

**School of InfoComm Technology**

**Applied Analytics Assignment**

Diploma in Cybersecurity & Digital Forensics

Diploma in Data Science

Diploma in Information Technology

Year 2/3 (2023/2024), Semester 3/5

**TEAM/INDIVIDUAL ASSIGNMENT**

(40% of AA Module)

**Deadline for Submission:**

**Presentation Slides: 13th August 2023 (Sunday),23:59hrs**

**Report & Code: 13th August 2023 (Sunday),23:59hrs**

|  |  |  |  |
| --- | --- | --- | --- |
| Tutorial Group | : |  | |
| Team Number | : |  | |
| Tutor | : |  | |
| Student Name | : | Huang Wen | |
| Student No. | : | S10241971 | |
| Members | : | Student No. | Student Name |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Penalty for late submission:**

10% of the marks will be deducted every day after the deadline.

**NO** submission will be accepted after 20th August 2023, 23:59 hrs.

Contents

[1 Introduction (250 - 750 words) (Done) 3](#_Toc142852574)

[1.1 Problem understanding and the approaches 3](#_Toc142852575)

[2 Text Data Preprocessing (1000 - 2000 words) (Done) 5](#_Toc142852576)

[2.1 Load and cleanse the text data 5](#_Toc142852577)

[2.1.1 Loading the Data 5](#_Toc142852578)

[2.1.2 Cleaning the text 5](#_Toc142852579)

[2.1.3 Tokenization 6](#_Toc142852580)

[2.1.4 Removal of Stopword 6](#_Toc142852581)

[2.1.5 Lemmatization 7](#_Toc142852582)

[2.2 Transform the text data using Bag of Word and TF-IDF techniques 8](#_Toc142852583)

[2.2.1 Bag of Words 8](#_Toc142852584)

[2.2.2 Creating the BOW matrix 8](#_Toc142852585)

[2.2.3 BOW Understanding 9](#_Toc142852586)

[2.2.4 Pros & Cons 10](#_Toc142852587)

[2.2.5 TF-IDF 10](#_Toc142852588)

[2.2.6 Term frequency 10](#_Toc142852589)

[2.2.7 Inverse document frequency 10](#_Toc142852590)

[2.2.8 TF \* IDF 10](#_Toc142852591)

[2.2.9 Pros & Cons 11](#_Toc142852592)

[2.2.10 BOW to TF-IDF 11](#_Toc142852593)

[2.2.11 TF-IDF Understanding 11](#_Toc142852594)

[3 Text Data Understanding (1000 - 2000 words) (Done) 12](#_Toc142852595)

[3.1 Keywords extraction 12](#_Toc142852596)

[3.2 Association rule mining on the extracted keywords 14](#_Toc142852597)

[3.2.1 Keyword manipulation 14](#_Toc142852598)

[3.2.2 Encoding keywords 15](#_Toc142852599)

[3.2.3 Generate frequent itemsets using the Apriori algorithm 16](#_Toc142852600)

[3.2.4 Generate Association Rules 18](#_Toc142852601)

[3.2.5 All categories (entire dataset) 25](#_Toc142852602)

[3.3 Other suitable methods 26](#_Toc142852603)

[3.3.1 Word Cloud 26](#_Toc142852604)

[4 Summary and Further Improvements (500 - 750 words) 34](#_Toc142852605)

[4.1 Summarize your findings 34](#_Toc142852606)

[4.2 Explain the possible further improvements 35](#_Toc142852607)

[4.2.1 Text data preprocessing 35](#_Toc142852608)

[4.2.2 Text data understanding 35](#_Toc142852609)

[5 Reflection (500 - 1000 words) 35](#_Toc142852610)

[5.1 Suggest possible further improvement(s) to the current solution. 35](#_Toc142852611)

[5.2 With reference to the module learning objectives stated, reflect on the skills learnt and the skills you could have learnt better. 35](#_Toc142852612)

# Introduction (250 - 750 words) (Done)

## Problem understanding and the approaches

In this assignment, the task at hand involves solving various Text Analysis problems using the Jupiter notebook, a web-based interactive development environment for notebooks, Python code, and data. EvolutionAI has collected a dataset of roughly 1M text Reddit posts, with 1013 distinct classes (1000 examples per class). The classes are based on the assumed ‘topic’ of the text post, the topics being a manually curated taxonomy based on subreddits. The dataset given is called “reddit\_5.csv”. It consists of 5000 articles from Reddit posts. Each of these articles belongs to one of the categories mentioned above. In the CSV file, there are two columns. The first column contains the article’s contents, and the second column refers to the category the articles belong to.

Our exploration is structured into three cohesive steps, each contributing to a comprehensive understanding of the dataset before building the Text Classification Modeling. These steps are listed below.

This problem will be approached in the following 3 distinct steps:

1. Text Data Preprocessing
2. Text Data Understanding
3. Summary and Further improvements

The objective of text data preprocessing is to refine the contents of the documents such as removing noise, punctuation, and irrelevant information. So that we can create a refined dataset that lays the foundation for effective Text Classification Model Training. This will aid in improving the speed and accuracy reached when the data is fed into the model for training. Additionally, we employ transformative techniques such as Bag of Words and TF-IDF to represent the textual content in a manner suitable for further analysis.

Diving deeper into the dataset, the Text Data Understanding phrase allows us to unravel the essential terms that characterize each article, shedding light on the core themes and concepts. Beyond keyword extraction, we employ Association Rule Mining to uncover hidden relationships between keywords, providing insights into potential co-occurring patterns. As we navigate this step, we remain open to exploring alternative methods, such as WordCloud, that could offer additional perspectives on the data.

Lastly, all the insights gathered from the previous steps will be summarized into generic statements. This section encapsulates the key takeaways from our analysis, highlighting the significant patterns, trends, and relationships within the dataset provided. Additionally, Further improvements will be mentioned as to how the data pre-processing and data-understanding steps could have been optimized and enhanced to generate higher quality and accuracy association rules for the summary. These improvements may either require sophisticated techniques that are challenging to implement or have been overlooked during the assignment.

In the subsequent sections of this report, we will delve deeper into each step, providing detailed accounts of my methodologies, findings, and reflections. By the end of this assignment, we anticipate not only a comprehensive understanding of text analysis techniques but also the ability to apply these skills in deriving meaningful insights from real-world textual data.

A screenshot of a computer

Description automatically generated

Note\*

Many of the techniques used in Text Data Preprocessing (Individual), Text Data Understanding (Individual) and Classification Modeling (Group) require support and help from existing libraries. Without the help and support from these libraries, it will take me a long time to code out the required code to achieve the rules and information I need for summary and analysis.

# Text Data Preprocessing (1000 - 2000 words) (Done)

## Load and cleanse the text data

### Loading the Data

A screenshot of a computer

Description automatically generated

From the code snippet above, I first read the “reddit\_5.csv” file provided by the school and created a pandas dataframe named “data” to store the information of the CSV file. I used .info() function to show the columns in the CSV file and the count for each column. As we can see, there is a total of 5000 rows for each column. I then used data[‘category’].value count() to find out the number of rows for each category, which is 1000 each.

From the output above, we can see that the data is loaded properly and correctly and all the values such as the number of rows for each column and category match what the introduction has mentioned.

### Cleaning the text

A screenshot of a computer

Description automatically generated

The code in Figure 3 cleans the text in the “text” column. It converts all text to lowercase letters and uses regular expression patterns to remove tags, special characters, and special digits.

The last line of code applies the function to every single row in the “text” column. A new column “text\_clean” is created in the dataframe which consists of the cleaned text.

It is important to remove these characters as they do not provide any insights for our analysis afterwards and will only add noise to the model causing its training to be inefficient.

### Tokenization

A green and black text

Description automatically generated

The code snippet above converts the text from a string of characters to a list of words using the code re.split(‘\W+’, text). W in ‘\W+’ matches the complement of w which matches any alphanumeric characters. Meaning that it matches any non-alphanumeric characters. Additionally, whitespaces are not included in the list due to the “+” sign. Lastly, punctuation from the text is also removed which is optimal since punctuations are redundant characters in this problem as we are doing text analysis.

Example of tokenization

Sentence: “His name is John, that is my friend”

Tokenized sentence: [“his”, “name”, “is”, “john”, “that”, “is”, “my”, “friend”]

Tokenization is important as it is the process of separating (detachment) words (Characters group) from a given sentence or input. This help to break the raw text into words, sentences called tokens, which would be used in sequential steps such as the removal of stop words and lemmatization to better understand its context. When text is in a string it is hard to manipulate it as strings in Python are immutable while lists can be easily mutable seeing that they are mutable objects.

By allowing for these two techniques to be implemented, the text for subsequent steps and feeding to the model for training will be of higher quality.

### Removal of Stopword

A screenshot of a computer

Description automatically generated

Moving on, the next step in data preprocessing is the removal of stopwords. Stopwords are words that do not have specific semantics in the article. Such words include “the”, “is”, and “and”. Stripping these words from the text leaves behind only the important words for the model training. This process will reduce noise encountered by the model during training, ultimately enhancing the model’s ability to generalize better and obtain higher accuracies.

Yet, the challenge lies in identifying all the stopwords. Listing a few certain stopwords off the top of our heads is easy however compiling a whole stopwords dictionary will be extremely tedious and difficult for me to finish to finish before the assignment deadline.

A more efficient approach is to leverage an existing set of stopwords. In this assignment, we're provided with a text file named "stopwords.txt" by the school, which serves as our repository for determining which words should be classified as stopwords.

In the code snippet above, the function get\_stop\_words(stop\_file\_path) opens and retrieves the stopwords in “stopwords.txt”. Meanwhile, the function remove\_stopwords(text) scan through every word and removes all stopwords found in “stopwords.txt” file. Using these two functions, we create a new column in the dataframe call “text\_no\_stopword” where it contains articles that has stopwords removed.

### Lemmatization

A white background with black text

Description automatically generated

The final step for loading and cleansing data is to perform lemmatization on the text. Lemmatization is the process of converting words back into their root form. For instance, the word “caring” is lemmatized back to its root form “care”.

This is beneficial because words are inflected in many forms to communicate many grammatical categories such as tense, case, voice, gender, mood etc. However, these different forms often carry identical semantic meanings, and whether a word conveys tense or not doesn't alter the sentence's sentiment. In the absence of lemmatization, the model is filled with redundant words. For instance, [“play”, “played”, “playing”] might be perceived as distinct terms by the model, despite sharing the same semantic essence. By implementing lemmatization, we improve the model training while reducing noise, enabling faster and better training outcomes.

It's worth noting that another preprocessing technique, stemming, that is extremely similar to lemmatization. Stemming also aims to reduce a given word to its root word. However, stemming employs a simpler approach by removing a portion of the word's end to derive its stem. The drawback is that stemming algorithms lack contextual comprehension and merely slice characters, disregarding word meaning. For instance, the word “caring” is stemmed back to its root form “car” instead of “caring.

On the other hand, lemmatization algorithms understand the context of the word. As a result, they consistently yield actual language words, whereas stemming might not. Nevertheless, lemmatization's contextual complexity results in a slightly slower computational process compared to stemming. In the context of this problem, since our dataset is not very large, using lemmatization instead of stemming will be advised as the time taken has little difference and the output is more meaningful and useful.

Below is an output of each of the stages.

A screenshot of a computer

Description automatically generated

## Transform the text data using Bag of Word and TF-IDF techniques

In the section below, I will be focusing on the process of text data transformation, the techniques employed and the benefits and importance of transforming the text data.

It is important to vectorize the data as the Machine Learning model can only understand numeric values and not words (our current data). Therefore, without vectorization, our data would not be understandable to the model and unable to train the model.

### Bag of Words

The Bag of Words (BOW) technique is a fundamental approach to converting text into vectors for analysis. In this method, all articles are standardized to a uniform length, which is determined by the total number of unique words across all articles. This involves creating a vocabulary of distinct words. Each word becomes a feature represented by a column, and the cells in each row denote the frequency of that specific word's occurrence in the corresponding sentence.

A close-up of a message

Description automatically generated

### Creating the BOW matrix

The provided code snippet serves the purpose of transforming the preprocessed text into a Bag of Words (BOW) matrix. By setting "max\_features=5000," I ensure that the vocabulary includes only the top 5000 words with the highest frequencies. Additionally, "max\_df=0.15" indicates that words appearing in 15% or more of all articles will be excluded. This threshold helps remove overly common words that might not contribute significantly to the model's insights due to their widespread presence.

These hyperparameters were configured in the CountVectorizer object , assigned to the variable "count\_vect". The resulting BOW matrix, referred to as "text\_count," is generated by training and transforming the "count\_vect" object using the preprocessed text data. Using text\_count.shape() function, we observed that the BOW matrix consists of 5000 rows (number of articles) and 5000 columns (features).

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### BOW Understanding

The code above lists out the top 20 words with the highest and lower frequency counts. Gives us a better intuition and understanding of the most and least popular words.

### Pros & Cons

BOW vectorization has certain limitations. If the model is trained on new unseen documents that contain new distinct words, it could lead to vector length expansion or need recompute for the “topn” frequent words. Additionally, the resulting matrices are often sparse due to the extended length of the vectors, causing computational inefficiency. Lastly, BOW does not retain any details and information about sentence grammar or word order within articles.

On the other hand, BOW has its advantages. it is very easy and simple to understand and implement and is very flexible for customization on specific text data.

### TF-IDF

The other vectorization technique that was used is the Term frequency-inverse document frequency (TF-IDF) vectorizer which is based on the BOW model. TF-IDF is a number that shows and reflects the relevance and importance of the word in that particular article in a corpus.

### Term frequency

TF (term frequency) measures how frequently a word (feature) appears in an article. It denotes the Number of times a word occurs in an article.

A close up of a text

Description automatically generated

Tf ­i.j  = number of times i occurs in j divided by the total number of terms in j

E.g.,

“I am John, I love to eat banana.”

TF(“I”) = 2/8 = 1/4

TF(“am”) = 1/8

### Inverse document frequency

IDF (inverse document frequency) is the Inverse of document frequency and looks at how common (or uncommon) a word is amongst the corpus. It is calculated by taking the logarithm of the number of documents over the number of documents containing the word we are calculating the IDF value for. The closer it is to 0, the more common a word is and vice versa. The reason we need IDF is to help correct for words like “of”, “as”, “the”, etc. since they appear frequently in an English corpus. Thus by taking inverse document frequency, we can minimize the weighting of frequent terms while making infrequent terms have a higher impact.

A black text on a white background

Description automatically generated

### TF \* IDF

The TF-IDF score is then calculated by multiplying the TF value with the IDF value. The higher the TF-IDF score for a word, the more relevant and important the word is in that particular article and hence, indicative of the context of the article, while the lower the score the less important the word is.

### Pros & Cons

TF-IDF offers several advantages. It is easy to compute and serves as a metric for identifying the most important words within an article. This allows the extraction of keywords, as we'll delve into further in the subsequent section. Furthermore, using TF-IDF allows for the efficient identification of similar article based on their TF-IDF values.

However, similar to the limitations faced by the Bag of Words (BOW) approach, TF-IDF encounters certain drawbacks since TF-IDF is transformed using the BOW matrix. It struggles to capture the semantics of the text and does not retain information regarding the order and grammar of the sentence. Making it often used only as a lexical-level feature. Also, synonyms are not accounted for and the highest TF-IDF score may not make sense with the topic of the document since IDF gives high weight when the DF of a term is low.

**A screenshot of a computer

Description automatically generated**

### BOW to TF-IDF

The code snippet above transforms the BOW matrix into a TF-IDF matrix. It does by using TFidfTransformer function where smooth\_idf=True is used to prevent any zero divisions. The print statements below were to ensure that the TF-IDF matrix is of the appropriate length (5000) as well as the number of features the transformer trained with.

### TF-IDF Understanding

A screenshot of a computer

Description automatically generatedThe above code prints out the top 20 highest and lowest TF-IDF value words. This gives us a better understanding of which words are considered relevant and which words are not so important in determining the category of a document. From the image we see words like “sprite”, “iceland”, and “streamable” has high relevance while words such as “klavan”, “ingolstadt”, and “etihad” have little relevance.

# Text Data Understanding (1000 - 2000 words) (Done)

## Keywords extraction

Using the entire length of the article for association mining is impractical and discouraged. Instead, a more efficient approach is to extract the keywords in the article that reveals the most information and context of the articles. In the section below, we will dive into the process of extracting and formatting the keywords for each article in the dataset so that it is suitable for association mining.

A screenshot of a computer program

Description automatically generated

The code snippet above serves the purpose of generating a list of keywords found within the article. This list comprises 5000 rows, where each row holds a dictionary. Within each dictionary, keywords act as the key, while their corresponding TF-IDF (Term Frequency-Inverse Document Frequency) values serve as the associated values.

The number of keywords extracted per article is configured using the variable “topn”, 7. If "topn" is set too high, the resulting huge number of keywords could lead to an overwhelming number of rules during the subsequent associate rule mining, potentially rendering them less meaningful and insightful.

The selection of keywords depends on the usage of TF-IDF. This metric aids in gauging the importance of words within a given document against their prevalence across the entire corpus. The keyword extraction process depends on the calculation of TF-IDF values for each word. Subsequently, the highest "topn" words, as determined by their TF-IDF values, are chosen as the keywords.

To maintain clarity and manageability, all identified keywords and their corresponding TF-IDF values are stored in a structured dataframe named 'temp’. This dataframe is sorted from highest TF-IDF to lowest TF-IDF values.

A white background with black dots

Description automatically generated

A white background with black and white lines

Description automatically generated

The output offer insights into the distribution of TF-IDF values by showcasing the top 10 highest and lowest feature numbers, each accompanied by its respective TF-IDF value.

Here it is observed that certain feature numbers such as 880, 4526, 4809 and 438 stand out as outliers due to their extremely high TF-IDF values. Furthermore, the range of TF-IDF values clustered spans from approximately 0.10 to 0.07. This can be estimated since a TF-IDF value of 0.07 nets feature number 434 to be the tenth lowest TF-IDF value and a TF-IDF value of 0.10 nets feature number 3648 to be the tenth highest TF-IDF value.

A screenshot of a computer

Description automatically generated

To conclude, the keywords are formatted using the code snippet above which iterates through the keywords and extracts the feature names. This ensures that the keywords are formatted and suitable to be input for association mining as association mining does not need the TF-IDF value.

## Association rule mining on the extracted keywords

### Keyword manipulation

I will now employ associate rule mining to generate relevant association rules from item sets based on the keywords extracted. This technique is a valuable tool that enables the exploration of significant relationships within item sets based on the keywords. These rules are constructed using statistical metrics like support, confidence, and lift, which aid in discovering interesting patterns, insights, and relationships within the data.

To generate meaningful association rules, a two-step process is undertaken. Firstly, employ the Apriori algorithm to generate the frequent item sets needed for further analysis. Next, configure parameters such as support, lift and confidence metrics to filter out the best itemset which would be the generated rules that are relevant to the data.

A screen shot of a computer

Description automatically generated

To start off, I performed data manipulation on the 'formatted keywords' column to create 5 new columns in the data frame, each of the new columns consists of a keyword from the ‘formatted keywords’ column. A new data frame called ‘keywords\_df’ and list ‘keywords\_list’ was created to store the values in these 5 columns.

### Encoding keywords

A screenshot of a computer program

Description automatically generated

The subsequent step involves encoding keywords to format them so the computer can understand them. The code snippet above is the code responsible for encoding the keywords. The process begins by creating a list named trans with a shape of (5000, 7), meaning that there are 5000 lists in this list, each with a length of 7 (containing the top 7 keywords extracted from each article).

Next, a transaction encoder object is initialized using “te = TransactionEncoder()”, followed by fitting and transforming the object on “trans”. The output is stored in the variable “data\_encoded” which has a shape of 5000 rows and 4835 columns (5000, 4835).

Transaction encoder is used to convert item lists into transaction data for frequent itemset mining. It works by receiving inputs as a list of transactions, where each transaction is represented as a list containing keywords. Each transaction corresponds to a single observation, and the items within a transaction are the categorical variables associated with that observation. The Transaction Encoder transforms the list of transactions into a binary matrix, where each row corresponds to a transaction and each column corresponds to a keyword. The matrix is a representation of the presence or absence of each item in each transaction. If an item is present in a transaction, the corresponding cell contains a "True"; otherwise, it contains a "False." In this case, the number 4835 represents the number of unique keywords there are, and all articles will be of length 4835. This binary matrix created by the TransactionEncoder essentially performs one-hot encoding for the categorical variables (keywords) in my data. One-hot encoding is a technique to represent categorical variables as binary vectors, where each category is represented by a unique column, and a 1 or 0 indicates the presence or absence of that category. By using the Transaction encoder, we are preparing the data for analysis using association rule mining algorithms like Apriori. It serves as a bridge between the raw categorical data and the input format required by these algorithms.

**A screenshot of a computer

Description automatically generated**

### Generate frequent itemsets using the Apriori algorithm

The Apriori algorithm generates frequent itemsets by finding the frequent item sets from the universe of all the possible item sets. The Apriori algorithm leverages two simple logical principles on the lattice item sets to reduce the number of item sets to be tested for the support measure. The Apriori principle states that “If an item set is frequent, then all its subset items will be frequent.” (Tan et al, 2005). The item set is “frequent” if the support for the item set is more than the support threshold. Conversely, if the item set is infrequent, then all its supersets will be infrequent Support score is the metric which it uses to identify itemsets that occur frequently, the higher the min\_support, the more frequently the itemset occurs.

A screen shot of a computer

Description automatically generated

From the code snippet above, the minimum support threshold was configured to be 0.5%. This means that itemsets appearing lesser than 0.5% of all articles will be filtered out from the list of frequent itemsets. Using the Apriori algorithm with a specific minimum support threshold of 0.5%, the output produced was a data frame with 144 rows and 2 columns where the first column contained the support value of the itemset, and the second column contained the frequent itemset that met the condition.

*A white background with black text

Description automatically generated*

Next, I employ the describe function to get the statistical details of the itemsets that were generated to gain a better understanding of the support values of these filtered frequent itemsets. From the output above, there are 144 itemsets in total with the mean and median support scores 0.009681 and 0.007200 support score respectively. Such statistical details are crucial as they help to determine if the minimum support threshold value configured is suitable and if alternative values should be considered as new minimum support thresholds. For instance, an excessive number of itemsets might prompt the evaluation of different threshold values.

In addition, I repeated the steps above to encode and generate the frequent itemsets for the 5 individual categories as I will be performing associate rule mining on the entire dataset as well as the individual categories.

The code snippet below shows how keyword encoding and generation of frequent itemsets were done for the individual categories. The line 'category' == 'soccer' determines the category of the article that will be selected for encoding and itemset generation.

A screen shot of a computer

Description automatically generated

**A computer screen shot of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

### Generate Association Rules

After generating the frequent itemsets, we will move on to performing associate rule mining (ARM) on the individual categories as well as the entire dataset to generate association rules. Associate rule is a learning technique that helps identify the dependencies between two data items. Based on the dependency, it then maps accordingly so that it can be more profitable. Association rule furthermore looks for interesting associations among the variables of the dataset. These association rules are useful to analyze as they can reveal patterns that cannot be seen in the previous steps.

The 3 main metrics that will be analyzed and used are support, confidence, and lift. The support of an item is simply the relative frequency of occurrence of an item set in the transaction set and is an indicator of whether a rule is worth considering. The confidence of a rule measures the likelihood of the consequent of the rule being present in a transaction that contains the antecedent of the rule, hence providing the reliability measure of the rule. Below is an equation for confidence.

A math equation with black text

Description automatically generated

Lastly, the lift of the rule is the ratio of the observed support of the antecedent and consequent with what is expected if the antecedent and consequent were independent. Lift values closer to 1 mean the antecedent and consequent of the rules are independent, and the rule is not interesting while higher lift values indicate that the rule is more interesting. The following is an example of an equation for lift.

A close-up of a sign

Description automatically generated

In general, I aimed to generate approximately 10 rules. Less than 10 rules would be difficult to find patterns and insights, and more than 10 rules generated would produce too much noise which will affect my analysis.

#### Soccer

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The metrics for the soccer category included a minimum confidence of 70%, a minimum lift of 15 and a minimum support threshold of 0.01. In total 10 rules were generated, and it is observed in soccer articles, the article covers a range of topics related to soccer, such as platforms to watch, and icons relating to different elements.

Let us analyze the rules one by one. The rule, (www, youtube) 🡪 (watch) suggests that the articles could be discussing soccer-related content on YouTube. This might include watching match highlights, analysis, tutorials, and other video content related to soccer. It also suggests that soccer articles often have YouTube links that direct people to YouTube to watch videos that are related to soccer.

Next, rules such as (sprite, ball) 🡪 (icon) indicate that the articles might be discussing the usage of icons and symbols related to soccer. Sprite is a clear, lemon and lime-flavoured soft drink created by the Coca-Cola Company, and Coca-Cola Company is also a World Cup sponsor. Whereas soccer balls may have printed logos and icons on it. The word “icon” could involve using graphical representations or symbols to represent different aspects of the game such as sponsors' icons.

Additionally, the rule (watch) 🡪 (youtube) may suggest that YouTube is one of the popular platforms to watch soccer matches, analysis, tutorials, or soccer-related content.

Lastly, the rule (sub, substitution) 🡪 (icon). "Substitution" likely refers to player substitutions in soccer matches. "Icon" could imply the use of visual symbols. This rule suggests that the articles might be discussing how player substitutions are represented visually using icons. It could refer to icons that symbolize players on the field, changes, or any graphical representation of substitutions.

The rules generated for soccer all had high confidence of more than 0.7 and high lifts of more than 15. Indicating that these rules were extremely interesting and reliable. The keywords also had high support of more than 1% meaning that they occurred frequently in the article.

#### Snowboarding

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

For snowboarding, the minimum support threshold has been adjusted downward to 0.006, which is slightly lower than the threshold used for the soccer category (0.01). Additionally, the measures of confidence and lift have been reduced to 0.35 and 6, respectively, indicating a less strict association between terms compared to the soccer category. In total, 10 rules were generated.

Moving on, many of the rules are related to snowboarding, such as location and boards used for snowboarding. The rule (mont) 🡪 (tremblant), shows that both words appear together in the articles frequently. The rule is referring to Mont Tremblant Ski Resort, a popular ski resort in Quebec, Canada. These rules indicate that the articles might be discussing various aspects of Mont Tremblant, such as its snowboarding facilities, trails, amenities, or experiences.

Next, rules such as (camber) 🡪 (board), (camber) 🡪 (rocket) and (bought) 🡪 (board) suggest that articles discuss about the world of Snowboarding Equipment. Camber and rockets refer to the different types of snowboards. Articles might be explaining the characteristics and applications of different snowboarding styles, specifically rocker and camber as well as topics like purchasing snowboards.

For (bear) 🡪 (mammoth) rule, even though the words refer to animals. However, in the snowboarding context, it referred to Mammoth Mountain Ski Area and Big Bear Mountain Resort in California, America. These rules suggest that the articles might be comparing or contrasting the snowboarding experiences at both venues.

Lastly, for rule (epic) 🡪 (pas). This rule implies that the articles could be discussing the Epic Pass, which is a well-known season pass offering access to multiple ski resorts.

#### Triathlon

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

For the Triathlon category, the minimum support threshold used is 0.009, a confidence threshold of 0.3, and a lift threshold of 3 which were lower than the soccer category. A total of 10 rules were generated. Articles in the triathlon category mainly discuss about triathlon training and equipment needed.

(swim, run) 🡪 (bike) rule suggests that articles are discussing about triathlon training, and preparing for the diverse challenges of a triathlon.

Rules such as (tt) 🡪 (bike) and (suit) 🡪 (tri) show that articles are discussing about equipment and swimwear for triathlon. TT bike refers to a triathlon bike, while a suit is for the swimming section of a triathlon.

For rule (plan) 🡪 (training), it indicates that the articles might explore the strategic dimension of triathlon training. Topics could encompass the formulation of structured training plans, setting attainable goals, and devising training strategies.

Lastly, (km) 🡪 (run) suggest that articles discuss about running training, what is the distance they have run in km.

#### Judo

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Regarding the Judo category, a minimum support threshold of 0.009, a confidence threshold of 0.35 and a lift threshold of 7 are configured to generate the association rules. In total, 9 association rules is generated. Articles in Judo category mainly discussed about belt rank and combat technique.

For rules such as (green) 🡪 (belt), (black) 🡪 (belt) and (white) 🡪 (belt), it suggests that many articles discussed about different belt ranks and various levels of expertise and experience within the martial art.

Rules such as (mata) 🡪 (uchi) and (seol) 🡪 (nage) is referring to Uchi mata and seoi nage. These two are one of the traditional forty throws of judo, Uchi mata is an inner-thigh reaping throw, while seoi nage is a shoulder throw. This shows that articles also mainly discuss Judo techniques to use.

Lastly, for (martial) 🡪 (art) shows that judo is a type of martial arts.

#### Surfing

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

To conclude the rule analysis for individual categories, the surfing category used a minimum support threshold of 0.007, a confidence threshold of 0.3 and a lift threshold of 2. A total of 10 rules were generated. Articles for surfing categories mainly discuss about location, equipment, and techniques relating to surfing.

Firstly, rules such as (cruz) 🡪 (santa), refer to Santa Cruz, a popular destination where many surfers go for surfing. Articles may discuss aspects related to Santa Cruz, such as travel tips.

Rules such as (box) 🡪 (fin), (shortboard) 🡪 (board), (fish) 🡪 (board), refers to the types of board used for surfing. The fin box allows users to mix and match fins depending on the surf conditions and their ability. Fish boards and shortboards are types of surfing boards. Topics of the article may include the advantages and disadvantages of shortboards, suitable wave conditions for riding them, and how to choose the right shortboard for different skill levels.

(camp) 🡪 (surf) rules indicate articles about surf camps, where individuals can learn to surf or improve their skills. Articles might cover the structure of these camps, instructors' expertise, and the benefits of joining such programs.

Lastly, rules such as (catch) 🡪 (wave), (foot) 🡪 (wave) and (paddle) 🡪 (wave). These show that the articles cover surfing techniques such as catching wave, paddling and footwork.

### All categories (entire dataset)

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

The minimum support used for frequent itemset generation was 0.005, the minimum lift threshold is 2 and the confidence threshold was 0.3. In total, 9 rules were generated.

The rules generated belonged to categories including “soccer”, “snowboarding”, “triathlon”, “judo”, and “surfing”. Regarding rules which belonged in the soccer category, the rules talked about YouTube, an online video-sharing and social media platform which can be seen from the rule (youtube) 🡪 (watch). This rule showed that soccer fans on Reddit mainly use YouTube to watch soccer-related content. Rules like (icon) 🡪 (sub) and (icon) 🡪 (sprite) refer to the appearance of the sprite icon and substitution icon during a soccer match or other soccer-related content.

For the judo category, the rules talked about BJJ, a short form for Brazilian Jiu-Jitsu, a grappling-based martial art. The articles for the judo category often compare these two similar martial arts and information regarding them.

For the triathlon category, the rules talked about Triathlon bikes and running. As the name suggests, triathlon bikes are suitable for triathlons, while running is one of the activities included in triathlon.

Lastly, there were no rules generated for snowboarding and surfing. It can be inferred that the keywords for these two categories are not very common compared to other categories.

## Other suitable methods

I used two other methods to perform data understanding on the keywords extracted. By using visualizations of the keywords, I was able to better my intuition and understanding of the most popular keywords.

### Word Cloud

**Words in a word cloud

Description automatically generated**

Word cloud is a good intuitive high-level way of visualizing the most popular keywords extracted. The bigger the word, the more frequently the word appears. Hence words such as bike, judo and board are the most popular keywords that can be easily identified. However, it is not so good at a more granular level seeing that it is hard to compare words of similar sizes E.g., bike and board. Another issue is that when there are too many words it becomes extremely messy to interpret. The word cloud above prints out the 20 most popular keywords.

Word cloud for other categories

Soccer

A close up of words

Description automatically generated

Snowboarding

A word cloud with text

Description automatically generated

Triathlon

A word cloud with text

Description automatically generated

Judo

A word cloud with text

Description automatically generated

Surfing

A close up of words

Description automatically generated

**Frequency Bar**

**A graph with blue and black text

Description automatically generated**

The other data visualization technique which I implemented was to use the frequency bar chart as it can compare many words at a granular level. The frequency bar chart might not be as appealing to interpret as the word cloud, but it solves both the problems of the word cloud. For instance, we can see even minute differences in the frequency of the keywords and can compare many of the keywords in an organized manner. From the graph, we see the 5 most popular keywords in order are board, judo, bike, surf, and wave.

Frequency bar for other categories

Soccer

A graph with blue and white lines

Description automatically generated

Snowboarding

A graph with blue and white lines

Description automatically generated

Triathlon

A graph with blue and black text

Description automatically generated

Judo

A graph with blue and white lines

Description automatically generated

Surfing

A graph with blue and white lines

Description automatically generated

# Summary and Further Improvements (500 - 750 words)

## Summarize your findings

In summary, the problem involves building a multi-class classification model for the dataset (reddit\_5.csv) that we were given and at the same time performing data understanding on the documents to uncover meaningful patterns that can help us predict and find patterns in the different classes of articles

To perform text data understanding, text data pre-processing had to be carried out first. This includes data loading, followed by data cleansing techniques such as removal of unnecessary characters, tokenization, removal of stop words and lemmatization. Following data cleansing, data transformation techniques like Bag of Word and TF-IDF techniques is applied to prepare the data for text data understanding.

The next step was to perform text data understanding. First, extract keywords from each article. The top 5 keywords with the highest TF-IDF value from every document will be extracted. Next, perform association rule mining on the extracted keywords. For the individual categories as well as the entire dataset, generating frequent item sets using the Apriori algorithm. Next, generate association rules using lift and confidence thresholds to filter out the itemsets which was how we obtained our list of association rules. Below is what was discovered during ARM:

1. Soccer: The article covers a range of topics related to soccer, such as platforms to watch (YouTube), and icons relating to different elements (Sprite).
2. Snowboarding: The articles cover topics related to snowboarding, such as location s and boards used for snowboarding.
3. Triathlon: The articles discuss about triathlon training and equipment needed.
4. Judo: The article discussed about belt rank such as white belt and combat techniques such as uchi mata.
5. Surfing:
6. Entire dataset. From the overall dataset, almost all the patterns above were reinforced and as there were no rules in the snowboarding and surfing category, it can be inferred that keywords relating to snowboarding and surfing are the least common among all five categories.

Finally, other than performing ARM on the data, I used other techniques such as word cloud and frequency bar chart to help visualize the keywords extracted better which would aid in my understanding and intuition of the data.

## Explain the possible further improvements

### Text data preprocessing

For text pre-processing, an improvement to be made would be to use a larger dataset as this would give a better representation of documents in the different categories which would allow the model to generate rules that are more meaningful. Additionally, it is good to know the time when the Reddit post is posted, so that we can better analyze on the data like finding out related events that happened during the time frame and establishing connections with it.

### Text data understanding

Regarding text data understanding, there were some improvements to be made pertaining to the testing of the values of the hyperparameters. For instance, I could have extracted a larger number of keywords by increase “topn” to a higher value, I could then compare the ARM results and see which set of rules generated I can discover more meaningful patterns from. Additionally, I could have experimented with different confidence threshold and lift threshold, allowing more rules to be generated as many of the rules are the repetition of each other E.g., ‘wi -> fi’ provide the same information as ‘fi -> wi’. By allowing more rules to be generated, I might have been able to discover some additional meaningful patterns in the data.

# Reflection (500 - 1000 words)

## Suggest possible further improvement(s) to the current solution.

Here are some suggestions for possible further improvements to the current solution in the context of the given assignment:

Enhanced Preprocessing:

Incorporating advanced preprocessing techniques like autocorrection for misspelled words using libraries such as TextBlob can help improve the accuracy of word representations, especially in informal online text like Reddit posts.

Handling emojis, special characters, and domain-specific jargon can provide a more accurate representation of the language used in the dataset, leading to better insights.

Feature Engineering:

Leveraging word embeddings like Word2Vec or GloVe allows capturing semantic relationships between words. These embeddings can uncover intricate contextual nuances, contributing to a more sophisticated understanding of the textual content.

## With reference to the module learning objectives stated, reflect on the skills learnt and the skills you could have learnt better.

I have learned many skills through this assignment. Here's my reflection on the journey of mastering these techniques:

1. Cleaning Text:

Cleaning text, which involves removing unnecessary characters, symbols, and special characters, has taught me the importance of preparing a clean and standardized dataset. This step not only improves the quality of the data but also ensures consistency throughout the analysis process.

2. Tokenization:

Tokenization, the process of breaking text into individual words or tokens, has been an enlightening experience. It's fascinating to observe how algorithms interpret language by dividing it into smaller units, laying the foundation for further analysis.

3. Removing Stop Words:

The concept of removing common stop words was eye-opening. It made me realize how these seemingly insignificant words can be noise and impact the accuracy of analyses. This step has reinforced the idea that meaningful content stands out when unnecessary words are eliminated.

4. Lemmatizing:

Lemmatizing, a technique to reduce words to their base or root form, has revealed the power of simplification in text analysis. Understanding that different forms of a word can be treated as the same entity aids in drawing meaningful insights.

5. Bag of Words:

Learning about the Bag of Words technique has been a cornerstone in my journey. The idea of representing text numerically and quantifying word occurrences has shown me how the intricacies of language can be translated into structured data that algorithms can process.

6. Bag of Words Matrix and TF-IDF:

Creating a Bag of Words matrix and exploring TF-IDF (Term Frequency-Inverse Document Frequency) has illuminated the concept of feature engineering. Transforming textual data into matrices with numerical values has broadened my perspective on how data representation can influence analysis outcomes.

7. Keywords Extraction:

The process of extracting keywords from text using TF-IDF has emphasized the importance of relevance. Witnessing how certain words stand out and carry more weight in a document has heightened my awareness of context-driven insights.

8. Association Rule Mining with Apriori Algorithm:

Exploring Association Rule Mining using the Apriori algorithm has been a stimulating experience. The ability to unveil hidden relationships between keywords has underscored the significance of patterns and connections that might otherwise remain concealed.

In conclusion, this journey of learning and applying text analysis techniques has provided a well-rounded understanding of how textual data can be harnessed for meaningful insights.