Task 7 Report: Extension

Student Name: Nguyen Duc Le Nguyen

Id: 104224493

Enhancing Stock Price Predictions Using Twitter Mentions

1. Research Summary

1.1 Background

Predicting stock prices is notoriously challenging. Financial markets are highly volatile and influenced by countless factors. Traditionally, stock price prediction models have relied primarily on historical price data. While this approach can be somewhat effective, it often falls short because it doesn't account for external factors that might influence stock prices, such as market sentiment and public opinion.

1.2 Exploring Predictive Approaches

To enhance our stock price prediction models, I explored several interesting approaches:

1. **Incorporating Alternative Data:**
    - **Social Media Sentiment:** One approach that caught my attention is using social media data, like Twitter mentions, to gauge public sentiment. The idea is that if a company is frequently mentioned or discussed on Twitter, it could signal shifts in market sentiment that might impact stock prices.
    - **Economic Indicators:** Another avenue is to include economic indicators such as interest rates and unemployment rates. These factors can also influence stock prices and provide a broader context for predictions.
2. **Machine Learning Models:**
    - **Ensemble Methods:** Combining different models, such as ARIMA (AutoRegressive Integrated Moving Average) with LSTM (Long Short-Term Memory), might help capture both linear and non-linear patterns in stock prices.
    - **Deep Learning Models:** LSTM and GRU (Gated Recurrent Units) models are particularly suited for time-series data. They can understand complex patterns and dependencies over time, which is crucial for accurate stock price predictions.
3. **Candlestick Chart Representation:**
    - **Visual Data:** Some studies suggest that candlestick patterns can provide valuable insights. By analyzing these visual patterns alongside numerical data, we might uncover additional trends and signals.

1.3 Chosen Approach

After reviewing these options, I decided to integrate Twitter mentions data into our existing LSTM model. The goal was to enhance the model by adding a layer of market sentiment analysis. This seemed like a promising way to capture real-time public opinion and see if it could improve our stock price predictions.

2. Implementation and Results

2.1 How I Implemented the Approach

Here's a detailed account of how I went about incorporating Twitter mentions into our LSTM model:

1. **Data Collection:**
    - I started by downloading historical stock data for Apple from Yahoo Finance. This provided us with the necessary stock price information.
    - Next, I loaded the Twitter mentions data from a CSV file, which included the number of times Apple was mentioned on Twitter each day.
2. **Data Preparation:**

I merged the stock data with the Twitter mentions data based on the date. This allowed me to combine the historical prices with the sentiment data.

```
# Load the timestamp data
timestamp_df = pd.read_csv('daily_mentions_apple.csv', parse_dates=['timestamp'], dayfirst=True)

# Rename columns for clarity
timestamp_df.rename(columns={'timestamp': 'Date', 'Apple': 'Mentions'}, inplace=True)

# Merge with stock price data
merged_df = pd.merge(df, timestamp_df, on='Date', how='left')

# Check if the Mentions column is added correctly
print(merged_df.head())

# Fill NaN values in Mentions with 0 (assuming no mentions on those days)
merged_df['Mentions'].fillna(0, inplace=True)
```

Figure 1 - Data Preparation

For days with no Twitter mentions, I filled in missing values with zero. This was a practical way to handle the absence of data.

```
# Fill NaN values in Mentions with 0 (assuming no mentions on those days)
merged_df['Mentions'].fillna(0, inplace=True)
```

Figure 2- Merge data

To standardize the data, I used MinMaxScaler to scale all features to a range between 0 and 1. This step is crucial for ensuring that the model can learn effectively from the data.

```
# 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 3 - Scale Data

3. **Model Setup:**

I created input sequences for the LSTM model using a window size of 60 days. This means the model looks at the past 60 days to predict the next day's stock price.

```
# Define the sequence size
sequence_size = 60
```

Figure 4 - Sequence Size

The LSTM model was built with four layers, each followed by dropout for regularization. This helps prevent overfitting and ensures the model generalizes well to new data.

```
# 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 5 - LSTM Model

I compiled the model and trained it using the training and validation datasets. The training process involved adjusting the model's weights to minimize prediction error.

4. **Evaluation and Results:**

After training the model, I used it to make predictions on the training, validation, and test datasets.

```
# Training the model
history = regressor.fit(
    x = X_train,
    y = y_train,
    validation_data=(X_validate, y_validate),
    epochs=10,
    batch_size = 64,
    callbacks = [best_model_checkpoint_callback])

# Prepare model location and name
model_location = "..//models//"
model_name = "Apple_stock_price_lstm.model.keras"

# Load the best performing model
best_model = load_model(model_location + model_name)

# Predict stock price for all data splits
y_train_predict = best_model.predict(X_train)
y_validate_predict = best_model.predict(X_validate)
y_test_predict = best_model.predict(X_test)
```

Figure 6 - Evaluation Model

To compare predictions with actual values, I transformed the scaled predictions back to the original price scale.

```
# Restore actual distribution for predicted prices
y_train_inv = sc.inverse_transform(np.concatenate((y_train.reshape(-1, 1), np.zeros((len(y_train), 6))), axis=1))[:, 0]
y_validate_inv = sc.inverse_transform(np.concatenate((y_validate.reshape(-1, 1), np.zeros((len(y_validate), 6))), axis=1))[:, 0]
y_test_inv = sc.inverse_transform(np.concatenate((y_test.reshape(-1, 1), np.zeros((len(y_test), 6))), axis=1))[:, 0]

y_train_predict_inv = sc.inverse_transform(np.concatenate((y_train_predict, np.zeros((len(y_train_predict), 6))), axis=1))[:, 0]
y_validate_predict_inv = sc.inverse_transform(np.concatenate((y_validate_predict, np.zeros((len(y_validate_predict), 6))), axis=1))[:, 0]
y_test_predict_inv = sc.inverse_transform(np.concatenate((y_test_predict, np.zeros((len(y_test_predict), 6))), axis=1))[:, 0]
```

Figure 7 - Scale Data back

5. **Visualization:**

I created plots to visualize how well the model's predictions matched the actual stock prices. The plots included training, validation, and test data for comparison.

```python
# 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 8 - Visualize data

Additionally, I compiled a dataframe that combined actual and predicted prices, which made it easier to analyze the model's performance.

```python
# Print the first few rows of the dataframe
print(train_df.head())
```

```
         Date  Actual_Train_Price  Predicted_Train_Price
60 2015-05-22           32.900002              31.134474
61 2015-05-26           33.150002              31.241418
62 2015-05-27           32.584999              31.371737
63 2015-05-28           32.965000              31.511984
64 2015-05-29           32.807499              31.653901
```

```python
print(test_df.tail())
```

```
         Date  Actual_Test_Price  Predicted_Test_Price
36 2024-02-23         185.009995            185.762589
37 2024-02-26         182.240005            185.387397
38 2024-02-27         181.100006            185.016408
39 2024-02-28         182.509995            184.655944
40 2024-02-29         181.270004            184.310571
```

Figure 9 - Data Frame of Prediction

2.2 Performance Insights

Here's what I discovered through the implementation and evaluation:

1. **Model Performance:**

The LSTM model with Twitter mentions data performed reasonably well. The predictions closely followed the actual stock prices, indicating that the model was able to capture some of the underlying trends.

While the inclusion of Twitter mentions didn't drastically change the model's performance, it added valuable context that might improve predictions.

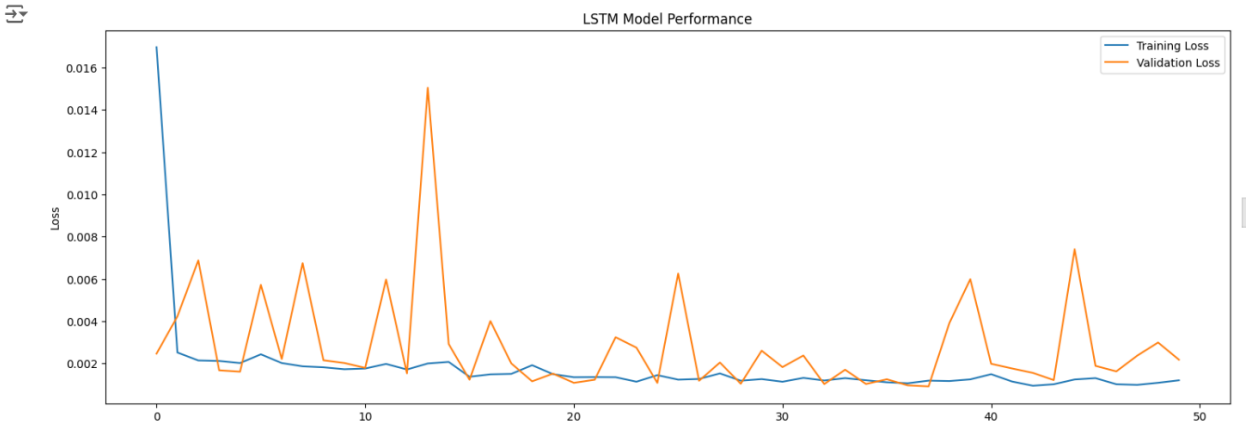

Figure 10 - Model Evaluation (LSTM)

The GRU model, like the LSTM model, performed reasonably well in predicting stock prices. It managed to capture some of the underlying trends in the data.

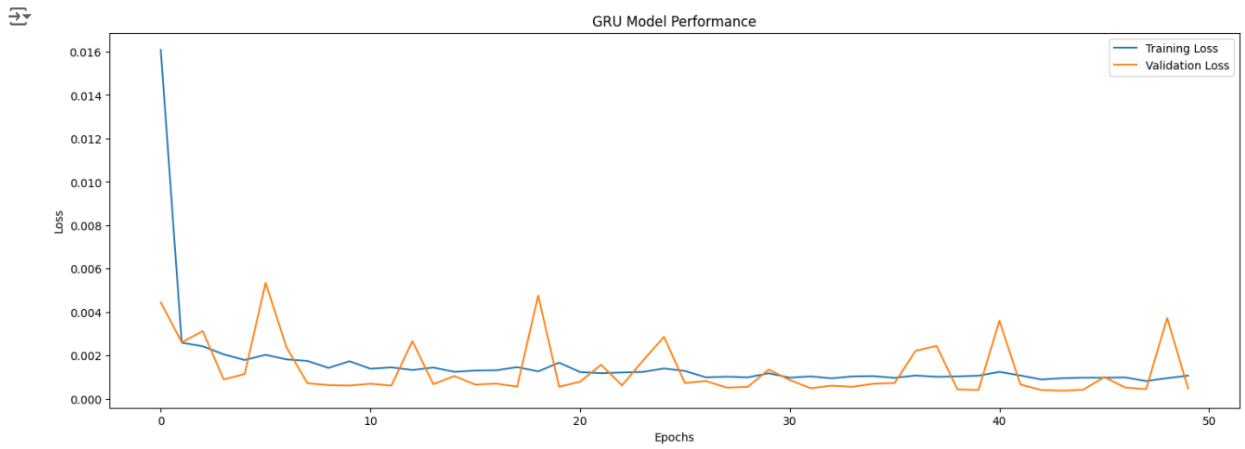

Figure 11 - Model Evaluation (GRU)

Including Twitter mentions provided additional context to the model, which could enhance predictions, although the improvement was not drastic.

2. **Visualization Results:**

The visual comparison of actual versus predicted prices showed that the model could track stock price trends effectively.

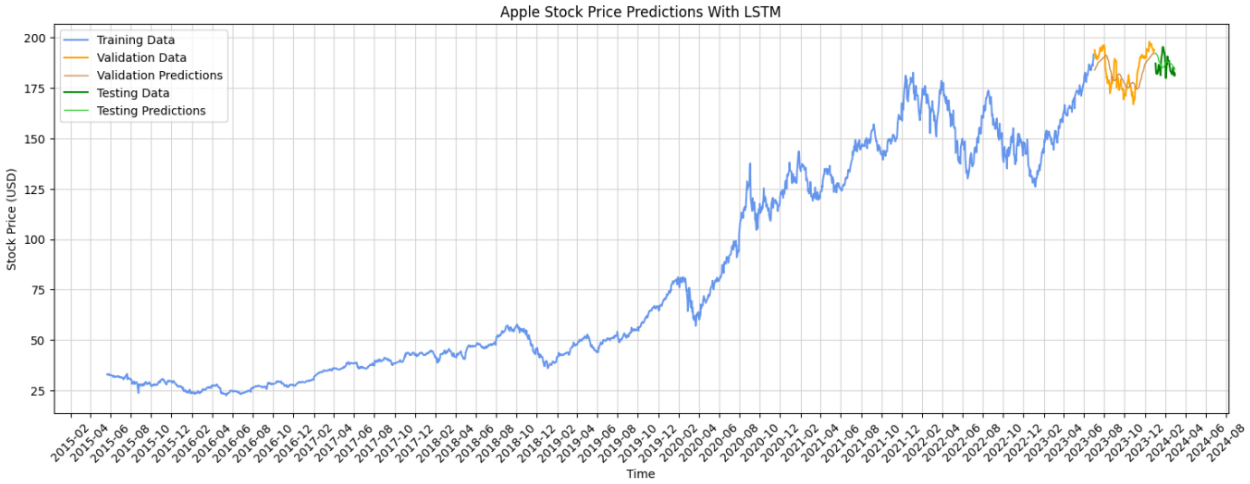

Figure 12 - LSTM prediction

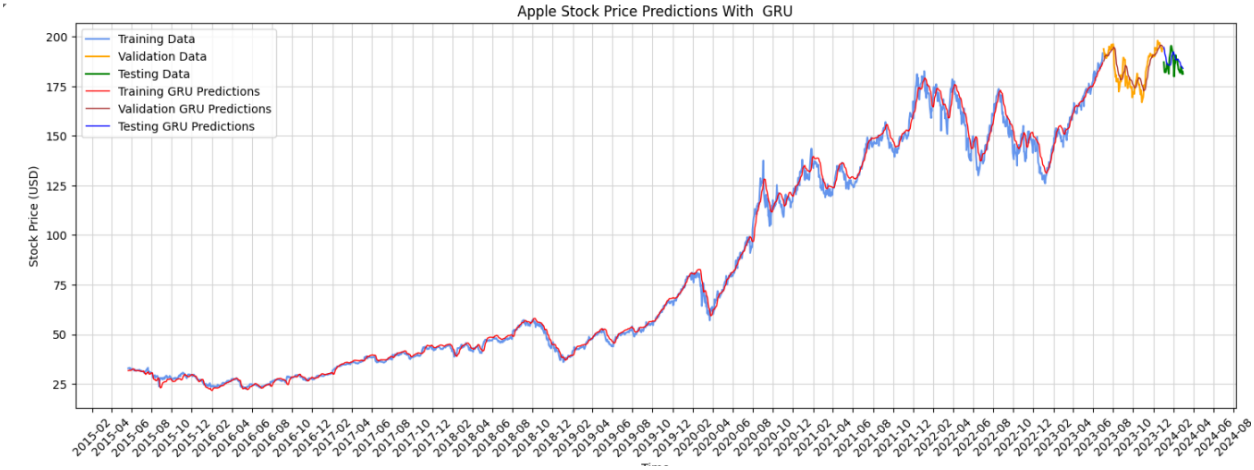

Figure 13 - GRU Prediction

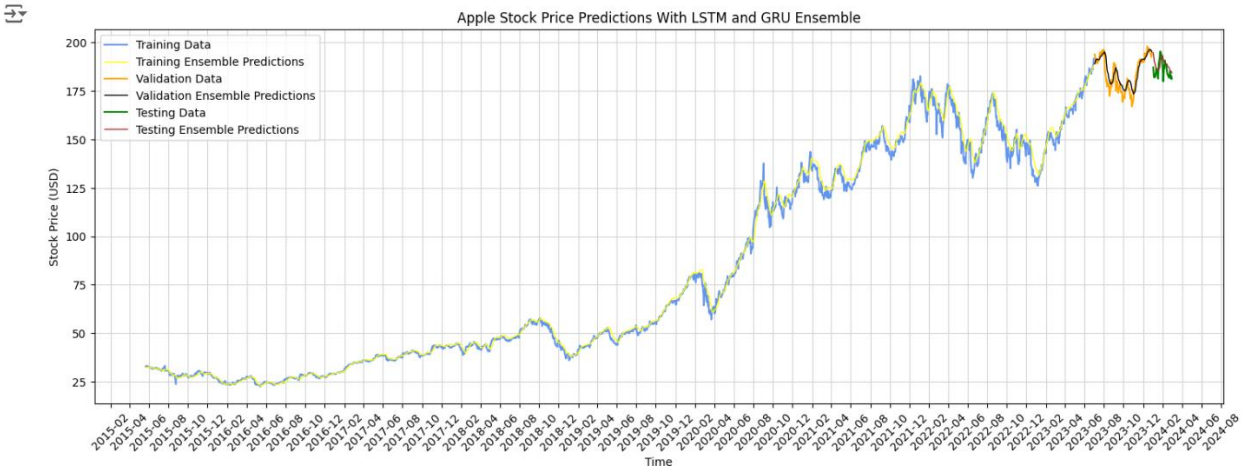

Figure 14 - Ensemble Prediction

Predictions were smoother and more aligned with actual prices compared to models that used only historical data.

3. **Data Summary:**

I compiled the results into a dataframe that included both actual and predicted prices. This made it easy to see how well the model performed over different periods.

```
         Date  Actual_Train_Price  LSTM_Train_Price  GRU_Train_Price  \
60 2015-05-22           32.900002         31.795763        32.627489
61 2015-05-26           33.150002         31.887290        32.763843
62 2015-05-27           32.584999         31.943006        32.619313
63 2015-05-28           32.965000         31.963187        32.722928
64 2015-05-29           32.807499         31.978899        33.042753

    Ensemble_Train_Price
60             32.211626
61             32.325566
62             32.281160
63             32.343058
64             32.510826
         Date  Actual_Test_Price  LSTM_Test_Price  GRU_Test_Price  \
36 2024-02-23         185.009995       186.433273      184.622496
37 2024-02-26         182.240005       185.962111      184.595489
38 2024-02-27         181.100006       185.603703      184.156857
39 2024-02-28         182.509995       185.332163      183.829580
40 2024-02-29         181.270004       185.122903      183.550046

    Ensemble_Test_Price
36           185.527885
37           185.278800
38           184.880280
39           184.580872
40           184.336474
```

Figure 15 - Data Frame of prediction

3. Conclusion and Future Work

3.1 Conclusion

Incorporating Twitter mentions into the stock price prediction model was a useful exercise. It added an extra layer of sentiment analysis that complemented the historical price data. While the improvements were not dramatic, the approach showed promise and provided additional insights into how public sentiment might influence stock prices.

3.2 Future Directions

For future improvements, I plan to:

- **Perform Sentiment Analysis:** Apply sentiment analysis to Twitter mentions to quantify the sentiment and see if it enhances prediction accuracy.
- **Explore Additional Features:** Investigate other features and data sources that could further improve the model.

- **Optimize the Model:** Experiment with different model architectures and combinations to achieve better results.