Capstone Report

Directional ETF Prediction

I. Definition

Domain Background:

Today's called Efficient Market Hypothesis (EMH) based on the research of Samuelson (1965) and Fama (1965) states that Stock markets behavior and more importantly, stock prices behavior, follow a "random walk" if investors behaved rationally (Shleifer and Summers, 1990); Shleifer And Summers (1990) stated that this theory lost grounds rapidly thanks to publications such as Shiller's (1981), Leory and Porter's (1981) and it particularly crashed after the market crash on October 19, 1987.

In other words, according to the EMH (Nardo, Petracco-Giudici and Naltsidis, 2016) financial prices are not predictable, however, many attempts have been made to predict them. More recently, thanks to technological and data processing advances new machine learning (ML) techniques have been used because these allow to uncover generalizable patterns not specified in advance (Mullainathan and Spiess, 2017). Some examples are: directional prediction (classification) based on news contents (Alsotad and Davalcu, 2017) and stock daily returns (Gunduz, Cataltepe and Yaslan, 2017 and Li, Xie and Wang, 2016) to mention a few.

This project intends to apply Supervised ML classification techniques to predict the directional movement of a selected company's stock or ETF. From a personal point of view, this study and its outcome(s) will help the author of this work to start a personal investment strategy based on these initial results and further improvements over time; this is because current investment alternatives (financial advisors, etc) offer high prices/costs for their services, however, they don't share the risks of the investment strategy suggested to their clients (a.k.a. the low side).

Performing this project will also train the author to work with time series data as well as understanding basic concepts of stock trading such as data preparation and processing.

Problem Statement:

This project's objective is to define directionality of the closing price of a stock/ETF *N* days in the future relative to its daily opening price i.e. whether its price is predicted to increase (up), or decrease (down). Given the binary nature of the outcome, supervised learning techniques will be used to achieve the objective; specifically, classification techniques such as Decision Tress, Gaussian Naïve Bayes, Support Vector Machine and Neural Networks (in order of complexity) will be used.

Evaluation Metrics

It was first intended to use k-fold cross-validation for the model's evaluation and selection; however, since this project deals with time series, techniques such as standard k-fold cross validation may not provide the best results given the intrinsic nature of time dependency on time series (https://stats.stackexchange.com/questions/14099/using-k-fold-cross-validation-for-time-series-model-selection, https://blog.insightdatascience.com/whats-wrong-with-my-time-series-model-validation-without-a-hold-out-set-94151d38cf5b). Because of this, the author will use a cross-validation technique specific for time series in the Sklearn library (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html)

Given the classification requirement of this project and its domain, historically many evaluation metrics have been used for measuring prediction's success including accuracy and F-Scores. More than half of the previous research studies used accuracy as evaluation metric, however, accuracy has predictive power on only one (the positive) of the possible choices (Fortuny et all, 2014). Another evaluation metric used is the Area Under the Curve of the Receiving Operating Curve (ROC) which allows the evaluation of a model when the target labels have a skewed distribution (Fortuny et all, 2014). Other metrics can also be used such as trading simulation and Sharpe ratio, but this project will focus on ROC.

A ROC Curve is a that shows the ability of a binary classifier when the discrimination threshold is varied or, in other words, it plots the True Positive Rate (TPR) against the False Positive Rate (FPR) along various discrimination thresholds. As a brief explanation, look at the example below where a disease is being tested. The vertical black line will work as the discriminant threshold where: TN = True Negatives, FN = False Negatives, FP = False Positives, TP = True Positives.

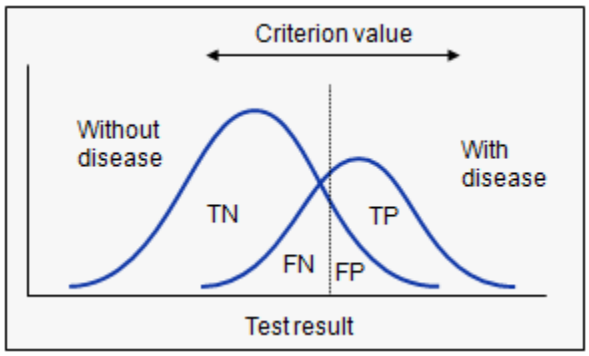

Figure 1 Test Results Example

And TN, TP, FN and FP are calculated as (*Columns* are the reality and *Rows* are the results of the test/classifier):

	Disease				
Test	**Present**	**n**	**Absent**	**n**	**Total**
Positive	True Positive (TP)	*a*	False Positive (FP)	*c*	*a + c*
Negative	False Negative (FN)	*b*	True Negative (TN)	*d*	*b + d*
Total		*a + b*		*c + d*	

Figure 2 Classification of Test Result

Where Sensitivity and Specificity are defined as shown below:

Sensitivity	$\frac{a}{a+b}$	**Specificity**	$\frac{d}{c+d}$
Positive Likelihood Ratio	$\frac{\text{Sensitivity}}{1-\text{Specificity}}$	**Negative Likelihood Ratio**	$\frac{1-\text{Sensitivity}}{\text{Specificity}}$
Positive Predictive Value	$\frac{a}{a+c}$	**Negative Predictive Value**	$\frac{d}{b+d}$

Figure 3 Sensitivity, Specificity and Probability calculations

Basically, the ROC Curve plots on the vertical axis the TPR (Sensitivity) and in the horizontal axis the FPR (1 – Specificity) for each possible discrimination threshold to obtain a chart similar to the one below. In this chart, the dotted line means that results are basically by "luck" (hence, a AUC = 0.5); the blue line shows a real case with $0.5 < AUC < 1$ and the best case scenario would be when the blue line moves directly on top of the Y-Axis (hence, FPR = 0 for every threshold tested and creating a AUC = 1).

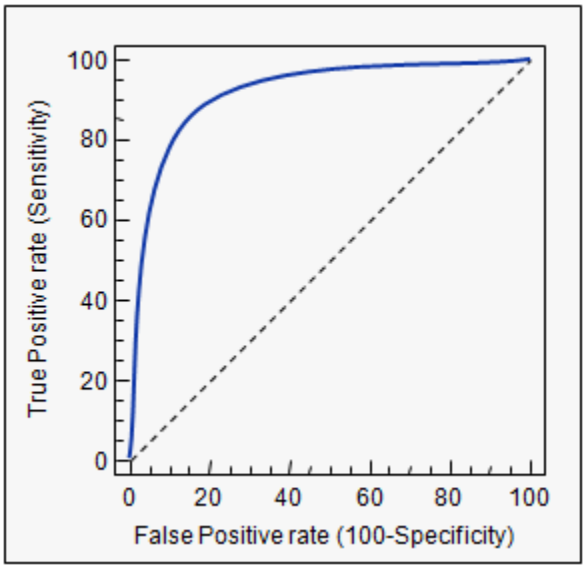

Figure 4 ROC Curve Example

II. Analysis

Datasets and Inputs (General View):

Basic stock market information is readily available for consumers in websites such as https://finance.yahoo.com/, www.google.com/finance or https://www.quandl.com/ to mention a few. There are also curated data sets in sites such as Kaggle.com. In particular, this project will utilize the Huge Stock Dataset as it already includes not only one stock, but also ETFs data as far back as 1999 depending on the selected stock.

Moreover, for this project the following datasets will be used as part of the analysis for each chosen stock/ETF and Index:

- "Daily and Intraday Stock Price Data" from Kaggle.com (https://www.kaggle.com/borismarjanovic/daily-and-intraday-stock-price-data). It contains full historical daily (an intraday) price and volume data for all U.S.-based stocks and ETFs trading on the NYSE, NASDAQ, and NYSE MKT (Date, Open, High, Low, Close, Volume, OpenInt). This data set is adjusted for splits and dividends. Specifically, the daily data will be used.
- NASDAQ-100 Technology Index (NDXT) https://finance.yahoo.com/quote/%5ENDXT/history?p=^NDXT
- Volatility Index (VIX) http://www.cboe.com/products/vix-index-volatility/vix-options-and-futures
    - 2004 through Jan 19, 2018
- NASDAQ Volatility Index (VXN) http://www.cboe.com/products/vix-index-volatility/volatility-on-stock-indexes/cboe-nasdaq-100-volatility-index-vxn
    - 2001 through Jan 19, 2018
- Engineered Features (as per definition on Madge, 2015):
    - Up_Down: defines whether the stock increased or decreased its price each day
    - ETF's Price Momentum
    - ETF's Price Volatility
    - Sector's Momentum - NDXT
    - Sector's Volatility - NDXT
    - Market's Momentum - VIX
    - Market's Volatility - VIX
    - Market's Momentum - VXN
    - Market's Volatility - VXN

These datasets will be combined into one dataframe for easy training and testing afterwards. This is, data from VIX, NXDT and VXN data sets will be brought in into one dataframe.

Data Exploration and Visualization

The author of this project is not experienced on investing techniques, hence, a "safer" approach will be used; this is, this project will be based in the selection of ETFs from a set of 49 technology sector ETFs based on their current performance given on the following link (as of 1/25/18): 49 Best ETFs. From these 49 ETFs a subset of 13 was selected because:

- ETFs were available in "Huge dataset"
- Data available for ALL ETFs and market indexes as far back as possible (2/23/2006)

The final combined dataset used for data exploration contains the following:

- 13 of the 49 best performing ETFs
- Daily difference calculation for each Close Price column as well as for each Market and volatility Index column. These differences (a.k.a delta values in the dataset) considered the grouping of each "Ticker" (ETF) to avoid creating outlier delta values coming from values in other groups
- Standardized values for each "close price"/Index value as well as for the Delta calculated features. These will allow to understand what to expect in terms of the balance of the features
- Labeled data for the variable to be predicted (as well as for other market indexes)

The combined data set created show the following characteristics:

1) The dataset contains 38921 rows for 13 "Tickers" with a data range from 2/23/2016 through 11/10/17.
2) Delta Price values shows a normal distribution

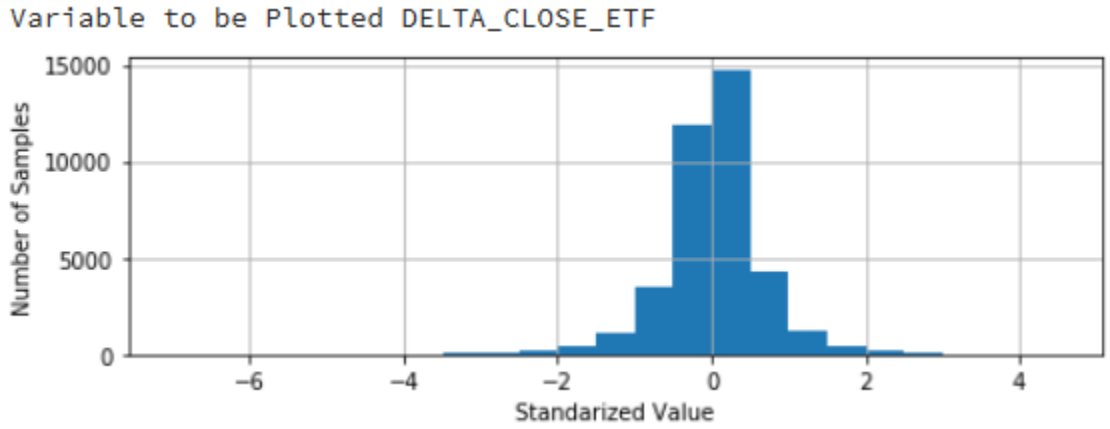

Figure 5 Distribution of Daily ETF Changes

3) Delta Price values (target feature) for *all* 13 selected "tickers" have a tendency to the "up" side with a total average of around 54.66% of the sample.

Ticker	Down	Up
IGM	44.76%	55.24%
IGN	46.73%	53.27%
IGV	45.37%	54.63%
IXN	44.63%	55.37%
IYW	44.46%	55.54%
PSI	45.66%	54.34%
PSJ	45.30%	54.70%
PXQ	45.17%	54.83%
SMH	47.00%	53.00%
SOXX	46.51%	53.49%
VGT	43.99%	56.01%
XLK	43.58%	56.42%
XSD	46.32%	53.68%
Grand Total	**45.34%**	**54.66%**

4) Daily Returns distribution is shown below. It is observed that the ETF group of tickers has a slight tendency to the “Up” side.

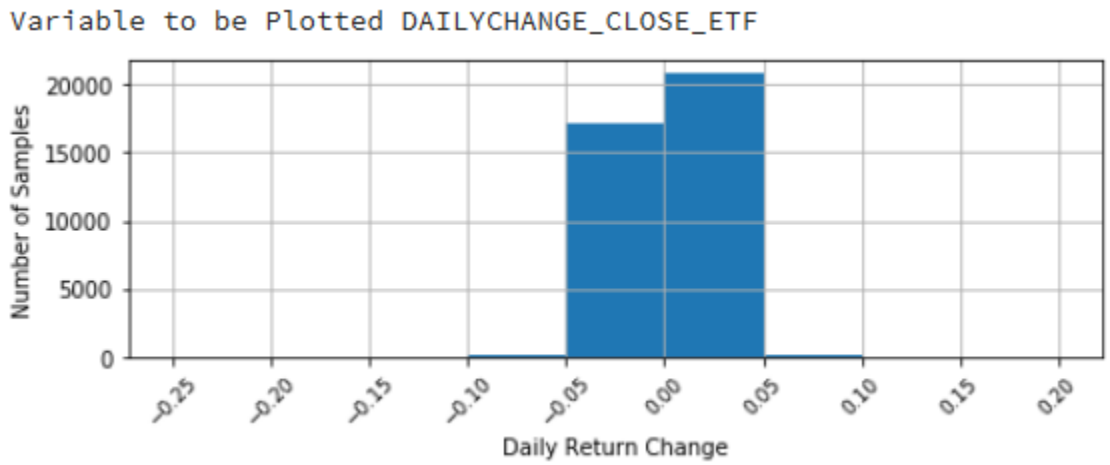

Figure 6 Distribution of Daily Returns Changes on ETFs

5) On the other hand, other market indexes show a daily return tendency to the low side but for the NDXT Index as shown below:

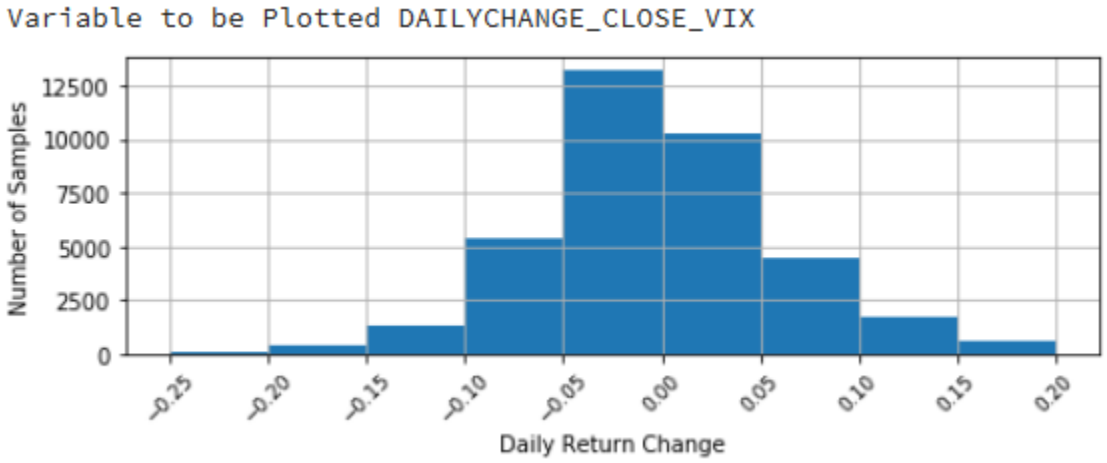

Figure 7 Distribution of Daily Returns Changes on VIX

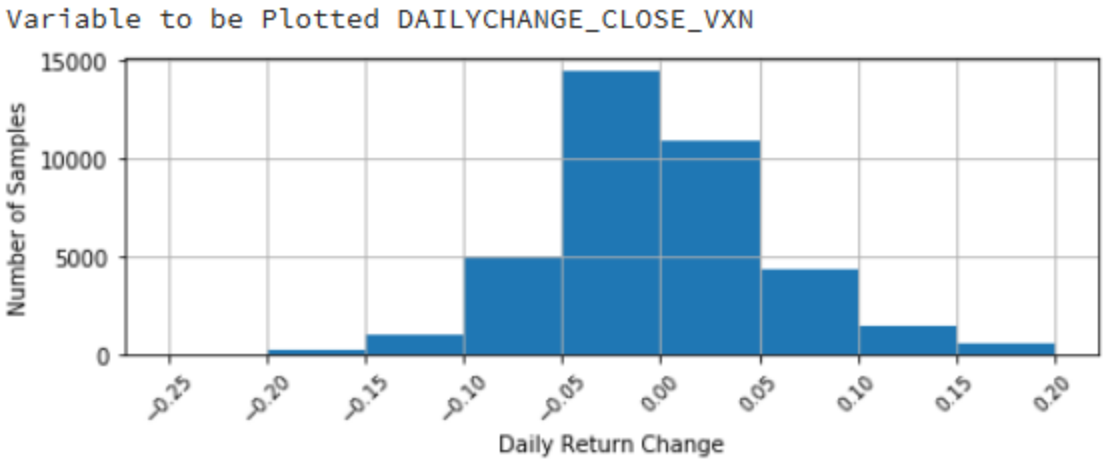

Figure 8 Distribution of Daily Returns Changes on VXN

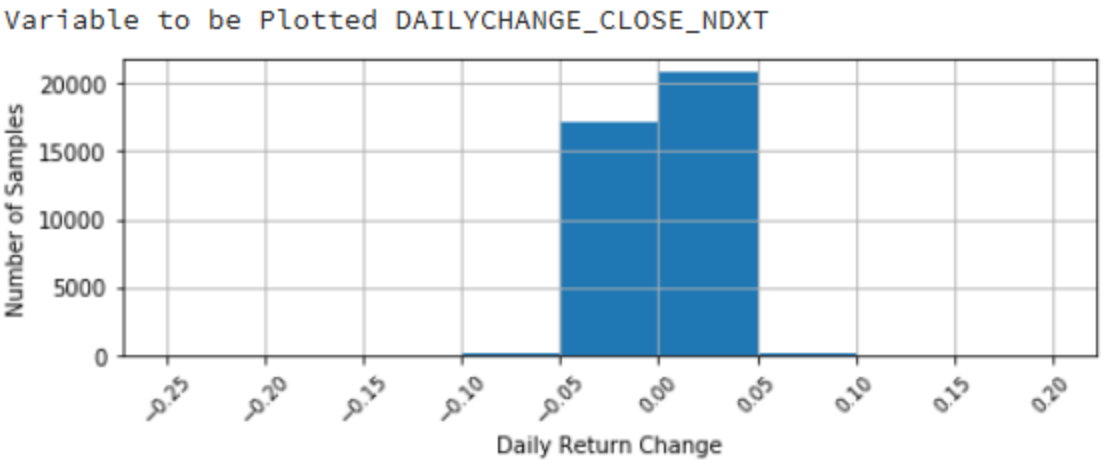

Figure 9 Distribution of Daily Returns Changes on NDXT Market Index

From an ETF and Market Index point of view, the summary statistics for the source features:

	close	vix_close	vxn_close	ndxt_close
count	38291.00	38291.00	38291.00	38291.00
mean	48.35	19.46	21.44	1643.56
std	30.47	9.52	8.96	741.25
min	7.40	9.14	10.31	525.92
25%	25.38	13.29	15.81	1094.53
50%	41.92	16.65	18.60	1389.07
75%	59.91	22.53	24.39	2185.03
max	178.90	80.86	80.64	4024.74

From these a few characteristics can be exposed:

- The NDXT index's value range is 34x and 84x larger than for the rest of the indexes and EFTs depending on the feature (based on averages). This would indicate that scaling features may be a good practice if using these in their raw form.

After engineering the basic features to understand daily changes in ticker price and return, their summary statistics are shown below:

	delta_close_etf	dailychange_close_etf	delta_close_vix	dailychange_close_vix	delta_close_vxn	dailychange_close_vxn	delta_close_ndxt	dailychange_close_ndxt
count	38291.00	3.83e+04	3.83e+04	3.83e+04	3.83e+04	3.83e+04	38291.00	3.83e+04
mean	0.03	4.91e-04	-2.00e-04	2.74e-03	-1.05e-04	2.02e-03	1.01	5.79e-04
std	0.65	1.50e-02	1.85e+00	7.62e-02	1.67e+00	6.50e-02	20.80	1.51e-02
min	-6.45	-1.10e-01	-1.74e+01	-2.96e-01	-1.30e+01	-2.69e-01	-129.84	-9.89e-02
25%	-0.26	-6.64e-03	-7.10e-01	-3.97e-02	-7.00e-01	-3.50e-02	-9.44	-6.35e-03
50%	0.04	1.04e-03	-8.00e-02	-5.28e-03	-9.00e-02	-4.95e-03	1.74	1.16e-03
75%	0.33	8.06e-03	5.70e-01	3.50e-02	6.00e-01	3.20e-02	12.67	7.95e-03
max	4.74	1.39e-01	1.65e+01	6.42e-01	1.28e+01	4.37e-01	100.77	1.25e-01

From these it can be seen:

- NDXT is the only index with a positive daily return average. Given its larger raw values, daily returns are much lower
- Delta_Close_ETF distribution shows a slightly unbalanced (towards the up side) distribution with 10.85% of samples potentially considered as outliers (given outliers are samples that fall outside the following range mean +/- 1.5*Std Deviation)

delta_close_etf	Contribution (%)
-6.825--5.85	0.01%
-5.85--4.875	0.01%
-4.875--3.9	0.05%
-3.9--2.925	0.19%
-2.925--1.95	0.68%
-1.95--0.975	4.39%
-0.975--0	40.01%
0-0.975	49.13%
0.975-1.95	4.80%
1.95-2.925	0.60%
2.925-3.9	0.10%
3.9-4.875	0.02%
Grand Total	**100.00%**

Algorithms and Techniques

Multiple Algorithms will be evaluated for this project:

- Decision Tress,
- Gaussian Naive Bayes,
- Support Vector Machine
- Neural Networks and
- Random Forest will be used as benchmark model

These models were selected considering what it is known about the data:

- Target variable is a binary outcome
- Target variable is slightly skewed/unbalanced
- Feature set is small
- Feature set is composed of numerical values:
    - All features have been standardized
    - Daily Returns were Log10 transformed
- Data set is not large (38k rows for 13 ETFs, hence, around 2900+ rows per ETF for training)
- Some outliers are present

Other important items considered for this selection were the limitations for this project, i.e. calculation power, memory and timelines for delivery; in this sense, models with "fast" training times were prioritized, yet including other "slow" models so both types could be assessed and contrasted. All selected models can be applied to a classification problem/task as the one in hands.

Below a summary of the strengths and weaknesses for these selected given the characteristics of the dataset and the problem:

Table 1 Machine Learning Algorithm Selection

Algorithm 1 2	Strengths	Weaknesses
Decision Tress 1 2 3	- Intuitive and Easy to Explain - Simple and easy to implement - Accounts for variable interaction - Not influenced by outliers - Does not make assumptions of space distribution - Training/Testing Speed is "fast"	- Highly biased (prompt to overfitting) -> CV helps to diminish this - May not perform well with new unseen data
Gaussian Naive Bayes	- Intuitive and Easy to Explain - Simple and easy to implement - Tends to not overfit - Training/Testing Speed is "fast"	-Assumes features are conditionally independent, however, it performs fairly well even with this does not hold true -May not work as well in large datasets -Does not learn interaction among features
Support Vector Machine	-Handles non-linear decision boundaries -Handles non-linear feature interactions -Robust to outliers	- NOT Intuitive nor Easy to Explain -Not efficient for large number of interactions -Finding the right Kernel may be a challenge - Training/Testing Speed is "slow" -May not work well when data is overlapping
Neural Networks 1	- Handles non-linear decision boundaries - Handles irrelevant features - Learn and model non-linear and complex relationships - Accounts for variable interaction - High accuracy - Works well in unseen data	- NOT Intuitive nor Easy to Explain -Training/Testing Speed is "slow" -Requires lots of parameter tuning
Random Forest 1 (out-of-the-box benchmark model)	- Accounts for variable interaction - High accuracy - Reduce overfitting and variance present in Decision trees	- NOT Intuitive nor Easy to Explain -Training/Testing Speed is "slow"

After considering the characteristics above along with recommendations from Microsoft Azure and SKLearn estimator selection cheat sheets, the selected algorithms were considered a good sample of the algorithm's toolset available in Machine Learning (ranging from a simple Gaussian Naive Bayes to a Neural Network algorithm) that will help to meet the goals of this project.

Other algorithms were considered (QuadraticDiscriminantAnalysis, GaussianProcessClassifier, GaussianProcessClassifier), however, during preliminary testing they did not show great prediction advantages as compared to the ones above and some of them showed long training timeframes, hence, in the interest of time for this project, they were dropped from the evaluation list.

These algorithms will be evaluated on each ETF ticker and set of simulation parameters (ticker, forecast horizon -m- and rolling timeframes -n1 and n2-).

As a Cross-Validation technique, given that this is a time series problem, the TimeSeriesSplit function from SKLearn will be used. This technique allows to cross-validate the training process while respecting the time series hierarchy of the data.

For the training process, and to optimize the hyper parameter selection for each of these algorithms, GridSearchCV will be used. As an initial stage, a large amount of hyper parameter options was tested as preliminary efforts to evaluate how they affected training time as well as to understand which hyper parameters were used the most; this allowed to define a smaller selection of hyper-parameters for the final run of the prediction and testing exercise. Below is presented the final list of hyper-parameters used for each algorithm.

Algorithm	Hyper-parameter to be assessed
SVC	'kernel':('linear', 'sigmoid'), 'random_state':[1975], 'C':[0.0001,.025,1], 'gamma':[0.0001,.025,1,10]
DecisionTreeClassifier	'max_depth':[100,10,5], 'max_features':('log2','sqrt'),'random_state':[1975]
RandomForest	As-Is (no changes to be used as benchmark)
Neural Network	'activation':['relu','logistic'],'solver':('sgd', 'adam'), 'alpha':[0.0001, 1,10]

Benchmark Model

Shleifer and Summers (1990) defined stock predictions as a "random walk". This basically means that any classification prediction made will have an expected average of 50% of success, however, from the results obtained in the data exploration section above, it is known the dataset is not perfectly balanced, hence, a purely naïve approach (where the benchmark is assumed as 50%) may not provide the best results. Therefore, a machine learning approach will be used, this is, the benchmark model will be defined as an out-of-the-box Random Forest on the project data.

By using this approach, a like-for-like comparison can be made between the Random Forest (i.e. benchmark) and the trained model(s) of this project.

III. Methodology

Data Preprocessing

The Preprocessing stage is executed for each training iteration and it is as follows:

- Dataset's output label's feature is shifted by "m" days to accommodate for a prediction "m" days in the future; along with this, the last "m" days are dropped from the dataset since they now have "NaN" values on them
- Output labels are encoded using "LabelEncoder". It is a binary class label
- Given that this project aims to analyze multiple forecasting (m) and moving average (n2 for ETF and n1 for Indexes), engineered features are calculated:
    - ETF's Price Momentum
    - ETF's Price Volatility
    - Sector's Momentum - NDXT
    - Sector's Volatility - NDXT
    - VIX Index's Momentum
    - VIX Index's Volatility
    - VXN Index's Momentum
    - VXN Index's Volatility
- Given n1 and n2 for rolling averages the first "max(n1, n2)+1" rows are eliminated since those present "NaN" values in the new calculated features. With this, the dataset will include only relevant and valid data for the training exercise
- Because the dataset origin is a dataframe, it is converted to a Numpy array so other calculations are simplified
- Dataset is broken into features set (X_All) and label set (Y_All) to facilitate training process
- Ticker Daily Return Values are transformed to LOG10s [numpy.log10(x+1)]. Even though the data is very close to normal, this will help to smooth out any outlier data points as well
- All data points from the explanatory feature set are scaled using "StandardScaler" to ensure all classifier algorithms will perform as expected by providing a dataset with zero mean and unitary variance. This is particularly important for Support Vector Machine Algorithm

Implementation

The implementation of the entire project followed the next steps:

- Data Collection Phase:
    - Read individual stock/ETF files and combine them in a unique file
    - Read individual Index files (VIX, VXN and NDXT) and merge them with the ETF dataset file
    - Filter the combined dataset to include only those rows where there's available data for ALL input features
- Data Exploration Phase to explore and understand the dataset
    - Created "delta price" columns
    - Created Standardize columns
    - Created Label Up/Down feature(s)
    - Generated summary tables to understand statistics of the dataset for ETF ticker groups and features

    - Generated distribution charts to understand overall behavior of ETFs tickers, volatility index and market indexes
- Training and Testing Phase:
    - A nested series of FOR loops for each m, n1, n2, ticker and classifier variable set is executed. Includes:
        - Filter only "Close prices" and daily returns features for ETF, volatility and market indexes from original combined dataset
        - Preprocess the data (as explained in the previous section)
        - Create engineered features: momentum and volatility for ETF and indexes for each ticker group, n, n1 and n2 variable set combination
        - Dataset is separated in Training and Testing set based on a 70% training size
        - Create time series cross-validator using TimeSeriesSplit (3 splits)
        - If the classifier possesses parameters to optimize, a GridSearchCV object is created. It receives as parameters: the estimator, time series validator and parameter grid set to be analyzed
        - "Fit" method is called to perform training in the classifier
        - Calculate:
            - Predicted Labels (Y_pred)
            - Accuracy score given Testing's set and Y_pred
            - AUC_ROC score given Testing's set and Y_pred
            - Get best grid parameter set with which training ended
        - Record all results in a data frame for later comparison and analysis

From the entire end to end process above, the most time-consuming piece was the gathering of the data (including to define a valid source for it) as well as its exploration which accounted for at least 70 % of the time of the project.

The author of this project is more familiar programming in "R Programming" than in Python which brought challenges along the coding process either because 1) syntax is different between the two languages, and 2) because of the subtle (but important) structure differences between them; for instance, basic actions such as:

- Slicing, or splicing represented a challenge often times because of the wealth of different methods that Python (with Pandas and Numpy libraries) offer to execute the same task (i.e. methods *.iloc* and *.loc* to slice dataframes)
- Adding columns or applying functions to these columns
- Variable assignment and references in memory (in Python assigning a variable to another means a reference to the original variable and not a copy of the original value into the second one)
- Pandas *Describe* function compared to *Summary* function in R Programming provided less information for categorical features, hence, needed workarounds to obtain similar information (Pandas *Crosstab*)
- Creating Summary tables using Pandas *Crosstab* function became a challenge because it was needed in a group basis for ETF Tickers

A key lesson learned as part of this project is related to the use of Machine Learning algorithms for Time Series. Since this project is based on time series (which were not taught along the MLND), this added a higher level of complexity and research to identify how to apply cross-validation techniques and hyper parameter optimization to this type of problems. Initially it was used *TimeSeriesSplit* for a step by step (using a FOR loop) cross-validation process; later, during the hyper parameter optimization research it was found that a simplification to this process was to directly pass the timeseries cross-validator object to the *GridSearchCV* object when it is created. Initial tests on the number of splits for the TimeSeries object did not show prediction advantages when increasing the number of splits, hence, it was decided to keep the number of splits as three(3).

As part of the training process, some of the algorithms either dot not have hyper parameters or it was the benchmark model, hence, those classifiers did not go through the *GridSearhCV* step but they went directly into the training (*fit*) step; this is because GridSearchCV would raise an error due to lack of parameters passed (this applied to *RandomForest* and *Gaussian Naive Bayes*).

A few functions were created to facilitate or simplify the coding process, for instance, *EvaluateROC* that given True and Predicted Labels, it returns the *trapezoidal AUC* and the *AUC_ROC* scores. From these, the *AUC_ROC* score is the one used for later analysis. This function also has the capability to plot ROC Curve if the "plot" parameter is set to *True*. The parameter plot was heavily used during initial stages of coding, testing the code and its preliminary results; it has been set to *false* for the final training execution to save processing time during the training process. In initial testing stages, it was found that *trapezoidal AUC* did not provide significative differences with *AUC_ROC,* hence, the latter was selected for this analysis.

Similarly, two other functions were created for running the training process (they could in reality be converted into one, but for easy differentiation, they were kept as two different ones). These are: *RunClassificationPredictions* and *RunGridSearchClassification*. They are the ones in charge of training or fitting the classifier given the input data, returning a vector with predicted values (so it can be used for model evaluation in later stages), Accuracy, Trapezoidal AUC and AUC_ROC scores as well as the "best" hyper-parameters as a result of the training.

As part of the looping process around all the variables, a dataframe is created that compiles the results of each training exercise (one row per training run), it specifically carries for each training run:

- Target ETF or Ticker symbol
- Moving Average Horizon for Indexes (n1)
- Moving Average Horizon for ETF (n2)
- Forecast Horizon (m)
- Name of Classifier/Estimator used
- Accuracy Score
- Trapezoidal AUD Score
- AUC_ROC Score
- Training Time Duration and
- Parameter set that provided the best results of the training stage

It is important to highlight that this project attempted to do training and predictions for many combinations of input variables (total of 9750). <u>One full run (along with optimizing hyper parameter</u>

selections) takes around 10hrs of training time in a Intel Core i7 6820 @ 2.7GHz and 32GB RAM with Windows 7-64bits.

Refinement

Several attempts for refinement were attempted. Given the long-expected running time for a full simulation, several simulations were made at initial stages in order to discard hyper-parameters that may not be useful for training purposes while adding unnecessary processing time; this was important because SVM and Neural Networks could take several minutes just for one training run depending on the number of parameters they may had. After this step, most common parameters were selected and kept for the *GridSearchCV* process. In average, per iteration, Neural networks training took 10 seconds and SVC took 5 seconds followed by Decision Tress with over 3 seconds and RandomForest and Gaussian Naïve Bayes with less than a second.

As explained earlier, the time series validation was initially performed as part of a FOR loop. Along the development process, this portion was improved/optimized by passing the *TimeSeries* cross-validator directly to the *GridSearchCV* object.

The first full training exercise was performed on the un-scaled data and as a mean to improve accuracy, then a full run of training was performed in the log10 and scale data. The results did not provide a high level of improvement, however, the log10 and scale data procedure were kept to improve the performance of the selected models in the presence of new data that might be more skewed. Results will be shown in the Results section of this report.

It is also worth noting that the coding style and implementation changed from the initial exploration to the training/prediction one; the training and testing code is more efficient as well as easy to follow.

IV. Results

Model Evaluation and Validation

It is important to note that this project does not deal with a few models, but with 9750 total models given all the variables that are included for training and prediction. This is:

ID	Variable	Number of Levels
m	Forecast Timeframe	6
n2	ETF Moving Average Timeframe	5
n1	Market and Volatility Index Moving Average Timeframe	5
-	Number of Ticker (i.e. ETFs)	13
-	Number of Evaluate ML Algorithm	5
	Total Iterations/Models	9750

Because of this, and considering that the financial instruments to be predicted are ETFs, then the model selection will target such model that offers the Max AUC_ROC score with forecasting timeframes greater

than 20 days (ETFs are not meant for daily trading but longer periods). Below it is shown the drill down to make this selection:

- Forecast Timeframe (m) greater than 20 days since ETFs are not daily trading instruments. ***Forecast = 90*** days was selected which provided and Avg. AUC_Score = 0.5013

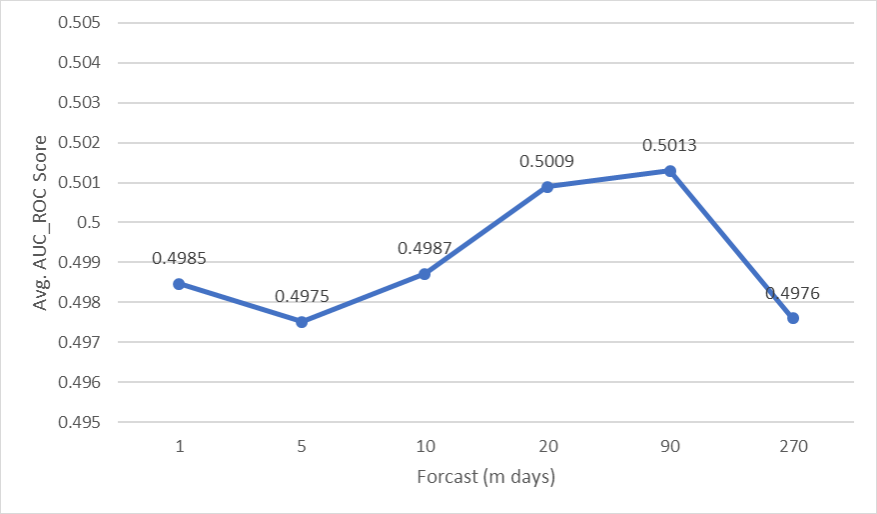

Figure 10 Overall Avg. AUC_ROC Score

- n1 and n2 combination that maximizes AUC_ROC score. ***N1 and n2 are selected as 270 and 90*** respectively with a AUC_Score of 0.505

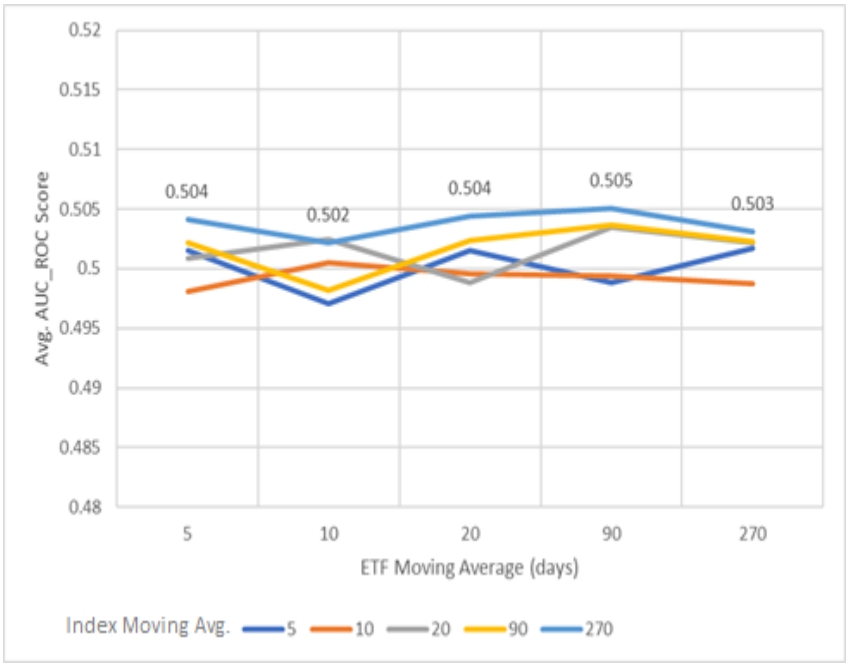

Figure 11 Selection of n1 (Index) and n2 (ETF) by Avg. AUC_ROC Score

- Select the best classifier/estimator given the selections above. Below is shown the final view including the average for all Tickers (13) as well as the Avg. Accuracy score. ***Gaussian Naïve Bayes*** has been selected. It can be seen how the Accuracy score is overconfident on the prediction it makes.

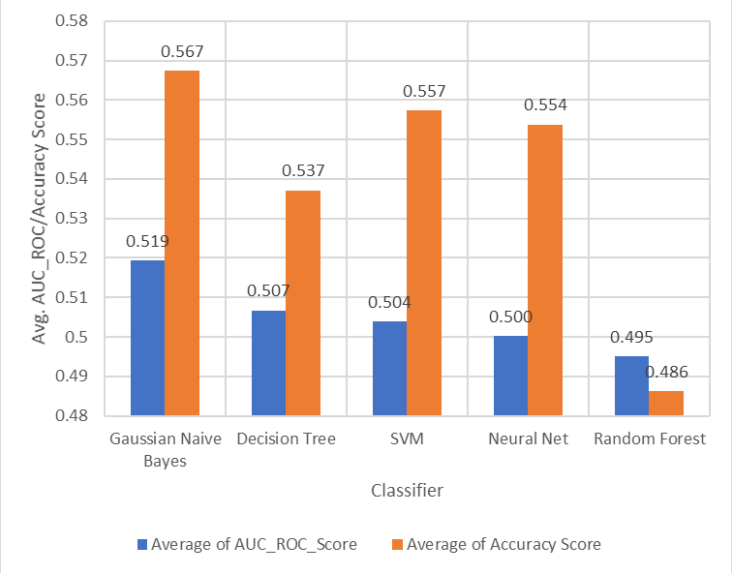

Figure 12 Classifier PErformance Comparison

- After selecting the classifier to be used, now tickers(ETFs) are looked after to define the ticker for which predictions are more accurate. Below a view of the top 5 tickers by Avg. AUC_ROC score. Once again, it is shown how the Accuracy score may be overconfident on its predictions when compared to more robust evaluation metrics such as AUC_ROC score.

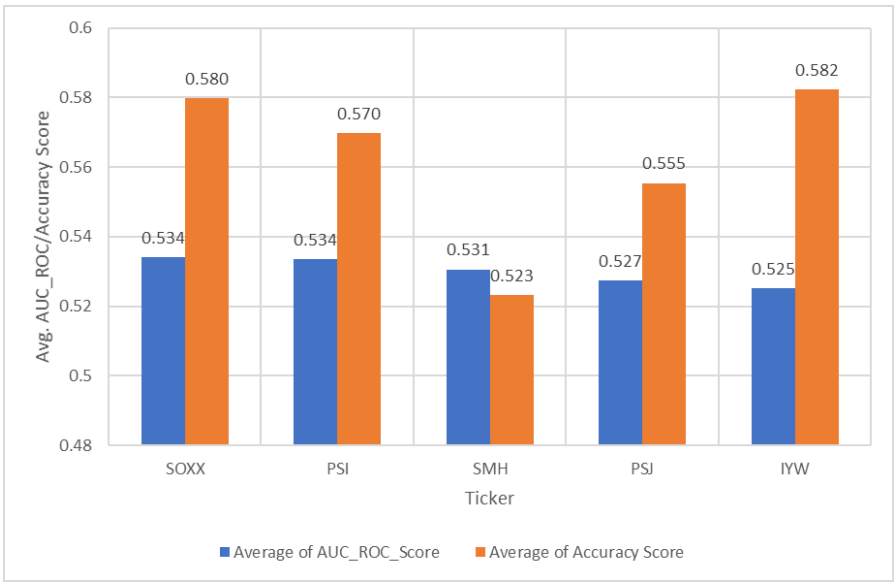

Figure 13 ETF Performance Comparison

To better understand the model(s) results for each ticket/ETF, below is a further drill down into how each ML Algorithm performed for each ticker/ETF. It is interesting to see how each ticker/ETF could potentially be better predicted by a different algorithm. In this case Gaussian Naïve Bayes was a clear top performer for all tickers, however, if the prediction is a specific ETF, then it is worth considering this view to ensure the absolute better ML algorithm has been selected.

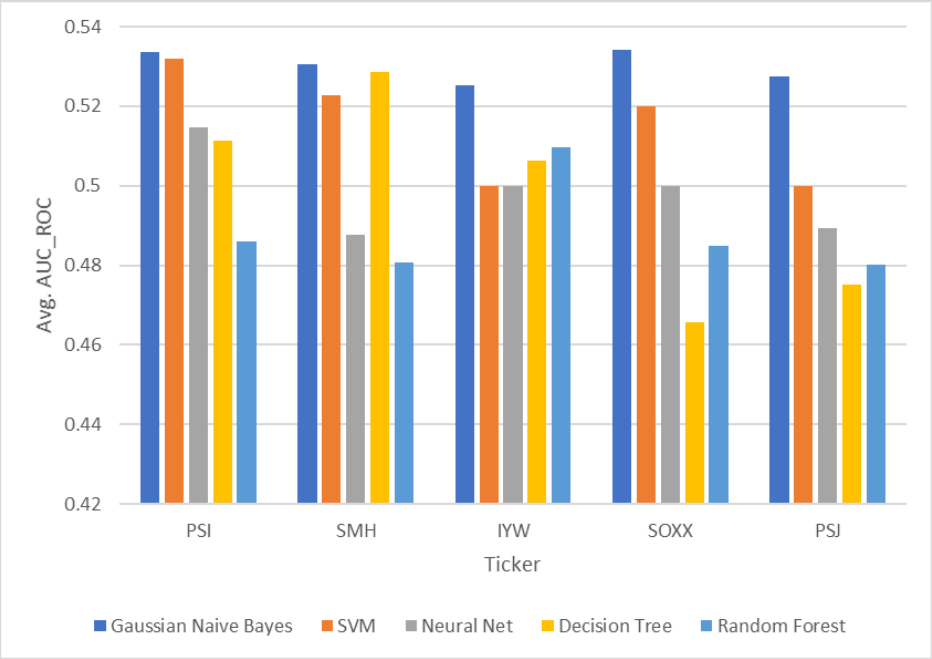

Figure 14 Classifier Performance by ETF

After this analysis, the Gaussian Naïve Bayes algorithm with n1 and n2 as 270 and 90 respectively along with a 90 day forecast horizon would be the parameters for the sought prediction model. Moreover, the best predictions are aligned with the SOXX, PSI, SMH, PSJ and IYW which gives a more specific sense to which ETF (tickers) to invest in.

Comparison with Non-Log10, Non-Scaled dataset

A comparison of the final model's results above with the initial one where the features dataset was not log10 transformed nor scaled shows that there were not advantages in scaling the data. The results shown in the chart below for the non-transformed model depicts the same results for both AUC_ROC and Accuracy scores for all 5 tickers as in the transformed dataset.

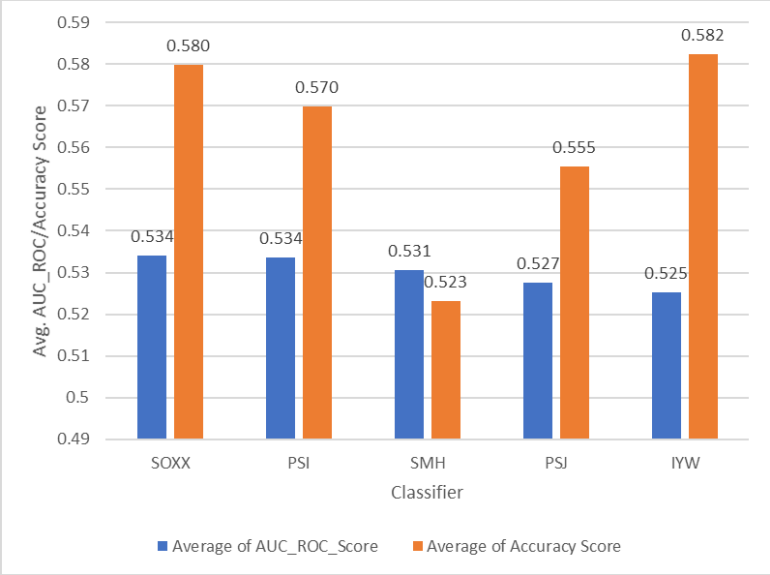

Figure 15 Non-Scaled Performance by ETF

In order to do a more high-level comparison, below is shown an average of the performance for all tickers, n1=270, n2=90 and m=90 values by classifier type comparing the scaled dataset vs the non-scaled one. The table shows very little improvement and where three out of five classifiers lowered their performance.

Table 2 Scaled vs Non-Scaled Performance

	Non-Scaled		Scaled			
Classifier	**Average of AUC_ROC Score**	**Average of Accuracy Score**	**Average of AUC_ROC Score**	**Average of Accuracy Score**	**Delta (%) AUC_ROC**	**Delta (%) Accuracy**
Gaussian Naive Bayes	0.506	0.536	0.519	0.567	2.60%	5.94%
SVM	0.497	0.486	0.504	0.557	1.46%	14.57%
Neural Net	0.502	0.572	0.500	0.554	-0.32%	-3.21%
Random Forest	0.500	0.573	0.495	0.486	-0.94%	-15.20%
Decision Tree	0.519	0.567	0.507	0.537	-2.44%	-5.33%
Grand Total	0.505	0.547	**0.505**	**0.540**	0.06%	-1.21%

While the improvement in prediction accuracy is not great, the author of this report considers that maintaining this transformation in place will allow to smooth out new incoming data into the model, hence, doing a better job generalizing new data for prediction purposes.

Model Sensitivity Analysis

As it was shown on previous sections, different accuracies are obtained for different combinations of the input variables and tested classifiers (total of 9750), hence, for the purposes of looking at a sensitivity analysis, the classifier (Gaussian Naïve Bayes) and forecast timeframe (90 days) will be fixated while other variables are updated/modified. Similarly, for simplicity purposes, results will be based on the average AUC_ROC score of all tickers (ETFs)

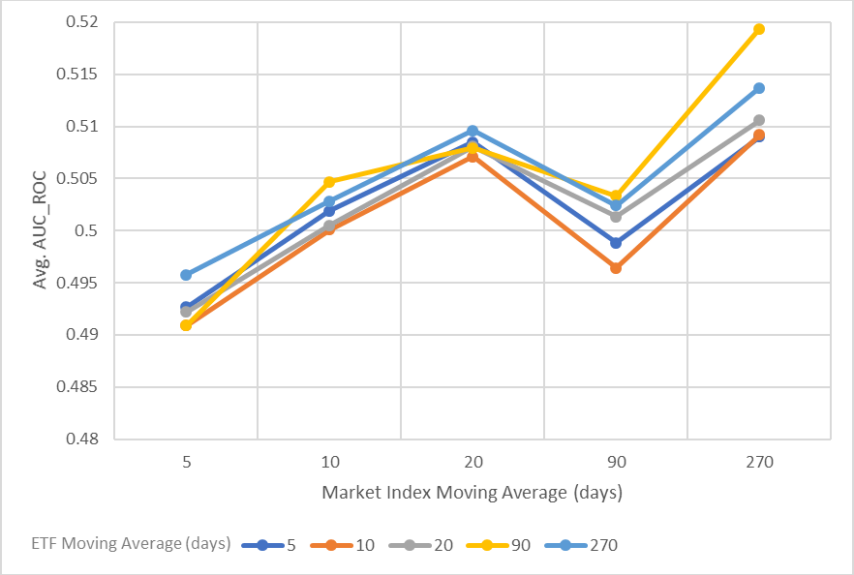

Figure 16 Sensitivity Analysis View

The figure above shows how the accuracy of the model would change given changes on n1 (x-axis) and n2 (y axis). Although n1 and n2 can be greater than 270 business days (1 calendar year in business days), n1 and n2 will be restricted to 270 days. The figure shows how the selected model (n1 = 270 and n2 = 90) is the parameter set that maximizes AUC_ROC. This confirms the selection of the model.

To generate a more detailed view of the model, another round of predictions has been run by fixing m and classifier (90 days and Gaussian Naïve Bayes), and changing n1 and n2 (both) from 1 through 560 in steps of 5 days. Results are shown below.

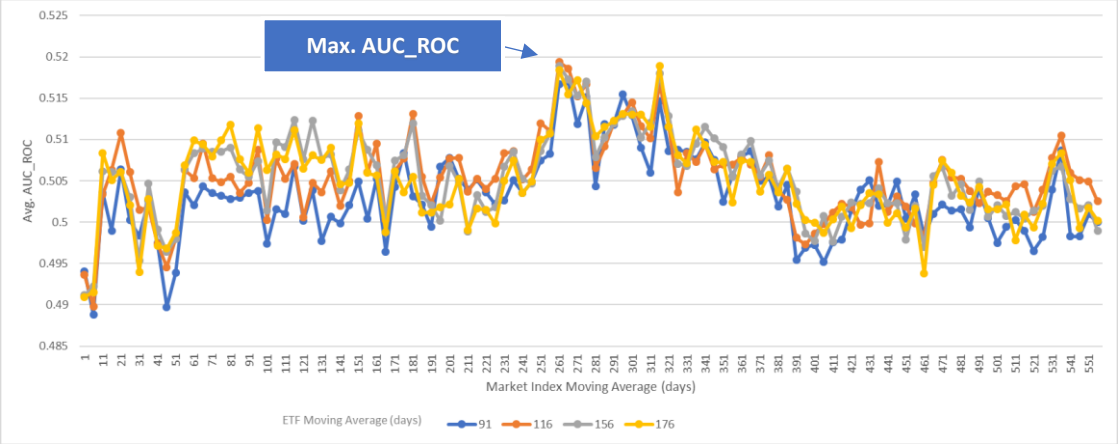

Figure 17 Sensitivity Analysis Detailed View (each 5 days)

It is clear in the chart above how for the selected forecast timeframe and algorithm, the highest prediction rate (AUC_ROC) is found between 261 and 266 days for the Market Index Moving Average, and to discern the best selection for the ETF Moving Average the table below will show a more detailed view of the AUC_ROC score given n1 and n2 (green means lowest, red means highest)

Table 3 "Maximum" Avg. AUC_ROC from Sensitivity Analysis

	ETF Moving Average (days)				
Market Index Moving Average (Days)	**91**	**116**	**156**	**176**	**Grand Total**
256	0.508	0.511	0.511	0.511	0.510
261	0.517	0.519	0.519	0.518	0.518
266	0.517	0.519	0.517	0.515	0.517
271	0.512	0.515	0.515	0.517	0.515
276	0.515	0.517	0.517	0.514	0.516
281	0.504	0.507	0.508	0.510	0.507
Grand Total	**0.513**	**0.514**	**0.514**	**0.515**	**0.514**

Given the information above, to maximize the predictive power of the algorithm the ETF moving average may be changed/set to 116 days; however, in practical terms the predictive accuracy improves from 0.517 to 0.519 (~ +3.8%) and raises the question, is it worth? This really depends on how much effort and resources are needed to reach that point. If this decision is made solely on what's presented on this report and algorithm, it could be argued that it is worth changing our model since it does not increase resource needs and it allows a few percentage points of advantage in the stock market which will make a difference in capital gains.

Justification

To compare the performance of the final prediction model to the benchmark (out-of-the-box Random Forest), below a chart comparing both performances for each of the tickers:

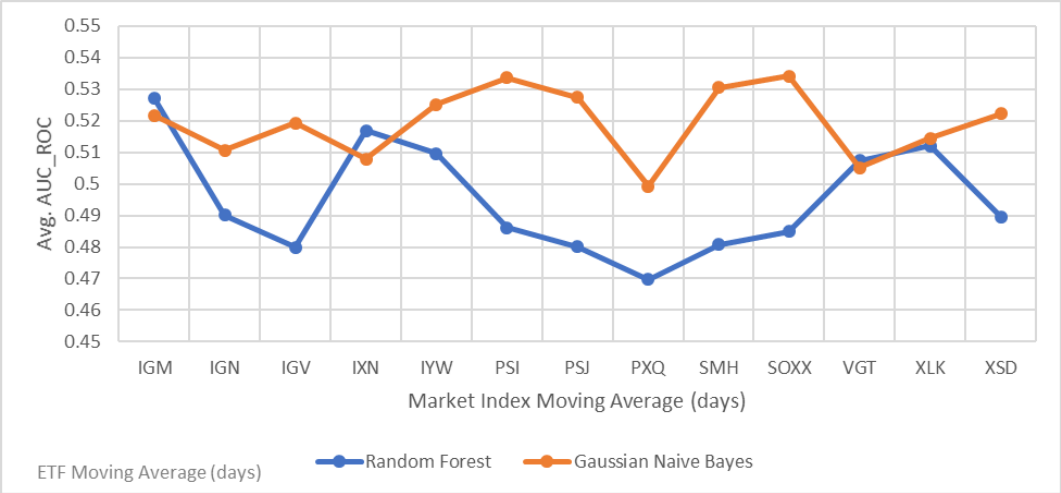

Figure 18 Benchmark vs Gaussian Naive Bayes Performance

It is easily noticeable that the GNB algorithm dominates in 10 out of the 13 tickers; also, the GNB algorithm shows AUC_ROC scores > 0.5 in 12 out of the 13 samples (versus 5 out of 13 for Random Forest). Looking at some of the basic descriptive statistics of the results shown in the table below other characteristics are apparent:

- GNB shows a 5% Avg better performance
- GNB provides more predictions over 0.5 AUC_ROC (by Median)
- Variability of the predictions is lower on GNB by 36%
- Range of predictions is lower by 39%

Table 4 Descriptive Statistics Benchmark vs Gaussian Naive Bayes

Random Forest (RF)	*AUC_ROC*	*Gaussian Naive Bayes (GNB)*	*AUC_ROC*	*Delta (%)*
Mean	0.495	Mean	0.519	**5%**
Standard Error	0.005	Standard Error	0.003	**-36%**
Median	0.490	Median	0.522	**7%**
Standard Deviation	0.018	Standard Deviation	0.011	**-36%**
Range	0.058	Range	0.035	**-39%**
Minimum	0.470	Minimum	0.499	**6%**
Maximum	0.527	Maximum	0.534	**1%**
Count	13.000	Count	13.000	**0%**

These are indicatives of a more stable model, however, are these differences statistically significant?

To test statistical significance, a Paired T-test will be run on both samples (though this sample is considered small). Below its results.

Table 5 Paired t-Test Benchmark vs Gaussian Naive Bayes

t-Test: Paired Two Sample for Means		
	Random Forest (RF)	***Gaussian Naive Bayes (GNB)***
Mean	0.49	0.52
Variance	0.00	0.00
Observations	13.00	13.00
Pearson Correlation	-0.16	
Hypothesized Mean Difference	**0.00**	
df	12.00	
t Stat	-3.95	
P(T<=t) one-tail	0.0010	
t Critical one-tail	1.7823	
P(T<=t) two-tail	0.0019	
t Critical two-tail	2.1788	

Two important results can be drawn from this table:

1) The mean differences are statistically different given a two-tailed p-value of 0.0019 (p-value < 0.05) which means the difference is not due to chance alone
2) Pearson Correlation shows -0.15 which basically means, there is not a strong correlation between both accuracy groups (RF vs GNB)

After proving that these results are statistically significant, the next questions to answer is: are they practically significant? In this particular case, understanding that having an accuracy of below 0.5 means to likely "lose money" then it'd be clear that an AUC_ROC score of 0.52 (Avg) will certainly bring practical advantages. Moreover, from the table below it is seen that, assuming a Gaussian distribution of the samples, more than 33% of the results are likely to stay above 0.5, hence, increasing the likelihood for "making money".

Table 6 Avg. Model Expected Variation given its Standard Deviation

Avg. GNB AUC_ROC	# Std Dev	AUC_ROC Low Interval	AUC_ROC High Interval
0.519	1	0.508	0.531
	2	0.497	0.542
	3	0.486	0.553

V. Conclusion

Free-Form Visualization

One view that is helpful to evaluate the model/classifier selection made is shown below in the form of a BoxPlot chart for each of the n1 and n2 combinations given the fixation on forecast (m) at 90 days. This view allows to have a more objective and graphic sense of the variability and performance of each of the models in the dataset; more specifically, a visual inspection raises a few observations:

- Random Forest and Decision Trees present the highest variability
- Variability in most cases involves falling below 0.5 AUC_ROC score
- Gaussian Naïve Bayes @ Market Index Moving Average = 270 is the only algorithm which shows low variability and interquantile samples greater than 0.5 for any ETF Moving average selected. This brings more confidence into the predictive power of the selected model.

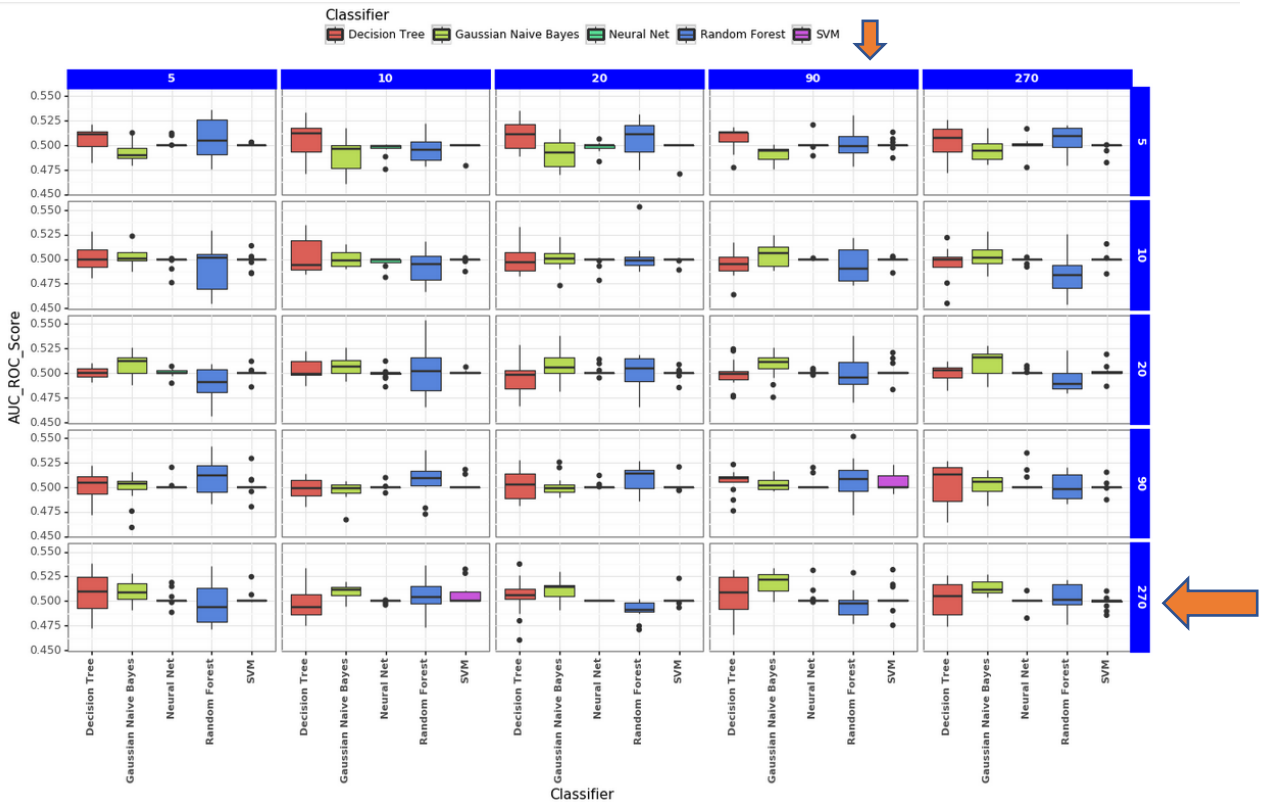

Figure 19 Classifier Performance by n1 and n2 and m = 90

This project also intended to use as feature volatility indexes to the model because much of academic research focuses on predictions using market indexes only which are generally positively correlated to the ETF and/or security; using volatility would allow to have a "second set of eyes" or second "opinion" on the input data set with the intent to make more robust predictions. As it can see below, the NDXT market index correlates with the ETFs' performance and VIX and VXN volatility indexes are negatively correlated.

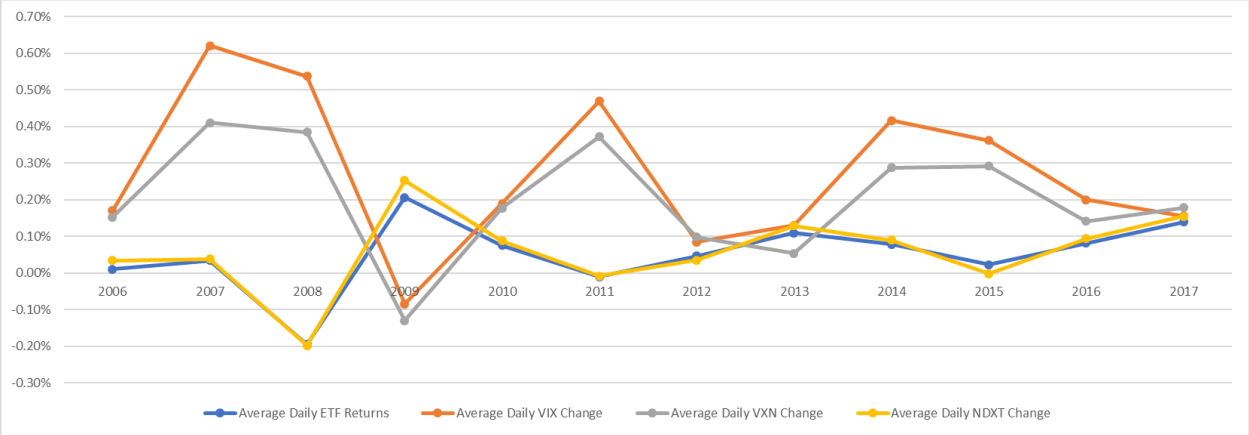

Figure 20 Daily Changes correlation among ETF, Market Index and Volatility Indexes

Furthermore, looking at the average close prices for ETF, market and volatility indexes, it is clearly seen how ETFs and market have enjoyed a "up" market for several years presumably thanks to confidence levels in the market given that volatility indexes have decreased over time as well.

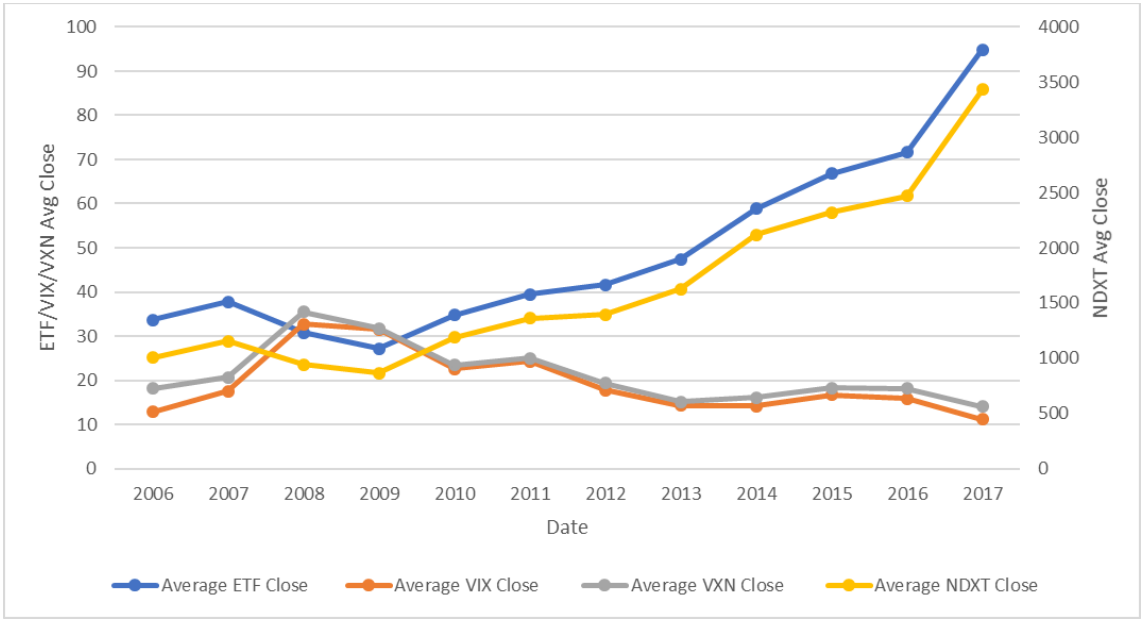

Figure 21 Correlation among ETF, Market Index and Volatility Indexes

Reflection

The nature of the problem of this project calls for a Supervised Machine Learning approach, hence, supervised ML techniques were used. This project intended to do directional stock predictions on a series of ETFs based on financial technical analysis techniques; as engineered features, the project generated momentum and volatility of the target feature, market index and market volatility indexes. Momentum and Volatility are features that can be calculated for different timeframes, hence, as a mean to select the best possible model for predictions, these features were calculated for multiple moving average timeframes (between 1 and 270 business days) and, for each of these combinations, train/test for five different machine learning algorithms and ETF ticker. In a sense, the project itself operated as a sensitivity analysis for selecting not only the best algorithm for prediction but also the best combination of variables to achieve maximum accuracy (AUC_ROC) results.

Below is shown a process flow chart that depicts how this project and its coding was implemented. Three clear stages are shown:

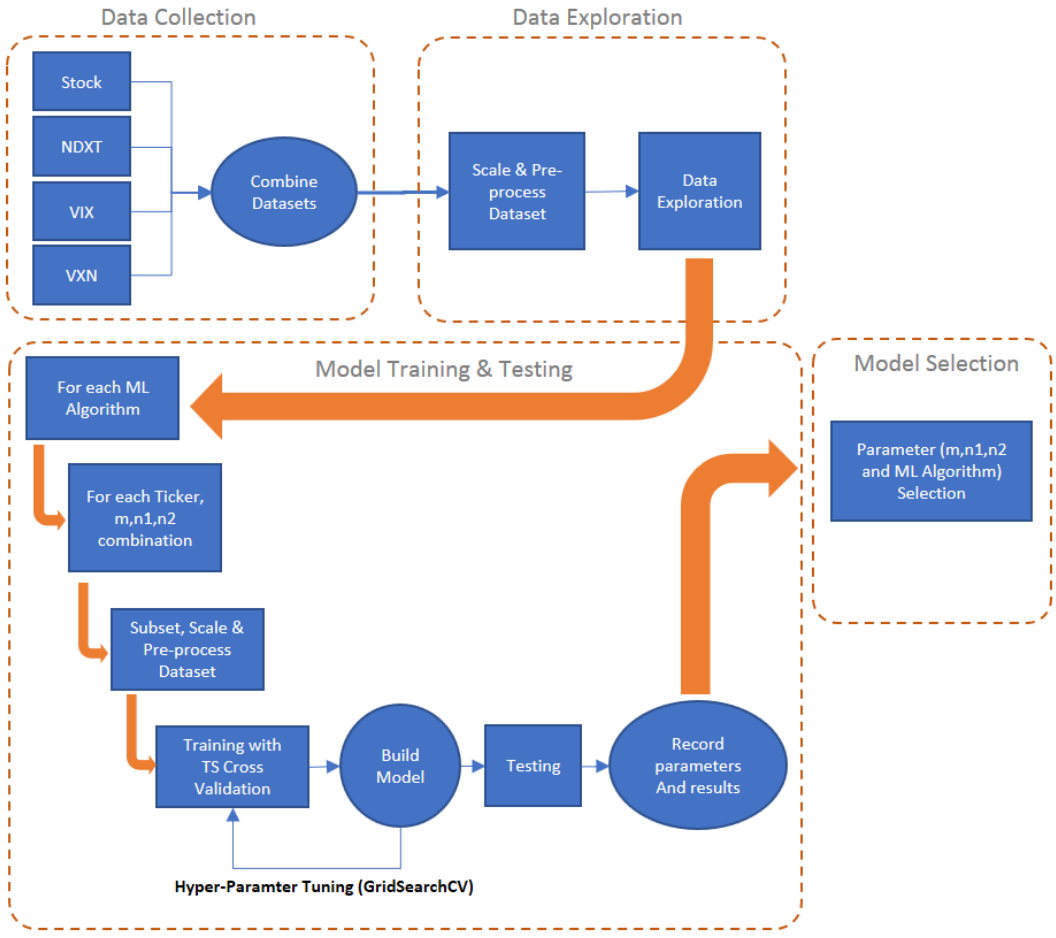

Figure 22 High level view of end-to-end model selection process

1) Data Collection: in this phase all ETF and index input files were read, merged into one dataset and then filtered and "massaged" to include only those ETFs that met the criteria (top performers and only ETFs that had the same amount of daily data available)

2) Data Exploration: in this phase, daily returns are calculated, the dataset is standardized, and daily returns transformed (log10), directional return labels are created. Exploration is performed using distributions (histograms) as well as summary tables of the input and output variables to understand distribution shapes, outliers, class balances, etc.
3) Model Training and Testing: in this phase, the processed data is now passed into a multi-layer *"For Loop"* iterating among ML Algorithm, Ticker, m, n1, n2 to train (using time series cross-validation and GridSearchCV for hyper parameter tuning), test such models and record its results for later analysis. An Out-of-the-Box Random Forest was included so it could be used as benchmark for comparison
4) Model Selection: this final phase, encompassed the selection of the model, input parameters needed to achieve maximum accuracy (based on AUC_ROC)

As part of any data analytics process, many tools were used to execute this project; this is to say that while efforts were made to utilize plotting and summarizing techniques from Python, Pandas or MatploLib other tools were also used when these tools would mean a more effective and efficient analysis (i.e. Excel, Excel charts, Pivot Tables). This is important because data scientists must recognize in which tasks some tools are more effective and/or efficient than others.

The most challenging pieces of the project were:

- ✓ Generate summaries and visualizations that would provide insights into what the characteristics of the dataset, features and labels. In this case, there are no rules on how it must be done, but just fully exploring the dataset by asking and responding as many questions as possible from it
- ✓ Use and implement a cross-validation technique valid for time series. This is a topic that is not as well documented as k-fold cross-validation techniques, hence, it required some time for researching, and implementing it

- ✓ Use GridSearchCV over a series of classifier in an automatic iteration. A forum question was opened, and no answer has been posted until the delivery of this report. However, a workaround for this issue, which produced the same results, was devised and it was implemented to finalize and deliver this report.

It was also noted during the execution and analysis of this project that, given a set of m, n1 and n2 parameters, the best performing ML Algorithm for predictions may be different for different ticker/ETFs which may be important if the objective is to maximize returns on one specific ticker/ETF.

Finally, this project successfully identified a new model (including m, n1 and n2) for predicting future performance; it surpassed the AUC_ROC values of the benchmark, however, the average accuracy (AUC_ROC) was not higher than 52% and it was not possible to obtain higher results despite optimization efforts on hyper-parameters. From an optimistic point of view, 52% is a slightly better performance than a "random walk" and this few points of accuracy advantage could potentially mean large amounts of gains in the stock market. On the other hand, as it can be shown from the results above, predicting the stock market is a daunting task and it likely needs more features to achieve higher levels of performance (this project was based on 4 input variables and 8 engineered features).

Improvement

This project attempted to use all techniques learned over the course of the Nanodegree program including variable encoding, function creation, data scaling and transformation, cross-validation, hyper parameter tuning to mention the most important ones. Dimensionality reduction techniques were not used for the final submission of this project. Although the number of features on this project is not large, it'd still be suggested to apply this technique since it might help removing "noisy" data from our dataset. Since this is basically a Supervised Machine Learning problem, it is recommended to look into Linear Discrimination Analysis for this task.

Another point worth digging deeper into is to find ways where SVM and Neural Network classifiers do perform better. From the results obtained in this project, these two classifiers did not perform as well as other "simpler" ML algorithms analyzed.

Given that this project performed a sensitivity analysis on top of predictions for ML, and given the specific framework or conditions of the project (to use only momentum and volatility as features), the solution obtained certainly would imply either the best solution or the second top one if we consider that dimensionality reduction may further improve the performance of the model. Other features could also be added to the analysis that could also provide information to improve performance, for instance: sentiment analysis of the market.

Another potential improvement that can be executed in the project is to add a multi-classification problem as opposed to the purely binary one as it stands today. This is, Adding a "No Change" to the Target Feature label to accommodate for changes where there's no real need for adding or dropping and ETF from the portfolio.

References:

- Alsotad and Davalcu (2017). Directional prediction of stock prices using breaking news on Twitter. Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015 IEEE / WIC / ACM International Conference. http://ieeexplore.ieee.org/document/7396858/?reload=true
- Fama, E. (1965). The Behavior of Stock-Market Prices. Journal of Business, Volume 38, Issue 1 (Jan., 1965), 34-105.
- https://www.jstor.org/stable/2350752?seq=1#page_scan_tab_contents
- Fortuny et all (2014). Evaluating and understanding text-based stock price prediction models. Information Processing and Management 50 (2014) 426–441. http://www.sciencedirect.com/science/article/pii/S0306457313001143?via%3Dihub
- Gunduz, Cataltepe and Yaslan (2017). Stock daily return prediction using expanded features and feature selection. Turkish Journal of Electrical Engineering & Computer Sciences (2017) 25: 4829 – 4840. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjo7NmyxurYAhVR7qwKHfpYAyEQFgguMAA&url=http%3A%2F%2Fjournals.tubitak.gov.tr%2Felektrik%2Fissues%2Felk-17-25-6%2Felk-25-6-32-1704-256.pdf&usg=AOvVaw3Rw6mCh3x1rWkD2fGjHaWZ
- Samuelson, Paul A., Proof That Properly Anticipated Prices Fluctuate Randomly Industrial
- Management Review, 6:2 (1965:Spring). http://jrjgb.jj.cqut.edu.cn/__local/6/35/CE/516B5F529EC5AAF4B9C1FFD5C4B_A532FC8A_B39E2.pdf
- Shleifer and Summers (1990). The Noise Trader Approach to Finance. The Journal of Economic Perspectives, Vol. 4, No. 2 (Spring, 1990), pp. 19-33. http://www.jstor.org/stable/1942888
- Madge, Saahil (2015). Predicting Stock Price Direction using Support Vector Machines
- https://www.cs.princeton.edu/sites/default/files/uploads/saahil_madge.pdf
- Mullainathan and Spiess (2017). Machine Learning: An Applied Econometric Approach. Journal of Economic Perspectives—Volume 31, Number 2—Spring 2017—Pages 87–106. https://www.aeaweb.org/articles?id=10.1257/jep.31.2.87
- Nardo, Petracco-Giudici and Naltsidis (2016). Walking down wall street with a tablet: a survey of stock market. Journal of Economic Surveys (2016) Vol. 30, No. 2, pp. 356–369. http://onlinelibrary.wiley.com/wol1/doi/10.1111/joes.12102/full
- Li, Xie and Wang (2016). Empirical analysis: stock market prediction via extreme learning Machine. Neural Comput & Applic (2016) 27:67–78. https://link.springer.com/article/10.1007/s00521-014-1550-z