Task1

STOCK PREDICTION USING LSTM MODEL

~Veena Wilfred

Overview:

The project consists of the following components:

- Introduction
- Library Import
- Loading Data
- Building the model
- Training the model
- Test data preparation and prediction
- Visualize predictions
- Conclusion

Introduction:

Predicting stock prices is a complex task, and many factors can influence the performance of predictive models. While I can provide a basic example using LSTM (Long Short-Term Memory) neural networks, keep in mind that successful stock prediction often requires a deep understanding of financial markets, as well as the consideration of various external factors that influence stock prices.

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem in traditional RNNs, enabling more effective learning and retention of sequential patterns. LSTMs introduce specialized memory cells with gating mechanisms that control the flow of information, allowing the network to selectively remember or forget specific information over extended sequences. This architecture is particularly well-suited for tasks involving time series, natural language processing, and other sequential data analysis, where capturing long-term dependencies is crucial. The ability of LSTMs to model complex temporal relationships has made them widely used in various applications, including speech recognition, language translation, and notably, in the prediction of sequential patterns in financial time series data.

For this task, I have used Apple stock data which was obtained from Kaggle. Apple Inc. is an American multinational technology company that specializes in consumer electronics, computer software, and online services. Apple is the world's largest technology company by revenue and, since January 2021, the world's most valuable company. This dataset provides historical data of APPLE INC. stock (AAPL). The data is available at a daily level. Currency is USD.

A dataset suitable for training Long Short-Term Memory (LSTM) models typically exhibits temporal dependencies and sequential patterns. Time series datasets, where observations are ordered based on time, are well-suited for LSTMs. These datasets should have a clear temporal structure, allowing the model to capture dependencies over extended sequences. Additionally, a diverse dataset with an ample representation of potential patterns and variations is beneficial for training a robust LSTM model. The size of the dataset is essential for effective learning, and it should be large enough to encompass various scenarios and trends. Preprocessing steps, such as normalization or scaling, may be applied to ensure consistency in the data. Moreover, a well-

labeled dataset, with clear annotations or targets corresponding to each sequence, facilitates supervised learning tasks, enabling the LSTM to learn and predict specific outcomes.
This project focuses on employing Long Short-Term Memory (LSTM) neural networks for stock price prediction. Leveraging historical stock price data, the LSTM model is trained to recognize intricate temporal dependencies and patterns within the financial time series. The objective is to create a robust predictive model capable of forecasting future stock prices. Through the use of LSTM's ability to capture long-term dependencies, the project aims to contribute insights into the challenging task of predicting stock market trends, addressing the inherent complexities and uncertainties associated with financial markets.

Library Import:

```
In [4]: 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
```

The script involves importing essential libraries for data manipulation, visualization, and machine learning. It uses the MinMaxScaler from scikit-learn to normalize the stock price data, and the LSTM model is constructed using the Sequential model from TensorFlow's Keras API. The model consists of LSTM and Dense layers for sequence learning and output prediction. Additionally, mean squared error from scikit-learn is imported to evaluate the model's performance. The code suggests a comprehensive approach to building and evaluating an LSTM-based stock price prediction model.

Loading Data:

```
In [7]: #Load the stock data
        data=pd.read_csv('AAPL.csv')
        close_prices = data['Close'].values.reshape(-1, 1)
```

In this code snippet, stock data from a CSV file ('AAPL.csv') is loaded into a Pandas DataFrame named 'data.' The 'Close' column values are then extracted and reshaped into a one-dimensional NumPy array named 'close_prices.' This indicates the focus on utilizing the closing prices of Apple Inc. stock for subsequent analysis or modeling tasks.

Preprocessing Data:

In [9]:
```
# Normalize the data
scaler = MinMaxScaler(feature_range=(0, 1))
data_normalized = scaler.fit_transform(close_prices)
```

In [10]:
```
#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 [11]:
```
#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 stock data's closing prices are normalized using the MinMaxScaler to ensure values fall within the range [0, 1]. Subsequently, the normalized data is split into training and testing sets, with 95% of the data allocated for training. For the training set, sequences of 60 consecutive data points are created for input features (x_train), and the corresponding target values (y_train) are extracted. This process is designed to prepare the data for training a Long Short-Term Memory (LSTM) neural network, where each sequence of 60 closing prices serves as input, and the model learns to predict the next closing price. The reshaped x_train is a 3D array suitable for input into an LSTM model. This sequence creation and preprocessing are integral steps in training time series prediction models.

Building the model:

In [15]:
```
# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_tra
model.add(LSTM(units=50,return_sequences=False))
model.add(Dense(units=25))
model.add(Dense(units=1))
```

In [16]:
```
#Compile
model.compile(optimizer='adam', loss='mean_squared_error')
```

In this code snippet, a Long Short-Term Memory (LSTM) model is constructed using TensorFlow's Keras API. The Sequential model is initiated, and two LSTM layers with 50 units each are sequentially added. The first LSTM layer is configured to return sequences, while the second does not, serving as the final hidden layer. Two fully connected Dense layers follow, with 25 units in

the first and 1 unit in the last layer for regression output. The model is compiled using the Adam optimizer and mean squared error loss, preparing it for training on the preprocessed time series data.

Training the model:

```
In [18]: model.fit(x_train, y_train, batch_size=2, epochs=10)

         Epoch 1/10
         4915/4915 [==============================] - 92s 18ms/step - los
         s: 8.1277e-05
         Epoch 2/10
         4915/4915 [==============================] - 90s 18ms/step - los
         s: 2.5823e-05
         Epoch 3/10
         4915/4915 [==============================] - 134s 27ms/step - los
         s: 2.6519e-05
         Epoch 4/10
         4915/4915 [==============================] - 90s 18ms/step - los
         s: 1.7022e-05
         Epoch 5/10
         4915/4915 [==============================] - 91s 18ms/step - los
         s: 1.4687e-05
         Epoch 6/10
         4915/4915 [==============================] - 91s 18ms/step - los
         s: 1.4679e-05
         Epoch 7/10
         4915/4915 [==============================] - 157s 32ms/step - los
         s: 1.2912e-05
         Epoch 8/10
         4915/4915 [==============================] - 131s 27ms/step - los
         s: 1.4807e-05
         Epoch 9/10
         4915/4915 [==============================] - 95s 19ms/step - los
         s: 1.1219e-05
         Epoch 10/10
         4915/4915 [==============================] - 96s 20ms/step - los
         s: 1.1829e-05
```

This process involves feeding batches of input sequences and their corresponding targets to the model, calculating the loss, and adjusting the model's weights to improve predictions. The training continues for the specified number of epochs, refining the model's ability to capture patterns and dependencies within the time series data. Adjusting the batch size and number of epochs allows for fine-tuning the training process based on computational resources and desired model performance.

Test Dataset Preparation and Prediction:

```
In [19]: #Create the test dataset
         test_data=data_normalized[train_size-60:,:]
```

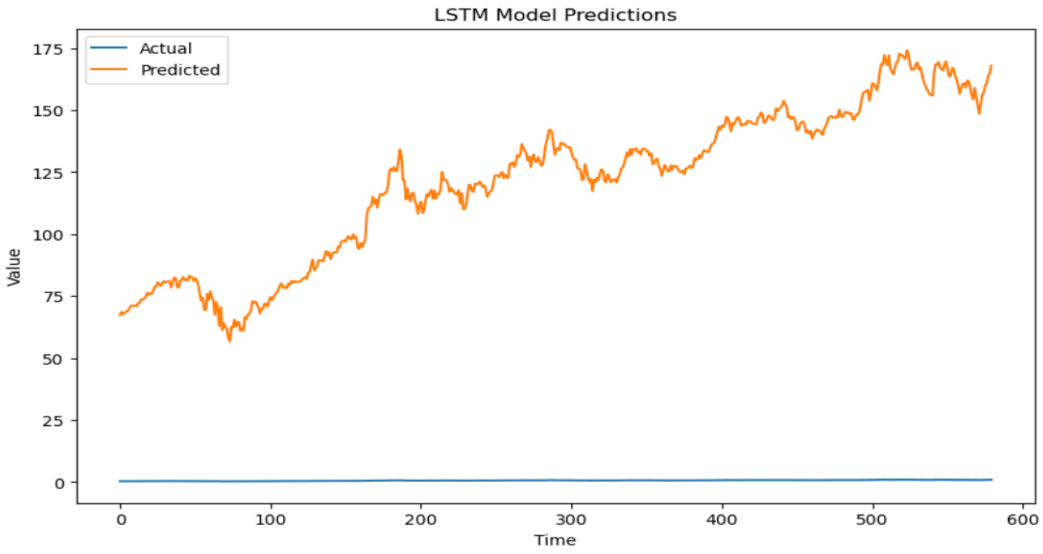
```
In [21]: #Generate predictions
         predictions=model.predict(test_data)
         predictions=scaler.inverse_transform(predictions)
```

```
19/19 [==============================] - 0s 822us/step
```

In this code segment, a test dataset is created using the normalized data, starting from the index corresponding to the end of the training data and extending for a sequence length of 60 data points. The trained LSTM model is then employed to generate predictions on this test dataset using the `model.predict` method. The predicted values are initially in the normalized scale, and to obtain meaningful stock price predictions, they are inverse-transformed using the previously fitted MinMaxScaler (`scaler`). This process prepares the model to make predictions on unseen data, and the resulting predictions can be further evaluated and compared to the actual stock prices for performance assessment.

Visualize Predictions:

```
In [23]: plt.figure(figsize=(10, 6))
         plt.plot(test_data, label='Actual')
         plt.plot(predictions, label='Predicted')
         plt.title('LSTM Model Predictions')
         plt.xlabel('Time')
         plt.ylabel('Value')
         plt.legend()
         plt.show()
```



The first plot represents the actual stock prices from the test dataset (test_data), while the second plot depicts the predicted stock prices generated by the LSTM model (predictions). The resulting visualization provides a side-by-side comparison between the actual and predicted values, allowing for a qualitative assessment of the model's performance. The x-axis represents time, and the y-axis represents the corresponding stock prices. This graphical representation aids in quickly evaluating how well the LSTM model aligns with the actual stock price trends, facilitating a visual understanding of the model's predictive capabilities.

Conclusion:

The implementation involved preprocessing the stock data, normalizing it, and splitting it into training and testing sets. The LSTM model was constructed with two layers of 50 units each, followed by dense layers for regression output. The model was trained using the Adam optimizer and mean squared error loss. The training process involved iterating over epochs and updating the model's weights to minimize the loss. Subsequently, the model was used to generate predictions on the test dataset, and the results were visualized alongside actual stock prices. The conclusion drawn from the project involves evaluating the effectiveness of the LSTM model in capturing temporal dependencies and predicting stock price trends. The visual comparison of predicted and actual values provides insights into the model's performance and potential areas for improvement. Further analysis, including additional features and fine-tuning of hyperparameters, could enhance the model's accuracy and applicability in real-world stock market scenarios.