

SMS Spam Detection

Ambika Methre and Kavali Veena

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

March 16, 2021

SMS SPAM DETECTION

Mini Project Report

Submitted in partial fulfillment of the III year(semester-1) BE degree as per the requirements of Osmania University, Hyderabad

By

Ambika Methre

Kavali Veena

160618733125

160618733148

CERTIFICATE

This is to certify that the mini project titled "SMS Spam Detection" is a bonafied work Carried over by Ms. Ambika Methre (Hall Ticket No.: 160618733125), Ms. Kavali Veena (Hall Ticket No.: 160618733148) in partial fulfillments for the award of the degree Bachelor of Engineering from Osmania University during the III Semester-1 of their BE course during the academic year 2020-2021

Dr. B.V Ramana Murthy Head, Department of CSE

MS. Nadia Anjum Project Guide

ACKNOWLEGEMENT

We express our deep sense of gratitude respected guide Ms.Nadia Anjum Asst. Professor, CSE Dept. for her valuable help and guidance. we are thankful to him for the encouragement she gave us in completing the project.

We sincerely thank our coordinator Mrs. Shivani Yadao and Mrs. Heerekar Meenal for their cooperation and help.

We are also grateful to respected Prof. DR. B.V Ramana Murthy HOD(CSE) for permitting us to utilize all the necessary facilities of the institution.

We are also thankful to all the other faculty and staff members of our department for their kind cooperation and help.

Lastly, we would like to express our deep appreciation towards our classmates and our indebtedness to our parents for providin^g us the moral support and encouragement.

CONTENTS OF FIGURES

Title	Page no
Abstract	01
LIST OF FIGURES	
Fig 1.1 Introduction	02
Fig1. 2 SMS spam detection	03
Fig 1.3 Flow chart of sms spam detection	04
Fig 2 Msg representation	06
Fig 3 Methodology	07
Fig 4 Reporting the spam	15
Fig 5 Measure of accuracy,SC,BH	20
Fig 6.1Testing the data	21
Fig 6.2 Evaluating the model	22
Fig 7.1 Importing libraries	23
Fig 7.2 Viewing the data	23
Fig 7.3 Dropping the unwanted columns	24
Fig 7.4 Viewing the data after removal of unwanted column	24
Fig 7.5 Viewing the unique data	25
Fig 7.6 importing the libraries top plot the graph	25
Fig 7.7 Plotting the graph for the given data	26
Fig 7.8 Converting all strings to their lowercase	26
Fig 7.9 Removing all punctuations	27
Fig 7.10 Tokenization	27

Fig 7.11 Counting frequencies & implementing

Bag of word in sckit-learn	28
Fig 7.12 Data preprocessing with	
Count vectorizer()	28
Fig 7.13 Splitting the data	29
Fig 7.14 Training the data	29
Fig 7.15 Fitting the data & return the matrics	30
Fig 7.16 Implementation of navie bayes theorem	30
Fig 7.17 Evaluating the model	31

CONTENT

Chapter Name	Page no.
Acknowledgement	
1.Introduction	02
2.Requirements	06
3.Methodology	07
4.Introduction to the machine	
Learning algorithms	08
5.Code/coding template	17
6.Testing	21
7.Output Screen	23
8.Bibilography	32

ABSTRACT

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollars industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. In parts of Asia, up to 30% of text messages were spam in 2012. Lack of real databases for SMS spams, short length of messages and limited features, and their informal language are the factors that may cause the established email filtering algorithms to underperform in their classification. In this project, a database of real SMS Spams from UCI Machine Learning repository is used, and after pre processing and feature extraction, different machine learning techniques are applied to the database. Finally, the results are compared and the best algorithm for spam filtering for text messaging is introduced. Final simulation results using 10-fold cross validation shows the best classifier in this work reduces the overall error rate of best model in original paper citing this dataset by more than half. Algorithms used in this technique are: Logistic regression (LR), K-nearest neighbour(K-NN) and Decision tree (DT) are used for classification of spam messages in mobile device communication. The SMS spam collection set is used for testing the method.

1.INTRODUCTION



Fig-1.1 .introduction

Short Message Services (SMS) is far more than just a technology for a chat. SMS technology evolved out of the global system for mobile communications standard, an internationally accepted. Spam is the abuse of electronic messaging systems to send unsolicited messages in bulk indiscriminately. While the most widely recognized form of spam is email spam, the term is applied to similar abuses in other media and mediums. SMS Spam in the context is very similar to email spams, typically, unsolicited bulk messaging with some business interest.

SMS spam is used for commercial advertising and spreading phishing links. Commercial spammers use malware to send SMS spam because sending SMS spam is illegal in most countries. Sending spam from a compromised machine reduces the risk to the spammer because it obscures the provenance of the spam. SMS can have a limited number of characters, which includes alphabets, numbers, and a few symbols. A look through the messages shows a clear pattern. Almost all of the spam messages ask the users to call a number, reply by SMS, or visit some URL. This pattern is observable by the results obtained by a simple SQL query on the spam. The low price and the high bandwidth of the SMS network have attracted a large amount of SMS spam. Every time SMS spam arrives at a user's inbox, and the mobile phone alerts the user to the incoming message. When the user realizes that the message is unwanted, he or she will be disappointed, and also SMS spam takes up some of the mobile phone's storage.



Fig-1.2 .SMS spam representation

SMS spam detection is an important task where spam SMS messages are identified and filtered. As more significant numbers of SMS messages are communicated every day, it is challenging for a user to remember and correlate the newer SMS messages received in context to previously received SMS. Thus, using the knowledge of artificial intelligence with the amalgamation of machine learning, and data mining we will try to develop web-based SMS text spam or ham detector.

A number of major differences exist between spam-filtering in text messages and emails. Unlike emails, which have a variety of large datasets available, real databases for SMS spams are very limited. Additionally, due to the small length of text messages, the number of features that can be used for their classification is far smaller than the corresponding number in emails. Here, no header exists as well. Additionally, text messages are full of abbreviations and have much less formal language that what one would expect from emails. All of these factors may result in serious degradation in performance of major email spam filtering algorithms applied to short text messages.

In this project, the goal is to apply different machine learning algorithms to SMS spam classification problem, compare their performance to gain insight and further explore the problem, and design an application based on one of these algorithms that can filter SMS spams with high accuracy. Feature extraction and initial analysis of data is done in MATLAB, then applying different machine learning algorithms is done in python using scikitlearn library.



Fig-1.3 flow chart of sms spam detection

This is three parts of bold series, where we will understand the in and out of spam or ham classifier from the aspect of Artificial Intellegence concepts, and work with various classification algorithm in jupyter notebook and select the one algorithm based on performance criteria. Then, we will develop the python web based SMS text spam or ham detector. Spam detection is one of the major applications of Machine Learning in the inter webs today. Pretty much all of the major email service providers have spam detection systems built in and automatically classify such mail as 'Junk Mail'.

In this project Naive Bayes algorithm is use to create a model that can classify dataset SMS messages as spam or not spam, based on the training we give to the model. Usually they have words like 'free', 'win', 'winner', 'cash', 'prize' and the like in them as these texts are designed to catch your eye and in some sense tempt you to open them. Also, spam messages tend to have words written in all capitals and also tend to use a lot of exclamation marks. To the recipient, it is usually pretty straightforward to identify a spam text and our objective here is to train a model to do that for us!

Being able to identify spam messages is a binary classification problem as messages are classified as either 'Spam' or 'Not Spam' and nothing else. Also, this is a supervised learning problem, as we will be feeding a labelled dataset into the model, that it can learn from, to make future predictions.

2. REQUIREMENTS

Hardware Requirements

Processor: 1.5GHz or above

RAM: 4GB or more

HDD: 100GB or above

Software Requirements

Anaconda Jupyter Note book

Anaconda3 jupyter note book is used to write and execute the python code using machine learning algorithms such as: Naive Bayes Theorem, Support Vector Machines, Neural Networks, K- Nearest Neighbour, AdaBoost, and Random Forest.

3.METHODOLOGY



Fig.3 Methodology

4.INTRODUCTION TO THE MACHINE LEARNING ALGORITHMS

I. K-Nearest Neighbours

k-nearest neighbour can be applied to the classification problems as a simple instance-based learning algorithm. In this method, the label for a test sample is predicted based on the majority vote of its k nearest neighbours.

K	Overall error %	Spam Caught (SC)	Blocked Hams (BH)
		%	%
2	2.78	81.3	0.46
10	2.53	82.6	0.40
20	2.98	78.8	0.35
50	3.4	74.8	0.24
100	4.14	68.4	0.16

TABLE III

10-fold cross validation error of k-nearest neighbour classifier

II. Support Vector Machines (SVM)

In support vector machine is applied to the dataset. Table I I shows the 10-fold cross validation results of SVM with different kernels applied to the dataset with extracted features. As it is shown in the table, linear kernel gains better performance compared to other mappings. Using the polynomial kernel and increasing the degree of the polynomial from two to three shows improvement in error rates, however the error rate does not improve when the degree is increased further. Radial basis function (RBF) is another kernel applied here to the dataset. RBF kernel on two samples x1 and x2 is expressed by following equation:

$$K(x1, x2) = \exp(-kx1 - x2k \ 2 \ 2 \ 2\sigma 2)$$

Kernel Function	Overall Error %	Spam Caught	Blocked Hams
		(SC) %	(BH) %
Linear	1.18	93.8	0.47
Degree-2	2.03	85.7	0.27
Polynomial			
Degree-3	1.64	89.7	0.4.
Polynomial			
Degree-4	1.70	90.65	0.6
Polynomial			

Radial Basis	2.61	81.45	0.32
Function			
Sigmoid	13.4	0	0

TABLE-II

10-fold cross validation error of SVM with different kernel functions on dataset

From the analysis of results, we notice that the length of the text message (number of characters used) is a very good feature for the classification of spams. Sorting features based on their mutual information (MI) criteria shows that this feature has the highest MI with target labels. Additionally, going through the misclassified samples, we notice that text messages with length below a certain threshold are usually hams, yet because of the tokens corresponding to the alphabetic words or numeric strings in the message they might be classified as spams.

While applying SVM with different kernels increases the complexity of the model and subsequently the running time of training the model on data, the results show no benefit compared to the multinomial naive Bayes algorithm in terms of accuracy.

III. Random Forest

Random forests is an averaging ensemble method for classification. The ensemble is a combination of decision trees built from a bootstrap sample from training set. Additionally, in building the decision tree, the split which is chosen when splitting a node is the best split only among a random set of features. This will increase the bias of a single model, but the averaging reduces the variance and can compensate for increase in bias too. Consequently, a better model is built. In this work, the implementation of random forests in scikitlearn python library is used, which averages the probabilistic predictions. Two number of estimators are simulated for this method. With 10 estimators, the overall error is 2.16%, SC is 87.7 %, and BH is 0.73%. Using 100 estimators will result in overall error of 1.41 %, SC of 92.2 %, and BH of 0.51 %. We observe that comparing to the naive Bayes algorithm, although the complexity of the model is increased, yet the performance does not show any improvement.

IV. Naive Bayes Theorem

Step 1: Introduction to the Naive Bayes Theorem

Bayes theorem is one of the earliest probabilistic inference algorithms developed by Reverend Bayes (which he used to try and infer the existence of God no less) and still performs extremely well for certain use cases. It's best to understand this theorem using an example. Let's say you are a member of the Secret Service and you have been deployed to protect the Democratic presidential nominee during one of his/her campaign speeches. Being a public event that is open to all, your job is not easy and you have to be on the constant lookout for threats. So one place to start is to put a certain threat-factor for each person. So based on the features of an individual, like the age, sex, and other smaller factors like is the person carrying a bag?, does the person look nervous? etc. you can make a judgement call as to if that person is viable threat.

If an individual ticks all the boxes up to a level where it crosses a threshold of doubt in your mind, you can take action and remove that person from the vicinity. The Bayes theorem works in the same way as we are computing the probability of an event(a person being a threat) based on the probabilities of certain related events(age, sex, presence of bag or not, nervousness etc. of the person).

One thing to consider is the independence of these features amongst each other. For example if a child looks nervous at the event then the likelihood of that person being a threat is not as much as say if it was a grown man who was nervous. To break this down a bit further, here there are two features we are considering, age AND nervousness. Say we look at these features individually, we could design a model that flags ALL persons that are nervous as potential threats. However, it is likely that we will have a lot of false positives as there is a strong chance that minors present at the event will be nervous. Hence by considering the age of a person along with the 'nervousness' feature we would definitely get a more accurate result as to who are potential threats and who aren't.

This is the 'Naive' bit of the theorem where it considers each feature to be independent of each other which may not always be the case and hence that can affect the final judgement.

In short, the Bayes theorem calculates the probability of a certain event happening(in our case, a message being spam) based on the joint probabilistic distributions of certain other events(in our case, a message being classified as spam). We will dive into the workings of the Bayes theorem later in the mission, but first, let us understand the data we are going to work with.

Step 2: Understanding our dataset

We will be using a <u>dataset</u> from the UCI Machine Learning repository which has a very good collection of datasets for experimental research purposes.

Ham	Go until jurong point, crazy Available only in bugis n great world la e buffet Cine then
Ham	Ok lar Joking wif u oni
	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 t
Spam	08452810075over18's
Ham	U dun say so early hor U c already then say
Ham	Nah I don't think he goes to usf, he lives around here though
	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun yo
Spam	to rcv
Ham	Even my brother is not like to speak with me. They treat me like aids patent.
Ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set a WINNER!! As a valued network customer you have been selected to receive a & £900 prize
Spam	only.
Spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles
Ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575.
Spam	info
-	URGENT! You have won a 1 week FREE membership in our å£100,000 Prize Jackpot! 7
Spam	4403LDNW1A7RW18
Ham	I've been searching for the right words to thank you for this breather. I promise i wont tak
Ham	I HAVE A DATE ON SUNDAY WITH WILL!!
	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message o
Spam	xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
Ham	Oh ki'm watching here:)

The columns in the data set are currently not named and as you can see, there are 2 columns.

The first column takes two values, 'ham' which signifies that the message is not spam, and 'spam' which signifies that the message is spam.

The second column is the text content of the SMS message that is being classified.

Step 3: Data Preprocessing

Now that we have a basic understanding of what our dataset looks like, lets convert our labels to binary variables, 0 to represent 'ham'(i.e. not spam) and 1 to represent 'spam' for ease of computation.

You might be wondering why do we need to do this step? The answer to this lies in how scikit-learn handles inputs. Scikit-learn only deals with numerical values and hence if we were to leave our label values as strings, scikit-learn would do the conversion internally(more specifically, the string labels will be cast to unknown float values).

Our model would still be able to make predictions if we left our labels as strings but we could have issues later when calculating performance metrics, for example when calculating our precision and recall scores. Hence, to avoid unexpected 'gotchas' later, it is good practice to have our categorical values be fed into our model as integers

Step 4: Bag of words

What we have here in our data set is a large collection of text data (5,572 rows of data). Most ML algorithms rely on numerical data to be fed into them as input, and email/sms messages are usually text heavy.

Here we'd like to introduce the Bag of Words (BOW) concept which is a term used to specify the problems that have a 'bag of words' or a collection of text data that needs to be worked with. The basic idea of BOW is to take a piece of text and count the frequency of the words in that text. It is important to note that the BOW concept treats each word individually and the order in which the words occur does not matter.

Using a process which we will go through now, we can convert a collection of documents to a matrix, with each document being a row and each word(token) being the column, and the corresponding (row, column) values being the frequency of occurrence of each word or token in that document.

Data preprocessing with CountVectorizer()

Some of important parameters of CountVectorizer().

- 1. lowercase = True The lowercase parameter has a default value of True which converts all of our text to its lowercase form.
- 2. Token pattern = (?u)\b\w\w+\b The token pattern parameter has a default regular expression value of (?u)\b\w\w+\b which ignores all punctuation marks and treats them as delimiters, while accepting alphanumeric strings of length greater than or equal to 2, as individual tokens or words.
- 3. Stop words The stop words parameter, if set to english will remove all words from our document set that match a list of English stop words which is defined in scikit-learn. Considering the size of our dataset and the fact that we are dealing with SMS messages and not larger text sources like e-mail, we will not be setting this parameter value.

Step 5: Training and testing sets

Now that we have understood how to deal with the Bag of Words problem we can get back to our dataset and proceed with our analysis. Our first step in this regard would be to split our dataset into a training and testing set so we can test our model later. Instructions: Split the dataset into a training and testing set by using the train_test_split method in sklearn. Split the data using the following variables:

- X__train is our training data for the 'sms_message' column.
- Y_train is our training data for the 'label' column
- X_test is our testing data for the 'sms_message' column.
- y_test is our testing data for the 'label' column Print out the number of rows we have in each our training and testing data.

Step 6: Applying Bag of Words processing to our dataset

Now that we have split the data, our next objective is to follow the steps from Step 2: Bag of words and convert our data into the desired matrix format. To do this we will be using CountVectorizer() as we did before. There are two steps to consider here:

- Firstly, we have to fit our training data (X_train) into CountVectorizer() and return the matrix.
- Secondly, we have to transform our testing data (X_test) to return the matrix. Note that X_train is our training data for the 'sms_message' column in our dataset and we will be using this to train our model.

X_test is our testing data for the 'sms_message' column and this is the data we will be using(after transformation to a matrix) to make predictions on. We will then compare those predictions with y_test in a later step.

Step-7: Implementation of Naive Bayes Machine Learning Algorithm

I will use sklearns sklearn.naive_bayes method to make predictions on our dataset for SMS Spam Detection.

Specifically, we will be using the multinomial Naive Bayes implementation. This particular classifier is suitable for classification with discrete features. It takes in integer word counts as its input.

from sklearn.naive_bayes import MultinomialNB

naive_bayes = MultinomialNB()

naive_bayes.fit(training_data,y_train)

MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

predictions = naive_bayes.predict(testing_data)

Step 8: Evaluating our model

Now that we have made predictions on our test set, our next goal is to evaluate how well our model is doing. There are various mechanisms for doing so, but first let's do quick recap of them.

Accuracy measures how often the classifier makes the correct prediction. It's the ratio of the number of correct predictions to the total number of predictions (the number of test data points).

Precision tells us what proportion of messages we classified as spam, actually were spam. It is a ratio of true positives(words classified as spam, and which are actually spam) to all positives(all words classified as spam, irrespective of whether that was the correct classification), in other words it is the ratio of

True Positives/(True Positives + False Positives)

Recall(sensitivity) tells us what proportion of messages that actually were spam were classified by us as spam. It is a ratio of true positives(words classified as spam, and which are actually spam) to all the words that were actually spam, in other words it is the ratio of

True Positives/(True Positives + False Negatives)

For classification problems that are skewed in their classification distributions like in our case, for example if we had a 100 text messages and only 2 were spam and the rest 98

Step 9: Conclusion

One of the major advantages that Naive Bayes has over other classification algorithms is its ability to handle an extremely large number of features. In our case, each word is treated as a feature and there are thousands of different words. Also, it performs well even with the presence of irrelevant features and is relatively unaffected by them. The other major advantage it has is its relative simplicity. Naive Bayes' works well right out of the box and tuning it's parameters is rarely ever necessary, except usually in cases where the distribution of the data is known. It rarely ever overfits the data. Another important advantage is that its model training and prediction times are very fast for the amount of data it can handle. All in all, Naive Bayes' really is a gem of an algorithm!



Fig-4 Reporting the spam

V. AdaBoost with Decision Tree

AdaBoost is a boosting ensemble method which sequentially builds classifiers that are modified in favour of misclassified instances by previous classifiers [5]. The classifiers it uses can be as weak as only slightly better than random guessing, and they will still improve the final model. This method can be used in conjunction with other methods to improve the final ensemble model.

In each iteration of Ada Boost, certain weights are applied to training samples. These weights are distributed uniformly before first iteration. Then after each iteration, weights for misclassified labels by current model are increased, and weights for correctly classified samples are decreased. This means the new predictor focuses on weaknesses of previous classifier.

Model	SC%	BH%	Accuracy %
Multinomial NB	94.47	0.51	98.88
SVM	92.99	0.31	98.86
K-Nearest neighbour	82.60	0.40	97.47
Random Forest	90.62	0.29	98.57

Ada Boost with	92.17	0.51	98.59
Decision tree			

TABLE-III

.

Final results of different classifiers applied to SMS Spam dataset

We tried the implementation of Ada boost with decision trees using scikit-learn library. Using 10 estimators, the simulation shows 2.1% overall error rate, 87.7% SC, and 0.74% BH. Increasing the number of estimators to 100 will change these values to 1.41%, 92.2%, and 0.51% respectively. Like Random Forests, although the complexity is much higher, naive Bayes algorithm still beats Ada boost with decision trees in terms of performance.

Performance Measure

Accuracy = True Positive + True Negetive Total Number of Test Data

Spams caught (SC) = False negative cases Number of Spams

Blocked hams (BH) = False Positive cases Number of Hams

Fig.5 Measure of Accuracy ,SC,BH

12

6.TESTING

🔿 OneDrive/Desktop/mini project/ 🗴 🧧 code - Jupyter Notebook 🛛 🗴 🕂		-	٥	×
\leftarrow \rightarrow C (1) localhost8888/notebooks/OneDrive/Desktop/mini%20project/code.ipynb	ť≡	Ē		
🕒 google 🚯 BigBlueButton - V S				
💭 JUPYter code Last Checkpoint: an hour ago (autosaved)	Logou	rt		
File Edit View Insert Cell Kernel Widgets Help Trusted	Python 3	0		
Instantiate the CountVectorizer method				*
<pre>In [45]: count_vector = CountVectorizer()</pre>				
Fit the training data and then return the matrix				
<pre>In [46]: training_data = count_vector.fit_transform(X_train)</pre>				
In [47]: training_data				
Out[47]: <4457x7774 sparse matrix of type ' <class 'numpy.int64'="">' with 59357 stored elements in Compressed Sparse Row format></class>				
<pre>In [48]: # Transform testing data and return the matrix. testing_data = count_vector.transform(X_test)</pre>				
In [49]: testing_data				
Out[49]: <1115x7774 sparse matrix of type ' <class 'numpy.int64'="">' with 13599 stored elements in Compressed Sparse Row format></class>				
📲 🖓 Type here to search O 🛱 💽 C I 🖉 🔚 😭 🖪 🛣 💆 📲	EN ENG	01:05 07-02	5 PM -2021	

Fig.6.1 Testing the data

🔿 OneDrive/Desktop/mini project/ 🗙 🧧 code - Jupyter Notebook 🛛 🗙 🕂				-	٥	Х
\leftarrow \rightarrow C () localhost.8888/notebooks/OneDrive/Desktop/mini%20project/code.jpynb		to	₹_=	Ē		
🗅 google 🧿 BigBlueButton - V S						
C JUPYTET code Last Checkpoint: an hour ago (autosaved)		ę	Logou	ıt		
File Edit View Insert Cell Kernel Widgets Help	Trusted		Python 3	0		
<pre>In [50]: from sklearn.naive_bayes import MultinomialNB naive_bayes = MultinomialNB() naive_bayes.fit(training_data,y_train)</pre>						4
<pre>Out[50]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)</pre>						
<pre>In [53]: predictions = naive_bayes.predict(testing_data)</pre>						
In [54]: predictions						
Out[54]: array([0, 0, 0,, 0, 0, 0], dtype=int64)						
Evaluating our SMS Spam Detection Model						
<pre>In [55]: from sklearn.metrics import accuracy_score, precision_score, recall_score f1_score print('Accuracy_score: {}'.format(accuracy_score(_test, predictions))) print('Precision score: {}'.format(precision_score(y_test, predictions))) print('Recall score: {}'.format(recall_score(y_test, predictions))) print('F1 score: {}'.format(f1_score(y_test, predictions)))</pre>						
Accuracy score: 0.9847533632286996 Precision score: 0.9420289855072463 Recall score: 0.935251798561151 F1 score: 0.9386281588447652						
■ P Type here to search O Ħ O O O O O O O O O O O O O O O O O		^	EN EN	01:02 3 07-02	5 PM -2021	, ,

Fig.6.2 Evaluating the model

7. OUTPUT SCREENS

🔵 OneDrive/Desktop/mini project/ 🗙 🧧 cc	ode - Jupyter Notebook × +						-	0 ×
\leftarrow \rightarrow C (i) localhost8888/	'notebooks/OneDrive/Desktop/mini%20p	roject/code.ipy	ynb		C8 - 1	io f≞	<u>ه</u>	
🕒 google 🛛 BigBlueButton - V S								
💭 jupyter 🛛 code L	ast Checkpoint: an hour ago (autosaved)				ć	Logo	ut	
File Edit View I	nsert Cell Kernel Widgets Help				Trusted	Python 3	0	
B + × 4 6 1	🔸 🗣 🕅 Run 🔳 🖒 🇭 Code	~						
								í
luon a								
Impo	orting libraries							
In [1]: import import import	numpy as np pandas as pd nltk							
In [2]: import df_sms	pandas = pd.read_csv('spam.csv',encoding='l	atin-1')						
In [3]: df_sms								
Out[3]:	v1 v/	? Unnamed: 2 U	Innamed: 3 Unnamed: 4					
0 1	nam Go until jurong point, crazy. Available only	. NaN	NaN NaN					
1 1	nam Ok lar Joking wif u oni	. NaN	NaN NaN					
2 sp	vam Free entry in 2 a wkly comp to win FA Cup fina	. NaN	NaN NaN					
3 1	iam U dun say so early hor U c already then say.	. NaN	NaN NaN					
4 1	nam – Nan i don timink në goës to usi, në ives aro	. Nan	nan nan					
 5567 st	pam. This is the 2nd time we have tried 2 contact u.	. NaN	NaN NaN					
♥ Type here to search	o 🛱 📀 💽	0	💼 📬 🖪	N 🖸 🖬		^ ∎ EN	G 01:11 F	°M □



OneDrive/Desktop/mini project/ X	🧧 code - Jupyter Notebook 🗙 🕂			- 0	×
\leftrightarrow \rightarrow C (i) localho	st8888/notebooks/OneDrive/Desktop/mini%20project/code.ipynb		చం ∠≐	G 🔋	
🕒 google 👩 BigBlueButton - V S					
💭 Jupyter	code Last Checkpoint: an hour ago (autosaved)		el Logou	ıt	
File Edit	ïew Insert Cell Kernel Widgets Help	Trusted	Python 3	0	
B + × 4	In A Y H Run ■ C I Code Y ■				
	4 ham Nah I don't think he goes to usf, he lives aro NaN NaN NaN				*
	m				
	5567 spam This is the 2nd time we have tried 2 contact u NaN NaN NaN				
	5568 ham Will _ b going to esplanade fr home? NaN NaN NaN				
	5569 ham Pity, * was in mood for that. Soany other s NaN NaN NaN NaN				
	5570 ham The guy did some bitching but I acted like i'd NaN NaN NaN				
	5571 ham Rofl. Its true to its name NaN NaN NaN				
	i572 rows × 5 columns				
In [4]:	Hf_sms.head()				
Out[4]:	vd v2 linnamed 2 linnamed 3 linnamed 4				
	Gountiliumon point crazy Available only NaN NaN NaN				
	1 ham OklarJoking wiji u oni NaN NaN NaN				
	2 spam Free entry in 2 a wkly comp to win FA Cup fina NaN NaN NaN				
	3 ham U dun say so early hor U c aiready then say NaN NaN NaN				
	4 ham Nah I don't think he goes to usf, he lives aro NaN NaN NaN				
In [5]:	H_sms.head()				
Out[5]:				01:11 PM	-
P Type here to search	O 🖽 💟 💟 🖉 🔚 🛄 🔤 🕍 💭 👊		▲ ENC	07-02-202	14

Fig.7.2 viewing the data

🔿 OneDrive/Desktop/mini project/ 🗴 🧧 code - Jupyter Notebook 🛛 🗴 🕂			-	٥	×
← → ♂ ① localhost8888/notebooks/OneDrive/Desktop/mini%20project/code.ipynb	τô	₹≡	ſ <u></u>		
🖰 google 🔞 BigBlueButton - V S					
🂢 JUpyter code Last Checkpoint: an hour ago (autosaved)	2	Logo	ut		
File Edit View Insert Cell Kernel Widgets Help Trusted	F	^D ython 3	0		
Ei + % (2) Ei ↑ ↓ NRun ■ C >> Code					
In [5]: df_sms.head()					*
Out [5]:					
Onlamenta 2 onnament 2 onnament 4 Onlamenta 4					
1 ham OklarJoking wifu oni NaN NaN NaN					
2 spam Free entry in 2 a widy comp to win FA Cup fina NaN NaN NaN					
3 ham U dun say so early hor U c already then say NaN NaN NaN					
4 ham Nah I don't think he goes to usf, he lives aro NaN NaN NaN					
Dropping the unwanted columns Unnamed:2, Unnamed: 3 and Unnamed	1:4				
<pre>In [6]: df_sms = df_sms.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1) df_sms = df_sms.rename(columns=("v1":"label", "v2":"sms"})</pre>					
In [7]: df_sms					
Out[7]:					
Autor Suis					
1 harn Ok lar. Joking will u oni					
📫 🖓 Type here to search 🛛 O 🛱 🏮 C 🥝 🐂 🖶 😋 🖪 📓 💭	^	🗈 EN	G 01:1 07-02	1 PM 2-2021	₽

Fig.7.3 dropping the unwanted columns

	> C 🕕 localh	ost8888/not	ebooks/OneDrive/Desktop/mini%	620project/code.ipynb	00	î∂ î≜	œ	
🗋 google	e 📵 BigBlueButton - V S							
	💭 jupyter	code Last (Checkpoint: an hour ago (autosaved)			e Lo	gout	
	File Edit	View Inser	t Cell Kernel Widgets	Help	Trusted	Pythor	13 O	
	B + × 2	· •	V Run 🔳 C 🕨 Code	✓				
		Droppi	ng the unwanted c	olumns Unnamed:2, Unnamed: 3 an	nd Unnamed	:4		
	In [6]:	df_sms = d	f_sms.drop(["Unnamed: 2", "Un	named: 3", "Unnamed: 4"], axis=1)				
		dt_sms = d	f_sms.rename(columns={"v1":"l	abe1", "v2":"sms"})				
	In [7]:	df_sms						
	In [7]: Out[7]:	df_sms						
	<pre>In [7]: Out[7]:</pre>	df_sms label		sms				
	In [7]: Out[7]:	df_sms label 0 ham	Go until jurong point, crazy. Available	sms				
	In [7]: Out[7]:	df_sms label 0 ham 1 ham	Go until jurong point, crazy. Available Ok Iar Joking wif u	sms only u onl				
	In [7]: Out[7]:	df_sms label 0 ham 1 ham 2 spam	Go until jurong point, crazy. Available Ok latJoking wif u Free entry in 2 a wkły comp to win FA Cup	sms only fina				
	In [7]: Out[7]:	df_sms label 0 ham 1 ham 2 spam 3 ham	Go until jurong point, crazy. Available Ok Iar Joking wif u Free entry in 2 a wkły comp to win FA Cup U dun say so early hor U c already ther	sms				
	In [7]: Out[7]:	df_sms label 0 ham 1 ham 2 spam 3 ham 4 ham	Gountil jurong point, crazy. Available OkiarJoking wif U Free entry in 2 a wkly comp to win FA Cup U dun ay so early hor U c already ther Nah I don't think he goes to usf, he lives	sms only 0 onl fina s ay				
	<pre>In [7]: Out[7]:</pre>	df_sms label 0 ham 1 ham 2 spam 3 ham 4 ham	Go until jurong point, crazy. Available Ok Iar Joking wif Free entry in 2 a wkły comp to win FA Cup U dun say so early hoc U c already ther Nah I don't think he goes to usť, he lived	sms only fina s aro 				
	<pre>In [7]: Out[7]:</pre>	df_sms	Go until jurong point, crazy. Available Ok lar. Joking wif Free entry in 2 a wkły comp to win FA Cup U dun say so early hor U c already ther Nah I don't think he goes to ust, he lives This is the 2nd time we have tried 2 cont	sms only u onl fina ssw aro aro aro				
	<pre>In [7]: Out[7]:</pre>	df_sms label 0 ham 1 ham 2 spam 3 ham 4 ham 5567 spam 5568 ham	Go until jurong point, crazy. Available Ok lar. Joking wift Free entry in 2 a wkly comp to win FA Cup U dun say so early hor U c already ther Nah I don't think he goes to ust, he lives This is the 2nd time we have tried 2 cont Will <u>L</u> b going to esplanade fr h	sms only u onl fina say aro act u oome?				
	<pre>In [7]: Out[7]:</pre>	df_sms label 0 ham 1 ham 2 spam 3 ham 4 ham 5567 spam 5568 ham 5569 ham	Go until jurong point, crazy. Available Ok lar Joking wif Free entry in 2 a wkiy comp to win FACuy U dun say so early hor U c already ther Nah I don't think he goes to usf, he lives This is the 2nd time we have tried 2 cont Will 1_b going to esplanade fi h PRy, * was in mood for that Soany oth	sms only ofma hinasy asu act u ome?				
	<pre>In [7]: Out[7]:</pre>	df_sms label 0 ham 1 ham 2 spam 3 ham 4 ham 5567 spam 5568 ham 5569 ham	Oo until jurong point, crazy. Available Ok lar Joking wift Free entry in 2 a wkiy comp to win FACup U dun say so early hor U c already ther Nah I don't think he goes to ust, he lives This is the 2nd time we have tried 2 cont Will <u>b</u> going to esplanade frh Phy, * was in mood for that. Soany off The guy did some bitching but I acted lik	sms only onl fina say say say actu ome? hers ex id				

Fig.7.4 viewing the data after removal of unwanted columns

🖇 OneDrive/Desktop/mini project/ 🗙	< 🧧 a	ode - Jupyter	Notebook	×	+											-	٥	×
ightarrow $ ightarrow$ $ ig$	ost:8888	/notebook	s/OneDrive	/Deskto	p/mini%20p	roject/c	ode.ipyi	nb					0.0	τô	£≜	G.		
J google 🔞 BigBlueButton - V S																		
💭 Jupyter	code	Last Checkpo	oint: an hour a	igo (auto	osaved)									0	Logo	ut		
File Edit	View	Insert C	ell Kerne	l Wic	igets Help								Trusted		Python 3	0		
B + >< 4	в.	↑ ¥ H	Run 🔳 C	•	Code	~ 📖												
	5568	ham	Will Ì_ b go	ng to espl	lanade fr home?													
	5569	ham Pity,	* was in moor	for that. S	Soany other s													
	5570	ham The g	uy did some b	itching bu	t I acted like i'd													
	5571	ham		Roff. Its	true to its name													
	5572 roi	ws × 2 colurr	ns															
In [8]:	print(]	Len(df_sms))															
	5572																	
In [9]:	df_sms.	.describe()															
Out[9]:		1-1-1																
	count	6672	5672															
	unique	2	5169															
	top	ham Sorry,	I'll call later															
	freq	4825	30															
In [10]:	df_sms	['length']	= df_sms	['sms']	.apply(len)													
In [11]:	df_sms.	.head()																
			0				-	д	-	 	~	-			-	01:1	1 PM	_

Fig.7.5 viewing the unique data

🔿 OneDrive/Desktop/mini project/ 🗴 🧧 code - Jupyter Notebook 🛛 🗴 +		-	٥	×
 c → O O localhost8888/notebooks/OneDrive/Desktop/mini%20project/codejpynb google O Bookston-VS. 	₹∕≡	<u>ه</u> (
💭 JUPYter code Last Checkpoint: an hour ago (autosaved)	Logou			
File Edit View Insert Cell Kamel Widgets Help Trustes F [2] + 5c 4c) 100 + 5c 4c) 100 + 5c 5c	^o ython 3	0		
freq 4825 30				*
<pre>In [10]: df_sms['length'] = df_sms['sms'].apply(len)</pre>				
In [11]: (dr_sms.head() Out[11]: Iabel sms length				h
0 ham Ge until jurong point, crzy, Available only 111 1 ham Ok laz, Joking wif u onl 29				
2 spam Free entry in 2 a wHoy comp to win FA Cup fina 155 3 ham U dun say so early hor Uc already then say 49				
4 ham Nah I don't hink he goes to usf, he lives aro 61				
<pre>In [12]: import matple1lb.yplot as plt import saborn as sns ?matplotllb inline df_sms['length'].plot(kins=50, kind='hist')</pre>				
Out[12]: (matplotlib.axessubplots.AxesSubplot at 8x26a8d1b4348)				
1400 - 12200 -				
📲 🔎 Type here to search 🛛 O 🛱 🚺 💽 🕑 🥥 🐂 💼 📚 🛃 💭 🖏 🔨	ED ENG	01:12 F	PM E	2

Fig .7.6 importing libraries to plot the graph

🔿 OneDrive/Desktop/mini project/ 🗙 🧧 code - Jupyter Notebook 🛛 🗙 🕂				-	٥	×
$\leftarrow \rightarrow$ C (localhost8888/notebooks/OneDrive/Desktop/mini%20project/code.ipynb		ίò	£≡	œ (
🗅 google 🔞 BigBlueButton - V S						
CJUPYTEr code Last Checkpoint: an hour ago (autosaved)		2	Logou	ıt		
File Edit View Insert Cell Kernel Widgets Help	Trusted	P	ython 3	0		
<pre>In [12]: import matplotlb.pyplot as plt import seaborn as son 'matplotlb inline df_sms['length'].plot(bins=50, kind='hist') Out[12]: (matplotlb.axessubplots.AxesSubplot at 0x26a0d1b4340)</pre>						
						ĺ
<pre>In [13]: df_sms.hist(column='length', by='label', bins=50,figsize=(10,4))</pre>						
<pre>Out[13]: array([cmatplotlib.axessubplots.AxesSubplot object at 0x0000026A805038C8>,</pre>						
				01:12	PM	*
🖷 🖓 Type here to search 🕐 🖳 💟 💟 🧭 📃 🔟 🔀 💭 🛄		^	E ENC	07-02-	2021	~

Fig.7.7 plotting graph for the given data

OneDrive/Desktop/mini project/	< 🧧 code - Jupyter Notebook 🛛 🗙 🕂				- 0	×
\leftarrow \rightarrow C (i) locally	ost8888/notebooks/OneDrive/Desktop/mini%20project/code.ipynb		ŵ	£_= (è 😩	
🗅 google 🛛 🗿 BigBlueButton - V S						
💭 Jupyter	code Last Checkpoint: an hour ago (autosaved)		2	Logout		
File Edit	View Insert Cell Kernel Widgets Help	Trusted	P	ython 3 C	0	
B + × 4	I 🚯 ♠ Ψ M Run 🔳 C 🏕 Code 👻 🚥					
In [14]:	<pre>df_sms.loc[:,'label'] = df_sms.label.map({'ham':0, 'spam':1}) print(df_sms.shape) df_sms.head() (5572 3)</pre>					*
Out[14]:	label sms length					
	0 Go until jurong point, crazy. Available only 111					
	2 1 Free entry in 2 a widy comp to win FA Cup fina 155					
	3 0 U dun say so early hor U c already then say 49					
	4 0 Nah I don't think he goes to usf, he lives aro 61					
	Implementation of Bag of Words Approach					
	Step 1: Convert all strings to their lower case form.					
In [15]:	<pre>documents = ['Hello, how are you!',</pre>					
P Type here to search	o 🖽 💿 🕞 🤗 🀂 📾 🐟 🖪 🗾 🗂 🚥		^	ENG	01:12 PM	

Fig.7.8 converting all strings to their lower case form

🔿 OneDrive/Desktop/mini project/ 🗙 🧧 cade - Jupyter Notzbook 🛛 🗙 🕂				-	٥	×
$\leftrightarrow \rightarrow \mathbf{C}$ (1) localhost8888/notebooks/OneDrive/Desktop/mini%20project/code.jpynb		ŵ	£≡	Ē		
🗅 google 🚯 BigBlueButton - V S						
JUpyter code Last Checkpoint: an hour ago (autosaved)		2	Logou	ıt		
File Edit View Insert Cell Kernel Widgets Help	Trusted	F	ython 3	0		
4 U Pean i dumi umitkine gues su usi, ne mes aru O t						*
Implementation of Bag of Words Approach						
Step 1: Convert all strings to their lower case form.						
To [15]: decuments - ['Wallo, how one woul'						
'Win money, win from home.', 'Call me now.'.						
'Hello, Call hello you tomorrow?']						
<pre>In [16]: lower_case_documents = []</pre>						
print(lower_case_documents)						
['hello, how are you!', 'win money, win from home.', 'call me now.', 'hello, call hello you tomorrow?']						
Stan 2. Demoving all supervisions						
Step 2. Removing an punctuations						
<pre>In [17]: sans_punctuation_documents = [] import string</pre>						
for i in lower_case_documents:						
cane municipation documents annond(i translate(ste maketnane("" "" steining municipation)))		^		01:1	2 PM	





Fig.7.10 Tokenization

🔿 OneDrive/Desktop/mini project/ 🗴 🧧 code - Jupyter Notebook 🛛 🗴 🕂	-	٥	×
C O localhost8888/notebooks/OneDrive/Desktop/mini%20project/code.ipynb	œ		
🕒 google 🧕 Jagoby 🔞 👌			
🂭 JUPYTER code Last Checkpoint: an hour ago (autosaved)	ut		
File Edit View Insert Cell Kernel Wildgets Help Trusted Python 3	0		
Step 4: Count frequencies In [19]: frequency_list = [] import point from collections import Counter			*
<pre>In [20]: frequency_list = [Counter(d) for d in preprocessed_documents] pp^int.pp^int(frequency_list) [Counter({Holio':1, 'money': 1, 'from': 1, 'you': 1}), Counter({'win':2, 'money': 1, 'from': 1, 'home': 1}), Counter({'call::1, 'me':1, 'now': 1}, 'tomorrow': 1})]</pre>			ļ
Implementing Bag of Words in scikit-learn			
<pre>In [21]: from sklearn.feature_extraction.text import CountVectorizer count_vector = CountVectorizer()</pre>			
In [22]: count_vector			
<pre>Out[22]: CountVectorizer(analyzer='word', binary=False, decode_error='strict',</pre>			¥
📲 🖓 Type here to search 🛛 O 🛱 💽 C 🖉 📮 🗟 🔂 📓 🛣 💆 💶 🔨 🗈 El	IG 01:	12 PM 02-2021	\Box

Fig.7.11 counting frequencies & implementing bag of words in scikit-learn

 OneDrive/Desktop/mini project/ 	< 🧧 code - Jupyter Notebook x +				-	٥	×
\leftarrow \rightarrow C (i) locally	ost8888/notebooks/OneDrive/Desktop/mini%20project/code.ipynb	Ξ.	۲ò	£∕≡	Ē		
🗅 google 🛛 BigBlueButton - V S							
💭 Jupyter	code Last Checkpoint: an hour ago (autosaved)		6	Logo	ıt		
File Edit	View Insert Cell Kernel Widgets Help	Trusted		Python 3	0		
B + × 4	1 🖪 🛧 V M Run 🔳 C 🍽 Code 🗸 📼						
	Data preprocessing with CountVectorizer()						*
In [23]:	count_vector.fit(documents)						
Out[23]:	['are',						
	'call', 'from'.						
	'hello', 'home'.						
	'how', 'me'						
	mey, 'money',						
	tomorrow',						
	"vin', 'you']						
In [24]:	<pre>doc_array = count_vector.transform(documents).toarray()</pre>						
	doc_array						
Out[24]:	array([[1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1], [0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 2, 0],						
	[0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0], [0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 1]], dtype=int64)						
	frequency matrix = pd.DataFrame(doc array, columns = count vector.get feature names())						
In [25]:							
In [25]:	frequency_matrix						*

Fig.7.12 Data preprocessing with CountVectorizer()

→ C (google © BigBlueBu Ç JU	 localhos Button - V S 	st8	888														
google 💿 BigBlueBl	Button - V S			/not	ebool	ks/One	eDrive	e/Des	ktop/n	nini%	20pr	oject/code	.ipyr	nb		œ (
💭 ju																	
	upyter c	:00	de ı	.ast (Checkp	oint: ar	hour :	ago (a	autosav	ed)					ele Logout		
File	Edit Vi	iew		Inser	t (Cell	Kerne	a .	Widgets	: H	lelp				Trusted Python 3	0	
B +	• * 2	6		•	V H	Run	• C	: >	Code		~	223					
,	In [25]: fr	red	auen	c v r	matri	x = p(d.Dat	aFram	ne (doc	arra	γ. ε	olumns =	cour	nt v	ctor.get feature names())		
	fi	red	quen	c y_r	matri	x					,, -				····· ·B······ · ······· · ······ ())		
C	Out[25]:		are	call	from	hello	home	how	me n	noney	now	tomorrow	win	you			
	C	0	1	0	0	1	0	1	0	0	0	0	0	1			
	1	1	0	0	1	0	1	0	0	1	0	0	2	0			
	2	2	0	1	0	0	0	0	1	0	1	0	0	0			
		3	0		U	2	U	0	U	0	U		U				
1	In [26]: f	nor	n sk	lear	≏n.mo	del_s	elect	ion i	mport	trai	n_te	st_split					
I	In [40]: X	_tr	rain	, x	_test	, y_tr	rain,	y_te	st =	train	_tes	t_split(d d r	lf_sr lf_sr ando	ms[' ms[' om_s	ms'], abel'],test_size=0.20, ate=1)		
1	In [41]: X	_tr	rain														
c	Out[41]: an	rra xt	ay([rat	'Go 'Ok "Fre e)Ta 'We	unti lar. ee en &C's , 'Ca ll it e=obj	l jurd Jok try ir apply n you s not ect)	ong p king n 2 a 0845 talk like	oint, wif u wkly 28100 with you	craz i oni. comp 75ove me actua	у А to w r18's ', 'Н lly с	vail in F ", mph. alle	able only A Cup fin Go head, d someone	/ in Wal 1 big e a p	bug tkts g ba punt	s n great world la e buffet Cine there got amore wat', 21st May 2005. Text FA to 87121 to receive entry question(std ler.', . That woulda been worse.'],		

Fig.7.13 Splitting the data



Fig.7.14 training the data

🔿 OneDrive/Desktop/mini project/ 🗴 🧧 code - Jupyter Natebook 🛛 🗴 🕂				-	٥	×
\leftrightarrow \rightarrow C (i) localhost8888/notebooks/OneDrive/Desktop/mini%20project/code.ipynb		τõ	£∕≡	Ē		
🗅 google 🔞 BigBlueButton - V S						
CJUPYTET code Last Checkpoint: an hour ago (autosaved)		2	Logou	ut		
File Edit View Insert Cell Kernel Widgets Help	Trusted	F	ython 3	0		
				_		
In [44]: y_test						^
Out[4]: 1978 0 4028 0 4674 1 						
Fit the training data and then return the matrix In [46]: training_data = count_vector.fit_transform(X_train)						
■ P Type here to search O Ħ O O ■ C O ■		^	EN EN	G 01:1 07-02	3 PM 2-2021	\Box

Fig.7.15 fitting the trained data and return the matrix



Fig.7.16 implementation of naïve bayes theorem

<pre>C) localhost8888/hotebooks/OneDrive/Desktop/mini%20project/code.jpynb</pre>	🔘 OneDrive/Desktop/mini project/ 🗙 🤌 code - Jupy	ner Notebook × +	- 0	×
BigBuleButch-VS image: I	$\leftarrow \rightarrow \mathbf{C}$ (i) localhost8888/notebo	joks/OneDrive/Desktop/mini%20project/code.ipynb	u 🔉 🔄 🖷 🛢 ·	
<pre>Code Last Checkpoint: an hour ago (autosaved) File Edit View Inset Cell Kernel Widgets Help Trusted Python 3 O File Edit View Inset Cell Kernel Widgets Help Trusted Python 3 O File Edit View Inset Cell Kernel Widgets Help Trusted Python 3 O File Edit View Inset Cell Kernel Widgets Help Trusted Python 3 O File Edit View Inset Cell Kernel Widgets Help Trusted Python 3 O File Edit View Inset Cell Kernel Widgets Help Trusted Python 3 O File Edit View Inset Cell Kernel Widgets Help Trusted Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset Cell Kernel View Inset File Python 3 O File Edit View Inset File Edit View Inset File Python 3 O File Edit View Inset File Edit View Inset File Edit View Inset File Python 3 O File Edit View Inset File Edit View Inset File Python 3 O File Edit View Inset File Edit View Inset File Python 3 O File Edit View Inset File Edit View Inset File Edit View Inset File Python 3 O File Edit View Inset File Edit View Inset File Python 3 O File Edit View Inset File Edit View Inset File Edit View Inset File Python 3 O File Edit View Inset File Edit View Inset F</pre>	google 😈 BigBlueButton - V S			
File Edit View hset Cell Kernel Widgets Help Trusted Python 3 0 Im	C JUPyter code Last Cher	ckpoint: an hour ago (autosaved)	n Logout	
<pre></pre>	File Edit View Insert	Cell Kernel Widgets Help	Trusted Python 3 O	
<pre>In [53]: predictions = naive_bayes.predict(testing_data) In [54]: predictions Out[54]: array([0, 0, 0,, 0, 0, 0], dtype=int64) Evaluating our SMS Spam Detection Model In [55]: from sklarn.metrics import accuracy.score, precision_score, recall_score, fi_score print('Accuracy.score', ').format(creation_score', test, predictions))) print('Precision score'; ().format(recall_score(y_test, predictions))) print('Precision score'; ().format(recall_score(y_test, predictions))) print('Recall score'; ().format(fi_score(y_test, predictions))) Accuracy score: 0.904733363280906 Precision score: 0.90473363280906 Precision score: 0.90473632 Precision score: 0.90473363280906 Precision score: 0.90473363280906 Precision score: 0.90473363280906 Precision score: 0.9047363 Precision score: 0.90473363 Precision score: 0.9047363 Precision scor</pre>	E + x C E + V	HRun E C > Code > (atprid=1.0, Class_prid=nune, itc_prid=1.0e)		*
<pre>In [54]: predictions Out[54]: array([0, 0, 0,, 0, 0, 0], dtype=int04) Evaluating our SMS Spam Detection Model In [55]: from sklean.metrics import accuracy_score, precision_score, recall_score, fi_score print('Accuracy score: {}'.format(accuracy_score(y_test, predictions))) print('Precision score: {}'.format(recall_score(y_test, predictions))) print('Facilis score: {}'.format(fi_score(y_test, predictions))) print('Facilis core: {}'.format(fi_score(y_test, predictions))) print('Facilis core: {}'.format(fi_score(y_test, predictions))) Accuracy score: 0.90473306322806906 Precision score: 0.904232080590572463 Recall score: 0.935511708561151 F1 score: 0.9386281580447652</pre>	In [53]: predictions =	naive_bayes.predict(testing_data)		
<pre>Out[54]: array([0, 0, 0,, 0, 0, 0], dtype=int64) Evaluating our SMS Spam Detection Model In [55]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score print('Accuracy score: {}'.format(accuracy_score(y_test, predictions))) print('Precision score: {}'.format(recall_score(y_test, predictions))) print('Facuracy score: {}'.format(f1_score(y_test, predictions))) print('Facuracy score: {}'.format(f1_score(y_test, predictions))) Accuracy score: 0.984733632286996 Precision score: 0.932531798561151 F1 score: 0.932631798561151 F1 score: 0.9386281588447652</pre>	In [54]: predictions			
Evaluating our SMS Spam Detection Model In [55]: from sklearn.metrics import accuracy_score, precision_score, recall_score, fi_score print('Accuracy score: {)'.format(accuracy_score(y_test, predictions))) print('Precision score: {)'.format(precision_score(y_test, predictions))) print('Facilis score: {)'.format(recall_score(y_test, predictions))) print('Fi score: {)'.format(fi_score(y_test, predictions))) Accuracy score: 0.9847533632286996 Precision score: 9.942208095672463 Recall score: 0.935551798561151 Fi score: 0.935551798561151	Out[54]: array([0, 0,	0,, 0, 0, 0], dtype=int64)		
<pre>print('Precision score: {}'.format(precision_score(y_test, predictions))) print('Recisil score: {}'.format(recall score(y_test, predictions))) print('Fi score: {}'.format(fi_score(y_test, predictions))) Accuracy score: 0.94733632286996 Precision score: 0.9427836895272463 Recall score: 0.935551798561151 F1 score: 0.9386281588447652</pre>	Evaluatii In [55]: from sklearn. print('Accura	ng our SMS Spam Detection Model metrics import accuracy_score, precision_score, recall_score, fi_score ey score: ()'.format(accuracy_score(y_test, predictions)))		
Accuracy score: 0.9047533632286996 Precision score: 0.9420280855072463 Recall score: 0.935251790561151 Fi score: 0.9306281580447652	print('Precis: print('Recall print('F1 sco	<pre>lon score: {}'.tormat(precision_score(y_test, predictions))) score: {}'.format(recall_score(y_test, predictions))) re: {}'.format(f1_score(y_test, predictions)))</pre>		
	Accuracy scor Preision sco Recall score: Fi score: 0.9	E: 0.9847533632286996 re: 0.942028965872463 0.935251795651151 386281588447652		
	-			
🖬 🔎 Type here to search 🕐 🛱 🚱 💽 🖉 🧱 😭 🔽 💆 🖬 🗖 🗤 🗖 🖓				Ţ

Fig.7.17 Evaluating the model

8.BIBILOGRAPHY

https://thecleverprogrammer.com/2020/06/12/sms-spam-detection-withmachine-learning/

http://www.ijstr.org/final-print/feb2020/Spam-Detection-In-Sms-Using-Machine-Learning-Through-Text-Mining