```
#Load the  Data set
import numpy as np
import pandas as pd
from google.colab import files
uploaded = files.upload();# Upload your CSV file
df = pd.read_csv("ddataset.csv")  # Replace with your file name
```

Choose Files ddataset.csv

- **ddataset.csv**(text/csv) - 685 bytes, last modified: 5/6/2025 - 100% done

Saving ddataset.csv to ddataset.csv

```
#Data Exploration
df.head()

#to Shape Dataset
print("Shape:",df.shape)

#Column Names
print("Column:",df.columns.tolist())

#Data types and Null Values
df.info()

#Summary statistics for numeric values
df.describe()
```

```
Shape: (6, 6)
Column: ['User Query', ' "Intent" ', ' "Response"', 'Sentiment Score', 'Timestamp', 'Category']
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User Query       6 non-null      object
 1    "Intent"        6 non-null      object
 2    "Response"      6 non-null      object
 3   Sentiment Score  6 non-null      object
 4   Timestamp        6 non-null      object
 5   Category         1 non-null      object
dtypes: object(6)
memory usage: 420.0+ bytes
```

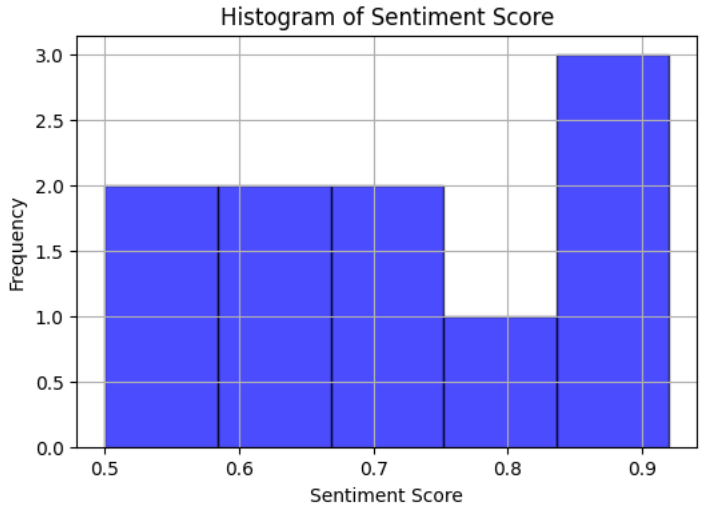
	User Query	"Intent"	"Response"	Sentiment Score	Timestamp	Category
count	6	6	6	6	6	1
unique	6	6	6	6	6	1
top	Where is my order?	"How do I return a product?"	"Do you have discounts?"	"I'm having trouble logging in."	"Can I speak to an agent?"	"I'll connect you to a support agent."

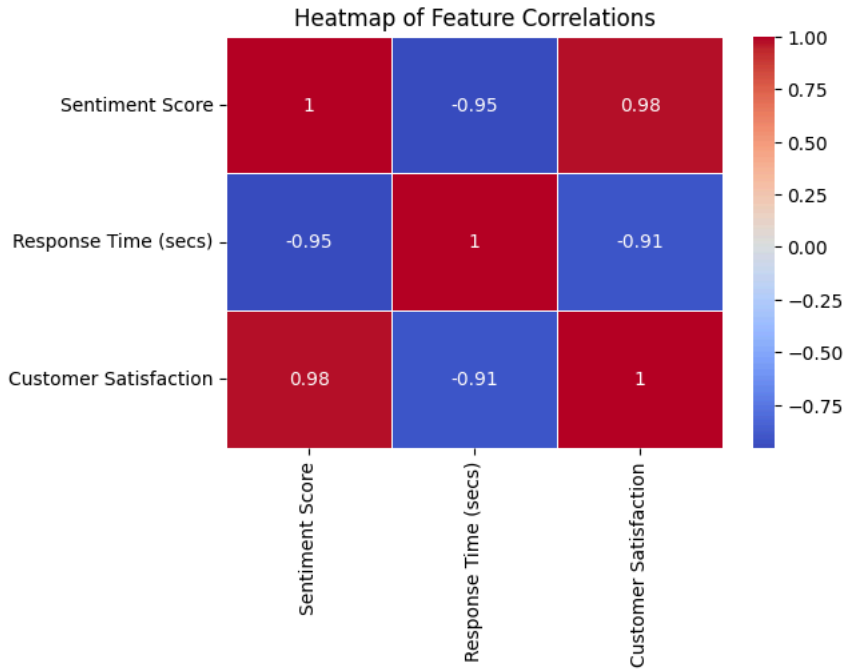
```
#Checking for missing values and Duplicates

#Check for missing values
print(df.isnull().sum())
```

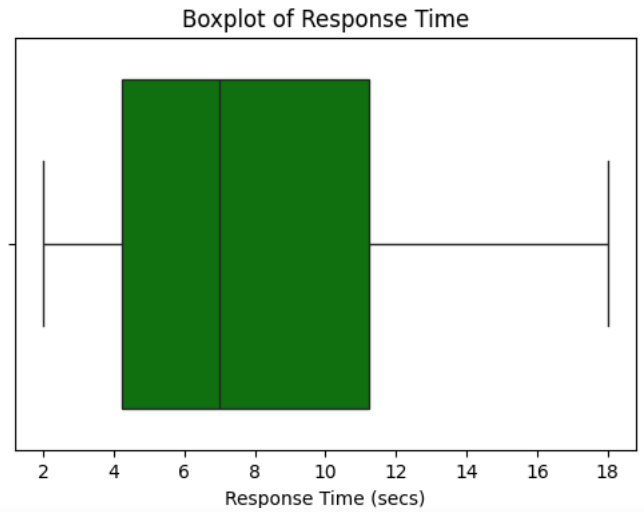
```
User Query         0
 "Intent"          0
 "Response"        0
Sentiment Score    0
Timestamp          0
Category           5
dtype: int64
```

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

data = {
    "Sentiment Score": [0.8, 0.7, 0.9, 0.6, 0.5, 0.85, 0.65, 0.92, 0.55, 0.75],
    "Response Time (secs)": [5, 8, 3, 12, 15, 4, 9, 2, 18, 6],
    "Customer Satisfaction": [4, 3, 5, 2, 1, 4, 3, 5, 2, 3]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# **Histogram for Sentiment Score**
plt.figure(figsize=(6,4))
```

```
plt.hist(df["Sentiment Score"], bins=5, color='blue', alpha=0.7, edgecolor='black')
plt.title("Histogram of Sentiment Score")
plt.xlabel("Sentiment Score")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()

# **Heatmap of Correlation Between Features**
plt.figure(figsize=(6,4))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Heatmap of Feature Correlations")
plt.show()

# **Boxplot for Response Time**
plt.figure(figsize=(6,4))
sns.boxplot(x=df["Response Time (secs)"], color="green")
plt.title("Boxplot of Response Time")
plt.xlabel("Response Time (secs)")
plt.show()
```








```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from textblob import TextBlob

target = data["Intent"]

# Display Features and Target
print("Extracted Features:\n", data[["User Query", "Sentiment Score"]])
print("\nTarget Labels:\n", target)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-7-3d9efe1a0aff> in <cell line: 0>()
      4 from textblob import TextBlob
      5
----> 6 target = data["Intent"]
      7
      8 # Display Features and Target

KeyError: 'Intent'
```

Next steps: Explain error

```
# 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)
print(df_encoded)
```

```
   Sentiment Score  Response Time (secs)  Customer Satisfaction
0             0.80                     5                      4
1             0.70                     8                      3
2             0.90                     3                      5
3             0.60                    12                      2
4             0.50                    15                      1
5             0.85                     4                      4
6             0.65                     9                      3
7             0.92                     2                      5
8             0.55                    18                      2
9             0.75                     6                      3
```

```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded.drop('Sentiment Score', axis=1))
y = df_encoded['Sentiment Score']
print(y)
```

```
0    0.80
1    0.70
2    0.90
3    0.60
4    0.50
5    0.85
6    0.65
7    0.92
8    0.55
9    0.75
Name: Sentiment Score, dtype: float64
```

```
#Train-Test Split
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
 # Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
#Model Building
 # Train model
model = LinearRegression()
model.fit(X_train, y_train)
 # Predict
y_pred = model.predict(X_test)
print(y_pred)
```

```
[0.44731915 0.71144681]
```

```
# Evaluate
print("MSE:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
MSE: 0.005337193300135808
R² Score: 0.05116563553141085
```

```
#Convert to DataFrame and Encode

# Convert to DataFrame
# Original: new_df = pd.DataFrame([data])
# Modified to handle 'Sentiment Score'
new_df = pd.DataFrame([{k: v[0] if isinstance(v, list) else v for k, v in data.items()}])


# Combine with original df to match columns
df_temp = pd.concat([df.drop('Sentiment Score', axis=1), new_df], ignore_index=True)

# One-hot encode
df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)

# Match the encoded feature order
df_temp_encoded = df_temp_encoded.reindex(columns=df_encoded.drop('Sentiment Score', axis=1).columns, fill_value=0)

# Scale (if you used scaling)
new_input_scaled = scaler.transform(df_temp_encoded.tail(1))
print(new_input_scaled)
```

```
[[-0.63295022  0.64051262]]
```

```
!pip install gradio
```

```
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-packages (from gradio) (0.15.3)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
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 (from gradio-client==1.10.0->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gra
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.4.26)
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) (
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.
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio)
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->gradi
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gra
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (1.5
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
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,>=
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,
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hu
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.2
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->
Downloading gradio-5.29.0-py3-none-any.whl (54.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.1/54.1 MB 8.4 MB/s eta 0:00:00
Downloading gradio_client-1.10.0-py3-none-any.whl (322 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 322.9/322.9 kB 24.7 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 6.7 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-0.11.8-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 109.6 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 5.9 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 4.8 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, starle
Successfully installed aiofiles-24.1.0 fastapi-0.115.12 ffmpy-0.5.0 gradio-5.29.0 gradio-client-1.10.0 groovy-0.1.2 pydub-0.25.1
```

```
#Create the gradio interface
# 🚀 Revolutionary AI Chatbot in Colab using Gradio
```

```
# STEP 1: Install dependencies
try:
    import gradio as gr
    import openai
except ImportError:
    import os
    os.system("pip install gradio openai --quiet")
    import gradio as gr
    import openai

# STEP 2: Set your OpenAI API key
openai.api_key = "YOUR_OPENAI_API_KEY"  # 🔐 Replace with your key

# STEP 3: Memory for chat history
chat_history = []

# STEP 4: Define function to call OpenAI GPT
def chat_with_gpt(message):
    chat_history.append({"role": "user", "content": message})
    response = openai.ChatCompletion.create(
        model="gpt-4",  # or "gpt-3.5-turbo"
        messages=[{"role": "system", "content": "You are an intelligent AI assistant."}] + chat_history
    )
    reply = response['choices'][0]['message']['content']
    chat_history.append({"role": "assistant", "content": reply})
    return reply

# STEP 5: Build the Gradio Interface
with gr.Blocks() as demo:
    gr.Markdown("<h1 style='text-align: center;'>🤖 Revolutionary AI Chatbot</h1>")

    chatbot = gr.Chatbot()
    msg = gr.Textbox(placeholder="Type your message here...", label="Your Message")
    clear = gr.Button("Clear")

    def respond(user_message, chat_history_ui):
        response = chat_with_gpt(user_message)
        chat_history_ui.append((user_message, response))
        return "", chat_history_ui

    def clear_history():
        global chat_history
```