Impact of Intermarket Data on Stock Market Prediction

A Major Qualifying Project (MQP) Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements
for the Degree of Bachelor of Science in

Computer Science,
Data Science

By:

N'yoma Diamond
Grant Perkins
Jillian Wright
Brad Cosma

Project Advisors:

Marcel Blais, Stephan Sturm, Xiaozhong Liu

Sponsored By:

Branka Hadji-Misheva (Berner Fachhochschule)
Jörg Osterrieder (Berner Fachhochschule, University of Twente)

Date: October 2022

IMPACT OF INTERMARKET DATA ON STOCK MARKET PREDICTION

N'yoma Diamond, Grant Perkins
Departments of Computer Science & Data Science
Worcester Polytechnic Institute
Worcester, MA 01609
{badiamond,gcperkins}@wpi.edu

Jillian Wright, Brad Cosma
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609
{jjwright,bcosma}@wpi.edu

November 3, 2022

ABSTRACT

Stock market forecasting is one of the most widely explored subjects in both academia and private industry. The ability to predict future market movement, both short- and long-term, has significant implications towards the identification of economic changes, adjustment of investment portfolios, and general moneymaking in investment markets. In our exploration, we identified a lack of consideration for intermarket data on the ability of a machine learning model to forecast stock market value, such as the bond, currency exchange, and futures markets. In this paper, we implemented Logistic Regression, Decision Tree/Random Forest, K-Nearest Neighbors, and Support Vector Machine models to explore the impact of intermarket data. Using these models we compare the quality of predictions made with respect to different combinations of assets and identified any potential usefulness of intermarket data towards the task of predicting S&P 500 Index movement. Significant differences in intermarket datasets and stock market datasets were not found. However, select intermarket datasets outperformed simple stock market datasets in all metrics.

Keywords: machine learning, financial forecasting, intermarket analysis, Forex, bonds, futures

1 Introduction

Financial market analysts spend countless hours and resources working to predict stock market trends. In order to optimize profits through strategic buying and selling, knowledge of the market's fluctuations is key. The Efficient Market Hypothesis (EHA)—developed in part by Eugene Fama in his 1965 dissertation and then popularized by a comprehensive review of existing research in 1970 [1, 2]—states that market prices represent all information available to the market and that, on exchanges, stocks are traded on their fair market value. Under this theory, Fama proposed that the market follows a random walk, suggesting that past value information cannot be used to predict market trends, as the direction and magnitude of future growth is completely random [1]. However, electronic trading and the use of complex machine learning models for optimal market analysis have seen increasing interest and research, bringing the validity of EHA into question.

Electronic trading is the automation of financial asset trading via automated computer systems. Its popularity stems from significant ease of use, elimination of geographical barriers, and potential for extremely high-volume and high-frequency trading that would otherwise be impossible [3]. Algorithmic trading utilizes automated trading in an advanced format, analyzing multiple different factors and trends over time following preset instructions (i.e. algorithms) that suggest taking different actions when certain criteria are met. The effectiveness and ubiquity of these automated strategies has led scientists to investigate the potential for machine learning models to revolutionize the investment industry.

The ability for a machine learning model to take in copious amounts of diverse data, analyze it, and make predictions for future data points make it particularly applicable to market prediction. The use of machine learning models can greatly decrease the complexity of data and calculations investors work with. Machine learning models, due to their complexity, can also discover hidden trends that may not be able to be found by simpler models. However, developing a functional and accurate model presents many challenges: the availability of reliable and accurate time series data is limited, as extensive recording for financial markets was only recently established, and large amounts of realistic financial data cannot be synthesized for training, unlike other thoroughly-explored machine learning tasks such as image recognition or signal processing. Furthermore, high-frequency trading data contains large amounts of noise, making producing accurate predictions substantially more difficult. Further, the black-box nature of certain models—such as ensemble techniques or deep learning models—is not ideal for an industry such as finance; monetary trading involves a high amount of risk, inducing reluctance toward trusting models that are difficult to explain at best [3].

According to Murphy [4], intermarket analysis began to gain attention in the 1980s when the US market plunged, followed by the rest of the world equity markets. The trends of the commodity market affect bond prices, which influence the stock market. Foreign markets are always influencing each other, as their economies often overlap and interact [4]. However, while there has been extensive research on machine learning applications in the financial world, the use of intermarket data for US stock market prediction has been insubstantial.

1.1 Purpose

Our goal in this paper is to identify and explain the impact, if any, of including intermarket data on predicting the movement of the S&P 500 Index utilizing machine learning techniques. Specifically, we intend to determine the impact of bonds, foreign currency exchange, derivatives, and commodities markets on stock market prediction. To do so, we collected daily data using the EOD Historical Data API, which we used to train multiple different machine learning models, including Logistic Regression, Decision Tree/Random Forest, Support Vector Machine, and K-Nearest Neighbor models. This provided us with a broad suite of empirical metrics and baselines which were used to accurately summarize the impact of utilizing intermarket data in predicting stock market performance.

This project expands upon a previous Major Qualifying Project (MQP) which examined the effects of data complexity on the performance of a financial forecasting model. Their model used deep Q-learning with the goal of maximizing profit by trading the SPDR S&P 500 Index Fund (ticker SPY) based on daily and intraday stock index fund data and daily Forex data.

This paper is organized as follows: In Section 2 we discuss the background and conduct a review of existing literature on the subject of our research. In Section 3 we explain the methodology of our experiments, some precursory data analysis, and how we will analyze our final results. In Section 4 we discuss the numerical results of our experiments, followed by our conclusions in Section 5. Lastly, in Section 6 we discuss potential future research that may be conducted based on the work done in this paper.

2 Background

2.1 Finance

Assets such as stocks, bonds, currency, and commodities can be traded or exchanged for money or other assets. This can be used as a means of generating significant wealth if bought, sold, and/or loaned with the correct timing. This is the core of investing. These assets hold a value which can fluctuate based on a variety of factors, such as economic state, supply and demand, or just general speculation. Some assets may also take the form of contracts, called derivatives, which place obligations on or give rights to the owner, such as the right to purchase an asset at a given time. Individual assets are typically sold and bought in markets devoted to specific types of assets. The most notable markets are the stock market, foreign currency exchange, bond market, and derivatives market. Each of these markets specializes in a different type of investment, each with different benefits and drawbacks.

2.1.1 Stock Market

The stock market is the platform on which companies sell “shares”, or equity securities, in their businesses in exchange for monetary investment. Generally, businesses maintain a fixed quantity of available shares, making them an asset that can shift in value depending on the performance and overall value of the company due to induced supply and demand. This makes investment in the stock market a prime opportunity for wealth growth and moneymaking.

A stock index is a grouping of securities that attempts to summarize the performance of the stock market or a specific industry (sector). Popular sectors include technology, energy, and consumer goods for various regions. Stock indices are used as a baseline for investors to gauge their success in terms of profit by comparing their returns to the applicable index. Notable stock market indices in the US are the Standard and Poor's 500 (abbreviated S&P 500), Dow Jones Industrial Average (DJIA), and Nasdaq Composite. Some major indices outside the US are the Deutscher Aktien Index (Germany), Financial Times Stock Exchange 100 (UK, abbreviated FTSE 100), Nikkei 225 (Japan), Hang Seng Index (Hong Kong), and Shanghai Stock Exchange Composite Index (China). Each index approximates their target market via different strategies, the most popular being value weighting. Value weighting is the practice of selecting a number of the most valuable publicly traded businesses and weighting their impact on the index by their relative market value. I.e. the larger the market value of the company, the more weight its stock has in the index. The most notable index that uses value weighting is the S&P 500. Other strategies include price weighting, equal weighting, and fundamental weighting.

Many financial institutions maintain and offer shares in funds containing multiple stocks which are designed to meet some goal. These funds are referred to as Exchange-Traded Funds, or ETFs. Most frequently these funds are designed to approximate the performance of market indices. One particularly notable ETF is the SPDR S&P 500 Trust ETF (ticker SPY), which is designed to mimic the S&P 500 stock market index. Other ETFs exist which approximate the general performance of particular industries, bonds, currencies, or businesses sharing some particular characteristic (such as environmental sustainability).

2.1.2 Currency Exchange

The foreign currency exchange market (Forex) specializes in the trading of international currencies. Forex is considered the largest and most liquid financial market, as cash is the most liquid asset available. The value of a currency can change rapidly depending on the economic and other factors impacting a country. As a result, investors can profit substantially by trading between different currencies [5]. According to the Bank for International Settlements, in 2019 the daily trading volume of the Forex peaked at \$6.6 trillion, with the most traded currencies being the United States dollar (USD, involved in 88.3% of trades), the Euro (EUR, 32.3%), the Japanese Yen (JPY, 16.8%), the British pound sterling (GBP, 12.8%), the Australian dollar (AUD, 6.8%), the Canadian dollar (CAD, 5.0%), and the Swiss franc (CHF, 5.0%) [5].

2.1.3 Bond Market

Bonds represent the lending of money by an investor directly to a government or business entity for a specified interest rate. Bonds are generally specified as having a set time to maturity, after which the whole value of the bond plus any additional unpaid interest must be paid to the owner of the bond by the seller. Bonds come in a couple of forms, the most notable of which are government- and corporate-issued bonds. As of 2019, the five countries which borrow the most in terms of bond selling are the United States, Japan, Italy, the United Kingdom, and France [6].

The general performance of bonds can be tracked broadly via bond indices. The value of a bond index aggregates and directly relates to the "yield"—i.e. the amount of interest being paid to investors—of active or matured bonds in a specified grouping of bonds (e.g. bonds from the same organization, bonds of a certain time-to-maturity, etc.).

2.1.4 Derivatives

A derivative is a contract whose value is derived from the value of another asset, such as stocks, bonds, currency, or commodities. This means that derivatives markets trade *contracts*, rather than securities or other liquid assets. Investment using derivatives contracts can be done in a number of ways: Notable examples include futures, options, and swaps. Futures contracts obligate the transaction of some quantity of asset at a future date for a predetermined price. Options contracts are similar to futures, but instead sell the exclusive right to purchase, rather than obligating the transaction outright as is the case with futures. Swap contracts specify the exchange of monetary flow over a period of time, one side of which is usually determined by an uncertain variable, such as asset value or interest rates. Derivative contracts can be made on various asset classes, including stocks, bonds, currencies, and commodities. It is worth mentioning that "spot commodity" markets are where commodities are exchanged directly instead of via long-term contracts. However, the trading of derivatives on commodities as opposed to commodities directly is significantly easier and safer for investors, and is thus preferred.

2.2 Data Analysis

2.2.1 Time Series Analysis

Time series analysis refers to the exploration and explanation of time series data, or data that changes over time. A wide variety of techniques exist to conduct time series analysis, including simple visual techniques, as well as complex statistical methods. For example, one of the simplest methods to analyze a time series is the use of line charts to visualize the movement of vaues over time. Such a chart can provide quick visual information on trends, seasonality, and outliers. Many time series can be described at a high level via general trends. For example, linear and non-linear methods of curve fitting can be applied to independent time series to describe trends within a given time series [7]. A simple method for linearly fitting a time series is Linear Least Squares, which approximates a linear relationship between the provided inputs and a desired output. Related to linear approximations, further methods to analyze time series include the use of correlation and covariance. These values represent how linearly related two variables are and the direction of the relation.

Seasonal variance may be present in time series as well. In a financial context, seasons are typically represented in financial quarters, months, or trading weeks. One way to estimate seasonal variance is by computing moving averages using known season lengths[7]. Another advanced method of time series analysis is spectral analysis, which can be used to investigate cyclic trends beyond seasonal trends. While typically used to analyze wave-like data (e.g. audio), spectral analysis has been successfully used to analyze financial markets [8].

2.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is a commonly-used technique for the analysis of large multivariate datasets, as well as a tool for reducing the overall dimensionality of a dataset while maintaining most of the variation of the dataset. PCA operates by computing an orthogonal linear transformation which minimizes the mean perpendicular distance (usually computed as mean squared error) of each data point to each axis, or principal component, of the new coordinate system. By performing this linear transformation, we can compute what proportion of variance can be explained by each of the principal components. This being the case, PCA is often performed iteratively with increasing quantities of desired principal components until a desired proportion of variance is explained. Two common thresholds techniques are to generate components until some desired total variance is explained or until a new component no longer exceeds some minimum explained variance threshold.

By performing PCA, it is possible to significantly reduce the overall dimensionality of a dataset. This is especially important for machine learning, as the time complexity of many machine learning techniques is proportional, or potentially exponential, with respect to the dimensionality of the training data. It is for this reason that PCA is frequently experimented with and used with certain predictive models. For some datasets, PCA also has the occasional side effect of physically separating classes along principal components. This can be very beneficial for certain boundary-, probability-, or clustering-based techniques, such as Support Vector Machines, Naive Bayes, Logistic Regression, or K-Nearest Neighbors. That said, class separability is not guaranteed or even an intentionally optimized target for PCA, instead just being a convenient side-effect in certain scenarios. In some scenarios it is possible that PCA may actually reduce class separability. This being the case, techniques like Linear Discriminant Analysis are better optimized for this task if desired. However, Linear Discriminant Analysis relies on assumptions which may not be true for many datasets.

2.3 Machine Learning for Financial Prediction

The task of predicting financial markets using machine learning has been extensively studied. Machine learning encompasses the large set of techniques that utilize existing data to fit procedures for predicting outcomes for a provided input [9]. To determine which machine learning methods might be effective for financial prediction, Milosevic [10] trained various models, including Support Vector Machine (SVM), Random Forest, Logistic Regression, Naive Bayes, and Bayesian Network. The models were trained to predict if the value of a particular stock would increase by 10% in a year or not, given a variety of financial information about the company. The most successful model was the Random Forest, with a score of 0.765 for precision, recall, and f-score [10]. Huang et al. [11] also applied an SVM to forecast the movement of the Nikkei 225 index. For their inputs they utilized the value of the S&P 500, as well as the exchange rate between USD and JPY as features. They compared the SVM to other classification models and concluded that the SVM performed, resulting in an accuracy of 73%, and attribute its success to its resistance to overfitting [11].

Wolff and Neugebauer [12] compared tree-based models with linear regression type models for predicting equity markets. They found that more complex linear regression models like Principal Component Regression and Ridge Regression significantly outperformed tree-based models, with accuracy in the 67-68% range compared to 58-67% respectively. The models' return on investments reflected these accuracies. They note that the results are likely due to a lower number of features and lower frequency of observations compared to other studies. This indicates the importance of both the quality and quantity of data used for training [12]. Leung et al. [13] used structural support vector machines (SSVMs), to predict stock prices based on the financial data of collaborating companies. The SSVM achieved a median testing accuracy of 58.6443%, leading them to conclude that the approach was effective [13].

2.3.1 Hyperparameter Searches

Almost all machine learning methods can be tailored to a particular application through the use of hyperparameters. Hyperparameters are the settings and arguments utilized when initializing machine learning models. Available hyperparameters differ significantly depending on the technique being used. The best parameters for a model may vary substantially based on the dataset and prediction task. The correct selection of these hyperparameters can drastically speed up model training and improve performance. Traditionally, a hyperparameter search is performed by selecting a manually inferred set of good parameters. However, when it is not possible to infer an ideal parameter set, parameter searches are utilised. One very common parameter each technique is Grid Search, which utilizes a grid of acceptable values for each parameter. Grid Search operates by generating the set containing all possible combinations of parameters in the provided parameter grid, following which models will be fitted on each of these parameter combinations. Each model is then ranked based on their performance with respect to a desired metric, and the best parameter set is returned. To improve the reliability of the search, cross-validation is commonly used internally and ranking is done on the mean cross-validation performance. However, Grid Search can be notably impractical when there is a large number of potential parameters, as the search space grows exponentially. It has also been determined that only a small subset of hyperparameters may have a significant impact on model performance, however, it is typically not known which these are for a given model and dataset without performing a search. To remedy this, there have been attempts to create methods that would algorithmically determine the best hyperparameters without having to manually specify a grid to search through [14].

2.3.2 Support Vector Machines

Support Vector Machines (SVM), are a type of supervised classification algorithm. SVMs can be used for classification, regression, and outlier detection tasks. By default, most SVMs are binary classifiers. To perform this task, SVMs attempt to find a hyperplane, or decision boundary, that optimally separates data points based on their class to maximize the distance between opposingly classified points [15]. The support vectors are defined as the points close to the hyperplane, as these influence the hyperplane's position and orientation the most. Unlike Logistic Regression, where output values are scaled to fit in the 0 to 1 range, SVMs scales the values from -1 to 1. SVMs also have the option of being fitted using a variety of kernels, enabling polynomial boundaries or multi-class classification. Notable types of kernels include radial basis function, polynomial, sigmoid, and linear. Regularization can also be applied at the kernel level to reduce bias and prevent overfitting. Each type of kernel has its own parameters—such as the degree of the polynomial for the polynomial kernel—thus requiring hyperparameter searches in many cases[15].

2.3.3 Logistic Regression

Logistic Regression is commonly used for classification problems where the target variable fits into one of two classes. To do this, logistic regression utilizes log odds, or logit—the probability of success divided by probability of failure—are used for prediction to identify a probability for a given outcome for some input. To make accurate predictions, a weight coefficient β for the logistic regression equation is estimated using the maximum likelihood equation (MLE). The equation is optimized by testing different β values through multiple iterations to see which provides optimal results with the log odds equation. Varying the value of β allows us to compute a log-likelihood function, our goal with which is to maximize, theoretically providing with the best value of β for prediction. The optimized model can then be used to calculate a probability for a particular class given some input[16].

2.3.4 Decision Trees

Decision Trees are a commonly used technique for explaining the relationships between data and certain outcomes. To do so, they make use of sequences of boolean operations in a tree-like fashion to model a relationship. Specifically, fitting a decision tree model generates a search tree where each node (split) in the tree represents a boolean operation, such that passing the operation induces rightward traversal, while failing the operation induces leftward traversal. When input data is numeric in nature, most (if not all) boolean operations in the tree are less-than comparisons (e.g. "is the opening price less than $300?" Move right if true, left if false). The endpoint of each path of the tree, called leaf nodes, specify a prediction value. In the case of classification problems, each leaf node will be given a value representing some class. Note that multiple leaf nodes may denote the same class, allowing for highly complex decision boundaries. Altogether, this structure makes decision trees very easily explainable models, allowing them to be used both as predictive and descriptive models.

Decision trees are fitted by splitting the provided training dataset based on a boolean operation which maximize the calculated quality of the split. The two most commonly used metrics of split quality for classification are Gini Impurity and Entropy (aka Log-Loss). This process is then repeated recursively for each newly created node and their corresponding subsets of the original dataset. Most frequently, the technique used to implement this is a greedy algorithm. Recursive splitting down a branch of the tree may end when some criteria is met, such as reaching the maximum acceptable depth, splitting the tree again is no longer valuable, the proportion of the dataset represented by a leaf node is below some threshold, or all inputs at a node have the same output value. This makes decision trees highly customizable in their complexity, without sacrificing their explainability. The downside to this is that decision trees are highly prone to overfitting if made too complex, while also potentially being ineffective if made too simple, requiring the implementer to strike a fine balance in complexity to build a performant model[16].

2.3.5 Random Forest

Random Forest is an ensemble technique which use decision trees. This means that multiple decision trees are fitted and utilized internally to generate a consensus in order to make a prediction. Random forest models use bootstrap aggregation (also known as "bagging") to fit a large set of (ideally) uncorrelated decision trees. The intent with this technique is that while the individual decision trees are not highly correlated with each other, the majority of them will still come to the same (correct) conclusion. This mitigates the mistakes of individual decision trees. The downside to this technique is that random forests are highly resource-intensive to train. Further, it is much harder to explain random forest models compared to individual decision trees, as a single random forest model can consist of a massive number of models, each coming to different conclusions[16].

2.3.6 Adaptive Boosting

Adaptive Boosting (AdaBoost) is a classification algorithm which utilizes the boosting ensemble approach. A boosting algorithm combines many weak predictors with a weighted sum to form an overall stronger predictor. Each weak predictor attempts to learn the correlation between the input and output, but generally is only marginally more accurate than chance. This allows AdaBoost to converge very rapidly. The weak predictors can be any machine learning algorithm. Boosting as a whole is also generally resistant to overfitting. The overall prediction generally has a high confidence after sufficient training [17].

2.3.7 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a commonly used machine learning technique which utilizes a consensus of similar results to make predictions. At its core, KNN identifies the k closest known samples to the current input and uses them to inform its decision. For classification, the output is chosen as whichever class is most common among the identified neighbors. For regression, the output is the mean value of the selected neighbors. With low values of k, small clusters of classes or values can be identified, but overfitting may occur more often. Larger values of k are better at mitigating overfitting and generalizing on a large dataset, but may not be able to account for micro-clusters, enclaves inside the space of another class, or sudden changes in the distribution of values. [18]

2.3.8 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a method of approximating true gradient descent optimization. This technique allows models to be trained quickly on large datasets or problems without a simple closed-form solution. SGD accomplishes this speed increase by calculating the gradient of a model's prediction with respect to multiple subsets, or batches, of the original dataset. The size of a batch impacts the time required for training to converge but does not significantly impact accuracy. To train a model using SGD, two nested iterative loops are performed: In the inner loop

SGD iterates over each available batch, while the outer loop repeats this process as many times as desired or is required before performance stops improving. Each iteration of the outer loop is referred to as an "epoch". For each batch in the inner loop, the model fits and updates its weights or other parameters utilizing the gradient descent with respect to a provided loss function. This results in fewer computations being done overall and therefore faster training. Despite noise introduced by the random sampling often used to select batches, SGD will still converge to a minimum in the loss function[19].

SGD can be used to train a variety of machine learning models and is often provided by machine learning libraries as an alternative to standard linear regression. Common models to use SGD include logistic regression, support vector machines, and perceptrons. SGD is also a commonly used technique for training deep neural networks.

2.3.9 Artificial Neural Networks

Artificial Neural Networks (ANN) refer to the collection of interconnected "neurons" (typically perceptrons) which take a collection of inputs and calculate an output. These neurons have "synapses" or connections between them that define where data flows within the ANN, and how much each specific data point weighs into the final output. As a result, ANNs can be generalized as graphs, where each neuron is a node each synapse is an edge. Edges have weights associated with them, the values of which are optimized to produce accurate outputs. To do this an ANN is treated as a function such that the error, or loss, of this function can be minimized with respect to the weights. Important to note is that the collection of weights and nodes are organized in layers which can be represented using matrix operations. The specific organization of layers and the usage of different operations can be used to define different types of ANNs. Some common types of ANNs include feed-forward networks, recurrent neural networks, and convolutional neural networks, autoencoders [20].

2.4 Deep Learning

Deep learning is an sub-category of machine learning, classified by modifying existing models (most commonly artificial neural networks) by adding multiple layers of computation, each of which are informed by the decisions of other layers. In exchange for substantially increasing complexity, deep learning allows for the modeling of otherwise exceedingly difficult problems. As a result, deep learning is often seen as more sophisticated and as having a greater ability to learn complex relationships when compared to traditional machine learning models [21].

Heaton et al. [22] discuss the potential applications of deep learning in financial prediction, proposing that autoencoders or Long Short Term Memory (LSTM, a type of recurrent neural network, or RNN) models, could be used to predict creditworthiness or predict the movement of a stock index [22]. The LSTM is well suited for predictions on time series, and as such has been studied extensively in financial contexts. Du et al. [23] proposed an Adaptive RNN (AdaRNN) consisting of a module to describe the distribution of data in a time series and an RNN based on the Gated Recurrent Unit (GRU) to predict future values of a time series. They apply the AdaRNN to a variety of contexts including financial prediction. They found that the new model outperformed other forms of RNN as well as Transformer models for financial prediction and other tasks [23]. In a review about the use of machine and deep learning models for financial analysis, Obthong et al. [24] stated that LSTMs were suited for the task due to their ability to exploit complex interactions between features and their memory components.

2.4.1 Recurrent Neural Networks

The Recurrent Neural Network (RNN) is a form of artificial neural network which maintains an internal "memory" state, enhancing its applicability to time series data. It does so by considering the previous output as an additional input by maintaining a "hidden" state. The hidden state produced by a node contains the previous time step's output as well as the current output. These components are weighted, and the output of the current time step is warped by an activation function before being saved for the next time step. Tanh and Softmax are commonly used as activation functions for RNNs. However, both can lead to vanishing or exploding gradients [25]. Due to their efficacy for time series problems, RNNs have been studied extensively for use in financial prediction.

2.4.2 Long Short-Term Memory Networks

The Long Short-Term Memory Network (LSTM) is an extension and modification of the RNN. LSTMs were created to mitigate the vanishing and exploding gradient problems inherent to RNNs. They also improve the long-term memory of recurrent models. The core difference between an LSTM and an RNN is the introduction of gated units in each cell. The LSTM cell consists of three gates, a “forget” gate, an“input” gate, and an “output” gate. The forget gate controls how much impact the previous hidden state and the current input have on the current decision using the sigmoid function. The input gate operates on the input and previous hidden state to determine importance utilizing both sigmoid and tanh activation functions. Finally, the output gate passes the cell’s output through a tanh activation function, passes the previous cell’s hidden state and current input through a sigmoid activation function, and multiplies them together to determine this cell’s current state. The current output and hidden state are then passed to the next time step [26].

RNNs have been used extensively in existing literature, being utilized in roughly 51.0% of financial time series forecasting papers published between 2005 and 2019. Among the papers that used RNNs, roughly 60.4% used LSTMs, providing credence to their functionality at predicting financial performance [27]. LSTMs also have the great advantage of being relatively easy to implement and train using modern deep learning libraries such as Keras (TensorFlow) and PyTorch, compared to many other applicable state-of-the-art models.

2.5 Previous MQP Work

The focus of the work done by the previous Switzerland MQP team was twofold: To develop reinforcement-learning based equity and Forex trading/forecasting models utilizing Deep Q-learning, and to identify whether the inclusion of greater quantities of data may actually result in the degradation of model performance.

2.5.1 Deep Q-Learning

Q-learning is a well-documented and explored reinforcement learning technique which utilizes a recursively updated “Q-table”, describing the cost/reward relationship between certain actions and states within a finite action-space and state-space. This technique allows for a model to estimate the cost or reward of an action taken by the model considering both the current context and potential future contexts. Q-learning operates on an “episode”-basis, where an episode refers to one iteration of the model recomputing the Q-value for its current action-state pair. This means that the longer we train a Q-learning model (i.e. the number of episodes), the more accurate its Q-table, and thus the quality of its value estimations should become.

Unfortunately full traversal of the options is effectively impossible for the vast majority of applications due to the nature of recursive exploration of all possible options within a state-space and action-space. This is because the number of possible scenarios increases exponentially as the state-space, action-space, or number of episodes grows. To mitigate this, deep Q-learning was proposed to replace the Q-table with a deep neural network to simulate the task of estimating and updating Q-values. I.e., rather than computing estimated values using a Q-table and recursively updating that table, we can train a neural network to estimate said values and update the table using gradient optimization techniques. By using a deep neural network in place of a Q-table, we can leverage the capability of neural networks to model extremely complex and unpredictable behavior efficiently.

2.5.2 Data Quantity Impact

For their research, the previous MQP team wanted to test whether giving more data to a model may result in performance *degradation* in certain contexts. Specifically, they hypothesized that increasing the dimensionality of a training dataset may actually increase its sparseness, resulting in overfitting. This hypothesis contradicts conventional wisdom that training with more data will produce better results.

2.5.3 Methodology

To test their hypothesis about the impact of excessive data, the team implemented a deep Q-learning model (described in section 2.5.1) to predict and act on the short-term movement of the S&P 500 stock market index (as approximated by the SPDR S&P 500 Trust ETF, ticker SPY) and GBP/USD exchange rates. The datasets used were pulled from Refinitiv and are described in Tables 1 and 2. For each dataset, the team trained six identical models, where each model was given a different subset features of the datasets. The features each model was trained on are described in Table 3, such that every model n is provided all the features given to model $n-1$ in addition to the stated features. This resulted in a total of 18 different models: 6 trained to trade SPY on a daily basis, 6 trained to trade SPY on an intraday basis, and 6 trained to trade GBP/USD on a daily basis.

Table 1: Datasets used by the previous MQP team.

Market	Target Asset	Related Asset	Unrelated Asset	Data Range	Interval
Stock	SPY	NDAQ.O, DIA	USO, GLD	18 Years	Daily
Stock	SPY	NDAQ.O, DIA	USO, GLD	3 Months	5 Minutes
Forex	GBP/USD	EUR/USD, CHF/USD	JPY/USD, NZD/CAD	18 Years	Daily

Table 2: Features of datasets used by the previous MQP team.

Asset	Target	Related	Unrelated
Features	1-Day Return 2-Day Return 5-Day Return 10-Day Return 21-Day Return Previous Action Previous Price	1-Day Return 5-Day Return 10-Day Return 21-Day Return	1-Day Return 5-Day Return 10-Day Return 21-Day Return

2.5.4 Results

Overall, the agent learned to buy more and short less over time. As the models were given more information, they became more informed and developed in skill, especially as episodes increased numerically (as time progressed). For index funds, earlier models bought 100% of the time after episode 300. Later models strategized, buying 70% of the time and shorting or holding 30% of the time. Overfitting began in models 4-6, but was not so severe that the success rate decreased. Similar trends were present in the Forex models. Model 1 always held, Model 2 always shorted the exchange, Model 3 and beyond learned to buy and sell respectively. Models 4-6 performed worse than 1-3 due to much more severe overfitting. The intraday results had a lesser return, but still beat the market return. The models developed their strategy in similarly to the previous interday models for both index fund and Forex data. There was still overfitting in later models and return increase was only seen in models 5-6, though optimal performance was seen in models 2, 3, and 4.

Table 3: Features of provided to each model trained by the previous MQP team.

Model #	Asset	Features
1	Target	1-day return
2	Target	Previous action
3	Target	Previous price
4	Target	2-day return 5-day return 10-day return 21-day return
5	Related	2-day return 5-day return 10-day return 21-day return
6	Unrelated	2-day return 5-day return 10-day return 21-day return

3 Methodology

3.1 Data

To collect our data, we made use of the EOD Historical Data API[1]. This service provides historical data for a wide variety of assets. Our initial data set was composed of data from the following five markets: stocks, foreign exchange, bonds, stock futures, and commodities. The stock market dataset only contains time series data of stock equity. The Forex dataset is a collection of time series tracking the relative exchange rates between a number of currency pairs. The bond dataset is a collection of time series tracking the value of government bonds issued by a variety of countries (with the addition of 30-year bonds for the United States[2]). The index and commodities futures datasets are collections of time series tracking the prices of futures for commodities as well as and futures tracking a variety of stock market indices. A full breakdown of the initial dataset by asset type can be seen in Appendix A. Daily pricing data was pulled for all of our selected assets, with a desired start-date of January 1, 2000 and end-date of January 1, 2021. The features of each time series include date, open price, closing price, high price, low price, volume (where available), and adjusted closing price (closing price adjusted for splits and dividents[3]). Specific assets in the initial dataset were chosen using the following reasoning:

- **Stocks**
  For stocks, we decided to exclusively analyze the SPDR S&P 500 Trust ETF because it is our target for prediction and closely mimics the S&P 500 index. Further, we chose not to include other stocks or ETFs because our focus is on the impact of intermarket data, to which the inclusion of additional stocks may make it harder to identify.
- **Forex**
  We chose all foreign currencies used in at least 5% of foreign exchange transactions, as reported by the Bank for International Settlements in 2019 [5]. The set of exchange rates were then created by generating all possible pairs of currencies.
- **Government Bonds**
  Governments bonds were chosen by selecting the top five countries by proportion of total central government marketable debt as reported by the Organisation for Economic Co-operation and Development (OECD) in 2019 [6]. Each country had data for 3-, 5-, and 10-year bonds, as well as 30-year time to maturity for the United States.
- **Commodities Futures**
  Our selection of commodities futures to analyze were decided based on their relative liquidity as measured by open interest and volume. We selected the top 5 commodities futures contracts by open interest and the top 5 commodities futures contracts by volume (all of which overlapped except for heating oil and soybean futures) as reported by CME Group[4].
- **Index Futures**
  We used a similar technique for index futures as used for commodities futures. Specifically, we included the E-mini S&P 500 as it was #1 by volume and #5 by open interest across all futures contracts (not just commodities), and #1 in among equity futures. We believe this may be beneficial as it is possible the value of S&P 500 futures may have different underlying information from the historic data of the S&P 500 itself (as approximated by SPY). E-mini Russel 2000 was included so as to increase the breadth of our dataset by consider small-cap indices, in addition to being the #2 equity future by both volume and open interest. We also looked at EURO STOXX 50, Nikkei 225, and Hang Seng Index Futures to allow us to consider international markets, which are notably different from the American market. Lastly, we included CBOE Volatility Index futures as an alternative measure to broaden the types of index futures explored.

3.2 Exploratory Data Analysis

Our initial dataset was unnecessarily large, making it infeasible to use with most machine learning models, in addition to containing redundant and/or unhelpful data. To filter down to a smaller, more appropriate dataset, we conducted

[1]More information available at https://eodhistoricaldata.com/

[2]United States 30-Year bond data only available as futures from the EOD Historical Data API

[3]More details available at https://eodhistoricaldata.com/financial-apis/adjusted-close-and-close-whats-the-difference/

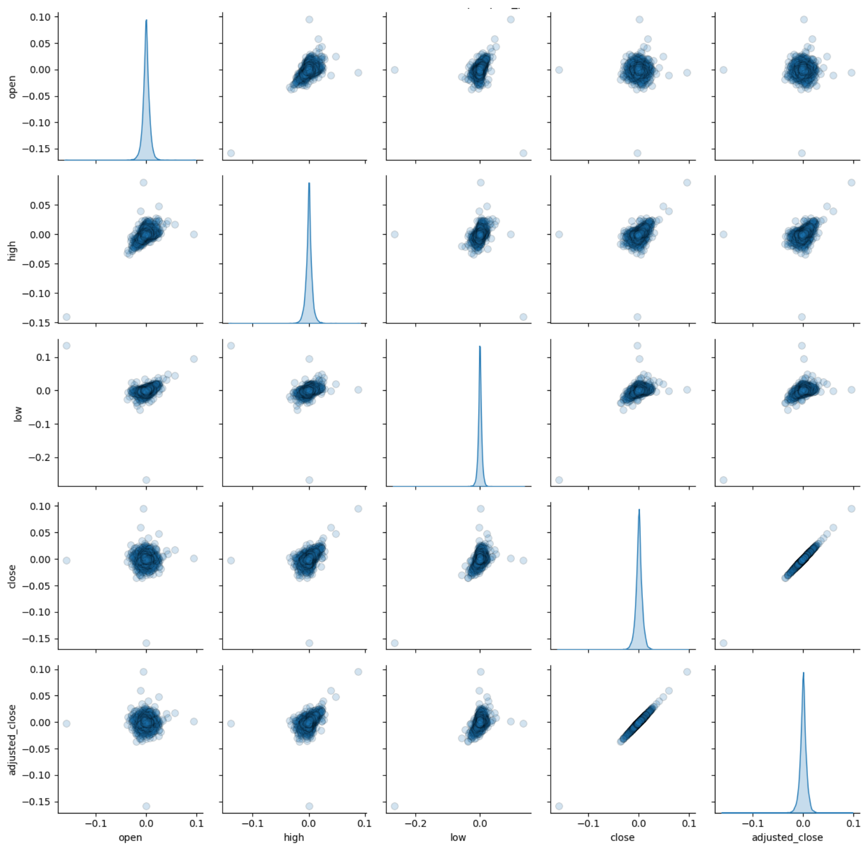
[4]When accessed at www.cmegroup.com/markets/products.html on September 6, 2022

exploratory data analysis (EDA). The goal of this EDA was to determine which time series were the most likely to be relevant and tailored toward prediction performance, as well as consider the use of certain transformations. To do this, we searched for correlations between variables, assessed the normality of our data before and after transformations, considered the impact of principal component analysis, and analyzed the presence of holes and outliers in our data. This analysis was conducted both on the raw data and on calculated day-over-day percent-change data.

3.2.1 Correlation

In order to examine the correlation trends of the dataset, pair-plots were generated for each asset paired with every other asset, both using raw data and percent-change adjusted data. An example pairplot is visible in Figure 1. Self-correlation and covariance were visualized using confusion matrices. Confusion matrices were also generated displaying correlation between the stock data and each of the other assets. An example confusion matrix displaying correlation with stock data is available in Figure 2. We utilized Pearson correlation coefficient r in our correlation assessments.

Figure 1: Pair-plot of percent-change in CAD-CHF features



In terms of the percent-change bond data, the US 5- and 10-year bonds appeared highly self-correlated for many features, as is visible in Figure 3. US 3-year bonds were slightly less self-correlated, displaying no substantial open/close price correlation. The UK bond data all had high self-correlation, excluding opening and closing prices. French bond data displayed effectively no self-correlation. The Italian 5-and 10-year bonds had high self-correlation across all features, while 3-year bonds had low self-correlation across all features. The Japanese bond data had little self-correlation across all bonds, with 3-year bonds only displaying correlation between low and close prices and between open open high prices, and with 5-year bonds displaying correlations between low and high prices. In general, the raw bond data attributes all correlated positively with the SPY volume data. The closing price and adjusted closing price also had a general positive correlation for bond data and Forex data.

The Forex data displayed correlation above 0.5 in most cases for high versus close prices, low versus open prices, and low versus close prices in the percent-change data. A few exceptions include EUR-GBD not displaying correlation for low versus open prices or low versus closing prices, and EUR-CHF not having correlation between low and open price.

Figure 2: Confusion matrix of correlation between percent-change of Nikkei 225 futures (NK) and SPY

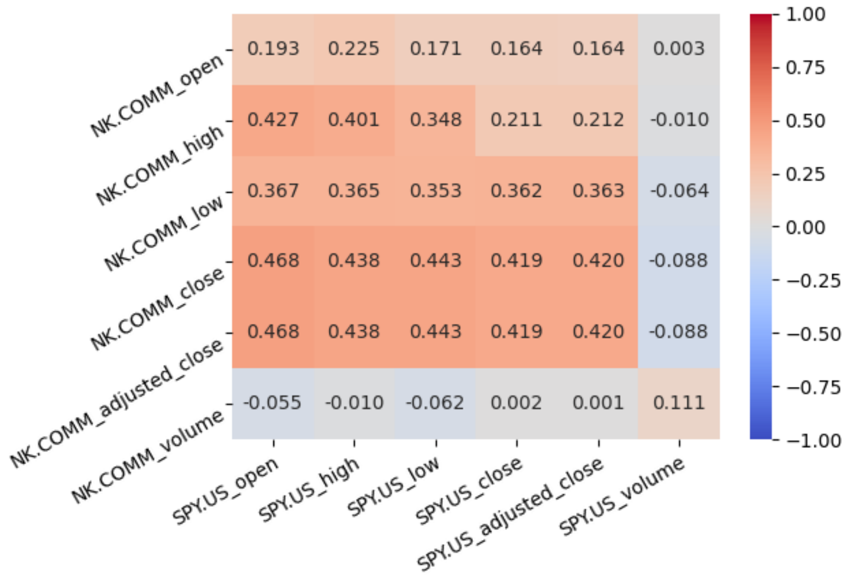
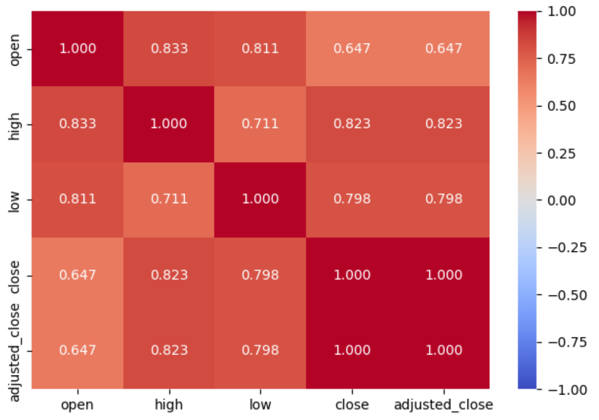


Figure 3: Self-correlation confusion matrix of US 5-year bonds.



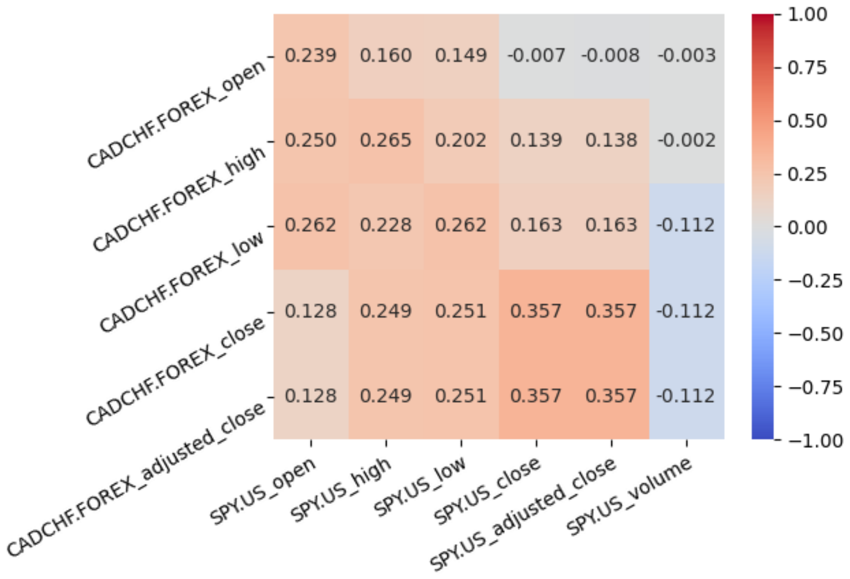
AUD-CAD exhibited strange behavior in having no correlation between low and close prices or high and open prices, but very high correlation between low and open prices, as well as a slight correlation between close and high prices. All of the Forex data had little to no correlation with SPY, with common behavior being visible in Figure 4. The only notable exception to this was a minor correlation between AUD-CHF's close prices and SPY close prices. USD-AUD closing price versus SPY closing price correlation was another unusual instance because there was an unexpected negative correlation of -0.456.

The futures data all followed similar trends; exhibiting self-correlation for high versus open prices, high versus close prices, and low versus close prices on most assets. An example of the commonly seen behavior is visible in Figure 5. There was also notable correlation with SPY on low, high, close, and adjusted close prices using RTY, ES, and NK, whereas minimal correlation was seen with the other futures assets.

3.2.2 Normality

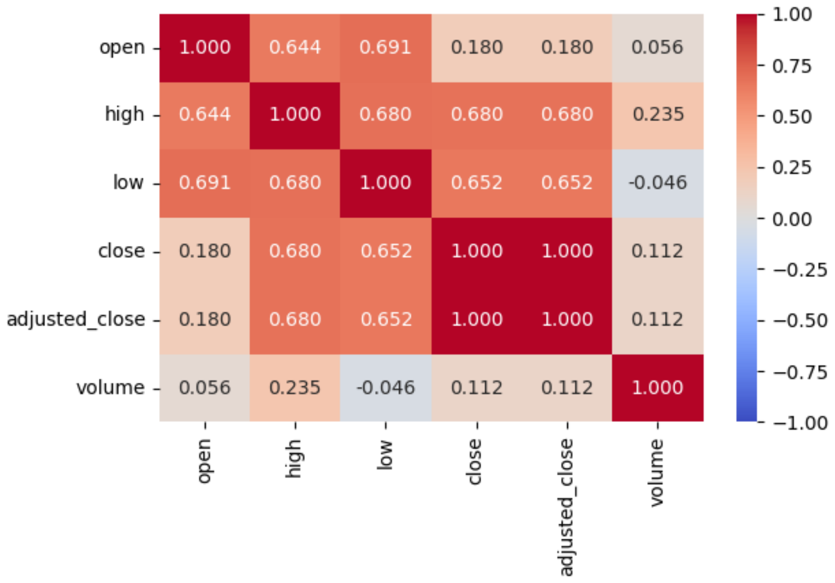
To understand the distribution of our data, we performed a suite of visual and statistical assessments of our data. We constructed kernel density estimate (KDE) plots [28, 29] to visualize the general distributions in our data, as well as

Figure 4: Confusion matrix of correlation between percent-change of CAD-CHF Forex and SPY



	SPY.US_open	SPY.US_high	SPY.US_low	SPY.US_close	SPY.US_adjusted_close	SPY.US_volume
CADCHF.FOREX_open	0.239	0.160	0.149	-0.007	-0.008	-0.003
CADCHF.FOREX_high	0.250	0.265	0.202	0.139	0.138	-0.002
CADCHF.FOREX_low	0.262	0.228	0.262	0.163	0.163	-0.112
CADCHF.FOREX_close	0.128	0.249	0.251	0.357	0.357	-0.112
CADCHF.FOREX_adjusted_close	0.128	0.249	0.251	0.357	0.357	-0.112

Figure 5: Self-correlation confusion matrix of natural gas futures (NG).



	open	high	low	close	adjusted_close	volume
open	1.000	0.644	0.691	0.180	0.180	0.056
high	0.644	1.000	0.680	0.680	0.680	0.235
low	0.691	0.680	1.000	0.652	0.652	-0.046
close	0.180	0.680	0.652	1.000	1.000	0.112
adjusted_close	0.180	0.680	0.652	1.000	1.000	0.112
volume	0.056	0.235	-0.046	0.112	0.112	1.000

quantile-quantile (Q-Q) plots [30] to identify similarity to the normal distribution. For statistical assessments, we used D'Agostino-Pearson normality tests [31] (all values for which are available in Appendix B).

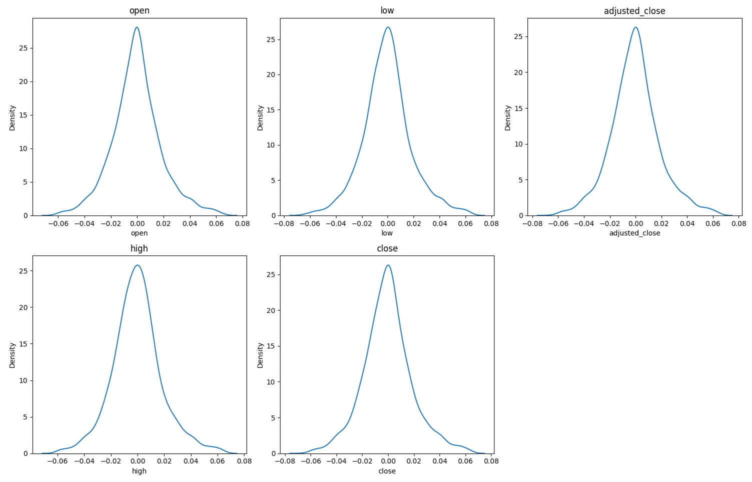
Raw Data: Based a visual analysis of the KDE plots, none of the features of the raw dataset appeared similar to the normal distribution in most cases. Most of the time series were multi-modal or had substantial skew. The only data that were unimodal were JPY-GBP, JPY-CAD, and AUD-CAD exchange rates. However, these assets all exhibited a negative skew. Removing outliers had minimal impact on the KDE plots. Interestingly bond data was consistently bimodal, with most distributions having a larger peak at the lower end of their range and a smaller peak at the higher end. By contrast, Forex and futures data was much less consistent in behavior.

From the Q-Q plots, it appeared that most of the data displayed little similarity to the normal distribution, aligning with prior conclusions from the KDE plots. Most of the data produced similarly shaped Q-Q plots, identifying multi-modality and significant skew (especially with respect to volume data, where available). Removing outliers had minimal effect on the Q-Q plots. A couple of assets appeared to potentially display similarity to the normal distribution, only showing skews towards the very outer quantiles or having other unusual anomalies. Specifically, EUR-AUD, EUR-CAD, AUD-CAD, JPY-CAD, and JPY-CHF exchange rates; natural gas futures; and UK 5- and 10-year bonds all displayed similarity to the normal distribution through much of their range.

Statistical normality testing was conducted with outliers removed, as it was clear most outliers were anomalous and had substantial impact on statistical tests. Our testing confirms that the vast majority of the raw data is non-normal, as p-values for all features of almost the entire dataset were lower than any reasonably significance level by multiple orders of magnitude. The lone exception to this was UK 10-year bonds, which failed to display significant non-normality at the $\alpha = 0.05$ significance level for three features, and $\alpha = 0.01$ for all five available features, receiving p-values of 0.0465, 0.0764, 0.0303, 0.0507, and 0.0507 for "open", "high", "low", "close", and "adjusted_close", respectively.

Percent-Change: Much of the percent-change data appears mostly normally distributed based on the KDE plots. However, it is visible that many extreme outliers are present due to long tails on many plots. Removing the outliers alleviated this problem. While the distributions appeared mostly normal, they curved much more sharply than a standard normal distribution, resulting in a very sharp change in density at the peak frequency (centered roughly around 0 for the price-based features of all assets). The most common behavior is visible in Figure 6. Some assets displayed extremely unusual behavior: JPY-AUD and JPY-CHF exchange rates displayed multi-modal behavior, with medium-sized peaks around ± 0.01 and smaller peaks around ± 0.015 on most price-based features (visible in Figure 7).

Figure 6: Kernel density estimates of features of US 10-year bonds.



The Q-Q plots for percent-change data indicate that most of the data is roughly normally distributed, with the exception of significant skews at the high and low quantiles. With outliers removed the skew is much less apparent, but still present in a minor capacity. We believe the skew seen in the Q-Q plots is a result of the sharpness of the curves seen previously in the KDE plots. In general, almost all the percent-change data displays a noteworthy similarity to the normal distribution once outliers have been removed. The most common behavior is displayed in Figure 8. However, JPY-AUD, JPY-CHF, and JPY-CAD display anomalous behavior, having multiple unusual steps (visible in Figure 9).

Figure 7: Kernel density estimates of features of JPY-AUD exchange rates.

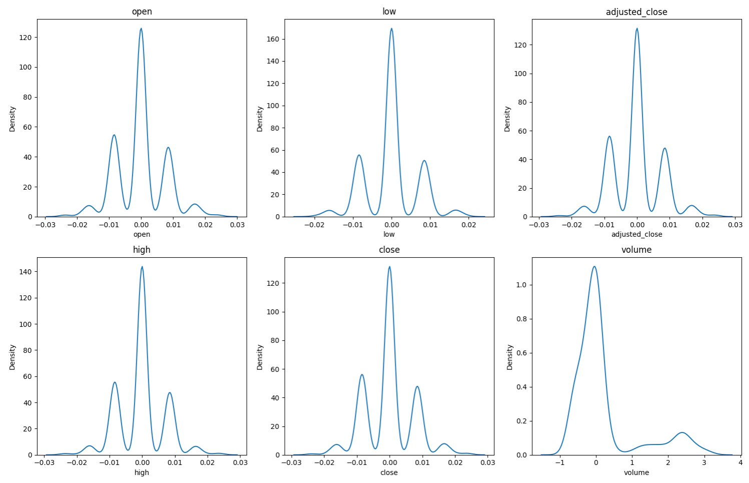


Figure 8: Quantile-quantile plot of features of UK 10-year bonds compared to the normal distribution.

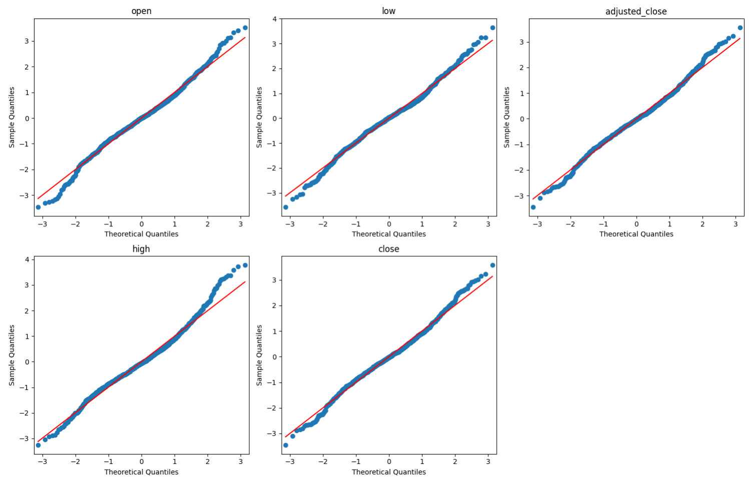
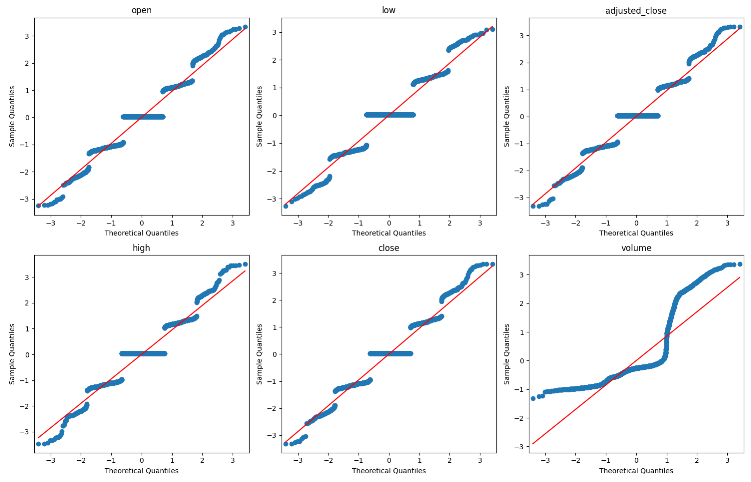


Figure 9: Quantile-quantile plot of features of JPY-AUD exchange rates compared to the normal distribution.



After performing statistical normality tests, we were still unable to identify normality for the features of most assets. Most of the data received p-values lower than any reasonable significance level by multiple orders of magnitude. However, there were a handful of exceptions: EUR-JPY exchange rates received p-values greater than any reasonable confidence level, suggesting potential normality, with p-values of 0.292, 0.188, 0.848, 0.610, and 0.610 for "open", "high", "low", "close", and "adjusted_close", respectively; and JPY-CAD exchange rates displayed similar behavior, with p-values of 0.286, 0.169, 0.445, 0.545, and 0.545. The latter of these two cases is particularly unusual, as JPY-CAD displayed notably unusual behavior on both the KDE and Q-Q plots.

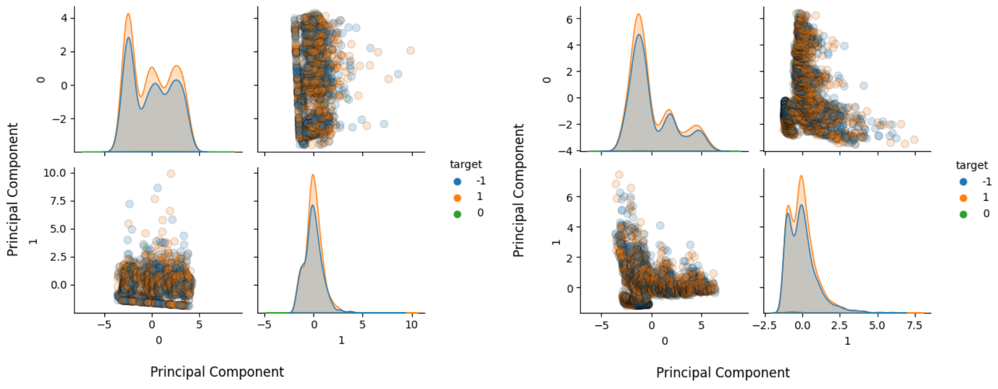
Conclusions: Based on our assessments for normality, it is clear that to make use of techniques which are dependent on normally distributed data, the raw data will likely be problematic or require normalization. By contrast, the percent-change adjusted data appears to be potentially quite reliable for normality-dependent techniques with minimal modification/normalization. We conclude it would be a good idea to omit Forex assets utilizing the Japanese Yen with the exception of USD-JPY, as exchange rates involving the Yen frequently display strange and unexplained behavior.

3.2.3 Principal Component Analysis

To see if our data could be simplified to fewer features and identify the usefulness of these minimal features toward predicting S&P 500 movement, we made use of principal component analysis (PCA). For each asset we generated the minimal number of principal components required to describe a minimum of 95% of the variance in the data. When only one principal component was generated, a KDE plot was constructed. When multiple principal components were generated we constructed pair-plots[5] between the components. For all plots we colored the individual data points and distributions by their corresponding future S&P 500 movement, where -1 indicates downard movement, 0 indicates no movement, and 1 indicates upward movement.

Raw Data: When applied to the raw data, PCA successfully simplified all of our assets to at most two principal components, with most only requiring one principal component to describe 95% of the variation in the data. That said, none of assets appear to display a visual separation between upward and downward movement of the S&P 500 as a result of PCA. It is worth noting that a few assets display unusual behavior in their principal components, such as having small isolated clusters or large amounts of data falling along a straight line. Interestingly, the SPY data specifically displayed unusual behavior having a small cluster separated from the majority of the data in addition to displaying a curve which was not seen from any other asset. Examples of unusual behavior, including SPY, is visible in Figure 10.

Figure 10: Principal component pairplots on raw data displaying unusual behavior
Left: Nikkei 225 futures. Right: SPY S&P 500 Index Fund.



Percent-Change: When applied to the percent-change adjusted data, PCA simplified almost all of our assets to three or four principal components. Exceptions to this occured with JPY-GBP exchange rates, which only had one principal component, and 5- and 10-year US bonds, which had two principal components. None of the assets displayed any notable separation of movement direction of the S&P 500 as a result of PCA. Some common behavior can be seen in Figure 11. Two assets (US 5- and 10-year bonds) displayed unusual behavior, having a portion of data in a straight line running through the center of their principal components (see Figure 12). This is interesting as these assets are also the only to require only two principal components to describe 95% of variance.

[5] As provided by the Seaborn Python library. See https://seaborn.pydata.org/generated/seaborn.pairplot.html.

Figure 11: Principal component pairplots on percent-change data.
Left: Nikkei 225 Futures. Right: SPY S&P 500 Index Fund.

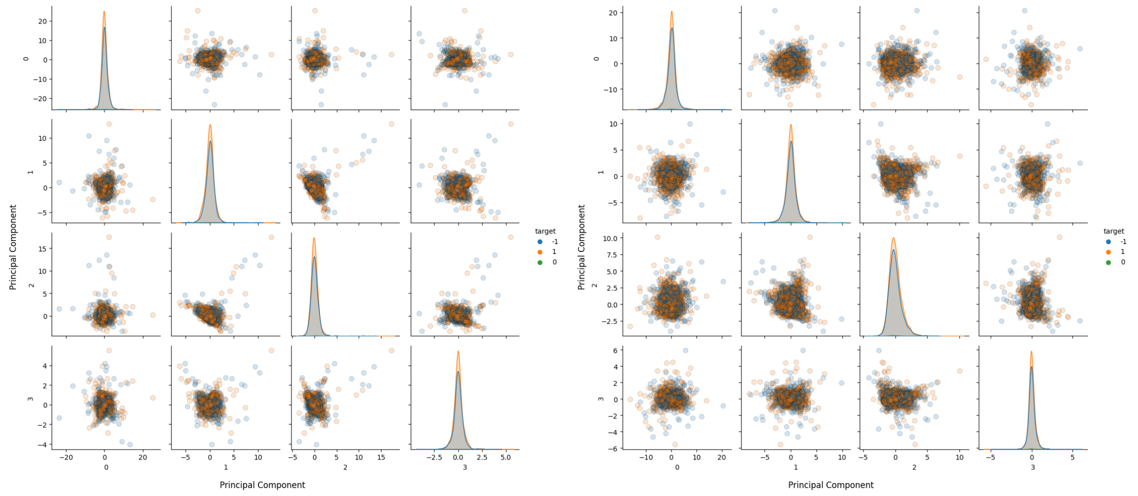
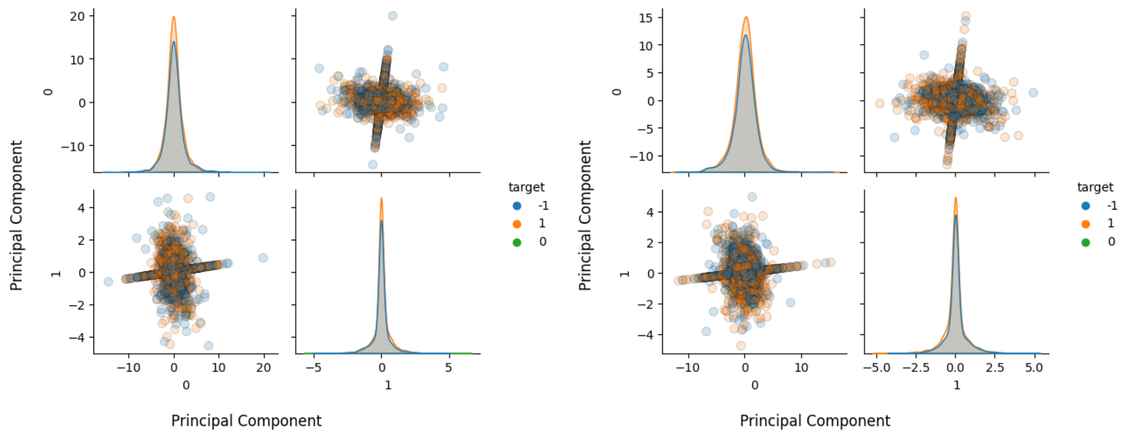


Figure 12: Principal component pairplots on percent-change data displaying unusual behavior.
Left: US 5-year bonds. Right: US 10-year bonds.



Conclusions: Based on our PCA, it is unclear whether applying PCA is beneficial towards classification. However, PCA consistently reduced the total number of features for each asset, as well as making much of the data appear substantially closer to being normally distributed, simplifying the data substantially. That said, a couple of assets display notably unusual behavior, suggesting care should be given when deciding which assets are selected for model fitting and which techniques may have difficulty handling these anomalies.

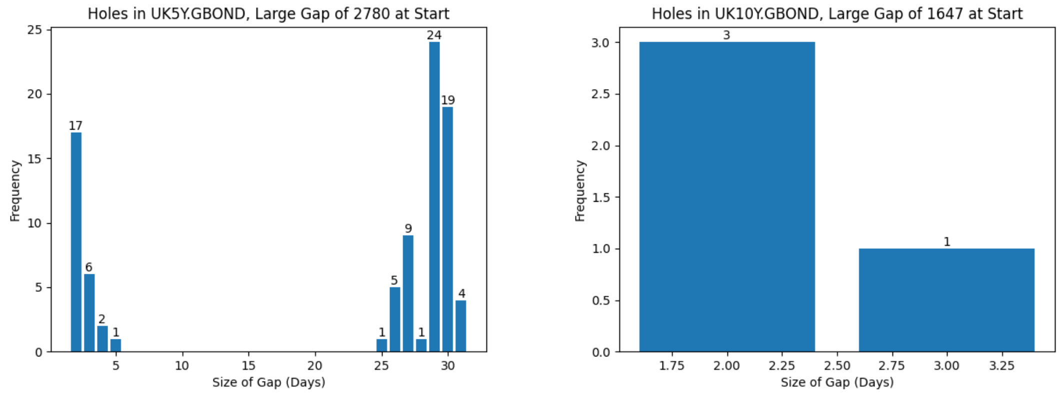
3.2.4 Hole Analysis

Data was missing from many of our time series. The first case where data was missing was when the API did not record time series data until after January 1, 2000. This case is particularly prevalent in our government bond data, where data does not exist until 2500 days after January 1, 2000. US 5 and 10 year are not missing any data from this starting date. However, this issue is present with other assets, namely index and commodities futures. HO, HSI, NG, NK, RTY, and VIX all have over 2000 days of missing data from the start of their time series. The Forex data was excellent, with all but one currency pair not having more than 0.05% of data missing.

More observations were made on the data when initial gaps were ignored. Firstly, gaps in data due to weekends were ignored to ensure fairness. The remaining gaps are gaps due to holidays or issues with the API data collection. The first obvious trend is that some of our government bonds are missing approximately 40% of the requested data (excluding the large initial gaps). Shown in figure 13, UK 5 year bonds have many holes, averaging at about 21 missing days per hole, with a standard deviation of 12 days. Some of the government bonds do not have such holes. The US 5 and 10 year bonds, IT 10 year, as well as UK 10 year bonds have reasonable quality data within the range where initial gaps are ignored. The UK 10 year bond holes are shown in figure 13. Our commodities data is much better when the initial gap is ignored, with no more than 6% missing data in the expected time range. NK and RTY had especially few holes, with less than 1% of expected data missing. Forex data was once again excellent, with no more than 0.3% of expected data missing.

Overall, our Forex data was excellent. Every time series had less than 0.2% of data missing in the twenty years of requested data, with the exception of GBP-AUD. Our SPY data was also excellent, with just 3% of data missing in the same period. Our commodities data suffered, as many assets were missing large amounts of data in the initial years of requested data. Our government bond data was largely poor, excluding US 5 and 10 year bonds. In future projects, a new data source would be recommended for end of day government bond data.

Figure 13: Distribution of the size of holes in example time series.
Left: UK 5-year bonds. Right: UK 10-year bonds.



3.2.5 Outlier Analysis

Outliers for percent-change in each time series were identified and analyzed based on their deviation from the mean of each feature. For our analysis, any value greater than 3 standard deviations away from the mean was considered an outlier. For raw data, JPY to AUD forex had the most outliers, with 306. This was 5.16% of the entire dataset. USD to GBP forex had the second most with 239, or 4.16% of the whole dataset. Percent-change data had more outliers on average, but a lower maximum. The EURO STOXX 50 futures dataset had the most outliers, with 269 (5.29% of the dataset), followed by the S&P 500 with 247 (4.91% of the dataset). Given the size of the datasets overall, we believe outlier counts in the low hundreds should not be much of a concern. That said, the magnitude and frequency of outliers may impact their effect on training behavior. The larger the value and the more frequent the outliers, the greater the negative impact they may have on training if included. To analyze this, we looked at the distributions of values in outliers observed. For the SPY data, outliers are within $\pm 10\%$, as visualized in Figure 14. This is fairly high, as almost three-quarters of the distributions only have smaller outliers, such as the USD-EUR Forex data, whose outliers varied within the range of roughly $\pm 4\%$, as seen in Figure 15. Natural gas and soy futures exhibit the highest outliers observed from percent-change transformed data, falling between values of roughly $\pm 25\%$ for some features, as seen in Figure 16. Due to the relative infrequency of these outliers, we believe even large outliers should not have a significant impact on model training. For a table containing all outlier counts, see Appendix D.

3.3 Finalized Data

After performing EDA, we used our results to create subsets of Forex pairs, government bonds, index futures, and commodity futures that we believed will be most useful for our models. These subsets construct out finalized dataset. For each asset type, we selected four to six individual assets based on qualities observed during EDA. Specifically, When considering assets, we wanted assets that displayed normality (as many machine learning models perform better on normally distributed data), demonstrated high correlation with SPY, had minimal holes, and displayed notable

Figure 14: Distributions of outliers in SPY percent-change data by feature.

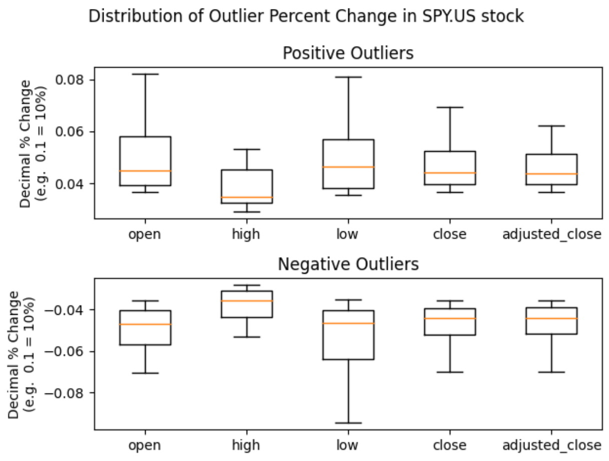


Figure 15: Distribution of outliers in USD-EUR percent-change data by feature.

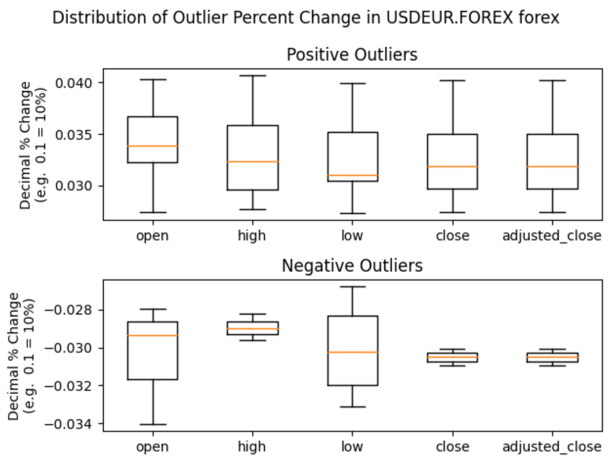
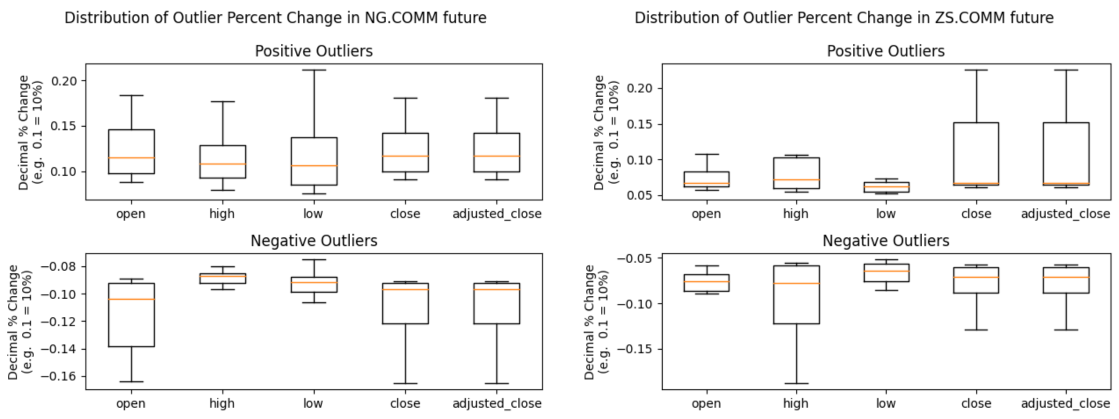


Figure 16: Distributions of outliers in commodities futures percent-change data by feature.
Left: Natural gas futures. Right: Soy futures.



behavior in PCA. We also chose to include some assets for their lesser entanglement with the United States economy. The selected assets are listed in Table 4.

An algorithmic description of how we generated our asset-based datasets can be seen in Algorithm 1. First, we constructed the power set ($\mathcal{P}$, line 3) of available asset types, excluding stocks. Each element of the power set then had SPY data inserted (X, line 4). This resulted in a total of 16 unique asset-type combinations. By training and testing with every possible combination we can evaluate the mean impact of including a specific asset type in a dataset independent of other asset types, as well as considering the interactions between multiple asset types. We also generated a handful of random data baseline datasets. Specifically, two random data baseline datasets were generated, one containing Brownian motion data, and the other containing random samplings from a normal distribution. The underlying distributions used to generate these baselines were designed to mimic the underlying distribution of SPY.

Table 4: Assets included in the final dataset

Asset Type	Asset
Stock	SPDR S&P 500 Trust (SPY)
Forex	USD/EUR USD/CAD USD/GBP USD/JPY EUR/GBP
Bond	US 5-Year US 10-Year US 30-Year Italy 10-Year Japan 10-Year United Kingdom 5-Year
Index Futures	E-Mini S&P 500 (ES) EURO STOXX 50 (FESX) Hang Seng Index (HSI) Nikkei 225 (NK) CBOE Volatility Index (VIX)
Commodities Futures	Gold (GC) Natural Gas (NG) Corn (ZC) Soy (ZS)

Algorithm 1: Asset-based dataset creation

Input: S&P 500 data s, list of intermarket asset types I
Output : List of asset-based datasets D

```
1  $Y \leftarrow \texttt{Shift}(\texttt{Sign}(\texttt{PercentChange}(s_{\text{closing price}})))$   // Generate base market movement values $Y$
2  $D \leftarrow \varnothing$   // Initialize the list of datasets $D$
3  for $A \in \mathcal{P}(I)$ do   // For each possible subset of asset types (power set)
4      $X \leftarrow A \cup \{s\}$   // Set input data $X$ to subset asset types and S&P 500 data
5      for $X' \in \{X, \texttt{PercentChange}(X)\}$ do   // Using $X$ or percent-change of $X$
6          $X' \leftarrow \texttt{Standardize}(\texttt{Align}(X', Y))$   // Align data by available days and standardize it
7          for $X'' \in \{X', \texttt{PCA}(X')\}$ do   // Using $X'$ or PCA-transformed $X'$
8              $D \leftarrow D \cup \{\texttt{Window}(X'')\}$   // Insert new dataset and PCA-transformed dataset into $D$
9          end
10     end
11 end
12 return $D$
```

For each dataset, we considered the usage of two transformations: percent-change and PCA. For each asset-based dataset, we generated new datasets using different combinations of these transformations. I.e. each dataset had raw, percent-change, PCA, and percent-change into PCA versions, resulting in a total of 64 ($4 \cdot 16$) asset-based datasets. The transformations operate as follows:

- **Percent-Change**
  Following the percent-change transformation, each data point represents the percent the value of a given feature changed from the previous recorded item. In our implementation we chose not to interpolate in the presence of holes, instead making the percent-change of data points immediately following holes be the percent-change across the full length of the hole. This transform is represented by the `PercentChange` function at line 5 of Algorithm 1.
- **PCA Decomposition**
  This transformation is used to reduce the dimensionality of our dataset by using only principal components, as described in Section 2.2.2. This is represented by the `PCA` function in line 7 of Algorithm 1.

Before applying additional transformations (i.e. PCA) or using the data with our models, all raw or percent-change transformed data were aligned by their available days of data and standardized. This is represented by the `Align` and `Standardize` functions on line 6 of Algorithm 1. From our EDA, we identified that many of our time series have gaps lasting multiple days. Further, our data source did not provide data starting at the same date for every asset analyzed. For datasets containing multiple assets, not only must holes in data be accounted for, but differing starting dates must be acknowledged. In our implementation, we chose to simply remove data for any dates where there is at least one data point missing in any of the assets in the dataset. As a result, the remaining data contains only dates where accurate data exists for all specified assets within the dataset. Following alignment, standardization was applied using the formula described in Equation 3.1. In this equation, X refers to our input data (raw or percent-change adjusted), μ and σ refer to the mean and standard deviation of the features of the dataset respectively, and Z refers to the final standardized valued.

$$Z = \frac{X - \mu}{\sigma} \tag{3.1}$$

Usable prediction labels must be constructed for the datasets as well. To do this, we calculated the sign of day-over-day change in SPY. This can be seen in line 1 of Algorithm 1. This gave us a set labels where -1 indicated downward movement, +1 indicated upward movement, and 0 indicated no movement at all. Note that in our implementation we decided to convert all 0s to -1, as this allowed us to treat the problem as binary classification and slightly mitigate class imbalances due to upward movement having more instances than downward movement. These values were then shifted backwards by one recorded day, making the label represent to the direction of movement between the recorded day and the following recorded trading day. Note that in the presence of holes in the dataset, this has the side effect of making some intervals require predicting multiple trading days into the future, instead of just one. However, this case is relatively uncommon due to the small proportion of gaps in our dataset, and thus was treated as negligible. For each dataset generated we then aligned the prediction labels with the dataset. Note that for datasets containing assets which start recording later than SPY this means we are ignoring large amounts of labels at the start of our range, as we do not have the information required to predict them.

Lastly, before our datasets are split into training- and testing-sets, we need to transform the data such that each output date has multiple days worth of input data. To do this, we employ a sliding window approach to generate our input/output pairs. This is represented by the `Window` function in line 8 of Algorithm 1. To go into detail, if we have n days (samples) worth of data, and we wish to use p days of data per prediction (which we refer to as the "period"), where $p \ll n$, then we will have $n - p$ total samples in our final dataset. Note that we lose the first p days worth of output labels in the dataset due to requiring a minimum of p days worth of data for a valid prediction. This means that in order to predict y_t (i.e. the movement of the market from day t to $t + 1$) we must use input values in the range $[X_{t-p}, X_t]$. In this paper we exclusively utilized a period of 5 days. As an example, this means that in order to predict the movement of the market following day $t = 7$, we will provide the model with data from and including days $t = 7 - 5 = 2$ through $t = 7$.

Once our windows are created our datasets can finally be split into training- and testing-sets. We split our data with an such that 80% of each dataset was used for training (in-sample) while 20% was used for testing (out-sample). Due to the temporal nature of the data, instead of using a randomized split, we split the data chronologically. I.e. the training sets contained the first 80% of our data, while the testing sets contained the remaining 20%. This is done to negate potential bias due to randomized splitting creating test-sets that exist inside the range of the training-set, making it an unrealistic experiment (interpolation), as our goal is to predict future performance (extrapolation). It is worth noting that this has two notable side effects: Firstly, because we compute the sliding-window before splitting the data, the first p samples in the testing sets will contain data that also existed in their corresponding training set. We allowed this as it enabled us to have a continuous range of predictions through our entire available data range, and we believe any impact on results would be negligible since $n \gg p$ for all datasets. Secondly, because we used a chronological split, training the models is an interpolation task, while testing is an extrapolation task. This may impact the comparative efficacy of certain models or techniques, however we did not explore this possibility in our work.

3.4 Models

Multiple models were implemented and tested on our collection of datasets. The following models were considered when choosing which models to test:

- Support Vector Machine
- Logistic Regression
- Stochastic Gradient Descent Classifier
- Shapelet Transform Classifier
- Adaptive Boosting
- K-Nearest Neighbors
- Decision Trees
- Random Forest

We determined some simple criteria for choosing which models we would use to test our datasets. The first criterion was model popularity: Every model tested must be widely-used in the context of time series prediction and/or financial forecasting. The next criterion was model complexity: Since the goal of this project is not to develop a state-of-the-art model, we wanted our models to be simple, thoroughly-explored techniques which are relatively quick to train and test. Finally, we wanted our models to be distinctly different so as to provide substantial variety in our results and potentially analyze whether the inclusion of certain assets proved more beneficial toward some models than others.

With these criteria in mind, we chose the following models for testing:

- Support Vector Machine
- Logistic Regression
- K-Nearest Neighbors
- Decision Trees
- Random Forest

3.5 Hyperparameter Search

Each dataset in our collection is distinctly different from the next in the number of input features, trends, and other qualities. As such, we believe that fixing model hyperparameters across all datasets may induce unfair (dis)advantages for different datasets and/or models. To mitigate this and ensure we are comparing the best possible models for each dataset, we performed exhaustive hyperparameter searches[6] for each model-dataset pair.

To limit any bias being introduced by the hyperparameter searches, we employed 5-fold time-series-aware cross-validation[7] (CV) during hyperparameter searching. This helps us ensure that the chosen hyperparameters for a model are truly optimal for the dataset based on extrapolation performance. The hyperparameter search iterates through all possible combinations of the provided hyperparameters. For each available hyperparameter combination, the macro F1 scores for each CV fold are calculated and averaged, creating the CV score. The search then compares all of the observed CV scores and chooses the hyperparameter set which resulted in the highest score. Note that where certain hyperparameter combinations are invalid we treated their CV score as zero.

For Decision Trees, we chose to vary the splitter, max depth, minimum number of samples to split a node, and the minimum number of samples required to create a leaf. The splitter can be set to choose randomly or chose the most

[6]As provided by the Scikit-Learn Python Library. See https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[7]As provided by the Scikit-Learn Python Library. See https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html.

optimal split based on a metric (Gini impurity by default). The maximum depth controls how complex the tree can be, with None meaning to expand the tree with no depth limit. The minimum number of samples at which to split an internal node and the minimum number of samples required to create a leaf control the splitting behavior of the tree. Varying these can significantly alter the shape of the tree and therefore its potential performance. For Random Forest we varied the hyperparameters identically to decision trees, with the addition of varying the number of estimators (the number of trees in the ensemble) to use. Varying the number of estimators allows us to vary the importance of individual trees in the ensemble and potentially produce more uncommon results.

For Support Vector Machines (SVM), we opted to exclusively utilize a linear kernel, as the running time of other kernels made their use impractical. The kernel is a function which changes the behavior of the generated classification boundary. However, most non-linear kernels were too complex to train within a reasonable time-frame. We also varied penalty techniques, regularization parameters, and loss functions.

For K-Nearest Neighbors, we varied the number of neighbors, the distance-weighting technique, and the distance metric. Increasing the number of neighbors reduces the chances of overfitting but decreases the precision of the model, making it difficult to identify highly localized behavior. Distance-weighting was also varied to either weigh all neighbors equally or weight them by distance (i.e. closer neighbors are more valuable). Lastly, distance calculation metrics were varied to consider any potential impact due to differing approaches to distance computation.

For logistic regression, we varied both the penalty method and regularization strength. Both parameters affect the model's resistance to overfitting. Certain penalty techniques and regularization strengths can also improve performance when applied correctly, so it is important to determine the correct regularization method for a given dataset. We also varied the solver used, which impacts training behavior and how optimal performance is identified.

Table 5: Parameter grids used for hyperparameter grid-search

Classifier	Parameter Grid
DecisionTree	splitter=[best, random], max_depth=[5, 10, 25, None], min_samples_split=[2, 5, 10, 50], min_samples_leaf=[1, 5, 10]
RandomForest	n_estimators=[50, 100, 500], criterion=[gini, entropy], max_depth=[5, 10, 25, None], min_samples_split=[2, 5, 10, 50], min_samples_leaf=[1, 5, 10]
LinearSVC	penalty=[l1, l2], C=[1, 4, 9, 16, 25], loss=[hinge, squared_hinge]
KNN	n_neighbors=[5, 10, 15, 20], weights=[uniform, distance], metric=[l1, l2, cosine]
LogisticRegression	penalty=[l1, l2], C=[1e-3, 1e-2, 1e-1, 1, 1e+1, 1e+2, 1e+3], solver=[newton-cg, lbfgs, liblinear]

3.6 Metrics

To analyze the performance of our models, we used a handful of different metrics:

1. **Accuracy, Macro F1, Weighted F1**
   These metrics provide simple numerical methods for evaluating and comparing classification model performance. Accuracy measures how frequently a model predicts correctly as a percentage of the overall provided data. F1 score measures model performance by comparing the model's precision (true positives divided by all positive predictions) and recall (true positives divided by all positive values in the dataset). F1 is slightly better than accuracy, as it also considers the impact of incorrect predictions and the types of errors observed. F1 scores are averaged by class using two strategies: Macro-averaging and weighted averaging. Macro F1 is calculated by summing all class-specific F1 scores and dividing that total by the number of classes. Weighted

F1 score takes into account the number of samples of each class, multiplying each class's F1 score by the class's proportion of the dataset. In our experiments we prioritized macro F1, as we are performing binary classification and have a slight class imbalance towards upward movement. By focusing on macro F1 we intentionally introduce a minor bias towards downward movement, making it harder for models to bias toward predicting upward movement as a result of its slight prevalence. That said, we still report weighted F1 for posterity.

2. **Receiver Operator Characteristic Curve**
The Receiver Operator Characteristic (ROC) curve represents the predictive capabilities of a model as a function of its true positive rate versus its false positive rate. This relationship is generated by varying the probability thresholds used to make a certain prediction. The area under this curve, or AUC, describes the ability of a model to separate the desired classes. I.e. a larger AUC indicates a better model, where an AUC of 1 indicates a perfect model. We generated ROC plots to comparing the different datasets when fitted on the same model, and compare the different models when fitted with the same dataset.

3.7 Baselines

To validate and contextualize the effectiveness of our models we compared their performance with a couple of benchmarks:

1. **Random Baseline**
As with any machine learning model, it is very important to compare performance to a random baseline to ensure that our model has actually identified any underlying behavior and is capable of "true" prediction (i.e. not predicting randomly; using real information). The random baseline is also particularly notable in the financial investment context, as it has been identified previously that random strategies are often comparable to more complexly defined strategies [32].
2. **Previous-Value Baseline**
The Previous-Value Baseline makes use of a naive repetition strategy. Specifically, the output of the baseline will be value equivalent to the previous value observed. Note that in our results we abbreviated the baseline's name to "Previous Baseline"
3. **Consensus Baseline**
Similar to the Previous-Value Baseline, the Consensus Baseline operates by setting the predicted value to the consensus of results from the previous n time-steps, where n is some user-chosen number. Specifically, in our implementation we use $n = 5$ days as this is also the period we used for our datasets. As a result, the baseline's prediction will be the consensus of the previous 5 days of data. Note that the Previous-Value Baseline is actually an instance of the Consensus Baseline where $n = 1$.
4. **Prior Baseline**
The Prior Baseline computes the prior distribution of the training data and predicts the class with the greatest number of overall occurrences in the training set. Note that this baseline can have an ROC curve generated using the probabilities of the computed prior distribution.

4 Results

4.1 Individual Dataset Comparison

The mean accuracy, macro F1, and weighted F1 were computed for all 66 datasets across all five tested models. For each dataset, the mean value for each metric was calculated in order to quantify the general performance of models with respect to the dataset. These results are visible in Table 6, sorted and ranked by mean macro F1.

Probably the most notable result is that S&P 500 datasets are all ranked 39th or below, fitting at or below the 41st percentile. As a result, this means that roughly 59% of our datasets saw better mean performance than all of the S&P 500 datasets. Further, raw S&P 500 data had the lowest mean F1 score out of all datasets, including both of the random datasets.

[8]The normal sample dataset is considered to be a percent-change dataset as its data is sampled from a distribution designed to be similar to the distribution of percent-changes of the S&P 500.

Table 6: Mean out-sample performance on the top 10 performing datasets by mean macro F1, S&P 500 datasets, and randomly generated baseline datasets. The letters in the Asset Types column stand for (F)orex, (B)ond, (I)ndex future, and (C)ommodities future.

Rank (n=66)	Asset Types	Transformation(s)	Accuracy	Macro F1	Weighted F1
1	I, C	%	0.571 ± 0.019	0.548 ± 0.015	0.562 ± 0.014
2	B, I, C	%	0.544 ± 0.041	0.528 ± 0.038	0.541 ± 0.039
3	B, I, C	% PCA	0.541 ± 0.023	0.522 ± 0.020	0.536 ± 0.020
4	F, I	Raw	0.541 ± 0.010	0.520 ± 0.013	0.532 ± 0.010
5	B, C	%	0.537 ± 0.017	0.512 ± 0.034	0.527 ± 0.028
6	I, C	% PCA	0.539 ± 0.032	0.511 ± 0.021	0.527 ± 0.023
7	B, I	%	0.535 ± 0.019	0.511 ± 0.015	0.524 ± 0.015
8	B	%	0.528 ± 0.023	0.510 ± 0.017	0.521 ± 0.018
9	F, I	%	0.536 ± 0.030	0.510 ± 0.026	0.522 ± 0.026
10	B, I	% PCA	0.537 ± 0.028	0.510 ± 0.015	0.523 ± 0.018
39	SPY Only	%	0.527 ± 0.021	0.480 ± 0.029	0.495 ± 0.019
46	SPY Only	% PCA	0.530 ± 0.022	0.474 ± 0.038	0.490 ± 0.026
50	SPY Only	Raw PCA	0.503 ± 0.030	0.472 ± 0.052	0.471 ± 0.067
61	Brownian Motion	Raw	0.508 ± 0.035	0.452 ± 0.075	0.456 ± 0.090
65	Normal Sample	%[8]	0.506 ± 0.045	0.435 ± 0.070	0.442 ± 0.074
66	SPY Only	Raw	0.516 ± 0.051	0.412 ± 0.049	0.426 ± 0.062

Looking at individual asset types, datasets containing index futures data saw notable success. Specifically, 8 out of the top 10 datasets (4 of which are in the top 5) by mean macro F1 contained index futures data. Commodities futures and bonds were almost as successful, with both being present in 6 of the top 10 datasets (the former of which being present in 4 of top 5). Lastly, Forex data was only present in 2 of the top 10, both times also being in the presence of index futures. Further analysis of the broader impact of individual assets is done in Section 4.2.1.

Looking at asset combinations, datasets containing both index and commodities futures, as well as datasets containing both bonds and index futures performed notably well. 4 out of the top 10 datasets contained both index and commodities futures, including the entire top 3. Similarly, 4 out of the top 10 datasets contained both bonds and index futures, 3 of which are in the top 5. Further analysis of the general impact of specific asset combinations can be found in Section 4.2.2.

Looking at the transforms used by the best performing datasets, it appears that percent-change is preferred for producing better datasets. This can be seen as 9 of the top 10 datasets use percent-change, as well as the top 2 S&P 500-exclusive datasets. Further, most of the top-performing datasets exclude the use of PCA, as 7 of the top 10 do not use it, including the only dataset in the top 10 that does not use percent-change.

Many datasets outperform the range of results seen from datasets containing S&P 500 data alone. One especially notable dataset is the set containing index futures and commodities futures converted to percent-change. This had the highest values for accuracy, macro F1, and weighted F1 out of all datasets analyzed. As a result, this dataset was the only dataset for which any metric's mean minus one standard deviation was higher than that metric's plus one standard deviation for all S&P 500 datasets. This occurred for the macro and weighted F1 metrics, as shown below:

$$\mu[\text{Macro F1}_{\text{IC}}] - \sigma[\text{Macro F1}_{\text{IC}}] = \mathbf{0.533} > \mathbf{0.527} = \max_{X \in \text{SPY}} (\mu[\text{Macro F1}_X] + \sigma[\text{Macro F1}_X])$$
$$\mu[\text{Weighted F1}_{\text{IC}}] - \sigma[\text{Weighted F1}_{\text{IC}}] = \mathbf{0.548} > \mathbf{0.546} = \max_{X \in \text{SPY}} (\mu[\text{Weighted F1}_X] + \sigma[\text{Weighted F1}_X])$$

It is also worth noting that only three datasets performed worse than the Brownian Motion dataset with respect to Macro F1. Specifically, these were the datasets containing raw Forex data (rank 62), raw bonds and commodities futures (rank 63), and raw forex, index futures, and commodities futures transformed using PCA (rank 64). All of these datasets underperformed compared to the Brownian Motion dataset across all metrics, with the exception of the lattermost dataset, which outperformed Brownian Motion on accuracy and weighted F1 metrics.

A table containing the full list of rankings for every dataset analyzed is available in Appendix E

4.2 Asset Types

As the focus of our research is to identify the impact, if any, of intermarket data on stock market prediction, we focus on the impact of individual assets and their combinations on model performance.

4.2.1 Individual Asset Impact

To compare the impact of each individual asset type the datasets were split based on the presence of specific asset types. Looking at the inclusion of specific assets, it appears that none of the assets improve performance significantly due to their inclusion. Specific results can be seen in Table 7. Based on this data, across all asset types we fail to see any substantial differences in performance following the inclusion of specific asset types.

Table 7: Mean model performance ($\pm\sigma$) with respect to the presence of specific asset types in datasets[9]

Asset Type	Presence	Accuracy	Macro F1	Weighted F1
Forex	Yes	0.522 ± 0.025	0.483 ± 0.035	0.498 ± 0.031
	No	0.529 ± 0.028	0.487 ± 0.038	0.502 ± 0.037
Bond	Yes	0.526 ± 0.025	0.491 ± 0.032	0.505 ± 0.030
	No	0.525 ± 0.028	0.480 ± 0.040	0.495 ± 0.038
Index Futures	Yes	$\mathbf{0.532 \pm 0.026}$	$\mathbf{0.492 \pm 0.035}$	$\mathbf{0.508 \pm 0.031}$
	No	0.519 ± 0.026	0.479 ± 0.036	0.492 ± 0.036
Commodities Futures	Yes	0.527 ± 0.028	0.487 ± 0.036	0.503 ± 0.033
	No	0.524 ± 0.026	0.484 ± 0.037	0.497 ± 0.036
Stock-only		0.519 ± 0.032	0.459 ± 0.048	0.471 ± 0.052

As a baseline, datasets containing only S&P 500 data did perform slightly worse on average across almost all metrics, with a mean accuracy of 51.9% ($\sigma = 3.2\%$), macro F1 of 0.459% ($\sigma = 0.048$), and weighted F1 of 0.471 ($\sigma = 0.052$). Contrasting this with the minimum values for each metric when considering the inclusion of other assets, which are 51.9% ($\sigma = 2.6\%$), 0.479 ($\sigma = 0.036$), and 0.492 ($\sigma = 0.036$) for accuracy, macro F1, and weighted F1 respectively. That said, this difference is not large enough to be significant as the range of observed values (as approximated by standard deviation) overlap for all metrics.

An important feature of our results is that both the best and worst results controlling for individual asset types occurred with respect to index futures. Specifically, the the best mean performance observed with respect to asset type inclusion occurred in the presence of index futures, receiving scores of 53.2% ($\sigma = 0.026$), 0.492 ($\sigma = 0.035$), and 0.508 ($\sigma = 0.031$) for accuracy, macro F1, and weighted F1 respectively. By contrast, the worst performance observed with respect to asset type inclusion occurred in the absence of index futures, receiving scores of 51.9% ($\sigma = 0.026$), 0.479 ($\sigma = 0.036$), and 0.492 ($\sigma = 0.036$) for accuracy, macro F1, and weighted F1 respectively. This seems to suggest empirically that the inclusion or exclusion of index futures specifically may have a some impact. However, we lack statistically significant evidence to suggest that this is the case, as the range of observed results still overlap when considering standard deviation.

Another interesting consideration to make is that mean model performance appears to consistently degrade across all metrics in the presence of Forex data. While not a statistically significant difference, this seems to imply that the inclusion of Forex data may be actively detrimental. A possible explanation for this is that Forex data may not be sufficiently correlated with the S&P 500 to extract any useful information for prediction. As a result, certain techniques may be overfitting on the Forex data, resulting in performance degradation. This contrasts with bonds, index futures, and commodities futures, which all saw improvements in mean performance across all metrics due to their inclusion. One potential explanation for this difference is that bonds and futures share the characteristic of having a defined time to maturity. Because these assets are reliant on a specified time to maturity, their prices fluctuates with respect to an expected future value or yield. Forex data lacks this characteristic, simply corresponding to the exchange rate at the exact moment of sampling.

[9] Note that rows are not exclusive across asset types. I.e. the set of datasets containing one specific asset type also includes datasets which may or may not have any number (including none) of the *other* asset types. This also means that rows representing the absence of a particular asset contain the SPY-only baseline datasets.

4.2.2 Asset Combinations

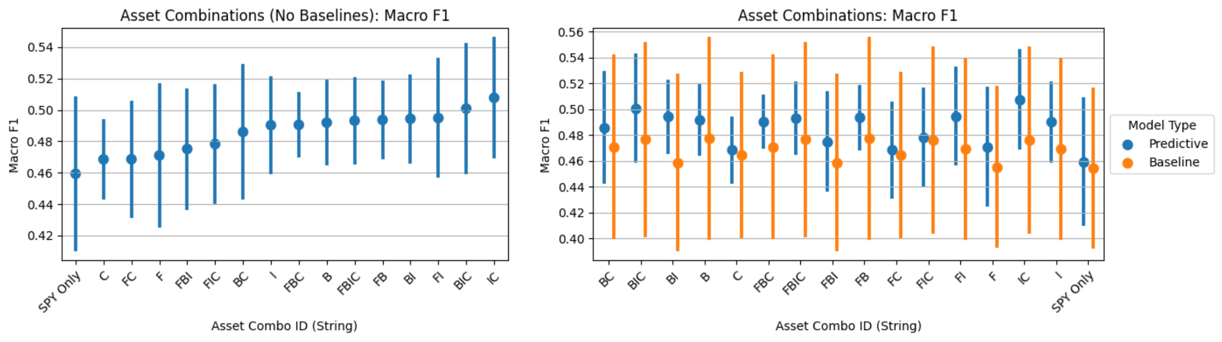
Results with respect to specific combinations of asset types can be viewed to attempt to estimate the impact of specific combinations of assets. Results separated by these combinations can be seen in Table 8. Based on these results, we lack statistically significant evidence to suggest that the mean performance on any of the asset combinations differs, however some interesting conclusions may be inferred for specific combinations.

Table 8: Mean performance of models ($\pm\sigma$) on asset type combinations. Asset type codes are (F)orex, (B)ond, (I)ndex future, and (C)ommodities future.

Asset Types	Accuracy	Macro F1	Weighted F1
SPY Only	0.519 ± 0.032	0.459 ± 0.048	0.471 ± 0.052
F	0.516 ± 0.025	0.471 ± 0.045	0.482 ± 0.047
B	0.521 ± 0.024	0.492 ± 0.026	0.503 ± 0.027
I	0.529 ± 0.022	0.490 ± 0.030	0.505 ± 0.025
C	0.514 ± 0.027	0.468 ± 0.025	0.486 ± 0.021
FB	0.517 ± 0.023	0.494 ± 0.024	0.504 ± 0.022
FI	0.530 ± 0.028	0.495 ± 0.037	0.508 ± 0.033
FC	0.524 ± 0.024	0.469 ± 0.036	0.489 ± 0.028
BI	0.536 ± 0.023	0.494 ± 0.027	0.510 ± 0.023
BC	0.528 ± 0.027	0.486 ± 0.042	0.500 ± 0.046
IC	$\mathbf{0.544 \pm 0.031}$	$\mathbf{0.508 \pm 0.038}$	$\mathbf{0.523 \pm 0.034}$
FBI	0.527 ± 0.027	0.475 ± 0.038	0.492 ± 0.029
FBC	0.512 ± 0.023	0.490 ± 0.020	0.501 ± 0.017
FIC	0.525 ± 0.028	0.478 ± 0.037	0.496 ± 0.031
BIC	0.540 ± 0.026	0.501 ± 0.041	0.517 ± 0.036
FBIC	0.529 ± 0.022	0.473 ± 0.027	0.511 ± 0.024

The asset type combination which saw the best mean performance across all metrics was the combination index futures and commodities futures. These datasets saw a mean accuracy of 54.4% ($\sigma = 3.1\%$), macro F1 of 0.508 ($\sigma = 0.038$), and weighted F1 of 0.532 ($\sigma = 0.034$). This seems to suggest the possibility of an interaction between index futures and commodities futures, resulting in improved model performance. Further, this also aligns with our results looking at assets individually, as both index futures and commodities futures saw improvements on average when included in datasets. However, we lack sufficient evidence to suggest statistical significance. 17 contains interval plots for each dataset, comparing macro F1 scores. The left side of the figure shows both that S&P 500 datasets perform slightly worse than all other datasets on average, and that datasets with index futures and commodities futures performs the best out of all other datasets by a small margin. It is worth recognizing that all intervals overlap, again suggesting a lack of evidence to confirm statistical significance. The right side of the figure compares datasets between our predictive models and our baselines. In all cases, our predictive models average macro F1 scores outperform our baselines' F1 scores. There is again a large overlap of intervals preventing significant conclusions from being drawn between predicitve models and baselines at a per-dataset basis.

Figure 17: Interval plots displaying mean out-sample macro F1 scores with respect to asset combinations.
Left: Ordered by mean macro F1, excluding baselines. Right: Ordered lexicographically, including baselines.



It is also worth noting that datasets containing only S&P 500 data had the lowest mean macro and weighted F1 scores observed, receiving values of 0.459 ($\sigma = 0.048$) and 0.471 ($\sigma = 0.052$) respectively. This suggests that the inclusion of any intermarket data on top of S&P 500 data may result in model improvement, though insignificant. That said, datasets containing only S&P 500 data did not have have the worst mean accuracy. Instead, the worst mean accuracy was observed for the datasets containing Forex, bond, and commodities futures data, receiving a mean accuracy of 51.2% ($\sigma = 2.3\%$), compared to 51.9% ($\sigma = 3.2\%$) for S&P 500 alone. Since this difference only occurs with respect to accuracy, we cannot suggest that the combination of Forex, bond, and commodities futures data may result in adverse effects on model performance. However, this does somewhat fall in line with previous inferences that the inclusion of Forex data specifically may be detrimental.

4.3 Transforms

For each model, the mean values for accuracy, macro F1, and weighted F1 scores across all datasets were computed with respect to the use of different transforms. Specifically, we want to consider the impact, if any, of using the percent-change and PCA transforms on predicting the S&P 500.

4.3.1 Percent-Change

To compare the impact of the percent-change transform, the datasets were split based on the presence of the transformation. Specific results can be seen in Table 9. Based on this data, across all asset types we fail to see any substantial differences in performance due to the use of percent-change, however minor differences are present.

Table 9: Mean out-sample performance of models ($\pm\sigma$) summarized by usage of the percent-change transform on data.

Percent	Model	Accuracy	Macro F1	Weighted F1
Yes	Decision Tree	0.509 ± 0.024	0.498 ± 0.024	0.506 ± 0.023
	KNN	0.521 ± 0.024	0.499 ± 0.028	0.512 ± 0.026
	Logistic Regression	0.534 ± 0.018	**0.503 ± 0.027**	**0.518 ± 0.024**
	Random Forest	0.537 ± 0.027	0.490 ± 0.029	0.511 ± 0.028
	Linear SVM	0.534 ± 0.020	0.501 ± 0.031	0.517 ± 0.027
	TOTAL (Mean)	0.527 ± 0.025	0.498 ± 0.028	0.513 ± 0.026
No	Decision Tree	0.503 ± 0.027	0.477 ± 0.038	0.482 ± 0.045
	KNN	0.523 ± 0.028	0.466 ± 0.042	0.483 ± 0.039
	Logistic Regression	0.536 ± 0.020	0.467 ± 0.043	0.489 ± 0.034
	Random Forest	0.520 ± 0.031	0.476 ± 0.038	0.485 ± 0.037
	Linear SVM	**0.538 ± 0.023**	0.475 ± 0.036	0.495 ± 0.029
	TOTAL (Mean)	0.524 ± 0.029	0.472 ± 0.039	0.487 ± 0.037

On average, every model appears to outperform on almost every metric when using percent-change transformed data compared to raw data. Specifically, every model outperforms in the presence of percent-change with respect to macro F1 and weighted F1, while KNN, Logistic Regression, and Linear SVMs slightly underperform with respect to accuracy. This suggests that predicting on percent-change data may generally be more effective than predicting on raw data. This falls in line with previous assumptions, as many of the analyzed models perform best when input data falls within a generally constant pre-determined distribution (i.e. stationary), as we believe is the case with percent-change, as opposed to the raw data, for which the distribution of input values changes over time (i.e. non-stationary). That said, we unfortunately do not have a satisfactory explanation as to why certain models underperform with respect to accuracy but not other metrics when using percent-change.

4.3.2 Principal Component Analysis

Similar to percent-change discussed previously, the datasets were split based on the presence of the PCA transform. Specific results can be seen in Table 10.

On average, every model appears to underperform when using the PCA transform compared to when PCA is absent. Across all models, the only a handful of models outperformed when using PCA. Specifically, only Decision Trees and Random Forest models outperformed by any metric, seeing slight improvements with respect to accuracy and weighted F1. That said, the difference between the presence and absence of PCA is very small. This aligns with traditional intuition about PCA, as the goal of PCA is to transform the data while maintaining as much variability as possible from the underlying data. As a result, we should not expect a substantial difference between the use of PCA or not. Further,

Table 10: Mean out-sample performance of models ($\pm\sigma$) summarized by usage of the PCA transform on data.

PCA	Model	Accuracy	Macro F1	Weighted F1
Yes	Decision Tree	0.508 ± 0.023	0.486 ± 0.024	0.497 ± 0.020
	KNN	0.521 ± 0.026	0.473 ± 0.042	0.489 ± 0.040
	Logistic Regression	0.534 ± 0.019	0.483 ± 0.035	0.502 ± 0.028
	Random Forest	0.531 ± 0.027	0.481 ± 0.034	0.499 ± 0.030
	Linear SVM	0.535 ± 0.020	0.481 ± 0.037	0.501 ± 0.030
	TOTAL (Mean)	0.526 ± 0.025	0.481 ± 0.035	0.497 ± 0.030
No	Decision Tree	0.504 ± 0.028	0.489 ± 0.041	0.492 ± 0.050
	KNN	0.524 ± 0.026	0.492 ± 0.033	0.507 ± 0.029
	Logistic Regression	0.536 ± 0.020	0.487 ± 0.045	0.505 ± 0.038
	Random Forest	0.527 ± 0.033	0.485 ± 0.035	0.497 ± 0.040
	Linear SVM	$\mathbf{0.537 \pm 0.023}$	$\mathbf{0.495 \pm 0.035}$	$\mathbf{0.511 \pm 0.029}$
	TOTAL (Mean)	0.526 ± 0.029	0.490 ± 0.038	0.502 ± 0.038

it may be the case that the minor observed performance degredation may be the result of the lossy nature of PCA. Specifically, in our implementation, principal components were only generated until a minimum of 95% of variability was explained. This means that in the worst case upwards of 5% of variability is unexplained after performing PCA, which may account for the observed degradation in prediction performance.

Across all models, the largest difference seen as a result of PCA happened with KNN models. KNN received a mean macro F1 of 0.492 ($\sigma = 0.033$) and mean weighted F1 of 0.507 ($\sigma = 0.029$) when not using PCA, while receiving a mean macro F1 of 0.473 ($\sigma = 0.042$) and mean weighted F1 of 0.489 ($\sigma = 0.040$) when using PCA. This resulted in a performance degradation of 0.019 and 0.018 for macro F1 and weighted F1 respectively. While still falling within the standard deviations of each metric across both the presence and absence of PCA, these differences are notably larger than those seen by other models. A possible explanation for this is that KNN may be more prone to being negatively impacted a result of lost variance after performing PCA. Alternatively, this may be the case because PCA is a class-agnostic transformation and may reduce separability of classes in certain cases (this specific reason is a primary justification for the use of Linear Discriminant Analysis as opposed to PCA for many classification tasks).

4.4 Models

For each model and each baseline, the mean values for accuracy, macro F1, and weighted F1 scores across all datasets were computed. These results can be used to identify how effectively our models trained on our datasets on average and if they produce meaningful predictions.

4.4.1 Out-sample Performance

To consider the quality of our models, we summarized the performance of each model across the out-sample (test) splits of all datasets containing S&P 500 data (i.e. excluding brownian motion and normal sample datasets). These results can be seen in Table 11.

Table 11: Mean out-sample model performance ($\pm\sigma$), compared to baseline models

Model	Accuracy	Macro F1	Weighted F1
Decision Tree	0.506 ± 0.026	0.487 ± 0.033	0.494 ± 0.038
KNN	0.522 ± 0.026	0.482 ± 0.039	0.498 ± 0.036
Logistic Regression	0.535 ± 0.019	0.485 ± 0.040	0.503 ± 0.033
Random Forest	0.529 ± 0.030	0.483 ± 0.034	0.498 ± 0.035
Linear SVM	0.536 ± 0.021	0.488 ± 0.036	0.506 ± 0.030
TOTAL (Mean)	0.526 ± 0.027	0.485 ± 0.036	0.500 ± 0.034
Consensus Baseline	0.495 ± 0.011	0.474 ± 0.011	0.502 ± 0.012
Previous Baseline	0.536 ± 0.020	$\mathbf{0.528 \pm 0.020}$	$\mathbf{0.536 \pm 0.020}$
Prior Baseline	$\mathbf{0.564 \pm 0.007}$	0.360 ± 0.003	0.406 ± 0.008
Random Baseline	0.516 ± 0.024	0.513 ± 0.024	0.518 ± 0.024

When comparing the mean performance of our models over all datasets to the performance of our baselines, we see that all of our models are worse than at least one baseline for each metric. Specifically, the highest mean accuracy was observed using the Prior Baseline with 56.4% ($\sigma = 0.7\%$), while the highest mean macro and weighted F1 scores were observed from the Previous Baseline with 0.528 ($\sigma = 0.020$) and 0.536 ($\sigma = 0.020$), respectively. This being the case, our results suggest that we lack sufficient evidence to claim that our models were capable of making meaningful predictions on average across all datasets. That said, because these results average performance across all datasets, it is possible that poor performance on some datasets is counteracting good performance on other datasets. Justifying this possibility, as described in Section 4.2, we saw that datasets containing index futures and commodities futures outperformed other datasets on all metrics, while the presence of Forex data is seemingly detrimental to model performance across all metrics. This means that summarizing our results by model across all datasets may not provide a meaningful comparison with baselines.

Comparing models with each other (excluding baselines), we saw that Linear SVMs performed best on average across all metrics, having mean accuracy, macro F1, and weighted F1 scores of 53.6% ($\sigma = 2.7\%$), 0.485 ($\sigma = 0.036$), and 0.506 ($\sigma = 0.034$) respectively. This aligns with existing literature, which has seen notable success with utilizing SVMs [13]. Further, Linear SVMs perform similarly or better than many of the baselines on multiple metrics. Specifically, Linear SVMs outperformed the Consensus Baseline on all metrics, the Prior Baseline on macro F1 and weighted F1, and the Random Baseline on accuracy. The only baseline which Linear SVMs did not outperform in any metric was the Previous baseline, which performed similarly with respect to mean accuracy. It is also worth mentioning that Logistic Regression models appeared to perform very similarly, though slightly worse than Linear SVMs.

4.4.2 In-sample Performance

To consider the effectiveness training of our models, we summarized the performance of each model across the in-sample (training) splits of all datasets containing S&P 500 data. These results can be seen in Table 12.

Table 12: Mean in-sample model performance ($\pm\sigma$), compared to baseline models

Model	Accuracy	Macro F1	Weighted F1
Decision Tree	0.802 ± 0.115	0.796 ± 0.122	0.799 ± 0.119
KNN	0.765 ± 0.163	0.759 ± 0.166	0.763 ± 0.164
Logistic Regression	0.604 ± 0.048	0.563 ± 0.076	0.575 ± 0.071
Random Forest	$\mathbf{0.957 \pm 0.089}$	$\mathbf{0.952 \pm 0.102}$	$\mathbf{0.954 \pm 0.098}$
Linear SVM	0.629 ± 0.061	0.597 ± 0.087	0.607 ± 0.082
TOTAL (Mean)	0.751 ± 0.164	0.733 ± 0.182	0.740 ± 0.177
Consensus Baseline	0.506 ± 0.006	0.496 ± 0.006	0.509 ± 0.006
Previous Baseline	0.479 ± 0.009	0.473 ± 0.010	0.479 ± 0.009
Prior Baseline	0.551 ± 0.007	0.355 ± 0.003	0.391 ± 0.008
Random Baseline	0.506 ± 0.010	0.504 ± 0.010	0.507 ± 0.010

In contrast to the out-sample performance described previously, the in-sample performance of every model outperforms all baselines across all metrics. This suggests that it is very likely substantial overfitting may have occurred for many models. Random Forest in particular saw the highest in-sample performance, predicting almost perfectly on in-sample data. Random Forest received mean accuracy, macro F1, and weighted F1 scores of 95.7% ($\sigma = 8.9\%$), 0.952 ($\sigma = 0.102$), and 0.954 ($\sigma = 0.098$) respectively. The most likely explanation for this is overfitting due to the extreme complexity of Random Forest, being an ensemble technique utilizing Decision Trees, which themselves are prone to overfitting due to excessive complexity. By contrast, the models which performed worst with respect to in-sample performance were actually our best performing models with respect to out-sample performance. Specifically, Linear SVMs and Logistic Regression models received mean accuracy, macro F1, and weighted F1 scores of 62.9% ($\sigma = 6.1\%$) and 60.4% ($\sigma = 4.8\%$); 0.594 ($\sigma = 0.087$) and 0.563 ($\sigma = 0.076$); and 0.607 ($\sigma = 0.082$) and 0.575 ($\sigma = 0.071$) respectively. This aligns with our overfitting hypothesis, as SVMs and Logistic Regression are generally regarded as being highly resistant to overfitting [16].

5 Conclusion

We found a large collection of specific combinations of asset types and transformations which resulted in substantial performance improvements compared to our S&P 500 baseline datasets. Datasets such as percent-change transformed index futures and commodities futures performed exceptionally well. Because this improvement was found using intermarket data, it may suggest a possible disagreement with the Efficient Market Hypothesis. Specifically, it implies that these datasets contain information relevant to S&P 500 value which do not factor in until the following trading day. However, we currently lack the proper statistical evidence to justify why this improvement was observed. We hypothesize that this outcome occurred because the prices of futures (and other derivatives) are reliant on a predicted future value associated with the contract's specified time to maturity. It may be possible that this allows us to meaningfully factor the derivatives market's predictions into stock market predictions.

No specific asset type was found to impact prediction performance significantly. This agrees with the Efficient Market Hypothesis, as we were unable to identify on the per-asset basis that any particular asset type's data contained important information not already considered in the value of the S&P 500. That said, it is worth mentioning that the inclusion of most asset types induced minor empirical improvements—though statistically insignificant. However, we lack sufficient evidence to suggest that performance in the presence or absence of any particular asset type differs from another significantly.

None of the data transformations explored significantly impacted average model performance. We did notice that the percent-change transformation tended to increase mean performance by a small margin, however we lack sufficient evidence to prove a statistically significant difference. We also noticed that PCA may induce a slight decrease in mean model performance in most cases. However, the reduction in dimensionality that PCA brings may outweigh this downside for certain models. In this context, a tradeoff is present between the use of PCA, the number of principal components generated, and model performance.

Across all of the models explored, we failed to identify any statistically significant differences in their average performances, both between each other and compared to our baselines. This being the case, no concrete conclusions can be drawn about any specific model performing better on average. However, when considering the poor mean performance observed in the models compared to the performance observed on many individual datasets, we cannot conclude that these models are inherently ineffective. This being the case, further analysis should be conducted which is robust to wide ranges or skews in the distributions of performance across datasets.

6 Future Work

There are multiple aspects of this work that could be expanded upon. This being the case, we would like to propose a number of potential future directions for additional research based on the work described in this paper.

6.1 Deep Learning

Our present work focuses on a variety of machine learning models. These are generally helpful and performed well, but there is room for improvement. As mentioned previously, deep learning models are well suited to tasks requiring vast amounts of data, and there are models specialized to deal with time series data. Further work should investigate the use of deep learning models such as RNNs and LSTMs with respect to the use of external market data in predicting the movement of the S&P 500. RNNs make up roughly 51.0% of financial time series forecasting papers published between 2005 and 2019. Of those, roughly 60.4% used LSTMs. These models have the potential to significantly improve prediction accuracy.

6.2 Regression

The work in this paper focuses on classification, only predicting if the value of the S&P 500 will go up or down. A further development would be to instead apply regression, predicting exact future values of the S&P 500. This would allow for a greater possibility to implement risk-awareness. It also may be the case that the use of intermarket data is more beneficial toward regression than classification.

6.3 Trading Strategy

The work discussed in this paper lays the groundwork for developing a trading strategy to be carried out by the model. This would entail using the predicted movement of SPY to carry out trades. At a base level, selling when the price is predicted to decrease, buying/holding when the price is predicted to increase. There are further nuances that could be incorporated into the predictive model to advance the trading strategy, such as regression. Predicting price points would further develop trading decisions, as investigation into the magnitude of movement could be taken into consideration when buying, selling, or holding.

6.4 Transformations

In this paper the only transformations explored were percent-change and PCA. However, many other potentially useful transformations also exist. Specifically, Linear Discriminant Analysis may prove beneficial towards the task of classification. Further, Fourier decomposition may help to identify underlying cyclic behavior, which may be useful towards market forecasting.

6.5 Window Size Experimentation

As stated in Section 3.3, we focused on a window size of 5 days. It would be valuable to evaluate the impact of varying the size of input window used, as providing a greater history of market behavior may induce performance improvements compared to shorter window sizes. Additionally, non-linear windows may be considered such that a large period of history such that the relative amount of samples in the window is proportional to its age (i.e. data is provided in smaller quantities the older it is).

6.6 Alternative Polling and Forecasting Rates

In our work we utilized data polled on a daily basis. However, the impact of intermarket data on predicting market movement may vary depending on the polling basis. Specifically, it may be the case that the value of intermarket data on forecasting capabilities is different between intraday, daily, weekly, monthly, or quarterly bases. This is especially valuable as data pertaining to certain assets—notably bond yields [33, 34]—are frequently used as indicators of future long-term economic changes. Additionally, consideration should be given to the relative value of intermarket asset data on predicting varying distances into the future. It is possible that intermarket data is more valuable towards predicting multiple days into the future than it is towards next-day predictions. Future work could consider how the predictive capabilities change with respect to the distance of the prediction from the input data.

6.7 Alternative Features or Data Sources

As stated in Section 3.1, our data only consisted of a handful of features which were shared for almost every asset. A potential downside of this is that we may have omitted features which could have proven beneficial towards our models. As an example, in our experimentation we excluded yield data from our bonds, which may have had a substantial impact had it been included [33, 34]. Further, the availability of our data was heavily limited by the API we chose to source our data from. This being the case, it may be beneficial to consider using alternative data sources which may have higher quality data and more features/metrics.

6.8 Hole Interpolation

As stated in Section 3.3, we opted to remove any and all days from a dataset where values were missing for any asset in the dataset. This has two notable downsides: Firstly, this technique causes us to lose potentially valuable data from the assets that do have values. Secondly, removing days of data outright makes the sampling basis of our models inconsistent, sometimes needing to predict multiple days in advance or use data that is older than the desired window size. This being the case, it may be a good idea to consider interpolating in the presence of holes so as to allow for the use of otherwise available data and keep the sampling basis consistent. This could be done in a number of ways, such as forward-filling holes with the last valid value, backward-filling holes with the next valid value, performing a linear interpolation between the last and next valid values, or setting the missing to a specified value (e.g. 0).

7 Acknowledgements

We thank Stephan Sturm, Marcel Blais, and Xiaozhong Liu for advising this project. We also thank Branka Hadji-Misheva and Jörg Osterrieder for sponsoring this project and being great mentors and points of assistance. We thank Connor West for his invaluable assistance in debugging. Results in this paper were obtained in part using a high-performance computing system at WPI acquired through NSF MRI grant DMS-1337943.

References

[1] Eugene F. Fama. The Behavior of Stock-Market Prices. *The Journal of Business*, 38(1):34–105, 1965.

[2] Eugene F. Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 25(2):383–417, 1970.

[3] Jaydip Sen, Rajdeep Sen, and Abhishek Dutta. *Introductory Chapter: Machine Learning in Finance-Emerging Trends and Challenges*. IntechOpen, December 2021.

[4] John J. Murphy. *Intermarket Technical Analysis: Trading Strategies for the Global Stock, Bond, Commodity, and Currency Markets*. John Wiley & Sons, September 1991.

[5] Monetary and Economic Department, Bank for International Settlements. Triennial Central Bank Survey - Foreign exchange turnover in April 2019, September 2019.

[6] OECD. *OECD Sovereign Borrowing Outlook 2019*. OECD Sovereign Borrowing Outlook. OECD, April 2019.

[7] Ngai Hang Chan and Jessie Chan. *Time Series: Applications to Finance*. John Wiley & Sons, Incorporated, Hoboken, 1st ed edition, 2004.

[8] Clive W. J. Granger and Oskar Morgenstern. SPECTRAL ANALYSIS OF NEW YORK STOCK MARKET PRICES. *Kyklos*, 16(1):1–27, February 1963.

[9] T Mitchell, B Buchanan, G DeJong, T Dietterich, P Rosenbloom, and A Waibel. Machine Learning. *Annual Review of Computer Science*, 4(1):417–433, 1990.

[10] Nikola Milosevic. Equity forecast: Predicting long term stock price movement using machine learning, November 2018.

[11] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522, October 2005.

[12] Dominik Wolff and Dr Ulrich Neugebauer. Tree-Based Machine Learning Approaches for Equity Market Predictions, June 2018.

[13] Carson Kai-Sang Leung, Richard Kyle MacKinnon, and Yang Wang. A machine learning approach for stock price prediction. In *Proceedings of the 18th International Database Engineering & Applications Symposium on - IDEAS '14*, pages 274–277, Porto, Portugal, 2014. ACM Press.

[14] Marc Claesen and Bart De Moor. Hyperparameter Search in Machine Learning, April 2015.

[15] Nello Cristianini. An introduction to support vector machines : And other kernel-based learning methods. In *An Introduction to Support Vector Machines : And Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, 2000.

[16] Asoke Nandi and Hosameldin Ahmed. *Condition Monitoring with Vibration Signals: Compressive Sampling and Learning Algorithms for Rotating Machine*. Wiley, first edition, December 2019.

[17] Robert E. Schapire. Explaining AdaBoost. In Bernhard Schölkopf, Zhiyuan Luo, and Vladimir Vovk, editors, *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, pages 37–52. Springer, Berlin, Heidelberg, 2013.

[18] Oliver Kramer. K-Nearest Neighbors. In Oliver Kramer, editor, *Dimensionality Reduction with Unsupervised Nearest Neighbors*, Intelligent Systems Reference Library, pages 13–23. Springer, Berlin, Heidelberg, 2013.

[19] Yves Lechevallier and Gilbert Saporta. *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. SpringerLink: Springer e-Books. Springer-Verlag Berlin Heidelberg, Heidelberg, online-ausg edition, 2010.

[20] Larry Hardesty. Explained: Neural networks. *MIT News*, April 2017.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[22] J. B. Heaton, N. G. Polson, and J. H. Witte. Deep Learning in Finance, January 2018.

[23] Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. AdaRNN: Adaptive Learning and Forecasting of Time Series, August 2021.

[24] Mehtabhorn Obthong, Nongnuch Tantisantiwong, Watthanasak Jeamwatthanachai, and Gary Wills. A Survey on Machine Learning for Stock Price Prediction: Algorithms and Techniques:. In *Proceedings of the 2nd International Conference on Finance, Economics, Management and IT Business*, pages 63–71, Prague, Czech Republic, 2020. SCITEPRESS - Science and Technology Publications.

[25] Soo-Han Kang and Ji-Hyeong Han. New RNN Activation Technique for Deeper Networks: LSTCM Cells. *IEEE Access*, 8:214625–214632, 2020.

[26] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306, March 2020.

[27] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, May 2020.

[28] Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[29] Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[30] M. B. Wilk and R. Gnanadesikan. Probability Plotting Methods for the Analysis of Data. *Biometrika*, 55(1):1–17, 1968.

[31] RALPH D'AGOSTINO and E. S. PEARSON. Tests for departure from normality. Empirical results for the distributions of b2 and $\sqrt{}$b1. *Biometrika*, 60(3):613–622, December 1973.

[32] Alessio Emanuele Biondo, Alessandro Pluchino, Andrea Rapisarda, and Dirk Helbing. Are Random Trading Strategies More Successful than Technical Ones? *PLOS ONE*, 8(7):e68344, July 2013.

[33] Andrew Ang, Monika Piazzesi, and Min Wei. What does the yield curve tell us about GDP growth? *Journal of Econometrics*, 131(1):359–403, March 2006.

[34] Jonathan H. Wright. The Yield Curve and Predicting Recessions, March 2006.

A Full Set of Analyzed Data

Asset Type	Asset	Data Start (Y-M-D)
Stock	SPY	2000-01-03
Government Bond	US 3-Year	2012-07-09
	US 5-Year	2000-01-03
	US 10-Year	2000-01-03
	US 30-Year[10]	2000-01-03
	France 3-Year	2012-07-08
	France 5-Year	2012-07-08
	France 10-Year	2012-07-09
	Italy 3-Year	2012-07-08
	Italy 5-Year	2012-07-08
	Italy 10-Year	2010-01-04
	Japan 3-Year	2012-07-09
	Japan 5-Year	2012-07-09
	Japan 10-Year	2012-07-09
	United Kingdom 3-Year	2012-07-09
	United Kingdom 5-Year	2012-07-09
	United Kingdom 10-Year	2012-09-24
Forex	USD/AUD	2000-01-04
	USD/CAD	2000-01-03
	USD/CHF	2000-01-03
	USD/EUR	2000-01-03
	USD/GBP	2000-01-04
	USD/JPY	2000-01-03
	EUR/AUD	2000-01-03
	EUR/CAD	2000-01-03
	EUR/CHF	2000-01-03
	EUR/GBP	2000-01-03
	EUR/JPY	2000-01-03
	JPY/AUD	2000-01-03
	JPY/CAD	2000-01-03
	JPY/CHF	2000-01-03
	JPY/GBP	2000-01-03
	GBP/AUD	2003-06-03
	GBP/CAD	2000-01-03
	GBP/CHF	2000-01-03
	AUD/CAD	2000-01-03
	AUD/CHF	2000-01-03
	CAD/CHF	2000-01-03
Index Futures	E-Mini SP 500 (ES)	2000-09-18
	E-Mini Russell 2000 (RTY)	2007-08-20
	EURO STOXX 50 (FESX)	2000-01-03
	Hang Seng Index (HSI)	2009-11-19
	Nikkei 225 (NK)	2007-10-12
	CBOE Volatility Index (VIX)	2009-04-24
Commodities Futures	Crude Oil (CL)	2000-08-23
	Gold (GC)	2000-08-30
	Heating Oil (HO)	2005-10-28
	Natural Gas (NG)	2009-04-24
	Corn (ZC)	2000-07-17
	Soy (ZS)	2000-01-03

[10] United States 30-Year bond data are only available as futures from the EOD Historical Data API.

B D'Agostino-Pearson Normality Test P-values

B.1 Raw Data

Asset Type	Asset	open	high	low	close	adjusted_close	volume
Stock	SPY	3.314e-201	6.613e-167	2.173e-203	1.037e-212	3.096e-213	0
Bond	US 3-Year	6.069e-159	8.895e-125	2.030e-146	7.112e-168	7.112e-168	_
	US 5-Year	1.198e-143	1.192e-138	5.732e-165	4.423e-146	4.423e-146	_
	US 10-Year	2.713e-82	1.764e-88	2.463e-108	3.700e-84	3.700e-84	_
	US 30-Year	0	0	0	1.976e-297	1.976e-297	0
	France 3-Year	0	0	0	0	0	_
	France 5-Year	0	0	0	0	0	_
	France 10-Year	0	0	0	0	0	_
	Italy 3-Year	0	0	0	0	0	_
	Italy 5-Year	1.889e-244	9.837e-238	0	6.598e-187	6.598e-187	_
	Italy 10-Year	1.274e-144	2.943e-198	6.200e-103	2.028e-113	2.028e-113	_
	Japan 3-Year	1.620e-293	3.096e-305	1.687e-186	0	0	_
	Japan 5-Year	0	0	0	0	0	_
	Japan 10-Year	0	0	0	0	0	_
	UK 3-Year	1.654e-215	1.405e-231	0	8.727e-278	8.727e-278	_
	UK 5-Year	3.663e-112	1.432e-160	2.246e-208	2.851e-152	2.851e-152	_
	UK 10-Year	1.256e-59	2.650e-123	3.635e-69	3.280e-72	3.280e-72	_
Forex	USD/AUD	0	0	5.582e-224	1.304e-274	1.304e-274	_
	USD/CAD	1.098e-43	1.626e-77	7.584e-54	5.981e-38	5.981e-38	0
	USD/CHF	0	0	0	0	0	0
	USD/EUR	9.133e-21	3.679e-21	7.074e-17	6.451e-18	6.451e-18	1.586e-122
	USD/GBP	1.086e-37	2.438e-43	3.108e-46	2.694e-56	2.694e-56	0
	USD/JPY	6.068e-121	1.008e-134	5.488e-239	1.042e-119	1.042e-119	0
	EUR/AUD	8.820e-77	1.088e-141	2.329e-68	1.532e-76	1.532e-76	0
	EUR/CAD	4.396e-71	8.350e-70	4.383e-50	5.580e-70	5.580e-70	0
	EUR/CHF	0	0	0	0	0	0
	EUR/GBP	1.853e-190	0	0	1.023e-172	1.023e-172	_
	EUR/JPY	1.145e-116	2.618e-109	1.569e-273	2.763e-113	2.763e-113	0
	JPY/AUD	4.595e-65	1.005e-201	1.470e-63	3.705e-68	3.705e-68	0
	JPY/CAD	1.967e-03	5.209e-03	***4.083e-02***	***9.323e-02***	***9.323e-02***	6.929e-253
	JPY/CHF	5.603e-53	5.322e-48	5.852e-149	5.419e-48	5.419e-48	_
	JPY/GBP	2.670e-221	2.689e-221	2.669e-221	2.678e-221	2.678e-221	_
	GBP/AUD	8.554e-112	5.354e-75	2.037e-201	8.870e-112	8.870e-112	0
	GBP/CAD	5.976e-234	3.387e-132	4.797e-275	2.825e-198	2.825e-198	0
	GBP/CHF	0	0	0	0	0	_
	AUD/CAD	4.035e-24	1.005e-27	1.909e-68	2.330e-20	2.330e-20	0
	AUD/CHF	0	0	0	0	0	_
	CAD/CHF	0	0	0	0	0	_
Index Futures	ES	2.671e-187	8.351e-162	6.842e-194	6.280e-214	6.280e-214	0
	RTY	8.042e-102	5.017e-89	3.261e-120	6.366e-107	6.366e-107	0
	FESX	2.920e-141	8.784e-146	1.586e-151	1.870e-135	1.870e-135	0
	HSI	6.683e-31	2.866e-30	1.456e-57	4.765e-32	4.765e-32	0
	NK	7.228e-153	1.714e-223	5.682e-202	4.108e-153	4.108e-153	0
	VIX	0	0	1.068e-205	0	0	0
Commodities Futures	CL	9.986e-120	4.860e-197	4.363e-175	8.748e-111	8.748e-111	0
	GC	9.059e-145	1.230e-137	1.067e-136	1.580e-115	1.580e-115	0
	HO	9.839e-46	1.986e-96	4.626e-82	1.176e-76	1.176e-76	0
	NG	8.537e-115	8.087e-158	1.178e-135	6.541e-98	6.541e-98	0
	ZC	1.118e-203	2.047e-274	1.180e-249	7.585e-181	7.585e-181	0
	ZS	0	0	0	0	0	0

P-values greater than the $\alpha = 0.01$ significance level are bolded, values greater than $\alpha = 0.05$ are italicized

B.2 Percent-Change Data

Asset Type	Asset	open	high	low	close	adjusted_close	volume
Stock	SPY	6.250e-28	5.814e-44	1.045e-38	9.772e-39	1.760e-39	2.322e-89
Bond	US 3-Year	5.193e-25	1.490e-27	1.492e-27	2.612e-23	2.612e-23	_
	US 5-Year	9.933e-32	4.925e-30	6.882e-31	1.164e-31	1.164e-31	_
	US 10-Year	2.488e-20	2.752e-25	3.808e-23	7.024e-23	7.024e-23	_
	US 30-Year	1.141e-04	4.836e-15	2.060e-26	7.554e-10	7.554e-10	0
	France 3-Year	2.431e-301	5.913e-306	5.708e-140	2.879e-58	2.879e-58	_
	France 5-Year	9.578e-256	4.445e-172	5.851e-133	4.337e-200	4.337e-200	_
	France 10-Year	6.865e-309	1.820e-310	4.050e-274	0	0	_
	Italy 3-Year	1.167e-265	9.489e-208	4.798e-155	1.500e-136	1.500e-136	_
	Italy 5-Year	2.892e-25	1.399e-34	3.476e-28	6.053e-26	6.053e-26	_
	Italy 10-Year	8.908e-16	1.146e-31	2.366e-15	6.820e-22	6.820e-22	_
	Japan 3-Year	1.094e-213	1.516e-128	1.694e-120	2.408e-117	2.408e-117	_
	Japan 5-Year	3.479e-66	2.671e-52	1.012e-72	2.868e-58	2.868e-58	_
	Japan 10-Year	4.393e-195	2.573e-193	2.282e-193	5.260e-288	5.260e-288	_
	UK 3-Year	3.315e-27	2.503e-20	3.099e-39	4.154e-24	4.154e-24	_
	UK 5-Year	4.429e-18	2.050e-23	2.743e-22	1.085e-21	1.085e-21	_
	UK 10-Year	2.727e-09	6.156e-18	2.211e-08	6.483e-10	6.483e-10	_
Forex	USD/AUD	1.908e-13	4.199e-20	1.543e-17	1.954e-16	1.954e-16	_
	USD/CAD	4.180e-07	7.403e-09	5.205e-11	1.343e-05	1.343e-05	0
	USD/CHF	1.781e-14	1.624e-17	1.404e-23	2.646e-15	2.646e-15	0
	USD/EUR	4.551e-05	7.129e-05	3.391e-05	6.674e-05	6.674e-05	9.553e-132
	USD/GBP	**1.068e-02**	**1.146e-02**	**4.617e-02**	***5.873e-02***	***5.873e-02***	0
	USD/JPY	1.419e-12	8.685e-13	1.581e-20	3.760e-10	3.760e-10	0
	EUR/AUD	2.866e-08	1.381e-13	2.292e-07	5.409e-10	5.409e-10	0
	EUR/CAD	1.138e-08	4.988e-08	8.424e-10	1.639e-07	1.639e-07	0
	EUR/CHF	1.019e-62	2.067e-65	1.900e-88	1.709e-61	1.709e-61	0
	EUR/GBP	3.134e-08	1.118e-16	6.239e-22	3.704e-06	3.704e-06	_
	EUR/JPY	6.262e-15	6.515e-16	3.840e-18	8.661e-12	8.661e-12	0
	JPY/AUD	7.385e-07	2.369e-09	1.440e-04	8.859e-07	8.859e-07	2.957e-202
	JPY/CAD	***2.861e-01***	***1.688e-01***	***4.452e-01***	***5.453e-01***	***5.453e-01***	1.254e-78
	JPY/CHF	2.488e-03	3.437e-05	1.182e-09	2.239e-05	2.239e-05	_
	JPY/GBP	8.708e-182	7.559e-178	3.101e-183	4.189e-180	3.993e-180	_
	GBP/AUD	5.367e-05	7.774e-09	4.526e-07	1.414e-04	1.414e-04	0
	GBP/CAD	7.377e-09	3.857e-11	8.284e-08	2.810e-08	2.810e-08	0
	GBP/CHF	7.138e-28	2.404e-38	2.531e-64	1.208e-28	1.640e-28	_
	AUD/CAD	4.104e-06	2.834e-06	2.042e-08	3.905e-04	3.905e-04	0
	AUD/CHF	5.415e-27	2.276e-33	7.056e-60	7.885e-34	2.127e-33	_
	CAD/CHF	3.606e-19	1.993e-28	1.037e-42	4.650e-18	5.738e-18	_
Index Futures	ES	9.666e-32	3.046e-45	8.691e-36	3.009e-43	3.009e-43	0
	RTY	2.550e-15	4.196e-21	9.716e-15	7.798e-20	7.798e-20	0
	FESX	7.275e-28	2.086e-33	1.465e-33	3.788e-22	3.788e-22	5.180e-214
	HSI	5.655e-05	1.301e-04	5.099e-07	1.498e-04	1.498e-04	5.887e-279
	NK	9.335e-09	7.780e-19	3.214e-22	3.913e-16	3.913e-16	3.790e-120
	VIX	4.673e-37	1.350e-48	5.230e-43	2.823e-48	2.823e-48	0
Commodities Futures	CL	7.784e-08	3.506e-17	6.113e-18	3.655e-07	3.655e-07	0
	GC	6.761e-15	4.024e-15	1.969e-21	1.137e-13	1.137e-13	0
	HO	3.724e-06	1.405e-13	2.082e-11	1.307e-06	1.307e-06	0
	NG	4.495e-10	1.646e-09	1.769e-11	1.817e-05	1.817e-05	9.530e-89
	ZC	1.609e-21	4.872e-33	8.068e-27	6.094e-15	6.094e-15	0
	ZS	5.074e-26	1.526e-40	8.726e-39	1.477e-22	1.477e-22	0

P-values greater than the $\alpha = 0.01$ significance level are bolded, values greater than $\alpha = 0.05$ are italicized

C Hole Analysis

C.1 Including Initial Gaps

Filename	Percent Missing	Number of Holes	Mean Hole	Std. Dev. of Hole	Maximum Hole
FR10Y.GBOND	61.314	71	63.085	308.459	2643
FR3Y.GBOND	61.834	70	64.529	341.752	2902
FR5Y.GBOND	61.821	70	64.514	341.755	2902
IT10Y.GBOND	50.376	11	334.545	1050.020	3655
IT3Y.GBOND	61.766	81	55.704	289.465	2643
IT5Y.GBOND	61.739	82	55.000	287.761	2643
JP10Y.GBOND	62.464	98	46.561	269.713	2700
JP3Y.GBOND	63.737	135	34.489	231.158	2706
JP5Y.GBOND	63.737	134	34.746	232.605	2713
UK10Y.GBOND	63.765	5	931.600	1858.700	4649
UK3Y.GBOND	62.341	90	50.600	297.112	2851
UK5Y.GBOND	61.930	90	50.267	275.090	2643
US10Y.GBOND	1.561	57	2.000	0.000	2
US3Y.GBOND	61.985	97	46.680	265.289	2643
US5Y.GBOND	2.300	84	2.000	0.000	2
AUDCAD.FOREX	0.027	1	2.000	0.000	2
AUDCHF.FOREX	0.027	1	2.000	0.000	2
CADCHF.FOREX	0.027	1	2.000	0.000	2
EURAUD.FOREX	0.027	1	2.000	0.000	2
EURCAD.FOREX	0.027	1	2.000	0.000	2
EURCHF.FOREX	0.027	1	2.000	0.000	2
EURGBP.FOREX	0.027	1	2.000	0.000	2
EURJPY.FOREX	0.027	1	2.000	0.000	2
GBPAUD.FOREX	17.084	1	1248.000	0.000	1248
GBPCAD.FOREX	0.027	1	2.000	0.000	2
GBPCHF.FOREX	0.027	1	2.000	0.000	2
JPYAUD.FOREX	0.027	1	2.000	0.000	2
JPYCAD.FOREX	0.027	1	2.000	0.000	2
JPYCHF.FOREX	0.027	1	2.000	0.000	2
JPYGBP.FOREX	0.027	1	2.000	0.000	2
USDAUD.FOREX	0.233	7	2.429	0.728	4
USDCAD.FOREX	0.027	1	2.000	0.000	2
USDCHF.FOREX	0.027	1	2.000	0.000	2
USDEUR.FOREX	0.027	1	2.000	0.000	2
USDGBP.FOREX	0.151	5	2.200	0.400	3
USDJPY.FOREX	0.000	0	0.000	0.000	0
CL.COMM	5.380	79	4.975	26.046	235
ES.COMM	5.777	81	5.210	28.487	260
FESX.COMM	2.081	53	2.868	0.870	5
GC.COMM	5.585	83	4.916	26.182	242
HO.COMM	31.910	97	24.031	214.635	2127
HSI.COMM	51.280	54	69.370	486.344	3610
NG.COMM	48.446	69	51.290	406.212	3401
NK.COMM	39.343	16	179.625	687.164	2841
RTY.COMM	38.535	15	187.667	694.701	2787
US.COMM	2.053	75	2.000	0.000	2
VIX.COMM	48.501	71	49.901	400.533	3401
ZC.COMM	4.969	83	4.373	21.273	197
ZS.COMM	3.094	109	2.073	0.763	10
SPY.US	3.682	131	2.053	0.356	5

C.2 Excluding Initial Gaps

Filename	Percent Missing	Number of Holes	Mean Hole	Std. Dev. of Hole	Maximum Hole
FR10Y.GBOND	39.382	70	26.229	7.918	31
FR3Y.GBOND	36.680	69	23.406	10.555	32
FR5Y.GBOND	36.657	69	23.391	10.583	32
IT10Y.GBOND	0.685	10	2.500	0.500	3
IT3Y.GBOND	40.090	80	23.363	10.743	31
IT5Y.GBOND	40.047	81	23.049	10.927	32
JP10Y.GBOND	40.456	97	19.206	12.715	34
JP3Y.GBOND	42.401	134	14.552	13.206	34
JP5Y.GBOND	42.313	133	14.609	13.177	34
UK10Y.GBOND	0.339	4	2.250	0.433	3
UK3Y.GBOND	38.235	89	19.135	12.747	32
UK5Y.GBOND	40.347	89	21.135	12.040	31
US10Y.GBOND	1.561	57	2.000	0.000	2
US3Y.GBOND	40.433	96	19.635	12.755	31
US5Y.GBOND	2.300	84	2.000	0.000	2
AUDCAD.FOREX	0.027	1	2.000	0.000	2
AUDCHF.FOREX	0.027	1	2.000	0.000	2
CADCHF.FOREX	0.027	1	2.000	0.000	2
EURAUD.FOREX	0.027	1	2.000	0.000	2
EURCAD.FOREX	0.027	1	2.000	0.000	2
EURCHF.FOREX	0.027	1	2.000	0.000	2
EURGBP.FOREX	0.027	1	2.000	0.000	2
EURJPY.FOREX	0.027	1	2.000	0.000	2
GBPAUD.FOREX	0.000	0	0.000	0.000	0
GBPCAD.FOREX	0.027	1	2.000	0.000	2
GBPCHF.FOREX	0.027	1	2.000	0.000	2
JPYAUD.FOREX	0.027	1	2.000	0.000	2
JPYCAD.FOREX	0.027	1	2.000	0.000	2
JPYCHF.FOREX	0.027	1	2.000	0.000	2
JPYGBP.FOREX	0.027	1	2.000	0.000	2
USDAUD.FOREX	0.233	7	2.429	0.728	4
USDCAD.FOREX	0.027	1	2.000	0.000	2
USDCHF.FOREX	0.027	1	2.000	0.000	2
USDEUR.FOREX	0.027	1	2.000	0.000	2
USDGBP.FOREX	0.151	5	2.200	0.400	3
USDJPY.FOREX	0.000	0	0.000	0.000	0
CL.COMM	5.559	79	4.975	26.046	235
ES.COMM	5.990	81	5.210	28.487	260
FESX.COMM	2.081	53	2.868	0.870	5
GC.COMM	5.777	83	4.916	26.182	242
HO.COMM	3.940	96	2.125	0.845	10
HSI.COMM	3.681	53	2.566	0.922	5
NG.COMM	3.535	68	2.029	0.241	4
NK.COMM	0.739	15	2.200	0.542	4
RTY.COMM	0.620	14	2.000	0.000	2
US.COMM	2.054	75	2.000	0.000	2
VIX.COMM	3.637	70	2.029	0.237	4
ZC.COMM	5.107	83	4.373	21.273	197
ZS.COMM	3.094	109	2.073	0.763	10
SPY.US	3.683	131	2.053	0.356	5

D Outliers

Raw Data		Percent-Change Data	
Asset	Outliers (%)	Asset	Outliers (%)
JPY/AUD	306 (5.16%)	EURO STOXX 50 Futures	269 (5.29%)
USD/GBP	239 (4.16%)	SPDR S&P 500 Trust	247 (4.91%)
USD/EUR	193 (3.43%)	E-Mini S&P 500 Futures	225 (4.62%)
JPY/CAD	159 (2.68%)	Corn Futures	184 (3.79%)
GBP/CAD	100 (1.69%)	AUD/CHF	173 (3.03%)
SPDR S&P 500 Trust	96 (1.91%)	E-Mini Russell 2000 Futures	160 (5.09%)
USD/CAD	89 (1.50%)	EUR/GBP	157 (3.01%)
EURO STOXX 50 Futures	88 (1.71%)	Crude Oil Futures	157 (3.23%)
CBOE Volatility Index Futures	82 (3.05%)	US 5-Year Bonds	156 (2.75%)
Gold Futures	81 (1.64%)	CAD/CHF	153 (2.68%)
AUD/CAD	79 (1.33%)	USD/AUD	151 (2.90%)
EUR/JPY	71 (1.19%)	Gold Futures	150 (3.38%)
USD/JPY	69 (1.20%)	JPY/CHF	147 (2.82%)
EUR/AUD	62 (1.05%)	US 10-Year Bonds	145 (2.65%)
Soy Futures	62 (1.20%)	US 30-Year Bond Futures	140 (2.78%)
Italy 3-Year Bonds	58 (2.70%)	GBP/CHF	131 (2.29%)
EUR/CAD	58 (0.98%)	Heating Oil Futures	121 (3.44%)
Crude Oil Futures	54 (1.09%)	JPY/AUD	118 (3.67%)
GBP/AUD	51 (1.05%)	Italy 10-Year Bonds	116 (4.02%)
Italy 5-Year Bonds	45 (2.09%)	USD/CAD	109 (3.43%)
EUR/CHF	45 (0.78%)	EUR/JPY	108 (3.37%)
Corn Futures	43 (0.87%)	EUR/AUD	104 (3.26%)
E-Mini S&P 500 Futures	40 (0.81%)	EUR/CAD	101 (3.15%)
France 3-Year Bonds	37 (1.73%)	USD/JPY	99 (3.09%)
France 5-Year Bonds	37 (1.78%)	Nikkei 225 Futures	99 (3.34%)
US 30-Year Bond Futures	36 (0.70%)	Natural Gas Futures	97 (3.61%)
USD/CHF	35 (0.61%)	GBP/CAD	95 (2.99%)
Nikkei 225 Futures	33 (1.03%)	AUD/CAD	93 (2.90%)
France 10-Year Bonds	30 (1.49%)	CBOE Volatility Index Futures	93 (3.46%)
Natural Gas Futures	30 (1.11%)	Soy Futures	90 (2.35%)
Japan 10-Year Bonds	29 (1.47%)	Italy 5-Year Bonds	88 (4.09%)
Japan 5-Year Bonds	29 (1.53%)	UK 5-Year Bonds	86 (4.28%)
JPY/CHF	29 (0.56%)	GBP/AUD	84 (2.63%)
UK 5-Year Bonds	27 (1.34%)	US 3-Year Bonds	82 (4.14%)
Japan 3-Year Bonds	26 (1.37%)	UK 3-Year Bonds	80 (4.04%)
UK 3-Year Bonds	24 (1.21%)	France 5-Year Bonds	76 (3.65%)
US 3-Year Bonds	22 (1.11%)	UK 10-Year Bonds	71 (3.72%)
E-Mini Russell 2000 Futures	17 (0.53%)	USD/CHF	69 (2.15%)
Heating Oil Futures	14 (0.39%)	Italy 3-Year Bonds	64 (2.98%)
Hang Seng Index Futures	12 (0.47%)	EUR/CHF	63 (1.97%)
JPY/GBP	1 (0.02%)	Japan 3-Year Bonds	57 (3.01%)
Italy 10-Year Bonds	0 (0%)	France 3-Year Bonds	54 (2.53%)
UK 10-Year Bonds	0 (0%)	Hang Seng Index Futures	47 (3.09%)
US 10-Year Bonds	0 (0%)	USD/EUR	44 (6.57%)
US 5-Year Bonds	0 (0%)	Japan 10-Year Bonds	40 (2.03%)
AUD/CHF	0 (0%)	France 10-Year Bonds	22 (1.09%)
CAD/CHF	0 (0%)	Japan 5-Year Bonds	22 (1.16%)
EUR/GBP	0 (0%)	JPY/CAD	11 (2.06%)
GBP/CHF	0 (0%)	USD/GBP	10 (1.25%)
USD/AUD	0 (0%)	JPY/GBP	2 (0.28%)

E Dataset Performance (Ranked by Macro F1)

Rank	Asset Types	Transform(s)	Accuracy	Macro F1	Weighted F1
1	I, C	%	0.571 ± 0.019	0.548 ± 0.015	0.562 ± 0.014
2	B, I, C	%	0.544 ± 0.041	0.528 ± 0.038	0.541 ± 0.039
3	B, I, C	% PCA	0.541 ± 0.023	0.522 ± 0.020	0.536 ± 0.020
4	F, I	Raw	0.541 ± 0.010	0.520 ± 0.013	0.532 ± 0.010
5	B, C	%	0.537 ± 0.017	0.512 ± 0.034	0.527 ± 0.028
6	I, C	% PCA	0.539 ± 0.032	0.511 ± 0.021	0.527 ± 0.023
7	B, I	%	0.535 ± 0.019	0.511 ± 0.015	0.524 ± 0.015
8	B	%	0.528 ± 0.023	0.510 ± 0.017	0.521 ± 0.018
9	F, I	%	0.536 ± 0.030	0.510 ± 0.026	0.522 ± 0.026
10	B, I	% PCA	0.537 ± 0.028	0.510 ± 0.015	0.523 ± 0.018
11	F, B, I, C	%	0.532 ± 0.011	0.509 ± 0.020	0.525 ± 0.014
12	B, C	% PCA	0.536 ± 0.028	0.509 ± 0.030	0.524 ± 0.028
13	I	Raw	0.520 ± 0.037	0.508 ± 0.031	0.515 ± 0.036
14	F, B, I, C	% PCA	0.534 ± 0.019	0.503 ± 0.030	0.522 ± 0.026
15	F, B, I	%	0.521 ± 0.023	0.502 ± 0.024	0.513 ± 0.022
16	F, B	%	0.533 ± 0.024	0.501 ± 0.016	0.516 ± 0.015
17	F, I, C	%	0.521 ± 0.032	0.501 ± 0.027	0.515 ± 0.028
18	I	%	0.528 ± 0.014	0.501 ± 0.014	0.514 ± 0.012
19	F, B	% PCA	0.522 ± 0.023	0.498 ± 0.018	0.510 ± 0.018
20	F, B, C	% PCA	0.518 ± 0.026	0.496 ± 0.018	0.510 ± 0.019
21	I, C	Raw	0.544 ± 0.028	0.494 ± 0.021	0.512 ± 0.017
22	F, I, C	% PCA	0.517 ± 0.029	0.493 ± 0.025	0.507 ± 0.025
23	F, B, C	%	0.511 ± 0.024	0.492 ± 0.013	0.505 ± 0.015
24	B	Raw PCA	0.519 ± 0.015	0.492 ± 0.025	0.504 ± 0.017
25	F	%	0.512 ± 0.012	0.492 ± 0.005	0.502 ± 0.003
26	I	% PCA	0.526 ± 0.008	0.490 ± 0.019	0.505 ± 0.016
27	F, B	Raw PCA	0.504 ± 0.014	0.490 ± 0.011	0.498 ± 0.012
28	F, B, I, C	Raw	0.529 ± 0.020	0.488 ± 0.025	0.505 ± 0.022
29	F	% PCA	0.528 ± 0.010	0.488 ± 0.028	0.503 ± 0.023
30	F, B, C	Raw PCA	0.500 ± 0.019	0.488 ± 0.016	0.495 ± 0.013
31	F, C	%	0.525 ± 0.026	0.486 ± 0.023	0.506 ± 0.019
32	B, I, C	Raw	0.525 ± 0.018	0.486 ± 0.030	0.497 ± 0.023
33	F, B, C	Raw	0.518 ± 0.026	0.486 ± 0.032	0.496 ± 0.022
34	F, B	Raw	0.508 ± 0.025	0.486 ± 0.043	0.493 ± 0.034
35	F, I	% PCA	0.514 ± 0.027	0.485 ± 0.023	0.498 ± 0.023
36	B	Raw	0.523 ± 0.033	0.484 ± 0.022	0.491 ± 0.036
37	B	% PCA	0.513 ± 0.026	0.481 ± 0.035	0.496 ± 0.030
38	B, I	Raw	0.527 ± 0.032	0.480 ± 0.039	0.495 ± 0.031
39	SPY Only	%	0.527 ± 0.021	0.480 ± 0.029	0.495 ± 0.019
40	B, I	Raw PCA	0.545 ± 0.005	0.477 ± 0.017	0.499 ± 0.015
41	C	% PCA	0.505 ± 0.015	0.477 ± 0.027	0.494 ± 0.020
42	F, I, C	Raw	0.534 ± 0.027	0.477 ± 0.041	0.493 ± 0.034
43	I, C	Raw PCA	0.523 ± 0.031	0.477 ± 0.047	0.492 ± 0.033
44	C	%	0.509 ± 0.025	0.475 ± 0.022	0.495 ± 0.022
45	F, C	% PCA	0.519 ± 0.017	0.474 ± 0.013	0.496 ± 0.009
46	SPY Only	% PCA	0.530 ± 0.022	0.474 ± 0.038	0.490 ± 0.026
47	B, C	Raw PCA	0.537 ± 0.030	0.472 ± 0.036	0.495 ± 0.033
48	F, B, I, C	Raw PCA	0.520 ± 0.034	0.472 ± 0.020	0.491 ± 0.021
49	F, B, I	% PCA	0.517 ± 0.036	0.472 ± 0.045	0.489 ± 0.040
50	SPY Only	Raw PCA	0.503 ± 0.030	0.472 ± 0.052	0.471 ± 0.067

E.1 Dataset Performance (cont.)

51	B, I, C	Raw PCA	0.550 ± 0.015	0.468 ± 0.046	0.493 ± 0.038
52	F, B, I	Raw	0.538 ± 0.023	0.465 ± 0.046	0.484 ± 0.029
53	F, I	Raw PCA	0.527 ± 0.036	0.464 ± 0.053	0.480 ± 0.042
54	C	Raw	0.511 ± 0.041	0.463 ± 0.028	0.476 ± 0.025
55	I	Raw PCA	0.541 ± 0.019	0.462 ± 0.036	0.487 ± 0.027
56	F, B, I	Raw PCA	0.531 ± 0.029	0.461 ± 0.027	0.481 ± 0.021
57	F, C	Raw	0.526 ± 0.036	0.460 ± 0.051	0.477 ± 0.035
58	C	Raw PCA	0.530 ± 0.020	0.458 ± 0.024	0.480 ± 0.015
59	F, C	Raw PCA	0.529 ± 0.021	0.455 ± 0.048	0.477 ± 0.036
60	F	Raw PCA	0.537 ± 0.017	0.454 ± 0.064	0.474 ± 0.048
61	Brownian Motion	Raw	0.508 ± 0.035	0.452 ± 0.075	0.456 ± 0.090
62	F	Raw	0.486 ± 0.025	0.450 ± 0.054	0.448 ± 0.071
63	B, C	Raw	0.503 ± 0.024	0.449 ± 0.041	0.454 ± 0.057
64	F, I, C	Raw PCA	0.531 ± 0.031	0.442 ± 0.031	0.468 ± 0.020
65	Normal Sample	%[11]	0.506 ± 0.045	0.435 ± 0.070	0.442 ± 0.074
66	SPY Only	Raw	0.516 ± 0.051	0.412 ± 0.049	0.426 ± 0.062

[11] The normal sample dataset is considered to be a percent-change dataset as its data is sampled from a distribution designed to be similar to the distribution of percent-changes of the S&P 500.