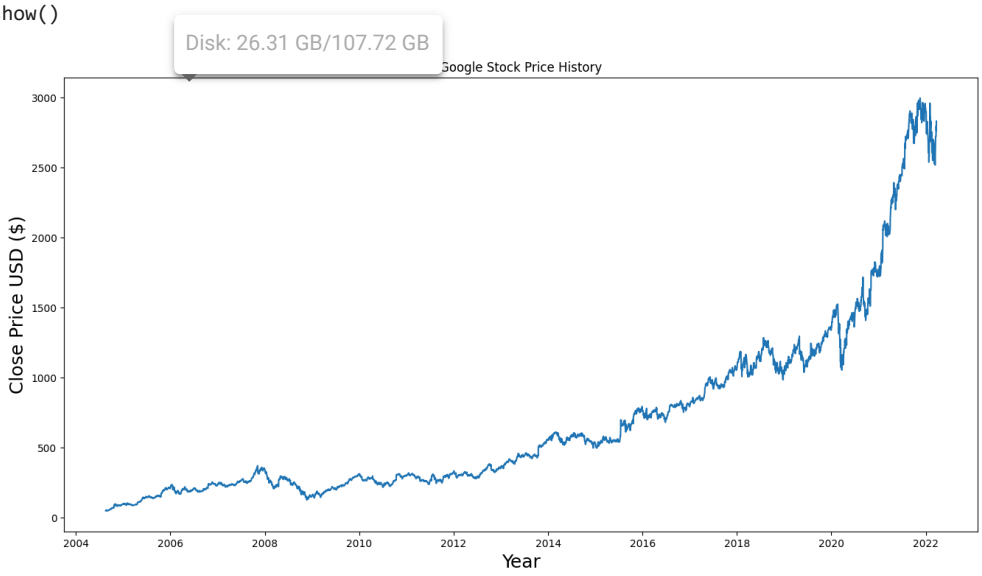
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
from google.colab import drive

file_path = '/content/GOOGL.csv'


# Load the dataset
df = pd.read_csv(file_path, sep = ",")


# Set the date as the index
df = df.set_index(pd.DatetimeIndex(df['Date'].values))


# Visualize the dataset
plt.figure(figsize=(16,8))
plt.title('Google Stock Price History')
plt.plot(df['Close'])
plt.xlabel('Year', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```




```
# Create a new dataframe with only the 'Close' column
data = df.filter(['Close'])


# Convert the dataframe to a numpy array
dataset = data.values


# Get the number of rows to train the model on
training_data_len = int(np.ceil(0.8 * len(dataset)))


# Scale the data
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)
```

```
# Create the training data
train_data = scaled_data[0:training_data_len, :]


# Define time_steps
time_steps = 30

# Split the data into x_train and y_train datasets
x_train = []
y_train = []

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


# Convert x_train and y_train to numpy arrays
x_train, y_train = np.array(x_train), np.array(y_train)


# Reshape the data for LSTM input
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))


# Build the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

Disk: 26.31 GB/107.72 GB

```
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')


# Train the model
model.fit(x_train, y_train, batch_size=1, epochs=5)
```

```
Epoch 1/5
3515/3515 [==============================] - 51s 14ms/step - loss: 2.6776e-04
Epoch 2/5
3515/3515 [==============================] - 45s 13ms/step - loss: 8.5209e-05
Epoch 3/5
3515/3515 [==============================] - 45s 13ms/step - loss: 5.6015e-05
Epoch 4/5
3515/3515 [==============================] - 46s 13ms/step - loss: 4.8317e-05
Epoch 5/5
3515/3515 [==============================] - 44s 12ms/step - loss: 3.4314e-05
<keras.src.callbacks.History at 0x7e3705bda5f0>
```

```
# Create the testing data
test_data = scaled_data[training_data_len - time_steps:, :]


# Split the data into x_test and y_test datasets
x_test = []
y_test = dataset[training_data_len:, :]

for i in range(time_steps, len(test_data)):
    x_test.append(test_data[i-time_steps:i, 0])


# Convert x_test to a numpy array
x_test = np.array(x_test)


# Reshape the data for LSTM input
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))


# Get the predicted stock prices
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

```
28/28 [==============================] - 1s 7ms/step
```

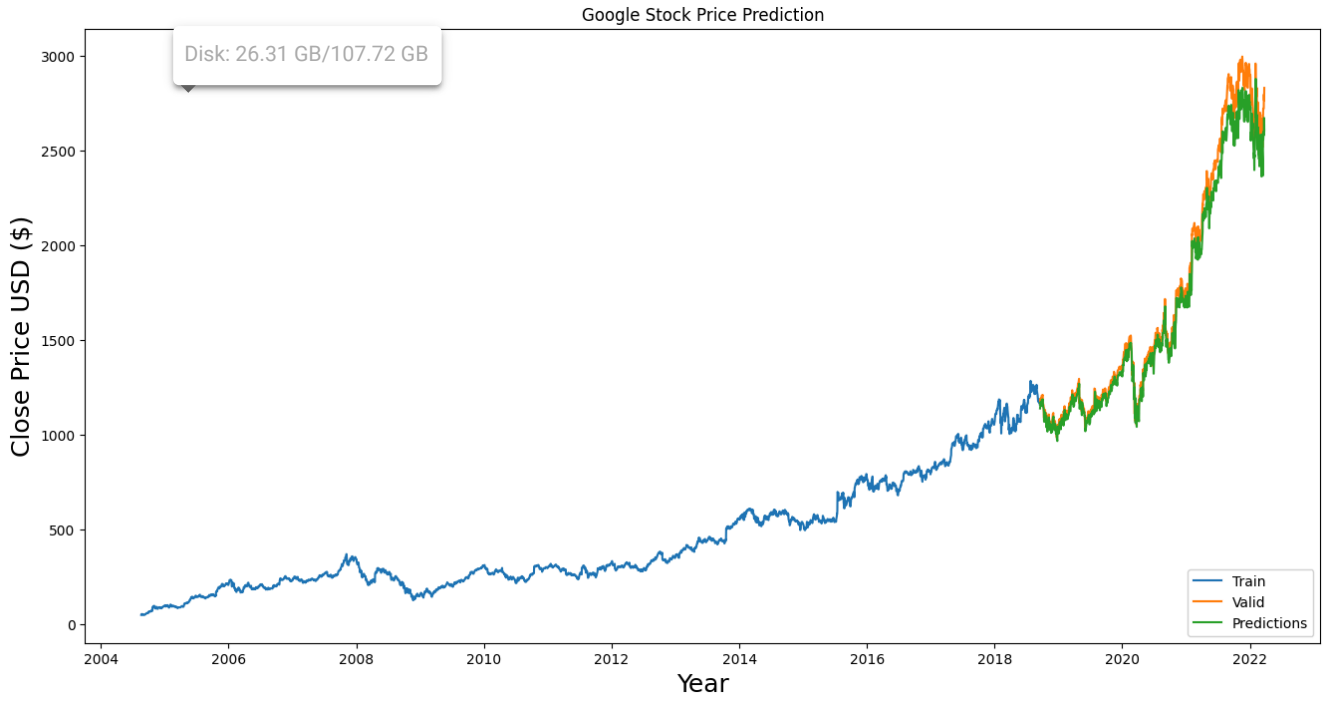
```
# Calculate the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
```

```
# Plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions

plt.figure(figsize=(16,8))
plt.title('Google Stock Price Prediction')
plt.xlabel('Year', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Valid', 'Predictions'], loc='lower right')
plt.show()
```

```
<ipython-input-22-91c8155f69f8>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
  valid['Predictions'] = predictions
```




```
print(rmse)
```

```
86.86651494443578
```

Disk: 26.31 GB/107.72 GB