News Articles Sorting

Objective

Development of a predictive model for classifying news article in different category . Integrate the model with python module and REST API, to predict a news article category from anywhere. Classified news allows users to access the information of interest quickly and effectively.

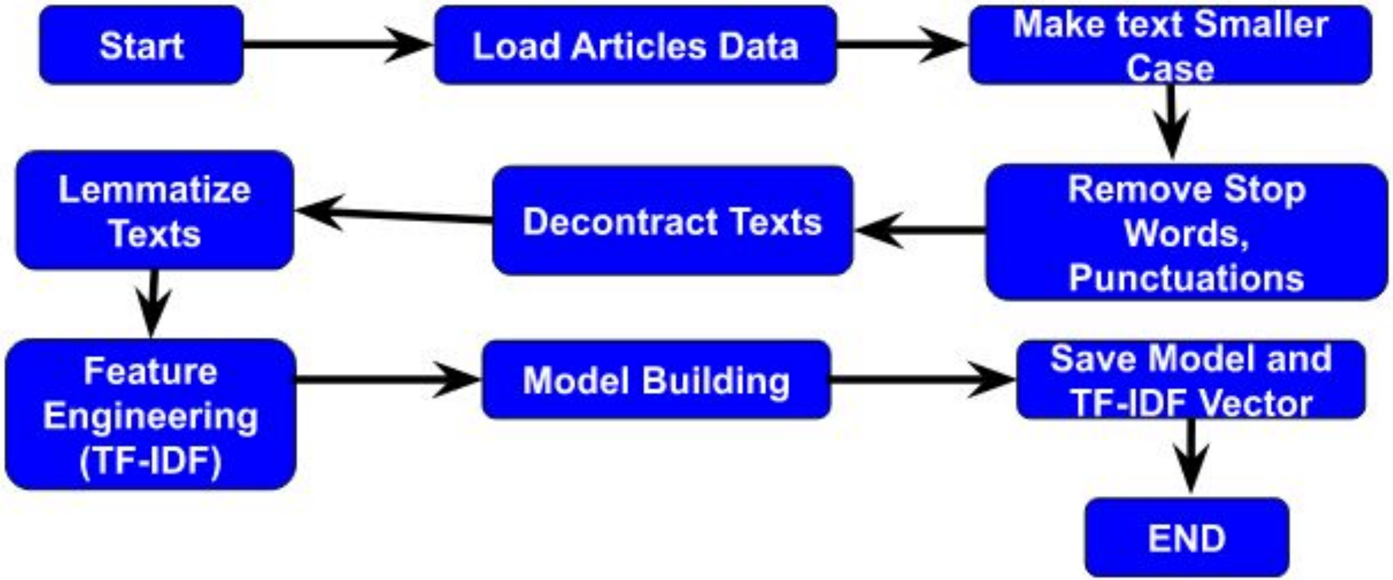
Benefits

1. Python module can detect category of a news articles, can be integrate with any python program

1. Sending Request along with news to our REST API server, user got category of news article as Response, can be integrate with any program written in any language (C/C++, Java, Javascript, Golang etc.)

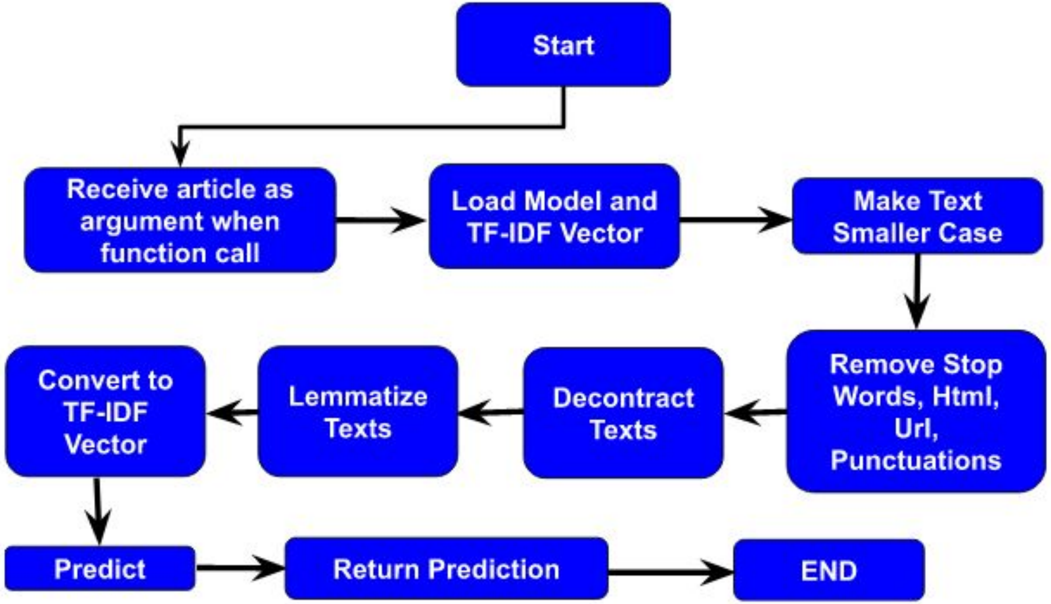
Architecture

Training Model and Save



Architecture

Python Module



Start

Receive article as argument when function call

Load Model and TF-IDF Vector

Make Text Smaller Case

Remove Stop Words, Html, Url, Punctuations

Decontract Texts

Lemmatize Texts

Convert to TF-IDF Vector

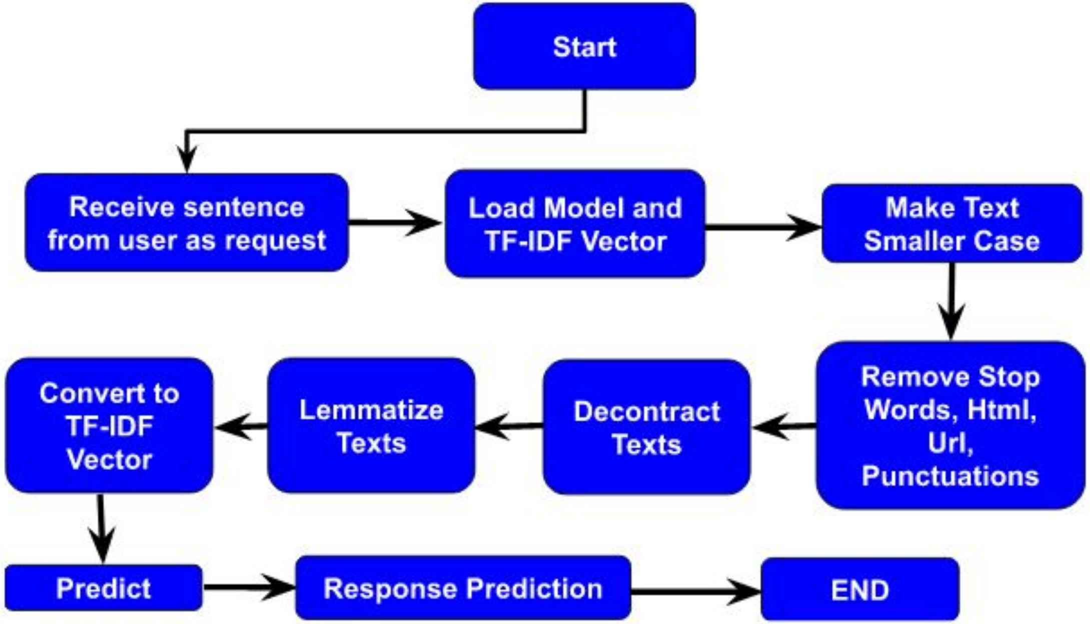
Predict

Return Prediction

END

Architecture

Rest API



Dataset

BBC News Classification

I am using a public dataset from the BBC comprised of 1490 articles, each labeled under one of 5 categories: business, entertainment, politics, sport or tech.

Dataset URL : https://www.kaggle.com/c/learn-ai-bbc/data

Data fields

- **ArticleId** - Article id unique # given to the record
- **Article** - text of the header and article
- **Category** - category of the article (tech, business, sport, entertainment, politics/li>

Text Preprocessing

Text Preprocessing is the practice of cleaning and preparing text data.

- Make smaller case
- Remove html
- Remove URL
- Removing punctuation
- Remove non-alphabetic characters
- Decontracted Text

Feature Engineering

Machines can’t understand characters or words or sentences hence we need to encode these words into some specific numeric form. There are various ways to perform feature extraction from text. Some popular and mostly used are:-

1. **Bag of Words model :** The idea is to take the whole text data and count their frequency of occurrence, and map the words with their frequency. This method doesn’t care about the order of the words.

1. **TF-IDF Model :** The BOW model doesn’t give good results since it has a drawback.
   a. **Term frequency (TF):** Number of times a term has appeared in a document.
   b. **Inverse Document Frequency (IDF):** The inverse document frequency (IDF) is a measure of how rare a word is in a document. If a word appears in almost every document means it’s not significant for the classification

IDF of a word is = log(N/n)
N: total number of documents.
n: number of documents containing a term (word)

TF-IDF Evaluates how relevant is a word to its sentence in a collection of sentences or documents.

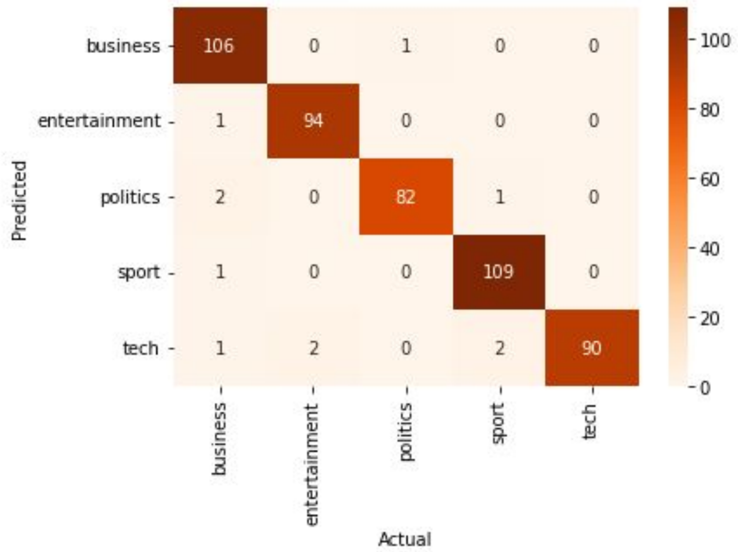
Model Training

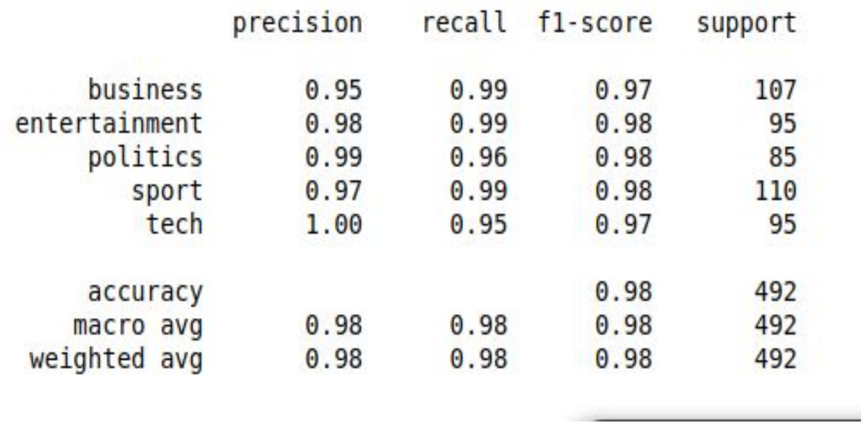
We are taking TF-IDF Vector of the dataset and split it to train and test set. After that we train our model with different algorithm and check accuracy, confusion matrix, precision, recall, f1-score, support etc.

Let's see classification report in more details…

Classification Report

1. Logistic Regression

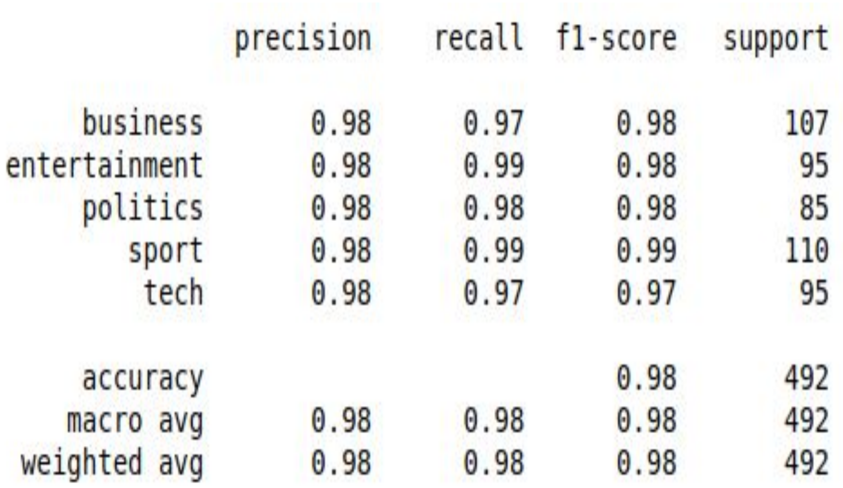




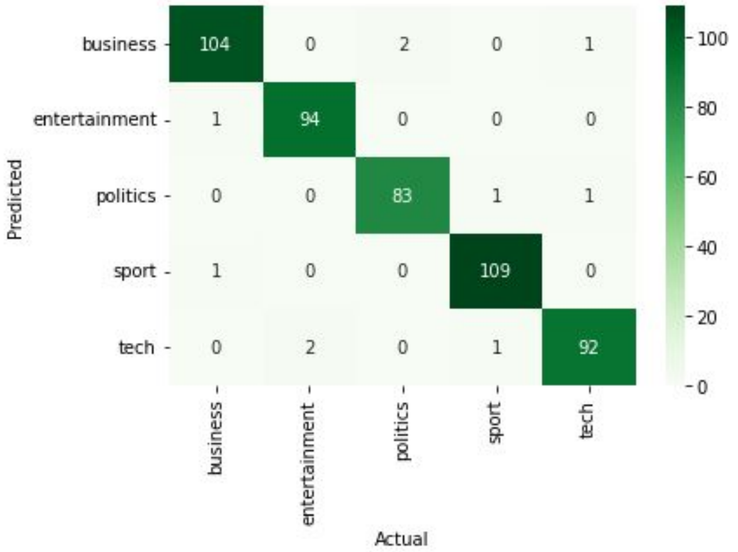
```
               precision    recall  f1-score   support

     business       0.95      0.99      0.97       107
entertainment       0.98      0.99      0.98        95
     politics       0.99      0.96      0.98        85
        sport       0.97      0.99      0.98       110
         tech       1.00      0.95      0.97        95

     accuracy                           0.98       492
    macro avg       0.98      0.98      0.98       492
 weighted avg       0.98      0.98      0.98       492
```

Classification Report

2. Logistic Regression with L2 Regularizations (Alpha = 50)



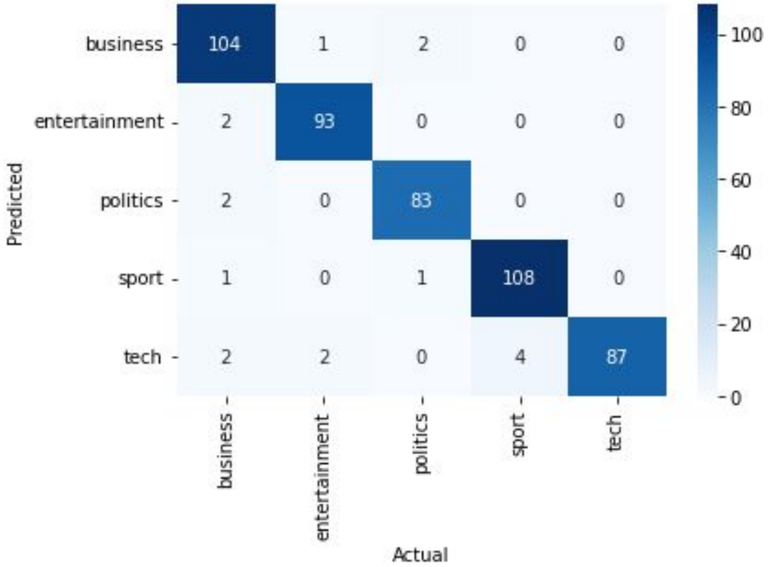
```
               precision    recall  f1-score   support

     business       0.98      0.97      0.98       107
entertainment       0.98      0.99      0.98        95
     politics       0.98      0.98      0.98        85
        sport       0.98      0.99      0.99       110
         tech       0.98      0.97      0.97        95

     accuracy                           0.98       492
    macro avg       0.98      0.98      0.98       492
 weighted avg       0.98      0.98      0.98       492
```

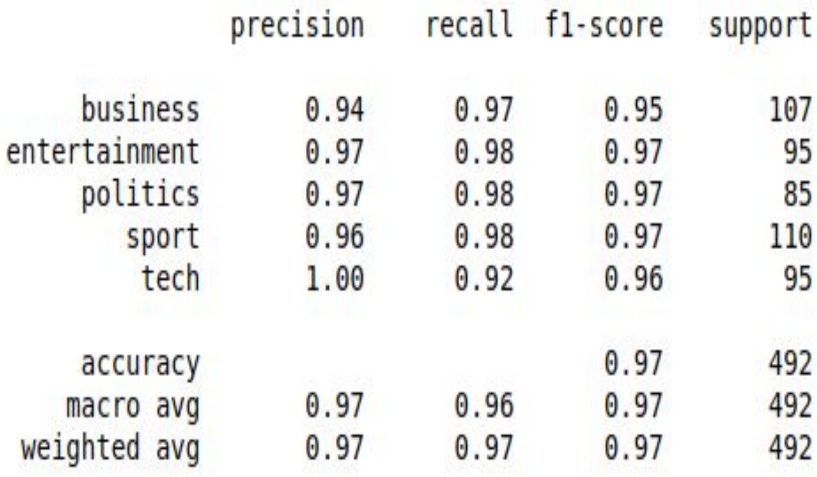


Predicted \ Actual	business	entertainment	politics	sport	tech
business	104	0	2	0	1
entertainment	1	94	0	0	0
politics	0	0	83	1	1
sport	1	0	0	109	0
tech	0	2	0	1	92

Classification Report

3. Logistic Regression with L1 Regularizations (Alpha = 20)

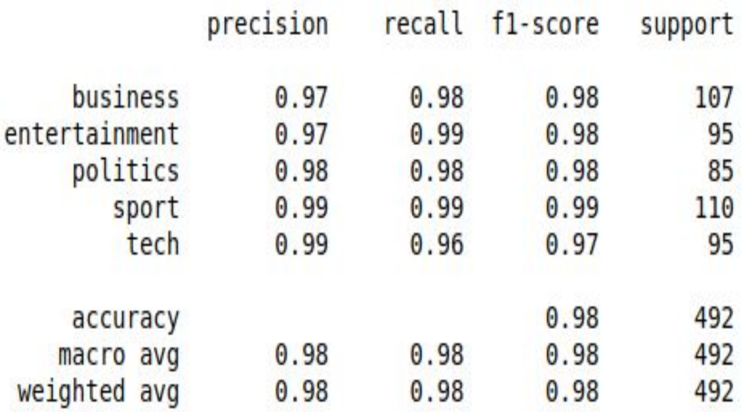




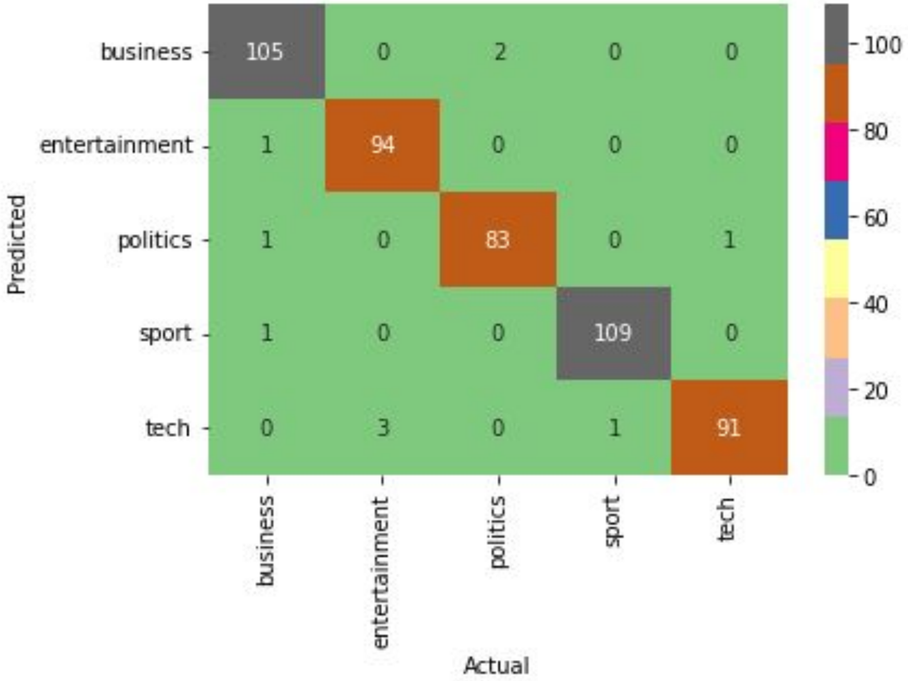
```
               precision    recall  f1-score   support

     business       0.94      0.97      0.95       107
entertainment       0.97      0.98      0.97        95
     politics       0.97      0.98      0.97        85
        sport       0.96      0.98      0.97       110
         tech       1.00      0.92      0.96        95

     accuracy                           0.97       492
    macro avg       0.97      0.96      0.97       492
 weighted avg       0.97      0.97      0.97       492
```

Classification Report

4. Support Vector Machine (Linear Kernel)

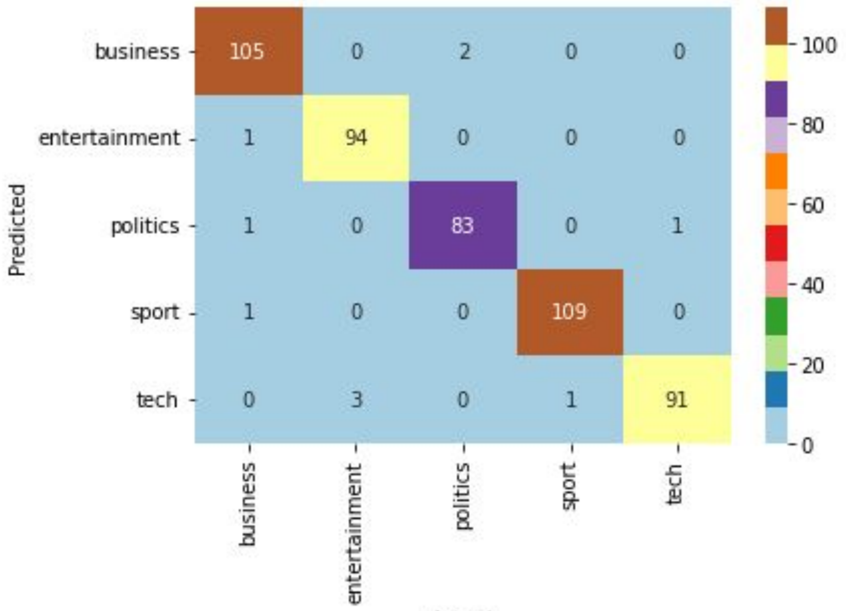


```
               precision    recall  f1-score   support

     business       0.97      0.98      0.98       107
entertainment       0.97      0.99      0.98        95
     politics       0.98      0.98      0.98        85
        sport       0.99      0.99      0.99       110
         tech       0.99      0.96      0.97        95

     accuracy                           0.98       492
    macro avg       0.98      0.98      0.98       492
 weighted avg       0.98      0.98      0.98       492
```

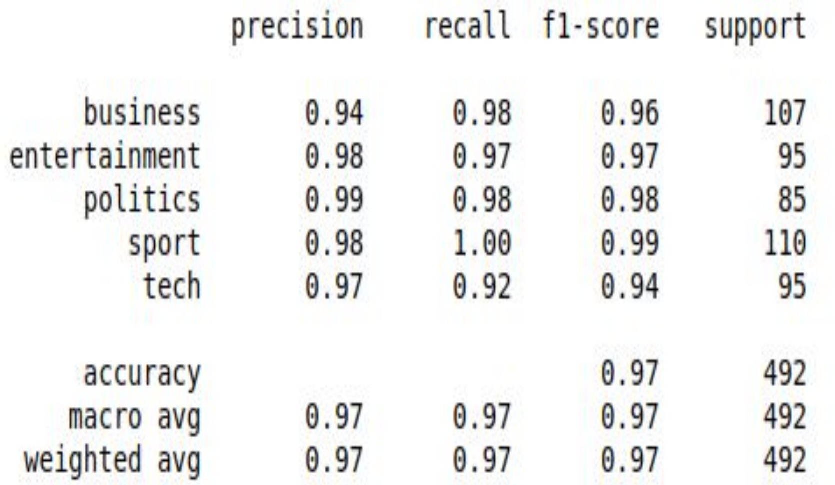


Classification Report

5. Random Forest Classifier



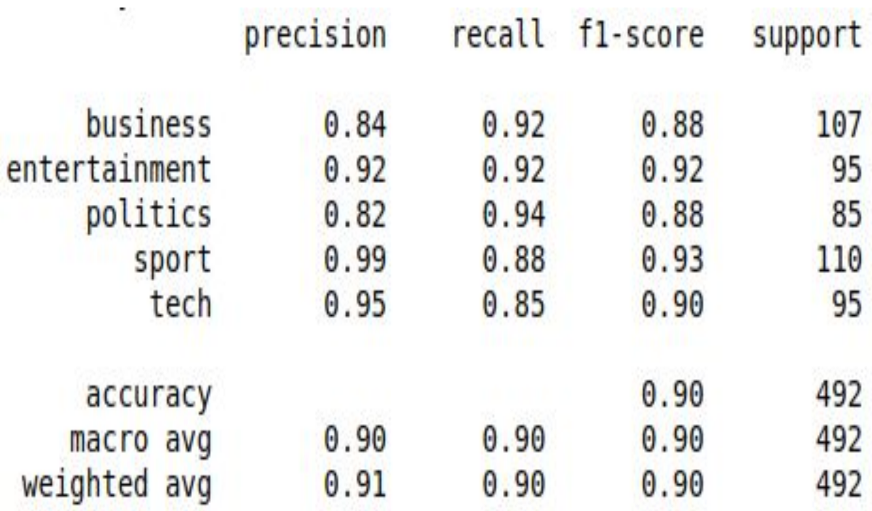
Predicted \ Actual	business	entertainment	politics	sport	tech
business	105	0	2	0	0
entertainment	1	94	0	0	0
politics	1	0	83	0	1
sport	1	0	0	109	0
tech	0	3	0	1	91



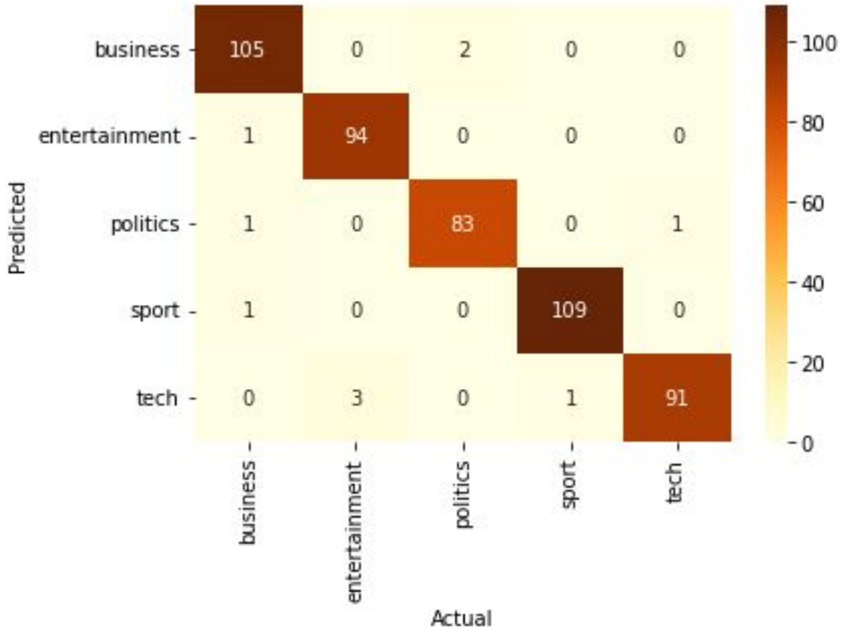
```
               precision    recall  f1-score   support

     business       0.94      0.98      0.96       107
entertainment       0.98      0.97      0.97        95
     politics       0.99      0.98      0.98        85
        sport       0.98      1.00      0.99       110
         tech       0.97      0.92      0.94        95

     accuracy                           0.97       492
    macro avg       0.97      0.97      0.97       492
 weighted avg       0.97      0.97      0.97       492
```

Classification Report

6. K Nearest Neighbors

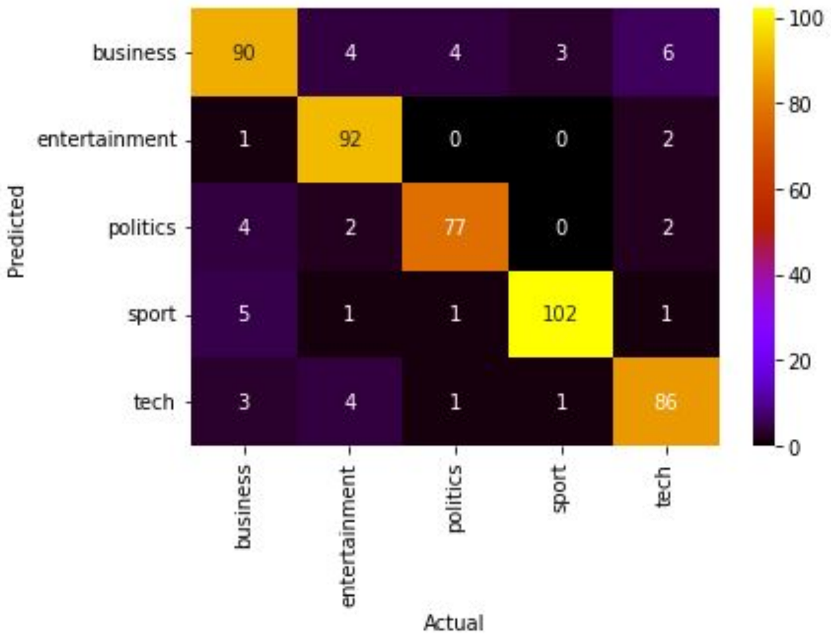


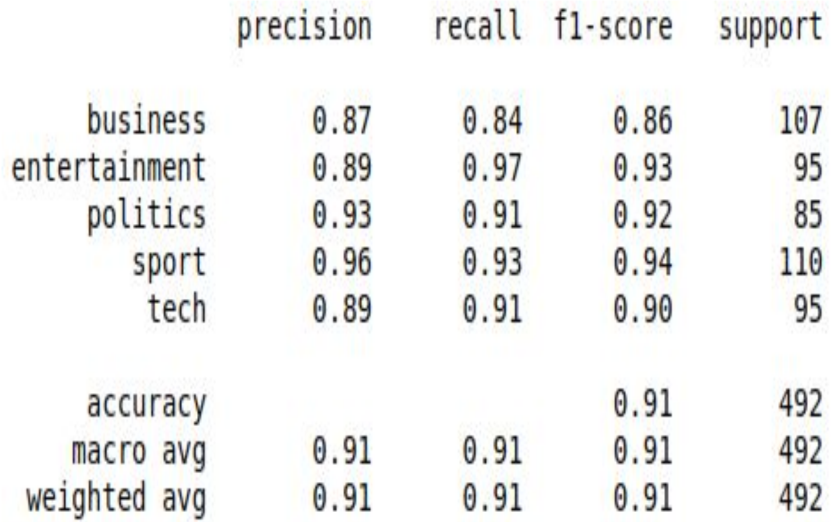
```
               precision    recall  f1-score   support

     business       0.84      0.92      0.88       107
entertainment       0.92      0.92      0.92        95
     politics       0.82      0.94      0.88        85
        sport       0.99      0.88      0.93       110
         tech       0.95      0.85      0.90        95

     accuracy                           0.90       492
    macro avg       0.90      0.90      0.90       492
 weighted avg       0.91      0.90      0.90       492
```



Classification Report

7. Gaussian Naive Bayes

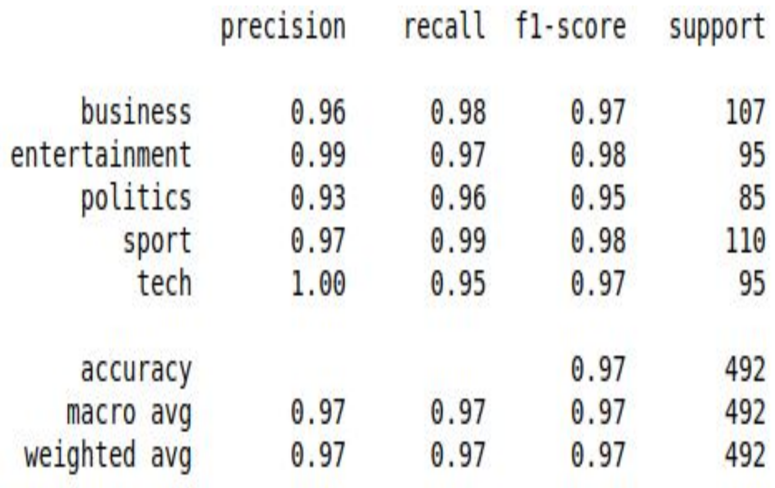




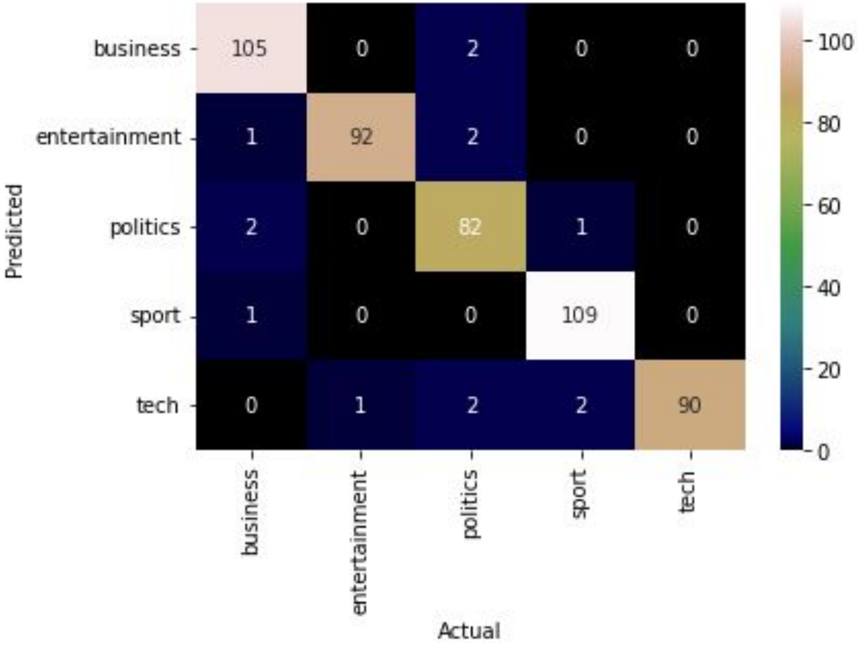
```
               precision    recall  f1-score   support

     business       0.87      0.84      0.86       107
entertainment       0.89      0.97      0.93        95
     politics       0.93      0.91      0.92        85
        sport       0.96      0.93      0.94       110
         tech       0.89      0.91      0.90        95

     accuracy                           0.91       492
    macro avg       0.91      0.91      0.91       492
 weighted avg       0.91      0.91      0.91       492
```

Classification Report

8. Multinomial Naive Bayes

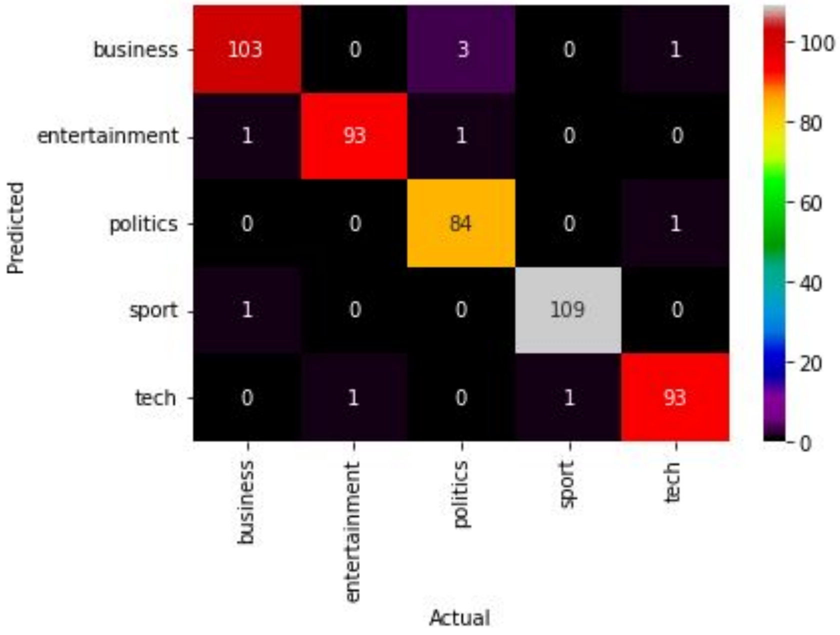


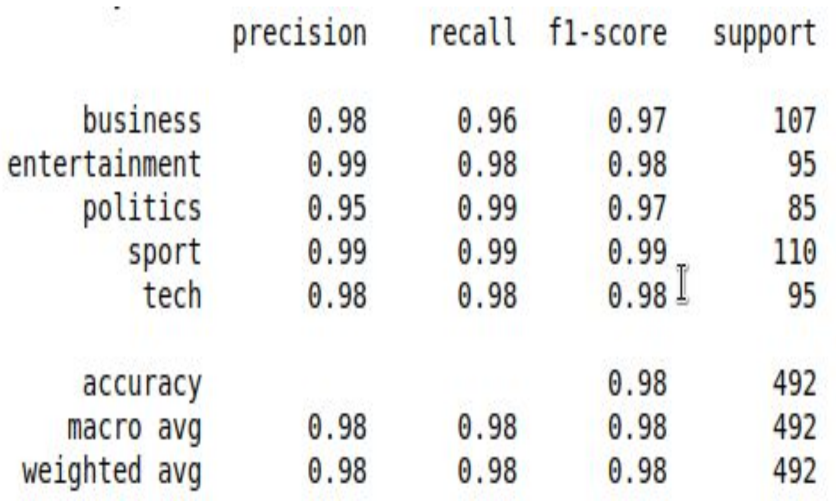
```
               precision    recall  f1-score   support

     business       0.96      0.98      0.97       107
entertainment       0.99      0.97      0.98        95
     politics       0.93      0.96      0.95        85
        sport       0.97      0.99      0.98       110
         tech       1.00      0.95      0.97        95

     accuracy                           0.97       492
    macro avg       0.97      0.97      0.97       492
 weighted avg       0.97      0.97      0.97       492
```



Classification Report

9. Complement Naive Bayes

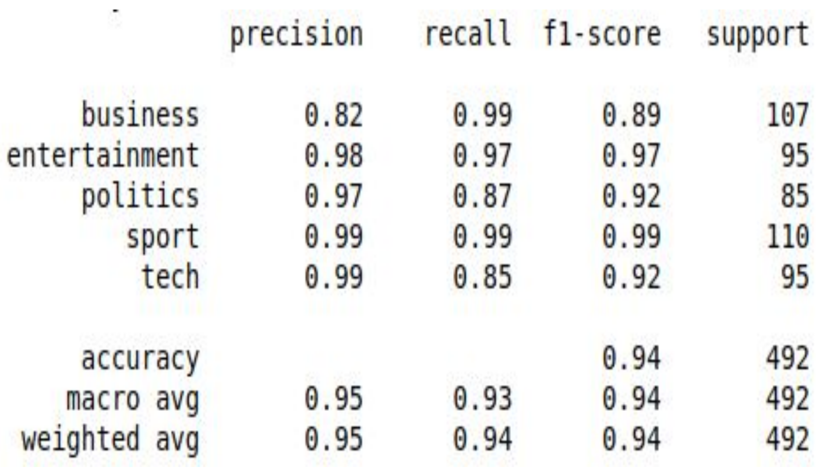




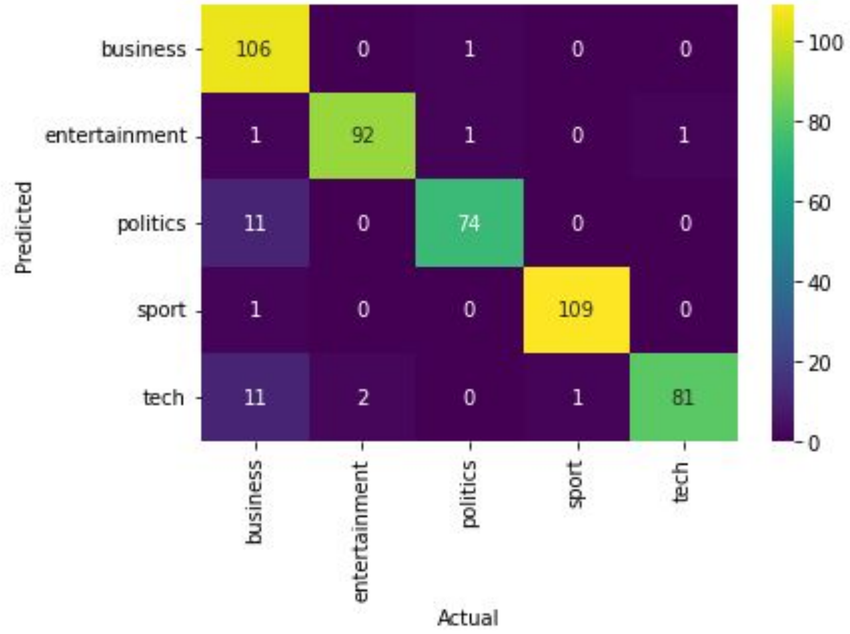
```
               precision    recall  f1-score   support

     business       0.98      0.96      0.97       107
entertainment       0.99      0.98      0.98        95
     politics       0.95      0.99      0.97        85
        sport       0.99      0.99      0.99       110
         tech       0.98      0.98      0.98        95

     accuracy                           0.98       492
    macro avg       0.98      0.98      0.98       492
 weighted avg       0.98      0.98      0.98       492
```

Classification Report

10. Bernoulli Naive Bayes



```
               precision    recall  f1-score   support

     business       0.82      0.99      0.89       107
entertainment       0.98      0.97      0.97        95
     politics       0.97      0.87      0.92        85
        sport       0.99      0.99      0.99       110
         tech       0.99      0.85      0.92        95

     accuracy                           0.94       492
    macro avg       0.95      0.93      0.94       492
 weighted avg       0.95      0.94      0.94       492
```



	business	entertainment	politics	sport	tech
business	106	0	1	0	0
entertainment	1	92	1	0	1
politics	11	0	74	0	0
sport	1	0	0	109	0
tech	11	2	0	1	81

Model Selection

Almost every algorithm working well, we are taking Complement Naive Bayes

- It gives good accuracy
- For all category precision is same (98%).

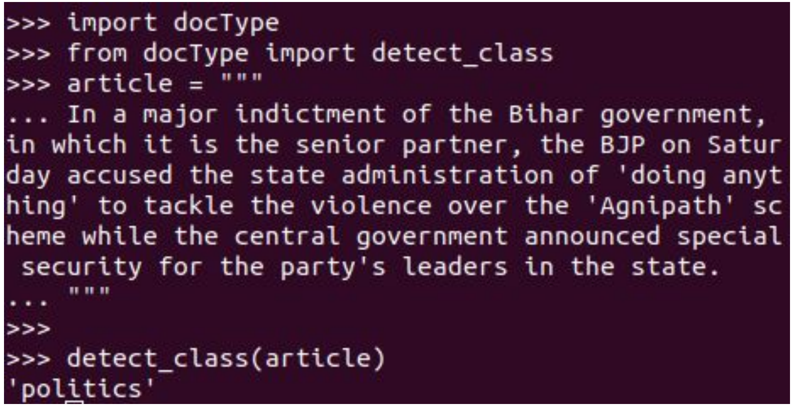
Save TF-IDF Vector and Model as pickle.

Prediction

- Get article as request (for REST API) or argument(for python module).
- Make smaller case
- Remove html
- Remove URL
- Removing punctuation
- Remove non-alphabetic characters
- Decontracted Text
- Lemmatize Text
- Load TF-IDF vector and convert article to TF-IDF vector.
- Load Model
- Predict category of news article

What is docType Python Module ?

1. docType is a python module available on pip perform this classification task by 3 lines of code.
2. User have to install it by **pip install docType.**
3. Assign article to a string.
4. Call **detect_class** function and pass news article as argument of function.
5. And we receive category of the article.



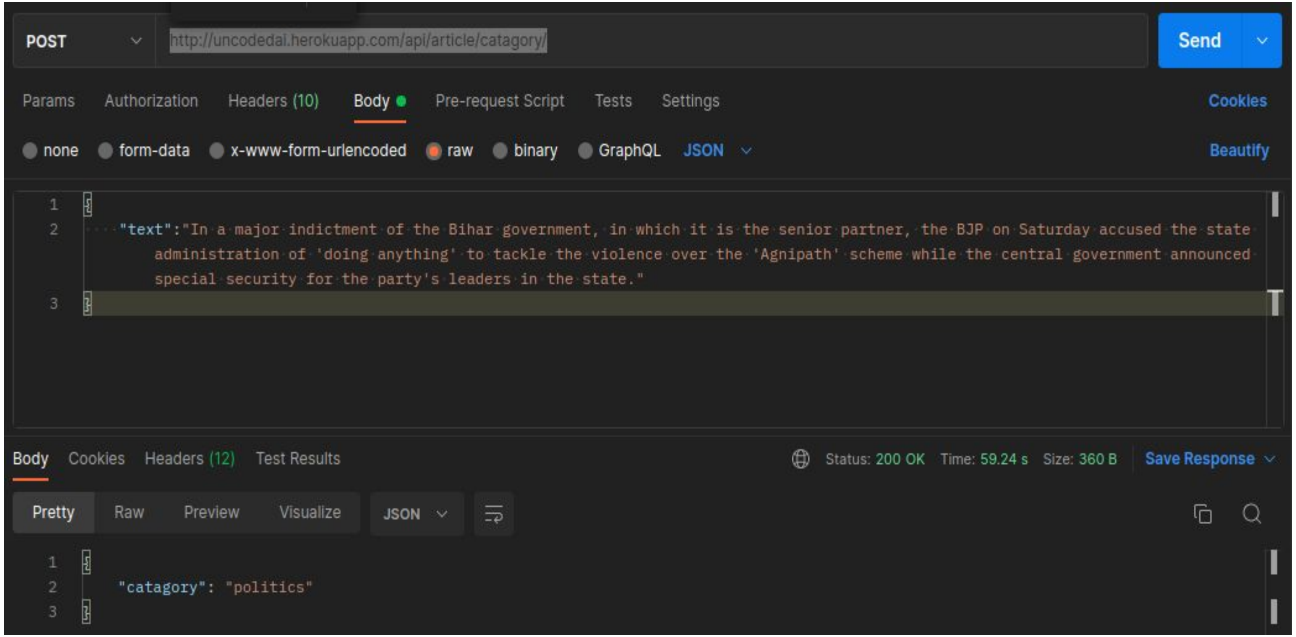
```
>>> import docType
>>> from docType import detect_class
>>> article = """
... In a major indictment of the Bihar government,
in which it is the senior partner, the BJP on Satur
day accused the state administration of 'doing anyt
hing' to tackle the violence over the 'Agnipath' sc
heme while the central government announced special
 security for the party's leaders in the state.
... """
>>>
>>> detect_class(article)
'politics'
```

How docType works ?

- Get news article through argument whenever someone call **detect_class** function and pass article as argument.
- Make smaller case
- Remove html
- Remove URL
- Removing punctuation
- Remove non-alphabetic characters
- Decontracted Text
- Lemmatize Text
- Load TF-IDF vector and convert article to TF-IDF vector.
- Load Model
- Predict category of news article and return it.

About REST API

1. REST Api is deployed on http://uncodedai.herokuapp.com/api/article/catagory.
2. User have to do a POST Request with news article.
3. Using docType module our server predict category of the article and send it as response.



POST http://uncodedai.herokuapp.com/api/article/catagory/

Body: ● raw JSON

```
{
    "text":"In a major indictment of the Bihar government, in which it is the senior partner, the BJP on Saturday accused the state administration of 'doing anything' to tackle the violence over the 'Agnipath' scheme while the central government announced special security for the party's leaders in the state."
}
```

Status: 200 OK Time: 59.24 s Size: 360 B

Pretty JSON

```
{
    "catagory": "politics"
}
```

END