COS30018 - Option B - Task 5: Machine Learning 2

Student Name: Nguyen Duc Le Nguyen

Id: 104224493

Introduction

In this task, I focused on implementing multistep prediction models for stock prices, first using univariate data (just the closing prices) and then extending it to multivariate data (including features like Open, High, Low, etc.). Here's a step-by-step account of how I approached this problem.

Summary of Efforts and Code Explanation

1. Implementing Multistep Prediction for Univariate Data

1.1 Data Preparation

I started by downloading Apple stock data using Yahoo Finance. I then extracted the closing prices and checked if the data was stationary. This involved plotting the rolling mean and standard deviation and running the Dickey-Fuller test.

```
df = yf.download("AAPL", start="2015-01-01", end="2024-01-01")
company = 'Apple'
# Create a new dataframe with relevant features
stock_data = df[['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']]
stock_data
```

```
[**********100%%**********]  1 of 1 completed
```

Date	Open	High	Low	Close	Adj Close	Volume
2015-01-02	27.847500	27.860001	26.837500	27.332500	24.402174	212818400
2015-01-05	27.072500	27.162500	26.352501	26.562500	23.714722	257142000
2015-01-06	26.635000	26.857500	26.157499	26.565001	23.716959	263188400
2015-01-07	26.799999	27.049999	26.674999	26.937500	24.049517	160423600
2015-01-08	27.307501	28.037500	27.174999	27.972500	24.973555	237458000
...	...	...	...	...	...	...

Figure 1 - Stock Data

1.2 Data Transformation

To stabilize the variance and mean, I transformed the data by applying log and square root transformations and then differenced the data to achieve stationarity.

```
df_close_log = df_close.apply(np.log)
df_close_tf = df_close_log.apply(np.sqrt)
df_close_shift = df_close_tf - df_close_tf.shift()
df_close_shift.dropna(inplace=True)
```

Figure 2 - Transform Data

1.3 Data Preprocessing for LSTM

I use a function to preprocess the data for multistep LSTM models. This function slices the time series into input-output pairs for training.

```
def preprocess_multistep_lstm(sequence, n_steps_in, k, features):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps_in
        out_end_ix = end_ix + k
        # check if we are beyond the sequence
        if out_end_ix > len(sequence):
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix:out_end_ix]
        X.append(seq_x)
        y.append(seq_y)

    X = np.array(X)
    y = np.array(y)

    X = X.reshape((X.shape[0], X.shape[1], n_features))

    return X, y
```

Figure 3 - Preprocessing for LSTM

1.4 LSTM Model

I used a simple LSTM model and trained it with preprocessed data.

```python
def vanilla_multistep_LSTM():
    model = Sequential()
    model.add(LSTM(units=50, input_shape=(nb_days, n_features)))
    model.add(Dense(k))
    return model

model = vanilla_multistep_LSTM()
model.summary()
model.compile(optimizer='adam',
              loss='mean_squared_error',
              metrics=[tf.keras.metrics.MeanAbsoluteError()])

model.fit(X_train,
          y_train,
          epochs=15,
          batch_size = 32)
```

Figure 4 - LSTM model

1.5 Evaluation and Prediction

After training, I evaluated the model and visualized the predictions against the actual data. I applied inverse transformations to map predictions back to the original scale.

```python
# Prediction
y_pred = model.predict(X_test)

# Note: The first day start at index 0
the_day = 0
y_pred_days = y_pred[the_day,:]

plt.figure(figsize=(10,6))
plt.grid(True)
plt.plot(y_test[the_day,:],label='Orginal data - transformed')
plt.plot(y_pred_days, color='red',label='Predictions - transformed')
plt.xlabel('Time (days)')
plt.ylabel('Closing Prices amplitude in the transformed space')
plt.title('Original data vs predictions in the transformed space')
```

Figure 5 - Evaluation prediction

2. Implementing Multivariate Prediction

2.1 Data Preparation

I extended the prediction to include multiple features. I split the data into training, validation, and test sets and scaled the features.

```
df = yf.download("AAPL", start="2015-01-01", end="2024-04-04")
company = 'Apple'
df.reset_index(inplace=True)
# Create a new dataframe with relevant features
data = df[['Date','Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']]
# Select stock price records for the last five years, starting from 2019
data_5years = data[data["Date"].dt.year >= 2019]
# Check filtered data shape
data_5years.shape
# Define selected features and target attribute
features = ["Open", "High", "Low","Close", "Adj Close", "Volume"]
target = "Close"
```

Figure 6 - Data Preparation

```
# Initialize scaler with range [0,1]
sc = MinMaxScaler(feature_range=(0,1))

# Fit and transform scaler to training set
data_train_scaled = sc.fit_transform(data_train)

# Transform validating and testing datasets
data_validate_scaled = sc.transform(data_validate)
data_test_scaled = sc.transform(data_test)
```

Figure 7 - Split Data

2.2 Constructing LSTM Data

I used a function to create input-output pairs for the LSTM model using multivariate data.

```python
# Define a method to construct the input data X and Y
def construct_lstm_data(data, sequence_size, target_attr_idx):

    # Initialize constructed data variables
    data_X = []
    data_y = []

    # Iterate over the dataset
    for i in range(sequence_size, len(data)):
        data_X.append(data[i-sequence_size:i,0:data.shape[1]])
        data_y.append(data[i,target_attr_idx])

    # Return constructed variables
    return np.array(data_X), np.array(data_y)
```

Figure 8 - Construct Data for LSTM

2.3 LSTM Model Implementation

I used a more complex LSTM model with two LSTM layers and trained it on the multivariate data.

```python
# Initializing the model
regressor = Sequential()

# Add input layer
regressor.add(Input(shape=(X_train.shape[1], X_train.shape[2])))

# Add first LSTM layer and dropout regularization layer
regressor.add(LSTM(units = 100, return_sequences = True))
regressor.add(Dropout(rate = 0.2))

# Add second LSTM layer and dropout regularization layer
regressor.add(LSTM(units = 100, return_sequences = True))
regressor.add(Dropout(rate = 0.2))

# Add third LSTM layer and dropout regularization layer
regressor.add(LSTM(units = 100, return_sequences = True))
regressor.add(Dropout(rate = 0.2))

# Add forth LSTM layer and dropout regularization layer
regressor.add(LSTM(units = 100))
regressor.add(Dropout(rate = 0.2))

# Add last dense layer/output layer
regressor.add(Dense(units = 1))

# Compiling the model
regressor.compile(optimizer = "adam", loss="mean_squared_error")
```

Figure 9 - LSTM model

2.4 Prediction and Visualization

I predicted stock prices using the trained model and visualized the results.

```
# Plot actual and predicted price
plt.figure(figsize=(18,6))
plt.plot(data_train_dates[sequence_size:,], y_train_inv, label="Training Data", color=train_actual_color)
plt.plot(data_train_dates[sequence_size:,], y_train_predict_inv, label="Training Predictions", linewidth=1, color=train_predicted_color)

plt.plot(data_validate_dates, y_validate_inv, label="Validation Data", color=validate_actual_color)
plt.plot(data_validate_dates, y_validate_predict_inv, label="Validation Predictions", linewidth=1, color=validate_predicted_color)

plt.plot(data_test_dates, y_test_inv, label="Testing Data", color=test_actual_color)
plt.plot(data_test_dates, y_test_predict_inv, label="Testing Predictions", linewidth=1, color=test_predicted_color)

plt.title("Apple Stock Price Predictions With LSTM")
plt.xlabel("Time")
plt.ylabel("Stock Price (USD)")
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=2))
plt.xticks(rotation=45)
plt.legend()
plt.grid(color="lightgray")
plt.show()
```

Figure 10 - Visualize the results

Results Summary

Stock information before predicting

Experiment 1: Multistep Prediction

Results:

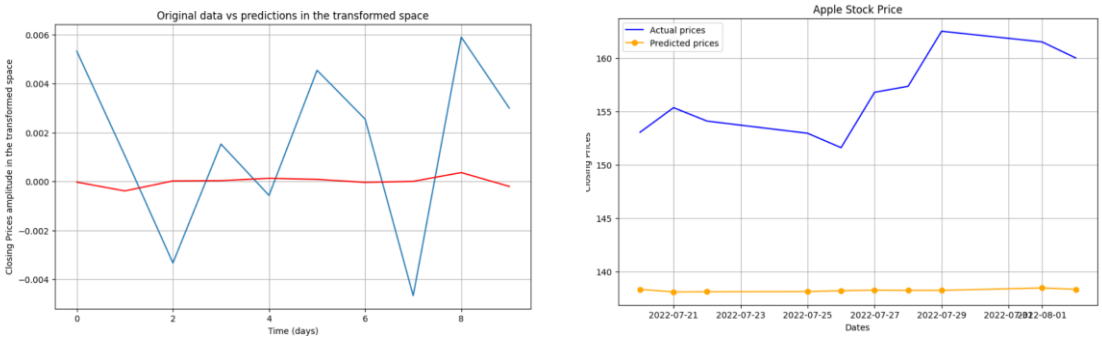

Figure 11 - Multistep Results

Experiment 2: Multivariate Prediction

Results:

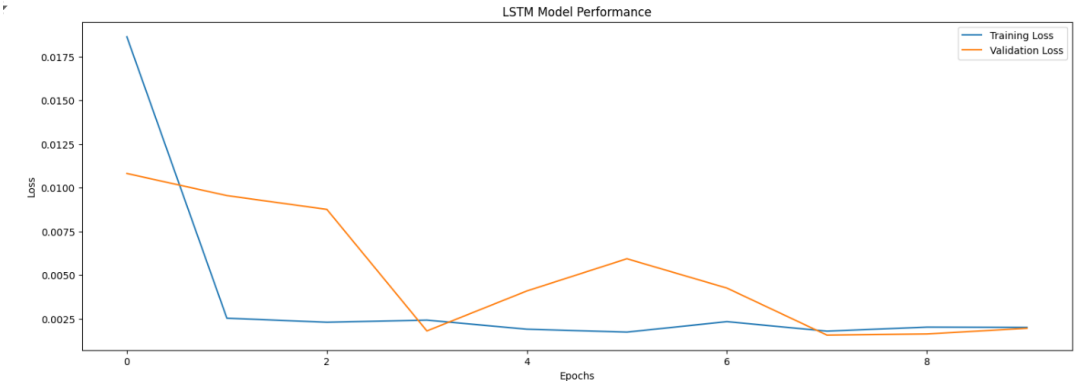

Figure 11 - Model Performance

Figure 12 - Visualize Results

Conclusion

In this task, I implemented both univariate and multivariate LSTM models for stock price prediction. I transformed and preprocessed the data, built and trained LSTM models, and evaluated their performance. Visualizing the predictions helped me understand how well the models performed.

References

SinanW. (2021). *LSTM Stock Price Prediction*. GitHub. Retrieved from https://github.com/sinanw/lstm-stock-price-prediction

Thurson, T. (2023). *Stock Price Prediction with LSTM: Multi-Step LSTM*. Kaggle. Retrieved from https://www.kaggle.com/code/thibauthurson/stock-price-prediction-with-lstm-multi-step-lstm