ANSHUL VATS (104491909)

TASK – B.5 – REPORT

OVERVIEW –

This report outlines the implementation and functionality of the multivariate and multistep prediction functions within a LSTM model designed for stock prediction. The main goal is to utilize multiple features (multivariate) to predict stock prices multiple steps into the future(multistep).

1.MULTIVARIATE DATA PREPARATION

Multivariate prediction involves using the several features the model about the stock's behaviour. In this case, we use the following features-

- adjclose – adjusted close price
- volume
- open price
- high price
- low price

Function: load_data()

Goal: To load a historical stock data, scale the features and prepare it to be used by LSTM model training.

Inputs:

- ticker: Stock ticker symbol.
- n_steps: Number of previous time steps that EP3net uses to predict the next value.
- scale (bool): whether to scale features between 0 and 1
- shuffle: Shuff qpbele the data.
- lookup_step: The number of days ahead to predict for.
- split_by_date : Boolean, default = True If True the code will split data chronologically.
- test_size - fraction of data to be used for testing.
- feature_columns : list of feature to be included.

Key Steps:

- Retrieve information via the Yahoo Finance API
- Scale features if specified.
- Add a target variable (future) for the future stock price next lookup_step.
- Create Input and Targetsequences.

2. Multistep Prediction

The primary goal of multistep prediction is to predict multiple future values using observed data up to previous time points. This involves handling the input sequences with precision.

Function: get_multistep_predictions

Task: Produces several future stock price forecasts from the last observed chain.

Inputs:

- model: Trained LSTM model.
- last_sequence: The last sequence to be predicted against input feature wise.
- steps: Number of future steps you want to predict.
- scaler: Scaler object to be used on predictions in order to convert it back into its orginal scale.

Key Steps:

Prediction Loop -For each future step:

- next stock price is predicted by the model.
- Append the new prediction in the latest sequence.
- Do the above for the number of times you wanna take a step.

Inverse Scaling: transform predictions back to its original price scale using a scaler that see ed with the data.

3. Integration in Model Training

The multivariate and multistep are the functions of the API class that are integrated within the overall training and evaluation.

Training Loop:

A multivariate dataset is prepared and models with variations in layer depth and units are trained using it.

For each trained model:

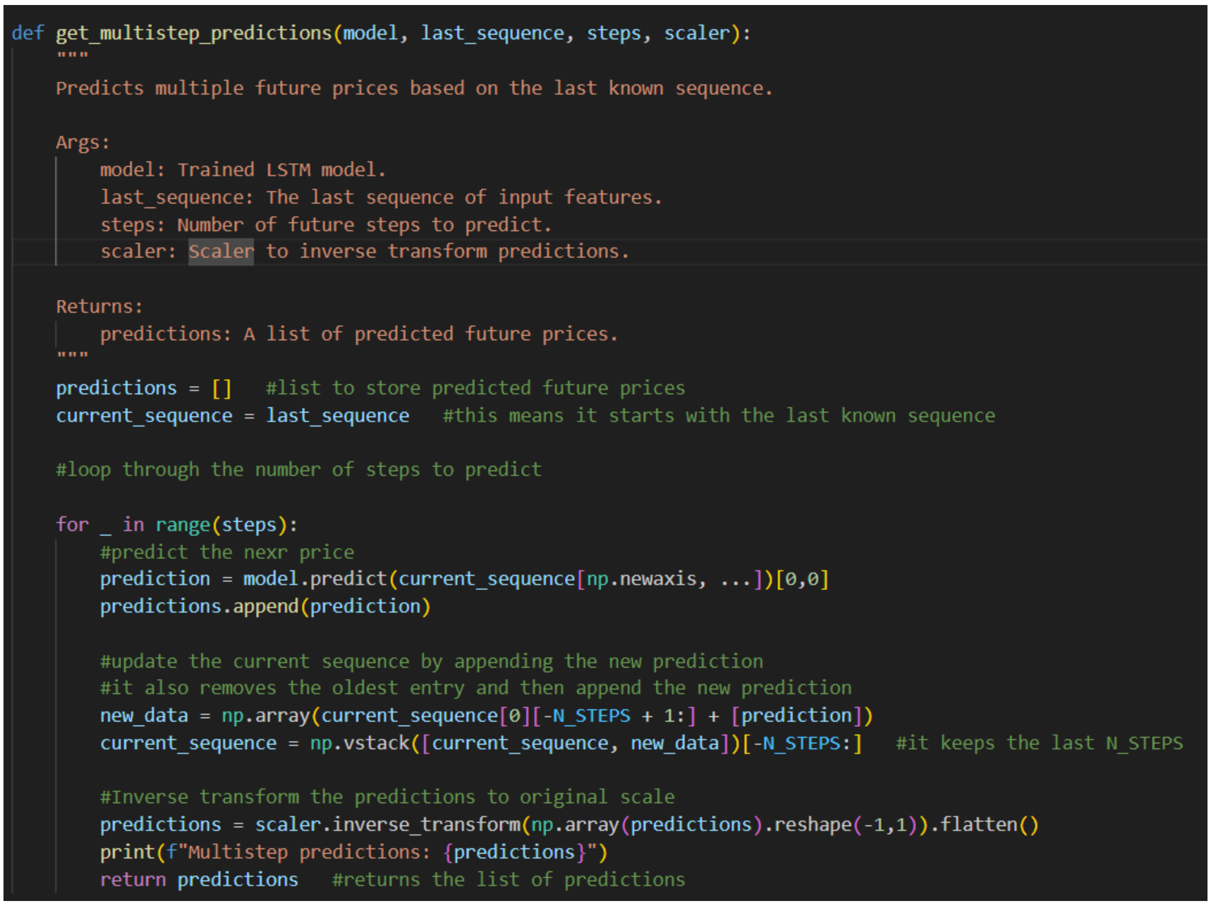
We create multistep predictions usingget_multistep_predictions()helper function.

Then, comparing predictions to true values and visualizing the results.

Conclusion

These functions are great as they allow the model to take in multiple features relating to the stock and make predictions accordingly. Putting this in the training pipeline, we guarantee that the LSTM model is capable of recognizing patterns between its multiple input variables over time which as a result makes the stock price movement forecasting process more accurate.

Codes



```python
def get_multistep_predictions(model, last_sequence, steps, scaler):
    """
    Predicts multiple future prices based on the last known sequence.

    Args:
        model: Trained LSTM model.
        last_sequence: The last sequence of input features.
        steps: Number of future steps to predict.
        scaler: Scaler to inverse transform predictions.

    Returns:
        predictions: A list of predicted future prices.
    """
    predictions = []   #list to store predicted future prices
    current_sequence = last_sequence   #this means it starts with the last known sequence

    #loop through the number of steps to predict

    for _ in range(steps):
        #predict the nexr price
        prediction = model.predict(current_sequence[np.newaxis, ...])[0,0]
        predictions.append(prediction)

        #update the current sequence by appending the new prediction
        #it also removes the oldest entry and then append the new prediction
        new_data = np.array(current_sequence[0][-N_STEPS + 1:] + [prediction])
        current_sequence = np.vstack([current_sequence, new_data])[-N_STEPS:]   #it keeps the last N_STEPS

        #Inverse transform the predictions to original scale
        predictions = scaler.inverse_transform(np.array(predictions).reshape(-1,1)).flatten()
        print(f"Multistep predictions: {predictions}")
        return predictions   #returns the list of predictions
```

Problems

Well , there is no as such problems in the file running in the terminal but the issue is , I have rather tried to add print statements for the multivariate and multistep function to give a satisfactory message to the user on the terminal that these parts of the code/file has been successfully executed but there is no such message being displayed on the terminal .

Apart from that , with research from machine-learning techniques on the web and the help of generative-AI(for learning purpose) , there is no such kind of error in the function.

Thank You.