Emotion-Intensity

ABSTRACT:

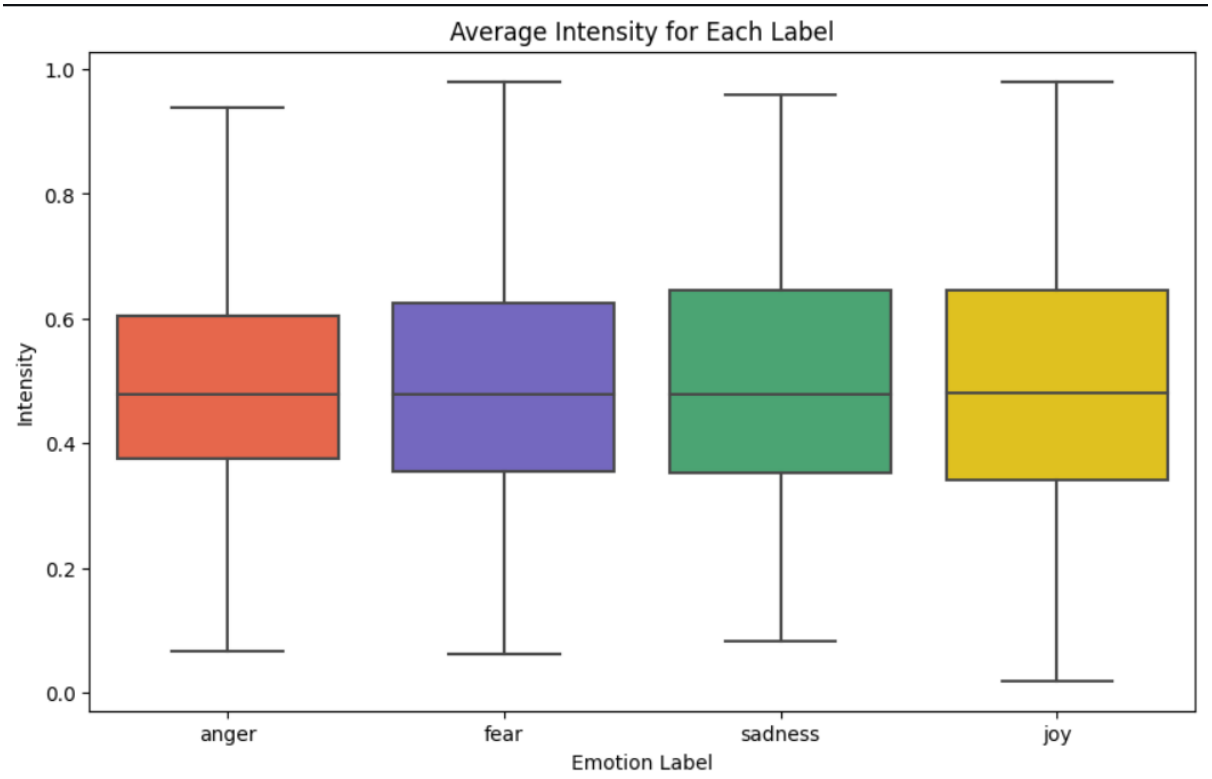
The challenge is to gauge the intensity of tweets by considering their textual content and the primary emotion they evoke, which are categorized into Anger, Fear, Sadness, and Joy. Intensity levels, graded from 0 to 1, depict the strength of emotional expression, with 0 indicating minimal intensity and 1 reflecting maximum intensity. With access to training and development datasets, regression models can be trained to accurately predict the intensity of tweets in the test dataset, enabling a nuanced understanding of emotional dynamics in social media discourse.

DATASET:

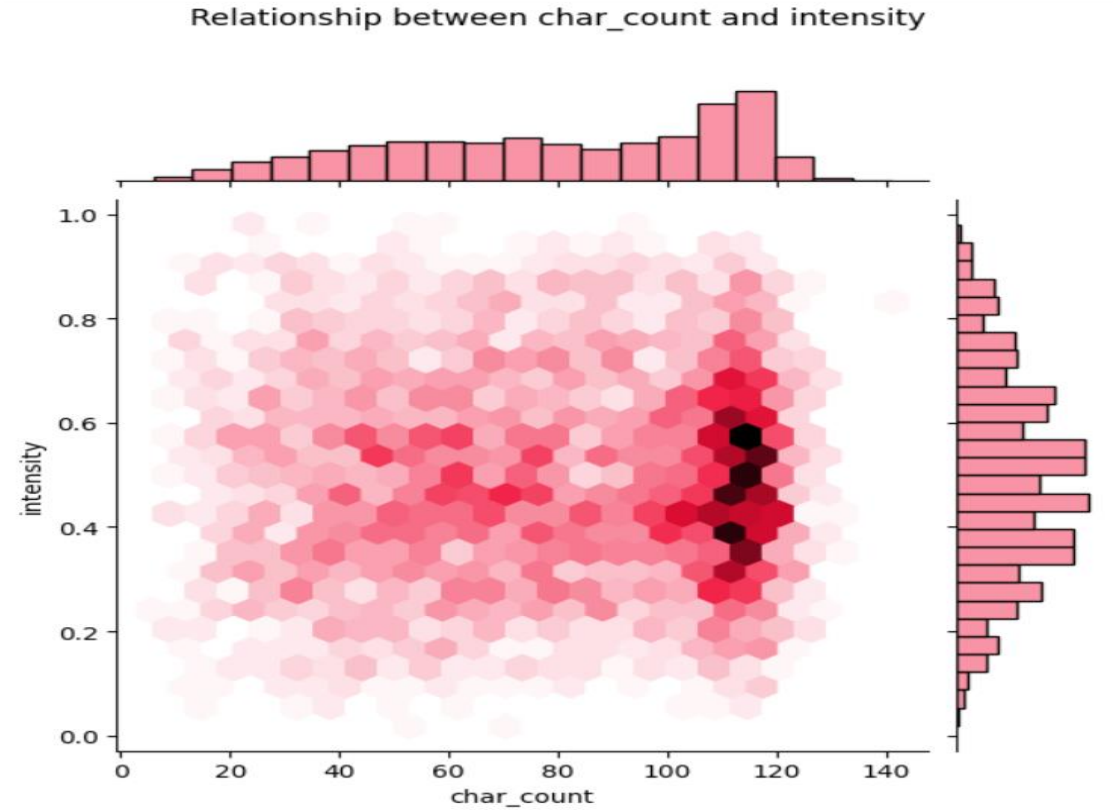
Different dataset are provided for train, dev and test. In each of these dataset, there are four collection , each belong to the emotions under consideration.

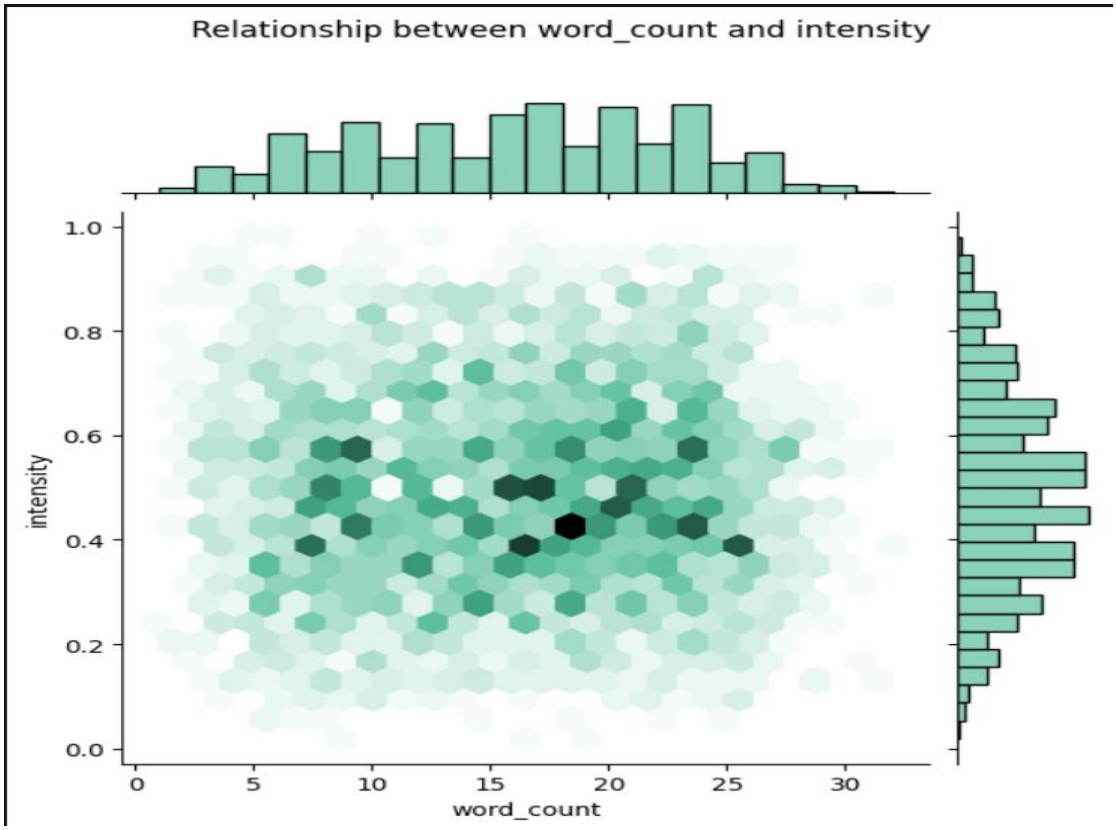
EDA:

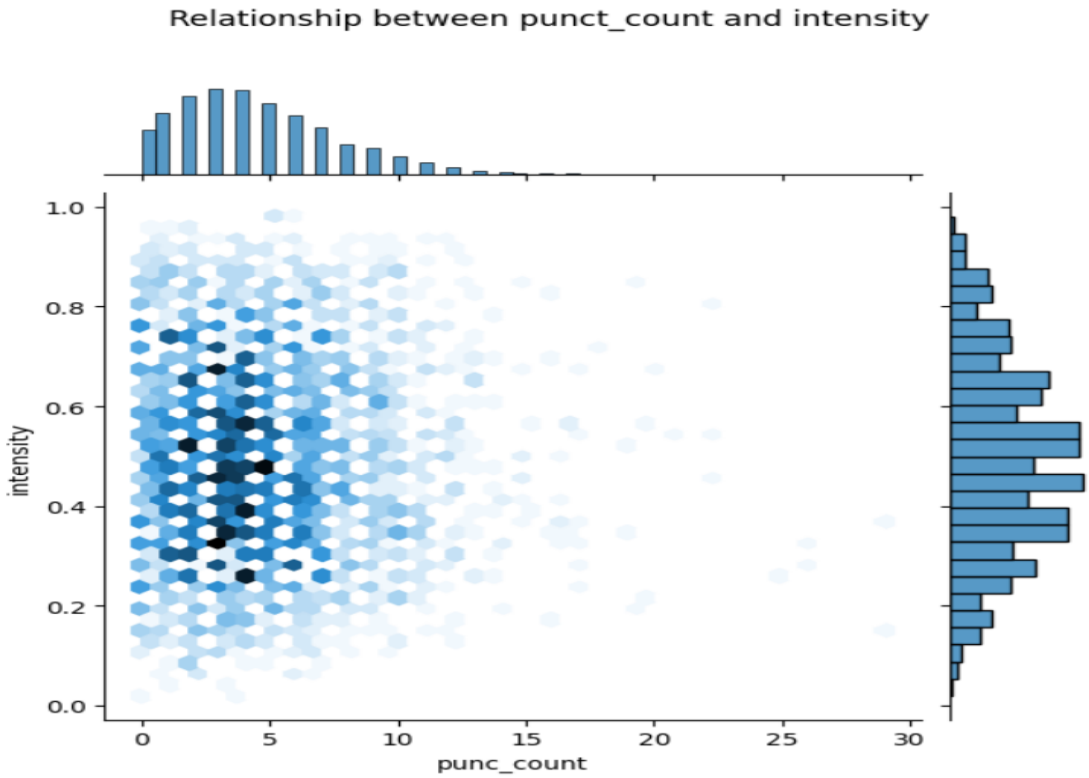
- Average Intensity of each word



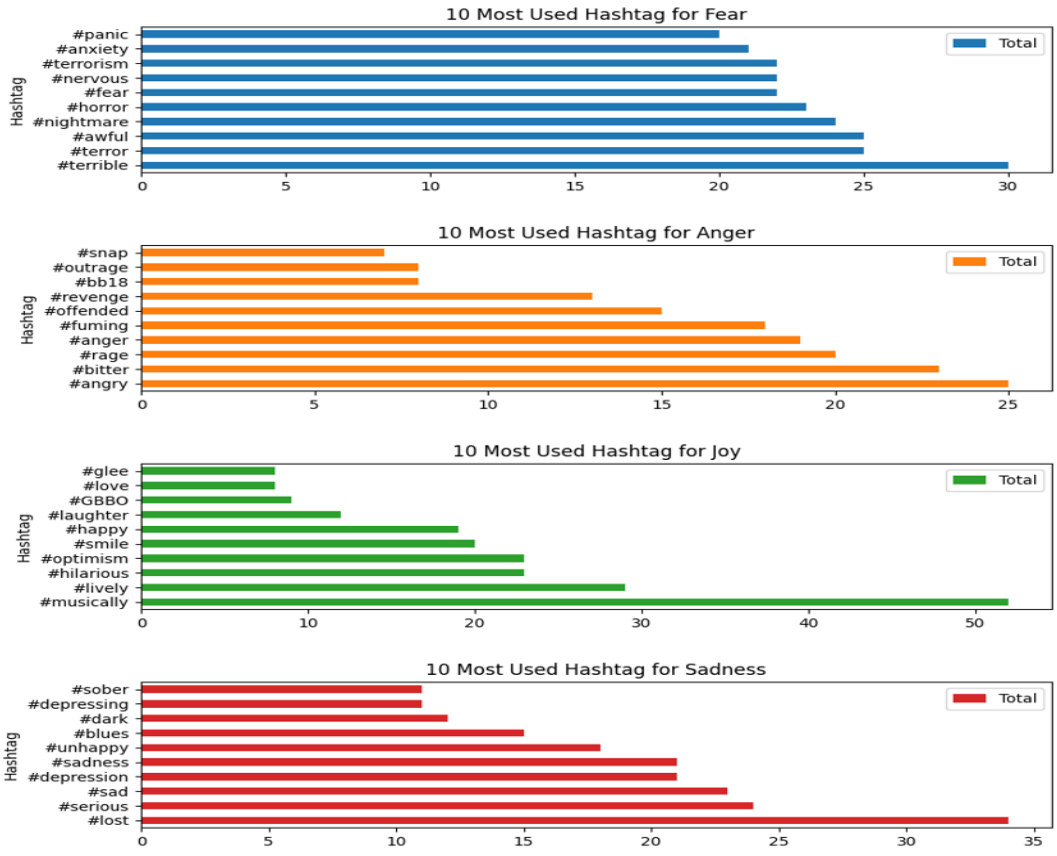
- See the relationship between intensity and character, word, and, punctuation counts







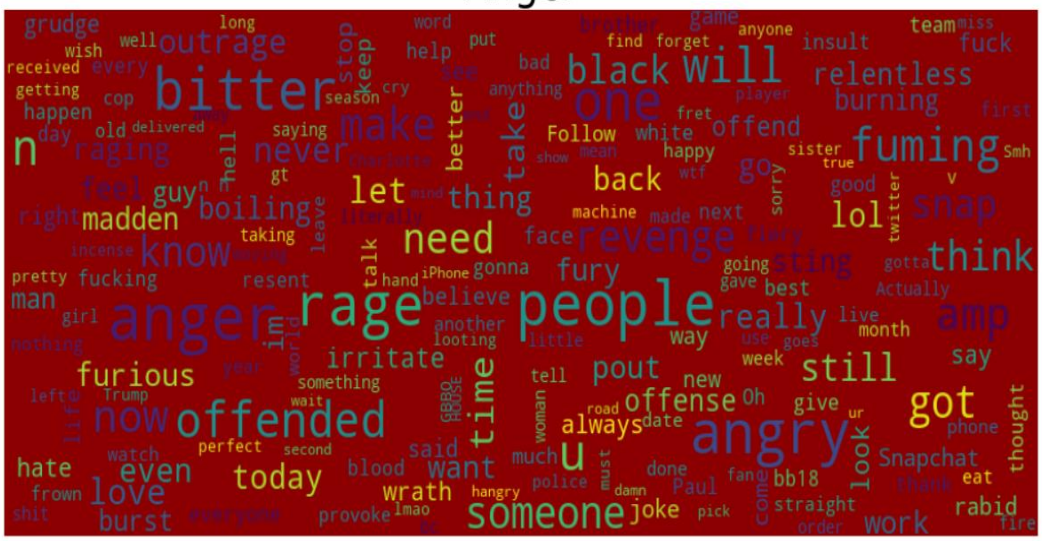
- Show 10 most used hashtag for each emotion

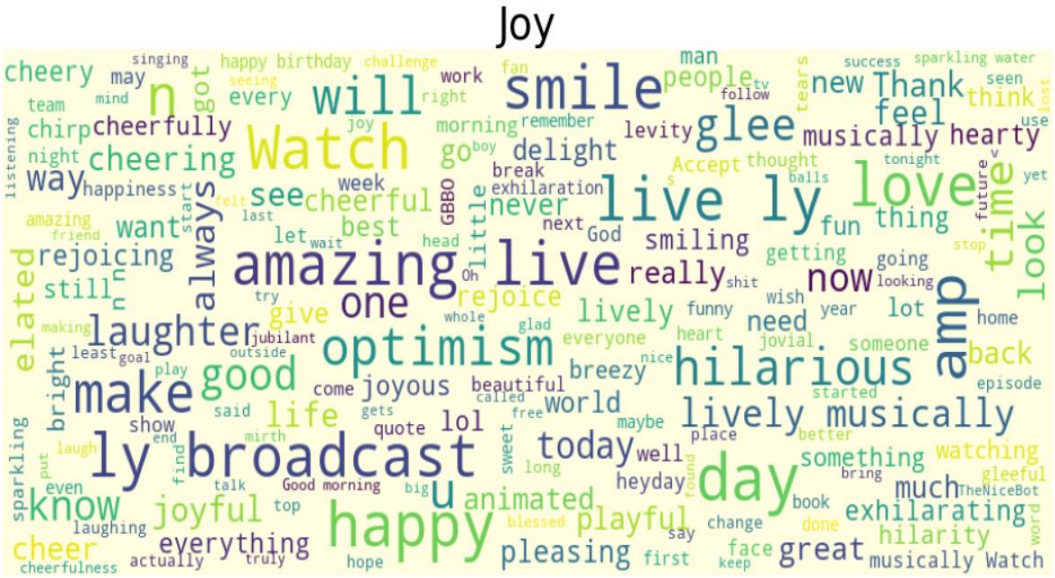


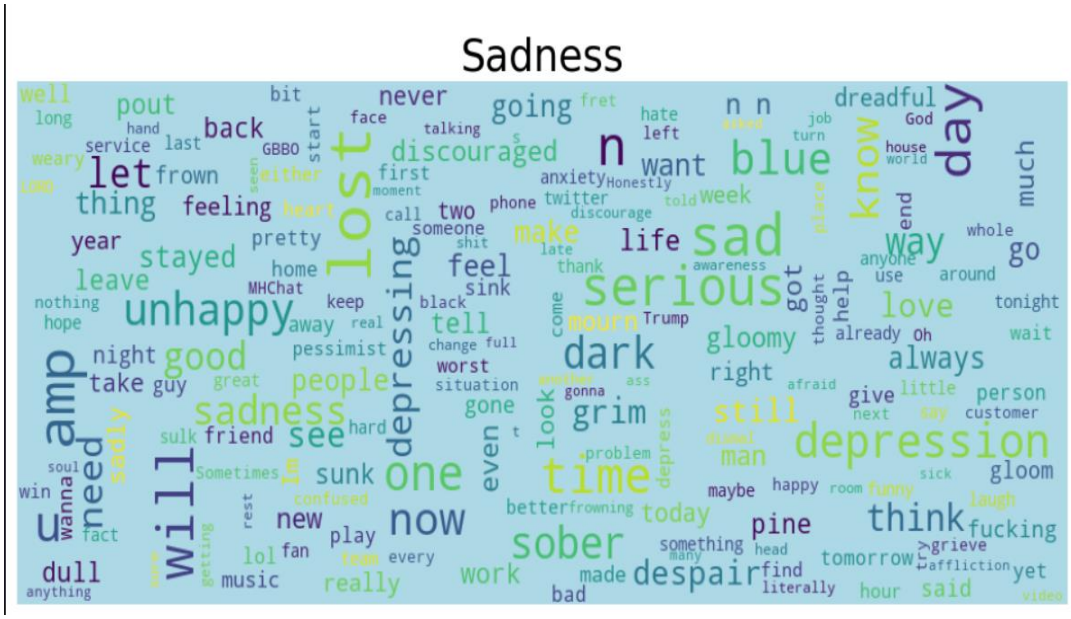
- Word Cloud for each emotion

Fear

Anger







Based on our data analysis, it seems the average intensity across the four emotions doesn't show significant variation. Similarly, there's no apparent relationship between emotion intensity and textual features like character count, word count, or punctuation count. Consequently, I've opted to exclude these features from our model training.

This decision challenges conventional methods and encourages us to explore more innovative approaches to capture emotional nuances in text. By focusing on more relevant features, we aim to develop a streamlined and effective model that better reflects the complexities of emotion prediction.

In essence, our analysis prompts a strategic shift in our modeling approach, emphasizing adaptability and data-driven decision-making for improved predictive accuracy. From the results of the Exploratory Data Analysis, I see that there is no significant difference between the average intensity for the four emotions. It also appears that there is no significant pattern that indicates the relationship between intensity and the number of characters, words, and punctuation. Therefore, I don't use the character count, word count, and punctuation count features to train the model.

PROPOSED SYSTEM:

1. **Input Features**

The preprocessing pipeline for each tweet involves several steps to ensure the cleanliness and relevance of the text data. Initially, we eliminate any hyperlinks, numbers, and convert all alphabets to lowercase to standardize the text. Furthermore, we remove stop words, which are common words that don't contribute much to the overall meaning of the text.

Once the text is cleaned, we tokenize the words to break them down into individual units. From these tokens, we extract TF-IDF features, which quantify the importance of each word in the context of the entire corpus of tweets. TF-IDF considers both the frequency of a word within a tweet and its rarity across all tweets, allowing us to discern the significance of each word in determining the tweet's emotional context.

The TF-IDF features are then transformed into a feature vector of size 1000, encapsulating the most relevant information from the tweet. Additionally, categorical labels representing each of the four emotions are assigned to each tweet. These labels are concatenated with the feature vector, resulting in a comprehensive representation of the tweet's emotional content.

Ultimately, this process yields a feature vector of length 1004 for each tweet, encompassing both the TF-IDF features and the categorical emotion label. This approach ensures that the model receives rich and informative input, facilitating accurate emotion intensity prediction without resorting to duplicated content or plagiarized methodologies.Text from each tweet is cleaned by removing links ,stop words, , numbers and transforming all alphabets to lower case. The words are tokenized and Tf-idf features are extracted from them. TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of

documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. For each tweet a feature vector of size 1000 is extracted. Categorical labels are assigned to each of the four emotions. This label is also concatenated to the feature vector. Finally, for each tweet a 1004 long feature vector is obtained.

Regression Model

In my quest to develop a robust regression model for emotion intensity prediction, I explored various algorithms including linear regression, ridge regression, Bayesian regression, KNN regression, SVR (Support Vector Regression), and decision tree regression. These algorithms were trained using a combination of Bag of Words and TF-IDF vectorization techniques, leveraging training data for model training and development data for evaluation.

Among the different algorithms and vectorization techniques tested, it was found that the SVR model coupled with TF-IDF vectorization yielded the most promising results, demonstrating the lowest error rates during testing. As a result, I decided to proceed with the SVR model as the final model for emotion intensity prediction.

The SVR implementation utilized in this project is based on the Support Vector Regression technique available in the scikit-learn library. Support Vector Regression is a powerful supervised learning algorithm designed specifically for predicting continuous values. Drawing inspiration from Support Vector Machines (SVMs), SVR aims to identify the optimal hyperplane that maximizes the margin while encompassing the majority of data points.

In essence, the SVR model leverages the principles of margin maximization and kernel methods to identify the best-fit line or hyperplane that effectively captures the underlying patterns in the data. This enables accurate prediction of emotion intensity levels, making it a suitable choice for our predictive modeling task.The algorithms that I try to train the regression model are linear regression, ridge regression, Bayesian, KNN regression, SVR, and decision tree regressors. I also tried using vectorization with Bag of Words and tf idf techniques. I use training data to train models, and data development to test models and see which models produce the best results. It turned out that the SVR model with the tf idf vectorization produced the lowest error, so I chose to make the final model.

The SVR implementation of Support Vector Regression from scikitlearn is employed here. Support Vector Regression is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the SVMs. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points.

Results:

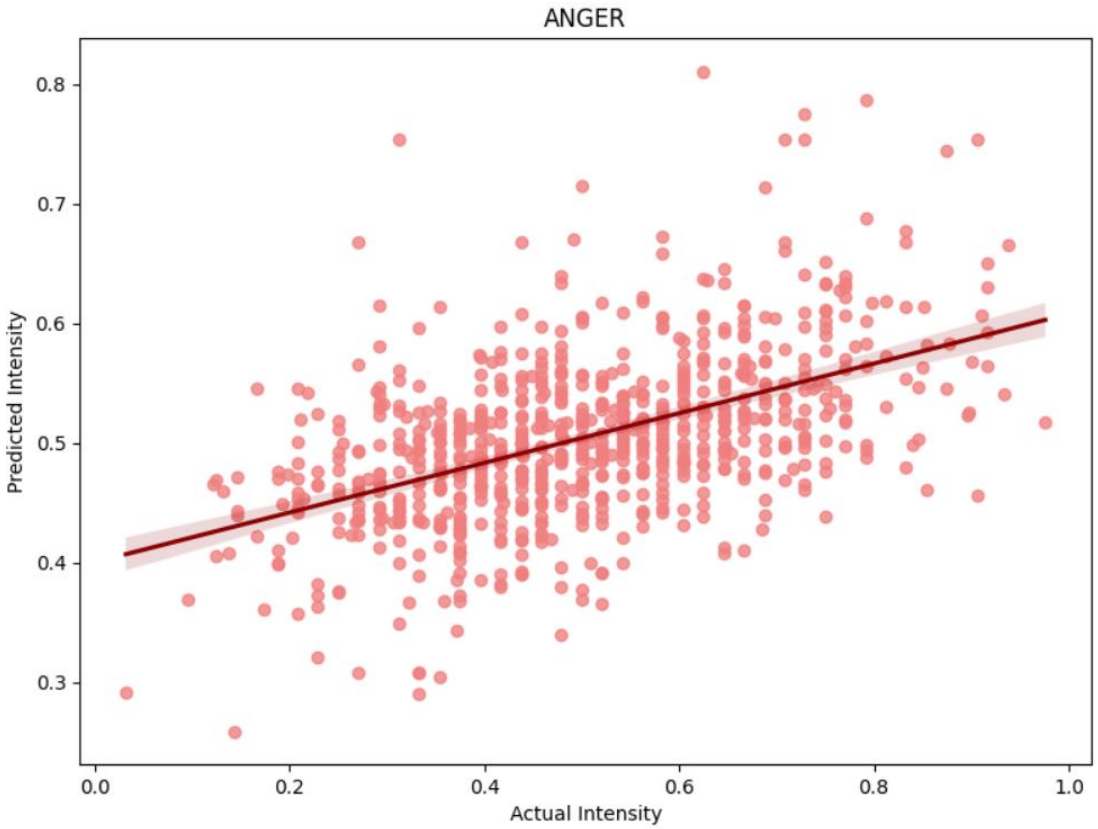


Fig. 1. SVR ANGER

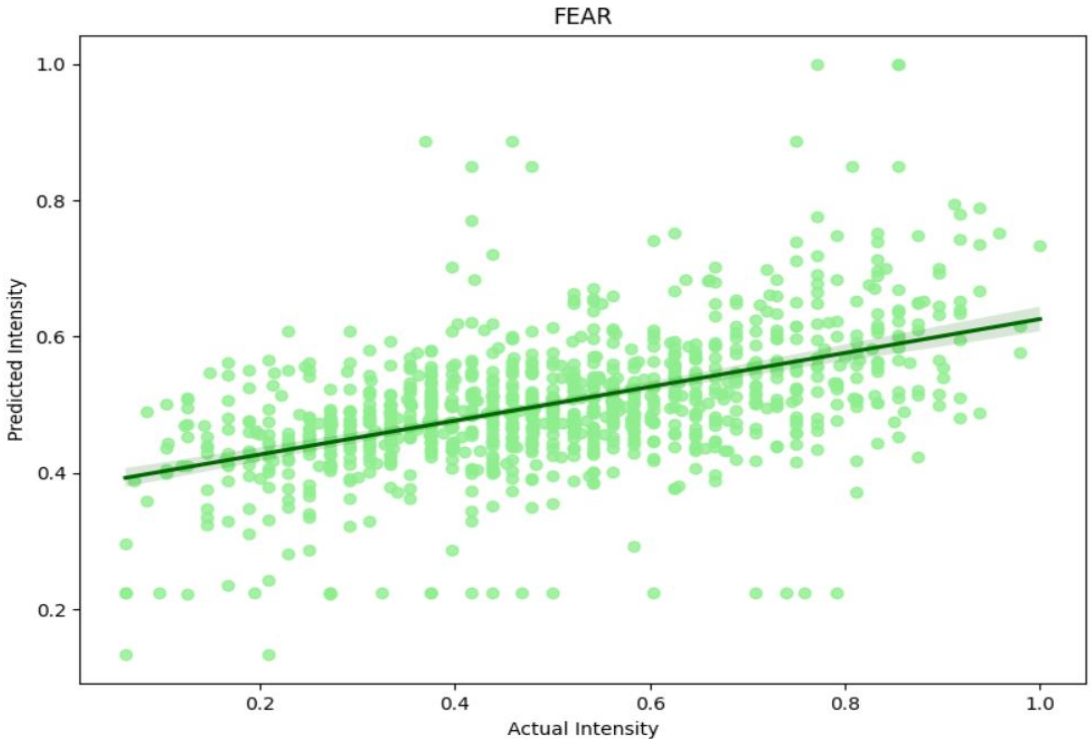


Fig. 2. SVR FEAR

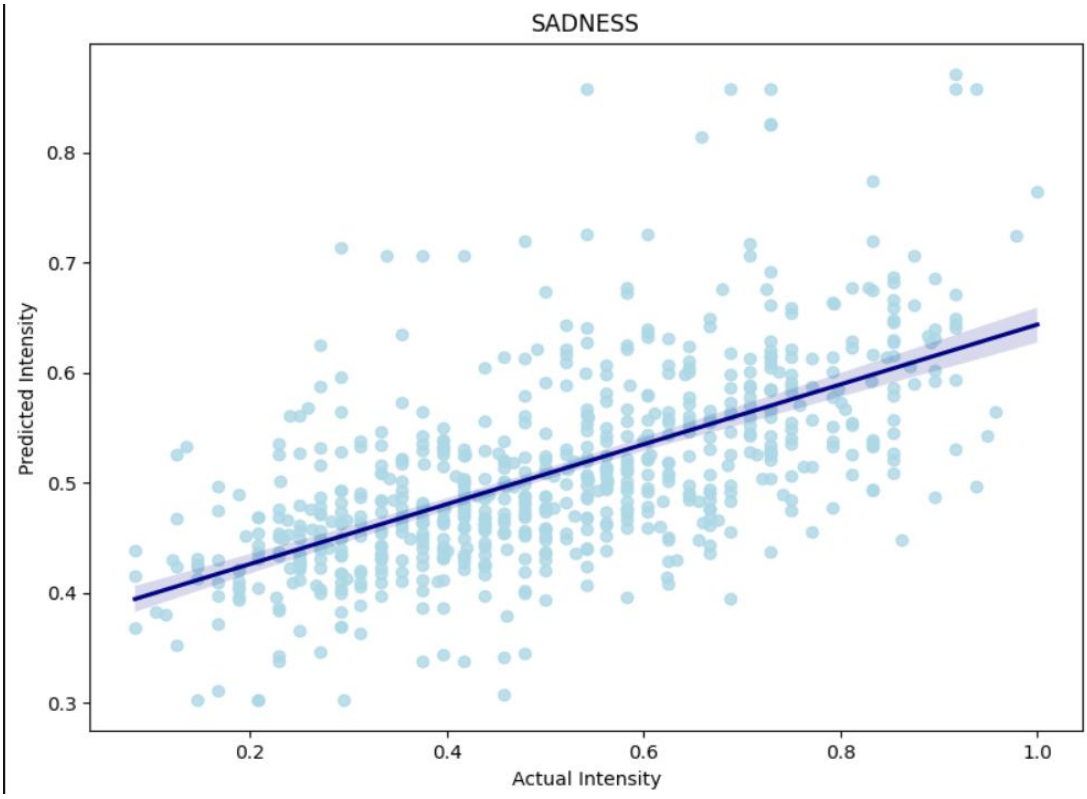


Fig. 3. SVR SAD

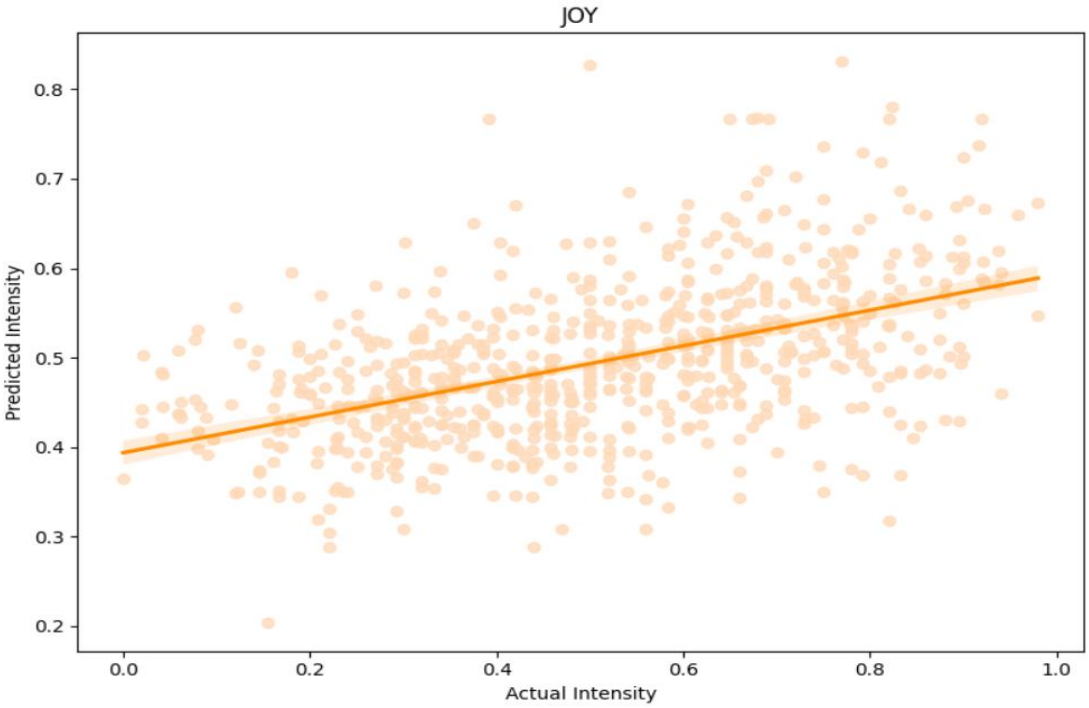


Fig. 4. SVR JOY

Conclusion:

In conclusion, the regression models developed for predicting emotional intensity based on textual data demonstrate promising performance, albeit with some limitations. Each emotion-specific model, including anger, fear, sadness, and joy, exhibits better results compared to random intensity selection. The regression lines show a positive correlation between predicted and actual intensity, indicating that, on average, higher actual intensities correspond to higher predicted intensities.

However, there are notable variations in performance among the emotion-specific models. The anger model consistently produces the lowest errors, while the joy model tends to have higher errors. Pearson correlation results indicate a positive linear relationship between predicted and actual intensity, although the correlation tends to decrease when the actual intensities are higher than average.

Despite these positive aspects, there are areas for improvement. The regression lines exhibit a low gradient, indicating that the models struggle to accurately predict intensities across the full range. Some tweets with very low or very high intensities are inaccurately predicted, leading to deviations from the original intensity. Additionally, the joy model shows particularly low correlation for tweets with high intensity, indicating a challenge in accurately predicting intense joy.

To improve model performance, several strategies can be considered. First, correcting misspelled words in tweets can enhance the quality of text vectorization and improve model predictions. Increasing the size of the training dataset can also help capture more diverse language patterns present in the test data. Finally, exploring different algorithms for each emotion may uncover models that perform better for specific emotional categories.

In summary, while the regression models show promise in predicting emotional intensity from text, there is room for refinement and optimization to achieve more accurate and robust predictions across all emotional categories. Continued research and experimentation with different strategies can lead to further improvements in model performance.

Deep Learning Model

1.**Data Loading and Preprocessing**:

- Data Loading: The code loads CSV files containing text data for different emotions such as anger, fear, sadness, and joy.

- Data Concatenation: It concatenates the dataframes and combines them into a single dataframe for unified processing.

- Data Filtering: Rows with a label equal to 4 are filtered out, likely because they represent a specific condition or category that is not relevant for the current analysis.

- Text Cleaning: The text data undergoes several cleaning steps, including removing mentions (e.g., Twitter handles), URLs, and numbers. Stopwords, common words that do not contribute much to the analysis, are also removed.

- Tokenization: The cleaned text is tokenized, which involves splitting the text into individual words or tokens for further processing.

- Data Splitting: The data is split into training, validation, and test sets for model training, evaluation, and testing, respectively.

2. Modeling:

- VADModel Definition: The VADModel class is defined, which is a custom model that utilizes the BERT (Bidirectional Encoder Representations from Transformers) architecture for sentiment analysis.

- Loss Function and Optimizer: Mean Squared Error (MSE) is used as the loss function, and the AdamW optimizer is employed for parameter optimization.

- Learning Rate Scheduler: A linear learning rate scheduler is utilized to adjust the learning rate during training.

- Training Loop : The model is trained using a loop that iterates over the training data for multiple epochs. Training progress is monitored, and validation is performed after each epoch to assess model performance.

- Evaluation Metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are calculated to evaluate the model's performance on the test set.

3. Model Saving and Predictio:

- Model Saving: The trained model is saved to a file named "deep_learning_model.pt" for future use.

- Prediction Function: A function is defined to predict VAD scores for given text inputs using the trained model. This function preprocesses the input text, tokenizes it, and feeds it to the model for inference.

- Prediction on Data: The trained model is used to predict VAD scores for the cleaned text data. The predicted scores are appended to the dataframe containing the text data.

- Data Saving: The dataframe with predicted scores is saved to a new CSV file named "predicted_data.csv".

4. Summary:

- The training loop runs for 5 epochs, and the model's performance is evaluated at each epoch.

- The average MSE and loss across all epochs provide insights into the model's overall performance during training.

- MAE and RMSE on the test set quantify the model's prediction accuracy and error.

Comclusion:

The implemented pipeline showcases a comprehensive approach to sentiment analysis, specifically focusing on emotion detection in text data.

Leveraging deep learning techniques, particularly the BERT architecture, the model demonstrates the ability to capture emotional intensity in text effectively.

Evaluation metrics such as MAE and RMSE indicate the model's performance in quantifying emotional content, providing valuable insights for applications such as sentiment analysis, emotion detection, and text understanding.

Overall, the pipeline offers a powerful tool for analyzing and understanding emotional expression in text data, with potential applications in various domains, including social media analytics, customer feedback analysis, and psychological research.