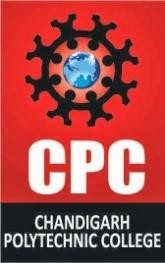
# PROJECT REPORT ON

**E COMMERCE STORE**

Submitted in the partial fulfillment of requirement for the award of the Diploma of

## COMPUTER ENGINEERING



**SUBMITTED BY**

ASHUWANI KUMAR SINGH 171076250619

# UNDER THE GUIDANCE OF

**ER.MANJEET KAUR Er. DIMPLE BHASIN** COMPUTER ENGG. DEPARTMENT ASSISTANT PROFESSOR HEAD OF DEPARTMENT COMPUTER ENGG DEPARTMENT

## DEPARTMENT OF COMPUTER ENGINEERING CHANDIGARH POLYTECHNIC COLLEGE, GHARUAN MAY – 2019

**CERTIFICATE**

This is to certify that the project **“Email Spam Detector”** submitted by ASHWANI KUMAR SINGH in partial fulfillment of the requirements for the award of diploma of “COMPUTER ENGINEERING” to Chandigarh Polytechnic College Gharuan, is a record of student’s own work carried out by them under my supervision and guidance.

# HEAD OF DEPARTMENT PROJECT GUIDE

## ER. MANJEET KAUR ER. DIMPLE BHASIN

Computer Engineering Computer Engineering

Chandigarh Polytechnic College Chandigarh Polytechnic College

Gharuan Gharuan

# ACKNOWLEDGEMENT

The Project work is a task which cannot be done by an individual in isolation. I have no exception to this rule and have gratefully accepted the helping hand offered.

I express my deep gratitude and indebtedness to **Dr. Gurmeet Singh Swag** (Principal Chandigarh Polytechnic College Gharuan) for providing us such a platform, infrastructure and learning environment.

I am also grateful to “**Er. MANJEET KAUR**” (HOD “COMPUTER ENGINEERING” Chandigarh Polytechnic College Gharuan) for extending all possible help in formation and completion of Project.

I am also grateful to my guide “**ER. DIMPLE BHASIN**” (Assist. Professor, Department of “COMPUTER ENGINEERING” Chandigarh Polytechnic College Gharuan) for her valuable guidance and constant encouragement at each and every step-in preparation of this Project work until its completion.

ASHUWANI SINGH

# ABSTRACT

The upsurge in the volume of unwanted emails called spam has created an intense need for the development of more dependable and robust antispam filters. Machine learning methods of recent are being used to successfully detect and filter spam emails. We present a systematic review of some of the popular machine learning based email spam filtering approaches. Our review covers survey of the important concepts, attempts, efficiency, and the research trend in spam filtering. The preliminary discussion in the study background examines the applications of machine learning techniques to the email spam filtering process of the leading internet service providers (ISPs) like Gmail, Yahoo and Outlook emails spam filters. Discussion on general email spam filtering process, and the various efforts by different researchers in combating spam through the use machine learning techniques was done. Our review compares the strengths and drawbacks of existing machine learning approaches and the open research problems in spam filtering. We recommended deep leaning and deep adversarial learning as the future techniques that can effectively handle the menace of spam emails.

In recent times, unwanted commercial bulk emails called spam has become a huge problem on the internet. The person sending the spam messages is referred to as the spammer. Such a person gathers email addresses from different websites, chatrooms, and viruses . Spam prevents the user from making full and good use of time, storage capacity and network bandwidth. The huge volume of spam mails flowing through the computer networks have destructive effects on the memory space of email servers, communication bandwidth, CPU power and user time . The menace of spam email is on the increase on yearly basis and is responsible for over 77% of the whole global email traffic . Users who receive spam emails that they did not request find it very irritating. It is also resulted to untold financial loss to many users who have fallen victim of internet scams and other fraudulent practices of spammers who send emails pretending to be from reputable companies with the intention to persuade individuals to disclose sensitive personal information like passwords, Bank Verification Number (BVN) and credit card numbers.

According to report from Kaspersky lab, in 2015, the volume of spam emails being sent reduced to a 12-year low. Spam email volume fell below 50% for the first time since 2003. In June 2015, the volume of spam emails went down to 49.7% and in July 2015 the figures was further reduced to 46.4% according to anti-virus software developer Symantec. This decline was attributed to reduction in the number of major botnets responsible for sending spam emails in billions. Malicious spam email volume was reported to be constant in 2015. The figure of spam mails detected by Kaspersky Lab in 2015 was between 3 million and 6 million. Conversely, as the year was about to end, spam email volume escalated. Further report from Kaspersky Lab indicated that spam email messages having pernicious attachments such as malware, ransomware, malicious macros, and JavaScript started to increase in December 2015. That drift was sustained in 2016 and by March of that year spam email volume had quadrupled with respect to that witnessed in 2015. In March 2016, the volume of spam emails discovered by Kaspersky Lab is 22,890,956. By that time the volume of spam emails had skyrocketed to an average of 56.92% for the first quarter of 2016. Latest statistics shows that spam messages accounted for 56.87% of e-mail traffic worldwide and the most familiar types of spam emails were healthcare and dating spam. Spam results into unproductive use of resources on Simple Mail Transfer Protocol (SMTP) servers since they have to process a substantial volume of unsolicited emails . The volume of spam emails containing malware and other malicious codes between the fourth quarter of 2016 and first quarter of 2018 is depicted in [Fig. 1](https://www.sciencedirect.com/science/article/pii/S2405844018353404" \l "fig1) below.

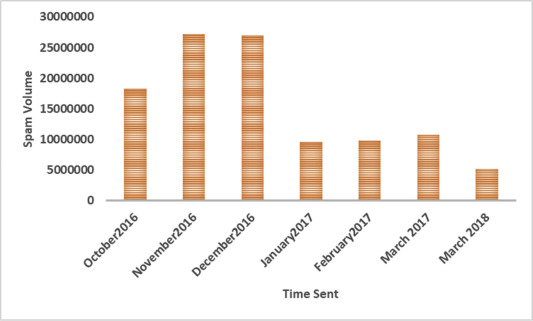


Fig. 1. The volume of spam emails 4th quarter 2016 to 1st quarter 2018.

To effectively handle the threat posed by email spams, leading email providers such as Gmail, Yahoo mail and Outlook have employed the combination of different machine learning (ML) techniques such as Neural Networks in its spam filters. These ML techniques have the capacity to learn and identify spam mails and phishing messages by analyzing loads of such messages throughout a vast collection of computers. Since machine learning have the capacity to adapt to varying conditions, Gmail and Yahoo mail spam filters do more than just checking junk emails using pre-existing rules. They generate new rules themselves based on what they have learnt as they continue in their spam filtering operation. The machine learning model used by Google have now advanced to the point that it can detect and filter out spam and phishing emails with about 99.9 percent accuracy. The implication of this is that one out of a thousand messages succeed in evading their email spam filter. Statistics from Google revealed that between 50-70 percent of emails that Gmail receives are unsolicited mail. Google's detection models have also incorporated tools called Google Safe Browsing for identifying websites that have malicious URLs. The phishing-detection performance of Google have been enhanced by introduction of a system that delay the delivery of some Gmail messages for a while to carry out additional comprehensive scrutiny of the phishing messages since they are easier to detect when they are analyzed collectively. The purpose of delaying the delivery of some of these suspicious emails is to conduct a deeper examination while more messages arrives in due course of time and the algorithms are updated in real time. Only about 0.05 percent of emails are affected by this deliberate delay.

Though there are several email spam filtering methods in existence, the state-of-the-art approaches are discussed in this paper. We explained below the different categories of spam filtering techniques that have been widely applied to overcome the problem of email spam.

•

**Content Based Filtering Technique:** Content based filtering is usually used to create automatic filtering rules and to classify emails using machine learning approaches, such as Naïve Bayesian classification, Support Vector Machine, K Nearest Neighbor, Neural Networks. This method normally analyses words, the occurrence, and distributions of words and phrases in the content of emails and used then use generated rules to filter the incoming email spams .

•

**Case Base Spam Filtering Method:** Case base or sample base filtering is one of the popular spam filtering methods. Firstly, all emails both non-spam and spam emails are extracted from each user's email using collection model. Subsequently, pre-processing steps are carried out to transform the email using client interface, feature extraction, and selection, grouping of email data, and evaluating the process. The data is then classified into two vector sets. Lastly, the machine learning algorithm is used to train datasets and test them to decide whether the incoming mails are spam or non-spam .

•

**Heuristic or Rule Based Spam Filtering Technique:** This approach uses already created rules or heuristics to assess a huge number of patterns which are usually regular expressions against a chosen message. Several similar patterns increase the score of a message. In contrast, it deducts from the score if any of the patterns did not correspond. Any message's score that surpasses a specific threshold is filtered as spam; else it is counted as valid. While some ranking rules do not change over time, other rules require constant updating to be able to cope effectively with the menace of spammers who continuously introduce new spam messages that can easily escape without been noticed from email filters . A good example of a rule based spam filter is Spam Assassin .

•

**Previous Likeness Based Spam Filtering Technique:** This approach uses memory-based, or instance-based, machine learning methods to classify incoming emails based to their resemblance to stored examples (e.g. training emails). The attributes of the email are used to create a multi-dimensional space vector, which is used to plot new instances as points. The new instances are afterward allocated to the most popular class of its K-closest training instances . This approach uses the k-nearest neighbor (kNN) for filtering spam emails.

•

**Adaptive Spam Filtering Technique:** The method detects and filters spam by grouping them into different classes. It divides an email corpus into various groups, each group has an emblematic text. A comparison is made between each incoming email and each group, and a percentage of similarity is produced to decide the probable group the email belongs to .

Many researchers and academicians have proposed different email spam classification techniques which have been successfully used to classify data into groups. These methods include probabilistic, decision tree, artificial immune system , support vector machine (SVM) , artificial neural networks (ANN) , and case-based technique . It have been shown in literature that it is possible to use these classification methods for spam mail filtering by using content-based filtering technique that will identify certain features (normally keywords frequently utilised in spam emails). The rate at which these features appear in emails ascertain the probabilities for each characteristic in the email, after which it is measured against the threshold value. Email messages that exceed the threshold value are classified as spam . ANN is a non-linear model that seeks to imitate the functions of biological neural networks. It is made up of simple processing components named neurons and carries out its computational operations by processing information. Several research work have employed neural network to classify unwanted emails as spam by applying content-based filtering. These techniques decide the properties by either computing the rate of occurrence of keywords or patterns in the email messages. Literatures show that Neural Network algorithms that are utilised in email filtering attain moderate classification performance. Some of the most popular spam email classification algorithms are Multilayer Perceptron Neural Networks (MLPNNs) and Radial Base Function Neural Networks (RBFNN). Researchers used MLPNN as a classifier for spam filtering but not many of them used RBFNN for classification.

# Table of Contents

## Contents Page No.

Certificate

Acknowledgements

Abstract \_ Table of Contents

## Chapter No. 1

1. Introduction
2. Learning Data
3. Data Pre-processing

## Chapter No. 2

Preparing Text Data

## Chapter No. 3 \_

Creating word dictionary

## Chapter No. 4

Feature extraction process

## Chapter No. 5

Training the classifiers

## Chapter No. 6

Training the classifiers

## Chapter No. 7

Checking Performance

# Chapter 1

# INTRODUCTION

In today’s globalized world, email is a primary source of communication. This communication can vary from personal, business, corporate to government. With the rapid increase in email usage, there has also been increase in the SPAM emails. SPAM emails, also known as junk email involves nearly identical messages sent to numerous recipients by email. Apart from being annoying, spam emails can also pose a security threat to computer system. It is estimated that spam cost businesses on the order of $100 billion in 2007. In this project, we use text mining to perform automatic spam filtering to use emails effectively. We try to identify patterns using Data-mining classification algorithms to enable us classify the emails as HAM or SPAM.

**LEARNING DATA**

The data used for this project was taken from the Spam Assassin public corpus website. It consists of two data sets: train and test. Each dataset contains a randomly selected collection of emails in plain text format, which have been labelled as HAM or SPAM. The training data is used to build a model for classifying emails into HAM and SPAM. The test data is used to check the accuracy of the model built with the training data. The training data set contains 400 emails with 283 ham and 117 spam emails. The test data contains 200 emails with 139 ham and 61 spam emails.

**DATA PREPROCESSING**

Text mining (deriving information from text) is a wide field which has gained popularity with the huge text data being generated. Automation of a number of applications like sentiment analysis, document classification, topic classification, text summarization, machine translation, etc has been done using machine learning models.

Spam filtering is a beginner’s example of document classification task which involves classifying an email as spam or non-spam (a.k.a. ham) mail. Spam box in your Gmail account is the best example of this. So lets get started in building a spam filter on a publicly available mail corpus. I have extracted equal number of spam and non-spam emails from ling-Spam corpus.

We will walk through the following steps to build this application :

1. Preparing the text data.
2. Creating word dictionary.
3. Feature extraction process
4. Training the classifier

Further, we will check the results on test set of the subset created.

### Preparing the text data. The data-set used here, is split into a training set and a test set containing 702 mails and 260 mails respectively, divided equally between spam and ham mails. You will easily recognize spam mails as it contains \*spmsg\* in its filename.

In any text mining problem, text cleaning is the first step where we remove those words from the document which may not contribute to the information we want to extract. Emails may contain a lot of undesirable characters like punctuation marks, stop words, digits, etc which may not be helpful in detecting the spam email. The emails in Ling-spam corpus have been already preprocessed in the following ways:

a) Removal of stop words – Stop words like “and”, “the”, “of”, etc are very common in all English sentences and are not very meaningful in deciding spam or legitimate status, so these words have been removed from the emails.

b) Lemmatization – It is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. For example, “include”, “includes,” and “included” would all be represented as “include”. The context of the sentence is also preserved in lemmatization as opposed to stemming (another buzz word in text mining which does not consider meaning of the sentence).

We still need to remove the non-words like punctuation marks or special characters from the mail documents. There are several ways to do it. Here, we will remove such words after creating a dictionary, which is a very convenient method to do so since when you have a dictionary, you need to remove every such word only once. So cheers !! As of now you don’t need to do anything.

### Creating word dictionary.

It can be seen that the first line of the mail is subject and the 3rd line contains the body of the email. We will only perform text analytics on the content to detect the spam mails. As a first step, we need to create a dictionary of words and their frequency. For this task, training set of 700 mails is utilized. This python function creates the dictionary for you.

def make\_Dictionary(train\_dir):

emails = [os.path.join(train\_dir,f) for f in os.listdir(train\_dir)]

all\_words = []

for mail in emails:

with open(mail) as m:

for i,line in enumerate(m):

if i == 2: #Body of email is only 3rd line of text file

words = line.split()

all\_words += words

dictionary = Counter(all\_words)

return dictionary

Once the dictionary is created we can add just a few lines of code written below to the above function to remove non-words about which we talked in step 1. I have also removed absurd single characters in the dictionary which are irrelevant here. Do not forget to insert the below code in the function def make\_Dictionary(train\_dir).

list\_to\_remove = dictionary.keys()

for item in list\_to\_remove:

if item.isalpha() == False:

del dictionary[item]

elif len(item) == 1:

del dictionary[item]

dictionary = dictionary.most\_common(3000)

Dictionary can be seen by the command print dictionary. You may find some absurd word counts to be high but don’t worry, it’s just a dictionary and you always have the scope of  improving it later. If you are following this blog with provided data-set, make sure your dictionary has some of the entries given below as most frequent words. Here I have chosen 3000 most frequently used words in the dictionary.

### 3. Feature extraction process.

Once the dictionary is ready, we can extract word count vector (our feature here) of 3000 dimensions for each email of training set. Each **word count vector** contains the frequency of 3000 words in the training file. Of course you might have guessed by now that most of them will be zero. Let us take an example. Suppose we have 500 words in our dictionary. Each word count vector contains the frequency of 500 dictionary words in the training file. Suppose text in training file was “Get the work done, work done” then it will be encoded as [0,0,0,0,0,…….0,0,2,0,0,0,……,0,0,1,0,0,…0,0,1,0,0,……2,0,0,0,0,0]. Here, all the word counts are placed at 296th, 359th, 415th, 495th index of 500 length word count vector and the rest are zero.

The below python code will generate a feature vector matrix whose rows denote 700 files of training set and columns denote 3000 words of dictionary. The value at index ‘ij’ will be the number of occurrences of jth word of dictionary in ith file.

def extract\_features(mail\_dir):

files = [os.path.join(mail\_dir,fi) for fi in os.listdir(mail\_dir)]

features\_matrix = np.zeros((len(files),3000))

docID = 0;

for fil in files:

with open(fil) as fi:

for i,line in enumerate(fi):

if i == 2:

words = line.split()

for word in words:

wordID = 0

for i,d in enumerate(dictionary):

if d[0] == word:

wordID = i

features\_matrix[docID,wordID] = words.count(word)

docID = docID + 1

return features\_matrix

### 4. Training the classifiers.

Here, I will be using [scikit-learn ML library](http://scikit-learn.org/stable/) for training classifiers. It is an open source python ML library which comes bundled in 3rd party distribution [anaconda](https://www.continuum.io/downloads) or can be used by separate installation following [this](http://scikit-learn.org/stable/install.html). Once installed, we only need to import it in our program.

I have trained two models here namely Naive Bayes classifier and Support Vector Machines (SVM). Naive Bayes classifier is a conventional and very popular method for document classification problem. It is a supervised probabilistic classifier based on Bayes theorem assuming independence between every pair of features. SVMs are supervised binary classifiers which are very effective when you have higher number of features. The goal of SVM is to separate some subset of training data from rest called the support vectors (boundary of separating hyper-plane). The decision function of SVM model that predicts the class of the test data is based on support vectors and makes use of a kernel trick.

Once the classifiers are trained, we can check the performance of the models on test-set. We extract word count vector for each mail in test-set and predict its class(ham or spam) with the trained NB classifier and SVM model. Below is the full code for spam filtering application. You have to include the two functions we have defined before in step 2 and step 3.

import os

import numpy as np

from collections import Counter

from sklearn.naive\_bayes import MultinomialNB, GaussianNB, BernoulliNB

from sklearn.svm import SVC, NuSVC, LinearSVC

# Create a dictionary of words with its frequency

train\_dir = 'train-mails'

dictionary = make\_Dictionary(train\_dir)

# Prepare feature vectors per training mail and its labels

train\_labels = np.zeros(702)

train\_labels[351:701] = 1

train\_matrix = extract\_features(train\_dir)

# Training SVM and Naive bayes classifier

model1 = MultinomialNB()

model2 = LinearSVC()

model1.fit(train\_matrix,train\_labels)

model2.fit(train\_matrix,train\_labels)

# Test the unseen mails for Spam

test\_dir = 'test-mails'

test\_matrix = extract\_features(test\_dir)

test\_labels = np.zeros(260)

test\_labels[130:260] = 1

result1 = model1.predict(test\_matrix)

result2 = model2.predict(test\_matrix)

print confusion\_matrix(test\_labels,result1)

print confusion\_matrix(test\_labels,result2)

### Checking Performance

Test-set contains 130 spam emails and 130 non-spam emails. If you have come so far, you will find below results. I have shown the confusion matrix of the test-set for both the models. The diagonal elements represents the correctly identified(a.k.a. true identification) mails where as non-diagonal elements represents wrong classification (false identification) of mails.

|  |  |  |
| --- | --- | --- |
| Multinomial NB | Ham | Spam |
| Ham | 129 | 1 |
| Spam | 9 | 121 |

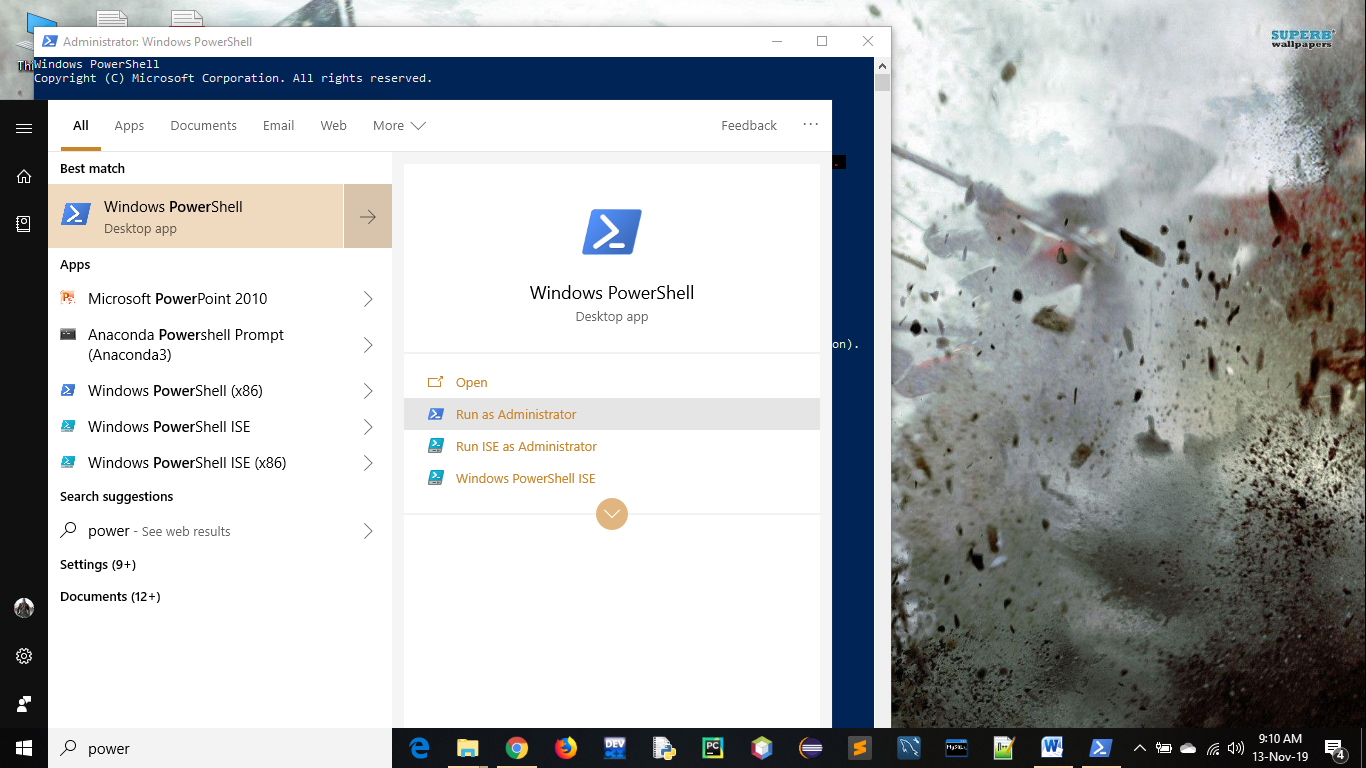
|  |  |  |
| --- | --- | --- |
| SVM(Linear) | Ham | Spam |
| Ham | 126 | 4 |
| Spam | 6 | 124 |

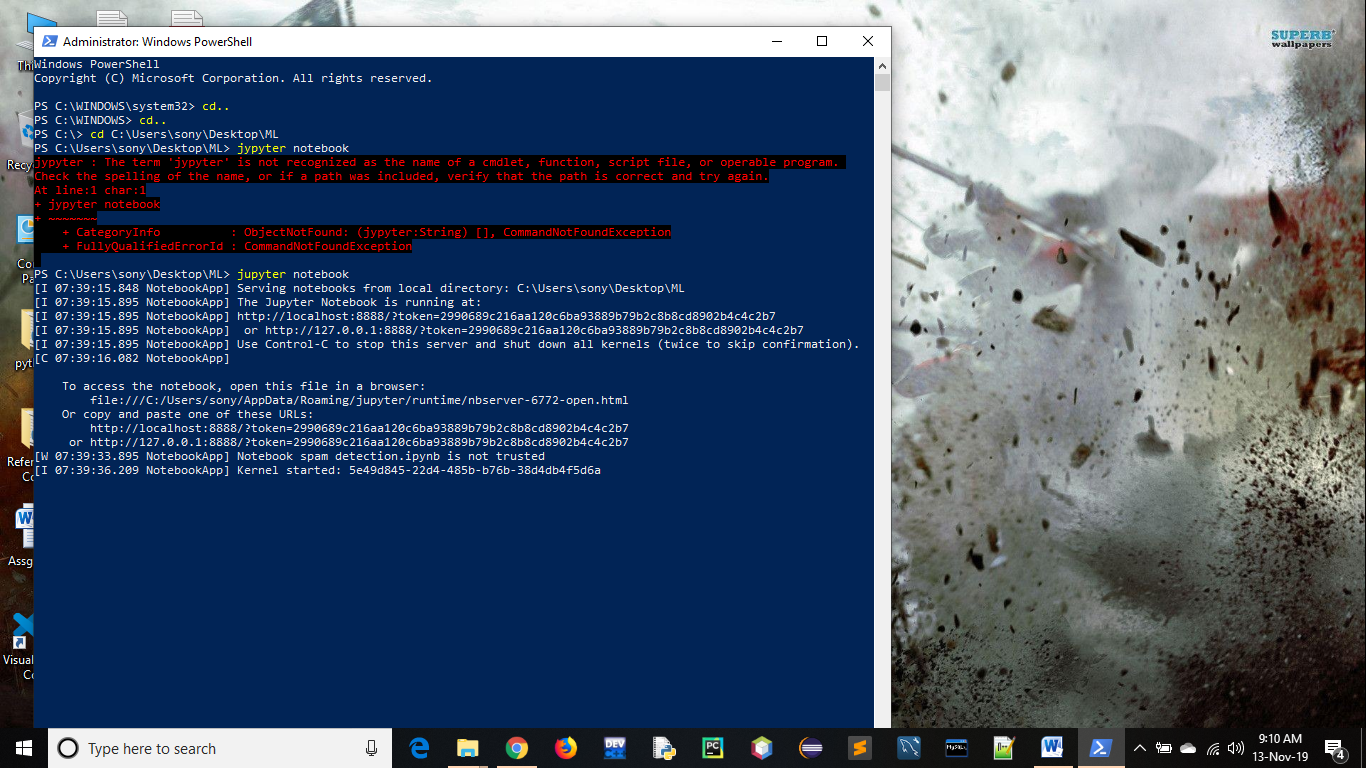
Both the models had similar performance on the test-set except that the SVM has slightly balanced false identifications. I must remind you that the test data was neither used in creating dictionary nor in the training set.

**Chapter 8**

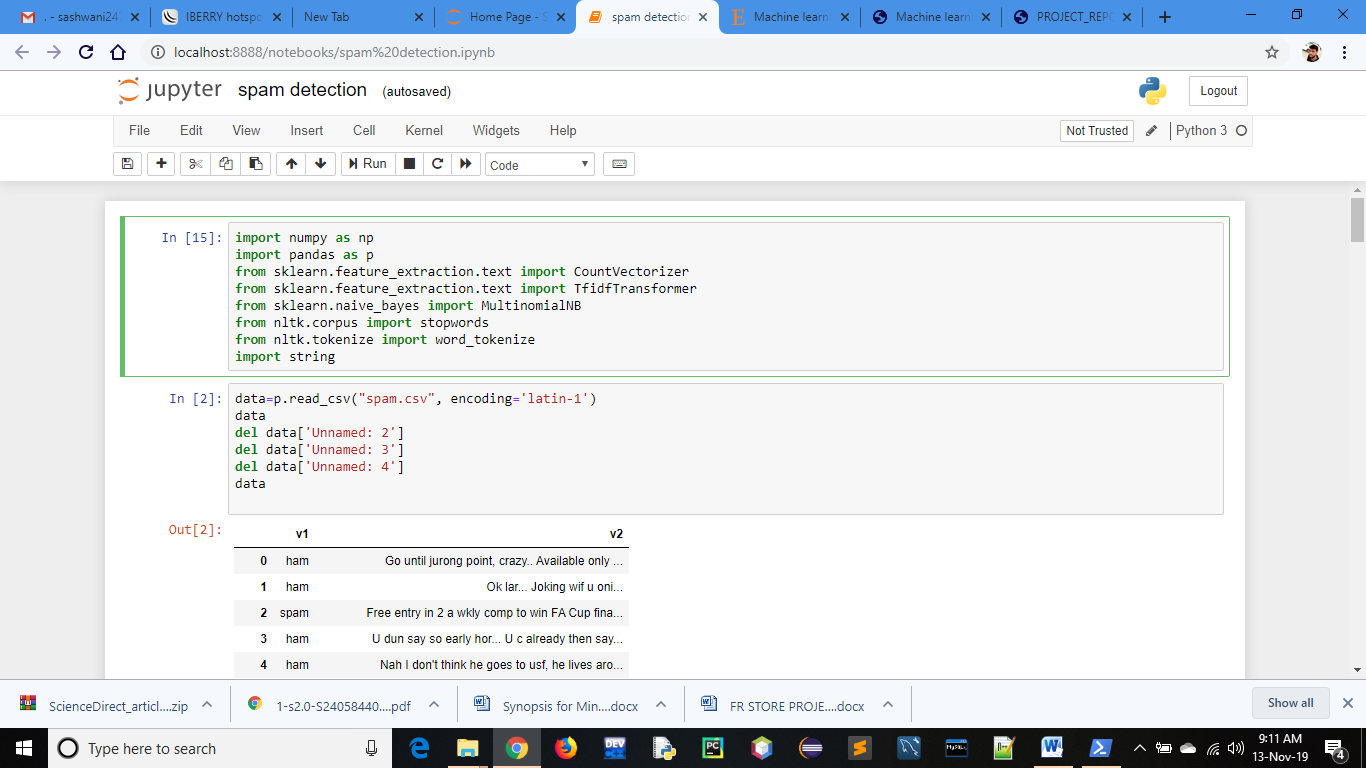
## Steps To Open Project

**Screenshots**

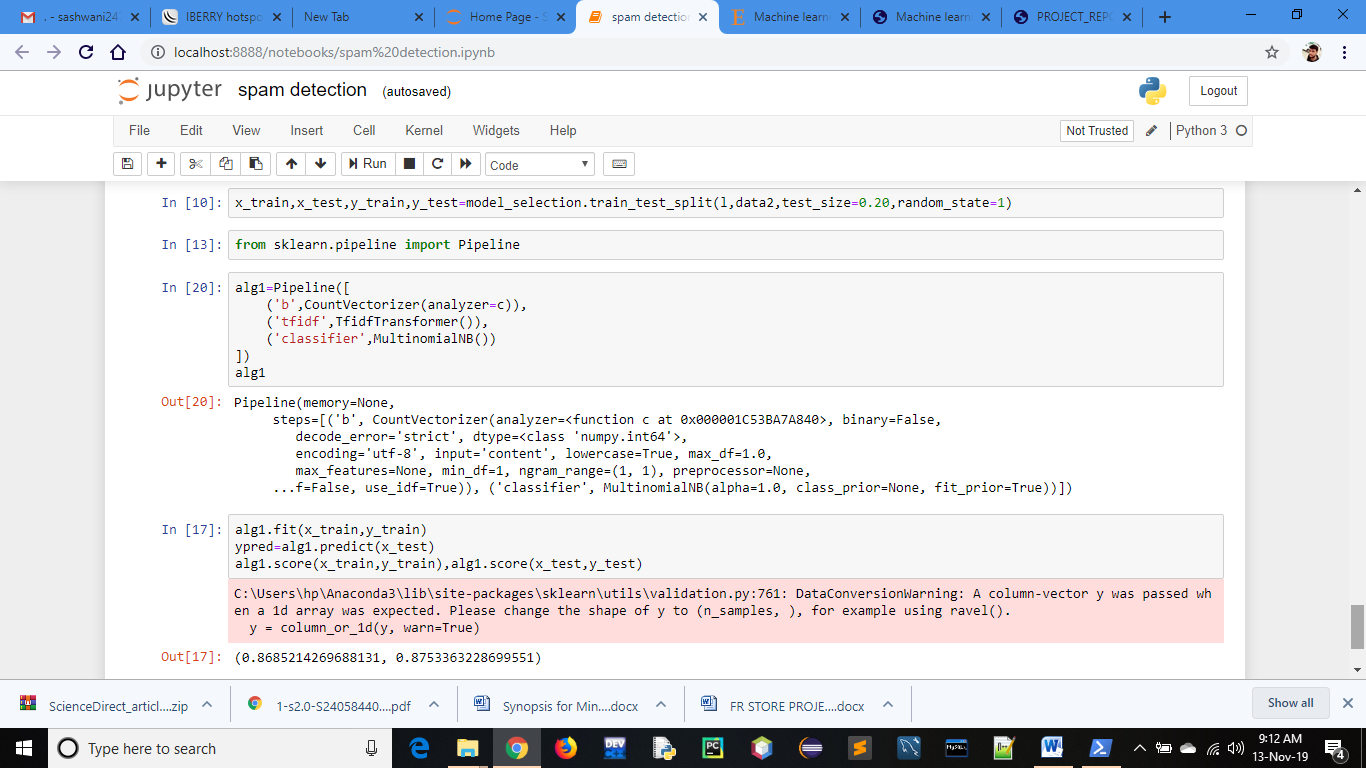
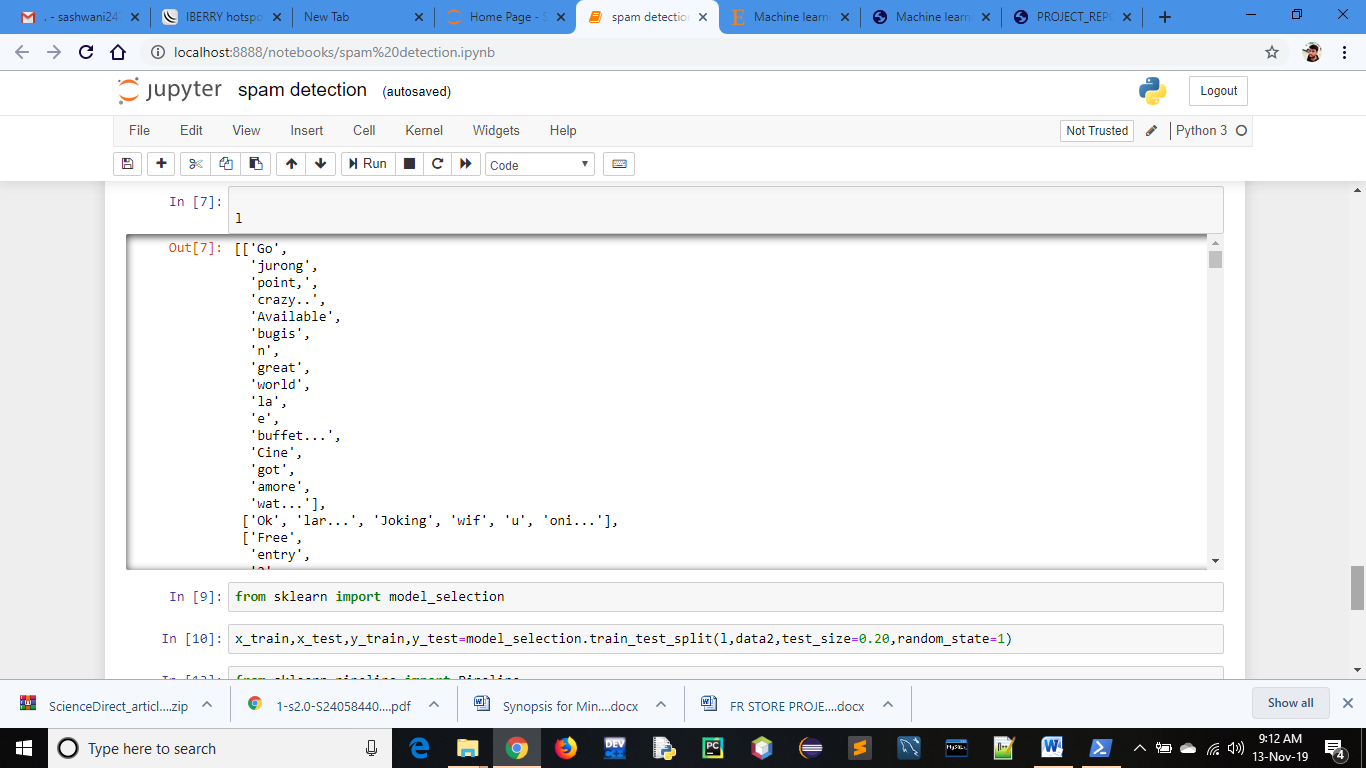
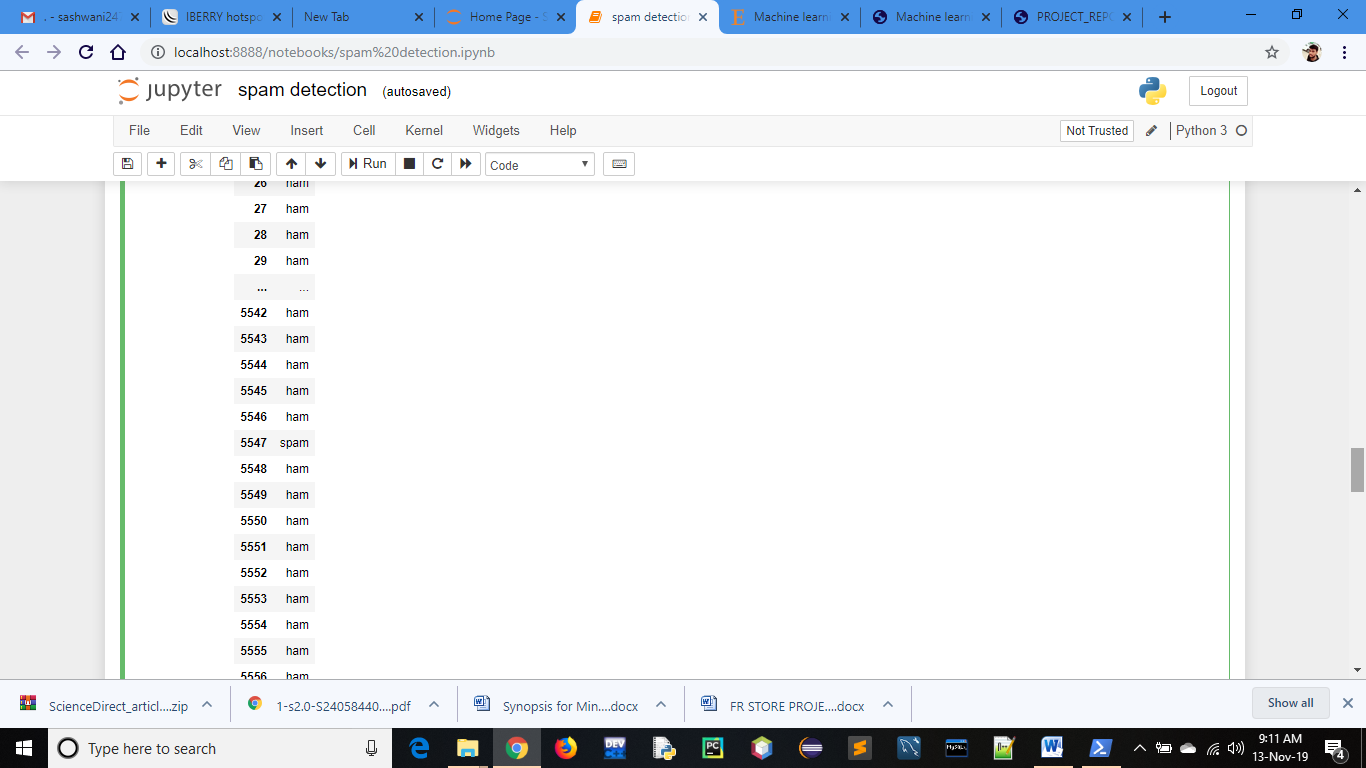
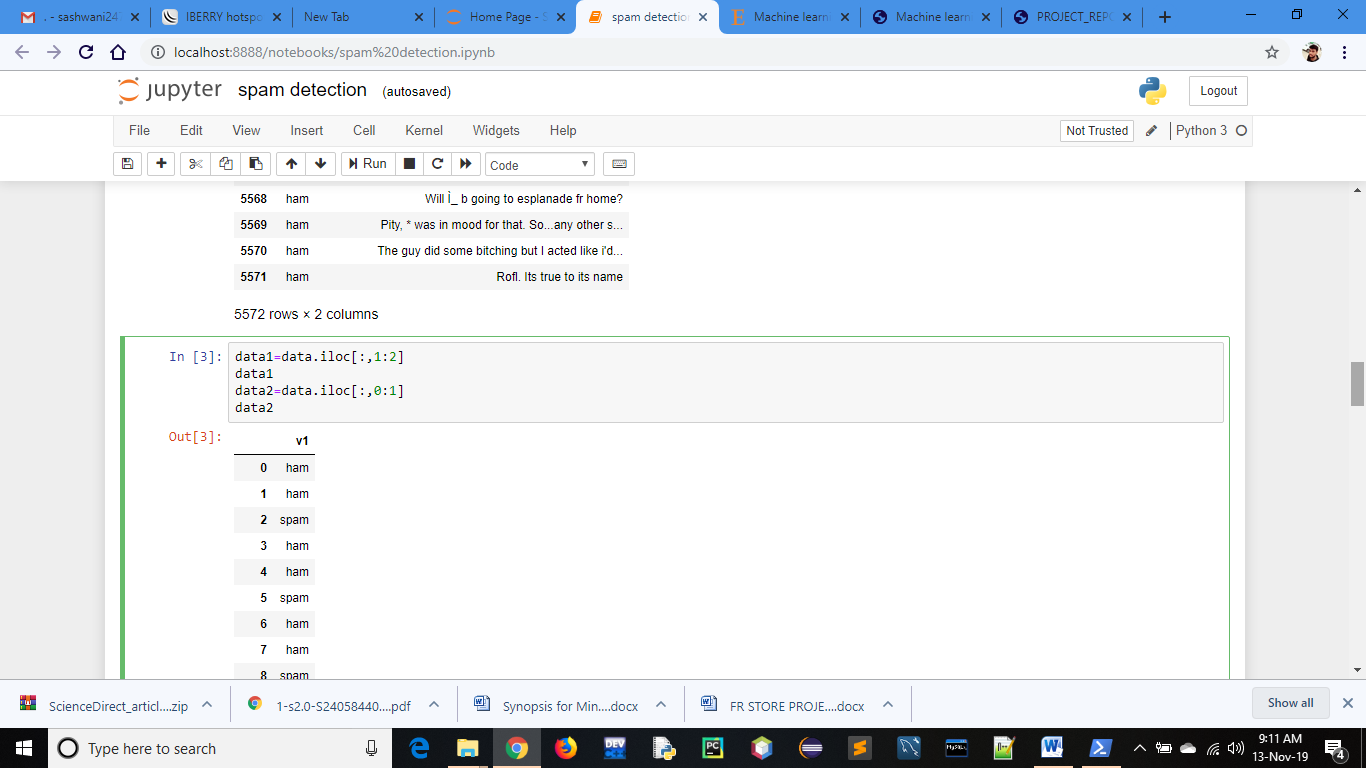
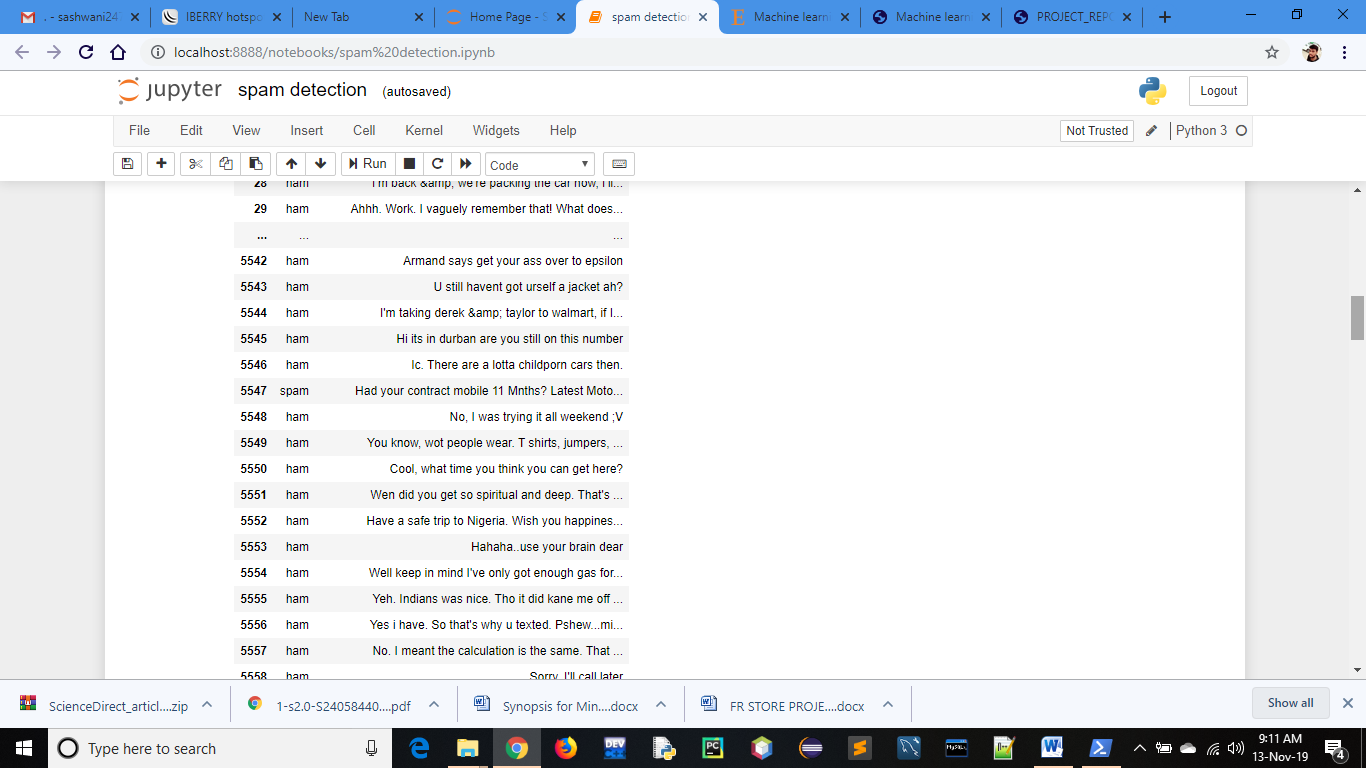
****Open Windows Power Shell as Administrator

****

Goto your Project Directory and type Command Jupyter Notebook

****

1. Jupyter Notebook Will Launch and Open The Project Named Spam ‘spam detection.ipynb’

****