



**UNIVERSITY
OF MALAYA**

**Data Mining
WQD7005**

Milestone 5 : Final submission

Name : Sidratul Muntaha

Matric : Wqd180079/17199116

Name: Nur Nuraini Afiqah Binti Rohanip

Matric : Wqd180115/17198049

Project Title : Text Mining on customer feedbacks of different laptop brands

Github links repository for this project:

<https://github.com/AfiqahRoha/DataMining>

<https://github.com/SidratulMuntaha/DataMining>

TABLE OF CONTENT

CHAPTER 1 INTRODUCTION	2
CHAPTER 2 METHODOLOGY	3
CHAPTER 3 DATA PRE-PROCESSING AND DESCRIPTION	5
CHAPTER 4 MODELLING AND RESULTS	11
CHAPTER 5 WEB APPLICATION	14
CHAPTER 6 MOBILE APPLICATION	18
APPENDIX	22

CHAPTER 1 : INTRODUCTION

Text mining is also referred to as text analytics is an artificial intelligence (AI) technology that uses natural language processing (NLP) to transform the free (unstructured) text in documents and databases into normalized, structured data suitable for analysis or to drive machine learning (ML) algorithms.

Natural Language Understanding helps machines “read” text by simulating the human ability to understand a natural language such as English, Spanish or Chinese.

As a technology, natural language processing has come of age over the past ten years, with products such as Siri, Alexa and Google's voice search. Sophisticated text mining applications are facilitating the area of medical research, risk management, customer care, fraud detection and contextual advertising.

Social Media like facebook, twitter ,reddit, instagram are one of the most influential mediums of communication and these platforms have an immense effect on marketing and advertising of products. The objective of this project is to extract data from social media platforms about various laptop brands and gather the feedback of the social media users. social media works as a powerful media in today's world for advertising, and the brands gaining better reviews from users most of the time gain the leading position. We have chosen twitter and reddit platforms for crawling data.

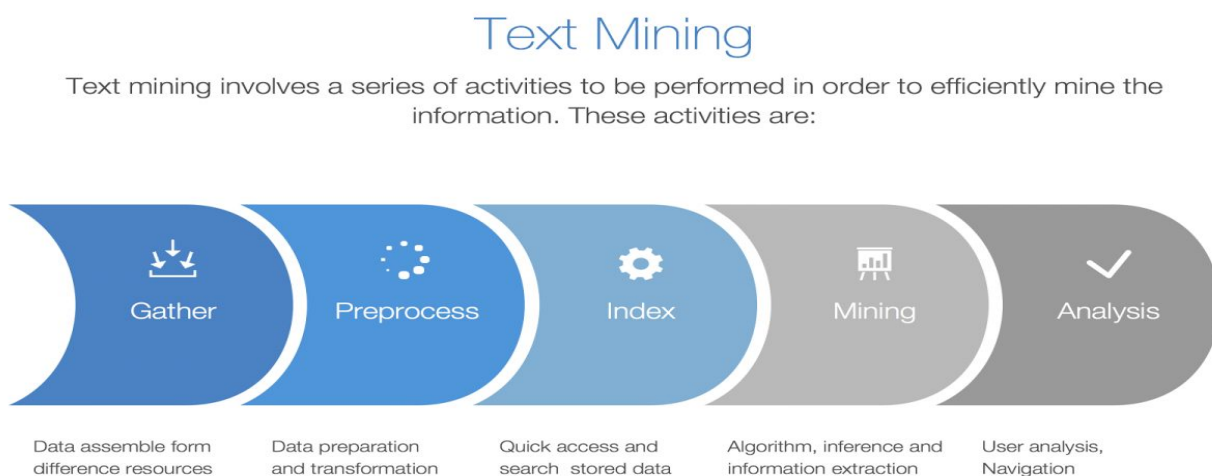


Figure 1: The steps of text mining

CHAPTER 2: METHODOLOGY

Python was used to do web crawling to extract the real time threads.. It is applied to gather unstructured data like, tweets, from twitter and comments and questions from reddit as well. Including web crawling, the sentiment analysis for later steps and the modelling for building classifiers have been carried out in Python to achieve the objectives of this study. On the other hand, SAS Enterprise Miner on-demand has been deployed by using text mining to perform data interpretation and some preprocessing steps. Machine learning algorithms used in this project were Random forest classifier, Gaussian Naive Bayes, Decision Tree, SVM, XgBoost, Logistic Regression etc. Model comparison has been done to determine the better model among them which is able to classify more accurately and predict most of the sentiments precisely.

Workflow :

ML1: Web crawling by python from twitter and reddit, and preprocessing/data cleaning

ML2: Storing Data in Data Warehouse or lake (Hive) : virtual machine and hadoop was in use.

ML3: Accessing Data from Database :switched to Mysql and accessed data from Mysql via python.

ML4: Interpretation of data: Sentiment analysis for reforming data and SAS enterprise miner for text mining and interpreting.

ML5: Web Application using Flask and mobile application using Kivy.

Data Collection:

Dataset File Name	Data Sources	Data Descriptions	File Size
reddit_encoded.csv	Reddit	<ol style="list-style-type: none">1. TITLE -The title of the subreddit post.2. SCORE -The number of upvotes minus the number of downvotes.3. ID -Unique value ID.4. SUBREDDIT -Name of the subreddits.5. URL -Url of the subreddit post.6. NUM_COMMENTS -Number of comments under the post.7. BODY -The body of the subreddit post.8. CREATED -Encoded timestamp.9. TIMESTAMP -Decoded timestamp.	4.50 MB
Comments_encoded.csv	Reddit	<ol style="list-style-type: none">1. ID2. COMMENTS -Comments under each subreddit post in reddit_encoded.csv.3. SUBREDDIT	4.01 MB
Twitter_encoded.csv	Twitter	<ol style="list-style-type: none">1. TIMESTAMP -Date and time of the tweet.2. TEXT -Tweets about laptops.3. MODEL -Laptop model tag in each tweet.	4.62 MB

Table 1: Details on crawled datasets.

CHAPTER 3: DATA PRE-PROCESSING AND DESCRIPTION

Prior to storing data into a data warehouse, we had done some pre-processing to clean the data. Applying the knowledge we had been given from last semester in Programming for Data Science, we use R programming language for this part. First and foremost, the data is converted into a corpus file for easier cleaning process. We also use the `tm_map` function from the `tm` library to map the regular expressions that we had declared inside a `gsub` function and return null if found. In order to maintain the dataset as close to real text as possible, we only remove urls, reddit tags or hashtags, symbols, certain stopwords, and specific punctuations. The reason we did not do thorough cleaning on the data is because when labelling the data using VADER sentiment analysis, we want to obtain more accurate results since VADER is able to calculate the depth of a sentiment based on punctuations, word elongations, and capital letters and returns it in the form of positive, neutral and negative scores varying from 0 to 1. After that, VADER will compute a compound score which is the result from summing all the lexicon ratings and normalize the sum into a scale from -1 (most extreme negative) to +1 (most extreme positive).

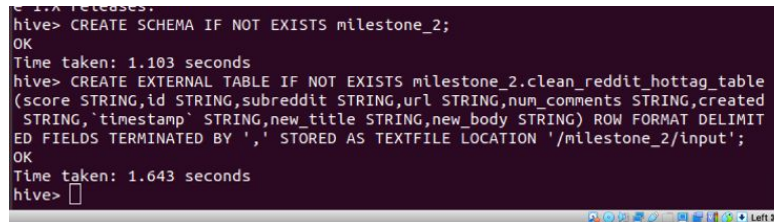
Dataset after cleaning:

Below is the screenshot after the twitter data has been cleaned:

TIMESTAMP	MODEL	NEW_TEXT	TIME_OF_DAY
28/5/2020 19:30	HP Spectre	90s fun for the kids HP HPSpectre laptops computers gaming pattern 90s 80s fun Product accessories vinyl skins decals wraps	Evening
28/5/2020 4:38	HP Spectre	Save 200 on the HP Spectre x360 Now 1349.99! Offer valid 5/21-5/31! Click Exclusive offer Code Brought to you by promo PRO	Night
24/5/2020 1:58	HP Spectre	Are you ready for a Change? Why not try an alternative to Amazon? HP Spectre x360 13.3" Intel Core™ i7 2 in 1 – 1 TB SSD BWT	Night
22/5/2020 22:38	HP Spectre	Amazon is under investigation in France declared economic terrorist in India. Why not try an alternative? Sign up today ! HP S	Evening
20/5/2020 11:40	HP Spectre	Brand new sealed HP SPECTRE X360 i7 512gb 8gb ram TOUCHSCREEN fingerprint reader for Gh7000! 0249799705 WhatsApp 020	Morning
19/5/2020 16:11	HP Spectre	Amazon declared economic terrorist in India. Why not try an alternative? HP Spectre x360 13.3" Intel Core™ i7 2 in 1 – 1 TB SS	Afternoon
19/5/2020 5:30	HP Spectre	HPSpectre x360 15ってやつ。授業でホワイトボード機能使えたらええなと思ったらタッチパネルかなと思った	Night
18/5/2020 21:27	HP Spectre	task without limit on your PC. Advanced Machines Gadgets TuesdayThoughts AmagetStore AmagetOnline Tuesday	Evening
18/5/2020 4:30	HP Spectre	hpspectre	Night
15/5/2020 11:08	HP Spectre	HPspectre 2月に頼んでようやっとようやっと届いてくれましたん ノ ㇏ ノ お待ち申し上げておりましたん T	Morning
13/5/2020 17:35	HP Spectre	Is there any public info regarding the release date of new Spectre x360 in Greece? Otherwise maybe a solution...by letting uk	Afternoon
9/5/2020 19:30	HP Spectre	It's like you can touch it HP HPSpectre laptops computers product accessories skins vinylskins vinyl vinylstickers MightySkins	Evening
8/5/2020 11:51	HP Spectre	Que vous soyez à la recherche d'une batterie qui dure toute la journée ou d'un PC convertible voici le HP Spectre le plus puis	Morning
28/4/2020 19:08	HP Spectre	HPSpectre x360 13-aw0003nn Intel Core™ i7-1065G7 16 GB LPDDR4-3200 SDRAM onboard 512 GB PCIe NVMe™ M.2 SSD 13 3" W	Evening
28/4/2020 11:35	HP Spectre	HP Spectre Folio Review hpspectre hpspectre360 hpspectrex360 hpspectre13	Morning
28/4/2020 8:38	HP Spectre	New video hp hpspectre x2 detachable	Morning
23/4/2020 2:30	HP Spectre	Can't wait to get mine this Friday hpSpectre	Night
22/4/2020 20:50	HP Spectre	I have used a lot of laptops but this is IT for me. hpspectre EarthDayAtHome EarthDay	Evening
14/4/2020 20:55	HP Spectre	Learning how to use autodesk sketchbook. I might have found a drawing app I can use on my laptop. If I can make some cool c	Evening

Figure 2: Cleaned Twitter data

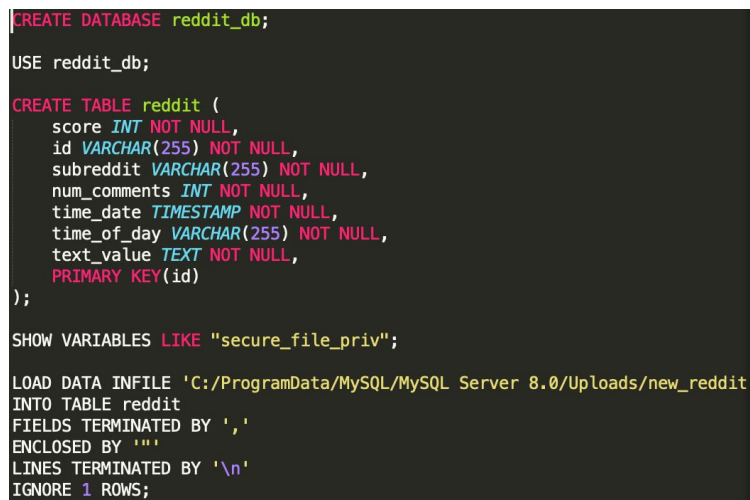
Storing data into data warehouse and accessing it from Python:



```
hive> CREATE SCHEMA IF NOT EXISTS milestone_2;
OK
Time taken: 1.103 seconds
hive> CREATE EXTERNAL TABLE IF NOT EXISTS milestone_2.clean_reddit_hottag_table
(score STRING,id STRING,subreddit STRING,url STRING,num_comments STRING,created
STRING,'timestamp' STRING,new_title STRING,new_body STRING) ROW FORMAT DELIMIT
ED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/milestone_2/input';
OK
Time taken: 1.643 seconds
hive>
```

Figure 3: Storing datasets into Hive

After Hive installation through Ubuntu Virtual Machine, firstly we created a new database using CREATE DATABASE IF NOT EXISTS. Then, import the table from our local system using CREATE TABLE IF NOT EXISTS query and gave the names of all the variables from our data as the name of the table columns along with STORED AS TEXTFILE LOCATION query with the link of the directory inside the HDFS that we want the data to be stored.



```
CREATE DATABASE reddit_db;

USE reddit_db;

CREATE TABLE reddit (
  score INT NOT NULL,
  id VARCHAR(255) NOT NULL,
  subreddit VARCHAR(255) NOT NULL,
  num_comments INT NOT NULL,
  time_date TIMESTAMP NOT NULL,
  time_of_day VARCHAR(255) NOT NULL,
  text_value TEXT NOT NULL,
  PRIMARY KEY(id)
);

SHOW VARIABLES LIKE "secure_file_priv";

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/new_reddit.'
INTO TABLE reddit
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Figure 4: Switched to MySQL and accessed it from python

To import the data into MySQL database, first we created a new database and named it as reddit_db based on the name of the data. In order to load the data into the database, we need to create an empty table first and declare the variable names or column names that we want MySQL to save as in the system. Set the variables data type also when creating a new table. Important thing to note, the ID variable should be set as the primary key for MySQL to detect it as a unique value ID. The next query in this image, SHOW VARIABLES is used for the purpose of getting the source folder(s) that MySQL can detect to load the data from it. MySQL cannot detect just any directory so we need to know which directory that it can load data. After that, load the data using LOAD DATA INFILE and put the directory of the data and into the empty table that we had created previously.

```

from sqlalchemy import create_engine
import pymysql
import pandas as pd

db_connection_str = 'mysql+pymysql://your_username:your_password@your_hostname/reddit_db'
db_connection = create_engine(db_connection_str)

reddit_df = pd.read_sql('SELECT * FROM reddit', con=db_connection)

```

Figure 5: Accessing MySQL from Python

After storing the data, we need to be able to access it from Python to proceed to the next part. Python has a library to help us with this called sqlalchemy which helps to connect Python to any sql database including MySQL. Before connecting it, we need to know the connection variables so again, run the query SHOW VARIABLES to know your username, password, and hostname. Then with the help of Python library pymysql, create the connection link along with the database name that you wish to connect. Using this, establish the connection with create_engine function from sqlalchemy and run read_sql function from pandas with queries that you want, in this case SELECT. After that, simply save the pandas dataframe as a csv file to have it in your local system.

Labelling the data using VADER sentiment analysis:

The initial and most important step in this part is to import the SentimentIntensityAnalyzer library from VADER and save it as an object for simpler use in the next part. Then, run the SentimentIntensityAnalyzer with polarity_scores function to obtain negative, neutral, positive and lastly compound scores for each row of our text column data. The outcome will appear like this below screenshot:

	neg	neu	pos	compound	reddit_text
0	0.079	0.812	0.108	0.7184	Ignore Private Messages Suggesting Laptops The...
1	0.030	0.833	0.137	0.9408	SuggestALaptop Stress Test Project! Submit Vie...
2	0.015	0.818	0.166	0.9883	I need a laptop for gaming coding video editin...
3	0.051	0.785	0.164	0.9577	So I can't buy this yet? If I can buy it where...
4	0.000	0.905	0.095	0.8753	Alienware M15 vs Lenovo Y740? Any thoughts on ...

Figure 6: VADER sentiment analysis outcome

Then, based on the compound scores, categorize the sentiments as positive, neutral or negative by using a conditional statement that is if compound score is more than a certain value, return a categorical value representing the overall sentiment of the text. The conditional statement must be according to the VADER rule of sentiment classification. After this step, we obtained a new column named as sentiments which contains value 1 for positive, 0 for neutral and -1 for negative. Now we can proceed with modelling as we have obtained the target or independent variable from this labelling process.

Word Cloud:

A word cloud is a collection, or cluster, of words depicted in different sizes. The bigger and bolder the word appears, the more often it's mentioned within a given text and the more important it is. Also known as tag clouds or text clouds, these are ideal ways to pull out the most pertinent parts of textual data, from blog posts to databases. Below are the three word clouds we generated from the last changes in the database. After the sentiment feature is added the word clouds are visualized to see what kinds of words appear in the specific datasets.

The reddit dataset contains mostly queries and inquiries about different laptops and sometimes information or complaints. According to the outcome of the word cloud, it seems that the data contains mostly overviews about the properties of laptops. The second word cloud from the reddit comment threads dataset is similar quite to the former word cloud. It highlights some of the gaming references like the ssd memory or Ryzen brand etc, which denotes that gaming properties or configurations are a factor to be noticed.



Figure 9: Twitter word cloud

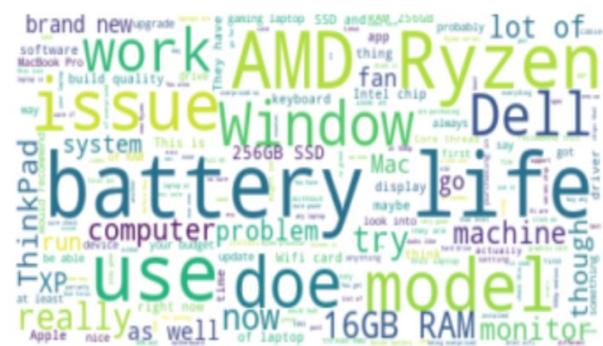


Figure 9: reddit Comments wordcloud



Figure 9: reddit word cloud

Interpretation in SAS:

We applied our dataset to SAS Enterprise Miner Workstation 14.1 to do pre-processing, observe the concept link between words and training the data using Decision Tree and Neural Network.

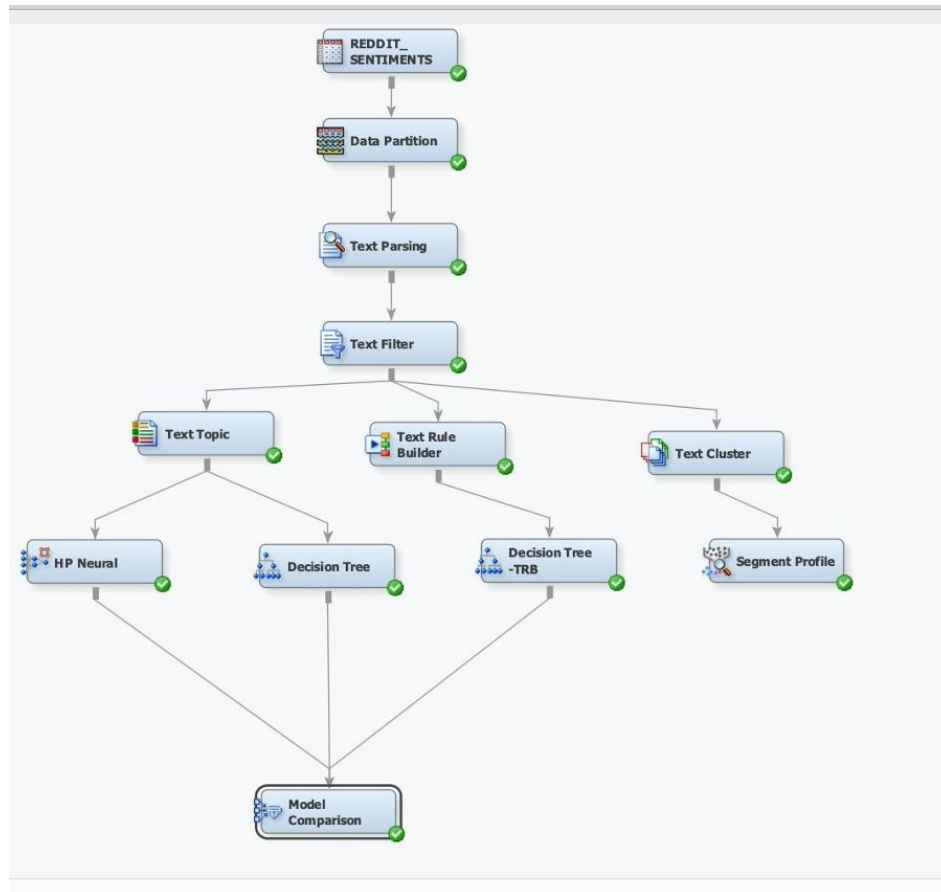
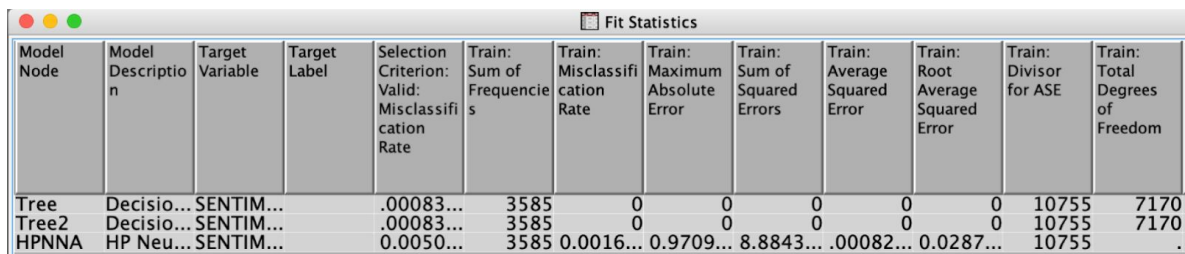


Figure 10: Data interpretation on SAS

Tasks performed :

1. Identify the reddit data source with an **Input Data** node.
2. Partition the input data using the **Data Partition** node.
3. Parse the document collection using the **Text Parsing** node.
4. Reduce the total number of parsed terms using the **Text Filter** node.
5. Cluster documents using the **Text Cluster** node.
6. View the results.
7. Examine data segments using the **Segment Profile** node.
8. Applied ML algorithms and compared.

The above task is performed also for twitter and reddit comments datasets. At first datasets were reformed and sentiment analysis was implemented, as in sas text mining, the sentiment column should be put as target and the rest of the column as features. Later data partition was performed based on training, test and validation set. Validation is mainly used for accessing the adequacy of the model. We have put 60% as training, 20% to validate, and 20% for testing. The **Text Parsing** node is enabled to parse the texts in order to quantify information about the terms that are contained therein. We applied text parsing in order to quantify all the social media relevant words mainly used for commenting or giving feedback about laptops. The next two steps included filtering the text streams for collecting relevant terms and segmenting profiles that generates various reports for facilitating the distribution of the factors within the segments of dataset samples. Machine learning algorithms like decision trees, neural networks were performed and compared.



Model Node	Model Description	Target Variable	Target Label	Selection Criterion: Valid: Misclassification Rate	Train: Sum of Frequencies	Train: Misclassification Rate	Train: Maximum Absolute Error	Train: Sum of Squared Errors	Train: Average Squared Error	Train: Root Average Squared Error	Train: Divisor for ASE	Train: Total Degrees of Freedom
Tree	Decisio...	SENTIM...		.00083...	3585	0	0	0	0	0	10755	7170
Tree2	Decisio...	SENTIM...		.00083...	3585	0	0	0	0	0	10755	7170
HPNNA	HP Neu...	SENTIM...		0.0050...	3585	0.0016...	0.9709...	8.8843...	.00082...	0.0287...	10755	.

Figure 11: Fit statistics from model comparison

CHAPTER 4: MODELLING AND RESULTS

The pie chart below shows the amount of texts about different brands of laptops. This chart shows how the relative feedback of different brands of products contribute to an overall total. A wedge of the circle represents each category's contribution. Here we can see most talked brands of laptops are mac and thinkpad, the second most appeared word is Dell. The least discussed products are acer swift models.

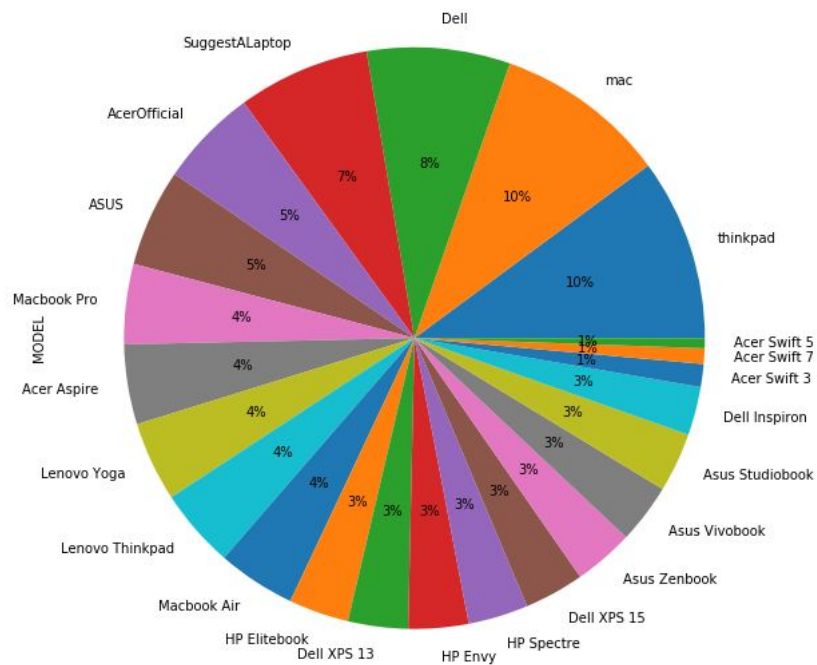


Figure 12: Data Visualization

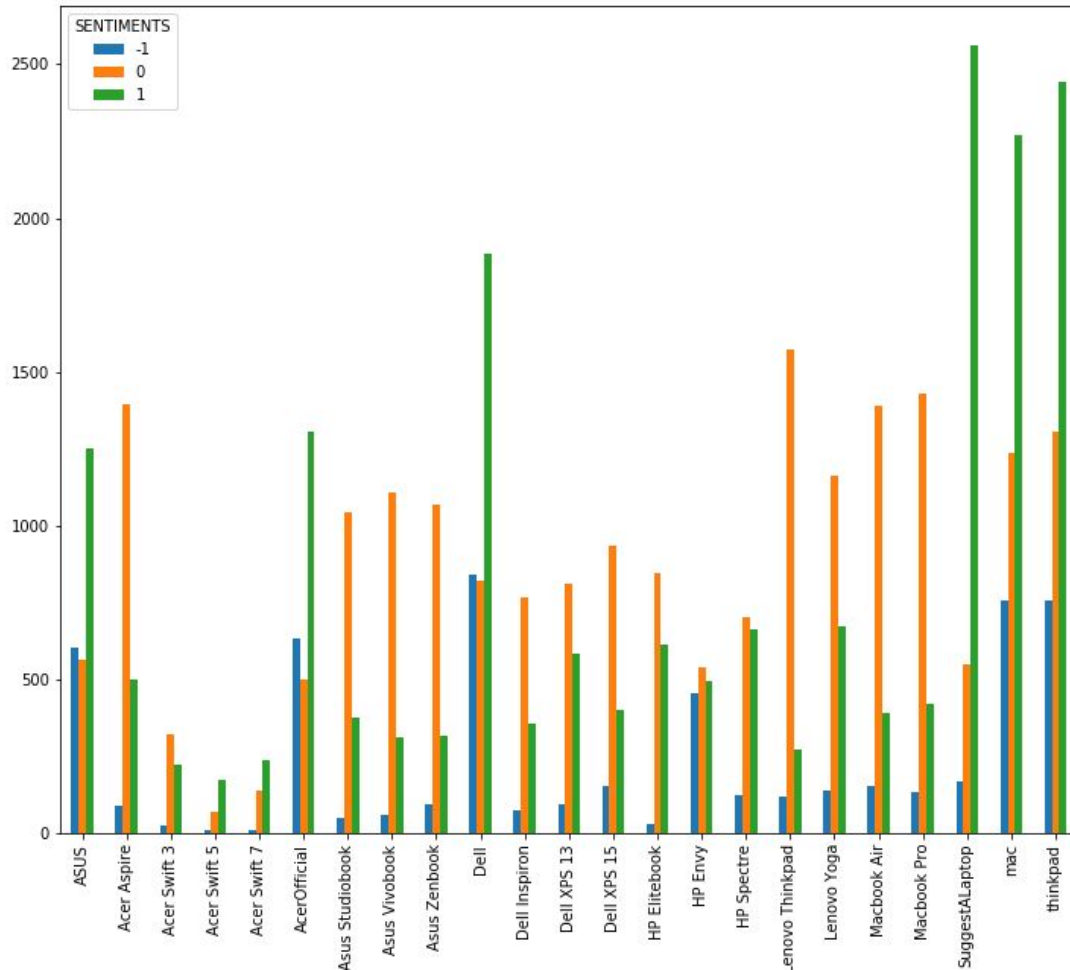


Figure 13: Sentiments or feedbacks based on different brands

From the diagram it can be seen that the highest amount of positive sentiments comes from subreddit r/SuggestALaptop and that is understandable as most of the posts are about recommendation of laptops and when recommending, users tend to state good reasons to buy a certain laptops instead of focusing on not buying specific laptops due to their drawbacks. The next highest positive sentiment seems to be coming from subreddit r/thinkpad meanwhile the same model receives a lot of neutral sentiment feedback from twitter users as you can see from the Lenovo Thinkpad which has the highest amount of neutral sentiment. On the other hand, reddit users also have a lot of complaints about laptops by Dell as can be seen above where it has the highest negative reviews.

Following on our previous milestone of interpreting data, we had trained and tested the data on five different algorithms and obtained satisfactory accuracy scores. The comparison of accuracy scores and f1-scores for each algorithm can be observed by the table below:

Model	Accuracy Score	F1-Score
Logistic Regression	82.72	74.00
Random Forest	83.62	71.77
Gaussian Naive Bayes	68.77	63.38
Support Vector Machine	84.06	76.10
XGBoost	84.88	76.81

Table 2: Comparison of model's accuracy and f1-score.

After training the model and testing it on different dataset portions that have been splitted prior to this, it can be seen that XGBoost classifier gives the highest accuracy and f1-score out of the algorithms. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data like ours, decision tree based algorithms are considered best-in-class. XGBoost approaches the process of sequential tree building using parallelized implementation. This is possible due to the interchangeable nature of loops used for building base learners. Moreover, XGBoost uses a more regularized model formalization to control over-fitting. It also implies cross validation at each iteration of the boosting process, which leads the way to get the precise optimum number of boosting iterations in a single run. Hence, the properties of the algorithm facilitated the modelling and increased the prediction outcome.

CHAPTER 5: WEB APPLICATION

After modeling is done, we moved to building a web application. In this application, we will classify the sentiment of the input text by user and return the sentiment result. We will start this part by explaining how to build the web application.

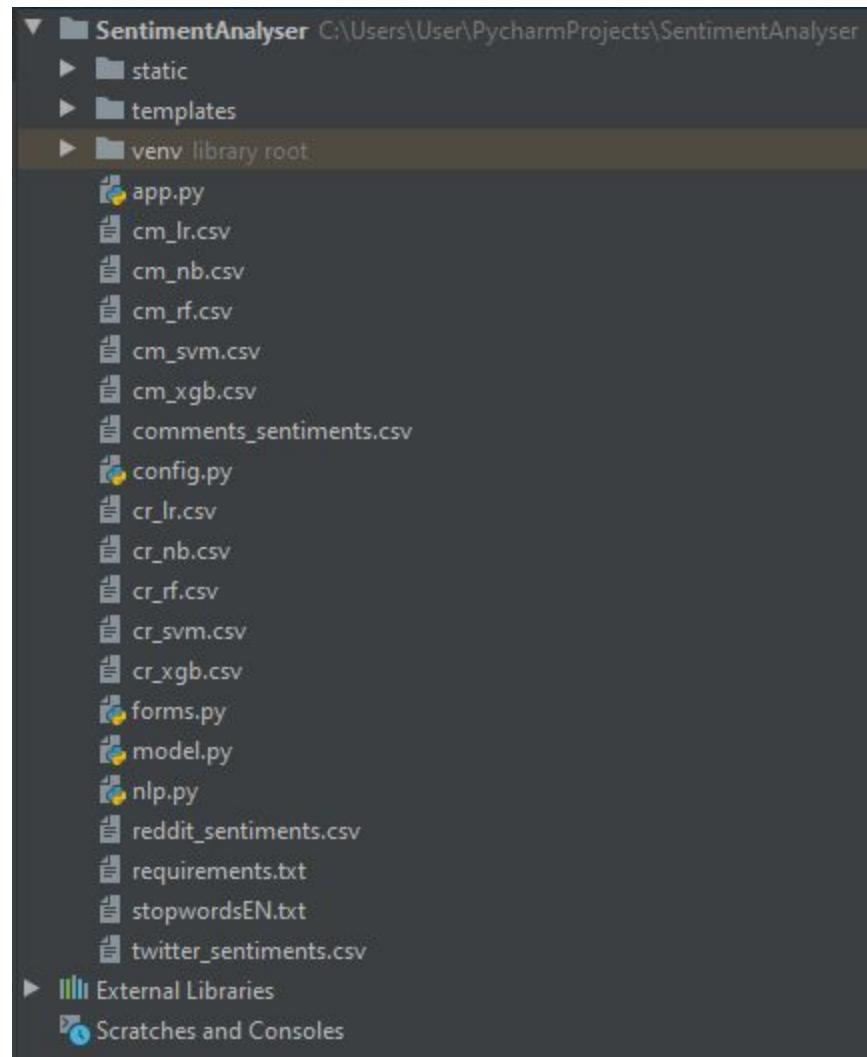


Figure 14: Files inside the SentimentAnalyser project

First and foremost, create a new project and make sure inside the project folder, there should be a python file to execute the application with Flask(__name__) function, a static folder containing a css file for styling the application and finally a templates folder consisting of a html file in order to display the Flask executed codes as a web application. The image above has more than those 3 files because the 3 files are only basic requirements, we can always extend it to apply to our project goal.


```

1  from flask_wtf import FlaskForm
2  from wtforms import TextAreaField, BooleanField
3  from wtforms.validators import DataRequired
4
5  class InputTextForm(FlaskForm):
6      inputText = TextAreaField(validators=[DataRequired()])
7      ignoreCase = BooleanField('ignore case', default=True)
8      ignoreStopWords = BooleanField('ignore stopwords', default=True)

```

Figure 15: forms.py

Let us look at the first python file we name as forms. In this python file, we create a class function to store the input text by the user as an object to be used in other functions, an ignore case of the input text object and also an ignore stopwords object. The input text object is easily understood, but the reason we create an ignore case and ignore stop words object is to allow the user to freely decide whether to lower all the letters of the input text or not and to remove common words that appear in the input text or not.

```

1  import re
2  import nltk
3  from nltk.stem import WordNetLemmatizer
4  from collections import Counter
5
6  class TextAnalyser:
7
8      # stemming values
9      NO_STEMMING = 0
10     STEM = 1
11     LEMMA = 2
12
13     # Text object, with analysis methods
14     def __init__(self, inputText, language="EN"):
15         self.text = inputText
16         self.tokens = []
17         self.sentences = []
18         self.language = language
19         self.stopWords = set(open('stopwords'+ language + '.txt', 'r').read().splitlines())
20
21     def length(self): # results
22         return len(self.text)

```

Figure 16: Snippets of nlp.py

Next, we created another python file and named it as `nlp` which stands for Natural Language Processing. In this python file, we create several functions according to its text cleaning purpose. For example, `tokenize` function for tokenization, `remove_punctuation` function to drop all punctuations, `lemmatize` function for lemmatization, `remove_stop_words` to drop all common words based on a list of common words and so on. Notice from the image above that there are 3 stemming values declared inside the `TextAnalyser` class. These values are for the purpose of letting the user choose either to stem the words using `PorterStemmer` function from `nlTK` library or only use the `split` function instead and if the user wants to lemmatize the input text or not. Other functions aside from preprocessing purpose are used in order to do word count, visualize the most common words (top 10 only) and the number of times a unique word appears.

Next, let us take a look at the main python file that contains the Flask function we named as `app`. For the web application, we also include basic text analysis which consists of word count and basic statistics and sentiment analysis in which the input text will be classified as negative, neutral or positive sentiment using `VADER` sentiment analysis. We choose to use `VADER` because it shows more accurate sentiment classification than other classifiers. `VADER` also considers a lot more variety in sentence constructions compared to others.

```
37     if 'TA' in request.form.values():
38
39         # Text Analysis
40         myText = TextAnalyser(userText, language) # new object
41
42         myText.preprocessText(lowercase = theInputForm.ignoreCase.data,
43                               removeStopWords = theInputForm.ignoreStopWords.data,
44                               stemming = stemmingType)
45
46         # display all user text if short otherwise the first fragment of it
47         if len(userText) > 99:
48             fragment = userText[:99] + " ..."
49         else:
50             fragment = userText
51
52         # check that there is at least one unique token to avoid division by 0
53         if myText.uniqueTokens() == 0:
54             uniqueTokensText = 1
55         else:
56             uniqueTokensText = myText.uniqueTokens()
```

Figure 17: Snippets of `app.py`

After receiving the input text, allowing the user to choose to ignore case, ignore stop words and to do stemming or lemmatizing, this part in the image above will come into play as the main function for analysis. The name 'TA' stands for Text Analysis while another choice is 'SA' for Sentiment Analysis. If the user chooses to see the result for text analysis, the input text will go to this if statement and pre-processing will take place as the TextAnalyser function from our nlp python file is executed. The next code which is an embedded if else statement is for the purpose of displaying the input text in the result page as a reference for our user to see the original text. The next embedded if else statement is for counting the number of unique words appearing in the input text and displaying it also in the result page. Then all the values are loaded into the results html file using render_template.

In another choice, for users who choose to see sentiment analysis results for the input text, the else statement will be executed. In this part, similar codes in which we had used for labelling the target variable in the previous part are again used to obtain the sentiment classification of whether the input text is negative, neutral or positive. The compound score for this sentiment analysis is also loaded into a html file named sentiment along with the sentiment results. We also display the results from our modeling performance in another page consisting of confusion matrix and classification report for each model. Users can observe the performance of each algorithm from this page to determine the accuracy of which model shows the highest among them. Aside from accuracy, there are also results for f1-score using macro average, true positive, true negative and others.

CHAPTER 6: MOBILE APPLICATION

Kivy cross-platform graphical framework a framework for cross-platform, touch-enabled graphical applications. This framework allows python language to be used for both android and linux based platforms.

In this project , the app is created in order to detect sentiments from various users. The process of using kivy and constructing app from it are given below :

Step 1: Installation

```
$ python -m pip install kivy
```

And then install the dependencies like kivymd,

```
python3 -m pip install kivymd
```

As the application framework is easier on the pycharm platform, the sentiment analysis application related codes were taken into pycharm.

Step 2: Importing dataset and the libraries

List of Required libraries :

- pickle
- re
- pandas
- NLTK
- sklearn
- os

```
df = pd.read_csv('/Users/sidratulmuntaha/Sentiment_data.csv', encoding='utf-8')
```

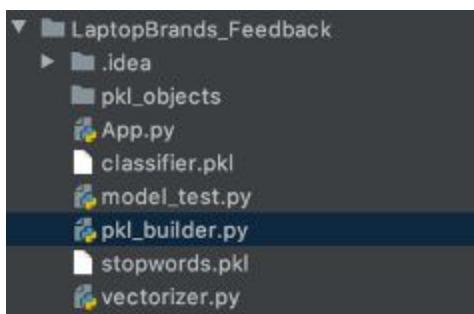
```

import re
from nltk.corpus import stopwords
import nltk

nltk.download("stopwords")
import pandas as pd
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier

```

Number of files and folders for this application:



Step 3: Tokenizing

After importing the dataset the tokenized function was created. Tokenizing is one of the important processes of text analysis where from sentences the words are extracted and refined by removing unnecessary punctuations, formats and modifiers.

```

def tokenizer(text):

    features = df.iloc[:, 1].values
    labels = df.iloc[:, 2].values

    processed_features = []

    for sentence in range(0, len(features)):
        # Remove all the special characters
        processed_feature = re.sub(r'\W+', ' ', str(features[sentence]))

        # Remove numbers
        processed_feature = re.sub(r'[0-9]+', ' ', processed_feature)

        # Remove all single characters
        processed_feature = re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)

```

Step 4: Pickling

In the same .py file .pkl files were created, one of which contains the classifiers to classify and another pkl with stopwords, Stop-words are those words that are extremely common in the English language and do not contain important information about the text, so it's better to identify and remove them.

```
pickle.dump(stop,
            open('stopwords.pkl', 'wb'),
            protocol=4)

pickle.dump(clf,
            open('classifier.pkl', 'wb'),
            protocol=4)
# creating the .pkl files
```

Step 5: Vectorizing

Later the text column is vectorized, vectorized means when the text is converted to analysable data.

```
vect = HashingVectorizer(decode_error='ignore',
                        n_features=2**21,
                        preprocessor=None,
                        tokenizer=tokenizer)
```

Step 6: Classifying

After importing the needed modules, the classifier is loaded and then the classify function is pasted in model_test.py.

The layout designing modules for kivy is as follows

:

```
# importing kivy moduels
import ...
kivy.require('1.11.1')
from kivy.uix.textinput import TextInput
from kivy.uix.button import Button
from kivy.uix.label import Label
from kivy.uix.boxlayout import BoxLayout
# ending importing kivy moduels
```

```
class Application(App):
    def build(self):
        # creating our layout. we want it to display objects vertically
        self.layout = BoxLayout(orientation='vertical')
        # creating the submit review button, our review field, the label displaying
        submit_button = Button(text='classify feedback', on_press=self.classify_f)
        self.review = TextInput(hint_text='Your laptop popularity detection')
        self.label = Label()
        self.layout.add_widget(self.review)
        self.layout.add_widget(submit_button)
        self.layout.add_widget(self.label)
        return self.layout

    def classify_f(self, action):
        print(self.review.text)
        pred, proba = classify(self.review.text)
        print(pred)
        print(str(round(proba, 2)*100) + " % ")
        self.label.text = pred + " " + str(round(proba, 2)*100) + " % "

if __name__ == '__main__':
    # running the application
    Application().run()
```

APPENDIX

The followings are screenshots of the Home page of the web application:

Navigation: [Home](#) | [About](#)

Welcome to the Sentiment Analyser App

WQD7005 - Data Mining Project

Sentiment Analyser is an app for Natural Language Processing (NLP). This app utilizes VADER Sentiment tool to classify whether a social media text on a laptop brand and model is positive, neutral or negative with calculated compound score.

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. The Compound score is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1 (most extreme negative) and +1 (most extreme positive). Finally, based on compound score, we will classify the text as positive, neutral or negative sentiment.

Input text

You can enter your text here:

Write something ...

User text - Language:

☒ English

Settings

☒ ignore case [\[What is this?\]](#)

☒ ignore stopwords (articles, conjunctions, ...) [\[What is this?\]](#)

☐ Enable stemming (English only, will be slower): Stem or ☐ Lemmas [\[What is this?\]](#)

Tools available

Basic Analysis

Sentiment Analysis

The images below are a look at the About page of the web application:

Navigation: [Home](#) | [About](#)

Evaluation of Models

This page explains the performance of each model on our social media dataset. We use five machine learning algorithms to train and test the dataset namely logistic regression, random forest, gaussian naive bayes, support vector machine and lastly, xgboost. All of the model shows a high accuracy score except naive bayes with accuracy more than 80%. The model that shows the highest accuracy is xgboost.

Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The columns are the predictions and the rows are the actual values. The main diagonal gives the correct predictions. That is, the cases where the actual values and the model predictions are the same.

Logistic Regression

	Negatives	Neutrals	Positives
Negatives	667	431	524
Neutrals	81	5610	329
Positives	187	685	4820

Random Forest

	Negatives	Neutrals	Positives
Negatives	451	366	835
Neutrals	45	5575	400
Positives	82	485	5145

Naive Bayes

	Negatives	Neutrals	Positives
Negatives	1079	327	246
Neutrals	678	5029	313
Positives	1725	897	3070

Support Vector

	Negatives	Neutrals	Positives
Negatives	695	438	519
Neutrals	50	5673	297
Positives	187	640	4985

XGBoost

	Negatives	Neutrals	Positives
Negatives	743	376	533
Neutrals	117	5624	279
Positives	234	475	4983

Classification Report

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report.

Logistic Regression

	precision	recall	f1-score	support
-1	0.722280	0.421913	0.532671	1652.00000
0	0.834077	0.931894	0.880276	8020.00000
1	0.849639	0.845803	0.848218	5692.00000
accuracy	0.832610	0.832610	0.832610	0.83261
macro avg	0.801998	0.735536	0.753722	13364.00000
weighted avg	0.826885	0.832610	0.826863	13364.00000

Random Forest

	precision	recall	f1-score	support
-1	0.780277	0.273002	0.404464	1652.00000
0	0.870278	0.926080	0.897312	8020.00000
1	0.809426	0.903900	0.852386	5692.00000
accuracy	0.835902	0.835902	0.835902	0.835902
macro avg	0.818994	0.700994	0.718061	13364.00000
weighted avg	0.831957	0.835902	0.817286	13364.00000

Naive Bayes

	precision	recall	f1-score	support
-1	0.306079	0.053148	0.420335	1652.000000
0	0.804254	0.835382	0.819523	8020.000000
1	0.845063	0.539353	0.658728	5562.000000
accuracy	0.686770	0.586770	0.686770	0.68677
macro avg	0.653365	0.675961	0.632862	13384.000000
weighted avg	0.780906	0.586770	0.701691	13384.000000

Support Vector

	precision	recall	f1-score	support
-1	0.745708	0.420702	0.537826	1652.000000
0	0.840320	0.842359	0.884419	8020.000000
1	0.856063	0.854708	0.855535	5562.000000
accuracy	0.840542	0.840542	0.840542	0.840542
macro avg	0.814130	0.739256	0.760627	13384.000000
weighted avg	0.835458	0.840542	0.831087	13384.000000

XGBoost

	precision	recall	f1-score	support
-1	0.679159	0.449758	0.541151	1652.000000
0	0.868571	0.934219	0.900200	8020.000000
1	0.859679	0.875439	0.867589	5562.000000
accuracy	0.849297	0.849297	0.849297	0.849297
macro avg	0.802537	0.753139	0.769947	13384.000000
weighted avg	0.841455	0.849297	0.841926	13384.000000

Mobile application from the user end :

