Email-Spam-Detection

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

SHAIK IRFAN [RA2011026010080]
ARAVIND KRISHNAN R [RA2011026010077]
MURALI KRISHNA [RA2011026010086]

Under the guidance of

Dr. M. S. Abirami

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM

INSTITUTE OF SCIENCE & TECHNOLOGY

Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"EMAIL-SPAM-DETECTION"** is the bona fide work of **SHAIK IRFAN (RA2011026010080),ARAVIND KRISHNAN R(RA2011026010077), MURALI KRISHNA (RA2011026010086)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

11/5/23

SIGNATURE

Dr. M. S. Abirami
GUIDE
Assistant Professor
Department of Computing Technologies

SIGNATURE

Dr. R.Annie Uthra
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computational Intelligence

Email-Spam-Detection

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

SHAIK IRFAN [RA2011026010080]
ARAVIND KRISHNAN R [RA2011026010077]
MURALI KRISHNA [RA2011026010086]

Under the guidance of

Dr. M. S. Abirami

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"EMAIL-SPAM-DETECTION"** is the bona fide work of **SHAIK IRFAN (RA2011026010080),ARAVIND KRISHNAN R(RA2011026010077), MURALI KRISHNA (RA2011026010086)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. M. S. Abirami
GUIDE
Assistant Professor
Department of Computing Technologies

SIGNATURE

Dr. R.Annie Uthra
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computational Intelligence

ABSTRACT

Email spam classification is the process of identifying and categorizing unsolicited and unwanted emails, known as spam, from legitimate and wanted emails, known as ham. The increasing volume of spam in email inboxes has prompted the development of various spam filtering techniques, including rule-based systems, content-based filtering, and machine learning algorithms.

Machine learning algorithms have shown great potential in email spam classification due to their ability to automatically learn and improve from data. These algorithms use statistical models to extract relevant features from the email content and header, such as sender information, subject, and message body. Based on these features, the algorithms can predict the likelihood of an email being spam or ham.

Overall, email spam classification plays a crucial role in ensuring that email users receive only relevant and wanted messages in their inbox, while keeping unsolicited and potentially harmful emails out.

TABLE OF CONTENTS

1. Problem Statement

Email has become an essential communication tool for both personal and business purposes. However, the increasing amount of spam emails in our inboxes has become a significant concern, with some estimates suggesting that up to 45% of emails sent worldwide are spam.

Email spam refers to unsolicited emails that are sent in bulk to a large number of recipients. These emails often contain malicious links or attachments, phishing scams, or unwanted advertisements, causing inconvenience and potentially even harm to the recipients.

To address this issue, email spam detection has become increasingly important. Email spam detection involves identifying and categorizing incoming emails as either spam or legitimate, also known as ham. This is accomplished by utilizing a variety of techniques, including rule-based systems, content-based filtering, and machine learning algorithms.

2. LITERATURE REVIEW

1. Almeida, T. A., & Hidalgo, J. M. G. (2011). Contributions to the study of SMS spam filtering: new collection and results. Expert Systems with Applications, 38(10), 12460-12468.

   This study focuses on SMS spam filtering and presents a new collection of SMS messages for use in spam filtering. The authors compare the performance of different machine learning algorithms and feature selection techniques for SMS spam filtering.

2. Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A Bayesian approach to filtering junk e-mail. In AAAI workshop on learning for text categorization (Vol. 62, No. 1, pp. 98-105).

   This paper presents a Bayesian approach to filtering junk email. The authors show that their approach can effectively detect spam emails with a high degree of accuracy.

3. Daramola, O., Atayero, A. A., & Oduwole, O. A. (2017). Machine learning based spam detection: A survey. Cogent Engineering, 4(1), 1291934.

   This survey paper provides an overview of machine learning-based spam detection techniques, including both supervised and unsupervised methods. The authors discuss the advantages and disadvantages of each technique and highlight the challenges of spam detection.

3.LIMITATIONS WITH CITATIONS

Lack of standard datasets: There is a lack of standard and widely accepted datasets for email spam detection, which can make it difficult to compare the results of different studies. Researchers often use different datasets, which may not be representative of real-world scenarios.

Imbalanced datasets: Many email spam datasets are imbalanced, with a significantly higher number of non-spam (ham) messages compared to spam messages. This can lead to biased results and lower performance in detecting spam messages.

Evolving spam techniques: Spammers are constantly evolving their techniques to evade detection, which can make it difficult to develop effective spam detection systems. New spamming techniques can be developed faster than detection systems can be updated.

Overfitting: Some studies may overfit their models to the specific dataset they used, resulting in poor generalization performance on new data. This can make it difficult to apply the results of these studies to real-world scenarios.

Lack of real-time detection: Many spam detection systems rely on batch processing, which means that emails are processed in batches rather than in real-time. This can lead to delays in detecting spam messages, which can be problematic in some scenarios.

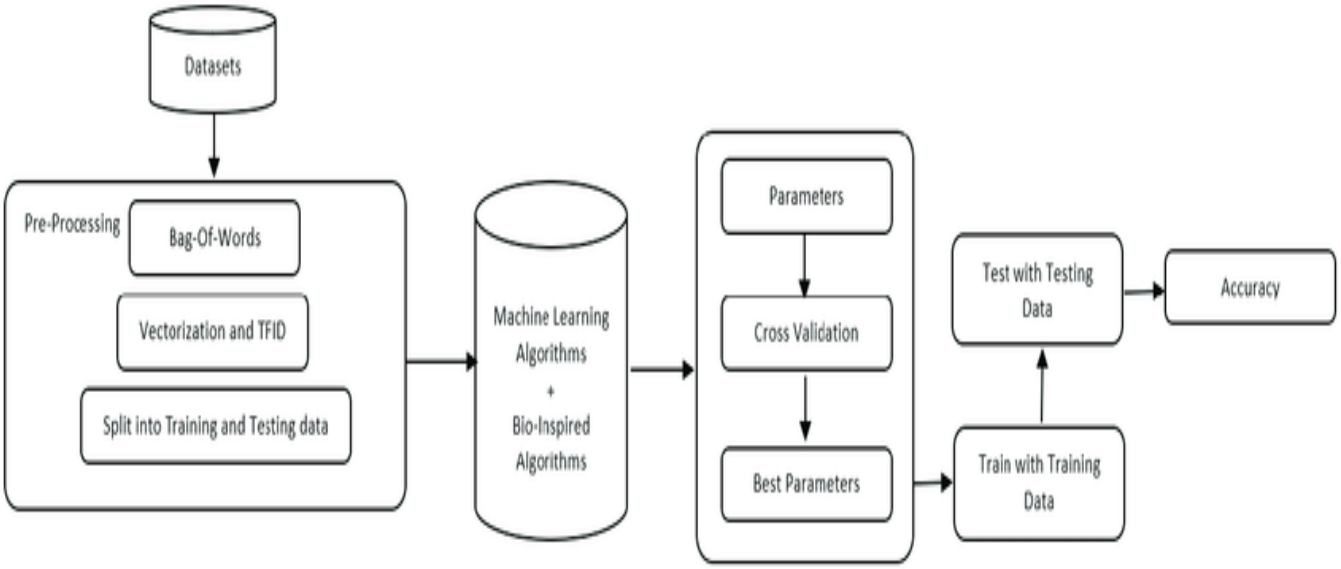
4.OBJECTIVES

The objective of email spam detection is to identify and filter out unwanted or unsolicited emails, commonly known as "spam," from an email inbox. Email spam can include a variety of content, such as advertisements, scams, phishing attempts, and unwanted newsletters. The goal of spam detection is to reduce the amount of time and effort that users have to spend sorting through unwanted emails, while also protecting them from potentially harmful content.

Email spam detection algorithms typically analyze various features of an email, such as the sender, subject line, body text, attachments, and metadata, to determine whether the email is likely to be spam or not. These algorithms may use machine learning techniques, such as decision trees, support vector machines, or neural networks, to automatically learn patterns and characteristics of spam emails and identify them with a high degree of accuracy.

Effective email spam detection can help to improve productivity, reduce the risk of cybersecurity threats, and enhance the overall user experience for email users.

5.SYSTEM ARCHITECTURE AND DESIGN



Modules and components that could be included in an email spam detection project using naive Bayes classification:

1. Data collection: This module is responsible for collecting email messages for building and testing the spam detection model. Data can be collected from various sources such as publicly available datasets, web scraping, or private email accounts.

2. Data pre-processing: This module is responsible for cleaning and transforming the collected data into a format that can be used for training the spam detection model. This includes removing irrelevant information, such as email headers, and converting the remaining text into a format suitable for analysis, such as tokenization and stemming.

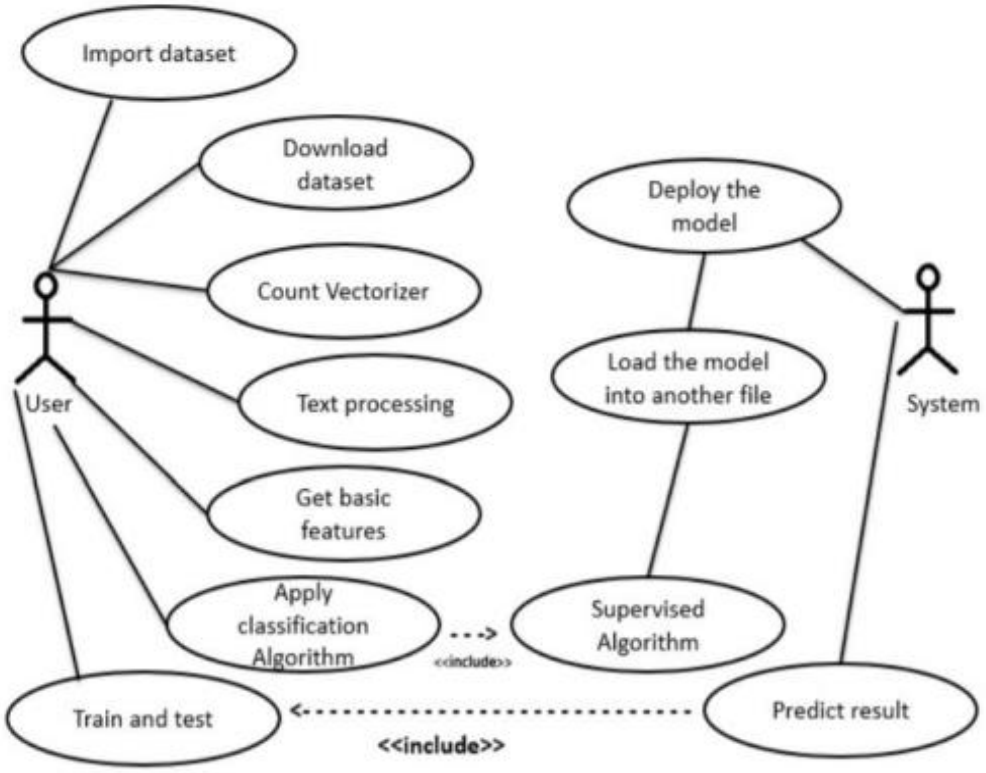
3. Feature extraction: This module is responsible for identifying relevant features from the pre-processed data that can be used to classify emails as spam or not spam. Common features include the frequency of certain words or phrases, the presence of certain types of attachments, and the length of the email message.

4. Model training: This module is responsible for training the naive Bayes classification model using the pre-processed data and extracted features. The model learns to classify emails as spam or not spam based on the provided training examples.

5. Model evaluation: This module is responsible for testing the performance of the trained model on a separate dataset of email messages that were not used for training. Performance metrics such as accuracy, precision, recall, and F1-score can be calculated to evaluate the effectiveness of the model.

6. Model deployment: This module is responsible for deploying the trained model in a production environment. This can involve integrating the model into an existing email service or creating a standalone application for users to check their emails for spam.

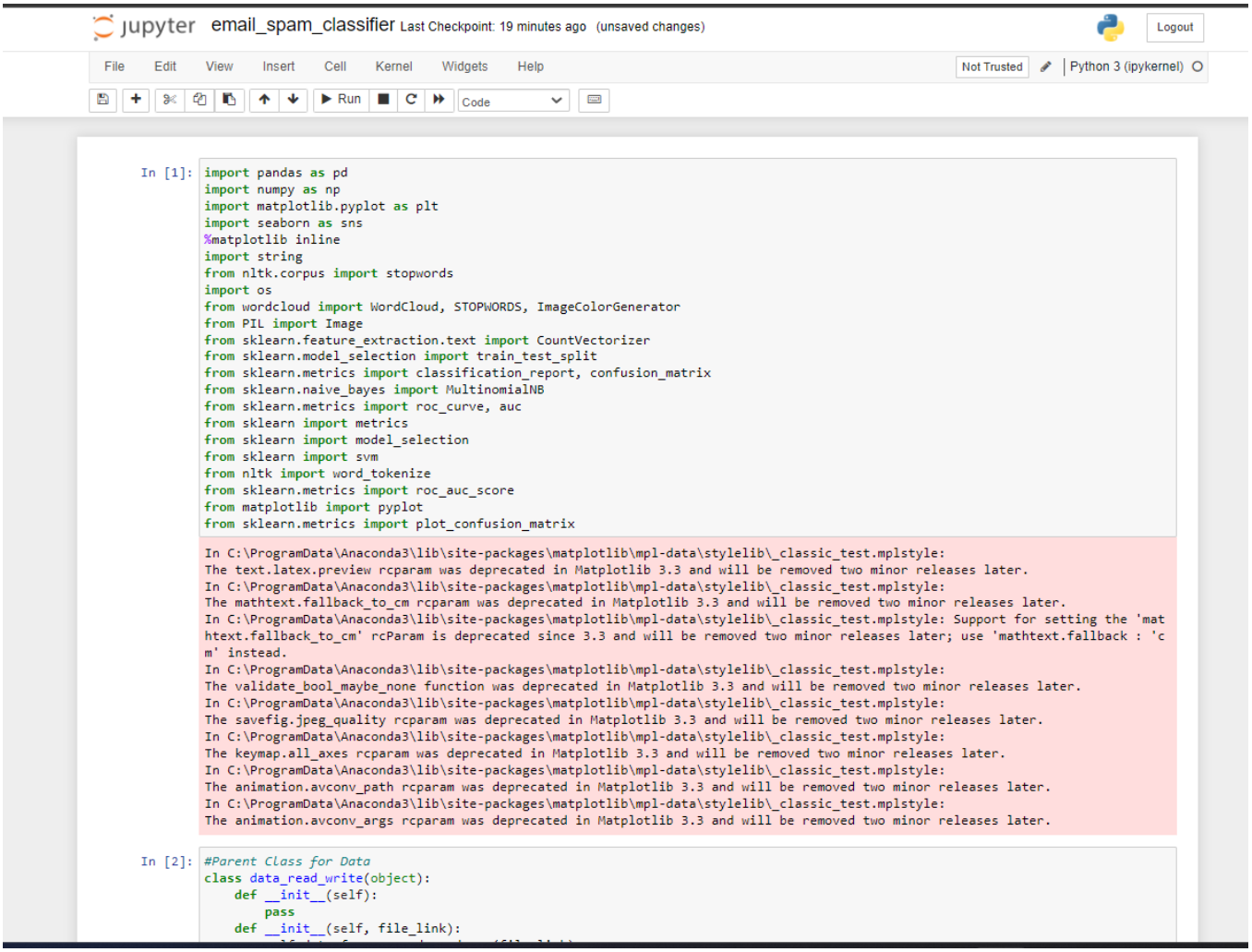
System design



6.METHODOLOGY

Naïve Bayes Classifiers is a popular mathematical method for filtering email. See usually use the wallet features to identify spam e-mail approach frequently used to separate text. Spam filtering of Naïve Bayes is a basic way to deal with such spam it can adapt to the email needs of individual users and provide low-level lies good spam detection rates that are generally accepted by users. One of the main benefits of screening Bayesian spam is that it can be trained per user. It can be especially helpful in avoiding false reasoning, where this is legal email is incorrectly labelled as spam. When deleting suspensions, the data set size decreases with the training time the model is also declining. Deleting stops can help improve vocabulary function as there are few and only meaningful tokens left. Using lemmatization recognizes the similarity of words like masculine, bed and beds etc. Accuracy is improved by using this Classifier Naïve Bayes. We get the accuracy of 98% use this process.

7.CODING AND TESTING



jupyter email_spam_classifier Last Checkpoint: 19 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

```
The animation.avconv_args rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
```

In [2]:
```
#Parent Class for Data
class data_read_write(object):
    def __init__(self):
        pass
    def __init__(self, file_link):
        self.data_frame =  pd.read_csv(file_link)
    def read_csv_file(self, file_link):
        #data_frame_read = pd.read_csv(file_link)
        #return data_frame_read
        #self.data_frame = pd.read_csv(file_link)
        return self.data_frame
    def write_to_csvfile(self, file_link):
        self.data_frame.to_csv(file_link, encoding='utf-8', index=False, header=True)
        return
```

In [3]:
```
#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 = '
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis("off")
        plt.show()
        wordcloud.to_file(output_image_file)
        return
```

In [4]:
```
#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.word
            final_join = ' '.join(Test_punc_removed_join_clean)
            return final_join
```

jupyter email_spam_classifier Last Checkpoint: 20 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [5]:
```
#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
```

jupyter email_spam_classifier Last Checkpoint: 20 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [5]:
```
#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
```

jupyter email_spam_classifier Last Checkpoint: 21 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

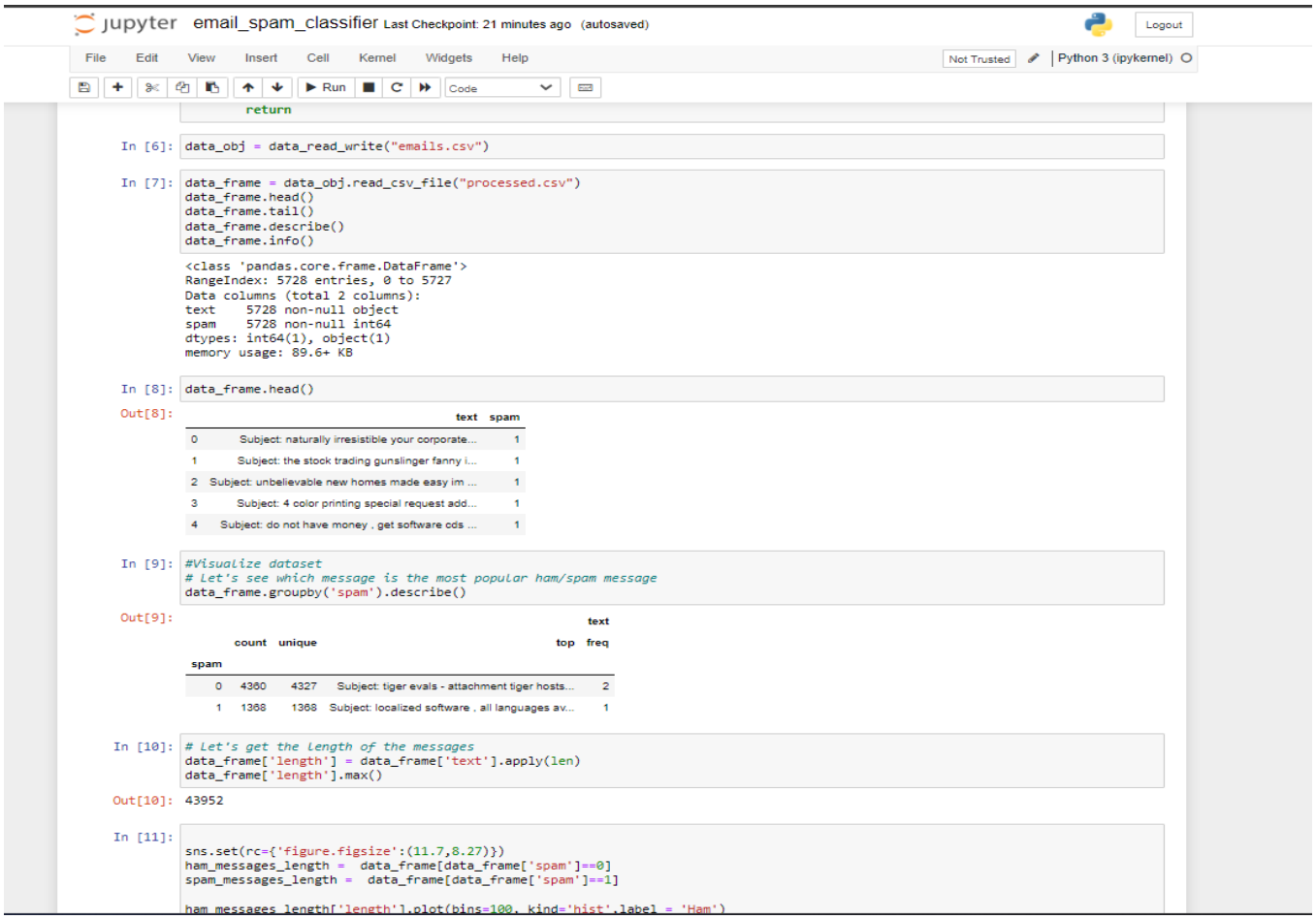
Run Code

```
    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
        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('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()
```


jupyter email_spam_classifier Last Checkpoint: 21 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
return
In [6]:
data_obj = data_read_write("emails.csv")
In [7]:
data_frame = data_obj.read_csv_file("processed.csv")
data_frame.head()
data_frame.tail()
data_frame.describe()
data_frame.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5728 entries, 0 to 5727
Data columns (total 2 columns):
text 5728 non-null object
spam 5728 non-null int64
dtypes: int64(1), object(1)
memory usage: 89.6+ KB
In [8]:
data_frame.head()
Out[8]:
text spam
0 Subject: naturally irresistible your corporate... 1
1 Subject: the stock trading gunslinger fanny i... 1
2 Subject: unbelievable new homes made easy im ... 1
3 Subject: 4 color printing special request add... 1
4 Subject: do not have money , get software cds ... 1
In [9]:
#Visualize dataset
# Let's see which message is the most popular ham/spam message
data_frame.groupby('spam').describe()
Out[9]:
text
count unique top freq
spam
0 4360 4327 Subject: tiger evals - attachment tiger hosts... 2
1 1368 1368 Subject: localized software , all languages av... 1
In [10]:
Let's get the length of the messages
data_frame['length'] = data_frame['text'].apply(len)
data_frame['length'].max()
Out[10]:
43952
In [11]:
sns.set(rc={'figure.figsize':(11.7,8.27)})
ham_messages_length = data_frame[data_frame['spam']==0]
spam_messages_length = data_frame[data_frame['spam']==1]
ham_messages_length['length'].plot(bins=100, kind='hist',label = 'Ham')

8.SCREENSHOTS AND RESULTS

jupyter email_spam_classifier Last Checkpoint: 22 minutes ago (autosaved)

Logout

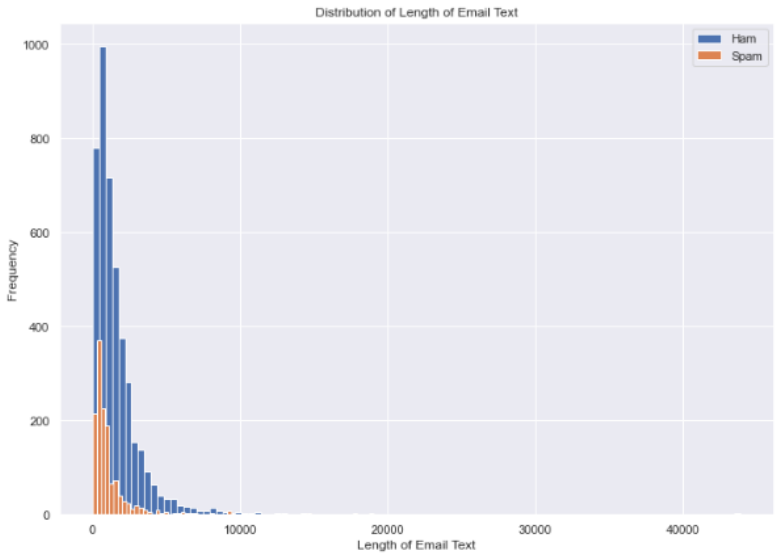
File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

```
plt.title('Distribution of Length of Email Text')
plt.xlabel('Length of Email Text')
plt.legend()
```

Out[11]: <matplotlib.legend.Legend at 0x2e158719f88>



In [12]:
```
#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))
```

```
8479
6131
```

In [13]:
```
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')
```

jupyter email_spam_classifier Last Checkpoint: 22 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

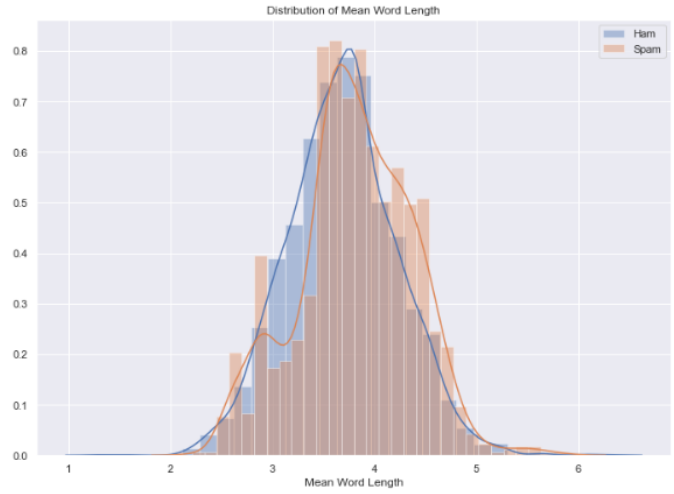
Not Trusted | Python 3 (ipykernel)

Run Code

In [14]:
```
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()
```



In [15]:
```
#Checking ratio of stop words

from nltk.corpus import stopwords
```

jupyter email_spam_classifier Last Checkpoint: 26 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

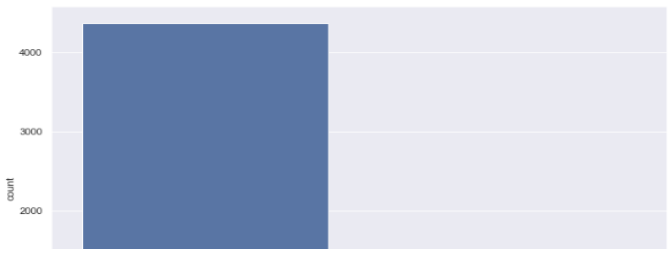
In [16]:
```
spam_stopwords
```

Out[16]:
```
0       0.230769
1       0.277778
2       0.397727
3       0.191919
4       0.396226
          ...
1363    0.342105
1364    0.365854
1365    0.437500
1366    0.446809
1367    0.320024
Name: text, Length: 1368, dtype: float64
```

In [33]:
```
# 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 %
```

Out[33]: <AxesSubplot:xlabel='Ham(0) and Spam(1)', ylabel='count'>



jupyter email_spam_classifier Last Checkpoint: 26 minutes ago (autosaved)

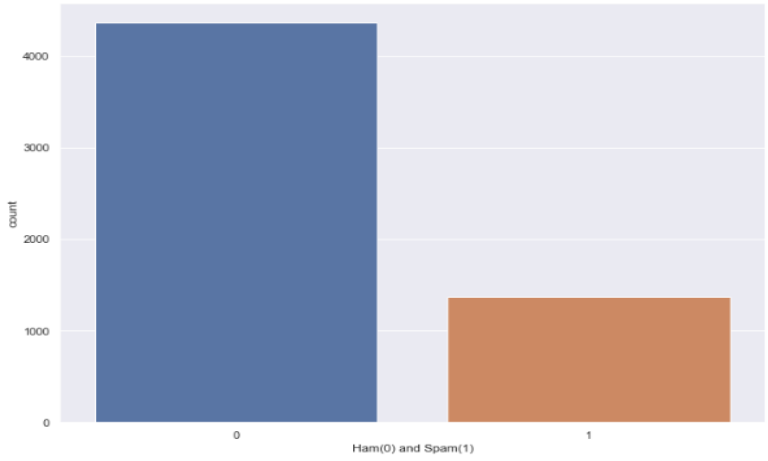
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

Out[33]: <AxesSubplot:xlabel='Ham(0) and Spam(1)', ylabel='count'>



In [18]:
```
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")
```

jupyter email_spam_classifier Last Checkpoint: 26 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [19]:
```
data_clean_obj = data_cleaning()
# Let's test the newly added function


data_frame['clean_text'] = data_clean_obj.apply_to_column(data_frame['text'])
```

In [20]:
```
data_frame.head()
```

Out[20]:

	text	spam	length	clean_text
0	Subject: naturally irresistible your corporate...	1	1484	Subject naturally irresistible corporate ident...
1	Subject: the stock trading gunslinger fanny i...	1	598	Subject stock trading gunslinger fanny merrill...
2	Subject: unbelievable new homes made easy im ...	1	448	Subject unbelievable new homes made easy im wa...
3	Subject: 4 color printing special request add...	1	500	Subject 4 color printing special request addit...
4	Subject: do not have money , get software cds ...	1	235	Subject money get software cds software compat...

In [21]:
```
data_obj.data_frame.head()
```

Out[21]:

	text	spam	length	clean_text
0	Subject: naturally irresistible your corporate...	1	1484	Subject naturally irresistible corporate ident...
1	Subject: the stock trading gunslinger fanny i...	1	598	Subject stock trading gunslinger fanny merrill...
2	Subject: unbelievable new homes made easy im ...	1	448	Subject unbelievable new homes made easy im wa...
3	Subject: 4 color printing special request add...	1	500	Subject 4 color printing special request addit...
4	Subject: do not have money , get software cds ...	1	235	Subject money get software cds software compat...

In [22]:
```
data_obj.write_to_csvfile("processed_file.csv")
```

In [23]:
```


# 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'])
```

jupyter email_spam_classifier Last Checkpoint: 27 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

```
#vectorizer = CountVectorizer()
cv_object = apply_embeddding_and_model()
spamham_countvectorizer = cv_object.apply_count_vector(data_frame['clean_text'])
```

In [24]:
```
#Separating Descriptive and Target Feature
X = spamham_countvectorizer
label = data_frame['spam'].values
y = label
```

In [25]:
```
cv_object.apply_naive_bayes(X,y)
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       901
           1       0.98      0.99      0.98       245

    accuracy                           0.99      1146
   macro avg       0.99      0.99      0.99      1146
weighted avg       0.99      0.99      0.99      1146

test set

Accuracy Score: 0.9921465968586387
F1 Score: 0.9817444219066936
Recall: 0.9877551020408163
Precision: 0.9758064516129032
Confusion matrix, without normalization
[[895   6]
 [  3 242]]
Normalized confusion matrix
```

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
```

jupyter email_spam_classifier Last Checkpoint: 28 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

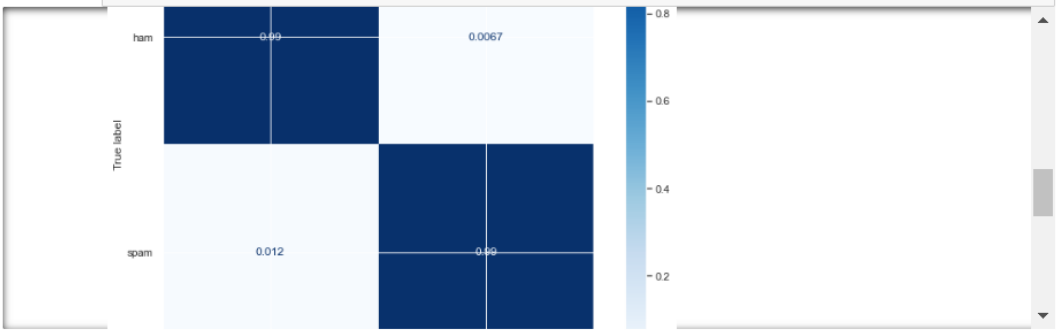
Not Trusted | Python 3 (ipykernel)

Run Code

```
#vectorizer = CountVectorizer()
cv_object = apply_embeddding_and_model()
spamham_countvectorizer = cv_object.apply_count_vector(data_frame['clean_text'])
```

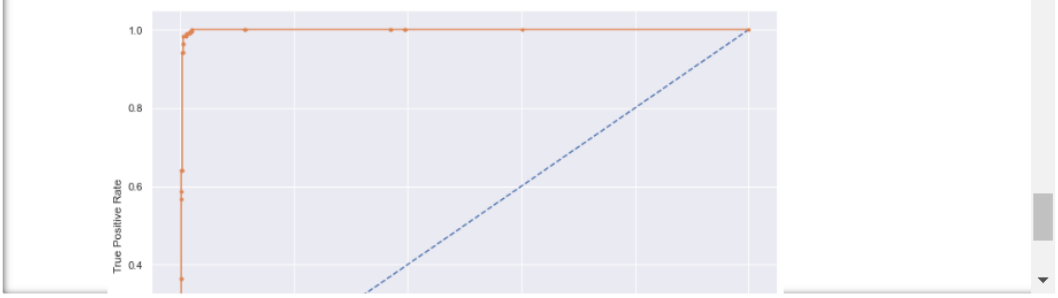
In [24]:
```
#Separating Descriptive and Target Feature
X = spamham_countvectorizer
label = data_frame['spam'].values
y = label
```

In [25]:
```
cv_object.apply_naive_bayes(X,y)
```



In [26]:
```
cv_object.apply_svm(X,y)
```

```
SVM: ROC AUC=0.998
```



9.CONCLUSION AND FUTURE ENHANCEMENTS

This project, spam detection is proficient of detecting mails giving to the content of the email. Detecting the spam emails can be done on the basis of the trusted and verified domain names. The spam email classification is incredibly significant in categorizing e-mails and distinct e-mails that are spam or non-spam. Naive Bayes could a baseline technique for managing with spam to the e-mail needs of individual users and provides low false positive spam detection rates that are generally acceptable to users. To further optimize the parameters of the Naïve Bayes approach is used, which results in increased the accuracy of the entire classification process. The accuracy of the spam detection can increase by using Naïve Bayes Classifier. In future the other optimization algorithm can be used with Naïve Bayes algorithm. Also, the other ML approach can be used instead of NB approach. The evaluation of the experiment is done on the basis of f1-score, precision, accuracy and recall. By evaluating the results, we are able to say that the integrated concept ends up in increased accuracy and precision than the individual Naïve Bayes approach.

Email spam detection has come a long way in recent years, but there is still room for improvement. Here are some potential enhancements that could be made to email spam detection in the future:

1. Hybrid models: Combining multiple machine learning algorithms and techniques such as deep learning, ensemble methods, and rule-based systems can improve the accuracy of email spam detection. Hybrid models can also be designed to handle different types of spam, including phishing attacks and malware.

2. Feature engineering: Feature engineering involves identifying and extracting relevant features from the email text and metadata that can be used to train machine learning models.

3. Human-in-the-loop systems: Human-in-the-loop systems involve incorporating human feedback into the spam detection process. This can help improve the accuracy of the system by providing additional context and feedback to the machine learning algorithms.

10.REFERENCES

[1]. Yuefeng Li, Abdulmohsen Algarni, Mubarak Albathan, Yan Shen, and Moch Arif Bijaksana," Relevance Feature Discovery for Text Mining"IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 6, JUNE 2015

[2]. Feldman, Moshe Fresko,Yakkov Kinar ,"Text Mining at the Term Level", Ronen Feldman , Moshe Fresko , Yakkov Kinar , Yehuda Lindell , Orly Liphstat , Martin Rajman , Yonatan Schler , Oren Zamir

[3]. Y. Li, A. Algarni, and N. Zhong, "Mining positive and negative patterns for relevance feature discovery," in Proc. ACM SIGKDD Knowl. Discovery Data Mining, 2010, pp. 753–762

[4]. Ann Nosseir, Khaled Nagati and Islam Taj-Eddin, "Intelligent Word-Based Spam Filter Detection Using Multi-Neural Networks", IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, No 1, March 2013 ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784.

[5]. R. Kishore Kumar, G. Poonkuzhali, P. Sudhakar," Comparative Study on Email Spam Classifier using Data Mining Techniques", Proceedings of the International MultiConference of Engineers and Computer Scientists 2012 Vol I, IMEC2012, March 14-16,2012, Hong Kong,

[6]. Rafiqul Islam and Yang Xiang, member IEEE, "Email Classification Using Data Reduction Method" created June 16, 2010.