E-MAIL SPAM DETECTION APP USING MACHINE LEARNING

HARSH GUPTA

DATE: 21-12-23

ABSTRACT:

Nowadays communication plays a major role in everything be it professional or personal. Email communication service is being used extensively because of its free use services, low-cost operations, accessibility, and popularity. Emails have one major security flaw that is anyone can send an email to anyone just by getting their unique user id. This security flaw is being exploited by some businesses and ill-motivated persons for advertising, phishing, malicious purposes, and finally fraud. This produces a kind of email category called SPAM. Spam refers to any email that contains an advertisement, unrelated and frequent emails. These emails are increasing day by day in numbers. Studies show that around 55 percent of all emails are some kind of spam. A lot of effort is being put into this by service providers. Spam is evolving by changing the obvious markers of detection. Moreover, the spam detection of service providers can never be aggressive with classification because it may cause potential information loss to incase of a misclassification. To tackle this problem we present a new and efficient method to detect spam using machine learning and natural language processing. A tool that can detect and classify spam. In addition to that, it also provides information regarding the text provided in a quick view format for user convenience.

PROBLEM STATEMENT:

Today, Spam has become a major problem in communication over internet. It has been accounted that around 55% of all emails are reported as spam and the number has been growing steadily. Spam which is also known as unsolicited bulk email has led to the increasing use of email as email provides the perfect ways to send the unwanted advertisement or junk newsgroup posting at no cost for the sender. This chance has been extensively exploited by irresponsible organizations and resulting to clutter the mail boxes of millions of people all around the world. Spam has been a major concern given the offensive content of messages; spam is a waste of time. End user is at risk of deleting legitimate mail by mistake. Moreover, spam also impacted the economical which led some countries to adopt legislation. Text classification is used to determine the path of incoming mail/message either into inbox or straight to spam folder. It is the process of assigning categories to text according to its content. It is used to organized, structures and categorizes text. It can be done either manually or automatically. Machine learning automatically classifies the text in a much faster way than manual technique. Machine learning uses pre-labeled text to learn the different associations between pieces of text and it output. It used feature extraction to transform each text to numerical representation in form of vector which represents the frequency of word in predefined dictionary. Text classification is important to structure the unstructured and messy nature of text such as documents and spam messages in a cost-effective way. Machine learning can make more accurate precisions in real-time and help to improve the manual slow process to much better and faster analyzing big data. It is important especially to a company to analyze text data, help inform business decisions and even automate business processes. In this project, machine learning techniques are used to detect the spam message of a mail. Machine learning is where computers can learn to do something 10 without the need to explicitly program them for the task. It uses data and produces a program to perform a task such as classification. Compared to knowledge engineering, machine learning techniques require messages that have been successfully pre-classified. The pre-classified messages make the training dataset which will be used to fit the learning algorithm to the model in machine learning studio. A combination of algorithms is used to learn the classification rules from messages. These algorithms are used for classification of objects of different classes. These algorithms are provided with pre labeled data and an unknown text. After learning from the pre labelled data each of these algorithms predict which class the unknown text may belong to and the category predicted by majority is considered as final.

MARKET/CUSTOMER NEEDS ASSESSMENT:

1. Market Overview:

- **Email Communication Trends:** With the increasing reliance on email communication, there is a growing concern about the rise of unsolicited and potentially harmful emails, commonly known as spam.
- **Security and privacy concerns:** users and organizations are seeking robust solutions to protect against phishing attacks, malware distribution, and privacy breaches through spam emails.

2. Customer Needs:

- **High Accuracy:** Users and organizations demand spam detection solutions with high accuracy to effectively filter out malicious emails while minimizing false positives.
- **Real-Time Detection:** Timely identification and blocking of spam emails are critical. Real time detection capabilities ensure immediate protection against emerging threats.
- **User-Friendly Integration:** Seamless integration with popular email platforms and user-friendly interfaces essential to ensure widespread adoption and ease of use.

EXTERNAL SEARCH (INFORMATION/ REFERENCES):

- **Almeida, T. A., Gomez, H. F., & Yamasaki, A. (2010). Contributions to the study of SMS spam filtering: New collection and results. Journal of Machine Learning Research, 11, 3611-3628**. This study focuses on SMS spam filtering but provides insights into feature selection and classification algorithms applicable to email spam detection using machine learning.
- **Carreras, X., & Marquez, L. (2001). Boosting trees for anti-spam email filtering. In Proceedings of the Conference on Recent Advances in Natural Language Processing (pp. 9-15).** The authors propose a boosting-based approach for email spam filtering. The study discusses the use of decision trees as weak learners in the boosting algorithm
- **Kotsiantis, S., Tzelepis, G., & Pintelas, P. (2007). Email classification using association rule-based filtering. Applied Intelligence, 27(3), 239-250.** This research explores the use of association rule-based filtering techniques for email classification, focusing on spam detection. The study discusses feature selection and classification algorithms.
- **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. 55-62).** This influential study introduces a Bayesian approach to email spam filtering, known as the"Naive Bayes" algorithm. The research provides insights into the effectiveness of probabilistic classifiers for email spam detection.
- **Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G., & Spyropoulos, C. D. (2000). An evaluation of naive Bayesian anti-spam filtering. In Proceedings of the Workshop on Machine learning in the New Information Age (Vol. 1, No. 1-3, pp.9-17)**. This study evaluates the performance of the Naive Bayes algorithm for email spam filtering. It compares different feature representations and discusses the impact of different factors on classification accuracy.
- **Dalvi, N., Kumar, R., Pang, B., & Ramakrishnan, R. (2004). Adventure: A scalable distributed system for mining massive datasets. In Proceedings of the 30th International Conference on Very Large Data Bases (pp. 833-844).** This study presents Adventure, a scalable distributed system for mining massive datasets, including email spam filtering. The research highlights the challenges of processing large volumes of email data and proposes solutions
- **Platt, J. C. (1999). Using analytic QP and sparseness to speed training of support vector machines. In Advances in Neural Information Processing Systems (pp. 557- 563).** This paper discusses the use of support vector machines (SVM) for email spam filtering. It focuses on the optimization techniques to speed up the training process of SVM models.
- **Bharti, S. K., Singh, S., & Malhotra, A. (2019). Machine learning-based spam email detection using optimized features. In Proceedings of the International Conference on Advanced Computing and Intelligent Engineering (pp. 147-158).** Springer, Singapore. This research proposes a machine learning-based approach for email spam detection using optimized features. The study evaluates the performance of different classifiers and feature selection techniques.

LET'S SEE OUR DATASET:

In [17]:
```python
import numpy as np
import pandas as pd
```

In [26]:
```python
df = pd.read_csv(r"C:\Users\TUFF\Downloads\archive (9)\spam.csv",encoding="ISO-8859-1")
```

In [28]:
```python
df.sample(5)
```

Out[28]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
1212	ham	Yo, the game almost over? Want to go to walmar...	NaN	NaN	NaN
1230	ham	I want to send something that can sell fast. ...	NaN	NaN	NaN
523	ham	That's very rude, you on campus?	NaN	NaN	NaN
5348	ham	Do I? I thought I put it back in the box	NaN	NaN	NaN
4574	ham	Not directly behind... Abt 4 rows behind ì_...	NaN	NaN	NaN

In [30]:
```python
df.shape
```

Out[30]: (5572, 5)

MORE INFORMATION ABOUT DATASET:

Data cleaning

In [31]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

In [33]:
```python
#drop last 3 columns
df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],inplace=True)
```

In [34]:
```python
df.sample(5)
```

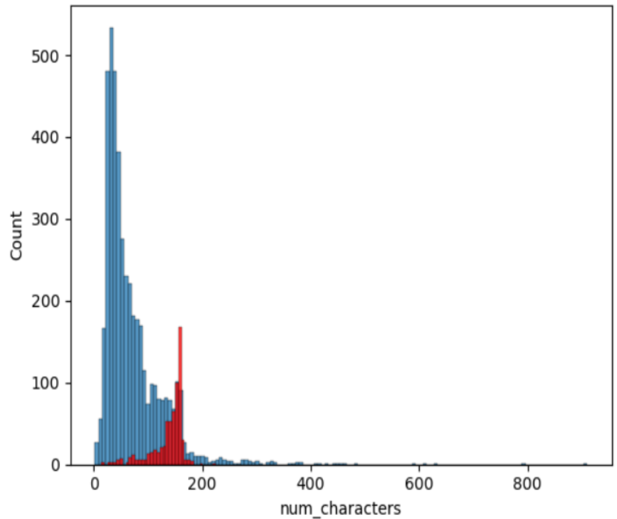
Out[34]:

	v1	v2
3323	ham	I don wake since. I checked that stuff and saw...
4349	ham	Yes. Rent is very expensive so its the way we ...
1565	ham	The <#> g that i saw a few days ago, th...
3106	ham	Hi. Happy New Year. I dont mean to intrude but...
4722	ham	HELLO PEACH! MY CAKE TASTS LUSH!

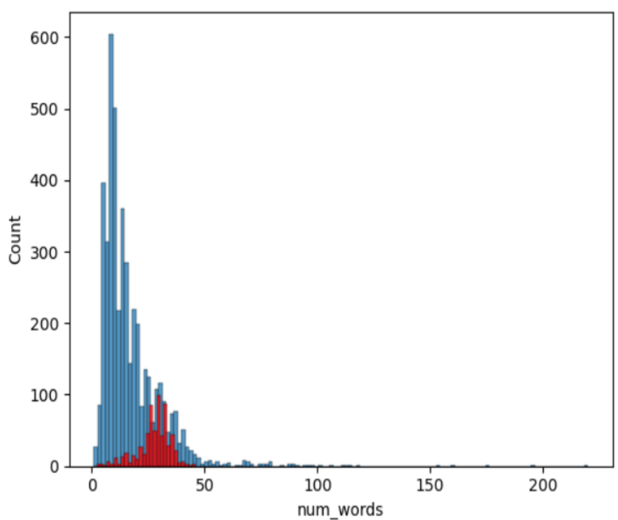
BENCHMARKING:

In [29]:
```
import seaborn as sns
```

In [30]:
```
sns.histplot(df[df['target']==0]['num_characters'])
sns.histplot(df[df['target']==1]['num_characters'],color='red')
plt.show()
```

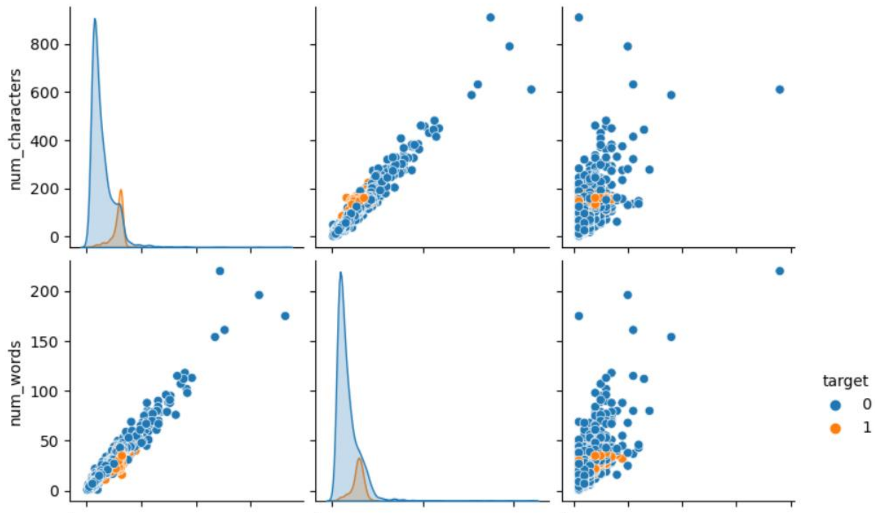


In [31]:
```
sns.histplot(df[df['target']==0]['num_words'])
sns.histplot(df[df['target']==1]['num_words'],color='red')
plt.show()
```



```
In [32]: sns.pairplot(df,hue='target')
```

```
Out[32]: <seaborn.axisgrid.PairGrid at 0x1830f11c7c0>
```

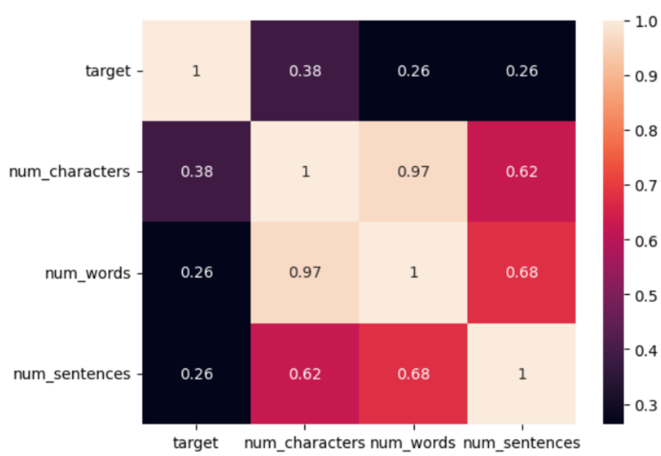


HEATMAP FOR BETTER UNDERSTANDING:

```
In [34]: sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\TUFF\AppData\Local\Temp\ipykernel_9800\4277794465.py:1: FutureWarning: The default value of numeric_only in DataFrame.
corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_on
ly to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

```
Out[34]: <Axes: >
```



APPLICABLE REGULATIONS:

The patents mentioned above might claim the technology used if the algorithms are not developed and optimized individually and for our requirements. Using a pre-existing model is off the table if it incurs a patent claim.

- Must provide access to the 3rd party websites to audit and monitor the authenticity and behavior of the service.
- Enabling open-source, academic and research community to audit the Algorithms and research on the efficacy of the product.
- Laws controlling data collection: Some websites might have a policy against collecting customer data in form of reviews and ratings.
- Must be responsible with the scraped data: It is quite essential to protect the privacy and intention with which the data was extracted.

BUSINESS MODEL:

The application overview has been presented below and it gives a basic structure of the application.

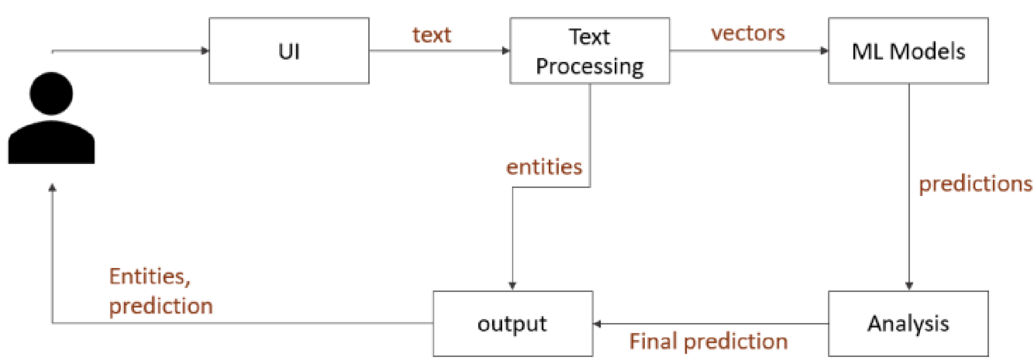


fig no. 4.1 Architecture

CONCEPT GENERTAION:

This product requires the tool of machine learning models to be written from scratch in order to suit our needs. . Tweaking these models for our use is less daunting than coding it up from scratch. A well trained model can either be repurposed or built. But building a model with the resources and data we have is dilatory but possible. The customer might want to spend the least amount of time giving input data. . This accuracy will take a little effort to nail, because it’s imprudent to rely purely on Classic Machine Learning algorithm.

MODEL BUILDING

In [56]:
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
tfidf=TfidfVectorizer()
```

In [57]:
```
X=tfidf.fit_transform(df['transformed_text']).toarray()
```

In [58]:
```
X.shape
```

Out[58]:
```
(5169, 6708)
```

In [59]:
```
y=df['target'].values
```

In [60]:
```
y
```

Out[60]:
```
array([0, 0, 1, ..., 0, 0, 0])
```

In [61]:
```
from sklearn.model_selection import train_test_split
```

In [62]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
```

In [63]:
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [64]:
```
gnb=GaussianNB()
mnb=MultinomialNB()
bnb=BernoulliNB()
```

In [65]:
```
gnb.fit(X_train,y_train)
y_pred1=gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
```

We will use five different models and we will finalize the model which will give good accuracy:

In [65]:
```
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))
```

```
0.8762088974854932
[[793 103]
 [ 25 113]]
0.5231481481481481
```

In [66]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
mnb=MultinomialNB()
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))
```

```
0.9593810444874274
[[896   0]
 [ 42  96]]
1.0
```

In [67]:
```
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))
```

```
0.9700193423597679
[[893   3]
 [ 28 110]]
0.9734513274336283
```

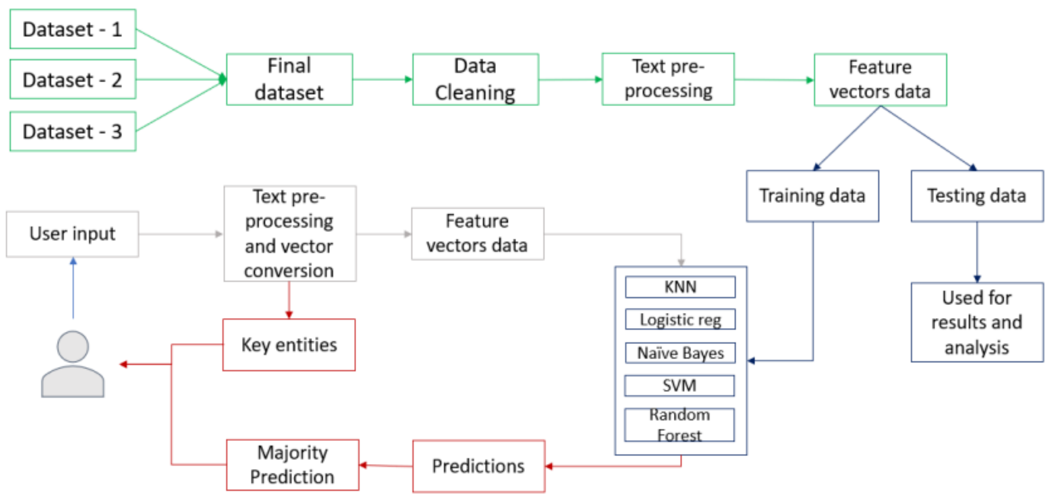
In [74]:
```
accuracy_scores = []
precision_scores = []

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)
```

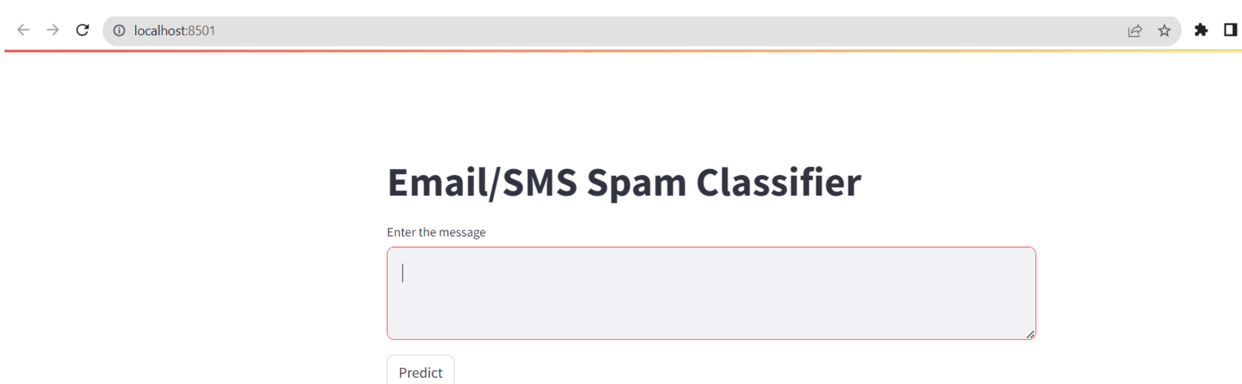
```
For  SVC
Accuracy -  0.9729206963249516
Precision -  0.9741379310344828
For  KN
Accuracy -  0.9003868471953579
Precision -  1.0
For  NB
Accuracy -  0.9593810444874274
Precision -  1.0
For  DT
Accuracy -  0.9361702127659575
Precision -  0.8461538461538461
For  LR
Accuracy -  0.9516441005802708
Precision -  0.94
For  RF
Accuracy -  0.971953578336557
Precision -  1.0
For  AdaBoost
Accuracy -  0.9613152804642167
Precision -  0.9454545454545454
For  BgC
Accuracy -  0.9584139264990329
```

Final Product Prototype (abstract) with Schematic Diagram:

The final product is a **GUI** based web application created using **Streamlit** where the customer just needs to put an Email and the machine learning model will tell whether the given Email is **spam or not.**

fig no. 4.2 Workflow





CONCLUSION AND FUTURE SCOPE:

Conclusion:

From the results obtained we can conclude that an ensemble machine learning model is more effective in detection and classification of spam than any individual algorithms. We can also conclude that TF-IDF (term frequency inverse document frequency) language model is more effective than Bag of words model in classification of spam when combined with several algorithms. And finally we can say that spam detection can get better if machine learning algorithms are combined and tuned to needs.

Future work:

There are numerous applications to machine learning and natural language processing and when combined they can solve some of the most troubling problems concerned with texts. This application can be scaled to intake text in bulk so that classification can be done more affectively in some public sites.

Other contexts such as negative, phishing, malicious, etc, can be used to train the model to filter things such as public comments in various social sites. This application can be converted to online type of machine learning system and can be easily updated with latest trends of spam and other mails so that the system can adapt to new types of spam emails and texts.

GITHUB LINK - https://github.com/haarsh567/E-Mail-Spam-Detection