Final Project Report: AI Model Comparison

Course: Introduction to AI
Project Title: Comparative Analysis of Five AI Models
Submitted by: Joel Molina, Nikola Tunguz
Date: December 12, 2024
GitHub Code Link: https://github.com/NikolaTunguz/Stock-Predictor
Demo Video Link: https://youtu.be/Tnjv-RA2c5E

1. Introduction and Objective (5 Marks)

Objective:
The goal of the project is to predict stock market prices. More specifically, the goal of the project is to predict NVIDIA stock daily high, low, open, and close prices within 5 dollars of the real value within 3 days.

Problem Statement:

The models are tasked with a regression problem, predicting the next three day's stock prices based on the current day's prices.

Overview of AI Models Chosen:

Model Number	Model Name	Purpose
1	Linear Regression	Find a line of best fit for a stock's value given its history.
2	XGBoost Regressor	Create a gradient boosted decision tree for a stock's value given its history.
3	Neural Network	Create a multi-layer perceptron to predict a stock's value given its history.
4	Support Vector Regression (SVR)	Create support vectors that reshape the data to a higher dimension, where a kernel can predict a stock's value given its history.
5	Long Short-Term Memory (LSTM)	Create a recurrent neural network based on time dependencies between variables to predict a stock's value given its history.

2. Justification of Model selection (2 Marks)

Justification for Model Selection:

Model Name	Reason for Selection
Linear Regression	Stock prices generally increase over time, and linear regression may be able to capture it accurately for long term predictions.
XGBoost Regressor	Decision trees may be able to capture dependencies between the open, high, low, and close prices to accurately create decision tree splits for predicting price.
Neural Network	A neural network perceptron could connect previous prices to future prices and capture time dependencies with the weights.
Support Vector Regression (SVR)	Support vectors may be able to convert the values to a higher dimensionality with a kernel that can predict future prices well.
Long Short-Term Memory (LSTM)	LSTM utilizes time-series dependencies to predict future outputs. Utilizing current day values to predict future values would be useful.

3. Model Descriptions (1 Marks)

Model Overview:

Model Number	Model Name	Architecture Details	Key Features
1	Linear Regression	Simple linear model.	Captures linear trends.
2	XGBoost Regressor	Gradient boosting of decision trees.	Captures non-linear relationships.
3	Neural Network	Fully connected multi-layer perceptron.	Learns complex relationships between features.
4	Support Vector Regression (SVR)	Uses kernels to reshape data to find relationships.	Models relationships in higher dimensions.
5	Long Short-Term Memory (LSTM)	Recurrent neural network with memory.	Captures time dependencies.

4. Dataset Description (2 Marks)

Dataset Information:

Dataset Attribute	Description
Name	NVIDIA Stock Data, Latest and Updated
Source	Available from Kaggle at https://www.kaggle.com/datasets/kalilurrahman/nvidia-stock-data-latest-and-updated
Size	6,442 days of stock information from January 22, 1999 to August 28, 2024.
Class Distribution	No classes. Each entry contains the day, open, high, low, close, trading volume, dividends, and stock splits.
Preprocessing Steps	Removing rows with missing values. Removing the dividends, and stock split columns. Adding a time lag to the data by inserting next day values to each row.

Dataset Justification:

The dataset is suitable for the models because it reflects the problem, predicting NVIDIA stock prices. By leaving the preprocessing steps as is, the data can further be modified if each specific model needs it, such as neural network. For example, by adding the time lag, a neural network can use the tomorrow price has the prediction based on current day information. Conversely, these columns can be removed in linear regression, as the model fundamentally does not need that additional information and would not benefit from further preprocessing.

5. Experimental Setup (10 Marks)

Experimental Design:

Metrics
MAPE
RMSE

As this is a regression project, accuracy, precision, recall, and F1 score are not valid metrics. Instead, we utilized valid regression metrics. This includes MAPE (Mean Absolute Percent Error) and RMSE (Root Mean Squared Error) for all models.

Parameter Settings:

Due to each model having unique hyperparameters, each column represents the first, second, …, seventh hyperparameter of the specific model. The models do not have many, if any, shared hyperparameters.

Model Name	**Hyperparameter 1**	**Hyperparameter 2**	**Hyperparameter 3**	**Hyperparameter 4**	**Hyperparameter 5**	**Hyperparameter 6**	**Hyperparameter 7**	**Additional Notes**
Linear Regression	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
XGBoost Regressor	objective = 'absoluteerror'	n_estimators = 300	learning_rate = 0.05	max_depth = 4	subsample = 0.7	alpha = 0.1	eval_metric = 'rmse'	N/A
Neural Network	Criterion = MSE loss	# epochs = 20	Learning rate = 0.0001	Optimizer = adam	Batch size = 64	Hidden layer1 size = 512	Hidden layer2 size = 256	Three Layer Model
Support Vector Regression (SVR)	Kernel = 'rbf'	Max_iter = -1 (until convergence)	C = 1.0	Epsilon = 0.1	Gamma = 'scale'	N/A	N/A	N/A
Long Short-Term Memory (LSTM)	Criterion = MSE loss	# epochs = 100	Learning rate = 0.001	Optimizer = Adam	Batch size = 64	N/A	N/A	1 layer model

Environment Details:

Component	Specification
Operating System	Windows 11
Software Version	Python 3.12.7
Hardware	Intel, AMD, NVIDIA
Link to the code base	https://github.com/NikolaTunguz/Stock-Predictor

6. Results and Analysis (50 Marks)

Performance Metrics:

Model Name	Mean Absolute Percent Error (MAPE)	Root Mean Squared Error (RMSE)
Linear Regression	1.960 %	1.557
XGBoost Regressor	75.581 %	50.899
Neural Network	5.413 %	3.618
Support Vector Regression (SVR)	87.349 %	53.411
Long Short-Term Memory (LSTM)	68.419 %	49.325

Comparative Analysis:

To summarize, we only had two models perform at an acceptable level. This was the neural network with a RMSE of 3.617 which correlates to a MAPE of 5.413%. For the linear regression, this was slightly better with an RMSE of 1.557 and a MAPE of 1.960%. These models are relatively accurate and utilize the historical trends to make a relevant stock price prediction. The XGBoost Regressor, SVR, and LSTM, on the other hand, performed significantly worse with substantial inaccuracies that make them invalid predictors of the stock price. This is due to the train/validation/test split which is elaborated on in the error analysis section.

Each model's specific performance had varied. We see that the SVR had the worst MAPE of 87.349%, the XGBoost Regressor was the second worst with a MAPE of 75.581%, and the LSTM was slightly better with a MAPE of 68.419% Overall these models succumbed to the same issue that crippled their accuracy, the train-validation-test split. This split is not helpful for sequential time-series data, as the more recent data and patterns would never be trained on. The reason that the neural network and the linear regression models were resistant to this is due to their reliance on the most recent day values. They ignore the time-series related issues by following their own training process, such as linear regression predicted based on the line of best fit.

Error Analysis:

The main source of error comes from the dataset itself and the data split. Due to the sequential nature of the data, the data cannot be randomized during training. Because of this, the models lack the ability to generalize well to test data, as the test data is newer, representing new relationships over time, not reflected in the training data.

In essence, the models cannot train on the more recent data, as that data is in the validation and test splits. The model believes that the "next day" for the test dataset is the first day in the validation set, causing high error in test, and lower error in validation, as seen in the plots in the next section.

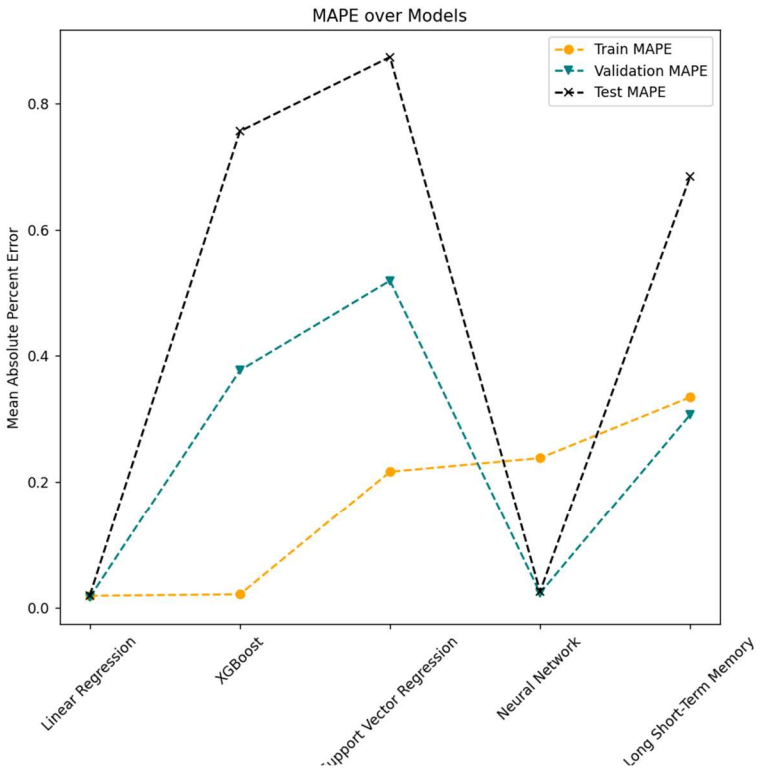
Using a train/validation/test split does not work well for this dataset. This could be removed by training on the entire dataset, instead of designating part to the validation subset, which would allow the model to learn on the most recent day, closest to the test set for its evaluation, or abstracted to real time data. Similarly, the models could be improved to include sliding windows for predicting more than one day in advance, to make predictions further away more accurate.

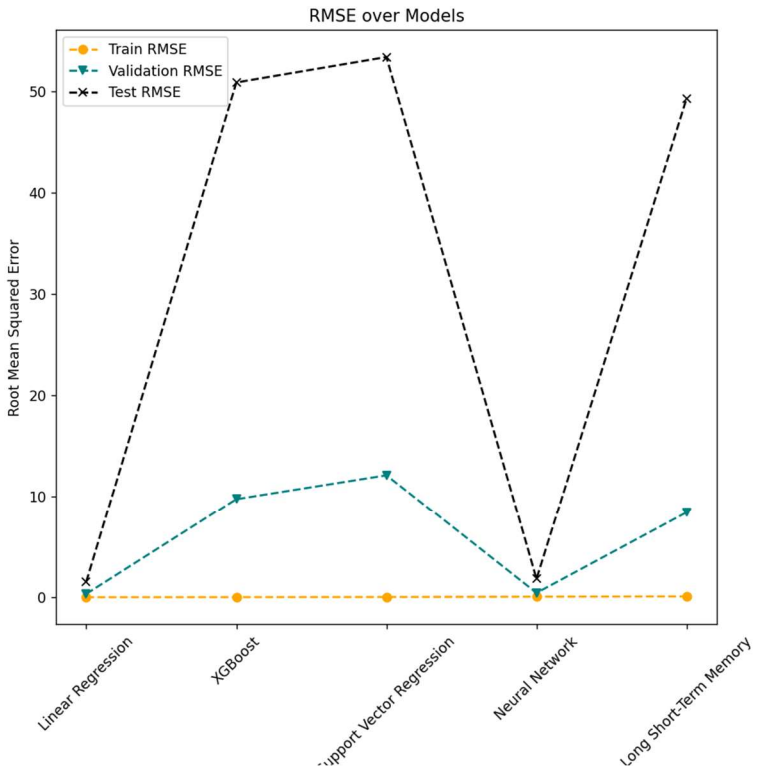
Plots

Model 3 Day Stock Predictions

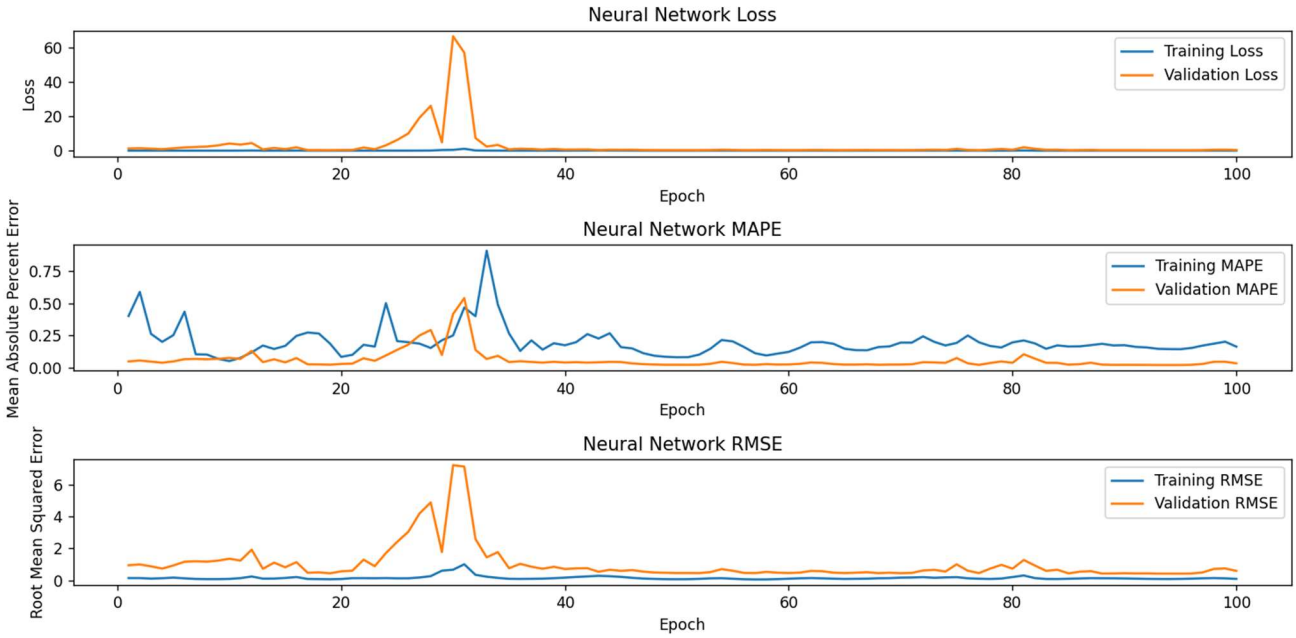
```
Linear Test Dataset Percent Error:  1.960115127172523 RMSE:  1.5569825058139086
XGBoost Test Dataset Percent Error:  75.58118905393339 RMSE:  50.898925478935645
SVR Test Dataset Percent Error:  87.3487785152734 RMSE:  53.41137609757833
NN Test Dataset Percent Error:  5.413322995644278 RMSE:  3.6181257
LSTM Test Dataset Percent Error:  67.81146320749957 RMSE:  49.177692
3 Day Predictions:
Linear Regression
1 Day Prediction: Open = 128.49595363606016, High = 130.3754278772023, Low = 125.87589906863376, Close = 127.90482323797167
2 Day Prediction: Open = 128.03960659568406, High = 130.02773015194737, Low = 125.8001509072376, Close = 127.98151109179184
3 Day Prediction: Open = 128.12046326758943, High = 129.9917350727089, Low = 125.92679990097555, Close = 127.98128464007497
XGBoost
1 Day Prediction: Open = 6.837416172027588, High = 6.919508934020996, Low = 6.756464004516602, Close = 6.812570571899414
2 Day Prediction: Open = 6.837416172027588, High = 6.919508934020996, Low = 6.756464004516602, Close = 6.812570571899414
3 Day Prediction: Open = 6.837416172027588, High = 6.919508934020996, Low = 6.756464004516602, Close = 6.812570571899414
Support Vector Regression
1 Day Prediction: Open = 3.5463941468659574, High = 3.60355996425713, Low = 3.4783161553371063, Close = 3.529978860064532
2 Day Prediction: Open = 3.540486229563961, High = 3.634189082298742, Low = 3.473805146542187, Close = 3.5510903013811097
3 Day Prediction: Open = 3.5596046324003994, High = 3.6524710211820652, Low = 3.4866636021138406, Close = 3.5656274479871795
Neural Network
1 Day Prediction: Open = 132.27020263671875, High = 133.06800842285156, Low = 133.70030212402344, Close = 134.77133178710938
2 Day Prediction: Open = 132.27020263671875, High = 133.06800842285156, Low = 133.70030212402344, Close = 134.77133178710938
3 Day Prediction: Open = 132.27020263671875, High = 133.06800842285156, Low = 133.70030212402344, Close = 134.77133178710938
Long Short-Term Memory
1 Day Prediction: Open = 9.180760383605957, High = 9.308748245239258, Low = 9.03711986541748, Close = 9.176881790161133
2 Day Prediction: Open = 9.180760383605957, High = 9.308748245239258, Low = 9.03711986541748, Close = 9.176881790161133
3 Day Prediction: Open = 9.180760383605957, High = 9.308748245239258, Low = 9.03711986541748, Close = 9.176881790161133
```

Training and Test MAPE and RMSE

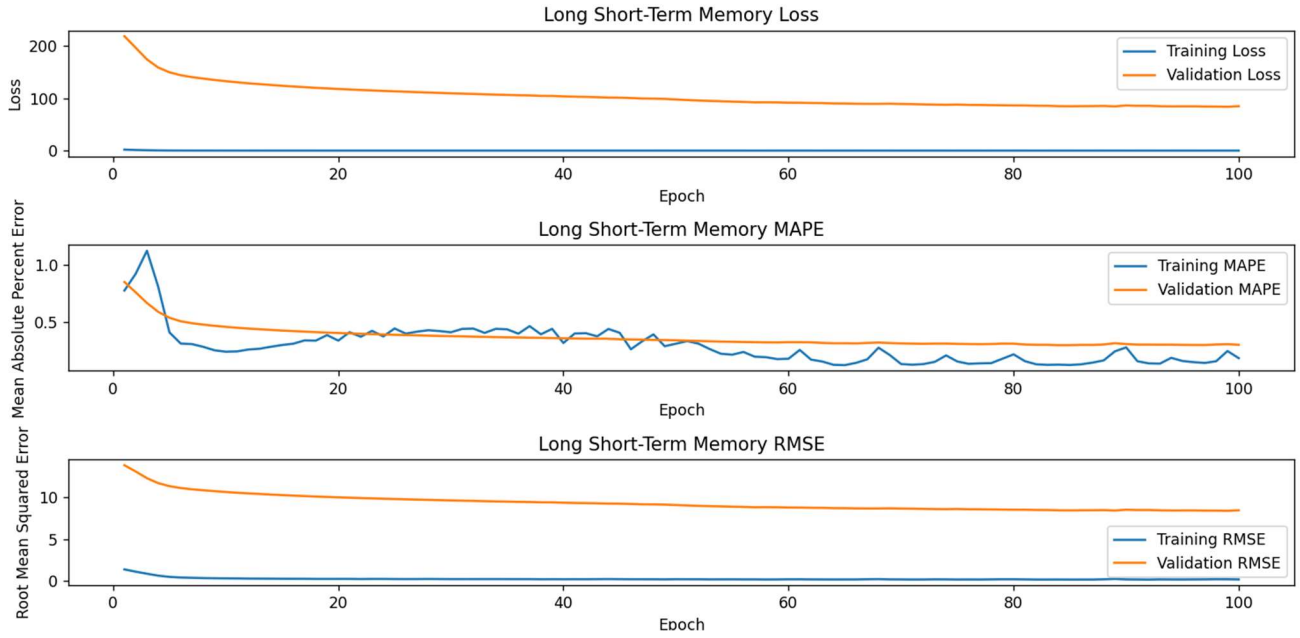




Neural Network Performance Over Epochs



Long Short-Term Memory Performance Over Epochs



7. Discussion and Insights (10 Marks)

Interpretation of Results:

The biggest insight gained from the model comparison was the importance of appropriate data. Two of our models were not designed to capture higher dimensional or time series data, however these performed the best due to our dataset being split sequentially to preserve time data. Conversely, for the other three models, they were designed for processing time data, however the split made the more relevant recent time data unavailable.

Another insight from this model comparison was the power of linear regression. Despite being one of the simplest models and statistical regression tools, it managed to outperform all four other models in every metric. More complex models such as LSTM had converged with a high loss. This means that the models were not able to analyze the relationships between the variables well, and a more sophisticated approach should be taken when processing and lagging the data, to allow the model to capture relationships.

Limitations:

Within the design process, there were limitations with the data splits. The project's goal was to make a time series-based approach, however this did not work well by using the train/validation/test split. The design of the project should have gone in a different direction. Similarly, the data itself had limitations in its future values. The original dataset only contained stock information for the same day. This needed to be processed by lagging values.

Additionally, each model had its own limitations. Our best performing model, linear regression, is very limited to long-term growth, as it represents the overall sentiment of the stock. Conversely, neural network was the opposite, performing well on short term predictions, however very poorly long term. The other models predicted the end of the training split and start of the validation data correctly, however this had very poor extrapolation to days ahead.

Future Directions:

The biggest step that the project could be improved is using real-time data. This change would allow for continuously updated and more accurate predictions. In tandem with a better dataset, there would be a time-series specific model that can handle all the nuances that come with working with time-series data.

The biggest change that could be made to improve the quality of these models would be to improve the sliding window. Currently, the window is only one day ahead of the current day is, and this is unchanging. This means that the window does not go backwards, and only goes one day in advance. Making this change would bias the accuracy of the model to rely on long-term relationships, instead of one day ahead.

8. Conclusion (5 Marks)

Overall, the models failed to capture the relationships between prior and future prices. In order to accurately capture time-series data, more complex data processing must be implemented, in addition to using a model designated for it, such as LSTM, or other more complex model, such as ARIMA.

For the models created in this project, Linear Regression performed decently well, and extrapolated will on a long-term basis. However, it failed to predict short-term changes to prices.

The XGBoost Regressor model performed poorly. The model always predicted a set value for the stock prices, evident of a poor decision tree being made.

Neural Network's performance was the best out of all the models. The model predicted short-term prices accurately, however the long-term predictions were very poor. While this is bad, short-term predictions were the goal of the project.

Support Vector Regression (SVR) performed poorly as well, as its designated use is with stationary data, where the mean and variance are consistent over time. With a stock such as NVIDIA, this performs extremely poorly, as NVIDIA has had large spikes in its prices multiple times.

Long Short-Term Memory (LSTM) performance was unexpectedly poor. The model captured time dependencies, however not to an accurate level. This could be due to the historical prices of the stock, as relatively recently NVIDIA's prices had drastically changed.