Emotion Detection for Social Media Posts using Classification

Cardiff University School of Computer Science and Informatics

BSc Computer Science

Semih Sarisoy - 22058649

Supervised by Professor Irena Spasic

Date - 08/05/2025

Abstract

Social media platforms have become integral to modern communication, with users expressing emotions through text-based interactions. This project investigates emotion classification in Bluesky, an open-source social media app, by developing a tool to classify emotions in user posts and generate emotional charts and comment analysis.

The aim of this project is to investigate how emotions are expressed in the text-based social media app Bluesky and to develop a tool that can classify emotions and generate useful charts and analysis.

The project objectives include assessing social media content, particularly examining how the absence of a dislike button complicates the evaluation of content quality, while still allowing comments to serve as a medium for discussion. It also involves developing an emotion classification system based on Ekman's six basic emotions (sadness, joy, love, anger, fear, and surprise) and evaluating five models using machine learning and deep learning approaches. In addition, a web-based tool was built to integrate real-time data from Bluesky, ensuring compliance with the platform's official API policies.

Five different machine learning and deep learning models were compared, including a Recurrent Neural Network (RNN) using Long Short-Term Memory (LSTM) with Bidirectional Processing, which achieved the best performance according to its accuracy score of %89.64. Along with LSTM model the second highest performing machine learning model was SVM with 87% accuracy, due to its nature of understanding higher dimensional vectors.

Model evaluation was conducted using multiple performance metrics, with deep learning outperforming traditional machine learning methods in accuracy score as expected. However, hardware limitations extended training times, and the dataset size around 500,000 texts may have constrained model accuracy.

The tool helps users determine whether a post is worth reading by analysing popular comments and analysing emotional reactions. It can also be applied to analyse user responses to videos and images on Bluesky.

These findings highlight the potential of deep learning in emotion detection via classification and its application in real-time social media analysis.

Acknowledgements

I would like to thank my supervisor Professor Irena Spasic for helping me in general throughout the entire project. I would also like to thank my family and friends for the motivation they gave me and their support.

Contents

1. Introduction

1.1 Text Classification and Emotion Detection

Natural Language Processing (NLP) is a subfield of Artificial Intelligence and Machine Learning encompassing a set of methods for making human language accessible to computers. Text classification is a fundamental aspect of NLP, where given a text document it assigns the document a discrete label $y \in Y$ where Y is the set of possible labels (Eisenstein, 2019). Emotion detection, sometimes referred to as emotion recognition, is the process of identifying a person's emotional state by analysing a text document written by them (Shivhare & Saritha Khethawat, n.d.). This project will utilize

text classification to detect emotions in Bluesky(*Discover — Bluesky*, n.d.) social media posts. The classification task involves assigning one of six possible labels from the dataset with Ekman's six basic emotions as they perform best in emotive language analysis when F1 measures are compared with others (Williams et al., 2019). Among these, joy, love, and surprise are categorized as positive emotions, while sadness, fear, and anger are considered negative emotions.

1.2 Aims and Objectives

1.2.1 Aim 1

The first aim of this project is to investigate how emotions are expressed through text on the social media platform Bluesky. Bluesky features an interface identical to X (formerly Twitter), where most posts contain text, although some may include images and videos. Even in text-based communication, emotions can be expressed in multiple ways, including plain text, punctuation marks, emojis, and hashtags. Classifying these different representations poses a challenge. Therefore, this aim focuses on a text-only approach, excluding other forms of emotional expression such as images and videos although emojis can be pre-processed to convert to text and hashtags can also be pre-processed and treated as text. The emotions expressed in comments may indicate the implications of not having a dislike button and how users assess content quality solely through comments. Objective 1 further examines this topic. Additionally, a tool will be developed to extract real-time data from Bluesky for emotion analysis. The details of this implementation are provided in Objective 3.

1.2.2 Aim 2

The second aim of this project is to develop a tool that can classify emotions and generate meaningful charts that analyses the posts overall emotion. These analyses include identifying the emotion classes of the ten most popular comments after classification and generating two bar graphs. One graph will show the distribution of the Ekman's six basic emotions (sadness, joy, love, anger, fear, and surprise) across these ten comments. The other graph will display the total number of positive and negative emotions, where positive emotions include joy, love, and surprise, and negative emotions include anger, fear, and sadness. The tool will utilize the machine learning or deep learning approach that demonstrates the best overall performance. Performance comparisons will be conducted using multiple evaluation metrics, such as accuracy, precision, and F1-score. Accuracy will be the primary metric, as it is the easiest to interpret and provides the most straightforward analysis into model performance. Additionally, a web-based tool will be developed to integrate the trained and saved model via an API and connect it to a dynamic webpage for real-time emotion classification. Objective 2 further explores the details of the machine learning and deep learning approaches, while Objective 3 discusses details about the web-based tool.

1.2.3 Objective 1

The main research component of this project is to research the effects of how dislike removal affects online content quality feedback for users. Multiple research papers will be analysed to assess how removing the negative feedback (dislike) button affects the assessment of quality. This section consists solely of research on the topic and aims to provide a general understanding of whether the tool being developed will be valuable or not since comments may provide charts about the overall content quality. In the end a decision will be made on whether the tool being developed will be useful or not.

1.2.4 Objective 2

This section focuses on selecting the appropriate machine learning or deep learning algorithms for emotion classification. One important aspect is choosing the right text analysis technique, with two main options considered: Term Frequency–Inverse Document Frequency (TF-IDF) and Bag of Words (BoW). Another key decision is selecting the algorithms to be trained, which may include Naïve Bayes, Support Vector Machines (SVM), Decision Trees, Random Forest, and Recurrent Neural Networks (RNNs). Training five different algorithms should provide sufficient insight into which model best suits the task.

1.2.5 Objective 3

The implementation of the web-based tool involves an API that will act as a bridge between the webpage and the best performing trained algorithm, processing input links and returning results. Two widely used frameworks for developing web-based tools are considered: Flask, known for its speed and efficiency in small to medium-sized projects, and Django, recognized for its advanced functionality in larger applications. In addition to the API, HTML and CSS will be sufficient for building the web-based tool, though JavaScript may be incorporated for additional functionalities if necessary.

2. Background

2.1 Related Work

2.1.1 Social Media Psychology and User Engagement

Social media platforms are growing every day, fostering an increasing herd mentality. User-engageable features such as up-votes, down-votes, and comments can be easily manipulated, making it difficult to assess the right quality of content. According to a study on collective psychology conducted in 2013, this issue existed even before the explosion in social media user numbers(Muchnik et al., 2013). Manipulated up-votes or likes can result in an increase ranging from 0.07 to 0.10 (42.86%). If the content creator is perceived as a friend, such manipulation can lead to a rise of 0.11 to 0.175 (59.09%)

in up-votes. This manipulation creates bias in electoral polling, stock market predictions, and product recommendations. Engagement metrics now dictate content value, as every creator strives to maximize interactions such as likes and dislikes. Moreover, these metrics are used in machine learning algorithms to promote content to new audiences, intensifying competition among content creators. These algorithms are used to promote that content to people who have not yet seen it, which creates an even bigger competition around content creators (Figure – 1).

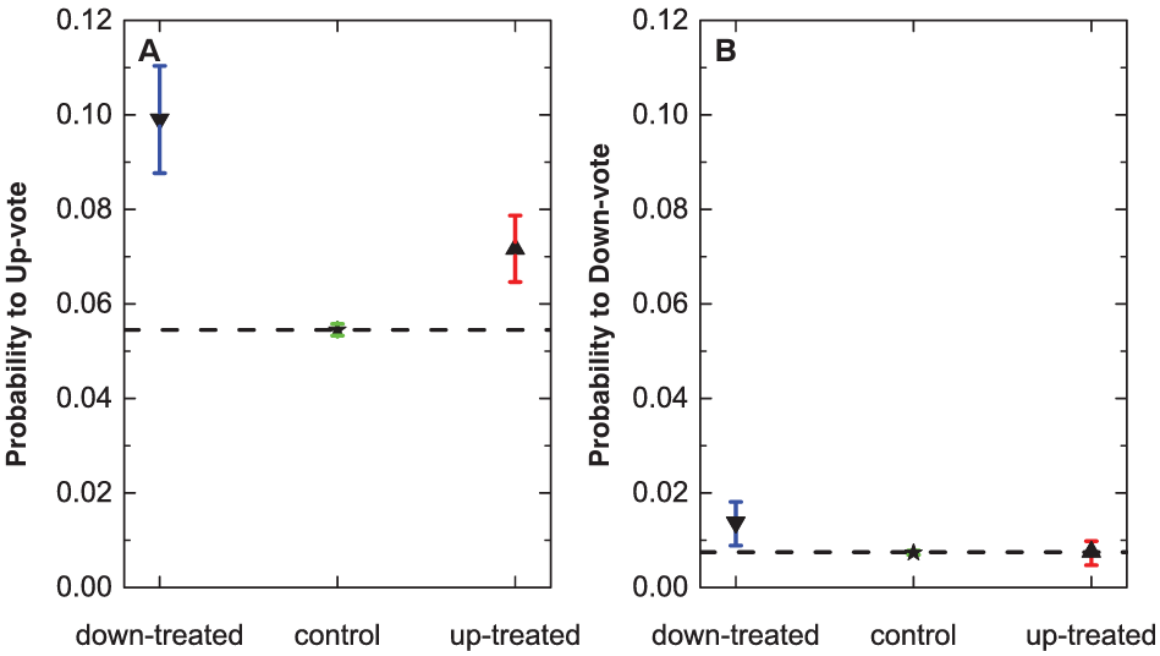


Figure 1 – Manipulated Up-votes and Down-votes probability (Muchnik et al., 2013)

The most critical section of the paper addresses crowd correction in the context of manipulating down-votes or dislikes. The research indicates that users are less likely to be manipulated by dislikes, and a larger number of people seem to correct the error caused by manipulated dislikes. The numbers can be seen as a small rise from 0.09 to 0.11 (22.22%) This section is valuable, as additional evidence is needed in the subsequent part to justify the company's decision to eliminate the negative feedback button.

2.1.2 Emotional Responses to Digital Content

Emotional responses by content creators often follow a specific Risk Sensitivity Theory (RTS) without their realizing it, according to a study by (Turel & Qahri-Saremi, 2024). This behaviour is observed in both animals and humans, where deprivation leads to the pursuit of higher risk-reward opportunities. In social media, content creators are less concerned with obtaining more likes or upvotes and are more focused on minimizing downvotes. While likes and dislikes were initially introduced as a means of providing

user feedback, today many users and content creators seem to dislike both especially downvotes.

Content creators increasingly engage in Technology-Mediated Dangerous Behaviours (TMDBs) because they aim to receive fewer dislikes on social media. According to a study using descriptive statistics, users are more likely to post a TMDB-based selfie which might include racist, sexist, risky, or rebellious content if they receive many likes and close to zero dislikes.

This study helps explain why major companies have decided to eliminate the dislike or downvote button and, in some cases, even hide the like or upvote button. Although downvotes cannot be manipulated, they may still be received if most users perceive the content unfavourably. However, understanding the reasons behind dislikes is challenging, as the absence of comments leaves the exact cause open to interpretation.

2.1.3 Online Feedback Mechanisms and Their Impact

Online feedback mechanisms date back to the early 1990s "specifically, 1993–1994" when the World Wide Web became publicly available, and a guestbook feature was introduced. This feature allowed users to leave comments on the few websites that existed at the time. A few years later, in 1997, the first social media platform, SixDegrees, was launched with commenting functionality. The 1990s were the era of commenting for online feedback, as the number of websites and data stored were relatively low.

The 2000s saw an exponential rise in the number of websites, with millions coming online each year. In 2005, YouTube was launched, introducing two innovative feedback mechanisms "like and dislike" representing up-votes and down-votes. With the increasing amount of data and users, the volume of comments grew significantly, making it challenging to manually analyse thousands of responses. The new like/dislike feature offered a binary way to store user reactions, making it easier to determine whether users enjoyed the content.

In modern times "specifically during the 2010s and 2020s" the like/dislike feature has harmed society, as discussed in previous sections. In 2021, YouTube decided to remove the public dislike counts because they were seen as encouraging antisocial behaviour toward both content creators and users. This change indicates a shift in industry trends back toward a focus on comments, with companies now also considering the removal of like counts.

2.1.4 The Dynamics of Likes and Dislikes

Analysing how likes and dislikes work is a critical topic that involves various elements, including psychological factors. However, the dynamics behind these social buttons go far beyond that. An important aspect is understanding the economy behind them. A study published in 2013 analysed Facebook (now Meta) and its economy built around likes (Gerlitz & Helmond, 2013). According to the paper, in 2013, Facebook used the like button to automatically collect certain cookies from users as they clicked. These cookies allowed Facebook to track users even when they were not using any Facebook app as long as they had an internet connection, they continued to be tracked by Meta. Deleting your account or disabling cookies does not help either, since Meta appears to have the right to retain the previously collected data.

This issue exists on other social media platforms as well, such as Bluesky, which I am using for this project. The data these companies collect can be sold, processed, or stored until the company decides to delete it. Third-party companies can purchase this data, whether processed or raw, fuelling an entire economy centred around social media likes. The reason Facebook still does not have a dislike button, according to the paper, is that it does not contribute to this economy. Yanis Varufakis, a respected economist, believes that this has created a new form of capitalism called technofeudalism.

Another study analysing YouTube's removal of the dislike button found that hiding the dislike count did not change the overall sentiment or toxicity levels in the comments (Zhang & Ng, 2024). The study used time series analysis to compare comments before and after hiding the dislike count and found no significant change based on the chosen p-value. This finding aligns with the notion that manipulation is harder to achieve with dislikes, as content that is not liked basically means it is not liked. There is little to no manipulation. While the frequency of comments per user decreased and the overall sentiment of comments became slightly more negative, there is still no evidence to suggest that removing the dislike button was definitively good or bad. The slight increase in negativity appears to be a straightforward response directed at the content creator, as the hidden dislike count diminishes the perceived meaning of dislikes.

2.1.5 The Impact of Removing the Dislike Button

Taking away a tool for expressing negative emotions limits people's freedom to share honest opinions. Previous research shows that removing the dislike button negatively affects freedom of expression and the ability to assess content quality. Ultimately, the lack of dislike button, pushes users to consider comments for the purpose of estimating quality because it is impossible to gauge content quality using only likes or up-votes.

2.1.6 Psychological and Social Implications

Will users want to engage with content without a negative feedback button? A research paper titled “Do Online Reviews Matter?” finds that online reviews or comments help drive engagement, as people rely on word-of-mouth, which produces an effect like that of reviews or comments (Chevalier & Mayzlin, 2006). These reviews affect individuals psychologically and also impact society as a whole.

2.1.7 Summary and Critical Review

To understand the role of feedback mechanisms such as the “like” and “dislike” buttons in social media platforms, I reviewed several key studies, including those by (Muchnik et al., 2013) and (Gerlitz & Helmond, 2013). Muchnik et al. conducted an experimental study on a news aggregation website, demonstrating that users’ opinions can be easily manipulated through early upvotes or downvotes, suggesting that these feedback systems can distort public perception rather than reflect genuine sentiment. On the other hand, Gerlitz & Helmond approached the issue from a socio-technical perspective, focusing on how feedback buttons function as data-collection tools for platforms. They argue that these interactions are commodified, with user engagement being harvested and monetised by corporations.

While both papers critique the use of feedback buttons, they emphasise different consequences—(Muchnik et al., 2013). highlight cognitive manipulation and bias, whereas Gerlitz & Helmond focus on surveillance and data exploitation. This contrast reveals that the negative impacts of feedback mechanisms are multifaceted, ranging from social engineering to privacy concerns.

In contrast to buttons, some researchers propose that user comments—when analysed correctly—offer a more nuanced and constructive form of feedback. (Chevalier & Mayzlin, 2006), for instance, studied consumer reviews and found that textual feedback provides richer insights into user satisfaction and content quality. Although comments can still be biased or emotional, they offer a layer of complexity that binary buttons inherently lack.

By synthesising these perspectives, I conclude that automated analysis of user comments, such as through natural language processing techniques, could provide a less harmful and more informative alternative to simplistic button-based feedback systems. This has informed both the theoretical and practical direction of my project.

2.2 Natural Language Processing for Emotion Detection

Natural Language Processing (NLP) is a subfield of Artificial Intelligence that includes a variety of tools and methods to make human language accessible to computers (Eisenstein, 2019). Emotion detection, a branch of NLP, enables machines to understand emotions in human language. Tools such as logic, statistics, machine

learning, linear algebra, and programming are used for emotion detection. This project focuses solely on a text classification approach to achieve emotion detection.

2.2.1 Preprocessing Text from Social Media Posts

To perform text classification an NLP approach on social media posts, it is necessary first to clean the text through preprocessing. The steps used in this project include tokenization, stop word removal, and emoji and hashtag processing.

- Tokenization: This involves separating the text into tokens, such as individual words.
- Stop Word Removal: This step removes extraneous words like “the” or “for,” which do not contribute meaningfully to understanding sentiment and emotions.
- Emoji and Hashtag Processing: This is a crucial step, as most social media posts include emojis and hashtags. These can be converted into emotionally relevant and useful text.

2.3 Machine Learning and Deep Learning Models

This project will include five different models for text classification. Each model will be evaluated using various predictive performance measures, including precision, recall, F1-score, accuracy, macro average, and weighted average, except for the deep learning approach, which will only use accuracy and validation accuracy.

Each approach will have a regularization parameter that will be adjusted to achieve optimal performance, except the deep learning approach, which is a completely different method that may adapt itself.

A balanced approach was taken in selecting the five machine learning algorithms to represent different families of models. While many algorithms are available, only a subset is particularly effective for text classification. Briefly, Multinomial Naïve Bayes is a probabilistic model commonly used in spam detection due to its efficiency with word frequency-based features. Linear Support Vector Classification (Linear SVC) is a linear model that performs well on high-dimensional data such as TF-IDF, making it suitable for tasks like emotion classification. Decision Tree is a tree-based model often used in industry for its interpretability and strong performance, especially when extended into ensemble methods like Random Forests. Stochastic Gradient Descent (SGDClassifier) is also a linear model, but it uses approximate optimisation techniques to enable faster training on large datasets. Finally, Bidirectional Long Short-Term Memory (BiLSTM), a deep learning model, was chosen to capture the contextual and semantic relationships in text sequences more effectively.

These algorithms were selected not only based on their prevalence in text classification literature but also to provide a diverse and balanced comparison across probabilistic,

linear, tree-based, and deep learning approaches, allowing for a broader understanding of how different model types perform on this task.

2.3.1 Multinomial Naïve Bayes

This is a widely used statistical approach for classifying discrete data. It employs Bayes' Theorem for calculations, with the "naïve" aspect referring to the conditional independence assumption. The term "multinomial" indicates that it is suitable for classification tasks involving discrete data. The formula for Bayes' Theorem is shown below along with an explanation.

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

The probability of A given B is equal to the probability of B given A multiplied by the probability of A, divided by the probability of B. This concept can be extended to multiple classes (or dimensions) in multinomial naïve Bayes (Figure – 2).

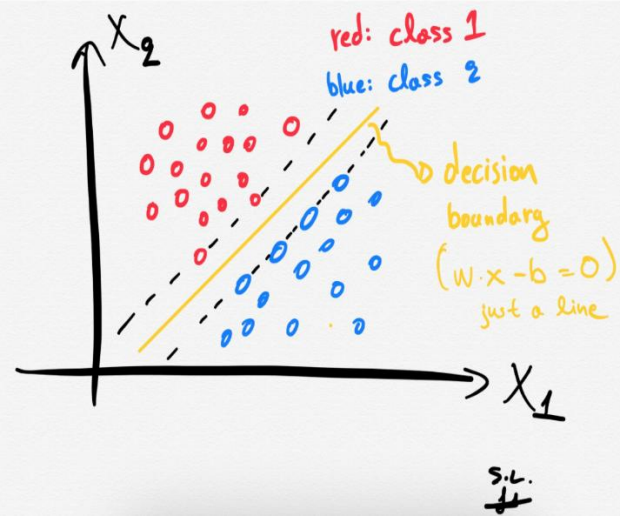
Figure 2 – Bayes Theorem(*Bayes Theorem - Statement, Proof, Formula, Derivation & Examples*, n.d.)

For the multinomial naïve Bayes algorithm, the alpha smoothing parameter will be adjusted as a regularization parameter. This additive smoothing factor helps control the model (*MultinomialNB — Scikit-Learn 1.6.1 Documentation*, n.d.).

2.3.2 Linear Support Vector Classification

This is a statistical approach that also utilizes linear algebra and optimization. It uses a hyperplane as a decision boundary to perform calculations. As the name suggests, a linear boundary is used in this algorithm for simplicity and improved performance, which also helps prevent overfitting. The formula and an example image are shown below, along with an explanation.

$$w_1 x_1 + \ldots + w_d x_d + \beta_0 = 0,$$



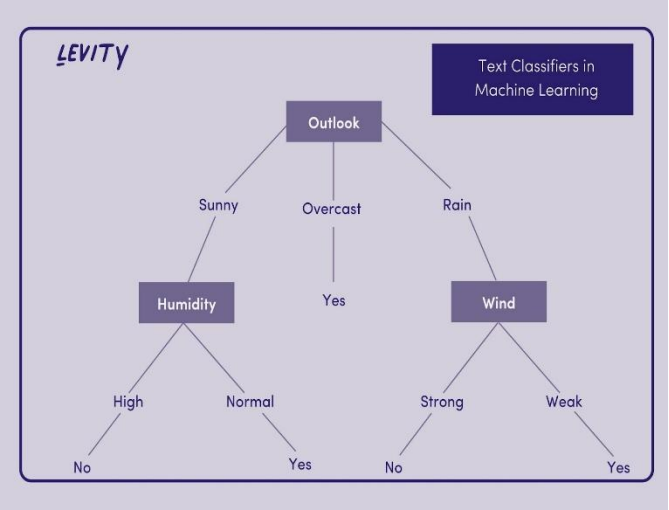
Mathematically, a hyperplane is defined as the weight vector multiplied by the feature vector, with a bias term added. It can incorporate as many dimensions as needed for classification. If there is no clear separation between classes, a new dimension may be added to provide a different perspective for finding a solution. In this context, support vectors are the points that are closest to the decision boundary (Figure – 3).

Figure – 3 Linear Support Vector Classifier(*Support Vector Machines (SVM) Clearly Explained: A Python Tutorial for Classification Problems... | Towards Data Science*, n.d.)

For the linear support vector classification algorithm, the penalty parameter C will serve as the regularization parameter, with the strength of regularization being inversely proportional to C (*LinearSVC — Scikit-Learn 1.6.1 Documentation*, n.d.).

2.3.3 Decision Tree

This is a non-mathematical decision-making approach. It does not use statistics or any other areas of mathematics, such as linear algebra. Essentially, the text data starts at the root of a tree and moves toward one of the leaves, where it is filtered through multiple nested if-else conditions. Once the filtering is complete, a decision is made about the class of the text.



One advantage of this approach is that the algorithm's decision-making process is easy to follow and visualize, as it is represented by a simple tree structure without multiple dimensions or complexities.

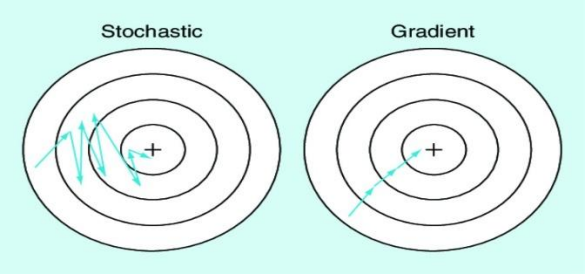
In industry, it remains the go-to algorithm for tabular data; however, its performance on specialized tasks, such as image or text analysis, is less predictable (Figure – 4).

Figure – 4 Decision Tree(*Text Classifiers in Machine Learning: A Practical Guide*, n.d.)

For the decision tree algorithm, the complexity parameter Cost Complexity Pruning (CCP) alpha serves as the regularization parameter. In this approach, the subtree with the largest cost complexity that is smaller than CCP alpha is chosen. By default, this value is set to zero (*DecisionTreeClassifier — Scikit-Learn 1.6.1 Documentation*, n.d.).

2.3.4 Stochastic Gradient Descent

This algorithm is an improvement over the standard gradient descent approach, incorporating foundational concepts from optimization and calculus. The perceptron loss function can be used, as it is particularly effective for classification tasks. Additionally, this algorithm approximates the gradient descent process by sampling from a large dataset.



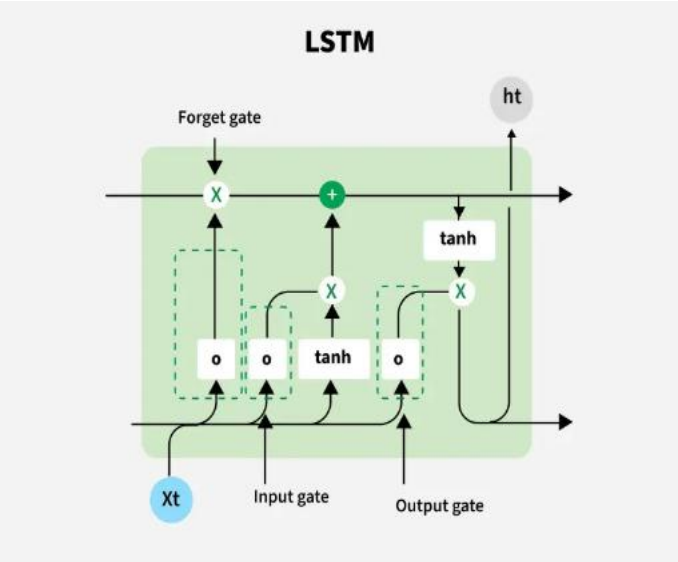
One advantage of this algorithm is that it operates much faster than regular gradient descent because it does not use the full dataset. However, it is important to note that the randomly selected data might be skewed rather than centred (Figure – 5).

Figure 5 – Stochastic Gradient Descent(*Efficient Optimi: Mastering Stochastic Gradient Descent*, n.d.)

For the stochastic gradient descent algorithm, alpha serves as the regularization term the higher its value, the stronger the regularization. Additionally, this term is used to compute the learning rate when it is set to "optimal," which is the default setting (*SGDClassifier — Scikit-Learn 1.6.1 Documentation*, n.d.).

2.3.5 Bidirectional Long-Short Term Memory (BiLSTM) (RNN)

This technique is a deep learning approach that outperforms traditional Recurrent Neural Networks (RNNs). It employs a bidirectional long short-term memory network designed to capture long-term dependencies in text sequences by simultaneously monitoring information flow from both previous and current timestamps in forward and backward directions (*Bidirectional Long Short-Term Memory Network - an Overview | ScienceDirect Topics*, n.d.).



LSTMs include an input gate, an output gate, and a forget gate for each cell, which makes them effective for long-term memory retention. They also perform well in tasks such as sentiment analysis, speech recognition, and language translation (Figure – 6).

One disadvantage is that, compared to transformers, these models are considered smaller; however, they still perform well in most cases.

Figure – 6 LSTM Cell(*RNN vs LSTM vs GRU vs Transformers | GeeksforGeeks*, n.d.)

This deep learning algorithm can adjust its weights and adapt itself, meaning it will not require a regularization parameter. However, it is important to include the correct number of layers, the appropriate loss function, and the right optimizer, as these choices are made during the initial model compilation.

2.4 Libraries, API and Tools

The Python programming language works well for this project, as it is a general-purpose language with many popular libraries for machine learning and natural language processing. For this project, we will use libraries such as scikit-learn, pandas, TensorFlow, NLTK, matplotlib, seaborn, NumPy, and datasets, each serving specific purposes like data manipulation or processing. The API used for this project is the AT Protocol SDK, an official API supported by Bluesky. For the website component, we will choose the Flask web development framework because it supports faster development compared to Django, which will be useful given the project's agile methodology.

2.5 Motivations and Further Applications

The main motivation for this project is to develop a response to social networking companies that remove certain features from social media such as up-vote/down-vote counts and specific comment filtering. The social media landscape has changed drastically in recent years, trending toward displaying less negative content about individuals. Although companies have their motivations for removing these features, research shows that each removed feature makes it harder for users to assess content quality.

Another aspect of my motivation is my passion for understanding why the software we use every day can change dramatically without clear explanations. My research highlights overall trends in why these changes occur and how they may continue in the future.

These potential changes in social media increase the number of applications for this project. The more vote counts that are removed, the more useful this web-based application becomes. People today are unwilling to invest time in long-term tasks, including navigating social media. For example, on YouTube, most users simply want to find entertaining content, or a guide quickly and then leave once they have what they need. When searching for a guide, the easiest way to rank videos is by the like/dislike ratio a method that websites have recommended.

Since the dislike counts are now hidden and there are plans to remove like counts as well, finding the desired content has become more difficult. This issue is also evident in movie apps like Netflix, where content is longer. Additionally, pure machine learning algorithms sometimes fail to recommend the appropriate content because they do not capture the user's mood. Overall, the potential applications for this project continue to grow day by day.

3. Methodology

For this project, I have decided to use the Agile methodology, as flexibility was required both in designing the web-based tool and in developing the machine learning algorithms. In project management, Agile is generally not associated with using a Gantt chart (Figure – 7) however, for this project, incorporating one is important due to the need for effective time management. To maintain flexibility, a product backlog will also be used, as it allows work to be carried out in any permissible order. This project adopts a sprint-based Agile framework called Scrum, which is designed to accelerate development without significantly compromising performance (Srivastava et al., 2017). While it becomes more complex to track what has been done since it can be changed it is still possible to have a rough idea.

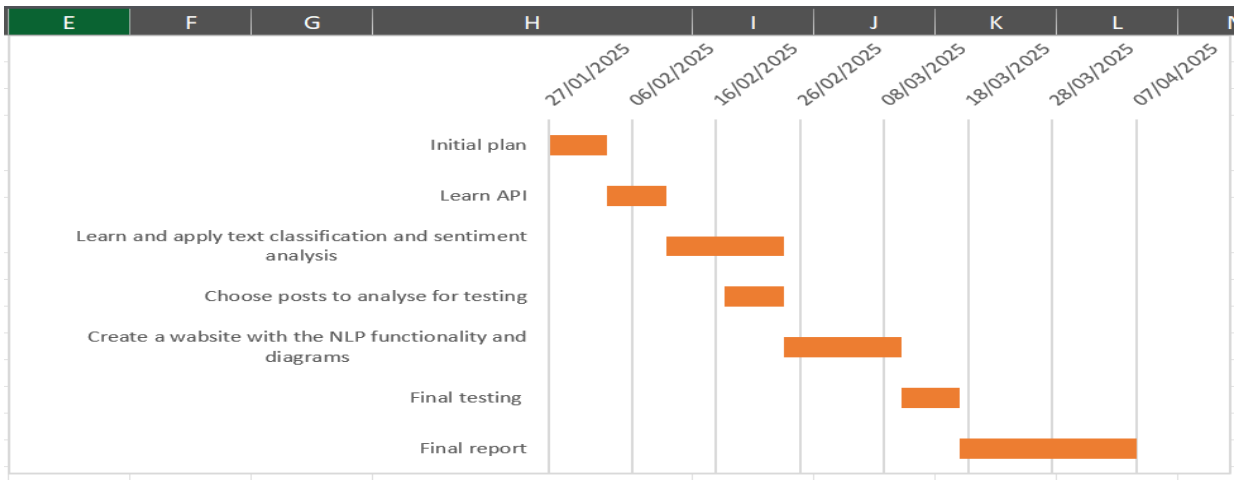


Figure – 7 Gantt chart created by excel.

Utilizing a product backlog is crucial for this project, as it allows tasks to be separated in a flexible manner and enables the use of a checklist to track progress efficiently. For example, learning how to use the API can be defined as a main task, with subtasks such as reading the API documentation, extracting a single post, and extracting multiple posts included in the checklist (Figure – 8).

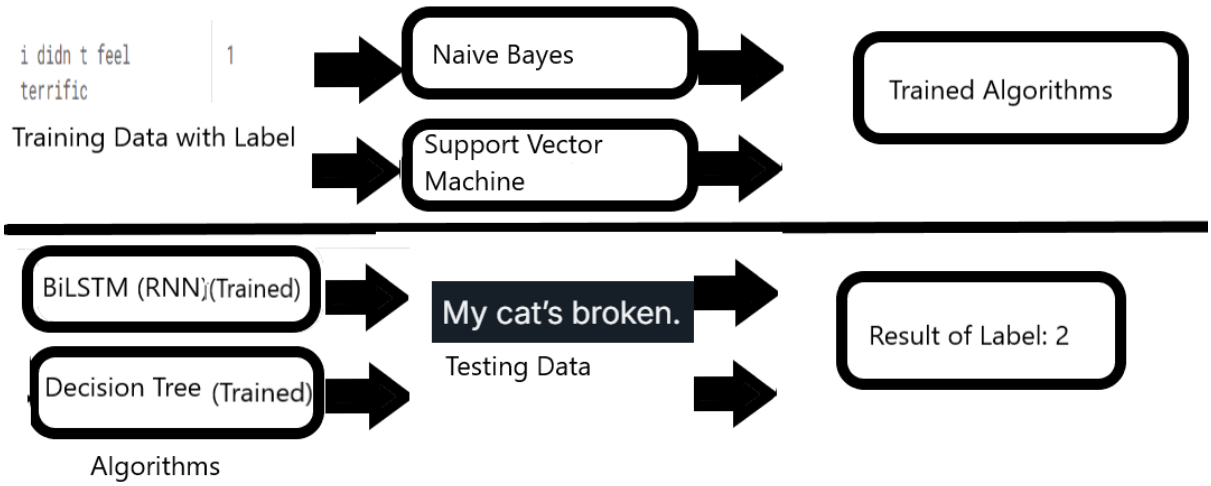


Figure – 8 Architecture of System

While having a rough plan is useful, unexpected challenges often arise in projects. The Scrum framework, combined with the Agile methodology, provides enough flexibility to adapt the plan as needed. The backlog supports these changes, as tasks can be moved up or down in priority based on new requirements or issues encountered during development.

For this project, I decided that Python would be the best choice of programming language. There are several reasons for this decision, including Python's flexibility and the strong support offered by its libraries. Additionally, Python can be used effectively as a backend for the web-based tool, making it ideal for this project. The Flask framework will be used, as it enables rapid development, which aligns well with the fast-paced nature of Agile methodology.

3.1 Dataset and Splitting

An appropriate dataset must be chosen to train the models and solve the problem for this project. The dataset should be open source and available for research purposes. It would be even better if it allows commercial use, in case it is needed for the web-based tool. While searching for a useful post-to-emotion dataset, I came across a very promising one that is open source, in the public domain, and available for commercial or research use (*Twitter Emotion Dataset*, n.d.). This dataset uses Ekman's six basic emotions (sadness, joy, love, anger, fear, and surprise) as labels and includes various social media posts updated a year ago, which are sufficiently recent. The training dataset does not include emojis or hashtags; they have likely been converted into text to facilitate training.

Dataset: https://www.kaggle.com/datasets/adhamelkomy/twitter-emotion-dataset

The dataset is provided in Comma Separated Values (CSV) format, making it easy to read and process using Python's Pandas library. After testing an 80/10/10 split and a

60/20/20 split—corresponding to training, validating, and testing—I decided on an 80/10/10 split. Given that the dataset contains 416,809 rows, training and validation can be performed using a lower percentage of the data.

The dataset has three columns: the first column is an ID field, which will be removed before training; the second column contains the post text, which serves as the input for training; and the third column contains the label (a number from 0 to 5 corresponding to Ekman's six basic emotions), which serves as the output for training.

The dataset distribution is unbalanced and biased toward sadness and joy (labels 0 and 1, respectively). Together, joy and sadness account for more than 200,000 rows. This may reflect an emotional bias in everyday life: love, anger, fear, and surprise are inherently less common. Love typically occurs only within family or romantic relationships; anger is a strong emotion that people experience less frequently; fear usually relates to specific objects, situations, or animals (phobias); and surprise only arises when an unexpected event occurs. Consequently, routine daily life involves fewer instances of these four emotions.

3.2 Comparison of Models and Techniques

The second objective was to select an appropriate feature extraction technique and the best-performing classification model. Starting with the techniques, two main approaches were considered: Bag of Words (BoW) and Term Frequency–Inverse Document Frequency (TF-IDF). According to a recent study ("Advances in Distributed Computing and Artificial Intelligence Journal : 9, Regular Issue 2, 2020," 2020), the BoW technique achieves a significantly better F1 score after text preprocessing. In contrast, TF-IDF tends to offer more stable performance, resulting in higher precision and accuracy. BoW appears particularly effective for imbalanced datasets, which is relevant in this case, as the Kaggle dataset shows a significantly higher number of samples for emotions like sadness and joy. Therefore, the BoW approach may be more suitable for this classification task.

To compare the machine learning models and determine the best performer, several evaluation metrics are considered. The four primary metrics include accuracy, precision, recall, and F1-score.

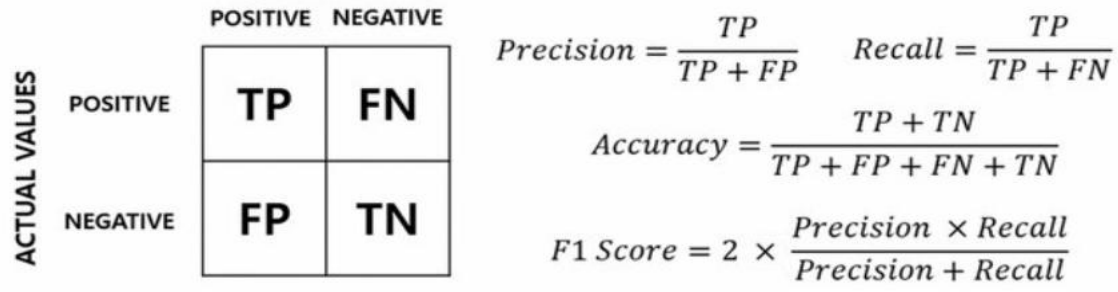


Figure – 9 Confusion matrix: Precision, Recall, Accuracy, and F1 score(Seol et al., 2023)

The figure above (Figure – 9) illustrates the formulas and definitions of each evaluation metric. Accuracy is simply the number of correct predictions divided by the total number of predictions, providing a general measure of performance. Precision is the ratio of true positives to the sum of true and false positives, and it is calculated per class. Recall is the ratio of true positives to the sum of true positives and false negatives, also specific to each class. The F1-score is the harmonic mean of precision and recall, and like the other two, it is calculated for each individual class.

For this project, the primary focus will be on F1-score and accuracy. Since the F1-score already incorporates both precision and recall, it is not necessary to analyse those metrics separately.

3.3 Web-based Tool

The web-based tool will include a text bar where the link to a post can be entered, along with a submit button to submit the selected post. Once the post is submitted, a positive/negative sentiment graph will appear, combining all the positive emotions (joy, love, and surprise) and all the negative emotions (sadness, fear, and anger). In addition to the sentiment frequency graph, there will be an emotion frequency graph that displays each emotion individually. The tool is designed to classify ten different replies for each post. Below the two graphs, the selected and classified posts will be displayed in a section labelled "Replies and Emotions."

The website will include a home page containing all the mentioned functionality and an About page explaining each feature. The posts are selected from the top ten based on relevance and recency. If a comment becomes outdated or irrelevant and a new one is added, the new comment will be prioritized.

To develop this tool, Flask will be used, as it is a Python framework that integrates well with all other tools used in the project. However, the tool will not be deployed to the cloud; it will run on a local machine for demonstration purposes.

4. Implementation

This section will include the code for the models as well as the code for the web tool. To achieve the second and third objectives, this section needs to be clear and well-structured. In the end, to access Bluesky via its API (AT Protocol), an application password was required. I generated an application password from the settings, allowing me to access the API directly through Python (Figure – 10).

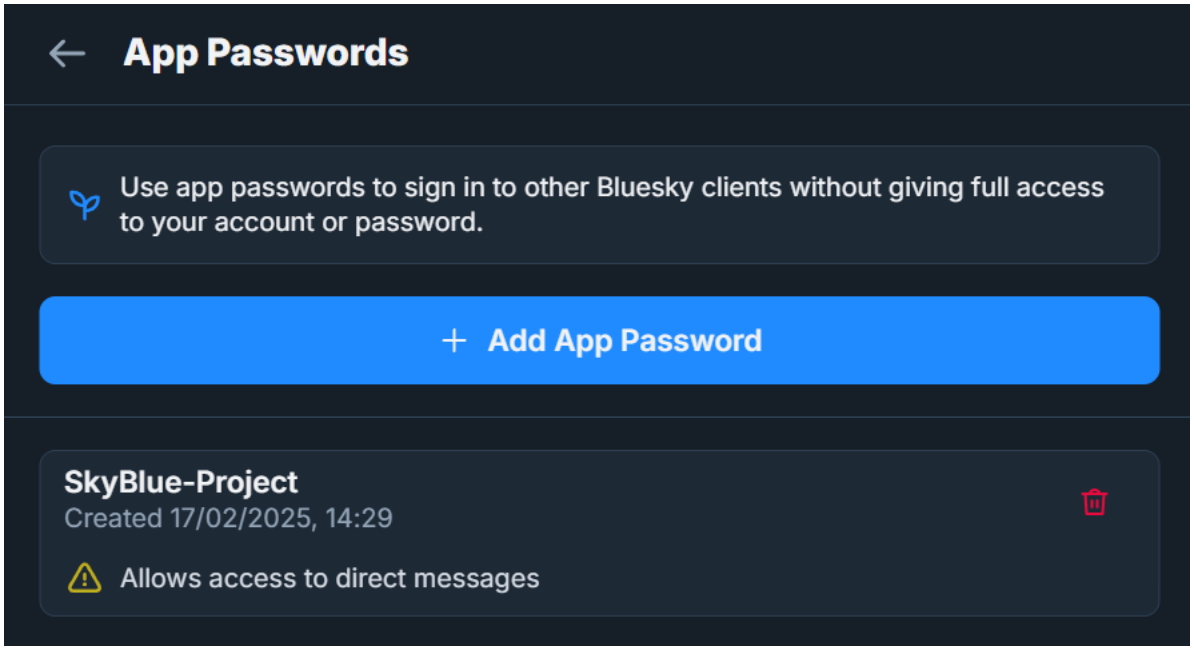


Figure – 10 Application Password of Bluesky

For the Flask application, I will use a Python file, as that is what Flask supports. However, to train the models, I used a Jupyter Notebook, as it allows me to quickly compare all the models.

4.1 Reading the Dataset and Preprocessing

The Pandas library and its read CSV function were used to read the dataset. The dataset consists of three fields: id, text, and label. Only the text and label fields were required, so the unnecessary id field was dropped (Figure – 11).

```
# Read the data and drop the first column
df = pd.read_csv('text.csv')

df = df.drop(columns=['Unnamed: 0'])

print(df)
```

Figure – 11 Reading the Dataset and Dropping the ID Field

As mentioned earlier, I made an 80/10/10 split (Figure – 12). To split the dataset, I used four different data frames: train, temporary, validate, and test. The temporary data frame was used to divide the remaining twenty per cent in half, allowing the creation of separate validation and test data frames.

```
# Split data into 3 parts train (80%), validate (10%) and test (10%)
train_df, temp_df = train_test_split(df, test_size=0.2, random_state=42, stratify=df['label'])

validate_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42, stratify=temp_df['label'])

print("Training set size:", train_df.shape)
print("Validation set size:", validate_df.shape)
print("Test set size:", test_df.shape)
```

Figure – 12 Splitting the Dataset

Removing the stop words was a crucial step in filtering and preprocessing the dataset. Using the Natural Language Toolkit (NLTK), I extracted all the English stop words and manually added another common word, “im,” which is often used in place of “I’m” in short posts (Figure – 13).

```
# Download the stopwords from nltk
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
stop_words.add('im')
print(stop_words)
```

Figure – 13 Stop Words

After extracting all the stop words to be removed, the next step was to iterate through each data frame that had been prepared (Figure – 14). To achieve this, the first task was to download PUNKT, which is used to split large texts into sentences. Afterwards, using iter rows, all the data frames were iterated over to remove each of the stop words identified earlier. To ensure consistency in text representation, all text was converted to lowercase inside the loop. Once a filtered sentence was stored in a list, the list was joined using a single space between each word. Finally, the correct index was checked to add the filtered sentence back into the same column.

```
# Remove the stopwords from each data frame
nltk.download('punkt')

for index, row in train_df.iterrows():
    word_tokens = word_tokenize(row["text"])
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
    filtered_sentence = " ".join(filtered_sentence)
    train_df.at[index, "text"] = filtered_sentence

for index, row in test_df.iterrows():
    word_tokens = word_tokenize(row["text"])
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
    filtered_sentence = " ".join(filtered_sentence)
    test_df.at[index, "text"] = filtered_sentence

for index, row in validate_df.iterrows():
    word_tokens = word_tokenize(row["text"])
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
    filtered_sentence = " ".join(filtered_sentence)
    validate_df.at[index, "text"] = filtered_sentence
```

Figure – 14 Removal of Stop Words

The final step in preprocessing is to apply one hot encoding, as some models, such as those used in deep learning, require a one-hot encoded field. One hot encoding takes all the possible classes and converts them into a vector where only the active class is marked as hot (1), while all other classes are marked as cold (0) for each row. This encoding is applied to all the labels in the train, test, and validate data frames generated from the CSV files (Figure – 15).

```
# Encoding the labels with one hot encoder
cat_encoder = OneHotEncoder()

array_label_train = train_df["label"].values
array_label_test = test_df["label"].values
array_label_validate = validate_df["label"].values

array_label_train = array_label_train.reshape(-1, 1)
array_label_test = array_label_test.reshape(-1, 1)
array_label_validate = array_label_validate.reshape(-1, 1)

label_cat_1hot_train = cat_encoder.fit_transform(array_label_train)
label_cat_1hot_test = cat_encoder.fit_transform(array_label_test)
label_cat_1hot_validate = cat_encoder.fit_transform(array_label_validate)

train_df["label"] = list(label_cat_1hot_train.toarray())
test_df["label"] = list(label_cat_1hot_test.toarray())
validate_df["label"] = list(label_cat_1hot_validate.toarray())
```

Figure – 15 One Hot Encoding

4.2 Machine Learning and Deep Learning Models

With preprocessing completed, it was time to move on to the models defined earlier. Each model follows a similar structure, where both the Bag of Words approach and the TF-IDF approach are used to allow for comparison during the results and evaluation stage (Figure – 16). It is worth mentioning that the deep learning approach is slightly different, as it involves creating a tokenizer that is saved as a JSON file as a precaution (Figure – 17). For the machine learning models, the process begins by defining the vectorizer—either as Count Vectorizer for the Bag of Words approach or as TF-IDF Vectorizer for the TF-IDF approach.

This is followed by hyperparameter tuning, where a few values are selected for the smoothing and regularization parameters. Once the best parameter is found, the algorithm stops running and outputs the results, including a confusion matrix visualized as a heatmap. The Matplotlib and Seaborn libraries are used to generate the heatmap, aiding in the visualization of results during the evaluation phase. Scikit-learn and TensorFlow libraries are used for the machine learning and deep learning models, respectively.

Overall, the deep learning (BiLSTM) model achieved the highest accuracy and performance, with an accuracy of around 90%. The machine learning models reached approximately 85% accuracy. However, it is important to note that each model, including the deep learning one, has its own advantages and disadvantages.

Additionally, accuracy alone does not fully define the performance of a model. These aspects will be explained in more detail in the results and evaluation section.

```python
# Use TF-IDF approach
vectorizer = TfidfVectorizer(max_features=1000)

X_train = vectorizer.fit_transform(train_df['text'])
X_test = vectorizer.transform(test_df['text'])
X_val = vectorizer.transform(validate_df['text'])

# Convert labels into numeric as naive bayes does not use one hot encoded labels
y_train = np.argmax(np.array(train_df['label'].tolist()), axis=1)
y_test = np.argmax(np.array(test_df['label'].tolist()), axis=1)
y_val = np.argmax(np.array(validate_df['label'].tolist()), axis=1)

# Hyperparameter tuning using validation dataset
alpha_values = [0.1, 0.5, 1.0, 5.0]
best_alpha = None
best_val_acc = 0
best_model_nb = None

for alpha in alpha_values:
    classifier_nb = MultinomialNB(alpha=alpha)  # Alpha is the smoothing parameter that can be adjusted
    classifier_nb.fit(X_train, y_train)

    y_val_pred = classifier_nb.predict(X_val)  # Predictions
    val_acc = accuracy_score(y_val, y_val_pred)  # Validation accuracy calculated

    print(f"Alpha={alpha} | Validation Accuracy: {val_acc:.4f}")

    # Remove the bad performing model until the best performing model is reached
    if val_acc > best_val_acc:
        best_alpha = alpha
        best_val_acc = val_acc
        best_model_nb = classifier_nb

print(f"\nBest Alpha: {best_alpha} with Validation Accuracy: {best_val_acc:.4f}")

# Test using the best model on set
y_test_pred = best_model_nb.predict(X_test)

print("\nFinal Test Accuracy:", accuracy_score(y_test, y_test_pred))
print("\nFinal Test Classification Report - Naive Bayes:\n", classification_report(y_test, y_test_pred))

# Confusion matrix calculation for further analysis of model on test set
test_cm_nb = confusion_matrix(y_test, y_test_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(test_cm_nb, annot=True, fmt='d', cmap='Blues', xticklabels=range(len(set(y_test))), yticklabels=range(len(set(y_test))))
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix - Test Set - Naive Bayes")
plt.show()
```

Figure – 16 Machine Learning Example Code

```
# Check if an available GPU exists
physical_devices = tf.config.list_physical_devices('GPU')
# Enable dynamic memory growth to save memory
tf.config.experimental.set_memory_growth(physical_devices[0], enable=True)


# Initalising training tokenizer structure and saving it as json for possible future use
vocab_size = 1000
max_length = 100
oov_token = "<OOV>"

tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_token)
tokenizer.fit_on_texts(train_df['text'].tolist())

X_train_seq = tokenizer.texts_to_sequences(train_df['text'].tolist())
X_val_seq   = tokenizer.texts_to_sequences(validate_df['text'].tolist())
X_test_seq  = tokenizer.texts_to_sequences(test_df['text'].tolist())

X_train = pad_sequences(X_train_seq, maxlen=max_length, padding='post', truncating='post')
X_val   = pad_sequences(X_val_seq, maxlen=max_length, padding='post', truncating='post')
X_test  = pad_sequences(X_test_seq, maxlen=max_length, padding='post', truncating='post')


tokenizer_json = tokenizer.to_json()
with open("tokenizer.json", "w") as json_file:
    json_file.write(tokenizer_json)


# Using the vocab size to convert one hot encoded labels into list structure
y_train = np.array(train_df['label'].tolist())
y_val   = np.array(validate_df['label'].tolist())
y_test  = np.array(test_df['label'].tolist())

num_classes = y_train.shape[1]
print("X_train: ", X_train[3])
print("y_train: ", y_train[3])
print("num_classes: ", num_classes)


# Designing of the RNN (Bidirectional LSTM) model
embedding_dim = 128

model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_length),
    SpatialDropout1D(0.2),
    Bidirectional(LSTM(128, return_sequences=True)),
    Dropout(0.3),
    Bidirectional(LSTM(128, return_sequences=True)),
    Dropout(0.3),
    Bidirectional(LSTM(64, return_sequences=False)),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(num_classes, activation='softmax')
])
```

Figure – 17 Deep Learning Example Code

4.3 Developing the Tool

To develop the web-based tool, it was important to use the same libraries that were used during the training process. The first task was to disable any GPUs, since unlike the training phase, the tool is only classifying 10 replies for each post's link. However, it is important to consider that for a tool intended for public release, a server equipped with industry-level GPUs would be necessary. The scope of this project is limited to running the web-based tool on a local machine, so a server is not required.

As mentioned earlier, the official API of Bluesky, known as ATPROTO, has been used as a web-based tool to manage clients and sessions (Figure – 18). This allows the tool to extract 10 replies after entering the API credentials. If no text is available and the reply consists of an image, video, or GIF, the program uses "No text available" to handle the attribute error. This is because the project is focused on text-based content, and processing images would require a completely different setup and algorithms.

```python
async def fetch_post_thread(link):
    """
    Input: A bluesky URL of a post
    Process: Extracting most popular 10 comments/replies via url_to_at_uri function
    Output: Most popular 10 comments/replies of the input URL post
    """
    uri = url_to_at_uri(link)
    print("Converted URI:", uri)
    client = AsyncClient("https://bsky.social")

    @client.on_session_change
    async def on_session_change(event: SessionEvent, session: Session):
        print("Session event:", event, session)

    # Log in using your API credentials
    await client.login("semih-cardiff.bsky.social", "[redacted]")
    res = await client.get_post_thread(uri=uri, depth=1)
    thread = res.thread

    # Extract text from up to 10 replies
    replies_text = []
    for reply in thread.replies[:10]:
        try:
            replies_text.append(reply.post.record.text)
        except AttributeError:
            replies_text.append("No text available")
    return replies_text
```

Figure – 18 Fetch Post Thread

The next step is to load the best-performing algorithm, which in this case is the Deep Learning (BiLSTM) model (Figure – 19). Along with the model, the tokenizer, classifier label list, maximum sequence length, and NLTK (for the same preprocessing steps) also need to be loaded. Each of these will be used in different functions to process the tweets, so they can ultimately be routed to the website pages. This step will be explained in detail in later sections.

```python
# Download required NLTK data
nltk.download('stopwords')
nltk.download('punkt')

# Create a set of stopwords for cleaning
stop_words = set(stopwords.words('english'))

# Mapping from classifier number to emotion labels
classifier_num_to_emotion = {
    0: "sadness",
    1: "joy",
    2: "love",
    3: "anger",
    4: "fear",
    5: "suprise"
}

# Load the trained model
model = load_model('RNN_Tweet_Classifier.h5')

# Load the tokenizer from the saved JSON file
with open('tokenizer.json', 'r') as f:
    tokenizer_json = f.read()
    tokenizer = tokenizer_from_json(tokenizer_json)

# Set the max sequence length used during training
MAX_SEQUENCE_LENGTH = 100
```

Figure – 19 Initialization of Algorithms

There are three main functions that run in the background of the web-based tool: clean text, classify text, and URL to at URI (Figure – 20).

Starting with the clean text function, the process involves several steps to handle the unprocessed text from a post. It begins by removing emojis and replacing them with descriptive text. This step is not performed on the training data, as it already contains only text. The second step removes any hashtags and converts the hashtag text into plain text. Again, this was not necessary for the training data, as it was already free of

hashtags. The third step combines the entire text into a single list and removes stop words. This step was applied to the training data as well since it contained many unnecessary words.

Moving on to the classify text function, it begins by taking processed but unclassified text. It then converts the cleaned text into a padded sequence based on the maximum sequence length defined earlier. This sequence length must match the one used during training. Next, the model is used to classify the comments under the post, and finally, the function returns the predicted emotion.

The final function is the URL to at URI. It starts by taking the URL of the post, which is the link copied and pasted into the web-based tool by the user. Once the URL is captured, it is parsed and separated into segments. The next step extracts the post ID and user tag from these segments. The function then returns a URI by combining the post ID and user tag into a standardized link.

```
def clean_text(text):
    """
    Input: Unprocessed text
    Process: Remove stopwords, Convert Emoji to useful text, Remove Hastags "#" but leave the useful text part
    Output: Cleaned text
    """
    text = emoji.demojize(text)
    text = re.sub(r'#(\S+)', r'\1', text)
    text = re.sub(r':([^:]+):', lambda m: " " + m.group(1).replace("_", " ") + " ", text)
    text = re.sub(r'\s+', ' ', text).strip()
    tokens = word_tokenize(text)
    filtered_tokens = [w for w in tokens if not w.lower() in stop_words]
    return " ".join(filtered_tokens)

def classify_text(text):
    """
    Input: Unclassified processed text
    Process: Labelling via trained RNN model
    Output: Label which is also the predicted emotion
    """
    # Clean the text exactly as done during training
    cleaned = clean_text(text)
    # Convert the cleaned text to sequences using the loaded tokenizer
    sequence = tokenizer.texts_to_sequences([cleaned])
    padded_sequence = pad_sequences(sequence, maxlen=MAX_SEQUENCE_LENGTH, padding='post', truncating='post')
    # Predict probabilities from the model
    prediction = model.predict(padded_sequence)
    class_index = np.argmax(prediction, axis=1)[0]
    predicted_emotion = classifier_num_to_emotion.get(class_index, "Unknown")
    return predicted_emotion

def url_to_at_uri(url):
    """
    Input: A bluesky URL of a post
    Process: Converting into URI
    Output: A blueksy URI of a post
    """
    parsed = urlparse(url)
    segments = parsed.path.strip('/').split('/')
    post_id = segments[-1]
    user_tag = segments[-3]
    return f"at://{user_tag}/app.bsky.feed.post/{post_id}"
```

Figure – 20 Main Functions

Using the app route function, URLs can be mapped to Python functionalities (Figure - 21). The route functions use GET and POST methods to retrieve and send data to the server. This allows the main functions explained above to be used in the background of the website itself.

```
# Home page which is also the main page
@app.route("/")
@app.route("/home", methods=['GET', 'POST'])
def home():
```

Figure – 21 Routing

The website itself functions similarly to the Wayback Machine, which is a tool that archives the entire internet. It features a simple user interface with a home page that includes a bar to paste the link and a submit button (Figure - 22). The other page is the About page, which explains how to use the tool (Figure - 23).

Figure – 22 Home Page

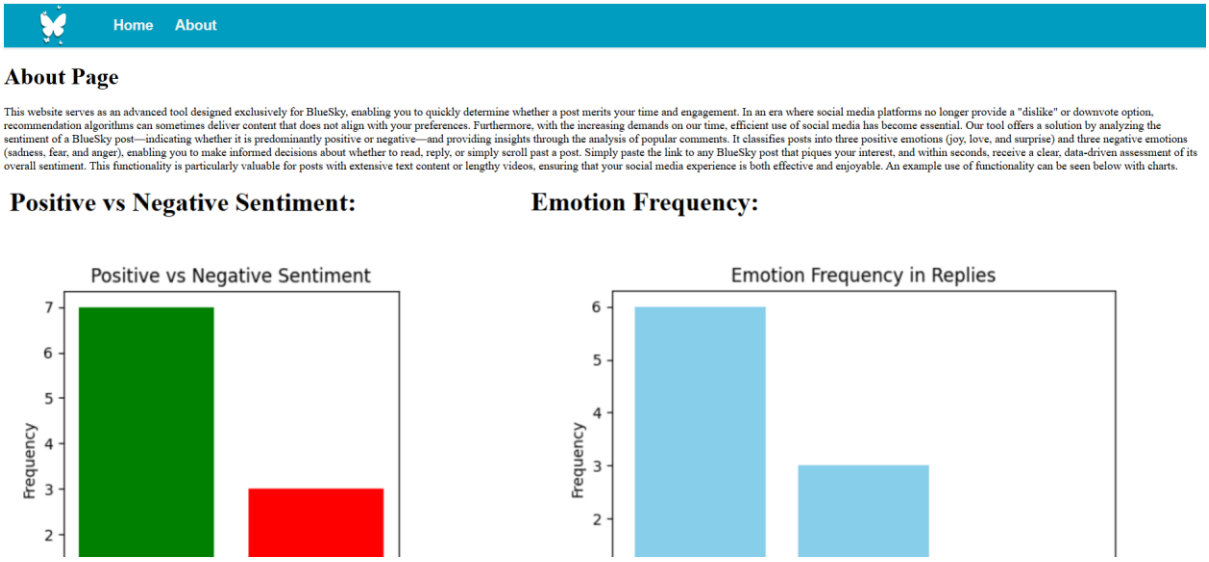


Figure – 23 About Page

5. Results and Evaluation

5.1 Evaluating Models

To evaluate the models I developed, I used all the main metrics: precision, recall, F1-score, and accuracy. For each of these metrics, both macro and weighted averages were used to evaluate performance across all classes. Additionally, a confusion matrix was generated for each model (Figures 24 to 32).

Starting with the highest accuracy, it is clear that the Bidirectional Long Short-Term Memory (BiLSTM) model achieved the best performance with a final accuracy of 89.85%. The lowest accuracy was 84.55%, recorded by the Naïve Bayes model using the TF-IDF approach. It is important to note that accuracy is the only metric reported for the deep learning model, as it operates differently from the machine learning models. This distinction will be explained in more detail later.

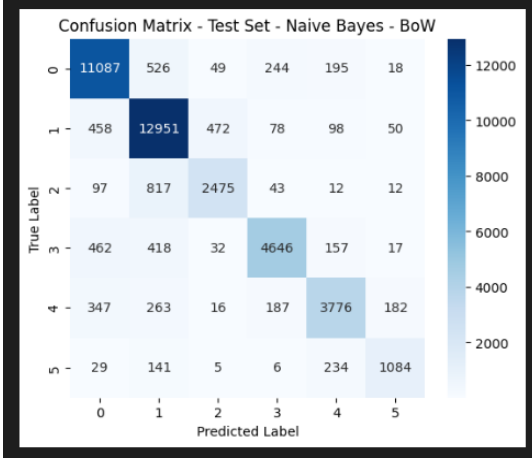
Moving on to the F1-score, the highest macro average (83%) was achieved by three models: Naïve Bayes with Bag of Words (BoW), and Support Vector Machine (SVM) with both BoW and TF-IDF. The lowest macro average F1-score was also recorded by Naïve Bayes with TF-IDF, which scored 78%. Looking at the F1-score based on the weighted average, the highest value (87%) was again achieved by SVM with BoW and TF-IDF. The lowest weighted average F1-score, 84%, was again observed with Naïve Bayes using TF-IDF.

These results show that Naïve Bayes with TF-IDF performs poorly in both macro and weighted F1-scores, which is concerning because the F1-score reflects both precision and recall. This underperformance is largely due to the model's very low macro average recall.

It is also important to note that during data splitting, I used a fixed random seed (random state = 42), which is commonly used in machine learning to ensure reproducibility. Setting the random state guarantees that the data is split the same way each time the code is run. This consistency is crucial because different splits could lead to different results, making fair comparisons between models impossible. This consistency is also reflected in the support values, as each model used the same distribution of data across classes.

In conclusion, I selected the BiLSTM model as the best-performing one. Deep learning models like BiLSTM utilize multiple layers and recurrent structures, allowing data to pass through several times, which improves performance significantly compared to traditional machine learning approaches. While the machine learning models averaged around 85% accuracy, the deep learning model reached approximately 90%, making the performance gap large enough to justify choosing the deep learning approach.

```
Alpha=0.1 | Validation Accuracy: 0.8637
Alpha=0.5 | Validation Accuracy: 0.8635
Alpha=1.0 | Validation Accuracy: 0.8638
Alpha=5.0 | Validation Accuracy: 0.8630

Best Alpha: 1.0 with Validation Accuracy: 0.8638

Final Test Accuracy: 0.8640965358410901

Final Test Classification Report - Naive Bayes:
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     12119
           1       0.86      0.92      0.89     14107
           2       0.81      0.72      0.76      3456
           3       0.89      0.81      0.85      5732
           4       0.84      0.79      0.82      4771
           5       0.80      0.72      0.76      1499

    accuracy                           0.86     41684
   macro avg       0.85      0.81      0.83     41684
weighted avg       0.86      0.86      0.86     41684
```




```
Alpha=0.1 | Validation Accuracy: 0.8454
Alpha=0.5 | Validation Accuracy: 0.8452
Alpha=1.0 | Validation Accuracy: 0.8448
Alpha=5.0 | Validation Accuracy: 0.8407

Best Alpha: 0.1 with Validation Accuracy: 0.8454

Final Test Accuracy: 0.8455282602437386

Final Test Classification Report - Naive Bayes:
              precision    recall  f1-score   support

           0       0.86      0.94      0.90     12119
           1       0.79      0.96      0.87     14107
           2       0.96      0.52      0.67      3456
           3       0.93      0.76      0.83      5732
           4       0.87      0.74      0.80      4771
           5       0.96      0.44      0.61      1499

    accuracy                           0.85     41684
   macro avg       0.89      0.73      0.78     41684
weighted avg       0.86      0.85      0.84     41684
```

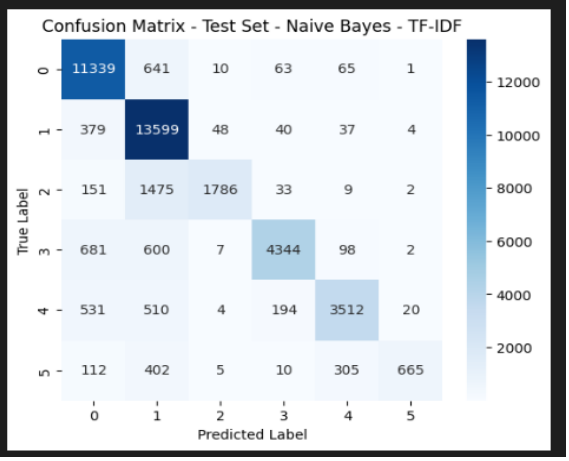
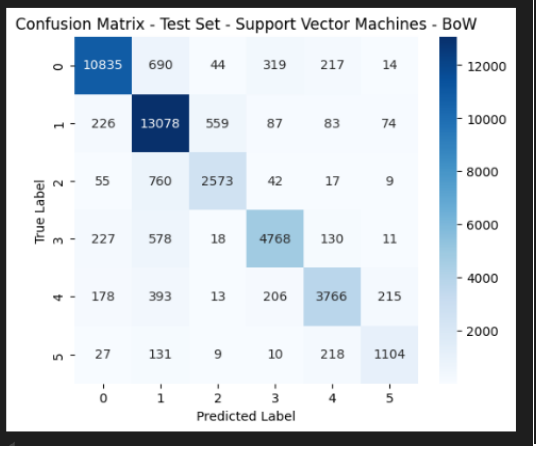


Figure – 24 and 25 Naïve Bayes

```
C=5.0 | Validation Accuracy: 0.8675

Best C: 0.5 with Validation Accuracy: 0.8676

Final Test Accuracy: 0.8666154879570099

Final Test Classification Report - Support Vector Machines:
              precision    recall  f1-score   support

           0       0.94      0.89      0.92     12119
           1       0.84      0.93      0.88     14107
           2       0.80      0.74      0.77      3456
           3       0.88      0.83      0.85      5732
           4       0.85      0.79      0.82      4771
           5       0.77      0.74      0.75      1499

    accuracy                           0.87     41684
   macro avg       0.85      0.82      0.83     41684
weighted avg       0.87      0.87      0.87     41684
```




```
C=0.1 | Validation Accuracy: 0.8674
C=0.5 | Validation Accuracy: 0.8680
C=1.0 | Validation Accuracy: 0.8682
C=5.0 | Validation Accuracy: 0.8683

Best C: 5.0 with Validation Accuracy: 0.8683

Final Test Accuracy: 0.8677430189041359

Final Test Classification Report - Support Vector Machines:
              precision    recall  f1-score   support

           0       0.93      0.90      0.92     12119
           1       0.84      0.93      0.88     14107
           2       0.81      0.74      0.77      3456
           3       0.88      0.83      0.86      5732
           4       0.84      0.79      0.82      4771
           5       0.77      0.72      0.75      1499

    accuracy                           0.87     41684
   macro avg       0.85      0.82      0.83     41684
weighted avg       0.87      0.87      0.87     41684
```

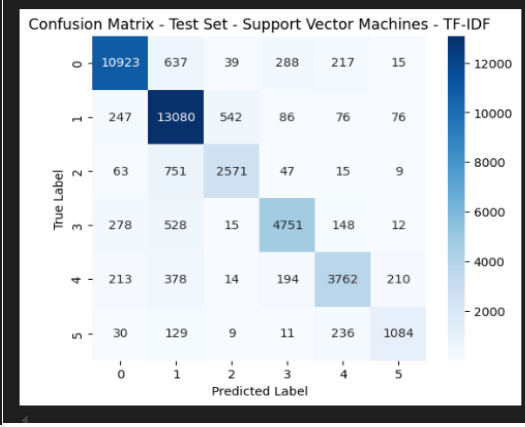


Figure – 26 and 27 Support Vector Machines

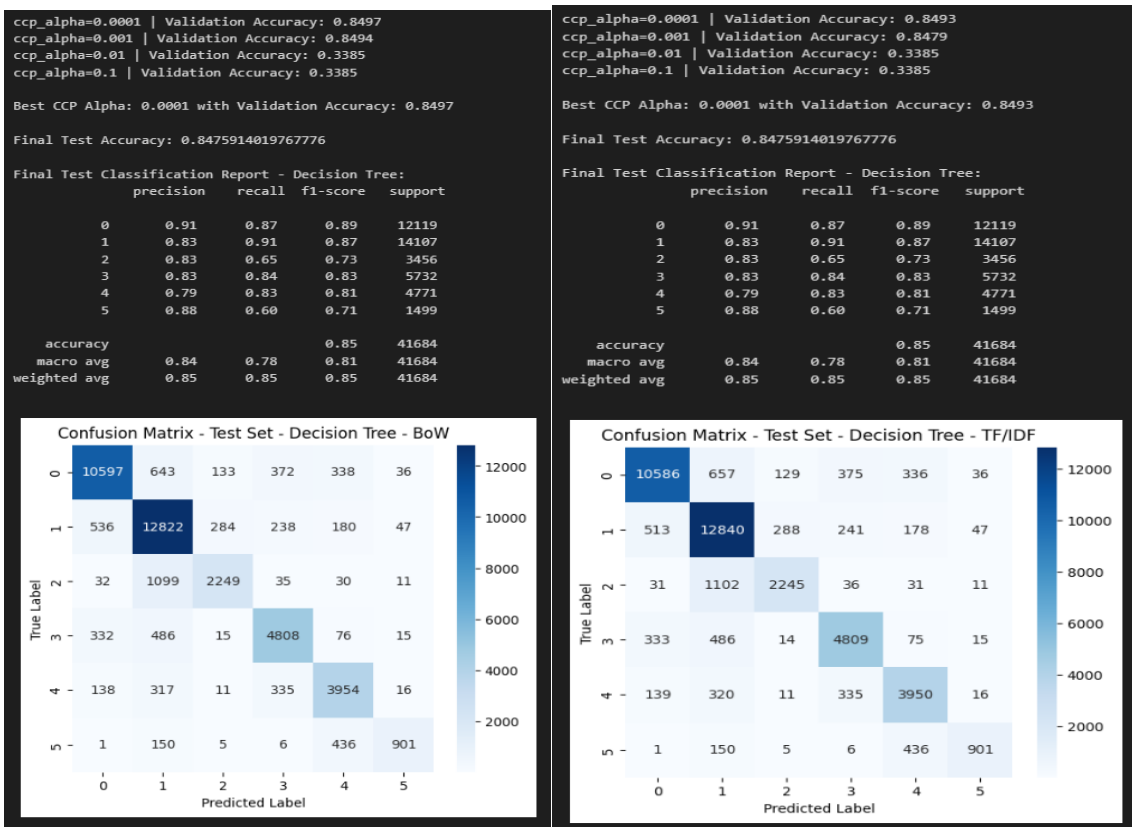
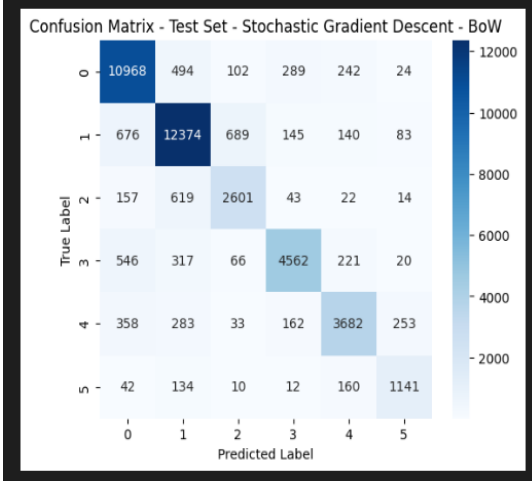


Figure – 28 and 29 Decision Tree

```
Alpha=1e-05 | Validation Accuracy: 0.8488
Alpha=0.0001 | Validation Accuracy: 0.8494
Alpha=0.001 | Validation Accuracy: 0.8477
Alpha=0.01 | Validation Accuracy: 0.8486

Best Alpha: 0.0001 with Validation Accuracy: 0.8494

Final Test Accuracy: 0.8475194319163228

Final Test Classification Report - Stochastic Gradient Descent:
              precision    recall  f1-score   support

           0       0.86      0.91      0.88     12119
           1       0.87      0.88      0.87     14107
           2       0.74      0.75      0.75      3456
           3       0.88      0.80      0.83      5732
           4       0.82      0.77      0.80      4771
           5       0.74      0.76      0.75      1499

    accuracy                           0.85     41684
   macro avg       0.82      0.81      0.81     41684
weighted avg       0.85      0.85      0.85     41684
```

```
Alpha=1e-05 | Validation Accuracy: 0.8478
Alpha=0.0001 | Validation Accuracy: 0.8464
Alpha=0.001 | Validation Accuracy: 0.8483
Alpha=0.01 | Validation Accuracy: 0.8469

Best Alpha: 0.001 with Validation Accuracy: 0.8483

Final Test Accuracy: 0.8494386335284522

Final Test Classification Report - Stochastic Gradient Descent:
              precision    recall  f1-score   support

           0       0.88      0.91      0.89     12119
           1       0.84      0.91      0.88     14107
           2       0.80      0.68      0.74      3456
           3       0.88      0.78      0.82      5732
           4       0.82      0.78      0.80      4771
           5       0.76      0.70      0.73      1499

    accuracy                           0.85     41684
   macro avg       0.83      0.79      0.81     41684
weighted avg       0.85      0.85      0.85     41684
```



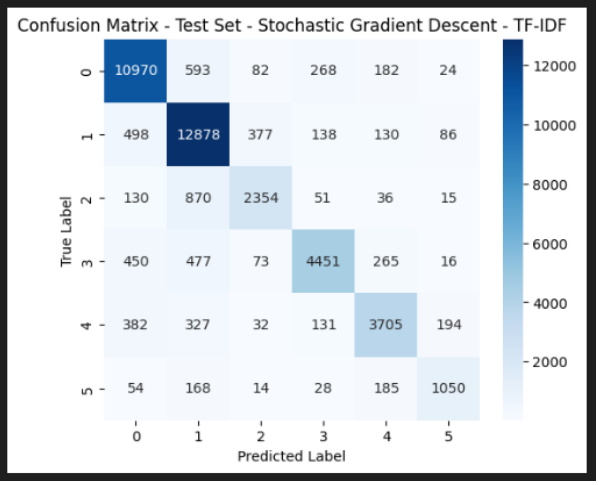


Figure – 30 and 31 Stochastic Gradient Descent

```
Epoch 1/5
5211/5211 [==============================] - 408s 76ms/step - loss: 0.3383 - accuracy: 0.8609 - val_loss: 0.2179 - val_accuracy: 0.8951
Epoch 2/5
5211/5211 [==============================] - 436s 84ms/step - loss: 0.2239 - accuracy: 0.8953 - val_loss: 0.2103 - val_accuracy: 0.8965
Epoch 3/5
5211/5211 [==============================] - 464s 89ms/step - loss: 0.2157 - accuracy: 0.8977 - val_loss: 0.2091 - val_accuracy: 0.8970
Epoch 4/5
5211/5211 [==============================] - 495s 95ms/step - loss: 0.2111 - accuracy: 0.8986 - val_loss: 0.2081 - val_accuracy: 0.8990
Epoch 5/5
5211/5211 [==============================] - 469s 90ms/step - loss: 0.2074 - accuracy: 0.8994 - val_loss: 0.2078 - val_accuracy: 0.8985
```

Figure – 32 BiLSTM

5.2 Results of the Tool

To analyse some results from the tool, I have chosen five different posts to help uncover how the tool is working and to see if anything is missing. Starting with post example 1 (Figures 33 to 35), it is possible to see that the tool generally works well, with some basic mistakes in classifying certain comments incorrectly. This also highlights some of the biggest issues of NLP models, which are difficulties in classifying jokes, exaggerations, and mocking comments. One example is the comment “Are those air biscuits?? 😆 ”, which explains the kneading move of cats from the cat image. However, the model classified it incorrectly, as jokes, exaggerations, and mockery are not easily understood by NLP models, even with deep learning algorithms.

Post example 2 (Figures 36 to 38) shows that the reverse can happen with positive emotions, where a person seems to be mocking. If we look at the comment “The perfect reason why drugs should not be advertised,” we can see that the person mocks drug advertisements and explains why they should not be done. This comment is supposed to be negative, as it is not joyful to see a drug advertisement. These misclassifications are expected because modern language models struggle to understand humour, which is heavily used on social media. However, when we look at the general sentiment, the model correctly identified that the post overall carries a negative tone, as no one with common sense likes drug advertisements due to their harmful nature.

Moving on to post example 3 (Figure 39 to 41), this example shows an entertaining post that was classified as negative due to wrong punctuation and lack of context in the comments. Checking the comment “Hello, I must be going!” we can see that it is an explanation of what the fish is doing in the video. However, without context and by looking only at the comment, it could be interpreted in many ways, such as a scene from a horror movie where a character is running away. As a result, it was misclassified as fear, which is incorrect.

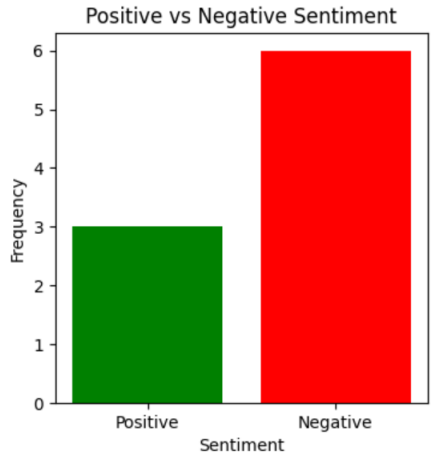
Looking at post example 4 (Figures 42 to 44), it is possible to see that some posts are emotionless, making it harder for the model to classify them. We can see this in the comment, “assuming Tinkle is named after what appear to be bells on their collar,” which is a neutral statement meant to inform rather than express emotion. The model, however, classified this as anger because it does not handle emotionless comments well. Some classification schemes include a "neutral" class for such sentences, but neutral is often overused in daily speech. Therefore, Ekman’s six emotions do not include a neutral category to maintain higher accuracy in emotion detection. This is an acceptable trade-off when focusing on pure emotion detection.

Moving on to post example 5, this one is an example where the tool worked almost perfectly. An interesting pattern was observed in this case. It is noticeable that the longest comment was the one misclassified as fear. This may be due to the training data consisting mostly of short posts or comments, leading the algorithm to associate many of the words with a fear context, even though this comment did not intend it. It shows that misclassifications can occur even without jokes or exaggerations, as no model is perfect.

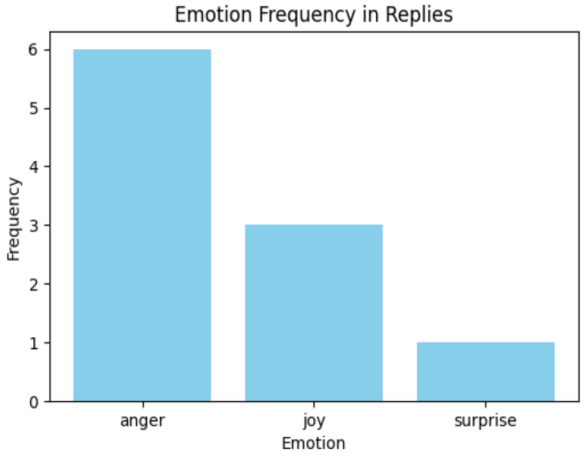
Overall, the model works well with some minor issues in classifying exaggerations, jokes, neutral posts, and long posts. These are expected issues even in larger models, as fully understanding natural language through mathematical expressions alone is impossible. Not providing background context may also lead to some misclassifications since the tool was designed only to assess comments. However, if background context were included, the tool would require significantly more time to classify and interpret the comments, making it beyond the scope of this project. Therefore, the tool is

intended solely to analyse comments to determine whether a post is generally positive or negative.

Positive vs Negative Sentiment:



Emotion Frequency:

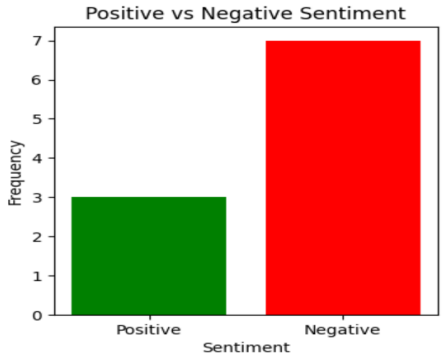


Replies and Emotions:

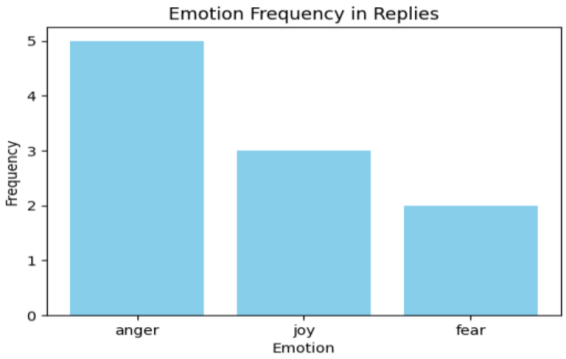
- It's cat yoga. Mine just perfected his Hummingbird Stretch. - **Classified as: anger**
- My cat looks like your cat - **Classified as: joy**
- So adorable - **Classified as: anger**
- It's cat-ching on - **Classified as: anger**
- Your cat is very good looking! - **Classified as: joy**
- Are those air biscuits?? 😆 - **Classified as: anger**
- 😬 - **Classified as: anger**
- I think that's what happened to my CatHole too!! - **Classified as: anger**
- 🐈‍⬛🐈😆 - **Classified as: joy**
- That's funny! Your cat looks like one of those dog toys with no stuffing, or a wadded towel, depending on the light. - **Classified as: surprise**

Figure – 33, 34 and 35 Post Example 1

Positive vs Negative Sentiment:



Emotion Frequency:

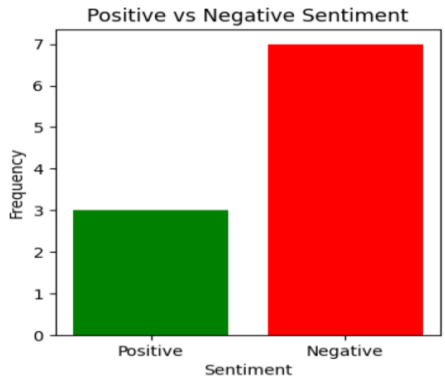


Replies and Emotions:

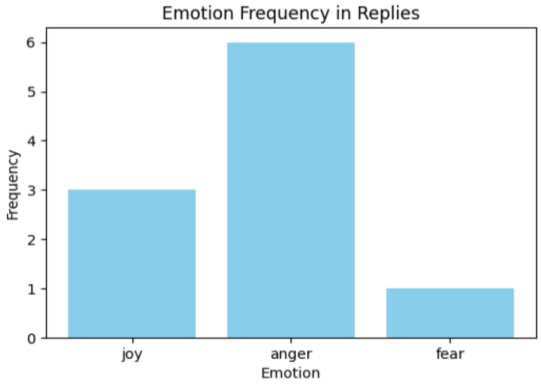
- Hurry home kids, it's almost time! - **Classified as: anger**
- ".. This commercial was written and directed by a sentient bag of cocaine" - **Classified as: anger**
- 🌞 They've made the pill look like the Teletubbies Sun-Baby. 🌞 - **Classified as: joy**
- The perfect reason why drugs should not be advertised. - **Classified as: joy**
- I'll chime in with the others and ask... "Why don't it tell us what this Claritin does?!" Because if I take it and I don't have an ego death mind trip... I would feel so confused. - **Classified as: fear**
- what's this shit even for??????? - **Classified as: anger**
- flight or flight. only flying is an option - **Classified as: anger**
- I love how the last trippy shot is a door floating in clouds opens to a bright white light - **Classified as: joy**
- Claritin - Now with 50% more LSD™ - **Classified as: anger**
- This is such a weird commercial, and I..... MUST BUY CLARITIN. CLARITIN IS PEACE. CLARITIN IS LOVE. MORE CLARITIN TO PURCHASE NOW PLEASE. - **Classified as: fear**

Figure – 36, 37 and 38 Post Example 2

Positive vs Negative Sentiment:



Emotion Frequency:

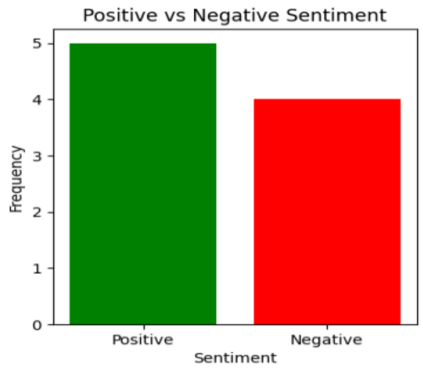


Replies and Emotions:

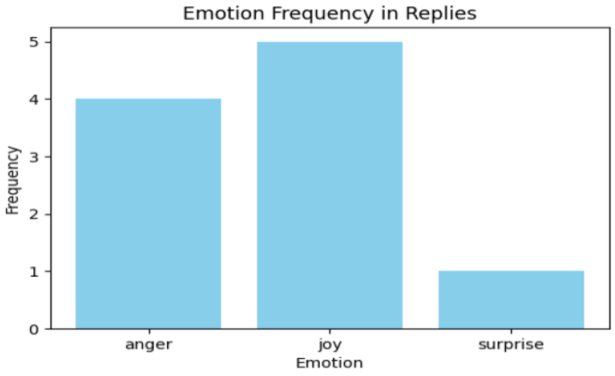
- Beautiful I think my blood pressure just went down 😊 - **Classified as: joy**
- it reminds me of someone ... can't remember who ... - **Classified as: anger**
- I can't stop watching this! So calming 😀 - **Classified as: anger**
- Hello, I must be going! - **Classified as: fear**
- Slidin in to a dm. - **Classified as: anger**
- HEY GURL - **Classified as: anger**
- Me on roller skates… - **Classified as: anger**
- Growing up in South Florida we called those cow-fish - **Classified as: anger**
- Love it! 😍 - **Classified as: joy**
- 😂 That's not a fish--it's a skkrrttt-ing iron with fins. - **Classified as: joy**

Figure – 39, 40 and 41 Post Example 3

Positive vs Negative Sentiment:



Emotion Frequency:

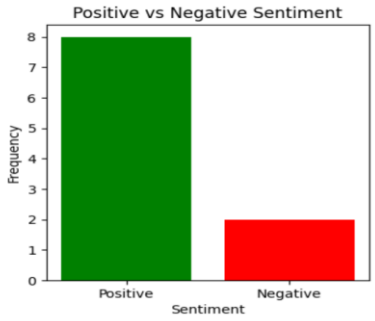


Replies and Emotions:

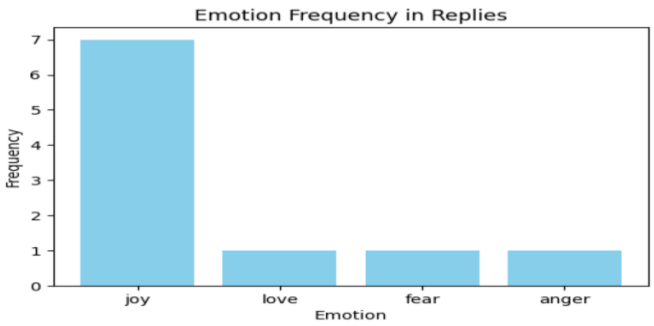
- DUSTER CAT - **Classified as: anger**
- I love it. What a beautiful painting. Hope it's on display now. - **Classified as: joy**
- Free tinkle! (Love her double bells) - **Classified as: joy**
- Make a print, frame it, and simply stick it on the wall when a museum guard isn't looking. There's temporary double side tape that are easily removable without marring the wall. youtu.be/m-OyRZoNb6o?... - **Classified as: anger**
- Sadly, Tinkle is not on view this year. The last time it was on display was in my "Creature Comforts" exhibition in 2020. Here's a photo. - **Classified as: joy**
- My mother-in-law painted her own version of Tinkle in the 70s. - **Classified as: anger**
- I like Tinkle she is beautiful- some of the other paintings kinda scare me😂 - **Classified as: joy**
- How intriguing! Fingers crossed this is the year "Tinkle" makes an appearance. A hidden gem from 1883? I'd love to see it in person! - **Classified as: joy**
- Oh, I went to that museum last year, it was amazing! Really loved the steam ship. - **Classified as: surprise**
- assuming Tinkle is named after what appear to be bells on their collar. - **Classified as: anger**

Figure – 42, 43 and 44 Post Example 4

Positive vs Negative Sentiment:



Emotion Frequency:



Replies and Emotions:

- Lovely! Look at its little tongue right at the end there - mlem mlem! - **Classified as: joy**
- Oh my, gorgeous! - **Classified as: joy**
- Loving this chocolate brown, fuchsia combo 🌺 - **Classified as: love**
- I have to follow for more of this content 🙀 - **Classified as: joy**
- I miss my garden in our previous home. So many bougainvillea and hibiscus and desert roses and gardenia and royal poinciana. We had the intense pink ones both in the front and back yards....ah, miss that. - **Classified as: fear**
- I was not prepared for something that beautiful. - **Classified as: joy**
- Gorgeous - **Classified as: joy**
- You just made my day - **Classified as: joy**
- www.hummingbirdcentral.com/hummingbird-... I will be on the lookout for Ruby-Throated Hummingbird. - **Classified as: anger**
- If you got bougainvillea advice I'll take it - **Classified as: joy**

Figure – 45, 46 and 47 Post Example 5

5.3 Evaluation of Objectives

To see how the project's first goal has been reached, we can observe that analyzing multiple research papers provided useful information about the assessment of online content. After the research, it was clear that this goal had been achieved by understanding that the content quality assessment has changed over time and that removing or not adding the dislike button was necessary to prevent certain issues on social media platforms. Another useful information gained from the research papers was that comments can provide valuable information, as many people already rely on them before purchasing something, suggesting that comments can be effectively used to analyse content quality. Along with the removal of the dislike button, the research also indicated that in the future, even the like button—which is considered an upvote or positive vote—may be removed, as companies realize that these features can drive some content creators toward greed, potentially leading to major societal issues.

Moving on to the second goal, which was choosing the appropriate machine learning or deep learning algorithm along with the correct text analysis technique, I believe I have successfully achieved this goal. I tested the five algorithms I mentioned, including the Bidirectional Recurrent Neural Network with Long Short-Term Memory (RNN with BiLSTM). I documented my work with the algorithms in a Jupyter Notebook, which allowed me to test the code cell by cell. I also tested two text analysis techniques—Term Frequency–Inverse Document Frequency and Bag of Words—and found that TF-IDF was more useful for some algorithms, while Bag of Words performed better for others. In the end, the decision was to use the BiLSTM Recurrent Neural Network with its tokenizer, as it had the best performance compared to the other algorithms.

For the third goal, my task was to build a web-based tool that uses the best-performing trained algorithm in its backend. Along with the backend, the web-based tool also needed a good user interface that would allow users to see the analysis results in multiple graph formats. This goal was achieved with the implementation of two graphs and corresponding results: one graph shows the overall positive and negative sentiments, the other graph shows the emotion frequency, and the results section displays the most recent 10 comments that were automatically selected and classified.

Overall, the goals set at the beginning of this project have been reached without missing anything major. This project successfully bridges the gap between the research component and the practical part, where the tool was built.

6. Conclusions and Future Work

6.1 Conclusions

This final report is divided into three parts overall. The first part is the pure research section, where I researched how the removal of the downvote or dislike button, or not adding a downvote or dislike button at all, affects the assessment of content quality on social media platforms. The second part is research by practice, where I attempted to script five different machine learning and deep learning algorithms to see which one achieves the best performance among them. The third part is the pure application section, where I built a web-based tool that allows users to analyse posts using only their links. All these components build the entire project overall, and they are interconnected.

Overall, I believe I have achieved all the goals that were defined at the beginning. I have also developed a practical tool that may have potential future applications, not just for users but also for research purposes, such as studying hate in social media or analysing the Bluesky platform and its user base specifically. Bluesky is a brand-new platform with a fresh set of users, many of whom have migrated from the platform previously known as Twitter and now called X. While completing this project, I learned about Bluesky's API, called the AT Protocol SDK. Learning Bluesky's official API was an important step, as it was the legal and proper way to build the web-based tool. Scraping with other tools such as Beautiful Soup could potentially violate the platform's rules. There are other social media platforms connected with Bluesky that also use the AT Protocol SDK as their API, so this could become part of future work and applications.

Although the API allows me to scrape the data legally, I believe it is still unethical to harvest users' data for such purposes. There are differing opinions in ethics: some argue that if a user posts a comment, they intend it to be seen and potentially used; others point out that some users seek only social interaction and do not want their comments employed for research. This project did not include any additional ethical review because the training dataset was publicly available for both research and commercial use, provided it is properly cited. Likewise, the Bluesky API permits the use of its users' data without legal issues; however, as discussed above, this practice may nonetheless be ethically questionable depending on user expectations.

An important point to consider in this project is that I used posts from Twitter (or X) as my training data. Since Bluesky is a new platform created by the former owner of Twitter or X, it shares many similarities in how posts are written and shared. Currently, no labelled dataset exists for Bluesky, so utilizing data from a similar platform was the only practical option.

There were some aspects that were beyond the scope of this project, such as enabling the trained model to understand complex concepts like humour or exaggeration. These concepts are already quite hard to teach a model; even modern transformers that require advanced hardware struggle with understanding humour, especially due to cultural differences within the English language, which is widely used globally.

In conclusion, the project demonstrates the effectiveness and application of using text classification to achieve sentiment analysis and highlights how it is possible to find new ways to assess content quality. The project successfully provided valuable charts via comment analysis into the field of text classification applied to social media.

6.2 Future Work

I believe I have achieved all the aims, objectives, and goals that I defined at the beginning of this project. However, close to finishing the project, I realized that many potential points and areas could be improved for future work in the same field.

The first and foremost area where I feel improvements can be made is in the Deep Learning algorithm. For this project, due to hardware limitations, I had to use Long Short-Term Memory (LSTM) based Recurrent Neural Networks (RNNs). In the future, if I undertake a similar project or directly continue this one to extend it further, I will prefer to use a transformer model, assuming I have access to more powerful hardware, such as a stronger GPU. Another improvement that would come with better hardware is the ability to use a larger dataset. Currently, I used a dataset of approximately 417,000 entries, which took around 25–30 minutes to train the deep learning model, while the machine learning models took approximately 5 minutes each.

In the future, I would also like to work on analysing platforms with higher attention rates and longer content. Thus, planning on cross-platform plugins would allow this classifier to work with many different platforms giving wide range of choices for users of the tool. Platforms like X or Bluesky typically offer shorter content except for floods or videos, whereas platforms like Netflix and YouTube provide longer, higher-attention content, making this tool much more valuable for those types of platforms.

Processing more comments is another future improvement I could implement. The current limit of 10 comments per post is relatively small compared to processing 100 comments, which would be possible with better hardware. It is also important to consider that sharing the tool publicly would require a dedicated server capable of handling the background processing for numerous requests.

Adding multi-language support to my tool will be one of the steps I take in future work. That way, anyone in the world can use my tool; currently it is limited to English speakers.

After feeding the model the most widely spoken languages, an online learning feature can be added so that the model improves with each user correction.

Processing other data types is another feature I might add. Images, for example, are used extensively on platforms like X and Bluesky. A Bag of Visual Words (BoVW) pipeline could detect when a comment includes an image and analyse it by converting its content into text. A similar feature could be added for videos, since videos are sequences of images with sound, by incorporating a speech-recognition pipeline alongside the (BoVW) approach.

In short, for future work, I plan to add support for multiple platforms (not just Bluesky) and multiple languages, so that anyone, not just English speakers, can use this tool. To maintain or even improve accuracy, I will implement online learning, which will also benefit multi-language support. Finally, I can adopt a transformer-based model for much better accuracy in the future, provided I have access to more powerful hardware.

6.3 Other Similar Literature

Other related works exist in the field of social-media text classification, and in this part of the project, I will analyse them, compare them with my research and results, and identify any differences or similarities. This analysis will be useful both for my conclusions and for future work by myself and others.

Starting with the “Emotion Detection in Roman Urdu Text using Machine Learning” paper (Majeed et al., 2020), this study is very similar to my project in that it employs algorithms such as Decision Tree and SVM, but it covers two different languages and does not use a social-media application for later testing. One key difference is that Roman Urdu is among the hardest languages for emotion detection—a point the authors confirm. They explain that Word2Vec, a word-embedding technique, requires a powerful GPU. Even with Word2Vec, the complexity of Roman Urdu makes achieving good results difficult. According to the paper, the scarcity of Roman Urdu datasets is another issue, which makes sense since, although Roman Urdu is popular, English remains the most widely used language today.

One similarity between that paper and my project is that, among non-deep-learning algorithms, SVM dominates both accuracy and F1-score measures. This is likely because SVM leverages support vectors in high-dimensional spaces—a crucial advantage for vector-based tasks like text classification, where texts are represented as collections of vectors.

Another paper I will explore is “Emotion Detection and Recognition from Text using Machine Learning,” which is more closely related to my project because it uses a social-media platform—Twitter (now X) (Salam & Gupta, 2018). The main difference is that the authors used X’s old free API, whereas today X’s API requires a subscription even for

small projects. This paper also employs algorithms similar to those in my project—SVM and Naïve Bayes—and again finds that SVM delivers the best performance. It's worth noting that my project and these two studies all use a linear SVM kernel, which is generally inferior to a Gaussian kernel; however, due to long training times, we have all opted for the linear kernel as the more practical choice.

Overall, there are several similar projects and papers, and it is highly likely that more researchers will enter the rapidly growing field of text classification, for purposes such as email spam detection or social-media sentiment analysis, as in this project. This literature review is critical because it surveys related work and highlights common patterns in text classification.

7. Reflection on Learning

From the beginning, I approached this project with both excitement and curiosity. While I had a strong foundation in programming, I was eager to challenge myself by venturing into unfamiliar territory, especially Natural Language Processing (NLP) and Machine Learning. This blend of new learning and prior experience made the project both engaging and rewarding.

Although I had little practical knowledge of NLP and Machine Learning, these fields had fascinated me for a long time. Working on this project allowed me to explore them in depth, and that process felt both intellectually stimulating and personally fulfilling. I found myself constantly learning—not just about algorithms and code, but about how language, emotion, and context intersect in fascinating ways.

One of the biggest surprises was how integral other disciplines were to my work. A whole section and one of the objectives focused on psychological and social science research, which initially felt outside my comfort zone. However, diving into these subjects broadened my understanding of how human behaviour and emotion influence technology. I began to appreciate how important interdisciplinary awareness is in AI development, especially when dealing with sensitive subjects like emotion detection.

Technically, working with deep learning models in TensorFlow was a major learning curve. I had never used TensorFlow before, so I struggled at first with its syntax and logic. There were moments when I spent hours debugging just to get a single model to run. But the eventual success—watching the model make accurate predictions—was deeply satisfying and gave me a sense of achievement.

My supervisor, Professor Irena Spasic, supported my development in subtle but meaningful ways. Her advice about being free to work on different parts of the project using a Scrum-based method gave me flexibility, and her recommendations, such as using tools like ChatGPT, Mendeley, and Grammarly, helped me polish my work more efficiently. For grammar correction, I used ChatGPT with the prompt "Fix grammar and

sentence structure without changing anything else," which ensured my original meaning was preserved. I then used Grammarly as a second layer of proofreading. To reflect on my progress, I saved both the raw and corrected versions of the report, which helped me see my improvement and refine my writing for future projects.

Learning core machine learning concepts like Naïve Bayes and Support Vector Machines (SVM) was both challenging and rewarding. I spent countless hours watching YouTube lectures and reading online blogs, often re-watching the same segments until things clicked. There were moments of frustration, especially when the theory seemed abstract or contradictory across sources. However, once I began applying what I had learned—testing models, comparing outputs, and tuning parameters—I felt real growth. Understanding the mathematical foundations behind the algorithms helped me make more informed decisions about what to adjust and how much to tweak. That level of control felt empowering.

There were also personal challenges. At times, I doubted whether I could keep up with the pace of the learning required. But each small win—a successful model run, a new insight into algorithm behaviour, or improved classification accuracy—helped build my confidence. The experience taught me to be patient with uncertainty and to treat failure as a natural part of problem-solving.

Ultimately, this project confirmed what I suspected going in: I want to pursue a career in machine learning and NLP. I now feel confident that I have not only the mathematical skills to understand how the algorithms work, but also the programming and analytical skills to implement and adapt them in real-world applications. This project helped me find direction, and I'm now actively considering both postgraduate study and employment opportunities in these fields.

In reflection, this project has been more than just an academic task—it has been a transformative learning experience. It helped me grow not just as a computer scientist, but as a thinker, a problem-solver, and someone who is now excited about what the future holds.

References

Advances in Distributed Computing and Artificial Intelligence Journal : 9, Regular Issue 2, 2020. (2020). *Advances in Distributed Computing and Artificial Intelligence Journal. - Quadrimestrale = Four-Monthly*, *9, regular issue*(2), 1–114.

Bayes Theorem - Statement, Proof, Formula, Derivation & Examples. (n.d.). Retrieved March 26, 2025, from https://byjus.com/maths/bayes-theorem/

Bidirectional Long Short-Term Memory Network - an overview | ScienceDirect Topics. (n.d.). Retrieved March 29, 2025, from

https://www.sciencedirect.com/topics/computer-science/bidirectional-long-short-term-memory-network

Chevalier, J. A., & Mayzlin, D. (2006). The Effect of Word of Mouth on Sales: Online Book Reviews. *Journal of Marketing Research*, *43*(3), 345–354. https://doi.org/10.1509/JMKR.43.3.345

DecisionTreeClassifier — scikit-learn 1.6.1 documentation. (n.d.). Retrieved March 28, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Discover — Bluesky. (n.d.). Retrieved January 28, 2025, from https://bsky.app/

Efficient Optimi: Mastering Stochastic Gradient Descent. (n.d.). Retrieved March 29, 2025, from https://statusneo.com/efficient-opti-mastering-stochastic-gradient-descent/

Eisenstein, J. (2019). *Introduction to Natural Language Processing. Adaptive Computation and Machine Learning serie*. 536. https://mitpress.mit.edu/books/introduction-natural-language-processing

Gerlitz, C., & Helmond, A. (2013). The like economy: Social buttons and the data-intensive web. *New Media and Society*, *15*(8), 1348–1365. https://doi.org/10.1177/1461444812472322/ASSET/IMAGES/LARGE/10.1177_1461444812472322-FIG2.JPEG

LinearSVC — scikit-learn 1.6.1 documentation. (n.d.). Retrieved March 26, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Majeed, A., Mujtaba, H., & Beg, M. O. (2020). *Emotion Detection in Roman Urdu Text using Machine Learning*. https://doi.org/10.1145/3417113.3423375

Muchnik, L., Aral, S., & Taylor, S. J. (2013). Social influence bias: A randomized experiment. *Science*, *341*(6146), 647–651. https://doi.org/10.1126/SCIENCE.1240466/SUPPL_FILE/MUCHNIK.SM.PDF

MultinomialNB — scikit-learn 1.6.1 documentation. (n.d.). Retrieved March 26, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

RNN vs LSTM vs GRU vs Transformers | GeeksforGeeks. (n.d.). Retrieved March 29, 2025, from https://www.geeksforgeeks.org/rnn-vs-lstm-vs-gru-vs-transformers/

Salam, S. A., & Gupta, R. (2018). Emotion Detection and Recognition from Text using Machine Learning. *International Journal of Computer Sciences and Engineering*, *6*(6), 341–345. https://doi.org/10.26438/IJCSE/V6I6.341345

Seol, D. H., Choi, J. E., Kim, C. Y., & Hong, S. J. (2023). Alleviating Class-Imbalance Data of Semiconductor Equipment Anomaly Detection Study. *Electronics (Switzerland)*, *12*(3). https://doi.org/10.3390/ELECTRONICS12030585

SGDClassifier — scikit-learn 1.6.1 documentation. (n.d.). Retrieved March 29, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier

Shivhare, S. N., & Saritha Khethawat, P. (n.d.). *EMOTION DETECTION FROM TEXT*.

Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for agile methodology. *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, *2017-January*, 864–869. https://doi.org/10.1109/CCAA.2017.8229928

Support Vector Machines (SVM) clearly explained: A python tutorial for classification problems... | Towards Data Science. (n.d.). Retrieved March 26, 2025, from https://towardsdatascience.com/support-vector-machines-svm-clearly-explained-a-python-tutorial-for-classification-problems-29c539f3ad8/

Text Classifiers in Machine Learning: A Practical Guide. (n.d.). Retrieved March 28, 2025, from https://levity.ai/blog/text-classifiers-in-machine-learning-a-practical-guide

Turel, O., & Qahri-Saremi, H. (2024). Role of "Likes" and "Dislikes" in Influencing User Behaviors on Social Media. *Journal of Management Information Systems*, *41*(2), 515–545. https://doi.org/10.1080/07421222.2024.2340829

Twitter Emotion Dataset. (n.d.). Retrieved April 10, 2025, from https://www.kaggle.com/datasets/adhamelkomy/twitter-emotion-dataset

Williams, L., Arribas-Ayllon, M., Artemiou, A., & Spasić, I. (2019). Comparing the Utility of Different Classification Schemes for Emotive Language Analysis. *Journal of Classification*, *36*(3), 619–648. https://doi.org/10.1007/S00357-019-9307-0/TABLES/13

Zhang, M. M., & Ng, Y. M. M. (2024). Beyond dislike counts: How YouTube users react to the visibility of social cues. *Https://Doi.Org/10.1177/14614448241287510*. https://doi.org/10.1177/14614448241287510

Appendices

Appendix A: Tools and Dataset

Dataset:

https://www.kaggle.com/datasets/adhamelkomy/twitter-emotion-dataset

Grammer Check Tools:

https://openai.com/index/chatgpt/

https://app.grammarly.com/

Referencing Tool:

https://www.mendeley.com/?interaction_required=true

Programming Language:

https://www.python.org/

Libraries:

https://www.nltk.org/

https://pandas.pydata.org/

https://www.tensorflow.org/

https://matplotlib.org/

https://scikit-learn.org/

https://numpy.org/

https://seaborn.pydata.org/

https://flask.palletsprojects.com/en/stable/

Appendix B: Posts Chosen for Analysis

Post Example 1:
https://bsky.app/profile/mrrickygervais.bsky.social/post/3lnagdaq26s2f

Post Example 2:

https://bsky.app/profile/maxkriegervg.bsky.social/post/3lnk3ub3s7c2y

Post Example 3:

https://bsky.app/profile/coralcitycamera.bsky.social/post/3lnkz2ftbf22s

Post Example 4:

https://bsky.app/profile/catsofyore.bsky.social/post/3lnqe62vq722o

Post Example 5:

https://bsky.app/profile/nnedi.bsky.social/post/3lnqi7on4ks2v

Appendix C: Charts and Images

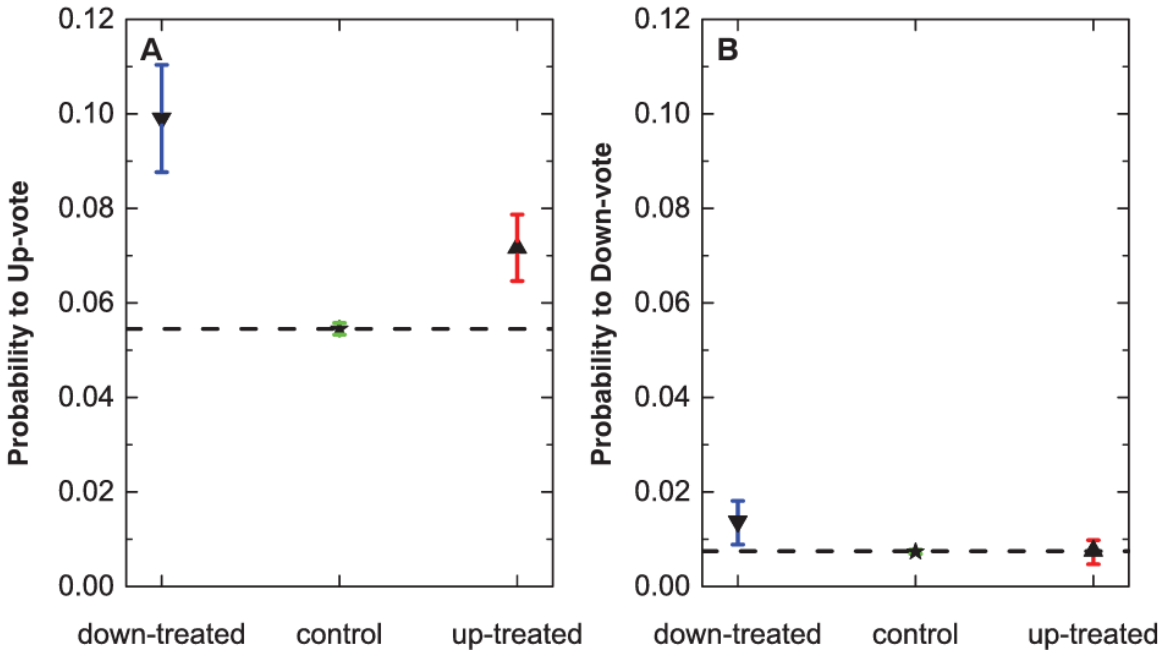


Figure 1 – Manipulated Up-votes and Down-votes probability (Muchnik et al., 2013)

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

Figure 2 – Bayes Theorem(*Bayes Theorem - Statement, Proof, Formula, Derivation & Examples*, n.d.)

$$w_1 x_1 + \ldots + w_d x_d + \beta_0 = 0,$$

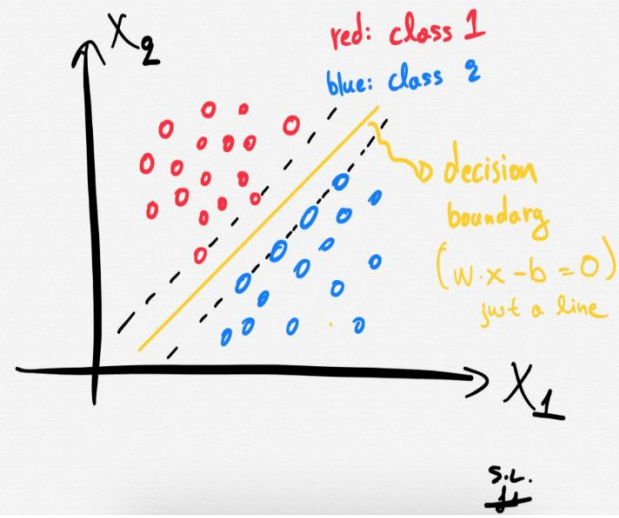


Figure – 3 Linear Support Vector Classifier(*Support Vector Machines (SVM) Clearly Explained: A Python Tutorial for Classification Problems... | Towards Data Science*, n.d.)

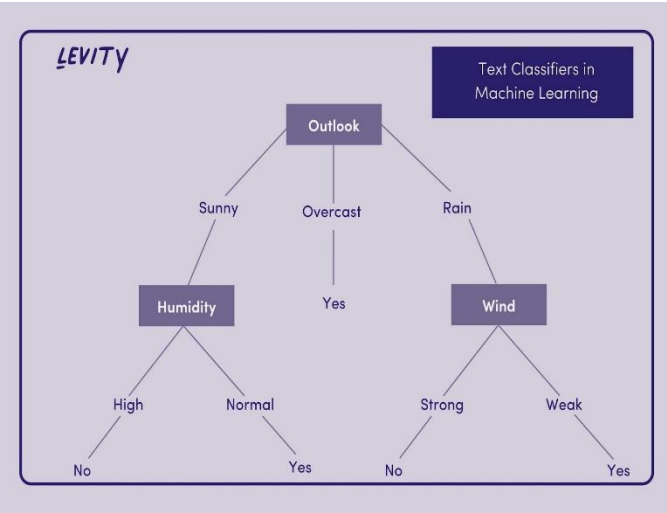


Figure – 4 Decision Tree(*Text Classifiers in Machine Learning: A Practical Guide*, n.d.)

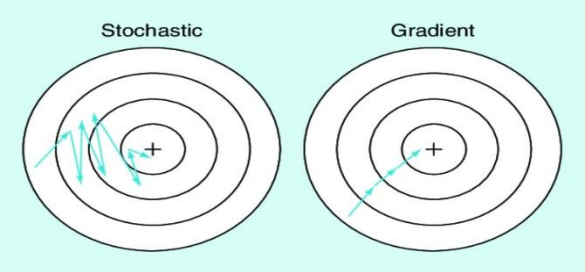


Figure 5 – Stochastic Gradient Descent(*Efficient Optimi: Mastering Stochastic Gradient Descent*, n.d.)

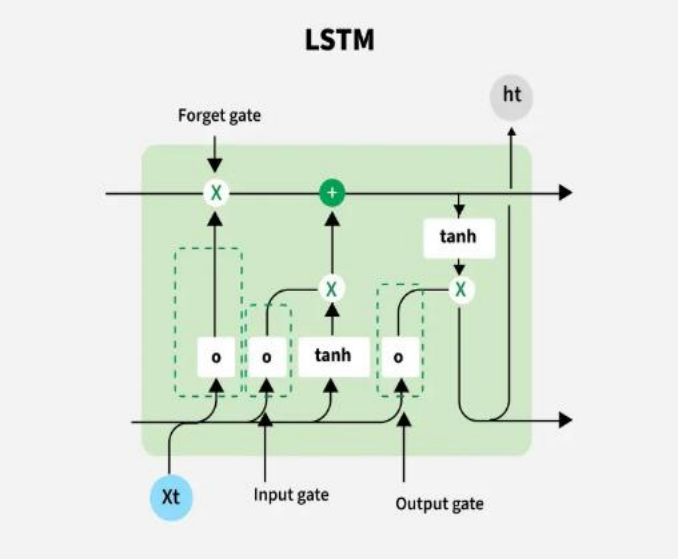


Figure – 6 LSTM Cell(*RNN vs LSTM vs GRU vs Transformers | GeeksforGeeks*, n.d.)

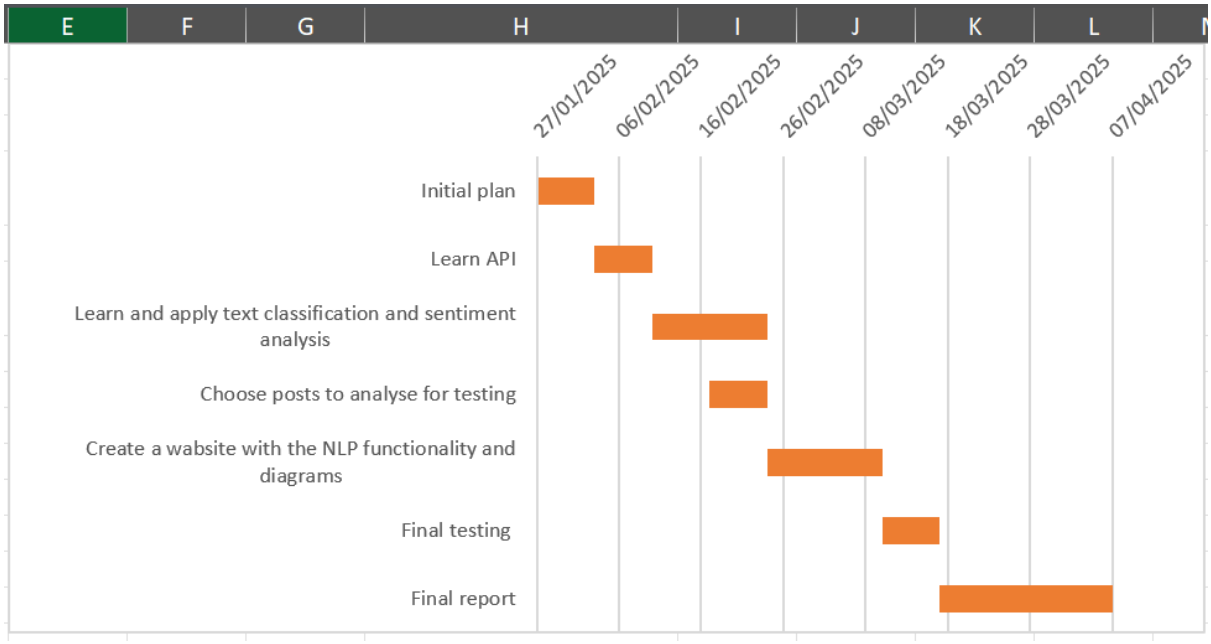


Figure – 7 Gantt chart created by excel.

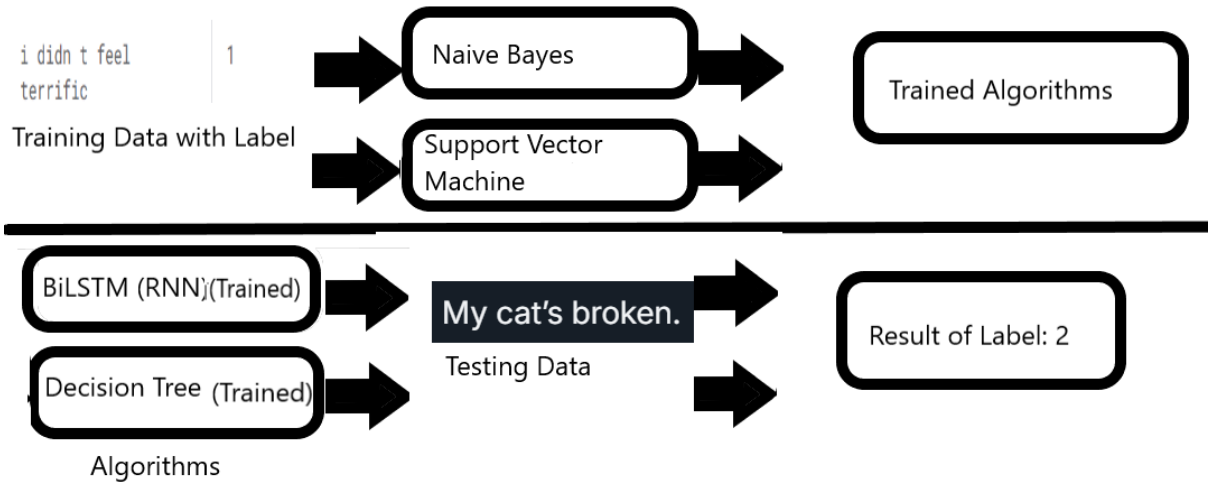


Figure – 8 Architecture of System

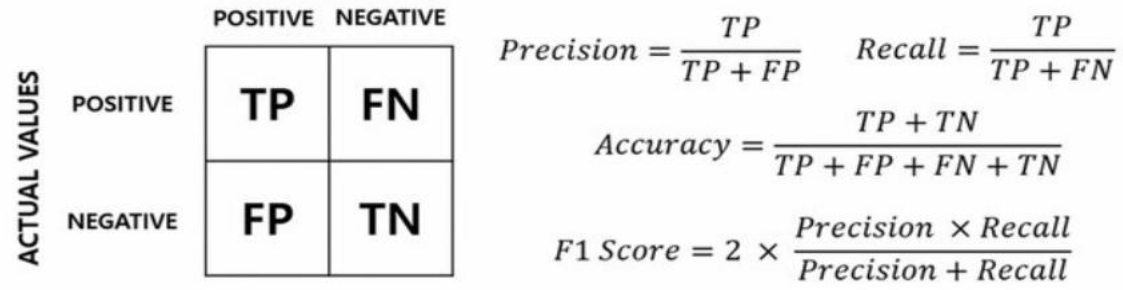


Figure – 9 Confusion matrix: Precision, Recall, Accuracy, and F1 score(Seol et al., 2023)

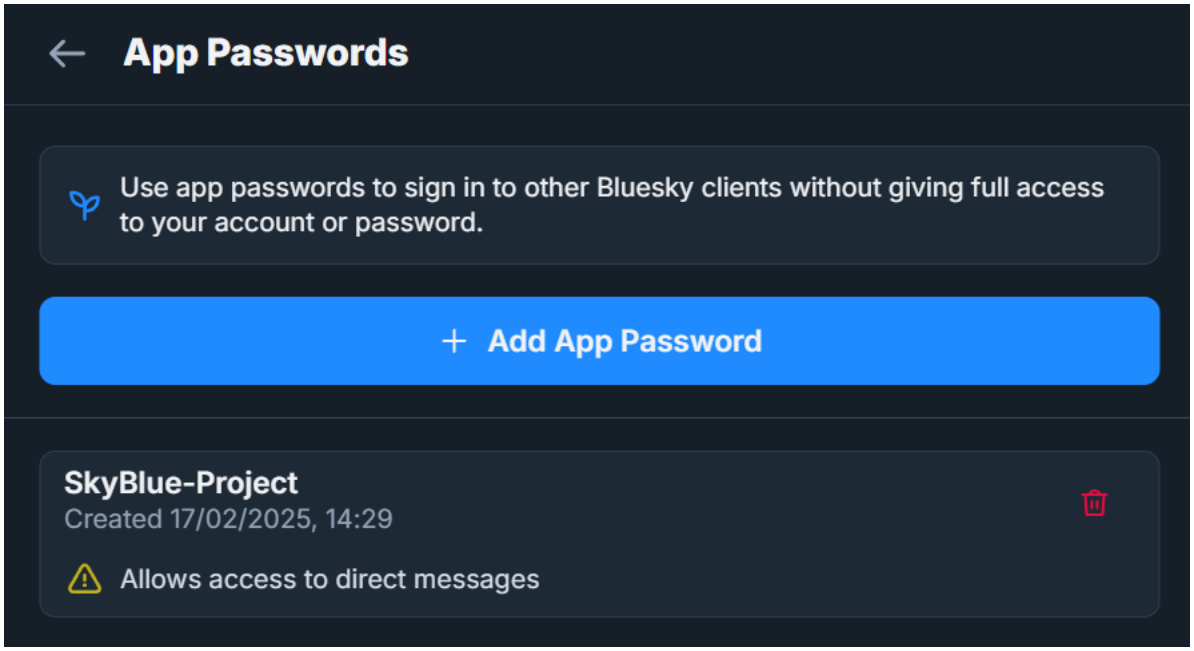


Figure – 10 Application Password of Bluesky

```
# Read the data and drop the first column
df = pd.read_csv('text.csv')

df = df.drop(columns=['Unnamed: 0'])

print(df)
```

Figure – 11 Reading the Dataset and Dropping the ID Field

```
# Split data into 3 parts train (80%), validate (10%) and test (10%)
train_df, temp_df = train_test_split(df, test_size=0.2, random_state=42, stratify=df['label'])

validate_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42, stratify=temp_df['label'])

print("Training set size:", train_df.shape)
print("Validation set size:", validate_df.shape)
print("Test set size:", test_df.shape)
```

Figure – 12 Splitting the Dataset

```
# Download the stopwords from nltk
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
stop_words.add('im')
print(stop_words)
```

Figure – 13 Stop words

```
# Remove the stopwords from each data frame
nltk.download('punkt')

for index, row in train_df.iterrows():
    word_tokens = word_tokenize(row["text"])
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
    filtered_sentence = " ".join(filtered_sentence)
    train_df.at[index, "text"] = filtered_sentence

for index, row in test_df.iterrows():
    word_tokens = word_tokenize(row["text"])
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
    filtered_sentence = " ".join(filtered_sentence)
    test_df.at[index, "text"] = filtered_sentence

for index, row in validate_df.iterrows():
    word_tokens = word_tokenize(row["text"])
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
    filtered_sentence = " ".join(filtered_sentence)
    validate_df.at[index, "text"] = filtered_sentence
```

Figure – 14 Removal of Stop Words

```
# Encoding the labels with one hot encoder
cat_encoder = OneHotEncoder()

array_label_train = train_df["label"].values
array_label_test = test_df["label"].values
array_label_validate = validate_df["label"].values

array_label_train = array_label_train.reshape(-1, 1)
array_label_test = array_label_test.reshape(-1, 1)
array_label_validate = array_label_validate.reshape(-1, 1)

label_cat_1hot_train = cat_encoder.fit_transform(array_label_train)
label_cat_1hot_test = cat_encoder.fit_transform(array_label_test)
label_cat_1hot_validate = cat_encoder.fit_transform(array_label_validate)

train_df["label"] = list(label_cat_1hot_train.toarray())
test_df["label"] = list(label_cat_1hot_test.toarray())
validate_df["label"] = list(label_cat_1hot_validate.toarray())
```

Figure – 15 One Hot Encoding

```
# Use TF-IDF approach
vectorizer = TfidfVectorizer(max_features=1000)

X_train = vectorizer.fit_transform(train_df['text'])
X_test = vectorizer.transform(test_df['text'])
X_val = vectorizer.transform(validate_df['text'])

# Convert labels into numeric as naive bayes does not use one hot encoded labels
y_train = np.argmax(np.array(train_df['label'].tolist()), axis=1)
y_test = np.argmax(np.array(test_df['label'].tolist()), axis=1)
y_val = np.argmax(np.array(validate_df['label'].tolist()), axis=1)

# Hyperparameter tuning using validation dataset
alpha_values = [0.1, 0.5, 1.0, 5.0]
best_alpha = None
best_val_acc = 0
best_model_nb = None

for alpha in alpha_values:
    classifier_nb = MultinomialNB(alpha=alpha)  # Alpha is the smoothing parameter that can be adjusted
    classifier_nb.fit(X_train, y_train)

    y_val_pred = classifier_nb.predict(X_val)  # Predictions
    val_acc = accuracy_score(y_val, y_val_pred)  # Validation accuracy calculated

    print(f"Alpha={alpha} | Validation Accuracy: {val_acc:.4f}")

    # Remove the bad performing model until the best performing model is reached
    if val_acc > best_val_acc:
        best_alpha = alpha
        best_val_acc = val_acc
        best_model_nb = classifier_nb

print(f"\nBest Alpha: {best_alpha} with Validation Accuracy: {best_val_acc:.4f}")

# Test using the best model on set
y_test_pred = best_model_nb.predict(X_test)

print("\nFinal Test Accuracy:", accuracy_score(y_test, y_test_pred))
print("\nFinal Test Classification Report - Naive Bayes:\n", classification_report(y_test, y_test_pred))

# Confusion matrix calculation for further analysis of model on test set
test_cm_nb = confusion_matrix(y_test, y_test_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(test_cm_nb, annot=True, fmt='d', cmap='Blues', xticklabels=range(len(set(y_test))), yticklabels=range(len(set(y_test))))
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix - Test Set - Naive Bayes")
plt.show()
```

Figure – 16 Machine Learning Example Code

```python
# Check if an available GPU exists
physical_devices = tf.config.list_physical_devices('GPU')
# Enable dynamic memory growth to save memory
tf.config.experimental.set_memory_growth(physical_devices[0], enable=True)


# Initalising training tokenizer structure and saving it as json for possible future use
vocab_size = 1000
max_length = 100
oov_token = "<OOV>"

tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_token)
tokenizer.fit_on_texts(train_df['text'].tolist())

X_train_seq = tokenizer.texts_to_sequences(train_df['text'].tolist())
X_val_seq   = tokenizer.texts_to_sequences(validate_df['text'].tolist())
X_test_seq  = tokenizer.texts_to_sequences(test_df['text'].tolist())

X_train = pad_sequences(X_train_seq, maxlen=max_length, padding='post', truncating='post')
X_val   = pad_sequences(X_val_seq, maxlen=max_length, padding='post', truncating='post')
X_test  = pad_sequences(X_test_seq, maxlen=max_length, padding='post', truncating='post')


tokenizer_json = tokenizer.to_json()
with open("tokenizer.json", "w") as json_file:
    json_file.write(tokenizer_json)


# Using the vocab size to convert one hot encoded labels into list structure
y_train = np.array(train_df['label'].tolist())
y_val   = np.array(validate_df['label'].tolist())
y_test  = np.array(test_df['label'].tolist())

num_classes = y_train.shape[1]
print("X_train: ", X_train[3])
print("y_train: ", y_train[3])
print("num_classes: ", num_classes)


# Designing of the RNN (Bidirectional LSTM) model
embedding_dim = 128

model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_length),
    SpatialDropout1D(0.2),
    Bidirectional(LSTM(128, return_sequences=True)),
    Dropout(0.3),
    Bidirectional(LSTM(128, return_sequences=True)),
    Dropout(0.3),
    Bidirectional(LSTM(64, return_sequences=False)),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(num_classes, activation='softmax')
])
```

Figure – 17 Deep Learning Example Code

```python
async def fetch_post_thread(link):
    """
    Input: A bluesky URL of a post
    Process: Extracting most popular 10 comments/replies via url_to_at_uri function
    Output: Most popular 10 comments/replies of the input URL post
    """
    uri = url_to_at_uri(link)
    print("Converted URI:", uri)
    client = AsyncClient("https://bsky.social")

    @client.on_session_change
    async def on_session_change(event: SessionEvent, session: Session):
        print("Session event:", event, session)

    # Log in using your API credentials
    await client.login("semih-cardiff.bsky.social", '[redacted]')
    res = await client.get_post_thread(uri=uri, depth=1)
    thread = res.thread

    # Extract text from up to 10 replies
    replies_text = []
    for reply in thread.replies[:10]:
        try:
            replies_text.append(reply.post.record.text)
        except AttributeError:
            replies_text.append("No text available")
    return replies_text
```

Figure – 18 Fetch Post Thread

```
# Download required NLTK data
nltk.download('stopwords')
nltk.download('punkt')

# Create a set of stopwords for cleaning
stop_words = set(stopwords.words('english'))

# Mapping from classifier number to emotion labels
classifier_num_to_emotion = {
    0: "sadness",
    1: "joy",
    2: "love",
    3: "anger",
    4: "fear",
    5: "suprise"
}

# Load the trained model
model = load_model('RNN_Tweet_Classifier.h5')

# Load the tokenizer from the saved JSON file
with open('tokenizer.json', 'r') as f:
    tokenizer_json = f.read()
    tokenizer = tokenizer_from_json(tokenizer_json)

# Set the max sequence length used during training
MAX_SEQUENCE_LENGTH = 100
```

Figure – 19 Initialization of Algorithms

```
def clean_text(text):
    """
    Input: Unprocessed text
    Process: Remove stopwords, Convert Emoji to useful text, Remove Hastags "#" but leave the useful text part
    Output: Cleaned text
    """
    text = emoji.demojize(text)
    text = re.sub(r'#(\S+)', r'\1', text)
    text = re.sub(r':([^:]+):', lambda m: " " + m.group(1).replace("_", " ") + " ", text)
    text = re.sub(r'\s+', ' ', text).strip()
    tokens = word_tokenize(text)
    filtered_tokens = [w for w in tokens if not w.lower() in stop_words]
    return " ".join(filtered_tokens)

def classify_text(text):
    """
    Input: Unclassified processed text
    Process: Labelling via trained RNN model
    Output: Label which is also the predicted emotion
    """
    # Clean the text exactly as done during training
    cleaned = clean_text(text)
    # Convert the cleaned text to sequences using the loaded tokenizer
    sequence = tokenizer.texts_to_sequences([cleaned])
    padded_sequence = pad_sequences(sequence, maxlen=MAX_SEQUENCE_LENGTH, padding='post', truncating='post')
    # Predict probabilities from the model
    prediction = model.predict(padded_sequence)
    class_index = np.argmax(prediction, axis=1)[0]
    predicted_emotion = classifier_num_to_emotion.get(class_index, "Unknown")
    return predicted_emotion

def url_to_at_uri(url):
    """
    Input: A bluesky URL of a post
    Process: Converting into URI
    Output: A blueksy URI of a post
    """
    parsed = urlparse(url)
    segments = parsed.path.strip('/').split('/')
    post_id = segments[-1]
    user_tag = segments[-3]
    return f"at://{user_tag}/app.bsky.feed.post/{post_id}"
```

Figure – 20 Main Functions

```
# Home page which is also the main page
@app.route("/")
@app.route("/home", methods=['GET', 'POST'])
def home():
```

Figure – 21 Routing

Figure – 22 Home Page

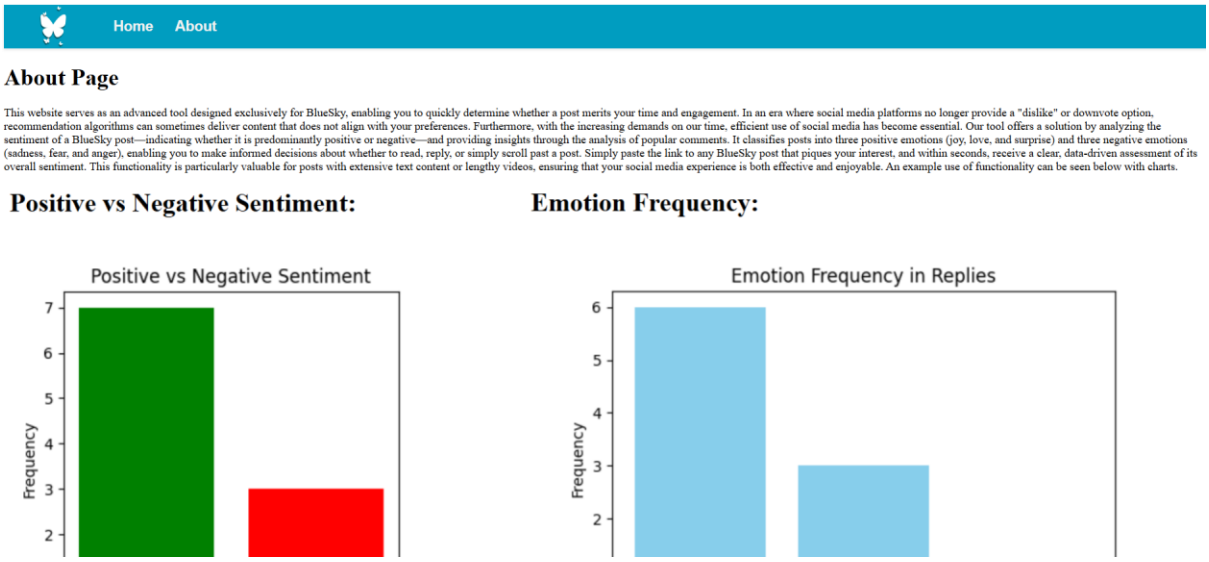


Figure – 23 About Page

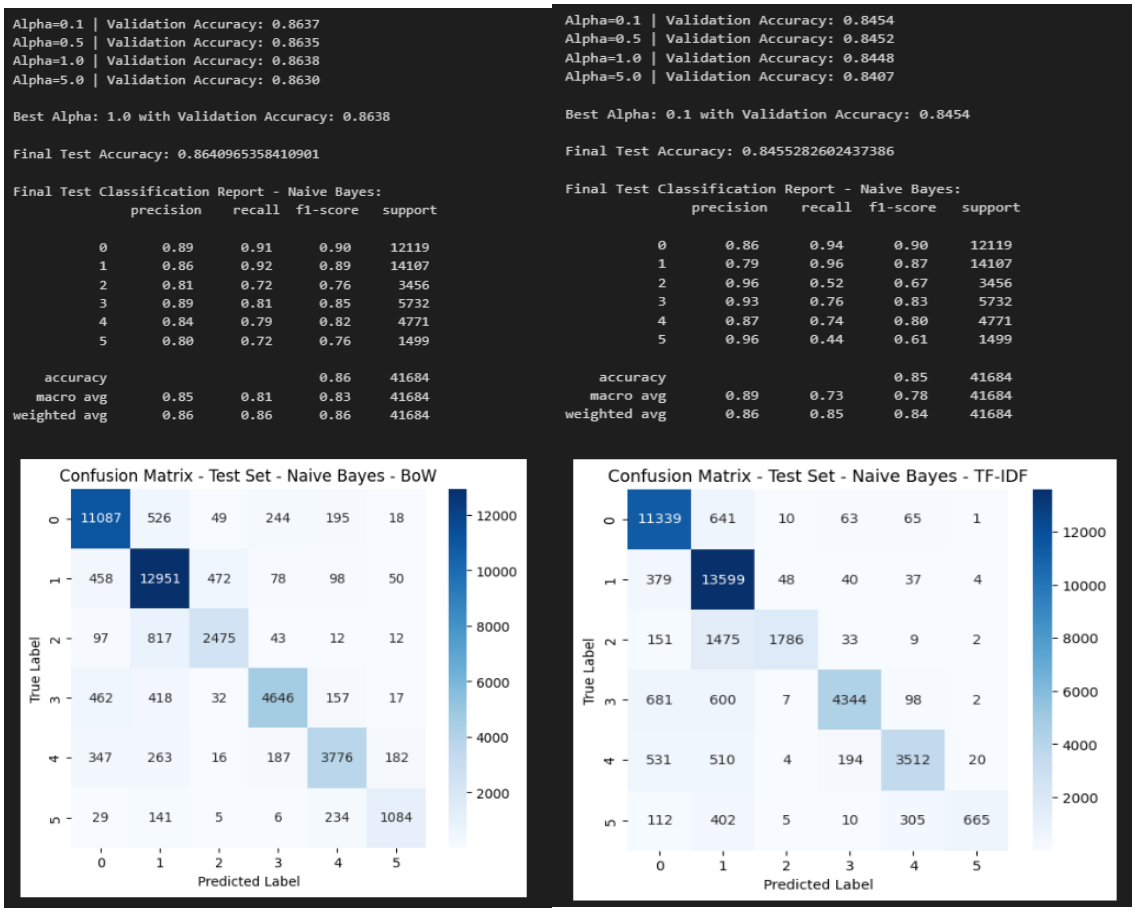
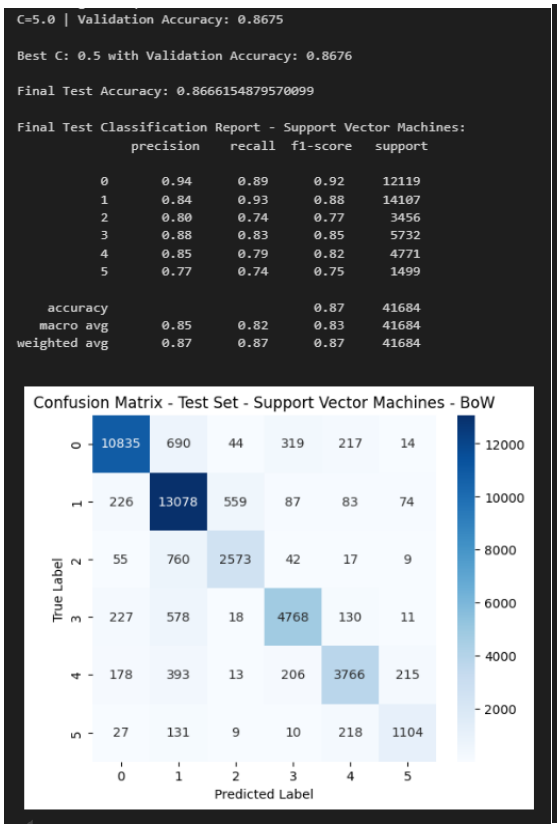


Figure – 24 and 25 Naïve Bayes



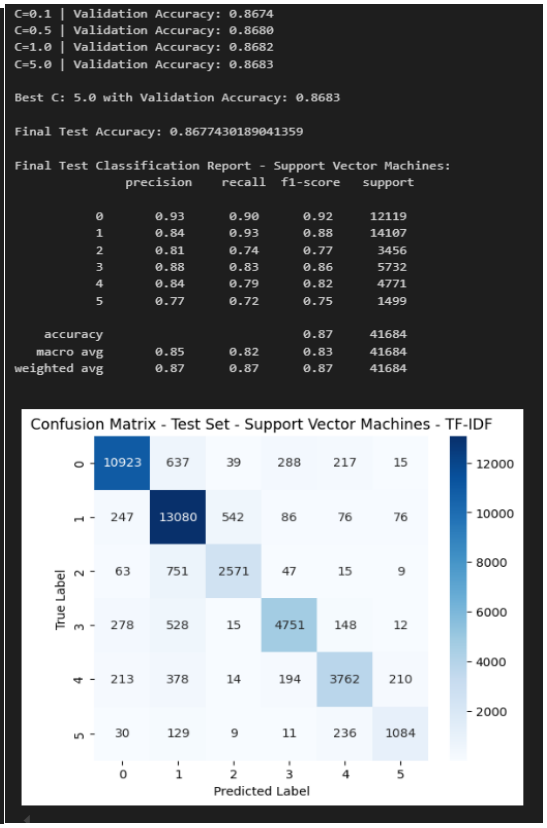
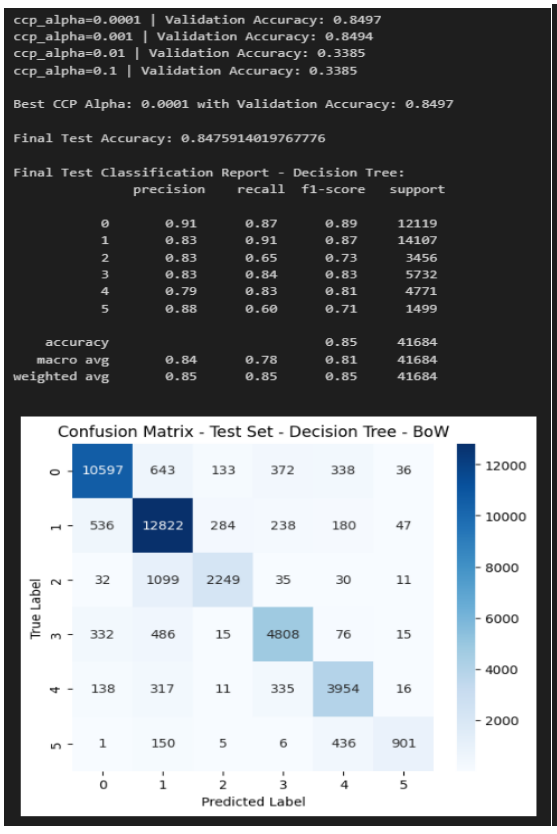


Figure – 26 and 27 Support Vector Machines



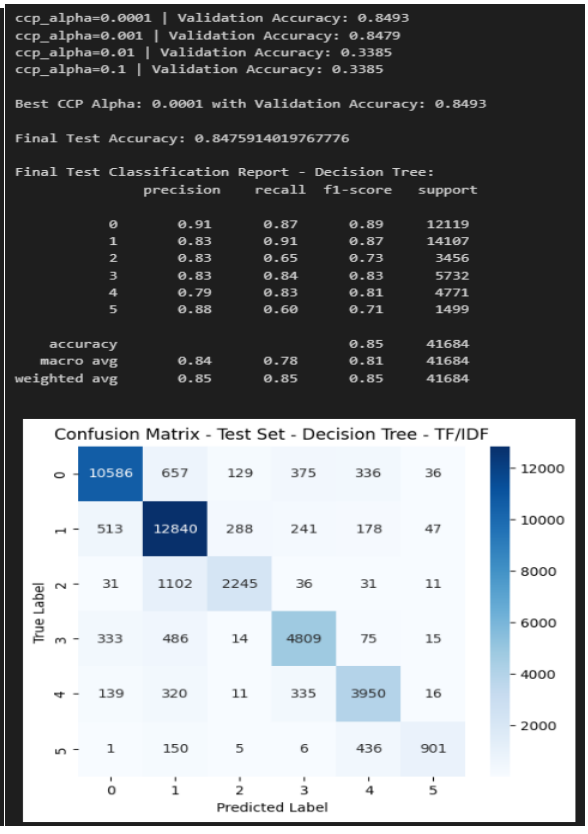
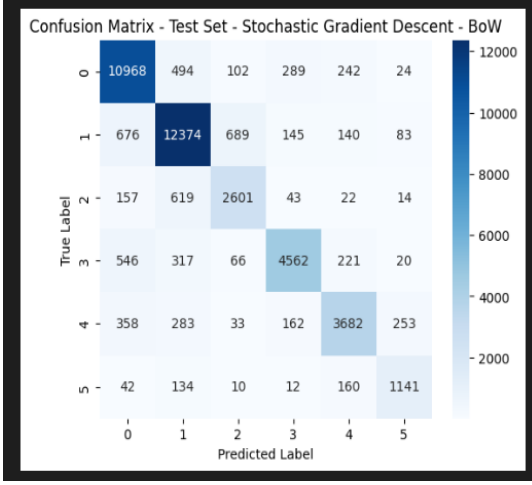


Figure – 28 and 29 Decision Tree

```
Alpha=1e-05 | Validation Accuracy: 0.8488
Alpha=0.0001 | Validation Accuracy: 0.8494
Alpha=0.001 | Validation Accuracy: 0.8477
Alpha=0.01 | Validation Accuracy: 0.8486

Best Alpha: 0.0001 with Validation Accuracy: 0.8494

Final Test Accuracy: 0.8475194319163228

Final Test Classification Report - Stochastic Gradient Descent:
              precision    recall  f1-score   support

           0       0.86      0.91      0.88     12119
           1       0.87      0.88      0.87     14107
           2       0.74      0.75      0.75      3456
           3       0.88      0.80      0.83      5732
           4       0.82      0.77      0.80      4771
           5       0.74      0.76      0.75      1499

    accuracy                           0.85     41684
   macro avg       0.82      0.81      0.81     41684
weighted avg       0.85      0.85      0.85     41684
```




```
Alpha=1e-05 | Validation Accuracy: 0.8478
Alpha=0.0001 | Validation Accuracy: 0.8464
Alpha=0.001 | Validation Accuracy: 0.8483
Alpha=0.01 | Validation Accuracy: 0.8469

Best Alpha: 0.001 with Validation Accuracy: 0.8483

Final Test Accuracy: 0.8494386335284522

Final Test Classification Report - Stochastic Gradient Descent:
              precision    recall  f1-score   support

           0       0.88      0.91      0.89     12119
           1       0.84      0.91      0.88     14107
           2       0.80      0.68      0.74      3456
           3       0.88      0.78      0.82      5732
           4       0.82      0.78      0.80      4771
           5       0.76      0.70      0.73      1499

    accuracy                           0.85     41684
   macro avg       0.83      0.79      0.81     41684
weighted avg       0.85      0.85      0.85     41684
```

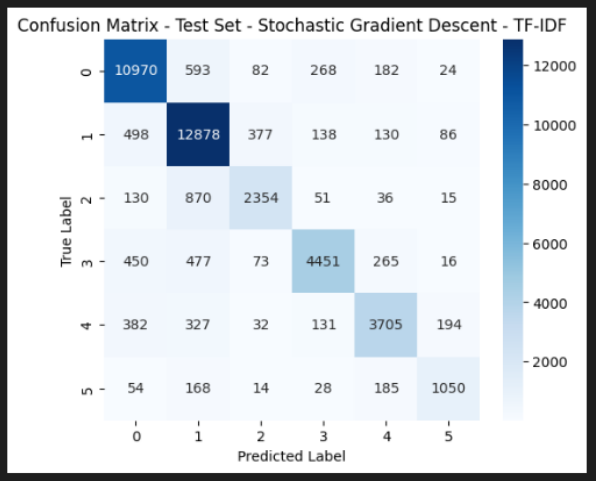
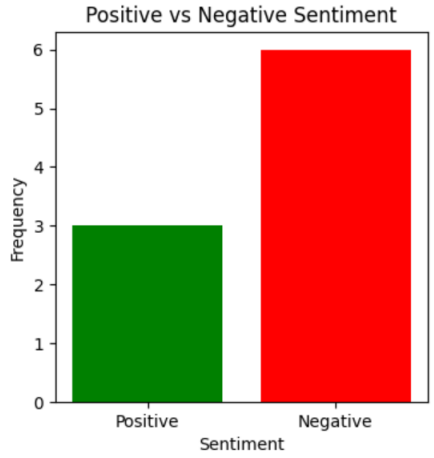


Figure – 30 and 31 Stochastic Gradient Descent

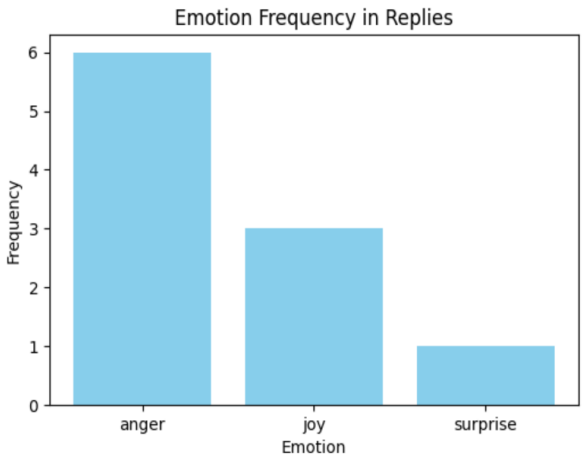
```
Epoch 1/5
5211/5211 [==============================] - 408s 76ms/step - loss: 0.3383 - accuracy: 0.8609 - val_loss: 0.2179 - val_accuracy: 0.8951
Epoch 2/5
5211/5211 [==============================] - 436s 84ms/step - loss: 0.2239 - accuracy: 0.8953 - val_loss: 0.2103 - val_accuracy: 0.8965
Epoch 3/5
5211/5211 [==============================] - 464s 89ms/step - loss: 0.2157 - accuracy: 0.8977 - val_loss: 0.2091 - val_accuracy: 0.8970
Epoch 4/5
5211/5211 [==============================] - 495s 95ms/step - loss: 0.2111 - accuracy: 0.8986 - val_loss: 0.2081 - val_accuracy: 0.8990
Epoch 5/5
5211/5211 [==============================] - 469s 90ms/step - loss: 0.2074 - accuracy: 0.8994 - val_loss: 0.2078 - val_accuracy: 0.8985
```

Figure – 32 BiLSTM

Positive vs Negative Sentiment:



Emotion Frequency:

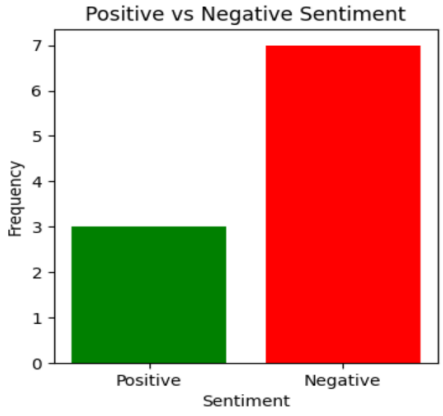


Replies and Emotions:

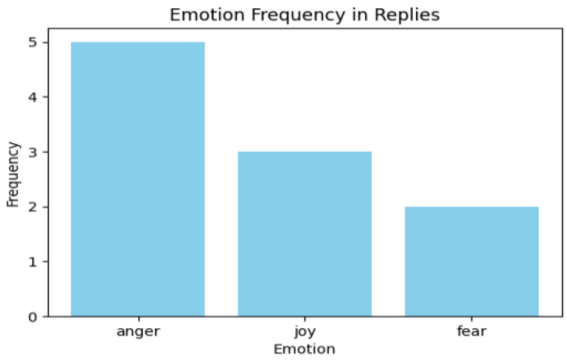
- It's cat yoga. Mine just perfected his Hummingbird Stretch. - **Classified as: anger**
- My cat looks like your cat - **Classified as: joy**
- So adorable - **Classified as: anger**
- It's cat-ching on - **Classified as: anger**
- Your cat is very good looking! - **Classified as: joy**
- Are those air biscuits?? 😆 - **Classified as: anger**
- 😬 - **Classified as: anger**
- I think that's what happened to my CatHole too!! - **Classified as: anger**
- 🐈‍⬛🐈😆 - **Classified as: joy**
- That's funny! Your cat looks like one of those dog toys with no stuffing, or a wadded towel, depending on the light. - **Classified as: surprise**

Figure – 33, 34 and 35 Post Example 1

Positive vs Negative Sentiment:



Emotion Frequency:

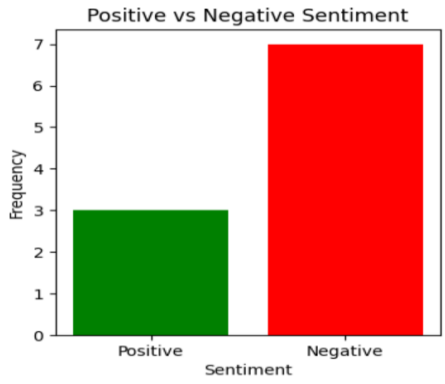


Replies and Emotions:

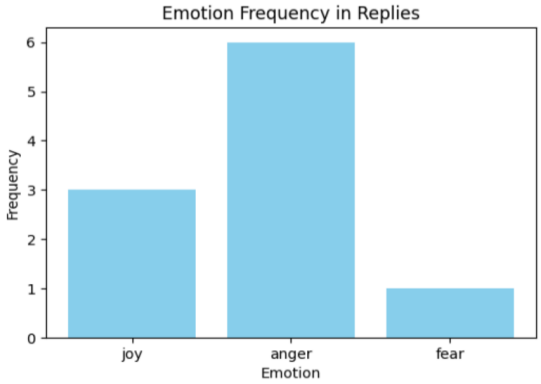
- Hurry home kids, it's almost time! - **Classified as: anger**
- ".. This commercial was written and directed by a sentient bag of cocaine" - **Classified as: anger**
- 🌞 They've made the pill look like the Teletubbies Sun-Baby. 🌞 - **Classified as: joy**
- The perfect reason why drugs should not be advertised. - **Classified as: joy**
- I'll chime in with the others and ask... "Why don't it tell us what this Claritin does?!" Because if I take it and I don't have an ego death mind trip... I would feel so confused. - **Classified as: fear**
- what's this shit even for??????? - **Classified as: anger**
- flight or flight. only flying is an option - **Classified as: anger**
- I love how the last trippy shot is a door floating in clouds opens to a bright white light - **Classified as: joy**
- Claritin - Now with 50% more LSD™ - **Classified as: anger**
- This is such a weird commercial, and I..... MUST BUY CLARITIN. CLARITIN IS PEACE. CLARITIN IS LOVE. MORE CLARITIN TO PURCHASE NOW PLEASE. - **Classified as: fear**

Figure – 36, 37 and 38 Post Example 2

Positive vs Negative Sentiment:



Emotion Frequency:

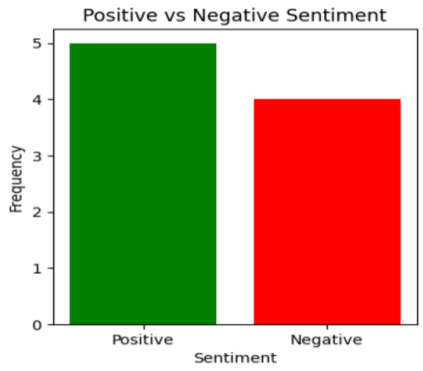


Replies and Emotions:

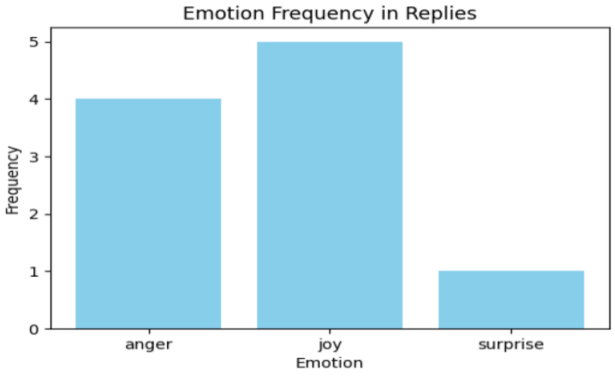
- Beautiful I think my blood pressure just went down 😊 - **Classified as: joy**
- it reminds me of someone ... can't remember who ... - **Classified as: anger**
- I can't stop watching this! So calming 😀 - **Classified as: anger**
- Hello, I must be going! - **Classified as: fear**
- Slidin in to a dm. - **Classified as: anger**
- HEY GURL - **Classified as: anger**
- Me on roller skates… - **Classified as: anger**
- Growing up in South Florida we called those cow-fish - **Classified as: anger**
- Love it! 😍 - **Classified as: joy**
- 😂 That's not a fish--it's a skkrrttt-ing iron with fins. - **Classified as: joy**

Figure – 39, 40 and 41 Post Example 3

Positive vs Negative Sentiment:



Emotion Frequency:

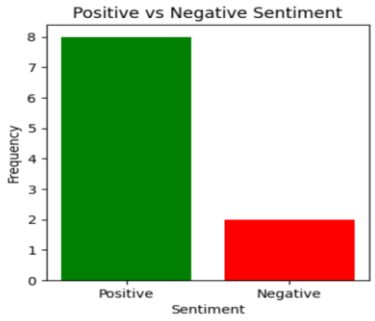


Replies and Emotions:

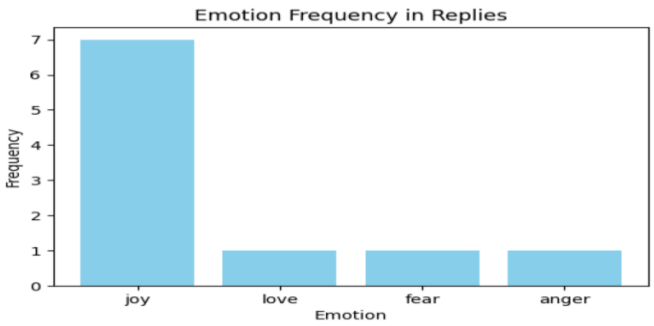
- DUSTER CAT - **Classified as: anger**
- I love it. What a beautiful painting. Hope it's on display now. - **Classified as: joy**
- Free tinkle! (Love her double bells) - **Classified as: joy**
- Make a print, frame it, and simply stick it on the wall when a museum guard isn't looking. There's temporary double side tape that are easily removable without marring the wall. youtu.be/m-OyRZoNb6o?... - **Classified as: anger**
- Sadly, Tinkle is not on view this year. The last time it was on display was in my "Creature Comforts" exhibition in 2020. Here's a photo. - **Classified as: joy**
- My mother-in-law painted her own version of Tinkle in the 70s. - **Classified as: anger**
- I like Tinkle she is beautiful- some of the other paintings kinda scare me😂 - **Classified as: joy**
- How intriguing! Fingers crossed this is the year "Tinkle" makes an appearance. A hidden gem from 1883? I'd love to see it in person! - **Classified as: joy**
- Oh, I went to that museum last year, it was amazing! Really loved the steam ship. - **Classified as: surprise**
- assuming Tinkle is named after what appear to be bells on their collar. - **Classified as: anger**

Figure – 42, 43 and 44 Post Example 4

Positive vs Negative Sentiment:



Emotion Frequency:



Replies and Emotions:

- Lovely! Look at its little tongue right at the end there - mlem mlem! - **Classified as: joy**
- Oh my, gorgeous! - **Classified as: joy**
- Loving this chocolate brown, fuchsia combo 🌺 - **Classified as: love**
- I have to follow for more of this content 🙀 - **Classified as: joy**
- I miss my garden in our previous home. So many bougainvillea and hibiscus and desert roses and gardenia and royal poinciana. We had the intense pink ones both in the front and back yards....ah, miss that. - **Classified as: fear**
- I was not prepared for something that beautiful. - **Classified as: joy**
- Gorgeous - **Classified as: joy**
- You just made my day - **Classified as: joy**
- www.hummingbirdcentral.com/hummingbird-... I will be on the lookout for Ruby-Throated Hummingbird. - **Classified as: anger**
- If you got bougainvillea advice I'll take it - **Classified as: joy**

Figure – 45, 46 and 47 Post Example 5