(rsiXLF,-1))
adxFNCL$ADX=c(NA,head(adxFNCL$ADX,-1))
adxIYF$ADX=c(NA,head(adxIYF$ADX,-1))
adxXLF$ADX=c(NA,head(adxXLF$ADX,-1))
trendfncl=c(NA,head(trendfncl,-1))
trendiyf=c(NA,head(trendiyf,-1))
trendxlf=c(NA,head(trendxlf,-1))
#Objective is to predict up or down on the daily price
#A clear binary classification problem
#Create the response variable
price=JPM$JPM.Close-JPM$JPM.Open
price=c(NA,head(price,-1))
class=ifelse(price > 0,1,0)
#Combine input features into a matrix
model_df =
data.frame(class,rsiFNCL,rsiIYF,rsiXLF,adxFNCL$ADX,adxIYF$ADX,adxXLF$ADX,tre
ndfncl,trendiyf,trendxlf)
model=data.matrix(model_df)
library(psych)
fa.parallel(model[,-1], fa ="pc", n.iter = 100, show.legend = FALSE, main =
"Scree Plot With Parallel Analysis")
princicomp <- principal(model[,-1], nfactors = 2)
princicomp
model=na.omit(model)
colnames(model)=c("class","rsiFNCL","rsiIYF","rsiXLF","adxFNCL","adxIYF","ad
xXLF","trendfncl","trendiyf","trendxlf")
#Data prep for training and test
train_sz=2/3
```

```
breakpt = nrow(model)*train_sz
traindata=model[1:breakpt,]
testdata=model[(breakpt+1):nrow(model),]
#Break train data into X and Y- Response Variable
X_train=traindata[,2:10]
Y_train=traindata[,1]
class(X_train)[1]
class(Y_train)
#Do the same for test data
X_test=testdata[,2:10]
Y_test=testdata[,1]
#Time to start XGBoost
set.seed(3213)
dtrain=xgb.DMatrix(data = X_train, label=Y_train)
xgmodel=xgboost(data=dtrain, nround=10, objective="binary:logistic")
set.seed(27)
model_xgb_pca <-train(outcome ~ .,
                      data=val_train_data,
                      method="xgbTree",
                      preProcess = "pca",
                      trControl = trainControl(method = "repeatedcv", number
= 5, repeats = 10, verboseIter = FALSE))
#Use cross validation
dtrain=xgb.DMatrix(data = X_train, label=Y_train)
cv=xgb.cv(data = dtrain, nround=10, nfold = 5, objective="binary:logistic")
preds=predict(xgmodel,X_test)
prediction=as.numeric(preds>0.543)
#Find AUC for this
library(pROC)
auc<-auc(Y_test,prediction)
print(paste('AUC Score:',auc))
rocg<-roc(Y_test,prediction)
plot(rocg, col="blue")
tablesamp <- table(prediction, Y_test)
tablesamp
library(class)
#Let's run another classification algorithm to test if our data is valid.
#This is a small script I wrote to automate and test for accuracy for up to
#k cases of KNN classification.
"Enter number of cases you wanna test for training data,
testdata, trainresponse and testresponse"
#Testing for accuracy which is TP + TN/ Total No. of Test Cases
getknnerr <-function(n, traindata, testdata, trainresp, testresp) {
```

```
  ncount=0
  err=0
  errcount=0
  tablen=0
  knnsamp <- knn(traindata, testdata, trainresp, k=1)
  tablesamp <- table(knnsamp, testresp)
  print(tablesamp)
  err=(tablesamp[1,1]+tablesamp[2,2])/(nrow(testdata))
  cat("Base error at k=1 :",err)
  for(i in 2:n){
    knnsamp2 <- knn(traindata, testdata, trainresp, k=i)
    tablesamp2 <- table(knnsamp2, testresp)
    errcheck=(tablesamp2[1,1]+tablesamp2[2,2])/(nrow(testdata))
    if(errcheck>err){
      ncount=i
      err=errcheck
      tablen=tablesamp2
      }
  }
  cat("\nfinal accuracy score:",err)
  cat("\nachieved at k=",ncount)
  tablen
  return(ncount)
}
#The function returns the k value with the highest accuracy.
knn<-getknnerr(10, traindata = X_train, testdata = X_test,trainresp =
Y_train,testresp = Y_test)
knnreturn <- knn(X_train, X_test, Y_train, k=knn)
tablesamp <- table(knnreturn, Y_test)
tablesamp
knnreturn <- sapply(knnreturn, as.numeric)
#Plot roc curve
rocg <- roc(Y_test,knnreturn)
plot(rocg)
aucsc <- auc(Y_test,knnreturn)
aucsc
#This library helps us visualise the xgboost trees
library(DiagrammeR)
xgb.plot.tree(model = xgmodel)
# View only the first tree in the XGBoost model
xgb.plot.tree(model = xgmodel, n_first_tree = 1)
```