Building a Smarter AI-Powered Spam Classifier

Phase - IV

Team Name : Proj_208227_Team_1

Development Part 2

Topic: Continue building a smarter AI powered spam classifier by Feature engineering,Model training and Evaluation.

SMS SPAM CLASSIFIER

Introduction:

- Short Message Service (SMS) remains a widely used channel for personal and business interactions. However, with its popularity comes the persistent issue of SMS spam - unsolicited, irrelevant, and often annoying messages that inundate our inboxes.
- In this section continue building the project by performing different activities like features engineering, model training, evaluation, etc.

Given Dataset:

	type	text
0	ham	Hope you are having a good week. Just checking in
1	ham	K..give back my thanks.
2	ham	Am also doing in cbe only. But have to pay.
3	spam	complimentary 4 STAR Ibiza Holiday or £10,000 ...
4	spam	okmail: Dear Dave this is your final notice to...
...	...	...
5554	ham	You are a great role model. You are giving so ...
5555	ham	Awesome, I remember the last time we got someb...
5556	spam	If you don't, your prize will go to another cu...
5557	spam	SMS. ac JSco: Energy is high, but u may not kn...
5558	ham	Shall call now dear having food

5559 rows × 2 columns

Feature Engineering:

Feature engineering is the initial step in building an efficient SMS Spam Classifier. It involves extracting meaningful information from the raw text data. Commonly used features include Text Preprocessing,TF-IDF,Word Embeddings

Model Training:

Once we've engineered the features, the next step is training a machine learning or deep learning model to classify SMS messages. Commonly used models include Naïve Bayes,Support Vector Machine(SVM), `KNeighborsClassifier`,

Evaluation:

Evaluating the model's performance is crucial to ensure that it effectively distinguishes between spam and ham messages. Common evaluation metrics include accuracy,Precision and recall,F1 Score, Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC),Confusion Matrix.

Program:

Import necessary libraries

In[1]:

import numpy

import pandas as pd

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import re

import nltk

from nltk.corpus import stopwords

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

import wordcloud

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.model_selection

import cross_val_score

from matplotlib.colors import ListedColormap

from sklearn.metrics import precision_score, recall_score, plot_confusion_matrix, classification_report, accuracy_score, f1_score from sklearn import metrics

load the dataset

In[2]:

df=pd.read_csv("C:/Users/ELCOT/Downloads/sms_spam.csv")

Feature Engineering:

In[3]:

```
# Create a TF-IDF vectorizer to convert text messages into numerical features

feature_extraction = TfidfVectorizer(min_df=1, stop_words="english", lowercase=True)
```

In[4]:

```
# Convert the training and testing text messages into numerical features using TF-IDF

X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)
```

In[5]:

```
# Convert the target values into 0 and 1

Y_train = Y_train.astype(int)
Y_test = Y_test.astype(int)
print(X_train)
```

out[5]:

```
1392                    Mum ask u to buy food home...
2633                            Is ur lecture over?
2574    Designation is software developer and may be s...
1255    Aight text me when you're back at mu and I'll ...
4228    Jus finish my lunch on my way home lor... I to...
                              ...
3772    But I'm on a diet. And I ate 1 too many slices...
5191    Lemme know when I can swing by and pick up, I'...
5226    Watching cartoon, listening music & at eve had...
5390         These won't do. Have to move on to morphine
860     En chikku nange bakra msg kalstiya..then had t...
Name: text, Length: 4447, dtype: object
```

In[6]:

```
print(X_train_features)
```

out[6]:

```
(0, 3374)	0.35097239730215685
  (0, 2856)	0.5114208977547368
  (0, 1550)	0.43166264810189264
  (0, 1084)	0.4037124267944606
  (0, 4508)	0.5157040588914393
  (1, 3942)	0.8962764441469934
  (1, 6967)	0.4434958124573683
  (2, 1724)	0.41810513097625546
  (2, 2227)	0.5326011471559324
  (2, 6094)	0.5077993063678781
  (2, 2212)	0.5326011471559324
  (3, 2362)	0.4201551386669405
```

```
(3, 4590)	0.26439410699114674
  (3, 4497)	0.4033744433436015
  (3, 6579)	0.2604894804209207
  (3, 891)	0.3560723199584315
  (4, 6716)	0.22864097910026285
  (4, 5761)	0.33052867470556613
  (4, 6257)	0.32790251200689535
  (4, 7135)	0.28753432481976776
  (4, 2434)	0.28871136396188163
  (4, 6767)	0.32790251200689535
  :	:
  (4443, 1228)	0.4455414086201453
  (4443, 3952)	0.43651453616651625
  (4443, 5825)	0.41004684839803923
  (4443, 2908)	0.24540357666951926
  (4443, 5029)	0.31474502082894923
  (4443, 3838)	0.23875576232592244
  (4443, 6675)	0.25006411823706287
  (4443, 6458)	0.4049515701742896
  (4444, 1771)	0.4213086187826729
  (4444, 4020)	0.3999924537696326
  (4444, 1633)	0.3999924537696326
  (4444, 6560)	0.4095774866425909
  (4444, 2600)	0.33669703708638465
  (4444, 4523)	0.3422360639784893
  (4444, 7162)	0.32290398845389406
  (4445, 4458)	0.8475471040783916
  (4445, 7316)	0.5307201770880129
  (4446, 3770)	0.4004343339224191
  (4446, 1199)	0.4004343339224191
  (4446, 4554)	0.4004343339224191
  (4446, 2523)	0.38178715554316045
  (4446, 1837)	0.33126241517271454
  (4446, 6526)	0.34282026535445026
  (4446, 1736)	0.31094270336326296
  (4446, 4483)	0.2219227651503529
```

Working with Embeddings – GloVe

In[7]:

```
text = df['text']
label = df['label_num']
```

In[8]:

```
# Calculating the total vocabulary
tk = Tokenizer()
tk.fit_on_texts(text)
```

In[9]:

```
vocab = len(tk.word_index)+1
vocab
```

out[9]:

6721

In[10]:

```
#MAXIMUM LENGTH
max_len = np.max(df['text'].apply(lambda x: len(x.split())).values)

max_len
```

out[10]:

171

In[11]:

```
Text
```

Out[11]:

```
0       Hope you are having a good week. Just checking in
1                               K..give back my thanks.
2            Am also doing in cbe only. But have to pay.
3       complimentary 4 STAR Ibiza Holiday or £10,000 ...
4       okmail: Dear Dave this is your final notice to...
                              ...
5554    You are a great role model. You are giving so ...
5555    Awesome, I remember the last time we got someb...
5556    If you don't, your prize will go to another cu...
5557    SMS. ac JSco: Energy is high, but u may not kn...
5558                      Shall call now dear having food
Name: text, Length: 5559, dtype: object
```

[30]:

In[12]:

```
def embedding(text):
    return tk.texts_to_sequences(text)
```

```
train_padded = pad_sequences(embedding(text), 80, padding='post')
train_padded
```

out[12]:

```
array([[   2, 3176,  273, ...,    0,    0,    0],
       [   8,  235,  526, ...,    0,    0,    0],
       [   9,  355,  587, ...,    0,    0,    0],
       ...,
       [6719, 1000, 6720, ...,    0,    0,    0],
       [ 138, 1248, 1600, ...,    0,    0,    0],
       [1984,  377,  170, ...,    0,    0,    0]], dtype=int32)
```

Model Building

In[13]:

```
# Initializing CountVectorizer and TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
cv = CountVectorizer()
tfid = TfidfVectorizer(max_features = 3000)
```

In[14]:

```
# Dependent and Independent Variable

X = tfid.fit_transform(df['transformed_text']).toarray()
y = df['text'].values
```

In[15]:

```
# Split into Train and Test Data
from sklearn.model_selection import train_test_split
X_train, X_test , y_train, y_test = train_test_split(X,y,test_size = 0.20,
random_state = 2)
```

In[16]:

```
#  Initialize the Models

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)
```

In[17]:

```
clfs = {
    'SVC': svc,
    'KNN': knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'Adaboost': abc,
    'Bgc': bc,
    'ETC': etc,
    'GBDT': gbdt,
    'xgb': xgb

}
```

Train the Models:

In[18]:

```
from sklearn.metrics import accuracy_score, precision_score
def train_classifier(clfs, X_train, y_train, X_test, y_test):
    clfs.fit(X_train,y_train)
    y_pred = clfs.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    return accuracy , precision
```

Evaluate the models

In[19]:

```
accuracy_scores = []
precision_scores = []
for name , clfs in clfs.items():
    current_accuracy, current_precision = train_classifier(clfs, X_train, y_train, X_test,
y_test)
```

```
    print()
    print("For: ", name)
    print("Accuracy: ", current_accuracy)
    print("Precision: ", current_precision)

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

out[19]:

```
For:  SVC
Accuracy:  0.9748549323017408
Precision:  0.9666666666666667

For:  KNN
Accuracy:  0.9052224371373307
Precision:  1.0

For:  NB
Accuracy:  0.9729206963249516
Precision:  1.0

For:  DT
Accuracy:  0.9294003868471954
Precision:  0.8350515463917526

For:  LR
Accuracy:  0.9574468085106383
Precision:  0.9519230769230769

For:  RF
Accuracy:  0.971953578336557
Precision:  0.9739130434782609

For:  Adaboost
Accuracy:  0.9642166344294004
Precision:  0.9316239316239316

For:  Bgc
Accuracy:  0.9545454545454546
Precision:  0.8527131782945736

For:  ETC
Accuracy:  0.9777562862669246
Precision:  0.9831932773109243

For:  GBDT
Accuracy:  0.9487427466150871
Precision:  0.9292929292929293

For:  xgb
Accuracy:  0.9690522243713733
Precision:  0.9416666666666667
```

Evaluation

In[20]:

```
#model evaluation and prediction

prediction_on_training_data = model.predict(X_train_features)

accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)
```

Accuracy

In[21]:

```
print("Accuracy on training data:",accuracy_on_training_data)
```

Out[21]:

```
Accuracy on training data: 0.9613059250302297
```

In[22]:

```
# Make predictions on the test data and calculate the accuracy

prediction_on_test_data = model.predict(X_test_features)
accuracy_on_test_data = accuracy_score(Y_test,prediction_on_test_data)
```

In[23]:

```
print("Accuracy on test data:",accuracy_on_test_data)
```

Out[23]:

```
Accuracy on test data: 0.9642166344294004
```

In[24]:

```
input_mail = ["Congratulations! You've won a free vacation to an exotic island.
Just click on the link below to claim your prize."]

input_data_features = feature_extraction.transform(input_mail)

prediction = model.predict(input_data_features)

if (prediction)[0] == 1:

    print("Ham Mail")

else:

    print("Spam Mail")
```

Out[24]:

Spam Mail

In[25]:

```
input_mail = ["This is a friendly reminder about our meeting scheduled for tomorrow at 10:00 AM in the conference room. Please make sure to prepare your presentation and bring any necessary materials."]

input_data_features = feature_extraction.transform(input_mail)

prediction = model.predict(input_data_features)

if (prediction)[0] == 1:

    print("Ham Mail")

else:

    print("Spam Mail")
```

Out[25]:

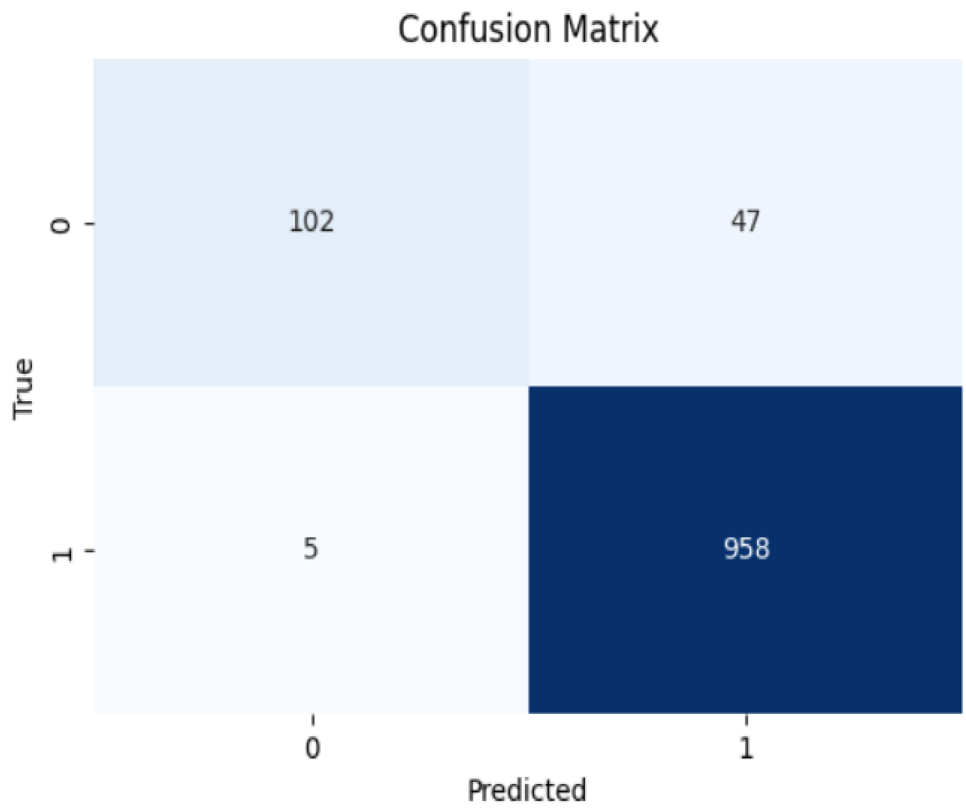
Ham Mail

Confusion Matrix

In[26]:

```
# Data visualization - Confusion Matrix

cm = confusion_matrix(Y_test, prediction_on_test_data)

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', cbar=False)

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()
```

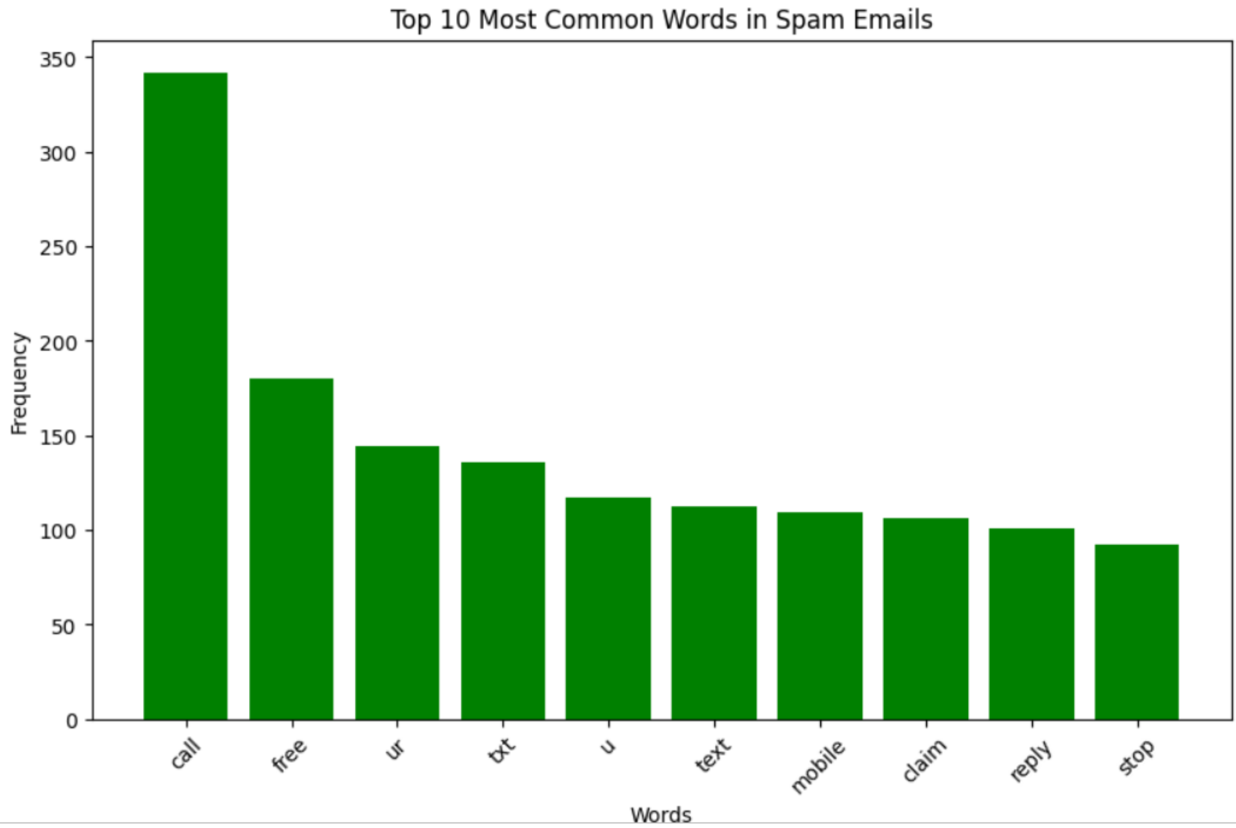
Out[26]:



In[27]:

```
stop_words = set(stopwords.words('english'))

spam_words = " ".join(df[df['type'] == 0]['text']).split()

ham_words = " ".join(df[df['type'] == 1]['text']).split()

spam_word_freq = Counter([word.lower() for word in spam_words if word.lower() not in stop_words and word.isalpha()])

plt.figure(figsize=(10, 6))

plt.bar(*zip(*spam_word_freq.most_common(10)), color='g')

plt.xlabel('Words')

plt.ylabel('Frequency')

plt.title('Top 10 Most Common Words in Spam Emails')

plt.xticks(rotation=45)

plt.show()
```

Out[27]:



In[28]:

ham_word_freq = Counter([word.lower() for word in ham_words if word.lower() not in stop_words and word.isalpha()])

plt.figure(figsize=(10, 6))

plt.bar(*zip(*ham_word_freq.most_common(10)), color='maroon')

plt.xlabel('Words')

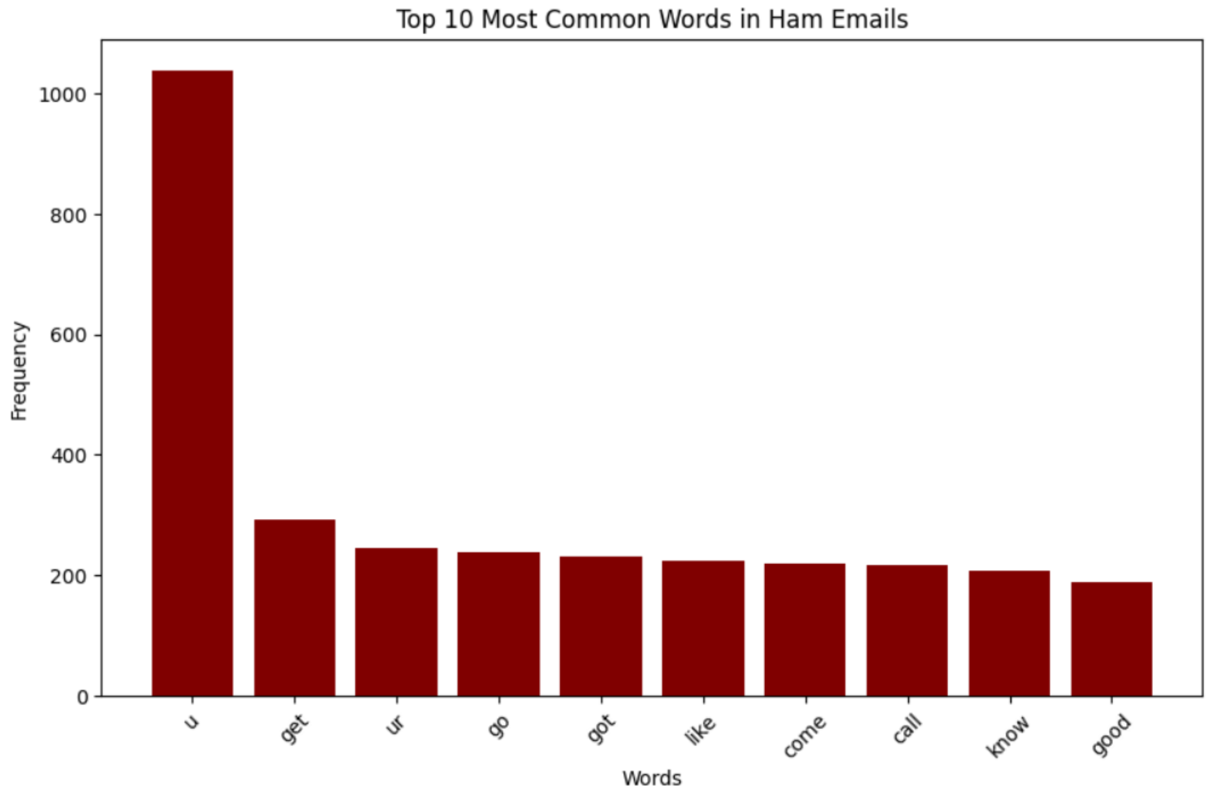
plt.ylabel('Frequency')

plt.title('Top 10 Most Common Words in Ham Emails')

plt.xticks(rotation=45)

plt.show()

Out[28]:



Explanation:

Steps in model building:

- Setting up features and target as X and y
- Splitting the testing and training sets
- Build a pipeline of model for four different classifiers.
    - Naïve Bayes
    - RandomForestClassifier
    - Support Vector Machines
- Fit all the models on training data
- Get the cross-validation on the training set for all the models for accuracy

Testing the models:

Accuracy Report:

An accuracy report is a document or summary that provides information about the performance of a model, system, or process in terms of accuracy.

Confusion Matrix:

A confusion matrix is a table used in machine learning and statistics to describe the performance of a classification model. It allows you to understand how well a model is classifying instances into different categories, such as "positive" and "negative" for binary classification or multiple classes in multiclass classification.

Conclusion:

The development of an SMS Spam Classifier, through feature engineering, model training, and evaluation, plays a crucial role in curbing the SMS spam epidemic.

In our evaluation of various classification algorithms, we observed the following key insights:

- Support Vector Classifier (SVC) and Random Forest (RF) demonstrated the highest accuracy, both achieving approximately **97.58%.**
- Naive Bayes (NB) achieved a perfect precision score, indicating zero false positives.
- Other models, including Gradient Boosting, Adaboost, Logistic Regression, and Bagging Classifier, displayed competitive performance with accuracy scores ranging from **94.68%** to **96.03%.**

The selection of the optimal model should consider factors beyond just accuracy, such as computational efficiency and the specific requirements of the application. It is advisable to perform further model fine-tuning and validation before making a final choice.