# Project Title

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Transformative Learning
with
TechSaksham – A joint CSR initiative of Microsoft & SAP

by

**Name: Shivani Sunil Sonawane**

**Email ID: shivanisunilsonawane@gmail.com**

Under the Guidance of

**Abdul Aziz Md**

**Master Trainer,Edunet Foundation**

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who contributed directly and indirectly to

the successful completion of this report on the Implementation of SMS Spam Detection System Using NLP.

First and foremost, I extend our deepest gratitude to our project guide – " Abdul Aziz Md" for his invaluable guidance, constant encouragement, and constructive feedback throughout the course of this project. His supervision and guidance proved to be the most valuable to overcome all the hurdles in the completion of the project. Their expertise and insights have been instrumental in shaping the project and overcoming the challenges encountered during development.

I extend my heartfelt thanks to "All the AICTE Technical Team" and my institution and faculty members for providing me with the resources and opportunity to undertake this project. Their insights and knowledge have greatly enhanced my understanding of the subject. Their support and mentorship have been a source of inspiration throughout our internship journey.

Lastly, I acknowledge the importance of the discussions, brainstorming sessions, and shared ideas contributed significantly to achieving the goals of the project. Thank you all for your invaluable contributions.

# ABSTRACT

Nowadays communication plays a major role in everything be it professional or personal. Email communication service is being used extensively because of its free use services, low-cost operations, accessibility, and popularity. Emails have one major security flaw that is anyone can send an email to anyone just by getting their unique user id. This security flaw is being exploited by some businesses and ill-motivated persons for advertising, phishing, malicious purposes, and finally fraud. This produces a kind of email category called SPAM. Spam refers to any email that contains an advertisement, unrelated and frequent emails. These emails are increasing day by day in numbers. Studies show that around 55 percent of all emails are some kind of spam. A lot of effort is being put into this by service providers. Spam is evolving by changing the obvious markers of detection. Moreover, the spam detection of service providers can never be aggressive with classification because it may cause potential information loss to incase of a misclassification. To tackle this problem we present a new and efficient method to detect spam using machine learning and natural language processing. A tool that can detect and classify spam. In addition to that, it also provides information regarding the text provided in a quick view format for user convenience

## TABLE OF CONTENT

## LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1

# Introduction

## 1.1 Problem Statement

Today, Spam has become a major problem in communication over internet. It has been accounted that around 55% of all emails are reported as spam and the number has been growing steadily. Spam which is also known as unsolicited bulk email has led to the increasing use of email as email provides the perfect ways to send the unwanted advertisement or junk newsgroup posting at no cost for the sender. This chances has been extensively exploited by irresponsible organizations and resulting to clutter the mail boxes of millions of people all around the world.

Spam has been a major concern given the offensive content of messages, spam is a waste of time. End user is at risk of deleting legitimate mail by mistake. Moreover, spam also impacted the economical which led some countries to adopt legislation.

## 1.2 Motivation:

The motivation behind this project is to develop an SMS Spam Detection System using Natural Language Processing (NLP) techniques that can effectively detect and filter out spam messages. The system aims to:

1. Improve the accuracy of spam detection

2. Reduce the false positive rate

3. Handle the complexity and variability of spam messages

4. Provide a scalable and efficient solution

## 1.1Objective:

The objectives of this project are:

1. To collect and preprocess a dataset of SMS messages

2. To develop and evaluate NLP-based models for spam detection

3. To compare the performance of different NLP techniques

4. To implement a real-time SMS spam detection system

## 1.2Scope of the Project:

Define the scope and limitations.

Functional Scope

1. Data Collection: Collect a dataset of SMS messages, including spam and legitimate messages.

2. Data Preprocessing: Preprocess the collected data by removing stop words, stemming, and lemmatization.

3. Feature Extraction: Extract relevant features from the preprocessed data using NLP techniques such as bag-of-words, TF-IDF, and word embeddings.

4. Model Development: Develop and train machine learning models using the extracted features to classify SMS messages as spam or legitimate.

5. Model Evaluation: Evaluate the performance of the developed models using metrics such as accuracy, precision, recall, and F1-score.

6. System Implementation: Implement a real-time SMS spam detection system using the developed models.

# CHAPTER 2

# Literature Survey

## 2.1 Objectives :

The objectives of this project are i. To create a ensemble algorithm for classification of spam with highest possible accuracy. ii. To study on how to use machine learning for spam detection. iii. To study how natural language processing techniques can be implemented in spam detection. iv. To provide user with insights of the given text leveraging the created algorithm and NLP.

## 2.2 Limitations :

This Project has certain limitations. i. This can only predict and classify spam but not block it. ii. Analysis can be tricky for some alphanumeric messages and it may struggle with entity detection. iii. Since the data is reasonably large it may take a few seconds to classify and anlayse the message.

## 2.3 Project Scope :

This project needs a coordinated scope of work. i. Combine existing machine learning algorithms to form a better ensemble algorithm. ii. Clean, processing and make use of the dataset for training and testing the model created. iii. Analyse the texts and extract entities for presentat

# Chapter 3

# Working Procedure

The working procedure includes the internal working and the data flow of application.

i. After running the application some procedures are automated.

1. Reading data from file

2. Cleaning the texts

3. Processing

4. Splitting the data

5. Intialising and training the models

ii. The user just needs to provide some data to classify in the area provided.
iii. The provided data undergoes several procedures after submission.

1. Textual Processing

2. Feature Vector conversion

3. Entity extraction

iv. The created vectors are provided to trained models to get predictions.

v. After getting predictions the category predicted by majority will be selected.

vi. The accuracies of that prediction will be calculated

vii. The accuracies and entities extracted from the step 3 will be provided to user. Every time the user gives something new the procedure from step 2 will be repeated.

# CHAPTER 4

# Proposed Methodology

## 4.1System Design

This chapter will explain the specific details on the methodology being used to develop this project. Methodology is an important role as a guide for this project to make sure it is in the right path and working as well as plan. There is different type of methodology used in order to do spam detection and filtering. So, it is important to choose the right and suitable methodology thus it is necessary to understand the application functionality itself.

System Architecture The application overview has been presented below and it gives a basic structure of the application.

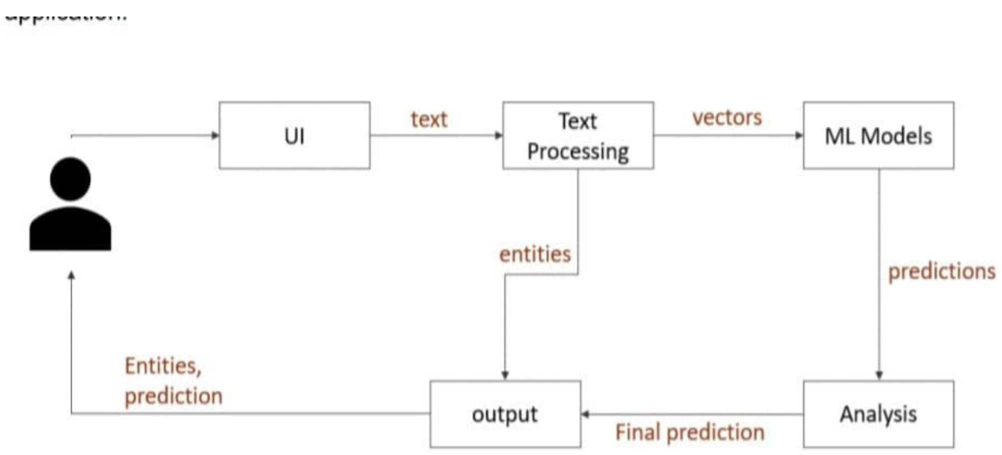


Fig no. 4.1 Architecture

The UI, Text processing and ML Models are the three important modules of this project. Each Module’s explanation has been given in the later sections of this chapter. A more complicated and detailed view of architecture is presented in the workflow section.

## 4.3Modules and Explanation

The Application consists of three modules.

i. UI

ii. ii. Machine Learning

iii. iii. Data Processing

## I. UI Module

a. This Module contains all the functions related to UI(user interface).

b. The user interface of this application is designed using Streamlit library from python based packages.

c. The user inputs are acquired using the functions of this library and forwarded to data processing module for processing and conversion.

d. Finally the output from ML module is sent to this module and from this module to user in visual form.

## II. Machine Learning Module

a. This module is the main module of all three modules.

b. This modules performs everything related to machine learning and results analysis.

c. Some main functions of this module are

i. Training machine learning models.

ii. Testing the model

iv. Determining the respective parameter values for each model.

v. iv. Key-word extraction. v. Final output calculation d. The output from this module is forwarded to UI for providing visual response to user

## III. Data Processing Module

a. The raw data undergoes several modifications in this module for further process. b. Some of the main functions of this module includes

i. Data cleaning

ii. Data merging of datasets

iii. Text Processing using NLP

iv. Conversion of text data into numerical data(feature vectors).

v. Splitting of data.

c. All the data processing is done using Pandas and NumPy libraries.

d. Text processing and text conversion is done using NLTK and scikit-learn libraries.

## 4.4 Requirement Specification

### 4.4.1 Hardware Requirements:

PC/Laptop

Ram – 8 Gig

Storage – 100-200 Mb

### 4.4.2 Software Requirements:

OS – Windows 7 and above

Code Editor – Pycharm, VS Code, Built in IDE

Anaconda environment with packages nltk, numpy, pandas, sklearn, tkinter, nltk data. Supported browser such as chrome, firefox, opera etc.

**Work Flow :**

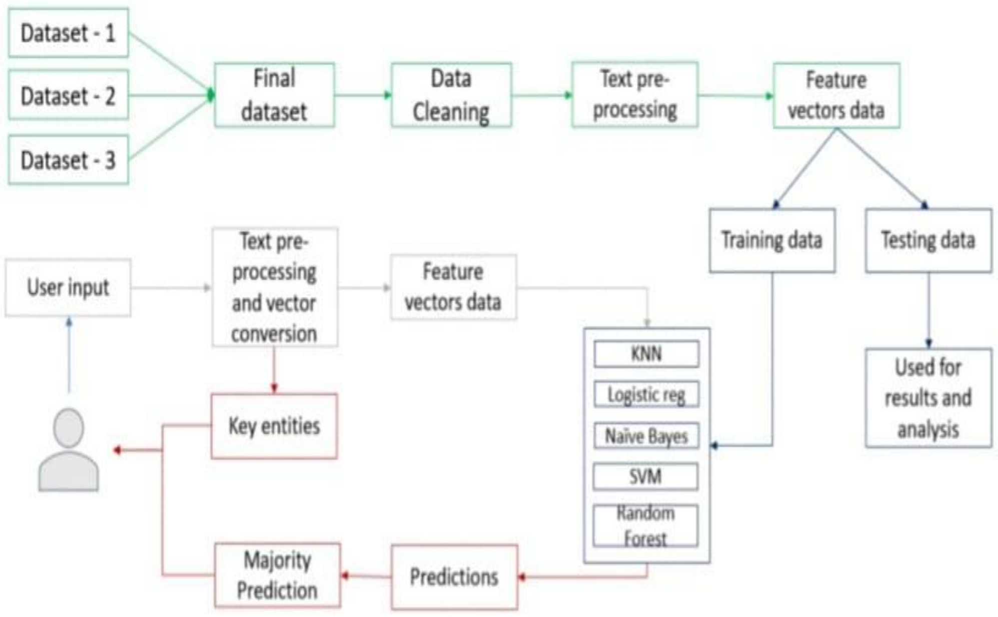


fig no. 4.2 Workflow

### 4.4.3 Data Collection and Description

17 ● Data plays an important role when it comes to prediction and classification, the more the data the more the accuracy will be.

● The data used in this project is completely open-source and has been taken from various resources like Kaggle and UCI .

● For the purpose of accuracy and diversity in data multiple datasets are taken. 2 datasets containing approximately over 12000 mails and their labels are used for training and testing the application.

● 6000 spam mails are taken for generalisation of data and to increase the accuracy.

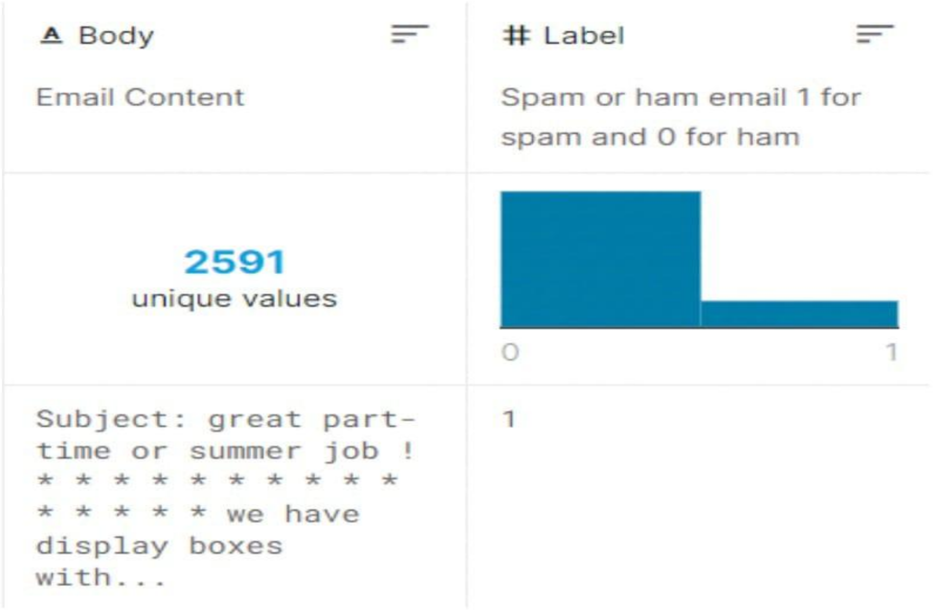
### 4.4.4Data Description

**Dataset** : enronSpamSubset.

**Source**: Kaggle

**Description :** this dataset is part of a larger dataset called enron. This dataset contains a set of spam and non-spam emails with 0 for non spam and 1 for spam in label attribute.

**Composition** : Unique values : 9687 Spam values : 5000 Non-spam values : 4687 fig no. 4.3 enron spam



Body

Email Content

2591
unique values

Subject: great part-time or summer job ! * * * * * * * * * * * * * * * we have display boxes with...

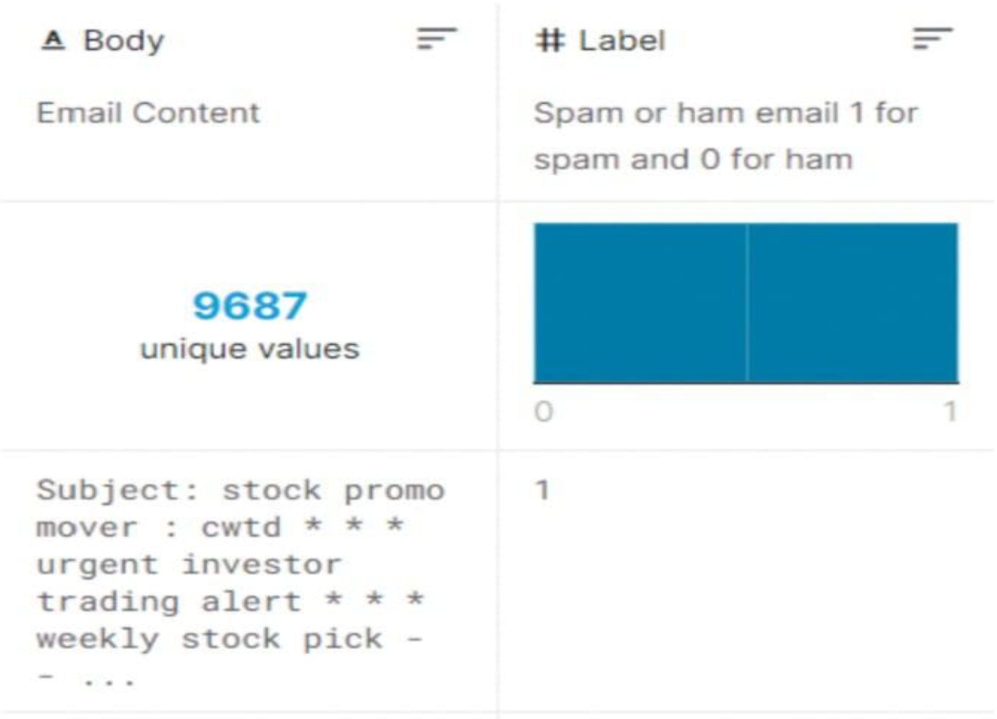
Label

Spam or ham email 1 for spam and 0 for ham

1

**Dataset** : lingspam.

**Source** : Kaggle



| Body | Label |
|---|---|
| Email Content | Spam or ham email 1 for spam and 0 for ham |
| 9687<br>unique values | |
| Subject: stock promo mover : cwtd * * * urgent investor trading alert * * * weekly stock pick - - ... | 1 |

fig no. 4.4 lingspam

**Description** : This dataset is part of a larger dataset called Enron1 which contains emails classified as spam or ham(not-spam).

**Composition** : Unique values : 2591 Spam values : 419 Non-spam values : 2172

### 4.4.5 Data Processing

**Overall data processing**

It consists of two main tasks

● Dataset cleaning It includes tasks such as removal of outliers, null value removal, removal of unwanted features from data.

● Dataset Merging After data cleaning, the datasets are merged to form a single dataset containing only two features(text, label). Data cleaning, Data Merging these procedures are completely done using Pandas library**.**

#### 4.4.5.1Textual data processing

● Tag removal

Removing all kinds of tags and unknown characters from text using regular expressions through Regex library.

● Sentencing, tokenization Breaking down the text(email/SMS) into sentences and then into tokens(words). This process is done using NLTK pre-processing library of python. ● Stop word removal Stop words such as of , a ,be , … are removed using stopwords NLTK library of python.

● Lemmatization Words are converted into their base forms using lemmatization and pos-tagging This process gives key-words through entity extraction. This process is done using chunking in regex and NLTK lemmatization.

● Sentence formation The lemmatized tokens are combined to form a sentence. This sentence is essentially a sentence converted into its base form and removing stop words. Then all the sentences are combined to form a text.
● While the overall data processing is done only to datasets, the textual processing is done to both training data, testing data and also user input data.

### 4.4.5.2 Feature Vector Formation

● The texts are converted into feature vectors(numerical data) using the words present in all the texts combined.

● This process is done using countvectorization of NLTK library.

● The feature vectors can be formed using two language models Bag of Words and Term Frequency-inverse Document Frequency.

### 4.4.5.3 Bag of Words

Bag of words is a language model used mainly in text classification. A bag of words represents the text in a numerical form. The two things required for Bag of Words are

- A vocabulary of words known to us.

- A way to measure the presence of words.

Ex: a few lines from the book “A Tale of Two Cities” by Charles Dickens.

**“ It was the best of times,**

**it was the worst of times,**

**it was the age of wisdom**

**it was the age of foolishness, ”**

unique words here (ignoring case and punctuation) are: [ “it”, “was”, “the”, “best”, “of”, “times”, “worst”,“age”, “wisdom”, “foolishness” ] The next step is scoring words present in every document.

After scoring the four lines from the above stanza can be represented in vector form as

“It was the best of times“ = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]

"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]

"it was the age of foolishness"= [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

This is the main process behind the bag of words but in reality the vocabulary even from a couple of documents is very large and words repeating frequently and important in nature are taken and remaining are removed during the text processing stage.

## 4.4.5.4 Term Frequency-inverse document frequency

Term frequency-inverse document frequency of a word is a measurement of the importance of a word. It compares the repentance of words to the collection of documentsand calculates the score.

Terminology for the below formulae:

t – term(word)

d – document(set of words)

N – count of documents

The TF-IDF process consists of various activities listed below.

### i)Term Frequency

The count of appearance of a particular word in a document is called term frequency $\boldsymbol{tf(t, d)} = \boldsymbol{count\ of\ t\ in\ d/\ number\ of\ words\ in\ d}$

**ii) Document Frequency** Document frequency is the count of documents the word was detected in. We consider one instance of a word and it doesn’t matter if the word is present multiple times.

$\boldsymbol{df(t)} = \boldsymbol{occurrence\ of\ t\ in\ documents}$

**iii) Inverse Document Frequency**

- IDF is the inverse of document frequency.

- It measures the importance of a term t considering the information it contributes. Every term is considered equally important but certain terms such as (are, if, a, be, that, ..) provide little information about the document. The inverse document frequency factor reduces the importance of words/terms that has highe recurrence and increases the importance of words/terms that are rare.

$\boldsymbol{idf}(\boldsymbol{t}) = \boldsymbol{N}/\boldsymbol{df}$

Finally, the TF-IDF can be calculated by combining the term frequency and inverse document frequency.

$\boldsymbol{tf\_idf}(\boldsymbol{t}, \boldsymbol{d}) = \boldsymbol{tf}(\boldsymbol{t}, \boldsymbol{d}) * \mathbf{lo\ g}\ (\boldsymbol{N}/(\boldsymbol{df} + \mathbf{1}))$

the process can be explained using the following example:

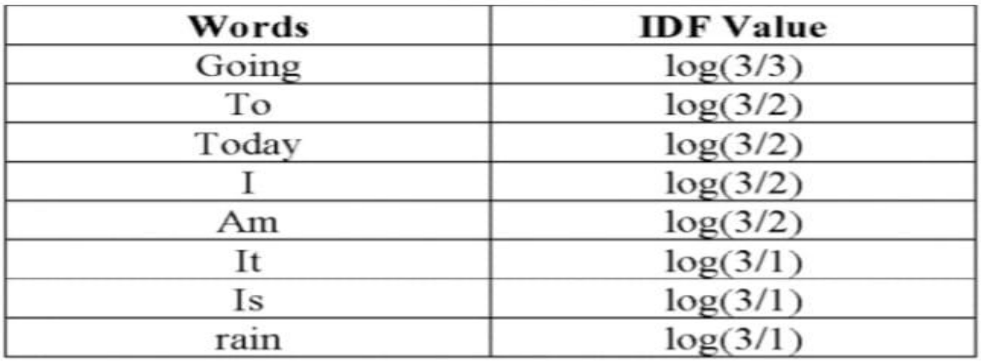
**“Document 1 It is going to rain today.**

**Document 2 Today I am not going outside.**

**Document 3 I am going to watch the season premiere.”**

The Bag of words of the above sentences is [going:3, to:2, today:2, i:2, am:2, it:1, is:1, rain:1

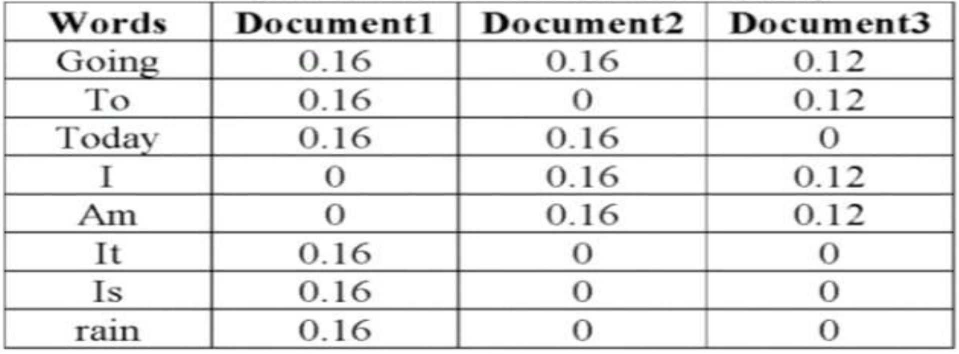
table no. 4.1 Term frequency



| Words | IDF Value |
|---|---|
| Going | log(3/3) |
| To | log(3/2) |
| Today | log(3/2) |
| I | log(3/2) |
| Am | log(3/2) |
| It | log(3/1) |
| Is | log(3/1) |
| rain | log(3/1) |

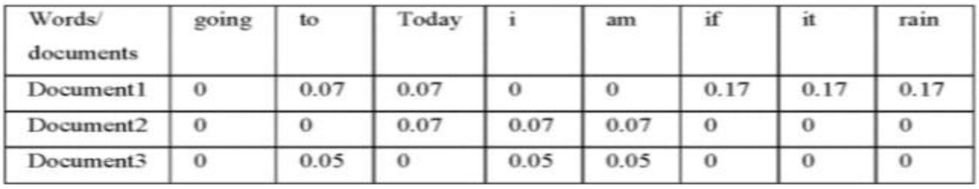
Then finding the inverse document frequency

table no. 4.2 inverse document frequency



| Words | Document1 | Document2 | Document3 |
|---|---|---|---|
| Going | 0.16 | 0.16 | 0.12 |
| To | 0.16 | 0 | 0.12 |
| Today | 0.16 | 0.16 | 0 |
| I | 0 | 0.16 | 0.12 |
| Am | 0 | 0.16 | 0.12 |
| It | 0.16 | 0 | 0 |
| Is | 0.16 | 0 | 0 |
| rain | 0.16 | 0 | 0 |

Applying the final equation the values of tf-idf becomes



| Words/ documents | going | to | Today | i | am | if | it | rain |
|---|---|---|---|---|---|---|---|---|
| Document1 | 0 | 0.07 | 0.07 | 0 | 0 | 0.17 | 0.17 | 0.17 |
| Document2 | 0 | 0 | 0.07 | 0.07 | 0.07 | 0 | 0 | 0 |
| Document3 | 0 | 0.05 | 0 | 0.05 | 0.05 | 0 | 0 | 0 |

table no. 4.3 TF-IDF

## 4.4.5.4Machine Learning

### Introduction

Machine Learning is process in which the computer performs certain tasks without giving instructions. In this case the models takes the training data and train on them. Then depending on the trained data any new unknown data will be processed based on the ruled derived from the trained data.

After completing the countvectorization and TF-IDF stages in the workflow the data is converted into vector form(numerical form) which is used for training and testing models.

For our study various machine learning models are compared to determine which method is more suitable for this task. The models used for the study include Logistic Regression, Naïve Bayes, Random Forest Classifier, K Nearest Neighbors, and Support Vector Machine Classifier and a proposed model which was created using an ensemble approach.

**4..4.5.5Algorithms** a combination of 5 algorithms are used for the classifications.

**4.4.5.6 Naïve Bayes Classifier** A naïve Bayes classifier is a supervised probabilistic machine learning model that is used for classification tasks. The main principle behind this model is the Bayes theorem. Bayes Theorem: Naive Bayes is a classification technique that is based on Bayes’ Theorem with an assumption that all the features that predict the target value are independent of each other. It calculates the probability of each class and then picks the one with the highest probability.

Naive Bayes classifier assumes that the features we use to predict the target are independent and do not affect each other. Though the independence assumption is never correct in real-world data, but often works well in practice. so that it is called “Naive” [14]

. P(A│B)=(P(B│A)P(A))/P(B) P(A|B) is the probability of hypothesis A given the data B. This is called the posterior probability.

P(B|A) is the probability of data B given that hypothesis A was true.

P(A) is the probability of hypothesis A being true (regardless of the data). This is called the prior probability of A.

P(B) is the probability of the data (regardless of the hypothesis) [15].

Naïve Bayes classifiers are mostly used for text classification. The limitation of the Naïve Bayes model is that it treats every word in a text as independent and is equal in importance but every word cannot be treated equally important because articles and nouns are not the same when it comes to language. But due to its classification efficiency, this model is used in combination with other language processing techniques.

## 4.4.5.6 Logistic Regression

Logistic Regression is a “Supervised machine learning” algorithm that can be used to model the probability of a certain class or event. It is used when the data is linearly separable and the outcome is binary or dichotomous [17]. The probabilities are calculated using a sigmoid function.

For example, let us take a problem where data has n features. We need to fit a line for the given data and this line can be represented by the equation z=b_0+b_1 x_1+b_2 x_2+b_3 x_3….+b_n x_n

here z = odds

generally, odds are calculated

**odds=p(event occurring)/p(event not occurring)**

**Sigmoid Function:**

A sigmoid function is a special form of logistic function hence the name logistic regression. The logarithm of odds is calculated and fed into the sigmoid function to get continuous probability ranging from 0 to 1. The logarithm of odds can be calculated by

**log(odds)=dot(features,coefficients)+intercept**

and these log_odds are used in the sigmoid function to get probability.

**h(z)=1/(1+e^(-z) )**

The output of the sigmoid function is an integer in the range 0 to 1 which is used to determine which class the sample belongs to. Generally, 0.5 is considered as the limit below which it is considered a NO, and 0.5 or higher will be considered a YES. But the border can be adjusted based on the requirement.

# CHAPTER 4

# Implementation and Result

## 4.1Snap Shots of Result:

### Language Model Selection :

While selecting the best language model the data has been converted into both types of vectors and then the models been tested for to determine the best model for classifying spam.

The results from individual models are presented in the experimentation section under methodology. Now comparing the results from the models.

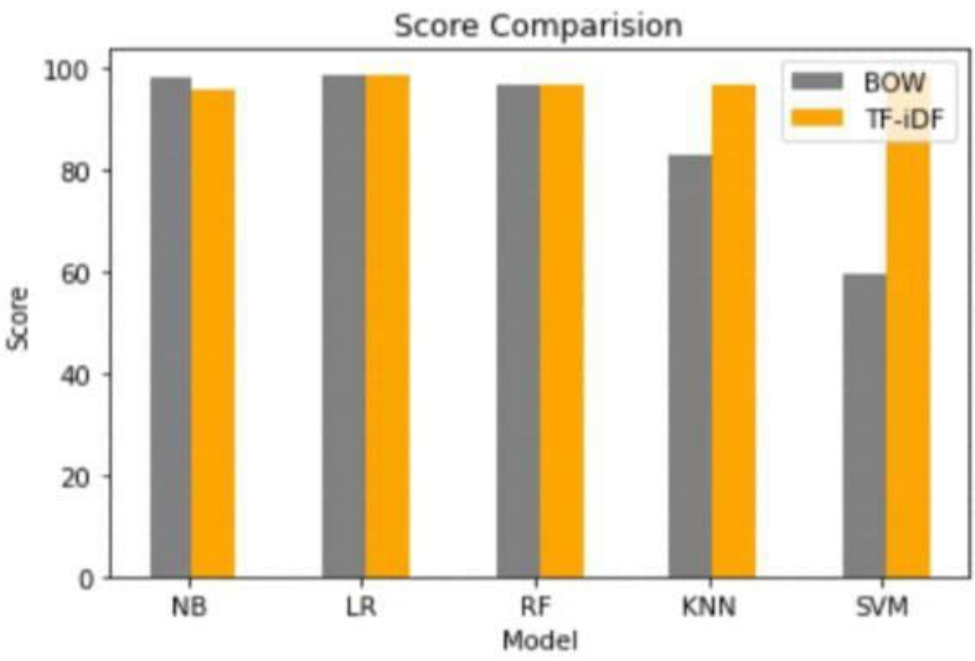


fig no. 4.1 Bow vs TF-IDF (Cumulative) From the figure it is clear that TF-IDF proves to be better than BoW in every model tested. Hence TF-IDF has been selected as the primary language model for textual data conversion in feature vector formation.

## 4.2Proposed Model results

To determine which model is effective we used three metrics Accuracy, Precision, and F1score. The resulted values for the proposed model are

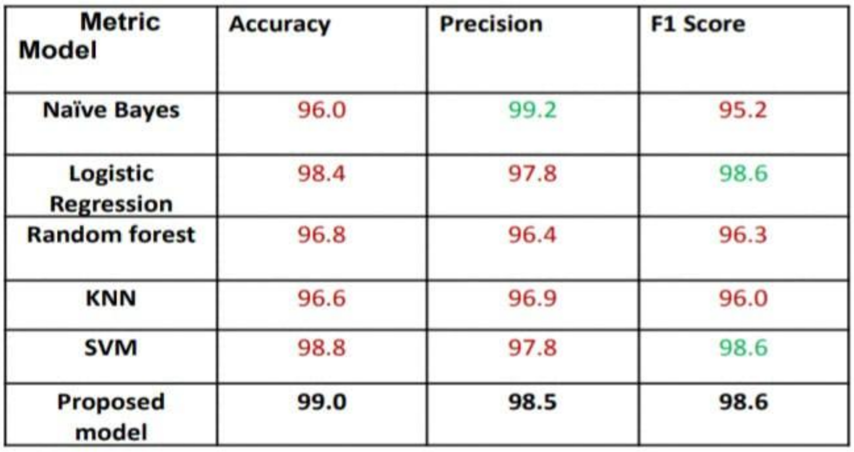
**Accuracy – 99.0**

**Precision – 98.5**

**F1 Score – 98.6 5.3**

## 4.3Comparison

The results from the proposed model has been compared with all the models individually in tabular form to illustrate the differences clearly.



| Metric<br>Model | Accuracy | Precision | F1 Score |
|---|---|---|---|
| Naïve Bayes | 96.0 | 99.2 | 95.2 |
| Logistic Regression | 98.4 | 97.8 | 98.6 |
| Random forest | 96.8 | 96.4 | 96.3 |
| KNN | 96.6 | 96.9 | 96.0 |
| SVM | 98.8 | 97.8 | 98.6 |
| Proposed model | **99.0** | **98.5** | **98.6** |

Table no. 4.1 Models and results The color RED indicates that the value is lower than the proposed model and GREEN indicates equal or higher. Here we can observe that our proposed model outperforms almost every other model in every metric. Only one model(naïve Bayes) has slightly higher accuracy than our model but it is considerably lagging in other metrics. The results are visually presented below for easier understanding and comparison.

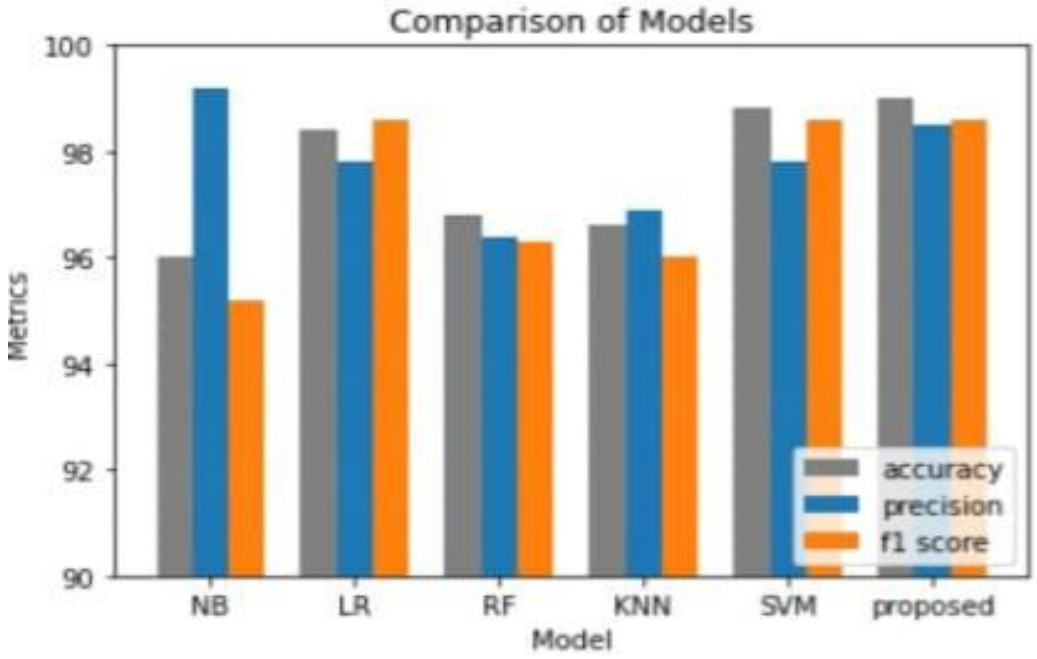
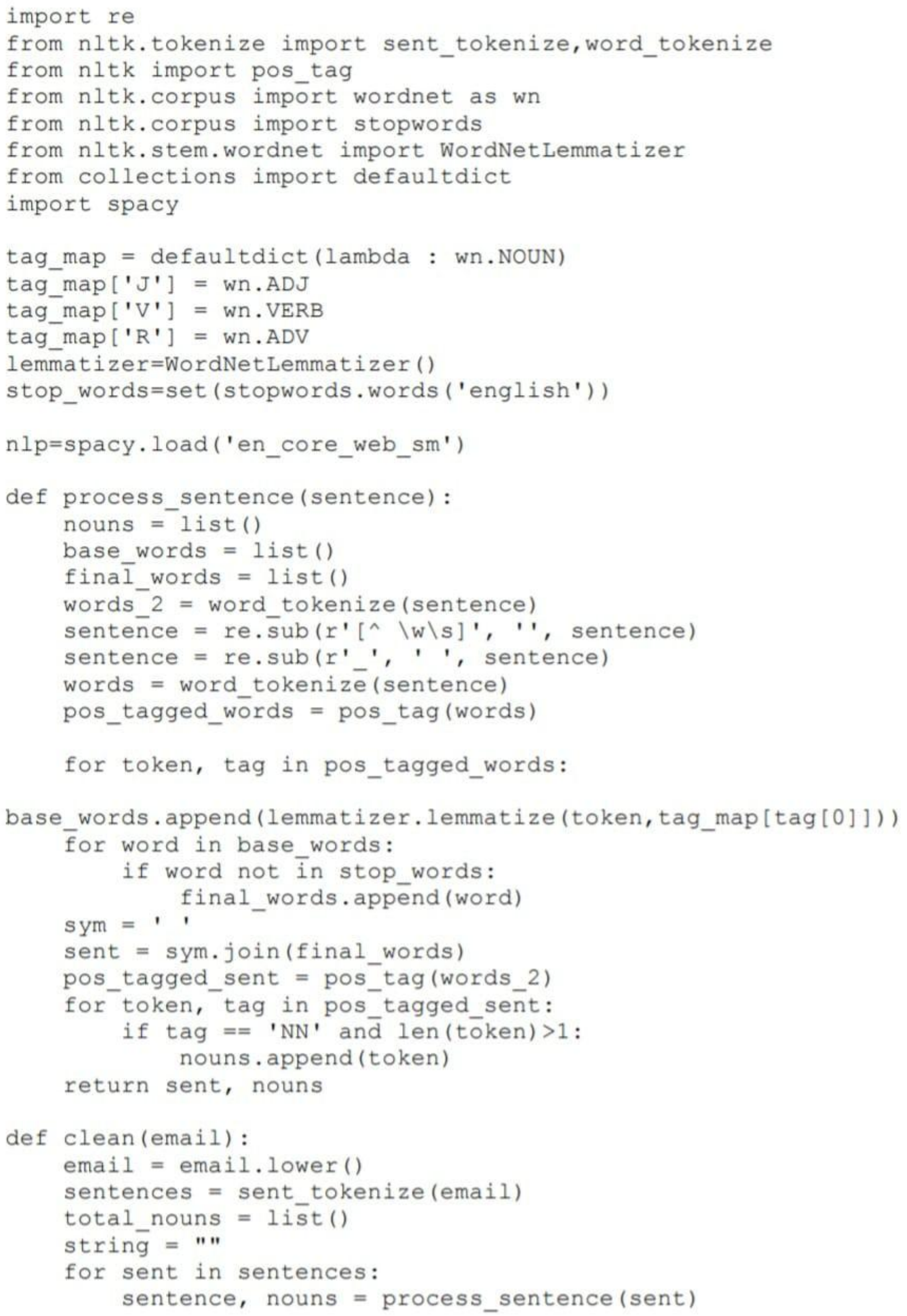


fig no.4.2 Comparision of Models From the above comparison barchart we can clearly see that all models individually are not as efficient as the proposed method.

### 4.4 GitHub Link for Code:

## A.Source code

# 1.Module-Data Processing



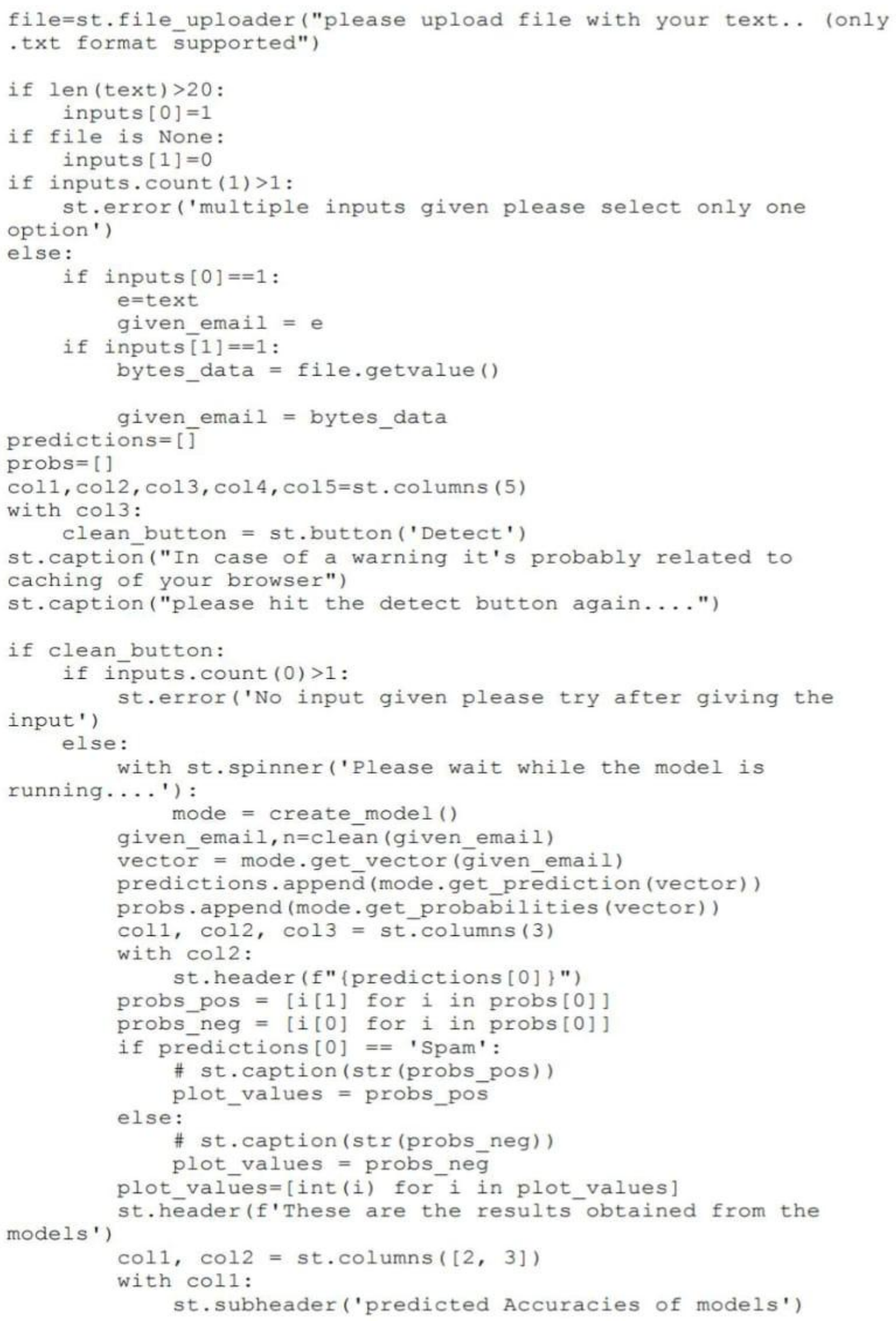
```
import re
from nltk.tokenize import sent_tokenize,word_tokenize
from nltk import pos_tag
from nltk.corpus import wordnet as wn
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from collections import defaultdict
import spacy

tag_map = defaultdict(lambda : wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV
lemmatizer=WordNetLemmatizer()
stop_words=set(stopwords.words('english'))

nlp=spacy.load('en_core_web_sm')

def process_sentence(sentence):
    nouns = list()
    base_words = list()
    final_words = list()
    words_2 = word_tokenize(sentence)
    sentence = re.sub(r'[^ \w\s]', '', sentence)
    sentence = re.sub(r'_', ' ', sentence)
    words = word_tokenize(sentence)
    pos_tagged_words = pos_tag(words)

    for token, tag in pos_tagged_words:

base_words.append(lemmatizer.lemmatize(token,tag_map[tag[0]]))
    for word in base_words:
        if word not in stop_words:
            final_words.append(word)
    sym = ' '
    sent = sym.join(final_words)
    pos_tagged_sent = pos_tag(words_2)
    for token, tag in pos_tagged_sent:
        if tag == 'NN' and len(token)>1:
            nouns.append(token)
    return sent, nouns

def clean(email):
    email = email.lower()
    sentences = sent_tokenize(email)
    total_nouns = list()
    string = ""
    for sent in sentences:
        sentence, nouns = process_sentence(sent)
        string += " " + sentence
        total_nouns += nouns
    return string, nouns


def ents(text):
    doc = nlp(text)
    expls = dict()
    if doc.ents:
        for ent in doc.ents:
            labels = list(expls.keys())
            label = ent.label_
            word = ent.text
            if label in labels:
                words = expls[label]
                words.append(word)
                expls[label] = words
            else:
                expls[label] = [word]
        return expls
    else:
        return 'no'
```

## 2. Module – Machine Learning

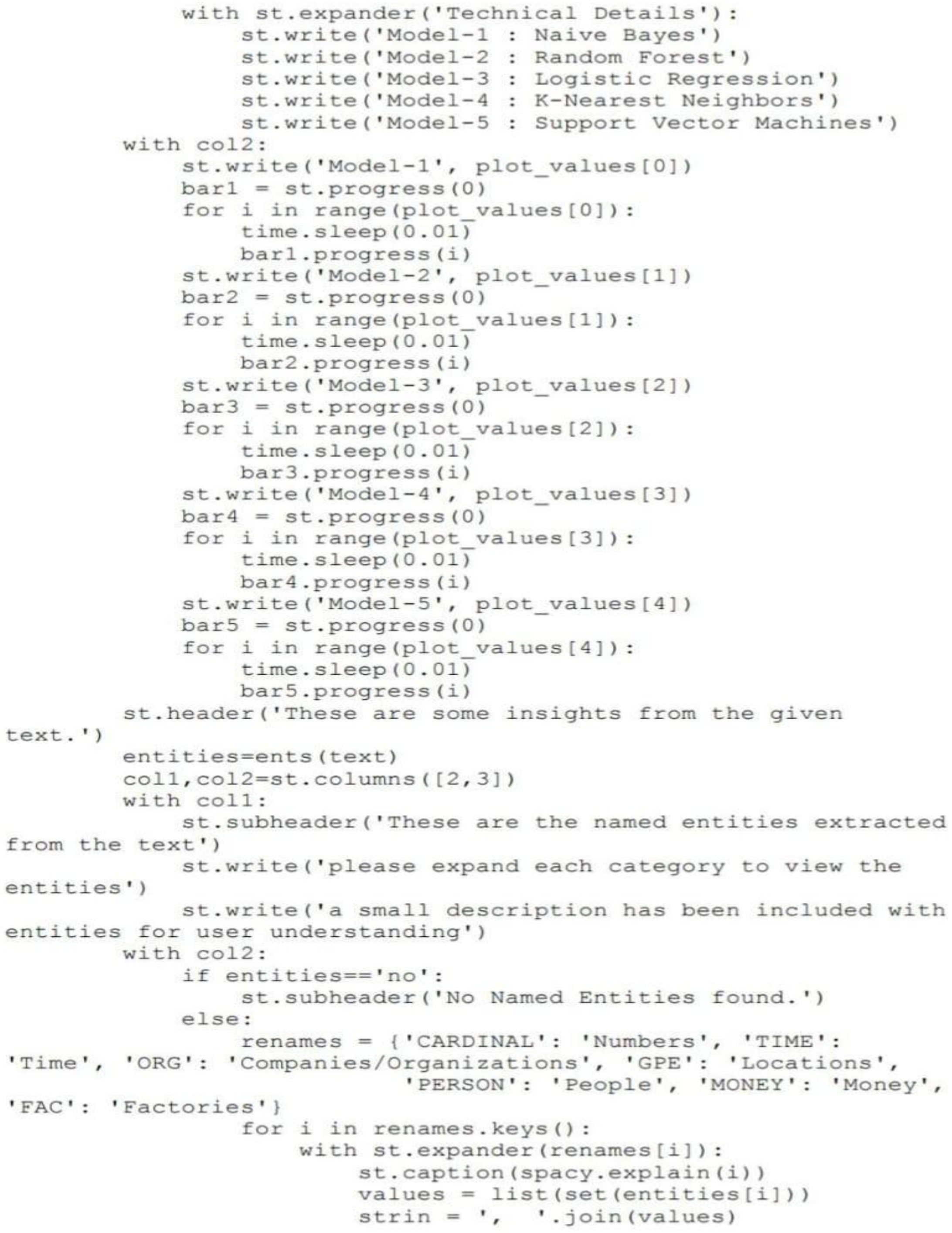
```
from sklearn.feature_extraction.text import
CountVectorizer,TfidfVectorizer
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
import pandas as pd

class model:
    def __init__(self):
        self.df = pd.read_csv('Cleaned_Data.csv')
        self.df['Email'] = self.df.Email.apply(lambda email:
np.str_(email))
        self.Data = self.df.Email
        self.Labels = self.df.Label
        self.training_data, self.testing_data,
self.training_labels, self.testing_labels =
train_test_split(self.Data,self.Labels,random_state=10)
        self.training_data_list = self.training_data.to_list()
        self.vectorizer = TfidfVectorizer()
        self.training_vectors =
self.vectorizer.fit_transform(self.training_data_list)
        self.model_nb = MultinomialNB()
        self.model_svm = SVC(probability=True)
        self.model_lr = LogisticRegression()
        self.model_knn = KNeighborsClassifier(n_neighbors=9)
        self.model_rf = RandomForestClassifier(n_estimators=19)
        self.model_nb.fit(self.training_vectors,
self.training_labels)
        self.model_lr.fit(self.training_vectors,
self.training_labels)
        self.model_rf.fit(self.training_vectors,
self.training_labels)
        self.model_knn.fit(self.training_vectors,
self.training_labels)
        self.model_svm.fit(self.training_vectors,
self.training_labels)
    def get_prediction(self,vector):
        pred_nb=self.model_nb.predict(vector)[0]
        pred_lr=self.model_lr.predict(vector)[0]
        pred_rf=self.model_rf.predict(vector)[0]
        pred_svm=self.model_svm.predict(vector)[0]
        pred_knn=self.model_knn.predict(vector)[0]
        preds=[pred_nb,pred_lr,pred_rf,pred_svm,pred_knn]
        spam_counts=preds.count(1)
        if spam_counts>=3:
            return 'Spam'
        return 'Non-Spam'
    def get_probabilities(self,vector):
        prob_nb=self.model_nb.predict_proba(vector)[0]*100
        prob_lr = self.model_lr.predict_proba(vector)[0] * 100
        prob_rf = self.model_rf.predict_proba(vector)[0] * 100
        prob_knn = self.model_knn.predict_proba(vector)[0] * 100
        prob_svm = self.model_svm.predict_proba(vector)[0] * 100
        return [prob_nb,prob_lr,prob_rf,prob_knn,prob_svm]

    def get_vector(self,text):
        return self.vectorizer.transform([text])
```

## 3. Module – User interface

```
import time
from ML import model
import streamlit as st
from DP import *
import matplotlib.pyplot as plt
import seaborn as sns
inputs=[0,1]
@st.cache()
def create_model():
    mode=model()
    return mode
col1,col2,col3,col4,col5=st.columns(5)
with col3:
    st.title("Spade")
st.write('welcome to Spade...')
st.write('A Spam Detection algorithm based on Machine Learning
and Natural Language Processing')
text=st.text_area('please provide email/text you wish to
classify',height=400,placeholder='type/paste more than 50
characters here')
```



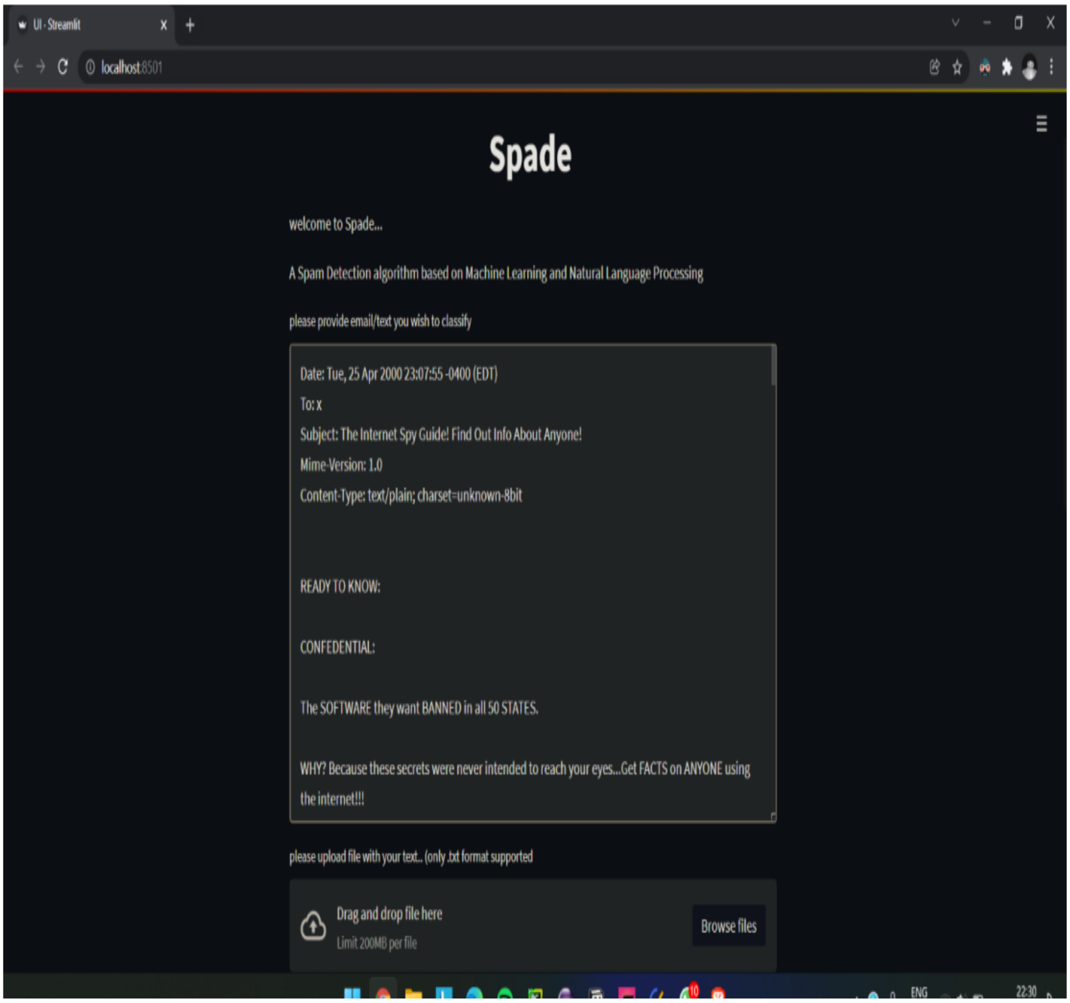
```
file=st.file_uploader("please upload file with your text.. (only
.txt format supported")

if len(text)>20:
    inputs[0]=1
if file is None:
    inputs[1]=0
if inputs.count(1)>1:
    st.error('multiple inputs given please select only one
option')
else:
    if inputs[0]==1:
        e=text
        given_email = e
    if inputs[1]==1:
        bytes_data = file.getvalue()

        given_email = bytes_data
predictions=[]
probs=[]
col1,col2,col3,col4,col5=st.columns(5)
with col3:
    clean_button = st.button('Detect')
st.caption("In case of a warning it's probably related to
caching of your browser")
st.caption("please hit the detect button again....")

if clean_button:
    if inputs.count(0)>1:
        st.error('No input given please try after giving the
input')
    else:
        with st.spinner('Please wait while the model is
running....'):
            mode = create_model()
        given_email,n=clean(given_email)
        vector = mode.get_vector(given_email)
        predictions.append(mode.get_prediction(vector))
        probs.append(mode.get_probabilities(vector))
        col1, col2, col3 = st.columns(3)
        with col2:
            st.header(f"{predictions[0]}")
        probs_pos = [i[1] for i in probs[0]]
        probs_neg = [i[0] for i in probs[0]]
        if predictions[0] == 'Spam':
            # st.caption(str(probs_pos))
            plot_values = probs_pos
        else:
            # st.caption(str(probs_neg))
            plot_values = probs_neg
        plot_values=[int(i) for i in plot_values]
        st.header(f'These are the results obtained from the
models')
        col1, col2 = st.columns([2, 3])
        with col1:
            st.subheader('predicted Accuracies of models')
```



```
            with st.expander('Technical Details'):
                st.write('Model-1 : Naive Bayes')
                st.write('Model-2 : Random Forest')
                st.write('Model-3 : Logistic Regression')
                st.write('Model-4 : K-Nearest Neighbors')
                st.write('Model-5 : Support Vector Machines')
        with col2:
            st.write('Model-1', plot_values[0])
            bar1 = st.progress(0)
            for i in range(plot_values[0]):
                time.sleep(0.01)
                bar1.progress(i)
            st.write('Model-2', plot_values[1])
            bar2 = st.progress(0)
            for i in range(plot_values[1]):
                time.sleep(0.01)
                bar2.progress(i)
            st.write('Model-3', plot_values[2])
            bar3 = st.progress(0)
            for i in range(plot_values[2]):
                time.sleep(0.01)
                bar3.progress(i)
            st.write('Model-4', plot_values[3])
            bar4 = st.progress(0)
            for i in range(plot_values[3]):
                time.sleep(0.01)
                bar4.progress(i)
            st.write('Model-5', plot_values[4])
            bar5 = st.progress(0)
            for i in range(plot_values[4]):
                time.sleep(0.01)
                bar5.progress(i)
        st.header('These are some insights from the given
text.')
        entities=ents(text)
        col1,col2=st.columns([2,3])
        with col1:
            st.subheader('These are the named entities extracted
from the text')
            st.write('please expand each category to view the
entities')
            st.write('a small description has been included with
entities for user understanding')
        with col2:
            if entities=='no':
                st.subheader('No Named Entities found.')
            else:
                renames = {'CARDINAL': 'Numbers', 'TIME':
'Time', 'ORG': 'Companies/Organizations', 'GPE': 'Locations',
                           'PERSON': 'People', 'MONEY': 'Money',
'FAC': 'Factories'}
                for i in renames.keys():
                    with st.expander(renames[i]):
                        st.caption(spacy.explain(i))
                        values = list(set(entities[i]))
                        strin = ',  '.join(values)
```

```
st.write(strin)
```

## B. Screenshots



welcome to Spade...

A Spam Detection algorithm based on Machine Learning and Natural Language Processing

please provide email/text you wish to classify

Date: Tue, 25 Apr 2000 23:07:55 -0400 (EDT)
To: x
Subject: The Internet Spy Guide! Find Out Info About Anyone!
Mime-Version: 1.0
Content-Type: text/plain; charset=unknown-8bit

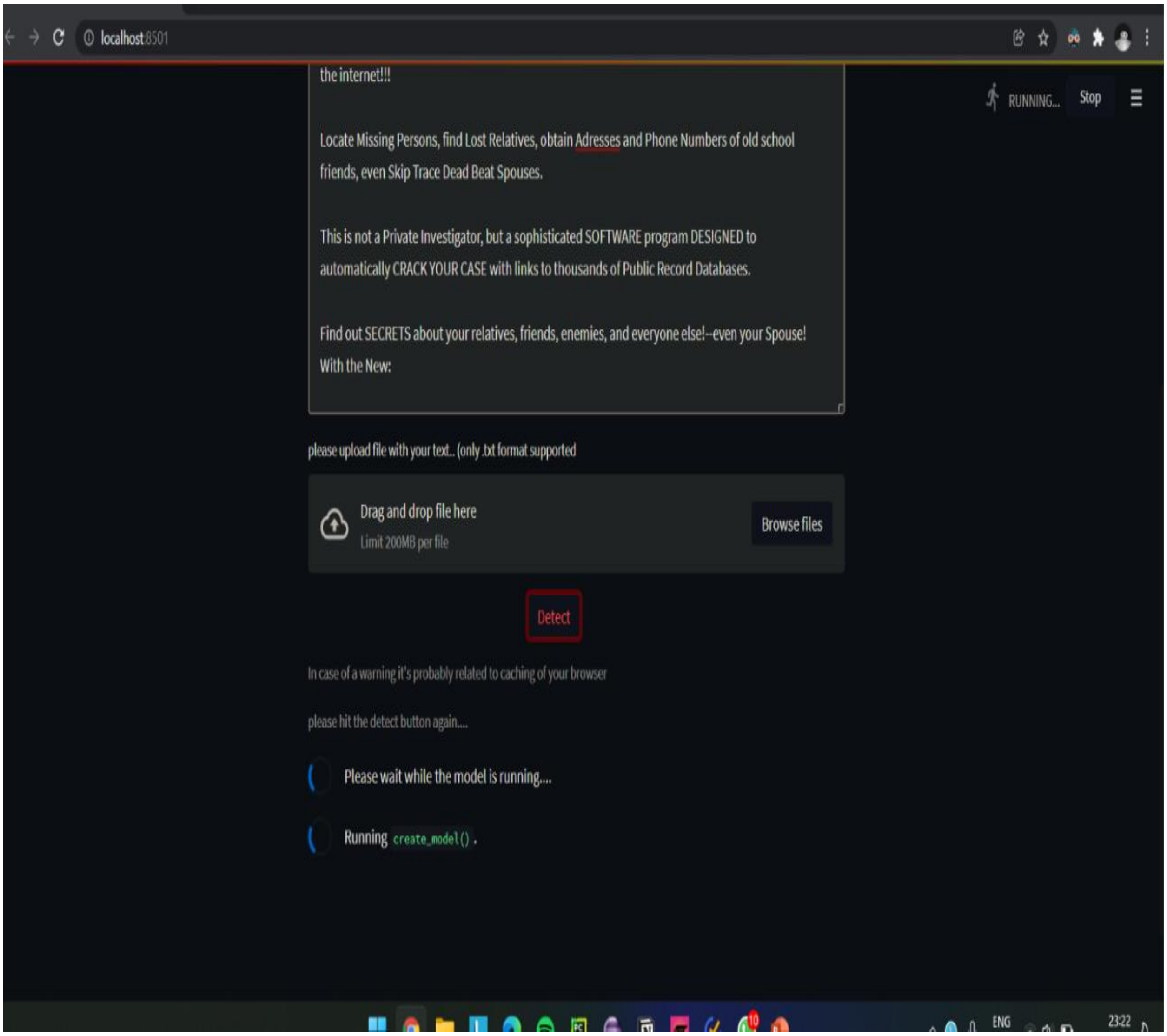
READY TO KNOW:

CONFEDENTIAL:

The SOFTWARE they want BANNED in all 50 STATES.

WHY? Because these secrets were never intended to reach your eyes...Get FACTS on ANYONE using the internet!!!

please upload file with your text.. (only .txt format supported



the internet!!!

Locate Missing Persons, find Lost Relatives, obtain Adresses and Phone Numbers of old school friends, even Skip Trace Dead Beat Spouses.

This is not a Private Investigator, but a sophisticated SOFTWARE program DESIGNED to automatically CRACK YOUR CASE with links to thousands of Public Record Databases.

Find out SECRETS about your relatives, friends, enemies, and everyone else!--even your Spouse! With the New:

please upload file with your text.. (only .txt format supported

Drag and drop file here
Limit 200MB per file

Browse files

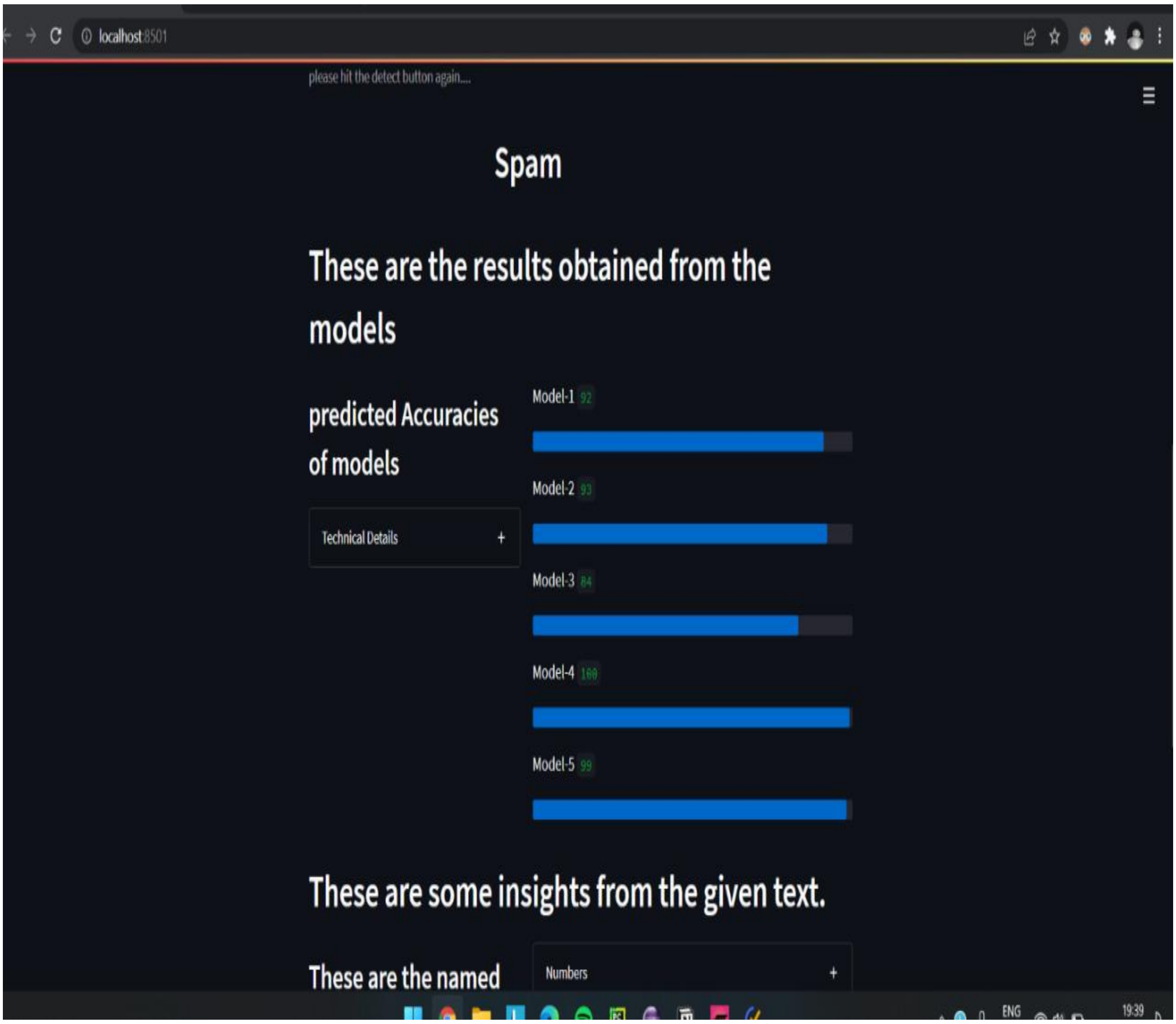
Detect

In case of a warning it's probably related to caching of your browser

please hit the detect button again....

Please wait while the model is running....

Running `create_model()`.



please hit the detect button again....

Spam

These are the results obtained from the models

predicted Accuracies of models

Model-1 92

Model-2 93

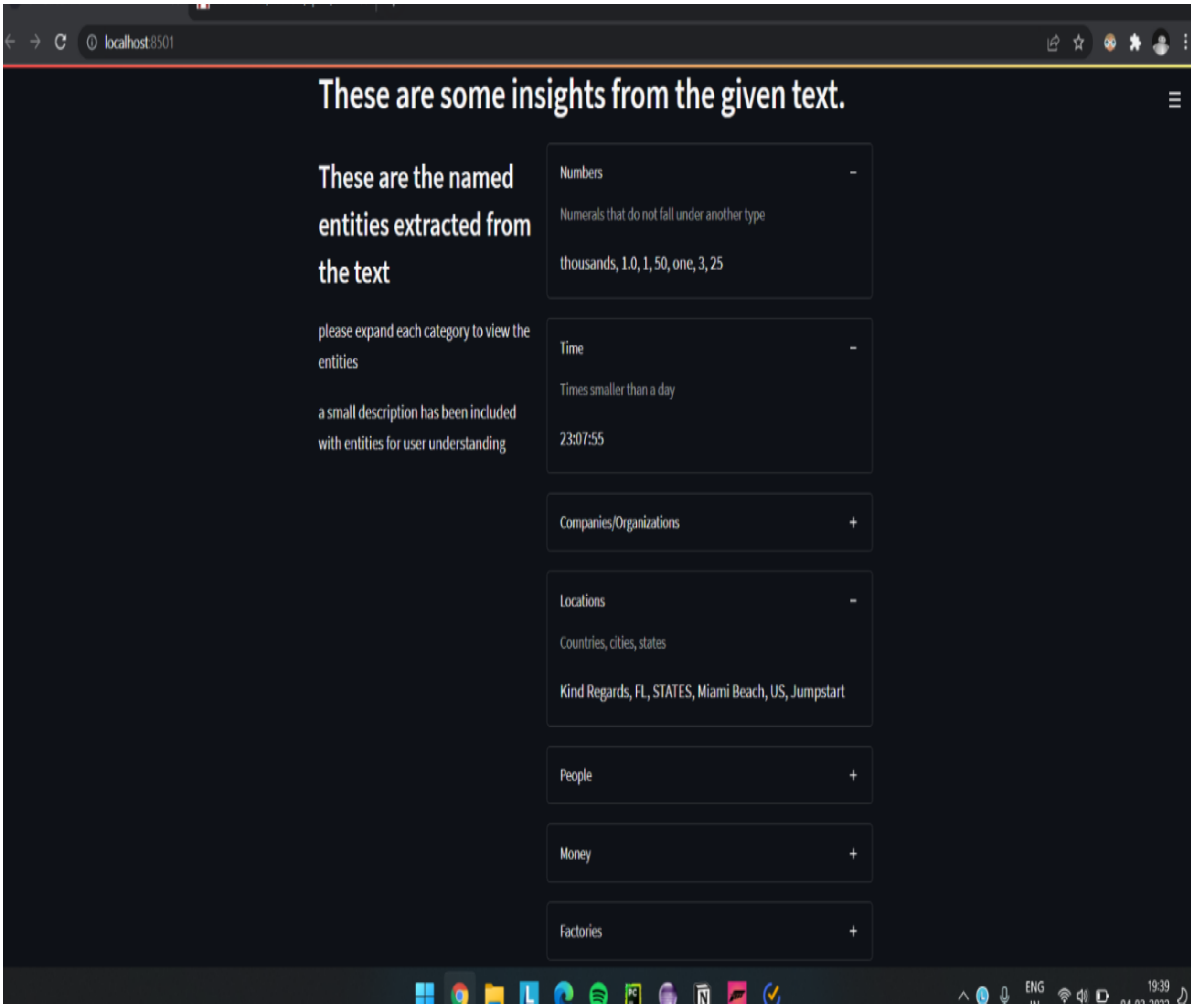
Model-3 84

Model-4 100

Model-5 99

These are some insights from the given text.

These are the named



These are some insights from the given text.

These are the named entities extracted from the text

please expand each category to view the entities

a small description has been included with entities for user understanding

Numbers -

Numerals that do not fall under another type

thousands, 1.0, 1, 50, one, 3, 25

Time -

Times smaller than a day

23:07:55

Companies/Organizations +

Locations -

Countries, cities, states

Kind Regards, FL, STATES, Miami Beach, US, Jumpstart

People +

Money +

Factories +

# CHAPTER 5

## Discussion and Conclusion

## 5.1 Future Work:

There are numerous appilcations to machine learning and natural language processing and when combined they can solve some of the most troubling problems concerned with texts. This application can be scaled to intake text in bulk so that classification can be done more affectively in some public sites.

Other contexts such as negative, phishing, malicious, etc,. can be used to train the model to filter things such as public comments in various social sites. This application can be converted to online type of machine learning system and can be easily updated with latest trends of spam and other mails so that the system can adapt to new types of spam emails and texts.

## 5.2 Conclusion:

From the results obtained we can conclude that an ensemble machine learning model is more effective in detection and classification of spam than any individual algorithms. We can also conclude that TF-IDF (term frequency inverse document frequency) language model is more effective than Bag of words model in classification of spam when combined with several algorithms. And finally we can say that spam detection can get better if machine learning algorithms are combined and tuned to needs.

# REFERENCES

[1]. Ming-Hsuan Yang, David J. Kriegman, Narendra Ahuja, “Detecting Faces in Images: A Survey”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume. 24, No. 1, 2002.

[2] S. H. a. M. A. T. Toma, "An Analysis of Supervised Machine Learning Algorithms for Spam Email Detection," in International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), 2021.

[3] S. Nandhini and J. Marseline K.S., "Performance Evaluation of Machine Learning Algorithms for Email Spam Detection," in International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020.

[4] A. L. a. S. S. S. Gadde, "SMS Spam Detection using Machine Learning and Deep Learning Techniques," in 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021, 2021.

[5] V. B. a. B. K. P. Sethi, "SMS spam detection and comparison of various machine learning algorithms," in International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), 2017.