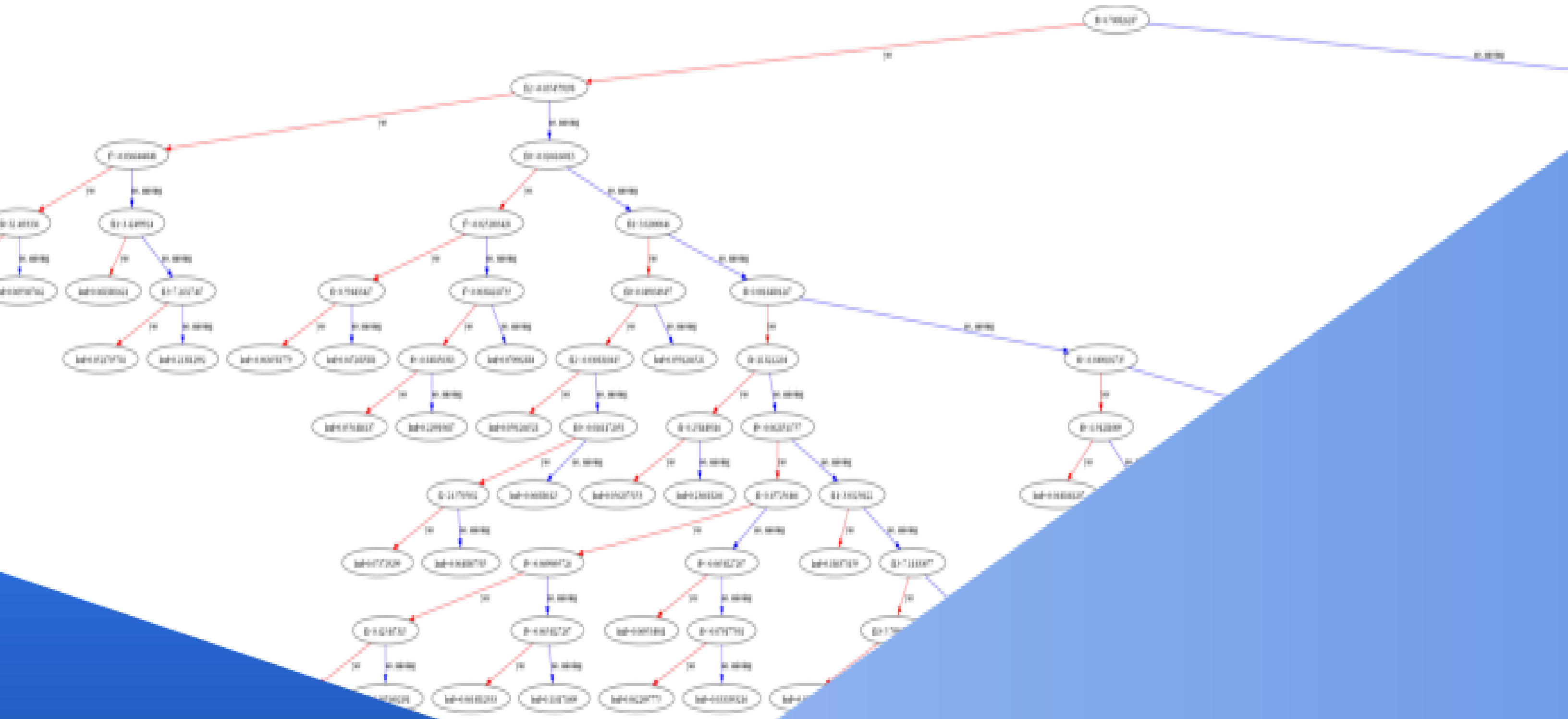
RESEARCH

Machine Learning Approach to Labeling and Predicting the Stock Market.

September 2023



Communication enquiries should be directed to the following contact:
Matas Urbonavicius
+370 6 196 2244
matas.urbonavicius@gmail.com
linkedin.com/in/matasurbonavicius/

Machine Learning Approach to Labeling and Predicting the Stock Market

Abstract

This paper delves into the methodology of training unsupervised machine learning models to first label unlabeled data (stock prices) and then train supervised models to predict them. Author uses K-Means, Hidden Markov Model, Extreme Gradient Boosting and Random Forest models to construct a unified framework and discusses their strengths and weaknesses.

Article achieves success at developing a very robust labeling technique ("Oracle"), however, predictions only have a partial success, due to the nature of the algorithm. The model can achieve a lower drawdown with a slightly higher return.

Table of Contents

Introduction

The stock market is characterized by its volatility and inherent unpredictability. Accurate forecasts of its movements from day to day, or even week to week, are notoriously difficult to make. However, market analysts often categorize the market's behavior into some distinct states: "Bear", "Bull" and sometimes "Static". Historical analysis reveals that transitions between these market states are quite infrequent. See Fig. 1. This observation leads to a hypothesis: if one could reliably identify the current state of the market, one might then make investment decisions that would be correct the majority of the time, given the rarity of state changes. Furthermore, if a method could be devised to anticipate these shifts before they occur, such a predictive system could potentially generate substantial returns.

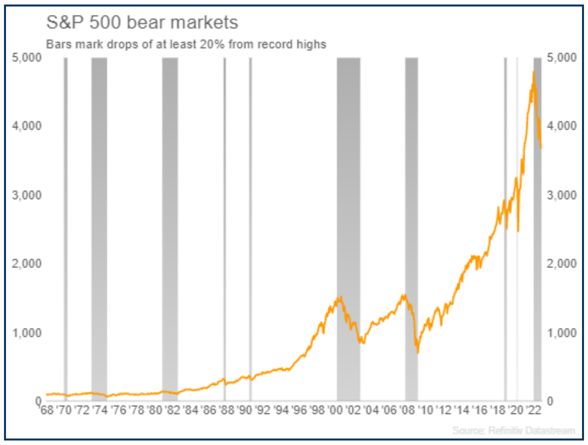


Fig. 1 - Bear & Bull SPX Periods[1]

A high overview of the steps to achieving this goal are as follows:

1. Get and process the data
2. Label the data (determine market states historically)
3. Use the labeled data for prediction

Data Processing

I. Sources: Two data sources of public use were considered when writing this paper- Yahoo Finance and FRED. They offer an extremely wide variety of both economical and price data to work with and have their API python wrappers, namely the "yfinance" and "fredapi" which make it simple to access their data quickly in the terminal.

II. Data Selection: Paper is based on weekly period data- all backtests are on a weekly time frame too. A large pool of different securities and economical indicators are considered. Categories that have been identified are- price data (and example being the VIX or Gold price), indicators (calculated on an underlying instrument price, such as a moving average on a VIX), economic data (such as inflation or unemployment), fixed income data (that is yields, yield spreads). Later on, as this paper will discuss, all of these features will need to be lagged by a certain amount, to avoid the forward looking bias.

Price Data: It has been widely regarded that some markets or stocks tend to be more forward-looking than others. Sometimes, Nasdaq Composite can be more forward looking than Dow Jones, especially in

[1]https://www.reuters.com/business/nasdaq-futures-tumble-3-aggressive-rate-hike-bets-2022-06-13/ters

the environment of low interest rates. The opposite may follow during a restrictive environment. Using the same logic, it is easy to make an argument for having cyclical stocks- that is, those that tend to follow the economy. The model should be able to capture the relationships described. The prices of stocks will of course be lagged to avoid looking into the future when predicting.

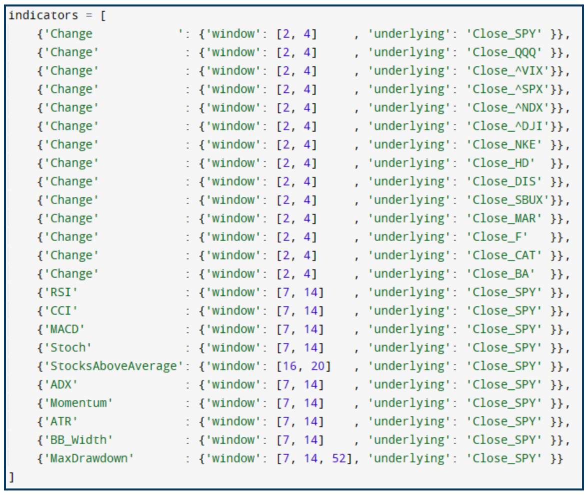
- Paper considers change of various index and their ETF price data: SPY, SPX, QQQ, NDX, IYY, DJI, VIX. Paper looks in the change of 1 week and 1 month.
- Some cyclical stock prices were considered too: BA, CAT, F, MAR, NKE, HD, DIS, SBUX. Paper looks in the change of 1 week and 1 month.

Indicators: Indicators are mathematical formulas, usually constructed from Open- High- Low- Close- Volume data of the instrument. They can give meaningful insights and offer a different perspective. Traders use indicators for various purposes, including risk, psychology management, market timing or position sizing (part of the risk management).

While the market is proven to be non mean-reverting, the indicators, however, are and can be used to construct various strategies that are exhibiting mean-reverting properties. The paper will evaluate some of the most widely used and accepted indicators. Each indicator will be given two windows for calculation - a shorter and a longer one to reflect both quick and slow changes.

- Trend:
  - Relative Strength Index (RSI)
  - Commodity Channel Index (CCI)
  - Moving Average Divergence Convergence (MACD)
  - Stochastic Oscillator (Stoch)
  - Percentage of stocks in the S&P 500 that are above a certain average (StocksAboveAverage)
- Momentum:
  - Average Directional Index (ADX)
  - Momentum Indicator (Momentum)
- Volatility:
  - Average True Range (ATR)
  - Bollinger Bands Width (BB Width)
- Change:
  - Trailing Max Drawdown (MaxDrawdown)
  - Change from a given period (Change)

All indicators (features) are shown in Fig. 2 below.



```
indicators = [
    {'Change             ': {'window': [2, 4]     , 'underlying': 'Close_SPY' }},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_QQQ' }},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_^VIX'}},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_^SPX'}},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_^NDX'}},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_^DJI'}},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_NKE' }},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_HD'  }},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_DIS' }},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_SBUX'}},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_MAR' }},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_F'   }},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_CAT' }},
    {'Change'             : {'window': [2, 4]     , 'underlying': 'Close_BA'  }},
    {'RSI'                : {'window': [7, 14]    , 'underlying': 'Close_SPY' }},
    {'CCI'                : {'window': [7, 14]    , 'underlying': 'Close_SPY' }},
    {'MACD'               : {'window': [7, 14]    , 'underlying': 'Close_SPY' }},
    {'Stoch'              : {'window': [7, 14]    , 'underlying': 'Close_SPY' }},
    {'StocksAboveAverage' : {'window': [16, 20]   , 'underlying': 'Close_SPY' }},
    {'ADX'                : {'window': [7, 14]    , 'underlying': 'Close_SPY' }},
    {'Momentum'           : {'window': [7, 14]    , 'underlying': 'Close_SPY' }},
    {'ATR'                : {'window': [7, 14]    , 'underlying': 'Close_SPY' }},
    {'BB_Width'           : {'window': [7, 14]    , 'underlying': 'Close_SPY' }},
    {'MaxDrawdown'        : {'window': [7, 14, 52], 'underlying': 'Close_SPY' }}
]
```

Fig. 2 - Indicators and their settings

Fixed Income Data: Fixed income market is larger than equities, hence it is logical to include some data that drives the market or is a direct consequence of increase or decrease of risk in the market. Spreads act in a similar manner like VIX for the SPX. The higher the unknown (or volatility) - the higher the spread. Paper also includes 3 data points that are directly influenced by the central bank as it is the main driver of the monetary policy.

- BBB Yield Spread (BBB)
- AA Yield Spread (AA)
- HY Spread (HY)
- 10Y-2Y Spread (10Y2Y)
- FedFunds (FEDF)
- US 3-Month rate (3M)

III. Data Preparation: An important and a widely used practice is to split the data into three parts. Training data, validation data and the testing data. Model will train and then optimize on the first two datasets, it will be tested on the unseen testing data. A common practice is to have the proportion of about 60/20/20. See Fig. 3 below.

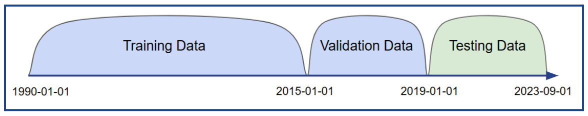


Fig. 3 - Data split

After the split, data needs to be prepared by filling in the missing values with the "forward fill" method. Next step is to remove the NaN values, such as those left in the beginning, since to calculate some indicators (ex. RSI), we first need to skip the amount of rows until data is sufficient for calculation.

An overview of the data preparation framework is shown in Fig. 4.

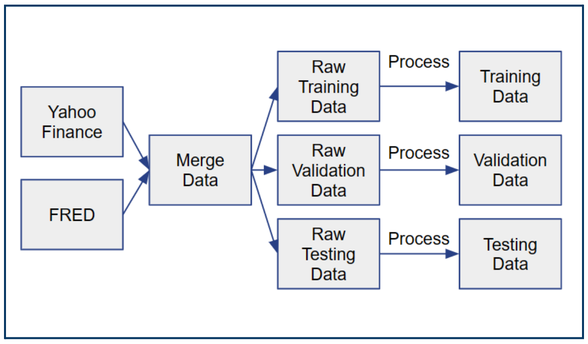


Fig. 4 - Data Processing Overview

Labeling Models

I. Introduction: There are two types of machine learning models: Supervised Learning and Unsupervised Learning, both will be used in this paper.

Supervised Learning relies on labeled input and output training data. In other words- model needs a test and the answers data. It learns by discovering patterns that lead to the correct answer.

Unsupervised Learning does not need any labels. It can take raw data and try to create something meaningful out of it. Because of it, each training-testing session tends to give slightly different results due to the fact that there are many ways to cluster or find "meaningfulness" in thousands of rows of numbers. This reason alone is enough to not utilize Unsupervised Learning models when predicting- it is highly unstable.

Notice how both of these models can work together. Unsupervised Learning can label the data (set the answers) and Supervised Learning models can use those labels to train themselves.

Prior to the paper, the author has researched and selected 2 different models for each type of machine learning algorithms that suit this task. Models will label the data into "bearish" and "bullish" states. Later, conditional rules will adjust their classification, add a "neutral" state for the market and will be called "Oracle".

II. First model: K-Means is by far one of the most popular machine learning algorithms due to its relative simplicity and ease for intuition.

Intuition: Given a set of observations (x1, x2, ..., xn), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (≤ n) sets S = {S1, S2, ..., Sk} so as to minimize the within-cluster sum of squares (WCSS)[2]

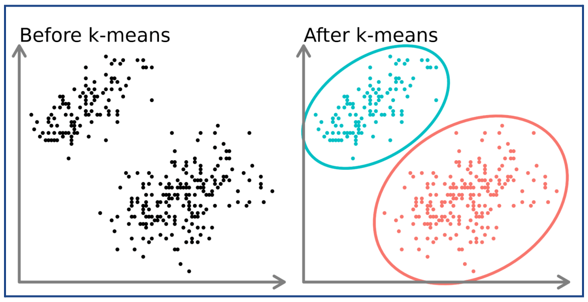


Fig. 5 - K-Means after clustering.[3]

Idea for K-Means in the context of this paper is that by finding different clusters from all these indicators, we could determine different market conditions. During the 2020 Covid crash, all volatility indicators were going "through the roof". K-Means would label them separately and this could be an identifier for us that the market is currently in a stressful period.

III. Second Model: Hidden Markov Model (HMM) makes an assumption that it can see the observations (indicator values), but those observations are directly determined by some hidden patterns or states. Formally speaking, it seeks to recover a sequence of hidden states from the observed data.

If we assume that the indicator values are a cause of an underlying market state, then HMM should be able to discover that underlying state.

Because HMM can recover the "true" states, it can calculate the probabilities of transitioning from one state to another at a given point, namely the transition matrix. However, in the context of this paper, the transition probabilities are over 90% of staying in the

[2]https://en.wikipedia.org/wiki/K-means_clustering

[3]https://www.datacamp.com/tutorial/k-means-clustering-python

same state, therefore it does not have a meaningful use for this study.

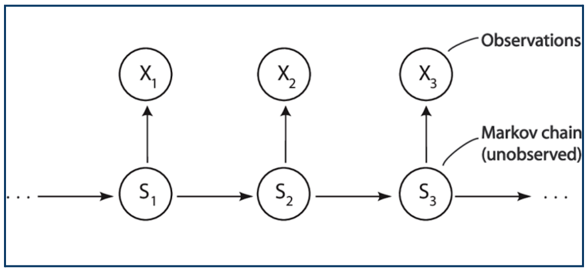


Fig. 6 - Hidden Markov Model intuition[4]

IV. Ensemble: Final result should be a single list of values, therefore we must combine the K-Means result together with HMM, to achieve a unified data structure.

One popular method is the majority voting. If 5 algorithms label the current state as bullish and 3 algorithms are on the contrary side, then the majority of 5 will win, thus labeling the current state as bullish. This paper considers only two models, a number that is too little for the majority voting, hence some modifications are necessary.

Stock market exhibits growth over time, therefore the author applies some bias towards bullish values. When ensembling and both models disagree on the state, a more bullish state will be assigned. Example: K-Means label "bullish" and HMM label "bearish" -> "bullish" will win.

V. Oracle labeling: Even though the majority of time we can classify markets as being in the bullish or bearish state, sometimes it is more accurate to say that the market is following a "neutral" or "static" regime.

Logically, the transition period from bullish to bearish could be labeled as neutral. One strategy is to identify the highest point of the state and label everything forward as neutral until the new state. However, the market tends to rebounce more quickly than it tends to exit the bullish period into the red territory, therefore for the bearish state, we can find the low point and anything forward should be labeled as bullish. Lastly, as the paper discussed in the beginning, market shifts occur rarely, therefore the algorithm filters all states that are smaller than 1 month assigning the previous state value to it. The process is shown in Fig. 7.

[4]https://dlab.berkeley.edu/news/brief-primer-hidden-markov-models

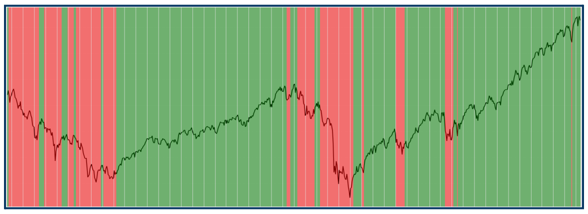


Fig. 7.1 - K-Means classification

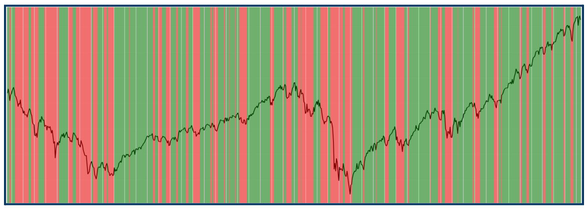


Fig. 7.2 - HMM classification

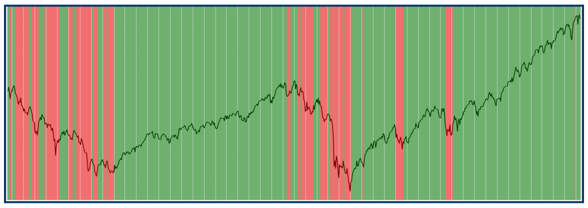


Fig. 7.3 - Ensembled classification

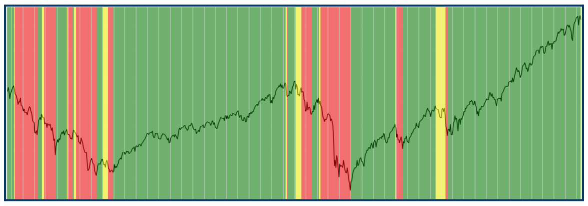


FIg. 7.4 - Oracle classification

For the sake of double confirmation to make sure the labels are a good training material, we can create a simple backtest and overlay it with the SPX. If it will beat SPX considerably, we could conclude that labels are set correctly. Backtest rules: hold 100% during a bullish market, 60% during neutral market and 30% during the bullish market. Backtest balance is shown in Fig. 7.

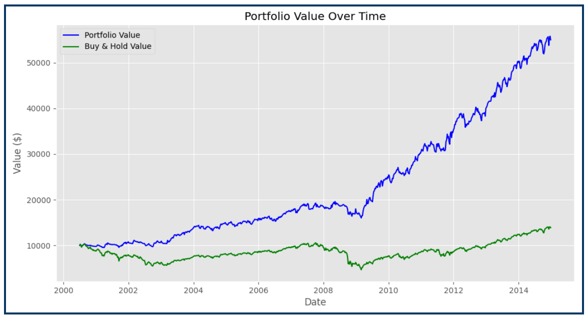


Fig. 8 - Oracle labeled backtest

Backtest confirms the "Oracle" vision of the labels. Please note, that this is by no means a realistic backtest because Oracle labels were constructed after the fact, meaning forward looking bias was introduced. For example- the author has filtered out all states that were less than 4 weeks long. In hindsight this would not be possible as we cannot forecast correctly how long the state, that has just been predicted, will last. An important note: Every backtest will be constructed with the same holding rules as in Fig. 8.

A simplified framework of the labeling process is shown in Fig. 9 below.

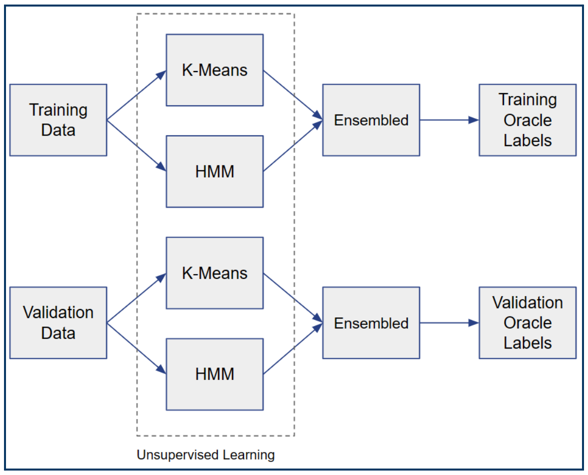


Fig. 9 - Overview of labeling

Prediction Models

I. Introduction: The labeling models could be applied to determine the state of the market, however we would not know the true state of the market until the end of the week, therefore it is crucial to forecast the state one week in advance. Additionally, study seeks to evaluate the accuracy of the models and this cannot be done directly with unsupervised models.

Paper showcases the process of using Oracle labeled training and validation data to train and optimize XGBoost and Random Forest, both of which are supervised prediction models. Ensembling will come as the natural final step.

A crucial step prior to training the prediction models is to shift the feature values forward by one. This is done to prevent the algorithm from looking forward to the future and also train itself to find patterns which would lead to a certain price movement in the future.

II. First Model: Extreme Gradient Boosting (XGB): XGB is a powerful model that uses a technique called gradient boosting to build and improve models in a step-by-step fashion. It creates a series of decision trees, where each subsequent tree attempts to correct the errors of the previous one. This process continues iteratively, combining the trees into a strong model that makes highly accurate predictions.

XGB has a built-in importance function that displays how important each feature (indicator) is to the overall prediction. This can help reduce the number of indicators thus increasing efficiency. Moreover, XGB has parameters for regularization and cross-validation to ensure both bias and variance are kept at a minimum.

It is noteworthy to tell, that the second model, Random Forest, has feature importance function too, however, XGB is considered a more advanced and accurate algorithm, therefore the paper will consider the feature importance of the XGB only. By utilizing the function, we can minimize the amount of indicators from 52 to 26- that is, split it by half. See Fig. 10. An interesting note is that none of the FRED economic indicators nor the yields proved important as predictors of the short term market movements.

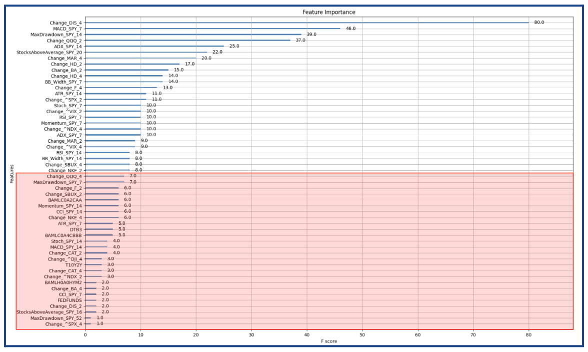


Fig. 10 - XGB feature importance chart

III. Second Model: Random Forest is an ensemble algorithm that builds multiple decision trees and merges them together to get a more accurate and stable prediction. Each tree in the forest is created from a random subset of the data, and it makes decisions based on random subsets of the features. The final prediction is made by a majority vote (classification).

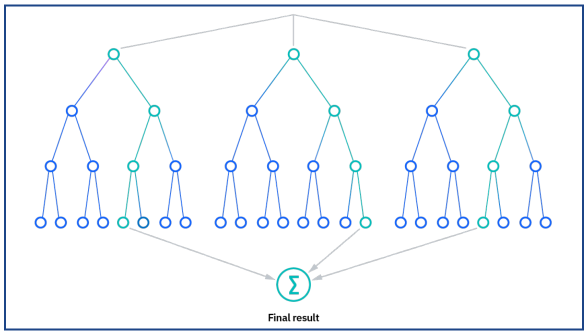


Fig. 11 - Random Forest architecture[5]

IV. Ensemble: Just like with the labeling models ensemble, the author applies the same rules to the prediction models. When ensembling and both models disagree on the state, a more bullish state will be assigned.

A high-level overview of the prediction architecture is shown in the Fig. 12 below:

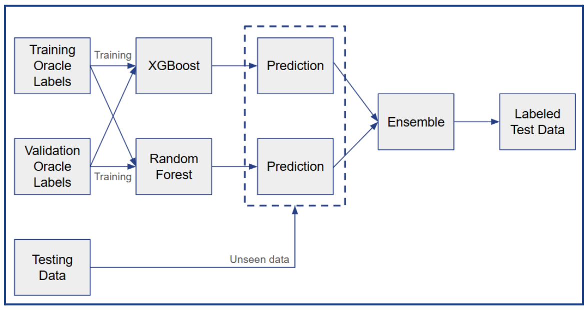


Fig. 12 - Prediction model architecture

Prediction Evaluation

I. Introduction: Evaluation of the model is a crucial part which must avoid biases and errors more than the other parts. We make our assumptions about the model based on how we evaluate it. We choose to trade the model based on evaluation. We present our model through evaluation.

As a benchmark, we can use the same “Oracle” labeling technique which, safe to assume, holds the “true” values of the market states. Overlay these with the predictions and it is easy to calculate the accuracy of the model. See “Oracle” labeling in Fig. 13 below and prediction accuracy, when compared to “Oracle” in Fig. 14.

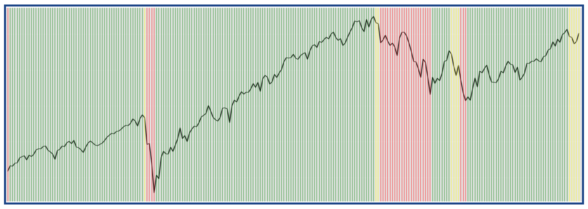


Fig. 13 - “Oracle” labeled test data as a benchmark

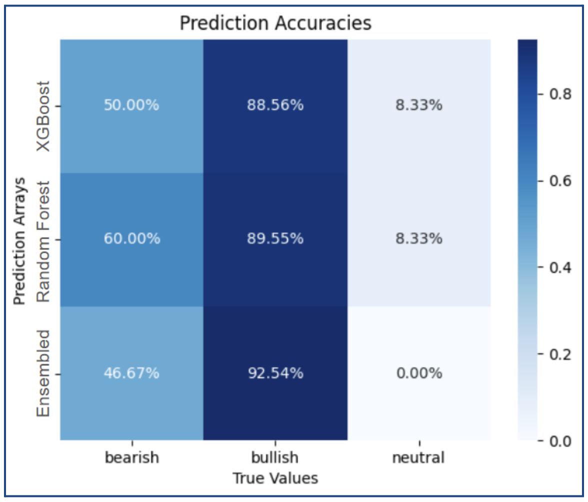


Fig. 14 - Model accuracy

The model seems to be highly accurate when forecasting the bullish states and the contrary holds about neutral. The most important measure is how accurate the algorithm is with bearish states, as practically speaking, one of the main goals for this paper is to predict the bear market. Everything else can be labeled as a bull market and it will be mostly correct. The worst thing about being in the market during the neutral state is opportunity cost, and this is not really painful. Reason for such low accuracy of neutral state is the fact that “Oracle” labeled state as neutral only a few times for a very short amount, leaving the window to predict it correctly extremely small.

Looking into the prediction chart, it is obvious that the models are a little lagging, which is okay for most part except when the market rebounds sharply or drops very quickly without any prior momentum. For this case, it is a good idea to have a model that is quicker to react. The predicted values are visualized in Fig. 15 below. Note the red circle, indicating the substantial lag.

[5]https://github.com/Andre-Luis-Lopes-da-Silva/Random-forest-to-predict-the-macroeconomics-scenario-to-buy-stocks-of-retail-sector

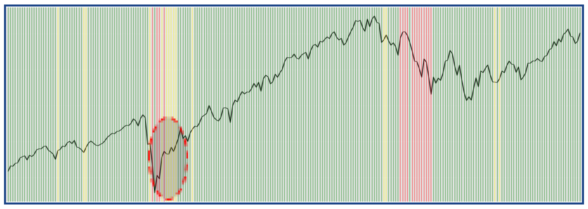


Fig. 15 - Predicted states

Backtest confirms that the market beats the current model.

Metric	Buy&Hold	Model
Max Drawdown	32.23%	27.39%
Return	74.22%	61.42%

Table 1 - Model backtest statistics

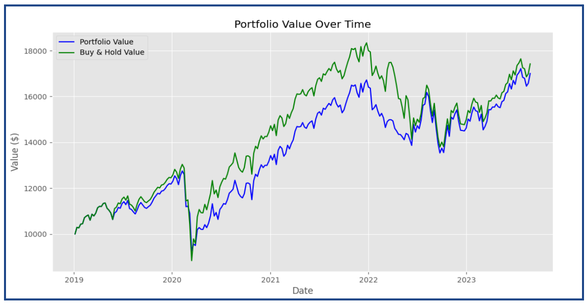


Fig. 16 - Backtest of the model

Optimization

I. Introduction: Paper discusses two types of optimization: hyperparameter optimization for each prediction model and a simplistic approach of changing indicator window values to a smaller number, with the idea of creating a faster algorithm.

II. Hyperparameter optimization: This is a critical process in machine learning that involves finding the most effective combination of parameters that govern the training of a model. These hyperparameters, unlike model parameters, are not learned from the data but are set prior to the training process and have a significant impact on the performance of the model. The goal of hyperparameter optimization is to search through a range of these settings to improve the model's ability to make accurate predictions.

Paper considers a technique called Randomized Search. It randomly samples the space of possible parameters to find the best values for a given model. Unlike grid search, which exhaustively tries all possible parameter combinations, randomized search only tries a fixed number of parameter settings from a specified distribution.

This approach is more efficient when dealing with a large number of hyperparameters and complex models, because it can discover good combinations with fewer iterations. The randomness helps to explore a wide range of values and can often lead to better results in less time than grid search.

XGB was optimized with these parameters:

```
param_dist = {
'eta': [0.01, 0.05, 0.1, 0.2, 0.3],
'n_estimators': [100, 200, 500, 1000],
'max_depth': [3, 4, 5, 6, 7, 8, 9, 10],
'subsample':  [0.5,  0.6,  0.7,  0.8,  0.9,
1.0],
'colsample_bytree':[0.5,0.6,0.7,0.8,0.9,1.0
],
'gamma': [0, 0.25, 0.5, 1.0],
'lambda': [0, 0.1, 1.0],
'alpha': [0, 0.1, 1.0]
}
```

While Random Forest was optimized with these parameters:

```
param_dist = {
'n_estimators':[10,  50,  100,  200,  500,
1000],
'max_depth': [1, 5,  10, 20, 30, 40, 50],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4],
'max_features': [1, 100],
'bootstrap': [True, False]
}
```

Fig. 17 demonstrates the effect on the testing data pre- and post hyperparameter optimization for the XGB model only. I will not demonstrate the result of Random Forest purely for space-saving purposes. It is noteworthy, how the balance of the strategy tends to get very close to the Buy & Hold result, however it keeps the drawdown at lower lever. Max drawdown decreases by 5%.

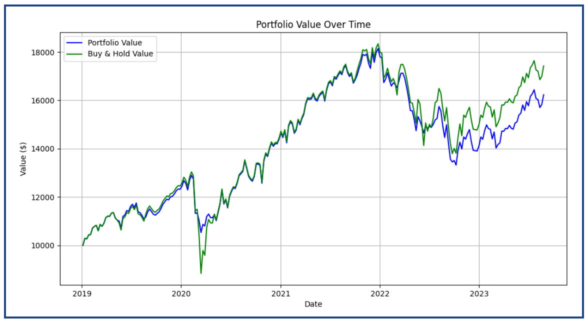


Fig. 17.1 - Pre- hyperparameter optimization backtest

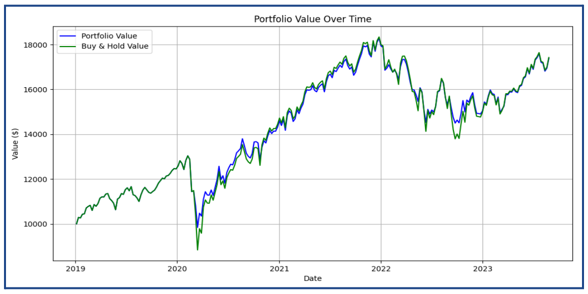
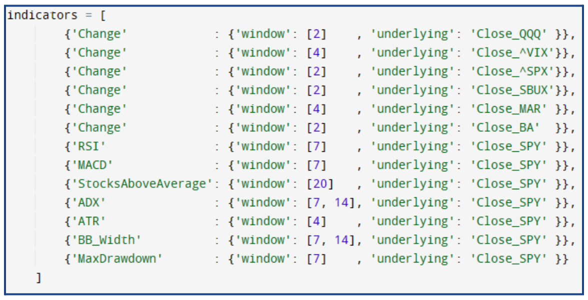


Fig. 17.2 - Post hyperparameter optimization backtest

III. Importance and window size reduction: Since one of the main tasks of this algorithm is to detect the bear market early, we can lower the window for all of our indicators by a couple of digit places and train our models on the bearish data only (that is, not using any bullish labels) and then run the importance function one more time, decreasing the number of features even more.

IV. Final evaluation: Final Set of features after the reduction is shown in Fig. 18 below:



```
indicators = [
        {'Change'           : {'window': [2]    , 'underlying': 'Close_QQQ' }},
        {'Change'           : {'window': [4]    , 'underlying': 'Close_^VIX'}},
        {'Change'           : {'window': [2]    , 'underlying': 'Close_^SPX'}},
        {'Change'           : {'window': [2]    , 'underlying': 'Close_SBUX'}},
        {'Change'           : {'window': [4]    , 'underlying': 'Close_MAR' }},
        {'Change'           : {'window': [2]    , 'underlying': 'Close_BA'  }},
        {'RSI'              : {'window': [7]    , 'underlying': 'Close_SPY' }},
        {'MACD'             : {'window': [7]    , 'underlying': 'Close_SPY' }},
        {'StocksAboveAverage': {'window': [20]   , 'underlying': 'Close_SPY' }},
        {'ADX'              : {'window': [7, 14], 'underlying': 'Close_SPY' }},
        {'ATR'              : {'window': [4]    , 'underlying': 'Close_SPY' }},
        {'BB_Width'         : {'window': [7, 14], 'underlying': 'Close_SPY' }},
        {'MaxDrawdown'      : {'window': [7]    , 'underlying': 'Close_SPY' }}
    ]
```

Fig. 18 - Final set of features

Training the model, then optimizing its hyperparameters, yields us the final backtest result for the 2019-01 to 2023-09 period. Even though the backtest has no transaction cost calculation, it should not make a big difference in practice, because of the little amount of trades.

Metric	Buy&Hold	Model
Max Drawdown	32.23%	27.70%
Return	74.22%	78.93%

Table 2 - Final backtest statistics

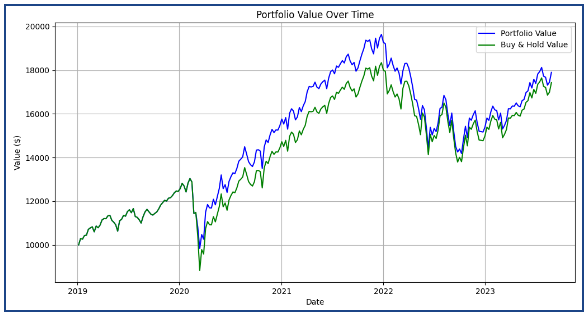


Fig. 19 - Final backtest

Final Thoughts

I. Conclusion: Paper discussed the methodology of data preparation, labeling and predicting the states of the market using four different machine learning algorithms.

The final model is able to achieve a smaller drawdown than the market with a slightly larger return. The model is not utilizing any market anomaly nor a concrete proven strategy, making it extremely difficult to extract any meaningful edge. Moreover, since we are using 1-week delayed data to predict the next week's state, we are essentially stating that the current information has influence on the future returns. If this is true, the edge should be extremely small, which is practically demonstrated by this paper.

In the future, the model should be expanded by adding a "trader" on the predicted values. Perhaps a conditional module which oversees how the trading is done. Additionally, even though this model has considered a significant number of features, all of them are wildly popular, hence already efficient and priced in, indicating the lack of edge to be extracted.

II. See GitHub repository for the code: https://github.com/matasurbonavicius/Quant/tree/main ant: Quant Connect Algorithm Framework (github.com)

III. About the author: Matas Urbonavičius is an aspiring capital markets professional with experience ranging from M&A in NYC to Prop trading for a Family office located in Luxembourg