Recognizing Sentimental Emotions in Text by Using Machine Learning

by

Tabassum Khan Bushra
18101163
Kallol Saha
18101461
Ammin Hossain Mulki
18101468
Sanjana Sabah Khan
18101502
Afrin Binta Amzad
19301267

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2022

Declaration

It is hereby declared that

1. The paper we submitted is the result of our own unique research, which we conducted while studying at Brac University.
2. The study does not incorporate anything previously published or created by a third party unless it is properly referenced by complete and correct referencing.
3. This paper does not contain any materials that have been accepted or applied for a degree or certificate from any academic or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Bushra

Tabassum Khan Bushra
18101163

Kallol

Kallol Saha
18101461

Ammin

Ammin Hossain Mulki
18101468

sanjana

Sanjana Sabah Khan
18101502

Afrin

Afrin Binta Amzad
19301267

Approval

The thesis titled "Recognizing Sentimental Emotions in Text by Using Machine Learning" submitted by

1. Tabassum Khan Bushra (18101163)
2. Kallol Saha (18101461)
3. Ammin Hossain Mulki (18101468)
4. Sanjana Sabah Khan (18101502)
5. Afrin Binta Amzad (19301267)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 11, 2022.

Examining Committee:

Supervisor:
(Member)

Moin Mostakim

Moin Mostakim
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The thesis paper we are composing is to be of the highest quality. With the level of integrity we are maintaining, our paper promises to be very reliable. The unbiased approach we took for conducting our research ensures the fairness of our analysis. We are hopeful that the findings, as well as the research itself and its methodology, will in one way or another contribute to humanity's good, and future advancement.

Abstract

As one of the fastest and most prominent deep learning technologies being fiddled with today, sentiment analysis is capable of revealing an individual's true emotions by analyzing their facial speech, text, facial expressions, gestures, and so on. The technology is being constantly used to understand how different individuals feel or react when they are put under certain circumstances or situations. The information obtained from such analyses is then processed to unravel the subject's sentimental reactions to said circumstances and situations which can further be utilized in a magnitude of ways. While the technology itself is constantly being improved upon, opportunities still exist to make it more efficient. This research aims to use a variety of machine learning algorithms and language models for sentiment detection in textual data, and understand how each of these algorithms and models approach the problems presented to them through the textual data. This is to be achieved utilizing five models that fall under three pairs namely primitive or simple models featuring TF-IDF and Bag of Words; mid complexity models featuring Naive Bayes; and advanced context-identifying state-of-the-art models namely LSTM and BERT. The datasets for this research include the Spotify App Reviews Dataset and 100K Coursera's Course Reviews Dataset. We used 10000 samples from these datasets for our research. After running the suggested models, the research aims to discover which of them works best and on which datasets, whether or not there are any similarity patterns between them, and whether or not any of the suggested models provide poor or disappointing results, all of which are provided in descriptive and quantified forms, as well as through graphical representation. For 5 label sentiment classification, Multinomial Naive Bayes gave the highest accuracy score for both the Coursera's Course Review and LSTM scored highest for Spotify App Review dataset which are 74.81% and 62.7%. For 3 label classification, pretrained BERT gave the highest accuracy score for the Coursera dataset and LSTM gave the highest score for Spotify dataset which are 91.2% and 78.3% respectively. However since our datasets very highly imbalanced, the accuracy score is a poor metric for performance evaluation of the algorithms so we looked at the f1 scores instead. We have also addressed the imbalance in out datasets by using different bias handling techniques, such as random oversampling of the minority classes. We finally reached the conclusion that both LSTM and BERT performed the best for both datasets after carefully observing the f1 scores for all the class predictions for our algorithms in both cases of sentiment label categorization.

Keywords: BERT; Bag of Words; TF-IDF; Naive Bayes; LSTM

Acknowledgement

Firstly, we would want to thank the Almighty for keeping us safe physically and mentally during the time of the COVID-19 pandemic, allowing us to continue our studies and research work despite the hardships. Additionally, we would like to convey our heartfelt appreciation to Mr. Moin Mostakim, our advisor, whose genuineness and encouragement we will always remember. We could not finish our research paper without his guidance. Furthermore, we would also like to thank our parents for their unconditional love and support, which helped us to fulfill our dreams. Last but not the least, we express our gratefulness to Brac University for giving us the opportunity to conduct our research, and providing us with the resources necessary for it.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AI	Artificial intelligence
$BERT$	Bidirectional Encoder Representations from Transformer
$LSTM$	Long Short-Term Memory
ML	Machine Learning
NLP	Natural Language Processing

Chapter 1

Introduction

Machines that have been programmed to think and behave like people use artificial intelligence (AI) to mimic human intelligence. The phrase might also be used to describe any computer that has human-like abilities like learning and problem-solving. AI enables software to automatically learn from patterns or characteristics in the data by combining massive volumes of data with quick, repeated processing and sophisticated algorithms. The development of systems that [20] include text-based emotional feeling has become increasingly the subject of research papers in recent years that are motivated by artificial intelligence (AI). A number of natural processes that AI systems represent depend on emotions. Perception, logic, education, and language processing are some of these. There are primarily 5 significant divisions of AI: 1.Machine Learning 2.Neutral Network 3.Deep Learning 4.Robotics 5.Artificial vision.

A crucial area of artificial intelligence (AI), machine learning has essentially taken the place of AI in many applications. One of the key components of our study article is this. Machine learning algorithms are used to identify the most relevant search results, detect objects in photographs, translate speech to text, match news items, communications, or products with users' interests, and more. Conventional machine learning algorithms are limited in their ability to process natural data in its raw form. In general, there are three types of machine learning algorithms: 1. Supervised Learning 3. Reinforcement learning and 2. Unsupervised learning. We may categorize data using machine language linked to supervised learning, one of such approaches, and understand it. A learning algorithm is inferred using labeled data or a data set that has been categorised in the supervised learning approach.

The major focus of our study is on automatically identifying emotions in text. Human emotions are affective states linked to physiological reactions. The [9] sentiment identification is being used in a variety of real-world scenarios where a person's emotional state acts as a clue to the machine learning system's efficiency. It may seem challenging to infer a person's emotional state from an analysis of a text document they have written, but this is frequently necessary because textual expressions are frequently the result of the interpretation of the meaning of concepts and the interaction of concepts stated in the text document. In the human-computer connection, understanding the text's mood is crucial. While text-based emotion identification systems still require development, speech recognition has seen a great deal of progress. From an [4] application standpoint, where emotion is conveyed as joy, sadness, rage, surprise, hatred, fear, and other things, the ability to recognize human emotions in text is becoming more and more crucial. The emphasis is on the sentiment-related studies in the field of cognitive psychology because there isn't a standard sentiment word hierarchy. To identify our text-based emotional sentiments, we are going to use Bag-of-Words, TF-IDF, LSTM, Naive Bayes, BERT, and more. Firstly, the bag-of-words model is an information retrieval and natural language processing structure for summarizing. This paradigm maintains multiplicity while ignoring syntax and even word order, and it portrays a text—such as a sentence or document—as the bag (individual designs) of its words. However, Bag of Words just provides a collection of vectors with the number of index terms in the document, but the TF-IDF model provides data on both the more and less significant words. Second, the metric known as TF-IDF, which stands for Term Frequency-Inverse Text Frequency, is a way to measure the importance or relevance of string representations in a document, such as words, phrases, or lemmas. It is referred to as a corpus and is employed in the domains of information retrieval (IR) and machine learning. We can quantify the importance of each word in a document by assigning it a numerical value with the aid of TF-IDF. We apply LSTM to circumvent this issue. This problem is constantly monitored by the Long Short Term Memory (LSTM) through a technique known as cell state. When the input or length increases, LSTMs find it difficult to keep the valuable information. Once more using the Bayes Theorem as its foundation, the Naive Bayes sorting algorithm presumes independence between predictors. Simply expressed, a Naive Bayes classifier thinks that the existence of one feature in a class has no bearing on the presence of any more features. In general, the Multinomial Naive Bayes classification method is a solid place to start when performing sentiment analysis. The Naive Bayes technique's basic tenet is that the probability of labels applied to texts are calculated using the cumulative values for words and classes. Finally, the BERT is used to resolve the issue. BERT's basic tenet is paying attention and recognizing how words interact with one another in context. Since BERT just encrypts data and creates a language model—not decoding it—an encoder is adequate. Unlike directional model such as LSTM, which conceptualize each input sequentially (left to right or right to left). Transformer and BERT are non-directional models since they read the complete phrase rather than the words in sequence as the input. This characteristic helps models to comprehend how a word fits into the overall context of the phrase. While Emotion Analysis seeks to identify certain sorts of sentiments expressed [7] in texts, such as anger, disgust, fear, happiness, sorrow, and surprise, Sentiment Analysis seeks to identify positive, neutral, or negative feelings from text.

1.1 Research Problems

Every year, the number of individuals throughout the world who have access to the internet grows. According to [19], DataReportal by April 2021, a new article named "Digital Around the World" clearly stated that 4.72 billion people around the world use the internet which accounts for more than 60% of the global population. They also noted that internet users are now rising at a 7.6% annual pace, equal to an average of over 900,000 new users every day. Social media and shopping platforms are mainly where people invest most of their time while using the internet. People are communicating with each other through texts, voice speeches or facial expressions on the internet and with each of them, they express their emotions. Keeping this in mind, the idea of a system by which a machine can predict and classify the sentiment of the users are more and more relevant than ever. Machine learning is a great way to let the machine learn about the user's sentiments so that it can suggest similar shopping options, show movies or posts or songs with the same theme, or suggest emoji or GIFs that express the same emotions as the user's text unlike before when machines could not recognize human emotions. That's how the machine learning approach can make the overall experience of users more delightful and effortless while using the internet.

Moreover, text is the basis for a machine to know what sentiment state the user is in initially and by using and storing the user's data, the machine will gradually self-taught itself on identifying the specific emotions. However, along with some advantages, there are some drawbacks in this system. According to a renowned article [22] by Apriorit, in the machine learning approach, we need to gather a huge amount of training data. It is quite a hassle to find a trusted corporation where we can receive and would be allowed to use their data for our experiment. Unfortunately, most of the public datasets are not sufficient. In those cases, we can create our own dataset or combine several datasets and keep modifying the data as the experiment progresses to solve this issue. Then, according to this research [14], inadequate or incomplete words are another challenge to solve for the model development part. Model development has two parts and they are - Data pre-processing and Feature extracting process. Data pre-processing turns the word into its original form and removes the unwanted words which do not represent any emotion. It also corrects any spelling mistake. Then, in the Feature extracting process, it turns the data into features which are capable of being used for machine learning models. Gradually, the method has to come across the classification of algorithms and measure the performances, they are solved by different equations and measurements.

Finally, In natural language processing sentiment analysis is an active research field. As in blogs, social networks or product reviews it performs organizing and extracting sentiments from user generated text. In this research paper we are going to explore sentiment analysis challenges which are the most complex in natural language processing. However, many organizations face the challenges of sentiment analysis but these are not difficult to overcome with the right solutions. In this guide, we have faced some research problems related to this field in our research work. First of all, this paper tackles a fundamental problem which is word score polarity categorization. For instance, words “Happy” and “Sad” are high on positive (+) and negative (-) polarity scores but in between there is mid polarity [10]. Secondly, in our research paper these are somewhat positive and somewhat negative; these words sometimes get left out and dilute the sentiment score. Another problem we have faced is that sometimes people use memes and sarcasm in social media which make it difficult for sentiments tools to detect the actual context. Because of this reason the result sometimes shows the negative value which is basically positive. In our research paper we have used two datasets- 1. Coursera course review dataset and 2. Spotify App review dataset. Among these datasets, Coursera Course dataset struggle to parse information because of biases. Another example we have faced, for instance ”Buy

Used Cars" and "Purchase Old Automobiles" are represented differently [10] in the Bag-of-Words model when the code is run through the models as text classification problems, for instance. Another illustration is that BERT performs slowly to train since it is a large and pre-trained model because there are many weights to update. Gradually, the method has to come across the classification of algorithms and measure the performances, they are solved by different equations and measurements. Finally, regardless of how this approach of machine learning solves the challenge of emotion recognition, we live in a period when additional issues may arise. As a result, according to this research paper [11], there will always be space for improvement in areas like language representation and categorization. The extraction of contextual information is critical during language representation because it serves as the foundation for enhancing categorization accuracy. Our goal in this paper is to train and test the selected models on datasets and find out which method is more appropriate and close to understanding a text just like a human.

1.2 Research Objectives

This research aims to figure out a suitable and effective method to predict and classify sentiments from text by using machine learning. Usually several tasks are divided so that altogether they can perform the human emotion reorganization process. The main objectives are -

1. Increasing the performance of the datasets mid polarities will be our primary intent.
2. We aim to comprehend sentiment analysis.
3. We like to understand how the algorithms function while dealing with various categorical matrices.
4. To identify and address any bias issues in our datasets.

Chapter 2

Related Works

This particular section indicates the importance of other research publications and how the works have impacted our process of understanding. After reading the research publications we have been critically analyzed that sentiments play promising roles in the existence or the complete make-up of individuals. According to Robert Plutchik's wheel model there are eight basic emotions such as joy, sadness, anger, fear, expectation, surprise, acceptance and disgust which are controlled by text from the paper works by other researchers. In this paper we try to show our task of sentiment analysis on text data and exchange of emotion through text message.

Without any interruption computers are unable to understand human emotion. A recent research work [21] by the research institute of Communication and Computer Systems (ICCS), Greece has proposed a method called NLP (Natural Language Processing) to understand human emotion by using text. It helps a computer to be capable of understanding the contents of speech which includes the contextual nuances of the languages. Neural NLP, one of the three types of NLP (Symbolic NLP, Statistical NLP, and Neural NLP), assists in determining the meaning of statements that are difficult for computers to decipher. The use of LSTM (Long Short-Term Memory Networks) in text classification [16] truly solves the problem of long-term learning dependencies. The accuracy level of the LSTM network has achieved 91.9% which helps to accurately estimate the expressed emotion in text. Therefore, We have converted classical text data into numerical data by using the method of TF-IDF as artificial intelligence works with numerical data. Despite its strength we have noticed when we find information from large document collections, it is not able to equate a word with its plural words [1]. We also contend that the Bag-of-Words model is the text categorization representation that manifests itself the most. Similar to the TF-IDF approach, Bag-of-Terms uses a sparse matrix to describe a document to show how frequent words are graded and given less weight than unusual words. Thereby sparse matrix representation shows the result as the vector contains many 0's which we like to avoid [3]. Another recent research work [12] indicates that a relationship between word and sentiment does not exist without a reductionist approach. However, it is also observed that a single word "happy" refers to the different emotional status such as "Don't be happy", "Look so happy" which is wildly different. [12] A machine learning system may be trained to identify the connections between words and tags or labels in texts.

The study [2] ”Recognizing emotions in text using ensemble of classifiers” depicts how people’s text messages and online content reflect their emotional states in the context of microblogging. Additionally, emotion differs from person to person and offers valuable information about the sentiments, according to a text-center approach. The second technique is commonly used for the automated identification of emotions in text and it is one of the two types of approaches that have been proposed: emotion and sentiment analysis method models. [2] The Naive Bayes hypothesis, which holds that words are mutually independent and may perform well in text classification, is briefly described in the article.

Additionally, As for sentiment analysis in many languages, the BERT model has become a widely used approach. However, we notice the BERT performs well in the evaluation of out of domain as the large information or most used dataset train on Google’s BooksCorpus and Wikipedia runs over the 95% best performance with BERT implementation. It is significantly useful that the machine predicts and learns while training the data in a large dataset, but as there are lots of weights to make updates it becomes slow to train [17]. To embed the semantically meaningful sentence we use modification of the BERT model which is Sentence-BERT. Computing with Cosine similarity and to train a siamese network with max-pooling, Sentence-BERT performs well rather than the BERT algorithm [5]. In addition, the International Survey on Emotion Antecedents and Reaction (ISEAR) helps to train up and affect text datasets.

From the discussion mentioned above, the impact of previous works reflected in our research is clearly noticeable. Our research paper has described the elaborative concept of NLP, LSTM, BERT which are effective in text datasets. Furthermore, it is also observed why the Natural Language Processing (NLP) model is considered to be the best suitable method for recognizing emotion in a text dataset. Thus the approach to consider that method NLP requires the best way to recognize sentiments in text, instead, complications of other methods also mentioned briefly in our research.

Chapter 3

Background Studies

3.1 Natural Language Processing (NLP)

As it is the information era, the volume and worth of information available now are greater than before. Because of the language barrier, individuals from all over the world could not access or comprehend knowledge that was spoken or written by humans in many languages in various ways and saved in books, CDs, and recorders. As we enter the information era, we can not only access the information but also understand it with the help of a technology named NLP. NLP helps a machine to analyze different languages and understand the meaning with context which was only possible for human beings up until now.

Natural Language Processing is the study of how a machine may learn to comprehend and interpret human language. It intersects a bunch of areas of technology and several other fields such as artificial intelligence (AI), computer science, data science, machine learning, linguistics, etc. NLP lets computers understand the text and the spoken words in the same way a human being would understand it.

We broke NLP into a few components to help a computer or machine comprehend how it works. When more time and data are given to the process, NLP performs better. Sentence segmentation is the first step in learning how NLP works. Sentences would be split from a paragraph in sentence segmentation since it would be much easier for a machine to grasp a sentence than the entire paragraph. If just the paragraph is more or less organized, an intelligent algorithm can separate sentences by acquiring the punctuation marks. If the paragraph is not organized, a complex algorithm must be applied.

The next task would be word tokenization. We would consider the words as tokens and figure out what parts of speech they belong to now that we've divided the sentences. Punctuation marks, on the other hand, all had their significance and must be regarded as tokens. A clever algorithm identifies and categorizes each word's parts of speech, allowing the computer to comprehend its functions. To do this, the computer must be pre-trained using the parts of speech classifier model. Then, text lemmatization comes. In a paragraph, most of the words remain at their root form. However, in many cases, words need to go through different grammatical forms. For instance, the words “Artist” and “Artistry” have been used in a paragraph. Despite having separate parts of speech, they both originated from the same root. This stage aids NLP in comprehending the context of any given text.

Prior to interpreting the data’s core meaning, NLP’s next goal will be to not only identify stop words but also filter them out. Every language includes unnecessary words that add nothing to the text, thus the system must eliminate them. These tokens are also identified as stop words and NLP usually skip them because they can be a cause why the NLP system can not get the insights from the data. After identifying the stop words, the following step would be figuring out the relation among all the words of a sentence. This process is called dependency parsing. Through dependency parsing, NLP algorithms construct a parse tree that identifies the sentence’s root word and connects the tokens. They may also build a parent term for each token in order to have a deeper comprehension and, as a consequence, to fully comprehend the core idea.

At present, we have a parse tree so after that, the Named Entity Recognition (NER) needs to be done. Data scientists begin extracting concepts from the text at this step of natural language processing by connecting tokens to actual things. For instance, “Sylhet is one of Bangladesh’s most great cities.” is a sentence where “Sylhet” and “Bangladesh” represent existing places and that can be detected by using NLP. The purpose of NER is to identify these nouns and associate them with the real-world ideas they represent. By removing this data from the text, an NLP model creates additional meaning that can be used to conduct a thorough study.

Now that NLP can understand the English language well, we still have an obstacle to overcome and that is in the English language, we use many pronouns such as he, she, it, they, etc. They express the same meaning as previously used nouns in a text. Coreference resolution is used to group all references to real-world concepts or objects in text. Thus, an NLP model will comprehend the meaning of words like ”he,” ”its,” and ”thus.”

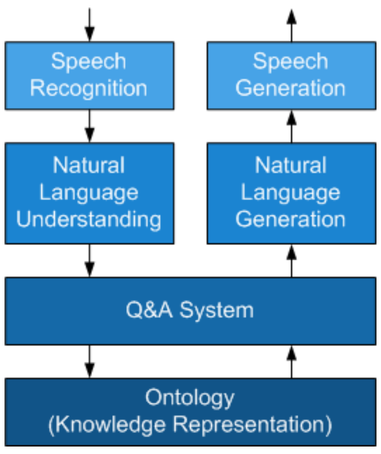

Speech Recognition
Speech Generation
Natural Language Understanding
Natural Language Generation
Q&A System
Ontology (Knowledge Representation)

Figure 3.1: Natural Language Processing

Without anyone recognizing it, NLP is being used everywhere. For instance, marketing is more complex and difficult than ever as the number of competitors is increasing along with the expectation of the customers. As businesses are going to use more online platforms, it creates a demand for smooth, personalized, and engaging experiences that will turn customers into loyal advocates. Here comes NLP to solve all these issues and helps the writer to overcome any writer's block and create the opportunity to unleash their creativity. Another application of NLP can be noticed in chatbots. Without NLP, chatbots cannot reply to users. Moreover, sentiment analysis softwares use NLP. Spam detection, fake news detection, subtitle generating, speech recognition, spell and grammar check, online search engine, email filters are also a few sectors to apply NLP. It is used by virtual assistants like Siri, Google Assistant, and Alexa. These days, NLP is extensively utilized in the finance and healthcare industries.

3.2 Sentiment Analysis

Sentiment analysis, sometimes referred to as opinion mining, uses natural language processing to determine the emotional tone of a body of text (NLP). Nowadays, businesses decide how consumers feel about any service, item, or idea. Today, social media is used by a sizeable section of the population from across the globe, allowing people to publicly express themselves. Sentiment analysis comes into play in this situation. Data mining, machine learning (ML), and artificial intelligence are used to mine text for sentiment and subjective information (AI).

A text's expression of positive, neutral, or negative sentiment is identified by sentiment analysis. The artificially intelligent bots are programmed to identify whether a message is positive, negative, or neutral based on millions of chunks of text. Sentiment analysis divides communication into topical pieces, giving each topic a sentiment score that is preset. To be able to analyze sentiments properly, the sentiment analysis system has to rely on an advanced sentiment library to detect the correct sentiment and score the words or phrases. Sentiment libraries are made up of a collection of dictionaries that have been carefully graded and include adjectives and phrases. The next step is POS-tagging. To effectively evaluate a phrase for sentiment, it must be broken down into bits using numerous sub-processes, including POS-tagging, as briefly seen above. The identification of the core constituents of a text, such as verbs, nouns, adjectives, and adverbs, is known as Part of Speech tagging. To acquire exact POS-tagging findings, which are critical for recognizing different phrase combinations, a reliable sentiment analysis system must be based on accurate natural language understanding software. However, there is a drawback and that is following strict rules cannot be always proven correct in terms of identifying sentiments. For example, "I am so glad that my book was lost." here sentiment analysis system cannot analyze the sentiment of this sentence like a human. The system will not understand the sarcasm of the sentence. That is the reason why sentiment analysis has to continuously improve all the time.

Different kinds of sentiment analysis exist. The most well-liked ones focus on emotion recognition in fine-grained sentiment analysis. The polarity category in fine-grained sentiment analysis contains extremely positive, positive, neutral, negative, and very negative. If the statement conveys a very favorable impression, it will receive a score of 5, and if it conveys a very negative impression, it will receive a score of 1. Aspect-based sentiment analysis is one more. It is mostly used to ascertain textual feelings. Whether a sentence expresses a favorable, neutral, or negative attitude can be determined via aspect-based classifiers. Next, due to the high data and resource requirements, multilingual sentiment analysis is more challenging than other types.

As the popularity of sentiment analysis increases now we can see its application of it in almost every sector. Sentiment analysis can be used to identify a brand or organization's awareness, reputation and popularity over time, track the consumer's opinion on the service or product, evaluate the success of any marketing campaign, social media, collect customers feedback on the services from online platforms and many more. The benefits of sentiment analysis are real-time analysis, sorting data at scale, consistent criteria etc.

3.3 Machine Learning (ML)

Machine learning refers to an algorithm that focuses observational data and can train predictions based on it. The machine concentrates on using data and algorithms, much like how people can learn, to increase accuracy. However, Random forest, Linear regression, Support vector machine, Logistic regression, Gradient descent and so on algorithms are used to make classifications and predictions. As the growing field of data science, machine learning is an important component that is actually a branch of artificial intelligence. There are three main parts to learn how the system of machine learning works [13].

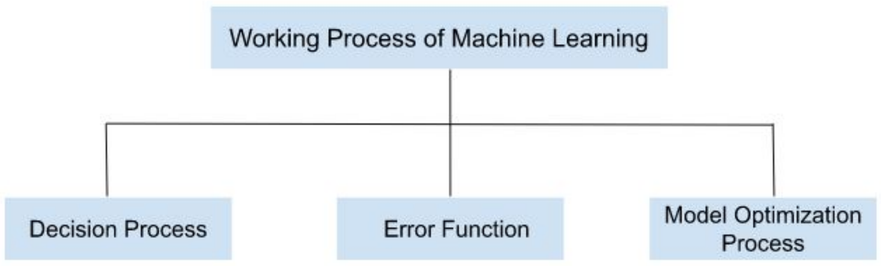

Working Process of Machine Learning
Decision Process
Error Function
Model Optimization Process

Figure 3.2: Working Process of Machine Learning

3.3.1 Decision Process

The decision process of a machine learning algorithm produces an estimate of the pattern in the data based on some input data and will basically predict the type of data by which it collects from input data.

3.3.2 Error Function

To assess the accuracy of the model error function can make a comparison between the known examples and model estimate data.

3.3.3 Model Optimization Process

Until to reach the threshold accuracy model optimization process will repeat the evaluation process and update weights autonomously.

Therefore, according to the type of signal or feedback machine learning models can be classified into three primary categories.

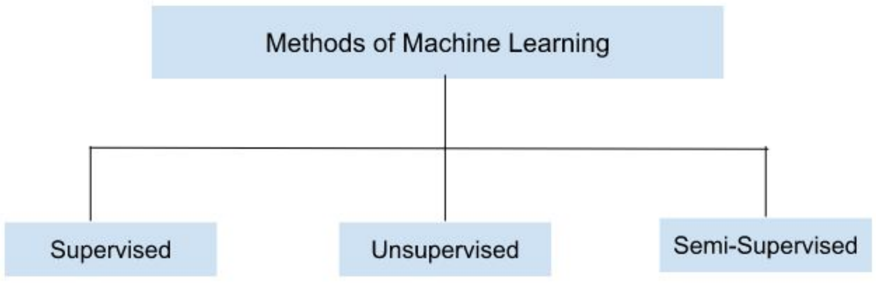


Figure 3.3: Machine Learning

3.3.4 Supervised Machine Learning

To classify the data or predict outcomes accurately supervised machine learning is defined by its training or labeled datasets which are represented using matrices. Moreover, until getting the desired outputs the model examines the result and adjusts the parameters. The main drawback of supervised learning is that classifying big data can be a real challenge but the accuracy of the result is a trustworthy method and highly accurate. Linear regression, logistics regression, neural networks are some of the methods used in supervised learning.

3.3.5 Unsupervised Machine Learning

In this unsupervised machine learning technique there is no need to supervise the model. As it deals with the unlabeled data it is necessary to allow the model to work on its own information.However, unsupervised learning has a dataset which only has inputs and is computationally complex. The main drawback of the unsupervised machine learning model, as the used data is not known it is not possible to get precise information. Furthermore, accuracy of the result is a trustworthy method but less accurate. Unsupervised learning uses some other algorithms which can be divided into different categories and they are Hierarchical clustering, Cluster algorithms, K-means and so on.

3.3.6 Semi-Supervised Learning

Semi-supervised learning refers to the intermediate ground between the learning algorithm of supervised and unsupervised machine learning. In this algorithm the training data consists of both labeled and unlabeled data and the results may not be accurate. Semi-supervised learning can be the solution of the problem when there is not enough labeled data to train the supervised learning algorithm. In the end, there are few assumptions followed by semi-supervised learning which are continuity assumption, cluster assumption, manifold assumption and so on.

However, apart from the above methods of machine learning algorithm, there is another type of learning which is called Reinforcement learning [18]. To interact with environment reinforcement learning, it enables a machine from its experience. In reinforcement, the output depends on the current input whereas the next input depends on the previous output. Reinforcement learning algorithms can be classified into two basic categories and they are positive and negative reinforcement learning algorithms.

3.4 Bag of Words

The random nature of text makes it difficult to model, and methods like machine learning algorithms favor inputs and outputs with clearly specified set lengths. Basic text cannot be used directly by machine learning algorithms; instead, the text must be transformed into numbers. more specifically, numerical vectors [8]. This is referred to in both feature extraction and feature encoding. The bag-of-words model of text is a well-known and simple method for extracting features from textual data. The technique is quite flexible and easy to use, and there are many different methods to use it to extract characteristics from texts. A textual representation of where words appear in a manuscript is called a "bag of words." There are two components:

1. A collection of well-known words.
2. A metric for the amount of well-known words.

Because any information pertaining to the organization or structure of the words inside the text is disregarded, it is referred to as a "bag" of words. The model is just concerned with whether recognized terms exist; it is not concerned with where in the article they appear. The bag-of-words technique is a widely popular feature extraction method for sentences and texts (BOW). In this method, each word count is taken into account as a feature when examining the histogram of the words in the text. If two papers include comparable information, it is considered that they are similar. The text's importance might potentially be inferred from its substance alone. The significance of the text might also be deduced from its content alone. You may make the bag-of-words as basic or as complex as you like. Determining how to define the vocabulary of well-known words (or tokens) and how to determine if known terms are present are challenging decisions.

Bag-of-Words Model designed by 3 steps. Those are-

1. Collecting Data
2. Designing Vocabulary
3. Creating Document Vectors

3.4.1 Step 1: Collecting Data

Let's take some random sentences of our regular climate. For examples-
-it is the best semester of my life
-it is the worst semester of my life
-it is the last semester of my graduation
-it is the last course of my graduation
Let's consider the four lines as our whole collection of articles for the purposes of this little example and each line as a distinct "document."

3.4.2 Step 2: Designing Vocabulary

We can now create a list of every word in our theoretical language.
In this sentence, the special words are (ignoring case and punctuation):

$\cdot It$
$\cdot Is$
$\cdot The$
$\cdot Best$
$\cdot Semester$
$\cdot of$
$\cdot my$
$\cdot Life$
$\cdot Graduation$
$\cdot Course$
$\cdot Last$
$\cdot Worst$

From a corpus of 32 words, that corresponds to a vocabulary of 12 words.

3.4.3 Step 3: Creating Document Vectors

The following stage involves scoring each document's words.

Every piece of free text will be converted into a vector that can be used as either an input or an output for a machine learning model. We may use a fixed-length document representation of 10 with one spot in the vector for each word since we are aware that the vocabulary comprises 12 words. Giving each word a boolean value to indicate its existence or absence (0 for absence, 1 for presence) is the simplest way to score text. We can walk through the first document (”It was the best of times”) and turn it into a binary vector since the phrases in our lexicon may be put in any sequence.

The following is how the document would be scored:

$\cdot It = 1$
$\cdot Is = 1$
$\cdot The = 1$
$\cdot Best = 1$
$\cdot Semester = 1$
$\cdot Of = 1$
$\cdot My = 0$
$\cdot Life = 0$
$\cdot Graduation = 0$
$\cdot Course = 0$
$\cdot Last = 0$
$\cdot Worst = 0$

This would appear as a binary vector, as follows:

1	1	1	1	1	1	0	0	0	0	0	0

[1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

The remaining three documents would resemble the following:

1. “it is the worst semester of my life” = [1, 1, 1, 0,1 1, 0, 0, 1, 0, 0, 0]
2. “it is the last semester of my graduation” = [1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0]
3. “it is the last course of my graduation “= [1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0,1]

The word orders are all theoretically ignored, and we can consistently extract characteristics from every document in our corpus that is prepared for modeling.

It is nevertheless possible to encode new texts that contain terms that fall outside the known vocabulary but overlap with it, as long as just the occurrence of known words is scored and unknown words are ignored.

We can see how this may logically scale to larger documents and vocabularies.

3.4.4 Managing Vocabulary

The vector representation of documents expands together with the language. The document vector in the preceding example has a length equal to the number of recognized words. You may imagine that the length of the vector could be thousands or even millions of locations for a very large corpus, such as thousands of books. Additionally, a very small number of the vocabulary's well-known words may appear in each document. This produces a sparse vector or sparse representation, which is a vector containing a lot of zero scores. Traditional methods may find it difficult to describe sparse vectors because of the enormous number of places or dimensions. Sparse vectors also require more memory and processing power. As a result, when utilizing a bag-of-words approach, pressure is put on the vocabulary to be smaller.

As a starter, there are easy text cleaning methods that can be employed, like:

1. Ignoring the issue.
   ignoring typical terms that don't offer much information, such as "the," "of," and related stop words.
2. Correcting spelled-out words.
3. Words are reduced to their stems using stemming algorithms (for example, "play" from "playing").

Establishing a lexicon of grouped terms is a sophisticated technique. The vocabulary's scope is changed as a result, and the bag-of-words is able to glean a little bit more meaning from the text.

$\cdot It\ is$
$\cdot Is\ the$
$\cdot The\ best$
$\cdot Best\ semester$
$\cdot Semester\ of$
$\cdot Of\ my$
$\cdot My\ life$

Trigram models, which track word triplets, are used for vocabulary, while the n-gram model, which describes the overall strategy, is used when there are many words in a group. A basic bigram method is usually more effective than a 1-gram bag-of-words model for applications like document classification. A bag-of-bigrams representation is sometimes fairly challenging to overcome and is far more powerful than a bag-of-words representation.

3.4.5 Scoring Words

After choosing a vocabulary, sample documents must have their word usage evaluated. In the last example, we saw one rather simple method of scoring: a binary score of the presence or absence of words.

Basic scoring algorithms that are not complex include:

Counts: Count the instances of each word in a document. Frequencies: Determine the proportion of a document's overall words that each word occupies.

3.4.6 Word Hashing

As we would remember from computer science, a hash function is a mathematical procedure that converts data to a fixed size collection of integers.

In programming, It might be used in hash tables where names might be changed to numbers for quick lookup.We can represent known terms in our lexicon as hashes. The issue of a very wide vocabulary for a large text corpus is resolved by setting the size of the hash space, which defines the size of the vector representation of the document.

Words are deterministically hashed to the same integer index in the target hash space. It is then possible to grade the word using a binary score or count. This is referred to as "feature hashing" or the "hash trick." The difficulty is choosing a hash space that can accommodate the specified vocabulary size while reducing collision risks and achieving a balance between sparsity.

3.4.7 TF-IDF

The drawback of word frequency scoring is that frequent keywords tend to dominate the text (larger score), yet these phrases could not give the model as much "informational content" as infrequent but possibly domain-specific words. One approach is to rescale keywords' frequency based on how frequently they appear across all publications, punishing words like "the" that are regularly used throughout several texts.

Term Frequency - Inverse Document Frequency, or TF-IDF for short, is a method of scoring that considers:

1. The phrase "term frequency" is used to rate the frequency of a word inside the current document.
2. Inverse Document Frequency is a metric for gauging a term's rarity across documents.

The ratings reflect a weighted assessment because not all words are equally important or engaging. The results highlight words in a given document that stand out or provide important information.

3.5 TF-IDF

Use the TF-IDF (term frequency-inverse document frequency) algorithm to determine the importance or relevance of string representations (words, phrases, lemmas, etc.) in a document relative to a set of documents (also known as a corpus). A score is often assigned to each word to show how significant it is to the corpus and content. This approach is often used in information retrieval and text mining applications.

There are two parts to the TF-IDF. (Term Frequency) and IDF are the acronyms for (Inverse Document Frequency). First, we'll talk about TF (Term Frequency). By looking at how frequently a certain phrase is used in connection to the document, term frequency is calculated. There are many different definitions and measures for frequency. instances of the term in the text (raw count). adjusted for term frequency (raw count of occurrences divided by word count in the document) and document length. After that, it applies a logarithmic scaling on frequency (for example, log(1 + raw count)). After that, use Boolean frequency (for instance, 1 if the phrase occurs in the document and 0 otherwise). Since TF is specific to each document and word, we may write it out as follows:

$$tf(t,d) = \frac{countoftind}{numberofwordsind}$$

So now the question is, Why not just use TF to determine the relevance between documents if we have already computed the TF value and this creates a vectorized representation of the document. On this point we will talk about the IDF(Inverse Document Frequency). We require IDF since terms like "of," "as," "the," etc. commonly arise in an English corpus and need to be corrected. As a result, we may decrease the weighting of common phrases while boosting the relevance of infrequent terms by employing inverse document frequency. Inverse document frequency analyzes the frequency (or absence) of a phrase in the corpus. IDF is determined using the following formula, where N is the number of documents (d) in the corpus and t is the term (word) we're seeking to estimate the frequency of (D). Thenumber of papers that contain the phrase "t" serves as the denominator.

$$idf(t,D) = log\frac{N}{count(d \in D : t \in d)}$$

In IDFs there is a chance that a term won't be found anywhere in the corpus, which could lead to a divide-by-zero error. Consider adding 1 to the current count to solve this. the denominator becomes (1 + count). IDFs can also be derived from a background corpus, which corrects for sample bias, or from the dataset being utilized in the current experiment.

In a nutshell, the essential idea behind TF-IDF is that the importance of a phrase is inversely associated with its frequency across texts. A term's frequency in a document is revealed by TF, while its relative rarity within the collection of documents is revealed by IDF. We may calculate our final TF-IDF value by averaging these numbers.

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

Textual input must first be turned into a vector of numerical data via a process known as vectorization, which machine learning approaches typically utilize, before being used in any natural language processing (NLP) activity, a subfield of ML/AI that works with text. The TF-IDF may be used by a search engine to rank search results according to relevance; results that are more pertinent to the user will have higher TF-IDF scores. This is due to the fact that TF-IDF can advise you of the term's pertinent relevance based on a document. Because TF-IDF weights words according to relevance, one may use this approach to determine that the words with the highest relevance are the most important. Using this, you may select keywords (or even tags) for a document or more accurately sum up articles.

However, TF-IDF has a number of drawbacks: In the word-count space, where it computes document similarity directly, it may be slow for large vocabularies. It is predicated on the idea that word counts serve as independent proof of similarity. - It does not make use of word semantic similarity.

To avoid these problems, we have to find some new models which can make our work more sufficient.

3.6 Logistic Regression

A supervised learning technique is logistic regression. The result of a dependent variable can be predicted using the statistical technique known as logistic regression based on prior data. It is a typical approach for resolving binary classification issues and is a subset of regression analysis. Logistic regression works with continuous data as well as discrete data.

In the field of machine learning, logistic regression has grown to be a crucial technique. It permits machine learning algorithms to categorize incoming data based on prior data in applications. The algorithms get more accurate at predicting classes within data sets as more relevant data is included. However, it can use model coefficients to determine the significance of a characteristic and give good accuracy for many simple datasets whereas Logistic Regression is easier to implement, interpret and very efficient to train.

3.7 Long Short-Term Memory (LSTM)

To tackle the gradient vanishing issue, the LSTM model was first introduced. This model approaches a typical recurrent neural network with a hidden layer, although instead of regular nodes, each memory cell acts as a node in the hidden layer. Each memory cell has a node with a self-connected recurrent edge of fixed weight one, which enables the gradient to go over several time steps without evaporating or exploding.

Weights operate as long-term memory in simple recurrent neural networks. In order to encode generic information about the data, the weights gradually change throughout training. Additionally, they have ephemeral activations that go from one node to the next, serving as their short-term memory. For this reason, It was phrased as "long short-term memory".

To manage this information flow, the memory cells in LSTM use special multiplicative units called gates. Each memory block featured an input gate and an output gate in the original design.

The LSTM method's gates are one of its distinctive features. A gate is a sigmoidal unit that, like an input node, gets activation from the current data point x as well as the hidden layer from the previous time step (t). The input gate regulates how input activations go into the input. If its value is 0, it acts as a gate, stopping flow from the other node. If the gate's value is one, then all flow flows through.

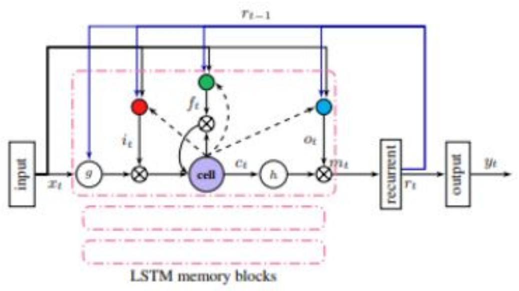


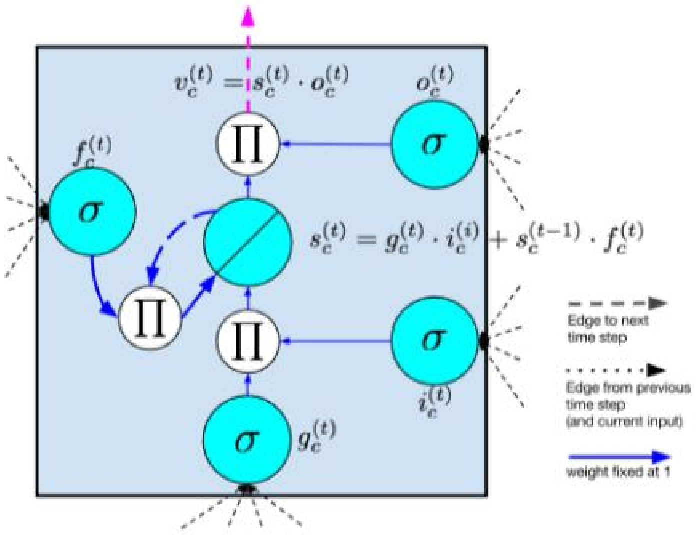
Figure 3.4: LSTM architecture

Cell activations are controlled by the output gate in terms of how they leave the cell and enter the rest of the network. It is typical to first pass the internal state through a tanh activation function in order to give the output of each cell the same dynamic range as an ordinary tanh hidden unit.

The forget gate was later added to the memory block. It was addressed that LSTM models cannot handle continuous input streams that are not broken up into subsequences. The forget gate adaptively clears or resets the cell's memory by scaling the internal state of the cell before adding it as input through the cell's self-recurrent link.

The present LSTM architecture additionally includes peephole connections from its internal cells to the gates in the same cell to enable precise timing analysis of the outputs. The equation to calculate the internal state during the forward pass while utilizing forget gates is :

$$s(t) = g(t)i(t) + f(t)s(t1)$$



$v_c^{(t)} = s_c^{(t)} \cdot o_c^{(t)}$
$o_c^{(t)}$
$f_c^{(t)}$
$s_c^{(t)} = g_c^{(t)} \cdot i_c^{(i)} + s_c^{(t-1)} \cdot f_c^{(t)}$
$i_c^{(t)}$
$g_c^{(t)}$
Edge to next time step
Edge from previous time step (and current input)
weight fixed at 1

Figure 3.5: LSTM memory cell with a forget gate

The vector h(t) reflects the value of the hidden layer of the LSTM at time t, whereas h(t1) represents the values that each memory cell in the hidden layer output at time t1. Notably, only the forget gate is included in these calculations and not peephole connections. The calculations for the simpler LSTM without forget gates are completed by setting f (t) = 1 for each t. We use the tanh function on the input node g.

Formally defined, computation in the LSTM model moves forward in accordance with the calculations that are carried out at each time step. The whole algorithm for a contemporary LSTM with forget gates is given by these equations:

$g^t = \theta(W^{gx}x^t + W^{gh}h^{t-1} + b_g)$

i$^t = \sigma(W^{ix}x^t + W^{ih}h^{t-1} + b_i)$

f$^t = \sigma(W^{fx}x^t + W^{fh}h^{t-1} + b_f)$

o$^t = \sigma(W^{ox}x^t + W^{oh}h^{t-1} + b_o)$

s$^t = g^t \cdot i^t \cdot s^{t-1} \cdot f^t$

h$^t = \theta(s^t) \cdot o^t$

3.8 BERT

BERT is one of the best natural language processing models developed in recent years.. It is a transformer-based machine learning model which is capable of a mechanism called "self-attention", making it a very popular and capable model in the field sentiment analysis and deep learning. BERT works with a layer of encoders for self-attention in which it processes data in sequence among the connected encoder layers.

3.8.1 Pre-Training

The BERT model's architecture is broken down into two phases: pre-training and finetuning. The two components of pre-training are MLM (Masked Language Modelling) and NSP (Next Sentence Prediction).

Masked Language Modelling (MLM)

In MLM, we substitute a random fraction of words with tokens (Mask). The model then uses the context derived from the remaining words to attempt to forecast the actual inputs from the hidden words. For this prediction, the encoder output must be layered with a classification layer, and the output vectors must then be multiplied by an embedding matrix in order to transform them into the vocabulary dimension. Each word in the vocabulary's probability may be calculated using Softmax. However, not all of the cloaked words are really replaced by the BERT. If we mask 15% of a text, roughly 80% of the words will be replaced by a token, 10% of the tokens will be replaced with a random token, and the remainder will be kept unaltered.

Next Sentence Prediction (NSP)

In order to understand the relationship between two phrases and to determine if the second sentence in the pair is the first sentence's concluding sentence, BERT employs NSP. Assume A and B are the two supplied sentences; 50% of the time, B will be the original second sentence that follows A, and 50% of the time, B will be a random sentence from the corpus. The goal is to get the random statement sound unrelated to the first one. As a result, a BERT model is anticipated to return 0 if the second sentence B follows the first sentence A and 1 if the first sentence A follows the second sentence B during training.

3.8.2 Fine-tuning

Fine-tuning is simple because of the transformer's self-modeling mechanism, which allows BERT to simulate a wide range of downstream jobs. We just plug in the exact inputs and outputs into BERT and fine-tune all of the settings for each activity.

3.8.3 Transformer

The BERT architecture uses transformers. The researchers show [6] that transformers use the self-attention mechanism to get a better knowledge of a language. The machine has to understand the context of each word in a text. To do that, the model must store the memory for a long period of time. It is when a self-attention mechanism is needed. “I purchased a pen yesterday,” for example. I misplaced it at school.” The machine must remember a few parts from the first phrase to recognize that “it” refers to the pen in the second sentence.

Instead of focusing just on self-attention, BERT utilizes a technique known as multi-head attention. The model may simultaneously attend to input from different representation subspaces at different places thanks to multi-head attention. By using one attention head, averaging avoids this.

There are two distinct mechanics in a transformer. The first is an encoder, which reads and processes text input, and the second is a decoder, which generates task predictions. The transformer in BERT, on the other hand, only requires encoders because this type operates in both directions (left-to-right and right-to-left) at the same time. As a consequence, the model will be able to anticipate a word’s context based on the context of all other words.

The BERT encoder has two models: BERT base and BERT big. It is a multi-layer bidirectional transformer encoder. The 12 encoders that make up the BERT base’s design each include eight layers, four multi-head self-attention layers, and four feed forward levels. For the classification, we must add a fully linked layer and a softmax layer.[15] BERT large doubled layers compared to the base model. Here, to predict the sentiments in text, we are going to use the BERT base model.

3.8.4 The input and output

Input

BERT is a model that receives inputs and creates outputs. The BERT uses certain unique tokens to generate the input. For instance, [CLS] and [SEP]. The first sentence begins with the token [CLS], and the last sentence ends with the token [SEP].

Token Embeddings: Token embeddings are numerical representations of the input sentence’s words. Larger or more complicated words are also broken down into simpler words, which are subsequently converted into tokens by BERT. For example, the word ”playing” is broken down into ”play” and ”##ing,” resulting in token embeddings. BERT does not regard complex words as new words in this case, but rather uses them in the context of the complex term.

Segment Embeddings: The purpose of segment embeddings is to aid BERT in distinguishing between two separate sentences in a single input. The embedding vec-

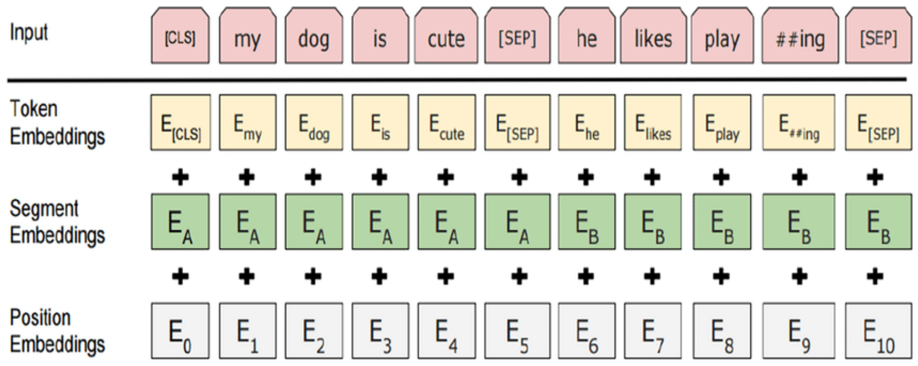

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 3.6: Embedding Layers in BERT

tor's elements are identical for words in the same phrase. So, if there are two phrases, one is "I went to the zoo." and the other is " I was alone". The tokenizer will begin by tokenizing the sentences ['[CLS]','[I]','[went]','[to]','[the]','[zoo]','[SEP]','[I]','[was]','[alone] ,'[SEP]']. The segment embeddings will then be [0,0,0,0,0,0,0,1,1,1,1]. The identical element 0 will be used in the first sentence, and element 1 will be used in the second phrase.

Mask TOKEN: The mask tokens assist BERT in determining if all of the inputs are relevant or not, as well as which ones are purely for padding. A 512-dimensional input is required for BERT. We have 12 words, for example. As a result, the final size padding will be 512-12=500. When BERT generates, it will use padding to provide a token size of 512. The index will include 1s if it is relevant to the words, and 0s if it is padding.

Position Embedding: Position Embeddings are generated internally in BERT and provide a feeling of order to the incoming data.

Output

Although the BERT base generates a 768-dimensional output, it is not required for all embeddings for classification. As a result, by default, BERT only analyzes the output corresponding to the first token [CLS] and ignores all other tokens. In a text, BERT can quite accurately anticipate spam. Token embeddings, segment embedding, and mask tokens for the input are generated and then passed to BERT. BERT will then produce a 768-dimensional output. Finally, the output is fed into a feed-forward network with the softmax activation function.

3.9 Naive Bayes

The Naive Bayes algorithm is a probabilistic classification method that can be used to classify text or other types of data. It is an easy-to-use algorithm that has been effective in many applications, including spam filtering and document categorization. Specifically, the algorithm works by treating all of the words in a piece of text as independent events and assigning a probability to each word based on how frequently it appears in text. Because it ignores the context in which words are employed, it is referred described as "naive."

Naive Bayes is a classification algorithm that uses Bayes' theorem to make predictions based on the relationships between variables.

The equation for Naive Bayes is:

$$posteriorprobability = \frac{conditionalprobability \times priorprobabilityevidence}{evidence}$$

The general notation of posterior probability can be written in the following equation form:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Here, A is the label of interest and B is an attribute being used to predict whether or not something will have label A.

The event is whether a given message contains spam or not. The algorithm uses the frequency of each word in both spam and non-spam messages as well as all of their other characteristics, such as capitalization and punctuation, to make predictions about whether a given message is spam.

The algorithm works by assigning probabilities to each word and then taking all those probabilities into account when calculating the overall probability for a given message being spam or not.

Naive Bayes is the most popular algorithm used in machine learning. It has an easy to understand concept, is simple to implement, and is effective in a wide range of applications.

The algorithm starts by looking at the words in a sentence and assigning them a probability for each possible word class. For example, when trying to classify emails as spam or not-spam, there might be classes like "sports," "travel," "business," and "personal." For each word in the sentence, it will use frequencies from training data to calculate its probability of belonging to each class. Finally, it will combine the probabilities of all the words together to produce an overall classification score.

There are many different ways of calculating these probabilities and even more ways of combining them into an overall score. However, they all come down to one thing. The model tries hard not to make mistakes by using as much information as possible.

In order to classify new text using Naive Bayes, it is first required to make an assumption about the distribution of the feature variables. Then train or fit the model using training data and evaluate it using testing data.

The training data set consists of a bunch of documents (sentences) that have already been classified into one category or another (i.e., spam or not spam). The training set includes both positive and negative examples, where a positive example contains words that are likely to appear in spam messages, while a negative example contains words that are unlikely to appear in spam messages.

The Naive Bayes classifier is pretty straightforward: it assumes that if some features are present in both the positive and negative examples, then those features must be important for classifying new instances correctly. Otherwise, they are not important.

3.10 Confusion Matrix

The classification model's accuracy in classifying instances into distinct groups is summarized in a table called the confusion matrix. The model's predicted label is on one axis of the confusion matrix, while the true label is on the other. When comparing several models, we may use the confusion matrix to assess how well each one predicted true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). We chose a model as our base model if it accurately predicted TP and TN compared to other models.

3.11 Precision

In terms of positive observations, precision is the proportion of correctly predicted observations to all expected positive observations. The classifier's capacity to refuse to classify a negative sample as positive comes intuitively. 1 is the best value, while 0 is the worst. Low false positive rates are related to high accuracy. The formula of precision is:

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

3.12 Recall

The recall is determined as the number of true positives divided by the total number of true positives and false negatives. This differs from precision, which counts the proportion of accurate positive predictions among all positive predictions given by models. The machine learning model is more adept at recognizing both positive and negative samples the higher the recall score. To gain a comprehensive perspective of the model's performance, the recall is sometimes combined with additional performance measures like precision and accuracy. Its formula is as follows:

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

3.13 F1-Score

A weighted average of recall and precision makes up the F1 score. As an alternative to accuracy measures, F1-score is a machine learning model performance statistic that equally weights Precision and Recall when assessing how accurate the model is. The harmonic mean of recall and accuracy is known as the F1-score. Another method of determining an "average" of values is the harmonic mean, which is typically seen as more suited for ratios (such as recall and accuracy) than the conventional arithmetic mean. The formula of the f1-score is:

$$F1Score = 2 \times \frac{(Recall \times Precision)}{(Recall + Precision)}$$

3.14 Accuracy

The simplest performance metric to understand is accuracy, which is just the proportion of correctly predicted observations to all observations. A model is considered to be good if its accuracy rate is high. But accuracy is only a valuable statistic when the datasets are symmetric and the values of false positives and false negatives are about equal. As a result, the performance of the model must also be evaluated using the other parameters.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

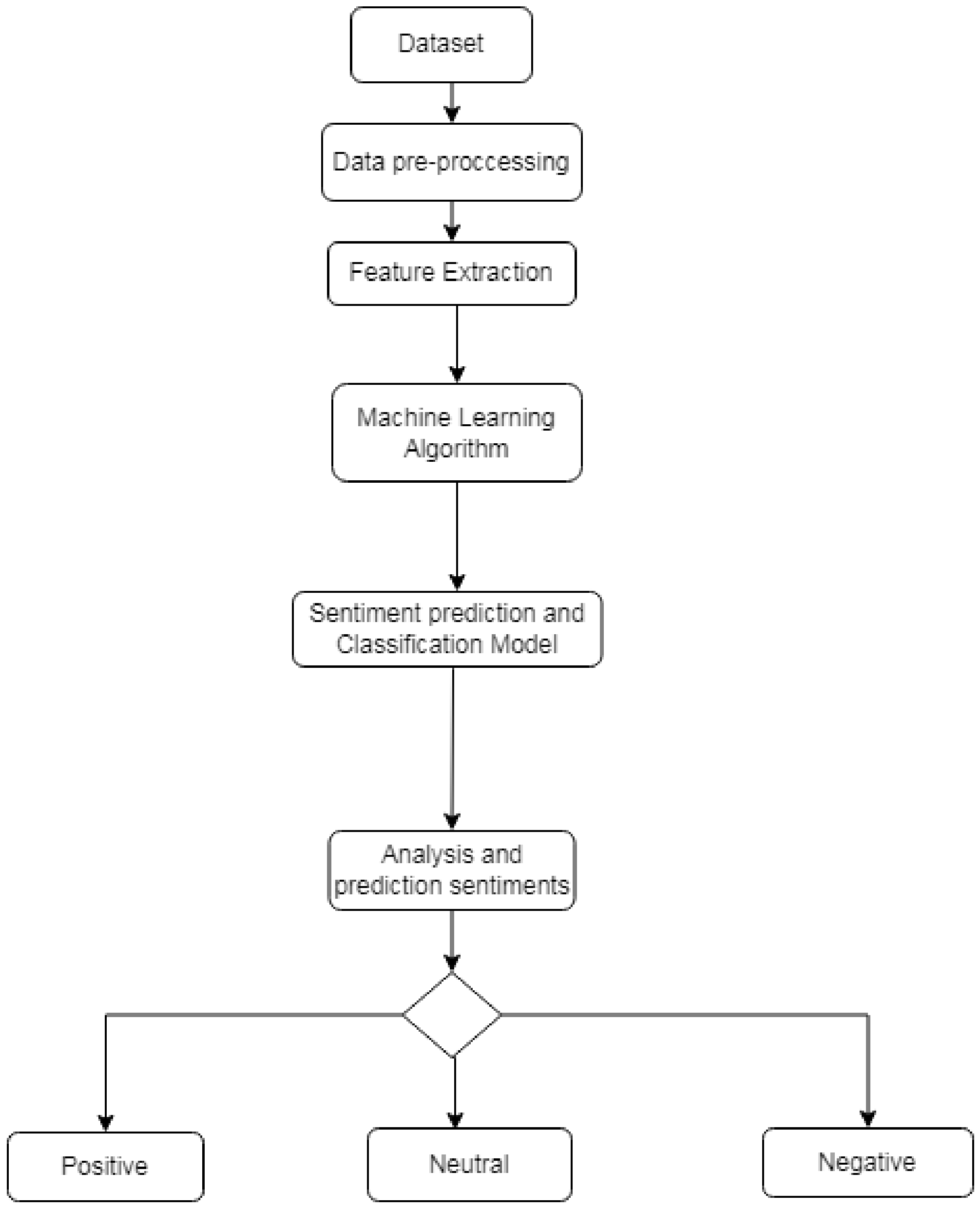
Chapter 4

Methodology

This work's major goal is to employ a review dataset suitable for sentiment analysis of various human judgment and emotion spectrums that captures the ranges between the positive and negative poles. This is done to predict and detect emotions as accurately as possible; as the human mind is complex and so is human judgment which can only be satisfied through spectrum classification rather than simply the binary classification of whether the customers/ audiences were satisfied by the products they reviewed or not. In order to do so, the system needed to design a process where the machine respectively takes the input data, processes the dataset, predicts, analyzes and classifies the data and produces the correct output not only through word vectorization and token predictions but also through more layered and structured domains of text classification and context understanding. To achieve this we took 5 models that can fall under the following categories in three pairs namely Primitive/ simple models namely, TF-IDF and Bag of Words; Mid complexity model which is Naive Bayes and advanced context-identifying model namely LSTM and BERT. Textual documents are used for classification models to train the polarity and are represented by vectors to adopt the machine learning approach. Through trials on two datasets that include Spotify App Reviews, Coursera's Course Reviews Dataset, the sentiment classification problem is analyzed. The suggested textual phenomena and the changes in language-specific expressions need little computational resources, which may have an influence on automating sentiment identification and retrieval.

4.1 Datasets

To perform our experiment, we have used two different datasets which were collected from the renowned Kaggle site. We have used two datasets containing 64k Spotify App Reviews and 100K Coursera's Course Reviews Dataset. These datasets were used in various experiments by other researchers from time to time. Aspect-based sentiment analysis in music: a case study with Spotify and Alternative Methods for Deriving Emotion Metrics in the Spotify Recommendation Algorithm, the 100K Spotify App Reviews dataset was utilized in both articles. Reviews of Coursera's offerings A Bayesian CNN-LSTM Model for Sentiment Analysis in Massive Open


Dataset
Data pre-proccessing
Feature Extraction
Machine Learning Algorithm
Sentiment prediction and Classification Model
Analysis and prediction sentiments
Positive
Neutral
Negative
Figure 4.1: Overview of Methodology

Online Courses MOOCs and an Analysis of Learners' Affective and Cognitive Traits in Context-Aware Recommender Systems (CARS) utilizing Feature Interactions and Factorization Machines (FMs) both utilized the dataset.

For Spotify App Review Dataset, Coursera Course Review Dataset, we kept 10000 reviews for each algorithm. However, as we used a pre-trained model in BERT algorithm that is why we kept 2000 reviews. We trained and tested on this size of datasets. For each of these datasets: we have chosen the columns that contains the reviews of the users and the ratings they have given to these courses/app/movie. From the coursera datasets, we have chosen the 'Review', 'Label' columns, from spotify datasets, we have chosen the 'Review' and 'Rating', and renamed each of these pairs of columns as 'review' and 'sentiments' respectively of all two datasets for the ease of coding. These datasets were selected as each of them contain labels comprising of the rating score 1 - 5 which is helpful for detailed sentiment analysis as we can derive these scores as such - **1: negative, 2: somewhat negative, 3: neutral, 4: somewhat positive and 5: positive**. Classifications of such variations allow for analysis along the spectrums of positive and negative aka the grey area between both extremes. After working with five label classification, we decided to simplify our sentiment category of five labels into three labels, namely **1: negative, 2: neutral and 3: positive**.**We did this by relabeling our initial five sentiment categories such that 5: 3, 4: 3, 3: 2, 2: 1, 1: 1 where 3, 2, 1 are categorized as positive, neutral and negative respectively**.

4.2 Data Pre-Processing

The datasets that we used in our paper were not fulfilling all the necessary requirements to run through the codes. That is why we needed to clean the datasets first by removing the special characters by importing and applying regular expression operation and also kept all the alphabets in lower case. Then pre-processing started where we used df.shape operation to know the shape of our dataset. We also collect the number of unique values of the sentiment column. To find out about the null values, we used df.isnull().sum(). Then, we needed to remove all the unnecessary rows and columns from our datasets. For doing this, we used df.drop() function. To drop an unnecessary column we had to keep the axis as 1 and for dropping the rows, we need to keep the axis as 0. By the end of all the pre-processing, we can get the datasets that we actually going to use for our codes.

4.3 TF-IDF Model Explanation

Firstly, we performed the data pre-processing steps and removed all the unnecessary special characters, rows and columns from the dataset. After doing these, we need to download the stop words after installing nltk and select the language in English as our dataset is in English. Then, we removed all the stopwords from review. Stopwords are those words which do not carry any sentimental value. For example, ‘I’, ‘You’ do not carry any sentiment in a sentence so for our models these words are unnecessary. That is why we removed them. Then we had to perform the lemmatization process. The Lemmatization process is where the code removes any parts of speech attached with words. For instance, if there is ‘songs’ word in the dataset, by applying lemmatization this word would turn into its true form which is ‘song’. After the lemmatization process, the column ‘review’ was renamed as ‘review_processed’.

Moreover, to perform train- test split, at first we had to assign two variables where x would store review_processed and y would store sentiment. Later, train_test_split() method was used to separate the data to train and test. In this particular method x, y, random_state set as 42, test_size set as 0.20, shuffle set as true and stratify set as y. The test_size = 0.20 means 20% of the reviews are going to be used for testing purpose only and the rest 80% is for training. By using the shape function again, we got to know the shape of x_train and x_test.

Furthermore, TfidfVectorizer has been imported from sklearn.feature_extraction.text and it would compute the word counts, idf and tf-idf values all at once. Then, fit the x_train and transform this into tv_train_transformed by using tvectorizer.fit_transform() function. With the help of tvectorizer.transform(), x_train was transformed into tv_test_transformed. As we know that TF-IDF can only vectorize the dataset but for obtaining the accuracy we had to perform logistic regression operation. In logistic operation, we needed to have these parameters and they are penalty as l2, max_iter as 1000, C as 1 and random_state as 5. Then, fit this train datasets by using the fit function in the logistic regression model. By using the predict function, we predicted the transformed tv_test_transformed. Now, we can find the accuracy score with the help of y_test and y_pred. We would use them to figure out the classification_report and the accuracy for five labels.

As the datasets we used are biased, we had to perform different techniques to get a better performance such as class weight assigning and oversampling. For the class weight assignment, it had to import from sklearn library. Then, the sentiment which was in a list is now stored in unique_classes. An empty out_dict is being called and for each unique class, the classes were stored in out_dict where the applied formula was df.shape[0] / ((df.loc[df['sentiment'] == classes].shape[0]) * len(unique_classes)). Then, printed the out_dict. Again by using logistic regression where class_weight is balanced and max_iter is 500, we fit the tv_train_transformed and y_train in it. In the following line, the class_weight.compute_class_weight was being set where the parameters were class_weight is balanced, classes set in np.unique(y_train) and y was set as y_train. After that, we predicted the transformed x_test. At last, figured out the classification report. On the other hand, to increase the number of minority data points, the oversampling approach duplicates them at random. Only the value counts of y_train are now transferred in each count_class of five labels. Then, we applied the Concat function on the x_train, y_train, and also kept the axis as 1. Then the five sentiments of the training part were moved to the five different classes. Now, in the sampling process where we took the class_5 and replaced it with the classes from 1 to 4 for oversampling. Again, by using the Concat function all the oversampled classes and class_5 and axis as 0. Now, we assigned the oversampled train dataset into new X_train and Y_train accordingly. After fitting the training parts in the logistic regression model and also predicting the tv_test_transformed1. Lastly, the classification_report was used to get the performance result and now the model is predicting better than before.

When we label the sentiment into three (Positive, Neutral and Negative) instead of five, we get a better accuracy score which is 78% however, the model could not predict the neutral sentiment. For the weight class assignment, improvement was noticeable. After performing the oversampling, the accuracy was 73% and the neutral sentiment was predicted well than before.

4.4 Bag of Words Model Explanation

At the start, we performed the cleaning and data pre-processing on our dataset. After completing these steps, we deleted stopwords and select English as the language because our dataset was in English. The lemmatization method was done. The lemmatization procedure involves removing any attached parts of speech from the code.

Additionally, in order to do a train-test split, we first had to assign two variables to which x would hold review-processed and y would store sentiment. The data for training and testing had been separated using the train_test_split() method. In this specific approach, the random state is set to 42, the shuffle set to true, stratify set to y, X, Y, and test_size is also set to 0.20. The test_size value of 0.20 indicates that only 20% of the reviews will be used for testing, with the remaining 80% being used for training. We learned the shapes of the x_train and x_test by utilizing the shape function once more.

We vectorize the code for the bag of words after importing CountVectorizer from sklearn.feature_extraction.text. Then, fit the x_train and transform this into tv_train_transformed by using cvectorizer. transform(x_train). As far as we are aware, bag of words can only vectorize the dataset; however, in order to gain accuracy, logistic regression had to be used. These settings, which are penalty as l2, max_iter as 500, C as 1, and random_state as 5, are necessary for logistic operations. Then use fit function to fit this model. We forecast the converted x_test using the predict function. With the aid of y_pred and y_test, we can now determine the accuracy score. The accuracy we obtain when we utilize them to determine the classification_report.

We had to employ several strategies, such as class weight assigning and oversampling, to improve performance because the datasets we used were biased. For the class weight assignment, it had to import from sklearn library. We kept the manual_weights as 5: 5, 4: 20, 3: 30, 2: 40, 1: 20 . Again by using logistic regression where class_weight was manual_weights and max_iter was 500, we fit the cv_train_transformed and y_train in it. In the following line, we predicted the transformed x_test. At last, figured out the classification report where the improvement in performance was noticeable. On the other hand, in oversampling, only the value counts of y_train are now transferred in each count_class of five labels. Then, we applied the Concat function on the x_train, y_train, and also kept the axis as 1. Then the five sentiments of the training part were moved to the five different classes. Now, in the sampling process where we took the class_5 and replaced it with the classes from 1 to 4 for oversampling. Again, by using the Concat function all the oversampled classes and class_5 and axis as 0. Now, we assigned the oversampled train dataset into the new X_train and Y_train accordingly. After fitting the training parts in the logistic regression model and also predicting the cv_test_transformed1. Lastly, the classification_report was used to get the performance result and now the model is predicting each sentiment better than before.

When we label the sentiment into three (Positive, Neutral and Negative) instead of five, we get a better accuracy score which is 77% because of fewer layers however, the model could not predict the neutral sentiment well. For the weight class assignment, we also predict the neutral sentiment. After performing the oversampling, the accuracy was 73% and the neutral sentiment was predicted well than before.

4.5 LSTM Model Explanation

We have set num_words and a variable X to 600 and 0 correspondingly at the start of the LSTM. After that, the model has been called tokenizer() method to keep the maximum number of words from the review in the tokenizer with the help of tokenizer.fit_on_texts() function. Now that we have done with our tokenization, it would be easy to convert them into integer by using tokenizer.texts_to_sequences(). Next, we have performed the padding as we have different length for different reviews. To have them all same space, we had to apply pad_sequences(). It would make the process easy and better. We have use vocab a variable which stores a word_index as it maps words to some number.

Later, in the code, we have tried to create the LSTM model and taken embed _dim and lstm_out as 128 and 196 respectively. After that, the model has been defined as a sequential model stacking multiple layers. To add the first layer, firstly embedding is done and the length of input has been defined as X.shape[1]. Then, to make it easier to process, we used the SpatialDropout1D() function as 0.4 where it is used to drop the entire 1D feature maps instead of an individual element. After that, a dropout layer is added in the code section. A dropout layer prevents the overfitting of the model and robust the model's performance. The layer randomly drops some inputs. Furthermore, the dropout rate is specified while adding the layer. In this model the rate is 0.2 and the recurrent dropout is also 0.2. In the following in of code, Dense() is called and the mentioned parameter is activation. Here softmax function is used as the activation potential.

In addition, before training the model, the learning process must be set up. A compile procedure with three parameters is invoked to do it. The loss function is the first input. The effectiveness of the algorithm in modeling the supplied dataset is examined by this function. Categorical_crossentropy is the most suitable loss function since we utilized softmax as an activation function. An optimizer is used as the second parameter to lower the model's losses. The learning rate in this instance of the Adam optimization method is 0.0001. Metrics make up the third variable.

This model employs an accuracy metric. This measure calculates the overall forecast accuracy rate. The final step is to return the model.

Then, by using one hot encoding with the help of get_dummies for sentiment and we could call the train-test split method to split the dataset into two for training and testing the reviews. In this code, we have kept the test_size = 0.2 as we need 20% of the data to test and the rest to train and also the random state as 42. To fit the model, the batch_size kept as 128, epochs as 7 and verbose as 1. Epochs are used to run the dataset again and again to achieve higher accuracy.

For the prediction, model.predict() is being used and x_test and y_test. With argmax function from numpy the pred and axis=1are stored in y_pred_ and then took the y_test and axis=1 and stored it in y_test_. The target_name is ['1', '2', '3', '4', '5']. Then, call the classifier report and return the y_test_ and y_test_.

When we performed the class weight assignment, unique_classes was taking a list of sentiment. By taking the manual_weights= 4: 5, 3: 20, 2: 25, 1: 50, 0: 10 , we assigned the weights. Then compute all the class_weights. Now, we had to fit this in history with the epochs is 7, batch_size=batch_size, verbose is 1. Then, print the history. The predict method is called and applied argmax function which is stored in y_1 and y_test1. The classification_report and accuracy were printed. After oversampling, we did the same process, however, x was being tokenized. Y needed to do one hot encoding and that was done by using get_dummies from pandas. Again, train, test split happened and then fitted the model.

Additionally, we wanted to check if we do not divide our dataset into 5 parts but rather 3 parts such as positive, neutral and negative, then how much will change in accuracy. That's why, we label 1 and 2 = -1(Negative), 3 = 0(Neutral) and 4 and 5 = 1(Positive). This changed dataset would go through all the same processes as the previous dataset was done. Finally, the accuracy score has been increased and it is 78% now(for the Spotify reviews dataset). Then after adding class weight, and after oversampling were performed.

4.6 Naive Bayes Model Explanation

For conducting preprocessing operations like changing all words to lowercase and deleting special characters, unnecessary rows and columns. Then, CountVectorizer implies breaking down a phrase or any text into words. Textual data has to be vectorized since NLP algorithms only take numerical data and cannot interpret language.

Then, we stored the review in a variable x and the sentiment of the dataset was stored in another variable called y. Now, with the help of the train_test_split() function, we split the whole dataset into two and the parameters are such as x, y, random_state is 42, test_size is 0.20. The test_size is 20% in this code means that 20% of this dataset is for testing and the rest of the 80% reviews of the dataset is for training.

Moreover, for getting accuracy we applyed MultinomialNB() classifier. When analyzing categorical text data, one of the most well-liked supervised learning classifiers is Multinomial Naive Bayes. Using the Bayes principle, it makes an educated prediction about a text's tag, like "story." It determines the likelihood of each tag for a certain sample and produces the tag with the highest likelihood. While fit function is scaling all the training data, the x_test is transformed into vectorizer and stored in cv_test_transformed. The predict function is working with the cv_test_transformed in it and stored it in y_pred. By applying accuracy score and classification report functions, we got an accuracy and that is 62%

Complement Naive Bayes is used when there is an imbalanced dataset and when Naive Bayes can not do that. So, in this case, we used it. At first, importing ComplementNB from sklearn library and also import class_weight. Then, class_weight= 'balanced', classes=np.unique(y_train), y = y_train were the parameters of class_weight.compute_class_weight. Then, we fit the necessary parameters by weighted_model.fit. With the help of predict function, we took cv_test_transformed. With accuracy_score, we got the accuracy. After applying oversampling, the value counts of y_train are now transferred in each count_class of five labels. Then, we applied the Concat function on the x_train, y_train, and also kept the axis as 1. Then the five sentiments of the training part were moved to the five different classes. Now, in the sampling process where we took the class_5 and replaced it with the classes from 1 to 4 for oversampling. Again, by using the Concat function all the oversampled classes and class_5 and axis as 0. Now, we assigned the oversampled train dataset into the new X_train and Y_train accordingly and X_test = x_test and Y_test = y_test . After fitting the training part and then transformed in cv_train_transformed1. In the following line, the training dataset got fitted in the classifier and also predicted the cv_test_transformed1. Lastly, the accuracy was called to get the performance result and now the model is predicting each sentiment more accurately.

If we label the ratings of sentiment 1 and 2 as 1, 3 as 2 and 4 and 5 as 3, that means we now we divided our dataset into positive, neutral and negative sentiment. As we apply the MultinomialNB() classifier, it gives the accuracy of 78% for sentiment label 3. However, class weight performed well. After oversampling all three sentiments prediction performed well.

4.7 BERT Model Explanation

Firstly, we used AutoTokenizer from pre-trained bert model (nlptown/bert-base-multilingual-uncased-sentiment) and download them. Then, we load the dataset by using pandas. We also performed the cleaning and pre-processing the dataset. We use the encode_plus() method and in it we got the following parameters and their values: review, add_special_tokens = True, truncation = True, padding = "max_length", return_attention_mask = True, return_tensors='pt'.

Then, we take reviews as reviews and count as 0. Now, running a for loop where count=count+1 and drop the unnecessary columns from from dataset and also used lower case.So, for the prediction part, num=0 and in another for loop reviews are rows and it has global num in it. Again, with the help of encode_plus () method, we have rows, add_special_tokens = True, truncation = True, padding = ”max_length”, return_attention_mask = True, return_tensors=’pt’. Now num=num+1. With the appended function we predict the (int (torch.argmax (result.logits)) + 1). At last print the num. Review will be stored in sentiment_score.

We need to import accuracy_score, confusion_matrix, classification_report from the sklearn library. When we code for classification report and accuracy model it gives accuracy of 60% for Spotify dataset.

For sentiment simplification into three categories (positive, negative, neutral) from five, we get the accuracy of 77% which is significantly higher than before.

Chapter 5

Result and Analysis

Google Collab has been used to run all the algorithms. In this research, we have shown the accuracy rate of all the models where the precision, recall and f1-score are also there.

The two datasets used for analysis were imbalanced in terms of the chunks of each sentiment classes, i.e., total numbers of reviews that fall under a single sentiment score class. For Coursera's Course Review dataset, we see that out of 10000 samples the total number of samples of reviews with score 5 are 7141, score 4 are 1777, score 3 are 495, 2 are 260 and 1 are 328. From this we understand that a huge number of samples fall under sentiment class 5 followed by 4 so most reviews are positive or somewhat-positive, whereas the 3, 2, 1 sentiment which are neutral, somewhat-negative and negative are very little in sample. In the same way for Spotify App Review dataset, we find 4181 samples for 5, 1352 for 4, 1021 for 3, 920 for 2 and 2526 for 1 so we see class 5 holds the highest number of samples here as well and class 1 comes in second while sample 2 is lowest in number so this dataset also carries some imbalance for neutral and mid pole classes (4, 2) however, it is much more leveled in comparison to the one for Coursera. Coursera has too few samples for any of the algorithms to be familiarized with reviews that are not positive or along its poles. In such imbalanced circumstances observed for both datasets, the algorithms/ models used are more likely to be predominantly biased towards positive reviews than the rest; for Spotify they are able to predict the negative reviews too for it being the second highest sample but cannot perform much for the other three classes. As a result of this we get skewed accuracy, especially in terms of precision, recall and f1 score if we run the data without any sort of bias handling. We also observe the same for 3 labels, Sentiment 1: 470, Sentiment 2: 401, Sentiment 3: 7129 samples for Coursera and sentiment 1: 2777 samples, sentiment 2: 829 samples, sentiment 3: 4394 samples for Spotify dataset respectively. So, classes 1 and 2 sample size being overwhelmingly lower than class 3 for Coursera and Class 2 having lowest sample but class 1 and 3 having more balanced distribution for Spotify. As a result, this also skews our prediction rate for 3 labels before bias handling is done.

We have used a technique known as Oversampling or Up-sampling to address the imbalance that exists in our datasets. Here we increase the sample size of the other classes to the sample size of the class with the highest sample, the majority class. In both our datasets, sentiment class of positive, 5, for 5-label-categorization or 3, for 3-label-categorization, contains the highest number of samples of all other classes. Therefore, the rest of the classes are each resampled to the sample size of class positive, 5 or 3 depending on the number of label category, and then the algorithms are trained with the resampled data where each class contain equal number of samples now. Such gives an equal ground to all the sentiment target classes in our data for better and more precise evaluation from the implemented models.

Additionally, there is the natural phenomena of most ML algorithms being able to predict binary data of positive and negative better than neutral and mid pole ones as these are tricky to predict for being abstract in nature. This is why we have first run our models with imbalanced data and then then applied biased handling techniques to our data and run them again for both 5 and 3 label categorizations. The results are compared below for both just accuracy score and then a detailed analysis of the classification report we get for each evaluation.

From the above tables we can see the accuracy scores, classification metrices of the models for two distinct division of labels: 5 label and 3 label classification before and after imbalance handling. Let us look at the scores for each of them.

For 5 label sentiment classes:

Models	Original value without any sort of imbalance addressing (5 labels)		After imbalanced dataset addressing through Oversampling/ Up-sampling (5 labels)	
	Coursera's Course Review	Spotify App Review	Coursera's Course Review	Spotify App Review
TF-IDF (Logistic Regression)	74.11%	61.25%	63.31%	52.35%
BoW (Logistic Regression)	73.36%	58.5%	65.36%	52.75%
Naive Bayes (Multinomial-NB)	74.81%	61.9%	67.52%	52.95%
LSTM	74.76%	62.7%	81.30%	61.15%
BERT (base-uncased-multilingual-sentiment)	68.24%	60.21%		

Table 5.1: Accuracy Table for Sentiment 1-5

Accuracy score analysis: For 5 labels, we can see that the accuracy before imbalanced/ bias handling is higher than the accuracy that came after balancing the data by oversampling the training samples to the size of the highest chunk amongst the classes, i.e., of class 5(positive). This happens because accuracy score is the simplest metric for scoring performance and works best incase of equal distribution of data, however, accuracy score proves to be highly unreliable when it comes to imbalanced datasets because it only considers the number of correct predictions as opposed to number of total predictions. What happens in case of imbalanced dataset such as ours is that if the models fail to or does not attempt to make predictions for minority classes and makes very accurate prediction for the majority class then accuracy will give a high score as majority of the predictions made were correct and it does not have a penalty for bias handling and ignores failure for minority classes. The f1 score metric is the one to be relied upon for imbalanced datasets as it has a weight metric that effectively calculates for both majority and minority class by considering true positives against false positives and false negatives. Despite the high accuracy scores before bias handling, the f1 scores are very low for the lower sampled class for our models and this indicates a failure in performance. For Coursera, the most imbalanced dataset, we observe higher accuracy scores than Spotify but f1 scores of 0s for Tf-idf and Naïve bayes for class 2; where Spotify managed to make some predictions, whether correct or not, under the same instances. This lowers the accuracy score for Spotify but the f1 score increases and it indicates clearly that Spotify performed better for having comparatively more data in the chunks for the classes. In case of oversampling the data for equal distribution, we see lesser accuracy score but improved precision, recall and f1 scores for minority class which indicates more attempts at predicting those classes and better performance as the f1 scores increases by some amount. Before Oversampling, state-of-the-art models LSTM and pretrained BERT (base-multilingual-uncased) performed best despite lower accuracy scores than TF-IDF, BOG and Naïve Bayes as they have higher f1 scores for neutral class (3) and mid polar class (2, 4), such is expected as these are complex context-understanding models and will perform better than simple word-vectorizing algorithms and prediction based naïve bayes despite great imbalance in the class samples. After Oversampling, LSTM still performs the best against the other three.

Models	Original value without any sort of imbalance addressing (5 labels)							After imbalanced dataset addressing through Oversampling/ Up-sampling (5 labels)					
		Coursera's Course Review			Spotify App Review			Coursera's Course Review			Spotify App Review		
TF-IDF (Logistic Regression)	Senti-ment	Precision	Recall	f1-score	Precision	Recall	f1-score	Precision	Recall	f1-score	Precision	Recall	f1-score
	1	0.43	0.09	0.15	0.56	0.82	0.67	0.35	0.45	0.39	0.60	0.52	0.56
	2	0.00	0.00	0.00	0.28	0.06	0.10	0.15	0.17	0.16	0.18	0.27	0.22
	3	0.75	0.06	0.11	0.23	0.07	0.11	0.17	0.27	0.21	0.18	0.24	0.21
	4	0.47	0.24	0.31	0.35	0.17	0.23	0.29	0.38	0.33	0.25	0.26	0.26
	5	0.77	0.97	0.86	0.72	0.89	0.79	0.86	0.75	0.80	0.83	0.74	0.78
BoW (Logistic Regression)	1	0.53	0.30	0.38	0.59	0.68	0.63	0.38	0.42	0.40	0.59	0.50	0.54
	2	0.14	0.06	0.08	0.20	0.14	0.16	0.15	0.19	0.17	0.13	0.17	0.15
	3	0.26	0.18	0.22	0.19	0.12	0.15	0.17	0.24	0.20	0.18	0.22	0.20
	4	0.41	0.23	0.30	0.30	0.18	0.23	0.31	0.35	0.33	0.28	0.28	0.28
	5	0.80	0.94	0.87	0.73	0.87	0.79	0.85	0.79	0.82	0.80	0.78	0.79
Naive Bayes (Multinomial-NB)	1	0.75	0.05	0.09	0.52	0.87	0.65	0.31	0.36	0.34	0.61	0.50	0.55
	2	0.00	0.00	0.00	0.23	0.03	0.06	0.14	0.19	0.16	0.18	0.33	0.24
	3	0.18	0.02	0.04	0.15	0.03	0.05	0.19	0.35	0.25	0.19	0.23	0.21
	4	0.42	0.29	0.34	0.46	0.26	0.33	0.37	0.41	0.39	0.29	0.34	0.32
	5	0.80	0.97	0.88	0.77	0.86	0.81	0.89	0.80	0.84	0.87	0.73	0.79
LSTM	1	0.45	0.44	0.45	0.58	0.82	0.68	0.82	0.75	0.78	0.71	0.59	0.65
	2	0.38	0.06	0.10	0.18	0.06	0.09	0.69	0.66	0.67	0.25	0.40	0.31
	3	0.27	0.20	0.23	0.30	0.25	0.27	0.50	0.64	0.56	0.27	0.36	0.31
	4	0.45	0.26	0.33	0.44	0.28	0.34	0.55	0.52	0.54	0.41	0.46	0.43
	5	0.82	0.94	0.88	0.80	0.84	0.82	0.90	0.90	0.90	0.89	0.77	0.82
BERT (base-uncased-multilingual-sentiment)	1	0.60	0.39	0.47	0.62	0.73	0.67						
	2	0.24	0.49	0.32	0.26	0.33	0.29						
	3	0.31	0.42	0.36	0.26	0.30	0.28						
	4	0.33	0.47	0.39	0.38	0.34	0.36						
	5	0.90	0.78	0.83	0.88	0.75	0.81						

Table 5.2: Precision, Recall and F1-Score (1-5) of Models

Classification report analysis: For 5 label classification: For Coursera course review dataset, the sentiment classes of 1-4 for precision, recall and f1-score are all low when the model is Tf-Idf (LR). For class 2, all the three metrices are 0% which means the algorithm could not predict it at all. Class 1 and 3 gives f1 score of 15 and 11% where for class 4 it is 31% as the sample for this class higher after class 5. However, for sentiment rating 5 (positive) having the highest sample, the Tf-Idf-LR algorithm has done well with the precision, recall and f1-score of 0.77, 0.97 and 0.86 which is drastically higher. After Oversampling for equalizing samples, we see some increase in f1 scores of the classes with low scores but they are still less than 40%. For Spotify dataset, it gives the highest score for class 5, 79% and class 1, 67% as the positive and negative samples were more balanced and low scores for class 2-4 with low samples. After Oversampling we see improvements in the scores for 2-4 but still below 30% and the scores of classes 1 and 5 have decreased by a tiny bit but the precision increased and recall decreased which means better at identifying false positives but became worse at identifying false negatives for these two classes. For Coursera, Bag-of-Words (LR) behaves similarly to Tf-Idf as both of these algorithms predict sentiment through word vectorization which makes them of similar nature. Among these two algorithms BOW-LR performs better than tf-idf-LR as the f1 scores are overall better for BOW. The precision, recall and f1 scores improves after oversampling. In addition, for Naive Bayes in Coursera, using the MultinomialNB classifier, the results for sentiment classes from 1-3 are very low where all of them are below 10% although for class 1 we see precision of 75% which means it does not predict many False Positives, the recall is only 5% meaning it predicts many false negatives and the recall values for these three are extremely low. Class 4 managed a f1 score of 30% with 41% precision and 23% recall. However, for sentiment 5 the precision, recall and f1-score are fairly impressive all above 80%. After oversampling, we see an improvement in the recall and f1 scores for classes 1-4 where they reached 19-41% from the prior 0-29% in recall and 16-39% from the prior 0-34% f1 scores. For Spotify, we see high f1 scores for class 1 and 5 where 5 being the highest 81%. Low f1 scores for 2-4 below 34%. After Oversampling, we see 1-5% decrease in the f1 scores for 1, 4 and 5 but increase for classes 2,3 of almost 16-19% for recall and f1 scores. Furthermore, for LSTM for Coursera, sentiment 5 has the highest score as usual, 82% precision, 94% recall and 88% f1 score while after oversampling all the scores become 90%. Class 4 performs the second best with 44-45% precision, recall and f1 score which is subpar. From classes 1-3 all the three matrices hold low scores with f1 scores, less than 34%; the model performed poorly for these three classes. After Oversampling we observe a drastic improvement for the f1 scores of all classes where all are above 53% and reaches up to 90%. Class 1, 2 and 3 has 78, 67 and 56% f1 scores now from previously seen 45, 10, 23%. For Spotify class 1 and 5 as always have given high scores for having higher sample rate; class 2-4 low scores with low precision and recall. After oversampling all the three classification metrices improves for these classes but we see the precision being lower than the recall here. Class 1 and 5 have high precision than recall after oversampling. Finally, for BERT we observe similar behavior as LSTM for both datasets. Class 5 having the highest score 83% followed by 1, 47% which is close to 50% but still substandard. Classes 2-4 have scores from 32-39% with lower precision (24-33%) and higher recall (42-49%) values which means they performed better with False negative detection than False positive. Among LSTM and BERT if we compare the f1 scores, we see that BERT does a better job at predicting the neutral and grey area classes 2, 3, 4 and also gives close values for class 1 and 5 before imbalance handling. After imbalance handling, LSTM gives overwhelmingly better performance over the pretrained BERT model used. For Sentiment 1 and 5 precision is higher than recall which means the model can predict a sentiment rating being of a particular class more precisely than it not being of that class. We observe an interesting pattern here which is that all five of the models do very well for sentiment score 5 (positive) metric but poorly for the negative metric (1); for neutral metric (3) and the 'gray areas' between the poles, the models almost cannot identify them at all. This is because the models are developing a heavy bias towards sentiment 5 due to the asymmetry in positive versus all other sentiment metrics ratio in the Coursera dataset. Coursera dataset is such that it is filed with positive reviews (5) and has very little negative, neutral of somewhat positive-negative review so the ratio is screwed and the models can only identify 5 as they have not seen other metrics much while training so that is why our algorithms could not perform well for other sentiment classes besides positive (5). For the Spotify App Review dataset, for sentiment 2, 3, 4 the precision, recall and f1- score are usually observed to be very low compared to the scores of sentiment classes 1 and 5. Sentiment class 5 has the highest scores for all the algorithms used. Sentiment class 1 gives better result than 2,3,4 by far and usually stays at a range closely above or below 50%. For the

Spotify dataset we no longer observe the overwhelming positive bias that we did for the Coursera dataset because for the Spotify dataset the ratio between positive and negative reviews are not as asymmetrical as the former one. They are in fact more balanced and the neutral and 'gray area' sentiment metrics are also present here in fair amount so the precision, recall, f1 scores are not giving values as absurd as it did for Coursera although we cannot term it as completely bias free since positive reviews are still more in amount for both datasets.

For 3 label sentiment classes:

Models	Original value without any sort of imbalance addressing (3 labels)		After imbalanced dataset addressing through Oversampling (3 labels)	
	Coursera's Course Review	Spotify App Review	Coursera's Course Review	Spotify App Review
TF-IDF (Logistic Regression)	91%	78.25%	87%	72.75%
BoW (Logistic Regression)	90.06%	77%	86.50%	72.55%
Naive Bayes (Multinomial-NB)	90.25%	78.2%	85.10%	72.2%
LSTM	89%	78.3%	96.45%	75.5%
BERT (base-uncase-multilingual-sentiment)	91.2%	77%		

Table 5.3: Accuracy Table for Sentiment 1-3

Accuracy score analysis: For 3 labels, we can see TF-IDF-LR gives 91% accuracy score which is higher than BOW-LR, Multinomial-Naïve Bayes and LSTM and neck-to-neck with the pretrained BERT scoring 91.2% for Coursera Dataset. Furthermore, for Spotify scores 78.25%, which is again higher than BOW, Naïve Bayes and BERT and 0.5% less than LSTM with the accuracy of 78.3%. It is interesting to see a simple algorithm like TF-IDF-LR gain on advanced state-of-the-art language models like BERT and improved neural network model like LSTM but we should also remember that accuracy score, no matter how high the score may be is a poor judge of performance when it comes to imbalanced datasets along with the fact that both of the datasets used for this research were more or less imbalanced so such unusual accuracy can be explain through closely analyzing the precision, recall and f1 scores for 3 labels. We can also see that after imbalanced handling the accuracy score went down a few notches for our models and this indicates how the algorithms really behaves when the class samples in the datasets are balanced. LSTM scores the best after bias-addressing our datasets and overall, judging the three significant classification report metrices, both BERT and LSTM performs better than the three primitive models (TF-IDF-LR, BOW-LR, Multinomial-Naïve Bayes) seen previously.

Models	Original value without any sort of imbalance addressing (3 labels)							After imbalanced dataset addressing through Oversampling/ Up-sampling (3 labels)					
	Coursera's Course Review				Spotify App Review			Coursera's Course Review			Spotify App Review		
TF-IDF (Logistic Regression)	Senti-ment	Precision	Recall	f1-score	Precision	Recall	f1-score	Precision	Recall	f1-score	Precision	Recall	f1-score
	1	0.86	0.27	0.41	0.72	0.83	0.77	0.51	0.66	0.57	0.75	0.67	0.71
	2	0.33	0.03	0.06	0.25	0.02	0.04	0.15	0.22	0.18	0.22	0.39	0.28
	3	0.91	1.00	0.95	0.83	0.90	0.86	0.96	0.91	0.94	0.89	0.82	0.85
BoW (Logistic Regression)	1	0.69	0.50	0.58	0.74	0.77	0.76	0.48	0.60	0.53	0.74	0.68	0.71
	2	0.21	0.10	0.14	0.23	0.11	0.15	0.14	0.18	0.16	0.21	0.31	0.25
	3	0.93	0.98	0.96	0.83	0.89	0.86	0.95	0.92	0.94	0.86	0.83	0.84
Naive Bayes (Multinomial-NB)	1	0.66	0.26	0.38	0.69	0.86	0.77	0.48	0.71	0.58	0.75	0.70	0.72
	2	0.50	0.01	0.02	0.25	0.02	0.04	0.19	0.38	0.26	0.22	0.42	0.29
	3	0.91	0.99	0.95	0.86	0.87	0.87	0.97	0.89	0.93	0.91	0.79	0.85
LSTM	1	0.53	0.36	0.43	0.73	0.82	0.77	0.92	0.81	0.86	0.78	0.76	0.77
	2	0.17	0.13	0.15	0.30	0.09	0.14	0.79	0.65	0.71	0.24	0.46	0.32
	3	0.93	0.97	0.95	0.85	0.89	0.87	0.97	0.99	0.98	0.93	0.80	0.86
BERT (base-uncased-multilingual-sentiment)	1	0.58	0.71	0.64	0.71	0.86	0.78						
	2	0.31	0.42	0.36	0.26	0.30	0.28						
	3	0.98	0.95	0.96	0.93	0.80	0.86						

Table 5.4: Precision, Recall and F1-Score (1-3) of Models

Classification report analysis: For 3 label classification of Coursera course review dataset, the sentiment class 2 for precision, recall and f1-score are all low 33%, 3%, 6% respectively, when the model is Tf-Idf (LR). For class 1, the f1 score is 41% with high precision and a low recall rate of 26%. Class 3 gives f1 score of 95%, the highest of the three with precision of 91% and 100% recall which means it does not predict any false negatives. After oversampling Class 2 f1score increases to 28% with increase in precision and recall rates; class 1 increases to 71% f1 score. For Spotify App Review dataset, it gives the high scores for class 1 and 5, 77 and 86% but class 2 lies at the lowest f1 score 4% which is disappointing. After oversampling, class 2 score increases, class 1 scores decreases and class 3 reaches a whopping 94%. For Coursera, Bag-of-Words (LR) perform better Tf-Idf in accordance to their f1 and recall scores. Class 2 is still below 15% while class 1 and 5 are above 55% and 95%. The precision, recall and f1 scores does not improve much after oversampling but only by 4 or 5%. Moreover, For Spotify, the algorithm gives similar results before and after imbalance handling although the scores for Spotify in classes 1 and 2 are better than that of Coursera. In addition, for Naive Bayes (MultinomialNB) behaves the same way as TF-IDF-LR and BOW-LR for the three classes prediction. For class 1, 2 and 3 - 38%, 2% and 95% f1 scores respectively. For Coursera it gives higher precision, recall than BOW in class 1 while lower for the same metrics under Spotify. After Oversampling, however, the changes are higher than that of BOW after oversampling for both datasets. The f1 score for class 2 increases to 26% and 29% respectively. Furthermore, for LSTM for Coursera, sentiment 3 has the highest score as usual, 93% precision, 97% recall and 95% f1 score while after oversampling all the scores reach up to 98%. Class 1 performs close to 45% which is less than 50% so it can be considered subpar. The model performed poorly for the neutral class (2) - 15%. After Oversampling we observe a drastic improvement for the f1 scores of all classes where all are above 60% and reaches up to 99% in recall with impressive f1 scores – 86% 71%, 98%. For Spotify, Class 1, 3 perform above 75% which is satisfactory but class 2 score is only 14% which is very low and it improved after oversampling by 20% at least, while f1 score stays the same for class 1 and deceases by 1 % for 3. Finally, for BERT class 1 haves the highest score for Coursera which is 64% and 2, 3 gives low f1 score of 36% each. The recall values are higher for class 1 and 2 than precision and for class 3 precision is higher than recall. For Spotify class 1 and 3 are showing high performance; class 5 has the highest score amongst the three and class 2 as usual gives lower performance. The recall value is higher than precision for this class so the model did a little bit better at not predicting false negatives than false positives for 2 sentiment score. Amongst the five models LSTM and BERT gives the best performances overall at predicting the neutral class 2, along with the positive (3) and negative class (1) before imbalance handling. Comparing f1 scores of LSTM and BERT, BERT is better. After imbalance handling, LSTM does better than BERT especially for Coursera dataset and for Spotify predicts neutral class at a higher rate comparatively.

We see that all the models perform significantly higher for the simplified 3 label classification of positive, negative and neutral than the previously 5 label classification of positive, somewhat positive, neutral, somewhat negative and negative. This is because reviews under the labels 'somewhat positive' and 'somewhat negative' can greatly confuse the algorithm as such concepts are abstract and subjective and the models and arrange data into positive and negative but get confused where to put reviews that are not entirely positive or negative, especially with the presence of 'neutral' label there. So, basically most algorithms, no matter how complex, cannot fully grasp the concept of 'gray areas' between the two poles/ extremes and do not seem to understand neutral data either, therefore they mis-predict such data.

Imbalance Handling in Dataset:

The datasets we used for our research has asymmetric number of samples for each class especially where there is a big different in the distribution of the classes and this causes our ML models to develop a bias towards the class with the highest number of samples and this bias works against other classes with lesser number of samples respectively. As a result, our accuracy score becomes skewed and unreliable as the ML algorithms develop the tendency to predict the dominant classes while ignoring the minority class. Such events run the risk of algorithms entering an accuracy paradox which is such a state where the accuracy score gives a high accuracy giving the impression of high performance but in actuality the model just predicts the class with the most information available, i.e., the most data available in the training set and is the easiest to predict and predicts those classes predominantly to acquire the highest accuracy score. The two datasets we used for this research, the Coursera Course Review Dataset in particular is one such that there is a huge difference among the distribution of the most frequent classes such as the positive class (5) and the least frequent classes such as class 2 (somewhat-negative). The Spotify App Review dataset also carries such imbalances but in Spotify classes 5 and 1 have lesser difference among their sample size compared to Coursera but the rest of the classes prove to have bigger differences in their sample sizes being compared to the majority class. Simple algorithms such as TF-IDF, BOW, Logistic Regression and Naïve Bayes do not have reliable mechanisms in them internally to be able to handle the problems that arises with imbalanced data chucks. Since we are observing the behavior of the five types of models we used on certain text classification problems in sentiment analysis, a skewed, unreliable and incorrect prediction rate is of little value. Therefore, we have decided to address this imbalance with the help of two techniques: 1. Class-weight computation and 2. Randomly oversampling/ Up-sampling the samples belonging to the minority classes to the sample size of the majority class. In Class_weight computation, we assign certain weight to the classes present in our dataset such that different weights are assignment for the majority and minority classes. The difference in weights will influence the classification of the classes during the training phase. The whole purpose is to penalize the misclassification made by the minority class by setting a higher class_weight and at the same time reducing weight for the majority class. We imported the class_weight from the 'sklearn.utils' library and used the 'compute_class_weight' method which automatically sets the class weights for the classes present in accordance to their sample size such that the most frequent and easy to predict class gets the least precedence over the class with the lowest samples and hardest to predict. Misclassification of the classes is penalized in accordance to the weights set. In random Oversampling we increase the sizes of the minority classes each to that of the majority class or the most frequent class in the dataset by randomly resampling them to the size of the most frequent class in the training data. This gives us an equal distribution of all the classes present in the dataset to work with during train then we test the ML algorithm with some testing data to evaluate the changes in precision, recall and f1 scores. Below is given two tables comparing the accuracy score and classification metrics of these two data-balancing techniques used on four of our models, TF-IDF-LR, BOW-LR, Multinomial Naïve Bayes and LSTM to observe the improvements in prediction rate.

Models	Class-Weight(5 labels)		Random Oversampling / Up-sampling (5 labels)		Class-Weight (3 labels)		Random Oversampling/ Up-sampling (3 labels)	
	Coursera's Course Review	Spotify App Review	Coursera's Course Review	Spotify App Review	Coursera's Course Review	Spotify App Review	Coursera's Course Review	Spotify App Review
TF-IDF (LR)	64.96.11%	51.15%	63.31%	52.35%	86.30%	69.3%	87%	72.75%
BOW (LR)	65.36%	50.5%	65.36%	52.75%	85.95%	68.7%	86.50%	72.55%
Naive Bayes (Multinomial-NB/ complement-NB)	74.81%	61.9%	67.52%	52.95%	90.04%	78.05%	85.10%	72.2%
LSTM	74.76%	62.7%	81.30%	61.15%	86%	73.7%	96.45%	75.5%

Table 5.5: Accuracy Table for Class Weight and Oversampling of Models

Accuracy analysis: From the above tables we can see that the accuracy of these two techniques go neck-to-neck which means that both class-weight and Random oversampling give close results for the two datasets. For Coursera dataset, we can see that TF-IDF-LR generates higher accuracy score for 5 labels with class weights than using random oversampling technique which is 1.6% higher. However Random oversampling gives 0.7% higher score than class-weights for 3 label classification. BOW-LR gives the same results for both for 5 label classification. For 3 labels, oversampling performs better. For Naïve bayes class-weights give better accuracy for both 5 label and 3 label sentiment classification with almost 5-7% gain over random oversampling. For LSTM Random oversampling gives better results than class weights by a drastic 7-10% increase in comparison. For Spotify dataset, TF-IDF-LR gives better score for oversampling than being weighted by balanced class weights in both cases of labels. BOW-LR also gives better results for oversampling than class-weights by 2-4%. For Complement-Naïve Bayes, a variation of Multinomial Naïve bayes specifically designed to handle class/ sample weights, we see higher value of class weights than oversampling for both labels. For LSTM for 5 labels oversampling gives better accuracy while for 3 labels oversampling gives better results.

Models	Class-weight Computation (5 labels)							Random Oversampling/ Up-sampling (5 labels)					
	Coursera's Course Review				Spotify App Review			Coursera's Course Review			Spotify App Review		
TF-IDF (Logistic Regression)	Senti-ment	Precision	Recall	f1-score	Precision	Recall	f1-score	Precision	Recall	f1-score	Precision	Recall	f1-score
	1	0.40	0.21	0.28	0.54	0.59	0.56	0.35	0.45	0.39	0.60	0.52	0.56
	2	0.16	0.12	0.13	0.15	0.20	0.17	0.15	0.17	0.16	0.18	0.27	0.22
	3	0.20	0.23	0.20	0.18	0.21	0.19	0.17	0.27	0.21	0.18	0.24	0.21
	4	0.29	0.41	0.32	0.26	0.29	0.27	0.29	0.38	0.33	0.25	0.26	0.26
	5	0.84	0.78	0.81	0.85	0.68	0.76	0.86	0.75	0.80	0.83	0.74	0.78
BOG (Logistic Regression)	1	0.41	0.29	0.34	0.54	0.59	0.56	0.38	0.42	0.40	0.59	0.50	0.54
	2	0.15	0.15	0.15	0.15	0.20	0.17	0.15	0.19	0.17	0.13	0.17	0.15
	3	0.19	0.23	0.21	0.18	0.21	0.19	0.17	0.24	0.20	0.18	0.22	0.20
	4	0.29	0.37	0.33	0.26	0.29	0.27	0.31	0.35	0.33	0.28	0.28	0.28
	5	0.85	0.79	0.82	0.85	0.68	0.76	0.85	0.79	0.82	0.80	0.78	0.79
Naive Bayes (Multinomial-NB/ Complement NB)	1	0.20	0.50	0.29	0.62	0.57	0.59	0.31	0.36	0.34	0.61	0.50	0.55
	2	0.10	0.27	0.14	0.21	0.30	0.21	0.14	0.19	0.16	0.18	0.33	0.24
	3	0.14	0.29	0.19	0.19	0.17	0.18	0.19	0.35	0.25	0.19	0.23	0.21
	4	0.30	0.28	0.29	0.30	0.28	0.27	0.37	0.41	0.39	0.29	0.34	0.32
	5	0.88	0.71	0.79	0.82	0.75	0.79	0.89	0.80	0.84	0.87	0.73	0.79
LSTM	1	0.26	0.17	0.20	0.61	0.71	0.66	0.82	0.75	0.78	0.71	0.59	0.65
	2	0.19	0.50	0.28	0.21	0.24	0.23	0.69	0.66	0.67	0.25	0.40	0.31
	3	0.14	0.16	0.15	0.27	0.31	0.29	0.50	0.64	0.56	0.27	0.36	0.31
	4	0.29	0.28	0.29	0.37	0.38	0.37	0.55	0.52	0.54	0.41	0.46	0.43
	5	0.85	0.82	0.83	0.87	0.72	0.79	0.90	0.90	0.90	0.89	0.77	0.82

Table 5.6: Precision, Recall and F1-Score (1-5) of Models for Class-weight computation and Random Oversampling

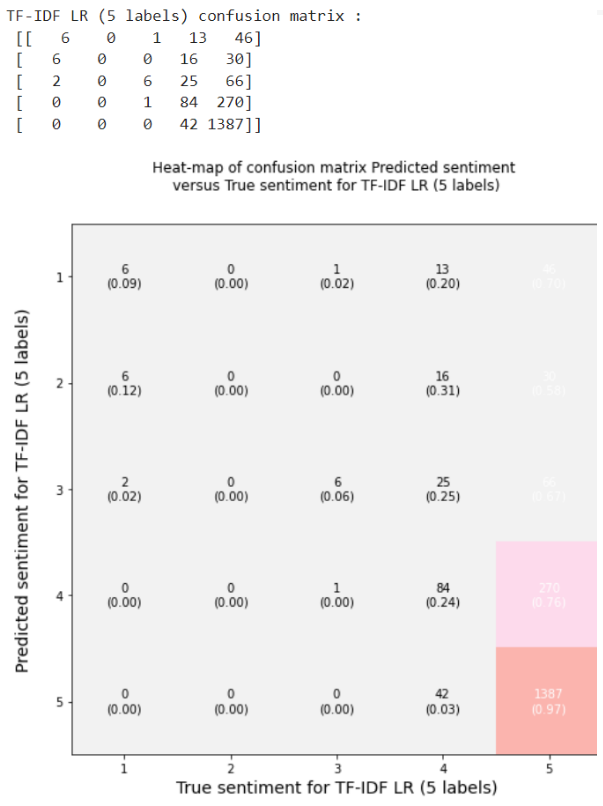
Models	Class-weight Computation (3 labels)							Random Oversampling/ Up-sampling (3 labels)					
	Coursera's Course Review				Spotify App Review			Coursera's Course Review			Spotify App Review		
TF-IDF (Logistic Regression)	Senti-ment	Precision	Recall	f1-score	Precision	Recall	f1-score	Precision	Recall	f1-score	Precision	Recall	f1-score
	1	0.51	0.38	0.44	0.67	0.72	0.69	0.51	0.66	0.57	0.75	0.67	0.71
	2	0.17	0.28	0.21	0.21	0.36	0.26	0.15	0.22	0.18	0.22	0.39	0.28
	3	0.95	0.93	0.94	0.91	0.74	0.81	0.96	0.91	0.94	0.89	0.82	0.85
BOG (Logistic Regression)	1	0.47	0.37	0.42	0.67	0.68	0.67	0.48	0.60	0.53	0.74	0.68	0.71
	2	0.15	0.24	0.18	0.21	0.34	0.26	0.14	0.18	0.16	0.21	0.31	0.25
	3	0.95	0.93	0.94	0.87	0.76	0.81	0.95	0.92	0.94	0.86	0.83	0.84
Naive Bayes (Multinomial-NB / Complement NB)	1	0.50	0.69	0.58	0.69	0.86	0.76	0.48	0.71	0.58	0.75	0.70	0.72
	2	0.23	0.09	0.13	0.24	0.03	0.05	0.19	0.38	0.26	0.22	0.42	0.29
	3	0.95	0.96	0.95	0.87	0.87	0.87	0.97	0.89	0.93	0.91	0.79	0.85
LSTM	1	0.42	0.64	0.51	0.73	0.78	0.75	0.92	0.81	0.86	0.78	0.76	0.77
	2	0.16	0.20	0.18	0.22	0.36	0.28	0.79	0.65	0.71	0.24	0.46	0.32
	3	0.96	0.91	0.93	0.92	0.78	0.84	0.97	0.99	0.98	0.93	0.80	0.86

Table 5.7: Precision, Recall and F1-Score (1-3) of Models for Class-weight computation and Random Oversampling

Confusion Matrix and Heat-maps:

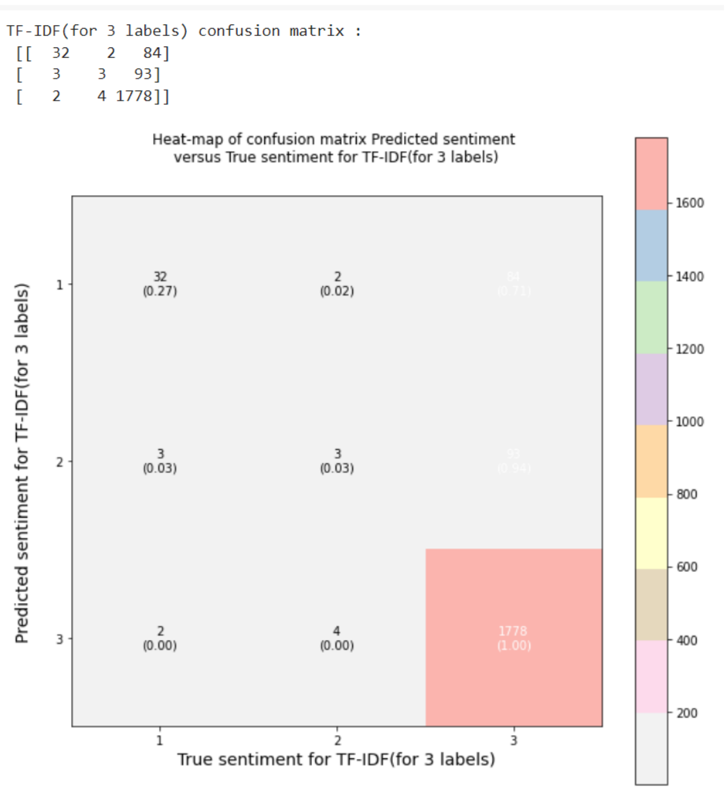
Here, all five models' confusion matrices and heat maps are given below to visualize their overall performances. On the left side, sentiment 1-5 confusion matrices and on the other side, sentiment 1-3 confusion matrices are placed to compare. The X axis of each graph represents the True sentiments classes in the test portion of the datasets for each model. By iterating through the heat-map row-wise, for each row summing the column values we will get the total number of sample distribution of that particular class in the test set. By iterating through the heat-map column-wise, for each column summing the row values we will get the total number of sample distribution of that particular class in the prediction set made by our models. If we compare both the distributions in the test set versus the prediction set, we will understand the pattern in which models predicted the given text data and how accurate they may be with such predictions. The colour scale from lighter to darker represents how accurate the predictions are- darker or brighter colours represents high accuracy, lighter or dull colours represent low accuracy. If we go diagonally through the heat map cells, we will get the actual accuracy of the model for those particular classes. The normalized values from 0 - 1 represents the accuracy of predictions made.

At first, all the original sentiment confusion matrices are placed. Then, all the class weight matrices are on the left and the oversampling confusion matrices are on the right side. So that we can compare the performance and improvement at once.



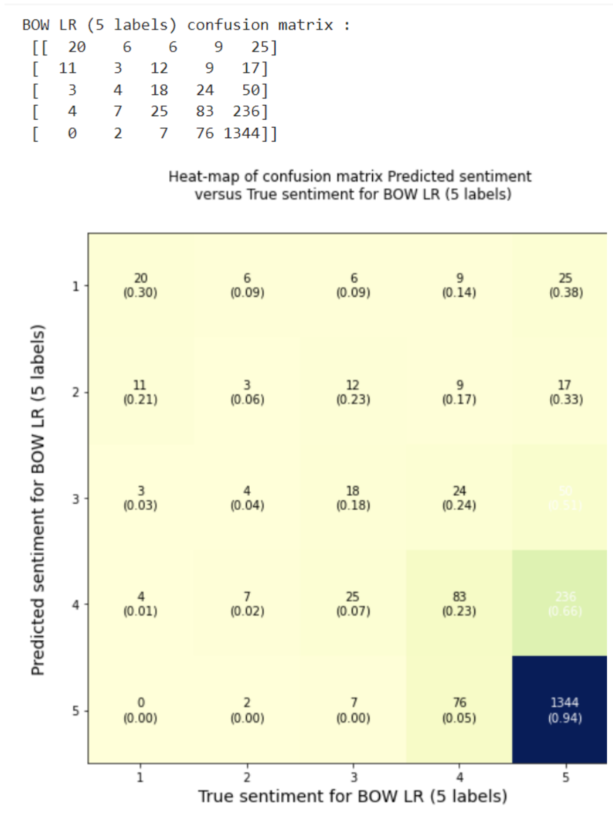
```
TF-IDF LR (5 labels) confusion matrix :
 [[   6    0    1   13   46]
 [   6    0    0   16   30]
 [   2    0    6   25   66]
 [   0    0    1   84  270]
 [   0    0    0   42 1387]]
```

Figure 5.1: Sentiment 1-5 for Coursera in Tf-IDF



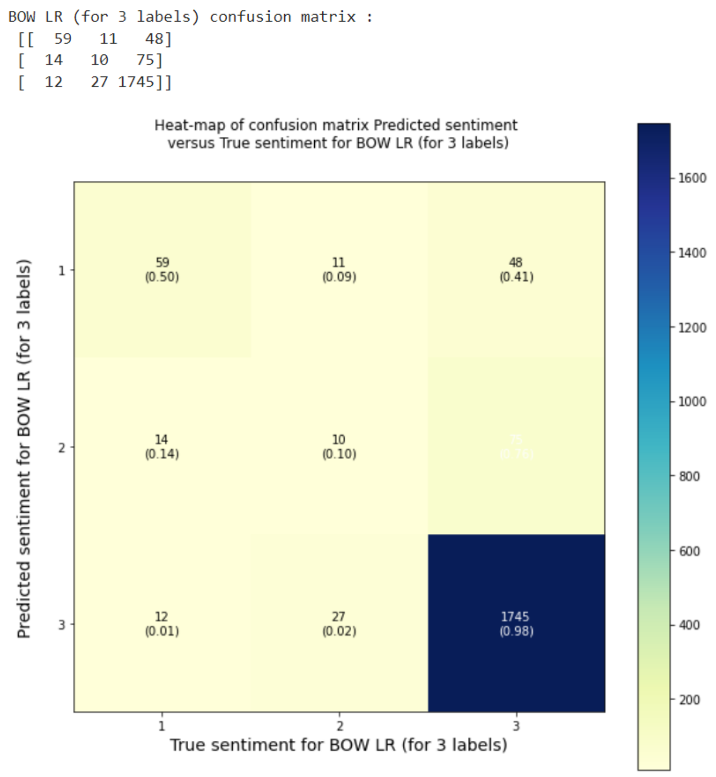
```
TF-IDF(for 3 labels) confusion matrix :
 [[  32    2   84]
 [   3    3   93]
 [   2    4 1778]]
```

Figure 5.2: Sentiment 1-3 for Coursera in Tf-IDF



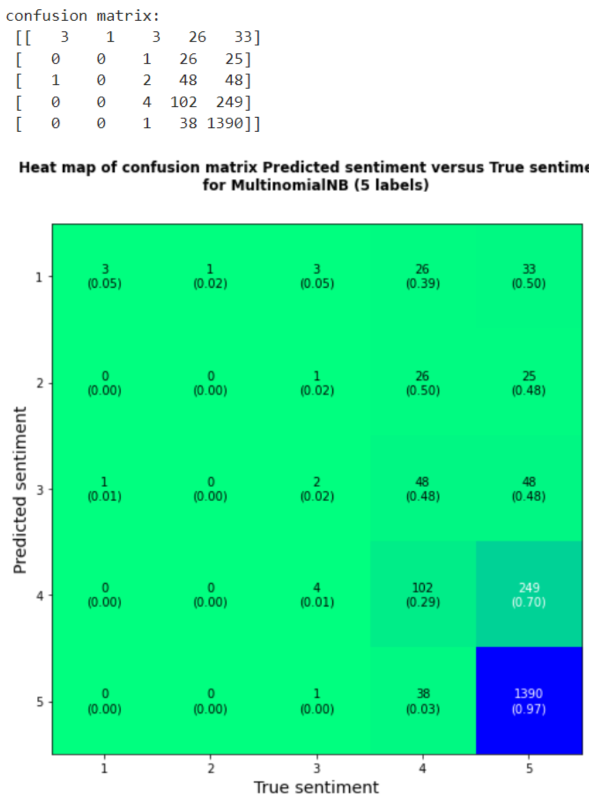
```
BOW LR (5 labels) confusion matrix :
 [[  20    6    6    9   25]
 [  11    3   12    9   17]
 [   3    4   18   24   50]
 [   4    7   25   83  236]
 [   0    2    7   76 1344]]
```

Figure 5.3: Sentiment 1-5 for Coursera in BOW



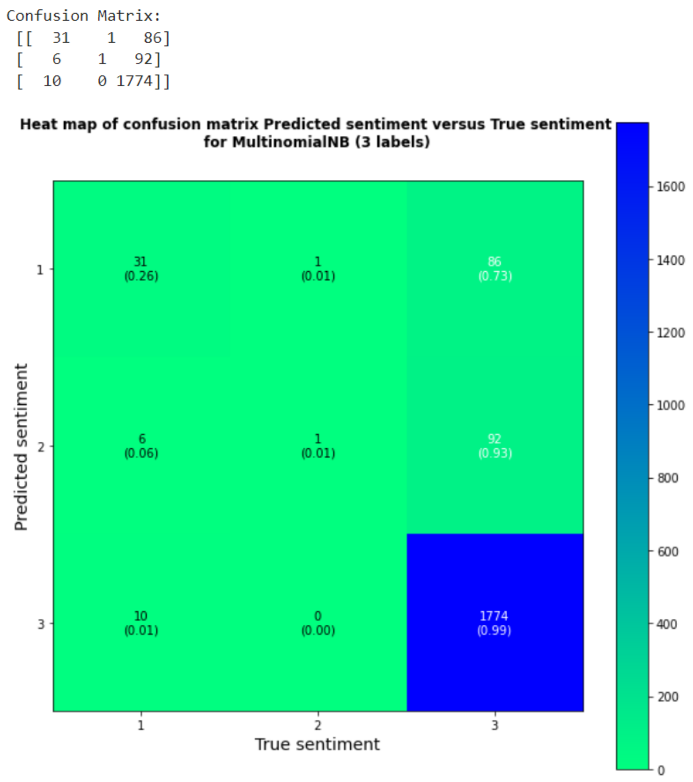
```
BOW LR (for 3 labels) confusion matrix :
 [[  59   11   48]
 [  14   10   75]
 [  12   27 1745]]
```

Figure 5.4: Sentiment 1-3 for Coursera in BOW



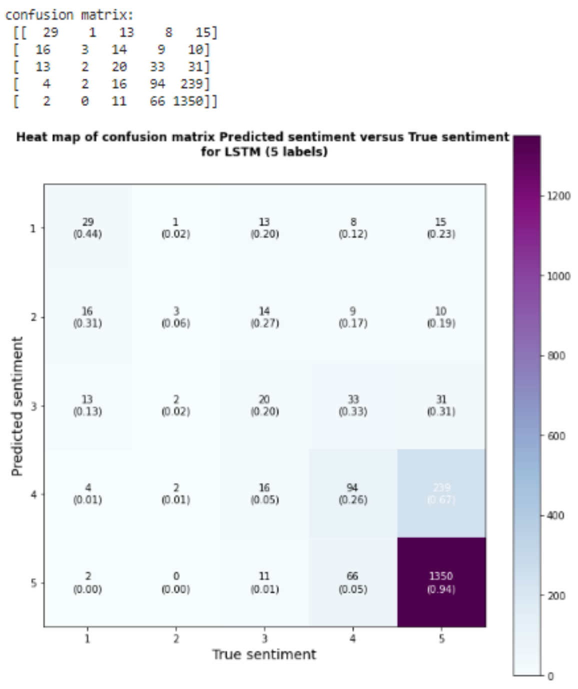
```
confusion matrix:
 [[   3    1    3   26   33]
 [   0    0    1   26   25]
 [   1    0    2   48   48]
 [   0    0    4  102  249]
 [   0    0    1   38 1390]]
```

Figure 5.5: Sentiment 1-5 for Coursera in Naive Bayes



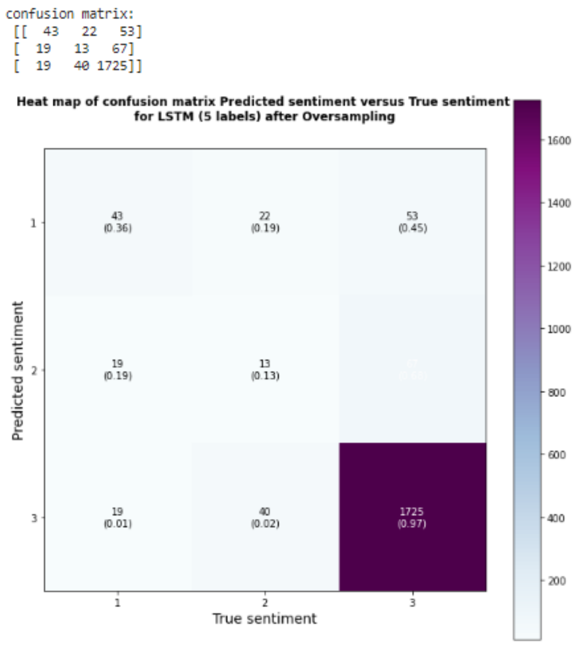
```
Confusion Matrix:
 [[  31    1   86]
 [   6    1   92]
 [  10    0 1774]]
```

Figure 5.6: Sentiment 1-3 for Coursera in Naive Bayes



```
confusion matrix:
 [[  29    1   13    8   15]
 [  16    3   14    9   10]
 [  13    2   20   33   31]
 [   4    2   16   94  239]
 [   2    0   11   66 1350]]
```

Figure 5.7: Sentiment 1-5 for Coursera in LSTM



```
confusion matrix:
 [[  43   22   53]
 [  19   13   67]
 [  19   40 1725]]
```

Figure 5.8: Sentiment 1-3 for Coursera in LSTM

```
[[  21   22    8    3    0]
 [   4   19   12    4    0]
 [   5   25   42   27    1]
 [   1    7   52  165  123]
 [   4    5   22  295 1133]]
```

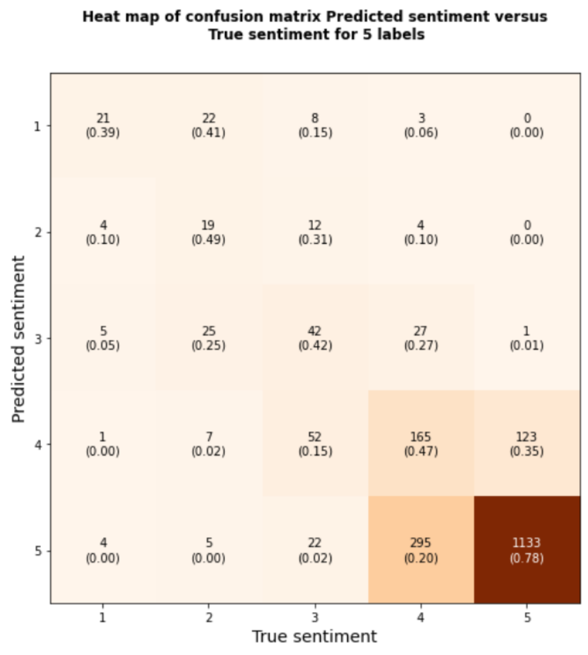


Figure 5.9: Sentiment 1-5 for Coursera in BERT

```
[[  66   20    7]
 [  30   42   28]
 [  17   74 1716]]
```

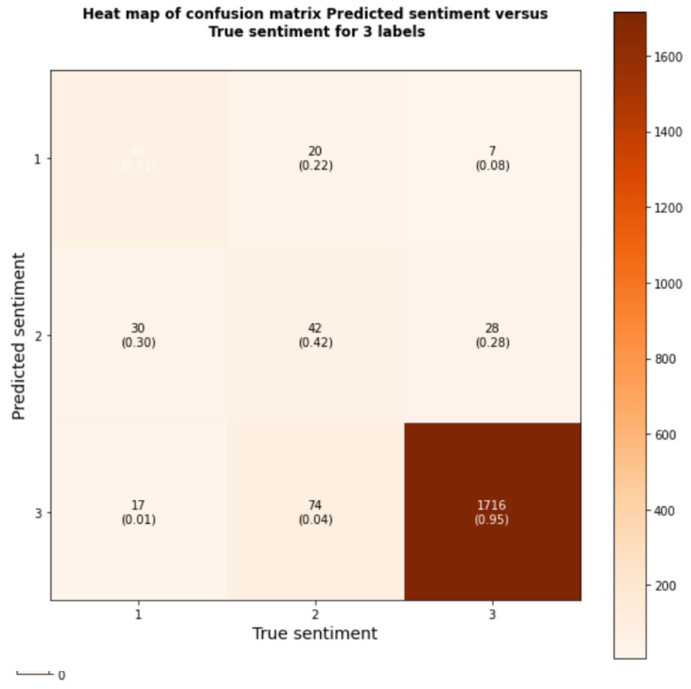


Figure 5.10: Sentiment 1-3 for Coursera in BERT

```
TF-IDF LR (5 labels) confusion matrix :
 [[413   6  11  12  63]
 [115  11  14  13  31]
 [ 95  10  15  25  59]
 [ 59   7  18  45 142]
 [ 51   5   6  33 741]]
```

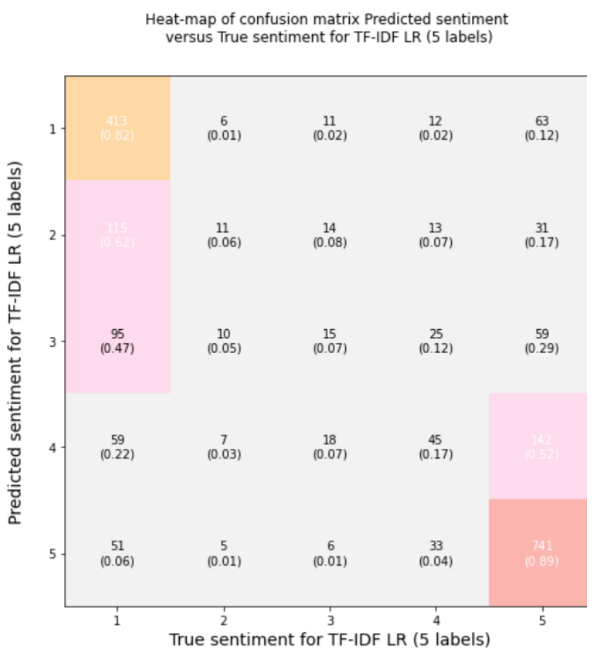


Figure 5.11: Sentiment 1-5 for Spotify in Tf-IDF

```
TF-IDF(for 3 labels) confusion matrix :
 [[570   4 115]
 [113   4  87]
 [108   8 991]]
```

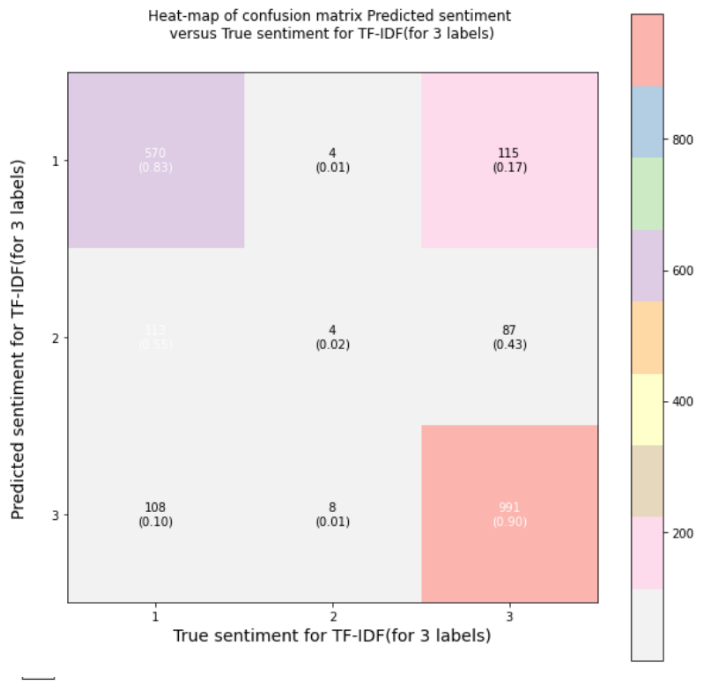
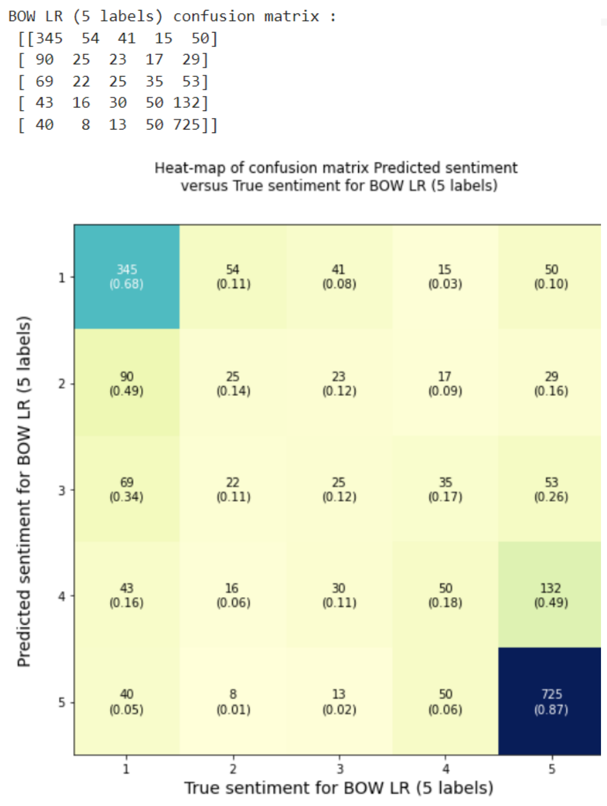
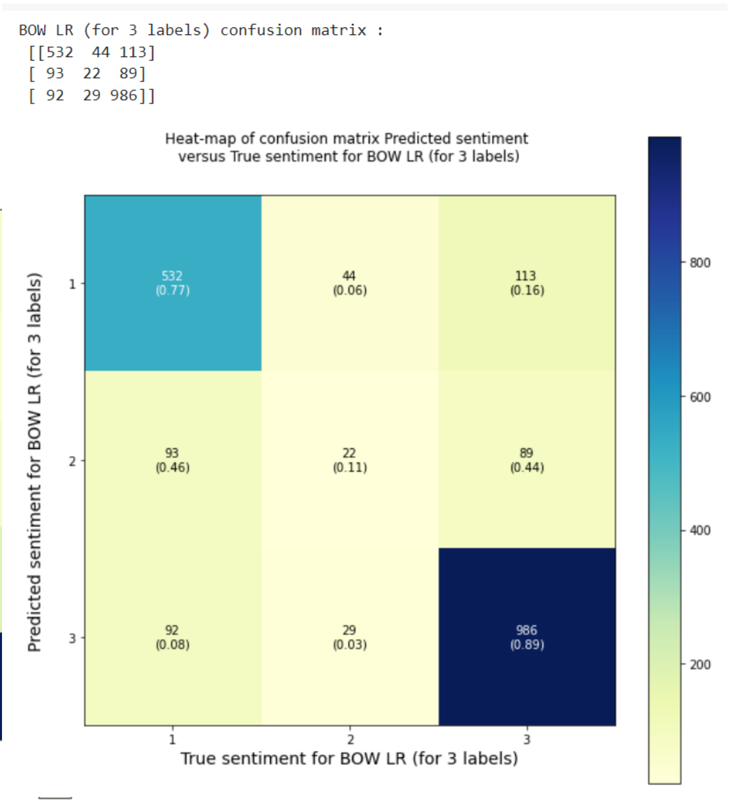


Figure 5.12: Sentiment 1-3 for Spotify in Tf-IDF



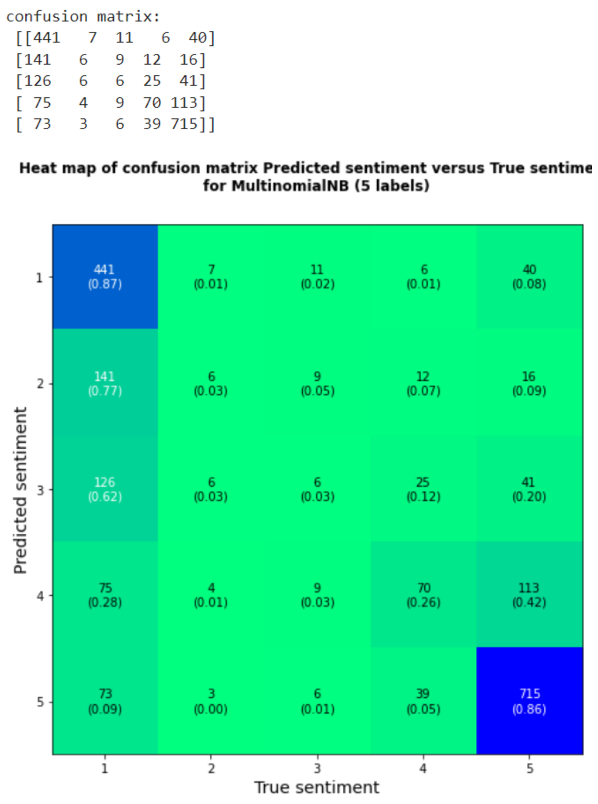
```
BOW LR (5 labels) confusion matrix :
 [[345  54  41  15  50]
 [ 90  25  23  17  29]
 [ 69  22  25  35  53]
 [ 43  16  30  50 132]
 [ 40   8  13  50 725]]
```

Figure 5.13: Sentiment 1-5 for Spotify in BOW



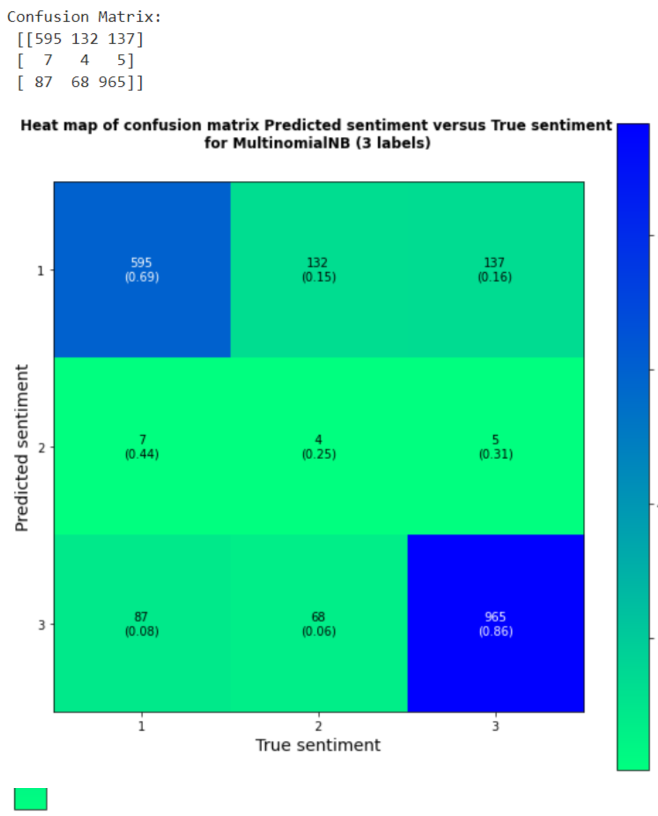
```
BOW LR (for 3 labels) confusion matrix :
 [[532  44 113]
 [ 93  22  89]
 [ 92  29 986]]
```

Figure 5.14: Sentiment 1-3 for Spotify in BOW



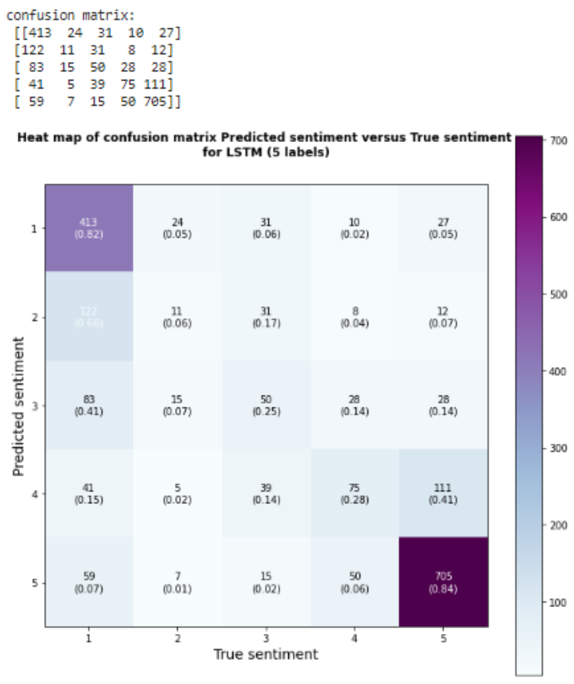
```
confusion matrix:
 [[441   7  11   6  40]
 [141   6   9  12  16]
 [126   6   6  25  41]
 [ 75   4   9  70 113]
 [ 73   3   6  39 715]]
```

Figure 5.15: Sentiment 1-5 for Spotify in Multinomial-Naive Bayes



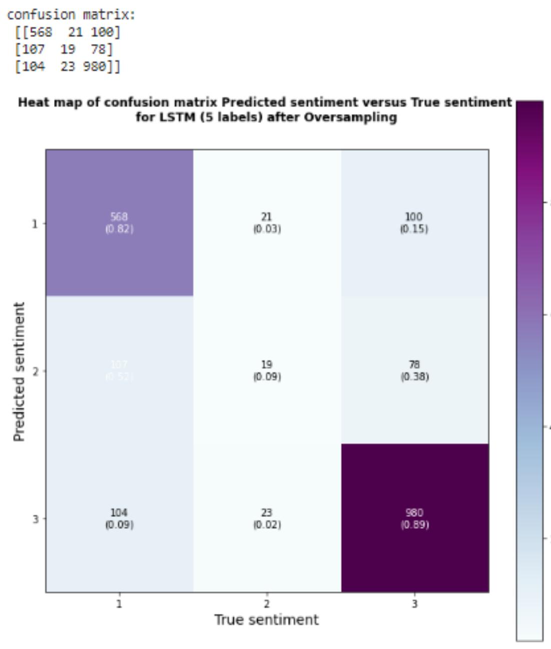
```
Confusion Matrix:
 [[595 132 137]
 [  7   4   5]
 [ 87  68 965]]
```

Figure 5.16: Sentiment 1-3 for Spotify in Multinomial-Naive Bayes



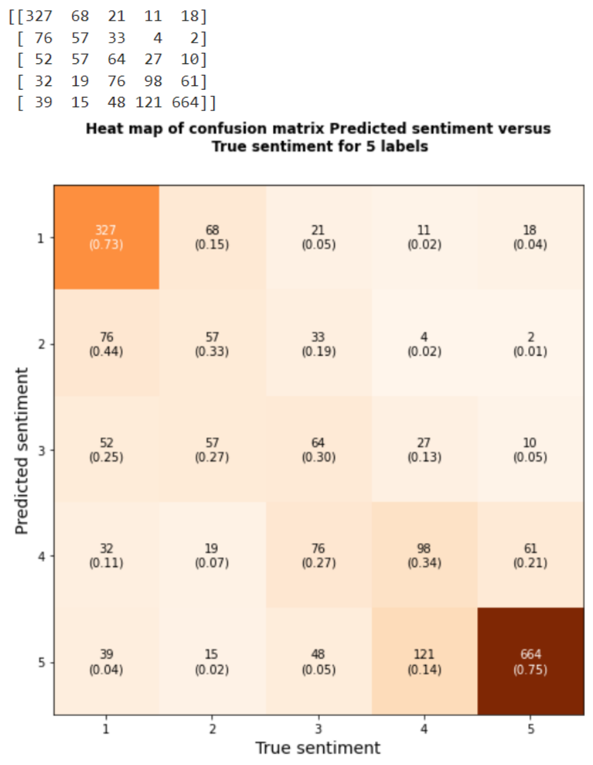
```
confusion matrix:
 [[413  24  31  10  27]
 [122  11  31   8  12]
 [ 83  15  50  28  28]
 [ 41   5  39  75 111]
 [ 59   7  15  50 705]]
```

Figure 5.17: Sentiment 1-5 for Spotify in LSTM



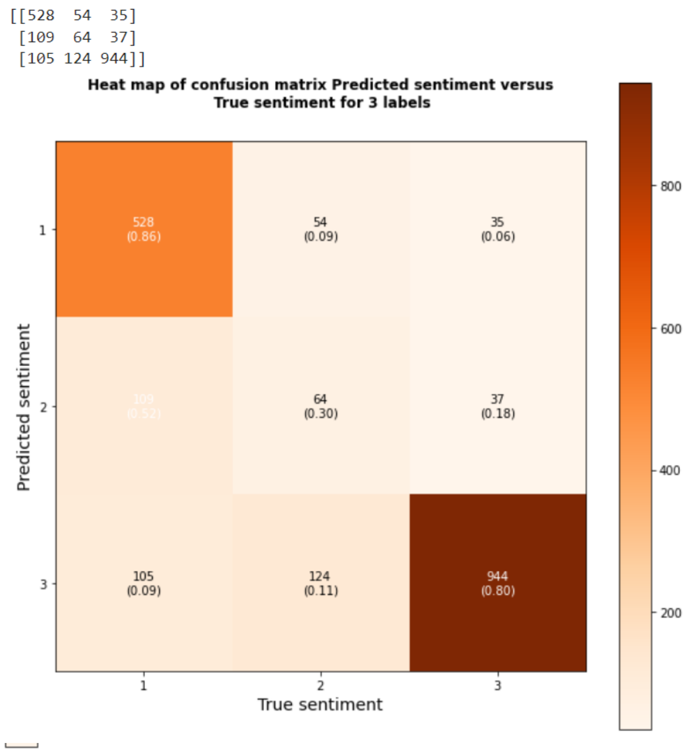
```
confusion matrix:
 [[568  21 100]
 [107  19  78]
 [104  23 980]]
```

Figure 5.18: Sentiment 1-3 for Spotify in LSTM



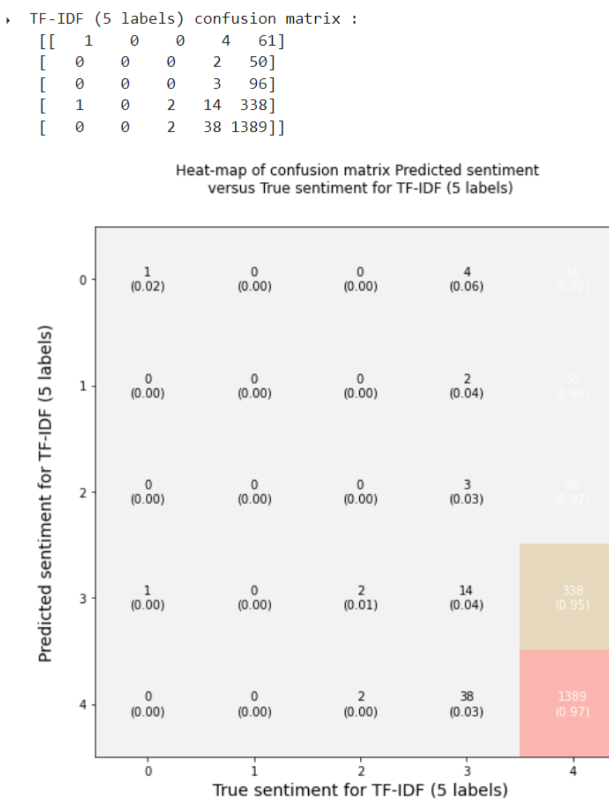
```
[[327  68  21  11  18]
 [ 76  57  33   4   2]
 [ 52  57  64  27  10]
 [ 32  19  76  98  61]
 [ 39  15  48 121 664]]
```

Figure 5.19: Sentiment 1-5 for Spotify in BERT



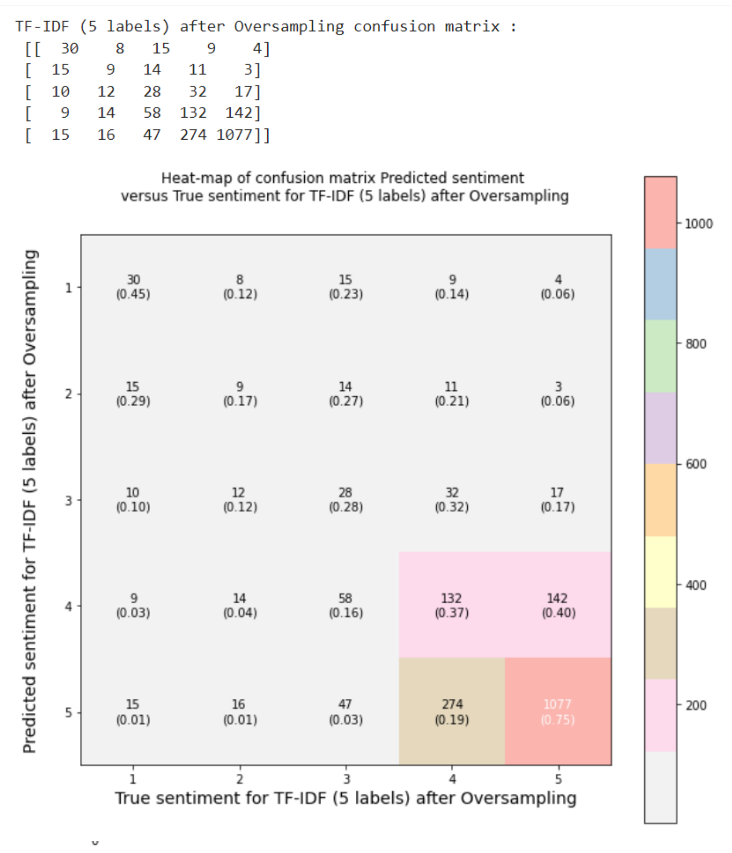
```
[[528  54  35]
 [109  64  37]
 [105 124 944]]
```

Figure 5.20: Sentiment 1-3 for Spotify in BERT



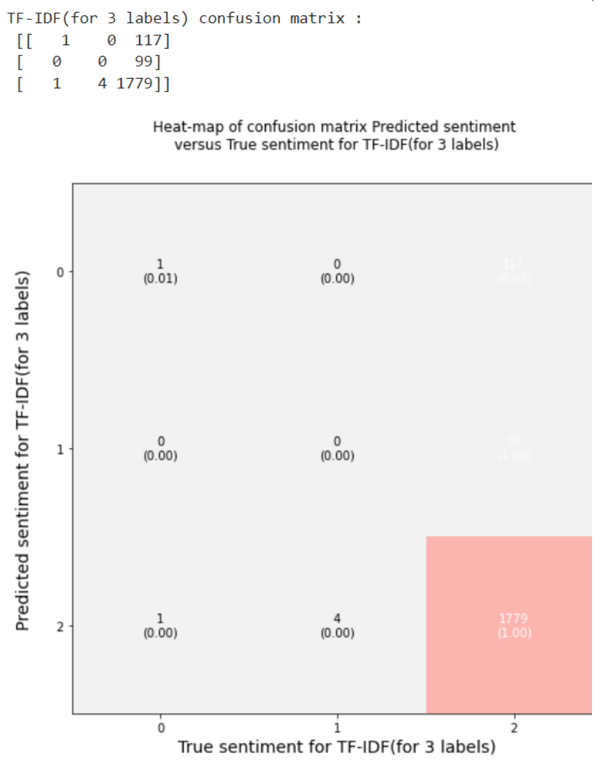
```
TF-IDF (5 labels) confusion matrix :
 [[   1    0    0    4   61]
  [   0    0    0    2   50]
  [   0    0    0    3   96]
  [   1    0    2   14  338]
  [   0    0    2   38 1389]]
```

Figure 5.21: Sentiment 1-5 for Coursera in TF-IDF after Class-weighting



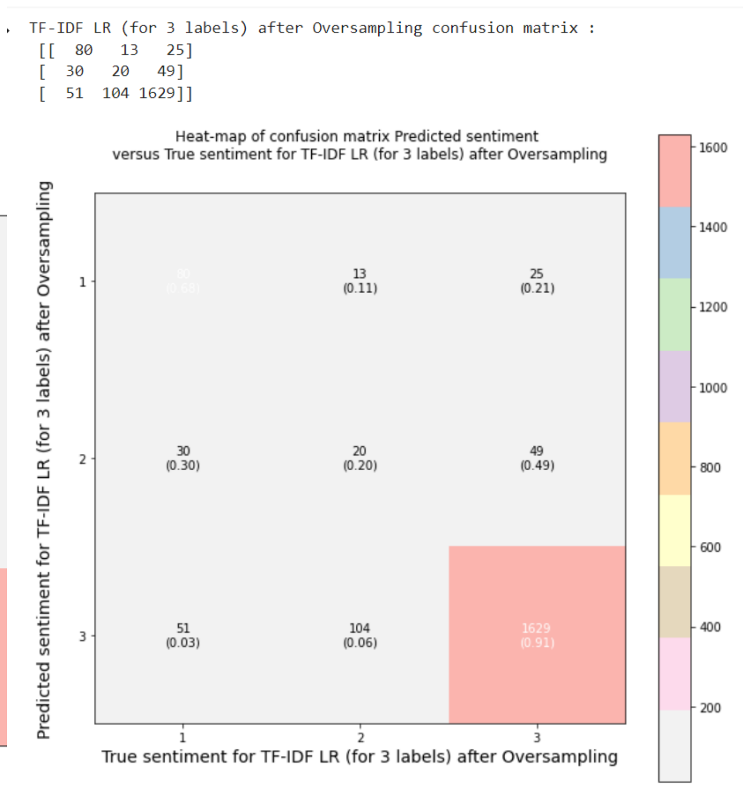
```
TF-IDF (5 labels) after Oversampling confusion matrix :
 [[  30    8   15    9    4]
  [  15    9   14   11    3]
  [  10   12   28   32   17]
  [   9   14   58  132  142]
  [  15   16   47  274 1077]]
```

Figure 5.22: Sentiment 1-5 for Coursera in TF-IDF after Oversampling



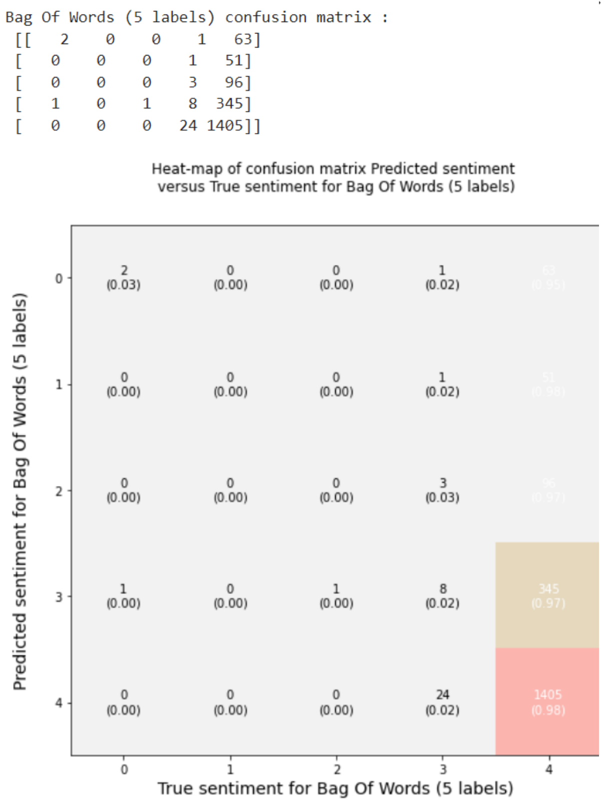
```
TF-IDF(for 3 labels) confusion matrix :
 [[   1    0  117]
 [   0    0   99]
 [   1    4 1779]]
```

Figure 5.23: Sentiment 1-3 for Coursera in TF-IDF after Class-weighting



```
TF-IDF LR (for 3 labels) after Oversampling confusion matrix :
 [[  80   13   25]
 [  30   20   49]
 [  51  104 1629]]
```

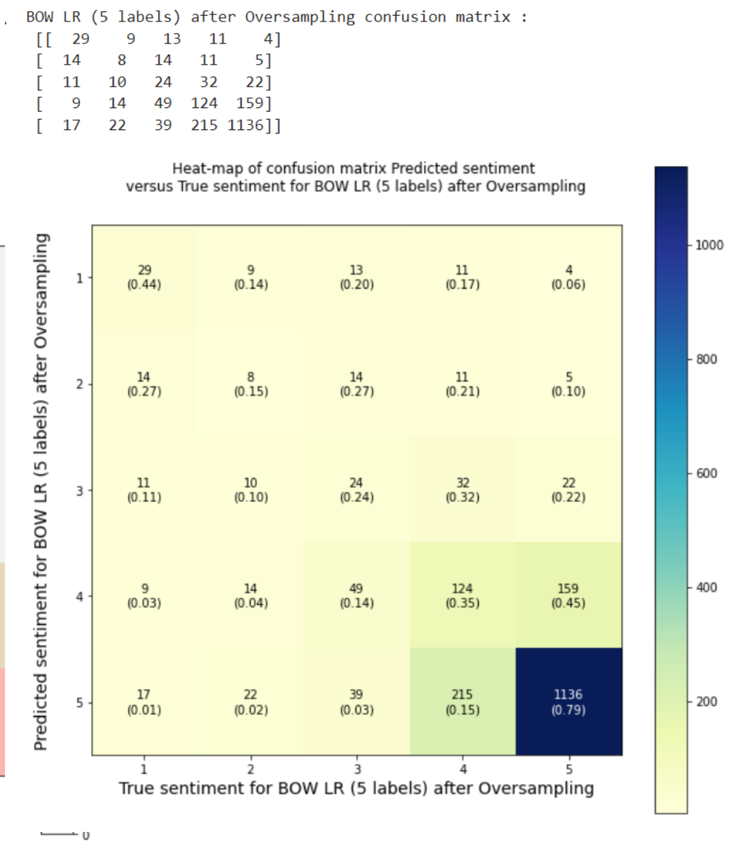
Figure 5.24: Sentiment 1-3 for Coursera in TF-IDF after Oversampling



```
Bag Of Words (5 labels) confusion matrix :
 [[   2    0    0    1   63]
 [   0    0    0    1   51]
 [   0    0    0    3   96]
 [   1    0    1    8  345]
 [   0    0    0   24 1405]]
```

Heat-map of confusion matrix Predicted sentiment
versus True sentiment for Bag Of Words (5 labels)
Predicted sentiment for Bag Of Words (5 labels)
True sentiment for Bag Of Words (5 labels)

Figure 5.25: Sentiment 1-5 for Coursera in BOW after Class-weighting



```
BOW LR (5 labels) after Oversampling confusion matrix :
 [[  29    9   13   11    4]
 [  14    8   14   11    5]
 [  11   10   24   32   22]
 [   9   14   49  124  159]
 [  17   22   39  215 1136]]
```

Heat-map of confusion matrix Predicted sentiment
versus True sentiment for BOW LR (5 labels) after Oversampling
Predicted sentiment for BOW LR (5 labels) after Oversampling
True sentiment for BOW LR (5 labels) after Oversampling

Figure 5.26: Sentiment 1-5 for Coursera in BOW after Oversampling

```
BOW(for 3 labels) confusion matrix :
 [[   2    0  116]
 [   0    0   99]
 [   1    2 1781]]
```

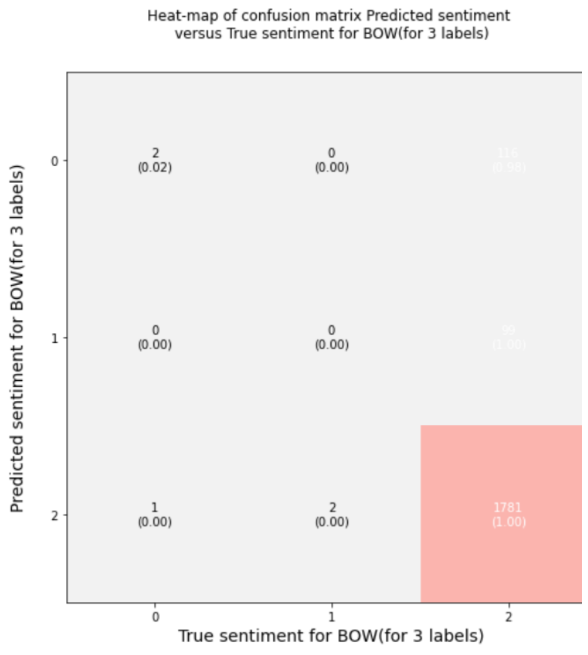


Figure 5.27: Sentiment 1-3 for Coursera in BOW after Class-weighting

```
BOW(for 3 labels) after Oversampling confusion matrix :
 [[  71   18   29]
 [  27   18   54]
 [  48   96 1640]]
```

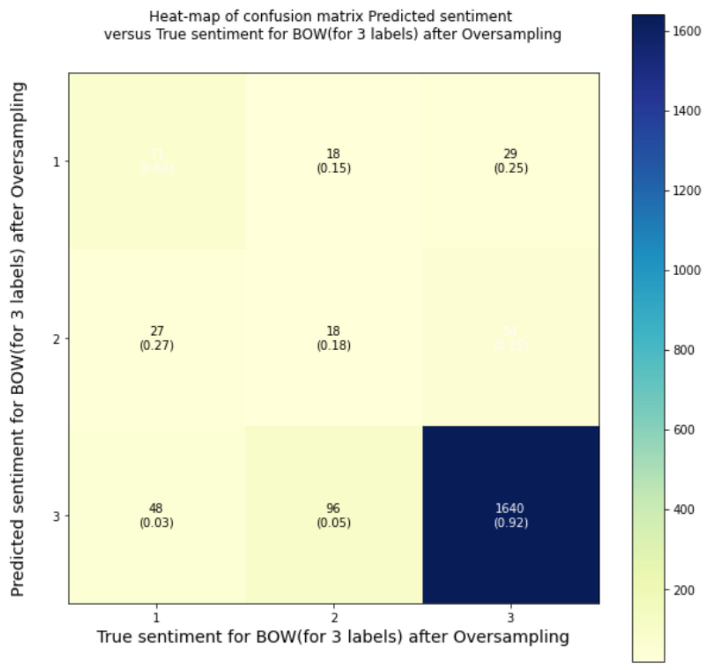


Figure 5.28: Sentiment 1-3 for Coursera in BOW after Oversampling

```
Confusion Matrix:
 [[  33   17   11    2    3]
 [  17   14   12    5    4]
 [  16   22   29   18   14]
 [  24   37   77   98  119]
 [  71   55   84  200 1019]]
```

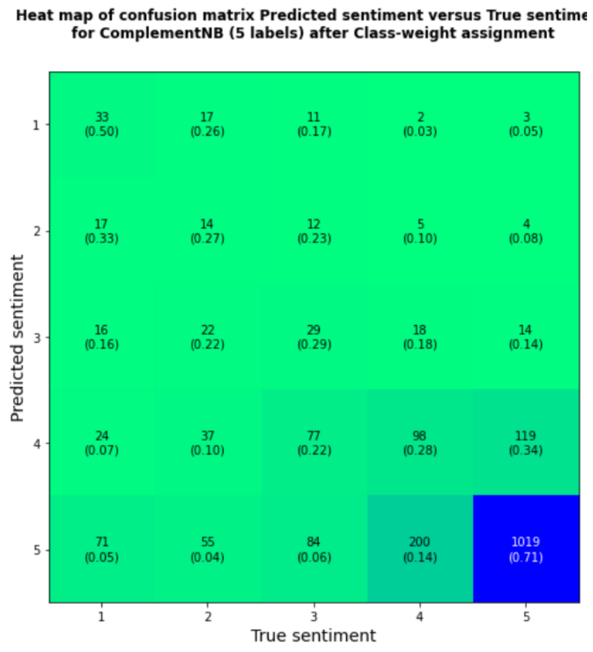


Figure 5.29: Sentiment 1-5 for Coursera in Multinomial-Naive Bayes after Class-weighting

```
Confusion Matrix:
[[  23   14   16    8    5]
 [  11   10   16   13    2]
 [  11    8   33   36   11]
 [   7   19   61  133  135]
 [  21   16   53  199 1140]]
```

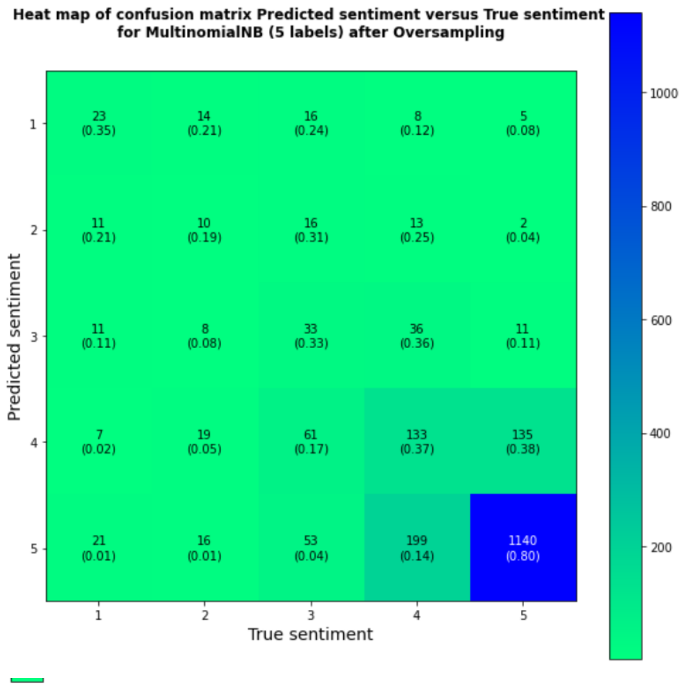
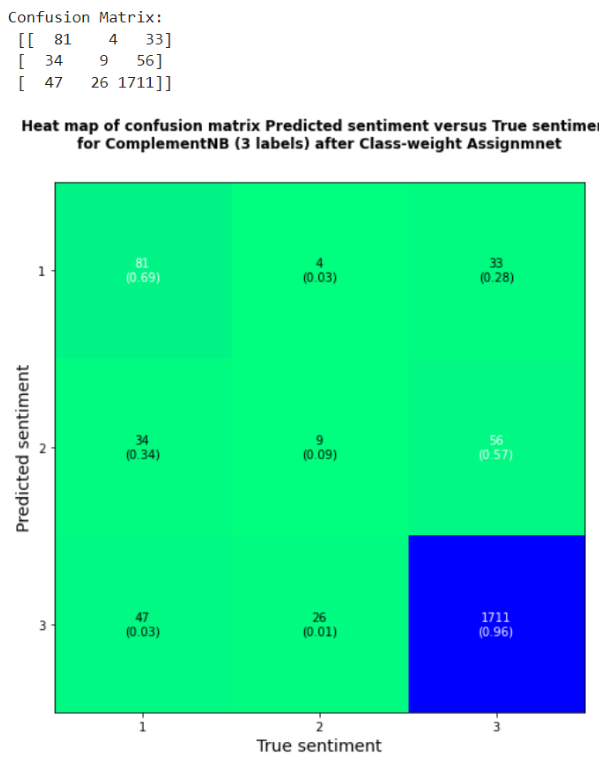
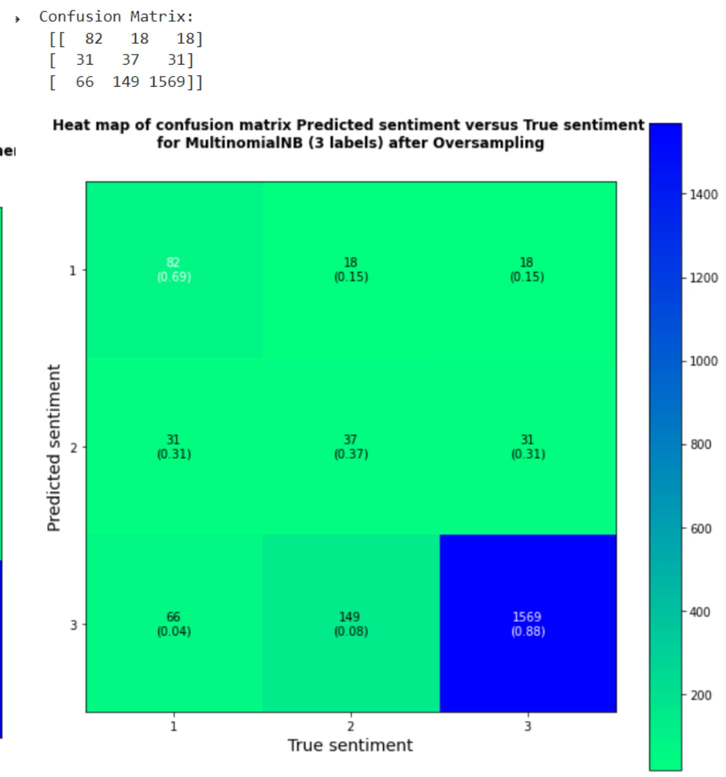


Figure 5.30: Sentiment 1-5 for Coursera in Multinomial-Naive Bayes after Oversampling



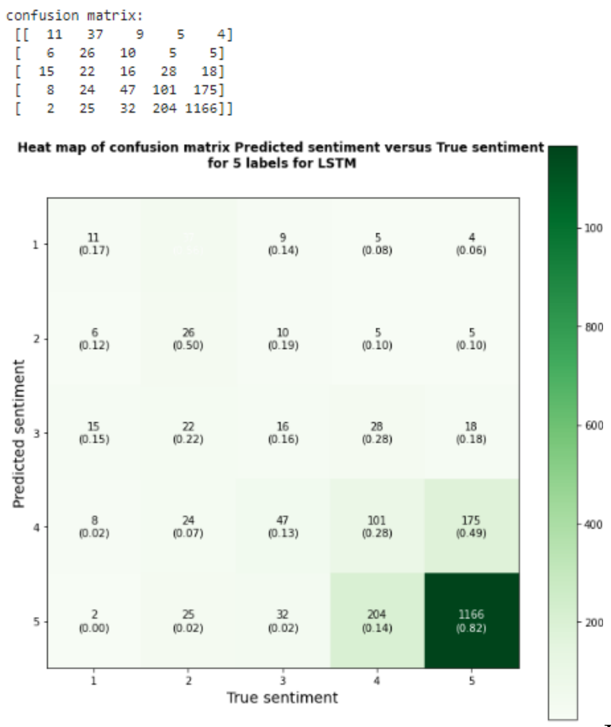
```
Confusion Matrix:
 [[  81    4   33]
 [  34    9   56]
 [  47   26 1711]]
```

Figure 5.31: Sentiment 1-3 for Coursera in Multinomial-Naive Bayes after Class-weighting



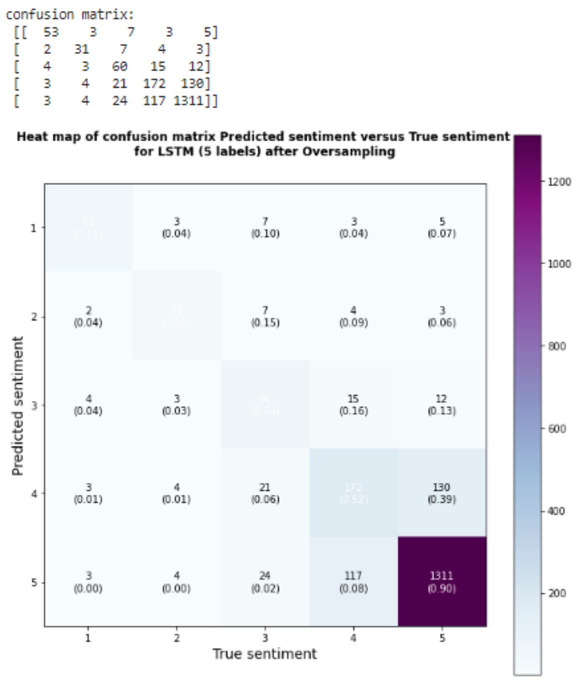
```
Confusion Matrix:
 [[  82   18   18]
 [  31   37   31]
 [  66  149 1569]]
```

Figure 5.32: Sentiment 1-3 for Coursera in Multinomial-Naive Bayes after Oversampling



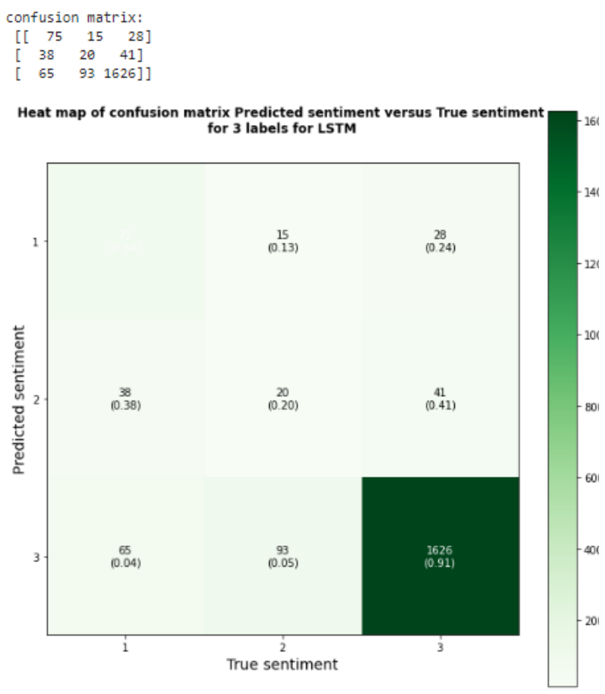
```
confusion matrix:
 [[  11   37    9    5    4]
 [   6   26   10    5    5]
 [  15   22   16   28   18]
 [   8   24   47  101  175]
 [   2   25   32  204 1166]]
```

Figure 5.33: Sentiment 1-5 for Coursera in LSTM after Class weighting



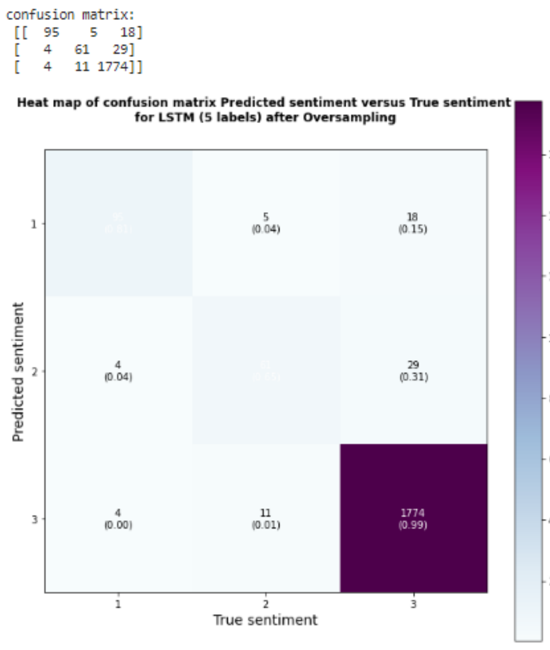
```
confusion matrix:
 [[  53    3    7    3    5]
 [   2   31    7    4    3]
 [   4    3   60   15   12]
 [   3    4   21  172  130]
 [   3    4   24  117 1311]]
```

Figure 5.34: Sentiment 1-5 for Coursera in LSTM after Oversampling



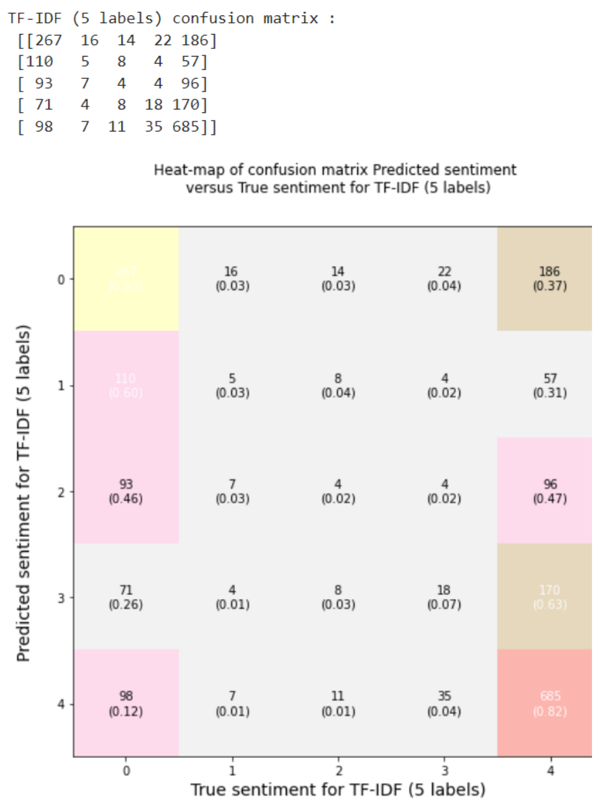
```
confusion matrix:
 [[  75   15   28]
 [  38   20   41]
 [  65   93 1626]]
```

Figure 5.35: Sentiment 1-3 for Coursera in LSTM after Class weighting



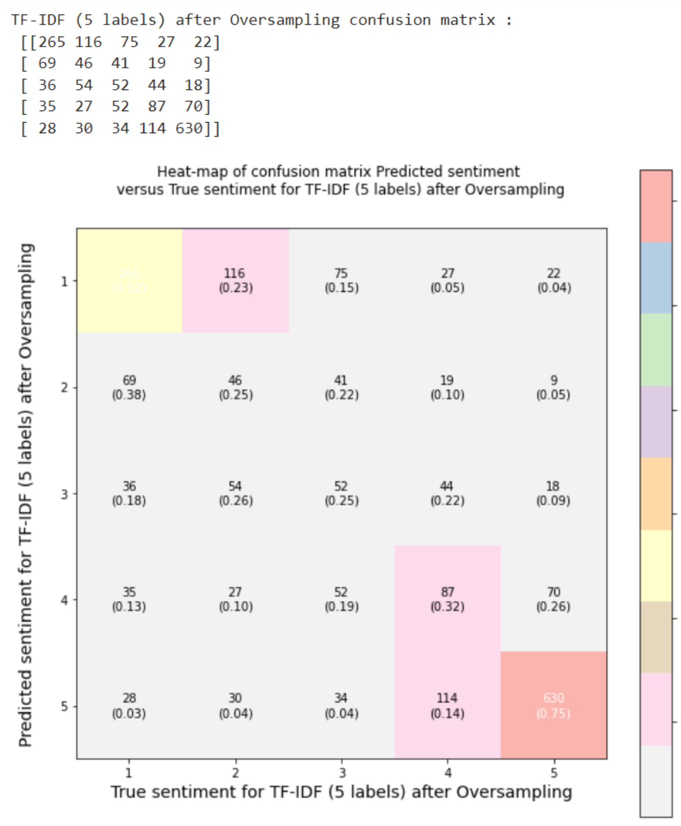
```
confusion matrix:
 [[  95    5   18]
 [   4   61   29]
 [   4   11 1774]]
```

Figure 5.36: Sentiment 1-3 for Coursera in LSTM after Oversampling



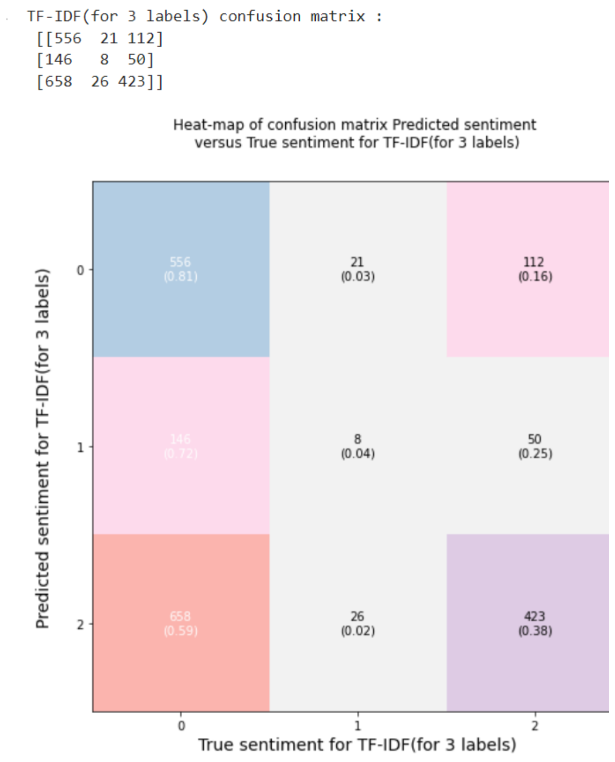
```
TF-IDF (5 labels) confusion matrix :
 [[267  16  14  22 186]
 [110   5   8   4  57]
 [ 93   7   4   4  96]
 [ 71   4   8  18 170]
 [ 98   7  11  35 685]]
```

Figure 5.37: Sentiment 1-5 for Spotify in TF-IDF after Class-weighting



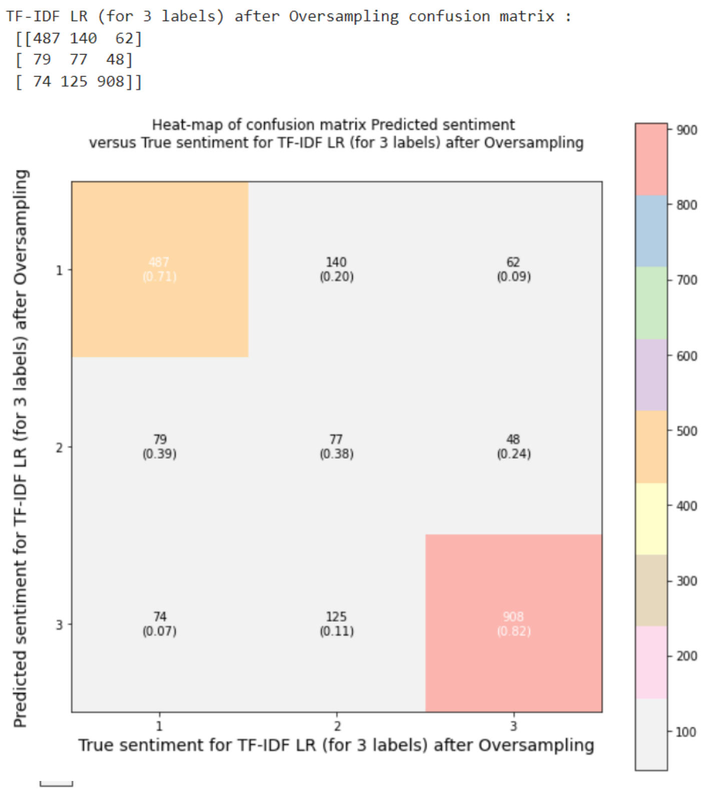
```
TF-IDF (5 labels) after Oversampling confusion matrix :
 [[265 116  75  27  22]
 [ 69  46  41  19   9]
 [ 36  54  52  44  18]
 [ 35  27  52  87  70]
 [ 28  30  34 114 630]]
```

Figure 5.38: Sentiment 1-5 for Spotify in TF-IDF after Oversampling



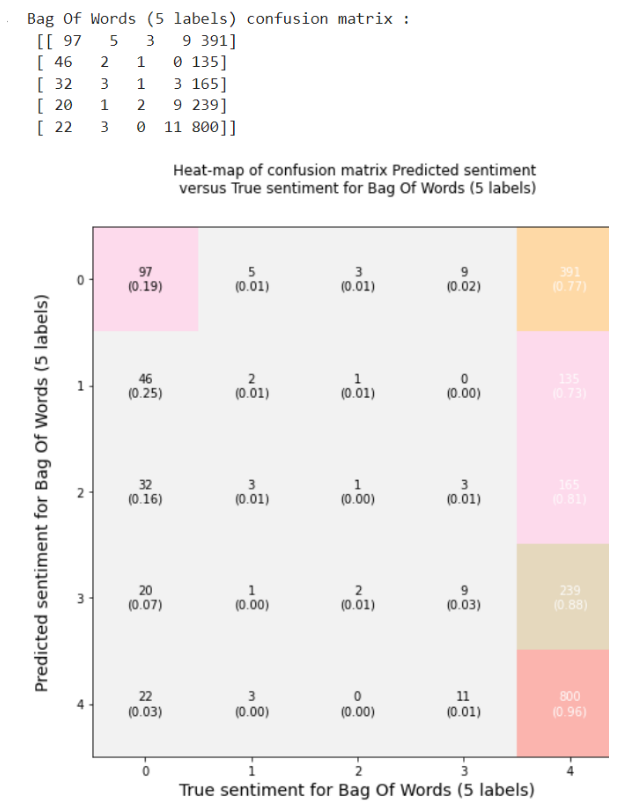
```
TF-IDF(for 3 labels) confusion matrix :
 [[556  21 112]
 [146   8  50]
 [658  26 423]]
```

Figure 5.39: Sentiment 1-3 for Spotify in TF-IDF after Class-weighting



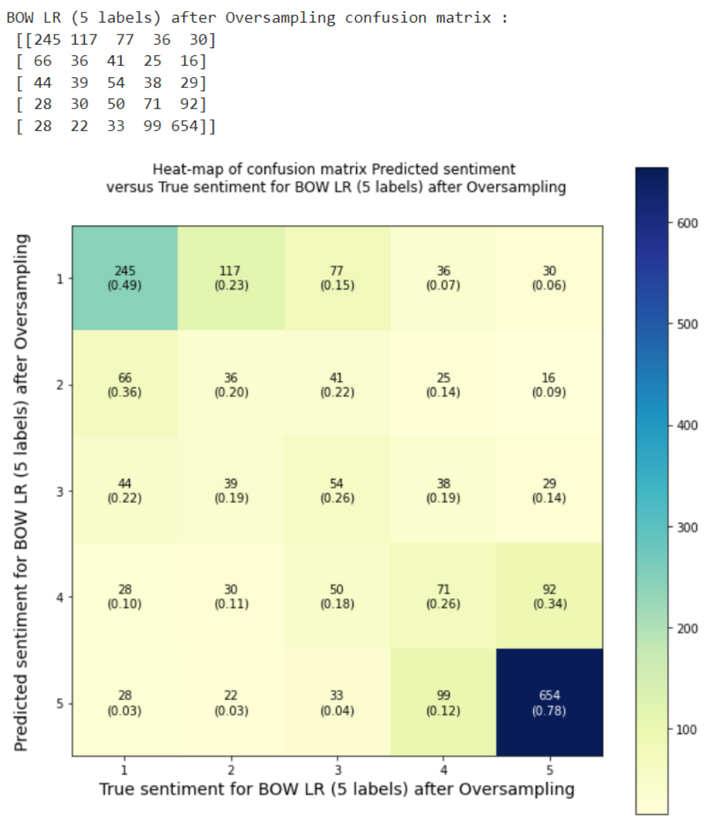
```
TF-IDF LR (for 3 labels) after Oversampling confusion matrix :
 [[487 140  62]
 [ 79  77  48]
 [ 74 125 908]]
```

Figure 5.40: Sentiment 1-3 for Spotify in TF-IDF after Oversampling



```
Bag Of Words (5 labels) confusion matrix :
 [[ 97   5   3   9 391]
 [ 46   2   1   0 135]
 [ 32   3   1   3 165]
 [ 20   1   2   9 239]
 [ 22   3   0  11 800]]
```

Figure 5.41: Sentiment 1-5 for Spotify in BOW after Class-weighting



```
BOW LR (5 labels) after Oversampling confusion matrix :
 [[245 117  77  36  30]
 [ 66  36  41  25  16]
 [ 44  39  54  38  29]
 [ 28  30  50  71  92]
 [ 28  22  33  99 654]]
```

Figure 5.42: Sentiment 1-5 for Spotify in BOW after Oversampling

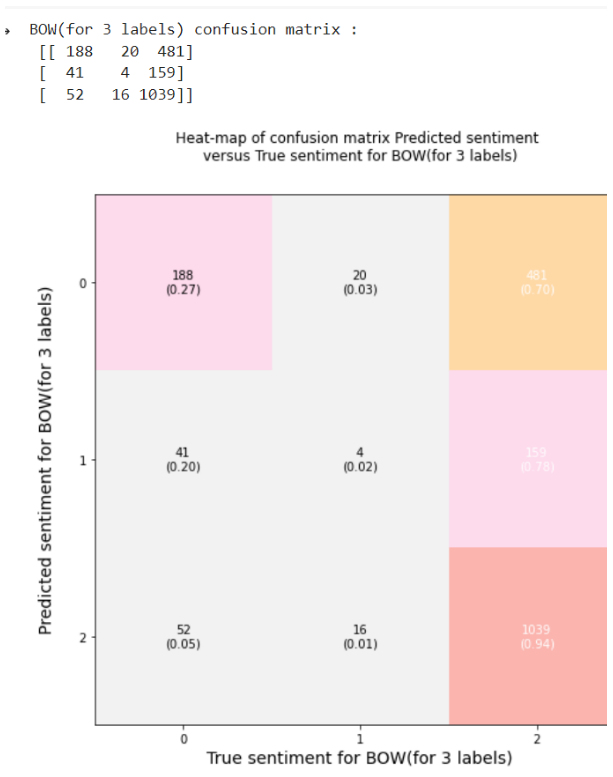

```
BOW(for 3 labels) confusion matrix :
 [[ 188   20  481]
 [  41    4  159]
 [  52   16 1039]]
```

Figure 5.43: Sentiment 1-3 for Spotify in BOW after Class-weighting

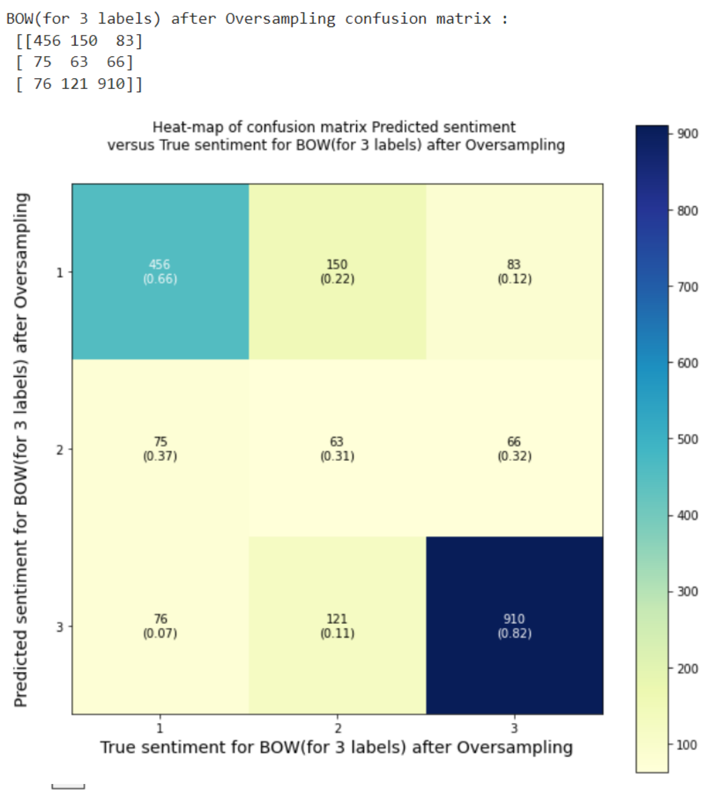
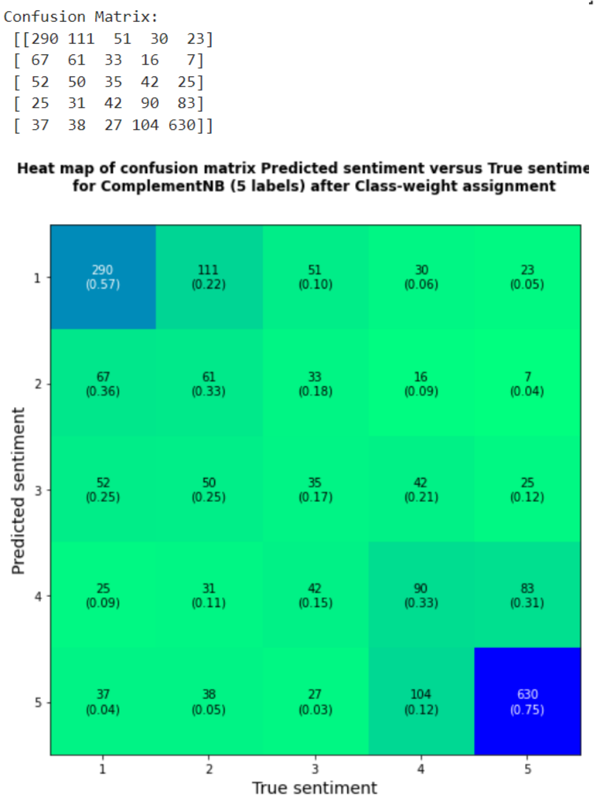
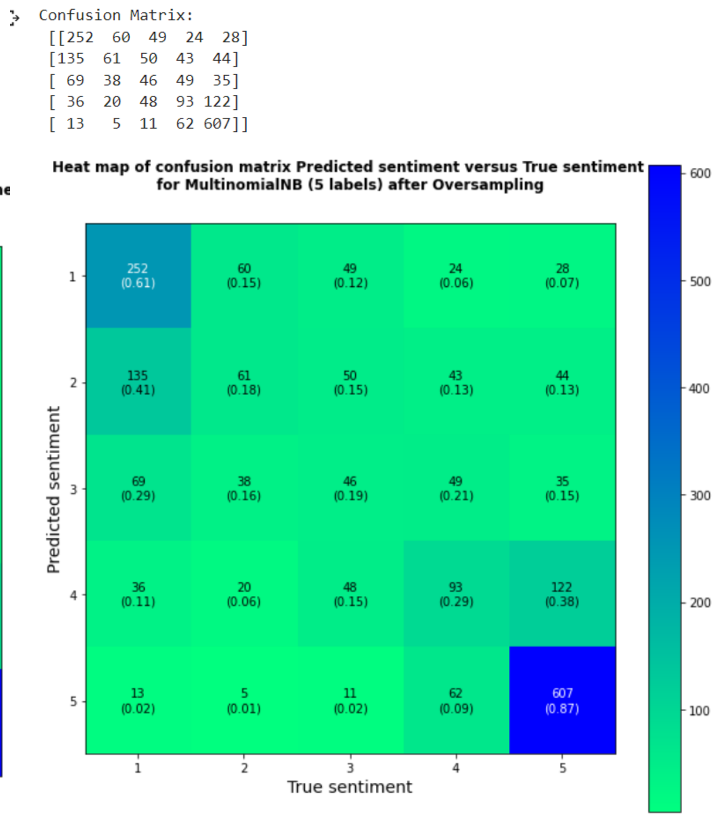

```
BOW(for 3 labels) after Oversampling confusion matrix :
 [[456 150  83]
 [ 75  63  66]
 [ 76 121 910]]
```

Figure 5.44: Sentiment 1-3 for Spotify in BOW after Oversampling



```
Confusion Matrix:
 [[290 111  51  30  23]
 [ 67  61  33  16   7]
 [ 52  50  35  42  25]
 [ 25  31  42  90  83]
 [ 37  38  27 104 630]]
```

Figure 5.45: Sentiment 1-5 for Spotify in Multinomial-Naive Bayes after Class-weighting



```
Confusion Matrix:
 [[252  60  49  24  28]
 [135  61  50  43  44]
 [ 69  38  46  49  35]
 [ 36  20  48  93 122]
 [ 13   5  11  62 607]]
```

Figure 5.46: Sentiment 1-5 for Spotify in Multinomial-Naive Bayes after Oversampling

```
Confusion Matrix:
 [[594  11  84]
 [134   6  64]
 [138   8 961]]
```

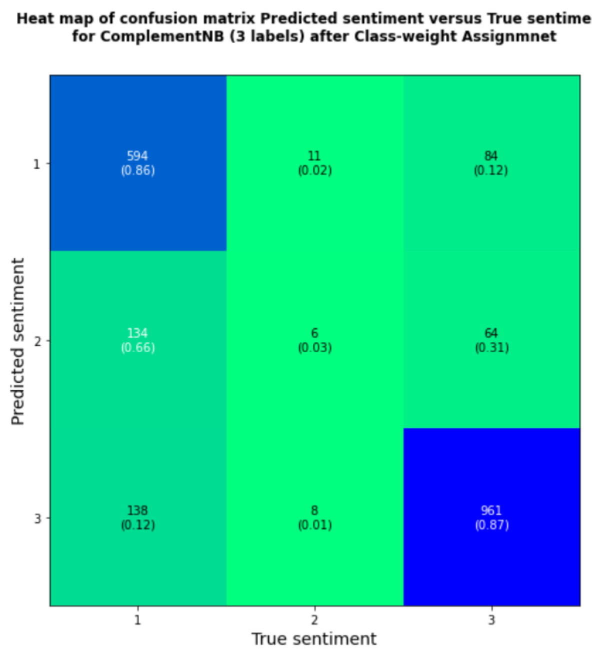


Figure 5.47: Sentiment 1-3 for Spotify in Multinomial-Naive Bayes after Class-weighting

```
Confusion Matrix:
 [[481  83  78]
 [153  85 151]
 [ 55  36 878]]
```

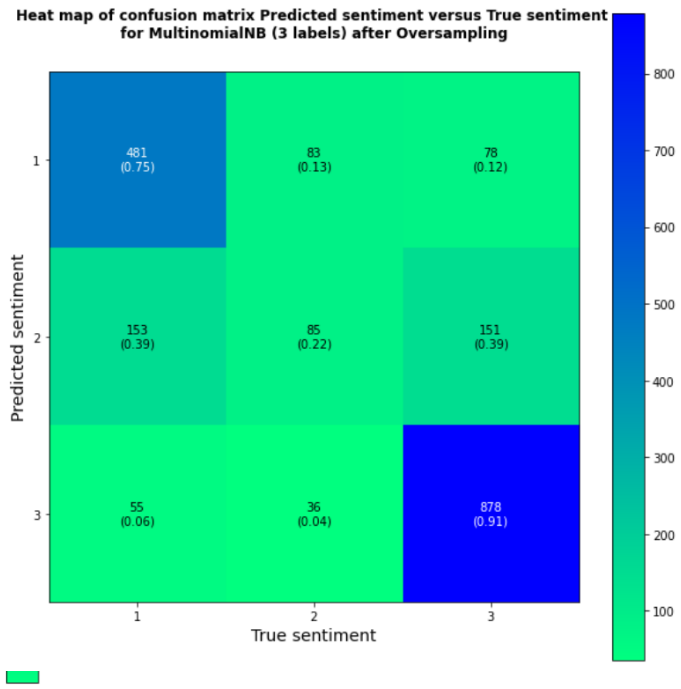


Figure 5.48: Sentiment 1-3 for Spotify in Multinomial-Naive Bayes after Oversampling

```
confusion matrix:
 [[357  80  39  14  15]
 [ 84  45  38  13   4]
 [ 61  40  64  27  12]
 [ 24  27  57 102  61]
 [ 55  20  41 120 600]]
```

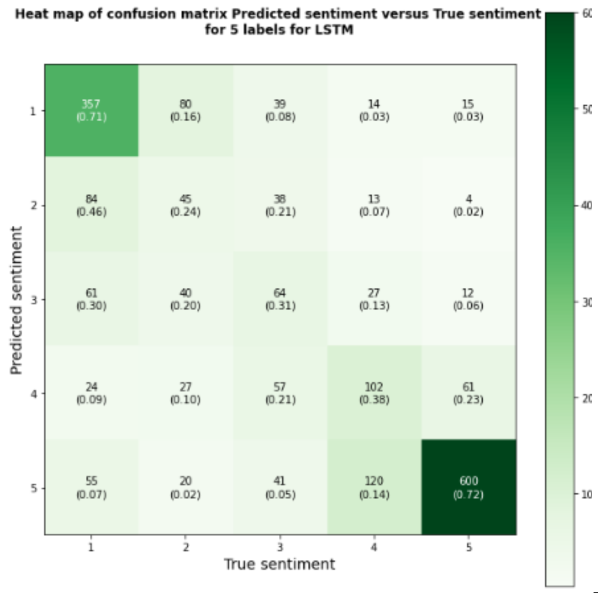


Figure 5.49: Sentiment 1-5 for Spotify in LSTM after Class weighting

```
confusion matrix:
 [[300 104  62  25  15]
 [ 40  66  42  11   4]
 [ 32  39  70  44   7]
 [ 21  30  44 129  58]
 [ 28  22  44 105 658]]
```

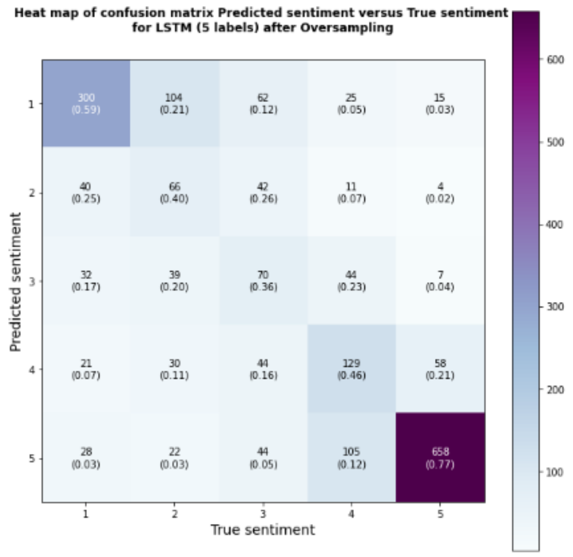
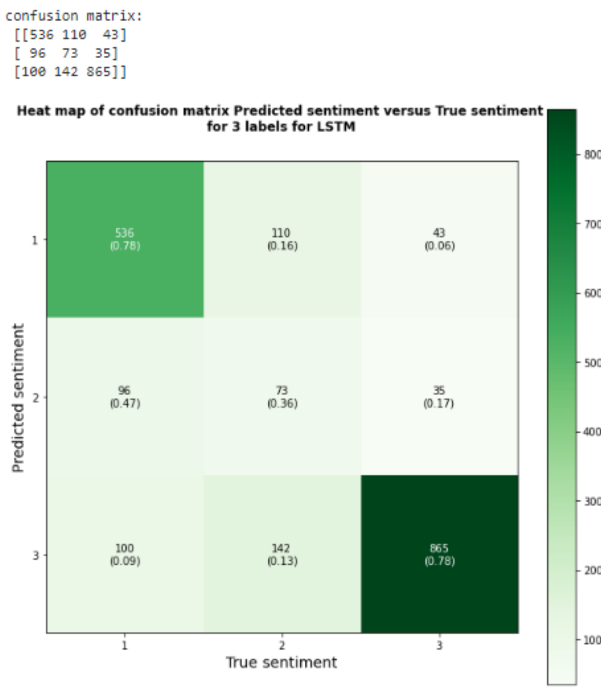
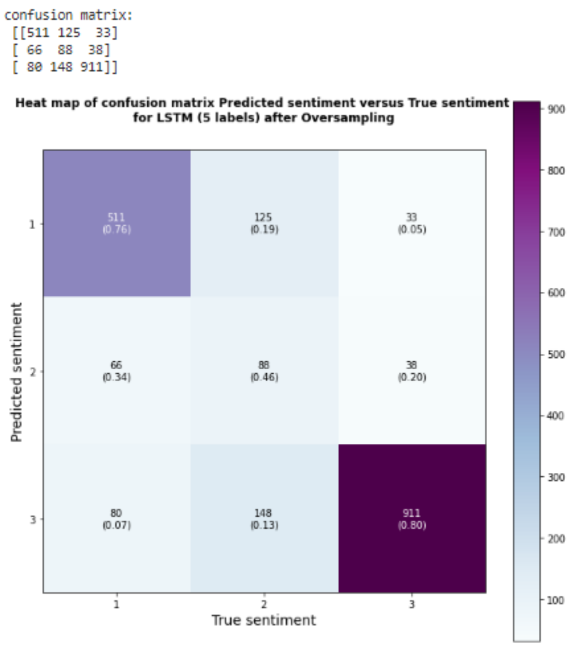


Figure 5.50: Sentiment 1-5 for Spotify in LSTM after Oversampling



```
confusion matrix:
 [[536 110  43]
 [ 96  73  35]
 [100 142 865]]
```

Figure 5.51: Sentiment 1-3 for Spotify in LSTM after Class weighting



```
confusion matrix:
 [[511 125  33]
 [ 66  88  38]
 [ 80 148 911]]
```

Figure 5.52: Sentiment 1-3 for Spotify in LSTM after Oversampling

Chapter 6

Conclusion

People in the modern world spend a lot of time on social media. Here we have many platforms to interact with people to convey our thoughts and feelings. But sometimes by reading a text it is quite difficult to understand the feelings as we cannot see the person's expression. Emotions in people are sentimental states which is connected to physiological reactions. Many real-world applications that use sentiment identification use a person's emotional state as an indication to how well the machine learning system is working. Although it may seem difficult, it is frequently necessary to infer a person's emotional state from an analysis of a text they have written because textual expressions are frequently the result of the interpretation of the meaning of concepts and the interaction of concepts stated in the text document. In our project our work is to recognize sentiments in Text by Using Machine Learning. Many different applications, such as speech recognition, email filtering, computer vision, and medicine, use machine learning algorithms when it is difficult or impossible to develop standard algorithms to do the necessary tasks. To identify our text-based sentiments, we use Bag of Words, TF-IDF, LSTM, Naive Bayes, and BERT. Our work was to find out the best model which will fit in our project. After using all of the models we found the BERT Model which is alright for our project. Though our field of work is still new and there are lots of new things to explore. So there are still some limitations that are yet to be resolved and in future we will work on it if we find something better then Bert. We do not want to stop our work here. In this case we are wishing to work on Sentence-Bert (SBERT). A variation of the pre-trained BERT network called Sentence-Bert (SBERT) creates semantically meaningful sentence embedding that may be assessed using cosine-similarity. It does this by using Siamese and triplet network architectures. Alternatively, SBERT is utilized as a sentence encoder, and similarity is tested using the cosine-similarity between both the gold labels and the sentence embedding. Lastly we want to make our research paper more efficient and resourceful for other people. So that in future people can use our paper for their own research. That's how our project will be more enriched and can be helpful for future researchers.

Bibliography

[1] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, Citeseer, vol. 242, 2003, pp. 29–48.

[2] C. O. Alm, D. Roth, and R. Sproat, "Emotions from text: Machine learning for text-based emotion prediction," in *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, 2005, pp. 579–586.

[3] C. Boulis and M. Ostendorf, "Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams," in *Proc. of the International Workshop in Feature Selection in Data Mining*, Citeseer, 2005, pp. 9–16.

[4] S. N. Shivhare and S. Khethawat, "Emotion detection from text," *arXiv preprint arXiv:1205.4944*, 2012.

[5] Z. C. Lipton, J. Berkowitz, and C. Elkan, *A critical review of recurrent neural networks for sequence learning*, Oct. 2015. [Online]. Available: https://arxiv.org/abs/1506.00019.

[6] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[7] A. Yadollahi, A. G. Shahraki, and O. R. Zaiane, "Current state of text sentiment analysis from opinion to emotion mining," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–33, 2017.

[8] J. Brownlee, *A gentle introduction to the bag-of-words model*, Aug. 2019. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model/#:~:text=A%5C%20bag%5C%2Dof%5C%2Dwords%5C%20is,the%5C%20presence%5C%20of%5C%20known%5C%20words..

[9] D. Mehta, M. F. H. Siddiqui, and A. Y. Javaid, "Recognition of emotion intensities using machine learning algorithms: A comparative study," *Sensors*, vol. 19, no. 8, p. 1897, 2019.

[10] N. Reimers and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, Aug. 2019. [Online]. Available: https://arxiv.org/abs/1908.10084.

[11] F. A. Acheampong, C. Wenyu, and H. Nunoo-Mensah, "Text-based emotion detection: Advances, challenges, and opportunities," *Engineering Reports*, vol. 2, no. 7, e12189, 2020.

[12] Baeldung, *Introduction to emotion detection in written text*, 2020. [Online]. Available: https://www.baeldung.com/cs/ml-emotion-detection.

[13] B. I. C. Education, *What is machine learning?* Jul. 2020. [Online]. Available: https://www.ibm.com/cloud/learn/machine-learning.

[14] A. F. A. Nasir, E. S. Nee, C. S. Choong, *et al.*, "Text-based emotion prediction system using machine learning approach," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 769, 2020, p. 012 022.

[15] A. Chiorrini, C. Diamantini, A. Mircoli, and D. Potena, "Emotion and sentiment analysis of tweets using bert.," in *EDBT/ICDT Workshops*, 2021.

[16] M. Krommyda, A. Rigos, K. Bouklas, and A. Amditis, "An experimental analysis of data annotation methodologies for emotion detection in short text posted on social media," in *Informatics*, Multidisciplinary Digital Publishing Institute, vol. 8, 2021, p. 19.

[17] M. M. Mutlu and A. Özgür, "A dataset and bert-based models for targeted sentiment analysis on turkish texts," *arXiv preprint arXiv:2205.04185*, 2022.

[18] *Reinforcement learning*, Aug. 2022. [Online]. Available: https://www.geeksforgeeks.org/what-is-reinforcement-learning/.

[19] *Digital around the world - datareportal – global digital insights.* [Online]. Available: https://datareportal.com/global-digital-overview.

[20] K. Dnyaneshwar, R. Ratnadeep, R. Ajay, and L. Troiano, "Evaluation of semeval 2016 restaurant and laptop reviews data for emotion and aspect classification,"

[21] *Sentiment analysis guide.* [Online]. Available: https://monkeylearn.com/sentiment-analysis/.

[22] A. Beliba, "Challenges of emotion recognition in images and video. apriorit," *https://datareportal.com/global-digital-overview*, 1 June 2021.