In [2]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [4]:
```
raw_mail_data = pd.read_csv('mail.csv')
raw_mail_data
```

Out[4]:

	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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü 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

In [5]:
```
mail_data = raw_mail_data.where((pd.notnull(raw_mail_data)),'')
```

In [6]:
```
mail_data.head()
```

Out[6]:

	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...

In [7]:
```
mail_data.tail()
```

Out[7]:

	Category	Message
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü 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

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

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

In [9]:
```
mail_data.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
```

In [10]:
```
mail_data.describe()
```

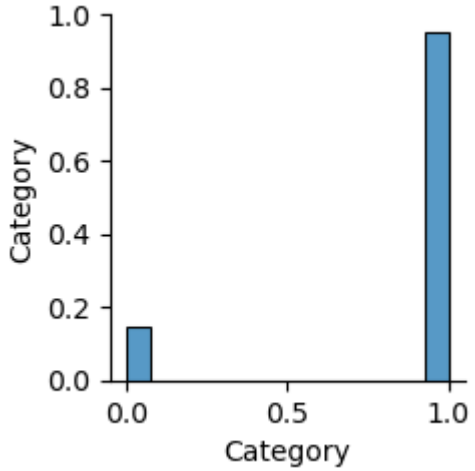
Out[10]:

	Category	Message
count	5572	5572
unique	2	5157
top	ham	Sorry, I'll call later
freq	4825	30

In [11]:
```
mail_data.loc[mail_data['Category'] == 'spam', 'Category',] = 0
mail_data.loc[mail_data['Category'] == 'ham', 'Category',] = 1
```

In [12]:
```
sns.pairplot(mail_data)
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x20ee4c06c50>
```



In [13]:
```
X = mail_data['Message']
X
```

Out[13]:
```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                 Will ü b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                           Rofl. Its true to its name
Name: Message, Length: 5572, dtype: object
```

In [15]:
```
Y = mail_data['Category']
Y
```

Out[15]:
```
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: Category, Length: 5572, dtype: object
```

In [16]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)
```

In [17]:
```
print(X.shape,X_train.shape,X_test.shape)
```

```
(5572,) (4457,) (1115,)
```

In [19]:
```
feature_extraction = TfidfVectorizer(min_df = 1, stop_words='english', lowercase=True)

X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)
Y_train = Y_train.astype('int')
Y_test = Y_test.astype('int')
```

In [20]:
```
X_test_features
```

Out[20]:
```
<1115x7431 sparse matrix of type '<class 'numpy.float64'>'
        with 7687 stored elements in Compressed Sparse Row format>
```

In [21]:
```
Y_train
```

Out[21]:
```
3075    1
1787    1
1614    1
4304    1
3266    0
       ..
789     0
968     1
1667    1
3321    1
1688    0
Name: Category, Length: 4457, dtype: int32
```

In [22]:
```
Y_test
```

Out[22]:
```
2632    0
454     1
983     0
1282    1
4610    1
       ..
4827    1
5291    1
3325    1
3561    1
1136    1
Name: Category, Length: 1115, dtype: int32
```

In [23]:
```
print(X_train_features)
```

```
(0, 5413)     0.6198254967574347
(0, 4456)     0.4168658090846482
(0, 2224)     0.413103377943378
(0, 3811)     0.34780165336891333
(0, 2329)     0.38783870336935383
(1, 4080)     0.18880584110891163
(1, 3185)     0.29694482957694585
(1, 3325)     0.31610586766078863
(1, 2957)     0.3398297002864083
(1, 2746)     0.3398297002864083
(1, 918)      0.22871581159877646
(1, 1839)     0.2784903590561455
(1, 2758)     0.3226407885943799
(1, 2956)     0.33036995955537024
(1, 1991)     0.33036995955537024
(1, 3046)     0.2503712792613518
(1, 3811)     0.17419952275504033
(2, 407)      0.509272536051008
(2, 3156)     0.4107239318312698
(2, 2404)     0.45287711070606745
(2, 6601)     0.6056811524587518
(3, 2870)     0.5864269879324768
(3, 7414)     0.8100020912469564
(4, 50)       0.23633754072626942
(4, 5497)     0.15743785051118356
:     :
(4454, 4602)  0.2669765732445391
(4454, 3142)  0.32014451677763156
(4455, 2247)  0.37052851863170466
(4455, 2469)  0.35441545511837946
(4455, 5646)  0.33545678464631296
(4455, 6810)  0.29731757715898277
(4455, 6091)  0.23103841516927642
(4455, 7113)  0.30536590342067704
(4455, 3872)  0.3108911491788658
(4455, 4715)  0.30714144758811196
(4455, 6916)  0.19636985317119715
(4455, 3922)  0.31287563163368587
(4455, 4456)  0.24920025316220423
(4456, 141)   0.292943737785358
(4456, 647)   0.30133182431707617
```

```
(4456, 6311)  0.30133182431707617
(4456, 5569)  0.4619395404299172
(4456, 6028)  0.21034888000987115
(4456, 7154)  0.24083218452280053
(4456, 7150)  0.3677554681447669
(4456, 6249)  0.17573831794959716
(4456, 6307)  0.2752760476857975
(4456, 334)   0.2220077711654938
(4456, 5778)  0.16243064490100795
(4456, 2870)  0.31523196273113385
```

In [24]:
```
print(X_test_features)
```

```
  (0, 7271)     0.1940327008179069
  (0, 6920)     0.20571591693537986
  (0, 5373)     0.2365698724638063
  (0, 5213)     0.1988547357502182
  (0, 4386)     0.18353336340308998
  (0, 1549)     0.2646498848307188
  (0, 1405)     0.3176863938914351
  (0, 1361)     0.25132445289897426
  (0, 1082)     0.2451068436245027
  (0, 1041)     0.28016206931555726
  (0, 405)      0.2381316303003606
  (0, 306)      0.23975986557206702
  (0, 20)       0.30668032384591537
  (0, 14)       0.26797874471323896
  (0, 9)        0.2852706805264544
  (0, 1)        0.2381316303003606
  (1, 7368)     0.29957800964520975
  (1, 6732)     0.42473488678029325
  (1, 6588)     0.3298937975962767
  (1, 6507)     0.26731535902873493
  (1, 6214)     0.3621564482127515
  (1, 4729)     0.22965776503163893
  (1, 4418)     0.3457696891316818
  (1, 3491)     0.496093956101028
  (2, 7205)     0.22341717215670331
  :     :
(1110, 3167)  0.5718357066163949
(1111, 7353)  0.4991205841293424
(1111, 6787)  0.40050175714278885
(1111, 6033)  0.4714849709283488
(1111, 3227)  0.44384935772735523
(1111, 2440)  0.4137350055985486
(1112, 7071)  0.33558524648843113
(1112, 6777)  0.32853717524096393
(1112, 6297)  0.3056896872268727
(1112, 5778)  0.22807428098549426
(1112, 5695)  0.3381604952481646
(1112, 5056)  0.2559183043595413
(1112, 4170)  0.3307835623173863
(1112, 2329)  0.241856898377491
(1112, 1683)  0.4017087436272034
```

```
  (1112, 1109)  0.35334496762883244
  (1113, 4080)  0.3045947361955407
  (1113, 4038)  0.37023520529413706
  (1113, 3811)  0.28103080586555096
  (1113, 3281)  0.33232508601719535
  (1113, 3113)  0.33840833425155675
  (1113, 2852)  0.5956422931588335
  (1113, 2224)  0.3337959267435311
  (1114, 4557)  0.5196253874825217
  (1114, 4033)  0.8543942045002639
```

In [25]:
```
model = LogisticRegression()
```

In [26]:
```
model.fit(X_train_features, Y_train)
```

Out[26]:
```
LogisticRegression
LogisticRegression()
```

In [27]:
```
Predict_train_data=model.predict(X_train_features)
train_data_accuracy=accuracy_score(Y_train,Predict_train_data)
```

In [28]:
```
print('Accuracy of training data:',train_data_accuracy)
```

```
Accuracy of training data: 0.9670181736594121
```

In [29]:
```
predict_test_data=model.predict(X_test_features)
test_data_accuracy=accuracy_score(Y_test,predict_test_data)
```

In [30]:
```
print('Accuracy of test data:',test_data_accuracy)
```

```
Accuracy of test data: 0.9659192825112107
```

In [40]:
```
input_mail = ["I've been searching for the right words to thank you for this breather. I promise i wont take your help

input_data_features = feature_extraction.transform(input_mail)
prediction = model.predict(input_data_features)
print(prediction)
if (prediction[0]==1):
 print('Ham mail')
else:
 print('Spam mail')
```

```
[1]
Ham mail
```

In [ ]: