An Exploratory Analysis of Machine Learning for Stock Market Predictions: Using the Power Set of Relevant Features to Enhance Performance

Eddie Aljamal, Alex Janosi, Sisir Potluri, and Shashank Verma

***Abstract*— In this study, we present a comprehensive investigation into stock price prediction, leveraging a dataset from the Japanese stock market provided by a Kaggle challenge. Our approach introduces a novel pipeline that tests a hypothesis concerning the efficacy of feature engineering in predictive modeling. Recognizing the critical role of feature engineering, we employ Principal Component Analysis (PCA) to distill and enhance the dataset's informative value. This preprocessing step facilitates the discovery of PCA components that capture the most variability of significant features for stock price predictions, and we traverse the power set of those components to train models across four model types: Long Short-Term Memory (LSTM) networks, Transformer models, Gradient Boosted Trees (via XGBoost), and LASSO regression. We then evaluate their performance in the context of stock price forecasting and assess how the PCA-based approach compared with training using the full feature set. Through rigorous experimentation, our findings highlight the superior performance of the XGBoost algorithm, which achieved a notable Sharpe Ratio of 0.133. This outcome was observed when utilizing a comprehensive feature set derived through PCA and trying different subsets of engineered features, underscoring the pivotal contribution of sophisticated feature engineering and selection to stock price predictions.**

I. Introduction

In the dynamic landscape of financial markets, especially in large market settings, investors and analysts are constantly refining their strategies to understand and foresee market movements. To tackle this challenge, they've increasingly turned to mathematical tools and algorithms, which range from basic statistical approaches to cutting-edge machine learning techniques. The prime objective is to create software that can anticipate market trends, and machine learning (ML) has emerged as a potent tool for predicting individual stock index prices in larger markets. ML techniques offer a robust set of methods for analyzing vast quantities of data and identifying complex patterns that may elude traditional statistical methods. Modern ML-based strategies leverage recurrent neural network techniques like Long Short Term Memory (LSTM) and Transformers to capture temporal dependencies and seasonality patterns in stock price data. Alternatively, there are ensemble learning techniques such as random forests and gradient boosting to enhance predictive accuracy and robustness by aggregating predictions from multiple models, mitigating weaknesses in individual algorithms, and providing more reliable predictions.

*Eddie Aljamal, Alex Janosi, Sisir Potluri, and Shashank Verma, are with the University of Michigan, Ann Arbor, MI 48109, USA ealjamal@umich.edu, janosi@umich.edu, sisir@umich.edu, shaaero@umich.edu

By developing more accurate and reliable predictive models, investors gain valuable insights into market trends and potential fluctuations, allowing them to make informed investment decisions with greater confidence. Furthermore, the refinement of these models can help identify lucrative investment opportunities and reduce risks associated with market volatility, ultimately leading to improved investment performance and portfolio management. Therefore, the advancement of research in this domain plays a crucial role in empowering investors to engage in the most effective and profitable investing strategies, thereby maximizing returns and minimizing losses in the stock market.

Since we have access to data from the "JPX Tokyo Stock Exchange Prediction Challenge" Kaggle Competition (JPX Challenge), we conduct our exploratory study in the context of that market. The Japanese market is one of the largest markets globally, and it offers insights into diverse economic, cultural, and geopolitical factors influencing stock prices. We anticipate that successfully exploring ML models and features for this challenge can lead to generalizable insights about ML's efficacy in stock index predictions, benefiting financial markets worldwide.

These predictions not only hold immense value for stock price forecasting and financial analysts, but they also extend their utility to a broader spectrum of time series problems. The identification of nuances in feature engineering and model selection is a general challenge in ML problems, so refinement of predictive models for stock price prediction serves as a foundation for advancing methodologies in overall time series analysis, benefiting an array of applications beyond financial markets.

The report is structured as follows: Section II outlines our research methodology and hypotheses, focusing on the role of PCA in feature engineering. Section III reviews related work and underscores the importance of feature engineering in stock prediction. Section IV details our experimental setup and model outcomes. Section V discusses our findings and their implications, addressing the hypotheses. Section VI concludes with remarks on future research directions and the broad applicability of our results. Finally, Section VII lists each author's contributions to the project.

II. Method

We denote "relevant components" as the set of PCA components obtained after executing PCA on the features we engineer (based on key financial indicators). Also, component subset and feature subsets may be used interchangeably, since the retained components become the features post-

PCA. We seek to evaluate the following three hypothesized claims:

1: Training models using every non-empty subset from the power set of relevant components will lead to the discovery of a best-performing model with a corresponding best PCA components subset for each of the LSTM, Transformer, XGBoost, and LASSO model types, and that model will have improved performance compared to the model in each type that uses the full set of engineered features.
2: The best component subset that results in the best-performing model in each of the LSTM, Transformer, XGBoost, and LASSO model types will not be the same among each of the four model types.
3: The feature weights assignment for LASSO and XGBoost will be similar to the features most correlated with the PCA components of the best-performing components subsets for these models.

A. Novel Approach with PCA

We propose an overarching pipeline to solve this problem and answer our hypotheses. Figure 1 represents an overview of the proposed pipeline, which we use to identify which specific subset of relevant components leads to the best performance in various models and compare the highest performance across models to derive insights about how they differ.

The JPX competition provides $2,332,531$ data points as input stock prices for 2000 security codes from April 1, 2017 to March 12, 2021. We split the data into two sets, Train and Test, and then generate a set of inputs with raw features and a set of labels for each. The input Train set is X_{train}, and the input Test set is X_{test}. The label Train set is Y_{train}, and the label Test set is Y_{test}. The label sets contain target stock prices, which we aim to predict in all attempted models, while the input sets contain a set of raw features. Using a function that performs feature engineering, we convert the raw input sets into sets with features that we deem interesting and relevant.

We then perform PCA on the full set of features to derive a smaller set of components that are most relevant — we retain the components that capture at least 99% of the variability. After this step, X_{train} and X_{test} both have components that correspond with the resulting components from PCA. We denote this set of components $\mathbb{D}$ and the power set of $\mathbb{D}$ as $P(\mathbb{D})$. A key aspect of the model fitting and prediction is that we can choose which column indices should be considered in both phases. We leverage this for the key aspect of our proposed method. For all non-empty sets of indices in the power set of indices of resulting components from PCA, we attempt each of the four model types, LSTM, Transformer, XGBoost, and LASSO. Therefore, for each non-empty set $s \in P(\mathbb{D})$, use the columns of X_{train} that intersect with s to fit each model and use the columns of X_{test} that intersect with the same s to make predictions from the corresponding model (the same components will be used in both cases). In this approach, there will be a total of $4 * (|P(\mathbb{D})| - 1)$ individual models that are fit and used for predictions, with $|P(\mathbb{D})| - 1$ for each of the four model types.

For each of the models within each model type, we compare their root-mean-square error (RMSE) and Sharpe Ratio (SR). We identify which subset s of components from the power set of the resulting components from PCA results in the highest SR for each model type. Considering only the LSTM model type, we denote the best model as LSTM-Opt and denote its feature set s'_{LSTM}, its RMSE as $\text{RMSE}'_{\text{LSTM}}$, and its SR as SR'_{LSTM}. We do the same for Transformer-Opt, XGBoost-Opt, and LASSO-Opt.

Figure 1 outlines this novel aspect of our study, which introduces a unique approach for attempting all subsets without the limitation of unbounded efficiency cost (since $\mathbb{D}$ contains a small number of PCA components compared to the 50 features). Apart from this step, we also find the performance metrics for a model in each model type that uses all the features we engineer, without PCA and without any selection of components subsets. We denote these models as LSTM-Full, Transformer-Full, XGBoost-Full, and LASSO-Full. For LSTM-Full, the corresponding metrics would be s_{LSTM}, $\text{RMSE}_{\text{LSTM}}$, and SR_{LSTM}, and it's the same for the other model types. Furthermore, we extract feature weights for the models in which that analysis is possible, and use these to compare with the most-represented features in the same models' best-performing PCA components subset.

B. Evaluation

We summarize the inputs, outputs, and evaluation metrics used across models.

If we deem n components after PCA capture 99% of the data variance, then each of X_{train} and X_{test} will have between 1 and n PCA components as feature columns for 2000 security codes at various dates, while Y_{train} and Y_{test} will have the target prices (labels) for the same entries. We note that this represents a continuous, supervised learning problem.

The equation for the root-mean-squared-error (RMSE) is computed as follows, where y is the predicted prices for Test data and $\hat{y}$ is the actual prices for Test data:

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}} \tag{1}$$

The equation for the Sharpe Ratio (SR) is computed as follows, where R is the overall return from a portfolio for one day after the top-ranked 200 stocks out of 2000 (by predicted gains) are bought, and the bottom-ranked 200 stocks are shorted:

$$\text{SR} = \frac{\text{Average of } R}{\text{Standard Deviation of } R} \tag{2}$$

To answer Hypothesis 1, we will compare the RMSE and SR metrics of LSTM-Opt, Transformer-Opt, XGBoost-Opt, and LASSO-Opt with those of LSTM-Full, Transformer-Full, XGBoost-Full, and LASSO-Full.

To answer Hypothesis 2, we will compare the s' sets of each of LSTM-Opt, Transformer-Opt, XGBoost-Opt, and

LASSO-Opt. We expect this proposed method to answer our claims because we perform the study on a large variety of models (4 model types, each with $|P(\mathbb{D})| - 1$ different subsets).

To answer Hypothesis 3, we will compare 1) the features with the highest weights that comprise the PCA components of the best-performing feature subset with 2) the feature weights assigned by the algorithm - either LASSO and XGBoost - when the full feature set is used.

C. Feature Representation

We have created a method to seamlessly integrate new features. Based on the data and the results of the related work, we have determined strong indicators that have been used in some of the most successful models. We used basic indicators like the daily percentage change, 52 week high, open to close price change, and the daily amplitude. However, with more technical indicators that are commonly used in stock trading, we wanted to see the impact of averages over different times periods. Through experimentation, we found some of the best time periods to analyze were 5, 10, 20, 40, and 60 day periods. With these periods, we calculated moving averages, exponential moving averages, and volatility over. We then integrated important technical indicators:

$$RSI_x = \frac{\text{Average Gain During Up Period } x}{\text{Average Loss During Down Period } x} \quad (3)$$

$$BIAS_x = \frac{\text{Closing Price} - x \text{ Day Average Price}}{x \text{ Day Average Price}} \quad (4)$$

$$MACD_x = \text{EMA } x - \text{EMA } 2x \quad (5)$$

$$\text{Exp. Div.}_x = I\left[\frac{\text{Exp. Dividend}}{\text{Closing Price}} > \frac{x}{500}\right] \quad (6)$$

RSI, BIAS, and MACD are some of the most commonly used technical indicators in stock trading. As proposed in our related works, we say that the competition definition of the expected dividend lined up target, and we wanted to explore its impact. Through our background in finance, we agreed that dividends can cause a large swing in stock price. For example, if a company is expecting a large dividend compared to its price, stock owners might wait to collect the dividend and then sell immediately. Therefore, we calculated this feature as a percentage based on the dividend value over the closing price. We also considered a baseline for expected dividends where $x = 0$, and we used this as a feature for the models.

Feature engineering in the domain of stock price prediction is a deeply complicated task that is susceptible to the specific models, parameters, and existing financial trends that influence the training or testing data. Therefore, it is difficult to clearly define a set of features that leads to improvement across all four models, but it is certainly valuable to analyze the specific subsets of components that lead to the best-performing models in each of the four model types as well as the feature weight assignments in some models. We expect to learn important insights about the potential for certain features to appear in the best feature sets in all model types, and we can also learn which features seem least reliable, suggesting that they might have only been relevant out of chance or from inherent noise in the data. We are aware that the choices we make during feature engineering itself will have a profound impact on the subsets that we generate, so we can only produce meaningful insights within the context of the features that we choose to engineer. Nevertheless, learning how these models differ in stock price prediction performance based on the specific component subsets we feed to them is a useful topic to learn, and we can understand if certain model types are more robust to changes in the component subsets they are given versus others, which can inform future work in the use of machine learning for stock price predictions.

III. Related Work

A. Feature Engineering and Selection

The primary objective of feature engineering for stock prediction revolves around the creation of unique and effective financial-indicators. A problem is that all information is made public regarding the trade and movement of a given stock, so the playing field is generally even at the beginning. Feature engineering is the only differentiating factor for models that can become very similar. With focus on the competition, the project data has been provided by Kaggle, so the only way to get an advantage is to create new features. However, creating the most effective features is a task that is still difficult to perfect.

A recent survey addressed feature engineering challenges in the stock market [1], summarizing over 32 approaches to feature engineering for market-related tasks. The survey highlighted the complexity of selecting an effective feature set, noting that each method prioritized different features, though all included basic data like closing stock prices. Our work aims to distinguish itself from these studies by focusing on financial information critical for the Kaggle competition and applying a comprehensive subset of engineered features. We find expected dividends particularly valuable, as this data is recorded two business days before the ex-dividend date. The competition outlines each feature's role, linking this feature to the target column defined by the change ratio of the adjusted closing price from $t+2$ to $t+1$, where t is the trade date.

The main findings of the survey were the effectiveness of correlation criteria in reducing the dimensionality of the created features. Most approaches revolved around principal component analysis (PCA). A different article states the importance of PCA in stock prediction due to its ability to help reduce noisy financial time series data [2]. When compared with other methods, PCA performed the best due to its ability to eliminate correlated data, which is valuable in use cases with a significant amount of time-related data. This article also states the importance of moving averages. Technical traders use moving averages to identify price momentum without autocorrelation issues. Additionally, stock data is noisy and volatile, so moving averages can smooth out the

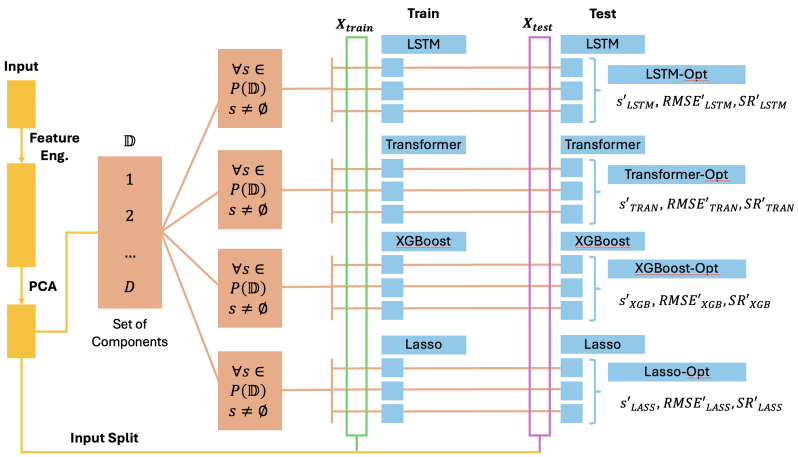

Fig. 1: Diagram of Proposed Pipeline

trends without losing information. We use moving averages as a feature and as a basis for building stronger technical indicators.

B. LSTM

The Long Short-Term Memory (LSTM) is a version of recurrent neural networks (RNNs) that capture and model sequential data with long-term dependencies, making it a robust choice for stock price predictions [3]. However, it is a fairly outdated method, with newer deep learning methods outperforming LSTM.

One recent study [4] used the LSTM method to predict stock prices and found that it resulted in an enhanced prediction accuracy compared to conventional machine learning algorithms, but the article was ambiguous about which exact algorithms LSTM outperformed. A key drawback of this study, along with many articles regarding stock price predictions, is that they rely on the stocks of a few companies. This study includes only four companies, all of which are in the technology or gaming sector. Another recent article [3] addressed a similar topic, but it was far more specific in its analysis by combining results from studies that assess the performance of various machine learning algorithms in stock price prediction, but Transformer, XGBoost, and LASSO were not a part of this analysis. Though the study shared that LSTM was the most accurate compared to SVM, KNN, Logistic Regression, and others, based on existing articles in this domain, it doesn't perform its own experiment and instead is only a systemic literature review, so myriad variables could cause differences in the performance statistics between ML models.

While these articles illustrate the capacity of the LSTM technique in stock price predictions, several of them also indicate that future research is recommended to help find new techniques for feature engineering and feature selection along with further study about the relative impact of various features [5]. This is where our study introduces novelty. Instead of tuning an LSTM-based model to help predict stock prices for data regarding a few large companies, we attempt to use LSTM for a dataset from the larger JPX market, which features 2000 security codes. Furthermore, the key component of our proposed method is finding how different subsets of relevant components cause changes in the performance of LSTM models and how those changes compare between LSTM and other techniques, such as Transformer and XGBoost. Therefore, our study places greater emphasis on feature exploration, PCA, and subset comparison than the potential for LSTM to predict the prices of a limited set of stocks, like most existing research.

C. Transformer

The Transformer model [6] represents a pivotal shift away from the traditional reliance on RNNs and convolutional neural networks (CNNs) for sequence transduction tasks. Characterized by its exclusive reliance on attention mechanisms, the Transformer architecture eliminates the need for recurrence and convolutions, facilitating a significant increase in the parallelization of training processes and a reduction in training times.

Central to the Transformer's design is its capacity to model long-range dependencies within data, a feature that has proven to be particularly beneficial for a broad spectrum of NLP tasks. These include, but are not limited to, language understanding, generation, translation, question answering, and text summarization. The architecture's foundational principles and novel contributions have catalyzed considerable advancements across these various domains, demonstrating the model's versatility and efficacy.

In addition to its successes in NLP, the Transformer's ability to capture intricate dependencies and interactions has also rendered it an attractive option for time series modeling. Recent studies [7], [8] have explored the application of the Transformer model to time series forecasting, highlighting its potential in this field. Notably, a recent work [9] expanded upon this application by integrating CNNs with Transformers to model both short-term and long-term dependencies within financial time series. Their approach aimed to forecast the future direction of stock prices (up, down, or flat) for the constituents of the S&P 500 index. The methodology demonstrated superior performance when

benchmarked against traditional statistical methods and other deep learning approaches commonly employed in this area.

However, it should be noted that despite the promising results, the study [9] faces substantial shortcomings in its presentation and clarity, particularly in the detailed explanation of its methodology. These issues underscore the necessity for more thorough documentation and replication studies to fully appreciate and utilize the potential of these hybrid models in financial forecasting and other applications. We plan to use the Transformer model alongside LSTM, LASSO, and XGBoost in the upcoming Kaggle competition for long-sequence time-series forecasting (LSTF).

D. Gradient Boosted Trees (XGBoost)

Boosted regression trees is a method that sequentially combines the predictions of weekly learning trees. More specifically, each new tree fits the residuals of the previous trees, and then this tree is shrunk to make it a weak learner, under the assumption that learning slowly achieves high generalization accuracy because of reduced overfitting. More specifically, XGBoost calculates a similarity score for each tree node and splits this node based on the highest gain of the similarity score which is the difference between the total similarity score of the child nodes and the similarity of score of the root node [gain = similarity(left) + similarity(right) - similarity(root)]. This provides a measure of the importance of a feature based on the value of the gain.

Gradient boosted Trees using the XGBoost algorithm were used to obtain very accurate predictions of the moving averages of various stocks and options of 800 different companies in the Karachi Stock Exchange within a 5-year span using the the features: closing price, open price, low price, high price [10]. Such a work motivated our choice of using the XGBoost algorithm, but we introduce novelty by predicting time-indexed prices for the much-larger JPX market and giving greater focus to feature subset exploration.

E. LASSO

LASSO is a linear regression model with L^1 penalty controlled by the regularization parameter λ. An important property of the L^1 penalty is that it sets the weights of unimportant features to zero and thereby selecting important features that contain a high amount of information. The feature selection property of LASSO regression has been used to derive an optimum set of features in market data, as shown in a study [11], and these properties were then fed into an LSTM pipeline. Instead, we use LASSO regression directly on time-indexed stock prices to give us a benchmark for the performance of LASSO as compared to the other models and see if the feature selection property of LASSO leads to similar outcomes as our methodology, where we attempt all subsets of relevant features.

IV. Experimental Results

A. Implementation

Data: Following the data source indicated in our method, we used a dataset containing historic data for 2000 Japanese stocks and options provided by the Japan Exchange Group on Kaggle under the competition: "JPX Tokyo Stock Exchange Prediction". We split the data into a Train Set containing entries with dates prior to 2020-01-01 and a Test Set containing entries with dates after 2020-01-06.

Features: Based on various financial indicators, we used 50 different features (see the Appendix, section VIII).

Implementation: Our implementation of the LSTM model relied on `keras`, where we imported the `Sequential` module. Our model was simple, with an LSTM layer with 64 units and an input shape corresponding to the dimensionality of our input sets. We included a Dropout layer with a 0.4 rate and a Dense layer with 1 neuron. We selected these parameters after attempting LSTM models in various leaderboard notebooks and found that the simpler structure worked well. Further parameter tuning is required, and K-Fold cross-validation is needed for this (it is implemented for the other models in our preliminary results).

To implement the gradient boosted trees model, we used the popular `XGBoost` Python library. We performed 5-fold cross-validation to test the performance of the following hyperparameters: number of regression trees (100 or 200), maximum of depth of trees (2, 4, 6, 8, 10), and learning rate (0.0001, 0.001, 0.01, 0.1, 1). We found that the best-performing hyperparameters for the entire set of features is the same as the best-performing hyperparameters for all subsets of the PCA components (which have the same set of best-performing hyperparameters) which are given by the following: The number of regression trees is 200, the maximum depth is 6, and the learning rate is 0.01.

For LASSO regression, we used `scikit-learn` Python package to perform 5-fold cross-validation on the regularization parameter in the range 10^{-8} to 1 in 100 bins. The best-performing regularization parameter for LASSO for the entire dataset was very small 9.66×10^{-8}.

For the Transformer model, we utilized a custom architecture implemented in `TensorFlow` with the `keras` API. The network architecture consists of several key layers detailed below:

- `InputLayer` serves as the entry point for data.
- `Dense` layer with 128 units to transform the input dimensionality.
- `LayerNormalization` to normalize the activations and speed up training.
- Multiple `TransformerBlock` layers, each with 99,584 parameters. We used four such blocks to capture dependencies sufficiently.
- `GlobalAveragePooling1D` to reduce the dimensionality of the output from the transformer blocks.
- Two `Dense` layers at the end, where the first has 20 units, and the final layer has 1 unit for regression output.

The total parameters for this model are 407,593, with all being trainable. This architecture aims to leverage the Transformer's capability to handle sequential data effectively, providing the framework for predicting stock price movements.

B. Experimental Outcomes

After running the models on both the entire feature set and subsets of the components resulting from PCA, we present the test set Sharpe Ratios in Table I.

Once we had our feature set, we performed Minka's PCA [12], which uses a probabilistic approach to determine the feature reduction size. With this, we were able to cover 99% of the variance of the original feature set with three components, as shown in Figure 5.

V. Discussion

We assess the PCA results, metrics and feature importances across the four models, and address our motivating hypotheses.

A. Features and PCA

We can see that our results proved that each of the different indicators had relevance during different time periods. Therefore, covering all the periods and indicators proved worthy, and the dimensionality reduction can eliminate the remaining noise. We also note that the expected dividend for a period of 5 days was a main factor in the primary component, so our hypothesis of this being an important feature was correct.

B. Gradient Boosted Trees (XGBoost)

From Table I, we see that gradient boosted trees using the XGBoost algorithm achieves the highest Sharpe Ratio out of all the models used on the Japenese market dataset. The subset with only the first (0th) and second (1th) PCA components, $\boldsymbol{S_{\{0,1\}}}$ performs best with a SR of 0.133 while the subset with the three PCA components $\boldsymbol{S_{\{0,1,2\}}}$ is close behind with a SR of 0.132 showing some signs of poor performance and overfitting when adding the third (2th) component - as confirmed by the low SR score of the model only using the third component, $\boldsymbol{S_{\{2\}}}$. Furthermore, this shows that the first and second principal components contain most of, if not all, the information responsible for the impressive performance of gradient boosted trees.

However, these relatively high SRs using the aforementioned PCA subsets are closely matched to $\sim 0.7\%$ by the gradient boosted trees model using the entire feature set of 50 features (listed in the Appendix, section VIII) which achieves a SR of 0.132. This is not an altogether surprising result because gradient boosted trees is an algorithm that assesses the importance of different features and performs branch splitting based on this assessment, as explained in section III-D. This is a very similar process of feature selection to PCA and therefore test results for XGBoost without PCA are very similar to test results of XGBoost with PCA.

This analysis raises an interesting question: are the features selected by the first two components of PCA - the best performing subset for XGBoost - similar to the most important components calculated by XGBoost internally? To answer this question, we compare the important features in XGBoost to the features contributing to the first principal components. From Figure 3, we see that first two principal components (0 and 1) are primarily explained by the same

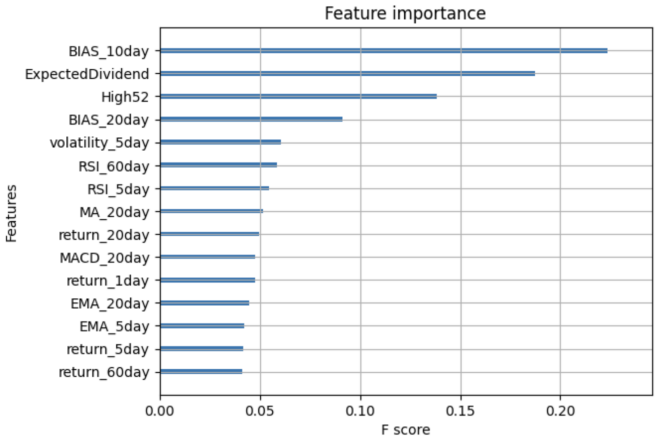

Fig. 2: Importance of features based on values of gain - obtained during the shrinkage process to create weak learners - assigned to each feature. Only the fifteen most important features are shown.

set of features, which are Open, Low, 10-day bias, 20-day bias, MACD, 60-day MACD, 40-day MACD, 5-day MACD, 52-week high, 40-day EMA, 5-day EMA, 5-day expected dividend, 5-day return, and 20-day RSI. Similarly, the most important features from XGBoost are (in descending order of importance): 5-day bias, 60-day return, 1-day return, 52-week high, 10-day return, amplitude, 20-day return, 5-day RSI, 20-day bias, 20-day MACD, expected dividend, 5-day volatility, 40-day return, and 5-day return. Only *five features* $\sim 36\%$ are shared between these two sets of fourteen features namely, 10-day bias, 20-day bias, 52-week high, 5-day EMA, and 5-day return. This may be misleading, however, as quantities taken over different periods of time (for example 10-day return and 5-day return) are highly correlated and therefore contain similar information. We note that open low are very important features in PCA, but do not appear in the most important fourteen features of XGB. Additionally, volatility, which is the 5th most important feature in XGBoost does not appear in PCA even though it is an essential ingredient in risk assessment of a stock and hence contributes to the Sharpe Ratio. All other features which is $\sim 86\%$ of the fourteen - ignoring disparate periods of time - occur in both the sets of important features given by PCA and XGBoost.

Unlike transformers and LASSO, our implementation of the gradient boosted trees does not have a negative Sharpe Ratio - which means that the model's projected return is lower than that of the benchmark - for any of the PCA subsets or the full-feature set model.

C. LASSO

Since LASSO is a primitive linear regression model with a L^1 penalty, we used this as a benchmark algorithm to which we compare gradient boosted trees, LSTM, and transformers. From Table I, we observe that LASSO achieves the highest SR of 0.126 for a subset containing only the second (1) principal component, $\boldsymbol{S_{\{1\}}}$. Therefore, the best performance of LASSO is only lower than that of gradient boosted trees and is better than both LSTM and transformer. The subset with the worst performing SR is the subset containing

SR using Subsets	LASSO	XGB	LSTM	Transformer
$S_{\{0,1,2\}}$	-0.036	0.132	0.090	0.007
$S_{\{0,1\}}$	0.059	0.133	0.119	0.089
$S_{\{0,2\}}$	-0.029	0.050	0.034	0.002
$S_{\{1,2\}}$	0.040	0.085	0.106	-0.015
$S_{\{0\}}$	0.056	0.029	0.034	0.024
$S_{\{1\}}$	0.126	0.064	0.106	0.027
$S_{\{2\}}$	-0.086	0.009	0.003	0.013
Best w/o PCA	**0.116**	**0.132**	**0.054**	**0.041**
Best w/ PCA	**0.126**	**0.133**	**0.119**	**0.089**

TABLE I: Test Set Sharpe ratios for the different models used for subsets of the first three PCA components. $S_{\{0,1,2\}}$ denotes the training with the set of features containing the 0 (first), 1 (second), and 2 (third) PCA components. Results w/o PCA are from using the full feature set. The second to last row title Best w/o PCA gives the our best SR achieved using the entire feature set (with 50 total features) for each model. The last row gives a summary for the best SR score after performing PCA.

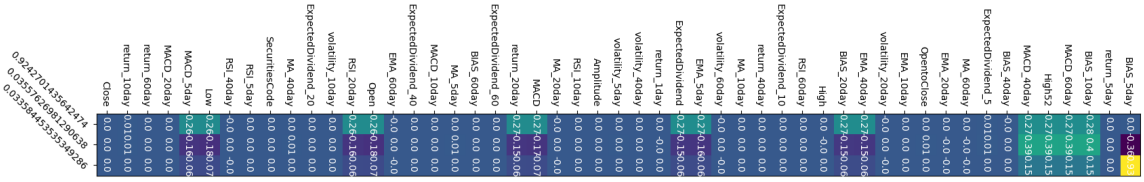

Fig. 3: Feature weights of the first three principal components.

only the third principal component $S_{\{2\}}$, showing that the second principal component displays poor performance in agreement with the analysis using gradient boosted trees. Unlike gradient boosted trees combining the first and second principal components, $S_{\{0,1\}}$, for LASSO performs a factor of two worst at a SR of 0.059, than a model using the first component alone.

Similar to gradient boosted trees, the high SR achieved by LASSO after performing PCA is only $\sim 7\%$ higher than the LASSO model on the entire 50 feature set. We attribute this, again, to the feature selection property of LASSO and we compare the importance scores of the LASSO features in Figure 4 (before PCA) to the features that have high weights in the second principal component (the one with the highest SR for LASSO) in Figure 3. We observe that the most important features given by LASSO are (in descending order of importance): 5-day bias, 40-day volatility, 20-day-volatility, 1-day return, 10-day bias, 10-day volatility, 40-day bias, expected dividend, 20-day bias, 20-day return, 60-day bias, 10-day return, 5-day return, 40-day return, and 60-day return. The features that have the highest weights contributing to the third principal (2) component are: expected dividend, open, MACD, 10-day MACD, 60-day RSI, 5-day EMA, 60-day EMA, 5-day return, 5-day volatility, high, 40-day bias, and 20-day bias. There are *four features*, $\sim 29\%$, that belong to both of these sets namely, expected dividend, 20-day bias, 40-day bias, and 5-day return. Ignoring disparate time periods for these features, MACD, EMA, RSI, open, and high are not considered important by LASSO while all such features that are considered important by PCA are also considered important by LASSO. Thus, ignoring the time periods, $\sim 71\%$ of features are common to both sets.

LASSO gets negative Sharpe Ratios for three subsets of the principal components: the subset including all three principal components, $S_{\{0,1,2\}}$, with a SR of -0.036, the subset with the first and third principal components, $S_{\{2\}}$, with a SR of -0.029, and the subset with the only the third principal component, $S_{\{3\}}$, with an SR of -0.086.

Weight	Feature
+0.030	BIAS_5day
+0.025	volatility_20day
+0.022	volatility_10day
+0.007	BIAS_20day
+0.003	return_20day
+0.003	BIAS_60day
+0.002	return_5day
+0.001	return_40day
+0.001	return_60day
	... 13 more positive ...
	... 17 more negative ...
-0.003	return_10day
-0.009	ExpectedDividend
-0.015	BIAS_40day
-0.023	BIAS_10day
-0.023	return_1day
-0.030	volatility_40day

Fig. 4: Weights on features given by LASSO weights. The higher the absolute value of the weights means that the feature is more important. Only the fifteen most important features are shown.

D. LSTM

When using LSTM for stock price predictions on the Japanese market dataset, we find that the results using all subsets of components resulting from PCA performs better than training with the full feature set. The improvement of the novel method for LSTM is far more pronounced than the difference in other models between using the full feature set versus all subsets of components. Specifically, the LSTM model's Sharpe Ratio for components 0 and 1 is the best at 0.119, indicating these two PCA components are most predictive of accurate target stock prices using the LSTM model compared to other combinations of components. Since LSTM is a neural network, there isn't a reliable method to access the feature weights when executing on the full feature set, although one method exists, which relies on permutations. However, that method is only robust when there aren't correlations between features in the data, which we definitely cannot claim for the set of features we've engineered.

Component 1 also has 5-day bias as a key feature, but this is not the case for component 0. Looking at the Sharpe Ratio when only one of component 0 or 1 is used as the feature subset, we see that the subset with 1 alone has a much better result. Therefore, we can see that that 5-day bias feature does have a strong impact on the predictive capacity when the LSTM model is employed, and it may be more significant than each the features that are explained in both components. This might be explained by the fact that the 5-day bias feature helps the model capture short-term movements more effectively without being as affected by noise in the day-by-day data. Since LSTM is meant for sequential time-based data, it's logical that many of the time-based features explain the variance in the PCA components which comprise the best-performing subset on LSTM.

Overall, LSTM without PCA is a much weaker predicive model compared to LASSO, XGBoost, and it doesn't perform much better than Transformers. Furthermore, while PCA-based subsets selection does improve the metrics significantly, LSTM is still worse than LASSO and XGBoost for predictions.

E. Transformer

In this study, we focus on the performance of the Transformer model for predicting stock market prices using a dataset from the Japanese stock market. Table I displays varying performance levels across different subsets of PCA components. The highest Sharpe Ratio (SR) achieved by the Transformer using PCA components was 0.089 for the subset $\boldsymbol{S_{\{0,1\}}}$, which is comparatively lower than the best performing models such as XGBoost (0.133) and LASSO (0.126). This outcome indicates that while Transformer models excel in domains like natural language processing due to their proficiency in handling sequential data, their application in stock price forecasting might not be as reliable as models more specifically tailored to this domain.

The performance of different subsets suggests that the Transformer is particularly sensitive to feature selection, with certain combinations like $\boldsymbol{S_{\{0,1\}}}$ yielding better SRs than others. This supports our first hypothesis. It is important to note that using all three PCA components $\boldsymbol{S_{\{0,1,2\}}}$ resulted in a significantly lower SR (0.007), indicating potential overfitting or the inclusion of less relevant information for the model. When trained on the full set of features (without PCA), the Transformer model achieved an SR of 0.041, which is lower than the best SR obtained with the PCA-based subsets. This highlights the advantages of dimensionality reduction through PCA, which aids in isolating more impactful features, thereby enhancing the model's performance and thus confirming our second hypothesis.

The best-performing subset, $\boldsymbol{S_{\{0,1\}}}$, likely included features that were more predictive of stock movements, aligned with the model's strengths in managing sequences and long-range dependencies. Furthermore, the Transformer model's attention mechanisms are adept at focusing on significant patterns represented in the first two components, minimizing distractions from less informative variables. Future research could explore enhancements to the Transformer architecture, possibly by integrating techniques like attention with convolution or different forms of sequence processing, to more effectively capture the dynamics of financial time series.

F. Analysis

1. Hypothesis 1: We trained our machine learning algorithms on the power set of three principal components for a full set of 50 features, and arrived at the best-performing subset for each model. In all these approaches, the performance after PCA was better than the performance on the entire feature set, although for both LASSO and gradient boosted trees, the margin between the two was very small.
2. Hypothesis 2: Except for LASSO, which achieved its best performance using a subset that included only the second principal component, we observe that the best-performing subset (highest Sharpe Ratio) for XGBoost, LSTM, and the transformer was the subset containing the first and second principal components.
3. Hypothesis 3: Finally, we saw that the important features given by LASSO only constitute $\sim 29\%$ of the features that contribute to the best performing PCA subset using LASSO, while the important features given by XGBoost amount to $\sim 36\%$ of the features that contribute to the best-performing XGBoost with PCA subset. If we ignore the time periods over which these features were calculated, these percentages increase to $\sim 71\%$ and $\sim 86\%$ for LASSO and XGBoost, respectively.

VI. Conclusion

This study demonstrated the effectiveness of using a power set of PCA components for predictive modeling in stock price forecasting, revealing that strategic feature engineering significantly enhances model performance. The XGBoost model, in particular, showed superior performance by achieving the highest Sharpe Ratio, underscoring its capability in handling complex datasets from the Japanese stock market. Our findings confirm the importance of precise feature selection, which not only simplifies computational complexity but also improves predictive accuracy. This approach has proven to be more effective than using full feature sets, indicating its potential applicability in other financial analytics domains. Furthermore, the research validated two central hypotheses: that training on non-empty subsets of PCA components identifies optimally performing models, and that the best-performing feature subset varies across different models. This underscores the nuanced interaction between machine learning models and feature subsets. For future work, the insights gained from our PCA-based strategies could inspire further research into feature selection and model optimization, enhancing the development of more reliable and effective predictive tools for financial markets. This study highlights the transformative potential of methodical feature analysis and selection in improving financial forecasting accuracy.

VII. Author Contributions

We present a summary of the authors' contributions:

- Eddie Aljamal implemented cross-validated linear and LASSO regression as well as cross-validated gradient boosted regression trees to obtain Sharpe ratios.
- Alex Janosi focused on feature engineering and selection. This involved creating feature creation for the model pipeline. This also included running PCA and doing analysis of important features.
- Sisir Potluri implemented the base notebook for the LSTM model, which was used for the other models too, wrote the proposed method, implemented the feature-subset selection from PCA components, and was in charge of the LSTM model.
- Shashank Verma implemented and wrote custom architecture implemented in `TensorFlow` for time-series forecasting used for stock price prediction using Transformers.

All co-authors were involved in writing this report. All co-authors equally contributed to this project.

References

[1] H. H. Htun, M. Biehl, and N. Petkov, "Survey of feature selection and extraction techniques for stock market prediction," 2023.

[2] M. Ghorbani, "Stock price prediction using principal components," 2020.

[3] N. Lumoring, D. Chandra, and A. A. S. Gunawan, "A systematic literature review: Forecasting stock price using machine learning approach," pp. 129–133, 2023.

[4] V. K. Polepally, N. R. Jakka, P. Vishnukanth, R. S. Raj, and G. Anish, "An efficient way to predict stock price using lstm-rnn algorithm," pp. 1240–1244, 2023.

[5] V. M. De Borba, "Hyperparameters and features impacts in artificial intelligence applied to stock market," pp. 231–235, 2022.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[7] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," *ArXiv*, vol. abs/2001.08317, 2020.

[8] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?," 2022.

[9] Z. Zeng, R. Kaur, S. Siddagangappa, S. Rahimi, T. Balch, and M. Veloso, "Financial time series forecasting using cnn and transformer," 2023.

[10] N. Hussain, A. Qasim, Z.-u.-d. Akhtar, A. Qasim, G. Mehak, L. del Socorro Espindola Ulibarri, O. Kolesnikova, and A. Gelbukh, "Stock market performance analytics using xgboost," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14391 LNAI, p. 3 – 16, 2024.

[11] X. Chen, L. Cao, Z. Cao, and H. Zhang, "A multi-feature stock price prediction model based on multi-feature calculation, lasso feature selection, and ca-lstm network," *Connection Science*, vol. 36, no. 1, 2024.

[12] T. Minka, "Automatic choice of dimensionality for pca," 2000.

VIII. Appendix

A. Features

The list of features are (in no particular order):

1. High
2. Low
3. Open
4. Close
5. Open to close
6. Amplitude
7. 52-week high
8. Expected dividend
9. 5-day expected dividend
10. 10-day expected dividend
11. 20-day expected dividend
12. 40-day expected dividend
13. 60-day expected dividend
14. 1-day return
15. 5-day return
16. 10-day return
17. 20-day return
18. 40-day return
19. 60-day return
20. 5-day volatility
21. 10-day volatility
22. 20-day volatility
23. 40-day volatility
24. 60-day volatility
25. 5-day bias
26. 10-day bias
27. 20-day bias
28. 40-day bias
29. 60-day bias
30. 5-day moving average
31. 10-day moving average
32. 20-day moving average
33. 40-day moving average
34. 60-day moving average
35. 5-day exponential moving average
36. 10-day exponential moving average
37. 20-day exponential moving average
38. 40-day exponential moving average
39. 60-day exponential moving average
40. 5-day Relative Strength Index (RSI)
41. 10-day RSI
42. 20-day RSI
43. 40-day RSI
44. 60-day RSI
45. Moving Average Convergence/Divergence (MACD)
46. 5-day MACD
47. 10-day MACD
48. 20-day MACD
49. 40-day MACD
50. 60-day MACD

B. PCA Explained Variance Plot

The y-axis represents all the features used in this study and the x-axis columns represent each of the PCs i.e. the first column represents the first PC (0) and the squares to the right of each feature represents the weight of that feature in the first PC. This is the same for the second column being the second PC (1) and so on. The values on the x-axis show the ratio of the total variance explained by each PC.

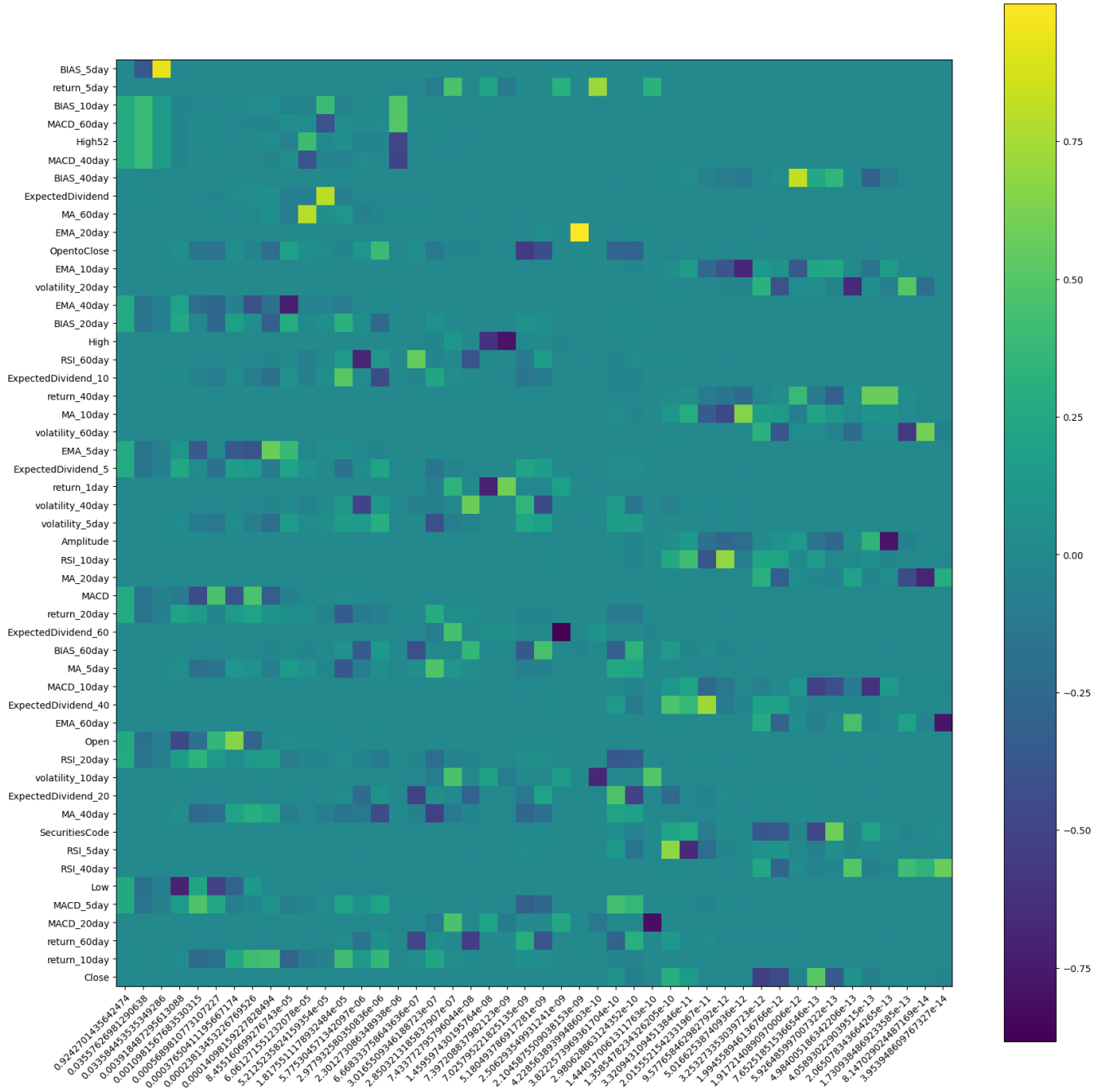

Fig. 5: PCA components (x-axis) with ratio of total variance explain and the features (y-axis). The weights for a feature is given by the color of the square for each PC.