Hong Kong Metropolitan University

School of Science & Technology

COMP S460F

Advanced Data Mining and Analytics

2023 Autumn

Project Title: SVM CLASSIFICATION FOR SENTIMENT ANALYSIS OF YOUTUBE COMMENTS

By: Brian Lee, Zhu Jiawei,
James Xie Ziyang, Yeung Cho Ho

Date: December 2023

Table of Content

1 Introduction

1.1 Problem Background

The purpose of this report is to provide a description of the project titled "SVM Classifier for Sentiment Analysis of YouTube Comments." Within this project, we delve into the problem description, research objectives, and associated challenges involved in conducting sentiment trend analysis on the YouTube platform. We address various challenges, including API limitations, potential alterations in the site's structure, and the diverse nature of comments. Additionally, we highlight the complexities associated with employing unsupervised learning techniques for sentiment analysis and interpreting studies containing implicit sentiment.

There are multiple challenges in conducting data collecting on the YouTube platform. First, YouTube's API has daily request and quota limits, which may limit our ability to obtain comment data. Second, the structure of the YouTube website may change over time, which may cause our web crawler code to fail. Additionally, reviews from different users may use other languages, vocabulary, and cultural backgrounds, increasing the review diversity's challenge.

As the review data lacks sentiment labels, unsupervised learning methods are required for sentiment analysis. This presents interpretive difficulties since the emotional significance of the comments cannot be determined directly. Additionally, judging sarcastic comments poses another issue. Sarcastic comments frequently carry concealed emotions, thereby amplifying the challenge of accurate assessment.

1.2 Research Objective

Our research objective is to analyze sentiment trends in YouTube comments, unveiling shifts in sentiment triggered by specific events, topics, or content. We aim to conduct sentiment analysis on reviews utilizing a Support Vector Machine (SVM) classifier, enabling us to ascertain whether the sentiment expressed is positive, negative, or neutral. Through the analysis of comment data, we will assess the influence of specific events or content on user emotions, providing

quantitative insights into emotional trends. Additionally, we will employ diverse performance metrics to evaluate the accuracy of the model and compare it with alternative SVM models.

2 Tabulation - Description of Dataset

The information about the English youtube comments can be found in Kaggle as shown in the link below, with a csv name “ytb_comments” used for data cleaning and preprocessing: https://www.kaggle.com/code/ahmedshahriarsakib/scrape-youtube-comments-for-free-no-google-api/output

After the completion of data cleaning and preprocessing in sections 3.2 and 3.3, there are a total of 213 records stored in the dataset, with a csv file named “ENGLISH_youtube_comments” as the preprocessed and cleaned output. There are 9 variables, 1 numeric and 8 categorical variables. All the records in the preprocessed and cleaned dataset will be used for data analysis, visualization and evaluation in sections 3, 4 and 5 respectively.

3 Methodology and Implementation

3.1 Flow Chart

The methodology section describes the sequence of the sections that will be discussed in section 3-5. Figure 3.1-1 shows the flowchart below.

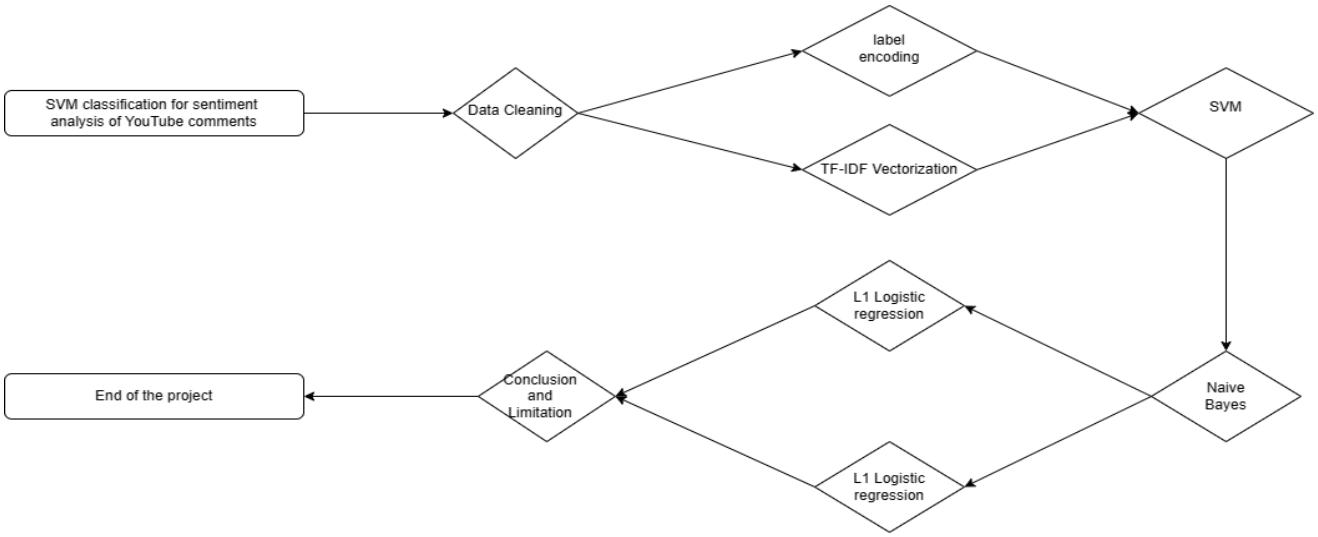


Figure 3.1-1 Flow Chart of all implementations in this project

3.2 Data Cleaning

Data cleaning is the process of fixing incorrect or incomplete data inside a dataset. It is one of the important parts in this project to conduct sentiment analysis. It plays a significant role in building a model. The purpose of data cleaning is to make the data that we collect more concise and readable. In this step, we make all the text (comments) become lowercase, so there will not be any uppercase after lowercasing. Moreover, the removal of special characters like square brackets, punctuation and numbers are also performed. After data cleaning, the remaining text will be all lowercase text.

```
# Lowercase the text
text = text.lower()

# Remove punctuation and special characters (keep only alphabets and spaces)
text = re.sub(r'[^a-z\s]', '', text)

# Tokenize the text by splitting on spaces
words = text.split()

# Remove stop words
words = [word for word in words if word not in stop_words]

# Lemmatize the words,reducing a word to its base or root form
words = [lemmatizer.lemmatize(word) for word in words]

# Join the words back into a single string
text = ' '.join(words)
```

Figure 3.2-1 Lowercasing

```
# example of re.sub() function which remove any characters
import re

text = " 'Python3' is a versatile programming language!"
text = re.sub(r'[^a-z\s]', '', text)
print(text)
```

```
ython is a versatile programming language
```

Figure 3.2-2 Character Removal

3.3 Data Preprocessing

Data preprocessing is another important part in this project to conduct sentiment analysis. In this step, we will continue to process the data. When the data is preprocessed, the computer will be able to read the input easily. We perform tokenization at this step, which is the process of converting a text into tokens. A token can be considered as an individual word inside a

sentence, inside that sentence, e.g. “I went to New-York to play football”, where each word represents a token.

Stop words are words that are separated out before or after the text preprocessing stage. There are 179 stop words in the English corpus, e.g. “you”, “you’re”, “you‘ve“, these are considered as the noise in the text, which these stop words should be removed.

Next, lemmatization is conducted in the text. Lemmatization is an organized procedure of obtaining the root form of the word. If the input is “running”, then the output will be “run”. The word “running” is converted into its basic form “run”. The benefit of using lemmatization is that it considers the context and converts the word to its meaningful base form. Finally, we join all the cleaned and preprocessed texts back together, to make them complete. Furthermore, an extra column “Comment_preprocessed” is added for better comparison. The figure below provides some examples after we did the data cleaning and preprocessing.

	Comment_content	Comment_preprocessed
0	Here's a link to the time-saving workouts you ...	here link timesaving workout download httpsbui...
1	Having a well-structured workout plan is key. ...	wellstructured workout plan key break long sho...
2	00:20 Strategy 1 Minimal volume\n04:30 Strateg...	strategy minimal volume strategy supersets str...
3	Some of my best progress was in a home gym doi...	best progress home gym set per week focusing p...
4	This is really good to know and matches with w...	really good know match ive im older started wo...
5	Hey Jeremy! With all the new studys and scienc...	hey jeremy new study science research id love ...
6	For the last couple years I've been doing a 3-...	last couple year ive day full body split x foc...
7	Thank you so much for the free workout program...	thank much free workout program wisdom ive str...
8	I have included drop sets (#3) the last 6 mo...	included drop set last month effective cut tim...
9	Hi Jeremy, first of all I want to thank you fo...	hi jeremy first want thank valuable content ac...
10	Hi Jeremy, how do you think doing sets of drop...	hi jeremy think set dropsets would work like r...
11	Huge factors for me are motivation, consistenc...	huge factor motivation consistency injury prev...
12	Good stuff. The only downside of drop sets is ...	good stuff downside drop set end hogging bunch...
13	Hey Thanks for this awesome video! I'm 15 and ...	hey thanks awesome video im fairly fit biceps ...
14	I’ve been doing Myo Reps lately and that’s bee...	ive myo rep lately thats fantastic saving time...
15	Love Your Channel.Its always informative.Keep ...	love channelits always informativekeep great work
16	Minimal volume worked wonders for me. More tha...	minimal volume worked wonder set find need way...
17	Thanks for the plans Jeremy always coming in c...	thanks plan jeremy always coming clutch
18	I work out using the 3/7 method you explained ...	work using method explained month ago work fin...
19	This does work. At 58 I don't have as much ene...	work dont much energy need rest time usually f...
20	Hope you guys enjoyed this one! See below for ...	hope guy enjoyed one see link study referenced...

Figure 3.3-1 preprocessed comments vs original comment content

3.3.1 TF-IDF Vectorization

TF-IDF vectorizer is a technique which is used to find the meaning of sentences consisting of words, which is good for a machine that reads words in numbers (Madan, 2019). The formulas are:

$$TF = \frac{Number\ of\ repetitions\ of\ word\ in\ a\ document}{Number\ of\ words\ in\ a\ document}$$

$$IDF = log\left(\frac{Number\ of\ documents}{Number\ of\ words\ in\ a\ document}\right)$$

By calculating TF-IDF, we will be able to find the frequency of occurrence of a word inside some particular documents. If the value is larger, we can say that the word is more important inside that document. Since we are using the column “Commment_preprocessed” with 212 rows of texts, we just simply apply TF-IDF to this column. Then a vocab of document is generated, each word inside that vocab of document represents 1 column. From the example below, we can see the importance of the word in each row (comments). In conclusion, the 212 rows * 1455 columns is the numerical representation of the ‘Comment_preprocessed’ data above.

```
Number of comments: 212
Number of unique words: 1455
TF-IDF scores for the first few comments:
          ab  abandon  ability  able  abruptly  absolute  absolutely  \
0   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
1   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
2   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
3   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
4   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
5   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
6   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
7   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
8   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
9   0.000000      0.0      0.0   0.0       0.0       0.0         0.0
10  0.000000      0.0      0.0   0.0       0.0       0.0         0.0
11  0.000000      0.0      0.0   0.0       0.0       0.0         0.0
12  0.000000      0.0      0.0   0.0       0.0       0.0         0.0
13  0.174905      0.0      0.0   0.0       0.0       0.0         0.0
14  0.000000      0.0      0.0   0.0       0.0       0.0         0.0
15  0.000000      0.0      0.0   0.0       0.0       0.0         0.0
16  0.000000      0.0      0.0   0.0       0.0       0.0         0.0
17  0.000000      0.0      0.0   0.0       0.0       0.0         0.0
18  0.000000      0.0      0.0   0.0       0.0       0.0         0.0
19  0.000000      0.0      0.0   0.0       0.0       0.0         0.0
```

Figure 3.3.1-1 TF-IDF output

	abstract	according	accuracy	...	write	xml	year	yes	yet	\
0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
1	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
2	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
3	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
4	0.0	0.0	0.0	...	0.0	0.0	0.143634	0.0	0.000000	
5	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
6	0.0	0.0	0.0	...	0.0	0.0	0.161779	0.0	0.000000	
7	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
8	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
9	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
10	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
11	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
12	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
13	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.146745	
14	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
15	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
16	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
17	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
18	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	
19	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.000000	

Figure 3.3.1-2 and 3.3.1-3 TF IDF outputs

3.3.2 Label Encoding

In this step, we will need to assign labels for the above data ‘Comment_preprocessed’ for later use. Considering that there are too many comments, which label the comments manually is not ideal and applicable. Here we use vader which is a rule-based sentiment analyzer, and is explicitly sensitive to suppositions communicated in web-based media. This sentiment analyzer is helpful for us to label the text of column ‘Comment_preprocessed’ automatically. The figure below is an example of VADER’s vocabulary. The more positive words have higher positive evaluations and more adverse words have lower negative grades. VADER examines a piece of text, it verifies whether any of the words in the content are available in the lexicon, and then produces sentiment measurements grading from words as shown in figure 3.3.2-1.

Word	Sentiment rating
tragedy	-3.4
rejoiced	2.0
insane	-1.7
disaster	-3.1
great	3.1

Figure 3.3.2-1

Overall, VADER returns a compound score, the value is between -1 and 1. When the compound score is > 0.05, we label them positive, when the compound score is < 0.05, we label them

negative, or otherwise we label them neutral. After we label all the text, we can also convert the 3 kinds of labels into numbers. 0 represents negative, 1 represents neutral, 2 represents positive. Figure 3.3.2-2 is the result after we labeled the preprocessed text.

	Comment_content	Comment_preprocessed	Sentiment	Sentiment_label_numbers
0	Here's a link to the time-saving workouts you ...	here link timesaving workout download httpsbui...	positive	2
1	Having a well-structured workout plan is key. ...	wellstructured workout plan key break long sho...	neutral	1
2	00:20 Strategy 1 Minimal volume\n04:30 Strateg...	strategy minimal volume strategy supersets str...	neutral	1
3	Some of my best progress was in a home gym doi...	best progress home gym set per week focusing p...	positive	2
4	This is really good to know and matches with w...	really good know match ive im older started wo...	positive	2
5	Hey Jeremy! With all the new studys and scienc...	hey jeremy new study science research id love ...	positive	2
6	For the last couple years I've been doing a 3-...	last couple year ive day full body split x foc...	positive	2
7	Thank you so much for the free workout program...	thank much free workout program wisdom ive str...	positive	2
8	I have included drop sets (#3) the last 6 mo...	included drop set last month effective cut tim...	positive	2
9	Hi Jeremy, first of all I want to thank you fo...	hi jeremy first want thank valuable content ac...	positive	2
10	Hi Jeremy, how do you think doing sets of drop...	hi jeremy think set dropsets would work like r...	negative	0
11	Huge factors for me are motivation, consistenc...	huge factor motivation consistency injury prev...	negative	0
12	Good stuff. The only downside of drop sets is ...	good stuff downside drop set end hogging bunch...	negative	0
13	Hey Thanks for this awesome video! I'm 15 and ...	hey thanks awesome video im fairly fit biceps ...	positive	2
14	I've been doing Myo Reps lately and that's bee...	ive myo rep lately thats fantastic saving time...	positive	2

Figure 3.3.2-2 preprocessed output after data preprocessing

4 Analysis and Implementation

4.1 SVM Classification

After we finished the steps in 3.2 and 3.3, we split the data into train sets and test sets for model training. The train set is the numerical representation of the ‘Comment_preprocessed’ data that is completed in section 3.3.1, and the test set we are using is the labeled data ‘Comment_preprocessed’ into 3 classes. 80% of the data will be used for training and 20% of the data will be used for testing.

We are conducting sentiment analysis in this project, we use SVM as our model to apply it. SVM stands for Support Vector Machine, which is a supervised algorithm that can be used for both regression and classification tasks, but generally, they work best in classification problems (Saini, n.d.). There are 2 types of SVM: Linear SVM and Non-Linear SVM. Linear SVM is used when the data is perfectly linearly separable. While the Non-Linear SVM is not linearly separable. The data we have here has multiple classes which include 3 categories - ‘negative’, ‘neutral’ and ‘positive’. For multiple classification, the principle is that we break down the multi-classification problem into smaller subproblems, all of which are binary classification problems. Here we will use One versus All (OVA). In this technique, if we have 3 class

problems, then we learn 3 SVMs. For example, SVM number -1 learns “class_output = positive” vs “class_output ≠ positive”. We predict the output for new input, just predict with each of the build SVMs and then find which one puts the prediction the farthest into the positive region. Bue there may be some limitations when we train N SVMs, like the unbalanced problem. For example, if we are working on an MNIST dataset, we have 10 classes, each class 1000 has points, for any one of the SVM having two classes, one may have 9000 points while the other only has 1000 points. One potential solution is that we can use the 3-sigma rule of the normal distribution which is we fit the data to a normal distribution and then resampled them appropriately, in order to maintain the class distribution.

There is a simple sample of OVA, consider the below figure, the green line tries to maximize the gap between green points and other two points.

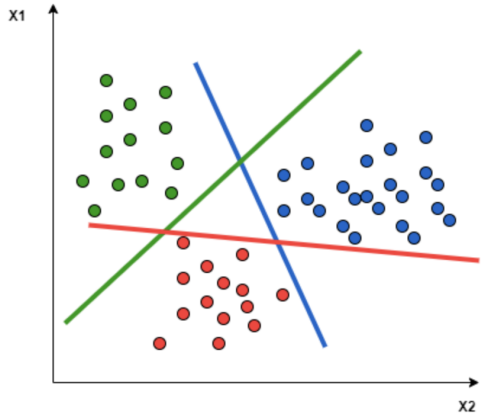


Figure 4.1-1

For making the classification easier, we use “Kernel Trick”. The Kernels can help us to convert the lower dimension space to a higher dimension space using some quadratic functions which will allow us to find a decision boundary that clearly divides the data. There are other Kernels like Polynomial Kernel,Sigmoid Kernel,RBF Kernel, etc. After we tune the model, we can find that the training score is around 0.864, and the best parameters are C =10, degree =1,gamma = 0.01 and the Kernel is linear, as well as the training score of the Linear Kernel.

```
: # Import all relevant libraries

from sklearn.svm import SVC

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn import preprocessing

import warnings



warnings.filterwarnings("ignore")

#Now let's read the dataset and look at the columns to understand the information better.

#Define support vector classifier with hyperparameters

X_train, X_test, y_train, y_test = train_test_split(features_df,labels, test_size=0.2)

svc = SVC(random_state=101)
svc.fit(X_train,y_train)

#svc = SVC(random_state=101)

accuracies = cross_val_score(svc,X_train,y_train,cv=5)
print("Train Score:",np.mean(accuracies))
print("Test Score:",svc.score(X_test,y_test))
```

Figure 4.1-2 parameter outputs

```
#svc = SVC(random_state=101)

accuracies = cross_val_score(svc,X_train,y_train,cv=5)
print("Train Score:",np.mean(accuracies))
print("Test Score:",svc.score(X_test,y_test))
```

```
Train Score: 0.8639928698752228
Test Score: 0.7906976744186046
```

```
29]: # hyper tune model
grid = {
    'C':[0.01,0.1,1,10],
    'kernel' : ["linear","poly","rbf","sigmoid"],
    'degree' : [1,3,5,7],
    'gamma' : [0.01,1]
}
svm  = SVC ()
svm_cv = GridSearchCV(svm, grid, cv = 5)
svm_cv.fit(X_train,y_train)
print("Best Parameters:",svm_cv.best_params_)
```

```
Best Parameters: {'C': 10, 'degree': 1, 'gamma': 0.01, 'kernel': 'linear'}
```

```
30]: print("Train Score:",svm_cv.best_score_)
print("Test Score:",svm_cv.score(X_test,y_test))
```

```
Train Score: 0.8641711229946523
Test Score: 0.813953488372093
```

Figure 4.1-3 parameter outputs

There are some parameters in SVM that can be adjusted, such as the width of margin, the noise points which are laid in the margin. In sk-learn default path, the library will use soft margin rather than kernel method. The result of soft margin is seriously influenced by noise, figure 4.1-4 to 4.1-5 show the plots.

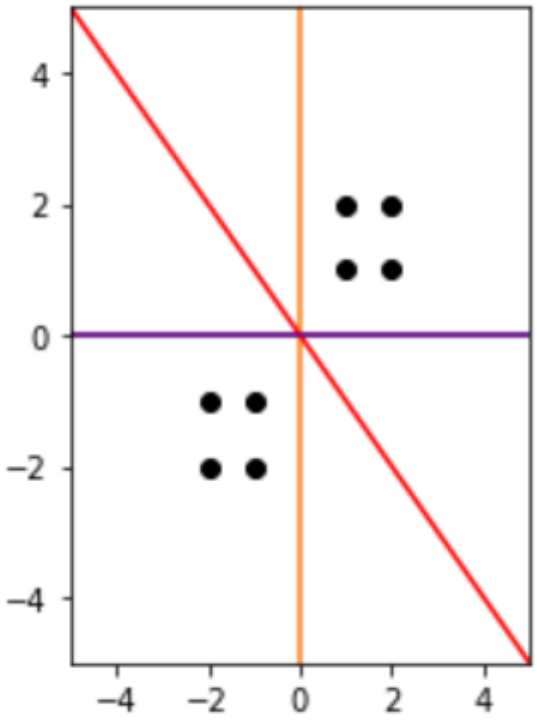


Figure 4.1-4 SVM output

In this graph, the red line and purple line are two options for SVM. It is obvious that the red one will be better, But sometimes the noise will influence it. Like the graph below.

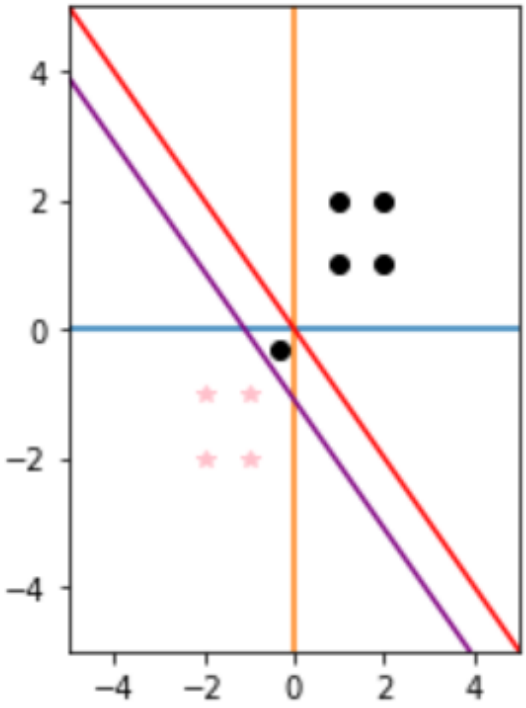


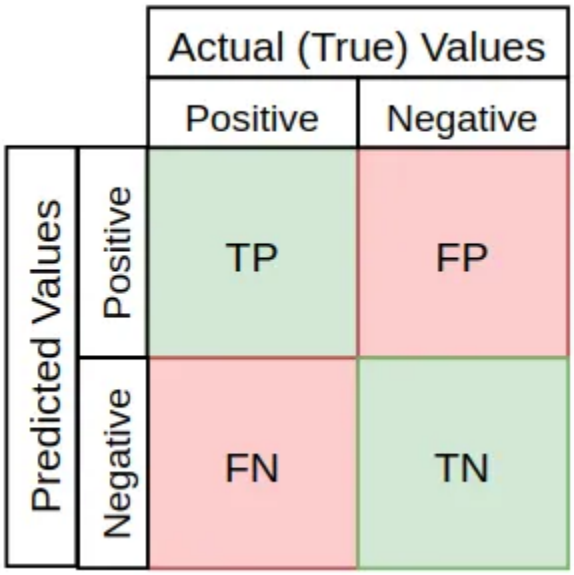
Figure 4.1-5 SVM output

The two kinds of points are classified by shape and color. If one black circle point is laid at (-0.3,-0.3), the user should decide to use which line to do SVM, whether the red one or the purple one. Obviously, the purple one is a better classifier, but for SVM, it means a much more narrow margin. There should be a balance between a better classifier and a wider margin. This is the “C” parameter in the sk-learn package SVM method. The larger “C” it is, the narrower margin it will be, also a better classifier.

4.2 Naive Bayes

4.2.1 Confusion Matrix

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm. The advantage of the confusion matrix lies in its ability to provide a clear and concise comparison between predicted and actual results. This makes the differentiation between various labels more distinct and comprehensible.



(Kanstrén, 2020)

Figure 4.2.1-1 Confusion Matrix Example

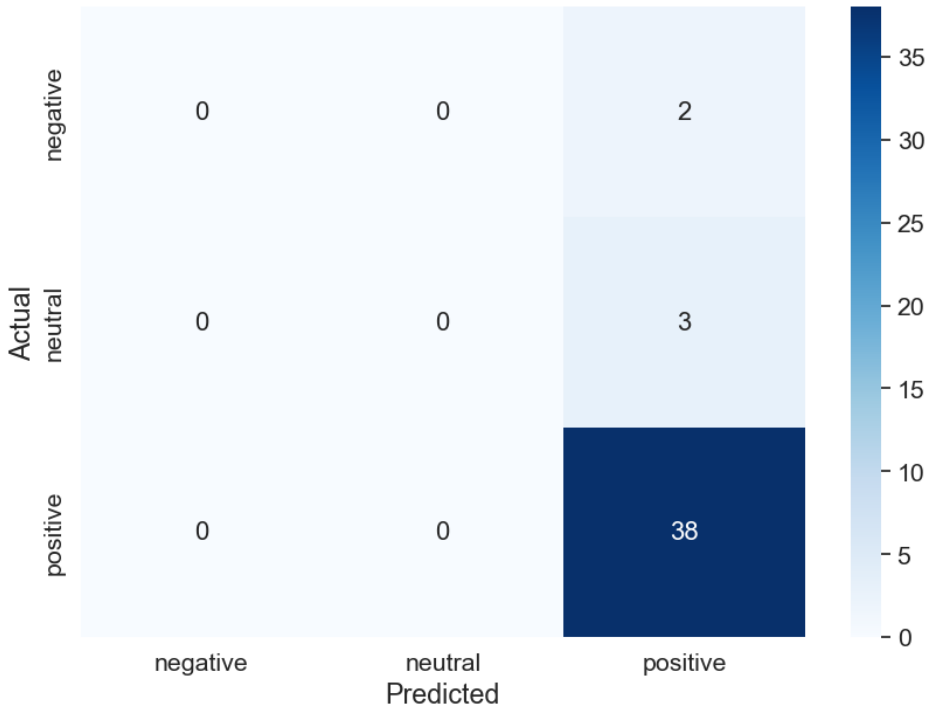


Figure 4.2.1-2 Confusion Matrix for Data Training and Testing

Normally, the confusion matrix will be considered as a binary table. Which include true positive, true negative, false positive, false negative. True and false is based on the classification of predicted labels, we set one label as true, then the other is false. Positive and negative is used to judge if the predict is correct, if the judgment is correct, it should be positive, otherwise, it should be negative.

But this time, the sample used for sentiment analysis is multiple level, so we could not easily use a binary way to estimate. We should use each kind of label, consider it is true, then get many binary confusion matrices. For example, setting [3,3] as true, then [3,3] will be true positive, [3,1], [3,2] will be true negative, [1,1], [2,2] will be false negative, the others are false negative.

After obtaining three binary confusion matrices, there are two ways for us to get the average, one is Macro, the other is Micro, here we choose Macro to do calculation, because Macro will be sensitive to the difference between samples.

4.2.2 Performance Metrics

Evaluation of a machine learning model is one of the essential steps for constructing a highly-accurate and effective model. Various metrics are used for model evaluation, such as precision, recall, F1 score, ROC and AUC in section 4.2.2.1 to 4.2.2.4, for interpreting the wellness of the model performance with the training data, and generalizing on test data and new data. From the evaluation process, the machine learning model can be improved and enhanced by fine-tuning the hyperparameters.

In the evaluation of precision, recall and F1 score in sections 4.2.2.1, 4.2.2.2 and 4.2.2.3 respectively, the macro average is used to compute the values of each of the performance metrics, by calculating arithmetic mean of the individual class. Then, all classes are treated equally to evaluate the overall performance of the classifier against the most common class labels. For ROC and AUC in section 4.2.2.4, the models are evaluated via the OVR (one-versus-rest) approach by transforming the ternary classification problem into three binary classification problems.

4.2.2.1 Precision

Precision is a performance metric for measuring how many of the positive predictions made are correct (true positives). Mathematically, it is the ratio of the amount of positive predictions that are classified as correct over the total amount of positive predictions (the sum of true positive and false positive) as shown in the formulas shown below in figure 4.2.2.1-1 to 4.2.2.1-2 (Kanstrén, 2020).

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

(Kundu, 2022)

Figure 4.2.2.1-1 formula of precision score

$$\frac{\text{True Positives}}{\text{True Positives + False Positives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Predictions you Made}}$$

(Kanstrén, 2020)

Figure 4.2.2.1-2

```
Training data Precision = 0.632258064516129
Test data Precision = 0.2945736434108527 4
```

Figure 4.2.2.1-3 training and testing data of precision score

As shown from the output in figure 4.2.2.1-3, the precision of the test data is rather low, with only 0.295. There is a substantial amount of false positives identified in the classifier, caused by imbalanced class and untuned hyperparameters. Class 2 (positive) is represented significantly more frequently than the other, with reference to the confusion matrix in section 4.2.1. The specificity of the data should be further enhanced to be more related to the targeted predicting variable.

4.2.2.2 Recall

The recall is the measure of our model correctly identifying True Positives, Mathematically, it is the ratio of the amount of positive predictions that are classified as correct over the total sum of true positives and false negatives as shown in the formulas shown below in figure 4.2.2.2-1 to 4.2.2.2-2 (Kanstrén, 2020).

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

(Kundu, 2022)

Figure 4.2.2.2-1 formula of recall score

$$\frac{\text{True Positives}}{\text{True Positives + False Negatives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Instances in the Dataset}}$$

(Kanstrén, 2020)

Figure 4.2.2.2-2

```
Training data Recall = 0.6078431372549019
Test data Recall = 0.3333333333333333
```

Figure 4.2.2.2-3 training and testing output of recall

As shown in the figure 4.2.2.2-3, the recall of the test data is rather low, with only 0.333. There is a substantial amount of false negatives identified in the classifier, caused by imbalanced class and untuned hyperparameters. Class 2 (positive) is represented significantly more frequently than the other, with reference to the confusion matrix in section 4.2.1. Similar to precision in section 4.2.2.1, more data collection should be conducted, as well as fine-tuning the model hyperparameters, implementing class weights and ensembling in order to improve the recall score.

4.2.2.3 F1 Score

The F1 score is an evaluation metric used in binary and multi-class classification. This metric can evaluate the performance of imbalanced class

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

(Kundu, 2022)

Figure 4.2.2.3-1 formula of F1 score

```
Training data F1 Score = 0.6162680125813766
Test data F1 Score = 0.3333333333333333
```

Figure 4.2.2.3-2 training and test data outputs of F1 Score

As shown in the figure 4.2.2.3-2, the F1 Score of the test data is rather low, with only 0.333, as a result of the low precision and recall scores from sections 4.2.2.1 and 4.2.2.2 respectively. There is a substantial amount of false negatives identified in the classifier, caused by imbalanced class and untuned hyperparameters. Class 2 (positive) is represented significantly more frequently than the other, with reference to the confusion matrix in section 4.2.1. The model may not be able to differentiate the minority class from the majority class, leading to poor performance with respect to the low F1 score overall. To combat the problem of imbalance

class, the dataset is required to undergo over and under-sampling, as well as fine-tuning hyperparameters and setting weight coefficients for each class in multiclass classification.

4.2.2.4 ROC and AUC

The ROC curve is a tool used for evaluating the performance of a classification model. It illustrates the balance between the true positive rate (TPR) and false positive rate (FPR) at various thresholds as shown in the formula in figure 4.2.2.4-1. TPR measures the model's accuracy in correctly identifying positive class samples, while the FPR measures its tendency to incorrectly classify negative class samples as positive. AUC is a more effective performance metric than accuracy in imbalance datasets, by evaluating all the thresholds rather than choosing a single threshold in classification to show the accuracy of the model performance (Singh, n.d.).

$$\text{TPR (sensitivity)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR (1-specificity)} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

Figure 4.2.2.4-1

The Area Under the Curve (AUC) represents the area under the ROC curve and serves as a measure to evaluate the overall classification performance of a model. It ranges from 0 to 1, where 0.5 signifies random guessing and 1 signifies perfect classification. A higher AUC value indicates superior classification ability of the model.

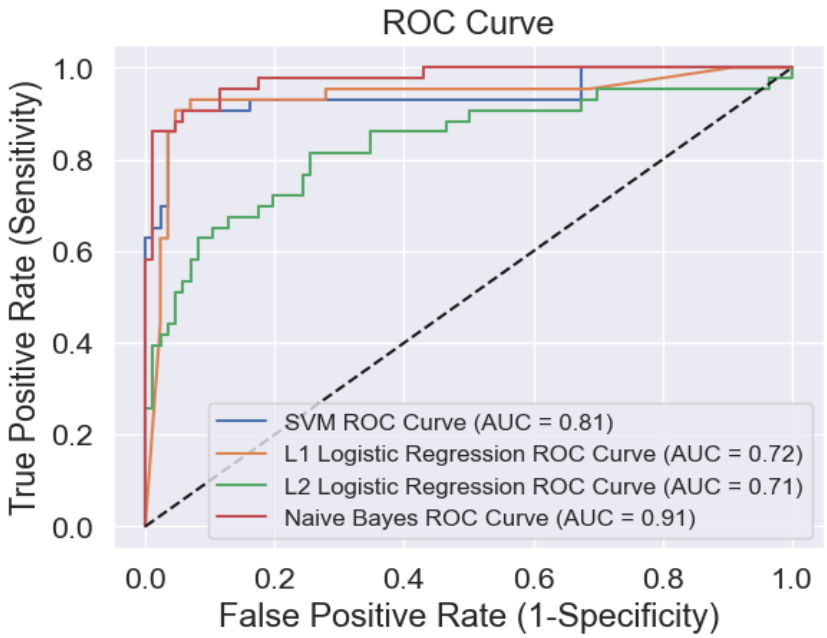


Figure 4.2.2.4-2

The ROC curves are plotted for the four models and computed their respective AUC values as shown in figure 4.2.2.4-2. Our findings reveal a well-balanced correlation between the true positive rate (TPR) and the false positive rate (FPR) for each category. Additionally, we observed that Naive Bayes achieved the highest AUC value of 0.91, whereas SVM yielded an AUC value of 0.81.

However, we should also be aware of the characteristics of ROC and AUC. In the Precision mentioned above, Recall and F1 Score, the model's performance is relatively poor due to data imbalance. On the contrary, ROC and AUC are not sensitive to the imbalanced data distribution between different categories because the ROC curve is based on TPR and FPR, which are independent of class distribution so the AUC of the four models are relatively high.

4.3 Logistic Regression

Logistic regression is a classification technique which uses supervised learning for the prediction of a categorical, dependent variable. Logistic regression can classify test data easily and efficiently for model training with linearly separable dataset, with the model coefficients acting as measures for interpreting feature importance (Raj, 2020). In sections 4.3.1 and 4.3.2, regularization parameters are implemented to the logistic regression functions with regards to the lasso (L1) and ridge (L2) logistic regressions in order to control the magnitude of the weight

coefficients, as well as lowering the complexity of the model by imposing penalties to the logistic regression functions to prevent overfitting and improving the performance of the models. Similar to Naive Bayes, the OVR (one-versus-rest) approach is employed by dividing the multiclass classification into multiple binary classification models. In the scikit-learn library, the hyperparameter C - inverse of regularization strength, is used for controlling the regularization strength, and evaluating the accuracy of the model at different regularization strengths logarithmically. The default value of C is 1.0 programmatically. In this project, the value of C is chosen as 0.1 as it is an optimal value for greater regularization, which can enable the reduction of the model complexity as represented in the program codes and the lasso and ridge logistic regression plots. Setting the value of C too large may lower the accuracy and performance of the model. The model complexity may skyrocket with high variance, with excessive adaptation to the training data and generalize poorly on new, unseen test data (Collimator, n.d.).

4.3.1 L1 Logistic Regression - LASSO

LASSO (Least Absolute Shrinkage and Selection Operator) is used for the estimation of the regression parameters. The formula of L1 regularization below is represented as the sum of the absolute values of the regression coefficients. The regression coefficients consist of the log-likelihood - the sum of least square residuals (RSS), and the log prior - Laplacian prior of regression coefficients which acts as the penalty function.

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

$$= RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

Figure 4.3.1-1 formula of lasso logistic regression (Sakia, 2021)

L1 regularization is advantageous by encouraging sparse, fewer parameters, lowering the irrelevant and unimportant features to zero. By performing feature selection automatically, this lowers the model complexity and enhances the accuracy and interpretability of test data. However, with the addition of the penalty term with the absolute values of coefficients, outliers can significantly raise the values of penalty, lowering the performance of the model in

generalizing test data. In this plot below, different values of accuracies are compared along with different values of C, with the parameter of C set as 0.1 as mentioned in section 4.3. The testing and training accuracy are both really high, indicating that the model always predicts the correct label in figure 4.3.1-3.

```
# L1 Logistic Regression for embedded feature selection
# https://www.youtube.com/watch?v=_aGWjt7GKBE
# https://necromuralist.github.io/student_intervention/choosing_best_model.html#:~:text=C%20is%20the%20inverse%20of,the%20coeffic
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)

lr1 = LogisticRegression(penalty='l1', C=0.1, solver='liblinear', multi_class='ovr')
# Note that C=1.0 is the default. You can increase
# or decrease it to make the regulariztion effect
# stronger or weaker, respectively.
lr1.fit(X_train_std, y_train)
print('Training accuracy:', lr1.score(X_train_std, y_train))
print('Test accuracy:', lr1.score(X_test_std, y_test))
```

Figure 4.3.1-2

```
Training accuracy: 0.8520710059171598
Test accuracy: 0.9069767441860465
```

Figure 4.3.1-3

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

# normalize the data
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# train Logistic Regression models for different values of C
# and collect train and test accuracies
scores = {}
for C in (10**k for k in range(-6, 6)):
    lr1 = LogisticRegression(penalty='l1', C=C, solver='liblinear', multi_class='ovr')
    lr1.fit(X_train, y_train)
    scores[C] = {'train accuracy': lr1.score(X_train, y_train),
                 'test accuracy': lr1.score(X_test, y_test)}

# plot the accuracy scores for different values of C
pd.DataFrame.from_dict(scores, 'index').plot(logx=True, xlabel='C (inverse of regularization strength)', ylabel='accuracy');
```

Figure 4.3.1-4

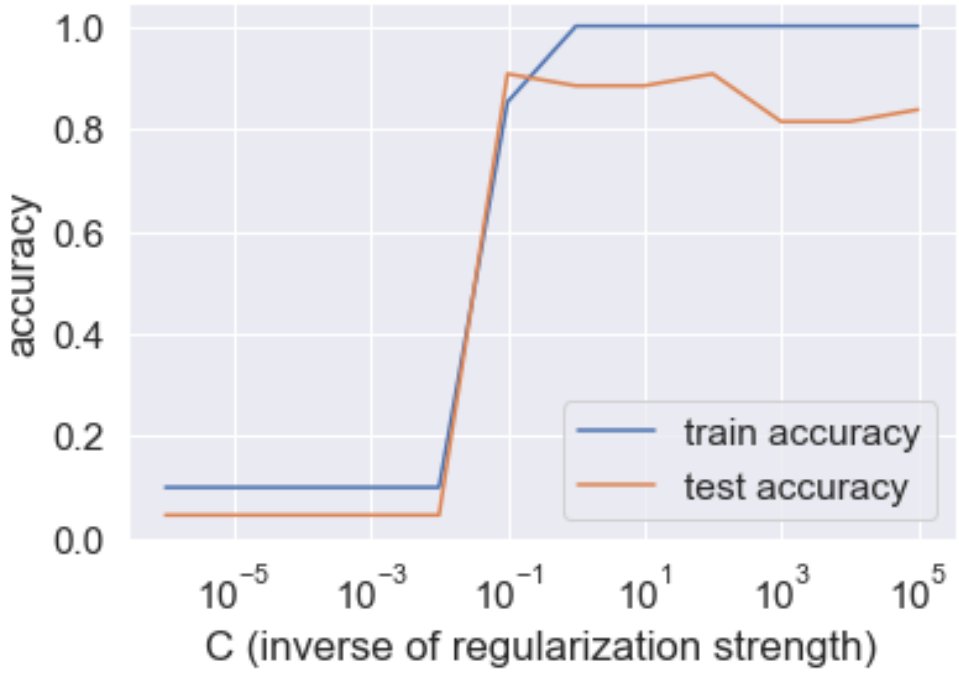


Figure 4.3.1-5 lasso plot with respect to different C and different accuracies

In figure 4.3.1-5, the training and testing accuracy of the model increases along with the increase in the inverse of regularization strength (decrease of regularization strength). In particularly, there is a steep, positive slope when C is approaching to 10^{-1}. Overall, with the resulting high accuracy of at least 0.8 along with increase in C, the model generalizes well for unseen data. The accuracy of the model can be improved by increasing the size of data used for training and testing.

4.3.2 L2 Logistic Regression - Ridge

Ridge is used for the estimation of the regression parameters. The formula of L2 regularization below is represented as the sum of the square values of the regression coefficients. The regression coefficients consist of the log-likelihood - the sum of least square residuals (RSS), and the log prior - Gaussian prior of regression coefficients which acts as the penalty function.

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j^2$$

$$= RSS + \lambda \sum_{j=1}^{p}\beta_j^2$$

Figure 4.3.2-1 formula of ridge logistic regression (Sakia, 2021)

L2 regularization is advantageous by lowering the standard error with the addition of bias in the regression estimates. This also handles multicollinearity and acts less sensitively against outliers. by adding some bias (penalty weights) in the regression estimates to lower the model complexity, enhances the accuracy and interpretability of test data. It also handles multicollinearity and is less sensitive to outliers. In this plot below, different values of accuracies are compared along with different values of C. The testing and training accuracy are both high, indicating that the model often predict the correct label in figure 4.3.1-3.

```
# L2 Logistic Regression for embedded feature selection
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)

lr2 = LogisticRegression(penalty='l2', C=0.1, solver='liblinear', multi_class='ovr')
# Note that C=1.0 is the default. You can increase
# or decrease it to make the regulariztion effect
# stronger or weaker, respectively.
lr2.fit(X_train_std, y_train)
print('Training accuracy:', lr2.score(X_train_std, y_train))
print('Test accuracy:', lr2.score(X_test_std, y_test))
```

Figure 4.3.2-2

```
Training accuracy: 1.0
Test accuracy: 0.6744186046511628
```

Figure 4.3.2-3

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

# normalize the data
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# train Logistic Regression models for different values of C
# and collect train and test accuracies
scores = {}
for C in (10**k for k in range(-6, 6)):
    lr2 = LogisticRegression(penalty='l2', C=C, solver='liblinear', multi_class='ovr')
    lr2.fit(X_train, y_train)
    scores[C] = {'train accuracy': lr2.score(X_train, y_train),
                 'test accuracy': lr2.score(X_test, y_test)}

# plot the accuracy scores for different values of C
pd.DataFrame.from_dict(scores, 'index').plot(logx=True, xlabel='C (inverse of regularization strength)', ylabel='accuracy');
```

Figure 4.3.2-4

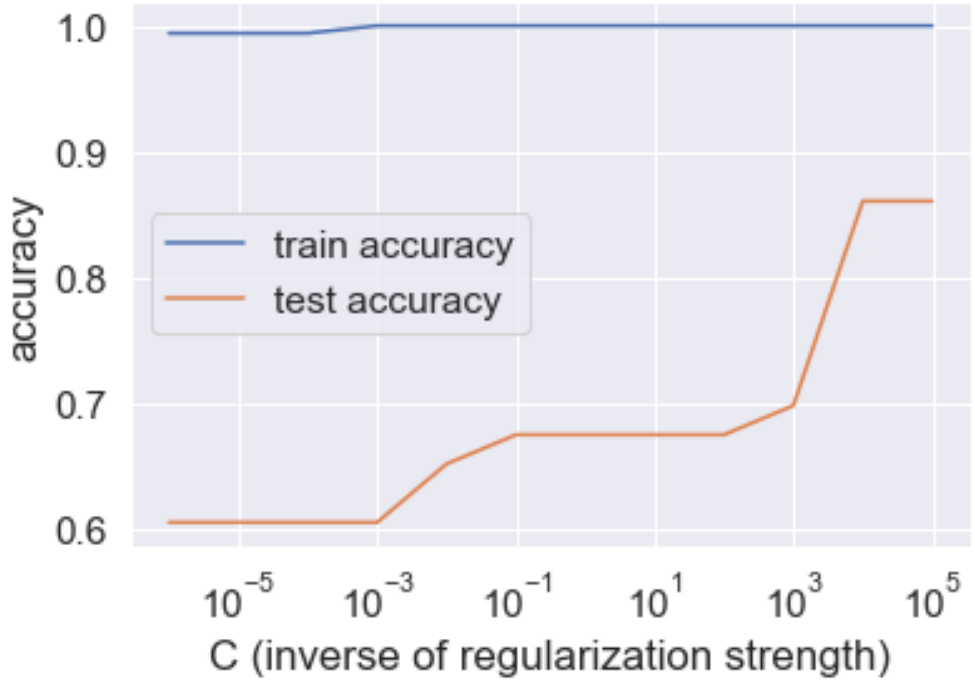


Figure 4.3.2-5 ridge plot with respect to different C and different accuracies

In figure 4.3.2-5, the testing accuracy remains at or near 1.0, indicating that the model always predicts the correct label. The testing accuracy of the model increases generally along with the increase in the inverse of regularization strength (decrease of regularization strength). In particularly, there is a steep, positive slope for the testing accuracy when C is approaching to 10^3. Overall, with the resulting high accuracy of at least 0.8 along with increase in C, the model generalizes well for unseen data. The accuracy of the model can be improved by increasing the size of data used for training and testing.

5 Conclusion and Limitation

In sentiment analysis, there are many properties that can be identified that enhance the investigation of the English Youtube comments in this project. Throughout the investigation and analysis of this problem using confusion matrix, L1 and L2 logistic regressions, Support Vector Machine classification, they show the issue of imbalanced class based on the low values of performance metrics, i.e. precision, recall, F1 score. There are also outliers identified that can boost the difference between regression and penalization in L1 and L2 regularization. These models provide quantitative insights into different social media users. Outliers create inflated penalties that affect model performance for L1 and L2 regularization.

When performing sentiment analysis, we also encountered the following limitations and made corresponding suggestions. Firstly, the model may not cover comments in all language backgrounds. To address this issue, expanding the dataset to include more samples in different languages is recommended to better cope with the diversity of reviews. Next, The significance of data volume in training a high-performing model. Hence, it is recommended to collect additional data and conduct more training to enhance the model's accuracy and generalization capabilities. The models may be too complex, leading to overfitting and increased computational cost. It is recommended to reduce the complexity of the model, which can be simplified by reducing the number of layers and the number of neurons. To enhance the overall performance of the predictive model, it is suggested to utilize ensemble learning methods - such as voting and averaging for classification and regression methods respectively. These methods can effectively combine predictions from multiple models, resulting in more accurate and robust predictions. Extra information on the standardization steps should be provided to enhance consistency and accuracy of this investigation.

6 Reference List

Collimator. (n.d.). *What are regularization techniques?* Medium. https://pub.towardsai.net/ridge-and-lasso-regression-51705b608fb9

Goyal, C. (n.d.). *Part 3: Step by Step Guide to NLP – Text Cleaning and Preprocessing*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/part-3-step-by-step-guide-to-nlp-text-cleaning-and-preprocessing/

Goyal, C. (n.d.). *Part 4: Step by Step Guide to Master NLP – Text Cleaning Techniques*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/part-4-step-by-step-guide-to-master-natural-language-processing-in-python/

HKMU. (2023). *COMP S460F Ch-5 Kernel Methods and Support Vector Machine*. HKMU.

Huilgol, P. (n.d.). Essential Metrics for Machine Learning (2023 Update). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/

Jones, C. (2014). *Estimation And Variable Selection With Ridge Regression And The Lasso*. Business Forecasting. https://businessforecastblog.com/estimation-and-variable-selection-with-ridge-regression-and-the-lasso/

Kanstrén, T. (2020). A Look at Precision, Recall, and F1-Score. Towards Data Science. https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec

Kundu, R. (2022). *F1 score in Machine Learning: Intro & Calculation*. V7. https://www.v7labs.com/blog/f1-score-guide

Madan, R. (2019). *TF-IDF/Term Frequency Technique: Easiest explanation for Text classification in NLP using Python (Chatbot training on words)*. Medium. https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

Saini, A. (n.d.). *Guide on Support Vector Machine (SVM) Algorithm*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/#h-what-is-a-support-vector-machine

Sakia, D.N. (2021). *Ridge and Lasso Regression*. Medium. https://pub.towardsai.net/ridge-and-lasso-regression-51705b608fb9

Sakib, A.S. (2022). *Scrape Youtube Comments For Free (No Google API)*. Kaggle. https://www.kaggle.com/code/ahmedshahriarsakib/scrape-youtube-comments-for-free-no-google-api/output

Scikit-learn. (n.d.). *sklearn.svm.SVC*. Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Singh, V. (n.d.). *ROC-AUC vs Accuracy: Which Metric Is More Important?* ShikSha Online. https://www.shiksha.com/online-courses/articles/roc-auc-vs-accuracy/#:~:text=AUC%20is%20considered%20a%20better,threshold%20that%20determines%20the%20accuracy.