UNIVERSITY of York
Europe Campus
CITY College

Computer Science

CCS6611: Deep Learning

Spring 2024

Experiments with training Neural Networks

Submission Deadline: 28/04/2024

Actual Submission Date: 28/04/2024

Student Registration Number
23032

Abstract

This project focuses on predicting the closing stock price of Google using historical stock data and recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) cells. The dataset includes features such as date, open price, high price, low price, close price, volume, etc. The project begins with data exploration and preprocessing, where the dataset is sorted, formatted, and normalized. Next, multiple LSTM-based models are built and trained with various hyperparameter combinations. Hyperparameters like the number of LSTM layers, units, activation functions, optimizers, learning rates, and batch sizes are tuned using a grid search approach. The models are evaluated using metrics such as mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE). The best-performing model is selected based on these metrics. The selected model is used to make predictions on the test set, and the results are evaluated against the actual stock prices. The project concludes with insights into the model's performance and suggestions for future research. Overall, this project demonstrates the potential of LSTM-based models in predicting stock prices and provides a foundation for further exploration in this field.

Table of contents

1. Introduction

The aim of this project is to experiment with a Google stock dataset [1] using recurrent neural networks (RNNs) with Long Short -Term Memory (LSTM) [2] cells. Different combinations of hyperparameters were used to test how the RNNs behave and perform. The goal was to create a model that accurately forecasts future stock prices based on past performance.

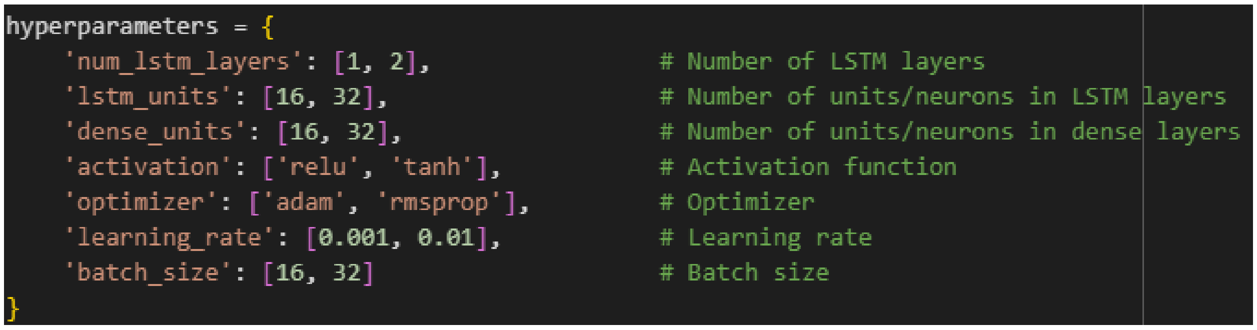
2. Data Exploration and Preprocessing

The dataset used for this project contains historical stock data for Google, including features such as date, open price, high price, low price, close price, volume, etc. The data exploration phase involved visualizing different features over time to understand trends, patterns, and relationships.

In the data preprocessing phase it was necessary to drop some of the features because for the features open price, high price, low price, close price, and volume there were corresponding features keeping the adjusted values for each which was adjusted based on stock splits and dividends of the company. Therefore it was decided to drop these adjusted values. Moreover, various other actions were applied for preprocessing the data including, sorting the data by date, formatting the date column, normalizing the numerical features using Min-Max scaling, and splitting the data into training and testing sets.

3. Model Building and Hyperparameter Tuning

In the beginning, the first model that was created was an experiment using combinations of hyperparameters that was predefined in a dictionary as Figure 1 shows. All possible combinations were created by using **itertools** python library.



```
hyperparameters = {
    'num_lstm_layers': [1, 2],                # Number of LSTM layers
    'lstm_units': [16, 32],                   # Number of units/neurons in LSTM layers
    'dense_units': [16, 32],                  # Number of units/neurons in dense layers
    'activation': ['relu', 'tanh'],           # Activation function
    'optimizer': ['adam', 'rmsprop'],         # Optimizer
    'learning_rate': [0.001, 0.01],           # Learning rate
    'batch_size': [16, 32]                    # Batch size
}
```

Figure 1: Hyperparameter combinations

The model architecture consists of LSTM layers followed by dense layers. Hyperparameters such as the number of LSTM layers, LSTM units, dense units, activation function, optimizer, learning rate, and batch size were tuned using a grid search approach [2]. The goal was to find the combination of hyperparameters that minimizes the validation loss and mean absolute error.

Table 1 below, presents the results of top 5 models sorted by mean absolute error (MAE) and validation loss. All these models trained for 50 epochs.

The first one, **Model 11** performed really well having a mean absolute error of **0.010149**. The second **Model 19,** had MAE **0.010319**, very close to the first one. The differences in the hyperparameters between these two models was the number of dense units (16 and 32) and the activation function (tanh and relu).

Table 1: Top 5 Models Validation Loss and Mean Absolute Error

Model index	**num_lstm _layers**	**lstm_unit s**	**dense_units**	**activation**	**optimizer**	**learning_ rate**	**batch_ size**	**val_loss**	**val_mae**
11	1	16	16	tanh	adam	0.010	32	0.000212	0.010149
19	1	16	32	relu	adam	0.010	32	0.000227	0.010319
27	1	16	32	tanh	adam	0.010	32	0.000231	0.010511
56	1	32	32	tanh	adam	0.001	16	0.000238	0.010564
41	1	32	16	tanh	adam	0.010	16	0.000271	0.010754

4. Experiment with the hyperparameters

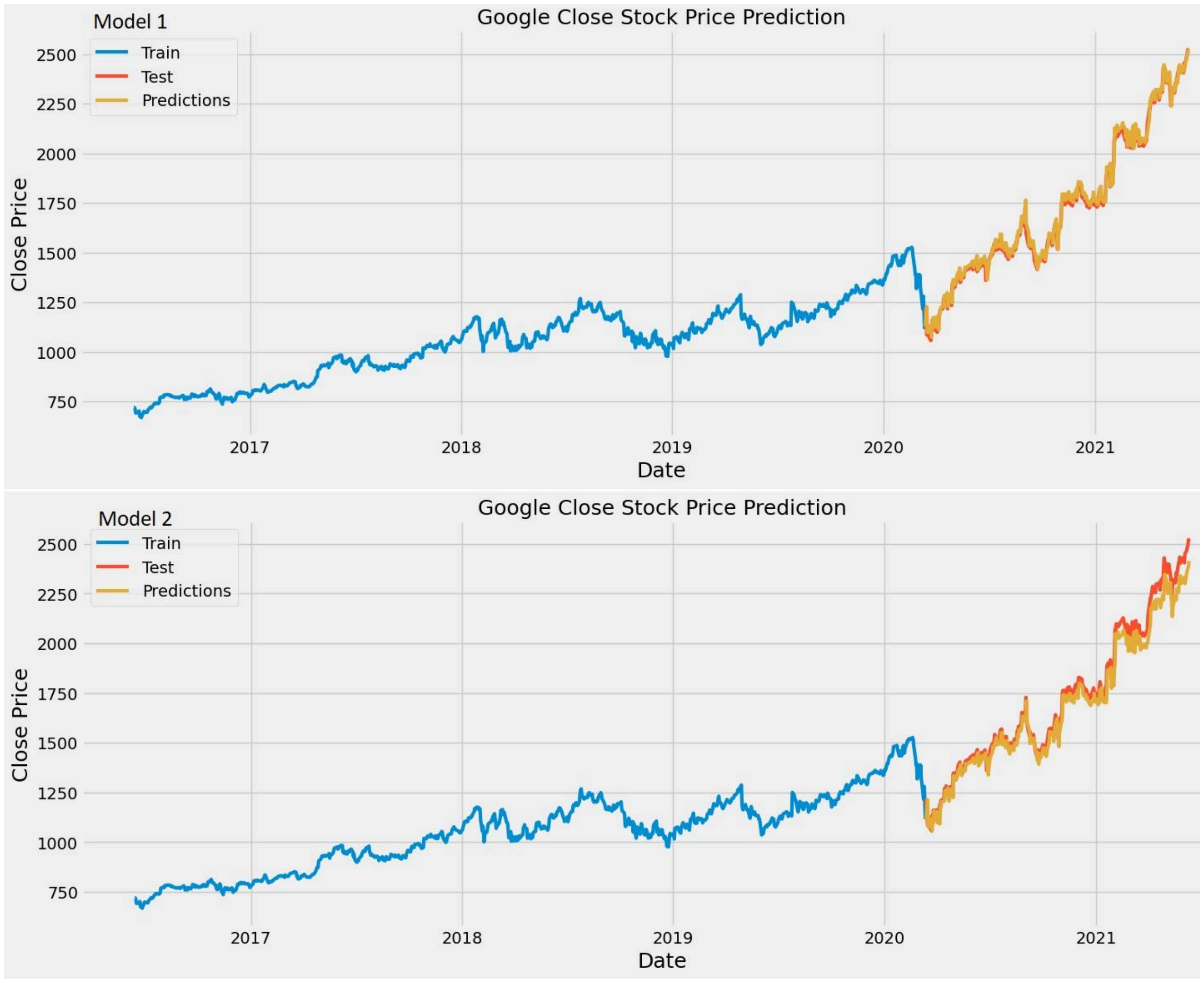
Several models were created, trained and evaluated using different hyperparameter combinations. The evaluation metrics that were used were mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE). The best-performing model was selected based on these metrics.

Once the best model was identified, it was used to make predictions on the test set. The RMSE was calculated to evaluate the model's performance on the test set. To apply a number of experiments, four new models were created. In these four models some hyperparameters remained the same, such num_lstm_layers, lstm_units, optimizer_name, learning_rate and batch_size. The first one was constructed by using the same hyperparameters as Model 11. The models in this phase run for 100 epochs. The first two models used to experiment with the hyperparameter **dense units.**

Table 2: Evaluation comparison of Model 1 & Model 2

Model index	**Activation**	**Dense units**	**Epochs**	**MSE**	**MAE**	**RMSE**	**Training time (sec)**
1	tanh	16	100	0.000121	0.007747	13.93	147.42
2	tanh	64	100	0.000118	0.007914	40.32	147.44

The first two models trained for 100 epochs and the activation function that was used was ‘tanh’. As Table 2 demonstrates, Model 1 had a slightly lower loss compared to Model 2. That indicated that it performs slightly better in terms of minimizing the error during training. Moreover, Model 1 has a lower MAE, which means its predictions were, on average, closer to the actual values compared to Model 2. Also, Model 1 has a significantly lower RMSE compared to Model 2. A lower RMSE indicates that the predictions of Model 1 are closer to the actual values on average, suggesting better overall performance. The training time of these models was almost the same besides the fact that they had large differences between them in the dense units 16 in Model 1 and 64 Model 2. Therefore, larger values in dense units seem like it does not have high affection in the convergence time of the models.



Feature 2: Prediction comparison of Model 1 & Model 2

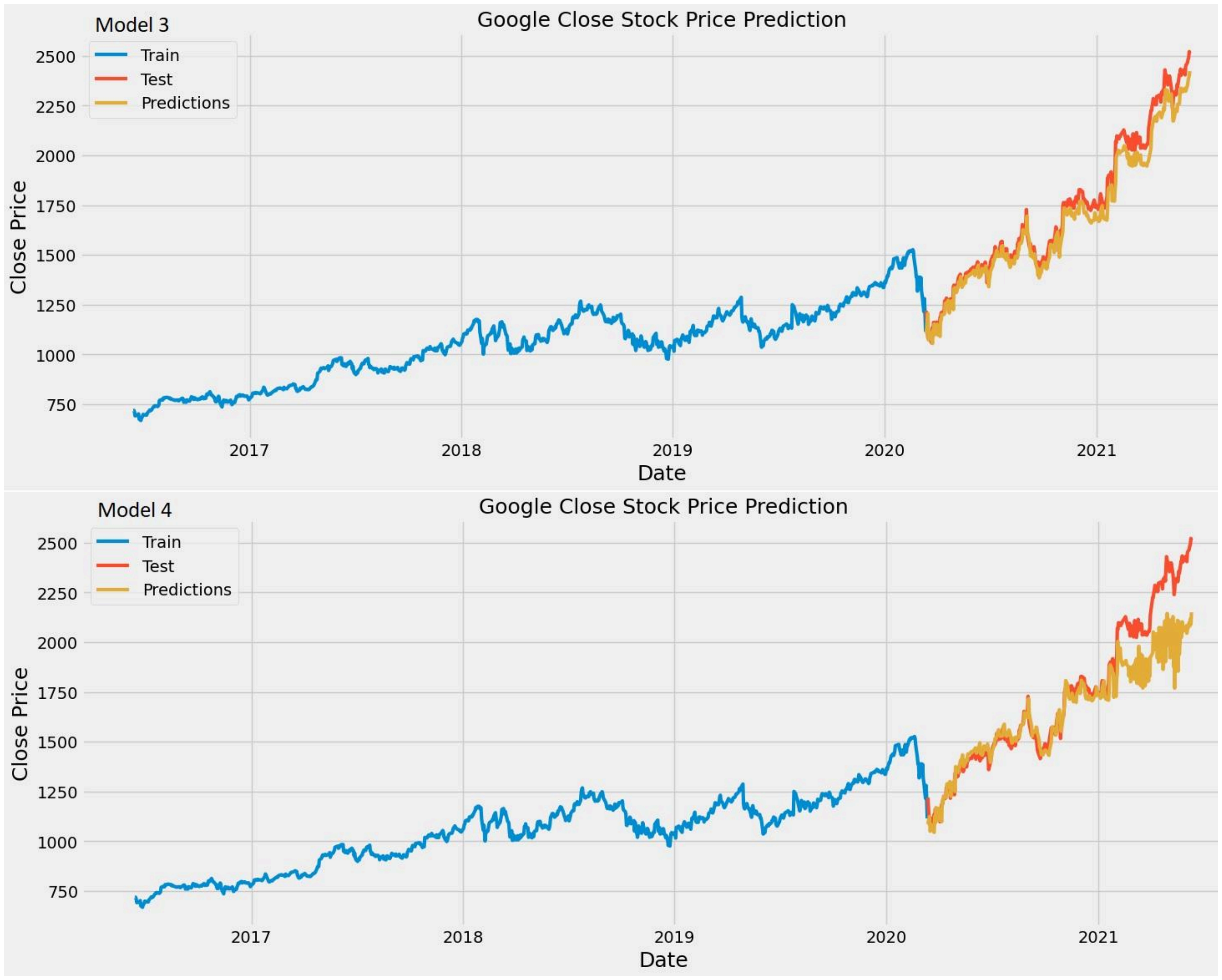
The predictions of Model 2 were actually far more accurate than Model 1 as Feature 2 presents. Model 1 until the year 2021, predicted just about the same as Model 2, but after 2021 the predictions of Model 1 were not much precise.

The next two models Model 3 and Model 4, were compared in the hyperparameters activation function and dense units, keeping all the other hyperparameters the same.

Table 3: Evaluation comparison of Model 3 & Model 4

Model index	**Activation**	**Dense units**	**Epochs**	**MSE**	**MAE**	**RMSE**	**Training time (sec)**
3	relu	32	100	0.000134	0.007902	50.6	146.65
4	relu	32	300	0.000102	0.007403	78.4	364.02

Based on the comparison of Table 3 and Table 2, it turned out that using the activation function ‘relu’ the models’ training time could be faster than using ‘tanh’. The first Model 3 had less training time than the first two. Moreover if we compare the first two with the Model 4 which trained for 300 epochs, three times more epochs, the training time of Model 4 was a bit more than the double time of the Model 1 & Model 2.



Feature 3: Prediction comparison of Model 3 & Model 4

As Feature 3 shows, after the year 2021 the predictions of Model 1 were a lot inaccurate, something that the larger number of epochs did not help the model to have accurate predictions.

5. Conclusion

In conclusion, in this project multiple LSTM-based models for predicting Google stock prices were developed and evaluated. The best model achieved a low validation loss and mean absolute error. This indicated its effectiveness. The predictions on the test set were close to the actual stock prices. That is evidenced by the low RMSE value. For this problem, 'relu' activation function compares the convergence time and was more effective than 'tanh'. Finally, increasing the epochs number from 100 to 300 led to worse predictions. More research and experimentation about epochs and dense units could potentially lead to even more accurate forecasts.

6. References

[1]"Google Stock Prediction," www.kaggle.com. https://www.kaggle.com/datasets/shreenidhihipparagi/google-stock-prediction/code (accessed Apr. 28, 2024).

[2]Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," Neural Computation, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: https://doi.org/10.1162/neco_a_01199.