A
Mini Project
On

TOXIC COMMENTS CLASSIFIER

(Submitted in partial fulfilment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

By

MOHD FEROZ	199P1A0597
ABDUL HYE	199P1A0596
MOHD JABER BIN TAHER	199P1A0581
K. NISHANTH	199P1A0590
B. VIJAYA KUMAR	199P1A05A5

Under the Guidance of

MR. RAMBABU

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SREE DATTHA GROUP OF INSTITUTIONS

(Affiliated to JNTUH, Approved by AICTE, New Delhi)

Sheriguda (V), Ibrahimpatnam (M), Ranga Reddy – 501510

2019-23

SREE DATTHA GROUP OF INSTITUTIONS

(Approved by AICTE and Affiliated to JNTUH) Sheriguda (V), Ibrahimpatnam (M), Ranga Reddy – 501510.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled “**TOXIC COMMENTS CLASSIFIER**” is being submitted by

Mohd Feroz	**(199P1A0597)**
Abdul Hye	**(199P1A0596)**
Mohd Jaber Bin Taher	**(199P1A0581)**
K. Nishanth	**(199P1A0590)**
B. Vijaya Kumar	**(199P1A05A5)**

In partial fulfilment of the requirements for the award of the degree of B. Tech in Computer Science and Engineering to the Sree Dattha Group of Institutions Hyderabad, is a record of bonafide work carried out by him/them under our guidance and supervision during the year 2022-23.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Mr. Rambabu
Internal Guide

Dr. M. Senthil Kumar
Principal

Dr. A Yashwanth Reddy
HOD

External Examiner

SREE DATTHA GROUP OF INSTITUTIONS

(Approved by AICTE and Affiliated to JNTUH) Sheriguda (V), Ibrahimpatnam (M), Ranga Reddy – 501510.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION BY THE CANDIDATE

We are hereby declaring that the project report titled "**TOXIC COMMENTS CLASSIFIER**" is submitted in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science & Engineering at Sree Dattha Group of Institutions, affiliated to Jawaharlal Nehru Technological University, Hyderabad is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Submitted by

Mohd Feroz	(199P1A0597)
Abdul Hye	(199P1A0596)
Mohd Jaber Bin Taher	(199P1A0581)
K. Nishanth	(199P1A0590)
B. Vijaya Kumar	(199P1A05A5)

ACKNOWLEDGEMENT

Apart from our efforts, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard for my guide **Mr. Rambabu,** Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We are also thankful to **Dr. A Yashwanth Reddy,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. M. Senthil Kumar,** Principal for being cooperative throughout this project. We would like to express our sincere gratitude to **Dr. GNV Vibhav Reddy**, Vice-Chairman and **Sri. G. Panduranga Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout this project.

The guidance and support were received from all the members of **Sree Dattha Group of Institutions** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

ABSTRACT

This project is titled as “TOXIC COMMENTS CLASSIFIER”. Platforms that aggregate user content are the foundation of knowledge sharing on the Internet. Blogs, forums, discussion boards, and, of course, Wikipedia. But the catch is that not all people on the Internet are interested in participating nicely, and some see it as an avenue to vent their rage, insecurity, and prejudices. Wikipedia runs on user generated content and is dependent on user discussion to curate and approve content. The problem with this is that people will frequently write things they shouldn’t, and to maintain a positive community this toxic content and the users posting it need to be removed quickly. But they don’t have the resources to hire full-time moderators to review every comment. This problem led the Conversation AI team, owned by Alphabet, to develop a large open dataset of labelled Wikipedia Talk Page comments, which will be the dataset used for the project. The dataset is available through Kaggle. We describe the concept of toxicity and characterize it in subcategories. The dataset has six labels that represent subcategories of toxicity. Further, we present various machine learning approaches. Then we are going to choose the best machine learning algorithm out of those, and we are going to use it in our flask app to determine the toxicity of the comment.

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 Project Scope

This project is titled as “Toxic Comment Classification”. First, we will explore the dataset to get a better picture of how the labels are distributed, how they correlate with each other, and what defines toxic or clean comments. Then, we are going to create a baseline score with a simple logistic regression classifier. After that, we will explore the effectiveness of multiple machine learning approaches and select the best for this problem. Tuning the parameters of the model to maximize performance is essential. Then, we build the final model with the best performing algorithm and parameters and test it on a holdout subset of the data. Finally, we develop our flask app with the model induced in it.

1.2 Project Purpose

The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments. The purpose of this project is to build a multi-headed model that’s capable of detecting different types of toxicity like threats, obscenity, insults, and identity hate. We will create a classifier model that can predict if input text is inappropriate (toxic) or not. We will be using a dataset of comments from Wikipedia’s talk page edits. This will hopefully help online discussion become more productive and respectful.

1.3 Project Features

The main feature of this project is that a comment is given to the best trained machine learning model. This model then predicts if the comment is toxic or not and it prints the probability of the comment falling under the six given categories.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analysed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once the analysis is completed the analyst has a firm understanding of what is to be done.

2.1 Problem Definition

A harsh, demeaning, or an unreasonable comment is referred to as toxic. Toxic comments can take various forms, including racism, discrimination, hate speech, harassment, profanity, and threats. Platforms that collect user material are the core of internet knowledge exchange. Blogs, forums, discussion boards, many social media apps such as instagram, facebook, youtube etc… and Wikipedia, of course. However, not everyone on the internet is interested in contributing politely and nicely, and others view it as a way to release their anger, insecurities, and prejudices. These blogs and discussion platforms are dependent on user discussion to curate and approve content. The issue with this is that people frequently post inappropriate things and feel free to insult others. This results in toxicity. Thus, it has to be monitored and censored on leading social networking sites. So, we build our own simple Toxic Comment Classifier using Natural Language Processing (NLP). The goal is to create a classifier model that can predict if input text is inappropriate (toxic) or not. A comment from social media platforms, our task is to identify if it has any toxic text content and classify it into given categories.

2.2 Proposed System

This section focuses on different algorithms and the various stages that are involved for the proposed Toxic comment classification system such as ‘logistic regression’, ‘BR Method with Multinomial Naive Bayes classifiers’ and ‘BR Method with SVM classifier’ which helps us in classifying the comments and results in a decisive outcome.

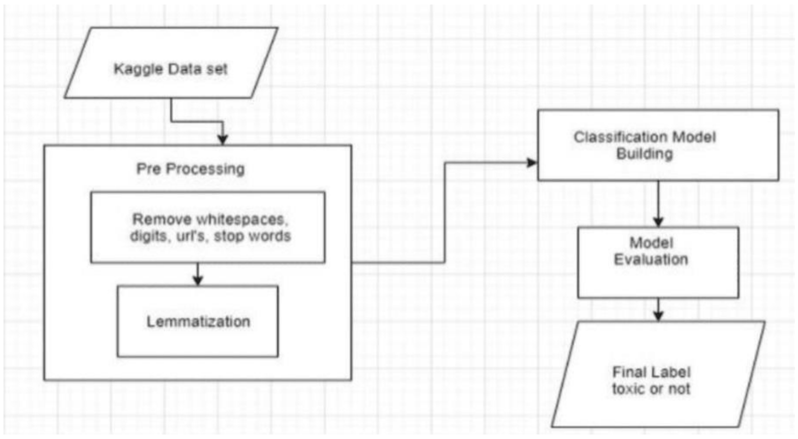


Figure 1: Proposed System

As the aim was to find out whether the data belongs to either zero, one or more than one from the six in the list, the initial agenda before working on the problem was to audibly differentiate between multi-label and multi-class classification. In multi-class classification, we undertake one basic assumption that our data can belong to only one label out of all that are available to us. Let us say, for a given picture of a vegetable may be a potato, cabbage, or onion only and not a combination of the above. Whereas, in multi-label classification, data can belong concurrently to more than one label. For example, in this project a comment may be belonging to more than one classification concurrently, like it may be toxic, hateful, obscene, and abusive at the same time it might concomitantly belong to non-toxic category and thus does not have an affinity to any of the six labels which are used for classification.

2.3 Advantages of The Proposed System

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features

- Ensure data accuracy.
- Minimum time needed for the various processing.
- Greater efficiency.
- Better service.
- User friendliness and interactive.
- Minimum time required.

2.4 Feasibility Study

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 Economic Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

2.4.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 Behavioral Feasibility

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioural aspects are considered carefully and conclude that the project is behaviourally feasible.

3. HARDWARE AND SOFTWARE REQUIREMENTS

3. HARDWARE AND SOFTWARE REQUIREMENTS

3.1 Hardware Requirements

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

• Processor	:	Intel Dual Core @ CPU 2.90GHz.
• Hard disk	:	16GB and above.
• RAM	:	4GB and above.
• Monitor	:	15 inches or above.

3.2 Software Requirements:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

• Operating system	:	Windows 8 above
• Languages	:	Python, HTML, CSS
• Backend	:	Python
• IDE	:	Jupyter Notebook, VS Code

4. PROPOSED METHODOLOGY

4. PROPOSED METHODOLOGY

4.1 Types Of Classifications

In this report, we will classify the given dataset (comments written by a user in an online forum) provided by Kaggle in six labels, i.e., toxic, obscene, identity hate, severe toxic, threat, or insult. The next step is to determine if the given data (comment) belongs to one or more than one or none of the mentioned six labels. For example, the given comment can be toxic and insulting, hence falling into more than one label, but the comment can also be non-toxic and not fall into any of the six labels. Before we begin, we should first understand the difference between multi-class and multi-label classification. The classes in multi-class problems are mutually exclusive, meaning that each input is assigned to only one label. So, for example, your phone operating system could be android or iOS, but it cannot run both simultaneously. However, Multi-label classification assigns each input a set of target labels, i.e., the input may be assigned to several labels at the same time. So, we may conclude that toxic comment classification is a multi-label classification problem.

4.2 Data Exploration

This dataset contains 1,59,571 comments from Wikipedia. The data consists of one input feature, the string data for the comments, and six labels for different categories of toxic comments: toxic, severe_toxic, obscene, threat, insult, and identity_hate. The figure on the following page contains a breakdown of how the labels are distributed throughout the dataset, including overlapping data. As you can see in the breakdown, while most comments with other labels are also toxic, not all of them are. Only “severe_toxic” is clearly a subcategory of “toxic.” And it’s not close enough to be a labelling error. This suggests that “toxic” is not a catch-all label, but rather a subcategory in itself with a large amount of overlap.

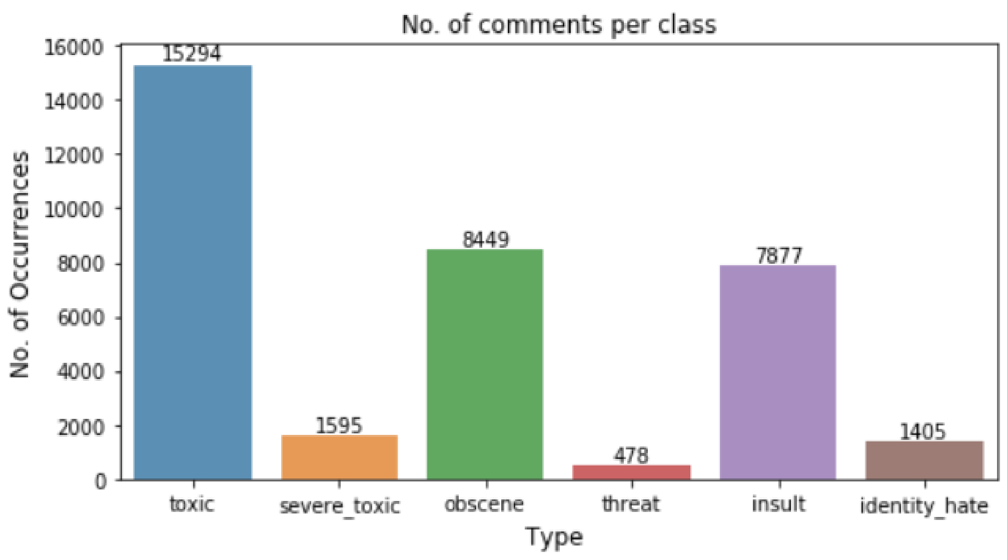


Figure 2: Number of Toxic Comments

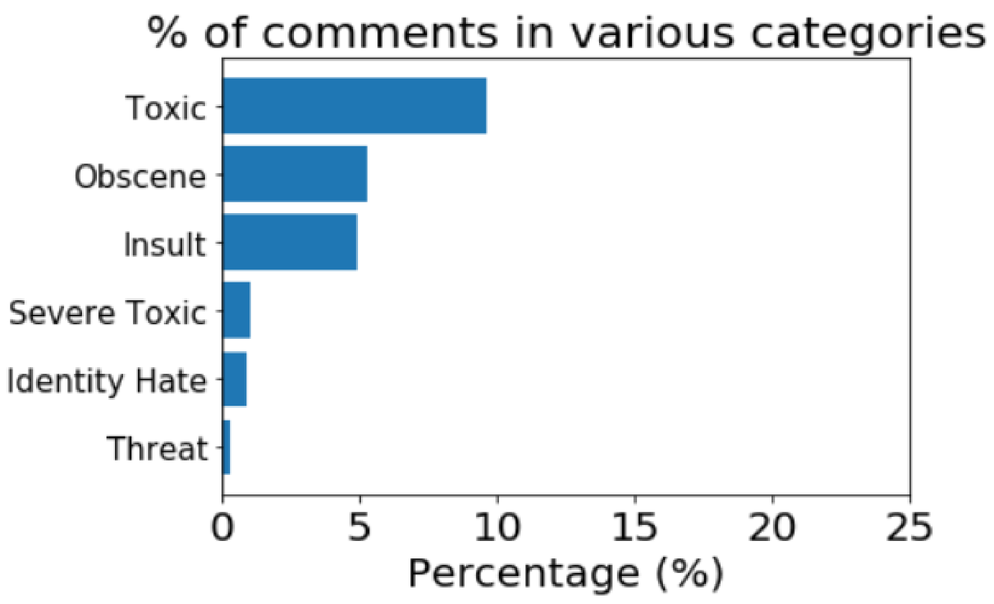


Figure3: Percentage of Toxic Comments

Only 39% of the toxic comments have only one label, and the majority have some sort of overlap. I believe that because of this, it will be much more difficult to train a classifier on specific labels than whether or not they are toxic.16,225 out of 1,59,571 comments, or 10.17%, are classified as some category of toxic.

i. 1595 severe_toxic comments (1.00% of all data).
   - 1595 or 100.00% were also toxic.
   - 1517 or 95.11% were also obscene.
   - 112 or 7.02% were also threat.
   - 1371 or 85.96% were also insult.
   - 313 or 19.62% were also identity_hate.

ii. 1405 identity_hate comments (0.88% of all data).
- 1302 or 92.67% were also toxic.
- 313 or 22.28% were also severe_toxic.
- 1032 or 73.45% were also obscene.
- 98 or 6.98% were also threat.
- 1160 or 82.56% were also insult.

iii. 15294 toxic comments (9.58% of all data).
- 1595 or 10.43% were also severe_toxic.
- 7926 or 51.82% were also obscene.
- 449 or 2.94% were also threat.
- 7344 or 48.02% were also insult.
- 1302 or 8.51% were also identity_hate.

iv. 7877 insult comments (4.94% of all data).
- 7344 or 93.23% were also toxic.
- 1371 or 17.41% were also severe_toxic.
- 6155 or 78.14% were also obscene.
- 307 or 3.90% were also threat.
- 1160 or 14.73% were also identity_hate.

v. 478 threat comments (0.30% of all data).
- 449 or 93.93% were also toxic.
- 112 or 23.43% were also severe_toxic.
- 301 or 62.97% were also obscene.
- 307 or 64.23% were also insult.
- 98 or 20.50% were also identity_hate.

vi. 8449 obscene comments (5.29% of all data).
- 7926 or 93.81% were also toxic.
- 1517 or 17.95% were also severe_toxic.
- 301 or 3.56% were also threat.
- 6155 or 72.85% were also insult.
- 1032 or 12.21% were also identity_hate.

4.3 Data Pre-Processing

Data pre-processing is a technique used to transform the raw data into an understandable and readable format to make it suitable for building and training Machine Learning models. For our dataset, this can be achieved in 2 stages: (1) Data Cleaning (Removal of unnecessary elements from our text); (2) Feature Engineering (extracting features from data and transforming them into formats that are suitable for Machine Learning algorithms)

Steps for Data (Text) Cleaning:

- Removing Punctuations and other special/ non-ASCII characters.
- Splitting the comments into individual words.
- Removing Stop Words.
- Stemming and Lemmatising.

Hence, we get the comments as lists of clean tokens, and now we need to convert each of those comments into a vector through feature engineering to make them suitable for the SciKit Learn's algorithms.

The next step is to extract features using two techniques:

- Count Vectorization
- TF-IDF Transformation.

Now our dataset is ready for train-test split and we can run it in any suitable Machine Learning model.

4.4 Creating Subset of Data For Classifications

The majority of conventional machine learning algorithms are designed for classification problems with single-label. Hence, we'll use create subsets of data to break the multi-label problem into several single-label problems, allowing us to use the existing conventional machine learning algorithms.

With each of these methods, we will use five machine algorithms to get optimal results.

- Logistic Regression
- Multinomial Naïve Bayes
- Random Forest Classifier
- Bernoulli Naïve Bayes
- K- nearest neighbour
- SVM

4.5 Result And Analysis

We used almost all machine learning algorithm, i.e., Logistic Regression, Multinomial Naïve Bayes, Random Forest Classifier, Bernoulli Naïve Bayes, SVM, k-nearest neighbour, etc for each our project

5. SOFTWARE TOOLS AND DESCRIPTION

5. SOFTWARE TOOLS AND DESCRIPTION

5.1 Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, including structured, object-oriented, and functional programming. Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

5.2 Anaconda

Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free. Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open source package as it ended up being useful on its own and for things other than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

5.3 Jupyter Notebook

Jupyter is a project with goals to develop open-source software, open standards, and services for interactive computing across multiple programming languages. It was spun off from IPython in 2014 by Fernando Pérez and Brian Granger. The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. Also, it is a web-based interactive computational environment. The Jupyter notebook can support various languages that are popular in data science such as Python, Julia, Scala, R, etc…

5.4 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Flask is used for developing web applications using python, implemented on Werkzeug and Jinja2. Advantages of using Flask framework are: There is a built-in development server and a fast debugger provided. Since it is a micro-framework, it is very easy to use and lacks most of the advanced functionality which is found in a full-fledged framework. Therefore, building a REST API in Flask is very simple.

5.5 Visual Studio Code

Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences. The source code is free and open source and released under the permissive MIT License. The compiled binaries are freeware and free for private or commercial use. 6.7 Node JS: Visual Studio Code is based on Electron, a framework which is used to deploy Node.js applications for the desktop running on the Blink layout engine. Although it uses the Electron framework, the software does not use Atom and instead employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services). Visual Studio Code is a source code editor that can be used with a variety of programming languages including solidity. In the Stack Overflow 2019 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 50.7% of 87,317 respondents claiming to use it.

6. SYSTEM DESIGN

6. SYSTEM DESIGN

6.1 Project Architecture

This project architecture shows the procedure followed for Toxic comment classification using machine learning, starting from input to final prediction.

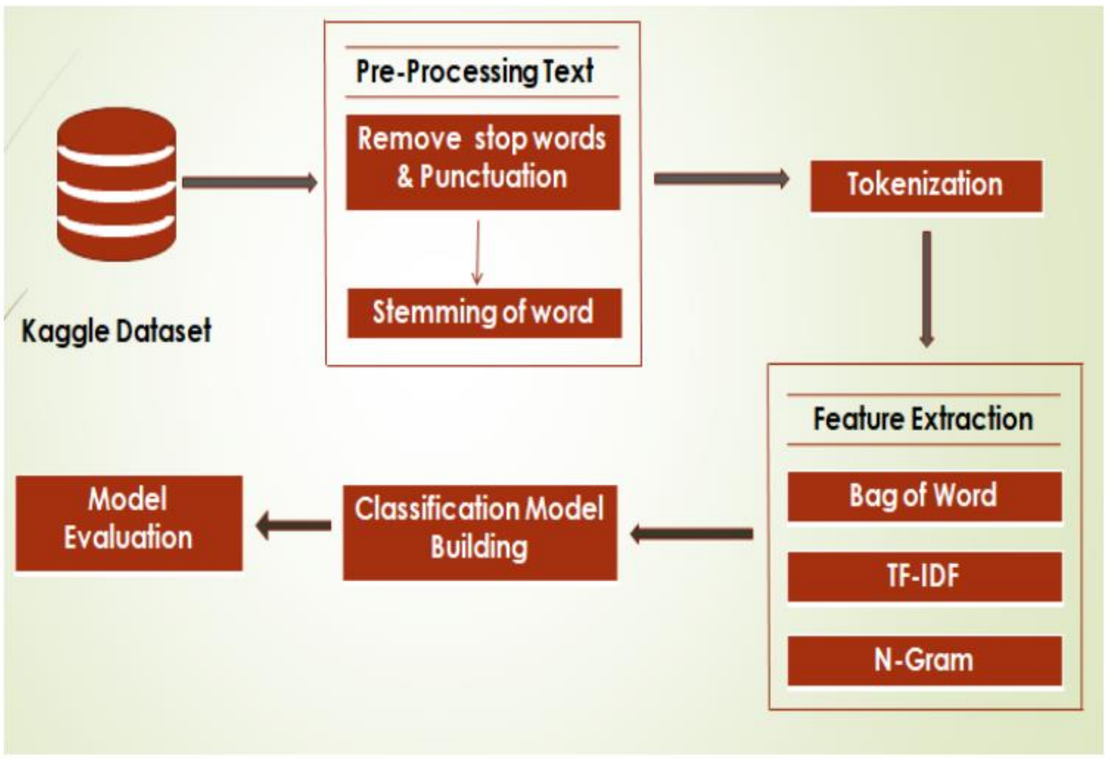


Figure 5: Project Architecture

6.2 Model Training

This diagram shows that how train to the model.

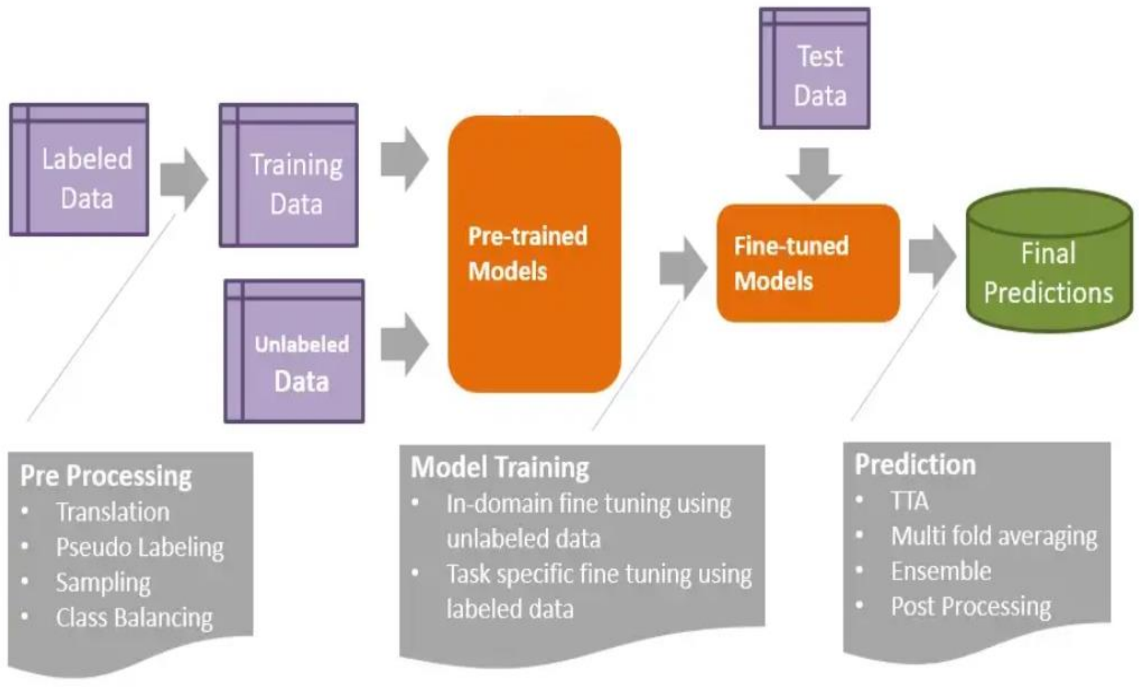


Figure 6: Training Models

6.3 Use Case Diagram

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The below diagram consists of user as actor. Each will play a certain role to achieve the concept.

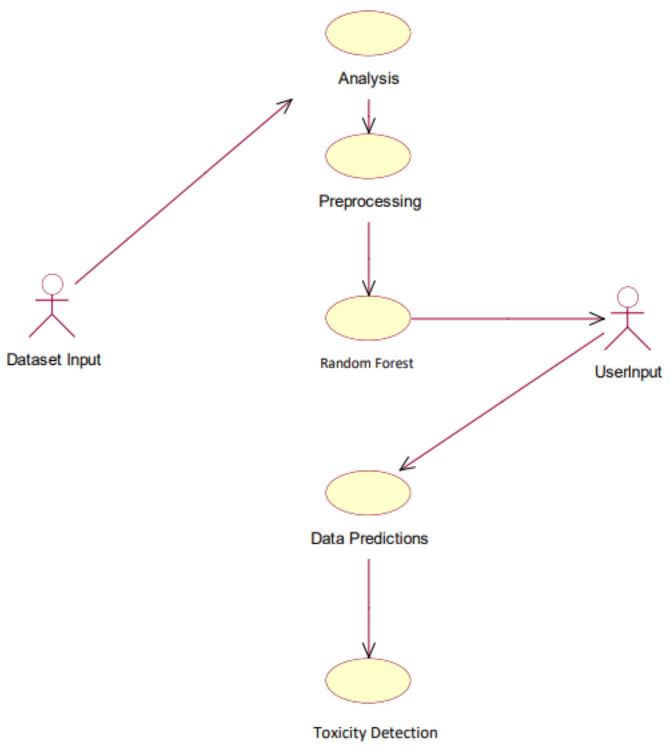


Figure 7: Use Case Diagram

6.4 Class Diagram

In this class diagram represents how the classes with attributes and methods are linked together from the below diagram shown the various classes involved in our project.

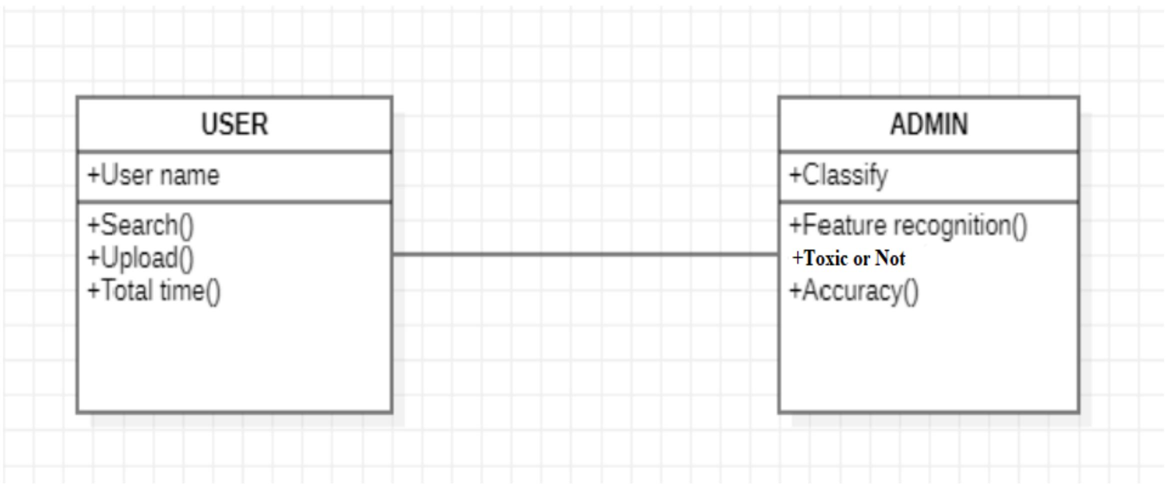


Figure 8: Class Diagram

6.5 Object Diagram

In the below digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods.

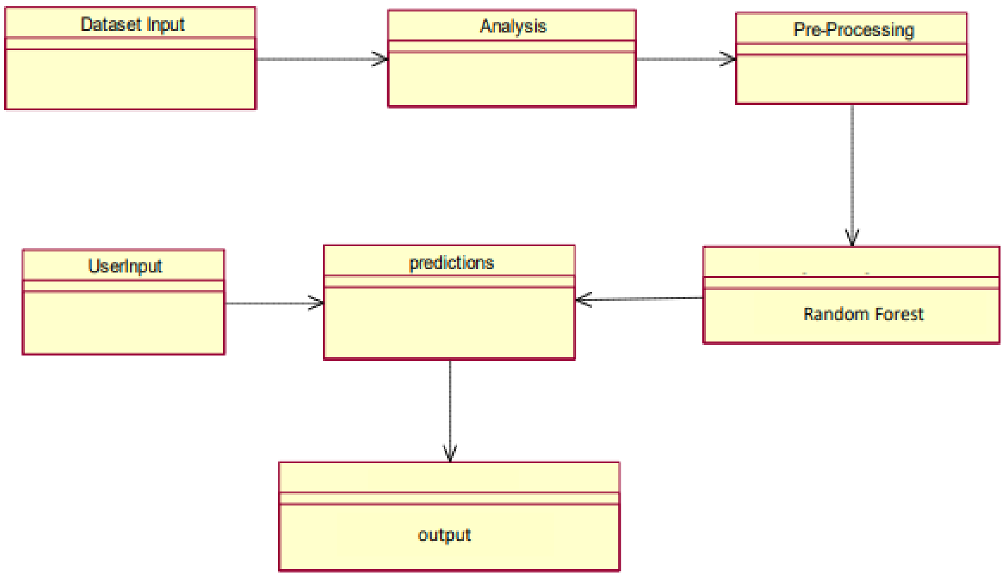


Figure 9: Object Diagram

6.6 State Diagram

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

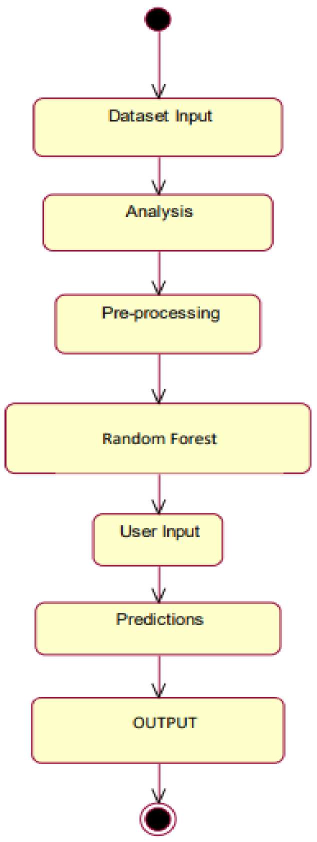


Figure 10: State Diagram

6.7 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

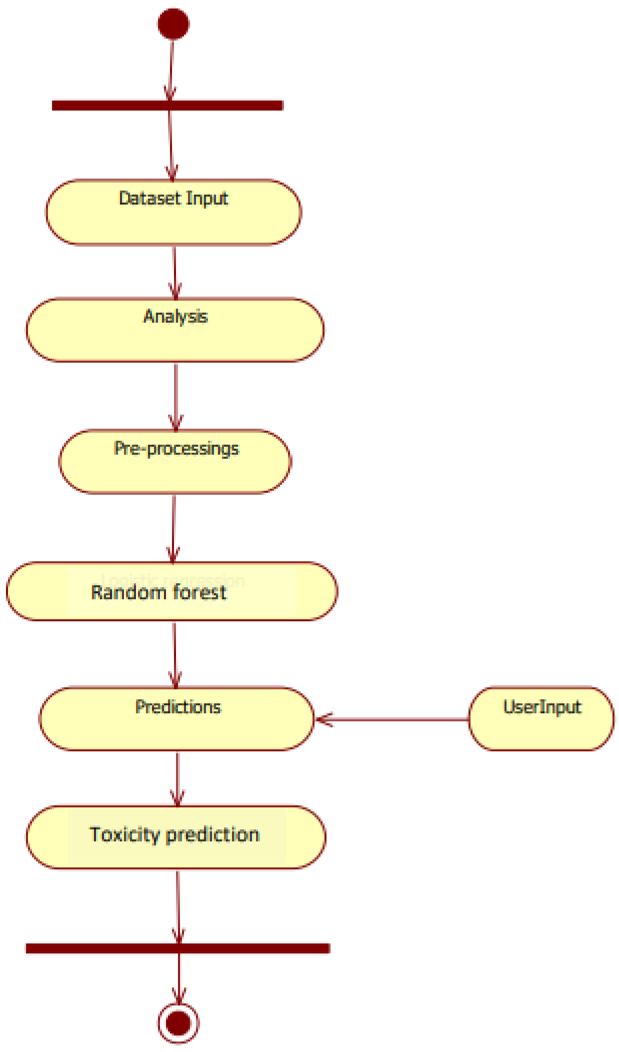


Figure 11: Activity Diagram

7. IMPLEMENTATION

7. IMPLEMENTATION

7.1 Jupyter Notebook Code

Import relevant libraries and load the data.

```
# Import relevant packages

import matplotlib.pyplot as plt

import nltk

import numpy as np

import re

import pandas as pd

import string

import seaborn as sns

from nltk.corpus import stopwords  # Remove useless words

# Import packages that help us to create document-term matrix

from sklearn.feature_extraction.text import CountVectorizer,  TfidfVectorizer
```

Load the CSV and take a peek at 1st 5 rows.

```
data = pd.read_csv('train.csv')

data.head()

data.info()

data['comment_text'][0]

data['comment_text'][1]

data['comment_text'][3]
```

Exploratory Data Analysis

```
# Check percentage of comments that are toxic compared to normal comments

data.toxic.value_counts(normalize=True)

#Create a new subset of the data by only taking the 2nd column onwards (comments and categories)
data_count=data.iloc[:,2:].sum()
# Plot a chart with the following size
```

```
plt.figure(figsize=(8,4))

# Plot a bar chart using the index (category values) and the count of each category. alpha = 0.8 to make the bars more translucent

ax = sns.barplot(data_count.index, data_count.values, alpha=0.8)

plt.title("No. of comments per class")

plt.ylabel('No. of Occurrences', fontsize=12)

plt.xlabel('Type ', fontsize=12)

#adding the text labels for each bar

rects = ax.patches

labels = data_count.values

for rect, label in zip(rects, labels):

height = rect.get_height()

ax.text(rect.get_x() + rect.get_width()/2, height + 5, label, ha='center', va='bottom')

plt.show()

num_rows = len(data)

num_rows

# Creating another bar graph

sum_tox = data['toxic'].sum() / num_rows * 100

sum_sev = data['severe_toxic'].sum() / num_rows * 100

sum_obs = data['obscene'].sum() / num_rows * 100

sum_thr = data['threat'].sum() / num_rows * 100

sum_ins = data['insult'].sum() / num_rows * 100

sum_ide = data['identity_hate'].sum() / num_rows * 100

# Initiate a list of 6 values that represent the 6 x-axis values for the categories

ind = np.arange(6)


# Let the ind variable be the x-axis, whereas the % of toxicity for each category be the y-axis.
# Sequence of % have been sorted manually. This method cannot be done if there are large numbers of categories.

ax = plt.barh(ind, [sum_tox, sum_obs, sum_ins, sum_sev, sum_ide, sum_thr])

plt.xlabel('Percentage (%)', size=20)

plt.xticks(np.arange(0, 30, 5), size=20)

plt.title('% of comments in various categories', size=22)

plt.yticks(ind, ('Toxic', 'Obscene', 'Insult', 'Severe Toxic', 'Identity Hate', 'Threat', ), size=15)

# Invert the graph so that it is in descending order.

plt.gca().invert_yaxis()

plt.show()
```

Pre-process the text

```
# Text preprocessing steps - remove numbers, capital letters, punctuation, '\n'

import re

import string

# remove all numbers with letters attached to them

alphanumeric = lambda x: re.sub('\w*\d\w*', ' ', x)

# '[%s]' % re.escape(string.punctuation),' ' - replace punctuation with white space

# .lower() - convert all strings to lowercase

punc_lower = lambda x: re.sub('[%s]' % re.escape(string.punctuation), ' ', x.lower())

# Remove all '\n' in the string and replace it with a space

remove_n = lambda x: re.sub("\n", " ", x)

# Remove all non-ascii characters

remove_non_ascii = lambda x: re.sub(r'[^\x00-\x7f]',r' ', x)

# Apply all the lambda functions wrote previously through .map on the comments column

data['comment_text'] = data['comment_text'].map(alphanumeric).map(punc_lower).map(remove_n).map(remove_non_ascii)

data['comment_text'][0]
```

Separate our dataset into 6 sections. Each section is comment + 1 category.

```
data_tox = data.loc[:,['id','comment_text','toxic']]

data_sev = data.loc[:,['id','comment_text','severe_toxic']]

data_obs = data.loc[:,['id','comment_text','obscene']]

data_thr = data.loc[:,['id','comment_text','threat']]

data_ins = data.loc[:,['id','comment_text','insult']]

data_ide = data.loc[:,['id','comment_text','identity_hate']]

data_tox.head()

data_sev.head()

data_obs.head()

data_thr.head()

data_ins.head()

data_ide.head()
```

For a start, we can take 5000 rows of comments that are toxic and concatenate them row-wise with those that are not toxic so that we have a balanced dataset.

```
data_tox_1 = data_tox[data_tox['toxic'] == 1].iloc[0:5000,:]

data_tox_1.shape

data_tox_0 = data_tox[data_tox['toxic'] == 0].iloc[0:5000,:]

data_tox_0.shape

data_tox_done = pd.concat([data_tox_1, data_tox_0], axis=0)

data_tox_done.shape

data_sev[data_sev['severe_toxic'] == 1].count()

data_ins_1 = data_ins[data_ins['insult'] == 1].iloc[0:5000,:]

data_ins_0 = data_ins[data_ins['insult'] == 0].iloc[0:5000,:]

data_ins_done = pd.concat([data_ins_1, data_ins_0], axis=0)

data_ins_done.shape

data_ide[data_ide['identity_hate'] == 1].count()
```

```
data_ide_1 = data_ide[data_ide['identity_hate'] == 1].iloc[0:1405,:] # 20%

data_ide_0 = data_ide[data_ide['identity_hate'] == 0].iloc[0:5620,:] # 80%

data_ide_done = pd.concat([data_ide_1, data_ide_0], axis=0)

data_ide_done.shape
```

Import relevant packages for modelling

```
# Import packages for pre-processing

from sklearn import preprocessing

from sklearn.feature_selection import SelectFromModel

# Import tools to split data and evaluate model performance

from sklearn.model_selection import train_test_split, KFold, cross_val_score

from sklearn.metrics import f1_score, precision_score, recall_score,
precision_recall_curve, fbeta_score, confusion_matrix

from sklearn.metrics import roc_auc_score, roc_curve

# Import ML algos

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import MultinomialNB, BernoulliNB

from sklearn.svm import LinearSVC

from sklearn.ensemble import RandomForestClassifier
```

Create simple function that takes in a dataset and allows user to choose dataset, toxicity label, vectorizer and number of ngrams

```
def cv_tf_train_test(df_done,label,vectorizer,ngram):

   ''' Train/Test split'''

   # Split the data into X and y data sets

   X = df_done.comment_text

   y = df_done[label]

# Split our data into training and test data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
''' Count Vectorizer/TF-IDF '''

# Create a Vectorizer object and remove stopwords from the table

cv1 = vectorizer(ngram_range=(ngram), stop_words='english')

X_train_cv1 = cv1.fit_transform(X_train) # Learn the vocabulary dictionary and return
term-document matrix

test_cv1  = cv1.transform(X_test)      # Learn a vocabulary dictionary of all tokens in the
raw documents.

# The fit method is calculating the mean and variance of each of the features present in our
data.

# The transform method is transforming all the features using the respective mean and
variance.

# Output a Dataframe of the CountVectorizer with unique words as the labels

# test = pd.DataFrame(X_train_cv1.toarray(), columns=cv1.get_feature_names())

''' Initialize all model objects and fit the models on the training data '''

lr = LogisticRegression()

lr.fit(X_train_cv1, y_train)

print('lr done')

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train_cv1, y_train)

print('knn done')

mnb = MultinomialNB()

mnb.fit(X_train_cv1, y_train)

print('mnb done')

svm_model = LinearSVC()

svm_model.fit(X_train_cv1, y_train)

print('svm done')

randomforest = RandomForestClassifier(n_estimators=100, random_state=42)

randomforest.fit(X_train_cv1, y_train)

print('rdf done')

# Create a list of F1 score of all models
```

```
f1_score_data = {'F1 Score':[f1_score(lr.predict(X_test_cv1), y_test),
f1_score(knn.predict(X_test_cv1), y_test),

f1_score(mnb.predict(X_test_cv1), y_test),

f1_score(svm_model.predict(X_test_cv1), y_test),
f1_score(randomforest.predict(X_test_cv1), y_test)]}

# Create DataFrame with the model names as column labels

df_f1 = pd.DataFrame(f1_score_data, index=['Log Regression','KNN', 'MultinomialNB',
'SVM', 'Random Forest'])

return df_f1
```

Let's create a TF-IDF vectorizer object for each category and calculate the F1 scores across all models

```
df_tox_cv = cv_tf_train_test(data_tox_done, 'toxic', TfidfVectorizer, (1,1))

df_tox_cv.rename(columns={'F1 Score': 'F1 Score(toxic)'}, inplace=True)

df_tox_cv

df_sev_cv = cv_tf_train_test(data_sev_done, 'severe_toxic', TfidfVectorizer, (1,1))

df_sev_cv.rename(columns={'F1 Score': 'F1 Score(severe_toxic)'}, inplace=True)

df_sev_cv

df_obs_cv = cv_tf_train_test(data_obs_done, 'obscene', TfidfVectorizer, (1,1))

df_obs_cv.rename(columns={'F1 Score': 'F1 Score(obscene)'}, inplace=True)

df_obs_cv

df_thr_cv = cv_tf_train_test(data_thr_done, 'threat', TfidfVectorizer, (1,1))

df_thr_cv.rename(columns={'F1 Score': 'F1 Score(threat)'}, inplace=True)

df_thr_cv

df_ins_cv = cv_tf_train_test(data_ins_done, 'insult', TfidfVectorizer, (1,1))

df_ins_cv.rename(columns={'F1 Score': 'F1 Score(insult)'}, inplace=True)

df_ins_cv

df_ide_cv = cv_tf_train_test(data_ide_done, 'identity_hate', TfidfVectorizer, (1,1))

df_ide_cv.rename(columns={'F1 Score': 'F1 Score(identity_hate)'}, inplace=True)

df_ide_cv
```

```
# We are combining the dataframes into a master dataframe to compare F1 scores across all categories.
f1_all = pd.concat([df_tox_cv, df_sev_cv, df_obs_cv, df_ins_cv, df_thr_cv, df_ide_cv], axis=1)
f1_scr = f1_all.transpose()
f1_scr
sns.lineplot(data=f1_scr, markers=True)
plt.xticks(rotation='90', fontsize=14)
plt.yticks(fontsize=14)
plt.legend()
plt.title('F1 Score of ML models (TF-IDF)', fontsize=20)
```

Testing if our code actually works. Probability of the comment falling in various categories should be output.

```
X = data_tox_done.comment_text
y = data_tox_done['toxic']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Initiate a Tfidf vectorizer
tfv = TfidfVectorizer(ngram_range=(1,1), stop_words='english')
X_train_fit = tfv.fit_transform(X_train)  # Convert the X data into a document term matrix dataframe
X_test_fit = tfv.transform(X_test)  # Converts the X_test comments into Vectorized format
randomforest = RandomForestClassifier(n_estimators=100, random_state=42)
randomforest.fit(X_train_fit, y_train)
randomforest.predict(X_test_fit)
```

Sample Prediction

```
tox_com = ['You piece of shit']
non_tox_com = ['What is up garden apple doing']
tox_com_vect = tfv.transform(tox_com)
randomforest.predict_proba(tox_com_vect)[:,1]
```

```
non_tox_com_vect = tfv.transform(non_tox_com)

randomforest.predict_proba(non_tox_com_vect)[:,1]
```

Pickling trained RandomForest models for all categories.

```
import pickle

#We have to pickle not only the TF-IDF vectorizer object, but also the RDF model trained on the related vectorizer.

def pickle_model(df, label):

X = df.comment_text

y = df[label]

# Initiate a Tfidf vectorizer

tfv = TfidfVectorizer(ngram_range=(1,1), stop_words='english')

# Convert the X data into a document term matrix dataframe

X_vect = tfv.fit_transform(X)

# saves the column labels (ie. the vocabulary)

# wb means Writing to the file in Binary mode, written in byte objects

with open(r"{}.pkl".format(label + '_vect'), "wb") as f:

pickle.dump(tfv, f)

randomforest = RandomForestClassifier(n_estimators=100, random_state=42)

randomforest.fit(X_vect, y)

# Create a new pickle file based on random forest

with open(r"{}.pkl".format(label + '_model'), "wb") as f:

pickle.dump(randomforest, f)
```

Create a loop to create pickle files all at one shot

```
datalist = [data_tox_done, data_sev_done, data_obs_done, data_ins_done, data_thr_done, data_ide_done]

label = ['toxic', 'severe_toxic', 'obscene', 'insult', 'threat', 'identity_hate']

for i,j in zip(datalist,label):

pickle_model(i, j)
```

7.2 Flask App Code

```
from flask import Flask, render_template, url_for, request, jsonify
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import pickle
import numpy as np

app = Flask(__name__)

# Load the TF-IDF vocabulary specific to the category
with open(r"toxic_vect.pkl", "rb") as f:
    tox = pickle.load(f)

with open(r"severe_toxic_vect.pkl", "rb") as f:
    sev = pickle.load(f)

with open(r"obscene_vect.pkl", "rb") as f:
    obs = pickle.load(f)

with open(r"insult_vect.pkl", "rb") as f:
    ins = pickle.load(f)

with open(r"threat_vect.pkl", "rb") as f:
    thr = pickle.load(f)

with open(r"identity_hate_vect.pkl", "rb") as f:
    ide = pickle.load(f)

# Load the pickled RDF models
with open(r"toxic_model.pkl", "rb") as f:
    tox_model = pickle.load(f)

with open(r"severe_toxic_model.pkl", "rb") as f:
    sev_model = pickle.load(f)

with open(r"obscene_model.pkl", "rb") as f:
    obs_model  = pickle.load(f)

with open(r"insult_model.pkl", "rb") as f:
    ins_model  = pickle.load(f)

with open(r"threat_model.pkl", "rb") as f:
    thr_model  = pickle.load(f)
```

```
with open(r"identity_hate_model.pkl", "rb") as f:
    ide_model  = pickle.load(f)

# Render the HTML file for the home page
@app.route("/")
def home():
    return render_template('index_toxic.html')

@app.route("/predict", methods=['POST'])
def predict():

    # Take a string input from user
    user_input = request.form['text']
    data = [user_input]

    vect = tox.transform(data)
    pred_tox = tox_model.predict_proba(vect)[:,1]

    vect = sev.transform(data)
    pred_sev = sev_model.predict_proba(vect)[:,1]

    vect = obs.transform(data)
    pred_obs = obs_model.predict_proba(vect)[:,1]

    vect = thr.transform(data)
    pred_thr = thr_model.predict_proba(vect)[:,1]

    vect = ins.transform(data)
    pred_ins = ins_model.predict_proba(vect)[:,1]

    vect = ide.transform(data)
    pred_ide = ide_model.predict_proba(vect)[:,1]


    out_tox = round(pred_tox[0], 2)
    out_sev = round(pred_sev[0], 2)
    out_obs = round(pred_obs[0], 2)
    out_ins = round(pred_ins[0], 2)
    out_thr = round(pred_thr[0], 2)
    out_ide = round(pred_ide[0], 2)

 print(out_tox)

return render_template ('index_toxic.html',
pred_tox = 'Prob (Toxic): {}'.format(out_tox),
pred_sev = 'Prob (Severe Toxic): {}'.format(out_sev),
pred_obs = 'Prob (Obscene): {}'.format(out_obs),
pred_ins = 'Prob (Insult): {}'.format(out_ins),
pred_thr = 'Prob (Threat): {}'.format(out_thr),
pred_ide = 'Prob (Identity Hate):{}'.format(out_ide))

# Server reloads itself if code changes so no need to keep restarting:
app.run(debug=True)
```

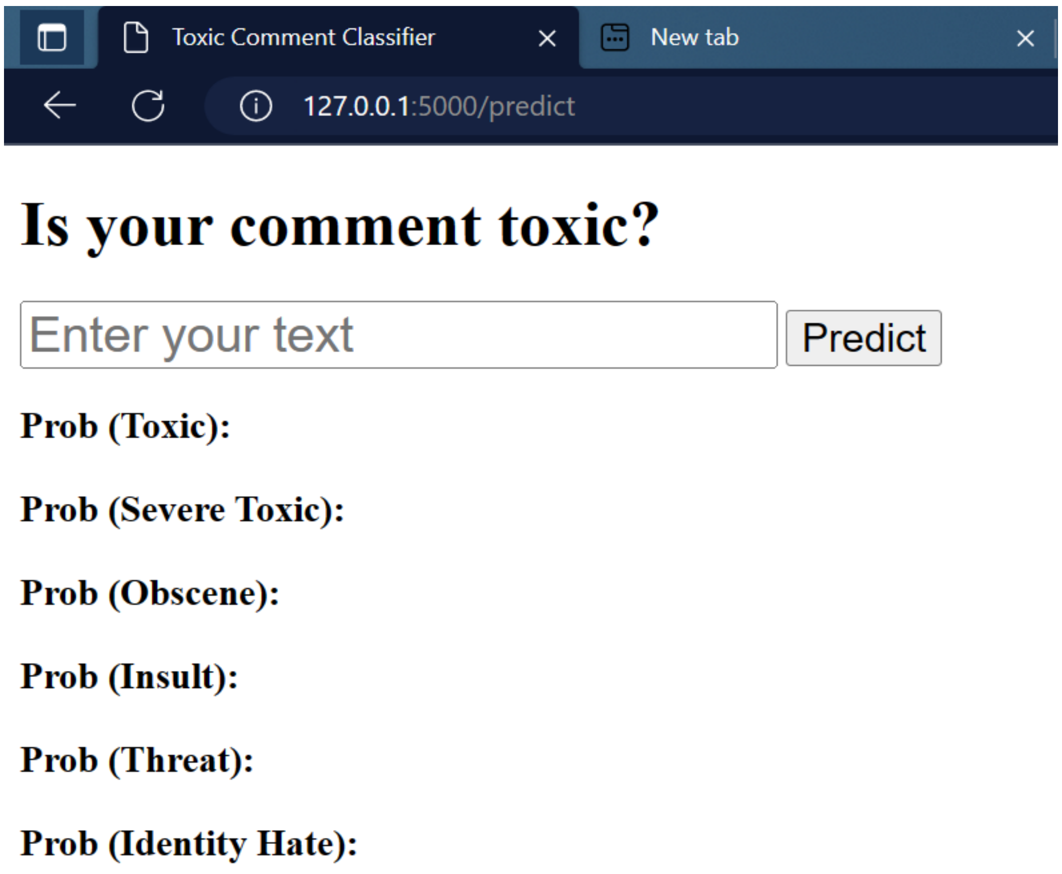
8. SCREENSHOTS

8. SCREENSHOTS

8.1 Introduction

The UI is designed using HTML. Following section contains the screenshots of the UI.

8.2 Screenshots



Is your comment toxic?

Prob (Toxic):

Prob (Severe Toxic):

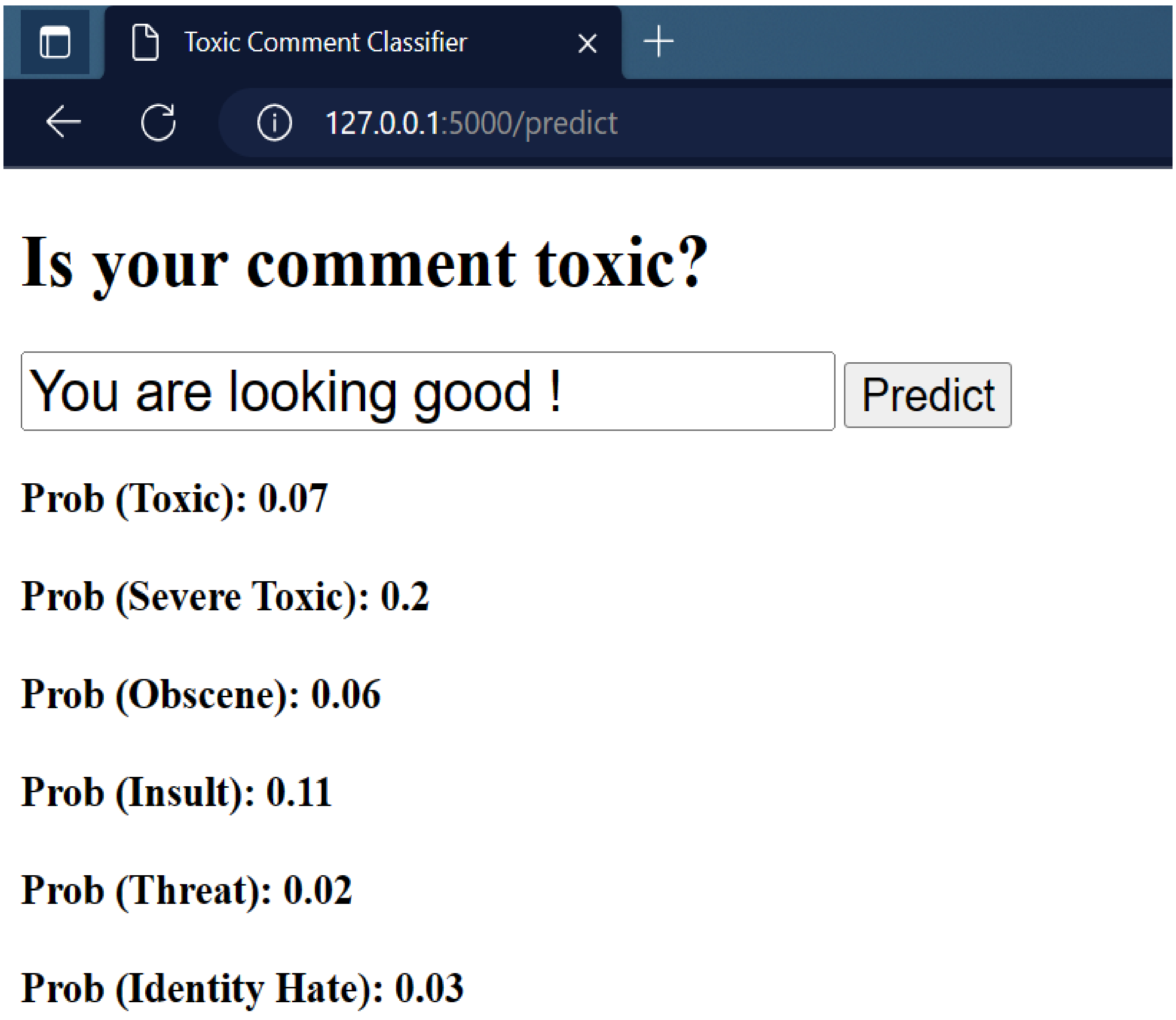
Prob (Obscene):

Prob (Insult):

Prob (Threat):

Prob (Identity Hate):

Screenshot 1: Graphical User Interface



Is your comment toxic?

You are looking good !

Prob (Toxic): 0.07

Prob (Severe Toxic): 0.2

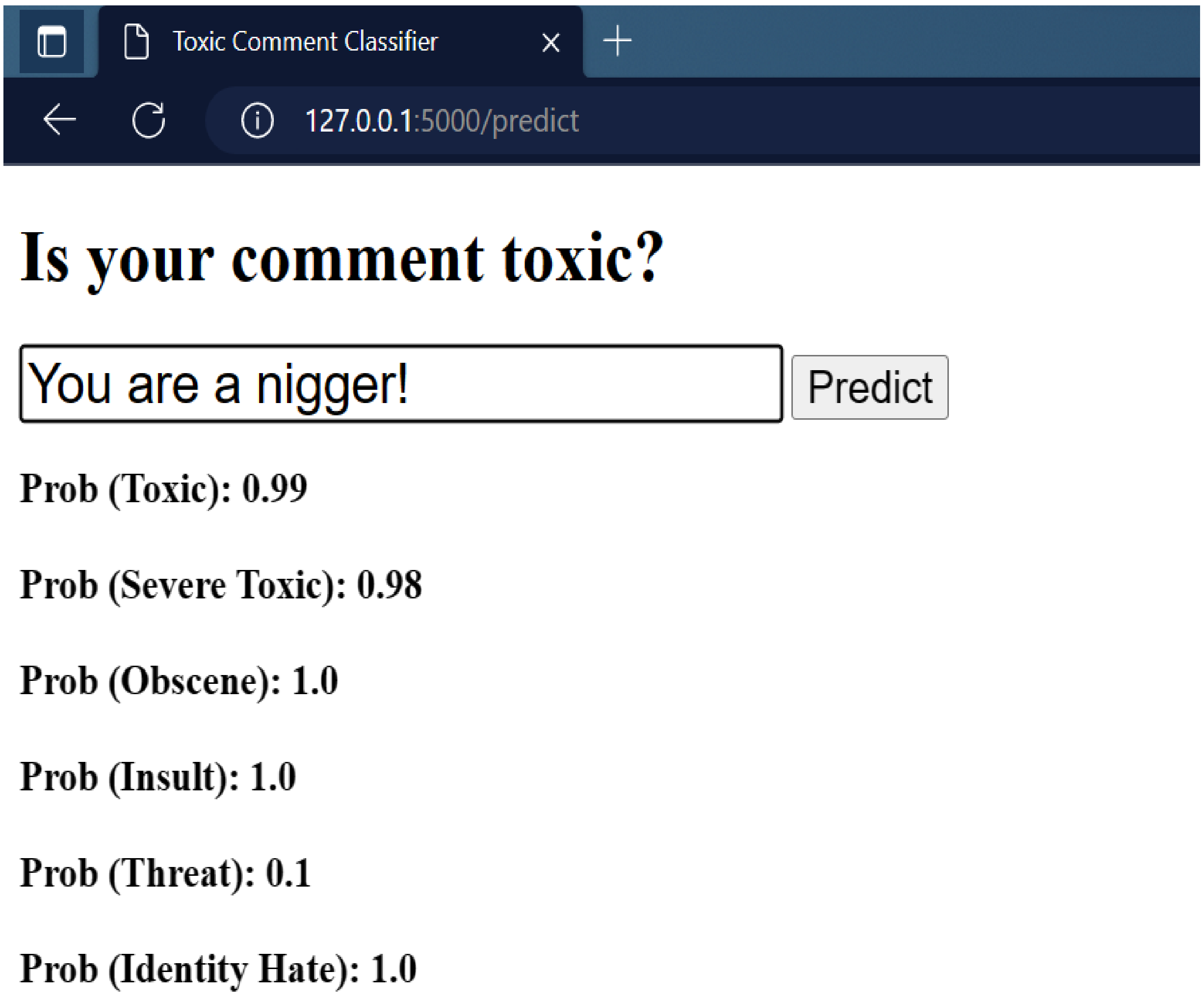
Prob (Obscene): 0.06

Prob (Insult): 0.11

Prob (Threat): 0.02

Prob (Identity Hate): 0.03

Screenshot 2: Result of non-toxic comment



Is your comment toxic?

You are a nigger!

Prob (Toxic): 0.99

Prob (Severe Toxic): 0.98

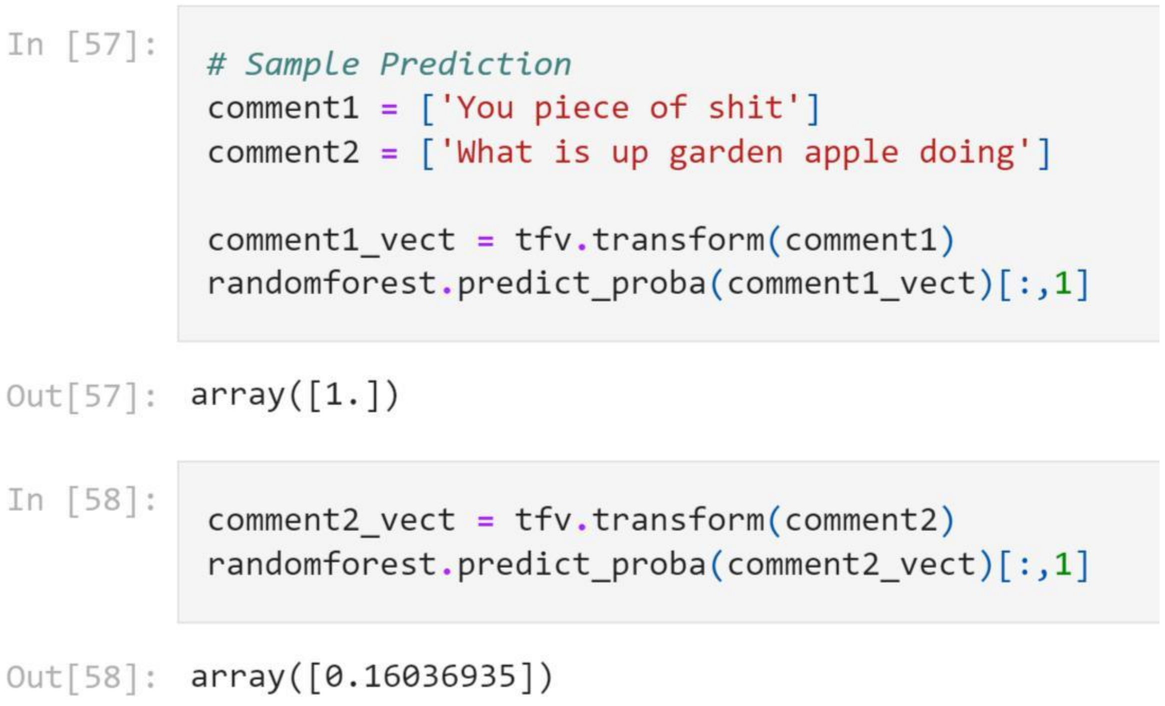
Prob (Obscene): 1.0

Prob (Insult): 1.0

Prob (Threat): 0.1

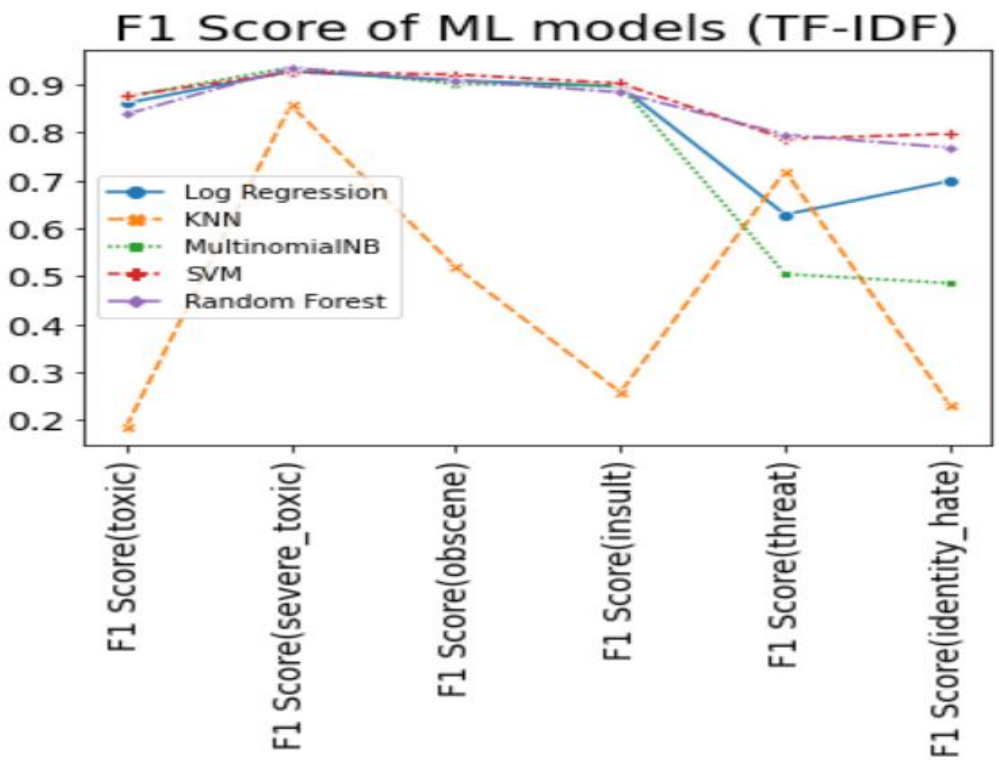
Prob (Identity Hate): 1.0

Screenshot 3: Result of toxic comment



```
In [57]:  # Sample Prediction
          comment1 = ['You piece of shit']
          comment2 = ['What is up garden apple doing']

          comment1_vect = tfv.transform(comment1)
          randomforest.predict_proba(comment1_vect)[:,1]

Out[57]:  array([1.])

In [58]:  comment2_vect = tfv.transform(comment2)
          randomforest.predict_proba(comment2_vect)[:,1]

Out[58]:  array([0.16036935])
```

Screenshot 4: Testing



Screenshot 5: Measured Model’s Accuracy

9. SYSTEM TESTING

9. SYSTEM TESTING

9.1 General

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

9.2 Developing Methodologies

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies
.

9.3 Types of Tests

9.3.1. Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

9.3.2. Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.
Functional testing is centered on the following items:

Valid	: identified classes of valid input must be accepted.
Invalid	: identified classes of invalid input must be rejected.

Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

9.3.3. System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

9.3.4. Performance Test

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

9.3.5. Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

9.3.6. Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node.
- The Route add operation is done only when there is a Route request in need.
- The Status of Nodes information is done automatically in the Cache Updating process.

9.3.7. Build The Test Plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

10. CONCLUSION AND FUTURE SCOPE

10. CONCLUSION AND FUTURE SCOPE

10.1 Conclusion

The project titled as “Toxic Comments Classifier” is a console-based application. This software provides facility for uploading the comments and identify the Weather it has any toxicity or not. This software is developed with scalability in mind. The software is developed with modular approach. All modules in the system have been tested with valid data and invalid data and everything work successfully. Thus, the system has fulfilled all the objectives identified and is able to provide accurate results. The constraints are met and overcome successfully. The system is designed as like it was decided in the design phase. The project gives good idea on developing a full-fledged application satisfying the user requirements. The system is very flexible and versatile. Validation checks induced have greatly reduced errors. Provisions have been made to upgrade the software. The application has been tested with live data and has provided a successful result. Hence the software has proved to work efficiently.

10.2 Future Scope

In future we can use algorithm adaptation methods that transform the algorithms to perform multi-label classification directly. Furthermore, we can also experiment with more complex deep learning algorithms like CNN (convolutional neural network), MLP (multilayer perceptron), and RNN (Recurrent neural networks) in the near future as we believe our approach could reach the top performance when combined with deep learning models.

11. BIBLIOGRAPHY

11. BIBLIOGRAPHY

11.1 Reference

- Revati Sharma , Meetkumar Patel, “Toxic Comment Classification Using Neural Networks and Machine Learning”, Vol. 5, Issue 9, September 2018, DOI 10.17148/IARJSET.2018.597,pg no:47- 52
- Hind Almerekhi, Haewoon Kwak, Bernard J. Jansen, Joni Salminen,” Detecting Toxicity Triggers in Online Discussions”, HT ’19, September 17–20, 2019, Hof, Germany, pg no: 291 – 292.
- NavoneelChakrabarty ,” A Machine Learning Approach to Comment Toxicity Classification”, 2016.

11.2 Websites

- https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
- https://www.aclweb.org/anthology/P12-2018
- https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html
- https://medium.com/@nupurbaghel/toxic-comment-classification-f6e075c3487a

11.3 GitHub Link

- https://conversationai.github.io/
- https://github.com/line-by-line/toxic_comments_classifier