Information Retrieval Coursework

Alexandra Espialidou
Faculty of Engineering, Environment and Computing 7071CEM
MSc Data Science
Coventry University
espialidoa@uni.coventry.ac.uk

***Abstract*— This report delves in two key tasks. The first one is the development of a Vertical Search Engine made for papers/books published by a member of the Center for Global Learning (CGL) at Coventry University. The second task is the implementation of a text classification system with the help of the Naïve Bayes classifier.**

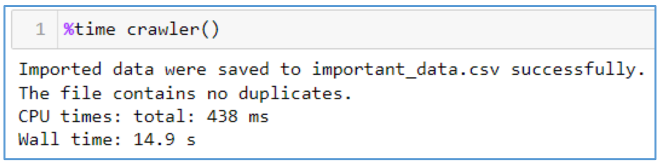
Keywords—Information Retrieval, Vertical Search Engine, Text Classification, Naïve Bayes

I. Task 1. Vertical Search engine

A. Crawler

The first step in constructing a vertical search engine is identifying the correct URL from which to gather relevant data. In this paper the selected URL is: https://pureportal.coventry.ac.uk/en/organisations/centre-global-learning/publications/. This link provides publications written by authors who are to this day, or once were members of the CGL. It contains information about the title of the publication, the authors, the date, the link to the publication and the link to the authors' profiles (also called "pureportal" profile) pages.
The crawler was built based on this URL. It iterates over the 6 pages and the 49 publications per page and pulls information regarding the publication's title, authors, date, link to the actual publication and all the "pureportal" profiles of the authors that are still members of the CGL. This information then gets saved to a CSV file, named '*important_data.csv*'. Data about 286 publications get saved.

Here is a screenshot showing the time the crawler() function needed to run.

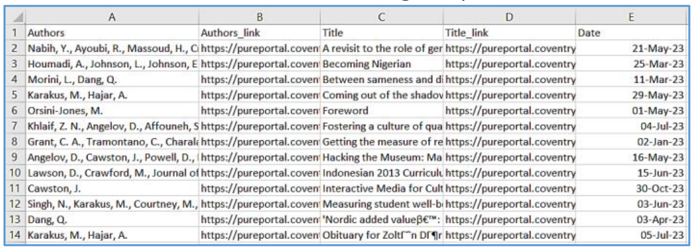


```
%time crawler()

Imported data were saved to important_data.csv successfully.
The file contains no duplicates.
CPU times: total: 438 ms
Wall time: 14.9 s
```

Image 1: Outcome of %time crawler()

The next step involves performing data validation and filtering to ensure the quality and accuracy of the pulled information. The newly created database gets checked for null values. This deletes any publications where none of the authors have a pureportal profile, indicating that they are not a part of the CGL anymore. After this process, there are 239 publications left, from 22 authors/members of the CGL, which serve as the foundation of the subsequent stages of the vertical search engine development. Below is a screenshot from the CSV file, showing the pulled data.



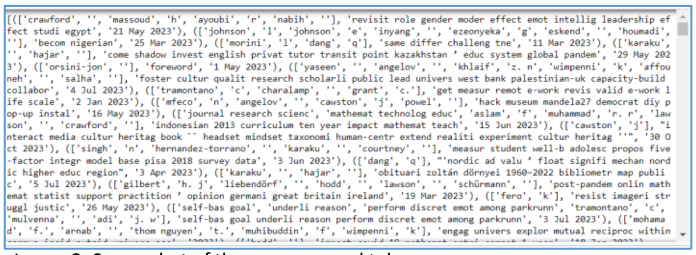
	A	B	C	D	E
1	Authors	Authors_link	Title	Title_link	Date
2	Nabih, Y., Ayoubi, R., Massoud, H., C	https://pureportal.coven	A revisit to the role of ger	https://pureportal.coventry	21-May-23
3	Houmadi, A., Johnson, L., Johnson, E	https://pureportal.coven	Becoming Nigerian	https://pureportal.coventry	25-Mar-23
4	Morini, L., Dang, Q.	https://pureportal.coven	Between sameness and di	https://pureportal.coventry	11-Mar-23
5	Karakus, M., Hajar, A.	https://pureportal.coven	Coming out of the shadov	https://pureportal.coventry	29-May-23
6	Orsini-Jones, M.	https://pureportal.coven	Foreword	https://pureportal.coventry	01-May-23
7	Khlaif, Z. N., Angelov, D., Affouneh, S	https://pureportal.coven	Fostering a culture of qua	https://pureportal.coventry	04-Jul-23
8	Grant, C. A., Tramontano, C., Charal	https://pureportal.coven	Getting the measure of re	https://pureportal.coventry	02-Jan-23
9	Angelov, D., Cawston, J., Powell, D.,	https://pureportal.coven	Hacking the Museum: Ma	https://pureportal.coventry	16-May-23
10	Lawson, D., Crawford, M., Journal of	https://pureportal.coven	Indonesian 2013 Curricul	https://pureportal.coventry	15-Jun-23
11	Cawston, J.	https://pureportal.coven	Interactive Media for Cult	https://pureportal.coventry	30-Oct-23
12	Singh, N., Karakus, M., Courtney, M.,	https://pureportal.coven	Measuring student well-b	https://pureportal.coventry	03-Jun-23
13	Dang, Q.	https://pureportal.coven	'Nordic added valueβ€™:	https://pureportal.coventry	03-Apr-23
14	Karakus, M., Hajar, A.	https://pureportal.coven	Obituary for ZoltΓ¯n DΓ¶r	https://pureportal.coventry	05-Jul-23

Image 2: Screenshot of the first 14 pulled data from '*important_data.csv*'

The constructed crawler is polite and preserves the robots.txt rules of the crawled website. The crawler is also scheduled to look for new information automatically once per week as the application is running. This happens with the help of the *datetime* and *time* libraries.

B. Data Preprocessing

In the next step of preparing the data for the inverted index, the columns 'Authors', 'Title', and 'Date' from the CSV file were subjected to a series of preprocessing steps. These steps are essential in Information Retrieval to enhance the accessibility of the data during the search process. The preprocessing steps involved tokenizing the text, which means breaking it down into individual words or tokens. Additionally, all the tokens were converted to lowercase to ensure that the search process is case-insensitive. Punctuation marks were also removed from the tokens. Furthermore, common stop-words, such as articles and prepositions, were eliminated from the text. Their removal helps in reducing noise and improves the efficiency of the retrieval process. The function PorterStemmer() from the nltk library was employed to perform stemming. Stemming involves reducing words to their root form by removing suffixes, which enables mapping variations of words to their common root. This process increases the likelihood of retrieving relevant documents, even if the search query includes variations of words. The abbreviations of the months were also replaced with their corresponding full names.

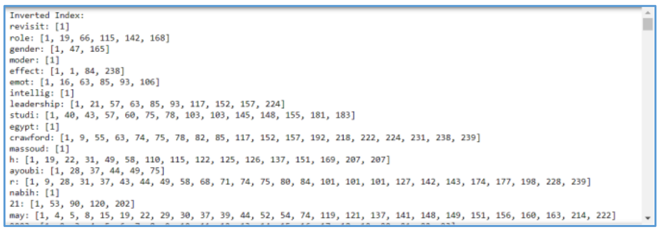


```
[(['crawford', '', 'massoud', 'h', 'ayoubi', 'r', 'nabih', ''], 'revisit role gender moder effect emot intellig leadership ef
fect studi egypt', '21 May 2023'), (['johnson', 'l', 'johnson', 'e', 'inyang', '', 'ezeonyeka', 'g', 'eskend', '', 'houmadi',
''], 'becom nigerian', '25 Mar 2023'), (['morini', 'l', 'dang', 'q'], 'same differ challeng tne', '11 Mar 2023'), (['karaku',
'', 'hajar', ''], 'come shadow invest english privat tutor transit point kazakhstan ’ educ system global pandem', '29 May 202
3'), (['orsini-jon', ''], 'foreword', '1 May 2023'), (['yaseen', '', 'angelov', '', 'khlaif', 'z. n', 'wimpenni', 'k', 'affou
neh', '', 'salha', ''], 'foster cultur qualit research scholarli public lead univers west bank palestinian-uk capacity-build
collabor', '4 Jul 2023'), (['tramontano', 'c', 'charalamp', '', 'grant', 'c.'], 'get measur remot e-work revis valid e-work l
ife scale', '2 Jan 2023'), (['mfeco', 'n', 'angelov', '', 'cawston', 'j', 'powel', ''], 'hack museum mandela27 democrat diy p
op-up instal', '16 May 2023'), (['journal research scienc', 'mathemat technolog educ', 'aslam', 'f', 'muhammad', 'r. r', 'law
son', '', 'crawford', ''], 'indonesian 2013 curriculum ten year impact mathemat teach', '15 Jun 2023'), (['cawston', 'j'], "i
nteract media cultur heritag book `` headset mindset taxonomi human-centr extend realiti experiment cultur heritag ''", '30 O
ct 2023'), (['singh', 'n', 'hernandez-torrano', '', 'karaku', '', 'courtney', ''], 'measur student well-b adolesc propos five
-factor integr model base pisa 2018 survey data', '3 Jun 2023'), (['dang', 'q'], "'nordic ad valu ’ float signifi mechan nord
ic higher educ region", '3 Apr 2023'), (['karaku', '', 'hajar', ''], 'obituari zoltán dörnyei 1960-2022 bibliometr map publi
c', '5 Jul 2023'), (['gilbert', 'h. j', 'liebendörf', '', 'hodd', '', 'lawson', '', 'schürmann', ''], 'post-pandem onlin math
emat statist support practition ’ opinion germani great britain ireland', '19 Mar 2023'), (['fero', 'k'], 'resist imageri str
uggl justic', '26 May 2023'), (['self-bas goal', 'underli reason', 'perform discret emot among parkrunn', 'tramontano', 'c',
'mulvenna', '', 'adi', 'j. w'], 'self-bas goal underli reason perform discret emot among parkrunn', '3 Jul 2023'), (['mohama
d', 'f.', 'arnab', '', 'thom nguyen', 't.', 'muhibuddin', 'f', 'wimpenni', 'k'], 'engag univers explor mutual reciproc within
```

Image 3: Screenshot of the preprocessed tokens

C. *Inverted Index*

The inverted index is an index data structure that stores the mapping of words in a document. It consists of a dictionary where tokens serve as the keys, and the values are mappings to the documents in which these tokens occur, known as the postings list. Below is a screenshot of the Inverted Index created with the defaultdict() function. This inverted index is designed to be incremental, meaning it grows and updates over time as new documents are added to the collection. Initially, the inverted index starts as an empty dictionary. As documents are processed, tokens from the document's titles, authors, and dates are added to the index, and the corresponding document IDs (postings list) are appended to the lists associated with each token. This process continues for each document in the collection.



```
Inverted Index:
revisit: [1]
role: [1, 19, 66, 115, 142, 168]
gender: [1, 47, 165]
moder: [1]
effect: [1, 1, 84, 238]
emot: [1, 16, 63, 85, 93, 106]
intellig: [1]
leadership: [1, 21, 57, 63, 85, 93, 117, 152, 157, 224]
studi: [1, 40, 43, 57, 60, 75, 78, 103, 103, 145, 148, 155, 181, 183]
egypt: [1]
crawford: [1, 9, 55, 63, 74, 75, 78, 82, 85, 117, 152, 157, 192, 218, 222, 224, 231, 238, 239]
massoud: [1]
h: [1, 19, 22, 31, 49, 58, 110, 115, 122, 125, 126, 137, 151, 169, 207, 207]
ayoubi: [1, 28, 37, 44, 49, 75]
r: [1, 9, 28, 31, 37, 43, 44, 49, 58, 68, 71, 74, 75, 80, 84, 101, 101, 101, 127, 142, 143, 174, 177, 198, 228, 239]
nabih: [1]
21: [1, 53, 90, 120, 202]
may: [1, 4, 5, 8, 15, 19, 22, 29, 30, 37, 39, 44, 52, 54, 74, 119, 121, 137, 141, 148, 149, 151, 156, 160, 163, 214, 222]
```

Image 4: Screenshot of the dictionary created from the Inverted Index

It can be seen that the term 'role' can be found in a total of 6 documents. It is also evident that in three of these documents, it is mentioned twice. Another insight is that 'Crawford', an author, is mentioned 19 times, meaning that they have taken part in 19 publications. In conclusion, during this process, all words from the titles, date and authors' lists are individually split and transformed to lowercase, while the index captures the occurrences of individual words and maps them to the corresponding documents.

D. *Ranked Retrieval*

Prior to building the query, it is important to establish a ranking algorithm that ranks the searched results. The primary purpose of this algorithm is to assign a rank or score to the search results, ensuring that the presented outcomes to the user are relevant and tailored to their information needs. The relevance score is calculated using the Term Frequency - Inverse Document Frequency (TF-IDF) scores. The TF method calculates the number of times that a term appears in a document compared to the total number of terms in this document. The IDF method calculates the proportion of the document that this term is included in[1]. Multiplied together they can provide a relevance score for a specific term. This function is applied within the Query Processor to rank the search results based on their relevance to the given search query.

E. *Query Processor*

The final step of the vertical search engine involves the implementation of the query processor. In this function, the user is prompted to input a query, and the engine retrieves relevant publications that match the query. The retrieved publications are then sorted by their relevance scores. The input query undergoes a series of preprocessing steps to ensure consistency in matching. It is split into individual words (tokens), subjected to stemming, and converted to lowercase for case-insensitive comparison. This preprocessing enables accurate matching of the query against the indexed documents. During the iteration of the inverted index, if there are documents that correspond to the searched query, their details (title, authors, and date) are appended to a list. That list contains the title, the authors and the date for any document from the postings list that contains this query. Subsequently, the initial CSV file is 'read' again to retrieve links to the relevant publications and the corresponding authors' 'pureportal' porfiles. These links are saved in a new list that gets included in the final output. With the gathered information, a new Data Frame is created, containing the title, authors, publication date, publication link, and authors' 'pureportal' pages. This Data Frame constitutes the final search results, accurately presenting the relevant publications in a structured format.

F. *Application*

A web-based application was built for the construction of the vertical search engine with the help of Flask. Below is a screenshot of the User Interface. The user can input a query in the search bar in the middle of the page and then click search to get some results. The green button in the middle can take the user to the Centre for Global Learning webpage where all the publications are available.

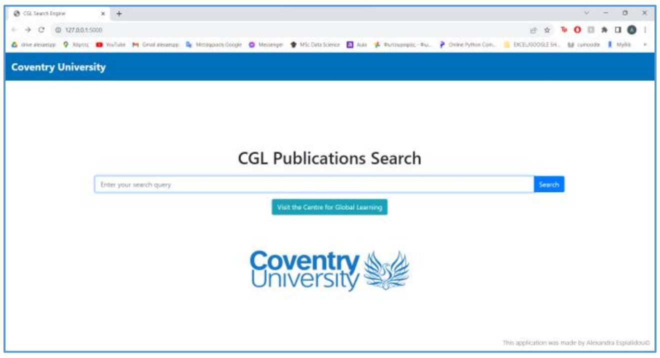
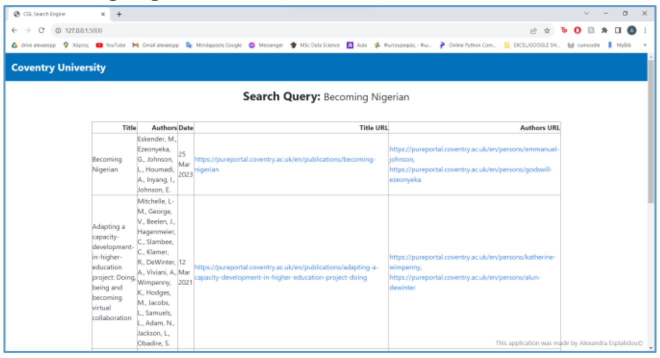

Coventry University
CGL Publications Search
Enter your search query
Search
Visit the Centre for Global Learning
This application was made by Alexandra Ezpialidou

Image 5: User Interface

[1] From https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/#:~:text=As%20its%20name%20implies%2C%20TF,of%20words%20in%20the%20document.

Here is a screenshot where the user has input the query: “Becoming Nigerian” and clicked *Search*.



Coventry University

Search Query: Becoming Nigerian

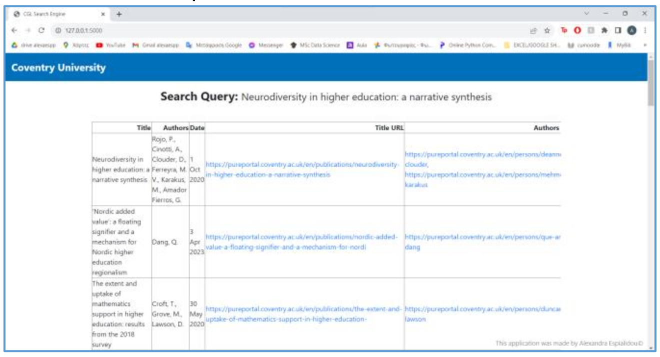
Title	Authors	Date	Title URL	Authors URL
Becoming Nigerian	Eskender, M., Ezeonyeka, G., Johnson, L., Houmadi, A., Inyang, I., Johnson, E.	25 Mar 2023	https://pureportal.coventry.ac.uk/en/publications/becoming-nigerian	https://pureportal.coventry.ac.uk/en/persons/emmanuel-johnson, https://pureportal.coventry.ac.uk/en/persons/godswill-ezeonyeka
Adapting a capacity-development-in-higher-education project: Doing, being and becoming virtual collaboration	Mitchelle, L-M., George, V., Beelen, J., Hagenmeier, C., Slambee, C., Klamer, R., DeWinter, A., Viviani, A., Wimpenny, K., Hodges, M., Jacobs, L., Samuels, L., Adam, N., Jackson, L., Obadire, S.	12 Mar 2021	https://pureportal.coventry.ac.uk/en/publications/adapting-a-capacity-development-in-higher-education-project-doing	https://pureportal.coventry.ac.uk/en/persons/katherine-wimpenny, https://pureportal.coventry.ac.uk/en/persons/alun-dewinter

This application was made by Alexandra Espialidou©

Image 6: Search query: ‘Becoming Nigerian’

By clicking search all the publications containing any of these two words in their title are being fetched. The one with the highest-ranking score is ‘Becoming Nigerian’ and appears first.

Some more examples follow below:



Coventry University

Search Query: Neurodiversity in higher education: a narrative synthesis

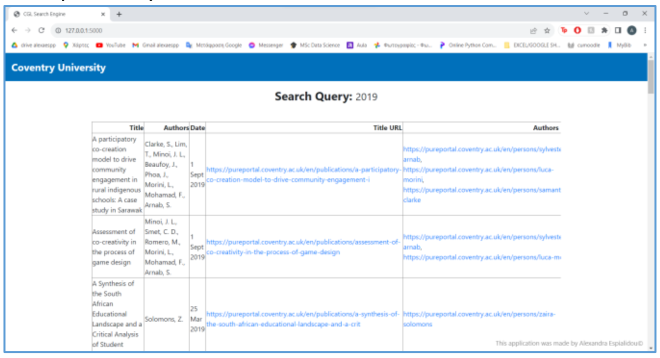
Title	Authors	Date	Title URL	Authors
Neurodiversity in higher education: a narrative synthesis	Rojo, P., Cinotti, A., Clouder, D., Ferreyra, M. V., Karakus, M., Amador Fierros, G.	1 Oct 2020	https://pureportal.coventry.ac.uk/en/publications/neurodiversity-in-higher-education-a-narrative-synthesis	https://pureportal.coventry.ac.uk/en/persons/deann clouder, https://pureportal.coventry.ac.uk/en/persons/mehm karakus
'Nordic added value': a floating signifier and a mechanism for Nordic higher education regionalism	Dang, Q.	3 Apr 2023	https://pureportal.coventry.ac.uk/en/publications/nordic-added-value-a-floating-signifier-and-a-mechanism-for-nordi	https://pureportal.coventry.ac.uk/en/persons/que-ar dang
The extent and uptake of mathematics support in higher education: results from the 2018 survey	Croft, T., Grove, M., Lawson, D.	30 May 2020	https://pureportal.coventry.ac.uk/en/publications/the-extent-and-uptake-of-mathematics-support-in-higher-education-	https://pureportal.coventry.ac.uk/en/persons/duncai lawson

This application was made by Alexandra Espialidou©

Image 7: Search query ‘Neurodiversity in higher education: a narrative synthesis’

A publication with all the query terms is presented first.

Example of a year:



Coventry University

Search Query: 2019

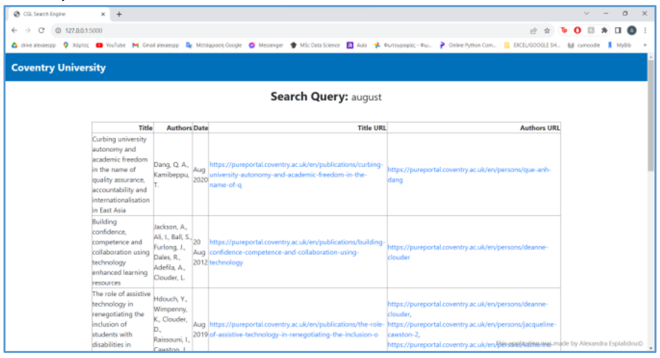
Title	Authors	Date	Title URL	Authors
A participatory co-creation model to drive community engagement in rural indigenous schools: A case study in Sarawak	Clarke, S., Lim, T., Minoi, J. L., Beaufoy, J., Phoa, J., Morini, L., Mohamad, F., Arnab, S.	1 Sept 2019	https://pureportal.coventry.ac.uk/en/publications/a-participatory-co-creation-model-to-drive-community-engagement-i	https://pureportal.coventry.ac.uk/en/persons/sylveste arnab, https://pureportal.coventry.ac.uk/en/persons/luca-morini, https://pureportal.coventry.ac.uk/en/persons/samant clarke
Assessment of co-creativity in the process of game design	Minoi, J. L., Smet, C. D., Romero, M., Morini, L., Mohamad, F., Arnab, S.	1 Sept 2019	https://pureportal.coventry.ac.uk/en/publications/assessment-of-co-creativity-in-the-process-of-game-design	https://pureportal.coventry.ac.uk/en/persons/sylveste arnab, https://pureportal.coventry.ac.uk/en/persons/luca-mi
A Synthesis of the South African Educational Landscape and a Critical Analysis of Student	Solomons, Z.	25 Mar 2019	https://pureportal.coventry.ac.uk/en/publications/a-synthesis-of-the-south-african-educational-landscape-and-a-crit	https://pureportal.coventry.ac.uk/en/persons/zaira-solomons

This application was made by Alexandra Espialidou©

Image 8: Search query ‘2019’

All publication from 2019 are presented to the user.

Example of a month:



Coventry University

Search Query: august

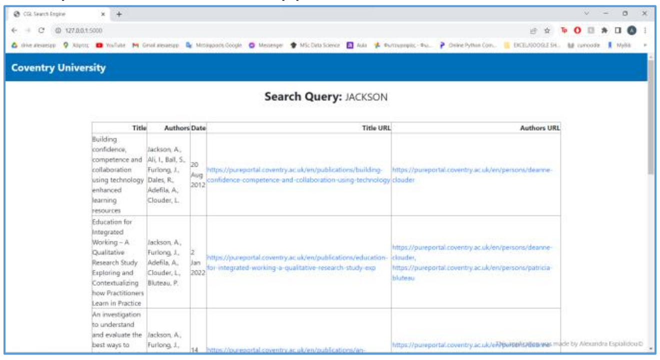
Title	Authors	Date	Title URL	Authors URL
Curbing university autonomy and academic freedom in the name of quality assurance, accountability and internationalisation in East Asia	Dang, Q. A., Kamibeppu, T.	Aug 2020	https://pureportal.coventry.ac.uk/en/publications/curbing-university-autonomy-and-academic-freedom-in-the-name-of-q	https://pureportal.coventry.ac.uk/en/persons/que-anh-dang
Building confidence, competence and collaboration using technology enhanced learning resources	Jackson, A., Ali, I., Ball, S., Furlong, J., Dales, R., Adefila, A., Clouder, L.	20 Aug 2012	https://pureportal.coventry.ac.uk/en/publications/building-confidence-competence-and-collaboration-using-technology	https://pureportal.coventry.ac.uk/en/persons/deanne-clouder
The role of assistive technology in renegotiating the inclusion of students with disabilities in	Hdouch, Y., Wimpenny, K., Clouder, D., Raissouni, I., Cawston, J.	Aug 2019	https://pureportal.coventry.ac.uk/en/publications/the-role-of-assistive-technology-in-renegotiating-the-inclusion-o	https://pureportal.coventry.ac.uk/en/persons/deanne-clouder, https://pureportal.coventry.ac.uk/en/persons/jacqueline-cawston-2, https://pureportal.coventry.ac.uk/en/persons/katherine-

This application was made by Alexandra Espialidou©

Image 9: Search query ‘august’

All papers published in August are presented.

Example of an author in uppercase:



Coventry University

Search Query: JACKSON

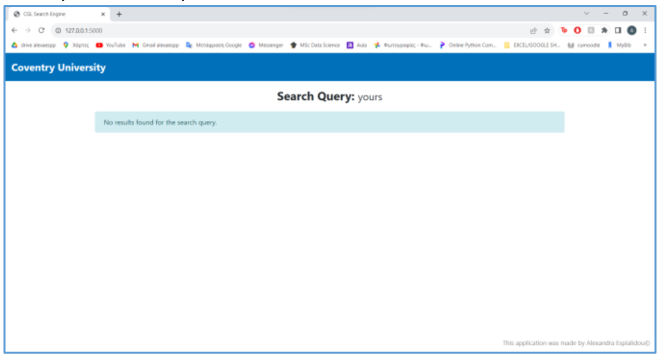
Title	Authors	Date	Title URL	Authors URL
Building confidence, competence and collaboration using technology enhanced learning resources	Jackson, A., Ali, I., Ball, S., Furlong, J., Dales, R., Adefila, A., Clouder, L.	20 Aug 2012	https://pureportal.coventry.ac.uk/en/publications/building-confidence-competence-and-collaboration-using-technology	https://pureportal.coventry.ac.uk/en/persons/deanne-clouder
Education for Integrated Working – A Qualitative Research Study Exploring and Contextualizing how Practitioners Learn in Practice	Jackson, A., Furlong, J., Adefila, A., Clouder, L., Bluteau, P.	2 Jan 2022	https://pureportal.coventry.ac.uk/en/publications/education-for-integrated-working-a-qualitative-research-study-exp	https://pureportal.coventry.ac.uk/en/persons/deanne-clouder, https://pureportal.coventry.ac.uk/en/persons/patricia-bluteau
An investigation to understand and evaluate the best ways to	Jackson, A., Furlong, J.,	14	https://pureportal.coventry.ac.uk/en/publications/an-	https://pureportal.coventry.ac.uk/en/

This application was made by Alexandra Espialidou©

Image 10: Search query ‘JACKSON’

All publications were Jasckon is an author or co-author are shown.

Example of a stopword:



Coventry University

Search Query: yours

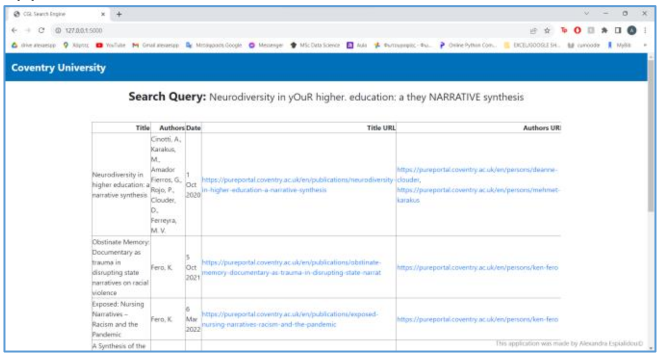
No results found for the search query.

This application was made by Alexandra Espialidou©

Image 11: Search query ‘yours’

There are no results found because all the stop-words have been removed from the database.

Example of a title with stop-words, punctuation and uppercase letters:


Coventry University

Search Query: Neurodiversity in yOuR higher. education: a they NARRATIVE synthesis

Title	Authors	Date	Title URL	Authors URL
Neurodiversity in higher education: a narrative synthesis	Cinotti, A., Karakus, M., Amador Fierros, G., Rojo, P., Clouder, D., Ferreyra, M. V.	1 Oct 2020	https://pureportal.coventry.ac.uk/en/publications/neurodiversity-in-higher-education-a-narrative-synthesis	https://pureportal.coventry.ac.uk/en/persons/deanne-clouder, https://pureportal.coventry.ac.uk/en/persons/mehmet-karakus
Obstinate Memory: Documentary as trauma in disrupting state narratives on racial violence	Fero, K.	5 Oct 2021	https://pureportal.coventry.ac.uk/en/publications/obstinate-memory-documentary-as-trauma-in-disrupting-state-narrat	https://pureportal.coventry.ac.uk/en/persons/ken-fero
Exposed: Nursing Narratives – Racism and the Pandemic	Fero, K.	6 Mar 2022	https://pureportal.coventry.ac.uk/en/publications/exposed-nursing-narratives-racism-and-the-pandemic	https://pureportal.coventry.ac.uk/en/persons/ken-fero
A Synthesis of the				

This application was made by Alexandra Espialidou©

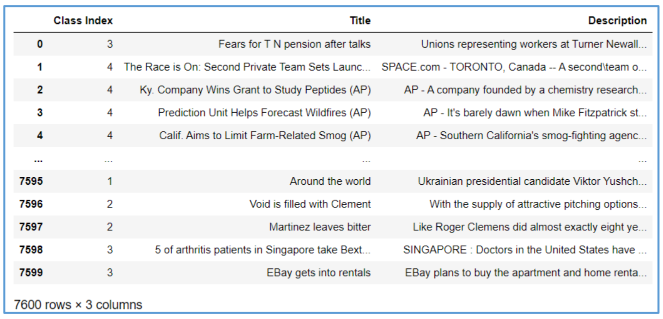
Image 12: Search query 'Neurodiversity in yOuR higher. education: a they NARRATIVE synthesis'

In this example, the first pulled publication is one with all the query terms minus the stop-words.

The vertical search engine presents all the publications where the search query is mentioned along with the Authors, the date, the Publication's URL and the authors' 'pureportal' pages.

II. Task 2. Text Classification

The second problem that this paper focuses on is a system that takes some text as input and gives as an output the class that the text corresponds to. The classes of interest are Sports, Business and Science. To achieve this objective, the system requires a carefully curated dataset as a foundational component. The dataset was sourced from Kaggle.com and was originally curated by Xiang Zhang for News classification purposes. The original folder contained 2 datasets and the one named *test.csv* was used. This dataset comprises 7600 rows and 3 columns, where the first column corresponds to class indices ranging from 1 to 4. The remaining two columns contain the titles and descriptions of news articles collected over the span of a year. The first column contains numerical data and the two other contain categorical data. Class Index 1 is related to *World News*, class Index 2 is related to *Sports*, Class Index 3 is related to *Business* and Class Index 4 is related to *Science*. These news articles are related to Sport, Business, Science and World News. Below is a preview of the dataset in its original state.



	Class Index	Title	Description
0	3	Fears for T N pension after talks	Unions representing workers at Turner Newall...
1	4	The Race is On: Second Private Team Sets Launc...	SPACE.com - TORONTO, Canada -- A second\team o...
2	4	Ky. Company Wins Grant to Study Peptides (AP)	AP - A company founded by a chemistry research...
3	4	Prediction Unit Helps Forecast Wildfires (AP)	AP - It's barely dawn when Mike Fitzpatrick st...
4	4	Calif. Aims to Limit Farm-Related Smog (AP)	AP - Southern California's smog-fighting agenc...
...	...	...	...
7595	1	Around the world	Ukrainian presidential candidate Viktor Yushch...
7596	2	Void is filled with Clement	With the supply of attractive pitching options...
7597	2	Martinez leaves bitter	Like Roger Clemens did almost exactly eight ye...
7598	3	5 of arthritis patients in Singapore take Bext...	SINGAPORE : Doctors in the United States have ...
7599	3	EBay gets into rentals	EBay plans to buy the apartment and home renta...

7600 rows × 3 columns

Image 13: The original dataset

A. *Data Exploration*

Before delving into the implementation of the text classification system, a series of preprocessing steps were conducted to ensure the dataset's readiness and suitability for analysis.

To begin with, the first column, representing the Class Index needed to be moved to the end of the dataset and the instances related to the class index for world news (which is index 1) needed to be discarded, as the system needs to work only for Sports, Business and Science related text. Next, The Class Index attribute underwent a remapping process to establish a new set of labels in a clearer and more intuitive manner. The new labels are 1 – for Sport, 2 – for Business and 3 – for science. In order to facilitate user comprehension of the remapped class labels, an additional column was introduced. The final dataset new contains 5700 instances and 4 columns.

Lastly, it is worth noticing that the dataset contains no null values. The dataset is also balanced as each class index is uniformly represented by 1900 instances. This equitable distribution ensures an unbiased and robust foundation for subsequent model training Below is a histogram depicting the frequency distribution of the 'Sports', 'Business', and 'Science' classes.

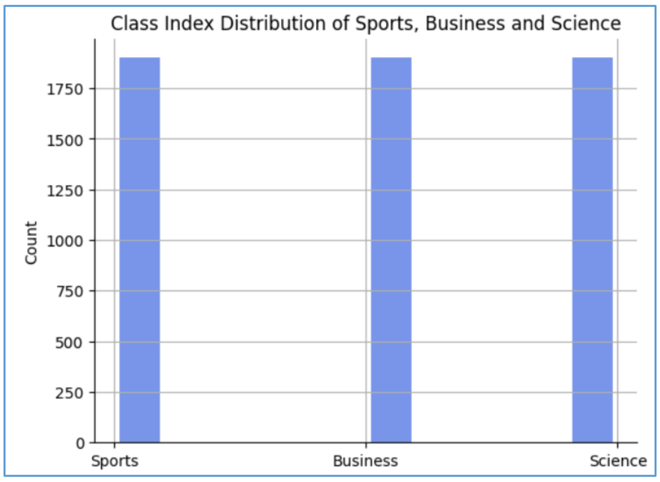


Image 14: Class Index Distribution

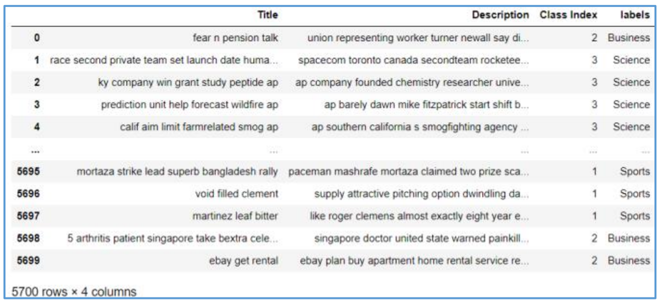
B. *Data Preprocessing*

In pursuit of constructing an effective text classification system, further preprocessing of the dataset was undertaken to optimize its suitability for accurate classification. Particular attention was directed towards the "Title" and "Description" columns, where certain modifications were applied. The primary focus of this step was the expansion of contractions and the replacement of specific symbols with their corresponding word forms. Presented below is a detailed table showing the symbols that underwent replacement:

%	percent
000.000	m
000	k

₹	rupee
$	dollar
€	euro
’	'
’	'
won't	will not
can’t	can not
shouldn’t	should not
what’s	what is
that’s	that is
he’s	he is
she’s	she is
it’s	it is
‘ve	have
‘re	are
‘ll	will
i’m	i am
n’t	not

Furthermore, URLs and HTML tags were systematically eliminated, ensuring the dataset's cleanliness and focus on essential textual content. Stop-words and punctuations were also removed and a lemmatization process was applied, whereby all words were reduced to their base or root form. For example, the word ‘running’ was reduced to its root form: ‘run’. Presented below is a snapshot of the dataset in its finalized state, prepared for utilization in the text classification system.



	Title	Description	Class Index	labels
0	fear n pension talk	union representing worker turner newall say di...	2	Business
1	race second private team set launch date huma...	spacecom toronto canada secondteam rocketee...	3	Science
2	ky company win grant study peptide ap	ap company founded chemistry researcher unive...	3	Science
3	prediction unit help forecast wildfire ap	ap barely dawn mike fitzpatrick start shift b...	3	Science
4	calif aim limit farmrelated smog ap	ap southern california s smogfighting agency ...	3	Science
...	...	...	...	...
5695	mortaza strike lead superb bangladesh rally	paceman mashrafe mortaza claimed two prize sca...	1	Sports
5696	void filled clement	supply attractive pitching option dwindling da...	1	Sports
5697	martinez leaf bitter	like roger clemens almost exactly eight year e...	1	Sports
5698	5 arthritis patient singapore take bextra cele...	singapore doctor united state warned painkill...	2	Business
5699	ebay get rental	ebay plan buy apartment home rental service re...	2	Business

5700 rows × 4 columns

Image 15: Final state of the dataset

To advance from textual data to a format suitable for machine learning models, the categorical data has been transformed into the vector space. This essential transformation was achieved using the TfidfVectorizer function, harnessed from the sklearn library.

C. *Naïve Bayes*

The Naïve Bayesian Classifier is a supervised machine learning algorithm used for classification purposes. It works as a probabilistic classifier and is ‘naïve’ by assuming the independence of each input variable [2]. This function operates by computing the probability of each class, in this case 3 classes, and the conditional probability of each class, given every different x value. The formula is:

$P(A|B) = P(B|A)* P(A) / P(B)$,

Where $P(A|B)$ is the probability of a class, $P(B|A)$ is the probability of the data given that it is that class, $P(A)$ is the prior probability, meaning before observing the data and $P(B)$ is the probability of the evidence[3].

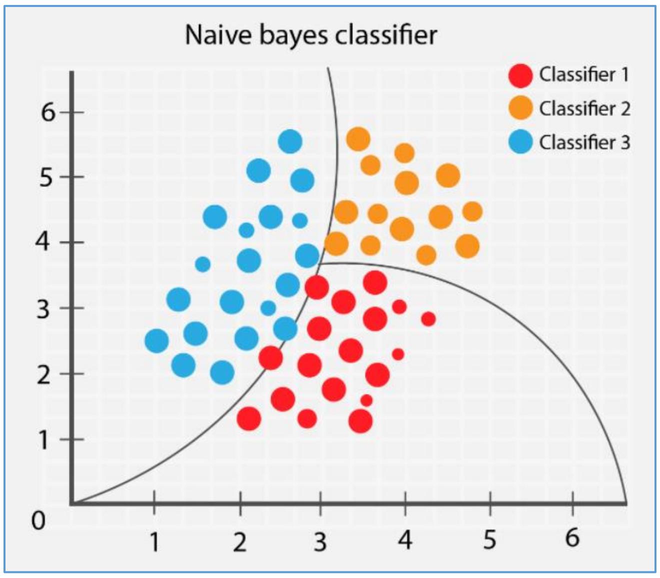


Image 16: sourced from https://kdagiit.medium.com/naive-bayes-algorithm-4b8b990c7319

The above diagram is a representation of the Naïve Bayes Classifier. This classifier is ideal for a text classification system as it quite simple and can handle sparse data well. For that reason, it reduces the risk of overfitting as it calculates the probability of individual presents being present in each class and does not consider their specific order. Consequently, when the Naïve Bayes classifier is paired with the TF-IDF vectorizer, the occurrence of a word multiple times amplifies its likelihood of being associated with a particular class.

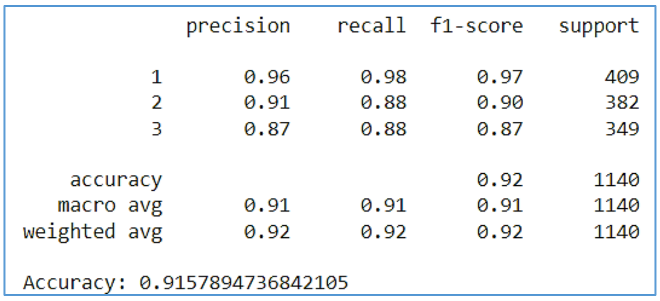
The dataset was partitioned into separate training and testing sets and the MultinomialNB() function was employed from the sklearn library.

D. *Model Evaluation*

Below is the classification report for the Naïve Bayes model.

[2] From https://www.sciencedirect.com/topics/mathematics/naive-bayes#:~:text=Naive%20Bayes%20is%20called%20naive,large%20range%20of%20complex%20problems.

[3] From https://www.analyticsvidhya.com/blog/2022/03/building-naive-bayes-classifier-from-scratch-to-perform-sentiment-analysis/



```
              precision    recall  f1-score   support

           1       0.96      0.98      0.97       409
           2       0.91      0.88      0.90       382
           3       0.87      0.88      0.87       349

    accuracy                           0.92      1140
   macro avg       0.91      0.91      0.91      1140
weighted avg       0.92      0.92      0.92      1140

Accuracy: 0.9157894736842105
```

Image 17: Classification accuracy report

From the above picture it is evident that the Naïve Bayes classifier is compatible with the model of the available dataset as it gives a very high result with an accuracy of 92%. The classification report gives some key observations of 7 metrics. Specifically, *precision* or Positive Predicted Value calculates the number of True Positives instances (TP) out of all the instances that were predicted from the model. *Recall*, which is also known as sensitivity, gives the percentage of True Positive instances out of all the actual positive instances. F1-score calculates the mean of precision and recall for all 3 classes. *Support* is the distribution of all the instances that were predicted for each class. For example, 409 instances were predicted as class 1 ('Sports'), 382 as class 2 ('Business') and 349 as class 3 ('Science') out of 1140. *Macro Avg* computes the average value of precision, recall and F1-score for each class and *weighted avg* determines the weighted average respectively. *Accuracy* measures the number of all True Positives and True Negatives, which ultimately gives the accuracy score of the model.

All the evaluation metrics presented above demonstrate high scores, providing compelling evidence of the model's effectiveness.

E. *Graphical User Interface*

To ensure user-friendly accessibility, a graphical user interface (GUI) application was developed, leveraging the capabilities of the tkinter library. This GUI interface serves as the frontend of the text classification system, enabling effortless interaction with the model for all users.

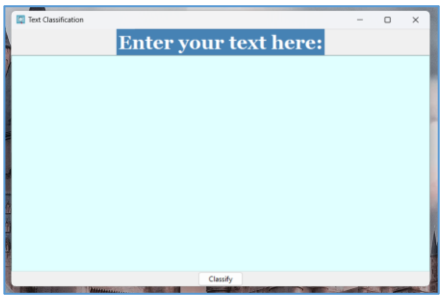
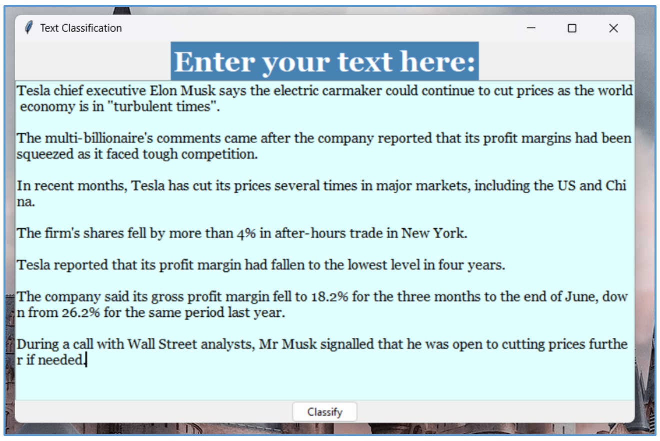


Image 18: User Interface for Text Classification

Upon executing the cell, a user-friendly window promptly opens, presenting an interface through which the user can input their desired text. In this experiment the following text was input:



Tesla chief executive Elon Musk says the electric carmaker could continue to cut prices as the world economy is in "turbulent times".

The multi-billionaire's comments came after the company reported that its profit margins had been squeezed as it faced tough competition.

In recent months, Tesla has cut its prices several times in major markets, including the US and China.

The firm's shares fell by more than 4% in after-hours trade in New York.

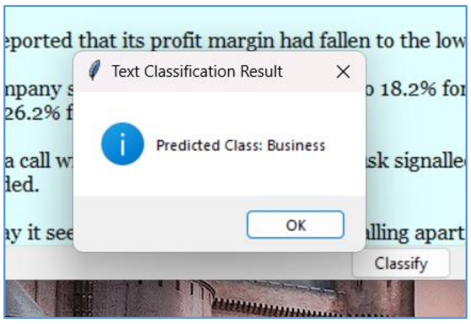
Tesla reported that its profit margin had fallen to the lowest level in four years.

The company said its gross profit margin fell to 18.2% for the three months to the end of June, down from 26.2% for the same period last year.

During a call with Wall Street analysts, Mr Musk signalled that he was open to cutting prices further if needed.

Image 19: Example of input text

Following the user's input of the text, the subsequent step involves pressing the ***Classify*** button. This action triggers the text classification system to process the entered text and perform the categorization.

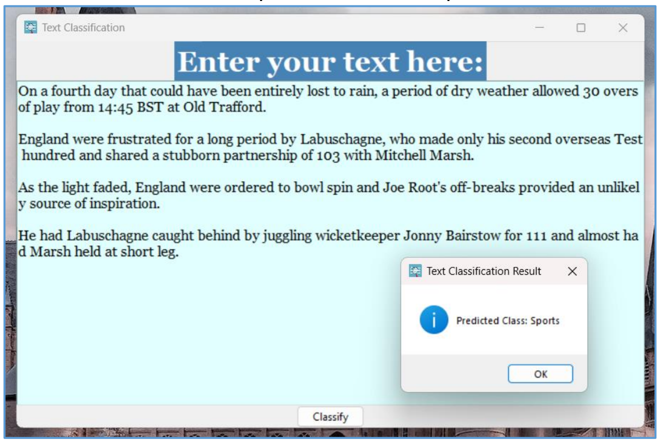


Predicted Class: Business

Image 20: Prediction

Upon pressing the ***Classify*** button, a new window promptly emerges, revealing the predicted class for the specific text inputted by the user. This display serves as an immediate and visually accessible representation of the model's classification outcome. In this case the text was taken from https://www.bbc.co.uk/news/business-66253044, and the predicted class is correctly predicted as ***Business***.

Below are 5 more examples and their respected results.



On a fourth day that could have been entirely lost to rain, a period of dry weather allowed 30 overs of play from 14:45 BST at Old Trafford.

England were frustrated for a long period by Labuschagne, who made only his second overseas Test hundred and shared a stubborn partnership of 103 with Mitchell Marsh.

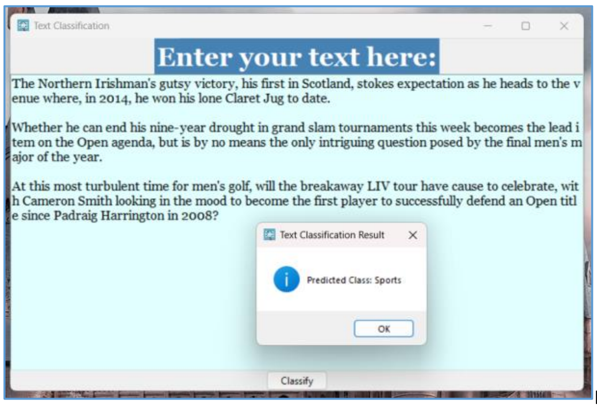
As the light faded, England were ordered to bowl spin and Joe Root's off-breaks provided an unlikely source of inspiration.

He had Labuschagne caught behind by juggling wicketkeeper Jonny Bairstow for 111 and almost had Marsh held at short leg.

Predicted Class: Sports

Image 21: Example about Sports

This text was taken from https://www.bbc.co.uk/sport/cricket/66272616 and was correctly classified as Sports.



The Northern Irishman's gutsy victory, his first in Scotland, stokes expectation as he heads to the venue where, in 2014, he won his lone Claret Jug to date.

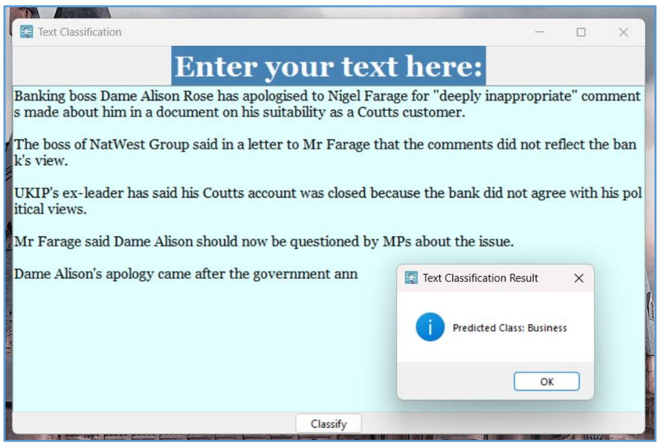
Whether he can end his nine-year drought in grand slam tournaments this week becomes the lead item on the Open agenda, but is by no means the only intriguing question posed by the final men's major of the year.

At this most turbulent time for men's golf, will the breakaway LIV tour have cause to celebrate, with Cameron Smith looking in the mood to become the first player to successfully defend an Open title since Padraig Harrington in 2008?

Predicted Class: Sports

Image 22: Example about Sports

This text was taken from https://www.bbc.co.uk/sport/golf/66218743 and was correctly classified as Sports.



Banking boss Dame Alison Rose has apologised to Nigel Farage for "deeply inappropriate" comments made about him in a document on his suitability as a Coutts customer.

The boss of NatWest Group said in a letter to Mr Farage that the comments did not reflect the bank's view.

UKIP's ex-leader has said his Coutts account was closed because the bank did not agree with his political views.

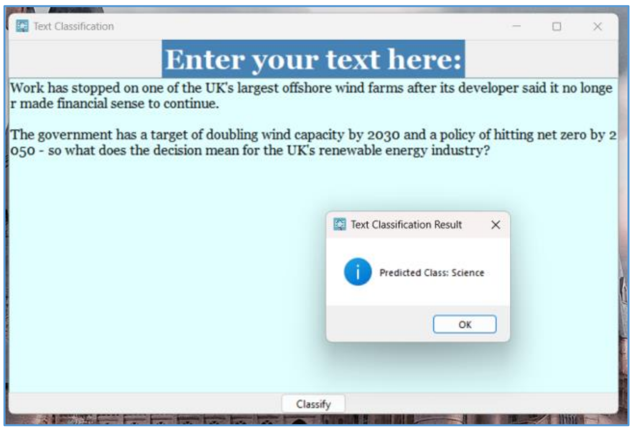
Mr Farage said Dame Alison should now be questioned by MPs about the issue.

Dame Alison's apology came after the government ann

Predicted Class: Business

Image 23: Example about Business

This text was taken from https://www.bbc.co.uk/news/business-66258137 and was correctly classified as Business.



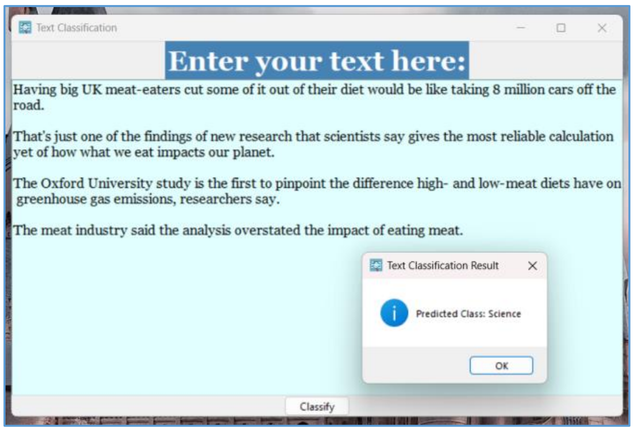
Work has stopped on one of the UK's largest offshore wind farms after its developer said it no longer made financial sense to continue.

The government has a target of doubling wind capacity by 2030 and a policy of hitting net zero by 2050 - so what does the decision mean for the UK's renewable energy industry?

Predicted Class: Science

Image 24: Example about science

This text was taken from https://www.bbc.co.uk/news/uk-england-norfolk-66267860 and was correctly classified as science.



Having big UK meat-eaters cut some of it out of their diet would be like taking 8 million cars off the road.

That's just one of the findings of new research that scientists say gives the most reliable calculation yet of how what we eat impacts our planet.

The Oxford University study is the first to pinpoint the difference high- and low-meat diets have on greenhouse gas emissions, researchers say.

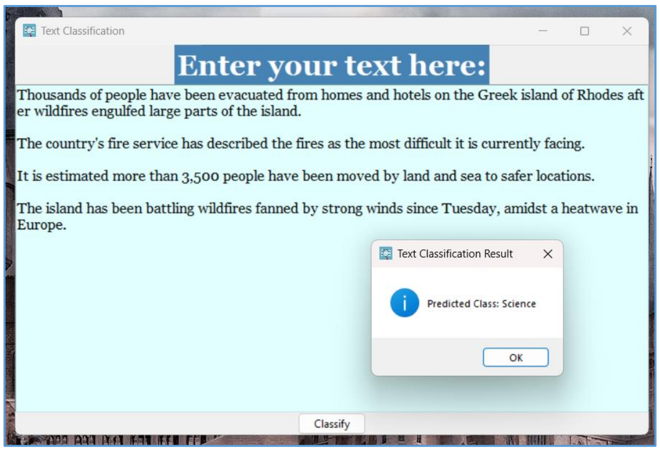
The meat industry said the analysis overstated the impact of eating meat.

Predicted Class: Science

Image 25: Example about science

This text was taken from https://www.bbc.co.uk/news/science-environment-66238584 and was correctly classified as science.

Lastly below are 2 texts non-related to either Sports, Business or Science.



Thousands of people have been evacuated from homes and hotels on the Greek island of Rhodes after wildfires engulfed large parts of the island.

The country's fire service has described the fires as the most difficult it is currently facing.

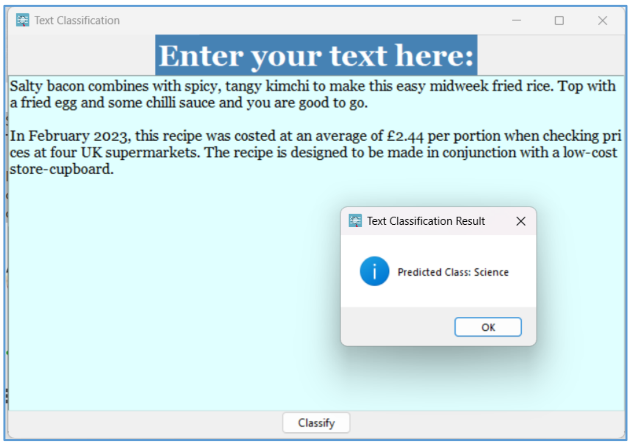
It is estimated more than 3,500 people have been moved by land and sea to safer locations.

The island has been battling wildfires fanned by strong winds since Tuesday, amidst a heatwave in Europe.

Predicted Class: Science

Image 26: Non-related example

This text was taken from https://www.bbc.co.uk/news/world-europe-66279520 and was classified as science.



Salty bacon combines with spicy, tangy kimchi to make this easy midweek fried rice. Top with a fried egg and some chilli sauce and you are good to go.

In February 2023, this recipe was costed at an average of £2.44 per portion when checking prices at four UK supermarkets. The recipe is designed to be made in conjunction with a low-cost store-cupboard.

Predicted Class: Science

Image 27: Non-related example

This text was taken from https://www.bbc.co.uk/food/recipes/bacon_and_kimchi_fried_71162 and was classified as science.

It is interesting to notice that both non-related texts were classified as science. The reason for this could be that some of the words might overlap with those commonly found in science-related texts. Another reason could be that the training data lack diversity in non-related text sample and it could result in a bias towards science.

III. References

[1] Anand, A. (2020). *AG News Classification Dataset*. [online] www.kaggle.com. Available at: https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset?resource=download [Accessed 22 Jul. 2023].

[2] BBC News (2021). *Home - BBC News*. [online] BBC News. Available at: https://www.bbc.co.uk/news.

[3] Clear Code (2023). *The ultimate introduction to modern GUIs in Python [with tkinter]*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=mop6g-c5HEY&t=1714s [Accessed 22 Jul. 2023].

[4] freecodecamp.com (2019). *Learn Flask for Python - Full Tutorial*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=Z1RJmh_OqeA&t=518s [Accessed 31 Jul. 2023].

[5] Koushiki Dasgupta Chaudhuri (2022). *Building Naive Bayes Classifier from Scratch to Perform Sentiment Analysis*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2022/03/building-naive-bayes-classifier-from-scratch-to-perform-sentiment-analysis/.

[6] Nihar Max (2023). *News Classification using NLP* . [online] GitHub. Available at: https://github.com/nihar-max/nlp_news_classification/blob/main/News%20Classification%20using%20NLP%20part%202.ipynb [Accessed 22 Jul. 2023].

[7] Sarangi, P.P., Panda, M., Mishra, S., Mishra, B.S.P. and Majhi, B.D. (2022). *Machine learning for biometrics : concepts, algorithms and applications*. Amsterdam: Academic Press.

IV. Appendix

Task 1- Vertical Search Engine

```
#Imports
from flask import Flask, render_template, request
import requests
import csv
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
import string
import nltk
import datetime
import time
nltk.download("stop-words")
from nltk.corpus import stop-words
from collections import defaultdict
import math
```

```python
import pandas as pd
from IPython.display import display, HTML

#Creating the app
app = Flask(__name__)
#Crawler
def crawler():
    coventry_url = "https://pureportal.coventry.ac.uk/en/organisations/centre-global-learning/publications/"
    all_data = []
    data = []
    for page_num in range(0, 10):
        url = coventry_url + f"?page={page_num}"
        page = requests.get(url)
        soup = BeautifulSoup(page.text, "html.parser")
        for i in range(55):
            classes = f"list-result-item list-result-item-{i}"
            results = soup.find_all('li', class_=classes)
            for result in results:
            #Name and links of authors
                author_names = []
                author_links = []
                all_authors = result.find_all("a", class_="link person", attrs={'span': ''})
                for author in all_authors:
                    author_names.append(author.text.strip())
                    author_links.append(author['href'])
            #Authors with no link
                non_cov_auth_names = []
                span_nolink_names = result.find_all('span')
                for name in span_nolink_names:
                    if name.string and ', ' in name.string:
                        author_name = name.string.strip()
                        if len(author_name) <= 20:
                             if not any(author_name in ad for ad in author_names):
                                non_cov_auth_names.append(author_name)

            #Title and link of publications
                titles =result.find('h3', class_='title')
                title_link=titles.a
                title =title_link.span.text.strip()
                title_url =title_link['href']
            #Date of publication
                dates= result.find('span', class_='date')
                date = dates.text.strip()
                all_authors = list(set(author_names + non_cov_auth_names))
                all_data.append([", ".join(all_authors), ",  ".join(author_links),
                                  title, title_url, date])
    filename = "important_data.csv"
    with open(filename, 'w', newline='', encoding='utf-8') as file:
        writer=csv.writer(file)
        writer.writerow(["Authors", "Authors_link", "Title", "Title_link", "Date"])
        writer.writerows(all_data)
    #print(f"Imported data were saved to {filename} successfully.")
    #Removal of null rows
    with open('important_data.csv', 'r', newline='', encoding='utf-8') as file:
        reader = csv.DictReader(file)
        for row in reader:
            data.append(row)
    filtered_data = [row for row in data if all(value for value in row.values())]
    with open('important_data.csv', 'w', newline='', encoding='utf-8') as file:
            writer = csv.DictWriter(file, fieldnames=reader.fieldnames)
            writer.writeheader()
            writer.writerows(filtered_data)
#%time crawler()
#Scheduling the crawler to crawl data once a week
interval = 7
days = 0
while days <= 1:
    crawler()
    time.sleep(interval)
    days = days + 1
def update_date_format(date):
    month_mapping = {
        "Jan": "January",
        "Feb": "February",
        "Mar": "March",
        "Apr": "April",
        "May": "May",
        "Jun": "June",
        "Jul": "July",
        "Aug": "August",
        "Sep": "September",
        "Oct": "October",
        "Nov": "November",
        "Dec": "December"}
    for month_abbr, month_full in month_mapping.items():
            if month_abbr in date:
                date = date.replace(month_abbr, month_full)
    return date
punc = set(string.punctuation)
filename = "important_data.csv"
documents = []
sw = stop-words.words('english')
with open(filename, 'r', newline='', encoding='utf-8') as file:
    reader = csv.reader(file)
    headers = next(reader)
    for row in reader:
        authors = [author.strip() for author in row[0].split(',')]
        title = row[2]
        date = row[4]
        documents.append((authors, title, date))
```

```python
#Tokenization, removal of punctuation, stop-words, stemming
ps = PorterStemmer()
ready_docs = []
for doc in documents:
    authors, title, date = doc
    date = update_date_format(date)
    ready_authors = []
    for author in authors:
        author_tokens = word_tokenize(author.lower())
        ready_author = " ".join([ps.stem(w) for w in author_tokens if w
                                 not in punc and w not in sw])
        ready_authors.append(ready_author)
    title_tokens = word_tokenize(title.lower())
    ready_title = " ".join([ps.stem(w) for w in title_tokens if w not in
                             punc and w not in sw])
    ready_date = word_tokenize(date.lower())
    ready_doc = (ready_authors, ready_title, ready_date)
    ready_docs.append(ready_doc)
#print(ready_docs)
#Inverted Index
index = defaultdict(list)
for post_list, (authors, title, date) in enumerate(ready_docs, start=1):
    tokens = title.split()
    for author in authors:
        tokens.extend(author.split())
    tokens.extend(date)
    for token in tokens:
        index[token].append(post_list)
#print("Inverted Index:")
#for token, post_lists in index.items():
#    print(f"{token}: {post_lists}")
stemmer = PorterStemmer()
def preprocess_query(query):
    query_terms = query.lower().split()
    stemmed_query = [stemmer.stem(term) for term in query_terms]
    return ' '.join(stemmed_query)
search_term = input("Enter your query here: ")
search_term = preprocess_query(search_term)
results = []
search_results = []
#Ranking of documents based on TF-IDF
def calculate_idf(term, index, total_documents):
    doc_count_with_term = len(index.get(term, []))
    return math.log(total_documents / (doc_count_with_term + 1))
def calculate_tfidf_scores(query, index, documents):
    tfidf_scores = {}
    query_terms = query.lower().split()
    total_documents = len(documents)
    for term in query_terms:
        tf = query_terms.count(term)
        idf = calculate_idf(term, index, total_documents)
        tfidf_scores[term] = tf * idf
    return tfidf_scores
tfidf_scores = calculate_tfidf_scores(search_term, index, documents)
def calculate_relevance_score(doc_id, tfidf_scores, index):
    score = 0.0
    for term, tfidf_score in tfidf_scores.items():
        if doc_id in index.get(term, []):
            score += tfidf_score
    return round(score, 2)
#Query processor
for doc_id in set(doc_id for term in search_term.lower().split()
                  for doc_id in index.get(term, [])):
    authors, title, date = documents[doc_id - 1]
    result_id = (tuple(authors), tuple(title), date)
    relevance_score = calculate_relevance_score(doc_id, tfidf_scores, index)
    search_results.append((doc_id, relevance_score))
    results.append((authors, title, date))
ranked_results = sorted(search_results, key=lambda x: x[1], reverse=True)
#Creating a dataframe to store the search results
def dataframe(results):
    df = pd.DataFrame(results, columns=[
        "Title",
        "Authors",
        "Date",
        "Title URL",
        "Authors URL"])
    return df
urls_dict = {}
authors_urls_dict = {}
with open(filename, 'r', newline='', encoding='utf-8') as file:
    reader = csv.reader(file)
    headers = next(reader)
    for row in reader:
        title = row[2]
        title_url = row[3]
        authors = tuple(row[0].split(", "))
        authors_url = row[1]
        urls_dict[title] = title_url
        authors_urls_dict[authors] = authors_url
for result in results:
    authors = tuple(result[0])
    authors_list = ', '.join(authors)
    title = result[1]
    date = result[2]
    title_url = urls_dict.get(title, "URL not available")
    authors_url = authors_urls_dict.get(authors, "URL not available")

    for doc_id in ranked_results:
        search_result = (
```

```python
                title,
                authors_list,
                date,
                title_url,
                authors_url)
            search_results.append(search_result)
search_results_df = dataframe(search_results)
search_results_df.dropna(subset=["Title URL", "Authors URL"], inplace=True)
search_results_df.drop_duplicates(subset=["Title"], keep="first", inplace=True)
search_results_df.reset_index(drop=True, inplace=True)
#Making the URL's clickable
def make_clickable(urls):
    clickable_links = []
    for url in urls.split(", "):
        clickable_links.append(f'<a href="{url}" target="_blank">{url}</a>')
    return ', '.join(clickable_links)
search_results_df["Title URL"] = search_results_df["Title URL"].apply(make_clickable)
search_results_df["Authors URL"] = search_results_df["Authors URL"].apply(make_clickable)
search_results_df_html = search_results_df.to_html(escape=False, index=False)
display(HTML(search_results_df_html))

#Defining the route of the app
@app.route('/', methods=['GET', 'POST'])
def search_page():
    if request.method == 'POST':
        search_query = request.form.get('search_query', '').strip()
        if search_query:
            search_term = preprocess_query(search_query)
            results = []
            search_results = []
            tfidf_scores = calculate_tfidf_scores(search_term, index, documents)
            for doc_id in set(doc_id for term in search_term.lower().split()
                              for doc_id in index.get(term, [])):
                authors, title, date = documents[doc_id - 1]
                result_id = (tuple(authors), tuple(title), date)
                relevance_score = calculate_relevance_score(doc_id, tfidf_scores, index)
                search_results.append((doc_id, relevance_score))
                results.append((authors, title, date))
            ranked_results = sorted(search_results, key=lambda x: x[1], reverse=True)
            def dataframe(results):
                df = pd.DataFrame(results, columns=[
                    "Title",
                    "Authors",
                    "Date",
                    "Title URL",
                    "Authors URL"])
                return df
            urls_dict = {}
            authors_urls_dict = {}
            with open(filename, 'r', newline='', encoding='utf-8') as file:
                reader = csv.reader(file)
                headers = next(reader)
                for row in reader:
                    title = row[2]
                    title_url = row[3]
                    authors = tuple(row[0].split(", "))
                    authors_url = row[1]
                    urls_dict[title] = title_url
                    authors_urls_dict[authors] = authors_url
            for result in results:
                authors = tuple(result[0])
                authors_list = ', '.join(authors)
                title = result[1]
                date = result[2]
                title_url = urls_dict.get(title, "URL not available")
                authors_url = authors_urls_dict.get(authors, "URL not available")
                for doc_id in ranked_results:
                    search_result = (
                        title,
                        authors_list,
                        date,
                        title_url,
                        authors_url)
                    search_results.append(search_result)
            search_results_df = dataframe(search_results)
            search_results_df.dropna(subset=["Title URL", "Authors URL"],
                                      inplace=True)
            search_results_df.drop_duplicates(subset=["Title"], keep="first",
                                               inplace=True)
            search_results_df.reset_index(drop=True, inplace=True)
            def make_clickable(urls):
                clickable_links = []
                for url in urls.split(", "):
                    clickable_links.append(f'<a href="{url}" target="_blank">{url}</a>')
                return ', '.join(clickable_links)

            search_results_df["Title URL"] = search_results_df["Title URL"].apply(make_clickable)
            search_results_df["Authors URL"] = search_results_df["Authors URL"].apply(make_clickable)
            search_results_df_html = search_results_df.to_html(escape=False, index=False)
            if not search_results_df.empty:
                search_results_df_html = search_results_df.to_html(escape=False, index=False)
            else:
                 search_results_df_html = None

            return render_template('results.html', search_query=search_query,
                                   search_results=search_results_df_html)
    return render_template('search.html')
#This script is being run as the main program
if __name__ == '__main__':
    app.run(debug=True)
```

```html
<!--HTML structure of the search.html page, where the user can input a search query-->
<!DOCTYPE html>
<html>
<head>
    <title>CGL Search Engine</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        body {
            padding-top: 200px;
            position: relative;
            min-height: 100vh; }
        .header {
            position: fixed;
            top: 0;
            left: 0;
            width: 100%;
            background-color: #0070BB;
            padding: 15px;
            text-align: left;
            color: white;
            font-size: 24px;
            font-weight: bold; }
        h1 {
            margin-bottom: 20px; }
        form {
            margin-top: 20px; }
        .search-image {
            max-width: 400px;
            margin-top: 20px; }
        .footer-text {
            position: absolute;
            bottom: 10px;
            right: 10px;
            font-size: 15px;
            color: #888; }
    </style>
</head>
<body>
    <div class="header">Coventry University</div>
    <div class="container mt-4">
        <h1 class="text-center">CGL Publications Search</h1>
        <!-- Search bar -->
        <form method="POST" action="/">
            <div class="input-group mb-3">
                <input type="text" name="search_query" class="form-control" placeholder="Enter your search query">
                <div class="input-group-append">
                    <button type="submit" class="btn btn-primary">Search</button>
                </div>
            </div>
        </form>
        <!-- CGL button -->
        <div class="text-center">
            <a href="https://pureportal.coventry.ac.uk/en/organisations/centre-global-learning"
class="btn btn-info"
             target="_blank">Visit the Centre for Global Learning </a>
        </div>
        <!-- Coventry University logo -->
        <div class="text-center">
            <img src="https://ww1.freelogovectors.net/wp-content/uploads/2023/04/coventry_university_logo-
freelogovectors.net_-640x360.png?lossy=1&ssl=1&fit=640%2C360"
            alt="Coventry University logo" class="search-image">
        </div>
        <div class="footer-text">This application was made by Alexandra Espialidou©</div>
    </div>
</body>
</html>

<!--HTML structure of the search results.html page, where the user can see the results of their query-->
<!DOCTYPE html>
<html>
<head>
    <title>CGL Search Engine</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .header {
            position: fixed;
            top: 0;
            left: 0;
            width: 100%;
            background-color: #0070BB;
            padding: 15px;
            text-align: left;
            color: white;
            font-size: 24px;
            font-weight: bold;}
        .footer-text {
            position: fixed;
            bottom: 10px;
            right: 10px;
            font-size: 15px;
            color: #888; }
        .search-query-text {
            font-size: 28px; }
        .search-query-data {
            font-size: 24px;
            font-weight: normal; }
    </style>
</head>
```

```
<body>
    <div class="header">Coventry University</div>
    <div class="container mt-4">
        <h1 class="text-center">Search Results</h1>
        <!-- Search query -->
        <div class="text-center mb-3">
            <strong class="search-query-text">Search Query: </strong>
            <span class="search-query-data">{{ search_query }}</span>
        </div>
        <!-- Search results -->
        <div class="table-responsive">
            {% if search_results %}
                <table class="table table-bordered table-hover table-layout-fixed">
                    <tbody><h2>
                        {{ search_results | safe }}
                    </h2>
                </tbody>
                </table>
            {% else %}
                <div class="alert alert-info">
                    No results found for the search query.
                </div>
            {% endif %}
        </div>
    </div>
    <!-- Footer -->
    <div class="footer-text">This application was made by Alexandra Espialidou©</div>
</body>
</html>
```

Task 2 – Text Classification

```
#Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
import string
from nltk.corpus import stop-words
from wordcloud import WordCloud, STOP-WORDS
nltk.download('stop-words')
nltk.download('wordnet')
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import tkinter as tk
from tkinter import ttk, font
from tkinter import messagebox
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score


#Importing the dataset
data = pd.read_csv('test.csv')
#Move the Class Index column to the end
first_column_name = data.columns[0]
first_column = data[first_column_name]
data = data.drop(columns=[first_column_name])
data[first_column_name] = first_column
#Remove unwanted data
data = data[data['Class Index'] != 1]
# 2- Sports, 3- Business, 4- Science
#Remap the Class Index values
columns_to_replace = ['Class Index']
new_values = {2: 1, 3: 2, 4: 3}
for column in columns_to_replace:
    data.loc[:, column] = data.loc[:, column].replace(new_values)
#Insert a new column for Labels
labels = {1:'Sports',2:'Business',3:'Science'}
data['Labels'] = data['Class Index'].map(labels)
#Checking for Class Imbalance with a histogram
plt.hist(data['Class Index'], color='royalblue', alpha=0.7, rwidth=0.8)
plt.ylabel('Count')
plt.title('Class Index Distribution of Sports, Business and Science')
plt.grid()
plt.xticks([1, 2, 3], ['Sports', 'Business', 'Science'])
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.show()
data.Labels.value_counts()
data.isnull().sum()

#Text preprocessing (Expanding contractions, removing of URLs, html tags, punctuations, stop-words and
performing lemmatizing)
data['Description'] = data['Description'].apply(lambda x: str(x).lower()
                                        .replace('%',' percent').replace('₹',' rupee').replace('$','
dollar').replace('€',' euro')\

.replace(',000,000','m').replace('000','k').replace('’',"'").replace("′","'")\
                                        .replace("won't","will not").replace("can't",'can
not').replace("shouldn't","should not")\
                                        .replace("what's",'"what is"').replace("that's",'that
is').replace("he's","he is")\
                                        .replace("she's","she is").replace("it's","it
is").replace("'ve"," have").replace("'re"," are")\
                                        .replace("'ll"," will").replace("i'm","i am").replace("n't", "
not"))

data['Title'] = data['Title'].apply(lambda x: str(x).lower()
                                .replace('%',' percent').replace('₹',' rupee').replace('$','
dollar').replace('€',' euro')\

.replace(',000,000','m').replace('000','k').replace('’',"'").replace("′","'")\
                                .replace("won't","will not").replace("can't",'can
not').replace("shouldn't","should not")\
                                .replace("what's",'"what is"').replace("that's",'that
is').replace("he's","he is")\
                                .replace("she's","she is").replace("it's","it is").replace("'ve","
have").replace("'re"," are")\
                                .replace("'ll"," will").replace("i'm","i am").replace("n't", " not"))

data['Description'] = data['Description'].apply(lambda x: re.compile(r'https?://\S+|www\.\S+').sub(r'',
str(x)))
data['Title'] = data['Title'].apply(lambda x: re.compile(r'https?://\S+|www\.\S+').sub(r'', str(x)))

data['Description'] = data['Description'].apply(lambda x: re.compile('<.*?>').sub(r'', str(x)))
data['Title'] = data['Title'].apply(lambda x: re.compile('<.*?>').sub(r'', str(x)))

lemmatizer = WordNetLemmatizer()
sw = nltk.corpus.stop-words.words('english')
columns_to_preprocess = ['Title', 'Description']

for column in columns_to_preprocess:
    cells = data[column].tolist()
    ready_cells = []
    for cell in cells:
        tokens = word_tokenize(str(cell))
        tmp = ""
        for i in tokens:
            if i not in sw:
                tmp += lemmatizer.lemmatize(i) + " "
                for j in string.punctuation:
                    tmp = tmp.replace(j, '')
        ready_cells.append(tmp)
    data[column] = ready_cells
#Moving to the vector space
vectorized_data = data['Title'] + ' ' + data['Description']
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(vectorized_data)
print(X.todense())
print(X)


#Naive Bayes Classifier
#Splitting the data
y= data['Class Index']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Training the data
model = MultinomialNB()
model.fit(X_train, y_train)

#Making predictions and Accuracy report
predictions=model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(classification_report(y_test, predictions))
print('Accuracy:', accuracy)

#Graphical User Interface
def text_classification():
    ''' A function that takes an input text and classifies it as Sports, Business or Science.'''
    input_text =text_input.get("1.0", "end-1c")
    vectorized_text =vectorizer.transform([input_text])
    predicted_text=model.predict(vectorized_text)
```

```python
    class_mapping = {1:'Sports', 2:'Business', 3:'Science'}
    class_prediction= class_mapping[predicted_text[0]]
    messagebox.showinfo("Text Classification Result", f"Predicted Class: {class_prediction}")

window= tk.Tk()
window.title("Text Classification")
style =ttk.Style()
style.theme_use('xpnative')
label = tk.Label(window, text="Enter your text here:",
                              background= 'steel blue',
                              foreground ='white',
                              font=('Georgia',24,'bold'))
label.pack()
text_input=tk.Text(window, height=20, width=70)
text_input.config(font=("Georgia",12), bg="light cyan", fg="black")
text_input.pack()
window.iconbitmap('icon.ico')
button_for_classification = ttk.Button(window,text="Classify",command=text_classification)
button_for_classification.pack()
window.mainloop()
```