Personality and Emotion–A Comprehensive Analysis Using Contextual Text Embeddings (SBERT)

Project & Thesis-II

CSE 4250

A thesis Report

Submitted in partial fulfillment of the requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

Submitted by

Risha Asfara	**180204001**
Rasel Ahmed	**190104093**
Tahira Ferdousi	**190104123**
Md. Ali Akber	**190104129**

Supervised by

Ms.Raqeebir Rab

Department of Computer Science and Engineering

Ahsanullah University of Science and Technology

Dhaka, Bangladesh

November 21,2023

CANDIDATES' DECLARATION

We, hereby, declare that the thesis presented in this report is the outcome of the investigation performed by us under the supervision of Ms.Raqeebir Rab, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final-year courses, CSE4100: Project and Thesis I and CSE4250: Project and Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma, or other qualifications.

——————————-

Risha Asfara
180204001

——————————-

Rasel Ahmed
190104093

——————————-

Tahira Ferdousi
190104123

——————————-

Md. Ali Akber
190104129

CERTIFICATION

This thesis titled, "**Personality and Emotion—A Comprehensive Analysis Using Contextual Text Embeddings (SBERT)**" serves as a partial requirement for the fulfillment of a B.Sc. degree in Computer Science and Engineering in November 2023.

Group Members:

Risha Asfara	**180204001**
Rasel Ahmed	**190104093**
Tahira Ferdousi	**190104123**
Md. Ali Akber	**190104129**

Ms.Raqeebir Rab
Assistant Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Dr. Md. Shahriar Mahbub
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

This paper and the research behind it would not have been possible without the exceptional support of our supervisor Ms.Raqeebir Rab, Assistant Professor, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology. We would like to thank Professor Dr. Md. Shahriar Mahbub, Head of Ahsanullah University of Science and Technology's Department of Computer Science and Engineering.
We also thank the university students, research assistants, and librarians who helped and encouraged us.

In conclusion, we express our gratitude to our parents for their unwavering encouragement and support during our academic journey.

Risha Asfara
Rasel Ahmed
Tahira Ferdousi
Md. Ali Akber

ABSTRACT

Personality refers to a person's core behavioral, thinking, and feeling tendencies, which are shaped by a delicate interplay of nature and nurture. Emotions, on the other hand, are patterns of emotional, behavioral, and physiological responses to specific situations. This research looks at the complex interplay that exists between these two key aspects of the human experience. The Myers-Briggs type indicator (MBTI) has become one of the most valuable instruments available for predicting personality due to its widespread application in a variety of fields. The objective is to use sentence transformers to discern the context of user-written social media posts and automate the process. In this research, a range of text pre-processing methods and data balancing techniques, such as random oversampling, are employed. The context of the text posts is determined using Sentence-BERT (SBERT), a pre-trained model created especially for sentence-level embeddings. Using the Myers-Briggs Type Indicator (MBTI) and a variety of machine learning techniques, such as Support Vector Machines (SVM), Logistic Regression (LR), K-Nearest Neighbors (KNN), and Random Forest (RF) Classifiers, it is possible to predict people's personalities based on text. SBERT, combined with the Random Forest Classifier, performs outstandingly in predicting the MBTI personality. We also created a robust mechanism for computing similarity scores with predefined three types of emotion (joy, sadness, and anger) using preprocessed MBTI-sentence-level contextual embeddings. Using a form of transfer learning technique, we trained a model (SVM) using datasets tagged with emotions to predict the emotions of various MBTI social media posts. A vetting method was created that incorporates similarity scores and model predictions and selects labels that correspond with both predictions. The study continued to quantify the emotional distribution within each type of personality. Our findings shed light on the complex interaction between the personality types of MBTI and emotions, allowing us to gain a better understanding of how people with varied personality features express and experience emotions in online settings.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Social media platforms have become a veritable gold mine of user-generated content, providing insight into people's thoughts, emotions, and personality traits. A widely used method to classify people into different personality types is the Myers-Briggs Type Indicator (MBTI). Combining these two elements, the development of an MBTI personality trait text classification machine learning model specifically trained on social media posts holds immense potential. Popular models, such as Ekman's six-emotion model, are also utilized to extract emotions from these text posts. Finally, a correlation between a person's personality and emotions can be formed through social media posts.

1.1 Motivation

A person's personality can be defined as the role they perform or the persona they present to the outside world. Personality assessments can help someone grow both personally and professionally. The applications of a MBTI personality trait text classification machine learning model to social media posts are extensive. Developing a machine learning model for the classification of MBTI personality traits in social media postings was motivated by a number of factors. To start with, social media has emerged as a prominent medium for communication and self-expression, allowing users to freely exchange their thoughts, experiences, and opinions. By utilizing this extensive volume of textual data generated by users, we can acquire profound understandings of the personality traits of individuals on a large scale. Additionally, it may facilitate the detection of possible associations between particular personality types and mental health disorders or modes of communication.
Emotional research is also crucial in human psychology and behavior, impacting personal well-being, interpersonal relationships, and mental health. Textual analysis offers valuable insights into emotions, allowing researchers to understand the underlying origins, triggers,

and consequences of emotions. This knowledge can improve emotional intelligence, empathy, and communication abilities on a personal and societal level. Emotions can also reveal relationships between emotional states and mental health issues, enabling early identification and intervention in mental health illnesses. Researchers can build methods by finding linguistic signals associated with distinct emotional states, thereby enhancing individuals' overall well-being. Thus, textual analysis is a valuable tool for understanding and addressing emotional states in individuals.
In summary, uncovering the correlation between personality traits and emotions expressed in social media posts holds immense potential for advancing psychological research, mental health support, and human-computer interaction. By exploring these connections, we can develop more nuanced and personalized approaches to emotional well-being, fostering a deeper understanding of human emotions in the digital age.

1.2 Problem Statement

In a world where communication is increasingly centered on social media, we want to see if there is a strong link between a person's personality and emotion based on text posts. People share their thoughts and emotions on social media every day. The context of the posts differs based on the author's personality and the emotions they are attempting to communicate. The context of a person's posts may additionally indicate his or her personality and emotions. Our goal is to apply state-of-the-art models, such as SBERT, to capture the context of individual posts and predict their personalities and emotions in order to uncover correlations between them.

1.3 Proposed Methodology

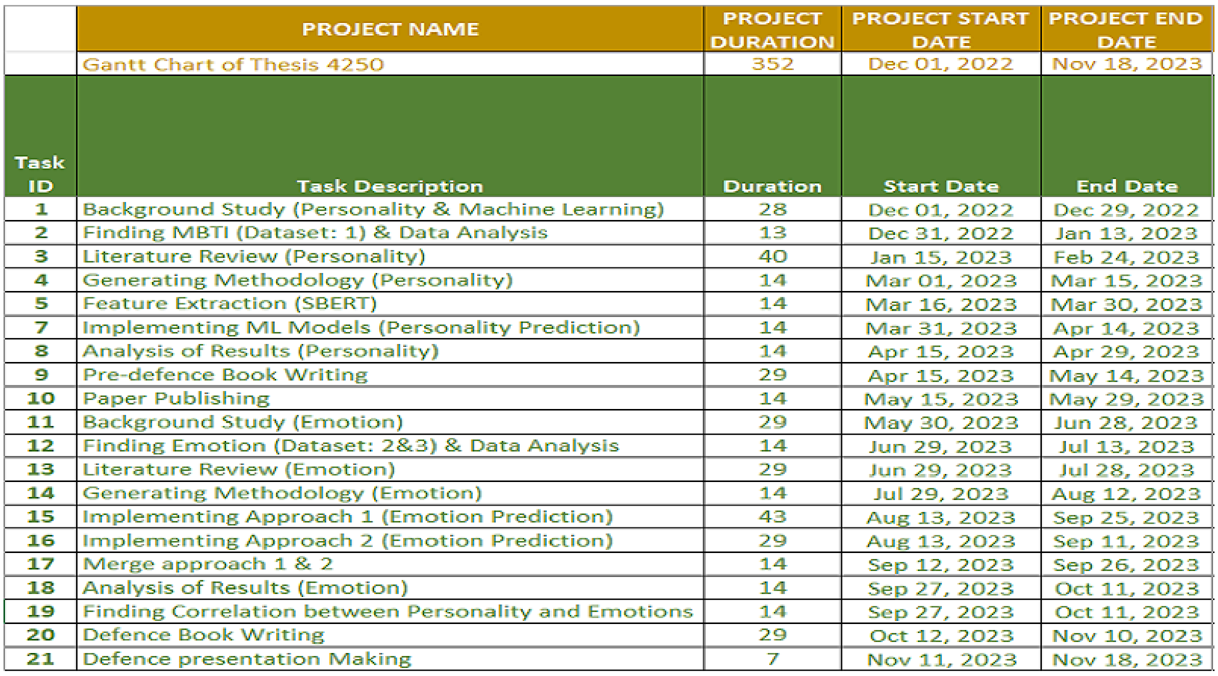
To analyze a person's personality and emotions, our first step would be to gather social media posts from individuals. We collected the MBTI dataset, which is freely accessible on Kaggle. Text data is unsuitable for machine learning models. Textual data must be transformed into numerical vectors. Prior to that, the data needs to be cleaned up, as it can contain URLs, images and videos that are not very useful for determining someone's personality and emotions. To understand the text posts' context for each person's post, a state-of-the-art model like SBERT is used, which provides contextualized sentence embeddings. Then machine learning models, such as SVM, LR, RF, and DT are used to predict MBTI personality traits. We got RF as our best model over the other three models with an average of 90.86% accuracy (I/E-96.04%, S/N-99.64%, T/F-83.62%, J/P-84.16%). Again, for predicting emotions for the same posts we evaluated for personality, we utilized two distinct techniques and then integrated them both to predict emotions. First, we compute cosine similarity scores between the embedded vectors of the posts and predefined emotion vectors. The vector with the highest similarity score among all these emotions is assigned as the emotion. Second, we trained various machine learning models (SVM, RF, LR and KNN) using a dataset of emotion-labeled texts. These models were then used to predict the emotions of the MBTI dataset. In this case, SVM outperformed all other machine learning models with an accuracy of 87.05%. We synchronized the two separate methods for predicting emotions in order to increase the prediction's reliability, and to do this, a vetting mechanism is used. We have also observed that these two separate techniques were able to predict around 88% to 90% of the data with the same emotion. Finally, we aimed to determine the relationship between personality and emotions using social media posts.

1.4 Gantt Chart

A Gantt chart is a visual tool for project management that shows how tasks or activities are scheduled and progress over time. It shows an order of the tasks involved in a project, including start and finish dates, task dependencies, and the present state of each work. In fig. 1.1 a detailed Gantt chart Data Table shows our project's schedule and work allocation. The several stages of the project, tasks, and corresponding start and completion dates are all graphically mapped out. Task dependencies are clearly shown in fig. 1.2, which also highlights important milestones and essential pathways.

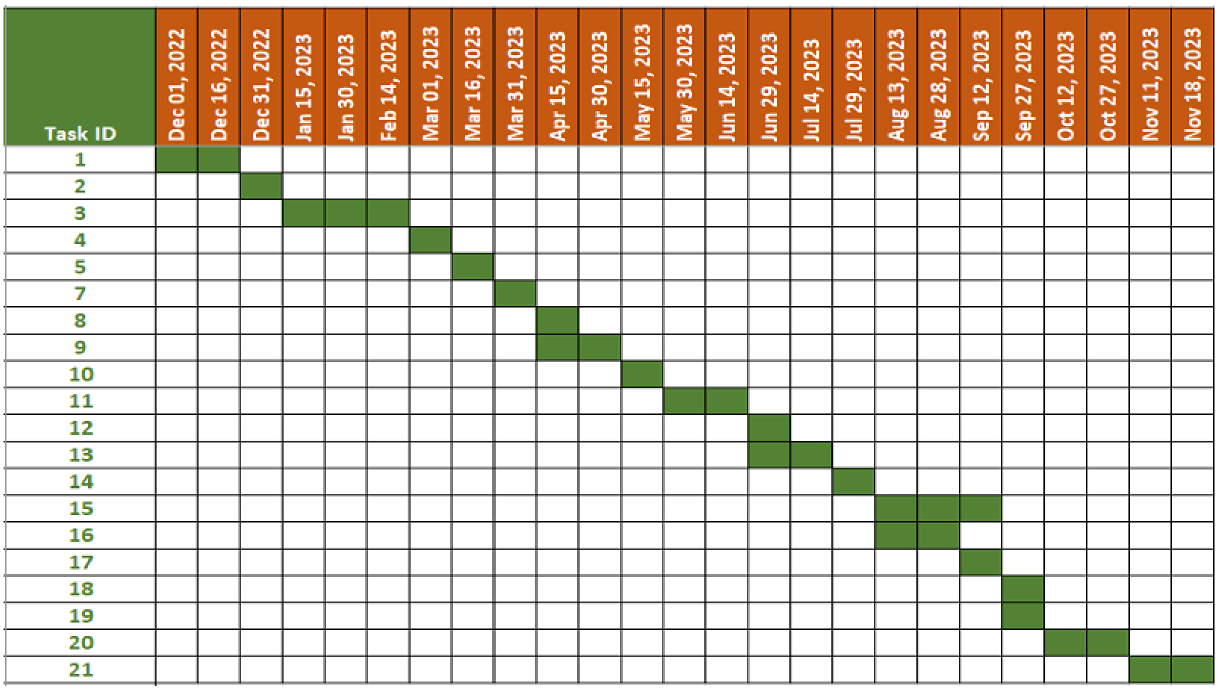
1.5 Organization of Thesis

The structure of our thesis is as follows: Chapter 1 provides a concise overview of our motivation and objectives for undertaking this research. Chapters 2 and 3 provide an examination of the theory of applied knowledge and present an overview of relevant research studies, respectively. Chapters 4 and 5 provide a detailed description of the proposed methodology employed in our study for predicting personality and emotion using the MBTI dataset. In Chapter 6, an evaluation is conducted on the performances of both approaches. In chapter five, we discuss our future plans and conclude.



	PROJECT NAME	PROJECT DURATION	PROJECT START DATE	PROJECT END DATE
	Gantt Chart of Thesis 4250	352	Dec 01, 2022	Nov 18, 2023
Task ID	Task Description	Duration	Start Date	End Date
1	Background Study (Personality & Machine Learning)	28	Dec 01, 2022	Dec 29, 2022
2	Finding MBTI (Dataset: 1) & Data Analysis	13	Dec 31, 2022	Jan 13, 2023
3	Literature Review (Personality)	40	Jan 15, 2023	Feb 24, 2023
4	Generating Methodology (Personality)	14	Mar 01, 2023	Mar 15, 2023
5	Feature Extraction (SBERT)	14	Mar 16, 2023	Mar 30, 2023
7	Implementing ML Models (Personality Prediction)	14	Mar 31, 2023	Apr 14, 2023
8	Analysis of Results (Personality)	14	Apr 15, 2023	Apr 29, 2023
9	Pre-defence Book Writing	29	Apr 15, 2023	May 14, 2023
10	Paper Publishing	14	May 15, 2023	May 29, 2023
11	Background Study (Emotion)	29	May 30, 2023	Jun 28, 2023
12	Finding Emotion (Dataset: 2&3) & Data Analysis	14	Jun 29, 2023	Jul 13, 2023
13	Literature Review (Emotion)	29	Jun 29, 2023	Jul 28, 2023
14	Generating Methodology (Emotion)	14	Jul 29, 2023	Aug 12, 2023
15	Implementing Approach 1 (Emotion Prediction)	43	Aug 13, 2023	Sep 25, 2023
16	Implementing Approach 2 (Emotion Prediction)	29	Aug 13, 2023	Sep 11, 2023
17	Merge approach 1 & 2	14	Sep 12, 2023	Sep 26, 2023
18	Analysis of Results (Emotion)	14	Sep 27, 2023	Oct 11, 2023
19	Finding Correlation between Personality and Emotions	14	Sep 27, 2023	Oct 11, 2023
20	Defence Book Writing	29	Oct 12, 2023	Nov 10, 2023
21	Defence presentation Making	7	Nov 11, 2023	Nov 18, 2023

Figure 1.1: Gantt Chart Data Table



Task ID	Dec 01, 2022	Dec 16, 2022	Dec 31, 2022	Jan 15, 2023	Jan 30, 2023	Feb 14, 2023	Mar 01, 2023	Mar 16, 2023	Mar 31, 2023	Apr 15, 2023	Apr 30, 2023	May 15, 2023	May 30, 2023	Jun 14, 2023	Jun 29, 2023	Jul 14, 2023	Jul 29, 2023	Aug 13, 2023	Aug 28, 2023	Sep 12, 2023	Sep 27, 2023	Oct 12, 2023	Oct 27, 2023	Nov 11, 2023	Nov 18, 2023
1	🟩	🟩																							
2			🟩																						
3				🟩	🟩	🟩																			
4							🟩																		
5								🟩																	
7									🟩																
8										🟩															
9										🟩	🟩														
10												🟩													
11													🟩	🟩											
12															🟩										
13															🟩	🟩									
14																	🟩								
15																		🟩	🟩	🟩					
16																		🟩	🟩						
17																				🟩					
18																					🟩				
19																					🟩				
20																						🟩	🟩		
21																								🟩	🟩

Figure 1.2: Gantt Chart

Chapter 2

Background Study

2.1 MBTI (Myers-Briggs Type Indicator)

Today, a variety of personality models, including the ”Big Five Personality Traits” model [1] and the ”Myers-Briggs Type Indicator (MBTI)” model [2], are used to describe personalities. Since its initial release in 1962, it has been found that ”MBTI” is more effective because it can be used in a wider range of disciplines. So, we selected the ”MBTI” personality model for this study due to its widespread use and the potential for it in a number of industries. The MBTI's fundamental principle is that each person's personality can be classified into only one of its 16 types. These classifications are based on four personality traits, each of which consists of two opposing preferences. The four dimensions are [3] :

1. Introversion (I) vs Extrovert (E): This component indicates the person's perceptual orientation. Extroverts are reputed to respond to direct and concrete environmental conditions. However, introverts focus inward on their internal and personal responses to their surroundings.

2. Intuition (N) vs Sensing (S): People who prefer sensing rely on that which can be perceived and are thought to be inclined toward reality. People who prefer intuition tend to rely more on their unconscious and nonobjective perceptual processes.

3. Feeling (F) vs Thinking (T): A prediction for thinking denotes the application of reason and logic to draw conclusions and make decisions. A preference for making judgments based on subjective considerations, including emotional responses to events, is represented by feeling.

4. Perception (P) vs Judgment (J): Briggs and Myers developed the judgment-perception 2 preference to show whether rational or illogical judgments predominate when a person interacts with the environment. The perceptive person uses the perceiving

and intuition processes, whereas the judging person uses a combination of thoughts and feelings when making decisions.

2.2 What is Machine Learning?

Machine learning is a way to teach computers how to learn and make decisions on their own by analyzing data, without being explicitly programmed for every specific task. Machine learning algorithms are designed to generalize from data, which means they can apply their learning to new, unseen data and make predictions or decisions based on that data. There are several types of machine learning algorithms, including:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

In supervised learning, the algorithm is trained on labeled data, where the input data is paired with corresponding output labels or targets. The algorithm learns to map the input data to the correct output based on the provided labels. This type of learning is commonly used for tasks such as classification.

2.3 What is Text Classification?

Text classification is an analytical procedure that takes any text content as input and assigns a label (or classification) from a specified set of class labels. Assigning the best class label automatically to a given text document based on its content is the aim of text categorization. The following phases are often included in the text classification process:

1. Data Preparation
2. Feature Extraction
3. Training Data Preparation
4. Model Selection and Training
5. Model Evaluation
6. Model Deployment

2.4 Classical Algorithms for Text Classification

- **Logistic Regression**: Based on the idea of probability, it is a predictive analysis method. When the dependent variable (target) is categorical, logistic regression is utilized [4].

- **Support Vector Machine:** Support Vector Machine, sometimes known as SVM, is a linear model used to solve classification and regression issues. The SVM concept is simple: A line or a hyperplane that divides the data into classes is produced by the algorithm. The SVM method identifies the points from both classes that are closest to the line. These points are called support vectors. The distance between the line and the support vectors are now calculated. The margin refers to this separation. Maximizing the margin is the goal. The best hyperplane is the one for which the gap is greatest. SVM aims to create decision boundaries that allow for the greatest feasible separation between the two groups [5].

- **Decision Tree:** A decision tree is a supervised learning algorithm that builds a tree-like model of decisions and their possible consequences. It recursively splits the input data based on the features to create a tree structure, where each internal node represents a decision based on a specific feature, and each leaf node represents a class label or outcome. Decision trees are intuitive, interpretable, and can handle both categorical and numerical features [6].

- **Random Forest:** Random Forest is an ensemble learning method that utilizes multiple decision trees to make predictions. It creates an ensemble of decision trees, where each tree is trained on a random subset of the training data and features. During prediction, each tree in the random forest independently predicts the class label, and the final prediction is determined by majority voting or averaging the individual predictions. Random forests are known for their ability to handle high-dimensional data, reduce overfitting and provide robust predictions [7].

2.5 Handling Imbalanced Data

Imbalanced datasets can pose challenges in text classification and may lead to biased or inaccurate models. The two most popular data balancing techniques available are oversampling and undersampling. Undersampling involves discarding a significant portion of the majority class instances, which can result in the loss of valuable information. By reducing the training data for the majority class, the classifier may not capture the full range of patterns and variations in that class, leading to a less accurate model. On the other hand, random oversampling includes selecting random examples from the minority class with replacement and supplementing the training data with multiple copies of this instance, hence it is possible that a single instance may be selected multiple times [8].

2.6 Tokenization

Tokenization involves identifying and dividing the text into pertinent chunks. The fundamental building blocks for further NLP tasks like text classification are provided by these units. Tokenization aims to convert unstructured text input into a structured format for computational analysis [9].

2.7 Lemmatization

A lemma is the base or canonical form of a word and lemmatization is a linguistic and natural language processing (NLP) technique that attempts to change words into lemma form. Lemmatization reduces inflected or derived words to a standard form to improve text data analysis and comprehension [10].

2.8 Hyperparameters

Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning. The prefix 'hyper' suggests that they are 'top-level' parameters that control the learning process and the model parameters that result from it. Support Vector Machine (SVM) have a number of hyperparameters that can be adjusted to enhance the SVM model's functionality. Here are some typical SVM hyperparameters [11]:

- **Kernel :** kernel function used to transform the input data into a higher-dimensional feature space. The kernel function plays a crucial role in SVMs as it defines the similarity measure between data points and enables the creation of non-linear decision boundaries. The kernel function can take other values such as linear, poly, rbf or sigmoid.

- **Regularization Parameter :**

  Regularization is a technique used to prevent overfitting which occurs when a model becomes too complex and starts to fit the training data too closely, resulting in poor

generalization to unseen data.
Higher ’C’ value: The SVM will try to classify all training examples correctly, potentially leading to a more complex decision boundary. This can be useful when the training data is noisy or contains overlapping classes.
Lower ’C’ value: The SVM will aim for a larger margin and tolerate more misclassified examples. It promotes a simpler decision boundary and helps to avoid overfitting. This can be effective when the training data is clean and well-separated.
Choosing the right ’C’ value: The optimal ’C’ value depends on the specific dataset and problem at hand. It is typically determined through hyperparameter optimization techniques like grid search or Bayesian optimization. Cross-validation is often used to assess the model’s performance for different ’C’ values and select the one that generalizes the best.

- **Gamma (Kernel Coefficient):** The ’gamma’ parameter in Support Vector Machine (SVM) models determines the influence of each training example on the decision boundary. It is a hyperparameter that affects the flexibility and sensitivity of the SVM model to the training data.Gamma can take the value of scale, auto, or a float value.

- **Degree (Polynomial Kernel):** If the SVM uses a polynomial kernel, the degree hyperparameter determines the degree of the polynomial function. Higher degree values can capture more complex relationships in the data but also increase the model’s complexity.

2.9 Hyperparameter-tuning with Bayesian Optimization

Hyperparameter-tuning is the process of searching the most accurate hyperparameters for a dataset with a Machine Learning algorithm. Bayesian optimization’s main goal is to simulate the unknown objective function that correlates hyperparameter adjustments to related performance values. A probabilistic model (typically a Gaussian Process) is updated to represent the connection between hyperparameters and performance by iteratively choosing hyperparameter configurations to evaluate based on previous observations [12].

2.10 Ekman’s Model

The study of emotions has been a subject of interest across various fields including psychology, neuroscience, and philosophy. Understanding and categorizing emotions is crucial for gaining insights into human behavior, mental health, and social interactions. Several models have been developed to classify and study emotions. Among them Ekman's model is the most prominent one because of its universal applicability, supported by extensive cross-cultural research, making it a robust framework for understanding emotions across different cultures. This model was developed by psychologist Paul Ekman in the 1992. This model posits that there are six basic, universally recognized emotions [13]. These emotions are believed to be innate and cross-culturally consistent and serve as building blocks for human emotional experiences. These emotions are:

- **Happiness:** This emotion is characterized by feelings of joy, contentment and general well-being. It is often accompanied by a smiling expression which is considered a universal indicator of happiness.
- **Sadness:** Sadness involves feelings of sorrow, grief or unhappiness. Physiologically, it is typically associated with a downturned mouth and in some cases tears.
- **Anger:** Anger encompasses feelings of frustration, irritability or hostility. It manifests physically through clenched fists, narrowed eyes and a furrowed brow.
- **Fear:** Fear arises in response to perceived threats or danger. It triggers a heightened state of arousal known as the "fight or flight" response. Physiologically, fear is often depicted through widened eyes and a tense facial expression.
- **Disgust:** This emotion involves feelings of aversion or revulsion towards something unpleasant or offensive. It is expressed through a wrinkled nose and an expression of distaste.
- **Surprise:** Surprise occurs in response to unexpected or startling events. It is characterized by widened eyes, raised eyebrows and an open mouth.

2.11 Cosine similarity

In natural language processing and information retrieval, cosine similarity is a fundamental metric that measures the similarity between two non-zero vectors in a multidimensional space. Instead of calculating the magnitude of these vectors, it calculates the cosine of the angle that separates them, indicating their directional alignment. Cosine similarity is a vital tool in text analysis since it allows one to compare the semantic similarity of documents or textual parts. It efficiently computes similarity scores even in complex feature spaces, facilitating its application to large-scale datasets and intricate textual corpora. Additionally, cosine similarity offers an interpretable and intuitive measure of similarity, with scores ranging from -1 (indicating dissimilarity) to 1 (representing identical documents). Some of the notable types of cosine similarity include [14]:

- **Standard Cosine Similarity:** This is the basic form of cosine similarity, which measures the cosine of the angle between two vectors in a multidimensional space.
- **TF-IDF Weighted Cosine Similarity:** This version takes into account the Term Frequency-Inverse Document Frequency (TF-IDF) values of terms in documents. It is commonly used in information retrieval and text mining tasks.
- **Jaccard Similarity:** While not exactly cosine similarity, Jaccard similarity is a related metric used to compare the similarity between sets of elements. It is commonly used in areas like document clustering and recommendation systems.

2.12 Transfer Learning

A model that has been pre-trained on a large set of data for specific tasks is fine-tuned or adjusted for a new but similar task using the machine learning technique known as transfer learning. By using the information from the first work, transfer learning makes use of less data and more efficient methods to complete the new task than training a model from scratch on a new dataset. For instance, a model that has been pre-trained to identify objects in a large dataset of natural images can be fine-tuned for a particular classification task such as differentiating between bird species, using a smaller dataset. When it comes to saving time and computational resources, this method can be far more efficient than building a model from the start. Transfer learning is a popular technique for enhancing model performance on tasks with less labeled data. It is applied in many different fields such as speech recognition, computer vision and natural language processing [15].

Chapter 3

Literature Review

Ontoum and Chan [16] utilized the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology to guide their learning process. To expedite development and minimize the development cycle, they combined CRISP-DM with an agile methodology, which is known for its rapid and iterative approach to software development.To enhance the accuracy of their classification task, the researchers decided to implement four binary classifications instead of a more complex 16-class multiclass classification. This approach aimed to simplify the task and improve the precision of their predictions.The initial step in their data preprocessing phase involved cleaning the text data. They utilized Python's NLTK (Natural Language Toolkit) to remove stop words, which are commonly used but carry little semantic value. This step helped reduce noise in the text data, improving the quality of the subsequent analysis.Following the cleaning step, the researchers performed lemmatization to normalize the words in the text. Lemmatization is a technique that transforms words into their base or dictionary form, reducing different variations of words to a common representation. This process helps to ensure consistency and improve the accuracy of subsequent analyses.For the Naive Bayes and Support Vector Machine Classifier models, tokenization was applied after lemmatization. The transformed words were tokenized using NLTK's word tokenizer, breaking the text into individual tokens or words. The text was then converted into a frequency representation using techniques such as Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). These techniques aimed to capture the importance and presence of words in the text, enabling the models to learn from these patterns.In the case of Recurrent Neural Networks (RNN), a Keras word tokenizer was used for tokenization after lemmatization. This allowed the RNN model to process the text in a sequential manner, considering the order and context of the words. Moving to the modeling phase, the researchers trained Naive Bayes and Support Vector Machines classifiers on the preprocessed data. These classifiers achieved overall accuracies of 41.03% and 41.97% respectively, indicating their effectiveness in predicting the desired outcomes.To further improve the performance of the Recurrent Neural Networks model, the researchers introduced a CONV1D layer, which enhanced the model's ability to capture local patterns in the text. Additionally, they added a Bi-directional Long Short-Term Memory (BI-LSTM) layer, which allowed the model to capture long-term dependencies and context. This modified RNN model achieved an overall accuracy of 49.75%, indicating its improved performance compared to the other models.However, despite the good accuracy achieved by the four binary classifications in each model, the researchers noted a general weakness in accurately classifying all four dimensions of the MBTI (Myers-Briggs Type Indicator). This suggests that there is still room for improvement in the models' ability to accurately classify all aspects of personality traits, highlighting a potential area for future research and development.

The major goal of the experiment conducted by Cui and Qi [17] was to identify the machine learning and deep learning models that are most successful in correctly predicting a person's personality from their social media posts. The feature selection process involved featurizing the posts using a "bag of words" approach. The size of the bag, denoted as B, was tuned as a hyperparameter and it was found that the most effective value for B was 50,000. Additional features such as bigrams, skip grams, part of speech tags, and capital letter count were also considered in the featurization process.The initial model used was a multiclass Softmax classifier, aiming to predict all 16 personality types. The baseline performance of this model showed a training accuracy of 19% and a test accuracy of 17%, which outperformed randomly choosing classes by 6.25%. However, in order to improve accuracy, the approach was modified to use four binary classifiers instead of a 16-class multi-class classifier.Naive Bayes was employed to predict the binary classes of (Extrovert/Introvert, Sensing/Intuition, Thinking/Feeling, Judging/Perceiving), achieving accuracies of 75%, 84.5%, 62.4%, and 60.3%, respectively. The overall accuracy of the Naive Bayes model improved to 26%, a significant improvement compared to the Softmax baseline of 17%.Support Vector Machines (SVM) were also utilized, using the hinge loss and aiming to maximize margin through L2 regularization. The hyperparameters were carefully tuned, and minibatch Stochastic Gradient Descent was employed until convergence. An ablative error analysis revealed that the preprocessing step had the largest impact on classifier accuracy. Therefore, focusing on improving the preprocessing steps was identified as a potential avenue for achieving better results. Deep learning techniques were explored, employing a multi-layer long-short-term memory (LSTM) recurrent neural network as the encoder and a 3-layer feed-forward neural network with rectified linear unit (ReLU) activations as the decoder. The decoder provided the probability of each class through a softmax function. The 16-class classifiers achieved an accuracy of 23%, while the four binary classifiers showed an accuracy of 38%. The paper also discussed a comparison of various deep learning parameters, examining their impact on the development and test accuracy of the models. Furthermore, it suggested potential future work, such as exploring alternative mechanisms for representing word embeddings, including char and k-char word embeddings, and utilizing pretrained GloVe vectors.Overall, the study aimed to identify the most effective machine learning and deep learning models for accurately predicting personality traits from social media posts, and it highlighted the importance of feature selection, preprocessing steps, and parameter optimization in achieving better classification performance. The future work suggestions indicate the potential for further improvements and advancements in the field of personality trait prediction using text classification models.

In their research [18], Pradhan et al. utilized a labeled dataset from a social network to develop a machine learning model for predicting personality types. The ultimate goal was to deploy this model on a website as a personality predictor. The methodology employed in this study consisted of four main steps: text mining, machine learning model development for classification, deep learning model development for classification, and the creation of a website for personality prediction.To preprocess the dataset, several rounds of data cleaning were conducted. This involved removing unnecessary punctuation, emoticons, and stop words using regular expressions in Python and the NLTK text processing library. In the feature selection stage for machine learning, it was essential to assign weights to words based on their frequency of use to classify personalities accurately. The count vectorizer and TF-IDF techniques were employed to convert the unstructured text into vectors, providing a structured representation of the data. The performance of the machine learning models yielded approximately 60% accuracy. However, the Naive Bayes classifier demonstrated only 32% accuracy due to high bias and the challenge of multi-class classification. To address this, a random forest classifier was trained, achieving an accuracy of 36.03%. Additionally, the linear support vector machine algorithm achieved an accuracy of 57.9%. For the deep learning approach, feature selection played a crucial role. An embedding layer was used to transform each word into a 1-dimensional vector, which was then fed into the CNN layer as input. Weights and biases were assigned to these input vectors based on features and max-pooling techniques. This approach achieved around 81.4% accuracy in multi-label classification. Instead of predicting 16 classes for the 16 personality types, four binary classifiers were utilized to enhance precision, recall, and accuracy.The study acknowledged certain limitations. The model exhibited bias since some personality types were more common than others in the dataset. To mitigate this, the researchers proposed scraping more information about the minority personality types to address the imbalance. Additionally, they suggested exploring the use of an RNN model to improve the overall accuracy of the predictions. In summary, they implemented a comprehensive methodology involving text mining, machine learning, and deep learning techniques to develop a personality prediction model. Their results demonstrated significant progress in accurately predicting personality types, with potential areas for further improvement and future research.

In their study [19], Bharadwaj et al. utilized a dataset consisting of 8,660 individuals'

Twitter posts, with each individual contributing a total of 49 posts. The researchers focused on preprocessing the data to ensure its quality and relevance for further analysis.During the data preprocessing stage, the tweets were carefully cleaned to remove any links, numbers, or punctuation marks. Additionally, lemmatization and stemming techniques were applied using WordNet Lemmatizer and Lancaster Stemmer, respectively, to standardize the words in the sentences. Tweet Tokenizer was employed to tokenize the tweets, breaking them down into individual words. The top 1,500 most frequent words in the tweets were selected to represent the overall content.To generate the feature vector, multiple sources of information were combined. TF-IDF (Term Frequency-Inverse Document Frequency), EmoSenticNet, LIWC (Linguistic Inquiry and Word Count), and ConceptNet features were integrated into a comprehensive representation of the tweets. EmoSenticNet, a sentiment analysis model, was used to analyze the emotional tone of the text. LIWC, on the other hand, focused on analyzing the linguistic content and identifying language patterns associated with psychological and social processes. ConceptNet, an open-source semantic network, provided information about word meanings and relationships. The resulting feature vector had a size of 1,850, incorporating various aspects of the tweets. To reduce the dimensionality of the feature space and improve computational efficiency, the researchers employed Singular Value Decomposition (SVD) and reduced the total number of features to 50.In the training phase, the four MBTI (Myers-Briggs Type Indicator) classes were individually treated by the training module. Each class comprised two traits, and binary values of 0 or 1 were assigned to represent the presence or absence of these traits. Based on the feature vectors, Support Vector Machines (SVM) and Naive Bayes classifiers were trained. Additionally, a neural network with softmax output was trained using the data.To evaluate the performance of the models, three metrics, including accuracy, were employed. Across all MBTI classes, SVM consistently outperformed other classifiers. Specifically, it was observed that SVM yielded the best results when using the LIWC and TF-IDF feature vector combination without feature reduction for the I/E (Introversion/Extraversion) and S/N (Sensing/Intuition) traits. Moreover, the performance of SVM was found to improve when incorporating ConceptNet and EmoSenticNet features and reducing the number of features, indicating the importance of these additional factors in enhancing the classification accuracy. Through their comprehensive approach, they successfully leveraged various feature selection techniques and machine learning models to analyze and classify individuals' Twitter posts based on their MBTI traits. The results highlighted the effectiveness of SVM in this context and emphasized the significance of incorporating diverse features for improved performance.

In their study [20], Brindha et al. aimed to evaluate the classification accuracy of various algorithms for distinct datasets. The researchers examined the performance of K Nearest Neighbor (KNN), Naive Bayes, Support Vector Machine (SVM), Decision Tree, and Logistic Regression algorithms.K Nearest Neighbor is a widely used pattern recognition algorithm known for its effectiveness. However, it has some limitations. One challenge is the calculation complexity associated with using all training data. Finding similarities between samples takes more time when there are fewer similarities present. Additionally, KNN requires recalculating if there is even a small change in the training data since it solely relies on that data. Another limitation is the lack of weight difference between samples, which may affect classification accuracy.Naive Bayes is a quick and scalable algorithm commonly used for reference scoring and model building. However, it faces an issue when there is a certain attribute value that has no instances of a class label, potentially impacting its performance. Support Vector Machine (SVM) transforms training data into a top-dimensional space using a nonlinear mapping. Although SVMs can train slowly, they excel at forming composite nonlinear decision boundaries. Decision Tree algorithms have shown promise in solving land cover mapping issues. They are easy to interpret due to the transparent decision tree structure. However, overfitting can be a drawback, especially when dealing with a large number of dictionary entries. Despite this, decision trees generally perform well in categorization tasks.Logistic Regression models are typically used to determine the probability of class membership for one of the two categories in a dataset. They are commonly employed in classification tasks. The researchers utilized datasets DS1 and DS2, varying in sample sizes from 200 to 5000 records, to evaluate the performance of KNN, Naive Bayes, SVM, Decision Tree, and Logistic Regression algorithms. For DS1, the error rate of KNN, SVM, and Decision Tree algorithms increased as the sample size increased. In contrast, the error rate decreased for Naive Bayes and Logistic Regression algorithms with increasing sample sizes. In the case of DS2, the error rate increased as the sample size increased, although the rate of increase was less noticeable for Naive Bayes and Logistic Regression compared to other methods. Hence, it can be concluded that a larger sample size does not necessarily guarantee a lower error rate.Regarding the evaluation metrics, SVM outperformed all other algorithms in terms of precision, recall, and F1 measure. The results indicated that SVM yielded the highest classification performance among the tested algorithms. In summary, their study provided valuable insights into the classification accuracy of different algorithms for distinct datasets. The findings shed light on the strengths and limitations of each algorithm and highlighted SVM's superior performance in terms of precision, recall, and F1 measure.

The link between personality and emotions is found in how a person's personality traits influence how they experience, express and control their emotions, ultimately determining their overall emotional responses and coping techniques in various life situations. Reisenzein and Weber [21] provided insights into the relationship between personality and emotion. They outlined a model of the emotion system as a subsystem of personality, highlighting the five major dimensions of personality postulated by the five-factor model, including neuroticism, extraversion, agreeableness, and openness to experience. It discusses the role of appraisal in determining emotions and suggests that inter individual differences in cognitive and motivational structures, as well as information-processing procedures, may influence appraisal and subsequently emotions. The paper also mentions the distinction between non automatic and automatic modes of appraisal and emotion generation, with non automatic appraisals being conscious inference strategies and automatic appraisals being triggered directly by perception. It emphasizes the importance of chronic accessibility of appraisal-relevant cognitive and motivational structures such as memory schemas, in determining appraisals and emotions. They mainly highlighted the influence of relatively stable and general desires, beliefs and preferences on appraisals and subsequent emotions. It is concluded that emotions are influenced by both cognitive and motivational factors, and that the appraisal of an event determines not only whether it elicits an emotion but also which emotion it elicits.

Li et al. [22] focused on multitask learning for emotion and personality detection, building on the correlation between personality traits and emotional behaviors. They proposed a novel multitask learning framework, SoGMTL, that simultaneously predicts personality traits and emotions. They discussed different information-sharing mechanisms between the two tasks and adopted a MAML-like framework for model optimization. They applied the Sigmoid function as the gate between two different models (SiGMTL) and the cross-entropy model as the gate (CAGMTL). They also introduced the Sigmoid weighted linear model as the gate (SiLMTL) and compared it with the proposed SoGMTL model. The methodology involves training the model on multiple famous personality and emotion datasets and evaluating its performance. The paper mentioned the use of multiple famous personality and emotion datasets for training and evaluation. The experiment resulted on the ISEAR dataset and the TEC dataset are presented and compared. Among all their proposed models SoGMTL+ MAML achieved high performance in emotion and personality detection tasks. The paper discussed the effectiveness of the SoGMTL model in personality trait detection and emotion detection, showing clear boundaries in predictions for personality traits and clear differences in maximum values for emotions, achieving state-of-the-art performance across multiple datasets. The performance on the TEC dataset is generally greater than that on the ISEAR dataset, possibly due to the larger amount of data in TEC . Future work is mentioned briefly, suggesting the inclusion of more contextual information such as personal preferences and current location to enhance the prediction capabilities of the model.

Nasir et al. [23] presented a text-based emotion prediction system using machine learning algorithms such as Multinomial Naïve Bayes, Support Vector Machine, Decision Trees, and k-Nearest Neighbors. The system aims to recognize and predict emotions based on Ekman's six basic emotions: anger, fear, disgust, joy, guilt, and sadness. Data pre-processing techniques like stemming, stop-words removal, spelling correction and tokenization were implemented. The system achieved an average accuracy of 64.08% using the Multinomial

Naïve Bayes classifier compared to SVM (15.37%), Decision Tree classifiers (52.36%) and KNN (47.95%). So, the best model was integrated into a graphical user interface using the Python Tkinter library. The paper also discusses performance measures such as tf-idf and count vectorizer, confusion matrix, precision-recall rate and ROC score. They described the process flow, strategy and components of the research, including the materials and methods used. It mentions the use of a benchmark dataset, ISEAR, for testing the models. The methodology section provides an overview of the different techniques used in data pre-processing and machine learning classification algorithms. It also discusses the lexicon-based approach and the construction and development of algorithms for supervised and unsupervised learning. The paper aims to evaluate the performance of four different text classifiers: Support Vector Machine, Naive Bayes, k-Nearest Neighbour and Decision Tree. They also provided conclusions and recommendations for future research works. It emphasized the importance of text-based emotion recognition in various applications, such as suicide prevention and customer satisfaction analysis. They suggested that the developed system can be further improved and extended to handle different languages and cultural contexts. Overall, the paper contributes to the field of text-based emotion prediction using machine learning approaches and highlights the potential for further advancements in this area.

Agrawal and An [24] proposed an unsupervised context-based approach to detecting emotion from text at the sentence level, without relying on manually crafted affect lexicons. It computes an emotion vector for each potential affect-bearing word based on semantic relatedness between words and various emotion concepts. Emotion detection from text is a relatively new classification task. The approach allows for the classification of sentences beyond Ekman's model of six basic emotions. The proposed methodology is an unsupervised context-based approach for emotion detection from text at the sentence level. The method computes an emotion vector for each potential affect-bearing word, referred to as NAVA (Noun Adjective Verb Adverb) words, from the input sentence based on semantic relatedness between words and various emotion concepts. The scores are then fine-tuned using the syntactic dependencies within the sentence structure. The methodology utilizes Pointwise Mutual Information (PMI) to compute the semantic relatedness between words, enriching it further using context-dependency rules. PMI is used to measure the similarity between relevant words and representative words for each emotion, taking into account their context. Their evaluation on various datasets shows that the framework is a more generic and practical solution to the emotion classification problem, yielding significantly more accurate results than recent unsupervised approaches. They compared three machine learning algorithms on a movie review dataset and concluded that SVM performs the best. The evaluation on the Aman Blog dataset shows the unsupervised results of the proposed approach alongside supervised approaches using SVM and a keyword baseline. The evaluation of the UnSED algorithm on three standard datasets shows that the text corpus has an effect on the semantic relatedness scores and ultimately on the accuracy of emotion detection. Deriving semantic relatedness scores from multiple measures and testing the use of other syntactic dependencies can be explored as future avenues of their research work. Text classification has gained attention due to the increase in digital documents and is used in various applications such as filtering and recommendation.

In paper [25], Park et al. proposed different approaches to improve text classification performance, including centroid-based classifiers (KNN), multinomial naïve Bayesian (MNB), support vector machines (SVM), and convolutional neural networks (CNN). They introduced a methodology that combines cosine similarity with conventional classifiers (MNB, SVM, and CNN) to enhance their accuracy. The enhanced classifiers were applied to famous datasets (20NG, R8, R52, Cade12, and WebKB) and showed significant increases in accuracy. They also explored the suitability of two knowledge representation techniques, word count and term frequency-inverse document frequency (TFIDF) for different classifiers. The performance of the constructed classifiers was evaluated using accuracy in the confusion matrix, with higher values indicating more accurate results. TFIDF-based SVMs performed better than word count-based SVMs, except for the WebKB dataset where word count-based SVMs were in the top three. In terms of performance, SVMTFIDF (TFIDF-based SVM) showed higher accuracy compared to SVMWC (word count-based SVM) in most cases. MNBTFIDF (TFIDF-based MNB) showed worse performance compared to MNBWC (word count-based MNB) across all datasets. Overall, all the experimental results showed the most suitable knowledge representation technique for each classifier, with TFIDF being more suitable for SVM and cosine similarity, while word count was more suitable for MNB. So, the study concludes that the methodology combining cosine similarity with conventional classifiers significantly enhances their performance in text classification, as evidenced by the outstanding results in terms of accuracy.

Chapter 4

Personality Prediction

In this chapter, we detailed the methodology for predicting personality using the MBTI dataset.

4.1 Methodology

The overall methodology of our proposed task is shown in fig. 4.1

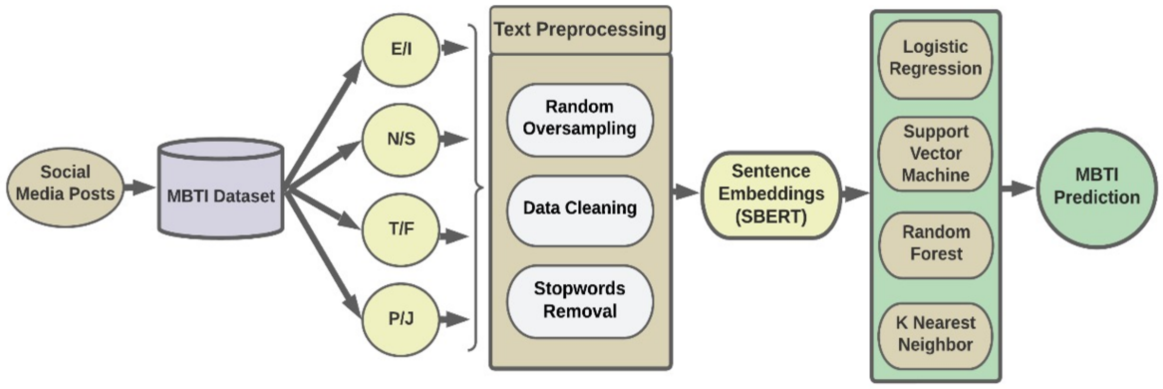


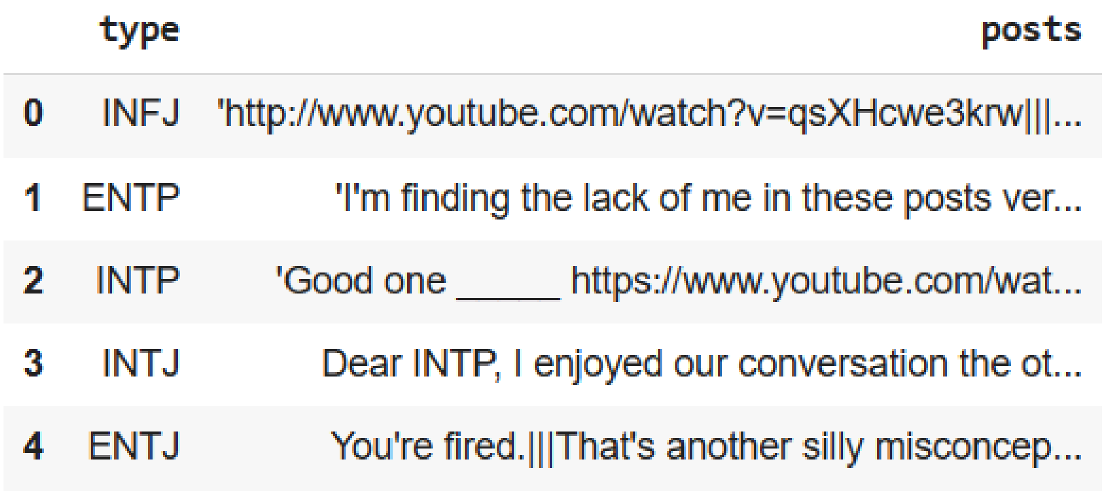
Figure 4.1: Proposed Methodology for MBTI Personality Prediction

1. **Data Collection :** The publicly available Myers-Briggs Personality Type Dataset is acquired from Kaggle [26].
2. **Dataset Modifications :** Divide the dataset into four dimensions in order to execute four independent binary classifications, which together make up a single MBTI personality type.

3. **Random Oversampling :** We can notice a significant data imbalance in each dimension after partitioning the dataset into the four MBTI type dimensions. Therefore, random over sampling is used to address the problem.
4. **Data Preprocessing :** This part does data cleansing and stop word removal.
5. **Feature extraction :** To extract the contextualized feature from the social media posts, a trained Sentence-BERT model is used.
6. **Classification :** We used five different classifiers, including Logistic Regression, Support Vector Machine, Decision Tree,K-Nearest Neighbor and Random Forest, to carry out the four separate dimensions' classification task that we derived from the MBTI types.
7. **Performance Evaluation :** The outcome of each model is computed using a separate performance metric and plotted for comparison.

4.2 Data Collection and Visualization

Along with the user's MBTI personality type, the dataset consists of last 50 things each user has posted (Each entry separated by "|||" (3 pipe characters)) for a total of 8675 users is shown in fig. 4.2. The labels are obtained by allowing the user to provide their MBTI type as account information. The messages are taken from the PersonalityCafe online forum, a venue for all kinds of talks and discussions. The visual representation is shown in fig. 4.3



	type	posts
0	INFJ	'http://www.youtube.com/watch?v=qsXHcwe3krw|||...
1	ENTP	'I'm finding the lack of me in these posts ver...
2	INTP	'Good one _____ https://www.youtube.com/wat...
3	INTJ	Dear INTP, I enjoyed our conversation the ot...
4	ENTJ	You're fired.|||That's another silly misconcep...

Figure 4.2: Example of MBTI dataset

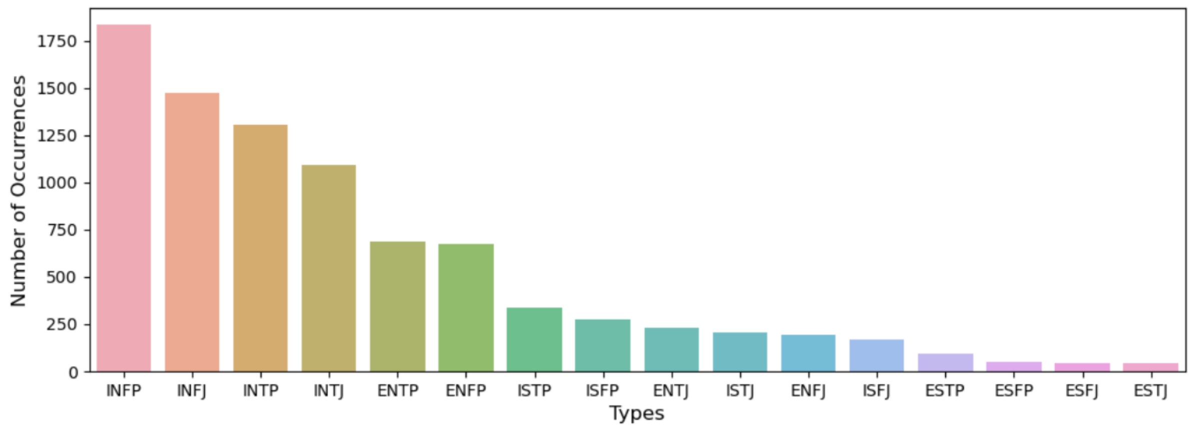


Figure 4.3: Visual Representation of all 16 MBTI types in the dataset

4.3 Dataset Modifications

According to [16], with 16 classes, the accuracy could not be improved because of the large number of classes to predict. Instead of predicting 16 classes for 16 personality types, four binary classifiers were used in order to get better precision, recall, and accuracy. So we added four additional columns (is_introvert,is_sensing, is_thinking, and is_judging) to the dataset that were labeled according to how respondents responded to the ”MBTI” model's four dimensions in order to perform multilabel classification. The process is intended to increase accuracy [17]. When is_introvert is set to 1, it indicates that the respondent is an introvert, while when it is set to 0, it indicates that they are an extrovert. Similar rules apply to the other three columns that identify which dimension of the ”MBTI” model a respondent belongs to.

4.4 Random Oversampling

The dataset is drastically skewed, with an introvert count of 6676 versus an extrovert count of 1999. Also, the following numbers contrast intuition (7478) with sensing (1197), feeling (4694) with thinking (3981), and perception (5241) with judgment (3434). The visual representation is shown in fig. 4.4. Classification algorithms may learn to predict the majority

class more correctly since it predominates the training set due to an unbalanced dataset. This causes the minority class to do poorly. Undersampling, oversampling, and balancing class weights are just a few of the strategies used to remedy the imbalance between classes. The random oversampling strategies worked well in our case, despite the fact that we also experimented with the class weights balancing scheme and undersampling. Random oversampling is the easiest type of oversampling because it essentially duplicates minority case to increase the imbalance proportion. The performance of the machine learning classifier

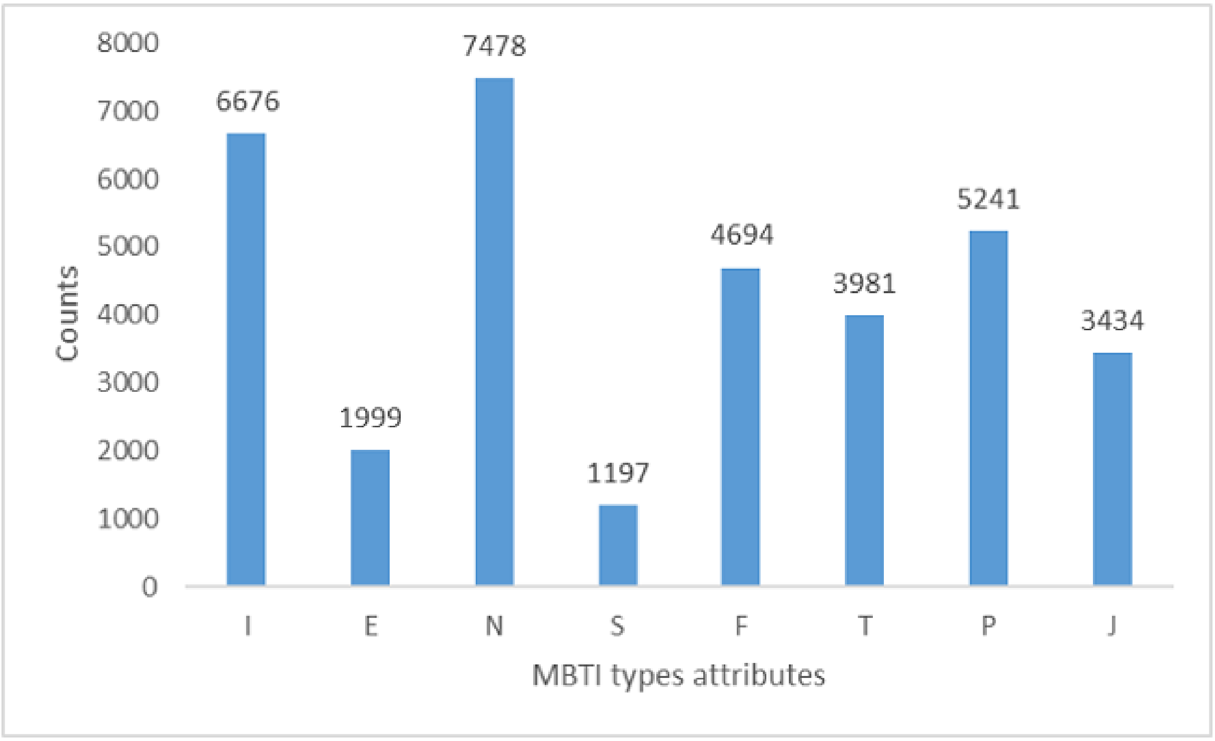


Figure 4.4: Before Random Oversampling

for effective predictions was significantly enhanced by this duplication of minority class enhancement [8]. After performing random oversampling the visual representation is shown in fig. 4.5.

4.5 Preprocessing

1. **Data Cleaning:** Numerous URLs and different punctuation marks were present in the data, which needed to be cleaned up and eliminated. The unnecessary punctuation, pictures, and emojis were all deleted over several rounds of the data cleaning process. Since these don't carry much significant value that can capture a person's personality.

2. **Tokenization and Stop Words removal:** Preprocessing activities like tokenization is performed internally by SBERT model.Tokenizing is the process of breaking a text down into smaller units, called tokens. Tokens are also referred to as individual words or significant textual elements. But stopwords removal have all been carried out using predefined lists of stop words from the Spacy library. Each token in the sentences are lowercased after stopwords removal.

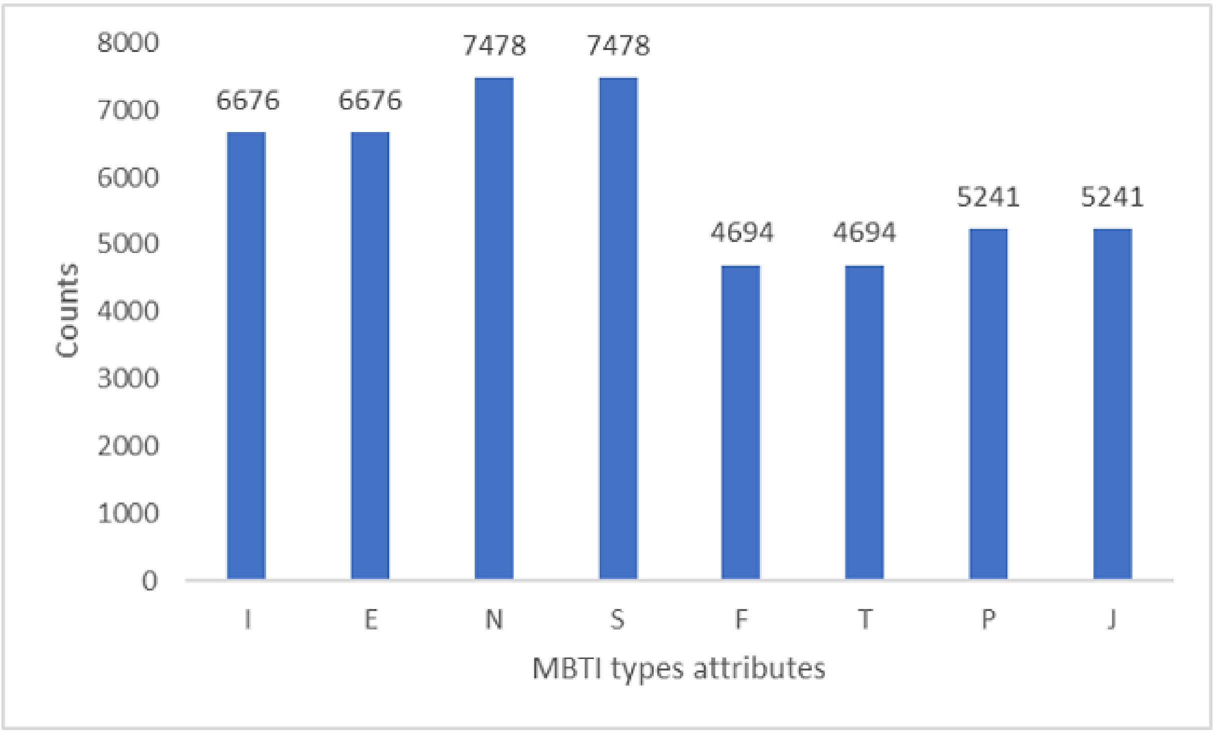


Figure 4.5: After Random Oversampling

4.6 Feature Extraction

The process of feature engineering plays a crucial role in machine learning by selecting and extracting relevant features from raw data. In the domain of Natural Language Processing (NLP), feature engineering involves transforming unstructured text data into a numerical representation that can be effectively utilized by machine learning algorithms.

1. **Sentence Embedding:** In NLP, BERT (Bidirectional Encoder Representations from Transformers) is a widely used model for translating sentences into vector representations. However, the vector space produced by BERT is not optimized for cosine similarity or other popular similarity measures. To overcome this limitation, Sentence-BERT (SBERT) was introduced. SBERT provides a computationally efficient solution and outperforms BERT in terms of clustering large sets of sentences. For example, hierarchical clustering of 10,000 sentences with BERT takes approximately 65 hours, while SBERT reduces the effort to merely 5 seconds [27].

2. **SBERT Model ALL-MPNET-BASE-V2:** SBERT modifies the BERT model, initially designed for generating contextual word embeddings, to produce numerical representations of text data. Instead of word embeddings, SBERT leverages a model known as "All-MPNetBaseV2" to generate sentence embeddings. This approach focuses on contextualized embeddings, wherein the embedding for a given word or sentence considers not only the specific word or sentence but also its surrounding context.

3. **Tokenization and Embedding Construction:** To create sentence embeddings using the "ALL-MPNET-BASE-V2" model, the input sentence undergoes tokenization using the Byte Pair Encoding (BPE) algorithm. This process converts the sentence into a series of subwords. The subword tokens are then processed through multiple layers of the transformer architecture, which results in contextualized embeddings for each subword.

4. **Integration of Subword Embeddings:** To obtain a fixed-length representation of the input sentence, the contextualized embeddings for all subwords are integrated. This can be achieved by taking the maximum pooling across all subwords embeddings. The integrated representation is a 768-dimensional vector for the ALLMPNET-Base-V2 model.

4.7 Classification

1. **Logistic Regression :**
    - Built-in module used : sklearn.linear_model.LogisticRegression
    - Parameters : Default (penalty='l2', solver='lbfgs', C=1.0, etc.)
2. **K-nearest Neighbor :**
    - Built-in module used: sklearn.neighbors.KNeighborsClassifier
    - Parameters: Default (n_neighbors=5, weights='uniform', algorithm='auto', etc.)
3. **Decision Tree :**
    - Built-in module used: sklearn.tree.DecisionTreeClassifier
    - Parameters: Default (criterion='gini', splitter='best', max_depth=None,min_samples_split=2, etc.)
4. **SVM :**
    - kernel : rbf
    - degree : 3
    - Regularization parameter C : 1.0
5. **Random Forest :**
    - Built-in module used: sklearn.forest.RandomforestClassifier
    - All parameters are kept default

Chapter 5

Emotion Prediction

In this chapter, we correlated the MBTI dataset with human emotion.

5.1 Methodology

The overall methodology of our 'Emotion Prediction' task is shown in fig. 5.1

1. **Data Collection :** Two new types of datasets are being employed. "Emotions in text" is one, and "NRC Emotion Lexicon" is another, and both are publicly available. [28] and [29]

2. **Dataset Modifications :** In the "Emotions in text" dataset, we only kept three basic emotions: 'joy', 'sad', and 'anger' and we excluded emotions like 'fear', 'love', and 'surprise'. We only looked at the words associated with the emotions 'joy', 'sad', and 'anger' in the "NRC Emotion Lexicon" dataset.

3. **Random Oversampling on "Emotions in text" :** We can notice a significant data imbalance in each emotion in "Emotions in text" dataset. Therefore, random over sampling is used to address the problem.

4. **Data Preprocessing :** This part cleans the data and removes stop words from the "Emotions in Text" dataset.

5. **Feature extraction :** A trained Sentence-BERT model is used to extract the contextualized feature from the texts of the "Emotions in Text" dataset.

6. **Emotion Vector of "NRC Emotion Lexicon" :** We build emotion vectors by selecting words from the dataset " NRC Emotion Lexicon" that express distinct emotions.

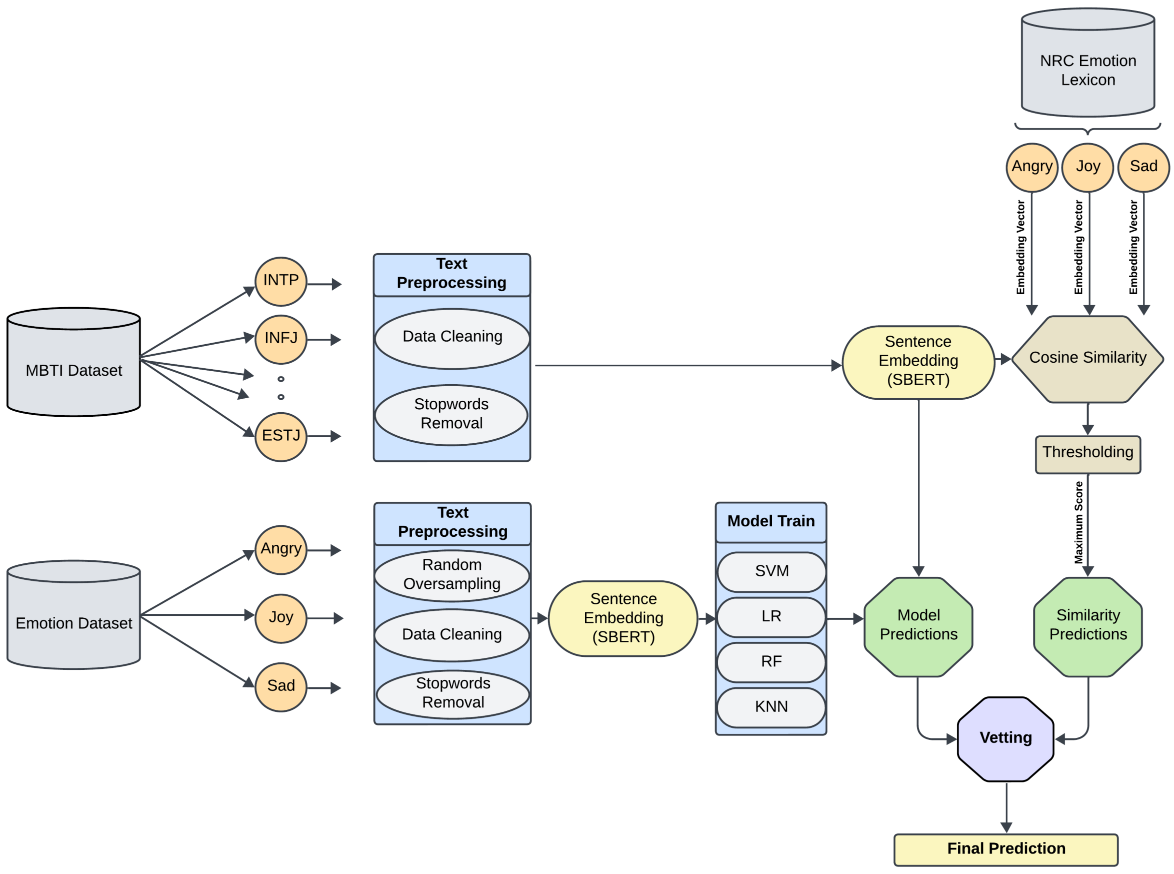


Figure 5.1: Proposed Methodology for Emotion Prediction

7. **Cosine Similarity using "NRC Emotion Lexicon" :** Find the cosine similarity scores between the vectors of different emotions and social media postings from the "MBTI" dataset.

8. **Model Training Using "Emotions in text" :** We used the "Emotions in Text" dataset to train four distinct models, including Logistic Regression, Support Vector Machine, Decision Tree, and Random Forest.

9. **Model Selection and Predictions :** The outcome of each model is evaluated using a performance metric, and the best model is selected for prediction.

10. **Vetting Process:** A vetting method is used to syncronize the outcomes of two independent approaches into a single result.

5.2 Data Collection and Visualization

Two emotion datasets are required for our emotion detection system; one is used for model training and the other for similarity measurement with the MBTI dataset. The "NRC Emotion

Lexicon" is used to measure similarity. It is basically a list of several English words with eight emotions (feelings anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (positive and negative). Here, it is also noteworthy that the "NRC Emotion Lexicon" includes Ekman's six fundamental emotions. However, the "Emotions in text" dataset, which has two columns labeled "Text" and "Emotions," was utilized to train our models. Six different emotions are included in this dataset: happy, sadness, love, fear, anger, and surprise. The visual representation is shown in fig. 5.2

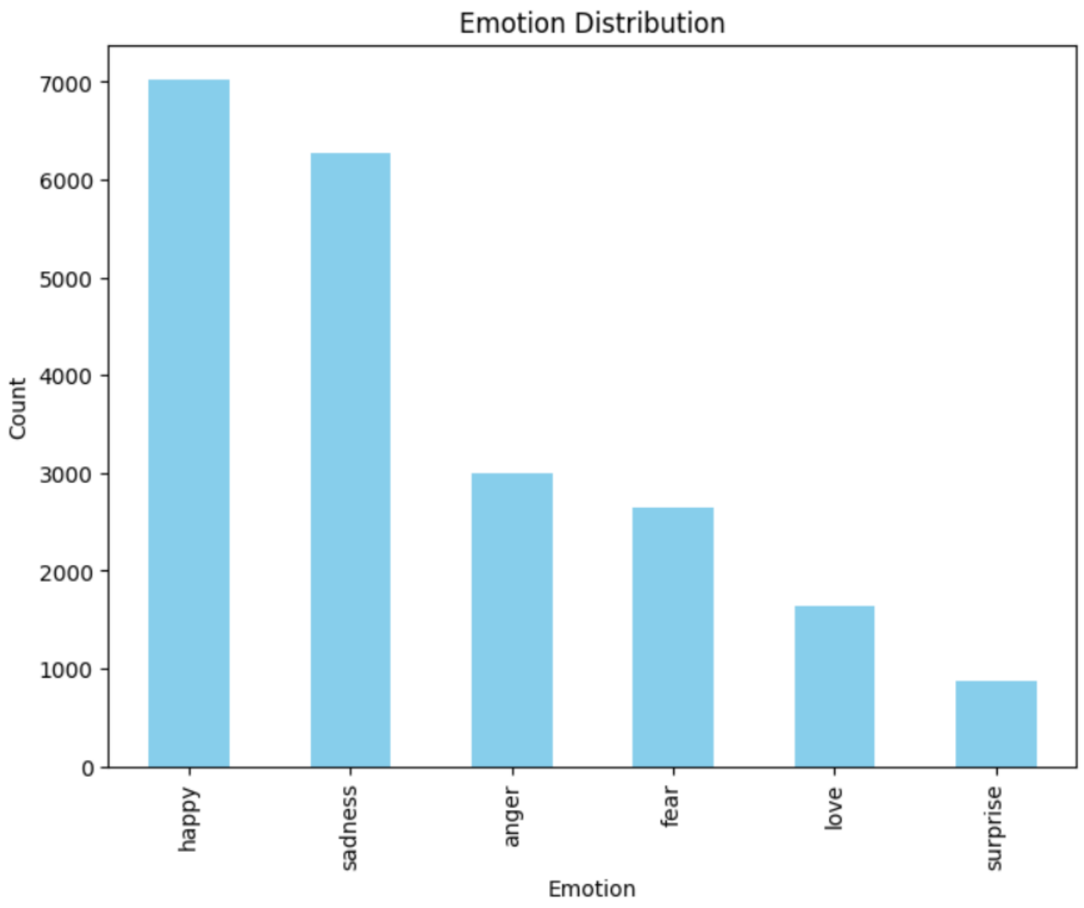


Figure 5.2: Visual Representation of "Emotions in text" dataset

5.3 Dataset Modifications

In the "Emotions in Text" dataset, we made a decision to focus on three specific emotions: 'joy','sad', and 'anger' instead of the broader set of six emotions, which includes 'joy','sad', 'anger', 'love', 'fear', and'surprise'. This is because initially we tried to find all six emotions from the MBTI dataset, and we noticed that a very small number of posts express 'love', 'fear', and 'surprise' emotions, and in some cases these emotions were found null. Emotion classification can vary widely across different genres, writing styles, and contexts. This decision could be driven by the observation that in the texts we analyzed or worked with, expressions of love, fear, and surprise were not as prevalent or significant. So the choice to narrow down the emotional focus was based on a detailed analysis of the specific content under consideration. By observing the patterns within the texts, it became apparent that joy, sadness, and anger are the predominant emotional themes, and a more detailed exploration of these emotions were present in the dataset. Focusing on joy, sadness, and anger may provide a more streamlined and targeted approach, so we discarded love, fear and surprise as they were negligible for the content of the texts. Lastly, we took approximately 90–100 words for each emotions only when they were associated with 'joy','sad' and 'anger' from the "NRC Emotion Lexicon" dataset.

5.4 Random Oversampling on "Emotions in text"

There appears to be a discrepancy in the "Emotions in text" dataset. The counts of the three target emotions in the dataset—'sad' (5362), 'anger' (4666) and 'joy' (2159)—show a notable imbalance. Unbalances in classification issues may lead to models that are biased toward predicting the majority class, which may make it more difficult for the model to predict minority classes properly because of the larger representation of the minority class in the dataset. Oversampling allows the model to retain all instances from the minority class, preserving valuable information that might be lost in undersampling [8]. We thus decided on oversampling, as it corrects the imbalance and guarantees a more reliable and objective classification model. After random over-sampling:

- Sad: 5362
- Anger: 5362
- Joy: 5362

5.5 Data Preprocessing

In the process of preparing the dataset for emotion prediction, a comprehensive series of preprocessing steps were carefully undertaken, the same as we did in the personality prediction work. Before eliminating stop words, we cleaned up the data and tokenized it. The dataset becomes more valuable for the categorization issue after completing all of these procedures (data cleaning, tokenization and stop words removal). By going through these preprocessing steps, the dataset becomes more refined, structured, and compliant to machine learning algorithms. The goal is to create a clean, normalized, and appropriately formatted dataset that allows the model to learn patterns and relationships effectively during the training phase.

5.6 Feature Extraction

In the context of predicting emotions and personality traits within Natural Language Processing (NLP), the process of feature engineering plays a pivotal role. Just as in the domain of personality prediction, where the extraction and transformation of relevant features from raw text are crucial, a similar methodology is applied to emotion prediction. Sentence Embedding techniques, such as Sentence-BERT (SBERT) with its ALL-MPNET-BASE-V2 variant, are employed to translate unstructured text into numerical representations optimized for machine learning algorithms. This approach, initially developed for personality analysis, proves equally valuable in emotion prediction. By utilizing contextualized embeddings and tokenization through techniques like Byte Pair Encoding (BPE), the model captures nuanced information from the surrounding context of words or sentences. The integration of subword embeddings, achieved through methods like maximum pooling, results in fixed-length vectors, enhancing the model's capacity to discern and predict both emotions and personality traits effectively. This shared feature engineering approach showcases the versatility of NLP models in addressing diverse tasks related to human communication and expression.

5.7 Emotion Vector of "NRC Emotion Lexicon"

We used the "NRC Emotion Lexicon" dataset as the methodological foundation for creating emotion vectors of three distinct emotions:'joy','sad' and 'anger' for the purpose of similarity measurement. This lexicon gives us the ability to choose word sets that are relevant to particular emotions by offering a thorough list of English words associated with various emotions. We took approximately 90–100 words for each emotion only when they were associated with 'joy','sad' and 'anger'. Subsequently, these carefully chosen words underwent vector representation to facilitate machine understanding. This transformation involved converting each word into a numerical vector using Sentence-BERT (SBERT). These vectors allow the machine learning model to capture the subtle meanings associated with each word that addresses strong emotions by encoding semantic links and contextual information.

5.8 Cosine Similarity Using "NRC Emotion Lexicon"

We first used the "NRC Emotion Lexicon" dataset to generate vectors that represented a range of emotions in our emotion classification approach. We next measured the relationship between these emotion vectors and social media posts in the MBTI dataset using cosine similarity scores. The computed similarity scores were quite near each other, which presented a problem for us because it was hard to definitely assign a particular emotion to each social media post. In order to resolve this problem, we implemented a threshold value of 0.03. We only kept the rows when the difference between the highest and second-highest emotion ratings was more than this threshold, which worked as a deciding criteria. In our situation, a lower threshold has led to an increased sensitivity to small differences in cosine similarity scores, making it more difficult to accurately identify specific emotions to social media posts. On the other hand, a threshold value higher than 0.03 have eliminated posts that were emotionally varied, which have resulted in the loss of important predictions. In order to keep posts with stronger emotional signals while filtering out those with subtle emotional variances, we make a trade-off between high and low threshold values and set it at 0.03. After filtering, we chose the emotion that had the maximum score among the three results to determine the final emotion classification. By using this filter, we were able to concentrate on cases in which the emotional signal was more evident, which improved the overall accuracy of our emotion classification. By using this threshold-based method, we greatly decreased ambiguity, which led to a more precise and subtle assessment of the emotions shown in the social media posts from the MBTI dataset.

5.9 Model Training Using "Emotions in text"

In this section, to improve the accuracy of emotion prediction, we carefully selected the "Emotions in text" dataset and implemented four different machine learning models—Logistic Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest, after completing the data preparation procedures (random oversampling, data preprocessing and sentence embedding(SBERT)). We intended to improve our models' ability in identifying and predicting emotions by submitting them to a variety of textual expressions of those emotions in this specific training dataset. Through an explicit training approach, our models were able to identify and understand a wide range of emotional characteristics. This strategy is justified by the need to predict emotions in the "MBTI dataset" by utilizing the strengths of the models trained on the "Emotions in text" dataset. Through particular training on a broad variety of emotional expressions, we hope to strengthen the models and achieve higher accuracy and reliability in emotion predictions for the particular context of the "MBTI dataset." This knowledge and learning transfer from one dataset to another guarantees a more robust and context-aware emotion prediction framework.

5.10 Model Selection and Predictions

In the previous section, we trained four different models using different machine learning techniques (Logistic Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest) using the "Emotions in Text" dataset. We evaluated their performance for emotion classification tasks using appropriate metrics, including accuracy, precision, recall, and F1 score. It was observed that the SVM model outperformed the others in terms of performance. After determining that SVM was the best model, we moved on to using it to predict emotions in the "MBTI dataset." SVM is a good option for emotion prediction because of its reputation for success in binary and multiclass classification problems.

5.11 Vetting Process

After deciding that the Support Vector Machine (SVM) was the best model to use for predicting emotions on the "MBTI dataset" in the previous section, we also took the predictions from the “Cosine Similarity Using Emotions in text” section. In order to improve the accuracy and reliability of our predictions, we implemented a vetting process. This required comparing the SVM model’s predictions with those obtained using cosine similarity. We decided to consider a prediction only when the same emotion was identified by both SVM and cosine similarity. We skipped that case if they didn’t match. By doing this, we hoped to boost confidence in our final predictions, as we were depending entirely on situations in which the text’s emotions could be identified using both approaches.

The vetting process makes sure that our final predictions are more reliable and less likely to include mistakes than they may be when using a single approach. By combining the advantages of the Cosine Similarity method with the SVM model, it produces a more trustworthy result for identifying and classifying emotions in the "MBTI dataset." The combined technique offers a more thorough and accurate representation of the emotional content of the text while minimizing the limits of individual methods.

Chapter 6

Performance Analysis

In this chapter, a comprehensive analysis is conducted on the efficacy of our methodologies through the utilization of diverse performance measures.

6.1 Evaluation Metrics

1. **Confusion Matrix -** Confusion matrices are frequently used in binary classification issues to map the expected class against the true class and produce true positives, true negatives, false positives and false negatives [30].

   (a) True Positive (TP): This represents the cases where the model predicted a positive class correctly. In other words, the model predicted a positive outcome and the actual outcome was also positive.

   (b) True Negative (TN): This represents the cases where the model predicted a negative class correctly. It means the model predicted a negative outcome and the actual outcome was also negative.

   (c) False Positive (FP): This represents the cases where the model predicted a positive class incorrectly. The model predicted a positive outcome but the actual outcome was negative.

   (d) False Negative (FN): This represents the cases where the model predicted a negative class incorrectly. The model predicted a negative outcome but the actual outcome was positive.

2. **Precision -** Precision is the ratio between correctly predicted positive classifications to the total predicted positive classifications.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{6.1}$$

3. **Recall -** Recall is the ratio between correctly predicted positive classifications to all positive and negative in the category.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{6.2}$$

4. **Accuracy -** Accuracy measures the performance of a classifier by finding the ratio of correct classifications against all classifications.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{False Negative} + \text{True Negative}} \tag{6.3}$$

5. **F1 Score -** The F1 score provides a balanced assessment of a model's performance by combining precision and recall into one metric.

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.4}$$

6.2 Result Analysis for Personality Prediction

Table 6.1 displays the overall performance of each model for the 16 MBTI personality types. Although it is clear that logistic regression and KNN outperform other models, overall accuracy is very poor in a multiclass environment. The possible explanation is that the models are unable to differentiate between the number of correctly classified examples of 16 distinct personality classes.

Classifiers	**Accuracy**
Logistic Regression	41.30%
K-Nearest neighbors	41.55%
SVM	24.97%
Decision Tree	27.91%
Random Forest	31.08%

Table 6.1: Accuracy for 16-MBTI Personality Classes

On the other hand, we used five distinct binary classifiers on four MBTI type dimensions. In each of the four scenarios, Random Forest and SVM perform better in terms of accuracy, precision, and other metrics than Logistic Regression, KNN and Decision Tree. According to [18] SVM and Random Forest performs better on text classification. So finally the performance metrics of all five models are shown in table 6.2 , 6.3 and 6.4 and 6.5

Model Name	Accuracy	Precision	Recall	F1 Score
Logistic Regression	74.54%	75.72%	77.22%	73.95%
KNN	73.19%	69.46%	82.77%	75.53%
Decision Tree	86.76%	80.89%	95.73%	87.86%
SVM	85.50%	84.11%	87.55%	85.79%
Random Forest	96.04%	97.36%	94.63%	95.91%

Table 6.2: Extroverts/Introverts

Model Name	Accuracy	Precision	Recall	F1 Score
Logistic Regression	75.26%	74.88%	76.11%	75.47%
KNN	80.38%	73.83%	94.15%	82.76%
Decision Tree	91.50%	85.92%	99.51%	92.31%
SVM	85.20%	83.18%	88.25%	85.64%
Random Forest	99.64%	99.89%	99.42%	99.63%

Table 6.3: Sensors/Intuitives

Model Name	Accuracy	Precision	Recall	F1 Score
Logistic Regression	76.53%	76.08%	77.39%	76.73%
KNN	70.73%	70.76%	70.66%	70.71%
Decision Tree	74.79%	71.62%	81.76%	76.57%
SVM	80.07%	78.82%	82.23%	80.59%
Random Forest	83.62%	84.76%	82.53%	83.62%

Table 6.4: Thinkers/Feelers

Model Name	Accuracy	Precision	Recall	F1 Score
Logistic Regression	69.31%	68.93%	70.13%	69.36%
KNN	63.80%	62.34%	69.62%	65.84%
Decision Tree	74.54%	70.95%	82.83%	76.55%
SVM	73.10%	72.13%	74.93%	73.55%
Random Rorest	84.16%	89.94%	77.31%	82.81%

Table 6.5: Judgers/Perceivers

In every category (I/E, N/S, T/F, and P/J), the Random Forest classifier outperforms the other four machine learning models by a significant margin. Predictions generated by the Random Forest Classifier are more precise and dependable on account of its composition as an ensemble learning algorithm consisting of numerous decision trees. The Confusion

Matrix for Random Forest results are displayed in fig. 6.1. We can see that the True positive and True negative values are high for each MBTI type dimension, which is desirable for a good classification model.

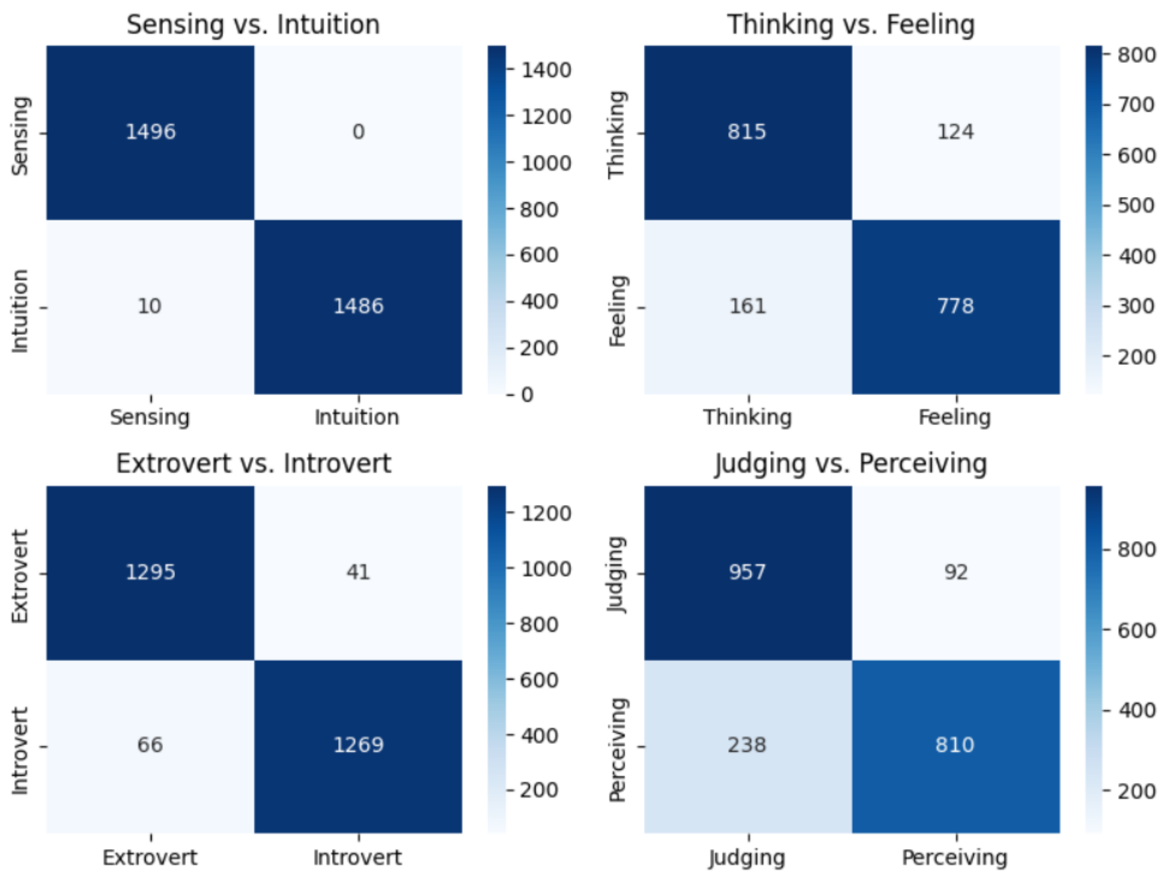


Figure 6.1: Confusion Matrix for Random Forest

6.2.1 Comparison

The experimental in table 6.6 shows that our suggested model outperforms the model [31] that uses BERT for text embeddings. Additionally, it outperforms models that uses Bag of Word (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) in [16] and count vectorizer and TF-IDF in [19].

6.2.2 Combined Accuracy for MBTI Personality Types

Our proposed model performs more efficiently with binary classes as shown in Table 6.5. However, the MBTI dataset includes 16 personality types. For this, we randomly selected five rows from the dataset that correspond to the personalities of five distinct individuals. We integrated the four MBTI attributes into a unique MBTI-type personality by predicting

Papers	**Text Embedding**	**I/E**	**S/N**	**T/F**	**J/P**	**Average**
[31]	BERT	83.1%	88.5%	84.1%	78.0%	83.4%
[31]	BERT + Statistical Feature Vectors(uni+p+e)	84.6%	88.8%	84.5%	78.7%	84.2%
[19]	TF-IDF + Count Vectorizer	89.5%	89.8%	69.0%	67.6%	78.97%
Out proposed model	SBERT	96.04%	99.64%	83.62%	84.16%	90.86%

Table 6.6: Accuracy Comparison

the four MBTI attributes via binary classification with Random Forest classifiers. As demonstrated in table 6.7 our model accurately predicts four of the five MBTI personality types with the highest prediction probability. Additionally, the highest prediction probability for each MBTI characteristic indicates that our model accurately predicts 80% of the MBTI types we randomly selected. Therefore, our proposed model takes personality prediction research in a new direction by implementing a feasible model that can effectively identify the personality of an individual.

Dataset Row Number	**MBTI Type**	**Predictions**	**Maximum Prediction Probability**
2	INFJ	I	90%
		N	70%
		F	80%
		J	75%
202	ESFP	E	89%
		S	100%
		F	75%
		P	90%
302	INFP	I	89%
		N	90%
		F	81%
		P	88%
402	INTJ	I	82%
		N	98%
		F	61%
		J	84%
531	ESFP	E	88%
		S	100%
		F	88%
		P	85%

Table 6.7: Combining Binary Classes into MBTI Personality

6.3 Result Analysis for Emotion Prediction

6.3.1 Cosine Similarity

Each post is analyzed separately with emotion vectors in this approach. In the "MBTI dataset," we examined approximately 500 social media posts from each personality type. Some posts from "INTJ" personality (as well as other personality types) have very close similarity ratings for distinct emotions. We only considered the posts whose similarity score had at least a distance of 0.03 within emotions. Maximum similarity with an emotion type is considered the primary emotion of that post. Similarity scores and assignment of emotion for type "INTJ" are shown in table 6.8.

Social Media Posts	**Sad**	**Joy**	**Angry**	**Personality Assignment**
I have exactly the same problem. People whose type preferences are substantially different from mine,(usually they are S types instead of N types) just can't relate to what I am saying.	0.06554	0.029942	0.101735	Angry
I rarely have difficulty knowing how I feel or why. I seem to be having more trouble knowing how I feel over the last few years though. Perhaps this is result of being in a crappy marriage.	0.125306	0.071357	0.05458	Sad
I get huge crushes on ENXPs. I really fucking love them	0.07669	0.116276	0.086046	Joy

Table 6.8: Cosine similarity

6.3.2 Emotion Predicting Models

On "Emotion in Text", we utilized four classification models (SVM,LR, KNN and RF). SVM outperforms other models in terms of accuracy, precision, recall, and f1 score. Table 6.9 displays the accuracy for all four different classifier models. Other performance analysis metrics for SVM, LR, KNN and RF are shown in table 6.10, 6.11, 6.12 and 6.13 respectively.

Since SVM outperforms other models. As a result, the SVM confusion matrix is depicted in figure 6.2.

Model Name	Accuracy
SVM	87.05%
Logistic Regression	78.06%
KNN	76.13%
Random Forest	84.20%

Table 6.9: Accuracy Comparison

Emotion Name	Precision	Recall	F1-Score
Angry	0.89	0.91	0.90
Joy	0.86	0.85	0.86
Sad	0.85	0.84	0.84

Table 6.10: SVM Classification Report

Emotion Name	Precision	Recall	F1-Score
Angry	0.79	0.79	0.79
Joy	0.80	0.80	0.80
Sad	0.76	0.75	0.76

Table 6.11: LR Classification Report

Emotion Name	Precision	Recall	F1-Score
Angry	0.72	0.87	0.79
Joy	0.80	0.73	0.76
Sad	0.79	0.70	0.74

Table 6.12: KNN Classification Report

Emotion Name	Precision	Recall	F1-Score
Angry	0.94	0.91	0.93
Joy	0.80	0.82	0.81
Sad	0.79	0.79	0.79

Table 6.13: Random Forest Classification Report

6.3.3 Vetting Process

To make the prediction more reliable, we syncronized the two independent approaches for predicting emotions, and for this, a vetting mechanism is employed. Table 6.14 shows some of the social media posts and their vetting mechanisms for final selection.

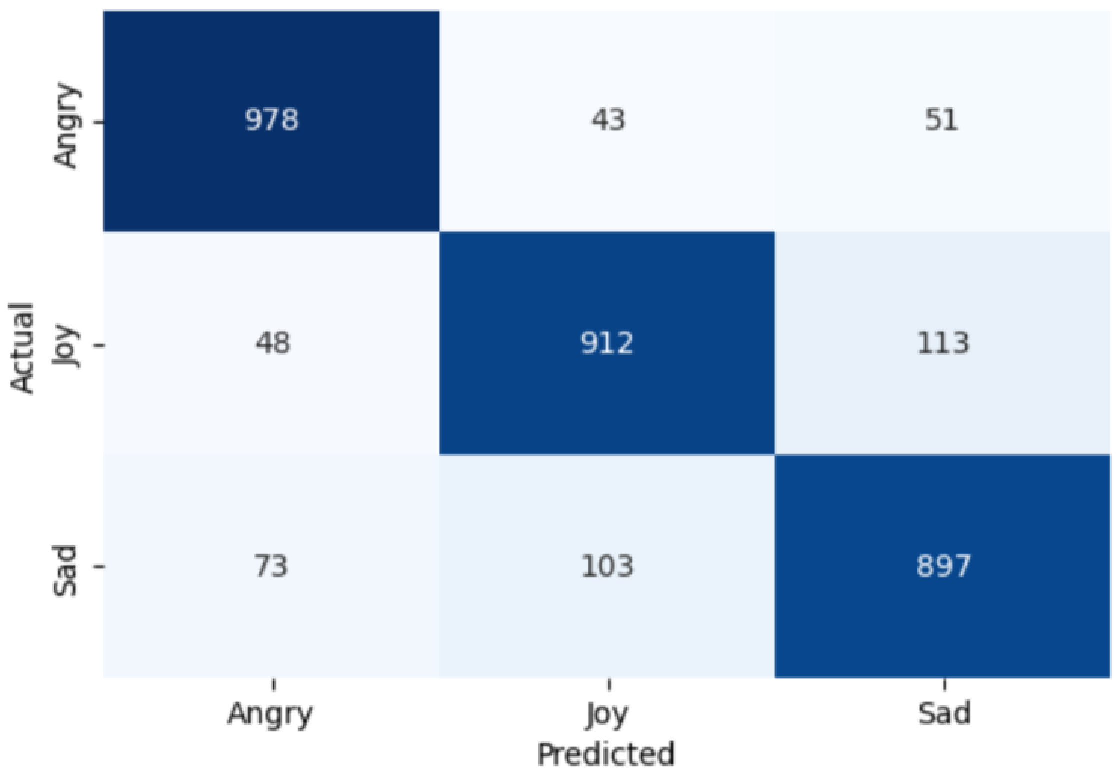


Figure 6.2: Confusion Matrix for SVM

Social Media Posts	Cosine Similarity Prediction	Models Prediction	Selection
How do I act Really irritable and pretty? obviously pissed off ! I like to solve stuff like that asap though. so I will Realize that I am angry and disappointed.	Angry	Angry	Selected
Life All the fun stuff	Joy	Joy	Selected
Felt like shocked and printing it out to spread the knowledge like see it is not a syndrome happy But for real felt like autopsy wasn't sure should I cry or laugh by now I'm ok with	Sad	Sad	Selected
Attempt to think less about things Be more decisive and confident in my decisions thoughts I can't help but play devils advocate with myself I guess this ties into number change my	Sad	Angry	Not Selected

Table 6.14: Vetting Mechanism for Final Prediction

6.3.4 Visual Representation of Posts for Different Emotions

After the vetting mechanism, we may presume that we have successfully identified various emotions in various posts from the "MBTI dataset". Here, we can see that posts that express similar emotions have been grouped together. Fig 6.3, 6.4, and 6.5 depicts the plotting of different posts containing different emotions from different viewing angle for personality type ESTJ, INTP and ISTP respectively.

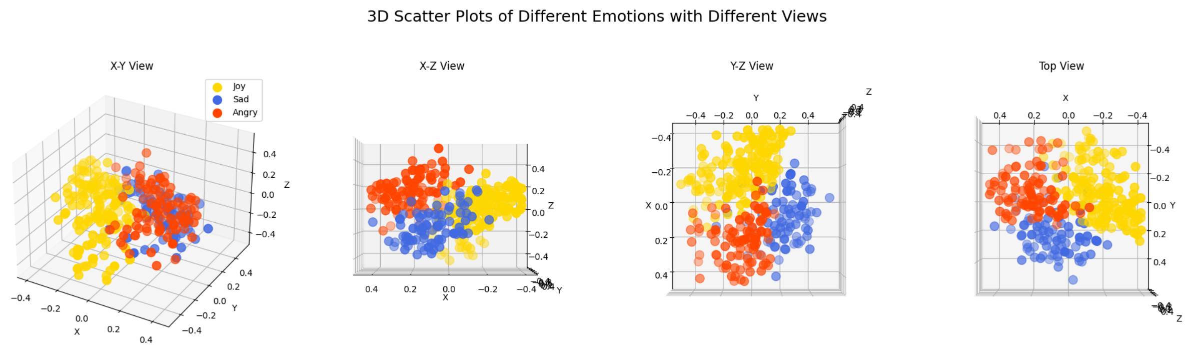


Figure 6.3: 3D Scatter Plot of Different Emotions of Personality : ESTJ

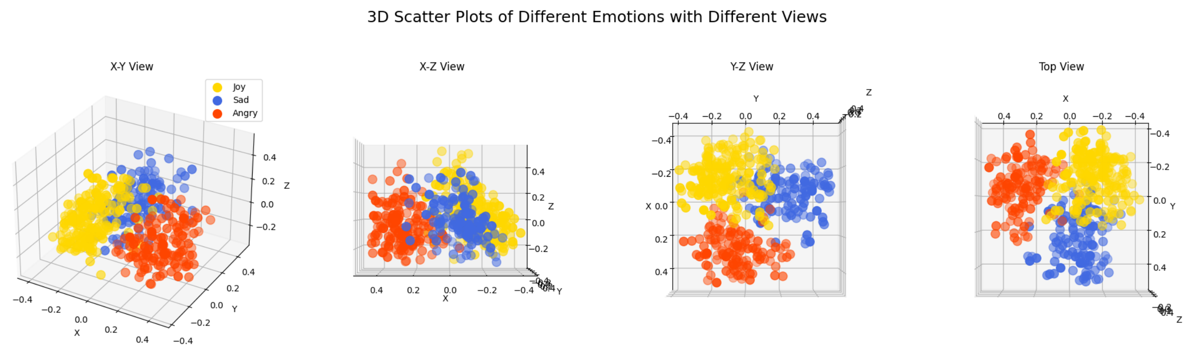


Figure 6.4: 3D Scatter Plot of Different Emotions of Personality : INTP

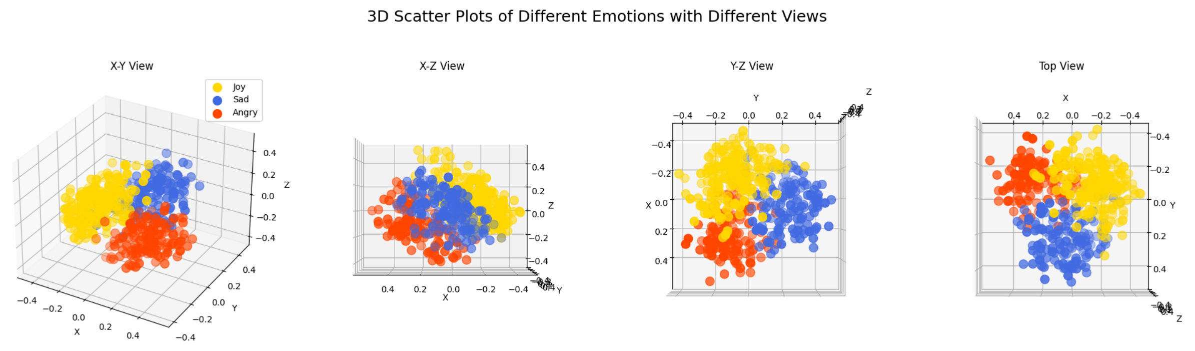


Figure 6.5: 3D Scatter Plot of Different Emotions of Personality : ISTP

6.4 Personality and Emotion Correlation

Finally, the correlation between personality and emotion is shown in fig. 6.6 where the 16 MBTI personality types are shown on the x-axis, and the percentage of posts with various emotions for each MBTI personality type is shown on the y-axis. This study uncovers notable trends in personality and emotion. We can see in fig. 6.7 and fig. 6.8 that individuals who are 'Extroverts' and who prefer to express 'Feelings' are more likely to post 'Joy' type posts than persons who are 'Introvert' and 'Thinkers'. Fig. 6.9 and 6.10 shows us people who are 'Introvert' and 'Intuitive' have more 'Sad' type posts than people who are 'Extrovert' and have strong 'Sensing' capabilities. Again, in fig. 6.11 we can see people with the 'Thinking' personality trait display 'Anger' more than people with the 'Feeling' personality trait.

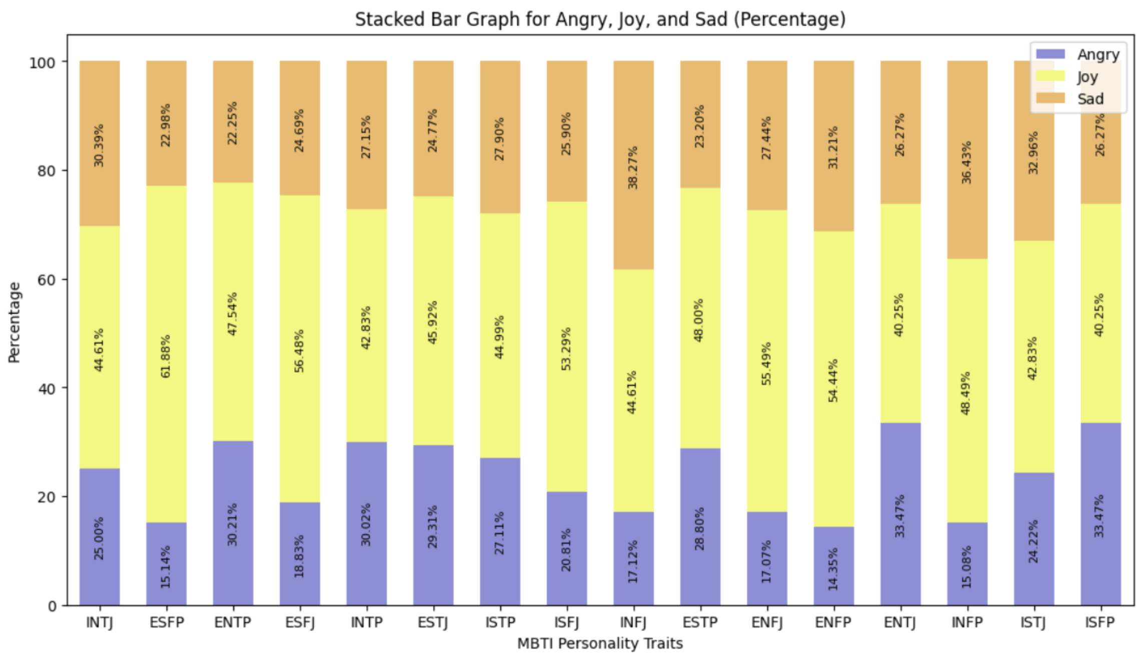


Figure 6.6: Personality and Emotion Correlation

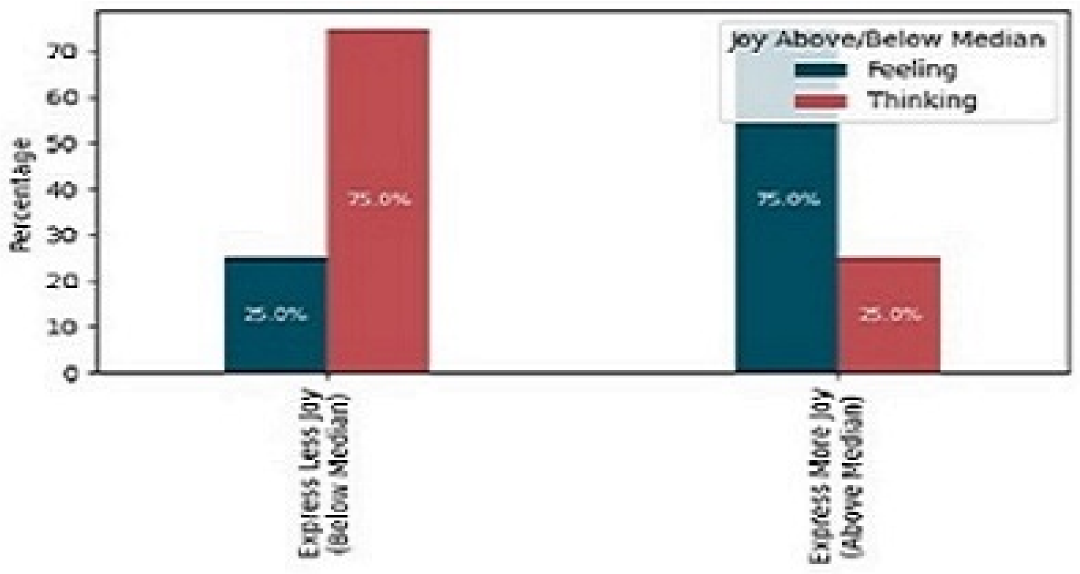


Figure 6.7: Feeling vs Thinking (Joy)

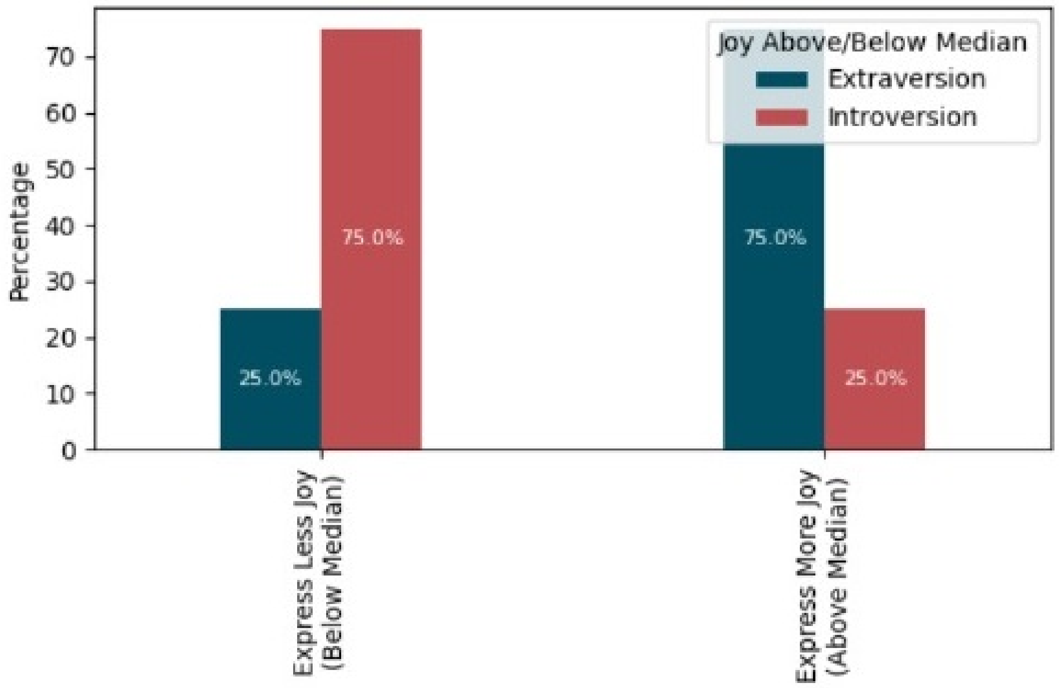


Figure 6.8: Extraversion vs Introversion (Joy)

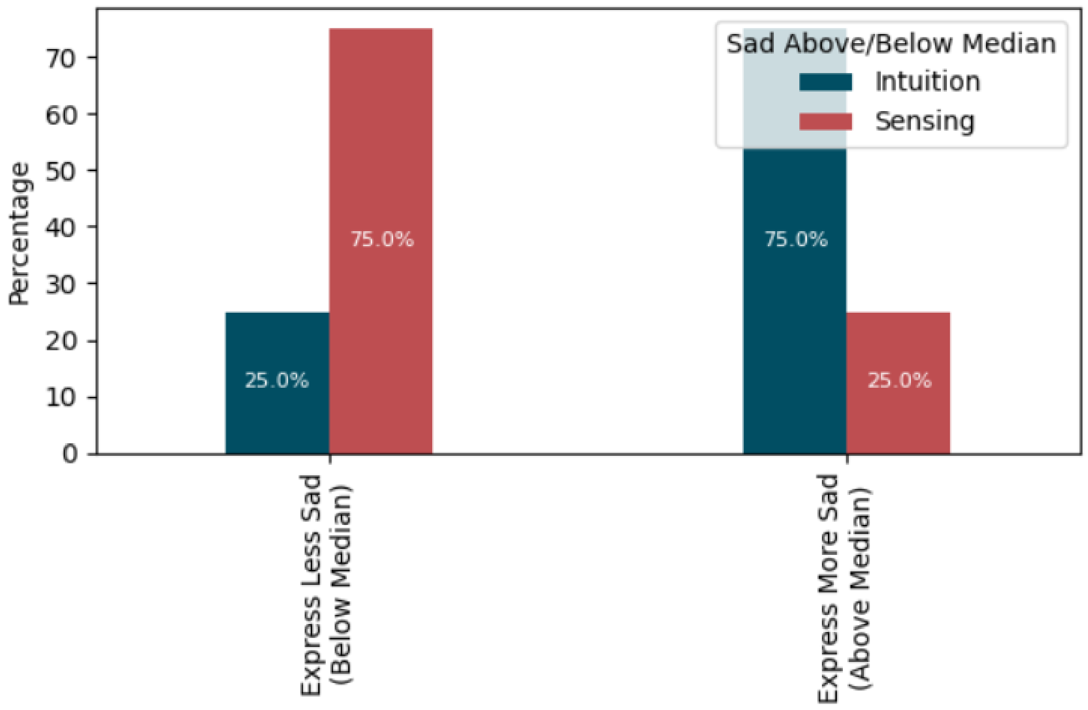


Figure 6.9: Intuition vs Sensing (Sad)

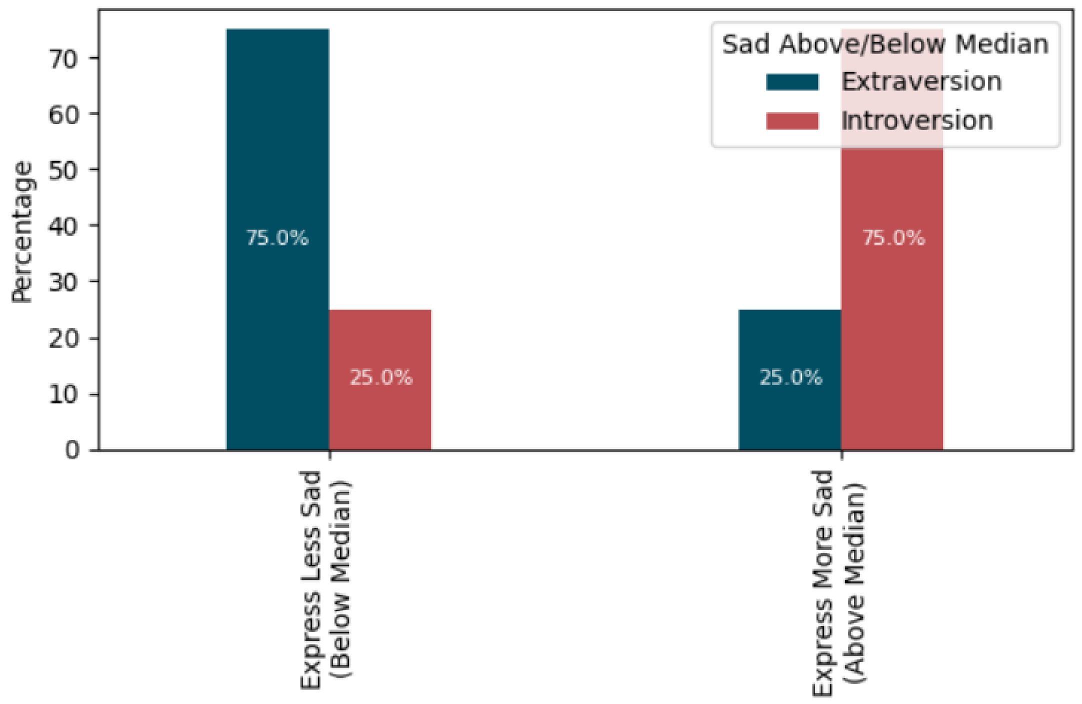


Figure 6.10: Extraversion vs Introversion (Sad)

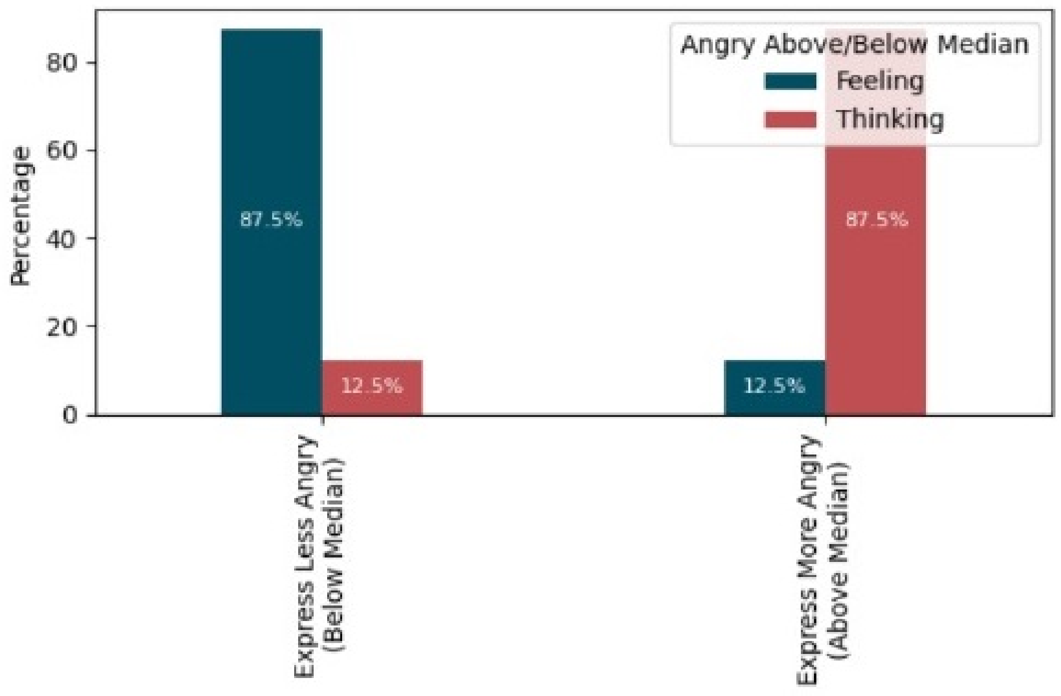


Figure 6.11: Feeling vs Thinking (Angry)

Chapter 7

Conclusion and Future Work

In this study, we examined the complex relationship between social media posts' emotional expressions and personality, as determined by the Myers-Briggs Type Indicator (MBTI). We aimed to automate the prediction of personality traits and emotions conveyed in user-generated material by utilizing state-of-the-art natural language processing techniques, specifically Sentence-BERT (SBERT) for contextual embeddings. In order to show how well SBERT combined with the Random Forest Classifier can predict MBTI personalities based on text, our study used a variety of machine learning models, such as Support Vector Machines (SVM), Logistic Regression (LR), K-Nearest Neighbors (KNN), and Random Forest (RF) Classifier.

Moreover, using preprocessed MBTI-sentence-level contextual embeddings, we presented a strong approach for calculating similarity scores associated with specified emotional categories (joy, sadness, and anger). We developed a Support Vector Machine (SVM) model using emotion-tagged datasets using a transfer learning strategy in order to anticipate emotions in MBTI social media posts. Emotion labeling accuracy was improved by using a vetting mechanism that took into account both similarity scores and model predictions.

Our work quantified the emotional distribution within each MBTI personality type, going beyond personality prediction and emotion classification. This thorough investigation offered insightful information about the complex ways people with different personality traits express and feel emotions when interacting online.

Although our study contributes significantly to our understanding of how personality and emotions interact in online environments, there are still a number of areas that warrant further investigation. First off, the accuracy and interpretability of personality and mood predictions may be improved by including more sophisticated deep learning architectures and attention mechanisms. Further research into the effects of various social media platforms and language differences on the functionality of our model may potentially lead to a more thorough comprehension of online behavior.

Furthermore, investigating the temporal dynamics of emotional changes and personality expression across time may shed light on how people's online personas change over time. We could improve our comprehension of the emotional spectrum within each personality type by including a wider variety of emotions and creating models that can capture minute emotional details.

Additionally, working with psychologists and subject matter experts could improve the data interpretability and provide a better knowledge of the psychological foundations of online communication. The automated analysis of personality and emotions in user-generated content raises a number of ethical issues that need to be carefully considered. These issues include privacy, bias and the possible misuse of these technologies in a variety of social circumstances. These approaches present viable avenues for developing the field and deciphering the intricacies of contemporary human behavior in the digital era.

References

[1] B. De Raad and B. Mlacic., “Big five factor model, theory and structure.,” in *International encyclopedia of the social behavioral sciences 2*, pp. 559–566, 2015.

[2] G. J. Boyle, “Myers-briggs type indicator (mbti): some psychometric limitations.,” in *Australian Psychologist 30.1*, pp. 71–74, 1995.

[3] D. J. Pittenger, “Measuring the mbti... and coming up short.,” in *Journal of Career Planning and Employment 54.1*, pp. 48–52, 1993.

[4] “Logistic regression — detailed overview..” https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc. Accessed: 2023-05-11.

[5] “Support vector machines(svm)—an overview..” https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/. Accessed: 2023-05-11.

[6] “Decision trees explained easily..” https://chirag-sehra.medium.com/decision-trees-explained-easily-28f23241248. Accessed: 2023-05-11.

[7] “Understanding random forest..” https://towardsdatascience.com/understanding-random-forest-58381e0602d2. Accessed: 2023-05-11.

[8] “Oversampling and undersampling..” https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf. Accessed: 2023-05-02.

[9] “Tokenization in nlp.” https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/. Accessed: 2023-05-02.

[10] "Lemmatization in nlp." https://towardsdatascience.com/stemming-vs-lemmatization-in-nlp-dea008600a0. Accessed: 2023-05-03.

[11] "Svm hyperparameter tuning." https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989. Accessed: 2023-05-03.

[12] "Baysian optimization." https://www.analyticsvidhya.com/blog/2021/05/bayesian-optimization-bayes_opt-or-hyperopt/#:~:text=Hyperparameter%2Dtuning%20is%20the%20process,we%20find%20the%20best%20accuracy. Accessed: 2023-05-04.

[13] "Ekman's six basic emotions model." https://en.wikipedia.org/wiki/Emotion_classification#Emotions_as_discrete_categories. Accessed: 2023-11-09.

[14] "Cosine similarity." https://en.wikipedia.org/wiki/Cosine_similarity#Definition. Accessed: 2023-11-09.

[15] "Transfer learning." https://en.wikipedia.org/wiki/Transfer_learning. Accessed: 2023-11-09.

[16] S. Ontoum and J. H. Chan., "Personality type based on myers-briggs type indicator with text posting style by using traditional and deep learning.," in *arXiv preprint arXiv:2201.08717*, 2022.

[17] B. Cui and C. Qi., "Survey analysis of machine learning methods for natural language processing for mbti personality type prediction.," in *Final Report Stanford University*, 2017.

[18] T. Pradhan, R. Bhansali, D. Chandnani, and A. Pangaonkar, "Analysis of personality traits using natural language processing and deep learning," in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 457–461, IEEE, 2020.

[19] S. Bharadwaj, S. Sridhar, R. Choudhary, and R. Srinath, "Persona traits identification based on myers-briggs type indicator (mbti)-a text classification approach," in *2018 international conference on advances in computing, communications and informatics (ICACCI)*, pp. 1076–1082, IEEE, 2018.

[20] K. P. Brindha, S. and S. Sukumaran., "A survey on classification techniques for text mining.," in *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS). Vol. 1. IEEE,*, 2016.

[21] R. Reisenzein and H. Weber, "Personality and emotion," *The Cambridge handbook of personality psychology*, vol. 1, pp. 54–71, 2009.

[22] Y. Li, A. Kazameini, Y. Mehta, and E. Cambria, "Multitask learning for emotion and personality detection," *arXiv preprint arXiv:2101.02346*, 2021.

[23] A. F. A. Nasir, E. S. Nee, C. S. Choong, A. S. A. Ghani, A. P. A. Majeed, A. Adam, and M. Furqan, "Text-based emotion prediction system using machine learning approach," in *IOP Conference Series: Materials Science and Engineering*, vol. 769, p. 012022, IOP Publishing, 2020.

[24] A. Agrawal and A. An, "Unsupervised emotion detection from text using semantic and syntactic relations," in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 1, pp. 346–353, IEEE, 2012.

[25] K. Park, J. S. Hong, and W. Kim, "A methodology combining cosine similarity with classifier for text classification," *Applied Artificial Intelligence*, vol. 34, no. 5, pp. 396–411, 2020.

[26] "(mbti) myers-briggs personality type dataset." `https://www.kaggle.com/datasets/datasnaek/mbti-type`. Accessed: 2023-01-07.

[27] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *EMNLP 2019*, 2019.

[28] "Emotions in text." `https://www.kaggle.com/datasets/ishantjuyal/emotions-in-text/data`. Accessed: 2023-11-07.

[29] "Nrc emotion lexicon." `https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm`. Accessed: 2023-09-13.

[30] "Understanding confusion matrix." `https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62`. Accessed: 2023-05-04.

[31] S. Majima and K. Markov., "Personality prediction from social media posts using text embedding and statistical features," in *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS). IEEE*, 2022.

Appendix A

Source Code

A.1 Basic Imports

```python
import pandas as pd
import numpy as np
# PyViz
import seaborn as sns
# %matplotlib inline
import matplotlib.pyplot as plt
# NLP
import nltk
import re
from mpl_toolkits.mplot3d import Axes3D
from textblob import TextBlob
import spacy
from nltk import word_tokenize
import sys
import spacy
from nltk import word_tokenize
import nltk
nltk.download('averaged_perceptron_tagger')
#Basics
import pandas as pd
import numpy as np
from pathlib import Path
import os
import seaborn as sns
import nltk
import re
import matplotlib.pyplot as plt
# %matplotlib inline
#SKLearn
```

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split, GridSearchCV,
   StratifiedKFold, cross_val_score
#Metrics
import sklearn
from sklearn.metrics import r2_score, mean_squared_error,
   mean_absolute_error, accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score,
    multilabel_confusion_matrix, confusion_matrix
from sklearn.metrics import classification_report
#Models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
   GradientBoostingClassifier
import xgboost as xgb
# preprocessing
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import make_pipeline
from sklearn.feature_selection import f_classif
from sklearn.feature_selection import SelectKBest
from sklearn.compose import ColumnTransformer
# class imbalance
from imblearn.pipeline import make_pipeline as imb_make_pipeline
from imblearn.under_sampling import RandomUnderSampler
# algorithms/models
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.svm import LinearSVC, SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
# model evaluation
from imblearn.metrics import classification_report_imbalanced
from imblearn.metrics import geometric_mean_score
from sklearn.metrics import average_precision_score
from sklearn.metrics import roc_auc_score
# performance check
import time
import warnings
warnings.filterwarnings("ignore")
# sparse to dense
from sklearn.base import TransformerMixin
```

```python

import pandas as pd
import numpy as np
import locale
import locale
import spacy
import string
!pip install scikit-optimize
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score,
    recall_score, f1_score, confusion_matrix
from skopt import BayesSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
from sklearn import metrics
```

Source Code A.1: Importing Basic Libraries and Models of Python.

A.2 Dataset Modifications and Oversampling

```python
class DenseTransformer(TransformerMixin):
    def fit(self, X, y=None, **fit_params):
        return self

    def transform(self, X, y=None, **fit_params):
        return X.todense()
from google.colab import drive
drive.mount('/content/drive/')
df2 = pd.read_csv("/content/drive/MyDrive/Dataset/Original_MBTI.csv"
    )
df2.head()
# to handle the class imbalance better, converting the 16 classes
    into 4 more balanced classes
df2["is_Thinking"] = df2["type"].apply(
    lambda x: 1 if x[2] == "T" else 0
)
# rearranging the dataframe columns
df_combined = df2[
    ["is_Thinking", "posts"]
]
df_combined.head()
class_counts = df_combined['is_Thinking'].value_counts()
print(class_counts)
is_ex1,is_In0 =df_combined.is_Thinking.value_counts()
df_ex1 = df_combined[df_combined['is_Thinking']==1]
df_in0 = df_combined[df_combined['is_Thinking']==0]
is_ex1,is_In0

# Oversample 1-class and concat the DataFrames of both classes
df_ex1_under = df_ex1.sample(is_ex1, replace=True)
df_test_over = pd.concat([df_ex1_under, df_in0], axis=0)
print('Random over-sampling:')
print(df_test_over.is_Thinking.value_counts())
df_combined.to_csv('MBTI789.csv', index=False)
# shuffle the rows of the DataFrame
shuffled_df = df_test_over.sample(frac=1, random_state=42)  # frac=1
    means shuffling the whole dataset
# reset the index of the shuffled DataFrame
shuffled_df = shuffled_df.reset_index(drop=True)
# save the shuffled DataFrame to a new CSV file
shuffled_df.to_csv('shuffled_dataset_sensing.csv', index=False)
```

Source Code A.2: Loading Dataset and Augmenting the Dataset.

A.3 SBERT Model

```
def getpreferredencoding(do_setlocale = True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding
!pip install -U sentence-transformers -q
np.random.seed(2022)
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-mpnet-base-v2')
```

Source Code A.3: Loading Pretrained SBERT Model.

A.4 Text Preprocessing

```
nlp = spacy.load("en_core_web_sm")
stop_words = nlp.Defaults.stop_words
print(stop_words)
punctuations = string.punctuation
print(punctuations)
#removes links and special characters using simple regex statements.
def clean_phrase(phrase):
  return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w
  +:\/\/\S+)", " ", phrase).split())
# Creating our tokenizer function
def spacy_tokenizer(sentence):
    # Creating our token object, which is used to create documents
    with linguistic annotations.
    doc = nlp(sentence)

    # print(doc)
    # print(type(doc))
    # Lemmatizing each token and converting each token into
    lowercase
    mytokens = [ word.lemma_.lower().strip() for word in doc ]

    # print(mytokens)
    # Removing stop words
    mytokens = [ word for word in mytokens if word not in stop_words
    and word not in punctuations ]

    sentence = " ".join(mytokens)
    # return preprocessed list of tokens
    return sentence
shuffled_df['tokenize'] = shuffled_df['posts'].apply(spacy_tokenizer
    )
shuffled_df.head()
```

```
shuffled_df['embeddings'] = shuffled_df['tokenize'].apply(model.
    encode)
shuffled_df.head()
```

Source Code A.4: Data Cleaning and Stopwords Removal.

A.5 Classical Machine Learning Models and Performance Evaluations

```
X = shuffled_df['embeddings'].to_list()
y = shuffled_df['is_Thinking'].to_list()
#LR with k fold
LR = LogisticRegression()
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)  #
    Using 5-fold cross-validation
accuracy_scores = cross_val_score(LR, X, y, cv=skf, scoring='
    accuracy')
precision_scores = cross_val_score(LR, X, y, cv=skf, scoring='
    precision')
recall_scores = cross_val_score(LR, X, y, cv=skf, scoring='recall')
f1_scores = cross_val_score(LR, X, y, cv=skf, scoring='f1')
print("Logistic Regression Accuracy:", accuracy_scores.mean())
print("Logistic Regression Precision:", precision_scores.mean())
print("Logistic Regression Recall:", recall_scores.mean())
print("Logistic Regression F1 score:", f1_scores.mean())
# Confusion matrix using the last fold
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, stratify=y, random_state=42)
LR.fit(X_train, y_train)
predicted = LR.predict(X_test)
cm = confusion_matrix(y_test, predicted)
print("Confusion Matrix:")
print(cm)

#KNN with k fold
KNN = KNeighborsClassifier(n_neighbors=5)
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)  #
    Using 5-fold cross-validation
accuracy_scores = cross_val_score(KNN, X, y, cv=skf, scoring='
    accuracy')
precision_scores = cross_val_score(KNN, X, y, cv=skf, scoring='
    precision')
recall_scores = cross_val_score(KNN, X, y, cv=skf, scoring='recall')
f1_scores = cross_val_score(KNN, X, y, cv=skf, scoring='f1')
print("KNN Accuracy:", accuracy_scores.mean())
print("KNN Precision:", precision_scores.mean())
print("KNN Recall:", recall_scores.mean())
print("KNN F1 score:", f1_scores.mean())
# Confusion matrix using the last fold
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, stratify=y, random_state=42)
KNN.fit(X_train, y_train)
predicted = KNN.predict(X_test)
cm = confusion_matrix(y_test, predicted)
print("Confusion Matrix:")
print(cm)

#Random Forest with K fold
random_forest = RandomForestClassifier()
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)  #
    Using 5-fold cross-validation
accuracy_scores = cross_val_score(random_forest, X, y, cv=skf,
    scoring='accuracy')
precision_scores = cross_val_score(random_forest, X, y, cv=skf,
    scoring='precision')
recall_scores = cross_val_score(random_forest, X, y, cv=skf, scoring
    ='recall')
f1_scores = cross_val_score(random_forest, X, y, cv=skf, scoring='f1
    ')
print("Random Forest Accuracy:", accuracy_scores.mean())
print("Random Forest Precision:", precision_scores.mean())
print("Random Forest Recall:", recall_scores.mean())
print("Random Forest F1 score:", f1_scores.mean())
# Confusion matrix using the last fold
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, stratify=y, random_state=42)
random_forest.fit(X_train, y_train)
predicted = random_forest.predict(X_test)
cm = confusion_matrix(y_test, predicted)
print("Confusion Matrix:")
print(cm)

#Decision Tree with K fold
decision_tree = DecisionTreeClassifier()
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)  #
    Using 5-fold cross-validation
accuracy_scores = cross_val_score(decision_tree, X, y, cv=skf,
    scoring='accuracy')
precision_scores = cross_val_score(decision_tree, X, y, cv=skf,
    scoring='precision')
recall_scores = cross_val_score(decision_tree, X, y, cv=skf, scoring
    ='recall')
f1_scores = cross_val_score(decision_tree, X, y, cv=skf, scoring='f1
    ')
print("Decision Tree Accuracy:", accuracy_scores.mean())
print("Decision Tree Precision:", precision_scores.mean())
print("Decision Tree Recall:", recall_scores.mean())
print("Decision Tree F1 score:", f1_scores.mean())
# Confusion matrix using the last fold
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, stratify=y, random_state=42)
decision_tree.fit(X_train, y_train)
predicted = decision_tree.predict(X_test)
cm = confusion_matrix(y_test, predicted)
print("Confusion Matrix:")
print(cm)

#SVM with k fold
SVM = SVC()
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)  #
    Using 5-fold cross-validation
accuracy_scores = cross_val_score(SVM, X, y, cv=skf, scoring='
    accuracy')
precision_scores = cross_val_score(SVM, X, y, cv=skf, scoring='
    precision')
recall_scores = cross_val_score(SVM, X, y, cv=skf, scoring='recall')
f1_scores = cross_val_score(SVM, X, y, cv=skf, scoring='f1')
print("SVM Accuracy:", accuracy_scores.mean())
print("SVM Precision:", precision_scores.mean())
print("SVM Recall:", recall_scores.mean())
print("SVM F1 score:", f1_scores.mean())
# Confusion matrix using the last fold
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, stratify=y, random_state=42)
SVM.fit(X_train, y_train)
predicted = SVM.predict(X_test)
cm = confusion_matrix(y_test, predicted)
print("Confusion Matrix:")
print(cm)
```

Source Code A.5: Performing ML models and Performance Evaluation.

A.6 Cosine Similarity of MBTI with emotions

```
df2 = pd.read_csv("/content/drive/MyDrive/Dataset/Original_MBTI.csv"
    )
df2.head()
num_rows = df2.shape[0]
print("Number of rows in the DataFrame:", num_rows)
unique_types = df2['type'].unique()
```

```
dfs_by_type = {}
# Split the original DataFrame into separate DataFrames based on
   unique types
for type_value in unique_types:
    type_df = df2[df2['type'] == type_value].copy()
    dfs_by_type[type_value] = type_df
INTJ_df = dfs_by_type['INTJ']

csv_file_path = '/content/drive/MyDrive/Dataset/INTJ_New.csv'
INTJ_df.to_csv(csv_file_path, index=False)
print(f"Filtered data with 'E' column values equal to 1 saved to '{
   csv_file_path}'")
num_rows = INTJ_df.shape[0]
print("Number of rows in the DataFrame:", num_rows)
num_rows = INTJ_df.shape[0]
print("Number of rows in the DataFrame:", num_rows)

start_index = 100
end_index = 102
INTJ_df = INTJ_df.iloc[start_index:end_index+1]
num_rows = INTJ_df.shape[0]
print("Number of rows in the specific range DataFrame:", num_rows)

new_rows = []
for _, row in INTJ_df.iterrows():
    sentences = row['posts'].split('|||')
    for sentence in sentences:
        new_rows.append({'Type': row['type'], 'Posts': sentence.
   strip()})
INTJ_df = pd.DataFrame(new_rows)

filtered_df = INTJ_df[INTJ_df['Posts'].apply(lambda x: len(x.split()
   ) > 2)]
INTJ_df = filtered_df[['Posts', 'Type']]

num_rows = INTJ_df.shape[0]
print("Number of rows in the DataFrame:", num_rows)


csv_file_path = '/content/drive/MyDrive/Dataset/
   separated_sentences_INTJ_4.csv'
INTJ_df.to_csv(csv_file_path, index=False)
nltk.download('stopwords')
Stop = stopwords.words('english')
personality_types = [x.lower() for x in INTJ_df["Type"].unique()]
stop_words = ["hmm", "b", "c", "ahh"]
import re
```

```python
def cleaner(text):
    sw = set(Stop)
    regex = re.compile("[^a-zA-Z ]")
    regex2 = re.compile("[   \+,'   ]")
    post = re.sub(r'''https?:\/\/[^| ]+''', '', text, flags=re.MULTILINE)
    post = re.sub(r'''[0-9]+''', '', post, flags=re.MULTILINE)
    post = post.replace('|||', " ")
    puncs1 = ['@', '#', '$', '%', '^', '~', '&', '*', '(', ')', '-', '_', '+', '=', '{', '}', '[', ']', '|',
              '\\', '"', "'", ';', ':', '<', '>', '/', ',', '.', '?', '!', '\n']
    for punc in puncs1:
        post = post.replace(punc, ' ')
    text = re.sub('\s+', ' ', post).strip()
    text = regex2.sub('', text)
    return text
INTJ_df["posts_cleaned"] = INTJ_df["Posts"].apply(cleaner)
INTJ_df.head()
INTJ_df["Type"].value_counts()

import locale
print(locale.getpreferredencoding())
import locale
def getpreferredencoding(do_setlocale = True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding

!pip install -U sentence-transformers -q
import pandas as pd
import numpy as np
np.random.seed(2022)
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-mpnet-base-v2')

import spacy
import string
nlp = spacy.load("en_core_web_sm")
stop_words = nlp.Defaults.stop_words
print(stop_words)
punctuations = string.punctuation
print(punctuations)

INTJ_df['embeddings'] = INTJ_df['posts_cleaned'].apply(model.encode)


posts = INTJ_df['posts_cleaned'].tolist()
```

```python
embeddings = INTJ_df['embeddings'].tolist()
angry_words = [
    "rage", "frustration", "fury", "outrage", "annoyance", "hatred",
    "resentment",
    "bitterness", "wrath", "agitation", "enraged", "irritated", "
    indignant",
    "fuming", "incensed", "livid", "outraged", "irate", "angry", "
    infuriated", "furious",
    "exasperated", "hateful", "provoked", "resentful", "wrathful", "
    infuriation",
    "disgusted", "enragement", "embittered", "aggravated", "acerbic"
    , "choleric", "acerbity",
    "acerbate", "cross", "antagonistic", "temper", "maddened", "
    clenched", "clenching",
    "pissed", "boiling", "lividity", "enmity", "distemper", "
    incensement", "aggrieved",
    "crossness", "displeased", "soreness", "crabby", "sulkiness", "
    acidulous", "huffish",
    "huffiness", "heated", "heatedly", "acerbate", "snappish", "
    fretful", "short-tempered",
    "ill-tempered", "ill-humored", "discontent", "ill-disposed", "
    short-fused", "spiteful",
    "grouchy", "grumpy", "surly", "cantankerous", "churlish", "
    ornery", "peevish", "testy",
    "touchy", "irascible", "cranky", "tetchy", "quick-tempered", "
    bad-tempered", "crotchety",
    "curmudgeonly", "crabbed", "sour", "vinegarish", "raspy", "
    snarky", "dislike", "abhorrence",
    "repulsion", "aversion", "revulsion", "nausea", "loathing", "
    detest", "abominate", "antipathy",
    "displeasure", "disdain", "contempt", "abhor", "repel", "sicken"
    , "scorn", "disapproval",
    "abomination", "disgusting", "hate", "disgustful", "revolting",
    "abhorrent", "repugnant",
    "appalled", "odious", "offensive", "vile", "malodorous", "rancid
    ", "foul", "putrid", "rank",
    "nasty", "gross", "rotten", "stench", "stink", "filthy", "
    gruesome", "hateful", "sickening",
    "repulsive", "loathsome", "terrible"
]

joy_words = [
    "happiness", "delight", "euphoria", "joyful", "excitement", "
    pleasure", "contentment",
    "bliss", "ecstasy", "jubilation", "elation", "glee", "merriment"
    , "exhilaration", "gratitude",
    "enjoyment", "cheerful", "upbeat", "thrilled", "satisfied", "
    festive", "joyous", "radiant",
     "elated", "cheerful", "jovial", "exuberant", "buoyant", "
    ecstatic",
     "hilarious", "laughter", "hilarity", "jocund", "jolly", "
    jubilant", "overjoyed", "blithe",
     "mirth", "mirthful", "rejoice", "rejoicing", "rhapsody", "
    jollity", "jocundity", "jocularity",
     "gleeful", "gay", "gayety", "gayness", "glad", "gladden", "
    gladsome", "heartwarming", "cheery",
     "merry", "mirthful", "festal", "festive", "paradise", "
    paradisiacal", "paradisiac", "sunny",
     "sunshine", "delighted", "delightful", "happily", "happiness", "
    happiest", "beaming",
     "sparkling", "charmed", "overjoyed", "thrilling", "jubilee", "
    joviality", "ebullient",
     "animated", "rapturous", "enraptured", "enthusiasm", "ecstasy",
    "ecstatically", "euphoric",
     "contented", "exultant", "exultation", "blissful", "blissfully",
     "cheeriness", "cheerily",
     "jocular", "felicity", "felicitous", "joie de vivre", "merriness
    ", "satisfaction", "lighthearted",
     "lively", "optimistic", "radiance", "radiant"
]

sad_words = [
     "sorrow", "grief", "melancholy", "despair", "unhappy", "
    heartbroken", "mournful", "tearful",
     "depressed", "dejected", "blue", "downcast", "woeful", "somber",
     "gloomy", "dismal", "forlorn",
     "disheartened", "troubled", "distressed", "bereaved", "grieved",
     "miserable", "mournful", "wistful",
     "crestfallen", "saddened", "regretful", "upset",
     "desolation", "anguish", "angst", "lament", "lamentation", "
    disconsolate", "heartrending",
     "brokenhearted", "weeping", "crying", "melancholic", "
    melancholia", "bitterness", "numb",
     "painful", "pain", "suffering", "suffer", "hurt", "cruel", "
    lonely", "solitude", "loneliness",
     "abandoned", "desolate", "desperation", "disappointed", "
    disappointment", "dismay", "downhearted",
     "downheartedness", "grieving", "heartache", "heartbreak", "
    heavyhearted", "mourner", "mourning",
     "pensive", "sorrowful", "sulking", "tears", "tragic", "tragedy",
     "wailing", "woe", "woefulness",
     "worry", "agonized", "angry", "displeased", "resentful", "
    frustrated", "sore", "tormented",
     "tormenting", "torment", "afflicted", "ailing", "bemoaning", "
    bewail", "complain", "complaining",
     "complaint", "dirge", "doleful", "dolor", "dolorous", "downer",
    "downfall", "downside"
 ]

 def get_emotion_score(embedding, emotion_words):
     similarity_scores = [np.dot(embedding, model.encode(word)) for
    word in emotion_words]
     return np.mean(similarity_scores)

 emotion_labels = []
 sad_scores = []
 joy_scores = []
 angry_scores = []

 for embedding in embeddings:
     sad_score = get_emotion_score(embedding, sad_words)
     joy_score = get_emotion_score(embedding, joy_words)
     angry_score = get_emotion_score(embedding, angry_words)

     max_score = max(sad_score, joy_score, angry_score)
     if max_score == sad_score:
         emotion_labels.append('Sad')
     elif max_score == joy_score:
         emotion_labels.append('Joy')
     else:
         emotion_labels.append('Angry')

     sad_scores.append(sad_score)
     joy_scores.append(joy_score)
     angry_scores.append(angry_score)

 INTJ_df['emotion_label'] = emotion_labels
 INTJ_df['sad_score'] = sad_scores
 INTJ_df['joy_score'] = joy_scores
 INTJ_df['angry_score'] = angry_scores


 INTJ_df["emotion_label"].value_counts()

 threshold = 0.03
 highest_emotion = INTJ_df[['sad_score', 'joy_score', 'angry_score'
    ]].max(axis=1)
 second_highest_emotion = INTJ_df[['sad_score', 'joy_score', '
    angry_score']].apply(lambda row: sorted(row)[-2], axis=1)
 filtered_df = INTJ_df[(highest_emotion - second_highest_emotion) >
    threshold]
```

```

filtered_df["emotion_label"].value_counts()

label_encoder = LabelEncoder()
filtered_df['Encoded_Category'] = label_encoder.fit_transform(
    filtered_df['emotion_label'])
filtered_df.head()
filtered_df = filtered_df.sample(frac=1, random_state=42).
    reset_index(drop=True)

filtered_df["Encoded_Category"].value_counts()

csv_file_path = '/content/drive/MyDrive/Dataset/
    separated_sentences_INTJ_444444_output.csv'
filtered_df.to_csv(csv_file_path, index=False)
```

Source Code A.6: Cosine Similarity of MBTI with emotions

A.7 Training the SVM model for Predicting Emotions

```
df = pd.read_csv("/content/drive/MyDrive/Dataset/Emotion1.csv")
df.head()

unique_labels = df['label'].value_counts()

for label, count in unique_labels.items():
    print(f"Label: {label}, Count: {count}")

filtered_df = df[~df['label'].isin([2, 4, 5])]

unique_labels = filtered_df['label'].value_counts()

for label, count in unique_labels.items():
    print(f"Label: {label}, Count: {count}")

conditions = [
    (filtered_df['label'] == 0),
    (filtered_df['label'] == 1),
    (filtered_df['label'] == 3)
]

choices = ['Sad', 'Joy', 'Anger']

filtered_df['emotion'] = np.select(conditions, choices, default='
    Unknown')
filtered_df["text_cleaned"] = filtered_df["text"].apply(cleaner)
filtered_df.head()
```

```python
filtered_df["emotion"].value_counts()
filtered_df = filtered_df.drop(columns=[’label’,’text’])

from imblearn.over_sampling import RandomOverSampler
import pandas as pd

# Count number of each type
class_counts = filtered_df[’emotion’].value_counts()

# Separate into features and target
X = filtered_df.drop(’emotion’, axis=1)
y = filtered_df[’emotion’]

# Oversample minority classes
oversampler = RandomOverSampler(random_state=0)
X_resampled, y_resampled = oversampler.fit_resample(X, y)

df_resampled = pd.concat((pd.DataFrame(X_resampled, columns=X.columns), pd.DataFrame(y_resampled, columns=[’emotion’])), axis=1)

df_resampled.to_csv(’emotion_balanced.csv’, index=False)

class_counts_resampled = df_resampled[’emotion’].value_counts()
min_class_count = class_counts_resampled.min()
df_resampled_balanced = pd.DataFrame(columns=[’cleaned_text’, ’emotion’])

for emotion_class in class_counts_resampled.index:
    class_data = df_resampled[df_resampled[’emotion’] == emotion_class]
    sampled_data = class_data.sample(n=min_class_count, random_state=0)
    df_resampled_balanced = pd.concat([df_resampled_balanced, sampled_data])

unique_labels = df_resampled_balanced[’emotion’].value_counts()

for label, count in unique_labels.items():
    print(f"Label: {label}, Count: {count}")


df_resampled_balanced = df_resampled_balanced.drop(df_resampled_balanced.columns[0], axis=1)
df_resampled_balanced = shuffle(df_resampled_balanced, random_state=0)

```

```
df_resampled_balanced['embeddings'] = df_resampled_balanced['
    text_cleaned'].apply(model.encode)
df_resampled_balanced.head()

label_encoder = LabelEncoder()
df_resampled_balanced['Encoded_Category'] = label_encoder.
    fit_transform(df_resampled_balanced['emotion'])
df_resampled_balanced.head()


#SVM with k fold
!pip install scikit-optimize
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score,
    recall_score, f1_score, confusion_matrix
from skopt import BayesSearchCV
!pip install scikit-optimize
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score,
    recall_score, f1_score, confusion_matrix
from skopt import BayesSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
from sklearn import metrics

X = df_resampled_balanced['embeddings'].to_list()
y = df_resampled_balanced['Encoded_Category'].to_list()



SVM = SVC()
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)  #
    Using 5-fold cross-validation

accuracy_scores = cross_val_score(SVM, X, y, cv=skf, scoring='
    accuracy')
```

```

print("SVM Accuracy:", accuracy_scores.mean())
# Confusion matrix using the last fold
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, stratify=y, random_state=42)
SVM.fit(X_train, y_train)
predicted = SVM.predict(X_test)
cm = confusion_matrix(y_test, predicted)
print("Confusion Matrix:")
print(cm)
```

Source Code A.7: Training the SVM model for Predicting Emotions

A.8 Correlation between Personality and Emotions

```
df_INTJ = pd.read_csv("/content/drive/MyDrive/Dataset/Personality/
    separated_sentences_INTJ_final_output.csv")
df_INTJ.head()

df_INTJ = df_INTJ.drop(columns=['Posts','Type','embeddings','
    sad_score','joy_score','angry_score'])
df11=df_INTJ
df11.head()
df11['embeddings'] = df11['posts_cleaned'].apply(model.encode)
df11.head()
X_new_df_INTJ = df11['embeddings'].to_list()
predictions = SVM.predict(X_new_df_INTJ)
df11['Predicted_Category'] = predictions

true_values = df11['Encoded_Category'].tolist()
predicted_values = df11['Predicted_Category'].tolist()

# Calculate accuracy
accuracy = accuracy_score(true_values, predicted_values)

print("Accuracy:", accuracy)

matching_df_INTJ = df_INTJ[df_INTJ['Encoded_Category'] == df_INTJ['
    Predicted_Category']]

import matplotlib.pyplot as plt

# Assuming you have the value counts stored in a dictionary
class_labels = {0: "Angry", 1: "Joy", 2: "Sad"}
value_counts = {0: 116, 1: 207, 2: 141, }

total_samples = sum(value_counts.values())
```

```
percentage_values = [(count / total_samples) * 100 for count in value_counts.values()]

class_names = [class_labels[class_num] for class_num in value_counts.keys()]

plt.pie(percentage_values, labels=class_names, autopct='%1.1f%%', startangle=140)
plt.axis('equal')
plt.title('Personality : INTJ', y=1.05)
plt.show()


num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
matching_df_INTJ['cluster'] = kmeans.fit_predict(list(matching_df_INTJ['embeddings']))

pca = PCA(n_components=3, random_state=42)
embeddings_3d = pca.fit_transform(list(matching_df_INTJ['embeddings']))
matching_df_INTJ['X'] = embeddings_3d[:, 0]
matching_df_INTJ['Y'] = embeddings_3d[:, 1]
matching_df_INTJ['Z'] = embeddings_3d[:, 2]

colors = {'Joy': '#FFD700', 'Sad': '#4169E1', 'Angry': '#FF4500'}

fig = plt.figure(figsize=(20, 5))

# X-Y view
ax1 = fig.add_subplot(141, projection='3d')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_zlabel('Z')
ax1.set_title('X-Y View')
for emotion_label, color in colors.items():
    data = matching_df_INTJ[matching_df_INTJ['emotion_label'] == emotion_label]
    ax1.scatter(data['X'], data['Y'], data['Z'], c=color, label=emotion_label, s=100)
ax1.legend()

# X-Z view
ax2 = fig.add_subplot(142, projection='3d')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_zlabel('Z')
```

```python
ax2.set_title('X-Z View')
for emotion_label, color in colors.items():
    data = matching_df_INTJ[matching_df_INTJ['emotion_label'] == emotion_label]
    ax2.scatter(data['X'], data['Y'], data['Z'], c=color, label=emotion_label, s=100)
ax2.view_init(elev=0, azim=90)

# Y-Z view
ax3 = fig.add_subplot(143, projection='3d')
ax3.set_xlabel('X')
ax3.set_ylabel('Y')
ax3.set_zlabel('Z')
ax3.set_title('Y-Z View')
for emotion_label, color in colors.items():
    data = matching_df_INTJ[matching_df_INTJ['emotion_label'] == emotion_label]
    ax3.scatter(data['X'], data['Y'], data['Z'], c=color, label=emotion_label, s=100)
ax3.view_init(elev=90, azim=0)

# Top view
ax4 = fig.add_subplot(144, projection='3d')
ax4.set_xlabel('X')
ax4.set_ylabel('Y')
ax4.set_zlabel('Z')
ax4.set_title('Top View')
for emotion_label, color in colors.items():
    data = matching_df_INTJ[matching_df_INTJ['emotion_label'] == emotion_label]
    ax4.scatter(data['X'], data['Y'], data['Z'], c=color, label=emotion_label, s=100)
ax4.view_init(elev=90, azim=90)

plt.suptitle('3D Scatter Plots of Different Emotions with Different Views', fontsize=18, y=1.1)

plt.tight_layout()
plt.show()
```

Source Code A.8: Correlation between Personality and Emotions