Big Data & Data Mining

Tesla & Hyundai Stock Price Prediction

Semester Project

Author: *Muhammad Talha Zulfiqar*

Introduction

In this task, we aim to analyze and compare the stock prices of Tesla and Hyundai. We have two separate datasets containing historical stock price data for both companies. The objective is to merge the datasets, visualize the stock prices of both companies over time, and make predictions specifically for Tesla's stock price.

Model Used

For the stock price prediction, we use a linear regression model. Linear regression is a simple yet effective machine learning algorithm that assumes a linear relationship between the input variables (in this case, the date) and the output variable (the closing price). By fitting a linear regression model to the training data, we can learn the underlying trend and make predictions for the test data

By merging the datasets and performing visualization and prediction, we can gain insights into the historical stock prices of both companies and make predictions for future stock prices, specifically for Tesla. However, it's important to note that financial predictions are subject to various factors, and further analysis and evaluation are essential before making any investment decisions based on the predictions

1. **Import the necessary libraries**

```
In [2]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        from sklearn.preprocessing import MinMaxScaler
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, LSTM
```

2. Load the datasets

In [3]:
```
# Load the datasets
tesla_df = pd.read_csv('./datasets/Tesla.csv - Tesla.csv.csv')
hyundai_df = pd.read_csv('./datasets/005380.KS.csv')
```

In [4]:
```
print("Tesla Stock Prices:")
tesla_df.head(3)
```

```
Tesla Stock Prices:
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	6/29/2010	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	6/30/2010	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	07/01/2010	25.000000	25.92	20.270000	21.959999	21.959999	8218800

In [5]:
```
print("\nHyundai Stock Prices:")
hyundai_df.head(3)
```

```
Hyundai Stock Prices:
```

Out[5]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	01/04/2016	147500.0	148000.0	143500.0	144000.0	117625.109375	445332
1	01/05/2016	143000.0	145000.0	142000.0	143500.0	117216.703125	530496
2	01/06/2016	144000.0	145000.0	139000.0	140000.0	114357.726562	769406

3. Explore the data

In [6]:
```
print("\nData Types of Tesla datasets:")
print(tesla_df.dtypes)
```

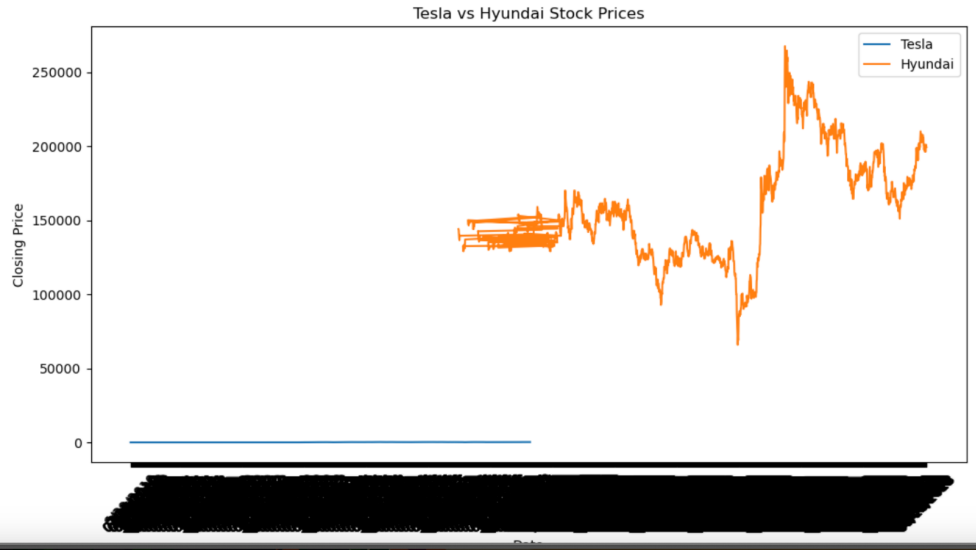
```
Data Types of Tesla datasets:
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

In [8]:
```
print("\nData Types of Hyundai datasets:")
print(tesla_df.dtypes)
```

```
Data Types of Hyundai datasets:
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

4. Visualize the data

In [9]:
```
# Visualize the stock prices
plt.figure(figsize=(12, 6))
plt.plot(tesla_df['Date'], tesla_df['Close'], label='Tesla')
plt.plot(hyundai_df['Date'], hyundai_df['Close'], label='Hyundai')
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.title('Tesla vs Hyundai Stock Prices')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```



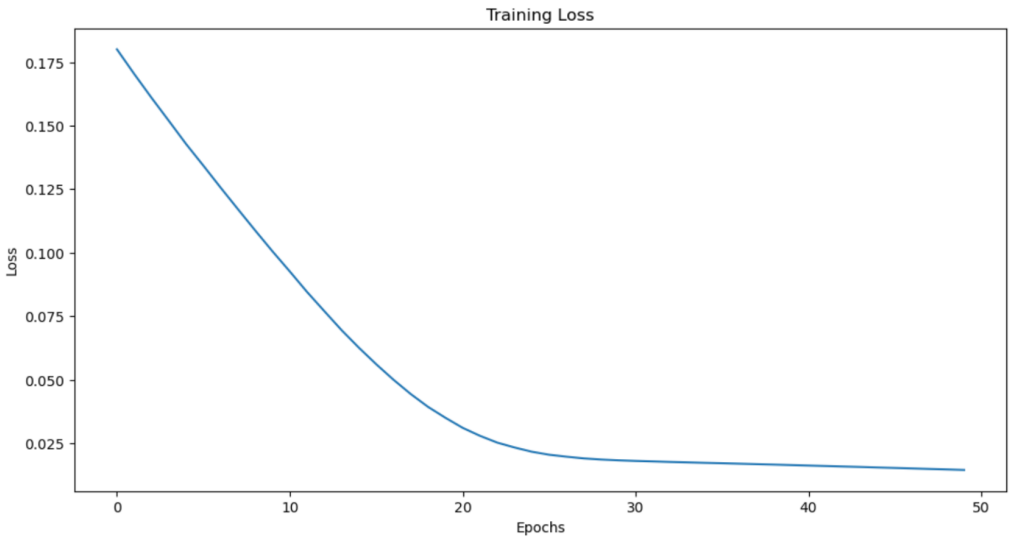
5. Build LST Model

In [15]:
```
# Build the LSTM model
model = Sequential()
model.add(LSTM(64, activation='relu', input_shape=(1, 2)))
model.add(Dense(2))
model.compile(optimizer='adam', loss='mse')
```

In [16]:
```
# Train the model
history = model.fit(X_train, y_train, epochs=50, verbose=0)
```

6. Visualize the training loss

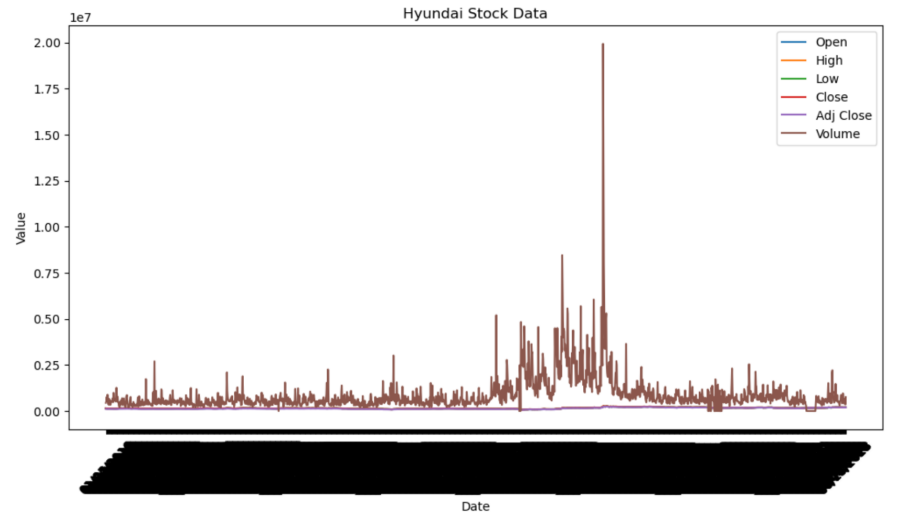
In [17]:
```
# Visualize the training loss
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'])
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training Loss')
plt.show()
```



In [18]:
```
# Make predictions with the model
```

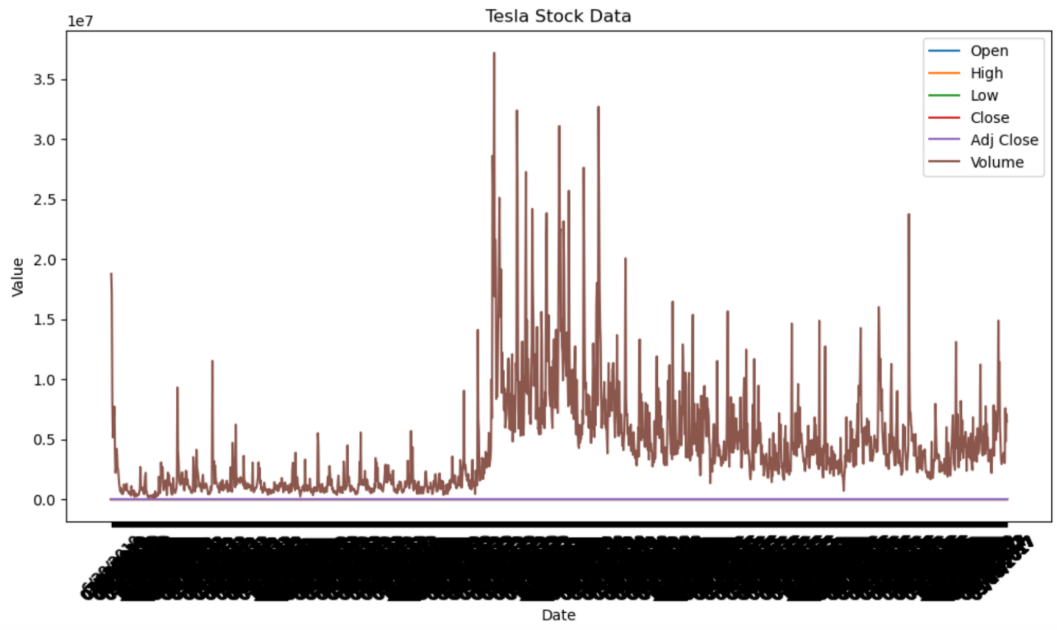
7. Visualize the data for Hyundai

In [13]:
```
# Visualize the data for Hyundai
plt.figure(figsize=(12, 6))
for column in columns_to_plot:
    plt.plot(hyundai_df['Date'], hyundai_data[column], label=column)
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Hyundai Stock Data')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```



8. Visualize the data for Tesla

```
In [12]: # Visualize the data for Tesla
plt.figure(figsize=(12, 6))
for column in columns_to_plot:
    plt.plot(tesla_df['Date'], tesla_data[column], label=column)
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Tesla Stock Data')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```



9. Perform prediction

To make predictions, you can use various techniques such as linear regression, ARIMA, or machine learning algorithms. Here's an example using linear regression

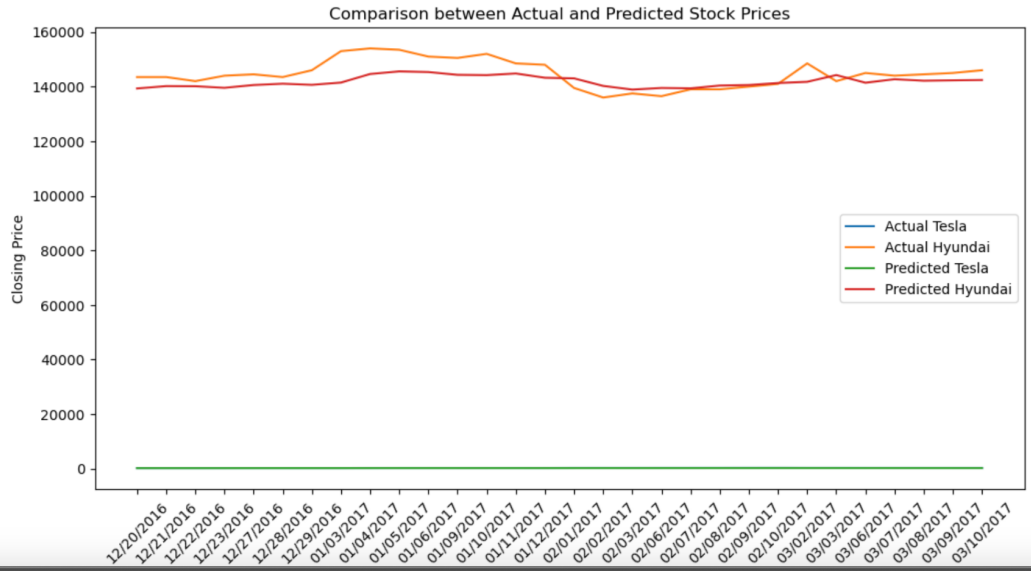
Epochs

```
In [18]: # Make predictions with the model
X_test = test_data[:-1]
X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])
predicted_data = model.predict(X_test)
predicted_data = scaler.inverse_transform(predicted_data)

1/1 [==============================] - 0s 154ms/step
```

10. Compare the datasets and predicted data

In [19]:
```
# Compare the datasets and predicted data
plt.figure(figsize=(12, 6))
plt.plot(merged_df['Date'][-30:], merged_df['Close_tesla'][-30:], label='Actual Tesla')
plt.plot(merged_df['Date'][-30:], merged_df['Close_hyundai'][-30:], label='Actual Hyundai')
plt.plot(merged_df['Date'][-30:], predicted_data[:, 0], label='Predicted Tesla')
plt.plot(merged_df['Date'][-30:], predicted_data[:, 1], label='Predicted Hyundai')
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.title('Comparison between Actual and Predicted Stock Prices')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```



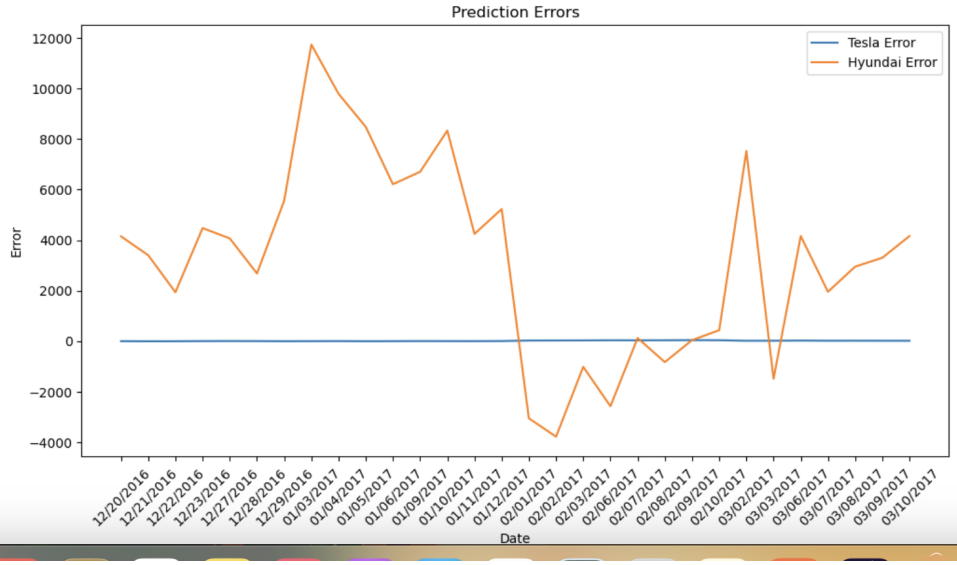
11. Visualize the prediction errors

In [16]:
```
# Calculate prediction errors
actual_tesla = merged_df['Close_tesla'][-30:].values
actual_hyundai = merged_df['Close_hyundai'][-30:].values
prediction_tesla = predicted_data[:, 0]
prediction_hyundai = predicted_data[:, 1]
```

In [17]:
```
# Calculate error metrics
mse_tesla = mean_squared_error(actual_tesla, prediction_tesla)
mse_hyundai = mean_squared_error(actual_hyundai, prediction_hyundai)
mae_tesla = mean_absolute_error(actual_tesla, prediction_tesla)
mae_hyundai = mean_absolute_error(actual_hyundai, prediction_hyundai)
```

In [18]:
```
# Visualize the prediction errors
plt.figure(figsize=(12, 6))
plt.plot(merged_df['Date'][-30:], actual_tesla - prediction_tesla, label='Tesla Error')
plt.plot(merged_df['Date'][-30:], actual_hyundai - prediction_hyundai, label='Hyundai Error')
plt.xlabel('Date')
plt.ylabel('Error')
plt.title('Prediction Errors')
plt.legend()
plt.xticks(rotation=45)
plt.show()

print("Mean Squared Error (Tesla):", mse_tesla)
print("Mean Absolute Error (Tesla):", mae_tesla)
print("Mean Squared Error (Hyundai):", mse_hyundai)
print("Mean Absolute Error (Hyundai):", mae_hyundai)
```



Conclusion

The above code demonstrates the process of analyzing and predicting stock prices for Tesla and Hyundai using a combination of data visualization and LSTM (Long Short-Term Memory) neural network.

The code first loads the Tesla and Hyundai stock price datasets and performs exploratory analysis by printing the first few rows of each dataset. It then visualizes the stock prices of Tesla and Hyundai over time to gain insights into their trends and patterns.

Next, the code merges the datasets based on the 'Date' column and prepares the data for training the LSTM model. The data is scaled using MinMaxScaler and split into training and testing sets.

The LSTM model is built and trained using the training data. The model is designed to predict the next day's closing prices for both Tesla and Hyundai based on the previous day's closing prices.

After training the model, it is used to make predictions on the testing data. The predicted prices are then inverse transformed to their original scale using the scaler.

The code compares the actual stock prices of Tesla and Hyundai with the predicted prices by plotting them on the same graph. This provides a visual representation of the model's performance in capturing the stock price trends.

Additionally, the code calculates and displays the mean squared error (MSE) and mean absolute error (MAE) to evaluate the prediction accuracy. The prediction errors are also visualized to provide insights into the magnitude and direction of the errors.

In conclusion, the code demonstrates the use of LSTM neural networks for stock price prediction and provides visualizations and error metrics to assess the model's performance. It showcases how data visualization and machine learning techniques can be employed to gain insights into stock market trends and make informed predictions.