**PROJECT : Create a chatbot in Python**

**S.Vigneshwaran**

**912421104054**

A hand holding a phone with a chat bubble

Description automatically generated

PHASE 3:*Development Part 1*

Start building the chatbot by preparing the environment and implementing basic user interactions.

**AI-Powered Diabetes Prediction System: Development Part 1**

***Table of Contents***

* + Introduction
  + Import Libraries and Load the Data
  + Loading the dataset
    - Preprocessing the Data
      * + Sentence Segmentation
        + Normalization
        + Tokenization
  + Conclusion

**Introduction**

* In the ever-evolving landscape of healthcare, the integration of artificial intelligence (AI) is revolutionizing patient care and promoting proactive health management.
* This project aims to create an AI-powered diabetes prediction chatbot that leverages machine learning algorithms to analyze medical data and predict an individual's likelihood of developing diabetes.
* The primary objective of this chatbot is to offer early risk assessment and personalized preventive measures, empowering individuals to take informed actions to safeguard their health.
* To kickstart this ambitious project, we need to lay the groundwork and set up a robust environment for developing and deploying the chatbot. Here's a brief overview of the initial steps:

Given data set:

****

**Necessary step to follow:**

**1.Import Libraries:**

Start by importing the necessary libraries:

**Program:**

import tensorflow as tf

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from tensorflow.keras.layers import TextVectorization

import re,string

from tensorflow.keras.layers import LSTM,Dense,Embedding,Dropout,LayerNormalization

**2.Load the Dataset:**

**Program:**

df=pd.read\_csv('Chatbot.txt',sep='\t',names=['question','answer'])

print(f'Dataframe size: {len(df)}')

df.head()

**3. Exploratory Data Analysis (EDA):**

Perform EDA to understand your data better. This includes checking for missing values, exploring the data's statistics, andvisualizing it to identify patterns.

**Program:**

# Check for missing values

print(df.isnull().sum())

# Explore statistics

print(df.describe())

# Visualize the data (e.g., histograms, scatter plots, etc.)

**4. Feature Engineering:**

Depending on your dataset, you may need to create new features or transform existing ones. This can involve one-hot encoding categorical variables, handling date/time data, or scaling numerical features.

**Program:**

# Example: One-hot encoding for categorical variables

df = pd.get\_dummies(df, columns=[' Avg. Area Income ', ' Avg. AreaHouse Age '])

**5. Split the Data:**

Split your dataset into training and testing sets. This helps you evaluateyour model's performance later.

**Program:**

X = df.drop('price', axis=1) # Features

y = df['price'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**6. Feature Scaling:**

Apply feature scaling to normalize your data, ensuring that all features have similar scales. Standardization (scaling to mean=0 and std=1) is a common choice.

**Program:**

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**Importance of loading and processing dataset:**

* Loading and processing datasets is of paramount importance in data-driven fields like machine learning and data analysis. A well-handled dataset serves as the foundation for accurate modeling, decision-making, and insights. Proper loading ensures data integrity, preventing errors in subsequent analyses.
* Data processing, which includes cleaning, normalization, and feature engineering, enhances data quality, making it more suitable for algorithmic applications.
* Effective handling of datasets enables researchers, data scientists, and AI systems to uncover valuable patterns, trends, and hidden information, thus facilitating informed decision-making, predictive modeling, and the advancement of various domains, from healthcare to finance and beyond.

**Challenges involved in loading and preprocessing chatbot dataset:**

**1.Data Variety:**

* Chatbot datasets often contain a wide variety of data formats, including text, images, and audio. Handling and processing these diverse data types can be challenging.

**2.Data Volume:**

* Chatbots interact with a large number of users, resulting in substantial amounts of data. Managing and processing large volumes of data efficiently is a challenge.

**3.Data Cleaning**:

* Cleaning text data is vital to remove noise, correct spelling errors, and standardize formats. However, chatbot data often includes user-generated content with typos, slang, and colloquial language, making cleaning and normalization challenging.

**4.Context Understanding:**

To provide relevant responses, chatbots need to understand the context of a conversation. This involves tracking user history, recognizing intent, and maintaining context, which can be complex

**5.Privacy and Security:**

Chatbot data often contains sensitive information, such as health data or personal details. Ensuring data privacy and security while processing and storing this information is crucial and presents significant challenges.

**How to overcome the challenges of loading and preprocessing Chatbot dataset:**

To overcome the challenges of loading and preprocessing a chatbot dataset, you can implement the following strategies and best practices

**Data Cleaning and Normalization:**

* Implement text preprocessing techniques to handle spelling errors, slang, and colloquial language.
* Use libraries for text cleaning, stemming, and lemmatization to standardize text data.

**Data Collection and Annotation:**

* Gather a diverse and representative dataset to ensure the chatbot can handle a wide range of user queries.
* Annotate the data with intent labels and entities to aid intent recognition.

**Context Management:**

* Develop context management systems that track user conversations and maintain context for more coherent interactions.

**Multilingual Support:**

* Implement language identification techniques to handle multilingual data.
* Use translation services or models to convert non-English queries into a common language for processing.

**Data Privacy and Security:**

* Anonymize or pseudonymize sensitive user data to protect privacy.
* Ensure compliance with data protection regulations (e.g., GDPR) through robust security measures.

**1.Loading the dataset:**

* Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
  + The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used.
  + However, there are some general steps that are common to most machine learning frameworks

**a.Identify the dataset:**

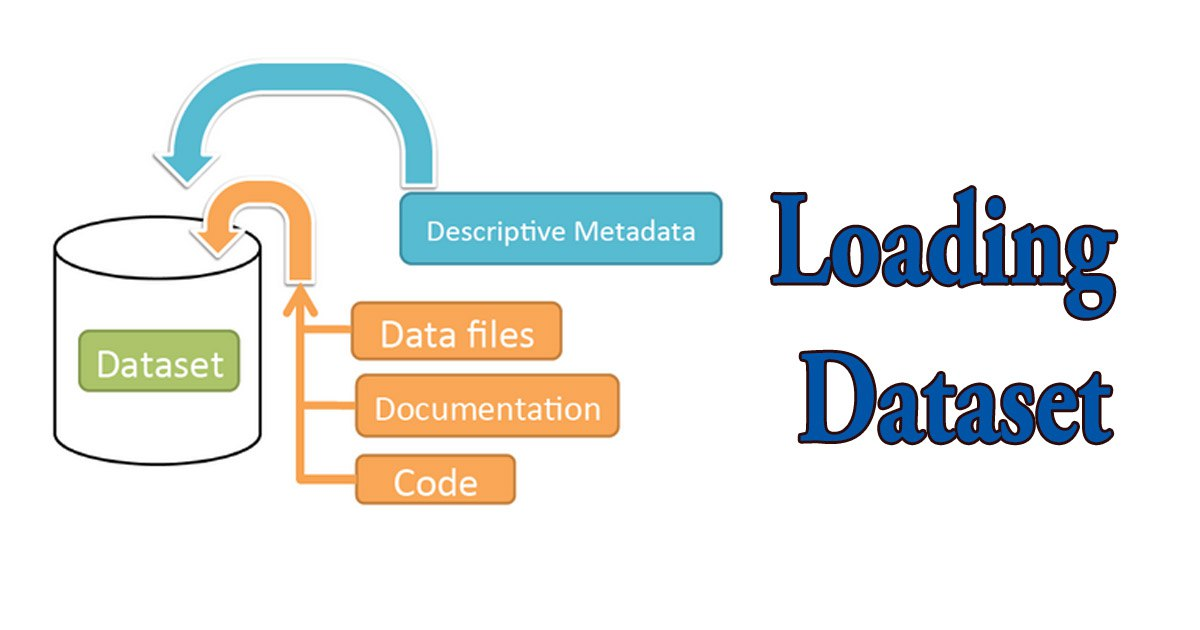
* The first step is to identify the dataset that you want to load.
* This dataset may be stored in a local file, in a database, or in a cloud storage service.

**b.Load the dataset:**

* Once you have identified the dataset, you need to load it into the machine learning environment.
* This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

**c.Preprocess the dataset:**

* Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model.
* This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.



Here, how to load a dataset using machine learning in Python

**Program:**

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 r2\_score,

mean\_absolute\_error,mean\_squared\_error

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Lasso from sklearn.ensemble

import RandomForestRegressor

from sklearn.svm import SVR import

xgboost as xg

%matplotlib inline

import warnings

warnings.filterwarnings("ignore")

/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5

warnings.warn(f"A NumPy version >={np\_minversion} and<{np\_maxversion}"

**Loading Dataset:**

dataset = pd.read\_csv('E:/USA\_Housing.csv')

**Data Exploration:**

**Output:**



**2.Preprocessing the dataset:**

* + - Data preprocessing is the process of cleaning, transforming, and integrating data in order to make it ready for analysis.
    - This may involve removing errors and inconsistencies, handling missing values, transforming the data into a consistent format, and scaling the data to a suitable range.

**Segmentation:**

In [1]:

data=open('/kaggle/input/simple-dialogs-for-chatbot/dialogs.txt','r').read()

In [2]:

QA\_list=[QA.split('**\t**') for QA **in** data.split('**\n**')]

print(QA\_list[:5])

Out [1]:

[['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."]]

In [3]:

questions=[row[0] for row **in** QA\_list]

answers=[row[1] for row **in** QA\_list]

In [4]:

print(questions[0:5])

print(questions[0:5])

Out [2]:

['hi, how are you doing?', "i'm fine. how about yourself?", "i'm pretty good. thanks for asking.", 'no problem. so how have you been?', "i've been great. what about you?"]

["i'm fine. how about yourself?", "i'm pretty good. thanks for asking.", 'no problem. so how have you been?', "i've been great. what about you?", "i've been good. i'm in school right now."]

**Normalization:**

In [5]:

def remove\_diacritic(text):

return ''.join(char for char in unicodedata.normalize('NFD',text)

if unicodedata.category(char) !='Mn')

In [6]:

def preprocessing(text):

#Case folding and removing extra whitespaces

text=remove\_diacritic(text.lower().strip())

#Ensuring punctuation marks to be treated as tokens

text=re.sub(r"([?.!,¿])", r" \1 ", text)

#Removing redundant spaces

text= re.sub(r'[" "]+', " ", text)

#Removing non alphabetic characters

text=re.sub(r"[^a-zA-Z?.!,¿]+", " ", text)

text=text.strip()

#Indicating the start and end of each sentence

text='<start> ' + text + ' <end>'

    return text

In [7]:

preprocessed\_questions=[preprocessing(sen) for sen in questions]

preprocessed\_answers=[preprocessing(sen) for sen in answers]

print(preprocessed\_questions[0])

print(preprocessed\_answers[0])

Out [3]:

<start> hi , how are you doing ? <end>

<start> i m fine . how about yourself ? <end>

**Tokenization:**

In [8]:

def tokenize(lang):

lang\_tokenizer = tf.keras.preprocessing.text.Tokenizer(

filters='')

#build vocabulary on unique words

lang\_tokenizer.fit\_on\_texts(lang)

return lang\_tokenizer

**Some common data preprocessing tasks include**:

**Data cleaning**:

* This involves identifying and correcting errors and inconsistencies in the data. For example, this may involve ere moving duplicate records, correcting typos, and filling in missing values.

**Data transformation:**

* This involves converting the data into a format that is suitable for the analysis task. For example, this may involve converting categorical data to numerical data, or scaling the data to a suitable range.

**Feature engineering:**

* This involves creating new features from the existing data. For example, this may involve creating features that represent interactions between variables, or features that represent summary statistics of the data.

**Data integration:**

* This involves combining data from multiple sources into a single dataset. This may involve resolving in consistencies in the data, such as different data formats or different variable names.
* Data preprocessing is an essential step in many data science projects. By carefully preprocessing the data, data scientists can improve the accuracy and reliability of their results.

**Conclusion:**

* In conclusion, the process of loading and preprocessing data for Diabetes Prediction in a chatbot is a critical and foundational step in building an effective and accurate predictive model. Proper data handling sets the stage for the success of the entire project.
* loading and preprocessing data for diabetes prediction in a chatbot is a multifaceted process that requires careful consideration and attention to detail.
* Ensuring data quality, feature engineering, proper scaling, and ethical handling of sensitive health data are all critical components of this process.

A well-prepared dataset lays the foundation for an accurate and reliable diabetes prediction model within your chatbot, contributing to its overall effectiveness in assisting and educated well.