Building A Smarter AI Powered Spam Classifier:

By: Kevin B

Register No: 211121104032

Introduction:

In the ever-evolving landscape of digital communication, the issue of unsolicited or unwanted messages, commonly known as spam, has become a pervasive challenge. The exponential growth of electronic communication channels, such as emails, social media platforms, and messaging apps, has led to an unprecedented influx of spam content. This surge not only clutters users' inboxes but also poses significant security risks and hampers productivity.

To combat this issue, a sophisticated approach is required—one that leverages cutting-edge technology and artificial intelligence. This paper introduces the concept of a Smart Powered AI Spam Classifier, a powerful tool designed to autonomously identify and categorize spam content across various digital communication platforms.

This advanced spam classifier capitalizes on the capabilities of artificial intelligence, drawing upon natural language processing, machine learning, and deep learning techniques. It combines these methodologies to create a robust and adaptable system capable of discerning between legitimate messages and unsolicited, potentially harmful content.

This paper aims to provide a comprehensive overview of the Smart Powered AI Spam Classifier, detailing its architecture, training process, and deployment strategies. Additionally, it explores the potential benefits and applications of this classifier in mitigating the impact of spam on individuals and organizations alike.

By harnessing the power of artificial intelligence, this classifier stands as a testament to the potential of technology to enhance digital communication experiences, making them safer, more efficient, and ultimately more enjoyable for users. Through this innovation, we embark on a journey towards a spam-free digital future, where users can engage with confidence and security in the online realm.

Exploration :

import numpy as np  
import pandas as pd

df = pd.read\_csv('spam.csv')

df.sample(5)

v1 v2 Unnamed: 2 \  
2464 ham They will pick up and drop in car.so no problem.. NaN   
1248 ham HI HUN! IM NOT COMIN 2NITE-TELL EVERY1 IM SORR... NaN   
1413 spam Dear U've been invited to XCHAT. This is our f... NaN   
2995 ham They released vday shirts and when u put it on... NaN   
4458 spam Welcome to UK-mobile-date this msg is FREE giv... NaN   
  
 Unnamed: 3 Unnamed: 4   
2464 NaN NaN   
1248 NaN NaN   
1413 NaN NaN   
2995 NaN NaN   
4458 NaN NaN

df.shape

(5572, 5)

# 1. Data cleaning  
# 2. EDA  
# 3. Text Preprocessing  
# 4. Model building  
# 5. Evaluation  
# 6. Improvement  
# 7. Website  
# 8. Deploy

## 1. Data Cleaning

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

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

df.sample(5)

v1 v2  
1947 ham The battery is for mr adewale my uncle. Aka Egbon  
2712 ham Hey you still want to go for yogasana? Coz if ...  
4428 ham Hey they r not watching movie tonight so i'll ...  
3944 ham I will be gentle princess! We will make sweet ...  
49 ham U don't know how stubborn I am. I didn't even ...

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

target text  
1418 ham Lmao. Take a pic and send it to me.  
2338 ham Alright, see you in a bit  
88 ham I'm really not up to it still tonight babe  
3735 ham Hows the street where the end of library walk is?  
3859 ham Yep. I do like the pink furniture tho.

from sklearn.preprocessing import LabelEncoder  
encoder = LabelEncoder()

df['target'] = encoder.fit\_transform(df['target'])

df.head()

target text  
0 0 Go until jurong point, crazy.. Available only ...  
1 0 Ok lar... Joking wif u oni...  
2 1 Free entry in 2 a wkly comp to win FA Cup fina...  
3 0 U dun say so early hor... U c already then say...  
4 0 Nah I don't think he goes to usf, he lives aro...

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

target 0  
text 0  
dtype: int64

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

403

# remove duplicates  
df = df.drop\_duplicates(keep='first')

df.duplicated().sum()

0

df.shape

(5169, 2)

## 2.EDA

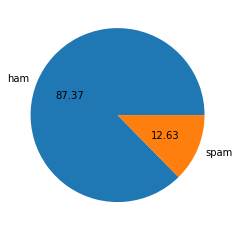
df.head()

target text  
0 0 Go until jurong point, crazy.. Available only ...  
1 0 Ok lar... Joking wif u oni...  
2 1 Free entry in 2 a wkly comp to win FA Cup fina...  
3 0 U dun say so early hor... U c already then say...  
4 0 Nah I don't think he goes to usf, he lives aro...

df['target'].value\_counts()

0 4516  
1 653  
Name: target, dtype: int64

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



# Data is imbalanced

import nltk

!pip install nltk

nltk.download('punkt')

[nltk\_data] Downloading package punkt to  
[nltk\_data] C:\Users\91842\AppData\Roaming\nltk\_data...  
[nltk\_data] Unzipping tokenizers\punkt.zip.

True

df['num\_characters'] = df['text'].apply(len)

df.head()

target text num\_characters  
0 0 Go until jurong point, crazy.. Available only ... 111  
1 0 Ok lar... Joking wif u oni... 29  
2 1 Free entry in 2 a wkly comp to win FA Cup fina... 155  
3 0 U dun say so early hor... U c already then say... 49  
4 0 Nah I don't think he goes to usf, he lives aro... 61

# num of words  
df['num\_words'] = df['text'].apply(lambda x:len(nltk.word\_tokenize(x)))

df.head()

target text num\_characters \  
0 0 Go until jurong point, crazy.. Available only ... 111   
1 0 Ok lar... Joking wif u oni... 29   
2 1 Free entry in 2 a wkly comp to win FA Cup fina... 155   
3 0 U dun say so early hor... U c already then say... 49   
4 0 Nah I don't think he goes to usf, he lives aro... 61   
  
 num\_words   
0 24   
1 8   
2 37   
3 13   
4 15

df['num\_sentences'] = df['text'].apply(lambda x:len(nltk.sent\_tokenize(x)))

df.head()

target text num\_characters \  
0 0 Go until jurong point, crazy.. Available only ... 111   
1 0 Ok lar... Joking wif u oni... 29   
2 1 Free entry in 2 a wkly comp to win FA Cup fina... 155   
3 0 U dun say so early hor... U c already then say... 49   
4 0 Nah I don't think he goes to usf, he lives aro... 61   
  
 num\_words num\_sentences   
0 24 2   
1 8 2   
2 37 2   
3 13 1   
4 15 1

df[['num\_characters','num\_words','num\_sentences']].describe()

num\_characters num\_words num\_sentences  
count 5169.000000 5169.000000 5169.000000  
mean 78.923776 18.456375 1.962275  
std 58.174846 13.323322 1.433892  
min 2.000000 1.000000 1.000000  
25% 36.000000 9.000000 1.000000  
50% 60.000000 15.000000 1.000000  
75% 117.000000 26.000000 2.000000  
max 910.000000 220.000000 38.000000

# ham  
df[df['target'] == 0][['num\_characters','num\_words','num\_sentences']].describe()

num\_characters num\_words num\_sentences  
count 4516.000000 4516.000000 4516.000000  
mean 70.456820 17.123339 1.815545  
std 56.356802 13.491315 1.364098  
min 2.000000 1.000000 1.000000  
25% 34.000000 8.000000 1.000000  
50% 52.000000 13.000000 1.000000  
75% 90.000000 22.000000 2.000000  
max 910.000000 220.000000 38.000000

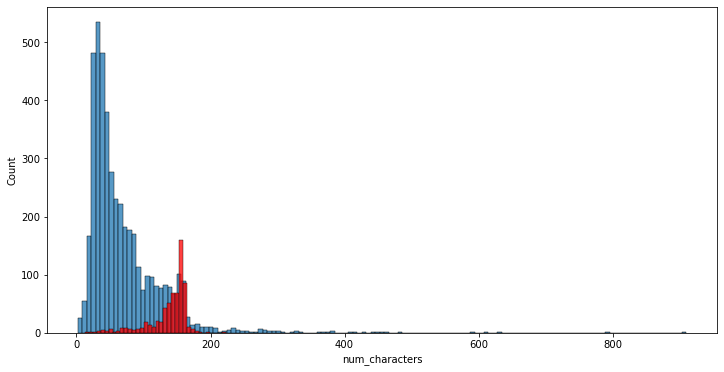
#spam  
df[df['target'] == 1][['num\_characters','num\_words','num\_sentences']].describe()

num\_characters num\_words num\_sentences  
count 653.000000 653.000000 653.000000  
mean 137.479326 27.675345 2.977029  
std 30.014336 7.011513 1.493676  
min 13.000000 2.000000 1.000000  
25% 131.000000 25.000000 2.000000  
50% 148.000000 29.000000 3.000000  
75% 157.000000 32.000000 4.000000  
max 223.000000 46.000000 9.000000

import seaborn as sns

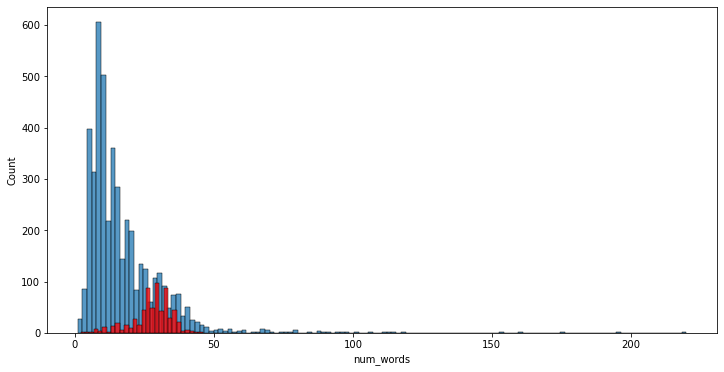
plt.figure(figsize=(12,6))  
sns.histplot(df[df['target'] == 0]['num\_characters'])  
sns.histplot(df[df['target'] == 1]['num\_characters'],color='red')

<AxesSubplot:xlabel='num\_characters', ylabel='Count'>



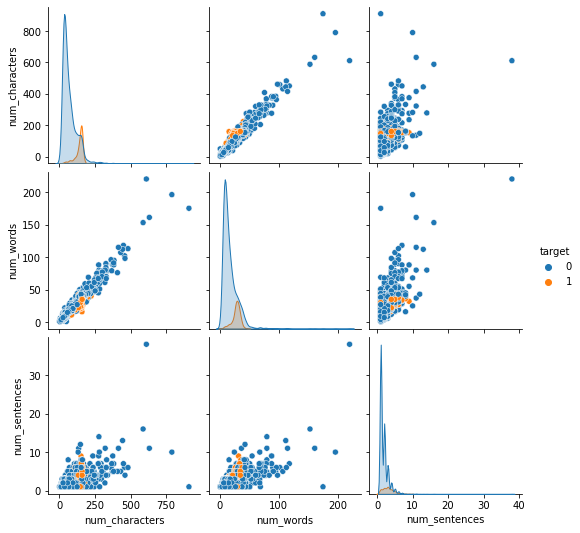
plt.figure(figsize=(12,6))  
sns.histplot(df[df['target'] == 0]['num\_words'])  
sns.histplot(df[df['target'] == 1]['num\_words'],color='red')

<AxesSubplot:xlabel='num\_words', ylabel='Count'>



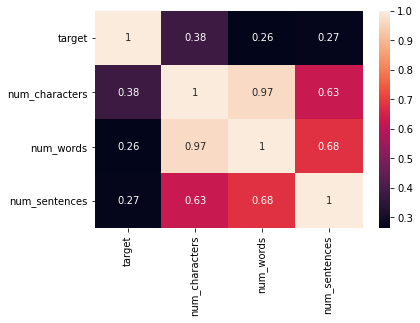
sns.pairplot(df,hue='target')

<seaborn.axisgrid.PairGrid at 0x16f88c4a4f0>



sns.heatmap(df.corr(),annot=True)

<AxesSubplot:>



## 3. Data Preprocessing

* Lower case
* Tokenization
* Removing special characters
* Removing stop words and punctuation
* Stemming

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.")

'gon na home soon want talk stuff anymor tonight k cri enough today'

df['text'][10]

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

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

'love'

df['transformed\_text'] = df['text'].apply(transform\_text)

df.head()

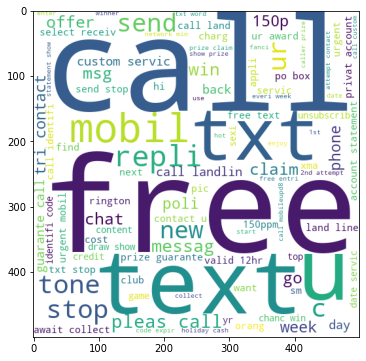
target text num\_characters \  
0 0 Go until jurong point, crazy.. Available only ... 111   
1 0 Ok lar... Joking wif u oni... 29   
2 1 Free entry in 2 a wkly comp to win FA Cup fina... 155   
3 0 U dun say so early hor... U c already then say... 49   
4 0 Nah I don't think he goes to usf, he lives aro... 61   
  
 num\_words num\_sentences transformed\_text   
0 24 2 go jurong point crazi avail bugi n great world...   
1 8 2 ok lar joke wif u oni   
2 37 2 free entri 2 wkli comp win fa cup final tkt 21...   
3 13 1 u dun say earli hor u c alreadi say   
4 15 1 nah think goe usf live around though

from wordcloud import WordCloud  
wc = WordCloud(width=500,height=500,min\_font\_size=10,background\_color='white')

spam\_wc = wc.generate(df[df['target'] == 1]['transformed\_text'].str.cat(sep=" "))

plt.figure(figsize=(15,6))  
plt.imshow(spam\_wc)

<matplotlib.image.AxesImage at 0x16f87ea8cd0>



ham\_wc = wc.generate(df[df['target'] == 0]['transformed\_text'].str.cat(sep=" "))

plt.figure(figsize=(15,6))  
plt.imshow(ham\_wc)

<matplotlib.image.AxesImage at 0x16f87f6c280>



df.head()

target text num\_characters \  
0 0 Go until jurong point, crazy.. Available only ... 111   
1 0 Ok lar... Joking wif u oni... 29   
2 1 Free entry in 2 a wkly comp to win FA Cup fina... 155   
3 0 U dun say so early hor... U c already then say... 49   
4 0 Nah I don't think he goes to usf, he lives aro... 61   
  
 num\_words num\_sentences transformed\_text   
0 24 2 go jurong point crazi avail bugi n great world...   
1 8 2 ok lar joke wif u oni   
2 37 2 free entri 2 wkli comp win fa cup final tkt 21...   
3 13 1 u dun say earli hor u c alreadi say   
4 15 1 nah think goe usf live around though

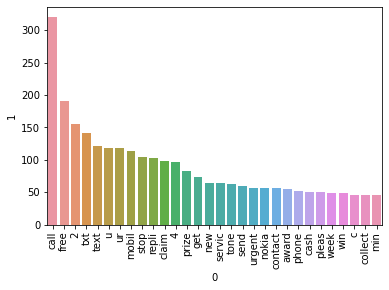
spam\_corpus = []  
for msg in df[df['target'] == 1]['transformed\_text'].tolist():  
 for word in msg.split():  
 spam\_corpus.append(word)

len(spam\_corpus)

9941

from collections import Counter  
sns.barplot(pd.DataFrame(Counter(spam\_corpus).most\_common(30))[0],pd.DataFrame(Counter(spam\_corpus).most\_common(30))[1])  
plt.xticks(rotation='vertical')  
plt.show()

C:\Users\91842\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
 warnings.warn(



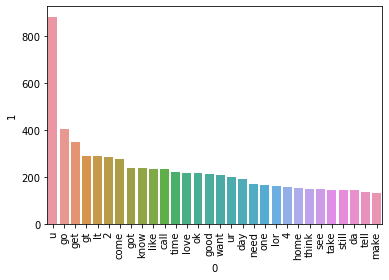
ham\_corpus = []  
for msg in df[df['target'] == 0]['transformed\_text'].tolist():  
 for word in msg.split():  
 ham\_corpus.append(word)

len(ham\_corpus)

35303

from collections import Counter  
sns.barplot(pd.DataFrame(Counter(ham\_corpus).most\_common(30))[0],pd.DataFrame(Counter(ham\_corpus).most\_common(30))[1])  
plt.xticks(rotation='vertical')  
plt.show()

C:\Users\91842\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
 warnings.warn(



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

target text num\_characters \  
0 0 Go until jurong point, crazy.. Available only ... 111   
1 0 Ok lar... Joking wif u oni... 29   
2 1 Free entry in 2 a wkly comp to win FA Cup fina... 155   
3 0 U dun say so early hor... U c already then say... 49   
4 0 Nah I don't think he goes to usf, he lives aro... 61   
  
 num\_words num\_sentences transformed\_text   
0 24 2 go jurong point crazi avail bugi n great world...   
1 8 2 ok lar joke wif u oni   
2 37 2 free entri 2 wkli comp win fa cup final tkt 21...   
3 13 1 u dun say earli hor u c alreadi say   
4 15 1 nah think goe usf live around though

## 4. 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

(5169, 3000)

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))

0.8916827852998066  
[[808 88]  
 [ 24 114]]  
0.5643564356435643

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))

0.971953578336557  
[[896 0]  
 [ 29 109]]  
1.0

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))

0.9835589941972921  
[[895 1]  
 [ 16 122]]  
0.991869918699187

# 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\_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  
}

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)

(0.9729206963249516, 0.9741379310344828)

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)

C:\Users\91842\anaconda3\lib\site-packages\sklearn\metrics\\_classification.py:1245: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))

For SVC  
Accuracy - 0.8665377176015474  
Precision - 0.0  
For KN  
Accuracy - 0.9284332688588007  
Precision - 0.7711864406779662  
For NB  
Accuracy - 0.9400386847195358  
Precision - 1.0  
For DT  
Accuracy - 0.9439071566731141  
Precision - 0.8773584905660378  
For LR  
Accuracy - 0.9613152804642167  
Precision - 0.9711538461538461  
For RF  
Accuracy - 0.9748549323017408  
Precision - 0.9827586206896551  
For AdaBoost  
Accuracy - 0.971953578336557  
Precision - 0.9504132231404959  
For BgC  
Accuracy - 0.9680851063829787  
Precision - 0.9133858267716536  
For ETC  
Accuracy - 0.97678916827853  
Precision - 0.975  
For GBDT  
Accuracy - 0.9487427466150871  
Precision - 0.9292929292929293

C:\Users\91842\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use\_label\_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num\_class - 1].  
 warnings.warn(label\_encoder\_deprecation\_msg, UserWarning)

[14:16:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64\_release\_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval\_metric if you'd like to restore the old behavior.  
For xgb  
Accuracy - 0.9700193423597679  
Precision - 0.9421487603305785

performance\_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy\_scores,'Precision':precision\_scores}).sort\_values('Precision',ascending=False)

performance\_df

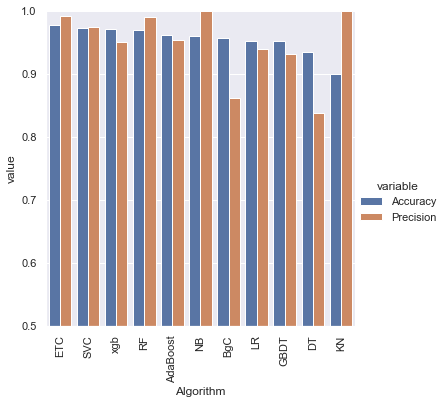
Algorithm Accuracy Precision  
1 KN 0.900387 1.000000  
2 NB 0.959381 1.000000  
8 ETC 0.977756 0.991453  
5 RF 0.970019 0.990826  
0 SVC 0.972921 0.974138  
6 AdaBoost 0.962282 0.954128  
10 xgb 0.971954 0.950413  
4 LR 0.951644 0.940000  
9 GBDT 0.951644 0.931373  
7 BgC 0.957447 0.861538  
3 DT 0.935203 0.838095

performance\_df1 = pd.melt(performance\_df, id\_vars = "Algorithm")

performance\_df1

Algorithm variable value  
0 ETC Accuracy 0.977756  
1 SVC Accuracy 0.972921  
2 xgb Accuracy 0.971954  
3 RF Accuracy 0.970019  
4 AdaBoost Accuracy 0.962282  
5 NB Accuracy 0.959381  
6 BgC Accuracy 0.957447  
7 LR Accuracy 0.951644  
8 GBDT Accuracy 0.951644  
9 DT Accuracy 0.935203  
10 KN Accuracy 0.900387  
11 ETC Precision 0.991453  
12 SVC Precision 0.974138  
13 xgb Precision 0.950413  
14 RF Precision 0.990826  
15 AdaBoost Precision 0.954128  
16 NB Precision 1.000000  
17 BgC Precision 0.861538  
18 LR Precision 0.940000  
19 GBDT Precision 0.931373  
20 DT Precision 0.838095  
21 KN Precision 1.000000

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')

Algorithm Accuracy Precision Accuracy\_max\_ft\_3000 \  
0 KN 0.900387 1.000000 0.905222   
1 NB 0.959381 1.000000 0.971954   
2 ETC 0.977756 0.991453 0.979691   
3 RF 0.970019 0.990826 0.975822   
4 SVC 0.972921 0.974138 0.974855   
5 AdaBoost 0.962282 0.954128 0.961315   
6 xgb 0.971954 0.950413 0.968085   
7 LR 0.951644 0.940000 0.956480   
8 GBDT 0.951644 0.931373 0.946809   
9 BgC 0.957447 0.861538 0.959381   
10 DT 0.935203 0.838095 0.931335   
  
 Precision\_max\_ft\_3000 Accuracy\_scaling Precision\_scaling \  
0 1.000000 0.905222 0.976190   
1 1.000000 0.978723 0.946154   
2 0.975610 0.979691 0.975610   
3 0.982906 0.975822 0.982906   
4 0.974576 0.971954 0.943089   
5 0.945455 0.961315 0.945455   
6 0.933884 0.968085 0.933884   
7 0.969697 0.967118 0.964286   
8 0.927835 0.946809 0.927835   
9 0.869231 0.959381 0.869231   
10 0.831683 0.932302 0.840000   
  
 Accuracy\_num\_chars Precision\_num\_chars   
0 0.928433 0.771186   
1 0.940039 1.000000   
2 0.976789 0.975000   
3 0.974855 0.982759   
4 0.866538 0.000000   
5 0.971954 0.950413   
6 0.970019 0.942149   
7 0.961315 0.971154   
8 0.948743 0.929293   
9 0.968085 0.913386   
10 0.943907 0.877358

# 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)

VotingClassifier(estimators=[('svm',  
 SVC(gamma=1.0, kernel='sigmoid',  
 probability=True)),  
 ('nb', MultinomialNB()),  
 ('et',  
 ExtraTreesClassifier(n\_estimators=50,  
 random\_state=2))],  
 voting='soft')

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

# 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))

Accuracy 0.9787234042553191  
Precision 0.9328358208955224

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

**Conclusion**

Thus, the design and innovation for the Building a Smarter AI-Powered Classifier has been done and completed successfully.