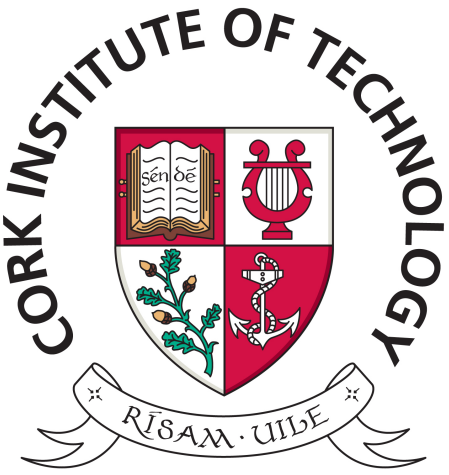


Sentiment analyis for large scale online reviews

by

Shrey Mishra

This thesis has been submitted in partial fulfillment for the degree of Master of Science in MSc in Cloud Computing

in the
Faculty of Engineering and Science
Department of Computer Science

February 2022

Declaration of Authorship

I, Shrey Mishra , declare that this thesis titled, "Sentiment analyis for large scale online reviews" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a masters degree at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- I understand that my project documentation may be stored in the library at CIT, and may be referenced by others in the future.

Signed: _____

Date: _____

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering and Science
Department of Computer Science

Master of Science

by Shrey Mishra

Sentiment analysis is a vital task in the domain of customer feedback. It is extremely crucial to determine the quality of a product or a service via user feedback, as this gives a precise representation of the brand value associated with the product/service. It also indicates any scope of development, but because of the intricate language rules (word usages), Heavy imbalances towards positive reviews along with unseemly word characters and emoticons used in the reviews poses a real challenge to visualise the data and precisely build machine learning models for deployment. In recent years, deep learning techniques have shown promising results in various application domains. In particular, the Transformer model (BERT) has been used extensively and executed well on applications involving sequence to sequence learning alongside attention modelling. For this project, We propose an automated learning model for accurate sentiment analysis (polarity). In this, we used the Amazon fine food reviews data set publically available at Kaggle. By using the sequence to sequence network models with transformers, we generated an F1 Score of **97%**

Acknowledgements

I want to express my deepest gratitude to the entire faculty of MSc in Artificial Intelligence for making this journey so smooth and enriching. I want to express the most profound appreciation for **Dr Mohammed Hasanuzzaman** who has continuously and convincingly shown a spirit of adventure concerning research, and excitement regarding teaching. It would not have been possible without his encouragement and constant support.

Lastly, I would like to thank Dr Victor Cionca, who introduced me to the Research and ethics module, which was applied exclusively for research studies, and whose enthusiasm has had a lasting impact....

Contents

List of Figures

List of Tables

Abbreviations

DL	**D**eep **L**earning
ML	**M**achine **L**earning
AI	**A**rtificial **I**ntelligence
NLP	**N**atural **L**anguage **P**rocessing
LSTM	**L**ong **S**hort **T**erm **M**emory
GPU	**G**raphics **P**rocessing **U**nit
TPU	**T**ensor **P**rocessing **U**nit
SOM	**S**elf **O**rganinizing **M**aps
BERT	**B**idirectional **E**ncoder **R**epresentations **T**ransformers
TPE	**T**ree **P**arzen **E**stimator
EDA	**E**xploratory **D**ata **A**nalysis
KNN	**K** **N**earest **N**eighbours
SVM	**S**upport **V**ector **M**achines
GBM	**G**radient **B**oosting **C**lassifier
LGBM	**L**ight **GBM**
ANN	**A**rtificial **N**eural **N**etworks
RNN	**R**ecurrent **N**eural **N**etworks
GRU	**G**ated **R**ecurrent **U**nit
SMOTE	**S**ynthetic **M**inority **O**ver-sampling **T**echnique
NAG	**N**estrov **A**ccelerated **G**radient Descent
CBOW	**C**ontinuous **B**ag **O**f **W**ords
BMU	**B**est **M**atching **U**nit
HTML	**H**yper **T**ext **M**arkup **L**anguage
POS	**P**art **O**f **S**peech
PCA	**P**rincipal **C**omponent **A**nalysis

TSNE	**t**-distributed **S**tochastic **N**eighbourhood **E**mbedding
UMAP	**U**niform **M**anifold **A**pproximation **P**rojection
LZMA	**L**empel **Z**iv **M**arkov **A**lgorithm

Dedicated to my Parents...

Chapter 1

Introduction

1.1 Motivation

Sentiment analysis is a crucial task of identifying public opinions about a product; It may be as service or even a general public perception of a political incident. It can be used in a diversity of tasks such as:

- Consumer feedback for improvement of quality of the product [3] **for selecting the target audience for marketing campaigns.**
- **Understanding public opinion for identifying political** decisions[4]
- Identifying and taking **actions against discriminatory and inapt language usage** among social media platforms such as Racist tweets.[5]

Many large scale industries have publicised their data for research purposes such as Yelp [6], Amazon, Twitter varying in a large number of categories widely ranging from a broad selection such as food, beer, and movie reviews [7]. The sheer volume of the reviews generated every day poses a new possibility of sentiment analysis, where there are a plethora of underlying applications. Since new data generated can accommodate to the latest trends and can understand the sentiment drift, and continuous algorithmic improvements have been made for almost a decade since the original Word2Vec was released in 2013. The domain of Natural Language Processing (NLP) has evolved a lot. This poses a relevant question, such as which algorithm performs well for the task of sentiment classification. With this project, we aim to answer the above question by comparing the performance of various techniques in the domain of Sentiment analysis.

Since the Natural language itself has many intricate rules of grammar, It is extremely challenging to interpret these reviews into vectors such that our classifiers can effectively draw high dimensional decision boundaries across them. With the current proposal along with classification task, we also investigate several visualisation techniques for high dimensional data alongside we also preserve Emoji based information present in the review.

1.2 Objectives

Following are the research objectives within the scope of the project :

- For a given a review, to be analysed, the classifier should be able to predict the polarity of the review.
- Handling Class imbalance in the reviews.
- Building a wide range of machine learning classification models and accessing the performance of each of the Classifiers.
- Building sequence to sequence models for capturing the neighbourhood as the context of words usage heavily depends on the surrounding words/sentences.

A brief list of challenges resolved:

- **Preprocessing challenges:** Remove duplicate entries, HTML tags, weblinks, retaining and transforming Emoji's to avoid data leakage.
- **Algorithmic challenges:** Building appropriate vector embedding to represent the data, Visualisation of sentimental data via appropriate dimensionality reduction algorithms, dealing with the class imbalance and selecting appropriate classifiers.
- **Tuning based:** Application of enormous space of hyperparameters and efficiently tuning classifiers to the exact set of hyperparameters that maximise the performance. Scaling the algorithms to GPU's and TPU's for faster training time and ensuring Low-bias variance tradeoff by applying three-fold cross-validation.
- **Run-time/ Space Complexity of the proposed solution:** Comparing the run-time performance of the proposed solution.

1.3 Contribution

Sentiment analysis is a challenging task as our results not only depend upon the classifier build but also on the several other aspects such as preprocessing techniques, bias-variance tradeoffs and the hyperparameter selection. The problem is especially typical when there exist class imbalance and use of special characters such as Emoji's which naturally exists and seems to favour all the positive reviews. With the implementations, the process can be automated within a few seconds helping large organisations in making strategic decisions based on customer feedback.

1.4 Structure of This Document

The document is structured as follows:

- Chapter 1 of this document presents a brief introduction to the project. Dataset and the underlying reason as to why this problem is imperative.
- Chapter 2 Discusses the existing methods which are present for sentiment analysis.
- Chapter 3 gives an overview of the architecture and the methodology needed to follow for solving this problem.
- Chapter 4 describes the detail implementation part of the architecture and the configurations used in our models.
- Chapter 5 shows the evaluation and results obtained on several experiments while comparing them to the standard baseline models.
- Chapter 6 concludes the thesis by observing the experiments as a whole. This chapter also mentions the potential areas that can be approached as future studies.

Chapter 2

Literature Review

In the initial years of sentiment analysis, such as in 2009, **Go et al.**[8] used sentiment analysis over tweets sent on data where the positive tweets were identified based on Naive Bayes and Support vector machine classifiers trained on Tweets, where the emoji's such as ":)" ":-)" resembled positive while ":(" ":-(" resembled as negative. In the proposed paper [8], authors outlined that Support vector machines beat all the other classifiers achieving 81.19%.

Their approach was based on unigram, bigram word selection in conjunction with POS tagging. Out of all the N-grams approach, authors found unigram approach with POS tags to be the most helpful. Later in 2011, **Apoorv Agarwal et al.**[9] group proposed a polarity based tagging where they created a manual polarity dictionary of all the positive words and all the negative words and then applied linguistic-based feature generation. While the approach was constructive, but they were only able to get 60% accuracy knocking the current state of the art at the time by 4% nearly.

While linguistic features are essential for the grammatical understanding of the language, however, they do not add any semantic meaning to the sentence for which the classifiers can identify and differentiate.

While much research with a combination of POS features and TF-IDF score was made to do decent feature engineering, However, they were failing to capture the semantic meaning of the sentences. In 2013, **Thomas Mikolov et al.** [10] published a new technique called word2vec, where the word embedding made incorporated the semantic meaning of the sentiment. As for testing, the word embeddings were trained on a large corpus of Google news data, and it did well in understanding the semantic meaning of words such as the Country, distances, Gender relationships, Competitor brands.

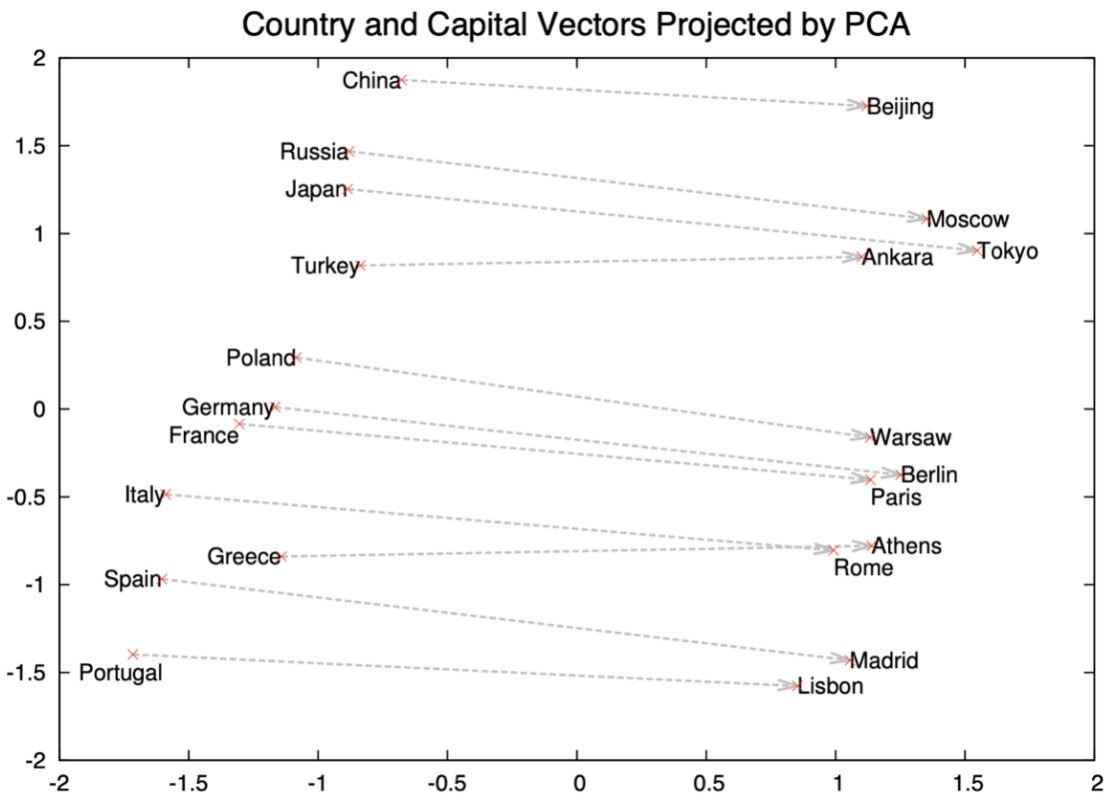


FIGURE 2.1: Word embeddings projected on PCA dimensionality reduced map.

In 2014, the Stanford group released open-source word embeddings known as **Glove embeddings** that were trained on even larger corpus such as Twitter, Wikipedia, and Commoncrawl total constituting more than 6 billion words in the vocabulary. However, the Glove embeddings took global count average for selecting a particular word.

These embeddings learnt were used in conjunction with the traditional Machine learning classifier such as SVM and Naive Bayes resulted in many impressive applications such as Spam message filtering[11] and Chinese sentiment analysis [12].

The vector embedding approach of preprocessing was implemented with a wide range of evolving classifiers in the field of Machine learning applications seemed to do well with many neural network-based applications using the Computation power of GPU's to train neural networks[13].

However, there were two significant problems with the Deep neural network-based approaches:

- The Gradient vanishing problem
- MLP's were failing to capture sequential information present in the data.

Many solutions have been proposed to tackle the aforementioned such as the 1D CNN'S for text classification alongside LSTM with pre-trained word embedding to capture Sequence as well as time-based information.

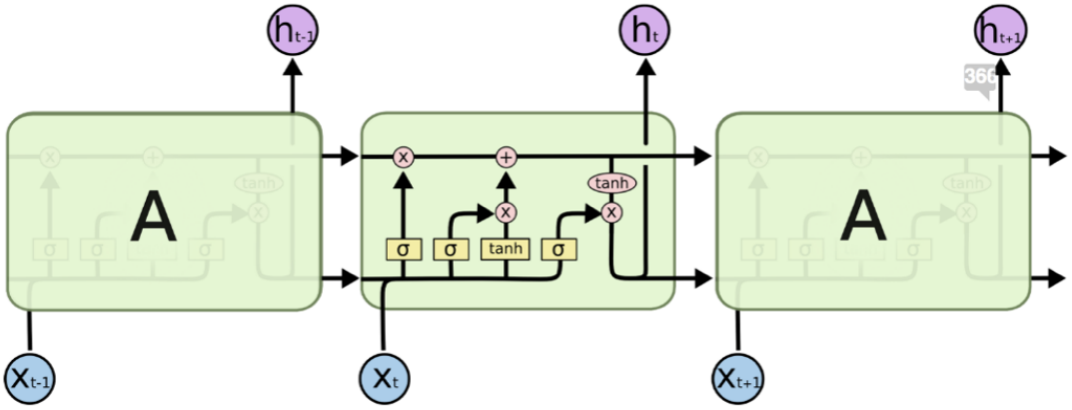


FIGURE 2.2: LSTM based architecture

Hence, Bidirectional LSTM's was used and gave a significant boost in performance specifically when the sequence information was dependent upon the words that appeared in the later part of sentences. This transformed many trivial applications such as machine translation where classic LSTM's faced troubles performing over massive sentences with many words. This approach helped to translate and to understand complex sentiment classification problems that have their flow of information in both directions.

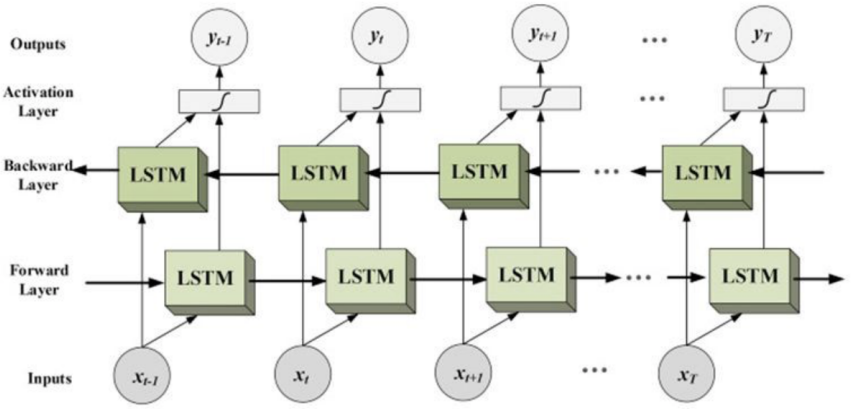


FIGURE 2.3: Bidirectional LSTM based architecture

These solutions could not work well when the context word was placed out of the window size of the kernel or appeared in the latter part of the sentence[14].

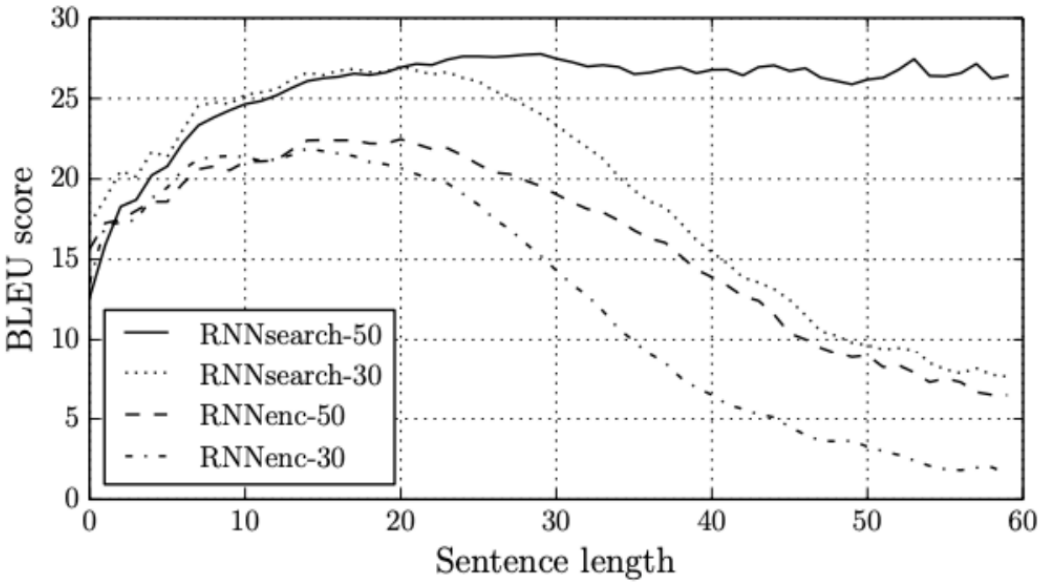


FIGURE 2.4: BLEU score for large sentences when using Transformer based models

Later in 2017, Google released their new attention-based models in the paper [14] titled **"Attention is all you need"** where the authors made an encoder structure using bidirectional RNN while the decoder section is a unidirectional RNN network. This architecture was known as a transformer model.

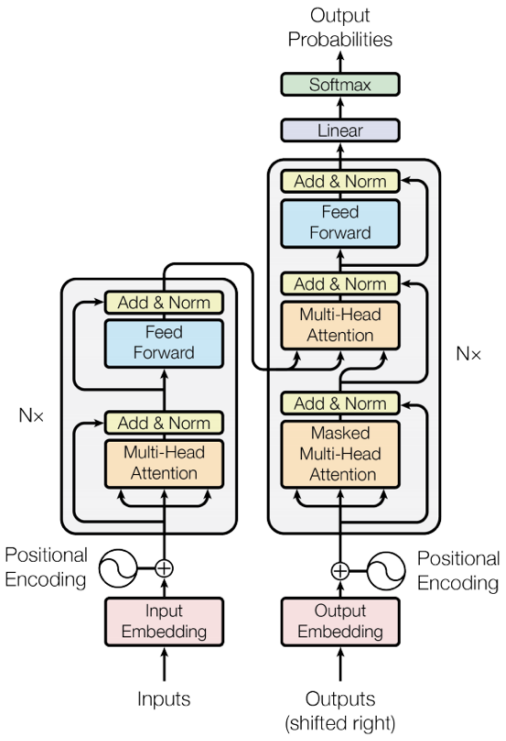


FIGURE 2.5: Attention model

On May 2019, A new research paper[15] from **Google AI language team** surfaced introducing **Deep Bidirectional Transformers for Language Understanding**, also known as BERT. The authors in the paper introduced a stack of Encoder, Decoder models where the Input was directly fed to all the Decoders causing the flow of sequence information to be held even in the last Decoder block. This approach also held the attention mechanism (referring to the context words) from the attention-based models.

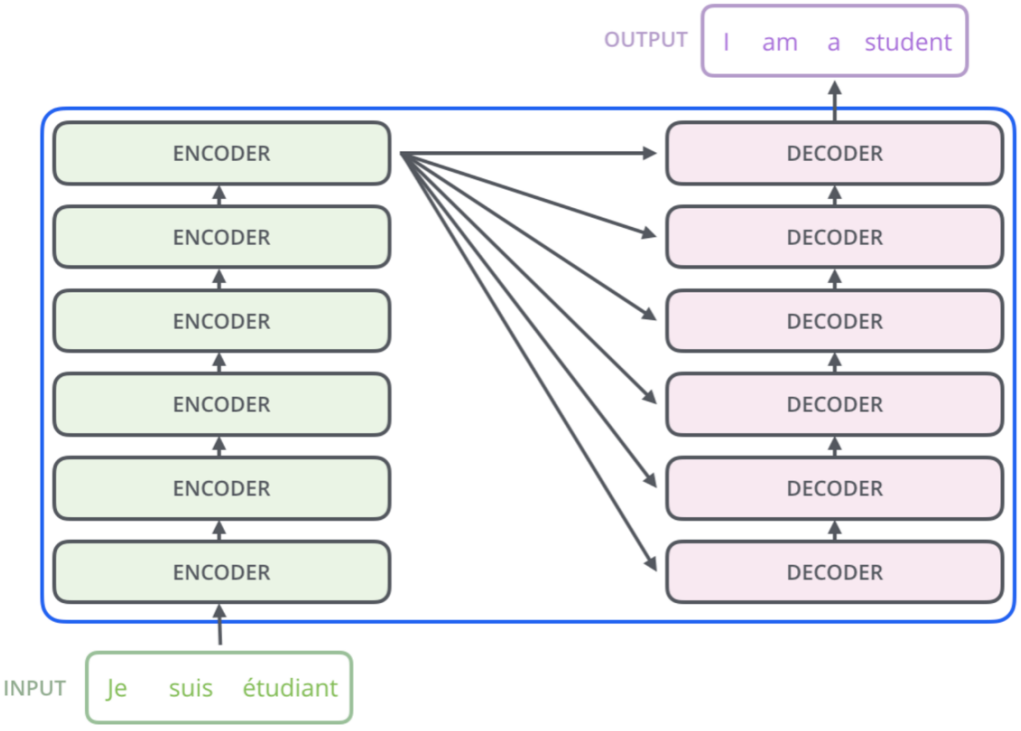


FIGURE 2.6: A representation of Encoder/Decoder stacks in BERT

Within each encoder block containing a dedicated self-attention unit posed to capture the contextual references of important words.

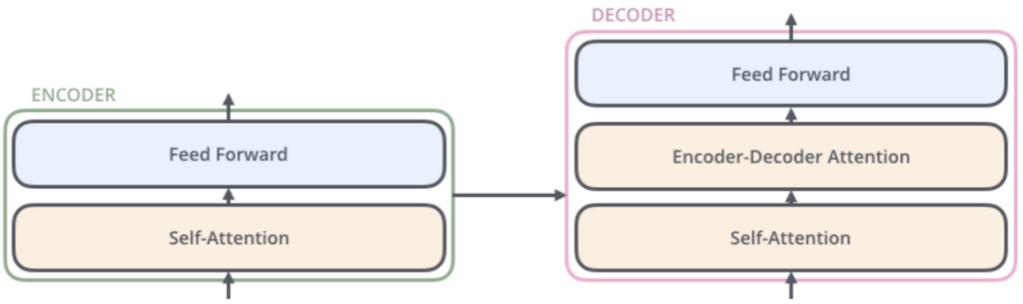


FIGURE 2.7: High level overview of Encoder block in BERT

The dimensionality of each word is kept consistent in the word embeddings to make sure the meaning or the flow of information, across the stacks of Encoder goes so there is no dimensionality modification. The recurrent connections from the encoder block ensure that if a particular encoder block is not learning anything the sequence information retained is still carried forward from the previous encoder blocks to the decoder blocks when generating the output.

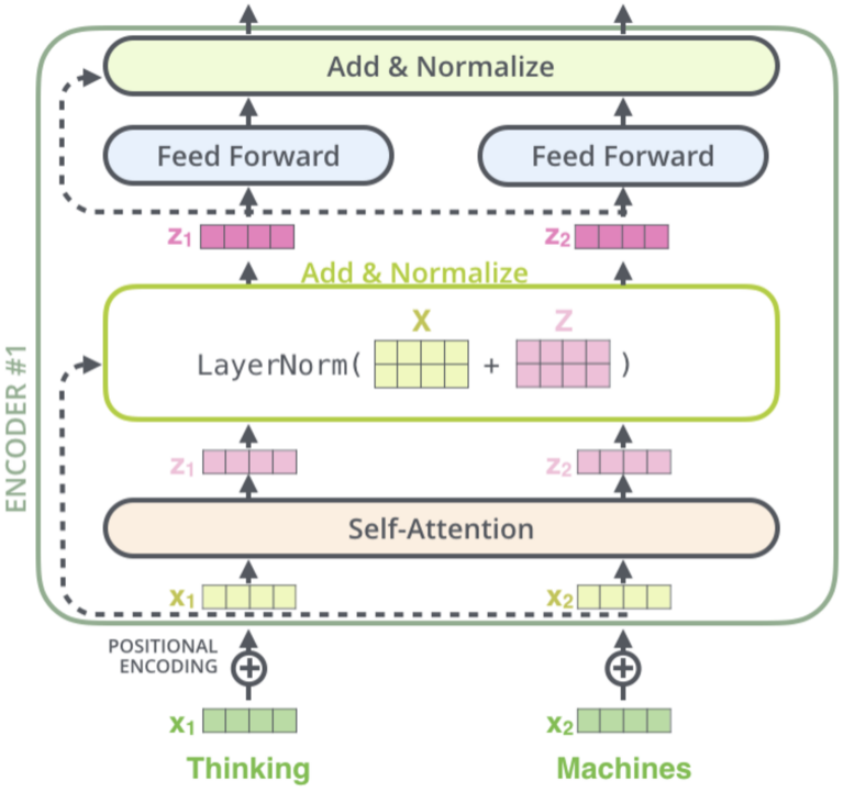


FIGURE 2.8: Dimensionality of words inside the encoder block

BERT based transformers show a notable advantage of having bidirectional self-attention over the Open AI's GPT-2 model with constrained self-attention. Hence outperforming GPT model by an average 7.1% margin resulting in a total of 82.1% accuracy[15].BERT scored 83.2% F1 score on Squad 2.0 dataset performing marginally close to the human performance of 86.8% as mentioned by Rajpurkar et al. [16].

Chapter 3

Methodology- An algorithmic overview

3.1 Problem Definition

The primary aim of the dissertation is to explore a variety of different solutions that can be posed as a Sentiment analysis problem, making it real-world relevant even for long, complex sentences where the context words appear in the later section of the review.

3.2 Functional Requirements

Functional Requirements mentioned below are different problems to be solved :

- The model should be able to measure performance when different sentimental reviews are feed, i.e. test data in our case.
- Evaluate a range of Machine Learning and Deep Learning Algorithms for sentiment classification.
- To solve the problem of imbalance by exploring various techniques
- To be able to classify reviews with a good accuracy along with decent F1 score.

3.3 Non-Functional Requirements

Below requirements are not critical to the completion of the project but shall add more value:

- To allow automatic retraining of model over a specific period to keep it updated with the changing pattern of reviews.
- Data protection, compliance with GDPR, intellectual property rights, shall be upheld.

3.4 Design

In this section, we shall discuss the overall Workflow.Also, the detailed explanation of the Workflow of this project.

3.4.1 System Architecture and Workflow

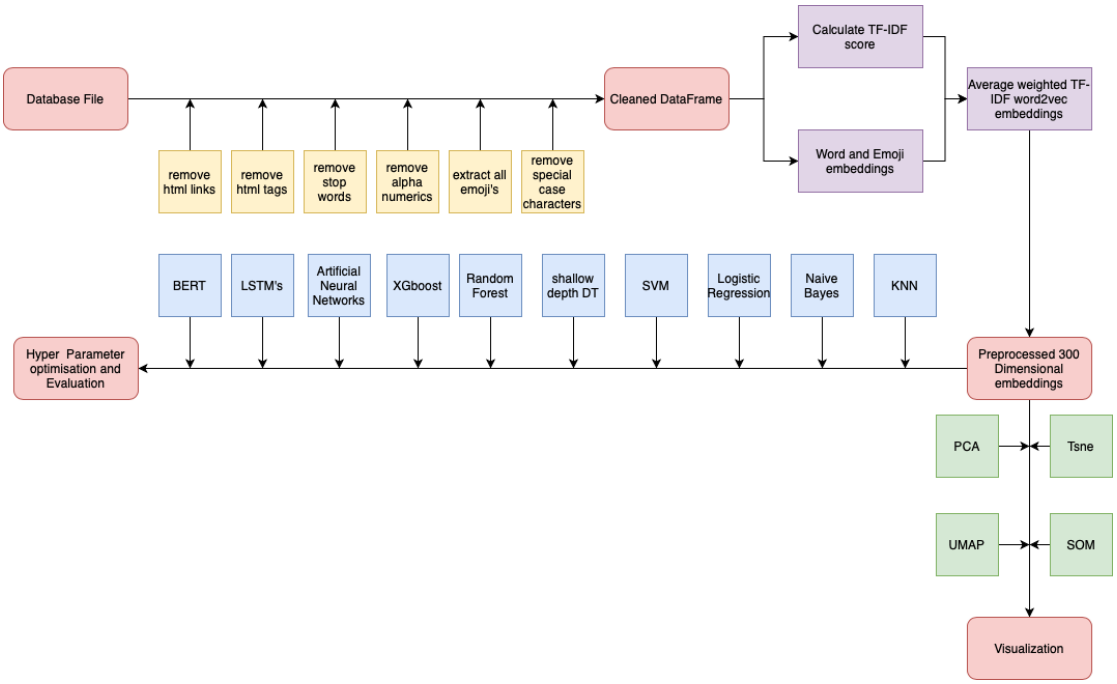


FIGURE 3.1: Design flow of the project

Figure 3.1 shows the overall architecture of the System. The overall flow is as follows:

1. Removal of duplicate entries, and irrelevant features such as Product ID, time of the review.
2. Text-based preprocessing such as removal of HTML tags, HTML links, special characters and alphanumeric.
3. Once the reviews are preprocessed we then convert words into vector embeddings where the entire sentence is transformed into a vector projection

4. Classifiers are trained on projected word embeddings generated for each sentence
5. Each of the Classifiers are Hyperparameter tuned to get the maximum F1 score evaluated over three-fold cross-validation data.

3.5 Methodology

This section gives a general overview of the core concepts in Artificial Intelligence (AI) and the methodologies which will be used in detecting the polarity of the review.

3.5.1 Preprocessing

Preprocessing is referred as getting the data ready cleaned up for the classifiers to act upon these to include filling up missing values, removal of irrelevant information from the data such as redundant features, duplicate entries and domain-specific preprocessing [17].

Preprocessing steps carried out for this dataset are as follows:

- Removal of all the 3-star reviews
- Removal of duplicate entries in the review
- Removal of all the rows where the people that found the review helpful is greater than the total number of people that voted.
- Removal of HTML links in the data
- Removal of HTML tags
- Transformation of abbreviations such as can't should've etc
- Removal of alphanumerics
- Removal of special characters if Emoji's do not exist
- Removal of all the stop words other than "nor", "not" and "no".
- Converting reviews into lowercases

Original review	Preprocessed review	label
This black strap molasses I use to get plenty of nutrients into my body...I take a tablespoon or two a day, to help with my moods and help to alleviate pain...this stuff works in the body like no bodies business ♥;	black strap molasses use get plenty nutrients body...i take tablespoon two day, help moods help alleviate pain...this stuff works body like no bodies business ♥	1
use no other salt ever... this is the real deal...buy it try it and get rid of all your other salts...which serve no real healthy bennies, compared to the real Salt Brand ♥ love the taste and love how I can consume it and not swell up like regular salt does to me ♥;	use no salt ever... real deal...buy try get rid salts...which serve no real healthy bennies, compared real salt brand ♥ love taste love consume not swell like regular salt ♥	1
Newman's Own®Organics Dog Treats for Small Size Dogs, Peanut Butter, 10-Ounce Bags (Pack of 6)/ My Pom would not eat one, he would live on Treats, if I let him. I only give him the best, like the above, but he did not go for this.	newman ® dog treats small size dogs, peanut butter, bags (pack pom would not eat one, would live ontreats, let him.i give best, like above, not go this.	0

FIGURE 3.2: Preprocessing of reviews

3.5.2 Word embeddings:

- **Bag of words:** Bag of words based techniques denote the presence of a word in a vector form, where the count of each word is represented in the vector. These embeddings do not effectively deal with negations and tend to completely deviate from the sentimental analysis.

FIGURE 3.3: Bag of words Example

- **TF-IDF score embeddings:** A survey conducted in 2015 showed that 83% of text-based recommender systems in digital libraries use tf–idf[18] which is similar to Bag of words except the vector representation is based upon how frequent or rare the word is inside the review/corpus.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{3.1}$$

Here the TF (denotes the term frequency) of the word, while the IDF (Inverse document frequency) The term was initially coined "term specificity", by Karen Spärck Jones in a 1972 paper [19]. Where the use of logarithmic function was initially based on Zipf's law [20] where the distribution of the usage of words was observed to follow a power-law distribution which on the application of a monotonic

function such as logarithm converts the distribution to a linear and hence does not overshadow the TF term in the final TF-IDF score.

- **CBOW/ SKIP-gram based Auto-encoder embeddings:**

  In order to incorporate the semantic meaning of the sentence, we will use the word2vector model introduced in 2013 by Thomas Mikolov [21], which takes the semantic meaning of the words.

  There are two general types of Word2vec models based on Auto-Encoders architecture.

  – CBOW(Continuous bag of words)

  – Skip gram

  In CBOW approach, we try to predict the focus word based on the context words around the focus word, here each of the words is represented as one-hot encoded representation. The activation used is a linear activation, or in other words, it is a simple weighted sum. The output of the function is measured by the softmax function proposed in the last layer. The word embedding is nothing but the transpose of the weights between the output (softmax layer) and the hidden layer.

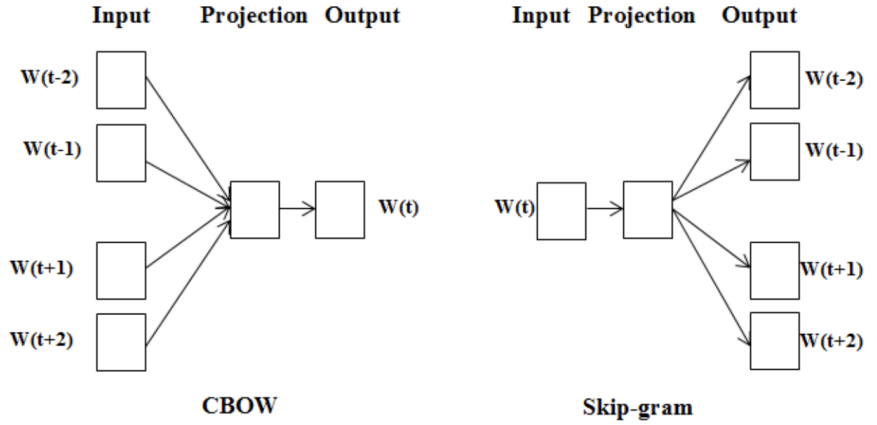


FIGURE 3.4: CBOW / Skipgram based auto encoder architecture

In the skip-gram approach, we try to predict the context words based on the focus word, and this approach Is computationally expensive as compared to CBOW because for each context word prediction we apply a softmax function hence for predicting n such context words we need n such softmax units.

The number of linear activation units put in the projection layer denote the dimensions we want our vector to be expressed in which is 300 for pre-trained word embeddings from Google News.

According to the author's note [22], CBOW performs faster training while skiagram learns much more valid embeddings.

- **Hybrid approach:** Mikolov in his paper [21], introduced negative sampling to train these words faster, According to the paper only a sample of weights associated with a word will be updated, The words are sampled by a probability distribution function denoted as

$$P\left(w_i\right) = 1 - \sqrt{\frac{t}{f\left(w_i\right)}} \tag{3.2}$$

Where the f(wi) denotes the frequency of the word "i" and in the original paper t was denoted as a constant of value 10^{-5} Intuitively, if the word is persistent, then it is not always picked to update its associated weights.

By leveraging out the benefits of both the approaches, I.e. TF-IDF approach and Word2vec, we will prepare reviews as an avg TF-IDF word2vec representation, where each of the word vectors is also multiplied by its TF-IDF score and then scaled with the total TF-IDF score of each word. This approach can be found in the paper [13], where the authors claimed to have an average 4.45% increase than the traditional approaches on various text classification tasks.

Since Word2vec embeddings do not provide an exhaustive representation of Emoji's we will use Emoji2vec based on the paper [23], For a theoretical perspective the embeddings were trained on twitter data, rather than google news and hence can't be added unless their origin of the feature transform is same, but for the practical implementation, we will add the two vector representations as evidenced in the paper [23] leading to an increment in the performance when using with word2vec when used in combination with emoji2vec.

A comparison of various Embeddings based on top similar words is mentioned below:

Given word	Google's Word2vec	Self word2vec trained on Amazon Data	Emoji2vec
coffee	"coffees" "Coffee" "Starbucks" "coffee_beans" "latté" "cappuccino"	"coffe" "coffees" "espresso" "columbian" "brew"	Not in vocabulary
great	"terrific" "fantastic" "tremendous" "good" "amazing" "awesome"	"fantastic" 'terrific' 'excellent' 'awesome' 'incredible'	Not in vocabulary
®	"TM" "™" "tm" "Avoderm_®" "®_SHOWTIME_FAMILYZONE"	"sauce¾" "formulas" "lecithin"	Not in vocabulary

FIGURE 3.5: Comparison of various Embedding schemes

Given word	Google's Word2vec	Self word2vec trained on Amazon Data	Emoji2vec
😍	Not in vocabulary	"👍" "again." "epicatechin" "mayospring"	"😻" "😘" "😽" "❤️" "💚"
Hogwarts	"Hogwart" "Death_Eaters" "wizarding" "Dumbledore" "Voldemort" "dementors"	Not in vocabulary	Not in vocabulary

FIGURE 3.6: Emoji embeddings

3.5.3 Dimensionality reduction techniques for visualisation :

- **Principal Component Analysis (PCA)[24]:** A dimensionality reduction technique used for visualisation of generally linearly separable data, where the axes with maximum variance are preserved. We calculate the Eigenvectors for each PCA transformed features where the largest Eigenvalue corresponds to the most dominant PCA axis. Top-2 or three-axis with most significant Eigenvalues is selected and used for a visualisation based on 2d/3d plot.

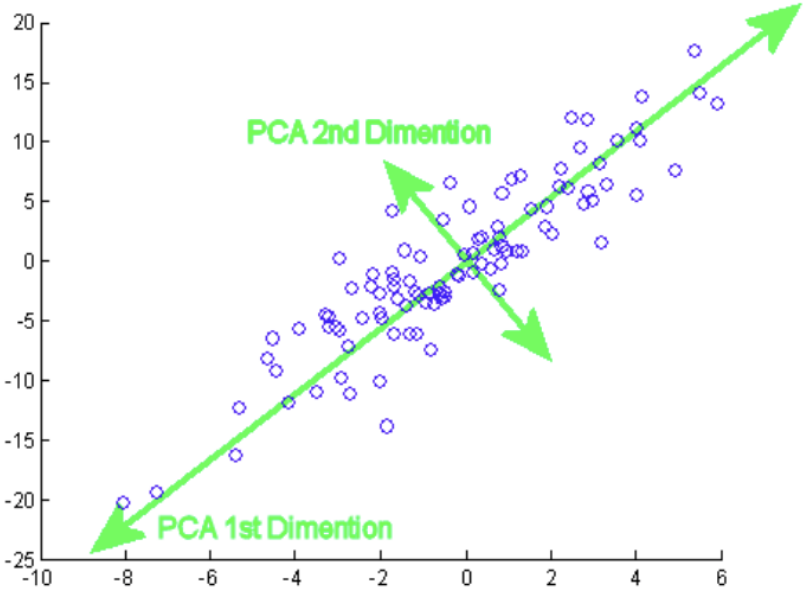


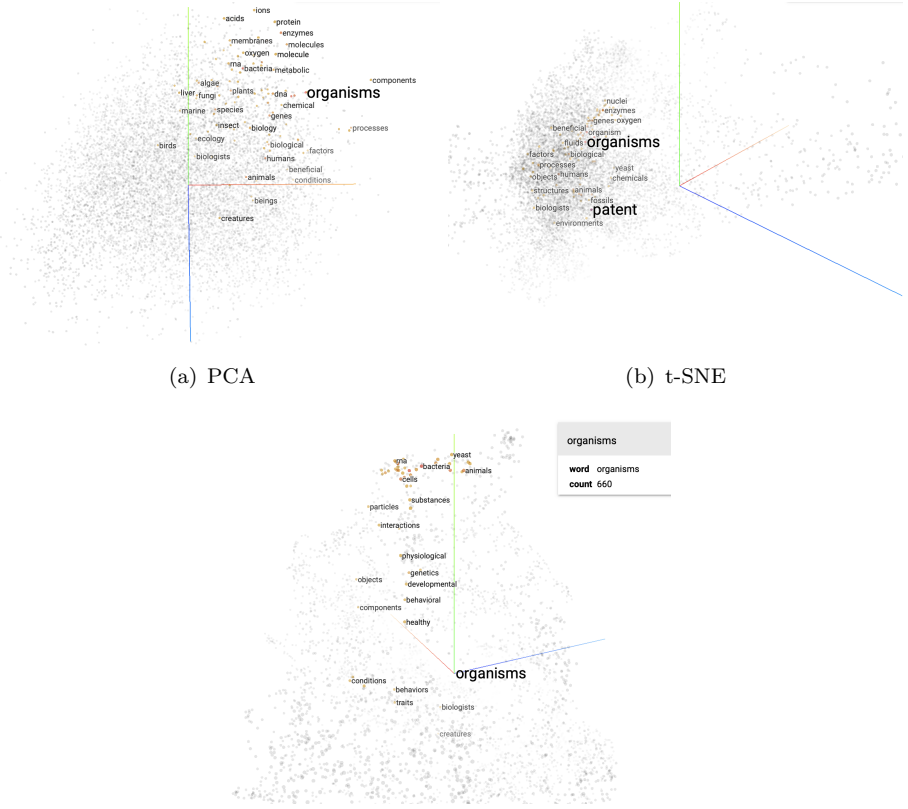
FIGURE 3.7: PCA projections over different axis

- **t-distributed stochastic neighbour embedding(t-SNE) [25]:**A dimensionality reduction technique for visualisation that does not assume data to be linearly separable, unlike PCA It maps the data in such a way that most similar points are grouped together while different points are kept much farther.

FIGURE 3.8: tsne based neighbourhood visualizations of Emoji vectors

- **UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction:** Similar to t-SNE, however, can actually be used for dimensionality reduction along with side visualisation. Data is assumed to be distributed uniformly across Riemannian manifold. From these assumptions, it is possible to model the manifold with a fuzzy topological structure. The embedding is found by searching for a low dimensional projection of the data that has the closest possible equivalent fuzzy topological structure [26].

  Each of these embedding techniques can be used to visualise the higher dimensional embeddings. Google's Tensor board shows an exhaustive visualisation of all the three techniques over the entire vocabulary of google's pertained word2vec model.



(a) PCA (b) t-SNE

(c) UMAP

FIGURE 3.9: Tensorboard Embedding projector

- **Self Organising Maps[27]:**An unsupervised machine learning algorithm based on neural networks, where competitive neuron based learning updates the weights. A grid of randomly initialised neutrons is selected. For any review given, among all the neurons, the Best matching unit (BMU) is selected, and all the neurons closer to the BMU under a particular bound radius have their weights updated. The procedure tends to map the relative topological features of the data revealing any clusters present in the data. The map is set to train for a large number of iterations and neurons are regularly updated based on the incoming reviews. The final map obtained is visualised using a heat map where the similar colour presence indicates similarity or clusters while variation in colours denotes a decision boundary.

  Self-organising maps are an excellent way visualising underlying clusters, while also performing the dimensionality reduction, unlike the t-SNE geographical structure of the map reveal the clusters and decision boundaries among individual classes. In recent paper [1], researchers in the medicine domain applied SOM over document classification of papers published in the medical domain.

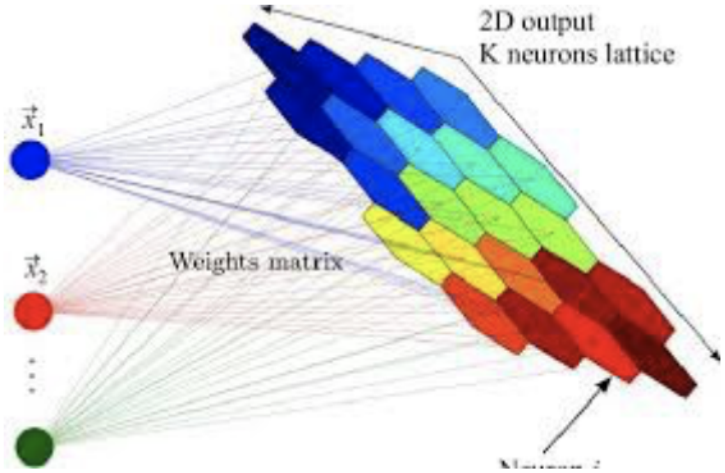


FIGURE 3.10: Color visualization of U-Matix training

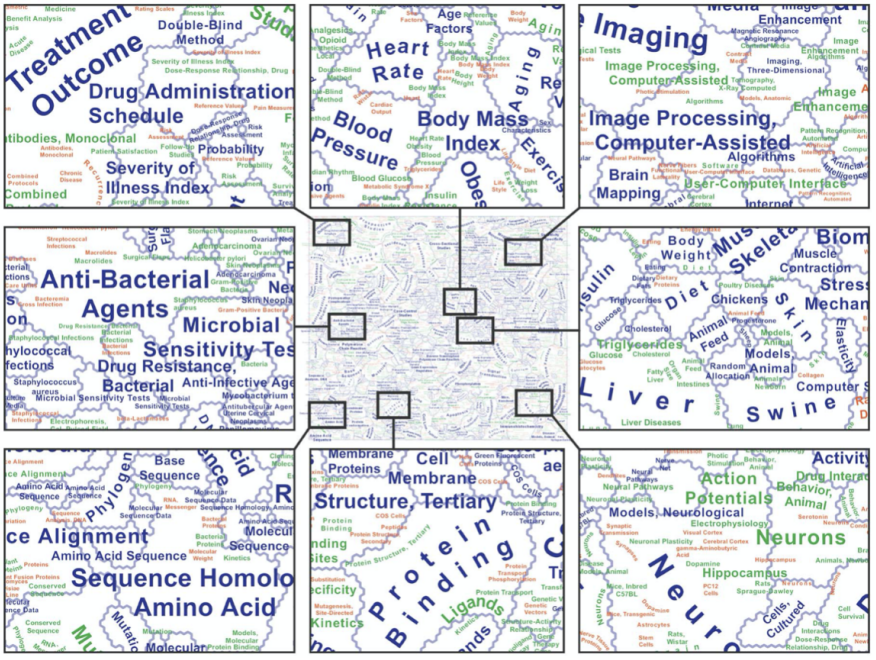


FIGURE 3.11: An image of self-organizing maps trained large medical document corpus [1]

3.5.4 Classification Algorithms:

- **Nearest neighbourhood-based approach [28]:** We select K nearest neighbours to the query point based on a set distance metric. The label of the query point is assigned with maximum majority class label, or a weighted majority where the weights assigned are inversely proportional to the distance, meaning closer

point to the query points are offered higher weight, and farther points are offered less weight.

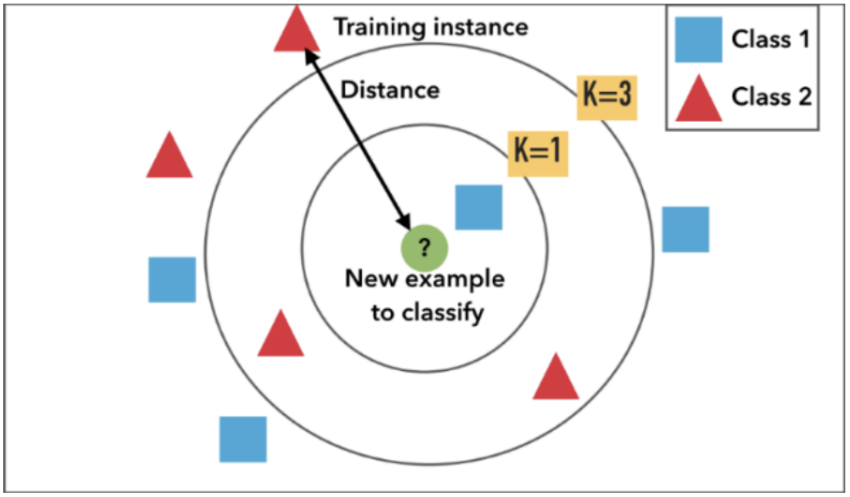


FIGURE 3.12: Majority Voting in KNN

- **Naive Bayes (Probabilistic approach) [29]:** The probability of class given a particular review (Posterior) is calculated based upon the prior evidence and likelihood probabilities. Appropriate class weights are assigned to compensate for the class imbalance.

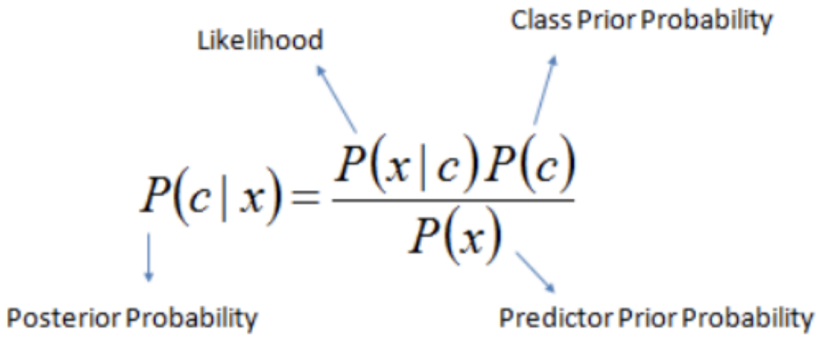


FIGURE 3.13: Posterior probability of class given word embedding

- **Logistic regression:** In logistic regression, we try to find linearly separable high dimensional Hyperplanes capable of differentiating the class labels. The equation of the plane is defined as Weights transpose X + b, where b is the intercept term while W is the coefficients associated with all the variables.

  For a perfectly linear separable decision boundary, all the points on one side of plane should result in the positive value while all the points on the other side of

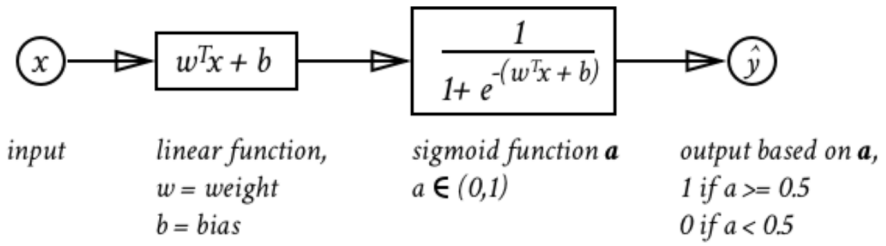


FIGURE 3.14: application of sigmoid function to determine the class label

the plane should yield negative values as shown in the figure below. Multi-class text classification can also be performed by using a "One Vs Rest" strategy.

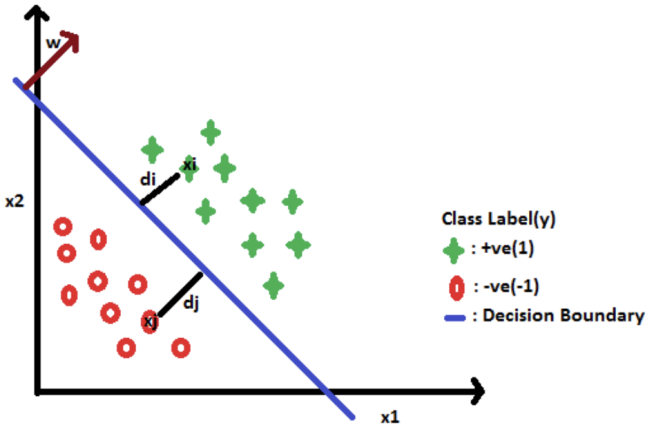
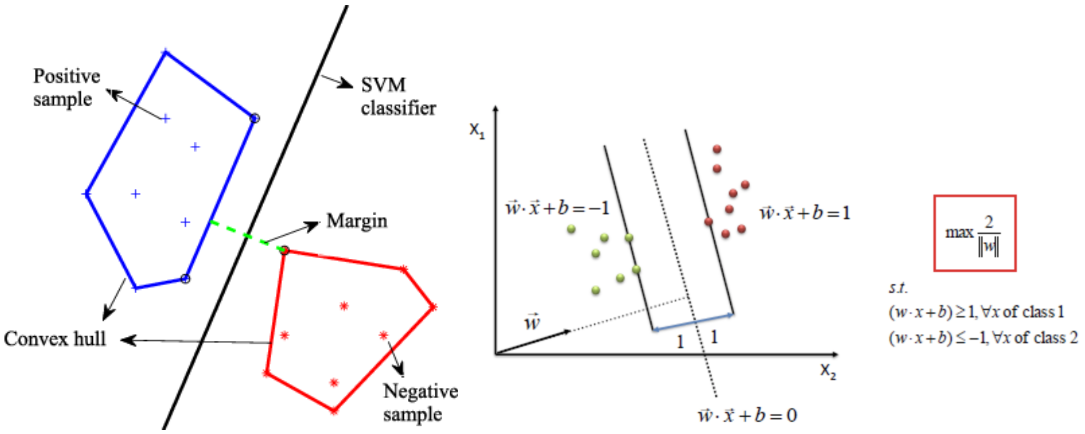


FIGURE 3.15: Logistic regression for an ideal classifier

A differentiable activation function such as a Sigmoid squashing function is applied to scale the values into probabilistic range. Application of the logarithmic function ensures a monotonic Cost function. Where regularisation term ensures that that values of the weight "W" do not explode causing a very large associated weight from growing too large. This problem is posed as an optimisation problem where we try to minimise the logistic loss function defined as:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log\left(\exp\left(-y_i \left(X_i^T w + c\right)\right) + 1\right) \tag{3.3}$$

- **Support vector machines [30]:**, also known as the large margin classifiers tries to find an optimum hyperplane such that the margin between the classes is maximised. The algorithm computes a large margin with the help of support vectors directly computed over calculating the maximal distanced points of the two convex-hulls possible for each of the classes in the data.



(a) convex hull (b) maximal distance

FIGURE 3.16: Formulation of support vectors

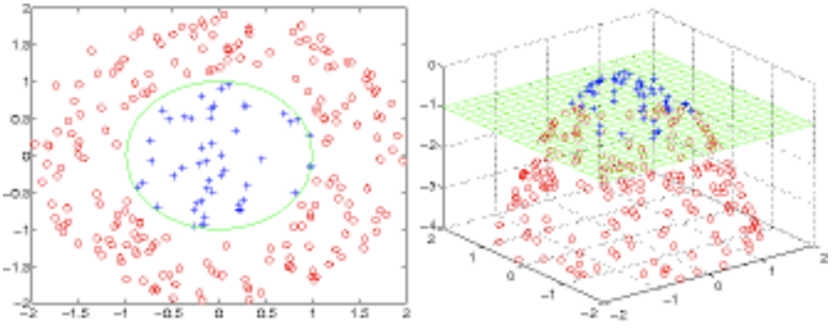
Hence, the Optimisation equation is also called the Primal form of SVM

$$\min_{w} \left\|w^2\right\| + c\sum_{i=1}^{n} \xi_i \tag{3.4}$$

This formulation is also equivalent to the second formulation [30] as defined by Vladimir Vapnik.

$$y_i \left(w^T x_i + b\right) \geq 1 - \xi_i \text{ for } i = 1, \ldots, n \tag{3.5}$$

The last term in the SVM denotes the Kernel function such that changing the kernel function allows fitting non-linear decision boundaries. The idea of non-linear decision boundaries when applied in higher dimensions results in a much more complex decision boundary; this is known as the kernel trick.



(a) Input Space (b) Projected Space

FIGURE 3.17: Kernel trick for visualizing in high dimensional space

- **Decision tree classifier [31]:** A tree-based structure with multiple if-else conditions at each stage denoting the class labels for each feature. The more depth of the decision tree designates the overfitting of the tree. Generally, tree's are meant to be of shallow length.

  Features that are to be selected as a node are based upon the highest entropy values, such that feature with the highest Entropy is selected which yields in the maximal split across both categories.

  The computations are performed over Gini impurity as opposition to Entropy avoiding the logarithmic term which massively boosts the train time complexity.

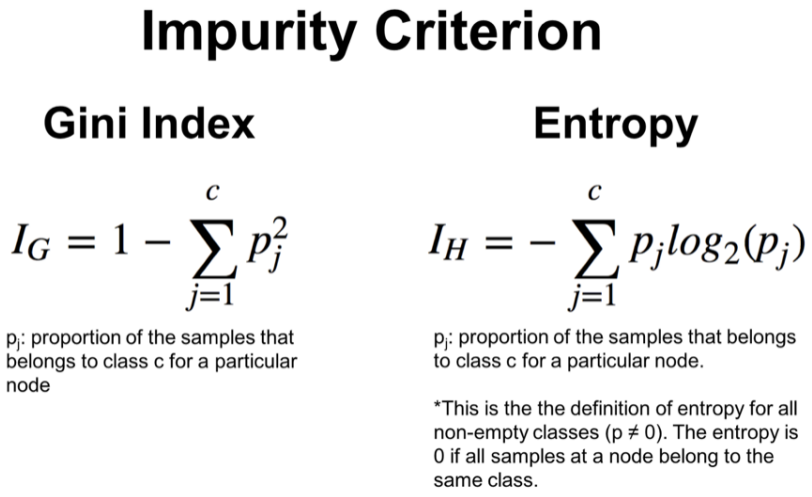


FIGURE 3.18: How to decide which column to split

- **Bagging (Random Forest) [32]:** Random forest is an ensemble technique to counter High variance problem by applying the concept of bagging. The entire dataset is broken into chunks of data such that in each chunk a subspace of columns or rows or both can be selected. A decision tree with large depth is trained upon each of these bags, and a majority vote is created for predicting a new query point.

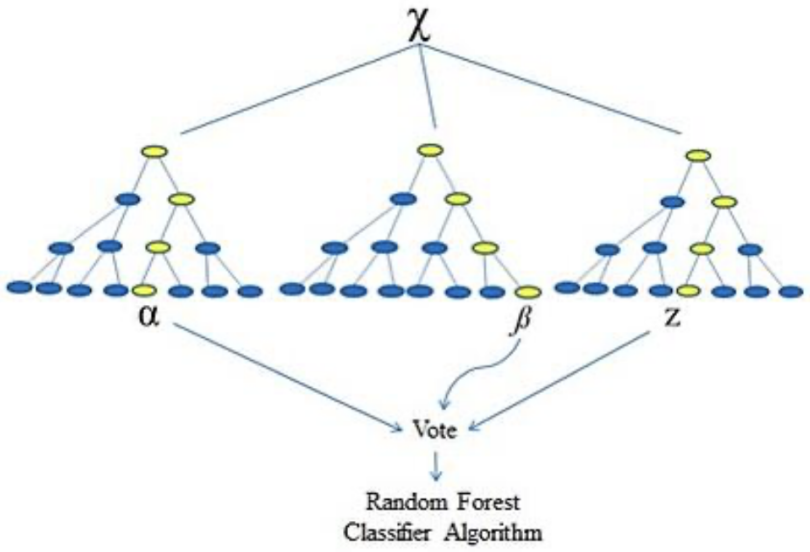


FIGURE 3.19: bagging Large depth and large number of decision Trees to reduce the variance

Randomness in column selection and row selection ensures that the model has low variance because there are multiple trees involved in the majority vote.

- **Boosting algorithms:**In boosting based algorithms, the main objective is to minimise the bias term associated with the errors known as pseudo residuals as opposed to the variance term in the Random Forest. Shallow depth decision trees are trained on errors of each model. The random selection of features ensures a low variance error. The final class label is determined by the weighted majority of all the model trained upon subsequent error terms. Boosting when incorporated with decision trees as base learners is called Gradient boosted decision Trees (GBDT) [33]. Multiple variants of GBDT's with sampling are available such as XGboost which can also be used for linear models [34], Catboost [35] for a mix of categorical and real-valued features, LightGBM [36] a fast optimized version of GBDT's.

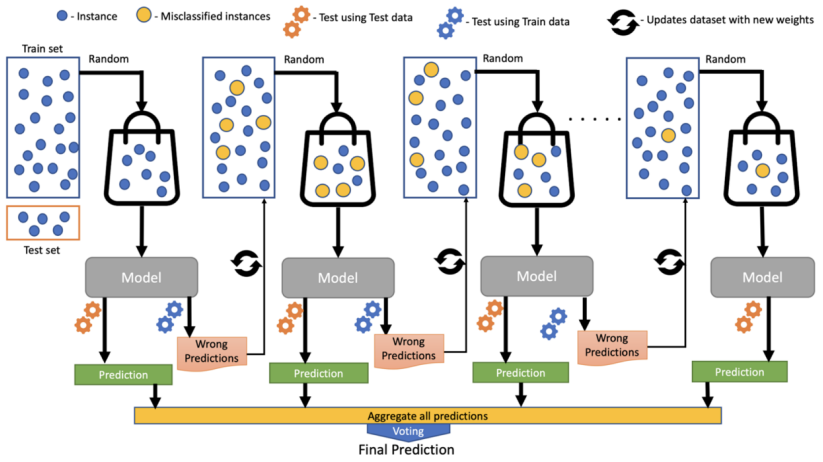


FIGURE 3.20: boosting on the mis-classification error of the previous model

- **Deep Neural networks [37]:** A deep neural network models are capable of learning complex, high dimensional non-linear, decision boundaries. Where a set of hidden layers are selected with each having a certain number of Neurons also called as nodes, each of these layers is connected with a weight where the output of the next layer is equal to the weighted sum of neutron outputs from the previous layer. A softmax layer is applied in the end to interpret the probabilistic class labels. Initially, the weights are randomly initialised while the after every batch is pushed this weight is updated via error correction through gradient flow. Thus it is mandatory for the neural network to have a differentiable loss function and activation function such as sigmoid,relu,tanh etc. where the derivatives can flow through the backpropagation, and weight updates can take place after every epoch.

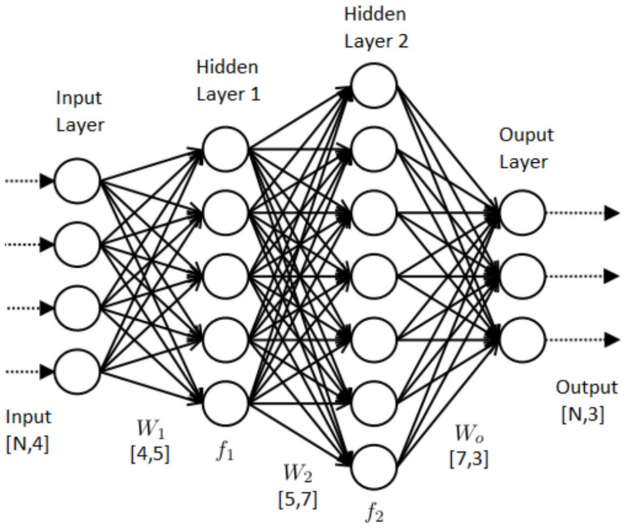


FIGURE 3.21: A two layered shallow neural network

- **Long short term memory:**LSTM's are a special type of RNN's meant to resolve temporal long term dependencies as the gradients decay exponentially with the increase of the sequence information as the word length Increases. In the ANN architecture, the weight update happens through backpropagation mechanism, and the gradients flow through the chain rule causing the early layers to update their weights at a much slower rate when compared to the later layers in the network. This problem is relevant in vanilla RNN's while LSTM block architecture as mentioned in 2.7 is meant to resemble and retain the information for longer and deeper networks where it is extremely crucial to avoid the gradient vanishing problem. Each LSTM cell is iteratively repeated for the classifier to capture unidirectional sequence information. A comparable but lower train time complexity architecture **(GRU's)** performs similar to LSTM's in a variety of classification tasks, one such evidence is music classification comparison. [38].

- **Transformer based models(BERT):** BERT [15] is a transformer-based model consisting of stacks of 12 Encoder, Decoder models (see figure 2.6 for BERT) where each of the encoder block itself comprises of an Attention model and a feed-forward neural network. BERT (base) model comprises of a total of 110 Million parameters. The bidirectional attention mechanism ensures learning for larger sentences with context word refereed in both the axis of the sentence meaning every word context is related to every other word a confusion matrix can be drawn denoting every word resemblance on the heat map.

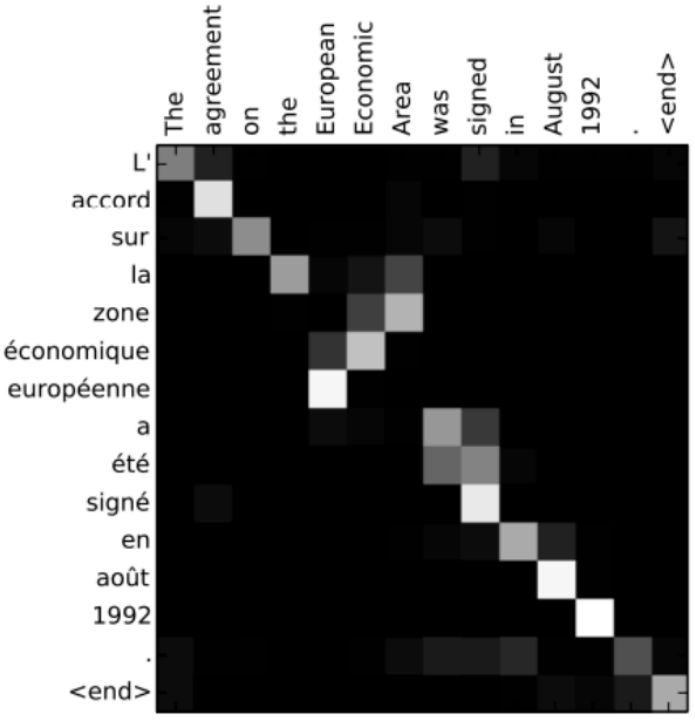


FIGURE 3.22: heatmap of correlated english to french words

3.5.5 Imbalanced Data

Classifiers tend to bias to the learning towards the majority class in the dataset hence performing poorly over minority class. To handle such kind of class Imbalance problem, there are several proposed solutions:

1. **Weight balancing**: Balancing the class imbalance by assigning greater proportionality weight to the minority class.
2. **Under Sampling:** In this technique, a subset of points from the majority class, are removed in order to maintain the class balance. This approach also reduces the useful information to a tradeoff with classifier performance and hence in practice is not much preferred.
3. **Naive Over Sampling:** We try to randomly replicate the entries in the minority class such that the learning of the classifier is incremental and is not overshadowed by the sheer abundance of the majority class.
4. **Synthetic Minority Over-sampling Technique (SMOTE)** : It is a type of oversampling technique which allows us to create synthetic data for the minority class based on the data that already exists. New data points are picked up from the closest hyperplanes that exist in a neighbourhood function defined by a KNN algorithm.

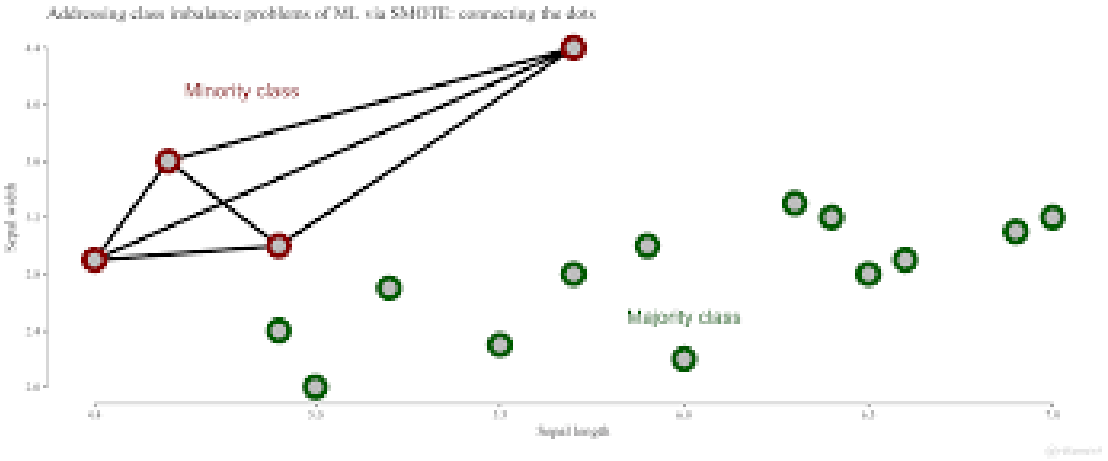
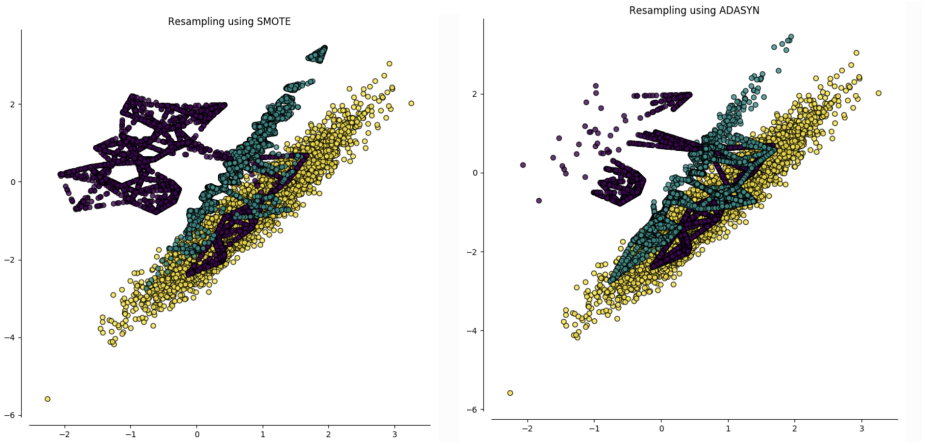


FIGURE 3.23: SMOTE generated samples using KNN

For the SMOTE approach [39] samples are generated on the nearest neighbourhood of the minority class, Whereas Adaptive Synthetic(ADASYN) [40] focuses on generating samples close to the wrongly classified points by the KNN algorithm.



(a) SMOTE (b) ADASYN

FIGURE 3.24: SMOTE Vs ADASYN based generations [2]

To reduce class imbalance, a hybrid approach of oversampling and undersampling can be applied such as SMOTEEN based on edited nearest neighbours [41] and SMOTETomek based on Tomek links [42].

Chapter 4

Implementation

This chapter discusses the implementation of the architecture discussed in the Methodology Chapter (Chapter 3). Section 4.1 discusses the data set, features involved etc. Section 4.2 explains about the Exploratory Data Analysis (EDA) and feature engineering. Section 4.3 navigates through the entire process of implementing the models along with the features.

4.1 Dataset introduction and EDA

For the analysis and the scope of the project, we have decided to choose the publicly available Amazon fine food reviews dataset [7] which in our opinion resembles the most with the context of the problem, The large Size (568,454 reviews) of the data set allows us to experiment the tradeoffs of each classifier in greater depths and less overfitting in the optimal hyperparameter tuning for building Ensembles.

The Dataset itself contains many closely related reviews with similar word usages however opposite polarity which each of the classifiers has to categorise correctly. This can be well explained with the word cloud approach and the bar-graph plot of the frequently occurring bi gram words in the data.

There are several crucial aspects of the Dataset (after removing the duplicate entries in the data):

- A high-class imbalance: to reduce the class imbalance and the overlap of many ambiguous words present in the reviews, We will discard all the 3-star reviews (see figure 4.4).

FIGURE 4.1: Segregating positive and negative reviews based on word clouds

- The use of many removal un-recognised characters (see figure 4.5) such as HTML links, tags which do not serve any relevant information in determining class labels should be removed while preserving emoji's that may be crucial for determining the polarity of the sentence[43].

FIGURE 4.2: Distribution of positive words

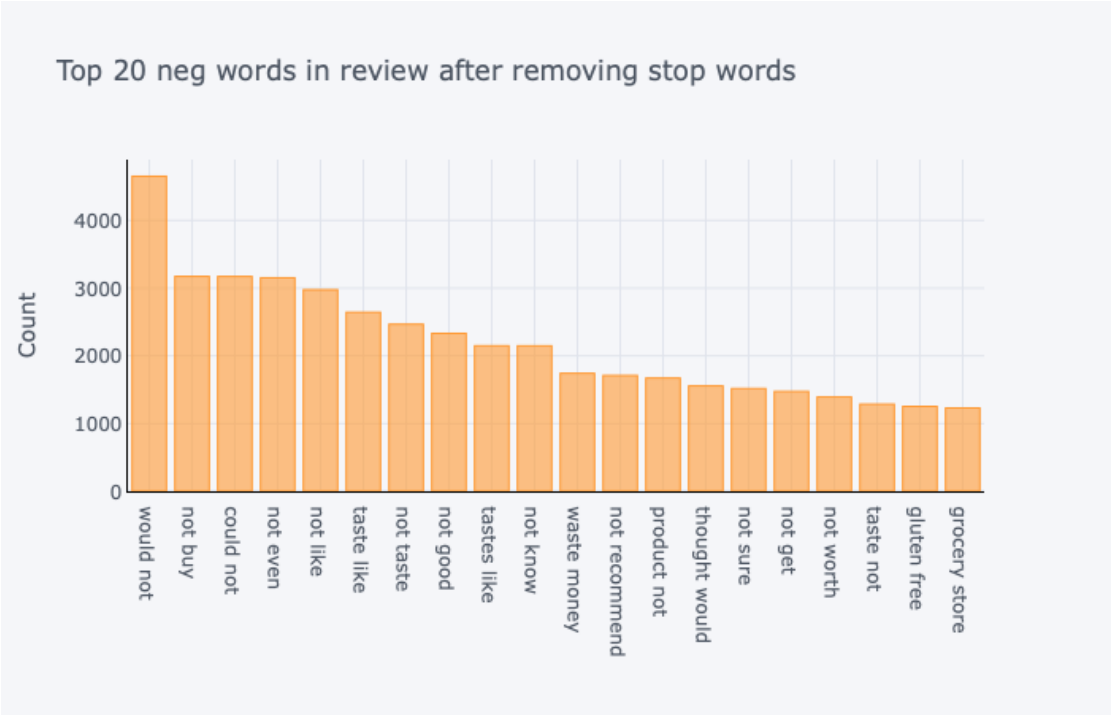


FIGURE 4.3: Distribution of negative words

4.2 Handling Imbalance:

For handling the imbalance in the data, we will follow the following methods:

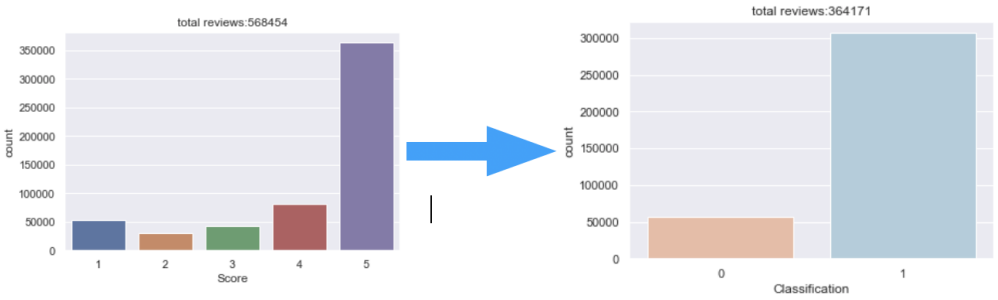


FIGURE 4.4: categorized reviews

Original reviews present in the data	Label
As someone who puts Habañero Tabasco® sauce on his eggs for breakfast, I don't find Tiger Sauce to be *hot* at all. But it is a very flavorful sauce with a soupçon of kick that works for a number of meals. I can't think of anything I haven't used it in. Beans. Chicken. Beef. Pork. Eggs. Tuna. Salmon. I haven't tried drinking it from the bottle. Yet. But trust me, this is THE one sauce to add to your kitchen.	Positive
Newman's Own®Organics Dog Treats for Small Size Dogs, Peanut Butter, 10-Ounce Bags (Pack of 6)/ My Pom would not eat one, he would live on Treats, if I let him. I only give him the best, like the above, but he did not go for this.	Negative
This product can be used for so many things. I use it for my sons (7months) hair and skin because he has mild eczema. I also use it for my hair as a hot oil treatment , I use it to seal in my oils when I moisturize and seal nightly. I love how it feels on my skin after I shave my legs. This is one of my favorite products and I will be purchasing again. 👍 😍	Positive

FIGURE 4.5: Special Character reviews

1. While splitting the data during train(80%) test(20%) split we will use stratify option as true. So, that the imbalance class is equally divided between the test and training set.

2. Use of class_weights for incremental learning present in the model parameters and check if varying the values makes any impact on our prediction.

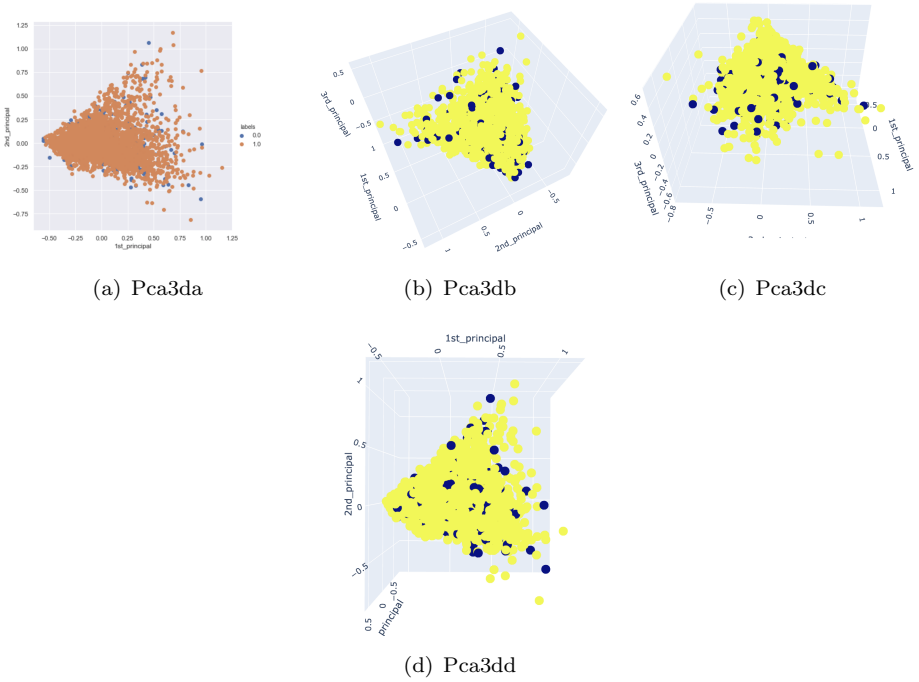
3. Using SMOTE based over-sampling techniques on the training data set and validate our model's performance. Do they benefit us or not?

4.3 Visualization based on dimensionality reduction algorithms

We begin by randomly sampling 0.01% with a random seed of 1 for reproducible results for all the visualisation algorithms implemented in this section.

4.3.0.1 Principal Component Analysis:

PCA works by applying an orthogonal linear transformation on the data shifting the coordinate axis to the axis where the variance is maximised.We ran the algorithm for the total number of features, i.e. 300 and then we select the top diagonal Eigenvalues corresponding to the most significant PCA axis. These axes are then plotted in a 2D/3D plot to visualise any underlying decision boundaries.



(a) Pca3da (b) Pca3db (c) Pca3dc

(d) Pca3dd

FIGURE 4.6: PCA Embeddings

Since we are visualising three PCA axis and we did not see any emerging pattern or a proper decision boundary, This is due to the fact that three-axis visualisation was not able to preserve the entire cumulative variance of the data. The below graph explains the energy distribution remaining when the data has its dimensionality reduced to lower dimensions. With three dimensions, we are able to preserve less than 0.2 variance or equivalent to 20% energy from the data. This is the reason why our data points do not show a clear separation boundary.

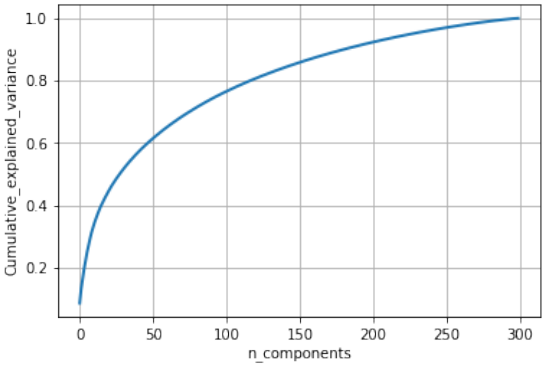
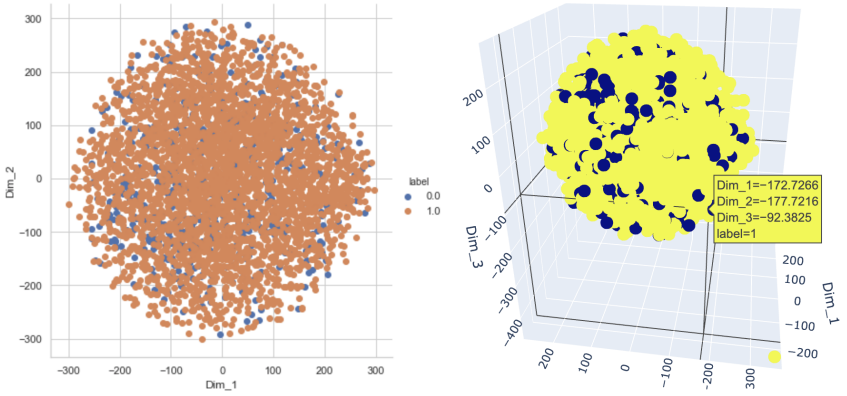


FIGURE 4.7: PCA cumulative variance distribution

4.3.1 Tsne

A dimensionality reduction technique that works well for visualising higher dimensional data.



(a) Tsne2d (b) Tsne3d

FIGURE 4.8: Tsne plots

Hyperparameters:

- Perplexity: Perplexity is related to the number of neighbours in the Neighbourhood; larger values indicate a bigger neighbourhood.
- Iterations: indicate the total number of runs of the algorithm to be sure of the plots produced. Significant iterations indicate the plots converged optimally.

We manually tried to tweak the perplexity value for set significant learning rate, while the data seems to circumcise revealing no particular distinguishable, separable cluster. The two possible reason why tsne plot could not separate cluster-based information-

Sentence embedding is made from a weighted sum of word embeddings. This means of the same or similar meaning words that are present in the sentence, and our embeddings result in closer; hence indistinguishable reviews thus, there is considerable overlap—this problem is inherent to the word usage and complex grammar usage of language. Changing the perplexity didn't promise a revealing cluster in our experiments. Hence we conclude that the data is inseparable using Tsne.

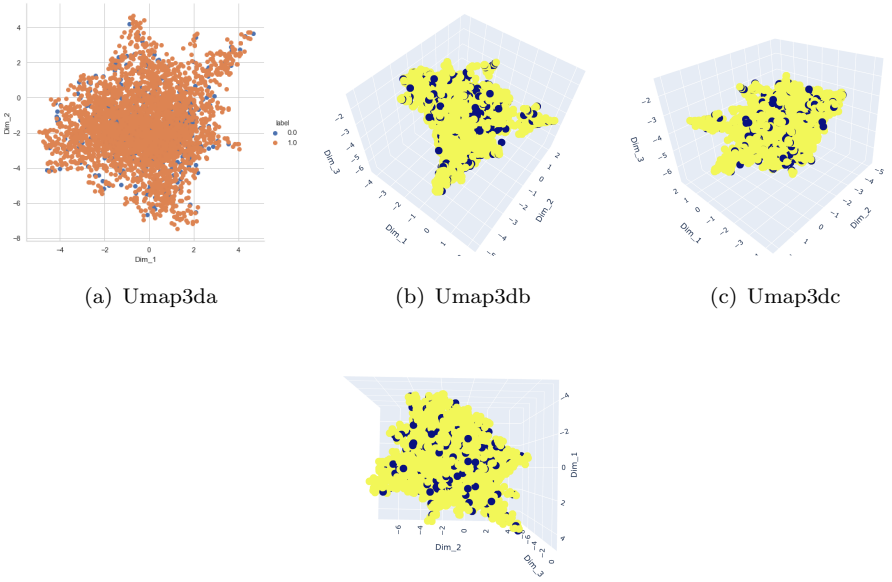
Tsne does not guarantee successfully mapping for higher-dimensional data into low dimension based on neighbourhood information as evidenced in the paper as a crowding problem.

4.3.1.1 UMAP

UMAP is the current state of the art technique used for not just visualisation but also for dimensionality reduction to avoid the curse of dimensionality. Since these embeddings itself are a compressed vector representation of long sentences where the output is extracted form of hidden 300 nodes in the autoencoder based systems. We Implement UMAP for visualisation purposes only.

Hyperparameters:

- n_neighbour: UMAP decides to make a tradeoff between the local structure of the data and the global structure of the data on its Hyperparameters. Which is further decided by the number of neighbours. Large neighbour size corresponds to the global structure, while a smaller value of neighbours indicates a focus on lower-dimensional data.
- min_dist: Can be useful for visualising clustering points. Substantial value denotes the structure is moved to more massive clusters if dealing with large clusters.



(a) Umap3da (b) Umap3db (c) Umap3dc

(d) Umap3dd

FIGURE 4.9: UMAP Embeddings

In the plots generated above, we can see a considerable overlap between the reviews indicating the overlapping distribution of points across the 3 UMAP dimensions.

These probability density functions are better highlighted with the pair plot access across each of the three dimensions.

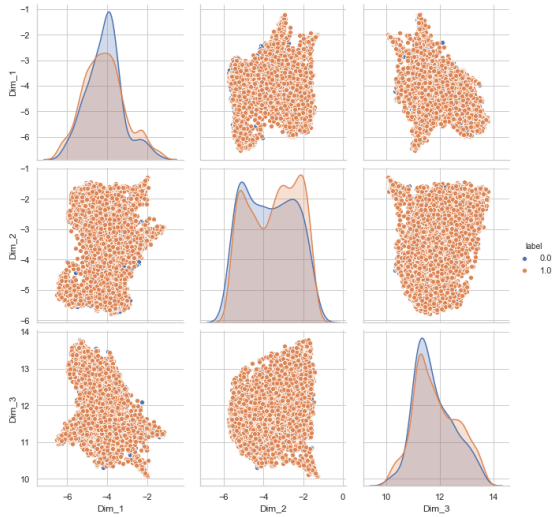


FIGURE 4.10: Probability density function of UMAP reduced features

4.3.2 Self organising maps:

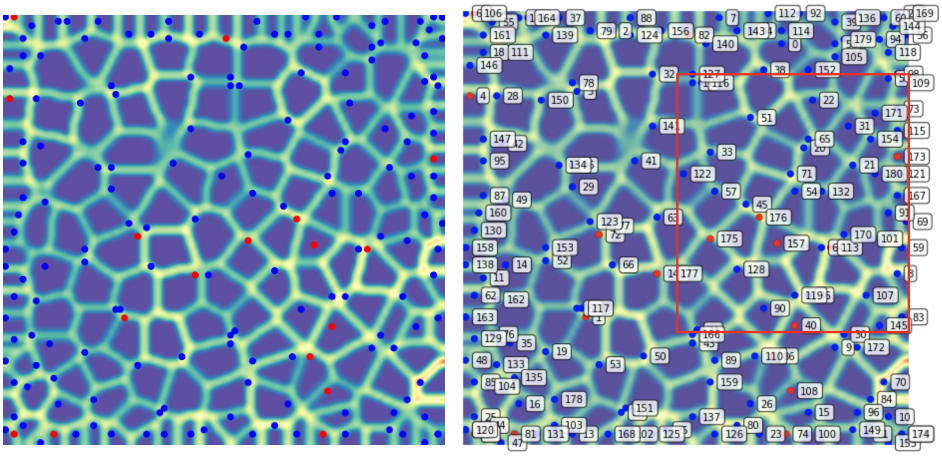
A neural network-based Dimensionality reduction technique where the data input from the reviews preserves the topological structure. This structure when visualised under a heat map reveals groups similar points together preserving the global symmetry. In Tsne and map the distances and cluster size do not mean anything while in self organising map, geographical topology is reserved. For a clear 2d view, we only take 181 random points to visualise the data.

Hyperparameters:

- Size of the map: Denotes the number of neurons to put in a grid of SOM map.
- distance_metric: The type of distance metric that we want to apply, specifically we used euclidean
- Neighbourhood: A Gaussian distribution for determining sigma value influencing nearby neurons to adjust the weight.

$$\Theta(t) = \exp\left(-\frac{dist^2}{2\sigma^2(t)}\right) \tag{4.1}$$

- iterations: Number of unsupervised iterations for the map to learn the geometric projection of the data.



(a) Som1 (b) Som2

FIGURE 4.11: Reviews with tags projected on a self organising map on the right

Let us look at a much broader section of the map and try to relate a few reviews.

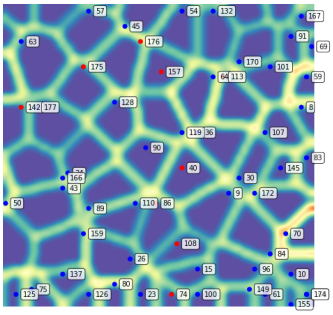


FIGURE 4.12: SOM zoomed for on a subset of map

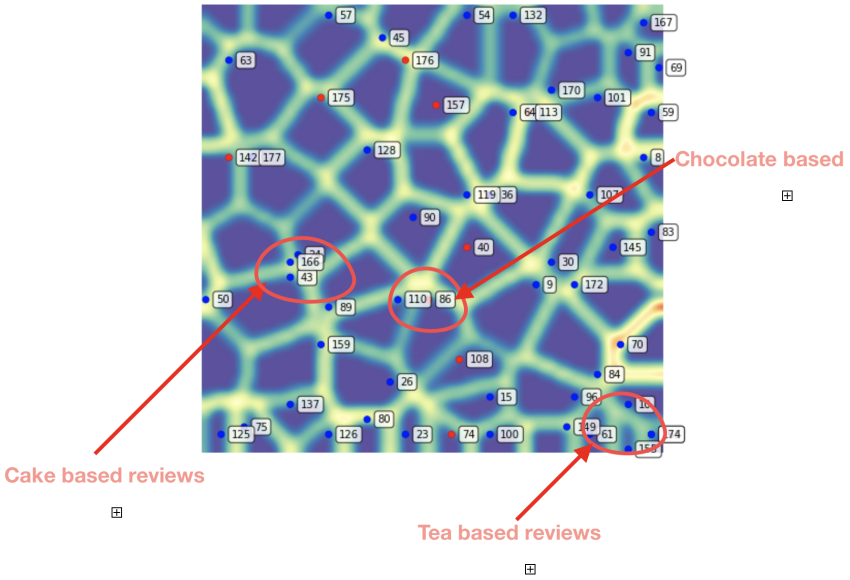


FIGURE 4.13: Partitioning Review clusters on a feature reduced map

In order to visualise the U-matrix of Self organising maps we can visualise the surface area of U- matrix over a heat map exposing any formation of large clusters which apparently does not exist in the data, please refer 4.13

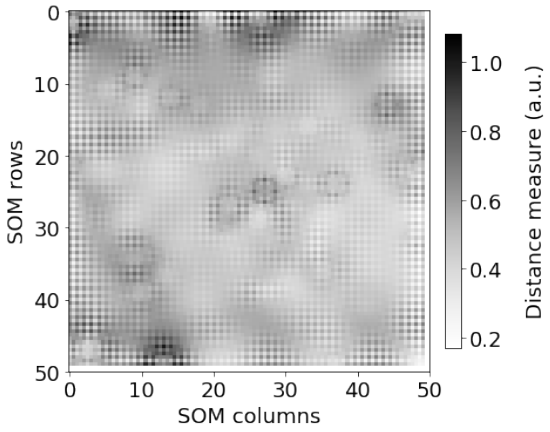


FIGURE 4.14: U-Matrix after 10,000 iterations

4.4 Classification based algorithms:

4.4.1 KNN

We select the nearest neighbour approach by employing a weighted KNN, ensuring that the points nearest to the query point are given more weightage. The weights associated are inversely proportional to the distance from the query point.

Hyperparameters:

- Neighbours: Selected odd Number of neighbours to avoid clashes in the majority vote
- distance: applied both uniform and distance weighted KNN
- Distance metric: Experimented with a wide variety of distance metric including the novel Hassant distance and found Hassanat distance to report the highest accuracy, However, the Use of Hassanat distance had severe time constraints imposed on the algorithms. Hence we will discard it for searching the right hyperparameters. The performance metrics for various types of distance can be seen in figure 4.15

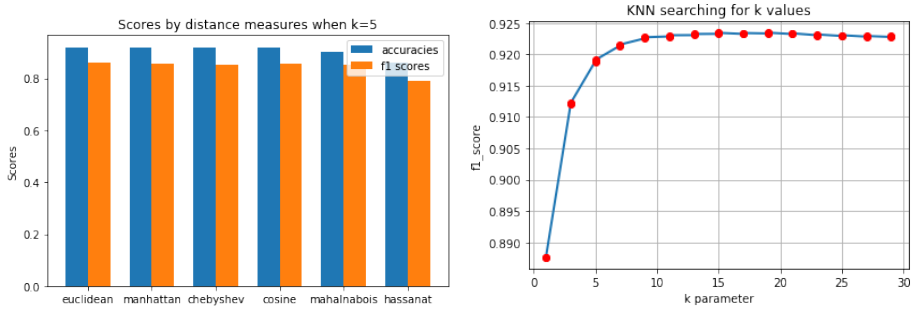
$$HassantDistance\left(A_i, B_i\right) = \begin{cases} 1 - \frac{1+\min(A_i B_i)}{1+\max(A_i B_i)} \\ 1 - \frac{1+\min(A_i B_i)+|min(A_i B_i)|}{1+\max(A_i B_i)+|\min(A_i B_i)|} \end{cases}, \min\left(A_i, B_i\right) \geq 0$$

$$EuclideanDistance(a, b) = \sqrt{\sum_{i=1}^{n} \left(b_i - a_i\right)^2} \tag{4.3}$$

We implemented GridSearch to find the optimal K values ranging from the 1 to 30 in odd values in both distance and uniform weighted KNN over a variety of distance metrics such as Jaccard, Manhattan, Mahalanobis, Hassanat, Cosine, Chebyshev and Euclidean. We restricted our search to find the top-performing similarity metric based upon the paper[44], where the authors have compared Hassanat metric with a significant variety of different other similarity metrics.

Euclidean gave the highest accuracy after Hassanat distance on a subsample of 0.01% training points. We remove training KNN on Hassanat distance due to considerable training time.

The confusion matrix can be seen below after tuning the KNN algorithm and results can be seen in the Confusion matrix in figure A.1.



(a) Distance metric evaluation over 0.01% subsample

(b) K parameter

FIGURE 4.15: Tuning KNN parameters

4.4.2 Naive Bayes

Implementation of Naive Bayes over word embeddings would require Complement Naive Bayes as there is a considerable class imbalance. Class weights are associated with avoiding the biasing over the majority class.

Hyperparameters:

- Alphas(Laplace smoothing parameter): To void numerical underflow in python and to zero probability if the feature is close to zero we employ Laplace smoothing parameter called alpha. Smoothing parameter is GridSearched upon values ranging from 1 to 5 with a 0.1 interval.

$$p_{i,\alpha-\text{ smoothed}} = \frac{x_i + \alpha}{N + \alpha d} \tag{4.4}$$

- Class Weights: Prior class weights are inversely proportional to the categories of labels co-occurring int he Dataset

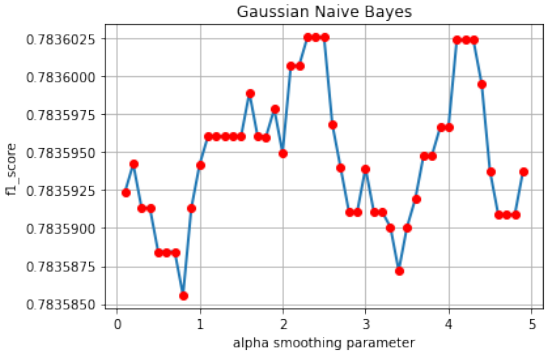


FIGURE 4.16: Tuning Alpha parameter Complement Naive bayes

4.4.3 Logistic regression:

We try to separate the embeddings based on a linearly separable hyperplane based on the optimal weight values obtained through logistic loss converging over a large number of set iterations.

Logistic loss

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{4.5}$$

Hyper parameters:

- C value: inversely proportional to the regularisation strength allowing the weights from exploding. A range of C lying between 10^-2 and 10^2 is selected from a lognormal distribution.
- Type of regularisation: a variety of Regularisation schemes such as L1, L2 and a combination of both also known as elastic net are employed. denoted as the power of the Theta term Class weights: To compensate for the minority class, appropriate class weights are assigned to the minority class.
- Class weights: To compensate for the minority class, appropriate class weights are assigned to the minority class.
- For better optimisation and searching over ample space we implemented Tree parson based estimator along with side GPU solver for logistic regression.

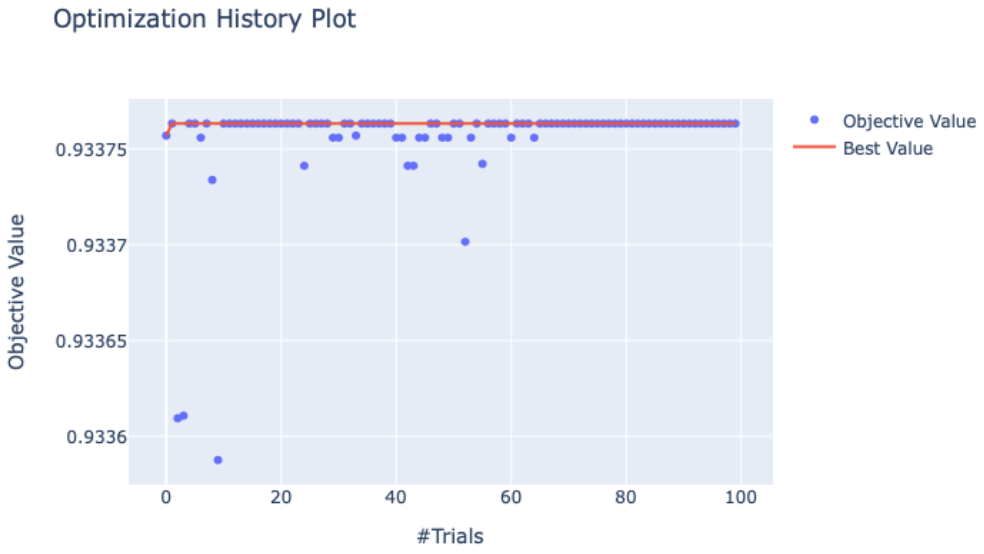


FIGURE 4.17: Tuning regularization strength

4.4.4 Support Vector machines

We try to minimise the Hinge loss over a variety of Kernel-based function, ensuring linear/non-linear decision boundaries.

Hyper parameters:

- Kernel function: The mapping function we want to apply for the decision boundary. A search space including Sigmoid, Linear(Large Margin), RBF (Radial basis function)
- C value: denotes the regularisation strength we want to employ
- Class weights: To compensate for the minority class, appropriate class weights are assigned to the minority class.

Disclaimer: No results were obtained on SVMs due to large size of the data

4.4.5 Decision Tree

A decision tree is trained based upon the Shanon Entropy where the features are picked up as nodes, accounting the maximum Information gain. For fast computation, we base our decision tree on Gini impurity rather than Shanon Entropy, which shows similar behaviour.

Hyper parameters:

– Largest depth of the tree: we limit our decision tree depth to be limited up to 5 depth, for interpret-ability.

– Class weights: appropriate class weights are assigned in favour of the minority class.

Tree generated by the classifier structure:

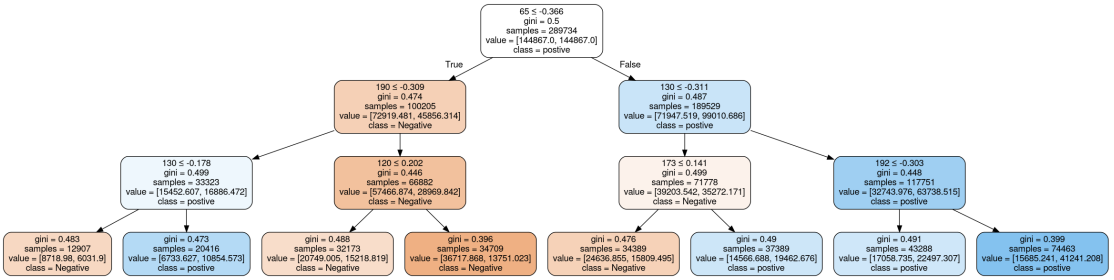


FIGURE 4.18: Shallow depth decision tree

4.4.6 Random Forest

We leverage the concept of Bagging to the decision tree where each of the classifiers is a significant depth decision tree grown without any limit. Column sampling employed, along with row sampling, ensures variation in learning in each of the enormous depth trees. At the aggregation stage, the majority vote from each of the tree is calculated.

Hyperparameters:

– number of trees: Number of decision trees to aggregate, selected ranging from 150 to 250 over ten gaps interval.

– Class weights: Calculated based upon the overall class weight or individual class of subsampled trees or base learners in the Random forest.

– Sampling parameters: Defining the column sampling and row sampling ratio.

4.4.7 Gradient boosted Decision trees(XGboost)

Incremental learning is based upon training the models on the errors of the previous base learner. These base learners can be linear models such as logistic regression or simple decision trees. The learning is based on Pseudo residual calculations over each base learner.

Hyperparameters:

- max_depth: The max depth of the decision tree, each classifier is trained. We limit this to the depth ignore to avoid overfitting.
- Regularisation strength: The tradeoff between overfitting and underfitting.
- Learning rate: For gradients to converge we employ a learning rate of 10^-5
- booster: Type of base learner, we employed both linear and Decision tree-based.

4.4.8 Artificial Neural network-based model

A large scale network comprising of multiple hidden nodes and layers is trained. Batch Normalisation is employed after every three layers along with a Dropout percentage after each layer to ensure robust learning. A batch size of 256 is kept with 60 number of epochs. Employing early stopping would ensure that we don't overfit on the validation data. Weight initialisation schemes have such as "Glorot Normal" and "HE" initialisation have been employed [45] for faster convergence. We also use ADAM optimiser[46] as opposed to SGD, which combines the momentum term along with an adaptive learning rate from NAG optimizer.

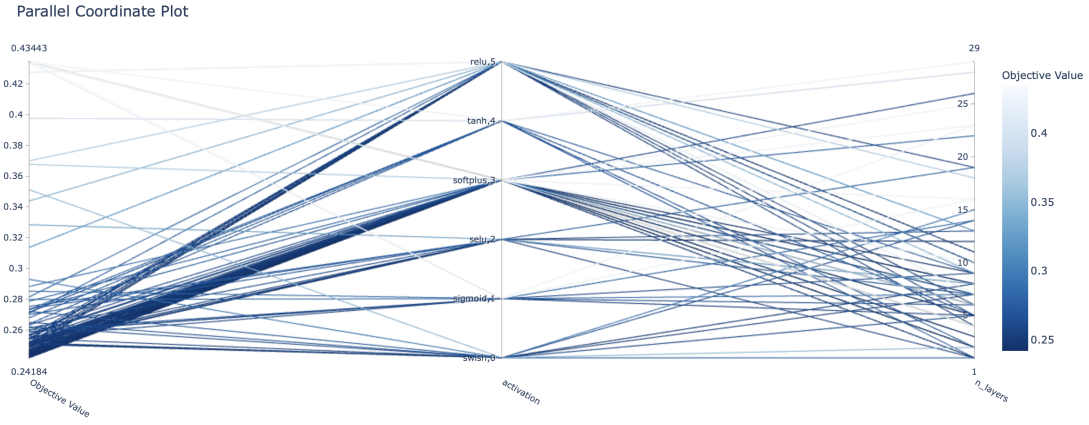
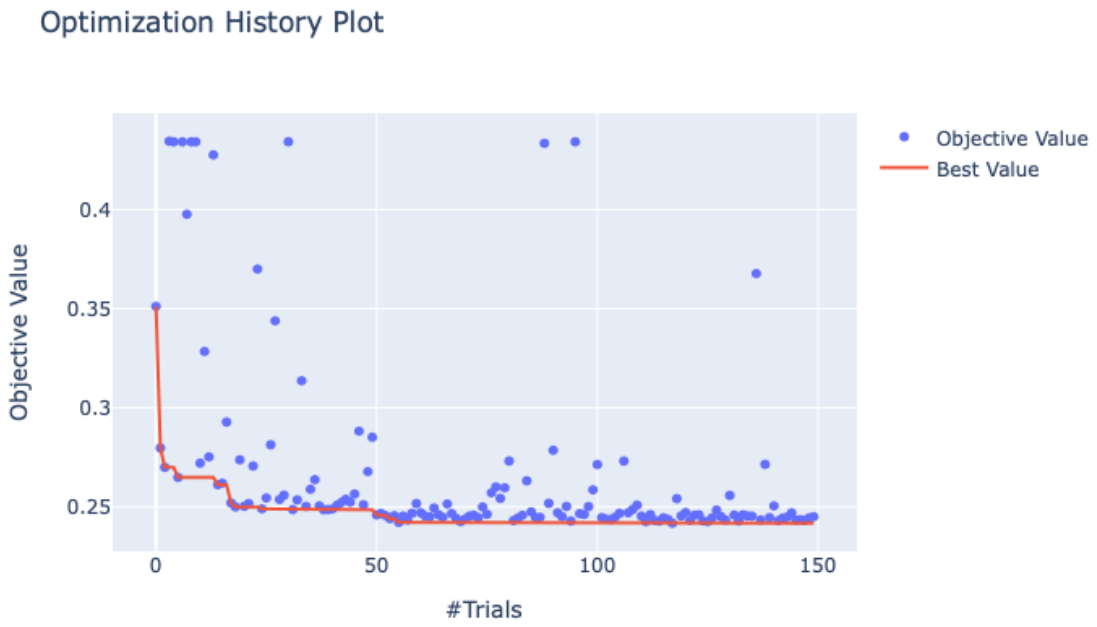


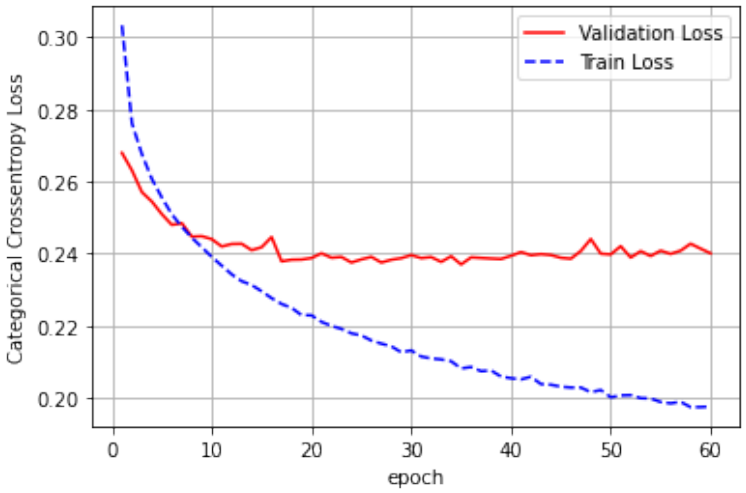
FIGURE 4.19: Hyperparameter search over ANN using Optuna

Hyperparameters:

- Number of layers: Determined by a TPE estimator, Too large can cause gradient vanishing problem, while too small can cause not learning a healthy decision boundary. We experiment our range of layers from 1 hidden layer to 40
- Number of neurons in each layer: More hidden neuron incorporate learning more complex decision boundary



(a) optimization history of Loss function / neural network Hyperparameter selection



(b) overfitting and the need of early stopping evaluating on loss

FIGURE 4.20: Neural network based best parameter selection

– Activation units: A wide variety of activation units ranging from the Relu, Sigmoid, Tanh, soft plus and finally Swish. Selecting underperforming activation units can cause dead neurons in the network, as evident in most of the sigmoid based neural networks. The final activation for the best network was found to be softplus activation unit.

$$softplus(y) = \ln(1 + e^x) \tag{4.6}$$

– Dropout rate: For robust learning, certain neurons after each epoch are randomly turned off. This rate ensures that the neural network is trained immune to overfitting.

– Epochs: The large epoch size means converging to the global minima, this has been kept constant to evaluate each of the networks only the top 10 networks with the lowest loss are further grown with more number of epochs while all other architectures are grown with 15 epochs

– Batch- normalisation: to avoid covariance shift for the deeper network we employ batch normalisation after every three layers

– Batch size: A random training subsample is pushed, and the gradients are calculated based on this subsample rather than the entire individual sample.

4.4.9 LSTM / GRU based model

LSTM's capture important essence of the data as there is a lot of sequence information present in words. The usage of context word depends upon the words used before it. In order to capture the sequence-based information, we will run a unidirectional LSTM model. The sequence information that has to be preserved is determined by the forget gate in the LSTM model. The block diagram below indicates an LSTM cell. One hundred such LSTM cells are stacked to together until the final output is received at the last LSTM block determining the polarity of the review.

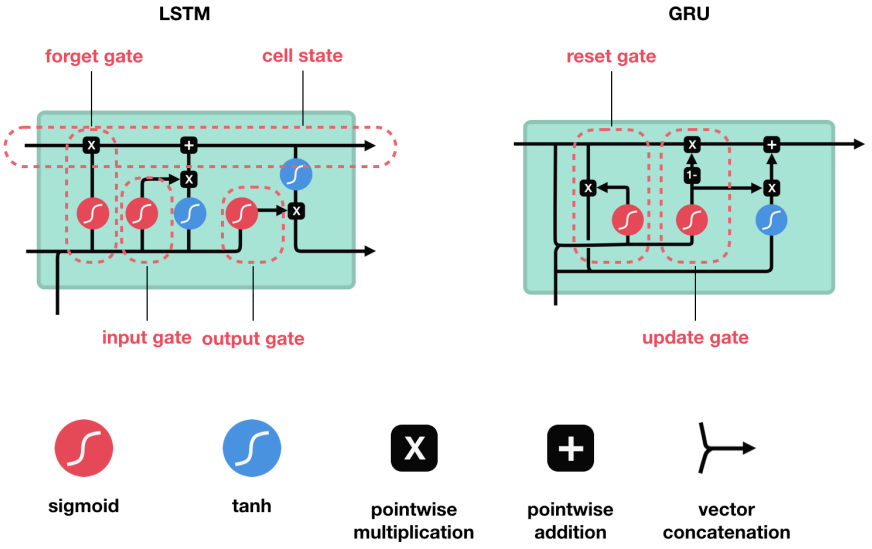


FIGURE 4.21: Comparison of LSTM vs GRU Block

GRU generally speeds up the training time, and for some applications can also boost the performance of the model[47].

In our analysis, we found GRU's to be taking 85 seconds per Epoch while LSTM's take roughly 108 seconds trained on Nvidia Tesla K80 GPU's. For comparison in terms of performance, we found LSTM's to be marginally outperforming GRU based gated structure. For comparison purposes, both the confusion matrix are plotted in appendix section of the report.

Hyperparameters:

- Number of LSTM's block: The total Number of LSTM blocks to stack. Stacking large LSTM models on top of each other does not drastically decrease the performance is in the case of fully connected neural networks(ANN) where gradient vanishing problem causes the gradient to fall close to zero, causing no incremental learning. LSTM's inherently do not face the issue because of the skip connections used.

4.4.10 BERT based transformer model

BERT is based upon transformer based model incorporating stacks of Encoder-Decoder structures are used and are well explained with detailed architecture explained in the chapter 2. BERT is a comes with a variety of different sizes denoting the length of the transformer model comprising of How many stacks of Encoders/Decoders.

Since Training transformer-based models take immense training time even when training on TPU's, we preferred to use BERT over GPT-2 because of the significant performance gain on standard datasets published by Google. BERT also serves a building block to many of the other transformer based models such as XLNet and Roberta where majorly the tunable parameters increase to a much higher magnitude. Hyperparameters:

- BERT model size:(total parameters): BERT base(108M), Large(334M) and Xlarge(1270M).

4.5 How Hyperparameters are evaluated ?

We employ mainly two different types of techniques such as

- **GridSearch with three-fold cross-validation:** IF the search space is not larger and is the order of 10^5 seconds we perform a brute force search over all the possible Hyperparameters.

- **Combination of Bayesian Search along with Pruning:** We use a next-generation state of the art distributed optimisation framework known as Optuna. The framework is based on tree-structured Parzen Estimator(TPE) along with pruning strategies to ensure that we don't essentially get stuck in the local minima a common problem amongst Bayesian Search-based methods. The optimisation framework works elegantly in combination with GPU based distributed solvers as in the case of H20 based logistic regression implemented. The Optimiser ensures tuned Hyperparameters in very less number of trials by employing a pruning mechanism along with bayesian search eliminating getting stuck at the local minimas.
- For large search spaces such as regularisation strength, we define the search space under a normal log distribution instead of uniform distribution.

4.6 Challenges:

- **Capturing Emoji Sentiment** First and foremost challenge was effectively dealing with the preprocessing challenges involving keeping the relevant content based information such as emoji's in the review while deleting HTML and many other characters that do not serve any important role in the prediction of the review.
- **Mislabelled review** certain reviews in the Dataset clearly denote negative review sentiment but are labelled as positive.
- **Large training and tuning time**Implementing and tuning large scale algorithms on GPU's takes longer time significantly.
- **Handling Imbalance**: Class Imbalance is a serious issue, and assigning class weights was the only appropriate option as compared to the widely used SMOTE based techniques fail tremendously when generating word embeddings. With a total of only 15% of the points are labelled negative.
- **Handling large Dataset:** some of the classifiers can take a huge amount of time as with the case of SVM's as the time complexity lies between O(n^2) and O(n^3) where n is the size of the row samples in the data.

Chapter 5

Evaluation and Results

In this section, we have discussed all the experiments that are planned in chapter 4 and evaluated the model against them. Along with the experiments, we have also discussed the results of those experiments and the best model for our problem.

5.1 Common Performance Metrics

1. **Confusion Matrix:** It is a summary forecast representation of a classification problem. For each class the number of correct and incorrect predictions is summarized. Figure 5.1 represents a problem of binary classification, as in our case where class 1 represents the positive review and class 0 represents negative review. In the figure 5.1:

   - TN stands for True Negative = Observation is negative,and predicted to be negative.
   - TP stands for True Positive = Observation is positive,and predicted to be positive.
   - FP stands for False Positive = Observation is negative,but predicted as positive.
   - FN stands for False Negative = Observation is positive,but predicted as negative

2. **Accuracy**: Accuracy is defined as the total correct predictions that are made in all classes during classification, i.e. how correctly the results are classified. It is not a good measure for an imbalance class problem as negative class will be in higher proportion and overall accuracy will be very high. The following formula gives precision in equation 5.1

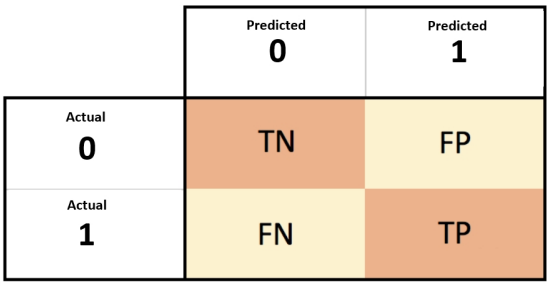


FIGURE 5.1: Confusion Matrix

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

3. **Precision:** It is defined as the ratio of positives correctly classified by the total number of positive classes predicted in the model. It demonstrates how the model is able to identify the classes precisely. High value of precision is always a good indication. The formula for Precision is given in equation 5.2:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.2}$$

4. **Recall:** It is defined as the ratio of the total number of positive classes correctly classified divide by the total number of positive classes. High recall value means the class is recognized correctly. The formula for the recall is given in equation 5.3.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.3}$$

5. **F1 score**: This is both the harmonic measure of Recall and Precision. As both of them are linked, a good choice is to strike a balance between them. It is given by the formula in equation 5.

$$\text{F-Score(Balanced between precision and recall)} = 2.\frac{\text{Precision.Recall}}{\text{Precision+Recall}} \tag{5.4}$$

5.2 Performance Metrics

In the majority of classification problems, performance measures such as Accuracy, Precision, Recall, ROC curve and AUC curve are usually used. However, for an

imbalance class Accuracy is misleading. So to avoid this confusion, we have used the F1 score, which has both Precision and Recall as the performance metric along with the classification report to identify the classes in our prediction.

5.3 Experiments:

5.3.1 Experiment 1:Tuned classifier with weights

The below experiments indicate the performance of the each of the tuned classifiers based on the optimal hyperparameters found along with balanced class weights assigned to the each of the classifiers.

TABLE 5.1: Performance of various classifiers

Baseline Performance				
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
KNN	86.36	0.99	0.87	0.92
Complement Naive Bayes	68.67	0.68	0.93	0.79
Logistic Regression	88.42	0.97	0.90	0.93
SVM	N/A	N/A	N/A	N/A
Decision Tree (depth = 3)	64.87	0.65	0.90	0.76
Random Forest	85.31	1.0	0.85	0.92
XG boost	88.37	0.97	0.90	0.93
Articial Neural network	90.42	0.97	0.92	0.94
LSTM	93.94	0.97	0.96	0.96
GRU	93.88	0.97	0.96	0.96
Transformer based (BERT)	97.23	0.99	0.98	0.98

Observations: From the table 5.1 We can infer that the highest F1 score of 98% was given by transformer based model. For the evaluation, we considered KNN and Naive Bayes as the potential baseline for benchmarking we have highlighted basic classifiers in red.

Note: We will discard the SVM from the model list as it is computation heavy and with an increase in features it took more than three days to train without any result.

5.3.2 Experiment 2: Impact of SMOTE on classifiers

We implemented various SMOTE generated dataset and the performance of each classifier on synthetic data both oversampled and undersampled and (hybrid such as SMOTEEEN) was found to be degrading the performance of the classifiers had been degraded. Smote generated data is available as a pickle file in the appendix section.

TABLE 5.2: Performance of Models relevant features

Performance of Best selected features				
Model	**Undersampling**	**Oversampling**	**Hybrid**	**base**
KNN	0.80	0..72	0.63	0.92
Naive Bayes	0.84	0.89	0.72	0.79
logistic Regression	0.56	0.54	0.52	0.93
Xgboost	0.76	0.74	0.81	0.93
ANN	0.89	0.72	0.80	0.94
Random Forest Classifier	0.61	0.62	0.57	0.92

Observations:

- From the table 5.2 we can see that there is a decrease of performance in all the models as compared to experiment 1, which signifies that the generation of synthetic word embeddings does not help in improving the performance of each of the Classifiers.
- For Upsampling, we used ADASYN (A KNN based approach) for downsampling, we performed a random down-sampling of data points. For hybrid approach we implemented SMOTE Tomek
- Now let us drop the models which are in red and consider only the top 3 models with good f1 score for further experiments.

5.3.3 Experiment 3: Critical Analysis: The DO's and DON'T

Each of the classifiers has a particular impact of the Outliers, Class imbalance and has different performance. From the table 5.3, we can see the High impact is denoted as RED while Low impact denoted as Green.

TABLE 5.3: Critical analysis

High Vs low color Response				
Model	**Outlier Impact**	**Class Imbalance**	**Train time complexity**	**Runtime Complexity**
KNN	**Low**	**Low**	**High**	**High**
Complement Naive Bayes	**Low**	**High**	**Low**	**Low**
Logistic Regression	**High**	**High**	**High**	**Low**
SVM	**Low**	**High**	**High**	**High**
Decision Tree (depth = 3)	**Low**	**High**	**Low**	**Low**
Random Forest	**Low**	**Low**	**High**	**High**
XG boost	**High**	**Low**	**Low**	**Low**
Articial Neural network	**Low**	**Low**	**Low**	**Low**
LSTM	**Low**	**Low**	**High**	**High**
GRU	**Low**	**Low**	**Low**	**High**
Transformer based (BERT)	**Low**	**Low**	**High**	**High**

5.3.4 Experiment 4: Model persistence : How to store my models efficiently ?

For the classifiers to work properly, we need to store them efficiently on the cloud ensuring their Read time (loading data from drive to Memory) / Write time (Writing data from Memory to drive) and Filesize (The size of the trained classifiers and the raw data file)

Since the problem is real-world relevant, it is essentially important to analyze the real-time aspect when deploying a server on the cloud as an API. To do so, we encounter a number of different issues, such as :

- Memory read time: The amount of time the data takes to be read from Memory. The more is the read time; the larger is the server wait time when initializing the classifier and data.
- Memory Dump time: The total amount of time taken to write the data from the Memory to the drive. The order of magnitude depends on the write size of the file. This is an important task when retraining models. This is the total time the model takes to replace the preexisting model on the drive.

– Space efficiency: The total amount of the space taken by the data on the server mainly varies the Cost of the hosting server. Powerful compression algorithms can mitigate the potential server space exhaustion.

We pickle(serializing and deserializing python object structure) and evaluate the performances of various compression-based algorithms comparing with baseline (Raw pickling)

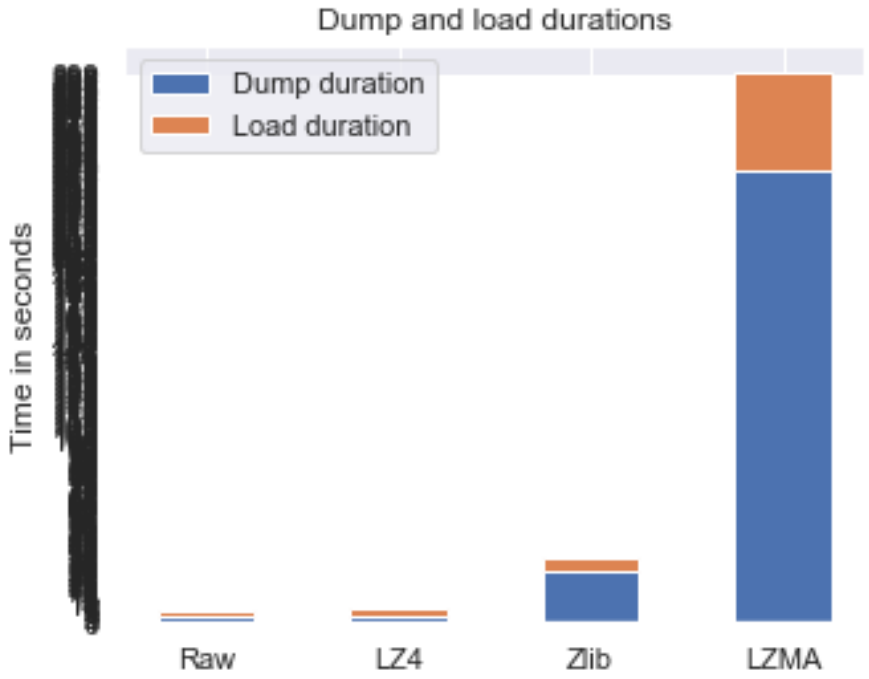


FIGURE 5.2: File read time from Memory when reading the amazon data

LZ4 compression algorithm offers the lowest read and write time to the Memory with an average of 3.11 seconds after ten runs of the algorithm.

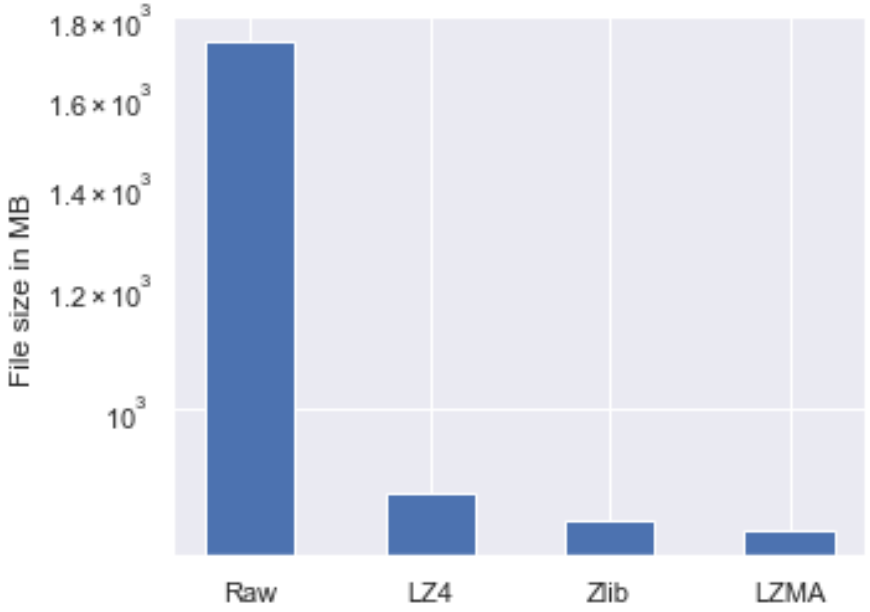


FIGURE 5.3: Compression size of the AMAZON file with various algorithms

LZMA compression provides the Lowest compression size. However, LZ4 is marginally behind from LZMA with only 34 MB's Extra compressed by LZMA. However, the loading time has a vast difference of 50 seconds on reading the file to the Memory. We propose the optimum choice of **LZ4** when using the Classifiers on low latency platforms.

Chapter 6

Conclusions and Future Work

The proposed solution solves the problem of sentiment classification with a decent F1 score of 97-98% when applying the **BERT** based model.

6.0.1 Conclusions

In this project, we demonstrated a capable sentiment classifier based on a variety of different algorithmic techniques and improvements.

A list of subtle conclusions made from this thesis are as follows:

- Transformer based models such as BERT gave the highest performance over sentiment analysis leveraging the benefit of Bidirectional LSTM's and self-attention units and gaining another 2% increase than the unidirectional LSTM's.
- For the comparison of proposed models, we fine-tuned the baselines to have a valid comparison against the finely tuned BERT model.
- Any variant of SMOTE (upsampling and downsampling or a hybrid approach) did not improve the class imbalance problem.
- BERT poses a considerable advantage in terms of precision and recall score; however, the large increase in the parameter can effectively scale up the training time.
- for emoji-based classification BERT is not able to capture the relevant learning due to the low sample size of emoji-based reviews.

6.1 Future Work

– BERT has two major problems which XL net transformer resolves[48] (1) BERT randomly masks 15% of the points while XL net uses permeation based scheme to pick neighbours without changing the order of words. (2) BERT takes an input of fixed sentence length while XL net adapts the ideas from Transformer XL model with no fixed sequence length.

– Application of larger transformer-based models such as the GPT-3 [49] essentially with a huge number of parameters (175B) when compared to BERT large(1.27B) is currently state of the art beating BERT based models on a variety of language task.

Bibliography

[1] A. Skupin, J. R. Biberstine, and K. Börner, "Visualizing the topical structure of the medical sciences: a self-organizing map approach," *PloS one*, vol. 8, no. 3, p. e58779, 2013.

[2] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[3] T. U. Haque, N. N. Saber, and F. M. Shah, "Sentiment analysis on large scale amazon product reviews," in *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*. IEEE, 2018, pp. 1–6.

[4] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan, "A system for real-time twitter sentiment analysis of 2012 us presidential election cycle," in *Proceedings of the ACL 2012 system demonstrations*, 2012, pp. 115–120.

[5] I. Kwok and Y. Wang, "Locate the hate: Detecting tweets against blacks," in *Twenty-seventh AAAI conference on artificial intelligence*, 2013.

[6] N. Asghar, "Yelp dataset challenge: Review rating prediction," *arXiv preprint arXiv:1605.05362*, 2016.

[7] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 897–908.

[8] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[9] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. J. Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the workshop on language in social media (LSM 2011)*, 2011, pp. 30–38.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[11] H. Lee and S. Kang, "Word embedding method of sms messages for spam message filtering," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2019, pp. 1–4.

[12] Y. Lin, H. Lei, J. Wu, and X. Li, "An empirical study on sentiment classification of chinese review using word embedding," *arXiv preprint arXiv:1511.01665*, 2015.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[17] D. Pyle, "Data preparation for data mining morgan kaufmann publishers," *Inc., California*, 1999.

[18] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "paper recommender systems: a literature survey," *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016.

[19] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[20] D. M. Powers, "Applications and explanations of zipf's law," in *New methods in language processing and computational natural language learning*, 1998.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[22] N. Sayer, "Google code archive-long-term storage for google code project hosting," *XP055260798, Retrieved from the Internet [retrieved on 20160323]*, 2014.

[23] B. Eisner, T. Rocktäschel, I. Augenstein, M. Bošnjak, and S. Riedel, "emoji2vec: Learning emoji representations from their description," *arXiv preprint arXiv:1609.08359*, 2016.

[24] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[25] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[26] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[27] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[28] B. Trstenjak, S. Mikac, and D. Donko, "Knn with tf-idf based framework for text categorization," *Procedia Engineering*, vol. 69, pp. 1356–1364, 2014.

[29] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, "Multinomial naive bayes for text categorization revisited," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2004, pp. 488–499.

[30] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[31] L. Rokach and O. Z. Maimon, *Data mining with decision trees: theory and applications*. World scientific, 2008, vol. 69.

[32] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[33] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[34] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[35] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *Advances in neural information processing systems*, 2018, pp. 6638–6648.

[36] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, 2017, pp. 3146–3154.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[38] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[39] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[40] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE, 2008, pp. 1322–1328.

[41] G. E. Batista, A. L. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: a case study." in *WOB*, 2003, pp. 10–18.

[42] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[43] P. Delobelle and B. Berendt, "Time to take emoji seriously: They vastly improve casual conversational models," *arXiv preprint arXiv:1910.13793*, 2019.

[44] V. Prasath, H. A. A. Alfeilat, A. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, and H. S. E. Salman, "Distance and similarity measures effect on the performance of k-nearest neighbor classifier–a review," *arXiv preprint arXiv:1708.04321*, 2017.

[45] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[47] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[48] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5753–5763.

[49] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

Appendix A

Code Snippets

full version of code : `https://github.com/mv96/sentiment_analysis_cit`

KNN confusion matrix:

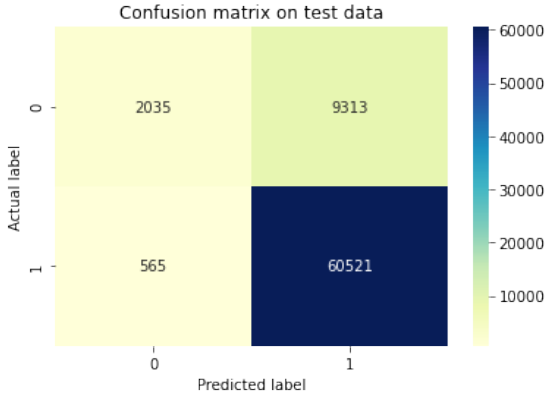


FIGURE A.1: Confusion Matrix for KNN

Naive Bayes confusion matrix:

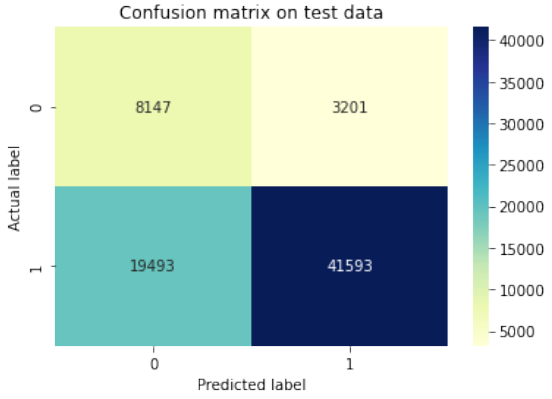


FIGURE A.2: Naive Bayes confusion matrix

logistic regression confusion matrix:

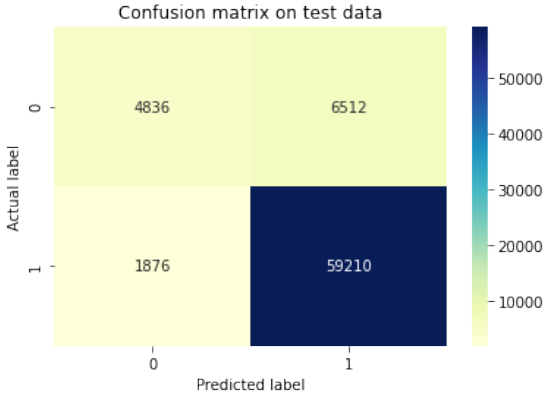


FIGURE A.3: Confusion matrix for tuned logistic regression

Decision tree confusion matrix:

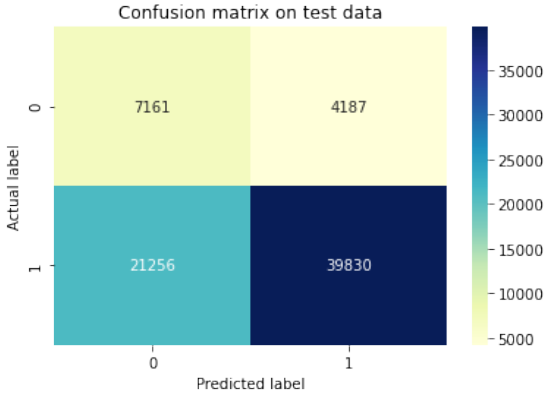


FIGURE A.4: Decision Tree confusion Matrix

Random Forest Confusion matrix

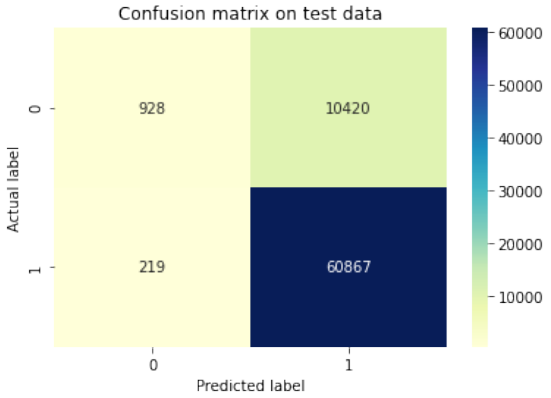


FIGURE A.5: Random forest confusion Matrix

Gradient Boosted Decision Tree's:

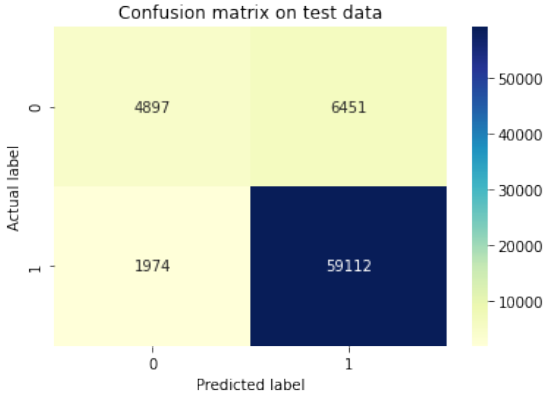


FIGURE A.6: Confusion matrix for XGBoost (Decision Trees)

Artificial Neural Network:

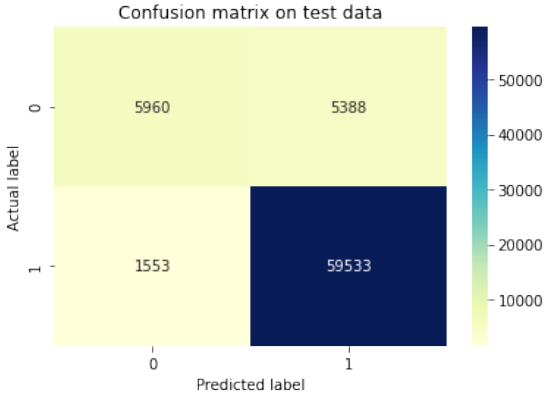
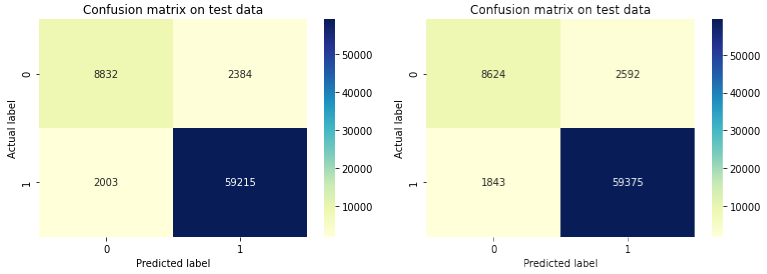


FIGURE A.7: Confusion matrix for neural network architecture

LSTM & GRU's (Sequence based models):



(a) LSTM based results (b) GRU based results

FIGURE A.8: Confusion matrix for Gated units(LSTM's/GRU's)

BERT(Large) based Transformer:

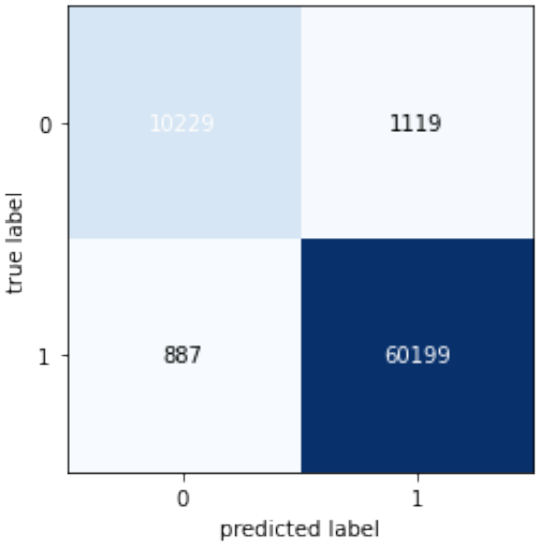


FIGURE A.9: Confusion matrix for BERT transformer

A.1 Text inputs from the final model:

```
In [7]: text="Betty bought some butter, \
        but the butter Betty bought was bitter, \
        so Betty bought some better butter, \
        and the better butter Betty bought \
        was better than the bitter butter Betty bought before!"
        data=[text]
        print(predictor.predict_proba(data))
        predictor.predict(data)

        [[0.08480582 0.9151942 ]]
```

FIGURE A.10: TEST:1

```
In [8]: text="all that glitters is not gold !"
        data=[text]
        print(predictor.predict_proba(data))
        predictor.predict(data)

        [[0.8230453  0.17695463]]

Out[8]: ['0']
```

FIGURE A.11: TEST:2

```
In [9]: text="made in china product"
        data=[text]
        print(predictor.predict_proba(data))
        predictor.predict(data)

        [[0.92454135 0.07545865]]

Out[9]: ['0']
```

FIGURE A.12: TEST:3

```
In [12]: text="never say never to good beer 👍"
         data=[text]
         print(predictor.predict_proba(data))
         predictor.predict(data)

         [[0.00448012 0.9955199 ]]

Out[12]: ['1']
```

FIGURE A.13: TEST:4

```
In [13]: text="winners never quit"
         data=[text]
         print(predictor.predict_proba(data))
         predictor.predict(data)

         [[0.00130827 0.99869174]]

Out[13]: ['1']
```

FIGURE A.14: TEST:5

```
In [14]: text="quitters never win"
         data=[text]
         print(predictor.predict_proba(data))
         predictor.predict(data)

         [[0.49264574 0.5073543 ]]

Out[14]: ['1']
```

FIGURE A.15: TEST:6

```
In [16]: text="Satisfied with Amazon delivery but bad product inside"
         data=[text]
         print(predictor.predict_proba(data))
         predictor.predict(data)

         [[0.9344007  0.06559937]]

Out[16]: ['0']
```

FIGURE A.16: TEST:7

```
In [17]: text="Satisfied with Amazon delivery and good product inside"
         data=[text]
         print(predictor.predict_proba(data))
         predictor.predict(data)

         [[2.6498438e-04 9.9973506e-01]]

Out[17]: ['1']
```

FIGURE A.17: TEST:8