# **Vertical Search Engine and Document Clustering using NLP**

Table of Contents

[Abstract: 2](#_Toc138431625)

[Task 1. Vertical Search Engine 3](#_Toc138431626)

[1.1. Crawler Component: 3](#_Toc138431627)

[1.1.1. BeautifulSoup : 3](#_Toc138431628)

[1.1.2. Schedule Library – Re-Crawling new information: 4](#_Toc138431629)

[1.1.3. Rules of the “robots.txt” file: 4](#_Toc138431630)

[1.2. Data pre-processing: 4](#_Toc138431631)

[1.3. Inverted Index: 7](#_Toc138431632)

[1.4. Query Processing: 7](#_Toc138431633)

[1.4.1. Long query processing using TF-IDF (Ranked Retrieval and Cosine Similarity): 7](#_Toc138431634)

[1.4.2. Ranked retrieval query processing: 10](#_Toc138431635)

[1.4.3. Query Processing using Cosine Similarity: 11](#_Toc138431636)

[Task 2. Document Clustering 13](#_Toc138431637)

[2.1. What is Clustering? 13](#_Toc138431638)

[2.2. K-Means Clustering: 13](#_Toc138431639)

[2.2.1. Input Document Collection: 14](#_Toc138431640)

[2.2.2. Data pre-processing: 14](#_Toc138431641)

[2.2.3. Implementing Tfidf Vectorizer: 15](#_Toc138431642)

[2.3. Naïve Bayes Model – Text Classification: 18](#_Toc138431643)

[2.3.1. Defining - BBC News Dataset: 19](#_Toc138431644)

[2.3.2. Data Pre-processing: 21](#_Toc138431645)

[2.3.3. Multinominal Naïve Bayes and Accuracy of the Model: 22](#_Toc138431646)

[References 24](#_Toc138431647)

[APPENDIX A: Document Clustering Code 26](#_Toc138431648)

[APPENDIX B: Naïve Bayes Classification Code 30](#_Toc138431649)

# Abstract:

In this project, the first task is to develop a ***Vertical Search Engine***similar to Google Scholar, but with the objective of retrieving all publications published by members of the ***School of Economics, Finance and Accounting (SEFA) from my university.*** (That is, at least one of the co-authors is a member of SEFA.)

As part of the development of the search engine, the system is designed to crawl relevant web pages and retrieve information about each publication and its author.

The rules in the ***“robots.txt”*** file have been considered when designing our search engine. We have configured the crawler so that it does not hit the server too fast to perform in a ***polite*** manner and it is ***scheduled*** to ***re-crawl*** new information every once a week and update the index with the new data.

Also, we have applied various ***data pre-processing*** steps to both the crawled data and to the user’s queries.

In **Task 2**, we are developing a ***document clustering system***, in which we have collected several documents from the BBC News website pertaining to different categories, such as Sports, Health and Politics. For the purpose of collecting these documents, we have implemented a crawler to extract the data from the BBC News website.

We have implemented clustering methods like ***K-Means*** and ***Naive Bayes*** and trained the data to cluster accordingly after collecting the required documents. In conclusion, a new document is assigned to an existing cluster using the created model, i.e., the user enters a document (e.g., a sentence) and your system determines the appropriate cluster.

# Task 1. Vertical Search Engine

Vertical search engines are different from general web search engines, in that they cater to a particular segment of online content. These search engines are also known as specialty search engines or topical search engines. Depending on the topicality, media type, or genre of content, a vertical content area may be defined.

Using Google Scholar, users can search for full text or metadata of scholarly literature across a variety of publishing formats and disciplines. Google Scholar is an excellent tool for searching scholarly literature in a broad manner. You can search across multiple disciplines and sources, including articles, theses, books, abstracts, and court opinions, available from academic publishers, professional societies, online repositories, and universities.

A similar vertical search engine was developed as part of our coursework. The search engine provides information about publications that have been published by members of the SEFA department.

Graphical user interface, text, application, email

Description automatically generated

Figure 1 - Installing and Importing all necessary Libraries.

## Crawler Component:

Using Python, we have created a crawler such that it retrieves detailed information for each book and paper as mentioned below:

* Publication/Book Title
* Publication/Book URL
* Author Name
* Author Profile Link
* Publication Year

In the crawler code, we have created separate lists to store each of these information and then we have converted them into a pandas data frame to process further.

### BeautifulSoup :

BeautifulSoup is a python package/library that is used web scraping purposes and to extract structured data from a website. It allows us to parse data from HTML and XML documents into a readable format. Below are the steps that are followed to Inspect the HTML page.

Following are the steps involved to retrieve the information about each publication:

* Using Python program to retrieve the information about each publication from the web pages.
* For seeing the HTML code of specific part of the page, for example the name of the first person in the page, you can right-click there and select the option provided to “***inspect***” the element.
* Then we have to select the exact tags for the title, author, and the data from the inspect page and use these tags, attributes, and classes in the BeautifulSoup to scrape the data from these URLs.
* Different tags that we have used in the code are <a> ,<h3> and <span> tags.
* The classes are link, link person and data class.

### Schedule Library – Re-Crawling new information:

The ***Schedule Library*** is used as part of the crawler code. The purpose of this feature is to schedule a task for a particular time every day or on a particular day of the week. As a result of Schedule Library, our systems time is matched to our scheduled time. The job function (command function that is scheduled) is invoked once the scheduled time matches the system time. Our crawler is ***scheduled to run once a week to re-crawl*** and extract new data automatically.

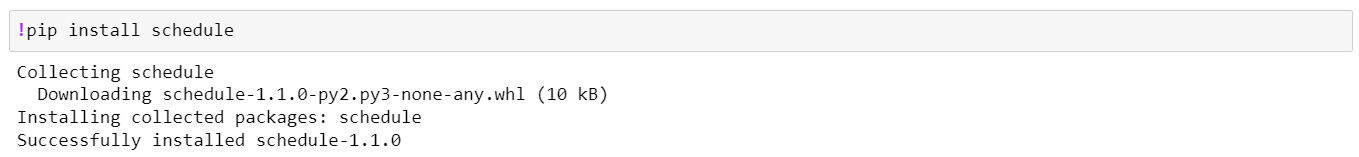
****

Figure 5 - Installing Schedule Library

In the below mentioned figure, we can see that the crawling at the scheduled time is executed and was able to fetch all the data from the website.

### Rules of the “robots.txt” file:

To ensure that the crawler is not hitting the server too fast, we have preserved the ***“robots.txt”*** rules and have used the ***time.sleep()*** method to suspends execution for the given number of seconds.

## Data pre-processing:

We have stored all the required information in separate lists, and we have converted them into pandas dataframe for further pre-processing.

Text

Description automatically generated with low confidence

Figure 8 - Converting all lists into a pandas dataframe

The data processing steps that we have applied for the “Title of the publication” are as follows:

* ***Removing punctuation*** – To remove all the special characters and punctuation marks from the title text.
* Converting all the data into ***lower case*** format.

Graphical user interface, application, Teams

Description automatically generated

Figure 10 - Punctuation Removal and Lower-case format function

* ***Tokenization documents*** - Collectively, the different units into which we can break down text (words, characters, or N-grams) are called “tokens” and breaking down text into such tokens is called “tokenization”. Figure 11 - Tokenization Example

Diagram, schematic

Description automatically generated

* ***Removing stop words*** – it is one of the basic steps that is followed in the pre-processing steps across different NLP (Natural Language Processing) techniques. The idea is to simply remove all the words that occur commonly across all the documents in the corpus.

Graphical user interface, application

Description automatically generated

Figure 12 - Tokenization and Stop words Removal Function

* ***Lemmatization*** – it is a text normalization technique used in the NLP. Essentially, lemmatization is a method that switches any kind of word to its base root word. In our project, instead of stemming we have used this technique to normalize the data.

Graphical user interface, application

Description automatically generatedChart, diagram

Description automatically generated

Figure 13 - Data Pre-processing process and Lemmatization

Graphical user interface, text, application, email

Description automatically generated

Figure 14 - Normalization using Lemmatization from NLP

* We have combined the Title Name, Author Name column and applied the data pre-processing steps to the data and created a new column called ***“Filtered Corpus”***.
* In order to view the entire contents of the dataframe, we have set the ***“pd.set\_option”*** for displaying the output.

## Inverted Index:

After performing the data pre-processing to the raw data, we can now be able to construct the inverted index based on the filtered data.

* An inverted index is a data structure that allows efficient, full-text searches in the database.
* It is a very important part of information retrieval systems and search engines that stores a mapping of words (or any type of search terms) to their locations in the database table or document.
* As Incidence Matrix is extremely sparse, this approach is inefficient.
* But Inverted Index, only keeps the “1” positions when compared to the Incidence Matrix.
* For each ***term “t”***, we must store the list of all documents that contains the term “t”.
* Each document is identified by a “***document*** ***ID***” or “document serial number”.
* ***Postings*** – A document is also called known as “Postings”.
* The inverted index acts like a dictionary, with the “terms” as their keys and the list of all the postings/document IDs for the term as their values.
* To locate any term faster and making the query processing easier for any given term, the keys and list of postings are **“sorted”** and stored in the Inverted Index.
* This is like the way we store words in a normal dictionary in sorted and alphabetical order.
* To increase the efficiency and speed of the search engine we must store the data in Inverted Index in a sorted manner. In our search engine, we are using “Inverted Index”, rather than Incidence matrix.

Figure 17 -Constructed Inverted Index on the Filtered Corpus and Postings list returning the list of documents in which the term "management" is presentGraphical user interface, text, application, email

Description automatically generated

Figure 18 - Inverted Index and Postings List

## Query Processing:

### Long query processing using TF-IDF (Ranked Retrieval and Cosine Similarity):

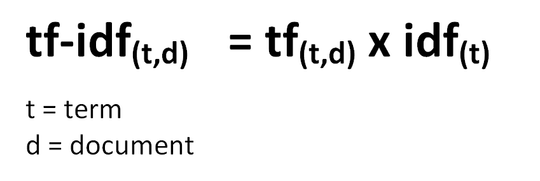
**What is TF-IDF?**

* Using the Inverted Index, we were able to retrieve the documents of the Publications for single word searches. However, for long queries, we need to follow a different approach to retrieve the data using TF-IDF technique.
* TF-IDF, short for Term Frequency - Inverse Document Frequency, is a text mining technique, that gives a numeric statistic as to how important a word is to a document in a collection or corpus.
* This is a technique used to categorize documents according to certain words and their importance to the document.
* When it comes to TF-IDF, this is also considered as a measure to categorize documents based on the terms that appear in it.
* But unlike BoW(Bag of Words), this does provide a weight for each term, rather than just the count. The TF-IDF value measures the relevance, not frequency.

**Why TF-IDF?**

TF-IDF allows us to score the importance of words in a document, based on how frequently they appear on multiple documents.

* If the word appears frequently in a document - assign a high score to that word (term frequency - TF) .
* If the word appears in a lot of documents - assign a low score to that word. (Inverse document frequency - IDF)
* TF-IDF gives the importance of each of terms in a document not only as an isolated term, but also as a term within the entire document collection.
* This leads to ranking and scoring documents, against a query as well as classification of documents and modelling documents and terms within a vector space.

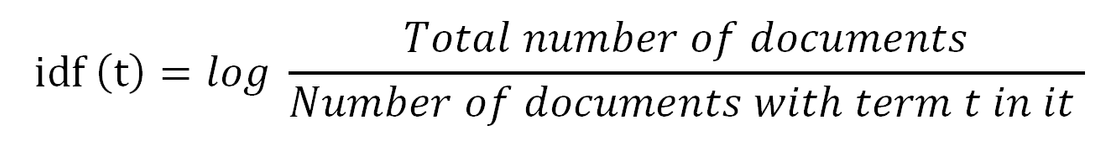
****

**Calculating Term Frequency:**

**A black text on a white background

Description automatically generated with low confidence**

**Calculating Inverse Document Frequency:**

****

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

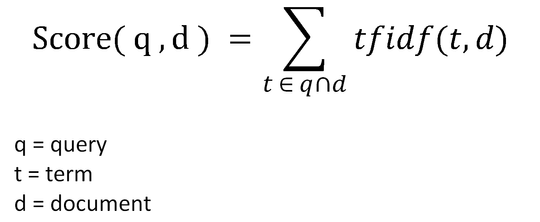
Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

### Ranked retrieval query processing:

* When a user queries for certain information, the system needs to retrieve the most relevant documents to satisfy the user's information need.
* This relevance is called document ranking which ranks the documents in the order of relevance, where the highest relevance ranked as 1st.
* Using this, finding the rank of documents for a query, we need to calculate the score of the document for a given query. We can use the following formula.

****

* The score is calculated by taking the terms which are both present in the document d, and the query q.
* We check for the TF-IDF values for each of those terms and get a summation.
* This is the score for document d, for the query q.
* The technique given above changes with the application and parameters being utilized.

**Text

Description automatically generated**

### Query Processing using Cosine Similarity:

* Cosine similarity measures the similarity between two vectors of an inner product space.
* It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction.
* It is often used to measure document similarity in text analysis.
* Cosine similarity is a measure of similarity that can be used to compare documents or, say, give a ranking of documents with respect to a given vector of query words.
* Let x and y be two vectors for comparison. Using the cosine measure as a similarity function, we have

Text, letter

Description automatically generated

* where ||x|| is the Euclidean norm of vector , defined as .
* Conceptually, it is the length of the vector.
* Similarly, ||y|| is the Euclidean norm of vector y.
* The measure computes the cosine of the angle between vectors x and y.
* A cosine value of 0 means that the two vectors are at 90 degrees to each other (orthogonal) and have no match.
* The closer the cosine value to 1, the smaller the angle and the greater the match between vectors.

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

# Task 2. Document Clustering

## What is Clustering?

Clustering is a set of methods that is used to partition the data into groups or clusters. Clusters are also defined as group of data objects that are more similar to other objects in the cluster than they are to data objects in other clusters. In practice, the main two aspects of clustering are that it helps to identify two qualities of data:

***Meaningfulness*** - Meaningful clusters expand domain knowledge.

***Usefulness*** – Useful clusters on the other hand, servers as an intermediate step in a data pipeline.

## K-Means Clustering:

The algorithm aims to produce a set of suitable labels (targets) under ***unsupervised*** ***learning***. In other words, we have inputs (X’s) that are used for analysis with no corresponding target (Y). Unsupervised learning seeks to model the underlying structure or distribution in the data to learn more about the data since it is not given labelled training data. It is, therefore, useful for exploring new data sets that are large and complex.

***K – Means is an example of Partitional Clustering and unsupervised machine learning algorithm.*** In this type of clustering, it specifically tries to put the data into the number of clusters as we configure.

**Algorithm of K-Means Clustering:**

Select the number of clusters we want to identify in the data. This represents the “K” in the K-means clustering.

*Step 1:* In case if we select, k = 3, it means that we are trying to identify 3 clusters.

*Step 2:* Randomly selects 3 distinct data points as the initial cluster.

*Step 3:* Measure the distance between the first data point and the three initial clusters.

*Step 4:* Assign the first data point to the nearest cluster (to the cluster with the lowest distance value).

*Step 5:* Now that we have all the data points assigned to each cluster groups based on their distance values. As next step, we calculate the mean of each cluster.

*Step 6:* Repeat step 3, to measure and cluster the data points using the mean values.

These steps are repeated until there are changes in the data points. We can assess the quality of the clustering by adding up the variation within each cluster.

### Input Document Collection:

* The first step is to collect number of documents from different categories such as Sports, Health, and Politics from BBC News website.
* For the document collection, we have used the BeautifulSoup Python package to scrape data from the BBC News website.
* We have created a list named ***bbc\_news\_list*** and stored all the documents from each of these categories.
* Also, there are three separated lists named sports\_list, health\_list and politics\_list to store the documents and to determine the length of the documents collected for clustering.
* Total number of documents collected are 182.

A picture containing text

Description automatically generated

Figure 20 - Number of documents collected per Categories and their total

### Data pre-processing:

In the K-means clustering method, once we have collected all the documents, we can begin the data processing seps by converting the documents into tokens, then removing the stop words and finally performing text normalization technique using lemmatization from NLP.

Graphical user interface, text, application, email

Description automatically generated

Figure 21 - Data Pre-processing steps

Text

Description automatically generated

Figure 22 -Implementation of Tokenization and Lemmatization

### Implementing Tfidf Vectorizer:

After performing the data processing on the document, the final ***“filtered\_bbc\_docs”*** are obtained. As a next step, we have converted this into a matrix of TF-IDF features using the ***TfidfVectorizer*** from ***sklearn.feature\_exrtaction.text*** library.

Table

Description automatically generated

Figure 23 - Converting to TF-IDF features

After fitting the filtered document using the vectorizer, we can perform the K-Means clustering by setting the value of “k” (number of clusters) as mentioned in the below shown figure.



Figure 24 - Applying the Tfidf Vectorizer and K-Means Clustering

We have printed the “top 5” terms from each cluster’s groups using the below mentioned code.

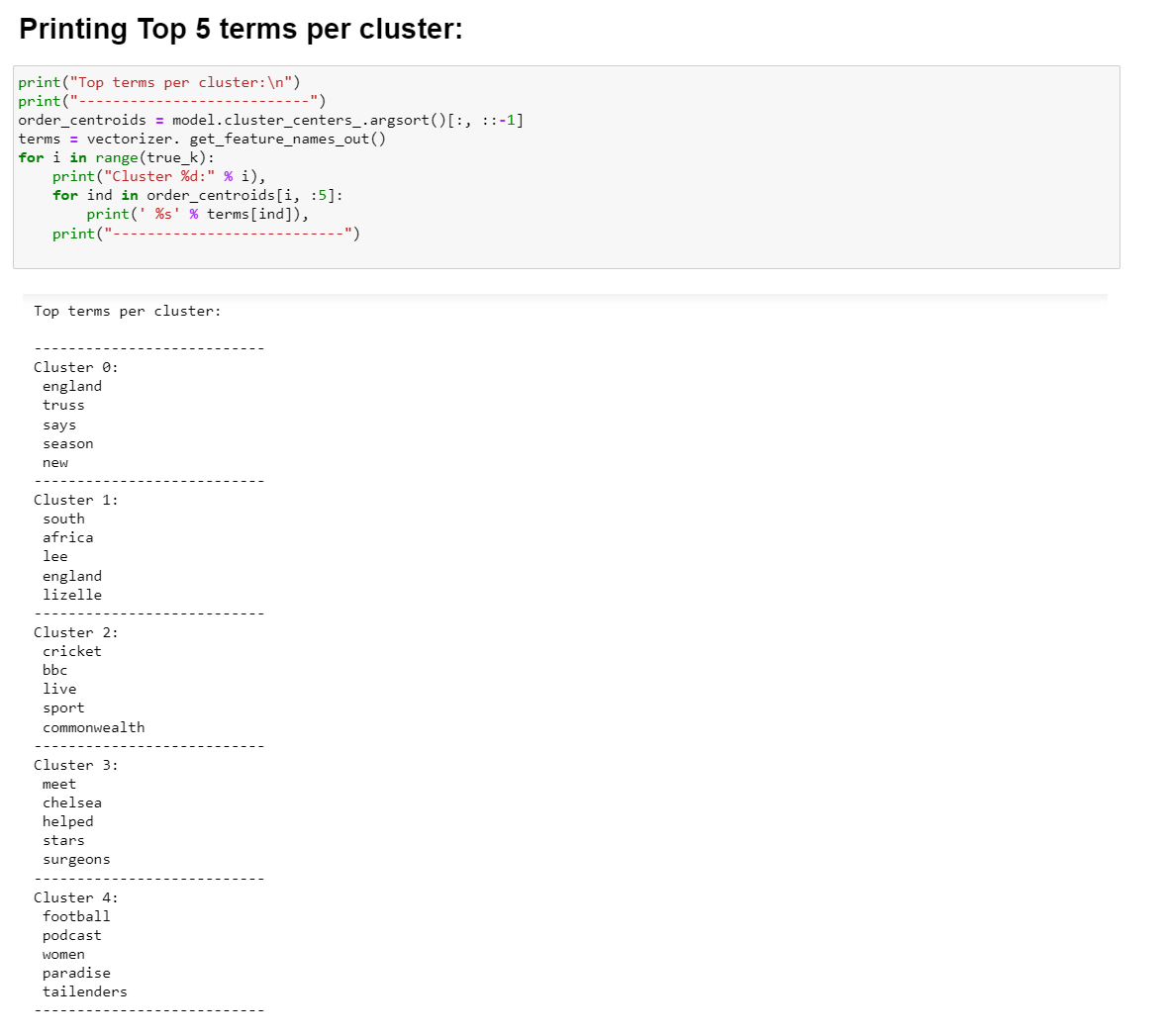


Figure 25 - Printing top 5 terms per Cluster

Text

Description automatically generated with medium confidence

Figure 26 – For new documents predicting the right cluster groups

Graphical user interface, text, application

Description automatically generated

## Naïve Bayes Model – Text Classification:

Supervised learning uses ***labelled data set*,**one that contains matched sets of observed inputs, X’s, and the associated outputs, Y’s. The algorithm is “***trained***,” i.e., the machine learning algorithm is applied to this data set to infer the patterns between the inputs and outputs.

Naive Bayes is a statistical classification technique based on Bayes Theorem.

It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm.

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

We have used Naïve Bayes algorithm for text classification, and this is one of the Supervised Learning models.

Table

Description automatically generated

Figure 27 - Difference between Supervised and Unsupervised Methods

Chart, scatter chart

Description automatically generated

### Defining - BBC News Dataset:

As a result, I have created a pandas dataframe with all the scraped data from the BBC News Website. This dataframe consists of 2 columns named the Category (Sport, Health, and Politics)and Text column.

Table

Description automatically generated with low confidence

Figure 28 - Crawling the data from BBC News website

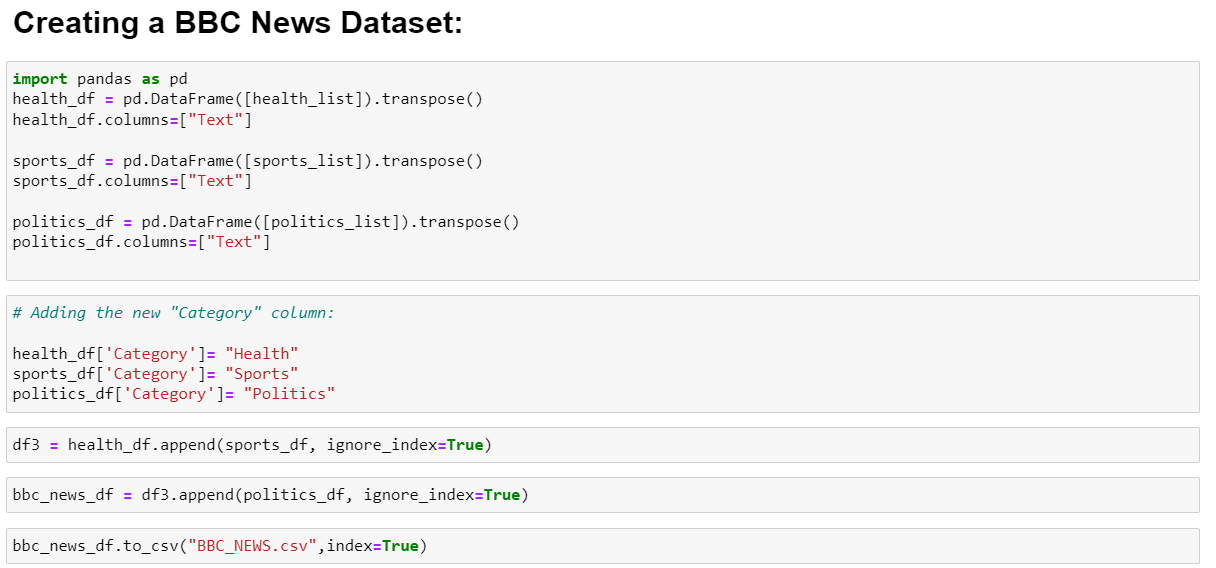


Figure 29 - Creating a pandas dataframe for the Scrapped data

We have created a dataframe named ***“bbc\_news\_df”*** with 2 columns known as Categories and the Text column containing the scraped data from the BBC News website.

Table

Description automatically generated

Figure 30 - BBC News dataframe

This bar graph represents the count of texts from different categories like Health, Sports and Politics. We have used matplotlib library to plot the bar graph.

Chart, bar chart

Description automatically generated

### Data Pre-processing:

Shape

Description automatically generated with medium confidence

Importing all the necessary libraries to perform the data pre-processing on the dataframe. After performing all the necessary data pre-processing steps like tokenization, converting the words into lower case, removing punctuation marks, stop words and normalization with lemmatization we got the final filtered text to perform the Naïve Bayes classification.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

After performing all the necessary data pre-processing steps on the “Text” column, we have obtained the output in the new column as “Filtered\_Text”.

Figure 31 - Filtered Text column after data pre-processing

Text, table

Description automatically generated

### Multinominal Naïve Bayes and Accuracy of the Model:

The Multinominal Naïve Bayes algorithm is a popular Bayesian learning approach used in Natural Language Processing. In this method, the program guesses the tag of text, such as an email or a newspaper story, using the Bayes theorem. It calculates each tag’s likelihood.

As the data is ready to perform the text classification, we are using the ***“CountVectorizer”*** from the sklearn.feature\_extraction library to transform the data into features for the training data.

We have split the data into Training and Test dataset. After training the model with the training dataset, we can apply the model on the test data and predict the accuracy of the model.

The Accuracy of the model is the number of data points classified correctly divided by all the points (in test set). Using the accuracy\_score from the sklearn.metrics we have calculated the accuracy score. Our model’s accuracy score is 88.24%, which means that 88% of the data points are correctly classified.

**Accuracy Score = (TP+TN)/ (TP+FN+TN+FP)**

Graphical user interface, text, application, email

Description automatically generated

Figure 32 - Implementing the Naive Bayes Model and Accuracy checking

Graphical user interface, application

Description automatically generated

Figure 33 - Confusion Matrix for the Actual and Predicted Values

Graphical user interface, text, application, email

Description automatically generated

Figure 34 - Predicting the Categories for the New documents

# References

*What is lemmatization in NLP?* (n.d.). Quora. Retrieved August 5, 2022, from <https://www.quora.com/What-is-lemmatization-in-NLP>

*Deep Learning for Text*. (2017, October 2). Manning. <https://freecontent.manning.com/deep-learning-for-text/>

*What is an inverted index?* (n.d.). Educative: Interactive Courses for Software Developers. Retrieved August 5, 2022, from <https://www.educative.io/answers/what-is-an-inverted-index>

*Supervised Machine Learning, Unsupervised Machine Learning, and Deep Learning*. (2021, March 6). CFA, FRM, and Actuarial Exams Study Notes. <https://analystprep.com/study-notes/cfa-level-2/quantitative-method/supervised-machine-learning-unsupervised-machine-learning-deep-learning/>

*(2022). Differencebetween.net.* <http://cdn.differencebetween.net/wp-content/uploads/2018/01/Differences-Between-Supervised-Learning-and-Unsupervised-Learning.png>

*BBC. (2019, February 6). BBC - Home. BBC Homepage.* <https://www.bbc.co.uk/>

*An Efficient Topic Modelling Approach for Text Mining and Information Retrieval through K-means Clustering - Scientific Figure on ResearchGate.* <https://www.researchgate.net/figure/PREPROCESSING-STEPS-FOR-TEXT-DOCUMENTS-COLLECTION_fig1_338501601>

*Information Retrieval: CHAPTER 14: RANKING ALGORITHMS*. (n.d.). Orion.lcg.ufrj.br. Retrieved August 5, 2022, from <http://orion.lcg.ufrj.br/Dr.Dobbs/books/book5/chap14.htm>

*Sugathadasa, K. M. (2017, June 5).* *TF-IDF in the Field of Information Retrieval*. Keetmalin. <https://keetmalin.wixsite.com/keetmalin/post/2017/06/05/tf-idf-in-the-field-of-information-retrieval#:~:text=When%20a%20user%20queries%20for>

*Cosine Similarity - an overview | ScienceDirect Topics*. (n.d.). Www.sciencedirect.com. <https://www.sciencedirect.com/topics/computer-science/cosine-similarity#:~:text=Cosine%20similarity%20measures%20the%20similarity>

*williamscott701. (2019, August 26). Information-Retrieval/TF-IDF.ipynb at master · williamscott701/Information-Retrieval*. GitHub. <https://github.com/williamscott701/Information-Retrieval/blob/master/2.%20TF-IDF%20Ranking%20-%20Cosine%20Similarity%2C%20Matching%20Score/TF-IDF.ipynb>

*Bhardwaj, A. (2020, May 27). Silhouette Coefficient : Validating clustering techniques. Medium*. <https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c>

*Naive Bayes Classifier Tutorial: with Python Scikit-learn*. (n.d.). Www.datacamp.com. Retrieved August 5, 2022, from <https://www.datacamp.com/tutorial/naive-bayes-scikit-learn>

*Python, R. (n.d.). K-Means Clustering in Python: A Practical Guide – Real Python. Realpython.com. Retrieved August 5, 2022, from* <https://realpython.com/k-means-clustering-python/#partitional-clustering>

Yadam, G. (2021, May 22). *Text Clustering with K-Means in an automated Approach*. Text Clustering with K-Means in an Automated Approach. <https://medium.com/text-clustering-with-k-means-in-an-automated/text-clustering-with-k-means-in-an-automated-approach-3302b48f98b6>

## APPENDIX A: Document Clustering Code









## APPENDIX B: Naïve Bayes Classification Code













