```
# importing required libraries
import pandas as pdpd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

from xgboost import XGBRegressor
```

```
[ ] import random

    intents = {
        "greeting": ["Hello! How can I assist you today?"],
        "goodbye": ["Goodbye! Have a great day!"],
        "thanks": ["You're welcome! How else can I help?"],
        "product_query": ["Product X is our latest innovation with top features."],
        "order_status": ["Please provide your order ID to check the status."],
        "refund_policy": ["You can request a refund within 30 days of purchase."],
        "support_hours": ["Our support team is available 24/7 to assist you."]
    }

    def chatbot_response(user_input):
        user_input = user_input.lower()
        for intent, responses in intents.items():
            if intent in user_input:
                return random.choice(responses)
        return "Sorry, I didn't understand that. Could you please rephrase?"

    # Example
    print(chatbot_response("greeting"))
    print(chatbot_response("product_query"))
```

```
Hello! How can I assist you today?
Product X is our latest innovation with top features.
```

```
[ ] import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt

    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import mean_squared_error, r2_score

    from xgboost import XGBRegressor
```

```
[ ] print(os.listdir())
```

```
['.config', 'Meenakshisajan_dataset.csv', 'Niraliivaghani chatbot_ dataset. csv', 'Niraliivaghani chat
```

```
[ ] from google.colab import files
    uploaded = files.upload()  # This will prompt you to select the file
```

```
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 chatbot_data.csv to chatbot_data.csv
```

```
[ ] from google.colab import files
    uploaded = files.upload()  # Upload the CSV file here

    import os
    print("Files in current directory:", os.listdir())
```

```
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 chatbot_logs.csv to chatbot_logs.csv
Files in current directory: ['.config', 'chatbot_logs.csv', 'chatbot_data.csv', 'Niraliivaghani chatbo
```

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

    # Step 1: Upload your file (this will open a file picker)
    uploaded = files.upload()

    # Step 2: Load your CSV (use the exact filename you upload)
    filename = list(uploaded.keys())[0]
    df = pd.read_csv(filename)
```

```
# Step 3: Show first few rows
print("File loaded successfully! Here's a preview:")
print(df.head())
```

```
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 Niraliivaghani chatbot_ dataset. csv
File loaded successfully! Here's a preview:
Empty DataFrame
Columns: [{"nbformat":4, nbformat_minor:0,
Index: []

[0 rows x 202 columns]
```

```
import pandas as pd

# Example DataFrame
data = {'A': [1, 2, None], 'B': [4, None, 6]}
df = pd.DataFrame(data)

print("\nMissing Values in Each Column:")
print(df.isnull().sum())
```

```
Missing Values in Each Column:
A    1
B    1
dtype: int64
```

```
print("Dataset Shape:", df.shape)
print("Columns:", df.columns)
print(df.head())
```

```
Dataset Shape: (3, 2)
Columns: Index(['A', 'B'], dtype='object')
     A    B
0  1.0  4.0
1  2.0  NaN
2  NaN  6.0
```

```
df = df.fillna(method='ffill')  # forward fill
```

```
<ipython-input-29-cd21e285b752>:1: FutureWarning: DataFrame.fillna with 'method' is deprecated and wil
  df = df.fillna(method='ffill')  # forward fill
```

```
if 'pattern' in df.columns and 'response' in df.columns:
    df['pattern_length'] = df['pattern'].apply(len)
    df['response_length'] = df['response'].apply(len)
else:
    print("pattern or response column missing")
```

```
pattern or response column missing
```

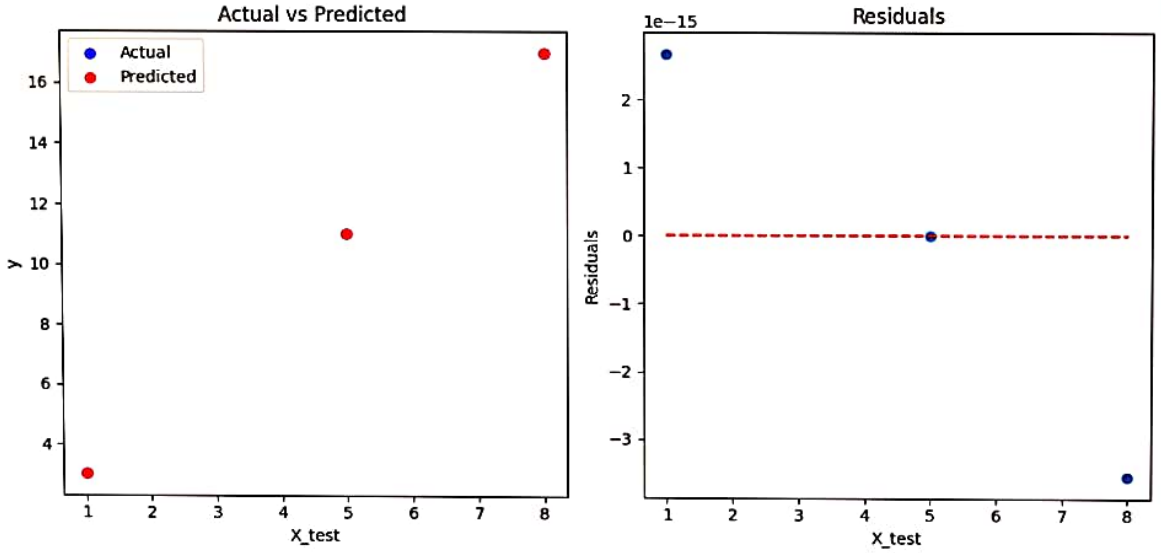
```
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import numpy as np

# Dummy data
X = np.arange(10).reshape(-1, 1)
y = np.array([1,3,5,7,9,11,13,15,17,19])

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Model create & train
model = LinearRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Calculate metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
[ ] print(f"Mean Squared Error: {mse:.2f}")
    print(f"R² Score: {r2:.2f}")
```

```
Mean Squared Error: 0.00
R² Score: 1.00
```

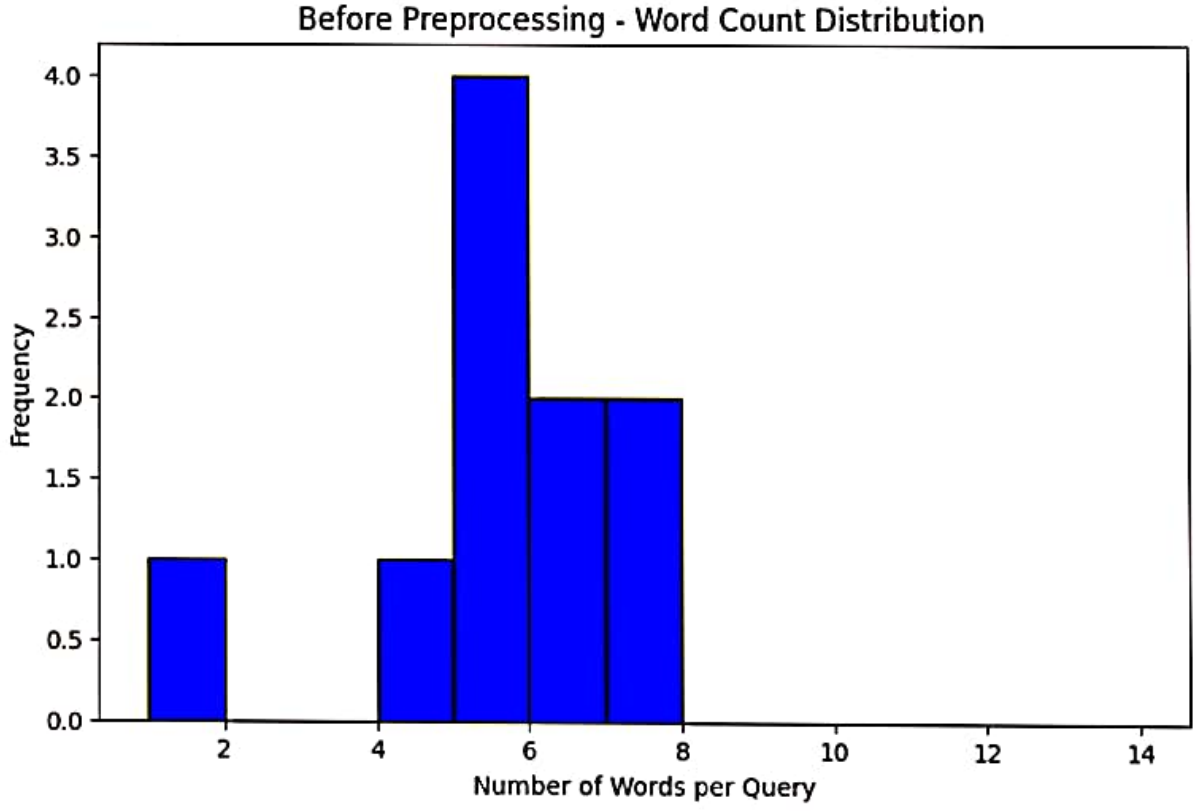
```
[ ] from sklearn.metrics import mean_squared_error, r2_score
    from sklearn.linear_model import LinearRegression
    from sklearn.model_selection import train_test_split
    import numpy as np
    import matplotlib.pyplot as plt

    # Dummy data
    X = np.arange(10).reshape(-1, 1)
    y = np.array([1,3,5,7,9,11,13,15,17,19])

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # Model create & train
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict
    y_pred = model.predict(X_test)

    # Calculate metrics
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"Mean Squared Error: {mse:.2f}")
    print(f"R² Score: {r2:.2f}")

    # Plot Actual vs Predicted
    plt.figure(figsize=(10,5))

    plt.subplot(1,2,1)
    plt.scatter(X_test, y_test, color='blue', label='Actual')
    plt.scatter(X_test, y_pred, color='red', label='Predicted')
    plt.title('Actual vs Predicted')
    plt.xlabel('X_test')
    plt.ylabel('y')
    plt.legend()

    # Plot residuals
    plt.subplot(1,2,2)
    residuals = y_test - y_pred
    plt.scatter(X_test, residuals)
    plt.hlines(y=0, xmin=min(X_test.flatten()), xmax=max(X_test.flatten()), colors='r', linestyles='dashe
    plt.title('Residuals')
    plt.xlabel('X_test')
    plt.ylabel('Residuals')

    plt.tight_layout()
    plt.show()
```

```
Mean Squared Error: 0.00
R² Score: 1.00
```




```
[ ] import pandas as pd
    import matplotlib.pyplot as plt

    # Dummy raw chatbot queries (unprocessed text)
```

```
[ ]
data = {
    'query': [
        "Hello, how can I reset my password?",
        "Pls help me with order status!!!",
        "I want to return a product",
        "How to update my profile?",
        "The app is crashing frequently...",
        "Need assistance with billing",
        "Where can I find the user manual?",
        "Can't login to my account",
        "What is your refund policy?",
        "Thx!"
    ]
}

df = pd.DataFrame(data)

# Calculate word counts (before preprocessing)
df['word_count'] = df['query'].apply(lambda x: len(x.split()))

# Plot word count distribution before preprocessing
plt.figure(figsize=(8,5))
plt.hist(df['word_count'], bins=range(1, 15), color='blue', edgecolor='black')
plt.title('Before Preprocessing - Word Count Distribution')
plt.xlabel('Number of Words per Query')
plt.ylabel('Frequency')
plt.show()
```



```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import string

    # Same raw data
    data = {
        'query': [
            "Hello, how can I reset my password?",
            "Pls help me with order status!!!",
            "I want to return a product",
            "How to update my profile?",
            "The app is crashing frequently...",
            "Need assistance with billing",
            "Where can I find the user manual?",
            "Can't login to my account",
            "What is your refund policy?",
            "Thx!"
        ]
    }

    df = pd.DataFrame(data)

    # Basic preprocessing function: lowercase, remove punctuation
    def preprocess_text(text):
        text = text.lower()
        text = text.translate(str.maketrans('', '', string.punctuation))
        return text

    # Apply preprocessing
    df['clean_query'] = df['query'].apply(preprocess_text)

    # Calculate word counts after preprocessing
    df['word_count_clean'] = df['clean_query'].apply(lambda x: len(x.split()))

    # Plot word count distribution after preprocessing
    plt.figure(figsize=(8,5))
    plt.hist(df['word_count_clean'], bins=range(1, 15), color='green', edgecolor='black')
    plt.title('After Preprocessing - Word Count Distribution')
    plt.xlabel('Number of Words per Cleaned Query')
    plt.ylabel('Frequency')
    plt.show()
```