
POLITECNICO DI TORINO

DIMEAS – DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING

MASTER OF SCIENCE DEGREE
IN
AUTOMOTIVE ENGINEERING

Master's degree thesis

**Text mining on product reviews to automatic
retrieve product strengths and defects**



Supervisors:

Prof. Giulia Bruno

Prof. Franco Lombardi

Candidate:

Bhaves Savaj

Matricola: 247042

October 2020

A Thesis Has Been Submitted In Partial Fulfilment Of The
Requirements For The Degree Of Master In Automotive
Engineering At Politecnico Di Torino.
2020

Acknowledgments

I would first like to express my sincere appreciation to Prof. Giulia Bruno for giving me the opportunity to participate in this interesting project. I would like also to thank for her continuous and her useful suggestions throughout the whole research period.

I would also like to thank my beloved family and friends who were always supporting me and staying by my side throughout my academic years.

Finally, I would like to take the chance to thank Politecnico Di Torino University with all its staff and professors whom had a hand in obtaining my master's degree in automotive engineering.

Bhavesh Savaj

Abstract

The aim of this thesis is to apply the text mining and clustering algorithms on the product reviews to automatic extract the strengths and defects of the product on the basis of customer experiences.

Analyze the reviews of the product that help to improve the product quality and services. Reviews from online shopping sites such as eBay, Amazon, etc. that not only helps a consumer to buy product but also can help a manufacturer or seller to know the advantage and disadvantage of their product. Only Amazon star ratings are not sufficient for this. By analysing through the text reviews to know specifically which feature of the product or service is lacking customer satisfaction.

Actually, product has many numbers of reviews and it's very difficult for a person to go through all the reviews. So, we have to make a system which can give output that defined the product strength and defects. The dataset which includes product details and customer reviews for product is collected from Amazon.com. The implementation of this system is achieved by using Anaconda Jupyter notebook and Google colab. The amazon reviews undergo Natural Language Processing and text mining in order to define text pre-processing and TF-IDF vectorizer. Then a clustering analysis is made to identify the strengths and defects of the product.

Keywords: text mining, text analysis, clustering algorithms, machine learning, data mining, and product reviews

Table of Contents

List of Figures	6
1. Introduction	9
2. Related work	
2.1. Text mining	11
2.2. Clustering analysis	13
2.3. Literature reviews of related work	15
3. Case study	
3.1. Data source	21
3.2. Data collection	22
3.3. CSV format	24
4. Method	
4.1. Overview	25
4.2. Data preparation	27
4.2.1. For positive reviews	29
4.2.2. For negative reviews	32
4.3. Text preparation	35
4.3.1. Contractions	35
4.3.2. Convert all characters to lowercase	36
4.3.3. Remove punctuations	37
4.3.4. Remove stopwords	37
4.3.5. Lemmatization	38
• Wordnet lemmatizer with nltk	39
• Wordnet lemmatizer with Pos tag	40
4.3.6. Positive reviews after clean the texts	41
4.3.7. Negative reviews after clean the texts	43
4.3.8. WordCloud	47
• WordCloud for positive reviews	48
• WordCloud for negative reviews	49
4.4. TF-IDF	50
4.4.1. Introduction	50
4.4.2. Mathematical means (Manual example)	51

4.4.3. Calculate TF-IDF by python	53
• TF-IDF on positive reviews	54
• TF-IDF on negative reviews	55
5. Clustering	
5.1. Introduction	57
5.2. Clustering algorithms	60
5.2.1. Agglomerative clustering	60
5.2.2. K-Means clustering	67
5.2.3. DBSCAN clustering	72
6. Input for clustering analysis and their final outcomes	76
6.1. K-means clustering algorithms	77
6.1.1. K-means clustering algorithms for positive reviews	77
6.1.2. K-means clustering algorithms for negative reviews	84
6.2. Hierarchical clustering algorithms	88
6.2.1. Agglomerative clustering for positive reviews	88
6.2.2. Agglomerative clustering for negative reviews	94
6.3. DBSCAN clustering algorithms	98
6.3.1. DBSCAN clustering for positive reviews	98
6.3.2. DBSCAN clustering for negative reviews	102
7. Final analysis of results	104
• Product strengths	105
• Product defects	105
8. Conclusion	106
References	

List of Figures

1	<i>Text mining WordCloud</i>	12
2	<i>Unclustered vs. clustered data</i>	13
3	<i>Working flow of text mining</i>	15
4	<i>sample of document term matrix</i>	16
5	<i>Structured and unstructured data sources across the PLC</i>	18
6	<i>Integrated text mining architecture</i>	20
7	<i>Bosch windshield wiper blade</i>	21
8	<i>AMZshark data collection tool</i>	22
9	<i>dataset format</i>	23
10	<i>Methods for text mining</i>	25
11	<i>WordCloud for positive reviews</i>	48
12	<i>WordCloud for negative reviews</i>	49
13	<i>Output of example results of tfidf analysis</i>	53
14	<i>Output of count vectorizer for positive reviews</i>	54
15	<i>Output of Tfidf vecorizer for positive reviews</i>	55
16	<i>Output of count vecorizer for negative reviews</i>	56
17	<i>Output of Tfidf vectorizer for negative reviews</i>	56
18	<i>Types of clustering methods</i>	58
19	<i>Concept of Agglomerative vs. Divisive clustering</i>	60
20	<i>Sample of Dendrogram</i>	61
21	<i>Single linkage</i>	62
22	<i>Complete linkage</i>	62
23	<i>Average linkage</i>	63
24	<i>Ward linkage</i>	63
25	<i>example results of dendrogram</i>	65
26	<i>dendrogram final results of example</i>	65
27	<i>Sample for elbow method</i>	69
28	<i>sample for silhouette score</i>	70
29	<i>Sample of Dbscan clustering</i>	72
30	<i>Parameters for Dbscan clustering</i>	73
31	<i>Output of elbow method of K-means for positive reviews</i>	78
32	<i>Output of silhouette score of K-means for positive reviews</i>	79
33	<i>Output of scatter plot of K-means for positive reviews</i>	80
34	<i>Output of elbow method of K-means for negative reviews</i>	84
35	<i>Output of silhouette score of K-means for negative reviews</i>	85
36	<i>Scatter plot of K-means for negative reviews</i>	85
37	<i>Output of dendrogram of agglo. For positive reviews</i>	89
38	<i>Output of silhouette score of agglo. For positive reviews</i>	90
39	<i>Output of scatter plot of agglo. For positive reviews</i>	91
40	<i>Output of dendrogram of agglo. For negative reviews</i>	94

41	<i>Output of silhouette score of agglo. For negative reviews</i>	95
42	<i>Output of scatter plot of agglo. For negative reviews</i>	95
43	<i>Output of Dbscan for positive reviews</i>	100
44	Output of Dbscan for negative clustering	102

1. Introduction

Text mining also called as text analysis. Text mining is the process of transforming unstructured text into valuable and actionable information. Text analysis uses different artificial intelligent technologies to automatically process data and generate meaningful insights, that enabling to company to make data driven decisions.

Every day, the large amount of data is generated that represent both an opportunities and a challenge. On the other side, data helps to companies to get smart insights on people opinions about a products or services. All the ideas could get from analyzing emails, social media, product reviews, support tickets, customer feedback, etc.

For example, when a person thinks of buying a product, his or her next immediate action would be to search for the product on the internet. Internet gives him/her a lot of choices based on brand, price, model, colours, quality, features, discounts, rating and many more. Introduction of new products, new fashion, new model, new business, new technology, new brand, new marketing strategies, new services happen daily. These may confuse the customers when having to make a choice. When it's tough to make a choice, we tend to get the feedback from the people who have already bought and used those products. Customers write their review on online shopping sites.

Amazon provides overall rating for each product based on all ratings from the reviewers of the product. But from that overall rating a person cannot necessarily conclude on quality of all features of the product. For example, considering the scenario where I want to buy car side mirror. When searching for one in Amazon.com, I came across a few products within my budget with ratings ranging from 4 to 5.

When I went through the text reviews I understood that most of the customers were not happy with proper fitting but quality of the product was excellent.

In another instance, I noticed that most of customers wrote lesser rating because of bad packaging for a product. In such cases the ratings are not reliable to decide on the quality of the product. In another scenario, the manufacturer or seller of the product may want to improve the quality of product.

In order to have the solution for these problems is to read through the text reviews to know specifically which feature of the product is not sufficient for customer satisfaction. But product may have thousands of reviews which would make this task very difficult. So, we need a system to do that for us. The text mining through the reviews texts for each products and get the results as pros and cons of the product that customers like and dislike.

For implementation of such system I have used Amazon reviews and products data as the dataset. As Amazon has a wide range of products, I chose “Bosch windshield wiper blade” category for illustration in this report. The Bosch wiper blade category dataset consist 899 reviews. The natural language processing and text mining techniques are used to identify major features of the product. Then clustering analysis is performed to identify the strengths and defects of reviews.

2. Related work

There are several possible ways to collect feedback from users about products they have purchased and used for a time. So, there is one way to do is to ask users to do surveys. Or another way is to do experimental research and observational study of users that interacting with devices. In this work, text analysis was completed by using on Amazon review data. Actually, considering the amazon reviews, are good source of data for getting consumer perceptions because of the large number of data points. And customers are also able to post their reviews after used the product and know its advantage and disadvantage, and while they are in no pressure, feelings and sharing their thoughts at home being in a timed lab setting.

In the following brief review of state of art is performed based on the topic related to this thesis. In this thesis, there are considered two main areas a) text mining and b) clustering analysis.

2.1. Text mining:

Basically, text analysis is an automatic process that uses natural language processing to extract meaningful insights from unstructured text. To transform data into information so that machine can understand, text mining is automatic process of classifying texts by sentiment, intent, topic and so on. Text mining also called data mining that is very similar to text analysis; text analysis is the process of deriving high quality of information from texts.

Structured data is mostly categorized as quantitative data. The dataset that fits within fixed fields and columns in relational databases and spreadsheets. Structured data include such as names, dates, credit card numbers, stock information, addresses, geo-location and so on.



Unstructured data is mostly categorized as quantitative data and unstructured data cannot be processed and analyzed by using conventional methods and tools. Texts, mobile activities, video, audio, social-media, surveillance imagery, etc. are considered as unstructured data. Unstructured data is very difficult to de-construct because it has no predefined model, so it cannot be organized in relational databases. For instance, text mining techniques applied on unstructured data that can help to companies to learn buying habits and pattern in purchases, sentiment on specific product and so more.

Information extraction, data mining and a KDD – knowledge discovery in databases process are three different perspectives of text mining. Text data mining basically involves the process of structuring of input text with the addition of some derived linguistic features and the remove of others, and subsequent insert into a database, derive the patterns within the structured data and finally evaluation and interpretation of the output results. Usually text mining tasks include text clustering, text categorization, entity extraction, sentiment analysis, document summarization.

Utilization the right data will allow the companies to:

- Reduce the operational costs
- Track current metrics and create new ones
- Understand its customers on a far deeper level
- Unveil smarter and more targeted marketing campaigns
- Find new product opportunities and offerings.

2.2. Clustering analysis:

Clustering analysis is basically a type of unsupervised learning method. In unsupervised learning method we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful insight structure, generative features, explanatory process, and groupings in a set of example.

Clustering analysis is the task of dividing the population and data into the number of groups such as that data in the same groups are more similar to other data in the same group and non similar to the data in other groups. Clustering analysis is basically a collection of objects on the basis of similarities and dissimilarities between them.

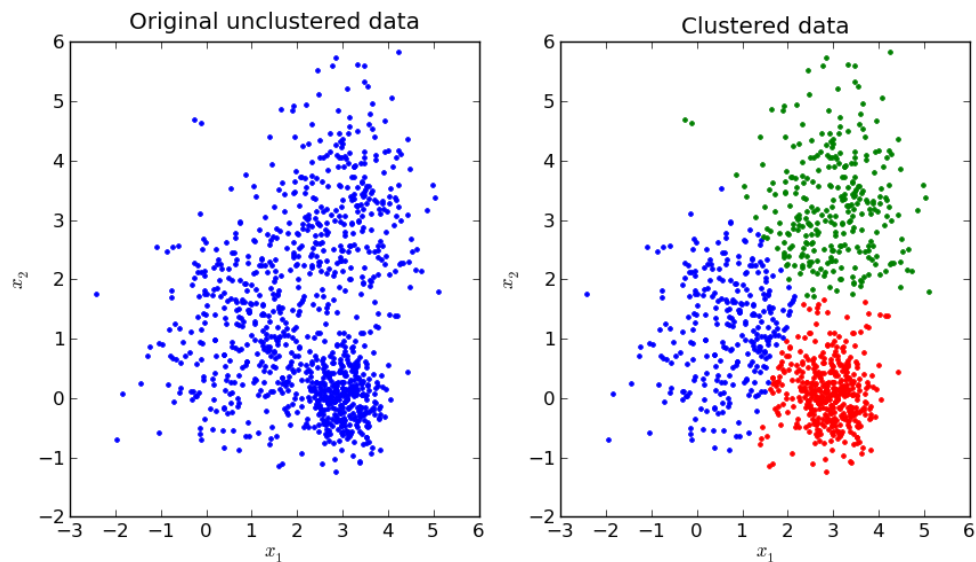


Figure 2: Unclustered data vs. cluster data

In theory, data points that are in the same group should have similar properties and/or features, while data points in different groups should have highly dissimilar properties and/or features. Clustering is a method of unsupervised learning and is a common technique for statistical data analysis used in many fields. There are many popular clustering algorithms we will discuss in following chapter 4.

- **Unsupervised Learning** allows users to perform more complex processing tasks that can compare to supervised learning. However, unsupervised learning method can be more un-predictable compared with other natural learning methods. Unsupervised learning algorithm includes clustering, anomaly detections, networks, etc. Unsupervised learning methods help to find features which can be useful for categorization.

In unsupervised learning method, only has input data and no corresponding output. The main aim of unsupervised learning method is to model of distribution or the underlying structure in the data in order to learn more about the data. Unsupervised learning can be grouped into clustering and their association problems.

- **Clustering problem:** In clustering problem, when try to find the inherent groupings in the data, for instance, grouping customers by their purchasing behaviour.
- **Association problems:** in an association learning problem where try to find the rules that can be described by huge portions of our data. For instance, some one that buy product x also tend to buy product y.

Unsupervised machine learning is the training of the machine using information that is neither classified nor labelled and allow the algorithms that act on the information without guidance. Machine task is to group unsorted information according to patterns, similarities, and differences without any prior of training of data.

2.3. Literature reviews of related work:

The authors in [1] presented using text mining to handle unstructured data in semiconductor manufacturing.

In the field of semiconductor manufacturing, people usually focus on the data of the designed database, such as the values from tool sensors, inline metrology data. These dataset are well structured and easily handled by engineers for further analysis. However, there are still many other unstructured data in semiconductor manufacturing for instance, hold records of lots or consigning records. Traditionally, these unstructured data are only for reference. People only check them while some unexpected accidents happen and they have to go back to confirm the causes. In the other words, most of these data are not reused again.

To utilize these data more efficiently and collocate with other structured data already in the foundry, so referred to experience of text mining to build up the methodology of handling unstructured data in semiconductor manufacturing.

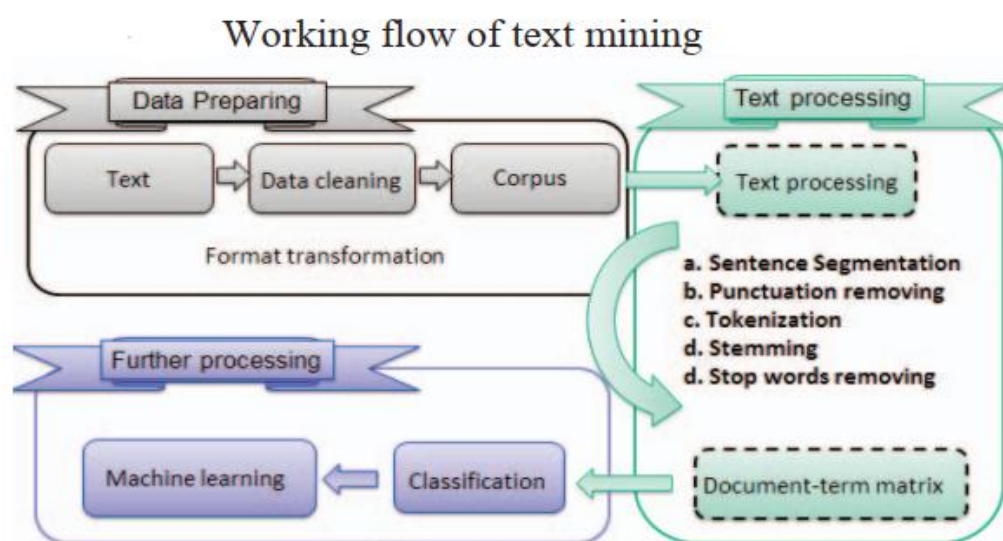


Figure 3: Working flow of text mining

There are three major loops:

- Data preparing
- Text processing
- Further processing

As first, to do the process of data cleaning to transfer original text to corpus and then apply a series of text processing for instance, sentence segmentation, punctuation removing, tokenization, stemming, stop words removing, and more to get the document term matrixes. Based on the matrixes advance analysis such as feature extraction or machine learning can be developed accordingly.

In the data preparing is the pre process for the procedure of text mining. Usually the original texts are stored in a semi structured formats for instance tables or XML formats. In this pre process removes unnecessary information and collects only what would like to use. In this work also includes the process of data cleaning. After this process transfer the texts to corpus, which usually are long strings stored in tables.

A part of a document-term matrix

Item	aa	ac	add	adiod	aim	airac	amp	and	apply	範本	錯誤 座標	錯誤	幫忙
2014010007	1	0	0	0	1	0	0	0	0	0	0	0	0
2014010008	1	0	0	0	1	0	0	0	0	0	0	0	0
2014010009	1	0	0	0	0	0	1	0	0	0	0	0	0
2014010010	2	0	0	0	2	0	0	0	0	0	0	0	0
2014010020	0	0	0	0	0	0	0	0	0	0	0	0	0
2014010036	0	0	0	0	0	0	1	0	0	0	0	0	0
2014010043	1	0	0	2	0	0	0	0	0	0	0	0	1
2014010046	1	0	0	0	0	0	0	0	0	0	0	0	0
2014010055	1	0	0	0	2	0	0	0	0	0	1	0	0
2014010056	1	0	0	0	2	0	0	0	0	0	1	0	0
2014010058	1	0	0	1	0	0	0	0	0	0	1	0	0
2014010059	1	0	0	1	0	0	0	0	0	0	1	0	0
2014010094	0	0	0	0	0	0	0	0	0	0	0	0	0
2014010153	0	0	0	0	0	0	0	0	0	0	0	0	0
2014010196	0	0	0	0	0	0	0	0	0	0	0	0	0
2014010214	0	0	0	0	0	0	0	0	0	0	0	0	0
2014010215	0	0	0	0	0	0	0	0	0	0	0	0	0
2014010216	0	0	0	0	0	0	0	0	0	0	0	0	0
2014010223	0	0	0	1	0	0	1	0	0	0	0	0	0
2014010233	0	0	0	0	0	0	1	0	0	0	0	0	1
2014010234	0	0	0	0	0	0	0	0	0	0	0	0	0
2014010237	0	0	0	0	0	0	0	0	0	0	0	0	0
2014010242	0	0	0	1	0	0	1	0	0	0	0	0	0
2014010252	0	0	0	1	0	0	1	0	0	0	0	0	0

Figure 4: sample of document term matrix

After getting the corpus, focus on operating long strings and splitting them by special rules. For instance, if have to do sentence segmentation in English texts can just to split multiple sentences by periods, exclamation marks or question marks. After finished the sentence segmentation, each sentence is split by spaces to independent words or terms. After all terms are split these terms may do additional operations for instance, to do the stemming process that can transfer related words into a single term. It can save the storage and simplify the following operations and removing stop words are also efficient methods to achieve this goal because they usually are not useful for further analysis.

The authors in [2] presented the product life cycle analysis and next generation data analysis on structured and unstructured data.

The large amount of data for instance, emails, customer complaints and failure reports are abundant around the product life cycle and that provide a huge potential for analysis optimization. An estimation of the unstructured data within companies range is among 50% to 80% of all data. In addition, in the product life cycle later phases, there are often unstructured form for example data from social web such as tweets, posts, and blogs.

To considering three major insufficiencies of limiting business improvement:

- Companies focus on the data sources from a single product life cycle for instance data are mined from frequent complaints of customer relationship management system by without considering manufacturing failure reports that related to same product.
- For holistic analysis companies do not use of structured data for instance, that co-relate unstructured failure reports with structured data of the manufacturing execution system.

- An analytics components and implementations of data integration are manual and case based, cost intensive, and without general framework.

Basically, structured data are stored in traditional databases that structured by rows and columns and mostly in numeric way. While unstructured data are content of documents for instance text files, images, pdfs, video files, audio files, etc. In this paper strongly focus on textual unstructured data. The following figure shows the data sources which are created and accessed during the entire product life cycle. In the production phases and planning phases there is higher volume of structured data. And during the design and usage phases there is higher volume of unstructured data. Basically, unstructured data sources are used for analysis of social media content and customer feedback from the maintenance and usage phases.

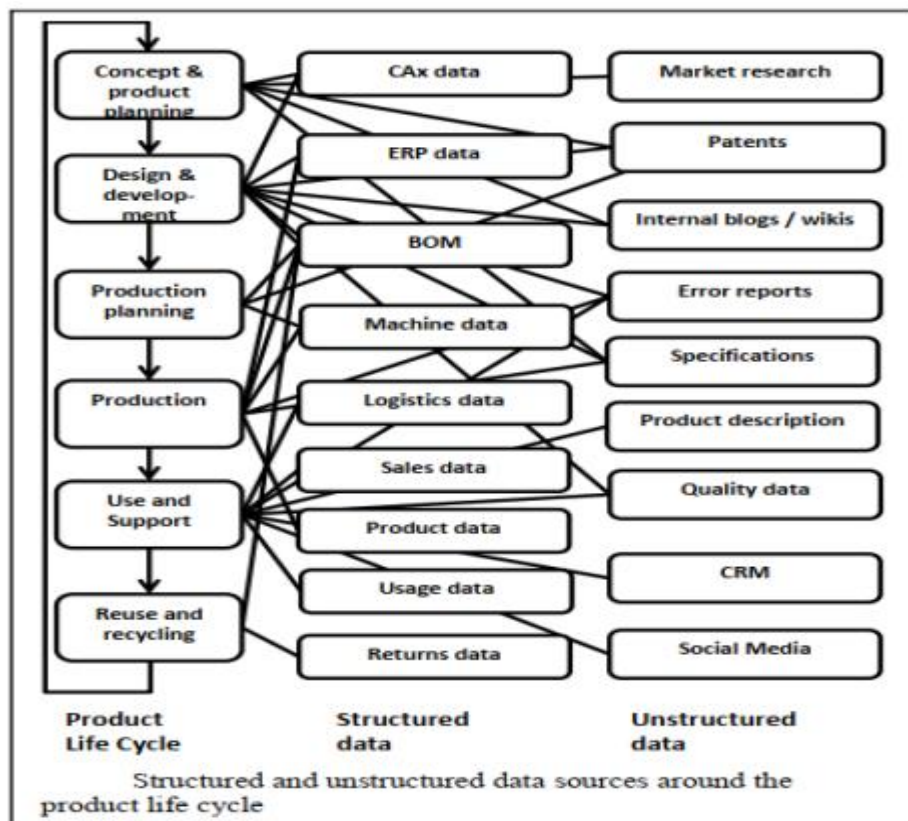


Figure 5: Structured and unstructured data sources across the PLC

Collection of unstructured data that are separated from structured data and extracted unstructured content that are integrated into the structured data warehouse. Basically unstructured data in the form of the textual reports in particular quality management that is related to development and maintenance phases of the product life cycle. Structured data may be connected with this textual quality report for instance machine error codes from production and machine diagnosis data from maintenance. Textual structured data from customer relationship management systems that also contain of quality information.

The authors in [3] presented the knowledge based production documentation analysis of integrated text mining architecture.

The technological complexity is rising steadily in high technological companies and tightening the global demands which are leading to conflict on the product for instance it has to be produced faster, cheaper, more customer related and the entire product without failures. And new need for production is rising. Organizational knowledge is captured and managed by increasing the number of methods and processes and achieved solutions to continuously reintegrated into the overall process of design. In order to gain the advantage in high technology production markets need to track the product, product related design and production related documents.

Production failures can create critical production scenarios that going to rising costs if that longer run may shift the market position. So needs to improve even single processes for example failure prevention, fully utilization of intellectual capital of the organization and improvements. In order to reach this stage access to information is essential and in case of existing information has to be used, reused, maintained, and stored. Text mining is the method for processing and analysis of unstructured textual data.

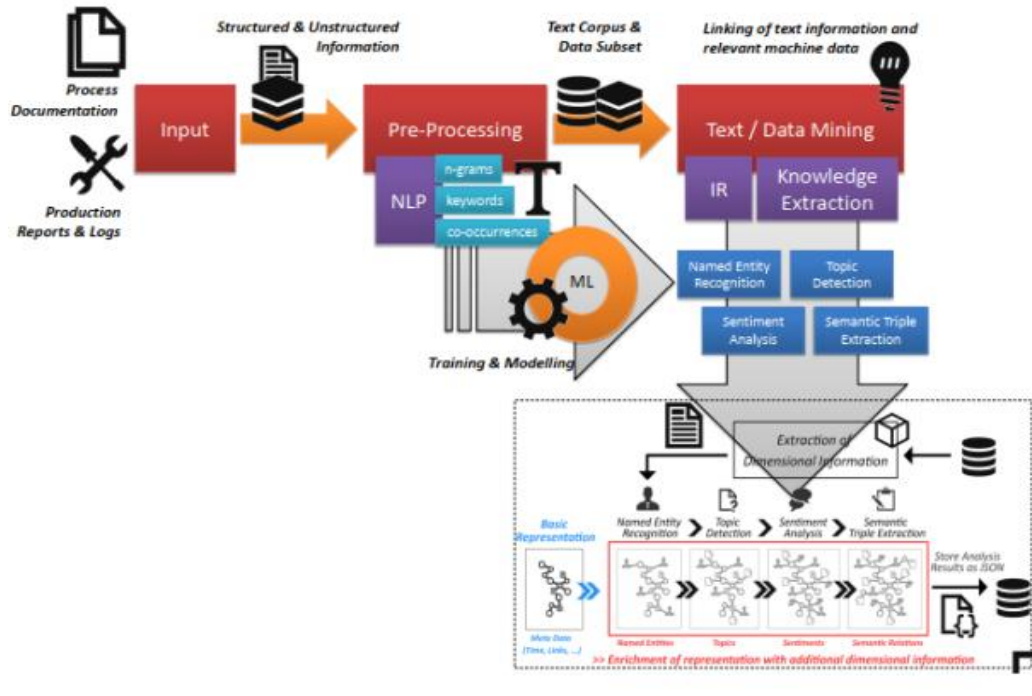


Figure 6: Integrated text mining architecture

Integration text mining that combines of multiple results from text mining methods into a holistic view of the data to cover different perspectives and individual analysis. Application of the text mining to extract, describe and contextualize unstructured textual sources and represent the results in a way that they can be useful in different stage of the overall product life cycle.

3. Case study:

3.1. Data source

Description: Automotive products dataset consists of reviews and product information was collected from Amazon. The Amazon review dataset for Bosch icon 21A windshield wiper blade product were considered. The reviews and ratings given by users to products as well as reviews about user's experience with the product were also considered.

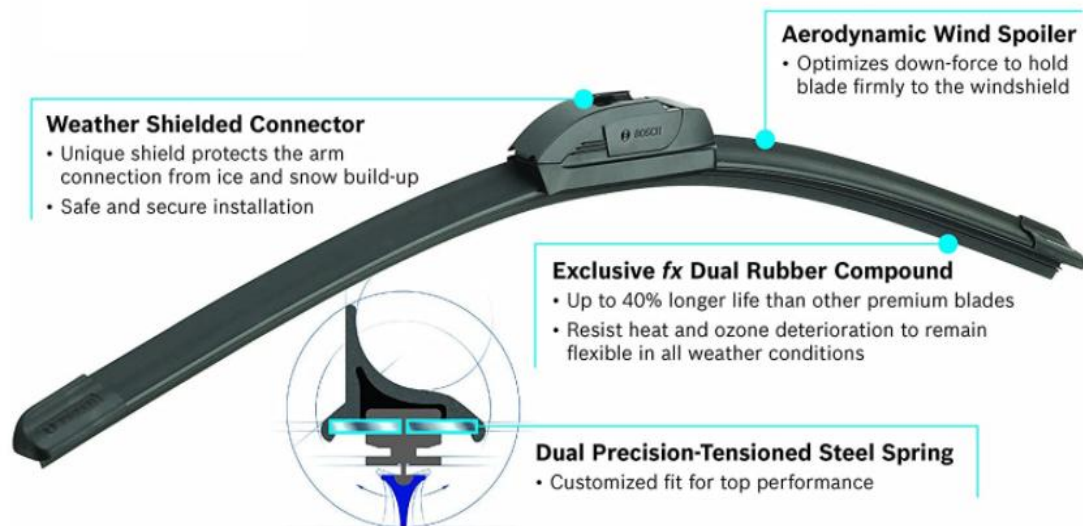
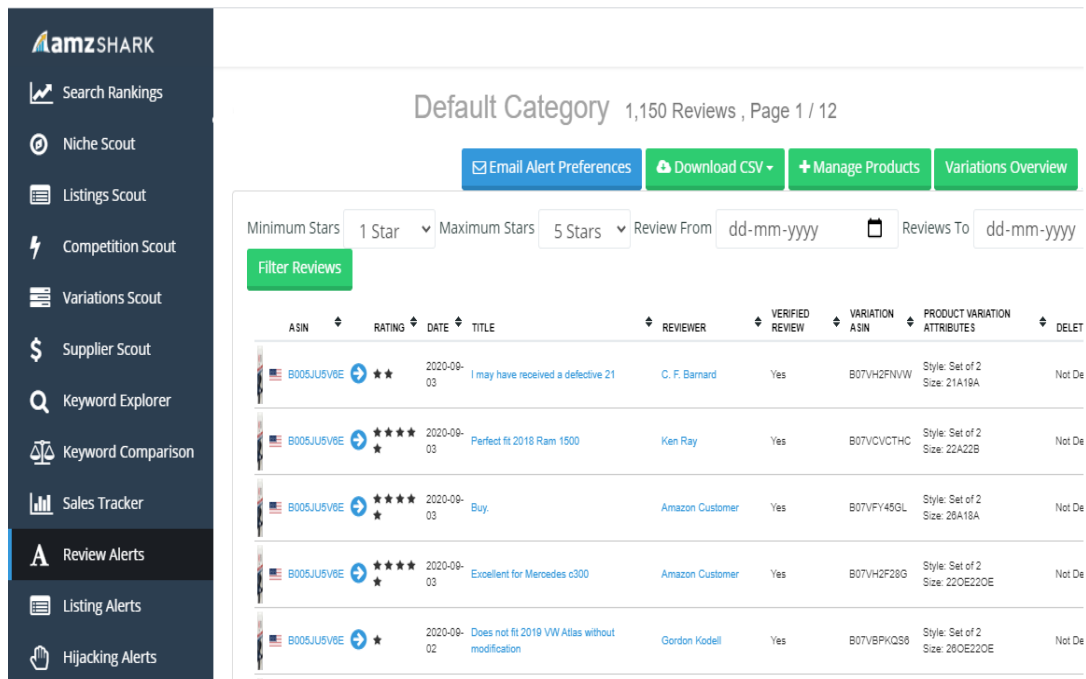


Figure 7: Bosch windshield wiper blade

Bosch bracket-less wiperblades are designed with no brackets and hinges that improved design and that performance compared to conventional wiper blades. The design of Bosch-ICON wiper distributes much uniform pressure along the entire length of the blade and for all seasonal performance that lasts up to 40% longer than other premium wiper blades. And also integrated design wind spoiler increases the down force on the wiperblade to prevent lift at high speeds of vehicles as well as bracket less design and enclosed springs provide extreme wiping performance in all weather conditions.

3.2. Data collection:

The dataset was obtained from amzshark.com. First, I chose the Bosch windshield wiper blade product from Amazon.com with the help of its product ID can download the dataset in CSV format. AMZShark provides the tracking sale records to seller and manufacturer.



The screenshot shows the AMZSHARK web application. On the left is a dark sidebar with the AMZSHARK logo and a list of tools: Search Rankings, Niche Scout, Listings Scout, Competition Scout, Variations Scout, Supplier Scout, Keyword Explorer, Keyword Comparison, Sales Tracker, Review Alerts (highlighted), Listing Alerts, and Hijacking Alerts. The main content area has a header with 'Default Category', '1,150 Reviews', and 'Page 1 / 12'. Below this are buttons for 'Email Alert Preferences', 'Download CSV', 'Manage Products', and 'Variations Overview'. A filter section includes 'Minimum Stars' (1 Star), 'Maximum Stars' (5 Stars), 'Review From' (dd-mm-yyyy), and 'Reviews To' (dd-mm-yyyy), with a 'Filter Reviews' button. The main table displays a list of reviews with columns: ASIN, RATING, DATE, TITLE, REVIEWER, VERIFIED REVIEW, VARIATION ASIN, PRODUCT VARIATION ATTRIBUTES, and DELET. The table contains five rows of review data.

ASIN	RATING	DATE	TITLE	REVIEWER	VERIFIED REVIEW	VARIATION ASIN	PRODUCT VARIATION ATTRIBUTES	DELET
B005JUSVBE	★★★	2020-09-03	I may have received a defective 21	C. F. Barnard	Yes	B07VH2FNWW	Style: Set of 2 Size: 21A19A	Not De
B005JUSVBE	★★★★★	2020-09-03	Perfect fit 2018 Ram 1500	Ken Ray	Yes	B07VCVCTHC	Style: Set of 2 Size: 22A22B	Not De
B005JUSVBE	★★★★★	2020-09-03	Buy.	Amazon Customer	Yes	B07VVFY45GL	Style: Set of 2 Size: 26A18A	Not De
B005JUSVBE	★★★★★	2020-09-03	Excellent for Mercedes c300	Amazon Customer	Yes	B07VH2F28G	Style: Set of 2 Size: 22OE22OE	Not De
B005JUSVBE	★★★	2020-09-02	Does not fit 2019 VW Atlas without modification	Gordon Kodell	Yes	B07VBPQD50	Style: Set of 2 Size: 26OE22OE	Not De

Figure 8: AMZshark data collection tool

AMZShark is an Amazon seller solution that actually complete package of 13 tools that give the right solution for every step of Amazon selling process from finding listing optimization, keyword research, product managing etc. It also supports every Amazon marketplace and comes with great educational support. The main features includes such as sales tracker review, alerts keyword explorer, feedback alerts, and more. It is one of largest and most comprehensive amazon management packages but it is not a free the pricing is high compared to other options out there.

This dataset includes reviews for instance ratings, product id, reviewer names, ASIN, reviewer texts, review dates, etc. In this dataset file has included total 899 review texts that provided by reviewers. The sample of dataset is shown below is in CSV file format.

	A	C	D	E	G	H	J	K	L	M
1	Category	ASIN	Country Code	Product Title	Review Rating	ReviewDate	ReviewerName	Reviewer ID	ReviewTitle	ReviewText
2	Default Category	B005JU5V6E	US	Bosch ICO	5	06-05-2020	Stanley Bryant	AFTCBACE	Awesome	I give them 5 stars!!! Well deserved cheaper than in store p
3	Default Category	B005JU5V6E	US	Bosch ICO	5	15-04-2020	Desmond Rader	AF7K4UQ	Quiet	Great wipers, they are much more effective and quiet comp
4	Default Category	B005JU5V6E	US	Bosch ICO	5	28-03-2020	Darshon Crudup	AHKOWM	Good purchase	Great price, great product!
5	Default Category	B005JU5V6E	US	Bosch ICO	5	23-03-2020	Kai Renee	AHGJHRZ	Best wipers i h	I went to autozone for replacement wipers and discovered
6	Default Category	B005JU5V6E	US	Bosch ICO	5	22-03-2020	Bob	AHMQVG	Excellent qualiti	Great product with a reasonable price. Used for several mo
7	Default Category	B005JU5V6E	US	Bosch ICO	1	21-03-2020	Amazon customer	AEAU5T	Poor quality	Don't last long.
8	Default Category	B005JU5V6E	US	Bosch ICO	1	19-03-2020	George W Farmer	AFP4HXAF	Awaiting refund	Even though my vehicle was logged into the search the
9	Default Category	B005JU5V6E	US	Bosch ICO	5	12-03-2020	Steven B.	AGVESNZ	Best wipers I h	I have ordered and used windshield wipers on my three car
10	Default Category	B005JU5V6E	US	Bosch ICO	5	29-02-2020	Amazon Customer	AF7MSRIB	Easy to install	Easy to install only took 2 minutes. The directions are very c
11	Default Category	B005JU5V6E	US	Bosch ICO	5	27-02-2020	Thomas Weber	AFLS7S5L	Great wipers!	C Best wipers I???ve ever purchased. Installed quickly. Does t
12	Default Category	B005JU5V6E	US	Bosch ICO	5	25-02-2020	Stephen Shufelt	AFZ2UAG	Only blade to b	Best blade ever
13	Default Category	B005JU5V6E	US	Bosch ICO	5	23-02-2020	MD	AGIVVHH	As described.	As described.
14	Default Category	B005JU5V6E	US	Bosch ICO	5	23-02-2020	yvette_molina	AFJB3D7I	Does the job gc	I struggled at first to put them on but after a quick YouTube
15	Default Category	B005JU5V6E	US	Bosch ICO	5	20-02-2020	rxzruby	AG47TTS	Thumbs Up	Lasted a year.
16	Default Category	B005JU5V6E	US	Bosch ICO	5	18-02-2020	Harry	AG4K6GBJ	Excellent	Excellent
17	Default Category	B005JU5V6E	US	Bosch ICO	5	14-02-2020	RENE 1-OSO	AHXSQVEI	Read the instru	Great product however installation, READ the instructions t
18	Default Category	B005JU5V6E	US	Bosch ICO	5	07-02-2020	Ron Borowski	AFBVXFG	As Expected	Bosch front wipers performs as expected on my Jeep. Excel
19	Default Category	B005JU5V6E	US	Bosch ICO	5	31-01-2020	Heliopolis	AEIB5S5F	Great quality w	Super clean wiping, these are solid quality units, very easy
20	Default Category	B005JU5V6E	US	Bosch ICO	5	28-01-2020	NEO	AEPRD7V	Easy installatio	These were super easy to install on my 1997 Subaru Outbac
21	Default Category	B005JU5V6E	US	Bosch ICO	3	14-01-2020	mark cramer	AHUYUN	It's a bad wipe	It sometimes don't clean windows coming up but cleans it g

Figure 9: dataset format

Description of most useful includes variables in the dataset:

- ASIN : ID of the product e.g B005JU5V6E
- Reviewer title: conclude of the review title
- Review text: text of the review
- Review rating: Rating of review e.g 3/5. Product score granted by the customer from 1 worst to 5 is best.
- Time of review

However, based on above variables there are several supervised and unsupervised techniques are used to perform on the dataset to get several information on customer preferences.

3.3. CSV file format:

A CSV is called comma separated values file which allows the data to be saved in tabular format. CSVs look like a variety of spreadsheet but with a .csv extension. CSV can be used in any spreadsheet file program such as Microsoft Excel and Google Spreadsheets. These files are different from other spreadsheet file types because they can have only a single sheet in a file; they cannot save cell, column, and row or cannot save formulas in this format. Basically, this type of file is used in companies in order to export a high volume of data to a more concentrated database for example to better organize a large amount of data and they are very easy to import into specific software.

4. Method:

4.1. Overview

In this study to consideration of text mining is the process of examine of large collections of text and converting the unstructured data text into structured data for further analysis like tf-idf analysis and clustering analysis in order to understand the strengths and defects of products as well as the depth analysis of the customer reviews. Moreover, Customer reviews are a great source of voice of customer and offer the lot insights into what consumers like and dislike about the product or service.

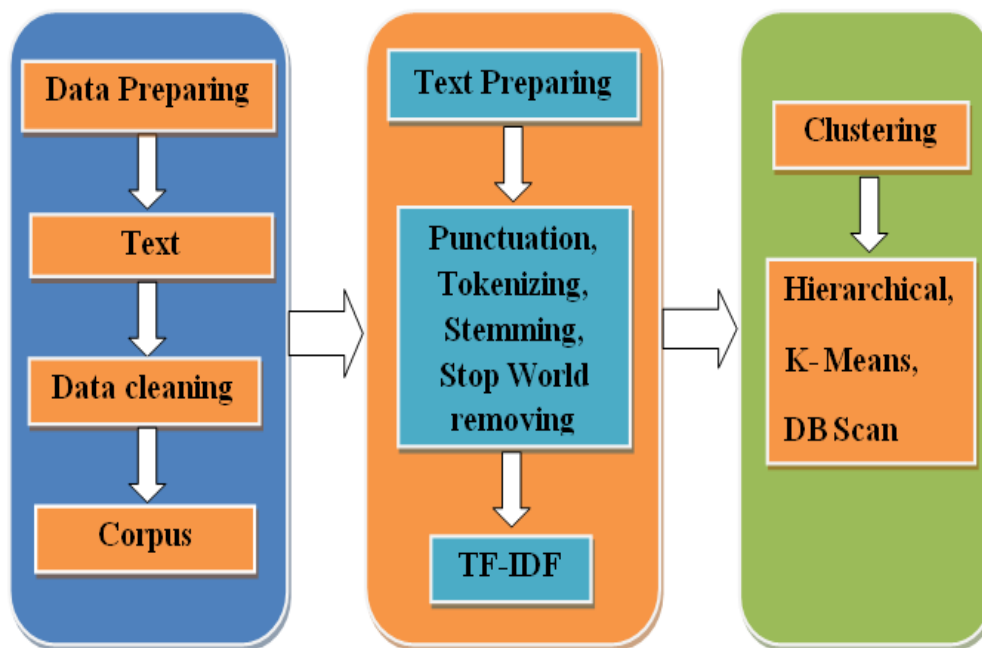


Figure 10: Methods for text mining

The above figure shows the architecture of the application using text mining and clustering analysis in order to get the results.

Basically machine learning model does not work with the text data. It needs to be cleaned and converted into numerical values. This process is called text pre processing. There are some basic steps are considered in the following steps. First of all, need to see what the data is about and what parameters are in the dataset.

In text mining, the step is the data cleaning where removes those words from the review texts which may not contribute to the information that we want to extract. In review texts may contain lot of undesirable characters like punctuation marks, stop words, digits which are not useful for analysis. And also there are some words which are same in the sense. For instance, words such as “friendly”, “friendliness”, “friends”, etc. Which are repeated therefore reduce them to single word such as friend.

Techniques are used for encoding text data: there are many techniques for encoding text data for instance:

- Bag of words
- Bi grams and n-grams
- TF-IDF
- Avg-word2vec

Note, IF-IDF method is considered for the analysis of above mentioned dataset.

Conclude, Machine learning algorithms are totally depends on data that makes model training possible. If make sense out of data before import for machine algorithms, machine will be useless. So, need to import always right data for instance, correct formatting contains etc.

4.2. Data preparation

In this section explained the some basic concepts and steps for data preparation. Data preparation is the first step after get hands on any kind of dataset. In data preparation, pre process raw data into form that can easily and accurately analyzed and proper data preparation allows for efficient analysis so it can be easily eliminate errors and inaccuracies that generated during the data gathering process and that help to removing some error from poor data quality.

In data preparation, there are several steps are involved such as loading dataset into software, transforming the data formats, and rearranging data and so on and used python pandas library for data preparation.

Before, load the data into a pandas data frame, first import the necessary libraries.

Input

```
import numpy as np
import pandas as pd
import seaborn as sns;
import matplotlib.pyplot as plt
import os
import re
from wordcloud import WordCloud
```

Now, import the dataset into pandas.

Input

```
reviews_df = pd.read_csv("Reviews.csv")
reviews_df
```

Code snippet 1: Import libraries and dataset

Checking the dataset

Output

Reviewer_Score	Review Date	Reviewer URL	Reviewer Name	Reviewer ID	Review Title	Reviewtext
5	06-05-2020	https://www.amazon.com/gp/profile/amzn1.account...	Stanley Bryant	AFTCBACEH2U2JIAMVLGU6XCQXZNA	Awesome	I give them 5 stars.Well deserved cheaper than...
5	15-04-2020	https://www.amazon.com/gp/profile/amzn1.account...	Desmond Rader	AF7K4UQOR4AQ5RHVG57E63QNATDA	Quiet	Great wipers, they are much more effective and...
5	28-03-2020	https://www.amazon.com/gp/profile/amzn1.account...	Darshon Crudup	AHKOWMTGC42AF3JDZDKBFJCEFLXQ	Good purchase.	Great price, great product

Input

```
reviews_df.shape
```

Output

```
899 rows x 18 columns
```

In this dataset, there are includes 899 rows and 18 feature variables and each row related to customer review. From those columns, reviewer score and review text are useful for further analysis.

The main objective is in given review text to predict whether the review is positive or negative.

Let's first see the reviewer score column

Input

```
reviews_df['Reviewer_Score'].value_counts()
```

Output

```
5    694
1     81
4     62
3     36
2     26
```

Code snippet 2: dividing the reviewtext based on rating score

According to output, the reviewer score column has values from 1 to 5. Our main aim is to predict weather given review is positive or negative. Here, for further analysis values 1 and 2 are considered as negative reviews and value 5 is considered as positive and other values discard those rows where reviewer score values 3 and 4. Because, there are probability of reviewer score values 3 and 4 are combined of positive and negative contains. So, in order to get the accurate results neglect the reviewer score values 3 and 4.

4.2.1. For positive reviews

Now, considering the data contain that only with the rows containing reviewer score value 5. It is a positive rating score. Let's convert the reviewer score values into the class label positive instead of 5 else negative.

Input

```
def xyz(x):  
    if x<=4:  
        return 'negative'  
    else:  
        return 'positive'  
s=reviews_df['Reviewer_Score']  
d=list(map(xyz,s))  
reviews_df['Reviewer_Score']=d  
reviews_df.head()  
  
pos= reviews_df1[reviews_df1.Reviewer_Score != 'negative']  
pos=pos.reset_index()  
pos
```

Code snippet 3: Consider only positive reviews

For viewing output as only positive reviews and removed all negative reviews which reviewer score values less than 5. So, get the filtered of only positive reviews which have only score value 5.

Output

index		Reviewtext	Reviewer_Score
0	0	I give them 5 stars.Well deserved cheaper than...	positive
1	1	Great wipers, they are much more effective and...	positive
2	2	Great price, great product	positive
3	3	I went to autozone for replacement wipers and ...	positive
4	4	Great product with a reasonable price. Used fo...	positive
...
689	891	I was shocked at how well it cleaned the winds...	positive
690	894	My windshield comes out so clean the glass act...	positive
691	895	It installed very easily and since there is no...	positive
692	896	easy to Install wipers were so smooth and quie...	positive
693	897	They are great.	positive

694 rows × 3 columns

After observing the output results, in dataset there are only 694 positive reviews which have reviewer score value only 5.

In order to get the corpus and string of review text which is useful for further analysis, applied the following algorithms.

Input

```
finalpositive=pos.drop(['Reviewer_Score','index'], axis=1)

for i in range(8):
    print("Review #",i+1)
    print(finalpositive.Reviewtext[i])
    print()
```

Code snippet 4: Final positive reviews

Output

Review # 1

I give them 5 stars. Well deserved cheaper than in store pricing and very easy install. Have no fear or worry if you need an type wiper blade this brand comes with 3 optional fittings yours is sure to be in there.

Review # 2

Great wipers, they are much more effective and quiet compared to my last pair

Review # 3

Great price, great product

Review # 4

I went to autozone for replacement wipers and discovered they are more than expensive for trucks. I had a checklist of things i needed from my new wipers. Steady swipes, No noise when operating, and longevity. they work better than my old factory wipers and the Goodyear wipers i have on my car

Review # 5

Great product with a reasonable price. Used for several months now with excellent performance.

Review # 6

I have ordered and used windshield wipers on my three cars for years. These are the best performing and longest lasting wipers I have experienced. You simply can not go wrong with these wiper blades. highly recommend

Review # 7

Easy to install only took 2 minutes. The directions are very clear

Review # 8

Best wipers I have ever purchased. Installed quickly.

4.2.2. For negative reviews

Now, considering the data contain that only with the rows containing reviewer score values 1 and 2. It is negative rating score. Let's convert the reviewer score values into the class label positive instead of 1 and 2.

Input

```
def xyz(x):  
    if x<3:  
        return 'negative'  
    else:  
        return 'positive'  
s=reviews_df['Reviewer_Score']  
d=list(map(xyz,s))  
reviews_df['Reviewer_Score']=d  
reviews_df.head()
```

For viewing output as only negative reviews and removed all other reviews which reviewer score values greater than 1 and 2. So, get the filtered of only negative reviews which have only score value 1 and 2.

Input

```
neg= reviews_df1[reviews_df1.Reviewer_Score != 'positive']  
neg=neg.reset_index()  
neg
```

Code snippet 5: Consider only negative reviews

Output

index		Reviewtext	Reviewer_Score
0	5	Don't last long.	negative
1	6	service is not good	negative
2	21	Consistently leave streaks in the middle of th...	negative
3	27	My only intention of purchasing these costly w...	negative
4	31	long lasting.	negative
...
102	859	The wipers are excellent, the two sets I recei...	negative
103	875	I think these wipers are worse than the cheap ...	negative
104	884	I put these on in June and now they are not on...	negative
105	887	Amazon said that this was an exact fit for my ...	negative
106	889	Expensive and loud when the blade reaches the ...	negative
107 rows × 3 columns			

After observing the output results, in dataset there are only 107 positive reviews which have reviewer score value only 1 and 2.

In order to get the corpus and string of reviewtext which is useful for further analysis, applied the following algorithms.

Input

```
finalnegative=neg.drop(['Reviewer_Score','index'], axis=1)
finalnegative.reset_index()

for i in range(8):
    print("Review #",i+1)
    print(finalnegative.Reviewtext[i])
    print()
```

Code snippet 6: Final negative reviews

Output

Review # 1

Don't last long.

Review # 2

service is not good

Review # 3

Consistently leave streaks in the middle of the windshield wiper and becomes a major annoyance during heavier storms. Would not recommend this particular product.

Review # 4

My only intention of purchasing these costly wipers was to minimize the noise and skipping with the blades. These are worse than the OEM wiper inserts from Forde. I am very disappointed with these wiper blades

Review # 5

long lasting.

Review # 6

Horrible, Leaves streaks and chatters.

Review # 7

So not really sure how to rate this. I bought two new sets for my Jeep and truck. The ones for my Jeep work great. The ones for the pick up I want to complain about not only a week and the are skipping and making noise. I wish I could return them and if get as new pair sent out for they do not work well and are not going to last long doubt they will make it through the winter and or I might just get rid of them due to the noise. Not happy with those but the Jeep works great.

Review # 8

Terrible wear quality. Was a dry summer with little use and already leaving lines and not wiping clean. Worse after 3 months than my last pair after a year. Not worth the money.

4.3. Text Preparation

To ensure that cleaned up and represent the root set of relevant words, set of preprocessing activities need to be performed on the string or corpus of the review text.

The following some of the pre processing common methods are considered in the text preparation task.

4.3.1. Contractions

Basically, Contractions are words or combination of words which are dropping the letters and replacing them by apostrophe for instance ain't = am not, can't= cannot, shan't= shall not etc. actually contractions are abbreviation for sequence of the words. There are main two reasons to deal with contractions in natural language processing. First, software doesn't know that contractions are abbreviations for sequence of words. So software considers can't and can not to be two different things and that does not recognize these two terms but that have same meaning. Second, that increases the dimension of document term matrix. So, computation becomes more expensive.

This implemented in the python packages. In order to enable this package use the following command:

```
contractions = { "ain't": "am not", "aren't": "are not", "can't":  
"cannot", "can't've": "cannot have", "'cause":  
"because", "could've": "could have",  
"couldn't": "could not", "couldn't've": "could not  
have", "didn't": "did not", "doesn't": "does not", "don't": "do  
not", "hadn't": "had not",  
"hadn't've": "had not have", "hasn't": "has not", "haven't": "have  
not", "he'd": "he would", "he'd've": "he would have", "he'll": "he  
will",
```

```
"he's": "he is", "how'd": "how did", "how'll": "how will", "how's":  
"how is", "i'd": "i would", "i'll": "i will", "i'm": "i am",  
"i've": "i have", "isn't": "is not", "it'd": "it would", "it'll":  
"it will", "it's": "it is", "let's": "let us", "ma'am": "madam",  
"mayn't": "may not", "might've": "might have", "mightn't": "might  
not", "must've": "must have", "mustn't": "must not", "needn't":  
"need not",  
"oughtn't": "ought not", "shan't": "shall not", "sha'n't": "shall  
not", "she'd": "she would", "she'll": "she will", "she's": "she  
is",  
"should've": "should have", "shouldn't": "should not", "that'd":  
"that would", "that's": "that is", "there'd": "there had",  
"there's": "there is", "they'd": "they would", "they'll": "they  
will", "they're": "they are", "they've": "they have", "wasn't":  
"was not",  
"we'd": "we would", "we'll": "we will", "we're": "we are", "we've":  
"we have", "weren't": "were not", "what'll": "what will",  
"what're": "what are", "what's": "what is", "what've": "what  
have", "where'd": "where did", "where's": "where is", "who'll":  
"who will",  
"who's": "who is", "won't": "will not", "wouldn't": "would  
not", "you'd": "you would", "you'll": "you will", "you're": "you  
are", }
```

Code snippet 7: Contractions

4.3.2. Convert all characters to lowercase

Transforming all texts to lowercase is a very common preprocessing method. In python “lower” is method used for string handling. These methods return the texts in lowercased string from the given strings. Basically, it converts the upper case characters to lower case characters if there is no upper case characters exists, it returns to the original strings.

```
Input  
text= "GREAT WIPER MUCH EFFECTIVE QUIET COMPARE LAST PAIR"  
lowercase = text.lower()  
lowercase  
  
Output  
great wiper much effective quiet compare last pair
```

Code snippet 8: Convert to lower case

4.3.3. Remove punctuations

Punctuation is mostly removed from strings or corpus when analyse the data. Punctuation includes the characters such as commas and period of semicolons such as “! # & ' () \$* + , - . / ; < > ? = @ [\] ^ _ : ' { | % } ~”. These punctuations are not useful and they do not carry any kind of information. So, in order to get accurate results remove the all punctuations from corpus.

4.3.4. Remove stopwords

There are different techniques are used for removing stop words from strings in python programming. Stopwords are words in NLP that have very little meaning for instance, “he, she, is, they, don’t, the, etc”. Stopwords are basically useless words and do not carry meaningful information to sentence.

By using the python programming language, there are options in order to remove stopwords from strings. We can use either one of the several natural language libraries for instance Spacy, Nltk, Textblob, Gensim, etc., or if require full control on the stopwords that can also remove certain words that we want to remove from strings. I used the nltk library, which is the one of the most commonly used in python libraries for NLP. Nltk supports stopwords removal and also find the full list of stopwords o the corpus. In order to remove the stopwords from sentence, need to divide the text into words and then try to remove the stopword in sentence if it is exits in the list of stopwords.

Input

```
from nltk.corpus import stopwords
stop=set(stopwords.words('english'))
print(stop)
```

Code snippet 9: input for stopwords

Output

```
{'few', 'don', 'not', 'over', 'been', "didn't", 'other',  
"doesn't", 'or', 'ours', 'if', 're', "wouldn't", 'the', 'wasn',  
"you've", 'were', "weren't", 'for', 'when', 'there', 'from',  
'once', 'has', 'that', 'both', 'after', 'whom', 'to', 'further',  
'having', 'didn', 'here', "shouldn't", 'him', 'he', 'aren',  
'this', 'won', 'only', "she's", 'its', 'an', 'she', 'and',  
'than', 'into', 'himself', 'wouldn', 'own', 'while', 'who',  
'is', "it's", 'can', 'against', 'no', 'which', 'theirs', 'do',  
'will', "hadn't", 'myself', 'they', 'had', 'down', 'too',  
'more', 'y', 'weren', 'me', 'i', 'ain', 'any', 'her', "that'll",  
'out', 'needn', 'have', 'about', 'should', 'yourself',  
'yourselves', 'how', 'each', 'mightn', 'does', 'shan',  
'shouldn', 'of', 'under', 'most', 'are', 'doesn', "you'd",  
"won't", 'on', 'am', 'before', 'but', 'these', 'mustn', 'in',  
'why', 'above', 'same', "hasn't", 'itself', 'hasn', 'ma',  
'your', 'was', 'them', 'so', "couldn't", "isn't", 'our', 've',  
'some', 'd', 'during', 'where', 'a', 'until', "you're", 'his',  
'with', 'then', "aren't", 'off', 'haven', 'such', "wasn't",  
'now', 'because', 'by', 'as', "mustn't", 'below', "should've",  
'doing', 'll', "haven't", 'again', "you'll", 'through', 'o',  
'what', "needn't", 'their', "mightn't", 'nor', 'themselves',  
's', "don't", 'at', 'hadn', 'my', 'did', 'ourselves', 'those',  
'isn', 'm', 'we', "shan't", 'yours', 'very', 'hers', 'being',  
'between', 'you', 'all', 'herself', 'be', 'just', 'up', 't',  
'couldn', 'it'}
```

4.3.5. Lemmatization

Lemmatization is the process of algorithmic of finding the lemma of word that depending on their meaning. The main aim of lemmatization is to remove the inflectional endings. Lemmatization helps to returning in base of the word which is called as the lemma. The method of the lemmatization with NLTK is based on the wordnet. In the text processing includes both stemming and lemmatization. But both are quite different meaning.

For instance, stemming process algorithm works as cutting the suffix from the word. In sense cuts either start of the word or end of the word. While, lemmatization is more powerful operation, and it returns lemma which is the root form of all inflectional forms.

Stemming process is general operation while lemmatization is smart operation where the proper form of the word. The process of lemmatization is converting word to its root form where as stemming process only remove the last few characters so it mostly leading to incorrect meaning and error in spelling.

For instance, lemmatization process is correctly identify the root form of “caring” to care, where stemming process just cut off the “ing” part and convert it to car. And, used part of speech tag for word and get appropriate lemma.

Input

```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

word_data = "wipers Leave streaks and chatters"
nltk_tokens = nltk.word_tokenize(word_data)
for w in nltk_tokens:
    print ("Actual: %s Lemma: %s" %
          (w,wordnet_lemmatizer.lemmatize(w)))
```

Code snippet 10: Input for lemmatizer

Output

```
Actual: wipers Lemma: wiper
Actual: Leave Lemma: Leave
Actual: streaks Lemma: streak
Actual: and Lemma: and
Actual: chatters Lemma: chatter
```

- Wordnet lemmatizer with nltk

Wordnet is a large and freely available lexical database for the English language. The main aim of the wordnet is to semantic relation between words. Nltk has interfaced to it but first need to download it when in order to use it and in order to lemmatize use above algorithms:

In order to lemmatize simple sentence, first need to tokenize the sentence into words by using the “`nltk.word_tokenize`” and then used “`lemmatizer.lemmatize`” on each word. Explained above algorithms is a simple example of wordnet lemmatizer on all words and strings.

- Wordnet lemmatizer with Pos tag

In wordnet lemmatizer with nltk is not perform well in some cases for example, ‘are’ form is not converted to ‘be’ form and ‘hanging’ form is not converted into ‘hang’ form as expected. This problem can be solved by proving the correct pos tag. It is not possible manually provide the correct pos tag to every word in the strings. So, need to input right character that wordnet lemmatizer accepts. By considering following algorithms it is very easy to understand the concept behind the pos tag.

Input

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

from nltk.corpus import wordnet

def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}

    return tag_dict.get(tag, wordnet.NOUN)

sentence = "The striped bats are hanging on their feet for best"
print([lemmatizer.lemmatize(w, get_wordnet_pos(w)) for w in
nltk.word_tokenize(sentence)])
```

Code snippet 11: input for pos tag

Output

```
'The', 'strip', 'bat', 'be', 'hang', 'on', 'their', 'foot',
'for', 'best'
```

4.3.6. Positive reviews after clean the texts

Subsequently, apply the following algorithms on dataset, In order to get the clean text of positive reviews.

Input

```
from nltk.corpus import wordnet
def get_wordnet_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
import string
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
def clean_text_positive(text, remove_stopwords = True):
    # Replace contractions with their longer forms
    if True:
        text = text.split()
        new_text = []
        for word in text:
            if word in contractions:
                new_text.append(contractions[word])
            else:
                new_text.append(word)
        text = " ".join(new_text)
    # Format words and remove unwanted characters
    text = re.sub(r'https?:\/\/\/*[\\r\\n]*', '', text,
flags=re.MULTILINE)
    text = re.sub(r'\<a href', ' ', text)
    text = re.sub(r'&', ' ', text)
    text = re.sub(r'[_\~;%( )|+&=%.,!?:#$@\\[\\]/]', ' ', text)
    text = re.sub(r'<br />', ' ', text)
    text = re.sub(r'\\', ' ', text)
```

```
# Optionally, remove stop words
if remove_stopwords:
    text = text.split()
    stops = set(stopwords.words("english"))
    text = [w for w in text if not w in stops]
    text = " ".join(text)

# tokenize text and remove puncutation
# lower text
text = text.lower()
# tokenize text and remove puncutation
text = [word.strip(string.punctuation) for word in
text.split(" ")]
# remove words that contain numbers
text = [word for word in text if not any(c.isdigit() for c
in word)]
# remove stop words
stop = stopwords.words('english')
text = [x for x in text if x not in stop]
# remove empty tokens
text = [t for t in text if len(t) > 0]
# pos tag text
pos_tags = pos_tag(text)
# lemmatize text
text = [WordNetLemmatizer().lemmatize(t[0],
get_wordnet_pos(t[1])) for t in pos_tags]
# remove words with only one letter
text = [t for t in text if len(t) > 1]
# join all
text = " ".join(text)
return(text)

# Clean the texts
clean_texts_positive = []
for text in finalpositive.Reviewtext:
    clean_texts_positive.append(clean_text_positive(text))
```

```
print('Positive reviews after clean the texts:')
for i in range(8):
    print("Clean Review #",i+1)
    print(clean_texts_positive[i])
    print()
```

Code snippet 12: Data cleaning for positive reviews

Output

```
Positive reviews after clean the texts:
Clean Review # 1
give star well deserve cheap store pricing easy install fear
worry need type wiper blade brand come optional fitting sure

Clean Review # 2
great wiper much effective quiet compare last pair

Clean Review # 3
great price great product

Clean Review # 4
go autozone replacement wiper discover expensive truck checklist
thing need new wiper steady swipe noise operating longevity work
well old factory wiper goodyear wiper car

Clean Review # 5
great product reasonable price use several month excellent
performance

Clean Review # 6
order used windshield wiper three car year best perform long
last wiper experience simply go wrong wiper blade highly
recommend

Clean Review # 7
easy install take minute direction clear

Clean Review # 8
best wiper ever purchase installed quickly
```

4.3.7. Negative reviews after clean the texts

Subsequently, apply the following algorithms on dataset, In order to get the clean text of negative reviews.

In the text preparation, algorithms required in order to get the clean texts of negative reviews is very similar to algorithms applied for clean texts of positive reviews.

Input

```
from nltk.corpus import wordnet
def get_wordnet_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
import string
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
def clean_text_negative(text, remove_stopwords = True):
    # Replace contractions with their longer forms
    if True:
        text = text.split()
        new_text = []
        for word in text:
            if word in contractions:
                new_text.append(contractions[word])
            else:
                new_text.append(word)
        text = " ".join(new_text)
    # Format words and remove unwanted characters
    text = re.sub(r'https?:\/\/\.[^\r\n]*', '', text,
flags=re.MULTILINE)
    text = re.sub(r'\<a href', ' ', text)
    text = re.sub(r'&', ' ', text)
    text = re.sub(r'[_"-;%()|+&=%.,!?:#$@\[ \] /]', ' ', text)
    text = re.sub(r'<br />', ' ', text)
    text = re.sub(r'\'', ' ', text)

    # Optionally, remove stop words
    if remove_stopwords:
        text = text.split()
        stops = set(stopwords.words("english"))
        text = [w for w in text if not w in stops]
        text = " ".join(text)
```

```
# tokenize text and remove puncutation
# lower text
text = text.lower()
# tokenize text and remove puncutation
text = [word.strip(string.punctuation) for word in
text.split(" ")]
# remove words that contain numbers
text = [word for word in text if not any(c.isdigit() for c
in word)]
# remove stop words
stop = stopwords.words('english')
text = [x for x in text if x not in stop]
# remove empty tokens
text = [t for t in text if len(t) > 0]
# pos tag text
pos_tags = pos_tag(text)
# lemmatize text
text = [WordNetLemmatizer().lemmatize(t[0],
get_wordnet_pos(t[1])) for t in pos_tags]
# remove words with only one letter
text = [t for t in text if len(t) > 1]
# join all
text = " ".join(text)
return(text)
```

```
# Clean the texts
clean_texts_negative = []
for text in finalnegative.Reviewtext:
    clean_texts_negative.append(clean_text_negative(text))
```

```
print('Negative reviews after clean the texts:')
for i in range(8):
    print("Clean Review #",i+1)
    print(clean_texts_negative[i])
    print()
```

Code snippet 13: Data cleaning for negative reviews

Output

```
Negative reviews after clean the texts:
Clean Review # 1
last long

Clean Review # 2
service good

Clean Review # 3
consistently leave streak middle windshield wiper becomes major
annoyance heavier storm would recommend particular product

Clean Review # 4
intention purchase costly wiper minimize noise skipping blade
bad oem wiper insert forde disappointed wiper blade

Clean Review # 5
long lasting

Clean Review # 6
horrible leaf streak chatter

Clean Review # 7
really sure rate buy two new set jeep truck one jeep work great
one pick want complain week skip make noise wish could return
get new pair send work well go last long doubt make winter might
get rid due noise happy jeep work great

Clean Review # 8
terrible wear quality dry summer little use already leave line
wipe clean bad month last pair year worth money
```

After analysed the clean text of negative reviews, we can see in review #1 and review #2 are look like positive review. Actually, review #1 and review #2 are belonging to negative reviews. It happened because during the process of stopword removing, remove the words such as don't, doesn't, etc. Therefore, original meaning of review #1 is "don't last long" and meaning of review #2 is "service is not good". If we would not apply these steps, there are lot of unnecessary words that will be added in the analysis. Therefore, we will not get accurate results as well as process is very long and become more expensive.

4.3.8. WordCloud

WordCloud is a simple data visualization technique that used to represent the data text. The size of each word indicates frequency of word and importance of word. WordCloud are most often used for analyzing data text from customer experience reviews and other social network website. WordCloud mostly filled with lot of texts in different sizes, and colours which represent the frequency and importance of each text. WordCloud also called the tag Cloud.

Basically, WordCloud format is useful for quickly identifying most prominent words. Bigger size of words means greater weight. There are main 3- types of cloud used. In the first type of cloud, it is based on the frequency of each term, whereas in second type of cloud, it is based on global tag clouds where the frequencies of texts are aggregated with all items and users. In the case of third, WordCloud contains are categorised with size that indicating the number of subcategories.

In theory, size of front in WordCloud is determined by its frequency. For instance, front sizes can be specified directly for smaller frequencies. But the larger values of frequency, a scaling should be needed. For generating the WordCloud in python, the following packets are needed to install: `“pip install matplotlib, pip install pandas, and pip install wordcloud”`.

Pros of WordCloud:

- Analyse the customer feedback.
- Identify the new keywords to target.

Cons of Word Cloud:

- WordCloud is not perfect in all situations.
- Data should be optimized for analysis.

- WordCloud for positive reviews

Input

```
print('WordCloud of positive reviews:')
for_wordcloud = clean_texts_positive
#count_vect.get_feature_names()
for_wordcloud_str = ' '.join(for_wordcloud)

wordcloud = WordCloud(width=1000, height=600, background_color
='white', min_font_size = 8).generate(for_wordcloud_str)

plt.figure(figsize=(10, 10), facecolor=None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)

plt.show()
```

Code snippet 14: WordCloud for positive reviews

Output

[illegible]

Figure 11: WordCloud for positive reviews

It seems words like “great wiper”, “wiper blade” and “easy install” are the most mentioned characters in the dataset. WordCloud is not perfect in all situations. So it is used for quick analysis for dataset.

-
- WordCloud for negative reviews

Input

```
print('WordCloud of negative reviews:')
for_wordcloud = clean_texts_negative
#count vect.get feature names()
for_wordcloud_str = ' '.join(for_wordcloud)

wordcloud = WordCloud(width=1000, height=600, background_color
='white', min_font_size = 8).generate(for_wordcloud_str)

plt.figure(figsize=(10, 10), facecolor=None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)

plt.show()
```

Code snippet 15: WordCloud for negative reviews

Output

WordCloud of negative reviews:



Figure 12: WordCloud for negative reviews

It seems words like “fit”, “leave streak”, “return”, and “windshield” are most mention characters in the dataset.

4.4. TF-IDF

4.4.1. Introduction

TF-IDF is called “term frequency – inverse data frequency”.

Computers are performing well with the numbers but that not performing well with textual data. There are several techniques used to perform text data, among them TF-IDF is one of most popular method. We are thinking that the word appears more frequently that should have greater weight in text data analysis but it's not always happen. The words like “they”, “should”, “will”, “the”, “she”, “he”, etc. is called stopwords. They always appear in the corpus text and they are not carrying any useful insights. On contrary, other words, which are rare in the corpus that useful for information carrying, they should have more weight. So, Tf-idf techniques are used often to solve this kind of problems and get accurate results. Work of Tf-idf proportionally increasing to the number of times of words appears in the sentence and it divided by the number of words in the sentence. So word is common in each sentence such as they, should, if, what, etc. it has low weight they appears many time in corpus. Basically, it defines the importance of the word in the dataset and corpus.

Tf-idf has many uses for instance, in automated text analysis and also often uses in machine learning algorithms for scoring words for natural language processing –NLP.

Applications of Tf-idf:

- Information retrieval
- Keyword extraction

Basically, machine learning algorithms with the numbers, so by the help of Tf-idf algorithms, words are allocating by the numerical value.

4.4.2. Mathematical means

Tf-idf combined two concept term frequency and inverse document frequency.

- TF- Term Frequency

Term frequency-TF defined the frequency of word in the each sentence in the corpus. Each sentence has not same length so it possible word appears in the long sentence take more time compared to word appears in sort sentence. Tf-idf is the ratio of the number of times word x appears in the sentence and total number of words in the same sentence. Each document has own term frequency. For the term frequency considered the following equation:

$$TF = \frac{\text{Frequency of word in the sentence}}{\text{Total number of words in the sentence}}$$

- IDF- Inverse Data Frequency

Equally importance is considered in term frequency while in IDF, measure the important term. For instance, certain terms like stopwords appear many times give them low importance. IDF is another concept that used to find the importance of word. So basically, IDF used to calculate the weight of words that have lower frequency across the whole documents in corpus and give them the higher importance. For the IDF considered the following equation:

$$IDF = \log \left[\frac{\text{total number of sentences(documents)}}{\text{number of sentences(documents) contain the word}} \right]$$

- TF-IDF

Tf-idf is multiplication of term frequency and inverse data frequency that output with Tf-idf score for the word in the corpus. The following equation is considered.

$$TF - IDF = \frac{\text{Frequency of word in the sentence}}{\text{Total number of words in the sentence}} * \log \left[\frac{\text{total number of sentences(documents)}}{\text{number of sentences(documents) contain the word}} \right]$$

- Example

Consider the following example to get the clear understanding.

Review1: Great wiper, they are much more effective and quiet compared to my last pair!!! .

Review2: Don't last long.

Review3: It sometimes doesn't clean windows coming up but cleans it going back down 12345 .

For example 1: I considered the word “wipers”

The word “Wiper” is one time in review1

TF = 1/15 = 0.066666666666 (the word “wipers” is one in first sentence and total 15 words in first sentence).

IDF = log (3/1) = 0.47712125 (total 3 sentences and one time word “wipers” in three sentences).

TF-IDF analysis = Tf*idf = 0.066666666666*0.47712125 = 0.031808

For Example 2: I considered the word “.”

$$TF = 1/15 = 0.066666666666$$

$$IDF = \log(3/3) = 0$$

$$TF\text{-}IDF \text{ analysis} = Tf * Idf = 0.066666666666 * 0 = 0.0000$$

The following results are the output of python coding which is similar to manual results:

```
[108]: import pandas as pd
       pd.DataFrame([tfidfBow0, tfidfBow1, tfidfBow2])
```

[108]:	.	it	don't	but	last	wipers,	clean	more	down	cleans	to	Great	quiet	compared	It
0	0.0	0.000000	0.000000	0.000000	0.011739	0.031808	0.000000	0.031808	0.000000	0.000000	0.031808	0.031808	0.031808	0.031808	0.000000
1	0.0	0.000000	0.000000	0.000000	0.044023	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.0	0.031808	0.031808	0.031808	0.000000	0.000000	0.031808	0.000000	0.031808	0.031808	0.000000	0.000000	0.000000	0.000000	0.031808

Figure 13: Output of example results of tfidf analysis

So, it seems the rare word has higher TF-IDF weight score.

From the above results we can see that for common words TF-IDF is zero which means these words are not significant. On the contrary, for other words like Wiper, great, cleans etc. are none zero TF-IDF score which means they are more significance.

4.4.3. Calculate TF-IDF by python

Now we will see that how implement TF-IDF by using sklearn (Scikit – learn) in Python.

First of all, we have to import “tfidfvectorizer” from “sklearn.feature_extraction.text”. Then start with the “vectorizer” and add the “fit.transform” in order to calculate the score of TF-IDF for text.

When uses the sklearn method, output values are different slightly because Scikit-learn uses smoothed version of IDF and few others optimizations. And it is very convenient to use when work with the large dataset or corpus.

- TF-IDF on positive reviews

Input for library

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from pandas import DataFrame
import pandas as pd
pd.set_option("display.max_columns", 100)
%matplotlib inline
```

Code snippet 16: import libraries for Tf-idf

Input for countVectorizer

```
vec_pos = CountVectorizer()
matrix_pos = vec_pos.fit_transform(clean_texts_positive)
pd.DataFrame(matrix_pos.toarray(),
columns=vec_pos.get_feature_names())
```

Code snippet 17: Count vectorizer for positive reviews

Output of countVectorizer

windshield	windsield	windy	winter	wipe	wiper	wiping	wisk	without	wonder	wonderfully	work	works	worry
0	0	0	0	0	1	0	0	0	0	0	0	0	1
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	0	4	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx
2	0	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	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	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14: Output of count vectorizer for positive reviews

Input for TfidfVectorizer

```
vec_pos = TfidfVectorizer(use_idf=True)
matrix_pos = vec_pos.fit_transform(clean_texts_positive)
tfidf_pos = pd.DataFrame(matrix_pos.toarray(),
columns=vec_pos.get_feature_names())
tfidf_pos
```

Code snippet 18: Tfidf vectorizer for positive reviews

Output for TfidfVectorizer

windshield	windy	winter	wipe	wiper	wiping	wisk	without	wonder	wonderfully	work	works	worry
0.0	0.0	0.0	0.000000	0.080420	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.26947
0.0	0.0	0.0	0.000000	0.153498	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000
0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000
0.0	0.0	0.0	0.000000	0.288552	0.0	0.0	0.000000	0.0	0.0	0.096537	0.0	0.000000
0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000
...
0.0	0.0	0.0	0.205245	0.000000	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000
0.0	0.0	0.0	0.000000	0.091783	0.0	0.0	0.24555	0.0	0.0	0.000000	0.0	0.000000
0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000
0.0	0.0	0.0	0.000000	0.122061	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000
0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000

Figure 15: Output of Tfidf vecorizer for positive reviews

- TF-IDF for negative reviews

Similar algorithms apply in this case as algorithms applied in positive reviews.

Input for countVectorizer

```
vec_neg = CountVectorizer()
matrix_neg = vec_neg.fit_transform(clean_texts_negative)
pd.DataFrame(matrix_neg.toarray(),
columns=vec_neg.get_feature_names())
```

Code snippet 19: Count vectorizer for negative reviews

Output of countVectorizer

windshield	winter	wipe	wiper	wish	within	work	worked	worn	worse	worth	would	wrap
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	1	0	0	0	0	0	0	0	1	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	2	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0

Figure 16: Output of count vecorizer for negative reviews

Input for TfidfVectorizer

```
vec_neg = TfidfVectorizer(use_idf=True)
matrix_neg = vec_neg.fit_transform(clean_texts_negative)
tfidf_neg = pd.DataFrame(matrix_neg.toarray(),
columns=vec_neg.get_feature_names())
tfidf_neg
```

Code snippet 20: Tf-idf for negative reviews

Output of TfidfVectorizer

windshield	winter	wipe	wiper	wish	within	work	worked	worn	worse	worth	would	wrap
0.000000	0.0	0.0000	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
0.000000	0.0	0.0000	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
0.151686	0.0	0.0000	0.136767	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.200069	0.0
0.000000	0.0	0.0000	0.381021	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
0.000000	0.0	0.0000	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
...
0.000000	0.0	0.0000	0.221131	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
0.000000	0.0	0.0000	0.139923	0.0	0.0	0.0	0.0	0.0	0.293436	0.0	0.000000	0.0
0.320793	0.0	0.0000	0.144621	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
0.000000	0.0	0.0000	0.137438	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0
0.107765	0.0	0.1661	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0

Figure 17: Output of Tfidf vectorizer for negative reviews

5. Clustering

5.1. Introduction

Basically clustering is the type of unsupervised machine learning method. An unsupervised machine learning method in which we can take references from dataset that consist of input data with no labelled responses. Clustering is often used as a method to find out meaningful insight, features, and inherent grouping in a set of examples.

Clustering is the process of dividing the data points and population into the number of groups and that data points in the same groups are very similar to another data points in same group and data points are dissimilar in other groups. So, we can say it formally, clustering is collection of objects that based on the similarity and dissimilarity between them or it is a group of objects that are belonging to same class.

Why Clustering needs?

It is an importance as finding out the grouping inherent among the without labelled in present data. Actually there is no particular standard for high-quality clustering. Clustering entirely depends on the user. For instance we have interested in finding out characteristic for homogeneous groups, also called data reduction, looking ordinary clusters and describe their unidentified properties, finding valuable and suitable groupings or in finding un-useful data objects is called outlier detection. These clustering techniques make some assumptions as similarity of data points and each assumption build different and quality suitable clusters. Clustering analysis is a type of exploratory data analysis in which observations are separated into dissimilar groups that allocate common characteristics from dataset.

In this chapter, I applied various different types of clustering algorithms. In following figure, I stated a little architecture of clustering algorithms that used in this chapter.

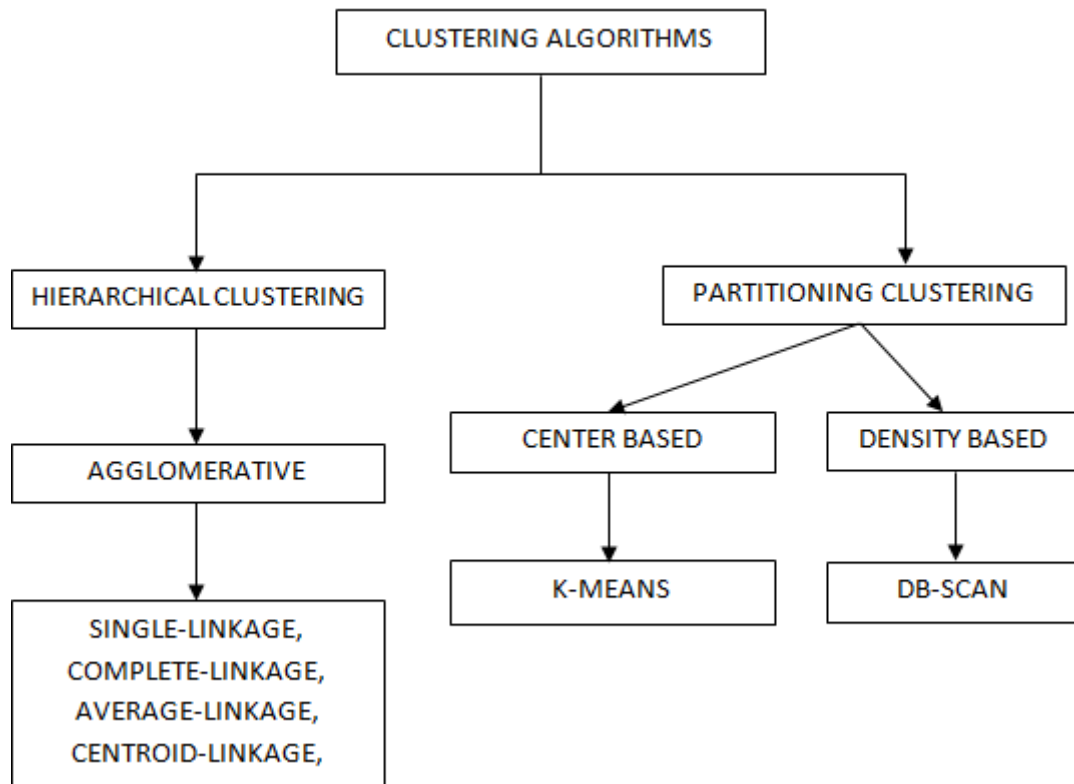


Figure 18: Types of clustering methods

The main types of clustering methods are:

- K-Means Clustering method
- Hierarchical clustering method
- DB SCAN clustering method

Hierarchical clustering method is further separated into two categories:

- Agglomerative clustering or bottom up approach
- Divisive clustering or top down approach

Applications of clustering analysis:

- Clustering analysis is generally used in various applications such as market researches, texts analysis, image processing, and pattern recognition and many more.
- Clustering can also help marketers to find out individual groups in their customer base and they can distinguish their customer groups based on the purchasing patterns.
- As a text mining function clustering analysis provides tool to achieve insight into allocation of data to observe characteristics of each cluster.
- Clustering analysis also supports in classifying documents on the web for information detection.
- Clustering analysis also used in outlier detection purpose such as detection of credit card fraud.

Requirements of clustering in text mining:

The following points are thrown on why clustering analysis is required in text mining.

- Scalability- we need very scalable clustering algorithms to deal with big databases.
- High dimensionality – the clustering algorithms shouldn't only able to handle low dimensional data but as well the high dimensional data.
- Capability to deal with noisy data- database includes noisy, missing or incorrect data. A few algorithms are sensitive to such data and may guide to poor quality clusters.
- Ability to deal with different class of attributes – algorithms should be competent to apply on any type of data such as numerical data, categorical and binary data.

5.2. Clustering algorithms

There are several types of clustering algorithms, among them I would like to express the most popular example of clustering algorithms that I used in my project.

5.2.1. Agglomerative clustering

In the hierarchical, the clustering algorithms group of comparable objects into groups is called clusters. There are basically two kinds of hierarchical clustering algorithms used:

- Agglomerative clustering – is also called “bottom up approach” that beginning with many small clusters and together merge them to create bigger clusters.
- Divisive clustering – is also called “top down approach” that beginning with single cluster and then break it into smaller clusters.

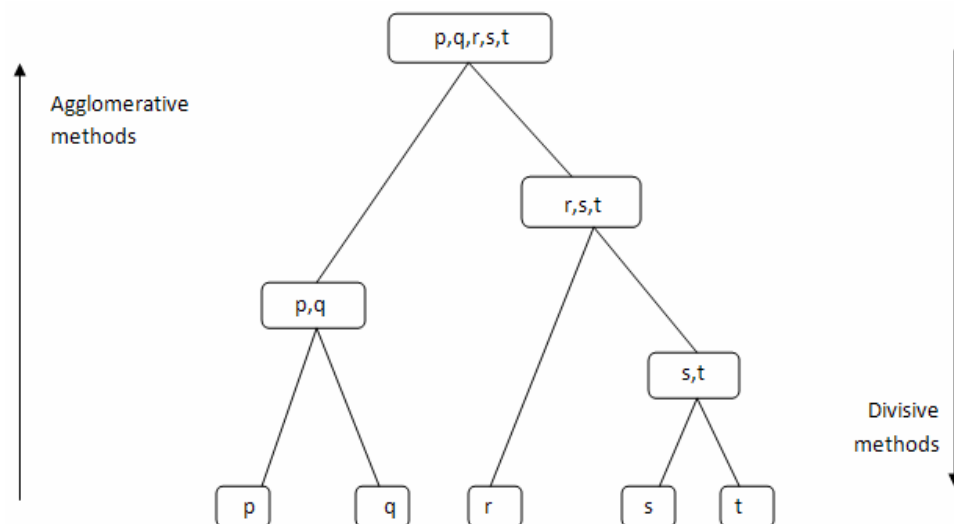


Figure 19: Concept of Agglomerative vs. Divisive clustering

An example of agglomerative and divisive clustering is shown in figure.

Dendrograms

We can utilize a dendrogram to visualize the record of groupings and figure out the best possible number of clusters.

- Find out the largest vertical distance that doesn't intersect with any of the other clusters.
- Draw the horizontal line at both extremities.
- The best possible number of clusters is equivalent to the number of vertical lines passing through the horizontal line.

For example, in the below case, greatest selection for number of cluster will be 4.

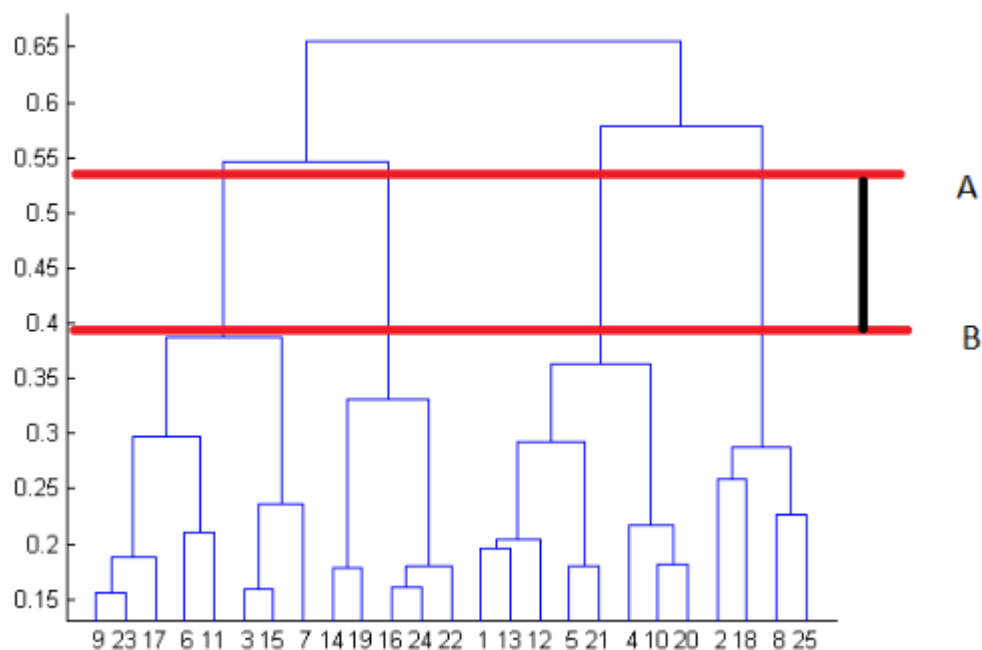


Figure 20: Sample of Dendrogram

The linkage criteria principle refers to how the distance between clusters is calculated. It says take the two closest clusters and make them one cluster.

Basically, there are four types of linkage criteria:

- **Single linkage**

The distance between two clusters is calculated as shortest distance between two points in every cluster.

$$\text{Min } \{d(r, s) : r \in A, s \in B\}$$

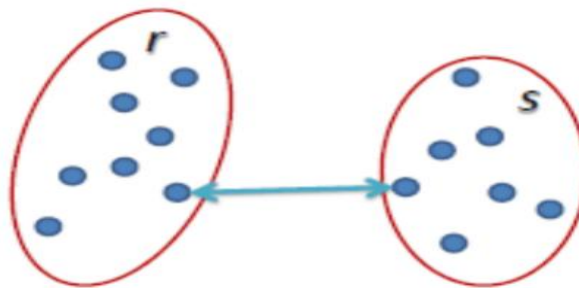


Figure 21: Single linkage

- **Complete linkage**

The distance between two clusters is calculated as longest distance between two points in every cluster.

$$\text{Max } \{d(r, s) : r \in A, s \in B\}$$

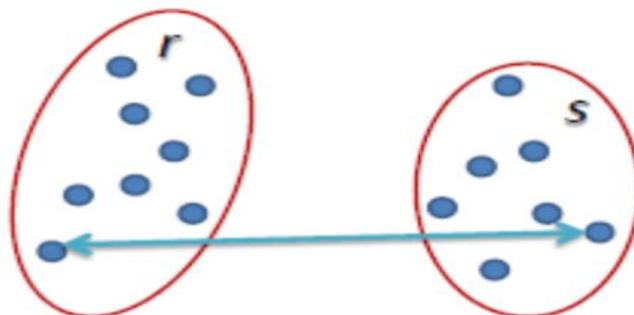


Figure 22: Complete linkage

- **Average linkage**

The distance between clusters is calculated in average linkage as the average distance between each point on certain cluster to all point in other cluster.

$$\frac{1}{|A| \cdot |B|} \sum_{r \in A} \sum_{s \in B} d(r, s)$$

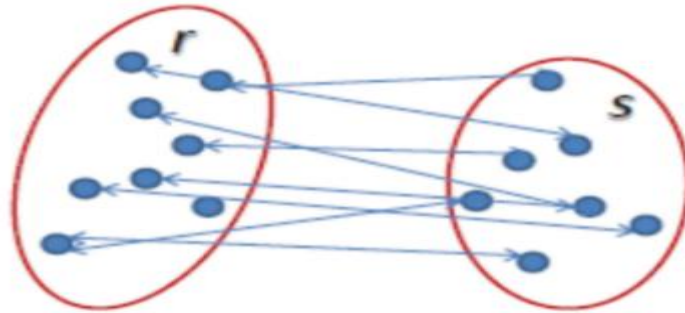


Figure 23: Average linkage

- **Ward linkage**

The distance between clusters is calculated by the sum of squared differences inside all clusters.

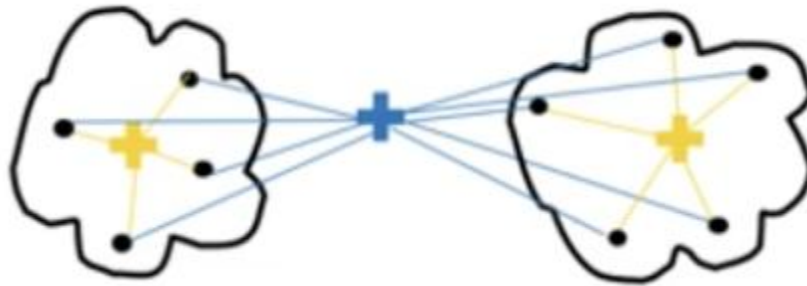


Figure 24: Ward linkage

Distance metric: It measured the distance among the data points

Euclidean distance: It calculated as the shortest distance between the two points.

If, $X = (A, B)$ and $Y = (C, D)$ so, distance is $\sqrt{(A - C)^2 + (B - D)^2}$

There are several ways to find out distance between the clusters.

- Measure the distance between the close points of two clusters
- Measure the distance between the farthest point of two clusters
- Measure the distance between the centroids of two clusters
- Measure the distance between all probable mixture of points between two clusters and take the mean.

Example:

	P1	P2	P3	P4	P5
P1	0				
P2	9	0			
P3	3	7	0		
P4	6	5	9	0	
P5	11	10	2	8	0

First of all, look at the small value on this matrix. Number 2 is the smallest one. So, point P3 and P5 is in the same cluster.

Now, find the minimum distance between P1 and [P3, P5]

$$= \min \{d(P1, [P3, P5])\}$$

$$= \min \{d(P1, P3), d(P1, P5)\}$$

$$= \min(3, 11)$$

$$= 3$$

Distance between P1 and [P3, P5] is 3.

Draw the dendrogram, according to the above output.

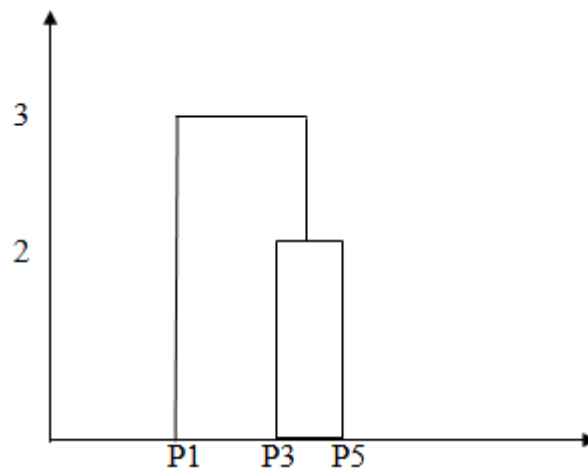


Figure 25: example results of dendrogram

First, draw the bar for P3 and P5 because they are in same cluster then and one line for P1. P1 is the minimum distance between cluster [P3, P5]. So, P1 and [P3, P5] are in the same cluster.

When applied same procedure for subsequence steps, we will get following type of dendrogram. It seems we have two numbers of clusters.

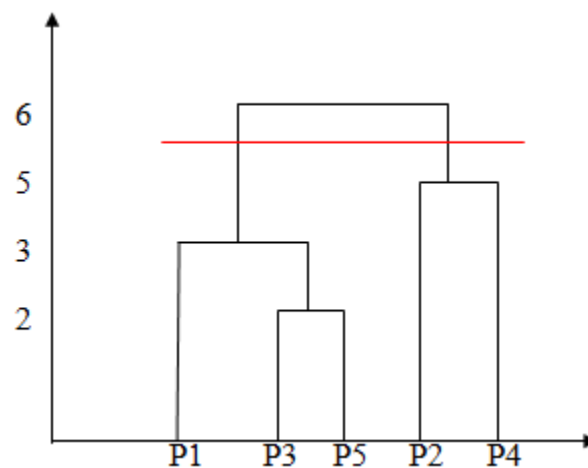


Figure 26: dendrogram final results of example

Dendrogram is referred to visualize the hierarchical clustering. It is a like tree such as the chart records the sequences of merges or splits.

Advantages:

- Agglomerative clustering is very easy to execute.
- Agglomerative clustering results in hierarchy base. For example, structure is more instructive and easy to understand while K-Means is un-structured set of flat clusters.
- Easy to make a decision the numbers of clusters.
- It is related to meaningful taxonomy.

Disadvantages:

- Time complexity is there, hierarchical clustering is not good for large dataset.
- In this clustering, it is not likely to undo the previous step.
- There is no mathematical purpose for this clustering.
- It is extremely conscious of outliers.
- Initial parameters are strongly effect on the end of the results.

5.2.2. K-Means Clustering

K-means clustering is the easiest and very famous unsupervised machine learning algorithms. Normally unsupervised machine learning algorithms make interfering from corpus by using input vectors not including referring to known or labelled results.

K-means clustering algorithm is also called iterative algorithm. That try to make the partition the corpus into k predefined individual non overlapping clusters where all data-points belong to only from single group. K-means clustering makes the inter cluster data-points and also keeps the clusters as different as possible. It allots the data-points to cluster that are the sum of the squared distance between the data-points and centroid of cluster is as the minimum. Mean that all the data-points that belong to that cluster. Less variation inside the clusters, more uniform data-points are in the same cluster.

In K-means clustering the following steps are considered:

- Specify the no. of clusters K .
- By first shuffling the dataset, initialize the centroids and randomly choose the K data-points for the centroids with no replacement.
- Lastly, keep repeating until no changes to centroids.

Then we need to calculate the sum squared distance between the data-points and their all centroids. And allocate each data-point to centroids cluster (closest one). In order to calculate the centroids for clusters, take average of all data-points that are belonging to each cluster.

In order solve the problems K-means approach are followed is known as the expectation maximization.

The following E- step is used to allocate the data-points to closest cluster while the M- step is used to calculate the centroids of each cluster. Describe the mathematical principle to solve it theoretically.

The objective function is:

$$J = \sum_{i=1}^m \sum_{k=1}^K w_{ik} \|x^i - \mu_k\|^2$$

Where, for cluster k , $w_{ik} = 1$ for data-point “ x^i ”. Else $w_{ik} = 0$ and μ_k is the centroid of “ x^i ” cluster.

First of all, we need to minimize the \hat{J} with respect to μ_k and w_{ik} fixed. So, we separate \hat{J} with respect to w_{ik} first and then update the cluster is called E- step. Then we separate \hat{J} with respect to μ_k and recalculate the centroids after the cluster the allocations from last step is called M- step.

So, E- step is:

$$\begin{aligned} \frac{\partial J}{\partial w_{ik}} &= \sum_{i=1}^m \sum_{k=1}^K \|x^i - \mu_k\|^2 \\ \Rightarrow w_{ik} &= \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Assign the closest cluster by its sum of squared distance from cluster centroids.

Then M- step is:

$$\begin{aligned} \frac{\partial J}{\partial \mu_k} &= 2 \sum_{i=1}^m w_{ik} (x^i - \mu_k) = 0 \\ \Rightarrow \mu_k &= \frac{\sum_{i=1}^m w_{ik} x^i}{\sum_{i=1}^m w_{ik}} \end{aligned}$$

This re-calculates the centroids of each cluster to reflect the new cluster allocations.

Actually we don't know what the real number of clusters in K- means clustering. There are various methods to choose the right K. basically these methods are based on the domain knowledge. Among them elbow method is one of the best method used to find the correct number of clusters.

- **Elbow method:**

Elbow method is very useful tool that represent the results graphically in order to estimate the correct number of cluster K for given dataset. If number of cluster K increases, the within- cluster-sum-of-squares (WCSS) distortion will be decrease. Because of this sample would be closer to the centroids they are allocated to.

Calculate the sum of the distances from the cluster centroids is called WCSS.

$$WCSS = \sum_{i \in n} (X_i - Y_i)^2$$

Where, “Yi” is centroids for “Xi”.

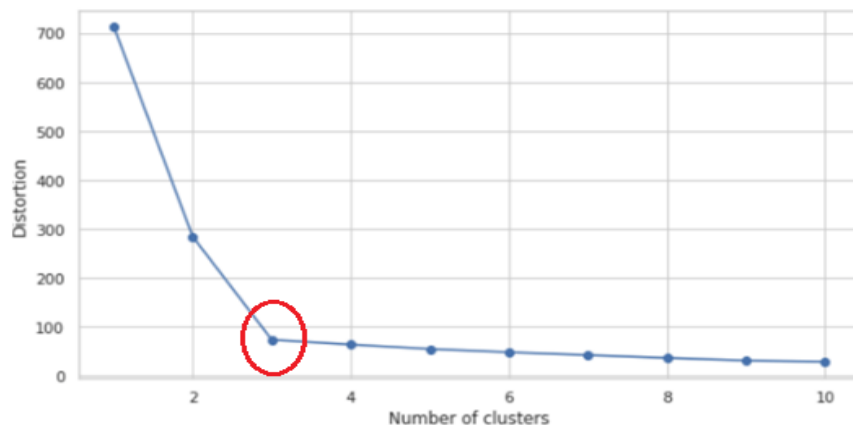


Figure 27: Sample for elbow method

It seems in the plot the elbow shape is located at K=3 that indicate the number of cluster 3 is the best choice for the dataset.

- **Silhouette co-efficient**

Silhouette co-efficient is another method to compute the number of right cluster K. it can be apply to any type of clustering. Silhouette co-efficient is calculated by using the following formula.

$$s_i = \frac{b - a}{\max(a, b)}$$

Where, “b” is the min. average distance to every other observation from every other cluster. And “a” is the average distance to all other observations inside the same cluster.

For the same data WCSS is comparable for the different clustering K. And that numbers is not comparable with the different clustering on different data. So, is does not have fix threshold.

SC with k=2-20

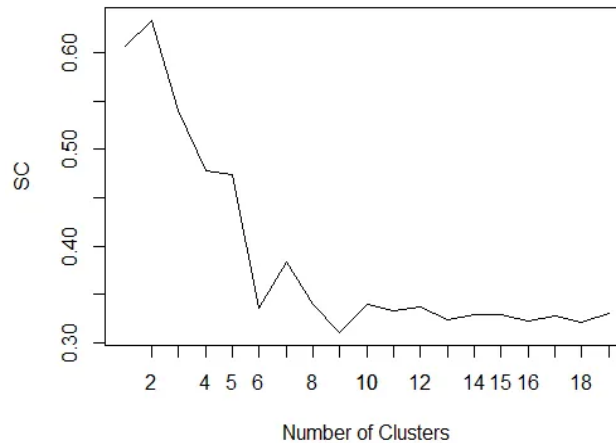


Figure 28: sample for silhouette score

In the plotting, it seems highest co-efficient of 0.63 with number of cluster 3. And second highest co-efficient of 0.60 with number of cluster 2. Silhouette co-efficient drops and stays low for high number of clusters.

Advantages:

- K- means clustering is easy to interpret
- It is comparatively fast and scalable for big datasets
- In k- means clustering, may competent to choose the positions of preliminary centroids in smart way which helps to speed up the convergence

Disadvantages:

- In K-means clustering doesn't presume how many number of clusters exists in the dataset.
- K-means clustering has sensitive to outliers
- In this clustering, no. of samples increases at each step, because K-means clustering algorithms access all data-points and computes distances.
- K-means clustering is good when cluster shape has spherical because it tries to make spherical shape across the cluster centroids. So, if we have complex shape of clusters, k-means algorithms would not work as expected and may get poor clustering results.
- We need to apply elbow method in order to choose the correct number of the clusters. So, sometime it has possibility to make error in selection of right number of clusters.

5.2.3. DBSCAN clustering

Dbscan clustering is called “density based spatial clustering of applications with noise”. It is a very famous unsupervised learning method that used in the machine learning algorithms. Dbscan is the method which is used in machine learning to divide the high density cluster from low density clusters. Density based clustering is good in the data that have high density observations versus the low density with observations.

Hierarchical clustering and K-means clustering methods are highly efficient with normal shaped of clusters. While density based clustering method is more efficient with arbitrary shaped clusters and detecting outliers.

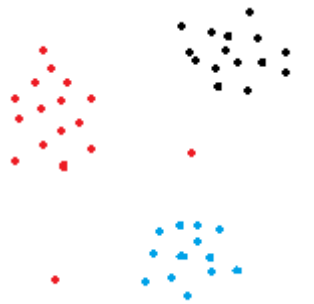


Figure 29: Sample of Dbscan clustering

The data-points in above figure are grouped in arbitrary shaped with outliers. Dbscan clustering algorithms are very efficient to find high density areas and with outliers. Outlier's detections are very importance in some task such as anomaly detection.

The density based clustering groups the points together which are close to each other that are based on the measured distance, considering Euclidean distance, and minimum no. of points. So, we can say density based cluster the point belongs to cluster when the point close to several points from that cluster.

Dbscan algorithm able to find out arbitrary shaped clusters and with clusters noise which is called outliers that is low density region.

Parameters:

The following figure shows the dbscan clustering example with minimum points is 4.

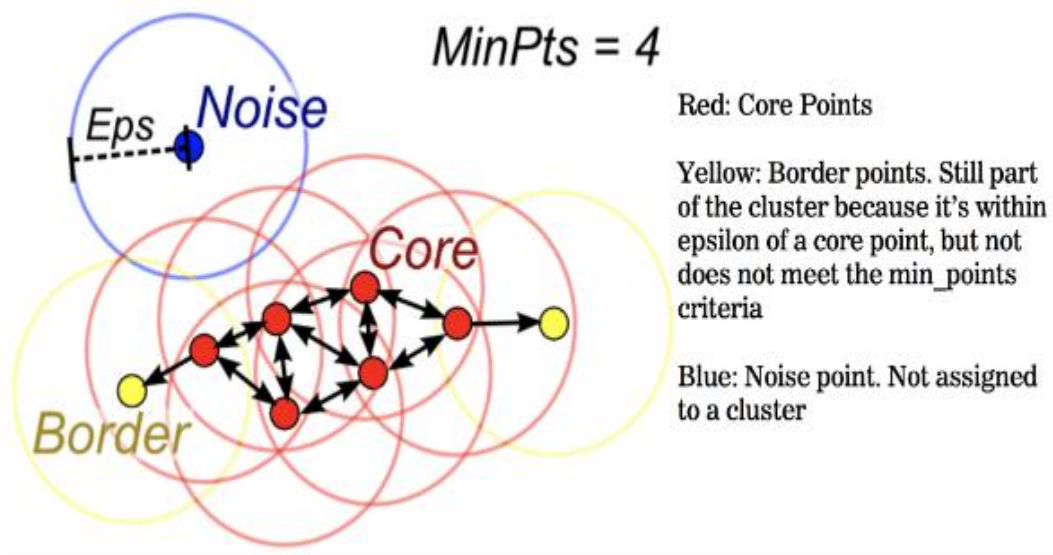


Figure 30: Parameters for Dbscan clustering

Basically, two parameters are required in Dbscan algorithms.

- **Eps – epsilon:** it indicates the close points should be to each other to be regard as a part of a cluster. It signifies that if the distance between two points is lower or equal to this eps-epsilon, these points are considered neighbors.
- **MinPts – minPoints:** it indicates the minimum no. of points to form a density region. For instance, if we set the minpts parameters as 4, then we need at least 4 points in density region. It is the min. no. of data-points to define cluster.

The points are classified as outliers, core points and border points based on the above two parameters:

-
- **Core point:** we can say point is core if there is at-least minPts no. of points with point itself in its surrounding region with eps radius.
 - **Border point:** we can say the point is border points if it is accessible from core points and less than minpoints no. of points within its enclosing area.
 - **Outlier:** we can say if point is not a core point and it does not accessible from any core points.

Parameter estimation:

The estimation of above parameter is a problem for every text mining work. To decide right parameters we need to know how they are used and have some basic previous information about the dataset that will be used.

Epsilon: If the epsilon value preferred is too small, a large part of the data wouldn't be clustered. It would consider outliers for the reason that doesn't satisfy the number of points to create a dense area. In contrast, if the value selected is too high, clusters will combine and the majority of objects will be in the same cluster. The epsilon should be selected based on the distance of the dataset. But commonly small epsilon values are preferable.

MinPts: Mini. minPoints can be obtained from a number of dimensions in the dataset, as $\text{minPts} \geq D+1$. Larger values are generally better for dataset with noise and will form more considerable clusters. A min. value for the minPts must be 3, although larger the dataset, the larger the minPts value that must be selected.

By applying Dbscan algorithms, which is able to find the high dense regions and divide them from low dense regions.

Advantages:

- Dbscan is the great clustering method in order to separate high density clusters versus low density clusters in given datasets.
- It is the good with outliers handling.
- When clusters shape is arbitration, Dbscan perform well with them. And it doesn't require indicating the cluster number beforehand.

Disadvantages:

- When deal with varying density cluster, Dbscan doesn't perform well as expected. While it is good at dividing with high density clusters form low dense clusters. And it struggles with similar density clusters.
- When high dimension data, sometimes Dbscan struggle with them.
- Sometimes, or in some cases, in order to determine the right eps is not easy and it requires some basic knowledge about it.

6. Inputs for clustering analysis and their final outcomes

In this section we will talk about K- mean clustering algorithms, hierarchical clustering algorithms and Dbscan clustering algorithms that are implemented in python with sklearn (Scikit-learn). And then finally we will observe their outcomes.

Before applying the K-means clustering, hierarchical clustering and Dbscan clustering, use some special tools such as normalizing the data, PCA in order to get accurate and speed up the results.

- **Standardize the data:** basically PCA is affected by the scale so first of all need to scale features in the data before applying PCA. So, in this case use the “StandardScalar” that helps to standardize the data into unit scale which is required for the most favourable performance of M/c learning algorithms. Unit scale means 0 =mean and 1= variance.
- **Normalizing the data:** normalization refers to re-scaling the real valued numeric points into 0 and 1 range. This lets our model to improved weights and in turn leads to accurate model.
- If we want to speed up the appropriate of M/c learning algorithms by changing the optimization of algorithm. A common way to speed up of M/c learning algorithms by using PCA. PCA is “principal component analysis”. If M/c leaning algorithms is too slow due to high input dimension so in order to speed up it using the PCA tool is a reasonable choice. It is the generally common application of PCA, and in addition it used for data visualization. PCA uses to reduce the 4 dimensional data into 2/3 dimensional data so that can be plot and understand the better data.

6.1. K- means clustering

6.1.1. K-means clustering algorithms for positive reviews

In the discussion above in K-means clustering we have learnt how K-means clustering works does. Let's apply it to our dataset (dataset considered as the Tf-idf output results) using the K-means class from sklearn.

Inputs for libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
```

Code snippet 21: import libraries for K-mean clutering

Input for standardscaler and PCA

```
BH = np.array(tfidf_pos)

scaler = StandardScaler()
scaled_df = scaler.fit_transform(BH)

# Normalizing the Data
normalized_df = normalize(scaled_df)

# Converting the numpy array into a pandas DataFrame
normalized_df = pd.DataFrame(normalized_df)

# Reducing the dimensions of the data
pca = PCA(n_components = 2)
X_pos = pca.fit_transform(normalized_df)
X_pos = pd.DataFrame(X_pos)
X_pos.columns = ['P1', 'P2']
X_pos.head()
```

Code snippet 22: standardscaler and PCA for positive reviews

Output

	P1	P2
0	-0.094483	-0.016729
1	-0.071109	-0.017278
2	0.316745	-0.242617
3	-0.089692	-0.008264
4	-0.056207	-0.0
....

The elbow method is to identify the number of the clusters k where the distortion starts to decrease which is very easy to understand if we make a plot for different numbers of value K .

Inputs

```
sse = {}
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(X_pos)
    sse[k] = kmeans.inertia_ # Inertia: Sum of distances of
    samples to their closest cluster center
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.ylabel("SSE")
plt.show()
```

Code snippet 23: input for elbow method for positive reviews

Output

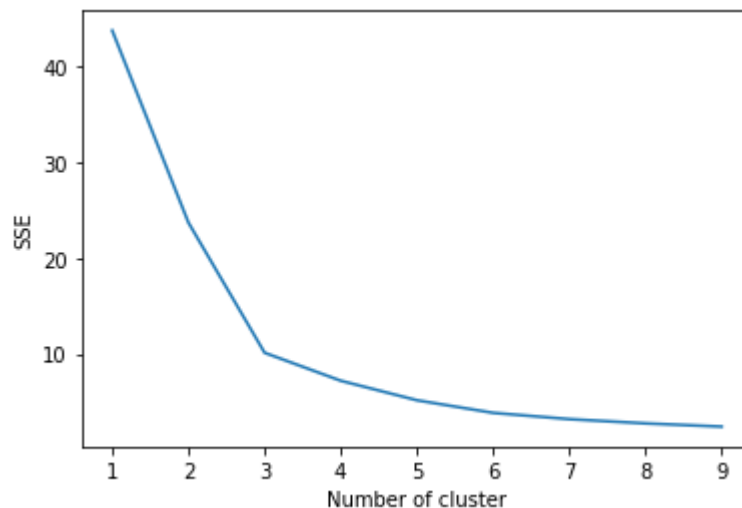


Figure 31: Output of elbow method of K-means for positive reviews

According the above elbow shape, it seems the correct number of clusters K will be 3.

Then, apply for the silhouette score. It is another method used to choose the right number of cluster K .

Input

```
silhouette_scores = []

for n_cluster in range(2, 8):
    silhouette_scores.append(
        silhouette_score(X_pos, KMeans(n_clusters =
n_cluster).fit_predict(X_pos)))

# Plotting a bar graph to compare the results
k = [2, 3, 4, 5, 6, 7]
plt.bar(k, silhouette_scores)
plt.xlabel('Number of clusters', fontsize = 10)
plt.ylabel('Silhouette Score', fontsize = 10)
plt.show()
```

Code snippet 24: silhouette score of K-means for positive reviews

Output plot

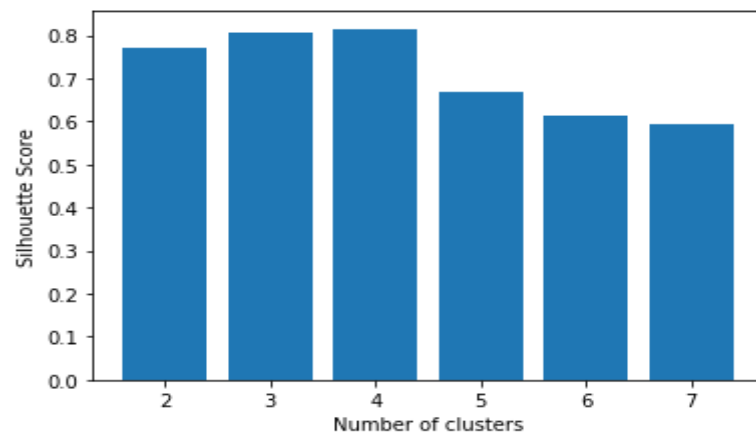


Figure 32: Output of silhouette score of K-means for positive reviews

After observing the elbow plot and silhouette score, the correct number of clusters K will be 3 for K-means clustering in given data. Now apply for K-means clustering using number of clusters K = 3 in order to visualization of scatter plot.

Inputs

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(X_pos)
a=kmeans.labels_

finalDf= pd.concat([X_pos, pd.DataFrame({'cluster':a})], axis=1)

plt.figure(figsize=(9,7))
ax = sns.scatterplot(x="P1", y="P2", hue="cluster",
data=finalDf,palette=['red', 'blue', 'green'])
plt.show()
```

Code snippet 25: Scatter plot of k-means for positive reviews

Output

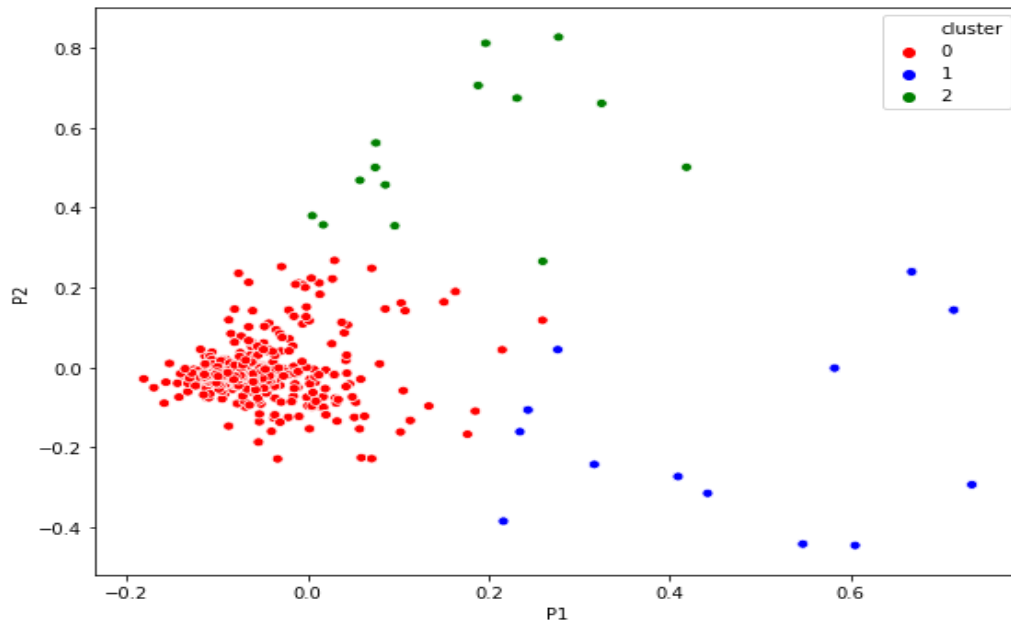


Figure 33: Output of scatter plot of K-means for positive reviews

Scatter plots are used to understand the relation among variables and by the help of dots to represent the relation among them. In order to draw, the scatter plot by using the scatter method in matplotlib library in python.

Then, assign the number of cluster to dedicated reviewtext by using the following codes.

Input

```
clusterspos=pd.concat([finalpositive,pd.DataFrame({'cluster':a})],  
axis=1)  
clusterspos.head()
```

Code snippet 26: Assign the no. of cluster to reviewtext for positive reviews

Output

	Reviewtext	cluster
0	I give them 5 stars.Well deserved cheaper than...	0
1	Great wipers, they are much more effective and...	0
2	Great price, great product	1
3	I went to autozone for replacement wipers and ...	0
4	Great product with a reasonable price. Used fo...	0

Then, we can observe that reviewtext number 0 belong to cluster number 0 while reviewtext number 2 is belonging to cluster number 1. This is just example of first 5 reviews.

Now we will look how many reviewtexts belong to dedicated cluster number from 0 to 2.

Input

```
clusterspos['cluster'].value_counts()
```

Code snippet 27: count no. reviews per cluster for positive reviews for K-means

Outputs

Cluter no.	reviewtext
0	607
1	58
2	29

Observation: we can see, in given data there are 607 reviews are belonging to cluster number 0, 58 reviews belong to cluster 1 and 28 reviews belong to cluster 2.

Then, we want to extract the most important words in the particular cluster. In order to know what things customers like about the product and service. By using the following code we can extract important words from particular cluster.

First, consider the cluster number 0. There are total 607 reviewtext among them extract the top important words.

Inputs

```
import re, nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
sw_list = ['!', '.', ',', 'wiper', 'wipers', 'bosch', 'icon']
stop_words = stopwords.words('english')
stop_words.extend(sw_list)
wordnet_lemmatizer = WordNetLemmatizer()
from sklearn.feature_extraction.text import CountVectorizer
```

```
pos = []
for i in clusters0['Reviewtext']:
    pletters = re.sub('[^a-zA-Z]', ' ', str(i))
    ptokens = nltk.word_tokenize(pletters)
    plowercase = [l.lower() for l in ptokens]
    filtered_presult = list(filter(lambda l: l not in
stop_words, plowercase))
    plemmas = [wordnet_lemmatizer.lemmatize(t) for t in
filtered_presult]
    pos.append(' '.join(lemmas))

cv = CountVectorizer(analyzer = 'word', stop_words =
'english', max_features = 10, ngram_range=(2,2))
most_positive_words = cv.fit_transform(pos)
temp1_counts = most_positive_words.sum(axis=0)
temp1_words = cv.vocabulary_
print('the most important words in Positive Reviews:')
print('-----')
temp1_words
```

Code snippet 28: extract imp. Words in positive reviews for k-means

Outputs for cluster number 0

```
the most important words in Positive Reviews:
-----
{'easy install': 1,
 'work better': 8,
 'great product': 4,
 'highly recommend': 6,
 'work great': 9,
 'best blade': 0,
 'great quality': 5,
 'great blade': 3,
 'fit perfectly': 2,
 'install work': 7}
```

We can observe the most of customers give their opinion such as product is highly recommended, great product, fit perfectly etc.. about the product. In this case we considered 5 rating score for the product.

By considering the subsequent number of cluster, there are 58 reviewtext in cluster number 1. By applying same coding steps can extract the top important words from the cluster number 1.

Outputs for cluster number 1

```
the most important words in Positive Reviews:
-----
{'great price': 5,
 'price great': 8,
 'great product': 6,
 'work great': 9,
 'great easy': 4,
 'easy install': 2,
 'great blade': 3,
 'blade great': 1,
 'install work': 7,
 'blade easy': 0}
```

By considering the subsequent number of cluster, there are 29 reviewtext in cluster number 2. By applying same coding steps can extract the top important words from the cluster number 2.

Output for cluster number 2

```
the most important words in Positive Reviews:
-----
{'best blade': 0,
 'best bought': 1,
 'blade car': 4,
 'best quality': 2,
 'excellent blade': 5,
 'far best': 6,
 'blade bought': 3,
 'simply best': 7}
```

The achieved above outputs for cluster number 0,1 and 2 are only for positive reviews that belong to review rating score 5. Because review rating score 3 and 4, there are combination of some positive texts and negative texts. So, sometimes outputs are confusing if we consider the review rating score 3 and 4.

6.1.2. K-means clustering algorithms for negative reviews

In order to get the following outcomes, apply the same coding steps what did in the positive reviews.

In the negative we are considering only review rating score 1 and 2. So, we will get filter of negative reviews without including positive terms.

In this section, we will skip some coding steps and directly jump to the final results because some procedures are same as positive reviews.

First of all, directly go to the elbow method.

Output

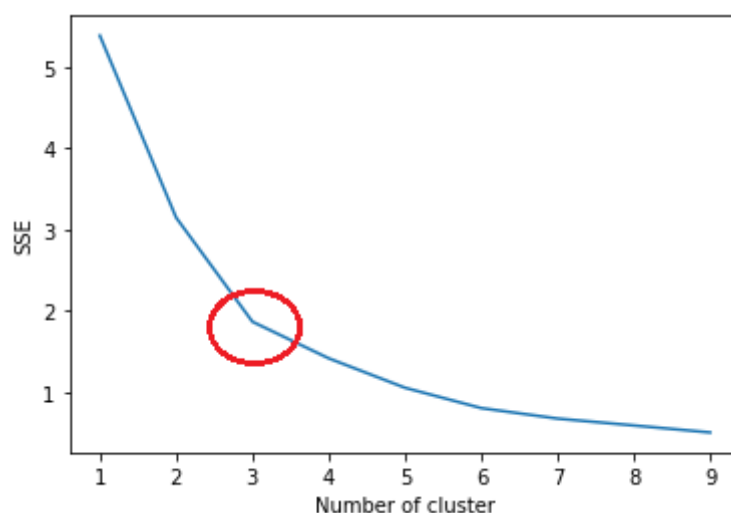


Figure 34: Output of elbow method of K-means for negative reviews

Based on the above graph, we can observe that there is 3 number of clusters for given dataset.

Then, apply for the silhouette score. It is another method used to choose the right number of cluster K. This method is directly implemented in python by using correct libraries with sklearn. Silhouette score method most of time gives same results as elbow method but some time little variation.

Output

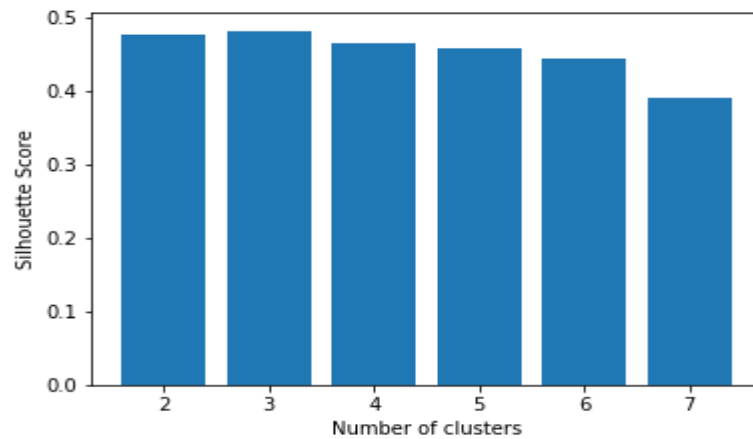


Figure 35: Output of silhouette score of K-means for negative reviews

Observation:

After observing the elbow plot and silhouette score, the correct number of clusters K is 3 for K-means clustering in given data.

Then, apply for K-means clustering using number of clusters $K = 3$ in order to visualization of scatter plot.

Output

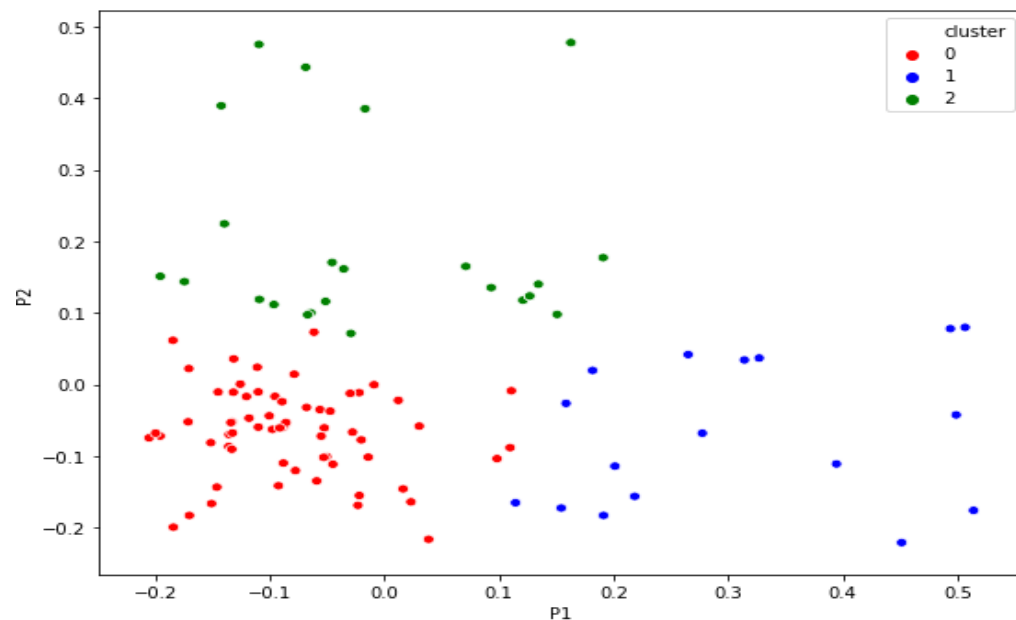


Figure 36: Scatter plot of K-means for negative reviews

In the next steps, in order to know which cluster includes the how many number of reviewtexts. After apply the following coding steps we get the correct output.

Input

```
clustersneg=pd.concat([finalnegative,
pd.DataFrame({'cluster':a})], axis=1)
clustersneg.head()
clustersneg['cluster'].value counts()
```

Code snippet 29: Count no. of reviewtexts per cluster for negative review for K-means

Outputs

Cluster no.	Reviewtext
0	64
2	26
1	17

Observation: we can see, in given data there are 64 reviews are belonging to cluster number 0, 17 reviews belong to cluster 1 and 26 reviews belong to cluster 2.

Now, we want to extract the most important words in the particular cluster. In order to know what things customers dislike about the product and service.

First, consider the cluster number 0. There are total 64 reviewtext among them extract the top important words.

Outputs for cluster number 0

```
the most important words in Negative Reviews:
-----
{'work bad': 13,
 'could return': 1,
 'waste money': 14,
 'even cleaning': 3,
 'passenger side': 7,
 'driver side': 2,
 'waste buy': 11,
 'volkswagen beetle': 9,
 'windshield rain': 12,
 'every time': 4,
 'fit vw': 5,
 'jeep work': 6,
 'vw beetle': 10,
 'blade fit': 0,
 'slide lock': 8}
```

Outputs for cluster number 1

```
the most important words in Negative Reviews:
-----
{'fit audi': 2,
 'stated would': 13,
 'bad fit': 10,
 'fit audia': 3,
 'despite amazon': 1,
 'amazon saying': 0,
 'fit camaro': 4,
 'fit oem': 8,
 'fit correctly': 6,
 'fit car': 5,
 'saying dear': 12,
 'fitting guide': 9,
 'guide vw': 11,
 'univesal fit': 14,
 'fit jetta': 7}
```

Outputs for cluster number 2

```
the most important words in Negative Reviews:
-----
{'leaf streak': 2,
 'streak windshield': 10,
 'really worst': 6,
 'well month': 12,
 'see driving': 7,
 'really bad': 5,
 'work horrible': 13,
 'week started': 11,
 'lasted six': 1,
 'six month': 8,
 'month split': 3,
 'split dissatisfied': 9,
 'dissatisfied paying': 0,
 'paying much': 4}
```

Notes:

In the negative reviewtext analysis, output of clusters some words like positive meaning because of during cleaning process removed stopwords that do not carrying the useful information. These words such as they, does, do, will, not, don't, are not, is not, etc. for example, customer would say “wiper blade doesn't work perfectly” but output result is “wiper blade work perfectly. In this case, remove “doesn't”.

So, keeping in mind outputs belong to something negative.

6.2. Hierarchical clustering

6.2.1. Agglomerative clustering for positive reviews

In the discussion above, in hierarchical clustering we have learnt how agglomerative clustering works. Let's apply it to our dataset (dataset considered as the Tf-idf output results) using the agglomerative class with sklearn in python.

Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
import scipy.cluster.hierarchy as shc
```

Code snippet 30: Import libraries for agglo. Clustering

Input for standardscaler and PCA

```
# Standardize data
scaler = StandardScaler()
scaled_df = scaler.fit_transform(tfidf_pos)

# Normalizing the Data
normalized_df = normalize(scaled_df)

# Converting the numpy array into a pandas DataFrame
normalized_df = pd.DataFrame(normalized_df)

# Reducing the dimensions of the data
pca = PCA(n_components = 2)
X_poshei = pca.fit_transform(normalized_df)
X_poshei = pd.DataFrame(X_poshei)
X_poshei.columns = ['P1', 'P2']

X_poshei.head(5)
```

Code snippet 31: standardscaler and PCA for agglo. Clustering

Output

	P1	P2
0	-0.094483	-0.016728
1	-0.071109	-0.017279
2	0.316744	-0.242598
3	-0.089692	-0.008264
4	-0.056207	-0.043939

In hierarchical clustering, there is a new step to find the optimal number of clusters is called the dendrogram. In agglomerative clustering we are not using elbow method to define the number of clusters.

By default we are using the ward method for analysis the agglomerative clustering in given data. Because ward method is actually reduce the variance in each cluster.

Below diagram is dendrogram

Inputs

```
plt.figure(figsize=(10, 7))
plt.title('Visualising the data')
Dendrogram = shc.dendrogram((shc.linkage(X_poshei, method='ward')))
```

Code snippet 32: dendrogram for positive reviews

Output

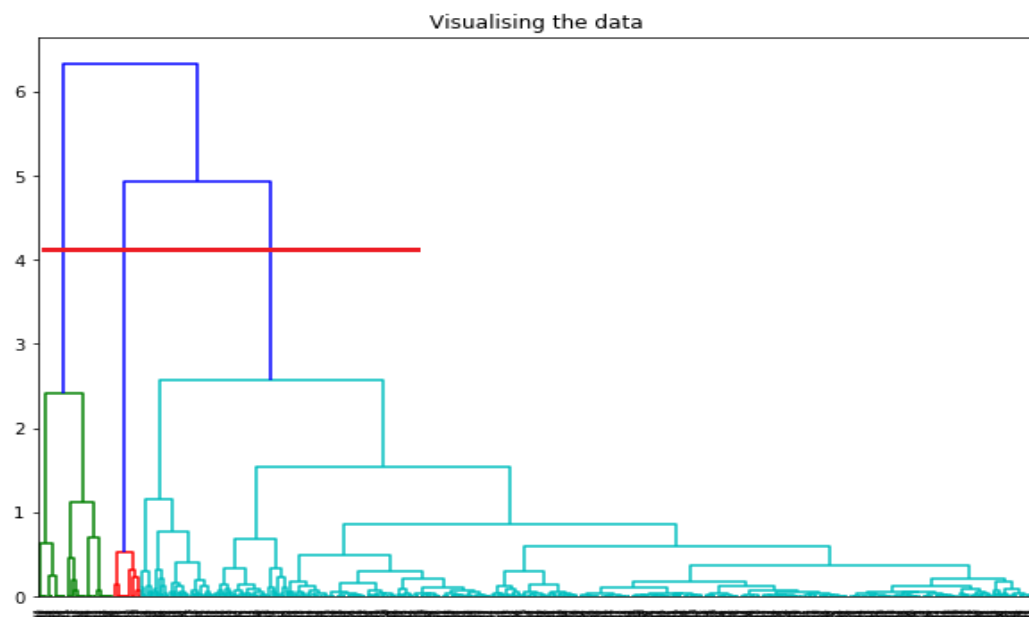


Figure 37: Output of dendrogram of agglo. For positive reviews

By looking at the dendrogram and considering the above theory, the best possible number of clusters is equivalent to the number of vertical lines passing through the horizontal line. So, we can say the optimal number of clusters will be 3.

Then, apply for the silhouette score. It is another method used to choose the right number of cluster K.

Input

```
silhouette_scores = []

for n_cluster in range(2, 8):
    silhouette_scores.append(
        silhouette_score(X_poshei,
            AgglomerativeClustering(n_clusters =
n_cluster).fit_predict(X_poshei)))

# Plotting a bar graph to compare the results
k = [2, 3, 4, 5, 6, 7]
plt.bar(k, silhouette_scores)
plt.xlabel('Number of clusters', fontsize = 10)
plt.ylabel('Silhouette Score', fontsize = 10)
plt.show()
```

Code snippet 33: silhouette score for agglo. For positive reviews

Output

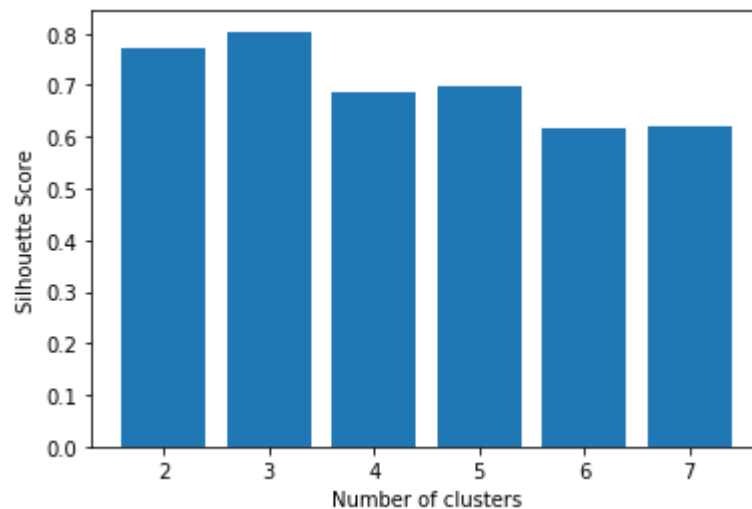


Figure 38: Output of silhouette score of agglo. For positive reviews

After observing the dendrogram diagram and silhouette score, the correct number of clusters K will be 3 for agglomerative clustering in given data.

Then, apply for agglomerative clustering using the number of clusters = 3 in order to visualization of scatter plot.

Input

```
agg=AgglomerativeClustering(n_clusters=3,affinity='euclidean',li
nkage='ward')
agg.fit(X_poshei)
b=agg.labels_

finalDf_agg = pd.concat([X_poshei, pd.DataFrame({'cluster':b})],
axis = 1)

plt.figure(figsize=(10,7))
ax = sns.scatterplot(x="P1", y="P2", hue="cluster",
data=finalDf_agg,palette=['red', 'blue', 'green'])
plt.show()
```

Code snippet 34: Scatter plot for agglo. For positive reviews

Output

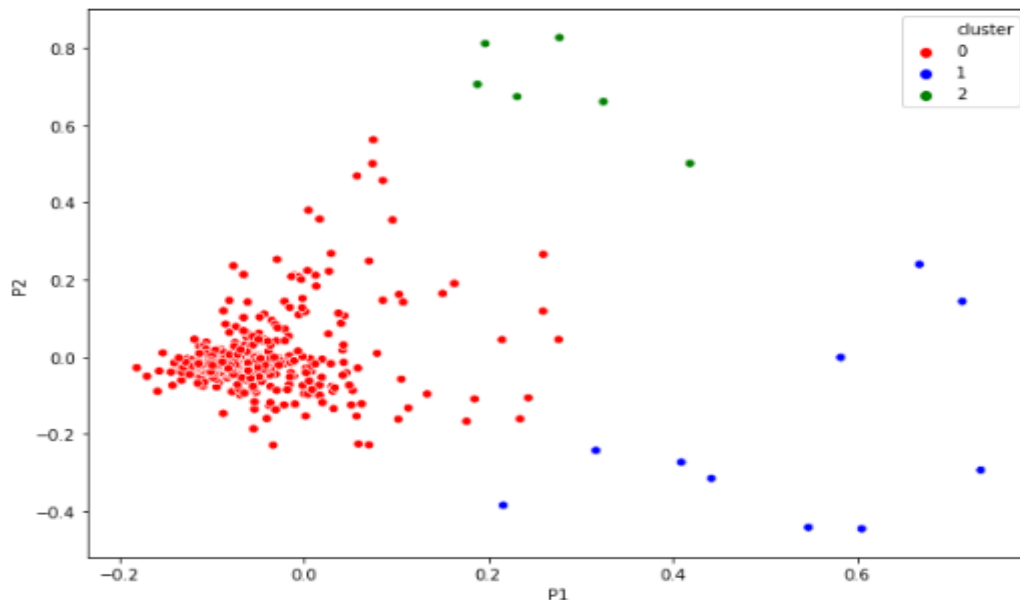


Figure 39: Output of scatter plot of agglo. For positive reviews

Then, assign the number of cluster to dedicated reviewtext by using the following codes. Means, we want to know how many reviewtexts belong to dedicated cluster number from 0 to 2.

Inputs

```
clusterspos = pd.concat([finalpositive,
pd.DataFrame({'cluster':b})], axis = 1)

clusterspos['cluster'].value_counts()
```

Code snippet 35: count no. of reviewtexts per cluster for agglo. For positive reviews

Outputs

Cluster no.	Reviewtext
0	622
1	53
2	19

Observation: we can see, in given data there are 622 reviews are belonging to cluster number 0, 53 reviews belong to cluster 1 and 19 reviews belong to cluster 2.

Then, we want to extract the most important words in the particular cluster. In order to know what things customers like about the product and service. By using the following code we can extract important words from particular cluster.

First, consider the cluster number 0. There are total 622 reviewtext among them extract the top important words. Then consider cluster 1. And then subsequently cluster 2. In order to get the most important words from each cluster we will apply same coding steps what did in K-means clustering, so, we will skip the input coding steps.

Outputs for cluster number 0

```
the most important words in Positive Reviews:
-----
{'easy install': 2,
 'perfect blade': 12,
 'better work': 7,
 'work better': 13,
 'great product': 5,
 'windshield wiper': 11,
 'highly recommend': 8,
 'best wiper': 0,
 'work perfect': 14,
 'great quality': 6,
 'excellent wiper': 3,
 'fit perfectly': 4,
 'blade best': 1,
 'perfect fit': 10,
 'install work': 9}
```

Outputs for cluster number 1

```
the most important words in Positive Reviews:
-----
{'great price': 4,
 'price great': 9,
 'great product': 5,
 'work great': 13,
 'great wiper': 6,
 'wiper blade': 12,
 'great blade': 2,
 'great great': 3,
 'blade great': 0,
 'easy install': 1,
 'install work': 8,
 'product great': 10,
 'product work': 11,
 'great work': 7}
```

Outputs for cluster number 2

```
the most important words in Positive Reviews:
-----
{'best blade': 0,
 'best wiper': 1,
 'best blade': 3,
 'good wiper': 2}
```

Observations:

We observed in cluster number 0, the most important word such as easy install, perfect fit, highly recommended, etc. given by the most of customers. While observing cluster number 1 is related to word “great”, and cluster 2 is related to word “best”. For given input just considered only positive reviews in which rating score greater than 4.

6.2.2. Agglomerative clustering for negative reviews

In the negative we are considering only review rating score 1 and 2. So, we will get filter of negative reviews without including positive terms.

In this section, we will skip some coding steps and directly jump to the final results because some procedures are same as agglomerative clustering for positive reviews.

First of all, visualise the data by dendrogram and define the number of clusters.

Output

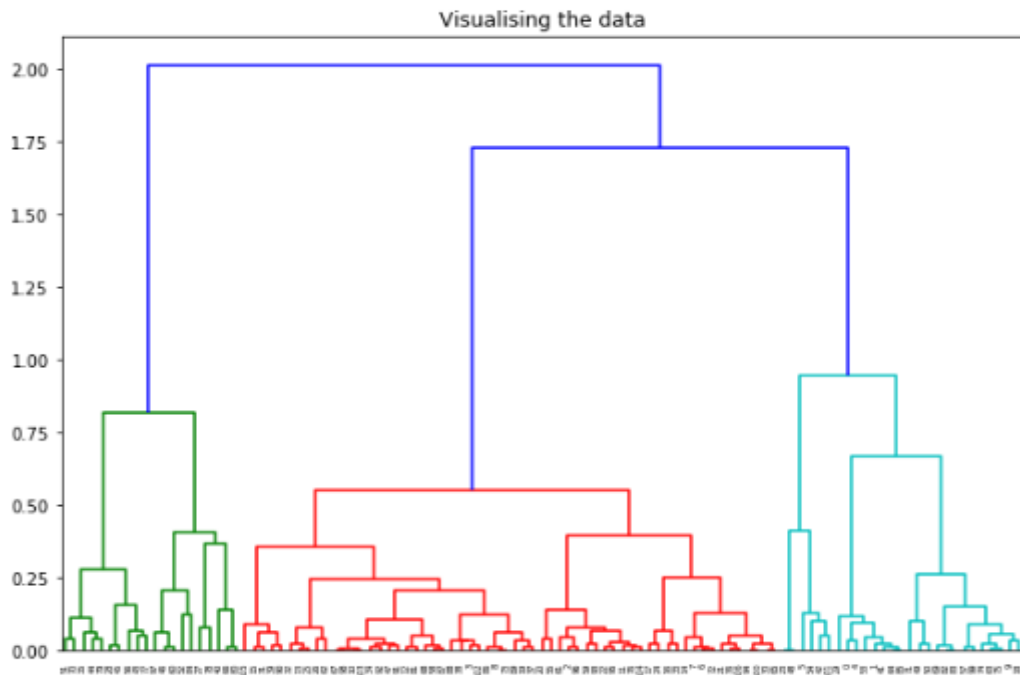


Figure 40: Output of dendrogram of agglo. For negative reviews

By looking at the dendrogram and considering the above theory in the section of hierarchical clustering, the best possible number of clusters is equivalent to the number of vertical lines passing through the horizontal line. So, we can say the optimal number of clusters will be 3. And we can also observe the dendrogram shapes of negative reviews are different from the dendrogram shape of positive reviews.

Then, also apply for silhouette score to know how many number of clusters for given data input.

Output

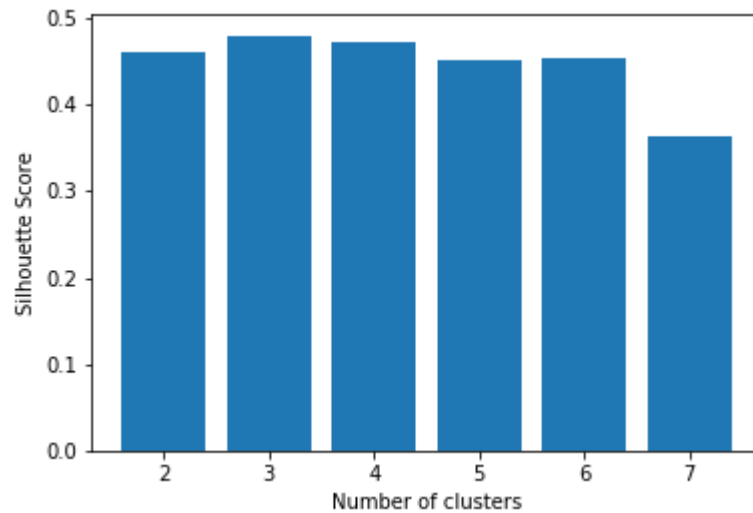


Figure 41: Output of silhouette score of agglo. For negative reviews

By observing the silhouette score and dendrogram, the number of clusters will be 3.

Then, apply for agglomerative clustering using the number of clusters = 3 in order to visualization of scatter plot.

Output

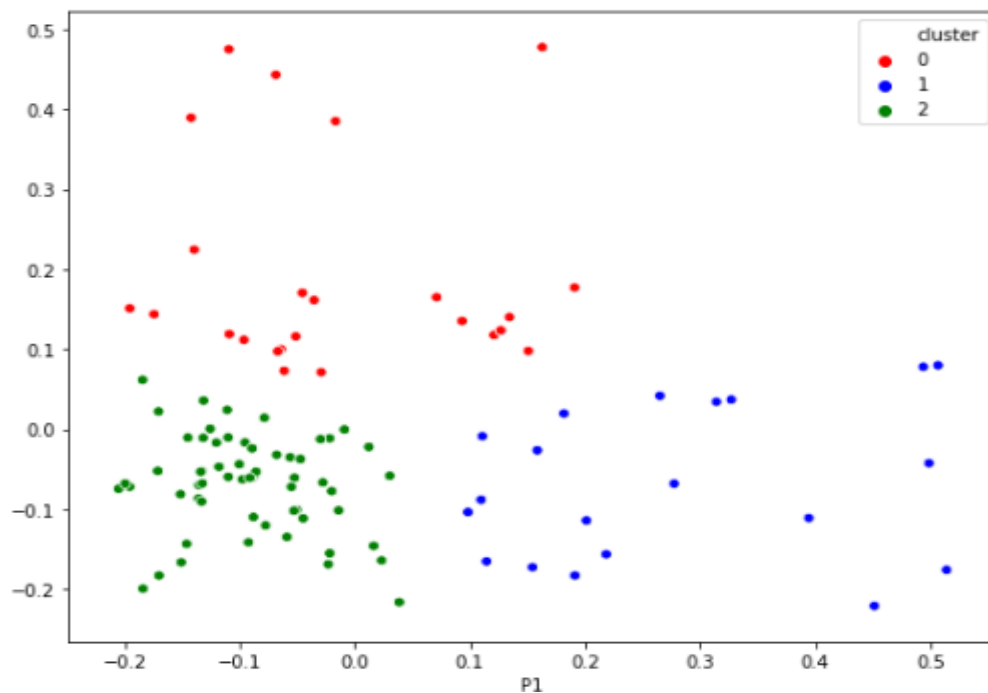


Figure 42: Output of scatter plot of agglo. For negative reviews

Then, assign the number of cluster to dedicated reviewtext by using the same codes. Means, we want to know how many reviewtexts belong to dedicated cluster number from 0 to 2.

Outputs

Cluster no.	Reviewtext
0	27
1	20
2	60

Observation: we can see, in given data there are 27 reviews are belonging to cluster number 0, 20 reviews belong to cluster 1 and 60 reviews belong to cluster 2.

Then, we want to extract the most important words in the particular cluster. In order to know what things customers dislike about the product and service. By using the same code we can extract important words from particular cluster.

First, consider the cluster number 0. There are total 27 reviewtext among them extract the top important words. Then consider cluster 1. And then subsequently cluster 2. In order to get the most important words from each cluster we will apply same coding steps what did in K-means clustering, so, we will skip the input coding steps.

Outputs for cluster number 0

```
the most important words in Negative Reviews:
-----
{'leaf streak': 4,
 'streak windshield': 10,
 'well month': 12,
 'see driving': 7,
 'worked well': 14,
 'bad see': 0,
 'streaking leaving': 11,
 'film windshield': 2,
 'lasted six': 3,
 'six month': 8,
 'month split': 5,
 'work perfectly': 13,
 'split dissatisfied': 9,
 'dissatisfied paying': 1,
 'paying much': 6}
```

Outputs for cluster number 1

the most important words **in** Negative Reviews:

```
-----  
{ 'fit audi': 2,  
  'stated would': 11,  
  'fit vw': 7,  
  'good fit': 9,  
  'fit audia': 3,  
  'fit car': 5,  
  'despite amazon': 1,  
  'amazon saying': 0,  
  'vw passat': 12,  
  'wiper arm': 14,  
  'fit camaro': 4,  
  'fit correctly': 6,  
  'fitting guide': 8,  
  'guide vw': 10,  
  'vw passats': 13}
```

Outputs for cluster number 2

the most important words **in** Negative Reviews:

```
-----  
{ 'jeep work': 9,  
  'could return': 1,  
  'worth money': 14,  
  'even cleaning': 4,  
  'driver side': 3,  
  'waste money': 12,  
  'fit volkswagen': 7,  
  'volkswagen beetle': 11,  
  'windshield rain': 13,  
  'every time': 5,  
  'curvature windshield': 2,  
  'blade fit': 0,  
  'slide lock': 10,  
  'fit vw': 8,  
  'fit guide': 6}
```

6.3. DBSCAN clustering

6.3.1. DBSCAN clustering for positive reviews

In the discussion above, in Dbscan clustering theory we have learnt how Dbscan clustering works does. Let's apply it to our dataset (dataset considered as the Tf-idf output results) using the Dbscan class with sklearn in python.

Import libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA
```

Code snippet 36: import libraries for Dbscan

Input for standardscalar and PCA

```
# Scaling the data to bring all the attributes to a comparable
level
scaler = StandardScaler()
X_scaled = scaler.fit_transform(tfidf_pos)

# Normalizing the data so that
# the data approximately follows a Gaussian distribution
X_normalized = normalize(X_scaled)

# Converting the numpy array into a pandas DataFrame
X_normalized = pd.DataFrame(X_normalized)

# Reducing the dimensions of the data
pca = PCA(n_components = 2)
X_posdb = pca.fit_transform(normalized_df)
X_posdb = pd.DataFrame(X_posdb)
X_posdb.columns = ['P1', 'P2']
X_posdb.head(2)
```

Code snippet 37: standardscaler and PCA for Dbscan

Basically, two parameters are required in Dbscan algorithms.

Eps – epsilon: it is used to define the neighbor across data-point. If we choose epsilon value too small, the larger parts are considered as the outliers in the given dataset. If we choose the large value, the most part of the data point in the same clusters. For given data we chose the eps value 0.259.

MinPts – minPoints: if we have larger data, must choose the larger value. The lowest value should be chosen at least 3. Then apply the Dbscan clustering algorithms by using the following coding steps.

Inputs

```
db = DBSCAN(eps=0.259, min_samples=3).fit(X_posdb)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

finalDf_db = pd.concat([X_posdb,
pd.DataFrame({'cluster':labels})], axis = 1)
finalDf_db.head()
```

```
# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)
print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)

# Plot result
import matplotlib.pyplot as plt

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]
```

```
class_member_mask = (labels == k)

xy = X_posdb[class_member_mask & core_samples_mask]
plt.plot(xy['P1'], xy['P2'], 'o',
markerfacecolor=tuple(col),
        markeredgecolor='k', markersize=6)

xy = X_posdb[class_member_mask & ~core_samples_mask]
plt.plot(xy['P1'], xy['P2'], 'o',
markerfacecolor=tuple(col),
        markeredgecolor='k', markersize=6)

plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```

Code snippet 38: scatter plot of Dbscan for positive reviews

Output

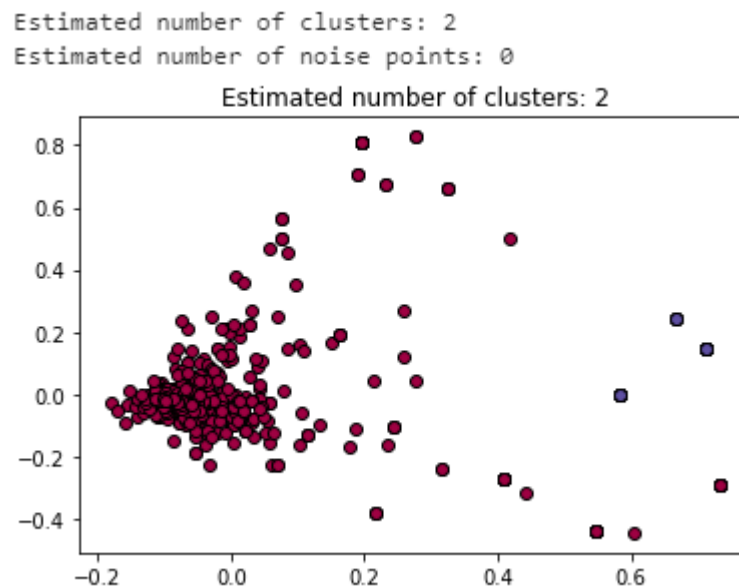


Figure 43: Output of Dbscan for positive reviews

By observing the output, the estimated number of clusters is 2 and estimated number of noise points is 0 for given data input.

Then, assign the number of cluster to dedicated reviewtext by using the following codes. Means, we want to know how many reviewtexts belong to dedicated cluster number from 0 and 1.

Inputs

```
clustersposdb = pd.concat([finalpositive,
pd.DataFrame({'cluster':labels})], axis = 1)
clustersposdb.head()

clustersposdb['cluster'].value_counts()
```

Code snippet 39: Count no. reviewtexts per cluster for Dbscan for positive reviews

Outputs

Cluster no.	Reviewtext
0	674
1	20

Observation: we can see, in given data there are 674 reviews are belonging to cluster number 0, 20 reviews belong to cluster 1.

Then, we want to extract the most important words in the particular cluster. In order to know what things customers like about the product and service. By using the same code we can extract important words from particular cluster.

Outputs for cluster number 0

```
the most important words in Positive Reviews:
-----
{'easy install': 5,
'great price': 9,
'great product': 10,
'work better': 11,
'highly recommend': 14,
'best blade': 0,
'work great': 17,
'great quality': 13,
'best market': 1,
'streak free': 12,
'blade used': 4,
'clear windshield': 3,
'good price': 8,
'fit perfectly': 7,
'best used': 2,
'excellent product': 6,
'perfect fit': 16,
'install work': 15}
```

Outputs for cluster number 1

```
the most important words in Positive Reviews:
-----
{'great blade': 1, 'great wiper': 0}
```

6.3.2. DBSCAN clustering for negative reviews

By choosing the eps value is 0.259 and MinPts value is 3, same as the Dbscan clustering for positive reviews. We can see in the following scatter plot, there is estimated number of clustering is 1 and estimated number of noise points is 0.

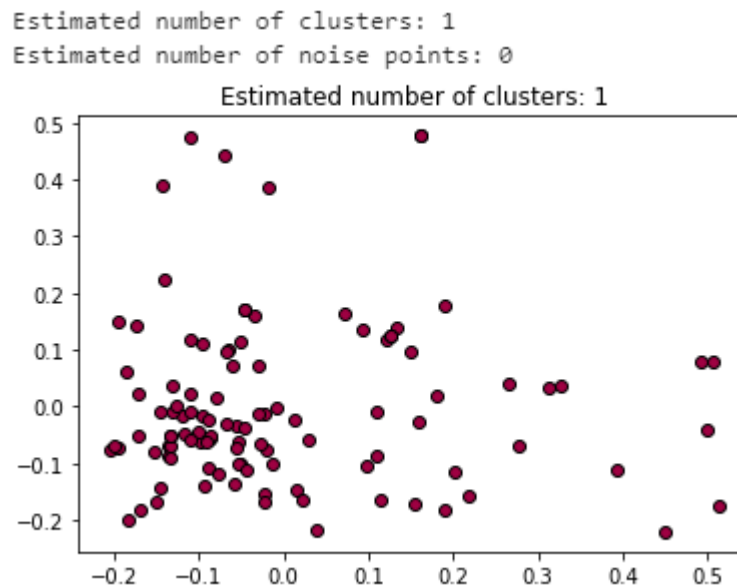


Figure 44: Output of Dbscan for negative clustering

Then, assign the number of cluster to dedicated reviewtext by using same code what did in positive reviews. Means, we want to know how many reviewtexts belong to dedicated cluster number 1.

Output

Cluster no.	Reviewtext
0	107

Observation: we can see in above output, in given data there are 107 reviews are belonging to cluster number 0

Then, we want to extract the most important words in the particular cluster for negative reviews. In order to know what things customers dislike about the product and service. By using the same code we did before, we can extract important words from particular cluster.

Outputs for cluster number 0

```
the most important words in Negative Reviews:
-----
{'leave streak': 9,
 'could return': 1,
 'worth money': 14,
 'even cleaning': 3,
 'driver side': 2,
 'waste money': 12,
 'windshield rain': 13,
 'every time': 4,
 'fit audi': 5,
 'fit vw': 8,
 'blade fit': 0,
 'slide lock': 10,
 'streak windshield': 11,
 'fit car': 6,
 'fit correctly': 7}
```

Notes:

In the negative reviewtext analysis, output of clusters some words like positive meaning because of during the cleaning process removed stopwords that do not carrying the useful information. These words such as they, does, do, will, not, don't, are not, is not, etc. for example, customer would say “wiper blade doesn't work perfectly” but output result is “wiper blade work perfectly. In this case, remove “doesn't” by cleaning process. So, keeping in mind outputs belong to something negative.

7.Final analysis of results:

After observation of previous sections, the output of 3 different types of clustering methods such as K-Means clustering, hierarchical clustering and DBSCAN clustering. Then we can create the final clusters by removing the duplicate words from various outputs of these clusters. We cannot depend only on one type of clustering method because each clustering technique has own advantages and disadvantages.

So, we have two types of reviews, Positive reviews and negative reviews

For the positive reviews, considered the review rating score is 5 which means customers are totally satisfied with the product or services. So, they gave positive outcome about the product. Among them we extracted the top words what they like about the product.

For the negative reviews, review rating score are considered only 1 and 2 which means customers are totally dissatisfied with the product or services. So, they wrote the negative reviews about product, among them we extracted top words what they do not like about the product or services.

Final cluster words for positive reviews	Final cluster words for negative reviews
Easy install, work better, best blade, it perfectly, great quality, excellent blade, perfect blade, highly recommend, great product, clear windshield	Passenger side, leave streak, work horrible, lasted month, paying much, fit Volkswagenbeetle, jeep fit, slide lock, driver side, could return, fit audi, bad fit, fit camaro, universal fit, waste money, streak on windshield, wiper arm, curvature windshield, fitting guide, fit jetta, bad fitting,

By using, the final cluster words for positive reviews and negative reviews we can define the product strengths and defects.

Product strengths:

- The Bosch ICON 21A wiper blade is easy to install, fit perfectly on windshield, over all material quality of product is great, excellent blade, clear cleaning the windshield, and last but not least many customers recommended this product.

Product defects:

- The main defect of this wiper blade is leaving streaking on windshield.
- Amazon and Bosch say “Bosch ICON 21A” wiper blade is universal blade; it can fit on all cars. But it cannot fit perfectly on Audi, Camaro, Jetta, Volkswagen beetle and jeep. Users of these cars facing the fitting problems.
- Windshield blade lasted long only month.
- This wiper blade is expensive compare to other blades.
- Not suitable for curvature windshield.
- Slide lock issue is some vehicle.
- Last but not least, in some vehicle leave streak or not clean on driver side or passenger side.

After seeing the defects of product, the manufacturer and seller should improve the product that user faces.

Note: consumers give their opinion on product through quality of products, feeling about products, price, brands, and get experiences after uses as well as they compare with other brand products.

8. Conclusion

In this thesis work, apply text mining techniques to understand the customer opinion on the product or services. By using the above text mining and clustering methods anyone can gain the insights about what are most of consumers are saying about the product or services. So, help of this information about customer reviews such as features and customer problems can be used to do improvement on product and various services. When these above methods are applied to reviews of one product, the most important words to users can be retrieved from both positive and negative reviews. In order to extract the most important words from the large dataset, the different task are implemented in the system that are the time consuming such as the various types of clustering algorithms, Tf-idf vectorizer. So, I would say Python libraries are perfect choice to use for the application of text mining on product reviews.

These thesis clustering outcomes are satisfied partially; however, the performance would be improved with the help of cost effective algorithms. But outputs of TF-IDF method are highly satisfied with the accuracy. For future applications, improving this idea will be useful. And this idea further focuses on research activities that may recognize the proper clustering algorithms for retrieve the data from large dataset.

References:

- [1] Ji Fu Kung, James Lin, Y B Hsu, “handle unstructured data using text mining techniques in semiconductor manufacturing”
- [2] Laura kassner, christoph GrKgera, Bernhard Mitschanga, Engelbert Westk “Product life cycle analysis on structured and unstructured data” from graduate school advanced manufacturing engineering university Stuttgart, Germany.
- [3] Johannes Zenkert, Christian weber, Andre Klahold and Madjid Fathi “Knowledge based production document analysis integrated with text mininig” from institute of knowledge based systems and knowledge management, siegen, Germany.
- [4] “Tf- idf analysis”, [online] Wikipedia.org
- [5] L.Jack and Y.D. Tsai “using text mining of amazon reviews to explore product highlights and issues” from Intel corporation, CA, USA.
- [6] Vaibav Joshi, Satyanarayan Iyengar, and Amritha venkataramana “big data analysis on amazon product reviews”
- [7] Hyun jeong Ban, Hyun woo Joung “text mining approach to understand seat comfort of airlile passengers through online review” from Kyung Sung University.
- [8] En Gir Kim, Se Hak Chun “Analyzin online car reviews are using text mining” from department of business administration, seoul national university of science and technology, Korea.
- [9] Charu C. Aggarwal, Cheng Xiang Zhai, “survey of text clustering algorithms”
- [10] Dhilip Subramanian “text mining in python” online: medium.com
- [11] C.Chen and M. Song “text mining with unstructured text” from Springer International Publishing AG 2017
- [12] “K-means clustering”, [online] Wikipedia.org
- [13] Cory Maklin “tf idf python example” [online] towardsdatascience.com
- [14] “hierarchical clustering analysis” [online] Wikipedia.org
- [15] Chaitanya reddy Patlolla “understand the concept of hierarchical clustering” [online] towardsdatascience.com
- [16] “dbscan clustering analysis” [online] Wikipedia.org
- [17] Hafsa jabeen “stemming and lemmatization in python” [online] datacamp.com
- [18] Python text analysis library [online] nltk 3.5 documentation from nltk.com