Double-click (or enter) to edit

1.upload the dataset

```
# Step 1: Import required libraries
from google.colab import files
import pandas as pd

# Step 2: Upload the file
uploaded = files.upload()

# Step 3: Load the file into a pandas DataFrame (assuming it's a CS
import io
df = pd.read_csv(io.BytesIO(uploaded[next(iter(uploaded))]))

# Step 4: Display the first few rows
df.head()
```

Choose files student-dataset.csv

- **student-dataset.csv**(text/csv) - 56993 bytes, last modified: 08/05/2025 - 100% done

```
Saving student-dataset.csv to student-dataset.csv
```

	school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob
0	
1	
2	
3	
4	

Next steps: Generate code with df | View recommended plots | New inter

2. Load the Dataset

```
# ----------------------------
# STEP 1: IMPORT 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
import math

# ----------------------------
# STEP 2: LOAD THE DATASET
# ----------------------------
# Instead of reading a new file, use the DataFrame `df` that was c
# when you uploaded the file.
```

FileNotFoundError

Please explain the error:

```
FileNotFoundError: [Errno 2] Unable
```

Hypothesis

The error message `FileNotFoundError: [Errno 2] Unable to open file (unable to open file: name = 'your_model.h5', errno = 2, error message = 'No such file or directory', flags = 0, o_flags = 0)` indicates that the file 'your_model.h5' which is supposed to contain your trained model cannot be found in the current directory.

Step-by-step explanation:

1. The code attempts to load the trained model using `load_model("your_model.h5")` within the `load_artifacts` function.
2. The operating system searches for a file named 'your_model.h5' in the current working directory.
3. Since the file does not exist in the current environment, the operating system raises a `FileNotFoundError`.

Suggested Changes

```
ort streamlit as st
ort pandas as pd
ort numpy as np
m tensorflow.keras.models import loa
ort joblib
ort matplotlib.pyplot as plt
ort os  # Import the os module

-----------------------------------
ETUP
-----------------------------------
set_page_config(page_title="📈 AI St
title("🧠 AI-Driven Stock Price Pred
markdown("Crack the market code with
```

```
# df = pd.read_csv('your_stock_data.csv')

# Show the first few rows
print("Dataset Head:")
print(df.head())

# ... (rest of your code)
```

```
Dataset Head:
  school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;r
0  GP;"F";18;"U";"GT3";"A";4;4;"at_home";"teacher...
1  GP;"F";17;"U";"GT3";"T";1;1;"at_home";"other";...
2  GP;"F";15;"U";"LE3";"T";1;1;"at_home";"other";...
3  GP;"F";15;"U";"GT3";"T";4;2;"health";"services...
4  GP;"F";16;"U";"GT3";"T";3;3;"other";"other";"h...
```

3.Data Exploration

Generate | create a dataframe with 2 columns and 10 ... | Close

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Set styles
sns.set(style="whitegrid")
# plt.style.use('seaborn-darkgrid')  # This line is causing the er
plt.style.use('seaborn-v0_8-darkgrid') # Use 'seaborn-v0_8-darkgri

# ---------------------------------
# STEP 2: LOAD DATA
# ---------------------------------
# df = pd.read_csv('your_stock_data.csv')  # Replace with your fil
# Instead of reading a new file, use the existing DataFrame `df`.
# If you want to reload the data, replace 'uploaded_file.csv' with


# Preview data
print("First 5 rows of the dataset:")
print(df.head())

# Summary statistics
print("\nSummary statistics:")
print(df.describe())

# Check for missing values
print("\nMissing values:")
print(df.isnull().sum())

# ... (Rest of your code)
```

```
First 5 rows of the dataset:
  school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;r
0  GP;"F";18;"U";"GT3";"A";4;4;"at_home";"teacher...
1  GP;"F";17;"U";"GT3";"T";1;1;"at_home";"other";...
2  GP;"F";15;"U";"LE3";"T";1;1;"at_home";"other";...
3  GP;"F";15;"U";"GT3";"T";4;2;"health";"services...
4  GP;"F";16;"U";"GT3";"T";3;3;"other";"other";"h...
```

```
-----------------------------------
OAD MODEL AND SCALER
-----------------------------------
.cache_resource
 load_artifacts():
 # Check if the model file exists be
 if os.path.exists("my_model.h5"):
     model = load_model("my_model.h5
 else:
     st.error("Error: Model file 'my
     return None, None  # Return Non

 # Check if the scaler file exists b
 if os.path.exists("minmax_scaler.pk
     scaler = joblib.load("minmax_sc
 else:
     st.error("Error: Scaler file 'm
     return None, None  # Return Non

 return model, scaler

el, scaler = load_artifacts()

heck if model and scaler were loaded
model is not None and scaler is not
 # ---------------------------------
 # UPLOAD USER FILE
 # ---------------------------------
 uploaded_file = st.file_uploader("U

 if uploaded_file:
     df = pd.read_csv(uploaded_file)

     # Data preview
     st.subheader("📄 Preview Uploa
     st.write(df.tail())

     # Ensure 'Close' column exists
     if 'Close' not in df.columns:
         st.error("CSV must contain
     else:
         # Scale 'Close' column
         df_close = df[['Close']]
         scaled_data = scaler.transf

         # Use last 60 time steps
         if len(scaled_data) < 60:
             st.error("Not enough da
         else:
```

```
Summary statistics:
       school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;F
count                                                    395
unique                                                   395
top     MS;"M";19;"U";"LE3";"T";1;1;"other";"at_home";...
freq                                                       1

Missing values:
school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;rea
dtype: int64
```

4.Check for Missing Values and Duplicates

```
# Assuming your uploaded file is named 'student-dataset.csv'
df = pd.read_csv('student-dataset.csv')
```

```
# ----------------------------------------
# STEP 2: LOAD DATA
# ----------------------------------------
# Replace with your dataset file name or use existing df
# df = pd.read_csv('your_stock_data.csv')
# Since the data was already loaded and is in the 'df' variable,
# you can simply use 'df' directly:

# No need to reload, just continue using the existing 'df'
print("Using existing DataFrame 'df'")

# Convert 'Date' to datetime if it exists
if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'])
```

```
Using existing DataFrame 'df'
```

5. Visualize a Few Features

```
# ----------------------------------------
# STEP 1: IMPORT LIBRARIES
# ----------------------------------------
import pandas as pd
import matplotlib.pyplot as plt

# ----------------------------------------
# STEP 2: LOAD DATA
# ----------------------------------------
# df = pd.read_csv('your_stock_data.csv')  # Replace with your fil
# The data is already loaded in the 'df' DataFrame.
# No need to load it again.

# Convert 'Date' to datetime if it exists
if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'])
    df.set_index('Date', inplace=True)
else:
    print("Warning: 'Date' column not found in DataFrame. Skipping
```

```
                last_60 = scaled_data[-
                X_input = last_60.resha

                # Make prediction
                prediction_scaled = mod
                predicted_price = scale

                # Display result
                st.success(f"📈 Predict

                # Plot
                st.subheader("📊 Closi
                plt.figure(figsize=(10,
                plt.plot(df_close[-60:]
                plt.axhline(predicted_p
                plt.legend()
                st.pyplot(plt)
e:
 st.error("Error: Could not load mod
```

Use code with caution

Explanation of changes:

1. **Import `os`:** The `os` module is imported to use the `os.path.exists()` function for checking if the model and scaler files exist.
2. **Check for model and scaler files:** Inside the `load_artifacts` function:
   - `os.path.exists("my_model.` checks if the model file ('my_model.h5') exists in the current directory. Replace 'my_model.h5' with the actual filename if different.
   - `os.path.exists("minmax_sc` checks for the scaler file. Replace 'minmax_scaler.pkl' with the actual filename if different.
   - If either file is not found, a `st.error` message is displayed to the user, and the function returns `None, None`.
3. **Handle loading errors:** After calling `load_artifacts()`, the code checks if `model` and `scaler` are not `None`. If either is `None`, it means

```
# ---------------------------------------
# STEP 3: VISUALIZE CLOSE PRICE
# ---------------------------------------
# Check if 'Close' column exists before plotting
if 'Close' in df.columns:
    plt.figure(figsize=(14, 5))
    plt.plot(df['Close'], color='blue', label='Close Price')
    plt.title('📈 Close Price Over Time')
    plt.xlabel('Date' if 'Date' in df.columns else 'Index')  # Use
    plt.ylabel('Price')
    plt.legend()
    plt.show()
else:
    print("Warning: 'Close' column not found in DataFrame. Skippir

# ---------------------------------------
# STEP 4: VISUALIZE VOLUME
# ---------------------------------------
# Check if 'Volume' column exists before plotting
if 'Volume' in df.columns:
    plt.figure(figsize=(14, 4))
    plt.plot(df['Volume'], color='orange', label='Volume')
    plt.title('📦 Volume Over Time')
    plt.xlabel('Date' if 'Date' in df.columns else 'Index')  # Use
    plt.ylabel('Volume')
    plt.legend()
    plt.show()
else:
    print("Warning: 'Volume' column not found in DataFrame. Skippi

# ---------------------------------------
# STEP 5: VISUALIZE HIGH & LOW
# ---------------------------------------
# Check if 'High' and 'Low' columns exist before plotting
if 'High' in df.columns and 'Low' in df.columns:
    plt.figure(figsize=(14, 5))
    plt.plot(df['High'], label='High Price', color='green')
    plt.plot(df['Low'], label='Low Price', color='red')
    plt.title('📊 High and Low Prices Over Time')
    plt.xlabel('Date' if 'Date' in df.columns else 'Index')  # Use
    plt.ylabel('Price')
    plt.legend()
    plt.show()
else:
    print("Warning: 'High' or 'Low' column not found in DataFrame.
```

```
Warning: 'Date' column not found in DataFrame. Skipping settir
Warning: 'Close' column not found in DataFrame. Skipping Close
Warning: 'Volume' column not found in DataFrame. Skipping Volu
Warning: 'High' or 'Low' column not found in DataFrame. Skippi
```

6. Identify Target and Features

```
# ---------------------------------------
# STEP 1: IMPORT LIBRARIES
# ---------------------------------------
import pandas as pd
```

there was an error loading them, and a `st.error` message is displayed.

4. **Proceed if loading is successful:** If both `model` and `scaler` are loaded successfully, the rest of the code (uploading the user file, making predictions, and plotting) is executed.

▶ Sources

Rate this answer

```
# ----------------------------------------
# STEP 2: LOAD DATA
# ----------------------------------------
# df = pd.read_csv('your_stock_data.csv')  # Replace with your fil
# The data is already loaded in the 'df' DataFrame from previous s
# No need to reload it again. Instead, continue using the existing

# Convert 'Date' to datetime if it exists
if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'])
    df.set_index('Date', inplace=True)
else:
    print("Warning: 'Date' column not found in DataFrame. Skipping

# ----------------------------------------
# STEP 3: DEFINE FEATURES AND TARGET
# ----------------------------------------

# You can choose your feature set depending on your modeling apprc
# For classic ML (not LSTM), you might use:
features = ['Open', 'High', 'Low', 'Volume']
target = 'Close'  # Assuming 'Close' is your target variable

# ... (Rest of your code to use features and target)
```

```
Warning: 'Date' column not found in DataFrame. Skipping settir
```

7.Convert Categorical Columns to Numerical

```
# ----------------------------------------
# STEP 1: IMPORT LIBRARIES
# ----------------------------------------
import pandas as pd

# ----------------------------------------
# STEP 2: LOAD DATA
# ----------------------------------------
# df = pd.read_csv('your_stock_data.csv')  # Replace with your fil

# The code was trying to read a file named 'your_stock_data.csv' w
# Since you've already loaded the data into a DataFrame named 'df'
# you should continue using that DataFrame instead of trying to lc

# Commenting out the line that reads the file:
# df = pd.read_csv('your_stock_data.csv')

# Instead, you'll be using the existing 'df' DataFrame.
# Add a print statement to confirm this:
print("Using existing DataFrame 'df'")

# Rest of your code...
# Check if 'Date' column exists before processing
if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'])
    df.set_index('Date', inplace=True)
else:
```

```
    print("Warning: 'Date' column not found in DataFrame. Skipping
# ----------------------------------------
# STEP 3: IDENTIFY CATEGORICAL COLUMNS
# ----------------------------------------
categorical_cols = df.select_dtypes(include=['object', 'category']
print(f"🔍 Categorical Columns Found: {categorical_cols}")

# ----------------------------------------
# STEP 4: ENC
```

```
Using existing DataFrame 'df'
Warning: 'Date' column not found in DataFrame. Skipping Date p
🔍 Categorical Columns Found: ['school;sex;age;address;famsiz
```

8. One-Hot Encoding

```
# ----------------------------------------
# STEP 1: IMPORT LIBRARIES
# ----------------------------------------
import pandas as pd

# ----------------------------------------
# STEP 2: LOAD DATA
# ----------------------------------------
# df = pd.read_csv('your_stock_data.csv')  # Replace with your fil
# The data has already been loaded into the 'df' DataFrame.
# We will continue using that.
# df['Date'] = pd.to_datetime(df['Date']) #This line and the next
# df.set_index('Date', inplace=True)     #Remove these as the data

# ----------------------------------------
# STEP 3: IDENTIFY CATEGORICAL COLUMNS
# ----------------------------------------
categorical_cols = df.select_dtypes(include=['object', 'category']
print(f"🔍 Categorical columns: {categorical_cols}")

# ----------------------------------------
# STEP 4: APPLY ONE-HOT ENCODING
# ----------------------------------------
df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_fir

# ----------------------------------------
# STEP 5: DISPLAY RESULTS
# ----------------------------------------
print("\n✅ Data after One-Hot Encoding:")
print(df_encoded.head())
```

```
3                                                   False
4                                                   False

   school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fj
0                                                   False
1                                                   False
2                                                   False
3                                                   False
4                                                   False

   school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fj
0                                                   False
1                                                   False
2                                                   False
3                                                   False
4                                                   False

   school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fj
0                                                   False
1                                                   False
2                                                   False
3                                                   False
4                                                   False

   school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fj
0                                                   False
1                                                   False
2                                                   False
3                                                   False
4                                                   False

   school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fj
0                                                   False
1                                                   False
2                                                   False
3                                                   False
4                                                   False

   school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fj
0                                                   False
1                                                   False
2                                                   False
3                                                   False
4                                                   False

[5 rows x 394 columns]
```

9.Feature Scaling

```
# ------------------------------------------
# STEP 1: IMPORT LIBRARIES
# ------------------------------------------
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# ------------------------------------------
# STEP 2: LOAD DATA
# ------------------------------------------
# df = pd.read_csv('your_stock_data.csv')  # Replace with your act
# The data has already been loaded into the 'df' DataFrame.
# We will continue using that.
```

```
# ---------------------------------------
# STEP 3: FEATURE SCALING
```

10. Train-Test Split

```
# df = pd.read_csv('your_stock_data.csv')  # Remove or replace wit


# ---------------------------------------
# STEP 1: IMPORT LIBRARIES
# ---------------------------------------
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# ---------------------------------------
# STEP 2: LOAD
```

11.Model Building

```
# ---------------------------------------
# STEP 1: IMPORT LIBRARIES
# ---------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

# ---------------------------------------
# STEP 2: LOAD AND PREPARE DATA
# ---------------------------------------
# df = pd.read_csv('your_stock_data.csv')  # Replace with
```

12.Evaluation

```
# Assuming 'df_encoded' from your previous step contains your feat
# Separate features (X) and target (y)
# Get all column names except the target
# features = [col for col in df_encoded.columns if col != target]
# OR

# Before one-hot encoding, store the 'Close' column
# target_values = df['Close']  # Store 'Close' values - This line
# Instead of trying to access 'Close' directly, we need to:

# 1. Split the single column in 'df' into multiple columns
df = df['school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjc
# 2. Give appropriate column names, assuming the last column is yc
# ... (replace with your actual column names based on your data)
columns = ['school', 'sex', 'age', 'address', 'famsize', 'Pstatus'
df.columns = columns
# Now you can access the 'Close' column
```

```
target_values = df['Close']  # Store 'Close' values

# ... your one-hot encoding code ...
categorical_cols = df.select_dtypes(include=['object', 'category']
print(f"🔍 Categorical columns: {categorical_cols}")

df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_fir

# After one-hot encoding, add the 'Close' column back
df_encoded['Close'] = target_values  # Add 'Close' back to df_encc

features = df_encoded.columns[~df_encoded.columns.isin([target])].

# Check if the target column is present in df_encoded
if target in df_encoded.columns:
    X = df_encoded[features]
    y = df_encoded[target]   # Assuming 'target' variable is defin
else:
    # If the target column is not present, print an error message
    raise KeyError(f"Target column '{target}' not found in df_encc

# Assuming 'scaler' is your MinMaxScaler object from previous step
X = scaler.fit_transform(X)  # Apply feature scaling to X

# Reshape input to be [samples, time steps, features] which is rec
X = X.reshape(X.shape[0], 1, X.shape[1])

# Split data into train, validation, and test sets
from sklearn.model_selection import train_test_split # Make sure t
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_siz
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, te
```

```
🔍 Categorical columns: ['school', 'sex', 'age', 'address', '
```

13.Make Predictions from New Input

```
# ... previous code ...

# Instead of loading from a file, use the existing 'scaler' object
# scaler = joblib.load('minmax_scaler.pkl')  # Comment out this li
# Since 'scaler' is already a MinMaxScaler object (from Global Var
# you can directly use it.

# ... rest of your code ...
```

14. Convert to DataFrame and Encode

```
# ----------------------------------------
# STEP 1: IMPORT LIBRARIES
# ----------------------------------------
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import os # Import the os module

# ----------------------------------------
# STEP 2: CONVERT NEW INPUT TO DATAFRAME
```

```
# ---------------------------------------
# Example: assume new raw data as a dictionary
new_input = {
    'Date': ['2025-05-09'],
    'Open': [152.45],
    'High': [155.00],
    'Low': [150.80],
    'Close': [154.20],
    'Volume': [2_000_000],
    'Ticker': ['AAPL'],            # Example categorical column
    'Sector': ['Technology']       # Example categorical column
}

df_new = pd.DataFrame(new_input)
df_new['Date'] = pd.to_datetime(df_new['Date'])
df_new.set_index('Date', inplace=True)

# ---------------------------------------
```

15.Predict the Final Grade

```
# Instead of:
# model = load_model('your_model.h5')  # Load trained model

# First, you need to train your model and save it.
# If you have already trained it in another notebook or session,
# you'll need to upload the 'your_model.h5' file to the current er

# Assuming you've trained the model and saved it as 'my_model.h5':
# If you have the trained model file, upload it to the current env
from google.colab import files
uploaded = files.upload()  # Upload 'my_model.h5'

# Then load the model:
from tensorflow.keras.models import load_model

# Check if the model file was uploaded successfully
if 'my_model.h5' in uploaded:
    model = load_model('my_model.h5')  # Load trained model
else:
    print("Error: 'my_model.h5' not found in uploaded files. Pleas
    # You might want to stop execution here to avoid further error
    # import sys
    # sys.exit(1)
```

Choose files student-dataset.csv

- **student-dataset.csv**(text/csv) - 56993 bytes, last modified: 08/05/2025 - 100% done

Saving student-dataset.csv to student-dataset (2).csv

16. Deployment-Building an Interactive App

```
pip install streamlit
```

```
Collecting streamlit
  Downloading streamlit-1.45.0-py3-none-any.whl.metadata (8.9
Requirement already satisfied: altair<6,>=4.0 in /usr/local/li
Requirement already satisfied: blinker<2,>=1.5.0 in /usr/local
Requirement already satisfied: cachetools<6,>=4.0 in /usr/loca
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/li
Requirement already satisfied: packaging<25,>=20 in /usr/local
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local
Requirement already satisfied: protobuf<7,>=3.20 in /usr/local
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/
Requirement already satisfied: requests<3,>=2.27 in /usr/local
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/loc
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/l
Requirement already satisfied: typing-extensions<5,>=4.4.0 in
Collecting watchdog<7,>=2.1.5 (from streamlit)
  Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.3/44.3 kB 1.7
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 ir
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local
Requirement already satisfied: jinja2 in /usr/local/lib/python
Requirement already satisfied: jsonschema>=3.0 in /usr/local/l
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/l
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/
Requirement already satisfied: tzdata>=2022.7 in /usr/local/li
Requirement already satisfied: charset-normalizer<4,>=2 in /us
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/loca
Requirement already satisfied: certifi>=2017.4.17 in /usr/loca
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/l
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/l
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib
Requirement already satisfied: jsonschema-specifications>=2023
Requirement already satisfied: referencing>=0.28.4 in /usr/loc
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/li
Requirement already satisfied: six>=1.5 in /usr/local/lib/pyth
Downloading streamlit-1.45.0-py3-none-any.whl (9.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.9/9.9 MB 45.1 MB
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 42.2 MB
Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl (
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.1/79.1 kB 5.0 M
Installing collected packages: watchdog, pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.45.0 watchdog-
```

```
import streamlit as st
import pandas as pd
import numpy as np
from tensorflow.keras.models import load_model
import joblib
import matplotlib.pyplot as plt
import os  # Import the os module

# ---------------------------------------
# SETUP
# ---------------------------------------
st.set_page_config(page_title="📈 AI Stock Predictor", layout="cent
```

```
    st.title("🧠 AI-Driven Stock Price Predictor")
    st.markdown("Crack the market code with LSTM-powered time series fo

    # --------------------------------------
    # LOAD MODEL AND SCALER
    # --------------------------------------
    @st.cache_resource
    def load_artifacts():
        # Check if the model file exists before loading
        if os.path.exists("my_model.h5"):  # Replace 'my_model.h5' with
            model = load_model("my_model.h5")  # Your trained LSTM mode
        else:
            st.error("Error: Model file 'my_model.h5' not found. Please
            return None, None  # Return None if model loading fails

        # Check if the scaler file exists before loading
        if os.path.exists("minmax_scaler.pkl"):  # Replace 'minmax_scal
            scaler = joblib.load("minmax_scaler.pkl")  # Trained scaler
        else:
            st.error("Error: Scaler file 'minmax_scaler.pkl' not found.
            return None, None  # Return None if scaler loading fails

        return model, scaler

    model, scaler = load_artifacts()

    # Check if model and scaler were loaded successfully
    if model is not None and scaler is not None:
        # --------------------------------------
        # UPLOAD USER FILE
        # --------------------------------------
        uploaded_file = st.file_uploader("Upload latest stock data (CSV

        if uploaded_file:
            df = pd.read_csv(uploaded_file)

            # Data preview
            st.subheader("📄 Preview Uploaded Data")
            st.write(df.tail())

            # Ensure 'Close' column exists
            if 'Close' not in df.columns:
                st.error("CSV must contain a 'Close' column.")
            else:
                # Scale 'Close' column
                df_close = df[['Close']]
                scaled_data = scaler.transform(df_close)

                # Use last 60 time steps
                if len(scaled_data) < 60:
                    st.error("Not enough data. Please upload at least 6
                else:
                    last_60 = scaled_data[-60:]
                    X_input = last_60.reshape(1, 60, 1)

                    # Make prediction
                    prediction_scaled = model.predict(X_input)
                    predicted_price = scaler.inverse_transform(predicti

                    # Display result
```

```
                st.success(f"📈 Predicted Closing Price: **${predic

                # Plot
                st.subheader("📊 Closing Price - Last 60 Days")
                plt.figure(figsize=(10, 4))
                plt.plot(df_close[-60:].values, label="Historical C
                plt.axhline(predicted_price, color='red', linestyle
                plt.legend()
                st.pyplot(plt)
else:
    st.error("Error: Could not load model or scaler. Please check t
```

```
2025-05-10 12:20:51.318 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.322 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.327 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.333 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.334 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.338 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.344 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.345 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.348 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.349 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.350 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.352 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.353 Thread 'MainThread': missing ScriptRur
2025-05-10 12:20:51.354 Thread 'MainThread': missing ScriptRur
```

Enter a prompt here

0/2000

Gemini can make mistakes, so double-check responses and use code with caution. Learn more