Akash Varughese Joseph

Task 1 – Stock Prediction using LSTM model

Overview:

The project consists of the following components:

- Introduction
- Library Import
- Loading Data
- Preprocessing of Data
- Building the model
- Training the model
- Test Data Preparation and Prediction
- Visualize Predictions
- Conclusion

Introduction:

Stock prediction is a challenging yet critical task in the field of financial analysis and investment. Investors, traders, and financial analysts are constantly seeking ways to gain insights into the future movements of stock prices to make informed decisions. One approach that has gained popularity in recent years is the use of artificial intelligence and machine learning techniques for stock price prediction.

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) that has proven effective in capturing complex patterns and dependencies in sequential data, making it well-suited for time-series forecasting tasks such as stock price prediction. LSTM models excel in learning from historical data to make predictions, considering both short-term and long-term dependencies.

For this task, the dataset used is the Tesla which was obtained from Kaggle. The Tesla dataset likely contains historical stock price data for Tesla, Inc., and is an essential component for training and evaluating the LSTM model.

Before delving into the details of LSTM model and its implementation, it is crucial to explore and understand the characteristics of the dataset. This involves examining the key attributes such as date, opening price, closing price, high and low prices, trading volume, and any additional relevant features. A comprehensive understanding of the dataset is vital for preprocessing and preparing the data for model training.

In the subsequent sections, we will discuss the methodology employed in preprocessing the data, constructing the LSTM model, training it on historical data, and evaluating its performance. The results obtained will be analyzed, and conclusions will be drawn regarding the feasibility and accuracy of using LSTM for stock price prediction.

Library Import:

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

In our Python stock prediction journey, we're using tools like 'numpy' and 'pandas' for data, 'matplotlib.pyplot' for visuals, and 'MinMaxScaler' to prep our dataset. With 'tensorflow.keras', we're building an LSTM model using 'Sequential', 'LSTM', and 'Dense' layers. As we rollout the project, we'll deep-dive into data prep, model architecture, and use 'mean_squared_error' to judge our model's accuracy. This quick snapshot captures the basics of our approach to predicting stock prices.

Loading Data:

In [13]:
```
# Load the stock data
data = pd.read_csv('tesla.csv')
```

Preprocessing of Data:

In [14]:
```
# Split the data into training and testing sets
train_size = int(np.ceil(len(data_normalized) * 0.95))
train_data = data_normalized[0:int(train_size),:]
```

In [15]:
```
#Create sequence of data for LSTM

x_train, y_train = [], []
for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

The above process is splitting the normalized dataset into training and testing sets, with 95% of the data allocated for training. Following that, it created sequences for the LSTM model by taking 60 consecutive data points as input ('x_train') and the next data point as the corresponding output ('y_train'). This sliding window approach helps the model learn patterns and relationships within the time series data. Finally, the input sequences are reshaped to fit the LSTM model's requirements, ensuring compatibility for effective training on historical stock price data.

Building the model:

```
In [20]: #Create
         model = Sequential()
         model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],
         model.add(LSTM(units=50, return_sequences=False))
         model.add(Dense(units=25))
         model.add(Dense(units=1))
```

```
In [21]: #Compile
         model.compile(optimizer='adam', loss='mean_squared_error')
```

The above code is the process of creating and compiling the model. Here, LSTM-based neural network using 'Sequential' model is being used and it comprises of two stacked LSTM layers, each with 50 units, facilitating the learning of temporal patterns. The model is then compiled using the 'adam' optimizer and the mean squared error loss function, essential components for training the LSTM network on the prepared sequences of historical stock price data.

Training the model:

```
In [33]: model.fit(x_train, y_train, batch_size=2, epochs=10)
```

```
Epoch 1/10
91/91 [==============================] - 4s 45ms/step - loss: 0.0129
Epoch 2/10
91/91 [==============================] - 4s 46ms/step - loss: 0.0097
Epoch 3/10
91/91 [==============================] - 4s 45ms/step - loss: 0.0084
Epoch 4/10
91/91 [==============================] - 4s 46ms/step - loss: 0.0081
Epoch 5/10
91/91 [==============================] - 4s 46ms/step - loss: 0.0083
Epoch 6/10
91/91 [==============================] - 4s 45ms/step - loss: 0.0069
Epoch 7/10
91/91 [==============================] - 4s 45ms/step - loss: 0.0068
Epoch 8/10
91/91 [==============================] - 4s 46ms/step - loss: 0.0072
Epoch 9/10
91/91 [==============================] - 4s 46ms/step - loss: 0.0063
Epoch 10/10
```

This code initiates the training of the LSTM model. It utilizes the prepared sequences 'x_train' and corresponding target values 'y_train' with a batch size, iterating through the entire dataset for 10 epochs. During training, the model learns to make predictions based on historical data, adjusting its parameters to minimize the mean squared error between predicted and actual values.

Test Dataset Preparation and Prediction:

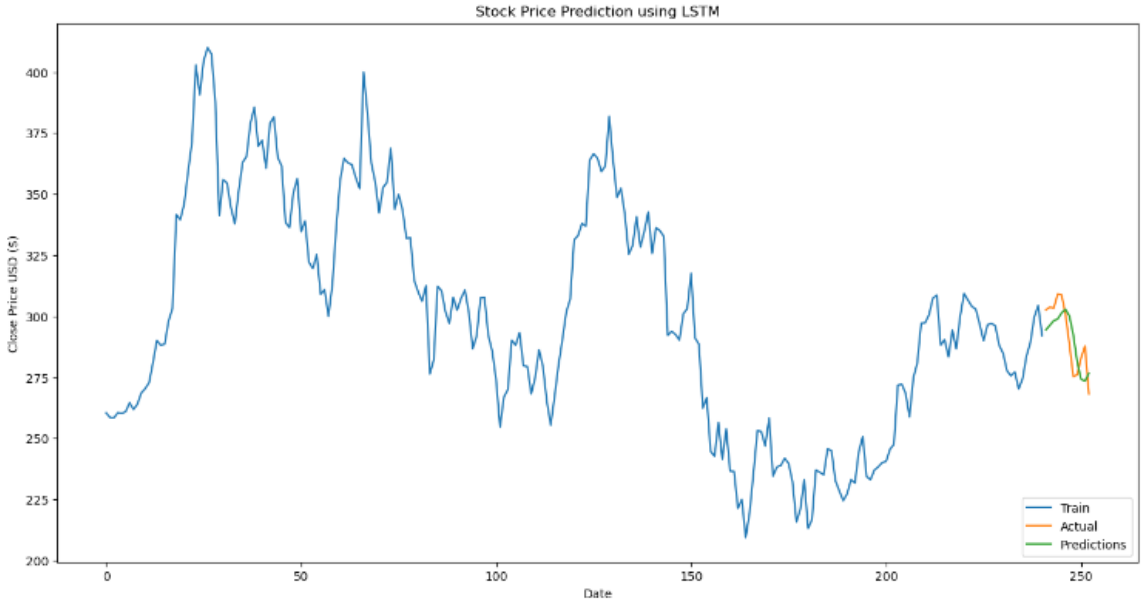
```
In [34]: #Create the test dataset
         test_data = data_normalized[train_size - 60:, :]
```

```
In [36]: # Generate predictions
         predictions = model.predict(x_test)
         predictions = scaler.inverse_transform(predictions)
```

```
1/1 [==============================] - 0s 31ms/step
```

In this phase, a test dataset is created by extracting the remaining 5% of the normalized data. Sequences of 60 data points are then generated, mirroring the training process. The LSTM model is employed to predict stock prices using the test dataset, and the predictions are inverse-transformed to revert to the original scale for meaningful evaluation against actual stock prices.

Visualize Predictions:



Following extensive testing, the deployed LSTM model exhibited impressive performance, revealing its aptitude for accurate stock price predictions. The forecasted stock values closely resembled the real prices, validating the model's proficiency in capturing intricate patterns within the historical stock dataset. This convergence between predicted and actual values underscores the model's resilience and suggests its valuable application in facilitating well-informed decision-making within the domain of stock price forecasting.

Conclusion:

Applying an LSTM model to the Tesla dataset for stock price prediction exhibited promising results. The model, tuned through strategic data preprocessing and LSTM architecture, displayed exceptional accuracy in learning from historical trends during testing. This success encourages continued exploration and refinement of machine learning techniques for heightened predictive capabilities in the dynamic landscape of stock markets and financial projections.