Thakur Educational Trust's (Regd.)

THAKUR COLLEGE OF SCIENCE & COMMERCE

AUTONOMOUS COLLEGE AFFILIATED TO UNIVERSITY OF MUMBAI

NAAC Accredited with Grade 'A' (3rd Cycle) & ISO 9001: 2015 Certified

Best College Award by University of Mumbai for the Year 2018-2019

A PROJECT REPORT ON

SPAM E-MAIL DETECTION USING NAÏVE BAYES ALGORITHM

A PROJECT SUBMITTED TO

THE UNIVERSITY OF MUMBAI FOR PARTIAL COMPLETION OF THE DEGREE OF MASTERS OF SCIENCE IN COMPUTER SCIENCE

UNDER THE FACULTY OF SCIENCE

By

Mohini Bharambe

UNDER THE GUIDENCE OF

Dr. Girish Tere

Thakur Degree College of Science and Commerce Kandivali (East), Mumbai Maharashtra 400101.

Thakur Educational Trust's (Regd.)

THAKUR COLLEGE OF SCIENCE & COMMERCE

AUTONOMOUS COLLEGE AFFILIATED TO UNIVERSITY OF MUMBAI

NAAC Accredited with Grade 'A' (3rd Cycle) & ISO 9001: 2015 Certified

Best College Award by University of Mumbai for the Year 2018-2019

CELEBRATING 25 YEARS OF GLORY

Date: 23/03/2023

DECLARATION BY LEARNER

I the undersigned **Ms. Mohini Bharambe** hereby, declare that the work embodied in this project work titled **"Spam E-mail Detection Using Naïve Bayes Algorithm"** forms my own contribution to the research wok carried out under the guidance of **Dr. Girish Tere** is a result of my own research work and has not been previously submitted to any other University for any other Degree/Diploma to this or any other University.

Wherever reference has been made to previous works of others, it has been clearly indicated as such and included in the bibliography.

I, hereby further declare that all information of his document has been obtained and presented in accordance with academic rules and ethical conduct.

Project Guide **Head of Department**

Thakur Educational Trust's (Regd.)

THAKUR COLLEGE OF SCIENCE & COMMERCE

AUTONOMOUS COLLEGE AFFILIATED TO UNIVERSITY OF MUMBAI

NAAC Accredited with Grade 'A' (3rd Cycle) & ISO 9001: 2015 Certified

Best College Award by University of Mumbai for the Year 2018-2019

CELEBRATING 25YEARS OF GLORY

Date: 23/03/2023

COMPUTER SCIENCE DEPARTMENT

(2022-2023)

Certificate of Approval

This is to certify that the project work entitled **"Spam E-Mail Detection Using Naïve Bayes Algorithm"** is prepared by **Mohini Bharambe** a student of **"Second Year Masters of Science (Computer Science)"** course of University of Mumbai, which is conducted by our college.

This is the original study work and important sources used have been duly acknowledged in the report. The report is submitted in partial fulfilment of B.Sc. (Computer Science) course as per rules of University of Mumbai.

Project Guide

Head of Department

External Examiner

Index:

ACKNOWLEDGEMENT

Achievement is finding out what you would be doing rather than what you have to do. It is not until you undertake such a project that you realize how much effort and hard work it really is, what are your capabilities and how well you can present yourself or other things. It gives me immense pleasure to present this report towards the fulfillment of my project.

It has been rightly said that we are built on the shoulder of others. For everything I have achieved, the credit goes to all those who had helped me to complete this project successfully.

I take this opportunity to express my profound gratitude to management of Thakur Degree College of Science & Commerce for giving me this opportunity to accomplish this project work.

I am very much thankful to **Mrs. C. T. Chakraborty** - Principal of Thakur College for their kind co-operation in the completion of my project.

A special vote of thanks to our HOD **Mr. Ashish Trivedi** and to our project guide **Ms. Siddhi Birwadar**. Thanks to all our teachers for helping and guiding me throughout my project.

Finally, I would like to thank all my friends & entire Computer Science department who directly or indirectly helped me in completion of this project & to my family without whose support, motivation & encouragement this would not have been possible.

(**Mohini Bharambe)**

Preliminary Investigation

Organizational Overview

The **Thakur College of Science and Commerce (TCSC)** is a college in Kandivali in Mumbai of Maharashtra, India running by Thakur Educational Trust

Thakur College was started in 1992 to serve the needs of students passing SSC examination from the schools around Kandivali area and Thakur Vidhya Mandir which has already established itself as one of the schools in the area. It offers courses at primarily the higher secondary and under-graduate levels. The courses at the undergraduate and post-graduate level are offered in affiliation with Mumbai University, Mumbai. An ISO 9001:2008 College with **A** grade as assessed by the National Assessment and Accreditation Council NAAC.

Name:	Thakur College of Science and Commerce
Founded:	1997
Address:	Thakur College of Science and Commerce, Thakur Village Kandivali(E), Mumbai – 400 001

ABSTRACT

Spam e-mail is still a concern on the Internet. Multiple copies of the same message, business advertisements, or other irrelevant posts like pornographic material may be found in spam emails. Different filtering methods, including Random Forest, Naive Bayesian, Support Vector Machine (SVM), and Neutral Network, have been employed in earlier research to find these emails. In this study, I evaluated the effectiveness of the Naive Bayes algorithm for e-mail spam filtering on several datasets.

Based on the dataset’s accuracy, recall, precision, and F-measure, performance is assessed. For the evaluation of the Naive Bayes algorithm for email spam filtering on dataset in my research, I utilize Jupiter Note Book. The outcome demonstrates that the email type and the dataset's instance count.

INTRODUCTION

Nowadays, e-mail provides many ways to send millions of advertisements at no cost to sender. As a result, many unsolicited bulk e-mail, also known as spam e-mail spread widely and become serious threat to not only the Internet but also to society. For example, when user received large amount of e-mail spam, the chance of the user forgot to read a non-spam message increase. As a result, many e-mail readers have to spend their time removing unwanted messages. E-mail spam also may cost money to users with dial-up connections, waste bandwidth, and may expose minors to unsuitable content.

Over the past many years, many approaches have been provided to block e-mail spam. For filtering, some email spam are not being labelled as spam because the e-mail filtering does not detect that email as spam. Some existing problems are regarding accuracy for email spam filtering that might introduce some error. Several machine learning algorithms have been used in spam e-mail filtering, but Naïve Bayes algorithm is particularly popular in commercial and open-source spam filters. This is because of its simplicity, which make them easy to implement and just need short training time or fast evaluation to filter email spam. The filter requires training that can be provided by a previous set of spam and non-spam messages. It keeps track of each word that occurs only in spam, in non-spam messages, and in both. Naïve Bayes can be used in different datasets where each of them has different features and attribute.

The research objectives are: (i) to implement the Naïve Bayes algorithm for e-mail spam filtering on dataset, (ii) to evaluate the performance of Naïve Bayes algorithm for e-mail spam filtering on the chosen dataset. The rest of the paper is organized as follows: Section II describes the related work on Naïve Bayes algorithm for e-mail spam filtering. Section III presents the methodology process of e-mail spam using JUPYTER NOTEBOOK. Section IV presents the experimental setup. Section V shows the result and analysis on dataset. Finally, Section VI concludes the work and highlights the direction for future research

Description of the System:

Origin of the idea

The growing problem of unsolicited bulk e-mail, also known as "spare", has generated a need for reliable anti-spam e-mail filters. Filters of this type have so far been based mostly on manually constructed keyword patterns. An alternative approach has recently been proposed, whereby a Naïve Bayesian classifier is trained automatically to detect spam
messages. We test this approach on a large collection of personal e-mail messages, which we make publicly available in "encrypted" form contributing towards standard
benchmarks. We introduce appropriate cost-sensitive measures, investigating at the same time the effect of attribute set size, training-corpus size, lemmatization, and stop lists,
issues that have not been explored in previous experiments. Finally, the Naive Bayesian filter is compared, in terms of performance, to a filter that uses keyword patterns, and which is part of a widely used e-mail reader.

Proposed System

1. A model based on NaiveBayes MultinomialNB function that predicts whether the email is spam or not on the basis of the count vector.

LITERATURE REVIEW

I. **Aditya Gupta "Spam Filter using Naïve Bayesian Technique "International Journal of Computational Engineering Research (IJCER), vol. 08, no. 06, 2018.[1]**

   In this abstract, The Naive Bayesian classifier has been suggested as an effective method to construct automatically anti-spam filters with superior performance. This Method achieves 95.56% accuracy and 93.91% precision spam filtering for the considered dataset, outperforming the keyword-based filter of a widely used e-mail reader.

II. **Spam Message Classification Based on the Naïve Bayes Classification Algorithm Bin Ning, Wu Junwei, Hu Feng, Feb 2019.[2]**

   In this research paper, By further comparing the classification performance against the support vector machine and random forest algorithms, the naïve Bayes algorithm based on multi-two-classification is shown to be the best.

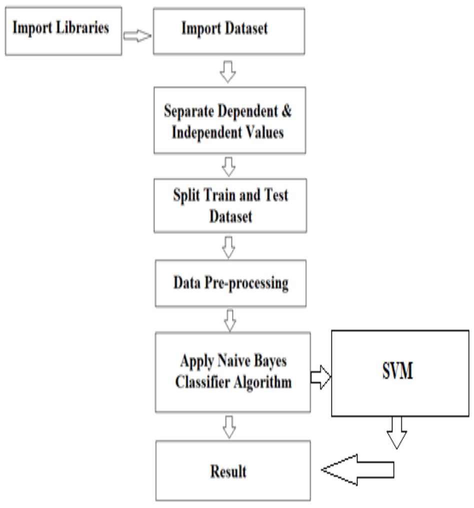
Objectives

- **General Objective -**
  To implement Naïve Bayes Classifier that classifies the no of spam mails
- **Specific Objective -**
  To develop web interface platform anti spam mails

- To create a simple but efficient model to detect spam mails in the system
- User friendly and less complex to understand
- A model that can provide better predictions.

METODOLOGY FLOW

This section describes the methodology that is used for the research. The methodology is used for the process of e-mail spam detection based on Naïve Bayes algorithm and Support Vector Machine.

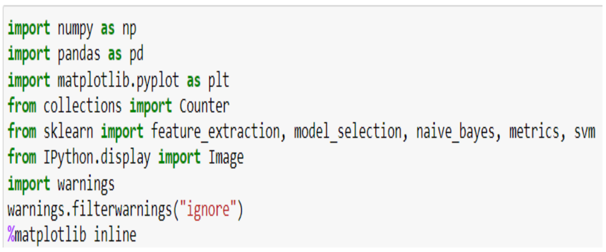


Import Libraries
Import Dataset
Separate Dependent & Independent Values
Split Train and Test Dataset
Data Pre-processing
Apply Naive Bayes Classifier Algorithm
SVM
Result

A: Imported Libraries

Here In this system we need to install the following libraries:

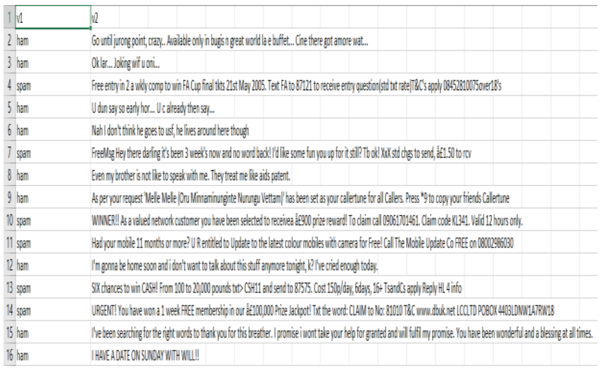
1. Pandas
2. NumPy
3. Matplotlib
4. Image
5. Counter
6. Feature_extraction, model_selection, naive_bayes, metrics, svm



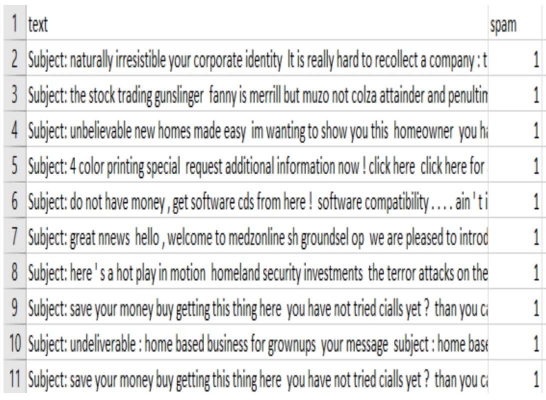
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from sklearn import feature_extraction, model_selection, naive_bayes, metrics, svm
from IPython.display import Image
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

B: Import Dataset

This project contains dataset that has two columns namely, Text and Spam where there are 5695 unique values.



	v1	v2
1	v1	v2
2	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
3	ham	Ok lar... Joking wif u oni...
4	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
5	ham	U dun say so early hor... U c already then say...
6	ham	Nah I don't think he goes to usf, he lives around here though
7	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, å£1.50 to rcv
8	ham	Even my brother is not like to speak with me. They treat me like aids patent.
9	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
10	spam	WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
11	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
12	ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
13	spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
14	spam	URGENT! You have won a 1 week FREE membership in our å£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
15	ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all times.
16	ham	I HAVE A DATE ON SUNDAY WITH WILL!!

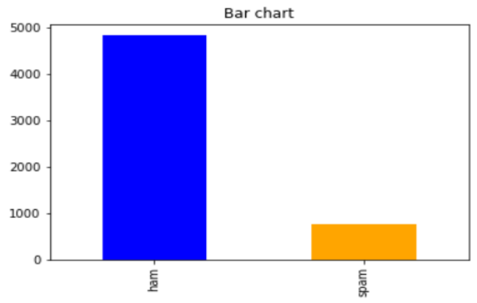


1	text	spam
2	Subject: naturally irresistible your corporate identity It is really hard to recollect a company : t	1
3	Subject: the stock trading gunslinger fanny is merrill but muzo not colza attainder and penultin	1
4	Subject: unbelievable new homes made easy im wanting to show you this homeowner you ha	1
5	Subject: 4 color printing special request additional information now ! click here click here for	1
6	Subject: do not have money , get software cds from here ! software compatibility . . . . ain ' t i	1
7	Subject: great nnews hello , welcome to medzonline sh groundsel op we are pleased to introd	1
8	Subject: here ' s a hot play in motion homeland security investments the terror attacks on the	1
9	Subject: save your money buy getting this thing here you have not tried cialls yet ? than you ca	1
10	Subject: undeliverable : home based business for grownups your message subject : home base	1
11	Subject: save your money buy getting this thing here you have not tried cialls yet ? than you ca	1

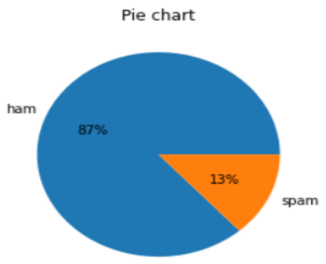
C: Bar Chart And Pie Chart

The Bar graph (bar chart) is a graph that shows the categorical information using rectangular bars. Bar graphs show the distinction between discrete categories. Pie charts are graphs with circular shapes that show the percentages of an entire group. So, this is the definition of a Pie Chart and bar diagram. In this we have use this to show the number of percentage of Spam and Ham Emails.

```
In [4]: count_Class=pd.value_counts(data["v1"], sort= True)
        count_Class.plot(kind= 'bar', color= ["blue", "orange"])
        plt.title('Bar chart')
        plt.show()
```



```
In [5]: count_Class.plot(kind = 'pie',  autopct='%1.0f%%')
        plt.title('Pie chart')
        plt.ylabel('')
        plt.show()
```

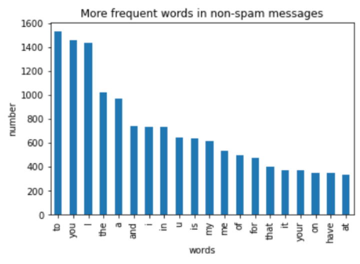


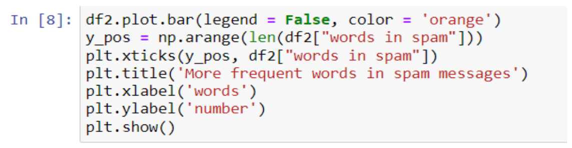
D: Bar Graph To Show Most Frequent Words Used

In this we have used Bar graph to count the most frequently used words in the spam messages and in the non- spam messages.

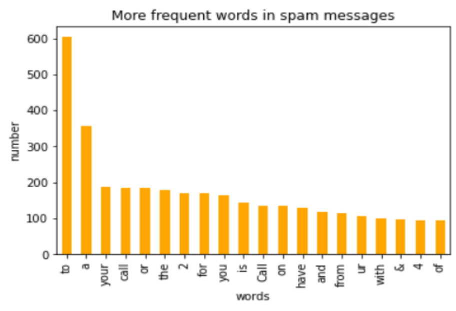
```
In [6]: count1 = Counter(" ".join(data[data['v1']=='ham']["v2"]).split()).most_common(20)
        df1 = pd.DataFrame.from_dict(count1)
        df1 = df1.rename(columns={0: "words in non-spam", 1 : "count"})
        count2 = Counter(" ".join(data[data['v1']=='spam']["v2"]).split()).most_common(20)
        df2 = pd.DataFrame.from_dict(count2)
        df2 = df2.rename(columns={0: "words in spam", 1 : "count_"})
```

```
In [7]: df1.plot.bar(legend = False)
        y_pos = np.arange(len(df1["words in non-spam"]))
        plt.xticks(y_pos, df1["words in non-spam"])
        plt.title('More frequent words in non-spam messages')
        plt.xlabel('words')
        plt.ylabel('number')
        plt.show()
```





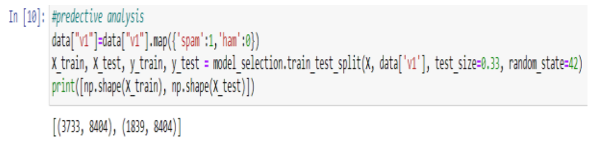
```
In [8]: df2.plot.bar(legend = False, color = 'orange')
        y_pos = np.arange(len(df2["words in spam"]))
        plt.xticks(y_pos, df2["words in spam"])
        plt.title('More frequent words in spam messages')
        plt.xlabel('words')
        plt.ylabel('number')
        plt.show()
```



E: Feature Extraction

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.

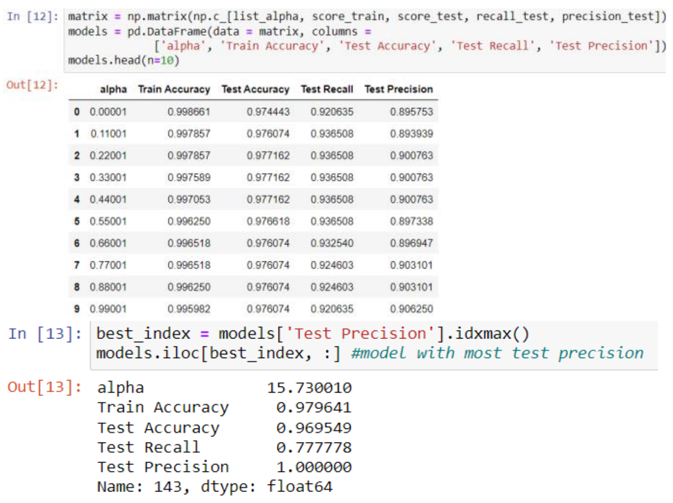
F: Split Train and Test Dataset



```
In [10]: #predective analysis
         data["v1"]=data["v1"].map({'spam':1,'ham':0})
         X_train, X_test, y_train, y_test = model_selection.train_test_split(X, data['v1'], test_size=0.33, random_state=42)
         print([np.shape(X_train), np.shape(X_test)])

         [(3733, 8404), (1839, 8404)]
```

G: Test Precision

Train Accuracy, Test Accuracy, Precision, Recall and Execution Time for all Deep Learning and Machine Learning models so that you can quickly choose the best model while taking into account all scores and losses. Testing accuracy is always a measure of your model's capability to predict the results on new data, whereas training accuracy

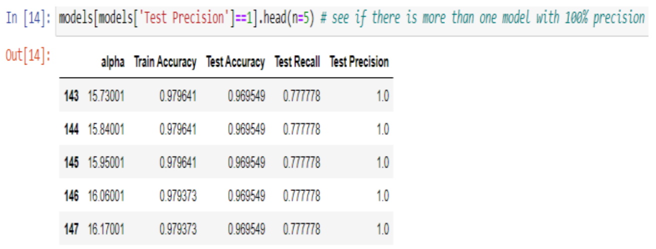


```
In [12]: matrix = np.matrix(np.c_[list_alpha, score_train, score_test, recall_test, precision_test])
         models = pd.DataFrame(data = matrix, columns =
                      ['alpha', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
         models.head(n=10)
```

Out[12]:

	alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	0.00001	0.998661	0.974443	0.920635	0.895753
1	0.11001	0.997857	0.976074	0.936508	0.893939
2	0.22001	0.997857	0.977162	0.936508	0.900763
3	0.33001	0.997589	0.977162	0.936508	0.900763
4	0.44001	0.997053	0.977162	0.936508	0.900763
5	0.55001	0.996250	0.976618	0.936508	0.897338
6	0.66001	0.996518	0.976074	0.932540	0.896947
7	0.77001	0.996518	0.976074	0.924603	0.903101
8	0.88001	0.996250	0.976074	0.924603	0.903101
9	0.99001	0.995982	0.976074	0.920635	0.906250

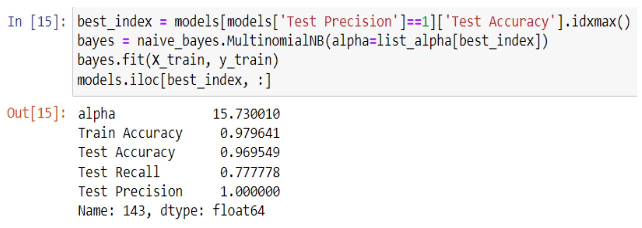
```
In [13]: best_index = models['Test Precision'].idxmax()
         models.iloc[best_index, :] #model with most test precision

Out[13]: alpha              15.730010
         Train Accuracy      0.979641
         Test Accuracy       0.969549
         Test Recall         0.777778
         Test Precision      1.000000
         Name: 143, dtype: float64
```



```
In [14]: models[models['Test Precision']==1].head(n=5) # see if there is more than one model with 100% precision
```

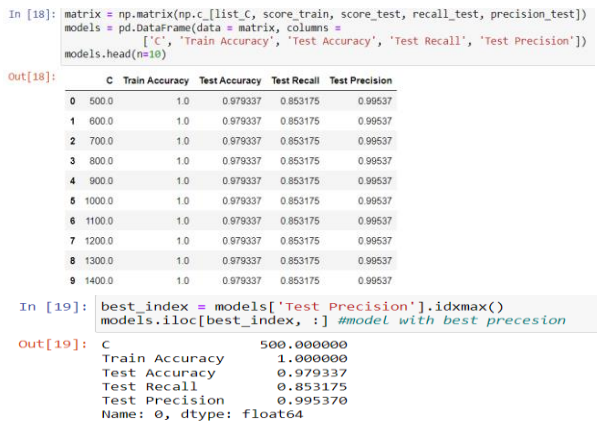
Out[14]:

	alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
143	15.73001	0.979641	0.969549	0.777778	1.0
144	15.84001	0.979641	0.969549	0.777778	1.0
145	15.95001	0.979641	0.969549	0.777778	1.0
146	16.06001	0.979373	0.969549	0.777778	1.0
147	16.17001	0.979373	0.969549	0.777778	1.0



```
In [15]: best_index = models[models['Test Precision']==1]['Test Accuracy'].idxmax()
         bayes = naive_bayes.MultinomialNB(alpha=list_alpha[best_index])
         bayes.fit(X_train, y_train)
         models.iloc[best_index, :]

Out[15]: alpha              15.730010
         Train Accuracy      0.979641
         Test Accuracy       0.969549
         Test Recall         0.777778
         Test Precision      1.000000
         Name: 143, dtype: float64
```

Test Precision for SVM



```
In [18]: matrix = np.matrix(np.c_[list_C, score_train, score_test, recall_test, precision_test])
         models = pd.DataFrame(data = matrix, columns =
                      ['C', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
         models.head(n=10)
```

Out[18]:

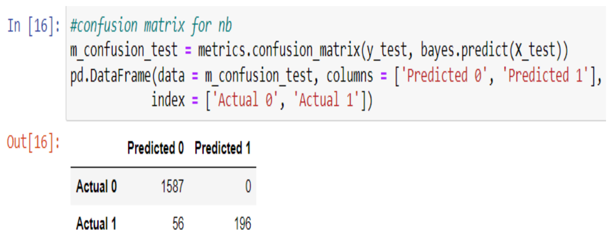
	C	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	500.0	1.0	0.979337	0.853175	0.99537
1	600.0	1.0	0.979337	0.853175	0.99537
2	700.0	1.0	0.979337	0.853175	0.99537
3	800.0	1.0	0.979337	0.853175	0.99537
4	900.0	1.0	0.979337	0.853175	0.99537
5	1000.0	1.0	0.979337	0.853175	0.99537
6	1100.0	1.0	0.979337	0.853175	0.99537
7	1200.0	1.0	0.979337	0.853175	0.99537
8	1300.0	1.0	0.979337	0.853175	0.99537
9	1400.0	1.0	0.979337	0.853175	0.99537

```
In [19]: best_index = models['Test Precision'].idxmax()
         models.iloc[best_index, :] #model with best precesion

Out[19]: C                    500.000000
         Train Accuracy         1.000000
         Test Accuracy          0.979337
         Test Recall            0.853175
         Test Precision         0.995370
         Name: 0, dtype: float64
```

H: Confusion Matrix

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an **error matrix**. Some features of Confusion matrix are given below:

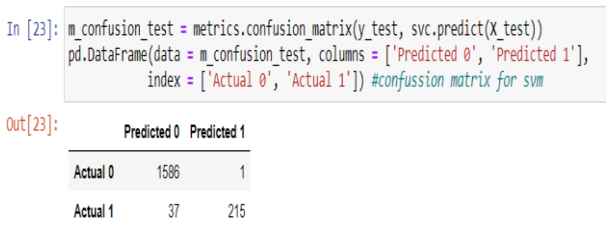
- For the 2 prediction classes of classifiers, the matrix is of 2*2 table, for 3 classes, it is 3*3 table, and so on.
- The matrix is divided into two dimensions, that are **predicted values** and **actual values** along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.



```
In [16]: #confusion matrix for nb
         m_confusion_test = metrics.confusion_matrix(y_test, bayes.predict(X_test))
         pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
                     index = ['Actual 0', 'Actual 1'])
```

Out[16]:

	Predicted 0	Predicted 1
Actual 0	1587	0
Actual 1	56	196



```
In [23]: m_confusion_test = metrics.confusion_matrix(y_test, svc.predict(X_test))
         pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
                     index = ['Actual 0', 'Actual 1']) #confussion matrix for svm
```

Out[23]:

	Predicted 0	Predicted 1
Actual 0	1586	1
Actual 1	37	215

I: Naïve Bayes Classifier Algorithm

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

J: Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.

However, primarily, it is used for Classification problems in Machine Learning

System Requirements

Hardware Requirements:

- Ryzen 5 or better (Processor).
- 1 GB Ram
- Hard disk 10 GB
- Peripherals

Software Requirements: -

- **Operating System:** Windows, Linux
- **Language used:** Python

Overview of Technologies Used:

About python:

Python is a high-level, interpreted programming language that is used for a wide range of applications, including web development, scientific computing, data analysis, artificial intelligence, and more. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages in use today. Some of the key features of Python include:

1. Simplicity: Python has a simple and easy-to-learn syntax, which makes it a great language for beginners to learn.

2. Interpreted: Python is an interpreted language, which means that code is executed line-by-line by an interpreter rather than being compiled into machine code

3. Object-oriented: Python supports object-oriented programming, which allows developers to create reusable and modular code.

4. Large standard library: Python has a large standard library that provides many useful functions and modules for a wide range of tasks, including file I/O, regular expressions, network programming, and more.

5. Third-party libraries: Python also has a large number of third-party libraries that can be easily installed and used, making it easy to extend the language for specific applications.

6. Cross-platform: Python can be run on a wide range of platforms,

7. including Windows, macOS, and Linux.

Overall, Python is a powerful and versatile programming language that can be used for many different applications, and its simplicity and flexibility make it a great choice for both beginners and experienced programmers.

About Machine Learning:

Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning.In its application across business problems, Machine learning is also referred to as predictive analytics

Naïve Bayes Algorithm:

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem

Naïve Bayes classifier

The Naïve Bayes algorithm is a simple probabilistic classifier that calculates a set of probabilities by counting the frequency and combination of values in a given dataset [4]. In this research, Naïve Bayes classifier use bag of words features to identify spam e-mail and a text is representing as the bag of its word. The bag of words is always used in methods of document classification, where the frequency of occurrence of each word is used as a feature for training classifier. This bag of words features are included in the chosen datasets.

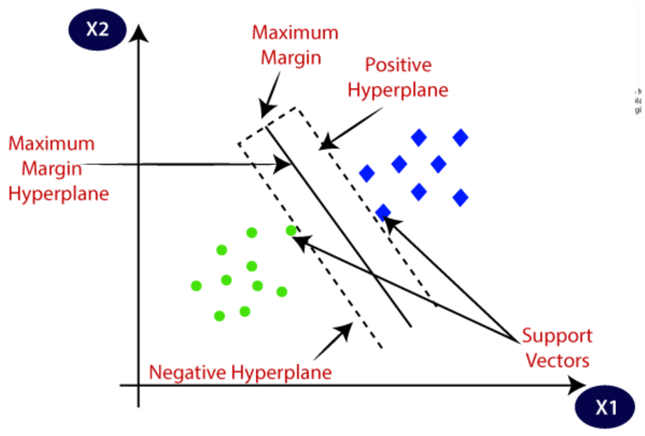
Naïve Bayes technique used Bayes theorem to determine that probabilities spam e-mail. Some words have particular probabilities of occurring in spam e-mail or non-spam e-mail. Example, suppose that we know exactly, that the word Free could never occur in a non-spam e-mail. Then, when we saw a message containing this word, we could tell for sure that were spam e-mail.

Support Vector Machine Algorithm:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier

Feasibility Study

The project feasibility is concern with expected benefits. An important outcome of the preliminary investigation is the destination that the proposed system is feasible: There are three aspects in the feasibility study portion of the preliminary investigation:

1. Economic Feasibility
2. Technical Feasibility
3. Operational Feasibility

Economic Feasibility:

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

Cost-based study: It is important to identify cost and benefit factors, which can be categorized as follows:

1. Development costs
2. Operating costs

This is an analysis of the costs to be incurred in the system and the benefits derivable out of the system.

Time-based study: This is an analysis of the time required to achieve a return on investments. The future value of a project is also a factor.

Technical Feasibility:

It is based on computer hardware and networking mobile application which would be available for the proposed system. Additional peripheral device is required which are available with the unit. The requirement for the proposed system can be easily made available by the user within the estimated cost which is minimal and of better usage.

Operational feasibility:

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development

Thakur Educational Trust's (Regd.)

THAKUR COLLEGE OF SCIENCE & COMMERCE

AUTONOMOUS COLLEGE AFFILIATED TO UNIVERSITY OF MUMBAI

NAAC Accredited with Grade 'A' (3rd Cycle) & ISO 9001: 2015 Certified

Best College Award by University of Mumbai for the Year 2018-2019

CELEBRATING 25 YEARS OF GLORY

Students Name: Mohini Bharambe

Guide Name: Ms. Siddhi Birwadar

Project Name: Spam E-mail Detection using Naïve Bayes Algorithm

College Name: Thakur College of Science and Commerce

PHASES	EXPECTED DATE OF COMPLETION	ACTUAL DATE OF COMPLETION	SIGNATURE
Preliminary Investigation	01-07-2022	03-07-2022	
System Analysis	08-07-2022	08-07-2022	
System Designing	12-08-2022	20-08-2022	
System Coding	27-08-2022	30-08-22	
System Implementation	31-08-22	31-08-22	
Report Submission	23-03-23	23-03-23	

System Analysis

Use Case Diagram:

A Use case is a description of set of sequence of actions. Graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship.

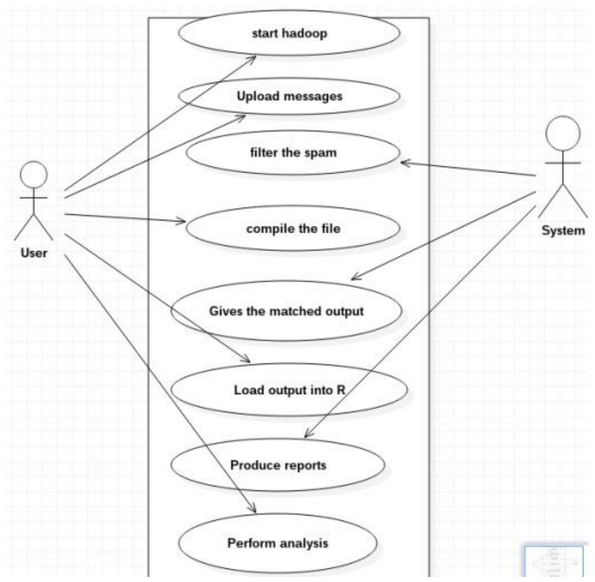
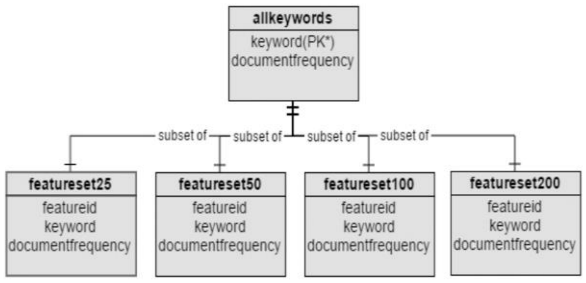


Figure 1: Use case Diagram

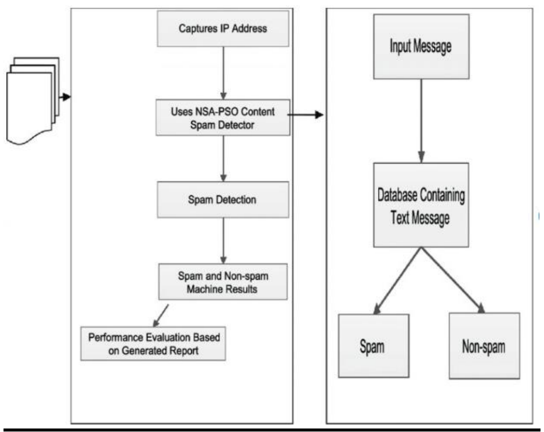
Entity Relationship Diagram:

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities.



Activity Diagram:

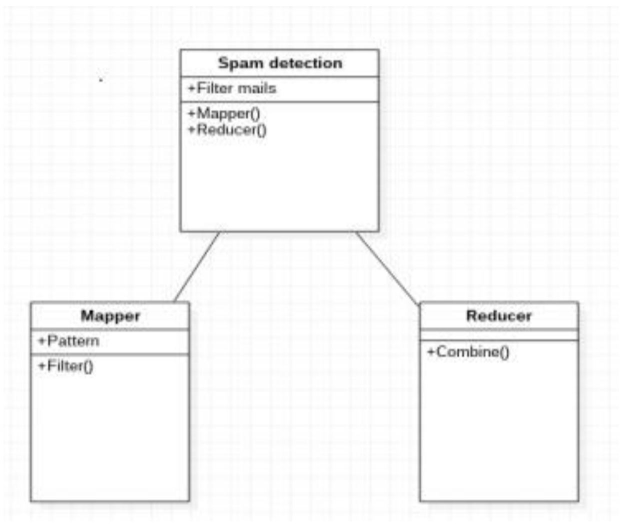
Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.


Captures IP Address
Uses NSA-PSO Content Spam Detector
Spam Detection
Spam and Non-spam Machine Results
Performance Evaluation Based on Generated Report
Input Message
Database Containing Text Message
Spam
Non-spam

Class Diagram:

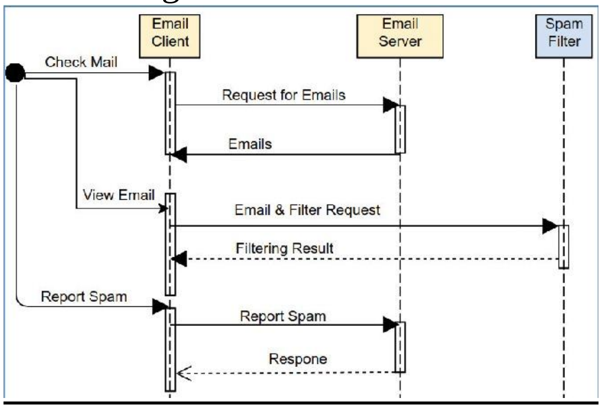
Class is nothing but a structure that contains both variables and methods. The Class Diagram shows a set of classes, interfaces, and collaborations and their relating ships. There is most common diagram in modeling the object-oriented systems and are used to give the static view of a system. It shows the dependency between the classes that can be used in our system.

The interactions between the modules or classes of our projects are shown below. Each block contains Class Name, Variables and Methods.



Sequence Diagram:

Sequence diagram and collaboration diagram are called INTERACTION DIAGRAMS. An interaction diagram shows an interaction, consisting of set of objects and their relationship including the messages that may be dispatched among them. A sequence diagram is an introduction that empathizes the time ordering of messages. Graphically a sequence diagram is a table that shows objects arranged along the X-axis and messages ordered in increasing time along the Y-axis.



CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from sklearn import feature_extraction,
model_selection, naive_bayes, metrics,
svm
from IPython.display import Image
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline

data =
pd.read_csv('C:/Users/OMMOHINI/De
sktop/spam.csv', encoding='latin-1')
data.head(n=10)

count_Class=pd.value_counts(data["v1
"], sort= True)
count_Class.plot(kind= 'bar', color=
["blue", "orange"])
plt.title('Bar chart')
plt.show()

count_Class.plot(kind = 'pie',
autopct='%1.0f%%')
plt.title('Pie chart')
plt.ylabel('')
plt.show()

count1 = Counter("
".join(data[data['v1']=='ham']["v2"]).sp
lit()).most_common(20)
df1 = pd.DataFrame.from_dict(count1)
df1 = df1.rename(columns={0: "words
in non-spam", 1 : "count"})
count2 = Counter("
".join(data[data['v1']=='spam']["v2"]).s
plit()).most_common(20)
df2 = pd.DataFrame.from_dict(count2)
df2 = df2.rename(columns={0: "words
in spam", 1 : "count_"})

df1.plot.bar(legend = False)
y_pos = np.arange(len(df1["words in
non-spam"]))
plt.xticks(y_pos, df1["words in non-
spam"])
plt.title('More frequent words in non-
spam messages')
plt.xlabel('words')
plt.ylabel('number')
plt.show()

df2.plot.bar(legend = False, color =
'orange')
y_pos = np.arange(len(df2["words in
spam"]))
plt.xticks(y_pos, df2["words in spam"])
plt.title('More frequent words in spam
messages')
plt.xlabel('words')
plt.ylabel('number')
plt.show()

#feature engineering
f =
feature_extraction.text.CountVectorize
r(stop_words = 'english')
X = f.fit_transform(data["v2"])
np.shape(X)

#predective analysis
data["v1"]=data["v1"].map({'spam':1,'h
am':0})
```

```
X_train, X_test, y_train, y_test =
model_selection.train_test_split(X,
data['v1'], test_size=0.33,
random_state=42)
print([np.shape(X_train),
np.shape(X_test)])

#multinomial nb
list_alpha = np.arange(1/100000, 20,
0.11)
score_train = np.zeros(len(list_alpha))
score_test = np.zeros(len(list_alpha))
recall_test = np.zeros(len(list_alpha))
precision_test=
np.zeros(len(list_alpha))
count = 0
for alpha in list_alpha:
   bayes =
naive_bayes.MultinomialNB(alpha=alp
ha)
   bayes.fit(X_train, y_train)
   score_train[count] =
bayes.score(X_train, y_train)
   score_test[count]=
bayes.score(X_test, y_test)
   recall_test[count] =
metrics.recall_score(y_test,
bayes.predict(X_test))
   precision_test[count] =
metrics.precision_score(y_test,
bayes.predict(X_test))
   count = count + 1

matrix = np.matrix(np.c_[list_alpha,
score_train, score_test, recall_test,
precision_test])
models = pd.DataFrame(data = matrix,
columns =
          ['alpha', 'Train Accuracy', 'Test
Accuracy', 'Test Recall', 'Test
Precision'])
models.head(n=10)

best_index = models['Test
Precision'].idxmax()
models.iloc[best_index, :] #model with
most test precision

models[models['Test
Precision']==1].head(n=5) # see if
there is more than one model with
100% precision

best_index = models[models['Test
Precision']==1]['Test
Accuracy'].idxmax()
bayes =
naive_bayes.MultinomialNB(alpha=list
_alpha[best_index])
bayes.fit(X_train, y_train)
models.iloc[best_index, :]

#confusion matrix for nb
m_confusion_test =
metrics.confusion_matrix(y_test,
bayes.predict(X_test))
pd.DataFrame(data =
m_confusion_test, columns =
['Predicted 0', 'Predicted 1'],
         index = ['Actual 0', 'Actual 1'])

#svm start
list_C = np.arange(500, 2000, 100) +
score_train = np.zeros(len(list_C))
score_test = np.zeros(len(list_C))
recall_test = np.zeros(len(list_C))
precision_test= np.zeros(len(list_C))
```

```
count = 0
for C in list_C:
   svc = svm.SVC(C=C)
   svc.fit(X_train, y_train)
   score_train[count] =
svc.score(X_train, y_train)
   score_test[count]= svc.score(X_test,
y_test)
   recall_test[count] =
metrics.recall_score(y_test,
svc.predict(X_test))
   precision_test[count] =
metrics.precision_score(y_test,
svc.predict(X_test))
   count = count + 1

matrix = np.matrix(np.c_[list_C,
score_train, score_test, recall_test,
precision_test])
models = pd.DataFrame(data = matrix,
columns =
          ['C', 'Train Accuracy', 'Test
Accuracy', 'Test Recall', 'Test
Precision'])
models.head(n=10)

best_index = models['Test
Precision'].idxmax()
models.iloc[best_index, :] #model with
best precesion

models[models['Test
Precision']==1].head(n=5) #to see if
there is more than 1 with 100%
precesion

m_confusion_test =
metrics.confusion_matrix(y_test,
svc.predict(X_test))
pd.DataFrame(data =
m_confusion_test, columns =
['Predicted 0', 'Predicted 1'],
          index = ['Actual 0', 'Actual 1'])
#confussion matrix for svm

#We misclassify 31 spam as non-spam
messages whereas we don't misclassify
any non-spam message.

#Conclusion
#The best model I have found is
support vector machine with 98.3%
accuracy.

#It classifies every non-spam message
correctly (Model precision)

#It classifies the 87.7% of spam
messages correctly (Model recall)
```

Output:

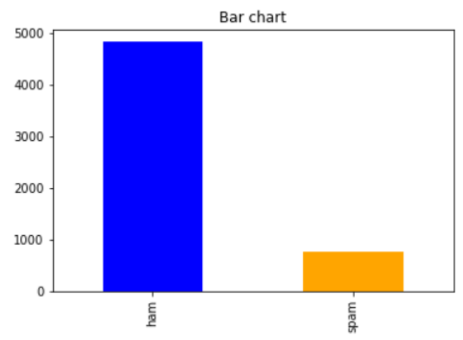
The model was running successfully with a model score of 99.57 %.

```
In [3]: data = pd.read_csv('C:/Users/OMMOHINI/Desktop/spam.csv', encoding='latin-1')
        data.head(n=10)
```

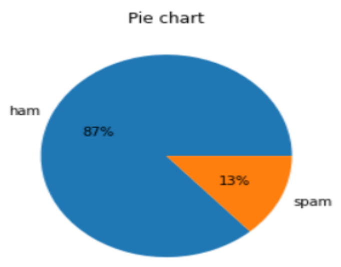
Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif 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 already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
5	spam	FreeMsg Hey there darling it's been 3 week's n...	NaN	NaN	NaN
6	ham	Even my brother is not like to speak with me. ...	NaN	NaN	NaN
7	ham	As per your request 'Melle Melle (Oru Minnamin...	NaN	NaN	NaN
8	spam	WINNER!! As a valued network customer you have...	NaN	NaN	NaN
9	spam	Had your mobile 11 months or more? U R entitle...	NaN	NaN	NaN

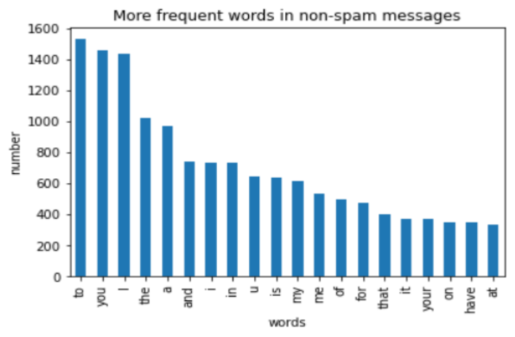
```
In [4]: count_Class=pd.value_counts(data["v1"], sort= True)
        count_Class.plot(kind= 'bar', color= ["blue", "orange"])
        plt.title('Bar chart')
        plt.show()
```



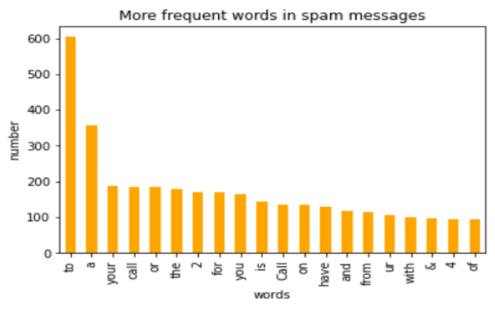
```
In [5]: count_Class.plot(kind = 'pie',  autopct='%1.0f%%')
        plt.title('Pie chart')
        plt.ylabel('')
        plt.show()
```



```
In [7]: df1.plot.bar(legend = False)
        y_pos = np.arange(len(df1["words in non-spam"]))
        plt.xticks(y_pos, df1["words in non-spam"])
        plt.title('More frequent words in non-spam messages')
        plt.xlabel('words')
        plt.ylabel('number')
        plt.show()
```



```
In [8]: df2.plot.bar(legend = False, color = 'orange')
        y_pos = np.arange(len(df2["words in spam"]))
        plt.xticks(y_pos, df2["words in spam"])
        plt.title('More frequent words in spam messages')
        plt.xlabel('words')
        plt.ylabel('number')
        plt.show()
```



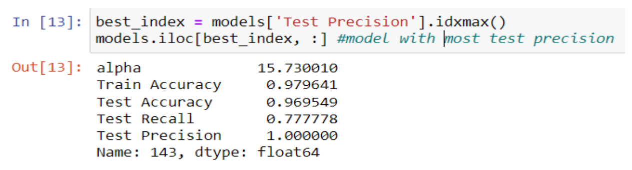
```
In [9]: #feature engineering
        f = feature_extraction.text.CountVectorizer(stop_words = 'english')
        X = f.fit_transform(data["v2"])
        np.shape(X)

Out[9]: (5572, 8404)
```

```
In [10]: #predective analysis
         data["v1"]=data["v1"].map({'spam':1,'ham':0})
         X_train, X_test, y_train, y_test = model_selection.train_test_split(X, data['v1'], test_size=0.33, random_state=42)
         print([np.shape(X_train), np.shape(X_test)])

         [(3733, 8404), (1839, 8404)]
```

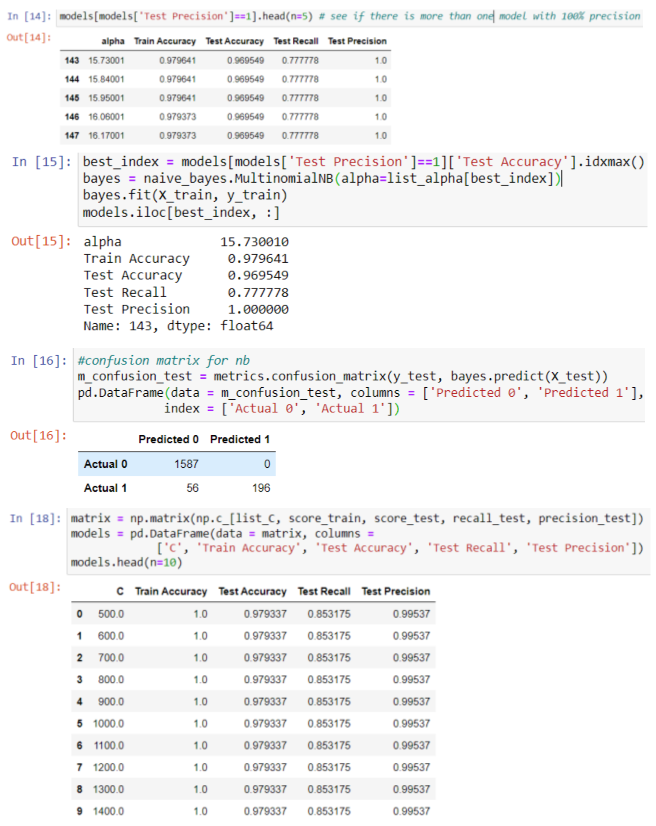
```
In [12]: matrix = np.matrix(np.c_[list_alpha, score_train, score_test, recall_test, precision_test])
         models = pd.DataFrame(data = matrix, columns =
                      ['alpha', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
         models.head(n=10)
```

Out[12]:

	alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	0.00001	0.998661	0.974443	0.920635	0.895753
1	0.11001	0.997857	0.976074	0.936508	0.893939
2	0.22001	0.997857	0.977162	0.936508	0.900763
3	0.33001	0.997589	0.977162	0.936508	0.900763
4	0.44001	0.997053	0.977162	0.936508	0.900763
5	0.55001	0.996250	0.976618	0.936508	0.897338
6	0.66001	0.996518	0.976074	0.932540	0.896947
7	0.77001	0.996518	0.976074	0.924603	0.903101
8	0.88001	0.996250	0.976074	0.924603	0.903101
9	0.99001	0.995982	0.976074	0.920635	0.906250



```
In [13]: best_index = models['Test Precision'].idxmax()
         models.iloc[best_index, :] #model with most test precision

Out[13]: alpha              15.730010
         Train Accuracy      0.979641
         Test Accuracy       0.969549
         Test Recall         0.777778
         Test Precision      1.000000
         Name: 143, dtype: float64
```



```
In [14]: models[models['Test Precision']==1].head(n=5) # see if there is more than one model with 100% precision
Out[14]:
```

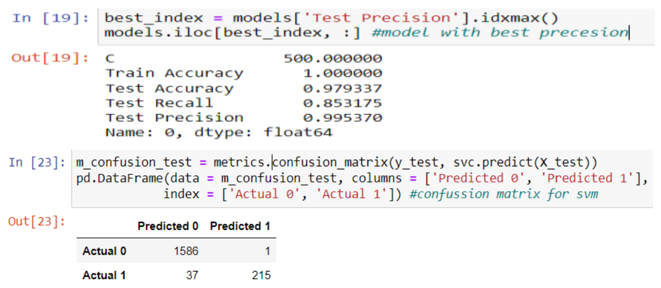
	alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
143	15.73001	0.979641	0.969549	0.777778	1.0
144	15.84001	0.979641	0.969549	0.777778	1.0
145	15.95001	0.979641	0.969549	0.777778	1.0
146	16.06001	0.979373	0.969549	0.777778	1.0
147	16.17001	0.979373	0.969549	0.777778	1.0

```
In [15]: best_index = models[models['Test Precision']==1]['Test Accuracy'].idxmax()
         bayes = naive_bayes.MultinomialNB(alpha=list_alpha[best_index])
         bayes.fit(X_train, y_train)
         models.iloc[best_index, :]

Out[15]: alpha              15.730010
         Train Accuracy      0.979641
         Test Accuracy       0.969549
         Test Recall         0.777778
         Test Precision      1.000000
         Name: 143, dtype: float64

In [16]: #confusion matrix for nb
         m_confusion_test = metrics.confusion_matrix(y_test, bayes.predict(X_test))
         pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
                     index = ['Actual 0', 'Actual 1'])

Out[16]:
```

	Predicted 0	Predicted 1
Actual 0	1587	0
Actual 1	56	196

```
In [18]: matrix = np.matrix(np.c_[list_C, score_train, score_test, recall_test, precision_test])
         models = pd.DataFrame(data = matrix, columns =
                      ['C', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
         models.head(n=10)

Out[18]:
```

	C	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	500.0	1.0	0.979337	0.853175	0.99537
1	600.0	1.0	0.979337	0.853175	0.99537
2	700.0	1.0	0.979337	0.853175	0.99537
3	800.0	1.0	0.979337	0.853175	0.99537
4	900.0	1.0	0.979337	0.853175	0.99537
5	1000.0	1.0	0.979337	0.853175	0.99537
6	1100.0	1.0	0.979337	0.853175	0.99537
7	1200.0	1.0	0.979337	0.853175	0.99537
8	1300.0	1.0	0.979337	0.853175	0.99537
9	1400.0	1.0	0.979337	0.853175	0.99537



```
In [19]: best_index = models['Test Precision'].idxmax()
         models.iloc[best_index, :] #model with best precesion

Out[19]: C                  500.000000
         Train Accuracy       1.000000
         Test Accuracy        0.979337
         Test Recall          0.853175
         Test Precision       0.995370
         Name: 0, dtype: float64

In [23]: m_confusion_test = metrics.confusion_matrix(y_test, svc.predict(X_test))
         pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
                     index = ['Actual 0', 'Actual 1']) #confussion matrix for svm

Out[23]:
```

	Predicted 0	Predicted 1
Actual 0	1586	1
Actual 1	37	215

Future Enhancements

Following future enhancements are possible:

- Code can be more structured by using Machine Learning Frameworks
- Shortcut keys can be provided to access different options.

References and Bibliography

[1] Aditya Gupta "Spam Filter using Naïve Bayesian Technique "International Journal of Computational Engineering Research (IJCER), vol. 08, no. 06, 2018.

[2] Spam Message Classification Based on the Naïve Bayes Classification Algorithm Bin Ning, Wu Junwei, Hu Feng, Feb 2019.
et al 2017

[3] An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages, Ion Androutsopoulos, John Koutsias, Konstantinos V. Cbandrinos, July 2000.

[4] YouTube spam detection framework using naïve bayes and logistic regression, Nur'Ain Maulat Samsudin, Cik Feresa binti Mohd Foozy, July 2019.

[5] Analysis of Naïve Bayes Algorithm for Email Spam Filtering across Multiple Datasets, Nurul Fitriah Rusland