SMS Spam Detection

Ambika Methre and Kavali Veena

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	160618733125
Kavali Veena	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	MS. Nadia Anjum
Head, Department of CSE	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 providing us the moral support and encouragement.

CONTENTS OF FIGURES

CONTENT

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

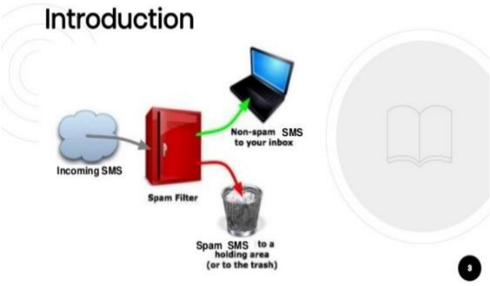

Fig-1.1 .introduction

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.

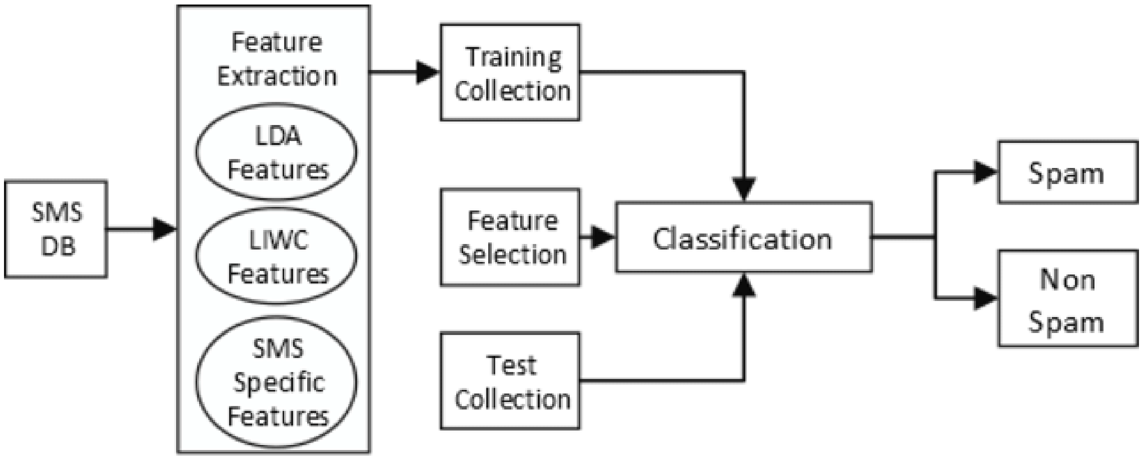

Fig-1.3 flow chart of sms spam detection

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

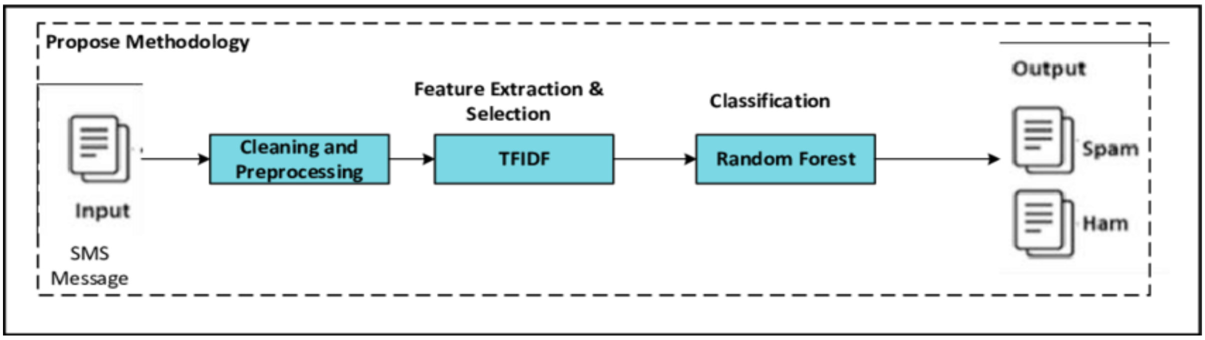

Fig.3 Methodology

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 2σ2)

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

Radial Basis Function	2.61	81.45	0.32
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 [dataset]() 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 ther
Ham	Ok lar... Joking wif u oni...
Spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 t 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
Spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun yo 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
Spam	WINNER!! As a valued network customer you have been selected to receivea å£900 prize 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 c
Spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. C info
Spam	URGENT! You have won a 1 week FREE membership in our å£100,000 Prize Jackpot! T 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!!
Spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message o xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
Ham	Oh k...i'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!

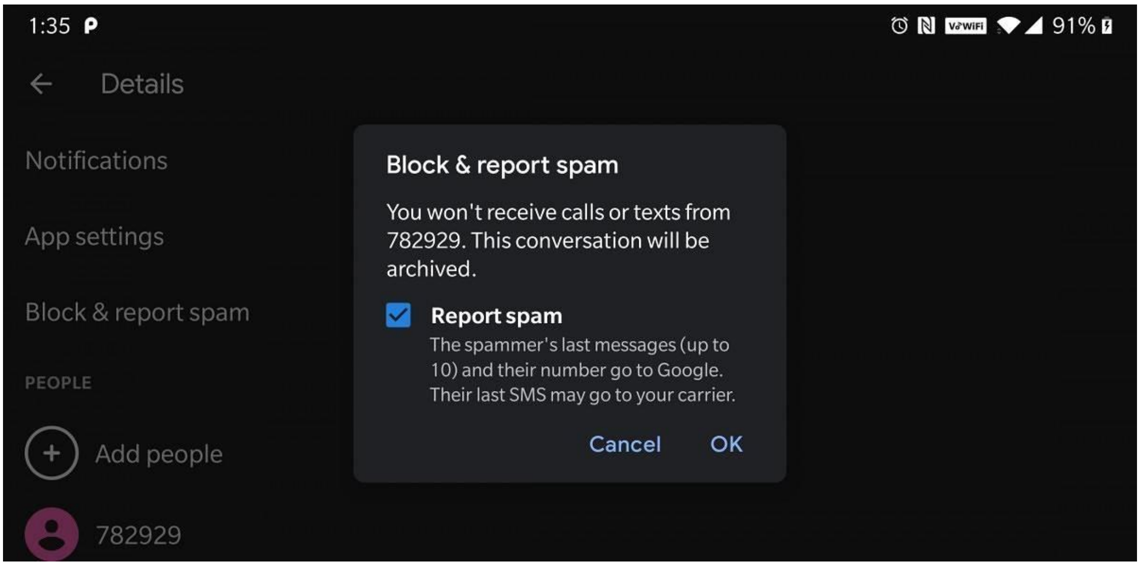

Fig-4 Reporting the spam

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 Decision tree	92.17	0.51	98.59

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.

.

.

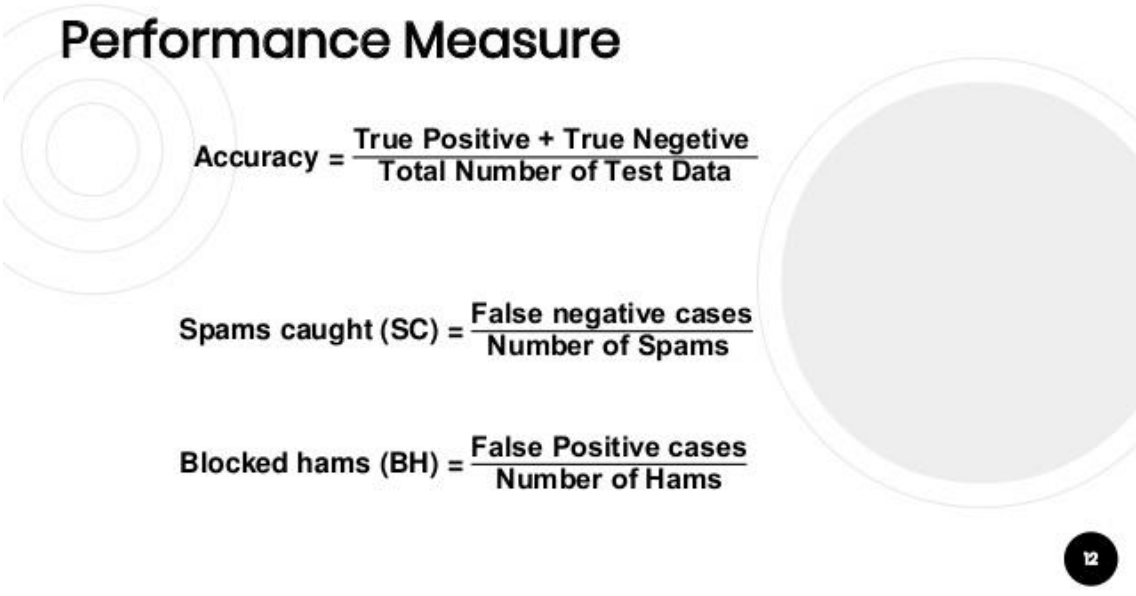

Fig.5 Measure of Accuracy ,SC,BH

6.TESTING

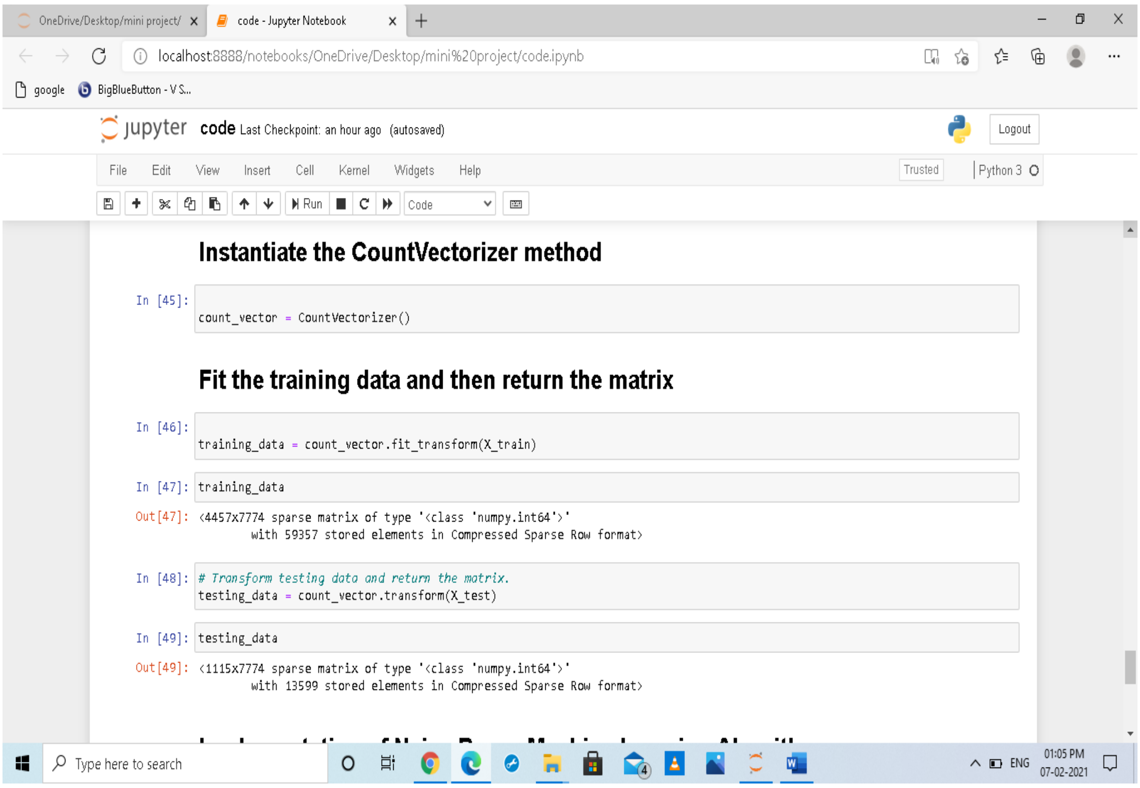

Fig.6.1 Testing the data

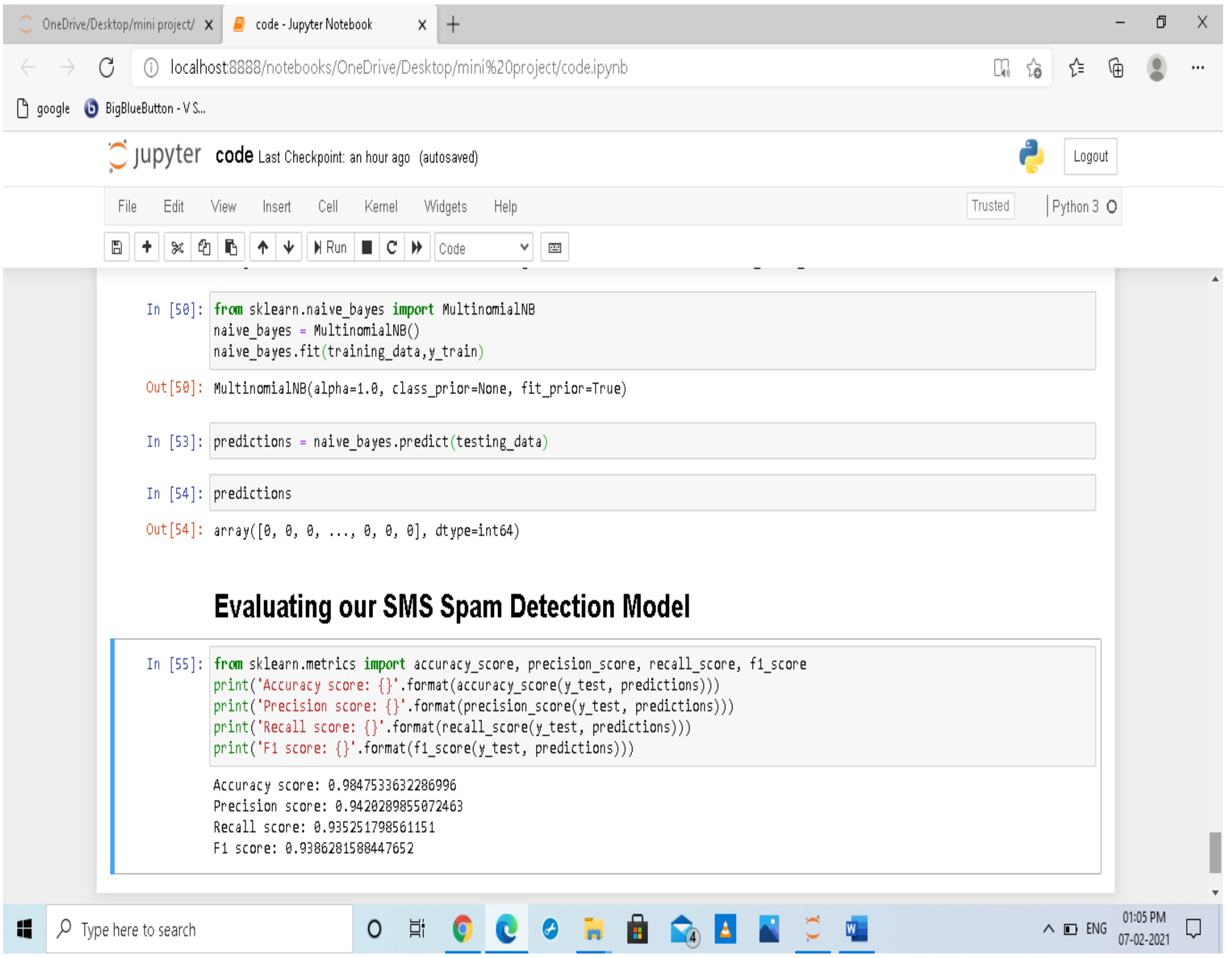

Fig.6.2 Evaluating the model

7. OUTPUT SCREENS

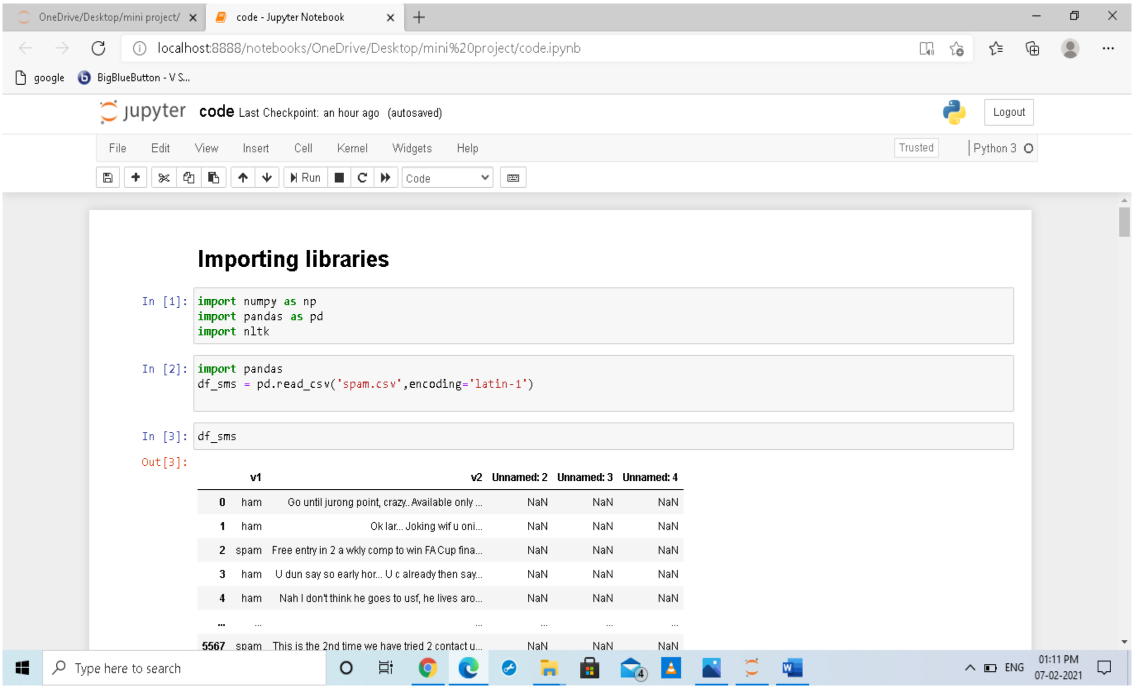

Fig.7.1 importing libraries

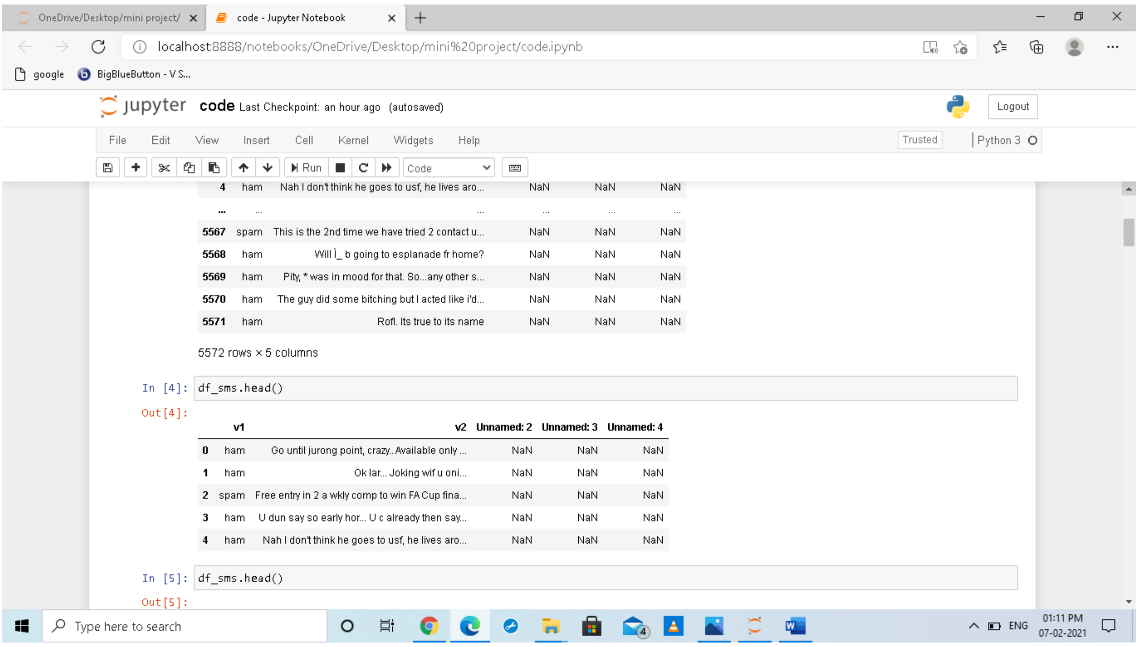

Fig.7.2 viewing the data

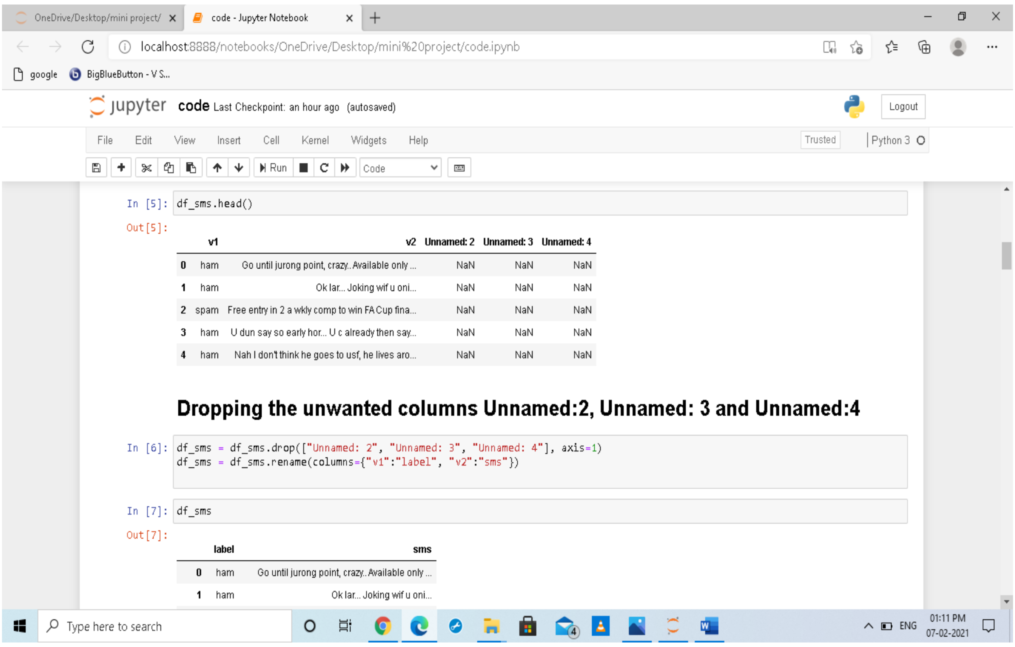

Fig.7.3 dropping the unwanted columns

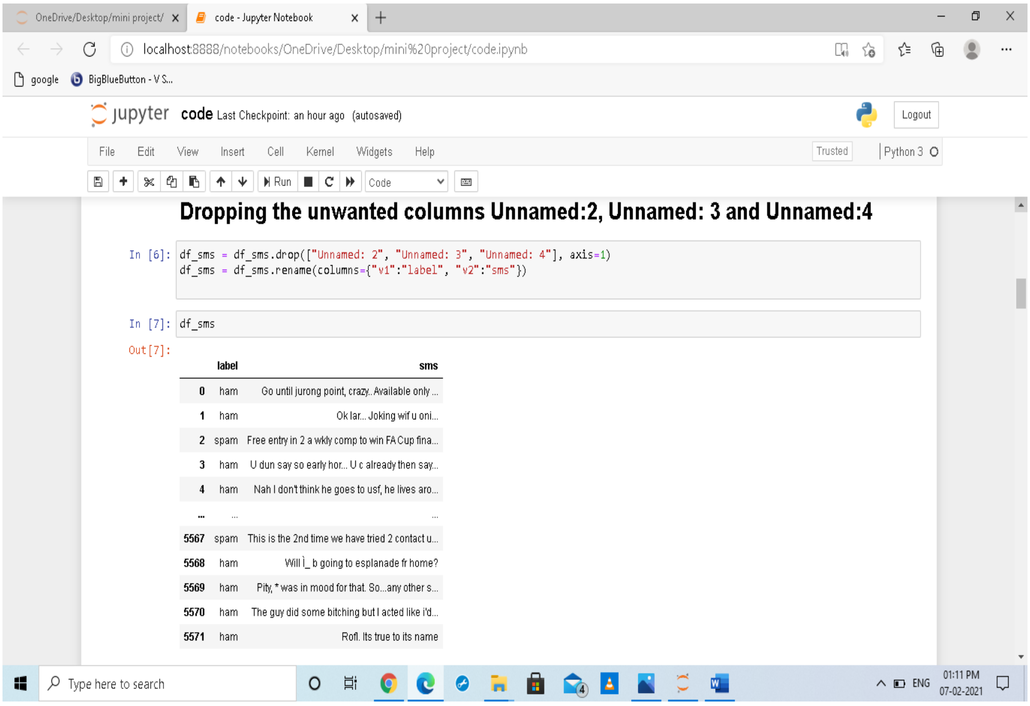

Fig.7.4 viewing the data after removal of unwanted columns

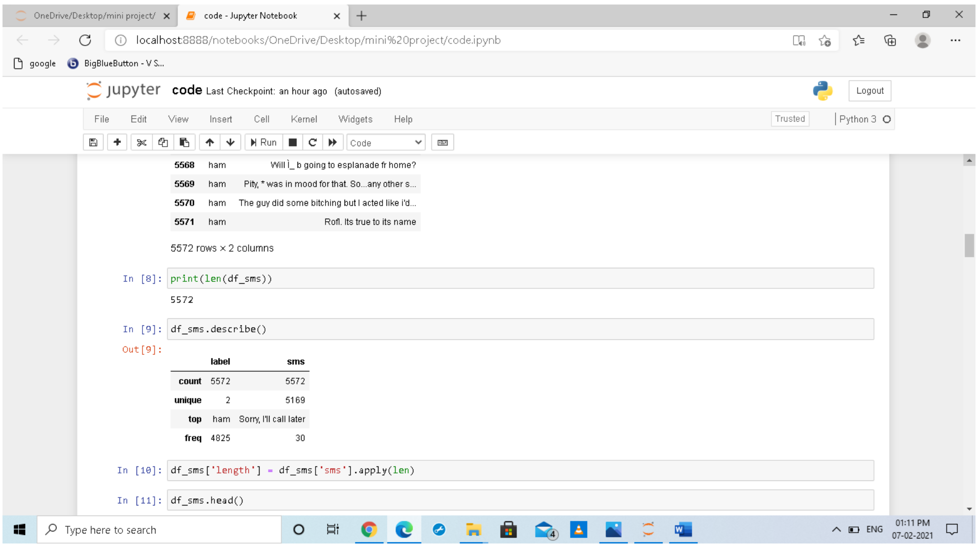

Fig.7.5 viewing the unique data

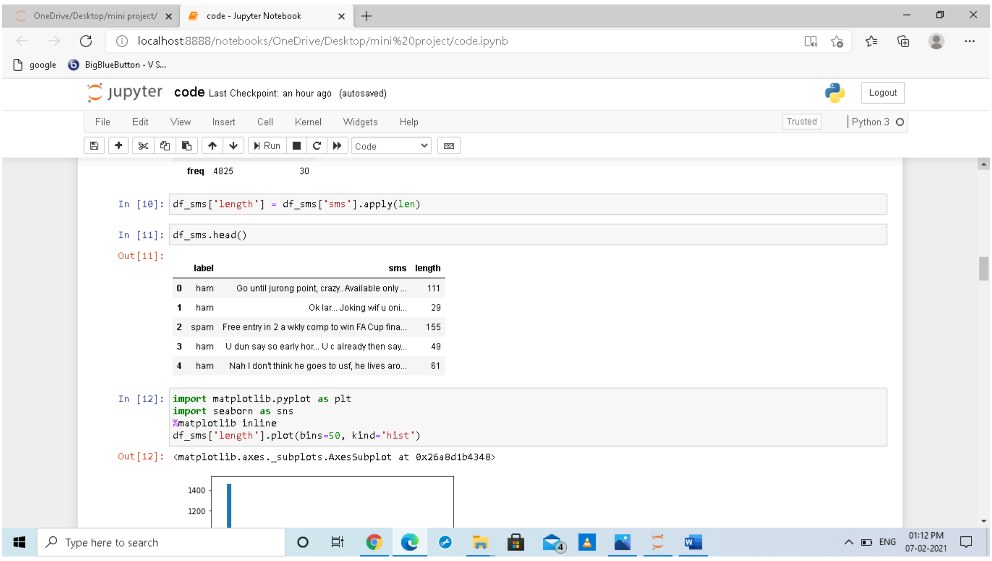

Fig .7.6 importing libraries to plot the graph

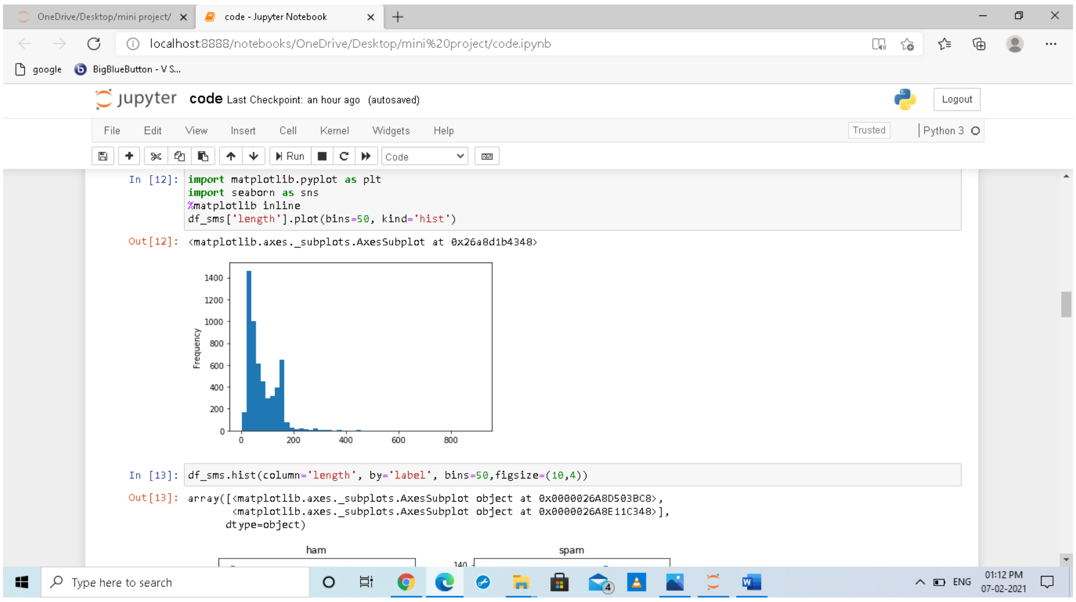

Fig.7.7 plotting graph for the given data

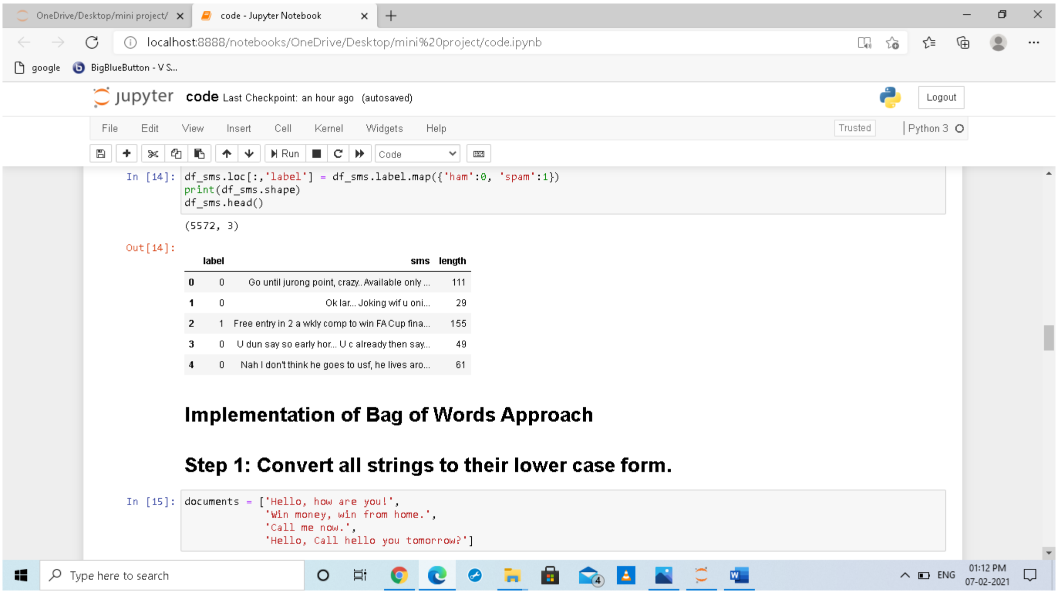

Fig.7.8 converting all strings to their lower case form

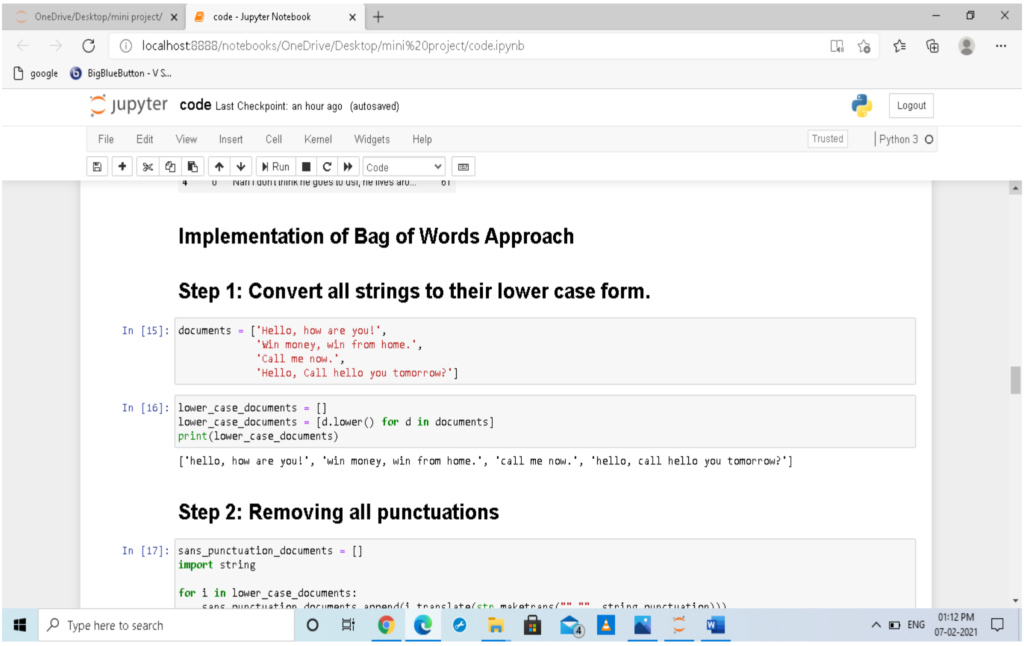

Fig.7.9 Removing all punctuations

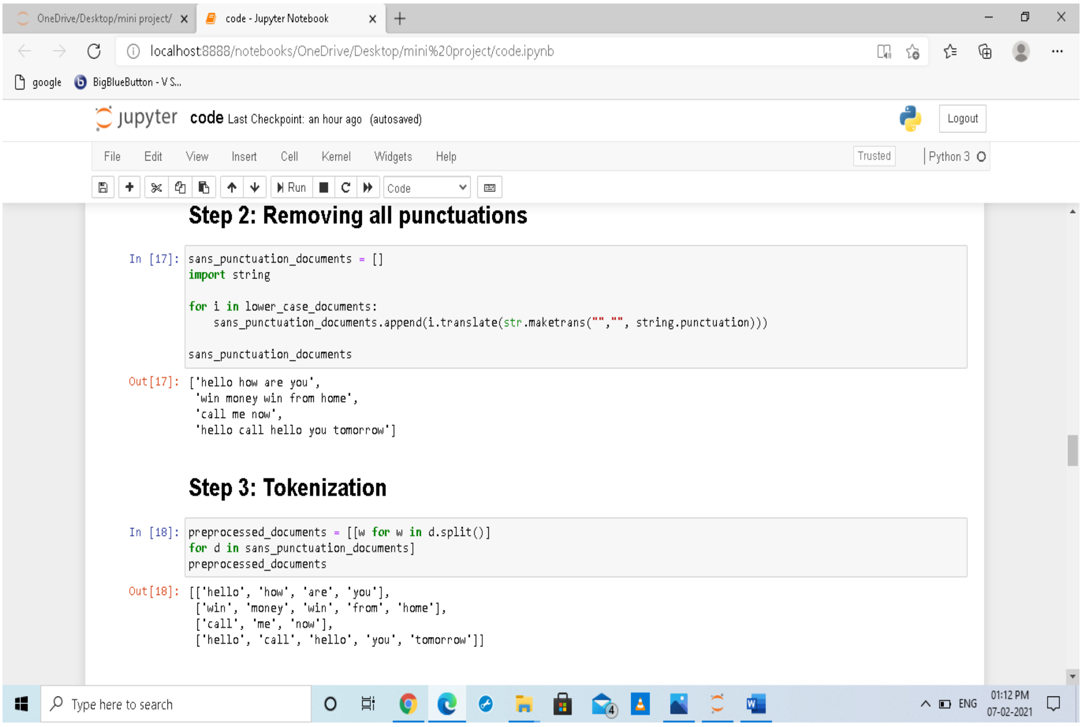

Fig.7.10 Tokenization

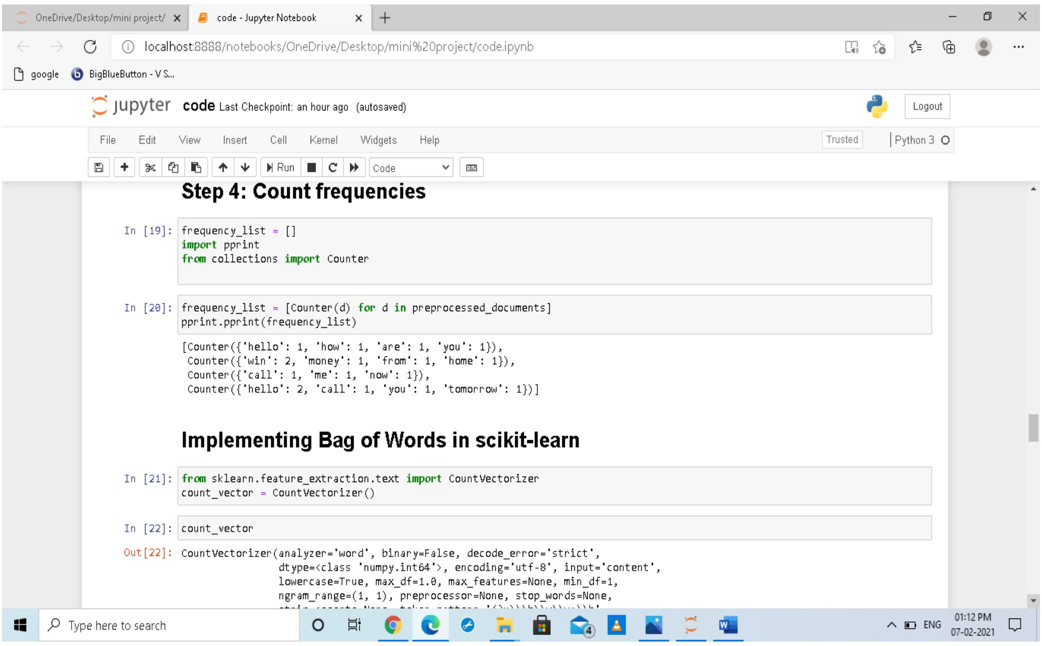

Fig.7.11 counting frequencies & implementing bag of words in scikit-learn

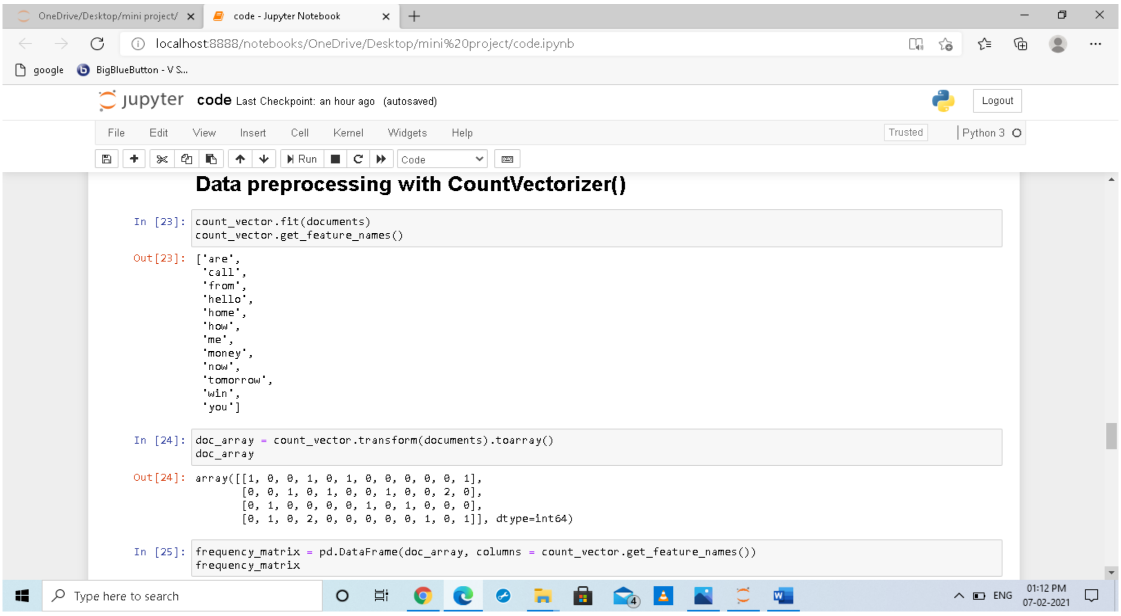

Fig.7.12 Data preprocessing with CountVectorizer()

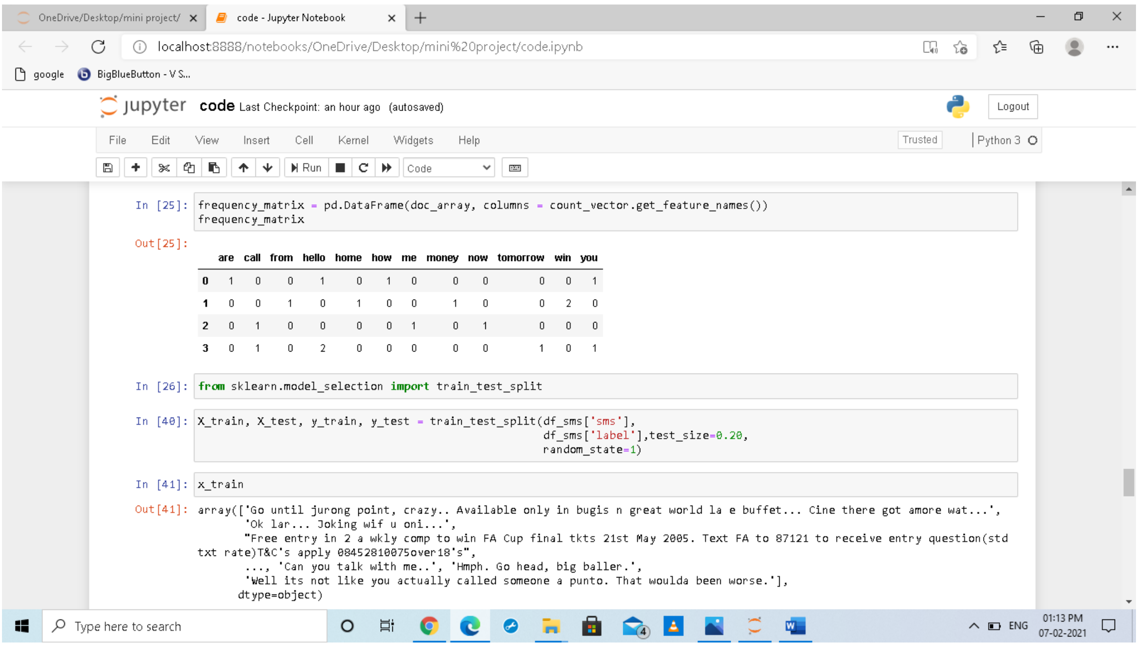

Fig.7.13 Splitting the data

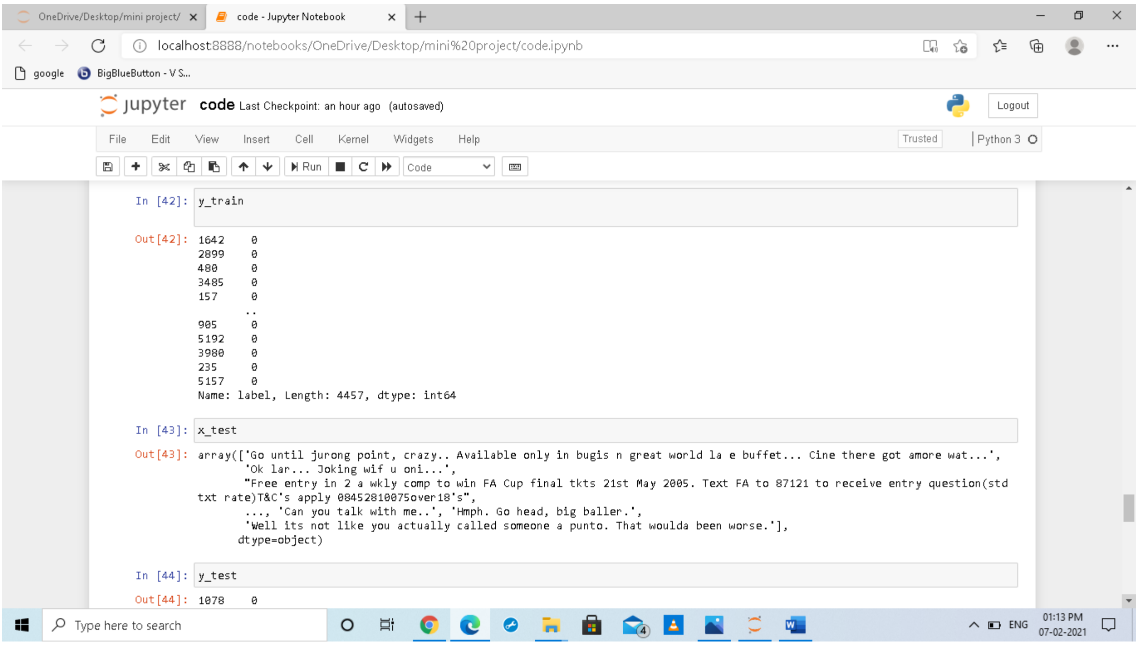

Fig.7.14 training the data

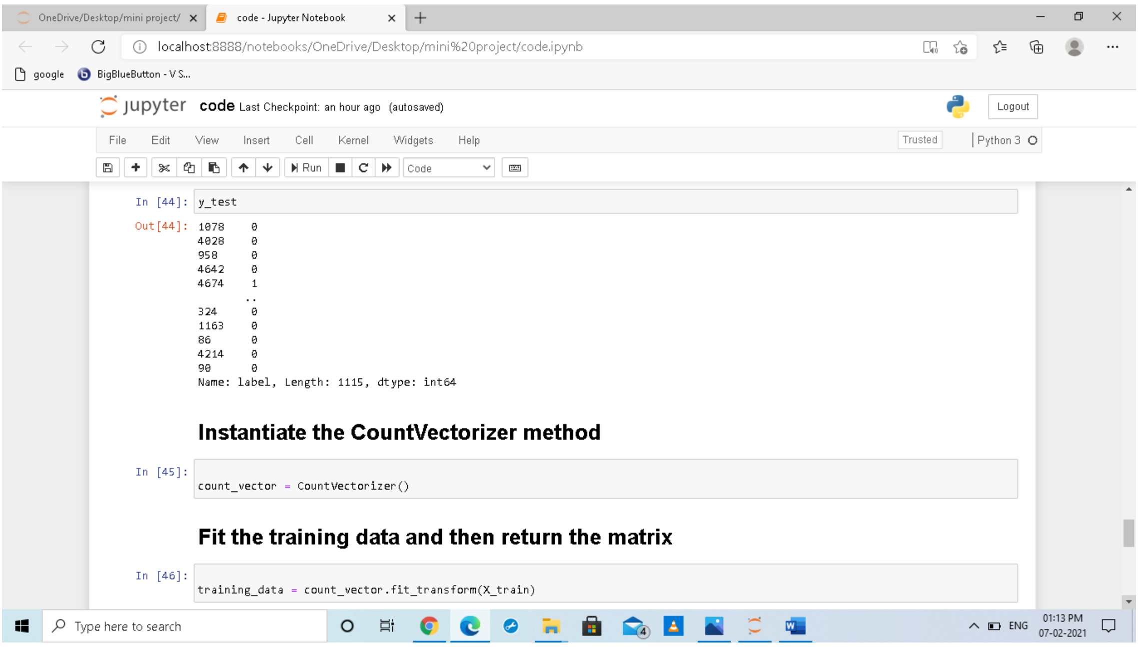

Fig.7.15 fitting the trained data and return the matrix

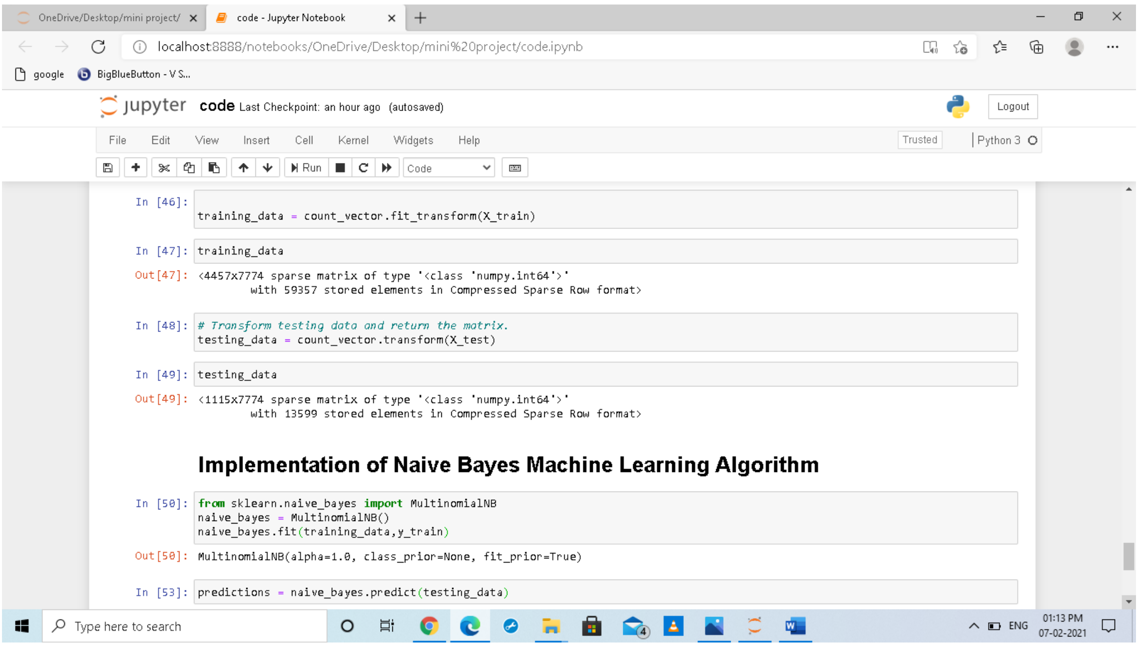

Fig.7.16 implementation of naïve bayes theorem

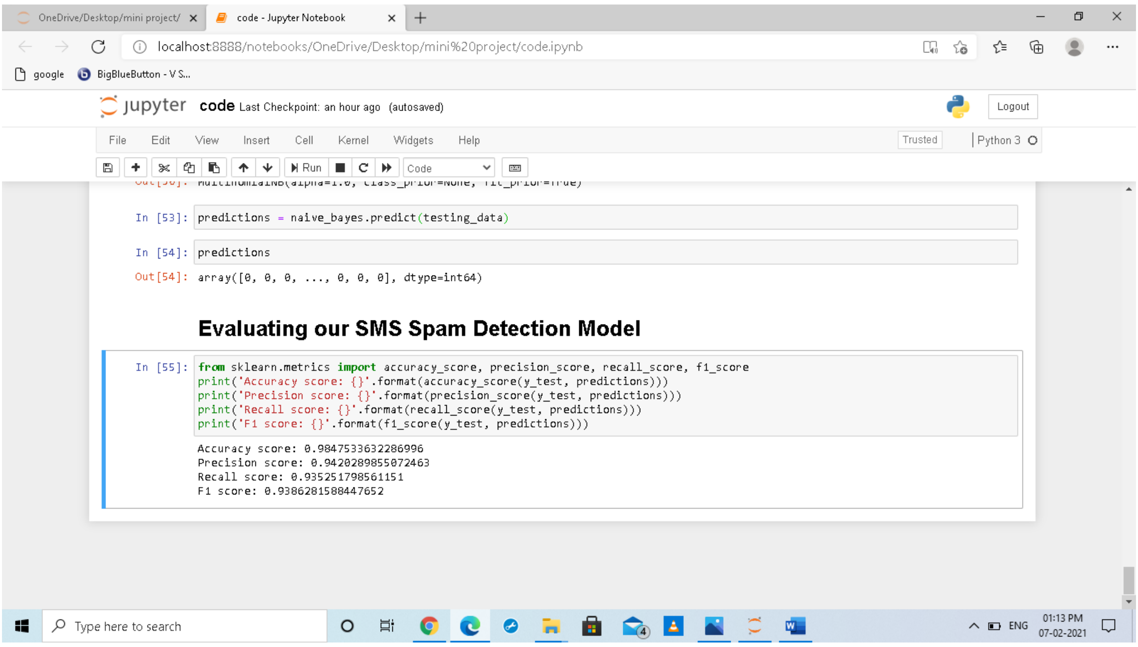

Fig.7.17 Evaluating the model

8.BIBILOGRAPHY

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

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