Mount Google Drive

```
from google.colab import drive
drive.mount('/content/gdrive/')
```

```
Drive already mounted at /content/gdrive/; to attempt to forcibly remount, call drive.mount("/content/
```

```
%cd /content/gdrive/MyDrive/D341_19CSE453_NLP/lab9
```

```
/content/gdrive/MyDrive/D341_19CSE453_NLP/lab9
```

Load the Dataset

```
import pandas as pd
df = pd.read_csv('bbc_sports.csv')
df = df.drop('Unnamed: 0',axis=1)
df.head()
```

	data	labels
0	England victory tainted by history\n\nAs Engla...	1
1	Australia complete sweep\n\nThird Test, Sydney...	1
2	UK Athletics agrees new kit deal\n\nUK Athleti...	0
3	Bekele sets sights on world mark\n\nOlympic 10...	0
4	Captains lining up for Aid match\n\nIreland's ...	3

Target class names

```
classes =['athletics', 'cricket', 'football', 'rugby', 'tennis']
```

Preprocess Data

```
%run preprocess.ipynb
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Vectorise Data

```
df['clean_text'] = df['data'].apply(preprocess)
df.head(10)
```

	data	labels	clean_text
0	England victory tainted by history\n\nAs Engla...	1	england victori taint histori england attempt ...
1	Australia complete sweep\n\nThird Test, Sydney...	1	australia complet sweep third test sydney day ...
2	UK Athletics agrees new kit deal\n\nUK Athleti...	0	uk athlet agre new kit deal uk athlet agre new...
3	Bekele sets sights on world mark\n\nOlympic 10...	0	bekel set sight world mark olymp champion kene...
4	Captains lining up for Aid match\n\nIreland's ...	3	captain line aid match ireland brian one four ...
5	Cantona issues Man Utd job hint\n\nFormer Manc...	2	cantona issu man utd job hint former manchest ...

```
from sklearn.feature_extraction.text import TfidfVectorizer

texts = df['clean_text'].values

tfidf_vectorizer = TfidfVectorizer()

X = tfidf_vectorizer.fit_transform(texts)

y = df['labels'].values

print(X.shape)
print(y.shape)
```

```
(737, 8900)
(737,)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3, shuffle = True, stratify = y,rando
```

ML Alog 1

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

```
knn.score(X_test,y_test)
```

```
0.9819819819819819
```

```
predicted1 = knn.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
knnscore = accuracy_score(y_test, predicted1)
```

```
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test, predicted1))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        31
           1       0.97      1.00      0.99        37
           2       0.98      0.99      0.98        80
           3       1.00      0.93      0.96        44
           4       1.00      1.00      1.00        30

    accuracy                           0.98       222
   macro avg       0.98      0.98      0.98       222
weighted avg       0.98      0.98      0.98       222
```

ML Algo 2

```
from sklearn.naive_bayes import MultinomialNB
```

```
mnb = MultinomialNB()
mnb.fit(X_train, y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
predicted2 = mnb.predict(X_test)
```

```
mnb.score(X_test,y_test)
```

```
0.8468468468468469
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_true=y_test, y_pred=predicted2)
```

```
array([[22,  0,  9,  0,  0],
       [ 0, 34,  3,  0,  0],
       [ 0,  0, 80,  0,  0],
       [ 0,  0,  6, 38,  0],
       [ 0,  0, 16,  0, 14]])
```

```
print(classification_report(y_test, predicted2))
```

```
              precision    recall  f1-score   support

           0       1.00      0.71      0.83        31
           1       1.00      0.92      0.96        37
           2       0.70      1.00      0.82        80
           3       1.00      0.86      0.93        44
           4       1.00      0.47      0.64        30

    accuracy                           0.85       222
   macro avg       0.94      0.79      0.84       222
weighted avg       0.89      0.85      0.84       222
```

```
from sklearn.metrics import accuracy_score
nbsscore = accuracy_score(y_test, predicted2)
```

```
nbsscore
```

```
0.8468468468468469
```

Test with Sample News

```
new_news = 'after 25 years team india in cricket final lift the world cup, today every indian dream succeded
new_news = preprocess(new_news)
vec = tfidf_vectorizer.transform([new_news])
```

```
y_predict = knn.predict(vec)
```

```
classes[y_predict[0]]
```

```
'cricket'
```

```
y_predict = mnb.predict(vec)
```

```
classes[y_predict[0]]
```

```
'cricket'
```

Compare best classifier

```
from sklearn.model_selection import cross_val_score
knncs = cross_val_score(knn,X, y, scoring='accuracy')
nbscs = cross_val_score(mnb,X, y, scoring='accuracy')
```

```
outcomeacc = [knncs,nbscs]
```

```
outcomeacc
```

```
[array([0.97972973, 0.99324324, 0.97959184, 0.97278912, 0.98639456]),
 array([0.86486486, 0.86486486, 0.83673469, 0.89795918, 0.86394558])]
```

```
model_names = ['KNN','NaiveBayes']
```

```
import matplotlib.pyplot as plt
fig = plt.figure()
fig.suptitle('Machine Learning Model Comparison')
ax = fig.add_subplot(111)
plt.boxplot(outcomeacc)
ax.set_xticklabels(model_names)
plt.show()
```