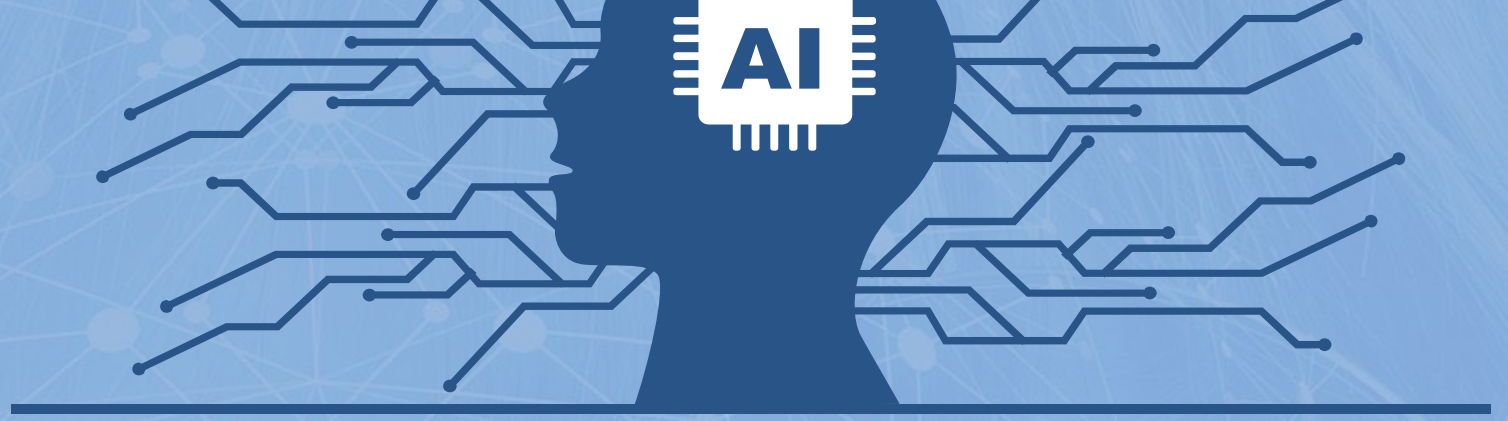


Artificial Intelligence Laboratory

Presentation Name : News Classification

Course Title: Artificial Intelligence Laboratory
Course Code: CSE 318

Our Teacher Details

ARTIFICIAL INTELLIGENCE

Work Experience

Prerequisite: CSE 111 (Structured Programming) or instructor's consent.
Session: Fall 2020

Instructor: Md. Ataur Rahman (AR)

Email address: shaoncsecu@gmail.com

Institution: Department of Computer Science and Engineering, Premier University

Team Name: Spy_FS

Meet Our Team Member

At a glance

Sadia Chowdhury Dola

ID: 1703310201465
Sec: B

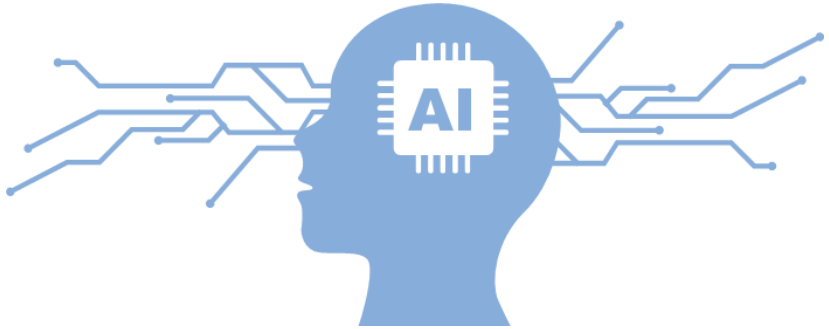
Student

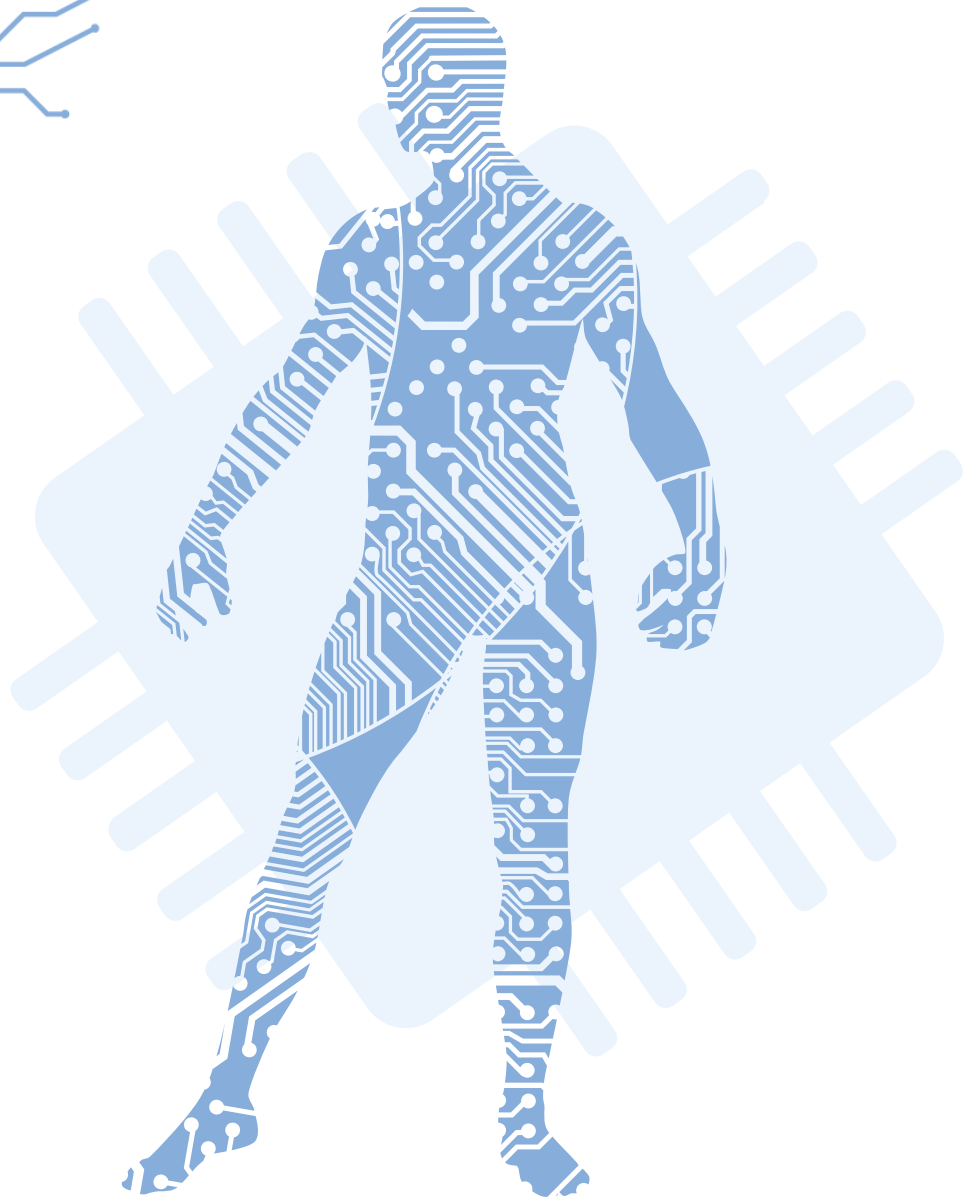
Fariha Chowdhury Bristy

ID: 1703310201445
Sec: B

Student

Contents



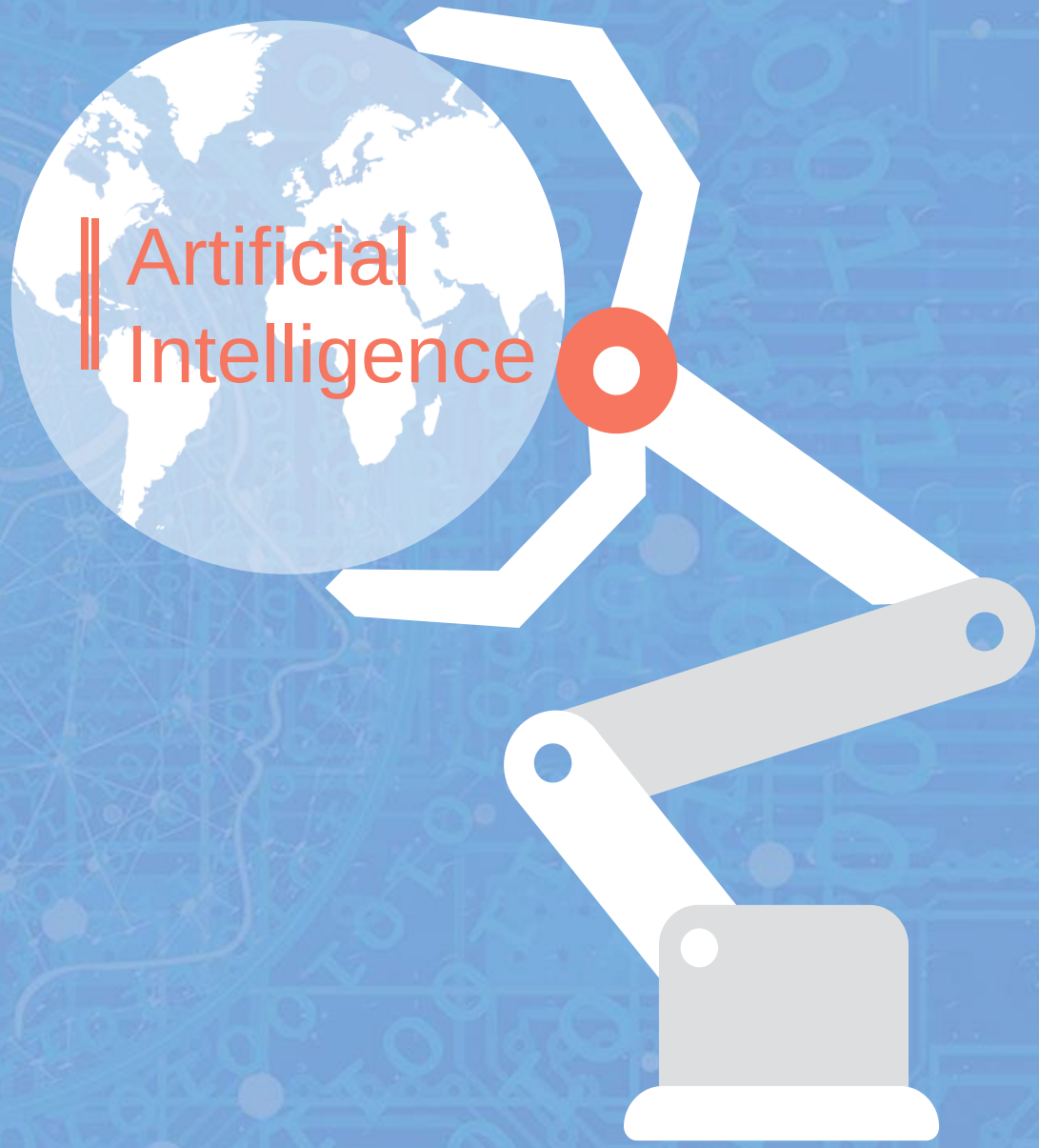


Section Break

News Classification

Introduction

Machine learning (ML) is the study of computer algorithms. Support Vector Machine and Naive Bayes classifier is a classification of ML in supervised learning. Those algorithm are quite popular to be used in NLP. Support Vector Machine gives better accuracy than KNN, Decision Trees and Naive Bayes Classifier and hence is quite useful.



Artificial Intelligence

Model Classifier

What is Naive Bayes Classifier?

Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes Classification Workflow

The classification has two phases, a learning phase, and the evaluation phase.

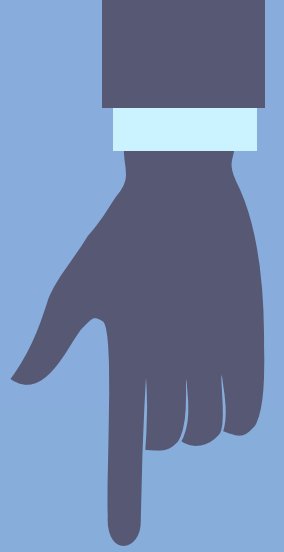
What is Support Vector Machines ?

Support Vector Machines is considered to be a classification approach and regression problems. It can easily handle multidimensional space to separate different classes.

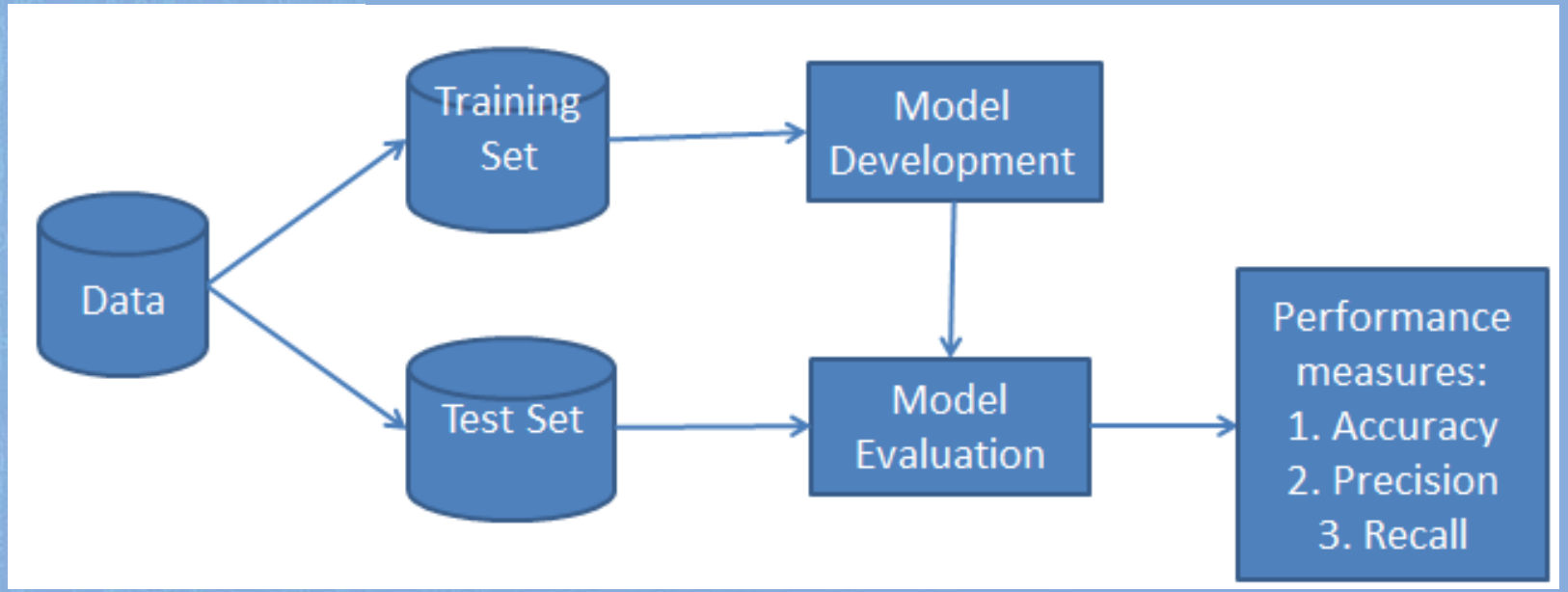
Support Vector Machines Classification Workflow

SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.

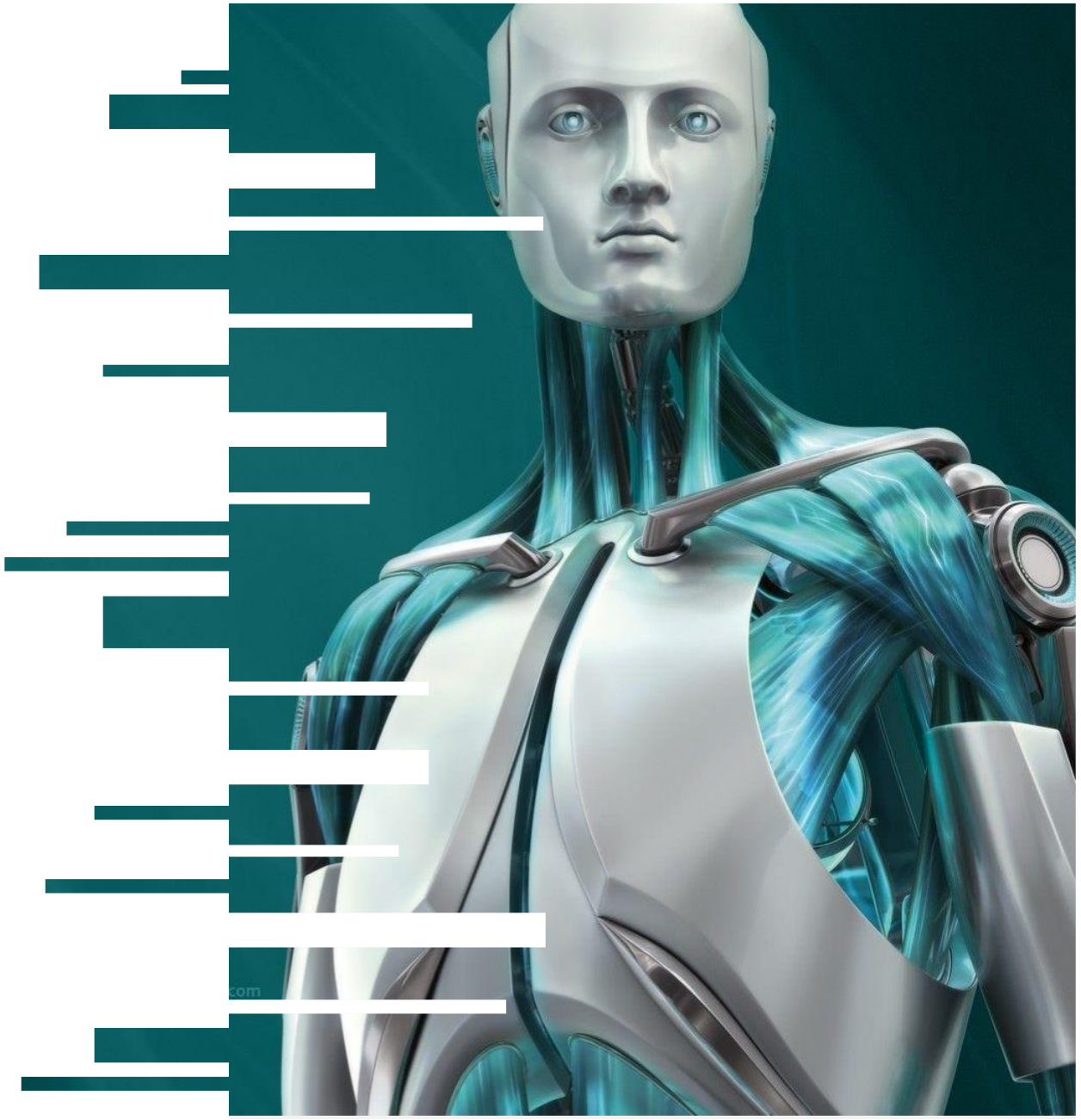
News Classification



Classifier Workflow



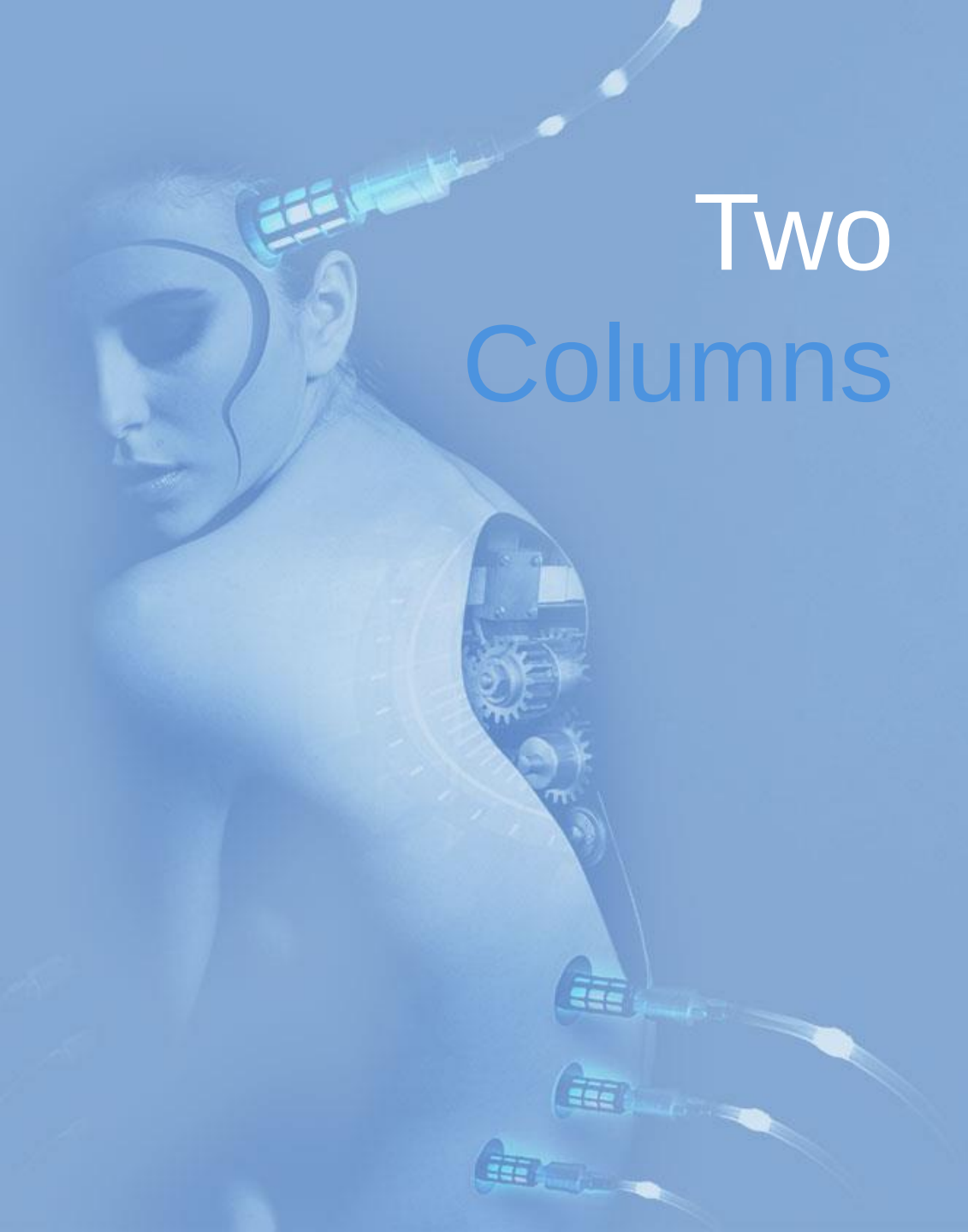
ARTIFICIAL INTELLIGENCE



Working Procedure

News classification

- **Datasets**
- **Pre- processing**
- **All functions**



Two Columns

01

Datasets

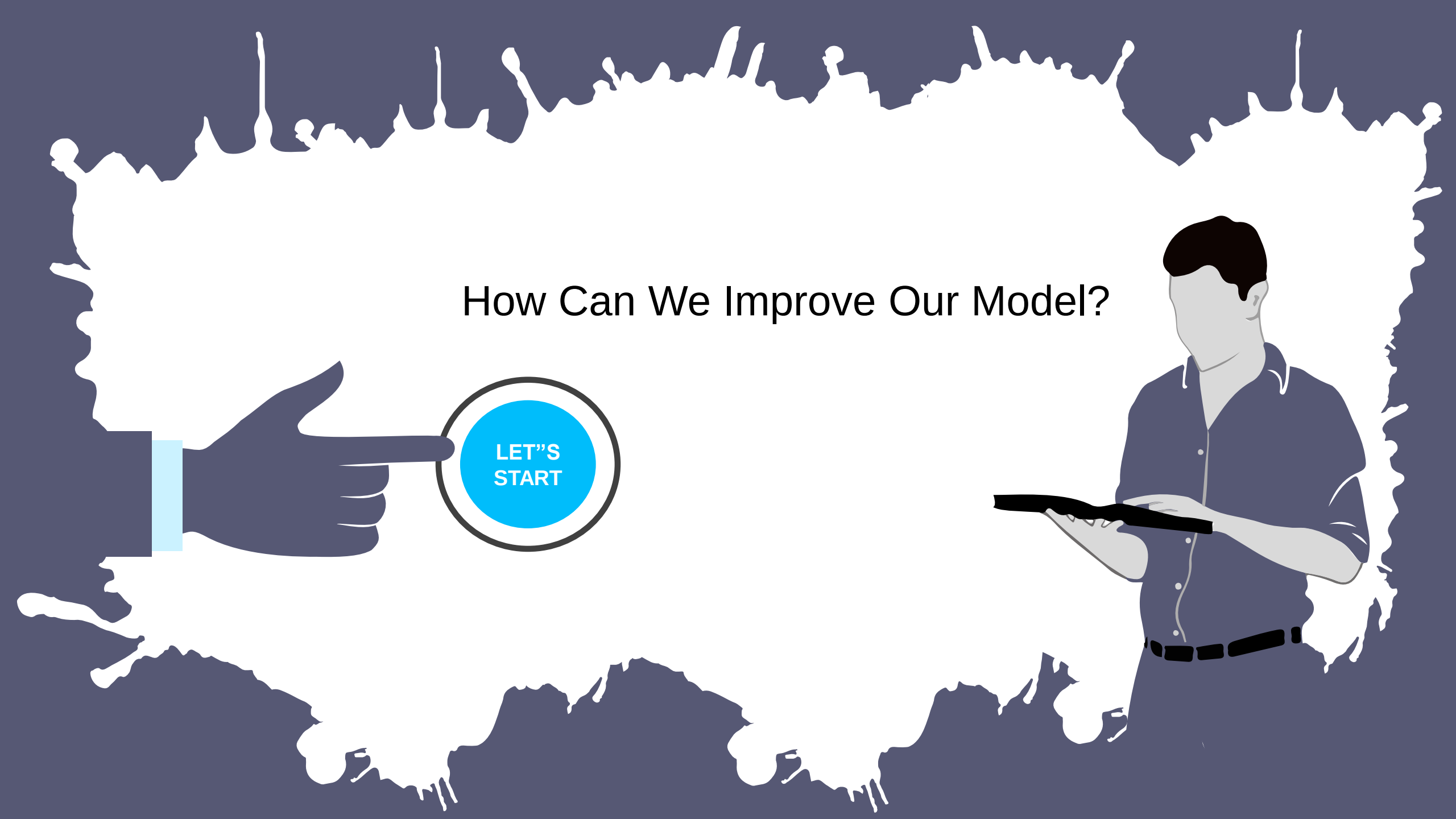
For the project, we are using the dataset containing about News Classification datasets. The reviews came from four classes (Business, Sci_Tech, Sports, World). The actual training data (‘train.txt’) contains a total of 1,20,000 reviews while the test/development data (‘dev.txt’) contains around 4,600 labelled examples. Also encoding ‘utf-8’. The training data is so large ,so we took 15,000 labelled examples to improved our model.

02

Pre-Processing List

1. Remove URL
2. Remove Hashtag
3. Remove Whitespaces
4. Tokenize Sentence
5. Remove Punctuation
6. Remove Stopwords
7. Apply Stemmer
8. Lemmatize Word.

How Can We Improve Our Model?



01 Presentation with News Classification Datasets

Import Library

Necessary libraries download and import. For example, sklearn, nltk, MultinomialNB, Pipeline etc.

Step 1

Google Drive

Mounted at /content/drive.

Step 2

Read Files

Read text data from a file in the drive, also encoding='utf-8‘.

Step 3

Separate Labels

Separate the labels/class and examples/documents from the dataset and split the whole document where it gets a tab

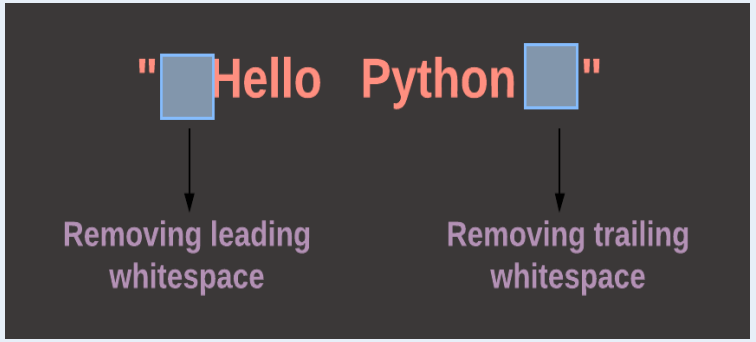
Step 4

News Classification

02

Pre-Processing

Remove URL, Remove Hashtag & Remove Whitespaces



Removes URL's from Texts

Remove all occurrences of # from the texts

Removes multiple whitespaces and replace them with a single whitespace

Pre-Processing

Tokenization

✓ Tokenization

is a process of breaking up a piece of **text** into many pieces, such as sentences and words. It works by separating words using spaces and punctuation.

```
[1]: from nltk.tokenize import sent_tokenize

     sentence = "I love ice cream. I also like steak."
     sent_tokenize(sentence)

[1]: ['I love ice cream.', 'I also like steak.']
```

Takes a line and provides tokens/words by splitting them using NLTK

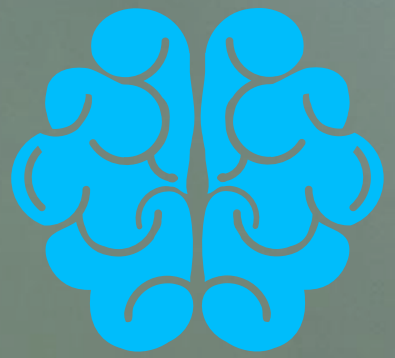
Pre-Processing

Remove Punctuation

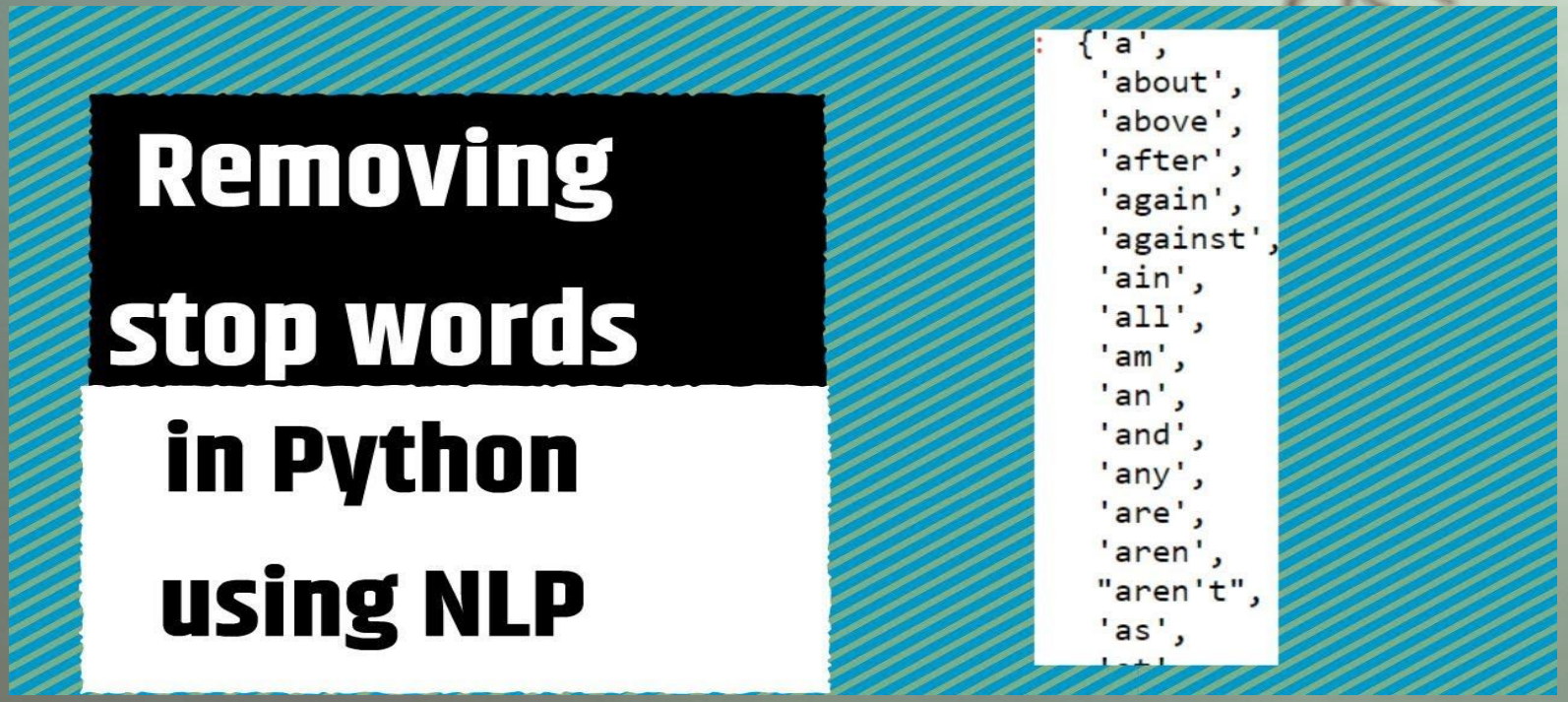
.	Full Stop or Period	()	Round Brackets
,	Comma	[]	Square Brackets
;	Semi-colon	""	Quatation Marks
:	Colon	...	Ellipsis Marks
?	Question Mark	/	Slash
!	Exclamation Mark	_	Underscore
'	Apostrophe	-	Hyphen
–	Underline	@	At sign

Remove all punctuation(? , ; etc.)

Pre-Processing



Remove stopwords



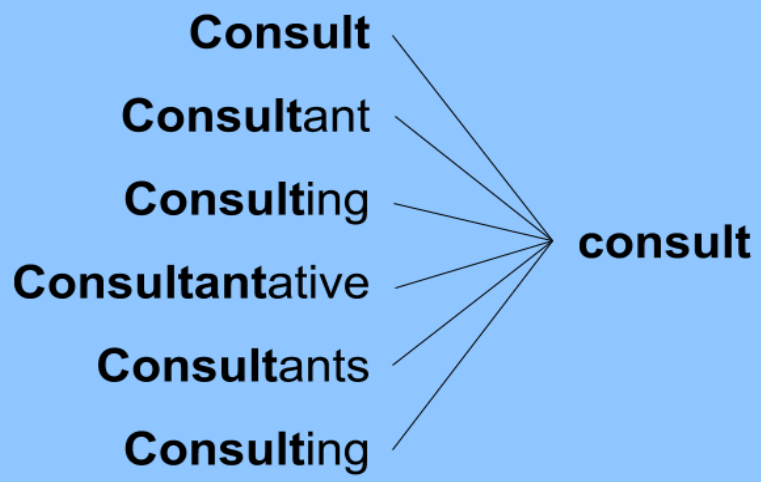
Remove all stopwords(is, are, and, we, from etc.)

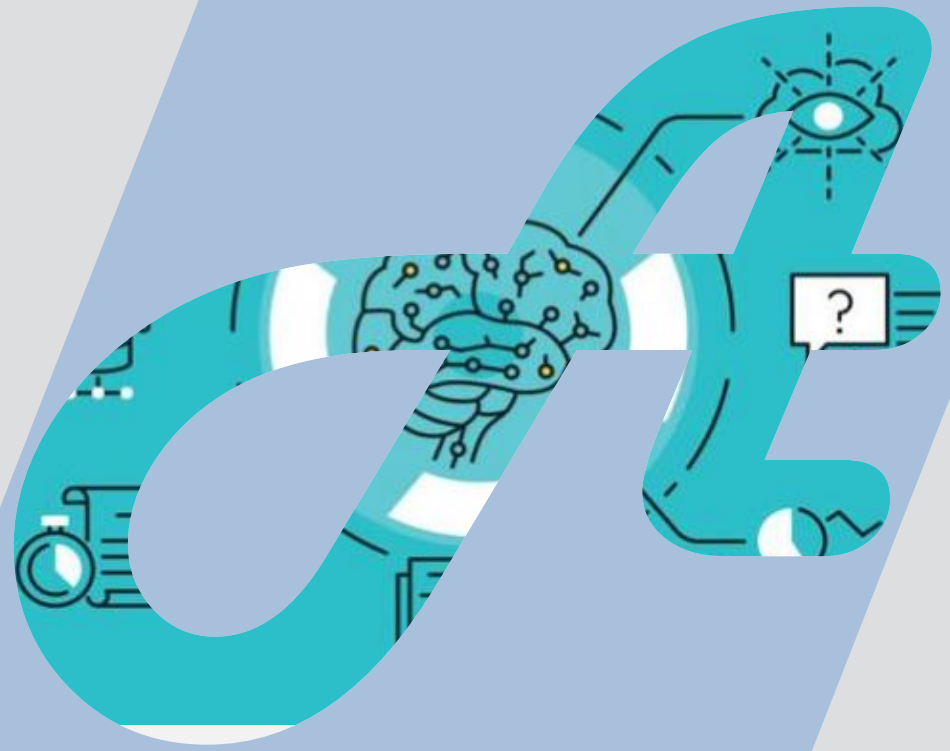
Pre-Processing

ARTIFICIAL

INTELLIGENCE

Stemmer





Used to extract the base form of the words by removing affixes

Pre-Processing

Lemmatization

Lemmatization

Mapping from text-word to lemma
help (verb)

text-word	to	**lemma**
help		help (v)
helps		help (v)
helping		help (v)
helped		help (v)

Remove inflectional endings only and to return the base for of a word.

All Function

Artificial Intelligence

All Function

CountVectorizer ()

Counts the word frequencies and convert to a collection of text documents.

TfidfVectorizer ()

Stands for term frequency – inverse document frequency, is a numerical statistic.

Accuracy score ():

Determine the accuracy of the model based on the labels the model.

Pipeline ()

Used to make an object of classifier where I pass tfidf vectorizer or count vectorizer, multinomialNB and SVM at the same time .

Classification report()

Determine the precision , recall , f1-score , support , accuracy , macro avg and weighted avg .

Artificial Intelligence

All Function

classifier.fit ()

Passed the train data into the classifier .

confusion_matrix()

This function is used to evaluate the performance of the model .

classifier.predict ()

Passed test document into the method and the model will predict the labels of the test document .

We Also Used Some Function

- **char_n_gram_ready**
  Takes space and join the string. It is using for remove punctuation, stopwords etc.

- **identity(X)**
  We use a dummy function as tokenizer and preprocessor.

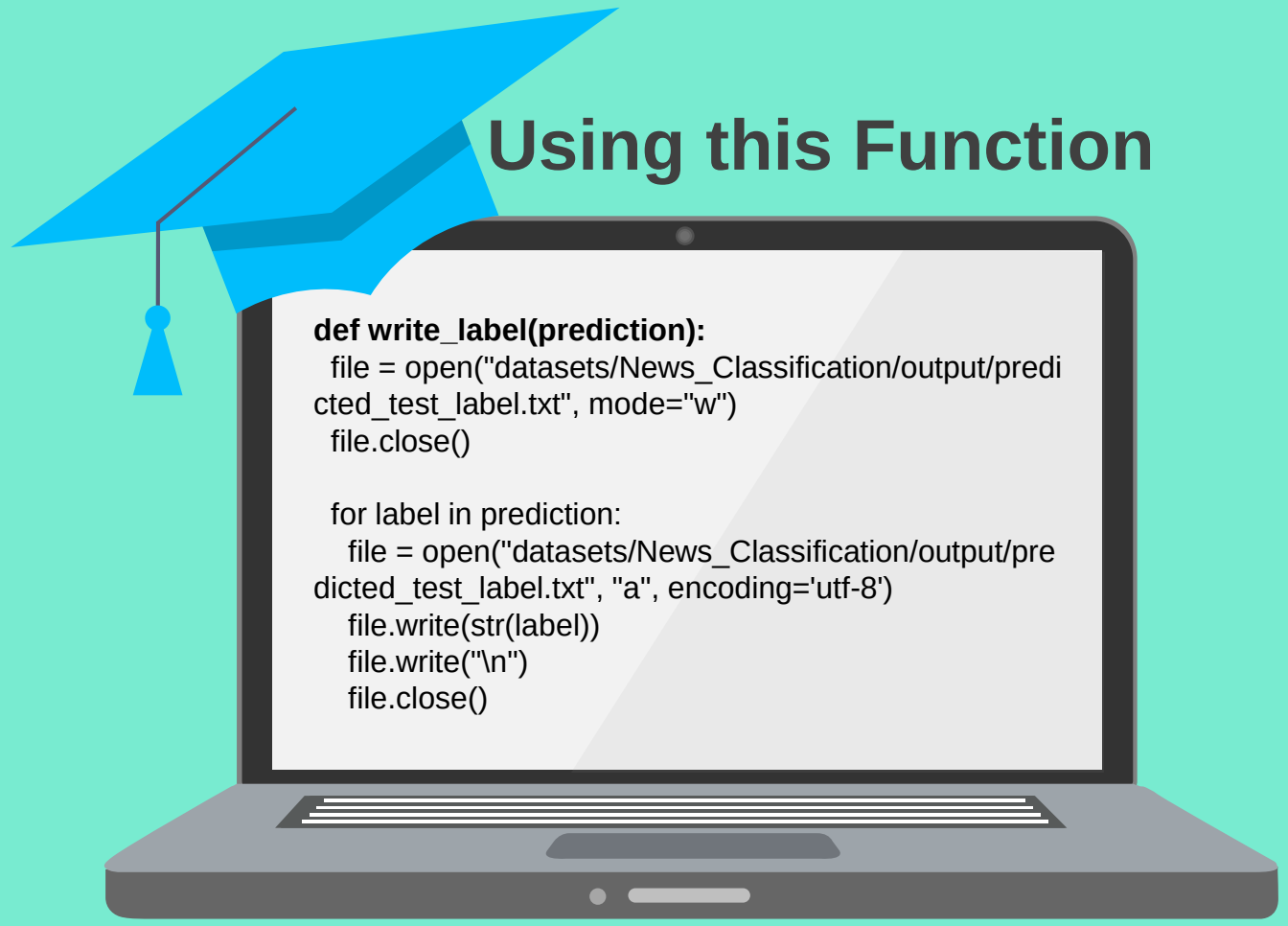
- **vec_tfidf(tfidf)**
  We use it for TfidfVectorizer and CountVectorizer.

System Output

- Predicted Label
- Best Naïve Bayes Accuracy
- Best SVM Accuracy

Get a Predicted Test Label

Using this Function



```
def write_label(prediction):
  file = open("datasets/News_Classification/output/predicted_test_label.txt", mode="w")
  file.close()

  for label in prediction:
    file = open("datasets/News_Classification/output/predicted_test_label.txt", "a", encoding='utf-8')
    file.write(str(label))
    file.write("\n")
    file.close()
```

Output

Which is save in our Google drive as predicted_test_label.txt.

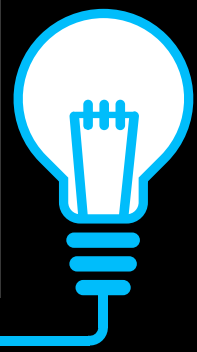
Naive Bayes Several Output

```
Reading The Dataset...

Training the Classifier...

Naive Bayes Accuracy =  0.9045652173913044

              precision    recall  f1-score   support

    Business       0.875     0.847     0.861      1132
    Sci_Tech       0.876     0.887     0.881      1158
      Sports       0.958     0.975     0.966      1182
       World       0.907     0.907     0.907      1128

    accuracy                           0.905      4600
   macro avg       0.904     0.904     0.904      4600
weighted avg       0.904     0.905     0.904      4600

Confusion Matrix :

 [[ 959  115   10   48]
 [  81 1027    8   42]
 [   9    6 1152   15]
 [  47   25   33 1023]]
```

```
Reading The Dataset...

Training the Naive_Bayes Classifier...

Naive Bayes Accuracy =  0.9060869565217391

              precision    recall  f1-score   support

    Business       0.880     0.852     0.865      1132
    Sci_Tech       0.877     0.889     0.883      1158
      Sports       0.959     0.974     0.966      1182
       World       0.906     0.907     0.907      1128

    accuracy                           0.906      4600
   macro avg       0.905     0.905     0.905      4600
weighted avg       0.906     0.906     0.906      4600

Confusion Matrix :

 [[ 964  114    9   45]
 [  77 1030    7   44]
 [   7    7 1151   17]
 [  48   24   33 1023]]
```


```
Reading The Dataset...

Training the Naive_Bayes Classifier ...

Naive Bayes Accuracy =  0.9108695652173913

              precision    recall  f1-score   support

    Business       0.877     0.872     0.875      1132
    Sci_Tech       0.889     0.888     0.888      1158
      Sports       0.960     0.975     0.968      1182
       World       0.915     0.906     0.910      1128

    accuracy                           0.911      4600
   macro avg       0.910     0.910     0.910      4600
weighted avg       0.911     0.911     0.911      4600

Confusion Matrix :

 [[ 987   99    7   39]
 [  83 1028    7   40]
 [   8    5 1153   16]
 [  47   25   34 1022]]
```

```
Reading The Dataset...

Training the Naive_Bayes Classifier ...

Naive Bayes Accuracy =  91.11 %
              precision    recall  f1-score   support

    Business       0.879     0.869     0.874      1132
    Sci_Tech       0.888     0.892     0.890      1158
      Sports       0.959     0.973     0.966      1182
       World       0.916     0.908     0.912      1128

    accuracy                           0.911      4600
   macro avg       0.910     0.911     0.910      4600
weighted avg       0.911     0.911     0.911      4600

Confusion Matrix :

 [[ 984  100    8   40]
 [  79 1033    8   38]
 [  11    5 1150   16]
 [  46   25   33 1024]]
```



Best Naive Bayes Classifier Report

	precision	recall	f1-score	support
Business	0.879	0.869	0.874	1132
Sci_Tech	0.888	0.892	0.890	1158
Sports	0.959	0.973	0.966	1182
World	0.916	0.908	0.912	1128
accuracy			0.911	4600
macro avg	0.910	0.911	0.910	4600
weighted avg	0.911	0.911	0.911	4600

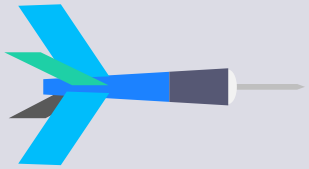
Best Naive Bayes Classifier Report

We Get Result

01
Reading The Dataset ...

02
Training the Naive Bayes Classifier ...

03
Naive Bayes Accuracy = 91.11 %



```
Confusion Matrix :

 [[ 984  100    8   40]
 [  79 1033    8   38]
 [  11    5 1150   16]
 [  46   25   33 1024]]
```

```
tfidf = False

'cls', MultinomialNB())

CountVectorizer(ngram_range =(9,9))
```

SVM Several Output

```
Reading The Dataset ...

Training the SVM Classifier ...
SVM Accuracy =  88.59 %
              precision    recall  f1-score   support

    Business       0.859     0.805     0.831      1132
    Sci_Tech       0.824     0.891     0.856      1158
      Sports       0.954     0.955     0.954      1182
       World       0.909     0.889     0.899      1128

    accuracy                           0.886      4600
   macro avg       0.886     0.885     0.885      4600
weighted avg       0.887     0.886     0.886      4600

Confusion Matrix :

 [[ 911  168   12   41]
 [  78 1032   12   36]
 [  14   16 1129   23]
 [  57   37   31 1003]]
```

```
Reading The Dataset ...

Training the SVM Classifier ...
SVM Accuracy =  88.74 %
              precision    recall  f1-score   support

    Business       0.861     0.802     0.830      1132
    Sci_Tech       0.821     0.893     0.856      1158
      Sports       0.957     0.962     0.959      1182
       World       0.913     0.889     0.901      1128

    accuracy                           0.887      4600
   macro avg       0.888     0.887     0.887      4600
weighted avg       0.888     0.887     0.887      4600

Confusion Matrix :

 [[ 908  173   11   40]
 [  79 1034   10   35]
 [  11   14 1137   20]
 [  57   38   30 1003]]
```

```
Reading The Dataset ...

Training the SVM Classifier ...
SVM Accuracy =  88.41 %
              precision    recall  f1-score   support

    Business       0.862     0.799     0.830      1132
    Sci_Tech       0.824     0.893     0.857      1158
      Sports       0.954     0.948     0.951      1182
       World       0.899     0.893     0.896      1128

    accuracy                           0.884      4600
   macro avg       0.885     0.883     0.883      4600
weighted avg       0.885     0.884     0.884      4600

Confusion Matrix :

 [[ 905  171   11   45]
 [  76 1034   11   37]
 [  15   15 1121   31]
 [  54   35   32 1007]]
```

```
Reading The Dataset ...

Training the SVM Classifier ...
SVM Accuracy =  88.52 %
              precision    recall  f1-score   support

    Business       0.856     0.801     0.828      1132
    Sci_Tech       0.819     0.893     0.855      1158
      Sports       0.957     0.955     0.956      1182
       World       0.912     0.888     0.900      1128

    accuracy                           0.885      4600
   macro avg       0.886     0.884     0.885      4600
weighted avg       0.886     0.885     0.885      4600

Confusion Matrix :

 [[ 907  173   12   40]
 [  80 1034    9   35]
 [  13   18 1129   22]
 [  59   37   30 1002]]
```

Best Support Vector machines Classifier Report

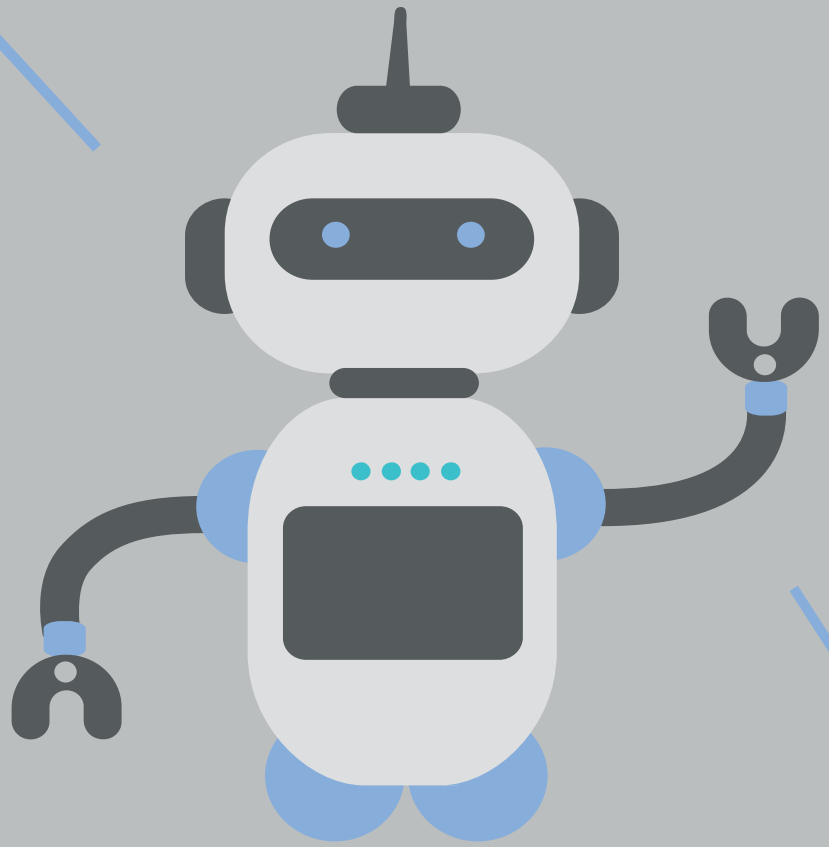
	precision	recall	f1-score	support
Business	0.861	0.802	0.830	1132
Sci_Tech	0.821	0.893	0.856	1158
Sports	0.957	0.962	0.959	1182
World	0.913	0.889	0.901	1128
accuracy			0.887	4600
macro avg	0.888	0.887	0.887	4600
weighted avg	0.888	0.887	0.887	4600

Best Support Vector machines Classifier Report

```
tfidf = True
SVC(kernel='linear', C=1.0))
TfidfVectorizer(ngram_range = (4,9))
```

Confusion Matrix

```
Confusion Matrix :

 [[ 908  173   11   40]
 [  79 1034   10   35]
 [  11   14 1137   20]
 [  57   38   30 1003]]
```



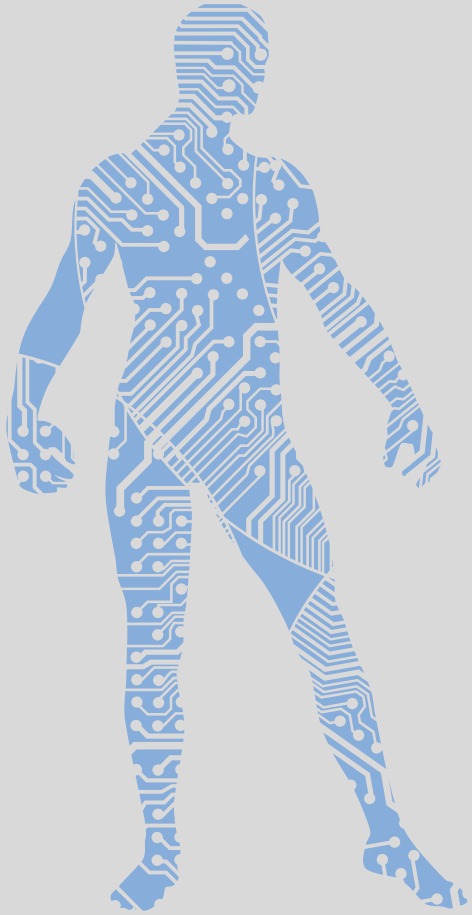
Training the SVM Classifier ...

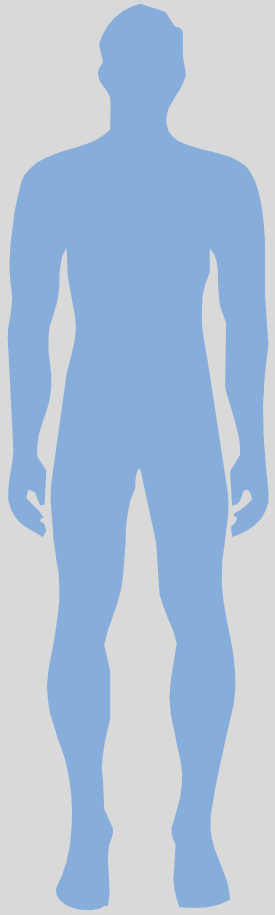
SVM Accuracy = 88.74 %.

Final Result

Which Classifier is Best for Our Model

Naive Bayes Accuracy = 91.11 %





SVM Accuracy = 88.74 %

Conclusion

In this project, Firstly the News Classification datasets are tokenize and separate the labels and documents with a function call.

Then, different pre-processing method to find out different function. The train.txt data is big so it take more and more time to get output. So, we use 15000 train data for SVM. Changing ngram_range to get a best result.
Finally, this project has been run successfully and performance of the system has been found satisfactory.

News Classification

Thank You