Final Project Report: AI Model Comparison

Course: Introduction to AI  
Project Title: Stock Price Predictor  
Submitted by:

* Cisco Dacanay
* Zach Greenhill

Date: 12/11/24

# 1. Introduction and Objective

**Objective:**The objective of this project is to create several different predictive models for stock price forecasting to identify the most accurate approach for predicting future stock prices based on historical market data and financial statements.

**Problem Statement:**

Accurately predicting stock prices can be difficult and inaccurate. Most prediction techniques focus on short-term predictions. Our goal is to try to predict the value of a given stock a month in the future. We will explore several different regression techniques to find methods to try to accurately predict stock prices.

**Overview of AI Models Chosen:**

| **Model Number** | **Model Name** | **Purpose** |
| --- | --- | --- |
| 1 | XGBoost | Uses gradient boosting to create an ensemble of decision trees. |
| 2 | Linear Regression | Fits a linear equation to predict values. |
| 3 | Neural Network | Used to find complex relationships in multi-dimensional data |
| 4 | Support Vector Regression (SVR) | Attempts to find a hyperplane that generalizes data. |
| 5 | Random Forest | Creates an ensemble of decision trees with bagging to get an average value. |

# 2. Justification of Model Selection

**Justification for Model Selection:**

| **Model Name** | **Reason for Selection** |
| --- | --- |
| XGBoost | Required |
| Linear Regression | Required |
| Neural Network | Required |
| Support Vector Regression | Good for regression tasks and avoids overfitting by attempting to create an optimal hyperplane. This model also requires few hyperparameters. |
| Random Forest | Good for tabular data and avoids overfitting by creating uncorrelated trees with randomly selected feature subsets. |

# 3. Model Descriptions

**Model Overview:**

| **Model Number** | **Model Name** | **Architecture Details** | **Key Features** |
| --- | --- | --- | --- |
| 1 | XGBoost | Ensemble decision tree model, boosting with gradient descent | Regularization |
| 2 | Linear Regression | Linear model | Linearity |
| 3 | Neural Network | Fully connected network with 2 hidden layers | Nonlinearity, feature representations |
| 4 | Support Vector Regression | Hyperplane/high dimensionality mapping | Regularization, nonlinearity |
| 5 | Random Forest | Ensemble decision tree model, bagging with uncorrelated trees | Randomized feature subsets |

# 4. Dataset Description

**Dataset Information:**

| **Name** | New York Stock Exchange |
| --- | --- |
| **Source** | Available from <https://www.kaggle.com/datasets/dgawlik/nyse>  (fundamentals.csv, prices-split-adjusted.csv) |
| **Size** | 1318 data rows after preprocessing (922 for training, 263 for validation, 133 for testing) |
| **Class Distribution** | n/a |
| **Preprocessing Steps** | 1. Select 2 consecutive years of financial statement reports. 2. Select 50 evenly spread price data points from the date of the 1st report to 30 days before the 2nd report. 3. Use the price on the day of the 2nd report as the output. 4. Normalize input features. |

**Dataset Justification:**  
This dataset is suitable for the models because it contains stock price data that can be turned into a regression problem by supplying price data during a given period and then taking the price 1 month afterwards as the output. Furthermore, it contains financial statements for each company/ticker symbol that can be used as additional features to enhance the models’ predictions.

# 5. Experimental Setup

**Experimental Design:**

| **Metric** |
| --- |
| Mean Squared Error |
| Root Mean Squared Error |
| Mean Absolute Error |
| R2 Score |
| Mean Absolute Percentage Error |

**Parameter Settings:**

| **Model Name** | **Learning Rate** | **Epochs** | **Optional Hyperparameter** | **Optional Hyperparameter** | **Additional Details** |
| --- | --- | --- | --- | --- | --- |
| XGB | 0.1 | 100 | Max Depth = 6 |  |  |
| Linear Regression | 0.001 | 120 | Batch Size = 128 |  |  |
| Neural Network | 0.0005 | 30 | Batch Size = 128 | 2 hidden layers | * 128 nodes at first hidden layer * 64 nodes at second hidden layer |
| SVR | N/A | N/A | C (Regularization Parameter) = 1.0 | Epsilon = 0.1 | Using a Linear kernel |
| Random Forest | N/A | N/A | Max Depth = inf | Number of Estimators = 100 |  |

**Environment Details:**

| **Component** | **Specification** |
| --- | --- |
| Operating System | Windows 11 |
| Software Version | Python v\_3.12.1  PyTorch v\_2.5.1  Sklearn v\_1.5.1  XGBoost v\_2.1.3 |
| Hardware | NVIDIA GeForce RTX 4060 GPU |
| Link to the code base | https://github.com/GreenhillZachary/CS482\_Final\_Project |

# 6. Results and Analysis

**Performance Metrics on Test Dataset:**

| **Model Name** | **Mean Squared Error (MSE)** | **Root MSE** | **Mean Absolute Error** | **R2 Score** | **Mean Absolute Percentage Error** |
| --- | --- | --- | --- | --- | --- |
| Neural Network | 1088.0126 | 32.9850 | 24.4299 | 0.5604 | 0.4317 |
| Linear Regression | 310.3066 | 17.6155 | 14.3498 | 0.8746 | 0.2749 |
| SVR | 62.6183 | 7.9132 | 4.1178 | 0.9747 | 0.0749 |
| XGBoost | 38.2049 | 6.1810 | 3.3770 | 0.9846 | 0.0521 |
| Random Forest | 31.9539 | 5.6528 | 3.2857 | 0.9871 | 0.0499 |

**Comparative Analysis:**  
In our project, we evaluated multiple models on the same testing dataset using the following metrics: Mean Squared Error (MSE), Root Mean Squared Error (Root MSE), Mean Absolute Error (MAE), R² Score, and Mean Absolute Percentage Error (MAPE). These metrics provided a comprehensive assessment of the accuracy and robustness of each model's predictions.

Among all the models evaluated, the tree-based models, particularly Random Forest and XGBoost, demonstrated the best performance. The Random Forest model achieved the lowest MSE, Root MSE, MAE, and MAPE, with an R² Score of 0.9871, indicating that it captured nearly all the variance in the data. The XGBoost model performed almost as well, slightly behind Random Forest, showing that it too effectively balanced bias and variance while capturing the data's structure. The strong performance of these models suggests that while the data may be largely linear, there are slight non-linear patterns that these models can effectively capture due to their ensemble and boosting capabilities.

The Support Vector Regression (SVR) model also performed strongly, coming close to the tree-based methods in terms of accuracy. Since the SVR model was trained with a linear kernel, its success further indicates that the data exhibits a predominantly linear structure. SVR's ability to fit the data well suggests that simpler linear relationships with limited non-linear deviations exist within the dataset.

In contrast, the Linear Regression and Neural Network models performed the worst. Linear Regression exhibited higher error rates across all metrics, implying that while the data may be mostly linear, the model's simplicity failed to capture the finer details or deviations. The Neural Network had the highest error rates, particularly in terms of MSE and MAPE, which suggests significant difficulties in fitting the data. This performance disparity could stem from issues like overfitting due to the model's complexity or underfitting if the network architecture or hyperparameters were not adequately tuned.

Overall, the comparative analysis highlights that while the data is largely linear, models that can handle slight non-linear deviations, such as tree-based ensembles and SVR, yield the best results. The poorer performance of simpler linear models and the neural network underscores the importance of selecting models appropriate to the data's complexity.

**Error Analysis:**  
The poor performance of the Neural Network warrants further investigation. Despite its ability to model complex, non-linear relationships, it appears to struggle with this dataset. One likely reason is that the data has a linear structure, making the added complexity of a neural network unnecessary and potentially detrimental. This complexity might lead to overfitting, where the model captures noise rather than meaningful patterns, resulting in higher error rates. Alternatively, the network may suffer from underfitting if the architecture is too simplistic or the hyperparameters (e.g., learning rate, number of epochs, hidden layers) are not optimized. Another reason could be the relatively limited size of the training data compared to the number of features (922 rows of training data with 196 input features), which would disproportionately affect neural networks due to their higher complexity, and make it more sensitive to noise. Although the Neural Network performed particularly poorly on this dataset, this lines up with the general consensus that neural networks are not as well suited to tabular data as other methods. The main benefit of neural networks is their ability to extract meaningful features from the data, but in our dataset, the input features coming from financial statements are already well defined.

In contrast, the SVR and tree-based models outperformed the Neural Network, reinforcing the idea that the dataset's structure is predominantly linear with minor deviations. SVR's success with a linear kernel suggests that the data does not require highly non-linear transformations to achieve accurate predictions. The tree-based models' ability to perform well can be attributed to their capability to handle slight deviations from linearity by creating flexible decision boundaries.

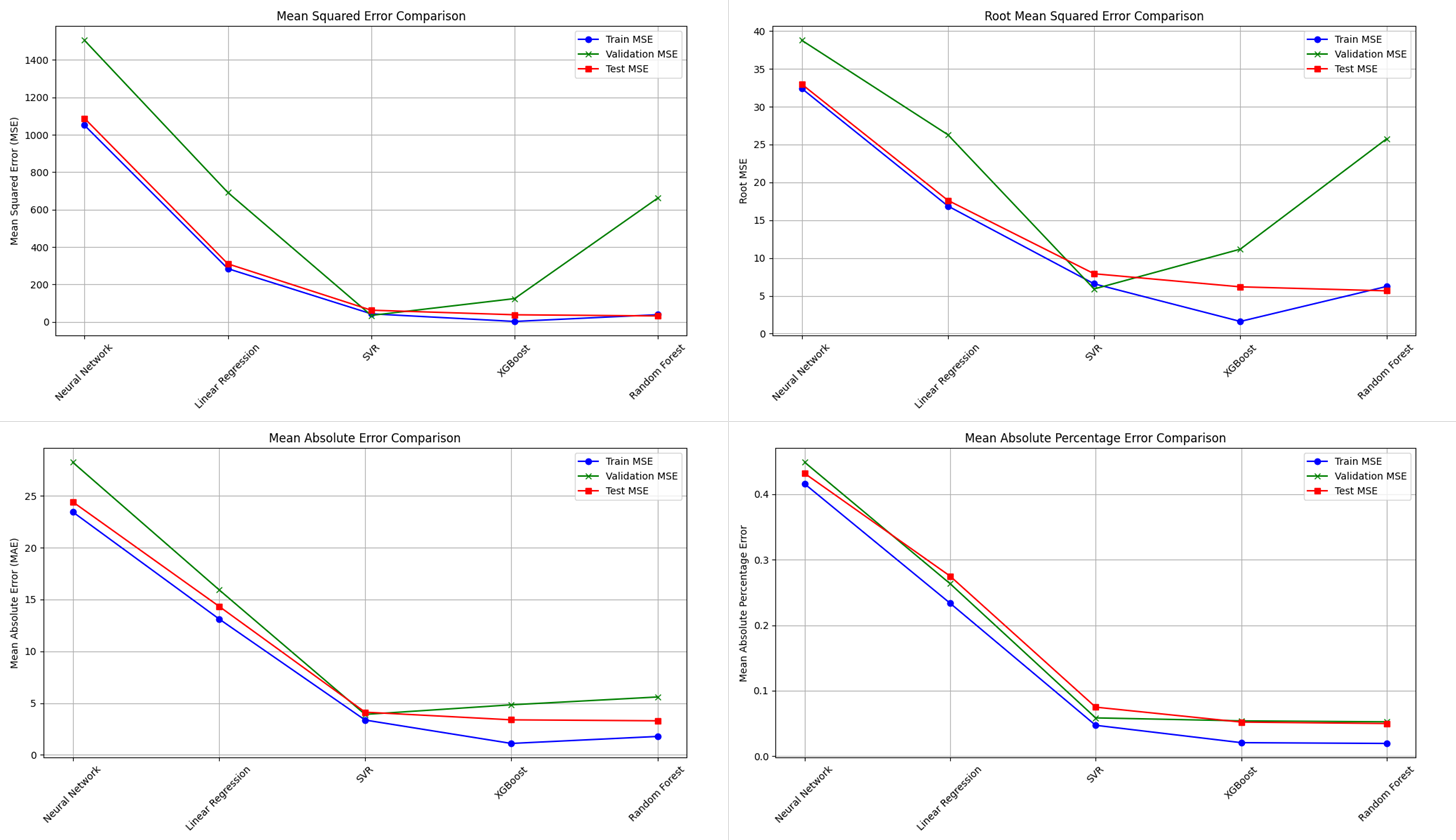
The Linear Regression model's relatively poor performance shows its inability to manage slight deviations from linearity, resulting in higher error rates. This suggests that while the data is mostly linear, the inability to capture subtle non-linear patterns impacts the model's accuracy.

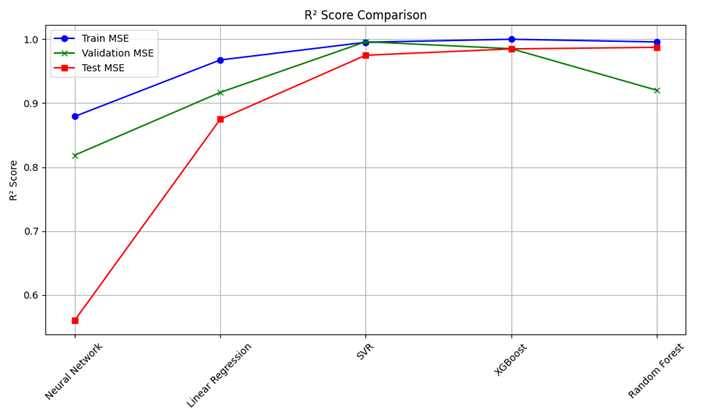
In order to improve these models, particularly the neural network, we could further tune the hyperparameters like the number of epochs, the learning rate, or the network architecture. Furthermore we could apply regularization or dropout to prevent overfitting and enforce a simpler solution. An easier solution however, would be focusing on models that resemble SVR or other tree-based ensembles.

In conclusion, the analysis indicates a primarily linear dataset with minor complexities that are effectively handled by SVR and tree-based models. Future improvements can be achieved by optimizing neural network parameters or enhancing linear models to account for subtle non-linearities.

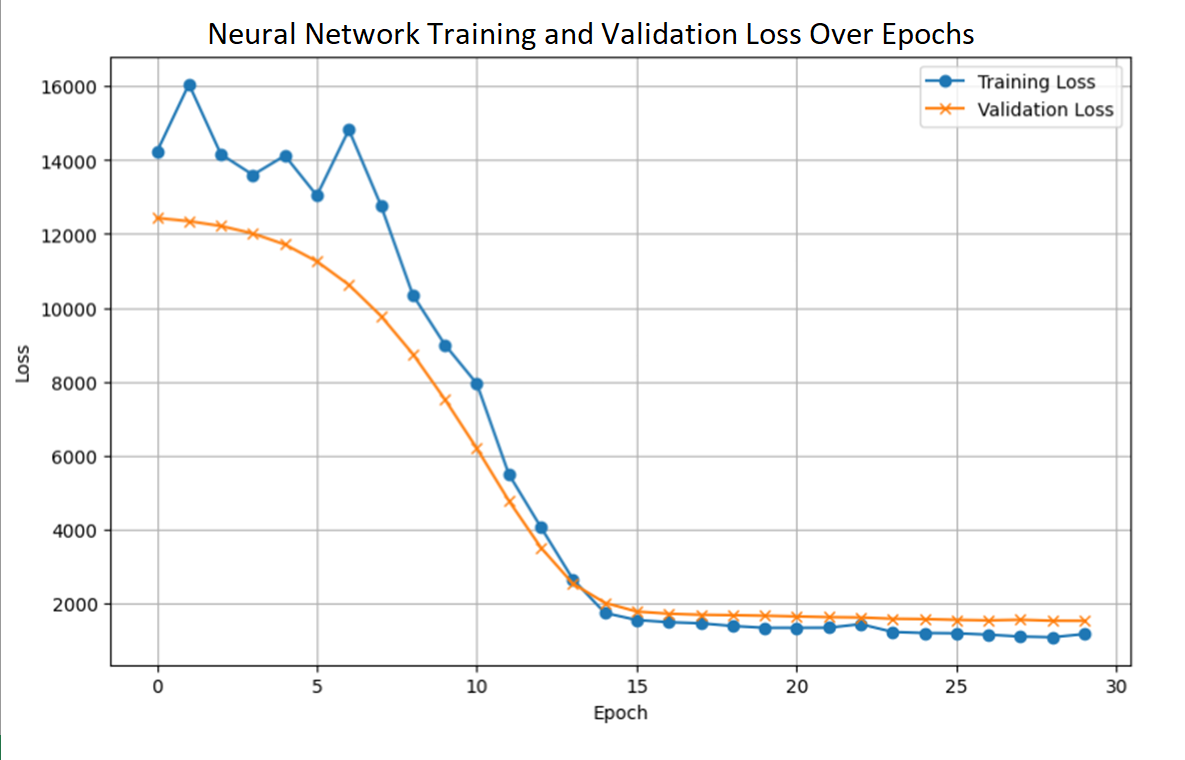
**Plots**

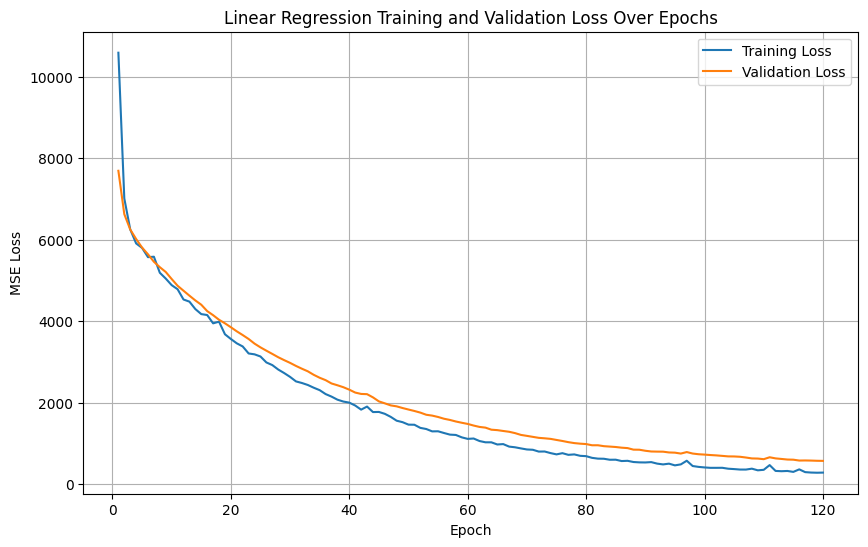
All Metrics for Train, Testing and Validation

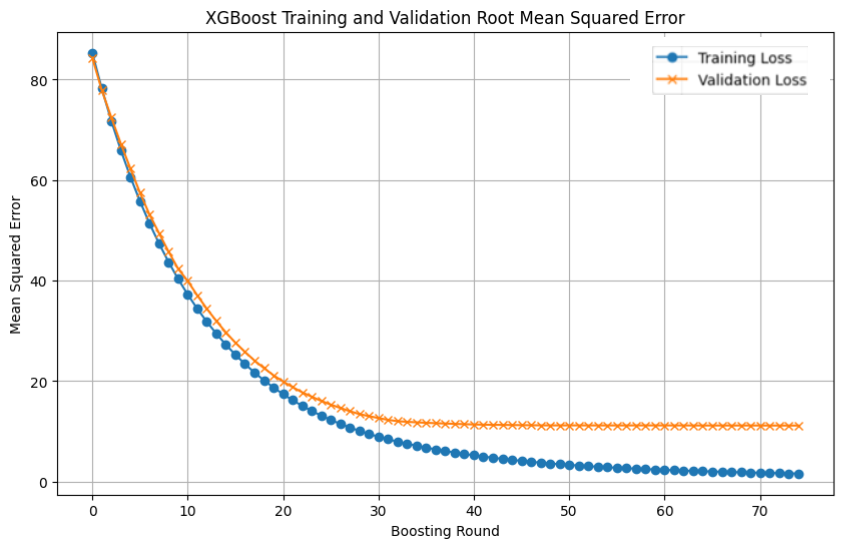




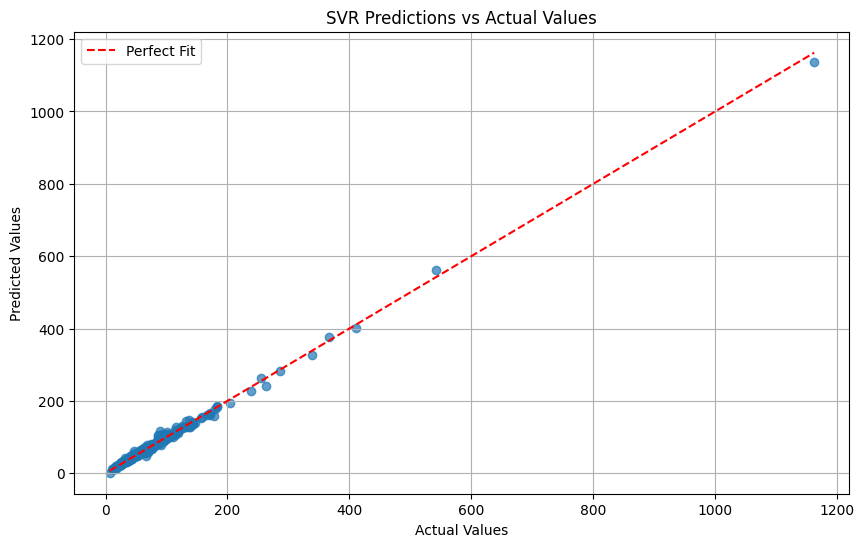
Mean Square Error Loss for Neural Network, Linear Regression, & XGBoost During Training

****

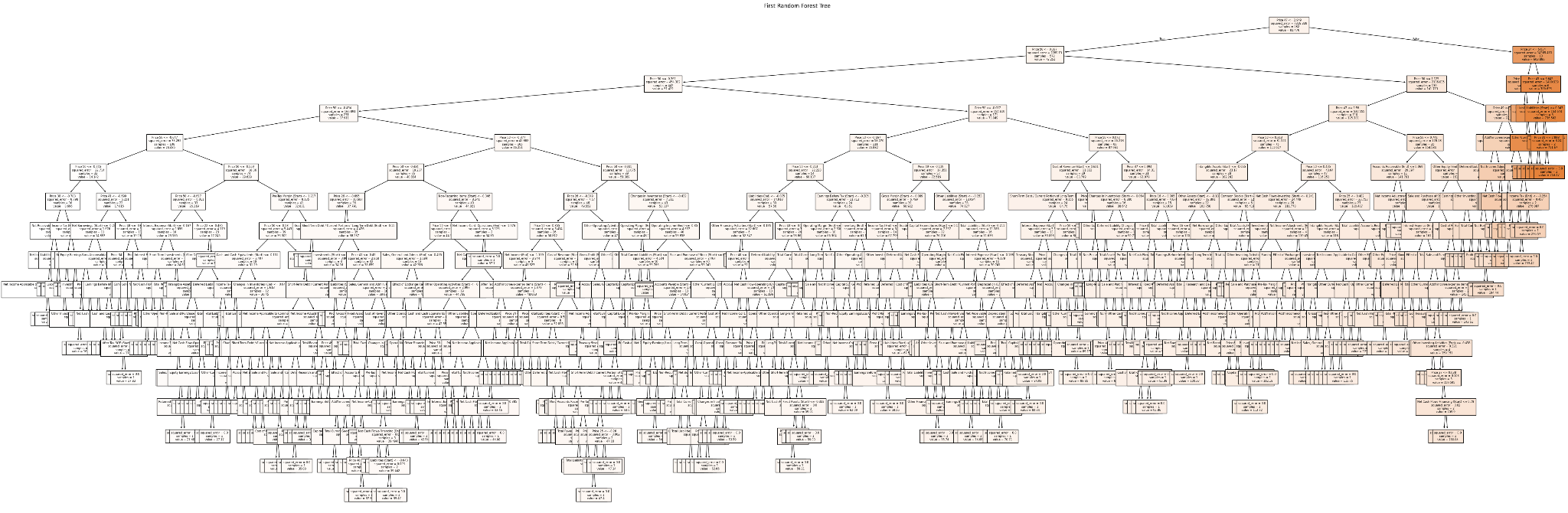
****



Visualization of Support Vector Regression in 2 dimensions



Visualization of Tree 1/100 Created by Random Forest

****

# 7. Discussion and Insights

**Interpretation of Results:**  
Comparative analysis of the five models revealed that the tree-based methods, Random Forest and XGBoost, outperformed other methods, achieving both the lowest error metrics and highest R2 scores. The strong performance of SVR with a linear kernel further validates the dataset’s linear tendencies, although it also highlights minor deviations that require more flexibility that linear models provide. While these models effectively captured the dataset’s predominantly linear patterns, the Linear Regression model struggled to address the slight non-linearities, and the Neural Network exhibited the worst performance, possibly due to overfitting or insufficient data. The Neural Network performance also underscores the importance of aligning model complexity with data characteristics, as attempting to train it on tabular data with relatively few samples may have also contributed to its high error rates.

**Limitations:**

Our chosen dataset presents several limitations. Although price data is given from a date range of 2010-2016, the data had to be limited to 2012-2016 in order to pair it with the data from company financial statements. This narrow range likely biases our models towards predicting general market trends during these years and may hinder its effectiveness when given data from before or after this time period. In addition, the financial statements are only provided yearly for each company since they are taken from SEC annual reports. When paired with our method of supplying two consecutive years of data from financial statements for each input, our dataset contained about three rows of data per company and resulted in a lower amount of data to work with. In order to work around this limitation, the final price that is being predicted by the models also had to be within the date range of the two financial statements. While this may likely not have a substantial impact on the models’ performances, since there is still a 30 day gap between input feature prices and the output price, it does not reflect a realistic use case for investors trying to predict future prices using current data.

Another large limitation comes from the subject of the stock market itself. Price movements for stocks on the S&P 500 and NASDAQ are generally slow and stock prices are likely to be within a few percent after one month. Company trajectories suggested by the data may also take longer than one month to be reflected in their stock prices. As an experiment, we tested our dataset on an additional model that simply predicted the output price after 1 month to be equal to the most recent price in the input features. The performance metrics for the model outperforms all models except for Random Forest. This demonstrates that the single price feature explains the vast majority of variance in the month-ahead price and that most of the models are failing to extract more meaningful information out of the other features supplied, or alternatively are having their performance degraded by them. Random Forest may be performing best here due to its ability to average or cancel out the effects of poorer predictions.

| **Model Name** | **Mean Squared Error (MSE)** | **Root MSE** | **Mean Absolute Error** | **R2 Score** | **Mean Absolute Percentage Error** |
| --- | --- | --- | --- | --- | --- |
| Neural Network | 1088.0126 | 32.9850 | 24.4299 | 0.5604 | 0.4317 |
| Linear Regression | 310.3066 | 17.6155 | 14.3498 | 0.8746 | 0.2749 |
| SVR | 62.6183 | 7.9132 | 4.1178 | 0.9747 | 0.0749 |
| XGBoost | 38.2049 | 6.1810 | 3.3770 | 0.9846 | 0.0521 |
| Random Forest | 31.9539 | 5.6528 | 3.2857 | 0.9871 | 0.0499 |
| **Price -30 days** | **32.9205** | **5.7376** | **3.4111** | **0.9867** | **0.0507** |

**Future Directions:**

There are several areas of the project that could be improved upon in future research. Regarding the dataset, efforts could be made to gather a wider timeframe of data for both stock prices and company financial statements. This should already be feasible due to the public nature of this data and the existence of stock APIs to help with data collection. This would vastly increase the size of the dataset and help the models learn to separate out noise and period-specific market trends. With an increased dataset time range, it would also enable the prediction of prices further ahead. Predicting prices only 1 month ahead may not leave enough time for meaningful price changes to take place, such as market corrections or responses to growth or stagnation in company performance. It may instead be more impactful to predict prices three, six, or twelve months ahead. Increasing the timeframe of the input data may also help the models to better capture long term trends and filter out noise and fluctuations from discrete events.

A large portion of our input features consist of metrics contained in two financial statements: one from the start of the year, and one at the end of the year. The purpose of supplying both was to allow models to capture the trajectory of the company and predict if the stock price will go up or down based on changes in metrics tied to company performance. However, simply inputting data from both financial statements may have also been inefficient and caused degraded performance for models that could not identify the relationship between the features. As such, it may be worth experimenting with inputting the differences between each year’s metrics instead.

Regarding the model selection and performance, several steps could be taken to address the poor performance of the Linear Regression and Neural Network models. Due to the high number of input features given, it is imperative that further research introduces some method of regularization to both models. This will help to deprioritize redundant or uncorrelated features and prevent them from degrading performance. Combating redundancy is also important in our dataset since adjacent price features are strongly correlated to each other and contain similar information. Another solution would be to apply Principal Component Analysis (PCA) to the input features prior to training. This would also help to reduce redundancy and identify uncorrelated input features.

# 8. Conclusion

In this project, we assessed multiple predictive models to forecast stock prices using both financial statement data and price trends. Random Forest and XGBoost outperformed other models, achieving the lowest error metrics and highest R2 scores, demonstrating their ability to capture both linear and slightly non-linear patterns in the data. The Support Vector Regression (SVR) model also performed well, which highlights the dataset’s predominantly linear structure. However, Linear Regression struggled with the slight non-linearities while the Neural Network performed by far the worst, likely due to shortcomings in the model architecture itself and limitations in our dataset. Surprisingly, a baseline model simply using the most recent input price significantly outperformed most models, raising concerns about their abilities to handle redundant or correlated features. Key limitations include the restricted timeframe of the dataset, limited data rows due to reliance on annual financial statements, and the inherent challenges of short-term stock price prediction. Future research could expand on the dataset’s scope and preprocessing methodology, and model shortcomings could be addressed to circumvent issues of redundancy and correlations.

# 9. Link to Demo

<https://drive.google.com/file/d/1qR24gND9iPwjrArUnYGjKtoWqyrCueO6/view?usp=sharing>