Upload the dataset

```
# Upload the Dataset
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving stock_data.csv to stock_data (1).csv
```

Load the dataset

```
# Load the Dataset
import pandas as pd

df = pd.read_csv('stock_data.csv')
df.head()
```

	Open	Close	High	Low	Volume	RSI	MACD	Bollinger_Upper	B
0	0.374639	0.374780	0.373510	0.378390	0.298909	0.847286	0.741715	0.367146	
1	0.950982	0.937746	0.938422	0.946158	0.094805	0.494543	0.881343	0.938396	
2	0.732198	0.719825	0.723644	0.723158	0.126348	0.195471	0.463179	0.710666	
3	0.598823	0.599865	0.596973	0.605322	0.180662	0.736684	0.289076	0.593793	
4	0.156053	0.163410	0.155891	0.166084	0.203646	0.418698	0.318761	0.164158	

Data Exploration

```
df.head()
```

	Open	Close	High	Low	Volume	RSI	MACD	Bollinger_Upper	Bc
0	0.374639	0.374780	0.373510	0.378390	0.298909	0.847286	0.741715	0.367146	
1	0.950982	0.937746	0.938422	0.946158	0.094805	0.494543	0.881343	0.938396	
2	0.732198	0.719825	0.723644	0.723158	0.126348	0.195471	0.463179	0.710666	
3	0.598823	0.599865	0.596973	0.605322	0.180662	0.736684	0.289076	0.593793	
4	0.156053	0.163410	0.155891	0.166084	0.203646	0.418698	0.318761	0.164158	

Check for Missing Values and Duplicates

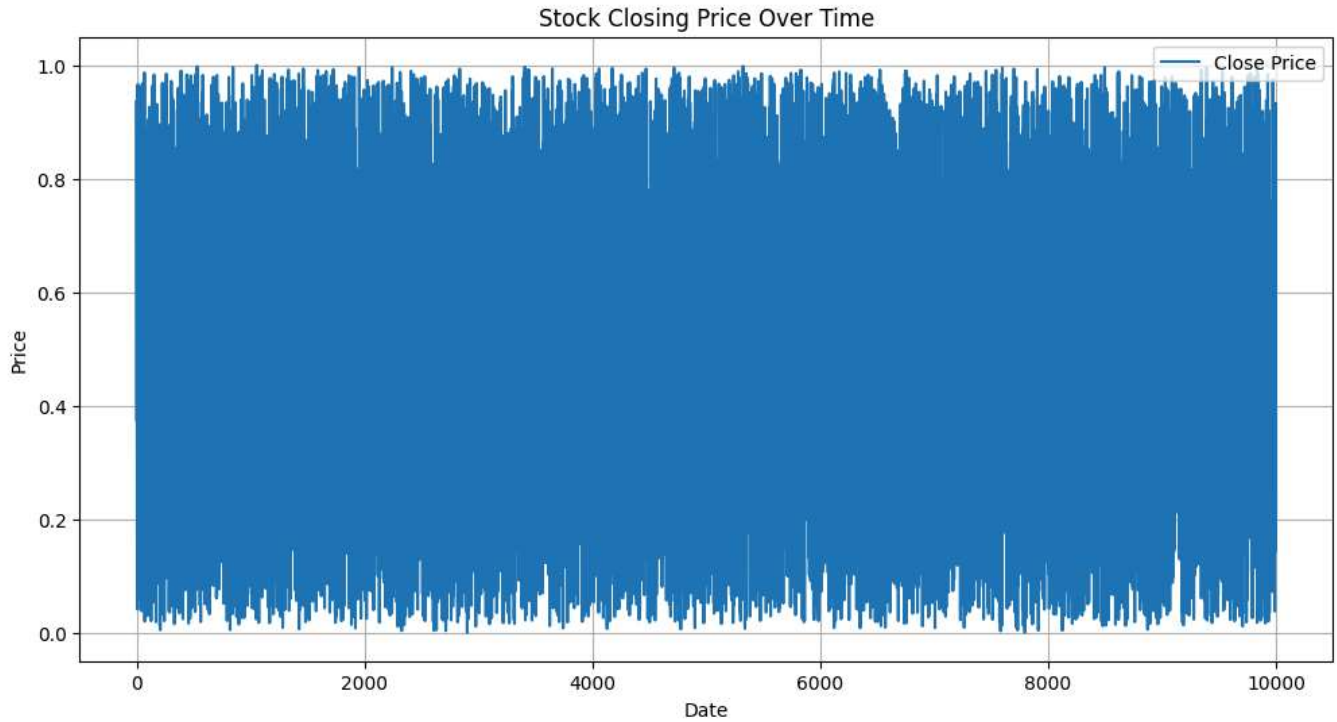
```
print(df.isnull().sum())
print("Duplicate rows:", df.duplicated().sum())
```

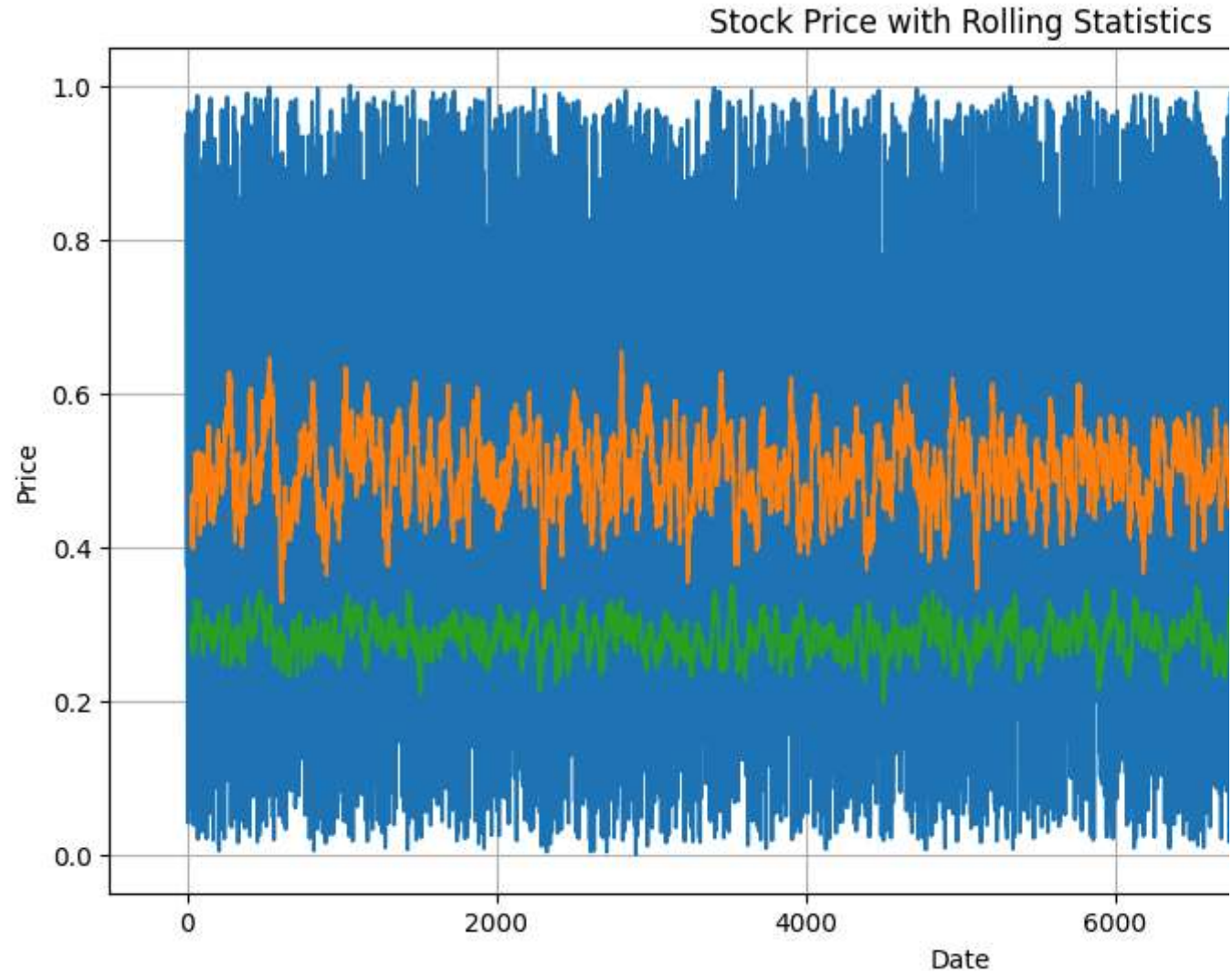
```
Open               0
Close              0
High               0
Low                0
Volume             0
RSI                0
MACD               0
Bollinger_Upper    0
Bollinger_Lower    0
Sentiment_Score    0
GDP_Growth         0
Inflation_Rate     0
Target             0
dtype: int64
Duplicate rows: 0
```

Visualize a Few Features

```
import matplotlib.pyplot as plt

# Plot the closing price over time
plt.figure(figsize=(12, 6))
plt.plot(df['Close'], label='Close Price')
plt.title('Stock Closing Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()

# You can also visualize other features like rolling mean and standard deviation
# Calculate rolling mean and standard deviation
```

```
df['Rolling Mean'] = df['Close'].rolling(window=30).mean()  # 30-day rolling mean
df['Rolling Std'] = df['Close'].rolling(window=30).std()

# Plot rolling statistics along with closing price
plt.figure(figsize=(12, 6))
plt.plot(df['Close'], label='Close Price')
plt.plot(df['Rolling Mean'], label='Rolling Mean (30 days)')
plt.plot(df['Rolling Std'], label='Rolling Std (30 days)')
plt.title('Stock Price with Rolling Statistics')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```


Stock Closing Price Over Time
Close Price
Price
Date
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000


Stock Price with Rolling Statistics
Price
Date
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000

Identify Target and Features

```
# Load the dataset
import pandas as pd

df = pd.read_csv('stock_data.csv')
print(df.columns)
```

```
Index(['Open', 'Close', 'High', 'Low', 'Volume', 'RSI', 'MACD',
       'Bollinger_Upper', 'Bollinger_Lower', 'Sentiment_Score', 'GDP_Growth',
       'Inflation_Rate', 'Target'],
      dtype='object')
```

Convert Categorical Columns to Numerical

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

```
Categorical Columns: []
```

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()

# 2. Select the numerical features to scale
#    (excluding the target variable if it's in
```

Train-Test Split

```
from sklearn.model_selection import train_test_split

# Assuming df_encoded is your DataFrame with features and target:
```

```
# and 'Close' is your target variable

# 1. Separate features (X) and target (y)
X = df_encoded.drop('Close', axis=1)  # Features (all columns except 'Close')
y = df_encoded['Close']              # Target variable ('Close')

# 2. Perform the split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Explanation of parameters:
# - X, y: Your features and target data
# - test_size: Proportion of data to include in the test split (e.g., 0.2 for 20%)
```

Model Building

```
# 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
```

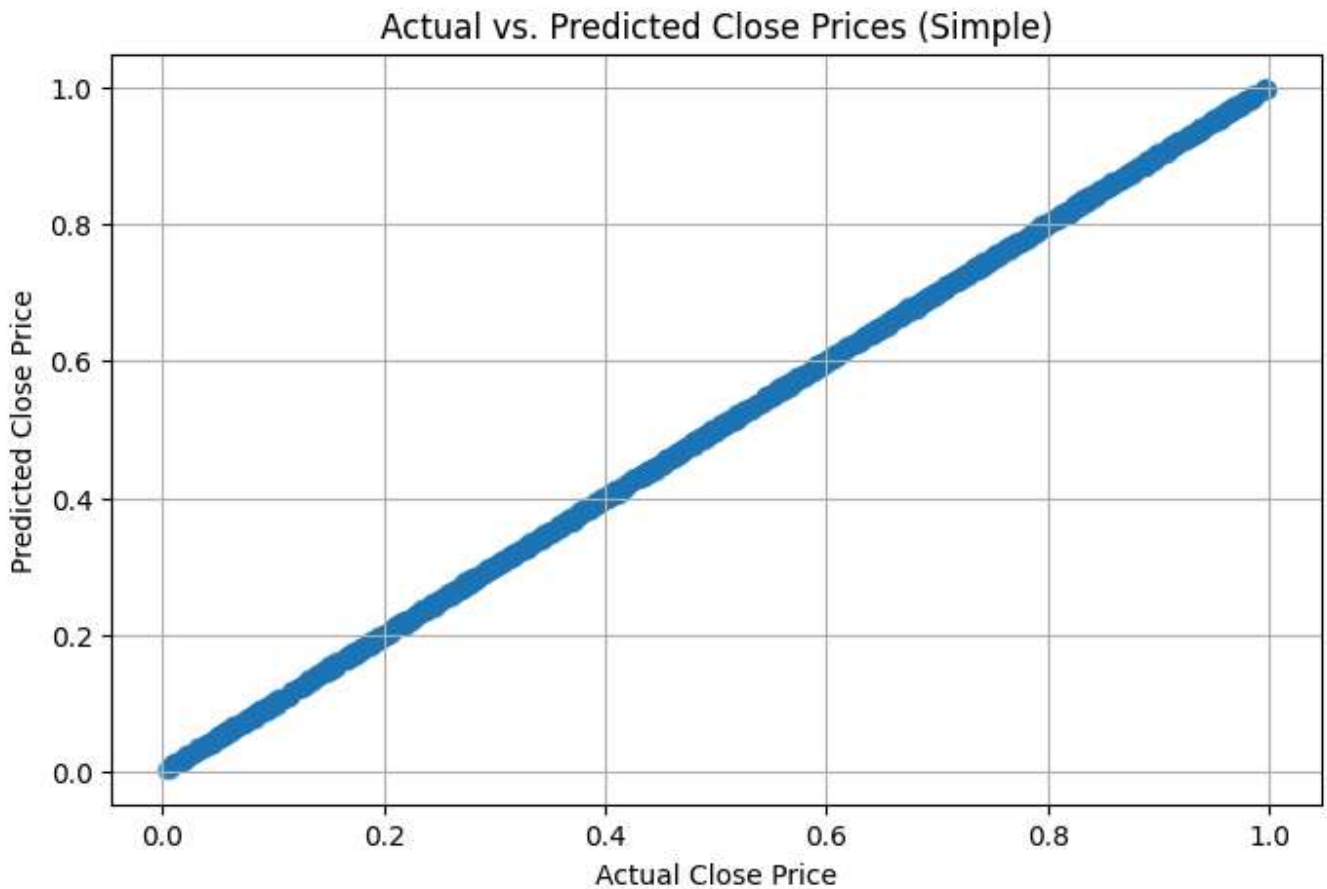
```
Model training complete.
```

Evaluation

```
# 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
```



Make Predictions from New Input

```
# 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
 }
```

Convert to DataFrame and Encode

```
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 Final Grade

```
# 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
```

Deployment-Building an Interactive App

```
!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}")
```

```
Requirement already satisfied: streamlit in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: blinker<2,>=1.5.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: protobuf<7,>=3.20 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /usr/local/lib/python3.
Requirement already satisfied: watchdog<7,>=2.1.5 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3
Requirement already satisfied: pydeck<1,>=0.8.0b4 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f
2025-05-13 09:26:37.721 WARNING streamlit.runtime.scriptrunner_utils.script_run_conte
2025-05-13 09:26:38.126
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run /usr/local/lib/python3.11/dist-packages/colab_kernel_launcher.py [A
2025-05-13 09:26:38.129 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.130 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.133 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.136 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.138 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.139 Session state does not function when running a script without
2025-05-13 09:26:38.140 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.143 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.144 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.149 Thread 'MainThread': missing ScriptRunContext! This warning c
```

```
2025-05-13 09:26:38.154 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.157 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.162 Thread 'MainThread': missing ScriptRunContext! This warning c
2025-05-13 09:26:38.169 Thread 'MainThread': missing ScriptRunContext! This warning c
```

Create a Prediction Function

```
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)
```

Create the Gradio Interface

```
!pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.29.1-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<25.0,>=22.0 (from gradio)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-pack
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
```

```
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client==1.10.1 (from gradio)
  Downloading gradio_client-1.10.1-py3-none-any.whl.metadata (7.1 kB)
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-pac
Collecting ruff>=0.9.3 (from gradio)
  Downloading ruff-0.11.10-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.me
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/di
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (fr
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dis
```

```
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
)

interface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://d705d95e6f7291690a.gradio.live
```