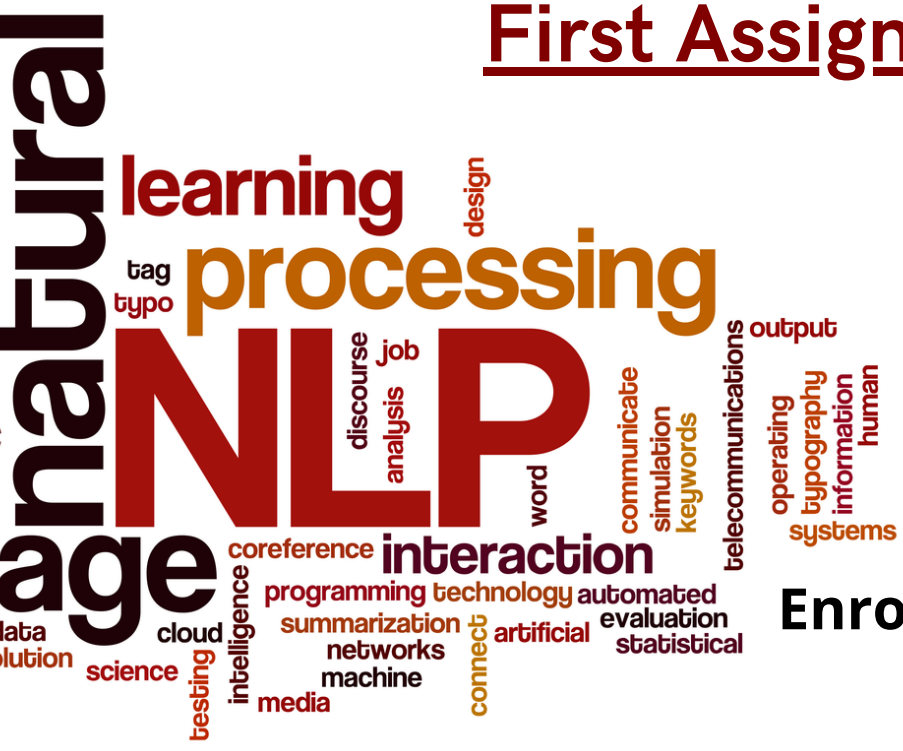
School of Data Science and Forecasting

Devi Ahilya Vishwavidyalaya

Session 2022-2023

Natural Language Processing

First Assignment



Submitted By :
Piyush Joshi

Submitted To :
Prof. Avinash Navlani

Batch : MSc DSA (3 Sem)

Roll No : DS5B - 2121

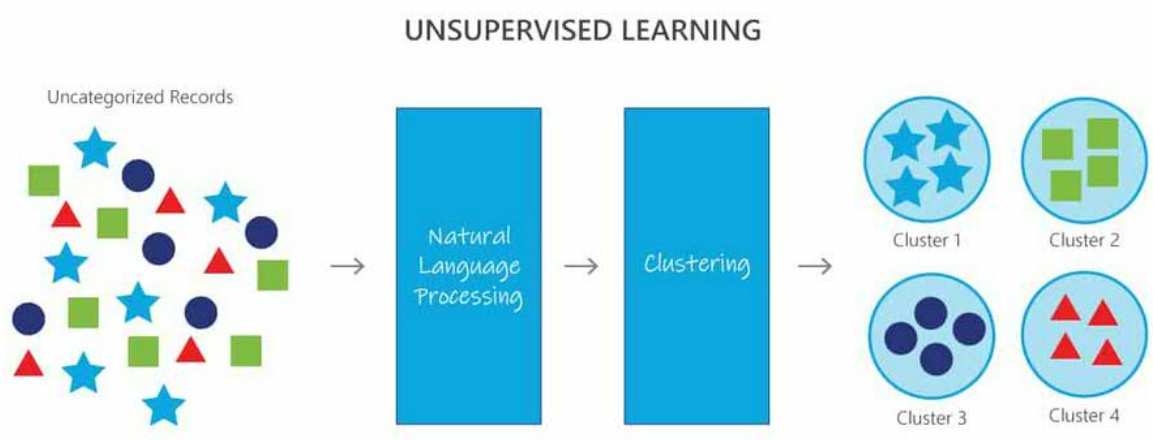
Enrollment No. : DS1821110

Submission Date :
06/10/2022

Text Clustering

Table of Contents

Introduction to Text Clustering



Text Clustering is a process of grouping most similar articles, tweets, reviews, and documents together. Clustering is also known as the **data segmentation method.** It divides the data points into a number of specific batches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense. Every group is known as a **cluster.** It is an **unsupervised learning** technique which implies it discovers hidden patterns or data groupings within the data without the need for human intervention.

Applications of Clustering

Clustering can be utilized in:

- Outlier detection problems such as fraud detection
- Clustering or organizing documents
- Text summarization.
- Customer Segmentation
- Recommender System
- Visualization

Types of Clustering

There are various clustering techniques based on differential evolution such as:

- **K-Means** based on the distance between points
- **Affinity propagation** based on graph distance
- **DBSCAN** based on the distance between the nearest points
- **Spectral clustering** based on graph distance
- **Gaussian Mixtures** based on Mahalanobis distance to centers
- **Hierarchical clustering** based on hierarchies

Working of Clustering techniques

Fundamentally, all clustering methods use the same approach i.e.

> First we calculate similarities, and then we use it to cluster the data points into groups or batches. Some techniques use distances while others use statistical distributions to compute similarities.

These techniques require text to be converted into some type of vectors by techniques such as

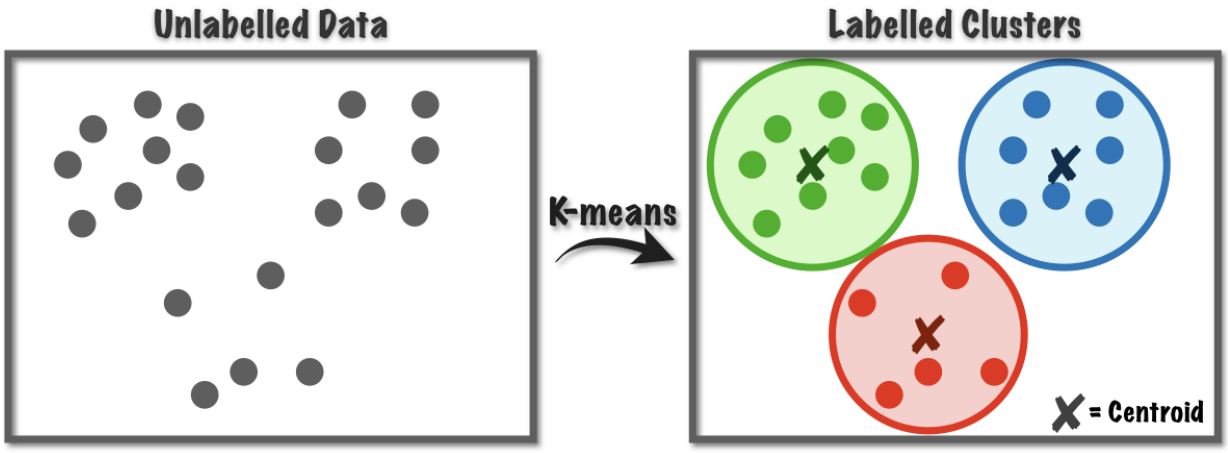
- Bag of Words(BoW)
- Term Frequency-Inverse Document Frequency (TF-IDF)
- Word2Vec
- Doc2Vec
- Sent2Vec, etc.

Let’s explore some of the clustering techniques in more detail

Various Clustering Techniques

K-Means Clustering

K-means is one of the simplest and most widely used clustering algorithms. It is a type of **partitioning clustering method** that partitions the dataset into random segments. K-means is a faster and more robust algorithm that generates spherical clusters.



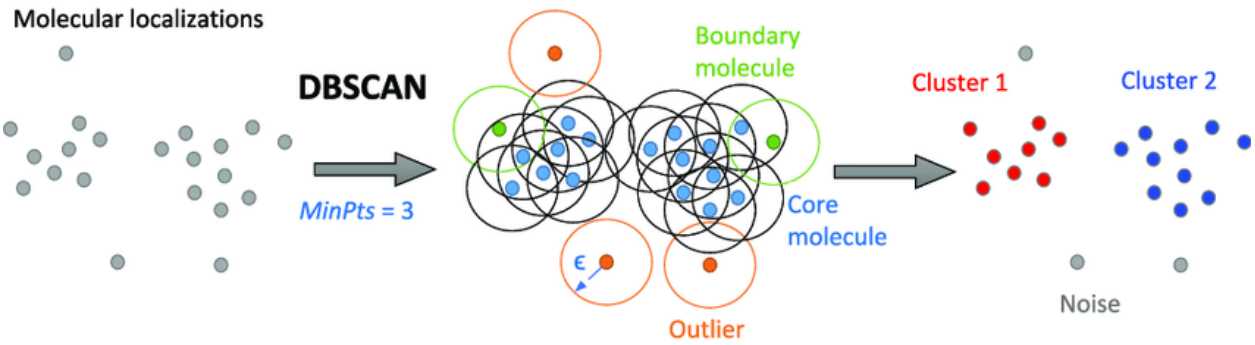
The k-means method requires the **number of clusters** as input at the beginning and it does not guarantee convergence to the global solution, rather its result may depend upon the initial cluster center. The k-means method is not suitable for finding **non-convex clusters** and **nominal attributes.**

The K-Means Clustering **groups similar data points** together and **discovers underlying patterns**. To achieve this objective, K-means looks for a fixed number **(k)** of clusters in a dataset. The K-means algorithm identifies the k number of **centroids**, and then allocates every data point to the nearest cluster.

The **'means'** in the K-means refers to averaging the data; that is, finding the centroid.

DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a base algorithm for **density-based clustering.** It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.

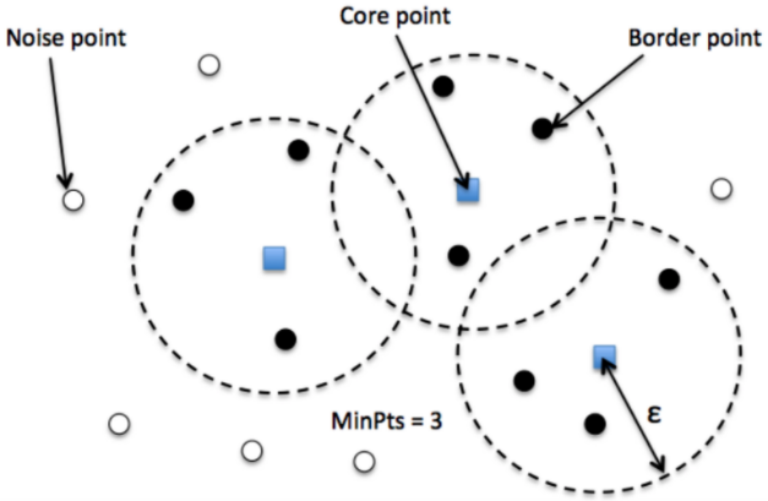


The DBSCAN algorithm uses two parameters:

- **minPts:** The minimum number of points (a threshold) clustered together for a region to be considered dense.
- **eps (ε):** A distance measure that will be used to locate the points in the neighbourhood of any point.

These parameters can be understood better if we explore two concepts called **Density Reachability** and **Density Connectivity.**

- **Reachability** in terms of density establishes a point to be reachable from another if it lies within a particular distance (eps) from it.
- **Connectivity**, on the other hand, involves a transitivity based chaining-approach to determine whether points are located in a particular cluster. For example, p and q points could be connected if p->r->s->t->q, where a->b means b is in the neighborhood of a.

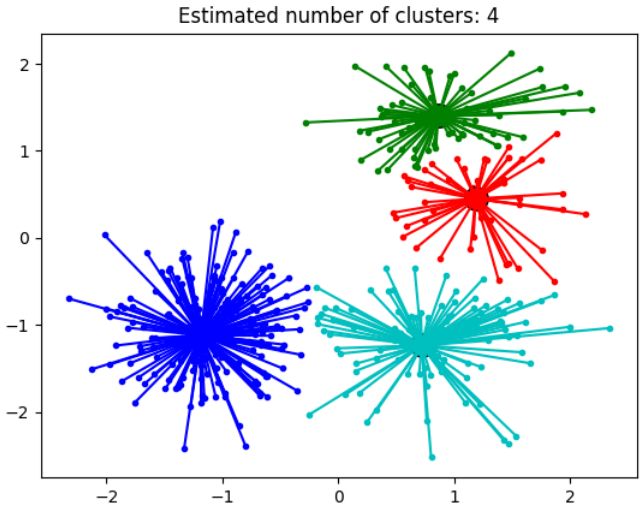


There are **three** types of points after the DBSCAN clustering is complete:

- **Core** — This is a point that has at least *m* points within distance *n* from itself.
- **Border** — This is a point that has at least one Core point at a distance *n*.
- **Noise** — This is a point that is neither a Core nor a Border. And it has less than *m* points within distance *n* from itself.

Affinity Propagation

Affinity Propagation was first published in 2007 by **Brendan Frey** and **Delbert Dueck**. In contrast to other traditional clustering methods, Affinity Propagation **does not require** you to specify the number of clusters. In layman's terms, in Affinity Propagation, each data point sends messages to all other points informing its targets of each target's relative attractiveness to the sender. Each target then responds to all senders with a reply informing each sender of its availability to associate with the sender, given the attractiveness of the messages that it has received from all other senders. Senders reply to the targets with messages informing each target of the target's revised relative attractiveness to the sender, given the availability messages it has received from all targets.



The message-passing procedure proceeds until a consensus is reached. Once the sender is associated with one of its targets, that target becomes the point's exemplar. All points with the same exemplar are placed in the same cluster.

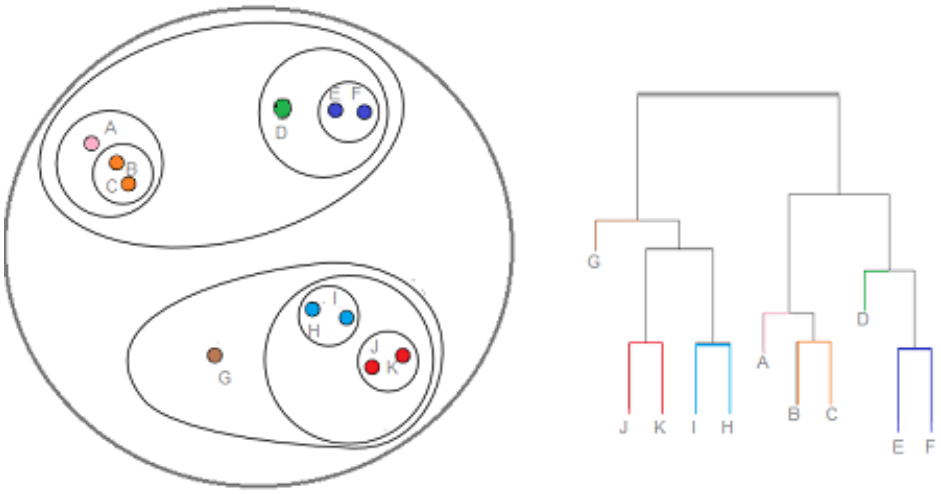
Affinity Propagation creates clusters by sending messages between data points until convergence. Unlike clustering algorithms such as **k-means** or **k-medoids**, affinity propagation does not require the number of clusters to be determined or estimated before running the algorithm, for this purpose the two important parameters are

- **Preference**, which controls how many exemplars (or prototypes) are used
- **Damping factor** which damps the responsibility and availability of messages to avoid numerical oscillations when updating messages.

A dataset is described using a small number of exemplars, **'exemplars'** are members of the input set that are representative of clusters. The messages sent between pairs represent the suitability for one sample to be the exemplar of the other, which is updated in response to the values from other pairs. This updating happens iteratively until convergence, at that point the final exemplars are chosen, and hence we obtain the final clustering.

Hierarchical Clustering

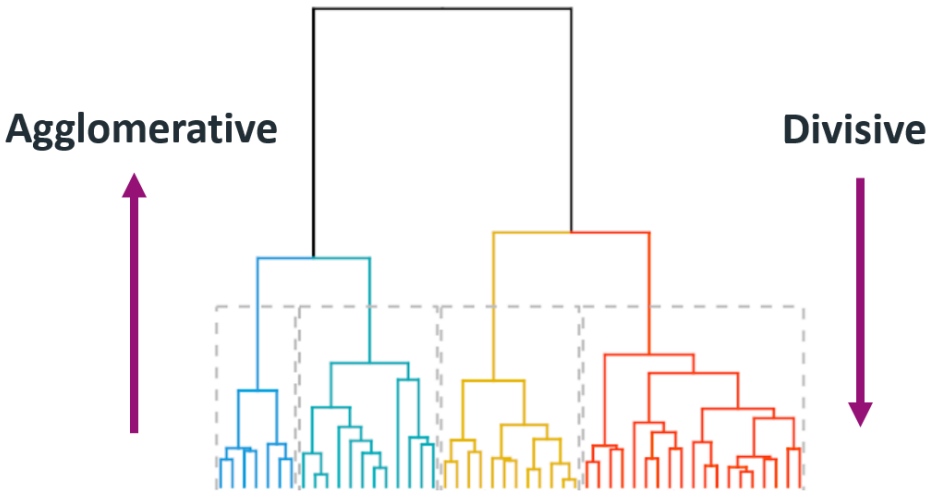
Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or **HCA**.



In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

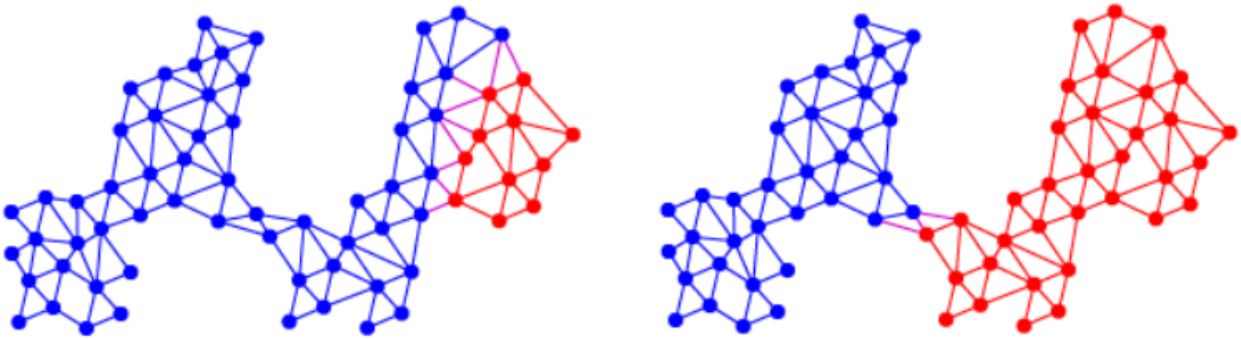
The hierarchical clustering technique has two approaches:



- **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left. These algorithms treat each document as a singleton cluster at the outset and then successively merge (or *agglomerate*) pairs of clusters until all clusters have been merged into a single cluster that contains all documents.
- **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach.** Top-down clustering requires a method for splitting a cluster. It proceeds by splitting clusters recursively until individual documents are reached.

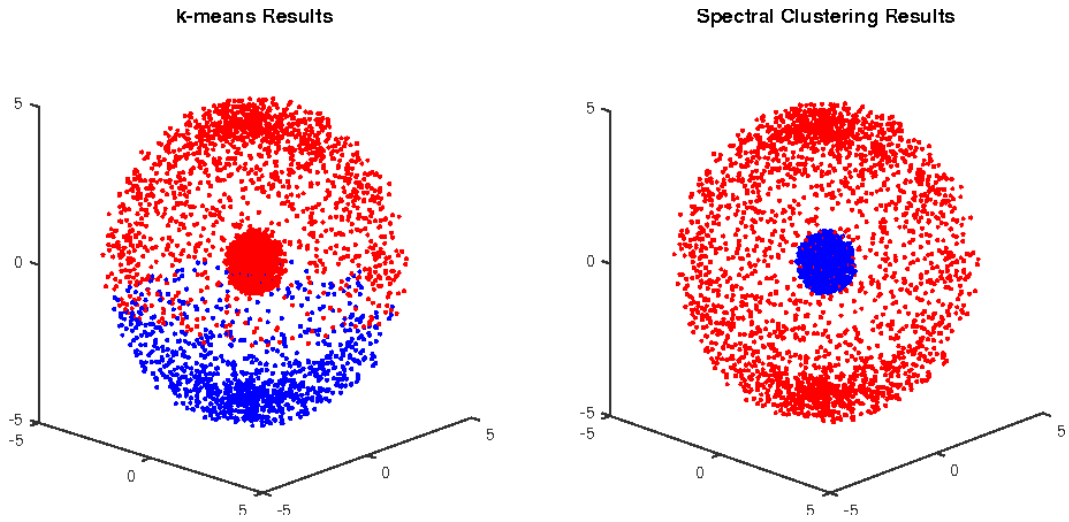
Spectral Clustering

Spectral Clustering is a growing clustering algorithm which has performed better than many traditional clustering algorithms in many cases. It treats each data point as a **graph-node** and thus transforms the clustering problem into a **graph-partitioning problem.**



Properties:

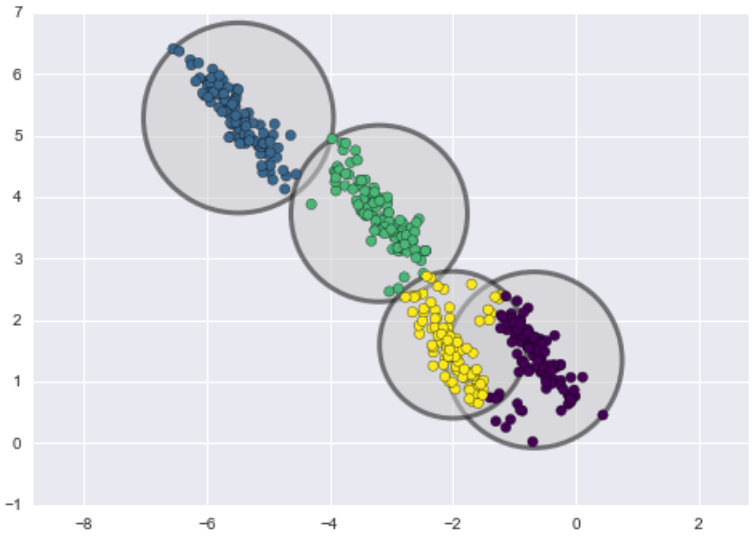
1. **Assumption-Less:** This clustering technique, unlike other traditional techniques do not assume the data to follow some property. Thus this makes this technique to answer a more-generic class of clustering problems.
2. **Ease of implementation and Speed:** This algorithm is easier to implement than other clustering algorithms and is also very fast as it mainly consists of mathematical computations.
3. **Not-Scalable:** Since it involves the building of matrices and computation of eigenvalues and eigenvectors it is time-consuming for dense datasets.



Gaussian Mixture Models (GMMs) Clustering

Gaussian Mixture Models (GMMs) assume that there are a certain number of **Gaussian distributions**, and each of these distributions represent a cluster. Hence, a

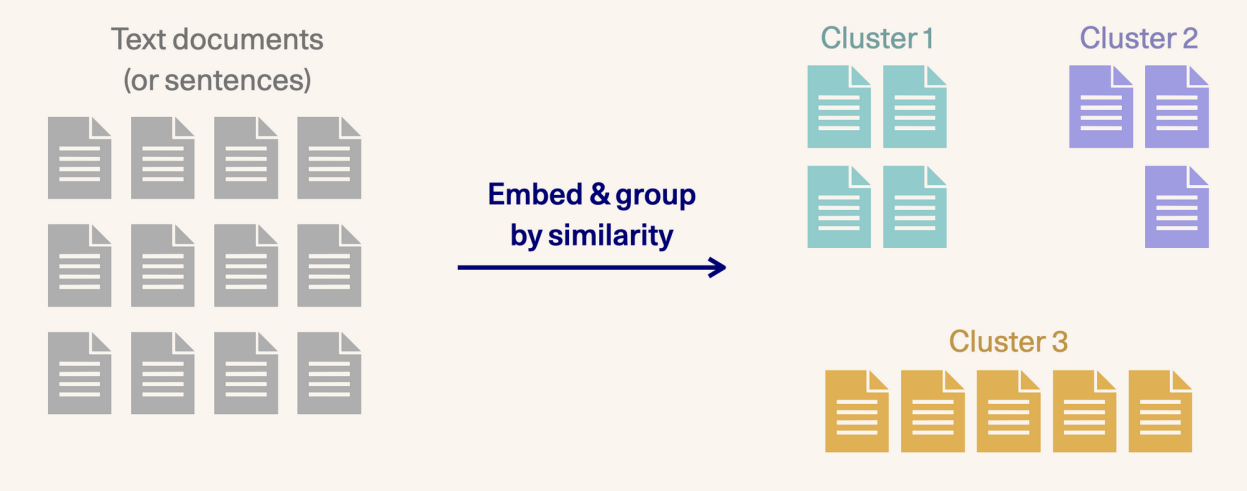
Gaussian Mixture Model tends to group the data points belonging to a single distribution together.



Gaussian Mixture Models are **probabilistic models** and use the **soft clustering approach** for distributing the points in different clusters. Each GMM would identify the probability of each data point belonging to a Gaussian distributions.

Use Case : Applying K-Means Clustering on articles from Machinelearninggeek.com

Overview



Document clustering (or **text clustering**) is the application of cluster analysis to textual documents. It has applications in automatic document organization, topic extraction and fast information retrieval or filtering.

Problem Statement

To scrap textual articles from Machine Learning Geek and create an NLP model to cluster those articles using K-Means Clustering

Libraries and Packages Used

- `pandas`
- `numpy`
- `re`
- `sklearn`
- `BeautifulSoup`
- `requests`

- `nltk`
- `wordcloud`
- `matplotlib`

Procedure

To create an NLP Model using scrapped articles, we need to follow the following steps:

- **Step 1 : Data Acquisition** - Acquire textual articles from https://machinelearninggeek.com/ using Python's `request` library and create a dataset
- **Step 2 : Text Preprocessing** - Preprocessing will involve steps such as
  - `tokenization`
  - `removing HTML Tags`
  - `removing URLs`
  - `removing stopwords`
  - `stemming`
  - `lemmatization`
- **Step 3 : Feature Engineering** - This step will involve creating vectors from text by vectorization techniques such as
  - `Bag of Words`
  - `TF-IDF(Term Frequency Inverse Docuement Frequency)`
- **Step 4 : Modelling** - This step will involve modelling using `K-Means Clustering`
- **Step 5 : Evaluation** - Evaluating the model using
  - `Silhouette score`
  - `Davies Bouldin score`

Once the model is ready we can use it to cluster the scrapped articles

Text Clustering on Machinelearninggeek.com Data

Applied `Tf-Idf Vectorizer` and `K-Means Algorithm` to better segment the articles fetched using `requests` library from [Machine Learning Geek]

- Submitted By : By Piyush Joshi
- Roll Number : DS5B-2121
- Submitted To : Prof. Avinash Nalvani
- Subject : Natural Language Processing
- Batch : MSc 3rd Sem (Data Science and Analytics)
- College : School of Data Science and Forecasting, DAVV, Indore

Building a NLP Pipeline for Text Clustering

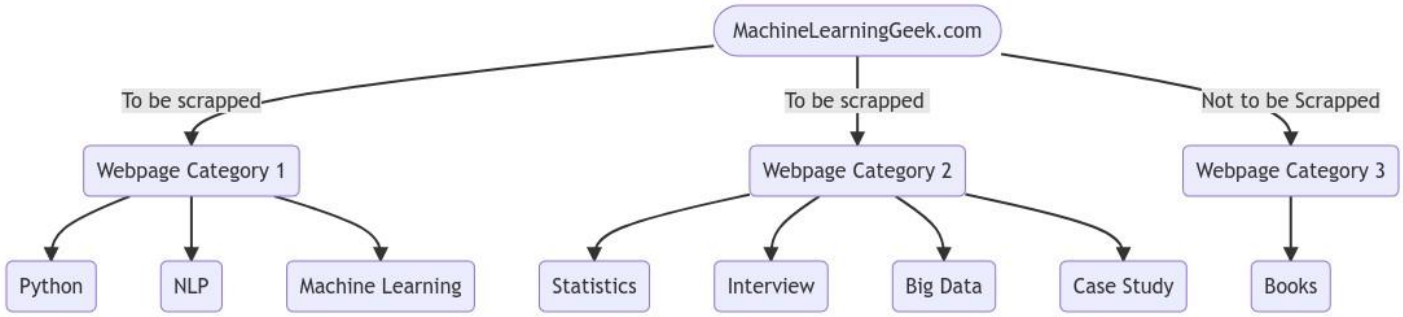
Step 1 : Data Acquisition

- In the first step we will get the data from website https://machinelearninggeek.com/ using the `requests` library and scrap it using the `BeautifulSoup` Library
- Once we are done with this step we will use `pandas` to convert it into a csv file

```
import requests
from bs4 import BeautifulSoup as bs
import re
import pandas as pd
```

Structure of the website

- In order to scrap the website we first need to understand the Document Object Model (DOM) of the website
- The website is composed of three category of pages which differ in their structure, link syntax and the way information is displayed on them
- The three category are being displayed in the diagram below out of which only two contains text which is to be scrapped



Finding the title and list of pages of various category

```
init_request = requests.get('https://machinelearninggeek.com/')
webpages = []
soup = bs(init_request.content,'html.parser')
header = soup.find('div',class_="menu-primary-container")
nav_bar = header.find('ul')
nav_bar_links = nav_bar.find_all('a')
for link in nav_bar_links:
  webpages.append([link.text,link.get('href')])
webpage_urls = webpages[0:7]
webpage_urls
```

```
[['Machine Learning', 'https://machinelearninggeek.com/machine-learning/'],
 ['NLP', 'https://machinelearninggeek.com/nlp/'],
 ['Statistics', 'https://machinelearninggeek.com/category/statistics/'],
 ['Interview', 'https://machinelearninggeek.com/category/interview/'],
 ['Python', 'https://machinelearninggeek.com/python/'],
 ['Big Data', 'https://machinelearninggeek.com/category/big-data/'],
 ['Case Studies',
  'https://machinelearninggeek.com/category/business-analytics/']]
```

Fetching Articles and Titles

```
def fetch_data(link):
    article = []
    r = requests.get(link)
    soup = bs(r.content,'html.parser')
    # r2 = soup.find('header',class_="entry-header")
    # r2 = re.sub('\n','',r2.text)
    r1 = soup.find('div',class_="entry-content clearfix")
    paras = r1.find_all('p')
    for para in paras:
      article.append(para.text)
    return " ".join(article)

def fetch_header(link):
```

```
  r = requests.get(link)
  soup = bs(r.content,'html.parser')
  r2 = soup.find('header',class_="entry-header")
  r2 = re.sub('\n','',r2.text)
  return r2

lst = []
for webpage in range(len(webpage_urls)):
  r = requests.get(webpage_urls[webpage][1])
  soup = bs(r.content,'html.parser')
  if (webpage==0)or(webpage==1)or(webpage==4):
    s = soup.find('div',class_="entry-content clearfix")
    unor = s.find_all('ul')
    for u in unor:
      links = u.find_all('a')
      for link in links:
        try:
          lst.append([webpage_urls[webpage][0],fetch_header(link.get('href')),fetch_data(link
        except:
          continue
  else:
    s = soup.find_all('div',class_="entry-content clearfix")
    for t in s:
      links = t.find_all('a')
      for link in links:
        try:
          lst.append([webpage_urls[webpage][0],fetch_header(link.get('href')),fetch_data(link
        except:
          continue
```

```
len(lst)
```

```
105
```

```
lst[1]
```

```
['Machine Learning',
 'Activation Functions ',
 'The activation function defines the output of a neuron in terms of the induced
local field. Activation functions are a single line of code that gives the neural
networks non-linearity and expressiveness.\xa0There are many activation functions
such as Identity function, Step function, Sigmoid function, Tanh, ReLU, Leaky ReLU,
Parametric ReLU, and Softmax function. We can see some of them in the following
table: In this tutorial, we are going to cover the following topics: The identity
function is a function that maps input to the same output value. It is a linear
operator in vector space. Also, a known straight-line function where activation is
proportional to the input. The simplest example of a linear activation function is a
linear equation. \xa0f(x) = a * x,where a ∈ R The major problem with such kind of
linear function it cannot handle complex scenarios.\xa0 In Binary Step Function, if
the value of Y is above a certain value known as the threshold, the output is True(or
activated) and if it’s less than the threshold then the output is false (or not
```

activated). It is very useful in the classifier. The main problem with the binary step function is zero gradients or it is not differentiable at zero. It cannot update the gradient in backpropagation. It only works with binary class problems because it maps to only two categories 0 and 1. In the Bipolar Step Function, if the value of Y is above a certain value known as the threshold, the output is +1and if it’s less than the threshold then the output is -1. It has bipolar outputs (+1 to -1). It can be utilized in single-layer networks.\xa0 It is also called S-shaped functions. Logistic and hyperbolic tangent functions are commonly used in sigmoid functions. There are two types of sigmoid functions. Binary Sigmoid Function or Sigmoid function is a logistic function where the output values are either binary or vary from 0 to 1. It is differentiable, non-linear, and produces non-binary activations But the problem with Sigmoid is the vanishing gradients. Also, sigmoid activation is not a zero-centric function. Hyperbolic Tangent Function or Tanh is a logistic function where the output value varies from -1 to 1. Also known as Bipolar Sigmoid Function. The output of Tanh centers around 0 and sigmoid’s around 0.5. Tanh Convergence is usually faster if the average of each input variable over the training set is close to zero. When you struggle to quickly find the local or global minimum, in such case Tanh can be helpful in faster convergence. The derivatives of Tanh are larger than Sigmoid that causes faster optimization of the cost function. Tanh suffered from vanishing gradient problems. ReLu stands for the rectified linear unit (ReLU). It is the most used activation function in the world. It output 0 for negative values of x. This is also known as a ramp function. The name of the ramp function is derived from the appearance of its graph. ReLu(Rectified Linear Unit) is like a linearity switch. If you don’t need it, you “switch” it off. If you need it, you “switch” it on. ReLu avoids the problem of vanishing gradient. ReLu also provides the benefit of sparsity and sigmoids result in dense representations. Sparse representations are more useful than dense representations. The main problem with ReLU is, it is not differentiable at 0 and may result in exploding gradients.\xa0 The main problem of ReLU is, it is not differentiable at 0 and may result in exploding gradients. To resolve this problem Leaky ReLu was introduced that is differentiable at 0. It provides small negative values when input is less than 0. The main problem with Leaky ReLu is not offering consistent predictions in terms of negative data. PReLU (Parametric ReLU) overcome the dying ReLU problem and Leaky ReLU inconsistent predictions for negative input values. The core idea behind the Parametric ReLU is to make the coefficient of leakage into a parameter that gets learned. \xa0 The softmax function is typically used on the output layer for multi-class classification problems. It provides the probability distribution of possible outcomes of the network. In conclusion, we can say in deep learning problems, ReLu is used on hidden layers and sigmoid/softmax on the output layer. Sigmoid is used for\xa0binary classification, and Softmax is used for\xa0multi-class classification problems. In this tutorial, we have discussed various activation functions, types of activation functions such as Identity function, Step function, Sigmoid function, Tanh, ReLU, Leaky ReLU, Parametric ReLU, and Softmax function. We have discussed the pros and cons of various activation

Exporting the Dataset

```
import pandas as pd
dataset = pd.DataFrame(lst,columns=['Category','Title','Article'])
dataset.head()
```

	Category	Title	Article
0	Machine Learning	Introduction to Artificial Neural Network	This is an introductory article for the artifi...
1	Machine Learning	Activation Functions	The activation function defines the output of ...

```
dataset.to_csv('21_Text_Clustering_Piyush_Joshi') # Dataset saved to Google Colab's Runtime M
```

Load the Dataset

```
# Load the Data
df = dataset
```

```
df.head()
```

	Category	Title	Article
0	Machine Learning	Introduction to Artificial Neural Network	This is an introductory article for the artifi...
1	Machine Learning	Activation Functions	The activation function defines the output of ...
2	Machine Learning	Multi-Layer Perceptron Neural Network using Py...	In this tutorial, we will focus on the multi-l...
3	Machine	Backpropagation Neural Network	Backpropagation neural network is used

Step 2 : Data Exploration and Pre-Processing

Data Exploration

```
df2 = df.groupby(['Category'])['Category'].count()
df2
```

```
Category
Big Data             9
Case Studies         7
Interview           10
Machine Learning    30
NLP                 16
Python              29
Statistics           4
Name: Category, dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105 entries, 0 to 104
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  105 non-null    object
 1   Title     105 non-null    object
 2   Article   105 non-null    object
dtypes: object(3)
memory usage: 2.6+ KB
```

Label Encoding the Category Column

- Encode the Categorical Column Category using the `preprocessing` class from **sklearn**
- Using this a new column by the name `Category_new` will be created that contains the encoded version of the category

```
from sklearn import preprocessing
obj_df = df.select_dtypes(include = ['object']).copy()
# these are categorical columns which we want to create a dummy variable
print(obj_df.head())
```

```
             Category                                                Title  \
0  Machine Learning         Introduction to Artificial Neural Network      
1  Machine Learning                                Activation Functions    
2  Machine Learning  Multi-Layer Perceptron Neural Network using Py...
3  Machine Learning       Backpropagation Neural Network using Python    
4  Machine Learning  Understanding Logistic Regression and Building...

                                          Article
0  This is an introductory article for the artifi...
1  The activation function defines the output of ...
2  In this tutorial, we will focus on the multi-l...
3  Backpropagation neural network is used to impr...
4  Learn about Logistic Regression, its basic pro...
```

```
lb_make = preprocessing.LabelEncoder()
df[f"Category_new"] = lb_make.fit_transform(df[f"Category"])
```

```
df.sample(5)
```

	Category	Title	Article	Category_new
13	Machine	Introduction to Ensemble	Ensemble Techniques are	2
24	Big Data	Apache Hive Hands-On	In this tutorial, we will focus on	2

Corpus Analysis

A corpus is the compilation of all the text under consideration. In this case its the aggregation of all the text in the column **'Article'**

```
text = str(' '.join(df.Article.tolist()))
```

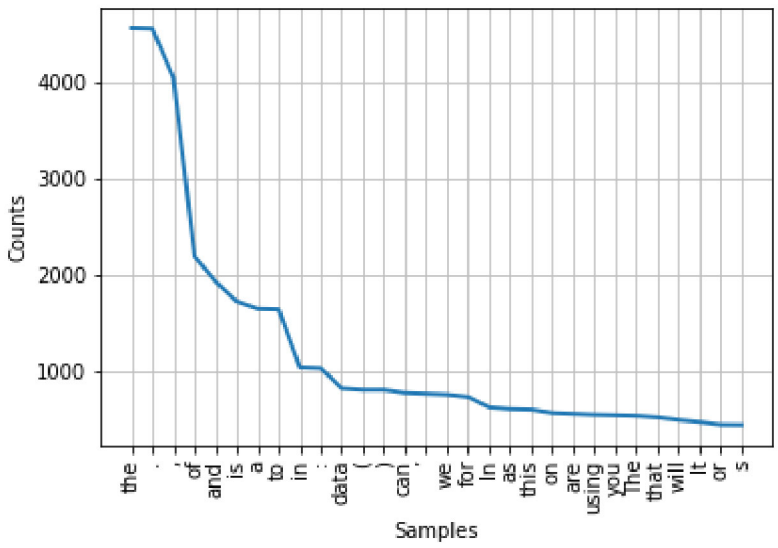
```
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
from nltk.probability import FreqDist
fd = FreqDist(word_tokenize(text))
fd.most_common(8)
```

```
[('the', 4559),
 ('.', 4553),
 (',', 4044),
 ('of', 2192),
 ('and', 1931),
 ('is', 1725),
 ('a', 1648),
 ('to', 1642)]
```

```
fd.plot(30,cumulative = False)
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc44e29f10>
```

As we can clearly observe from the corpus analysis that the majority of the words within it are stop words (that is the, of, a, is etc.) and punctuations(,:) which do not contribute to the meaning of the sentence hence we need to remove them

Text Normalization

We will create a new column in the dataframe by the name `Cleaned_Article` and perform all the pre-processing operations there

```
def lower_text(text):
  return text.lower()
df['Cleaned_Article'] = df['Article'].apply(lower_text)
```

```
df.head()
```

	Category	Title	Article	Category_new	Cleaned_Article
0	Machine Learning	Introduction to Artificial Neural Network	This is an introductory article for the artifi...	3	this is an introductory article for the artifi...
1	Machine Learning	Activation Functions	The activation function defines the output of ...	3	the activation function defines the output of ...
2	Machine	Multi-Layer Perceptron Neural Network using	In this tutorial, we will	3	in this tutorial, we will focus on the

Remove URL

As the data is web scrapped there is always a possibility of presence of URLs which needs to be removed using **Regular Expressions**

```
def url_remover(lst):
  URLless_string = re.sub(r'\w+:\/{2}[\d\w-]+(\.[\d\w-]+)*(?:(?:\/[^\s/]*))*', '', lst)
  return URLless_string
df['Cleaned_Article'] = df['Cleaned_Article'].apply(url_remover)
df['Cleaned_Article'][1:5]
```

```
1    the activation function defines the output of ...
2    in this tutorial, we will focus on the multi-l...
3    backpropagation neural network is used to impr...
4    learn about logistic regression, its basic pro...
Name: Cleaned_Article, dtype: object
```

Remove HTML Tags

The below **regular expressions** helps to remove the HTML Tags within the text

```
CLEANR = re.compile('<.*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')

def cleanhtml(raw_html):
  cleantext = re.sub(CLEANR, '', raw_html)
  return cleantext
df['Cleaned_Article']=df['Cleaned_Article'].apply(cleanhtml)
df['Cleaned_Article'][1:5]
```

```
1    the activation function defines the output of ...
2    in this tutorial, we will focus on the multi-l...
3    backpropagation neural network is used to impr...
4    learn about logistic regression, its basic pro...
Name: Cleaned_Article, dtype: object
```

Tokenization

Sentence Tokenization

Here we will tokenize each of the document (or column within dataframe) into specific sentences separated by full stops. Here we will use **nltk**

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
from nltk.tokenize import sent_tokenize
def sen_token(text):
  return sent_tokenize(text)
df['Cleaned_Article'].apply(sen_token)
```

```
0      [this is an introductory article for the artif...
1      [the activation function defines the output of...
2      [in this tutorial, we will focus on the multi-...
3      [backpropagation neural network is used to imp...
4      [learn about logistic regression, its basic pr...
                            ...
100    [in this world of the internet, information is...
101    [in this python tutorial, explore movie data o...
102    [learn how to calculate customer life time val...
103    [in this tutorial, you’re going to learn how t...
104    [analyze employee churn, why employees are lea...
Name: Cleaned_Article, Length: 105, dtype: object
```

Word Tokenization

Here we will tokenize the text into separate words listings

```
from nltk.tokenize import word_tokenize
def wrd_token(text):
  return word_tokenize(text)
df['Cleaned_Article'] = df['Cleaned_Article'].apply(wrd_token)
df['Cleaned_Article'][1:5]
```

```
1    [the, activation, function, defines, the, outp...
2    [in, this, tutorial, ,, we, will, focus, on, t...
3    [backpropagation, neural, network, is, used, t...
4    [learn, about, logistic, regression, ,, its, b...
Name: Cleaned_Article, dtype: object
```

Remove Stop Words

In the word it can be clearly observed that the stop words needs to be removed from the text

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
```

```
import string
punctuations = list(string.punctuation)
print(punctuations)
```

```
['!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<
```

```
other_stop_words = ["'s",'’','‘','“','”']
```

```
df['Cleaned_Article'] = df['Cleaned_Article'].apply(lambda x: [item for item in x if item not
df['Cleaned_Article'][1:5]
```

```
1    [activation, function, defines, output, neuron...
2    [tutorial, focus, multi-layer, perceptron, wor...
3    [backpropagation, neural, network, used, impro...
4    [learn, logistic, regression, basic, propertie...
Name: Cleaned_Article, dtype: object
```

Stemming

Now we will reduce words into their stem using SnowStemmer

```
from nltk.stem import SnowballStemmer
stemmed_words = []
sb = SnowballStemmer("english")
```

```
df['Cleaned_Article'].apply(lambda x: [sb.stem(y) for y in x])[1:5]
```

```
1    [activ, function, defin, output, neuron, term,...
2    [tutori, focus, multi-lay, perceptron, work, h...
3    [backpropag, neural, network, use, improv, acc...
4    [learn, logist, regress, basic, properti, work...
Name: Cleaned_Article, dtype: object
```

Lemmatization

Here we will reduce each word to its stem word that exist within the language

```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
import nltk
nltk.download('omw-1.4')
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

```
def lemmatize(s):
    s = [lem.lemmatize(word) for word in s]
    return s
df['Cleaned_Article']=df['Cleaned_Article'].apply(lambda x: lemmatize(x))
df['Cleaned_Article'][1:5]
```

```
1    [activation, function, defines, output, neuron...
2    [tutorial, focus, multi-layer, perceptron, wor...
```

```
3    [backpropagation, neural, network, used, impro...
4    [learn, logistic, regression, basic, property,...
Name: Cleaned_Article, dtype: object
```

Step 3 : Feature Engineering

Here we will create features for the machine learning model, but lets first explore each of the category of articles and try to infere what the feature could be. The larger a word appears in a wordcloud more it will represent each of the category

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

def word_cloud(title_, text):
    """ Create WorldCloud """
    stopword_list = set(STOPWORDS)

    wordcloud = WordCloud(width = 400, height = 200,
                    background_color ='white',
                    stopwords = stopword_list,
                    min_font_size = 10).generate(text)

    # plot the WordCloud image
    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)
    plt.title(title_, fontsize=20)
    plt.show()
```

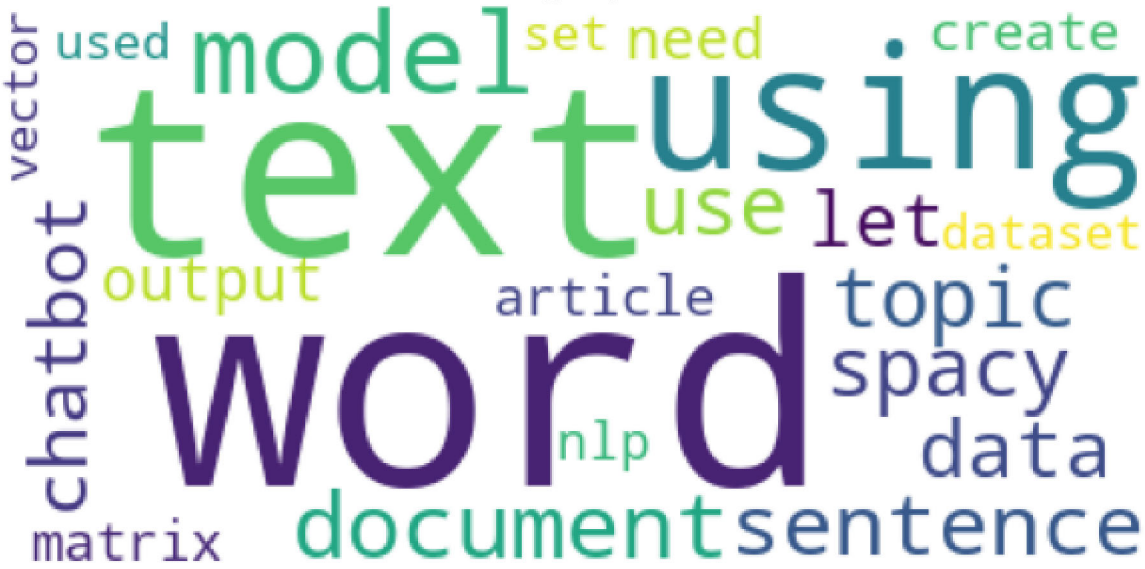
```
def joiner(txt):
  return " ".join(txt)
df['Cleaned_Text'] = df['Cleaned_Article'].apply(joiner)
```

```
for i in df.Category.unique():
  paragraph=' '.join(df[df.Category==i].Cleaned_Text.tolist())
  print(word_cloud(f"{i} Category Articles", paragraph))
```

Machine Learning Category Articles

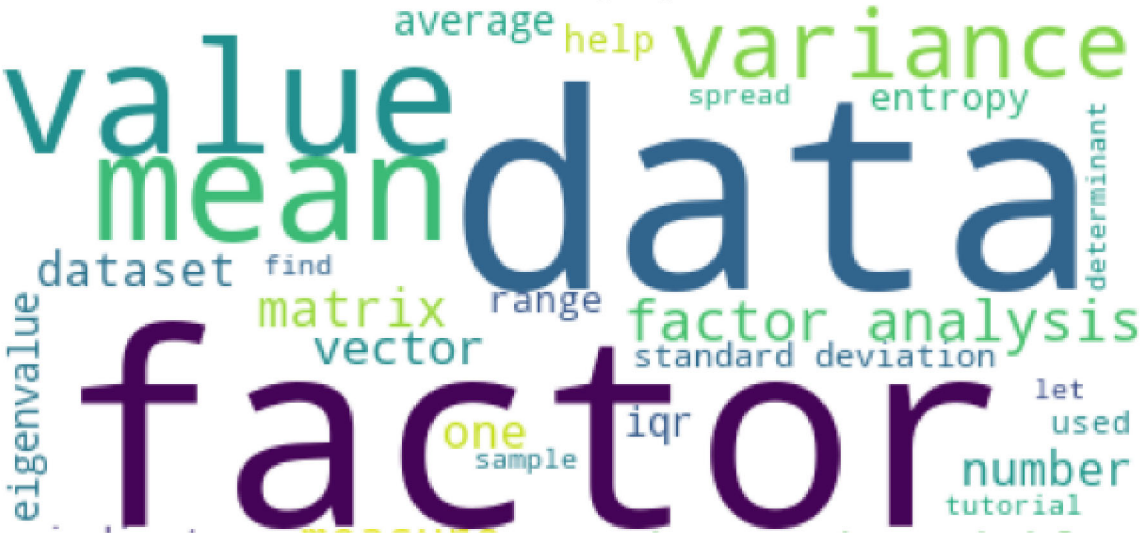
None

NLP Category Articles



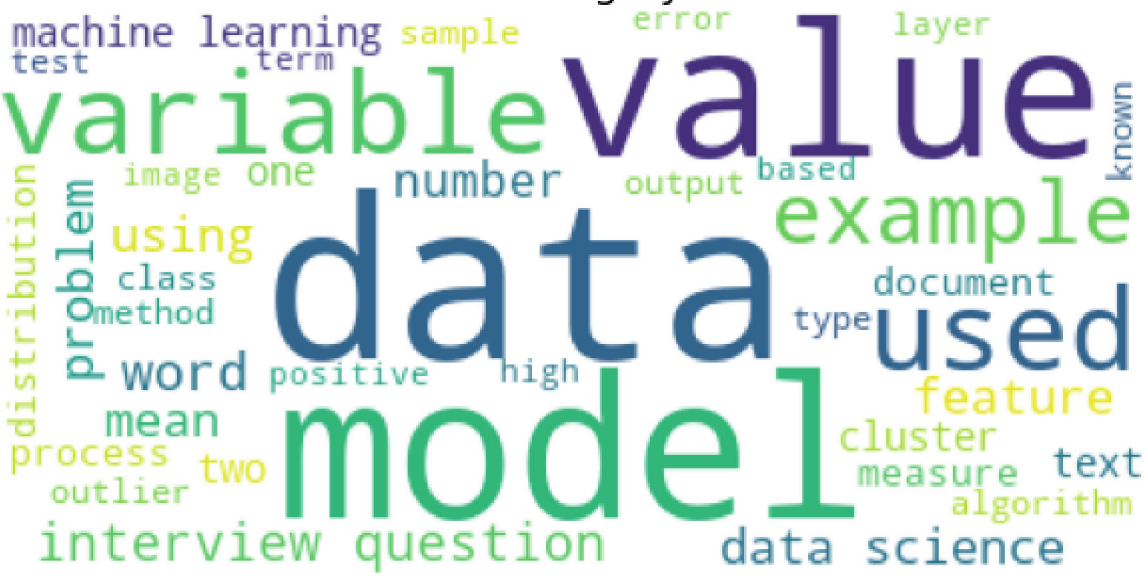
None

Statistics Category Articles



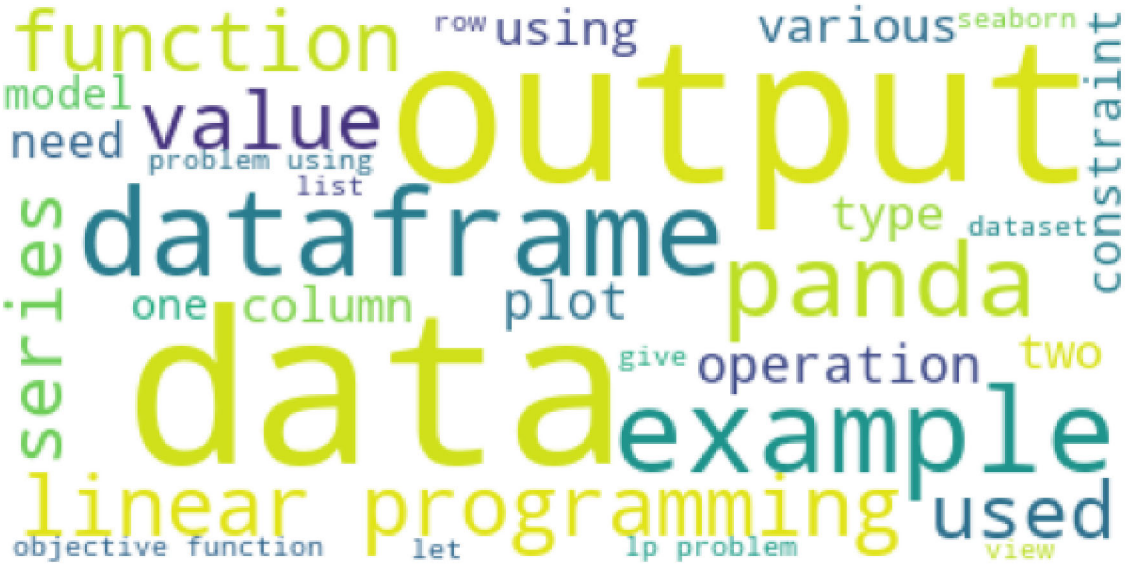
None

Interview Category Articles



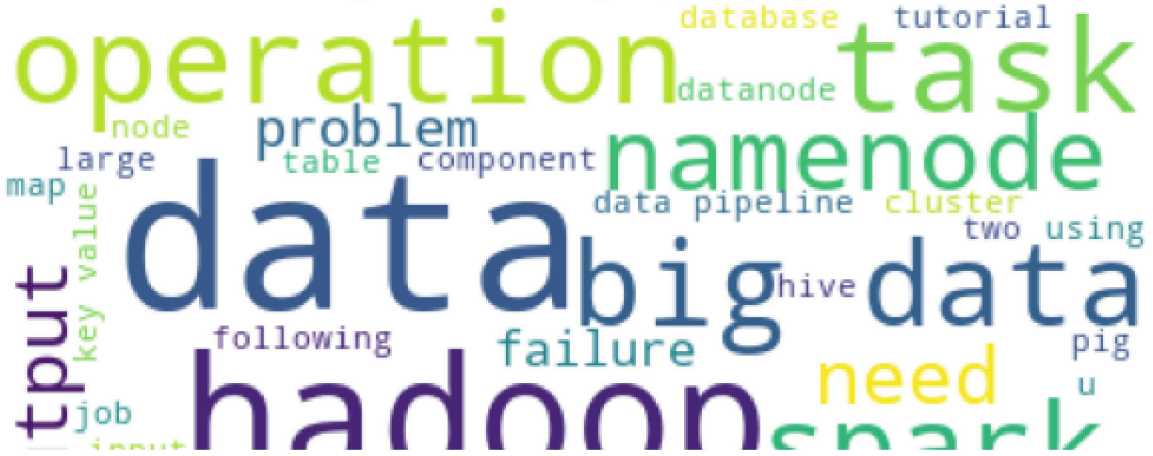
None

Python Category Articles

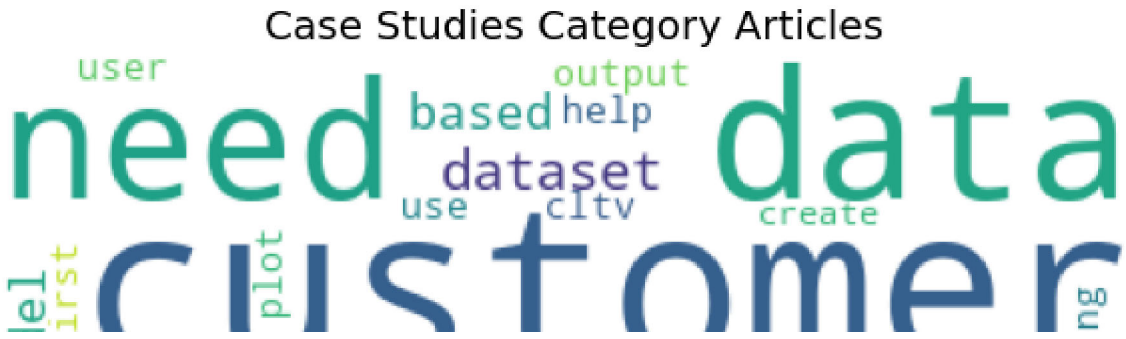


None

Big Data Category Articles



None



Vectorization

Here we will use both Bag of Words and Term Frequency-InverseDocument Frequency Vectorization techniques to get the features from text

Bag of Words (BOW)

Here we will use Bag of Words to create features for each set of documents (or articles) for our clustering model

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
bow = count_vect.fit_transform(df['Cleaned_Text'].values)
bow.shape
```

```
(105, 4381)
```

```
bow
```

```
<105x4381 sparse matrix of type '<class 'numpy.int64'>'
        with 21230 stored elements in Compressed Sparse Row format>
```

To understand what kind of words generated as columns by BOW, let's see the 10 columns generated by it

```
terms = count_vect.get_feature_names()
terms[1000:1010]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
  warnings.warn(msg, category=FutureWarning)
['costly',
 'could',
 'count',
 'counting',
```

```
 'country',
 'countvector',
 'countvectorized',
 'countvectorizer',
 'couple',
 'coupon']
```

Term Frequency Inverse Document Frequency (TfIDF)

Here we will use the TfIdf Vectorization Technique to compute the features

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vect = TfidfVectorizer()
tfidf = tfidf_vect.fit_transform(df['Cleaned_Text'].values)
tfidf.shape
```

```
(105, 4381)
```

```
terms1 = tfidf_vect.get_feature_names()
terms1[1000:1010]
```

```
['costly',
 'could',
 'count',
 'counting',
 'country',
 'countvector',
 'countvectorized',
 'countvectorizer',
 'couple',
 'coupon']
```

Step 4 : Modelling

In this step, we will model the features using the K-Means Clustering Algorithm

K-Means Clustering in BoW

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters = 7,init='k-means++',random_state=99)
model.fit(bow)
```

```
KMeans(n_clusters=7, random_state=99)
```

```
labels = model.labels_
cluster_center=model.cluster_centers_
```

```
cluster_center
```

```
array([[0.        , 0.05555556, 0.05555556, ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.01785714, 0.        ,
        0.01785714],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [1.        , 0.        , 0.        , ..., 0.        , 1.        ,
        0.        ]])
```

K-Means Clustering in TfIDf

```
from sklearn.cluster import KMeans
model_tf = KMeans(n_clusters = 7,random_state=99)
model_tf.fit(tfidf)
```

```
KMeans(n_clusters=7, random_state=99)
```

```
labels_tf = model_tf.labels_
cluster_center_tf=model_tf.cluster_centers_
```

```
cluster_center_tf
```

```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.00117377, 0.00044139, 0.0006321 , ..., 0.        , 0.00117377,
        0.00096973],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.00608162, 0.        ,
        0.        ]])
```

Step 5 : Evaluation

Here we will use Silhoutte Score and Davis Bouldin Score for evaluation

Evaluation in case of Bag of Words and K-Means

Silhoutte Score

```
from sklearn import metrics
silhouette_score = metrics.silhouette_score(bow, labels, metric='euclidean')
silhouette_score
```

```
0.10230793210846831
```

Davis Bouldin Score

```
dbi = metrics.davies_bouldin_score(bow.toarray(), labels)
dbi
```

```
2.1641759843140784
```

Evaluation in case of TfIdf and K-means

Silhouette-Score

```
from sklearn import metrics
silhouette_score_tf = metrics.silhouette_score(tfidf, labels_tf, metric='euclidean')
```

```
silhouette_score_tf
```

```
0.07699650210767844
```

Davis Bouldin Score

```
dbi_tf = metrics.davies_bouldin_score(tfidf.toarray(), labels_tf)
dbi_tf
```

```
2.9724067327446027
```

Inference

Top Features within Clusters using the BOW and K-Means

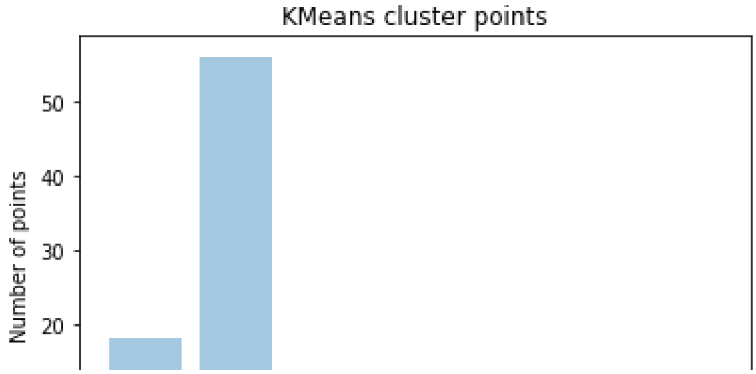
```
df['Bow Clus Label'] = model.labels_ # the last column you can see the label numebers
df.groupby(['Bow Clus Label'])['Cleaned_Text'].count()
print("Top 4 Features per cluster:")
order_centroids = model.cluster_centers_.argsort()[:, ::-1]
terms = count_vect.get_feature_names()
for i in range(7):
```

```
    print("Cluster %d:" % i, end='')
    for ind in order_centroids[i, :4]:
        print(' %s' % terms[ind], end='')
        print()
```

```
 Top 4 Features per cluster:
 Cluster 0: model
  data
  using
  function
 Cluster 1: data
  using
  output
  value
 Cluster 2: word
  text
  model
  using
 Cluster 3: problem
  using
  linear
  constraint
 Cluster 4: factor
  variable
  analysis
  observed
 Cluster 5: movie
  system
  using
  recommender
 Cluster 6: data
  mean
  value
  sample
 /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
   warnings.warn(msg, category=FutureWarning)
```

Number of Articles in various clusters

```
import matplotlib.pyplot as plt
plt.bar([x for x in range(7)], df.groupby(['Bow Clus Label'])['Cleaned_Text'].count(), alpha
plt.title('KMeans cluster points')
plt.xlabel("Cluster number")
plt.ylabel("Number of points")
plt.show()
```



Title of Articles in Various Clusters

```
for i in range(7):
  print("Articles in Cluster ", i)
  for j in range(60):
    try:
      print(df.iloc[df.groupby(['Bow Clus Label']).groups[i][j]]['Title'])
    except:
      continue
  print("-" * 70)
```

```
Working with Pandas Date and Time
Working with Strings in Pandas
Data Visualization using Pandas
Data Visualization using Matplotlib
Data Visualization using Seaborn
Data Visualization using Seaborn
Apache Airflow: A Workflow Management Platform
Apache Sqoop
Apache Hive Hands-0n
Apache Pig Hands-On
Introduction to Apache Spark
MapReduce Algorithm
Hadoop Distributed File System
Understanding BigData: Its Characteristics, Challenges, and Benefits
Introduction to Hadoop
Introduction to Customer Segmentation in Python
----------------------------------------------------------------------
Articles in Cluster  2
Text Analytics for Beginners using Python NLTK
Discovering Hidden Themes of Documents
Sentiment Analysis using Python
Text Classification using Python spaCy
Text Analytics for Beginners using Python spaCy Part-2
Text Analytics for Beginners using Python spaCy Part-1
Explore Python Gensim Library For NLP
Data Science Interview Questions Part-6 (NLP & Text Mining)

----------------------------------------------------------------------
Articles in Cluster  3
Solving Linear Programming using Python PuLP
Solving Staff Scheduling Problem using Linear Programming
Solving Cargo Loading Problem using Integer Programming in Python
```

```
Solving Transportation Problem using Linear Programming in Python
Solving Blending Problem in Python using Gurobi
Solving Assignment Problem using Linear Programming in Python
Transshipment Problem in Python Using PuLP
Solving Balanced Diet Problem in Python using PuLP
Solving Multi-Period Production Scheduling Problem in Python using PuLP
Sensitivity Analysis in Python
----------------------------------------------------------------------
Articles in Cluster  4
Introduction to Factor Analysis in Python
Introduction to Factor Analysis in Python
----------------------------------------------------------------------
Articles in Cluster  5
Spotify Song Recommender System in Python
Building Movie Recommender System using Text Similarity
Book Recommender System using KNN
Recommendation System for Streaming Platforms
Building Movie Recommender System using Text Similarity
Spotify Song Recommender System in Python
Building Movie Recommender System using Text Similarity
Book Recommender System using KNN
Recommendation System for Streaming Platforms
----------------------------------------------------------------------
Articles in Cluster  6
Measures of Dispersion
Data Science Interview Questions Part-7(Statistics)
```

Top Features within Clusters using the TfIdf and K-Means

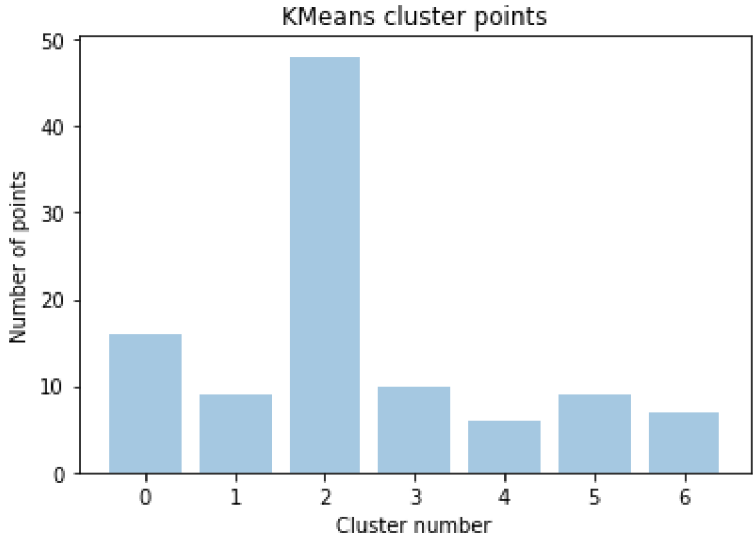
```
df1 = df
df1['Tfidf Clus Label'] = model_tf.labels_
df1.groupby(['Tfidf Clus Label'])['Cleaned_Text'].count()
print("Top Features per cluster:")
order_centroids = model_tf.cluster_centers_.argsort()[:, ::-1]
for i in range(7):
    print("Cluster %d:" % i, end='')
    for ind in order_centroids[i, :4]:
        print(' %s' % terms1[ind], end='')
        print()
```

```
Top Features per cluster:
Cluster 0: panda
 dataframe
 series
 output
Cluster 1: movie
 system
 recommender
 song
Cluster 2: model
 text
 data
```

```
 word
Cluster 3: problem
 constraint
 linear
 pulp
Cluster 4: plot
 data
 seaborn
 output
Cluster 5: hadoop
 data
 big
 spark
Cluster 6: cluster
 clustering
 distance
 data
```

Number of Articles in Each Cluster

```
plt.bar([x for x in range(7)], df1.groupby(['Tfidf Clus Label'])['Cleaned_Text'].count(), alp
plt.title('KMeans cluster points')
plt.xlabel("Cluster number")
plt.ylabel("Number of points")
plt.show()
```



Title of Articles in various Clusters

```
for i in range(7):
  print("Articles in Cluster ", i)
  for j in range(50):
    try:
      print(df.iloc[df.groupby(['Tfidf Clus Label']).groups[i][j]]['Title'])
```

```
    except:
      continue
  print("-" * 70)
```

```
    Text Analytics for Beginners using Python spaCy Part-1
    Custom Entity Recognition Model using Python spaCy
    Explore Python Gensim Library For NLP
    Measures of Central Tendency
    Measures of Dispersion
    Introduction to Factor Analysis in Python
    Demystifying Mathematical Concepts for Deep Learning
    Data Science Interview Questions Part-8(Deep Learning)
    Data Science Interview Questions Part-6 (NLP & Text Mining)
    Data Scientist explores new problems and improves products/solutions every day. – Avi
    Data Science Interview Questions Part-5 (Data Preprocessing)
    Data Science Interview Questions Part-3 (Classification)
    Data Science Interview Questions Part-2 (Regression Analysis)
    Data Science Interview Questions Part-1
    Predicting Customer Lifetime Value in Python
    Introduction to Customer Segmentation in Python
    Predicting Employee Churn in Python
    ----------------------------------------------------------------------
    Articles in Cluster  3
    Solving Linear Programming using Python PuLP
    Solving Staff Scheduling Problem using Linear Programming
    Solving Cargo Loading Problem using Integer Programming in Python
    Solving Transportation Problem using Linear Programming in Python
    Solving Blending Problem in Python using Gurobi
    Solving Assignment Problem using Linear Programming in Python
    Transshipment Problem in Python Using PuLP
    Solving Balanced Diet Problem in Python using PuLP
    Solving Multi-Period Production Scheduling Problem in Python using PuLP
    Sensitivity Analysis in Python
    ----------------------------------------------------------------------
    Articles in Cluster  4
    Feature Scaling: MinMax, Standard and Robust Scaler
    Data Science Interview Questions Part-7(Statistics)
    Data Visualization using Pandas
    Data Visualization using Matplotlib
    Data Visualization using Seaborn
    Data Visualization using Seaborn
    ----------------------------------------------------------------------
    Articles in Cluster  5
    How to find an internship in Data Science?
    Apache Airflow: A Workflow Management Platform
    Apache Sqoop
    Apache Hive Hands-0n
    Apache Pig Hands-On
    Introduction to Apache Spark
    Hadoop Distributed File System
    Understanding BigData: Its Characteristics, Challenges, and Benefits
    Introduction to Hadoop
    ----------------------------------------------------------------------
    Articles in Cluster  6
    Introduction to Cluster Analysis
    K-Means Clustering
```

```
DBSCAN Clustering
Spectral Clustering
Agglomerative Clustering
Evaluating Clustering Methods
Data Science Interview Questions Part-4 (Unsupervised Learning)
```

Result

The lower the Silhouette and Davis Bouldin Score in positive the better the clustering.

- In case of the Silhouette Score, the Tf-Idf performs much better
- In case of Davis Bouldin Score the Bag of Words Model outperforms the Tf-Idf

```
output_df = pd.DataFrame([[silhouette_score,dbi],[silhouette_score_tf,dbi_tf]],columns=['Silh
output_df.index = ['Bag of Words','Tf-Idf']
output_df
```

	Silhouette Score	Davis Bouldin Score
Bag of Words	0.102308	2.164176
Tf-Idf	0.076997	2.972407