### SPAM DETECTION

### USING ML

#### A PROJECT REPORT

*Submitted by*

**Manavjeet Singh [Reg No:RA2011042010106]**

**Abhishek JHA [Reg No: RA2011042010078]** *Under the Guidance of*

**Dr. JEEVA S**

(Assistant Professor, Department of Computer Science and Engineering)

*In partial fulfillment of the Requirements for the Degree of*

**BACHELOR OF TECHNOLOGY**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**FACULTY OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603203**

#### November 2022



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this project report titled “**SPAM DETECTION USING ML”** is the bonafide work of **Manavjeet singh rathore [Reg No: RA2011042010106] Abhishek Jha [Reg No: RA2011042010078],** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of whicha degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. JEEVA S Dr. M Lakshmi

#### SUPERVISOR HEAD OF THE DEPARTMENT

Assistant Professor Department of Data Science and

Department of Data Science and Business System Business System

Signature of Internal Examiner Signature of External Examiner



Own Work Declaration

Department Of Computer Science and Engineering

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Own Work Declaration Form**

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments Degree/ Course : B.TECH

**Student Name :** Manavjeet Singh Rathore , Abhishek Jha

**Registration Number :** RA2011042010106, RA2011042010078]

**Title of Work :** SPAM DETECTION USING ML

I / We hereby certify that this assessment compiles with the University’s Rules and Regulations relating to Academic misconduct and plagiarism\*\*, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I

/ We have met the following conditions:

* Clearly references / listed all sources as appropriate
* Referenced and put in inverted commas all quoted text (from books, web, etc)
* Given the sources of all pictures, data etc. that are not my own
* Not made any use of the report(s) or essay(s) of any other student(s) either past or present
* Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
* Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalised in accordance with the University policies and regulations.

|  |
| --- |
| **DECLARATION:** |
| I am aware of and understand the University’s policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above. |
| If you are working in a group, please write your registration numbers and sign with the date for every student in  your group. |

Manavjeet singh rathore Abhishek jha

#### ACKNOWLEDGEMENT

We express our humble gratitude to , Vice Chancellor (I/C), SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to , Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her invaluable support.

We wish to thank , Professor & Head, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for her valuable suggestions and encouragement throughout the period of the project work.

We are extremely grateful to our Academic Advisor , Assistant Professor,

Department of Computer Science and Engineering, SRM Institute of Science and Technology, for her great support at all the stages of project work. We would like to convey our thanks to our Panel Head , Associate Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for his inputs during the project reviews.

We register our immeasurable thanks to our Faculty Advisor, , Assistant

Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology,for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **,** Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, for providing me an opportunity to pursue my project under her mentorship. She provided me the freedom and support to explore the research topics of my interest. Her passion for solving real problems and making a difference in the world has always been inspiring.

We sincerely thank staff and students of the Computer Science and Engineering Department, SRM Institute of Science and Technology, for their help during my research. Finally, we would like to thank my parents, our family members and our friends for their unconditional love, constant support and encouragement.

Manavjeet Singh Rathore (RA2011042010106)

Abhishek jha (RA2011042010078)

#### ABSTRACT

The project titled “Email Spam classification” is implemented using the CRISP-DM methodology. You will get to know Business understanding, Data Understanding (Data Description and Exploration), Data Preparation, Modelling, and Evaluation steps. Project is implemented using Python class object-based style.

Email spam detection is done using machine learning algorithms Naive Bayes and SVM (Support vector machines).

Further, it shows the complete program flow for Python-based email spam classifier implementation such as Data Retrieval Flow, Data Visualization Flow, Data Preparation Flow, Modeling, and Evaluation Flow. Also, including the section regarding Data ethics

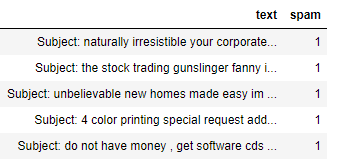
Table of Contents

[](https://pythonbaba.com/email-spam-classification-project-report/)

* [Email Spam classification Project Report:](https://pythonbaba.com/email-spam-classification-project-report/#Email_Spam_classification_Project_Report_CRISP-DM)
  + [Data understanding](https://pythonbaba.com/email-spam-classification-project-report/#Data_understanding)
  + [Data preparation](https://pythonbaba.com/email-spam-classification-project-report/#Data_preparation)
  + [Modeling](https://pythonbaba.com/email-spam-classification-project-report/#Modeling)
  + [Evaluation](https://pythonbaba.com/email-spam-classification-project-report/#Evaluation)
  + [Deployment](https://pythonbaba.com/email-spam-classification-project-report/#Deployment)
  + [Data Ethics](https://pythonbaba.com/email-spam-classification-project-report/#Data_Ethics)
* [Email Spam Classifier: Object-oriented model with program flow](https://pythonbaba.com/email-spam-classification-project-report/#Email_Spam_Classifier_Object-oriented_model_with_program_flow)
  + [Object-oriented model for the Project.](https://pythonbaba.com/email-spam-classification-project-report/#Object-oriented_model_for_the_Project)
  + [Program Flow](https://pythonbaba.com/email-spam-classification-project-report/#Program_Flow)

#### ****Data Description****

The dataset contains two columns. The total corpus of 5728 documents. The descriptive feature consists of text. The target feature consists of two classes ham and spam, the column name is spam. The classes are labeled for each document in the data set and represent our target feature with a binary string-type alphabet of {ham; spam}. Classes are further mapped to integer 0 (ham) and 1 (spam).



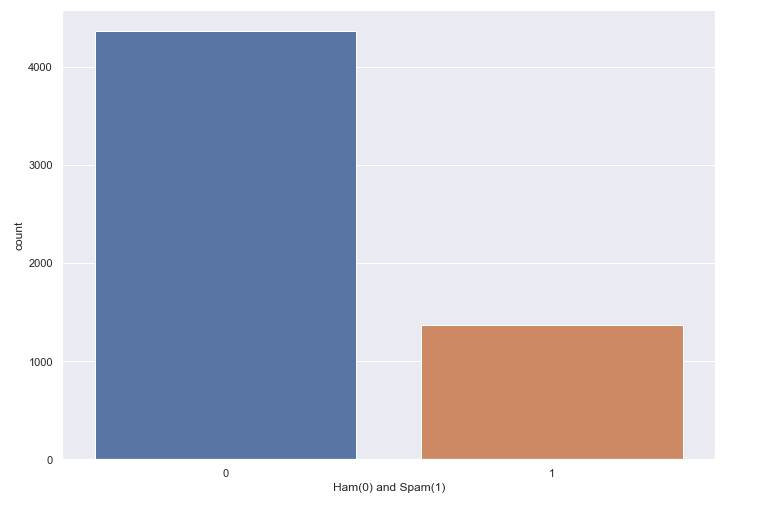
*Figure 1: Data Description*

#### ****Data exploration****

**Spam email percentage in the dataset = 23.88268156424581 %**

**Ham email percentage in the dataset = 76.11731843575419 %**

The bar graph given below depicts the percentage of Ham and Spam emails in the given dataset. The blue bar represents the count of ham emails and the red bar shows the count of spam emails in the dataset.



*Figure 2: Count of ham and spam emails*

**Plotting histogram for email text length of spam vs ham**

Both ham and spam emails are more prevalent for shorter lengths. As spam emails are 24% of the whole data, so its obvious frequency of spam emails is less as compared to ham emails.

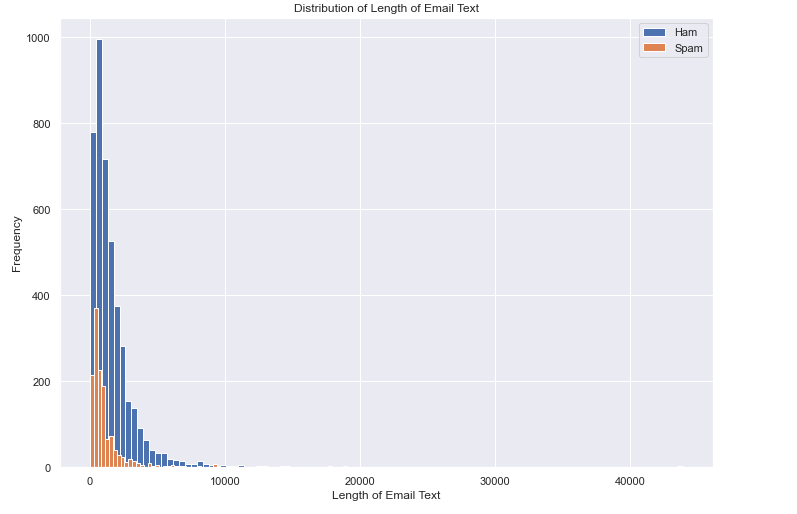


Figure 3: Distribution of Emails length: Ham and Spam

**Word cloud for ham emails**



Figure 4: Word cloud for ham emails

**Word cloud for spam emails**

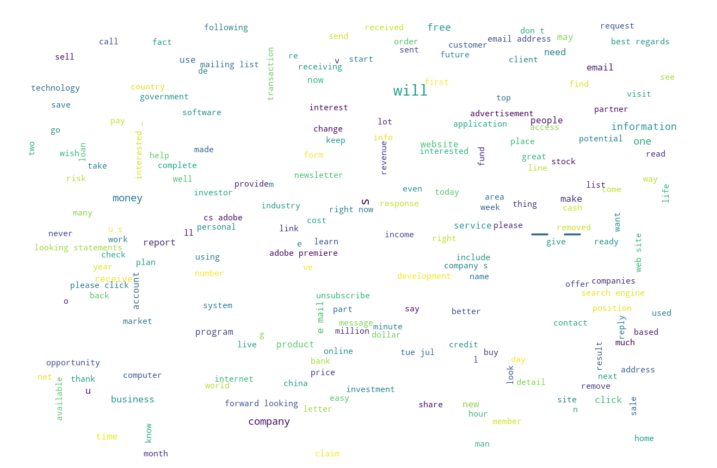


Figure 5: Word cloud for Spam emails

**Distribution of the number of words:**

From the below figure we can see that, there is a spike in spam emails with a smaller number of words, even when our dataset includes 24 percent of spam emails out of total emails.

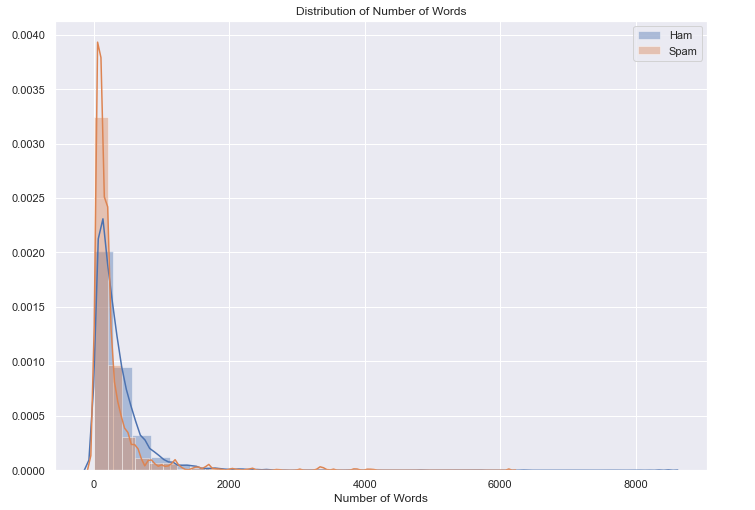


Figure 6: Distribution of the number of words: Ham and Spam emails

**Exploring mean word length:**

 There is not a significant difference in the length of words used by ham and spam emails.

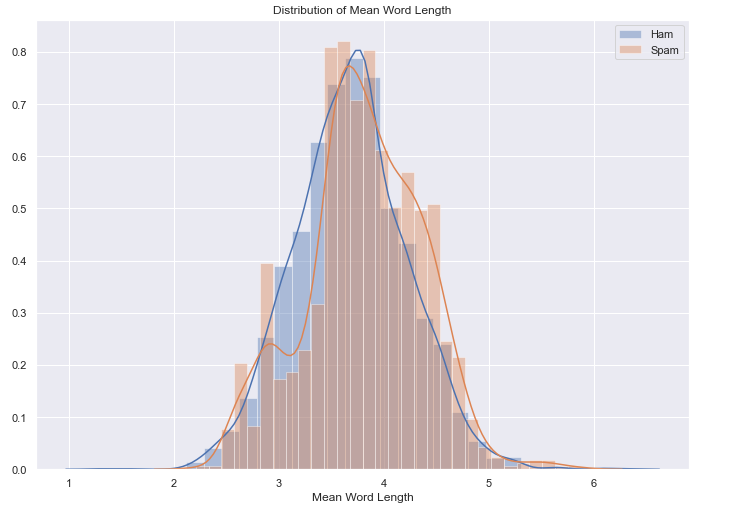


Figure 7: Mean word length Plot: Ham and Spam emails

**Distribution of Stop-word ratio:**

* All Spam emails contain stop words with a mean of 0.281
* All Ham emails contain stop words with a mean of 0.278
* But we can see from the graph, spam email contains high stop words ratio as compared to ham emails.

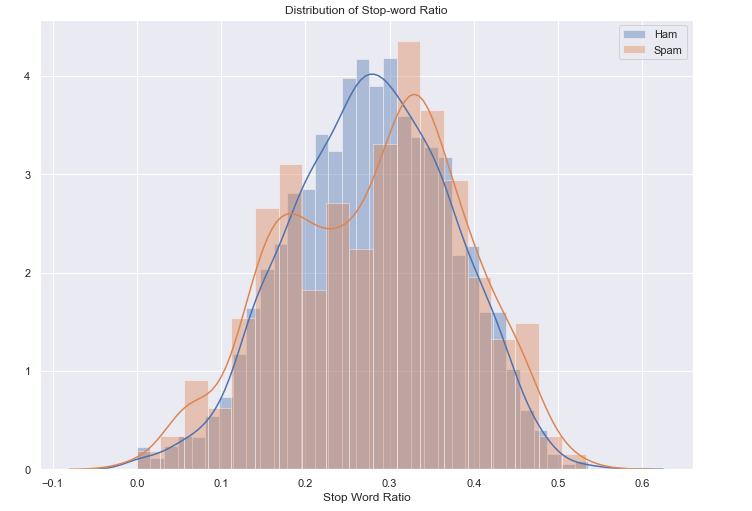


Figure 8: Distribution of Stop-word ratio: Ham and Spam emails

**Most Frequent word analysis using stop words:**

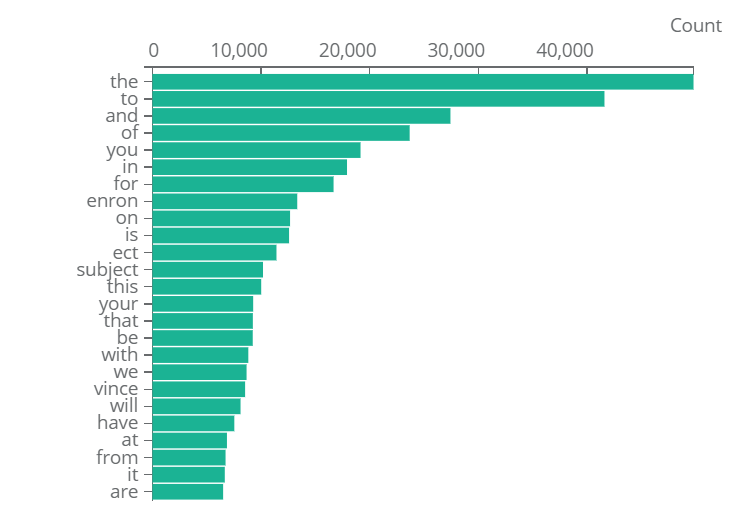


Figure 9: Frequent word analysis using stop words

**Most Frequent word analysis without using stop words:**

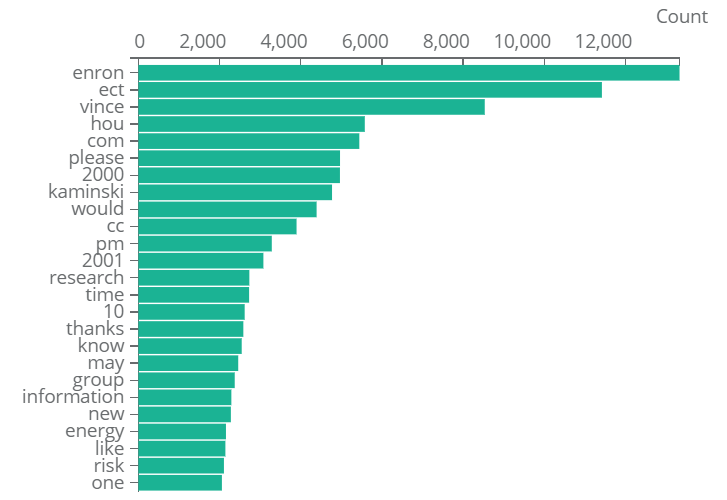


Figure 10: Frequent word analysis without using stop words

### ****Data preparation****

The following steps we used for data preparation.

* Identifying Missing values.
* Converting all text to lower case.
* Performing tokenization.
* Removing Stop words.
* Labelling classes: ham/spam: {0;1}
* Splitting Train and Test Data: 80% and 20%.

### ****Modeling****

**Feature representation**: Using word embedding technique CountVectorizer.

**Models used**: **Naive Bayes and SVM**. Email spam classification done using traditional machine learning techniques comprise Baive Bayes and SVM (support vector machines), due to not having sufficient hardware resources, takes less time to train. Also, not opting for neural algorithms due to less data and computing resources.

**Reason for choosing SVM and Naïve Bayes**: Both are good at handling large number of features; in the case of text classification each word is a feature and we have thousands of words based on the vocabulary of the corpus.  SVM works best with high dimensional data, a vocabulary with 1000 words means each text in the corpus will be represented with a vector of 1000 dimension.

When we have a sufficient number of features, both SVM and Naïve Bayes can work with less data as well.

Naïve Bayes does not suffer from curse-of-dimensionality because it treats all features as independent of one another. Also, one of the benefits of features being independent is: For example, most spam emails contain words such as money and investment, etc, but it is not necessary that all the mails containing both words money and investment are considered to be spam.

**Feature representation**: Word embeddings can be broadly classified into two categories: Frequency and prediction-based. I have chosen a count vector that shows the count of occurrence of a feature in the given document, thus it is a matrix of document vs vocabulary (containing all the features as a column).

In our case, the size of the Count vector matrix is 5728 x 20114, where 5728 represents the number of documents in the corpus and 20114 represents the number of features in the vocabulary.

**Splitting Training and Testing Data**: Splitting the data into training and test datasets, where training data contains 80 percent and test data contains 20 percent.

**Applying model SVM and Naïve Bayes:**I trained the model for both SVM and Naive without tuning hyperparameters as I got results with default parameter settings.

### ****Evaluation****

**Naïve Bayes Result on Test dataset:**

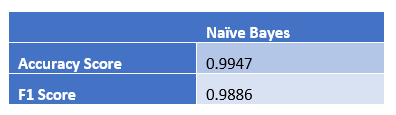


Figure 11: Accuracy and F1 score

**Confusion matrix for Naïve Bayes without normalization**

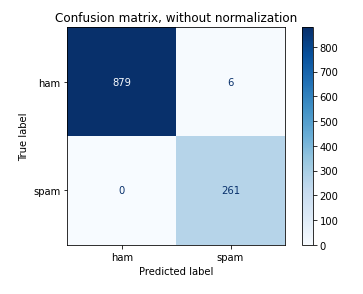


Figure 12: Confusion matrix for Naive Bayes without normalization

**Confusion matrix for Naïve Bayes with normalization**

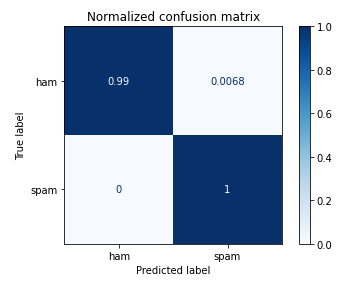


Figure 13: Confusion matrix for Naive Bayes with normalization

**Naïve Bayes ROC Curve**

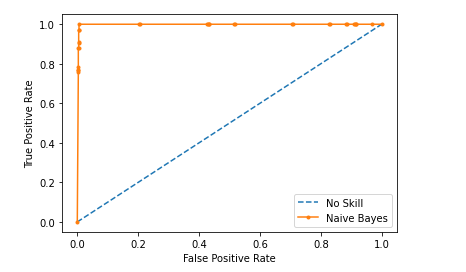


Figure 14: Naive Bayes: ROC Curve

**SVM Result on Test data set**

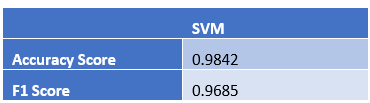
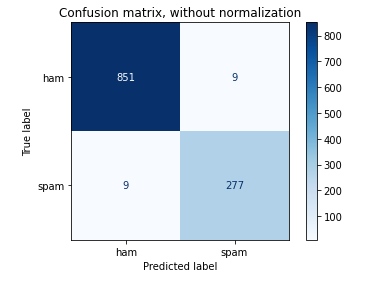


Figure 15: Accuracy and F1 score for SVM Model

**Confusion matrix for SVM without normalization**



Confusion matrix for SVM without normalization

**Confusion matrix for SVM with normalization**

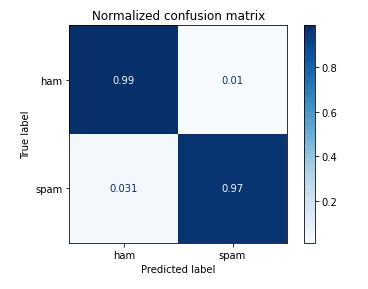


Figure 17: Confusion matrix for SVM with normalization

**ROC curve for SVM model**

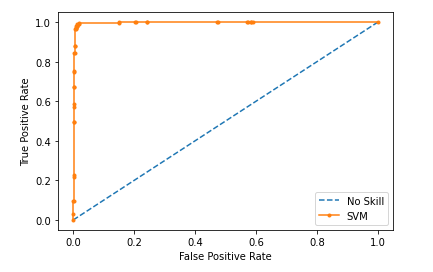


Figure 18: SVM: ROC Curve

**Results**

Comparing both Naïve Bayes and SVM, I found that Naïve Bayes has 1% improvement over the SVM model when the result compared to test data set.

**Deployment**

A supportive tool using a browser plugin or API can be built for companies running their own email servers so that they can keep a check on emails and can identify and flag spam emails. Such a supportive tool can be used in conjunction with existing email service providers as well.

### ****Data Ethics****

There are many ethical and legal issues that can really take a toll on designing such models. Bank and Investment organizations that run their own email servers have confidential emails and identity information related to customers. Using such confidential information for the predictive analysis required to safeguard the client data with the care of a professional fiduciary. Need to protect the customer data from both intentional and inadvertent disclosure, also protecting it from misuse.

Also, while implementation it is possible to generate false positive, which means the emails which are not spam fall into the spam category. An important piece of information a company can miss if the user’s legit email is marked as spam. A client or user who is a loyal customer, his email can be marked spam, which is an ethical issue.

## ****Email Spam Classifier: Object-oriented model with program flow****

### Object-oriented model for the Project.

We have one parent class and three child class which inherit data and functions from the parent class.  
**Parent class**: data\_read\_write **Child class**: generate\_word\_cloud |data\_cleaning| apply\_ embedding\_and\_model

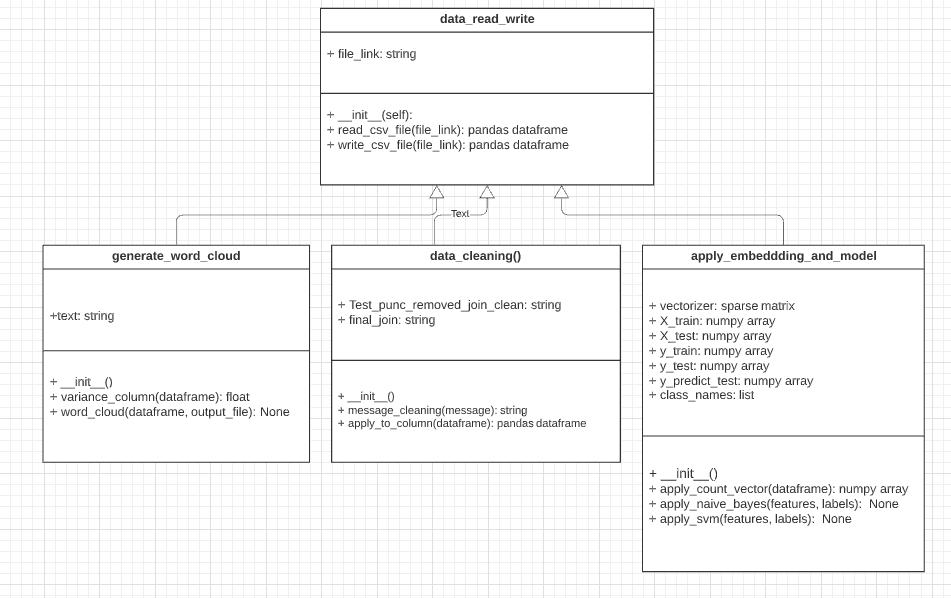


Figure 19: UML diagram for the Project

### ****Program Flow****

We have four classes in total. One is the parent and three child classes.  
• Parent class: data\_read\_write  
• Child class: generate\_word\_cloud  
• Child class: data\_cleaning  
• Child class: apply\_embedding\_and\_model

#### ****Data Retrieval Flow****

Creating object on the parent class: data\_read\_write

data\_obj = data\_read\_write("emails.csv")

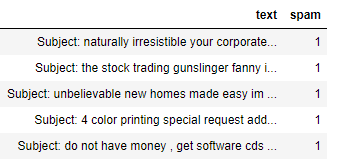
We create an object by initializing it using the dataset file emails.csv which is passed to the constructor. It will read the name of the file and store it in file\_link variable which is a string type and return the object reference. We can now  
read the CSV file by calling read\_csv\_file() function defined in our parent class data\_read\_write by accessing it through the object. The function will return the content of the file as pandas dataframe

data\_frame = data\_obj.read\_csv\_file()

Now, we can print the top 5 rows of our dataframe

data\_frame.head()

The below table shows the text and spam, as two columns, the text feature is the descriptive feature which contains the email: subject and body content. The spam column contains two ham and spam class labels, where 0 refers to ham and 1 refers to spam

Data Description

#### ****Data Visualization Flow****

The below code snippet separates the ham and spam emails and counts the max word length used in any spam or ham  
email. For ham email, the maximum number of words used in an email is 8479 and for spam email, the maximum word used is 6131

#data\_frame['spam']==0

data\_frame[data\_frame['spam']==0].text.values

ham\_words\_length = [len(word\_tokenize(title)) **for** title **in** data\_frame[data\_frame['spam']==0].text.values]

spam\_words\_length = [len(word\_tokenize(title)) **for** title **in** data\_frame[data\_frame['spam']==1].text.values]

print(max(ham\_words\_length))

print(max(spam\_words\_length))

From the above code snippet, we get the number of words for each document for the spam and ham category. Now in the below code snippet, we plotted a histogram that shows the distribution of the number of words.

#There is spike in spam emails with less number of words

#Even when our dataset include 24 percent of spam emails out of total emails-

#Looks like Spam emails have less words as compared to ham emails

sns.set(rc={'figure.figsize':(11.7,8.27)})

ax = sns.distplot(ham\_words\_length, norm\_hist = **True**, bins = 30, label = 'Ham')

ax = sns.distplot(spam\_words\_length, norm\_hist = **True**, bins = 30, label = 'Spam')

#ham\_words\_length.plot(bins=100, kind='hist',label = 'Ham')

#spam\_words\_length.plot(bins=100, kind='hist',label = 'Spam')

plt.title('Distribution of Number of Words')

plt.xlabel('Number of Words')

plt.legend()

plt.show()

Then we plotted histogram for distribution of mean word length used in spam and ham category using the below code snippet.

**def** mean\_word\_length(x):

word\_lengths = np.array([])

**for** word **in** word\_tokenize(x):

word\_lengths = np.append(word\_lengths, len(word))

**return** word\_lengths.mean()

ham\_meanword\_length = data\_frame[data\_frame['spam']==0].text.apply(mean\_word\_length)

spam\_meanword\_length = data\_frame[data\_frame['spam']==1].text.apply(mean\_word\_length)

sns.distplot(ham\_meanword\_length, norm\_hist = **True**, bins = 30, label = 'Ham')

sns.distplot(spam\_meanword\_length , norm\_hist = **True**, bins = 30, label = 'Spam')

plt.title('Distribution of Mean Word Length')

plt.xlabel('Mean Word Length')

plt.legend()

plt.show()

#There is not a significant difference for the length of words used by ham and spam emails

Then we plotted histogram for distribution of stop word ratio in each mail used in spam and ham category using the below code snippet.

#Checking ratio of stop words

#Both spam and ham email contain stopwords

#All Spam emails contain stop words with a mean of 0.281

#All Ham emails contain stop words with a mean of 0.278

#But we can see from the graph, spam email contain high stop words ratio as compared to ham emails.

**from** *nltk.corpus* **import** stopwords

stop\_words = set(stopwords.words('english'))

**def** stop\_words\_ratio(x):

num\_total\_words = 0

num\_stop\_words = 0

**for** word **in** word\_tokenize(x):

**if** word **in** stop\_words:

num\_stop\_words += 1

num\_total\_words += 1

**return** num\_stop\_words/num\_total\_words

ham\_stopwords = data\_frame[data\_frame['spam']==0].text.apply(stop\_words\_ratio)

spam\_stopwords = data\_frame[data\_frame['spam']==1].text.apply(stop\_words\_ratio)

sns.distplot(ham\_stopwords, norm\_hist = **True**, label = 'Ham')

sns.distplot(spam\_stopwords, label = 'Spam')

print('Ham Mean: {:.3f}'.format(ham\_stopwords.values.mean()))

print('Spam Mean: {:.3f}'.format(spam\_stopwords.values.mean()))

plt.title('Distribution of Stop-word Ratio')

plt.xlabel('Stop Word Ratio')

plt.legend()

The next code snippet shows the histogram for the count of ham and spam emails present in our document and also calculate the percentage of the number of spam and ham email present.

# Let's divide the messages into spam and ham

ham = data\_frame[data\_frame['spam']==0]

spam = data\_frame[data\_frame['spam']==1]

spam['length'].plot(bins=60, kind='hist')

ham['length'].plot(bins=60, kind='hist')

data\_frame['Ham(0) and Spam(1)'] = data\_frame['spam']

print( 'Spam percentage =', (len(spam) / len(data\_frame) )\*100,"%")

print( 'Ham percentage =', (len(ham) / len(data\_frame) )\*100,"%")

sns.countplot(data\_frame['Ham(0) and Spam(1)'], label = "Count")

Spam percentage = 23.88268156424581 %

Ham percentage = 76.11731843575419 %

Now, we will generate a word cloud for both ham and spam emails separately using the below code. First, it creates the object for a child class generate word cloud then calling the function word cloud ham() which take two arguments, column and image filename need to be generated for the word cloud.

word\_cloud\_obj = generate\_word\_cloud()

word\_cloud\_obj.word\_cloud(ham["text"], "ham\_word\_cloud.png")

word\_cloud\_obj.word\_cloud(spam["text"], "spam\_word\_cloud.png")

Below can see a complete class, data, and methods of the child class generate\_word\_cloud child class generate\_word\_cloud

#Child Class for Data\_read\_write

**class** generate\_word\_cloud(data\_read\_write):

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

**pass**

#Child own Function

**def** variance\_column(self, data):

**return** variance(data)

#Polymorphism

**def** word\_cloud(self, data\_frame\_column, output\_image\_file):

text = " ".join(review **for** review **in** data\_frame\_column)

stopwords = set(STOPWORDS)

stopwords.update(["subject"])

wordcloud = WordCloud(width = 1200, height = 800, stopwords=stopwords, max\_font\_size = 50,

margin=0, background\_color = "white").generate(text)

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.show()

wordcloud.to\_file(output\_image\_file)

**return**

**Data Preparation Flow**

After generating word cloud, we need to perform data cleaning steps. In the below code snippet, we need to first create an object on child class data\_cleaning and then calling function apply\_to\_column using the created object. The function takes input as text feature which is a data frame column and returns the processed data frame, and stored in  
another column named clean\_text.

data\_clean\_obj = data\_cleaning()

data\_frame['clean\_text'] = data\_clean\_obj.apply\_to\_column(data\_frame['text'])

We can see that the data\_cleaning class consists of two methods apply\_to\_column which calls another function message\_cleaning which further removes stop words, remove punctuation, and do necessary data processing steps.

#Child Class for Data\_read\_write

**class** data\_cleaning(data\_read\_write):

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

**pass**

**def** message\_cleaning(self, message):

Test\_punc\_removed = [char **for** char **in** message **if** char not **in** string.punctuation]

Test\_punc\_removed\_join = ''.join(Test\_punc\_removed)

Test\_punc\_removed\_join\_clean = [word **for** word **in** Test\_punc\_removed\_join.split()

**if** word.lower() not **in** stopwords.words('english')]

final\_join = ' '.join(Test\_punc\_removed\_join\_clean)

**return** final\_join

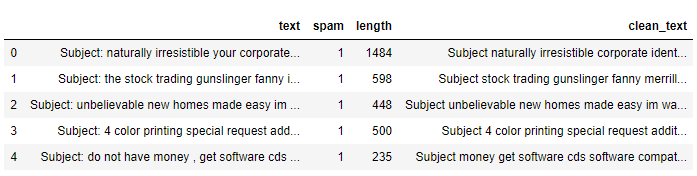
**def** apply\_to\_column(self, data\_column\_text):

data\_processed = data\_column\_text.apply(self.message\_cleaning)

**return** data\_processed

We can now check the additional columns created by using the pandas head function on the data frame

data\_obj.data\_frame.head()

Dataframe

#### ****Modeling and Evaluation Flow****

Then applying countvectorizer on the processed data column clean text. It first creates an object for child class apply\_embedding\_and\_model. Then calling the function apply count vector which takes the column input and returns the countvectorizer sparse matrix.

#APPLY COUNT VECTORIZER TO OUR MESSAGES LIST

# Define the cleaning pipeline we defined earlier

#vectorizer = CountVectorizer()

cv\_object = apply\_embeddding\_and\_model()

spamham\_countvectorizer = cv\_object.apply\_count\_vector(data\_frame['clean\_text'])

Now we need to separate descriptive and target features from our data set.

#Separating Descriptive and Target Feature

X = spamham\_countvectorizer

label = data\_frame['spam'].values

y = label

Now, we need to call the function apply\_naive\_bayes using the object created for child class apply\_embedding\_and\_model

cv\_object.apply\_naive\_bayes(X,y)

The apply\_naive\_bayes function performs the below mention jobs.  
• It split the training and test set to 80% and 20% ratio.  
• Apply the Naive Bayes on training data.  
• Predict the outcome of the test dataset.  
• Calculate confusion matrix without normalization and with normalization.  
• Calculate Accuracy, F1, Recall, and Precision.  
• Plot ROC curve.

Now, we need to call the function apply\_svm using the object created for child class apply\_embedding\_and\_model

cv\_object.apply\_svm(X,y)

The apply\_svm function performs the below mention jobs.  
• It split the training and test set to 80% and 20% ratio.  
• Apply the Naive Bayes on training data.  
• Predict the outcome of the test dataset.  
• Calculate confusion matrix without normalization and with normalization.  
• Calculate Accuracy, F1, Recall, and Precision.  
• Plot ROC curve.

Given below is the code snippet for child class apply embedding and model which shows all the data variables and three methods used in the class.

#Child Class for Data\_read\_write

**class** apply\_embeddding\_and\_model(data\_read\_write):

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

**pass**

**def** apply\_count\_vector(self, v\_data\_column):

vectorizer = CountVectorizer(min\_df=2,analyzer = "word",tokenizer = **None**,preprocessor = **None**,stop\_words = **None**)

**return** vectorizer.fit\_transform(v\_data\_column)

**def** apply\_naive\_bayes(self, X, y):

#DIVIDE THE DATA INTO TRAINING AND TESTING PRIOR TO TRAINING

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

#Training model

NB\_classifier = MultinomialNB()

NB\_classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_predict\_test = NB\_classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_predict\_test)

#sns.heatmap(cm, annot=True)

#Evaluating Model

print(classification\_report(y\_test, y\_predict\_test))

print("test set")

print("\nAccuracy Score: " + str(metrics.accuracy\_score(y\_test, y\_predict\_test)))

print("F1 Score: " + str(metrics.f1\_score(y\_test, y\_predict\_test)))

print("Recall: " + str(metrics.recall\_score(y\_test, y\_predict\_test)))

print("Precision: " + str(metrics.precision\_score(y\_test, y\_predict\_test)))

class\_names = ['ham', 'spam']

titles\_options = [("Confusion matrix, without normalization", **None**),

("Normalized confusion matrix", 'true')]

**for** title, normalize **in** titles\_options:

disp = plot\_confusion\_matrix(NB\_classifier, X\_test, y\_test,

display\_labels=class\_names,

cmap=plt.cm.Blues,

normalize=normalize)

disp.ax\_.set\_title(title)

print(title)

print(disp.confusion\_matrix)

plt.show()

# generate a no skill prediction (majority class)

ns\_probs = [0 **for** \_ **in** range(len(y\_test))]

# predict probabilities

lr\_probs = NB\_classifier.predict\_proba(X\_test)

# keep probabilities for the positive outcome only

lr\_probs = lr\_probs[:, 1]

# calculate scores

ns\_auc = roc\_auc\_score(y\_test, ns\_probs)

lr\_auc = roc\_auc\_score(y\_test, lr\_probs)

# summarize scores

print('No Skill: ROC AUC=%.3f' % (ns\_auc))

print('Naive Bayes: ROC AUC=%.3f' % (lr\_auc))

# calculate roc curves

ns\_fpr, ns\_tpr, \_ = roc\_curve(y\_test, ns\_probs)

lr\_fpr, lr\_tpr, \_ = roc\_curve(y\_test, lr\_probs)

# plot the roc curve for the model

yplot.plot(ns\_fpr, ns\_tpr, linestyle='--', label='No Skill')

pyplot.plot(lr\_fpr, lr\_tpr, marker='.', label='Naive Bayes')

# axis labels

pyplot.xlabel('False Positive Rate')

pyplot.ylabel('True Positive Rate')

# show the legend

pyplot.legend()

# show the plot

pyplot.show()

**return**

**def** apply\_svm(self, X, y):

#DIVIDE THE DATA INTO TRAINING AND TESTING PRIOR TO TRAINING

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

#Training model

#'linear', 'poly', 'rbf'

params = {'kernel': 'linear', 'C': 2, 'gamma': 1}

svm\_cv = svm.SVC(C=params['C'], kernel=params['kernel'], gamma=params['gamma'], probability=**True**)

svm\_cv.fit(X\_train, y\_train)

# Predicting the Test set results

y\_predict\_test = svm\_cv.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_predict\_test)

#sns.heatmap(cm, annot=True)

#Evaluating Model

print(classification\_report(y\_test, y\_predict\_test))

print("test set")

print("\nAccuracy Score: " + str(metrics.accuracy\_score(y\_test, y\_predict\_test)))

print("F1 Score: " + str(metrics.f1\_score(y\_test, y\_predict\_test)))

print("Recall: " + str(metrics.recall\_score(y\_test, y\_predict\_test)))

print("Precision: " + str(metrics.precision\_score(y\_test, y\_predict\_test)))

class\_names = ['ham', 'spam']

titles\_options = [("Confusion matrix, without normalization", **None**),

("Normalized confusion matrix", 'true')]

**for** title, normalize **in** titles\_options:

disp = plot\_confusion\_matrix(svm\_cv, X\_test, y\_test,

display\_labels=class\_names,

cmap=plt.cm.Blues,

normalize=normalize)

disp.ax\_.set\_title(title)

print(title)

print(disp.confusion\_matrix)

plt.show()

# generate a no skill prediction (majority class)

ns\_probs = [0 **for** \_ **in** range(len(y\_test))]

# predict probabilities

lr\_probs = svm\_cv.predict\_proba(X\_test)

# keep probabilities for the positive outcome only

lr\_probs = lr\_probs[:, 1]

# calculate scores

\_auc = roc\_auc\_score(y\_test, ns\_probs)

lr\_auc = roc\_auc\_score(y\_test, lr\_probs)

# summarize scores

print('No Skill: ROC AUC=%.3f' % (ns\_auc))

print('SVM: ROC AUC=%.3f' % (lr\_auc))

# calculate roc curves

ns\_fpr, ns\_tpr, \_ = roc\_curve(y\_test, ns\_probs)

lr\_fpr, lr\_tpr, \_ = roc\_curve(y\_test, lr\_probs)

# plot the roc curve for the model

pyplot.plot(ns\_fpr, ns\_tpr, linestyle='--', label='No Skill')

pyplot.plot(lr\_fpr, lr\_tpr, marker='.', label='SVM')

# axis labels

pyplot.xlabel('False Positive Rate')

pyplot.ylabel('True Positive Rate')

# show the legend

pyplot.legend()

# show the plot

pyplot.show()

**return**

Base code and source page

Text

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, email, Teams

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

Text

Description automatically generated with medium confidence Chart, histogram

Description automatically generated



Chart, histogram

Description automatically generated

Application

Description automatically generated with low confidence

Chart, histogram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Chart, histogram

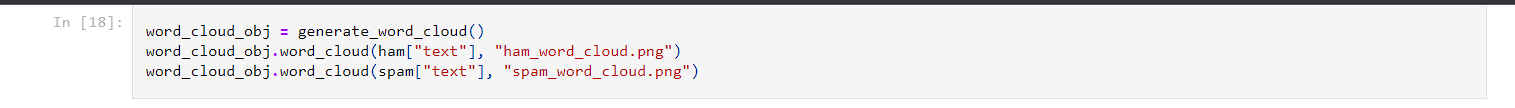
Description automatically generated

Graphical user interface, text, application

Description automatically generated

Chart, bar chart

Description automatically generated



Text, letter

Description automatically generated

A picture containing calendar

Description automatically generated

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Chart

Description automatically generated

Chart

Description automatically generated

No Skill: ROC AUC=0.500

Naive Bayes: ROC AUC=0.998

Chart, scatter chart

Description automatically generated

In [26]:

cv\_object**.**apply\_svm(X,y)

precision recall f1-score support

0 0.99 0.99 0.99 901

1 0.98 0.98 0.98 245

accuracy 0.99 1146

macro avg 0.99 0.98 0.99 1146

weighted avg 0.99 0.99 0.99 1146

test set

Accuracy Score: 0.9904013961605584

F1 Score: 0.9775051124744377

Recall: 0.9755102040816327

Precision: 0.9795081967213115

Confusion matrix, without normalization

[[896 5]

[ 6 239]]

Normalized confusion matrix

[[0.99445061 0.00554939]

[0.0244898 0.9755102 ]]

Chart

Description automatically generated

Chart

Description automatically generated

No Skill: ROC AUC=0.500

SVM: ROC AUC=0.998

Chart, scatter chart

Description automatically generated

REFERENCES:

• Basu, Atreya Watters, Carolyn Author, Michael. (2003). Support Vector Machines for Text     Categorization. 103. 10.1109/HICSS.2003.1174243.

• Zhang, Wei Gao, Feng. (2011). An Improvement to Naive Bayes for Text

Classification.       Procedia Engineering. 15. 2160-2164. 10.1016/j.proeng.2011.08.404.

• <https://www.kaggle.com/venky73/spam-mails-dataset>

* GEEK OF GEEK
* CODECHEF