Investigation of Financial Forecasting Systems: A Survey and Comprehensive Examination

Wren Paris-Moe, Computer Science and Mathematics Major, Occidental College

September 15, 2023

Abstract

The stock market has the allure to many investors who believe they can beat market returns. The tools investment firms use to analyze the market are constantly evolving to give their developers advantages over the competition. The latest example is using major technological advancements to analyze information and outsmart the market. New developments in computing capabilities offer potential to bring in greater profits. Today 80% of trades are automated indicating a massive change in how traditional trading is done. Bringing computing into stock price analysis puts mathematicians and computer scientists at the forefront of the industry. However, the question whether computational forecasting techniques can consistently beat the market is still unknown and highly debated. Few studies exist that provide a complete analysis of the performances of financial prediction systems as a whole. However, none consist of holistic component-wise comparisons of entire financial time series forecasting systems. This research offers a comprehensive study of how combinations of input data, preprocessing methods, feature engineering techniques, and popular prediction models interact when tasked with predicting the prices of futures. Specific arrangements are determined that provide the most optimal systems for each model. Research procedures show models can predict prices within a range of high accuracy. Though predicting market directions, up or down, with consistent success to beat the market is more challenging. Ultimately, support vector regression and Gaussian processes demonstrate the most consistent ability to produce low errors while also predicting market directions accurately.

1 Introduction

The stock market is evolving due to advancements in computing. The work of mathematicians, computer scientists, and financial engineers has become increasingly important to the industry. Developing profitable prediction models is a focus of both small- and large-scale financial institutions. The accurately predicting market movements is incredibly difficult. Producing a successful financial trading system on model predictions has the potential to provide lucrative profits to its developer. Information on the best models to forecast the market is not public knowledge. The greatest challenge for a developer is determining which methods to use. There are countless models that can be applied to financial trading and forecasting. However, no method has yet to be universally agreed upon as the best. Specific factors cause algorithms to fit certain indices, markets, or tasks better than others [25, 36]. The lack of a known algorithm that universally fits the market drives the ongoing study of financial prediction systems.

This project proposes a comprehensive study of each aspect of financial forecasting systems. The question stands whether it is possible to establish the information and methods required to accurately predict market prices and directions. The primary considerations for the study are data input, data preprocessing methods, feature engineering techniques, and forecasting models. Prediction models fall into two overarching classes: hard computing and soft computing. Hard computing is conventional programming where rules and conditions are precisely coded into the structure of an algorithm. These methods tend to be simpler to understand and implement. Soft computing models are more complex, incorporating structures such as neural networks (NNs) and genetic algorithms (GAs). Broadly defined, soft computing is the use of estimated calculations that provide imprecise but informative solutions to computationally complex problems. Soft computing approaches exploit tolerance for imprecision, uncertainty, and partial truth to solve their applications. A common debate among experts is whether soft computing techniques can consistently outperform their more simplistic counterparts in hard computing.

The most popular financial prediction models are grouped into three distinct subcategories: classical statistical methods, machine learning (ML) techniques, and deep learning (DL) frameworks. Statistical methods are a prime example of hard computing while ML and DL fall under soft computing. Studying the applications of hard and soft computing in finance provides relevant and valuable information to other areas. How components interact and influence one another with financial prediction systems is knowledge directly applicable to any task of time series analysis. This research offers insight to such interactions and the performances that come from specific model combinations. Thus, this survey serves as a resource for programmers aspiring to build and learn about the foundations of financial forecasting systems.

1.1 Statement of Problem

Stock price forecasting is an exceedingly complex subject. The primary factor lies within the structure of financial time series data. A time series is identified as a data set formed by series of values obtained at successive times. Financial time series consists of historical open, high, low, and close (OHLC) prices, as well as recorded volume. In the most basic sense, time series forecasting analyzes past OHLC data to make predictions of future stock prices [21]. The aim is to find or build a function capable of accurately mapping current and past values to future ones.

There are numerous reasons why accurately forecasting financial time series is a difficult problem. Relative to other types of time series, financial time series are particularly inconsistent. This presents a massive hurdle for mathematicians and computer scientists attempting to predict market movements. Financial time series are inherently embedded with extreme levels of randomness and high volatility. This characteristic of time series data is known as noise. The nonlinearity and randomness of financial time series makes building forecasting models a challenging tasks. [5, 22, 40]. The unpredictability of financial time series is defined by the Efficient Market Hypothesis (EMH).

1.1.1 Efficient Market Hypothesis

The EMH states that at any given time, the current price of a share presents all information that can be known about a stock. According to the EMH, day to day variations in stock prices are completely random. The theory proposes that all known financial information leads to a completely unpredictable and random market. Thus, financial time series are formed by a "random walk" making it impossible for investors to beat the market [45, 5]. Random and unpredictable data can

be the most challenging to process within a programming model used for the prediction of future stock prices. Market prices are not determined internally, but rather by external variables. Events such as rumors, pandemics, and presidential tweets control market directions, not current and past share prices.

The goal of financial experts is to prove the EMH false. Many studies exist claiming their models can consistently beat the market [3, 5, 13, 18, 19, 21, 24, 25, 32, 39, 38, 44]. If certain prediction models can indeed perform better than a "random walk" then the authenticity of the EMH is challenged in true practice. Such studies claim that stock movements are not truly random but instead derived from dynamic systems so complex that no individual can fully master them [24]. This leads financial engineers to develop intricate forecasting systems that incorporate computational intelligence to prove the EMH wrong.

1.1.2 System Component-Wise Diversity

Input Data

Research on the applications of soft computing to finance has increased exponentially over the past decade [38]. However, there is little agreement among experts on most questions pertaining to the field. The specific components that build optimal trading systems are unknown. Determining input data is the first step to building financial prediction systems. Technical analysis indicators are frequently incorporated in model feature spaces. Lv found that this practice can greatly improve errors for specific ML algorithms while it having little to no benefit with others [24]. Extensive academic research has been done to investigate which combinations of input data are best. However, there is no conclusion on the issue. Common within ML, the problem must be analyzed at an individual level. Adding more complexity, the usefulness of a given feature can very depending on what preprocessing method is applied. Research is needed to evaluate the effect of different inputs on the performance of financial forecasting systems.

Preprocessing

In soft computing, data must be cleaned and reformatted to be structured as proper input. Preprocessing methods alter input data so it is compatible with a given model. The methods are necessary prerequisites to prediction algorithms, especially with ML and DL based techniques [31]. The broad class of data preprocessing includes many subcategories such as cleaning, normalization, and feature extraction. Within these are additional precisely defined subclasses used for specific cases of preprocessing. Scaling, standardization, normalization, feature extraction, and feature selection are the methods focused on in this research. Scaling, standardization, and normalization concentrate on adjusting and reorganizing values to have the proper structure for a given forecasting algorithm. Such techniques reduce bias towards larger values and outliers in soft computing projections [39, 40]. The methods have different effects depending on the required input range and distribution for the forecasting algorithm in use. Combinatorially testing the methods provides insight to how they interact with the other components of a given system. The specific meaning of each term is as follows:

i) **Scaling:** Adjusting the range of values for each feature where the data distribution remains generally unaltered holding its initial properties. Minimum and maximum values are used to achieve wanted scalings, either to the range $[-1, 1]$ or $[0, 1]$.

ii) **Standardization:** Removing the mean and scaling by the standard deviation so the data distribution exhibits a statistical unit-variance. Standardization does not imply data is within the range $[-1, 1]$ or $[0, 1]$.

iii) **Normalization:** Transforming data to force a more Gaussian-like distribution. Zero-mean and unit-variance are applied to the transformed feature space. The result is not within the range $[-1, 1]$ or $[0, 1]$.

Feature selection and extraction focus on reformatting the feature space of input data. The broad category is known as feature engineering. Feature selection algorithms deduce a subset of the original feature space to use as input. The resulting feature space has reduced dimensionality and is more informative and less redundant. Feature extraction derives an alternatively structured feature set from the original space so information, patterns, and trends are more apparent within input data [30]. Techniques of both types provide distinct, restructured feature spaces with narrowed dimensionality so they can better fit predictive models.

Lastly, data must be split into disjointed sets for training and testing predictive models. The procedure is essential to preprocessing and leads to considerable errors or model misinterpretations when done incorrectly [40, 42]. Splits must ensure training data is never too small relative to global trends in inputs and the size of testing data. Relevant studies use a wide range of ratios and methods for training and testing financial prediction systems. Though, the problem is more complex than traditional train-test splitting due to the continuous time series structure of the data. Rolling training methods like k-fold validation and walk-forward analysis can be utilized to minimize these errors [24]. Few studies implement each train-test strategies the same leaving a wide range of techniques, though, with no instruction on which to select.

Selecting the ideal preprocessing methods for a soft computing model is not as simple as one would hope. Each method has a distinct set of behaviors, advantages, and disadvantages making it unique. A performance boost due to a preprocessing method is entirely dependent on how it interacts with system components. Additionally, employing a slightly different set of parameters can lead to varying outcomes. Determining the optimal way to implement the steps of preprocessing methods within a forecasting system is essential to its success. This study offers insight to optimal preprocessing selections based on combinations of data types and prediction models.

Forecasting Models

The central component to financial forecasting systems is the selected regression algorithm. Combining all statistical, ML, and DL frameworks, there is an abundance of available regression models. The issue is determining which are superior for financial time series forecasting. Algorithms must be capable of noisy time sequence based predictions in order to forecast values accurately. Statistical methods are relatively simple, easy to implement, and often without the need for extreme data processing and/or supplementary components. Precisely defined, traditional coding structures generally give statistical forecasting models consistent results with low maximum errors. Conversely, Soft computing techniques are incredibly task dependent meaning they will perform exceptionally well for one application but fail miserably with another. Often minor details of implementation can significantly alter the predictions of ML or DL models causing underfitting or overfitting to occur. However, their complex, novel based structures offer the potential to discover hidden nonlinear relationships within data samples. ML and DL frameworks are more complicated to implement due to a heavy reliance on "sufficient" input data. Thus, soft computing techniques depend on receiving adequately preprocessed input data. A combinatorial testing procedure as proposed in this paper evaluates changes in model performance based on varying sets of those components. This allows for simple analysis of feature importance, or preprocessing significant relative to each forecasting model.

ML and especially DL based models are highly dependent on parameter optimization [8,

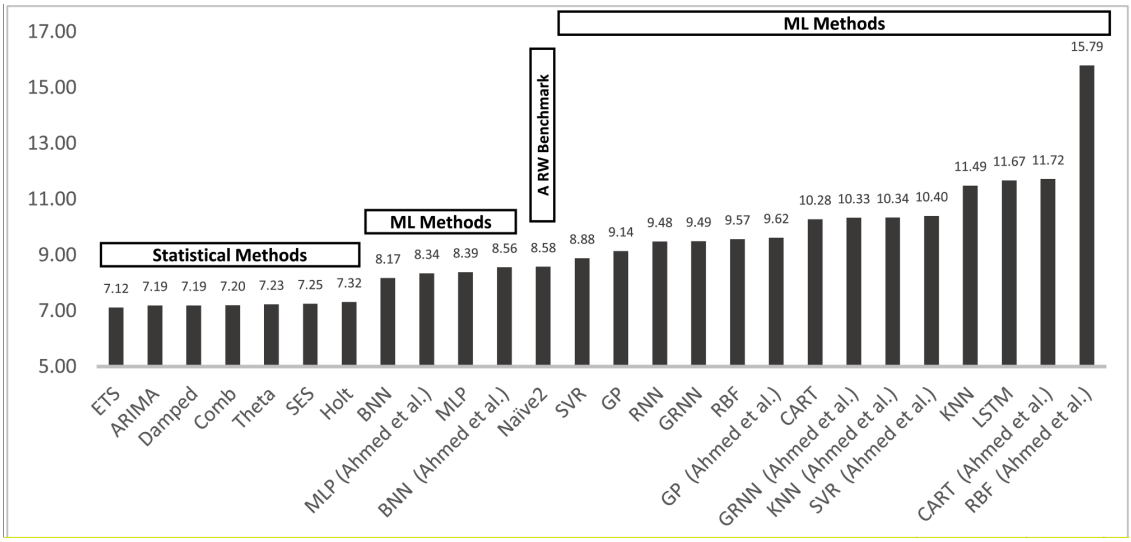

Figure 1: Performance of ML, DL, and statistical forecasting methods measured by MAPE [28].

11, 46]. This makes implementing soft computing techniques a challenge. A change in a single parameter can drastically change how a model functions and performs against forecasting the market. Referring to Figure 1, Makridakis [28] included variations of algorithms from an alternate study in his analysis of numerous forecasting models. There are several cases where the equivalent models place differently in the rankings of resulting symmetrical mean absolute percentage errors (sMAPE). As discussed in the study, this was an issue of different parameter selections and implementation details Sometimes this lead to vastly different metrics. Specifically, the radial basis function (RBF) Developed by Makridakis produced an sMAPE of 9.57% while the additional version gave an sMAPE of 15.79%. The clearly apparent differences in performance due to details of implementation illustrates the complexity of soft computing and its applications. This issue highlights the difficulty and importance of producing reproducibility in research. A major factor is the specific and inherent complexity of forecasting financial time series data. Remember, by standards of the EMH, attempting to forecast market prices and directions is theoretically impossible. Consequently, it is not infrequent that studies arrive at dissimilar conclusions due to specific individual implementations of identical models resulting in contradictory claims to be made. This problem further indicates the need for a comprehensive evaluation of the components foundational to financial forecasting systems. Such analysis establishes a needed step to serve as a basis to conducting additional research of financial prediction models.

1.1.3 Additional Problems

Finance Industry

Finance is an elitist and secretive sector. The sole motive of companies is to maximize their excess profits. The trading and forecasting algorithms developed by finance companies are kept private to have competitive advantages. Thus, the entities conducting the most research in the field have an incentive to ensure their innovations remain confidential. This trend defines the industry where selfish tendencies commonly prevent groundbreaking innovations from becoming public knowledge. This adds to the difficulty of determining the optimal components to a financial forecasting system.

Issues with Online Sources

Establishing the best methods to use in a financial forecasting model is a central question to this research. However, available resources fail to sufficiently answer many of the issues outlined previously. There is an abundance of internet sources relating to financial prediction systems. However, the majority of online sources are flawed or offer incomplete solutions. Tutorials offer narrow and oversimplified perspectives to an incredibly broad and complex problem. Online sources often provide misleading material to their readers. A common example is using the assumption that traditional train-test splitting can be applied to time series forecasting. Such resources are incapable of providing the knowledge needed to understand the inherent intricacies of financial time series forecasting. The scope of these sources is either too large or small. An article can cover many categories but with little detail making them deceptively valuable to programmers. Oppositely, an article will emphasize a particular aspect in detail or illustrate a single application of strategy. This can be greatly informative when wanting to study a sole component of a system. However, for the individual attempting to master the field and build the ultimate forecasting model this becomes an impractical venture. Hundreds of hours at minimum would be spent sifting through endless online resources with substantial time spent studying subcategories of topics irrelevant to overall ambition.

The Limitations of Scholarly Articles

Other than online sources, it is necessary to seek out scholarly articles and established research to learn the most important details about financial prediction systems. The most relevant and profound discoveries discussed in immense detail occur in these types of articles. There is ample research and scholarly work that surveys the use of soft computing techniques in financial trading systems. This type of work is meant for updating other academics on the most recent findings in their field. Explained with such depth, authors concentrate on specific details that are unimportant to the average individual. Not to mention the complex language that one must learn to understand an article. Making it even more difficult, most of these sources are unavailable to the general public. Individuals must be a part of an academic or research institution or pay expensive fees to receive access to databases where scholarly sources are stored. Essentially, the most informative sources from accredited research projects are extremely difficult to access, use complex language, and focus on details insignificant to the average reader. These works further their fields in academia but do little to help the individual attempting to teach himself how to build a model himself.

By far the largest burden to investors, scholarly literature is largely inconclusive on many issues pertaining to building the optimal forecasting system. Too much debate exists in the literature to objectively define one model or class of models as the best financial forecasting technique. There is an ongoing debate among experts on which input types, preprocessing methods, and forecasting algorithms create the best systems. As a primary question still facing the field, it is unknown whether ML or DL based prediction algorithms can consistently beat a random walk or statistical method [28, 27]. Often studies use different evaluation techniques making it impossible to compare findings. Research can investigate the same topic but use slightly different parameter sets leading to vastly different claims. Studies can utilize distinct preprocessing methods for the same prediction model making it impossible to make an accurate inference about the superiority of the method. With so much speculation from those who are supposed to have the answers, I want to find an answer.

1.2 Goals and Research Questions

This research has one aim: to find the ideal three components to build the perfect financial prediction system. This field so complex not even the most qualified personnel have the answers to these questions. The preliminary research did not reveal a source that had a specific focus and structure, testing a wide range of prospective forecasting models by individually examining each combination from a plethora of chosen preprocessing methods, predictive algorithms, and input types. As no source exists, the best route will be to conduct the research by building a comprehensive study of optimal stock price forecasting models. To learn how to build the optimal trading system, there is no better way to become an expert on a subject than to study the exact intricacies of each component that go into that subject.

The audience for this research proposal is the aspiring financial engineers wanting to build stock price forecasting systems for the first time. Assuming they have failed to learn the exact components of a perfect system, research tuned specifically for their needs could be immensely helpful. A series of tests on the main aspects of proposed prediction models backed by extensive data and performance evaluation would be an invaluable guide for future research. Additionally, it would be a great contribution to public knowledge in the field.

For each of the main system components outlined, a set of the most notable techniques will be carefully selected to use in the study. Through extensive implementation and testing, all combinations of the chosen input types, preprocessing methods, and prediction models will be examined. A standard set of evaluation metrics is employed to measure the performance of each combination. This creates sufficient data to allow an attempt at making an objective claim. Though in soft computing, models are expected to experience varying performance, succeeding on a case by case basis. With the current state of technology, it is impractical to objectively claim that one model, a class of models, or a combination of models will always outperform all others. Such an assertion requires an algorithm to have predictive capabilities so outstanding that no variable can reduce its performance. So, no matter the level of noise, unique trends in the data, or evaluation metrics used to measure performance, the proposed model would miraculously reign supreme. This concept may be possible once AI can flawlessly beat a Turing test, but sadly we are yet to achieve such a feat. Thus, any attempt to make such a claim indicates a flawed application of the scientific method. Therefore, the focus of this research is not to make an objective generalized claim on the best system, but rather to discover and analyze the specific variables and conditions that allow for given model combinations to succeed. By testing all combinations and analyzing any notable trends, insight is given to the specific benefits and faults of each method included in the study.

The following research goals arise from the specified problems:

1. Investigate the use cases for machine learning and deep learning prediction models and criteria under which they perform well, and specifically whether soft computing techniques can consistently outperform their simpler statistical counterparts.

2. Determine the combinations of input types, preprocessing methods, and prediction algorithms that create ideal systems for financial forecasting.

3. By testing all component-wise arrangements of financial forecasting systems with a standardized evaluation procedure, discover and analyze trends within the data that offer insight into the hidden interactions between components.

2 Background and Related Research

Using 3003 different time series, the M3 Competition in 2000 was the first large scale study to compare neural networks (NNs) and ML techniques to traditional statistical methods. The results indicate that the newer innovative techniques performed averagely or even worse in comparison to classical ones [28, 27]. Specifically, no ML or DL model outperformed the theta statistical method. Since 2000, thousands of new developments have arisen allowing for debate over whether the claims of the M3 Competition still stand. More recently, in 2019 another large-scale study examined ML and DL strategies used for stock trading algorithms [24]. It analyzed 424 S&P 500 and 185 CSI stock indices to compare six ML algorithms and six deep neural network (DNN) models. They found that ML techniques performed better than the DNN based models in almost every evaluation metric.

2.1 Literature on Evaluation Criteria

Comparing unique models from different studies is one of the most difficult tasks preventing anyone from objectively defining one method as better than another. There is a enormous range of methods used to compare the successes of proposed models, and there is no industry standard. This is a tricky issue since it allows for studies to make contradictory claims when evaluation criteria cannot easily be converted to formats where they can be compared. Therefore, the performance of proposed prediction models may tend to be overly optimistic [24]. Metrics from training algorithms such as accuracy, precision, and recall are commonly used as model performance indicators. Error measurements like final prediction error (FPE), mean squared error (MSE), and mean absolute percentage error (MAPE) are also used by studies like [5, 40, 46] to compare the success of models.

A great model comparison technique is implementing a price prediction algorithm as a buy-and-sell strategy [5]. This can be used as a general evaluation strategy by comparing proposed models by the profits they bring in. Additionally, time and space complexities should be considered when analyzing any algorithm even though it is left out of most studies included in this paper's research. There are many examples of miscellaneous criterion like approximate entropy utilized in [22]. This metric evaluates the quality of a time series before and after the dataset is manipulated by quantifying the regularity and unpredictability of fluctuations within the data. Many levels can be integrated into a model making some much more complex. Models can include methods used for data cleaning, feature encoding, or even in-corporate a combination of stacked ML and DL methods. This makes it even harder to compare hybrid models to more simplistic ones. Since there is no industry standard for predictive model comparisons and due to the mixed results of superiority claims from different studies, the performance of classical statistical methods should serve as a baseline for evaluating any newly proposed technique. This baseline is necessary to confirm the additional complexity from a newly developed algorithm actually adds predictive capability to a forecast. Shi and Zhuang [40] a random-walk where a single day prediction is equal to yesterday's price as a baseline for comparing the alternative techniques.

2.2 Literature on Input Data

With the goals and focuses of this research in mind, we must investigate the relevant literature for information on the individual criteria and conditions that cause certain components of financial forecasting systems to perform well. It is known that specific factors cause algorithms to fit certain indices or markets better than others but often they are unidentified [24, 36]. Datasets can be

obtained as input data from single or multiple sources. Meng [30] makes a point of claiming that multi-sourced heterogeneous data is advantageous to prediction models. Lawrence [1998] confirms this detail outlining how multivariate models have a better potential to discover additional factors that may affect the trends of data. It is plain and simple: larger training sets equal better forecasts for soft computing methods. Additionally, incorporating additional data such as technical stock analysis indicators can greatly improve predictive accuracies [32]. Models utilizing macroeconomic variables, industry policy, and regulation data have more holistic information on the general status of the economy leading to better performances [24, 5]. Outside the scope of this research, models with higher capacities can find meaning in large online extracted text inputs greatly increasing their predictive capabilities [30, 41]. These are often pulled from online sources using text mining tools to extract information like breaking financial news. Google trends is a resource of this type that has been widely explored as input data to DL predictive models [3, 10, 18, 34].

2.3 Literature on Preprocessing Methods

Within the structure of a hybrid forecasting system, preprocessing methods are implemented before a prediction algorithm is trained to ensure the data is compatible with the specific model. The position of preprocessing methods within prediction models is represented in Figure 2 where a denoising technique is used. Lv and Huang [24] note that the optimal preprocessing technique depends on what aspect of the input data your focus is, and the specific design of a given model. As an additional aspect of preprocessing, it is important that optimal training sample sizes take patterns and volatility of stock data into account [40, 42].

2.3.1 Data Normalization

Normalization techniques used to adjust financial time series have three purposes each focusing on a different component of a time series. They are employed to deseasonalize, detrend, and/or denoise data. Methods can emphasize a single component or take a comprehensive approach to apply all three types of time series normalization.

A basic normalization technique uses a multiplier over an entire set of OHLC data. This includes methods such as multiplying the adjusting ratio [39] or inverse of the maximal stock price [40] over a time series.

$$AdjustingRatio = \frac{Closing}{AdjustedClosing} \tag{1}$$

These are known as denoising methods as they reduce the inherent randomness existing in data.

Denoising is one of the most crucial methods to preprocessing time series. The high fluctuations in data created by noise must be diminished to create an optimal input to a prediction algorithm. There are countless techniques used for this type of data transformation. A simple logarithmic transformation is exercised as a denoising technique by Makridakis [28]. A log function is applied to reduce levels of variance resulting in a normalized time series. When data experiences extreme changes on a seasonal basis, deseasonalization methods must be utilized as preprocessing methods [28]. These normalization techniques focus on partitioning data by regions that experience distinct trends in financial time series. To seasonally adjust input data the seasonal component is removed from the original series decreasing day to day variations in price. Rater than focusing on seasonal patterns, detrending techniques focus on diminishing global trends in

the data. Denoising is most heavily discussed in the literature. As an example of a more complex technique, Leng [22] utilizes a forward-backward filter (FBF) to denoise a time series. The filtration system is comprised of two components: a second-order matrix and a scalar vector. The time series is first passed through the FBF in the forward direction. The order of the FBF is reversed and the partially altered data is passed back through the matrix and scalar vector leaving a fully denoised time series as input to a prediction model. A defining feature of the problem is

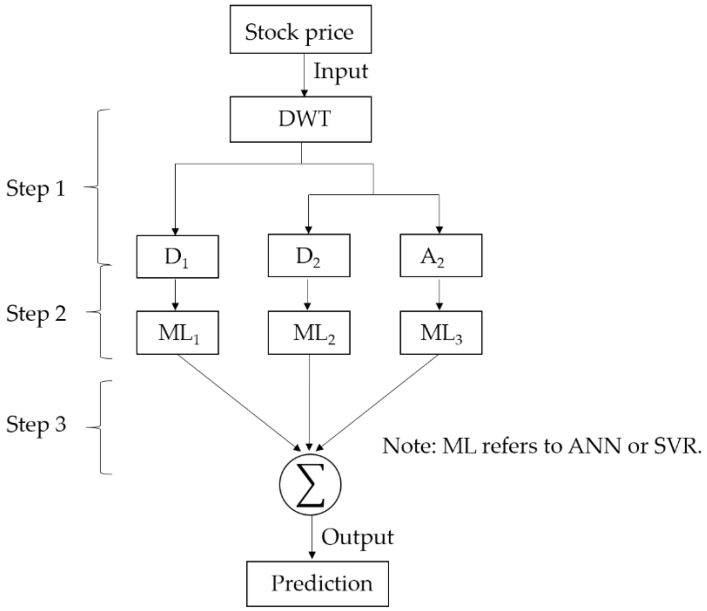

Figure 2: Integration of preprocessing methods and prediction models (DWT-based denoising) [40].

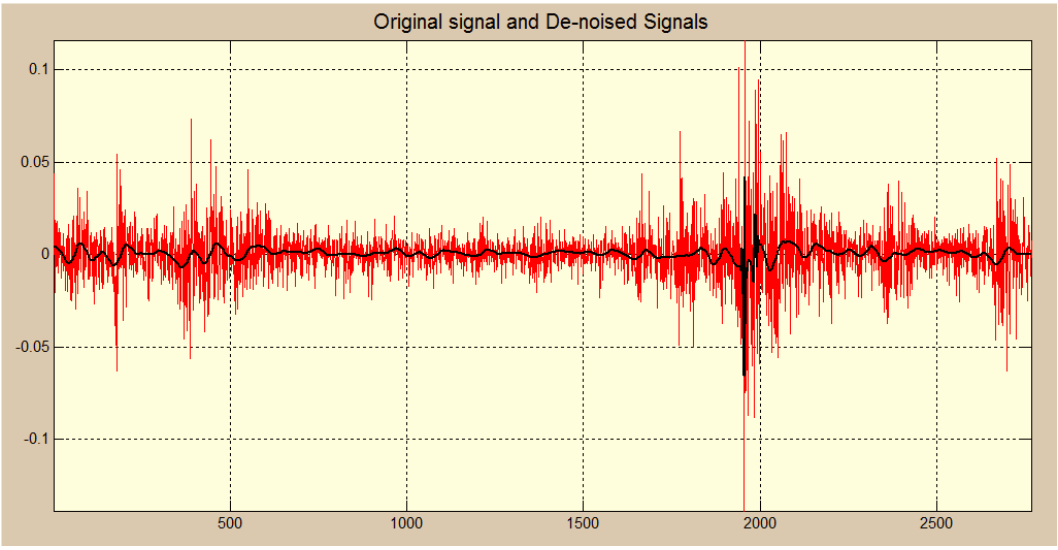

Figure 3: Raw time series (red) vs wavelet (black) [22].

the capacity to discover and characterize long-ranging dependencies within financial time series. For time series corrupted by noise it is impossible to observe these trends. Bao [5] claims that

existing statistical methods are in-sufficient solutions to this problem. Therefore, more complex models are required to denoise time series effectively. Commonly applied to signal normalization, wavelet transformations (WTs), specifically discrete wavelet transformations (DWTs), present a consistent method to denoise financial data [5, 22, 40]. DWTs interact with prediction models as illustrated in Figure 2. They possess the exceptional ability to accurately pin-point time domains and frequency spectrum within big data. A WT decomposes a signal into a sequence of orthogonal projections, each partitioned by a distinct dependency found within the data. Each section is individually denoised by removing unwanted oscillations from the projections [22]. The resulting projections are recomposed into an enhanced noise-filtered time series formatted as input for the predictive model. The filtered version is called a wavelet. Figure 3 displays a raw time series alongside its wavelet illustrating the power of WTs. When combined with a DL framework, WTs are revealed to notably improve the performance of a predictive model. Additionally, WTs have been demonstrated to outperform many denoising techniques tested on financial time series [40]. It is not uncommon to include a combination of data transformations to retrieve optimal inputs for forecasting models.

2.3.2 Additional Feature Extraction

As outlined earlier, feature extraction is a broad sub-category of preprocessing that encompasses further subclasses each with distinct objectives and practices. Often it is unknown whether a feature adds value to a predictive model. Label data may be more effective when reformatted or encoded to have a different structure. With feature selection techniques, the goal is to project an n-dimensional space into a lower dimensional space so that features are independent of each other [8]. Feature extraction is necessary to accurately model complex data from the real world. When applied to time series, feature extraction captures relevant information within a dataset to achieve better performance from a model.

A novel non-linear topological approach using stacked autoencoders (SAEs) is proposed as a reliable method for feature extraction in DL hybrid models [5]. It is used to learn the deep features of a financial time series in an unsupervised manner. Technically, autoencoders are a type of feedforward NN, specifically an MLP. A detailed explanation of MLPs is given in section 2.6.1. In this model, historical stock data, technical indicators, and macroeconomic variables were all used as inputs to the SAEs. Essentially, an SAE is a NN consisting of multiple single layered autoencoders where the output features of each layer is passed on as the input to the next layer of autoencoders. Completed one autoencoder at a time, this minimizes any errors between the inputs and outputs. This mechanism allows SAEs to discover invariant and abstract features concealed in the original input data. According to recent studies, SAEs and similar feature extracting methods utilizing DL techniques lead to better approximations of nonlinear functions than models with more shallow architectures.

Transformation and encoding techniques can be applied to data types other than financial time series. Lv [24] calculates 44 different technical indicators to use as features for his input data. To ensure no metrics are given priority, features are transformed into a more simplistic form. If an indicator is positive, the label value is set to 1, otherwise it is set to 0. This transformation technique allows for models to have an unbiased interpretation of input data. This is known as a discrete representation of the data. When comparing performance of regular technical indicators versus their discrete representation, one study found that the discrete version led to higher prediction accuracies for all tested models [32].

3 Prominent Algorithms & Models in the Literature

3.1 Traditional Statistical Methods

Statistical methods are the simplest and oldest existing methods in the field. Generally speaking, a statistical model is a linear process used to fore-cast univariate time series. Their simplistic nature limits them to taking only a single or few variables into account. As will be demonstrated, sometimes their simplistic nature can be beneficial. As one of the most common statistical methods, an autoregressive (AR) model uses past values to make predictions of future behavior. This is a broad category including all statistical regression techniques. Falling under this category, linear regression (LR) expresses the current value of a financial time series linearly in terms of its past data to return a residual prediction.

Given a set of data points where the y value is price and x value date, linear regression can be thought of as drawing a best fit line through the data. The predicted stock price is wherever the y value of best fit line falls for a given date. Specifically, the best fit line commonly used in linear regression is a least-squares line. To build what can be defined as a "best fit" line, $y = Ax+B$, suitable coefficients values must be found. Values for A and B are calculated by solving a linear system of equations defined by partial derivatives of the sum of squared errors. These errors quantify how far the actual values deviate from their approximated values. Using Gaussian processes, or maybe just back substitution, solving the linear system minimizes the root-mean-square error so that optimal coefficients for the best fit line are found. Similar procedures where systems of equations are solved to minimize approximating errors are used for power curve fitting and least-squares parabolas. Regression can come close to blurring the line between statistical models and machine learning. There are certainly more complex techniques like multivariate regression that experts may consider to fall under the class of ML and not statistics.

There are several hybrid statistical models that are often used as a base standard for comparing the performance of more complex models. Two of these are autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA), also known as the Box-Jenkins method. ARMA is the combination of an autoregressive and simple moving average model that attempts to diminish the failures and better the pros of each method by integrating them into a singular model. The prediction value is modelled as the weighted linear sum of all previous values from the financial time series. ARIMA is the same as ARMA except it incorporates an automated denoising technique to normalize the data and make trend distinction easier. Error, trend, seasonal models (ETS) are another commonly used statistical prediction model. ETS takes advantage of using the additive or multiplicative errors, trends, and seasonal components within a time series. The prediction is made similarly to ARMA and ARIMA where it is modeled as the weighted sum of past data. However, the distinctive factor is that instead of using linear weighting, the model utilizes an exponentially decreasing ratio to designate weights for previous data points. When implemented as an automated trading algorithm like in [28, 27] ARIMA and ETS performed better than all other techniques, including the newer ML algorithms. Somewhat counter-intuitive, from this we can hypothesize that added complexity does not necessarily lead to better performance.

However, statistical methods can have extreme limitations. Their biggest problem comes from the noisiness and non-stationarity of financial time series. Stationarity means that the statistical properties of a time series do not change over time. However, that distinction is not true with financial data. Therefore, for statistical methods to achieve any level of success, a quality denoising technique must be integrated to normalize the data and make it stationary. As described earlier, the two methods that automatically include this process have better performance metrics than

other statistical models. Further illustrating this is a study that tested the denoising techniques of FBF and WT on a regular financial time series as input to an AR model [22]. Fitting results showed that both methods greatly increased the accuracy of their models while decreasing FPE and MSE. Without denoising, the AR model had a terrible fit to estimation accuracy of 0.3754% with an FPE of 0.0002323 and MSE of 0.0002308. Using the FBF technique with the AR model, a fit to estimation accuracy of 99.98% was reached while returning an FPE of 1.557e-12 and MSE of 1.548e-12. Lastly, the WT method gave a fit to estimation data of 99% with an FPE of 6.884e-10 and an MSE of 6.842e-10. Clearly both denoising techniques are highly effective and necessary in order to make certain statistical methods even reasonably employable. Statistical methods are based entirely on quantitative formulas and hard data without considering any external effects that may influence certain sections of a time series. Statistical prediction methods can be highly effective for single day trading but become extremely inaccurate for medium to long-term price forecasts [28]. As simpler techniques, they are unable to reliably pick up on long term dependencies and trends. Statistical methods cannot learn from themselves, so ML algorithms come into play.

3.2 Machine Learning Models

The field of machine learning utilizes non-linear algorithms to forecast data capable of using multivariate time series as input. Based on statistical and probabilistic theory, ML algorithms evolve into better algorithms without needing specific rules intrinsically programmed like statistical methods. As they run, they train themselves to build finalized models provide accurate predictions when generalized to other datasets, at least hopefully [6]. ML algorithms are much more computationally complex than statistical methods. This allows them to make use of more complicated feature sets as inputs potentially leading to better performance. Though, relative to statistical methods, ML algorithms need more prior knowledge in order to get higher accuracies. This means using preprocessing techniques to clean, denoise, and extract features from the data. There are countless ML models that can be applied to financial forecasting. However, it can be difficult to determine which approaches are generally the best due to variability in performance and mixed results from published studies. Each algorithm has its own intricacies that can lead to failures in predictions. Nobody knows the perfect input for a given ML algorithm. This is why there is such a vast range of techniques used on top of ML models to determine the best input [28]. Detailed below are a number of ML models frequently used in financial forecasting.

3.2.1 Naïve Bayes

Naïve Bayes (NB) is a simplistic and extremely common model used for classification in machine learning. In finance, naïve Bayes is used in the implementation of automated trading strategies where it determines buy, sell, and hold points for a given index [24]. Provided a set of classification labels, a naïve Bayesian approach estimates the probabilities for each label by assuming the features are conditionally independent of each other. If the features are not independent, calculated probabilities are likely to be inaccurate or misrepresentative [32]. Thus, feature extraction techniques are needed to ensure the input data is applicable to a naïve Bayes model. Classic Bayesian models have sometimes been shown to be unreliable for financial forecasting comparative to statistical methods. Though, other studies claim that NB can outperform most other machine learning techniques [13]. By altering specific aspects of the algorithm, novel Bayesian classification techniques have been developed claiming better performances with prediction tasks [35].

3.2.2 Support Vector Machine

Support vector machines (SVMs) are one of the most common ML models and appear the most often out of any other algorithm in studies of financial prediction models [4, 8, 9, 13, 24, 28, 32, 40]. SVMs are used to deal with data that is linearly non-separable. There are two main categories that SVMs fall into: support vector classification (SVC) and support vector regression (SVR). Using a mapping function, the idea is to transform the input from its original space into a high-dimensional feature space that is then used to make predictions. The method attempts to find the hyperplane that maximizes the margin between corresponding class labels while minimizing the total error [32]. Denoising and other feature extraction techniques are needed to linearly scale input data for a SVM. Shi and Zhuang [40] found that SVMs were unable to beat statistical methods and a random walk. However, a separate study contradicts that claiming an SVM was indeed able to beat a random walk when implemented as a stock price forecasting system [19].

3.2.3 K-Nearest Neighbor

K-nearest neighbor (KNN) is a nonparametric model which can be used for classification [4, 13] or regression [28]. For both types, provided n inputs, the algorithm finds the k closest points in the training data using a specified distance function such as Euclidean distance. The prediction is based on the calculated distances between data points in the training and testing sets. In KNN classification, the output is a class determined by a voting mechanism comprised of the input's k closest neighbors. A regressive model sets the prediction as the average of the output values for the input's k closest neighbors. The number assigned to k can greatly change the accuracy of the model depending on the provided inputs. Makridakis [28] uses a 10-fold validation process to find the optimal k value.

3.2.4 Decision Tree

A decision tree (DT) is a machine learning model that can easily be used for both classification and regression, though both methods behave similarly. Thus, DTs are commonly called classification and regression trees (CARTs). A DT recursively partitions the input feature space into the structure of a tree dividing a samples into regions called terminal leaves. An input begins at the root node and traverses through the tree. A sequence of tests is completed at each node to decide which node the input should be passed to in the next level. These tests are applied until the input has traveled through the entire tree and has reached a terminal leave where the final prediction is outputted. Compared with other methods, no study few studies found DTs to perform the best. Though, DTs are used by a number of studies to compare the results of other prediction models [13, 24, 28].

3.2.5 Random Forest

Random forests (RFs) fall into the category of ensemble learning algorithms and are used by many studies of machine learning algorithms for financial predictions [4, 8, 20, 24, 32]. RFs operate by constructing a multitude of DTs each providing its own output. As a parameter, the number of trees built in the RF can be wide ranging. For example, Ballings [4] uses 500 DTs in his implementation of RF. The final output is determined by aggregating the individual outputs of each DT. RF models are commonly used to overcome the problem of overfitting when using only

a single DT. When setup as an automated trading strategy, a RF model was found to be wildly ineffective and significantly worse than SVM [8].

3.2.6 Debate over Machine Learning Models in the Literature

Common to the inconclusive theme of this topic, it is common for articles to dispute over whether ML techniques can beat statistical methods and random walks reliably [28, 27, 40]. Several prominent research studies contradict the argument that ML methods are inferior to statistical methods. Huang [19] observed that SVMs were able to beat a random walk. Dash and Dash [13] found that NB, SVM, KNN, and DT consistently beat a MA model when setup as an automated trading system. This led to the inference that ML models perform better than statistical methods. However, no other statistical methods were compared to the ML algorithms, so it remains unknown whether this claim can be generalized. The same study discovered that NB produced superior performance metrics to SVM, KNN, and DT. Contrary to that, other research points to NB being significantly less effective than DT, RF, and SVM [24]. In a separate study conducted by one of the same authors, it was found that SVMs produced the highest prediction accuracies out of the tested ML models [24]. Comparably, when using high prices as input instead of the more commonly used closing price, two different variations of an SVM model consistently produced better metrics than NB [17]. By using 10 technical indicators as input, RF outperformed both SVM and NB [32]. Though, once input data was converted to discrete representation, the models produced similar prediction accuracies at around 90%. Additionally, transforming the input data to its discrete form caused the accuracy of every model to leap by 4% or more. Evidently, preprocessing input data is an essential precursor to effectively applying ML models in financial forecasting.

Ballings [4] ranked RF, SVM, and KNN above the NN based model they were tested against. This indicates that ML algorithms may perform better than DL models when used for financial forecasting. Zhang [44] came to a similar conclusion. However, both studies only compared a single more basic DL model making it possible to dispute their claims. Diving further into the assessment of the two model types, some studies included several more complex NN based algorithms in their comparison of ML and DL methods [28, 27, 40]. Results revealed that the increased complexity in DL models led to better performances than the ML based models. This suggests that further investigation needs to be done on the predictive capabilities of DL algorithms in finance. Though, simple NN models were found to be ineffective [4, 44], perhaps more intricately structured DL models can consistently beat their ML counterparts.

3.3 Deep Learning and Neural Network based Models

Considering the complexity of financial time series, DL has the potential to greatly enhance prediction accuracies with revolutionary developments advancing computing capabilities at an alarming rate. The key advantage of DL architectures is their ability to 'learn' new features from input data [20, 39, 39]. In the application of financial analysis and stock price forecasting, DL models possess the power to discover hidden patterns in the data. One can think of ML techniques as memorizing trends and patterns within data while DL extract new generalizations from input data to represent dependencies throughout the feature space. DL can perform well on time series with higher noise levels due to their unique skill sets leading to accurate price predictions and correct trading decisions. DL models train themselves for prediction tasks by learning through trial and error based on provided inputs. Think of this as a baby learning to decipher its new surroundings. The use of DL frameworks is made possible by the availability of big data. Massive

amounts of training data are needed for DL to be successful at predicting future stock movements [45]. DL has an exciting potential for applications to financial stock price prediction. Though, employing any DL method must be done with careful selection of input variables and network parameters as they are essential to model success [11].

Inspired by biological neural networks such as the human brain, artificial neural networks (ANNs) are the DL techniques being used for intricate prediction tasks [20, 46]. In their simplest form, ANNs have three components: an input, output, and hidden layer each consisting of a set of nodes. NNs are based in the mathematical study of graph theory. Any ANN made up of more than three layers, so having multiple hidden layers, is considered a deep neural network (DNN). For the most part, these are the types of ANNs this paper focuses on. A DNN framework is self-teaching meaning it learns as it processes and filters information through its hidden layers. There are many types of ANN and DNN architectures each with unique characteristics and behaviors. Outlined below are descriptions of model types commonly discussed in the literature.

3.3.1 Feedforward Neural Networks

A feedforward neural network is a broadly defined structure and the simplest class of ANNs. In its most basic form, information moves in only one direction through the network. In terms of graph theory, a feedforward neural network is defined as a simple graph that contains no cycles or loops. The graph behind a feedforward neural network is directed, fully connected, and acyclic. Data is fed to the input nodes where it is then passed through the individual hidden layers until it reaches an output node. Feedforward neural networks have the capacity to classify data that cannot be separated linearly making them useful for complex applications where ML algorithms do not produce satisfactory results.

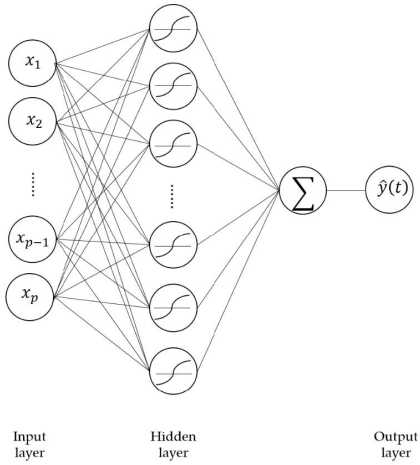

Figure 4: Graphical illustration of a three layered MLP [40].

Multilayered Perceptron

A multilayered perceptron is the simplest class of feedforward neural networks and is the foundation to many other types of ANNs. MLPs are the most commonly applied ANNs to forecasting financial time series since they are simplistic in nature relative to other DNNs [9, 15, 20, 24, 19, 28, 32, 39, 38]. A graphical representation of a simple three layered MLP is shown in Figure 4.

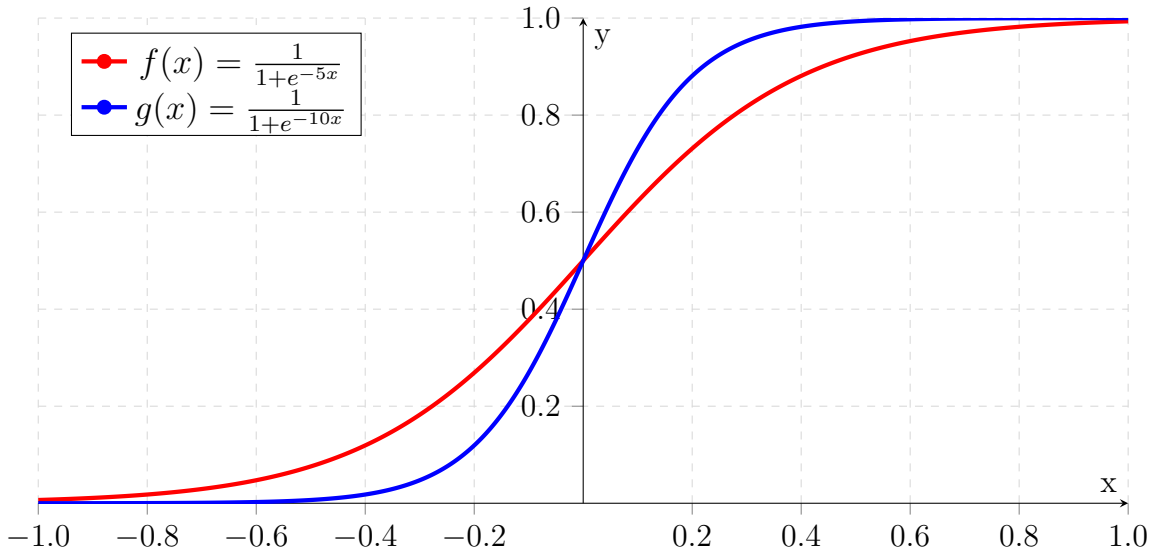

Figure 5: Logistic Sigmoid Function with $\theta^T = 5$ and $\theta^T = 10$

The title MLP is often used interchangeably with the term feedforward neural network. The ideal number of input nodes is usually optimized through a k-fold validation process [28]. For financial time series, the process determines the ideal time period leading up to the current date that an MLP should consider in making a prediction. The activation function is the key component of an MLP. Each node within the hidden layers holds an activation function that maps weighted inputs to the next layer of the network. Sigmoid functions are regularly used in MLP architectures for constructing outputs of individual hidden layers. Graphically, sigmoid functions are represented by an 'S'-shaped curve on the (x, y) coordinate plane. A logistic sigmoid function is illustrated in Figure 5 with $\theta^T = 5$ and $\theta^T = 10$. The sigmoid function provides a node output that is asymptotically bounded between 0 and 1. The θ^T value controls the slope of the activation function at $x = 0$.

$$S(x)=\frac{1}{1+e^{-\theta^T x}} \tag{2}$$

The other key aspect of an MLP is the weight optimization method that is utilized to facilitate training the model. These mechanisms adjust node weights to minimize the cost function, a measure of the error between outputs and their expected values. Backpropagation is the most well-known weight optimization method where the partial derivatives of the cost function are calculated with respect to the current network weights. This process allows for MLPs to work with complex time series. Without the technique, ANNs failed to achieve much success in financial forecasting until it was developed.

Bayesian Neural Network

Bayesian Neural Networks (BNNs) are simpler feedforward neural netowrks with similar structures to MLPs. The main distinction is that their inner structures are based off Bayesian concepts [28]. MLPs can struggle with the problem of vanishing gradient. Tasked with forecasting future stock prices, this specific ANN takes an alternative approach to diminish the issue of vanishing gradient. The goal of any ANN is not only to succeed in training, but to perform well with novel inputs. A neural network that accomplishes this has sufficient generalization. The process of

improving a model's generalization is known as regularization. Different than activation and cost functions, the objective function is the equation that models aim to minimize in order to perform well. A typical objective function is equal to the sum of squared errors:

$$F = E_D = \sum_{i=1}^{n}(t_i - a_i)^2 \tag{3}$$

However, objective functions become more intricate when regularization methods are incorporated. The objective function for the BNN adds complexity to the model by including the sum of squared network weights, E_W, and objective function parameters α and β.

$$F = \beta * E_D + \alpha * E_W \tag{4}$$

The aim is to find the optimal parameters for the objective function as they are highly indicative to the model's success. The regularization process uses Bayesian and Gaussian concepts to maximize the posterior probability in turn minimizing the regularized objective function By assuming some priori distribution of errors, applying additional Bayesian processes, and focusing on where the gradient is equal to zero one inevitably comes across a Hessian Matrix [28]. However, mathematicians want to avoid calculating Hessian matrices. Depending on the complexity of the system, Hessian matrices can be extremely computationally demanding as they involve second-order partial derivatives of nonlinear systems. Foresee [12] optimized the regularization method for BNNs by applying a method to approximate the Hessian matrix. Possibly the most useful and well know algorithm of all time, the vanilla Newton-Raphson method can achieve the rare quadratic convergence for approximating roots. Geometrically the method is simple as it is identical to drawing tangent lines along a curve getting closer and closer to the x-axis until an error tolerance is reached. The revolutionary method can be expanded to k-dimensions for solving nonlinear systems of equations by employing a vector modification of the original method. This results in finding the inverse of the Jacobian matrix. Gradients, Jacobi matrices, and Hessian matrices are naturally connected so with further derivations and utilizing the Gauss-Newton Method an approximation to the Hessian matrix is reached. The model was tested with and without the regularized objective function parameters. It was found that the optimized version had drastically better results. This illustrates how important optimization and parameterization methods are to DL models.

Radial Basis Neural Network

As another form of feedforward neural network, radial basis neural networks (RBNN) are structured similarly to simple MLPs except radial basis functions are used as their activation functions. An RBNN typically has one hidden layer with a specified number of nodes, n. The output of an RBNN is the linear combination of the vectors produced by the n Gaussian based radial basis functions. RBNNs are good for achieving faster convergence rates when deep feature extraction is not needed, and outputs are directly related to the input vectors through the hidden radial basis functions. There is no continuous feature filtering and re-weighting like in a deep MLP due to only having a single hidden layer. Therefore, due to the complex nature of financial time series, RBNNs tend to perform worse than MLPs when applied to financial forecasting problems [28].

Limitations of Feedforward Neural Networks

A traditional feedforward neural network functions off the assumption that all units within an input vector are linearly independent from each other. Due to the complex nature of financial time series, this is a limiting factor when considering the use of feedforward neural networks for predicting future stock movements [5]. Even though a weight optimization method can be used, there is no true concept of memory within a feedforward neural network. Essentially, there is no way to determine how previously trained data may affect certain aspects of a prediction. For example, the correlation between closing prices and a particular financial metric may be of importance to a specific stock index. However, directed acyclic graph structures like in feedforward neural networks are incapable of analyzing these relationships. Consequently, feedforward neural networks cannot make good use of sequential information for financial predictions. Therefore, other more complex types of neural networks are applied to forecasting financial time series in order to minimize this problem.

3.3.2 Recurrent Neural Networks

A recurrent neural network (RNN) is a type of DNN architecture that attempts to build off the failures of a regular MLP due to having no concept of memory [24, 19, 26, 28, 38]. An RNN is similar to an MLP but includes feedback connections between hidden layers so that previous outputs can be considered along with current values in order to produce the final output. The hidden layers are comprised of what are called recurrent nodes. Essentially, a copy of every previous node's output is saved and used as additional input when passed to the next layer. RNNs are good at modelling time series, but bad at learning long-term dependencies due to the vanishing gradient problem. RNN architectures have a backward dependence over time. Consequently, RNNs become increasingly complex as the learning period increases in size. Therefore, even though the main focus of an RNN is to learn the long-term dependencies within provided data, it becomes notably difficult to train RNNs when information is stored for especially long periods of time.

Long Short-Term Memory

Long short-term memory (LSTM) is a frequently used variation of the RNN architecture that attempts to fix the problem of vanishing gradient [5, 16, 20, 23, 19, 24, 28, 38]. Feedback links are attached to hidden layers in the network to retain time related information for longer periods. Essentially, the difference between LSTMs and classic RNNs is that the recurrent nodes have a better capacity to store information from past nodes over longer periods of time. Therefore, the functions within the hidden nodes are much more complex in LSTM architectures. LSTMs have been proven to be better than conventional RNNs when it is appropriate for models to be utilizing data from long periods of time. The question arises whether the ability to store more past information truly aids the model when forecasting stock movements. Liu [23] found LSTMs to have higher accuracies when financial time series were not denoised beforehand. Though, in a comprehensive study, Makridakis [28] concluded that LSTMs were close to the bottom in predictive accuracy when compared to other DL techniques for stock price prediction. When implemented as an automated trading system, Kamalov [20] found that LSTMs were able to produce higher daily returns than any other type of model.

3.3.3 Convolutional Neural Network

A convolutional neural network (CNN) is a DNN framework that builds on top of the traditional MLP architecture. CNNs are commonly used for image processing-based classification problems. In addition to the regular hidden layers of an MLP, a CNN has multiple convolutional layers based on convolutional operations. The outputs of the convolutional layers are calculated by a sliding filter across the input array that takes the dot product with the corresponding section of the array [20]. This process takes advantage of any existing structure within the input to refine the data as it is passed through the layers. This is good for exploiting underlying patterns. One advantage of CNNs is the number of parameters they have compared to conventional deep MLPs due to having the additional convolutional layers [38]. Optimization algorithms are commonly used alongside CNNs to determine the optimal hyperparameters for the model. Sezer [39] notes that CNNs have higher potential for classification problems rather than regression ones. However, Kamalov [20] implemented a CNN as an automated trading system to find that the daily returns were insignificant compared to other models like LSTM and MLP.

3.3.4 Granular Neural Network

Granular Neural Networks (GNNs) are another type of ANN based on the theory of granular computing (GrC) [14]. GrC is an emerging field that focuses on information processing and the complexity of data at the lowest level. Computational reasoning is a central component of GrC. Granular data can be broken down and grouped together based on things like similarity, functionality, adjacency, etc. to create granules. The concept is used to deal with massive amounts of noisy, incomplete, or uncertain datasets. Regular NNs can struggle to analyze highly dimensional, complex, and large datasets leading to terrible time complexities. The combination of GrC and NNs to create GNNs has the potential to solve these problems [14].

A GNN functions by partitioning a granular dataset into disjoint subsets, known as granules. From these groups, the GNN uses computational reasoning to create 'fuzzy if-then rules' which classify the granules into categories. These rules develop the fully trained model which can then be applied to new inputs to generate new prediction values. Zhang [45] implements a GNN by using three inputs of granular data from a financial time series: open, high, and low prices. These inputs are used to build 'fuzzy' rules which predict close prices for each day. To test the GNN's success, he compared the model to a basic MLP using backpropagation as the weight optimization method. The GNN resulted in an average error of 1.39 whereas the MLP gave an average error of 3.38. The results indicate that GNNs are superior to that of a classic ANN.

3.3.5 Debate over Deep Learning Models in the Literature

Similar to ML, there are mixed results when it comes to the success of certain DL techniques used for financial predictions. First of all, it must be noted that ANNs require the careful selection of network parameters in order for them to succeed [11]. This is major contributor to a difference in results from separate studies. One study may use a certain set of parameters leading to wildly different accuracies than studies using another.

In an analysis of the M-3 Competition in 2000, Makridakis [27] found that no DL based model was able to beat the best statistical methods for stock price prediction. Later in 2018, Makridakis completed an-other comprehensive study of commonly used methods for financial time series forecasting. He again concluded that DL techniques were unable to beat the five statistical

methods he tested in the study [28]. Oppositely, Shi [40] found that an MLP framework using back-propagation was indeed able to outperform the statistical method it was tested against. Though, the method was still unable to beat a random walk. Unlike Makridakis, Shi used only a single statistical method in his comparisons. Therefore, it remains unknown whether his implementation would outperform the other statistical methods Makridakis included in his study. Interestingly, Shi also found that adding a denoising technique to the preprocessing stage improved the prediction accuracy of some stock indices but not others. If an index had high fluctuations, a wavelet transform was able to greatly improve performance metrics. However, if fluctuations were low, the denoising technique worsened the overall success of the model. As mentioned earlier, Liu [23] determined that LSTMs performed better without denoising the data beforehand. These findings imply that denoising techniques should be applied on a per case basis depending on what stock index is being analyzed and which model is being implemented.

In a review of the literature, DL techniques are found to have mixed results when compared to their ML counterparts. Lv and Huang [19] concluded that no DL method was able to consistently beat ML algorithms. From their results, they determined that DL based models were much less consistent than ML based models. They either performed exceptionally well or extremely poor. For example, an MLP per-formed the best with a few stock indices but produced inadequate results with the rest. This suggests that certain hidden factors within a time series lead to the success or failure of a given DL model. Similarly, Chong [11] found that MLPs were rarely able to beat an autoregressive model. Patel [32] also found that an MLP performed worse than the three ML algorithms it was tested against, two of which were RF and NB. Contrary to this, Lv and Huang [19] claimed that an MLP was able to outperform DT, RF, NB, and SVM in all industries except for energy where the models were almost equivalent. Duan [15] found that a deep MLP with three hidden layers performed better than SVM and DT. In another study, LSTM was able to consistently beat RF and a logistic regression model when setup as a trading strategy [16]. Similarly, Nikou [31] found that LSTM produced higher accuracies than both SVM and RF.

Adding to the mixed results, there is no universal conclusion on which DL models perform the best relative to each other. Duan [15] tested multiple versions of an MLP against a number of ML techniques. He determined that DNNs produced the best metrics. Specifically, a deep MLP with three hidden layers was much better than a typical simplistic MLP with a single hidden layer. This implies that DNNs are more suitable for financial forecasting than shallow three layered ANNs. As a common pattern in the literature, RNNs were found to only succeed when specific conditions exist within a financial time series [26]. This was also found to be true with most other DL frameworks. Therefore, it is important to pick evaluation criteria, input data, NN parameters, and NN structures carefully [46]. Patel [32] tested DL models with technical indicators in their regular form and in a discrete representation. He found that converting the input to a discrete form increased model accuracies by over 10%. Evidently, preprocessing is an essential component to the implementation of DL techniques. Multiple studies found LSTM to outperform CNN, RNN, and MLP models [20, 21, 31]. These studies determined that CNNs and MLPs had varying success. They did well with some indices but poor with others. Fischer [16] claimed that LSTM could consistently beat a deep MLP. Contrary to the findings of these studies, Lv and Yuan [24] found that an MLP outperformed RNN and LSTM by a significant amount when using 44 technical indicators as input. In his comprehensive study, Makridakis [28] determined that LSTMs were one of the worst models for stock price prediction. Clearly, the superiority of different DL models for financial forecasting does not have a universal agreement among experts. By no means are NNs a final solution to time series forecasting. However, they do seem to improve forecasting especial among time series with high levels of noise [46]. DL is not yet a concrete solution to the forecasting

of financial markets, but the future looks bright as they have increasingly become more effective as more research has been completed.

3.4 Noteworthy Hybrid Models

Hybrid models consist of combinations of different input data types, preprocessing methods, and mostly soft computing techniques that are intentionally chosen to absorb the benefits of each individual component. The aim is to build a single dynamic system that functions better than a predictive model on its own. A number of hybrid systems in the literature have produced intriguing results indicating they have an advantage over basic models. Novel hybrid prediction models developed within the last decade have some of the best performance metrics out of proposed models. When compared to their more simplistic versions they often perform considerably better and are more consistent. Hybrid models are the future of predicting financial markets since they take into account different variables that simplistic models are unable to account for [19].

3.4.1 WSAEs-LSTM = WT + SAEs + LSTM

In 2017 Bao [5] developed a novel trading strategy that combines a WT, SAEs, and an LSTM into a single model. The model preprocesses a time series with a WT to denoise the data. The cleaned OHLC data, technical indicators, and macroeconomic variables are passed to a system of stacked autoencoders (SAEs) to pull more information from the data and form the final input to the LSTM. The novel version was tested against several variations of the algorithm to determine the true success of the model: WLSTM (WT + LSTM), LSTM, and RNN. LSTM and RNN were found to have larger variations in the predicted values from the actual values than WSAEs-LSTM and WLSTM. WSAEs-LSTM had less volatility than WLSTM and was closer to the actual data. Specifically, WSAEs-LSTM presented a unique advantage in markets that were less developed. A profitability test was done on the different models to give the following average percent earnings per year:

1. *WSAEs-LSTM: 64%*
2. *WLSTM: 38%*
3. *LSTM: 18%*
4. *RNN: 12%*
5. *Traditional Buy-and-Hold: 5%*

The novel system developed by Bao generated by far the most profit out of the tested models. The findings of this study indicate that complexity does indeed add to the predictive capabilities of financial prediction models. The addition of each method starting from the base model, RNN, resulted to an increase in performance each time. The incorporating SAEs in the prediction system caused the largest increase in profitability out of each component. Therefore, it can be implied that SAEs are a decent method to include in hybrid models for feature extraction.

3.4.2 Xuanwu

Xuanwu is a model developed by Zhang [44] that can predict both stock price movements and the interval of growth, or decline, rate within the specified prediction intervals. The goal of the model is for there to be no human intervention: by the click of one button the program will run and tell the user all he/she wants to know. The model is based in the principal, "let the data speak for itself".

The first component of the model is an unsupervised pattern recognition algorithm used to generate training samples from a raw financial time series comprised of OHLC data. A sliding window method is utilized to section the time series into pieces, called clips. The length of these sections is determined by predefined fixed prediction durations (PD). Also predefined is the training duration (MD) which outlines how many days to include in the training sample. The sliding window moves day by day analyzing the shape of the time series deciding when to initialize and end a given clip based on patterns discovered within the data. The clips are classified into four categories: up, down, flat, and unknown based on the discovered trends. The clips classified as up or down are further classified into the subclasses continuous and sideways that reflect the extent of their growth, or decline, with respect to closing price and return rates. The method generates enough clips to cover every situation that may be encountered. For example, in the study's experimental setup, 70 different learning models were created to account for every possibility. The clips represent the existing patterns in the data that the prediction model should be interested in.

The feature set of a given training sample can be extended by adding extra technical indices to the input data. 74 different technical indices are considered. However, without filtering the indices, a highly dimensional feature space is created. Additionally, we do not know whether additional information truly has a positive impact on the prediction model. Thus, a forward sequential search method is used for fast feature selection to narrow down the feature space.

The prediction model is built by a RF classifier that initializes a multitude of DTs at runtime. Each DT partitions the features through its internal nodes to build the classification model. RF is used to overcome the over-fitting problem that exists when using a single DT. The output of each DT is a vector consisting of the probabilities that a clip belongs to each class. The final class label of a given clip is determined by a plurality voting mechanism. However, the number of clips classified under each category tends to be unequal. Therefore, this becomes a problem of imbalanced learning. The model embeds an undersampling technique into the RF classifier to balance the number of samples present within each class. Classification accuracies for each predefined PD-MD pair are displayed in Table 1. Lastly, a trading strategy is implemented on top of Xuanwu using the predicted classifications. The return per trade for each predetermined period is shown in Table 2. Xuanwu is compared to a number of models to tests its relative success.

Table 1: Model comparison by accuracy (percent) [44].

PD-MD	SVM	MLP	KNN	Xuanwu
10-6	58.9±3.6	55.2±9.3	43.9±6.6	**65.2±2.9**
15-10	60.1±4.6	55.9±10.5	41.4±5.4	**67.2±3.1**
20-13	61.5±3.6	57.0±8.1	42.1±6.4	**69.9±2.8**
30-20	59.6±3.7	52.6±10.0	40.8±5.1	**74.2±2.6**
40-26	60.4±5.1	54.4±9.5	41.4±5.2	**75.1±3.7**
50-33	60.9±5.3	55.4±8.8	41.8±5.4	**72.2±2.8**

Table 2: Model comparison by return per trade (percent) [44].

PD-MD	SVM	MLP	KNN	Xuanwu
10-6	4.23±0.68	3.43±1.45	1.78±1.14	**4.58±1.45**
15-10	5.34±0.99	4.52±2.01	2.12±1.52	**6.02±1.73**
20-13	6.41±1.18	5.43±2.28	2.36±1.89	**7.25±2.28**
30-20	7.72±1.68	6.02±3.35	2.94±2.46	**9.65±2.64**

These are SVM, MLP, and KNN. It was found that Xuanwu performed the best. This can be seen in Table 1 and Table 2 above. Xuanwu produced the highest accuracy and return per trade out of the tested models for every PD-MD pair. Only SVM came relatively close to producing the same metrics as Xuanwu. It would be interesting to see if this advantage persists when compared to additional models.

The model was first trained and tested with data that only included OHLC values. This gave an average prediction accuracy of 67.5%. It was found that a medium-term prediction resulted in the highest accuracy at 70.8%. Once extra features such as volume, transaction data, and technical indicators were added to the feature space prediction accuracies increased by an average of 3%. These are the values displayed in Table 1 and Table 2. Medium range predictions were able to reach up to 75.1% accuracy when including the extra features. The addition of the extra features gave the model more knowledge in order to successfully extract the most informative features and better recognize patterns within the data. This further illustrates the importance of preprocessing methods for prediction models. The study designates the success of Xuanwu to the comprehensive integration of special feature selection, pattern imbalance learning, and random forest variations all into a single uniform process.

3.4.3 GA + MLP

In 2017 Sezer [39] introduces a novel trading approach that combines evolutionary optimized technical analysis indicators and a NN based stock trading model into a single system. A genetic algorithm (GA) is used to optimize the feature selection process to provide the ideal input to the prediction model. The NN based model used in the system is a deep MLP with seven layers.

The novel trading system consists of six phases. First, a time series with OHLC data is downloaded and normalized by the adjusting ratio. In the second phase, a collection of RSI and MA values are calculated for a number of different intervals within the time series. RSI is used to format the buy-sell point predictions while MA values are utilized to determine the trend directions of the specified intervals.

Now the GA phase of the trading system takes place. The GA computationally represents Darwin's theory of natural selection by selecting the best fit parameter combinations that lead to the highest performance levels. Initially, 50 unique chromosomes are prepared to the inspected by the GA. Each chromosome is randomly populated with 8 genes. The chromosome is split into two classes: down trend and up trend. MA values are used to find pairs of RSI buy values, RSI buy intervals, RSI sell values, and RSI sell intervals for each of the two classes subsequently filling the chromosomes. Then, the fitness value, or profitability index, is calculated for each of the 50 chromosomes using the training data. The fittest chromosome is selected for the first generation and added to the genetic pool. The GA functions, crossover and mutation are applied on the remaining population slightly altering each chromosome. This process is repeated several times

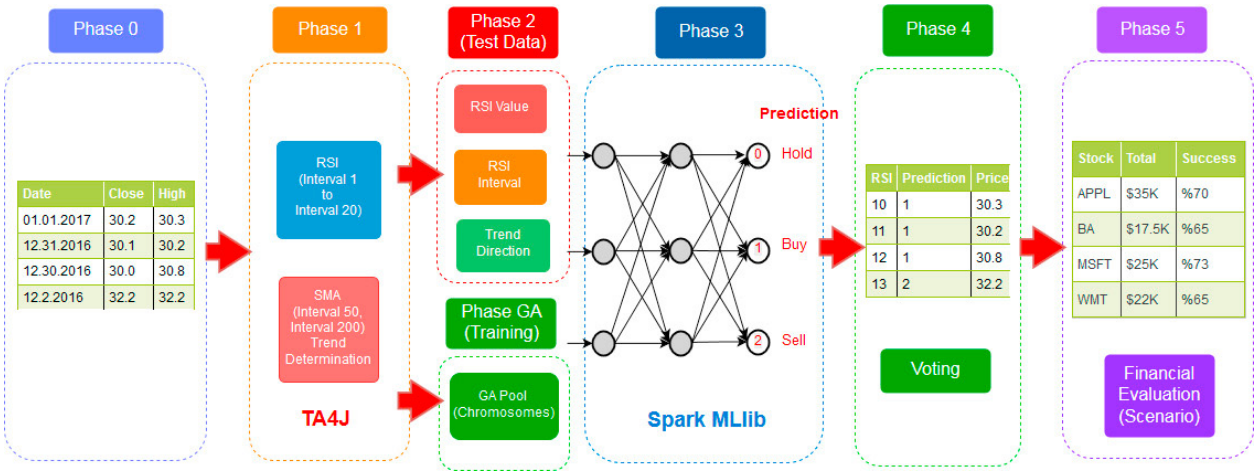

Figure 6: GA + MLP model architecture [39]

where the fittest chromosome from each generation is added to the genetic pool. To end the GA phase, chromosomes in the genetic pool are added to the training data with buy-sell and trend direction labels for each RSI value-interval gene pair. Two hold labels are added to the training data with representative in-between values to build the final input for the MLP.

In the fourth phase of the model, the optimized features from the GA are fed into the MLP which outputs a list of prediction points where the time series is labelled sell, buy, or hold. This list includes every possible RSI interval prediction for the test dataset. In phase five, all RSI interval predictions are reduced to one value using a voting mechanism. The number of predictions under each of the three labels are counted. If a label exceeds a count of 14, it is used as an actual prediction for the trading strategy. Finally, the trade signals are assessed by a financial evaluation method that calculates the total capital, annual return rate, number of transactions, profit per transaction, percent of success, and other metrics outlining the performance of the model. The architecture of the proposed model is illustrated in Figure 6.

The GA + MLP model was tested against several variations including a stand-alone GA model, MLP, and buy-hold strategy. Each was given an initial starting capital of $10,000. The proposed GA+MLP model gave an average annual return of 11.93% with a percent of success of 71.63%. The GA method had an average annual return of 15.83% and a success rate of 70.88%. The MLP resulted in a 10.3% annual return and success ratio of 71.63%. Lastly, the average annualized return for the standard buy-hold strategy was 13.83%. GA+MLP performed better than the buy-hold strategy for 12 out of 29 stocks that were tested. The GA method performed better than the buy-hold strategy in 16 out of 29 stocks. The results indicate that the GA+MLP strategy produces similar and sometimes better metrics when compared to buy-hold. However, the GA method resulted in the highest average annual return rate. Thus, it becomes evident that using GAs for feature extraction and selection is advantageous. For future work, Sezer intends to utilize more technical parameters and test the GA method with a CNN or other DNN. Perhaps a GA for feature extraction is better suited for another NN based model rather than an MLP. Additionally, further research should be done to find what conditions within a time series lead the GA+MLP to better successes.

4 Experimental Design

In this empirical analysis, the aim is to discover which hybrid forecasting models perform the best at predicting market prices. The three main aspects of a model have been outlined: input data, preprocessing methods, and forecasting algorithms. As discussed in the literature, the success of certain models is highly dependent on conditions that exist within time series and other input variables. Testing combinations of system components with a standard evaluation criteria provides easy analysis of the selected models' performances. This procedure evaluates the conditions that lead to a given model's success. A combinatorial approach is proposed to find the optimal hybrid systems for forecasting market prices. Note, day closing price is the exact target value for each model. Models are tested for their performance at one-step forecasting, or predicting tomorrows closing price. This study includes methods coded by hand as well as imported from outside libraries.

4.1 Input Types and Data Collection

4.1.1 Financial Time Series

Three types of input data are examined : financial time series OHLC data, technical analysis indicators, and macroeconomic variables. Yahoo Finance [2]is an online platform used by investment professionals and non-professional investors to access accurate and updated financial and economic records. This platform is used to retrieve basic financial time series data in the form of OHLC prices and volume. To be clear, when OHLC is referred to as input, daily recorded volume is included in the feature space. Data from Yahoo Finance can be accessed through Python application programming interfaces (APIs) allowing data to be extracted from Yahoo finance's web domain. Research indicates that exchange-traded funds (ETFs) are generally sufficient for evaluating stock market forecasting systems [19, 23, 32, 42, 38]. Four futures are examined to give a general idea of how models perform against different markets. Each future represents a distinct weighted collection of stock indices. These are the SPDR S&P 500 ETF Trust (SPY), Invesco QQQ Trust (QQQ), iShares Russell 2000 ETF (QQQ), and SPDR Dow Jones Industrial Average ETF (DIA). SPY consists of large-cap companies from the S&P 500 Index. QQQ represents big tech where its largest holdings are in AAPL, MSFT, FB, and TSLA. IWM tracks a widely diversified set of small-cap companies. DIA has holdings in 30 large-cap companies, weighted more heavily than SPY in sectors like industrials, healthcare, and financials. OHLC time series data for each ETF is the base input to prediction models.

4.1.2 Technical Analysis Indicators

As highlighted previously, technical stock analysis indicators have the potential to improve predictions considerably [32, 24]. Technical indicators are calculated with OHLC data imported from yahoo finance. A total of 80 technical indicators are included in the advanced dataset. The added features fall into five main categories: momentum, return, trend, volatility, and volume-based technical analysis indicators. Providing up to 80 additional features gives algorithms a chance to pick up on a wide variety of patterns hidden discretely within basic OHLC input. Detailed below are some of the indicators included in the advanced feature space.

Moving Average Convergence Divergence (MACD) A moving average (MA) is a technical analysis tool used to smooth out data by taking a rolling average of the values over a set period of time. Short-run and long-run MAs are commonly used in trading strategies to determine buy-sell signals where the two trends intersect. A simple moving average (SMA) is solely the arithmetic mean of prices over the specified period length. An n-day SMA leading up to time t:

$$SMA_t(n) = \frac{P_t + P_{t-1} + ... + P_{t-(n-1)}}{n} \tag{5}$$

$$= \frac{1}{n} \sum_{i=0}^{n-1} P_{t-i} \tag{6}$$

Taking the tool a step further, an exponential moving average (EMA) is a moving average that places greater emphasis on the values closer to t by weighting them more heavily. EMAs are used similarly to SMA in trading systems. A recursive mechanism is implemented to weight the closing prices and form the EMA function.
n-day EMA leading up to time t with smoothing coefficient β:

$$EMA_t(n) = \begin{cases} P_1, & \text{t} = 1 \\ \beta P_t + (1-\beta) EMA_{t-1}(n), & \text{t} > 1 \end{cases} \tag{7}$$

where the smoothing coefficient β is often

$$\beta = \frac{2}{n+1} \tag{8}$$

Moving average convergence divergence (MACD) is a technical analysis indicator that reveals the momentum of the relationship between two MAs. The two MAs must be computed for the entire dataset before the MACD values can be found. We are going to use two EMAs with a short-term period of 9 days and long-term period of 24 days. The MACD at time t is simply found by computing the difference between the long-term and short-term EMAs. Provided the sets $EMA_t(9)$ and $EMA_t(24)$, the MACD at time t is given by:

$$MACD_t = EMA_t(9) - EMA_t(24) \tag{9}$$

Rate of Change (ROC): Momentum is the speed or velocity at which stock prices change over a specified period of time within a financial time series. Momentum in its most simplest form is a measurement calculating the difference between current price and a close price nn trading days prior. Momentum indicators, also known as momentum oscillators, use additional means to examine how prices change over a given period of time. This outlines the severity of uptrends and downtrends within a time series. Rate of change is the first momentum oscillator examined. Rate of change builds off simple momentum calculations by measuring the percent change in price over the same time period. The indicator is mostly used by investors to alert them when rising or falling divergences or price reversals may be near.
MTM over nn days at time tt:

$$MTM_t(n) = P_t - P_{t-n} MTM_t(n) = P_t - P_{t-n} \tag{10}$$

ROC over an nn-day MTM period at time tt:

$$ROC_t(n) = \frac{MTM_t(n)}{P_{t-n}} * 100 \tag{11}$$

$$= \frac{P_t - P_{t-n}}{P_{t-n}} * 100 \tag{12}$$

Relative Strength Index (RSI): Relative strength index is an alternative momentum indicator used to measure recent prices changes within a financial time series. It has the same aim as ROC but achieves it in a slightly different way. Relative strength index produces values between 0 and 100 to indicate where a stock is considered either overbought (RSI>70) or oversold (RSI<30). The computed value compares the losses and gains of price values within a time period. These are incorporated by calculating the average price gain and average price loss over the time period. Though, there is no universal standard for computing the two values. Sources use a number of techniques to obtain them resulting in slightly different values though each with the same goal. This version of relative strength index calculates the MTMs of each rise and fall in closing price to compute their respective SMAs.
A general notion of the method is given by,

$$RSI = 100 - \frac{100}{1+RS} \tag{13}$$

with relative strength (RS) measuring the ratio of average gains to average losses:

$$RS = \frac{\text{Average Gain}}{\text{Average Loss}} \tag{14}$$

A detailed process to find the RSI over n days at time t is given below:
For each increment in t, upward changes U_t and downward changes D_t are calculated,

$$U_t = \begin{cases} P_t - P_{t-1}, & P_t > P_{t-1} \\ 0, & P_t \leq P_{t-1} \end{cases} \tag{15}$$

and

$$D_t = \begin{cases} 0, & P_t \geq P_{t-1} \\ P_{t-1} - P_t, & P_t < P_{t-1} \end{cases} \tag{16}$$

By using SMAs over the same period n for U and D respectively we get,

$$SMA_t(U,n) = \frac{1}{n}\sum_{i=0}^{n-1} U_{t-i} \tag{17}$$

and

$$SMA_t(D,n) = \frac{1}{n}\sum_{i=0}^{n-1} D_{t-i} \tag{18}$$

Now the RS over the last n days is,

$$RS = \frac{SMA_t(U,n)}{SMA_t(D,n)} \tag{19}$$

to give the final equation:

$$RSI = 100 - \frac{100}{1+RS} = 100 - \left[\frac{100}{1+\frac{SMA_t(U,n)}{SMA_t(D,n)}}\right] \tag{20}$$

On-Balance Volume (OBV): On-balance volume is an additional momentum oscillator which aims to measure closing price fluctuations based on changes in volume rather than price. Often included in OHLC data, volume, or trading volume, is the number of stock shares traded during a given period of time. On-balance volume is a cumulative measure of existing buy and sell pressures that sums all positive and negative volume values within a given time period.
OBV over n days with volume V_t at time t:

$$OBV_t = \begin{cases} V_1, & t = 1 \\ OBV_{t-1} + \begin{cases} V_t, & P_t > P_{t-1} \\ 0, & P_t = P_{t-1} \\ -V_t, & P_t < P_{t-1} \end{cases}, & 1 < t \leq n \end{cases} \tag{21}$$

There is a piecewise function inside a recursive equation making the formula for on-balance volume look more complicated than it truly is. Starting at the first date of the chosen time period we set the OBV to the volume. For the next day, if the close price is greater than the previous day then we add the next days volume to the OBV. Otherwise, we subtract the next days volume. This is repeated until we reach the current date.

4.1.3 Macroeconomic Variables

The last input explored in this research proposal is less studied than the previous two but still presents a potential benefit to financial forecasting systems. Macroeconomic variables is used to provide models with a global view of the current economic status. The economy as a whole can have massive impacts on specific stock indices so including them in input data has the potential to increase performances. It is theorized that DL methods benefit the most from the additional inputs due to the unique ability to extract new features and dependencies within data. The goal is to determine whether the addition of these variables results in better evaluation metrics.

Accurate and up to date macroeconomic data is obtained from the Federal Reserve. Federal Reserve Economic Data (FRED) [1] is a public database maintained by the Federal Reserve Bank of St. Louis with more than 750,000 economic time series from over 90 sources. The data is viewed online as well as downloaded for importing to a local database. The API "fred_api" [29] is used to import time series from the government website domain. API keys are obtained through the Federal Reserve Bank of St. Louis to enable free access to their databases. My working API key from the FRED allows me to set up a working data importation system for macroeconomic variables. This method is similar to the process described at the beginning of this section for OHLC data. However, some FRED data must be cleaned before it can be used. As an example, GDP numbers are released by the FED on a quarterly basis meaning time series values only occur once every three months. A simple preprocessing method must be implemented to fill in values for trading days in between listings in the original import set. Listed below are the three macroeconomic variables this study will examine:

1. **Federal Funds Rate (FEDFUNDS):** Set by the Federal Open Market Committee (FOMC) of the Federal Reserve (FED), the effective federal funds rate is the interest rate at which commercial banks and credit unions borrow and lend their excess reserve balances overnight. The federal funds rate influences short-term rates on consumer loans and credit card balances and has a significant impact on the stock market.

2. **U.S. Dollar Index (DXY):** The U.S. Dollar Index is a measure of the value of the United States dollar relative to a specific set of foreign currencies. The value is a weighted geometric mean of the U.S. dollar's value relative to the EUR, JPY, GBP, CAD, SEK, and CHF. The DXY raises when the U.S. dollar gains strength, or value, when compared to the value of other worldwide currencies.

3. **Consumer Price Index (CPI):** Consumer price index is used as a measure of inflation. Inflation measures the rate at which the average price level of a basket of specifically selected goods and services in the U.S. economy increases in value over a given period of time. Changes in the CPI are used to assess price changes associated with the cost of living and can have significant effects on the stock market.

4.2 Preprocessing Methods

Data must be processed prior to using as input to ML and DL forecasting algorithms. As alluded to in the introduction, the different processes and purposes of these methods can vary immensely. Several types of preprocessing methods are explored to test their effectiveness. Research indicates that some initial preprocessing is needed depending on the components implemented in a system. Two techniques are tested as initial preprocessors for each model. The resulting performance boost varies depending on the model. The first is enforcing stationarity in input data by removing trends. The Dickey-Fuller test is utilized to check for stationarity in data on a per feature basis. When the test fails, a feature is differenced by a single lag variable to achieve stationarity in the mean. When used the technique is is done on the target variable, tomorrows closing price. After each forecast data is inversely differenced to achieve the predicted closing price. Without enforcing stationarity some models produce significantly worse results while others have varying results. This is highlighted in the results section. Note, that when trend is removed from the target, models are essentially now predicting the forecasted changes in price one-step ahead rather than the true price. Second, features are scaled by their maximum absolute value to remove the significance of outliers in data. This step is performed after initial scaling, standardization, and normalization to force all data inputs to a more similar range. This technique scales data to the range $[-1, 1]$ with only the absolute maximum value keeping data in its original distribution. Note that this technique is needed for certain algorithms which need inputs to be within $[-1, 1]$ such as SVR. This step results in increasing model performance for almost every tested model combination.

4.2.1 Scaling, Standardization, and Normalization

Standard Scaler A typical method for standardization uses the mean and standard deviation to achieve the adjusted output. The method is applied on a per feature basis enforce zero-mean and unit variance. The mean of the data is differenced and then divided by the features standard deviation. Usually, around 68% of the values will lie within the range of $[-1, 1]$ after the method is applied. Though, that range is obtained after data is scaled by its maximum absolute value immediately after. With mean value $\bar{x}$ and standard deviation σ, the adjusted values are obtained by $X_t = \frac{x-\bar{x}}{\sigma}$.

Minimum Maximum Scaler One of the most popular scaling techniques uses the maximum value, α, and minimum value, β, within a given domain to remap data values. This process

rescales the original range of data to the range of $[0, 1]$. The method was used by Lv and Yuan in 2019 to normalize a set of 44 technical indicators [24] and is commonly used within machine learning and data science since it maintains the distribution structures within the original data while rescaling it. Adjusted values are attained by subtracting the the minimum value and dividing by the maximum minus the minimum. Where α is the maximum and β the minimum, the adjusted values over an n day period at time t is obtained by $X_t = \frac{x_t - \beta}{\alpha - \beta}$.

Yeo-Johnson Power Transformation A power transform is a family of functions applied piecewise to datasets forcing them to a monotone structure. Power transformations are meant to remove variances over time while manipulating data to have a more Gaussian-like distribution. The Box-Cox transformation is the most popular technique within this family. However, it cannot be applied to data with negative values. The Yeo-Johnson transformation is used as an alternative where both negative and zero values can be handled. Enforcing positivity to the data and applying the Box-Cox was tested and found to produce worse results than the plain Yeo-Johnson transformation. A parameter, $\lambda \in \mathbb{R}$, is taken as input to the function where it is best optimized through maximum likelihood, consequently minimizing skewness and stabilizing variance [43]. The version of the algorithm used in this study was slightly modified from its traditional implementation to test additional parameter values during optimization so that $\lambda \in [-5, 5]$. The optimization method provides the λ that produces the lowest standard deviation over the entire dataset. Power transformations are used by Makridakis in his study from 2018 [28] and similar log transformations used in others [5]. The piecewise function for the transformed feature values X_t at time t is given by

$$X_t \begin{cases} ((x_t + 1)^\lambda - 1)/\lambda, & \lambda \neq 0, x_t \geq 0 \\ \log(x_t + 1), & \lambda = 0, x_t \geq 0 \\ -[(-x_t + 1)^{2-\lambda} - 1]/(2 - \lambda), & \lambda \neq, x_t < 0 \\ -\log(-x_t + 1), & \lambda = 2, y < 0 \end{cases} \tag{22}$$

4.2.2 Feature Selection Methods

Feature selection reduces the dimensionality of a given feature space to a subset of the original space. High dimensionality within input data can easily cause runtimes to be inefficient. One might assume, with more features the better performance becomes. This is an incorrect assumption since data can be redundant and/or contradictory. Certain features are given unwanted bias leading to misinterpreted data. Feature selection algorithms are applied to reduce the total feature space from its original dimension of 83. The goal is to determine which technical analysis and macroeconomic variables are the most valuable as input features. As a standard across all methods, 16 features are selected or extracted to pass to the forecasting algorithm as input.

Pearson's Correlation Coefficient Pearson's correlation coefficient is used to indicate the linear relationship between two variables. Essentially, the correlation value illustrates the extent that a pair of variables fluctuate together. The calculated correlation value is always within the range: -1 to 1. A value of 0 means there is no correlation between the two variables being evaluated. A correlation closer to 1 signifies a strong positive correlation while a value of -1 indicates a strong negative correlation. The method is used to measure the correlation between each feature and the target label, closing price (P). Pearson's correlation coefficient is denoted by

$r_{F,P}$ where F is a given feature and P the target variable, closing price. The mean of values for feature F is represented by $\bar{F}$ and similarly $\bar{P}$ for P. These calculations are used in computing the coefficients. For each feature F, the Pearson's correlation to P is given by

$$r_{F,P} = \frac{\sum(F_i - \overline{F})(P_i - \overline{P})}{\sqrt{\sum(F_i - \overline{F})^2 \sum(P_i - \overline{P})^2}} \tag{23}$$

Spearman's Correlation Coefficient Spearman's correlation coefficient is another statistical method that can be used as a technique for feature selection. Like Pearson's, Spearman's correlation is an incredibly popular method used in data science. Unlike Pearson's where a linear relationship is calculated, Spearman's correlation measures the strength and direction of the monotonic relationship between features and target variable. The resulting value represent the extent to which an increase, or decrease, in a feature F is associated with an increase or decrease in P. The tie between variables is measured by the covariance between ranks of a feature, (rg_F, and the target, (rg_P, divided by the product of each rank's standard deviation, σ_{rg_F} and σ_{rg_P}. The formula for covariance of the ranks and Spearman's correlation coefficient are obtained by

$$cov(\mathrm{rg}_F, \mathrm{rg}_P) = \frac{\sum_{i=1}^{N}(\mathrm{rg}_{F,i} - \bar{\mathrm{rg}}_\mathrm{F})(\mathrm{rg}_{P,i} - \bar{\mathrm{rg}}_\mathrm{F})}{N-1} \tag{24}$$

$$r_s = \frac{\mathrm{cov}(\mathrm{rg}_F, \mathrm{rg}_P)}{\sigma_{\mathrm{rg}_F}\sigma_{\mathrm{rg}_P}} \tag{25}$$

Mutual Information (MI) Feature Selection Mutual information (MI) is a measure of the codependency between two variables. It quantifies the amount of information gained by one variable through examining another. It measures the uncertainty reduced in one variable based off a known value of another variable. The idea is similar to correlation but a bit more generalized. MI is commonly used in applications to ML where MI quantifies how much information is contributed by a feature's presence when making an accurate prediction on the target label. Considering feature X and label Y, if X and Y are independent then the MI, $I(X;Y)$, is 0. If X is deterministic of Y then $I(X;Y)$ is the entropy of X. For each feature, X, the MI with respect to closing price, Y, is:

$$I(X;Y) = \sum_{x,y} P_{XY}(x,y) log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} \tag{26}$$

4.2.3 Feature Extraction

Feature extraction is similar to feature selection in that it reduces the dimensionality of the given feature space. However, instead of simply selecting a subset of the original feature space, feature extraction derives new values for a feature space with reduced dimensionality. The new features are meant to be more informative and less redundant. Feature extraction involves constructing new combinations of preexisting features to ensure overfitting does not occur in ML algorithms while maintaining an accurate description of the original data. Feature extracting is essentially constructing, rather than selecting, a new feature space for the training data.

Principal Component Analysis (PCA) Principal component analysis (PCA) is one of the most commonly used linear dimensionality reduction methods. The attempt is to find a new combination of the original feature set that includes a better summary of the data's distribution.

PCA maximizes variances and minimizes the reconstruction errors that may occur by focusing on pairwise distances. First, the input is projected onto a set of orthogonal axes which get ranked in order of importance.

4.3 Algorithms

In this section, the algorithms that are used in this study's procedure are listed and discussed. Many of them are explained in detail in section 3. Thus, for those algorithms, there is less explanation of their innerworkings and functions. Instead, the discussion includes more information on the implementation of the models. Makridakis' comprehensive study of statistical, ML, and DL methods for financial time series forecasting in 2018 is used to aid in deciding which methods to select for this research [28]. Three algorithms are chosen for statistical methods and four for machine learning. Several DL frameworks are analyzed in the study, though they are not included in the standard testing procedure. Overall, there are seven total models included in the experimental design. Algorithm parameters and implementations are optimized specifically for predicting market directions correctly.

4.3.1 Traditional Statistical Methods

Linear Regression (LR) Linear Regression (LR) is one of the most commonly used forecasting methods in all of data analysis. Univariate LR is the simplest version of the model which can be thought of as drawing a best fit line through the data. This is a statistical method, though regression can be turned into a ML technique. Obviously, one can foresee problems occurring when using this for financial time series forecasting due to the randomness and noise associated with the data. However, LR serves as a solid base model to use in tests. An ordinary least squares method is used to minimize the errors in predicted values.

Autoregressive Integrated Moving Average Model (ARIMA) ARIMA is one of the most common models used in statistical time series forecasting. It is a combination of a linear regression and MA forecasting model including the idea of integration to make the time series more stationary. Three parameters, p, d, and q, are optimized through the training process. p is the number of lag observations in the model. d is the amount of times that the values are differences (i.e.: the integrated component). q is the period length of the MA to use. Further investigation will be done to determine the best parameter selections. In this case, grid searching will be used to optimize the parameter selection for the model.

ETS (Error, Trend, Seasonal) Models ETS is a widely used family of univariate time series forecasting models known as one of the most successful classic statistical methods. Similar to EMAs, ETS builds forecasts by exponentially weighting values closer to time t with a smoothing function of the previous observations. The family of models can use any arrangement of additive and/or multiplicative errors, trends, and seasonal periods. Two ETS models are examined in this study. The first is simple exponential smoothing (SES), the simplest of ETS models. This model uses additive errors, no trend, and no seasonality. The second, Holt-Winters exponential smoothing (HWES), includes an additive seasonal and trend component to the forecast.

4.3.2 Machine Learning Methods

Linear Regression Model with Stochastic Gradient Descent (LRSGD) Linear regression can be converted to a multivariate ML method, rather than statistical, by incorporating an optimization algorithm. We will be using a Stochastic gradient descent algorithm for the cost function. Gradient descent functions calculate the partial derivatives of the loss function with respect to each variable. Updating weights accordingly optimal values for the model are found. The method allows for testing the ML version against its statistical counterpart, the simple LR model.

Support Vector Regression (SVR) The function of SVMs is explained in section 3.2.2. SVR is the regressive version of the SVM model. SVMs may be the most common model included in academic studies on financial forecasting and trading strategies. Providing an n-featured training set, SVR creates an $(n-1)$-dimensional subspace to draw the distinguishing hyperplane to separate datapoints. Provided the entire advanced input space, this creates an 82-dimensional subspace for the hyperplane to be drawn. This hyperplane is drawn using a specified kernel function, which can be as simple as a linear or polynomial function. Due to the high dimensionality of the feature space and complexity of the data, a radial basis function (RBF) with length scale equal to 1.0 is used. The mathematical formula for the RBF is illustration in equation 27 where $d(x_i, x_j)$ is the Euclidean distance between points. The literature suggests that RBFs are the best kernel to implement in this case due to their versatility, speed, and accuracy in highly dimensional feature spaces [8, 7, 28, 17]. The hyperplane within the subspace is drawn with two boundary planes whose distance from the hyperplane is specified by a parameter ϵ. The points within margin of ϵ are used in producing the output. The parameter C is used to control the tolerance for values outside of ϵ. In studies a value for C is usually chosen within the range $[0, 100]$ depending on the forecasting horizon and provided feature space. After several parameter optimization tests, a C value of 1 and ϵ of 0.001 is found to fit the differenced data for one-step forecasting the best.

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i,x_j)^2}{2l^2}\right) \tag{27}$$

Gaussian Processes (GP) Gaussian Process is a Bayesian probability model based in the assumption that the provided target variable is a function of one or many normally distributed random variables. Together a multivariate normal distribution is built by combining the distributions of each individual independent distribution. A prior distribution is assumed for the training data and then the posterior is calculated from the provided inputs. These are combined by a measure of similarity between points using a kernel function. A radial basis function (RBF) with initial length scale 1 is implemented to measure the distances between values and create the interpolation matrix. The equation for the kernel matrix created by an RBF is shown above below where d is the Euclidean distance between variables and l the length scale parameter. An initial noise level constant of 1 is also added to the diagonal of the kernel matrix to account for datapoint-dependent noise levels within samples. This leads to considerable increases in model performance. The BFGS algorithm, a quasi-Newton method that computes the inverse of the Hessian matrix, is used to optimize the log marginal likelihood function for for each kernel parameter. A single iteration of optimization is found to be sufficient for determining a valid length scale and noise level. This method specifically performs better when the target variable is normalized if the input is within the range of $[-1, 1]$ so this is implemented as a sole case for GP. The advantage of Gaussian pro-

cesses regression is that its prediction is probabilistic allowing for variance and standard deviation intervals to be reported with each prediction. These signify the region of which a prediction probabilistically should lie. Analyzing the resulting variance and standard deviation provides insight to the performance of a model and whether refitting or adjusting is necessary.

AdaBoost (AB) Decision trees are briefly explained in section 3.2.4. In this research, a form of boosted DTs are tested with AdaBoost for its strength in forecasting market prices. Adaboost turns DTs from a weak learner into a strong learner by converting it to an ensemble learning model. Adaboost functions differently than a traditional bagging model like random forests. The base DT in this implementation has a maximum depth of 2 and which Friedman mean squared error to determine the quality of a split in the tree. 50 total DTs are iteratively built where the following DT is built from the last to minimize the residual errors. The best performing trees from fitting the training data are given heavier weights with a learning rate of 0.75. The AdaBoost algorithm is implemented through the sklearn library [37, 33].

4.4 Evaluation Criteria

One of the most important steps to creating a research procedure is developing a standard evaluation set that allows for objective analysis of model combinations. Makridakis makes the point that the performance evaluation criteria used in many scholarly articles is flawed and making universalized claims by these studies unsubstantiated [28]. Therefore it is important to incorporate multiple means for assessing the success of a model. This study includes several types of evaluation metrics. Stock price prediction is usually done in two ways: numerical price prediction and price direction prediction. The specified metrics offer insight to the ability of regression models at both tasks.

To study the predictive accuracy of forecasting models one must examine the residuals, or error measurements, between the predicted closing prices, $\tilde{P}_t$, and their true values, P_t. The residuals, e_t, for n predictions at time t are

$$e_t = P_t - \tilde{P}_t, \qquad 1 \leq t \leq n \tag{28}$$

Taken from numerical analysis, norms based off the residuals are used to measure how far predicted values deviate from their true values. For the purposes of this study, maximum error (ME), mean absolute error (MAE), mean absolute percentage error (MAPE), and root-mean-squared error (RMSE) are used. ME highlights the worst prediction produced by an algorithm. It is important to note that high MEs do not necessarily correlate to high additional errors as there may be a single outlier in a set of 600 predictions. Thus, it is important to analyze the relations that arise between different types of errors to fully understand the results of a model. MAE simply measures the arithmetic average of the absolute residuals and is common within time series forecasting analysis. MAPE quantifies the prediction accuracy of forecasted values. Of the four norms, MAPE signifies a bad performance of an algorithm the best when its values are high. Note that the percentage version of the statistic is used where the regular formula is multiplied by 100. RMSE highlights when disproportionately larger residuals exists from to a given prediction set. The formulas for listed norms are:

$$ME = \max_{1 \leq t \leq n} |e_t| \tag{29}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{30}$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|e_t|}{P_t} * 100 \tag{31}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}|e_t|^2} \tag{32}$$

One problem arises when only analyzing the norms of a predicted series, there is no indication of profitability. In the end, the only metric that matters in financial forecasting is whether a model can produce profits. Models frequently produce low errors, but are unable to provide sufficient profits. This dynamic adds complexity to the process of creating and analyzing financial prediction systems. To quantify a systems profit-making abilities, the hit ratio (HR) metric is defined. This metric provides information about the percentage of forecasts that predicted an upward or downward trend correctly.

Ignoring the residual for a given instance, if the percent changes for the predicted value and the true value from the previous closing price having matching signs, then it is classified as a correct prediction, $P_t^{(+)}$. Otherwise, it is classified as an incorrect prediction, $P_t^{(-)}$. The hit ratio calculated by dividing the sum off the total number of correct predictions by the number of total predictions, $|P|$. This is multiplied by 100 to return a percentage for the HR. HR is arguably the most important metric tested in this research. It determines whether profits would be positive if a trading strategy were based off a given forecasting algorithm. Though, it is important to consider the additional statistics simultaneously as HR can be misleading given certain instances. The formula is show below

$$HitRatio = HR = \frac{P^{(+)}}{|P|} * 100 \tag{33}$$

The final statistic used to evaluate models is the increase ratio (IR). This is the percentage of predictions that forecasted an increase in value from the previous trading day. It is possible for a model to learn the long-term trend of upward growth in the stock market and forecast that prices will rise regardless of additional input. When this occurs it is an obvious indication of model failure. A model can produce low errors and a high HR making it seem like the system is ideal. This metric is crucial to knowing when false successes occur allowing for an empirical analysis of model results. For each prediction at time t, an increased prediction is denoted $P_t^{(>)}$. The IR value is given by the number of increasing predictions divided by the number of total predictions, $|P|$, multiplied by 100.

$$IncreaseRatio = IR = \frac{P^{(>)}}{|P|} * 100 \tag{34}$$

The bottom criteria for a evaluating a models performance is whether it produces consistently low errors and can beat a random walk. A random walk occurs when tomorrows price is always predicted as the current price. This results in a HR of 50%. At the bare minimum an model must consistently outperform a random walk and return HRs greater than 50%. We will consider this benchmark #1. However, predicting the upward/downward trends of markets just 50% of the time is insufficient to considering a model truly successful. Hypothetically, an unfavorable algorithm could learn to predict prices will only increase resulting in a 100% IR. We would hope a model can pick up on the increasing prices easily due to the global trend. Thus, the number of downward

trend reversals that an algorithm forecasts correctly becomes an important metric for analyzing model performances. One way it can be evaluated is by examining the relationships between HR and IR when errors are low. However, the specific criterion is whether it can beat the HR produced by guessing market prices will only increase. We will call this benchmark #2 for evaluating model performances with each ETF. The particular cases for the HR with this benchmark for each ETF are as follows:

$$SPY \longrightarrow 56.87\%$$
$$QQQ \longrightarrow 57.98\%$$
$$IWM \longrightarrow 54.66\%$$
$$DIA \longrightarrow 55.29\%$$

4.5 Procedure

4.5.1 General Testing

The aim is to determine which combinations of input data, preprocessing methods, and forecasting algorithms best forecast market data. One must acknowledge that univariate statistical models are limited to taking a single input, closing price, when making predictions. This allows for separate testing of the univariate models from the ML multivariate algorithms. In two prominent studies, Makridakis claims that ML and DL multivariate models are unable to beat their simpler statistical counterparts [27, 28]. The partitioning of tests between univariate and multivariate models makes testing that claim easy. For the ML and DL algorithms, the closing price is the label, so it is removed from the input feature vector. All together, the feature space for the multivariate algorithms is made up of open, high, and low prices plus volume, 75 technical indicators, and three macroeconomic variables. The resulting advanced feature space includes 83 input features for each of the four ETFs. Rather than combinatorically testing all possible combinations of the feature space as input (83! different permutations), feature selection and extraction methods are used to find the optimal feature sets. Models are tested for accuracy in one-step ahead forecasting. The procedure for statistical methods generally consists of optimizing parameter selections and testing train-test split ratios for model performance. Scaling and/or feature engineering is an unnecessary step for statistical methods. The combinatorial approach is used for testing the multivariate ML model combinations. ML algorithms are tested one by one with each arrangement of components. First, basic OHLC data is provided to each model with only the base preprocessing as described in the beginning of section 4.2. Then scaling, standardization, and normalization techniques are tested for each model with the basic input. This process is repeated for the complete advanced feature space of 83 columns. Subsequently, each of the four feature engineering methods is tested with all preprocessing techniques to test different derived feature spaces and their impact as input to the model. The standard evaluation criteria is applied to each model combination as results were recorded. Looking at both the error norms and HR offer insight to the true performance of algorithms. Additionally, parameter optimization is continuously tested with each model throughout the process to find the best model combinations. For all tests daily data sample were observed from January 1st, 2012 to September 31st, 2020. This is a total of 2920 time series observations.

4.5.2 Naive Procedural Design

The above procedure was consistent through all procedural designs. As a base methods for testing models, initially stationarity was not implemented to give predictions of the raw data. This leaves systems with the task of forecasting the true closing price rather than differenced values. In the first test of models a traditional train-test split was used with a training ratio of 75%. Though due to the inherent structure of financial time series data, this traditional machine learning approach is flawed. Models are trained on one portion of data so they can forecast values within that range. The model is then tested on values within a drastically different range. For an index like QQQ, this means models are trained to predict closing prices within $50-150$, then tasked with predicting values from $150-300$. This leads to incredibly poor results. Predictions deviate from their true values almost immediately in this type of implementation. For ML algorithms such as GP, SVR, and AB, models can hardly predict values higher than the maximum value within testing data. Predictions turned almost inverse trend-wise to that of their true values. Only when recessions or pullbacks occurred in data where prices return to the range from training could models predict prices accurately again. This is pictured as an extreme in Figure 77. If the same trends in stock data repeat over every date range this process could work. However, with the noisiness and inherent randomness of financial time series this property is far from holding true. Consequently, models predict values based on trends from the first 2190 data samples assuming those same patterns continue in the next 730 days. Clearly, models are setup for failure in this implementation, so true indications of how models perform against financial markets cannot be concluded from results obtained with this approach.

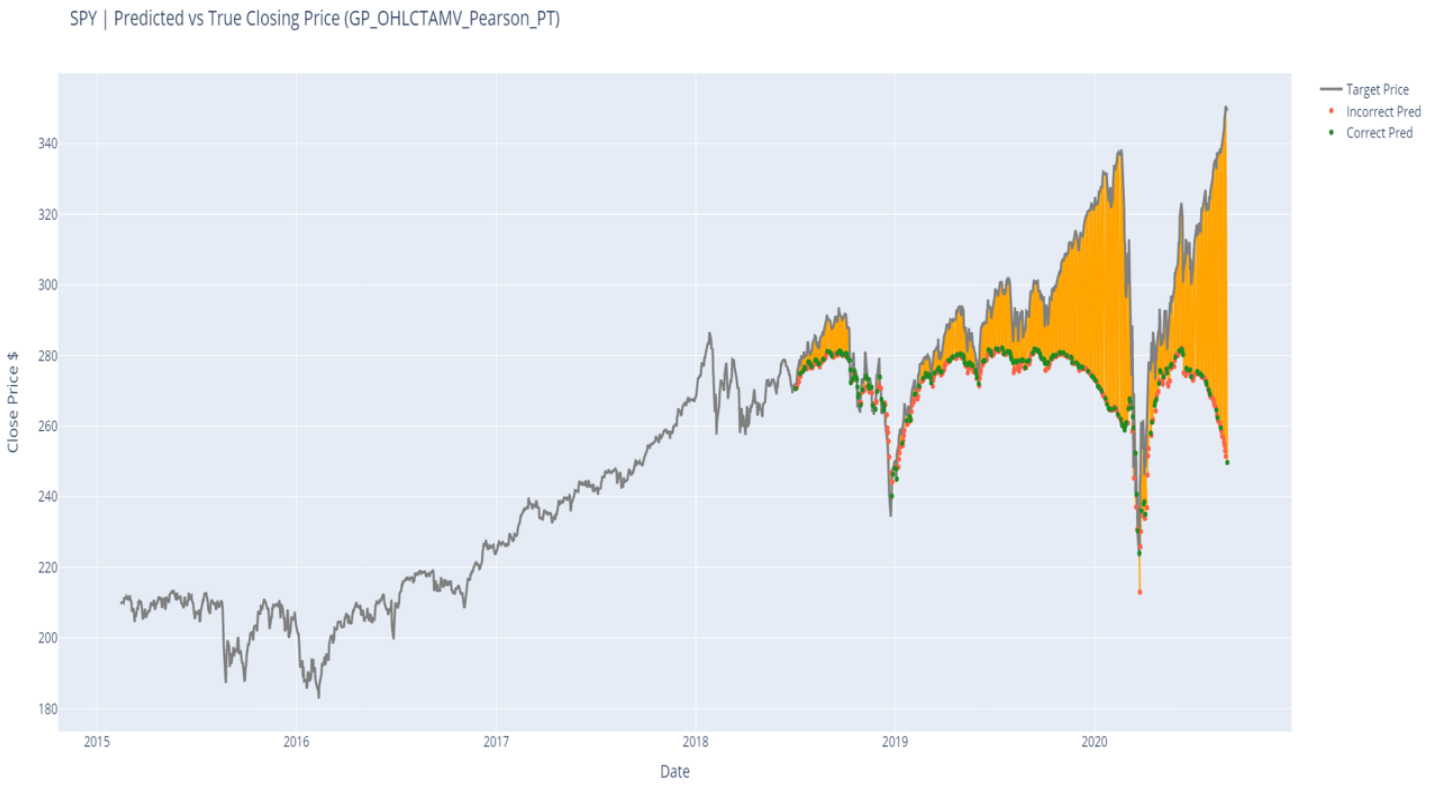

Figure 7: Gaussian Processes Regressor with unshuffled 75-25 train-test split

The intuitive next approach is to test the same training ratio, but with shuffled data. Models using this strategy consistently produced MAPEs under 1% with incredibly high HRs. Models such as GP reached an average HR of 90% while SVR and AB obtained values near 80%. However, this approach is flawed similar to the prior method. When models are trained and tested on the same range of time series data, overfitting begins to occur. When implemented in this way algorithms learn the exact trends that exist within a certain range of dates. Thus, predicting the missing 25% of time series data is incredibly easy for algorithms. If the model knows the exact prices surrounding a given date, it is simple to predict the value in between. This is especially true for a probabilistic model like GP. Similarly, SVR predictions are accurately produced within the tolerance boundaries of the hyperplane. DTs easily overfit the testing data due to its similarity to the training data which is learned to a point of high granularity. When tested on out of sample data the same issues as the previous method occurs. If new prices were below the maximum value within training predictions could be accurate. However, if values are higher than the maximum larger prediction errors start to occur after only a few new observation.

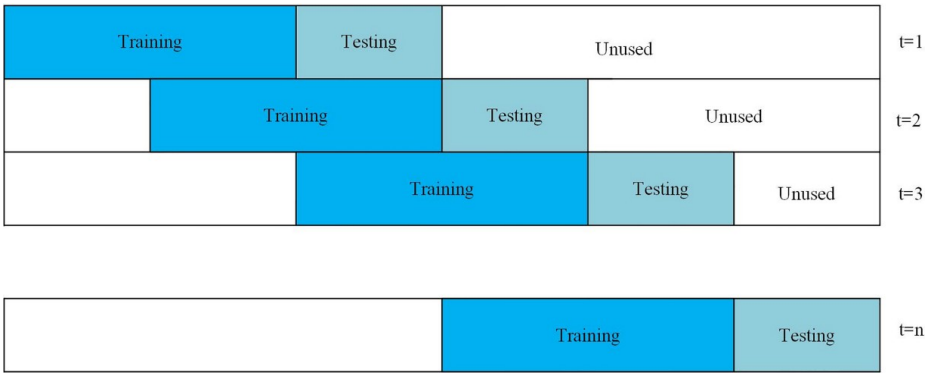

Figure 8: The systematic approach to walk-forward validation

4.5.3 Walk-Forward Validation

Walk-forward validation (WFV) is the ultimate method for testing time series forecasting methods. WFV is utilized for financial time series forecasting and prediction systems to allow for iterative rolling optimization of models [7, 8, 17, 24, 25]. WFV solves many of the issues of that arise in the previous approaches. The method functions by isolating data into distinct segments, or walks, each divided into training and testing sets. A non-anchored WFV method is used creating a sliding window based approach where the training set keeps the same size throughout model testing. This procedure is illustrated in Figure 88. The model is trained on the first period leading up to time tt. Then a value is tested for time t+1t+1. After a prediction is made and recorded the training set slides forward for a single period (1 day) and the process of training and testing is completed again. This creates a real life one-step forecasting scenario where historical data of specified length is used to forecast the closing price for the following day. Data preprocessing and feature selection is individually performed for each iteration on the training data. This ensembles the appropriate scaling and feature set for each given forecast. This method is ideal since the best features for a one time period may not be the same as months later. Additionally, model parameters can be optimized for the current iteration's training data. This allows for continuous adaptation of models relative to changes in market conditions. The size of the optimal sliding window varies depending on the model. Market data from 200 days ago may be irrelevant to one

algorithm, but add value to another. Carta tested multiple different walk sizes against an SVR model. He found that decreasing the size of the window was advantageous to his forecasting model [8]. Rather than using a set training size using up to 200 days prior, a window of length 80 was more appropriate for his model implementations. In this study the initial end point of the training data was set to February 26, 2018. For example with a window size of 100, data beginning at November 2nd, 2017 is used for initial training. The window is shifted by a single observation each iteration ultimately making 633 forecasts.

5 Results and Evaluation

5.1 Are Statistical Methods Truly Reliable?

The first asked is whether statistical based univariate methods are superior financial forecasting models as Makridakis and the findings of the M-3 competition suggest [28, 27]. linear regression (LR) is the most basic technique that can be applied to predicting market prices . Applied with univariate input the forecasting method models the linear relationship between time t and closing price P_t. Applied on the entire dataset of 2920 samples this simply draws a best fit line through the $P_t - plane$ where the errors are minimized by ordinary least squares method. Obviously, a straight line through axes does a poor job of estimating future prices. ME reached up to 65.35075 and MAPEs neared 8% confirming that observation. However, when WFV is utilized the results change. With window size 100, this statistical technique is essentially identical to an SMA forecasting method where the predicted value is expected to continue the exact trend as the previous 100 days. A line drawn between forecasted values is a smooth trend line. The larger the window size is, the farther predictions lag from their true values. By decreasing the window size to an extremely small value the model becomes incredibly accurate. For a window size of 7 the method achieves an average MAPE of 1.2667%1.2667% and HR of 64.89%64.89%. Decreasing the window size even more to a value of 3 obtains a mean HR of 74%74% and 1.2%1.2% MAPE. Implementing the same values with stationary data reaches MAPEs of 0.9323%0.9323% and 0.8058%0.8058%, all with extremely low ME. This illustrates the reliability of statistical forecasting techniques and the ability to consistent produce accurate predictions. As the window gets smaller this methods gets closer and closer to a modified random walk. Rather than setting $P_{t+t} = P_t$, the model predicts tomorrows closing price as the next point on the best fit line drawn through the size of the window. With a training window size of 3, the model technically produces the best metrics from this study. However, that is somewhat of a facade since as mentioned, this is really a modified random walk. Indeed the model produces good metrics, however employing this strategy in real life is unreasonable. Assuming tomorrows closing price will always continue the exact trend of the past few days has a high probability of being correct. However, there is no capacity to predict reversals in trend. This illustrates the naiveness of some statistical forecasting methods, simple to an extent that its unrealistic. Though it does work to show the effect of implementing walk-forward optimization and result from varying window sizes.

Other statistical methods take advantage of more complex techniques providing results that more realistically represent how they fit different markets. ARIMA is implemented with an optimized parameter order of (4,1,4)(4,1,4). This includes 4 lag observations, 1 differencing term, and a moving average window of 4 samples. This order is configured through a grid search algorithm that ensures overfitting does not occur. ARIMA is able to produce an average MAPE of 1.404%1.404% with a ME of 35. Relative to other models, these metrics are not particularly

outstanding. The MAPE produced by ARIMA shows it can predict values mostly within a certain threshold. However, the ME is high and the method fails to produce an HR above 52%52%, averaging at 50%. This is no better than a random walk. ARIMA can be used for numerical approximation within a certain error tolerance, but not better than many of the other algorithms tested in this study. Additionally, its capacity to predict the trends within the given error boundaries is insufficient. The two ETS models tested from the statistical category produced errors very similar to ARIMA. Simple exponential smoothing (SES), which includes additive error and trend decompositions, results in an average MAPE of 1.3812% and HR of 46.52%. Considering the HR remained under 50%, this model clearly has little ability to notice the directional trends of forecasts, though like ARIMA, it has a similar error tolerance. HWES, the other ETS model, performs slightly different than the other statistical models. Unlike SES, HWES includes an additive seasonal component. An optimization algorithm is used to find that a assigning 2 seasonal periods to the model produces the best metrics. HWES results in an average MAPE of 1.3925%, no better relatively than the other statistical models. However, the average HR is 51.4%. This shows adding the seasonal component gives it an edge to predict trends correctly, though the errors stayed relatively similar.

5.1.1 Soft Computing Techniques

The combination of enforcing stationarity within data and utilizing WFV allowed for soft computing models to beat the given benchmarks outlined in the evaluation criterion. ML models outperform traditional statistical time series forecasting techniques such as ARIMA, SES, and HWES once these techniques are implemented. SVR struggles to predict values accurately without input data being stationary. Once differenced, predictions become much more consistent and the average HR for each model combination increased. A training window of length 50 produces the best outputs. With only basic OHLC data as input, SVR performs the best in each metric with solely the maximum absolute scaler used in preprocessing. The average MAPE was 0.987%0.987% with an HR of 59.44%. Errors increase and HR decrease when other preprocessing techniques are tested on the OHLC data. MAPEs surpass 1% and HRs remain near 54% when using a standard scaler, min-max scaler, and the Yeo-Johnson transform on the basic OHLC data. When tested on the complete advanced feature space SVR performs better. For this input a min-max scaler produces the best metrics in each categories with an average ME of 23.6, MAE of 2.0, MAPE of 0.951%, and HR of 63.86%. After testing each feature engineering method, mutual information was found to produce the best forecasting models. It consistently has the lowest MAPEs and best HR of all SVR systems. The single best SVR model is attained by combining mutual information with standardization. A MAPE of 0.939% is obtained with an HR of 64.04%. The results of this system are plotted in Figure 99.

SVR performs differently with the other feature selection techniques. Pearson's correlation and Spearman's correlation both result in worse metrics than when using the entire advanced feature space. This indicates that these methods select poor features for SVR while mutual information achieves the best input combination. Support vector machines typically do better with higher dimensioned feature spaces. This is illustrated by the increase in performance from using OHLC data to applying the advanced feature space. The significant increase in HR and decrease in errors combined with mutual information as feature selection implies a number of those were unnecessary inputs.Thus, methods based on Pearson and Spearman correlations were unable to select the right features for SVR. It is important to also note the change in IR that occurs between models. Though OHLC data with a basic maximum absolute scaler produces the best metrics

Figure 9: Support Vector Regression with a standard scaler and mutual information

within that input space, the IR is 88.23%. This implies the model picked up on the global trend and is almost entirely predicting the price will increase. The best combination without any feature engineering on the advanced space had an IR of 70.62% and once mutual information is implemented that decreased to 66.85%. Therefore mutual information leads to better metrics and less of a focus on the global increasing trend of the data. This illustrates the need for additional features, but only those which add valuable information to the SVR prediction model. In this case, mutual information is the best means of deriving feature importance for SVR.

GP displayed an interesting dynamic distinct from that shown by SVR. GP was not able to produce metrics above the benchmarks when data was not considered stationary. Though, the results differed greatly from when data is and is not differenced beforehand. The optimal size of training window was found to be 100 for GP. First with the stationary OHLC data, GP performed the best with no additional preprocessing and solely a maximum absolute scaler. This system produces a MAPE of 0.9904% and a HR of 56.83%. When tested on the advanced feature space, GP produced the best HR when a standard scaler is utilized. The MAPE for this combination is 0.994% with HR 66.67%. However, when no preprocessing except for the maximum absolute scaler was used on the advanced feature space, the MAPE decreased while the HR increases. In fact, the basic input and advanced feature space the models produced almost identical results for the stationary data with only an absolute maximum scaler. This has to do with GPs characteristic inner functions. The output of GP is produced by calculating the posterior distribution of each feature individually, then the predictive distribution is calculated by combining them weighted by their measure of similarity according to the covariance matrix created by the RBF. From the results it is apparent that the RBF determines the differenced OHLC prices are the most heavily weighted

in the predictive distribution for tomorrows closing price. Additionally, almost identical metrics are produced when each of the feature selection techniques are implemented with only an absolute maximum scaler. With each of the feature selection techniques, open, high, low, and close prices are always part of the feature subset passed to the model. This indicates the same process occurs each time. Additionally, feature selection never has a large impact on the model, making it worse in some cases, regardless of the preprocessing method used. In some sense, GP already completes a sort of feature selection when it combines the individual distributions weighted by the kernel function's measure of covariance between each feature and the target variable from training data. When testing on non-stationary data GP produces an interesting set of results. Errors increase for every model commonly within the range of 1.1% to 1.5%, though HR increases for every model. Contrary to what one would assume this is not due to a higher IR. The IR for all GP models tested on stationary data varies between 64% and 67%, always staying within that range. Though when tested on non-stationary inputs, ignoring a few outliers, the IR ranges from 30% to 45%. This means GP is predicting the price will decrease more than 50% of the time when provided with non-stationary data. The reason behind this is similar to why the naive procedure fails. As future prices increased, the model was trained on values from a lower price range. Thus, when tested on new values outside of that range, the model only knew how to predict values like the ones that arose in the training set. It was not until pullbacks occurred, like with the coronavirus crash, that GP predicted prices would increase again. When values are stationary, the newly introduced testing data cannot be significantly outside of the range from training. This gives the algorithm an edge in performance.

Linear Regression with Stochastic Gradient Descent (LRSGD) is the least impressive of the soft computing models. By using stationary data and WFV, LRSGD can achieve low errors relative to the three basic statistical methods. However unlike the other ML models, LRSGD was unable to achieve an average MAPE under 1% with any combination of preprocessing and feature engineering. Of the preprocessing methods tested with OHLC data, min-max scaler gives the best results. It is also the best combination of any arrangement of components for LRSGD. With these components, the system achieves an HR over 50% for each ETF with the average being 53.79%. Though better than the benchmark #1, a random walk, relative to the other models in this section this is unimpressive. A standard scaler and the Yeo-Johnson transform do not produce HRs over 48% for any ETF. Errors become considerably worse when the full advanced feature space is applied to LRSGD. The ME goes from 28.19 to 61.01 with a min-max scaler and 146.03 with a standard scaler. This is not surprising, but rather expected. The other soft computing techniques are capable of learning trends within and handling higher dimensioned feature spaces. LRSGD is really just an adapted version of simple linear regression with a learning component to minimize the squared loss of the ordinary least squares method. Features in the advanced space do not necessarily provide information directly applicable to predicting tomorrow's closing price. Thus, features like RSI which are not so obviously closely tied to the target variable hurt the algorithm's performance. Errors get better once feature selection techniques are implemented. However, they do not reach the errors as low as from using only OHLC data.

AdaBoost (AB) was the final ML model tested in this research. Basic DT regressors are tested in the initial procedure. With WFV, single DTs struggle to perform well and produce results with an IR below 50%. AB performs better, especially with stationary data, due to its weighted ensemble learning structure. A training window of size 100 is used to build the AB model. Interestingly, preprocessing has little to no effect on AB when given basic OHLC data as input. An ME close to 35 and MAPE around 1.05% with average HR of 56.5% was achieved by each of these four models. One of the main purposes of preprocessing is to eradicate outliers. DTs

are not effected heavily by outliers since the partitioning of each tree looks for a single location to make a split within each feature's distribution. AB gives outliers even less of a bias due to its iterative scheme. Additionally, a power transformation only shifts the location of an optimal split point, so the structure of the tree remains somewhat the same. AB performs worse when provided with the entire advanced feature set. Unlike with OHLC data, results vary between the preprocessing techniques for each derived input from the feature engineering methods. This is due to different sets of selected features due to varying distributions and values depending on the scaling, standardization, or normalization of data. When mutual information is combined with the model similar results are achieved. Results suggests the AB algorithm is insufficient at determining the correct splits when provided with features less directly tied to the target. Pearson's and Spearman's correlation also failed at providing AB with an optimal feature set to forecast values better than with basic OHLC data.

6 Conclusions

Soft computing and computational intelligence has progressed an astonishing amount over the past decade. What used to be a field dominated by traditional statistical forecasting has been taken over by novel and adaptive learning based algorithms. This work provides a comprehensive review of methods and models proposed in financial prediction systems. Then, the application of chosen statistical and ML financial forecasting models are tested to compare their performances. The procedure tests over 150 forecasting systems to determine the effects of input data, preprocessing methods, and feature engineering on each model's ability to predict market prices. A standard evaluation set is used to measure the quality of each system so that both forecasting accuracy and price direction prediction is analyzed together. This verifies the true performance of forecasting systems since low errors were achieved by many, but only a few provide further insight to the trends of the market.

The first research question was clearly answered on whether ML models can consistently outperform their statistical counterparts. Though they can produce somewhat low errors, classical forecasting techniques show little ability to forecast the direction of stock movements within the margin of error. Forecast errors decrease relative to the statistical methods for many of the tested soft computing based systems. The results of this study contradict the conclusions of others, notably, Makridakis' investigation in 2018 and the M-3 competition results. [28, 28]. Specific techniques are used that allow for those claims to be confidently refuted. Looking at only the one-step forecasting results from the study, the procedure trained models and tested models a single time on just the last portion of data samples for each stock. No additional training or testing was completed for any given ticker. It is noted that by testing over 1000 tickers the method is still valid, though it alone cannot be used to assure the claim that statistical methods are superior. Furthermore, no notion of profit or evaluation other than errors and model fitness was included in the study. The use of walk-forward validation to backtest model performances is the determining factor illustrating that ML models are not inferior to their fellow statistical based methods. The simulation of real life scenarios with no bias or data leakage between training and testing gives the results of walk-forward optimization validity. This is backed by the findings of countless others. [8, 25, 17, 19, 21].

Two benchmarks are set out as base criteria for models to overcome. Specifically, it was found that two forecasting models stand out from the rest showcasing the capacity to beat each benchmark. With the addition of 75 different technical analysis indicators and 3 macroeconomic

variables, GP and SVR gain that distinction. Not only were these algorithms able to predict values with MAPEs under 1%, but trends in price movement are predicted correctly on average up to 59% and 64% of the time within that margin of error. GP illustrates a certain robustness to input when data is stationary by weighting features effectively and accounting for noise in order to predict market directions sufficiently. As a relatively less researched model compared to others discussed in this research, the application of GPs to additional machine learning tasks offers a promising potential. SVR also demonstrated that combined with the right feature selection, price directions can be predicted with great ability.

The importance of individually based model application is emphasized by the findings of this study. Across model results, performances vary greatly depending on the additional components of the system. One research question was to determine whether it is possible to universally claim certain model components to be generally superior. SVR and GP were found to perform consistently well. However, the improvement that any given preprocessing method or feature engineering technique added was completely model dependent. The fact that a component is beneficiary in one instance does not guarantee it will be useful in others. There is one claim that can broadly be made about ML models in the field of financial time series forecasting. That is the importance of stationarity within feature spaces for supervised machine learning regression problems. Every model tested in this study experienced an increase in predictive ability once stationarity was enforced on a feature-wise basis.

It is important to acknowledge aspects of research that can be improved or points of question within any given study. It is also noted that the tested ML systems generally have a worse ability to forecast all time highs and all time lows within data. Fixing the issue of accurately predicting local all time highs is difficult. However, increasing training size so that local lows are within the range of training data is a potential solution. Though, this can work counterintuitively since it was found that increasing window size always reaches a point where additional observations hurt predictions rather than help. One additional point of critique is that more statistical methods should be included in the study in order to assert the claim that ML models are superior. There may be other methods not considered that are capable of producing metrics similar to GP and SVR. It is especially important to note that enforcing stationarity by differencing and subsequently inverse differencing predictions makes it easier for models to produce low errors. The new target values represent the change in tomorrows closing price meaning they are smaller numbers and distributed closely to zero. Thus, models are more capable of predicting values within this sort of range, i.e: -3 to 7 compared with 250 to 330. It would be naive not to acknowledge this dilemma. At the same time this is the beauty of the technique since testing data used for making out of sample predictions cannot be significantly outside the range of values the model has already been introduced to. Lastly, additional forecasting horizons should be tested to verify the results from this study. GP and SVR are found to perform the best with one-step forecasts of stationary financial data, but the same may not be true of week out predictions.

7 Future Work

This study analyzes what is a small portion of a vast and incredibly dense field. Thus, this research is an ongoing study that will continue past the horizons of the material confined to this paper. The first step is implementing new procedures to test the acknowledgements highlighted in the previous section. Multiple forecasting horizons should be assessed for each model to specifically to see if ML models can continue to outperform statistical methods. Additional models

should be examined with a similar walk-forward optimization procedure to increase the breadth of research. One preprocessing technique commonly used in the literature but not included in this study is wavelet transforms. Wavelet transforms can provide significant improvements to model performances according to numerous studies. As another step in preprocessing, wavelet transforms are a category of noise minimization methods that may be able to additionally increase models accuracies. They take advantage of looking at time series data in the format of frequencies where Fourier transforms will often be used to manipulate input data. Looking into such methods in the future would be advantageous. Similarly, SAEs offer the potential for highly informative feature spaces to be extracted. Deep learning models offer great potential to discover hidden trends in data that statistical and ML based methods are incapable of. Neural network frameworks have become a major subject of focus within the field of financial engineering over recent years. LSTMs have the capacity to hold "memory" and analyze entire batches of time series data within single samples of input giving promise to financial forecasting. The method with the most potential to increase model performance is incorporating external variables into model inputs. It is impossible to predict certain events that have large effects on the stock market. Creating news APIs that extract and analyze data from Google Trends, Twitter, or Reddit offer the best means to truly have predictive powers and consistently beat the EMH. Altogether, the endless stream of components that offer great potential to financial forecasting models is what drives the study and discovery of novel innovative methods that often break the existing barriers within the field of machine learning and computational intelligence.

System Specifications:

Lenovo ThinkPad X1 Extreme
Processor: Intel(R) Core(TM) i7-8850H CPU @ 2.60GHz (Turbo 4.30GHz), 9M Cache, 6-core(s), 12 Logical Processor(s)
Graphics: GeForce GTX 1050 Ti Max-Q (4GB)
RAM: 32GB (16GB + 16GB) DDR4, (2400MHz)
Storage: 1TB PCIe NVMe SSD & 516GB PCIe NVMe SSD (read/write - 3,500/3,300 MB/s)

References

[1] Federal reserve economic data (fred). https://fred.stlouisfed.org.

[2] Yahoo finance - stock market live, quotes, business & finance news. https://finance.yahoo.com.

[3] Farrukh Ahmed, Dr. Raheela, Dr. Saman, and Muhammad Muzammil. Financial Market Prediction using Google Trends. *International Journal of Advanced Computer Science and Applications*, 8(7), 2017. Publisher: The Science and Information Organization.

[4] Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, and Ruben Gryp. Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42(20):7046–7056, November 2015.

[5] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7):e0180944, 2017. Publisher: Public Library of Science (PLoS).

[6] Houssem Ben Braiek and Foutse Khomh. On Testing Machine Learning Programs. *Journal of Systems and Software*, page 110542, 2020.

[7] L.J. Cao and F.E.H. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518, 2003. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[8] Salvatore Carta, Andrea Corriga, Anselmo Ferreira, Diego Reforgiato Recupero, and Roberto Saia. A Holistic Auto-Configurable Ensemble Machine Learning Strategy for Financial Trading. *Computation*, 7:67, 2019.

[9] Rodolfo C. Cavalcante, Rodrigo C. Brasileiro, Victor L. F. Souza, Jarley P. Nobrega, and Adriano L. I. Oliveira. Computational Intelligence and Financial Markets: A Survey and Future Directions. *Expert Systems with Applications*, 55:194 – 211, 2016.

[10] Damien Challet and Ahmed Bel Hadj Ayed. Predicting Financial Markets with Google Trends and Not so Random Keywords. *SSRN Electronic Journal*, 2013. Publisher: Elsevier BV.

[11] Eunsuk Chong, Chulwoo Han, and Frank C. Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017. Publisher: Elsevier BV.

[12] F. Dan Foresee and M.T. Hagan. Gauss-Newton approximation to Bayesian learning. IEEE.

[13] Rajashree Dash and Pradipta Kishore Dash. A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science*, 2(1):42–57, 2016. Publisher: Elsevier BV.

[14] Scott Dick, Andrew Tappenden, Curtis Badke, and Olufemi Olarewaju. A granular neural network: Performance analysis and application to re-granulation. *International Journal of Approximate Reasoning*, 54(8):1149–1167, 2013. Publisher: Elsevier BV.

[15] Jing Duan. Financial system modeling using deep neural networks (DNNs) for effective risk assessment and prediction. *Journal of the Franklin Institute*, 356(8):4716–4731, May 2019.

[16] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018. Publisher: Elsevier.

[17] Marija Gorenc Novak and Dejan Velušček. Prediction of stock price movement based on daily high prices. *Quantitative Finance*, 16(5):793–826, 2016. Publisher: Informa UK Limited.

[18] Hongping Hu, Li Tang, Shuhua Zhang, and Haiyan Wang. Predicting the direction of stock markets using optimized neural networks with Google Trends. *Neurocomputing*, 285:188 – 195, 2018.

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

[20] Firuz Kamalov. Forecasting significant stock price changes using neural networks. *ArXiv*, abs/1912.08791, 2019.

[21] Ramon Lawrence. Using neural networks to forecast stock market prices. January 1998.

[22] Jinsong Leng. Modelling and Analysis on Noisy Financial Time Series. *Journal of Computer and Communications*, 02(02):64–69, 2014. Publisher: Scientific Research Publishing, Inc,.

[23] Jialin Liu, Fei Chao, Yu-chen Lin, and Chih-Min Lin. Stock Prices Prediction using Deep Learning Models. *ArXiv*, abs/1909.12227, 2019.

[24] Dongdong Lv, Zhenhua Huang, Meizi Li, and Yang Xiang. Selection of the optimal trading model for stock investment in different industries. *PLOS ONE*, 14(2):e0212137, 2019. Publisher: Public Library of Science (PLoS).

[25] Dongdong Lv, Shuhan Yuan, Meizi Li, and Yang Xiang. An Empirical Study of Machine Learning Algorithms for Stock Daily Trading Strategy. *Mathematical Problems in Engineering*, 2019:7816154, April 2019. Publisher: Hindawi.

[26] Nijolė Maknickienė, Aleksandras Vytautas Rutkauskas, and Algirdas Maknickas. Investigation of financial market prediction by recurrent neural network. *Innovative Infotechnologies for Science, Business and Education*, 2(11):3–8, 2011.

[27] Spyros Makridakis and Michèle Hibon. The M3-Competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000. Publisher: Elsevier BV.

[28] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3):e0194889, 2018. Publisher: Public Library of Science (PLoS).

[29] Mortada Mehyar. fredapi. https://github.com/mortada/fredapi, 2019.

[30] Tong Meng, Xuyang Jing, Zheng Yan, and Witold Pedrycz. A survey on machine learning for data fusion. *Information Fusion*, 57:115–129, 2020.

[31] Mahla Nikou, Gholamreza Mansourfar, and Jamshid Bagherzadeh. Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management*, 26(4):164–174, 2019.

[32] Jigar Patel, Sahil Shah, Priyank Thakkar, and K. Kotecha. Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259 – 268, 2015.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] Tobias Preis, Helen Susannah Moat, and H. Eugene Stanley. Quantifying Trading Behavior in Financial Markets Using Google Trends. *Scientific Reports*, 3(1), 2013. Publisher: Springer Science and Business Media LLC.

[35] Biao Qin, Yuni Xia, Shan Wang, and Xiaoyong Du. A Novel Bayesian Classification Technique for Uncertain Data. page 25, 2011.

[36] Peter Sarlin and Kaj-Mikael Björk. Machine learning in finance—Guest editorial. *Neurocomputing*, 264:1, 2017.

[37] scikit learn. scikit-learn. https://github.com/scikit-learn/scikit-learn, 2020.

[38] 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, 2020.

[39] Omer Berat Sezer, Murat Ozbayoglu, and Erdogan Dogdu. A Deep Neural-Network Based Stock Trading System Based on Evolutionary Optimized Technical Analysis Parameters. *Procedia Computer Science*, 114:473–480, 2017. Publisher: Elsevier BV.

[40] Chao Shi and Xiaosheng Zhuang. A Study Concerning Soft Computing Approaches for Stock Price Forecasting. *Axioms*, 8(4):116, 2019.

[41] Xiaobo Tang, Shixuan Li, Mingliang Tan, and Wenxuan Shi. Incorporating textual and management factors into financial distress prediction: A comparative study of machine learning methods. *Journal of Forecasting*, n/a(n/a), January 2020.

[42] Jar-Long Wang and Shu-Hui Chan. Stock market trading rule discovery using pattern recognition and technical analysis. *Expert Systems with Applications*, 33(2):304 – 315, 2007.

[43] Sanford Weisberg. Yeo-Johnson Power Transformations. January 2001.

[44] Jing Zhang, Shicheng Cui, Yan Xu, Qianmu Li, and Tao Li. A novel data-driven stock price trend prediction system. *Expert Systems with Applications*, 97:60 – 69, 2018.

[45] Y.-Q. Zhang, S. Akkaladevi, G. Vachtsevanos, and T. Y. Lin. Granular neural web agents for stock prediction. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 6(5):406–413, 2002. Publisher: Springer Science and Business Media LLC.

[46] Lin Zhao. Neural Networks In Business Time Series Forecasting: Benefits And Problems. In *BIS 2011*, 2011.