Task B5: Machine Learning 2

Dr Ru Jia

Friday, 14:30 – 16:30

Name: Min Khant Aung
ID: 103833225

Task 5 Report: Machine Learning 2

1. Introduction

The objective of this task was to implement predictive functions for forecasting stock prices using both **multistep** and **multivariate** approaches, and to combine these for a **multivariate, multistep** prediction model. These models aim to provide accurate predictions for stock prices by leveraging different aspects of the data, such as:

- **Multistep Prediction**: Predicting closing prices for multiple future days based on past closing prices.
- **Multivariate Prediction**: Using multiple stock features (e.g., opening price, highest price, lowest price, trading volume) to predict the closing price.
- **Multivariate and Multistep Prediction**: Combining both methods to predict future prices using multiple features over multiple days.

This report provides a detailed explanation of the implementations, the results of each model, and a comparative analysis of their performance.

2. Implementations

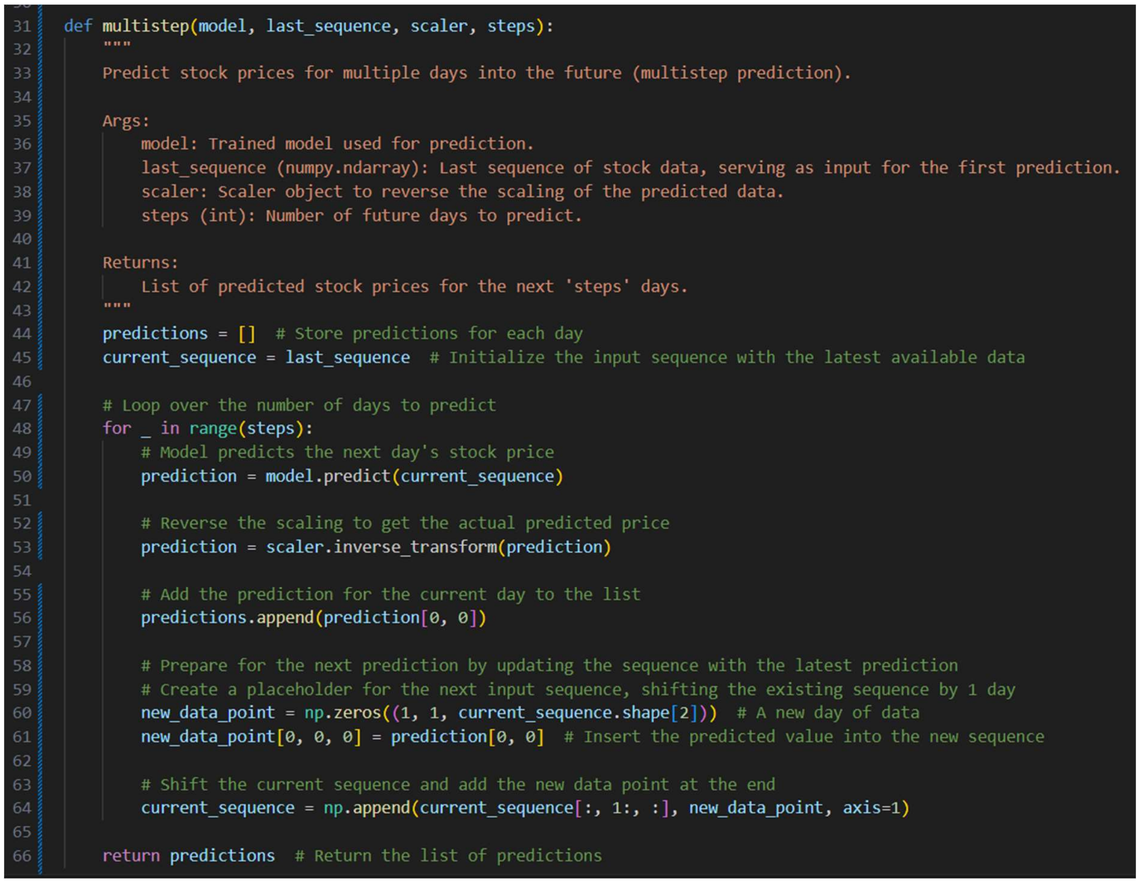
All explanations are within the screenshots of each part with appropriate comments.

1. Multistep Prediction

Implementation

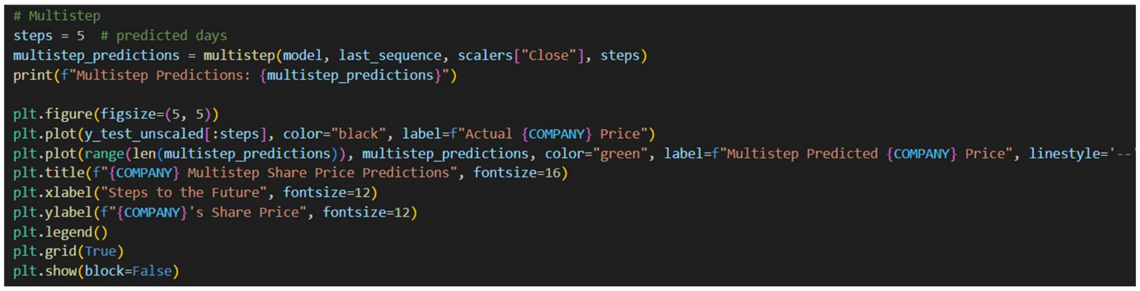
The multistep() function predicts the closing prices for multiple days (k steps) into the future based solely on previous closing prices. The model predicts the next price using the last sequence of stock data, updates the sequence with the new prediction, and continues to predict for the specified number of days.

Below is the implementation with appropriate comments describing the code's works.



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

    Args:
        model: Trained model used for prediction.
        last_sequence (numpy.ndarray): Last sequence of stock data, serving as input for the first prediction.
        scaler: Scaler object to reverse the scaling of the predicted data.
        steps (int): Number of future days to predict.

    Returns:
        List of predicted stock prices for the next 'steps' days.
    """
    predictions = []  # Store predictions for each day
    current_sequence = last_sequence  # Initialize the input sequence with the latest available data

    # Loop over the number of days to predict
    for _ in range(steps):
        # Model predicts the next day's stock price
        prediction = model.predict(current_sequence)

        # Reverse the scaling to get the actual predicted price
        prediction = scaler.inverse_transform(prediction)

        # Add the prediction for the current day to the list
        predictions.append(prediction[0, 0])

        # Prepare for the next prediction by updating the sequence with the latest prediction
        # Create a placeholder for the next input sequence, shifting the existing sequence by 1 day
        new_data_point = np.zeros((1, 1, current_sequence.shape[2]))  # A new day of data
        new_data_point[0, 0, 0] = prediction[0, 0]  # Insert the predicted value into the new sequence

        # Shift the current sequence and add the new data point at the end
        current_sequence = np.append(current_sequence[:, 1:, :], new_data_point, axis=1)

    return predictions  # Return the list of predictions
```

Multistep function

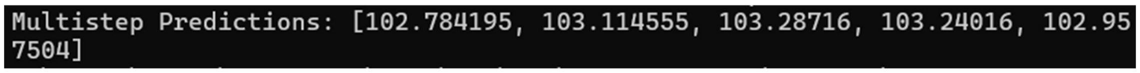


```python
# Multistep
steps = 5  # predicted days
multistep_predictions = multistep(model, last_sequence, scalers["Close"], steps)
print(f"Multistep Predictions: {multistep_predictions}")

plt.figure(figsize=(5, 5))
plt.plot(y_test_unscaled[:steps], color="black", label=f"Actual {COMPANY} Price")
plt.plot(range(len(multistep_predictions)), multistep_predictions, color="green", label=f"Multistep Predicted {COMPANY} Price", linestyle='--
plt.title(f"{COMPANY} Multistep Share Price Predictions", fontsize=16)
plt.xlabel("Steps to the Future", fontsize=12)
plt.ylabel(f"{COMPANY}'s Share Price", fontsize=12)
plt.legend()
plt.grid(True)
plt.show(block=False)
```

Multistep visualization

Results

The **Multistep Prediction** for 5 days resulted in the following predictions for the company's closing prices:



```
Multistep Predictions: [102.784195, 103.114555, 103.28716, 103.24016, 102.95
7504]
```

Prediction outcome

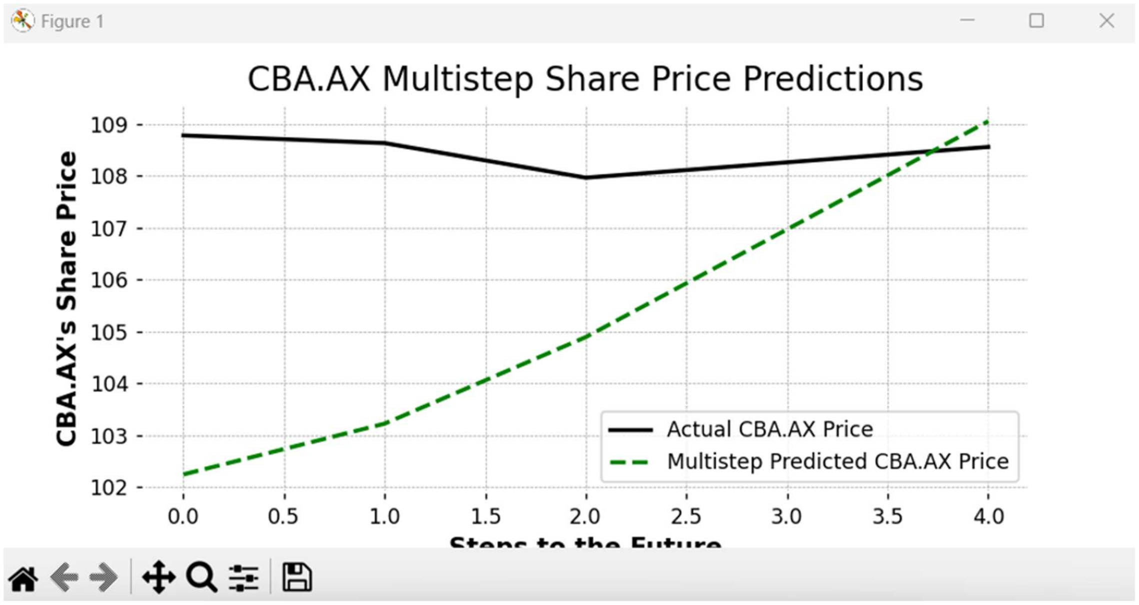
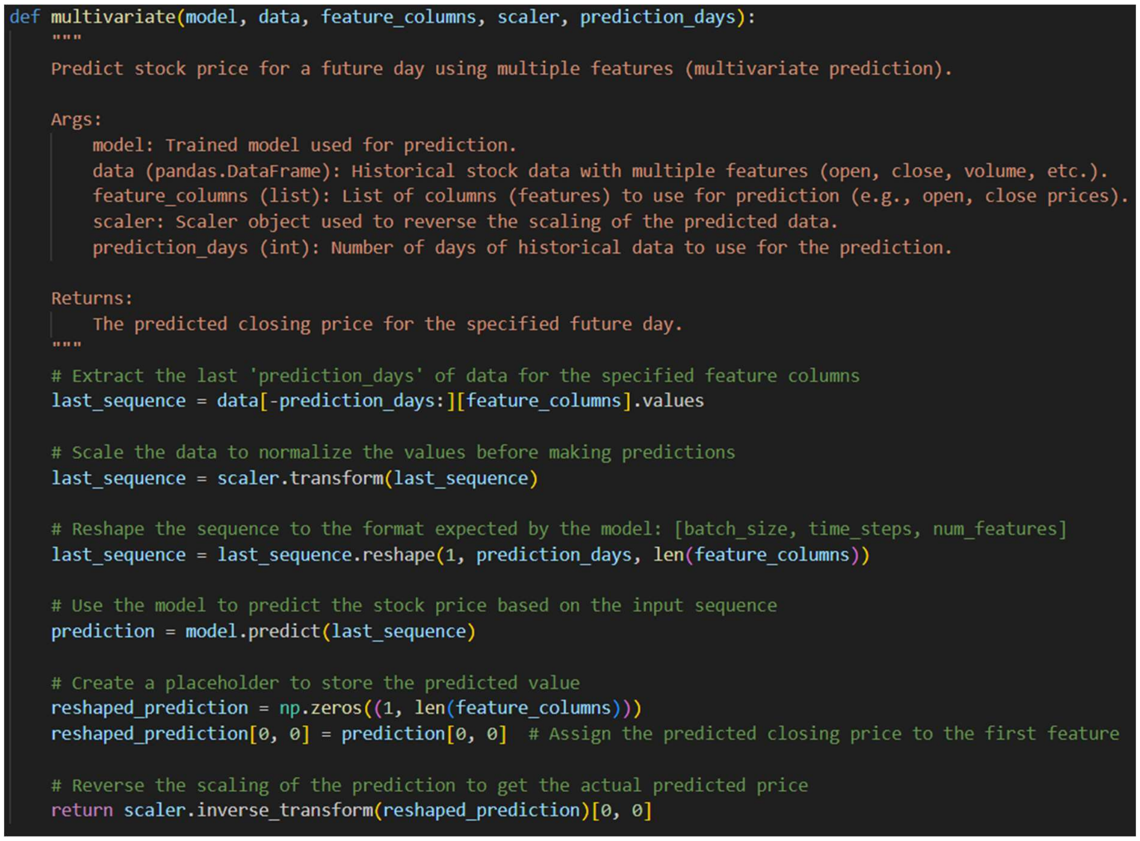


Figure screenshot

2. Multivariate Prediction

Implementation

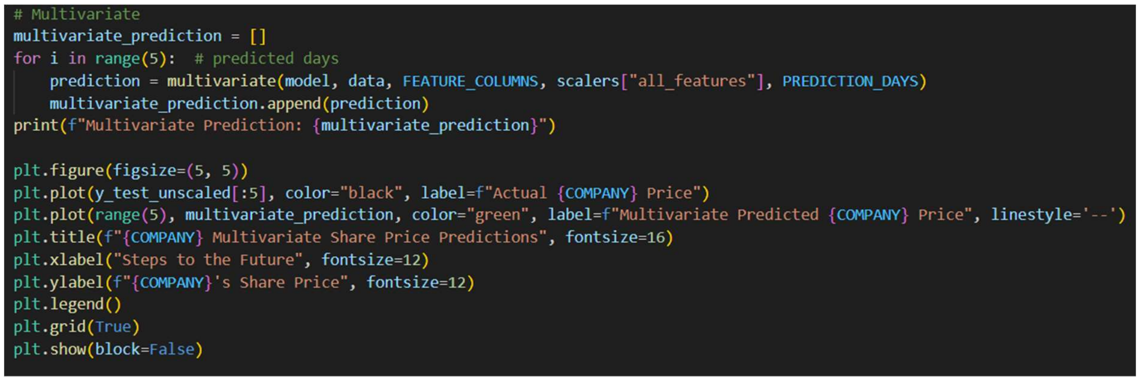
The multivariate() function predicts the closing price using multiple stock features (e.g., opening price, highest price, lowest price, trading volume). The function takes a sequence of days as input, transforms it using a scaler, reshapes the sequence, and makes a prediction based on multiple features.



```python
def multivariate(model, data, feature_columns, scaler, prediction_days):
    """
    Predict stock price for a future day using multiple features (multivariate prediction).

    Args:
        model: Trained model used for prediction.
        data (pandas.DataFrame): Historical stock data with multiple features (open, close, volume, etc.).
        feature_columns (list): List of columns (features) to use for prediction (e.g., open, close prices).
        scaler: Scaler object used to reverse the scaling of the predicted data.
        prediction_days (int): Number of days of historical data to use for the prediction.

    Returns:
        The predicted closing price for the specified future day.
    """
    # Extract the last 'prediction_days' of data for the specified feature columns
    last_sequence = data[-prediction_days:][feature_columns].values

    # Scale the data to normalize the values before making predictions
    last_sequence = scaler.transform(last_sequence)

    # Reshape the sequence to the format expected by the model: [batch_size, time_steps, num_features]
    last_sequence = last_sequence.reshape(1, prediction_days, len(feature_columns))

    # Use the model to predict the stock price based on the input sequence
    prediction = model.predict(last_sequence)

    # Create a placeholder to store the predicted value
    reshaped_prediction = np.zeros((1, len(feature_columns)))
    reshaped_prediction[0, 0] = prediction[0, 0]  # Assign the predicted closing price to the first feature

    # Reverse the scaling of the prediction to get the actual predicted price
    return scaler.inverse_transform(reshaped_prediction)[0, 0]
```

Multivariate function

The model is fed with a sequence containing multiple features, and it outputs a prediction for the closing price.

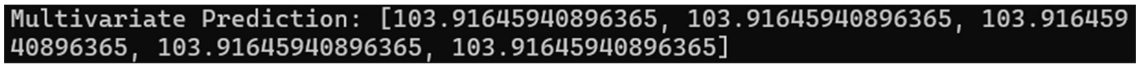


```python
# Multivariate
multivariate_prediction = []
for i in range(5):  # predicted days
    prediction = multivariate(model, data, FEATURE_COLUMNS, scalers["all_features"], PREDICTION_DAYS)
    multivariate_prediction.append(prediction)
print(f"Multivariate Prediction: {multivariate_prediction}")

plt.figure(figsize=(5, 5))
plt.plot(y_test_unscaled[:5], color="black", label=f"Actual {COMPANY} Price")
plt.plot(range(5), multivariate_prediction, color="green", label=f"Multivariate Predicted {COMPANY} Price", linestyle='--')
plt.title(f"{COMPANY} Multivariate Share Price Predictions", fontsize=16)
plt.xlabel("Steps to the Future", fontsize=12)
plt.ylabel(f"{COMPANY}'s Share Price", fontsize=12)
plt.legend()
plt.grid(True)
plt.show(block=False)
```

Multivariate visualization

Results

The **Multivariate Prediction** for the next 5 days gave the following predicted closing prices:



```
Multivariate Prediction: [103.91645940896365, 103.91645940896365, 103.916459
40896365, 103.91645940896365, 103.91645940896365]
```

Prediction outcome

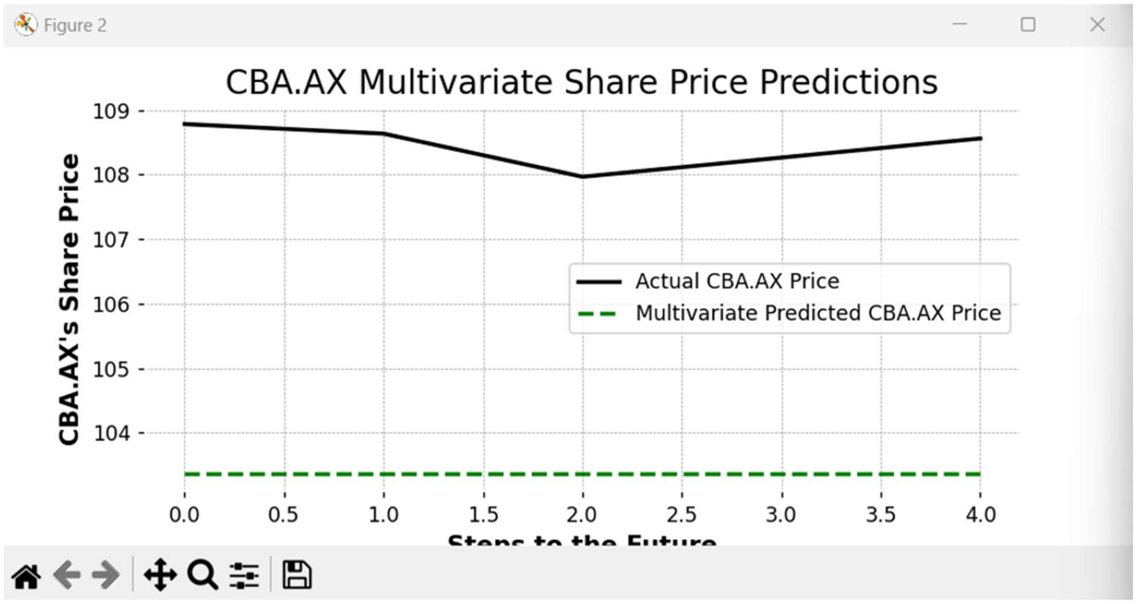
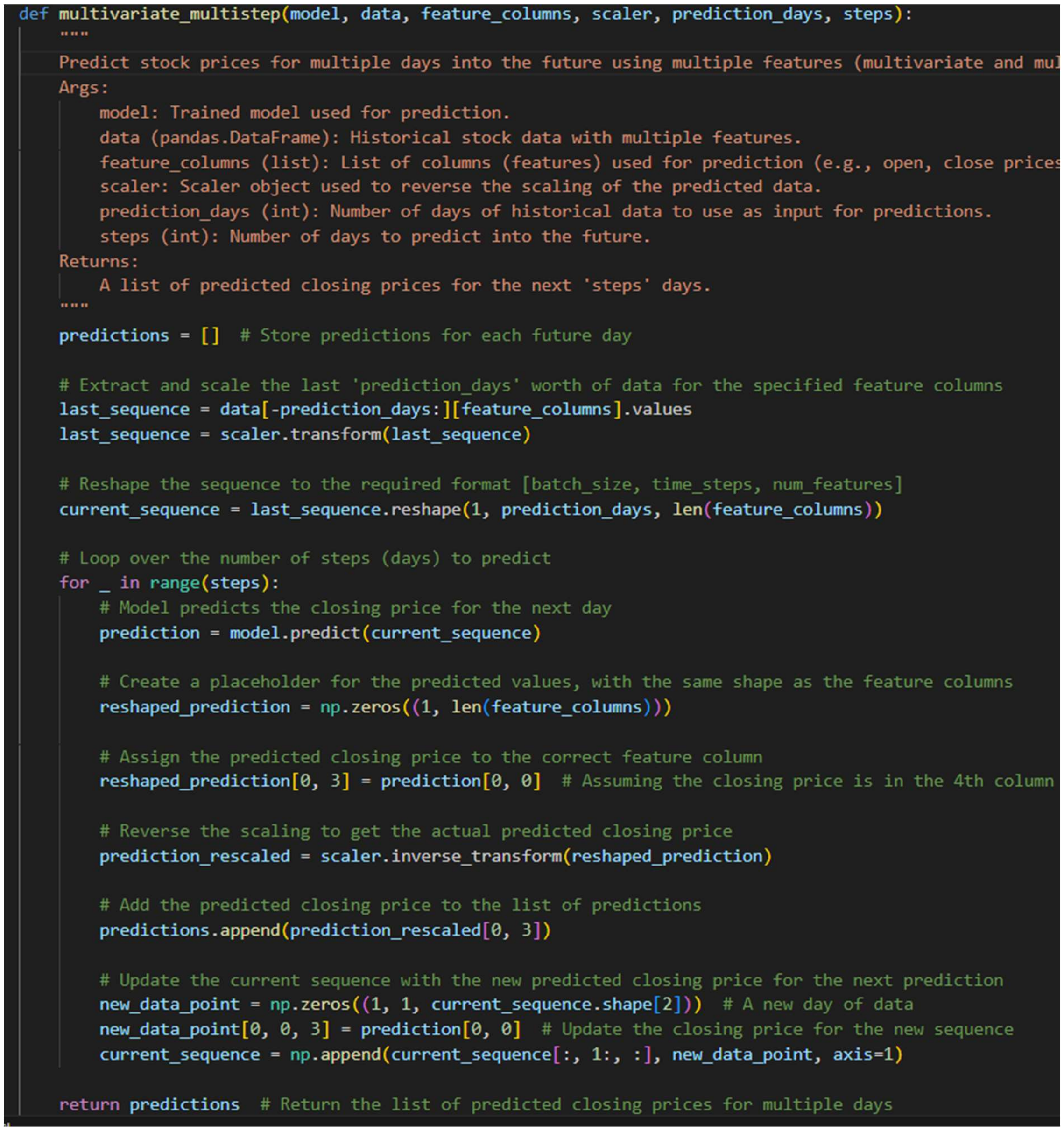


Figure Screenshot

3. Multivariate and Multistep Prediction

Implementation

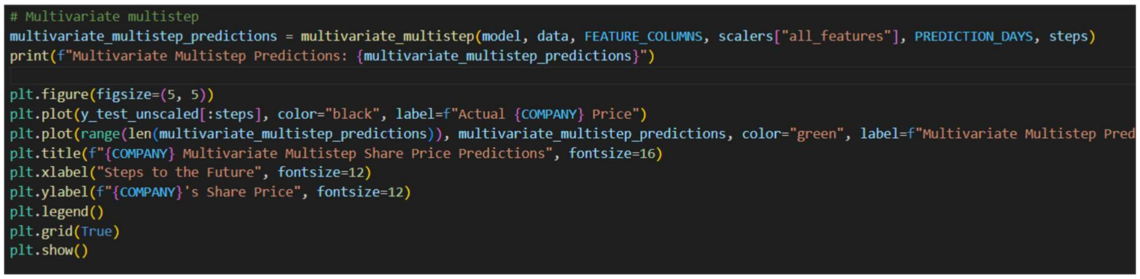
The multivariate_multistep() function combines both the multistep and multivariate approaches to predict closing prices for multiple days into the future using multiple stock features.



```python
def multivariate_multistep(model, data, feature_columns, scaler, prediction_days, steps):
    """
    Predict stock prices for multiple days into the future using multiple features (multivariate and mul
    Args:
        model: Trained model used for prediction.
        data (pandas.DataFrame): Historical stock data with multiple features.
        feature_columns (list): List of columns (features) used for prediction (e.g., open, close prices
        scaler: Scaler object used to reverse the scaling of the predicted data.
        prediction_days (int): Number of days of historical data to use as input for predictions.
        steps (int): Number of days to predict into the future.
    Returns:
        A list of predicted closing prices for the next 'steps' days.
    """
    predictions = []  # Store predictions for each future day

    # Extract and scale the last 'prediction_days' worth of data for the specified feature columns
    last_sequence = data[-prediction_days:][feature_columns].values
    last_sequence = scaler.transform(last_sequence)

    # Reshape the sequence to the required format [batch_size, time_steps, num_features]
    current_sequence = last_sequence.reshape(1, prediction_days, len(feature_columns))

    # Loop over the number of steps (days) to predict
    for _ in range(steps):
        # Model predicts the closing price for the next day
        prediction = model.predict(current_sequence)

        # Create a placeholder for the predicted values, with the same shape as the feature columns
        reshaped_prediction = np.zeros((1, len(feature_columns)))

        # Assign the predicted closing price to the correct feature column
        reshaped_prediction[0, 3] = prediction[0, 0]  # Assuming the closing price is in the 4th column

        # Reverse the scaling to get the actual predicted closing price
        prediction_rescaled = scaler.inverse_transform(reshaped_prediction)

        # Add the predicted closing price to the list of predictions
        predictions.append(prediction_rescaled[0, 3])

        # Update the current sequence with the new predicted closing price for the next prediction
        new_data_point = np.zeros((1, 1, current_sequence.shape[2]))  # A new day of data
        new_data_point[0, 0, 3] = prediction[0, 0]  # Update the closing price for the new sequence
        current_sequence = np.append(current_sequence[:, 1:, :], new_data_point, axis=1)

    return predictions  # Return the list of predicted closing prices for multiple days
```

Multivariate, multistep function

This function uses the multiple features to make predictions for multiple future days, updating the sequence with each new prediction.

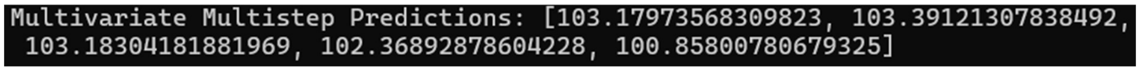


```python
# Multivariate multistep
multivariate_multistep_predictions = multivariate_multistep(model, data, FEATURE_COLUMNS, scalers["all_features"], PREDICTION_DAYS, steps)
print(f"Multivariate Multistep Predictions: {multivariate_multistep_predictions}")

plt.figure(figsize=(5, 5))
plt.plot(y_test_unscaled[:steps], color="black", label=f"Actual {COMPANY} Price")
plt.plot(range(len(multivariate_multistep_predictions)), multivariate_multistep_predictions, color="green", label=f"Multivariate Multistep Pred
plt.title(f"{COMPANY} Multivariate Multistep Share Price Predictions", fontsize=16)
plt.xlabel("Steps to the Future", fontsize=12)
plt.ylabel(f"{COMPANY}'s Share Price", fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

Multivariate, multistep visualization

Results

The **Multivariate Multistep Prediction** for 5 days yielded the following predictions:



```
Multivariate Multistep Predictions: [103.17973568309823, 103.39121307838492,
 103.18304181881969, 102.36892878604228, 100.85800780679325]
```

Prediction outcome

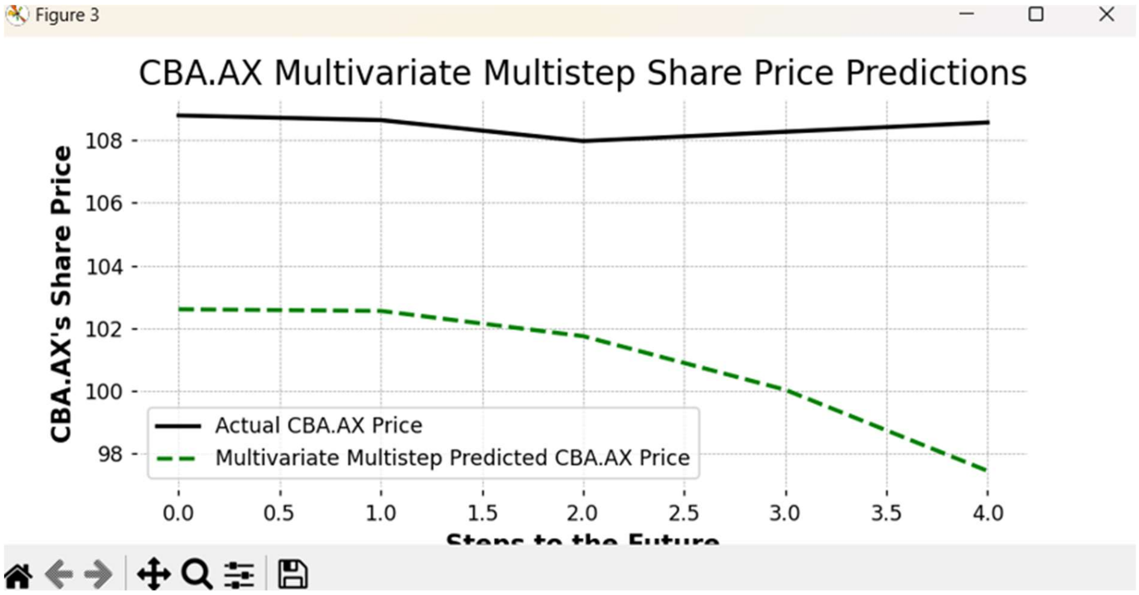


Figure screenshot

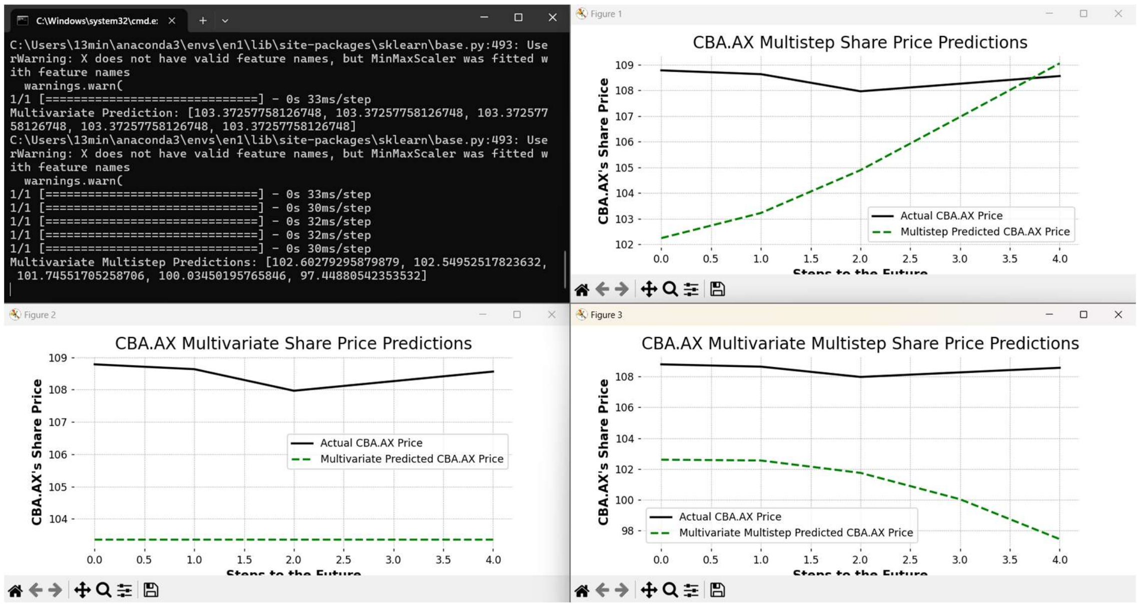
4. Comparison of Results

Below is a comparison of the predictions from each model:

Prediction Method	Day 1	Day 2	Day 3	Day 4	Day 5
Multistep	102.784195	103.114555	103.28716	103.24016	102.957504
Multivariate	103.91645940896365	103.91645940896365	103.91645940896365	103.91645940896365	103.91645940896365
Multivariate Multistep	103.17973568309823	103.39121307838492	103.18304181881969	102.36892878604228	100.85800780679325

Comparative Graphical Analysis

The graphs illustrate the performance of each model compared to the actual stock prices.


C:\Windows\system32\cmd.e:
Figure 1
CBA.AX Multistep Share Price Predictions
CBA.AX's Share Price
Steps to the Future
Actual CBA.AX Price
Multistep Predicted CBA.AX Price
Figure 2
CBA.AX Multivariate Share Price Predictions
CBA.AX's Share Price
Steps to the Future
Actual CBA.AX Price
Multivariate Predicted CBA.AX Price
Figure 3
CBA.AX Multivariate Multistep Share Price Predictions
CBA.AX's Share Price
Steps to the Future
Actual CBA.AX Price
Multivariate Multistep Predicted CBA.AX Price

```
C:\Users\13min\anaconda3\envs\en1\lib\site-packages\sklearn\base.py:493: Use
rWarning: X does not have valid feature names, but MinMaxScaler was fitted w
ith feature names
  warnings.warn(
1/1 [==============================] - 0s 33ms/step
Multivariate Prediction: [103.37257758126748, 103.37257758126748, 103.372577
58126748, 103.37257758126748, 103.37257758126748]
C:\Users\13min\anaconda3\envs\en1\lib\site-packages\sklearn\base.py:493: Use
rWarning: X does not have valid feature names, but MinMaxScaler was fitted w
ith feature names
  warnings.warn(
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
Multivariate Multistep Predictions: [102.60279295879879, 102.54952517823632,
 101.74551705258706, 100.03450195765846, 97.44880542353532]
```

Comparison of all 3 functions

Strengths and Weaknesses Comparison

Prediction Method	**Strengths**	**Weaknesses**
Multistep Prediction	- Predicts stock prices for multiple future days.	- Predictions can accumulate errors over time.
	- Updates input sequence dynamically with each prediction.	- Limited to a single feature (closing price), which may miss broader trends.
Multivariate Prediction	- Utilizes multiple features for a more comprehensive analysis.	- Predicts only one day at a time, lacking multi-day forecasts.
	- More robust to noise due to the inclusion of additional features.	- Same input sequence leads to constant predictions over multiple days.
Multivariate Multistep	- Combines the strengths of multivariate and multistep approaches.	- More complex implementation, requiring careful management of input data.

	- Provides dynamic forecasts with variations across multiple days.	- Still susceptible to error accumulation in long-term predictions.

3. Conclusion

In this task, I developed functions to solve multistep, multivariate, and combined multivariate multistep prediction problems. Each function adds more complexity to the stock price prediction process.

The multistep function predicts multiple future prices based on previous data, updating its input sequence with each new prediction. This allows the model to forecast several days ahead, simulating a realistic future scenario.

The multivariate function improves prediction accuracy by incorporating multiple features, such as opening price and trading volume, instead of relying solely on the closing price. This gives the model a broader view of the market.

The multivariate multistep function combines both approaches, predicting multiple days ahead using several features. The key challenge here was to maintain accurate input sequences while adjusting for both features and future steps.

Through these functions, I demonstrated how incorporating more data points and predicting multiple steps improves the model's performance, providing better insights into stock market trends.

4. References

Goodfellow, I., Bengio, Y., & Courville, A. (2016). ***Deep Learning***. MIT Press.

Retrieved from https://www.deeplearningbook.org/

MinMaxScaler Documentation (scikit-learn). (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Keras Sequential Model Documentation. (n.d.). Retrieved from https://keras.io/guides/sequential_model/

Inverse Transform Example (scikit-learn). (n.d.). Retrieved from https://scikit-learn.org/stable/modules/preprocessing.html#reversing-transformations

LSTM Layer Documentation. (n.d.). Retrieved from https://keras.io/api/layers/recurrent_layers/lstm/