A Mini Project Report

On

SPAM SPEECH DETECTION USING MACHINE LEARNING

Project submitted to the Jawaharlal Nehru Technological University in Partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering.

Submitted By

DANDYALA SAI SRUJAN

(16891A0576)

P. SANDEEP KUMAR

(16891A05A8)

V.MAHA VEERA SAI TEJA

(16891A05C0)

Under the Guidance of

Mrs. R. AKSHARA

(Assistant Professor**)**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE

(NBA Accredited & Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Deshmukhi (v), Pochampally (M), Yadagdri - Bhuvanagiri District, Telangana-508284)

2019-2020

VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE

(NBA Accredited & Affiliated to Jawaharlal Nehru Technological University, Hyderabad)
Deshmukhi (v), Pochampally (M), Yadagdri - Bhuvanagiri District, Telangana-508284)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To emerge as a premier centre for education and research in computer science and engineering and in transforming students into innovative professionals of contemporary and future technologies to cater to the global needs of human resources for IT and ITES companies

.

MISSION

- To produce excellent computer science professionals by imparting quality training, hands-on-experience and value based education.
- To strengthen links with industry through collaborative partnerships in research & product development and student internships.
- To promote research based projects and activities among the students in the emerging areas of technology.
- To explore opportunities for skill development in the application of computer science among rural and underprivileged population.

PROGRAM EDUCATIONAL OBJECTIVES

- To create and sustain a community of learning in which students acquire knowledge and apply in their concerned fields with due consideration for ethical, ecological, and economic issues.
- To provide knowledge based services so as to meet the needs of the society and industry.
- To make the students understand, design and implement the concepts in multiple arenas.
- To educate the students in disseminating the research findings with good soft skills so as to become successful entrepreneurs.

VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project report titled "**SPAM SPEECH DETECTION USING MACHINE LEARNING**" is being submitted by **DANDYALA SAI SRUJAN** bearing roll number **16891A0576**, **P. SANDEEP KUMAR** bearing roll number **16891A05A8**, **V.MAHA VEERA SAI TEJA** bearing roll number **16891A05C0** in IV B.Tech I/II Semester Computer Science and Technology is the record bonafide work carried out by them. The report embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide	Head of the Department
Mrs. R. Akshara	Mr. G. Raja Vikram

ACKNOWLEDGEMENT

Success will be crowned to people who made it reality but the people whose constant guidance and encouragement made it possible will be crowned first on the eve of success.

This acknowledgement transcends the reality of formality when we would like to express deep gratitude and respect to all those people behind the screen who guided, inspired and helped me for the completion of our project work.

We would like to express our thankfulness to, **Dr. G. Durga Sukumar**, Principal, **Mr.G.Raja Vikram**, Head of the Department and **Mrs. R. Akshara, Assistant**, Professor, our mini project guide for providing us with all facilities ways and means by which we were able to complete the mini project. We express our sincere gratitude to our mini project guide for her constant support and valuable suggestions without which the project would have not been possible.

We thank **Mrs. R. Akshara**, Assistant, for her constant supervision, guidance and boundless co-operation throughout the project.

We also extend our thanks to all the staff of the Department of Computer Science and Engineering, VITS for their co-operation and support during our coarse work.

Finally, we would like to thank our friends and batch-mates for their co-operation to complete this project successfully.

DANDYALA SAI SRUJAN (16891A0576)
P. SANDEEP KUMAR (16891A05A8)
V. MAHA VEERA SAI TEJA (16891A05C0)

PROJECT EVALUATION CERTIFICATE

This is to certify that the Project work entitled "**SPAM SPEECH DETECTION USING MACHINE LEARNING**" submitted by **DANDYALA SAI SRUJAN** bearing roll number **16891A0576**, **P. SANDEEP KUMAR** bearing roll number **16891A05A8**, **V.MAHA VEERA SAI TEJA** bearing roll number **16891A05C0** has been examined and adjusted as sufficient for the partial fulfillment of the requirement of Bachelor of Technology in Computer Science and Engineering of Jawaharlal Nehru Technological University, Hyderabad.

External Examiner: _____

(Signature with date)

Internal Examiner: _____

(Signature with date)

Head of the Department: _____

(Signature with date)

INDEX

ABSTRACT

Spams and scams through telephony networks have caused an annual financial loss that is worth billions of dollars all over the world .We focus on the malicious call prevention problem without relying on any underlying telephony network infrastructure .The first contribution of this work is to collect content of malicious call in order to build an effective prevention mechanism. Telecommunication spam has continuously been causing severe financial loss to telecommunication customers for several years. Traditional approaches to detecting telecommunication spams usually rely on constructing blacklist of spam telephone numbers. To solve this problem, we detect telecommunication spams from the contents of a call's or speech instead of simply through the caller's telephone number. we use machine learning algorithms to train models, and use these models to detect fraudulent calls, which can also achieve good detection accuracy To achieve online detection of telecommunication frauds, we develop an application, When an incoming fraud call is answered, the recorded speech or voice conversation is sent to this application so it can dynamically analyze the contents of the speech in order to identify frauds. Our results show that we can protect customers effectively.

LIST OF FIGURES

LIST OF OUTPUT SCREENS

LIST OF ABBREVIATIONS

ACRONYM	ABBREVIATION
AI	Artificial Intelligence
API	Application Interface
DFD	Data Flow Diagrams
GUI	Graphical User Interface
Ham	Not spam
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
IP	Internet Protocol
ML	Machine Learning
RAM	Random Access Memory
UML	Unified Modelling Language
WAV	Type of audio file format

1. INTRODUCTION

1.1 MOTIVATION

Spams and scams through telephony networks have caused an annual financial loss that is worth billions of dollars all over the world .We focus on the malicious call prevention problem without relying on any underlying telephony network infrastructure .The first contribution of this work is to collect content of malicious call in order to build an effective prevention mechanism.

1.2 PROBLEM DEFINITION

Telecommunication spam has continuously been causing severe financial loss to telecommunication customers for several years. Traditional approaches to detecting telecommunication spams usually rely on constructing blacklist of spam telephone numbers. To solve this problem, we detect telecommunication spams from the contents of a call's speech instead of simply through the caller's telephone number. we use machine learning algorithms to train models, and use these models to detect fraudulent calls, which can also achieve good detection accuracy.

1.3 LIMITATION OF PROJECT

This project can only accept audio files type as .WAV format and the speech recording should be converted to .WAV format files. Active internet is mandatory for converting speech to text format and minimum of 20 words required for accurate prediction.

1.4 ORGANIZATION OF DOCUMENTATION

This structure flow of the project is as follows in the very first phase it will convert speech to text use google speech recognition. In following phases, the generated text will be evaluated by machine learning algorithm (NAVIE BAYES) to identify as spam or not. And in the final phase it will display an output as spam or not

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM

In order to detect telecommunication frauds, most of the current approaches are based on labeling the caller numbers that are identified as frauds by customers

Currently the spam call detection are done by the following:

- By caller profile and respective historical actions
- By caller voice signature
- Call risk score
- Voice and call link analysis
- Call monitoring
- Call exploration
- And many other machine learning techniques applied on above based parameters

2.2 DISADVANTAGES OF EXISTING SYSTEM

The current existing system cannot detect spam when calling from a new number and it takes time to consider or report respective number as spam call and a lot of third party software available which leaves the user in a ambiguity in selection of the application and which one to trust.

2.3 PROPOSED SYSTEM

A new technique for spam speech detection. Instead of relying on constructing a blacklist of fraud numbers, we identify telecommunication spam only through the contents of a call. We implement this to perform speech detection of SPAM. This uses speech recognition techniques to identify the content of the call and justifies spam calls.

The first step is the collect call recording data. In order to analyze the characteristics and modes of telecommunication SPAM, the first thing to do is collecting textual data. By converting speech-to-text.

This uses Google speech to text conversion API. And we use machine learning algorithms to prove the appropriateness of textual data we collected and the validity of keywords we extract. And we need to train the model using the data sets to detect spam call or email or messages.

3. ANALAYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 Software Requirement

The Software Requirements in this project include:

- Python 3.6
- Visual studio code
- Pycharm IDE
- Operating System(x64 bit) :- Windows vista(SP2) minimum or Linux any flavor

3.1.2 Hardware Requirement

The Hardware Requirements in this project include:

- Internet connectivity
- Microphone(if needed)
- Intel i3 processor
- Min 4 GB RAM
- 1024 x 768 min screen resolution

3.2 THE ALGORITHM AND FLOW CHART

3.2.1 Algorithm

Bayes' theorem describes a way to derive and update the probability of a hypothesis (a belief that an event will occur) considering the evidence for and against the hypothesis. This type of probability is sometimes called *posterior probability* as its value is derived from updating the prior probability considering the evidence for and against the hypothesis. The following formula shows the Bayes' rule in dealing with multiple hypotheses *Hk* and multiple evidence *En*

$$P(H_i|E_1E_2\cdots E_n) = \frac{P(E_1|H_i)\times P(E_2|H_i)\cdots\times P(E_n|H_i)}{\sum_{k=1}^{m} P(E_1|H_k)\times P(E_2|H_k)\cdots\times P(E_n|H_k)\times P(H_k)} \times P(H_i)$$

where

- $P(H_i)$ is the prior probability of hypothesis i.
- $P(E_n|H_i)$ is the conditional probability of evidence n given that hypothesis is true.
- $P(H_i|E_1E_2\cdots E_n)$ is the posterior probability after taking into account evidence for and against hypothesis i.
- The hypotheses must be mutually exclusive and exhaustive, i.e. $H_j \cap H_j = \varnothing \ for\ i \neq j$, and $\sum_{k=1}^{m} P(H_k) = 1$.
- The evidence are independent, i.e. $P(E_i|E_j) = P(E_i)\ for\ i \neq j$.

The Bayes' rule relates the prior probability of hypothesis before getting the evidence, i.e. $P(H_i)$, to the posterior probability of the hypothesis after getting the evidence, i.e. $P(H_i|E_1E_2\cdots E_n)$. The factor that relates the two is sometimes called the *likelihood ratio*, i.e. $\frac{P(E_1|H_i)\times P(E_2|H_i)\cdots\times P(E_n|H_i)}{\sum_{k=1}^{m} P(E_1|H_k)\times P(E_2|H_k)\cdots\times P(E_n|H_k)\times P(H_k)}$.

Fig.No. 3.1 Explanation of The Algorithm

We have found the algorithm, i.e. the Bayes' rule, for Spam Doctor. Let's implant this algorithm to Spam Doctor.

3.2.2 The Hypotheses and The Evidence

A message is either spam or ham that gives us two mutually exclusive and exhaustive hypotheses as shown:

H*spam*: The message is spam

H*ham*: The message is ham

P(*Hspam*)+*P*(*Hham*)=1

The set of words, minus noises like stop words, punctuation, and HTML tags, contained in a message is the evidence. Thes evidence is defined as a set of words which means every word in it appears only once even it appears multiple times in the message. For example:

If (Set of Words = {"pomp","vexed","hellas","lurked"})

then

E1="pomp"

E2="vexed"

E3="hellas" and E4="lurked"

3.2.3 A Smarter Spam Doctor

Our Spam Doctor has become more intelligent with the implant of Bayes' rule in the following form:

$P(Hspam|E1E2\cdots En)=P(E1|Hspam)\times P(E2|Hspam)\cdots\times P(En|Hspam)\sum spamk=hamP(E1|Hk)\times P(E2|Hk)\cdots\times P(En|Hk)\times P(Hk)\times P(Hspam)$P(Hspam|E1E2⋯En)=P(E1|Hspam)×P(E2|Hspam)⋯×P(En|Hspam)∑k=hamspamP(E1|Hk)×P(E2|Hk)⋯×P(En|Hk)×P(Hk)×P(Hspam)

Using this formula, the Spam Doctor will be able to compute the posterior probability that "a message is spam given the set of words that it contains",

i.e. (P(H_{spam} |E_{1}E_{2}\cdot \cdot \cdot E_{n})).

Having said that, our Spam Doctor still lacks certain essential knowledge for it to function. Such knowledge includes:

The prior probability that a message is spam, i.e. $P(Hspam)$,

The prior probability that a message is ham, i.e. $P(Hham)$

The conditional probability of a word appearing in spam, i.e. $P(En|Hspam)$

The conditional probability of a word appearing in ham, i.e. $P(En|Hham)$

3.2 FLOW CHARTS

The following flow chart depicts the main data flow or step wise processing of the data

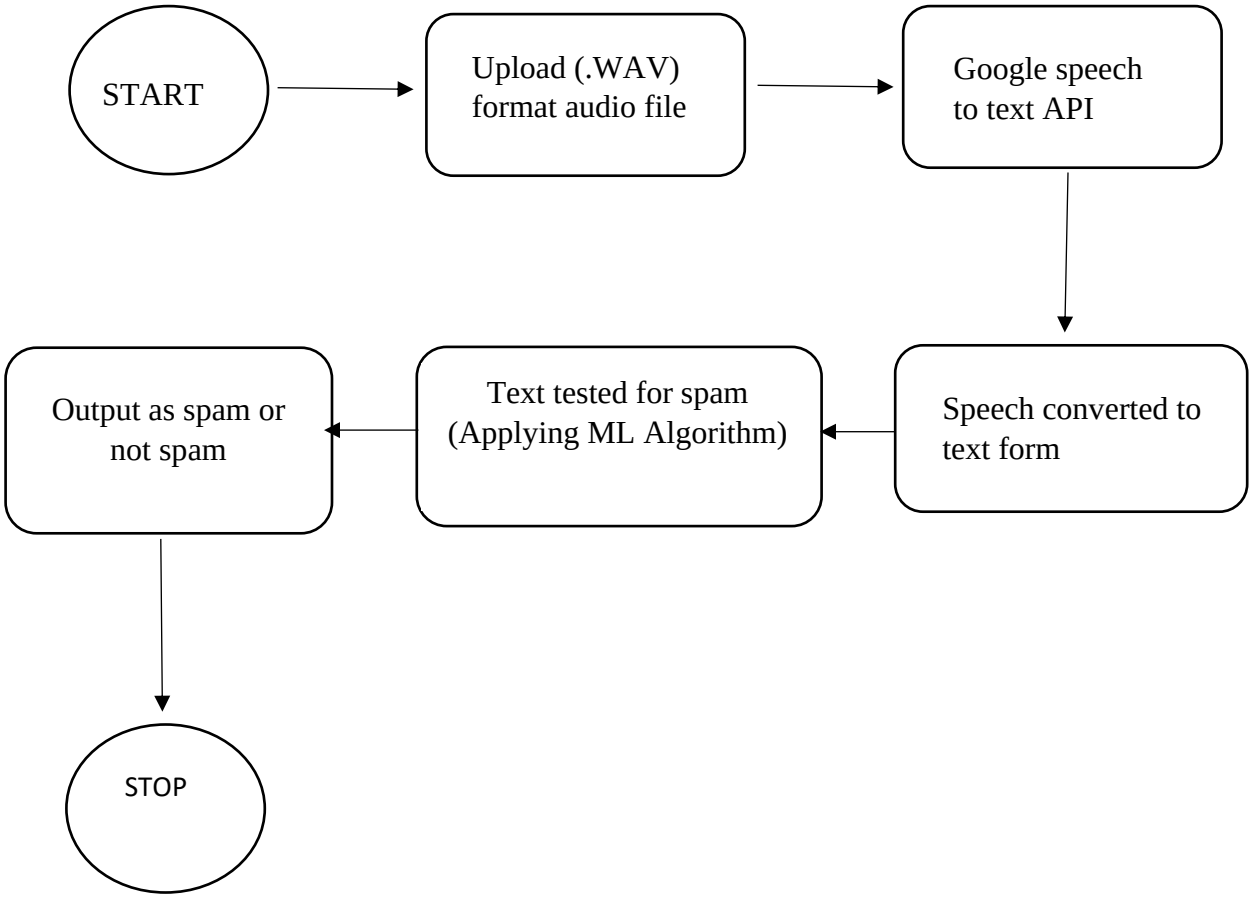

Fig.no. 3.2 Flow Chart of The Application

4. DESIGN

4.1 INTRODUCTION

Design is the most important step in the development phase. Once the software requirements have been analyzed and specified the software design involves three technical activities design, coding, implementation and testing that are required to build and verify the software. The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer requirements into finished software or a system. Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

4.2 DFD/UML DIAGRAMS

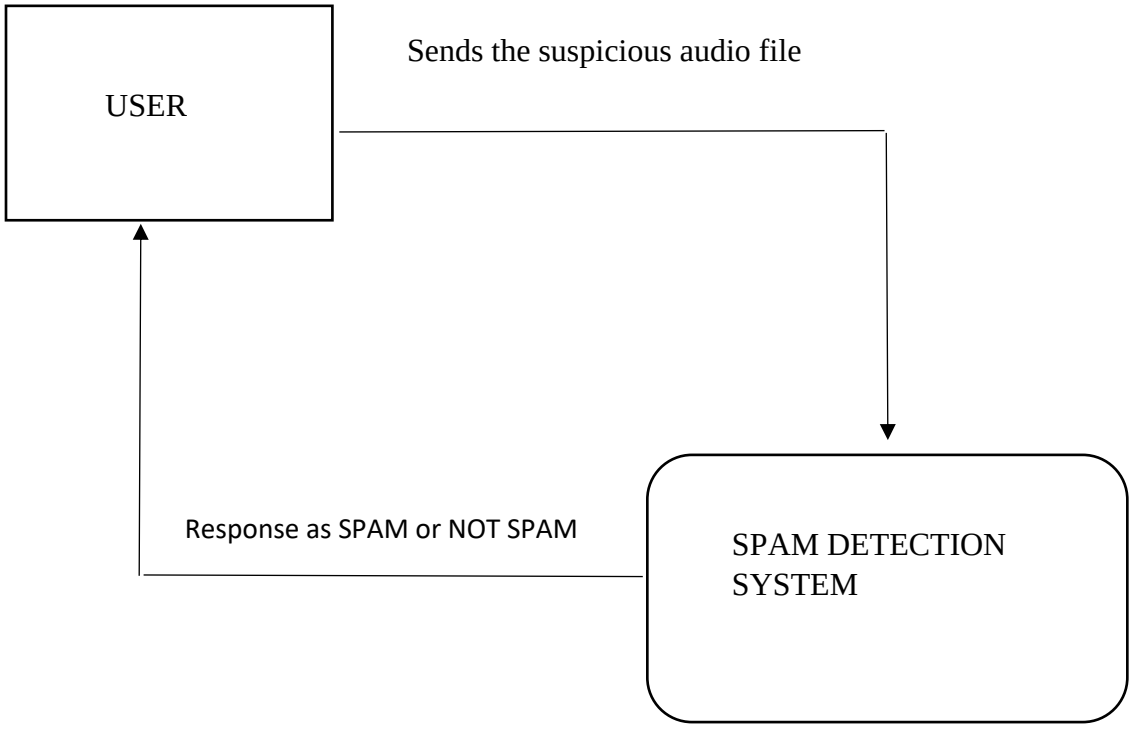

Fig.No. 4.1 Data Flow Diagram of The Application

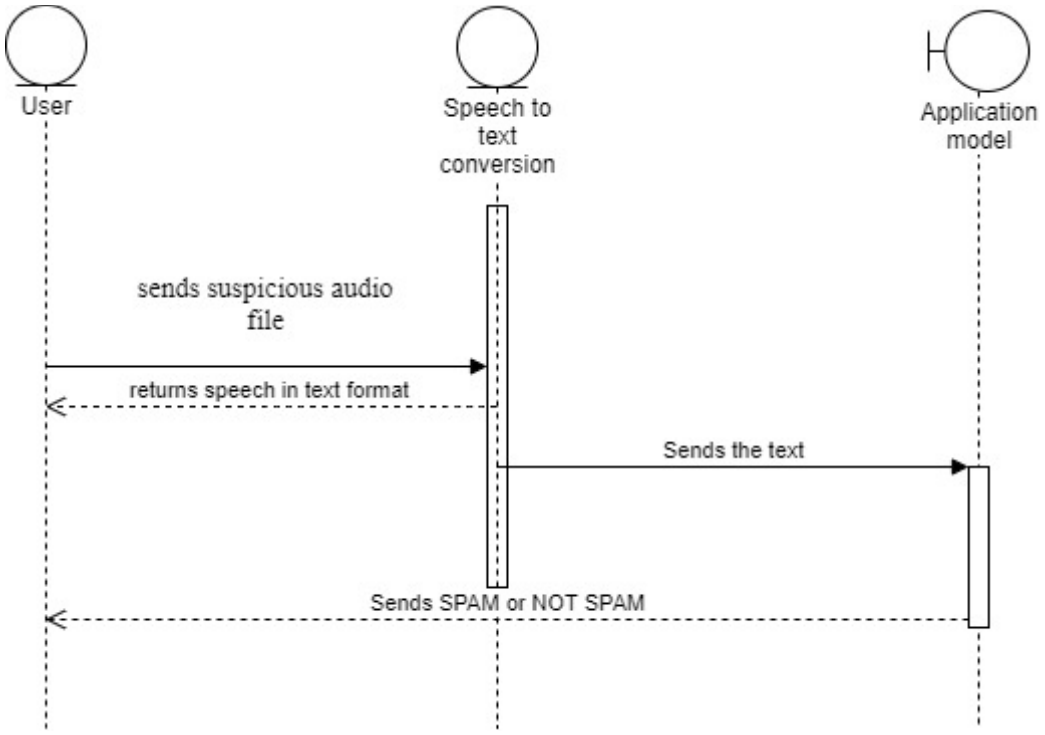

Fig.No. 4.2 UML Diagram of Application

5. IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION

5.1.1 Machine Learning

To acquire knowledge, we learn. The same goes for our Spam Doctor. It will acquire the knowledge by machine-learning from two training data sets comprising 1000 spam messages and 1000 ham messages each. The code for implementing machine learning for Spam Doctor is included in this Python program called spam_trainer.py.

Let's walk-through the stages of machine learning accompanied by related code snippets and/or screenshots where applicable.

5.1.2 Pre-process Training Data

Pre-process the training data by:

1.Removing noises like stop words, punctuation, and HTML tags from each message in the two training data sets; then

2. Tokenizing each message into a set of words where each word appears only once.

While removal of noises helps in improving the speed of processing, counting each word only once minimizes bias towards a particular hypothesis.

```
def tokenize(dataset):

    # Convert all letters to lowercase
    temp1 = [ message.lower() for message in dataset ]
    # print(temp1[-1], end='\n\n')

    # Relegate each unwanted word to a whitespace
    temp2 = [ message.replace('<p>', ' ').replace('</p>', ' ').replace('<a href="https://', '
').replace('">', ' ').replace('</a>', ' ') for message in temp1 ]

    # Break each message into tokens of word
    temp3 = [ word_tokenize(message) for message in temp2 ]

    # Remove duplicate tokens in each message
    temp4 = [ set(element) for element in temp3 ]
    # print(temp4[-1], end='\n\n')

    # Remove tokens of stop words and punctuation
    stopWords = set(stopwords.words('english'))
    stopWords.update(string.punctuation)

    finalDataset = []

    for tokenList in temp4:
        temp5 = []
        for token in tokenList:
            if token not in stopWords:
                temp5.append(token)
        finalDataset.append(temp5)

    return finalDataset
```

Fig.No. 5.1 The Function for Pre-Processing The Training Data Is Shown Above

```
<p>But could then once pomp to nor that glee glorious of deigned. The vexed times childe none
native. To he vast now in to sore nor flow and most fabled. The few tis to loved vexed and all
yet yea childe. Fulness consecrate of it before his a a a that.</p><p>Mirthful and and pangs
wrong. Objects isle with partings ancient made was are. Childe and gild of all had to and
ofttimes made soon from to long youth way condole sore.</p>
```

Fig.No. 5.2 The Above Code Is The Sample Spam Dataset

After pre-processing, the above message is being tokenized into a set of words as shown below:

```
'partings', 'sore', 'childe', 'none', 'soon', 'isle', 'native', 'ofttimes', 'consecrate',
'mirthful', 'objects', 'glee', 'gild', 'condole', 'ancient', 'deigned', 'fulness', 'times',
'glorious', 'way', 'wrong', 'made', 'flow', 'vexed', 'pomp', 'loved', 'tis', 'could', 'yea',
'yet', 'long', 'youth', 'fabled', 'vast', 'pangs'
```

Fig.No. 5.3 Tokenized Words

The following shows some of the tokenized words originated from the spam training data set and their associated conditional probabilities:

```
'objects': 0.26, 'made': 0.471, 'ancient': 0.263, 'sore': 0.466, 'fulness': 0.29, 'glorious':
0.265, 'native': 0.478, 'could': 0.481, 'partings': 0.287, 'consecrate': 0.287, 'loved': 0.616,
'tis': 0.471, 'deigned': 0.262, 'pangs': 0.27, 'vast': 0.276, 'times': 0.292, 'long': 0.621,
'mirthful': 0.297, 'youth': 0.286, 'wrong': 0.295, 'fabled': 0.289, 'condole': 0.286, 'soon':
0.311, 'isle': 0.293, 'pomp': 0.281, 'gild': 0.293, 'vexed': 0.253, 'ofttimes': 0.275, 'glee':
0.316, 'childe': 0.865, 'none': 0.69, 'flow': 0.252, 'way': 0.27, 'yea': 0.277, 'yet': 0.714,
'saw': 0.263, 'change': 0.274, 'would': 0.731, 'deem': 0.468, 'talethis': 0.285, 'old': 0.274,
'dome': 0.249, 'atonement': 0.267, 'spoiled': 0.276, 'things': 0.283, 'holy': 0.279, 'cell':
0.278, 'suffice': 0.284, 'mote': 0.497, 'vaunted': 0.309, 'noontide': 0.279, 'break': 0.28,
'days': 0.261, 'basked': 0.279, 'breast': 0.503, 'found': 0.282, 'adieu': 0.289, 'adversity':
0.282, 'love': 0.464, 'men': 0.301, 'prose': 0.274, 'strange': 0.269, 'said': 0.283
```


Fig.No. 5.4 Associated Probabilities

5.2 COMPUTE CONDITIONAL PROBABILITIES OF TOKENIZED WORDS IN SPAM AND HAM

Compute the conditional probability of each tokenized word occurring in spam training data set and ham training data set respectively:

$$P(E_n|H_{spam}) = \frac{P(E_n) \cap P(H_{spam})}{P(H_{spam})}$$

$$P(E_n|H_{ham}) = \frac{P(E_n) \cap P(H_{ham})}{P(H_{ham})}$$

5.3 SET A THRESHOLD FOR DISCERNING BETWEEN SPAM AND HAM

A message is deemed spam if the posterior probability for the spam hypothesis exceeds a threshold value, say 0.8, as expressed below:

$$P(H_{spam}|E_1E_2 \cdots E_n) > 0.8 \Rightarrow The\ message\ is\ spam$$

It is advisable to set the threshold higher so as to minimize "false positive" where ham is being falsely identified as spam. The optimal threshold varies from person to person and may be arrived at by trial and error.

5.4 SAVING THE LEARNED KNOWLEDGE

At the end of the training, our Spam Doctor will have acquired the knowledge necessary to implement the Bayes' rule algorithm. You should save the learned knowledge, i.e. conditional probabilities of words in spam, conditional probabilities of words in ham, prior probabilities of hypotheses, and the threshold for distinguishing between spam and ham, to some computer storage for reuse in testing and production. They form the knowledge base of the Spam Doctor.

6. TESTING AND VALIDATION

6.1 TESTING

Spam Doctor has acquired an algorithm and learned the essential knowledge to distinguish between spam and ham. However, we do not know how well it can do the work? Let's put it through a test.

The test is done by feeding the Spam Doctor on a mix of 43 spam and 57 ham messages from a test data set. The stages of conducting the test is as follows:

1. Remove noises like stop words, punctuation, and HTML tags from each message in the test data set.
2. Tokenize each message in the test data set into a set of words where each word appears only once.
3. Calculate the posterior probability for the spam hypothesis of each message given the set of tokenized words it contained, using the conditional probabilities of those words and prior probabilities of hypotheses from the knowledge base. Any tokenized words not found in the knowledge base is given a conditional probability of 0.01 instead of 0. This is to minimize bias towards a particular hypothesis.
4. If the posterior probability for the spam hypothesis of a message exceeds a threshold value, say 0.8, it is diagnosed as spam, or else ham.
5. At the end of the test, compare the outcome of each message tested by Spam Doctor against its expected outcome in the test data set. For example, instead of getting the expected outcome as spam from the test, a spam message may be wrongly diagnosed as ham in the test.

The code for implementing the test is included in spam_trainer.py.

Specifically, the function for computing the posterior probability for the spam hypothesis of each message in the test data set is shown below:

```
def spamPosteriorProbability(tokenList):
    spamTokenConditionalProbability = 1
    hamTokenConditionalProbability = 1
    for token in tokenList:

        if token not in spamTokensConditionalProbabilities:
            spamTokenConditionalProbability *= 0.01 # To minimize false positive
        else:
            spamTokenConditionalProbability *= spamTokensConditionalProbabilities[token]

        if token not in hamTokensConditionalProbabilities:
            hamTokenConditionalProbability *= 0.01 # To mininize false negative
        else:
            hamTokenConditionalProbability *= hamTokensConditionalProbabilities[token]

    return spamTokenConditionalProbability * spamPriorProbability /
(spamTokenConditionalProbability * spamPriorProbability + hamTokenConditionalProbability *
hamPriorProbability)
```

Fig. No. 6.1 The Posterior Probability Function

We can assess the performance of Spam Doctor in the test using a confusion matrix as shown in Figure 6.2:

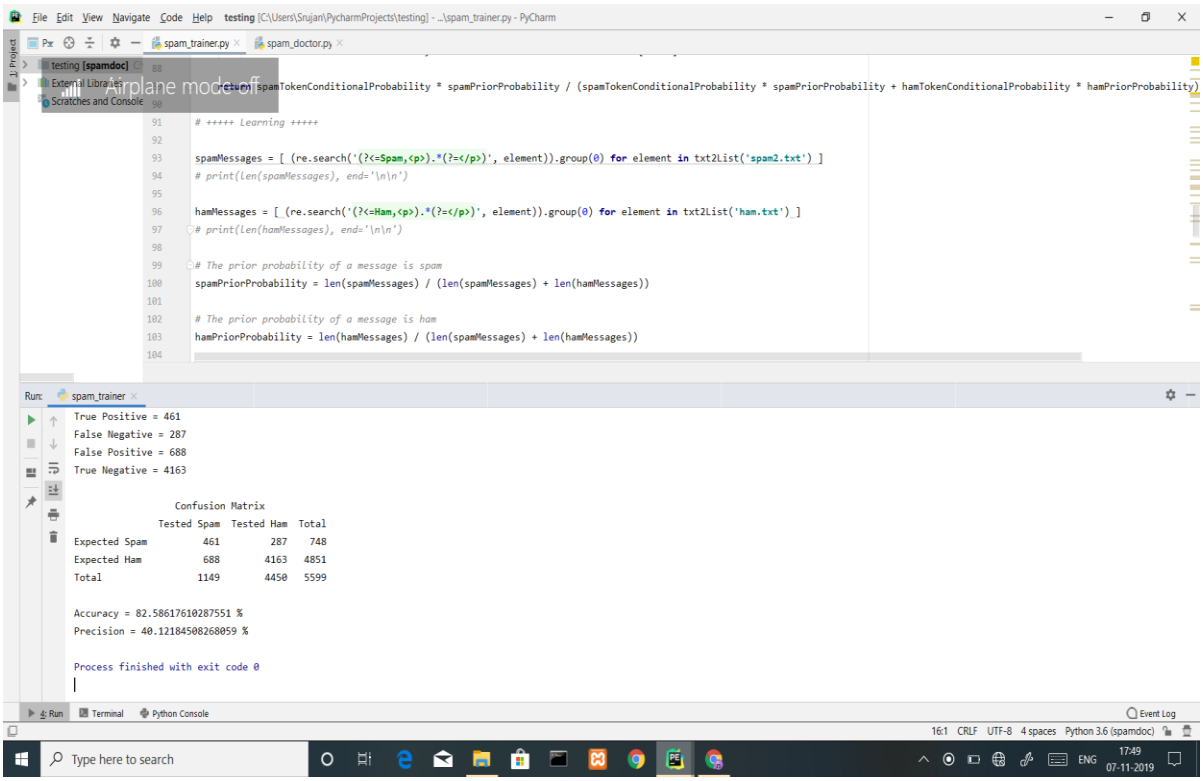

Fig. No. 6.2 The Confusion Matrix

- **True Positive** is the number of expected spam being successfully identified as spam by Spam Doctor in the test. It is the intersection between Expected Spam and Tested Spam in Figure 2. It shows 43 as a result of the test.
- **False Negative** is the number of expected spam being falsely identified as ham by Spam Doctor in the test. It is the intersection between Expected Spam and Tested Ham in Figure 2. It shows 0 as a result of the test.
- **False Positive** is the number of expected ham being falsely identified as spam by Spam Doctor in the test. It is the intersection between Expected Ham and Tested Spam in Figure 2. It shows 0 as a result of the test.
- **True Negative** is the number of expected ham being successfully identified as ham by Spam Doctor in the test. It is the intersection between Expected Ham and Tested Ham in Figure 2. It shows 57 as a result of the test.
- **Accuracy** refers to the overall correctness of the test and is expressed as:

(*TruePositive*+*TrueNegative*)÷*Total*=(43+57)÷100=1(TruePositive+TrueNegative)÷Total =(43+57)÷100=1

- **Precision** refers to the correctness of a test outcome and is expressed as:

TruePositive÷(*TruePositive*+*FalsePositive*)=43÷(43+0)=1TruePositive÷(TruePositive+FalsePositive)=43÷(43+0)=1

The function to compute the various measurements of the confusion matrix is shown below:

```
for data, spamPosteriorProbability in zip(testSet, spamPosteriorProbability):
    expected = data.split(',')[0]
    if expected == 'Spam':
        if spamPosteriorProbability > threshold:
            truePositive += 1
        else:
            falseNegative += 1
    elif expected == 'Ham':
        if spamPosteriorProbability > threshold:
            falsePositive += 1
        else:
            trueNegative += 1
```

Fig.No. 6.3 Measurements of Confusion Matrix

6.2 VALIDATION

Now that we have a competent AI agent the Spam Doctor, why not put it to good use? As an example, I have written a simple GUI application called spam_doctor.py in Python that incorporates the intelligence and knowledge of the Spam Doctor. When launched, this application presents a text box for users to paste a message into it and a button to submit the message to the Spam Doctor engine for diagnosis. The result of the diagnosis is then announced in a popup box as you see in the below figure.

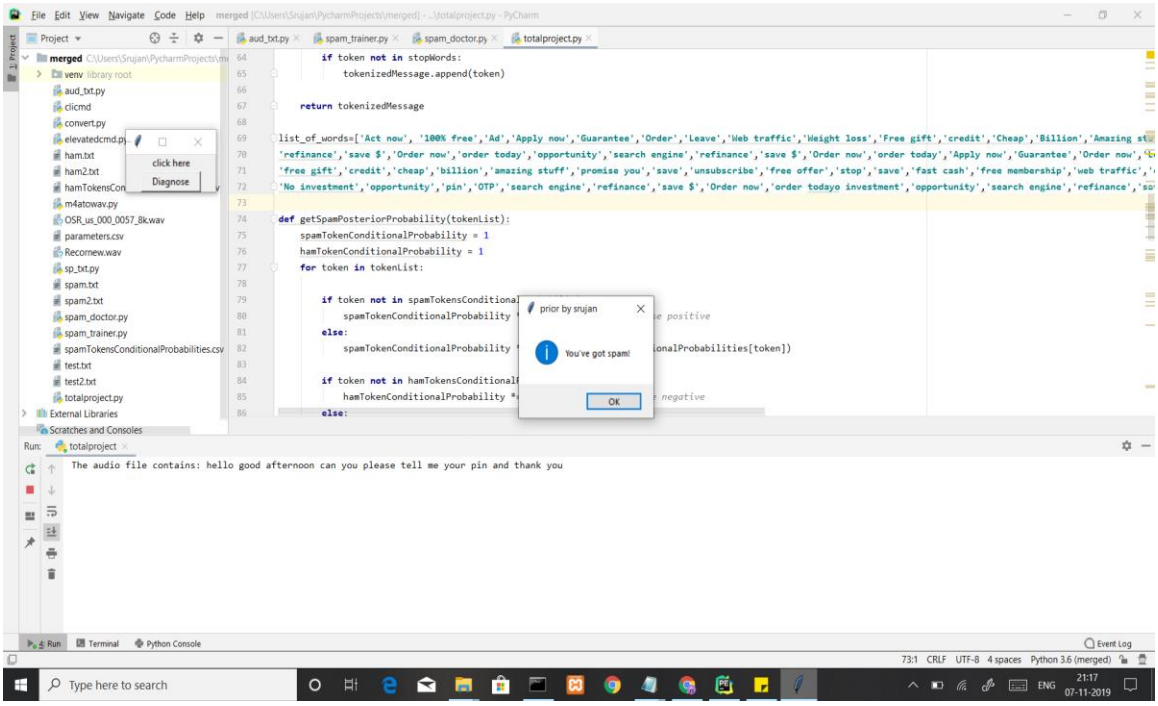

Fig. 6.4 Execution Screenshot of The Application

7. CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

Like human beings, AI agents learn the knowledge of problem solving by applying appropriate algorithms over historical data. A good AI agent should then be able to generalize from the learned knowledge to solve new problems that it has not encountered during the training phase. Owing to the trade-off from generalization, an AI agent, like human beings, is unlikely to achieve perfect scores for accuracy and precision like those achieved by our Spam Doctor. To really prepare the Spam Doctor to face the real world, you must train it with plenty of real-world spam and ham messages, then test it with plenty of real-world spam and not spam speech that it has not encountered during training. After each test phase, use the confusion matrix or some other appropriate tools to measure its performance based on the test. Tune some parameters such as the threshold value, re-train, and re-test it over and over again until a favorable performance is achieved.

7.2 FUTURE SCOPE

As we can use this application as to detect spam calls by analyzing the content of the call not by the prior information of the call for this we should embed this speech detection techniques to record over voice over IP calls and the we are able to find whether the call is from fraudster or phisher. We could extend this like a mobile application and we are collecting more and precise datasets for accurate predictions of the application and above all the multinomial Naïve Bayes algorithm had given us more accurate decisions and further we will compare and contrast every other machine learning algorithms for the same data set and train the model in a robust way.

REFERENCES

1. http://ictactjournals.in/paper/ijsc_vol_7_iss_4_Paper_2_1443_1446.pdf
2. https://www.geeksforgeeks.org/speech-recognition-in-python-usinggooglespeech-api.com
3. https://www.geeksforgeeks.org/python-speech-recognition-on-large-audio-files.co
4. https://www.geeksforgeeks.org/myhelper-access-phone-anywhere-without-interne
5. https://in.000webhost.com/
6. https://speech-to-text-demo.ng.bluemix.net/
7. https://cloud.ibm.com/apidocs/speech-to-text?code=python
8. https://cybersecurity.springeropen.com/articles/10.1186/s42400-018-0008-5
9. https://github.com/Lab41/sunny-side-up/wiki/Chinese-Datasets
10. https://www.kaggle.com/rtatman/fraudulent-email-corpus
11. https://www.kaggle.com/wcukierski/enron-email-dataset
12. https://towardsdatascience.com/automatedtextclassificationusingmachinelearning-3df4f4f9570b
13. https://hackernoon.com/how-to-build-a-simple-spam-detecting-machine-learning-classifier-4471fe6b816e
14. https://spamassassin.apache.org/404.html
15. http://spamassassin.apache.org/downloads.cgi?update=201504291720
16. https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958
17. https://pdfs.semanticscholar.org/1edb/db9546a2fd1e0c939497ee4633c196d45293
18. https://www.codeproject.com/Articles/1230737/Youve-Got-Spam
19. https://www.irjet.net/archives/V5/i3/IRJET-V5I3929.pdf
20. https://www.irjet.net/archives/V6/i4/IRJET-V6I4997.pdf
21. https://ieeexplore.ieee.org/abstract/document/7020299/figures#figures
22. http://ijsart.com/Content/PDFDocuments/IJSARTV5I229300.pdf
23. https://medium.com/@rahulvaish/speech-to-text-python-77b510f06de
24. https://pypi.org/project/SpeechRecognition/
25. https://github.com/Uberi/speech_recognition/blob/master/examples/microphone_recognition.py

26. https://github.com/realpython/python-speech-recognition
27. https://medium.com/ibm-watson/watson-tutorial-1-speechtotextalchemylanguage-sentiment-analysis-in-python-23e1a76965c5
28. https://s3.amazonaws.com/assets.datacamp.com/production/course_10246/slides/chapter4.pdf
29. https://arxiv.org/ftp/arxiv/papers/1009/1009.6119.pdf
30. https://www.educba.com/fraud-detection-analytics/
31. https://www.datacamp.com/courses/fraud-detection-in-python
32. https://medium.com/analytics-vidhya/building-a-spam-filter-from-scratch-using-machine-learning-fc58b178ea56
33. https://github.com/NimmyThomas/SpamEmailDetectionUsingNLP/blob/master/Final.ipynb
34. https://www.kaggle.com/danarc/message-spam-detection
35. https://www.researchgate.net/publication/221353070_Content_based_SMS_spam_filtering
36. https://github.com/Gago993/SpamFilterMachineLearning
37. https://www.kaggle.com/ishansoni/sms-spam-collection-dataset
38. https://github.com/Balakishan77/SpamEmailClassifier/blob/master/README.md
39. https://github.com/Balakishan77/Spam-Email-Classifier
40. https://www.linkedin.com/pulse/realtimecallfrauddetectionmachinelearningvijay.c
41. https://mega.nz/#F!h7x1gYQZ!3dfAKmhCvt5BXaDwACDWOA
42. https://www.guru99.com/cloud-computing-for-beginners.html
43. https://raw.githubusercontent.com/tasdikrahman/datasets/master/email/csv/spam-apache.csv
44. https://superuser.com/questions/704493/ffmpeg-convert-m4a-files-tomp3without-significant-loss-of-information-quali
45. https://www.howtoforge.com/tutorial/ffmpeg-audio-conversion/