TRIBHUVAN UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

A Major Project Report

On

Spam Detection using Naive Bayes Algorithm

Submitted By:

Puspa Thakur (2075CSIT33)

Sangam Khatri (2075CSIT36)

Santosh Chaudhary (2075CSIT38)

Submitted To:

Department of Computer Science and Information Technology

Chyasal, lalitpur

April, 2023

Acknowledgement

We take this occasion to thank our parents for their consistent support and encouragement. We are immensely thankful to our college, Himalaya College of Engineering, for including the project in the syllabus. We are also very grateful for the college for providing us with this opportunity. Furthermore, we extend our sincere and heartfelt thanks to our Head of Department, Er. Himal Chand Thapa, for providing us with the right guidance and for showing us the right way. We would like to express our deep gratitude towards our supervisor Er.Drish Mali as well as other faculty members for their proper guidance and inspiration. Finally, we would like to give special thanks to acknowledge all the people who have helped us towards the development of this project.

Group Members:
Puspa Thakur(2075CSIT33)
Sangam Khatri (2075CSIT36)
Santosh Chaudhary (2075CSIT38)

Abstract

Spam Detection is a task in natural language processing that aims to identify whether the given message is spam or not. The Naive Bayes algorithm is a popular and effective method for spam detection due to its simplicity and speed. Spam detection is big problem. One popular approach to this problem is using the Naive Bayes algorithm.The algorithm works by building a model from a training set of labeled messages, in which the occurrence of each word is used to calculate the likelihood of that word being present in a spam message.The data used was spam dataset which was cleaned,pre-processed ,vectorized using TF-IDF Vectorizer to train and build a model using Naive Bayes Algorithm .It is a probabilistic classifier, which means it predicts on the idea of the probability of an object and it is selected for the email spam detection having best precision and accuracy.

Keywords : Message ,Spam, Algorithm, Probability

Table of Contents

List of Figures

List of Tables

List of Abbreviation

IOT	Internet of Things
IDE	Integrated Development Environment
OS	Opersting System
DFD	Data Flow Diagram
TF-IDF	Term Frequency-Inverse Document Frequency
UCI	University of California Irvine
NLTK	Natural Language Processing Toolkit

Chapter 1:Introduction

1.1 Introduction

Spam is any kind of unwanted, unsolicited digital communication that gets sent out in bulk. Often spam is sent via email, but it can also be distributed via text messages, phone calls, or social media.Spam is annoying, but it's also a threat. While many of us might think we're smart enough to recognize any form of it, spammers regularly update the methods and messages to trick potential victims.

Nowadays, messages and texts are the most efficient way of communication in almost every field, from business to education. Messages have two subcategories, i.e., ham and spam. Message spam, also called junk messages or unwanted messages, is a type of message that can be used to harm any user by wasting people's time, computing resources, and stealing valuable information. The ratio of spam message is increasing rapidly day by day. Spam detection and filtration are significant and enormous problems for email and IOT service providers nowadays. Among all the techniques developed for detecting and preventing spam, detecting spam message is one of the most essential and prominent approaches. Several machine learning and deep learning techniques have been used for this purpose, i.e., Naive Bayes, decision trees, neural networks, and random forest.

This project is a spam detection system which detects whether the message/text is spam or ham..Naive Bayes Algorithm is used in this project. Naive Bayes is a probabilistic algorithm. This spam detection project is based on calculation of probability of the words that comes in a spam or ham classes.Prediction of the spam/ham is done on the basis of the probability of the word occurring in the classes..This project helps user detect the messages and help the users understand that the text/messages they are getting is a spam and ham.This project detects the messages that are spam and alerts the users to not go behind the spam messages that leads to fraud, stealing or corruption of personal data and valuables.

1.2 Problem Statement

Spammers expect only a small number of recipients to respond or interact with the spam messages, but spammers can still swindle a way to a big payday because the spammers can easily send shady messages to so many people in a single stroke. That is why spam continues to be a big problem in the modern digital economy. Spam messages lure people by sending them attractive and undeniable offers that leads to theft and corruption of one's important data and information.

1.3 Objective

The main objective of this project is:

- To implement Naive Bayes Algorithm for spam/ham detection.

1.4 Limitation

The limitation of this project is that it has only one feature that is to predict the message as spam or ham and sometimes the complex words that are not in the training dataset but are in the testing dataset may result to failure in prediction.The main limitation of Naive Bayes is the assumption of independent predictor features. Naive Bayes implicitly assumes that all the attributes are mutually independent. In real life, it's almost impossible that we get a set of predictors that are completely independent or one another.

1.5 Report Organization

Chapter-1 :Introduction

The first chapter serves as an introduction. This section discusses the system's introduction, problem statement, objectives, limitations and the organization of the report.

Chapter-2 :Literature Review

The second chapter carries on with project analysis. It includes a literature review and background part which summarizes the research done before doing the project.

Chapter-3 :Requirement Analysis and Feasibility study

The third chapter carries on with the system analysis. It includes requirement analysis, feasibility study.

Chapter-4 :System Design

The fourth chapter discusses the system design part and also includes the algorithm details part where the algorithm is explained in brief.

Chapter-5 :Methodology

The fifth chapter discusses about the implementing and testing of the project. It consists details of how the project was implemented and tested.

Chapter-6:Conclusion and Recommendation

The sixth chapter consist of conclusion and recommendation of the project.It sums up the project with a conclusion and also suggests future enhancement of the project.

The references for the projects are included at the last section of the report along with the appendix.

Chapter 2:Literature Review

Tuteja S.K. collected email datasets from the online available websites and used Naive Bayes for filtering of emails. This model proposed a hybrid approach using secure hash method and Naive Bayes to filter email data but could not provide information regarding the misuse of storage resources and network bandwidth. By using Secure Hash Algorithm, the email is considered as a message M due to a generated function. The message M is further classified into S and L where L stands for ham email or genuine email and on the other hand S stands for spam email.[1]

Performed analysis on email classification on two different datasets by using Naive Bayes algorithm based on the Accuracy, Precision, F-Measure and Recall. The process was divided into 3 steps. First step is data pre- processing in which all articles, conjunctions and undesired words is removed from the text. Next is the feature extraction followed by training of the Naive Bayes model, Based on the training of the model, it predicts whether the given text is ham or spam. By using Spam data Datasets, the author achieved an accuracy of 91.13% and for the other Spam Base datasets, accuracy achieved was 88%. By this analysis, the author concluded that the performance of Naive Bayes algorithm is better on spam data datasets compare to spam base.[2]

In this,S. Ajaz and etal. have compared the performance of Naive Bayes and Support Vector Machine algorithm for classification of emails. The datasets used consists of 5574 rows and 2 columns. One column is used for storing emails and other is used as label (Ham or Spam). Data Collection, Data Pre-processing, Data Transformation and Classification System for classification of emails are done. Data Pre-processing was used to clean the data and make it free from any kind of ambiguities, errors, redundancy. In Data Transformation, pre-processed data is converted into lowercase and converted into format as desired by the algorithm for classification. And at last desired 5 attributes are identified and by using feature extraction, algorithm classifies the content into Ham or Spam. Accuracy

obtained by using Naive Bayes was 99.49% and it was 86.35% by using Support Vector Machine. So, the author concluded that Naive Bayes algorithm performed exceptionally well as compared to SVM for classification of emails.[3]

Proposed a method for classifying an email as Ham or Spam by using feature selection and also worked to improve the training time as well as the accuracy of the spam filtering model. also performed a comparison between the Naive Bayes algorithm and Support Vector Machine. Based on the accuracy as well as the computation time of the algorithm for the given datasets. The whole process was divided into four steps. First was preparation of data in which the given datasets was divided into training set consisting of 702 mails and testing set with 260 mails. The second step was creation of word dictionary followed by the third step which was feature selection process by generating the feature vector matrix. The last step was to train the model and based on its training the model predicts the email as ham or spam. Based on the results obtained, the author concluded that Naive Bayes gives better accuracy in comparison with Support Vector Machine.[4]

Chapter 3: Requirement Analysis and Feasibility study

3.1 Requirement Analysis

The requirements are to be collected before starting projects' development life cycle. To design and develop system, functional as well as non-functional requirement of the system has been studied.

3.1.1 Functional Requirements

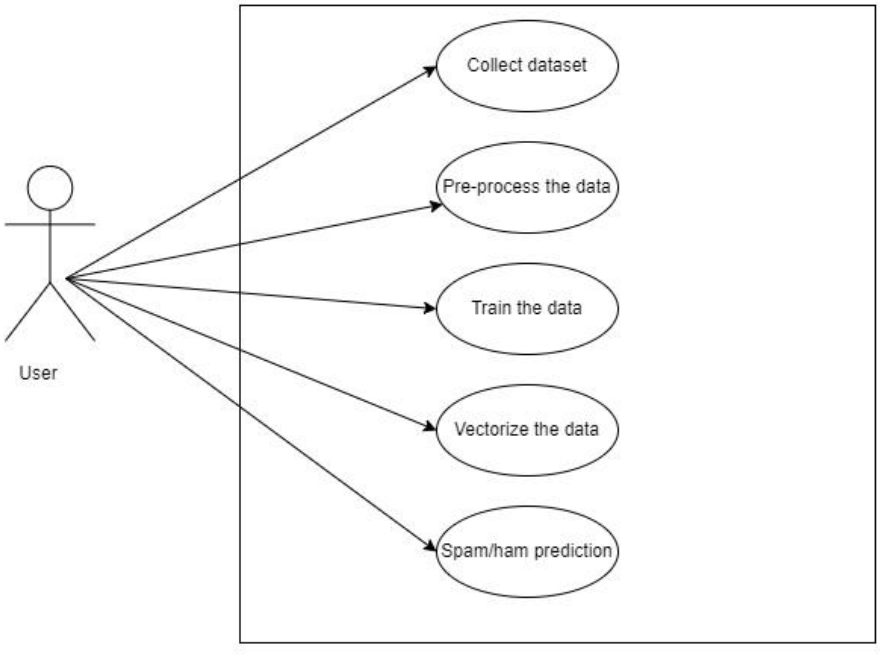


Fig 3.1 Use Case Diagram of Spam Detection

The figure 3.1 is the use case diagram of spam detection system .In this system there is an actor which is user.User collects the dataset and load it.The data that are loaded and pre-processed.After the data are pre-processed the textual data are converted to numbers i.e Vectorization process.This project uses TF-IDF Vectorizer to vectorize the data.Once the data are vectorized train/test split is done.The model is build using Naive Bayes Algorithm and the system build then predicts whether the input message is spam or not.

3.2 Feasibility Study

The feasibility study is necessary to determine if creating a new or improved system is friendly with the cost,benefits, operation, technology and time. Some feasibility studies done in our project are as follows:

Economic feasibility

In context to this project work, the system developed is a web application; which uses all the tools and technical mechanism with open source I.e free of cost.No huge initial investment is required for the implementation of the project. This project is economically feasible.

Operational feasibility

The project “spam detection “ is easy to understand and implement, with proper guide and understanding of the algorithm anybody can operate this project.This project is operationally feasible.

Technical feasibility

All the tools and software product required to construct this project are familiar to the developers.Manuals and tutorials are easily available over the internet .The tools and software product used for this project are affordable and easy to understand and operate.This project is technically feasible.

3.2.1 Schedule

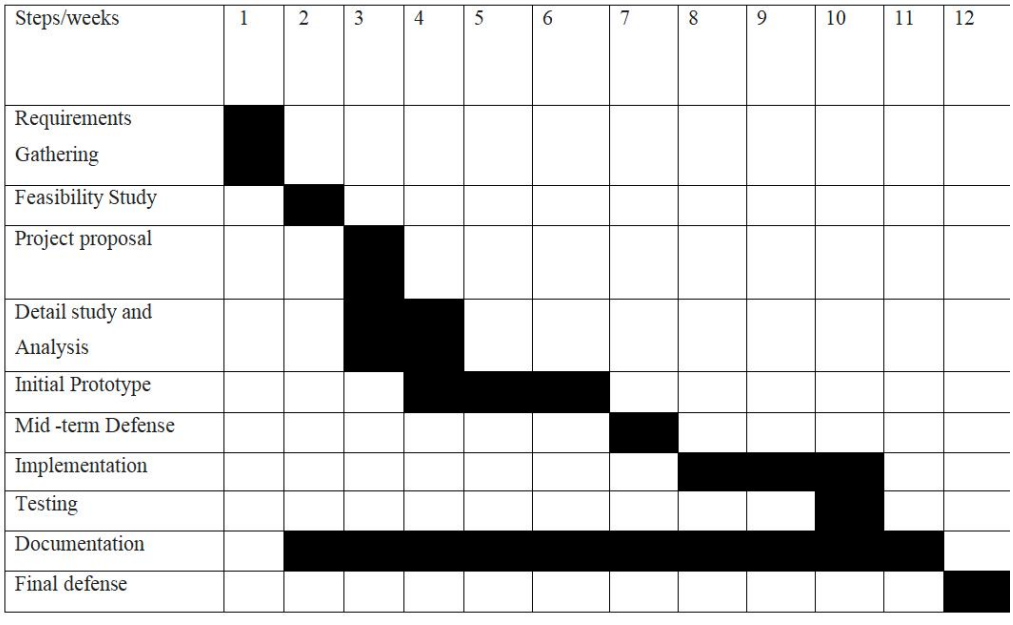


Fig 3.2 Gantt Chart of Spam Detection

3.3 Software and Hardware Requirement

3.3.1 Software Requirement

Following are the software requirement necessary for the project.

- Python Programming Language.
- Python IDE
- Windows/UNIX OS (Operating System)

3.3.2 Hardware Requirement

Following are the hardware requirement necessary for the project.

OS: Windows 8/10/11

Processor: Intel Core i5 or better*

Memory: 4gb or more.

Hard disk : 4gb or more

3.4 Structuring System Requirements

3.4.1 Process Modeling

Dataflow Diagram

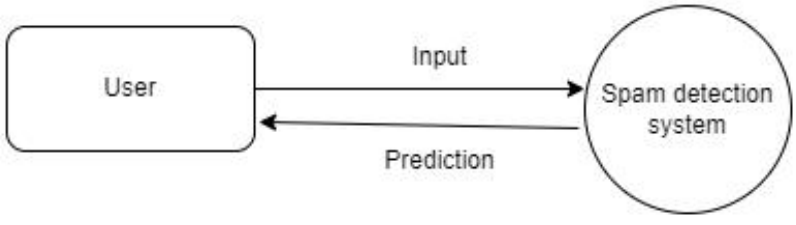


FIg 3.3 DFD level 0 of Spam Detection

Fig 3.3 is the DFD level 0 of Spam detection system .This gives a surface view of the data flow of the system.It explains that the user gives input message to the system and system gives the predicted result of the input message to the user.

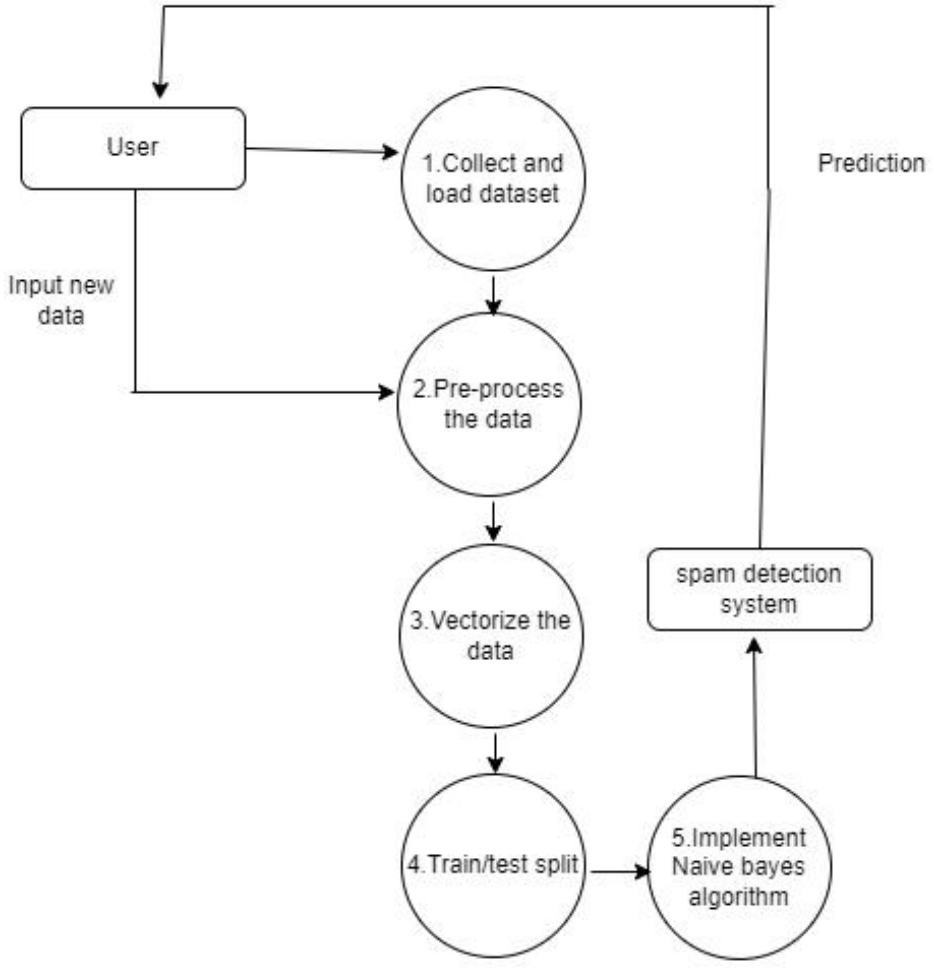


Fig 3.4 DFD level 1 of Spam Detection

Fig 3.4 is the DFD level 1 of Spam detection system .This gives a more detail view of the data flow of the system.It shows the processes involved between the user and the system .There are various processes involved in the system.The user collects the dataset,it is pre-processed,the data is vectorized then split into train/test data to train the model and then the model is trained with Naive Bayes Algorithm for spam detection.Once the model is trained with the classifying algorithm when the user inputs a message,the system gives the predicted message to the user.

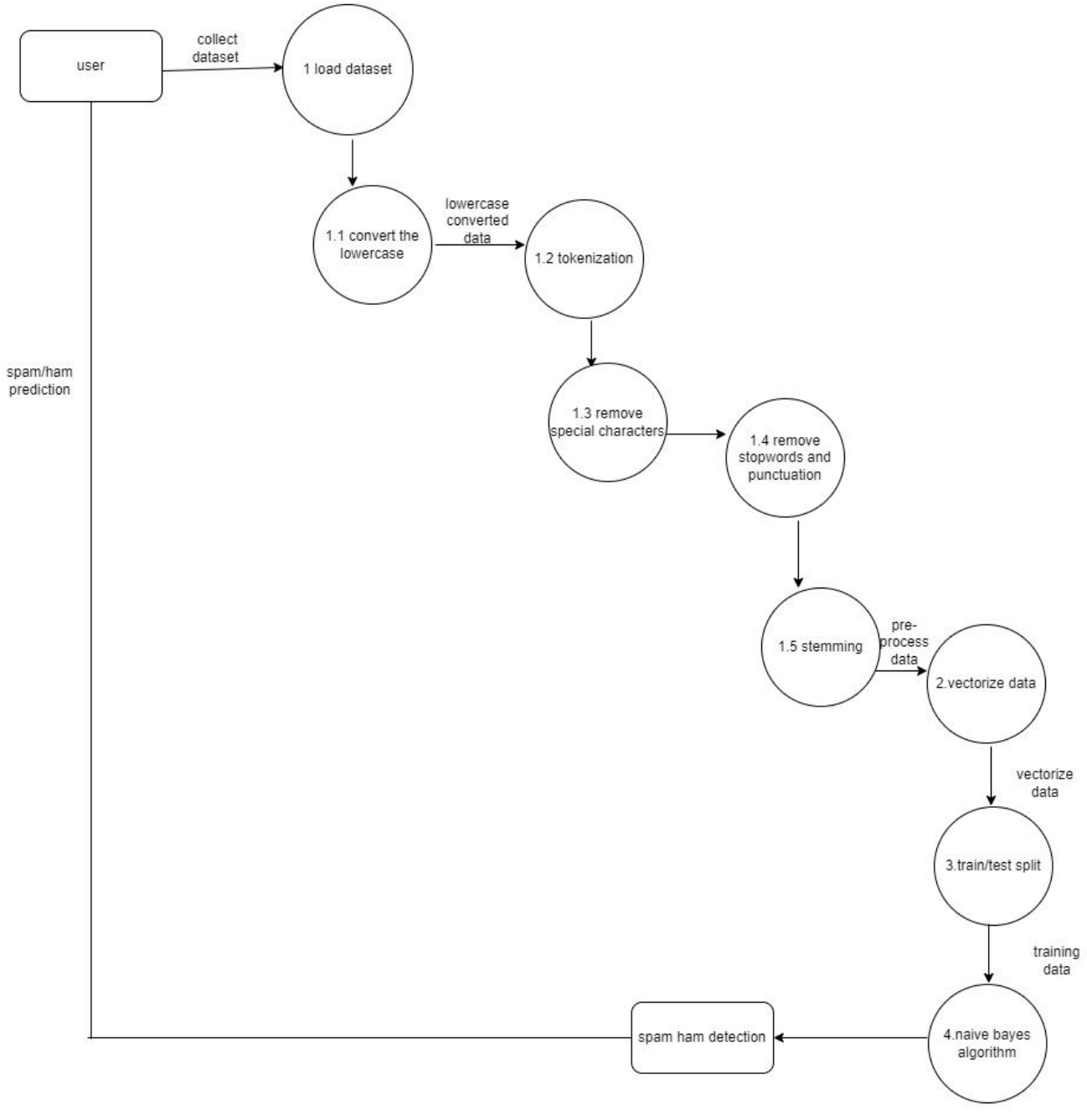


Fig 3.5 DFD level 2 of Spam Detection

Fig 3.5 is DFD level 2 of Spam Detection System.It gives the detail data flow view of the system. The user collects the data and load the dataset.The dataset are pre-processed .The pre-processing of the data is done in multiple processes.

Pre-processing of the data involves converting the data into lower case,removing special characters,tokenization(breaking the data into words),removing stopwords and punctuation and stemming.the pre-processed data is then vectorized i.e textual data are converted into number so that machine could understand.After the Vectorization of data the data is split into training data and testing data.The training data is trained with Naive Bayes Algorithm and a model for the system is built.When a user input message to the system,the system predicts the message as spam or ham and gives the result to the system.

Chapter 4:System Design

4.1 System Architecture

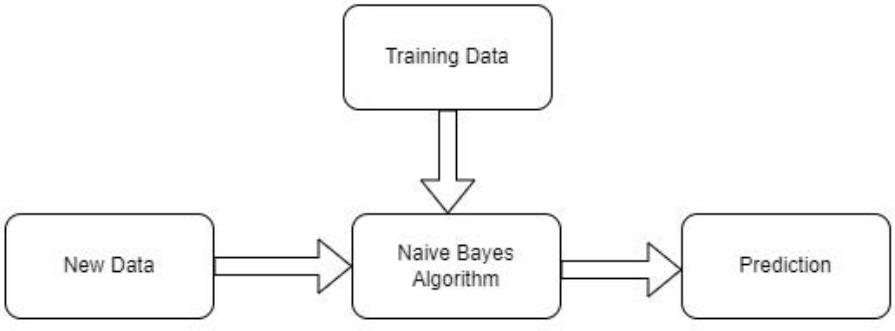


Fig 4.1 general view of spam detection using Naive Bayes Algorithm

The input to the system was taken from sample messages.These inputs were pre-processed to generate appropriate condition to extract the feature from the input. Data were cleaned by removing unnecessary punctuation, emoji and symbols,removing duplicate values,removing stopwords,tokenization.After the Pre-processing, the given data are vectorized using TFIDF vectorizer. This Calculation helped to find importance of individual word in given input.The training data is processed with Naive Bayes Algorithm and a model is built. After the model is built new test data is given to Naive Bayes classification model which is used to classify the given input in spam or ham message.

4.2 Activity Diagram

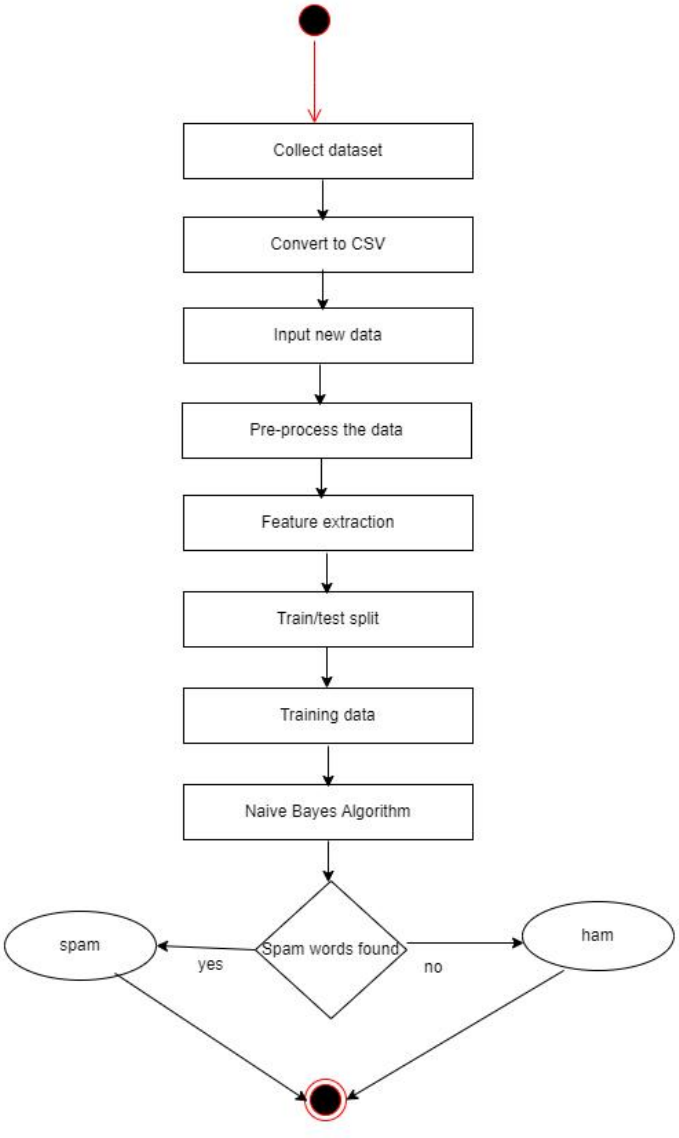


Fig 4.2 Activity Diagram of Spam Detection

Fig 4.2 visually presents a series of actions performed in the system.explains,firstly the dataset are loaded and all the packages,libraries needed are also loaded. The data is pre-processed ,vectorized and train/test split is done.After the train/test split the training data is implemented with Naive Bayes Algorithm .when new data is given to the system,the system processes and predicts the message as spam or ham

4.3 Algorithm

The project is executed in the following steps:

1. Load the dataset:

The dataset used for the project is spam dataset from Kaggle.It has two classes ham and spam.the first step for the project is to collect and load dataset.

2. Pre-process the data:

Once the dataset is loaded it needs to be pre-processed. The data required to train and build a model needs to prepared.The pre-processing of the data includes the following steps:

- Converting the data into lower case:
- Tokenization
- Removal of special characters
- Removal of stopwords
- Removal of Punctuation
- Stemming using Porter stemmer

3. Vectorize the data:

The pre-processed data are converted into a suitable form (in numbers)so that a machine can understand it .Vectorization of data is done using TF-IDF vectorizer.

4. Train the model with Naive Bayes Algorithm:

After Vectorization of the data the Naive Bayes algorithm is implemented to the dataset and model is trained and built.Naive Bayes Algorithm calculates the probability of a word falling under spam/ham classes. The more the probability of the word a class,more the probability of the messages to fall under that class.

5. Predict the result:

After the model is trained and built.New test data is given as a input and the system predicts the result as spam or ham.

Chapter 5: Methodology

5.1 Implementation

There are different steps to implement the Naive Bayes algorithm for proposed system are explained below:

1. Load the dataset:

For our purpose, we will use the collection of email messages. It is free and can be downloaded from the UCI Machine Learning Repository.

Dataset structure is simple. It contains two columns one for the category of message “spam/ham” and another for the text of the message. It contains 5572 records of different messages.

2. Data Pre-processing:

Before the algorithm application, we need to pre-process the data for implementation. The steps involved in Pre-processing of the data are:

A. Converting the data into lower case:

The data are converted into lower case using lower() method from nltk library.

For example:

original_string = HELLO WORLD

Lowercase_string = hello world

B. Tokenization:

Tokenization is the process of dividing text into a set of meaningful pieces called tokens.In this project, by calling word_tokenize() method from nltk library, the messages are broken down into a list of words, where each word is a separate item in the list.

For example:

sentence = This is a sample of tokenization.

Tokenized output =['This', 'is', 'a', 'sample', 'of', 'tokenization','.']

C. Removing stop Words and Punctuation:

Stop words are those words in a language that provide too little essence and can be ignored. For example in English language we can consider words like this, the, that etc. as stop words.These words does not add importance to the sentence. Removal of Stopwords and Punctuation are done using nltk library.

For example:

sentence = The quick brown fox jumped over the lazy dog!!

Output after stopwords and punctuation removal = quick brown fox jumped lazy dog

D. Stemming:

It is the process of removing prefixes and suffixes from words so that the base word is generated. For example stem word of both words programing, programmers is program. In this project porterstemmer is used for stemming.

For example:

Word = Loving

Output after stemming = love

3. Vectorization:

After the data is pre -processed it needs to converted into a suitable form (in numbers)so that a machine can understand it. This process of converting text into numbers is called Vectorization. The following technique is used in spam detection:

TF-IDF(Term Frequency-Inverse Document Frequency):

Implementation of TF-IDF is done in these following steps:

- Create the vocabulary.
- Compute the term frequency (TF) of each word in each document.
- Compute the inverse document frequency (IDF) of each word in the spam and ham corpus.
- Compute the TF-IDF score for each word in each document.
- Concatenate the spam and ham TF-IDF matrices to create the TF-IDF matrix.

The method to calculate the TF-IDF weights of a term in a document is given by the following formula:

TF = (frequency of tern 't' in document 'd')/(total terms in document 'd')
IDF = log10(total number of document/total documents with term 't')

The TF-IDF score for any term in a document is the product of these two terms:

*TF-IDF = TF *IDF*

4. Train/Test Split

After Vectorization the data is split into train dataset and test dataset.In this project the dataset is split 20% test dataset and 80% test dataset.

5. Implementation of Naive Bayes Algorithm

Naive Bayes algorithm is a popular and simple probabilistic algorithm used in spam detection to classify text data into spam and ham.
The Naive Bayes algorithm assumes that the features (words) in the text are independent of each other. This assumption simplifies the calculation of the conditional probability of each word.
The formula for the Naive Bayes algorithm is:

*P(S|W) = P(W|S) * P(S) / P(W)*

where:
P(S|W) is the probability of a message being spam given the words W in the Message.
P(W|S) is the probability of observing the words W in a Message that is spam.
P(S) is the prior probability of an Message being spam.
P(W) is the probability of observing the words W in any Message.

To use the Naive Bayes algorithm for spam detection, we need to compute the probabilities P(S|W) and P(W|S) for each message. We can then compare the value of P(S|W) to a threshold value to determine whether the email is spam or not.

To compute the probabilities, we can use the following formulas:

P(S) = number of spam messages / total number of messages

P(W|S) = number of times the word W appears in spam messages / total number of words in spam messages

P(W) = (number of times the word W appears in spam messages + number of times the word W appears in ham messages) / (total number of words in spam messages + total number of words in ham message.

5.1.1Tools Used:

Following are the tools and framework used for the accomplishment of this project:

1. Python

Python is used as the programming language for this project.

2. PyCharm

Pycharm is used as and IDE for the project. In the project Pycharm is used for creating a Python script that utilizes the Naive Bayes Algorithm for spam detection and then run it in the PyCharm IDE.

1. Streamlit

Streamlit is a Python Library. Streamlit is used to create an interactive web application for the project that allows users to input text data and predict the classification results. The app includes features such as visualizations, text boxes, and predict button.

2. MS Office

This is used for writing,editing and presenting the document of Spam detection System.

3. Draw.io

This is used to generate diagrams for system analysis and design of Spam Detection System. Diagrams were created using this tool in order to save time since all components are available with drag and drop functions.

5.2 Testing

System testing is done by giving different training and testing datasets. This test is done to evaluate whether the system is providing accurate summary or not. During the phase of the development of the system, our system is tested time and again. Since the project has only one feature(to predict the given text/message as spam or ham) unit testing and system testing was only done.

5.2.1. Unit Testing

In this project Spam detection every time the code is written it was run and tested to see errors.The unit testing cases are represented below:

Pre-Processing

Table 5.1 Test cases for pre-processing of spam detection

S no	Test case	Input	Expected Output	Actual Ouput	Test Result
1	convert the data into Lower case	SPAM DETECTION	spam detection	spam detection	Pass
2	Tokenization	Spam Detection	‘Spam’, ’Detection’	‘Spam’, ’Detection’	Pass
3	Removing special characters	Spam Detection@$	SpamDetection	SpamDetection	Pass
4	Removing stop words and punctuation	This is spam detection!!!	This spam detection	This spam detection	Pass
5	Stemming using porter stemmer	Loving	love	love	Pass

Spam Prediction

Table 5.2 Test cases for Spam Prediction of Spam Detection

S no	Input	Expected Output	Actual Output	Test Result
1	click on this link to receive your reward	The Message is Spam	The Message is Spam	Pass
2	congratulation you have won 1000$ call on this number	The Message is Spam	The Message is Spam	Pass

Ham Prediction

Table 5.3 Test cases for Ham Prediction of Spam Detection

S no	Input	Expected Output	Actual Output	Test Result
1	Hi how are you?	The Message is ham	The Message is ham	Pass
2	lets hang out after college	The Message is ham	The Message is ham	Pass

5.2.2. System Testing

The system testing cases are presented below:

Pre-Processing on data for ham Prediction

Table 5.4 Test cases of Pre-processing on data for Ham Prediction

S no	Test case	Input	Expected Output	Actual Output	Test Result
1	convert the data into Lower case	Hi, how are you??	hi,how are you?	hi,how are you?	Pass
2	Tokenization	Hi, how are you??	‘hi’,’,’,’how’, ’are’.’you’,’?’	‘hi’,’,’,’how’, ’are’.’you’,’?’	Pass
3	Removing special characters	Hi, how are you??	hi ,how are you?	hi ,how are you?	Pass
4	Removing stop words and punctuation	Hi, how are you??	hi	hi	Pass
5	Ham Prediction	Hi, how are you??	The message is ham	The message is ham	Pass

Pre-Processing on data for spam Prediction

Table 5 5 Test cases of Pre-processing on data for Spam Prediction

S no	Test case	Input	Expected Output	Actual Output	Test Result
1	convert the data into Lower case	CONGRATULATIONS!!!! you have won 1000$	congratulations ! !!!you have won 1000$	congratulations ! !!!you have won 1000$	Pass
2	Tokenization	CONGRATULATIONS!!you have won 1000$	‘congratulations’ ,’!’,’!’,’you’, ’have’ ’won’.’1000’. ‘$’	‘congratulations’ ,’!’,’!’,’you’, ’have’ ’won’.’1000’. ‘$’	Pass
3	Removing special characters	CONGRATULATIONS!!you have won 1000$	congratulations ! !!!you have won 1000	congratulations ! !!!you have won 1000	Pass
4	Removing stop words and punctuation	CONGRATULATIONS!!you have won 1000$	Congratulations have won 1000	Congratulations have won 1000	Pass
5	Ham Prediction	CONGRATULATIONS!!you have won 1000$	The message is spam	The message is spam	Pass

Chapter 6: Conclusion and Recommendation

6.1 Conclusion

This project “Spam Detection system” mostly focuses on predicting whether the message is spam or not.Spam detection is a must these days,a lot of spam messages keep on circulating around creating unwanted disturbance to people and luring them with attractive schemes .Detecting of those spam messages helps to stay alert and safe from frauds and thefts.The project have been completed by using python as the programming Language,Pycharm as IDE ,Streamlit for the output presentation.We imported the TF-IDF vectorizer and Naive Bayes algorithm from scikit library of python.Although there was a problem in loading the pickle file to predict the result we overcame the problem and system was made ready.

The dataset chosen was imbalance ,it consisted of more ham messages than spam messages so training the model with best accuracy was a difficult task.The system was able to predict whether the given input message was a spam or not.The model we created is not 100% accurate ,few attempts of prediction have been found wrong but there is still space for improvement in future.During the project implementation the data that was used was studied and analyzed. The Analysis of data plays a vital role on what algorithms gives the best result.To achieve the wanted result of the project data was analyzed first and then the proper selection of model was done on the basis of the data analysis.

6.2. Recommendations

The dataset used in this project is imbalanced so it doesn't give accurate result in some cases,but the performance of the model can be improved in future using more balanced and large dataset to train the model.

References:

[1] S. K. Tuteja, “Classification Algorithms for Email Spam Filtering", 2016.

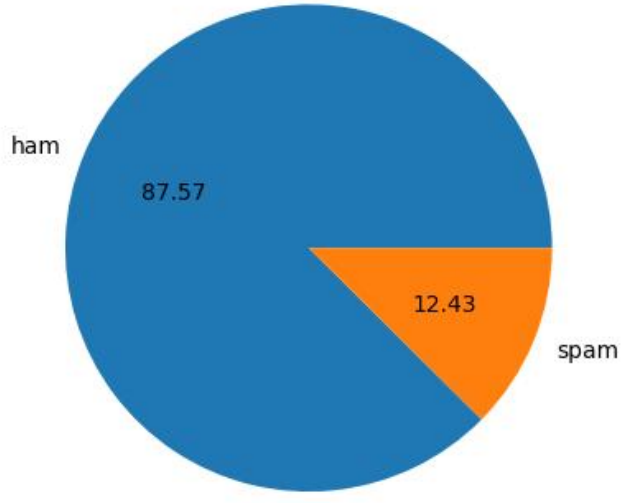
[2] G. Mujtaba, L. Shuib, R. G. Raj, N. Majeed, and M. A. Al-Garadi, "Email Classification Research Trends: Review and Open Issues", 2017.

[3] S. Ajaz, M. T. Nafis, and V. Sharma, "Spam Mail Detection Using Hybrid Secure Hash Based Naive Classifier, 2017.

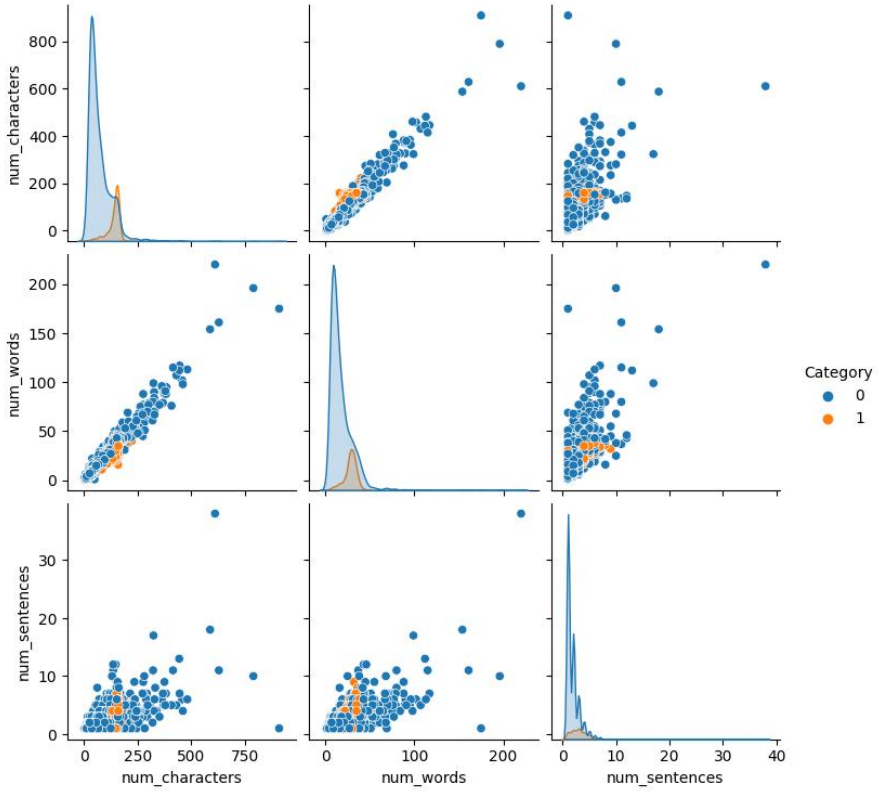
[4] Muhammad Abdul Hamid, M. S., Osho, O., Ismaila, I., & Alhassan, J. K. “Comparative Analysis of Classification Algorithms for Email Spam Detection”, 2018.

Appendix

EDA



	Category	Message	num_characters	num_words	num_sentences
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1



Testing

```
transform_text("SPAM DETECTION.")
```

```
'spam detection.'
```

```
transform_text("Spam Detection")
```

```
['Spam', 'Detection']
```

```
transform_text = "Spam Detection@$"
```

```
SpamDetection
```

```
tranform_text = "This is spam detection!!!"
```

```
This spam detection
```

```
transform_text("loving")
```

```
'love'
```

Enter the message

click on this link to receive your reward

Predict

The Message is Spam

Enter the message

congratulation you have won 1000$ call on this number

Predict

The Message is Spam

Enter the message

lets hang out after college

Predict

The message is ham

In [104]:
```
from nltk.corpus import stopwords
import string
from nltk.stem import PorterStemmer
```

In [105]:
```
def transform_Message(Message):
    Message = Message.lower()
    Message = nltk.word_tokenize(Message)
    ps = PorterStemmer()

    y = []
    for i in Message:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()

    for i in Message:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    Message = y[:]
    y.clear()

    for i in Message:
        y.append(ps.stem(i))


    return " ".join(y)
```

In [106]:
```
transform_Message('hi i am Sangam. I love dancing !!')
```

Out[106]:
```
'hi sangam love danc'
```

Vectorization

In [115]:
```
df.head()
```

Out[115]:

	Category	Message	num_characters	num_words	num_sentences	transformed_Message
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi .. avail bugi n great wo...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar ... joke wif u oni ...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor ... u c alreadi say ...
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah n't think goe usf live around though

In [116]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=3000)
```

In [117]:
```
X = tfidf.fit_transform(df['transformed_Message']).toarray()
```

Model Building

In [118]:
```
X.shape
```

Out[118]:
```
(5157, 3000)
```

In [119]:
```
y = df['Category'].values
```

In [120]:
```
#train/test split
from sklearn.model_selection import train_test_split
```

In [121]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=
```

In [122]:
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [123]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [124]:
```
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.872093023255814
[[789 116]
 [ 16 111]]
0.4889867841409692
```

In [125]:
```
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
```

```
0.9777131782945736
[[905   0]
 [ 23 104]]
1.0
```

In [126]:
```
bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
```

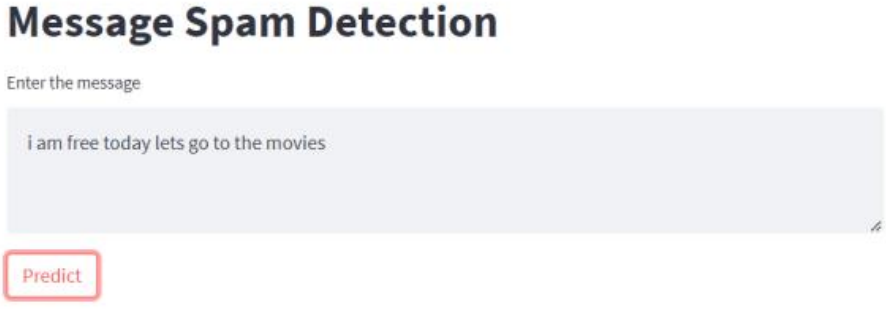
```
0.9854651162790697
[[904   1]
 [ 14 113]]
0.9912280701754386
```

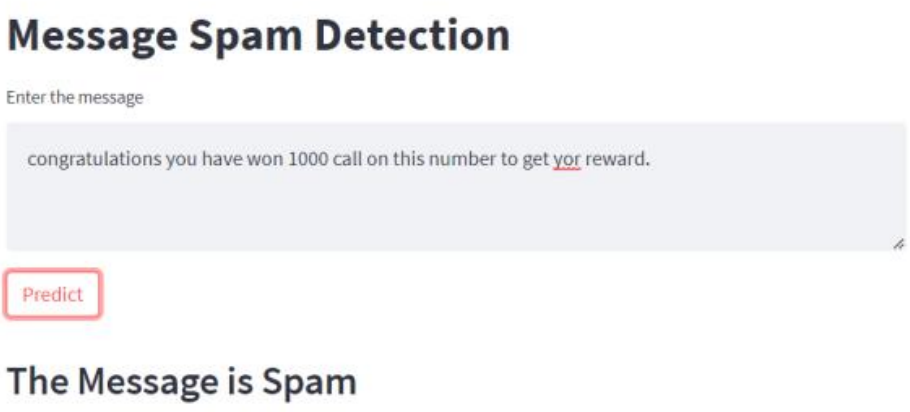
ectorizer.pkl × app.py ×

```python
import streamlit as st
import pickle
import string
from nltk.corpus import stopwords
import nltk
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()


def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))

    return " ".join(y)

tfidf = pickle.load(open('vectorizer.pkl','rb'))
model = pickle.load(open('model.pkl','rb'))

st.title("SMS Spam Classifier")

input_sms = st.text_area("Enter the message")

if st.button('Predict'):

    # 1. preprocess
    transformed_sms = transform_text(input_sms)
    # 2. vectorize
    vector_input = tfidf.transform([transformed_sms])
    # 3. predict
    result = model.predict(vector_input)[0]
    # 4. Display
    if result == 1:
        st.header("The Message is Spam")
    else:
        st.header("The message is not Spam")
```


Message Spam Detection
Enter the message
Predict


Message Spam Detection
Enter the message
You have a secret admirer! Click this link to find out who it is.
Predict
The Message is Spam


Message Spam Detection
Enter the message
i am free today lets go to the movies
Predict
The message is not Spam


Message Spam Detection
Enter the message
congratulations you have won 1000 call on this number to get yor reward.
Predict
The Message is Spam