COS30018

Intelligent Systems

Option B: Stock Prediction

Task B.5 – Machine Learning 2

Name: Duc Thuan Tran
Student ID: 104330455
Tutor: Dr. Ru Jia
Tutorial: Friday 2:30 – 4:30

Table of Contents

1. Introduction

This code has been updated to version v0.4, which includes improvements to handle more complex stock price prediction tasks. In prior versions, the model projected stock prices based on a single feature (for example, the closing price), limiting its ability to forecast prices correctly across numerous days or when multiple features were included.

This version focuses on solving more complex prediction tasks, such as:

- **Multistep prediction,** which estimates the closing price for numerous days in the future using previous closing values.

- **Simple multivariate prediction**, which uses multiple features such as the opening price, highest price, lowest price, adjusted closing price, and volume to predict the closing price for a specific day in the future.

- **Multivariate multistep prediction**, which combines the multistep and multivariate approaches to predict the closing price for multiple future days using a sequence of input features.

The goal of this version is to enhance the model's ability to forecast stock prices by leveraging more data and predicting further into the future, which provides a more realistic and practical application for stock price analysis.

2. Implementation of Multistep Prediction Function

2.1 Description

The multistep prediction function was designed to allow the model to predict the closing prices for multiple future days (k days) based on the most recent sequence of historical closing prices.

2.2 Code Implementation

The function uses a recurrent neural network (RNN) architecture such as LSTM or GRU to generate predictions iteratively. At each time step, the model predicts the next closing price and updates the input sequence with this prediction to forecast the next time step.

```python
def multistep_predict(model, last_sequence, scaler, steps):
    """
    Predict stock prices for multiple days into the future (multistep prediction).
    """
    predictions = []
    current_sequence = last_sequence

    for _ in range(steps):
        prediction = model.predict(current_sequence)
        prediction = scaler.inverse_transform(prediction)
        predictions.append(prediction[0, 0])

        # Update the current sequence with the new prediction
        new_data_point = np.zeros((1, 1, current_sequence.shape[2]))  # Create a placeholder for the next input
        new_data_point[0, 0, 0] = prediction[0, 0]  # Add the predicted value to the new sequence
        current_sequence = np.append(current_sequence[:, 1:, :], new_data_point, axis=1)

    return predictions
```

Image 1. multistep_predict function

The multistep prediction implementation in the main function predicts stock prices for 5 days into the future. The code below demonstrates how the model is used in practice:

```python
# Multistep prediction
steps = 5  # predicted days
multistep_predictions = multistep_predict(model, last_sequence, scalers["Close"], steps)
print(f"Multistep Predictions: {multistep_predictions}")
plt.figure(figsize=(10, 6))
plt.plot( *args: y_test_unscaled[:steps], color="black", label=f"Actual {COMPANY} Price")
plt.plot( *args: range(len(multistep_predictions)), multistep_predictions, color="blue", label=f"Multistep Predicted {COMPANY} Price", linestyle='--')
plt.title( label: f"{COMPANY} Multistep Share Price Prediction", fontsize=16)
plt.xlabel( xlabel: "Steps into Future", fontsize=12)
plt.ylabel( ylabel: f"{COMPANY} Share Price", fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

Image 2. Multistep Prediction plot

2.3 Results

The multistep prediction performed well in generating smooth curves. However, the model exhibited a tendency to **overestimate** the future prices, producing an upward trend that did not reflect the actual stock price. This suggests that while the model captures certain trends, it may require further tuning to prevent overestimation.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
Multistep Predictions: [100.511635, 120.40545, 142.82173, 160.35721, 172.72081]
```

Image 3. Multistep Prediction Output

3. Implementation of Simple Multivariate Prediction Function

3.1 Description

The simple multivariate prediction function used multiple features, including the opening price, highest price, lowest price, and volume, to predict the closing price for a future day.

3.2 Code Implementation

This function takes the last sequence of feature data as input and predicts the closing price using an RNN model. It relies on multiple features to provide a more informed prediction than just using the closing price alone.

```python
def multivariate_predict(model, data, feature_columns, scaler, prediction_days):
    """
    Predict stock price for a future day using multiple features (multivariate prediction).
    """
    last_sequence = data[-prediction_days:][feature_columns].values
    last_sequence = scaler.transform(last_sequence)
    last_sequence = last_sequence.reshape(1, prediction_days, len(feature_columns))

    prediction = model.predict(last_sequence)

    reshaped_prediction = np.zeros((1, len(feature_columns)))
    reshaped_prediction[0, 0] = prediction[0, 0]  # Assign the predicted closing price

    # Perform inverse scaling with all features (closing price will be used)
    return scaler.inverse_transform(reshaped_prediction)[0, 0]
```

Image 4. multirative_predict function

This implementation allows for visually comparing the actual stock prices and the predicted values over the 5 future steps.

```python
# Multivariate prediction
multivariate_prediction = []
for i in range(5):  # Predict for 5 future days
    prediction = multivariate_predict(model, data, FEATURE_COLUMNS, scalers["all_features"], PREDICTION_DAYS)
    multivariate_prediction.append(prediction)
print(f"Multivariate Prediction: {multivariate_prediction}")
plt.figure(figsize=(10, 6))
plt.plot( *args: y_test_unscaled[:5], color="black", label=f"Actual {COMPANY} Price")
plt.plot( *args: range(5), multivariate_prediction, color="red", label=f"Multivariate Predicted {COMPANY} Price", linestyle='--')
plt.title( label: f"{COMPANY} Multivariate Share Price Prediction", fontsize=16)
plt.xlabel( xlabel: "Steps into Future", fontsize=12)
plt.ylabel( ylabel: f"{COMPANY} Share Price", fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

Image 5. Mutilative prediction plot

3.3 Results

The multivariate prediction model showed **flat predictions** for future prices, indicating that it struggled to capture the stock's variability. The predicted values remained almost constant over time, suggesting that the model might need more tuning or additional data to improve its performance.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
Multivariate Prediction: [101.64303311346589, 101.64303311346589, 101.64303311346589, 101.64303311346589, 101.64303311346589]
```

Image 6. Multirative_predict output

4. Implementation of Multivariate Multistep Prediction Function

4.1 Description

The multivariate multistep prediction function aimed to predict multiple days into the future using a combination of historical data across multiple features.

4.2 Code Implementation

The function used the same architecture as the previous models but iteratively predicted multiple time steps, updating the input features for each step.

```python
def multivariate_multistep_predict(model, data, feature_columns, scaler, prediction_days, steps):
    """
    Predict stock prices for multiple days into the future using multiple features (multivariate and multistep prediction).
    """
    predictions = []
    last_sequence = data[-prediction_days:][feature_columns].values
    last_sequence = scaler.transform(last_sequence)
    current_sequence = last_sequence.reshape(1, prediction_days, len(feature_columns))

    for _ in range(steps):
        prediction = model.predict(current_sequence)

        reshaped_prediction = np.zeros((1, len(feature_columns)))
        reshaped_prediction[0, 3] = prediction[0, 0]


        prediction_rescaled = scaler.inverse_transform(reshaped_prediction)

        # Store the predicted closing price
        predictions.append(prediction_rescaled[0, 3])

        # Update the current sequence for the next step
        new_data_point = np.zeros((1, 1, current_sequence.shape[2]))
        new_data_point[0, 0, 3] = prediction[0, 0]
        current_sequence = np.append(current_sequence[:, 1:, :], new_data_point, axis=1)

    return predictions
```

Image 7. multirative_multistep_predict function

```python
# Multivariate multistep prediction
multivariate_multistep_predictions = multivariate_multistep_predict(model, data, FEATURE_COLUMNS, scalers["all_features"], PREDICTION_DAYS, steps)
print(f"Multivariate Multistep Predictions: {multivariate_multistep_predictions}")
plt.figure(figsize=(10, 6))
plt.plot( *args: y_test_unscaled[:steps], color="black", label=f"Actual {COMPANY} Price")
plt.plot( *args: range(len(multivariate_multistep_predictions)), multivariate_multistep_predictions, color="purple", label=f"Multivariate Multistep Predicted {COMPANY} Price",
plt.title( label: f"{COMPANY} Multivariate Multistep Share Price Prediction", fontsize=16)
plt.xlabel( xlabel: "Steps into Future", fontsize=12)
plt.ylabel( ylabel: f"{COMPANY} Share Price", fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

Image 8. Multivariate multistep plot

4.3 Results

The multivariate multistep prediction provided smoother predictions than the simple multivariate model, but it still predicted a **downward trend** that did not align with the actual stock price, which remained stable. Despite capturing some level of change, the model's predictions remained off in terms of direction.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
Multivariate Multistep Predictions: [100.76811482425389, 95.42338529876703, 87.31303424683257, 79.23758920629986, 72.55247596389371]
```

Image 9. Multivariate multistep output

5. Visual Comparison of Multistep, Multivariate, and Multivariate Multistep Predictions

To evaluate the performance of the three prediction functions **Multistep Prediction**, **Multivariate Prediction**, and **Multivariate Multistep Prediction** visualizations were created to compare their predictions with actual stock prices. This comparison highlights the strengths and weaknesses of each approach in forecasting stock prices.

5.1 Multistep Prediction

Behavior: The multistep prediction shows a consistently upward trend, far exceeding the actual values.
Strengths: The model generates a smooth curve, making it easy to interpret.
Weaknesses: It clearly overpredicts the stock price and doesn't align with the actual stock price after the first couple of steps.
Conclusion: This function is likely overfitting or unable to generalize to future data. The prediction grows unrealistically high compared to the actual data, meaning it doesn't perform well here.

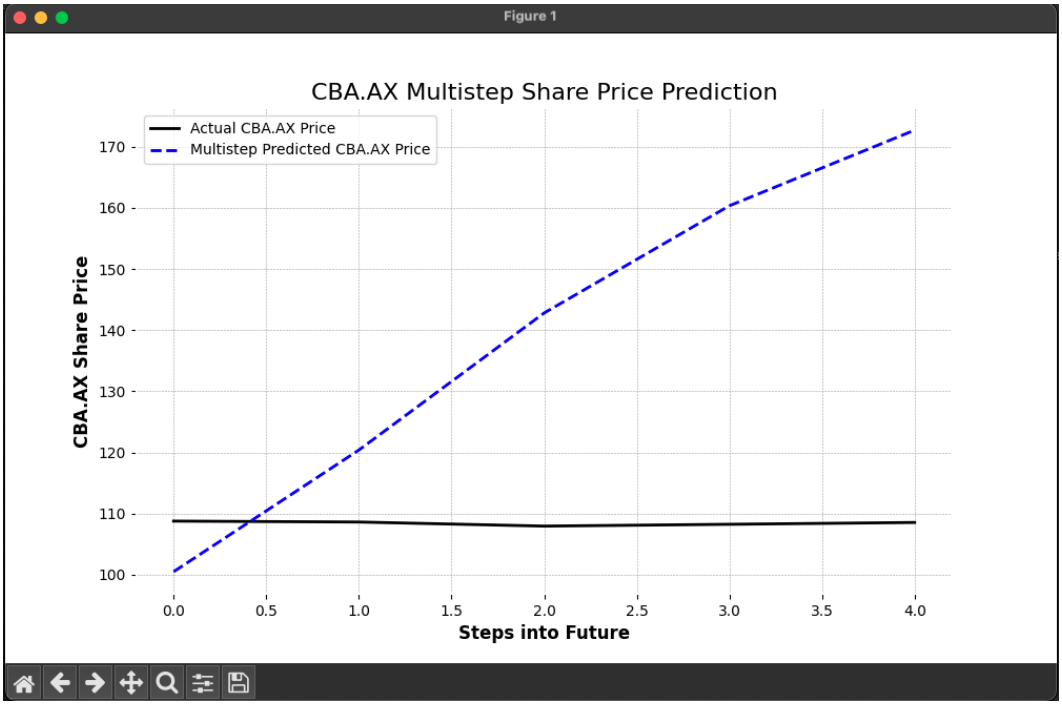

Image 10. Multistep Share Price Prediction

5.2 Multivariate Prediction

Behavior: The predicted line remains almost flat (or very low and straight) for all five days, showing no real change in the forecasted price.
Strengths: The predicted line is stable, which could be seen as conservative.
Weaknesses: The prediction doesn't capture any of the trends present in the actual stock price. It fails to adapt to any fluctuations and simply stays flat. This suggests that the model is not learning from the features well or may need more complex training for multivariate predictions.
Conclusion: This is the least accurate of the three methods. The flat line suggests that the model is underfitting or that the features used aren't providing sufficient information for accurate predictions.

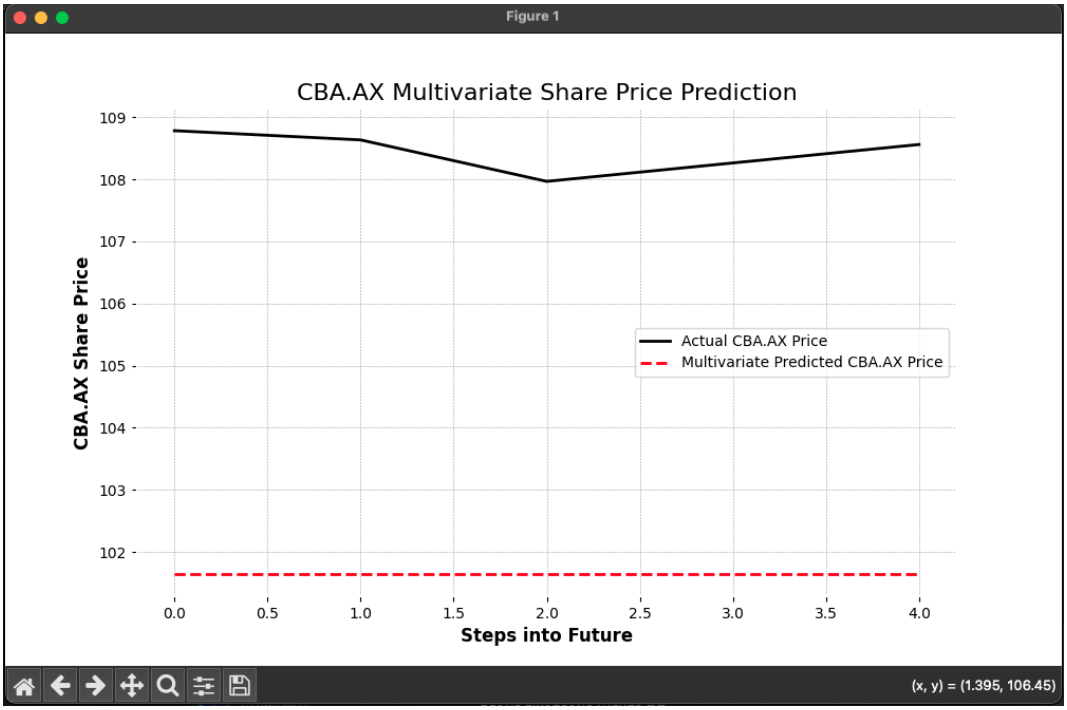

Image 11. Mutilative Share Price Prediction

5.3 Multivariate Multistep Prediction

Behavior: This prediction shows a downward trend as it predicts multiple days into the future, while the actual stock price remains mostly flat.

Strengths: The trend is smoother than in the multivariate case and it successfully picks up some change in stock price rather than staying flat.

Weaknesses: The direction of the prediction is incorrect, while the actual stock price remains stable, the predicted prices keep dropping.

Conclusion: This is better than the simple multivariate prediction, but still performs poorly because it predicts the wrong trend (a significant decline when the actual stock price remains stable).

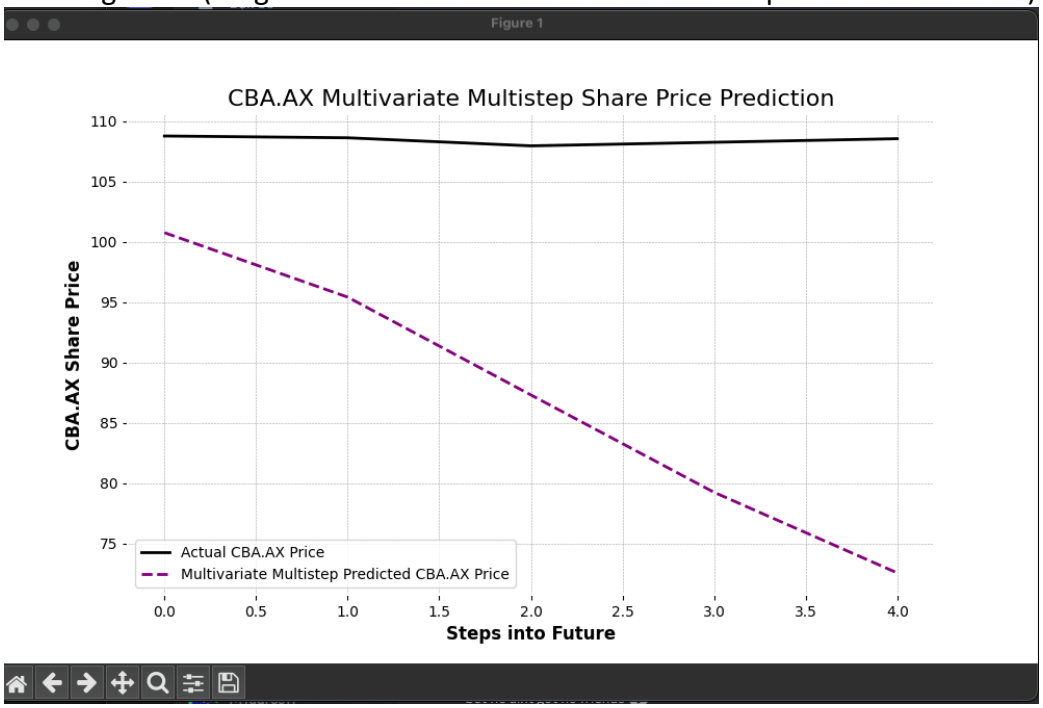

Image 12. Mutilative Multistep Share Price Prediction Chart

5.4 Comparison Summary

Function	Prediction Behavior	Strengths	Weaknesses	Overall
Multistep Prediction	Overestimates with an upward trend	Smooth curve, consistent predictions	Significantly overpredicts the future prices	**Nicest Curve**, but inaccurate
Multivariate Prediction	Flat and underpredicts future values	Stable, simple prediction	Doesn't capture stock trends; stays too flat	**Least accurate**, unrealistic
Multivariate Multistep	Smooth downward trend	Captures some trends, smooth line	Wrong direction, underpredicts consistently	**More balanced** but incorrect

6. GitHub Repository

The project's code base is hosted on GitHub for version control and review. Everyone can access the repository via the following link: GitHub Repository. This repository contains all necessary files for the project and is available for the tutor to review the work.

7. References

- https://www.youtube.com/watch?v=UuBigNaO_18NeuralNine. (2022, October 1). *Stock Price Prediction using LSTM in Python*. [Video]. YouTube. Retrieved from
- TensorFlow. "Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)." TensorFlow, 2023. Available at: https://www.tensorflow.org/guide/keras/rnn
- Fischer, Thomas, and Christopher Krauss. "Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions." European Journal of Operational Research, 270(2): 654-669, 2018. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0377221717301440
- Hyndman, Rob J., and Athanasopoulos, George. "Forecasting: Principles and Practice." OTexts, 2018. Available at: https://otexts.com/fpp3/