OASIS INFOTECH : DATA SCIENCS

Author: Sujata Gaikwad

In [1]:
```
import pandas as pd
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
from tqdm.auto import tqdm
```

In [2]:
```
data=pd.read_csv(r"C:\Users\HP\Downloads\spam.csv",encoding='latin-1')
data
```

Out[2]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	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 fina...	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 aro...	NaN	NaN	NaN
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

In [3]:
```
data.drop(["Unnamed: 2","Unnamed: 3","Unnamed: 4"],axis=1,inplace=True)
```

In [4]:
```
data.head()
```

Out[4]:

	v1	v2
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...

In [5]:
```
data.tail()
```

Out[5]:

	v1	v2
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

EDA : 1) Handling Null Values

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
v1    0
v2    0
dtype: int64
```

2) Handling Duplicate Values

In [7]:
```
data["v2"].nunique()
```

Out[7]:
```
5169
```

In [8]:
```
data.shape
```

Out[8]:
```
(5572, 2)
```

In [9]:
```
data["v2"].drop_duplicates(inplace=True)
```

In [10]:
```
data.shape
```

Out[10]:
```
(5572, 2)
```

In [11]:
```
data
```

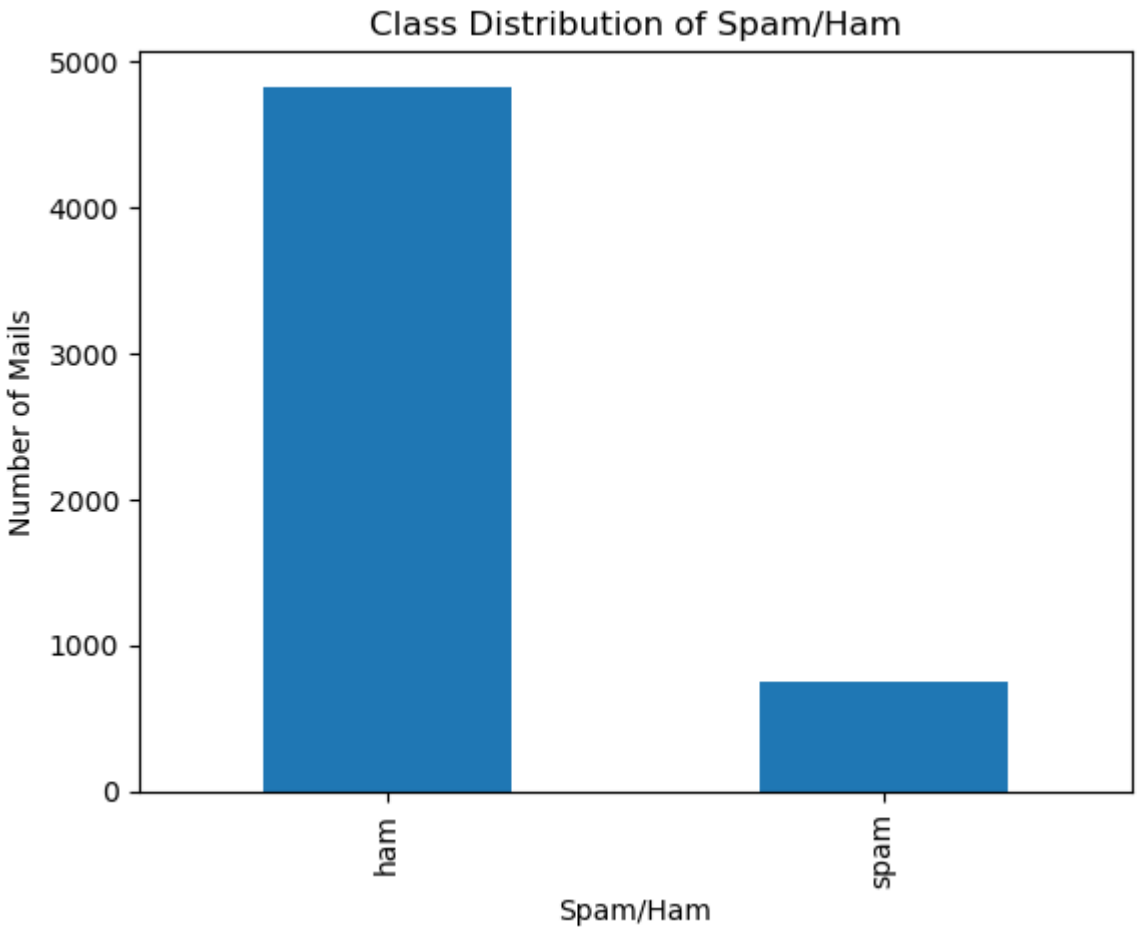
Out[11]:

	v1	v2
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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

3) Class Distributions

In [12]:
```
# Create a bar plot of the class distribution
class_counts=data["v1"].value_counts()
class_counts.plot(kind="bar")
plt.title("Class Distribution of Spam/Ham")
plt.xlabel("Spam/Ham")
plt.ylabel("Number of Mails")
plt.show()
```



Word Count

In [13]:
```
!pip install stopwords
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: stopwords in c:\users\hp\appdata\roaming\pytho
n\python39\site-packages (1.0.0)
```

In [14]:
```
from collections import Counter
import re
```

In [15]:
```
# Concatenate all tweet texts into a sigle string
all_text ="".join(data["v2"].values)
# remove URLs,mentions,and hastags from the text
all_text = re.sub(r'http\S+', '', all_text)
all_text = re.sub(r'@\S+', '', all_text)
all_text = re.sub(r'#\S+', '', all_text)
```

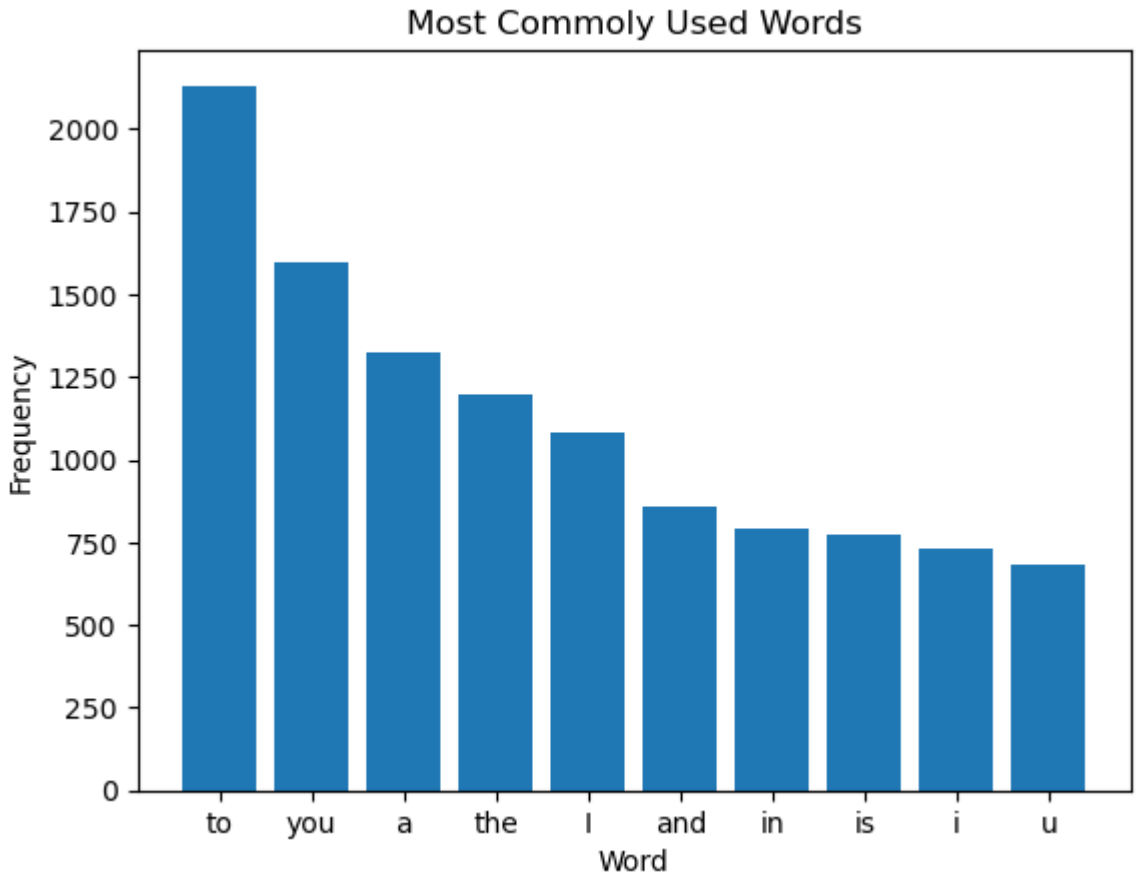
In [16]:
```
# split the text into individual words
words=all_text.split()
```

In [17]:
```
# Count the frequency of each word
word_counts = Counter(words)
top_words = word_counts.most_common(100)
top_words
```

Out[17]:
```
[('to', 2131),
 ('you', 1594),
 ('a', 1327),
 ('the', 1196),
 ('I', 1083),
 ('and', 857),
 ('in', 793),
 ('is', 775),
 ('i', 731),
 ('u', 683),
 ('for', 643),
 ('my', 627),
 ('of', 591),
 ('your', 559),
 ('me', 538),
 ('on', 481),
 ('have', 471),
 ('2', 447),
 ('that', 408),
 ('are', 392),
 ('it', 387),
 ('or', 373),
 ('call', 364),
 ('be', 362),
 ('at', 349),
 ('with', 346),
 ('not', 335),
 ('will', 329),
 ('get', 325),
 ('can', 302),
 ('ur', 293),
 ('so', 292),
 ('but', 280),
 ('&lt;', 269),
 ('from', 255),
 ('4', 247),
 ('do', 238),
 ('U', 236),
 ('just', 236),
 ('if', 233),
 ('go', 233),
 ('when', 232),
 ('up', 223),
 ('this', 221),
 ('we', 220),
 ('like', 219),
 ('know', 218),
 ('.', 216),
 ('all', 213),
 ('got', 203),
 ('was', 199),
 ('out', 195),
 ('come', 195),
 ("I'm", 183),
 ('am', 183),
 ('now', 172),
 ('by', 155),
```

```
('want', 154),
('send', 149),
('about', 148),
('time', 148),
('You', 145),
('?', 144),
('Call', 141),
('going', 141),
('need', 141),
('...', 141),
('then', 138),
('n', 136),
('what', 135),
('still', 134),
('as', 133),
('only', 130),
('one', 129),
('he', 127),
('its', 127),
('our', 125),
('text', 124),
('no', 123),
("I'll", 121),
('been', 119),
('some', 114),
('think', 113),
('has', 113),
('good', 113),
('there', 112),
('r', 112),
('But', 111),
('any', 111),
("don't", 110),
('see', 110),
('love', 109),
('how', 108),
('an', 108),
('back', 107),
('&', 107),
('Ì_', 104),
('tell', 104),
('take', 100),
('home', 99)]
```

In [18]:
```
# Create a bar chart of the most common words
top_words=word_counts.most_common(10)
x_values=[word[0] for word in top_words]
y_values=[word[1] for word in top_words]
plt.bar(x_values,y_values)
plt.xlabel('Word')
plt.ylabel('Frequency')
plt.title('Most Commoly Used Words')
plt.show()
```



Natural Language Processing

1).Data Cleaning

In [19]:
```
# Clean the data
def clean_text(text):
    # remove HTML tags
    text = re.sub("<.*?>","",text)
    # remove non-alphabetic characters and convert to lower case
    text = re.sub('[^a-zA-Z]', ' ', text).lower()
    # Tokenize the text
    words = nltk.word_tokenize(text)
    # Remove stopwords
    words = [w for w in words if w not in stopwords.words('english')]
    # Stem the words
    stemmer = PorterStemmer()
    words = [stemmer.stem(w) for w in words]
    # Join the words back into a string
    text = ' '.join(words)
    return text
```

2) Feature Extraction

In [20]:
```
cv=CountVectorizer(max_features=5000)
x=cv.fit_transform(data["v2"]).toarray()
y=data["v1"]
```

In [21]:
```
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=
```

Classification Model

1) Logistic Regression Model

In [22]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
l1=LogisticRegression()
l1.fit(x_train,y_train)
```

Out[22]:
```
LogisticRegression()
```

2) Prediction

In [23]:
```
y_pred=l1.predict(x_test)
y_pred
```

Out[23]:
```
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype=object)
```

3) Accuracy

In [24]:
```
acc=accuracy_score(y_test,y_pred)*100
print("Accuracy:",acc)
```

```
Accuracy: 98.20627802690582
```

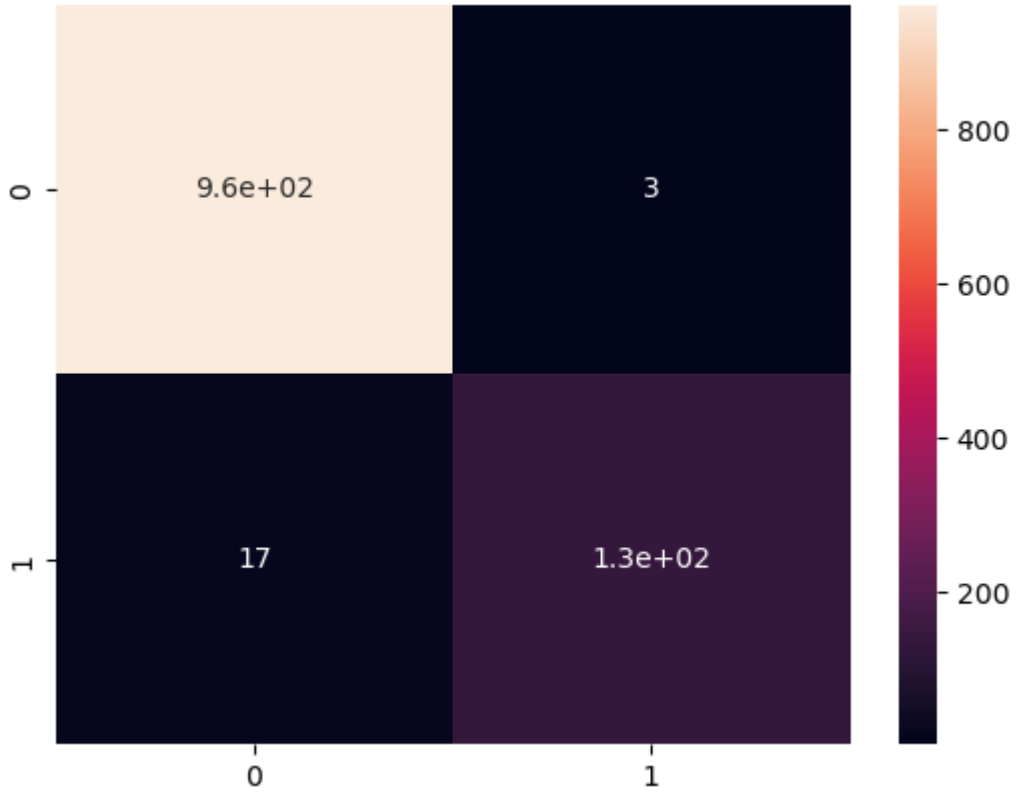
4)Confusion Matrix

In [25]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[962   3]
 [ 17 133]]
```

In [26]:
```
import seaborn as sns
sns.heatmap(cm,annot=True)
```

Out[26]: <AxesSubplot:>



In [27]:
```
from sklearn.metrics import classification_report
report=classification_report(y_test,y_pred)
print(report)
```

```
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99       965
        spam       0.98      0.89      0.93       150

    accuracy                           0.98      1115
   macro avg       0.98      0.94      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```

Thank you

In [ ]:
```

```