CREATE A CHATBOT IN PYTHON

NAME:GOPAL LOHAR

REG NO:720421104301

INTRODUCTION :

A chatbot is a computer program that simulates conversation with human users. Chatbots are often used in customer service applications, where they can provide information and answer questions about a company's products or services. They can also be used in educational applications, where they can provide tutoring or help students learn new concepts.

One of the most popular programming languages for creating chatbots is Python. Python is a general-purpose language that is easy to learn and use. It also has a number of libraries and frameworks that make it easy to develop chatbots.

METHOD USED:

K-Nearest Neighbours (KNN) is a popular machine learning algorithm used for classification and regression tasks. It is a **lazy learning**, non-parametric algorithm that uses data with several classes to predict the classification of the new sample point. KNN is non-parametric since it doesn't make any assumptions on the data being studied.

During the training phase, the KNN algorithm stores the entire training dataset as areference. When implementing an algorithm,

you will always need a data set. So, you start by loading the training and the test data. Then, you choose the nearest data points (the value of K). K can be any integer.

The working of KNN Algorithm in Machine Learning can be summarized in threesteps:

1. Load the data
2. Choose the nearest data points (the value of K)
3. Do the following, for each test data –
    - Calculate the distance between test data and each row of training data
    - Sort the calculated distances in ascending order based on distancevalues
    - Get top K rows from sorted array
    - Get the most frequent class of these rows
    - Return this class as output.

PROCESS:

```
# 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 arrays
```

```
X = 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 before

print(knn.predict(X_test))
```

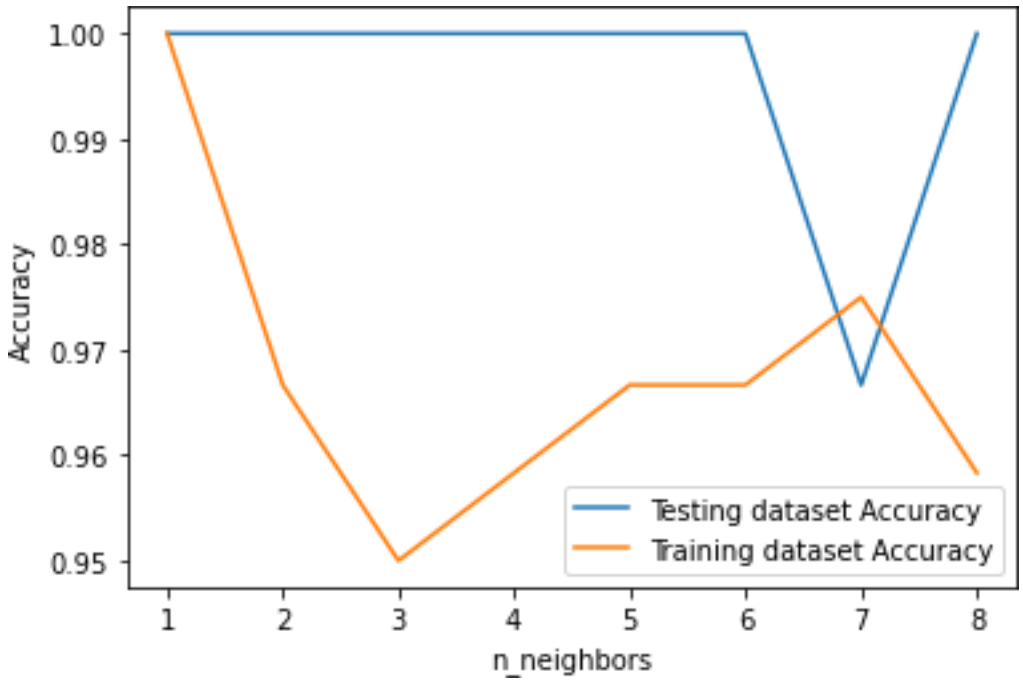
OUTPUT:

```
[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 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)

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

OUTPUT:
0.9666666666666667

# 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 plt

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(
# 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 dataset
Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset
Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```

OUTPUT:



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.