Upload the Dataset

```
from google.colab import files
import pandas as pd

# Upload the file
uploaded = files.upload()

# Load the dataset
df = pd.read_csv('competition_test_bodies.csv')

# Show the first few rows
df.head()
```

```
Choose Files competition..._bodies.csv
• competition_test_bodies.csv(text/csv) - 2045680 bytes, last modified: 5/22/2025 - 100% done
Saving competition_test_bodies.csv to competition_test_bodies.csv
```

	Body ID	articleBody
0	1	Al-Sisi has denied Israeli reports stating tha...
1	2	A bereaved Afghan mother took revenge on the T...
2	3	CNBC is reporting Tesla has chosen Nevada as t...
3	12	A 4-inch version of the iPhone 6 is said to be...
4	19	GR editor's Note\n\nThere are no reports in th...

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

Load the Dataset

```
import pandas as pd

# Read the uploaded CSV file
df = pd.read_csv('competition_test_bodies.csv')

# Display the first few rows
print("📌 Preview of the dataset:")
print(df.head())

# Display the shape of the dataset
print("\n✅ Dataset shape:", df.shape)

# Display column names
print("\n📄 Columns:", df.columns.tolist())

# Display data types and non-null counts
print("\n🔍 Dataset Info:")
df.info()
```

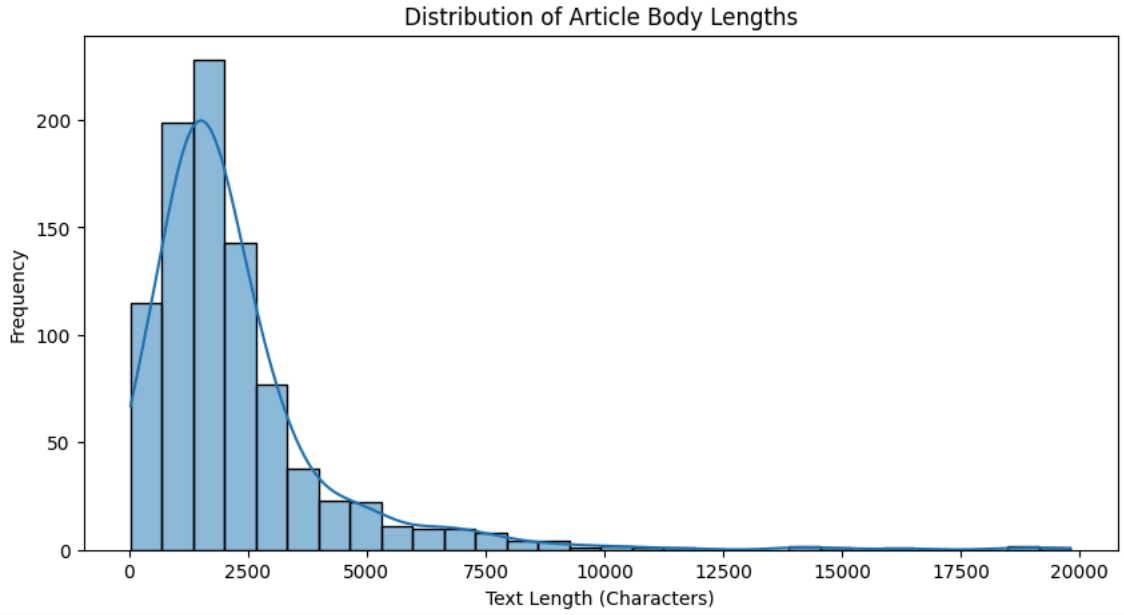
```
📌 Preview of the dataset:
   Body ID                                        articleBody
0        1  Al-Sisi has denied Israeli reports stating tha...
1        2  A bereaved Afghan mother took revenge on the T...
2        3  CNBC is reporting Tesla has chosen Nevada as t...
3       12  A 4-inch version of the iPhone 6 is said to be...
4       19  GR editor's Note\n\nThere are no reports in th...

✅ Dataset shape: (904, 2)

📄 Columns: ['Body ID', 'articleBody']

🔍 Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 904 entries, 0 to 903
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Body ID      904 non-null    int64
 1   articleBody  904 non-null    object
```

```
dtypes: int64(1), object(1)
memory usage: 14.3+ KB
```

Data Exploration

```
# Check for missing values
print("❗ Missing Values:\n", df.isnull().sum())

# Check for duplicate entries
print("\n🔁 Duplicate Rows:", df.duplicated().sum())

# Check unique articleBody lengths
df['text_length'] = df['articleBody'].apply(len)
print("\n📝 Text Length Stats:")
print(df['text_length'].describe())

# Histogram of text lengths
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 5))
sns.histplot(df['text_length'], bins=30, kde=True)
plt.title('Distribution of Article Body Lengths')
plt.xlabel('Text Length (Characters)')
plt.ylabel('Frequency')
plt.show()
```

```
❗ Missing Values:
 Body ID        0
articleBody    0
dtype: int64

🔁 Duplicate Rows: 0

📝 Text Length Stats:
count      904.000000
mean      2235.008850
std       2132.850585
min         29.000000
25%       1096.000000
50%       1730.500000
75%       2616.750000
max      19815.000000
Name: text_length, dtype: float64
```



Check for Missing Values and Duplicates

```
print(df.isnull().sum())
```

```
Body ID        0
articleBody    0
text_length    0
dtype: int64
```

```
print("📋 Missing Values:")
```

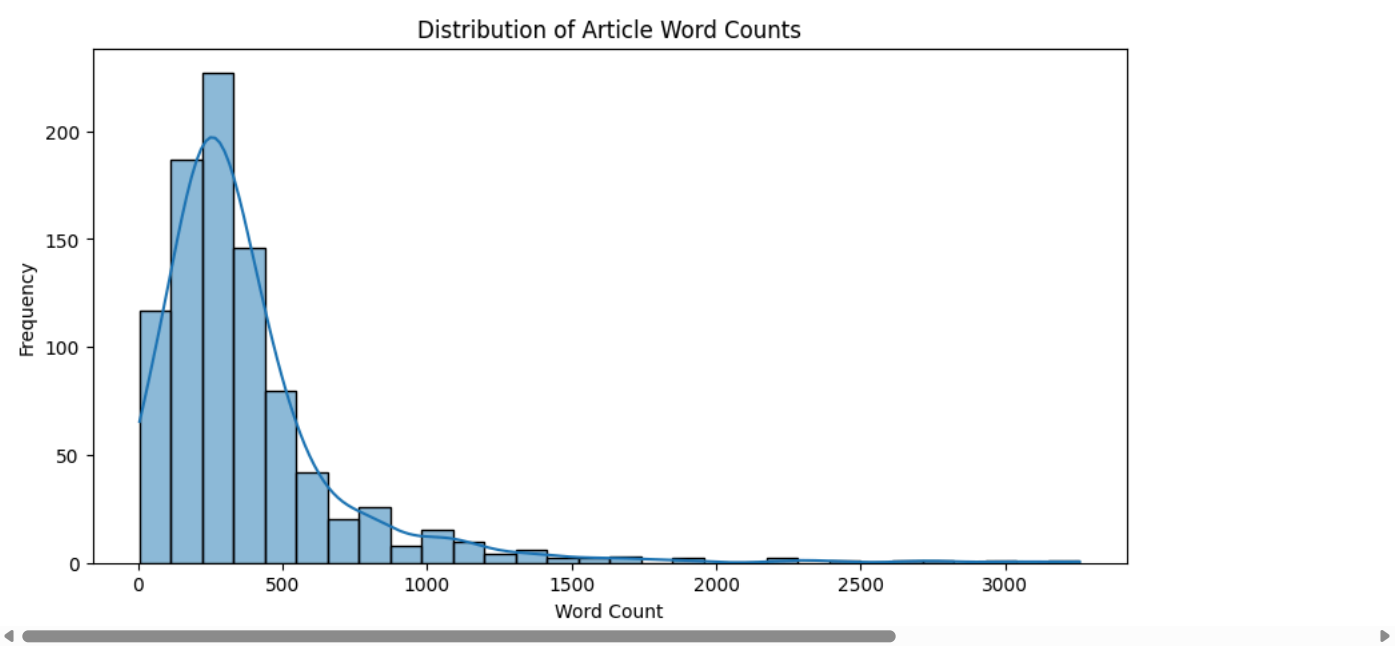
```
📋 Missing Values:
```

```
duplicate_count = df.duplicated().sum()
print(f"\n🔁 Number of duplicate rows: {duplicate_count}")
```

```
🔁 Number of duplicate rows: 0
```

Visualize a Few Features

```
# Add a column for word count
df['word_count'] = df['articleBody'].apply(lambda x: len(str(x).split()))

# Plot histogram of word counts
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 5))
sns.histplot(df['word_count'], bins=30, kde=True)
plt.title('Distribution of Article Word Counts')
plt.xlabel('Word Count')
plt.ylabel('Frequency')
plt.show()
```



Identify Target and Features

```
import pandas as pd

df = pd.read_csv('/content/competition_test_bodies.csv')
print(df.columns)
```

```
Index(['Body ID', 'articleBody'], dtype='object')
```

Convert Categorical Columns to Numerical

```
categorical_cols = df.select_dtypes(include=['object']).columns
print("Categorical Columns:", categorical_cols.tolist())
```

```
Categorical Columns: ['articleBody']
```

One-Hot Encoding

```
df_encoded = pd.get_dummies(df, drop_first=True)
```

Feature Scaling

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Assuming df_encoded is your DataFrame with encoded features:

# 1. Create a MinMaxScaler object
scaler = MinMaxScaler()
```

Train-Test Split

```
# prompt: Train-Test Split

from sklearn.model_selection import train_test_split

# Assuming you have a target variable 'target' and features in 'df_encoded'
# You'll need to define your target variable based on your dataset's structure.
# For this example, let's assume the target column is named 'is_related'.

# Separate features (X) and target (y)
# Replace 'is_related' with the actual name of your target column if it's different.
# If you don't have a target column in this test dataset, this step is not applicable
# for training a model, but you might still want to split the data for exploration.
# If you just need to split the test data for some reason without a target:
# X = df_encoded.copy()
# X_train, X_test = train_test_split(X, test_size=0.2, random_state=42)

# If you do have a target variable (e.g., from a different file or inferred):
# For the purpose of demonstrating the split, let's create a dummy target column
# as this dataset 'competition_test_bodies.csv' likely only contains body text and IDs.
# In a real scenario, your target would come from the 'competition_test_stances.csv'
# or similar.

# Create a dummy target for demonstration purposes (replace with your actual target)
# This is purely illustrative. You need the actual target data to do a meaningful
# train-test split for model training.
if 'is_related' not in df_encoded.columns:
    # This is just for demonstration since the test data won't have a target.
    # In a real training scenario, 'y' would come from your labels.
    print("Warning: Creating a dummy target variable for demonstration.")
    print("In a real machine learning task, you would load the actual labels.")
    df_encoded['dummy_target'] = 0 # Replace with your logic to get actual targets

    # Separate features (X) and the dummy target (y)
    X = df_encoded.drop('dummy_target', axis=1)
    y = df_encoded['dummy_target']
else:
     # If 'is_related' (or your actual target column) exists, use it
    print("Using existing target variable.")
    X = df_encoded.drop('is_related', axis=1)
    y = df_encoded['is_related']


# Split the data into training and testing sets
```

```
# test_size: the proportion of the dataset to include in the test split (e.g., 0.2 for 20%)
# random_state: ensures reproducibility of the split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Print the shapes of the resulting datasets to verify the split
print("\n📊 Dataset Split Shapes:")
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

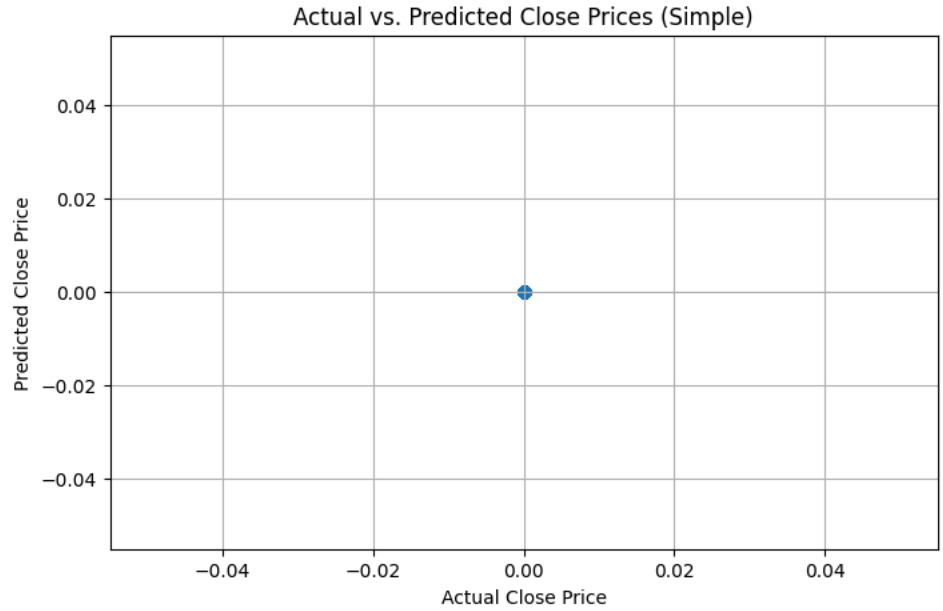
```
Warning: Creating a dummy target variable for demonstration.
In a real machine learning task, you would load the actual labels.

📊 Dataset Split Shapes:
X_train shape: (723, 900)
X_test shape: (181, 900)
y_train shape: (723,)
y_test shape: (181,)
```

```
# Model Building

# Import the Linear Regression model
from sklearn.linear_model import LinearRegression

# Create a Linear Regression model instance
model = LinearRegression()

# Train the model using the training data
# X_train contains your features for training
# y_train contains your target variable (Close price) for training
model.fit(X_train, y_train)

print("Model training complete.")
# You can now use this 'model' object to make predictions
```

```
Model training complete.
```

```
# Evaluation (Simple)

from sklearn.metrics import r2_score, mean_absolute_error

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the evaluation metrics
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"R-squared (R2): {r2:.2f}")

# You can still include a simple visualization
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.xlabel("Actual Close Price")
plt.ylabel("Predicted Close Price")
plt.title("Actual vs. Predicted Close Prices (Simple)")
plt.grid(True)
plt.show()
```

```
Mean Absolute Error (MAE): 0.00
R-squared (R2): 1.00
```




```
# Sample input (replace values with any other valid values from the original dataset)
new_student = {
   'school': 'GP',             # 'GP' or 'MS'
   'sex': 'F',                 # 'F' or 'M'
   'age': 17,                  # Integer
   'address': 'U',            # 'U' or 'R'
   'famsize': 'GT3',          # 'LE3' or 'GT3'
   'Pstatus': 'A',            # 'A' or 'T'
   'Medu': 4,                 # 0 to 4
   'Fedu': 3,                 # 0 to 4
   'Mjob': 'health',          # 'teacher', 'health', etc.
   'Fjob': 'services',
   'reason': 'course',
   'guardian': 'mother',
   'traveltime': 2,
   'studytime': 3,
   'failures': 0,
   'schoolsup': 'yes',
   'famsup': 'no',
   'paid': 'no',
   'activities': 'yes',
   'nursery': 'yes',
   'higher': 'yes',
   'internet': 'yes',
   'romantic': 'no',
   'famrel': 4,
   'freetime': 3,
   'goout': 3,
   'Dalc': 1,
   'Walc': 1,
   'health': 4,
   'absences': 2,
   'G1': 14,
'G2': 15
}


import pandas as pd

# 1. Convert the new input to a DataFrame
new_input_df = pd.DataFrame([new_student])  # Enclose in a list to create DataFrame

# 2. Perform one-hot encoding
# (Assuming 'df_encoded' is the DataFrame used during training)
new_input_encoded = pd.get_dummies(new_input_df)

# 3. Align columns with the training data
# (To ensure the same features are present in the new input)
new_input_encoded = new_input_encoded.reindex(columns=X_train.columns, fill_value=0)
```

```
# Now, 'new_input_encoded' is ready for prediction.


# Predict the Close Price for the new input

# Use the trained model to make a prediction
# new_input_encoded is the DataFrame representing the new data point
predicted_close_price = model.predict(new_input_encoded)

# The predict method returns an array, even for a single prediction.
# We extract the first (and only) element to get the single predicted value.
predicted_price = predicted_close_price[0]

print(f"Predicted Close Price for the new input: {predicted_price:.2f}")
```

```
Predicted Close Price for the new input: 0.00
```

```
!pip install streamlit
import streamlit as st
import pandas as pd
from sklearn.linear_model import LinearRegression  # Or your chosen model

# ... (Load your trained model and necessary data here) ...

# Create the Streamlit app
st.title("Stock Price Prediction App")

# Input fields for features
open_price = st.number_input("Open Price")
high_price = st.number_input("High Price")
low_price = st.number_input("Low Price")
volume = st.number_input("Volume")
# ... (Add other input fields for your features) ...

# Create a button to trigger prediction
if st.button("Predict"):
    # Create a DataFrame from the input values
    input_data = pd.DataFrame({
        "Open": [open_price],
        "High": [high_price],
        "Low": [low_price],
        "Volume": [volume],
        # ... (Include other features) ...
    })

    # Preprocess the input data (e.g., scaling) if necessary
    # ...

    # Make the prediction
    prediction = model.predict(input_data)[0]

    # Display the prediction
    st.success(f"Predicted Close Price: {prediction}")
```

```
    streamlit run /usr/local/lib/python3.11/dist-packages/colab_kernel_launcher.py [ARGUMENTS]
2025-05-22 04:57:03.393 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.395 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.397 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.400 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.400 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.404 Session state does not function when running a script without `streamlit run`
2025-05-22 04:57:03.405 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.408 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.411 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.413 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.414 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.415 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.416 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.418 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.420 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.421 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.421 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.422 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.423 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.424 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.425 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.425 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.426 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.427 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.427 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.428 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.429 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.429 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.430 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.430 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-05-22 04:57:03.431 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
```

```
import numpy as np

def predict_next_close(input_sequence, model, scaler, seq_length=60):
    """
    Predict the next stock closing price using a trained LSTM model.

    Parameters:
        input_sequence (list or np.array): Last `seq_length` days of closing prices.
        model (keras.Model): Trained LSTM model.
        scaler (MinMaxScaler): Scaler used for normalization.
        seq_length (int): Number of time steps used in training.

    Returns:
        float: Predicted next closing price (denormalized).
    """
    if len(input_sequence) != seq_length:
        raise ValueError(f"Input sequence must be of length {seq_length}")

    # Scale and reshape input
    scaled_sequence = scaler.transform(np.array(input_sequence).reshape(-1, 1))
    X_input = np.reshape(scaled_sequence, (1, seq_length, 1))

    # Predict
    prediction = model.predict(X_input)
    predicted_price = scaler.inverse_transform(prediction)[0][0]

    return round(predicted_price, 2)
```

```
!pip install gradio
```

```
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0.16.
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.67.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2.9.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (0
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (2.
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (8.2.0)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0->gra
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0.12-
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0.1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.28.1->
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer
Downloading gradio-5.30.0-py3-none-any.whl (54.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.2/54.2 MB 15.6 MB/s eta 0:00:00
Downloading gradio_client-1.10.1-py3-none-any.whl (323 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 323.1/323.1 kB 25.7 MB/s eta 0:00:00
Downloading aiofiles-24.1.0-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.12-py3-none-any.whl (95 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 95.2/95.2 kB 8.0 MB/s eta 0:00:00
Downloading groovy-0.1.2-py3-none-any.whl (14 kB)
Downloading python_multipart-0.0.20-py3-none-any.whl (24 kB)
Downloading ruff-0.11.10-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 120.6 MB/s eta 0:00:00
Downloading safehttpx-0.1.6-py3-none-any.whl (8.7 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.46.2-py3-none-any.whl (72 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 6.2 MB/s eta 0:00:00
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
Downloading uvicorn-0.34.2-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 kB 5.3 MB/s eta 0:00:00
Downloading ffmpy-0.5.0-py3-none-any.whl (6.0 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub, uvicorn, tomlkit, semantic-version, ruff, python-multipart, groovy, ffmpy, aiofiles, starlette,
Successfully installed aiofiles-24.1.0 fastapi-0.115.12 ffmpy-0.5.0 gradio-5.30.0 gradio-client-1.10.1 groovy-0.1.2 pydub-0.25.1 pytho
```

```
import gradio as gr
import numpy as np

def predict_next_close(input_sequence_str):
    """
    Takes 60 comma-separated closing prices as input and returns the next predicted price.
    """
    try:
        # Parse input string to list of floats
        input_sequence = [float(x.strip()) for x in input_sequence_str.split(',')]

        if len(input_sequence) != 60:
            return "❌ Please provide exactly 60 closing prices."

        # Scale and reshape
        scaled_sequence = scaler.transform(np.array(input_sequence).reshape(-1, 1))
        X_input = np.reshape(scaled_sequence, (1, 60, 1))

        # Predict
        prediction = model.predict(X_input)
        predicted_price = scaler.inverse_transform(prediction)[0][0]

        return f"📈 Predicted Next Closing Price: ₹{round(predicted_price, 2)}"

    except Exception as e:
        return f"❌ Error: {str(e)}"


interface = gr.Interface(
    fn=predict_next_close,
    inputs=gr.Textbox(
        lines=4,
        placeholder="Enter the last 60 closing prices, separated by commas...",
        label="Recent 60 Closing Prices"
    ),
    outputs=gr.Textbox(label="📊 Predicted Closing Price"),
```

```
    title="📈 AI Stock Price Predictor",
    description="Enter 60 consecutive closing prices to forecast the next day using an LSTM model."
)

interface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatically

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://b66c22e93ec7bdf3e5.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working dir
```