Data Engineer INTERN at HACKVEDA LIMITED

AUTHOR: BANDANA PRAKASH
TASK 4: Stock_Market_Prediction_Model_Creation
PURPOSE : to develop a model for analyzing and predicting stock market trends.

It involves:

Data Collection: Fetching historical stock price data (e.g., Google stock) using yfinance.
Data Preparation: Cleaning and structuring the data for analysis.
Trend Analysis: Identifying patterns in stock prices over time.
Prediction Modeling: Building a predictive model to forecast future stock prices based on historical data.
This project aims to assist in making informed investment decisions by leveraging data-driven insights and predictive analytics.

Steps Involved

Import Libraries: Load necessary libraries such as numpy, pandas, matplotlib, and yfinance.
Define Timeframe: Set the start and end dates for the historical data to be analyzed (from January 1, 2012, to December 21, 2022).
Fetch Data: Use yfinance to download historical stock data for Google (GOOG).
Reset Index: Prepare the data by resetting the index for easier manipulation.
Data Exploration: Inspect the dataset to understand its structure and contents.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
```

```
start = '2012-01-01'
end = '2022-12-21'
stock = 'GOOG'

data = yf.download(stock, start, end)
```

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

```
data.reset_index(inplace=True)
```

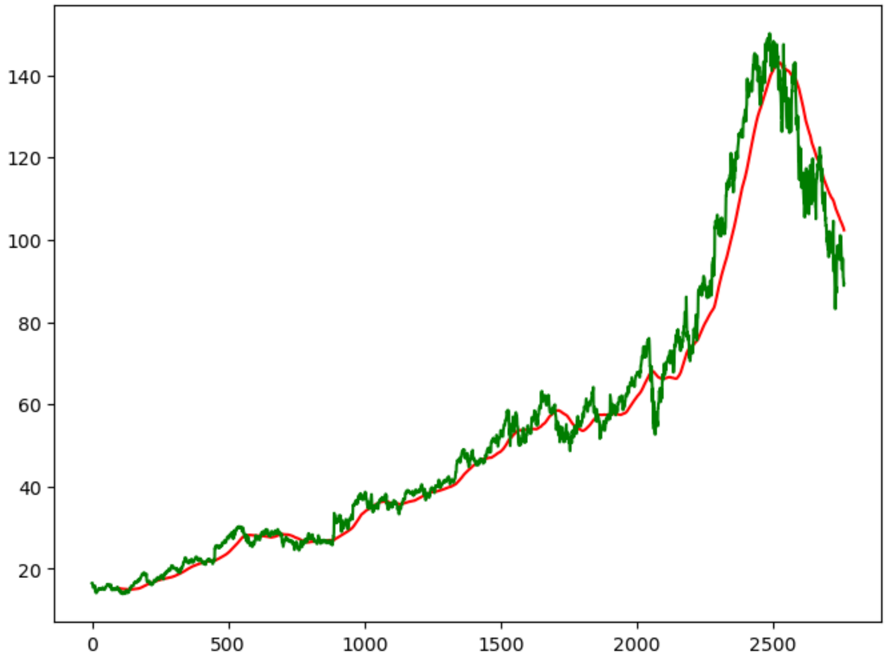
```
data
```

Price	Date	Close	High	Low	Open	Volume
Ticker		GOOG	GOOG	GOOG	GOOG	GOOG
0	2012-01-03	16.513794	16.581795	16.190173	16.204321	147611217
1	2012-01-04	16.585020	16.633911	16.394919	16.504364	114989399
2	2012-01-05	16.354961	16.478056	16.285969	16.432392	131808205
3	2012-01-06	16.131853	16.379531	16.126144	16.358435	108119746
4	2012-01-09	15.447884	16.056905	15.417357	16.044495	233776981
...	...	...	...	...	...	...
2756	2022-12-14	94.968765	96.871931	93.603675	95.197945	26452900
2757	2022-12-15	90.873482	93.693352	90.106242	93.205108	28298800
2758	2022-12-16	90.534691	91.421504	89.687736	90.873470	48485500
2759	2022-12-19	88.830826	90.873482	88.606633	90.554628	23020500
2760	2022-12-20	89.309097	89.458562	87.724794	88.412326	21976800

2761 rows × 6 columns

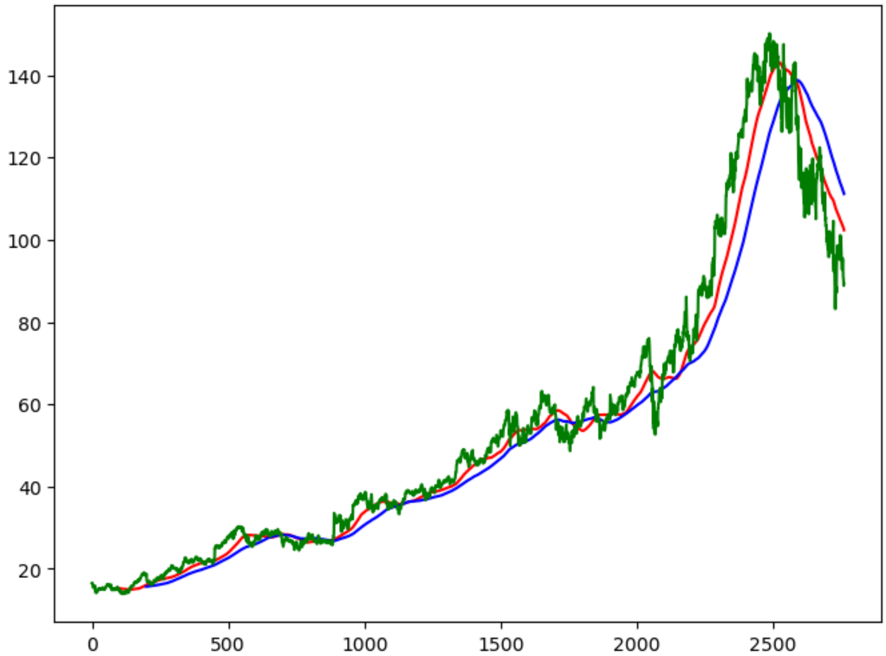
```
ma_100_days = data.Close.rolling(100).mean()
```

```
plt.figure(figsize=(8,6))
plt.plot(ma_100_days, 'r')
plt.plot(data.Close, 'g')
plt.show()
```



```
ma_200_days = data.Close.rolling(200).mean()
```

```
plt.figure(figsize=(8,6))
plt.plot(ma_100_days, 'r')
plt.plot(ma_200_days,'b')
plt.plot(data.Close,'g')
plt.show()
```



```
data.dropna(inplace=True)
```

```
data_train = pd.DataFrame(data.Close[0: int(len(data)*0.80)])
data_test = pd.DataFrame(data.Close[int(len(data)*0.80): len(data)])
```

```
data_train.shape[0]
```

```
2208
```

```
data_test.shape[0]
```

```
553
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))


data_train_scale = scaler.fit_transform(data_train)


x = []
y = []

for i in range(100, data_train_scale.shape[0]):
    x.append(data_train_scale[i-100:i])
    y.append(data_train_scale[i,0])



x, y = np.array(x), np.array(y)


from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential


model = Sequential()
model.add(LSTM(units = 50, activation = 'relu', return_sequences = True,
               input_shape = ((x.shape[1],1))))
model.add(Dropout(0.2))

model.add(LSTM(units = 60, activation='relu', return_sequences = True))
model.add(Dropout(0.3))

model.add(LSTM(units = 80, activation = 'relu', return_sequences = True))
model.add(Dropout(0.4))

model.add(LSTM(units = 120, activation = 'relu'))
model.add(Dropout(0.5))

model.add(Dense(units =1))


model.compile(optimizer = 'adam', loss = 'mean_squared_error')


model.fit(x,y, epochs = 50, batch_size =32, verbose =1)
```

```
Epoch 1/50
2025-01-23 18:37:52.856514: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz
66/66 [==============================] - 9s 113ms/step - loss: 0.0372
Epoch 2/50
66/66 [==============================] - 8s 116ms/step - loss: 0.0075
Epoch 3/50
66/66 [==============================] - 8s 118ms/step - loss: 0.0066
Epoch 4/50
66/66 [==============================] - 8s 117ms/step - loss: 0.0053
Epoch 5/50
66/66 [==============================] - 8s 118ms/step - loss: 0.0056
Epoch 6/50
66/66 [==============================] - 8s 118ms/step - loss: 0.0046
Epoch 7/50
66/66 [==============================] - 9s 141ms/step - loss: 0.0044
Epoch 8/50
66/66 [==============================] - 10s 147ms/step - loss: 0.0043
Epoch 9/50
66/66 [==============================] - 10s 153ms/step - loss: 0.0041
Epoch 10/50
66/66 [==============================] - 10s 147ms/step - loss: 0.0040
Epoch 11/50
66/66 [==============================] - 10s 148ms/step - loss: 0.0041
Epoch 12/50
66/66 [==============================] - 10s 149ms/step - loss: 0.0039
Epoch 13/50
66/66 [==============================] - 10s 146ms/step - loss: 0.0037
Epoch 14/50
66/66 [==============================] - 10s 148ms/step - loss: 0.0037
Epoch 15/50
66/66 [==============================] - 9s 139ms/step - loss: 0.0040
Epoch 16/50
66/66 [==============================] - 9s 139ms/step - loss: 0.0032
Epoch 17/50
66/66 [==============================] - 9s 139ms/step - loss: 0.0033
Epoch 18/50
66/66 [==============================] - 9s 139ms/step - loss: 0.0034
Epoch 19/50
66/66 [==============================] - 9s 140ms/step - loss: 0.0030
Epoch 20/50
66/66 [==============================] - 9s 139ms/step - loss: 0.0029
Epoch 21/50
66/66 [==============================] - 9s 143ms/step - loss: 0.0028
```

```
Epoch 22/50
66/66 [==============================] - 9s 136ms/step - loss: 0.0031
Epoch 23/50
66/66 [==============================] - 9s 138ms/step - loss: 0.0032
Epoch 24/50
66/66 [==============================] - 9s 142ms/step - loss: 0.0028
Epoch 25/50
66/66 [==============================] - 9s 144ms/step - loss: 0.0025
Epoch 26/50
66/66 [==============================] - 9s 141ms/step - loss: 0.0027
Epoch 27/50
66/66 [==============================] - 9s 143ms/step - loss: 0.0028
Epoch 28/50
66/66 [==============================] - 9s 133ms/step - loss: 0.0026
Epoch 29/50
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 dropout (Dropout)           (None, 100, 50)           0

 lstm_1 (LSTM)               (None, 100, 60)           26640

 dropout_1 (Dropout)         (None, 100, 60)           0

 lstm_2 (LSTM)               (None, 100, 80)           45120

 dropout_2 (Dropout)         (None, 100, 80)           0

 lstm_3 (LSTM)               (None, 120)               96480

 dropout_3 (Dropout)         (None, 120)               0

 dense (Dense)               (None, 1)                 121

=================================================================
Total params: 178,761
Trainable params: 178,761
Non-trainable params: 0
_________________________________________________________________
```

```
pas_100_days = data_train.tail(100)
```

```
data_test = pd.concat([pas_100_days, data_test], ignore_index=True)
```

```
data_test_scale  =  scaler.fit_transform(data_test)
```

```
x = []
y = []

for i in range(100, data_test_scale.shape[0]):
    x.append(data_test_scale[i-100:i])
    y.append(data_test_scale[i,0])
x, y = np.array(x), np.array(y)
```

```
y_predict = model.predict(x)
```

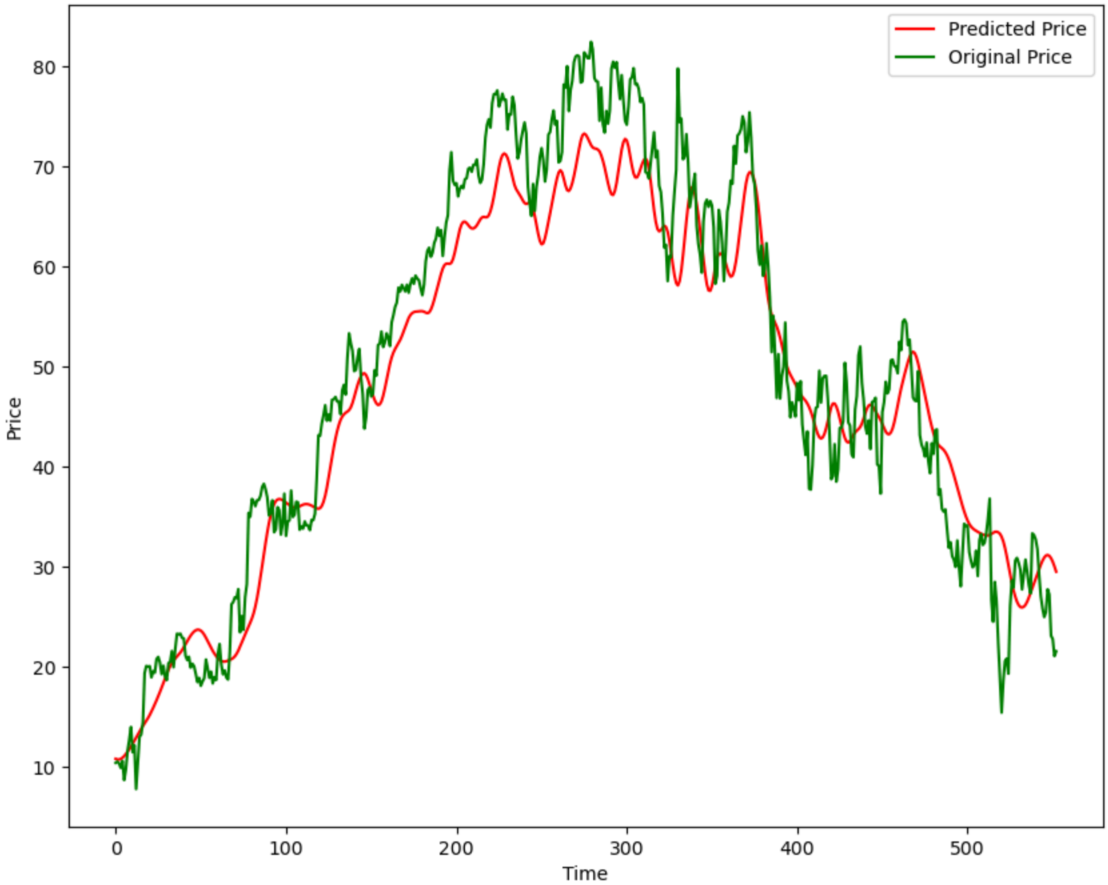
```
18/18 [==============================] - 1s 29ms/step
```

```
scale =1/scaler.scale_
```

```
y_predict = y_predict*scale
```

```
y = y*scale
```

```
plt.figure(figsize=(10,8))
plt.plot(y_predict, 'r', label = 'Predicted Price')
plt.plot(y, 'g', label = 'Original Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```




```
model.save('Stock Predictions Model.keras')
```