**Optimizing Spam Filtering with Machine Learning**

|  |  |
| --- | --- |
| **TEAM ID :**NM2023TMID24667 |  |

**P.Kavin(F1C55CEB10F7371830640E58076944D0)**

**B.Vikram(681481477CD1E3D9EFB4B11539ECE46B)**

**S.Praveen Kumar(62B3476D82830B58E4E404BECA19D24D)**

**A.Gloria(E2CA1054239DFA8853D4E0151D1B5724)**

**INTRODUCTION**

Have you ever come across spam emails? In order to avoid such fraudulent emails, we use a Spam Detector which helps us to identify them and prevent you from getting tricked. Let us learn how to create Spam Filtering project using Machine Learning Techniques.

Spam Filtering?

Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the nonspam texts. It uses a binary type of classification containing the labels such as ‘ham’ (nonspam) and spam. Application of this can be seen in Google Mail (GMAIL) where it segregates the spam emails in order to prevent them from getting into the user’s inbox.

About the Project:

In this Machine Learning Spam Filtering project, we will develop a Spam Detector app using Support Vector Machine (SVM) technique for classification and Natural Language Processing. We will detect whether the piece of input text is “ham”(nonspam) or “spam”. We will split our dataset into training and testing and then train our classifier with SVM classifier.

Support Vector Machine?

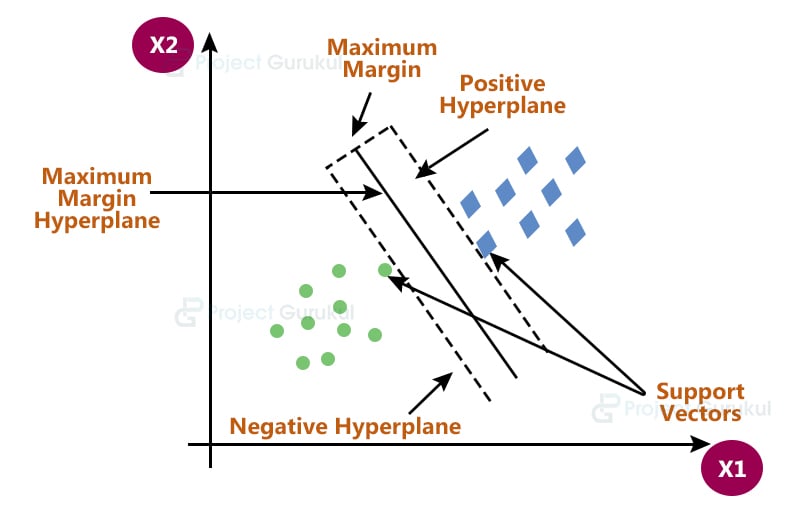
Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression problems. The main objective of SVM is to find a hyperplane in an N( total number of features)-dimensional space that differentiates the data points. So we need to find a plane that creates the maximum margin between two data point classes.

Hyperplane and Support Vectors

Hyperplanes are nothing but a boundary that helps to separate and group the data into particular classes. A Hyperplane in 2-Dimension is basically just a line. So the dimension of the hyperplane is decided on the basis of the number of features in the dataset minus 1. So a hyperplane in R2 is a line and in R3 is a plane.

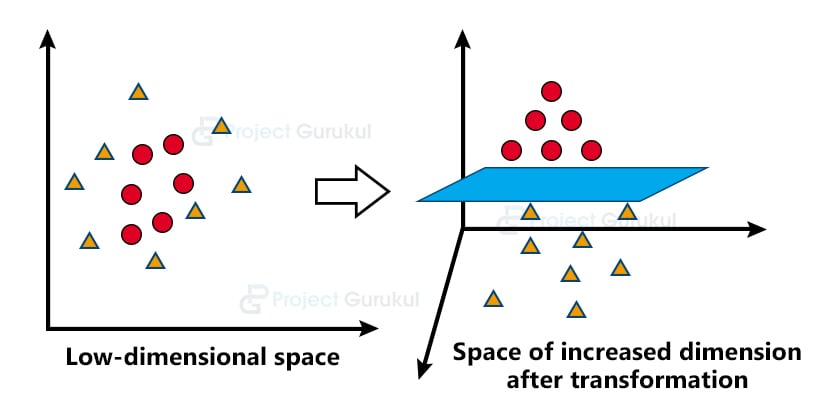
Hyperplane\_Dimension = (number **of** features) - 1

Support Vectors are data points that are close to the hyperplane and it helps to maximize the margin of the classifier. With the help of a support vector from both classes, we can form a negative hyperplane and a positive hyperplane. So basically we want to maximize the distance between the decision boundaries i.e. maximum margin hyperplane and support vectors from both sides which will minimize the error.

[](https://projectgurukul.org/wp-content/uploads/2021/08/svm-diagram.jpg)

The Kernel trick

SVM works very well on the linearly separable data i.e. the data points which can be classified using a straight line. But the question is do we need to manually decide which dimensional plane we are supposed to have for our dataset. So the answer is NO. SVM has a technique called kernel trick which takes low dimensional input space and converts into high dimensional space i.e. it converts non-separable problems into separable problems.

[](https://projectgurukul.org/wp-content/uploads/2021/08/kernel-trick.jpg)

Natural Language Processing (NLP)?

Human languages include unstructured forms of data i.e. texts and voices which cannot be understood by computers. Natural Language Processing (NLP) is an Artificial Intelligence (AI) field that enables computer programs to recognize, interpret, and manipulate human languages.

Prerequisites:

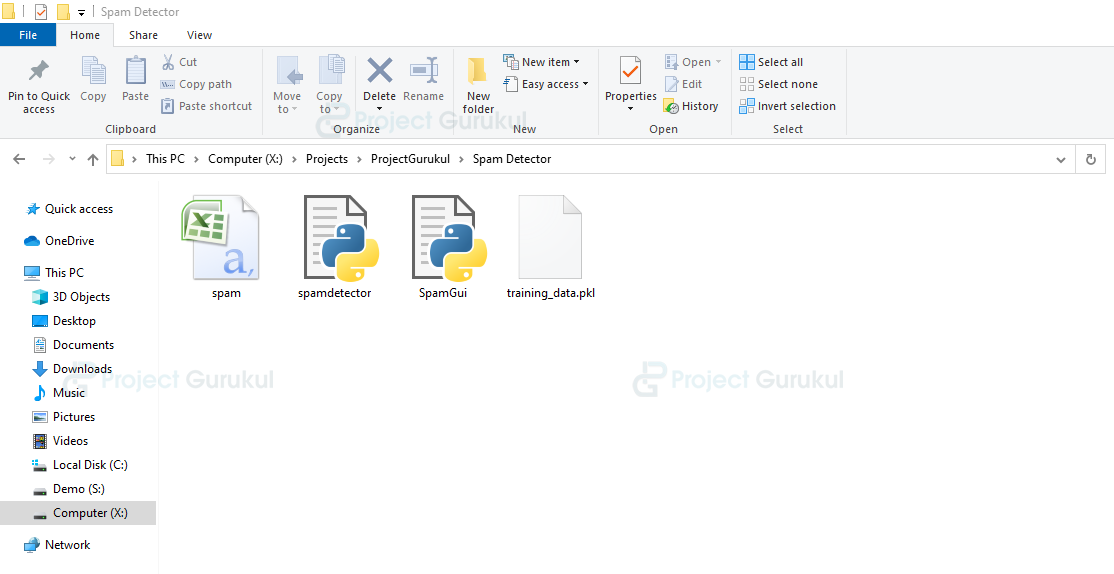
This project requires you to have a good knowledge of Python and Natural Language Processing(NLP). Modules required for this project are pandas, pickle , sklearn , numpy and nltk. You can install with using following command:

pip install pandas , pickle , sklearn , numpy , nltk

The versions which are used in this project for python and its corresponding modules are as follows:

1) python : 3.8.5  
2) sklearn : 0.24.2  
3) pickle : 4.0  
4) numpy : 1.19.5  
5) pandas : 1.1.5  
6) nltk : 3.2.5

Project Structure:

[](https://projectgurukul.org/wp-content/uploads/2021/08/project-structure.png)

**spam.csv**: Dataset for our project. It contains Labels as “ham” or “spam” and Email Text.

**spamdetector.py**: This file is used to load the dataset and train our classifier.

**training\_data.pkl**: This file contains a trained classifier in binary format which will be used to predict the output.

**SpamGui.py**: Gui file for our project where we load the trained classifier and predict the output for a given message.

Steps for developing a Spam Detector:

1) Import Libraries and initialize variables.

Firstly create a file called “spamdetector.py” and import all libraries which have been shared in the prerequisites section.

**Code:**

#import all the modules

import nltk

nltk.download("punkt")

from nltk.tokenize import word\_tokenize

from nltk.stem import LancasterStemmer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split as ttsplit

from sklearn import svm

import pandas as pd

import pickle

import numpy as np

2) Preprocessing the data

In this, we will use python’s pandas’ module to read the dataset file which we are using for training and testing purposes. Then we will use the “message\_X” variable to store features (EmailText column) and “labels\_Y” variable to store target (Label column) from our dataset.

**Code:**

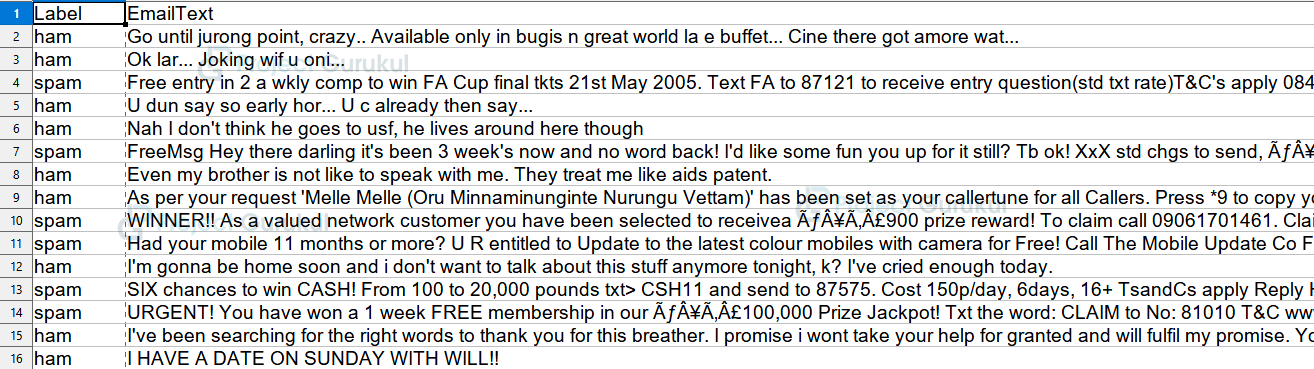
#read the dataset file

df = pd.read\_csv("spam.csv")

message\_X = df.iloc[:,1] #EmailText column

labels\_Y = df.iloc[:,0] #Label

**Dataset file:**

[](https://projectgurukul.org/wp-content/uploads/2021/08/dataset-file.png)

After we get the features and targets from our dataset we will clean the data. Firstly, we will filter out all the non-alphabetic characters like digits or symbols, and then using the natural language processing module ‘nltk’ we will tokenize our messages.

Also we will stem all the words to their root words.

**Stemming**: Stemming is the process of reducing words into their root words.  
For example, if the message contains some error word like “frei” which might be misspelled for “free”. Stemmer will stem or reduce that error word to its root word i.e. “fre”. As a result, “fre” is the root word for both “free” and “frei”.

**Code:**

#stemming variable initialization

lstem = LancasterStemmer()

**def** mess(messages):

message\_x = []

**for** me\_x **in** messages:

#filter out other datas except alphabets

me\_x=''.join(filter(lambda mes:(mes.isalpha() or mes==" ") ,me\_x))

#tokenize or split the messages into respective words

words = word\_tokenize(me\_x)

#stem the words to their root words

message\_x+=[' '.join([lstem.stem(word) **for** word **in** words])]

**return** message\_x

3) Bag of Words (vectorization)

Bag of words or vectorization is the process of converting sentence words into binary vector format. It is useful as models require data to be in numeric format. So if the word is present in that particular sentence then we will put 1 otherwise 0. This can be achieved by “TFidfVectorizer”.

Also, we will remove words that do not add much meaning to our sentence which in technical terms are called “stopwords”. For example, these words might be vowels, articles, or some common words. So for this, we will add a parameter called “stop\_words” in “TFidfVectorizer”.

**Code:**

message\_x = mess(message\_X)

#vectorization process for Machine learning Spam Filtering project

#ignore stop words i.e. words that are of least importance

tfvec=TfidfVectorizer(stop\_words='english')

#vectorizing feature data

x\_new=tfvec.fit\_transform(message\_x).toarray()

For labels, we will replace “ham” with 0 and “spam” with 1 as we can have only two outputs.

**Code:**

#replace ham and spam label with 0 and 1 respectively

y\_new = np.array(labels\_Y.replace(to\_replace=['ham', 'spam'], value=[0, 1]))

4) Training the model

We will split our dataset into 80:20 ratio where 80 for training and 20 for testing. For classifying our data, we will use the Support Vector Machine classifier technique which we have discussed above.

**Code:**

#split our dataset into training and testing part

x\_train , x\_test , y\_train , y\_test = ttsplit(x\_new,y\_new,test\_size=0.2,shuffle=**True**)

#use svm classifier to fit our model for training process

classifier = svm.SVC()

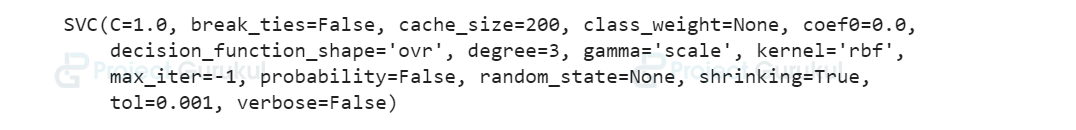
classifier.fit(x\_train,y\_train)

After training the classifier we will get the following information about best-fit parameters of SVM for our dataset.

**Kernel**: The kernel is decided on the basis of data transformation. By default, the kernel is the Radial Basis Function kernel (RBF). We can change it to linear or Polynomial depending on our dataset.

**C parameter:** The c parameter is a regularization parameter that tells the classifier how much misclassification to avoid. If the value of C is high then the classifier will fit training data very well which might cause overfitting. A low C value might allow more misclassification(errors) which can lead to lower accuracy for our classifier.

**Gamma**: Gamma is a nonlinear hyperplane parameter. High values indicate that data points that are very close to each other can be grouped. A low value indicates that data points can be grouped together even if they are separated by large distances.

[](https://projectgurukul.org/wp-content/uploads/2021/08/svm-output.png)

Now, we will save our classifier and other variables in binary or byte stream-like object format files using python’s ‘pickle’ module which we will use in the GUI file for prediction.

**Code:**

#store the classifier as well as messages feature for prediction

pickle.dump({'classifier':classifier,'message\_x':message\_x},open("training\_data.pkl","wb"))

5) Prediction using Graphical User Interface

We will load our trained model to predict whether the message is “ham” or “spam”. Now we’ll make a graphical user interface with Python’s Tkinter module and name the file as “SpamGui.py”. The Tkinter library provides the quickest and easiest way to construct GUI applications as it has several helpful libraries. Also load all the other required modules.

In this file, we will create a “SpamHam” Class which has a constructor where we will initialize all variables and load ‘training\_data.pkl’ which contains trained models and other variables using python “pickle” module.

**Code:**

#import all the modules

from tkinter import \*

import nltk

from nltk.tokenize import word\_tokenize

from nltk.stem import LancasterStemmer

from sklearn.feature\_extraction.text import TfidfVectorizer

import pickle

BG\_COLOR="#89CFF0"

FONT\_BOLD="Melvetica %d bold"

**class** SpamHam:

**def** \_\_init\_\_(self):

#initialize tkinter window

self.window=Tk()

self.main\_window()

self.lstem = LancasterStemmer()

self.tfvec=TfidfVectorizer(stop\_words='english')

self.datafile()

**def** datafile(self):

#get all datas from datafile and load the classifier.

datafile = pickle.load(open("training\_data.pkl","rb"))

self.message\_x = datafile["message\_x"]

self.classifier = datafile["classifier"]

Load the main window for tkinter which contains all the required widgets.

**Code:**

**def** main\_window(self):

#add title to window and configure it

self.window.title("Spam Detector")

self.window.resizable(width=**False**,height=**False**)

self.window.configure(width=520,height=400,bg=BG\_COLOR)

#head label for the window heading

head\_label=Label(self.window,bg="#FFA500",fg="#000",text="Welcome to

ProjectGurukul", font=FONT\_BOLD%(14),pady=10)

head\_label.place(relwidth=1)

line = Label(self.window,width=200,bg="#000")

line.place(relwidth=0.5,relx=0.25,rely=0.08,relheight=0.008)

#mid\_label

mid\_label=Label(self.window,bg=BG\_COLOR,fg="#0000FF",text="Spam Or

Ham ? Message Detector", font=FONT\_BOLD%(18),pady=10)

mid\_label.place(relwidth=1,rely=0.12)

#answer label where our prediction about user input message will be displayed

self.answer=Label(self.window,bg=BG\_COLOR,fg="#000",text="Please type

message below.", font=FONT\_BOLD%(16),pady=10,wraplength=525)

self.answer.place(relwidth=1,rely=0.30)

#textbox for user to write msg for checking

self.msg\_entry=Text(self.window,bg="#FFF",

fg="#000",font=FONT\_BOLD%(14))

self.msg\_entry.place(relwidth=1,relheight=0.4,rely=0.48)

self.msg\_entry.focus()

#check button to call the prediction function

check\_button=Button(self.window,text="Check",

font=FONT\_BOLD%(12),width=8,bg="#000",fg="#FFF",

command=lambda: self.on\_enter(None))

check\_button.place(relx=0.40,rely=0.90,relheight=0.08,relwidth=0.20)

Now we will do the same Preprocessing and vectorization process for input messages also which we have discussed earlier.

**Code:**

**def** bow(self,message):

#bag of words

#transform user's message to fixed vector length

mess\_t = self.tfvec.fit(self.message\_x)

message\_test=mess\_t.transform(message).toarray()

**return** message\_test

**def** mess(self,messages):

message\_x = []

**for** me\_x **in** messages:

#filter out other datas except alphabets

me\_x=''.join(filter(lambda mes:(mes.isalpha() or mes==" ") ,me\_x))

#tokenize or split the messages into respective words

words = word\_tokenize(me\_x)

#stem the words to their root words

message\_x+=[' '.join([self.lstem.stem(word) **for** word **in** words])]

**return** message\_x

And finally, after preprocessing and vectorizing the user input we will predict whether the piece of message is “ham” or “spam”.

**Code:**

**def** on\_enter(self,event):

#get the user input from textbox

msg=str(self.msg\_entry.get("1.0","end"))

#preprocess the message

message=self.mess([msg])

#predict the label i.e. ham or spam for users message

self.answer.config(fg="#ff0000",text="Your message is : "+

("spam" **if** self.classifier.predict(self.bow(message)).reshape(1,-1)

**else** "ham"))

#runwindow

**def** run(self):

self.window.mainloop()

# run the file

**if** \_\_name\_\_=="\_\_main\_\_":

app = SpamHam()

app.run()

Machine Learning Spam Filtering Output



Conclusion:

In this project, we learned how to develop a Spam Detector app that will identify the spam messages using the Support Vector Machine technique and Natural Language Processing and then display the output(spam or ham) on GUI. Spam Detector is useful for almost all internet service providers in order to differentiate spam contents and prevent the users from cheating.