AI Chatbot Using Python:

Problem Definition:

The problem is to build an AI-powered diabetes prediction system that uses machine learning algorithms to analyze medical data and predict the likelihood of an individual developing diabetes. The system aims to provide early risk assessment and personalized preventive measures, allowing individuals to take proactive actions to manage their health.

Design Thinking:

1. Functionality: Define the scope of the chatbot's abilities, including answering common questions, providing guidance, and directing users to appropriate resources.
2. User Interface: Determine where the chatbot will be integrated (website, app) and design a user-friendly interface for interactions.
3. Natural Language Processing (NLP): Implement NLP techniques to understand and process user input in a conversational manner.
4. Responses: Plan responses that the chatbot will offer, such as accurate answers, suggestions, and assistance.
5. Integration: Decide how the chatbot will be integrated with the website or app.
6. Testing and Improvement: Continuously test and refine the chatbot's performance based on user interactions

Preprocessing The Data Set:

Data preprocessing is a crucial step in building a chatbot, as it ensures that the dataset used for training and natural language understanding (NLU) is clean and properly formatted. Here are common data preprocessing steps in chatbot development:

1. **Data Cleaning:**
    - **Remove Duplicates:** Check for and remove duplicate entries in your dataset. Duplicates can lead to biased training and affect the chatbot's performance.
    - **Handle Missing Values:** Address any missing or null values in your dataset. Missing data can lead to issues during training.
2. **Text Cleaning and Normalization:**
    - **Lowercasing:** Convert all text to lowercase to ensure consistency.
    - **Remove Special Characters:** Remove unnecessary special characters, punctuation, and symbols that don't carry significant meaning for the chatbot.
    - **Tokenization:** Split sentences into individual words or tokens. Tokenization is essential for text analysis.

- **Stemming and Lemmatization:** Reduce words to their root form to improve the chatbot's ability to understand variations of a word.

3. **Handling Imbalanced Data:**
    - If your dataset has imbalanced classes, you may need to oversample the minority class or under sample the majority class to ensure balanced training.
4. **Remove Stop Words:**
    - Stop words like "a," "an," "the," and other common words don't typically carry much meaning and can be removed from the text.
5. **Entity Recognition:**
    - Identify and mark entities (e.g., names, dates, locations) in your dataset. Many chatbot frameworks allow you to specify and label entities in your training data.
6. **Feature Engineering:**
    - Create additional features if needed. For instance, you might want to extract features like sentiment scores or named entity types to improve the chatbot's understanding of user input.
7. **Intent and Response Mapping:**
    - Ensure that each user input is correctly mapped to the intended intent and associated response.

8. **Dataset Splitting:**
    - Split your dataset into training, validation, and test sets. The training set is used to train the chatbot, the validation set helps tune hyperparameters, and the test set is used to evaluate the chatbot's performance.
9. **Data Format Conversion:**
    - Convert your dataset into a format suitable for your chatbot framework. For example, Rasa uses Markdown-style training data, while other frameworks may have different data format requirements.
10. **Remove Noise:**
    - Remove any irrelevant or noisy data that doesn't contribute to the chatbot's training.
11. **Data Augmentation:**
    - In some cases, you might consider data augmentation techniques to generate additional training examples, especially if your dataset is small.

Dataset Link:

https://drive.google.com/file/d/1lFech-MF0AVgyUhMf3Brb4x-vSSMoTDl/view?usp=drive_link

Model Training:

Training a chatbot involves several key steps. Below are the general steps involved in training a chatbot:

1. **Define Objectives and Use Cases:**
    - Clearly define the objectives and use cases for your chatbot. Understand what tasks or functions the chatbot should perform and who the target users are.
2. **Data Collection:**
    - Gather or generate training data, including conversations, user messages, and corresponding responses. High-quality and relevant training data is crucial for chatbot success.
3. **Data Preprocessing:**
    - Clean and preprocess the training data. This may involve removing duplicates, irrelevant information, and handling special characters or formatting issues.
4. **Select a Chatbot Framework or Platform:**
    - Choose a chatbot development framework or platform. Popular options include Dialogflow, Microsoft Bot Framework, Rasa, and custom solutions.
5. **Natural Language Processing (NLP) Model Selection:**
    - Choose a suitable NLP model for your chatbot. Pre-trained models like GPT-3, BERT, or others can be fine-tuned for your specific chatbot task.
6. **Model Training:**
    - Train the NLP model using the preprocessed training data. Fine-tuning a pre-trained model can help it understand and generate human-like responses.
7. **Conversation Flow Design:**
    - Design the conversation flow by defining user intents, entities, and expected responses. Create a dialogue tree or flowchart to map out the conversation structure.
8. **Implement Logic and Actions:**
    - Develop the chatbot's logic and actions. Write code to handle user input, trigger appropriate responses, and potentially integrate with external APIs or databases.

9. **User Interface Integration:**
    - Implement the user interface for users to interact with the chatbot. This could be on a website, messaging app, voice interface, or any other platform.
10. **Testing and Evaluation:**
    - Thoroughly test the chatbot to ensure it understands user inputs, provides relevant responses, and handles various scenarios. Gather user feedback and make improvements as necessary.
11. **Deployment:**
    - Deploy the chatbot to the chosen platform or application so that users can start using it.
12. **Monitoring and Maintenance:**
    - Continuously monitor the chatbot's performance, gather user feedback, and make updates to improve its capabilities and accuracy. Regularly update the model to adapt to changing language trends.
13. **Data Privacy and Security:**
    - If your chatbot handles sensitive information, ensure that you have robust security measures in place to protect user data and that you comply with data privacy regulations.
14. **Scaling:**
    - As the chatbot's user base grows, be prepared to scale your infrastructure to handle increased user load.

DEVELOPMENT PART-1

Code for Training:

```
# import modules
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# read the dataset
df = pd.read_csv('Real estate.csv')

# get the locations
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

```
# split the dataset
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.05, random_state=0)
```

Output:

```
1 X_train
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude
37	38	2013.167	12.0	1360.13900	1	24.95204	121.54842
334	335	2012.917	30.0	1013.34100	5	24.99006	121.53460
54	55	2013.083	16.1	289.32480	5	24.98203	121.54348
145	146	2012.917	2.1	451.24380	5	24.97563	121.54694
284	285	2012.917	15.0	383.28050	7	24.96735	121.54464
...	...	...	...	...	...	...	...
323	324	2013.417	28.6	197.13380	6	24.97631	121.54436
192	193	2013.167	43.8	57.58945	7	24.96750	121.54069
117	118	2013.000	13.6	4197.34900	0	24.93885	121.50383
47	48	2013.583	35.9	640.73910	3	24.97563	121.53715
172	173	2013.583	6.6	90.45606	9	24.97433	121.54310

```
1 y_train
```

```
37     25.3
334    22.8
54     51.7
145    45.5
284    34.4
       ...
323    42.5
192    42.7
117    13.0
47     61.5
172    58.1
Name: Y house price of unit area, Length: 393, dtype: float64
```

Explanation:

In the above code, **We import the pandas package and sklearn package**. after that to import the CSV file we use the read_csv() method.

The variable df now contains the data frame. in the example “house price” is the column we’ve to predict so we take that column as y and the rest of the columns as our X variable. test_size = 0.05 specifies only 5% of the whole data is taken as our test set, and 95% as our train set.

The random state helps us get the same random split each time.

Development part-2

Step1: setting upon our environment:

```
import requests

api_key = "your_api_key"

def get_weather(city_name):
api_url =
"http://api.openweathermap.org/data/2.5/weather?q={}&appid={}".format(city_na
me, api_key)
```

```
response = requests.get(api_url)
response_dict = response.json()

    weather = response_dict["weather"][0]["description"]

if response.status_code == 200:
return weather
else:
print('[!] HTTP {0} calling [{1}]'.format(response.status_code, api_url))
return None
```

Output:

```
Chatbot: Hi there! How can I assist you today?
You: What's the weather in New York?
Chatbot: The weather in New York is overcast clouds with a temperature of 18
You: Tell me the 5-day weather forecast for London.
Chatbot: Sorry, I couldn't fetch the weather information at the moment.
You: Bye
Chatbot: Goodbye!
```

Explanation:

you'll create a chatbot capable of figuring out whether the user wants to get the current weather in a city, and if so, the chatbot will use the get_weather() function to respond appropriately.

Step2:execution of a chatbot:

```
import random

# Define a list of greetings and responses
greetings = ["hello", "hi", "hey", "greetings", "howdy"]
responses = ["Hello!", "Hi there!", "Hey!", "Greetings!", "Howdy!"]

# Define a function to respond to user input
def chatbot_response(user_input):
    user_input = user_input.lower()

    if user_input in greetings:
        return random.choice(responses)
    else:
        return "I'm just a simple chatbot. I don't understand that."

# Main conversation loop
print("Chatbot: Hello! How can I help you today? (Type 'exit' to end)")

while True:
    user_input = input("You: ")
    if user_input.lower() == 'exit':
```

```
        print("Chatbot: Goodbye!")
        break

    response = chatbot_response(user_input)
    print("Chatbot:", response)
```

Output:

```
Chatbot: Hello! How can I help you today? (Type 'exit' to end)
You: Hi
Chatbot: Hey!
You: What's the weather like today?
Chatbot: I'm just a simple chatbot. I don't understand that.
You: exit
Chatbot: Goodbye!
```

Explanation:

Copy and paste this code into a Python environment, and you can have a basic conversation with the chatbot. It will respond to greetings with appropriate responses and display a default response if it doesn't recognize the input. Type "exit" to end the conversation.

Conclusion:

phase1. Planning and Design: This initial phase focused on defining the chatbot's purpose, target audience, and the conversation flow. It involved designing the user experience and outlining the bot's capabilities and limitations.

phase2. Data Collection and Processing: Gathering and preprocessing the necessary data and training materials were essential in building a chatbot that could understand and respond to user inputs effectively. This phase included data cleaning, text preprocessing, and language model selection.

phase3. Model Training: Training the chatbot's natural language processing model was a critical step. This involved using machine learning or deep learning techniques to enable the chatbot to comprehend and generate human-like responses.

phase4. Integration and Deployment: Integrating the chatbot into the desired platform or communication channels and deploying it for real-world use was the phase where the chatbot became accessible to users.

phase5. Testing and Improvement: Continuous testing and refinement were crucial to enhancing the chatbot's performance. Gathering user feedback and making iterative improvements helped in ensuring the chatbot provided a better user experience over time.

These five phases, when executed meticulously and iteratively, resulted in a functional chatbot that could engage in meaningful conversations and provide value to its users. Building and maintaining a chatbot involves ongoing effort and improvement to keep it relevant and user-friendly."

This conclusion summarizes the key phases involved in developing a chatbot in Python and underscores the importance of each phase in creating a successful and effective chatbot.

Saravanan.S

Nandha college of technology