**Sentiment Analysis for Amazon Musical Instruments User Reviews**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
|  | **INTRODUCTION** |  |
|  | * 1. General Introduction |  |
|  | * 1. Project Objectives |  |
|  | * 1. Problem Statement |  |
|  | **SYSTEM** **PROPOSAL** |  |
|  | * 1. Existing System |  |
|  | * + 1. Disadvantages |  |
|  | * 1. Proposed System |  |
|  | 2.2.1 Advantages |  |
|  | * 1. Literature Survey |  |
|  | **SYSTEM DIAGRAMS** |  |
|  | * 1. Architecture Diagram |  |
|  | * 1. Flow Diagram |  |
|  | * 1. UML Diagrams |  |
|  | **IMPLEMENTATION** |  |
|  | * 1. Modules |  |
|  | * 1. Modules Description |  |
|  | **SYSTEM** **REQUIREMENTS** |  |
|  | * 1. Hardware Requirements |  |
|  | * 1. Software Requirements |  |
|  | * 1. Software Description |  |
|  | * 1. Testing of Products |  |
|  | **CONCLUSION** |  |
| **7.** | **FUTURE** **ENHANCEMENT** |  |
| **8.** | **SAMPLE** **CODING** |  |
| **9.** | **SAMPLE** **SCREENSHOT** |  |
| **10.** | **REFERENCES** |  |

**Abstract**

Sentiment analysis is a rapidly emerging domain in the area of research in the field of Natural Language Processing (NLP). It has gained much attention in recent years. Sentiment classification is used to verify or analyze the comments given by the user to extract the opinion from it. Sentiment analysis is a machine learning approach in which machines classify and analyze the human’s sentiments, ratings, opinions etc. about the products which are expressed in the form of text, star rating, thumbs up and thumbs down. The instrumentation review data used in this process is online product reviews collected from the sample website that we have created. Words such as adjectives and adverbs are able to convey opposite sentiment with the help of negative prefixes. Negation phrase identification algorithm is used to find such words. The performance is evaluated through evaluation measures like Accuracy, precision, recall and F1-score.

**CHAPTER 1**

**INTRODUCTION**

* 1. **General Introduction:**

Sentiment analysis is often referred to as opinion mining, because the opinion collected from the customer will be mined to reveal the rating of the product. It comes under machine learning. Since the online data’s are tremendously growing data-by-day, it is considered to be very important in the current situation because, lots of user opinionated texts are available in the web now. Sentiment analysis is considered to be the study of user’s thought and feeling towards a product. Both SA and OM are interchangeable. The importance of the sentiment analysis or opinion mining is increasing day by day, as data grows day by day. Machines must be reliable and efficient to interpret and understand human emotions and feelings.

Sentiment analysis is a machine learning tool which is used for analyze the texts for polarity from positive to negative. Machine automatic learn how to analyze the sentiment of the human without the human input or interruption. Nowadays social media is a part of the people’s life; people uses social media for give their review over some political field, movie review or marketing area. There are many social media sites like Twitter, Facebook, Instagram etc. They use this social media sites as the medium to express their view on many topics. So, sentiment analysis analyzes the text which inputted by any person from the different country by using the training data set it will analyze the sentiment of that particular text by knowing the emotion of that people.

* 1. **Problem Statement**
* The problem in sentiment analysis is **classifying the polarity Negative or Positive.**
* The tweets are highly unstructured.

**1.3 Project Objectives**

* The problem in sentiment analysis to classify the tweet positive or negative.
* NLP used to clean the text.
* Using pipeline connection is used to increases the performances.
* Whether the review text is positive, negative, or neutral.
* To implement Machine Learning Algorithm to find Accuracy.

**CHAPTER 2**

**SYSTEM PROPOSAL**

**2.1 EXISTING SYSTEM**

Existing System mainly implements sentiment classification by mining language symbols information in online product site. Few studies consider sentiment information contained in emoticons symbols and punctuation symbols in online product website. Emoticons symbols and punctuation symbols containing sentiment information are important sources of sentiment information, and have important value in solving the lack of text context semantics. Existing System mainly extracts sentiment features and models text content from document level, but seldom consider extracting sentiment features and modeling text content from multiple levels such as words, sentences and documents. By extracting sentiment features and modeling text content from multiple levels, we can extract sentiment information from different levels better, and provide richer information for solving the lack of text context semantics in online product website.

**2.1.1 DISADVANTAGES:**

* The results is low when compared with proposed.
* Time consumption is high.
* Existing system find only the market value.
* Theoretical limits.
  1. **PROPOSED SYSTEM:**

Our Proposed System – Vedar sentiment analysis is implements and analyse the polarity of the text. Then, we can create the sentiment labels based on compound score. The pipeline connection between machine learning model and tfidf vectorizer is used to efficiently analyse the text data. There is no necessary to clean the text data.

**2.2.1 ADVANTAGES:**

* It is efficient for large number of datasets.
* The experimental result is high when compared with existing system.
* Time consumption is low.
* Provide accurate prediction results

**2.3 LITERATURE SURVEY:**

# **2.3.1 How the news media activate public expression and influence national agendas**

**Author***:* Hatoon AlSagri, Mourad Ykhlef

**Methodology**

We demonstrate that exposure to the news media causes Americans to take public stands on specific issues, join national policy conversations, and express themselves publicly—all key components of democratic politics—more often than they would otherwise. After recruiting 48 mostly small media outlets, we chose groups of these outlets to write and publish articles on subjects we approved, on dates we randomly assigned. We estimated the causal effect on proximal measures, such as website pageviews and Twitter discussion of the articles’ specific subjects, and distal ones, such as national Twitter conversation in broad policy areas. Our intervention increased discussion in each broad policy area by ~62.7% (relative to a day’s volume), accounting for 13,166 additional posts over the treatment week, with similar effects across population subgroups.

**Advantages:**

* It Can handle high amount of data.

**Disadvantages**:

* It performs trained on review data are **often much less accurate**

# **2.3.2 Selective exposure shapes the Facebook news diet**

### **Author**: **Ana Lucia Schmidt, Matteo Cinelli**

**Methodology**

The social brain hypothesis approximates the total number of social relationships we are able to maintain at 150. Similar cognitive constraints emerge in several aspects of our daily life, from our mobility to the way we communicate, and might even affect the way we consume information online. Indeed, despite the unprecedented amount of information we can access online, our attention span still remains limited. Furthermore, recent studies have shown that online users are more likely to ignore dissenting information, choosing instead to interact with information adhering to their own point of view. In this paper, we quantitatively analyse users’ attention economy in news consumption on social media by analysing 14 million users interacting with 583 news outlets (pages) on Facebook over a time span of six years. In particular, we explore how users distribute their activity across news pages and topics. On the one hand, we find that, independently of their activity, users show a tendency to follow a very limited number of pages. On the other hand, users tend to interact with almost all the topics presented by their favoured pages. Finally, we introduce a taxonomy accounting for users’ behaviour to distinguish between patterns of selective exposure and interest. Our findings suggest that segregation of users in echo chambers might be an emerging effect of users’ activity on social media and that selective exposure—i.e. the tendency of users to consume information adhering to their preferred narratives—could be a major driver in their consumption patterns.

**Advantages:**

* It performs accurate classification comparison with other methods**.**.

**Disadvantages**:

* Run to failure prediction is low.

# **2.3.3 A Survey of Sentiment Analysis from Social Media Data**

# **Author**: Siddhartha Bhattacharyya, Rajib Bag

**Methodology**

In the current era of automation, machines are constantly being channelized to provide accurate interpretations of what people express on social media. The human race nowadays is submerged in the idea of what and how people think and the decisions taken thereafter are mostly based on the drift of the masses on social platforms. This article provides a multifaceted insight into the evolution of sentiment analysis into the limelight through the sudden explosion of plethora of data on the internet. This article also addresses the process of capturing data from social media over the years along with the similarity detection based on similar choices of the users in social networks. The techniques of communalizing user data have also been surveyed in this article. Data, in its different forms, have also been analyzed and presented as a part of survey in this article. Other than this, the methods of evaluating sentiments have been studied, categorized, and compared, and the limitations exposed in the hope that this shall provide scope for better research in the future.

**Advantages:**

* More Reliable.

**Disadvantages**:

* It is less in efficiency and not give perfect result.

# **2.3.4 Social context in sentiment analysis: Formal definition, overview of current trends and framework for comparison**

# **Author**: J. Fernando Sánchez-Rada

**Methodology**

Sentiment analysis in social media is harder than in other types of text due to limitations such as abbreviations, jargon, and references to existing content or concepts. Nevertheless, social media provides more information beyond text, such as linked media, user reactions, and relations between users. We refer to this information as social context. Recent works have successfully leveraged the fusion of text with social context for sentiment analysis tasks. However, these works are usually limited to specific aspects of social context, and there have not been any attempts to analyze and apply social context systematically. This work aims to bridge this gap by providing three main contributions a formal definition of social context a framework for classifying and comparing approaches that use social context a review of existing works based on the defined framework

**Advantages:**

* Low Cost

**Disadvantages**:

* inadequate accuracy and performance.

# **2.3.5 A Review of Affective Computing: From Unimodal Analysis to Multimodal Fusion**

# **Author**: Soujanya Poria1, Erik Cambria , Rajiv Bajpai , Amir Hussain

# **Methodology**

Affective computing is an emerging interdisciplinary research field bringing together researchers and practitioners from various fields, ranging from artificial intelligence, natural language processing, to cognitive and social sciences. With the proliferation of videos posted online (e.g., on YouTube, Facebook, Twitter) for product reviews, movie reviews, political views, and more, affective computing research has increasingly evolved from conventional unimodal analysis to more complex forms of multimodal analysis. This is the primary motivation behind our first of its kind, comprehensive literature review of the diverse field of affective computing. Furthermore, existing literature surveys lack a detailed discussion of state of the art in multimodal affect analysis frameworks, which this review aims to address. Multimodality is defined by the presence of more than one modality or channel, e.g., visual, audio, text, gestures, and eye gage. In this paper, we focus mainly on the use of audio, visual and text information for multimodal affect analysis, since around 90% of the relevant literature appears to cover these three modalities. Following an overview of different techniques for unimodal affect analysis, we outline existing methods for fusing information from different modalities.

**Advantages**:

* It is efficient and scalable method
* The execution of various classifiers for depression detection in a shorter time.

**Disadvantages**:

* The result accuracy is low.

# **2.3.6 Deep learning for sentiment analysis: A survey**

# **Author**: Shuai Wang, Bing Liu

# **Methodology**

Deep learning has emerged as a powerful machine learning technique that learns multiple layers of representations or features of the data and produces state-of-the-art prediction results. Along with the success of deep learning in many application domains, deep learning is also used in sentiment analysis in recent years. This paper gives an overview of deep learning and then provides a comprehensive survey of its current applications in sentiment analysis.

**Advantages:**

* More Reliable

**Disadvantages**:

* This method can not only take a long test time and have high cost.

# **2.3.7 A Method for Ranking Products Through Online Reviews Based on Sentiment Classification and Interval-Valued Intuitionistic Fuzzy TOPSIS**

# **Author**: Yang Liu, Jian-Wu Bi

# **Methodology**

Studies have shown that online product reviews significantly affect consumer purchase decisions. However, it is difficult for the consumer to read online product reviews one by one because the number of online reviews is very large. Thus, to facilitate consumer purchase decisions, how to rank products through online reviews is a valuable research topic. This paper proposes a method for ranking products through online reviews based on sentiment classification and the interval-valued intuitionistic fuzzy Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS). The method consists of two parts: (1) identifying sentiment orientations of the online reviews based on sentiment classification and (2) ranking alternative products based on interval-valued intuitionistic fuzzy TOPSIS. In the first part, the online reviews of the alternative products concerning multiple attributes are preprocessed, and an algorithm based on support vector machine and one-versus-one strategy is developed for classifying the sentiment orientations of online reviews into three categories: positive, neutral, and negative. In the second part, based on the percentages of the online reviews with different sentiment orientations and the numbers of online reviews of different products crawled from the website, an interval-valued intuitionistic fuzzy number is constructed to represent the performance of an alternative product with respect to the product attribute. Additionally, the interval-valued intuitionistic fuzzy TOPSIS method is employed to determine a ranking of the alternative products. Finally, a case analysis is provided to illustrate the application of the proposed method.

**Advantages:**

* To higher accuracy and prevents the problem of overfitting.

**Disadvantages**:

* .Decrease training size in sampling methods.

**CHAPTER 3**

**SYSTEM DIAGRAMS**

**3.1 SYSTEM ARCHITECTURE:**

SELECT AND VIEW DATASET

PRE-PROCESSING

VADER SENTIMENTAL ANALYSIS

DATA SPLITTING

FEATURE EXTRACTION

CLASSIFICATION

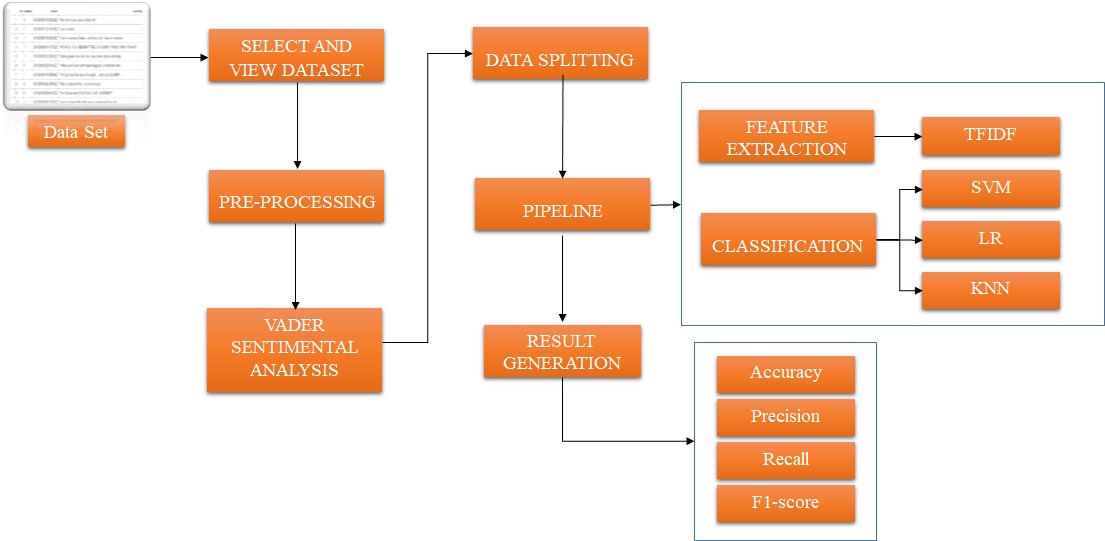
(SVM, LR, KNN)

Data Set



RESULT GENERATION

**3.2 Flow Diagram**

****

**3.3 Use Case Diagram**

**Performance Analysts**

**3.4 Activity Diagram**

Input Data

Preprocessing

Data splitting

Performance metrics

TFIDF Vectorization

NLP

SVM, LR, KNN algorithm

**3.5 Sequence Diagram**

DATA SELECTION & LOAD

DATA PREPROCESSING

Vader Sentiment

TFIDF VECTORIZATION

Select Dataset

Missing data Removal

Classification and Prediction

RESULT GENERATION

Select Best Features

Test

Find Sentiment Score

Train

DATA SPLITTING

**3.6. Class Diagram**

Select data ()

Load data ()

View data ()

INPUT

Train ()

Data Splitting

Test()

AAAgjhuhggkjgj AdaBoost()

DecisionTree()

daBhgduoost()

Decis AdaBoost()

DecisionTree()

ionTree()

bmjnkhjlkglkgffkllllllllljjggdsagdfghfdhdhhdfshhjpkhopngkjesrahgmjjfjkhjgyjtjfhgyjjgtf

Preprocessing

Missing values ()

Label encode ()

Vader Sentiment()

K,A

Algorithm

TFIDF Vectorization

Result Generation

SVM  
LR  
KNN

**3.7 ER DIAGRAM**

Classification report

Data Splitting

Data selection

Preprocessing

TFIDF Vectorization

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 MODULES:**

* Data Selection and Loading
* Data Preprocessing
* Data Splitting
* Feature Extraction
* Classification
* Prediction
* Result Generation
  1. **Modules Description**

**DATA SELECTION AND LOADING**

* The data selection is the process of selecting the data for product sentiment analysis.
* The dataset contains the information about review text.

**DATA PREPROCESSING**

* Data pre-processing is the process of removing the unwanted data from the dataset.
* Missing data removal

**SPLITTING DATASET INTO TRAIN AND TEST DATA**

* Data splitting is the act of partitioning available data into two portions, usually for cross-validate purposes.
* One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
* Separating data into training and testing sets is an important part of evaluating image processing models.
* Typically, when you separate a data set into a training set and testing set, most of the image data is used for training, and a smaller portion of the data is used for testing.

**FEATURE EXTRACTION**

TfidfVectorizer - Transforms text to feature vectors that can be used as input to estimator. Vocabulary is a dictionary that converts each token (word) to feature index in the matrix, each unique token gets a feature index. It tells you that the token 'me' is represented as feature number 8 in the output matrix.

**CLASSIFICATION**

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points

A Logistic Regression is type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure.

**PREDICTION**

* It’s a process of Detection of review sentiment from the dataset.
* This project will effectively detect the Dangerous webpages from dataset by enhancing the performance of the overall prediction results.

**RESULT GENERATION**

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

* Accuracy
* Precision
* Recall
* F1-score
* Specificity.

**CHAPTER 5**

**SYSTEM REQUIREMENTS**

**5.1 HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz
* Hard Disk : 200 GB
* Mouse : Logitech.
* Keyboard : 110 keys enhanced
* Ram : 4GB

**5.2 SOFTWARE REQUIREMENTS:**

* O/S : Windows 7.
* Language : Python
* Front End : Anaconda Navigator – Spyder

**5.3 SOFTWARE DESCRIPTION:**

**5.3.1 Python**

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

## **5.3.2 Features of Python**

### **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### **Free and Open Source**

Python is an example of a FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

### **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org) to create games for your computer and for iPhone, iPad, and Android.

### **Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

### **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

### **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

### **Embeddable**

You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.

### **Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

**5.4 TESTING PRODUCTS:**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

**5.4.1 UNIT TESTING:**

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’. The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

**5.4.2 INTEGRATION TESTING:**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

i) Top-down integration testing. ii) Bottom-up integration testing.

**5.4.3 TESTING TECHNIQUES/STRATEGIES:**

* **BLACK BOX TESTING:**

1. Black box testing is done to find incorrect or missing function
2. Interface error
3. Errors in external database access
4. Performance errors.
5. Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

WHITE BOX TESTING:

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

**5.4.4 SOFTWARE TESTING STRATEGIES**

**VALIDATION TESTING:**

* After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many,
* But a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer

**USER ACCEPTANCE TESTING:**

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

**OUTPUT TESTING**:

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

**CHAPTER 6**

**CONCLUSION**

To solve the problems in the existing sentiment classification methods of online product website, we propose a sentiment classification method of online product website based on pipeline modelling. Our methods have partly solved the lack of context semantics of online product website text by extracting multi-levels and multi-dimensions sentiment information. The experimental results show that our method greatly improves the accuracy of sentiment classification of online product website compared with the existing methods. Our Proposed ML Algorithm improves accuracy.

**CHAPTER 7**

**FUTURE ENHANCEMENT**

We will further explore a better network structure to achieve multi-dimensional and multi-level modelling of online product website. At the same time, we will explore sentiment classification method of online product website by fusing multi-modal information in online product website.

**CHAPTER 8**

**SAMPLE CODING**

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import pandas as pd

import nltk

import re

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfTransformer, TfidfVectorizer

from sklearn.svm import LinearSVC

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.pipeline import Pipeline

from sklearn import metrics

import scikitplot as skplt

#===================================INPUT======================================

df=pd.read\_csv("Musical\_instruments\_reviews.csv")#, encoding='latin-1')

print(df.head())

#=============================PRE\_PROCESSING===================================

#Data Cleaning

df['reviewText']=df['reviewText'].apply(str)

#============================VADER SENTIMENT===================================

#Find Sentiment score

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyser = SentimentIntensityAnalyzer()

scores=[]

for i in range(len(df['reviewText'])):

score = analyser.polarity\_scores(df['reviewText'][i])

score=score['compound']

scores.append(score)

sentiment=[]

for i in scores:

if i>=0.3:

sentiment.append('Positive')

elif i<=(-0.3):

sentiment.append('Negative')

else:

sentiment.append('Neutral')

df['sentiment']=pd.Series(np.array(sentiment))

#Count Plot

sns.countplot(x ='sentiment', data = df, palette = "Set2")

plt.show()

#============================Data processing===================================

nltk.download('words')

nltk.download('punkt')

nltk.download('maxent\_ne\_chunker')

nltk.download('stopwords')

stop\_words = nltk.corpus.stopwords.words('english')

#train['OriginalTweet'][0]

X = df['reviewText'].copy()

y = df['sentiment'].copy()

def data\_cleaner(text):

# remove urls

text = re.sub(r'http\S+', ' ', text)

# remove html tags

text = re.sub(r'<.\*?>',' ', text)

# remove digits

text = re.sub(r'\d+',' ', text)

# remove hashtags

text = re.sub(r'#\w+',' ', text)

# remove mentions

text = re.sub(r'@\w+',' ', text)

#removing stop words

text = text.split()

text = " ".join([word for word in text if not word in stop\_words])

return text

X\_cleaned = X.apply(data\_cleaner)

X\_cleaned.head()

#Label Encoding

from sklearn import preprocessing

label\_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.

y= label\_encoder.fit\_transform(y)

#==============================Data Splitting==============================

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

#==============================Feature-Extraction==============================

#convert a collection of text documents to a vector of term/token counts

count\_vect = CountVectorizer()

X\_train\_counts = count\_vect.fit\_transform(X\_train)

print(X\_train\_counts.shape)

#

tfidf\_transformer = TfidfTransformer()

X\_train\_tfidf = tfidf\_transformer.fit\_transform(X\_train\_counts)

print(X\_train\_tfidf.shape)

pd.DataFrame(X\_train\_tfidf)[0]

vectorizer = TfidfVectorizer()

X\_train\_tfidf = vectorizer.fit\_transform(X\_train)

print(X\_train\_tfidf.shape)

pd.DataFrame(X\_train\_tfidf)[0]

#==================================Classification==============================

'''Support Vector Machine'''

print('Support Vector Machine')

svm = LinearSVC()

svm.fit(X\_train\_tfidf,y\_train)

#pipe-line connection

svm\_text\_clf = Pipeline([('tfidf', TfidfVectorizer()),

('svm', LinearSVC()),])

svm\_text\_clf.fit(X\_train, y\_train)

svm\_pred = svm\_text\_clf.predict(X\_test)

svm=metrics.accuracy\_score(y\_test,svm\_pred)\*100

print('SVM Accuracy:',svm)

print('\nClassification Report')

svm\_cr=metrics.classification\_report(y\_test,svm\_pred)

print(svm\_cr)

print('\nConfusion Matrix')

svm\_cm=metrics.confusion\_matrix(y\_test,svm\_pred)

print(svm\_cm)

tp = svm\_cm[0][0]

fp = svm\_cm[0][1]

fn = svm\_cm[1][0]

tn = svm\_cm[1][1]

Total\_TP\_FP=svm\_cm[0][0]+svm\_cm[0][1]

Total\_FN\_TN=svm\_cm[1][0]+svm\_cm[1][1]

specificity = tn / (tn+fp)

specificity=format(specificity,'.3f')

print('SVM\_specificity:',specificity)

print()

plt.figure()

skplt.estimators.plot\_learning\_curve(LinearSVC(), X\_train\_tfidf, y\_train,

cv=7, shuffle=True, scoring="accuracy",

n\_jobs=-1, figsize=(6,4), title\_fontsize="large", text\_fontsize="large",

title="SVM Digits Classification Learning Curve");

plt.figure()

sns.heatmap(metrics.confusion\_matrix(y\_test,svm\_pred),annot = True)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("True")

plt.show()

'''Logistic Regression'''

print('Logistic Regression')

lr = LogisticRegression(solver='liblinear')

lr.fit(X\_train\_tfidf,y\_train)

#pipe-line connection

lr\_text\_clf = Pipeline([('tfidf', TfidfVectorizer()),

('lr', LogisticRegression(solver='liblinear')),])

lr\_text\_clf.fit(X\_train, y\_train)

lr\_pred = lr\_text\_clf.predict(X\_test)

lr=metrics.accuracy\_score(y\_test,lr\_pred)\*100

print('LR Accuracy:',lr)

print('\nClassification Report')

lr\_cr=metrics.classification\_report(y\_test,lr\_pred)

print(lr\_cr)

print('\nConfusion Matrix')

lr\_cm=metrics.confusion\_matrix(y\_test,lr\_pred)

print(lr\_cm)

tp = lr\_cm[0][0]

fp = lr\_cm[0][1]

fn = lr\_cm[1][0]

tn = lr\_cm[1][1]

Total\_TP\_FP=lr\_cm[0][0]+lr\_cm[0][1]

Total\_FN\_TN=lr\_cm[1][0]+lr\_cm[1][1]

specificity = tn / (tn+fp)

specificity=format(specificity,'.3f')

print('LR\_specificity:',specificity)

print()

plt.figure()

skplt.estimators.plot\_learning\_curve(LogisticRegression(solver='liblinear'), X\_train\_tfidf, y\_train,

cv=7, shuffle=True, scoring="accuracy",

n\_jobs=-1, figsize=(6,4), title\_fontsize="large", text\_fontsize="large",

title="Logistic Regression Digits Classification Learning Curve");

plt.figure()

sns.heatmap(metrics.confusion\_matrix(y\_test,lr\_pred),annot = True)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("True")

plt.show()

print()

print('K Nearest Neighbors')

knn = KNeighborsClassifier()

knn.fit(X\_train\_tfidf,y\_train)

#pipe-line connection

knn\_text\_clf = Pipeline([('tfidf', TfidfVectorizer()),

('knn', KNeighborsClassifier()),])

knn\_text\_clf.fit(X\_train, y\_train)

knn\_pred = knn\_text\_clf.predict(X\_test)

knn=metrics.accuracy\_score(y\_test,knn\_pred)\*100

print('KNN Accuracy:',knn)

print('\nClassification Report')

knn\_cr=metrics.classification\_report(y\_test,lr\_pred)

print(knn\_cr)

print('\nConfusion Matrix')

knn\_cm=metrics.confusion\_matrix(y\_test,knn\_pred)

print(knn\_cm)

tp = knn\_cm[0][0]

fp = knn\_cm[0][1]

fn = knn\_cm[1][0]

tn = knn\_cm[1][1]

Total\_TP\_FP=knn\_cm[0][0]+knn\_cm[0][1]

Total\_FN\_TN=knn\_cm[1][0]+knn\_cm[1][1]

specificity = tn / (tn+fp)

specificity=format(specificity,'.3f')

print('KNN\_specificity:',specificity)

print()

plt.figure()

skplt.estimators.plot\_learning\_curve(KNeighborsClassifier(), X\_train\_tfidf, y\_train,

cv=7, shuffle=True, scoring="accuracy",

n\_jobs=-1, figsize=(6,4), title\_fontsize="large", text\_fontsize="large",

title="KNeighborsClassifier Digits Classification Learning Curve");

plt.figure()

sns.heatmap(metrics.confusion\_matrix(y\_test,knn\_pred),annot = True)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

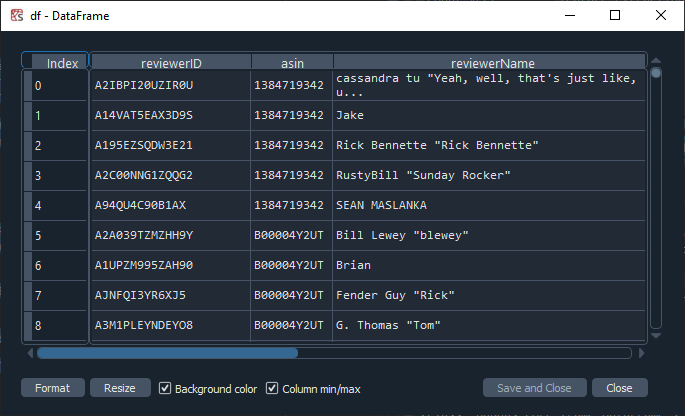
plt.ylabel("True")

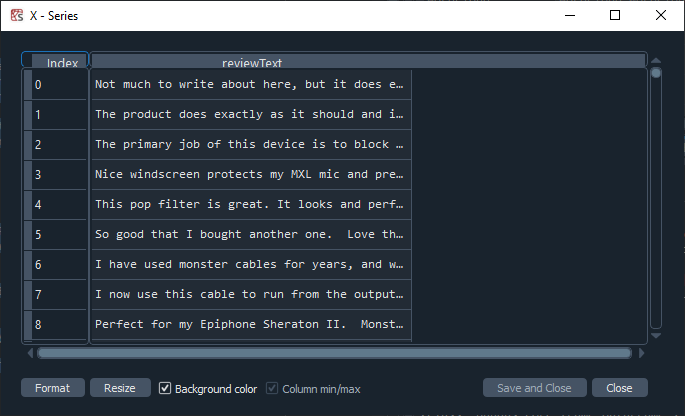
plt.show()

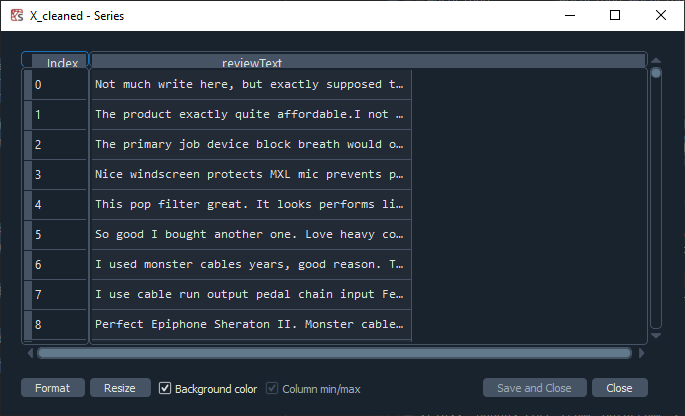
print()

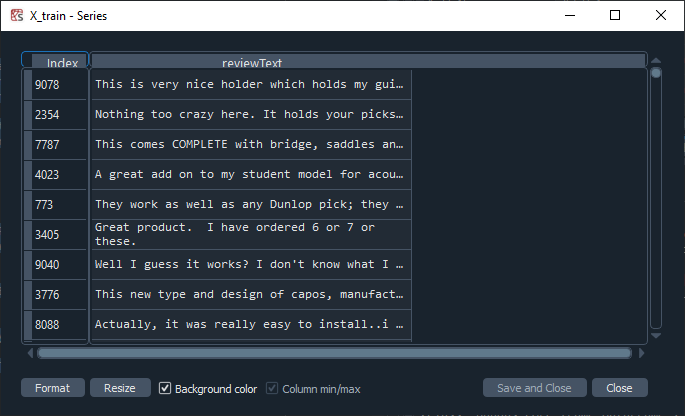
**CHAPTER 9**

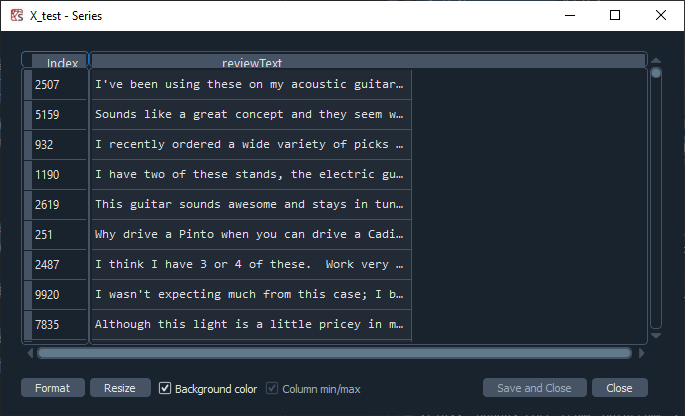
**SAMPLE SCREENSHOTS**

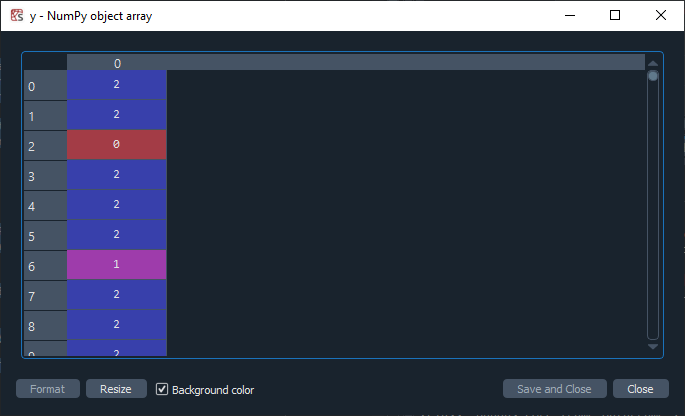


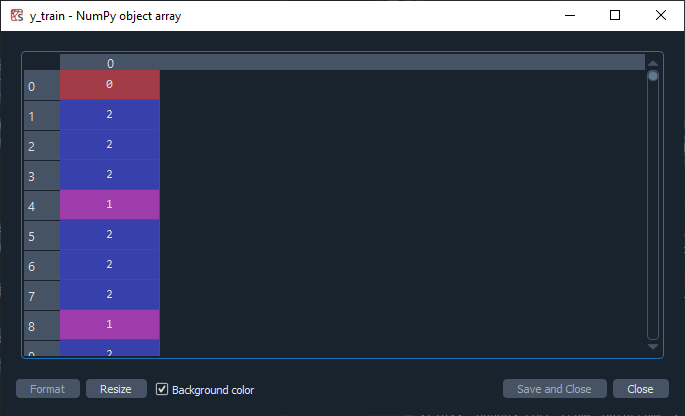


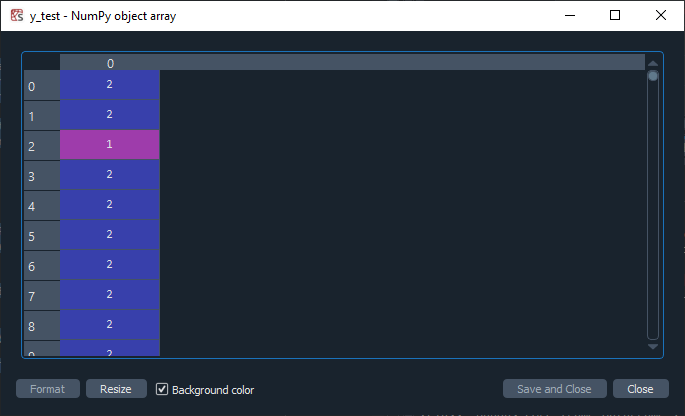


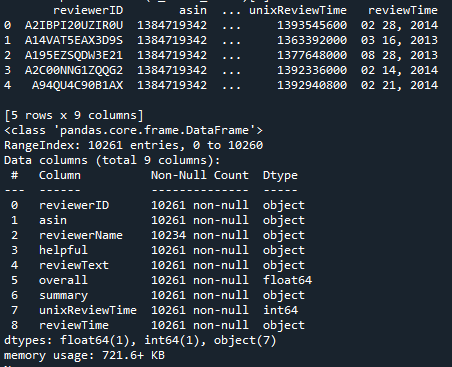


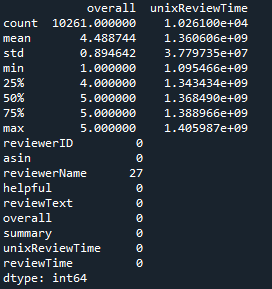


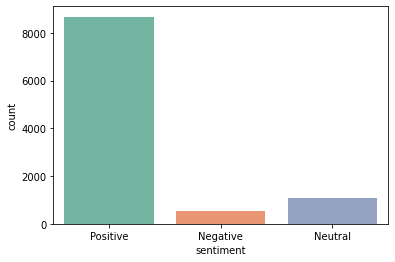


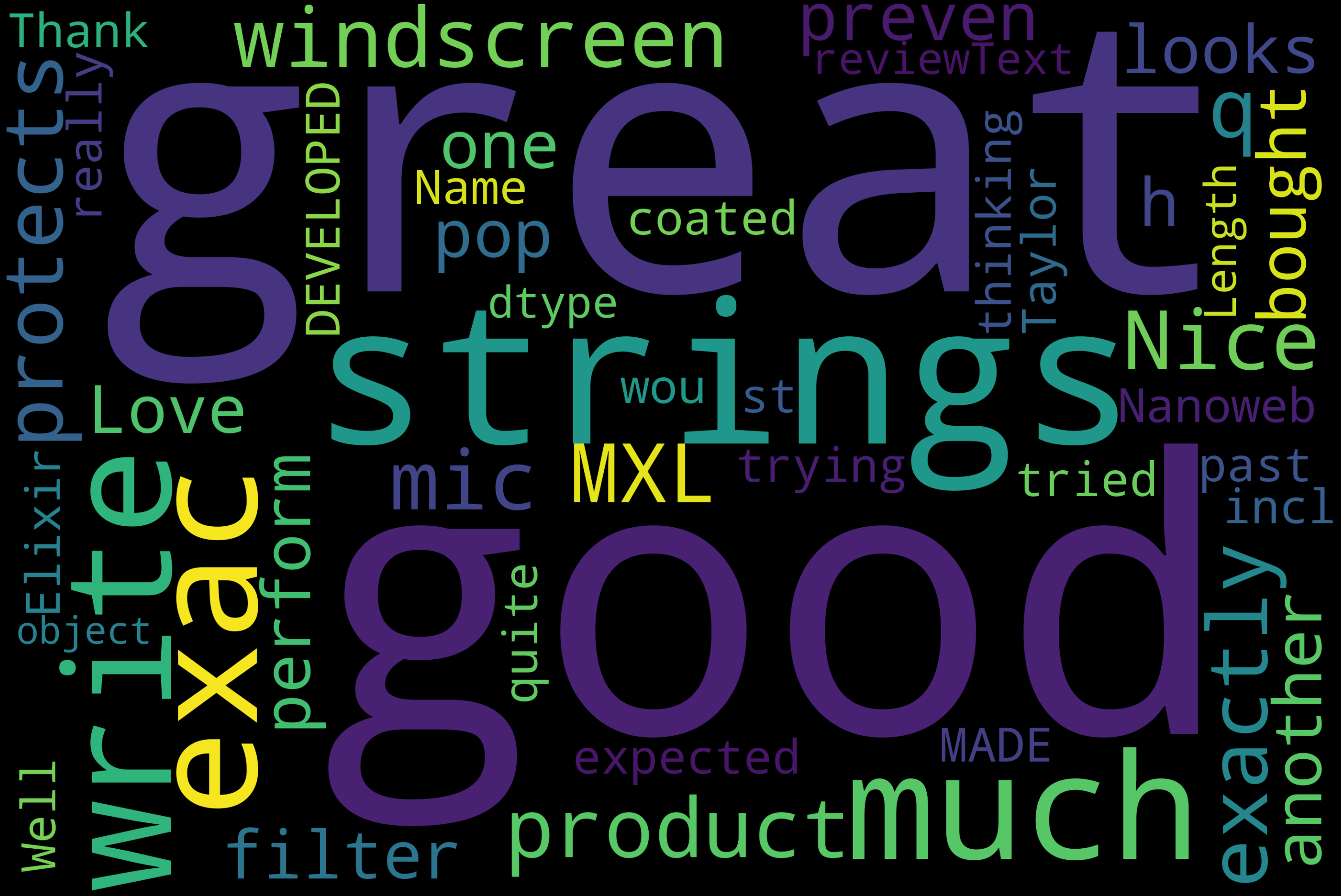


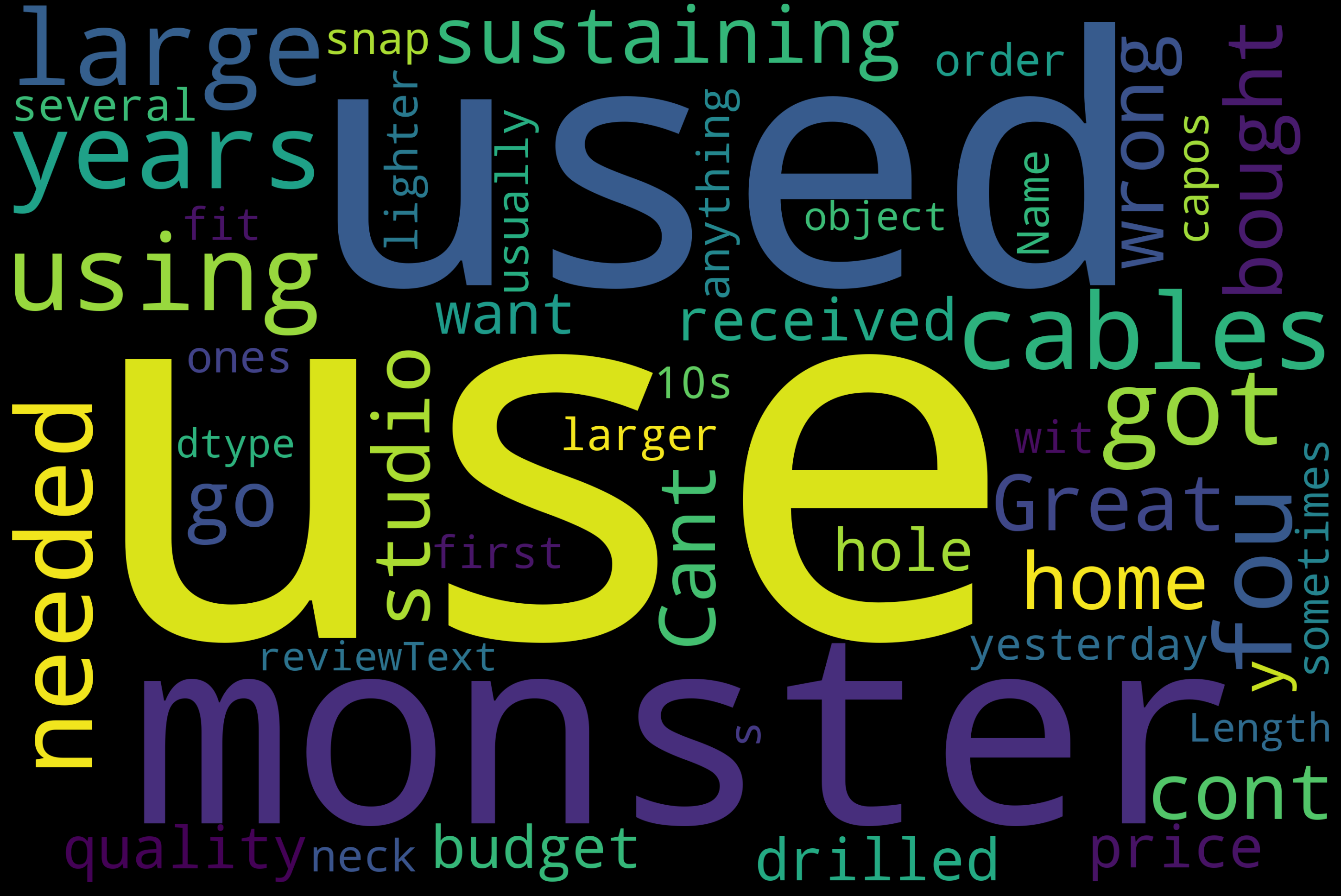


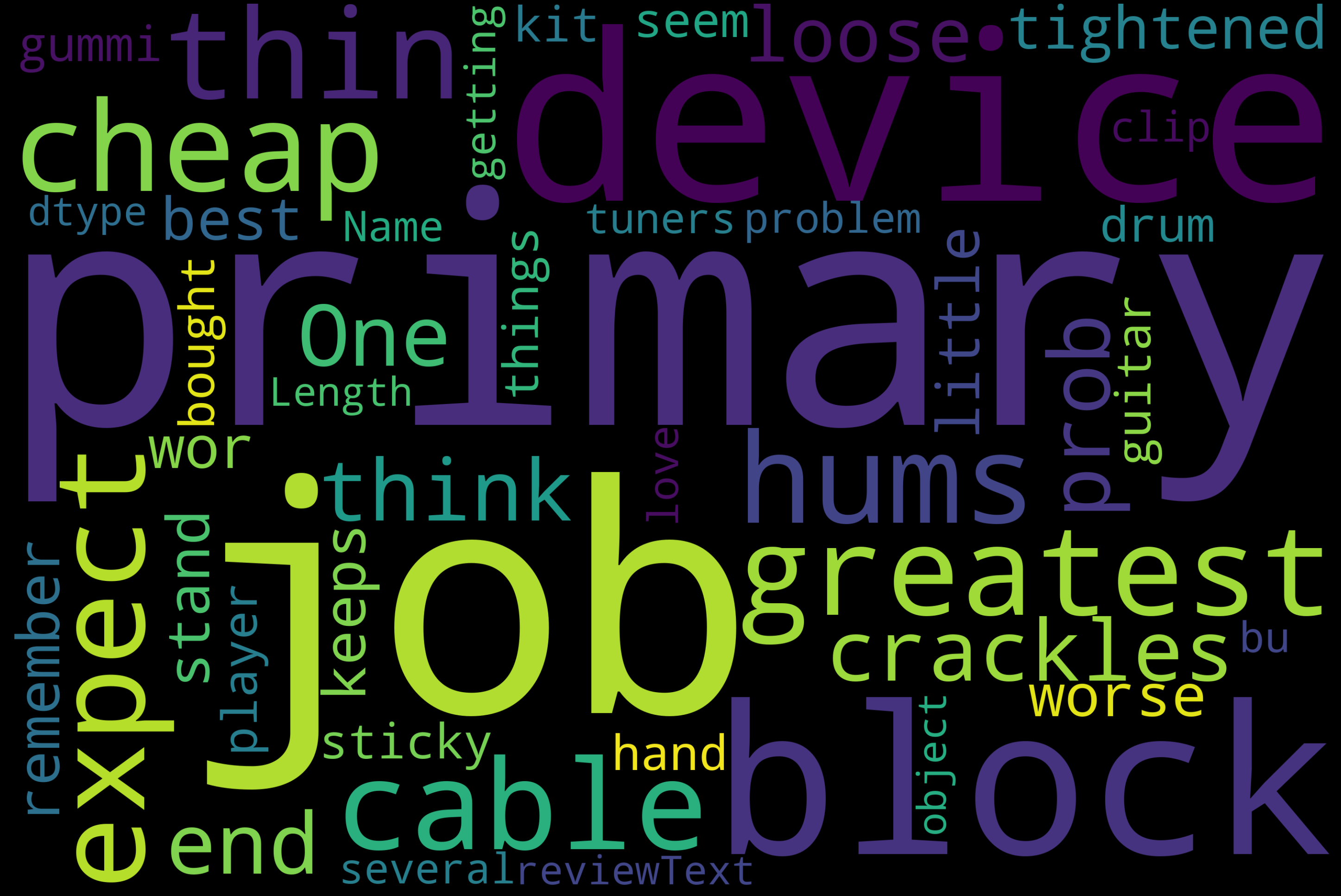


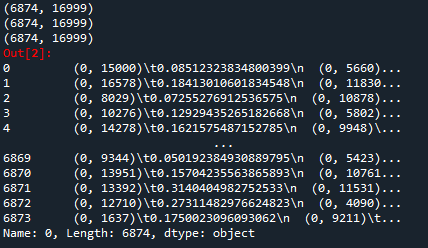


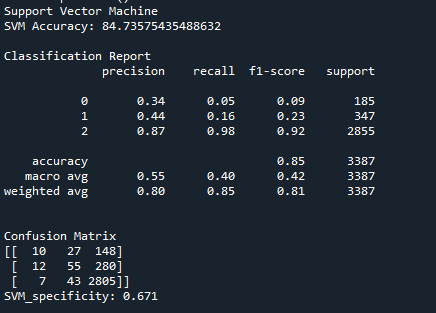


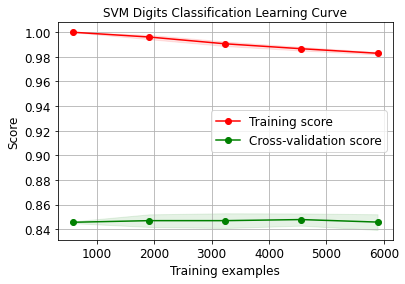


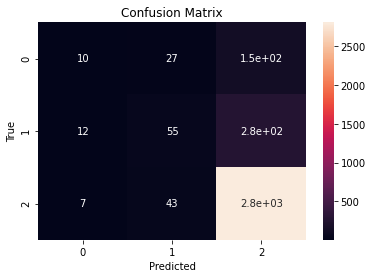


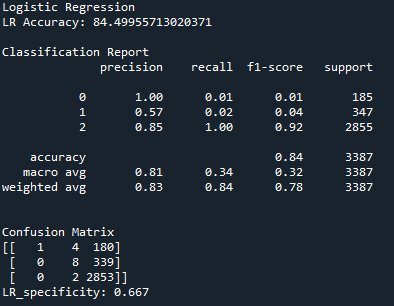


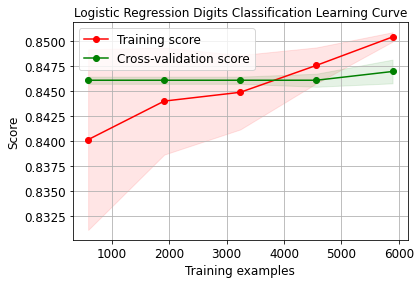


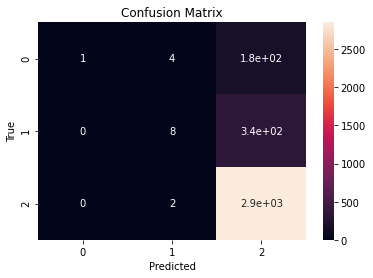


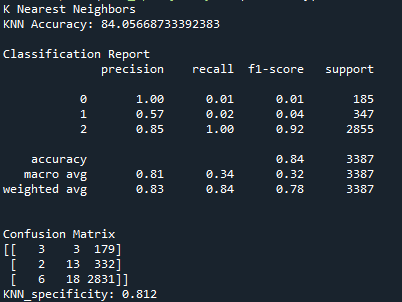


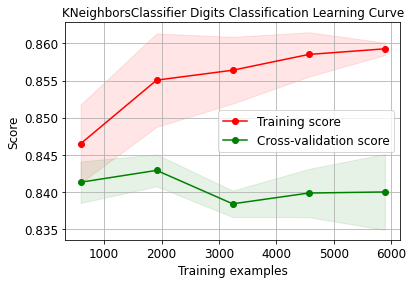


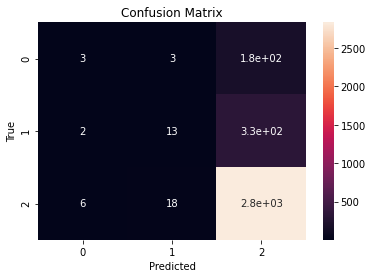












**CHAPTER 10**

**REFERENCES**

1. G. King, B. Schneer and A. White, “How the news media activate public expression and influence national agendas”, Science, vol. 358, no. 6364, pp.776-780, 2017, doi: 10.1126/science.aao1100. [
2. MD. Vicario, G. Vivaldo, A. Bessi, F. Zollo, A. Scala, G. Caldarelli and W. Quattrociocchi, “Echo Chambers: Emotional Contagion and Group Polarization on Facebook”, Scientific Reports, vol. 6, no.12, Dec. 2016, doi: 10.1038/srep37825.
3. K. Chakraborty, S. Bhattacharyya and R. Bag, "A Survey of Sentiment Analysis from Social Media Data", IEEE Transactions on Computational Social Systems, vol. 7, no. 2, pp. 450-464, April 2020, doi: 10.1109/TCSS.2019.2956957.
4. JF. Sánchez-Rada and CA. Iglesias, “Social context in sentiment analysis: Formal definition, overview of current trends and framework for comparison”, Information Fusion, vol. 52, no. 12, pp. 344-356, Dec. 2019, doi: 10.1016/j.inffus.2019.05.003.
5. S. Poria, E. Cambria, R. Bajpai and A. Hussain, “A review of affective computing: From unimodal analysis to multimodal fusion”, Information Fusion, vol. 37, no. 9 pp. 98-125, Sep. 2017, doi: 10.1016/j.inffus.2017.02.003.
6. L. Zhang, S. Wang, and B. Liu, “Deep learning for sentiment analysis: A survey”, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 8, no. 4, Aug. 2018, doi: 10.1002/widm.1253.
7. Y. Liu, JW. Bi and ZP. Fan, “A method for multi-class sentiment classification based on an improved one-vs-one (OVO) strategy and the support vector machine (SVM) algorithm,” Information Sciences, vol. 394, pp. 38–52, July, 2017, doi: https://doi.org/10.1016/j.ins.2017.02.016
8. FH. Khan, U. Qamar and S. Bashir, “eSAP: a decision support framework for enhanced sentiment analysis and polarity classification,” Information Sciences, vol. 367, pp, 862–873, November 2016, doi: https://doi.org/10.1016/j.ins.2016.07.028
9. Y. Kim, “Convolutional Neural Networks for Sentence Classification,” presented at EMNLP 2014 Annual Meeting [online].
10. ] G. Rao, W. Huang, Z. Feng, and Q. Cong, “LSTM with sentence representations for document-level sentiment classication”, Neurocomputing, vol. 308, pp. 49-57, Sep. 2018, doi: 10.1016/ j.neucom.2018.04.045
11. M. Ling, Q. Chen, Q. Sun and Y. Jia, "Hybrid Neural Network for Sina Weibo Sentiment Analysis," IEEE Transactions on Computational Social Systems, vol. 7, no. 4, pp. 983-990, Aug. 2020, doi: 10.1109/TCSS.2020.2998092.
12. Z. Yang, D. Yang, C. Dyer, X. He and E. Hovy, “Hierarchical Attention Networks for Document Classification,” p resent ed at NAACL 2016 Annual Meeting [online]. Available: https://www.aclweb.org/anthology/ volumes/ N16-3/.
13. L. G. Singh, A. Anil and S. R. Singh, "SHE: Sentiment Hashtag Embedding Through Multitask Learning," IEEE Transactions on Computational Social Systems, vol. 7, no. 2, pp. 417-424, April 2020, doi: 10.1109/TCSS.2019.2962718.
14. L. Wang, J. Niu and S. Yu, "SentiDiff: Combining Textual Information and Sentiment Diffusion Patterns for Twitter Sentiment Analysis," IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 10, pp. 2026-2039, 1 Oct. 2020, doi: 10.1109/TKDE.2019.2913641.
15. S. Akhtar, D. Ghosal, A. Ekbal, P. Bhattacharyya and S. Kurohashi, "All-in-One: Emotion, Sentiment and Intensity Prediction using a Multi-task Ensemble Framework," IEEE Transactions on Affective Computing, doi: 10.1109/TAFFC.2019.2926724.