1. Text Classification of News Articles using NLP.

Article Id – Article id unique given to the record
Article – Text of the header and article
Category – Category of the article (tech, business, sport, entertainment, politics)

Consider BBC News as corpus for implementing question 1

Text Classification of News Articles

Know about Data

For the task of news classification with machine learning, I have collected a dataset from Kaggle, which contains news articles including their headlines and categories.

Data Fields

- **Article Id** – Article id unique given to the record
- **Article** – Text of the header and article
- **Category** – Category of the article (tech, business, sport, entertainment, politics)

Data Cleaning and Data Preprocessing

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

Import Libraries

let's import the necessary Python libraries and the dataset that we need for this task.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
nltk.download('punkt')
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import make_scorer, roc_curve, roc_auc_score
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
```

Import Dataset

Import the dataset which we will have to use.

Shape of Dataset

Check the shape (row and column) of the dataset.

```
dataset.shape
```

```
(1490, 3)
```

Check Information of Columns of Dataset

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1490 entries, 0 to 1489
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   ArticleId  1490 non-null   int64 
 1   Text       1490 non-null   object
 2   Category   1490 non-null   object
dtypes: int64(1), object(2)
memory usage: 35.0+ KB
```

Count Values of Categories

There are five news categories i.e. Sports, Business, Politics, Entertainment, Tech.

```
dataset['Category'].value_counts()
```

```
sport            346
business         336
politics         274
entertainment    273
tech             261
Name: Category, dtype: int64
```

Convert Categories Name into Numerical Index

Convert the given news categories into categorical values.

```
# Associate Category names with numerical index and save it in new column
CategoryId
target_category = dataset['Category'].unique()
print(target_category)
```

```
['business' 'tech' 'politics' 'sport' 'entertainment']
```

```
dataset['CategoryId'] = dataset['Category'].factorize()[0]
dataset.head()
```

	ArticleId	Text	Category	CategoryId
0	1833	worldcom ex-boss launches defence lawyers defe...	business	0
1	154	german business confidence slides german busin...	business	0
2	1101	bbc poll indicates economic gloom citizens in ...	business	0
3	1976	lifestyle governs mobile choice faster bett...	tech	1
4	917	enron bosses in $168m payout eighteen former e...	business	0

Show Category's Name w.r.t Category ID

Here you can show that news category's name with respect to the following unique category ID.

```
# Create a new pandas dataframe "category", which only has unique Categories,
also sorting this list in order of CategoryId values
category = dataset[['Category',
'CategoryId']].drop_duplicates().sort_values('CategoryId')
category
```

	Category	CategoryId
0	business	0
3	tech	1
5	politics	2
6	sport	3
7	entertainment	4

Exploratory Data Analysis (EDA)

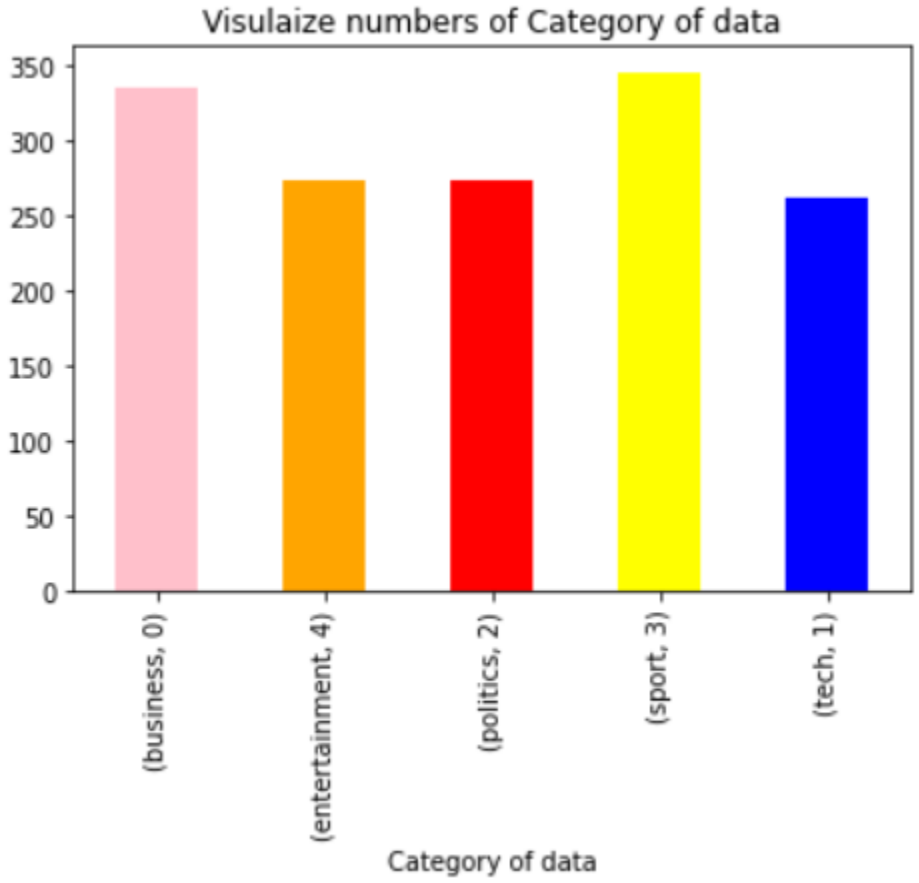
In data mining, Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. EDA is used for seeing what the data can tell us before the modeling task. It is not easy to look at a column of

numbers or a whole spreadsheet and determine important characteristics of the data. It may be tedious, boring, and/or overwhelming to derive insights by looking at plain numbers. Exploratory data analysis techniques have been devised as an aid in this situation.

Visualizing Data

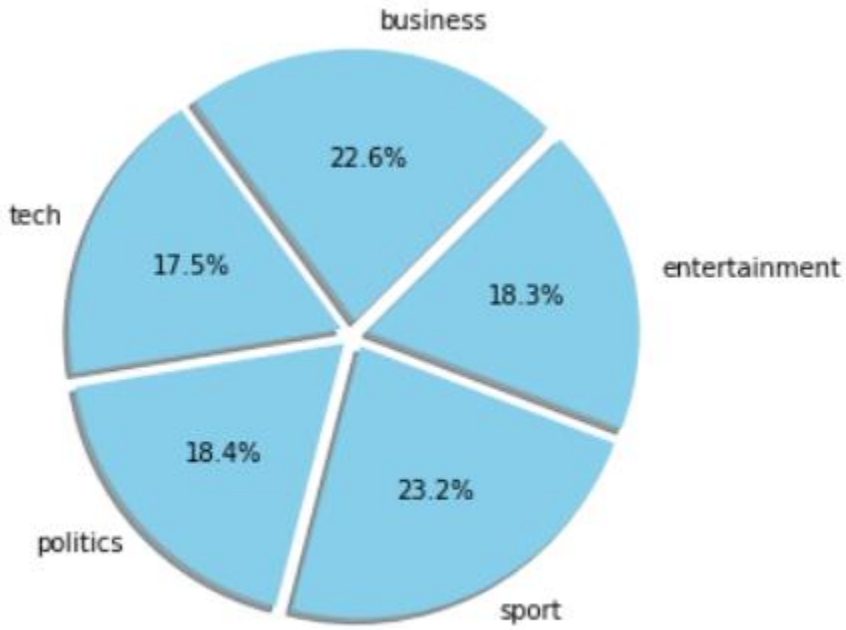
The below graph shows the news article count for category from our dataset.

```
dataset.groupby('Category').CategoryId.value_counts().plot(kind = "bar", color =
["pink", "orange", "red", "yellow", "blue"])
plt.xlabel("Category of data")
plt.title("Visulaize numbers of Category of data")
plt.show()
```




```
fig = plt.figure(figsize = (5,5))
colors = ["skyblue"]
business = dataset[dataset['CategoryId'] == 0 ]
tech = dataset[dataset['CategoryId'] == 1 ]
```

```
politics = dataset[dataset['CategoryId'] == 2]
sport = dataset[dataset['CategoryId'] == 3]
entertainment = dataset[dataset['CategoryId'] == 4]
count = [business['CategoryId'].count(), tech['CategoryId'].count(),
politics['CategoryId'].count(), sport['CategoryId'].count(),
entertainment['CategoryId'].count()]
pie = plt.pie(count, labels = ['business', 'tech', 'politics', 'sport',
'entertainment'],
              autopct = "%1.1f%%",
              shadow = True,
              colors = colors,
              startangle = 45,
              explode = (0.05, 0.05, 0.05, 0.05,0.05))
```



Visualizing Category Related Words

Here we use the word cloud module to show the category-related words.

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analyzing data from social network websites.

```
from wordcloud import WordCloud

stop = set(stopwords.words('english'))
```

```
business = dataset[dataset['CategoryId'] == 0]

business = business['Text']

tech = dataset[dataset['CategoryId'] == 1]

tech = tech['Text']

politics = dataset[dataset['CategoryId'] == 2]

politics = politics['Text']

sport = dataset[dataset['CategoryId'] == 3]

sport = sport['Text']

entertainment = dataset[dataset['CategoryId'] == 4]

entertainment = entertainment['Text']

def wordcloud_draw(dataset, color = 'white'):

words = ' '.join(dataset)

cleaned_word = ' '.join([word for word in words.split()

if (word != 'news' and word != 'text')])

wordcloud = WordCloud(stopwords = stop,

background_color = color,

width = 2500, height = 2500).generate(cleaned_word)

plt.figure(1, figsize = (10,7))

plt.imshow(wordcloud)

plt.axis("off")

plt.show()

print("business related words:")

wordcloud_draw(business, 'white')
```

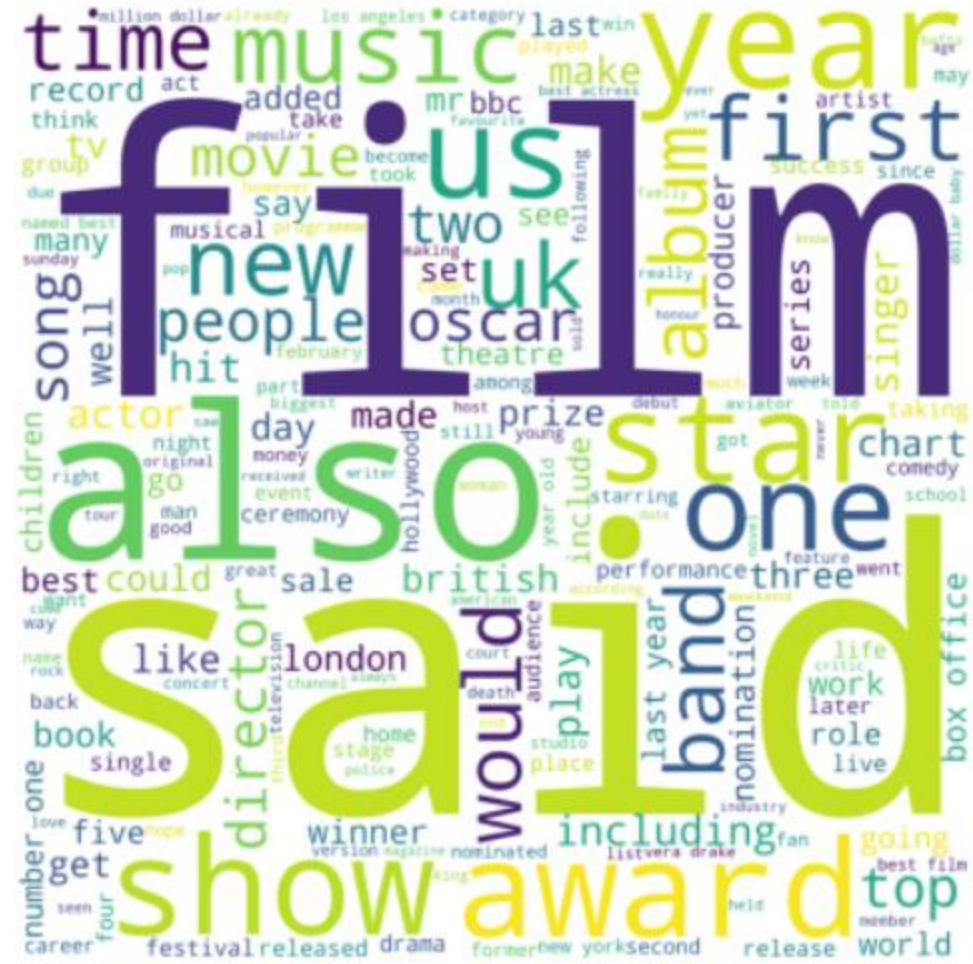
```
print("tech related words:")

wordcloud_draw(tech, 'white')

print("politics related words:")

wordcloud_draw(politics, 'white')

print("sport related words:")

wordcloud_draw(sport, 'white')

print("entertainment related words:")

wordcloud_draw(entertainment, 'white')
```

tech related words:

politics related words:

sport related words:

entertainment related words:



Show Text Column of Dataset

```
text = dataset["Text"]
text.head(10)
```

```
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...
5    howard  truanted to play snooker  conservative...
6    wales silent on grand slam talk rhys williams ...
7    french honour for director parker british film...
8    car giant hit by mercedes slump a slump in pro...
9    fockers fuel festive film chart comedy meet th...
Name: Text, dtype: object
```

Show Category Column of Dataset

```
category = dataset['Category']
category.head(10)
```

```
0          business
1          business
2          business
3              tech
4          business
5          politics
6             sport
7     entertainment
8          business
9     entertainment
Name: Category, dtype: object
```

Remove All Tags

First, we remove all tags which are present in our given dataset.

```
def remove_tags(text):
  remove = re.compile(r'')
  return re.sub(remove, '', text)
dataset['Text'] = dataset['Text'].apply(remove_tags)
```

Remove Special Characters

Here we remove all the special characters.

```
def special_char(text):
  reviews = ''
  for x in text:
```

```
    if x.isalnum():
      reviews = reviews + x
    else:
      reviews = reviews + ' '
  return reviews
dataset['Text'] = dataset['Text'].apply(special_char)
```

Convert Everything in Lower Case

We convert all articles or text to lower case.

It is one of the simplest and most effective forms of text preprocessing. It is applicable to most text mining and NLP problems and can help in cases where your dataset is not very large and significantly helps with the consistency of expected output.

```
def convert_lower(text):
   return text.lower()
dataset['Text'] = dataset['Text'].apply(convert_lower)
dataset['Text'][1]
```

Remove all Stopwords

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.
We would not want these words to take up space in our database, or take up the valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

```
def remove_stopwords(text):
  stop_words = set(stopwords.words('english'))
  words = word_tokenize(text)
  return [x for x in words if x not in stop_words]
dataset['Text'] = dataset['Text'].apply(remove_stopwords)
dataset['Text'][1]
```

Lemmatizing the Words

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meanings to one word. lemmatization is preferred over Stemming because lemmatization does morphological analysis of the words.

```
def lemmatize_word(text):
  wordnet = WordNetLemmatizer()
  return " ".join([wordnet.lemmatize(word) for word in text])
dataset['Text'] = dataset['Text'].apply(lemmatize_word)
dataset['Text'][1]
```

After Cleaning Text our Dataset

```
dataset
```

	ArticleId	Text	Category	CategoryId
0	1833	worldcom ex bos launch defence lawyer defendin...	business	0
1	154	german business confidence slide german busine...	business	0
2	1101	bbc poll indicates economic gloom citizen majo...	business	0
3	1976	lifestyle governs mobile choice faster better ...	tech	1
4	917	enron boss 168m payout eighteen former enron d...	business	0
...	...	...	...	...
1485	857	double eviction big brother model caprice holb...	entertainment	4
1486	325	dj double act revamp chart show dj duo jk joel...	entertainment	4
1487	1590	weak dollar hit reuters revenue medium group r...	business	0
1488	1587	apple ipod family expands market apple expande...	tech	1
1489	538	santy worm make unwelcome visit thousand websi...	tech	1

1490 rows × 4 columns

Declared Dependent and Independent Value

```
x = dataset['Text']
y = dataset['CategoryId']
```

Create and Fit Bag of Words Model

In this step, we construct a vector, which would tell us whether a word in each sentence is a frequent word or not. If a word in a sentence is a frequent word, we set it as 1, else we set it as 0.

Whenever we apply any algorithm in NLP, it works on numbers. We cannot directly feed our text into that algorithm. Hence, the Bag of Words model is used to preprocess the text by converting it into a *bag of words*, which keeps a count of the total occurrences of the most frequently used words.

```
from sklearn.feature_extraction.text import CountVectorizer
x = np.array(dataset.iloc[:,0].values)
y = np.array(dataset.CategoryId.values)
cv = CountVectorizer(max_features = 5000)
x = cv.fit_transform(dataset.Text).toarray()
print("X.shape = ",x.shape)
print("y.shape = ",y.shape)
```

```
Test Accuracy Score of Basic Logistic Regression: % 97.09
Precision : 0.970917225950783
Recall    : 0.970917225950783
F1-score   : 0.9709172259507831

X.shape =  (1490, 5000)
y.shape =  (1490,)
```

Train Test and Split the Dataset

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,
random_state = 0, shuffle = True)
print(len(x_train))
print(len(x_test))
```

```
1043
447
```

Create Empty List

```
#create list of model and accuracy dicts
perform_list = [ ]
```

Create, Fit and Predict all ML Model

```
def run_model(model_name, est_c, est_pnlty):

mdl=''

if model_name == 'Logistic Regression':

mdl = LogisticRegression()

elif model_name == 'Random Forest':

mdl = RandomForestClassifier(n_estimators=100 ,criterion='entropy' ,
random_state=0)
```

```
elif model_name == 'Multinomial Naive Bayes':

mdl = MultinomialNB(alpha=1.0,fit_prior=True)

elif model_name == 'Support Vector Classifer':

mdl = SVC()

elif model_name == 'Decision Tree Classifier':

mdl = DecisionTreeClassifier()

elif model_name == 'K Nearest Neighbour':

mdl = KNeighborsClassifier(n_neighbors=10 , metric= 'minkowski' , p = 4)

elif model_name == 'Gaussian Naive Bayes':

mdl = GaussianNB()

oneVsRest = OneVsRestClassifier(mdl)

oneVsRest.fit(x_train, y_train)

y_pred = oneVsRest.predict(x_test)

# Performance metrics

accuracy = round(accuracy_score(y_test, y_pred) * 100, 2)

# Get precision, recall, f1 scores

precision, recall, f1score, support = score(y_test, y_pred, average='micro')

print(f'Test Accuracy Score of Basic {model_name}: % {accuracy}')

print(f'Precision : {precision}')

print(f'Recall : {recall}')

print(f'F1-score : {f1score}')

# Add performance parameters to list

perform_list.append(dict([
```

```
('Model', model_name),

('Test Accuracy', round(accuracy, 2)),

('Precision', round(precision, 2)),

('Recall', round(recall, 2)),

('F1', round(f1score, 2))

]))
```

Logistic Regression

```
run_model('Logistic Regression', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic Logistic Regression: % 97.09
Precision : 0.970917225950783
Recall    : 0.970917225950783
F1-score   : 0.9709172259507831
```

Random Forest

```
run_model('Random Forest', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic Random Forest: % 97.99
Precision : 0.9798657718120806
Recall    : 0.9798657718120806
F1-score   : 0.9798657718120806
```

Multinomial Naive Bayes

```
run_model('Multinomial Naive Bayes', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic Multinomial Naive Bayes: % 97.09
Precision : 0.970917225950783
Recall    : 0.970917225950783
F1-score   : 0.9709172259507831
```

Support Vector Machine

```
run_model('Support Vector Classifer', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic Support Vector Classifer: % 96.64
Precision : 0.9664429530201343
Recall    : 0.9664429530201343
F1-score   : 0.9664429530201343
```

Decision Tree

```
run_model('Decision Tree Classifier', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic Decision Tree Classifier: % 83.22
Precision : 0.8322147651006712
Recall    : 0.8322147651006712
F1-score   : 0.8322147651006712
```

KNN

```
run_model('K Nearest Neighbour', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic K Nearest Neighbour: % 73.6
Precision : 0.7360178970917226
Recall    : 0.7360178970917226
F1-score   : 0.7360178970917226
```

Gaussian Naive Bayes

```
run_model('Gaussian Naive Bayes', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic Gaussian Naive Bayes: % 76.06
Precision : 0.7606263982102909
Recall    : 0.7606263982102909
F1-score   : 0.7606263982102909
```

Create Dataframe of Model, Accuracy, Precision, Recall, and F1

```
model_performance = pd.DataFrame(data=perform_list)
model_performance = model_performance[['Model', 'Test Accuracy', 'Precision',
'Recall', 'F1']]
model_performance
```

	Model	Test Accuracy	Precision	Recall	F1
0	Logistic Regression	97.09	0.97	0.97	0.97
1	Random Forest	97.99	0.98	0.98	0.98
2	Multinomial Naive Bayes	97.09	0.97	0.97	0.97
3	Support Vector Classifer	96.64	0.97	0.97	0.97
4	Decision Tree Classifier	83.22	0.83	0.83	0.83
5	K Nearest Neighbour	73.60	0.74	0.74	0.74
6	Gaussian Naive Bayes	76.06	0.76	0.76	0.76

Best Model to Perform Accuracy Score

Here, after training and testing the model we find that Random Forest Classifier model has given the best accuracy from all machine learning models.

```
model = model_performance["Model"]
max_value = model_performance["Test Accuracy"].max()
print("The best accuracy of model is", max_value,"from Random")
```

```
The best accuracy of model is 97.99 from Random
```

Fit & predict best ML Model

Here we fit and predict our best model i.e. Random Forest.

```
classifier = RandomForestClassifier(n_estimators=100 ,criterion='entropy' ,
random_state=0).fit(x_train, y_train)
classifier
y_pred = classifier.predict(x_test)
```

Predict News Article

Now, here, after the completion of model analysis, we can also predict any news articles.

```
y_pred1 = cv.transform(['Hour ago, I contemplated retirement for a lot of
reasons. I felt like people were not sensitive enough to my injuries. I felt like
a lot of people were backed, why not me? I have done no less. I have won a lot of
games for the team, and I am not feeling backed, said Ashwin'])
yy = classifier.predict(y_pred1)
result = ""
if yy == [0]:
  result = "Business News"
```

```
elif yy == [1]:
  result = "Tech News"
elif yy == [2]:
  result = "Politics News"
elif yy == [3]:
  result = "Sports News"
elif yy == [1]:
  result = "Entertainment News"
print(result)
```

```
Sports News
```

Conclusion

Finally after doing Data cleaning and Data Preprocessing (cleaning data, train_test_split model, creating a bag of words NLP model, and machine learning model) we got the accuracy scores and we can say that Random Forest Classification gives the best accuracy among all machine learning models.

And at last, we also predict the category of different news articles.

==========

==========