Machine Learning Engineer Nanodegre

Capstone Project

Shubra Chowdhury
January 25, 2017

I. Definition

Project Overview

Back testing of trading strategy (Financial Industry).

What is Back testing:

Back testing is a term used in oceanography, meteorology and the financial industry to refer to testing a predictive model using existing historic data. Back testing is a kind of retrodiction, and a special type of cross-validation applied to time series data. (Reference: https://en.wikipedia.org/wiki/Backtesting)

Back testing in this project relates to the domain of finance which deals with investment. Back testing in finance is the process of testing a trading strategy on relevant historical data to ensure its viability before the trader risks any actual capital. A trader can simulate the trading of a strategy over an appropriate period of time and analyze the results for the levels of profitability and risk. (Reference http://www.investopedia.com/terms/b/backtesting.asp)

What will I back test:
In this project, I will back test Exxon Mobile, ticker symbol is XOM.
The idea of Back testing and predicting XOM was the result of recent Presidential Election and subsequently on December 13, 2016 then President Elect selected Rex Tillerson as nominee for Secretary of State. Tillerson has served as CEO of Exxon Mobile from 2006 to 2016 and I have been wondering will it have any impact price on XOM, can I do some sort of prediction for XOM price.

Pre-cursor to back test:

At first I have to identify certain commodity stocks, market indicators, Indexes (Oil and Volatility) and Exchange traded Funds (ETF), the prices of which moves in sync with XOM (directional price movement).

Based on industry/sector movement research conducted by analysts at (Morningstar.com, S&P Capital IQ, http://oilprice.com/, https://www.merrilledge.com/) it is seen that movement of Oil stocks have some sort of similarity with Crude Oil, Market Indexes and Volatility.

As a next step, I will download data for market indicator(SPY), crude oil (WTI and BRENT), volatility index (VIX), XOM and perform a technical analysis (visual analysis of price and return movements).
In financial market, not all stocks are traded on a daily basis however SPY is traded on all open market days. Thus, I will consider SPY (market indicator) as my reference for transaction date and for initial comparison purpose rows for ticker(s) will be dropped where there is no match with transaction dates.
For this research project historically trading data has been downloaded from Yahoo(finance.yahoo.com) and Quandl (https://www.quandl.com/)

For technical analysis, I will consider only one column for each ticker symbols, that is called as the "Adjusted Close", basically this is the adjusted closing price for a ticker for that day. Based on the "Adjusted Close" I will calculate the "Daily Return" for all ticker symbol (SPY, WTI, BRENT, VIX, XOM) included in this research project.

The graphical plot of "Adjusted Close" between SPY, WTI, BRENT, VIX, XOM and the graphical plot of "Daily Return" between SPY, WTI, BRENT, VIX, XOM will help me to understand the directional movement of a return between the tickers thus helping to select ticker(s) with similar directional movement for back testing of XOM.

How will I Back Test:

Based on the technical analysis I will select the tickers which has similar directional movement that as of XOM. I will use these tickers to perform back testing/prediction. Back testing will be done using classification and regression models and will look for the best accuracy sores.

Before performing classification and regression I will go through multiple steps to clean and normalize datasets, create multiple calculated returns and then based on dates will eventually create training and testing datasets. Upon successful data preparation and cleansing I will run the models to get best accuracy score. The model(s) with best accuracy score will the model to back test XOM.

Some of available research and reading on Back testing :

- Investopedia http://www.investopedia.com/articles/trading/05/030205.asp
- Fidelity "Strategy Testing Tools" https://www.fidelity.com/
- CFA Digest http://www.cfapubs.org/doi/full/10.2469/dig.v30.n4.773
- The Street
- Quantpedia http://quantpedia.com/Links/Backtesters

Problem Statement

Back testing of Exxon Mobile ticker symbol XOM

For my exploration purpose, I will consider market indicator, volatility and worldwide crude prices. For market indicator, I have considered SPY, this is the granddaddy of ETFs, the oldest and most easily recognized of the exchange-traded funds. It tops the list in terms of AUM and trading volume, making it appropriate for both tactical traders and buy-and-hold investors. The fund tracks the S&P 500, which are a group of equities from the U.S. large-cap space (not always the largest, keep in mind) and selected by committee. This can be considered to have Beta of 1. (Reference http://www.investopedia.com/articles/etfs/top-etfs-long-term-investments/#ixzz4W4a5oahs)

For volatility I will consider Volatility Index ^VIX , VIX is the ticker symbol for the Chicago Board Options Exchange (CBOE) Volatility Index, which shows the market's expectation of 30-day volatility. It is constructed using the implied volatilities of a wide range of S&P 500 index options. This volatility is meant to be forward looking, is calculated from both calls and puts, and is a widely-used measure of market risk, often referred to as the "investor fear gauge." (Reference http://www.investopedia.com/terms/v/vix.asp#ixzz4W4YteIHC)

For worldwide crude I have considered BRENT and WTI. West Texas Intermediate (WTI) crude oil is a grade of crude oil used as a benchmark for crude oil pricing. The US Department of Energy provides daily prices for WTI Crude, and NYMEX uses WTI Crude as the underlying instrument for its crude oil futures contracts. Brent (North Sea) crude oil is a grade of crude oil used as a benchmark for crude oil pricing. The US Department of Energy provides daily prices for Brent Crude, and ICE uses Brent Crude as the underlying instrument for its crude oil futures contracts. (Reference https://www.quandl.com/collections/markets/crude-oil)

This research project has been divided into two parts:

1. Technical Analysis: Technical analysis is a trading tool employed to evaluate securities and attempt to forecast their future movement by analyzing statistics gathered from trading activity, such as price movement, daily return movement and volume. Technical analysts focus on charts of price movement and various analytical tools to evaluate a security's strength or weakness and forecast future price changes.(http://www.investopedia.com/terms/t/technicalanalysis.asp#ixzz4Xa6C74ku)

   For technical analysis, I will consider only one column for each ticker symbols, that is called as the “Adjusted Close”, basically this is the adjusted closing price for a ticker for that day.

   Based on the “Adjusted Close” I will calculate the “Daily Return” for all ticker symbol (SPY, WTI, BRENT, VIX, XOM) included in this research project.

   I will plot the below mentioned plot of “Adjusted Close” and “Daily Return” to observe directional movement of Daily return :

I. Relationship between market (SPY) and VIX
II. Relationship between VIX and Crude oil (BRENT and WTI), I am assuming spike in volatility may spike the Crude
III. Relationship between Crude and XOM and my assumption is if crude spikes XOM should spike.
IV. If my assumptions are correct then use BRENT, WTI and VIX to backtest XOM

The above plots will help me to understand the directional movement of return between the tickers thus helping to select ticker(s) with similar directional movement for back

2. Back test/Prediction: I will perform back testing/prediction using Classification and Regression Models. This will include multiple steps:
   I. Data is acquired from open data site (yahoo and quandl) and then loaded in a data frame.
   II. Data Cleansing is performed to remove all NULL and Infinite value rows
   III. Adding calculated fields/ features to data set
   IV. Preparing data and algorithm for Classification
      - Classification data is labeled based on the Daily Return, if the “Daily Return” is greater than Zero then the Label is 'Days_Return_Gain' if the “Daily Return” is less than Zero then the Label is 'Days_Return_Loss'
      - Further Classification data is prepared for training and testing. Datasets rows that is less than test start date is designated as training data set and rows greater or equal to test start date is designated as testing data sets.
      - Algorithms that I will use for classifications are RandomForestClassifier, KNeighborsClassifier, SVC, AdaBoostClassifier, GradientBoostingClassifier, QDA. I will use the test dataset score of each of these models to see which model yields the best score and that will be the best classification model for back testing XOM.
   V. Preparing data and algorithm for Regression
      - After cleansing of data, dataset is divided into training and testing sets.
      - The regression algorithm I will use are DecisionTreeRegressor, KNeighborsRegressor, RandomForestRegressor, SVR, BaggingRegressor, AdaBoostRegressor, GradientBoostingRegressor,
      - Next I will calculate the mean_squared_error and r2_score and look for the highest r2_score value and that will be the best regression model for back testing XOM
   VI. For both Classification and Regression model I will use multiple parameters (based on the model) to get the best scores.

Metrics

In general, as the VIX rises (which indicates greater fear of market volatility) stocks tend to decline and vise versa. However my assumption is that volatility spikes the oil market, I will first try to understand the following relationship:

1. Relationship between market (SPY) and VIX , if VIX rises market declines
2. Relationship between VIX and Crude oil (BRENT and WTI), I am assuming spike in volatility may spike the Crude
3. Relationship between Crude and XOM and my assumption is if crude spikes XOM should spike.
4. If my assumptions are correct then use BRENT, WTI and VIX to backtest XOM
5. Idea is to come to a conclusion that should I only consider Crude(WTI and BRENt) and Volatility(VIX) to analyze XOM or should Include the market movement (SPY) in addition to Crude and Volatility.
6. For back test, I will use multiple classification and regression and see which one predicts the best
    1. Algorithm for Classification
        1. Algorithms that I will use for classifications are RandomForestClassifier, KNeighborsClassifier, SVC, AdaBoostClassifier, GradientBoostingClassifier, QDA.
        2. I will fit training dataset clf.fit(X_train, y_train) and then get the accuracy of testing data sets clf.score(X_test, y_test) for each of these models.
        3. My baseline accuracy score will be 0.5, I will change multiple parameters to reach accuracy greater than 0.5
        4. The best classification model will be highest score above 0.5
    2. Algorithm for Regression
        1. The regression algorithm I will use are DecisionTreeRegressor, KNeighborsRegressor, RandomForestRegressor, SVR, BaggingRegressor, AdaBoostRegressor, GradientBoostingRegressor,
        2. I will fit training data reg.fit(X_train, y_train) , then predict test data *reg.predict(X_test*) and then finally calculate mean_squared_error and r2_score Next I will calculate the mean_squared_error mean_squared_error(y_test,Predicted), and r2_score r2_score(y_test, Predicted) .and look for the highest r2_score
        3. My baseline score will be 0.5, I will change multiple parameters to reach score greater than 0.5
        4. The best regression model will be highest score above 0.5

<u>Reasons for selecting baseline accuracy score of 0.5S</u>

I. Both case Classification and Regression my model parameters should be tuned to an extent to avoid overfitting.

II. The XOM model has been solely built based on *quantitative* factors that is movement of Crude and Volatility Index which is *only half of the factors that impact the price/daily return movement.* The other half qualitative factors that impacts score has not been considered in this research projects these factors are Macro-economic conditions, country specific GDP growth, Geo Political conditions etc. Impacts OIL prices.

Macro-Economic indicator encompassing the following area (Changes in the Gross Domestic Product (GDP) GDP is typically considered by economists to be the most important measure of the economy's current health, Income and Wages, Unemployment Rate, Consumer Price Index (Inflation), Currency Strength (buying power), Interest Rates. Corporate Profits. Balance of Trade (Trade Deficit))

For a company with global operations, geopolitical risk is always an issue. Past events in Russia, Nigeria, and Venezuela underscore the risk associated with doing business in those countries. These risks will only become greater as Exxon expands its global production portfolio through partnerships with national oil companies. By investing in large, capital-intensive projects, Exxon also runs the risk that commodity prices will decrease dramatically, making those projects no longer economical.

II. Analysis

Data Exploration

Data exploration starts with downloading above mentioned data sets from Yahoo and Quandl. Python module “DataDownload.py” is used to download data. API and Libraries related to Quandl and pandas_datareader are used to download data. I am downloading data for a specific time period (start and end date).

You will note that not all stock/Index are traded on a daily basis however only SPY is traded every day on which market is open. Thus, for my data exploration SPY is the base line for transaction dates. For my initial analysis SPY is the first data set in the data frame, also for initial analysis I am only considering the Adjusted Closing price for all tickers and Index . The days on which SPY is not transacted I am dropping the rows for the data frame (df.dropna(subset=["SPY"])). Once SPY is added to the data frame for the transaction dates of SPY I am adding the Adjusted Close dates for XOM, BRENT and WTI (for code refer In [52]: of Python Notebook). Since the Adjusted Close price varies widely among different ticker and Index , I am calculating the daily return of Adjusted Close price which means I am normalizing data for visualization perspective (*example sp['Return_SPY'] = sp['AdjClose_SPY'].pct_change()*), I am also changing the names of the columns AdjClose and Return by appending the name of the Ticker (Example AdjClose_XOM , Return_XOM).

SPY DATA SET							
Date	Open_SPY	High_SPY	Low_SPY	Close_SPY	Volume_SPY	AdjClose_SPY	Return_SPY
1/29/1993	43.9687	43.9687	43.75	43.9375	1003200	28.000838	
2/1/1993	43.9687	44.25	43.9687	44.25	480500	28.19999	0.007112359
2/2/1993	44.2187	44.375	44.125	44.3437	201300	28.259704	0.002117518
2/3/1993	44.4062	44.8437	44.375	44.8125	529400	28.558465	0.010571979
2/4/1993	44.9687	45.0937	44.4687	45	531500	28.677956	0.004184083
2/5/1993	44.9687	45.0625	44.7187	44.9687	492100	28.658009	-0.000695552
2/8/1993	44.9687	45.125	44.9062	44.9687	596100	28.658009	0

Above is the example of SPY Data set , the last column “Return_SPY” is the calculated column. You will also see that there has been no trading on 2/6/1193 and 2/7/1993 thus for visual analysis any transaction for tickers XOM, BRENT and WTI on these days will be dropped.

Sample XOM Dataset

Date	Open_XOM	High_XOM	Low_XOM	Close_XOM	Volume_XOM	AdjClose_XOM	Return_XOM
1/2/1990	49.875	50.5	49.25	50	5326000	5.582459	
1/3/1990	49.875	50	49.125	49.5	4980400	5.526634	-0.010000073
1/4/1990	49.375	49.625	48.5	49	6013200	5.470809	-0.010101085
1/5/1990	49	49.25	48.375	48.75	3854800	5.442897	-0.005101988
1/8/1990	48.75	49.75	48.625	49.5	4302000	5.526634	0.015384638
1/9/1990	49.5	49.625	48.5	48.5	3485600	5.414985	-0.020201989
1/10/1990	48.5	48.875	48.25	48.75	4522400	5.442897	0.005154585
1/11/1990	49.125	49.375	48.875	49	3047600	5.470809	0.005128151
1/12/1990	48.25	48.625	47.625	47.75	4554400	5.331248	-0.025510121

BRENT Data Set		
Date	AdjClose_BRENT	Return_BRENT
1/2/1990	21.2	
1/3/1990	22.65	0.068396226
1/4/1990	22.5	-0.006622517
1/5/1990	23.13	0.028
1/8/1990	21.38	-0.075659317
1/9/1990	21.03	-0.01637044
1/10/1990	21.95	0.043747028
1/11/1990	21.88	-0.003189066
1/12/1990	22.13	0.01142596
1/15/1990	21.6	-0.02394939
1/16/1990	21.15	-0.020833333
1/17/1990	20.45	-0.033096927

WTI Data Sets		
Date	AdjClose_WTI	Return_WTI
1/2/1990	22.88	
1/3/1990	23.81	0.040646853
1/4/1990	23.41	-0.016799664
1/5/1990	23.07	-0.014523708
1/8/1990	21.64	-0.061985262
1/9/1990	22.25	0.02818854
1/10/1990	22.9	0.029213483
1/11/1990	23.15	0.010917031
1/12/1990	23.17	0.000863931
1/15/1990	22.36	-0.034958999

Descriptive STST on the Raw Data set that will be predicted (XOM)							
	Open_XOM	High_XOM	Low_XOM	Close_XOM	Volume_XOM	AdjClose_XOM	Return_XOM
count	1768	1768	1768	1768	1.77E+03	1768	1768
mean	83.98323	84.592698	83.365	84.021618	1.63E+07	76.013994	0.000314
std	10.241562	10.231227	10.253956	10.241973	8.29E+06	11.653264	0.011901
min	56.849998	56.990002	55.939999	56.57	4.16E+06	47.005403	-0.061882
25%	78.970001	79.889999	78.370003	79.2175	1.08E+07	69.997888	-0.005881
50%	86.005001	86.5	85.349998	86.019997	1.41E+07	78.605822	0
75%	90.372501	90.980003	89.7825	90.43	1.95E+07	85.074864	0.006876
max	104.419998	104.760002	103.949997	104.379997	1.18E+08	96.106976	0.055159

Note the data received from Yahoo contains 6 columns (open, close, high, low, adjusted close, volume) and data received from Quandl has only one column. It is imperative that data is fully prepared before I can use it for this research project.

Sample Data Set After Adding 4 Features												
Date	Return_XOM	XOM_ADJ_CL_PCT_CHG	XOM_Mov_Agv_Day_Interv	XOM_ADJ_CL_PCT_CHG	XOM_Mov_Agv_Day_Interv	XOM_ADJ_CL_PCT_CHG	XOM_Mov_Agv_Day_Interv	XOM_ADJ_CL_PCT_CHG	XOM_Mov_Agv_Day_Interv	Return_WTI	WTI_ADJ_CL_PCT_CHG	WTI_Mov_
1/6/2017	-0.00057	-0.015463	-0.007736	-0.026296	-0.008825	-0.019499	-0.004874	-0.019499	NaN	0.003906	0.013519	0.006741
1/7/2017	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1/8/2017	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1/9/2017	-0.0165	-0.016497	NaN	-0.016497	NaN	-0.017053	NaN	-0.031705	NaN	-0.03761	-0.037607	NaN
1/10/2017	-0.01275	-0.02904	-0.014625	-0.02904	NaN	-0.02904	NaN	-0.029588	NaN	-0.02175	-0.05854	-0.02967

Here is the Statistics for NaN and Infinity in the Data Sets Before its Cleaned

- Size of merged data frame: (2567, 18)
- Number of NaN after merging: 21722
- Percentage of NaN after merging: 47.0112106653 %
- For the missing dates data has been interpolated using < Portfolio_dataset.interpolate(method = 'time')>
- Interpolated 46100L
- Number of NaN after time interpolation: 106
- Next Filled the Na using < Portfolio_dataset.fillna(Portfolio_dataset.mean()) and temporal shifting>
- Number of NaN after mean interpolation and temporal shifting: 0
- Size of data frame after feature creation: (2564, 20)

Descriptive Stat Of Data After Cleaning																				
	Return_XO	XOM_ADJ_	XOM_Mov_	XOM_ADJ_	XOM_Mov_	XOM_ADJ_	XOM_Mov_	XOM_ADJ_	XOM_Mov_	Return_WT	WTI_ADJ_C	WTI_Mov_A	WTI_ADJ_C	WTI_Mov_A	WTI_ADJ_C	WTI_Mov_A	WTI_ADJ_C	WTI_Mov_A	Return_XOI	Return_WTI2
count	2564	2564	2564	2564	2564	2564	2564	2564	2564	2564	2564	2.56E+03	2564	2564	2564	2564	2564	2564	2564	2564
mean	0.000245	0.000517	0.000372	0.000861	4.18E-04	0.001125	0.000495	0.001012	0.000526	-0.000154	-0.000224	4.71E-05	0.000128	0.000375	-0.000083	0.000339	-0.000528	0.000168	0.000256	-0.00013
std	0.010906	0.014509	0.007486	0.016777	5.86E-03	0.018729	0.004847	0.020712	0.004041	0.020121	0.026017	1.34E-02	0.023466	0.010308	0.033131	0.008436	0.037252	0.006375	0.010838	0.020102
min	-0.061882	-0.067932	-0.034569	-0.084747	-2.84E-02	-0.084747	-0.021745	-0.098518	-0.016349	-0.105232	-0.111705	-5.75E-02	-0.124141	-0.042803	-0.151329	-0.040052	-0.151329	-0.030616	-0.061882	-0.105232
25%	-0.005414	-0.00689	-0.003487	-0.007416	-2.68E-03	-0.008623	-0.002211	-0.009394	-0.001862	-0.010369	-0.013524	-7.03E-03	-0.01551	-0.0057	-0.018202	-0.004323	-0.021221	-0.003745	-0.005392	-0.010327
50%	0.00025	0.000561	0.000382	0.001061	6.52E-04	0.001288	0.000586	0.001554	0.000508	0	-0.000205	1.14E-07	0	0.000287	-0.000217	0.000617	0	0.000535	0.000255	0
75%	0.006251	0.008113	0.004364	0.009294	3.83E-03	0.011023	0.003177	0.012609	0.002909	0.009351	0.012499	6.92E-03	0.015255	0.006142	0.016862	0.004879	0.019357	0.004092	0.006251	0.009351
max	0.055159	0.089361	0.043787	0.092563	3.02E-02	0.092404	0.022592	0.092018	0.015855	0.119511	0.202024	9.64E-02	0.156833	0.050971	0.184982	0.044312	0.277922	0.025769	0.055159	0.119511

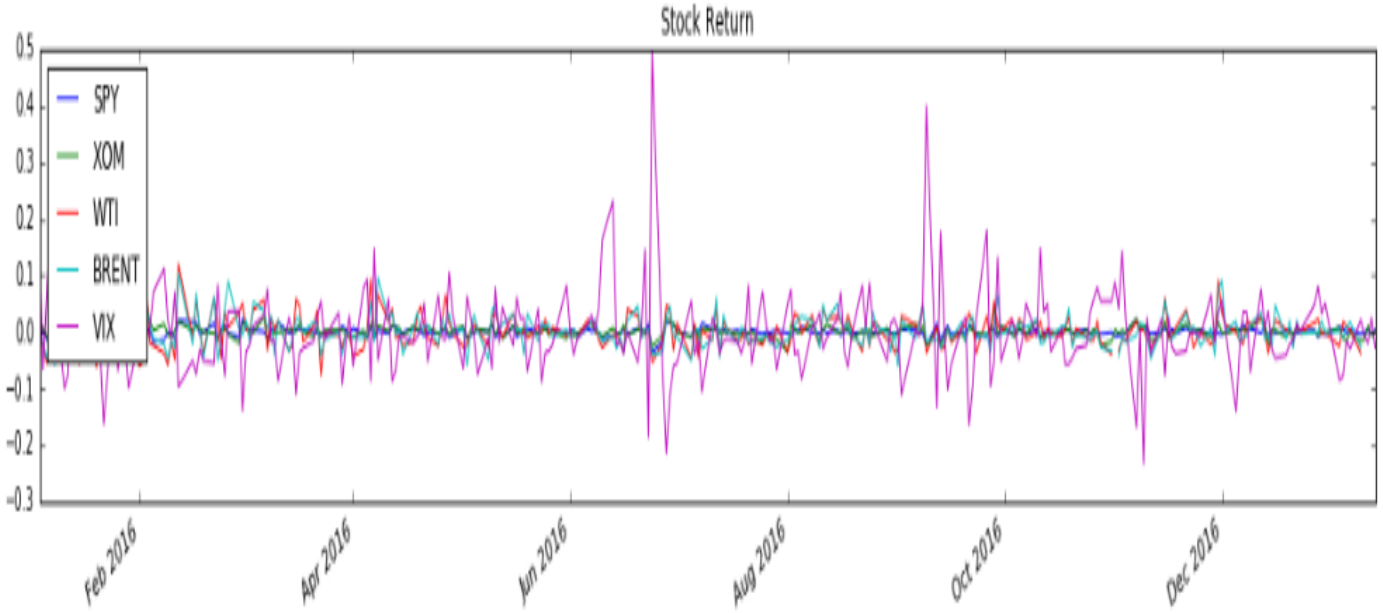
Exploratory Visualization

At first I try to explore and understand the movement of Adjusted Price for a time period Jan 2010 to Jan 2017. Here I am trying to validate my assumptions that if there is an uptick in VIX does the market reacts negatively. In the below mentioned figure we do see that for majority of the upward spikes in VIX we have a negative market(SPY) reaction.

Figure 1

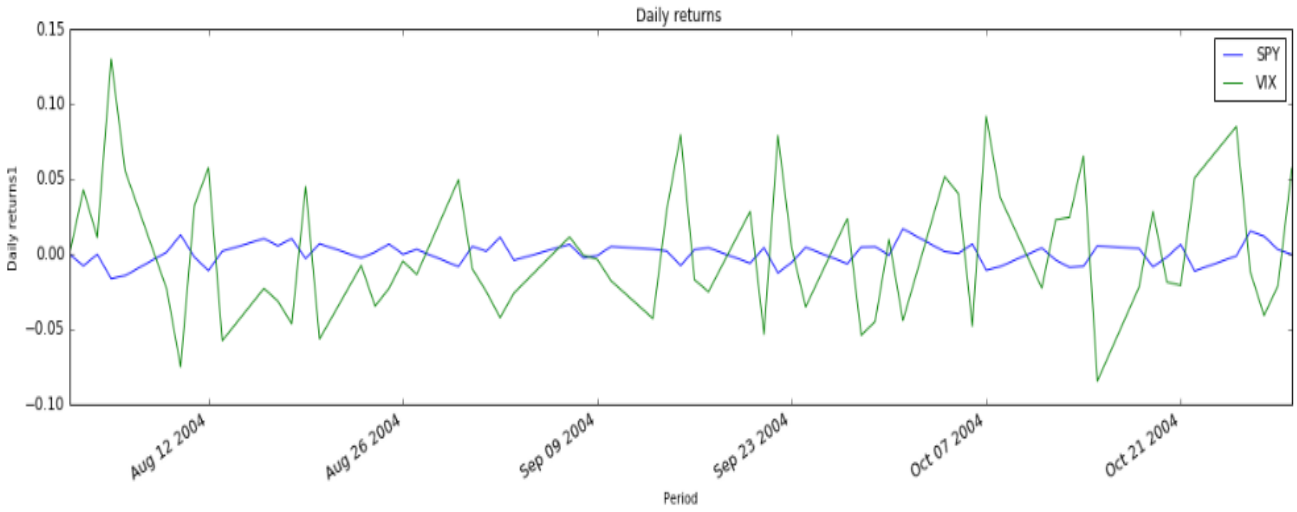
Next I am trying to explore and understand relationship between the Oil market and VIX. For Oil market, I have considered XOM and crude WTI and BRENT. Here we see in **general** for upward spike of VIX there is an upward spike in Oil market. Note not all upward spike in VIX causes an upward spike in Oil market, idea is to see if visually (as a technically trader and not as an fundamental trader) we identify if VIX provides a relatively good identifier for future analysis and which turns out to be true.

Figure 2



Next I am performing a more visual analysis between Volatility and Market to get more clarity than the one we got in Figure-1. Here instead of Adjusted Close we are comparing with Return (Daily Return). Here we are in line with the visualization shown in Figure-1 that for every upward spike in volatility (VIX) there is a downward spike in market (SPY)

Figure 3



Next I will try to validate my assumption that if the market (SPY) goes up Crude moves down and vice versa. Here I have taken a very small time frame of Dec 2016 to Jan 2017. I could say that my assumption of crude movement opposite to market is almost correct.

Figure 4

Daily return comparison between SPY, WTI and VIX . This is more in-depth exploration and visual analysis to check my assumptions and here we see that the market moves opposite to crude and volatility.

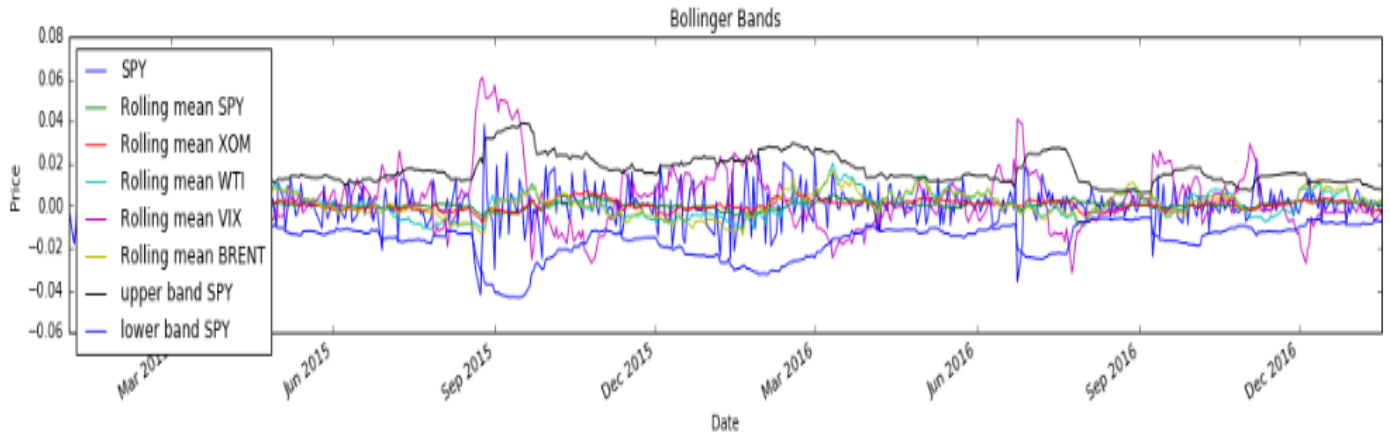
Figure 5

This (Figure-6) I am trying to perform a visual analysis of all five ticker

Figure 6

Next as a final Visual analysis I have sued Bollinger Band with upper and lower limit of the band is for SPY daily return. The Objective is that if there is any crossover over to this upper and lower band . if there is any crossover then I will finally conclude that SPY may not be included in my modeling.
Based on ***Figure 1 thru Figure 7*** I can fairly say that for Back test and Prediction model of XOM I will use Crude (WTI and BRENT) along with Volatility(VTI) and I will exclude market movement (SPY)

Figure 7



Algorithms and Techniques

For Back testing and Prediction of XOM I will use Crude (WTI and BRENT) and Volatility (VIX). I have downloaded XOM , ^VIX and SPY from yahoo using pandas_datareader and WTI and BRENT I have downloaded from Quandl using the quandl API for Python.

I will be using Classification and Regression models to perform back-testing. Data will be stored in pandas data frame sklearn.
Following are the Classification models I will use from sklearn(Refer CallClassification.py):

- ensemble.RandomForestClassifier : A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).(http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)
  Some of the Advantages of are that it runs efficiently on large datasets, it can handle large number of input variables , it can provide estimates for the variable that are important for classification,
  Disadvantage can be that if the data is not well prepped it can overfit.

- neighbors.KNeighborsClassifier: Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.(http://scikit-learn.org/stable/modules/neighbors.html#classification) . Advantages are that cost of learning process is negligible , can be used for noisy datasets. It is disadvantageous for very large datasets

- SVC : Support Vector Classification implementation is based on libsvm. Disadvantage is fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples

- AdaBoostClassifier: An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. Advantages are that the algorithm is fast and only one parameter requires tuning (T), can be used for text, numeric or discrete datasets. Disadvantage the algorithm cannot be used for noisy datasets.

- GradientBoostingClassifier: Gradient Boosting for classifer builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes_ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Binary classification is a special case where only a single regression tree is induced.(http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html) Advantages are that training errors of subsequent trees are corrected based on the tree results of previous one however the process is slow and chances of overfitting are there.

- QDA: A classifier with a quadratic decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.(http://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html#sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis)

For some of the above packages I will use GridSearchCV which will include parameters for respective classifications and make_scorer and r2_score.
The above mentioned Classification are fed with training and testing data sets , these data sets are prepared in Python file PrepDataForClassification.py . For each of the above classification models I will return the accuracy/score.

Following are the Regression models I will use from sklearn(Refer RegressionFunctions.py):

- DecisionTreeRegressor: Decision Trees are a non-parametric supervised learning method used for regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.(http://scikit-learn.org/stable/modules/tree.html#tree) Disadvantage can be that an over complex tree created and may not be good for generalized case. Small change in the training data can cause a big change in final prediction. Advantage is that the process is simple to use and good for fast exploration, output is easy to interpret.

- KNeighborsRegressor: Neighbors-based regression can be used in cases where the data labels are continuous rather than discrete variables. The label assigned to a query point is computed based the mean of the labels of its nearest neighbors. The basic nearest neighbors regression uses uniform weights: that is, each point in the local neighborhood contributes uniformly to the classification of a query point. Under some circumstances, it can be advantageous to weight points such that nearby points contribute more to the regression than faraway points. This can be accomplished through the weights keyword.

The default value, weights = 'uniform', assigns equal weights to all points. weights = 'distance' assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied, which will be used to compute the weights.

- ensemble.RandomForestRegressor: A random forest regressor is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default). (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html)
  Some of the Advantages of are that it runs efficiently on large datasets, it can handle large number of input variables , it can provide estimates for the variable that are important for classification,
  Disadvantage can be that if the data is not well prepped it can overfit.

- SVR: The method of Support Vector Classification can be extended to solve regression problems. This method is called Support Vector Regression. The model produced by Support Vector Regression depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction. (http://scikit-learn.org/stable/modules/svm.html#svm-regression). Disadvantage is that the model is complex and has scalability issues.

- ensemble.BaggingRegressor: A Bagging regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. This algorithm encompasses several works from the literature. When random subsets of the dataset are drawn as random subsets of the samples, then this algorithm is known as Pasting . If samples are drawn with replacement, then the method is known as Bagging . When random subsets of the dataset are drawn as random subsets of the features, then the method is known as Random Subspaces . Finally, when base estimators are built on subsets of both samples and features, then the method is known as Random Patches . Advantages is meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.(http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html)

- ensemble.AdaBoostRegressor: An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. Advantage is that subsequent regressors focus more on difficult cases.(http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html)

- ensemble.GradientBoostingRegressor: Gradient Boosting for regression builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function. (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html) Advantages are GBRT is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems. It has easily handle of data of mixed type (= heterogeneous features). It has better predictive power. It has robustness to outliers in output space (via robust loss functions) Disadvantages are that is less scalable, due to the sequential nature of boosting it can hardly be parallelized (http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting)

For some of the above packages I will use GridSearchCV which will include parameters for respective classifications and make_scorer and r2_score.
The above mentioned Regression are fed with training and testing data sets , these data sets are prepared in Python file PrepDataForRegression.py . For each of the above classification models I will return the Mean Square Error (MSE) and accuracy/score. I will also plot the “Comparative Portfolio Predicted Value with XOM”. For DecisionTreeRegressor I will also perform and plot the “Learning Performance” for multiple depth and Model Complexity.

Following are the Python files used to build and execute the above mentioned models.

0. Run the file DataDownload.py to Download Specific Ticker Related Data
1. BackTest.py is the main file that you need to run
2. SECTION 1 -- LEARNING AND PERFORMANCE CALCULATION (LearnndPerfCalc.py)
3. SECTION 2 -- PLOT LEARNING AND MODEL COMPLEXITY (PlotModelCcomplexity.py) --- This is now obselete use LearnndPerfCalc.py instead
4. SECTION 3 -- GET DATA SETS (GetDataSets.py)
5. SECTION 4-- DATA PREPRATION AND NORMALIZATION (DataPrepNormalization.py)
6. SECTION 5 PREPARE DATA FOR CLASSIFICATION MODELS (PrepDataForClassification.py)

7. SECTION 6 PREPARE DATA FOR REGRESSION MODELS and Call regression models(PrepDataForRegression.py)
8. SECTION 7 CALL THE CLASSIFICATION MODEL (CallClassification.py)
9. SECTION 8 REGRESSION FUNCTIONS (RegressionFunctions.py)
10. SECTION 9 CLASSIFICATION FUNCTIONS (CallClassification.py)

BackTest.py sets multiple parameters and calls multiple models and data cleansing efforts to provide proper output.
Following are the parameters set in the file:

- Ticker_To_Analyze ='XOM'
- Indexing_Ticker = ['WTI','BRENT','VIX']
- Analysis_start_date = datetime.datetime(2010,1,1)
- Analysis_end_date = datetime.datetime(2017,1,10)
- Index_Comp_Start_Date = datetime.datetime(2010,1,1)
- Index_Comp_End_Date = datetime.datetime(2017,1,10)
- Train_start_Date = datetime.datetime(2016, 3, 20)
- MoveUp = range(2, 3)

Explanation for each of the above parameters:

- Ticker_To_Analyze This the ticker that we would like to analyze
- Indexing_Ticker This is the Index Tickers that we will build our models to comapre with
- Analysis_start_date This is the start date for the ticker data
- Analysis_end_date This is the end date for the ticker data)
- Index_Comp_Start_Date This is the start date for the Index ticker data
- Index_Comp_End_Date This is the end date for the Index ticker data
- Train_start_Date Training will be performed on the datasets before this date
- MoveUp--> Shift up by the range mentioned in the range to remove any NaN

Steps:

- getPortfolioData is called to download Index data for the specific date ranges , in this case it is XOM data for date range between 10th Jan 2017 and 1st Jan 2010
- downloadStockToPredict is called to download Index data for the specific date ranges , in this case it is VIX, BRENT and WTI data for date range between 10th Jan 2017 and 1st Jan 2010
- Adding XOM as the first data set in the frame
- check_AddFeatures Add additional 4 features features for <Ticker>_ADJ_CL_PCT_CHG_DAY_<range> and <Ticker>_Mov_Agv_Day_Interval_<Range>
    - addAdjClosePercentChangeAndMovingAvgOfReturn() is called to add additional features(DataPrepNormalization.py)
    - keepCalculatedColumnsOnly() keeps only the calculated normalized columns and removes all others not required columns (DataPrepNormalization.py)

    - interpolate() method used to fill in the dates between the missing dates
    - fillna() method is used to remove the next set of NaN
    - Remove_NaN() is used to remove any left out NaN data (DataPrepNormalization.py)
- Calling prepareDataForClassification to prepare training and testing data sets to be used By CLASSIFICATION MODELS(PrepDataForClassification.py)
- Calling Multiple Model Of Classification
- Calling callRegressionModel to prepare training and testing data sets to be used By REGRESSION MODELS(PrepDataForRegression.py)
- Calling Multiple Model For Regression (RegressionFunctions.py)

Benchmark

For Classification model if the accuracy score is higher than 0.5 will be considered as the pool of models that can be considered for back testing XOM , after the selecting these models I will try changing the model parameter using GridSearchCV and look for the best accuracy score above 0.5 and model with best accuracy score will be the model and the output for back testing XOM.

For Regression model if the r2_score is higher than 0.5 will be considered as the pool of models that can be considered for back testing XOM , after the selecting these models I will try changing the model parameter using GridSearchCV and look for the best accuracy score above 0.5 and model with best accuracy score will be the model and the output for back testing XOM.

Reasons for selecting baseline accuracy score of 0.5S

The XOM model has been solely built based on *quantitative* factors that is movement of Crude and Volatility Index which is *only half of the factors that impact the price/daily return movement.* The other half *qualitative* factors that impacts score has not been considered in this research projects these factors are Macro-economic conditions, country specific GDP growth, Geo Political conditions etc. Impacts OIL prices. (Refer analyst articles from Morning Star https://olui2.fs.ml.com/MDWSODUtility/PdfLoader.aspx?src=%2fnet%2fUtil%2fGetPdfFile%3fdockey%3d200-30231G10-5K2MGH153Q70TM7EDB9U0LKL5P , **S&P CAPITAL IQ KEEPS HOLD OPINION ON SHARES OF EXXON MOBIL CORPORATION**)

Macro-Economic indicator encompassing the following area (Changes in the Gross Domestic Product (GDP) GDP is typically considered by economists to be the most important measure of the economy's current health, Income and Wages, Unemployment Rate, Consumer Price Index

(Inflation), Currency Strength (buying power), Interest Rates. Corporate Profits. Balance of Trade (Trade Deficit))

For a company with global operations, geopolitical risk is always an issue. Past events in Russia, Nigeria, and Venezuela underscore the risk associated with doing business in those countries. These risks will only become greater as Exxon expands its global production portfolio through partnerships with national oil companies. By investing in large, capital-intensive projects, Exxon also runs the risk that commodity prices will decrease dramatically, making those projects no longer economical.

III. Methodology

Data Preprocessing

What data do I need ?

I will analyze XOM against BRENT , WTI and VIX

How will I get data?

Using "**DataDownload.py**" I will download datasets from yahoo and quandl

- Modules pandas_datareader is imported to get data from Yahoo, and import quandl to get data from Quandl.
- Set the path where you want to save downloaded datasets. In my case it stores in the directory assigned to the variable "path"
- Change the name of the last column to "AdjClose"
- Append the column name with ticker symbol
- Add one additional column "Return_<<Ticker Name>>" to the csv file

For code detail refer file "**DataDownload.py**".

How will I get XOM data in a data Frame?

In file ***BackTest.py*** a call is made to function downloadStockToPredict(Ticker_To_Analyze, Analysis_start_date, Analysis_end_date,stock_file_directory) and the value is stored in Ticker_To_Predict.

GetDataSets.py contains the method downloadStockToPredict() It uses the following parameters supplied to the method.

Ticker_To_Analyze='XOM'
Analysis_start_date = datetime.datetime(2010,1,1)
Analysis_end_date = datetime.datetime(2017,1,10)

stock_file_directory=
'C:/Training/udacity/MachineLearningEngineerNanodegree/P5/Project/Final/BackTestClassRegression/data/'

- First see if the csv file is available in the mentioned directory
- If data file is there then get the data between the start date and end date
- If data file is not there then get data from Yahoo , change the last column name to AdjClose and add "Return" column which is the change in the AdjClose column and append _XOM to all the columns.

For code details refer file ***BackTest.py*** and ***GetDataSets.py***

How will I get Index data (BRENT, WTI and VIX)?

In file ***BackTest.py*** a call is made to function
getPortfolioData(stock_file_directory,Index_Comp_Start_Date,Index_Comp_End_Date,Indexing_Ticker)

GetDataSets.py contains the method getPortfolioData () It uses the following parameters supplied to the method:

Ticker_To_Analyze ='XOM'
Indexing_Ticker = ['WTI','BRENT','VIX']
Index_Comp_Start_Date = datetime.datetime(2010,1,1)
Index_Comp_End_Date = datetime.datetime(2017,1,10)
stock_file_directory=
'C:/Training/udacity/MachineLearningEngineerNanodegree/P5/Project/Final/BackTestClassRegression/data/'

The method reads the CSV file stored as the ticker name and passed as a parameter "Indexing_Ticker". Collect Index ticker stored in Indexing_Ticker and store it in a dataframe. Here I am getting data between specific dates and not complete dataset that has been downloaded from yahoo or quandal.

For code details refer file ***BackTest.py*** and ***GetDataSets.py***

How do I combine the two sets of data?

From the previous steps I get two data sets and then make XOM as my First dataset and then rename the full data set as Portfolio_Analysis_Dataset (File ***BackTest.py***) . I use the "insert" function to add XOM as my first dataset.

For Code details refer file ***BackTest.py***

How bad or good is my dataset?

One basic thing to understand is that Classification and Regression model cannot be run on the downloaded data stored in data frame. Here are the reasons and solutions I have adopted:

I. Data downloaded from yahoo has 6 columns and data downloaded from Quandl has just one column
II. First, I have to identify which column should be used from each of these downloaded file
III. In my case the column “Adjusted Close” is the column best suited for analysis
IV. Note the value of “Adjusted Close” cannot be used directly as the values of different tickers varies widely so I have to think about normalizing.
V. To address the varied value of “ Adjusted Closed” I will create a column “Return” based on the daily changes of Adjusted close data.
VI. Next I will have to add features for the models, remove NaN , interpolate data and scale up to clean any anomalies and non-numeric values.
VII. Then I will remove all original columns and only keep the calculated columns. This sanitizes my datasets for running models.

Here are the anamolies in data and the results after the data has been cleansed and sanatized:

- Size of merged data frame: (2567, 18)
- Number of NaN after merging: 21722
- Percentage of NaN after merging: 47.0112106653 %
- For the missing dates data has been interpolated using < Portfolio_dataset.interpolate(method = 'time')>
- Interpolated 46100L
- Number of NaN after time interpolation: 106
- Next Filled the Na using < Portfolio_dataset.fillna(Portfolio_dataset.mean()) and temporal shifting>
- Number of NaN after mean interpolation and temporal shifting: 0
- Size of data frame after feature creation and dropping all the original columns and just keeping the calculated columns: (2564, 20)

Descriptive Stat Of Data After Cleaning

	Return_XO	XOM_ADJ_	XOM_Mov_	XOM_ADJ_	XOM_Mov_	XOM_ADJ_	XOM_Mov_	XOM_ADJ_	XOM_Mov_	Return_WT	WTL_ADJ_C	WTL_Mov_A	WTL_ADJ_C	WTL_Mov_A	WTL_ADJ_C	WTL_Mov_A	WTL_ADJ_C	WTL_Mov_A	Return_XOI	Return_WTI2
count	2564	2564	2564	2564	2564	2564	2564	2564	2564	2564	2564	2.56E+03	2564	2564	2564	2564	2564	2564	2564	2564
mean	0.000245	0.000517	0.000372	0.000861	4.18E-04	0.001125	0.000495	0.001012	0.000526	-0.000154	-0.000224	4.71E-05	0.000128	0.000375	-0.000083	0.000339	-0.000528	0.000168	0.000256	-0.00013
std	0.010906	0.014509	0.007486	0.016777	5.86E-03	0.018729	0.004847	0.020712	0.004041	0.020121	0.026017	1.34E-02	0.029466	0.010308	0.033131	0.008496	0.037252	0.006975	0.010898	0.020102
min	-0.061882	-0.067992	-0.034569	-0.084747	-2.84E-02	-0.084747	-0.021745	-0.098518	-0.016349	-0.105292	-0.111705	-5.75E-02	-0.124141	-0.042803	-0.151329	-0.040052	-0.151329	-0.030616	-0.061882	-0.105292
25%	-0.005414	-0.00689	-0.003487	-0.007416	-2.68E-03	-0.008623	-0.002211	-0.009994	-0.001862	-0.010369	-0.013524	-7.03E-03	-0.01551	-0.0057	-0.018202	-0.004323	-0.021221	-0.003745	-0.005392	-0.010327
50%	0.00025	0.000561	0.000382	0.001061	6.52E-04	0.001288	0.000586	0.001554	0.000508	0	-0.000205	1.14E-07	0	0.000287	-0.000217	0.000617	0	0.000535	0.000255	0
75%	0.006251	0.008113	0.004364	0.009294	3.83E-03	0.011023	0.003177	0.012609	0.002909	0.009351	0.012499	6.92E-03	0.015255	0.006142	0.016862	0.004879	0.019357	0.004092	0.006251	0.009351
max	0.055159	0.089361	0.043787	0.092563	3.02E-02	0.092404	0.022592	0.092018	0.015855	0.119511	0.202024	9.64E-02	0.156833	0.050971	0.184982	0.044312	0.277922	0.025769	0.055159	0.119511

How can I add additional normalized features to the dataset?

In file ***BackTest.py*** I will call
check_AddFeatures(Portfolio_Analysis_Dataset,Add_Features,Ticker_To_Analyze,MoveUp)

The method performs following tasks:

- It will create additional computed columns for each ticker by calling addAdjClosePercentChangeAndMovingAvgOfReturn(Individual_dataset, adjclose, returns, n)
- It will remove all non-calculated columns for each ticker by calling keepCalculatedColumnsOnly(Portfolio_Analysis_Dataset)

```
def check_AddFeatures(Portfolio_Analysis_Dataset,Add_Features,Ticker_To_Analyze,MoveUp):
    for Individual_dataset in Portfolio_Analysis_Dataset:
        columns = Individual_dataset.columns
        adjclose = columns[-2]
        returns = columns[-1]
        for n in Add_Features:
            addAdjClosePercentChangeAndMovingAvgOfReturn(Individual_dataset, adjclose, returns, n)

    Portfolio_dataset = keepCalculatedColumnsOnly(Portfolio_Analysis_Dataset)

    print ('Initial Size of portfolio: ', Portfolio_dataset.shape)
    print ('Total Percentage of Non Numeric and Infinity in Initial Portfolio: ', (Find_NaN(Portfolio_dataset

    """
```

- It will create additional rows by using interpolate
- It will fill in the NaN column
- Finally I will remove the any pending NaN or Infinite value column by calling Remove_NaN(Portfolio_dataset, MoveUp, Add_Features, back)
- Detail of addAdjClosePercentChangeAndMovingAvgOfReturn(Individual_dataset, adjclose, returns, n) **(refer file DataPrepNormalization.py)**

Pick up the Ticker Symbol from AdjClose_Ticker means if you are analyzing for XOM (dataset.insert(0,out where out =XOM in this case) so form AdjClose_XOM Pick XOM then to it add the word "_ADJ_CL_PCT_CHG_DAY" followed by days range which is defined in delta = range(2, 5) this is the value received in "n"

Let Say n = 2,3,4

Date	Open	High	Low	Close	Volume	Adj Close
12/31/2013	100.489998	101.389999	100.43	101.199997	8509600	91.899766
12/30/2013	101.529999	101.550003	100.309998	100.309998	9007900	91.091558
12/27/2013	101.239998	101.739998	100.989998	101.510002	10209000	92.181282
12/26/2013	99.419998	101.029999	99.379997	100.900002	9531200	91.62734
12/24/2013	98.330002	99.440002	98.330002	99.220001	4168300	90.101731
12/23/2013	99	99.290001	98.389999	98.510002	10127600	89.456981
12/20/2013	99.389999	99.599998	98.599998	98.68	23331000	89.611356

Then
XOM_DELTA_2 = (91.899766 - 92.181282)/92.181282=-0.003054 -- % Change in 2 days
XOM_DELTA_3 = (91.899766 - 91.62734)/91.62734=0.002973196 -- % Change in 3 days
XOM_DELTA_4 = (91.899766 - 90.101731)/90.101731=0.01995561 -- % Change in 4 days

Get the name of the Ticker from Return_Ticker Name example XOM from 'Return_XOM' Then calculate the Moving Average of return's (daily % return of Adj Close) for intervals received from the value of n (n=2,3,4) and sdd it to a new column in Dataframe

- Details of keepCalculatedColumnsOnly(Portfolio_Analysis_Dataset). For code refer file **(File DataPrepNormalization.py)**

  This method removes (anything upto AdjClose) original columns of "Open,High,Low,Close, AdjClose). It will first drop the First data set that is in this case XOM and will store the remaining datasets in a new dataset called dataset_subset_rest.

  Now Find the column name anything that has "AdjClose" by using the following

  tt = (Individual_dataset.columns[Individual_dataset.columns.str.startswith('AdjClose_')])

  tt =''.join(map(str,tt))

  Once the name is found then find the location of the column using

  pos_adj_cls=Individual_dataset.columns.get_loc(tt)

  After finding the location of the column add 1 as you want to select all the columns(calculated columns) after the AdjClose columns. Do the same thing for the first data set (XOM) and then join the two . Now join/merge the Y and X this will provide XOM, BRENT etc with XOM in first and with only with the relevant calculated columns

  datasets[0].iloc[:,pos_adj_cls_for_ticker_to_predict:].join(dataset_subset_rest, how = 'outer')
- Detail of Remove_NaN(dataset, MoveUp, delta, back). For code refer file **(File DataPrepNormalization.py)**

  Here we are moving the rows one up to remove the first row of NaN created by the moving average and returns created earlier adding features.

How do I get training and testing data for Classification Models?

Call the method prepareDataForClassification(Portfolio_dataset, Train_start_Date,Ticker_To_Analyze) , this will return X_train, y_train, X_test, y_test this method will be called in file ***BackTest.py***

- Details of prepareDataForClassification() , this method is available in file ***PrepDataForClassification.py*** Data will be classified based on the daily return . There will be 2 types of classification one "***'Days_Return_Gain***" if the daily return is greater than 0 and if the daily return is less than 0 then it will be classified as "***Days_Return_Loss***"

  1. Add a column to the dataset called "Classification"

  2. Assign Return Value of Ticker value to the column "Classification"

  3. If the Classification is >=0 then return 'Days_Return_Gain'

  4. If the Classification is <0 then return 'Days_Return_Loss'

  5. Use sklearn preprocessing.LabelEncoder() ' fit and transform to transform Days_Return_Gain and Days_Return_Loss to 0 or 1

  6. All columns except the last column "Classification" is feature and the last column is for testing

  7. All training data belongs to the timeperiod less than the start_test

  8. All testing data belongs to the timeperiod = > than the start_test

How do I prepare data for Regression Models?

Call the method callRegressionModel(Portfolio_dataset,.6)this will return X_train_reg, y_train_reg, X_test_reg, y_test_reg,output this method will be called in file ***BackTest.py***

- Details of callRegressionModel() ***(File PrepDataForRegression.py)***
  Based on the value of split (>0 and < 1) and shape of data set create an index value and based on the index value split the datasets in training and testing

Implementation & Refinement

Software and Libraries required are mentioned in README document. So far I have explained about Downloading data (DataDownload.py), Read data and get into the data Frame (GetDataSets.py and BackTest.py), adding features and calculated columns (DataPrepNormalization.py), prepare data for classification model (PrepDataForClassification.py), prepare data for regression models (PrepDataForRegression.py).
In this section I will provide the details related to Classification models, Regression models and complexity of model and performance of learning.

What Regression Models I have used?

Regression models are called in BackTest.py , here is how it is called .

```
"""
Calling Multiple Model For Regression
"""
print('========== Regression Results ==============================')
print ('GridSearchCV Tuning on Decission Tree Regressor ', exeDecisionTreeRegressor(X_train_reg, y_train_reg, X_test_reg, y_test_reg))
print('')
print ('KNN Regression Score  ', exeKNeighborsRegressor(X_train_reg, y_train_reg, X_test_reg, y_test_reg))
print('')
print ('RandomForest Regression Score  ',exeRandomForestRegressor(X_train_reg, y_train_reg, X_test_reg, y_test_reg))
print('')
print ('SVR Regression Score  ',exeSVR(X_train_reg, y_train_reg, X_test_reg, y_test_reg))
print('')
print ('Bagging Regression Score  ',exeBaggingRegressor(X_train_reg, y_train_reg, X_test_reg, y_test_reg))
print('')
print ('AdaBoost Regression Score  ',exeAdaBoostRegressor(X_train_reg, y_train_reg, X_test_reg, y_test_reg))
print('')
print ('GradientBoosting Regression Score  ',exeGradientBoostingRegressor(X_train_reg, y_train_reg, X_test_reg, y_test_reg))
```

- exeDecisionTreeRegressor This method is available in file **RegressionFunctions.py**
  Tunes a decision tree regressor model using GridSearchCV on the input data X and target labels y and returns this optimal model. Predicted (XOM which is being analyzed) and the Portfolio is plotted in this method. This method also plots learning performance and model complexity graphs.

  Here are the parameters after **Refinement**

```
regressor = DecisionTreeRegressor( max_depth=2,min_samples_leaf=1, min_samples_split=2,splitter='best')
 # Set up the parameters we wish to tune
 parameters = [{'max_depth':(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20),'presort':['True']}]
 # Make an appropriate scoring function
 scoring_function = None
```

```
#scoring_function = make_scorer(performance_metric(), greater_is_better=False)
scoring_function = make_scorer(r2_score, greater_is_better=True)
# Make the GridSearchCV object
reg = None
reg = GridSearchCV(regressor, parameters,scoring=scoring_function, cv=10)
```

- exeKNeighborsRegressor , this calls the K Nearest Neighbour regression and returns MSE and r2_score. Next training data set is fit followed by predicting the testing data sets. Finally the mean_square and r2_square is calculated on test and predict. For details of code refer This **RegressionFunctions.py**

- Method exeRandomForestRegressor () invokes the regression method ensemble.RandomForestRegressor. **Refinement** has provided better results for this repressor. Here are the **Refinement parameters**

  parameters = [{'n_estimators':[20],'criterion':['mse'],'min_weight_fraction_leaf':[0.25],'n_jobs':[-1]}]
  scoring_function = make_scorer(r2_score, greater_is_better=True)
  # Make the GridSearchCV object
  clf = GridSearchCV(clf, parameters,scoring=scoring_function, cv=10)

  Next training data set is fit followed by predicting the testing data sets. Finally, the mean_square and r2_square is calculated on test and predict For details of code refer This **RegressionFunctions.py**

- Method exeSVR() calls repressor SVR(). **Refinement** has provided better results for this repressor.
  Here are the **Refinement parameters**

  parameters =[{'C': [1, 10, 100, 1000], 'gamma': [1e-1, 1, 1e1], 'kernel': ['rbf','linear', 'poly','sigmoid'],'degree': [3],'epsilon':[0.9]}]
  scoring_function = make_scorer(r2_score, greater_is_better=True)
  # Make the GridSearchCV object
  clf = GridSearchCV(clf, parameters,scoring=scoring_function,cv=10)

  Next training data set is fit followed by predicting the testing data sets. Finally, the mean_square and r2_square is calculated on test and predict For details of code refer This **RegressionFunctions.py**

- Methods exeBaggingRegressor, exeAdaBoostRegressor , exeGradientBoostingRegressor have no additional Refinement and the code can be found in **RegressionFunctions.py**

What Classification Models I have used?

Classification models are called in **BackTest.py** , here is how it is called .

```
"""
Calling Multiple Model Of Classification
"""
print('========== Classification Results ==============================')
print("Classification Randon Forest Score :", performRFClass(X_train, y_train, X_test, y_test))
print("Classification KNN Score :", performKNNClass(X_train, y_train, X_test, y_test))
print("Classification SVM Score :", performSVMClass(X_train, y_train, X_test, y_test))
print("Classification GradientBoostingClassifier Score :", performGTBClass(X_train, y_train, X_test, y_test))
print("Classification QDA Score :", performQDAClass(X_train, y_train, X_test, y_test))
print('')
```

The above-mentioned methods are available in file **CallClassification.py.** I will discuss the implementation details of each of these methods

- performRFClass this method invokes ensemble.RandomForestClassifier with n_estimators=100, n_jobs=-1. This method was initially implemented by using GridSearchCV(clf, parameters,scoring=scoring_function, cv=10),
  parameters = [{'n_estimators':[10],'criterion':['entropy'],'min_weight_fraction_leaf':[0.5],'n_jobs':[-1]}] and been tested for better results by changing the parameters however the native provided the best results thus as a **Refinement** these parameter were removed for best accuracy score.
  After calling the classifier train data is fit and then the accuracy of test data is derived.

- performKNNClass, this method invokes the neighbors.KNeighborsClassifier multiple Refinement has been tried out but the best results are with the following parameters:
  parameters = [{'n_neighbors':[20],'weights':['distance'],'algorithm':['auto'],'n_jobs':[-1]}]
  scoring_function = make_scorer(r2_score, greater_is_better=True)
  clf = GridSearchCV(clf, parameters,scoring=scoring_function, cv=10)
  After calling the classifier train data is fit and then the accuracy of test data is derived.

- **performSVMClass** invokes SVC , **performAdaBoostClass** invokes AdaBoostClassifier, **performGTBClass** invokes GradientBoostingClassifier and **performQDAClass** invokes QDA
  After calling the classifier train data is fit and then the accuracy of test data is derived. For code details refer file **CallClassification.py**

IV. Results

Model Evaluation and Validation and Jsutification

In this section I will provide results from all 5 Classification models and 7 regression models. These results are provided to show a comparative result from different types of model and gauge the level of suitability. In the beginning of this document I have indicated that I am looking for a model which will have accuracy/score higher than 0.5.

For Classification except KNN all other models have yielded accuracy higher than 0.5 however Gradient Boost Classification has the highest score .

Classification Scores:

Randon Forest Score	0.57770270270270274
KNN Score	-0.78582202111613864
SVM Score	0.53378378378378377
GradientBoosting Score	**0.58108108108108103**
QDA Score	0.5033783783783784

Classification Scores 1

For regression models the best result is from Gradient Boosting. In next section I will also show the learning performance and model complexity, for this I will use Decision Tree Repressor.

Regression Scores

DecisionTree MSE	6.4470216005132615e-05	**DecisionTree R2**	0.50564567164627916
KNeighbors MSE	6.5617126369942904e-05	**KNeighbors R2**	0.49685122146121108
RandomForest MSE	6.3701700044828074e-05	**RandomForest R2**	0.51153861283566837
SVR MSE	0.00015178520299729577	**SVR R2**	0.1638811955552868
Bagging MSE	6.4357554154635896e-05	**Bagging R2**	0.50650955697014655
AdaBoost MSE	7.8034688265385581e-05	**AdaBoost R2**	0.40163399014119183
GradientBoosting MSE	6.013268661360343e-05	**GradientBoosting R2**	**0.53890562580702239**

Regression Scores 1

I have varied the size of data sets to validate both classification and regression models to see if the model yields score at par with the assumption of 0.5. Random forest, Gradient Boost and Decision tree has yielded score better than my assumptions.

Based on multiple run with varying size of train and test data size for the type of datasets I have used, Gradient Boosting has yielded best scores. This supports my previous discussions I have mentioned in this paper "*Advantages of Gradient Boosting is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems*" mainly because Gradient Boosting for regression and classification builds an additive model in a forward stage-wise fashion; it allows for the

optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

For the above mentioned reasons I have considered Gradient Boosting as best algorithm for Classification and Regression to back test and predict XOM.

V. Conclusion

Free-Form Visualization & Reflection

At first I have tried to understand what tickers/index can provide a comparative analysis for XOM , based on readings from Barrons.com, Investopedia, Yahoo Finance and Google Finance I was able to understand that Oil stock can be analyzed based on the crude market and volatility index. On further reading I was able to see that there exists a relationship between XOM (Exxon Mobile), Crude (BRENT and WTI) and Volatility (VIX).
Next I have tried to confirm my understanding by calculating the daily change of Adjusted close and plotting the same for all 4 ticker and plot (Refer Figure 1 Through Figure7). I was able to confirm volatility moves opposite to the market(SPY) and moves in tandem with Crude and next was able to confirm majority of the time Oil moved in the same direction of Crude.

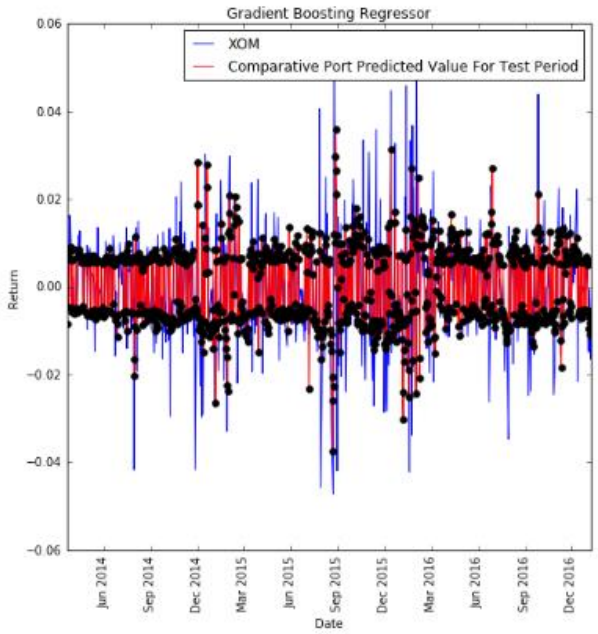
Next after deciding on ticker I have tried to predict XOM based on classification and regression models. I have tried the prediction based on multiple models Of which I found that for Classification Random forest , Gradient Boost are best models where as for Regression Decision Tree Random forest , Gradient Boost are best fit .

I have tried to perform analysis on “Learning Performance” and “Model Complexity” based Decision Tree models.

Here is the Visualization Results for Gradient Boost Regression , which shows that more than 50% of the time XOM return can be in line with the portfolio return (WTI, BRENT and VIX)

Gradient Boost Regression Plot 1

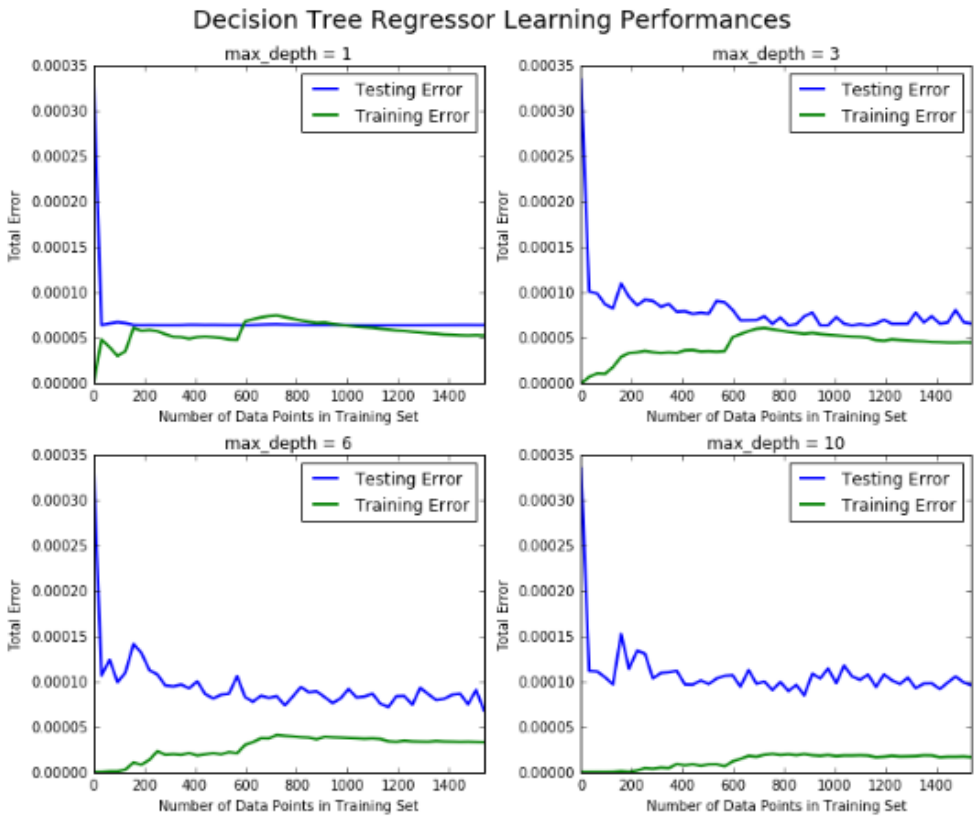
```
('GradientBoostingRegressor Score = ', 0.53890562580702239)
```




```
('GradientBoosting Regression Score  ', ('GradientBoosting MSE =', 6.013268661360343e-05, 'GradientBoosting R2 =', 0.
53890562580702239))
```

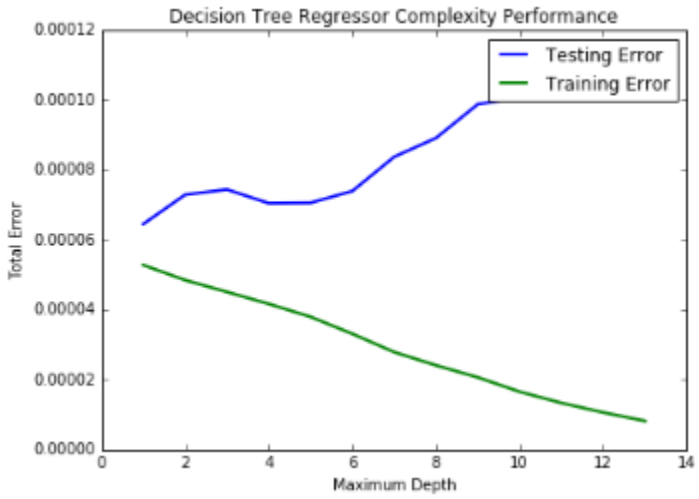
Next is the visualization output of Decision Tree Regressor Learning Performance. I have tested the Learning performance by varying the depth and in this case I shown results with depth of 1,3,6 and 10. The data set I have analyzed I have seen with the increase of depth parameter for Decision Tree Regression the difference between testing and training error increases with the increase of depth parameter. Probably for the type of data set I am using the Decision Tree Model may be best fit with depth 1.

Decision Tree Model Performance At Multi 1



In my case the model complexity (Decision Tree Regression) increases with the increase of depth. This is shown in the below mentioned graph.

Decision Tree Model Complexity 1



Improvement

The XOM model has been solely built based on quantitative factors that is movement of Crude and Volatility Index which is only half of the factors that impact the price/daily return movement. The other half qualitative factors that impacts score has not been considered in this research projects these factors are Macro-economic conditions, country specific GDP growth, Geo Political conditions etc. Impacts OIL prices. (Refer analyst articles from Morning Star https://olui2.fs.ml.com/MDWSODUtility/PdfLoader.aspx?src=%2fnet%2fUtil%2fGetPdfFile%3fdockey%3d200-30231G10-5K2MGH153Q70TM7EDB9U0LKL5P , S&P CAPITAL IQ KEEPS HOLD OPINION ON SHARES OF EXXON MOBIL CORPORATION)

Macro-Economic indicator encompassing the following area (Changes in the Gross Domestic Product (GDP) GDP is typically considered by economists to be the most important measure of the economy's current health, Income and Wages, Unemployment Rate, Consumer Price Index (Inflation), Currency Strength (buying power), Interest Rates. Corporate Profits. Balance of Trade (Trade Deficit))

For a company with global operations, geopolitical risk is always an issue. Past events in Russia, Nigeria, and Venezuela underscore the risk associated with doing business in those countries. These risks will only become greater as Exxon expands its global production portfolio through partnerships with national oil companies. By investing in large, capital-intensive projects, Exxon also runs the risk that commodity prices will decrease dramatically, making those projects no longer economical.

In order to improve model an effort should be put in to find an index(s) that supports Macr-Economic indicator encompassing the following area (Changes in the Gross Domestic Product (GDP) GDP is typically considered by economists to be the most important measure of the economy's current health, Income and Wages, Unemployment Rate, Consumer Price Index (Inflation), Currency Strength (buying power), Interest Rates. Corporate Profits. Balance of Trade (Trade Deficit))
Indexes related to GDP growth (http://www.investopedia.com/articles/economics/08/leading-economic-indicators.asp)

In addition to the above-mentioned improvement's I will try TensorFlow developed and tested by Google. TensorFlow is Google's next generation machine learning library, allowing you to build high performance, state-of-the-art, scalable deep learning models. Cloud Platform provides the compute and storage on demand required to build, train and test those models. The two together are a marriage made in heaven and can provide a tremendous force multiplier for your business. (https://cloudplatform.googleblog.com/2016/03/TensorFlow-machine-learning-with-financial-data-on-Google-Cloud-Platform.html)

References:

1. MachineLearningGeorgiaTech
2. IntroToMachineLearning
3. Python for finance (Analyze big financial data by O' Reily)
4. Machine Learning (Tom M Mitchell)
5. http://finance.yahoo.com
6. https://www.google.com/finance
7. quandl.com
8. https://studywolf.wordpress.com/2012/11/25/reinforcement-learning-q-learning-and-exploration/