CREATE A CHATBOT IN PYTHON

COLLEGENAME: CMS COLLEGE OF ENGINEERING AND TECHNOLOGY

NAME: GOWTHAM.K

REGNO: 720421104302

➢ INTRODUCTION

Chatbots are computer programs that can simulate conversation with humans. They are often used in customer service applications, but they can also be used for education, entertainment, and other purposes.

Python is a popular programming language for creating chatbots because it is easy to learn and use. Python also has a large community of developers, which means that there are many resources available to help you create and debug your chatbot.

➢ PROBLEM DEFINITION

The challenge is to create a chatbot in Python that provides exceptional customer service, answering user queries on a website or application. The objective is to deliver high-quality support to users, ensuring a positive user experience and customer satisfaction.

➢ DESIGN THINKING

Functionality

The chatbot should be able to:

- Answer common questions about the product or service
- Provide guidance on how to use the product or service
- Direct users to appropriate resources, such as help articles or support videos

➢ USER INTERFACE

The chatbot should be integrated into the website or app in a user-friendly way. The user interface should be clear and concise, and the chatbot should be easy to interact with.

➢ NATURAL LANGUAGE PROCESSING (NLP)

NLP techniques should be used to understand and process user input in a conversational manner. The chatbot should be able to recognize the intent of the user's query and provide an appropriate response.

➢ RESPONSES

The chatbot should offer accurate answers, suggestions, and assistance to users. The responses should be friendly and helpful, and the chatbot should avoid using technical jargon.

➢ INTEGRATION

The chatbot should be integrated with the website or app in a way that allows it to access the necessary information. For example, the chatbot may need to access the user's account information or product catalog data.

➢ TESTING AND IMPROVEMENT

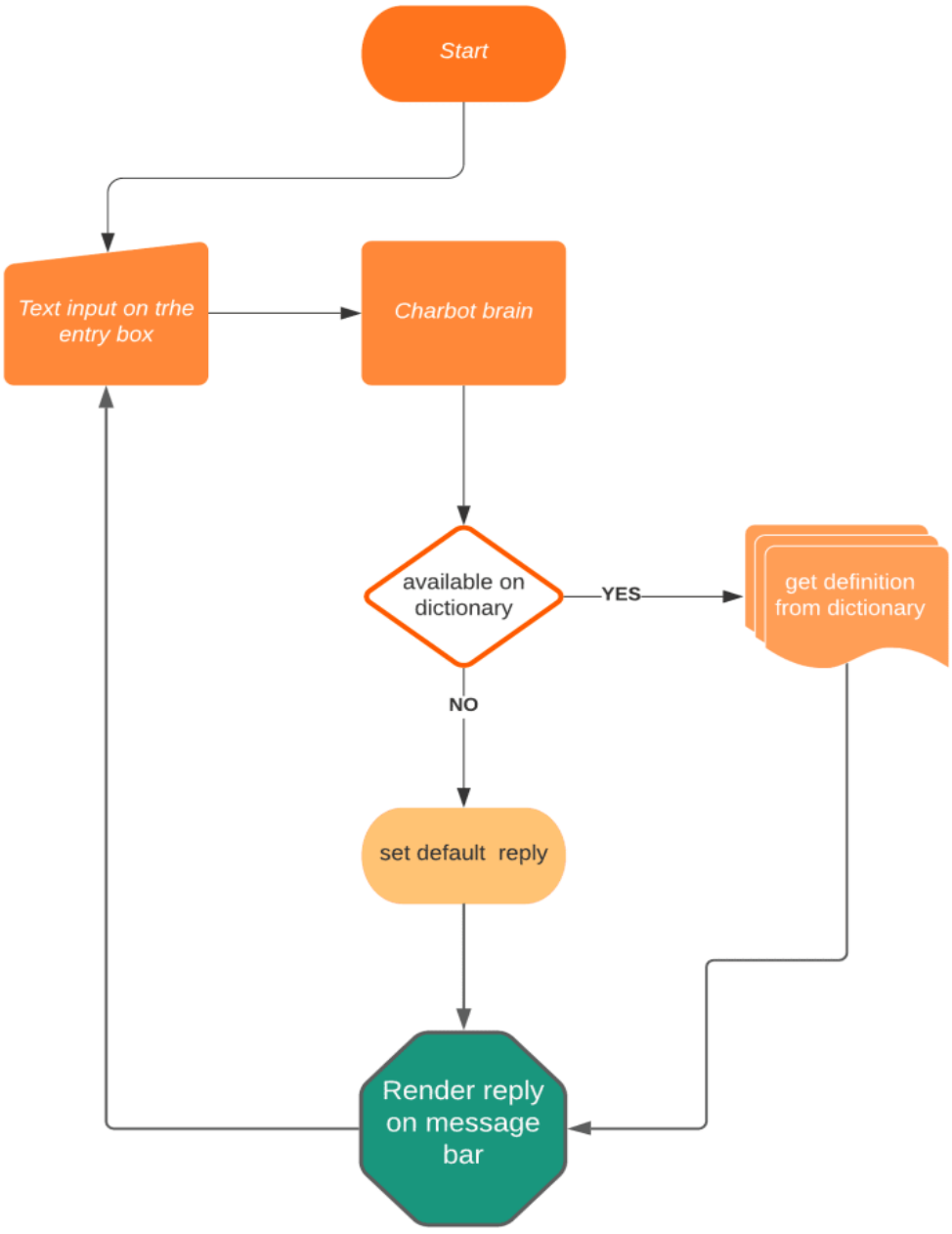
The chatbot should be continuously tested and refined based on user interactions. This will help to ensure that the chatbot is providing accurate and helpful information.

➢ IMPLEMENTATION

The chatbot can be implemented using a variety of Python libraries, such as Rasa or Dialogflow. These libraries provide pre-built NLP models and chatbot development tools.

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

FLOW CHART:



- PROGRAM

```
# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split from sklearn.datasets import load_iris

# Loading data
irisData =load_iris()

# Create feature and target
arraysX = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size = 0.2, random_state=42)
```

```
knn = KNeighborsClassifier(n_neighbors=7)

knn.fit(X_train, y_train)

# Predict on dataset which model has not seen

beforeprint(knn.predict(X_test))
```

- **OUTPUT:**

```
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]

# Import necessary modules
from sklearn.neighbors import
KNeighborsClassifierfrom sklearn.model_selection
import train_test_splitfrom sklearn.datasets import
load_iris


# Loading data
irisData =
load_iris()

 # Create feature and target
 arraysX = irisData.data
 y = irisData.target

 # Split into training and test set
 X_train, X_test, y_train, y_test = train_test_split(
                        X, y, test_size = 0.2,

 random_state=42)                knn                =
```

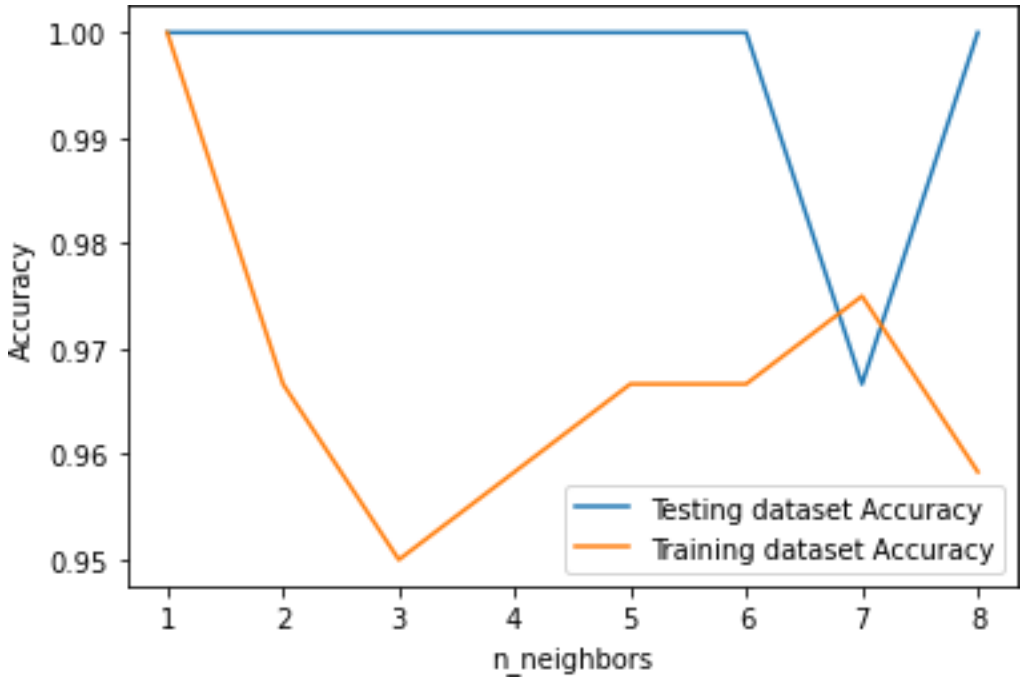
```
KNeighborsClassifier(n_neighbors=7)      knn.fit(X_train,

y_train)

# Calculate the accuracy of the
model          print(knn.score(X_test,
y_test))
```

OUTPUT:

```
0.966666666666667

# Import necessary modules
from sklearn.neighbors import
KNeighborsClassifier from
sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import numpy as np
import  matplotlib.pyplot  as

pltirisData = load_iris()

# Create feature and target
arraysX = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
# Loop over K values
for i, k in enumerate(neighbors):
        knn                                    =
        KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train, y_train)

        # Compute training and test data accuracy
        train_accuracy[i]    =    knn.score(X_train,
```

```
        y_train)test_accuracy[i] = knn.score(X_test,
        y_test)

# Generate plot
plt.plot(neighbors,  test_accuracy,  label  =  'Testing
datasetAccuracy')
plt.plot(neighbors,  train_accuracy,  label  =  'Training
datasetAccuracy')

plt.legend()
plt.xlabel('n_neighbor
s')
plt.ylabel('Accuracy')
plt.show()
```

OUTPUT:



➢ ADVANTAGES

- Cost-effective: Python is a free and open-source language, so there are no licensing costs associated with creating and deploying Python chatbots.
- Easy to learn: Python is a relatively easy language to learn, even for beginners. This makes it a good choice for creating chatbots, as it allows developers to focus on the chatbot's functionality rather than the programming language itself.
- Large community: Python has a large community of developers, which means that there are many resources available to help you create and debug your chatbot. There are also many Python libraries and frameworks available for developing chatbots.

- Scalability: Python chatbots can be scaled to handle large volumes of traffic by adding more servers or using cloud computing platforms.
- Flexibility: Python chatbots can be customized to meet the specific needs of your business. You can add new features, integrate with other systems, and change the chatbot's behavior to meet the needs of your customers.

➢ DISADVANTAGES

- Performance: Python is not as fast as some other programming languages, such as C++ and Java. This can be a disadvantage for chatbots that need to process large amounts of data in real time.
- Complexity: Creating a chatbot that can understand and respond to a wide range of user queries can be complex. This is especially true for chatbots that need to provide customer support or answer technical questions.
- Maintenance: Python chatbots need to be regularly maintained and updated to keep them working properly and to ensure that they are providing the best possible experience for users.

CONCLUSION:

In this article, we covered the workings of the KNN algorithm and its implementation in Python. It's one of the most basic yet effective machine- learning models. For KNN implementation in R, you can go through this tutorial: kNN Algorithm using R. You can also go for our free course – K- Nearest Neighbors (KNN) Algorithm in Python and R, to further your foundations of KNN.

In this article, we used the KNN model directly from the *scikit-learn* library. You can also implement KNN from scratch (I recommend this!), which is covered in this article: KNN simplified.