TELEGRAM SPAM OR HAM DETECTION USING NLP

ABOUT DATASET

The Telegram Spam or Ham dataset is designed to classify messages as either spam or ham (non-spam). This dataset typically includes messages from the Telegram messaging platform, labeled accordingly to help train and evaluate machine learning models for spam detection. This dataset is valuable for those interested in natural language processing (NLP) and machine learning applications related to spam detection. It provides a practical way to apply various text classification techniques and evaluate their effectiveness in a real-world scenario. It contains text messages labeled as ‘spam’ or ‘ham’, facilitating the training and evaluation of machine learning models for spam detection. This dataset is ideal for experimenting with natural language processing techniques and building models to automatically identify spam messages.

AIM: To identify whether a given Telegram message is spam or ham.

In [66]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

In [67]:
```
df=pd.read_csv('dataset.csv')
```

In [68]:
```
df
```

Out[68]:

	text_type	text
0	spam	naturally irresistible your corporate identity...
1	spam	the stock trading gunslinger fanny is merrill ...
2	spam	unbelievable new homes made easy im wanting to...
3	spam	4 color printing special request additional in...
4	spam	do not have money get software cds from here s...
...	...	...
20343	ham	/ban
20344	ham	/ban
20345	ham	/ban
20346	ham	Kaisi hii
20347	ham	Shock q

20348 rows × 2 columns

PREPROCESSING

In [69]:
```
df.head()
```

Out[69]:

	text_type	text
0	spam	naturally irresistible your corporate identity...
1	spam	the stock trading gunslinger fanny is merrill ...
2	spam	unbelievable new homes made easy im wanting to...
3	spam	4 color printing special request additional in...
4	spam	do not have money get software cds from here s...

In [70]:
```
df.tail()
```

Out[70]:

	text_type	text
20343	ham	/ban
20344	ham	/ban
20345	ham	/ban
20346	ham	Kaisi hii
20347	ham	Shock q

In [71]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20348 entries, 0 to 20347
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   text_type  20348 non-null  object
 1   text       20348 non-null  object
dtypes: object(2)
memory usage: 318.1+ KB
```

In [72]:
```
df.shape
```

Out[72]:
```
(20348, 2)
```

In [73]:
```
df.dtypes
```

Out[73]:
```
text_type    object
text         object
dtype: object
```

In [74]:
```
df.nunique()
```

Out[74]:
```
text_type        2
text         20334
dtype: int64
```

In [75]:
```
df.isna().sum()
```

Out[75]:
```
text_type    0
text         0
dtype: int64
```

LABEL ENCODING

In [76]:
```
le_data=LabelEncoder()
model=le_data.fit_transform(df['text_type'])
df['text_type']=model
```

In [77]:
```
model
```

Out[77]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [78]:
```
df.dtypes
```

Out[78]:
```
text_type     int32
text         object
dtype: object
```

In [79]:
```
df
```

Out[79]:

	text_type	text
0	1	naturally irresistible your corporate identity...
1	1	the stock trading gunslinger fanny is merrill ...
2	1	unbelievable new homes made easy im wanting to...
3	1	4 color printing special request additional in...
4	1	do not have money get software cds from here s...
...	...	...
20343	0	/ban
20344	0	/ban
20345	0	/ban
20346	0	Kaisi hii
20347	0	Shock q

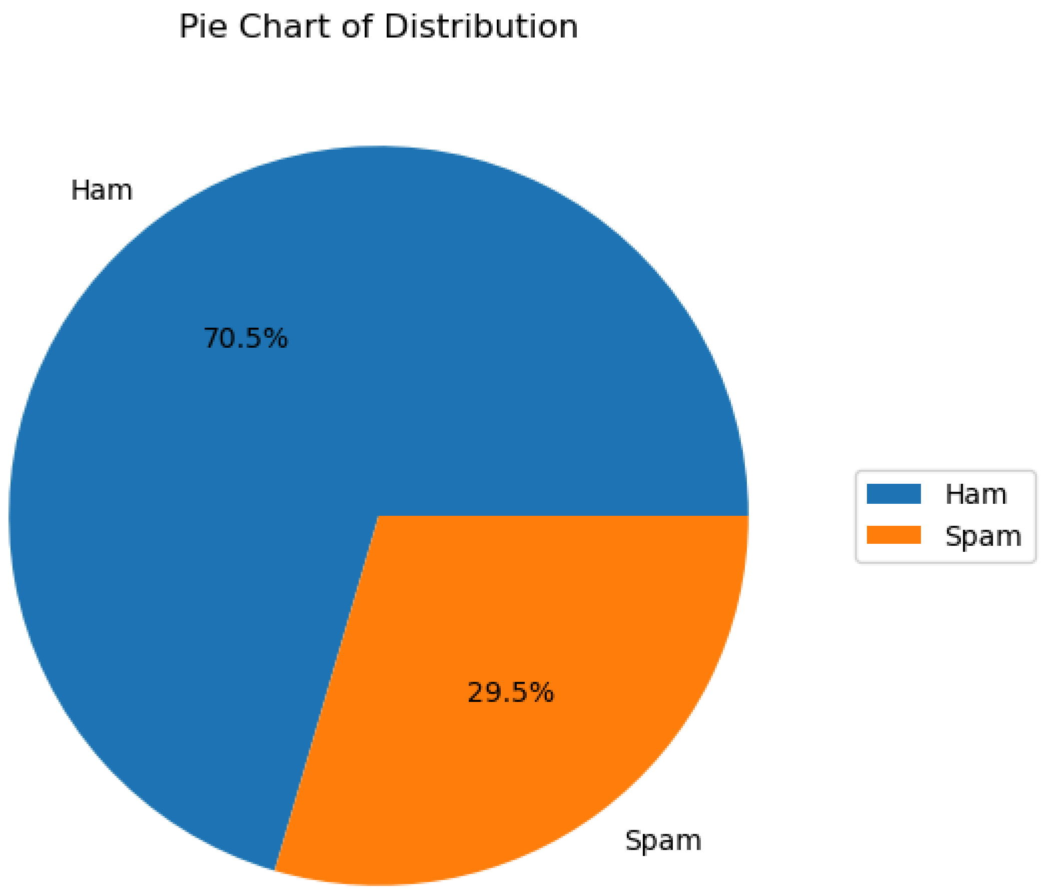
20348 rows × 2 columns

A text_type of 1 indicates that the message is a Spam, whereas a text_type of 0 signifies that the message is not spam (Ham)

In [80]:
```
df['text_type'].value_counts()
```

Out[80]:
```
text_type
0    14337
1     6011
Name: count, dtype: int64
```

In [111]:
```
# Calculate the value counts of the 'category' column
category_counts = df['text_type'].value_counts()
# Pie chart
plt.figure(figsize=(8,6))
labels=['Ham','Spam']
plt.pie(category_counts,labels=labels,autopct='%1.1f%%')
plt.title('Pie Chart of Distribution')
# # Add legend
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```



REMOVING STOPWORDS,SPECIAL CHARACTERS

In [89]:
```
import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [105]:
```
stp_words=stopwords.words('english')
def clean_text(text):
    cleantext="  ".join(word for word in text.split() if word not in stp_words)
    return cleantext
df['text']=df['text'].apply(clean_text)
def remove_numbers(text):
    return re.sub(r'\d+','',text)
df['text']=df['text'].apply(remove_numbers)
print(df.head())
```

```
   text_type                                               text  \
0          1  naturally  irresistible  corporate  identity  ...
1          1  stock  trading  gunslinger  fanny  merrill  mu...
2          1  unbelievable  new  homes  made  easy  im  want...
3          1  color  printing  special  request  additional ...
4          1  money  get  software  cds  software  compatibi...

                                    tokenized_column  \
0  [naturally, irresistible, corporate, identity,...
1  [stock, trading, gunslinger, fanny, merrill, m...
2  [unbelievable, new, homes, made, easy, im, wan...
3  [color, printing, special, request, additional...
4  [money, get, software, cds, software, compatib...

                                        stemmed_text  \
0  natur irresist corpor ident lt realli hard rec...
1  stock trade gunsling fanni merril muzo colza a...
2  unbeliev new home made easi im want show homeo...
3  color print special request addit inform click...
4  money get softwar cd softwar compat great grow...

                                     lemmatized_text
0  naturally irresistible corporate identity lt r...
1  stock trading gunslinger fanny merrill muzo co...
2  unbelievable new home made easy im wanting sho...
3  color printing special request additional info...
4  money get software cd software compatibility g...
```

TOKENISATION ,STEMMING AND LEMMATIZATION

In [107]:
```
from nltk.stem import PorterStemmer,WordNetLemmatizer
from nltk.tokenize import word_tokenize
import nltk
```

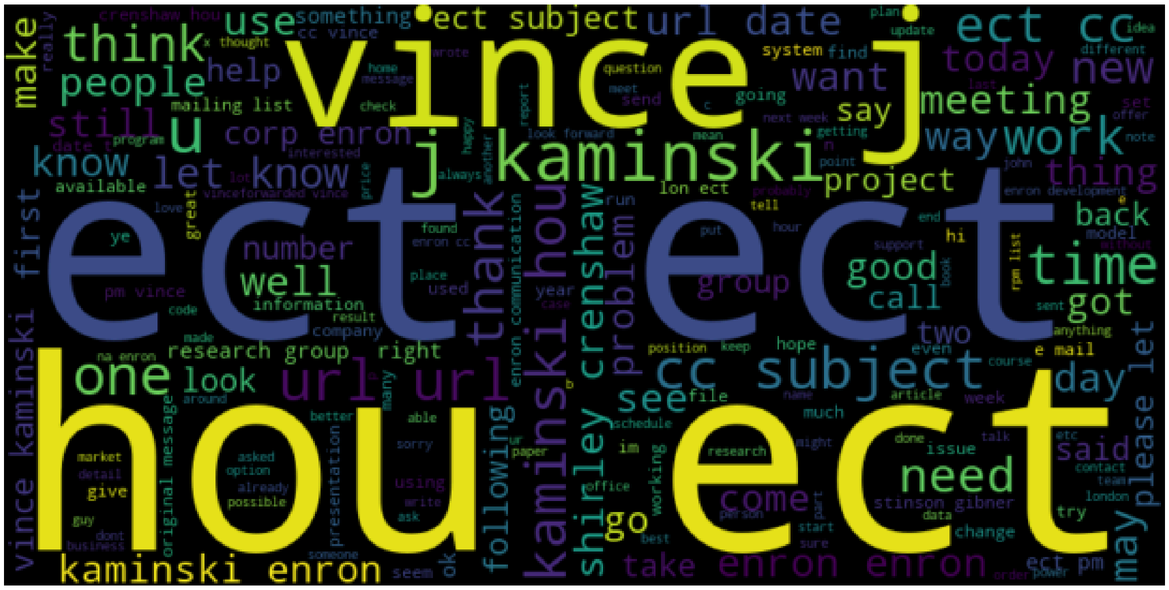
In [109]:
```
stemmer=PorterStemmer()
lemmatizer=WordNetLemmatizer()
```

In [110]:
```
def lower_case_and_tokenize(text):
    text=text.lower()
    tokens=word_tokenize(text)
    return tokens
def stem_text(text):
    tokens=word_tokenize(text)
    return ' '.join([stemmer.stem(token) for token in tokens])
def lemmatize_text(text):
    tokens=word_tokenize(text)
    return ' '.join([lemmatizer.lemmatize(token) for token in tokens])
df['tokenized_column']=df['text'].apply(lower_case_and_tokenize)
df['stemmed_text']=df['text'].apply(stem_text)
df['lemmatized_text']=df['text'].apply(lemmatize_text)
print(df.head())
```

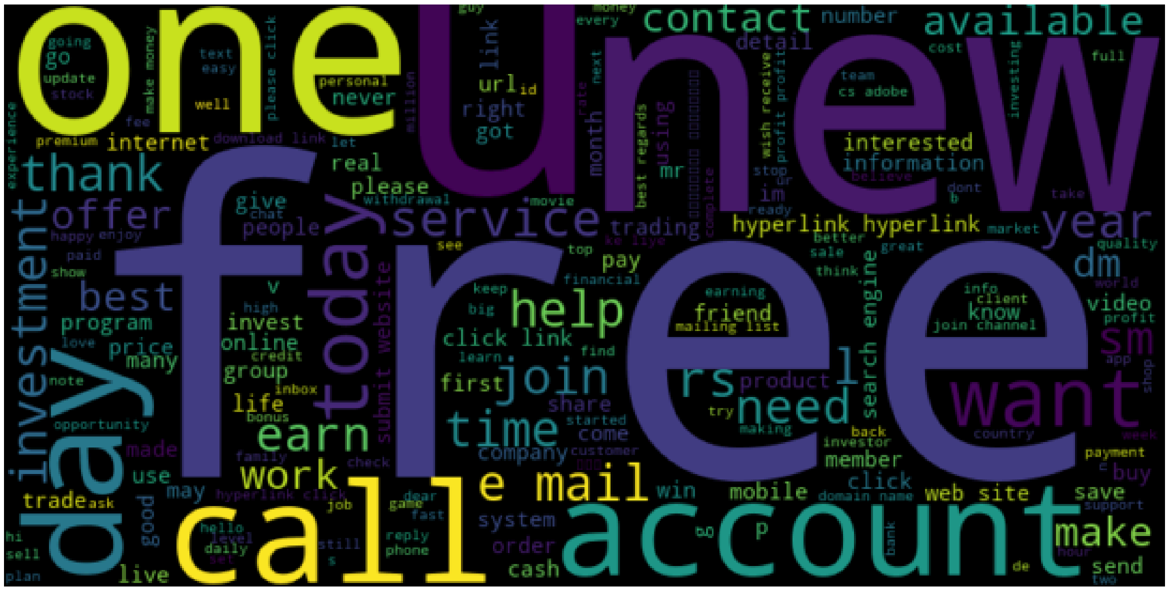
```
   text_type                                               text  \
0          1  naturally  irresistible  corporate  identity  ...
1          1  stock  trading  gunslinger  fanny  merrill  mu...
2          1  unbelievable  new  homes  made  easy  im  want...
3          1  color  printing  special  request  additional ...
4          1  money  get  software  cds  software  compatibi...

                                    tokenized_column  \
0  [naturally, irresistible, corporate, identity,...
1  [stock, trading, gunslinger, fanny, merrill, m...
2  [unbelievable, new, homes, made, easy, im, wan...
3  [color, printing, special, request, additional...
4  [money, get, software, cds, software, compatib...

                                        stemmed_text  \
0  natur irresist corpor ident lt realli hard rec...
1  stock trade gunsling fanni merril muzo colza a...
2  unbeliev new home made easi im want show homeo...
3  color print special request addit inform click...
4  money get softwar cd softwar compat great grow...

                                     lemmatized_text
0  naturally irresistible corporate identity lt r...
1  stock trading gunslinger fanny merrill muzo co...
2  unbelievable new home made easy im wanting sho...
3  color printing special request additional info...
4  money get software cd software compatibility g...
```

CREATING A WORD CLOUD

In [42]:
```
consolidated=' '.join(word for word in df['text'][df['text_type']==0].astype(st
wordCloud=WordCloud(width=800,height=400,random_state=21)
plt.figure(figsize=(8,6))
plt.imshow(wordCloud.generate(consolidated),interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [39]:
```
consolidated=' '.join(word for word in df['text'][df['text_type']==1].astype(st
wordCloud=WordCloud(width=800,height=400,random_state=21)
plt.figure(figsize=(8,6))
plt.imshow(wordCloud.generate(consolidated),interpolation='bilinear')
plt.axis('off')
plt.show()
```



VECTORIZATION

In [40]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [41]:
```
tf=TfidfVectorizer()
X=tf.fit_transform(df['text'])
```

In [42]:
```
print(X)
```

```
  (0, 13753)	0.09839569157349609
  (0, 50166)	0.06701231080701202
  (0, 17624)	0.0877682484116645
  (0, 20698)	0.11571594165116575
  (0, 38738)	0.09721748634965262
  (0, 45564)	0.10311136257142375
  (0, 13846)	0.1067162535970773
  (0, 25049)	0.10582574242835895
  (0, 16627)	0.1275729902363153
  (0, 40244)	0.11702238337883315
  (0, 16933)	0.0819418692765788
  (0, 45052)	0.12215516621608143
  (0, 29779)	0.11479927401251358
  (0, 35277)	0.07818558338270445
  (0, 30340)	0.07667389856048695
  (0, 23128)	0.09602066536165323
  (0, 14346)	0.12822595979688872
  (0, 9146)	0.1080488517946785
  (0, 12580)	0.10156481269479387
  (0, 29559)	0.06735278462962481
  (0, 17853)	0.09930817141444329
  (0, 30812)	0.09255774919660822
  (0, 34251)	0.12125502323356341
  (0, 24298)	0.1289055884339165
  (0, 6061)	0.1199899650461909
  :	:
  (20340, 38385)	0.18631575677814602
  (20340, 3779)	0.20098490798510654
  (20340, 40750)	0.1871813255088002
  (20340, 4071)	0.1942773426327866
  (20340, 4381)	0.18807837628292803
  (20340, 44963)	0.36196471831788335
  (20340, 15593)	0.1831285972468348
  (20340, 50184)	0.16964745305754977
  (20340, 48979)	0.14837969471652807
  (20340, 14114)	0.1687968109099309
  (20340, 37811)	0.15067587963526774
  (20340, 44940)	0.2673581229321878
  (20340, 6263)	0.1687968109099309
  (20340, 50173)	0.12819510685221785
  (20340, 2433)	0.1081746515441681
  (20340, 31566)	0.0927817077092429
  (20341, 24503)	0.7023494240988789
  (20341, 28523)	0.7118323443536217
  (20342, 8557)	1.0
  (20343, 8557)	1.0
  (20344, 8557)	1.0
  (20345, 8557)	1.0
  (20346, 27426)	0.7281528531108766
  (20346, 23848)	0.6854147813597911
  (20347, 43736)	1.0
```

In [43]:
```
y=df['text_type'].values
y
```

Out[43]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [44]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1
X_train
```

Out[44]:
```
<14243x60269 sparse matrix of type '<class 'numpy.float64'>'
	with 396507 stored elements in Compressed Sparse Row format>
```

MODEL CREATION

In [36]:
```
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import classification_report
sv=SVC()
lr=LogisticRegression()
rf=RandomForestClassifier()
ab= AdaBoostClassifier()
models=[sv,lr,rf,ab]
for model in models:
    print(model)
    model.fit(X_train,y_train)
    y_pred=model.predict(X_test)
    print(classification_report(y_test,y_pred))
```

```
SVC()
              precision    recall  f1-score   support

           0       0.93      0.98      0.95      4274
           1       0.96      0.82      0.88      1831

    accuracy                           0.93      6105
   macro avg       0.94      0.90      0.92      6105
weighted avg       0.94      0.93      0.93      6105

LogisticRegression()
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      4274
           1       0.94      0.76      0.84      1831

    accuracy                           0.92      6105
   macro avg       0.92      0.87      0.89      6105
weighted avg       0.92      0.92      0.91      6105

RandomForestClassifier()
              precision    recall  f1-score   support

           0       0.92      0.99      0.96      4274
           1       0.98      0.80      0.88      1831

    accuracy                           0.94      6105
   macro avg       0.95      0.90      0.92      6105
weighted avg       0.94      0.94      0.93      6105

AdaBoostClassifier()
              precision    recall  f1-score   support

           0       0.88      0.95      0.92      4274
           1       0.86      0.71      0.78      1831

    accuracy                           0.88      6105
   macro avg       0.87      0.83      0.85      6105
weighted avg       0.88      0.88      0.87      6105
```

Now I am trying this with a comment which should be classified as Ham

In [38]:
```
y_new=model.predict(tf.transform(["If he started searching,he will get job in f
if y_new==1:
    print("Spam")
if y_new==0:
    print("Ham")
```

```
Ham
```

BEST MODEL:

The Random Forest Classifier emerged as the best model,achieving a remarkable accuracy score of 96%.