Machine Learning Project Submission

Avraham Rahimov 214423147
Avichay Mezin 211780267

July 2024

1 Data Description

The dataset consists of 416,809 entries, each with three columns: an index, a text segment, and a sentiment label. Specifically, it contains English Twitter messages annotated with six fundamental emotions: anger, fear, joy, love, sadness, and surprise. The emotions are represented by numerical labels: sadness (0), joy (1), love (2), anger (3), fear (4), and surprise (5). This dataset is useful for sentiment analysis, emotion classification, and text-mining tasks. It provides a comprehensive view of emotional expressions in short social media texts. The dataset has two main columns: 'text' (object type) and 'label' (int64 type), with no missing values.

2 Our questions about the project

- Models cannot understand text, so how can we convert the text into numeric data?
- The data is imbalanced, what can we do to fix that?
- What analyses can be done for the data?
- Which embedding technique should we use?
- What is the correlation between the main features and the label?
- How should we split the data?
- The tuning process takes a long time, what can we do to make it more efficient?
- How much accuracy would the dummy model give?
- Why do the XGBoost and Random Forest models give close accuracies?
- Why does the KNN model give a weak accuracy with the GloVe embedding?
- What are the key hyperparameters for each model, and how do they impact performance?
- Why the CountVectorizer embedding give better results than the GloVe embedding?
- What are the potential overfitting issues with complex models, and how can we fix them?
- How do different evaluation metrics (e.g., accuracy, precision, recall, F1-score) affect our understanding of model performance?
- How can we ensure our model works well on new, unseen data?
- What are the advantages and disadvantages of using deep learning models (like CNN) compared to traditional machine learning models for text classification?
- How do various regularization techniques (e.g., dropout, L1/L2 regularization) impact model training and performance?

3 Preprocessing and Exploratory Data Analysis (EDA)

3.1 Data Imbalance

The initial exploration of the dataset revealed a significant imbalance in the distribution of labels. As shown in Figure 1, certain classes such as joy (1) and sadness (0) are more frequent than others like surprise (5). This imbalance can bias the models later towards the more frequent classes, leading to poor generalization and lower accuracy for the less frequent classes.

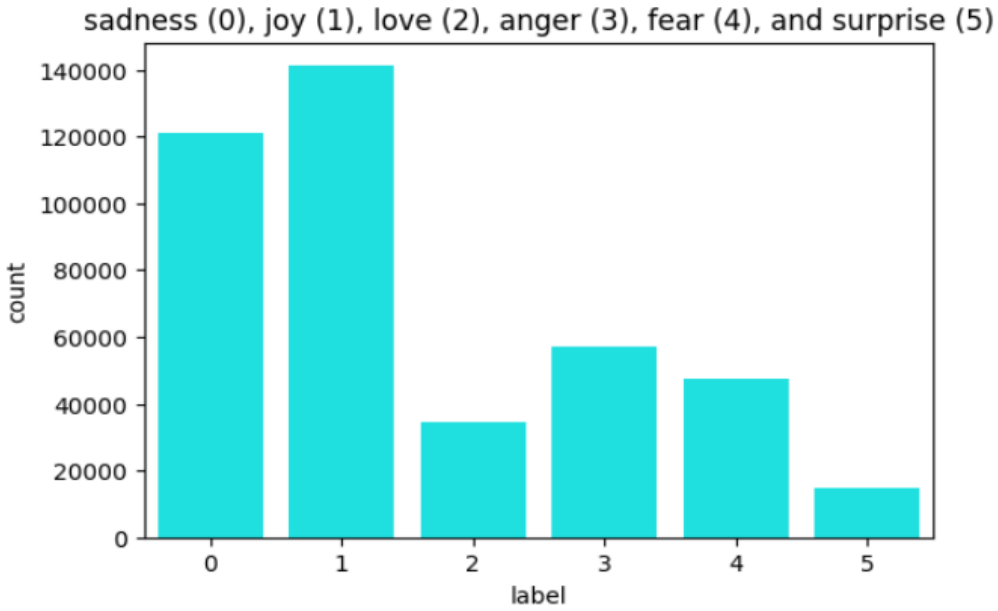

Figure 1: Distribution of Labels in the Dataset before equalization

To solve this issue, we took the class with the smallest number of samples (class number 5) and divide it by 2 and put in all the rest lables the same amount of samples. As shown in Fig 2. We take the smallest and divide it by 2, for simplify our models later and not give them too much data and still make them give a good results.

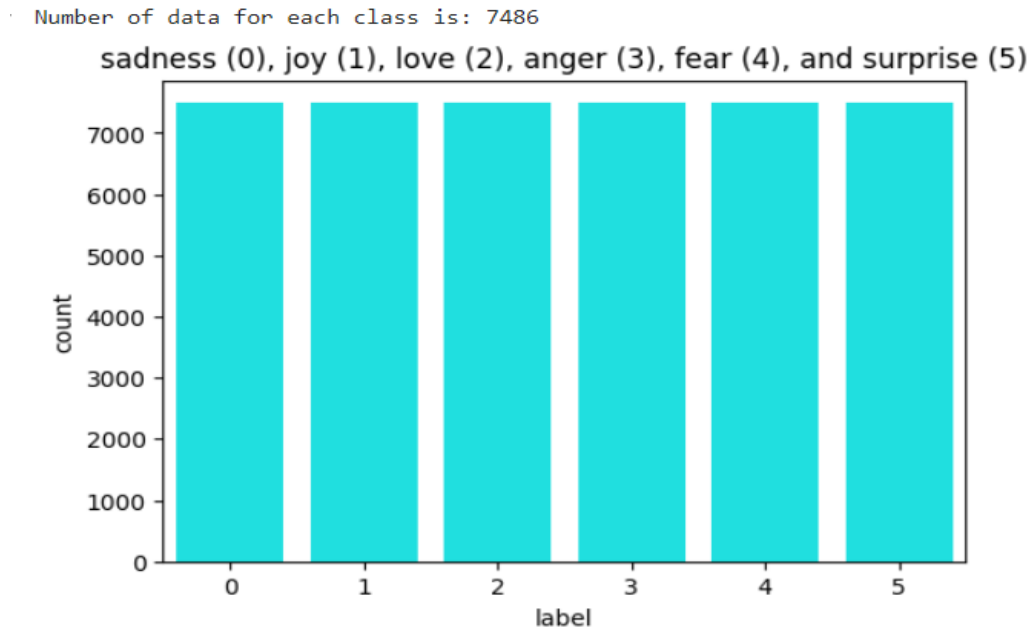

Figure 2: Distribution of Labels in the Dataset after equalization, as we can see, in every class there exist 7486 samples and now the dataset is well-balanced in the number of samples in every class.

The Dataset from now will called 'df downsampled'.

3.2 Text Length Distribution

The distribution of text lengths across different classes is shown in Figure 3. As we can see, most text samples are relatively short, with lengths between 20 and 120 words. Each class has a similar distribution pattern, although there are slight variations in the average text length among different classes.

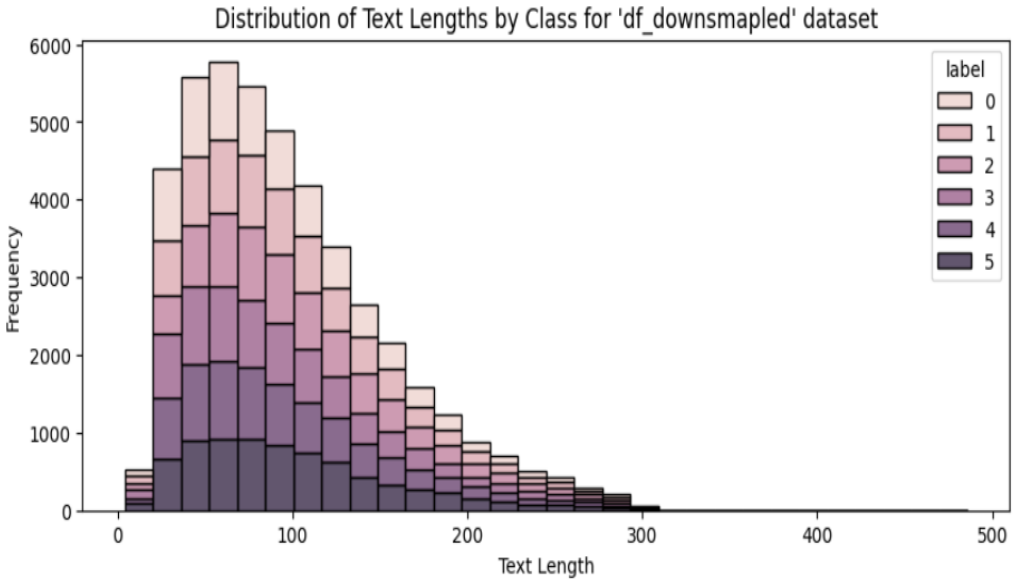

Figure 3: Distribution of Text Lengths by Class in the Downsampled Dataset. We can observe that across various text lengths, the rectangles representing different classes are of similar sizes. This is a positive indication as it suggests that the data is well-balanced across the classes, because it means that in every class there is approximately the same number of text length.

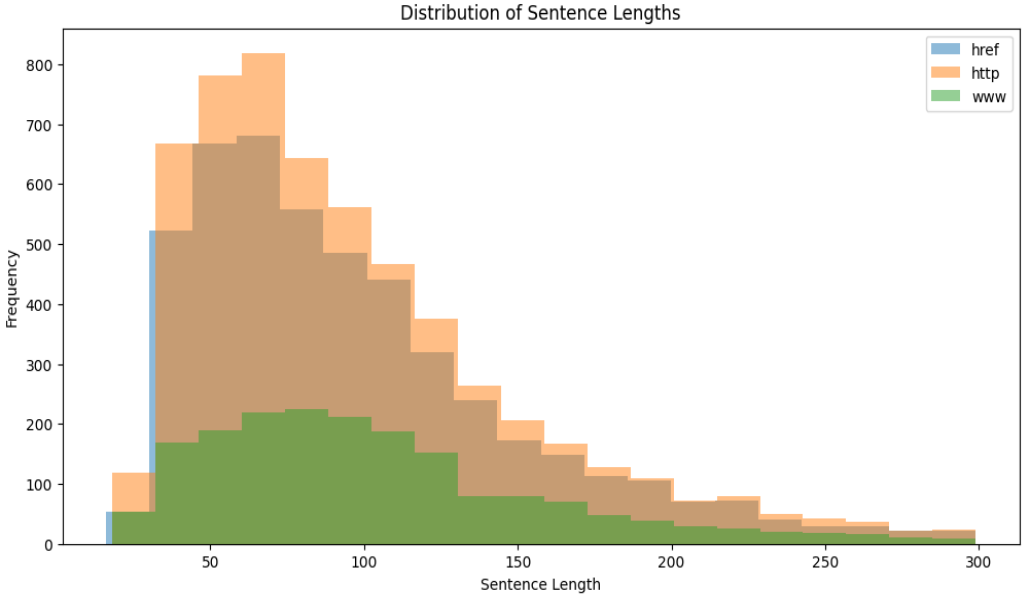

Figure 4: In this plot we can see the distribution of sentnces that include the words 'href', 'http', 'www'. We can notice that the sentnces with this words are distribute same as the overall distributions of texts length.

As shown in Fig 4, the distribution of the sentnces with this 3 words is approximately as the overall distribution. Later we have seen in the project that this 3 words ('href', 'http', 'www') are the most frequent words so they are very effect the model and also the overall distribution. But, this words are not important for the models, because they not give information about the emotion of the person that wrote the twitter post. This words are words that came from smilies, images and that's why this include 'http' and the rest, because this is links to somewhere else in the internet. As we said, this words are not important and that's why we will see later that we omit them from the dataset using stop words technique (a technique used to remove text that don't contribute to the efficiency of the models). As shown in Fig 7, we omit those words by adding them to the stop words list.

3.3 Most frequent words in every class

Here, we make plot for every class that show the most frequent words in that class:

Figure 5: The most frequent words in every class. When the word is larger in the plot, it means that the frequent is higher and this word appears a lot. And when the word is smaller in the plot, it means that it's not so frequent and not appear to much.
(top left) - label 0. (top right) - label 1. (under top left) - label 2. And so on..

We can infer from Fig 5 that in every class the word feel or feeling is very frequent (because as we said, when the word is bigger in the plot, it means that the frequency of that word is higher). This indicate that people in twitter, when they tweet, very often they tweets that they feel something.

3.4 Top 10 frequent words in each class

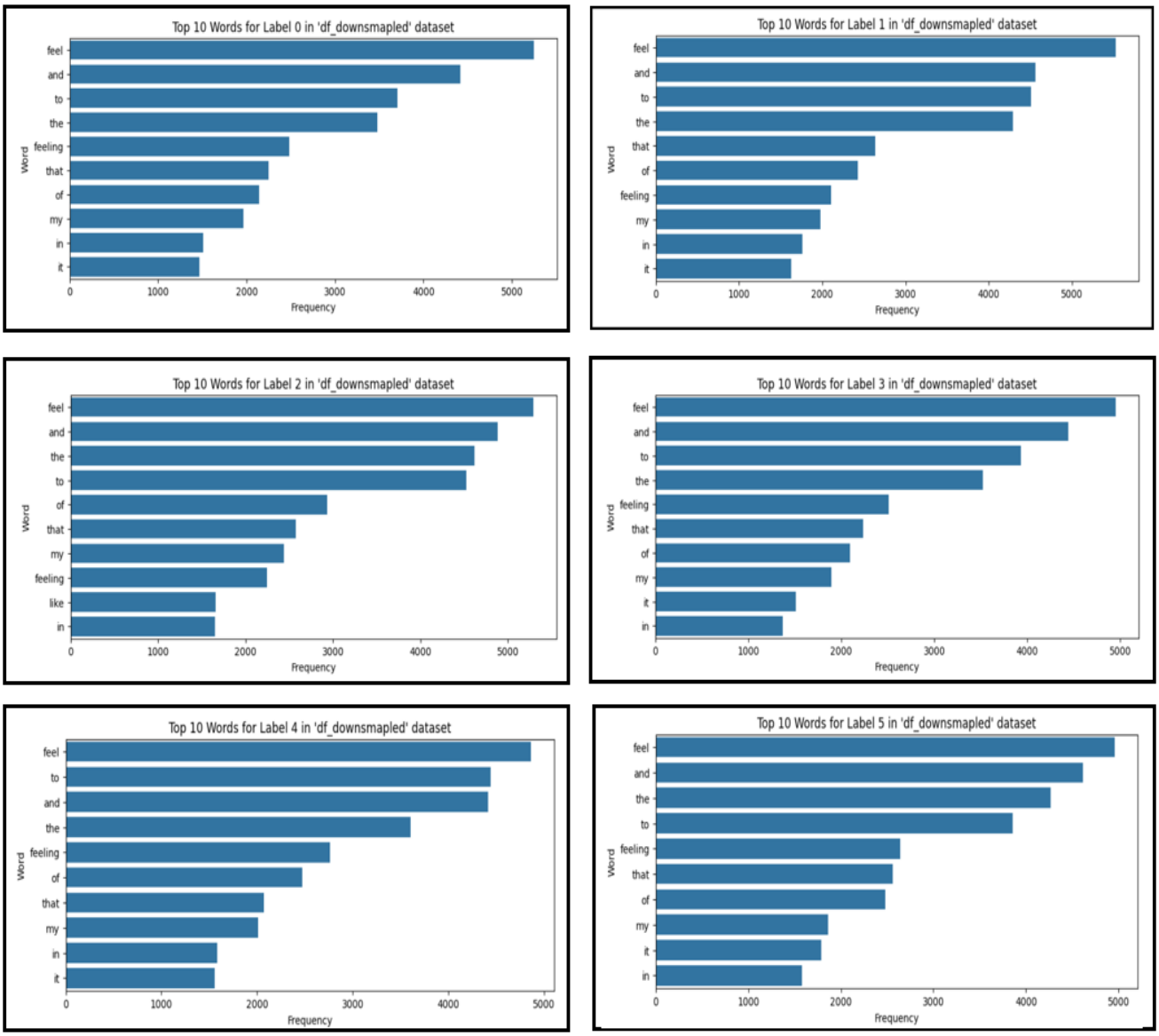

Figure 6: The top 10 most frequent words in every class.
(top left) - label 0. (top right) - label 1. (under top left) - label 2. (under top right) - label 3 And so on..

We can infer from Fig 6 that The words 'feel', 'and', 'to', 'the', 'feeling', 'that', 'of', 'my', 'in',and' 'it' appear consistently across all labels and their frequency is high. These words are likely common in everyday language and might not be highly distinctive for differentiate between labels.

We can see in Fig 7 that we omit those words because they don't help the models to predict the emotion.

We omitted those words by adding them to the stop words (words that don't helps the model, this is specific to our dataset and not from NLTK library).

```
custom_stopwords = {"i","im","like", "feel", "feeling", "my", "the", "to", "still"
                    "for","know","littl","think","time","thing","would","go",
                    "really","feel","am","so","get", "one", "to", "and", "at", "can",
                    "day","way", "make", "me", "want", "could" , "would", "tri", "u", "href", "http", "www", "com", "https"
                    }
```

Figure 7: Words that do not contribute to the models to predict the emotion and was omitted during the preproccesing.

3.5 Sentiment polarity distribution

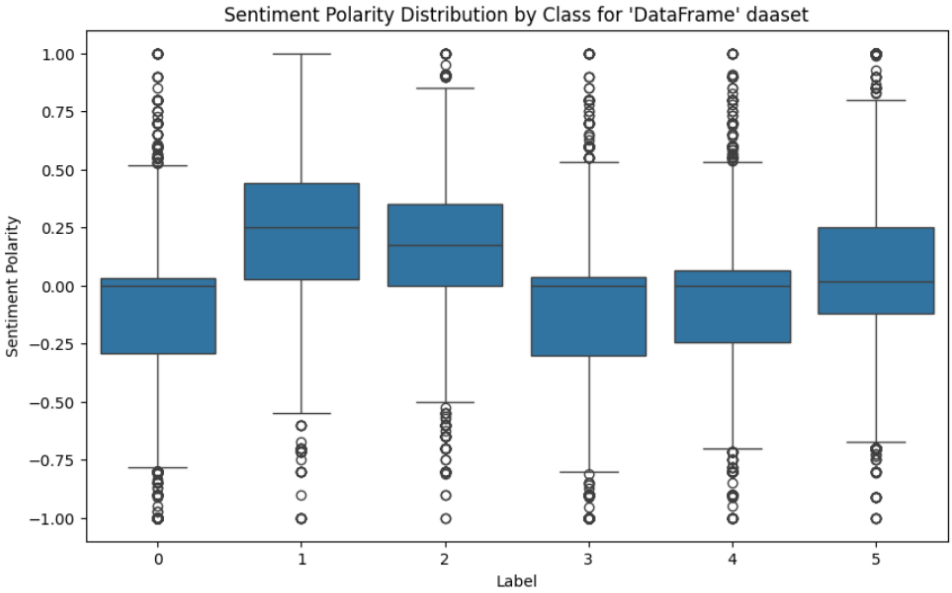

Figure 8: The sentiment polarity distribution. The boxplot shows the distribution of sentiment polarity for different classes. Each box represents the range where most of the data falls, with the line inside each box indicating the median sentiment polarity for that class. The dots outside the boxes are outliers, showing sentiment values that are much higher or lower than the rest.

Sentiment indicates whether a feeling is positive, negative, or neutral.
The sentiment range is from -1 to 1, where -1 is very negative and 1 is very positive.
As shown in Fig 8, the boxplot shows the distribution of sentiment polarity for different labels. Each box represents the range of sentiment values, with the line inside each box indicating the median sentiment polarity. The median is necessary as it provides the central tendency of sentiment values, unaffected by extreme values (outliers).
- **Label 1 (joy)**, **Label 2 (love)**, and **Label 5 (surprise)** are positive feelings, with boxes above 0.
- **Label 0 (sadness)**, **Label 3 (anger)**, and **Label 4 (fear)** are negative emotions, with boxes below 0.
The dots outside the boxes are outliers, showing extreme sentiment values. This plot visualizes how different emotions align with their expected sentiment polarity.

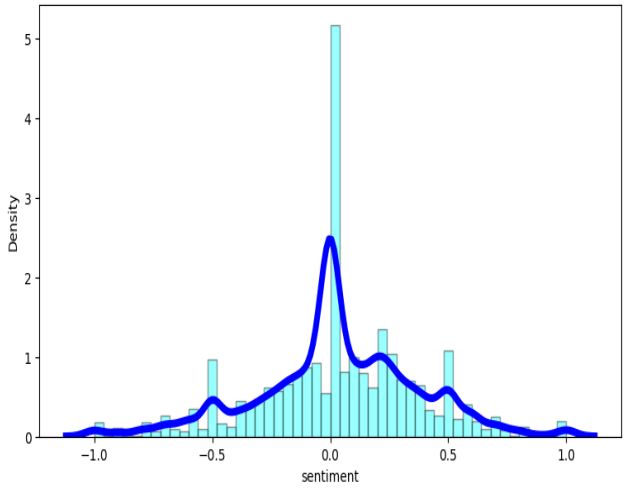

Figure 9: This plot shows the distribution of sentiment values. The histogram bars represent the frequency of different sentiment scores, ranging from -1 (very negative) to 1 (very positive). The blue line represents the density curve, indicating how the sentiment values are spread out. Most sentiment values are close to 0, meaning they are close to neutral.

This plot above give us another information about the sentiment and we can see that most of the data sentiment are close to 0.

3.6 Top 10 words after preprocessing

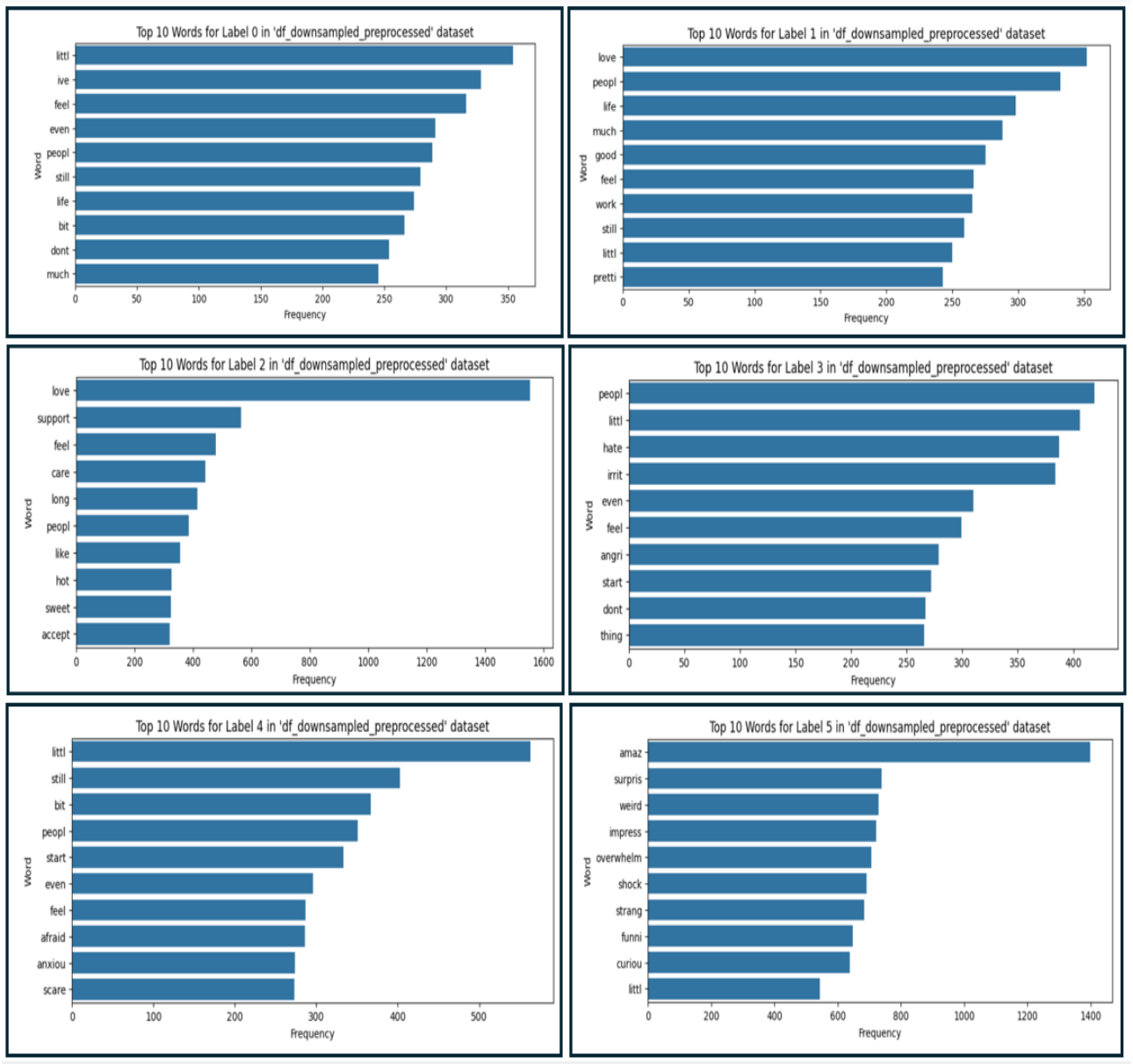

Figure 10: Top 10 most frequent words for each class after preprocessing. Each subplot represents a different class.
(top left) - label 0. (top right) - label 1. (under top left) - label 2. (under top right) - label 3 And so on..

.

Now the words closely related to the emotions that each class represents. For example, class 5 (surprise) includes words like 'overwhelm' , 'surprise,' and 'shock,' accurately reflecting the emotion that this class represents.

The figure above shows the top 10 most frequent words for each class in the dataset after preprocessing. Each class represents a different emotion, and now, after preprocess, the frequent words are closely related to these emotions. This alignment of words with their respective classes helps us understand the effectiveness of the preprocessing steps and the relevance of the words in capturing the intended emotions.

For example we can notice that:

- **Class 1 (Joy):** Words like 'love,' 'good' 'life', and 'work' are frequent. These words are associated with happiness and positivity, aligning well with the emotion of joy.
- **Class 2 (Love):** The words 'love', 'support', 'care,' and 'sweet' are dominant. These words are directly related to the emotion of love, indicating a strong alignment with the intended class.
- **Class 3 (Anger):** Frequent words include 'hate', 'angry', 'irrit' These words are often used in contexts of frustration or anger, matching the emotion of this class.

- **Class 4 (Fear):** Words such as 'afraid,' and 'scare' are prevalent. These words are indicative of fear or anxiety, showing a clear connection to the emotion of fear.
- **Class 5 (Surprise):** The words 'amazing,' 'surprise,' 'weird,' 'impress,' 'overwhelm,' and 'shock' are common. These words directly convey surprise and astonishment, accurately reflecting the emotion of this class.

This alignment of words with their respective emotions suggests that the preprocessing steps were effective in isolating relevant words for each class. It demonstrates that the model has a good understanding of the context and is capable of identifying words that are closely associated with specific emotions. This can improve the accuracy of sentiment analysis by ensuring that the words used for each class are representative of the underlying emotions. Additionally, the presence of these words in their respective classes helps validate the labels and provides confidence in the dataset's quality and the preprocessing techniques used.

3.7 Heatmap and Correlation

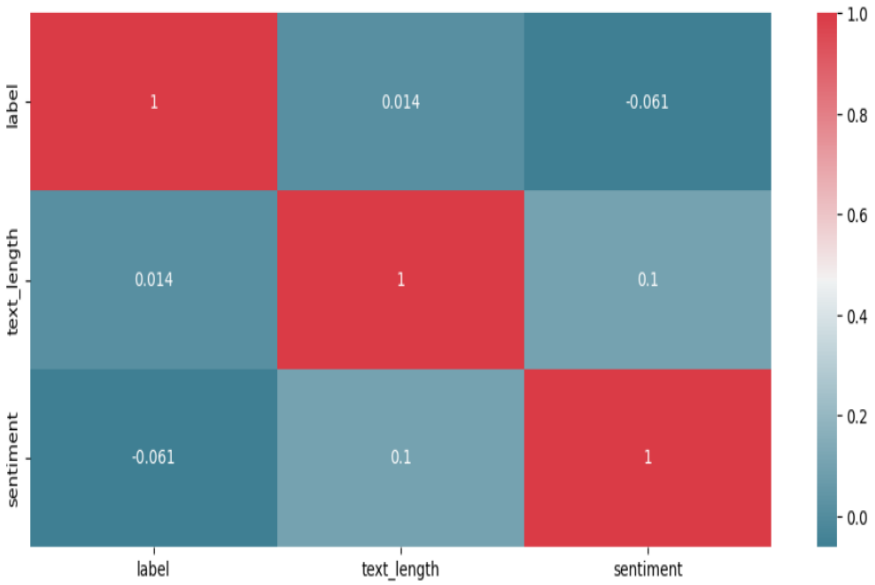

Figure 11: Correlation heatmap showing the relationship between label, text length, and sentiment in the dataset.

Fig 11 is a correlation heatmap that shows the relationships between three variables: label, text length, and sentiment. The values range from -1 to 1, indicating the strength and direction of the relationship between the variables.

- **Label and Text Length:** The value 0.014 shows almost no relationship between labels and text length. This means that knowing the label does not help predict the length of the text. - **Label and Sentiment:** The value -0.061 indicates a very weak negative relationship between labels and sentiment. This suggests that different labels are slightly associated with different sentiment scores, but the relationship is very weak. - **Text Length and Sentiment:** The value 0.1 shows a weak positive relationship between text length and sentiment. This implies that longer texts tend to have a slightly more positive sentiment, but the relationship is not strong.

In summary, the heatmap shows that the relationships between these variables are generally weak, meaning that changes in one variable do not strongly predict changes in another.

4 Algorithms used in the project

- **Dummy model**

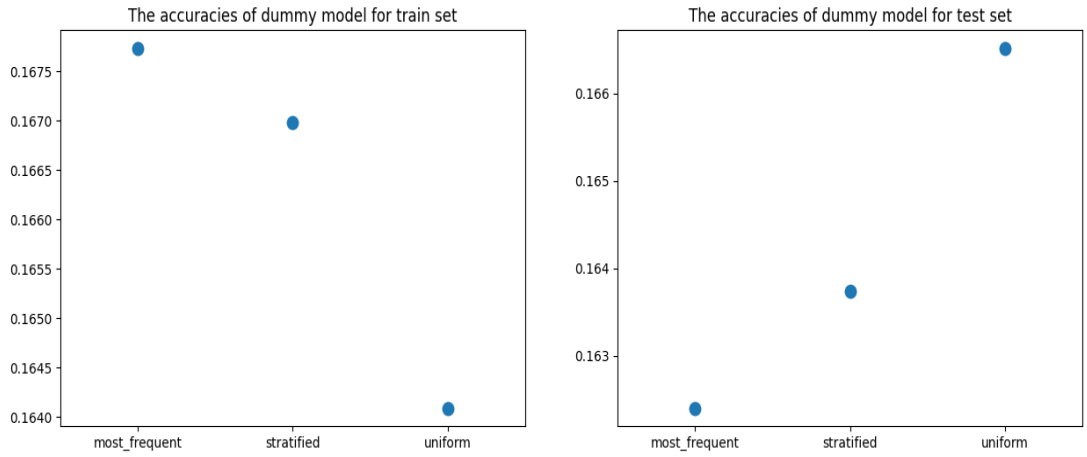

Figure 12: Train and test accuracy score for dummy model. We can see that the result is approximately 16% accuracy which is make sense because we have a classification 1/6 which is approximately 16%.

- **K-Nearest Neighbors (KNN)**
  - *Best Parameters:*
    - n_neighbors = 22
    - weights = 'distance'
    - algorithm = 'ball_tree'
- **Random Forest**
  - *Best Parameters:*
    - n_estimators = 119
    - max_features = 'sqrt'
    - max_depth = 15
    - min_samples_split = 5
    - min_samples_leaf = 10
- **XGBoost**
  - *Best Parameters:*
    - n_estimators = 95
    - max_depth = 8
    - learning_rate = 0.0499
    - subsample = 0.6713
    - colsample_bytree = 0.5097
- **AdaBoost**
  - *Best Parameters:*
    - n_estimators = 160
    - learning_rate = 0.3001
- **Linear Support Vector Machine (SVM)**
  - *Best Parameters:*
    - C = 1.3446
    - kernel = 'linear'
    - degree = 4
    - gamma = 'scale'
    - random_state = 42

- **Convolutional Neural Network (CNN)**
  - *Best Parameters:*
    - vocab_size = 10000
    - embedding_dim = 100
    - max_length = 100
    - trunc_type = 'post'
    - padding_type = 'post'
    - optimizer = 'adam'
    - loss = 'sparse_categorical_crossentropy'
    - epochs = 20
  - *Model Architecture:*
    - Embedding(vocab_size, embedding_dim, input_length=max_length)
    - Conv1D(128, 5, activation='relu')
    - GlobalMaxPooling1D()
    - Dense(128, activation='relu')
    - Dense(6, activation='softmax')

5 Models evaluation

5.1

Those are the accuracies for all machine learning models on train and test, using the 'metrics.accuracy_score' method in the sklearn library.
We will show first the accuracies with the pre-trained embedding names Glove and then with the CountVectorizer embedding:

```
The train Accuracy for knn model is: 0.9715295558276745
The test Accuracy for knn model is: 0.41763134461264473

The train Accuracy for linearSvc model is: 0.447122342201937
The test Accuracy for linearSvc model is: 0.43544078361531613

The train Accuracy for xgb model is: 0.8235834353779361
The test Accuracy for xgb model is: 0.4718388245770258

The train Accuracy for abc model is: 0.39173995324501837
The test Accuracy for abc model is: 0.3705476402493321

The train Accuracy for randomForest model is: 0.847573193810531
The test Accuracy for randomForest model is: 0.44367764915405167
```

Figure 13: GloVe embedding

```
The train Accuracy for knn model is: 0.9907445691930918
The test Accuracy for knn model is: 0.7525788497217069

The train Accuracy for linearSvc model is: 0.9429407461594733
The test Accuracy for linearSvc model is: 0.8784415584415585

The train Accuracy for xgb model is: 0.8622499284373907
The test Accuracy for xgb model is: 0.8485343228200372

The train Accuracy for abc model is: 0.3169110397251996
The test Accuracy for abc model is: 0.3044155844155844

The train Accuracy for randomForest model is: 0.8519449126936166
The test Accuracy for randomForest model is: 0.8473469387755102
```

Figure 14: CountVectorizer embedding

And this is the accuracy for the CNN model (self-embedding):

```
Epoch 20/20
1123/1123 - 10s - loss: 0.0227 - accuracy: 0.9871 - val_loss: 0.6462 - val_accuracy: 0.8975 - 10s/epoch - 9ms/step
281/281 [==============================] - 1s 3ms/step - loss: 0.6462 - accuracy: 0.8975
Test Accuracy: 0.8974844217300415
```

Figure 15: CNN with self-embedding

The hyper-parameters for each algorithm were tuned using Optuna with cross-validation. The tuning process involved running multiple trials (n_trials=50), where Optuna suggested different hyper-parameter values. Each trial's performance was evaluated using 5-fold cross-validation, and the best hyperparameters were selected based on the highest average accuracy score.

5.2 Confusion Matrices Analysis

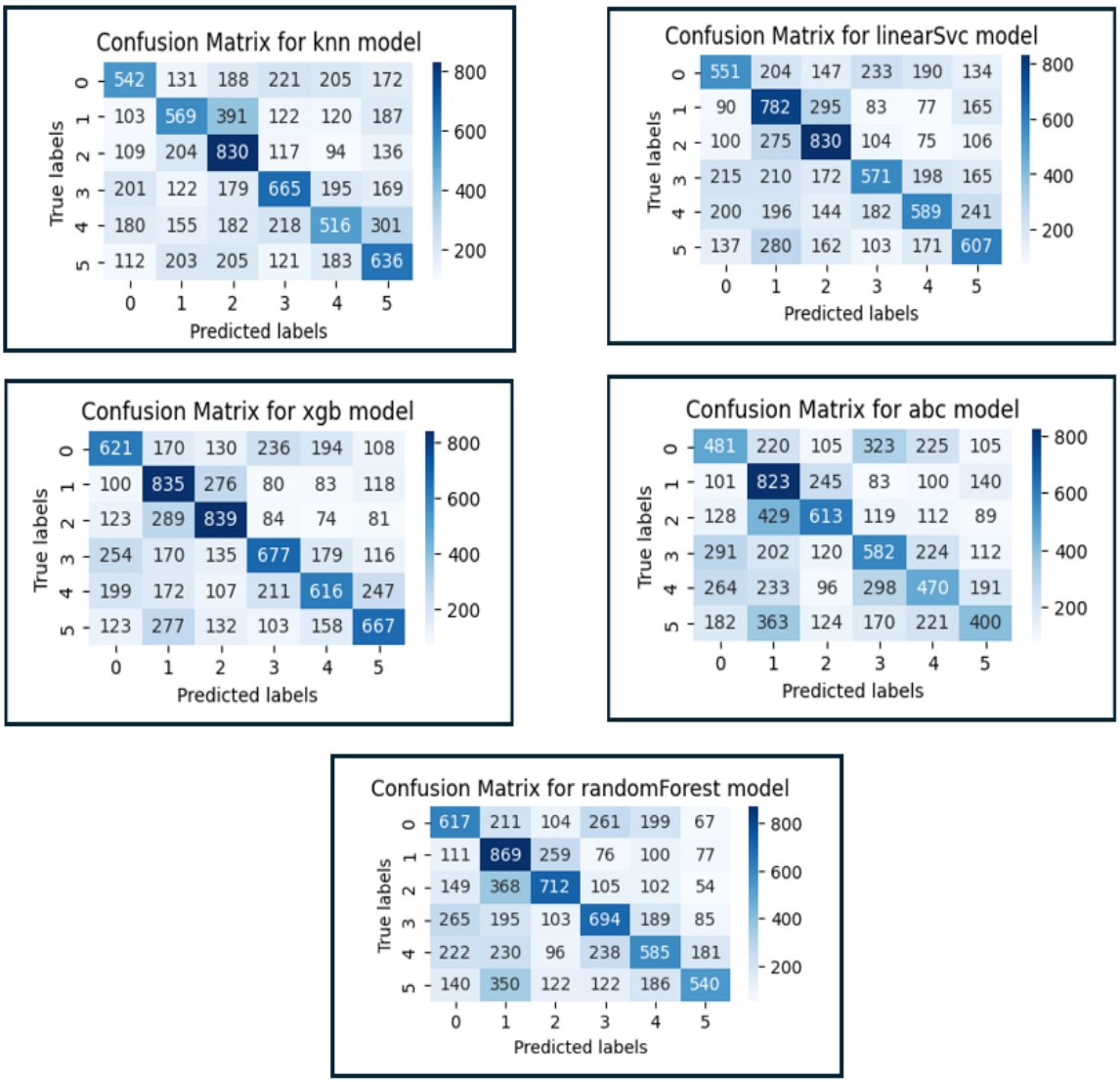

Figure 16: Confusion matrices for different models: KNN, Linear SVC, XGBoost, AdaBoost, and Random Forest. Each matrix shows the performance of the respective model in classifying the emotions labeled as 0 (sadness), 1 (joy), 2 (love), 3 (anger), 4 (fear), and 5 (surprise). The diagonal elements represent correct classifications, while the off-diagonal elements represent misclassifications.

Fig 16 shows the confusion matrices for various models, including KNN, Linear SVC, XGBoost, AdaBoost, and Random Forest. Each confusion matrix provides a detailed breakdown of the model's performance in classifying the different emotion labels.

5.2.1 Key Observations

1. **Correct Classifications (Diagonal Elements):**
   - The diagonal elements (from top-left to bottom-right) represent correctly classified instances. For example, in the KNN model, 542 instances of class 0 are correctly classified as class 0.
2. **Misclassifications (Off-Diagonal Elements):**
   - These elements represent misclassifications. For example, in the KNN model, class 0 is often misclassified as class 1 (131 instances) and class 2 (188 instances).
3. **Performance of Specific Models:**
   - **KNN Model:** Performs well for class 2 (love) with 830 correct classifications but struggles with class 0 (sadness).
   - **Linear SVC Model:** Performs well for class 1 (joy) with 782 correct classifications.

- **XGBoost Model:** Shows high accuracy for class 2 (love) and class 3 (anger) with 839 and 677 correct classifications respectively.
- **AdaBoost Model:** Struggles with class 0 (sadness) and class 3 (anger) but performs well for class 2 (love).
- **Random Forest Model:** Displays balanced performance across all classes with notable accuracy for class 1 (joy) and class 2 (love).

5.2.2 Insights on Misclassification

- The most significant misclassification occurs between label 1 (joy) and label 2 (love). This is evident in all models, where instances of joy are often classified as love and vice versa.
- We investigated this misclassification and found that it is likely due to the similarity between these emotions, both of which have positive sentiment and are closely related.
- Additionally, there is an overlap in the top 10 frequent words for these classes. For example, the word 'love' appears frequently in both joy and love classes, contributing to the confusion.

In summary, the confusion matrices highlight the performance differences across models and shed light on specific challenges, such as the misclassification between joy and love due to their positive sentiment and overlapping vocabulary. This analysis helps in understanding the strengths and weaknesses of each model and provides direction for further improvements.

6 Our Questions about the Project

We want you to know that what we are going to write is according to our project and we know there is more techniques in the world for some things like embedding for example but we are going to write only the relevant things.

6.1 Converting Text to Numeric Data

Question: Models cannot understand text, so how can we convert the text into numeric data?
Answer: We can convert text into numeric data using emedding techniques.

- **Word Embeddings:** Such as Word2Vec, GloVe. This is mapping words to high-dimensional vectors capturing semantic meaning.

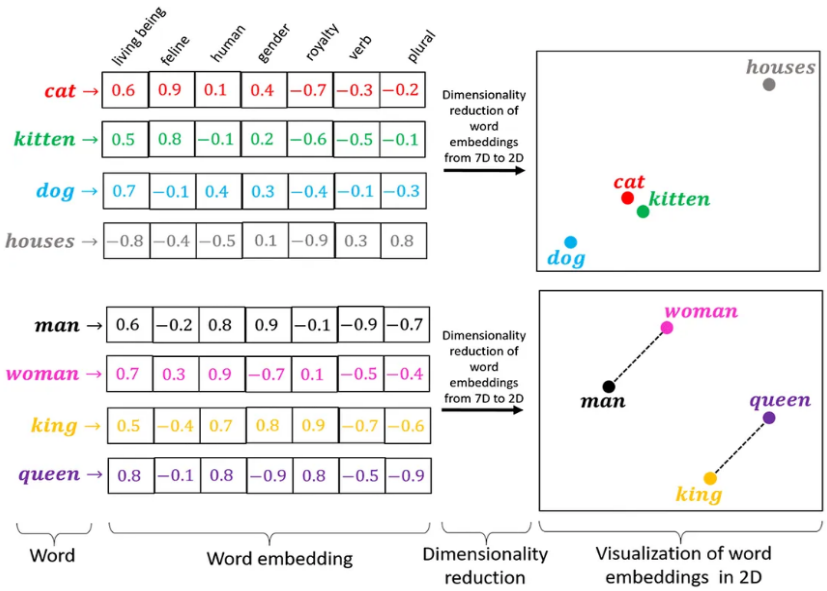

Figure 17: Plot that show word embedding process. We can notice that word that close to each other (like man and woman) will be close also in the embedding space.

6.2 Handling Imbalanced Data

Question: The data is imbalanced, what can we do to fix that?
Answer: To address imbalanced data, we can:

- **Resampling Techniques:** Oversample the minority class or undersample the majority class. That's what we have done in our project. We take the minority class, divide it by 2 and put this number of examples to every class.

6.3 Data Analysis

Question: What analyses can be done for the data?
Answer: Various analyses can be performed on the data, such as:

- **Exploratory Data Analysis (EDA):** Understand the distribution of data, relationships between features and more.
- **Sentiment Analysis:** Assess the sentiment of text data.
- **Correlation Analysis:** Determine relationships between different features.

6.4 Embedding Techniques

Question: Which embedding technique should we use?
Answer: Different embedding techniques which was used:

- **Pre-trained Embeddings:** Such as GloVe and Word2Vec, which capture semantic meaning. We have used this techniques in our project.

6.5 Correlation Analysis

Question: What is the correlation between the main features and the label?
Answer: To determine the correlation between main features and the label, we can:

- Visualize correlations using heatmaps to identify strong and weak relationships. You can see again the correlation in Fig 11.

6.6 Data Splitting

Question: How should we split the data?
Answer: We used a 80-20 split for training and testing.

6.7 Efficient Tuning Process

Question: The tuning process takes a long time, what can we do to make it more efficient?
Answer: The tuning process can be made more efficient by:

- **Random Search:** Instead of grid search, which explores fewer points in the hyperparameter space.
- **Bayesian Optimization:** Uses probabilistic models to find the best hyperparameters efficiently.

In our project, we used Optuna for efficient hyperparameter tuning. Optuna employs advanced optimization techniques, including random search and Bayesian optimization, to explore the hyperparameter space effectively.

This approach significantly reduced the time required for hyperparameter tuning compared to traditional grid search, while also improving the performance of our models by finding more optimal hyperparameter combinations.

6.8 Dummy Model Accuracy

Question: How much accuracy would the dummy model give?
Answer: The dummy model gives a baseline accuracy:

- Typically, for a 6-class classification problem, the dummy model would give around 16.67% accuracy (1/6). And we saw that this is what happened.

6.9 XGBoost vs. Random Forest Accuracy

Question: Why do the XGBoost and Random Forest models give close accuracies?
Answer: XGBoost and Random Forest models give close accuracies because:

- Both are ensemble methods that reduce variance by combining multiple models.
- XGBoost often performs better due to gradient boosting but might be similar in some cases due to data or hyperparameters.

6.10 KNN and GloVe Embedding

Question: Why does the KNN model give a weak accuracy with the GloVe embedding?
Answer: The KNN model gives weak accuracy with the GloVe embedding because:

- KNN relies on distance metrics, and high-dimensional embeddings like GloVe might not work well with it.

6.11 Key Hyperparameters and Impact

Question: What are the key hyperparameters for each model, and how do they impact performance?
Answer: Key hyperparameters for each model include:

- **KNN:** Number of neighbors, distance metric.
- **Random Forest:** Number of trees, max depth, min samples split.
- **XGBoost:** Learning rate, number of estimators, max depth.
- **SVM:** C (regularization), kernel type, gamma.

Hyperparameters affect model complexity, training time, and performance.

6.12 CountVectorizer vs. GloVe Embedding

Question: Why does the CountVectorizer embedding give better results than the GloVe embedding?
Answer: CountVectorizer gives better results than GloVe embedding because:

- Simpler models can sometimes perform better on specific tasks due to less noise.
- **Pretrained Nature of GloVe:** GloVe is a pretrained embedding model, which means it is trained on a general corpus and may not be perfectly suited to the specific vocabulary and nuances of our dataset. In contrast, CountVectorizer is built directly from the training data, making it more tailored to the specific task.

6.13 Overfitting Issues

Question: What are the potential overfitting issues with complex models, and how can we fix them?
Answer: Potential overfitting issues with complex models can be addressed by:

- **Cross-Validation:** Ensure robust performance across different data splits.
- **Regularization:** Techniques like L1/L2 regularization, dropout.
- **Simpler Models:** Use less complex models if overfitting is detected.

6.14 Evaluation Metrics

Question: How do different evaluation metrics (e.g., accuracy, precision, recall, F1-score) affect our understanding of model performance?
Answer: Different evaluation metrics affect our understanding of model performance:

- **Accuracy:** Proportion of correctly predicted instances.
- **Precision:** Proportion of true positives out of predicted positives.
- **Recall:** Proportion of true positives out of actual positives.
- **F1-Score:** Harmonic mean of precision and recall, useful for imbalanced data.

6.15 Generalization to New Data

Question: How can we ensure our model works well on new, unseen data?
Answer: To ensure the model works well on new, unseen data:

- Use cross-validation to simulate performance on new data.
- Regularize the model to avoid overfitting.

6.16 Deep Learning vs. Traditional Models

Question: What are the advantages and disadvantages of using deep learning models (like CNN) compared to traditional machine learning models for text classification?
Answer: Advantages and disadvantages of using deep learning models (like CNN) compared to traditional machine learning models for text classification:

- **Advantages:** Capture complex patterns, better performance on large datasets, automatic feature extraction.
- **Disadvantages:** Require more data and computational resources, longer training times.

6.17 Regularization Techniques

Question: How do various regularization techniques (e.g., dropout, L1/L2 regularization) impact model training and performance?
Answer: Various regularization techniques impact model training and performance:

- **Dropout:** Prevents overfitting by randomly dropping neurons during training.
- **L1/L2 Regularization:** Adds a penalty to the loss function to constrain model complexity.
- **Early Stopping:** Stops training when performance on a validation set stops improving.

You can watch and explore more about the project in this Link to the project