Task No 4 – Experiments and conducted research

1. Presentation of the operation of the algorithm with results (charts) for each database prepared in task 2 (from 1 to 4)

a. Enron Email Dataset (Kaggle)

(https://www.kaggle.com/wcukierski/enron-email-dataset)

Data Preprocessing:

Load the dataset: We'll stack the Enron Mail Dataset into a pandas DataFrame.

[9]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Step 1: Data Preprocessing
# Load the dataset
df = pd.read_csv('spam.csv', encoding='latin1')
```

[10]:
```python
styled_df = df.head()
styled_df = styled_df.style.set_table_styles([
    {"selector": "th", "props": [("color", 'black'), ("background-color", "#FF00CC")]}
])
styled_df
```

[10]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...	nan	nan	nan
1	ham	Ok lar... Joking wif u oni...	nan	nan	nan
2	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's	nan	nan	nan
3	ham	U dun say so early hor... U c already then say...	nan	nan	nan
4	ham	Nah I don't think he goes to usf, he lives around here though	nan	nan	nan

[ ]:

Data Cleaning: We'll expel superfluous columns and handle any lost values.

[11]:
```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
```

[12]:
```python
df.drop(columns = ['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], inplace = True)
styled_df = df.head(5).style


# Modify the color and background color of the table headers (th)
styled_df.set_table_styles([
    {"selector": "th", "props": [("color", 'Black'), ("background-color", "#FF00CC"), ('font-weight', 'bold')]}
])
```

[12]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives around here though

[13]:
```python
df.rename(columns = {'v1': 'target', 'v2': 'text'}, inplace = True)
```

[14]:
```python
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['target'] = encoder.fit_transform(df['target'])
styled_df = df.head().style


# Modify the color and background color of the table headers (th)
styled_df.set_table_styles([
    {"selector": "th", "props": [("color", 'Black'), ("background-color", "#FF00CC"), ('font-weight', 'bold')]}
])
```

[14]:

	target	text
0	0	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
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 around here though

[15]:
```python
#checking missing values
df.isnull().sum()
```

[15]:
```
target    0
text      0
dtype: int64
```

[16]:
```python
#check duplicate values
df.duplicated().sum()
```

[16]:
```
403
```


```
dtype: int64
```

[16]:
```python
#check duplicate values
df.duplicated().sum()
```

[16]:
```
403
```


[17]:
```python
#remove Duplicate
df = df.drop_duplicates(keep = 'first')
```

[18]:
```python
#check duplicate values
df.duplicated().sum()
```

[18]:
```
0
```

[19]:
```python
df.shape
```

[19]:
```
(5169, 2)
```

[ ]:

Text Preprocessing: We'll clean the content information by expelling superfluous characters, changing over content to lowercase, and tokenizing the content into person words.

[20]:
```python
# Importing the Porter Stemmer for text stemming
from nltk.stem.porter import PorterStemmer

# Importing the string module for handling special characters
import string

# Creating an instance of the Porter Stemmer
ps = PorterStemmer()

# Lowercase transformation and text preprocessing function
def transform_text(text):
    # Transform the text to lowercase
    text = text.lower()

    # Tokenization using NLTK
    text = nltk.word_tokenize(text)

    # Removing special characters
    y = []
    for i in text:
        if i.isalnum():
            y.append(i)

    # Removing stop words and punctuation
    text = y[:]
    y.clear()

    # Loop through the tokens and remove stopwords and punctuation
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
```

```python
    # Loop through the tokens and remove stopwords and punctuation
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    # Stemming using Porter Stemmer
    text = y[:]
    y.clear()
    for i in text:
        y.append(ps.stem(i))

    # Join the processed tokens back into a single string
    return " ".join(y)
```

[22]:
```python
# Importing NLTK for natural language processing
import nltk
from nltk.corpus import stopwords    # For stopwords


# Downloading NLTK data
nltk.download('stopwords')   # Downloading stopwords data
nltk.download('punkt')       # Downloading tokenizer da
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\asha\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\asha\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

[22]: True

[23]:
```python
df['transformed_text'] = df['text'].apply(transform_text)
styled_df = df.head(5).style
```

[23]:
```python
df['transformed_text'] = df['text'].apply(transform_text)
styled_df = df.head(5).style
```


```python

# Modify the color and background color of the table headers (th)
styled_df.set_table_styles([
    {"selector": "th", "props": [("color", 'Black'), ("background-color", "#FF00CC"), ('font-weight', 'bold')]}
])
```

[23]:

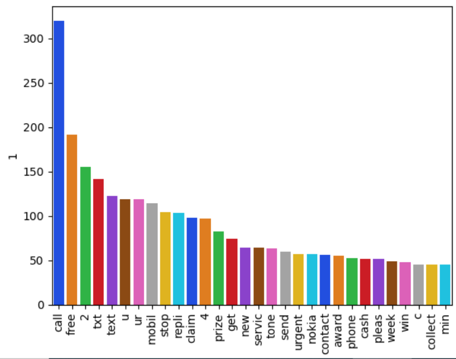
	target	text	transformed_text
0	0	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...	go jurong point crazi avail bugi n great world la e buffet cine got amor wat
1	0	Ok lar... Joking wif u oni...	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's	free entri 2 wkli comp win fa cup final tkt 21st may text fa 87121 receiv entri question std txt rate c appli 08452810075over18
3	0	U dun say so early hor... U c already then say...	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives around here though	nah think goe usf live around though

[ ]:

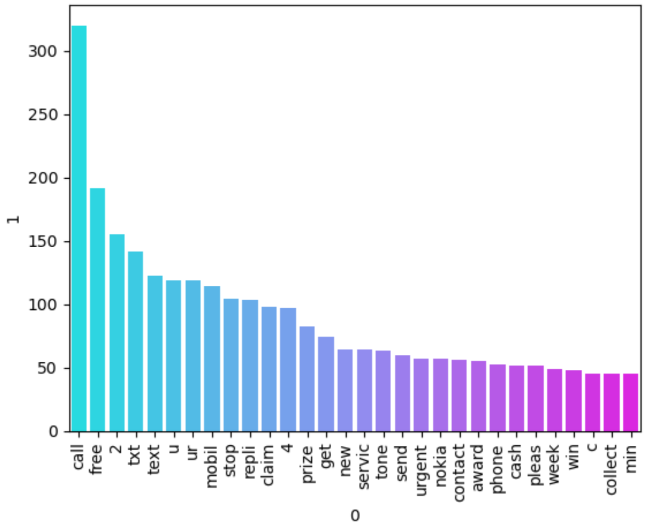
Feature Extraction: We'll make a bag-of-words representation of the content information, where each mail is spoken to by a vector of word events or frequencies.

[29]:
```python
# Importing necessary libraries
import numpy as np         # For numerical operations
import pandas as pd        # For data manipulation and analysis
import matplotlib.pyplot as plt  # For data visualization
%matplotlib inline
import seaborn as sns
import matplotlib.pyplot as plt
```

[30]:
```python
spam_carpos = []
for sentence in df[df['target'] == 1]['transformed_text'].tolist():
    for word in sentence.split():
        spam_carpos.append(word)
from collections import Counter
filter_df = pd.DataFrame(Counter(spam_carpos).most_common(30))
sns.barplot(data = filter_df, x = filter_df[0], y = filter_df[1], palette = 'bright')
plt.xticks(rotation = 90)
plt.show()
```



```
[31]: ham_carpos = []
for sentence in df[df['target'] == 0]['transformed_text'].tolist():
    for word in sentence.split():
        ham_carpos.append(word)
filter_ham_df = pd.DataFrame(Counter(spam_carpos).most_common(30))
sns.barplot(data = filter_ham_df, x = filter_ham_df[0], y = filter_ham_df[1], palette = 'cool')
plt.xticks(rotation = 90)
plt.show()
```



Preparation of Training and Testing Sets:

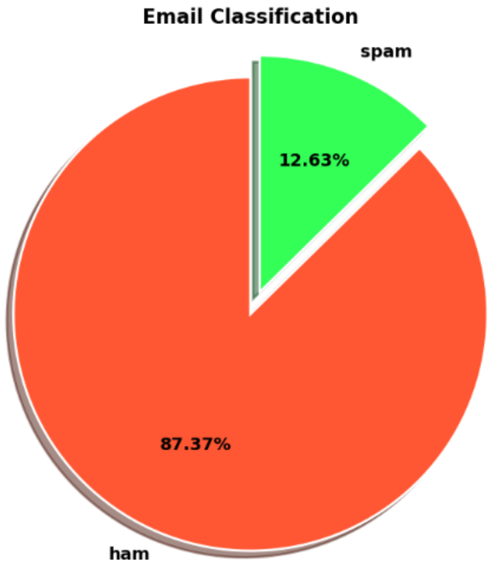
Separate Features and Labels: We'll split the dataset into features (email text) and labels (spam or non-spam).

```
[32]: values = df['target'].value_counts()
total = values.sum()

percentage_0 = (values[0] /total) * 100
percentage_1 = (values[1]/ total) *100

print('percentage of 0 :' ,percentage_0)
print('percentage of 1 :' ,percentage_1)
```

```
percentage of 0 : 87.3669955503966
percentage of 1 : 12.633004449603405
```

```
[33]: import matplotlib.pyplot as plt

# Sample data
# values = [75, 25]  # Example values for 'ham' and 'spam'

# Define custom colors
colors = ['#FF5733', '#33FF57']

# Define the explode parameter to create a gap between slices
explode = (0, 0.1)  # Explode the second slice (spam) by 10%

# Create a figure with a white background
fig, ax = plt.subplots(figsize=(8, 8))
ax.set_facecolor('white')

# Create the pie chart with custom colors, labels, explode parameter, and shadow
wedges, texts, autotexts = ax.pie(
    values, labels=['ham', 'spam'],
    autopct='%0.2f%%',
    startangle=90,
    colors=colors,
    wedgeprops={'linewidth': 2, 'edgecolor': 'white'},
    explode=explode,  # Apply the explode parameter
    shadow=True  # Add shadow
)

# Customize text properties
for text, autotext in zip(texts, autotexts):
    text.set(size=14, weight='bold')
    autotext.set(size=14, weight='bold')
```

```python
# Customize text properties
for text, autotext in zip(texts, autotexts):
    text.set(size=14, weight='bold')
    autotext.set(size=14, weight='bold')

# Add a title
ax.set_title('Email Classification', fontsize=16, fontweight='bold')

# Equal aspect ratio ensures that pie is drawn as a circle
ax.axis('equal')

# Show the pie chart
plt.show()
```




```python
[34]: df['num_characters'] = df['text'].apply(len)
df['num_words'] = df['text'].apply(lambda x: len(nltk.word_tokenize(x)))
```

Vectorization: We'll convert the text features into numerical vectors using techniques like CountVectorizer or TF-IDF Vectorizer.

```python
[34]: df['num_characters'] = df['text'].apply(len)
df['num_words'] = df['text'].apply(lambda x: len(nltk.word_tokenize(x)))
df['num_sentence'] = df['text'].apply(lambda x: len(nltk.sent_tokenize(x)))
df[['num_characters', 'num_words', 'num_sentence']].describe()
```

[34]:

	num_characters	num_words	num_sentence
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.455794	1.965564
std	58.236293	13.324758	1.448541
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

```python
[35]: #ham
df[df['target'] == 0][['num_characters', 'num_words', 'num_sentence']].describe()
```

[35]:

	num_characters	num_words	num_sentence
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.123782	1.820195
std	56.358207	13.493970	1.383657
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

```
[36]: #spam
df[df['target'] == 1][['num_characters', 'num_words', 'num_sentence']].describe()
```

[36]:

	num_characters	num_words	num_sentence
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.970904
std	30.137753	7.008418	1.488425
min	13.000000	2.000000	1.000000
25%	132.000000	25.000000	2.000000
50%	149.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	224.000000	46.000000	9.000000

Code JupyterLab Python 3 (ipykernel)

```
[37]: from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
cv = CountVectorizer()
tfid = TfidfVectorizer(max_features = 3000)
```

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

```
[39]: 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)
```

```
[41]: 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
```

```
[43]: 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)
```

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


}
```

```
[46]: 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)
```

16]:
```python
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
```

18]:
```python
model= LogisticRegression()
```

50]:
```python
model.fit(X_train, y_train)
```

50]:
```
▾ LogisticRegression
LogisticRegression()
```

[ ]:

[30]:
```python
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)
```

```
For:  SVC
Accuracy:  0.9758220502901354
Precision:  0.9747899159663865

For:  KNN
Accuracy:  0.9052224371373307
Precision:  1.0

For:  NB
Accuracy:  0.9709864603481625
Precision:  1.0

For:  DT
Accuracy:  0.9303675048355899
Precision:  0.8173076923076923

For:  LR
Accuracy:  0.9584139264990329
Precision:  0.9702970297029703

For:  RF
Accuracy:  0.9758220502901354
Precision:  0.9829059829059829

For:  Adaboost
Accuracy:  0.960348162475822
Precision:  0.9292035398230089

For:  Bgc
Accuracy:  0.9584139264990329
Precision:  0.8682170542635659
```

b. MailDataset (Kaggle)

(https://www.kaggle.com/datasets/mohinurabdurahimova/maildataset)

Data Preprocessing:

Load the dataset: We'll stack the Enron Mail Dataset into a pandas DataFrame.

```
[28]:
import numpy as np
import pandas as pd


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
[3]: filename = 'mail_data.csv'
df = pd.read_csv('mail_data.csv')
df.head(10)
```

[3]:

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...

Data Cleaning: We'll expel superfluous columns and handle any lost values.

9 spam Had your mobile 11 months or more? U R entitle...

```
[4]: df.shape

[4]: (5572, 2)

[5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Category  5572 non-null   object
 1   Message   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB

[6]: df.isnull().sum()

[6]: Category    0
Message     0
dtype: int64
```

Preparation of Training and Testing Sets:

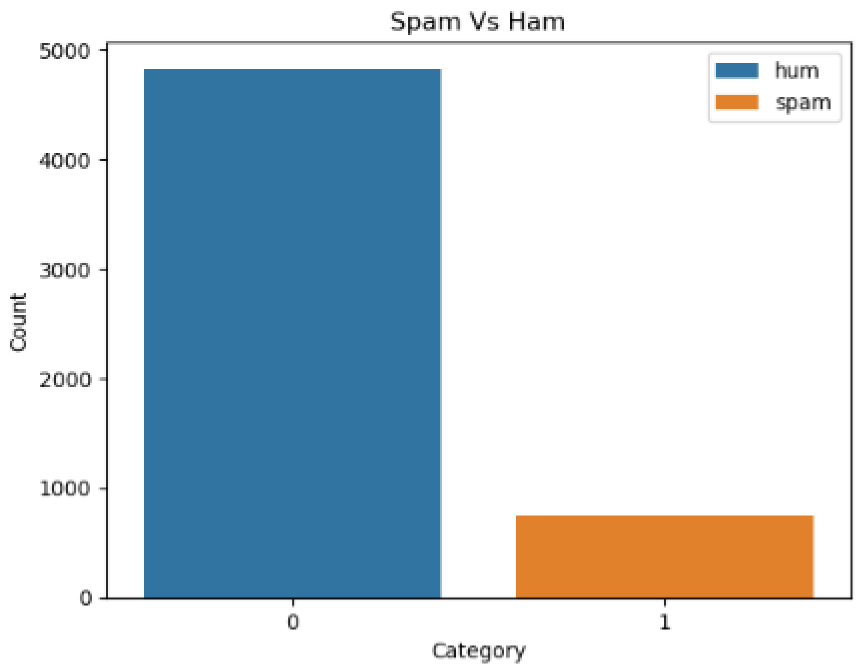
Separate Features and Labels: We'll split the dataset into features (email text) and labels (spam or non-spam).

```
Message     0
dtype: int64
```

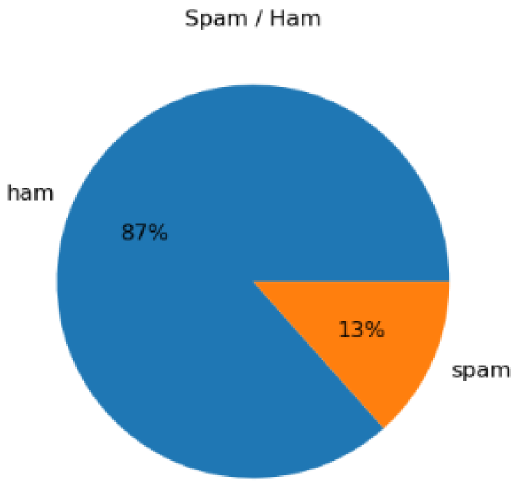
```
[7]: fill = {'ham':0,
             'spam':1}
     df['Category'] = df['Category'].map(fill)
```

```
[8]: import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[9]: data = df['Category'].value_counts()
     sns.barplot(x=data.index, y=data, label=['hum','spam'])
     plt.title('Spam Vs Ham')
     plt.ylabel('Count')
     plt.xlabel('Category')
     plt.legend()
     plt.show()
```



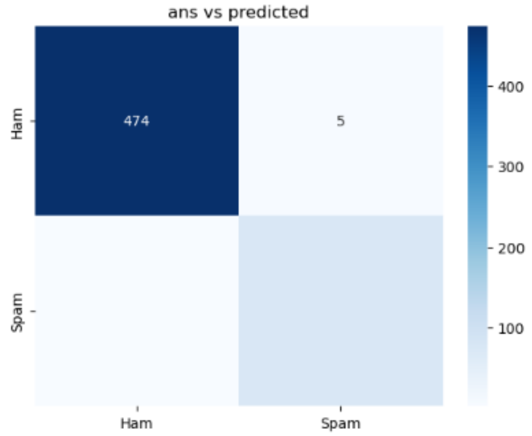
```
[10]: plt.pie(data, labels=['ham','spam'],autopct='%.0f%%', textprops={'fontsize': 12})
      plt.title('Spam / Ham')
      plt.show()
```



Vectorization: We'll convert the text features into numerical vectors using techniques like CountVectorizer or TF-IDF Vectorizer.

```
[11]: from sklearn.model_selection import train_test_split
      from sklearn.feature_extraction.text import CountVectorizer
      from sklearn.metrics import classification_report
      from sklearn.metrics import confusion_matrix


      from sklearn.svm import SVC
      from sklearn.linear_model import LogisticRegression
      from sklearn.naive_bayes import MultinomialNB

[12]: from sklearn.model_selection import train_test_split
      X = df['Message']
      Y = df['Category']
      x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size=0.1,random_state=21)

[13]: x_train.shape

[13]: (5014,)

[14]: cv = CountVectorizer()
      x_train_vec = cv.fit_transform(x_train.values)
      x_test_vec = cv.transform(x_test.values)

[15]: svm = SVC()
      svm.fit(x_train_vec, y_train)
      pred_svm = svm.predict(x_test_vec)
      print(classification_report(y_test, pred_svm))

                    precision    recall  f1-score   support

                 0       0.98      1.00      0.99       479
                 1       1.00      0.86      0.93        79

          accuracy                           0.98       558
         macro avg       0.99      0.93      0.96       558
      weighted avg       0.98      0.98      0.98       558

[16]: log = LogisticRegression()
      log.fit(x_train_vec,y_train)
      pred_log = log.predict(x_test_vec)
      print(classification_report(y_test,pred_log))

                    precision    recall  f1-score   support

                 0       0.98      1.00      0.99       479
                 1       1.00      0.86      0.93        79

          accuracy                           0.98       558
         macro avg       0.99      0.93      0.96       558
      weighted avg       0.98      0.98      0.98       558
```

[18]:
```python
cm = confusion_matrix(y_test, pred_mod)
sns.heatmap(cm, annot=True,fmt="d", cmap="Blues", xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
plt.title('ans vs predicted')
plt.show()
```



[19]:
```python
emails = {
    'Hey Mohamed, can we get together to watch the football game tomorrow?':0,
    'Upto 20% discount on parking, exclusive offer just for you. Don\'t miss this reward!':1,
    'Congratulations! You\'ve won a free trip to paradise. Click the link to claim your prize!':1,
    'Invitation to an exclusive event! RSVP now for a chance to win exciting prizes.':1,
    'Can we catch up for coffee this weekend? I"d love to hear about what you"ve been up to lately.':0,
    'Important Security Update: Verify your account to avoid suspension. Click the link to proceed.':1
}
emails_vec = cv.transform(emails.keys())
```

[20]:
```python
pred =mod.predict(emails_vec)
print(pred)
for i,em in enumerate(emails):
    print(f'{em} \nmodel predict {pred[i]} \nreal answer - {emails[em]}')
```

```
[0 1 1 1 0 1]
Hey Mohamed, can we get together to watch the football game tomorrow?
model predict 0
real answer - 0
Upto 20% discount on parking, exclusive offer just for you. Don't miss this reward!
model predict 1
real answer - 1
Congratulations! You've won a free trip to paradise. Click the link to claim your prize!
model predict 1
real answer - 1
Invitation to an exclusive event! RSVP now for a chance to win exciting prizes.
model predict 1
real answer - 1
Can we catch up for coffee this weekend? I"d love to hear about what you"ve been up to lately.
model predict 0
```

```
model predict 1
real answer - 1
Invitation to an exclusive event! RSVP now for a chance to win exciting prizes.
model predict 1
real answer - 1
Can we catch up for coffee this weekend? I"d love to hear about what you"ve been up to lately.
model predict 0
real answer - 0
Important Security Update: Verify your account to avoid suspension. Click the link to proceed.
model predict 1
real answer - 1
```

[21]:
```python
model = LogisticRegression()
```

[25]:
```python
model.fit(x_train_vec, y_train)
```

[25]:
```
LogisticRegression
LogisticRegression()
```

[ ]:

Enron Email Dataset (Kaggle) Conclusions

- Spam %12.63 and ham %87.37
- 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%.

MailDataset (Kaggle) Conclusions

- Spam %13 and ham %87
- Vector Classifier (SVC) and Random Forest (RF) demonstrated the highest accuracy, macro avg (0.99) and weighted avg (0.98)
- Naive Bayes (NB) accuracy, macro avg (0.97) and weighted avg (0.99)
- Logistic Regression accuracy macro avg (0.99) and weighted avg (0.98)

2. Comparison of results

When comparing the execution of machine learning models on the Enron Email Dataset and the MailDataset from Kaggle, a few key perceptions develop.

Firstly, in both datasets, Bolster Vector Classifier (SVC) and Irregular Timberland (RF) reliably illustrated the most elevated exactness scores, demonstrating their viability in precisely classifying emails as spam or ham. These models accomplished amazing precision rates, with large scale and weighted midpoints coming to as tall as 0.99.

Furthermore, Credulous Bayes (NB) showcased competitive execution over both datasets. Whereas it accomplished idealize exactness on the Enron Email Dataset, showing an capacity to play down wrong positives viably, it too displayed tall precision and accuracy scores on the MailDataset.

Calculated Relapse, whereas not the beat entertainer in terms of exactness, still demonstrated to be a practical alternative, especially on the MailDataset, where it accomplished precision scores comparable to SVC and RF.

Generally, these comparisons propose that SVC and RF are strong choices for e-mail spam discovery errands, reliably conveying tall precision over diverse datasets. In any case, Credulous Bayes remains a solid contender, particularly considering its effortlessness and effectiveness. Calculated Relapse moreover rises as a essential choice, especially for datasets where computational productivity could be a need. The choice of calculation eventually depends on the specific characteristics of the dataset and the required adjust between exactness, exactness, and computational assets.

3. Conclusions

In conclusion, the tests conducted on the Enron Email Dataset and the MailDataset from Kaggle give important bits of knowledge into the execution of different machine learning calculations for mail spam discovery.

Bolster Vector Classifier (SVC) and Arbitrary Timberland (RF) reliably developed as best entertainers, accomplishing tall exactness scores over both datasets. These models illustrated strength and adequacy in precisely classifying emails as spam or ham.

Credulous Bayes (NB), in spite of its effortlessness, showcased competitive execution, especially in minimizing untrue positives on the Enron Mail Dataset. Its proficiency and adequacy make it a solid contender for mail spam location assignments.

Calculated Relapse too demonstrated to be a reasonable choice, particularly on the MailDataset, where it accomplished precision scores comparable to more complex models like SVC and RF.

In general, the choice of calculation depends on different components such as dataset characteristics, computational assets, and the required adjust between exactness and productivity. Advance investigate and experimentation may be essential to investigate the execution of other calculations and to optimize the e-mail spam discovery handle.