TITLE : By using multinomial Naive Bayes, RandomForest, Logistic Regression classifiers, perform news classification and analysis to categorize news articles into predefined categories and extract actionable insights. Use AG_news dataset.

NAME : Shinde Shubham Dnyandev,

ROLL NO : 23107121,

BATCH : B.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
df = pd.read_csv("/home/admin1/test.csv")
df
```

Out[3]:

	Class Index	Title	Description
0	3	Fears for T N pension after talks	Unions representing workers at Turner Newall...
1	4	The Race is On: Second Private Team Sets Launc...	SPACE.com - TORONTO, Canada -- A second\team o...
2	4	Ky. Company Wins Grant to Study Peptides (AP)	AP - A company founded by a chemistry research...
3	4	Prediction Unit Helps Forecast Wildfires (AP)	AP - It's barely dawn when Mike Fitzpatrick st...
4	4	Calif. Aims to Limit Farm-Related Smog (AP)	AP - Southern California's smog-fighting agenc...
...	...	...	...
7595	1	Around the world	Ukrainian presidential candidate Viktor Yushch...
7596	2	Void is filled with Clement	With the supply of attractive pitching options...
7597	2	Martinez leaves bitter	Like Roger Clemens did almost exactly eight ye...
7598	3	5 of arthritis patients in Singapore take Bext...	SINGAPORE : Doctors in the United States have ...
7599	3	EBay gets into rentals	EBay plans to buy the apartment and home renta...

7600 rows × 3 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7600 entries, 0 to 7599
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Class Index  7600 non-null   int64 
 1   Title        7600 non-null   object
 2   Description  7600 non-null   object
dtypes: int64(1), object(2)
memory usage: 178.2+ KB
```

In [7]:
```
df.describe()
```

Out[7]:

	Class Index
count	7600.000000
mean	2.500000
std	1.118108
min	1.000000
25%	1.750000
50%	2.500000
75%	3.250000
max	4.000000

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Class Index    0
Title          0
Description    0
dtype: int64
```

In [11]:
```
df.dtypes
```

Out[11]:
```
Class Index     int64
Title          object
Description    object
dtype: object
```

In [13]:
```
X = df["Description"]
Y = df["Class Index"]
```

In [15]:
```
X
```

Out[15]:
```
0       Unions representing workers at Turner   Newall...
1       SPACE.com - TORONTO, Canada -- A second\team o...
2       AP - A company founded by a chemistry research...
3       AP - It's barely dawn when Mike Fitzpatrick st...
4       AP - Southern California's smog-fighting agenc...
                              ...
7595    Ukrainian presidential candidate Viktor Yushch...
7596    With the supply of attractive pitching options...
7597    Like Roger Clemens did almost exactly eight ye...
7598    SINGAPORE : Doctors in the United States have ...
7599    EBay plans to buy the apartment and home renta...
Name: Description, Length: 7600, dtype: object
```

In [17]:
```
Y
```

Out[17]:
```
0       3
1       4
2       4
3       4
4       4
       ..
7595    1
7596    2
7597    2
7598    3
7599    3
Name: Class Index, Length: 7600, dtype: int64
```

In [19]:
```
Y.value_counts()
```

Out[19]:
```
Class Index
3    1900
4    1900
2    1900
1    1900
Name: count, dtype: int64
```

In [21]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()

X_vec = vectorizer.fit_transform(X)
```

In [23]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X_vec, Y, train_size=0.8)
```

MULTINOMIAL NAIVE BAYES :

In [26]:
```
from sklearn.naive_bayes import MultinomialNB
NB = MultinomialNB()

NB.fit(x_train, y_train)
```

Out[26]:
```
▾ MultinomialNB

MultinomialNB()
```

In [28]:
```
y_pred_NB = NB.predict(x_test)
```

In [30]:
```
from sklearn.metrics import accuracy_score, classification_report
NB_ACC = accuracy_score(y_test, y_pred_NB)
NB_CR = classification_report(y_test, y_pred_NB)

print("Accuracy :", NB_ACC)
print("Classification Report :\n", NB_CR)
```

```
Accuracy : 0.8690789473684211
Classification Report :
               precision    recall  f1-score   support

           1       0.89      0.86      0.88       392
           2       0.90      0.97      0.93       364
           3       0.83      0.85      0.84       391
           4       0.84      0.80      0.82       373

    accuracy                           0.87      1520
   macro avg       0.87      0.87      0.87      1520
weighted avg       0.87      0.87      0.87      1520
```

RANDOM FORECT CLASSIFIER :

In [33]:
```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier()

RF.fit(x_train, y_train)
```

Out[33]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [35]:
```
y_pred_RF = RF.predict(x_test)
```

In [37]:
```
from sklearn.metrics import accuracy_score, classification_report
RF_ACC = accuracy_score(y_test, y_pred_RF)
RF_CR = classification_report(y_test, y_pred_RF)

print("Accuracy :", RF_ACC)
print("Classification Report :\n", RF_CR)
```

```
Accuracy : 0.8098684210526316
Classification Report :
               precision    recall  f1-score   support

           1       0.85      0.81      0.83       392
           2       0.84      0.92      0.88       364
           3       0.85      0.73      0.78       391
           4       0.72      0.79      0.75       373

    accuracy                           0.81      1520
   macro avg       0.81      0.81      0.81      1520
weighted avg       0.81      0.81      0.81      1520
```

LOGISTIC REGRESSION CLASSIFIER :

In [40]:
```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()

LR.fit(x_train, y_train)
```

Out[40]:
```
▾ LogisticRegression
LogisticRegression()
```

In [42]:
```
y_pred_LR = LR.predict(x_test)
```

In [44]:
```
from sklearn.metrics import accuracy_score, classification_report
LR_ACC = accuracy_score(y_test, y_pred_LR)
LR_CR = classification_report(y_test, y_pred_LR)

print("Accuracy :", LR_ACC)
print("Classification Report :\n", LR_CR)
```

```
Accuracy : 0.8493421052631579
Classification Report :
               precision    recall  f1-score   support

           1       0.89      0.85      0.87       392
           2       0.90      0.94      0.92       364
           3       0.85      0.79      0.82       391
           4       0.77      0.83      0.80       373

    accuracy                           0.85      1520
   macro avg       0.85      0.85      0.85      1520
weighted avg       0.85      0.85      0.85      1520
```