Stock Market Price Prediction Using Regression Machine Learning Models

Justin Dy
University of Texas at Dallas: ECS Graduate School
M.S. Computer Engineering Program
Richardson, TX, U.S.A.
Justin.Dy@utdallas.edu

Junchul Kim
University of Texas at Dallas: ECS Graduate School
M.S. Computer Engineering Program
Richardson, TX, U.S.A.
Junchul.Kim@UTDallas.edu

***Abstract—Background*: Stock market price prediction is a challenging and essential task in the realm of financial forecasting. The efficient and accurate prediction of stock prices have a big impact for investors, financial analysts, and policy-makers. In recent times, the growing availability of financial data and advancements in machine learning techniques have opened up new avenues for developing predictive models that can capture complex patterns inherent in stock market data. This paper will detail our experimentation with various machine learning regression models for stock market price prediction and compares their results and effectiveness at predicting prices.**

***Methods:* The Machine Learning regression methods that we will be using include: Linear Regression, Decision Tree Regression, K-Neighbors Regression, Gradient Boosted Tree Regression, Support Vector Machines (SVM), Recurrent Neural Networks (RNN) - Multi Layer Perceptron (MLP) Regressor, and Long Short-Term Memory (LSTM). We also applied PySpark Map Reduce techniques to SVM. We will be performing comprehensive comparisons of the core concepts of all these models and their resulting performance metrics and how well they perform in their predictions. The metrics that we will be focusing on are: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and coefficient of determination or R-squared(R^2).**

***Results and Conclusions:* For prediction of stock market prices the RNN regression models have the best performance metric results out of all the listed models that we are comparing.**

Keywords—Linear Regression; Decision Tree Regression; K-Neighbors Regression; Gradient Boosted Tree Regression; Support Vector Machines (SVM); Recurrent Neural Networks (RNN); Multi Layer Perceptron (MLP); Long Short-Term Memory (LSTM); Mean Absolute Error (MAE); Root Means Squared Error (RMSE); and coefficient of determination or R-squared(R^2)

I. INTRODUCTION (*TIME-SERIES PREDICTIONS*)

In this paper we are going to explore the different methods of machine learning regression models in the space of time-series predictions. Time-series prediction is a method in which future values are predicted based on the patterns and trends observed in historical data. In our case we will be focusing on predicting the future stock prices using certain machine learning regression models. The prediction of future stock prices is a very difficult task as there are many aspects that go into determining the price. There is a sense of randomness when it comes to stock prices as it is based on real-life events that are always perpetually occurring and makes regression models the best fit. In a simpler overview of the prediction that we are trying to make it can be seen as a time vs price graph. A stock as it exists in time will continue to go down the x-axis of that graph and we want to predict the y-axis that is the price. But given just time it would be impossible to say for certain whether a stock price will rise or fall, and this is where we introduce machine learning regression models as it can take into account other factors that affect the price, which we will address as the “features” from this point on. These features will be important to the prediction of stock prices as they can more effectively take into account the people who are accusing the prices to rise or fall by buying or selling stock and their sentimental values.

II. THE DATASET AND FEATURES

A. Stock Price Dataset (Using YFinance Library)

The dataset that we used to get our stock data from is Yahoo Finance API. We imported the yfinance library and downloaded the dataset with fields specified such as the time window for stock data. In our case we will focus on the last 5 years with a daily interval in order to get enough data points to train the machine learning models. We then used an 80 to 20 percent split in regards to our training and testing data respectively. We then used sklearn’s library for their Min-Max scaler which would rescale the data range to be within 0 to 1 to normalize the data and ensure that all features are within the same scale. We do some further processing of the data to correspond inputs X and outputs y for our machine learning models using a sliding window approach that is using prior time points to predict future ones, and we limit it to 100 previous days’ closing prices as the input and next day’s closing price as the corresponding output. This creates sequences of length 100 as inputs being our X train and X test data and their corresponding outputs being y train and y test. We shaped these sequences as two-dimensional arrays to be compatible with the models that we used from existing libraries as well as our SVM that we coded from scratch with use of MapReduce to demonstrate potential gains in using for larger datasets.

B. *Features to Consider for Predictions*

In order to be more organized in the consolidation of our results we mainly focused on using closing prices from previous days as our feature to predict the next day's closing price. But some other features that were considered to be impactful include but are not limited to: open, high, low, volume. These are the primary features that would carry heavy importance in a stock's past and present performance trends.

III. The Regression Models

It is known that stock price data is a regression problem. Stock prices are a continuous numerical value that changes with sentiment and are not discretely categorized. We therefore are going to introduce a brief summary of each machine learning regression model that we have used for comparison of performance metrics in their ability to predict stock prices. For more in depth information on the following models we highly recommend that you explore these models for yourselves using our sources and beyond.

C. *Linear Regression*

Linear regression is a machine learning model used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship, aiming to find the best-fitting line that minimizes the difference between observed and predicted values. It's commonly employed for predicting outcomes and understanding the impact of variables on the target which matches with what we are trying to accomplish with time-series prediction of stocks.. A simple equation to represent this is y=mx+b where y is the target dependent variable and x is the independent variable which is the predictor. In the equation m is slope and b is y-intercept. In the case of multiple x values we would fit a hyperplane to the data giving multiple points of b and x: y = b0 + b1*x1 + b2*x2 + ... + bn*xn.

D. *Decision Tree Regression*

Decision tree is a non-linear regression algorithm used to model complex relationships between a dependent variable and multiple independent variables and is generally considered as a classification model. It builds a tree-like structure, where each internal node represents a test on a feature, each branch represents an outcome of the test, and each leaf node represents a predicted value. The algorithm recursively splits the data based on the feature that best separates the data into homogeneous regions. In the end, the average (for instance) of the target variable within each leaf node is taken as the predicted value. We can see that this method of regression will most likely not perform well but it is useful to see why some models are based more so for classification tasks.

E. *K-Neighbors Regression*

The model works such that with a new data point, it identifies its K nearest neighbors based on a distance metric such as Euclidean distance, in the feature space. It then calculates the average/weighted average of the target values of these K neighbors, which becomes the predicted value for the new data point.

K-NN Regression is different from other models as it memorizes the entire training dataset. However, it can be sensitive to the amount of k neighbors it has, requires careful selection of distance metrics hence the use of euclidean distance, and might not perform well with high-dimensional data or noisy datasets. Additionally, it doesn't provide insights into the underlying relationships between features and the target variable, making it less interpretable compared to some other regression techniques.

F. *Gradient Boosted Tree Regression*

Gradient Boosted Tree regression is an ensemble learning technique that combines multiple decision trees to make accurate regression predictions. It works by sequentially adding decision trees, each one correcting the errors of its predecessor. The algorithm starts with a single weak learner such as a shallow tree, and then iteratively builds additional trees to minimize the residual errors. The new trees are fitted to the negative gradient of the loss function, optimizing the model's predictions at each step. By summing the predictions of all trees, the final model produces a more accurate and robust regression prediction. Gradient Boosted Tree regression is powerful, resistant to overfitting, and can be applied to various regression tasks. However, it may require tuning and can be computationally intensive which in the case of stock price prediction may be more harmful than just doing simple linear regression.

G. *Support Vector Machine Regression (SVM/SVR)*

Support Vector Machine Regression (SVM/SVR) is a supervised learning algorithm used for regression tasks. Unlike traditional linear regression, SVM regression does not aim to find a line that best fits the data but seeks to fit a "tube" around the data points.

The key concepts in SVM regression are as follows:

1. Hyperplane: In short is a decision boundary that will separate the two classes in a feature space in a classification problem. Since we are using it with a regression problem however, we use it to find a hyperplane that is best fit for data points to predict continuous time-series values in this case.
2. Epsilon (ε) Tube: provides a margin of error allowed for our predictions using the hyperplane mentioned allowing for the model to have more flexibility.
3. Loss Function: Loss(y, y_hat, epsilon) = max(0, |y - y_hat| - epsilon). This loss function will just check the target value y and prediction y_hat falls within our epsilon value. If the absolute difference is greater than ε then the deviation is larger.
4. Kernel Trick: y = Σ (α_i * K(x_i, x)) + b. Where y is prediction, α_i are lagrange multipliers from training, x_i are the support vectors from the training data, x is the input feature vector for which we want to make a prediction, and K(x_i, x) is the kernel function measuring the similarity between x_i and x. For

linear time-series predictions the most common is the linear kernel: K(x_i, x) = x_i^T * x

SVM regression is particularly useful when dealing with datasets that have complex relationships or when there are outliers which may cause worse results as we see later. It is also effective in high-dimensional spaces and provides good generalization even with a limited number of data points. Still SVM regression like other models is susceptible to requiring more tuning of hyperparameters. This model will also be more taxing the larger the dataset, which in our case we are using a particularly small dataset but we did choose this model to implement from scratch using PySpark MapReduce which should counter some of the shortcomings when dealing with extremely large datasets..

Overall, SVM regression is a powerful algorithm for regression tasks, especially in scenarios where the typical linear regression approaches may not perform as well as they should, or in the case of a stock dataset if we include more features than just the closing price for previous times and the next day.

H. Recurrent Neural Networks

1.) Multilayer Perceptron Regressor (MLP)

The MLPRegressor (Multi-Layer Perceptron Regressor) is a type of feedforward artificial neural network used for regression tasks. It is a supervised learning algorithm that can learn complex non-linear relationships between input features and the target variable. The neurons in this specific type of RNN can be seen as nodes that are interconnected to form a layer.

Key features of MLP:

- ➢ MLP's architecture has multiple layers of interconnected neurons, the first layer is the input layer while the last layer is the output. There is also one or more hidden layers in between these which continuous the interconnectivity from the first to the last layer
- ➢ There is also an activation function that is typically nonlinear that is applied to the neurons in the hidden layer. Some common activation functions include ReLU (Rectified Linear Unit), sigmoid, tanh (hyperbolic tangent), and linear activation for the output layer.
- ➢ MLP's are trained using gradient-based optimization techniques to improve predictions.
- ➢ Hyperparameters: MLP has different hyperparameters than the previously mentioned models since it is a type of RNN. Some of its hyperparameters are: the number of hidden layers, number of neurons in a layer, the learning rate, batch size, and the number of training epochs.
- ➢ Overfitting prone: like other RNNs it is prone to overfitting caused by the model sometimes being too complex and/or when there is limited training data. Regularization techniques can help mitigate these unwanted overfitting.

2.) Long Short-Term Memory (LSTM)

LSTM is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem in traditional RNNs. LSTM has the ability to capture and keep memory of long-term dependencies.

Key features of LSTM:

- ➢ Long-term dependencies: in time-series data like stock market prices there can be more in depth relationships between say price from a couple of months ago and the closing of a price on another day. LSTM networks, with their recurrency and memory cells are designed to retain this relationship that would otherwise be lost.
- ➢ Memory Cells: allows LSTM to store and "remember" information over extended periods of time.
- ➢ LSTM can also process data sequences of different lengths more efficiently. Time-series data, especially stock data, comes in different lengths.
- ➢ LSTM can also filter outliers and other irregular data points using memory cells to smooth out the variations in data.
- ➢ The non-linearity of stock prices makes it such that LSTM is a good fit for predicting data points
- ➢ LSTM can handle time lags and time dependencies extremely well which makes it a better model for use with time-series prediction of stock market prices
- ➢ LSTM can also process multivariate data which in the case of stock data multiple features are definitely involved in the influence of the target variable.

RNNs are generally the best for predicting stock prices due to their ability to capture dependencies in sequential data, handle non-linear data, and the model has memory cells. They are also very flexible in processing multivariable inputs and real-time data like stocks. RNNs can learn from historical patterns, incorporate multiple factors and it can handle missing data more effectively.

IV. Experimentation and Results

A. Log file of experiment results

Model	Hyperparameter	Results
SVM	Number of iteration=1000 Learning rate = 1e-5 Lambda param = 1e-5	Train/Test Split = 70:30 Training RMSE = 0.0440 Test RMSE = 0.1357 Training R2 = 0.9749 Test R2 = 0.6875 Training MAE = 0.0325 Test MAE =0.1190

Linear Regression	Fit intercept = True
	Copy_X = True
	N_jobs = None
	Normalize = False
		Train/Test Split = 70:30
		Training RMSE = 0.0126
		Test RMSE = 0.0431
		Training R2 = 0.9980
		Test R2 = 0.9684
		Training MAE = 0.0090
		Test MAE = 0.0334

Decision Tree Regression	Max_depth=5
	Criterion=mse
	Splitter=best
	Min_samples_split=2
	Min_samples_leaf=1
	Max_features=None
	Random_state=None
		Train/Test Split = 70:30
		Training RMSE = 0.0129
		Test RMSE = 0.1773
		Training R2 = 0.9979
		Test R2 = 0.4668
		Training MAE = 0.0095
		Test MAE = 0.1338

K-Neighbors Regressor	N_neighbors=5	Train/Test Split = 70:30 Training RMSE = 0.0089 Test RMSE = 0.2023 Training R2 = 0.9989 Test R2 = 0.3057 Training MAE = 0.0063 Test MAE = 0.1700

Gradient Boosted Tree Regression	N_estimators=100
	Learning_rate=0.1
	Max_Depth=3
	Min_saples_slit=2
	Min_samples_leaf=1
	Subsample=1
	Loss=ls
	Criterion=friedman_mse
	Random_state=None
		Train/Test Split = 70:30
		Training RMSE = 0.0071
		Test RMSE = 0.0814
		Training R2 = 0.9994
		Test R2 = 0.8877
		Training MAE = 0.0052
		Test MAE = 0.0642

MLP Regressor	Hidden_layer_sizes=(100,)
	Activation=relu
	Solver=adam
	Alpha=0.0001
	Learning_rate=constant
	Batch_size=auto
	Random_state=None
	Max_iter=200
		Train/Test Split = 70:30
		Training RMSE = 0.0293
		Test RMSE = 0.0927
		Training R2 = 0.9890
		Test R2 = 0.8543
		Training MAE = 0.0219
		Test MAE = 0.07388

LSTM	Number of LSTM layer=2 Dense layer and unit: 25□1 Optimizer=Adam Loss function=MSE Batch size = 1 Number of epochs=30	Train/Test Split = 70:30 Training RMSE = 1.2497 Test RMSE = 1.1937 Training R2 = 0.9994 Test R2 = 0.9994 Training MAE = 0.6721 Test MAE = 0.6137

B. Evaluation Criteria

1) RMSE (Root Mean Squared Error)

RMSE is the square root of the average of squared differences between prediction and actual observation. The RMSE gives a relatively high weight to large errors because the differences are squared before they are averaged, which means the RMSE should be more useful when large errors are particularly undesirable. It has the same units as the quantity being estimated, which can be useful for interpretation.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_i^{hat}\right)^2}$$

2) R2 (R-Squared)

R2, also known as the coefficient of determination, measures the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It provides a measure of how well future outcomes are likely to be predicted by the model. R2 score of 1 means the model can explain all the variability of the response data around its mean, while an R2 score of 0 means the model can't explain any of the variability.

$$R^2 = 1 - \frac{Sum\ of\ Squared\ Residuals}{Total\ Sum\ of\ Squares}$$

3) MAE (Mean Absolute Error)

MAE is the average of the absolute difference between the predicted and actual values. It gives an idea of how wrong the predictions were, where all individual differences have equal weight, no matter the direction (positive or negative errors). It is less sensitive to outliers compared to RMSE.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i^{hat}|$$

C. Analysis

1) SVM regression

1. RMSE is low which indicates a good performance.
2. R2 scores for both train and test are fine so we can expect a good performance.
3. MAE values are relatively small which indicates accurate prediction.

2) Linear Regression

1. RMSE values are low which indicates a good performance.
2. R2 scores for train and test sets are high which indicates the model performs well.
3. MAE are small which indicates accurate predictions.

3) Decision Tree Regression

1. RMSE values for train(0.0129) are lower than test(0.1773), the model does not generalize well to unseen data.
2. R2 values for train(0.9979) are higher than test(0.4668) which are potential overfitting.
3. MAE values for train(0.0095) are lower than test(0.1338), the model does not generalize well to unseen data.

4) K Nearest Neighbors Regressor

1. RMSE values are low which indicates a good performance.
2. R2 values for train(0.9989) are higher than test(0.3057) which are potential overfitting.
3. MAE values for train(0.0063) are lower than test(0.1700), the model does not generalize well to unseen data.

5) Gradient Boosted Tree Regression

1. RMSE values are low which indicates a good performance.
2. R2 score for both train and test set are high which indicates a good performance.
3. MAE values are small which indicates accurate prediction.

6) MLP Regressor

1. RMSE values are low which indicates a good performance.
2. R2 score for both train and test set are high which indicates a good performance.

3. MAE values are low which indicates a good performance.

7) LSTM

1. RMSE scores are relatively higher than other models, but it is acceptable range. (train RMSE=1.2497 and test RMSE=1.1937)
2. R2 values for both train and test set are high which indicates a good performance.
3. MAE values are relatively small which indicates accurate prediction.

D. Additional experiment

Interesting point from the experiment was that the LSTM RMSE was low enough but relatively high than other models. Since RMSE are computed based on the model's predictions compared to the actual values, higher RMSE means that LSTM is not a best model for the project which is a paradox of what the model built conceptually. Therefore, we tried to tune the LSTM model to see if we can reduce the RMSE value with different hyperparameter settings as below.

- Number of layers: [1, 2, 3]
- Number of units: [32, 64, 128]
- Dropout rate: [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
- Learning rate: [1e-2, 1e-3, 1e-4]

Despite tuning the LSTM model, we were unable to achieve an RMSE score as low as that of other models. However, it's worth noting that the LSTM did obtain the highest R2 score, indicating that our model's predictions are relatively close to the actual values (with a low RMSE) and that it effectively captures a large proportion of the variance in the data, as evidenced by the R2 score being close to 100%. To potentially improve the RMSE score, we can explore the option of increasing the number of layers as a hyperparameter, which may lead to better results in training the LSTM model

** Results of Hyperparamter tuning

Experiment No.	**Dropout rate**	**Learning rate**	**Number of LSTM layers**	**Number of LSTM units**	**Loss**	**MAE test**	**MAE train**	**R2 test**	**R2 train**	**rmse test**	**rmse train**
1	0	0.0001	3	128	1.378	0.433	0.471	0.999	0.999	0.757	0.801
10	0	0.0001	2	128	1.120	0.463	0.494	0.999	0.999	0.841	0.850
20	0.2	0.0001	1	128	1.080	0.572	0.592	0.999	0.999	1.028	1.038
30	0	0.0001	2	64	2.662	0.689	0.757	0.99	0.999	1.351	1.520
40	0.2	0.0001	3	64	2.915	0.673	0.787	0.999	0.998	1.431	1.662
50	0.1	0.001	1	128	5.243	0.992	1.057	0.998	0.998	1.702	1.782
60	0.4	0.001	2	32	4.233	1.179	1.293	0.998	0.998	2.118	2.240
70	0.1	0.01	2	32	83.40	2.254	2.566	0.991	0.990	4.467	4.974
80	0.2	0.01	2	128	87.24	4.643	5.011	0.973	0.970	8.135	8.918
90	0	0.01	1	128	98.35	5.922	5.826	0.967	0.970	8.957	8.911
100	0.5	0.0001	1	32	143.5	3.518	4.217	0.958	0.951	10.12	11.443
110	0.5	0.01	2	32	126.8	6.144	6.872	0.951	0.948	10.90	11.821

V. Conclusion

In this study, we explored the application of various machine learning regression models for stock market price prediction. Through systematic experimentation and analysis, we investigated the performance of Linear Regression, Decision Tree Regression, K-Neighbors Regression, Gradient Boosted Tree Regression, Support Vector Machines (SVM), Recurrent Neural Networks (RNN) - Multi Layer Perceptron (MLP) Regressor, and Long Short-Term Memory (LSTM).

Our study demonstrated that these regression models, especially LSTM, were effective at predicting stock market prices, taking into account complex patterns inherent in the data. We measured the performance of these models based on key metrics such as Root Mean Squared Error (RMSE), R-squared (R2), and Mean Absolute Error (MAE), enabling us to compare their effectiveness quantitatively.

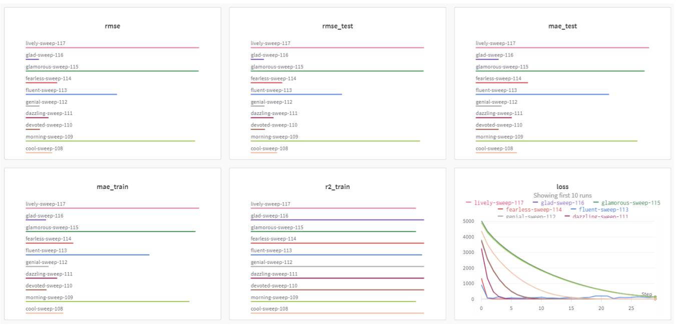
The results obtained from the RNN regression models, particularly LSTM, yielded the best performance metrics compared to all the other models. However, this was accompanied by a slightly higher RMSE value, indicating that there is a potential for further optimization. To further improve our model's performance, we experimented with various hyperparameters and found that certain configurations could help reduce the RMSE value.

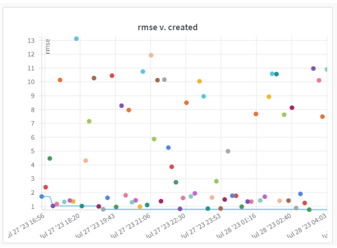
In conclusion, our research provides valuable insights into the predictive power of various machine learning regression models in the context of stock market price prediction. Our findings suggest that RNN-based models, particularly LSTM, offer significant potential in this domain. We believe that with further tuning and optimization, these models could provide even more accurate and reliable predictions, assisting investors, financial analysts, and policy-makers in their decision-making processes.

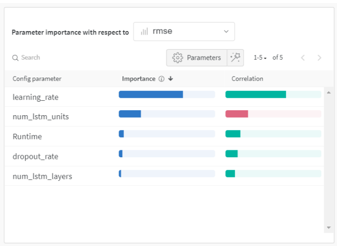
References

- https://pypi.org/project/yfinance/
- https://spark.apache.org/docs/latest/rdd-programming-guide.html
- https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html
- https://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html
- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
- https://en.wikipedia.org/wiki/Long_short-term_memory
- https://en.wikipedia.org/wiki/Linear_regression
- https://en.wikipedia.org/wiki/Decision_tree_learning
- https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- https://en.wikipedia.org/wiki/Gradient_boosting
- https://en.wikipedia.org/wiki/Support_vector_machine
- https://en.wikipedia.org/wiki/Recurrent_neural_network
- https://scikit-learn.org/stable/supervised_learning.html
- https://docs.wandb.ai/ref/python/sweep

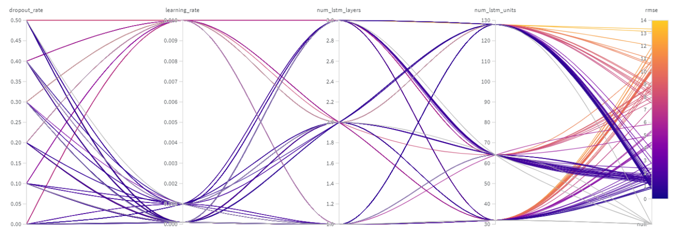
LSTM Hyperparameter tuning results:







Config parameter	Importance	Correlation
learning_rate		
num_lstm_units		
Runtime		
dropout_rate		
num_lstm_layers		


dropout_rate
learning_rate
num_lstm_layers
num_lstm_units
rmse