

**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

**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 | : | P01 | |
| Team Number | : | 4 | |
| Tutor | : | Joey Chew | |
| Members | : | Student No. | Student Name |
|  |  | S10244252C | Isaac Khoo |

**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.

Table of Contents

[Introduction 3](#_Toc142845525)

[Text Data Processing 3](#_Toc142845526)

[Load the Data 3](#_Toc142845527)

[Cleanse the Text Data 3](#_Toc142845528)

[Bag-of-Word 3](#_Toc142845529)

[TF-IDF 3](#_Toc142845530)

[Text Data Understanding 3](#_Toc142845531)

[Keywords Extraction using TF-IDF 3](#_Toc142845532)

[Association Rule Mining 3](#_Toc142845533)

[Pictures 3](#_Toc142845534)

[Writeup 3](#_Toc142845535)

[Summary and Further Improvements 3](#_Toc142845536)

[Summary of findings 3](#_Toc142845537)

[Possible further improvements 3](#_Toc142845538)

[Reflection 3](#_Toc142845539)

# Introduction

In this assignment, we will solve various Text Analysis problems using Python. 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. In this assignment, I will be dealing with a subset of that data, a total of 5000 articles, split amongst 5 labels - each label contributing 1000 articles to the pool. The 5 labels being 'soccer', 'snowboarding', 'triathlon', 'judo', and 'surfing'. This report will be split into a total of 5 sections. Introduction, Text Data Processing, Text Data Understanding, Summary and Further improvements and Reflection. The introduction will be on the problem understanding and approaches used. In the Text Data Processing, I will be explaining the loading and cleansing of the text data, after which I will transform the text data using Bag of Word and TF-IDF techniques. The Text Data Understanding will be about Keywords extraction, Association rule mining on the extracted keywords and Other suitable methods. For summary and Further improvements, I will be touching on summarizing my findings and explaining the possible further improvements I can make. Lastly, the reflection will be suggesting possible further improvement(s) to the current solution and with reference to the module learning objectives stated, reflect on the skills learnt and the skills I could have learnt better.

# Text Data Processing

## Load the Data

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

In this stage, I am importing several necessary packages that I’ll be using throughout my analysis. These packages include numpy, pandas, matplotlib, seaborn, sklearn (scikit-learn), mlxtend, and others. These packages provide tools for data manipulation, visualization, model building, and evaluation. Next I loaded the dataset from the file named 'reddit\_5.csv' using pd.read\_csv(), and then displaying the first few rows of the dataset using dat.head(). After which I did some exploratory data analysis using ‘dat.info()’ and ‘dat.describe()’

## Cleanse the Text Data

A screenshot of a computer code

Description automatically generatedA close-up of a text

Description automatically generated

In this Stage I am performing text preprocessing on the 'text' column of your dataset. Text preprocessing is a crucial step before performing any analysis or building models on text data.

Firstly, I start by extracting the content of the 'text' column from the first row of the DataFrame. Next, I create 2 functions for text processing, pre\_process(text) takes a text input and converts the text to lowercase, Removes any HTML-like tags, Removes special characters, digits and non-word characters. get\_stop\_words(stop\_file\_path) reads stop words from a file and returns them as a set. Next, I will load a set of stop words from a file named "stopwords.txt". Stop words are words that will be removed during preprocessing to focus on the more meaningful words. Next, I start preprocessing the data. The 'text\_all' variable is created by applying the pre\_process function to each element in the 'text' column of your dataset using the apply() function. This preprocesses all the text data in your dataset. Lastly, I display the preprocessed text and print the shape of the preprocessed text data which gives us and idea on how many text entries I will be dealing with.

## Bag-of-Word

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

In this Stage I will be using the Bag-of-Word method to create a vocabulary of words, ignore words that appear in 15% of documents and eliminate stop words. We start by using the CountVectorizer with specific parameters. max\_df Ignores terms that appear in more than 15% of the documents, stop\_words the list of stopwords you've previously loaded, max\_features limits the vocabulary to the top 5000 words by frequency. Next, we transform the preprocessed text data (text\_all) into a matrix of token counts using the fit\_transform() method of the CountVectorizer. Each row represents a document, and each column corresponds to a word in the vocabulary then I show the number of documents (rows) and the number of unique words (columns) in the token counts matrix. I also print the list of stop words used by the countvectorizer. Next, I calculate the sum of word counts for each word across all documents. Create a list of tuples, where each tuple contains a word and its frequency across all documents and print the highest frequency words and lowest frequency words. After which I retrieve the list of words that was used in the CountVectorizer, get the total number of words in the vocabulary and onverts the token counts matrix into a DataFrame.

## TF-IDF

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

In this section I will be generating a TF-IDF matrix. I start by creating an instance of ‘TfidfTransformer’ with the parameters ‘smooth\_idf=True’ and ‘use\_idf=True’. I also transforms the token counts matrix (text\_counts) into a TF-IDF matrix. using the ‘fit\_transform()’ method of the ‘TfidfTransformer’. Next, I print the IDF value for each word in the vocabulary and print the shape of IDF values which indicates the number of words in the vocabulary. Next, I sort the features based on their IDF values in ascending order and print the first 100 words with the lowest IDF values. These are words that are more common. Next, I convert the TF-IDF matrix (text\_tfidf) into a DataFrame. ‘max\_value = text\_tfidf.max(axis=0).toarray().ravel()’ finds the maximum TF-IDF value for each feature (word) across the dataset. I sort the features based on their maximum TF-IDF values in ascending order and print the words with the lowest TF-IDF values, these are words that are less unique and less important in individual documents. Similarly, I print the features with the highest TF-IDF values. These are words that are more unique and important in individual documents.

# Text Data Understanding

## Keywords Extraction using TF-IDF

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

In this section, I will be performing Keywords Extraction using TF-IDF. I start by retrieving the preprocessed text of the document at the specified index and retrieving the TF-IDF vector for the same document. I also print the data that I have retrieved to understand what im working with. Next I Create a DataFrame (temp) for the TF-IDF Vector with corresponding TF-IDF values in the document's TF-IDF vector. Next, I sort the DataFrame (temp) in descending order based on the TF-IDF values and print the sorted DataFrame (temp) to see the words with the highest TF-IDF values. Next, I will be performing keyword extraction for each document. I start by obtaining the TF-IDF vector for the current document, creating the temp data frame Selecting the top, n, words with the highest TF-IDF values from the DataFrame and lastly extracting the words and their corresponding TF-IDF values and storing them in the results list. Next, I am updating DataFrame with Keywords and Preprocessed Text. This is done by ‘dat['cleansed\_txt'] = text\_all’ which adds the preprocessed text to the DataFrame under the column 'cleansed\_txt' and ‘dat['keywords'] = results’ which adds the extracted keywords to the DataFrame under the column 'keywords'. Lastly, I will be doing classification modelling. I start by preparing the data for classification by assigning the token counts matrix as an array to ‘X’ and the ‘category' column as numeric labels to ‘y’. Next, I split the data into training and testing sets using ‘train\_test\_split’. Using ‘cross\_val\_score’ I am able to compute the cross-validation accuracy for logistic regression which is 0.95. Finally, I evaluate the final logistic regression model's performance on the test set and print the test score which is 0.96.

## Association Rule Mining

### Pictures

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

### Writeup

In this section, I will be performing Association Rule Mining and analyzing the results for different sports categories. I start by creating separate DataFrames for each category ('soccer', 'snowboarding', 'triathlon', 'judo', 'surfing') based on the 'category' column. Next, I will be extracting Keywords Lists for each Category. This is done by iterating through the first 1000 rows and extracting the list of keywords from the 'keywords' column for each Dataframe. Next, I created a list of keywords for each category ('soccer', 'snowboarding', 'triathlon', 'judo', 'surfing'). After which I will be performing Transaction Encoding for each categorys keyword list to create binary-encoded itemsets. Next, using the Frequent Itemset Generation (Apriori Algorithm), I am applying the Apriori algorithm to each encoded DataFrame to find frequent itemsets with at least 1% support. My frequent itemsets are stored in DataFrames(frequent\_soccer, frequent\_snowboard, frequent\_triathlon, frequent\_judo, and frequent\_surfing). Next I will be using ‘describe()’ to display descriptive statistics for the 'support' column in each frequent itemset DataFrame. Finally I will be performing Association Rules Based on Confidence, Lift and both Confidence and Lift Thresholds. For Confidence Thresholds, I start by generating association rules for each sport category using the confidence metric and a minimum threshold of 0.3. and sorting the rules based on confidence in descending order. For Lift Thresholds, Similar to the confidence-based rules, I generate the association rules using the lift metric and a minimum threshold of 1 and then sort the rules based on lift in descending order. Lastly for Lift and Confidence Thresholds, For each sport category, I am selecting association rules that meet a specific combination of lift and confidence thresholds. The selected rules are then displayed.

# Summary and Further Improvements

## Summary of findings

A screenshot of a computer code

Description automatically generated

For TF-IDF my results came back with a mean cross-validation accuracy of 0.95 which indicates that the logistic regression model I built is performing well and is making accurate predictions on average across different subsets of your data It also retuned a Best Cross-Validation Score of 0.95 which indicates that that the model is performing well after hyperparameter tuning. It suggests that the chosen hyperparameters are leading to a high average accuracy across different subsets of the data used in cross-validation. Lastly, it returned a Test Score of 0.96. This indicates that the logistic regression model is performing very well on new, unseen data, and it's making accurate predictions.

After performing Association Rules on each of the categories of sports, I have successfully displayed the lift threshold, confidence threshold and both confidence and lift thresholds. It shows a high confidence value, indicating that the presence of the consequent is highly likely when the antecedent is present. High confidence suggests a strong relationship between the items in the rule. It also shows a lift value significantly higher than 1 indicates a meaningful and interesting association.

## Possible further improvements

Some possible improvements I can make for better Association Rule Mining Results is I could do better data cleaning, Adjust the minimum support threshold for frequent itemsets. A combination of data preparation, parameter tuning, and domain knowledge can lead to improved association rule results.

Some possible improvements I can make for better TF-IDF results are choosing more appropriate strategies for handling missing values, such as imputation or removal, to prevent biased model performance. Data augmentation, Hyperparameter Tuning, etc. Starting with one or a few strategies at a time, evaluating the impact on the model's performance, and continuing refining my model until I achieve desired results. However, there might be tradeoffs between matrix optimizing one matrix my lead to a decrease in another.

# Reflection

The completion of this assignment on Applied Analytics has been a great learning experience. I have gained a deeper understanding on how to do text data processing, text data understanding, and some sorts of basic machine learning understanding. I have also learned about the various techniques for cleaning, transforming, and wrangling to make it ready for analysis.

Working with Jupyter Notebook has been particularly useful, as it allows for an interactive and intuitive experience while working with data. The ability to execute code in cells and visualize outputs in real-time has made the data preparation process much more efficient and streamlined.

In addition, I have also learned about various techniques such as TF-IDF, Association Rule Mining, and Bag-of-Word. I have come to appreciate the power of Analyzing data, as it can help uncover patterns, relationships, and trends in the data. It can also help fuel basic machine learning ideology.

Overall, this assignment has greatly enhanced my understanding of Applied, and I am confident that these skills will be invaluable in my future work with data.