Task C.5 - Machine Learning 2
COS318 - Intelligent Systems

Rahul Raju
Student ID: 105143065

Introduction

The progression of the stock_prediction.py codebase from version 0.4 to version 0.5 marks a significant transformation in both design and purpose. While v0.4 provided a functional framework for univariate single-step prediction of stock prices using configurable deep learning models, v0.5 redefined the project into a flexible forecasting platform capable of handling multistep, multivariate, and combined multivariate–multistep prediction.

The primary motivation for this upgrade was to address limitations of v0.4 that restricted experimentation and practical applicability. In real-world financial prediction tasks, investors and analysts require the ability to anticipate multiple days ahead while accounting for the interplay between various market features such as opening price, trading volume, and volatility. v0.5 was therefore designed to expand predictive scope and realism while preserving the modularity of the codebase.

Data Handling and Preprocessing

One of the most fundamental changes in v0.5 was the extension of the data processing pipeline to support multivariate inputs. In v0.4, only the closing price was scaled and used as the predictive feature. This design limited the model's view of market dynamics, as closing price alone cannot always explain intraday volatility or trading patterns.

In v0.5, the load_process_data function was enhanced to include multiple financial indicators - Open, High, Low, Close, Adjusted Close, and Volume. Each feature underwent independent MinMax scaling, ensuring that variations in magnitude (for example, the scale of trading volume compared to price movements) did not distort the model's learning process. This separation is critical because deep learning models are highly sensitive to feature scales. Without individual scaling, a large-magnitude feature such as trading volume could dominate the loss function, undermining the contribution of smaller-magnitude features like daily price changes.

This design choice worked effectively because it preserved the relative importance of each feature while ensuring numerical stability during training. By incorporating multiple features, the model could learn not only temporal dependencies within the closing price but also cross-feature dependencies such as the relationship between intraday volatility and subsequent price direction

Training Data Preparation

In v0.4, training data preparation was strictly limited to single-step targets. Each sequence of past values was mapped to a single next-day closing price. While this approach is straightforward, it fails to model extended horizons, which are essential in practice when decisions must account for several days into the future.

v0.5 introduced a new create_training_data design that supported multistep target generation. Instead of assigning only the next day's closing price as the label, the function can now assign multiple future values. This is achieved by slicing the scaled data across both the temporal lookback window and the chosen prediction horizon.

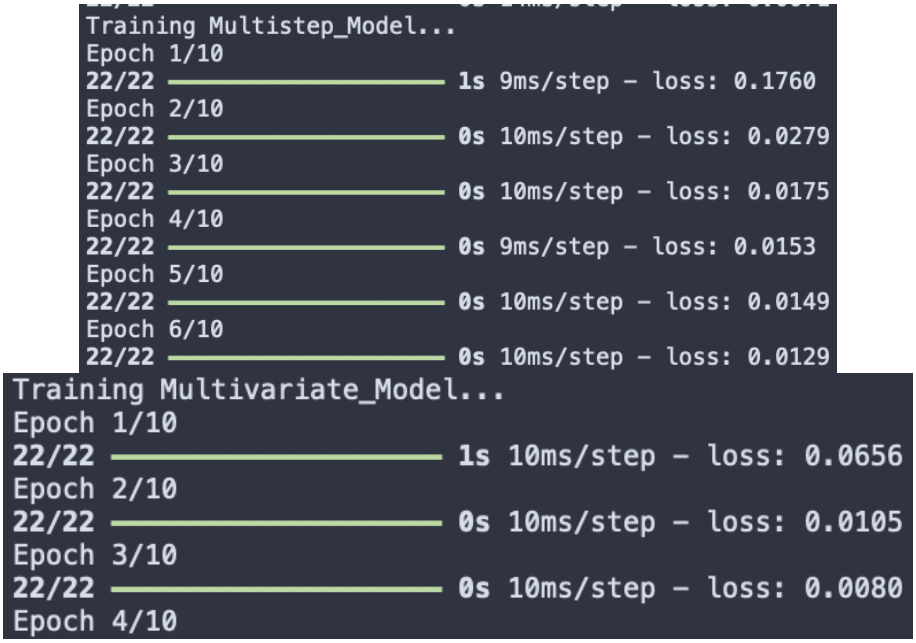
This approach was chosen because it reflects how prediction error behaves in practice: the further into the future we predict, the more uncertainty accumulates. By explicitly training the model to handle multiple future points, v0.5 makes the network more robust and realistic in its forecasting. Furthermore, this design creates a flexible framework for testing the impact of different prediction horizons (for example, one-day vs. five-day forecasts).

```
# Multivariate model
print("Training Multivariate_Model...")
multivariate_model = create_model(
    sequence_length=PREDICTION_DAYS,
    n_features=len(train_data.columns),  # Number of features
    units=50,
    cell=LSTM,
    n_layers=2,
    dropout=0.2,
    steps_ahead=1
)

# Multivariate multistep model
print("Training Multivariate_Multistep_Model...")
mv_ms_model = create_model(
    sequence_length=PREDICTION_DAYS,
    n_features=len(train_data.columns),  # Number of features
    steps_ahead=STEPS_AHEAD_MV_MS,
    units=50,
    cell=LSTM,
    n_layers=2,
    dropout=0.2
)

mv_ms_model.fit(x_train_mv_ms, y_train_mv_ms, epochs=10, batch_size=32)
```

```
Training Multivariate_Multistep_Model...
Epoch 1/10
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.1061
Epoch 2/10
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0193
Epoch 3/10
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0157
Epoch 4/10
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0129
Epoch 5/10
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0127
```



Model Architecture Modifications

Another critical enhancement in v0.5 was the adaptation of the model creation function. In v0.4, the model architecture was versatile in terms of layer type supporting LSTM, GRU, SimpleRNN, and bidirectional LSTMs, but the output layer was rigid and always producing a single scalar output through Dense(units=1). This design was sufficient for single-step prediction but incompatible with multistep targets.

In v0.5, the output layer was redesigned to dynamically adjust according to the prediction horizon. The final dense layer now scales with the value of steps_ahead, allowing the network to produce multiple outputs in a single forward pass. For example, if the user specifies a three-step forecast, the dense layer will output three consecutive predictions.

This modification was chosen because it integrates seamlessly with the revised data preparation strategy. By training the model to predict multiple time points simultaneously, the network learns not just pointwise mappings but also the underlying structure of short-term trajectories. This is particularly effective in financial data, where consecutive daily movements often follow correlated patterns.

```
# Multivariate prediction
print("Setting up multivariate prediction...")
scaled_multivariate_data = train_data.values  # All features
x_train_multivariate, y_train_multivariate = create_training_data(scaled_multivariate_data, PREDICTION_DAYS, steps_ahead=1, multivariate=True)

# Multivariate multistep prediction
print("Setting up multivariate multistep prediction...")
STEPS_AHEAD_MV_MS = 3  # Predict 3 days ahead
x_train_mv_ms, y_train_mv_ms = create_training_data(scaled_multivariate_data, PREDICTION_DAYS, steps_ahead=STEPS_AHEAD_MV_MS, multivariate=True)
```

```
if multivariate:
    # For multivariate prediction
    for x in range(prediction_days, len(scaled_data) - steps_ahead + 1):
        x_train.append(scaled_data[x-prediction_days:x])
        if steps_ahead == 1:
            y_train.append(scaled_data[x, 3])  # 3 is the index for Close price
        else:
            y_train.append(scaled_data[x:x+steps_ahead, 3])  # Predict next k days of Close prices
else:
    # For univariate prediction
    for x in range(prediction_days, len(scaled_data) - steps_ahead + 1):
        x_train.append(scaled_data[x-prediction_days:x])
        y_train.append(scaled_data[x:x+steps_ahead])  # Predict next k days

x_train, y_train = np.array(x_train), np.array(y_train)

if not multivariate:
    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))

return x_train, y_train
```

Expansion of Experimentation Scope

The overarching goal of v0.5 was to transform the codebase from a proof-of-concept baseline into an experimental laboratory for financial forecasting. v0.4 confined experiments to univariate single-step predictions. This was useful for establishing a baseline but provided limited insight into real-world applicability.

In v0.5, the scope was widened to support three major forecasting paradigms:

- Univariate multistep prediction: extending forecasts across multiple days based only on closing price.

- Multivariate single-step prediction: incorporating multiple features while predicting the next day only.

- Multivariate multistep prediction: combining both approaches, enabling the model to learn feature interactions while forecasting multiple steps ahead.

This design choice was deliberate: it allows users to systematically evaluate trade-offs across approaches, such as whether additional features truly improve single-day accuracy, or whether multistep forecasting is worth the increased computational cost.

Why These Changes Work

The changes introduced in v0.5 were not arbitrary; each was driven by clear limitations observed in v0.4 and grounded in time-series forecasting theory.

1. Multivariate input works because financial markets are multidimensional systems. Closing price cannot capture all relevant dynamics, whereas incorporating features like volume and intraday range allows the model to approximate hidden market states.

2. Multistep prediction works because it forces the network to learn patterns that span across multiple future points rather than extrapolating recursively, which compounds errors.

3. Dynamic output layers ensure that architectural design reflects the prediction task, avoiding mismatches between training labels and model predictions.

4. Expanded experimentation scope enables broader empirical testing, ensuring that conclusions are not limited to narrow baselines but extend to realistic forecasting scenarios.

Together, these enhancements transformed v0.5 into a more capable, flexible, and research-ready codebase, directly addressing the practical challenges of financial time series forecasting.