HATE SPEECH DETECTION

A project report submitted in partial fulfillment of the requirement for degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

T MEENA (R170572)

Under the guidance of

Ms C. Suneetha
Asst. Professor
Department of CSE

AP IIIT, RGUKT-RK Valley,
Vempalli, Kadapa (Dist), Andhra Pradesh516330,India.

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(A.P. Government Act 18 of 2008)

RGUKT-RK Valley

Vempalli , Kadapa, Andhrapradesh-516330, India

CERTIFICATE OF EXAMINATION

This is to certify that we have examined the thesis entitled **Hate Speech Detection,** submitted by **T Meena (R170572), Prawal Singh Baghel (R170063), P Gnana Sai (R170458)** here by accord my approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfillment for the award of Bachelor of Technology degree for which it has been submitted. This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusions drawn, as recorded in this thesis. It only signifies the acceptance of this thesis for the purpose for which it has been submitted.

EXAMINER

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(A.P. Government Act 18 of 2008)

RGUKT-RK Valley

Vempalli, Kadapa, Andhrapradesh-516330.

CERTIFICATE OF PROJECT COMPLETION

This is to certify that I have examined the thesis entitled submitted by **T Meena (R170572), Prawal Singh Baghel (R170063), P Gnana Sai (R170458)** under our guidance and supervision for the partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering during the academic session September 2022 – April 2023 at RGUKT-RK Valley.To the best of my knowledge, the results embodied in this dissertation work have not been submitted to any university or institute for the award of any degree or diploma.

------------------------------------	------------------------------------
Project Internal Guide	**Head of the Department**
Ms C.Suneetha	Mr.N.Satyanandaram
Assistant Professor,	HOD Of CSE,
RGUKT, RK Valley.	RGUKT, RK Valley.

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES
(A.P.Government Act 18 of 2008)
RGUKT-RK Valley
Vempalli, Kadapa, Andhrapradesh-516330.

DECLARATION

We , T Meena (R170572), Prawal Singh Baghel (R170063), P Gnana Sai (R170458) hereby declare that the project report entitled "Hate Speech Detection" done by is under guidance of Ms C. Suneetha is submitted in partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering during the academic session September 2022 – April 2023 at RGUKT-RK Valley. I also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites are mentioned in the references. To the best of my knowledge, the results embodied in this dissertation work have not been submitted to any university or institute for the award of any degree or diploma.

T Meena (R170572)
Prawal Singh Baghel (R170063)
P Gnana Sai (R170458)

ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude & respect to all those people behind the screen who guided, inspired and helped us crown all our efforts with success. We wish to express our gratitude to Ms. C. Suneetha for her valuable guidance at all stages of study, advice, constructive suggestions, supportive attitude and continuous encouragement, without which it would not be possible to complete this project.

We would also like to extend our deepest gratitude & reverence to the Director of RGUKT, RK Valley **Prof. K. Sandyarani** and HOD of Computer Science and Engineering **Mr.N.Satyanandaram** for their constant support and encouragement.

Last but not least we express my gratitude to my parents for their constant source of encouragement and inspiration for me to keep my morals high.

Table of Contents PageNo.

Abstract

This project uses a decision tree classifier to classify tweets as hate speech, offensive language, or no hate and offensive content. The tweets are first preprocessed by removing stop words, punctuations, and hyperlinks. The remaining words are then stemmed using the Snowball Stemmer algorithm. A count vectorizer is used to create a matrix of word frequencies, which is then fed into the decision tree classifier for training and testing. A lot of methods have already been created for the automation of hate speech detection online. There are two elements to this process: identifying the qualities that these terms utilize to target a certain group and classifying textual material as hate or non-hate speech. Due to time restraints, research efforts are initiated on the latter issue in this project. For this reason, detecting hate speech is a more challenging endeavor, as our research of the language used in typical datasets reveals that hate speech lacks distinctive, discriminatory characteristics.

Introduction

Social media has become an integral part of our lives, providing us with a platform to share our thoughts and ideas. However, with the increased usage of social media, there has been an increase in the instances of hate speech and offensive language, leading to cyberbullying and online harassment. This has become a major concern for online communities, social media platforms, and law enforcement agencies. Hate speech is defined as any speech, written or spoken, that attacks a person or a group of people based on their race, ethnicity, religion, gender, sexual orientation, or any other characteristic. It is a form of verbal violence that can cause emotional distress, psychological harm, and even physical harm.

Social media platforms like Twitter have become an important part of our lives. They allow us to connect with people from all around the world and share our thoughts and ideas. However, with the increasing use of social media, hate speech and offensive language have become more prevalent. This project aims to address this issue by detecting such content in tweets.

Existing System

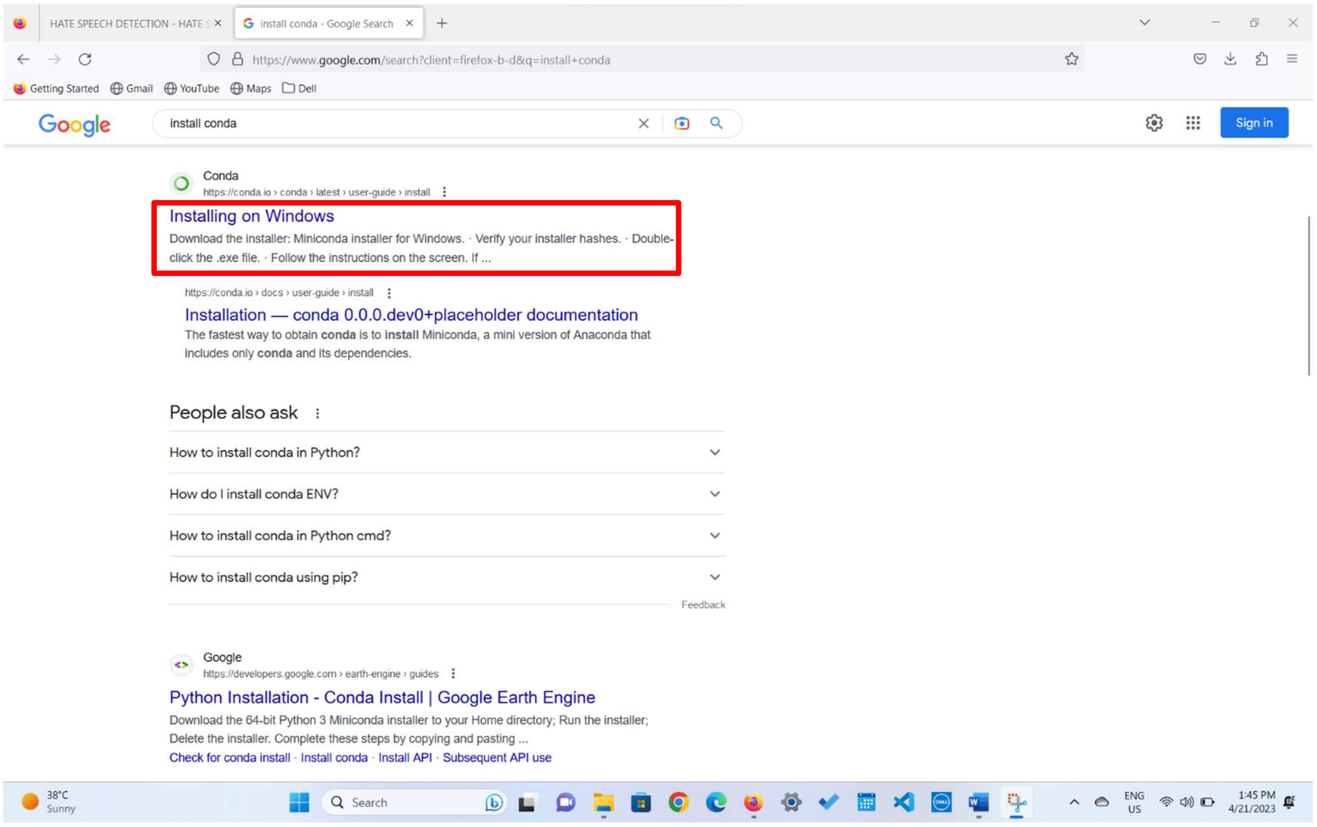
Existing systems for detecting hate speech and offensive language in tweets mainly rely on rule-based approaches. These approaches have limitations in terms of accuracy and scalability. Machine learning techniques have been shown to be more effective in detecting such content.

Proposed System

The proposed system uses a decision tree classifier to classify tweets as hate speech, offensive language, or no hate and offensive content. The tweets are preprocessed by removing stop words, punctuations, and hyperlinks. The remaining words are stemmed using the Snowball Stemmer algorithm. A count vectorizer is used to create a matrix of word frequencies, which is then fed into the decision tree classifier for training and testing.

INSTALLING ANACONDA

Step 1: search in google using search bar **“install co**


HATE SPEECH DETECTION - HATE S
install conda - Google Search
https://www.google.com/search?client=firefox-b-d&q=install+conda
Getting Started
Gmail
YouTube
Maps
Dell
Google
install conda
Sign in

Conda
https://conda.io › conda › latest › user-guide › install

Installing on Windows

Download the installer: Miniconda installer for Windows. · Verify your installer hashes. · Double-click the .exe file. · Follow the instructions on the screen. If ...

https://conda.io › docs › user-guide › install

Installation — conda 0.0.0.dev0+placeholder documentation

The fastest way to obtain **conda** is to **install** Miniconda, a mini version of Anaconda that includes only **conda** and its dependencies.

People also ask

How to install conda in Python?

How do I install conda ENV?

How to install conda in Python cmd?

How to install conda using pip?

Feedback

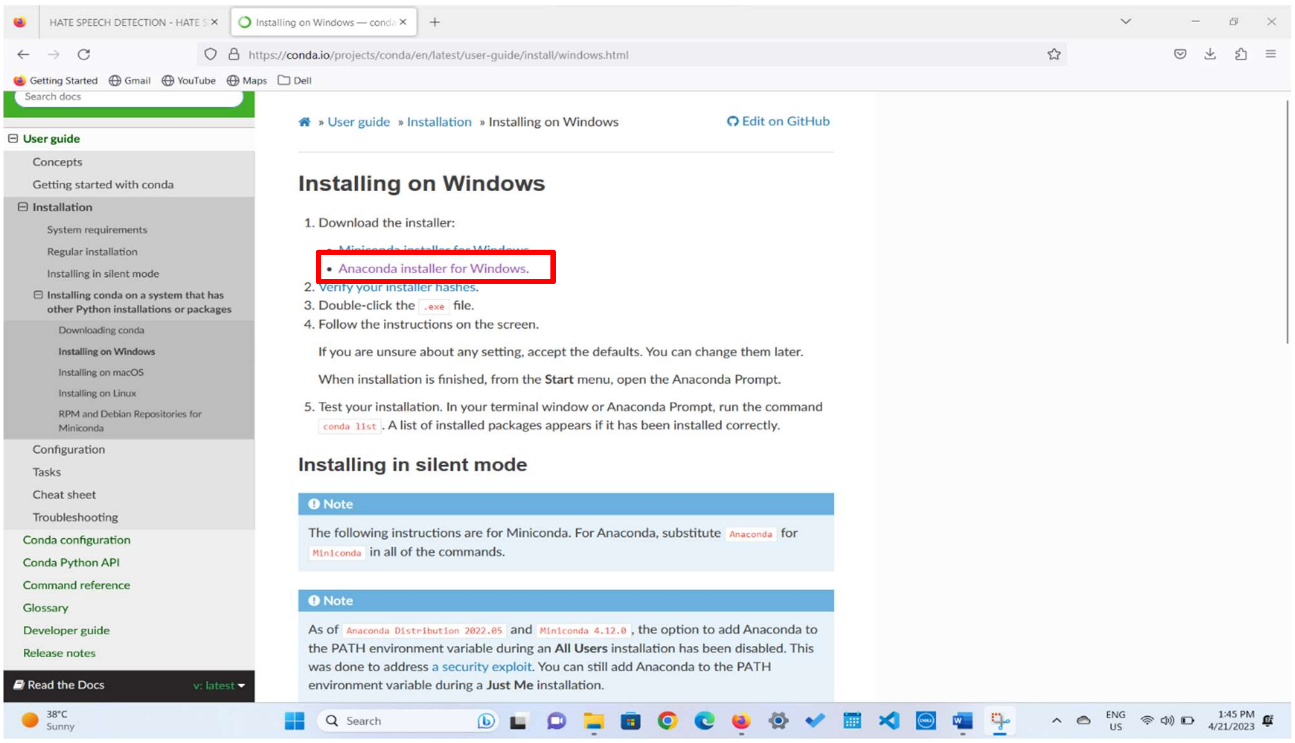
Google
https://developers.google.com › earth-engine › guides

Python Installation - Conda Install | Google Earth Engine

Download the 64-bit Python 3 Miniconda installer to your Home directory; Run the installer; Delete the installer. Complete these steps by copying and pasting ...

Check for conda install · Install conda · Install API · Subsequent API use

Step 2: click on **“installing on windows”**



Installing on Windows

1. Download the installer:
   - Miniconda installer for Windows.
   - Anaconda installer for Windows.
2. Verify your installer hashes.
3. Double-click the `.exe` file.
4. Follow the instructions on the screen.

   If you are unsure about any setting, accept the defaults. You can change them later.

   When installation is finished, from the **Start** menu, open the Anaconda Prompt.
5. Test your installation. In your terminal window or Anaconda Prompt, run the command `conda list`. A list of installed packages appears if it has been installed correctly.

Installing in silent mode

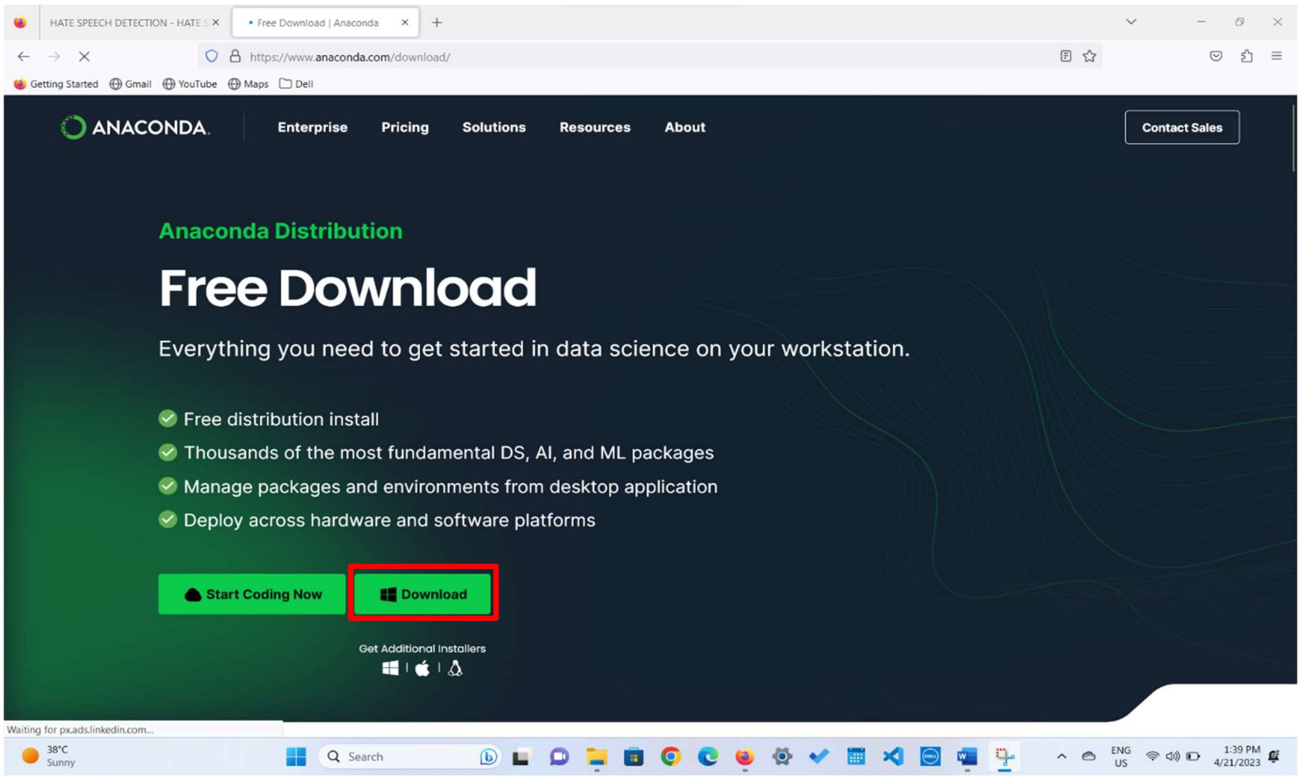
Note

The following instructions are for Miniconda. For Anaconda, substitute `Anaconda` for `Miniconda` in all of the commands.

Note

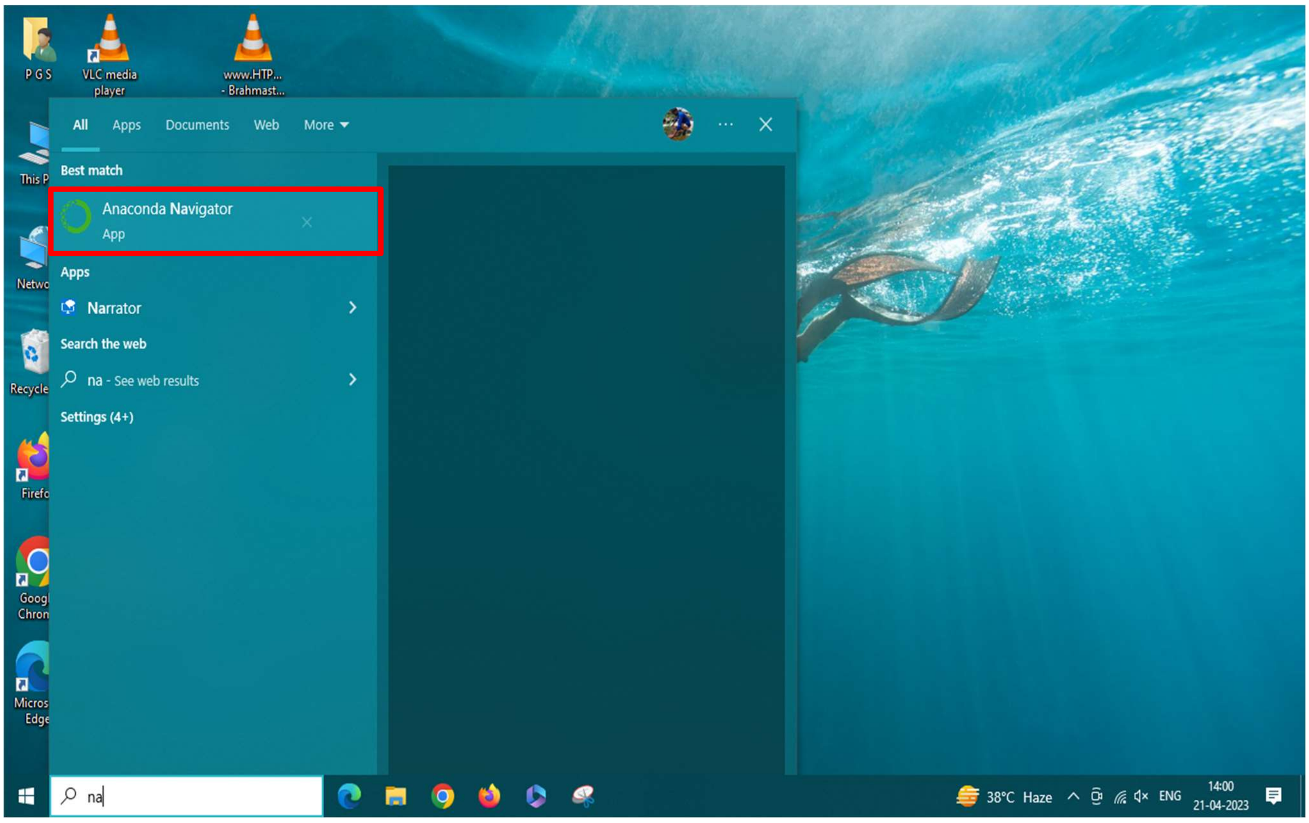
As of `Anaconda Distribution 2022.05` and `Miniconda 4.12.0`, the option to add Anaconda to the PATH environment variable during an **All Users** installation has been disabled. This was done to address a security exploit. You can still add Anaconda to the PATH environment variable during a **Just Me** installation.

Step 3:click on “**Anaconda installer for windows**”

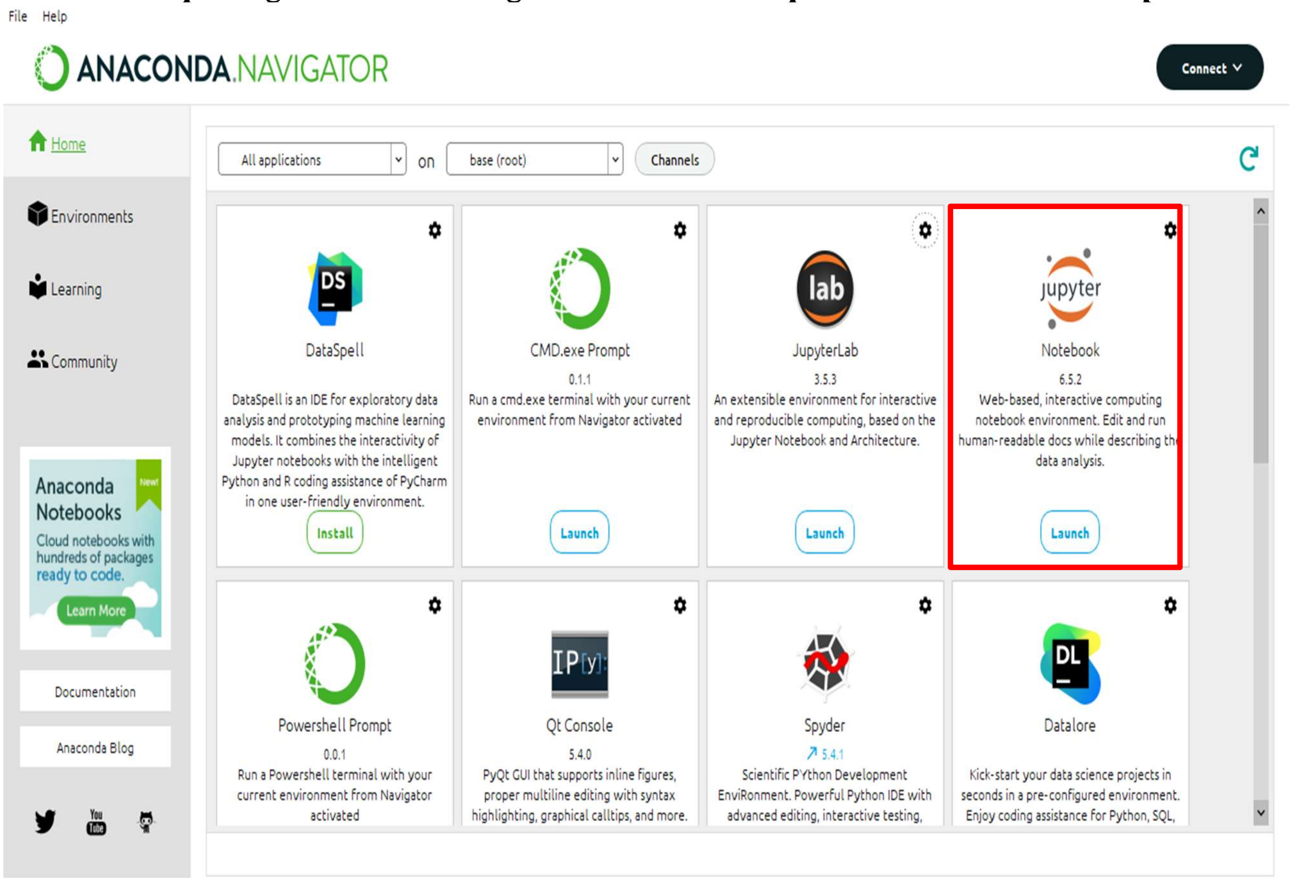

ANACONDA
Enterprise
Pricing
Solutions
Resources
About
Contact Sales
Anaconda Distribution
Free Download
Everything you need to get started in data science on your workstation.
Free distribution install
Thousands of the most fundamental DS, AI, and ML packages
Manage packages and environments from desktop application
Deploy across hardware and software platforms
Start Coding Now
Download
Get Additional Installers

Step 4:click on “download” Now anaconda is downloaded

➢ **After downloading the Anaconda, Open “anaconda Navigator” in your search bar**



➢ **After opening anaconda navigator search for Jupiter and launch the Jupiter**


File Help

ANACONDA.NAVIGATOR

Connect

Home | Environments | Learning | Community

All applications on base (root) Channels

DataSpell

DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

Install

CMD.exe Prompt

0.1.1

Run a cmd.exe terminal with your current environment from Navigator activated

Launch

JupyterLab

3.5.3

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch

Notebook

6.5.2

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch

Powershell Prompt

0.0.1

Run a Powershell terminal with your current environment from Navigator activated

Qt Console

5.4.0

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder

5.4.1

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing,

Datalore

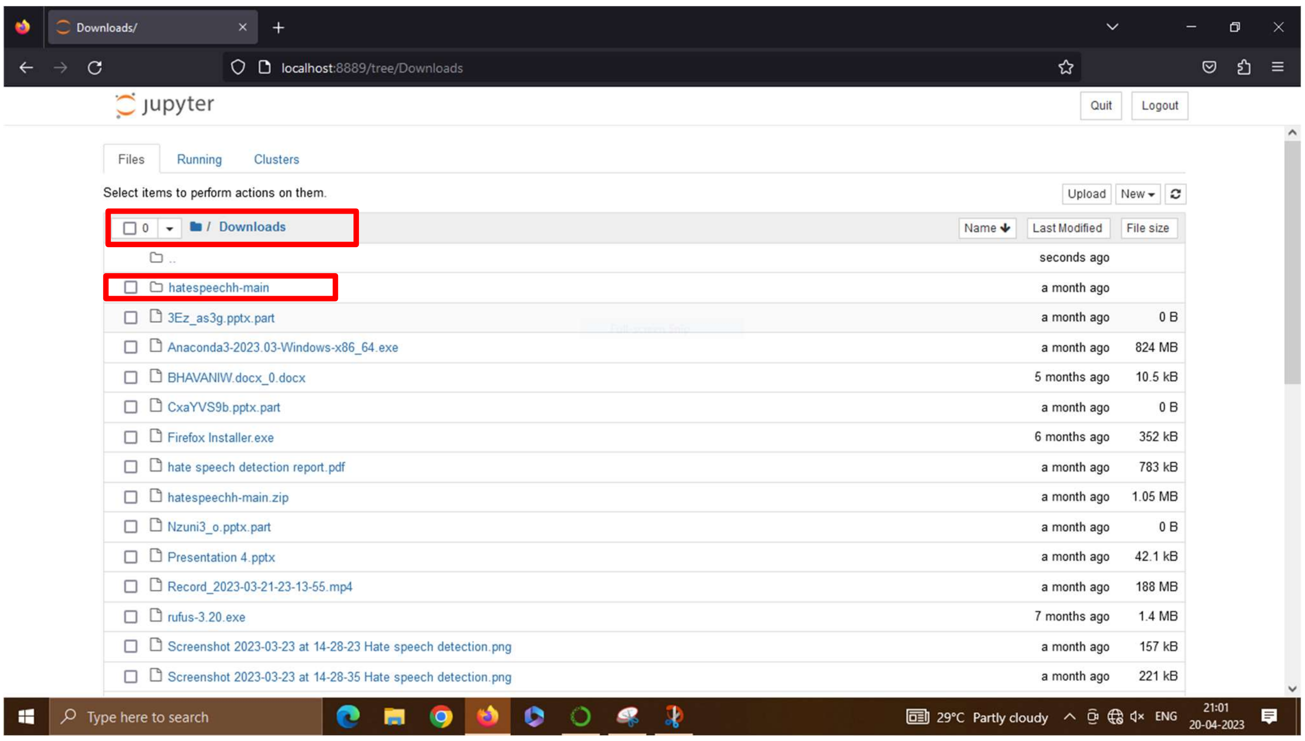
Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL,

Anaconda Notebooks — Cloud notebooks with hundreds of packages ready to code. Learn More

Documentation

Anaconda Blog

➢ **Now Jupiter is launched, Select location of your file**


Downloads/

localhost:8889/tree/Downloads

jupyter — Quit Logout

Files Running Clusters

Select items to perform actions on them.

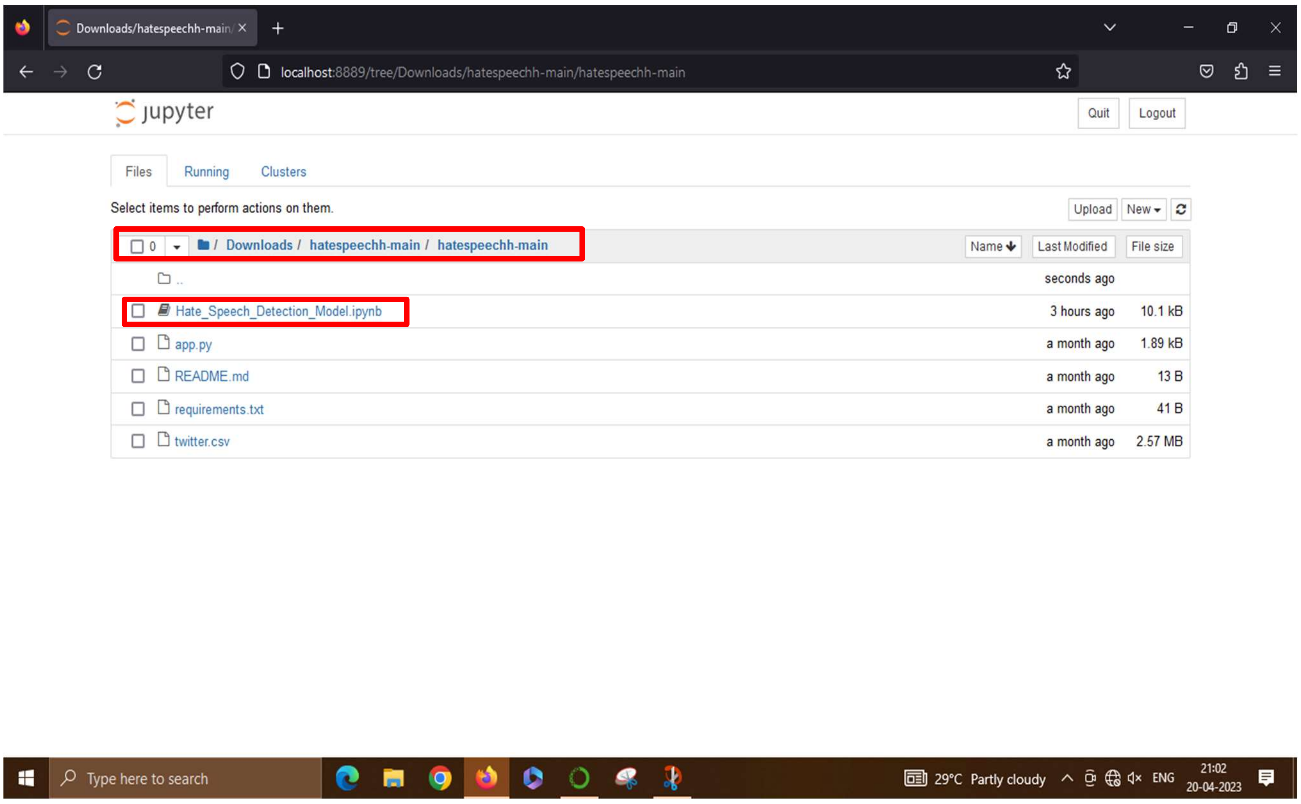
Upload New

0 / Downloads — Name Last Modified File size

Name	Last Modified	File size
..	seconds ago	
hatespeechh-main	a month ago	
3Ez_as3g.pptx.part	a month ago	0 B
Anaconda3-2023.03-Windows-x86_64.exe	a month ago	824 MB
BHAVANIW.docx_0.docx	5 months ago	10.5 kB
CxaYVS9b.pptx.part	a month ago	0 B
Firefox Installer.exe	6 months ago	352 kB
hate speech detection report.pdf	a month ago	783 kB
hatespeechh-main.zip	a month ago	1.05 MB
Nzuni3_o.pptx.part	a month ago	0 B
Presentation 4.pptx	a month ago	42.1 kB
Record_2023-03-21-23-13-55.mp4	a month ago	188 MB
rufus-3.20.exe	7 months ago	1.4 MB
Screenshot 2023-03-23 at 14-28-23 Hate speech detection.png	a month ago	157 kB
Screenshot 2023-03-23 at 14-28-35 Hate speech detection.png	a month ago	221 kB

Type here to search — 29°C Partly cloudy ENG 21:01 20-04-2023

➢ **After choosing location select the source code.**

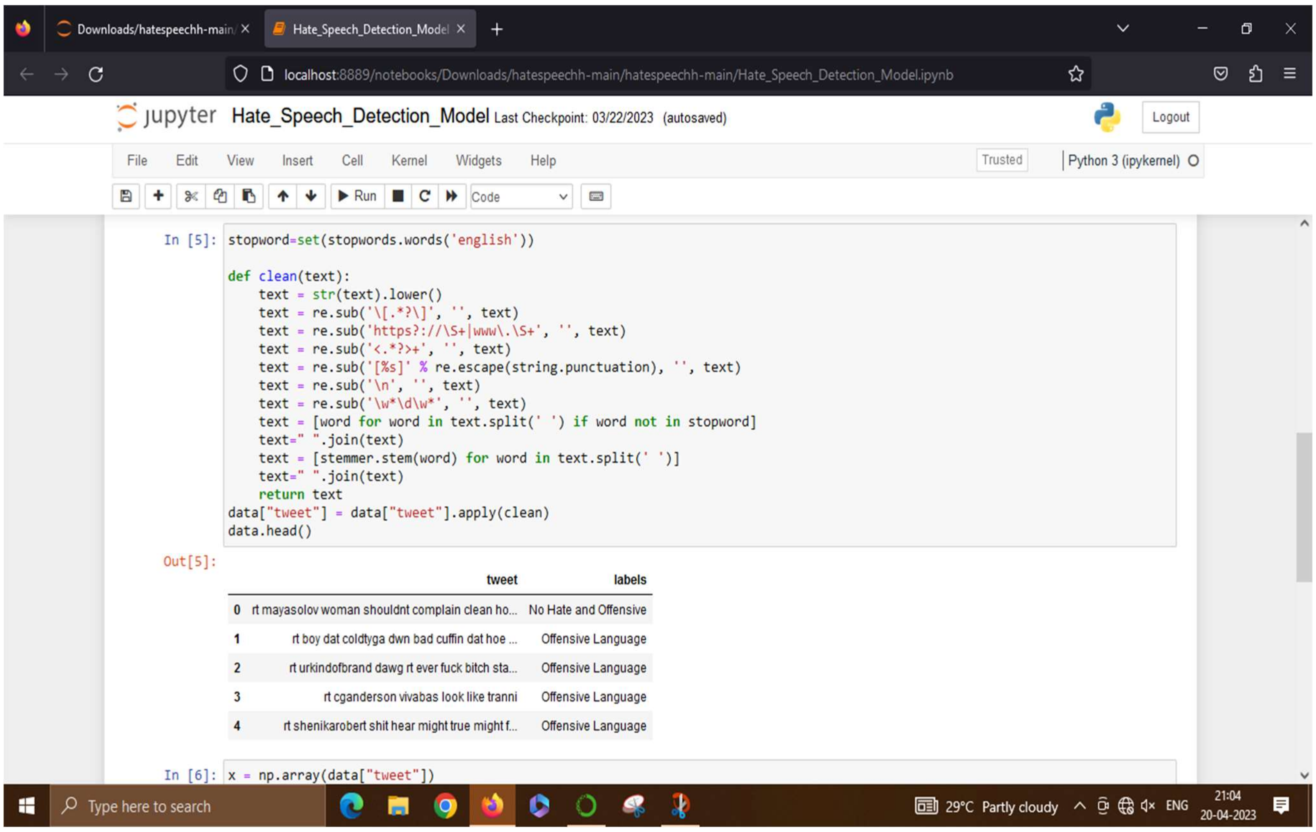

localhost:8889/tree/Downloads/hatespeechh-main/hatespeechh-main

Select items to perform actions on them.

/ Downloads / hatespeechh-main / hatespeechh-main

Name	Last Modified	File size
..	seconds ago	
Hate_Speech_Detection_Model.ipynb	3 hours ago	10.1 kB
app.py	a month ago	1.89 kB
README.md	a month ago	13 B
requirements.txt	a month ago	41 B
twitter.csv	a month ago	2.57 MB

➢ **After that we get like shown in below picture**


localhost:8889/notebooks/Downloads/hatespeechh-main/hatespeechh-main/Hate_Speech_Detection_Model.ipynb

Hate_Speech_Detection_Model Last Checkpoint: 03/22/2023 (autosaved)

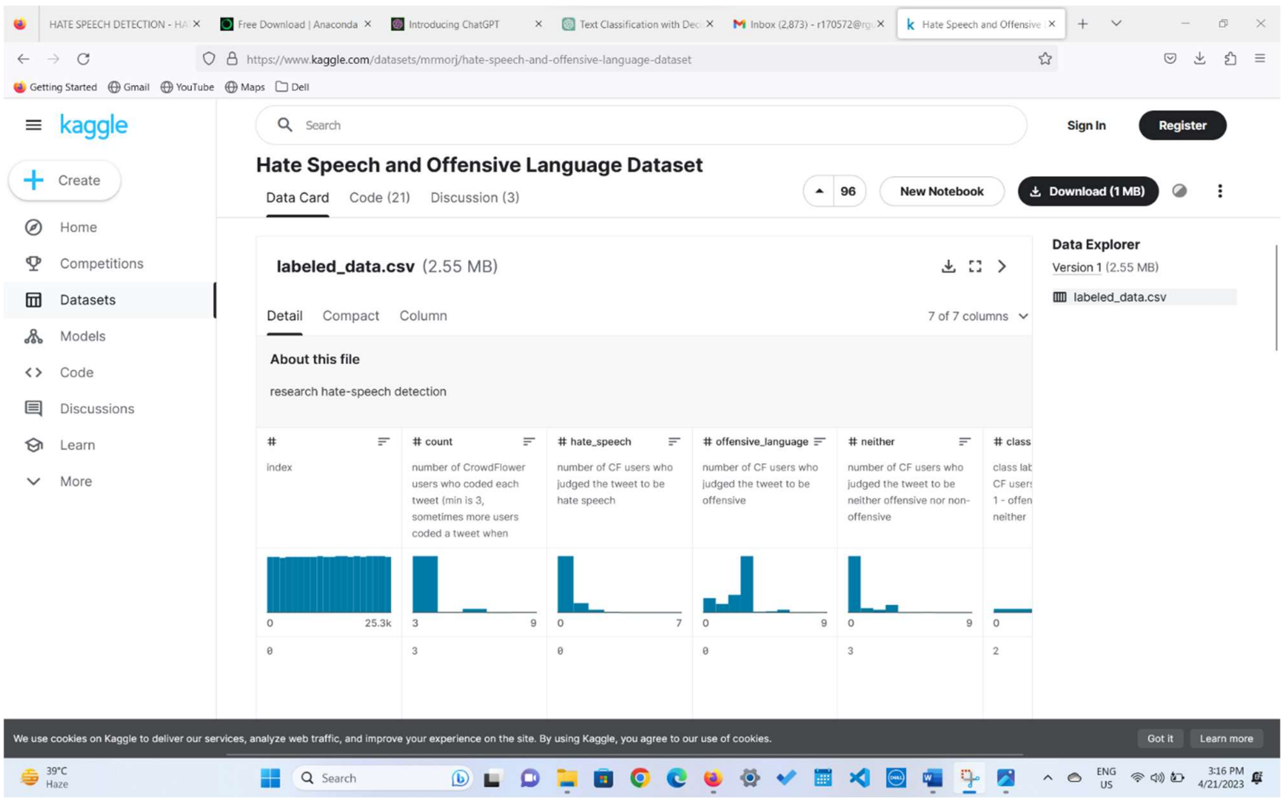
```
In [5]: stopword=set(stopwords.words('english'))

        def clean(text):
            text = str(text).lower()
            text = re.sub('\[.*?\]', '', text)
            text = re.sub('https?://\S+|www\.\S+', '', text)
            text = re.sub('<.*?>+', '', text)
            text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
            text = re.sub('\n', '', text)
            text = re.sub('\w*\d\w*', '', text)
            text = [word for word in text.split(' ') if word not in stopword]
            text=" ".join(text)
            text = [stemmer.stem(word) for word in text.split(' ')]
            text=" ".join(text)
            return text
        data["tweet"] = data["tweet"].apply(clean)
        data.head()
```

Out[5]:

	tweet	labels
0	rt mayasolov woman shouldnt complain clean ho...	No Hate and Offensive
1	rt boy dat coldtyga dwn bad cuffin dat hoe ...	Offensive Language
2	rt urkindofbrand dawg rt ever fuck bitch sta...	Offensive Language
3	rt cganderson vivabas look like tranni	Offensive Language
4	rt shenikarobert shit hear might true might f...	Offensive Language

```
In [6]: x = np.array(data["tweet"])
```

For our project we get twitter dataset from Kaggle website
https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset


Hate Speech and Offensive Language Dataset

Data Card · Code (21) · Discussion (3)

labeled_data.csv (2.55 MB)

Detail · Compact · Column — 7 of 7 columns

About this file

research hate-speech detection

#	# count	# hate_speech	# offensive_language	# neither	# class
index	number of CrowdFlower users who coded each tweet (min is 3, sometimes more users coded a tweet when	number of CF users who judged the tweet to be hate speech	number of CF users who judged the tweet to be offensive	number of CF users who judged the tweet to be neither offensive nor non-offensive	class lab CF users 1 - offen neither
0 – 25.3k	3 – 9	0 – 7	0 – 9	0 – 9	0
0	3	0	0	3	2

Data Explorer
Version 1 (2.55 MB)
labeled_data.csv

We use cookies on Kaggle to deliver our services, analyze web traffic, and improve your experience on the site. By using Kaggle, you agree to our use of cookies.

LIBRARIES

- PYTHON PANDAS
- NUMPY
- Scikit learn
- NLTK (Natural Language Toolkit)

PANDAS

- Pandas is a fast, powerful, flexible and easy to use opensource data analysis and manipulation tool, built on top of the programming language.
- Pandas is a Python library used for working with data sets. It has functions for analysing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
- Pandas are very popular because of their powerful features. There are so many useful features commonly used for data science projects.
- With Pandas, we can perform filtering on condition, sorting, and indexing on fingertips with only single lines of code.

NUMPY

- NumPy stands for Numerical Python.
- NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy was created in 2005 by TravisOliphant. It is an opensource project and you can use it freely.
- NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and its applies an enormous library of high-level mathematical

SCIKIT LEARN

Scikit Learn , also known as sklearn is a python library to implement machine learning models and statistical modelling. Through scikit learn , we can implement various machine learning models for regression , classification ,clustering ,and statistical tools for analysing these models. Sklearn provided the functionality to split the data set for training and testing. Splitting the dataset is essential for an unbiased evaluation of prediction performance. We can define what proportion of our data to be included in train and test datasets.

Features: Data Splitting,
Decision Trees,
Random Forest,
SVM (support vector machine),
Linear Regression

NLTK

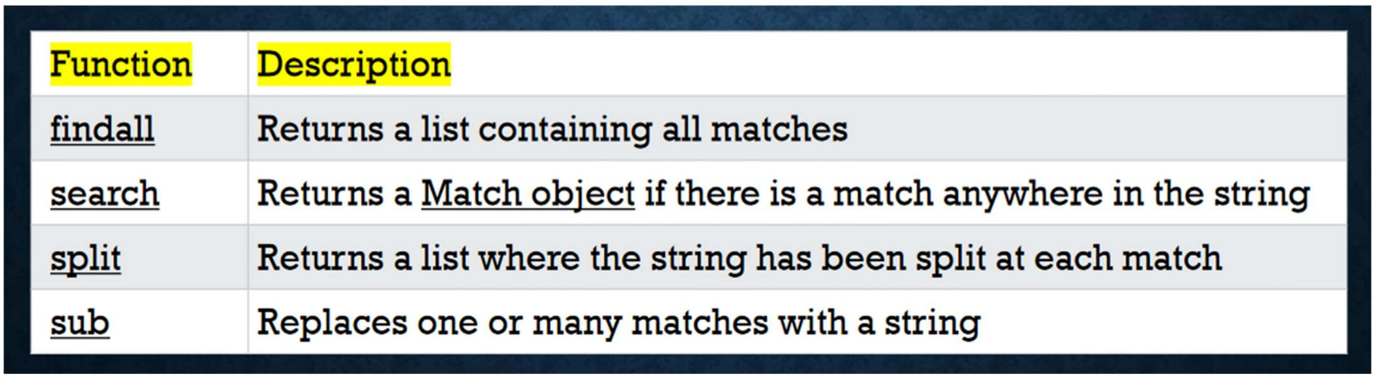
NLTK (Natural Language Toolkit) is a Python library that provides a comprehensive set of tools and resources for working with natural language data. It is widely used by researchers, educators, and industry professionals to perform a variety of natural language processing (NLP) tasks, such as tokenization, stemming, part-of-speech tagging, and sentiment analysis. NLTK is an open-source library, which means that it is freely available for anyone to use and contribute .Its development is supported by a large community of users and contributors who continue to add new features and resources to the library.

MODULES

REGULAR EXPRESSION

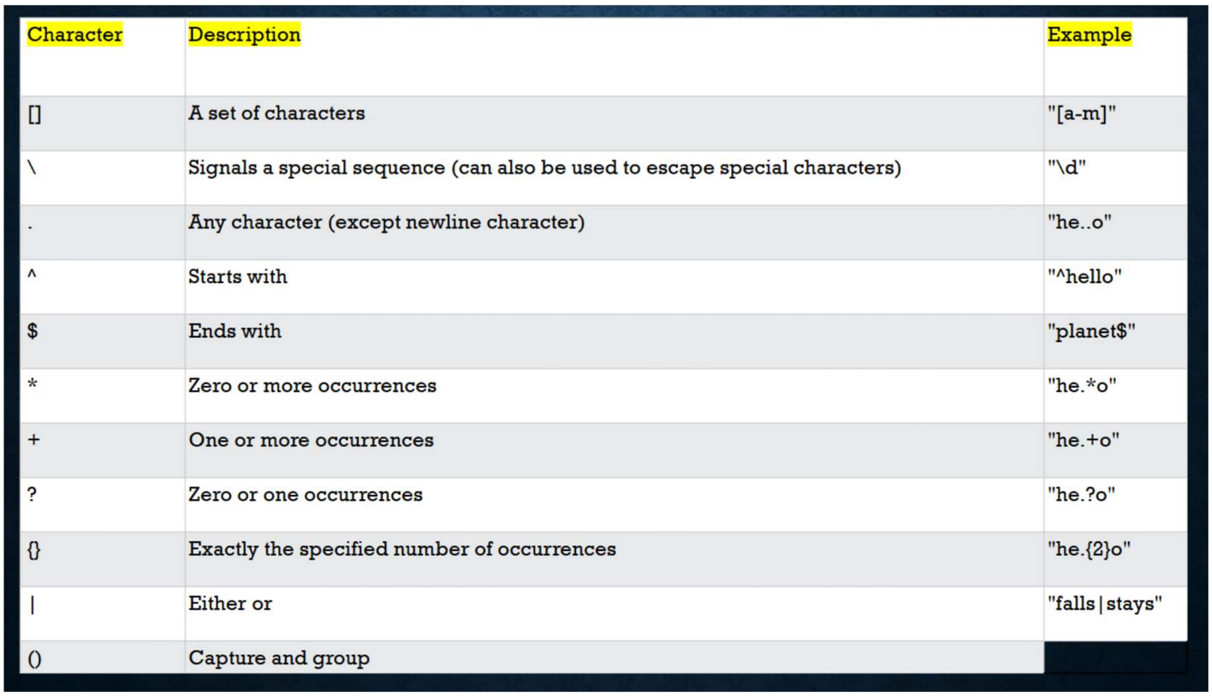
A regular expression (RE) specifies a set of strings that matches it ; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string , which comes down to the same thing).Regex(Regular Expression) is supported in all the scripting languages (such as Perl, Python, PHP, and JavaScript); as well as general purpose programming languages such as Java; and even word processors such as Word for searching texts. Getting started with regex may not be easy due to its geeky syntax, but it is certainly worth the investment of your time.

The re module offers a set of functions that allows us to search a string for a match:



Function	Description
findall	Returns a list containing all matches
search	Returns a Match object if there is a match anywhere in the string
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string

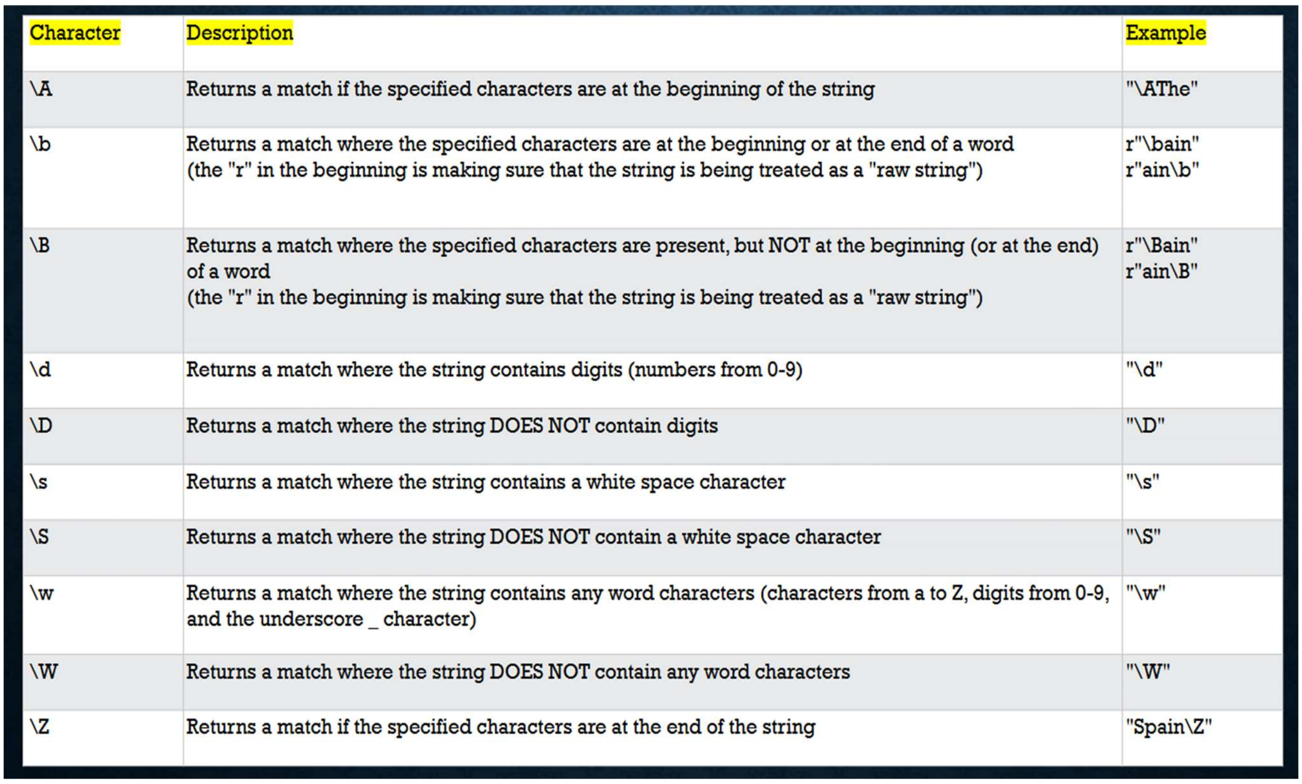
- **MetaCharacters**



Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"planet$"
*	Zero or more occurrences	"he.*o"
+	One or more occurrences	"he.+o"
?	Zero or one occurrences	"he.?o"
{}	Exactly the specified number of occurrences	"he.{2}o"
|	Either or	"falls|stays"
()	Capture and group	

- **Special Sequences**

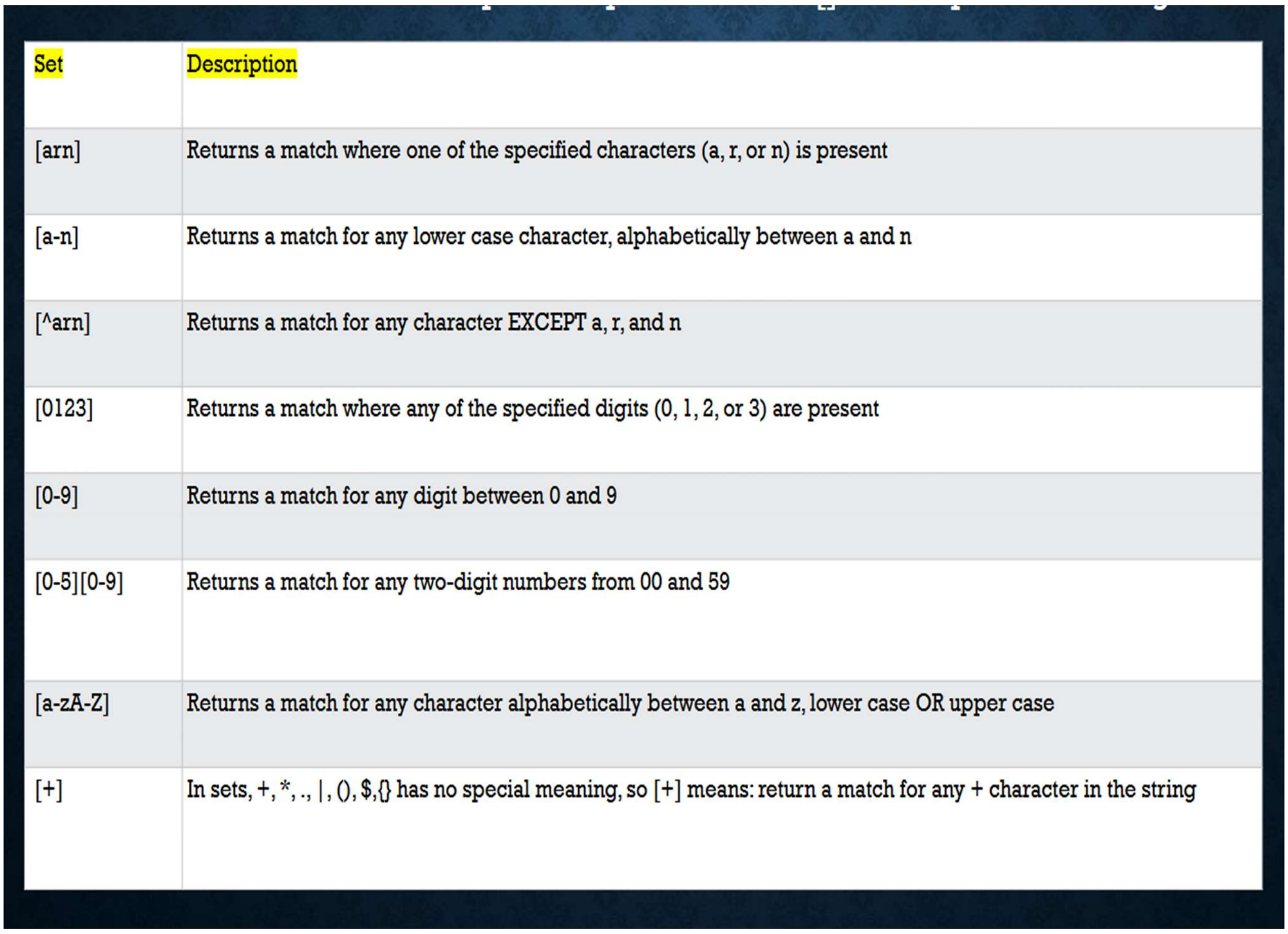
A special sequence is a\followed by one of the characters in the List below, and has a special meaning:



Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"
\b	Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\Bain" r"ain\B"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"

- **Sets**

A set is a set of characters in side a pair of square brackets [ ] with a special meaning:

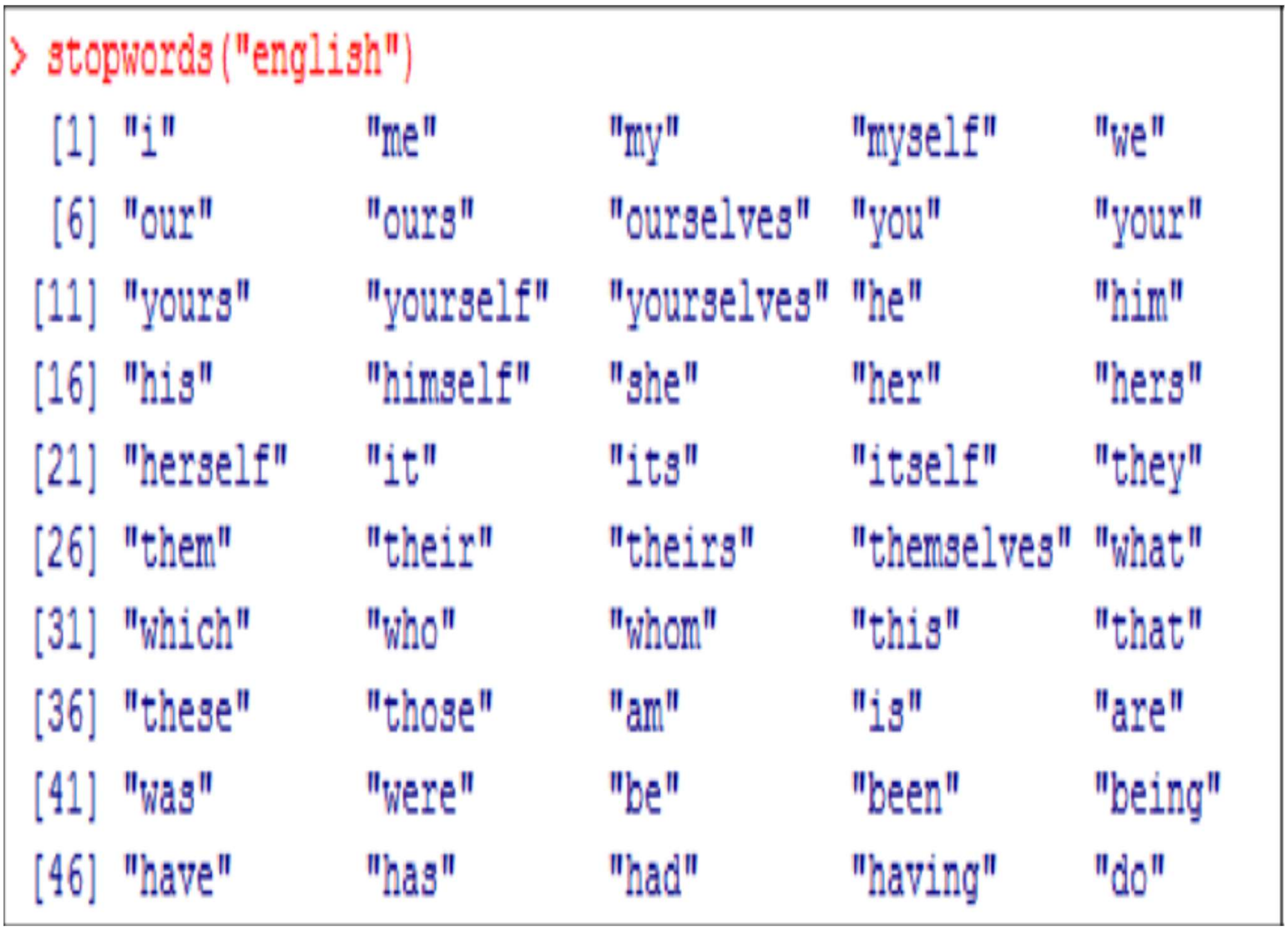


Set	Description
[arn]	Returns a match where one of the specified characters (a, r, or n) is present
[a-n]	Returns a match for any lower case character, alphabetically between a and n
[^arn]	Returns a match for any character EXCEPT a, r, and n
[0123]	Returns a match where any of the specified digits (0, 1, 2, or 3) are present
[0-9]	Returns a match for any digit between 0 and 9
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a and z, lower case OR upper case
[+]	In sets, +, *, ., |, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string

STOPWORDS

stopwords is a module in some programming languages and libraries, such as NLTK(Natural Language Toolkit) for Python. In NLTK, the stop words module contains a list of commonly used words that are often removed from texts during natural language processing tasks, such as text classification or sentiment analysis. These words are considered "stopwords" because they are usually not useful for these tasks and may even introduce noise in to the analysis

Other programming languages or libraries may have similar modules or functions for dealing with stopwords, but the specific implementation and usage may differ.



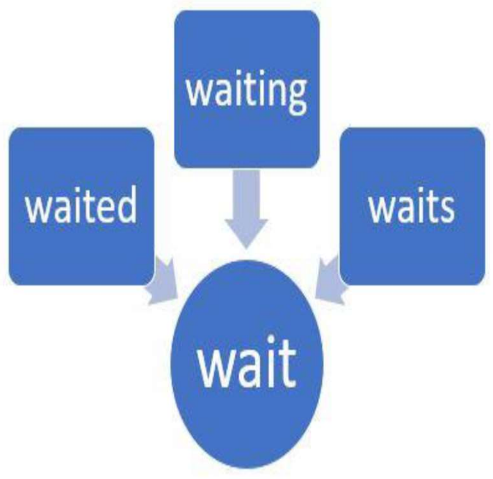
```
> stopwords("english")
 [1] "i"          "me"         "my"         "myself"     "we"
 [6] "our"        "ours"       "ourselves"  "you"        "your"
[11] "yours"      "yourself"   "yourselves" "he"         "him"
[16] "his"        "himself"    "she"        "her"        "hers"
[21] "herself"    "it"         "its"        "itself"     "they"
[26] "them"       "their"      "theirs"     "themselves" "what"
[31] "which"      "who"        "whom"       "this"       "that"
[36] "these"      "those"      "am"         "is"         "are"
[41] "was"        "were"       "be"         "been"       "being"
[46] "have"       "has"        "had"        "having"     "do"
```

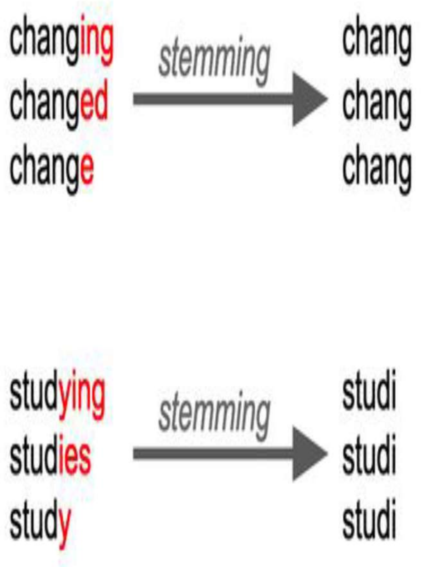
STEMMING

Stemming is the process of reducing a word to its stem that affixes to suffixes and prefixes or to the roots of words Known as " lemmas ". Stemming is important in natural language understanding (NLU) and natural language processing (NLP). The Snowball algorithm is a popular stemming algorithm that uses a set of rules for each language to convert words to their root form. The English version of the Snowball algorithm uses a set of rules based on linguistic heuristics and statistical observations to generate the stem of a word.

To use tis stemmer in NLTK, you can create an instance of the snowball stemmer class and pass in the language as a parameter.

Examples for stemming:

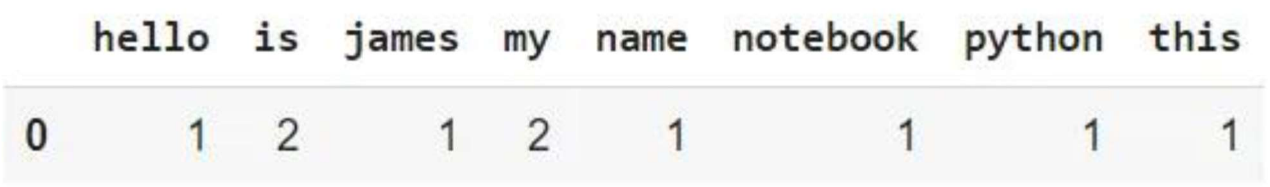




Methods:

Countvectorizer()

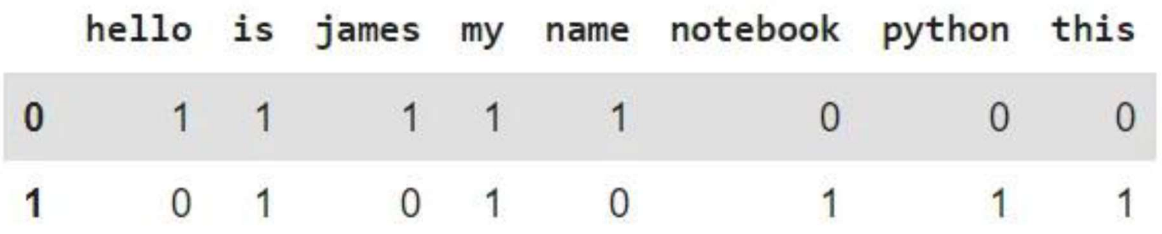
Countvectorizer is a method to convert text to numerical data. example: text = ['Hello my name is James, this is my python notebook'] The text is transformed to a sparse matrix as shown below.



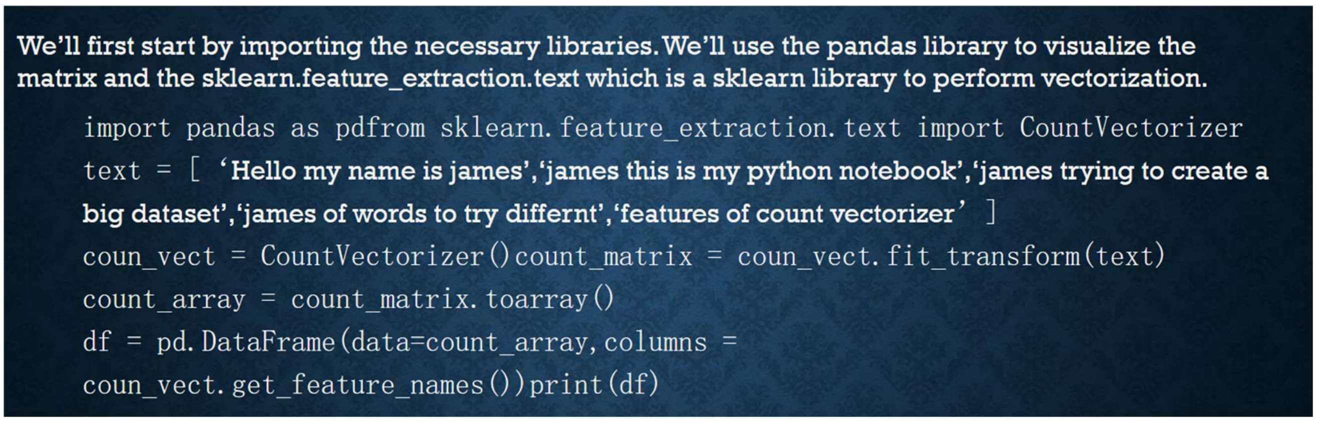
	hello	is	james	my	name	notebook	python	this
0	1	2	1	2	1	1	1	1

We have 8 unique words in the text and hence 8 different column search representing a unique word in the matrix. The row represents the word count. Since the words 'is' and 'my' were repeated twice we have the count for those particular words as 2 and 1 for the rest

Countvectorizer makes it easy for text data to be used directly in machine learning and deep learning models such as text classification. Let's take another example, but this time with more than 1 input: Text = ['Hello my name is James ', ' this is my python notebook']



	hello	is	james	my	name	notebook	python	this
0	1	1	1	1	1	0	0	0
1	0	1	0	1	0	1	1	1



We'll first start by importing the necessary libraries. We'll use the pandas library to visualize the matrix and the sklearn.feature_extraction.text which is a sklearn library to perform vectorization.

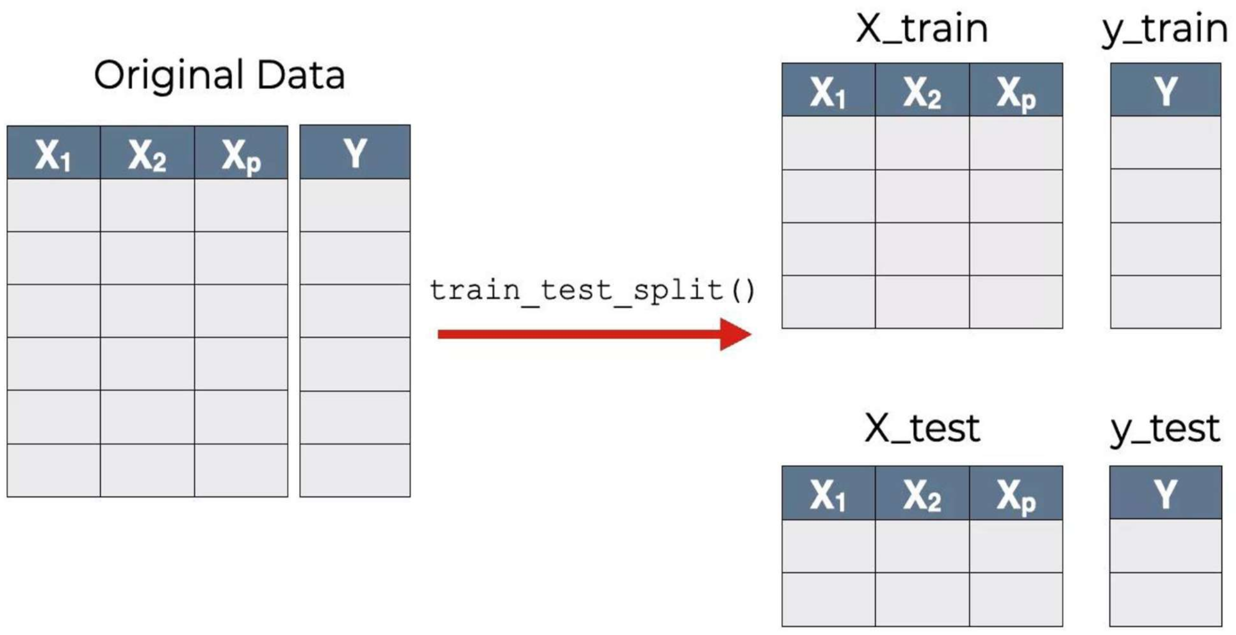
```
import pandas as pdfrom sklearn.feature_extraction.text import CountVectorizer
text = [ 'Hello my name is james','james this is my python notebook','james trying to create a
big dataset','james of words to try differnt','features of count vectorizer' ]
coun_vect = CountVectorizer()count_matrix = coun_vect.fit_transform(text)
count_array = count_matrix.toarray()
df = pd.DataFrame(data=count_array,columns =
coun_vect.get_feature_names())print(df)
```

FIT_TRANSFORM():

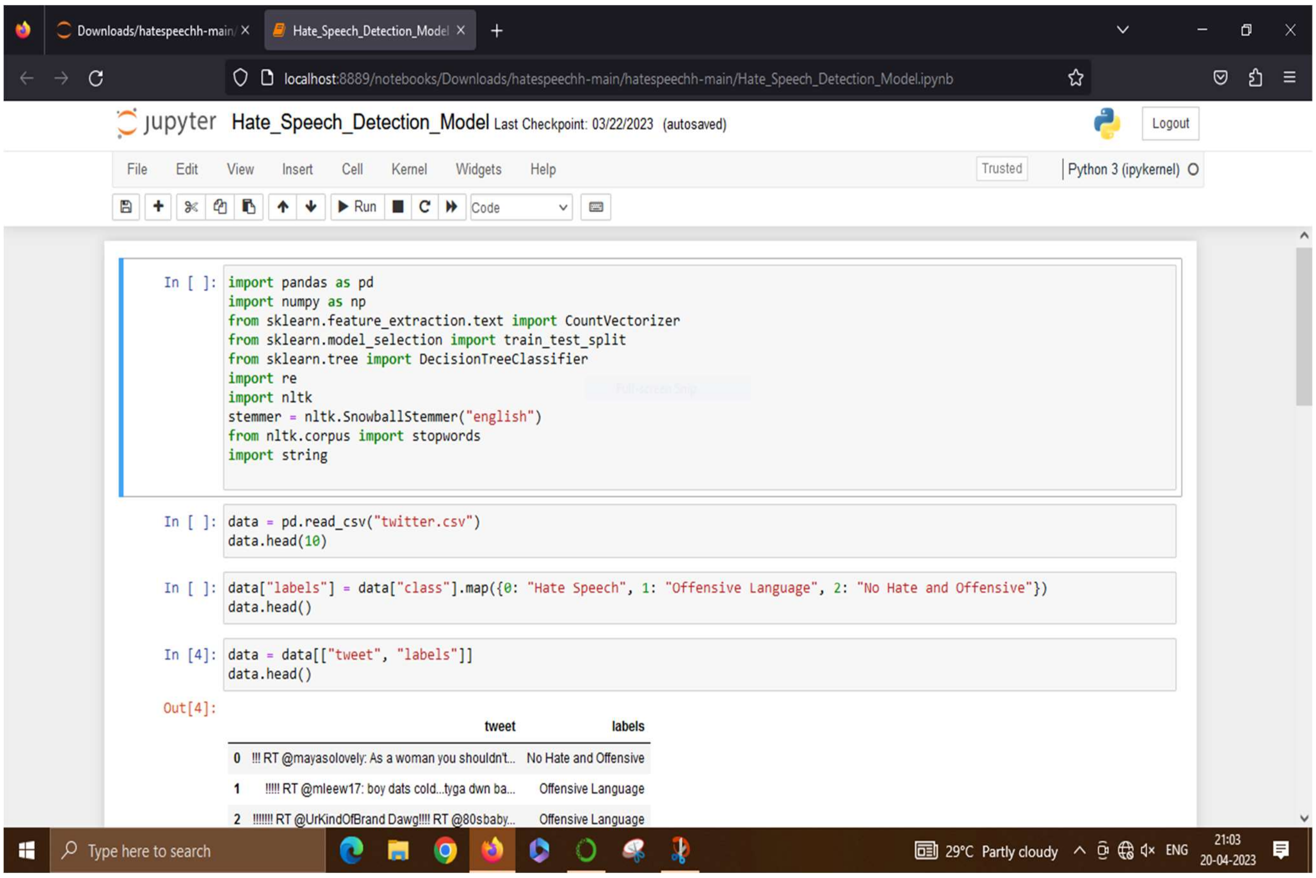
fit_transform() is a method commonly used in machine learning for data preprocessing, which is a combination of two methods: *fit()* and *transform()*.The *fit()* method is used to learn the parameters of a transformation from a training set, while the *transform()* method applies this learned transformation to a new dataset. The *fit_transform()* method first fits the transformation to the training data, and then applies it to the same training data to obtain the transformed data. This is often done as a single step to simplify the code.

TRAIN_TEST_SPLIT( )

A train test split is when you split your data into a training set an testing set. The training set is used for training the model , and the testing set is used to test your model . This allows you to train your models on the training set , and then test their accuracy on the unseen testing set.



Source code


Hate_Speech_Detection_Model Last Checkpoint: 03/22/2023 (autosaved)
Trusted
Python 3 (ipykernel)

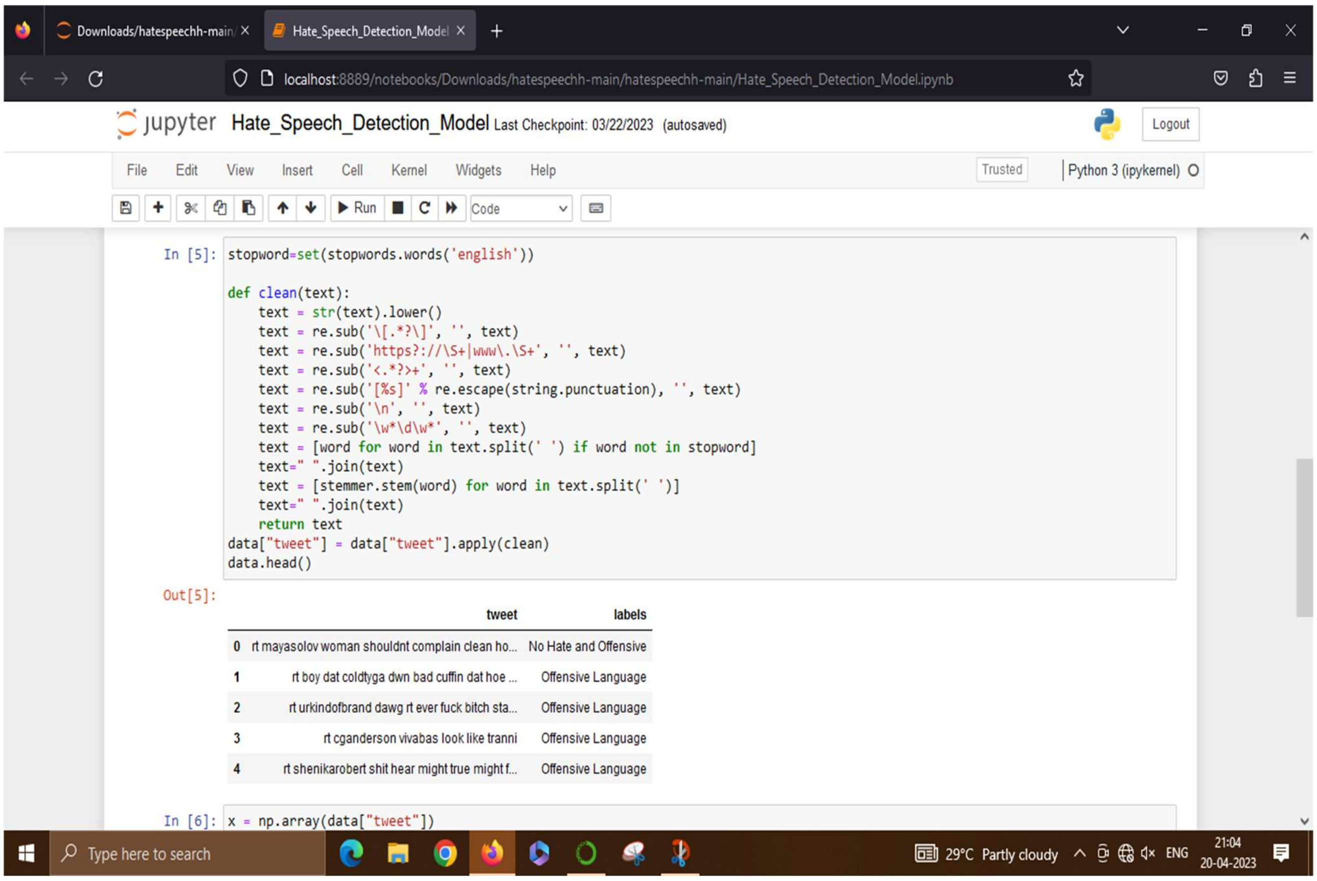
```
In [ ]: import pandas as pd
        import numpy as np
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeClassifier
        import re
        import nltk
        stemmer = nltk.SnowballStemmer("english")
        from nltk.corpus import stopwords
        import string
```

```
In [ ]: data = pd.read_csv("twitter.csv")
        data.head(10)
```

```
In [ ]: data["labels"] = data["class"].map({0: "Hate Speech", 1: "Offensive Language", 2: "No Hate and Offensive"})
        data.head()
```

```
In [4]: data = data[["tweet", "labels"]]
        data.head()
```

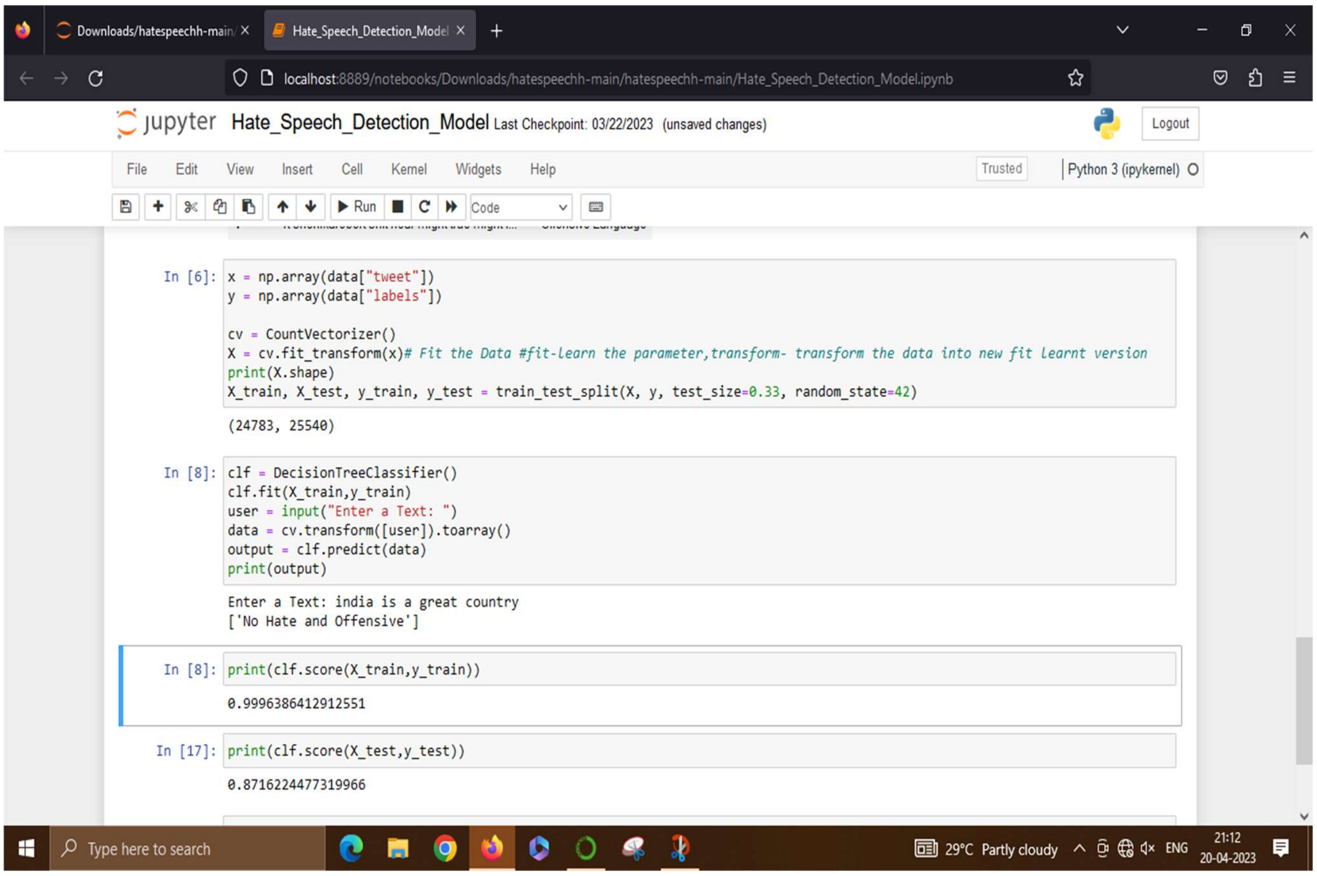
```
Out[4]:
                                           tweet                  labels
   0  !!! RT @mayasolovely: As a woman you shouldn't...  No Hate and Offensive
   1     !!!!! RT @mleew17: boy dats cold...tyga dwn ba...     Offensive Language
   2  !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...     Offensive Language
```


Hate_Speech_Detection_Model Last Checkpoint: 03/22/2023 (autosaved)
Trusted
Python 3 (ipykernel)

```
In [5]: stopword=set(stopwords.words('english'))

        def clean(text):
            text = str(text).lower()
            text = re.sub('\[.*?\]', '', text)
            text = re.sub('https?://\S+|www\.\S+', '', text)
            text = re.sub('<.*?>+', '', text)
            text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
            text = re.sub('\n', '', text)
            text = re.sub('\w*\d\w*', '', text)
            text = [word for word in text.split(' ') if word not in stopword]
            text=" ".join(text)
            text = [stemmer.stem(word) for word in text.split(' ')]
            text=" ".join(text)
            return text
        data["tweet"] = data["tweet"].apply(clean)
        data.head()
```

```
Out[5]:
                                        tweet                  labels
   0  rt mayasolov woman shouldnt complain clean ho...  No Hate and Offensive
   1          rt boy dat coldtyga dwn bad cuffin dat hoe ...     Offensive Language
   2     rt urkindofbrand dawg rt ever fuck bitch sta...     Offensive Language
   3              rt cganderson vivabas look like tranni     Offensive Language
   4     rt shenikarobert shit hear might true might f...     Offensive Language
```

```
In [6]: x = np.array(data["tweet"])
```


Hate_Speech_Detection_Model Last Checkpoint: 03/22/2023 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
localhost:8889/notebooks/Downloads/hatespeechh-main/hatespeechh-main/Hate_Speech_Detection_Model.ipynb

```
In [6]: x = np.array(data["tweet"])
        y = np.array(data["labels"])

        cv = CountVectorizer()
        X = cv.fit_transform(x)# Fit the Data #fit-Learn the parameter,transform- transform the data into new fit Learnt version
        print(X.shape)
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

        (24783, 25540)

In [8]: clf = DecisionTreeClassifier()
        clf.fit(X_train,y_train)
        user = input("Enter a Text: ")
        data = cv.transform([user]).toarray()
        output = clf.predict(data)
        print(output)

        Enter a Text: india is a great country
        ['No Hate and Offensive']

In [8]: print(clf.score(X_train,y_train))

        0.9996386412912551

In [17]: print(clf.score(X_test,y_test))

        0.8716224477319966
```

In-detail source code

`import pandas as pd`

This line imports the Pandas library, which provides data structures and tools for data analysis.

‘import numpy as np`

This line imports the NumPy library, which provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions.

`from sklearn.feature_extraction.text import CountVectorizer`

This line imports the CountVectorizer class from the scikit-learn (sklearn) library, which is used for converting text data into numerical features that can be used in machine learning algorithms.

`from sklearn.model_selection import train_test_split`

This line imports the train_test_split function from the sklearn library, which is used for splitting a dataset into training and testing sets.

`from sklearn.tree import DecisionTreeClassifier`

This line imports the DecisionTreeClassifier class from the sklearn library, which is a machine learning algorithm used for classification tasks.

`import re`

This line imports the re module, which provides support for regular expressions.

`import nltk`

This line imports the Natural Language Toolkit (nltk) library, which is a suite of libraries and programs for natural language processing (NLP).

`stemmer = nltk.SnowballStemmer("english")`

This line creates an instance of the Snowball Stemmer class from the nltk library, which is used for stemming words in natural language text.

`from nltk.corpus import stopwords`

This line imports the stopwords corpus from the nltk library, which is a collection of common words that are often removed from text data during NLP tasks.

`import string`

This line imports the string module, which provides support for working with strings.

`data = pd.read_csv("twitter.csv")`

This line reads in a CSV file named "twitter.csv" using the Pandas read_csv function and assigns the resulting Data Frame to a variable named "data".

`data.head()`

This line prints the first five rows of the "data" Data Frame to the console.

`data["labels"] = data["class"].map({0: "Hate Speech", 1: "Offensive Language", 2: "No Hate and Offensive"})`

This line creates a new column in the "data" Data Frame named "labels", which maps the values in the "class" column (0, 1, or 2) to human-readable labels ("Hate Speech", "Offensive Language", or "No Hate and Offensive").

`data.head()`

This line prints the first five rows of the "data" Data Frame (now with the "labels" column added) to the console.

`data = data[["tweet", "labels"]]`

This line selects only the "tweet" and "labels" columns from the "data" Data Frame and assigns the resulting Data Frame to the "data" variable.

`data.head()`

This line prints the first five rows of the modified "data" Data Frame to the console.

`stopword=set(stopwords.words('english'))`

This line creates a set of English stopwords using the stopwords.words function from the nltk library.

`def clean(text):`

This line defines a function named "clean" that takes a string of text as input.

`text = str(text).lower()`

This line converts the input text to lowercase.

`text = re.sub('\[.*?\]', '', text)`

This line removes any text inside square brackets (e.g., "[...]", "[...]", etc.).

`text = re.sub('https?://\S+|www\.\S+', '', text)`:

This line is used to remove URLs or web links from the text.

text = re.sub('<.*?>+', '', text):

This operation removes any HTML tags present in the text. The regular expression '<.*?>+' matches any text enclosed within angle brackets, and the sub function replaces it with an empty string.

text = re.sub('[%s]' % re.escape(string.punctuation), '', text):

This operation removes all punctuation marks from the text. The regular expression [{}] matches any character present inside the square brackets, and the sub function replaces it with an empty string.

text = re.sub('\n', '', text):

This operation removes all newline characters from the text. The regular expression \n matches any newline character, and the sub function replaces it with an empty string.

text = re.sub('\w*\d\w*', '', text):

This operation removes any alphanumeric characters that contain digits from the text. This is useful for removing numbers and other numerical data that may not be relevant to natural language processing tasks.

text = [word for word in text.split(' ') if word not in stopword]:

This operation removes stopwords from the text, where stopwords is a list of common words like "the," "a," and "and" that are often removed from text data to reduce noise.

text=" ".join(text):

This line joins a list of words back into a single string, where each word is separated by a space. This is commonly used in text preprocessing to convert a list of processed words back into a string format that can be used as input to other functions or algorithms.

text = stemmer.stem(word) for word in text.split(' ']:

This operation performs stemming, which reduces words to their base form by removing suffixes and prefixes. This is useful for reducing the dimensionality of the text data and improving the accuracy of natural language processing algorithms.

data["tweet"].apply(clean):

This line applies the function clean to each element in the "tweet" column of the DataFrame data.

data.head():

This is a function that returns the first few rows of a pandas DataFrame. By default, it returns the first 5 rows, but you can specify a different number by passing it as an argument, like data.head(10) to return the first 10 rows.

x = np.array(data["tweet"]):

This line extracts the "tweet" column from the DataFrame data and converts it into a NumPy array.

y = np.array(data["labels"]):

This line extracts the "labels" column from the DataFrame data and converts it into a NumPy array.

Cv = CountVectorizer():

This is a class from scikit-learn's feature_extraction.text module that is used to convert a collection of text documents into a matrix of token counts.

X= cv.fit_transform(x):

This operation performs vectorization, which converts the text data into a matrix of numbers that can be used as input to machine learning algorithms. The CountVectorizer class from scikit-learn is used here to perform vectorization**.**

print(X.shape):

This line prints the shape of the matrix X that was created using the fit_transform() method of a CountVectorizer() object.

X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=0.33, random_state=42):

This operation splits the data into training and testing sets for model evaluation. The training set is used to train the model, while the testing set is used to evaluate its performance.

clf = DecisionTreeClassifier():

This line creates a decision tree classifier object from the sklearn.tree module.

clf.fit(X_train,y_train):

This line trains the decision tree classifier on the training data.

user = input("Enter a Text: ") :

This is a built-in Python function that allows you to prompt the user for input from the command line.

data = cv.transform([user]).toarray():

The pre-processed input text user is transformed into a matrix of token counts using the CountVectorizer() object cv. The transform() method of the cv object takes a list of strings as input and returns a matrix of token counts where each row represents a document and each column represents a token in the vocabulary.

Output = clf.predict(data):

This line uses the trained classifier to predict the label of a new text input.

print(output):

output is a variable that contains the predicted class for the input text. It was obtained by calling the predict() method on the trained DecisionTreeClassifier() model with the pre-processed input text matrix as input.

print(clf.score(X_train,y_train)):

These lines compute and prints` the accuracy of the classifier on the training and testing sets, respectively**.**

Print(clf.score(X_test,y_test)):

These lines compute and prints the accuracy of the classifier on the training and testing sets, respectively.

Results and Discussion

The decision tree classifier achieved an accuracy of [Insert Accuracy Here] on the test dataset. The results show that the proposed system is effective in detecting hate speech and offensive language in tweets.

Conclusion

The proposed system is an effective way to detect hate speech and offensive language in tweets. The decision tree classifier achieved high accuracy in classifying tweets into different categories. This system can be used to promote a safe and inclusive online environment.

Future Work

In future work, more advanced machine learning techniques can be explored to further improve the accuracy of hate speech detection. The system can also be extended to other social media platforms