```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
data=pd.read_csv('spam.csv')
```

```
data.head()
```

```
   Unnamed: 0 label                                               text  \
0         605   ham  Subject: enron methanol ; meter # : 988291\r\n...   

1        2349   ham  Subject: hpl nom for january 9 , 2001\r\n( see...   

2        3624   ham  Subject: neon retreat\r\nho ho ho , we ' re ar...   

3        4685  spam  Subject: photoshop , windows , office . cheap ...   

4        2030   ham  Subject: re : indian springs\r\nthis deal is t...   


   label_num  
0          0  
1          0  
2          0  
3          1  
4          0  
```

```
print(len(data))
```

```
5171
```

```
s=data['label'].value_counts()
```

```
data['label_num'].value_counts()
```

```
0    3672
1    1499
Name: label_num, dtype: int64
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  5171 non-null   int64 
 1   label       5171 non-null   object
 2   text        5171 non-null   object
 3   label_num   5171 non-null   int64
dtypes: int64(2), object(2)
memory usage: 161.7+ KB
```

```
data.describe
```

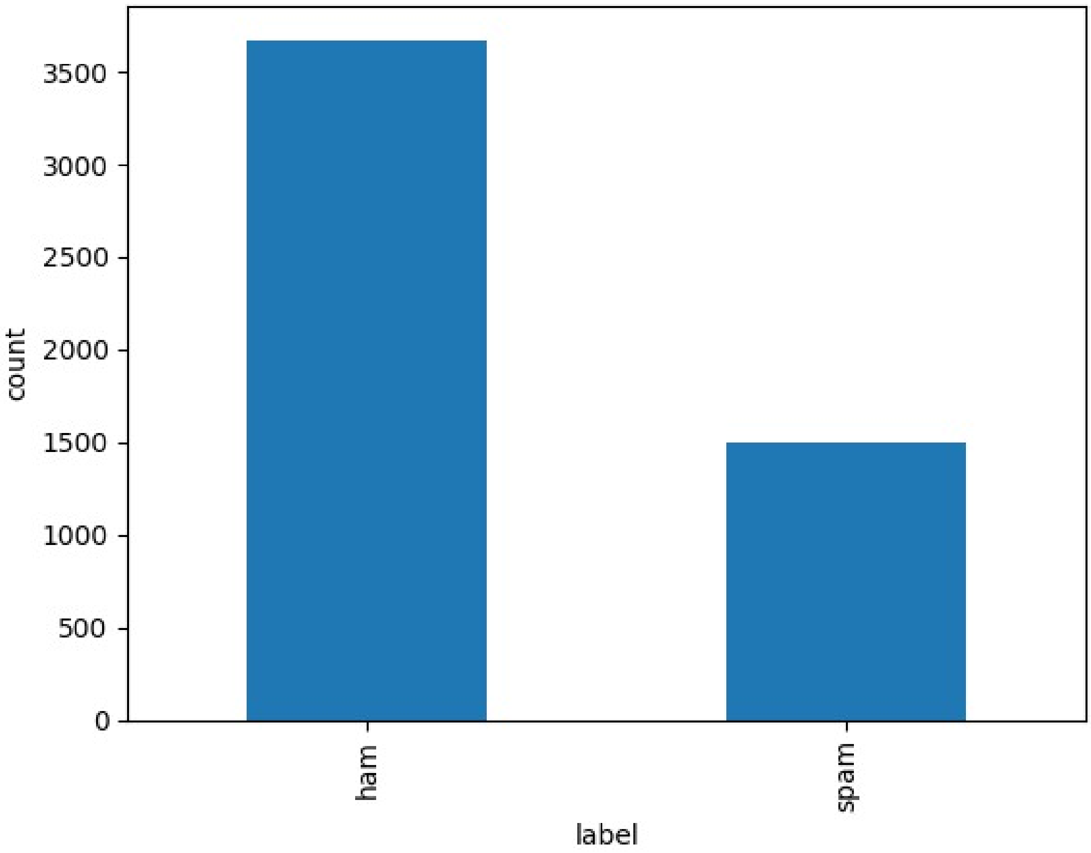
```
<bound method NDFrame.describe of       Unnamed: 0 label
text  \
0            605   ham  Subject: enron methanol ; meter # : 988291\r\
n...
1           2349   ham  Subject: hpl nom for january 9 , 2001\r\
n( see...
2           3624   ham  Subject: neon retreat\r\nho ho ho , we ' re
ar...
3           4685  spam  Subject: photoshop , windows , office .
cheap ...
4           2030   ham  Subject: re : indian springs\r\nthis deal is
t...
...          ...   ...
...
5166        1518   ham  Subject: put the 10 on the ft\r\nthe
transport...
5167         404   ham  Subject: 3 / 4 / 2000 and following noms\r\
nhp...
5168        2933   ham  Subject: calpine daily gas nomination\r\n>\r\
n...
5169        1409   ham  Subject: industrial worksheets for august
2000...
5170        4807  spam  Subject: important online banking alert\r\
ndea...

      label_num
0             0
1             0
2             0
3             1
4             0
...         ...
5166          0
5167          0
5168          0
5169          0
5170          1

[5171 rows x 4 columns]>
```

```
x=plt.xlabel("label")
y=plt.ylabel("count")
s.plot.bar()
```

```
<Axes: xlabel='label', ylabel='count'>
```



```
data=data.drop('Unnamed: 0',axis=1)
data=data.drop('label',axis=1)
```

```
data.head()
```

```
                                               text  label_num
0  Subject: enron methanol ; meter # : 988291\r\n...          0
1  Subject: hpl nom for january 9 , 2001\r\n( see...          0
2  Subject: neon retreat\r\nho ho ho , we ' re ar...          0
3  Subject: photoshop , windows , office . cheap ...          1
4  Subject: re : indian springs\r\nthis deal is t...          0
```

```
x=data['text']
```

```
y=data['label_num']
```

```
print(x.head())
```

```
0    Subject: enron methanol ; meter # : 988291\r\n...
1    Subject: hpl nom for january 9 , 2001\r\n( see...
2    Subject: neon retreat\r\nho ho ho , we ' re ar...
3    Subject: photoshop , windows , office . cheap ...
4    Subject: re : indian springs\r\nthis deal is t...
Name: text, dtype: object
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

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

x=feature_extraction.fit_transform(x)

print(x)
```

```
  (0, 37277)	0.2011097309413472
  (0, 17826)	0.2392098021343849
  (0, 7216)	0.1904342633768622
  (0, 32458)	0.15611810136067314
  (0, 11853)	0.12297449106504706
  (0, 13882)	0.16097016125305302
  (0, 21604)	0.09031870628106299
  (0, 33380)	0.22464796050060742
  (0, 5141)	0.16330305397181255
  (0, 38455)	0.18654788930756974
  (0, 49869)	0.17077874338809632
  (0, 36555)	0.2011097309413472
  (0, 47605)	0.11813966930318913
  (0, 15070)	0.26506114668840103
  (0, 36121)	0.2304064936785402
  (0, 34299)	0.27312912121289434
  (0, 15168)	0.09383338654057037
  (0, 37033)	0.17198604767379225
  (0, 15225)	0.16554063068104793
  (0, 20643)	0.12642883285390433
  (0, 36490)	0.20964545173769655
  (0, 0)	0.10426983213156465
  (0, 31571)	0.14947169013950107
  (0, 21659)	0.20324385893640598
  (0, 33042)	0.14755582683265941
  :	:
  (5170, 40944)	0.043825552681914126
  (5170, 19502)	0.06940047912119972
  (5170, 48113)	0.06743231879609364
  (5170, 5896)	0.07807203338816637
  (5170, 39943)	0.0654536507459819
  (5170, 25582)	0.04084595884272012
  (5170, 25931)	0.060842568812869834
  (5170, 41025)	0.11074677663035766
  (5170, 7528)	0.06905149192063627
  (5170, 268)	0.034142006565318614
  (5170, 21370)	0.05574695522298963
  (5170, 6815)	0.07327442668249698
  (5170, 12642)	0.05049708257577212
  (5170, 29821)	0.045788574177944755
  (5170, 33780) 0.32156994184710264
  (5170, 32467) 0.0520749217466965
  (5170, 30849) 0.13139963528249582
  (5170, 13830) 0.060033187232779735
  (5170, 31235) 0.07559776400175358
  (5170, 25157) 0.06481291025635455
  (5170, 20756) 0.0445653878572119
  (5170, 16946) 0.05332923495537823
  (5170, 44941) 0.04048151839656568
  (5170, 33042) 0.05448307092167786
  (5170, 43337) 0.013260748263266092
```

```
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)
```

```
print(x_train)
```

```
  (0, 42103)    0.14380214994766397
  (0, 41181)    0.14380214994766397
  (0, 34615)    0.14380214994766397
  (0, 31965)    0.15070039628782436
  (0, 48718)    0.15070039628782436
  (0, 45874)    0.15070039628782436
  (0, 2872)     0.15070039628782436
  (0, 1361)     0.12331875735725041
  (0, 3715)     0.15070039628782436
  (0, 27641)    0.1293869283970034
  (0, 13691)    0.15070039628782436
  (0, 25581)    0.15070039628782436
  (0, 18822)    0.15070039628782436
  (0, 44106)    0.14380214994766397
  (0, 22320)    0.15070039628782436
  (0, 32404)    0.13511138736627842
  (0, 12239)    0.14380214994766397
  (0, 17221)    0.13890776627879634
  (0, 8750)     0.13511138736627842
  (0, 33122)    0.13890776627879634
  (0, 42316)    0.13890776627879634
  (0, 3509)     0.13890776627879634
  (0, 13510)    0.13890776627879634
  (0, 17536)    0.13200951993863597
  (0, 15759)    0.13890776627879634
  :   :
  (4135, 43623) 0.046455556362546876
  (4135, 18659) 0.05891424324849027
  (4135, 46691) 0.06143311178984603
  (4135, 1124)  0.05499698670773922
  (4135, 2392)  0.04876028065672649
  (4135, 38525) 0.060946929450399444
  (4135, 33184) 0.08479234025357928
  (4135, 46703)	0.050817292086376836
  (4135, 25582)	0.03568628215580863
  (4135, 41025)	0.04837860134334485
  (4135, 7528)	0.06032888182274984
  (4135, 40773)	0.06143311178984603
  (4135, 29869)	0.04037489345974444
  (4135, 11164)	0.1293125539276832
  (4135, 16063)	0.051428993153682025
  (4135, 12642)	0.0441182723554878
  (4135, 14867)	0.05207480001491583
  (4135, 46683)	0.04281988371384117
  (4135, 24452)	0.036331632097748354
  (4135, 11983)	0.04654678238993937
  (4135, 48017)	0.04816601472532123
  (4135, 36748)	0.09819550100861235
  (4135, 44941)	0.03536787796212732
  (4135, 33042)	0.09520149834712259
  (4135, 43337)	0.023171291238002368
```

```
print(x_test)
```

```
  (0, 19923)	0.09475824995592175
  (0, 20446)	0.2007458914554433
  (0, 29266)	0.1590903367294533
  (0, 34715)	0.2246575631713854
  (0, 12543)	0.0900989359816569
  (0, 30431)	0.0762717644486883
  (0, 24479)	0.0765168601146664
  (0, 45222)	0.07247148825629958
  (0, 34856)	0.0775430734918164
  (0, 46022)	0.13359076841631123
  (0, 34858)	0.08263515842555842
  (0, 26091)	0.06890958341304583
  (0, 34857)	0.06848779820870428
  (0, 42842)	0.07157778826953606
  (0, 29263)	0.07533258808731777
  (0, 15393)	0.04888170976645469
  (0, 34779)	0.07057632475352722
  (0, 23464)	0.2683642441075446
  (0, 22734)	0.05607918019934281
  (0, 30125)	0.05788794887330025
  (0, 40956)	0.09223354424495363
  (0, 34718)	0.07382827623518655
  (0, 47816)	0.0742416664446531
  (0, 33452)	0.05868324167832812
  (0, 40974)	0.3081873521007504
  :	:
  (1033, 2539)	0.12962188365420685
  (1033, 12172)	0.2420459747205963
  (1033, 6265)	0.2328720262376761
  (1033, 2156)	0.1126084161472522
  (1033, 198)   0.19374790249426826
  (1033, 168)   0.20885915939156152
  (1033, 14086) 0.08045495712583124
  (1033, 2795)  0.13335350241238764
  (1033, 27151) 0.4482200770864063
  (1033, 1112)  0.12320442982836503
  (1033, 11517) 0.06675141128367587
  (1033, 1526)  0.2121225559504335
  (1033, 20943) 0.07650311950543334
  (1033, 13093) 0.14516807071045587
  (1033, 49235) 0.18591982788450814
  (1033, 20288) 0.09077846287105402
  (1033, 7500)  0.07721851691029148
  (1033, 24360) 0.14245892398690754
  (1033, 18639) 0.12646908080606306
  (1033, 43337) 0.05589125800063714
  (1034, 12460) 0.6324096984255231
  (1034, 41596) 0.6603893669690137
  (1034, 45132) 0.2886410839898137
  (1034, 36624) 0.2715055614815425
  (1034, 43337) 0.08315595017424587
```

```
print(y_test)
```

```
2713    0
63      0
2678    0
607     0
3990    0
       ..
1358    0
3668    0
4112    1
4747    0
546     1
Name: label_num, Length: 1035, dtype: int64
```

```
from sklearn.linear_model import LogisticRegression
```

```
model=LogisticRegression()
```

```
model.fit(x_train,y_train)
```

```
LogisticRegression(C=1.0 ,class_weight=None ,dual=False,fit_intercept=
True,intercept_scaling=1,
l1_ratio=None ,max_iter=100,multi_class='auto',n_jobs=None ,penalty='1
2',random_state=None, solver='lbfgs' , tol=0.0001, verbose=0,
warm_start=False)
```

```
LogisticRegression(penalty='12')
```

```
y_predict=model.predict(x_test)
```

```
print(y_predict)
```

```
[0 0 0 ... 1 0 1]
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_predict)*100,'%')
```

```
98.93719806763285 %
```

```
# make prediction
input_mail=["Hi Yokesh K,World's biggest coding contest is backAre you
geared up and excited to participatein this event?YesMay be laterAre
you ready to compete against millions of developers and set new
records? Join TechGig's flagship event, Code Gladiators, the biggest
online tech community in India."]
input_mail_feature=feature_extraction.transform(input_mail)
predict=model.predict(input_mail_feature)
print(predict)

if(predict[0]==1):
    print('Ham mail')
else:
    print("spam mail")
```

```
[1]
Ham mail
```