CREATE A CHATBOT USING PYTHON

TEAM MEMBER

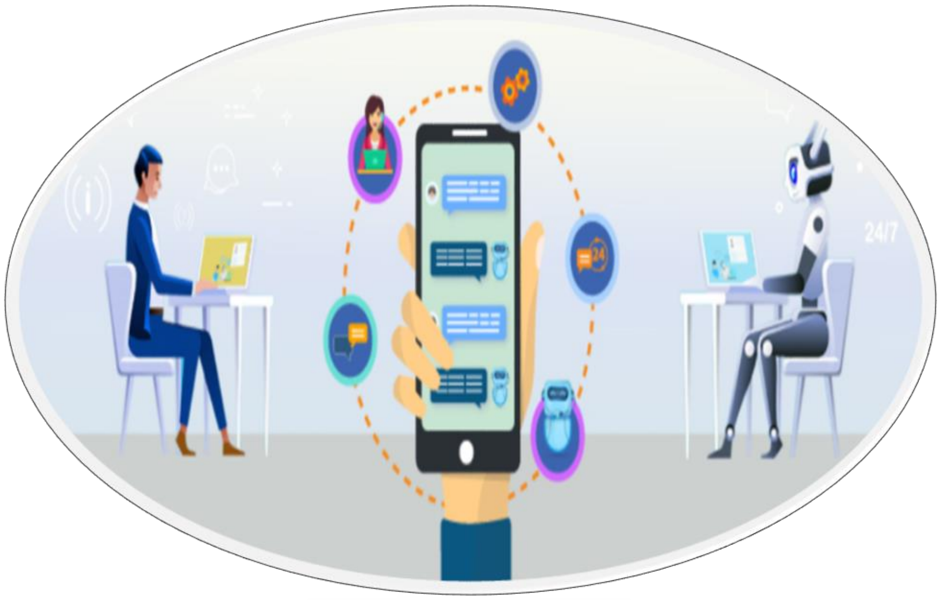
NAME :S.SHRUTHIKA

REG NO:820421205065

PHASE 3:

Development Part 1

SYNOPSIS



INTRODUCTION:

Start project by load and prepare your dataset, configure your development environment, and create basic user interactions. Install necessary libraries like transformers and Flask for easy GPT-3 integration and web application development. This sets the foundation for a complex, user-friendly chatbot interface. As you explore the dataset's complexities, you'll create a dynamic, intelligent chatbot that engages people effectively.

SETTING UP THE ENVIRONMENT AND INSTALLING REQUIRED LIBRARIES:

Starting the set up the environment by installing the necessary libraries and frameworks. Here I use virtual environments to manage dependencies. Here's an example of how to set up a virtual environment and install some essential packages:

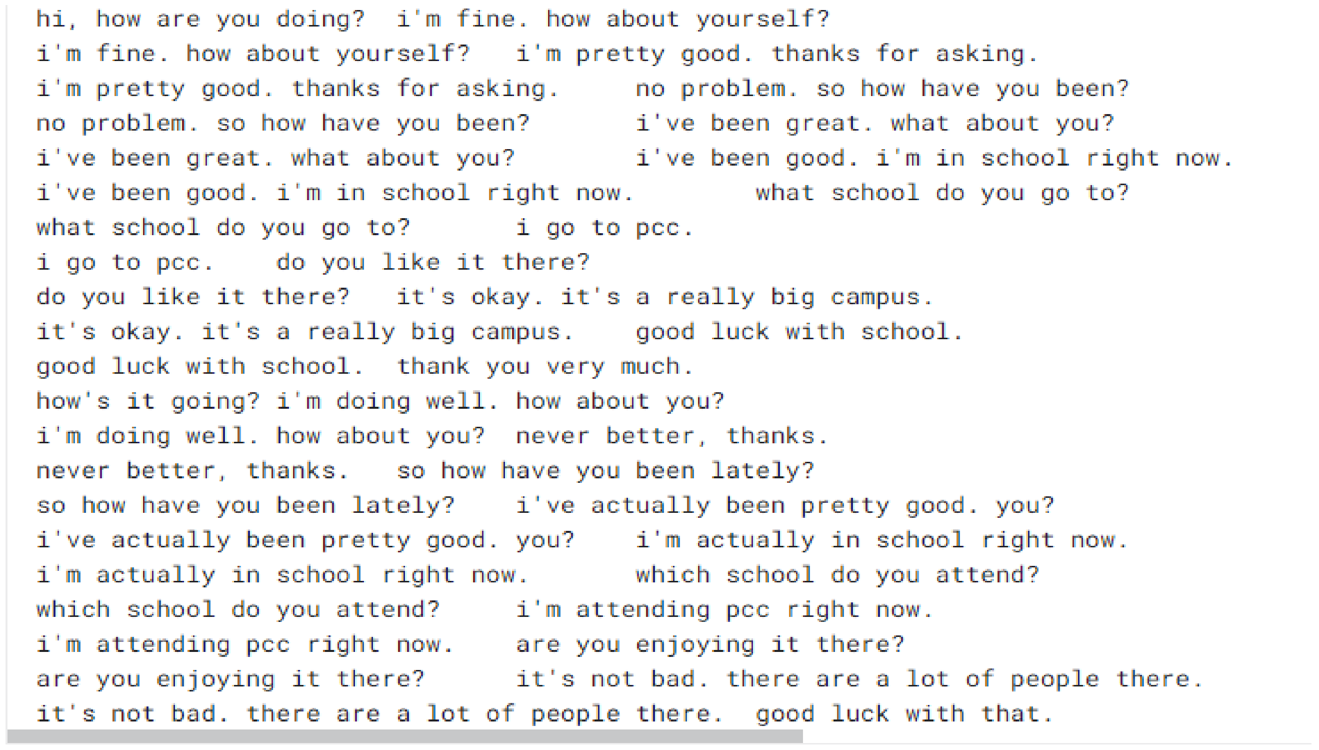
```
# Create a virtual environment
    python -m venv chatbot-env
# Activate the virtual environment
    source chatbot-env/bin/activate
# On Windows
    use "chatbot-env\Scripts\activate"
# Install required libraries
    pip install transformers flask nltk
```

LIBRARIES:

```
pip install pandas
pip install numpy
pip install io
pip install nltk
pip install scikit-learn
```

LOADING THE DATASET:

https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot

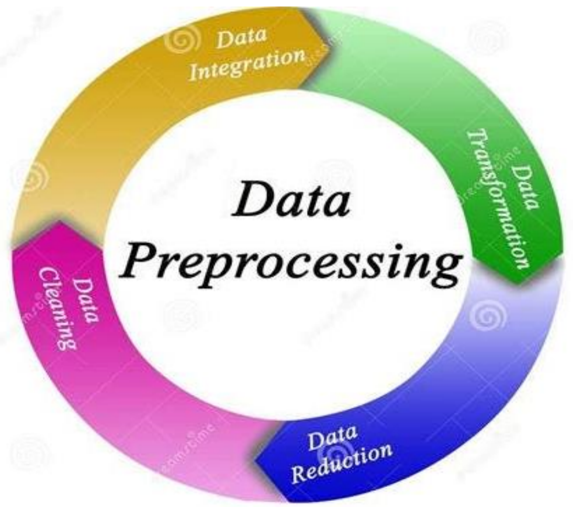


```
hi, how are you doing?  i'm fine. how about yourself?
i'm fine. how about yourself?   i'm pretty good. thanks for asking.
i'm pretty good. thanks for asking.     no problem. so how have you been?
no problem. so how have you been?       i've been great. what about you?
i've been great. what about you?        i've been good. i'm in school right now.
i've been good. i'm in school right now.        what school do you go to?
what school do you go to?       i go to pcc.
i go to pcc.    do you like it there?
do you like it there?   it's okay. it's a really big campus.
it's okay. it's a really big campus.    good luck with school.
good luck with school.  thank you very much.
how's it going? i'm doing well. how about you?
i'm doing well. how about you?  never better, thanks.
never better, thanks.   so how have you been lately?
so how have you been lately?    i've actually been pretty good. you?
i've actually been pretty good. you?    i'm actually in school right now.
i'm actually in school right now.       which school do you attend?
which school do you attend?     i'm attending pcc right now.
i'm attending pcc right now.    are you enjoying it there?
are you enjoying it there?      it's not bad. there are a lot of people there.
it's not bad. there are a lot of people there.  good luck with that.
```

DATA PREPROCESSING:

- **DEFINITION:**

  - ✓ Data preprocessing is the initial step in data analysis and machine learning.
  - ✓ Data Preprocessing is a process to convert the raw data into meaningful data using different techniques.



IMPORTANCE:

- Data Quality Improvement
- Handling Missing Data
- Scaling and Normalization
- Categorical Data Handling
- Time and Resource Efficiency

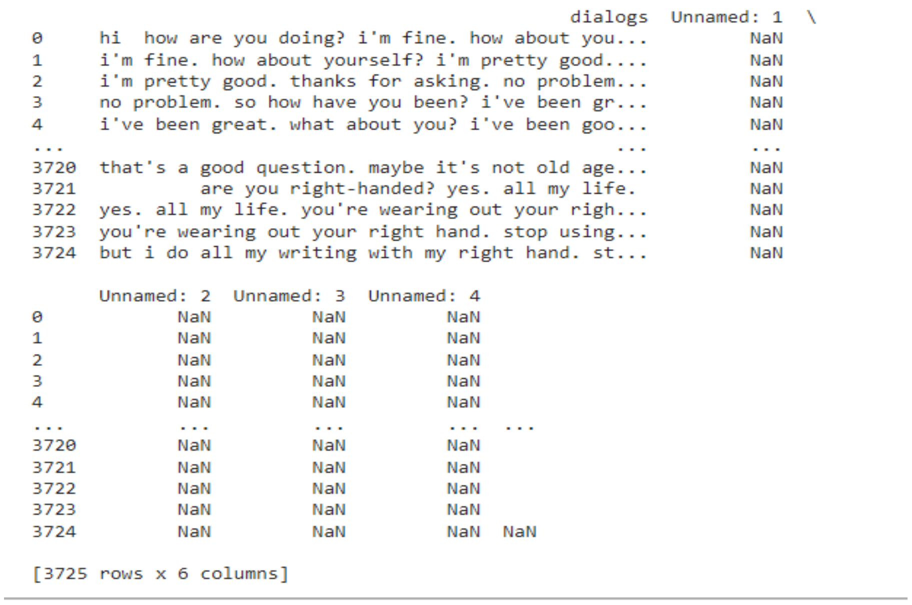
TECHNIQUES:

- Data Collection
- Data Cleaning
- Data Reduction
- Data Transformation
- Data Discrimination

DATA COLLECTION:

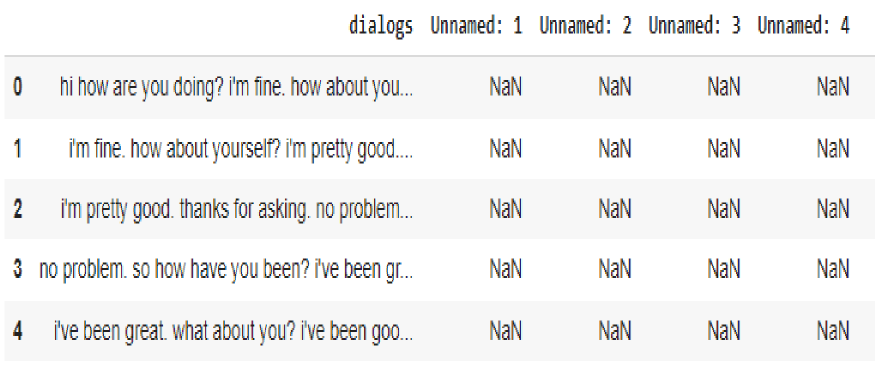
Data collection involves gathering and analyzing information from various sources to solve research issues, answer questions, evaluate results, and anticipate trends in various fields like business, and healthcare.

```
import pandas as pd
import io
df =
pd.read_csv(io.BytesIO(uploaded['dialog1.csv']))
print(df)
```



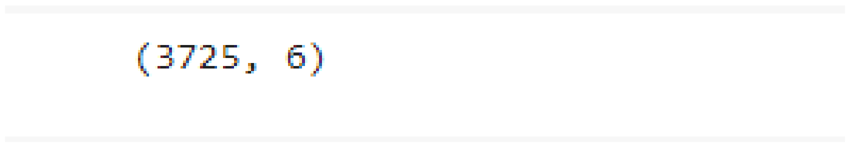
```
                                                  dialogs  Unnamed: 1  \
0     hi  how are you doing? i'm fine. how about you...         NaN
1     i'm fine. how about yourself? i'm pretty good....         NaN
2     i'm pretty good. thanks for asking. no problem...         NaN
3     no problem. so how have you been? i've been gr...         NaN
4     i've been great. what about you? i've been goo...         NaN
...                                                 ...         ...
3720  that's a good question. maybe it's not old age...         NaN
3721           are you right-handed? yes. all my life.         NaN
3722  yes. all my life. you're wearing out your righ...         NaN
3723  you're wearing out your right hand. stop using...         NaN
3724  but i do all my writing with my right hand. st...         NaN

      Unnamed: 2  Unnamed: 3  Unnamed: 4
0            NaN         NaN         NaN
1            NaN         NaN         NaN
2            NaN         NaN         NaN
3            NaN         NaN         NaN
4            NaN         NaN         NaN
...          ...         ...         ...  ...
3720         NaN         NaN         NaN
3721         NaN         NaN         NaN
3722         NaN         NaN         NaN
3723         NaN         NaN         NaN
3724         NaN         NaN         NaN  NaN

[3725 rows x 6 columns]
```

```
df.head()
```



	dialogs	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	hi how are you doing? i'm fine. how about you...	NaN	NaN	NaN	NaN
1	i'm fine. how about yourself? i'm pretty good....	NaN	NaN	NaN	NaN
2	i'm pretty good. thanks for asking. no problem...	NaN	NaN	NaN	NaN
3	no problem. so how have you been? i've been gr...	NaN	NaN	NaN	NaN
4	i've been great. what about you? i've been goo...	NaN	NaN	NaN	NaN

```
df.shape()
```



```
(3725, 6)
```

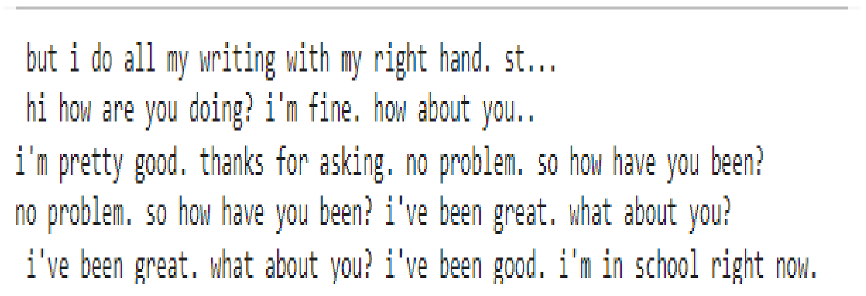
➢ DATA CLEANING:

➢ DEFINITION:

Data cleaning means fill in missing values ,smooth out noise while identifying outliers and correct inconsistencies data

LOWERCASING:

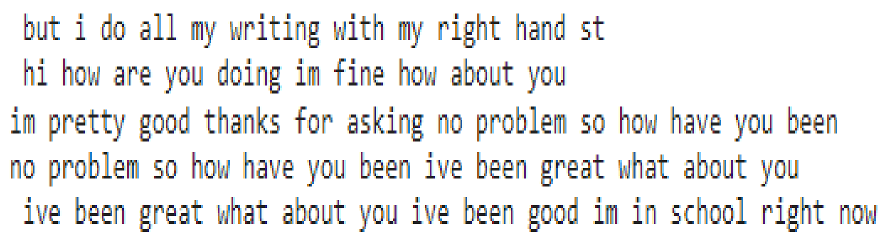
```
m_str=" But I do all my writing with my right hand. st...\n      Hi
how are you doing? i'm fine. How about you..\ni'm  pretty good.
Thanks for as69king. no Problem. so How have you been? \nno
problem. So how have you been? I've been great. what about you?
\n I've been great. what about you? I've been good. I'm in school
right now.  "

text=m_str.lower()
```



```
 but i do all my writing with my right hand. st...
 hi how are you doing? i'm fine. how about you..
i'm pretty good. thanks for asking. no problem. so how have you been?
no problem. so how have you been? i've been great. what about you?
 i've been great. what about you? i've been good. i'm in school right now.
```

REMOVING PUNCTUATION:

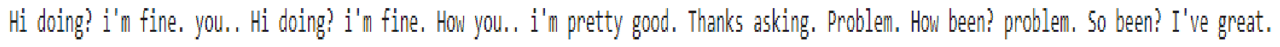
```
punc='''!()-[]{};:'"\,<>./?@#$%^&*_~'''
no_punct=""
for char in text:
  if(char not in punc):
     no_punct=no_punct+char
print(no_punct)
```



```
 but i do all my writing with my right hand st
 hi how are you doing im fine how about you
im pretty good thanks for asking no problem so how have you been
no problem so how have you been ive been great what about you
 ive been great what about you ive been good im in school right now
```

REMOVING STOPWARD:

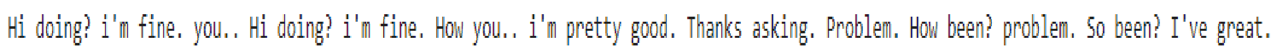
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop=stopwords.words("english")
text_cleaned=""
for word in text.split():
  if word in stop:
    pass
  else:
    text_cleaned +=" "
    text_cleaned +=word
text_cleaned
```



```
Hi doing? i'm fine. you.. Hi doing? i'm fine. How you.. i'm pretty good. Thanks asking. Problem. How been? problem. So been? I've great.
```

LEMMATIZATION:

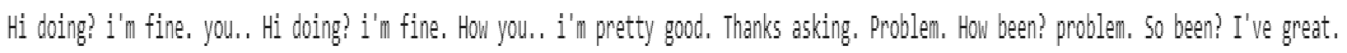
```
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
lem=WordNetLemmatizer()
text_cleaned1=" "
for word in text_cleaned.split():
  word=lem.lemmatize(word,pos="v")
  text_cleaned1 +=" "
 text_cleaned1 +=word
```



```
Hi doing? i'm fine. you.. Hi doing? i'm fine. How you.. i'm pretty good. Thanks asking. Problem. How been? problem. So been? I've great.
```

STEMMING:

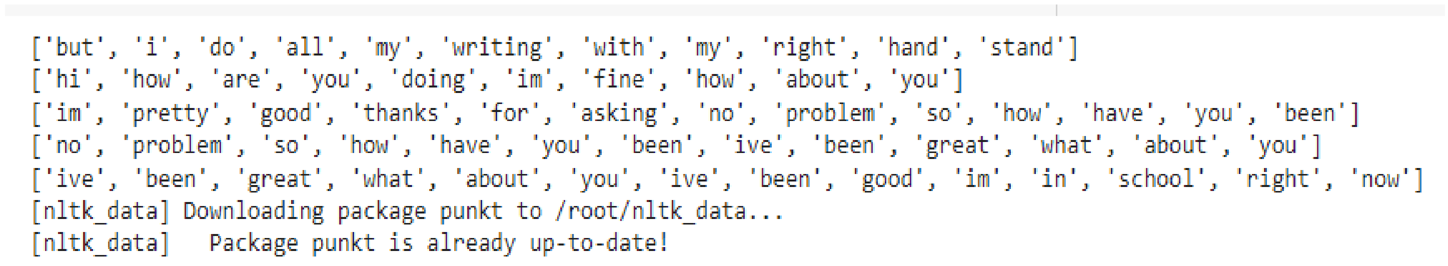
```
text=text_cleaned.strip()
print(text)
```



```
Hi doing? i'm fine. you.. Hi doing? i'm fine. How you.. i'm pretty good. Thanks asking. Problem. How been? problem. So been? I've great.
```

TOKENIZATION:

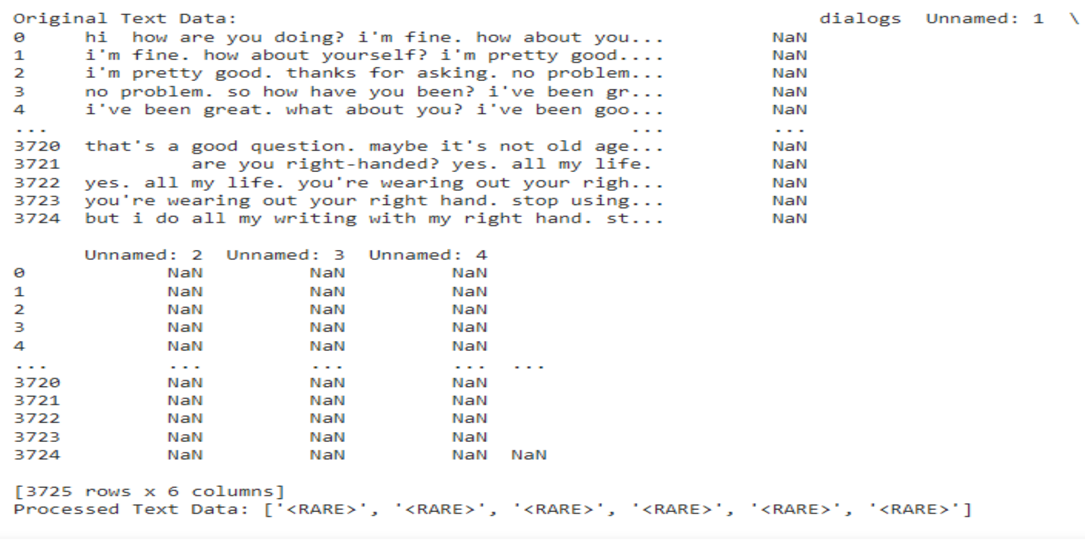
```
from nltk.tokenize import word_tokenize
nltk.download('punkt')
text_dataset = [" but i do all my writing with my right hand stand",
" hi how are you doing im fine how about you",
"im pretty good thanks for asking no problem so how have you been ",
"ive been great what about you ive been good im in school right now "]
tokenized_dataset = [word_tokenize(sentence)for sentence in
text_dataset]
for tokens in tokenized_dataset:
print(tokens)
```



```
['but', 'i', 'do', 'all', 'my', 'writing', 'with', 'my', 'right', 'hand', 'stand']
['hi', 'how', 'are', 'you', 'doing', 'im', 'fine', 'how', 'about', 'you']
['im', 'pretty', 'good', 'thanks', 'for', 'asking', 'no', 'problem', 'so', 'how', 'have', 'you', 'been']
['no', 'problem', 'so', 'how', 'have', 'you', 'been', 'ive', 'been', 'great', 'what', 'about', 'you']
['ive', 'been', 'great', 'what', 'about', 'you', 'ive', 'been', 'good', 'im', 'in', 'school', 'right', 'now']
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

HANDLING DUPLICATES:

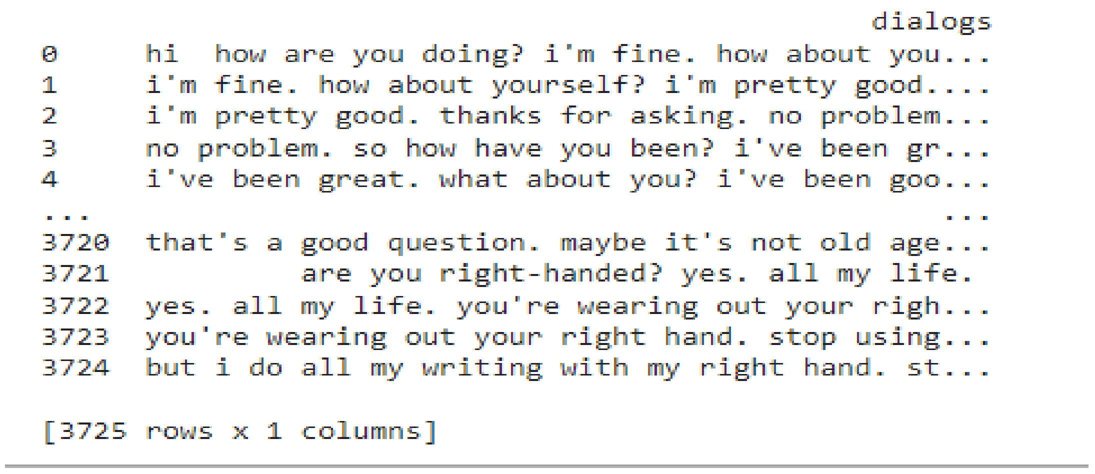
```
import pandas as pd
import io
def remove_duplicates_with_set(text_list):
    unique_text_set = set()
    result = []
    for text in text_list:
        if text not in unique_text_set:
            unique_text_set.add(text)
            result.append(text)
return result
def remove_duplicates_by_comparison(text_list):
    for text in text_list:
        if text not in result:
            result.append(text)
    return result
text_dataset
=pd.read_csv(io.BytesIO(uploaded['dialog1.csv']))
unique_texts_set =
remove_duplicates_with_set(text_dataset)
print("Method 1 - Using a Set:", unique_texts_set)
unique_texts_comparison =
```



```
Original Text Data:                                              dialogs  Unnamed: 1  \
0     hi  how are you doing? i'm fine. how about you...          NaN
1     i'm fine. how about yourself? i'm pretty good....          NaN
2     i'm pretty good. thanks for asking. no problem...          NaN
3     no problem. so how have you been? i've been gr...          NaN
4     i've been great. what about you? i've been goo...          NaN
...                                                 ...          ...
3720  that's a good question. maybe it's not old age...          NaN
3721           are you right-handed? yes. all my life.           NaN
3722  yes. all my life. you're wearing out your righ...          NaN
3723  you're wearing out your right hand. stop using...          NaN
3724  but i do all my writing with my right hand. st...          NaN

      Unnamed: 2  Unnamed: 3  Unnamed: 4
0            NaN         NaN         NaN
1            NaN         NaN         NaN
2            NaN         NaN         NaN
3            NaN         NaN         NaN
4            NaN         NaN         NaN
...          ...         ...         ...  ...
3720         NaN         NaN         NaN
3721         NaN         NaN         NaN
3722         NaN         NaN         NaN
3723         NaN         NaN         NaN
3724         NaN         NaN         NaN  NaN

[3725 rows x 6 columns]
Processed Text Data: ['<RARE>', '<RARE>', '<RARE>', '<RARE>', '<RARE>', '<RARE>']
```

IGNORE MISSING DATA:

```
import pandas as pd
import numpy as np
import io
df= pd.read_csv(io.BytesIO(uploaded['dialog1.csv']))
df.isnull()
text_data=df.dropna(axis = 1)
print(text_data)
```



```
                                                dialogs
0     hi  how are you doing? i'm fine. how about you...
1     i'm fine. how about yourself? i'm pretty good....
2     i'm pretty good. thanks for asking. no problem...
3     no problem. so how have you been? i've been gr...
4     i've been great. what about you? i've been goo...
...                                                 ...
3720  that's a good question. maybe it's not old age...
3721            are you right-handed? yes. all my life.
3722  yes. all my life. you're wearing out your righ...
3723  you're wearing out your right hand. stop using...
3724  but i do all my writing with my right hand. st...

[3725 rows x 1 columns]
```

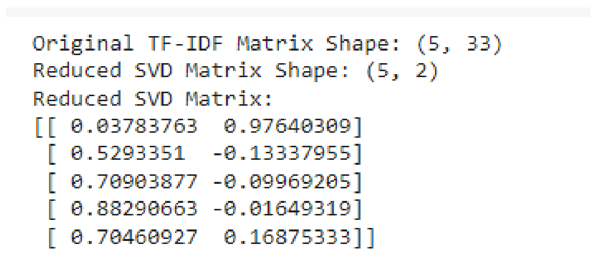
➢ DATA REDUCTION:

DEFINTION:

Data reduction is aiming to reduce complexity while retaining essential information. This reduces computational resources, improves training efficiency, and minimizes noise

DIMENSIONAL REDUCTION:

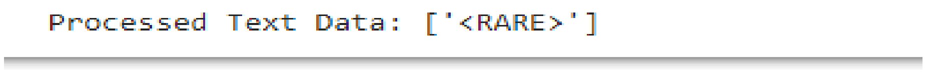
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
documents = [" but i do all my writing with my right hand stand"," hi
how are you doing im fine how about you", "im pretty good thanks for
asking no problem so how have you been ", "no problem so how have
you been ive been great what about you" , "ive been great what about
you ive been good im in school right now "]
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
n_components = 2
svd = TruncatedSVD(n_components=n_components)
svd_matrix = svd.fit_transform(tfidf_matrix)
print("Original TF-IDF Matrix Shape:", tfidf_matrix.shape)
print("Reduced SVD Matrix Shape:", svd_matrix.shape)
```



```
Original TF-IDF Matrix Shape: (5, 33)
Reduced SVD Matrix Shape: (5, 2)
Reduced SVD Matrix:
[[ 0.03783763  0.97640309]
 [ 0.5293351  -0.13337955]
 [ 0.70903877 -0.09969205]
 [ 0.88290663 -0.01649319]
 [ 0.70460927  0.16875333]]
```

HANDLE RARE WORD:

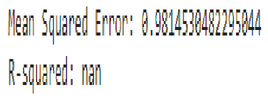
```
import pandas as pd
import io
from collections import Counter
def handle_rare_words(text_data, threshold=2,
rare_token="<RARE>"):
    word_counts = Counter(text_data)
    rare_words = [word for word, count in
word_counts.items() if count <= threshold]
    processed_text = [rare_token if word in rare_words else
word for word in text_data]
return processed_text
threshold = 2
processed_text = handle_rare_words(text_data, threshold)
```



```
Processed Text Data: ['<RARE>']
```

REGRESSION:

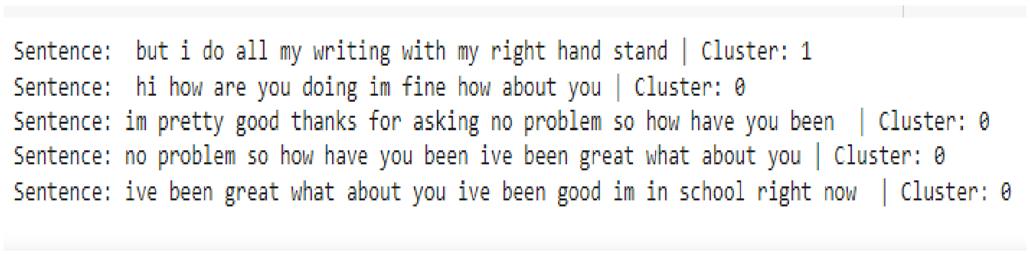
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
data = [" but i do all my writing with my right hand stand", " hi how are you
doing im fine how about you", "im pretty good thanks for asking no problem
so how have you been ", "no problem so how have you been ive been great
what about you" , "ive been great what about you ive been good im in
school right now "]
target = [3, 4, 1, 5,2]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data)
X_train, X_test, y_train, y_test = train_test_split(X, target, test_size=0.2,
random_state=42)
regressor = LinearRegression()
regressor.fit(X_train, y_train)
predictions = regressor.predict(X_test)
mse = mean_squared_error(y_test, predictions)
```



```
Mean Squared Error: 0.9814530482295044
R-squared: nan
```

CLUSTER:

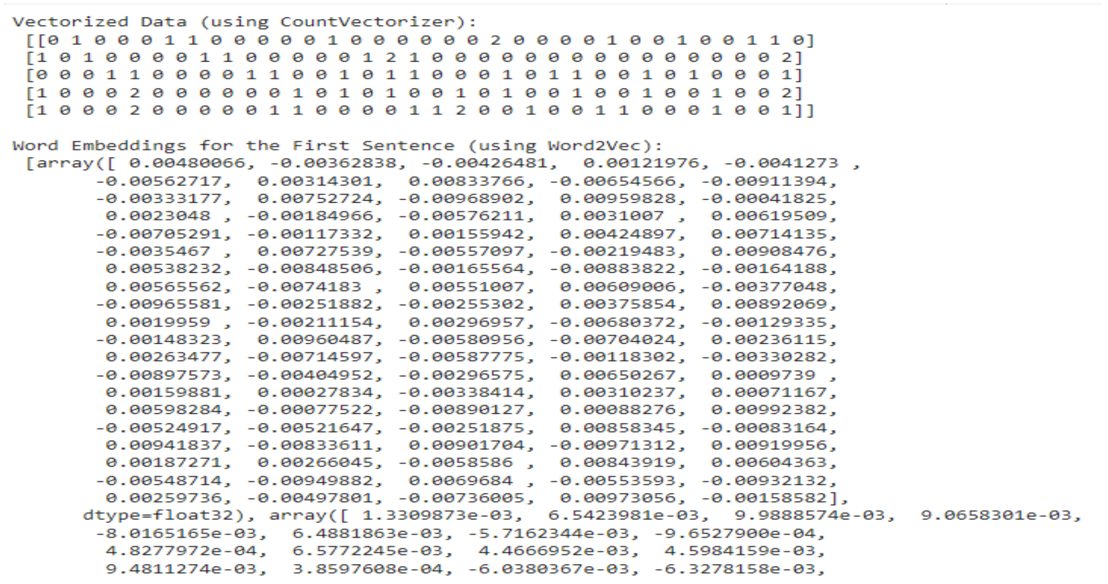
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score
import pandas as pd
data = [" but i do all my writing with my right hand stand", " hi
how are you doing im fine how about you","im pretty good thanks
for asking no problem so how have you been ","no problem so how
have you been ive been great what about you" , "ive been great
what about you ive been good im in school right now "]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data)
num_clusters = 2
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(X)
cluster_labels = kmeans.labels_
for i, sentence in enumerate(data):
   print(f"Sentence: {sentence} | Cluster: {cluster_labels[i]}")
```



```
Sentence:  but i do all my writing with my right hand stand | Cluster: 1
Sentence:  hi how are you doing im fine how about you | Cluster: 0
Sentence: im pretty good thanks for asking no problem so how have you been  | Cluster: 0
Sentence: no problem so how have you been ive been great what about you | Cluster: 0
Sentence: ive been great what about you ive been good im in school right now  | Cluster: 0
```

VECTORIZATION:

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')
data = [" but i do all my writing with my right hand stand","
hi how are you doing im fine how about you","im pretty good
thanks for asking no problem so how have you been ","no
problem so how have you been ive been great what about
you" , "ive been great what about you ive been good im in
school right now "]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(data)
vectorized_data = X.toarray()
```



```
Vectorized Data (using CountVectorizer):
 [[0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 2 0 0 0 0 1 0 0 1 0 0 1 1 0]
 [1 0 1 0 0 0 0 1 1 0 0 0 0 0 1 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]
 [0 0 0 1 1 0 0 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1 0 1 0 0 0 1]
 [1 0 0 0 2 0 0 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 2]
 [1 0 0 0 2 0 0 0 0 0 1 1 0 0 0 0 1 1 2 0 0 1 0 0 1 1 0 0 0 1 0 0 1]]

Word Embeddings for the First Sentence (using Word2Vec):
 [array([ 0.00480066, -0.00362838, -0.00426481,  0.00121976, -0.0041273 ,
       -0.00562717,  0.00314301,  0.00833766, -0.00654566, -0.00911394,
       -0.00333177,  0.00752724, -0.00968902,  0.00959828, -0.00041825,
        0.0023048 , -0.00184966, -0.00576211,  0.0031007 ,  0.00619509,
       -0.00705291, -0.00117332,  0.00155942,  0.00424897,  0.00714135,
       -0.0035467 ,  0.00727539, -0.00557097, -0.00219483,  0.00908476,
        0.00538232, -0.00848506, -0.00165564, -0.00883822, -0.00164188,
        0.00565562, -0.0074183 ,  0.00551007,  0.00609006, -0.00377048,
       -0.00965581, -0.00251882, -0.00255302,  0.00375854,  0.00892069,
        0.0019959 , -0.00211154,  0.00296957, -0.00680372, -0.00129335,
       -0.00148323,  0.00960487, -0.00580956, -0.00704024,  0.00236115,
        0.00263477, -0.00714597, -0.00587775, -0.00118302, -0.00330282,
       -0.00897573, -0.00404952, -0.00296575,  0.00650267,  0.0009739 ,
        0.00159881,  0.00027834, -0.00338414,  0.00310237,  0.00071167,
        0.00598284, -0.00077522, -0.00890127,  0.00088276,  0.00992382,
       -0.00524917, -0.00521647, -0.00251875,  0.00858345, -0.00083164,
        0.00941837, -0.00833611,  0.00901704, -0.00971312,  0.00919956,
        0.00187271,  0.00266045, -0.0058586 ,  0.00843919,  0.00604363,
       -0.00548714, -0.00949882,  0.0069684 , -0.00553593, -0.00932132,
        0.00259736, -0.00497801, -0.00736005,  0.00973056, -0.00158582],
      dtype=float32), array([ 1.3309873e-03,  6.5423981e-03,  9.9888574e-03,  9.0658301e-03,
       -8.0165165e-03,  6.4881863e-03, -5.7162344e-03, -9.6527900e-04,
        4.8277972e-04,  6.5772245e-03,  4.4666952e-03,  4.5984159e-03,
        9.4811274e-03,  3.8597608e-04, -6.0380367e-03, -6.3278158e-03,
```

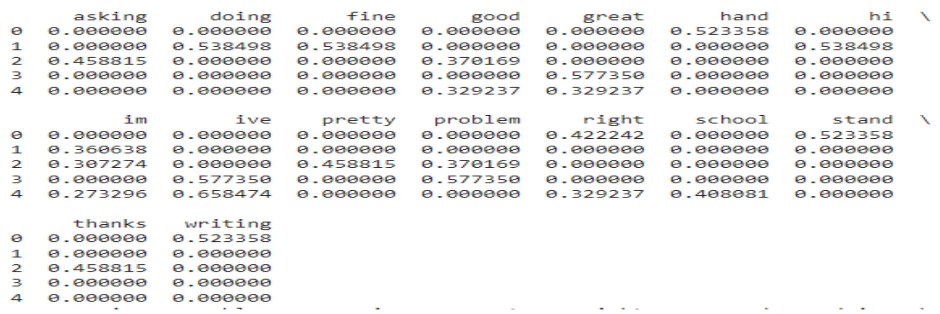
➢ DATA TRANSFORMATION:

DEFINITION:

Data transformation is a crucial step in preprocessing a chatbot dataset, converting, modifying, or structuring data for analysis, model training, and interaction with the chatbot.

ATTRIBUTE SELECTION:

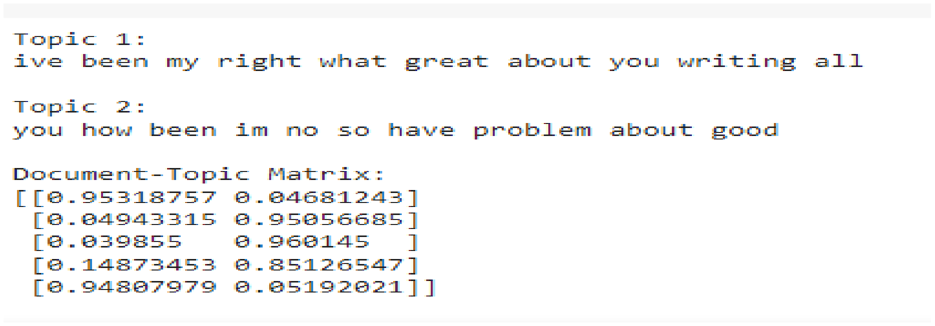
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
data = [
   " but i do all my writing with my right hand stand"," hi how are
you doing im fine how about you","im pretty good thanks f or
asking no problem so how have you been ","no problem so how
have you been ive been great what about you" ,"ive been great
what about you ive been good im in school right now "
]
df = pd.DataFrame(data, columns=["text"])
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf_vectorizer.fit_transform(df['text'])
feature_names = tfidf_vectorizer.get_feature_names_out()
tfidf_df = pd.DataFrame(data=tfidf_matrix.toarray(),
columns=feature_names)
print(tfidf_df)
top_n_features = 10
selected_features = tfidf_df.sum().nlargest(top_n_features).index
selected_features_df = tfidf_df[selected_features]
print(selected_features_df)
```



```
      asking     doing      fine      good     great      hand        hi  \
0   0.000000  0.000000  0.000000  0.000000  0.000000  0.523358  0.000000
1   0.000000  0.538498  0.538498  0.000000  0.000000  0.000000  0.538498
2   0.458815  0.000000  0.000000  0.370169  0.000000  0.000000  0.000000
3   0.000000  0.000000  0.000000  0.000000  0.577350  0.000000  0.000000
4   0.000000  0.000000  0.000000  0.329237  0.329237  0.000000  0.000000

          im       ive    pretty   problem     right    school     stand  \
0   0.000000  0.000000  0.000000  0.000000  0.422242  0.000000  0.523358
1   0.360638  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2   0.307274  0.000000  0.458815  0.370169  0.000000  0.000000  0.000000
3   0.000000  0.577350  0.000000  0.577350  0.000000  0.000000  0.000000
4   0.273296  0.658474  0.000000  0.000000  0.329237  0.408081  0.000000

      thanks   writing
0   0.000000  0.523358
1   0.000000  0.000000
2   0.458815  0.000000
3   0.000000  0.000000
4   0.000000  0.000000
```

CONCEPT HIERARCHY GENERATION:

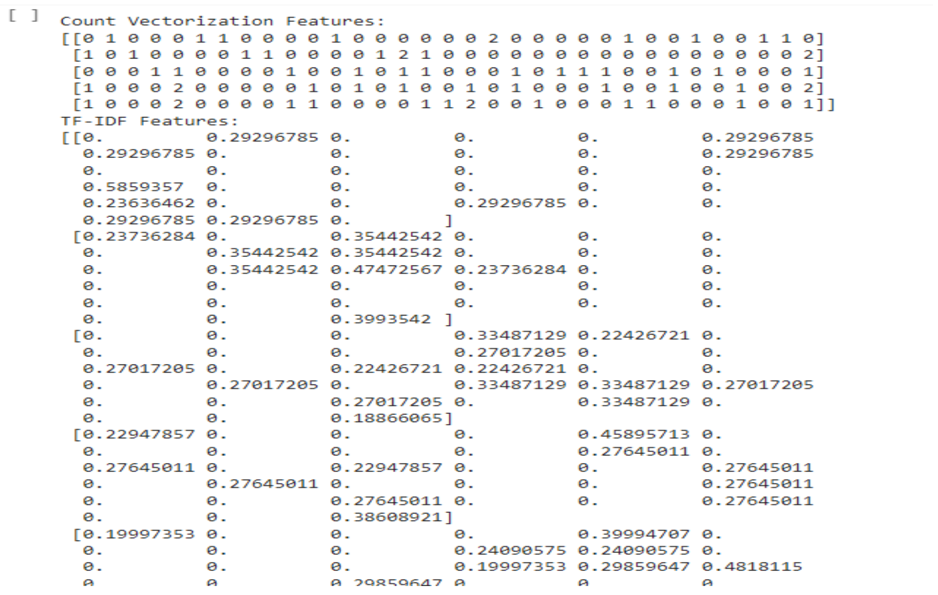
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
text_data =  [
   " but i do all my writing with my right hand stand"," hi how are
you doing im fine how about you","im pretty good thanks f or
asking no problem so how have you been ","no problem so how
have you been ive been great what about you" ,"ive been great
what about you ive been good im in school right now "
]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(text_data)
num_topics = 2
lda = LatentDirichletAllocation(n_components=num_topics,
random_state=42)
lda.fit(X)
feature_names = vectorizer.get_feature_names_out()
for topic_idx, topic in enumerate(lda.components_):
    print(f"Topic {topic_idx + 1}:")
    print(" ".join([feature_names[i] for i in topic.argsort()[:-10 - 1:-
1]]))
    print()
doc_topic_matrix = lda.transform(X)
print("Document-Topic Matrix:")
print(doc_topic_matrix)
```



```
Topic 1:
ive been my right what great about you writing all

Topic 2:
you how been im no so have problem about good

Document-Topic Matrix:
[[0.95318757 0.04681243]
 [0.04943315 0.95056685]
 [0.039855   0.960145  ]
 [0.14873453 0.85126547]
 [0.94807979 0.05192021]]
```

FEATURE ENGINEERING:

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')
text_data =  [
   " but i do all my writing with my right hand stand", " hi how
are you doing im fine how about you","im pretty good thanks f
or asking no problem so how have you been ", "no problem so
how have you been ive been great what about you" , "ive been
great what about you ive been good im in school right now "]
labels = [1, 2, 3, 1,2]
tokenized_text = [word_tokenize(text) for text in text_data]
count_vectorizer = CountVectorizer()
count_features = count_vectorizer.fit_transform(['
'.join(tokens) for tokens in tokenized_text])
tfidf_vectorizer = TfidfVectorizer()
tfidf_features = tfidf_vectorizer.fit_transform([' '.join(tokens)
for tokens in tokenized_text])
word2vec_model = Word2Vec(sentences=tokenized_text,
vector_size=100, window=5, min_count=1, sg=0)
word_embeddings =
np.array([np.mean([word2vec_model.wv[word] for word in
tokens], axis=0) for tokens in tokenized_text])
X_train, X_test, y_train, y_test =
train_test_split(word_embeddings, labels, test_size=0.2,
random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("Count Vectorization Features:")
print(count_features.toarray())
print("TF-IDF Features:")
print(tfidf_features.toarray())
print("Word Embeddings:")
print(word_embeddings)
print("X_train_scaled:")
print(X_train_scaled)
print("X_test_scaled:")
print(X_test_scaled)
```



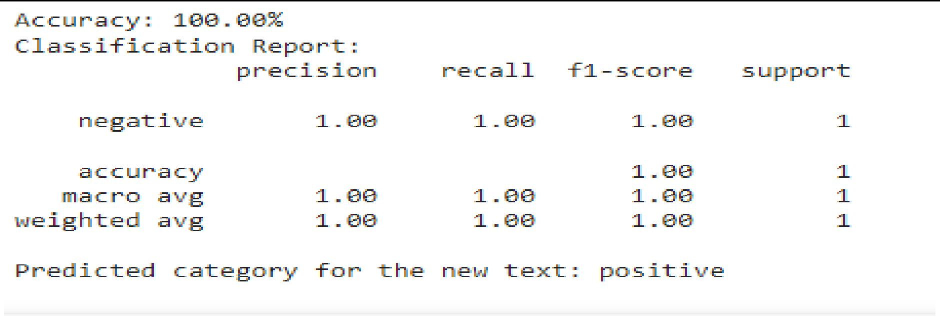
```
[ ]  Count Vectorization Features:
     [[0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 2 0 0 0 0 0 1 0 0 1 0 0 1 1 0]
      [1 0 1 0 0 0 0 1 1 0 0 0 0 1 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]
      [0 0 0 1 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 0 1]
      [1 0 0 0 2 0 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 2]
      [1 0 0 0 2 0 0 0 0 1 1 0 0 0 0 1 1 2 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1]]
     TF-IDF Features:
     [[0.         0.29296785 0.         0.         0.         0.29296785
       0.29296785 0.         0.         0.         0.         0.29296785
       0.         0.         0.         0.         0.         0.
       0.5859357  0.         0.         0.         0.         0.
       0.23636462 0.         0.         0.29296785 0.         0.
       0.29296785 0.29296785 0.        ]
      [0.23736284 0.         0.35442542 0.         0.         0.
       0.         0.35442542 0.35442542 0.         0.         0.
       0.         0.35442542 0.47472567 0.23736284 0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.3993542 ]
      [0.         0.         0.         0.33487129 0.22426721 0.
       0.         0.         0.         0.27017205 0.         0.
       0.27017205 0.         0.22426721 0.22426721 0.         0.
       0.         0.27017205 0.         0.33487129 0.33487129 0.27017205
       0.         0.         0.27017205 0.         0.33487129 0.
       0.         0.         0.18866065]
      [0.22947857 0.         0.         0.         0.45895713 0.
       0.         0.         0.         0.         0.27645011 0.
       0.27645011 0.         0.22947857 0.         0.         0.27645011
       0.         0.27645011 0.         0.         0.         0.27645011
       0.         0.         0.27645011 0.         0.         0.27645011
       0.         0.         0.38608921]
      [0.19997353 0.         0.         0.         0.39994707 0.
       0.         0.         0.         0.24090575 0.24090575 0.
       0.         0.         0.         0.19997353 0.29859647 0.4818115
       0.         0.         0.29859647 0.         0.         0.
```

➢ DATA DISCRETIZATION:

Data discretization is the process of dividing continuous or numerical data into distinct intervals, bins, or categories. This simplifies analysis, interpretation, and use in chatbot interactions.

TEXT CATEGORIZATION:

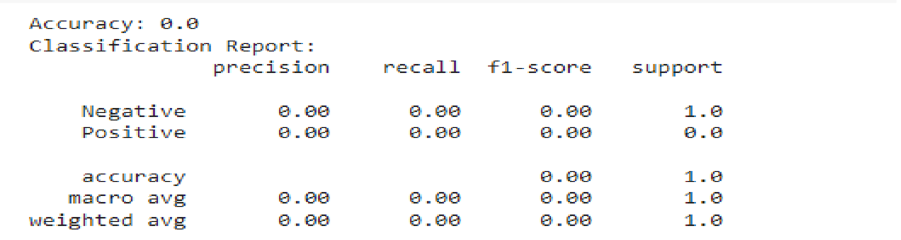
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
text_data = [
   " but i do all my writing with my right hand stand"," hi how are you doing
im fine how about you", "im pretty good thanks f or asking no problem so
how have you been ", "no problem so how have you been ive been great
what about you" , "]
labels = ['positive', 'negative', 'positive', 'negative', 'positive']
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(text_data)
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2,
random_state=42)
classifier = MultinomialNB()
classifier.fit(X_train, y_train)
```



```
Accuracy: 100.00%
Classification Report:
              precision    recall  f1-score   support

    negative       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1

Predicted category for the new text: positive
```

DECISION TREE:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
texts =[
   " but i do all my writing with my right hand stand"," hi how are
you doing im fine how about you","i was sick. how were you sick?
","no problem so how have you been ive been great what about
you" , "my bad  i had chores to do. that's all right."]
labels = ["Positive", "Negative", "Neutral", "Positive", "Negative"]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(texts)
X_train, X_test, y_train, y_test = train_test_split(X, labels,
test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
report = classification_report(y_test, predictions)
print("Accuracy:", accuracy)
print("Classification Report:")
print(report)
```

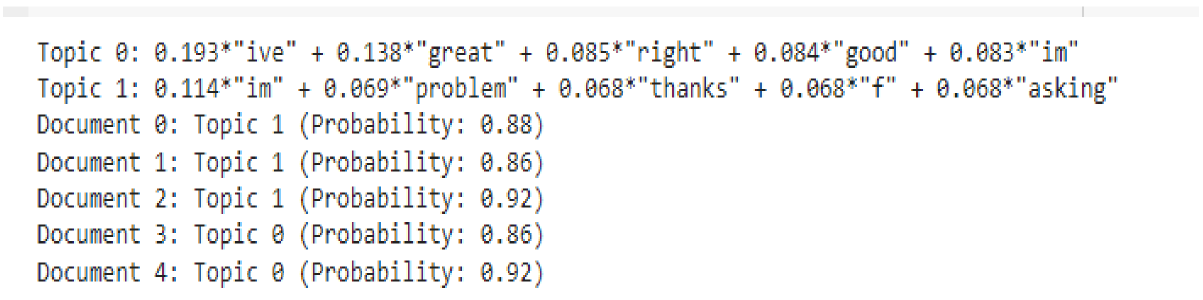


```
Accuracy: 0.0
Classification Report:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00       1.0
    Positive       0.00      0.00      0.00       0.0

    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0
```

TOPIC MODELING:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from gensim import corpora, models
import gensim
import string
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
text_data = [
   " but i do all my writing with my right hand stand"," hi how are
you doing im fine how about you","im pretty good thanks f or asking
no problem so how have you been ", "no problem so how have you
been ive been great what about you" , "ive been great what about
you ive been good im in school right now "]
def preprocess_text(text):
    tokens = word_tokenize(text.lower())
    tokens = [word for word in tokens if word.isalpha() and word not
in stopwords.words('english')]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return tokens
processed_text_data = [preprocess_text(doc) for doc in text_data]
dictionary = corpora.Dictionary(processed_text_data)
corpus = [dictionary.doc2bow(doc) for doc in processed_text_data]
num_topics = 2
lda_model = gensim.models.LdaModel(corpus,
num_topics=num_topics, id2word=dictionary, passes=15)
topics = lda_model.print_topics(num_words=5)
for topic in topics:
    print("Topic {}: {}".format(topic[0], topic[1]))
for i, doc in enumerate(corpus):
    topic = lda_model.get_document_topics(doc)
    dominant_topic = sorted(topic, key=lambda x: x[1],
reverse=True)[0]
    print("Document {}: Topic {} (Probability: {:.2f})".format(i,
dominant_topic[0], dominant_topic[1]))
```

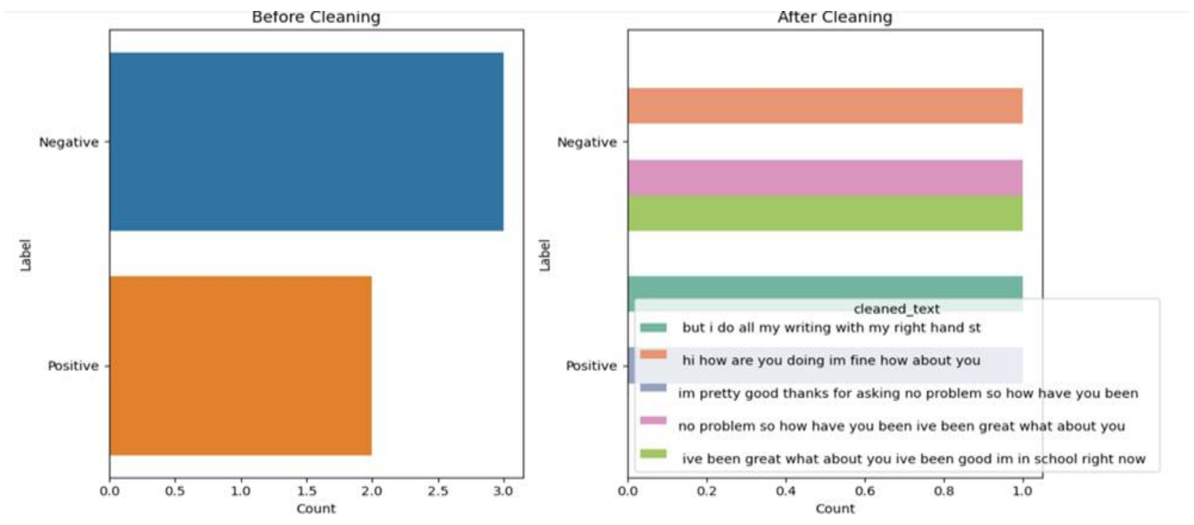


```
Topic 0: 0.193*"ive" + 0.138*"great" + 0.085*"right" + 0.084*"good" + 0.083*"im"
Topic 1: 0.114*"im" + 0.069*"problem" + 0.068*"thanks" + 0.068*"f" + 0.068*"asking"
Document 0: Topic 1 (Probability: 0.88)
Document 1: Topic 1 (Probability: 0.86)
Document 2: Topic 1 (Probability: 0.92)
Document 3: Topic 0 (Probability: 0.86)
Document 4: Topic 0 (Probability: 0.92)
```

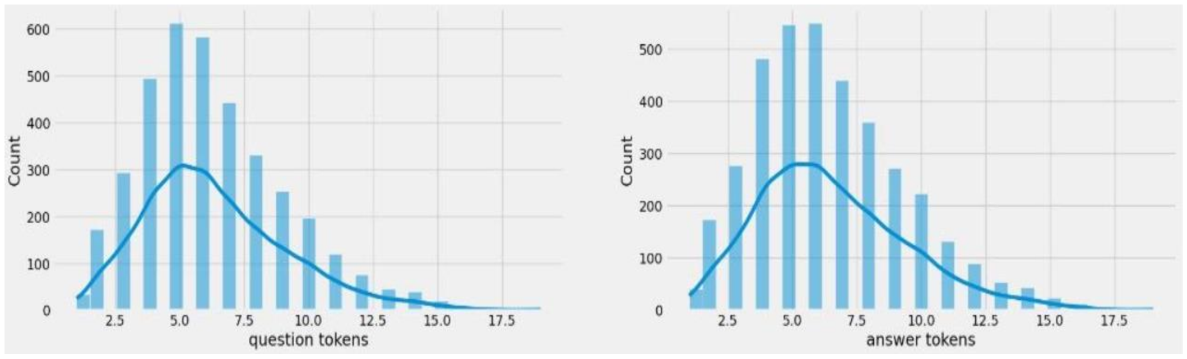
DATA VISUALIZATION:

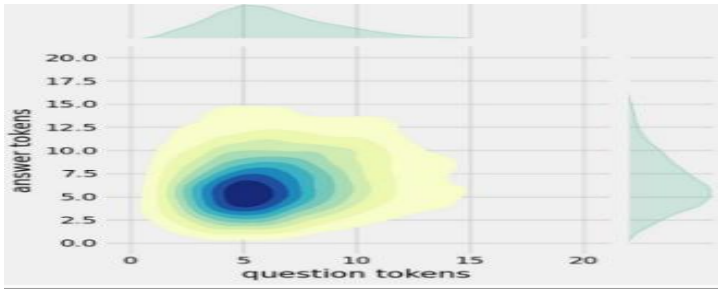
- **DATA CLEANING:**

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.title('Before Cleaning')
plt.xlabel('Count')
plt.title('After Cleaning')
plt.xlabel('Count')
plt.tight_layout()
plt.show()
```



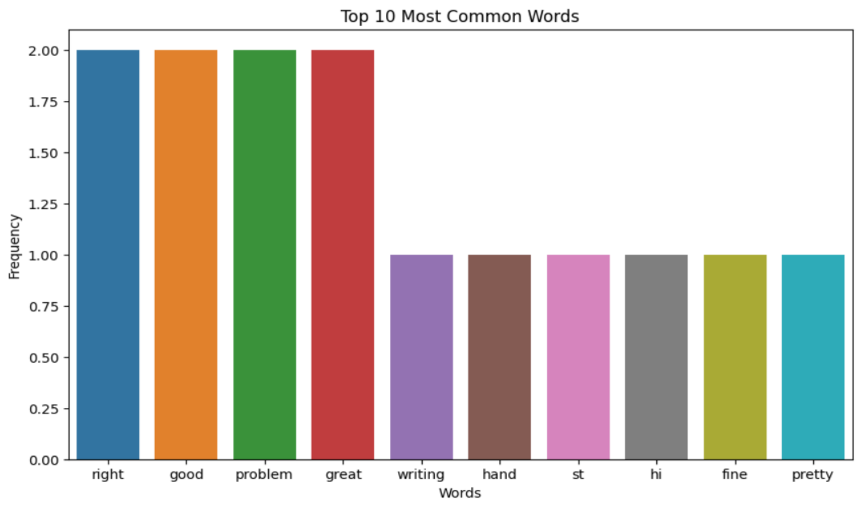
```
df['questiontokens']=df['question'].apply(lambda
x:len(x.split()))
df['answer tokens']=df['answer'].apply(lambda
x:len(x.split()))
plt.style.use('fivethirtyeight')
fig,ax=plt.subplots(nrows=1,ncols=2,
figsize=(205))
sns.jointplot(x='question tokens',y='answer
tokens',data=df,kind='kde',fill=True,cmap='YlGn
Bu')
plt.show()
```





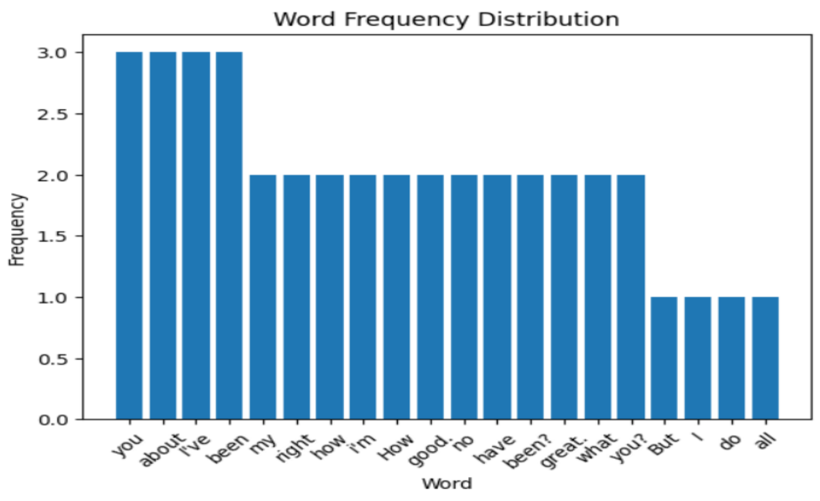
➢ DATA REDUCTION:

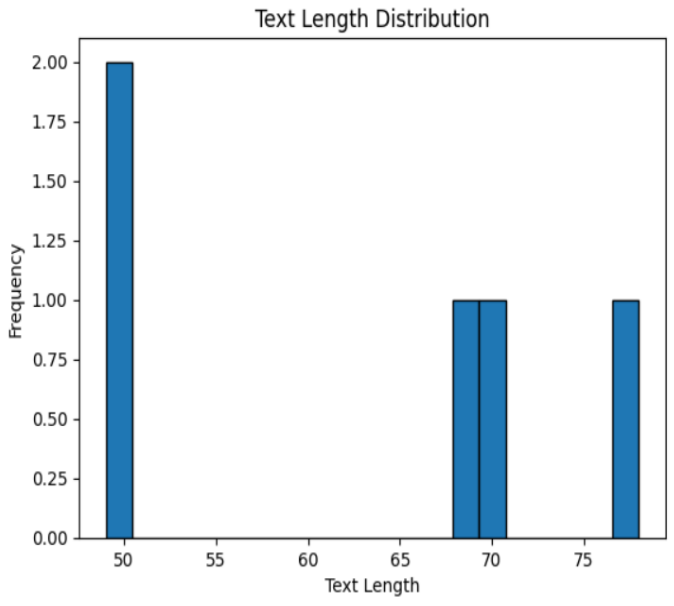
```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
sns.barplot(x=[word[0] for word in
common_words], y=[word[1] for word in
common_words])
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 10 Most Common Words')
plt.show()
```



- **DATA TRANFORMATION:**

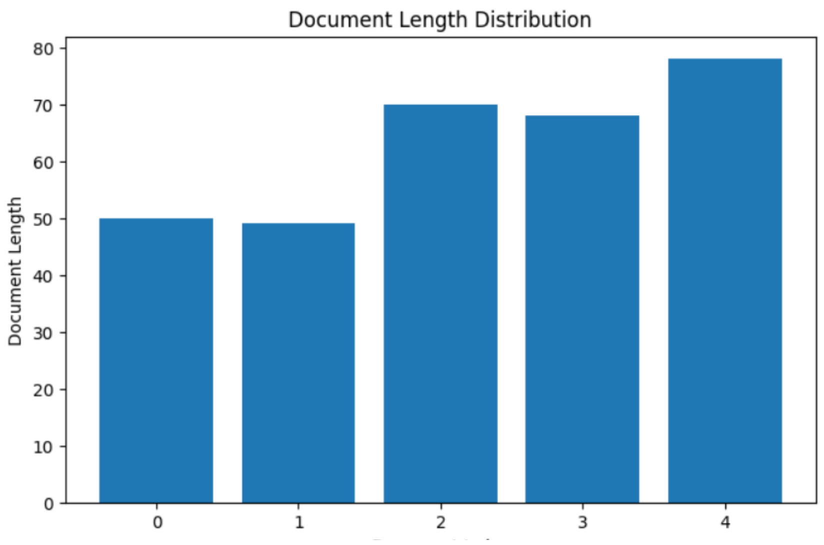
```
import matplotlib.pyplot as plt
import numpy as np
words, counts =
zip(*word_counts.most_common(20))
plt.bar(words, counts)
plt.title("Word Frequency Distribution")
plt.xlabel("Word")
plt.ylabel("Frequency")
plt.xticks(rotation=45)
plt.show()
```

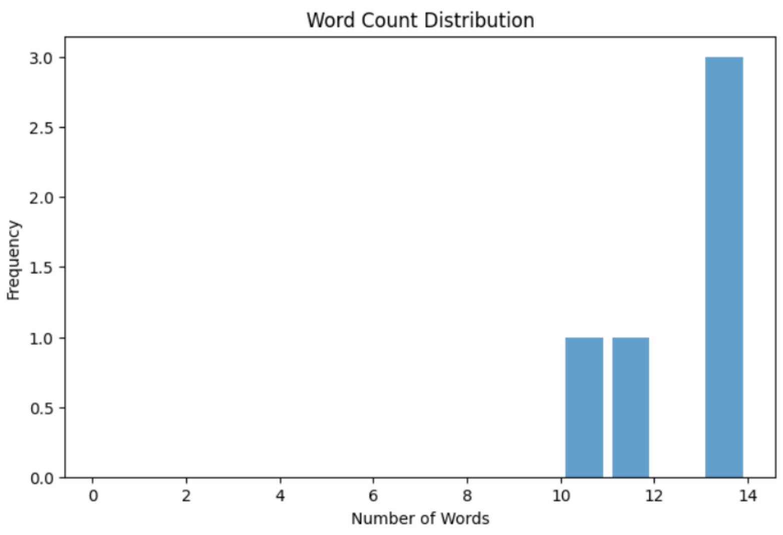




➢ DATA DISCRETIZATION:

```
import matplotlib.pyplot as plt
import pandas as pd
from wordcloud import WordCloud
df["Document Length"] = df["Text"].apply(len)
plt.figure(figsize=(8, 5))
plt.bar(df.index, df["Document Length"])
plt.xlabel("Document Index")
plt.ylabel("Document Length")
plt.title("Document Length Distribution")
word_counts = df["Text"].str.split().apply(len)
plt.figure(figsize=(8, 5))
plt.hist(word_counts, bins=range(0,
max(word_counts) + 1), rwidth=0.8, alpha=0.7)
plt.xlabel("Number of Words")
plt.ylabel("Frequency")
plt.title("Word Count Distribution")
plt.show()
```



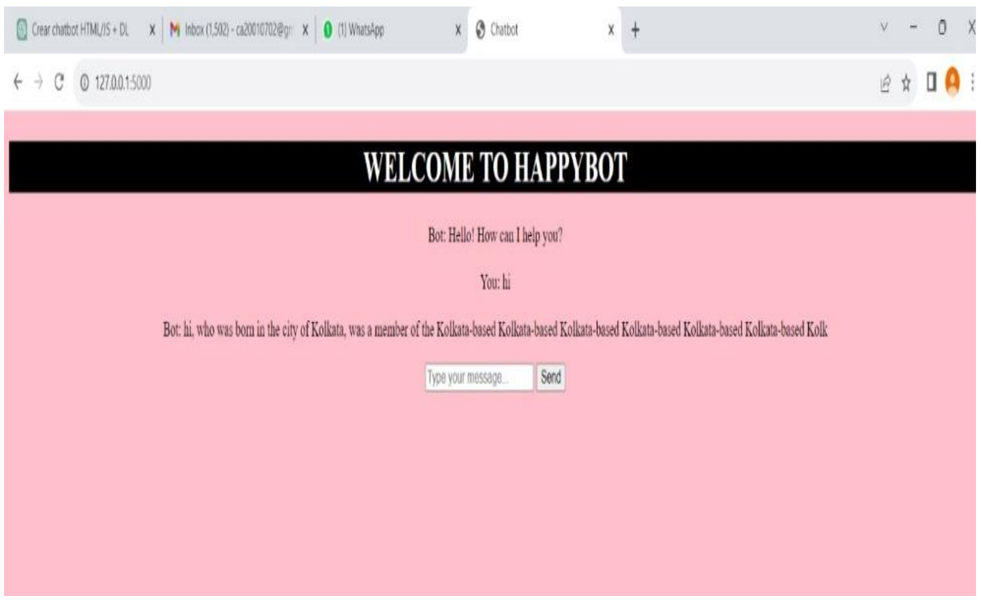


GPT3:

Libraries like transformers for GPT-3 integration was not freely accessible to the public, and access to the model was provided by OpenAI through an API key on a paid basis.So here I am using the GPT-2.

GPT-2(Generative Pre-trained Transformer 2)is a advanced natural language processing model that can understand and generate human-like text for various tasks. It can be fine-tuned for specific tasks, making it suitable for chatbots, content generation, summarization, and translation. However, it is not perfect and may generate incorrect or biased information.

```
from transformers import GPT2LMHeadModel,
GPT2Tokenizer
model_name = "gpt2"  # You can use other models as well
tokenizer =GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)

def generate_response(user_message):
    input_ids = tokenizer.encode(user_message,
return_tensors='pt')
    response_ids = model.generate(input_ids,
max_length=50, num_return_sequences=1)
    response_text = tokenizer.decode(response_ids[0],
skip_special_tokens=True)
  return response_text
```



WELCOME TO HAPPYBOT

Bot: Hello! How can I help you?

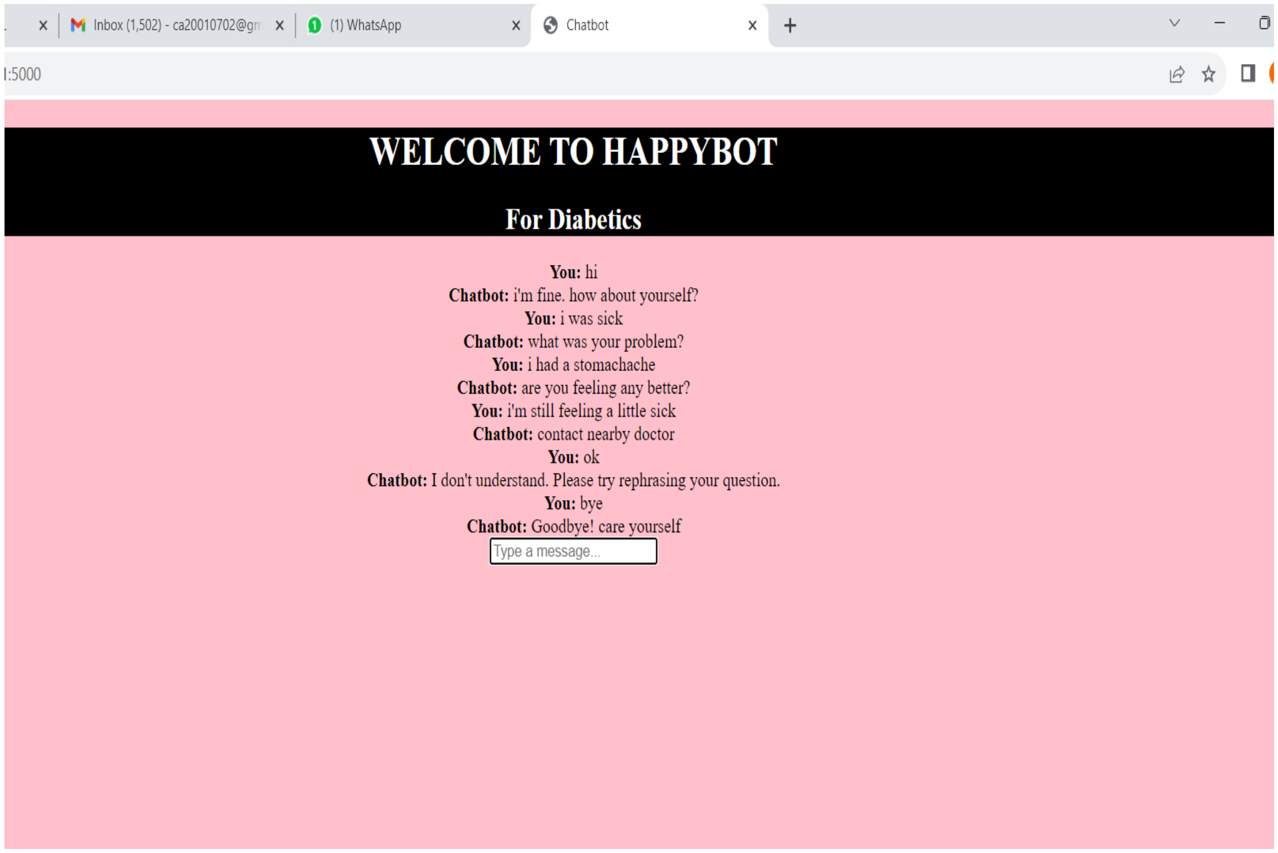
You: hi

Bot: hi, who was born in the city of Kolkata, was a member of the Kolkata-based Kolkata-based Kolkata-based Kolkata-based Kolkata-based Kolkata-based Kolk

FLASK:

Flask is a Python web framework designed for developers to build web applications, APIs, and interactive services with minimal effort. It provides basic tools for routing, handling requests, and rendering web pages. Flask is often used in combination with other Python libraries to create chatbot interfaces and web applications, allowing users to interact with chatbots through a web browser.

```
from flask import Flask, request, jsonify
app = Flask(_name_)
@app.route('/')
def index():
    return open('index.html', 'r').read()
@app.route('/ask', methods=['POST'])
def ask():
    user_message = request.json.get('userMessage', '')
    response = chatbot_response(user_message)
    return jsonify({'response': response})
if _name_ == '_main_':
    app.run(debug=True)
```



WELCOME TO HAPPYBOT

For Diabetics

You: hi
Chatbot: i'm fine. how about yourself?
You: i was sick
Chatbot: what was your problem?
You: i had a stomachache
Chatbot: are you feeling any better?
You: i'm still feeling a little sick
Chatbot: contact nearby doctor
You: ok
Chatbot: I don't understand. Please try rephrasing your question.
You: bye
Chatbot: Goodbye! care yourself

CONCLUSION:

Loading and preprocessing the dataset is crucial for creating a sophisticated chatbot. This involves meticulous preparation of the environment and incorporating user interactions. Installing necessary libraries like transformers for GPT-3 integration and Flask for web app development streamlines the process and equips you with tools for a seamless user experience, ensuring a successful and engaging project.