CHAT BOT USING NLP

Pinisingi Saikarthik

VU21CSEN0100398

1. Abstract

The project focuses on building a machine learning-based chatbot that can respond to user inputs by predicting appropriate responses. Leveraging natural language processing (NLP) techniques, the chatbot processes text data to understand and generate human-like conversations.

2. Skills Required

Python Programming: Essential for implementing data processing, machine learning models, and NLP tasks.

Natural Language Processing (NLP): Core for understanding and transforming textual data, including tokenization, lemmatization, and vectorization.

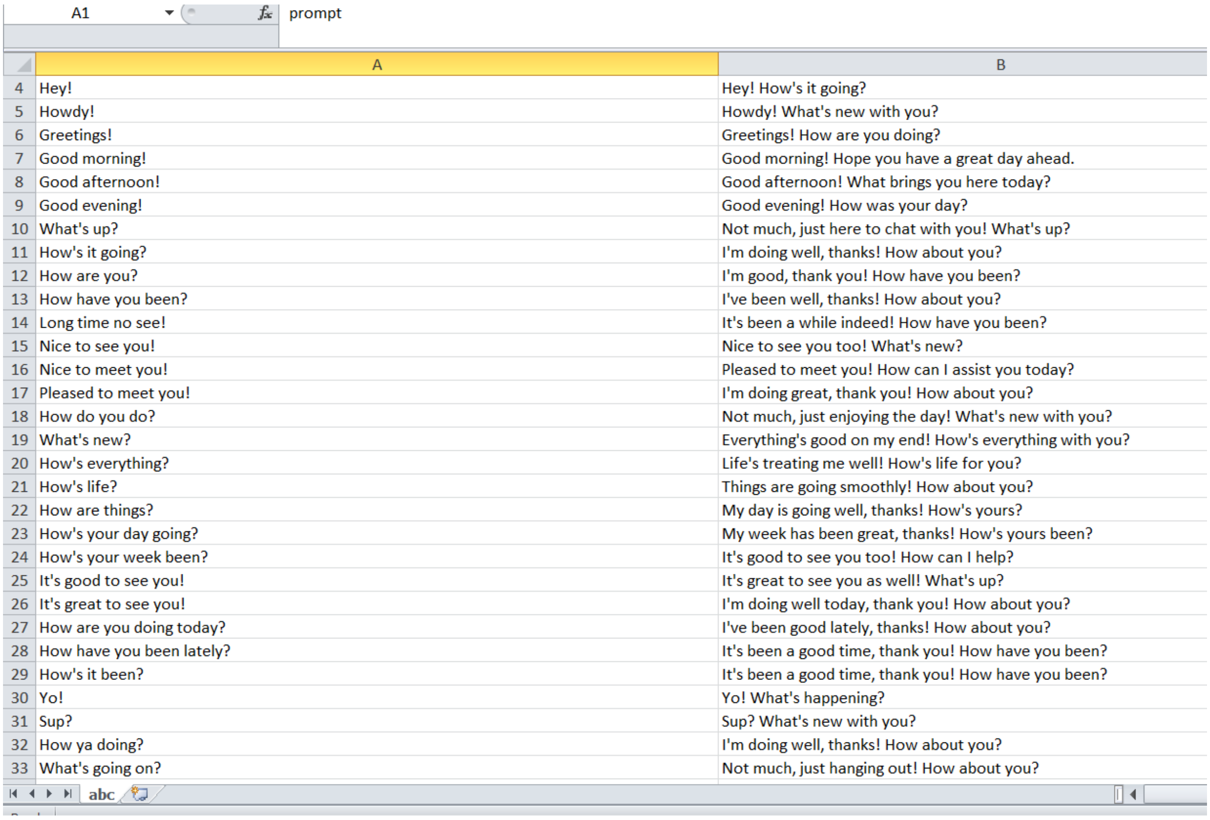
Machine Learning: Specifically, expertise in classification algorithms like Random Forest, used for training and predicting responses.

Data Preprocessing: Necessary to clean and prepare textual data for analysis and modeling.

Model Evaluation: Skills to assess the performance of models using metrics like accuracy, precision, and recall.

3. Dataset

The dataset should consist of a variety of user inputs (prompt) and corresponding responses (reaction). This text-based dataset can be sourced from real conversations or generated synthetically, and it should be diverse enough to train the chatbot on multiple conversational scenarios.



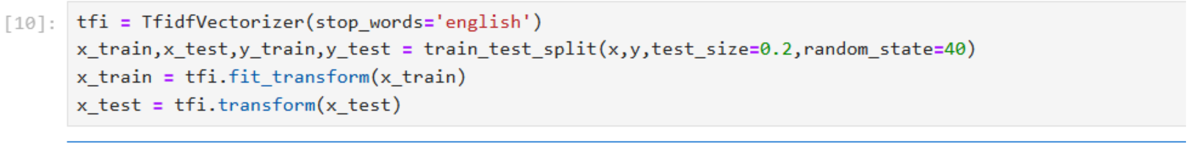
A1 fx prompt

	A	B
4	Hey!	Hey! How's it going?
5	Howdy!	Howdy! What's new with you?
6	Greetings!	Greetings! How are you doing?
7	Good morning!	Good morning! Hope you have a great day ahead.
8	Good afternoon!	Good afternoon! What brings you here today?
9	Good evening!	Good evening! How was your day?
10	What's up?	Not much, just here to chat with you! What's up?
11	How's it going?	I'm doing well, thanks! How about you?
12	How are you?	I'm good, thank you! How have you been?
13	How have you been?	I've been well, thanks! How about you?
14	Long time no see!	It's been a while indeed! How have you been?
15	Nice to see you!	Nice to see you too! What's new?
16	Nice to meet you!	Pleased to meet you! How can I assist you today?
17	Pleased to meet you!	I'm doing great, thank you! How about you?
18	How do you do?	Not much, just enjoying the day! What's new with you?
19	What's new?	Everything's good on my end! How's everything with you?
20	How's everything?	Life's treating me well! How's life for you?
21	How's life?	Things are going smoothly! How about you?
22	How are things?	My day is going well, thanks! How's yours?
23	How's your day going?	My week has been great, thanks! How's yours been?
24	How's your week been?	It's good to see you too! How can I help?
25	It's good to see you!	It's great to see you as well! What's up?
26	It's great to see you!	I'm doing well today, thank you! How about you?
27	How are you doing today?	I've been good lately, thanks! How about you?
28	How have you been lately?	It's been a good time, thank you! How have you been?
29	How's it been?	It's been a good time, thank you! How have you been?
30	Yo!	Yo! What's happening?
31	Sup?	Sup? What's new with you?
32	How ya doing?	I'm doing well, thanks! How about you?
33	What's going on?	Not much, just hanging out! How about you?

4. Feature Extraction

Text Preprocessing: Before feature extraction, text is cleaned by removing punctuation, converting to lowercase, and applying lemmatization, which standardizes words to their base forms (e.g., "running" to "run").

TF-IDF Vectorization: The Term Frequency-Inverse Document Frequency (TF-IDF) method is used to convert text into numerical features. TF-IDF scores represent how important a word is to a document relative to a collection of documents, helping in distinguishing between commonly used words and those that are more contextually significant[1].



```
[10]: tfi = TfidfVectorizer(stop_words='english')
      x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)
      x_train = tfi.fit_transform(x_train)
      x_test = tfi.transform(x_test)
```

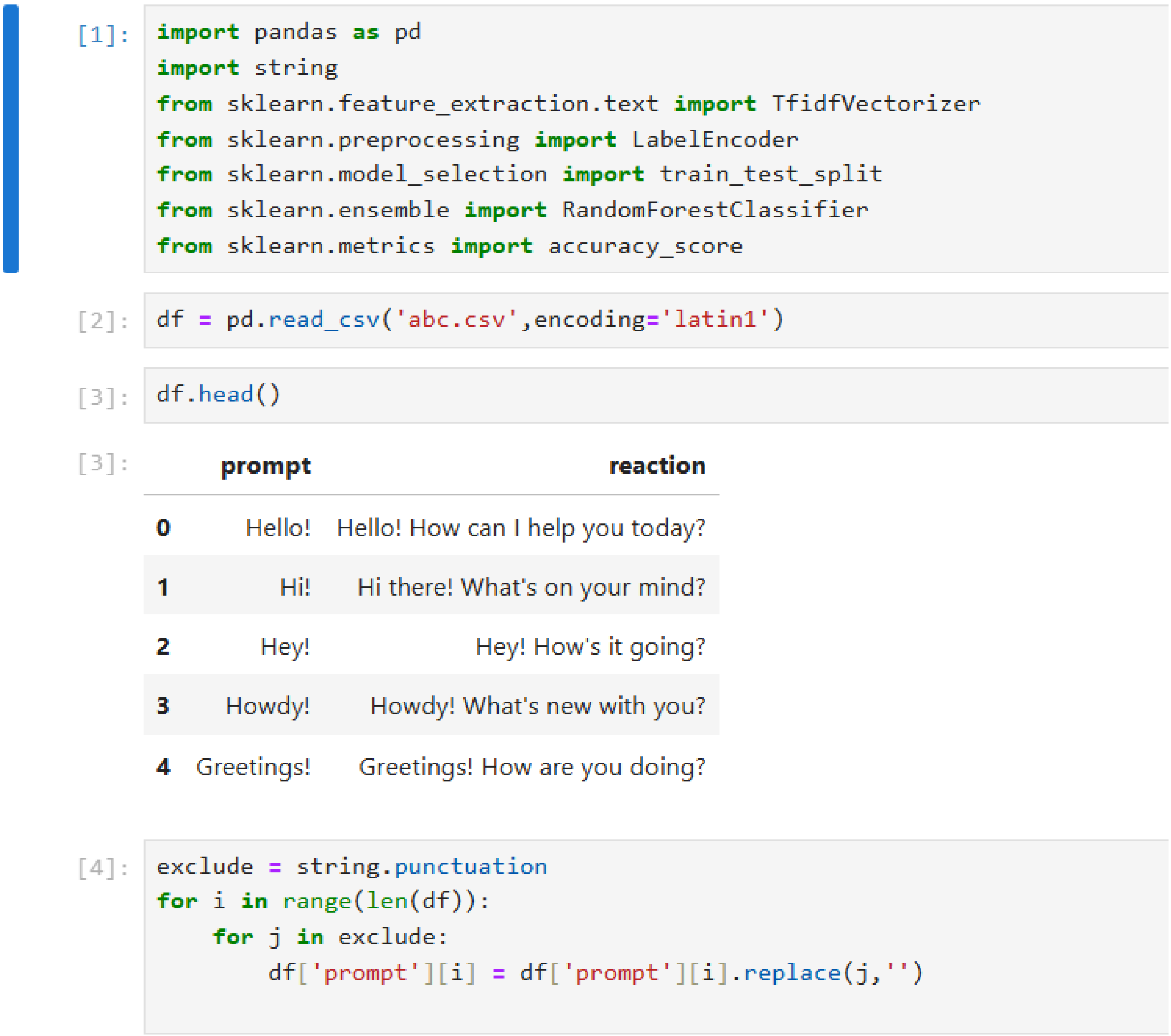
5. Model Training

Random Forest Classifier: A Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) of the individual trees[2]. This technique reduces overfitting and improves accuracy, making it suitable for handling complex datasets like text.

6. Model Evaluation

Accuracy Score: The accuracy score is the ratio of correctly predicted instances to the total instances in the dataset[3]. It is commonly used for evaluating classification models, especially in a well-balanced dataset.

7. IMPLEMENTATION



```
[1]: import pandas as pd
     import string
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.preprocessing import LabelEncoder
     from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
```

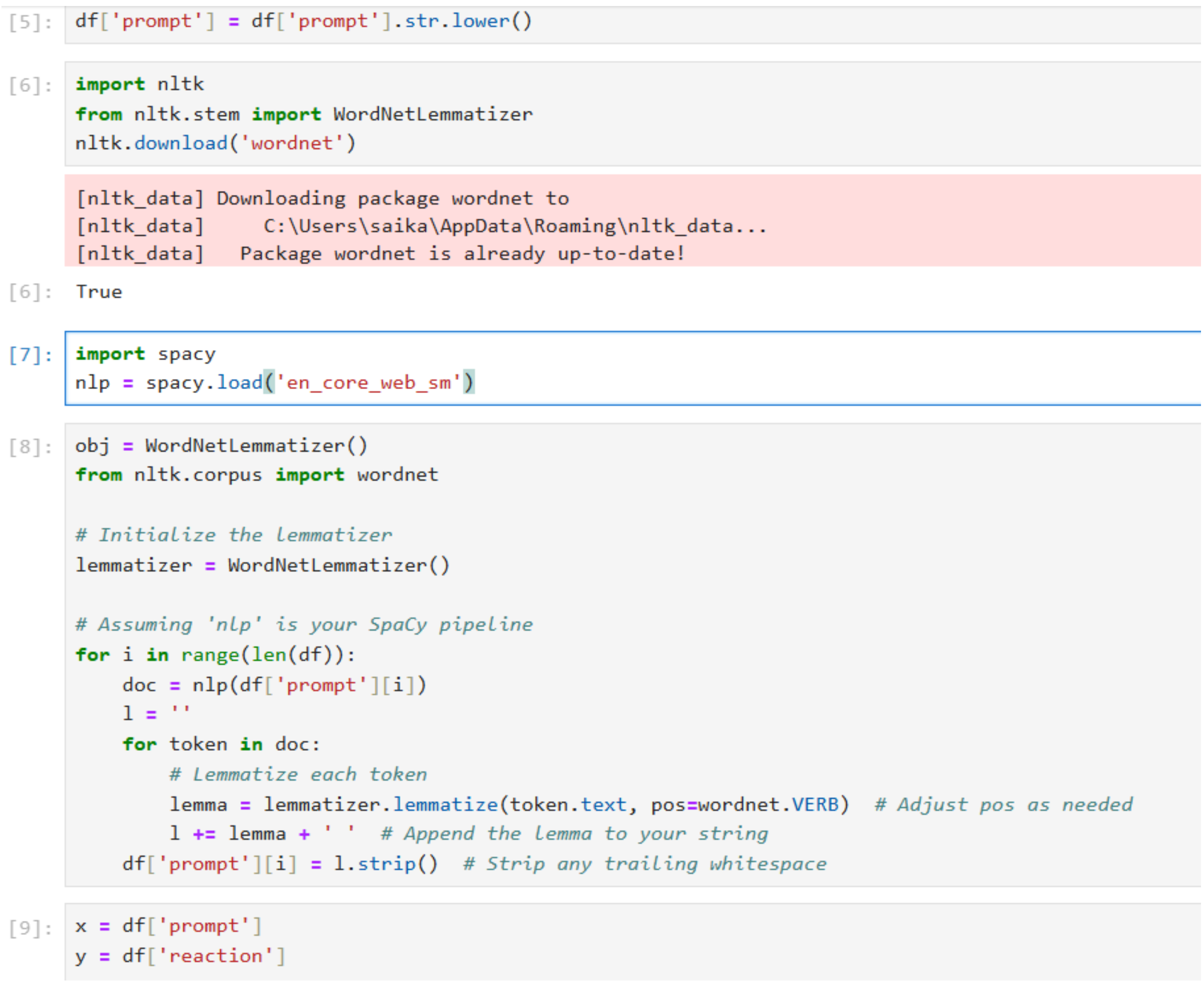
```
[2]: df = pd.read_csv('abc.csv',encoding='latin1')
```

```
[3]: df.head()
```

[3]:

	prompt	reaction
0	Hello!	Hello! How can I help you today?
1	Hi!	Hi there! What's on your mind?
2	Hey!	Hey! How's it going?
3	Howdy!	Howdy! What's new with you?
4	Greetings!	Greetings! How are you doing?

```
[4]: exclude = string.punctuation
     for i in range(len(df)):
         for j in exclude:
             df['prompt'][i] = df['prompt'][i].replace(j,'')
```



```
[5]: df['prompt'] = df['prompt'].str.lower()
```

```
[6]: import nltk
     from nltk.stem import WordNetLemmatizer
     nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\saika\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
[6]: True
```

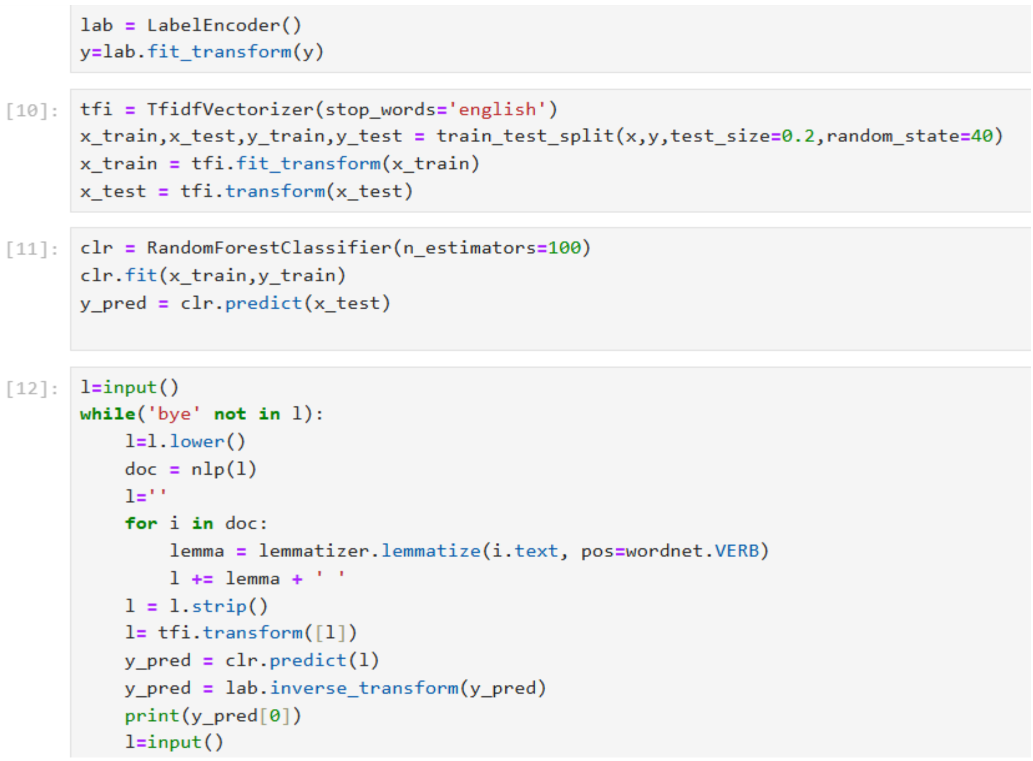
```
[7]: import spacy
     nlp = spacy.load('en_core_web_sm')
```

```
[8]: obj = WordNetLemmatizer()
     from nltk.corpus import wordnet

     # Initialize the lemmatizer
     lemmatizer = WordNetLemmatizer()

     # Assuming 'nlp' is your SpaCy pipeline
     for i in range(len(df)):
         doc = nlp(df['prompt'][i])
         l = ''
         for token in doc:
             # Lemmatize each token
             lemma = lemmatizer.lemmatize(token.text, pos=wordnet.VERB)  # Adjust pos as needed
             l += lemma + ' '  # Append the lemma to your string
         df['prompt'][i] = l.strip()  # Strip any trailing whitespace
```

```
[9]: x = df['prompt']
     y = df['reaction']
```



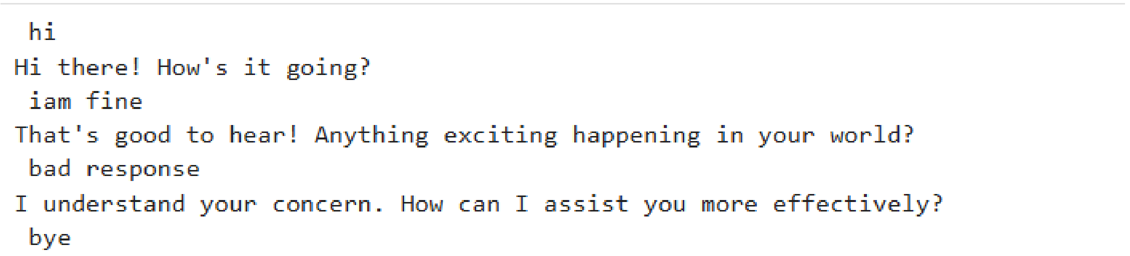
```
lab = LabelEncoder()
y=lab.fit_transform(y)
```

```
[10]: tfi = TfidfVectorizer(stop_words='english')
      x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)
      x_train = tfi.fit_transform(x_train)
      x_test = tfi.transform(x_test)
```

```
[11]: clr = RandomForestClassifier(n_estimators=100)
      clr.fit(x_train,y_train)
      y_pred = clr.predict(x_test)
```

```
[12]: l=input()
      while('bye' not in l):
          l=l.lower()
          doc = nlp(l)
          l=''
          for i in doc:
              lemma = lemmatizer.lemmatize(i.text, pos=wordnet.VERB)
              l += lemma + ' '
          l = l.strip()
          l= tfi.transform([l])
          y_pred = clr.predict(l)
          y_pred = lab.inverse_transform(y_pred)
          print(y_pred[0])
          l=input()
```

8. Output



```
 hi
Hi there! How's it going?
 iam fine
That's good to hear! Anything exciting happening in your world?
 bad response
I understand your concern. How can I assist you more effectively?
 bye
```

🌐 Sources

scikit-learn.org - TfidfVectorizer

geeksforgeeks.org - Random Forest Classifier using Scikit-learn

geeksforgeeks.org - Difference between score() and accuracy_score() methods in scikit-learn