Type *Markdown* and LaTeX: α^2

In [1]:

```
import numpy as np
import pandas as pd
from pandas import read_csv
import seaborn as sns
import matplotlib.pyplot as plt
import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split

## Case 1 : Would use Word2Vec and then apply different classifiers
from gensim.models import Word2Vec

## Case 2 : Would use TfidfVectorizer and then apply different classifiers
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
import warnings
warnings.filterwarnings("ignore")
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Mahi\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Mahi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Mahi\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [2]:

```
filename = 'BBC_News.csv'
dataset = read_csv(filename)
print(dataset)
```

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

          Category
0         business
1         business
2         business
3             tech
4         business
...            ...
1485  entertainment
1486  entertainment
1487       business
1488           tech
1489           tech

[1490 rows x 3 columns]
```

In [3]:

```
print("Shape of Dataset: ", dataset.shape)
```

```
Shape of Dataset:  (1490, 3)
```

In [4]:

```
print("Columns of Dataset: ", dataset.columns)
```

```
Columns of Dataset:  Index(['ArticleId', 'Text', 'Category'], dtype='objec
t')
```

In [5]:

```
print("Categories of Dataset: ", dataset.Category.unique())
```

```
Categories of Dataset:  ['business' 'tech' 'politics' 'sport' 'entertainme
nt']
```

In [6]:

```
print("Samples of Dataset: ", dataset.sample(n=5))
```

```
Samples of Dataset:        ArticleId
Text  Category
759         890  mobiles rack up 20 years of use mobile phones ...      te
ch
926        1212  qatar and shell in $6bn gas deal shell has sig...  busine
ss
285          78  hodgson relishes european clashes former black...     spo
rt
1431        788  lse  sets date for takeover deal  the london s...  busine
ss
1204        923  budget aston takes on porsche british car make...  busine
ss
```

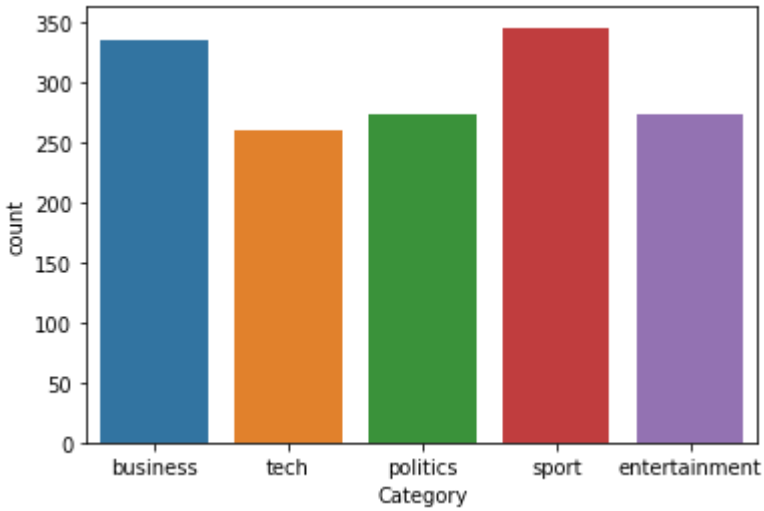
In [7]:

```
# Plotting number of samples within each category
print('NUMBER OF SAMPLES IN EACH CATEGORY: \n')
sns.countplot(dataset.Category)
```

```
NUMBER OF SAMPLES IN EACH CATEGORY:
```

Out[7]:

```
<AxesSubplot:xlabel='Category', ylabel='count'>
```



In [8]:

```
# checking for any null() values
dataset.isna().sum()
```

Out[8]:

```
ArticleId    0
Text         0
Category     0
dtype: int64
```

In [9]:

```
dataset.columns = dataset.columns.str.lower()
print("Columns of Dataset: ", dataset.columns)
```

```
Columns of Dataset:  Index(['articleid', 'text', 'category'], dtype='objec
t')
```

Note : Data set seems balanced.

In [10]:

```
# DATA CLEANING
print('Data cleaning in progress...')

# Tokenize : dividing Sentences into words
dataset['text_clean'] = dataset['text'].apply(nltk.word_tokenize)
print('Tokenization complete.')


# Remove stop words
stop_words=set(nltk.corpus.stopwords.words("english"))
dataset['text_clean'] = dataset['text_clean'].apply(lambda x: [item for item in x if item
print('Stop words removed.')


# Remove numbers, punctuation and special characters (only keep words)
regex = '[a-z]+'
dataset['text_clean'] = dataset['text_clean'].apply(lambda x: [item for item in x if re.m
print('Numbers, punctuation and special characters removed.')


# Lemmatization : lemma means base form of a word.  // Example : leaf and leaves get lemm
lem = nltk.stem.wordnet.WordNetLemmatizer()
dataset['text_clean'] = dataset['text_clean'].apply(lambda x: [lem.lemmatize(item, pos='v
print('Lemmatization complete.\nData cleaning complete.\n')
```

```
Data cleaning in progress...
Tokenization complete.
Stop words removed.
Numbers, punctuation and special characters removed.
Lemmatization complete.
Data cleaning complete.
```

Solution : Using sklearn.feature_extraction.text.TfidfVectorizer

In [ ]:

```
# Classification using TFIDF vectorizer

# Vectorize training and testing data. Here we would pass TfidfVectorizer() to vec
def Vectorize(vec, X_train, X_test):

    X_train_vec = vec.fit_transform(X_train)
    X_test_vec = vec.transform(X_test)

    print('Vectorization complete.\n')

    return X_train_vec, X_test_vec


# Use multiple classifiers and grid search for prediction
def ML_modeling(models, params, X_train, X_test, y_train, y_test):

    if not set(models.keys()).issubset(set(params.keys())):
        raise ValueError('Some estimators are missing parameters')

    for key in models.keys():

        model = models[key]
        param = params[key]
        gs = GridSearchCV(model, param, cv=5, error_score=0, refit=True)
        gs.fit(X_train, y_train)
        y_pred = gs.predict(X_test)

        # Print scores for the classifier
        print(key, ':', gs.best_params_)
        print("Accuracy: %1.3f \tPrecision: %1.3f \tRecall: %1.3f \t\tF1: %1.3f\n" % (acc

    return


## Preparing to make a pipeline
models = {
    'Naive Bayes': MultinomialNB(),
    'Decision Tree': DecisionTreeClassifier(),
    'Perceptron': MLPClassifier()
}

params = {
    'Naive Bayes': { 'alpha': [0.5, 1], 'fit_prior': [True, False] },
    'Decision Tree': { 'min_samples_split': [1, 2, 5] },
    'Perceptron': { 'alpha': [0.0001, 0.001], 'activation': ['tanh', 'relu'] }
    }


# Encode label categories to numbers
enc = LabelEncoder()
dataset['category'] = enc.fit_transform(dataset['category'])
labels = list(enc.classes_)

# Train-test split and vectorize
X_train, X_test, y_train, y_test = train_test_split(dataset['text'], dataset['category'],
X_train_vec, X_test_vec = Vectorize(TfidfVectorizer(), X_train, X_test)

ML_modeling(models, params, X_train_vec, X_test_vec, y_train, y_test)
```

```
## ML modeling method also prints performance scores for each classifier
```

```
Vectorization complete.

Naive Bayes : {'alpha': 0.5, 'fit_prior': False}
Accuracy: 0.960         Precision: 0.960        Recall: 0.960           F
1: 0.959

Decision Tree : {'min_samples_split': 5}
Accuracy: 0.718         Precision: 0.728        Recall: 0.715           F
1: 0.720
```

Conclusion of this Ananlysis : TfidfVectorizer seems to have performed far better with good results.