**BBC News Category Prediction**

**Solution Definition:**

We are using a Multinomial Naive Bayes classifier for text classification to predict the category of BBC News statement.

**Multinomial Naive Bayes**

The Multinomial Naive Bayes classifier is designed for classification with discrete features, especially word counts in text classification problems.

**Solution:**

Read the CSV file Training Data using the Pandas library  
  
data\_file = pd.read\_csv('BBC News Train.csv')

Splitting training and testing data to find accuracy of the model.

Training data is 80% and testing data is 20% got an accuracy of 95% with the MulitnomialNB classifier. For data cleaning removed english stopwords from Natural Language Toolkit ([NLTK](https://www.nltk.org/)).

x = data\_file['Text']

y = data\_file['Category']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

vectorizer = TfidfVectorizer(stop\_words=stopwords.words('english'))

x\_train\_vec = vectorizer.fit\_transform(x\_train)

x\_test\_vec = vectorizer.transform(x\_test)

clf = MultinomialNB()

clf.fit(x\_train\_vec, y\_train)

y\_pred = clf.predict(x\_test\_vec)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy}')

Exporting/Saving the trained models to disk using joblib with `.pkl` extension with dump function and then we can load the models in the API.

Dump:

joblib.dump(clf, 'model.pkl')

joblib.dump(vectorizer, 'vectorizer.pkl')

Load:

clfModel = joblib.load('model.pkl')

vector= joblib.load('vectorizer.pkl')

Python API:

Using flask created an API that takes BBC News Input as text and predicts the output category with existing models.

from flask import Flask, request, jsonify

import joblib

app = Flask(\_\_name\_\_)

clfModel = joblib.load('./model/pkl/model.pkl')

vectorizer = joblib.load('./model/pkl/vectorizer.pkl')

@app.route("/", methods=['POST'])

def predictCat():

    data = request.get\_json()

    text=data['text']

    vec = vectorizer.transform(text)

    pred = clfModel.predict(vec)

    res=list(pred)

    return jsonify(res)