```
# 📤 STEP 1: Upload Dataset (Kaggle file)
from google.colab import files
uploaded = files.upload()
```

Choose Files symbols_valid_meta.csv

- **symbols_valid_meta.csv**(text/csv) - 611626 bytes, last modified: 5/30/2025 - 100% done

```
Saving symbols_valid_meta.csv to symbols_valid_meta.csv
```

```
# 🔐 Upload kaggle.json API key
from google.colab import files
files.upload()
```

Choose Files kaggle.json

- **kaggle.json**(application/json) - 70 bytes, last modified: 5/30/2025 - 100% done

```
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"apurbobiswas23","key":"54541e14c169788db2b5f0e74706a09a"}'}
```

```
# 📁 Setup Kaggle credentials
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
# ⬇️ Download dataset
!kaggle datasets download -d jacksoncrow/stock-market-dataset

# 📦 Unzip it
!unzip stock-market-dataset.zip
```

```
  inflating: stocks/ZIONO.csv
```

```
  inflating: stocks/ZIONP.csv
  inflating: stocks/ZIONW.csv
  inflating: stocks/ZIOP.csv
  inflating: stocks/ZIV.csv
  inflating: stocks/ZIXI.csv
  inflating: stocks/ZKIN.csv
  inflating: stocks/ZLAB.csv
  inflating: stocks/ZM.csv
  inflating: stocks/ZN.csv
  inflating: stocks/ZNGA.csv
  inflating: stocks/ZNH.csv
  inflating: stocks/ZOM.csv
  inflating: stocks/ZS.csv
  inflating: stocks/ZSAN.csv
  inflating: stocks/ZTO.csv
  inflating: stocks/ZTR.csv
  inflating: stocks/ZTS.csv
  inflating: stocks/ZUMZ.csv
  inflating: stocks/ZUO.csv
  inflating: stocks/ZVO.csv
  inflating: stocks/ZYME.csv
  inflating: stocks/ZYNE.csv
  inflating: stocks/ZYXI.csv
replace symbols_valid_meta.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: symbols_valid_meta.csv
```

```
import os
files = os.listdir()
print(files)
```

```
['.config', 'kaggle.json', 'symbols_valid_meta.csv', 'etfs', 'stocks', 'drive', 'stock-market-dataset.zip', 'sampl
```

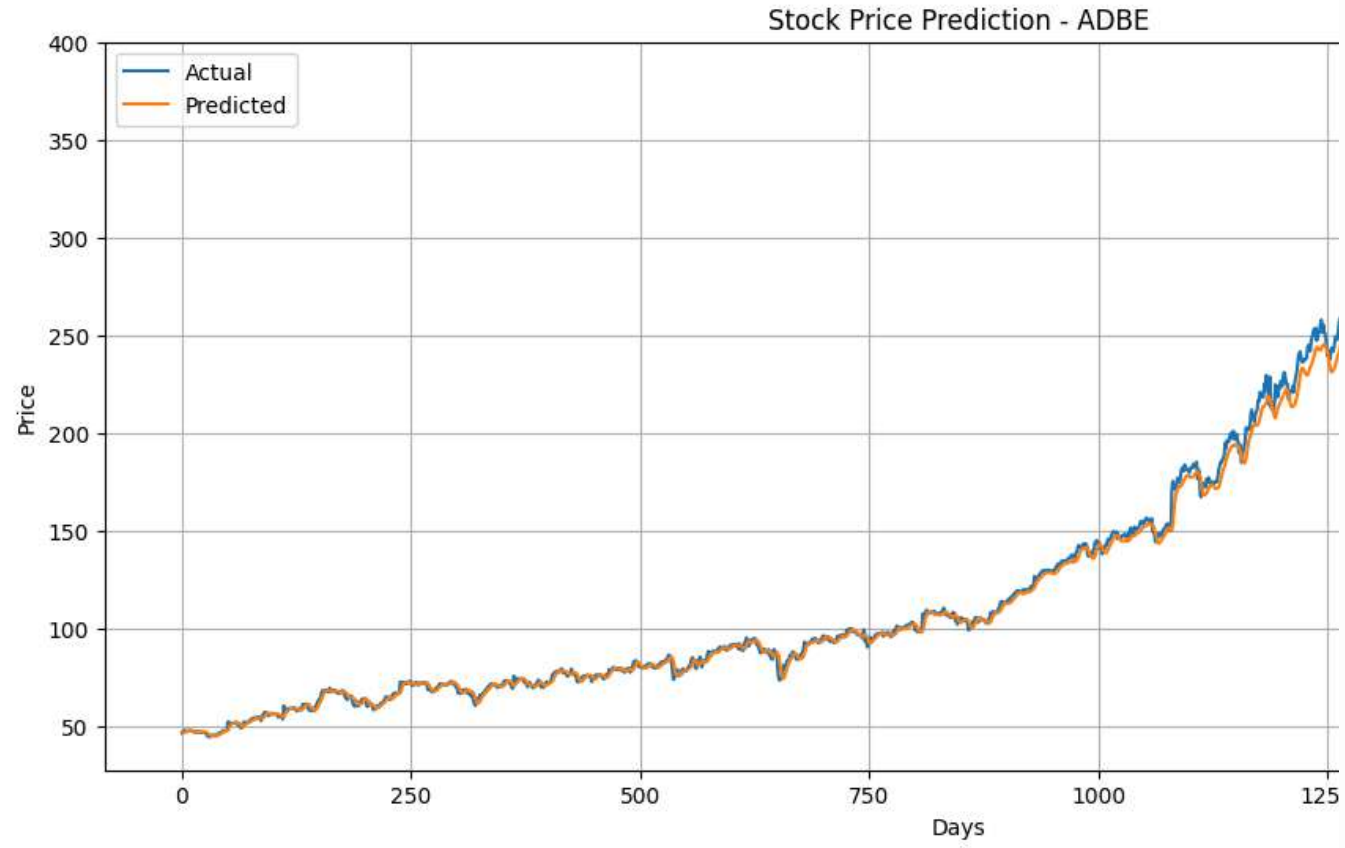
```
import os
os.listdir("stocks")
```

```
        'SAVE.csv',
        'BAND.csv',
        'GFF.csv',
        'TARA.csv',
        'UNM.csv',
        'PFLT.csv',
        'AXL.csv',
        'UCI.csv',
        'DMLP.csv',
        'UHS.csv',
        'OPRX.csv',
        'GLYC.csv',
        'ELF.csv',
        'AXSM.csv',
        'QTT.csv',
        'PGC.csv',
        'VMD.csv',
        'PNC.csv',
        'MSVB.csv',
        'GWPH.csv',
        'MAXR.csv',
        'EVM.csv',
        'ADM.csv',
        'WRN.csv',
        'VCYT.csv',
        'SDPI.csv',
        'EL.csv',
        'GRC.csv',
        'EQX.csv',
        'FMBH.csv',
```

```
# 🧠 LSTM Stock Price Prediction on ADBE.csv

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense, LSTM

# Load the file
df = pd.read_csv("stocks/ADBE.csv")  # adjust if path is different
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

# Use 'Close' price
data = df[['Close']].values

# Normalize
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)

# Split into train/test
train_len = int(len(scaled_data) * 0.8)
train_data = scaled_data[:train_len]
test_data = scaled_data[train_len - 60:]

# Sequence function
def create_dataset(dataset, step=60):
    X, y = [], []
    for i in range(step, len(dataset)):
        X.append(dataset[i-step:i, 0])
        y.append(dataset[i, 0])
    return np.array(X), np.array(y)

X_train, y_train = create_dataset(train_data)
X_test, y_test = create_dataset(test_data)
```

```
# Reshape for LSTM input
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.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))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32)

# Predict
predicted = model.predict(X_test)
predicted = scaler.inverse_transform(predicted)
actual = scaler.inverse_transform(y_test.reshape(-1, 1))

# RMSE Evaluation
rmse = np.sqrt(mean_squared_error(actual, predicted))
print(f"RMSE: {rmse}")

# 📈 Plot results
plt.figure(figsize=(14,6))
plt.plot(actual, label='Actual')
plt.plot(predicted, label='Predicted')
plt.title("Stock Price Prediction - ADBE")
plt.xlabel("Days")
plt.ylabel("Price")
plt.legend()
plt.grid(True)
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_sha
  super().__init__(**kwargs)
Epoch 1/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 17s 58ms/step - loss: 2.0056e-04
Epoch 2/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 22s 64ms/step - loss: 1.0952e-05
Epoch 3/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 20s 64ms/step - loss: 1.0547e-05
Epoch 4/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 17s 80ms/step - loss: 9.4785e-06
Epoch 5/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 17s 63ms/step - loss: 9.1696e-06
Epoch 6/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 20s 60ms/step - loss: 7.8889e-06
Epoch 7/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 13s 64ms/step - loss: 6.5854e-06
Epoch 8/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 20s 60ms/step - loss: 6.0763e-06
Epoch 9/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 20s 56ms/step - loss: 5.5013e-06
Epoch 10/10
211/211 ━━━━━━━━━━━━━━━━━━━━ 22s 64ms/step - loss: 5.3621e-06
53/53 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step
RMSE: 8.169597286445006
```




```
plt.figure(figsize=(14,6))
plt.plot(actual, label='Actual')
plt.plot(predicted, label='Predicted')
plt.title("Stock Price Prediction - ADBE")
plt.xlabel("Days")
plt.ylabel("Price")
plt.legend()
plt.grid(True)
plt.savefig("adbe_prediction_plot.png")
```


Stock Price Prediction - ADBE
Actual
Predicted
Price
400
350
300
250
200
150