Email Spam Filter

Shruti Korpade, Hemal Kulkarni, Rahul Kumbhar

Department of Electronics Engineering, Vishwakarma Institute of Technology, 666, Upper Indira Nagar, Bibwewadi, Pune – 411037

Email: - Email: - shruti.korpade21@vit.edu , hemal.kulkarni22@vit.edu , rahul.kumbhar21@vit.edu

milind.kamble@vit.edu

Guided by : Prof Milind Kamble

Abstract:

This project presents an efficient Email Spam Filter that uses Multinomial Naive Bayes, Term Frequency-Inverse Document Frequency (TF-IDF), a curated email corpus, the Natural Language Toolkit (NLTK), and a Streamlit app for user-friendly interaction. The Multinomial Naive Bayes classifier is used to classify incoming messages based on word occurrence probabilities in spam and non-spam emails. TF-IDF transforms the text into numerical features, enhancing its ability to distinguish between spam and legitimate content. A curated email corpus is used to train and test the model, ensuring its accuracy and robustness. The NLTK library is used for natural language processing tasks like tokenization, stemming, and stop-word removal, enhancing the input data quality. A user-friendly Streamlit app is developed to make the spam filter accessible and user-friendly, allowing users to input and classify their emails as spam or not spam. This robust solution effectively filters out unwanted emails and empowers users with control over their inboxes.

Keyword: multinomial naïve bayes, TF-IDF, NLTK, corpus, Stremlit app

I. INTRODUCTION

The Email Spam Filter project aims to create a robust and accurate Email Spam Filter using advanced techniques such as Multinomial Naive Bayes, Term Frequency-Inverse Document Frequency (TF-IDF), a carefully curated email corpus, the Natural Language Toolkit (NLTK), and a user-friendly Streamlit application. The project aims to address the persistent problem of email spam by utilizing machine learning and natural language processing. The Multinomial Naive Bayes classifier is used to distinguish spam from non-spam messages by analyzing the probabilities of word occurrences in known spam and legitimate emails. TF-IDF is used to convert email text into numerical features, considering the importance of each word in the document. The project benefits from a meticulously curated email corpus, which is essential for training and testing the filter. The Natural Language Toolkit (NLTK) is used to enable preprocessing tasks such as tokenization, stemming, and stop-word removal, improving the quality of input data and enhancing the filter's ability to differentiate between spam and legitimate emails. A user-friendly Streamlit application is developed to make the spam filter accessible and user-friendly. Users can input their emails, classify them as spam or not spam, and display the classification results, allowing them to take appropriate actions. In summary, the Email Spam Filter project represents a comprehensive solution to the persistent problem of email spam by combining advanced machine learning techniques, a well-structured email corpus, natural language processing, and a user-friendly interface. This system streamlines the process of filtering spam and empowers users to regain control over their email communications.

II. LITERATURE REVIEW

The China National Ministry of Industry and Information reports that mobile phone users in China exceed 1,300 million, but spam SMSs have become a significant issue. Dahan Tricom Corporation, responsible for main short message (SMS) service traffic and value-added services in East China, has made significant progress in the past 13 years. Currently, Dahan sends over 300 million SMSs per month. However, the issue of spam SMSs threatens other SMS services providers and terminal mobile users in China. Spam SMSs can contain advertisement information, illegal information, and fraud information, wasting traffic and spreading unintended information. The main SMS service providers, including China Telecom, China Mobile, and China Unicom, use keywords to filter spam SMS. However, mechanical matching often judges incorrectly. To address this issue, a Vector Space Model (VSM)-based spam SMS filter is proposed. This method addresses the particularity of SMS, such as short, vocal, and domain-related messages. The technology has been deployed in Dahan Tricom Corporation's production environment and has shown potential in SMS commercial companies.[1]

Emailing is a convenient and ubiquitous form of communication for both professional and personal use, allowing for the fast transmission of complex information such as text, images, videos, documents, and URLs. It has become a major part of corporate and academic operations, managing all major communications and daily operations. However, email protocols like SMTP and POP are easily accessible and vulnerable to misuse. Many irrelevant and unsolicited emails are sent daily, with most being auto-generated. Spam emails, which cause a loss of around $20 million annually, are used for advertising, ransomware, phishing, fake purchase receipts, increasing traffic to malicious websites, loading scripts, ransomware, malicious websites, loading scripts, crimeware, rootkits, and underlying executable files. A survey by the Radicati Group found that 18.5% of emails are irrelevant to the recipient and 22.8% are sent unnecessarily. Handling spam also wastes time for the average user, reducing productivity. Additionally, the arrival of spam takes up memory space on servers, incurring additional costs for providers, users, and companies.[2]

Email is a crucial communication method in the education industry, with administrators dealing with high volumes of emails daily. Emails can herald important meetings, work messages, and transfer important documents, such as international student information and scanned applications for admission. However, important work-related emails are often found in spam folders. To improve spam classification accuracy, a spam filter named LingerIG was implemented in 2003 in an email classification system called Linger. However, this solution has unstable accuracy in classifying non-spam emails into folders. A context-based email classification model was developed to better adapt to classifying emails into homogenous groups. This paper aims to combine the spam filter with the context-based email classification model to improve spam email classification accuracy to 100%. The proposed system uses Linger's information gain classifier and neural network to classify emails into homogenous clusters. The proposed solution provides 100% accuracy in filtering spam from mixed emails.[3]

Email spam filtering is an online supervised learning task for binary text classification (TC) that involves two supervised processes: online learning and online predicting. Previous algorithms often treat an email as a single plain-text document, but a full email, which includes five natural text fields, is a multi-field text document. The challenge of multi-field structural features presents an opportunity to improve the performance of previous algorithms. The paper proposes a multi-field learning (MFL) approach, which is an alignment technique of text feature sources, enhancing text features by field information and reducing disturbances among different fields. An effective string-frequency index (SFI) binary TC algorithm is also proposed within the MFL framework, which has low space-time complexity for both online learning and online predicting. The MFL framework can improve the performance of some statistical TC algorithms, and the proposed SFI algorithm can reach state-of-the-art performance at greatly reduced computational cost. Further research will focus on online semi-supervised learning, active learning, and personal learning for email spam filtering.[4]

Email has become the most widely used communication mechanism for internet users, but its increased usage has led to issues with spam emails. Spammers earn around 3.5 USD million from spam annually, causing financial losses for both personal and institutional purposes. Email users spend significant time on spam, which accounts for over 50% of email server traffic. Spammers use these emails for marketing purposes, causing malicious activities like identity theft and financial disruptions. Email management and classification are crucial for organizations to increase productivity and reduce financial losses. Research has identified high adoption rates for supervised machine learning approaches, Naïve Based and SVM algorithms, and multi-algorithm systems for better outcomes. Future research should focus on email features like BoW and Body text.[5]

The internet plays a crucial role in various fields, including communication and information exchange. Email, an electronic mail source, is a powerful tool for information exchange, including text, voice messages, video, and graphics. However, the widespread use of email has led to the problem of spam, which can be either unwanted or unsolicited. Various methods and techniques have been developed to identify and filter spam, with the goal of reducing wastage of time and bandwidth. Detection methods include machine learning (ML) and non-ML techniques. ML is a statistical computer algorithm that learns from data and continuously improves its performance with new data. Email filtration includes all contents of the email, including text, images, and header files. Organizations are using various tools, such as contracted anti-spam services, email filtering gateways, corporate email systems, and end user training, to counter the email spam problem. Identifying spam from legitimate emails is crucial, and research has been conducted to generate algorithms to cope with this challenge.[6]

III. METHODOLOGY

Data Cleaning:

1) Dropping Last 3 Columns in spam.csv: In the initial data cleaning phase, the last three columns of the 'spam.csv' dataset were dropped. These columns were presumably not relevant to the spam classification task.

2) Renaming the Columns: Column names were possibly renamed to enhance clarity and readability. This step is crucial for ensuring that the dataset is well-structured and the columns have meaningful names.

3) Checking the Missing Values: A check for missing values in the dataset was conducted. Missing values, if any, were identified and addressed appropriately, such as through imputation or data removal.

4) Checking the Duplicate Values: Duplicate rows in the dataset were identified by checking for rows with identical content.

5) Removing the Duplicates: Duplicate rows were removed to ensure that each data point is unique and to avoid skewing the model's training.

EDA (Exploratory Data Analysis):

1) Data is Imbalanced: The dataset was examined to determine if there is an imbalance between spam and non-spam (ham) emails. Class imbalance can impact model performance.

2) 'apply' is a Function in 'pandas' Library: The 'apply' function in the 'pandas' library was likely used for specific data transformations or operations on the dataset.

3) Fetching the Number of Words in Each SPAM: The number of words in each spam email was computed, which can provide insights into the text characteristics of spam messages.

4) Fetching the Number of Sentences in Each SPAM: Similar to the previous point, the number of sentences in each spam email was calculated.

5) Analyzing HAM: Exploratory analysis was conducted on ham (non-spam) messages to gain insights into their characteristics.

6) Analyzing SPAM: Similar to the previous point, analysis was performed on spam messages.

7) Analyzing the Ham and SPAM Messages with a Histogram: A histogram was likely used to visualize the distribution of messages based on the number of characters and words distinctly, allowing for a better understanding of the data distribution.

8) To Know the Relation Between Number of Characters, Words, and Sentences Using Correlation Map: A correlation map might have been generated to explore relationships between the number of characters, words, and sentences in the dataset.

Data Preprocessing:

1) Lower Case: The text data was converted to lowercase to ensure consistency in text processing.

2) Tokenization: Tokenization was applied to break down the text into individual words or tokens for further analysis.

3) Removing Special Characters: Special characters that may not contribute to the analysis were removed from the text.

4) Removing Stop Words and Punctuation: Common stop words and punctuation were likely removed from the text to improve the quality of the data.

5) Stemming: The text data was subjected to stemming to reduce words to their root forms, making text analysis more effective.

MODEL BUILDING:

1) Vectorizing the Data Using Bag of Words/TF-IDFVectorizer: Text data was vectorized using the Bag of Words (BoW) or TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique.

2) Give This as Input in Naïve Bayes: The vectorized data was used as input to train a Naïve Bayes classifier for spam classification.

3) Find the Accuracy to Check the Performance: The model's accuracy was computed to assess its performance in classifying spam and non-spam emails.

4) Precision Score Must Be High: Precision, a measure of the model's ability to correctly classify spam, was emphasized over accuracy, indicating that minimizing false positives is a priority. Multinomial Naïve Bayes was a selected algorithm for its high precision.

5) Tfidf → MNB: TF-IDF vectorization was paired with the Multinomial Naïve Bayes classifier.

6) ETC: Extra Tree Classifier Has the Highest Accuracy Score: The Extra Tree Classifier (ETC), similar to Random Forest, achieved the highest accuracy score, although precision may differ.

7) Naïve Bayes Has the Highest Precision: Despite differences in accuracy, Naïve Bayes demonstrated the highest precision, which is essential for minimizing false positives in spam classification.

8) Improving the Model:

a. Modifying the max_features parameter of TFIDF to control the number of words used.
b. Restricting the number of words when vectorizing the text.
c. Scaling the X array, though it didn't improve precision.
d. Appending the number of characters column to X, leading to changes in accuracy and precision.

e. Using a Voting Classifier, which combines multiple algorithms with equal weightages.
f. Employing Stacking, similar to Voting, by combining algorithms based on predefined weightages. It did not yield significant improvements.

IV. IMPLEMENTATION

```
import numpy as np

import pandas as pd

df = pd.read_csv('spam.csv', encoding='latin-1')

df.sample(5)

df.shape
```

#Data Cleaning

```
df.info()

# drop last 3 cols
df.drop(columns=['Unnamed: 2','Unnamed:
3','Unnamed: 4'],inplace=True)

df.sample(5)

# renaming the cols
df.rename(columns={'v1':'target','v2':'text'},inplace=
True)
df.sample(5)

from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

df['target'] = encoder.fit_transform(df['target'])

df.head()

# missing values
df.isnull().sum()

# check for duplicate values
df.duplicated().sum()

# remove duplicates
df = df.drop_duplicates(keep='first')

df.duplicated().sum()

df.shape
```

#Exploratory Data Analysis

```
df.head()

df['target'].value_counts()

import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(),
labels=['ham','spam'],autopct="%0.2f")
plt.show()
```

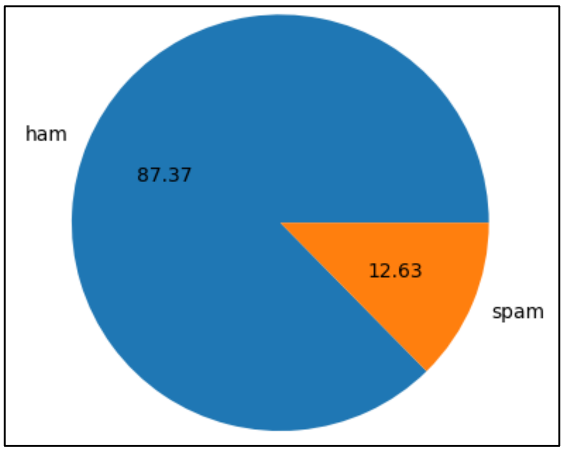
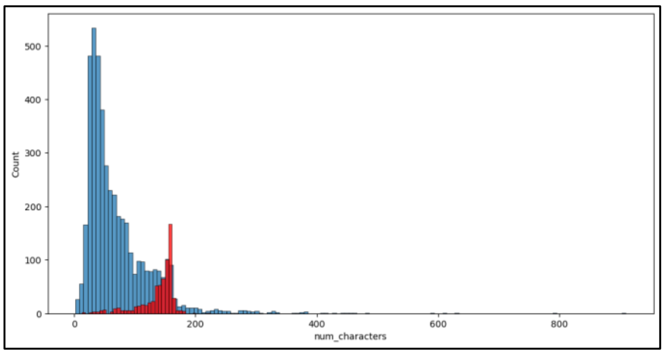
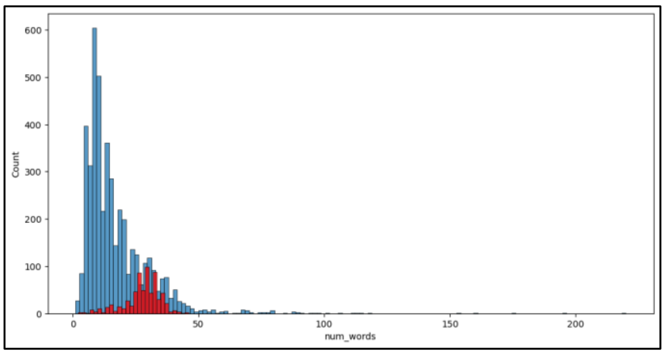


Fig no 1. Pie-chart

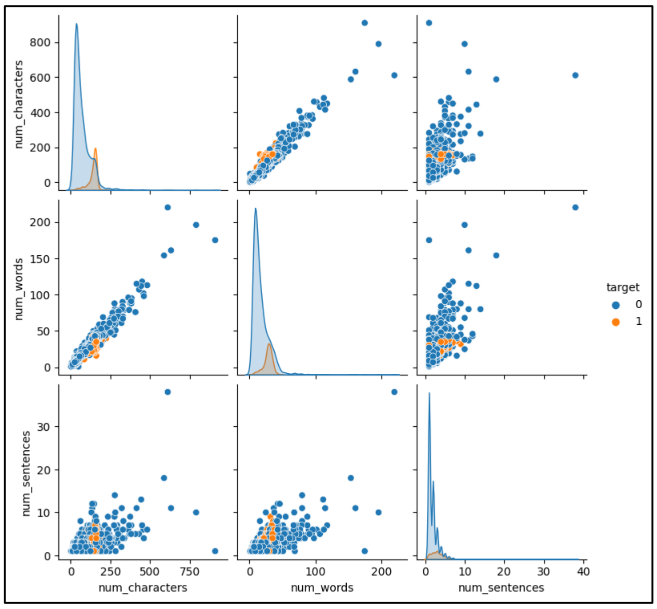
```
import nltk

!pip install nltk

nltk.download('punkt')

df['num_characters'] = df['text'].apply(len)

df.head()

# num of words
df['num_words'] = df['text'].apply(lambda
x:len(nltk.word_tokenize(x)))

df.head()

df['num_sentences'] = df['text'].apply(lambda
x:len(nltk.sent_tokenize(x)))

df.head()

# ham
df[df['target'] ==
0][['num_characters','num_words','num_sentences']].desc
ribe()

#spam
df[df['target'] ==
1][['num_characters','num_words','num_sentences']].desc
ribe()

import seaborn as sns

plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] ==
1]['num_characters'],color='red')
```



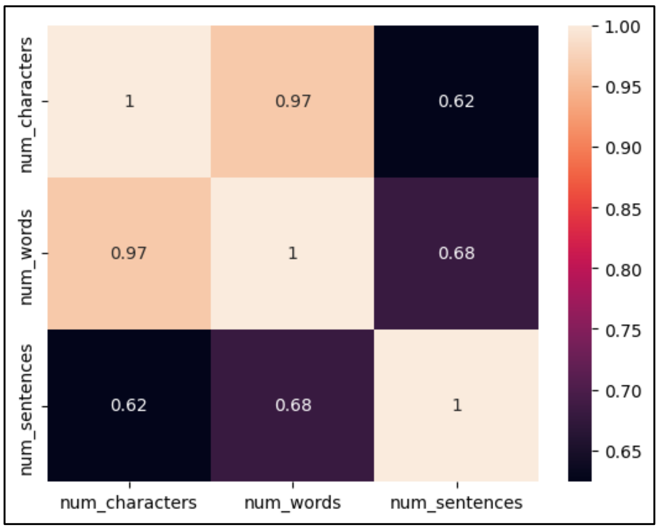
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] ==
1]['num_words'],color='red')
```



```
sns.pairplot(df,hue='target')
```

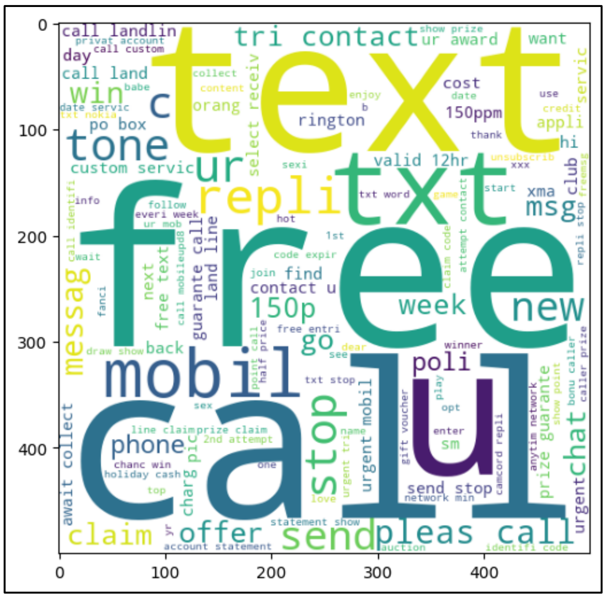


```
import seaborn as sns
import pandas as pd

numeric_columns = df.select_dtypes(include=['float64',
'int64'])

correlation_matrix = numeric_columns.corr()

sns.heatmap(correlation_matrix, annot=True)
```

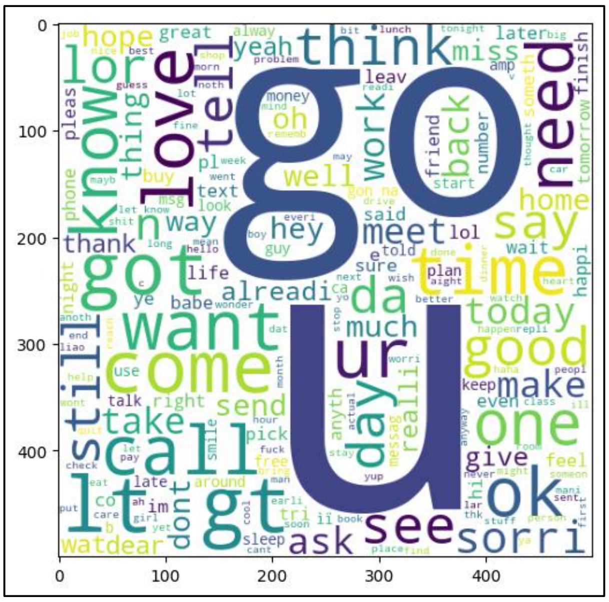


#Data Preprocessing

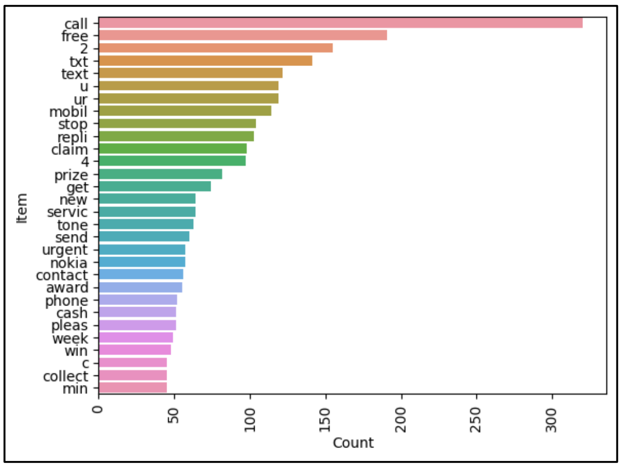
```
nltk.download('stopwords')

import string
from nltk.corpus import stopwords

def transform_text(text):
    y = []
    text = text.lower()  # Convert text to lowercase
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in
string.punctuation:
            y.append(i)
    text = y[:]
    return text

result = transform_text("I'm gonna be home soon and i
don't want to talk about this stuff anymore tonight, k?
I've cried enough today.")
print(result)

import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

ps = PorterStemmer()

def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in
string.punctuation:
            y.append(i)
    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))

    return " ".join(y)

transform_text("I'm gonna be home soon and i don't
want to talk about this stuff anymore tonight, k? I've
cried enough today.")

     df['text'][10]

from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')

df['transformed_text'] = df['text'].apply(transform_text)

     df.head()

from wordcloud import WordCloud
wc =
WordCloud(width=500,height=500,min_font_size=10,ba
ckground_color='white')

spam_wc = wc.generate(df[df['target'] ==
1]['transformed_text'].str.cat(sep=" "))

plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```



```
ham_wc = wc.generate(df[df['target'] ==
0]['transformed_text'].str.cat(sep=" "))

plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```



```
df.head()

spam_corpus = []
for msg in df[df['target'] ==
1]['transformed_text'].tolist():
   for word in msg.split():
      spam_corpus.append(word)

len(spam_corpus)

from collections import Counter
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

counter = Counter(spam_corpus)

top_30 = pd.DataFrame(counter.most_common(30),
columns=["Item", "Count"])

sns.barplot(x="Count", y="Item", data=top_30)
plt.xticks(rotation='vertical')
plt.show()
```



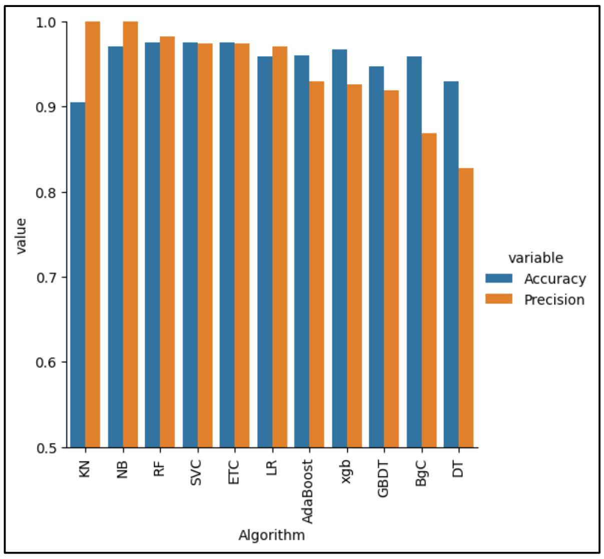
```
# Text Vectorization using Bag of Words
df.head()
```

#Model Building

```
from sklearn.feature_extraction.text import
CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)

    X =
    tfidf.fit_transform(df['transformed_text']).toarray()

#from sklearn.preprocessing import MinMaxScaler
#scaler = MinMaxScaler()
#X = scaler.fit_transform(X)

# appending the num_character col to X
#X = np.hstack((X,df['num_characters'].values.reshape(-
1,1)))

X.shape

y = df['target'].values

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=2)

from sklearn.naive_bayes import
GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import
accuracy_score,confusion_matrix,precision_score

gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()

gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))

mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))

bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))

# tfidf --> MNB

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_sta
te=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
    }

def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision

    train_classifier(svc,X_train,y_train,X_test,y_test)

    accuracy_scores = []
    precision_scores = []

    for name,clf in clfs.items():

        current_accuracy,current_precision =
    train_classifier(clf, X_train,y_train,X_test,y_test)

        print("For ",name)
        print("Accuracy - ",current_accuracy)
        print("Precision - ",current_precision)

        accuracy_scores.append(current_accuracy)
        precision_scores.append(current_precision)

performance_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accura
cy_scores,'Precision':precision_scores}).sort_values('Pre
cision',ascending=False)

performance_df
```

```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")

performance_df1

sns.catplot(x = 'Algorithm', y='value',
            hue = 'variable',data=performance_df1, kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```



```
# model improve
# 1. Change the max_features parameter of TfIdf

temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accuracy_scores,'Precision_max_ft_3000':precision_scores}).sort_values('Precision_max_ft_3000',ascending=False)

temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scores,'Precision_scaling':precision_scores}).sort_values('Precision_scaling',ascending=False)

new_df = performance_df.merge(temp_df,on='Algorithm')

new_df_scaled = new_df.merge(temp_df,on='Algorithm')

temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accuracy_scores,'Precision_num_chars':precision_scores}).sort_values('Precision_num_chars',ascending=False)

new_df_scaled.merge(temp_df,on='Algorithm')

# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

from sklearn.ensemble import VotingClassifier

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

voting.fit(X_train,y_train)

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

# Applying stacking
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))

import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```

V. RESULT

In result we are using streamlit app for display the email is spam or not. Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Data scientists or machine learning engineers are not web developers and they're not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can display data and collect needed parameters for modeling. Streamlit allows you to create a stunning-looking application with only a few lines of code.

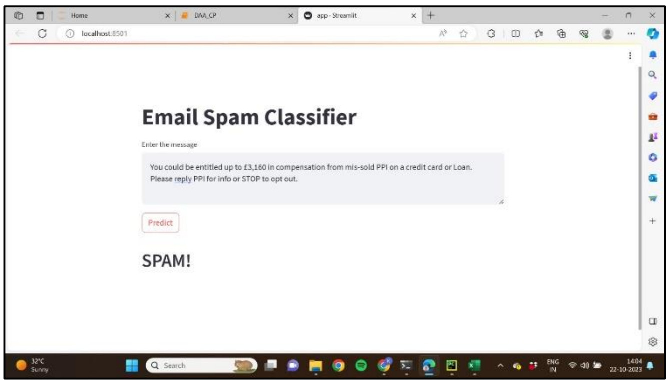
- **App.py**

```
import streamlit as st
import pickle
import string
from nltk.corpus import stopwords
```

```
import nltk
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()
def transform_text(text):
text = text.lower()
text = nltk.word_tokenize(text)

          y = []
          for i in text:
              if i.isalnum():
                  y.append(i)

          text = y[:]
          y.clear()
for i in text:
 if i not in stopwords.words('english') and i not in string.punctuation:
y.append(i)

          text = y[:]
          y.clear()

          for i in text:
              y.append(ps.stem(i))

          return " ".join(y)
tfidf = pickle.load(open('vectorizer.pkl','rb'))
model = pickle.load(open('model.pkl','rb'))

st.title("Email Spam Classifier")

input_sms = st.text_area("Enter the message")

if st.button('Predict')

 # 1. preprocess
 transformed_sms = transform_text(input_sms)
 # 2. vectorize
vector_input = tfidf.transform([transformed_sms])
# 3. predict
result = model.predict(vector_input)[0]
# 4. Display
 if result == 1:
st.header("SPAM!")
      else:
st.header("NOT Spam!")
```

A carefully curated email corpus is employed to train and test the model, ensuring its accuracy and robustness. The corpus contains a diverse set of email samples, including both spam and non-spam messages, allowing the system to generalize well to different types of emails.

The NLTK library is utilized for natural language processing tasks such as tokenization, stemming, and stop-word removal. This preprocessing step enhances the quality of the input data and contributes to the overall effectiveness of the spam filter.


Email Spam Classifier

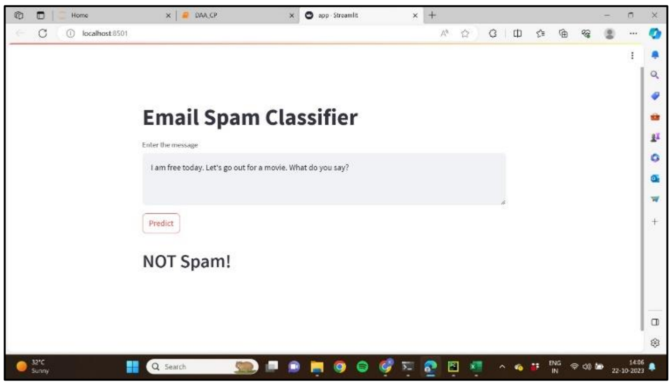
Enter the message

You could be entitled up to £3,160 in compensation from mis-sold PPI on a credit card or Loan. Please reply PPI for info or STOP to opt out.

Predict

SPAM!

Fig no 2. Email is Spam


Email Spam Classifier

Enter the message

I am free today. Let's go out for a movie. What do you say?

Predict

NOT Spam!

Fig no 3. Email is not Spam

VI. FUTURE SCOPE

The text suggests several improvements to a spam filter, including the use of advanced machine learning models, real-time email classification, a user feedback mechanism, multilingual support, customizable filters, improved performance, integration with email services, enhanced data collection, security features, education and user awareness, cross-platform compatibility, machine learning explainability, scalability, compliance with regulations, and research and development. The filter should be able to handle multiple languages and international character sets, provide users with adjustable sensitivity levels, and be able to adapt to evolving spam tactics. Additionally, the filter should be able to handle increasing email traffic and adapt to growing user bases. The project should also comply with relevant laws and regulations.

VII. APPLICATION

Email spam filters are used by major email service providers like Gmail, Outlook, and Yahoo to detect and filter out spam emails in real-time. They protect employees from phishing attacks, online marketplaces, social media platforms, messaging apps, financial services, government agencies, educational institutions, online retailers, subscription services, and healthcare providers. These filters ensure that users' inboxes are free of unsolicited and harmful content, maintain trust in online transactions, and prevent fraudulent schemes. They also protect users from unsolicited messages in messaging apps, financial services, government agencies, educational institutions, online retailers, subscription services, and healthcare providers.

VIII. CONCLUSION

The Email Spam Filter project aims to combat spam emails by using the Multinomial Naive Bayes algorithm, TF-IDF transformation, NLTK for text preprocessing, and a user-friendly Streamlit app for interaction. The system uses probabilities to distinguish between spam and non-spam emails, with TF-IDF transformation enhancing its analysis. The project's rigorous training and testing on a diverse email corpus ensure accuracy and generalization. The Natural Language Toolkit (NLTK) is used for text preprocessing, including lemmatization, stop-word removal, and lowercasing. The Streamlit app makes the Email Spam Filter accessible and user-friendly, allowing users to input emails for classification and display results intuitively. The project's future scope includes advanced machine learning models, real-time email classification, user feedback mechanisms, multilingual support, and integration with email services.

REFERENCES

[1**]** W. Li and S. Zeng, "A Vector Space Model based spam SMS filter," *2016 11th International Conference on Computer Science & Education (ICCSE)*, Nagoya, Japan, 2016, pp. 553-557, doi: 10.1109/ICCSE.2016.7581640.

[2] A. A. Alurkar *et al*., "A proposed data science approach for email spam classification using machine learning techniques," *2017 Internet of Things Business Models, Users, and Networks*, Copenhagen, Denmark, 2017, pp. 1-5, doi: 10.1109/CTTE.2017.8260935.

[3] M. K. Chae, A. Alsadoon, P. W. C. Prasad and A. Elchouemi, "Spam filtering email classification (SFECM) using gain and graph mining algorithm," *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, 2017, pp. 1-7, doi: 10.1109/CCWC.2017.7868411.

[4] W. -Y. Liu, L. Wang and T. Wang, "Online supervised learning from multi-field documents for email spam filtering," 2010 International Conference on Machine Learning and Cybernetics, Qingdao, China, 2010, pp. 3335-3340, doi: 10.1109/ICMLC.2010.5580676.

[5] M. RAZA, N. D. Jayasinghe and M. M. A. Muslam, "A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms," *2021 International Conference on Information Networking (ICOIN)*, Jeju Island, Korea (South), 2021, pp. 327-332, doi: 10.1109/ICOIN50884.2021.9334020.

[6] A. Akhtar, G. R. Tahir and K. Shakeel, "A mechanism to detect urdu Spam emails," *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, New York, NY, USA, 2017, pp. 168-172, doi: 10.1109/UEMCON.2017.8249019.