Data Mining Project Report

June 2021

PREPARED BY
TRAN VIET HANG
TRAN THANH DAT
LAM HUE DUNG

INSTRUCTOR: DR TRAN MANH HA

TABLE OF CONTENTS

ABSTRACT

Online retail store grows very fast as it helps people buy the desired product more quickly and comfortably. The fierce competition among online retail store providers gives a rise to technology development. Many online retail systems not only display the product but also need to be supported by proper products selection to attract the attention of website visitors. As a result, many website visitors are confused when they are going to buy products in online store. The number of product variety offered to a customer when he/she buys goods is sometimes more than one product.This report proposed an online store product recommendation system with the application of Apriori algorithm towards a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The project begins with pre-processing the data and data transformation, then the cleaned data will be tested with the Apriori algorithm to capture the customers' preference. Thus, hopefully, by identifying the customers' preferences, a valid product recommendation can be developed.

Key words: online retail, apriori algorithm, data mining, association rules, customer-centric marketing

I. INTRODUCTION

Compared with traditional shopping in retail stores, online shopping has some unique characteristics: each customer's shopping process and activities can be tracked instantaneously and accurately, each customer's order is usually associated with a delivery address and a billing address, and each customer has an online store account with essential contact and payment information. These desirable, special online shopping characteristics have enabled online retailers to treat each customer as an individual with personalized understanding of each customer and to build upon customer-centric business intelligence.

In this report, a case study of using data mining techniques in customer-centric business intelligence for an online retailer is presented. The online retailer considered here is a typical one: a small business and a relatively new entrant to the online retail sector, knowing the growing importance of being analytical in today's online businesses and data mining techniques, however, lacking technical awareness and recourses. The main purpose of this analysis is to help the business better understand its customers and therefore conduct product recommendation system which helps attract the attention of website visitors.

The rest of this article is organized as follows. The next section provides the background information about the online retailer along with the associated dataset to be explored. The section after that discusses in detail about the main steps and tasks for data pre-processing in order to create an appropriate target dataset for the required further analyses. After that, Apriori Algorithm and Association Rules will be applied. Finally, the concluding remarks are given in the last section.

II. MATERIAL AND METHOD

This section explains the study area, the process of the research and also the data used and the algorithm that had been applied.

A. Study Area

The online retailer under consideration in this project is a UK-based and registered non-store business with some 80 members of staff. The company was established in 1981 mainly selling unique all-occasion gifts. For years in the past, the merchant relied heavily on direct mailing catalogues, and orders were taken over phone calls. It was 11 years ago that the company launched its own web site and shifted completely to the Web. Since then the company has maintained a steady and healthy number of customers from all parts of the United Kingdom and Europe, and has accumulated a huge amount of data about many customers. The company also uses Amazon.co.uk to market and sell its products.

The customer transaction dataset held by the merchant has 8 variables as shown in **Table 1**. From 1 December 2010 to 9 December 2011, there were 541909 instances (record rows) in the dataset, each for a particular item contained in a transaction.

Table 1: Variables in the customer transaction dataset

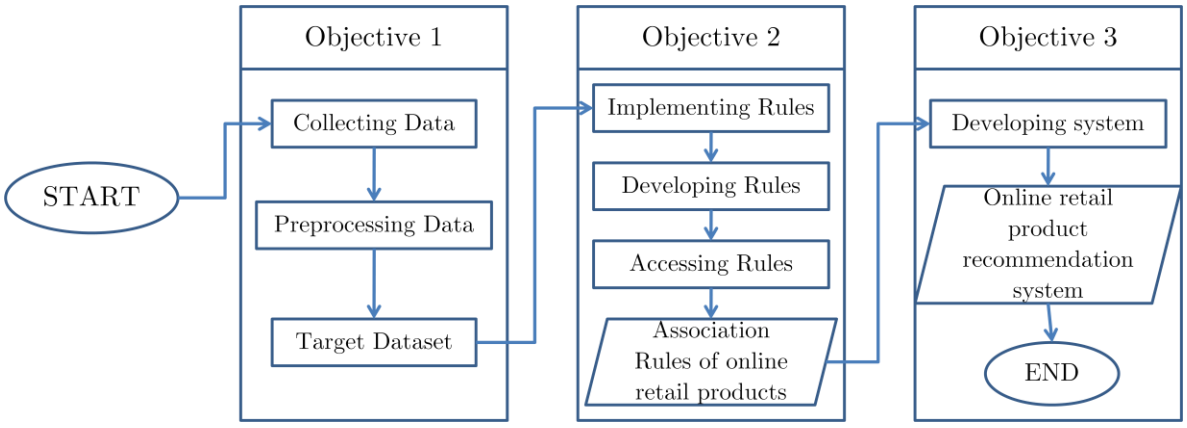
Variable name	Data type	Descriptions, typical values and meanings
InvoiceNo	Nominal	A 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation
StockCode	Nominal	A 5-digit integral number uniquely assigned to each distinct product
Description	Nominal	Product (item) name
Quantity	Numeric	The quantities of each product (item) per transaction
InvoiceDate	Numeric	The day and time when each transaction was generated. Example from dataset: 12/1/2010 8:26
UnitPrice	Numeric	Product price per unit in sterling
CustomerID	Nominal	A 5-digit integral number uniquely assigned to each customer
Country	Nominal	The name of the country where each customer resides

B. Project Workflow

In order to accomplish the project objectives, a workflow is designed with the project approach.

According to Figure 1, the project starts with the collecting online retail data from UCI Machine Learning Repository. Some invalid transactions were removed to create the desired dataset. Then, this dataset will be integrated to develop a dataset to be mined using Apriori Algorithm. The project finds that the output of phase 1 is the collection of 536493 valid transactions from the original dataset which contains 541909 rows. Moving to phase two, the Apriori algorithm will be run using the target dataset to generate association rules of online retail products. Finally, the rules will be used at phase three to create the online retail product recommendation system.

Figure 1: Proposed workflow of online retail product recommendation system



III. RESULTS AND DISCUSSION

1. LOADING LIBRARIES AND IMPORTING DATA

Firstly, we will load the libraries required. A short description of the libraries is given in the Table 2 as follows:

Table 2: Required libraries and their meaning

Library	Meaning
time	Calculate execution time
pandas	Dataframe functions
numpy	Numerical functions
string.punctuation	Punctuation removal
nltk.corpus.stopwords	Stopwords removal
matplotlib.pyplot	Plotting graphs
mlxtend.frequent_patterns.apriori	Apriori
mlxtend.frequent_patterns.association_rules	Association Rules
apyori.apriori	Apriori
warnings	Ignore warnings
IPython.core.display	Display multiple dataframes as once

2. CLEANING THE DATASET

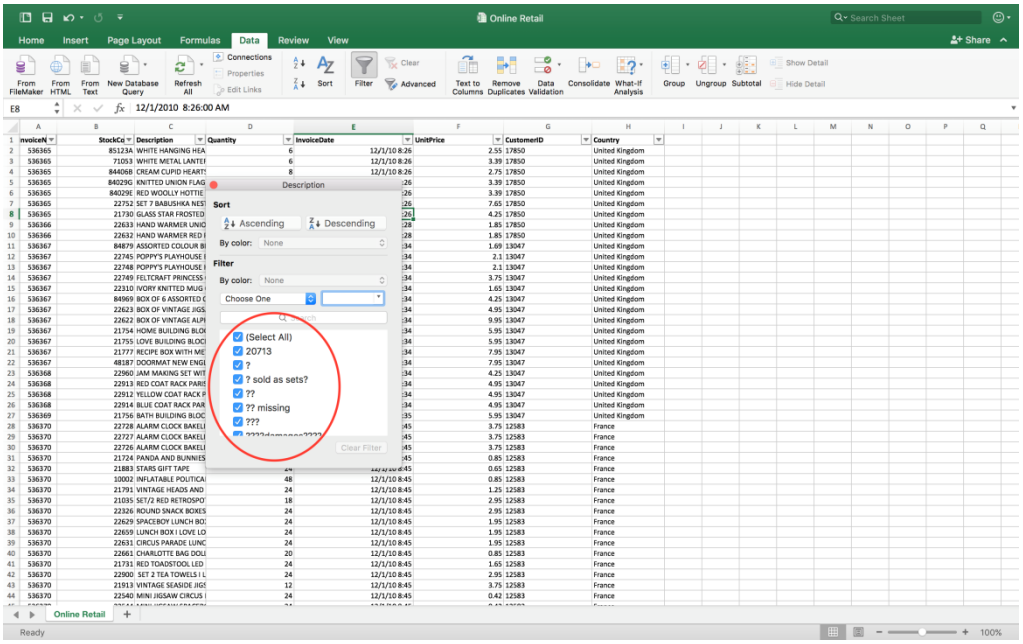
After importing the dataset 'Online Retail.xlsx' into Python, we pay attention to each of the attributes. We decided to import the dataset in R and use function summary()to have a quick overview about 541909 transactions.

```
 InvoiceNo          StockCode          Description          Quantity            InvoiceDate
Length:541909      Length:541909      Length:541909      Min.   :-80995.00   Min.   :2010-12-01 08:26:00
Class :character   Class :character   Class :character   1st Qu.:     1.00   1st Qu.:2011-03-28 11:34:00
Mode  :character   Mode  :character   Mode  :character   Median :     3.00   Median :2011-07-19 17:17:00
                                                         Mean   :     9.55   Mean   :2011-07-04 13:34:57
                                                         3rd Qu.:    10.00   3rd Qu.:2011-10-19 11:27:00
                                                         Max.   : 80995.00   Max.   :2011-12-09 12:50:00

   UnitPrice          CustomerID       Country
Min.   :-11062.06   Min.   :12346   Length:541909
1st Qu.:     1.25   1st Qu.:13953   Class :character
Median :     2.08   Median :15152   Mode  :character
Mean   :     4.61   Mean   :15288
3rd Qu.:     4.13   3rd Qu.:16791
Max.   : 38970.00   Max.   :18287
                    NA's   :135080
```

Immediately, we figured out that there were some transactions having negative number of Quantity and UnitPrice, so it is necessary to remove them in our dataset by writing two code lines in Python below:

```
# Remove non-transactions (dumps/refunds)
df = df[df.UnitPrice>0]
df = df[df.Quantitty>0]
```

Besides, a closer look at Filter in Excel in Online Retail.xlsx reveals some unidentified characters in Description column.



Hence, the next step is to remove transactions with NULL description with the code line below:

```
# Remove transactions with NULL description
df = df[df.Description.notnull()]
```

Moreover, we could figure out that there are some special characters such as "+", "&", "/" existing in some transactions of the dataset through R.

Filter

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
	All	All	/	All		All	All	All
537807	536943	85018D	YULETIDE IMAGES S/6 PAPER BOXES	1	2010-12-03 12:11:00	2.55	17884	United Kingdom
537808	537047	85018D	YULETIDE IMAGES S/6 PAPER BOXES	1	2010-12-05 11:02:00	2.55	13069	United Kingdom
537809	537135	85018D	YULETIDE IMAGES S/6 PAPER BOXES	6	2010-12-05 12:35:00	2.55	17059	United Kingdom
537810	537136	85018D	YULETIDE IMAGES S/6 PAPER BOXES	1	2010-12-05 12:42:00	2.55	12748	United Kingdom
537811	538635	85018D	YULETIDE IMAGES S/6 PAPER BOXES	1	2010-12-13 13:32:00	2.55	17303	United Kingdom
537812	539953	85018D	YULETIDE IMAGES S/6 PAPER BOXES	1	2010-12-23 12:03:00	2.55	12748	United Kingdom
537813	541223	85018D	YULETIDE IMAGES S/6 PAPER BOXES	1	2011-01-14 14:39:00	2.55	17954	United Kingdom
537228	536945	37489A	YELLOW/PINK FLOWER DESIGN BIG MUG	1	2010-12-03 12:24:00	1.95	14083	United Kingdom
537229	550626	37489A	YELLOW/PINK FLOWER DESIGN BIG MUG	4	2011-04-19 14:23:00	0.39	13184	United Kingdom
537230	553657	37489A	YELLOW/PINK FLOWER DESIGN BIG MUG	2	2011-05-18 11:28:00	0.39	14583	United Kingdom
537231	554682	37489A	YELLOW/PINK FLOWER DESIGN BIG MUG	4	2011-05-25 16:08:00	0.39	14085	United Kingdom
537232	555853	37489A	YELLOW/PINK FLOWER DESIGN BIG MUG	1	2011-06-07 13:44:00	0.39	14375	United Kingdom
537233	559165	37489A	YELLOW/PINK FLOWER DESIGN BIG MUG	2	2011-07-06 16:40:00	0.39	12977	United Kingdom

Showing 1 to 13 of 13,227 entries, 8 total columns (filtered from 541,909 total entries)

Filter

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
	All	All	+	All		All	All	All
537160	541265	84461	12 PINK HEN+CHICKS IN BASKET	1	2011-01-16 16:23:00	2.55	17609	United Kingdom
537161	544074	84461	12 PINK HEN+CHICKS IN BASKET	1	2011-02-15 14:49:00	2.55	14156	EIRE
537162	546261	84461	12 PINK HEN+CHICKS IN BASKET	1	2011-03-10 15:01:00	2.55	17841	United Kingdom
537163	546365	84461	12 PINK HEN+CHICKS IN BASKET	6	2011-03-11 11:35:00	2.55	12520	Germany
537164	C546886	84461	12 PINK HEN+CHICKS IN BASKET	-6	2011-03-17 18:13:00	2.55	12520	Germany
537165	547243	84461	12 PINK HEN+CHICKS IN BASKET	1	2011-03-21 16:59:00	2.55	17231	United Kingdom
537166	547374	84461	12 PINK HEN+CHICKS IN BASKET	1	2011-03-22 13:54:00	2.55	16110	United Kingdom
537167	547672	84461	12 PINK HEN+CHICKS IN BASKET	1	2011-03-24 13:54:00	2.55	17754	United Kingdom
537168	548192	84461	12 PINK HEN+CHICKS IN BASKET	2	2011-03-29 15:22:00	2.55	17364	United Kingdom
537169	548317	84461	12 PINK HEN+CHICKS IN BASKET	6	2011-03-30 12:43:00	2.55	18179	United Kingdom
537170	548664	84461	12 PINK HEN+CHICKS IN BASKET	12	2011-04-01 14:45:00	2.55	14911	EIRE
537171	548708	84461	12 PINK HEN+CHICKS IN BASKET	1	2011-04-03 12:41:00	2.55	13269	United Kingdom
537172	549059	84461	12 PINK HEN+CHICKS IN BASKET	1	2011-04-06 10:46:00	2.55	15529	United Kingdom

Showing 1 to 13 of 6,008 entries, 8 total columns (filtered from 541,909 total entries)

Filter

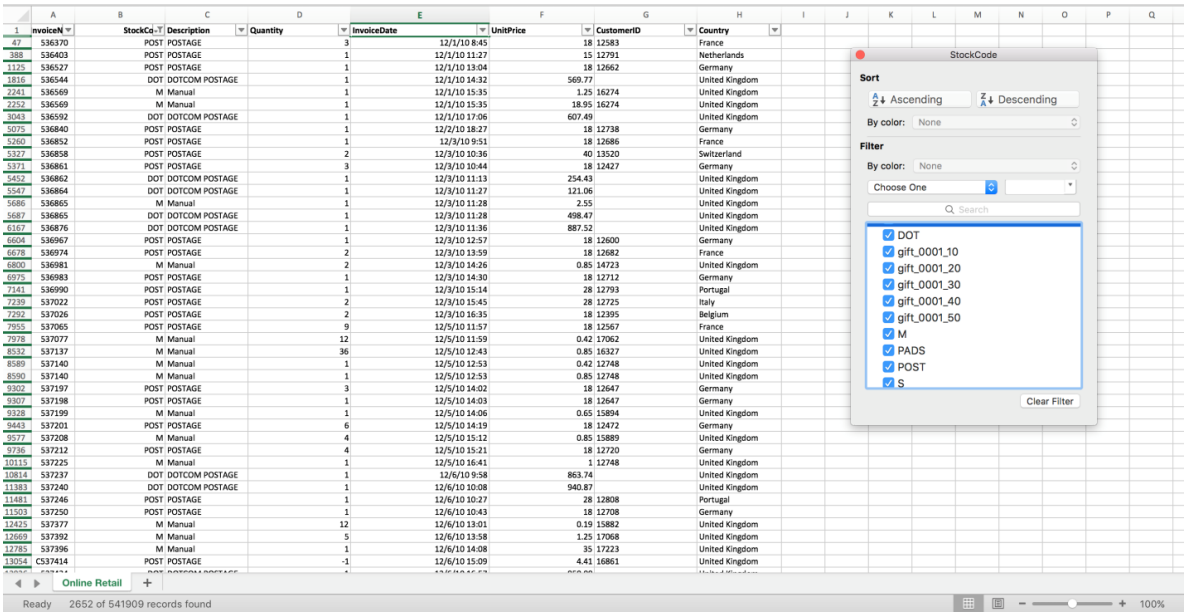
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
	All	All	&	All		All	All	All
492573	537195	84563B	BLUE & WHITE BREAKFAST TRAY	1	2010-12-05 13:55:00	5.95	15311	United Kingdom
492574	539958	84563B	BLUE & WHITE BREAKFAST TRAY	1	2010-12-23 13:26:00	4.21	NA	United Kingdom
492575	540255	84563B	BLUE & WHITE BREAKFAST TRAY	1	2011-01-05 16:50:00	4.21	NA	United Kingdom
492576	540468	84563B	BLUE & WHITE BREAKFAST TRAY	1	2011-01-07 13:55:00	4.21	NA	United Kingdom
492577	540551	84563B	BLUE & WHITE BREAKFAST TRAY	2	2011-01-10 09:43:00	4.21	NA	United Kingdom
492578	540646	84563B	BLUE & WHITE BREAKFAST TRAY	1	2011-01-10 14:32:00	4.21	NA	United Kingdom
492579	540681	84563B	BLUE & WHITE BREAKFAST TRAY	1	2011-01-10 16:25:00	4.21	NA	United Kingdom
492580	540821	84563B	BLUE & WHITE BREAKFAST TRAY	1	2011-01-11 13:16:00	4.21	NA	United Kingdom
492581	540848	84563B	BLUE & WHITE BREAKFAST TRAY	1	2011-01-12 09:26:00	4.21	NA	United Kingdom
492582	541219	84563B	BLUE & WHITE BREAKFAST TRAY	2	2011-01-14 14:06:00	4.13	NA	United Kingdom
492583	541421	84563B	BLUE & WHITE BREAKFAST TRAY	1	2011-01-17 17:44:00	4.13	NA	United Kingdom
492584	541497	84563B	BLUE & WHITE BREAKFAST TRAY	1	2011-01-18 15:19:00	4.13	NA	United Kingdom
492585	541516	84563B	BLUE & WHITE BREAKFAST TRAY	1	2011-01-18 17:34:00	4.13	NA	United Kingdom

Showing 1 to 13 of 2,718 entries, 8 total columns (filtered from 541,909 total entries)

In order to remove characters “+”, “&”, “/”, we wrote these following code lines:

```
# Replace special characters with ' '
df["Description"] = df["Description"].str.replace("/"," ")
df["Description"] = df["Description"].str.replace("&"," ")
df["Description"] = df["Description"].str.replace("+"," ")
```

Furthermore, in the StockCode column, we observed that there existed some transactions with non-purchase stock code by using Filter in Excel.



This also includes any gift card items, since each gift card has a unique item code, we need to remove all transactions whose StockCode starts with “gift”.

```
# Remove transactions with non-purchase stock codes
df = df[~df.StockCode.isin(["D", "DOT", "S", "POST", "M", "C2", "B", "m"])]
df = df[~df.StockCode.isin(["AMAZONFEE", "BANK CHARGES", "CRUK", "PADS"])]
df = df[~df.StockCode.apply(str).str.contains("gift")]
```

After deleting some invalid transactions, a target dataset for the analysis has been generated. The original dataset which has 541909 rows was transformed into the final target dataset which has 536493 rows. Part of the target dataset is shown in Figure 1 after using ICD.display().

```
# Re-enumerate indexes due to deleted transactions
df.reset_index(drop = True, inplace = True)
ICD.display(df)
print("--- Executed in %.5s seconds ---" % (time.time() - start_time))
```

Figure 2: Part of the target dataset

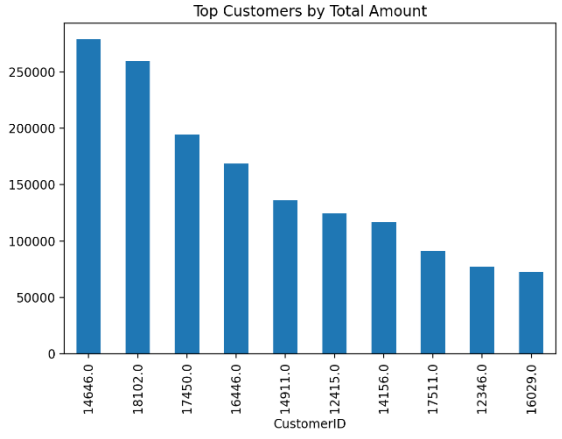
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
536488	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
536489	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
536490	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
536491	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
536492	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

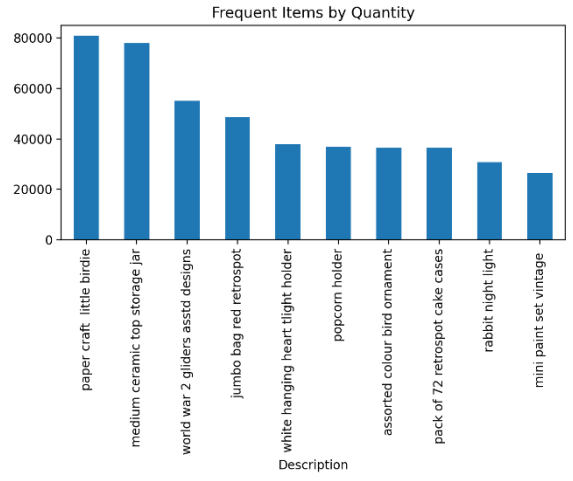
[536493 rows x 8 columns]
--- Executed in 66.41 seconds ---

3. SUMMARIZING DATA

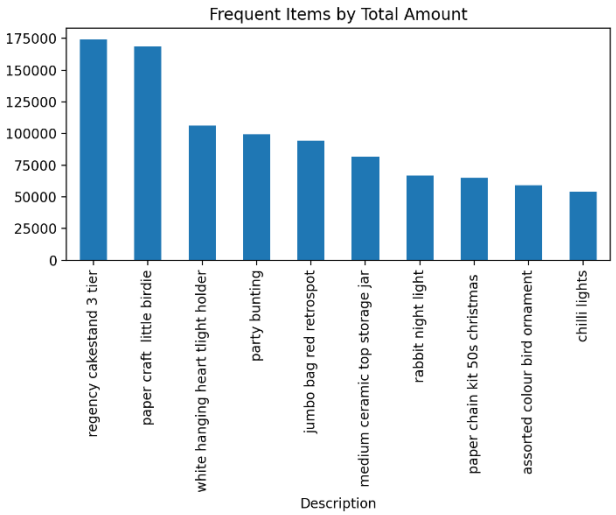
To find out some interesting insights behind the dataset, we plot 8 graphs that can demonstrate some helpful information as follows:

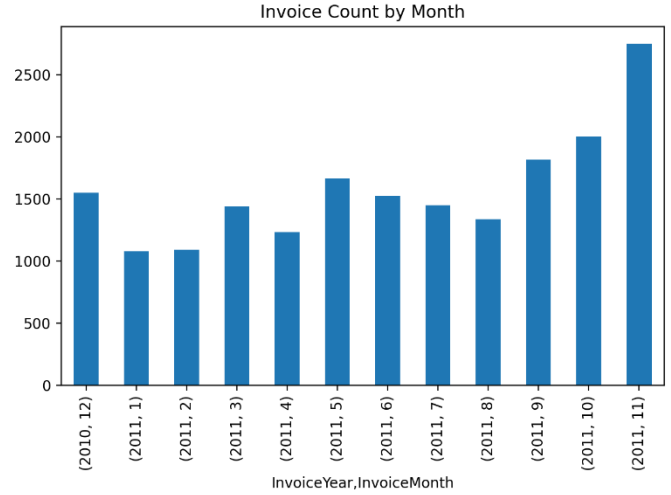
- Top 10 customers based on the total amount spent
- Top 10 most frequently purchased items based on total quantity
- Top 10 most frequently purchased items based on total amount
- Top 10 months having the most invoices
- Top 10 months having the highest profit
- Top 10 countries based on the total amount
- Top 10 countries based on the total number of invoices
- Top 10 countries based on the total number of customers purchasing products

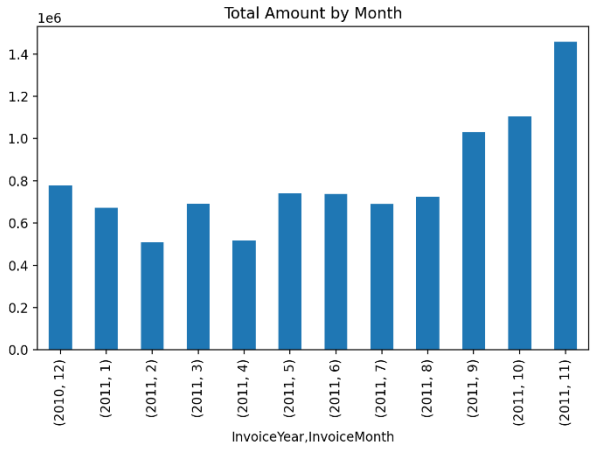


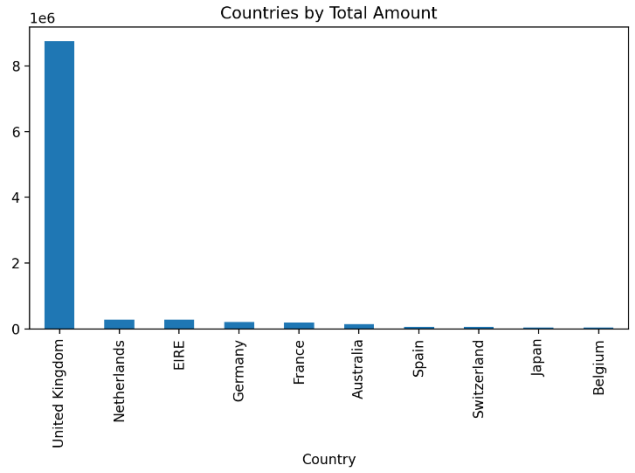


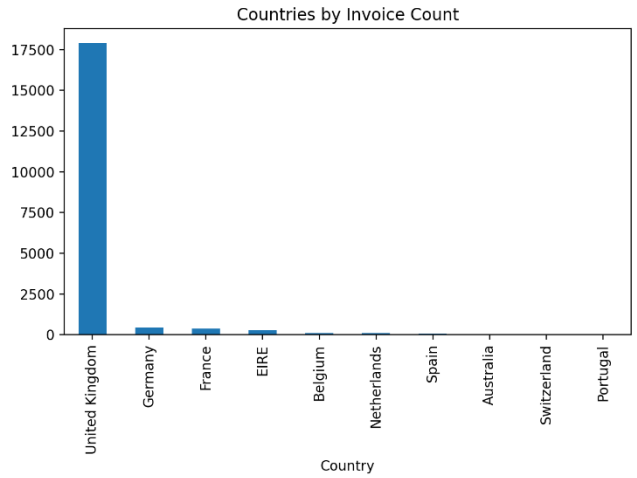
Online Retail Product Recommendation System Uses Apriori Algorithm

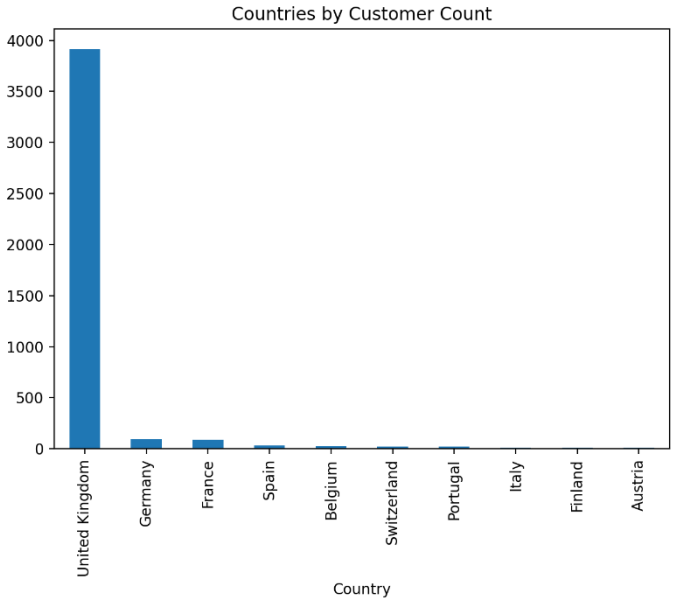












4. PREPROCESSING DATA

4.1: Transform dataframe into basket format

Before using Apriori algorithm, we need to transform data from the data frame format into basket format as Figure 3 below.

```
def ap1_transform_data(df_original):
    invoice = ''
    transactions = []
    for index, value in enumerate(df_original['InvoiceNo']):
        if invoice != value:
            invoice = value
            transactions.append([df_original['Description'][index]])
            continue
        transactions[-1].append(df_original['Description'][index])
    df_transform = pd.DataFrame(transactions)
    df_transform.fillna(value = pd.np.nan, inplace = True)
    return df_transform
```

Figure 3: Pattern of Customer Purchase Transaction

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	WHITE HANGING HEART T-LIGHT HOLDER	WHITE METAL LANTERN	CREAM CUPID HEARTS COAT HANGER	KNITTED UNION FLAG HOT WATER BOTTLE	RED WOOLLY HOTTIE WHITE HEART.	SET 7 BABUSHKA NESTING BOXES	GLASS STAR FROSTED T-LIGHT HOLDER	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	HAND WARMER UNION JACK	HAND WARMER RED POLKA DOT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	ASSORTED COLOUR BIRD ORNAMENT	POPPY'S PLAYHOUSE BEDROOM	POPPY'S PLAYHOUSE KITCHEN	FELTCRAFT PRINCESS CHARLOTTE DOLL	IVORY KNITTED MUG COSY	BOX OF 6 ASSORTED COLOUR TEASPOONS	BOX OF VINTAGE JIGSAW BLOCKS	BOX OF VINTAGE ALPHABET BLOCKS	HOME BUILDING BLOCK WORD	LOVE BUILDING BLOCK WORD	RECIPE BOX WITH METAL HEART	DOORMAT NEW ENGLAND	NaN	NaN	NaN
3	JAM MAKING SET WITH JARS	RED COAT RACK PARIS FASHION	YELLOW COAT RACK PARIS FASHION	BLUE COAT RACK PARIS FASHION	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	BATH BUILDING BLOCK WORD	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19768	LUNCH BAG RED RETROSPOT	6 CHOCOLATE LOVE HEART T-LIGHTS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19769	RED FLOCK LOVE HEART PHOTO FRAME	6 CHOCOLATE LOVE HEART T-LIGHTS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19770	BLACK TEA TOWEL CLASSIC DESIGN	ASSORTED BOTTLE TOP MAGNETS	VICTORIAN GLASS HANGING T-LIGHT	EMBOSSED GLASS TEALIGHT HOLDER	ZINC WILLIE WINKIE CANDLE STICK	RABBIT NIGHT LIGHT	ASSORTED COLOUR BIRD ORNAMENT	MULTI COLOUR SILVER T-LIGHT HOLDER	GREY HEART HOT WATER BOTTLE	LOVE HOT WATER BOTTLE	ALARM CLOCK BAKELIKE GREEN	ALARM CLOCK BAKELIKE RED	LARGE CHINESE STYLE SCISSOR	SET 12 RETRO WHITE CHALK STICKS	BOX OF 24 COCKTAIL PARASOLS
19771	LARGE CAKE STAND HANGING STRAWBERY	SET OF 3 HANGING OWLS OLLIE BEAK	RED RETROSPOT ROUND CAKE TINS	DOORMAT RED RETROSPOT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19772	CIRCUS PARADE LUNCH BOX	PLASTERS IN TIN CIRCUS PARADE	PLASTERS IN TIN STRONGMAN	ALARM CLOCK BAKELIKE PINK	ALARM CLOCK BAKELIKE RED	ALARM CLOCK BAKELIKE GREEN	ALARM CLOCK BAKELIKE IVORY	CHILDRENS APRON SPACEBOY DESIGN	SPACEBOY LUNCH BOX	CHILDRENS CUTLERY SPACEBOY	PACK OF 20 SPACEBOY NAPKINS	CHILDREN'S APRON DOLLY GIRL	CHILDRENS CUTLERY DOLLY GIRL	CHILDRENS CUTLERY CIRCUS PARADE	BAKING SET 9 PIECE RETROSPOT

19773 rows × 1112 columns

4.2: Text mining

For our dataset, we need to have to add several text pre-processing steps, including theremoval of punctuations and removal of stopwords.

```
PUNCT = string.punctuation

def remove_punctuation(text):
    return text.translate(str.maketrans('', '', PUNCT))

", ".join(stopwords.words('english'))
STOPW = set(stopwords.words('english'))
```

```
def remove_stopwords(text):
    return " ".join([word for word in str(text
                                          ).split() if word not in ST
OPW])

def preprocessing(dataframe):
    for column in dataframe:
        col = dataframe[column]
        col = col.astype(str)
        col = col.str.lower()
        col = col.apply(lambda text: remove_punctuation(text))
        col = col.apply(lambda text: remove_stopwords(text))
        dataframe[column] = col
```

5. TRANSACTION DATA MATRIX FROM ONE HOT ENCODING

Machine learning algorithms cannot work with categorical data directly so it must be converted to numbers.Customer purchase transaction data will be adjusted in matrix format to get the pattern of goods formed and used to know how many times the goods are purchased by the customer on each transaction, the following data is presented in Figure 4 below. Because the number of products in our dataset is too large, most of the number in the transaction data matrix is 0.

```
def one_hot_encoding(dataframe, items):
    itemset = set(items)
    envals = []
    for idx, row in dataframe.iterrows():
        rowset = set(row)
        labels = {}
        uncomms = list(itemset - rowset)
        commons = list(itemset.intersection(rowset))
        for uc in uncomms: labels[uc] = 0
        for cm in commons: labels[cm] = 1
        envals.append(labels)
    return envals
```

Figure 4: Part of the customer purchase transaction matrix

	white bell honeycomb paper garland	36 pencils tube woodland	set 10 cards dressing 17077	wood stamp set flowers	french wc sign blue metal	12 pink hen chicks basket	pink cat floral cushion cover	70s alphabet wall art	set 4 napkin charms cutlery	wrap poppies design	cotton apron pantry design	white heart confetti tube	champagne tray blank card	frying pan pink polkadot	felt farm animal rabbit	bird decoration green polkadot	potting shed candle citronella	scottie dog hot water bottle	yellow blue retro radio	vintage jingle bells heart	pink fluffy christmas decoration	pack 12 skull tissues	ceramic cake stand hanging cakes	12 pencils tall tube woodland	glow dark dolphins	set 12 mini loaf baking cases
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19768	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19769	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19770	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19771	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19772	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6. APPLYING APRIORI ALGORITHM

Basically, the association rules apply if/then statements to discover the relationship between unrelated data in the data repository. The basic formula of association rules is

$$A => B \ (1)$$

A will be the 'if statement' and B will be the 'then statement'. In association rules, A is the antecedent and B is the consequent. Support and Confidence are calculated to analyze which one is the best rules.

$$\text{Support} = P(A \cap B) = \frac{number\ \ of\ transactions\ \ containing\ \ both\ A\ and\ B}{total\ \ number\ \ of\ transactions} \ (2)$$

The support for a particular association rule A => B is the proportion of transactions in D that contain both A and B. The rule indicates how frequently the items in the rule occurred together.

$$\text{Confidence} = P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{number\ \ of\ transactions\ \ containing\ \ both\ A\ and\ B}{number\ \ of\ transactions\ \ containing\ \ A} \ (3)$$

The confidence of the association rule A => B is a measure of the accuracy of the rule. It is determined by the percentage of the transaction containing A that also contains B. It is the conditional probability of the consequent given the antecedent.

On the other hand, to find the best rules, we will consider rules that have high support or high confidence and usually both. Rules that categorized as strong rules or best rules are rules which surpass certain minimum support (called minsup) and minimum confidence (called minconf) that we specified earlier.

In obtaining the rules, there are two major steps applied, which are:

• Find all sets of items that have support value greater than the minimum support. In this project, we set minsup = 0.01. These item sets are called large item sets. All others are called small item sets.

• Use the large item sets to generate the desired rules.

Additionally, we set up the rank of rules that will be generated according to lift metric. Lift use to measure the quality and interestingness of the rule. In this research lift is the third measure to evaluate the quality of the rule after the support and confidence. Lift is defined as

$$\text{Lift (A => B)} = \frac{support\ (A \cup B)}{(support\ (antecedent\ ) \times suppo\ rt\ (consequence\ ))} \quad (4)$$

If some rule had a lift of 1, it showed that the probability of the relationship between antecedent and consequent are independent of each other. When two events independent, hence no rule can be haggard involving the events. Besides, if the lift is bigger than 1 value, the two occurrences are dependent to one another. Likewise, the rules are potentially useful for predicting consequences in future data sets.

Lastly, in data interpretation phase, the patterns or rules or information generated from the phases before are then will be interpreted. Then, based on the best rules obtained, we can develop an online retail product recommendation system.

```
Apriori(T, ε)
    L1 ← {large 1 - itemsets}
    k ← 2
    while Lk-1 is not empty
        Ck ← Apriori_gen(Lk-1, k)
        for transactions t in T
            Dt ← {c in Ck : c ⊆ t}
            for candidates c in Dt
                count[c] ← count[c] + 1

        Lk ← {c in Ck : count[c] ≥ ε}
        k ← k + 1

    return Union(Lk)

Apriori_gen(L, k)
     result ← list()
     for all p ⊆ L, q ⊆ L where p1 = q1, p2 = q2, ..., pk-2 = qk-2 and pk-1 < qk-1
         c = p ∪ {qk-1}
         if u ⊆ c for all u in L
             result.add(c)
      return result
```

Table 3 and **Table 4** indicate 10 rules generated after applying Apriori algorithm.

Table 3: Top 10 rules by support

	support	itemsets
0	0.114095	(white hanging heart tlight holder)
1	0.105649	(jumbo bag red retrospot)
2	0.100541	(regency cakestand 3 tier)
3	0.085217	(party bunting)
4	0.079098	(lunch bag red retrospot)
5	0.073585	(assorted colour bird ornament)
6	0.070045	(set 3 cake tins pantry design)
7	0.066758	(pack 72 retrospot cake cases)
8	0.064937	(lunch bag suki design)
9	0.064381	(lunch bag black skull)

From the table, we can observe that:

- White hanging heart tlight holder appears in 11.4% of the whole transactions.
- Jumbo bag red retrospot appears in 10.5% of the whole transactions.
- Regency cakestand 3 tier appears in 10.05% of the whole transactions.
- Party bunting appears in 8.52% of the whole transactions.
- Lunch bag red retrospot appears in 7.9% of the whole transactions.
- Assorted colour bird ornament appears in 7.35% of the whole transactions.
- Set 3 cake tins pantry design appears in 7% of the whole transactions.
- Pack 72 retrospot cake cases appears in 6.67% of the whole transactions.
- Lunch bag suki design appears in 6.49% of the whole transactions.
- Lunch bag black skull appears in 6.44% of the whole transactions.

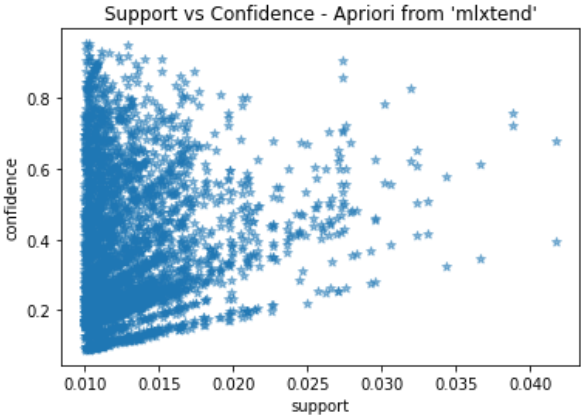
Table 4: Top 10 rules ordered by descending lift

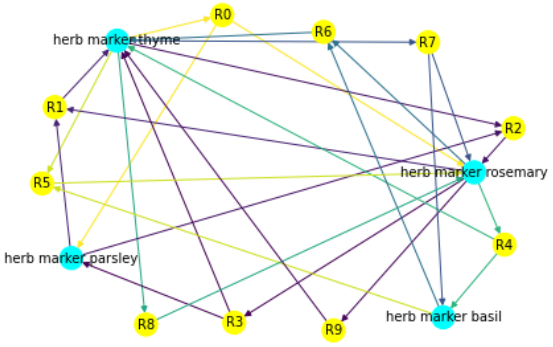
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
2884	(herb marker thyme)	(herb marker parsley, herb marker rosemary)	0.011986	0.010873	0.010267	0.856540	78.773800	0.010136	6.894794
2881	(herb marker parsley, herb marker rosemary)	(herb marker thyme)	0.010873	0.011986	0.010267	0.944186	78.773800	0.010136	17.701917
2880	(herb marker parsley, herb marker thyme)	(herb marker rosemary)	0.010772	0.012138	0.010267	0.953052	78.519542	0.010136	21.041466
2885	(herb marker rosemary)	(herb marker parsley, herb marker thyme)	0.012138	0.010772	0.010267	0.845833	78.519542	0.010136	6.416612
3042	(herb marker rosemary)	(herb marker thyme, herb marker basil)	0.012138	0.010621	0.010115	0.833333	78.464286	0.009986	5.936277
3039	(herb marker thyme, herb marker basil)	(herb marker rosemary)	0.010621	0.012138	0.010115	0.952381	78.464286	0.009986	20.745107
3040	(herb marker rosemary, herb marker basil)	(herb marker thyme)	0.010823	0.011986	0.010115	0.934579	77.972318	0.009985	15.102499
3041	(herb marker thyme)	(herb marker rosemary, herb marker basil)	0.011986	0.010823	0.010115	0.843882	77.972318	0.009985	6.336081
2038	(herb marker thyme)	(herb marker rosemary)	0.011986	0.012138	0.011177	0.932489	76.825475	0.011031	14.632709
2039	(herb marker rosemary)	(herb marker thyme)	0.012138	0.011986	0.011177	0.920833	76.825475	0.011031	12.480176

Top 10 rules with descending lift order are listed as below:

- **Rule 1:** Herb marker thyme => Herb marker parsley & Herb marker rosemary
- **Rule 2:** Herb marker parsley & Herb marker rosemary => Herb marker thyme
- **Rule 3:** Herb marker parsley & Herb marker thyme => Herb marker rosemary
- **Rule 4:** Herb marker rosemary => Herb marker parsley & Herb marker thyme
- **Rule 5:** Herb marker rosemary => Herb marker thyme & Herb marker basil
- **Rule 6:** Herb marker thyme & Herb marker basil => Herb marker rosemary
- **Rule 7:** Herb marker rosemary & Herb marker basil => Herb marker thyme
- **Rule 8:** Herb marker thyme => Herb marker rosemary & Herb marker basil
- **Rule 9:** Herb marker thyme => Herb marker rosemary
- **Rule 10:** Herb marker rosemary => Herb marker thyme

7. VISUALIZING RESULTS





IV. CONCLUSIONS

Based on the results and discussion part, it can be concluded that by applying the Apriori algorithm, the system provides product recommendations to Online Retail customers based on the trust value of a combination of products purchased at a given time period. Application of Apriori method in this project is to find the most combination of items based on transaction data and then form the association pattern of item combination. This research can be further developed using a weighted product method to get the weight of each product brand so that it can provide an alternative to the most recommended product brands.

Latent Semantics Analysis (LSA) Case Study

1. Introduction

One of the primary applications of natural language processing is to automatically extract what topics people are discussing from large volumes of text. Some examples of large text could be feeds from social media, customer reviews of hotels, movies, etc, user feedbacks, news stories, e-mails of customer complaints etc.

Knowing what people are talking about and understanding their problems and opinions is highly valuable to businesses, administrators, political campaigns. And it's really hard to manually read through such large volumes and compile the topics.

Thus, is required an automated algorithm that can read through the text documents and automatically output the topics discussed.

In this report, we will take group9 dataset and use LSA to extract the naturally discussed topics.

I will be using the Latent Semantic Analysis (LSA) from Gensim package along with the We will also extract the volume and percentage contribution of each topic to get an idea of how important a topic is.

2. Core packages

Core packages to encounter LSA problem are NLTK and Gensim:

- The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.

- Gensim is an open-source library for unsupervised topic modelling and natural language processing, using modern statistical machine learning. Gensim is implemented in Python for performance. Gensim is designed to handle large text collections using data streaming and incremental online algorithms, which differentiates it from most other machine learning software packages that target only in-memory processing.

3. Import Packages

Other than mentioned in part 2., matplotlib, numpy and pandas for data handling and visualization and other trivial packages are all included in the following code:

```
# Import Libraries
importtime
start_time = time.time()

importnumpyasnp
importpandasaspd
pd.options.mode.chained_assignment = None
importio
importrequests
importpkg_resources
importwarnings
warnings.filterwarnings('ignore')

importnltk

fromnltkimportword_tokenize
fromnltk.probabilityimportFreqDist

fromnltk.corpusimportstopwords
fromcollectionsimportCounter
fromnltk.stem.porterimportPorterStemmer
fromnltk.stemimportWordNetLemmatizer
fromnltk.tokenizeimportRegexpTokenizer

fromcollectionsimportCounter
fromwordcloudimportWordCloud
importre
importstring
frombs4importBeautifulSoup

fromsymspellpy.symspellpyimportSymSpell

fromautocorrectimportSpeller

fromgensimimportcorpora
fromgensim.modelsimportLsiModel
fromgensim.models.coherencemodelimportCoherenceModel

importmatplotlib.pyplotasplt
fromIPython.coreimportdisplayasICD
```

4. What does LSA do?

LSA's approach to topic modeling is it considers each document as a collection of topics in a certain proportion. And each topic as a collection of keywords, again, in a certain proportion.
Once we provide the algorithm with the number of topics, all it does it to rearrange the topics distribution within the documents and keywords distribution within the topics to obtain a good composition of topic-keywords distribution.

A topic is a collection of dominant keywords that are typical representatives. Just by looking at the keywords, we can identify what the topic is all about.

The following are key factors to obtaining good segregation topics:

- The quality of text processing.
- The variety of topics the text talks about.
- The choice of topic modeling algorithm.
- The number of topics fed to the algorithm.
- The algorithms tuning parameters.

5. Import data

We will be using the group9 dataset. This dataset contains 17 columns and 10322 rows. We are only interested in columns 5 and 17 which are interpretable.

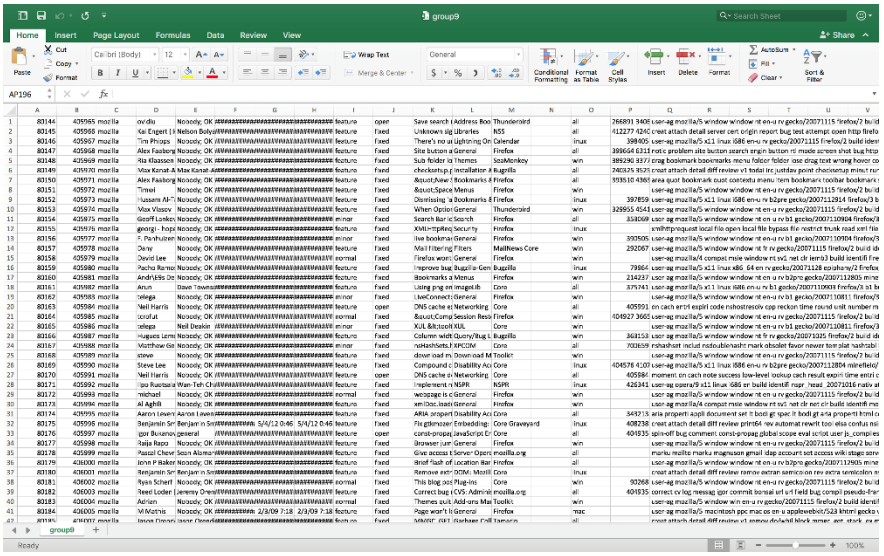


Figure 1: Group9 data set viewed in Excel.

This is imported using pandas.read_csv. But before using pandas, we shall need to upload the data file in GitHub to get URL for downloading file:

```
url = https://raw.githubusercontent.com/ThanhDatIU/File-CSV-
Storage/main/group9.csv
s = requests.get(url).content
df = pd.read_csv(io.StringIO(s.decode('utf-8')), header = None)
df = df[[4, 16]]
df.rename({4: 'Title', 16: 'Description'}, axis = 'columns', inplace = True)
ICD.display(df)
```

We use Google Collab to execute Python. Running the above lines, we get the data frame:

	Title	Description
0	Nobody; OK to take it and work on it	user-ag mozilla/5 window window nt en-u rv gec...
1	Nelson Bolyard (seldom reads bugmail)	creat attach detail server cert origin report ...
2	Nobody; OK to take it and work on it	user-ag mozilla/5 x11 linux i686 en-u rv gecko...
3	Nobody; OK to take it and work on it	notic problem site button search engin button ...
4	Nobody; OK to take it and work on it	drag bookmark bookmarks-menu folder folder los...
...	...	...
10294	Dave Townsend (:Mossop)	extens instal pointer file directori point ext...
10295	Taras Glek (:taras)	creat attach detail diff review replac quot en...
10296	Zack Weinberg (:zwol)	creat attach detail testcas testcas crash trun...
10297	Aravind Gottipati [:aravind]	backup khan-vm khan bug info khan khan khan mo...
10298	Nobody; OK to take it and work on it	user-ag mozilla/5 x11 linux i686 en-u rv b4pre...

10299 rows × 2 columns

--- Executed in 1.689 seconds ---

Figure 2: Group 9 data frame viewed in Python

6. Pre-processing data:

6.1. Punctuation and Numbers:

Punctuation is not applied for Title because punctuations have certain meaning in those columns. Additionally, numbers have no interpretation in Description so too be removed.

```
# Remove Punctuations
PUNCT = string.punctuation
defremove_punctuation(text):
returntext.translate(str.maketrans(PUNCT, ' ' * len(PUNCT)))

defpreprocessing_1(df):
df['nonum'] = df['Description'].str.replace('\d+', '')
df['nopun'] = df['nonum'].apply(lambdatext: remove_punctuation(text))
print('Preprocessing complete.')
```

Processing_1 comprises of 2 actions, replacing numbers by blank space and executing function remove punctuation(text)

A quick look at the data:

```
preprocessing_1(df)
ICD.display(df)
```

Preprocessing complete.

	Title	Description	nonum	nopun
0	Nobody; OK to take it and work on it	user-ag mozilla/5 window window nt en-u rv gec...	user-ag mozilla/ window window nt en-u rv geck...	user ag mozilla window window nt en u rv geck...
1	Nelson Bolyard (seldom reads bugmail)	creat attach detail server cert origin report ...	creat attach detail server cert origin report ...	creat attach detail server cert origin report ...
2	Nobody; OK to take it and work on it	user-ag mozilla/5 x11 linux i686 en-u rv gecko...	user-ag mozilla/ x linux i en-u rv gecko/ fire...	user ag mozilla x linux i en u rv gecko fire...
3	Nobody; OK to take it and work on it	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...
4	Nobody; OK to take it and work on it	drag bookmark bookmarks-menu folder folder los...	drag bookmark bookmarks-menu folder folder los...	drag bookmark bookmarks menu folder folder los...
...	...	...	...	...
10294	Dave Townsend (:Mossop)	extens instal pointer file directori point ext...	extens instal pointer file directori point ext...	extens instal pointer file directori point ext...
10295	Taras Glek (:taras)	creat attach detail diff review replac quot en...	creat attach detail diff review replac quot en...	creat attach detail diff review replac quot en...
10296	Zack Weinberg (:zwol)	creat attach detail testcas testcas crash trun...	creat attach detail testcas testcas crash trun...	creat attach detail testcas testcas crash trun...
10297	Aravind Gottipati [:aravind]	backup khan-vm khan bug info khan khan khan mo...	backup khan-vm khan bug info khan khan khan mo...	backup khan vm khan bug info khan khan khan mo...
10298	Nobody; OK to take it and work on it	user-ag mozilla/5 x11 linux i686 en-u rv b4pre...	user-ag mozilla/ x linux i en-u rv bpre gecko/...	user ag mozilla x linux i en u rv bpre gecko ...

10299 rows × 4 columns

--- Executed in 0.532 seconds ---

Figure 3: Adding no number and no punctuation columns

6.2. Remove long words, stop words, URLs, HTML Tags, long words and split, stem, lemmatize words:

This stage includes:

Preprocessing complete.

	Title	Description	nonum	nopun	short	split	stopw	stopf	stem	lemma	url	html	long	spell
0	Nobody; OK to take it and work on it	user-ag mozilla/5 window window nt en-u rv gec...	user-ag mozilla/ window window nt en-u rv geck...	user ag mozilla window window nt en u rv geck...	user ag mozilla window window nt en u rv gecko...	user ag mozilla window window nt en u rv gecko...	user ag mozilla window window nt en u rv gecko...	user ag nt en u rv gecko build identifi versio...	user ag nt en u rv gecko build identifi versio...	user ag nt en u rv gecko build identifi versio...	user ag nt en u rv gecko build identifi versio...	user ag nt en u rv gecko build identifi versio...	user gecko build identifi version add save sea...	user gecko build identify version add save sea...
1	Nelson Bolyard (seldom reads bugmail)	creat attach detail server cert origin report ...	creat attach detail server cert origin report ...	creat attach detail server cert origin report ...	creat attach detail server cert origin report ...	creat attach detail server cert origin report ...	creat attach detail server cert origin report ...	creat server cert origin report test attempt o...	creat server cert origin report test attempt o...	creat server cert origin report test attempt o...	creat server cert origin report test attempt o...	creat server cert origin report test attempt o...	creat server cert origin report test attempt o...	create server cert origin report test attempt ...
2	Nobody; OK to take it and work on it	user-ag mozilla/5 x11 linux i686 en-u rv gecko...	user-ag mozilla/ x linux i en-u rv gecko/ fire...	user ag mozilla x linux i en u rv gecko fire...	user ag mozilla x linux i en u rv gecko firefo...	user ag mozilla x linux i en u rv gecko firefo...	user ag mozilla x linux en u rv gecko firefox ...	user ag x linux en u rv gecko build identifi e...	user ag x linux en u rv gecko build identifi e...	user ag x linux en u rv gecko build identifi e...	user ag x linux en u rv gecko build identifi e...	user ag x linux en u rv gecko build identifi e...	user linux gecko build identifi easi lightn de...	user linux gecko build identify east light del...
3	Nobody; OK to take it and work on it	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notic problem site button search engin button ...	notice problem site button search engine butto...
4	Nobody; OK to take it and work on it	drag bookmark bookmarks-menu folder folder los...	drag bookmark bookmarks-menu folder folder los...	drag bookmark bookmarks menu folder folder los...	drag bookmark bookmarks menu folder folder los...	drag bookmark bookmarks menu folder folder los...	drag bookmark bookmarks menu folder folder los...	drag bookmark bookmarks menu folder folder los...	drag bookmark bookmark menu folder folder lose...	drag bookmark bookmark menu folder folder lose...	drag bookmark bookmark menu folder folder lose...	drag bookmark bookmark menu folder folder lose...	drag bookmark bookmark menu folder folder lose...	drag bookmark bookmark menu folder folder lose...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10294	Dave Townsend (:Mossop)	extens instal pointer file directori point ext...	extens instal pointer file directori point ext...	extens instal pointer file directori point ext...	extens instal pointer file directori point ext...	extens instal pointer file directori point ext...	extens instal pointer file directori point ext...	extens instal pointer file directori point ext...	exten instal pointer file directori point exte...	exten instal pointer file directori point exte...	exten instal pointer file directori point exte...	exten instal pointer file directori point exte...	exten instal pointer file directori point exte...	extent install pointer file director point ext...
10295	Taras Glek (:taras)	creat attach detail diff review replac quot en...	creat attach detail diff review replac quot en...	creat attach detail diff review replac quot en...	creat attach detail diff review replac quot en...	creat attach detail diff review replac quot en...	creat attach detail diff review replac quot en...	creat diff review replac endif endif patch int...	creat diff review replac endif endif patch int...	creat diff review replac endif endif patch int...	creat diff review replac endif endif patch int...	creat diff review replac endif endif patch int...	creat diff review replac endif endif patch int...	create diff review replace endif endif patch i...
10296	Zack Weinberg (:zwol)	creat attach detail testcas testcas crash trun...	creat attach detail testcas testcas crash trun...	creat attach detail testcas testcas crash trun...	creat attach detail testcas testcas crash trun...	creat attach detail testcas testcas crash trun...	creat attach detail testcas testcas crash trun...	creat testcas testcas crash trunk build load r...	creat testca testca crash trunk build load reg...	creat testca testca crash trunk build load reg...	creat testca testca crash trunk build load reg...	creat testca testca crash trunk build load reg...	creat testca testca crash trunk build load reg...	create testcap testcap crash trunk build load ...
10297	Aravind Gottipati [:aravind]	backup khan-vm khan bug info khan khan khan mo...	backup khan-vm khan bug info khan khan khan mo...	backup khan vm khan bug info khan khan khan mo...	backup khan vm khan bug info khan khan khan mo...	backup khan vm khan bug info khan khan khan mo...	backup khan vm khan bug info khan khan khan mo...	backup khan vm khan info khan khan khan monito...	backup khan vm khan info khan khan khan monito...	backup khan vm khan info khan khan khan monito...	backup khan vm khan info khan khan khan monito...	backup khan vm khan info khan khan khan monito...	backup khan khan info khan khan khan monitor f...	backup khan khan info khan khan khan monitor f...
10298	Nobody; OK to take it and work on it	user-ag mozilla/5 x11 linux i686 en-u rv b4pre...	user-ag mozilla/ x linux i en-u rv bpre gecko/...	user ag mozilla x linux i en u rv bpre gecko ...	user ag mozilla x linux i en u rv bpre gecko m...	user ag mozilla x linux i en u rv bpre gecko m...	user ag mozilla x linux en u rv bpre gecko min...	user ag x linux en u rv bpre gecko minefield b...	user ag x linux en u rv bpre gecko minefield b...	user ag x linux en u rv bpre gecko minefield b...	user ag x linux en u rv bpre gecko minefield b...	user ag x linux en u rv bpre gecko minefield b...	user linux bpre gecko minefield bpre build ide...	user linux pre gecko minefield pre build ident...

10299 rows × 14 columns

Figure 4: All of pre-processing stage

(1) short: Remove long words, words that are more than 86. Even if we split these long words into meaning full parts, it still has small impact on overall results.

Pseudocode:

short = Description(len(w)<85)

(2) split: Here we use sym_spell.word_segmentation to split long term into meaningful English words. Example: embedglobalhistori will return a list of 3 words 'embed','global', 'histori'

Pseudocode:

for words in short:

    if word is very long:

        split.appends(short.def_split())

(3) stopw: Remove stop words. Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence.

Pseudocode:

stopw = split.drop_stop_words()

(4) stopf: remove frequent words that appears in most of documents, with inverse term-frequency low and do not have any value for calculating Tfidf.

Pseudocode:

stopf = stopw.drop_frequent_words()

(5) stem: Stemming is used to produce root forms of words and the word produced.

Pseudocode:

stem = stopf.stem_words()

(6) lemma: Lemmatization helps us to achieve the root forms (sometimes called synonyms in search context) of inflected (derived) words.

Pseudocode:

lemma= stem.lemmtize_words()

(7) url: remove urls

Pseudocode:

url = lemma.remove_url()

(8) html: remove html

Pseudocode:

Html = url.remove_html()

(9) long: remove short words that is less than 2 letters

Pseudocode:

long = html(len(w)>2)

(10) spell: spelling check for the corpus. Import English spell dictionary and apply to long columns.

Pseudocode:

Spell = html.spell()

7. Creating dictionary and corpus:

The two main inputs to the LSA topic model are the dictionary(id2word) and the corpus.

```
ICD.display(dictionary)
```

```
['aaa',
 'aaaa',
 'aaaaaa',
 'aaaaaaaaaaab',
 'aaaaaaaaacw',
 'aaaaaaaacli',
 'aaaaaaadcb',
 'aaaabgdb',
 'aaaadf',
 'aaabab',
 'aaabc',
 'aaabceeedcffa',
 'aaacd',
 'aaafcacc',
 'aaaobntcbmj',
 'aaaobzcb',
 'aaargh',
 'aaatest',
 'aaawbxbtwc',
 'aaawlj',
 'aaawvytqhv',
 'aab',
 'aabcfa',
```

Figure 5: Dictionary

Gensim creates a unique id for each word in the document. The produced corpus shown above is a mapping of (word_id, word_frequency).

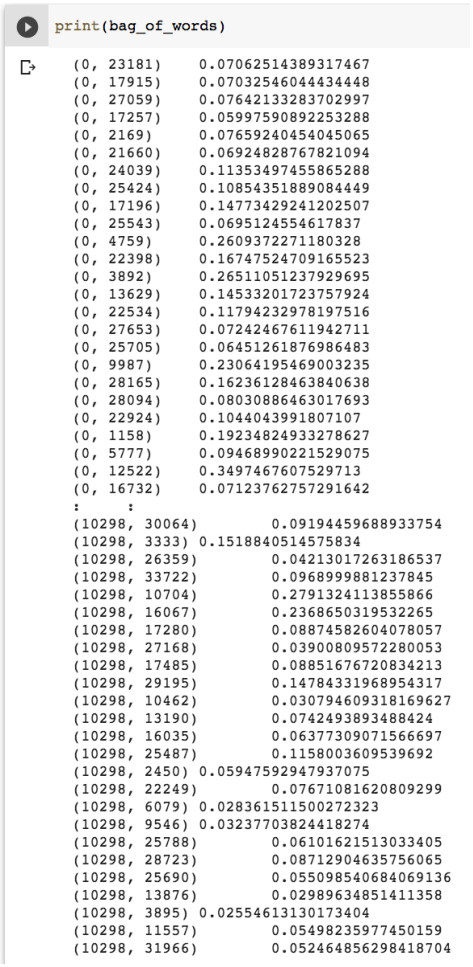


Figure 6: Bag of words

For example, (0, 23181) above implies, word id 0 occurs in the 23181, 17915 and so on. Likewise, word id 10298 occurs in 30064, 3333 and so on. This is used as the input by the LSA model.

8. Building the Topic Model:

A closer look at the line graph indicates that Coherence score decreases from 0.45 to approximately 0.3 when k is set from 5 to 25.

We have everything required to train the LSA model. In addition to the corpus and dictionary, you need to provide the number of topics as well.

Firstly, we need to quickly view the pattern of k, i.e the first 25 integers. Because one iteration for generating k is extremely slow, step is set to be equal 5.

```
num_topics = 5 -> coherence = 0.44820413609932874
num_topics = 10 -> coherence = 0.4159463964593108
num_topics = 15 -> coherence = 0.4110017195573915
num_topics = 20 -> coherence = 0.38803925680209983
num_topics = 25 -> coherence = 0.3471357766674331
```

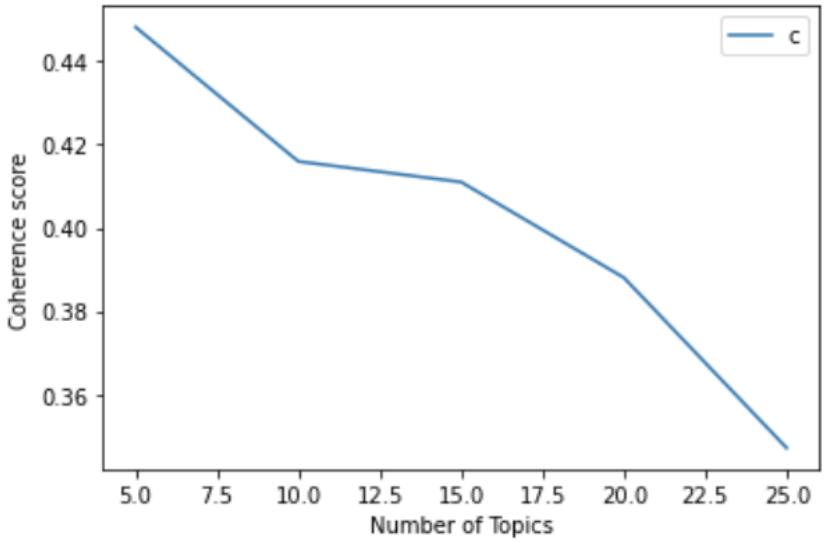
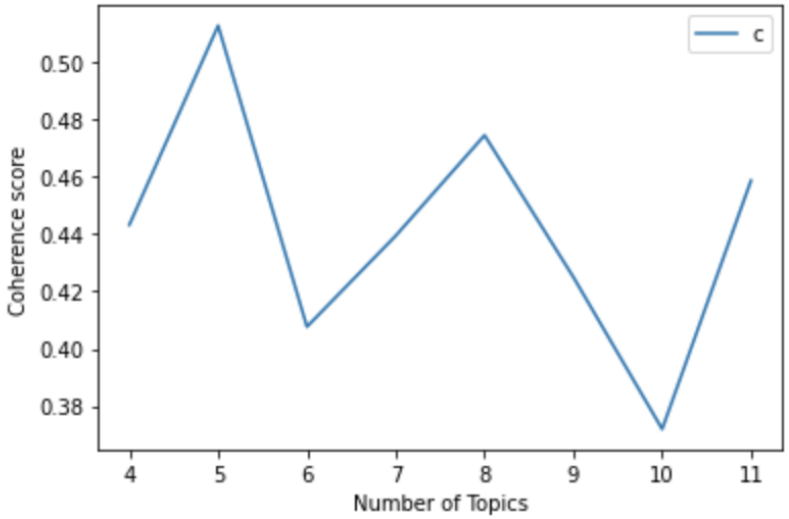


Figure 7: Testing k from 5 to 25

Specifically, with k is set from 4 to 11 with step equaling 1. This time, the algorithm produces the following:

```
num_topics = 4 -> coherence = 0.443258321082598
num_topics = 5 -> coherence = 0.5128640320931555
num_topics = 6 -> coherence = 0.4076678544807724
num_topics = 7 -> coherence = 0.4394689889947627
num_topics = 8 -> coherence = 0.474564790806965
num_topics = 9 -> coherence = 0.4250204047662205
num_topics = 10 -> coherence = 0.3719100679285356
num_topics = 11 -> coherence = 0.45873653144562515
```




```
Best num_topic: 5 with coherence =  0.5128640320931555
```

Figure 8: Testing k from 4 to 11

9. Viewing the Topics Model:

9.1 : Bag of Words:

Importing TfidfVectorizer, with min_df = 1 meaning the 100% proportion of text and remove stop words in English. Next, we shall need to transform it to vector

```
body = df['spell']

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=1, stop_words = 'english')
bag_of_words = vectorizer.fit_transform(body)

print(bag_of_words.todense())
```

Implementing "to dense" method yields the following matrix, with most of elements in the matrix is zero, others are 1:

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

9.2: TruncatedSVD:

The truncated refers to the fact that we are not going to get the output with as many vectors as input

```
from sklearn.decomposition import TruncatedSVD

svd = TruncatedSVD(n_components=5)
lsa = svd.fit_transform(bag_of_words)

topic_encoded_df = pd.DataFrame(lsa, columns = ['topic_1','topic_2','topic_3','topic_4','topic_5'])
topic_encoded_df['body'] = body
ICD.display(topic_encoded_df[['body','topic_1','topic_2','topic_3','topic_4','topic_5']])
```

This entire process has transformed our original data into topic-encoded data.

The data now consists of 5 columns, one representing each of the 5 topic that we requested from the TruncatedSVD.

	body	topic_1	topic_2	topic_3	topic_4	topic_5
0	user ag mozilla window window nt en u rv gecko...	0.326577	0.035839	0.080249	0.006019	-0.045868
1	create attach detail server cert origin report...	0.148042	-0.062792	-0.080877	-0.018870	-0.084154
2	user ag mozilla x linux i en u rv gecko firefo...	0.236455	0.018152	0.148012	-0.102574	-0.035013
3	notice problem site button search engine butto...	0.154942	0.015918	-0.064506	-0.067891	-0.045383
4	drag bookmark bookmarks menu folder folder los...	0.148197	-0.026185	0.048588	-0.171678	-0.032079
...	...	...	...	...	...	...
10294	extent install pointer file director point ext...	0.224373	-0.021225	-0.157330	-0.125154	-0.117055
10295	create attach detail diff review replace quot ...	0.171065	-0.170205	0.000457	-0.023840	-0.078916
10296	create attach detail testcap testcap crash tru...	0.195890	-0.092267	-0.167544	-0.078186	-0.032287
10297	backup khan vm khan bug info khan khan khan mo...	0.043830	-0.000806	-0.025162	0.002889	-0.016183
10298	user ag mozilla x linux i en u rv pre gecko mi...	0.317588	0.056027	-0.119278	-0.019918	0.278120

10299 rows × 6 columns

However, this may not yield much conclusion because of a thousands of rows include.

10. Byproducts

10.1 Topic Encoded:

The dictionary is an attribute of a fit count vectorizer model and can be assessed using

ICD.display(topic_encoded_df[['body','topic_1','topic_2','topic_3','topic_4','topic_5']])

	body	topic_1	topic_2	topic_3	topic_4	topic_5
0	user ag mozilla window window nt en u rv gecko...	0.326577	0.035830	0.080317	0.005397	-0.048502
1	create attach detail server cert origin report...	0.148042	-0.062788	-0.080918	-0.018505	-0.085631
2	user ag mozilla x linux i en u rv gecko firefo...	0.236455	0.018167	0.147792	-0.101698	-0.033398
3	notice problem site button search engine butto...	0.154942	0.015909	-0.064472	-0.068676	-0.045444
4	drag bookmark bookmarks menu folder folder los...	0.148197	-0.026167	0.048722	-0.172082	-0.029610
...	...	...	...	...	...	...
10294	extent install pointer file director point ext...	0.224373	-0.021252	-0.157237	-0.124811	-0.117955
10295	create attach detail diff review replace quot ...	0.171065	-0.170197	0.000424	-0.024136	-0.079384
10296	create attach detail testcap testcap crash tru...	0.195890	-0.092296	-0.167212	-0.082217	-0.039612
10297	backup khan vm khan bug info khan khan khan mo...	0.043830	-0.000813	-0.025144	0.003486	-0.015393
10298	user ag mozilla x linux i en u rv pre gecko mi...	0.317588	0.056055	-0.119412	-0.019152	0.279831

10299 rows × 6 columns

10.2 Encoding Matrix:

	topic_1	topic_2	topic_3	topic_4	topic_5	term
0	0.001176	-0.000675	-0.000498	0.000080	-0.000404	aa
1	0.000980	-0.000078	-0.000274	-0.000203	-0.000582	aaa
2	0.000240	-0.000115	0.000079	-0.000129	-0.000436	aaaa
3	0.000017	-0.000042	-0.000076	-0.000096	-0.000097	aaaaaa
4	0.000023	-0.000055	-0.000039	0.000015	-0.000024	aaaaaaaaaaab
...	...	...	...	...	...	...
51266	0.000005	-0.000012	-0.000016	-0.000015	-0.000007	zzwsdiyjrkwsfembdkfatjexllonqqxoxqriockwkmkspy...
51267	0.000032	0.000027	-0.000070	0.000074	-0.000180	zzxc
51268	0.000005	-0.000012	-0.000016	-0.000015	-0.000007	zzyrraufdskyo
51269	0.000133	-0.000262	0.000028	0.000309	0.000174	zzz
51270	0.000005	-0.000012	-0.000016	-0.000015	-0.000007	zzzfmdqdwkitg

51271 rows × 6 columns

The encoding is made up of the components stored as attribute of a fit truncated SVD This is a quick look of the encoding matrix. We could see that negative values are as important as the positive one.

11. View the topics in LSA model:

The above LSA model is built with 5 different topics where each topic is a combination of keywords and each keyword contributes a certain weightage to the topic.

We can see the keywords for each topic and the weightage(importance) of each keyword

```
encoding_matrix['abs_topic_1'] = np.abs(encoding_matrix['topic_1'])
encoding_matrix_1 = encoding_matrix.sort_values('abs_topic_1', ascending = False)
encoding_matrix_1 = encoding_matrix_1.head(10)
encoding_matrix_1.reset_index(drop = True, inplace = True)
ICD.display(encoding_matrix_1)
foriinrange(10):
print(encoding_matrix_1.at[i, 'term'], '*', end = ' ')
print("%.2f" % encoding_matrix_1.at[i, 'abs_topic_1'], end = ' ')
ifi<9: print('+', end = ' ')
```

	topic_1	topic_2	topic_3	topic_4	topic_5	term	abs_topic_1
0	0.422253	-0.399524	0.517081	0.261858	-0.034010	quot	0.422253
1	0.264096	-0.366114	-0.337277	0.367692	0.483651	gt	0.264096
2	0.221614	0.300586	0.003087	-0.034355	0.145756	window	0.221614
3	0.209601	0.043522	-0.025406	-0.158739	-0.069587	bug	0.209601
4	0.209412	0.268246	-0.004842	0.186732	-0.175388	firefox	0.209412
5	0.170596	0.131318	-0.079594	-0.011003	-0.076328	mozilla	0.170596
6	0.140814	0.124456	0.008105	-0.278024	0.315706	pre	0.140814
7	0.125364	0.171473	0.008317	-0.015542	0.060112	reproduce	0.125364
8	0.121854	0.142891	-0.001920	-0.043400	0.071777	en	0.121854
9	0.121436	0.153091	0.004084	-0.062637	0.099355	rv	0.121436

Topic 1:
quot * 0.42 + gt * 0.26 + window * 0.22 + bug * 0.21 + firefox * 0.21 + mozilla * 0.17 + pre * 0.14 + reproduce * 0.13 + en * 0.12 + rv * 0.12
It means the top 10 keywords that contribute to this topic are: ‘quot, ‘gt, ‘window.. and so on and the weight of ‘quot on topic 1 is 0.42.

The weights reflect how important a keyword is to that topic.

	topic_1	topic_2	topic_3	topic_4	topic_5	term	abs_topic_1	abs_topic_2
0	0.422253	-0.399543	0.517067	0.261806	-0.030320	quot	0.422253	0.399543
1	0.264096	-0.366133	-0.337320	0.366983	0.485208	gt	0.264096	0.366133
2	0.221614	0.300530	0.003038	-0.036264	0.152567	window	0.221614	0.300530
3	0.209412	0.268260	-0.004866	0.185533	-0.176862	firefox	0.209412	0.268260
4	0.125364	0.171483	0.008290	-0.015728	0.058637	reproduce	0.125364	0.171483
5	0.121078	0.156980	0.005640	-0.063303	0.097699	gecko	0.121078	0.156980
6	0.099615	0.155774	0.009940	-0.012534	0.065465	nt	0.099615	0.155774
7	0.121436	0.153092	0.004064	-0.063204	0.099802	rv	0.121436	0.153092
8	0.082889	-0.152674	-0.174942	-0.196092	-0.157660	review	0.082889	0.152674
9	0.076983	-0.145158	-0.169298	-0.192680	-0.147287	diff	0.076983	0.145158

Topic 2:
quot * 0.40 + gt * 0.37 + window * 0.30 + firefox * 0.27 + reproduce * 0.17 + gecko * 0.16 + nt * 0.16 + rv * 0.15 + review * 0.15 + diff * 0.15

	topic_1	topic_2	topic_3	topic_4	topic_5	term	abs_topic_1	abs_topic_2	abs_topic_3
0	0.422253	-0.399543	0.517067	0.261806	-0.030320	quot	0.422253	0.399543	0.517067
1	0.264096	-0.366133	-0.337320	0.366983	0.485208	gt	0.264096	0.366133	0.337320
2	0.103578	-0.035589	0.290344	-0.199595	-0.020379	bookmark	0.103578	0.035589	0.290344
3	0.062266	-0.072346	0.200355	-0.084800	-0.079059	filter	0.062266	0.072346	0.200355
4	0.089325	-0.047807	0.190165	-0.096678	-0.031749	select	0.089325	0.047807	0.190165
5	0.111119	-0.115556	-0.176683	-0.227047	-0.111556	attach	0.111119	0.115556	0.176683
6	0.082889	-0.152674	-0.174942	-0.196092	-0.157660	review	0.082889	0.152674	0.174942
7	0.076983	-0.145158	-0.169298	-0.192680	-0.147287	diff	0.076983	0.145158	0.169298
8	0.059129	-0.047770	0.159491	-0.071093	-0.068603	delete	0.059129	0.047770	0.159491
9	0.077612	-0.129490	-0.152858	-0.168917	-0.124212	patch	0.077612	0.129490	0.152858

Topic 3: quot * 0.52 + gt * 0.34 + bookmark * 0.29 + filter * 0.20 + select * 0.19 + attach * 0.18 + review * 0.17 + diff * 0.17 + delete * 0.16 + patch * 0.15

	topic_1	topic_2	topic_3	topic_4	topic_5	term	abs_topic_1	abs_topic_2	abs_topic_3	abs_topic_4
0	0.264096	-0.366133	-0.337320	0.366983	0.485208	gt	0.264096	0.366133	0.337320	0.366983
1	0.140814	0.124440	0.008142	-0.277384	0.317576	pre	0.140814	0.124440	0.008142	0.277384
2	0.422253	-0.399543	0.517067	0.261806	-0.030320	quot	0.422253	0.399543	0.517067	0.261806
3	0.111119	-0.115556	-0.176683	-0.227047	-0.111556	attach	0.111119	0.115556	0.176683	0.227047
4	0.075464	0.134211	-0.033029	0.202639	-0.197222	profil	0.075464	0.134211	0.033029	0.202639
5	0.103578	-0.035589	0.290344	-0.199595	-0.020379	bookmark	0.103578	0.035589	0.290344	0.199595
6	0.082889	-0.152674	-0.174942	-0.196092	-0.157660	review	0.082889	0.152674	0.174942	0.196092
7	0.076983	-0.145158	-0.169298	-0.192680	-0.147287	diff	0.076983	0.145158	0.169298	0.192680
8	0.209412	0.268260	-0.004866	0.185533	-0.176862	firefox	0.209412	0.268260	0.004866	0.185533
9	0.077612	-0.129490	-0.152858	-0.168917	-0.124212	patch	0.077612	0.129490	0.152858	0.168917

Topic 4: gt * 0.37 + pre * 0.28 + quot * 0.26 + attach * 0.23 + profil * 0.20 + bookmark * 0.20 + review * 0.20 + diff * 0.19 + firefox * 0.19 + patch * 0.17

	topic_1	topic_2	topic_3	topic_4	topic_5	term	abs_topic_1	abs_topic_2	abs_topic_3	abs_topic_4	abs_topic_5
0	0.264096	-0.366133	-0.337320	0.366983	0.485208	gt	0.264096	0.366133	0.337320	0.366983	0.485208
1	0.140814	0.124440	0.008142	-0.277384	0.317576	pre	0.140814	0.124440	0.008142	0.277384	0.317576
2	0.075464	0.134211	-0.033029	0.202639	-0.197222	profil	0.075464	0.134211	0.033029	0.202639	0.197222
3	0.209412	0.268260	-0.004866	0.185533	-0.176862	firefox	0.209412	0.268260	0.004866	0.185533	0.176862
4	0.070999	0.065309	0.012088	-0.137788	0.164090	minefield	0.070999	0.065309	0.012088	0.137788	0.164090
5	0.082889	-0.152674	-0.174942	-0.196092	-0.157660	review	0.082889	0.152674	0.174942	0.196092	0.157660
6	0.055687	0.080924	-0.040288	0.153571	-0.156541	support	0.055687	0.080924	0.040288	0.153571	0.156541
7	0.070384	0.056025	-0.060133	0.113784	-0.155418	update	0.070384	0.056025	0.060133	0.113784	0.155418
8	0.221614	0.300530	0.003038	-0.036264	0.152567	window	0.221614	0.300530	0.003038	0.036264	0.152567
9	0.073443	0.085819	0.007563	-0.044849	0.151507	tab	0.073443	0.085819	0.007563	0.044849	0.151507

Topic 5: gt * 0.49 + pre * 0.32 + profil * 0.20 + firefox * 0.18 + minefield * 0.16 + review * 0.16 + support * 0.16 + update * 0.16 + window * 0.15 + tab * 0.15

Summary:

Topic 1:

quot * 0.42 + gt * 0.26 + window * 0.22 + bug * 0.21 + firefox * 0.21 + mozilla * 0.17 + pre * 0.14 + reproduce * 0.13 + en * 0.12 + rv * 0.12

Topic 2:

quot * 0.40 + gt * 0.37 + window * 0.30 + firefox * 0.27 + reproduce * 0.17 + gecko * 0.16 + nt * 0.16 + rv * 0.15 + review * 0.15 + diff * 0.15

Topic 3: quot * 0.52 + gt * 0.34 + bookmark * 0.29 + filter * 0.20 + select * 0.19 + attach * 0.18 + review * 0.17 + diff * 0.17 + delete * 0.16 + patch * 0.15

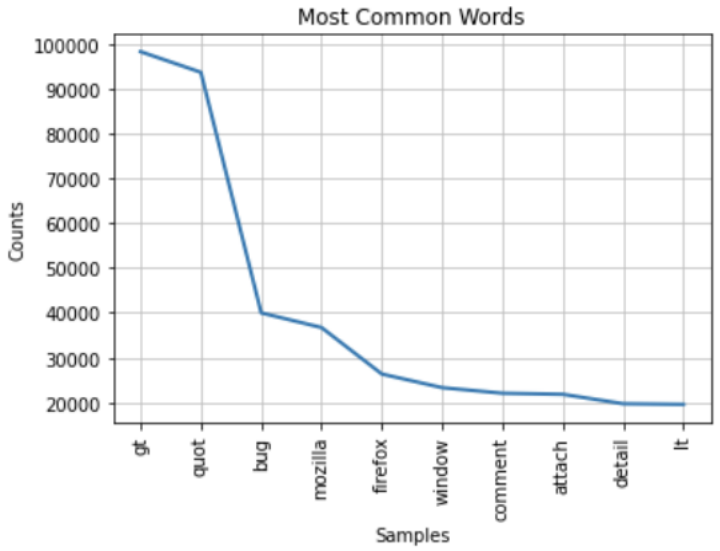
Topic 4: gt * 0.37 + pre * 0.28 + quot * 0.26 + attach * 0.23 + profil * 0.20 + bookmark * 0.20 + review * 0.20 + diff * 0.19 + firefox * 0.19 + patch * 0.17

Topic 5: gt * 0.49 + pre * 0.32 + profil * 0.20 + firefox * 0.18 + minefield * 0.16 + review * 0.16 + support * 0.16 + update * 0.16 + window * 0.15 + tab * 0.15

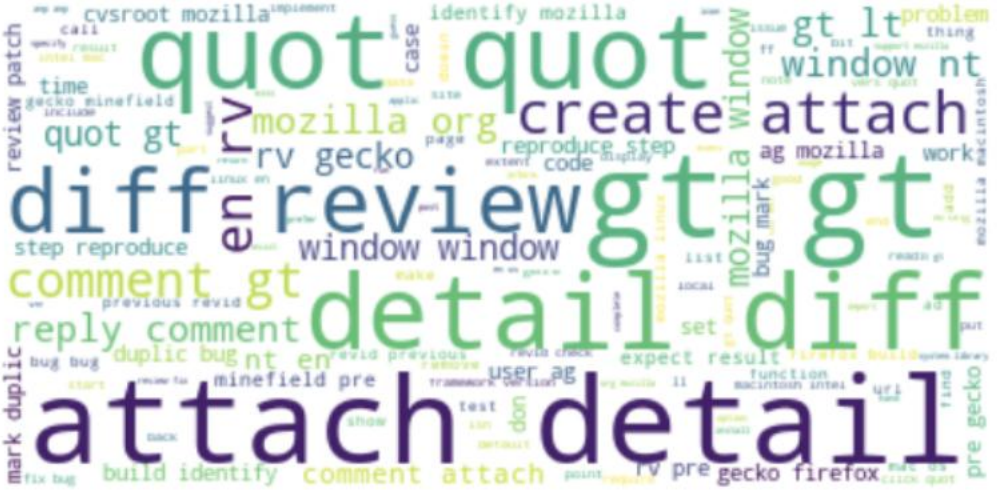
12. Some illustrations:

```
defword_frequency(dataframe):
# Word Frequency
    descriptions = " ".join(text for text in dataframe['spell'])
    words = word_tokenize(descriptions)
    unique_words = list(set(words))
print('Dictionary Length:', len(unique_words))
    fdist = FreqDist(words)
    fdist.plot(10, title = 'Most Common Words')
# Generating WordCloud
    word_cloud = WordCloud(background_color = 'white').generate(descriptions)
    plt.figure(figsize = (10, 10))
    plt.imshow(word_cloud, interpolation = 'bilinear')
    plt.axis('off')
    plt.title('WordCloud, Collocations Included')
    plt.show()
# Generating WordCloud
    word_cloud = WordCloud(background_color = 'white',
collocations = False).generate(descriptions)
    plt.figure(figsize = (10, 10))
    plt.imshow(word_cloud, interpolation = 'bilinear')
    plt.axis('off')
    plt.title('WordCloud, Collocations Excluded')
    plt.show()

defsort_unique_dataframe(dataframe, group, fsort):
    df_group = dataframe.groupby(group)
    uniques = df_group[fsort].unique()
    result = uniques.agg(np.size).sort_values(ascending = False)
return result

defplot_bar(dataframe, head, df_title):
    dataframe.head(head).plot(kind = 'bar', title = df_title)
    plt.show()

defbrief_summary(dataframe):
    start_time = time.time()
print('Title Count: ',
len(dataframe['Title'].unique()))
print('Description Count: ', len(dataframe['spell'].unique()))
    word_frequency(dataframe)
    sort_description = sort_unique_dataframe(dataframe, 'Title', 'spell')
    plot_bar(sort_description, 10, 'Titles by Description Count')
print('--- Executed in %.5s seconds ---' %(time.time() - start_time))
returnlist(sort_description.index)
```

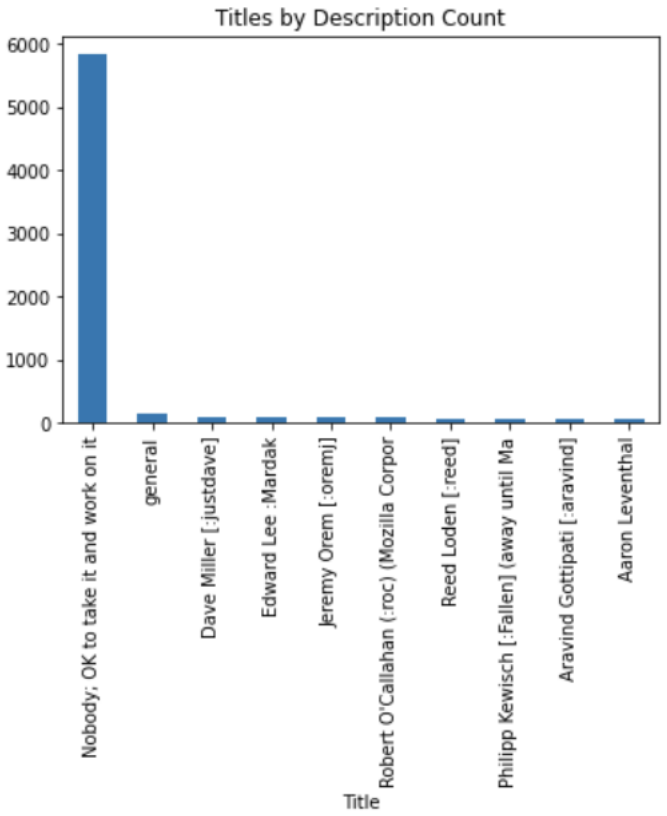
```
Title Count:  559
Description Count:  10286
Dictionary Length: 51551
```



WordCloud, Collocations Included



WordCloud, Collocations Excluded



13. Testing queries:

Coding:

```
start_time = time.time()
# Split Dataset to Training & Testing Subsets
df_fair = df[df.Title.isin(df_titles_agg[0:11])]
df_train, df_test = train_test_split(df_fair[['Title', 'spell']], test_size = 0.20)
df_train.reset_index(drop = True, inplace = True)
df_test.reset_index(drop = True, inplace = True)

# Building Classifier
tokenized_des = df_train['spell'].apply(lambdax: x.split())
detokenized_des = []
for i inrange(len(tokenized_des)):
    t = ' '.join(tokenized_des[i])
    detokenized_des.append(t)
vectorizer = TfidfVectorizer(stop_words = 'english', max_features = 1000,
smooth_idf = True)
X = vectorizer.fit_transform(detokenized_des)
svd_model = TruncatedSVD(n_components = best_nt, algorithm = 'randomized',
n_iter = 100, random_state = 122)
test_matrix = svd_model.fit_transform(X)

# Predicting
df_test['predict'] = ''
for i inrange(len(df_test)):
    sample = df_test.at[i, 'spell']
    sample_vector = vectorizer.transform([sample])
    sample_matrix = svd_model.transform(sample_vector)
    distance = cosine_similarity(sample_matrix, test_matrix).flatten()
    index = np.argwhere(distance == np.amax(distance))
    df_test.at[i, 'predict'] = df_train.at[index[0][0], 'Title']
```

```
# Display Accuracy, Precision & Error
print(confusion_matrix(df_test['Title'], df_test['predict']))
print(classification_report(df_test['Title'], df_test['predict']))
print('--- Executed in %.5s seconds ---' % (time.time() - start_time))
```

The simplest and most ordinarily used way is accuracy. This evaluation simply calculates the ratio between the quantity of correctly predicted points and the total number of points within the test data set.

However, to find out specifically how each class is assessed, which class is most correctly classified, and which class data is often misclassified into another. To be ready to evaluate these values, we use a matrix called the confusion matrix.Basically, the confusion matrix represents what number of data points belong to a category and are expected to fall under a class. For better understanding, see the table below:

CONFUSION MATRIX

```
[[   0    0    0    2    0    0    9    0    0    1    0]
 [   0    1    1    0    0    0    8    0    1    0    1]
 [   0    1    4    0    0    1   11    0    1    0    1]
 [   1    0    0    0    0    0   10    0    0    0    1]
 [   0    0    0    0    1    0    8    0    1    1    0]
 [   0    1    0    0    0    5   12    0    2    0    0]
 [  11    5    9   16   11   11 1050    7    7   11   31]
 [   0    0    0    0    0    0    9    1    0    1    0]
 [   1    0    2    0    0    0    8    1    1    0    0]
 [   0    0    0    0    1    1   10    2    0    0    0]
 [   0    0    0    0    0    0   24    0    0    0    1]]
```

Example results for the table below: there are 12 units within the class Aaron Leventhal, but none of them are properly classified and misclassified in Dave Townsend (:Mossop) 2 units , Nobody OK to take it and work on it 9 units and Robert O' Callahan (:roc) Mozilla Corpor 1 unit . Similar comments for the remainder of the classes

Therefore, we can infer that the Sum of the elements in this whole matrix is the number of points in the test set. The elements on the diagonal of the matrix are the number of correctly classified points of each data class.

	precision	recall	f1-score	support
Aaron Leventhal	0.00	0.00	0.00	12
Aravind Gottipati [:aravind]	0.12	0.08	0.10	12
Dave Miller [:justdave]	0.25	0.21	0.23	19
Dave Townsend (:Mossop)	0.00	0.00	0.00	12
Edward Lee :Mardak	0.08	0.09	0.08	11
Jeremy Orem [:oremj]	0.28	0.25	0.26	20
Nobody; OK to take it and work on it	0.91	0.90	0.90	1169
Philipp Kewisch [:Fallen] (away until Ma	0.09	0.09	0.09	11
Reed Loden [:reed]	0.08	0.08	0.08	13
Robert O'Callahan (:roc) (Mozilla Corpor	0.00	0.00	0.00	14
general	0.03	0.04	0.03	25
accuracy			0.81	1318
macro avg	0.17	0.16	0.16	1318
weighted avg	0.82	0.81	0.81	1318

With a way of defining a class as positive

- Precision is defined as the ratio of the number of true positives among those classified as positive (TP+ FP).

$$Precision = \frac{TP}{TP + FP}$$

- Recall is defined as the ratio of the number of true positives among those that are actually positives (TP + FN).

$$Recall = \frac{TP}{TP + FN}$$

- F1-score, is the harmonic mean of precision and recall (assuming that these two quantities are non-zero):

$$F_1 - score = 2\times\frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} = 2\times\frac{Precision\times Recall}{Precision + Recall}$$

F1-score has a value in the half range (0,1]. The higher the F_1, the better the classifier.

Intuitively, we can reduce the multiclass classification problem to the binary classification problem by looking at each class. For each class, we consider the data therein that class to possess a label as positive, all the remaining data have a label as negative. Then the Precision, Recall, and PR curve values are applied to each layer. For each class, we will get a pair of precision and recall values. For problems with few data classes, we can plot the PR curve for each class on an identical graph. However, for problems with many layers of data, this can be sometimes inconceivable. Instead, two Precision-Recall-based metrics are used, micro-average and macro-average.

$$Micro\_Average\_Precision = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} TP_c + FP_c}$$

$$Micro\_Average\_Recall = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} TP_c + FN_c}$$

where TPc, FPc, FNc are TP, FP, and FN of class c, respectively.

- Micro-average F-Score is calculated similarly to F-score but based on micro-average precision and micro-average recall.
- Macro-average precision is the average of precisions by class, like Macro-average recall.
- Macro-average F-Score is calculated similarly to F-score but based on macro-average precision and macro-average recall.
- $$Weighted\ average = \sum Recall \times Precision$$

Support is the actual outcome of the predictor variables.

APPENDIX

APPENDIX A:

Code for problem 1:
https://colab.research.google.com/drive/14qLcGHfREGtURXyjlNnHSJVeSBD7zpqt?usp=sharing

```
# Lam Hue Dung - MAMAIU18060
# Tran Thanh Dat - MAMAIU17036
# Tran Viet Hang - MAMAIU18079

# Import Libraries
import time
start_time = time.time()

import pandas aspd
pd.options.mode.chained_assignment = None
importnumpyas np
import warnings
warnings.filterwarnings("ignore")

importnltk
fromnltk.corpusimportstopwords
import string

frommlxtend.frequent_patternsimportapriorias ap1
frommlxtend.frequent_patternsimportassociation_rules
fromapyoriimportapriorias ap2

importmatplotlib.pyplotasplt
importnetworkxasnx
fromIPython.coreimport display as ICD

print("--- Executed in %.5s seconds ---" %(time.time() - start_time))

start_time = time.time()
# Import 'online_retail.xlsx' From URL
url = r'https://raw.githubusercontent.com/ThanhDatIU/File-CSV-Storage/main/Online%20Retail.xlsx'
df = pd.read_excel(url)
# Remove non-transactions (dumps/refunds)
df = df[df.UnitPrice>0]
df = df[df.Quantity>0]
# Remove transactions with NULL Description
df = df[df.Description.notnull()]
# Remove transactions with non-purchase stock codes
df = df[~df.StockCode.isin(["B", "C2", "D", "m", "M", "S", "DOT", "POST"])]
df = df[~df.StockCode.isin(["AMAZONFEE", "BANK CHARGES", "CRUK", "PADS"])]
df = df[~df.StockCode.apply(str).str.contains("gift")]
```

```
# Remove Punctuations
PUNCT = string.punctuation
defremove_punctuation(text):
returntext.translate(str.maketrans('', '', PUNCT))
col = df["Description"]
col = col.astype(str)
col = col.str.lower()
col = col.apply(lambdatext: remove_punctuation(text))
df["Description"] = col
# Re-enumerate indexes due to deleted transactions
df.reset_index(drop = True, inplace = True)
ICD.display(df)
print("--- Executed in %.5s seconds ---" %(time.time() - start_time))

"""**Part 1. Brief Summary**"""

defextra_fields(dataframe):
dataframe['TotalAmount'] = dataframe['Quantity'] * dataframe['UnitPrice']
dataframe['InvoiceYear'] = dataframe['InvoiceDate'].dt.year
dataframe['InvoiceMonth'] = dataframe['InvoiceDate'].dt.month

defsort_dataframe(dataframe, group, fsort):
df_group = dataframe.groupby(group)
    result = df_group[fsort].agg(np.sum).sort_values(ascending = False)
return result

defsort_unique_dataframe(dataframe, group, fsort):
df_group = dataframe.groupby(group)
uniques = df_group[fsort].unique()
    result = uniques.agg(np.size).sort_values(ascending = False)
return result

defplot_bar(dataframe, head, df_title):
dataframe.head(head).plot(kind = 'bar', title = df_title)
plt.show()

defgeneral_info(dataframe):
print('Transaction Count: ',
len(dataframe['InvoiceNo'].unique()))
print('Anonymous Transaction Count: ',
len(dataframe[dataframe['CustomerID'].isnull()]['InvoiceNo'
].unique()))
print('Customer Count: ', len(dataframe['CustomerID'].unique()) - 1)
print('Total Profit: ', round(sum(dataframe['TotalAmount']), 2))
top_customers = sort_dataframe(dataframe, 'CustomerID', 'TotalAmount')
plot_bar(top_customers, 10, 'Top Customers by Total Amount')
sort_quantity = sort_dataframe(dataframe, 'Description', 'Quantity')
plot_bar(sort_quantity, 10, 'Frequent Items by Quantity')
sort_amount = sort_dataframe(dataframe, 'Description', 'TotalAmount')
plot_bar(sort_amount, 10, 'Frequent Items by Total Amount')

defexplore_month(dataframe):
df_month = dataframe.sort_values('InvoiceDate').groupby(['InvoiceYear',
'InvoiceMonth'])
month_invoice = df_month['InvoiceNo'].unique().agg(np.size)
plot_bar(month_invoice, 12, 'Invoice Count by Month')
```

```
month_amount = df_month['TotalAmount'].agg(np.sum)
plot_bar(month_amount, 12, 'Total Amount by Month')

defexplore_country(dataframe):
sort_amount = sort_dataframe(dataframe, 'Country', 'TotalAmount')
plot_bar(sort_amount, 10, 'Countries by Total Amount')
sort_invoice = sort_unique_dataframe(dataframe, 'Country', 'InvoiceNo')
plot_bar(sort_invoice, 10, 'Countries by Invoice Count')
sort_customer = sort_unique_dataframe(dataframe, 'Country', 'CustomerID')
plot_bar(sort_customer, 10, 'Countries by Customer Count')

defbrief_summary(dataframe):
start_time = time.time()
df_brief = dataframe
extra_fields(df_brief)
general_info(df_brief)
explore_month(df_brief)
explore_country(df_brief)
print("--- Executed in %.5s seconds ---" %(time.time() - start_time))

brief_summary(df)

"""**Part 2. Applying Apriori**

**2.1/ Apriori from *mlxtend* library:**
"""

defap1_transform_data(df_original):
    invoice = ''
    transactions = []
for index, value inenumerate(df_original['InvoiceNo']):
ifinvoice != value:
            invoice = value
transactions.append([df_original['Description'][index]])
continue
        transactions[-1].append(df_original['Description'][index])
df_transform = pd.DataFrame(transactions)
df_transform.fillna(value = pd.np.nan, inplace = True)
returndf_transform

defunique_items(dataframe):
    squeeze = dataframe.values.ravel()
nan_items = pd.unique(squeeze)
    items = [x for x innan_itemsifstr(x) != 'nan']
return items

defone_hot_encoding(dataframe, items):
itemset = set(items)
envals = []
foridx, row indataframe.iterrows():
rowset = set(row)
          labels = {}
uncomms = list(itemset - rowset)
          commons = list(itemset.intersection(rowset))
forucinuncomms: labels[uc] = 0
for cm in commons: labels[cm] = 1
```

```
envals.append(labels)
returnenvals

defap1_plot(envals):
ohedt = pd.DataFrame(envals)
fitems = ap1(ohedt, min_support = 0.01, use_colnames = True)
fitems.sort_values('support', ascending = 0, inplace = True)
fitems.reset_index(drop = True, inplace = True)
ICD.display(fitems.head(10))
    rules = association_rules(fitems, metric = "confidence",
min_threshold = 0.02)
rules.sort_values('support', ascending = 0, inplace = True)
ICD.display(rules.head(10))
plt.scatter(rules['support'], rules['confidence'],
alpha = 0.5, marker = "*")
plt.xlabel('support')
plt.ylabel('confidence')
plt.title('Support vs Confidence - Apriori from \'mlxtend\'')
plt.show()
return rules

defap1_draw_graph(rules, rules_to_show):
    G1 = nx.DiGraph()
color_map = []
    N = 50
colors = np.random.rand(N)
strs=['R0', 'R1', 'R2', 'R3', 'R4', 'R5',
'R6', 'R7', 'R8', 'R9', 'R10', 'R11']
foriinrange(rules_to_show):
        G1.add_nodes_from(["R" + str(i)])
for a inrules.iloc[i]['antecedents']:
            G1.add_nodes_from([a])
            G1.add_edge(a, "R" + str(i), color = colors[i] , weight = 1)
for c inrules.iloc[i]['consequents']:
            G1.add_nodes_from([c])
            G1.add_edge("R" + str(i), c, color = colors[i],  weight = 1)
for node in G1:
found_a_string = False
for item instrs:
if node == item: found_a_string = True
iffound_a_string: color_map.append('yellow')
else: color_map.append('cyan')
edgelist = G1.edges()
edge_color = [G1[u][v]['color'] for u, v inedgelist]
    weights = [G1[u][v]['weight'] for u, v inedgelist]
pos = nx.spring_layout(G1, k = 16, scale = 1)
nx.draw(G1, pos, edgelist = edgelist, node_color = color_map,
edge_color = edge_color, width = weights, font_size = 10,
with_labels = True)
plt.show()

defapriori_1(dataframe):
start_time = time.time()
df_transform = ap1_transform_data(dataframe)
    items = unique_items(df_transform)
envals = one_hot_encoding(df_transform, items)
```

```
    rules = ap1_plot(envals)
    ap1_draw_graph(rules, 10)
print("--- Executed in %.5s seconds ---" % (time.time() - start_time))

apriori_1(df)

"""**2.2/ Apriori from *apyori* library:**"""

defap2_transform_data(df_original):
gp_invoiceno = df_original.groupby('InvoiceNo')
    transactions = []
for name, group ingp_invoiceno:
transactions.append(list(group['Description'].map(str)))
return transactions

defap2_plot(transactions):
    rules = ap2(transactions, min_support = 0.01,
min_confidence = 0.2, min_length = 2)
    results = list(rules)
    tabular = pd.DataFrame(np.random.randint(low = 0, high = 1,
size = (len(results), 6)),
columns = ['rules', 'antecedents',
'consequents', 'support',
'confidence', 'lift'])
    index = 0
for g, s, iin results:
tabular.iloc[index] = [' _&&_ '.join(list(g)),
' _&&_ '.join(list(i[0][0])),
' _&&_ '.join(list(i[0][1])),
                           s, i[0][2], i[0][3]]
        index = index + 1
tabular.sort_values('support', ascending = 0, inplace = True)
tabular.reset_index(drop = True, inplace = True)
ICD.display(tabular.head(10))
tabular.to_csv('rules.csv')
    rules = pd.read_csv('rules.csv', index_col = 0)
ICD.display(rules.head(10))
plt.scatter(tabular['support'], tabular['confidence'],
alpha = 0.5, marker = "*")
plt.xlabel('support')
plt.ylabel('confidence')
plt.title('Support vs Confidence - Apriori from \'apyori\'')
plt.show()
return tabular

defap2_draw_graph(rules, rules_to_show):
    G1 = nx.DiGraph()
color_map = []
    N = 50
colors = np.random.rand(N)
strs=['R0', 'R1', 'R2', 'R3', 'R4', 'R5',
'R6', 'R7', 'R8', 'R9', 'R10', 'R11']
foriinrange(rules_to_show):
        G1.add_nodes_from(["R" + str(i)])
        a, c = rules.at[i, 'antecedents'], rules.at[i, 'consequents']
        G1.add_nodes_from([a])
```

```
        G1.add_edge(a, "R" + str(i), color = colors[i] , weight = 1)
        G1.add_nodes_from([c])
        G1.add_edge("R" + str(i), c, color = colors[i],  weight = 1)
for node in G1:
found_a_string = False
for item instrs:
if node == item: found_a_string = True
iffound_a_string: color_map.append('yellow')
else: color_map.append('cyan')
edgelist = G1.edges()
edge_color = [G1[u][v]['color'] for u, v inedgelist]
    weights = [G1[u][v]['weight'] for u, v inedgelist]
pos = nx.spring_layout(G1, k = 16, scale = 1)
nx.draw(G1, pos, edgelist = edgelist, node_color = color_map,
edge_color = edge_color, width = weights, font_size = 10,
with_labels = True)
plt.show()

defapriori_2(dataframe):
start_time = time.time()
    transactions = ap2_transform_data(dataframe)
    tabular = ap2_plot(transactions)
    ap2_draw_graph(tabular, 10)
print("--- Executed in %.5s seconds ---" % (time.time() - start_time))

apriori_2(df)
```

APPENDIX B:

Code for problem 6:
https://colab.research.google.com/drive/1FxYCpsKL4bAmw5_0FvOBG2PriwXbEA-a?usp=sharing

```
# Lam Hue Dung - MAMAIU18060
# Tran Thanh Dat - MAMAIU17036
# Tran Viet Hang - MAMAIU18079

# Import Libraries
import time
start_time = time.time()

importnumpyas np
import pandas aspd
pd.options.mode.chained_assignment = None
importio
import requests
importpkg_resources
import warnings
warnings.filterwarnings('ignore')

importnltk
nltk.download('punkt')
fromnltkimportword_tokenize
fromnltk.probabilityimportFreqDist
nltk.download('stopwords')
fromnltk.corpusimportstopwords
fromnltk.stem.porterimportPorterStemmer
nltk.download('wordnet')
fromnltk.stemimportWordNetLemmatizer
fromnltk.tokenizeimportRegexpTokenizer

from collections import Counter
fromwordcloudimportWordCloud
import re
import string
from bs4 importBeautifulSoup
!pip install symspellpy
fromsymspellpy.symspellpyimportSymSpell
!pip install autocorrect
from autocorrect import Speller

fromgensimimport corpora
fromgensim.modelsimportLsiModel
fromgensim.models.coherencemodelimportCoherenceModel

fromsklearn.feature_extraction.textimportCountVectorizer, TfidfVectorizer
fromsklearn.decompositionimportTruncatedSVD
fromsklearn.model_selectionimporttrain_test_split
fromsklearn.metrics.pairwiseimportcosine_similarity
fromsklearn.metricsimportclassification_report, confusion_matrix

importmatplotlib.pyplotasplt
fromIPython.coreimport display as ICD
```

```
print('--- Executed in %.5s seconds ---' % (time.time() - start_time))

"""**1. Import - Preprocessing:**"""

start_time = time.time()
url = 'https://raw.githubusercontent.com/ThanhDatIU/File-CSV-Storage/main/group9.csv'
s = requests.get(url).content
df = pd.read_csv(io.StringIO(s.decode('utf-8')), header = None)
df = df[[4, 16]]
df.rename({4: 'Title', 16: 'Description'}, axis = 'columns', inplace = True)
ICD.display(df)
print('--- Executed in %.5s seconds ---' % (time.time() - start_time))

# Remove Punctuations
PUNCT = string.punctuation
defremove_punctuation(text):
returntext.translate(str.maketrans(PUNCT, ' ' * len(PUNCT)))

defpreprocessing_1(df):
df['nonum'] = df['Description'].str.replace('\d+', '')
df['nopun'] = df['nonum'].apply(lambdatext: remove_punctuation(text))
print('Preprocessing complete.')

start_time = time.time()
preprocessing_1(df)
ICD.display(df)
print('--- Executed in %.5s seconds ---' % (time.time() - start_time))

# Splitting Long Words
# Set max_dictionary_edit_distance to avoid spelling correction
sym_spell = SymSpell(max_dictionary_edit_distance = 0,
prefix_length = 7)
dictionary_path = pkg_resources.resource_filename(
'symspellpy', 'frequency_dictionary_en_82_765.txt')
# term_index is the column of the term
# count_index is the column of the term frequency
sym_spell.load_dictionary(dictionary_path, term_index = 0,
count_index = 1)
defsplit_word(df, input, output):
foriinrange(len(df.index)):
input_doc = df.at[i, input]
        result = []
for w ininput_doc.split():
iflen(w) >10:
clean_w = sym_spell.word_segmentation(w)
result.append(clean_w.corrected_string)
else: result.append(w)
        df.at[i, output] = ' '.join(result)

# Remove Stopwords
'', ''.join(stopwords.words('english'))
STOPW = set(stopwords.words('english'))
defremove_stopwords(text):
return' '.join([word for word instr(text).split() if word notin STOPW])

# Remove Frequent Words
```

```
defremove_freqwords(text, FREQWS):
return' '.join([word for word instr(text).split() if word notin FREQWS])

# Stemming
stemmer = PorterStemmer()
defstem_words(text):
return' '.join([stemmer.stem(word) for word intext.split()])

# Lemmatization
lemmatizer = WordNetLemmatizer()
deflemmatize_words(text):
return' '.join([lemmatizer.lemmatize(word) for word intext.split()])

# Remove URLs
defremove_urls(text):
url_pattern = re.compile(r'https?://\S+|www\.\S+')
returnurl_pattern.sub(r'', text)

# Remove HTML Tags
defremove_html(text):
returnBeautifulSoup(text, 'lxml').text

defpreprocessing_2(df):
df['short'] = df['nopun'].apply(lambdax: ' '.join([w for w inx.split() iflen(w) <85]))
df['split'] = ''
split_word(df, 'short', 'split')
df['stopw'] = df['split'].apply(lambdatext: remove_stopwords(text))
cnt = Counter()
for text indf['stopw'].values:
for word intext.split(): cnt[word] += 1
    FREQWS = set([w for (w, wc) incnt.most_common(10)])
df['stopf'] = df['stopw'].apply(lambdatext: remove_freqwords(text, FREQWS))
df['stem'] = df['stopf'].apply(lambdatext: stem_words(text))
df['lemma'] = df['stem'].apply(lambdatext: lemmatize_words(text))
df['url'] = df['lemma'].apply(lambdatext: remove_urls(text))
df['html'] = df['url'].apply(lambdatext: remove_html(text))
df['long'] = df['html'].apply(lambdax: ' '.join([w for w inx.split() iflen(w) >2]))
print('Preprocessing complete.')

start_time = time.time()
preprocessing_2(df)
ICD.display(df)
print('--- Executed in %.5s seconds ---' % (time.time() - start_time))

spell = Speller('en', fast = True)

defpreprocessing_3(df):
df['spell'] = df['long'].apply(lambdatext: spell(text))
print('Preprocessing complete.')

start_time = time.time()
preprocessing_3(df)
ICD.display(df)
print('--- Executed in %.5s seconds ---' % (time.time() - start_time))

"""**3. Gensim:**"""
```

```
start_time = time.time()
tokenizer = RegexpTokenizer(r'\w+')
des_list = []
forraw_desindf['spell']:
    token = tokenizer.tokenize(raw_des)
des_list.append(token)
dictionary = corpora.Dictionary(des_list)
ICD.display(dictionary)
doc_term_matrix = [dictionary.doc2bow(token) for token indes_list]
print('--- Executed in %.5s seconds ---' % (time.time() - start_time))

start_time = time.time()
coherence_values = []
model_list = []
coherence_best = 0
best_nt = 4
x = range(4, 9, 1)
fornum_topicsin x:
# generate LSI model
    model = LsiModel(doc_term_matrix, num_topics, id2word = dictionary)
model_list.append(model)
coherencemodel = CoherenceModel(model = model, texts = des_list,
dictionary = dictionary, coherence = 'c_v')
    result = coherencemodel.get_coherence()
print('num_topics =', num_topics, '-> coherence =', result)
if result >coherence_best: coherence_best, best_nt = result, num_topics
coherence_values.append(result)
plt.plot(x, coherence_values)
plt.xlabel('Number of Topics')
plt.ylabel('Coherence score')
plt.legend(('coherence_values'), loc = 'best')
plt.show()
print('Best num_topic:', best_nt, 'with coherence = ', coherence_best)
print('--- Executed in %.5s seconds ---' % (time.time() - start_time))

ICD.display(model_list)

body = df['spell']

fromsklearn.feature_extraction.textimportTfidfVectorizer
vectorizer = TfidfVectorizer(min_df=1, stop_words = 'english')
bag_of_words = vectorizer.fit_transform(body)

print(bag_of_words.todense())

fromsklearn.decompositionimportTruncatedSVD

svd = TruncatedSVD(n_components=5)
lsa = svd.fit_transform(bag_of_words)

topic_encoded_df = pd.DataFrame(lsa, columns = ['topic_1','topic_2','topic_3','topic_4','topic_5'])
topic_encoded_df['body'] = body
ICD.display(topic_encoded_df[['body','topic_1','topic_2','topic_3','topic_4','topic_5']])

dictionary = vectorizer.get_feature_names()
```

```
encoding_matrix = pd.DataFrame(svd.components_, index = ['topic_1','topic_2','topic_3','topic_4','topic_5']).T
encoding_matrix['term'] = ' '
foriinrange(len(dictionary)): encoding_matrix.at[i, 'term'] = dictionary[i]
ICD.display(encoding_matrix)

print(bag_of_words)

ICD.display(topic_encoded_df)

encoding_matrix['abs_topic_1'] = np.abs(encoding_matrix['topic_1'])
encoding_matrix_1 = encoding_matrix.sort_values('abs_topic_1', ascending = False)
encoding_matrix_1 = encoding_matrix_1.head(10)
encoding_matrix_1.reset_index(drop = True, inplace = True)
ICD.display(encoding_matrix_1)
foriinrange(10):
print(encoding_matrix_1.at[i, 'term'], '*', end = ' ')
print("%.2f" % encoding_matrix_1.at[i, 'abs_topic_1'], end = ' ')
ifi<9: print('+', end = ' ')

encoding_matrix['abs_topic_2'] = np.abs(encoding_matrix['topic_2'])
encoding_matrix_2 = encoding_matrix.sort_values('abs_topic_2', ascending = False)
encoding_matrix_2 = encoding_matrix_2.head(10)
encoding_matrix_2.reset_index(drop = True, inplace = True)
ICD.display(encoding_matrix_2)
foriinrange(10):
print(encoding_matrix_2.at[i, 'term'], '*', end = ' ')
print("%.2f" % encoding_matrix_2.at[i, 'abs_topic_2'], end = ' ')
ifi<9: print('+', end = ' ')

encoding_matrix['abs_topic_3'] = np.abs(encoding_matrix['topic_3'])
encoding_matrix_3 = encoding_matrix.sort_values('abs_topic_3', ascending = False)
encoding_matrix_3 = encoding_matrix_3.head(10)
encoding_matrix_3.reset_index(drop = True, inplace = True)
ICD.display(encoding_matrix_3)
foriinrange(10):
print(encoding_matrix_3.at[i, 'term'], '*', end = ' ')
print("%.2f" % encoding_matrix_3.at[i, 'abs_topic_3'], end = ' ')
ifi<9: print('+', end = ' ')

encoding_matrix['abs_topic_4'] = np.abs(encoding_matrix['topic_4'])
encoding_matrix_4 = encoding_matrix.sort_values('abs_topic_4', ascending = False)
encoding_matrix_4 = encoding_matrix_4.head(10)
encoding_matrix_4.reset_index(drop = True, inplace = True)
ICD.display(encoding_matrix_4)
foriinrange(10):
print(encoding_matrix_4.at[i, 'term'], '*', end = ' ')
print("%.2f" % encoding_matrix_4.at[i, 'abs_topic_4'], end = ' ')
ifi<9: print('+', end = ' ')

encoding_matrix['abs_topic_5'] = np.abs(encoding_matrix['topic_5'])
encoding_matrix_5 = encoding_matrix.sort_values('abs_topic_5', ascending = False)
encoding_matrix_5 = encoding_matrix_5.head(10)
encoding_matrix_5.reset_index(drop = True, inplace = True)
ICD.display(encoding_matrix_5)
foriinrange(10):
```

```
print(encoding_matrix_5.at[i, 'term'], '*', end = ' ')
print("%.2f" % encoding_matrix_5.at[i, 'abs_topic_5'], end = ' ')
ifi<9: print('+', end = ' ')

defword_frequency(dataframe):
# Word Frequency
    descriptions = " ".join(text for text indataframe['spell'])
    words = word_tokenize(descriptions)
unique_words = list(set(words))
print('Dictionary Length:', len(unique_words))
fdist = FreqDist(words)
fdist.plot(10, title = 'Most Common Words')
# Generating WordCloud
word_cloud = WordCloud(background_color = 'white').generate(descriptions)
plt.figure(figsize = (10, 10))
plt.imshow(word_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.title('WordCloud, Collocations Included')
plt.show()
# Generating WordCloud
word_cloud = WordCloud(background_color = 'white',
collocations = False).generate(descriptions)
plt.figure(figsize = (10, 10))
plt.imshow(word_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.title('WordCloud, Collocations Excluded')
plt.show()

defsort_unique_dataframe(dataframe, group, fsort):
df_group = dataframe.groupby(group)
uniques = df_group[fsort].unique()
    result = uniques.agg(np.size).sort_values(ascending = False)
return result

defplot_bar(dataframe, head, df_title):
dataframe.head(head).plot(kind = 'bar', title = df_title)
plt.show()

defbrief_summary(dataframe):
start_time = time.time()
print('Title Count: ',
len(dataframe['Title'].unique()))
print('Description Count: ', len(dataframe['spell'].unique()))
word_frequency(dataframe)
sort_description = sort_unique_dataframe(dataframe, 'Title', 'spell')
plot_bar(sort_description, 10, 'Titles by Description Count')
print('--- Executed in %.5s seconds ---' %(time.time() - start_time))
returnlist(sort_description.index)

df_titles_agg = brief_summary(df)

start_time = time.time()
# Split Dataset to Training & Testing Subsets
df_fair = df[df.Title.isin(df_titles_agg[0:11])]
df_train, df_test = train_test_split(df_fair[['Title', 'spell']], test_size = 0.20)
df_train.reset_index(drop = True, inplace = True)
```

```
df_test.reset_index(drop = True, inplace = True)

# Building Classifier
tokenized_des = df_train['spell'].apply(lambdax: x.split())
detokenized_des = []
foriinrange(len(tokenized_des)):
    t = ' '.join(tokenized_des[i])
detokenized_des.append(t)
vectorizer = TfidfVectorizer(stop_words = 'english', max_features = 1000,
smooth_idf = True)
X = vectorizer.fit_transform(detokenized_des)
svd_model = TruncatedSVD(n_components = best_nt, algorithm = 'randomized',
n_iter = 100, random_state = 122)
test_matrix = svd_model.fit_transform(X)

# Predicting
df_test['predict'] = ''
foriinrange(len(df_test)):
    sample = df_test.at[i, 'spell']
sample_vector = vectorizer.transform([sample])
sample_matrix = svd_model.transform(sample_vector)
    distance = cosine_similarity(sample_matrix, test_matrix).flatten()
    index = np.argwhere(distance == np.amax(distance))
df_test.at[i, 'predict'] = df_train.at[index[0][0], 'Title']

# Display Accuracy, Precision & Error
print(confusion_matrix(df_test['Title'], df_test['predict']))
print(classification_report(df_test['Title'], df_test['predict']))
print('--- Executed in %.5s seconds ---' % (time.time() - start_time))
```

REFERENCES

Kaggle.com. 2021. *Market Basket Analysis - Apriori Algorithm*. [online] Available at: <https://www.kaggle.com/oliviapointon/market-basket-analysis-apriori-algorithm?fbclid=IwAR2YfXXIRNNj-BlhHqOZzwhP-0t99ZjpNMHERAJRabbmn8daZhdbdVRi3y0> [Accessed 12 June 2021].

GitHub. 2021. *dinajankovic/Market-Basket-Analysis-Online-Retail*. [online] Available at: <https://github.com/dinajankovic/Market-Basket-Analysis-Online-Retail/blob/master/Market_Basket_Analysis_Online_Retail.R> [Accessed 12 June 2021].

Moffitt, C., 2021. *Introduction to Market Basket Analysis in Python - Practical Business Python*. [online] Pbpython.com. Available at: <https://pbpython.com/market-basket-analysis.html> [Accessed 12 June 2021].

Machinelearningplus.com. 2021. [online] Available at: <https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/> [Accessed 12 June 2021].

2021. [online] Available at: <https://www.datacamp.com/community/tutorials/discovering-hidden-topics-python> [Accessed 12 June 2021].

Python), T. and Joshi, P., 2021. *Topic Modelling In Python Using Latent Semantic Analysis*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2018/10/stepwise-guide-topic-modeling-latent-semantic-analysis/> [Accessed 12 June 2021].