

MSc Dissertation Report

"Sentiment Analysis of online product review using Artificial Neural Network"

A dissertation submitted in partial fulfilment of the requirements of Sheffield Hallam University for the degree of Master of Science in **Big Data Analytics**

Student Name	Dhivya Nataraj
Student ID	29034198
Supervisor	Kostas Domdouzis
Date of Submission	13/9/2021

This dissertation does NOT contain confidential material and thus can be made available to staff and students via the library.

Abstract

Understanding the user need from the feedbacks are most important for the business to grow in the competitive market. Sentiment analysis helps the organisation to know how people react to the service and the product offered by them and what changes can be incorporate to improve their business in the competitive market.

Hence finding the sentiment form the feedback provide an optimal solution for the organisation to come with the better data driven decision. Even though there are several techniques that are being in use to find the sentiment behind the reviews. Most of them are failed to address the several issues and the result is not accurate. This project work focusses on the sentiment of the product in amazon.

The goal of the project is to create a classification model that classify the reviews associate with the amazon product using different machine learning and deep learning algorithms. Out of using the various algorithm ANN(Artificial Neural Network) with supervised word embedding and pre-trained word embedding (word2vec) as a vectorizer tends to give higher accuracy for the obtained dataset.

Acknowledgement

I like to thank the tutors at Sheffield Hallam University, in particular my supervisor Kostas Domdouzis who supported and guiding me in this research project. This project couldn't have been completed without the valuable feedback from Kostas Domdouzis. I also like to thank my family and my friends who helped me in the proofreading of the report and testing the model.

Contents

Chapter 1 –Introduction.....	1
1.1 Project Rational.....	1
1.2 Project Scope.....	1
1.3 Project Goal.....	2
1.4 Project objective.....	2
1.5 Challenges.....	2
1.6 Project Benefits.....	3
1.7 Achieving the Research Objective.....	3
Chapter 2 – Literature Review.....	5
2.1 Data Pre-processing.....	5
2.2 Algorithm.....	8
Chapter 3 – Research Design.....	11
3. 1 Research Ethics.....	11
Chapter 4 – Dataset.....	12
4.1 Amazon Customer Review – Product.....	12
4.2 Dataset Summary.....	14
Chapter 5 – Applications.....	15
Chapter 6 – Data Upload and Feature Selection.....	18
6.1 Data Upload.....	18
6.2 Defining Variables.....	18
6.3 Feature Selection.....	19
Chapter 7 – Model Development.....	22
7.1 Data Pre-processing.....	22
7.2 Text Analysis.....	26
7.3 Training and Test and Validation datasets.....	28
7.3.1 Handling Imbalanced dataset.....	28
7.4 Algorithm Computation.....	30
7.4.1 Random Forest Classifier.....	31
7.4.2 Decision Tree.....	32
7.4.3 Logistic Regression/SGD Classifier.....	33
7.4.4 Naïve Bayes.....	34
7.4.5 GridSearchCV.....	34

7.4.6 Deep Learning Model – Artificial Neural Network.....	35
7.4.7 Algorithm Computation Experiments.....	37
7.4.8 Evaluation Metrics.....	40
7.4.9 Testing Algorithm Performance.....	41
7.5 Final Analysis.....	47
Chapter 8 – Project Conclusion.....	48
References.....	49
Appendix A.....	51
1. Introduction and Justification.....	51
2. Research Question, Aim and Objectives.....	51
2.1 Research Question.....	51
2.2 Objectives.....	51
2.3 Deliverable.....	52
3. Literature Review.....	52
4. Research Design.....	54
4.1 Research Philosophy, Approach and Methodology.....	54
4.2 Artificial Neural Network.....	54
4.3 Tools and Techniques.....	54
4.4 Methodology.....	54
4.5 Proposed Methodology.....	55
4.6 User Interface.....	55
5. Ethics.....	56
6. Timeline.....	57
7. References.....	57
Appendix B.....	59
1.Completed Research Ethics Checklist.....	59
2. Publication procedure form.....	65
Appendix E.....	66
1.Python Code.....	66
1.1 TF – IDF for Machine Learning Algorithm.....	67
1.2 Word Embedding for Machine Learning.....	73
1.3 Word Embedding For ANN.....	83
1.4 ANN Word2Vec.....	90

Chapter 1 - Introduction

1.1 Project Rationale

In March 2020, about 40% of UK buyers claimed they were spending more online than they had been before the coronavirus (COVID-19) outbreak. However, by February 2021, this ratio had increased to almost 75%. According to these statistics, many consumers in the United Kingdom have switched to internet shopping.

So many businesses are switching to an online platform these days because people's habits are changing, and they are increasingly acquiring products online. As a result, there is a wide range of products available on the internet, as well as consumer reviews on the products they have purchased.

As amount of data is growing rapidly which need the special techniques and high computational power to process the data. Sentiment analysis of online product review Classifying reviews as positive, negative, or neutral is beneficial to both the consumer and the stakeholder.

1.2 Project Scope

In this research amazon product review: shoe category is considered and split the dataset based on brand of shoes on amazon such as adidas review, Skechers review, and crocs review. For which a deep learning ANN(Artificial Neural Network) and 4 machine learning algorithms like Random Forest, Decision tree, Logistic regression/SGD(Stochastic gradient descent), and Naïve bayes are examined to classify reviews as positive, Negative, and Neutral. And the obtained dataset is imbalanced to make it balanced the smote Tomek techniques is used. In addition, machine learning model is hyperparameter tuned using GridSearchCV. All these algorithms are experimented using a different vectorization technique such as supervised word embedding, TF-IDF, and pre-trained Word2vec.

1.3 Project Goal

The project goal is to predict whether the reviews are positive or negative or neutral.

With the dataset of n reviews, reviews are listed as

$$R = r_1, r_2, r_3, \dots, r_n, \text{ where } R \text{ is collection of reviews.}$$

Each review contains series of words such as for a review $r_1 = w_1, w_2, w_3, \dots, w_n$.

The project will focus to classify the reviews into positive, negative, or neutral. So, the project aim is to create a predictive algorithm to create a classification structure which has the capability to predict if a review is positive, negative, or neutral.

1.4 Project Objectives

To achieve the project goals, project objectives are listed below.

1. Conduct a thorough literature review to gain a better understanding of the existing studies.
2. Select an acceptable dataset for model development.
3. Use data pre-processing to remove ineffective data from the dataset and balance the dataset if it is imbalanced.
4. Identify the predictor and target variable for model creation.
5. Create a model by using various machine learning and deep learning algorithm to classify the reviews.
6. Collect information on accuracy and other parameters to determine the best algorithm.
7. Run a test on the model to see how it performs.
8. Determine the scope of future work.

1.5 Challenges

There are numerous obstacles in the subject of sentiment analysis. The author of this research work (Shahnawaz, Parmanand Astya, 2017) listed some of the open problems and important concerns in sentiment analysis as follows:

- Because of these concerns, comparative and objective sentences, classification accuracy, and sarcasm are the main issues in all systems for sentiment analysis. Most sentiment analysis models incorrectly classify most pieces of text from opinionated

data into wrong classes, with a large amount of the opinionated data being incorrectly labelled as neutral.

- Because most of the sentiment analysis and opinion mining research focuses on English-language text, many resources, libraries, and tools (such as sentiment corpora and lexicons) have been generated exclusively in English. It's often difficult to apply these tools and research to other languages.
- Classification of feelings from a limited amount of labelled opinionated data remains an ambitious and difficult task, owing to the difficulty and cost of obtaining labelled opinionated data. Unlabelled opinionated data, on the other hand, is relatively straightforward to get and affordable.

1.6 Project Benefits

For the sentiment categorization problem in online product reviews, several methods have been developed. Because the project was established by experimenting with various machine learning and a deep learning algorithm, its benefits are limited.

1. Understand the importance of deep learning model in the sentiment classification of online product reviews.
2. Analysis of the given dataset using multiple machine learning and a deep learning algorithm.
3. Determined the key factors that influence the classification system's model performance.
4. Determine the scope of future research.

1.7 Achieving the Research Objectives

The initial proposal for the project is attached in Appendix A (research project plan) and its ethical approval is attached in Appendix B.

The project flow (figure 1.1) began with literature review of data collection, data pre-processing and identified different type of model to build on selected dataset. This follows the research design where the main research was identified before the analysis. Then model development is carried out to prepare the model by giving training with different algorithm. Next model testing was conducted to carry out the performance of the model. Finally, conclusion figure out the model outcome and scope for the further work.

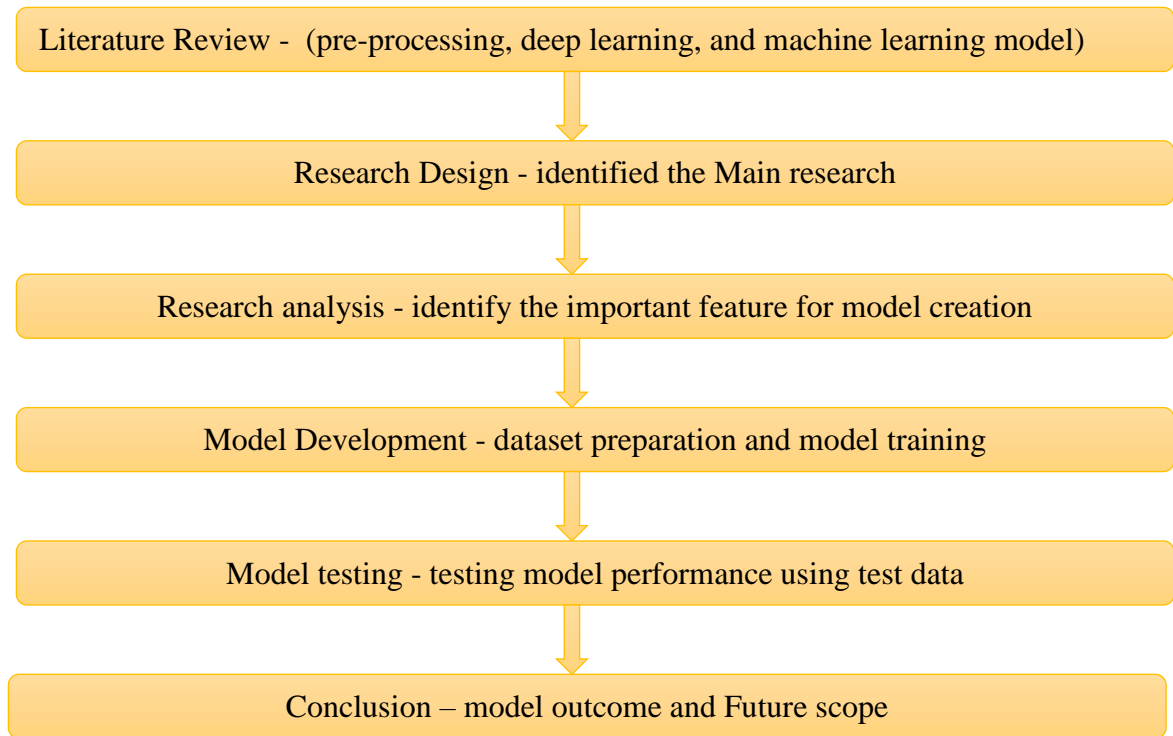


Figure :1.1 – Project Flow

Chapter : 2 – Literature Review

Table 2.1 shows the key literature review area for the proposed project. Literature review area contain two sections such as data pre-processing, and approach to various algorithm. In data pre-processing the unwanted word like stop words, non-English words, numerical values are removed. Furthermore, tokenized, lemmatized each review in the dataset and word embedding used to build the deep neural network model and count vectorizer, term frequency and inverse document(TDIDF) is applied to create machine learning model. And dataset is applied to various machine learning and deep learning algorithm. Finally, the performance of the model is evaluated to ensure the validity.

Literature Area	Literature Review Justification
Data pre-processing	<ul style="list-style-type: none">➤ To convert the textual data(amazon product reviews) into machine understandable form and to remove the noise present in the data. pre-processing is performed on the dataset. Better performance is achieved only when the data is clean and noiseless.➤ Identified the most appropriate feature reduction techniques for machine learning model.➤ Identified sequence representation techniques such as supervised word embedding for deep learning model.
Approach to various algorithm	<ul style="list-style-type: none">➤ Identify the most suitable algorithm for the sentiment analysis classification problem.➤ Deep learning and Machine learning algorithm need different approach to solve the sentiment classification.➤ Analysis performance of one versus another algorithm.

Table 2.1 – Literature Review Justification

2.1 Data Pre-processing

To obtain the correct sentiment for successful decision making, a pre-processing procedure is used. Remove the noise and clean the text to make it helpful for classification of sentiment associated with the text is obtained using pre-processing. There are several operations are applied to clean text like lower the text, white space removal, punctuation removal, removal of encoded text, stop word removal, tokenization, stemming, etc. (Saurav Pradha, 2019) mentions a technical comparison between stemming and lemmatization to determine which

technique is better for text classification. The application of each technique, as well as its descriptions, were also discussed.

Some of the pre-processing approaches used to clean and remove noise and convert text into vector representation are listed below.

- a) Lower case:** The main reason for doing lower case is to ensure the upper and lower case of same word are not processed differently. It aids in the reduction of number of words used in the dictionary.
- b) Tokenization:** It is the process of breaking down a string into separate pieces such as words, keywords, phrases, symbols, and other tokens. Individual words, phrases, or even entire sentences can be used as tokens. (2018, Tanjim) mentions that tokens are used as input for various processes such as parsing and text mining. Before tokenizing the text, it must be cleaned; otherwise, it will consider punctuation, white space, encoded text, and non-English words. Those words aren't helpful in solving the text classification challenge.
- c) Stop words:** Stop words approaches are based on the notion that some words in a language contribute no meaningful information to a text mining process. In the paper author(2018, Gerardo) mentioned that the terms include articles, conjunctions, prepositions, and determiners, among others are removed from the text using this procedure. These words are so commonly used that they carry very little useful information.
- d) Lemmatization:** Lemmatization is a term that relates to doing things correctly using a vocabulary and morphological analysis of words, with the goal of removing inflectional endings solely and returning the base or dictionary form of a word, known as the lemma. In paper (2017, Mayuri Mhatre) author used word's dictionary form from the WordNet dictionary. For example, the lemma of the word "looks" is "look." Comparative study was conducted between stemming and lemmatization in paper (Vimala Balakrishnan, 2014) and found that lemmatization yielded superior results compared to stemming. Lemmatization returns the base or dictionary form of a word, whereas stemming returns all words with the same stem to a common form.
- e) Vectorization:** To transform text to numerical representation, the text vectorization technique is used. Text vectorization can be done in a variety of ways.

- a) **Bag of Words (BOW)** : Bag of words is a numerical representation of text data when modelling a machine learning. This represents each sentence in document as a bag of words vectors. If the word is appearing in the sentence, then the value is '1' if it is not appearing then the value is '0'. This also determining the vocabulary size by counting the number of times a word appears within the document and organising it into a list of words. The semantic information is more important for finding the polarity. To enhance the BOW in (Bijoyan ,2018) paper used bag of words model with term frequency-inverse document frequency scores.
- b) **Term frequency** : Inverted Document Frequency (TF-IDF) – The documents are represented as vectors in this case as well, but instead of a vector of 0s and 1s, the document now has scores for each of the words. For individual terms, these scores are obtained by multiplying TF and IDF. As a result, the score of any word in any document may be represented using the equation below. There are two metrics to calculate: term frequency and inverse document frequency. The following are the formulas for calculating both.

$$\text{TF}(\text{term frequency}) = \frac{\text{frequency of word in sentence}}{\text{no of words in sentence}}$$

$$\text{Inverse document frequency} = \log\left(\frac{\text{no of sentences}}{\text{no of sentence containing words}}\right)$$

Bijoyan(2018) employed TF-IDF to tackle a text sentiment classification problem and improved the model's performance significantly.

- f) **Word embedding**: The word embedding can be computed using two methods: supervised learning and self-supervised learning. Word2vec and Glove are used in self-supervised learning.

- a) **Supervised learning**: The supervised way of word embedding method creates a dense representation of words and their meanings. Each sentence is encoded using one hot encoding techniques and padded to same length of sequence. To train the model, this type of representation can be employed on both machine learning and deep learning models. The embedding layer is utilised mostly on the front

end of a neural network and is fitted with the backpropagation method in a supervised manner.

- b) **Word2Vec:** Google's word2vec model was trained using a portion of the Google News dataset, which contains roughly 100 billion words. This approach includes a 300-dimensional vector representation. The word2vec tool accepts a text corpus as input and outputs word vectors. Most importantly for constructing vector representations of words, this tool provides an efficient implementation of the continuous bag-of-words and skip-gram designs. Author (Seyed Mahdi, 2017) applied word2vec technique pre-trained by Google News achieved better results for deep learning than Glove.

Data pre-processing summary:

Following a thorough examination of the various procedures and operations involved in data pre-processing, lowering the sentence, tokenization, and stop word removal are essential for removing noise and reducing dimensionality, and these must be done.

When it comes to extracting root words, the lemmatization algorithm appropriately associates dictionary words and performs better than the stemming method. As a result, lemmatization techniques will be used in the project.

The simplest form of vector representation of text is a bag of words, which represents each word as either '0' or '1' values. TFIDF (term frequency and inverse document frequency) is used to improve vectorization and generate the word score. However, these methods do not maintain semantic information, which is critical for text classification.

In supervised learning of word embedding sequence ordering of the word are retained and represent the word in vector. Whereas in the self-supervised word embedding semantically meaningful dense value vector is generating using pertained model like word2vec. The project will use BOW+TFIDF, word embedding and pretrained word2vec vector representation methods and compare the result.

2.2 Algorithm

The type of method used is determined by the machine learning challenge and the amount of processing required to reach the desired result. The goal of this project is to use a star rating as a label to categorise the sentiment associated with customer reviews on ecommerce

products. As a result, this is a classification issue. Furthermore, because the system classifies sentiment based on label data, supervised classification algorithm is used.

There are a variety of supervised classification methods for detecting sentiment in online product reviews. In the work (Dr. Shailendra Narayan Singh, & Twinkle Sarraf, 2020) used Random Forest classifier because it focuses on the ensemble of decision trees, which is made up of numerous tree combinations. Also mentioned that Random Forest, on the other hand, is far more resistant to noise as well as the provided randomness.

(A. Kafi, M., H. Arif, 2019) In this work, Amazon mobile phone reviews were collected, and sentiment analysis was performed using supervised machine learning algorithms such as Naive Bayes, Support Vector Machine, Logistic Regression, and the Stochastic Descent. The mobile phone features are taken and used in the model as a feature set. Finally, the model assigns a rating to each feature, and the average of the ratings is used to determine the polarity of the cell phone. The model's accuracy is compared, and it is discovered that SVC and random forest perform better in terms of accuracy.

In the work (Faisal ,2018) Amazon Product reviews are categorised into three categories: cell phones and accessories, musical instruments, and electronics. For determining polarity, machine learning models such as Naive Bayes, Stochastic gradient descent (SGD), linear regression, and ensemble algorithms such as decision tree and random forest are employed. According to the findings of the trial, SVC has the highest accuracy of 94.02 percent. For Stochastic gradient descent (SGD), Random Forest and Decision tree classifier, however, the model provides more than 90% accuracy.

(P. Lakshmi Prasanna & D. Rajeswara Rao,2018) The author examined the performance of traditional approaches versus Artificial Neural Networks (ANN) in text classification. Furthermore, it was demonstrated that text classification using ANN has a significant advantage over other algorithms and can be implemented quickly. Traditional approaches include Naive Bayes, KNN, SVM, and others.

Previous paper and researchers have proposed many effective solutions to improve the performance of sentiment classification. At the same time several techniques need to be examine for the selected variable on machine learning and deep learning algorithm.

Algorithm Summary – The project will employ four machine learning algorithms and one deep learning algorithm and compare the performance of these algorithms using the metric accuracy for sentiment classification of online product reviews.

1. Random Forest (RF)
2. Decision Tree (DT)
3. Logistic Regression (LR)/SGD(Stochastic gradient descent)
4. Naïve Bayes (NB)
5. Artificial Neural Network(ANN)

Chapter 3 – Research Design

The research methods and approach will be examined in this portion of the proposal. This project is a classification model based on the literature review system design. The research used a deductive strategy to acquire data and dataset has been utilised in several previous research reviews. A pragmatic approach was applied to model development to investigate a variety of options. Models are created by using various pre-processing techniques and then feeding the processed data into multiple models. Combination of qualitative and quantitative data was employed. (Ayat Zaki Ahmed & Manuel Rodríguez-Díaz, 2020) used information from quantitative data for carry out the sentiment analysis of online customers reviews of airlines.

- Qualitative data – The datasets are text and will not produce a score or number unless they are converted to numerical data.
- Quantitative data - The model accuracy, computational time, and performance are all quantifiable and result in a score. Various machine learning and a deep learning model are constructed, and their scores are compared to determine the best model.
- Below diagram demonstrates the flow of research methodologies

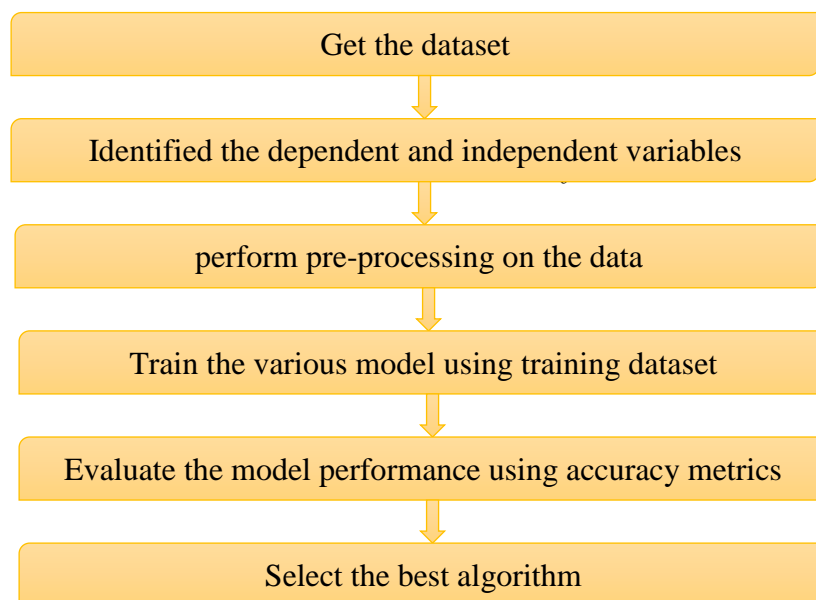


Figure 3.1 Research methodology

3.1 Research Ethics

Please refer to Ethics checklist in Appendix E.

Chapter 4 – Dataset

To obtain a quantifiable value such as a classification report, confusion matrix, or other performance measures, both machine learning and deep learning models require a superior dataset. First and foremost, the dataset must be labelled before the model must be built.

Although most of the Amazon customer review dataset is unlabelled, the project requires a tagged dataset to create a model. The project focuses on reviews of Amazon products, notably shoes. The project focuses on the below datasets.

4.1 Amazon Customer Reviews – Product

This dataset was provided by Amazon and contain customer's feedback of a purchased product on their website. This feedback indicates how people feel about a product and these data are collected in a specific geographical area, such as the United States East. There are 10,48,545 datapoints available for the product category shoes, but they are not organised by brand, for example Skechers, Adidas and Crocs.

The dataset includes a field called product title, which lists each product and project filters only those that are related to Skechers, adidas, or Croc's brand shoes. As a result, there are 41,583 datapoints in the dataset, that contains data linked to the three brand categories specified. The dataset is then separated by brand, such as adidas, Skechers, and Crocs, to discover the polarity for each branded shoes and develop the model.

Furthermore, the star rating column is utilised to label the dataset, which categorises the reviews as positive, negative, or neutral. If the star rating is 3, the review is labelled as neutral; if the star rating is less than 3, it is labelled as negative; and if the star rating is greater than 3, it is labelled as positive. The table below depicts an example of data from the collected dataset.

Column Name	Values
Marketplace	US
Customer id	22272389
Review id	R3B1NURKMCVAL1
Product id	B00LX65PQO
Product parent	848850234
Product title	Skechers Performance Women's Go Walk Extend Slip-On Walking Shoe
Product category	Shoes
Star rating	3

Helpful votes	2
Total votes	2
vine	N
Verified purchase	Y
Review headline	Three Stars
Review body	I like the go walk3 more than go walk. The textile is too hard.
Review date	2015-08-31

Table 4.1 Amazon Customer reviews Example

The dataset has 41,583 datapoints and is divided into 3 sets. Table 4.2 shows the distribution of all three subcategories of branded shoes in 3 sub-datasets.

Set	Adidas	Skechers	Crocs	Total
Train	6,327	9,853	10,431	26,611
Validation	1,582	2,464	2,608	6,654
Test	1,978	3,080	3,260	8,318
Total	9,887	15,397	16,299	41,583

Table 4.2 Distribution of subcategories brand shoe in dataset

The three types of reviews are positive, negative, and neutral, therefore there are more than two ways to classify the reviews in this dataset. As a result, the result will be multiclass. The distribution of three types of reviews is shown in Table 4.3.

Set	Neutral	Negative	Positive	Total
Train	2,081	2,100	22,430	26,611
Test	665	652	7,001	8,318
Validate	503	510	5,641	6,654
Total	3,249	3,262	35,072	41,583

Table 4.3 Distribution of three types of reviews in dataset

To investigate the distribution of reviews based on the type of review, the following subcategories of brand shoe are listed: adidas, Skechers, and crocs.

The dataset for machine learning models is validated using cross validation using the scikit package, whereas the dataset for deep learning models is split as previously stated.

Adidas Set	Neutral	Negative	Positive	Total
Train	459	474	5,394	6,327
Test	142	145	1,691	1,978
Validate	117	115	1,350	1,582
Total	718	734	8,435	9887

Table 4.4 Distribution of three types of reviews in adidas dataset

Skechers Set	Neutral	Negative	Positive	Total
Train	723	704	8,426	9,853
Test	235	207	2,638	3,080
Validate	170	191	2,103	2,464
Total	1,128	1,102	13,167	15,397

Table 4.5 Distribution of three types of reviews in Skechers dataset

Crocs Set	Neutral	Negative	Positive	Total
Train	899	922	8,610	10,431
Test	288	300	2,672	3,260
Validate	216	204	2,188	2,608
Total	1,403	1,426	13,470	16,299

Table 4.6 Distribution of three types of reviews in crocs dataset

The datapoints for each class, such as positive, negative, and neutral, in all three datasets: adidas, Skechers, and Crocs, are not balanced, as can be seen in table above. Positive reviews have more datapoints than negative and neutral reviews. It must be balanced before training the data to develop the model.

4.2 Dataset Summary

Many academic scholars in the fields of Natural Language Processing (NLP) use this dataset. This dataset is used in this study to determine the polarity associated with the review body. Also, this dataset is used to test the performance of both the machine learning and deep learning models to determine which model provides the best accuracy score for the sentiment classification task.

Chapter 5 – Applications

Various programming languages are available for the building of machine learning and deep learning models. The model used in the project was developed using Python as a programming language. Because of its numerous advantages, including ease of use, increased productivity as compared to other programming languages, free and open source, a wide range of supporting libraries, and portability. The Jupyter Notebook is used to access the Python framework for the project's implementation. The following libraries were utilised in the project's development (see table 5.1).

No	Library used	Description
1.	Pandas	<p>It is one of the cleaning and analysis methods used in machine learning models. NumPy is an open-source package built on top of it. For series, it employs a one-dimensional data structure, while for data frames, it uses a two-dimensional data structure. The following are some of the benefits of utilising pandas.</p> <ul style="list-style-type: none">• Missing data can be easily handled.• It gives a mechanism to slice, combine, concatenate, and restructure data, as well as times series to work with. <p>It is used to manipulate and analyse data as well as quickly conduct operations on this structure.</p>
2.	Natural Language Toolkit(NLTK)	<p>It is a method for a machine to understand text. It includes strong statistical language processing modules and packages. This package enables the machine to comprehend human speech. It offers many pre-trained models and corpora that make text analysis simple. It includes features like as tokenization, POS (Part of Speech), NER (Named Entity Recognition), classification, sentiment analysis, and access to many corpora.</p>
3.	Scikit - Learn	<p>For machine learning algorithms, it is the most used python machine learning library. Scikit-learn is a Python library that provides a uniform interface for supervised and unsupervised learning techniques. Data mining and data analysis are also</p>

		possible using the library. Classification, regression, clustering, dimensionality reduction, model selection, and pre-processing are the core machine learning functions that the Scikit-learn library can handle.
4.	Tensor Flow	<p>TensorFlow is a tool for building large-scale neural networks with a lot of layers. Dataflow graphs—structures that explain how data passes through a graph, or a sequence of processing nodes—can be created with TensorFlow. Each node in the graph symbolises a mathematical process, and each node-to-node connection or edge is a tensor, or multidimensional data array.</p> <p>TensorFlow has a modular architecture that allows it to run on a wide range of computing systems, including CPUs, GPUs, and TPUs. Tensor processing unit (TPU) is a machine learning and artificial intelligence hardware chip based on TensorFlow.</p>
5.	Keras	<p>It is a foundation library that can be used to develop Deep Learning models directly or via wrapper libraries built on top of TensorFlow to make the process easier. Layers, objectives, activation functions, and optimizers are some of the neural-network building pieces used by Keras.</p> <p>Keras also has several image and text image processing tools that come in handy when creating Deep Neural Network. It supports convolutional and recurrent neural networks in addition to the basic neural network.</p>
6.	Matplotlib	Matplotlib is a data visualisation package that may be used to create quality picture plots and figures in several formats for 2D plotting. Histograms, plots, error charts, scatter plots, and bar charts can all be generated using the library. It's a set of methods that allow matplotlib to behave like MATLAB.
7.	NumPy	NumPy is a well-known array-processing library for Python. NumPy is capable of processing enormous multi-dimensional arrays and matrices. For linear algebra, Fourier transforms, and random numbers, NumPy is quite useful. Other libraries, such

		as TensorFlow, employ NumPy to manipulate tensors on the backend.
8.	Gensim	Gensim is a Natural Language Processing tool that is being used to process texts, create word vectors like word2vec, word embedding model. It provides several advantages, including the ability to transfer big files without needing to load them into memory. Because Gensim uses unsupervised models, it does not require expensive annotations or manual labelling of documents.
9.	Word Cloud	Word cloud is a text data visualization technique in which each word is visualised in relation to its relevance in the context or frequency. By deleting the stop words, it allows a faster presentation of words in the visual manner.

Figure 5.1 python library used in the project

Chapter 6 – Data upload and Feature Selection

6.1 Data Upload

The first step is to upload the data to the Jupyter notebook once the dataset has been collected and the application has been determined. The dataset is saved locally and then uploaded to the jupyter notebook using the panda's python library. Appendix E contains the code for uploading the dataset as a data frame using pandas. The flow of data uploading is depicted in the diagram 6.1 below.



Figure 6.1 data upload flow

6.2 Defining Variables

The target and predictor variables must be defined after uploading the dataset into the jupyter notebook for model development.

- **Target Variable** - The target variable is a variable that is used to train the model via the supervised machine learning technique. Depending on the method, these target variables may be manually labelled or automated. The project uses the star rating feature to manually label the target variable. The label is separated into positive, negative, and neutral categories based on the number of stars. Simultaneously, the sentiments were translated to numerical values by determining the type of sentiment:

$$f(x) = \begin{cases} 0 & \text{if } x \text{ is neutral} \\ 1 & \text{if } x \text{ is negative} \\ 2 & \text{if } x \text{ is positive} \end{cases}$$

where x denotes the emotion (positive, negative, or neutral).

- **Predictor variable** – The variables that predict the target are known as predictor variables. In addition, except for the target variable, all variables in the dataset are predictor variables. The term "predictor variable" can also refer

to an independent variable or a set of features. The predictor variable selection varies according to the business challenge and requirements.

6.3 Feature Selection

The dataset contains a variety of attributes, but only a few of them influence the target variable. The selection of features is more significant to have a higher performance in a machine learning and deep learning model. The author (F. P. Shah and V. Patel, 2016) of this research employed a variety of strategies to reduce the greater dimensionality of the features in the text classification problem and increase performance by deleting irrelevant characteristics and selecting only the necessary ones. (Mita K. Dalal and Mukesh A. Zaveri ,2011) emphasises the machine learning techniques can be used to successfully automate text classification; however, pre-processing and feature selection like TF - IDF processes are critical in determining the size and quality of training data given to the classifier, which influences the classifier's performance.

The feature selection method is used in this project to ignore irrelevant features from dataset and effectively estimate the target or dependent variable. The table 6.1 below summarises the process of selecting features for the Amazon customer review dataset.

No	Feature	Selected	Description	Reason
1.	Marketplace	No	Two letter country code of the marketplace where the review was written.	It was not chosen since it has no useful information about the target variable.
2.	Customer_id	No	Random identifier that can be used to aggregate reviews written by a single author.	Because the classifications task is more focused in the sentiment regarding the product brand, so it was not chosen.
3.	Review_id	No	The unique ID of the review.	It doesn't contain any useful information for the prediction problem.

4.	Product_id	No	The unique Product ID the review pertains to. In the multilingual dataset the reviews for the same product in different countries can be grouped by the same productid.	Because all the reviews are written in the same language, this column was not picked as a predictor variable.
5.	Product_parent	No	Random identifier that can be used to aggregate reviews for the same product.	Product title column is used to group products. As a result, this field was not picked to group the products.
6.	Product_title	Yes	Title of the product.	The product title is used to categorise products by brand. If the product title begins with the word Skechers, it is considered a Skechers brand shoe review.
7.	Product_category	No	Broad product category that can be used to group reviews (also used to group the dataset into coherent parts).	There is only one item in the column that expressly states that the collection contains reviews of the product shoe.
8.	Star_rating	Yes	The 1–5-star rating of the review.	Star rating of the customer review which is a general evaluation of the service quality given by the customer about the product. Based on the rating given the

				sentiment is labelled for training and used as a Target Variable.
9.	Helpful_votes	No	Number of helpful votes.	This feature is not selected it does not hold any useful information about the target variable.
10.	Total_votes	No	Number of total votes the review received.	This field, like the helpful votes field, has no information that may not be used to predict the target variable.
11.	Vine	No	Review was written as part of the Vine program.	It does not include any relevant information for prediction.
12.	Verified_purchase	No	The review is on a verified purchase.	Because there is more interest in the reviews alone, it is less of a concern whether people purchased.
13.	Review_headline	No	The title of the review.	This contains the same information as review body.
14.	Review_body	Yes	The review texts.	Machine learning system uses the actual review field to classify it as positive, negative, or neutral.
15.	Review_date	No	The date the review was written.	This is ineffective in prediction of polarity.

Table 6.1 Feature Selection

Chapter 7 – Model Development

Following the selection of the predictor and target variable, data pre-processing will commence. The machine learning and deep learning model was then built by training and testing it.

7.1 Data Pre-Processing

Data pre-processing, as stated in the literature review, it is a vital component for making data suitable for analysis and building an effective machine learning model. Libraries used for data pre-processing are pandas and NLTK . Appendix E contains the code for data pre-processing. Figure 7.1 depicts a high-level overview of data pre-processing.

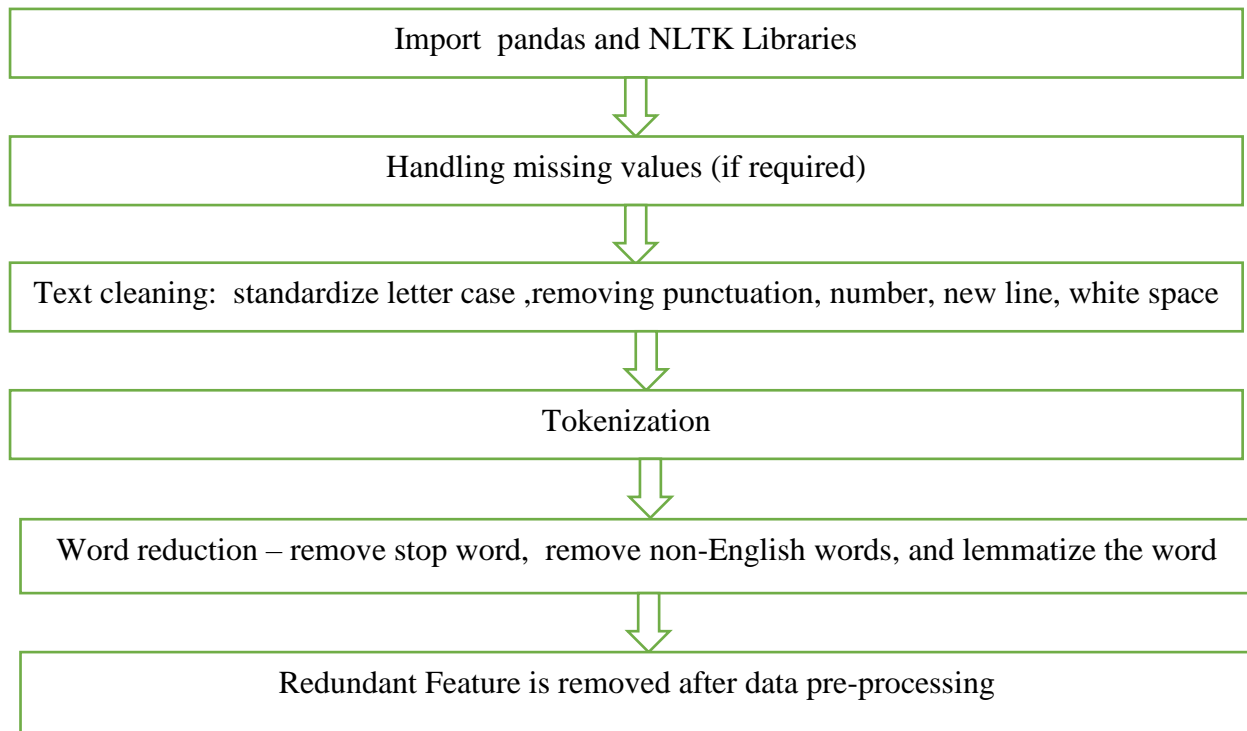


Figure 7.1 Data Pre-processing outline

Amazon product Reviews:

There are several datapoints associated with each brand of shoe in the amazon product review dataset, however only the specific brand such as adidas, Skechers, and Crocs was identified using the field product title. If the product title begins with the brand name indicated above, the shoe's brand name is extracted and placed in a separate column called brand. As a result, the brand column only has three entries: adidas, Skechers, and Crocs. Reviews relevant to each brand are separated and used for further processing.

The pre-processing is the same for all three shoe brands. The sole difference between the machine learning and deep learning models is the vector format utilised to generate them. Four machine learning models and one deep learning model are created in the project, as described in the literature study. Two types of vector representations are explored for the machine learning model: tf-idf and supervised word embedding, and the model is tested. Two vector representations are utilised to evaluate the deep learning model artificial neural network: supervised word embedding and pre-trained word embedding provided by Google called Word2Vec.

- a. Blank Values:** The dataset is verified for blank values and no blank values are identified, hence no datapoints are lost and the entire dataset is used for model building. When the three brands' shoes reviews adidas, Skechers, and Crocs are considered, the total number of datapoints in the dataset is 41,583. The total number of datapoints for adidas shoe brand reviews is 9887, for Skechers 15,397, and for crocs 16,299.
- b. Text Cleaning:** Each statement in the dataset must be changed to lower case before being cleaned. Because Python treats the same term in various cases as though it were two separate words. Consider the word "Words" and "words" as two distinct words. The numbers and special characters will remain unchanged because of the method employed to transform the word to lowercase. So, the next stage is to clean up the text and remove any words that aren't relevant to the text's analysis. As a result, the regular expression library in Python is used to remove digits, punctuation, new blank lines, and white space.
- c. Tokenization:** Tokenization is an important component of Natural Language Processing (NLP). It is the process of breaking down a sentence into individual words or smaller parts known as tokens. It divides a sentence into discrete pieces using white space as a delimiter. Utilising this benefit, each statement in the dataset is tokenized before being processed further.
- d. Word reduction:** Word reduction is nothing more than the removal of undesirable words that provide no useful information for text analysis. By following the procedures, the word is reduced in the project. The stop word and non-English terms are eliminated first using the NLTK library, and then each word is lemmatized to obtain the base word.

- **Stop word:** Stop words are words that are commonly used in natural languages and are referred to as useless words or commonly used words such as "the," "a," "an," and "in," which have no meaning for the text analysis process. As a result, these terms must be extracted using the NLTK (Natural Language Toolkit) module provided by python. This library contains stop words in 16 different languages. The corpus of English words was used in this experiment.
 - **Non-English terms:** As stated in the dataset section, the collected dataset used in the project is collection of review obtained from the United States region. Even if it has been determined that some non-English words are present in the dataset, they must be eliminated to solve the text classification challenge. NLTK library is used for the removal of non-English word.
 - **Lemmatization:** Lemmatization is text normalization techniques used in the field of NLP(Natural Language Processing). This approach is in responsibility of classifying various inflected forms of words into root forms. The term "corpora" is lemmatized as "corpus." To obtain the lemma, the project uses the word net lemmatization technique.
- e. **Removal of Redundant feature:** After applying the aforementioned pre-processing approaches, all redundant features are deleted, leaving only the necessary features for the text classification problem described as predictor variable (reviews body) in the dataset.
- f. **Vectorization:** For the machine to comprehend or represent the text in numerical form, many vector representation approaches are available. The project will compare the results of the TF-IDF and supervised word embedding methods on several machine learning algorithms such as Random Forest, Decision Tree, NB, and Logistic Regression/SGD, as described in the literature review. Investigate the differences in performance between the two strategies, word embedding and word2vec, in a deep learning model such an artificial neural network. The following is a detailed description of each of the three ways of text representation techniques or vector representation used:
- **TF-IDF:** The weight of each word in the phrase is calculated using term frequency and inverse document frequency, which indicates the

significance of each word in the document. This method of representation has more advantages than the BOW(Bag of Words) model alone since each word is assigned the weight in zeros and ones. The literature review includes a formula for calculating the weight of each word using TF-IDF Method. The scikit learn library's pipeline module, on the other hand, combines vectorizer and algorithm. As a result, Tf-idf vectorization is done throughout the project's model development phase.

- **Word Embedding:** Word embedding is transforming words into vector representation so that the algorithm may generalise and predict these words. Then each sentence in the document is one hot encoded and converted into vector. The maximum size of the sequence provided in the document is found to be 291, 300, and 235 for adidas, Skechers, and Crocs. Then, using the TensorFlow library's padded sequence, one hot vector is post padded to equal length. This post padding adds a zero to the end of the sentence to assist with vector sequencing. It's also known as a predictor variable (defined by X on the code).
- **Word2Vec:** word2vec is nothing more than a word vector with one hot encoding (a particular word is encoded as '1' while all other words are encoded as '0') in which each element represents a word in the dictionary. This dictionary should be defined with the largest possible vocabulary. This dictionary has all the words in the vocabulary in alphabetical order with an index. The adidas, Skechers, and Croc's datasets have a dictionary size of 6000,5000,5000 respectively . However, the unique word found in each sample is 4157, 4801, and 5033. The size of the vector space is defined as 300 dimensions as part of the model construction. This form of vector encoding allows for meaningful comparisons between dictionary words. The semantic and syntactic meanings of the words are kept in this fashion. This method of vector representation is suitable for determining the polarity of the reviews in the document. The Word2vec pretrained model produced by Google is employed in the project, as described in the literature study.

In the project, multiple vector representation strategies are employed to compare the results of machine learning and deep learning algorithms.

7.2 Text Analysis

The Amazon product review dataset is used to classify the sentiment associated with the reviews of the goods on the Amazon portal. This classification will categorise the polarity involved with the customer's review as positive, negative, and neutral. Machines can only understand numbers; thus, text analysis is required in this case.

(Naramula Venkatesh and A.Kalaivani, 2019) present a sentiment analysis for online mobile phone reviews based on a visualisation of textual data using a word cloud. Furthermore, the author stated that utilising Word cloud, the classification findings can be visualised in the form of a rating chart based on consumer comments about the product to capture and evaluate any products.

(Rajkumar S. Jagdale, 2016) showed a sentiment analysis of event form the twitter and used the word cloud of a certain event which highlights the most frequently used terms from tweets, as well as the quantity of positive, negative, and neutral tweets from each event.

Even though various strategies are used in NLP (Natural Language Processing) to do text analysis utilising modules available in Python. Based on an examination of the literature, a word cloud looks to be a viable alternative for visualising text data.

The project employed a word cloud to analyse a text and visualise textual data based on the word's frequency of occurrence in the document. Appendix E contains the code for creating a word cloud. Stop words, which do not add any meaning to the sentence and are non-informative, can be removed using the built-in feature of the word cloud library.

For the Amazon shoe product review dataset, a three-word cloud was constructed. Because the task is mainly concerned with determining the sentiment associated with each shoe brand. The data is broken down into three categories based on the brand, such as adidas, Skechers, and Crocs. A word cloud is created for each dataset. The word cloud for each brand shoe review is shown in Figures 7.2(a), 7.2(b), 7.2(c).



Figure 7.2 (a): word cloud Skechers shoe brand reviews



Figure 7.2 (c): word cloud Croc's shoe brand reviews



Figure 7.2(b): word cloud for Skechers shoe brand reviews

All three brand shoes review datasets have prominent words like as comfortable, love, shoe, wear, good, like, foot, pair, bought, comfort, and many more, as shown in the above image. It implies that the reviews are more focused on the positive reviews than on the negative and neutral reviews. It should also be noted that the substance of the comments in all three datasets is connected to shoes and their comforts.

It is evident from this that the data is weighted toward positive responses. As a result, it must be resampled to ensure that each class of reviews or comments, such as positive, negative, and neutral, is balanced.

7.3 Training and Test and Validation datasets

The dataset is separated into train, test, and validation datasets after pre-processing and text analysis. The training dataset is used to train both machine learning and deep learning models, and the validation dataset is used to validate the performance of trained models. Finally, the trained model is evaluated using the test dataset. The dataset is divided with the aid of the scikit learn python library. The flow of partitioning the dataset is depicted in Figure 7.3.

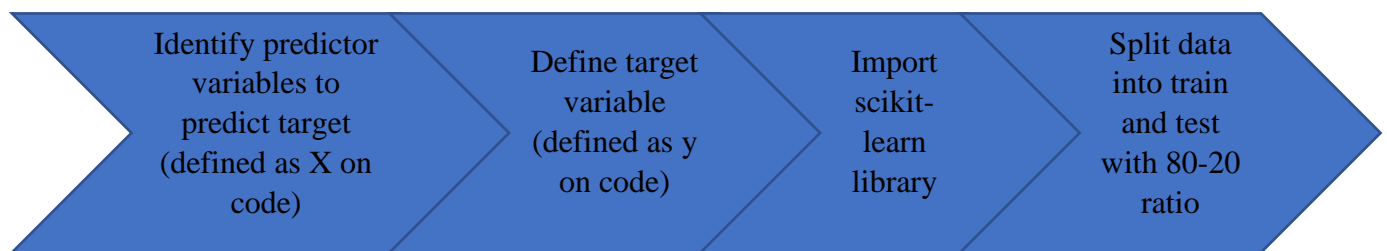


Figure 7.3: Flow to split train and test datasets

The dataset is broken down into three sections: train, valid, and test. Initially, the data is separated into 80:20 train and test groups. The trained model dataset is then separated into two sets: train and validation, with a validation split of 20%. As previously stated, the scikit library is used for all this splitting. The model is trained using the train dataset, with a portion of the train dataset kept aside for validation purposes. 20% validation split from the train dataset was employed in this project. The test dataset consists of data samples that are not used to create or fine-tune the model. Using that test dataset, the trained model is then tested or evaluated.

7.3.1 Handling Imbalanced dataset:

The dataset is unbalanced, as shown via a text analysis utilising a word cloud. Also, the distribution of datapoints in each class of reviews is indicated, and it is stated that datapoints for each class must be balanced before the training begins.

The train dataset must be resampled to balance the datapoints in each type of review, such as positive, negative, and neutral once the dataset has been separated into train and test. Resampling the train data aids in the development of an efficient model that is not biased toward one type of review.

For balancing each class of reviews in the dataset the project used SMOTE Tomek techniques for machine learning model. These techniques will resample the train data for model building. SMOTETomek is a hybrid method which is a mixture of both up sampling(SMOTE) and down sampling(Tomek). Oversampling or up sampling is nothing but the class containing less data is made equivalent to the class containing more data by duplicating it. Down sampling samples down or reduces the samples of the class containing more data equivalent to the class containing the least sample. It is also called as under sampling. This method can be import using the imblearn (imbalanced learn) module in python. Below table represent the distribution of each class before resampling(train data) and after resampling(using smotetomek) the datapoints for each dataset: adidas, Skechers and crocs.

Adidas Dataset	Class 0: Neutral	Class 1: Negative	Class2: Positive
Before Resampling	576	1,589	6,744
After resampling	6,741	6,741	6,740

Skechers Dataset	Class 0: Neutral	Class 1: Negative	Class2: Positive
Before Resampling	893	895	10,529
After resampling	10,525	10,525	10,523

Dataset	Class 0: Neutral	Class 1: Negative	Class2: Positive
Before Resampling	1,115	1,126	10,798
After resampling	10,795	10,796	10,795

But for deep learning model or artificial neural network if the class is imbalanced then the class weight is used for balancing each class in the dataset. Class weight parameter is passed as a dictionary in the model fitting which contain name of each class as key and the values as weight. This parameter can be implicitly imposed or call compute_class_weight function available in the sklearn utils in python which will

compute and balance the class accordingly. The project used function `compute_class_weight` for balancing the class.

7.4 Algorithm Computation

Once the dataset has been chosen and divided into train and test groups, each model's method is computed using train data, and the results are predicted using test data for evaluation.

Machine learning algorithms are computed using the scikit-learn module in Python, whereas deep learning algorithms are computed using TensorFlow and keras. Appendix E contains the code. Figure 7.4 depicts the flow of algorithm calculation for both machine learning and deep learning models.

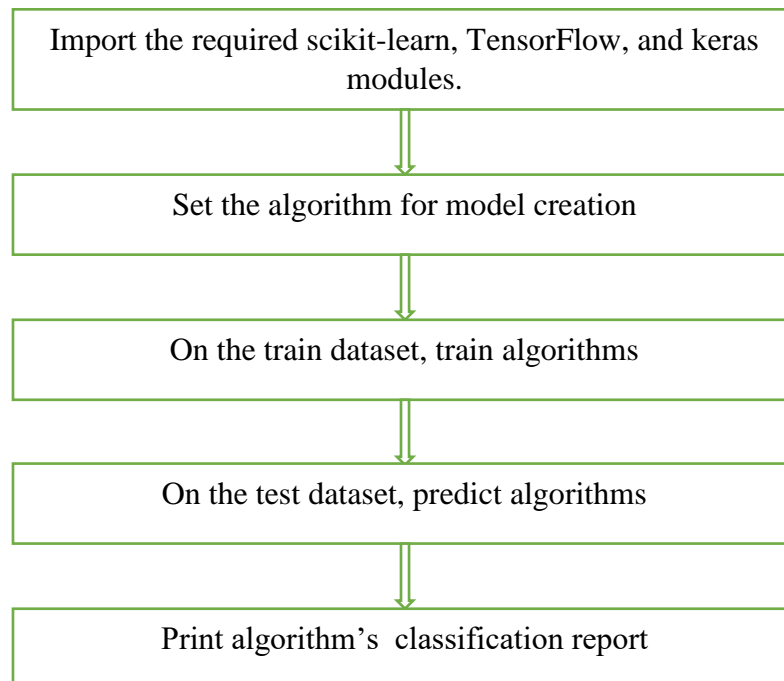


Figure 7.4 outline for algorithm computation

Scikit-learn is used to train four machine learning algorithms, while TensorFlow and the Keras library are utilised to develop deep learning algorithms, as outlined in the literature study. The following are the details:

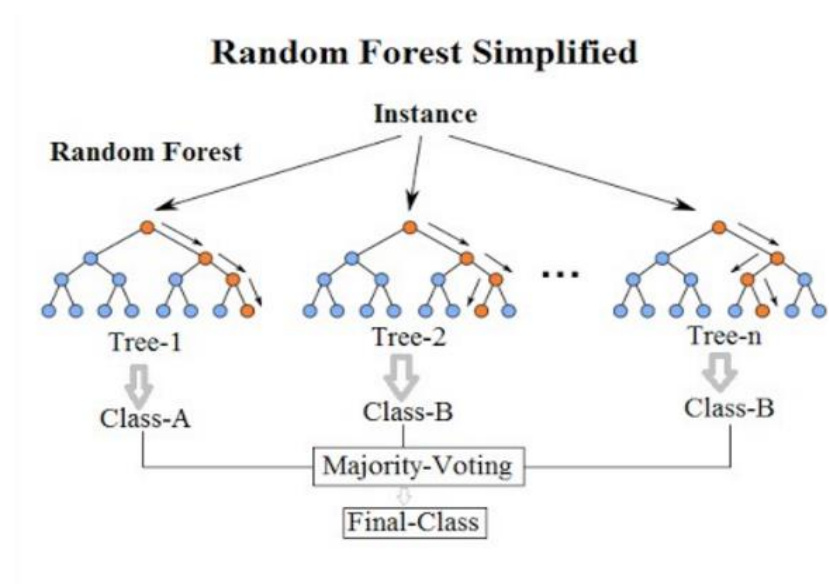
- Random Forest (RF)
- Decision Tree (DT)
- Logistic Regression (LR)/SGD(Stochastic gradient descent)
- Naïve Bayes (NB)
- Artificial Neural Network(ANN)

7.4.1 Random Forest classifier:

Random forest is a supervised machine learning algorithm. It creates a forest out of an ensemble of decision trees, which are commonly trained using the 'bagging' method. Bagging is a process that combines the results of all trained models. To obtain more precise and reliable outcomes, the random forest constructs numerous decision trees and averages the results.

Random forest is used to solve both classification and regression problems, however the focus of this project is on the categorization of sentiment associated with online product reviews. As a result, the model is built using a random forest classifier.

Random forest works by generating several trees with nodes and leaves. Instead of searching for the most significant features, a tree grows by selecting the best feature from a random group of features. As a result, this model usually produces the best outcomes. The random forest with three trees is depicted in the diagram below. The bagging method is used to estimate output. Bagging involves using different samples of data in training data rather than just one sample. Each tree produces one output, which is then combined with the votes of the majority to produce a result.

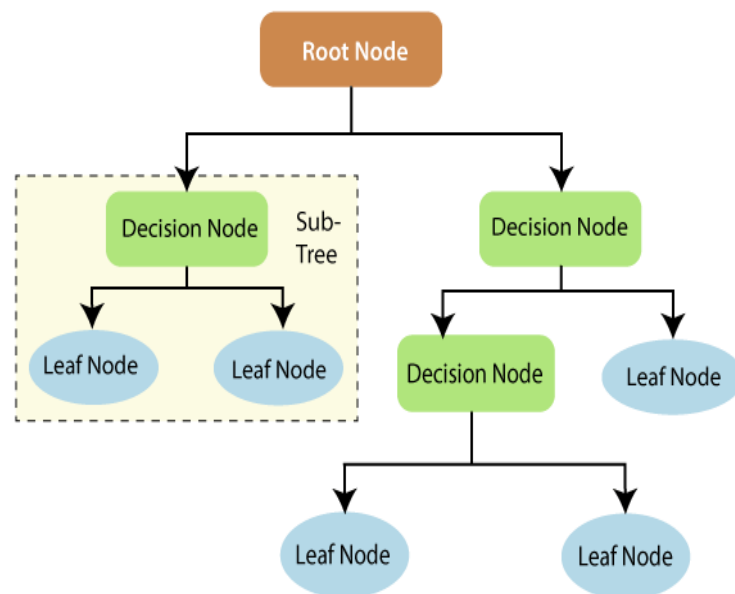


Overfitting is one of the most serious issues in machine learning. Overfitting may be a concern with decision trees. Even though random forest is a collection of decision trees, it usually avoids the problem of overfitting. Furthermore, even with the default

parameter, Random Forest frequently produces good prediction results and effectively handles missing data.

7.4.2 Decision Tree:

The decision tree will take the shape of a tree with three components: a decision node, a leaf node, and a root node. The decision tree works by dividing the train set into branches, which can then be further divided into other branches. This pattern continues until the leaf node is reached. There is no possibility for a leaf node to divide into more nodes.



The components of a decision tree are depicted in the diagram above. The attribute that is utilised to predict the outcome is represented by each node in the decision tree. The decision rules are represented by decision nodes, which offer a link to the leaf node. The outcome is represented by the leaf node.

As like the random forest, decision tree also used for both classification and regression problems. Ways of splitting the node for the categorical outcome (project used categorical outcome) in the decision tree is as follows:

- Information gain
- Gini Index

The concept of information gain is based on entropy, and it measures how much knowledge a feature provides about the class. The goal of a decision tree is to

maximise the amount of information gained. In addition, features with the highest information gain will be divided first. It is given the following formula:

$$\text{Information gain} = \text{Entropy}(\text{decision node}) - \text{wt} * \text{Entropy}(\text{leaf node})$$

$$\text{Entropy} = - \sum p(x) * \log(p(x))$$

$$p(x) = \text{fraction of class in given data}$$

The entropy of a node is a measure of its impurity. The higher the purity of the node, the lower the entropy value.

The Gini index is a metric that determines how frequently a randomly selected feature is wrongly detected. It means that a lower Gini index attribute should be chosen.

7.4.3 Logistic Regression/ SGD classifier:

Logistic regression is a statistical method for analysing a dataset in which the outcome is determined by more than one independent variable. The datapoints are classified using a sigmoid function. The sigmoid function is an s-shaped curve with values ranging from 0 to 1. The sigmoid function is calculated using the following formula.

$$S(z) = 1 / (1 + e^{-\text{value}})$$

Where $s(z)$ = output between 0 and 1, e = base of natural logarithms, $\text{value} = mx + b$

Gradient Descent is a technique for reducing a loss function. The difference between the actual and expected value is the loss function. It is one of the most widely used optimization algorithms, and it is mostly used to optimise neural networks. There are three different types of gradient descent algorithms. Batch gradient descent, stochastic gradient descent (SGD), and mini-batch gradient descent are the three techniques. SGD was utilised in this project to compute the gradient using a single sample.

SGD (Stochastic Gradient Descent) was employed as a linear classifier for logistic regression in this project. Linear SVM and logistic regression are two concepts that the machine learning model can apply with SGD. Loss is a parameter that determines whether the SGD classifier is SVM or Logistic regression. The project uses SGD as a

linear logistic regression in a machine learning model built with word embedding as a vector representation.

7.4.4 Naïve Bayes:

It's a Bayes Theorem-based classification method. It is assumed that the variables are highly independent in this case. In other words, a Naive Bayes classifier believes that the existence of one characteristic has no effect on the presence of another.

The Naive Bayes will come under the family of probabilistic algorithms that uses Bayes Theorem. Bayes Theorem, which describes the probability of a feature based on prior knowledge or probability related to that feature, is used to obtain these probabilities.

The Bayes theorem describes the likelihood of an event based on prior knowledge of conditions that may be relevant to the event. Conditional probability is used.

Conditional Probability: What is the likelihood of something happening if something else has already occurred?

Given a hypothesis H and evidence E, Bayes' Theorem asserts that the probability of the hypothesis before receiving evidence P(H) and the probability of the hypothesis after receiving evidence P(H|E) is:

$$P(H|E) = \frac{P(H) * P(E|H)}{P(E)}$$

P(H|E) = posterior probability of 'H' given the evidence

P(H) = prior probability

P(E|H) = likelihood of the evidence 'E' if the Hypothesis 'H' is true

P(E) = prior probability that the evidence itself is true

The multi nominal Nave bayes are used in this work. It applies the Bayes theorem to each review in the dataset, and the class with the highest probability is chosen as the result.

7.4.5 GridSearchCV

All the above-mentioned algorithms are fine-tuned using GridSearchCV hyperparameter tuning. It's a sklearn model selection package library function. It fits the model to the train data by looping through the given hyperparameter. As an

outcome, the best parameter from the list is chosen. The parameters utilised in each algorithm are summarised in the table 7.1 below.

Algorithm	Parameter used	Best parameter	Description
Random Forest	'max_depth': [10, 20] 'max_features': ['auto', 'sqrt'] n_estimators': [15, 10]	max_depth=20 max_features='sqrt' n_estimators=15	n_estimators: numbers of trees in the forest. max_features: number of features to consider when looking for the best split. max_depth: maximum depth of tree. If the value is none then tree will split until all leaves are split.
Decision Tree	'criterion':['gini', 'entropy'] 'max_depth':[1,2,3,4,5,20,30,40,50,60,70,80,90,120,150]	criterion='gini', max_depth=50,	Criterion: function to Measure the quality of the split. max_depth: maximum depth of tree. If the value is none then tree will split until all leaves are split.
Logistic Regression	'C': [0.1, 0.2, 0.3, 0.4, 0.5]	C=0.3	C: Inverse of regularization strength.
Naïve Bayes	'alpha': (0.1,0.2)	alpha=0.2	alpha: additive smoothing parameter.

Table 7.2 Hyperparameter Tuning

7.4.6: Deep learning model - Artificial Neural Network:

An artificial neural network is made up of layers of connected neurons. It is made up of three layers: the input layer, the concealed layer, and the output layer. The input layer is responsible for bringing the initial data into the system for processing. The input layer and the output layer are connected by a hidden layer. The neuron uses an

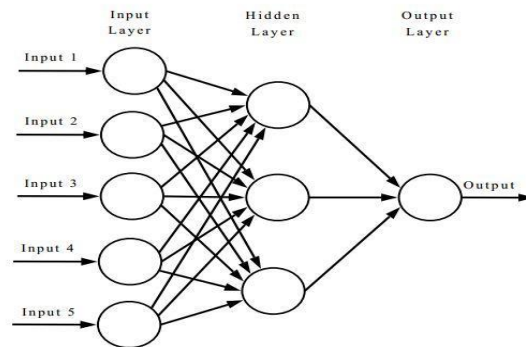
activation function to process the weighted input and produce the output. The output layer is responsible for generating the network's output.

There are two types of operations that take place once the weight is transmitted to the hidden neuron. The weights and input are added together first. Then the activation function takes in, causing the neuron in the hidden layer to activate.

$$y = x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4 + x_5w_5 + \text{bias}$$

$$\sum_{i=1}^n x_i * w_i + \text{bias}$$

$$Z = \text{Act}(y)$$



In deep learning, there are numerous activation functions such as Relu, sigmoid, SoftMax, and so on. This project employed the SoftMax activation function. Softmax activation is used to solve a classification problem with more than two classes, such as the one that is employed in this case. It's the same as sigmoid activation, and the sigmoid activation function's formula is:

$$1/1+e^{(-y)}$$

Where y is the result of the product of the input features, weights, plus bias. The above equation will transfer the values between 0 and 1. When the output value is 0, the neuron is said to be deactivated. The sigmoid output value will be probability. The probability value's sum is always one. The output is the value with the highest probability. The SoftMax activation is calculated using the formula

$$S(x_j) = e^{x_j} / \sum_{k=1}^K e^{x_k}, j=1,2,3,4...K$$

The artificial neural network is trained by activating neurons in each layer by forward propagation, like calculating input features with weight and bias. The neuron in hidden layers is then activated using any of the activation functions. The output of the hidden layer with the weights is then predicted in the output layer. The loss function is now calculated in the output section. The project's used categorical cross entropy as

a loss function. If the loss function value is too high, the weights of the neuron in back propagation need to be adjusted by optimizer for reduce the loss. The rmsprop optimizer was utilised in this project. The neural network is trained by repeating this forward and backward propagation iteratively for many times.

7.4.7 Algorithm Computation Experiments

The project compares the result of different vectorization techniques (TF-IDF vs supervised Word embedding) for machine learning model such as random forest, decision tree, Logistic regression/SGD, and naïve bayes. Likewise, it also compares the result of different vectorization techniques(supervised word embedding vs pretrained word embedding called word2vec) for deep learning model artificial neural network.

Figure 7.5 shows the algorithm experiment flow. Overall, algorithm computational experiments are performed for all 4 classifiers in machine learning and one neural network in deep learning are recorded and compared.

1. TF-IDF vectorization followed by 4 machine learning classifiers
2. Supervised word embedding followed by machine 4 machine learning classifier
3. Supervised word embedding followed by artificial neural network
4. Pre trained word embedding(Word2Vec) followed by artificial neural network

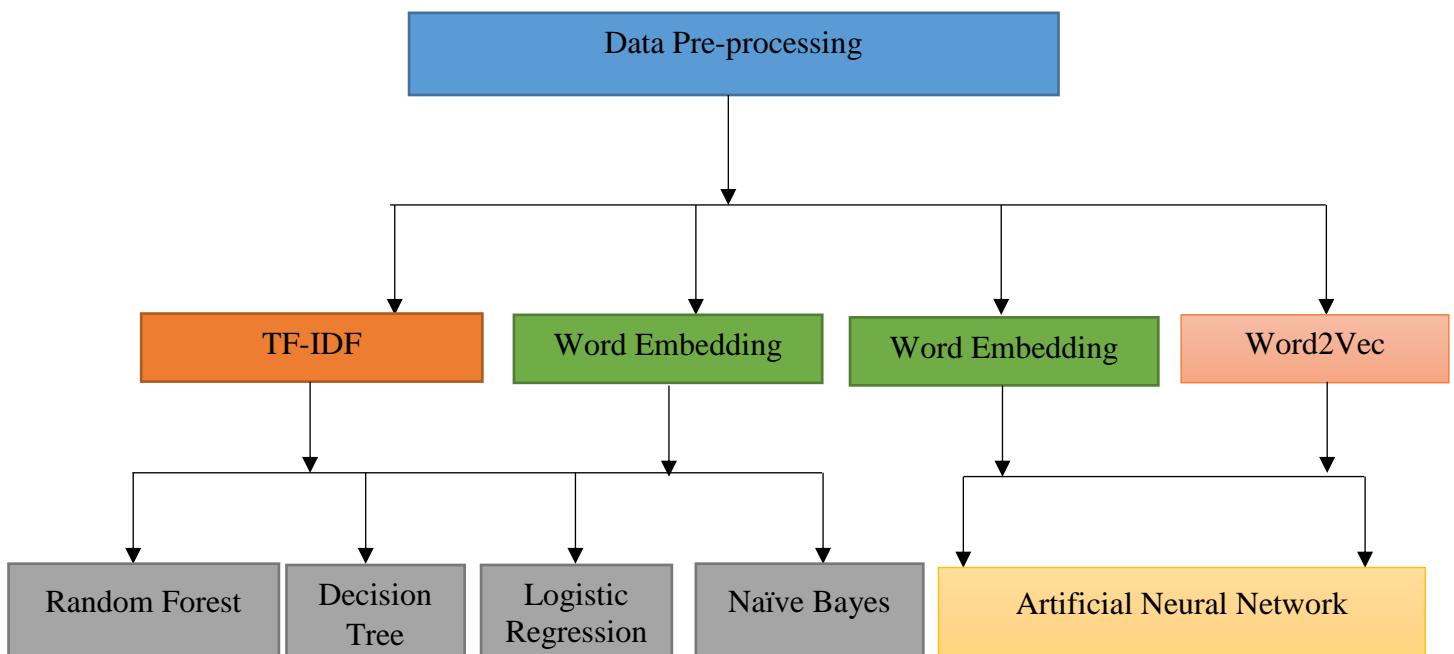


Figure 7.5 Algorithm Experiment Flow

➤ **TF-IDF vectorization followed by 4 machine learning classifiers:**

In project for the vector representation(TF-IDF) followed by machine learning model used Scikit learn pipeline module, that create a machine learning pipeline. With the help of pipeline vectorization is performed during the model development. After the pipeline was set up, each machine learning algorithm was run in a Jupyter notebook and trained on the train dataset before being predicted on the test dataset.

Figure 7.6 shows the experimental set for vector representation(TF-IDF) followed by machine learning model for obtained dataset. Code for this is present in the Appendix E.

```
text_clf1 = Pipeline([ ('tfidf', TfidfTransformer()),('clf', mla)])
print(text_clf1)
kfold = StratifiedKFold(n_splits=5)
scores = cross_val_score(text_clf1, x_resampled, y_resampled, cv=kfold, scoring='accuracy')
validation_score = scores.mean()
print('cross validation:',validation_score)

### fit the model
from sklearn.metrics import accuracy_score
text_clf1.fit(x_resampled,y_resampled)
### prediction
y_pred_train = text_clf1.predict(x_resampled)
y_pred_test = text_clf1.predict(x_test)
train_score = accuracy_score(y_pred_train,y_resampled)
test score = accuracy score(y_pred_test,v test)
```

Figure 7.6 set up for TF – IDF followed by machine learning model

➤ **Supervised word embedding followed by machine 4 machine learning classifier:**

Word embedding for the machine learning model used TensorFlow, keras package that create a one hot encoding and padded sequence through which the word embedding is developed, and it is passed to the machine learning model for the model development. Then, in the Jupyter notebook, each machine learning algorithm was trained on the train dataset and predicted on the test dataset.

Figure 7.7 shows the experimental set for word embedding followed by machine learning model for each dataset such as adidas, crocs, and Skechers. Appendix E contains the necessary code.

```

## define size of vocabulary and max length of the sentence in the review dataset
vocab_size = 5032
max_length = 235

## name the y variable as sentiment
import numpy as np
sentiment = np.array(crocs['positivity'])
sentiment

from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.preprocessing.sequence import pad_sequences
## one hot encoding for the words
encoded_reviews = [one_hot(d, vocab_size) for d in crocs['st_cleaned_message']]

## padding
padded_reviews = pad_sequences(encoded_reviews, maxlen=max_length, padding='post')

## define X and y variable
x=padded_reviews
y=sentiment
return x,y

```

Figure 7.7 set up for word embedding

➤ **Supervised word embedding followed by artificial neural network:**

The word embedding is introduced to the artificial neural network using the TensorFlow and keras package. The neural network is then built by adding a keras layer, such as flatten and dense. The model neural network is then compiled and fitted. A train dataset is used to train or create the model, and a test dataset is used to predict it.

Figure 7.8 shows the experimental set for word embedding followed by artificial neural network. Code for this is present in the Appendix E.

```

# define the model
model = Sequential()
model.add(Embedding(vocab_size, embeded_vector_feature, input_length=max_length))
model.add(Flatten())
model.add(Dense(units = 16, kernel_initializer = 'he_uniform', input_dim = (x.shape[1])))
model.add(Dense(3, activation='softmax'))

# compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
# summarize the model
model.summary()

# model fit
history = model.fit(x_train,
                    y_train,
                    callbacks=[es],
                    class_weight=weights,
                    epochs=8000000,
                    batch_size=500,
                    shuffle=True,
                    validation_split=0.20,
                    verbose=1)

```

Figure 7.8 set up for artificial neural network

➤ **Pre trained word embedding(Word2Vec) followed by artificial neural network**

This way of vectorization is same as word embedding but only difference is the pertained model is downloaded and used as a weighted matrix for the embedding layer and build the neural network. It uses genism library for downloading the word2vec model. From pre-trained model it calculates the weight matrix for words in the vocabulary with 300 as a dimensions vector size. Figure 7.9 below shows the how the pre-trained model is used as an embedding layer in artificial neural network.

```
import gensim.downloader as api
word2vec_model = api.load('word2vec-google-news-300')

[=====] 100.0% 1662.8/1662.8MB downloaded

### padding the sequence to equal length by post padding
from tensorflow.keras.preprocessing.sequence import pad_sequences
padded_reviews = pad_sequences(sequence, maxlen=max_length, padding='post')

### converting the vector to matrix of shape (4157,300)
unique_words = len(word_index)
total_words = unique_words + 1
no_of_skipwords = 0
embedding_dim = 300
embedding_matrix = np.zeros((total_words, embedding_dim))
for word, index in tokenizer.word_index.items():
    try:
        embedding_vector = word2vec_model[word]
    except:
        no_of_skipwords = no_of_skipwords + 1
        pass
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
print('shape of matrix is:', embedding_matrix.shape)
print('no of skipped words:', no_of_skipwords)

embedding_layer = Embedding(total_words, embedding_dim, weights=[embedding_matrix], input_length=max_length, trainable = False)
model = Sequential()
model.add(embedding_layer)
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
# compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
# summarize the model
model.summary()
```

Figure 7.9 set for word2vec

7.4.8 Evaluation Metrics:

True Positives (TP): True positives are the cases where the predicted value is positive and actual is also positive. **Example:** The case where the review is actually positive review and the model classified as positive review.

True Negatives (TN): True negatives are the cases where the predicted value is negative and actual is also negative. **Example:** The case where the review is negative review, and the model is classified as negative review.

False Positives (FP): False positives are the cases where the predicted value is true the actual value is false. This is also called as type I error. **Example:** A review is a negative review and the model classified as positive.

False Negatives (FN): False negatives are the cases where the predicted value is false and actual value is true. This is also called as type II error. **Example:** A review is a positive review and the model classified as negative.

Evaluation metrics used in the project are precision, recall, f1-score, and accuracy.

Accuracy: Accuracy in classification is the ratio of correct predictions made by the model to the overall prediction.

Precision: Precision refers to the percentage of records that our model identified to be relevant.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Recall: The ability to find all relevant instances in a dataset is referred to as recall.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

F1-score: It is single metric that combines recall and precision using the harmonic mean. F1 Score is the weighted average of Precision and Recall.

$$\text{F1-score} = 2(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

7.4.9 Testing Algorithm performance

The model is predicted on the test data once it has been trained on the train data. The prediction is compared to each class to evaluate the algorithm's performance. The scikit learn package, which has a function to build a classification report, is used to compare each class. Each review class is assigned a precision, recall, and f1-score in the classification report. The classification report for the random forest classifier (tf-idf following machine learning model) for the croc's dataset is shown in Figure 7.10. Appendix E contains the necessary code.

```

classification report:
              precision    recall  f1-score   support

     0       0.396       0.112       0.174       1021
     1       0.243       0.340       0.283        215
     2       0.652       0.861       0.742       2024

 accuracy          0.592       3260
 macro avg       0.430       0.437       0.400       3260
 weighted avg    0.545       0.592       0.534       3260

```

Figure 7.10 classification report of Random Forest

The performance evaluation metric for all the experimental combinations for the adidas shoe brand dataset is shown in the Table 7.2 below. P- precision, R- recall, F1- f1score. RT – Random Forest, DT – Decision Tree, SGD - Stochastic Gradient Descent, LR- Logistic Regression, NB – Naïve Bayes, ANN- Artificial Neural Network.

Multi class classification – Adidas dataset											
Vectorization method	Algorithm	For class-neutral			For class-negative			For class - Positive			Accuracy
		P	R	F1	P	R	F1	P	R	F1	
TF-IDF	RF	7%	25%	11%	55%	16%	25%	76%	90%	83%	70%
TF-IDF	DT	7%	9%	8%	74%	21%	32%	74%	92%	82%	69%
TF-IDF	SGD	25%	14%	18%	48%	28%	35%	81%	93%	87%	75%
TF-IDF	NB	30%	12%	17%	45%	42%	44%	80%	92%	85%	74%
Word - embedding	RF	24%	13%	16%	24%	15%	18%	80%	90%	84%	72%
Word - embedding	LR	27%	9%	13%	33%	9%	15%	58%	89%	70%	54%
Word - embedding	DT	18%	9%	12%	26%	14%	18%	77%	90%	83%	70%
Word - embedding	NB	14%	8%	11%	14%	8%	10%	77%	88%	82%	69%
Word - embedding	ANN	15%	20%	18%	35%	48%	41%	95%	89%	92%	81%
Word2vec	ANN	18%	24%	21%	34%	46%	39%	93%	88%	91%	80%

Table 7.2 performance evaluation metrics for all experimental combination of adidas dataset

1. **Highest accuracy:** The highest level of accuracy was 81 percent. This is the result of using word embedding as a vectorization technique for an artificial neural network.
2. **Comparing TD-IDF, Word embedding for machine learning model:** when comparing to TF-IDF form of vectorization, word – embedding performed well for the machine learning model.
3. **Comparing word embedding and word2vec for the deep learning model:** Both vectorizations performed admirably, with accuracy scores of 80 percent and higher.
4. **Comparing algorithm:**
 - a) Random Forest: For both types of vectorizations, it attained an accuracy of higher than 70%.
 - b) Decision Tree: It also performed well and scored nearing to 70% for both type of vectorization.
 - c) Logistic regression: Out of all the algorithms, this is the one that performs the worst.
 - d) SGD(Stochastic Gradient Descent): Even though this algorithm uses logistic regression internally, it achieved a 75 percent accuracy score.
 - e) Naïve Bayes: It performed equally well when comparing to other algorithms.
 - f) ANN(Artificial Neural Network): This has attained the maximum level of accuracy, with an accuracy of 81 percent.

Observation summary:

Based on the observation, for multiclass classification for amazon product review shoes. it is observed that ANN(Artificial Neural Network) performed better then other algorithm. word embedding form of vectorization is better than TF-IDF for machine learning model. Both word embedding and pre-trained word2vec vectorization performs well for the deep learning model. All algorithm categories the review in positive categories then the negative and the neutral categories, due to this the f1 score for positive reviews are more than other two. Logistic regression achieved least accuracy, so it is not effective for this dataset. Because the ANN performs well, the test prediction is kept in the csv file for the adidas dataset so that the actual and predicted values can be compared.

Table 7.3 below shows the performance evaluation metric for all the experimental combination for Skechers shoe brand review dataset. P- precision, R- recall, F1-

f1score. RT – Random Forest, DT – Decision Tree, SGD - Stochastic Gradient Descent, LR- Logistic Regression, NB – Naïve Bayes.

Multi class classification – Skechers dataset											
Vectorization method	Algorithm	For class-neutral			For class-negative			For class - Positive			Accuracy
		P	R	F1	P	R	F1	P	R	F1	
TF-IDF	RF	12%	11%	12%	35%	13%	19%	76%	89%	82%	68%
TF-IDF	DT	9%	11%	10%	67%	18%	29%	75%	92%	83%	69%
TF-IDF	SGD	26%	15%	19%	48%	31%	38%	82%	92%	87%	75%
TF-IDF	NB	29%	14%	19%	48%	44%	46%	82%	92%	87%	75%
Word - embedding	RF	17%	10%	13%	24%	14%	18%	79%	89%	84%	71%
Word - embedding	LR	39%	8%	14%	25%	10%	14%	51%	90%	65%	48%
Word - embedding	DT	27%	13%	17%	30%	17%	22%	77%	91%	83%	70%
Word - embedding	NB	4%	6%	5%	26%	11%	16%	80%	88%	84%	71%
Word - embedding	ANN	21%	22%	21%	37%	65%	47%	94%	88%	91%	81%
Word2vec	ANN	22%	28%	24%	33%	49%	39%	93%	87%	90%	80%

Table 7.3 performance evaluation metrics for all experimental combination of Skechers dataset

- Highest Accuracy:** highest accuracy is 81% for word- embedding with ANN(Artificial neural network).
- Comparing TD-IDF, Word embedding for machine learning model:** supervised word embedding performed well compares to the tf-idf form of vectorization.
- Comparing word embedding and word2vec for the deep learning model:** Both vectorization methods worked equally well, with greater than 80% accuracy.
- Comparing algorithm:**
 - Random Forest: It has achieved the average accuracy score of 70% for both the vectorization.

- b) Decision Tree: This algorithm, like random forest, performs admirably and achieves a score of up to 70%.
- c) Logistic Regression: In comparison to other algorithms, it scored 48 percent, which is the lowest.
- d) SGD(Stochastic Gradient Descent): It works in the same way as logistic regression, but it scored 75%.
- e) Naïve Bayes: The accuracy of this method was more than 70%.
- f) ANN(Artificial Neural Network): This is highest performing algorithm out of all five algorithms.

Observation Summary:

Based on the observations, for Skechers data multi class classification for amazon product reviews, ANN performed better than other algorithm and achieved an accuracy score of 81%. Both Word embedding and Word2vec way of vectorization worked better than TF-IDF. Because the ANN performs well, the test prediction is kept in the csv file for the Skechers dataset so that the actual and predicted values can be compared.

Table 7.4 below shows the performance evaluation metric for all the experimental combination for crocs shoe brand review dataset. P- precision, R- recall, F1-f1score. RT – Random Forest, DT – Decision Tree, SGD - Stochastic Gradient Descent, LR- Logistic Regression, NB – Naïve Bayes.

Multi class classification – Crocs dataset											
Vectorization method	Algorithm	For class-neutral			For class-negative			For class - Positive			Accuracy
		P	R	F1	P	R	F1	P	R	F1	
TF-IDF	RF	40%	11%	17%	24%	34%	28%	65%	86%	74%	59%
TF-IDF	DT	7%	9%	8%	60%	20%	31%	73%	90%	81%	66%
TF-IDF	SGD	27%	16%	20%	48%	44%	46%	82%	91%	86%	74%
TF-IDF	NB	31%	17%	22%	46%	40%	43%	80%	90%	85%	73%
Word - embedding	RF	25%	14%	18%	25%	15%	19%	76%	87%	81%	67%
Word - embedding	LR	25%	9%	14%	28%	9%	14%	57%	88%	69%	52%

Word - embedding	DT	24%	12%	16%	28%	16%	21%	73%	89%	80%	66%
Word - embedding	NB	10%	8%	9%	11%	8%	9%	80%	85%	82%	68%
Word - embedding	ANN	23%	36%	28%	46%	56%	51%	94%	86%	90%	79%
Word2vec	ANN	25%	32%	28%	39%	58%	47%	92%	84%	88%	78%

Table 7.4 performance evaluation metrics for all experimental combination of crocs dataset

1. **Highest accuracy:** highest accuracy is 79% for word embedding with ANN model.
2. **Comparing TD-IDF, Word embedding for machine learning model:** supervised word embedding performed well when compares to other TF-IDF vectorizer.
3. **Comparing word embedding and word2vec for the deep learning model:** Both word embedding and pre-trained word2vec performed well on the artificial neural network.
4. **Comparing algorithm:**
 - a) Random forest: it achieved an average accuracy score of 60% for both the vectorization.
 - b) Decision Tree: it performs well and obtained an accuracy score of 66%.
 - c) Logistic Regression: logistic regression has achieved 52%. This is the least performing algorithm out of 5 algorithms used.
 - d) SGD(Stochastic Gradient Descent): this classifier achieved accuracy score of 74% which is the third highest score when compares to other algorithms.
 - e) ANN(Artificial Neural Network): ANN has achieved highest accuracy score of 79%.

Observation Summary:

Based on the observation for the crocs brand shoe reviews data multiclass classification, ANN performed well as like the other two datasets. It achieved highest accuracy score of 79% for word embedding form of vectorization.

Because the ANN performs well, the test prediction is kept in the csv file for the crocs dataset so that the actual and predicted values can be compared.

7.5 Final Analysis:

After analysing the data and looking at all the machine learning and deep learning algorithm experiment combinations, it was determined that each approach performed equally well on the adidas, Skechers, and Croc's datasets. The highest accuracy result for Adidas(81%), Skechers(81%), and crocs(79%).

In terms of vectorization used for machine learning model, word embedding performed well when compared to TF-IDF vectorizer. Both vectorization, such as supervised word embedding and word2vec, worked well for the ANN in the deep learning model, with an accuracy variance of 1% in all datasets used.

When comparing the machine learning and deep learning model for the multiclass classification of amazon product review, deep learning model with word embedding or word2vec performed well. And, based on each dataset's observations, the f1 score for the positive class is higher. Good prominent terms are impacting the datasets, as seen in the word cloud, implying that positive reviews are more frequent.

The artificial neural network is the best way for classifying online product reviews, according to the evaluation metric Logistic regression is not performed well for this task. Naïve Bayes, and SGD(Stochastic Gradient Descent) achieved a highest score, which can be the choice after ANN.

To summarise the findings, utilising word embedding or word2vec vectorization for the Artificial Neural Network to classify online product evaluations yields better results than other approaches. The best models to consider for the second choice for the classification problem are Naive Bayes and SGD (Stochastic Gradient Descent).

Chapter 8: Project Conclusion

Future scope of work:

For the further work to present new improved method for sentiment classification by analysis various deep learning algorithm which has proved to give more accuracy. Furthermore, consider identifying product quality by examining each feature of the product, analysing sentiment based on that, and making a recommendation to the end consumer.

Final Conclusion:

The Artificial Neural Network proved to be a better medium for doing sentiment analysis for online product reviews, with accuracy in the range of 79–81 percent, according to the results of the trial. Furthermore, the project's purpose has been met, since the project develops a classification system that can classify reviews as positive, negative, or neutral.

References:

1. S. Pradha, M. N. Halgamuge and N. Tran Quoc Vinh(2019). "Effective Text Data Preprocessing Technique for Sentiment Analysis in Social Media Data," 2019 11th International Conference on Knowledge and Systems Engineering (KSE), (pp. 1-8). Da Nang, Vietnam: IEEE.
2. Tanjim Ul Haque, Nudrat NAWAL Saber, Faisal Muhammad Shah(2018). "Sentiment analysis on large scale Amazon product reviews" 2018 IEEE International Conference on Innovative Research and Development (ICIRD), (pp. 1-6). Bangkok, Thailand.
3. G. Orellana, B. Arias, M. Orellana, V. Saquicela, F. Baculima and N. Piedra(2018), "A Study on the Impact of Pre-Processing Techniques in Spanish and English Text Classification over Short and Large Text Documents," International Conference on Information Systems and Computer Science (INCISCOS),(pp. 277-283). Quito, Ecuador: IEEE.
4. M. Mhatre, D. Phondekar, P. Kadam, A. Chawathe and K. Ghag(2017), "Dimensionality reduction for sentiment analysis using pre-processing techniques" International Conference on Computing Methodologies and Communication (ICCMC), (pp. 16-21). Erode, India: IEEE.
5. Balakrishnan, Vimala & Ethel, Lloyd-Yemoh. (2014). "Stemming and Lemmatization: A Comparison of Retrieval Performances". Lecture Notes on Software Engineering. 2. 262-267. 10.7763/LNSE.2014.V2.134.
6. Bijoyan Das, Sarit Chakraborty (2018). "An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation". arXiv:1806.06407
7. Seyed Mahdi Rezaeinia, Ali Ghodsi, Rouhollah Rahmani(2017). "Improving the Accuracy of Pre-trained Word Embeddings for Sentiment Analysis". arXiv:1711.08609
8. A. Kafi, M. S. Ashikul Alam, S. Bin Hossain, S. B. Awal and H. Arif(2019), "Feature-Based Mobile Phone Rating Using Sentiment Analysis and Machine Learning Approaches," 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT),(pp. 1-6). Dhaka, Bangladesh :IEEE.

9. S. N. Singh and T. Sarraf(2020), "Sentiment Analysis of a Product based on User Reviews using Random Forests Algorithm," 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence),pp. 112-116, Noida, India. IEEE.
10. P. Lakshmi Prasanna, Dr. D.Rajeswara Rao(2018), "Text Classification using artificial neural networks" International Journal of Engineering and Technology(UAE),(pp.603-606).
11. Ayat Zaki Ahmed, Manuel Rodríguez-Díaz(2020), "Significant Labels in Sentiment Analysis of Online Customer Reviews of Airlines" doi:10.3390/su12208683.
12. F. P. Shah and V. Patel(2016), "A review on feature selection and feature extraction for text classification," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), (pp. 2264-2268), Chennai, India. IEEE.
13. Mita K. Dalal, Mukesh A. Zaveri(2011), " Automatic Text Classification: A Technical Review", International Journal of Computer Applications (0975 – 8887), Volume 28– No.2.
14. Naramula Venkatesh , A.Kalaivani(2019), "Word Cloud for Online Mobile Phone Tweets towards Sentiment Analysis" International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8 Issue-6.
15. Rajkumar S. Jagdale et al(2016), "sentiment analysis of events from twitter using open-source Tool " International Journal of Computer Science and Mobile Computing, (pg. 475-485). Vol.5 Issue.4.
16. Shahnawaz and P. Astya, "Sentiment analysis: Approaches and open issues(2017)," International Conference on Computing, Communication and Automation (ICCCA),(pp. 154-158), Greater Noida, India. IEEE.
17. <https://s3.amazonaws.com/amazon-reviews-pds/readme.html>
18. <https://www.statista.com/statistics/1230225/changes-in-online-buying-among-uk-consumers-since-covid-19/>

Appendix A:

Task 2: Research Project Plan: Sentiment Analysis of Online Product Review using Artificial Neural Network Dhivya Nataraj 29034198

1.Introduction and justification

With the advent increase in the online shopping platforms, many businesses are intended to depend on online product review to make a better analysis on buying pattern of the customers. So, there are large variety of product are available on the internet and consumer review about their purchased product. To find the quality of the product in e-commerce site and recommend the product sentiment analysis can be taken place based on the customer reviews about the product.

And this type of analysis brings beneficial for the businesspeople to find the selling product and non-selling product. Both large quantity and various forms of online reviews bring difficulties for traditional machine learning algorithms to classify all the heterogeneous reviews. Sentiment analysis has gained much attention in recent years and Artificial Neural Network plays a major role in sentiment or opinion analysis of the product.

2.RESEARCH QUESTIONS, AIMS AND OBJECTIVES

This project aims to develop a system which will analyse the reviews for a set of products which makes easier for both the customer and stakeholder to find the quality and make decision accordingly.

2.1 Research question

How can Artificial neural network learning algorithm be applied to perform sentiment analysis of online products?

2.2 Objectives

Deep learning Model:

1. Investigate existing **pre-processing techniques**
 - **POS Tagging**
 - **Automatic labelling of the input data**
 - **Named entity extraction technique for boundary analysis**
2. Investigate existing algorithms (regression and classification) and either select or build the most appropriate option for this project.
3. Research (and build if necessary) a suitable classification **system**, which will use Artificial neural network to classify the reviews for improving the quality of the product and for better decision-making process in business.
4. **Evaluate performance** of the proposed model.

User Interface:

5. **Design a prototype** User Interface which will present the Machine Learning output data in a format both appropriate for the target audience and the collected data.
6. **Develop the UI** using appropriately researched and selected technologies.

Evaluation:

7. **Design suitable process** for evaluating the performance of the combined Deep learning model.
8. Perform the **evaluation based on the various performance metrics**.

2.3 Deliverable

A system which performs analysis for the given products and classify the reviews of the customers as positive feedback or negative feedback and to make an analysis on the buying pattern of the customer. The analysis will be optimized for higher dimensionality of data.

3.Literature Review

Extracting the reviews about the mobile (Xiaomi Mi3) in one of the biggest e-commerce sites in India called flipkart. Reviews like star rating, date of review, text of the review and score. Then text review is processed using natural language processing and classified as positive review and negative review using Naive Bayes [1]. Overall score is arrived by counting the rating, dates of review, vote of the review and the feature classification. Once the review is extracted the text review is extracted to perform the natural language processing. Each sentence is tagged and classified as positive and negative using Naive Bayes classifier. In their work feature based extraction (e.g., Speed display, flash etc.) is done to improve the product in terms of marketing purpose and for recommendation. The product score calculation is performed by adding the average star rating, polarity of review, age of the review and average score. The authors have used additional features apart from the original review text and have also done feature-based analysis like speed, display, etc. This analysis will be very useful for the end customers to get accurate information about the product.

The author used Support vector machine[2] for the classification based on the polarity of the review. Hybrid approach for the prediction of sentiment analysis which gives the higher accuracy, precision, and efficiency. Movie reviews are classified using SVM(Support Vector machine). Deep learning approach was used to find the sentiment behind the tweets and its application need to be further improved for the polarity identification.

Product is selected and POS tagging is performed. Score of each word in the sentence is calculated and clustering is performed. Finally, the work shows the success or failure of the product using SVM classification. The proposed work composed of five steps. In text pre-processing, the token is assigned for each sentence and frequently used words are removed. Clustering and SVM classification are performed to find the polarity of the review. Finally, the graph is display for number of positive terms and negative terms. The authors of this research work have used Part of Speech Tagging which help us understand and gain much more information about the review than just analysing the sentiment alone directly on the raw text. Also clustering technique is used on the data and that helps group similar reviews together. These two additional pre-processing steps can be used as a generic pro-processing step in Text based classification problems as they will significantly enhance the outcomes of classification algorithm used in the final stage of text classification.

The author in [3] used WEKA classifier which uses five group of classification called Bayes, distance-based classifier, discriminant classifier, neural network and decision tree. WEKA classification helps to find the polarity of the product and doing Analytic Hierarchy Process the purchase behaviour of the consumer can be identified. Novel feature weighting algorithm is used for sentiment classification where the document is weighted, and vector is assigned.

Ontology based sentiment classification uses a lexical variable for classification of the online product review.

Weka tool that is simple to do analysis with UI driven approach, which helps initial learning and working on Machine Learning & Data Mining activities easier. The Random Forrest model has given 100% accuracy implies that the model has overfit on the input data. But the Lexicon based Data Pre-processing technique to automatically label data before classification helps if the huge amounts of data that needs to be labelled for further analysis using machine learning algorithms.

In [4] the author used a fuzzy function to analysis the sentiment of the product and find the polarity of it. The approach is to do pre-processing for removing the noise and extract the feature. The fuzzy logic performs good in the prediction of sentiment analysis. There are three levels of sentiment analysis can be performed they are document level, sentence level and phrase level. In document level each review is consider for analysis.

The author referred the fuzzy logic for finding the emotional word in the document. Some of the technologies used to find the sentiment analysis which are referred in the paper are finding the minimum cut in the graph this helps to find the cross sentence, adaptive interface fuzzy logic for the prediction of sales.

Named entity extraction pre-processing is performed to find the organisation name, branch name, product name, which is then divided into two sub-division one for identify the boundaries and another is to find its type. Boundaries is identified by counting the occurrence of the word in the document and its importance. The model used two approaches tree bank for feature extraction and fuzzy opinion mining for classification of sentiments. The phases in proposed work are pre-processing, feature extraction and classification. In pre-processing stage where the unstructured dataset is structured. In feature extraction where the structure review is feed into it and frequently occurring noun and phrase are treated as a feature word. The polarity of the word is identified using the fuzzy value. Finally, fuzzy score is calculated. The proposed system does not require labelled training set for classification.

The automatic labelling of the input data helps in time consuming. Named entity extraction technique and the boundary analysis classify the data more accurately. The non-dictionary words also being included in the analysis by replacing them with their synonyms is a good technique as usually this information is lost because the non-dictionary words usually discarded in the pre-processing stage itself, which results in loss of information.

The author in [5] referred some of the techniques they are collaborative filtering for recommendation system. Product filtering is performed using Artificial neural network and fuzzy logic for removing the unwanted information in website that might change the customer's mind. To increase the shopping on e-commerce the value-added search mechanism is implemented. Author states that there are many techniques are used in e-commerce, but nothing is specific on quality of the product.

The method using ANN work like the user comment is feed into ANN as an input and rating of this user comment from 1 to 10. And based on the learning ANN will process the data. The result shows that product score will be in the range of 1-100%. The implementation starts from training the neural network and it will provide the result in the form of 1-100% scale as a regression output.

The result provided benefit to the e-commerce site but helping the consumers identify better quality products. By doing so the product quality is available as a measurable quantity, so it makes possible for the e-commerce companies to remove or discontinue the product from the site if the quality is not too acceptable. The author states that the approach looks good in the prediction of quality of the product. Unlike most research work where the sentiment analysis of reviews focuses on the classification of the text, the authors of this research work has taken the text and performed regression. The regression approach makes the overall results much more useful as they can be directly used for ranking the products in terms of product quality. Here fuzzy logic used in the test pre-processing stage to remove text that does not contribute to the analysis outcomes. Artificial Neural Networks is one of the most accurate algorithms useful in both the regression and classification problems and hence provide very good accuracy in this research work.

4.Research Design

4.1 Research Philosophy, Approach & Methodology

Pragmatistic philosophy approach is used in Artificial neural network model to acquire the objectives and resolve the research issues. The research starts with a literature survey where existing work is analysed and based on the important features and drawbacks in the existing models where a new model can be identified to fill the research gap with a deductive approach. The two main sections of this project (Pre-processing and Artificial Neural Network model) need combined evaluation against the research objectives.

4.2 Artificial Neural Network

Quantitative data is considered to assess the performance of the model in terms of RMSE, accuracy, processing time. The quantitative data which can be collected from different participants feedback for the selected product.

4.3 Tools and Techniques

There are variety of tools to implement the proposed project. Most of the tools and packages has its own advantage and disadvantage. Python which is a general-purpose programming language has many built in packages to build a neural network model and it is used in this project. Tensor flow and Keras is used in this project which is the most popular open-source package for the implementation of Artificial neural network model.

4.4 Methodology

Relevant research in this area which makes use of Artificial Neural network from the literature survey includes:

1. Specific Features of the product like Speed, Display, etc., are used for ranking the most relevant product according to user preference [1]
2. Part of Speech Tagging to help more focus on exact features of product than just considering entire text as simple vector [2]
3. Lexicon based Data Pre-processing technique can be used for Auto labelling the data [3].
4. Named entity extraction technique used and the boundary analysis done with the extracted entities gives additional information for the classifiers. The non-dictionary words also being included in the analysis by replacing them with their synonyms is a good technique as usually this information is lost because the non-dictionary words usually discarded in the pre-processing stage itself, which results in loss of information[4].
5. Unlike most research work where the sentiment analysis of reviews focuses on the

classification of the text, the authors of this research work has taken the text and performed regression. The regression approach makes the overall results can be directly used for ranking the products in terms of product quality. Artificial Neural Networks is one of the most accurate algorithms useful in both the regression and classification problems and hence provide very good accuracy in this research work.[5]

4.5 Proposed Methodology

The proposed method will have three phases pre-processing phase and neural network model and post processing.

Phase 1: Text Pre-processing:

- i)Part of Speech Tagging to help more focus on exact features of product than just considering entire text as simple vector
- ii)Lexicon based Data Pre-processing technique can be used for Auto labelling the data
- iii)The non-dictionary words also being included in the analysis by replacing them with their synonyms.
- iv) TF IDF approach will be used to remove features without sentiment and to limit the input vector size fed into the ANN network.

Phase 2: Neural Network Model:

- 1. In this work, ANN will be used for analysing the text.
- 2.Regression will be done on the text and will generate a Product score based on the Review text which is given as vectorized input to the ANN network.

Phase 3: Post Processing:

- i) In addition to the product Score obtained by Regression output of ANN model, the ranking of Specific Features of the product like Speed, Display, etc., are extracted using Named entity extraction technique which gives additional information used for ranking the most relevant product according to user preference.
- ii) In addition to generic features of the product, user can give custom features based on which the product will be ranked according to the input query string given by the user. This will be hybrid ranking algorithm that will both take the Regression score of the ANN model for the product along with the text reviews as input and produce the most relevant result using the combined score.

4.6 User Interface

This project is implemented to interact with the user through a simple Command Line Interface (CLI) option. The focus of the research is to evaluate the model performance in terms of Accuracy, Score and R Squared value to ensure that the model can accurately predict the product score based on reviews. Hence the results of the analysis and model performance with the above metrics are displayed to the user at the end of the analysis.

5. ETHICS

The ethics is very important to be maintained in a Machine Learning project. In the aspect of ethics concerns related to this project, it is important to maintain the anonymity of the reviewers so that companies or individuals do not target the customers who have given reviews against their product. This is ensured by not including the research activity with any specific users. Ignore the user details like username or specific user ID in the pre-processing stage itself so that the entire data processing process is anonymous.

RISKS AND POTENTIAL ISSUES

A non-exhaustive list of some potential risks and their mitigations and consequences are listed below:

Potential Risk	Issues	Mitigations/Consequences
Not securing access to a specialist ML computer system	The data in the system with code, documents and data could be accessed by third party users	The computers used to process the data are protected by password so that anonymous users cannot get access to the data from the computer.
Client organization does not grant or removes access to copyrighted documents.	This research is done on one publicly available dataset and open-source software	The dataset ensured to be copy right free and allows freely without any restriction to perform any academic research activity. Also, all external libraries used are ensured to be opensource and license free so that there are no liabilities that could arise from the usage of those libraries

6. Timeline:

Task	W k1	W k2	W k3	W k4	W k5	W k6	W k7	W k8	W k9	Wk 10	Wk 11	Wk 12	Wk 13	Wk 14	Wk 15
MACHIE LEARNING															
General Research															
Obtain Software															
Obtain Technology															
Pre-processing algorithm															
Regression algorithm															
Model Evaluation & Metrics Collection															
Post Processing, Results Genration															
USER INTERFACE															
Research															
Design, Implement & Test User Interface															
DOCUMENTA TION & EVALUATION															
Write-Up															
First Draft Submitted															
Proof Reading & Contingency															
Submission Deadline															

7. References

1. Venkata Rajeev P, Smrithi Rekha V (2015). Recommending Products to Customers using opinion Mining of Online Product Reviews and Features. International Conference on Circuit, Power and Computing Technologies [ICCPCT](pp.1-5). Nagercoil, India: IEEE.
2. Upma Kumari, Arvind K Sharma, Dinesh Soni (2017). Sentiment Analysis of Smart Phone Product Review using SVM Classification Techniques. International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)(pp.1469-1474). Chennai, India: IEEE.
3. Aditya A. Kshirsagar, Prarthana A. Deshkar(2015). Review Analyzer analysis of Product reviews on WEKA Classifiers. IEEE Sponsored 2nd International Conference on Innovations in information, Embedded and Communication systems (ICIIECS)(pp.1-5). Coimbatore, India: IEEE.

4. Indhuja K, Reghu Raj P C(2014). Fuzzy Logic Based Sentiment Analysis of Product Review Documents. First International Conference on Computational Systems and Communications (ICCSC)(pp.18-22). Trivandrum, India: IEEE.
5. Aditya Parashar, Eshan Gupta (2017). ANN Based Ranking Algorithm for Products on e-Commerce Website. 3rd International Conference on Advances in Electrical, Electronics, Information, Communication and Bioinformatics (AEEICB)(pp.362-366). Chennai, India: IEEE.

Appendix B:

1. COMPLETED RESEARCH ETHICS CHECKLIST



RESEARCH ETHICS CHECKLIST FOR STUDENTS (SHUREC7)

This form is designed to help students and their supervisors to complete an ethical scrutiny of proposed research. The SHU [Research Ethics Policy](#) should be consulted before completing the form.

Answering the questions below will help you decide whether your proposed research requires ethical review by a Designated Research Ethics Working Group.

The final responsibility for ensuring that ethical research practices are followed rests with the supervisor for student research.

Note that students and staff are responsible for making suitable arrangements for keeping data secure and, if relevant for keeping the identity of participants anonymous. They are also responsible for following SHU guidelines about data encryption and research data management.

The form also enables the University and Faculty to keep a record confirming that research conducted has been subjected to ethical scrutiny.

For student projects, the form may be completed by the student and the supervisor and/or module leader (as applicable). In all cases, it should be counter-signed by the supervisor and/or module leader, and kept as a record showing that ethical scrutiny has occurred. Students should retain a copy for inclusion in their research projects, and staff should keep a copy in the student file.

Please note if it may be necessary to conduct a health and safety risk assessment for the proposed research. Further information can be obtained from the Faculty Safety Co-ordinator.

General Details

Name of student	Dhivya Nataraj
SHU email address	b9034198@my.shu.ac.uk
Course or qualification (student)	MSc Big Bata Analytics
Name of supervisor	Konstantinos domdouzis
email address	Aceskd1@exchange.shu.ac.uk
Title of proposed research	Sentiment Analysis of Online Product Review using Artificial Neural Network
Proposed start date	May
Proposed end date	September
Brief outline of research to include, rationale & aims (250-500 words).	This project aims to develop a system which will analyse the reviews for a set of products which makes easier for both the customer and stakeholder to find the quality and make decision accordingly.
Where data is collected from individuals, outline the nature of data, details of anonymisation, storage and disposal procedures if required (250-500 words).	Publicly available data anonymized by the provider. The data has mobile phone reviews entered by customers of Amazon E-commerce portal.

1. Health Related Research Involving the NHS or Social Care / Community Care or the Criminal Justice Service or with research participants unable to provide informed consent

Question	Yes/No
<p>1. Does the research involve?</p> <ul style="list-style-type: none"> • Patients recruited because of their past or present use of the NHS or Social Care • Relatives/carers of patients recruited because of their past or present use of the NHS or Social Care • Access to data, organs or other bodily material of past or present NHS patients • Foetal material and IVF involving NHS patients • The recently dead in NHS premises • Prisoners or others within the criminal justice system recruited for health-related research* • Police, court officials, prisoners or others within the criminal justice system* • Participants who are unable to provide informed consent due to their incapacity even if the project is not health related 	No
<p>2. Is this a research project as opposed to service evaluation or audit? <i>For NHS definitions please see the following website</i> http://www.hra.nhs.uk/documents/2013/09/defining-research.pdf</p>	No

If you have answered **YES** to questions **1 & 2** then you **must** seek the appropriate external approvals from the NHS, Social Care or the National Offender Management Service (NOMS) under their independent Research Governance schemes. Further information is provided below.

NHS <https://www.myresearchproject.org.uk/Signin.aspx>

* All prison and probation projects also need HM Prison and Probation Service (HMPPS) approval. Further guidance at: <https://www.myresearchproject.org.uk/help/hlphmpps.aspx>

NB FRECs provide Independent Scientific Review for NHS or SC research and initial scrutiny for ethics applications as required for university sponsorship of the research. Applicants can use the NHS proforma and submit this initially to their FREC.

2. Research with Human Participants

Question	Yes/No
Does the research involve human participants? This includes surveys, questionnaires, observing behaviour etc.	Yes
Question	Yes/No
1. <i>Note If YES, then please answer questions 2 to 10 If NO, please go to Section 3</i>	Yes
2. Will any of the participants be vulnerable? <i>Note: Vulnerable' people include children and young people, people with learning disabilities, people who may be limited by age or sickness, etc. See definition on website</i>	No
3. Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants or will the study involve invasive, intrusive or potentially harmful procedures of any kind?	No
4. Will tissue samples (including blood) be obtained from participants?	No
5. Is pain or more than mild discomfort likely to result from the study?	No
6. Will the study involve prolonged or repetitive testing?	No
7. Is there any reasonable and foreseeable risk of physical or emotional harm to any of the participants? <i>Note: Harm may be caused by distressing or intrusive interview questions, uncomfortable procedures involving the participant, invasion of privacy, topics relating to highly personal information, topics relating to illegal activity, etc.</i>	No
8. Will anyone be taking part without giving their informed consent?	No
9. Is it covert research? <i>Note: 'Covert research' refers to research that is conducted without the knowledge of participants.</i>	NA
10. Will the research output allow identification of any individual who has not given their express consent to be identified?	No

If you answered **YES only** to question **1**, the checklist should be saved and any course procedures for submission followed. If you have answered **YES** to any of the other questions

you are **required** to submit a SHUREC8A (or 8B) to the FREC. If you answered **YES** to question **8** and participants cannot provide informed consent due to their incapacity you must obtain the appropriate approvals from the NHS research governance system. Your supervisor will advise.

3. Research in Organisations


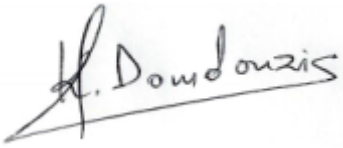
Question	Yes/No
1. Will the research involve working with/within an organisation (e.g. school, business, charity, museum, government department, international agency, etc.)?	No
2. If you answered YES to question 1, do you have granted access to conduct the research? <i>If YES, students please show evidence to your supervisor. PI should retain safely.</i>	-
3. If you answered NO to question 2, is it because: A. you have not yet asked B. you have asked and not yet received an answer C. you have asked and been refused access. <i>Note: You will only be able to start the research when you have been granted access.</i>	-

4. Research with Products and Artefacts

Question	Yes/No
1. Will the research involve working with copyrighted documents, films, broadcasts, photographs, artworks, designs, products, programmes, databases, networks, processes, existing datasets or secure data?	Yes
2. If you answered YES to question 1, are the materials you intend to use in the public domain? <i>Notes: 'In the public domain' does not mean the same thing as 'publicly accessible'.</i> <ul style="list-style-type: none"> Information which is 'in the public domain' is no longer protected by copyright (i.e. copyright has either expired or been waived) and can be used without permission. Information which is 'publicly accessible' (e.g. TV broadcasts, websites, artworks, newspapers) is available for anyone to consult/view. It is still protected by copyright even if there is no copyright notice. In UK law, copyright protection is automatic and does not require a copyright statement, although it is always good practice to provide one. It is necessary to check the terms and conditions of use to find out exactly how the material may be reused etc. <i>If you answered YES to question 1, be aware that you may need to consider other ethics codes. For example, when conducting Internet research, consult the code of the Association of Internet Researchers; for educational research, consult the Code of Ethics of the British Educational Research Association.</i>	No

3. If you answered NO to question 2, do you have explicit permission to use these materials as data? <i>If YES, please show evidence to your supervisor.</i>	Yes
4. If you answered NO to question 3, is it because: A. you have not yet asked permission B. you have asked and not yet received and answer C. you have asked and been refused access. <i>Note You will only be able to start the research when you have been granted permission to use the specified material.</i>	A/B/C

Adherence to SHU policy and procedures

Personal statement	
I can confirm that:	
<input type="checkbox"/> I have read the Sheffield Hallam University Research Ethics Policy and Procedures <input type="checkbox"/> I agree to abide by its principles.	
Student	
Name: Dhivya Nataraj	Date: 7- 7- 2021
Signature: 	
Supervisor or other person giving ethical sign-off	
I can confirm that completion of this form has not identified the need for ethical approval by the FREC or an NHS, Social Care or other external REC. The research will not commence until any approvals required under Sections 3 & 4 have been received.	
Name: KONSTANTINOS DOMDOUZIS	Date: 9-7-2021
Signature: 	
Additional Signature if required:	
Name:	Date:
Signature:	

Please ensure the following are included with this form if applicable, tick box to indicate:

Yes No N/A

Research proposal if prepared previously	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Any recruitment materials (e.g. posters, letters, etc.)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Participant information sheet	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Participant consent form	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Details of measures to be used (e.g. questionnaires, etc.)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Outline interview schedule / focus group schedule	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Debriefing materials	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Health and Safety Project Safety Plan for Procedures	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

APPENDIX B: Risk Log

ID	Description	Probability	Impact	Risk Category	Action	Risk Owner
1	Improper planning	Medium	Very high	Very high	Getting proper feedback from the supervisor	Development team
2	Inadequate knowledge	Low	Medium	High	Guidance from mentor and to possess better skills	Development team
3	The design not properly checked	Medium	High	High	Proper implementation of framework with the help of supervisor	Development team
4	Equipment Failure	Very Low	Very high	Very High	Appropriate backup	Development team
5	Failure to meet the deadline	Low	high	high	Regular segregation of work and proper checking	Development team

2. Publication Procedure form

**Sheffield
Hallam
University**

College of Business,
Technology and
Engineering

**Research Skills and
Dissertation Module
(55-706556).**


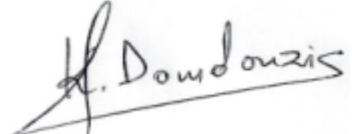
PUBLICATION PROCEDURE FORM

In this module, while you create your own research question or topic area, your supervisor makes a significant intellectual contribution to this work as the research progresses. Your supervisor will make the decision on whether your work merits publication based on the quality of the work you have produced. Your supervisor will co-author the paper for publication with you and your supervisor will both be listed as authors. You are required to sign the declaration below to confirm that you understand and will follow this procedure.

Declaration:

I Dhivya Nataraj confirm that I understand will comply with the Publication Procedure outlined in the Module Handbook and the Blackboard Site.

G5

Student:		Date: 7/7/2021
Supervisor:		Date: 10/7/2021

Appendix E

1. Python Code

Dataset Extraction code

```
tsv_file = pd.read_csv("shoes_reviews.tsv", sep='\t', error_bad_lines=False)
tsv_file.head()
```

```
tsv_file.to_excel("shoes.xlsx", index = False, header=True)
```

Text Analysis

```
comment_words = "
from nltk.corpus import stopwords
import string
stop_words =
set(stopwords.words('english')+list(string.punctuation)+["let", "more", "`", "''", "s"])
from wordcloud import WordCloud
```

```
for val in adidas['st_cleaned_message']:
```

```
    # typecaste each val to string
    val = str(val)
```

```
    # split the value
    tokens = val.split()
```

```
    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()
```

```
    comment_words += " ".join(tokens)+" "
wordcloud = WordCloud(width = 800, height = 800,
                      background_color ='white',
                      stopwords = stop_words,
                      min_font_size = 10).generate(comment_words)
```

```
# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
```

```
plt.show()
```

1.1 TF-IDF for Machine learning algorithm

```
def load_data():
    ## importing the necessary library
    import pandas as pd
    import numpy as np
    import datetime
    import seaborn as sns
    import matplotlib.pyplot as plt
    %matplotlib notebook
    import warnings
    warnings.filterwarnings("ignore")

    ### reading the data as dataframe
    df = pd.read_excel('shoes.xlsx',sheet_name="Sheet2")

    ### deleting the unwanted column
    del(df["marketplace"])
    del(df["customer_id"])
    del(df["review_id"])
    del(df["product_parent"])
    del(df["vine"])
    del(df["review_headline"])
    del(df["total_votes"])
    del(df["product_category"])

    ##### create a brand subset
    df['product_title']=df['product_title'].apply(lambda x: x.lower())

    conditions =[(df['product_title'].str.contains('adidas')),
                  (df['product_title'].str.contains('crocs')),
                  (df['product_title'].str.contains('skechers'))]
    values=['adidas','crocs','skechers']
    df['Brand']=np.select(conditions,values)

    ##### labeling the data using star rating
    df["verified_purchase"]=df.verified_purchase.map({'Y':1,'N':0})
    df["positivity"] = df["star_rating"].apply(lambda x: 2 if x>3 else(0 if x==3 else 1))

    ### #Text Cleaning
    # 1.1 Define preprocess function
    df["review_body"] = df["review_body"].astype("str")
    import string
    import nltk
    #nltk.download('words')
    words = set(nltk.corpus.words.words())
    stopwords = nltk.corpus.stopwords.words('english')
    new_stopwords = ["i've","i'm",'on','ie','these for','im']
```



```

stopwords.extend(new_stopwords)
import re
wn=nlTK.WordNetLemmatizer()

def removing_punc(ele):
    # Convert the text into lowercase
    ele = ele.lower()
    #punctuation
    ele = re.sub('[%s]' % re.escape(string.punctuation), "", ele)
    # number
    ele = re.sub(r'[0-9]', "", ele)
    #new line
    ele = re.sub('\n', "", ele)
    #white space
    ele= re.sub("^\s+", "", ele)
    return ele
df["review_body"]=df["review_body"].apply(lambda x: removing_punc(x))

def tokenize(txt):
    """tokenize each word by using split() function"""
    tokens=re.split("\W+", txt)
    return tokens
df['tokenized_message']=df['review_body'].apply(lambda x: tokenize(x))

def clean_word(txt_tokenized):
    """removed the stopword and remove the numbers and get the base word using
    lemmatize function"""
    new_word = [word for word in txt_tokenized if word not in stopwords]
    new_word = [word for word in new_word if word.isalpha()]
    new_word = [word for word in new_word if word in words]
    new_word = [wn.lemmatize(word) for word in new_word]
    return " ".join(new_word)
df['st_cleaned_message']=df['tokenized_message'].apply(lambda x:clean_word(x))
return df

def switch_fun(choice, t):
    """ based on brand of shoes the corresponding function calls"""

    from sklearn.svm import SVC
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.linear_model import SGDClassifier

    if choice == 1 and t == 1:
        df=load_data()
        x1,y1=adidas(df)

```

```

    rf = RandomForestClassifier(max_depth=10, max_features='sqrt', n_estimators=15)
    rf_cval,rf_tra,rf_te = machine_learning_model(x1,y1,rf)
elif choice == 1 and t == 2:
    df=load_data()
    x1,y1=adidas(df)
    dt=DecisionTreeClassifier(max_depth=20)
    dt_cval,dt_tra,dt_te = machine_learning_model(x1,y1,dt)
elif choice == 1 and t == 3:
    df=load_data()
    x1,y1=adidas(df)
    sgd=SGDClassifier(loss="log", penalty="l2")
    sgd_cval,sgd_tra,sgd_te = machine_learning_model(x1,y1,sgd)
elif choice == 1 and t == 4:
    df=load_data()
    x1,y1=adidas(df)
    nb=MultinomialNB(alpha=0.2)
    nb_cval,nb_tra,nb_te = machine_learning_model(x1,y1,nb)

elif choice == 2 and t == 1:
    df=load_data()
    x1,y1=skechers(df)
    rf = RandomForestClassifier(max_depth=10, max_features='sqrt', n_estimators=15)
    rf_cval_s,rf_tra_s,rf_te_s = machine_learning_model(x1,y1,rf)
elif choice == 2 and t == 2:
    df=load_data()
    x1,y1=skechers(df)
    dt=DecisionTreeClassifier(max_depth=20)
    dt_cval_s,dt_tra_s,dt_te_s = machine_learning_model(x1,y1,dt)
elif choice == 2 and t == 3:
    df=load_data()
    x1,y1=skechers(df)
    sgd=SGDClassifier(loss="log", penalty="l2")
    sgd_cval_s,sgd_tra_s,sgd_te_s = machine_learning_model(x1,y1,sgd)
elif choice == 2 and t == 4:
    df=load_data()
    x1,y1=skechers(df)
    nb=MultinomialNB(alpha=0.2)
    nb_cval_s,nb_tra_s,nb_te_s = machine_learning_model(x1,y1,nb)

elif choice == 3 and t == 1:
    df=load_data()
    x1,y1=crocs(df)
    rf = RandomForestClassifier(max_depth=10, max_features='sqrt', n_estimators=15)
    rf_cval_c,rf_tra_c,rf_te_c = machine_learning_model(x1,y1,rf)
elif choice == 3 and t == 2:
    df=load_data()
    x1,y1=crocs(df)
    dt=DecisionTreeClassifier(max_depth=20)
    dt_cval_c,dt_tra_c,dt_te_c = machine_learning_model(x1,y1,dt)
elif choice == 3 and t == 3:

```

```

        df=load_data()
        x1,y1=crocs(df)
        sgd=SGDClassifier(loss="log", penalty="l2")
        sgd_cval_c,sgd_tra_c,sgd_te_c = machine_learning_model(x1,y1,sgd)
    elif choice == 3 and t == 4:
        df=load_data()
        x1,y1=crocs(df)
        nb=MultinomialNB(alpha=0.2)
        nb_cval_c,nb_tra_c,nb_te_c = machine_learning_model(x1,y1,nb)

    else:
        unknown_action()

def unknown_action():
    "if the user input is wrong"
    print('invalid entry')
    print('Please enter 1 or 2 or 3 for brand reviews')
    print('Please enter 1 or 2 or 3 or 4 for machine learning model')
    return 0

def adidas(dataset):
    ### extracting only the adidas shoe brand reviews from the dataset
    adidas = dataset[dataset["Brand"]=="adidas"].sort_values(by=["review_date"],
ascending=False)

    ### Removing unwanted column
    del(adidas['product_title'])
    del(adidas['review_body'])
    del(adidas['Brand'])
    del(adidas["review_date"])
    del(adidas['star_rating'])
    del(adidas['tokenized_message'])

    from sklearn.feature_extraction.text import CountVectorizer
    #instantiate CountVectorizer()
    #no of feature is 7391
    cv=CountVectorizer(max_features=6000)

    # this steps generates word counts for the words
    word_count_vector=cv.fit_transform(adidas['st_cleaned_message'])
    x=word_count_vector.toarray()
    y=adidas['positivity']
    return x,y

def skechers(dataset):
    ### extracting only the adidas shoe brand reviews from the dataset

```

```
skechers = dataset[dataset["Brand"]=="skechers"].sort_values(by=["review_date"],
ascending=False)
```

```
### Removing unwanted column
del(skechers['product_title'])
del(skechers['review_body'])
del(skechers['Brand'])
del(skechers["review_date"])
del(skechers['star_rating'])
del(skechers['tokenized_message'])
```

```
from sklearn.feature_extraction.text import CountVectorizer
#instantiate CountVectorizer()
# no of features is 6000
cv=CountVectorizer(max_features=5000)
```

```
# this steps generates word counts for the words
word_count_vector=cv.fit_transform(skechers['st_cleaned_message'])
x=word_count_vector.toarray()
y=skechers['positivity']
return x,y
```

```
def crocs(dataset):
```

```
### extracting only the adidas shoe brand reviews from the dataset
crocs = dataset[dataset["Brand"]=="crocs"].sort_values(by=["review_date"],
ascending=False)
```

```
### Removing unwanted column
del(crocs['product_title'])
del(crocs['review_body'])
del(crocs['Brand'])
del(crocs["review_date"])
del(crocs['star_rating'])
del(crocs['tokenized_message'])
```

```
from sklearn.feature_extraction.text import CountVectorizer
#instantiate CountVectorizer()
# no of features is 6000
cv=CountVectorizer(max_features=5000)
```

```
# this steps generates word counts for the words
word_count_vector=cv.fit_transform(crocs['st_cleaned_message'])
x=word_count_vector.toarray()
y=crocs['positivity']
return x,y
```

```

def machine_learning_model(x1,y1,mla):

    # split the dataset as train and test for evaluation
    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x1, y1, test_size=0.20, random_state = 42)
    print(x_train.shape,y_train.shape)

    from collections import Counter
    print('Before balancing the data')
    print(sorted(Counter(y_train).items()))
    from imblearn.combine import SMOTETomek
    smote_tomek = SMOTETomek(random_state=0)
    x_resampled, y_resampled = smote_tomek.fit_resample(x_train, y_train)
    print('After balancing the data')
    print('class 0 --> neutral\n class 1 --> negative \n class 2 --> positive')
    print(sorted(Counter(y_resampled).items()))

    # RANDOM FOREST CLASSIFIER
    from sklearn.pipeline import Pipeline
    from sklearn.feature_extraction.text import TfidfTransformer
    from sklearn.model_selection import StratifiedKFold,cross_validate,cross_val_score
    from sklearn.metrics import classification_report

    text_clf1 = Pipeline([ ('tfidf', TfidfTransformer()),('clf', mla)])
    print(text_clf1)
    kfold = StratifiedKFold(n_splits=5)
    scores = cross_val_score(text_clf1, x_resampled, y_resampled, cv=kfold,
scoring='accuracy')
    validation_score = scores.mean()
    print('cross validation:',validation_score)

    ### fit the model
    from sklearn.metrics import accuracy_score
    text_clf1.fit(x_resampled,y_resampled)
    ### prediction
    y_pred_train = text_clf1.predict(x_resampled)
    y_pred_test = text_clf1.predict(x_test)
    train_score = accuracy_score(y_pred_train,y_resampled)
    test_score = accuracy_score(y_pred_test,y_test)
    print('train score:',train_score)
    print('test score:',test_score)
    print('*****')
    print(' classification report:\n',classification_report(y_pred_test,y_test,digits=3))

    return validation_score,train_score,test_score

print('Choose brand based dataset to know about Machine Learning model')

```

```

print('choose the below options')
print("\n 1.adiads Brand reviews \n 2.skechers Brand reviews \n 3.crocs Brand Reviews')
try:
    r = int(input("Enter number of your choice : "))

except TypeError:
    print("TypeError")
except:
    print('invalid entry')

print("\n*****
*****')
print("To run the particular classifier model')
print('choose the below options')
print("\n 1.Random Forest classifier \
\n 2.Decision Tree classifier \
\n 3.SGD classifier\
\n 4.Naive Bayes')
try:
    m = int(input("Enter number of your choice : "))

except TypeError:
    print("TypeError")
except:
    print('invalid entry')
switch_fun(r,m)

```

1.2 Word Embedding for machine learning

```

def load_data():
    ## importing the necessary library
    import pandas as pd
    import numpy as np
    import datetime
    import seaborn as sns
    import matplotlib.pyplot as plt
    %matplotlib notebook
    import warnings
    warnings.filterwarnings("ignore")

    ### reading the data as dataframe
    df = pd.read_excel('shoes.xlsx',sheet_name="Sheet2")

    ### deleting the unwanted column
    del(df["marketplace"])
    del(df["customer_id"])
    del(df["review_id"])
    del(df["product_parent"])
    del(df["vine"])
    del(df["review_headline"])
    del(df["total_votes"])

```

```

del(df["product_category"])

#### create a brand subset
df['product_title']=df['product_title'].apply(lambda x: x.lower())

conditions =[(df['product_title'].str.contains('adidas')),
              (df['product_title'].str.contains('crocs')),
              (df['product_title'].str.contains('skechers'))]
values=['adidas','crocs','skechers']
df['Brand']=np.select(conditions,values)

#### labeling the data using star rating
df["verified_purchase"]=df.verified_purchase.map({'Y':1,'N':0})
df["positivity"] = df["star_rating"].apply(lambda x: 2 if x>3 else(0 if x==3 else 1))

### #Text Cleaning
# 1.1 Define preprocess function
df["review_body"] = df["review_body"].astype("str")
import string
import nltk
nltk.download('words')
words = set(nltk.corpus.words.words())
stopwords = nltk.corpus.stopwords.words('english')
new_stopwords = ['i've','i'm','on','ie','thesefor','im']
stopwords.extend(new_stopwords)
import re
wn=nltk.WordNetLemmatizer()

def removing_punc(ele):
    # Convert the text into lowercase
    ele = ele.lower()
    #punctuation
    ele = re.sub('[%s]' % re.escape(string.punctuation), "", ele)
    # number
    ele = re.sub(r'[0-9]', "", ele)
    #new line
    ele = re.sub('\n', "", ele)
    #white space
    ele= re.sub("^\s+", "", ele)
    return ele
df["review_body"]=df["review_body"].apply(lambda x: removing_punc(x))

def tokenize(txt):
    """tokenize each word by using split() function"""
    tokens=re.split('\W+', txt)
    return tokens
df['tokenized_message']=df['review_body'].apply(lambda x: tokenize(x))

```

```

def clean_word(txt_tokenized):
    """removed the stopword and remove the numbers and get the base word using
    lemmatize function"""
    new_word = [word for word in txt_tokenized if word not in stopwords]
    new_word = [word for word in new_word if word.isalpha()]
    new_word = [word for word in new_word if word in words]
    new_word = [wn.lemmatize(word) for word in new_word]
    return " ".join(new_word)
df['st_cleaned_message']=df['tokenized_message'].apply(lambda x:clean_word(x))
return df

```

```

def switch_fun(choice, t):
    """ based on brand of shoes the corresponding function calls"""
    if choice == 1 and t == 1:
        df=load_data()
        x1,y1=adidas(df)
        rf_trs,rf_ts,rf_vals = random_forest(x1,y1)
    elif choice == 1 and t == 2:
        df=load_data()
        x1,y1=adidas(df)
        lg_trs,lg_ts,lg_vals = logistic_regression(x1,y1)
    elif choice == 1 and t == 3:
        df=load_data()
        x1,y1=adidas(df)
        dt_trs,dt_ts,dt_vals = decision_tree(x1,y1)
    elif choice == 1 and t == 4:
        df=load_data()
        x1,y1=adidas(df)
        nb_trs,nb_ts,nb_vals = nb(x1,y1)
    elif choice == 2 and t == 1:
        df=load_data()
        x1,y1=skechers(df)
        srf_trs,srf_ts,srf_vals = random_forest(x1,y1)
    elif choice == 2 and t == 2:
        df=load_data()
        x1,y1=skechers(df)
        slg_trs,slg_ts,slg_vals = logistic_regression(x1,y1)
    elif choice == 2 and t == 3:
        df=load_data()
        x1,y1=skechers(df)
        sdt_trs,sdt_ts,sdt_vals = decision_tree(x1,y1)
    elif choice == 2 and t == 4:
        df=load_data()
        x1,y1=skechers(df)
        snb_trs,snb_ts,snb_vals = nb(x1,y1)
    elif choice == 3 and t == 1:
        df=load_data()
        x1,y1=crocs(df)

```



```

        crf_trs,crf_ts,crf_vals = random_forest(x1,y1)
    elif choice == 3 and t == 2:
        df=load_data()
        x1,y1=crocs(df)
        clg_trs,clg_ts,clg_vals = logistic_regression(x1,y1)
    elif choice == 3 and t == 3:
        df=load_data()
        x1,y1=crocs(df)
        cdt_trs,cdt_ts,cdt_vals = decision_tree(x1,y1)
    elif choice == 3 and t == 4:
        df=load_data()
        x1,y1=crocs(df)
        cnb_trs,cnb_ts,cnb_vals = nb(x1,y1)

    else:
        unknown_action()

def unknown_action():
    "if the user input is wrong"
    print('invalid entry')
    print('Please enter 1 or 2 or 3 for brand reviews')
    print('Please enter 1 or 2 or 3 or 4 for machine learning model')
    return 0

def adidas(dataset):
    """ extracting only the adidas shoe brand reviews from the dataset
    adidas = dataset[dataset["Brand"]=="adidas"].sort_values(by=["review_date"],
ascending=False)

    """ Removing unwanted column
    del(adidas['product_title'])
    del(adidas['review_body'])
    del(adidas['Brand'])
    del(adidas["review_date"])
    del(adidas['star_rating'])
    del(adidas['tokenized_message'])

    ## define size of vocabulary and max length of the sentence in the review dataset
    vocab_size = 4156
    max_length = 291

    ## name the y variable as sentiment
    import numpy as np
    sentiment = np.array(adidas['positivity'])
    sentiment

    from tensorflow.keras.preprocessing.text import one_hot
    from tensorflow.keras.preprocessing.sequence import pad_sequences

```

```

## one hot encoding for the words
encoded_reviews = [one_hot(d, vocab_size) for d in adidas['st_cleaned_message']]

## padding
padded_reviews = pad_sequences(encoded_reviews, maxlen=max_length, padding='post')

## define X and y variable
x=padded_reviews
y=sentiment
return x,y

def skechers(dataset):
    ### extracting only the adidas shoe brand reviews from the dataset
    skechers = dataset[dataset["Brand"]=="skechers"].sort_values(by=["review_date"],
ascending=False)

    ### Removing unwanted column
    del(skechers['product_title'])
    del(skechers['review_body'])
    del(skechers['Brand'])
    del(skechers["review_date"])
    del(skechers['star_rating'])
    del(skechers['tokenized_message'])

    ## define size of vocabulary and max length of the sentence in the review dataset
    vocab_size = 4800
    max_length = 330

    ## name the y variable as sentiment
    import numpy as np
    sentiment = np.array(skechers['positivity'])
    sentiment

    from tensorflow.keras.preprocessing.text import one_hot
    from tensorflow.keras.preprocessing.sequence import pad_sequences
    ## one hot encoding for the words
    encoded_reviews = [one_hot(d, vocab_size) for d in skechers['st_cleaned_message']]

    ## padding
    padded_reviews = pad_sequences(encoded_reviews, maxlen=max_length, padding='post')

    ## define X and y variable
    x=padded_reviews
    y=sentiment
    return x,y

def crocs(dataset):

```

```

### extracting only the adidas shoe brand reviews from the dataset
crocs = dataset[dataset["Brand"]=="crocs"].sort_values(by=["review_date"],
ascending=False)

### Removing unwanted column
del(crocs['product_title'])
del(crocs['review_body'])
del(crocs['Brand'])
del(crocs["review_date"])
del(crocs['star_rating'])
del(crocs['tokenized_message'])

## define size of vocabulary and max length of the sentence in the review dataset
vocab_size = 5032
max_length = 235

## name the y variable as sentiment
import numpy as np
sentiment = np.array(crocs['positivity'])
sentiment

from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.preprocessing.sequence import pad_sequences
## one hot encoding for the words
encoded_reviews = [one_hot(d, vocab_size) for d in crocs['st_cleaned_message']]

## padding
padded_reviews = pad_sequences(encoded_reviews, maxlen=max_length, padding='post')

## define X and y variable
x=padded_reviews
y=sentiment
return x,y

def random_forest(x1,y1):

# split the dataset as train and test for evaluation
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x1, y1, test_size=0.20, random_state = 0)

from collections import Counter
print('Before balancing the data')
print(sorted(Counter(y_train).items()))
from imblearn.combine import SMOTETomek
smote_tomek = SMOTETomek(random_state=0)
x_resampled, y_resampled = smote_tomek.fit_resample(x_train, y_train)
print('After balancing the data')
print('class 0 --> neutral\n class 1 --> negative \n class 2 --> positive')

```

```

print(sorted(Counter(y_resampled).items()))

# RANDOM FOREST CLASSIFIER
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score,KFold,cross_validate,GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

random_grid = {'max_depth': [10, 20],
               'max_features': ['auto', 'sqrt'],
               'n_estimators': [ 15, 10]}

randomforest = GridSearchCV(RandomForestClassifier(),random_grid,cv=5)
randomforest_fit = randomforest.fit(x_resampled,y_resampled)

##### best estimator
best_rf_model = randomforest_fit.best_estimator_
print('the best hyperparameters:',best_rf_model)

##### cross validation score
kfold = StratifiedKFold(n_splits=5)
scores = cross_val_score(best_rf_model, x_resampled, y_resampled, cv=kfold,
scoring='accuracy')
rf_validation_score = scores.mean()

### fit the model
from sklearn.metrics import accuracy_score
best_rf_model.fit(x_resampled,y_resampled)

### predict the model
y_pred_train = best_rf_model.predict(x_resampled)
y_pred_test = best_rf_model.predict(x_test)
rf_train_score = accuracy_score(y_pred_train,y_resampled)
rf_test_score = accuracy_score(y_pred_test,y_test)

print('train accuracy:',rf_train_score)
print('test accuracy:',rf_test_score)
print('validation score:',rf_validation_score)
print('*****')
print(' classification report:\n',classification_report(y_pred_test,y_test,digits=3))
return rf_train_score,rf_test_score,rf_validation_score

def logistic_regression(x1,y1):
    # split the dataset as train and test for evaluation
    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x1, y1, test_size=0.20, random_state = 0)

```

```

from collections import Counter
print('Before balancing the data')
print(sorted(Counter(y_train).items()))
from imblearn.combine import SMOTETomek
smote_tomek = SMOTETomek(random_state=0)
x_resampled, y_resampled = smote_tomek.fit_resample(x_train, y_train)
print('After balancing the data')
print('class 0 --> neutral\n class 1 --> negative \n class 2 --> positive')
print(sorted(Counter(y_resampled).items()))

### LOGISTIC REGRESSION CLASSIFIER
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score, KFold, cross_validate, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

##### hyperparameter tuning
param_grid = {'C': [0.1, 0.2, 0.3, 0.4, 0.5]}
logistic = GridSearchCV(LogisticRegression(), param_grid, cv=5)
logistic_fit = logistic.fit(x_resampled, y_resampled)

##### best estimator
best_lg_model = logistic_fit.best_estimator_
print('the best hyperparameters:', best_lg_model)

##### cross validation score
kfold = StratifiedKFold(n_splits=5)
scores = cross_val_score(best_lg_model, x_resampled, y_resampled, cv=kfold,
scoring='accuracy')
lg_validation_score = scores.mean()

### fit the model
from sklearn.metrics import accuracy_score
best_lg_model.fit(x_resampled, y_resampled)

### predict the model
y_pred_train = best_lg_model.predict(x_resampled)
y_pred_test = best_lg_model.predict(x_test)
lg_train_score = accuracy_score(y_pred_train, y_resampled)
lg_test_score = accuracy_score(y_pred_test, y_test)

print('train accuracy:', lg_train_score)
print('test accuracy:', lg_test_score)
print('validation score:', lg_validation_score)
print('*****')
print(' classification report:\n', classification_report(y_pred_test, y_test, digits=3))
return lg_train_score, lg_test_score, lg_validation_score

def decision_tree(x1, y1):

```

```

# split the dataset as train and test for evaluation
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x1, y1, test_size=0.20, random_state = 0)

from collections import Counter
print('Before balancing the data')
print(sorted(Counter(y_train).items()))
from imblearn.combine import SMOTETomek
smote_tomek = SMOTETomek(random_state=0)
x_resampled, y_resampled = smote_tomek.fit_resample(x_train, y_train)
print('After balancing the data')
print('class 0 --> neutral\n class 1 --> negative \n class 2 --> positive')
print(sorted(Counter(y_resampled).items()))

## DECISION TREE CLASSIFIER
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import
cross_val_score,KFold,cross_val_predict,cross_validate,GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

##### hyperparameter tuning
tree_para = {'criterion':['gini','entropy'],'max_depth':[1,2,3,4,5,20,30,40,50,70,90,120,150]}
decision_tree = GridSearchCV(DecisionTreeClassifier(), tree_para,scoring='accuracy')
decision_tree_fit =decision_tree.fit(x_resampled,y_resampled)

##### best estimator
best_dt_model = decision_tree_fit.best_estimator_
print('the best hyperparameters:',best_dt_model)

#### cross validation score
kfold = StratifiedKFold(n_splits=5)
scores = cross_val_score(best_dt_model, x_resampled, y_resampled, cv=kfold,
scoring='accuracy')
dt_validation_score = scores.mean()

### fit the model
from sklearn.metrics import accuracy_score
best_dt_model.fit(x_resampled,y_resampled)

### predict the model
y_pred_train = best_dt_model.predict(x_resampled)
y_pred_test = best_dt_model.predict(x_test)
dt_train_score = accuracy_score(y_pred_train,y_resampled)
dt_test_score = accuracy_score(y_pred_test,y_test)

print('train accuracy:',dt_train_score)
print('test accuracy:',dt_test_score)
print('validation score:',dt_validation_score)

```

```

print('*****')
print(' classification report:\n',classification_report(y_pred_test,y_test,digits=3))
return dt_train_score,dt_test_score,dt_validation_score

def nb(x1,y1):
    # split the dataset as train and test for evalution
    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x1, y1, test_size=0.20, random_state = 0)

    from collections import Counter
    print('Before balancing the data')
    print(sorted(Counter(y_train).items()))
    from imblearn.combine import SMOTETomek
    smote_tomek = SMOTETomek(random_state=0)
    x_resampled, y_resampled = smote_tomek.fit_resample(x_train, y_train)
    print('After balancing the data')
    print('class 0 --> neutral\n class 1 --> negative \n class 2 --> positive')
    print(sorted(Counter(y_resampled).items()))

    ###NAIVE BAYES
    from sklearn.model_selection import StratifiedKFold
    from sklearn.model_selection import cross_val_score,KFold,cross_validate,GridSearchCV
    from sklearn.naive_bayes import GaussianNB,MultinomialNB
    from sklearn.metrics import classification_report
    import numpy as np

    params_NB = {'alpha': (0.1,0.2)}

    nb = GridSearchCV(MultinomialNB(),params_NB,scoring='accuracy')
    nb_fit = nb.fit(x_resampled,y_resampled)

    ##### best estimator
    best_nb_model = nb_fit.best_estimator_
    print('the best hyperparameters:',best_nb_model)

    ### cross validation score
    kfold = StratifiedKFold(n_splits=5)
    scores = cross_val_score(best_nb_model, x_resampled, y_resampled, cv=kfold,
scoring='accuracy')
    nb_validation_score = scores.mean()

    ### fit the model
    from sklearn.metrics import accuracy_score
    best_nb_model.fit(x_resampled,y_resampled)

    ### predict the model
    y_pred_train = best_nb_model.predict(x_resampled)
    y_pred_test = best_nb_model.predict(x_test)
    nb_train_score = accuracy_score(y_pred_train,y_resampled)
    nb_test_score = accuracy_score(y_pred_test,y_test)

```

```

print('train accuracy:',nb_train_score)
print('test accuracy:',nb_test_score)
print('validation score:',nb_validation_score)
print('*****')
print(' classification report:\n',classification_report(y_pred_test,y_test,digits=3))
return nb_train_score,nb_test_score,nb_validation_score

print('Choose brand based dataset to know about Machine Learning model')
print('choose the below options')
print('\n 1.adiads Brand reviews \n 2.skechers Brand reviews \n 3.crocs Brand Reviews')
try:
    r = int(input("Enter number of your choice : "))

except TypeError:
    print("TypeError")
except:
    print('invalid entry')

print("\n*****")
print("To run the particular classifier model")
print('choose the below options')
print("\n 1.Random Forest classifier \
\n 2.Logistic Regression classifier \
\n 3.Decision tree classifier \
\n 4.Naive Bayes')
try:
    m = int(input("Enter number of your choice : "))

except TypeError:
    print("TypeError")
except:
    print('invalid entry')
switch_fun(r,m)

```

1.3 Word Embedding for ANN

```

def load_preprocessing():
    """preprocessing the data like removing unwanted column,clean the text
    stop word, non-english word removed, tokenized, lemmatized"""
    ## importing the necessary library
    import pandas as pd
    import datetime
    import seaborn as sns
    import numpy as np
    import warnings
    warnings.filterwarnings("ignore")

```



```

### reading the data as dataframe
df = pd.read_excel('shoes.xlsx',sheet_name="Sheet2")

### deleting the unwanted column
del(df["marketplace"])
del(df["customer_id"])
del(df["review_id"])
del(df["product_parent"])
del(df["vine"])
del(df["review_headline"])
del(df["total_votes"])
del(df["product_category"])
del(df["product_id"])
del(df["helpful_votes"])
del(df["verified_purchase"])

#### create a brand subset
df['product_title']=df['product_title'].apply(lambda x: x.lower())

conditions =[(df['product_title'].str.contains('adidas')),
              (df['product_title'].str.contains('crocs')),
              (df['product_title'].str.contains('skechers'))]
values=['adidas','crocs','skechers']
df['Brand']=np.select(conditions,values)

#### labeling the data using star rating
df["positivity"] = df["star_rating"].apply(lambda x: 2 if x>3 else(0 if x==3 else 1))

### #Text Cleaning
# 1.1 Define preprocess function
df["review_body"] = df["review_body"].astype("str")
import string
import nltk
nltk.download('words')
words = set(nltk.corpus.words.words())
stopwords = nltk.corpus.stopwords.words('english')
new_stopwords = ["i've","i'm",'on','ie','thesefor','im']
stopwords.extend(new_stopwords)
import re
wn=nltk.WordNetLemmatizer()

def removing_punc(ele):
    # Convert the text into lowercase
    ele = ele.lower()
    #punctuation
    ele = re.sub('[%s]' % re.escape(string.punctuation), "", ele)
    # number
    ele = re.sub(r'[0-9]', "", ele)

```

```

#new line
ele = re.sub('\n', "", ele)
#white space
ele= re.sub("^\s+", "", ele)
return ele
df["review_body"]=df["review_body"].apply(lambda x: removing_punc(x))

def tokenize(txt):
    """tokenize each word by using split() function"""
    tokens=re.split('\W+', txt)
    return tokens
df['tokenized_message']=df['review_body'].apply(lambda x: tokenize(x))

def clean_word(txt_tokenized):
    """removed the stopword,non-english words and remove the numbers and get the base
word using lemmatize function"""
    new_word = [word for word in txt_tokenized if word not in stopwords]
    new_word = [word for word in new_word if word.isalpha()]
    new_word = [word for word in new_word if word in words]
    new_word = [wn.lemmatize(word) for word in new_word]
    return ' '.join(new_word)

df['st_cleaned_message']=df['tokenized_message'].apply(lambda x:clean_word(x))
return df

def switch_fun(choice):
    """ based on brand of shoes the corresponding function calls"""
    if choice == 1:
        df=load_preprocessing()
        a,b,c=adidas(df)
        model_build_fit(a,b,c)
    elif choice == 2:
        df=load_preprocessing()
        a,b,c=skechers(df)
        model_build_fit(a,b,c)

    elif choice == 3:
        df=load_preprocessing()
        a,b,c=crocs(df)
        model_build_fit(a,b,c)

    else:
        unknown_action()

def unknown_action():
    """if the user input is wrong"""
    print('invalid entry')
    print('enter 1 or 2 or 3')
    return 0

```

```

def adidas(df):
    """get the adidas dataset from the whole dataset"""
    ### extracting only the adidas shoe brand reviews from the dataset
    adidas = df[df["Brand"]=="adidas"].sort_values(by=["review_date"], ascending=False)
    ### Removing unwanted column
    del(adidas['product_title'])
    del(adidas['review_body'])
    del(adidas['Brand'])
    del(adidas["review_date"])
    del(adidas['star_rating'])
    del(adidas['tokenized_message'])
    vocab_size = 4156
    max_length = 291
    return adidas,vocab_size,max_length

def skechers(df):
    """get the skechers dataset from the whole dataset"""
    ### extracting only the skechers shoe brand reviews from the dataset
    skechers = df[df["Brand"]=="skechers"].sort_values(by=["review_date"],
ascending=False)
    ### Removing unwanted column
    del(skechers['product_title'])
    del(skechers['review_body'])
    del(skechers['Brand'])
    del(skechers["review_date"])
    del(skechers['star_rating'])
    del(skechers['tokenized_message'])
    vocab_size = 4800
    max_length = 330
    return skechers,vocab_size,max_length

def crocs(df):
    """get the crocs dataset from the whole dataset"""
    ### extracting only the crocs shoe brand reviews from the dataset
    crocs = df[df["Brand"]=="crocs"].sort_values(by=["review_date"], ascending=False)
    ### Removing unwanted column
    del(crocs['product_title'])
    del(crocs['review_body'])
    del(crocs['review_date'])
    del(crocs['Brand'])
    del(crocs['star_rating'])
    del(crocs['tokenized_message'])
    vocab_size = 5032
    max_length = 235
    return crocs,vocab_size,max_length

def model_build_fit(data1,vocab_size,max_length):
    """ Buliding ANN model with word embedding layer"""

```

```

## importing necessary library to build word embedding in the Artificial neural network
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Embedding
import keras
from keras.callbacks import EarlyStopping
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

## name the y variable as sentiment
sentiment = np.array(data1['positivity'])
sentiment

## one hot encoding for the words
encoded_reviews = [one_hot(d, vocab_size) for d in data1['st_cleaned_message']]

## padding
padded_reviews = pad_sequences(encoded_reviews, maxlen=max_length, padding='post')

## define X and y variable
x=padded_reviews
y=sentiment

from keras.utils import np_utils
## convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(y)

## split the dataset as train and test for evaluation
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, dummy_y, test_size=0.20, random_state
= 42)

## import necessary library to add word embedding layer
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.embeddings import Embedding
embedded_vector_feature = 20

# define the model
model = Sequential()
model.add(Embedding(vocab_size, embedded_vector_feature, input_length=max_length))
model.add(Flatten())

```

```

model.add(Dense(units = 16, kernel_initializer='he_uniform', input_dim = (x.shape[1])))
model.add(Dense(3, activation='softmax'))

# compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
# summarize the model
model.summary()

# early stopping callback
# This callback will stop the training when there is no improvement in
# the validation loss for 5 consecutive epochs.
es = keras.callbacks.EarlyStopping(monitor='val_loss',
                                   mode='min',
                                   patience=5,
                                   restore_best_weights=True)

from sklearn.utils import class_weight
class_weights = list(class_weight.compute_class_weight('balanced',
                                                       np.unique(data1['positivity']),
                                                       data1['positivity']))

weights={ }
for index, weight in enumerate(class_weights):
    weights[index]=weight

# model fit
history = model.fit(x_train,
                   y_train,
                   callbacks=[es],
                   class_weight=weights,
                   epochs=8000000,
                   batch_size=500,
                   shuffle=True,
                   validation_split=0.20,
                   verbose=1)

history_dict = history.history
# learning curve
# accuracy
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']

# loss
loss = history_dict['loss']
val_loss = history_dict['val_loss']

# range of X (no. of epochs)
epochs = range(1, len(acc) + 1)

# plot

```

```

# "r" is for "solid red line"
plt.plot(epochs, acc, 'r', label='Training accuracy')
# b is for "solid blue line"
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title("Training and validation accuracy")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

## new data come how the model predict
y_test_pred = model.predict_classes(x_test)

loss, accuracy = model.evaluate(x_test, y_test)
print('test accuracy:', accuracy)

from keras.utils import np_utils
## convert integers to dummy variables (i.e. one hot encoded)
dummy_y_pred = np_utils.to_categorical(y_test_pred)

### classification report and confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
matrix = confusion_matrix(y_test.argmax(axis=1), dummy_y_pred.argmax(axis=1))
print(matrix)
from sklearn.metrics import classification_report
print(classification_report(y_test.argmax(axis=1), dummy_y_pred.argmax(axis=1),
digits=3))

print("To know about ANN model with word embedding")
print('choose the below options')
print("\n 1. adiads Brand reviews \n 2. skechers Brand reviews \n 3. crocs Brand Reviews")
try:
    r = int(input("Enter number of your choice : "))
except TypeError:
    print("TypeError")
except:
    print('invalid entry')
switch_fun(r)

#####packages installed

### install packages for word embedding
pip install gensim==3.8.1 --user
pip install nltk --user

### install packages for ANN
!pip install --upgrade pip --user
!pip install conda install tensorflow --user

```

```
!pip install conda install keras
!pip install imblearn --user
!pip install scikit-learn==0.24.1 --user
!pip install --upgrade scikit-learn --user
```

1.4 ANN word2vec

###Pretrained mode

```
import gensim.downloader as api
word2vec_model = api.load('word2vec-google-news-300')
```

###code ANN word2vec for adidas dataset

```
## importing the necessary library
import pandas as pd
import numpy as np
import datetime
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib notebook
import warnings
warnings.filterwarnings("ignore")

### reading the data as dataframe
df = pd.read_excel('shoes.xlsx',sheet_name="Sheet2")

### deleting the unwanted column
del(df["marketplace"])
del(df["customer_id"])
del(df["review_id"])
del(df["product_parent"])
del(df["vine"])
del(df["review_headline"])
del(df["total_votes"])
del(df["product_category"])

#### create a brand subset
df['product_title']=df['product_title'].apply(lambda x: x.lower())

conditions =[(df['product_title'].str.contains('adidas')),
              (df['product_title'].str.contains('crocs')),
              (df['product_title'].str.contains('skechers'))]
values=['adidas','crocs','skechers']
df['Brand']=np.select(conditions,values)
```

```

#### labeling the data using star rating
df["verified_purchase"]=df.verified_purchase.map({'Y':1,'N':0})
df["positivity"] = df["star_rating"].apply(lambda x: 2 if x>3 else(0 if x==3 else 1))

### #Text Cleaning
# 1.1 Define preprocess function
df["review_body"] = df["review_body"].astype("str")
import string
import nltk
nltk.download('words')
words = set(nltk.corpus.words.words())
stopwords = nltk.corpus.stopwords.words('english')
new_stopwords = ["i've", "i'm", 'on', 'ie', 'these for', 'im']
stopwords.extend(new_stopwords)
import re
wn=nltk.WordNetLemmatizer()

def removing_punc(ele):
    # Convert the text into lowercase
    ele = ele.lower()
    #punctuation
    ele = re.sub('[%s]' % re.escape(string.punctuation), "", ele)
    # number
    ele = re.sub(r'[0-9]', "", ele)
    #new line
    ele = re.sub('\n', "", ele)
    #white space
    ele= re.sub("^\\s+", "", ele)
    return ele
df["review_body"]=df["review_body"].apply(lambda x: removing_punc(x))

def tokenize(txt):
    """tokenize each word by using split() function"""
    tokens=re.split('\\W+', txt)
    return tokens
df['tokenized_message']=df['review_body'].apply(lambda x: tokenize(x))

def clean_word(txt_tokenized):
    """removed the stopword and remove the numbers and get the base word using lemmatize function"""

```



```

new_word = [word for word in txt_tokenized if word not in stopwords]
new_word = [word for word in new_word if word.isalpha()]
new_word = [word for word in new_word if word in words]
new_word = [wn.lemmatize(word) for word in new_word]
return " ".join(new_word)

df['st_cleaned_message']=df['tokenized_message'].apply(lambda x:clean_word(x))

### extracting only the adidas shoe brand reviews from the dataset
adidas = df[df["Brand"]=="adidas"].sort_values(by=["review_date"], ascending=False)

### Removing unwanted column
del(adidas['product_id'])
del(adidas['product_title'])
#del(adidas['review_body'])
del(adidas['review_date'])
del(adidas['Brand'])

## define x and y variable
x=adidas['st_cleaned_message']
y=np.array(adidas['positivity'])

## tokenization
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=6000)
tokenizer.fit_on_texts(x)

## convert text to sequence
sequence = tokenizer.texts_to_sequences(x)
word_index = tokenizer.word_index

### finding the max length of the sentence present in the document
c=[]
for i in adidas['st_cleaned_message']:
    c.append(len(i))
max_length = max(c)
print('maximun length of the sentence present in the document',max_length)

### padding the sequence to equal length by post padding
from tensorflow.keras.preprocessing.sequence import pad_sequences
padded_reviews = pad_sequences(sequence, maxlen=max_length, padding='post')

```

```

### converting the vector to matrix of shape (4157,300)
unique_words = len(word_index)
total_words = unique_words + 1
no_of_skipwords = 0
embedding_dim = 300
embedding_matrix = np.zeros((total_words, embedding_dim))
for word, index in tokenizer.word_index.items():
    try:
        embedding_vector = word2vec_model[word]
    except:
        no_of_skipwords = no_of_skipwords + 1
        pass
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
print('shape of matrix is:',embedding_matrix.shape)
print('no of skipped words:',no_of_skipwords)

## define data as independent variable
data = padded_reviews
from keras.utils import np_utils
## convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(y)

## split the dataset as train and test for evaluation
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, dummy_y, test_size=0.20,
random_state = 42)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Embedding
import keras
from keras.callbacks import EarlyStopping
import warnings

embedding_layer = Embedding(total_words, embedding_dim,
weights=[embedding_matrix],input_length=max_length,trainable = False)
model = Sequential()
model.add(embedding_layer)
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
# compile the model

```

```

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
# summarize the model
model.summary()

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
# early stopping callback
# This callback will stop the training when there is no improvement in
# the validation loss for 5 consecutive epochs.
es = keras.callbacks.EarlyStopping(monitor='val_loss',
                                   mode='min',
                                   patience=10,
                                   restore_best_weights=True)

from sklearn.utils import class_weight
class_weights = list(class_weight.compute_class_weight('balanced',
                                                       np.unique(adidas['positivity']),
                                                       adidas['positivity']))

weights={}
for index, weight in enumerate(class_weights) :
    weights[index]=weight

# model fit
history = model.fit(x_train,
                    y_train,
                    callbacks=[es],
                    class_weight=weights,
                    epochs=100,
                    batch_size=500,
                    shuffle=True,
                    validation_split=0.20,
                    verbose=1)

history_dict = history.history
# learning curve
# accuracy
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']

# loss
loss = history_dict['loss']

```

```

val_loss = history_dict['val_loss']

# range of X (no. of epochs)
epochs = range(1, len(acc) + 1)

# plot
# "r" is for "solid red line"
plt.plot(epochs, acc, 'r', label='Training accuracy')
# b is for "solid blue line"
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

## new data come how the model predict
y_test_pred = model.predict_classes(x_test)

loss, accuracy = model.evaluate(x_test, y_test)
print('test accuracy:', accuracy)

from keras.utils import np_utils
## convert integers to dummy variables (i.e. one hot encoded)
dummy_y_pred = np_utils.to_categorical(y_test_pred)
#### classification report and confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
matrix = confusion_matrix(y_test.argmax(axis=1), dummy_y_pred.argmax(axis=1))
print(matrix)
from sklearn.metrics import classification_report
print(classification_report(y_test.argmax(axis=1), dummy_y_pred.argmax(axis=1), digits=3))
#### prediction output for adidas dataset
y_test
df = pd.DataFrame(y_test)
df['Actual'] = df.cumsum(axis=1).ne(1).sum(axis=1)
actual = df['Actual']
data = {'Actual': actual, 'predicted': y_test_pred}
adidas_df = pd.DataFrame(data)
adidas_df.to_csv("adidas_prediction.csv", index=False)

#### ANN word2vec for Skechers Dataset

```

```

## importing the necessary library
import pandas as pd
import numpy as np
import datetime
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib notebook
import warnings
warnings.filterwarnings("ignore")

### reading the data as dataframe
df = pd.read_excel('shoes.xlsx',sheet_name="Sheet2")

### deleting the unwanted column
del(df["marketplace"])
del(df["customer_id"])
del(df["review_id"])
del(df["product_parent"])
del(df["vine"])
del(df["review_headline"])
del(df["total_votes"])
del(df["product_category"])

#### create a brand subset
df['product_title']=df['product_title'].apply(lambda x: x.lower())

conditions =[(df['product_title'].str.contains('adidas')),
              (df['product_title'].str.contains('crocs')),
              (df['product_title'].str.contains('skechers'))]
values=['adidas','crocs','skechers']
df['Brand']=np.select(conditions,values)

#### labeling the data using star rating
df["verified_purchase"]=df.verified_purchase.map({'Y':1,'N':0})
df["positivity"] = df["star_rating"].apply(lambda x: 2 if x>3 else(0 if x==3 else 1))

### #Text Cleaning
# 1.1 Define preprocess function
df["review_body"] = df["review_body"].astype("str")
import string
import nltk
nltk.download('words')
words = set(nltk.corpus.words.words())

```

```

stopwords = nltk.corpus.stopwords.words('english')
new_stopwords = ['i've', 'i'm', 'on', 'ie', 'these for', 'im']
stopwords.extend(new_stopwords)
import re
wn=nltk.WordNetLemmatizer()

def removing_punc(ele):
    # Convert the text into lowercase
    ele = ele.lower()
    #punctuation
    ele = re.sub('[%s]' % re.escape(string.punctuation), "", ele)
    # number
    ele = re.sub(r'[0-9]', "", ele)
    #new line
    ele = re.sub('\n', "", ele)
    #white space
    ele= re.sub("^\s+", "", ele)
    return ele
df["review_body"]=df["review_body"].apply(lambda x: removing_punc(x))

def tokenize(txt):
    """tokenize each word by using split() function"""
    tokens=re.split("\W+", txt)
    return tokens
df['tokenized_message']=df['review_body'].apply(lambda x: tokenize(x))

def clean_word(txt_tokenized):
    """removed the stopword and remove the numbers and get the base word using lemmatize function"""
    new_word = [word for word in txt_tokenized if word not in stopwords]
    new_word = [word for word in new_word if word.isalpha()]
    new_word = [word for word in new_word if word in words]
    new_word = [wn.lemmatize(word) for word in new_word]
    return " ".join(new_word)

df['st_cleaned_message']=df['tokenized_message'].apply(lambda x:clean_word(x))

### extracting only the adidas shoe brand reviews from the dataset
skechers = df[df["Brand"]=="skechers"].sort_values(by=["review_date"], ascending=False)

### Removing unwanted column

```

```

del(skechers['product_id'])
del(skechers['product_title'])
#del(skechers['review_body'])
del(skechers['review_date'])
del(skechers['Brand'])

## define x and y variable
x=skechers['st_cleaned_message']
y=np.array(skechers['positivity'])

## tokenization
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=6000)
tokenizer.fit_on_texts(x)

## convert text to sequence
sequence = tokenizer.texts_to_sequences(x)
word_index = tokenizer.word_index

### finding the max length of the sentence present in the document
c=[]
for i in skechers['st_cleaned_message']:
    c.append(len(i))
max_length = max(c)
print('maximun length of the sentence present in the document',max_length)

### padding the sequence to equal length by post padding
from tensorflow.keras.preprocessing.sequence import pad_sequences
padded_reviews = pad_sequences(sequence, maxlen=max_length, padding='post')

### converting the vector to matrix of shape (4801,300)
unique_words = len(word_index)
total_words = unique_words +1
no_of_skipwords = 0
embedding_dim = 300
embedding_matrix = np.zeros((total_words, embedding_dim))
for word, index in tokenizer.word_index.items():
    try:
        embedding_vector = word2vec_model[word]
    except:
        no_of_skipwords = no_of_skipwords + 1
        pass
    if embedding_vector is not None:

```

```

        embedding_matrix[index] = embedding_vector
print('shape of matrix is:',embedding_matrix.shape)

## define data as independent variable
data = padded_reviews
from keras.utils import np_utils
## convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(y)

## split the dataset as train and test for evalution
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, dummy_y, test_size=0.20,
random_state = 42)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Embedding
import keras
from keras.callbacks import EarlyStopping
import warnings

embedding_layer = Embedding(total_words, embedding_dim,
weights=[embedding_matrix],input_length=max_length,trainable = False)
model = Sequential()
model.add(embedding_layer)
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
# compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
# summarize the model
model.summary()

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
# early stopping callback
# This callback will stop the training when there is no improvement in
# the validation loss for 5 consecutive epochs.
es = keras.callbacks.EarlyStopping(monitor='val_loss',
                                mode='min',
                                patience=10,

```



```

        restore_best_weights=True)

from sklearn.utils import class_weight
class_weights = list(class_weight.compute_class_weight('balanced',
                                                         np.unique(skechers['positivity']),
                                                         skechers['positivity']))

weights={}
for index, weight in enumerate(class_weights) :
    weights[index]=weight

# model fit
history = model.fit(x_train,
                    y_train,
                    callbacks=[es],
                    class_weight=weights,
                    epochs=100,
                    batch_size=500,
                    shuffle=True,
                    validation_split=0.20,
                    verbose=1)

history_dict = history.history
# learning curve
# accuracy
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']

# loss
loss = history_dict['loss']
val_loss = history_dict['val_loss']

# range of X (no. of epochs)
epochs = range(1, len(acc) + 1)

# plot
# "r" is for "solid red line"
plt.plot(epochs, acc, 'r', label='Training accuracy')
# b is for "solid blue line"
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

```

```

plt.show()

## new data come how the model predict
y_test_pred1 = model.predict_classes(x_test)

loss, accuracy = model.evaluate(x_test, y_test)
print('test accuracy:',accuracy)

from keras.utils import np_utils
## convert integers to dummy variables (i.e. one hot encoded)
dummy_y_pred = np_utils.to_categorical(y_test_pred1)

### classification report and confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
matrix = confusion_matrix(y_test.argmax(axis=1), dummy_y_pred.argmax(axis=1))
print(matrix)
from sklearn.metrics import classification_report
print(classification_report(y_test.argmax(axis=1),dummy_y_pred.argmax(axis=1), digits=3))

### prediction output for Skechers dataset
y_test
df = pd.DataFrame(y_test)
df['Actual'] = df.cumsum(axis=1).ne(1).sum(axis=1)
actual = df['Actual']
data = { 'Actual': actual,'predicted':y_test_pred1 }
skechers_df = pd.DataFrame(data)
skechers_df
skechers_df.to_csv("skechers_prediction.csv",index=False)

###ANN Word2vec for crocs Dataset

## importing the necessary library
import pandas as pd
import numpy as np
import datetime
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib notebook
import warnings
warnings.filterwarnings("ignore")

### reading the data as dataframe
df = pd.read_excel('shoes.xlsx',sheet_name="Sheet2")

```

```

#### deleting the unwanted column
del(df["marketplace"])
del(df["customer_id"])
del(df["review_id"])
del(df["product_parent"])
del(df["vine"])
del(df["review_headline"])
del(df["total_votes"])
del(df["product_category"])

#### create a brand subset
df['product_title']=df['product_title'].apply(lambda x: x.lower())

conditions =[(df['product_title'].str.contains('adidas')),
              (df['product_title'].str.contains('crocs')),
              (df['product_title'].str.contains('skechers'))]
values=['adidas','crocs','skechers']
df['Brand']=np.select(conditions,values)

#### labeling the data using star rating
df["verified_purchase"]=df.verified_purchase.map({'Y':1,'N':0})
df["positivity"] = df["star_rating"].apply(lambda x: 2 if x>3 else(0 if x==3 else 1))

### #Text Cleaning
# 1.1 Define preprocess function
df["review_body"] = df["review_body"].astype("str")
import string
import nltk
nltk.download('words')
words = set(nltk.corpus.words.words())
stopwords = nltk.corpus.stopwords.words('english')
new_stopwords = ["i've", "i'm", 'on', 'ie', 'these for', 'im']
stopwords.extend(new_stopwords)
import re
wn=nltk.WordNetLemmatizer()

def removing_punc(ele):
    # Convert the text into lowercase
    ele = ele.lower()
    #punctuation
    ele = re.sub('[%s]' % re.escape(string.punctuation), "", ele)

```

```

# number
ele = re.sub(r'[0-9]', '', ele)
#new line
ele = re.sub('\n', '', ele)
#white space
ele= re.sub("^\\s+", "", ele)
return ele
df["review_body"]=df["review_body"].apply(lambda x: removing_punc(x))

def tokenize(txt):
    """tokenize each word by using split() function"""
    tokens=re.split('\\W+', txt)
    return tokens
df['tokenized_message']=df['review_body'].apply(lambda x: tokenize(x))

def clean_word(txt_tokenized):
    """removed the stopword and remove the numbers and get the base word using lemmatize
function"""
    new_word = [word for word in txt_tokenized if word not in stopwords]
    new_word = [word for word in new_word if word.isalpha()]
    new_word = [word for word in new_word if word in words]
    new_word = [wn.lemmatize(word) for word in new_word]
    return " ".join(new_word)

df['st_cleaned_message']=df['tokenized_message'].apply(lambda x:clean_word(x))

### extracting only the adidas shoe brand reviews from the dataset
crocs = df[df["Brand"]=="crocs"].sort_values(by=["review_date"], ascending=False)

### Removing unwanted column
del(crocs['product_id'])
del(crocs['product_title'])
#del(crocs['review_body'])
del(crocs['review_date'])
del(crocs['Brand'])

## define x and y variable
x=crocs['st_cleaned_message']
y=np.array(crocs['positivity'])

## tokenization
import tensorflow as tf

```

```

from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=6000)
tokenizer.fit_on_texts(x)

## convert text to sequence
sequence = tokenizer.texts_to_sequences(x)
word_index = tokenizer.word_index

### finding the max length of the sentence present in the document
c=[]
for i in crocs['st_cleaned_message']:
    c.append(len(i))
max_length = max(c)
print('maximun length of the sentence present in the document',max_length)

### padding the sequence to equal length by post padding
from tensorflow.keras.preprocessing.sequence import pad_sequences
padded_reviews = pad_sequences(sequence, maxlen=max_length, padding='post')

### converting the vector to matrix of shape (4157,300)
unique_words = len(word_index)
total_words = unique_words + 1
no_of_skipwords = 0
embedding_dim = 300
embedding_matrix = np.zeros((total_words, embedding_dim))
for word, index in tokenizer.word_index.items():
    try:
        embedding_vector = word2vec_model[word]
    except:
        no_of_skipwords = no_of_skipwords + 1
        pass
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
print('shape of matrix is:',embedding_matrix.shape)

## define data as independent variable
data = padded_reviews
from keras.utils import np_utils
## convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(y)

## split the dataset as train and test for evaluation
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, dummy_y, test_size=0.20,

```

```

random_state = 42)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Embedding
import keras
from keras.callbacks import EarlyStopping
import warnings

embedding_layer = Embedding(total_words, embedding_dim,
weights=[embedding_matrix],input_length=max_length,trainable = False)
model = Sequential()
model.add(embedding_layer)
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
# compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
# summarize the model
model.summary()

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
# early stopping callback
# This callback will stop the training when there is no improvement in
# the validation loss for 5 consecutive epochs.
es = keras.callbacks.EarlyStopping(monitor='val_loss',
                                   mode='min',
                                   patience=10,
                                   restore_best_weights=True)

from sklearn.utils import class_weight
class_weights = list(class_weight.compute_class_weight('balanced',
                                                       np.unique(crocs['positivity']),
                                                       crocs['positivity']))

weights={}
for index, weight in enumerate(class_weights) :
    weights[index]=weight

# model fit

```

```

history = model.fit(x_train,
                    y_train,
                    callbacks=[es],
                    class_weight=weights,
                    epochs=100,
                    batch_size=500,
                    shuffle=True,
                    validation_split=0.20,
                    verbose=1)

history_dict = history.history
# learning curve
# accuracy
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']

# loss
loss = history_dict['loss']
val_loss = history_dict['val_loss']

# range of X (no. of epochs)
epochs = range(1, len(acc) + 1)

# plot
# "r" is for "solid red line"
plt.plot(epochs, acc, 'r', label='Training accuracy')
# b is for "solid blue line"
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

## new data come how the model predict
y_test_pred2 = model.predict_classes(x_test)

loss, accuracy = model.evaluate(x_test, y_test)
print('test accuracy:', accuracy)

from keras.utils import np_utils
## convert integers to dummy variables (i.e. one hot encoded)
dummy_y_pred = np_utils.to_categorical(y_test_pred2)

```

```

### classification report and confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
matrix = confusion_matrix(y_test.argmax(axis=1), dummy_y_pred.argmax(axis=1))
print(matrix)
from sklearn.metrics import classification_report
print(classification_report(y_test.argmax(axis=1), dummy_y_pred.argmax(axis=1), digits=3))

### prediction output for Skechers dataset
y_test
df = pd.DataFrame(y_test)
df['Actual'] = df.cumsum(axis=1).ne(1).sum(axis=1)
actual = df['Actual']
data = {'Actual': actual, 'predicted': y_test_pred2}
crocs_df = pd.DataFrame(data)
crocs_df
crocs_df.to_csv("crocs_prediction.csv", index=False)

```