International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 164-168
Published Online June 2023 in IJEAST (http://www.ijeast.com)

SPAM EMAIL DETECTION USING DIFFERENT CLASSIFIERS

Ayushi Tiwari, Saumya Kushwah, Ritika Khanduri, Nahid Fatima,
Students
Department of Computer science
HMR Institute of Technology & Management, Hamidpur Delhi-110036

Prof Gurpreet Kaur,
Guide
Department of Computer science
HMR Institute of Technology & Management, Hamidpur Delhi-110036

Abstract:- **Due to the widespread use of social media, the amount of unwanted emails has increased, necessitating the implementation of a reliable system to filter them out. Spam emailsare now the most common issue on the internet, despite email beingone of the fastest and most cost-effective forms of communication. Over the past few years, there has been a significant increase in spam emails due to the growing number of email subscribers.**
This study employs four classifiers (Random Forest, XG Boost, Naïve Bayesian) to classify email data, with varying data and feature sizes. The final classification result is '1' if the email is spamand '0' if it is not. The study was conducted using Python and implemented in a Jupyter notebook. This technique can also be utilized in monitoring social media and brand activity. The paper presents a machine learning approach for identifying spam emails by detecting spam content within the message. With machine learning, computers can learn how to perform a task without being explicitly programmed. This method uses data to generate a program that performs a task, such as classification. Unlike knowledge engineering, machine learning techniques require pre-classified data to create a training dataset that is used to fit the learning algorithm in the machine learning studio.
The objective of this study is to train, test, and compare various classifiers. The rest of the paper is organized as follows: Section 2 defines the researchers' contribution in this field. Section 3 describes the experimentation framework, dataset, procedures, andlibraries. Section 4 summarizes the findings, and Section 5 concludes the paper's potential for future research.

I. INTRODUCTION

Email is a widely used means of communication that allows for the exchange of various types of information including messages, documents, pictures, videos, and links. Despite its convenience, spam has become a significant issue on the internet, accounting for 65% of all email messages in 2022. This offensive content can be a waste of time, consume storage space and connection bandwidth, and lead to the accidental deletion of legitimate messages. Some countries have even implemented legislation to address this problem.
Text classification is essential for organizing the unstructured nature of text, including documents and spam communications. By leveraging machine learning, text classification can improve the accuracy of predictions and facilitate the analysis of large amounts of data.
This is particularly useful for businesses looking to gain insights from text data to inform decisions and automate processes. Text classification can be applied to brief texts like headlines and tweets, as well as larger documents like media articles.

II. LITERATURE REVIEW

To automate email filtering processes using AI, several researchers have emphasized the significance of this field.
[1]One proposed a solution where email data were classified with four classifiers, including Neural Network, SVM classifier, Naïve Bayesian Classifier, and J48 classifier. The experiment was conducted with different data sizes and feature sizes, and the result of the categorization was either "1" if it was determined to be spam or "0" if it wasn't. This study demonstrated that a simple J48 classifier that creates a binary tree was effective for the dataset that could be categorized as a binary tree.
[2]In another research, an author introduced an improved spam detection model based on Extreme Gradient Boosting (XGBoost), which has received little attention for spam email detection problems. The proposed model outperformed previous approaches across various evaluation parameters, according to experimental results.
The findings of the model were extensively examined and compared to those of previous studies.
[3] This article examines various machine learning techniques used to differentiate between legitimate and

spam messages, with a benchmark dataset consisting of 9324 records and 500 attributes. The article determines the most effective classification method, which can significantly aid in the removal of unwanted commercial communications, worms, electronic fraud, and other undesirable situations.
[4] In their paper, the authors propose a model that utilizes Bayes' theorem and the Naive Bayes' Classifier to determine whether a message is spam or not. The sender's IP address is also frequently detected.
Another research paper [5] surveys machine learning methods for spam filtering in email and IoT platforms, categorizing them into appropriate groups and making a thorough comparison of different methods based on accuracy, precision, recall, and other metrics. The paper also covers detailed findings and potential future research directions.
Lastly, the proposed work in [6] showcases differentiating features of the content of documents for spam filtering, an area of research that has seen much effort but mainly focuses on either natural language processing methodology with a single machine learning algorithm or a single natural language processing technique on multiple machine learning algorithms. The project creates a modelling pipeline to evaluate various machine learning approaches.

III. 3.MATERIALS AND METHODS

This section presents an approach for employing classifiers to predict whether an email is Spam or Ham based on the Spam Email Dataset. We initiate the process by performing data conversion, preprocessing, and partitioning to suit the algorithms being considered. Following that, we train and assess several models, using performance metrics to compare and evaluate them.

3.1 Framework

3.1.1 Data collection: We collect a labelled email dataset to train and evaluate the classifiers. The dataset should encompass various email types to represent the emails that the system will encounter in practice.

3.1.2 Data pre-processing: The emails in the dataset undergo preprocessing to make them suitable for use in the classifiers. This may involve text cleaning and normalization, eliminating stop words and punctuation, and transforming the text into a numerical format.

3.1.3 Feature extraction: Features are extracted from preprocessed emails that will be used to train classifiers. These features may include word frequency, specific keywords, and other text-based characteristics.

3.1.4 Classifier training: Classifiers are trained on a labeled dataset using the extracted features as inputs, the aim of learning patterns that correspond to different email categories.
3.1.5 Evaluation: Metrics like accuracy, precision, recall, and F1 score are utilized to evaluate the classifiers' performance. The final system employs the classifier that exhibits the highest performance on the evaluation dataset.

3.1.6 Ensemble methods: The system's overall performance is enhanced through the utilization of a blend of classifiers.

1.1.1. Deployment: The chosen classifier(s) are deployed in the email classification system, where they are used to automatically categorize incoming emails

1.1.2. Regular evaluation and update: To ensure that the system remains effective, it undergoes regular evaluations and updates as required. These updates may involve the incorporation of newly labeled data into the classifier and re-training it to adjust to evolving email data patterns.

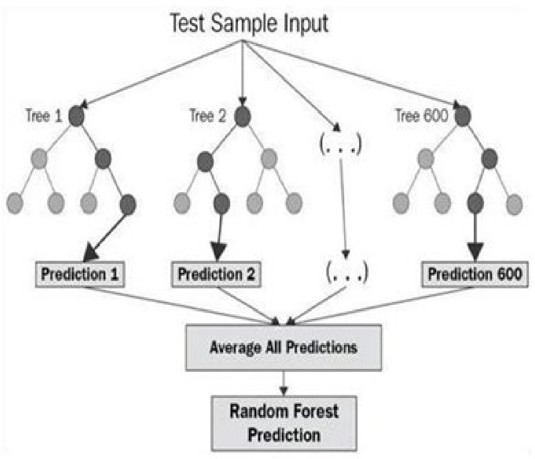
IV. THE MACHINE LEARNINGAPPROACHES

Naive Bayes- The Bayesian classifier, commonly utilized in text categorization, is a probabilistic technique that aims to determine whether an email is spam or not by analysing its word usage. It uses a Bayesian approach to assign the most probable label to a new email. The basic form of this network is a naive Bayes network where all attributes are assumed to be independent of the class variable. The classification problem is solved by finding the maximum value of the equation mentioned below.

$$P(a_j | y_1 y_2 y_3 y_4 \ldots y_n) = \frac{P(y_1\ y_2\ y_3\ y_4 y_5 \ldots y_n | a_j) * P(a_j)}{P(y_1 y_2 y_3 y_4 y_5 \ldots y_6)}$$

P(aj) represents the probability of a random sample belonging to category aj. Given that the training sample is already in category aj, P(y1, y2, y3….. yn|aj) denotes the likelihood of category aj containing the feature vector y = (y1, y2,, y3,…., yn). The overall joint probability of all potential categories is denoted by P(a1, a2, a3,…..,an).
It is an approach of machine learning used for solving to the second decision tree. These individual Classification& regression issues. Random Forest makes use of classifiers/predictors then ensemble to give a strong and

more Ensemble learning to solve complex problems by combining many classifiers. precise model. It can be used to solve problems including regression, classification, ranking, and custom prediction.



.
A large set of decision trees, also called estimators, make up the random forest. The final predictions of the random forest are determined by taking the average of the predictions made by each tree. These equations are used to solve the problem.

$$RF\mathrm{fi}_i = \frac{\sum_{j \in all\ trees} norm\mathrm{fi}_{ij}}{T}$$

$$norm\mathrm{fi}_i = \frac{\mathrm{fi}_i}{\sum_{j \in all\ features} \mathrm{fi}_j}$$

$$\mathrm{fi}_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_k}$$

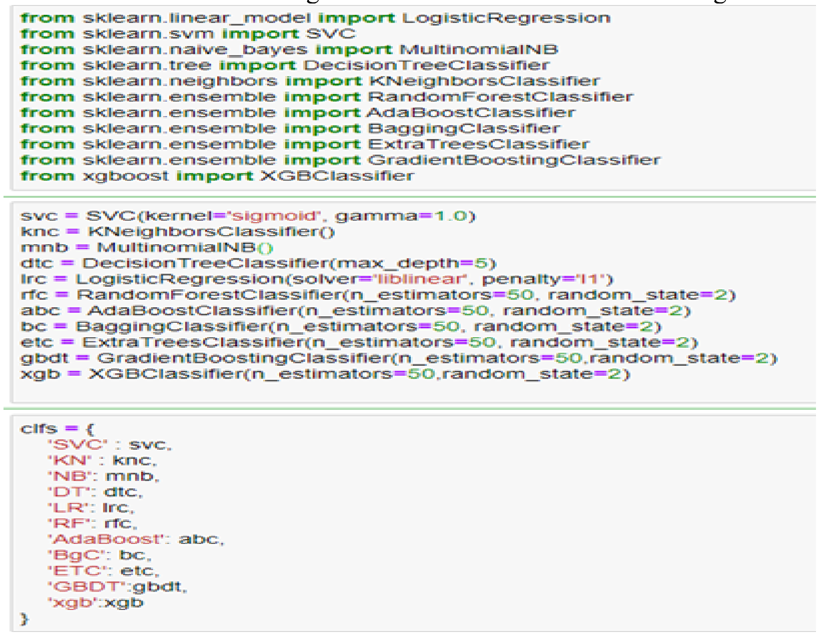
$$ni_j = W_j C_j - W_{left(j)} C_{left(j)} - W_{right(j)} C_{right(j)}$$

Decision Tree- Another algorithm that has been employed more frequently in the supervised learning approach research is the decision tree machine learning algorithm. The decision tree algorithm is a popular supervised learning approach used in machine learning research that can handle both numerical and categorical data. Its output is similar to a binary tree and consists of branches representing options and leaf nodes used for classification. This algorithm utilizes association rules to predict and associate target labels.

XG Boost Classifier- XGBoost is an implementation of Gradient Boosted decision trees that is commonly used in Kaggle Competitions. In this approach, decision trees are created sequentially and each independent variable is given a weight before being fed into the decision tree. The weight of variables predicted incorrectly by the tree is increased and thenfed back into the tree for further improvement. Weights play acrucial role in the XGBoost algorithm.

precise model. It can be used to solve problems including regression, classification, ranking, and custom prediction.

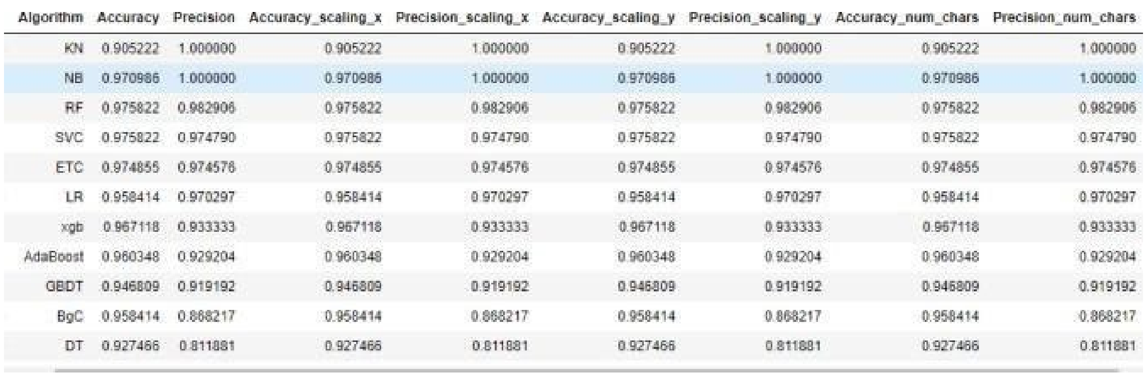
Table1:Machine Learning ClassifierResults evaluated on testing data.



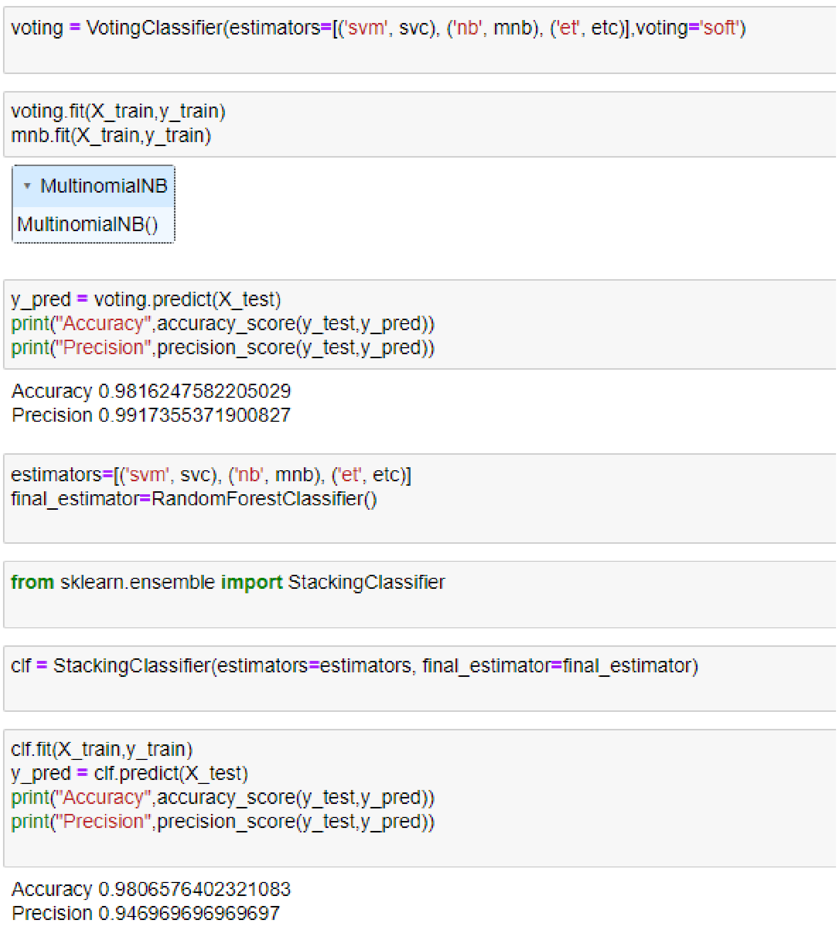
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

```
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
```

```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```



Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_scaling_y	Precision_scaling_y	Accuracy_num_chars	Precision_num_chars
KN	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000
NB	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000
RF	0.975822	0.982906	0.975822	0.982906	0.975822	0.982906	0.975822	0.982906
SVC	0.975822	0.974790	0.975822	0.974790	0.975822	0.974790	0.975822	0.974790
ETC	0.974855	0.974576	0.974855	0.974576	0.974855	0.974576	0.974855	0.974576
LR	0.958414	0.970297	0.958414	0.970297	0.958414	0.970297	0.958414	0.970297
xgb	0.967118	0.933333	0.967118	0.933333	0.967118	0.933333	0.967118	0.933333
AdaBoost	0.960348	0.929204	0.960348	0.929204	0.960348	0.929204	0.960348	0.929204
GBDT	0.946809	0.919192	0.946809	0.919192	0.946809	0.919192	0.946809	0.919192
BgC	0.958414	0.868217	0.958414	0.868217	0.958414	0.868217	0.958414	0.868217
DT	0.927466	0.811881	0.927466	0.811881	0.927466	0.811881	0.927466	0.811881



```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='soft')
```

```
voting.fit(X_train,y_train)
mnb.fit(X_train,y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9816247582205029
Precision 0.9917355371900827
```

```
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
```

```
from sklearn.ensemble import StackingClassifier
```

```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
```

```
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9806576402321083
Precision 0.946969696969697
```

V. CONCLUSION

This article presents a machine learning-based spam mail detection system designed to classify and filter emails as either spam or non-spam. To evaluate the system's effectiveness and identify areas for improvement, testing isa critical step. The testing process involves preparing test data, executing tests, evaluating performance, comparing results, debugging, and retesting. There are various types of testing that can be used, including unit testing, integration testing, functional testing, performance testing, security testing, and acceptance testing. The system's performance metrics, such as accuracy, precision, recall, and F1-score, are among the results obtained from testing, as well as any bugs or issues that were discovered during the process. These results can be utilized to assess the system's performance against other methods and to improve its performance. There are several ways to improve the spam mail detection system's capabilities, including using more diverse and representative data, improving feedback mechanisms, utilizing advanced machine learning techniques, enhancing scalability, improving security, incorporating language and cultural diversity, and integrating adaptability and feedback from end-users.

To enhance the classifier's performance, CNN and RNN can be utilized, along with incorporating sender, subject, recipients, and email time as additional features to provide valuable information for classification. Transfer learning can also improve the performance of the classifier by fine-tuning the pre-trained model on the labeled email dataset. Moreover, supporting multiple languages or multilingual support can be added to the current model that only supports English. Additionally, involving human input in the training process using active learning can further enhance the classifier's performance.

VI. REFERENCE

[1]. https://link.springer.com/chapter/10.1007/978-1-4020-6264-3_67

[2]. https://www.researchgate.net/publication/348078728_Effective_Email_Spam_Detection_System_using_Extreme_Gradient_Boosting

[3]. https://www.researchgate.net/publication/357203782_Classification_of_Spam_Messages_using_Random_Forest_Algorithm

[4]. https://www.researchgate.net/publication/342113653_Email_based_Spam_Detection

[5]. https://www.ijert.org/email-based-spamdetection

[6]. https://www.hindawi.com/journals/scn/2022/1862888/

[7]. https://www.irjet.net/archives/V8/i4/IRJETV8I472.pdf

[8]. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4145123