Text Classification of News Articles using NLP.

1. Article Id – Article id unique given to the record

2. Article – Text of the header and article

3. Category – Category of the article (tech, business, sport, entertainment, politics)

In [4]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```
df = pd.read_csv("C:/Users/nvsiv/OneDrive/Desktop/Data Science with Python/Assignments data/BBC News.csv")
df
```

Out[5]:

	ArticleId	Text	Category
0	1833	worldcom ex-boss launches defence lawyers defe...	business
1	154	german business confidence slides german busin...	business
2	1101	bbc poll indicates economic gloom citizens in ...	business
3	1976	lifestyle governs mobile choice faster bett...	tech
4	917	enron bosses in $168m payout eighteen former e...	business
...	...	...	...
1485	857	double eviction from big brother model caprice...	entertainment
1486	325	dj double act revamp chart show dj duo jk and ...	entertainment
1487	1590	weak dollar hits reuters revenues at media gro...	business
1488	1587	apple ipod family expands market apple has exp...	tech
1489	538	santy worm makes unwelcome visit thousands of ...	tech

1490 rows × 3 columns

In [6]:
```
len(df)
```

Out[6]:
```
1490
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
ArticleId    0
Text         0
Category     0
dtype: int64
```

In [8]:
```
df.drop('ArticleId', axis = 1)
```

Out[8]:

	Text	Category
0	worldcom ex-boss launches defence lawyers defe...	business
1	german business confidence slides german busin...	business
2	bbc poll indicates economic gloom citizens in ...	business
3	lifestyle governs mobile choice faster bett...	tech
4	enron bosses in $168m payout eighteen former e...	business
...	...	...
1485	double eviction from big brother model caprice...	entertainment
1486	dj double act revamp chart show dj duo jk and ...	entertainment
1487	weak dollar hits reuters revenues at media gro...	business
1488	apple ipod family expands market apple has exp...	tech
1489	santy worm makes unwelcome visit thousands of ...	tech

1490 rows × 2 columns

In [9]:
```
df['Category'].unique()
```

Out[9]:
```
array(['business', 'tech', 'politics', 'sport', 'entertainment'],
      dtype=object)
```

In [10]:
```
df['ArticleId'].describe()
```

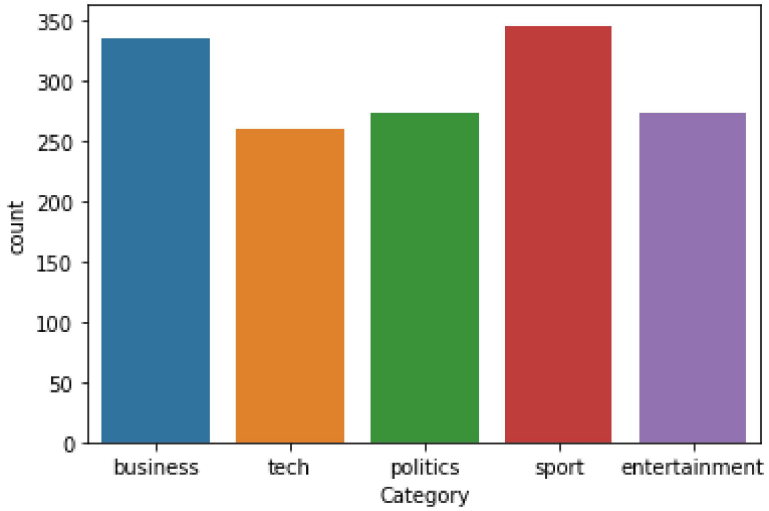
Out[10]:
```
count    1490.000000
mean     1119.696644
std       641.826283
min         2.000000
25%       565.250000
50%      1112.500000
75%      1680.750000
max      2224.000000
Name: ArticleId, dtype: float64
```

In [11]:
```
df['Category'].value_counts(normalize = True)
```

Out[11]:
```
sport            0.232215
business         0.225503
politics         0.183893
entertainment    0.183221
tech             0.175168
Name: Category, dtype: float64
```

In [12]:
```
sns.countplot(df['Category'])
```

Out[12]:
```
<AxesSubplot:xlabel='Category', ylabel='count'>
```



In [13]:
```
from sklearn.model_selection import train_test_split

X = df['Text']
y = df['Category']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [14]:
```
X
```

Out[14]:
```
0       worldcom ex-boss launches defence lawyers defe...
1       german business confidence slides german busin...
2       bbc poll indicates economic gloom citizens in ...
3       lifestyle  governs mobile choice  faster  bett...
4       enron bosses in $168m payout eighteen former e...
                              ...
1485    double eviction from big brother model caprice...
1486    dj double act revamp chart show dj duo jk and ...
1487    weak dollar hits reuters revenues at media gro...
1488    apple ipod family expands market apple has exp...
1489    santy worm makes unwelcome visit thousands of ...
Name: Text, Length: 1490, dtype: object
```

In [15]:
```
y
```

Out[15]:
```
0            business
1            business
2            business
3                tech
4            business
              ...
1485    entertainment
1486    entertainment
1487         business
1488             tech
1489             tech
Name: Category, Length: 1490, dtype: object
```

In [16]:
```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()

X_train_counts = count_vect.fit_transform(X_train)
print(X_train_counts.shape)
print(X_train_counts)
```

```
(998, 20875)
  (0, 14877)    3
  (0, 18978)    13
  (0, 8359)     1
  (0, 1739)     1
  (0, 14083)    4
  (0, 11111)    4
  (0, 13209)    1
  (0, 12584)    16
  (0, 1641)     10
  (0, 4881)     5
  (0, 10265)    3
  (0, 1869)     3
  (0, 2498)     1
  (0, 18589)    1
  (0, 16661)    2
  (0, 16525)    2
  (0, 9721)     12
  (0, 18778)    23
  (0, 19448)    2
  (0, 1383)     2
  (0, 2047)     5
  (0, 60)       1
  (0, 121)      1
  (0, 20757)    2
  (0, 13187)    1
  :     :
  (997, 9973)   1
  (997, 8667)   1
  (997, 18137)  1
  (997, 9691)   1
  (997, 14790)  1
  (997, 1994)   1
  (997, 4177)   2
  (997, 1549)   1
  (997, 10186)  1
  (997, 15432)  1
  (997, 19710)  1
  (997, 9614)   1
  (997, 14616)  1
  (997, 13910)  1
  (997, 18367)  2
  (997, 9094)   2
  (997, 14679)  1
```

```
  (997, 7100)   1
  (997, 14750)  1
  (997, 20776)  1
  (997, 13608)  2
  (997, 18733)  1
  (997, 19758)  1
  (997, 2017)   1
  (997, 18536)  1
```

In [17]:

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()

X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
print(X_train_tfidf.shape)
print(X_train_tfidf)
```

```
(998, 20875)
  (0, 20804)    0.0616912886065074
  (0, 20792)    0.09422625290210854
  (0, 20757)    0.027849536428000365
  (0, 20660)    0.014201980695658225
  (0, 20637)    0.018477191857034246
  (0, 20632)    0.03586039874568629
  (0, 20618)    0.06765562752747772
  (0, 20616)    0.02379862613725528
  (0, 20552)    0.020171310369424333
  (0, 20546)    0.043907755673260135
  (0, 20531)    0.02876455068168828
  (0, 20489)    0.03607157173632206
  (0, 20485)    0.04844000229876237
  (0, 20456)    0.03607157173632206
  (0, 20448)    0.032604005104225606
  (0, 20445)    0.0146791123973463
  (0, 20422)    0.05210770870386959
  (0, 20414)    0.03336877070200023
  (0, 20368)    0.02003095602591858
  (0, 20287)    0.015467965090728918
  (0, 20284)    0.03607157173632206
  (0, 20242)    0.020818402108144864
  (0, 20080)    0.04044018904116368
  (0, 19872)    0.06867201421954602
  (0, 19692)    0.05822372197441094
  :     :
  (997, 2051)   0.09249120169408614
  (997, 2049)   0.13078045416915438
  (997, 2047)   0.04534493373600035
  (997, 2021)   0.0361175892543833
  (997, 2017)   0.06650162403236264
  (997, 1994)   0.058054419089917556
  (997, 1953)   0.023871707865995538
  (997, 1926)   0.04624560084704307
  (997, 1850)   0.050785698934084966
  (997, 1807)   0.04856244680742958
  (997, 1792)   0.0384344352147698
  (997, 1786)   0.038217509595959896
  (997, 1704)   0.03016452253325912
  (997, 1641)   0.09293035755774524
  (997, 1621)   0.013237947381796365
  (997, 1549)   0.06276367629802855
  (997, 1526)   0.08635089337493325
```

```
  (997, 1519)   0.03759577426606561
  (997, 1480)   0.09054858274653231
  (997, 1469)   0.03459064499250703
  (997, 1315)   0.016489479408291294
  (997, 389)    0.038217509595959896
  (997, 356)    0.045274291373266155
  (997, 344)    0.03530122314953035
  (997, 43)     0.043593484723051454
```

In [18]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()

X_train_tfidf = vectorizer.fit_transform(X_train)
print(X_train_tfidf.shape)
print(X_train_tfidf)
```

```
(998, 20875)
  (0, 20284)    0.03607157173632207
  (0, 12811)    0.0154105041471869
  (0, 9371)     0.03506428278321159
  (0, 20448)    0.03260400510422562
  (0, 14828)    0.025668871840032716
  (0, 6094)     0.04792726035848692
  (0, 5052)     0.033348132629451584
  (0, 1505)     0.03882177791227634
  (0, 18631)    0.028584500418998852
  (0, 3858)     0.030037489144874335
  (0, 18628)    0.04711216087747208
  (0, 13264)    0.03797991136217772
  (0, 3840)     0.041582159070502556
  (0, 7324)     0.034692394992770714
  (0, 1544)     0.02867405172511228
  (0, 9552)     0.041999416047404495
  (0, 18787)    0.028537068235696076
  (0, 19376)    0.03246264533059357
  (0, 10766)    0.02639573988091263
  (0, 5060)     0.12338257721301484
  (0, 20080)    0.04044018904116369
  (0, 18609)    0.05097757001368143
  (0, 13664)    0.02340078363314239
  (0, 6990)     0.04290046672014966
  (0, 2599)     0.02742028831836916
  :     :
  (997, 10235)  0.03205881532556317
  (997, 1526)   0.08635089337493324
  (997, 20660)  0.015309370286648721
  (997, 1953)   0.023871707865995535
  (997, 16329)  0.021459309954633472
  (997, 9014)   0.040495403299055446
  (997, 20445)  0.03164741200345572
  (997, 20552)  0.03261623145004439
  (997, 20242)  0.033662553852158265
  (997, 20422)  0.014042692784046821
  (997, 15999)  0.03412899059336392
  (997, 7873)   0.04939356424439194
  (997, 2557)   0.07076174900895837
  (997, 8965)   0.022304617721905493
  (997, 8061)   0.012250953417679336
  (997, 13128)  0.11151642906929428
  (997, 20485)  0.0522170778689048
```

```
  (997, 20757)  0.030021084708143193
  (997, 2047)   0.04534493373600034
  (997, 18778)  0.3410997972567003
  (997, 9721)   0.0744187447173515
  (997, 2498)   0.022906156015038866
  (997, 1641)   0.09293035755774523
  (997, 13209)  0.05090499693587914
  (997, 18978)  0.07412136000992117
```

In [19]:
```
from sklearn.svm import LinearSVC
clf = LinearSVC()
clf.fit(X_train_tfidf,y_train)
```

Out[19]:
```
LinearSVC()
```

In [20]:
```
from sklearn.pipeline import Pipeline

text_clf = Pipeline([('tfidf', TfidfVectorizer()),
                     ('clf', LinearSVC()),
])

text_clf.fit(X_train, y_train)
```

Out[20]:
```
Pipeline(steps=[('tfidf', TfidfVectorizer()), ('clf', LinearSVC())])
```

In [21]:
```
# Form a prediction set
predictions = text_clf.predict(X_test)
```

In [22]:
```
from sklearn import metrics
print(metrics.confusion_matrix(y_test,predictions))
```

```
[[114   0   2   0   1]
 [  1  87   0   0   0]
 [  2   1  88   0   2]
 [  0   0   0 110   0]
 [  1   1   0   0  82]]
```

In [23]:
```
print(metrics.classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
business	0.97	0.97	0.97	117
entertainment	0.98	0.99	0.98	88
politics	0.98	0.95	0.96	93
sport	1.00	1.00	1.00	110
tech	0.96	0.98	0.97	84
accuracy			0.98	492
macro avg	0.98	0.98	0.98	492
weighted avg	0.98	0.98	0.98	492

In [24]:
```
print(metrics.accuracy_score(y_test,predictions))
```

```
0.9776422764227642
```