reference material

data preprocessing

sklearn pipeline

vectorizer

In [92]:
```
!python --version
```

```
Python 3.8.8
```

In [93]:
```
# !pip install nltk
# !pip install beautifulsoup4
# !pip install contractions
# !pip install Unidecode
# !pip install textblob
# !pip install pyspellchecker
```

In [94]:
```
pwd
```

Out[94]:
```
'C:\\Users\\sudip'
```

In [201…
```
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 matplotlib.pyplot as plt
import numpy
from sklearn import metrics
```

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

In [176…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
nltk.download("wordnet")
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
import string
import re
import contractions

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

from bs4 import BeautifulSoup
from textblob import TextBlob
from unidecode import unidecode
import contractions
```

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

In [96]:

```
lemmatizer = WordNetLemmatizer()
```

In [97]:

```
dataset = pd.read_csv('flipitnews-data.csv')
```

In [98]:

```
dataset.shape
```

Out[98]: (2225, 2)

In [99]:

```
dataset.head(5)
```

Out[99]:

	Category	Article
0	Technology	tv future in the hands of viewers with home th...
1	Business	worldcom boss left books alone former worldc...
2	Sports	tigers wary of farrell gamble leicester say ...
3	Sports	yeading face newcastle in fa cup premiership s...
4	Entertainment	ocean s twelve raids box office ocean s twelve...

In [100…

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2225 entries, 0 to 2224
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  2225 non-null   object
 1   Article   2225 non-null   object
dtypes: object(2)
memory usage: 34.9+ KB
```

In [101…

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

Out[101…

```
Sports           511
Business         510
Politics         417
Technology       401
Entertainment    386
Name: Category, dtype: int64
```

In [102…

```
target_category = dataset['Category'].unique()
print(target_category)
```

```
['Technology' 'Business' 'Sports' 'Entertainment' 'Politics']
```

In [103…

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

Out[103…

	Category	Article	CategoryId
0	Technology	tv future in the hands of viewers with home th...	0
1	Business	worldcom boss left books alone former worldc...	1
2	Sports	tigers wary of farrell gamble leicester say ...	2
3	Sports	yeading face newcastle in fa cup premiership s...	2
4	Entertainment	ocean s twelve raids box office ocean s twelve...	3

In [104…

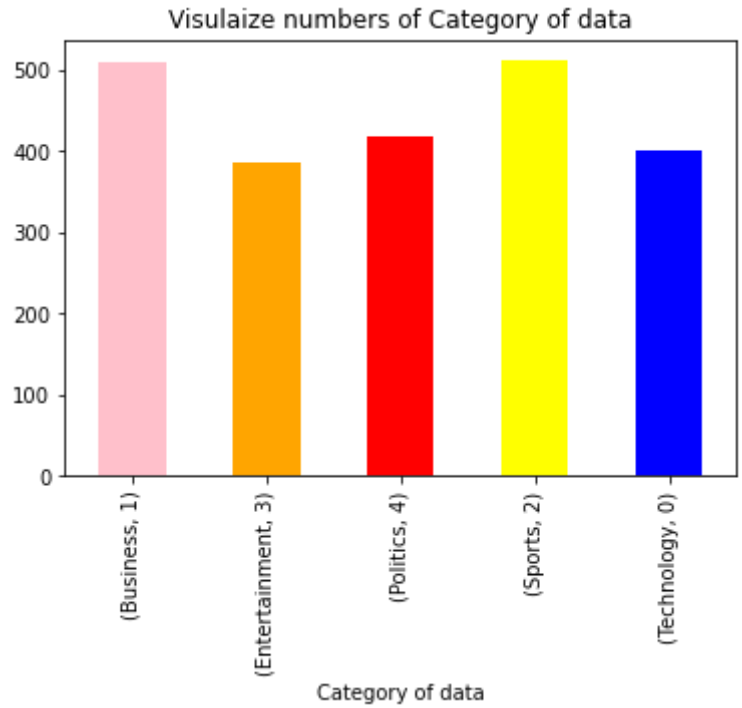
```
category = dataset[['Category', 'CategoryId']].drop_duplicates().sort_values('Catego
category
```

Out[104…

	Category	CategoryId
0	Technology	0
1	Business	1
2	Sports	2
4	Entertainment	3
5	Politics	4

In [105…

```
dataset.groupby('Category').CategoryId.value_counts().plot(kind = "bar", color = ["p
plt.xlabel("Category of data")
plt.title("Visulaize numbers of Category of data")
plt.show()
```



In [106…

```
# !pip install contractions
```

In [107…

```
# import contractions
# def deal_contractions(text):
#     expanded_text = contractions.fix(text)
#     return expanded_text
```

In [108…

```
# dataset['Article'] = dataset['Article'].apply(deal_contractions)
```

how to use contraction

In [117…

In [119…

```
def lemmatize_pos_tagged_text(text, lemmatizer, post_tag_dict):
    sentences = nltk.sent_tokenize(text)
    new_sentences = []

    for sentence in sentences:
        sentence = sentence.lower()
        new_sentence_words = []
        #one pos_tuple for sentence
        pos_tuples = nltk.pos_tag(nltk.word_tokenize(sentence))

    for word_idx, word in enumerate(nltk.word_tokenize(sentence)):
        nltk_word_pos = pos_tuples[word_idx][1]
        wordnet_word_pos = pos_tag_dict.get(
                           nltk_word_pos[0].upper(), None)
        if wordnet_word_pos is not None:
            new_word = lemmatizer.lemmatize(word, wordnet_word_pos)
        else:
            new_word = lemmatizer.lemmatize(word)

        new_sentence_words.append(new_word)
```

```
    new_sentence = " ".join(new_sentence_words)
    new_sentences.append(new_sentence)

    return " ".join(new_sentences)
```

In [129…

```
def download_if_non_existent(res_path, res_name):
    try:
        nltk.data.find(res_path)
    except LookupError:
        print(f'resource {res_path} not found. Downloading now...')
        nltk.download(res_name)
```

In [132…

```
resource corpora/wordnet not found. Downloading now...
resource corpora/omw-1.4 not found. Downloading now...
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\sudip\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\sudip\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

In [177…

```
class NltkPreprocessingSteps:
    def __init__(self, X):
        self.X = X
        download_if_non_existent('corpora/stopwords', 'stopwords')
        download_if_non_existent('tokenizers/punkt', 'punkt')
        download_if_non_existent('taggers/averaged_perceptron_tagger',
                                 'averaged_perceptron_tagger')
        download_if_non_existent('corpora/wordnet', 'wordnet')
        download_if_non_existent('corpora/omw-1.4', 'omw-1.4')
        self.sw_nltk = stopwords.words('english')
        new_stopwords = ['<*>']
        self.sw_nltk.extend(new_stopwords)
        self.sw_nltk.remove('not')
        self.pos_tag_dict = {"J": wordnet.ADJ,
                             "N": wordnet.NOUN,
                             "V": wordnet.VERB,
                             "R": wordnet.ADV}

        # '!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~' 32 punctuations in python
        # we dont want to replace . first time around
        self.remove_punctuations = string.punctuation.replace('.','')


    def deal_contractions(self):
        print('in remove contractions')
        self.X = self.X.apply(lambda x:contractions.fix(x))
        return self

    def remove_html_tags(self):
            print('in remove_html_tags')
            self.X = self.X.apply(
                    lambda x: BeautifulSoup(x, 'html.parser').get_text())
            return self

    def replace_diacritics(self):
            print('in replace_diacritics')
            self.X = self.X.apply(
```

```
                lambda x: unidecode(x, errors="preserve"))
        return self
    def to_lower(self):
        print('in to_lower')
        self.X = np.apply_along_axis(lambda x: x.lower(), self.X)
        return self

    def expand_contractions(self):
        print('in expand_contractions')
        self.X = self.X.apply(
                lambda x: " ".join([contractions.fix(expanded_word)
                                    for expanded_word in x.split()]))
        return self

    def remove_numbers(self):
        print('in remove_numbers')
        self.X = self.X.apply(lambda x: re.sub(r'\d+', '', x))
        return self

    def replace_dots_with_spaces(self):
        print('in replace_dots_with_spaces')
        self.X = self.X.apply(lambda x: re.sub("[.]", " ", x))
        return self

    def remove_punctuations_except_periods(self):
        print('in remove_punctuations_except_periods')
        self.X = self.X.apply(
                    lambda x: re.sub('[%s]' %
                     re.escape(self.remove_punctuations), '' , x))
        return self

    def remove_all_punctuations(self):
        print('in remove_all_punctuations')
        self.X = self.X.apply(lambda x: re.sub('[%s]' %
                                re.escape(string.punctuation), '' , x))
        return self

    def remove_double_spaces(self):
        print('in remove_double_spaces')
        self.X = self.X.apply(lambda x: re.sub(' +', ' ', x))
        return self

    def fix_typos(self):
#             print('in fix_typos')
#             self.X = self.X.apply(lambda x: str(TextBlob(x).correct()))
        return self

    def remove_stopwords(self):
        print('in remove_stopwords')
        # remove stop words from token list in each column
        self.X = self.X.apply(
                lambda x: " ".join([ word for word in x.split()
                         if word not in self.sw_nltk]) )
        return self

    def lemmatize(self):
        print('in lemmatize')
        lemmatizer = WordNetLemmatizer()
        self.X = self.X.apply(lambda x: lemmatize_pos_tagged_text(
                                x, lemmatizer, self.pos_tag_dict))
        return self

    def get_processed_text(self):
        return self.X
```

In [ ]:

In [178…

```
from sklearn.base import BaseEstimator, TransformerMixin
class NltkTextPreprocessor(TransformerMixin, BaseEstimator):
  def __init__(self):
    pass

  def fit(self, X):
    return self

  def transform(self, X):
    txt_preproc = NltkPreprocessingSteps(X.copy())
    processed_text = \
            txt_preproc \
            .deal_contractions()\
            .remove_html_tags()\
            .replace_diacritics()\
            .expand_contractions()\
            .remove_numbers()\
            .fix_typos()\
            .remove_punctuations_except_periods()\
            .lemmatize()\
            .remove_double_spaces()\
            .remove_all_punctuations()\
            .remove_stopwords()\
            .get_processed_text()

    return processed_text
```

In [179…

In [189…

```
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import BernoulliNB
from sklearn.feature_extraction.text import TfidfVectorizer
# X = pd.read_csv("....")
X_train, X_test, y_train, y_test = train_test_split(dataset['Article'], dataset['Cat
pure_transformation_pipeline = Pipeline(steps=[
           ('text_preproc', NltkTextPreprocessor())
            ,('tfidf', TfidfVectorizer(use_idf=True))
])
# pure_transformation_pipeline.fit(X_train)
# Call fit_transform if we only wanted to get transformed data
X_train = pure_transformation_pipeline.fit_transform(X_train)
X_test = pure_transformation_pipeline.transform(X_test)
```

```
resource corpora/wordnet not found. Downloading now...
resource corpora/omw-1.4 not found. Downloading now...
in remove contractions
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\sudip\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\sudip\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
in remove_html_tags
in replace_diacritics
```

```
in expand_contractions
in remove_numbers
in remove_punctuations_except_periods
in lemmatize
in remove_double_spaces
in remove_all_punctuations
in remove_stopwords
resource corpora/wordnet not found. Downloading now...
resource corpora/omw-1.4 not found. Downloading now...
in remove contractions
in remove_html_tags
in replace_diacritics
in expand_contractions
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\sudip\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\sudip\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
in remove_numbers
in remove_punctuations_except_periods
in lemmatize
in remove_double_spaces
in remove_all_punctuations
in remove_stopwords
```

In [ ]:
```
# first_vector_tfidfvectorizer=tfidf_data[0]
```

In [190…
```
x_train, x_test, y_train, y_test = X_train, X_test, y_train, y_test
```

In [191…
```
# get the first vector out (for the first document)
# first_vector_tfidfvectorizer=tfidf_data[0]
# place tf-idf values in a pandas data frame
# df = pd.DataFrame(first_vector_tfidfvectorizer.T.todense(), index=tfidf_vectorizer
# df.sort_values(by=["tfidf"],ascending=False)
# first_vector_tfidfvectorizer
```

In [192…
```
perform_list = [ ]
```

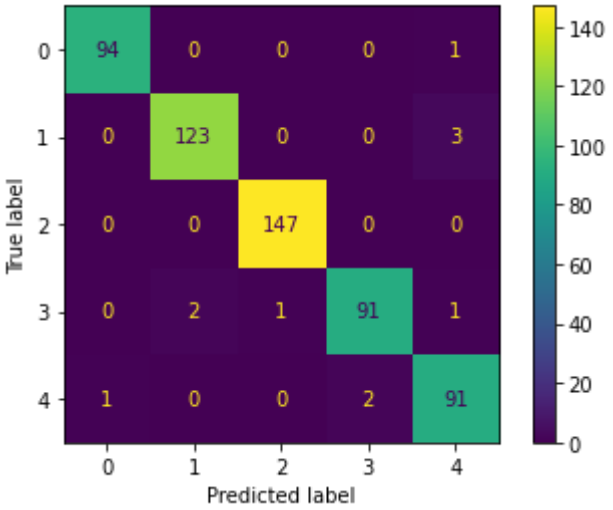
In [212…
```
def run_model(model_name, est_c, est_pnlty):
        mdl=''
        if model_name == 'Logistic Regression':
                mdl = LogisticRegression()
        elif model_name == 'Random Forest':

                mdl = RandomForestClassifier(n_estimators=100 ,criterion='entropy' ,

        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= 'cosine' , 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)
        confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
        cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_mat
        cm_display.plot()
        plt.show()
        # 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))
        ]))
```

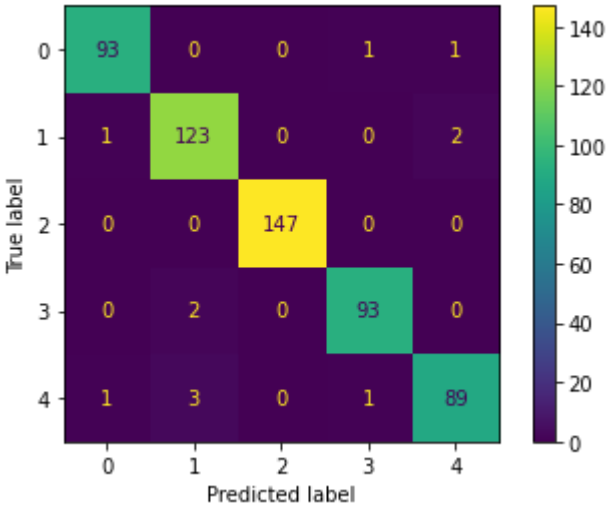
In [ ]:

In [213…

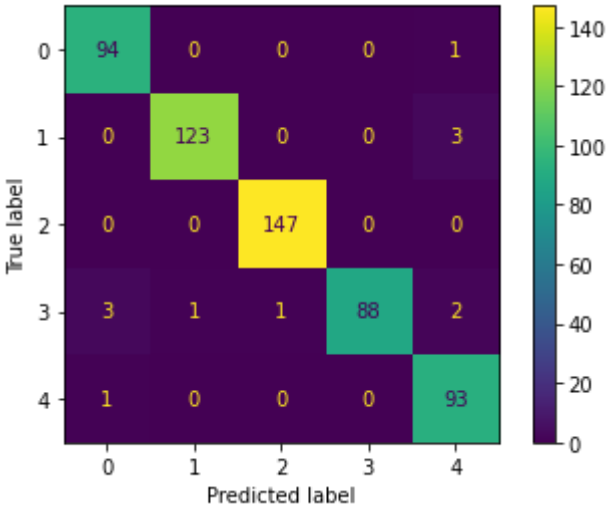
```
run_model('Logistic Regression', est_c=None, est_pnlty=None)
run_model('Random Forest', est_c=None, est_pnlty=None)
run_model('Multinomial Naive Bayes', est_c=None, est_pnlty=None)
run_model('Support Vector Classifer', est_c=None, est_pnlty=None)
run_model('Decision Tree Classifier', est_c=None, est_pnlty=None)
run_model('K Nearest Neighbour', est_c=None, est_pnlty=None)
# run_model('Gaussian Naive Bayes', est_c=None, est_pnlty=None)
```



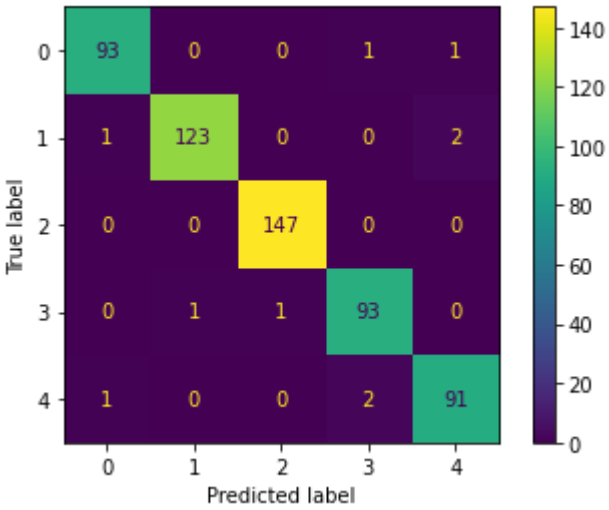

```
Test Accuracy Score of Basic Logistic Regression: % 98.03
Precision : 0.9802513464991023
Recall : 0.9802513464991023
F1-score : 0.9802513464991023
```



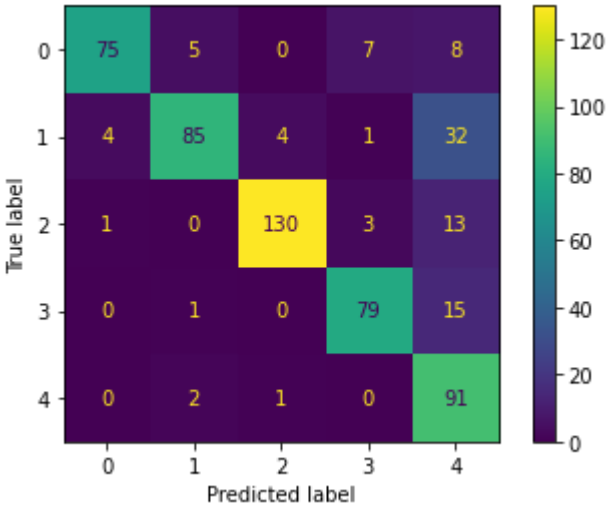
Test Accuracy Score of Basic Random Forest: % 97.85
Precision : 0.9784560143626571
Recall : 0.9784560143626571
F1-score : 0.9784560143626571



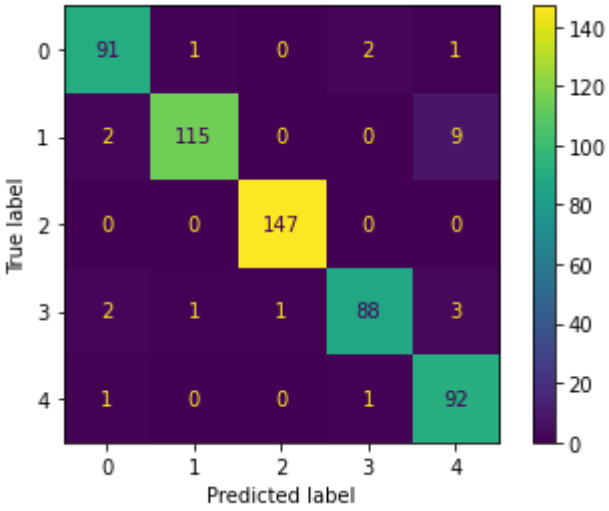
Test Accuracy Score of Basic Multinomial Naive Bayes: % 97.85
Precision : 0.9784560143626571
Recall : 0.9784560143626571
F1-score : 0.9784560143626571



Test Accuracy Score of Basic Support Vector Classifer: % 98.2
Precision : 0.9820466786355476
Recall : 0.9820466786355476
F1-score : 0.9820466786355476




```
Test Accuracy Score of Basic Decision Tree Classifier: % 82.59
Precision : 0.8258527827648114
Recall : 0.8258527827648114
F1-score : 0.8258527827648114
```




```
Test Accuracy Score of Basic K Nearest Neighbour: % 95.69
Precision : 0.9569120287253142
Recall : 0.9569120287253142
F1-score : 0.9569120287253142
```

In [ ]:
```
y_test = y_test.tolist()
```

In [220…
```
mdl = LogisticRegression()
mdl.fit(x_train,y_train)
```

Out[220…
```
LogisticRegression()
```

In [239…
```
prediction = mdl.predict(x_test[114])
```

In [240…
```
prediction
```

Out[240…
```
array([2], dtype=int64)
```

In [243…

In [242...
```
y_test[114]
```

Out[242...
```
2
```

Questions

How many news articles are present in the dataset that we have?

In [247...
```
dataset.shape[0]
```

Out[247...
```
2225
```

Most of the news articles are from _ category.

In [252...
```
dataset['Category'].value_counts().idxmax()
```

Out[252...
```
'Sports'
```

In [253...
```
dataset['Category'].value_counts().max()
```

Out[253...
```
511
```

In [254...
```
dataset['Category'].value_counts()
```

Out[254...
```
Sports           511
Business         510
Politics         417
Technology       401
Entertainment    386
Name: Category, dtype: int64
```

Only ___ no. of articles belong to the 'Technology' category.

In [255...
```
dataset['Category'].value_counts()['Technology']
```

Out[255...
```
401
```

Why we should remove stop words ?

They provide no meaningful information, especially if we are building a text classification model. Therefore, we have to remove stopwords from our dataset. As the frequency of stop words are too high, removing them from the corpus results in much smaller data in terms of size. Reduced size results in faster computations on text data and the text classification model need to deal with a lesser number of features resulting in a robust model.

Explain the difference between Stemming and Lemmatization.

Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.
For instance, lemmatizing the word 'Caring' would return 'Care'. Stemming is used in case of large dataset where performance is an issue.

Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma. For instance, stemming the word 'Caring' would return 'Care'. Lemmatization is computationally expensive since it involves look-up tables and what not.

Which of the techniques Bag of Words or TF-IDF is considered to be more efficient than the other?

An issue with the bag-of-word approach is that it loses the semantic meaning of the words. For instance, not bad semantically means decent or even good. But both the words not and bad bring negative sentiment when considered alone.

There are other approaches like Bag-of-n-Grams, which uses bigram, trigram to capture the words that occur often together. But this leads to an increase in feature space.

In TF-IDF log transform tends to zero out all words that appear in all documents. It effectively means that the word is removed from the feature space. Thus, Tf-idf makes rare words more prominent and effectively ignores common words.

What's the shape of train & test data sets after performing a 75:25 split.

In [256...
```
X_train, X_test, y_train, y_test = train_test_split(dataset['Article'], dataset['Cat
```

In [257...
```
X_train.shape
```

Out[257...
```
(1668,)
```

In [258...
```
X_test.shape
```

Out[258...
```
(557,)
```

Which of the following is found to be the best performing model..

a. Random Forest
b. Nearest Neighbors
c. Naive Bayes

Test Accuracy Score of Basic Random Forest: % 97.85
Precision : 0.9784560143626571
Recall : 0.9784560143626571
F1-score : 0.9784560143626571

Test Accuracy Score of Basic Multinomial Naive Bayes: % 97.85
Precision : 0.9784560143626571
Recall : 0.9784560143626571
F1-score : 0.9784560143626571

Test Accuracy Score of Basic K Nearest Neighbour: % 95.69
Precision : 0.9569120287253142
Recall : 0.9569120287253142
F1-score : 0.9569120287253142

According to this particular use case, both precision and recall are equally important. (T/F)

False

For rare cancer data modeling, anything that doesn't account for false-negatives is a crime.

Recall is a better measure than precision.

For YouTube recommendations, false-negatives is less of a concern. Precision is better here.

In [ ]: