Tommy Lin
METCS777
Final Project Report
10/14/24

Introduction

For my final project, I decided to analyze a dataset called “Emotions” which contains close to 400K Tweets from Twitter that have been labeled a certain emotion. The set of emotions contain “sad”, “joy”, “love”, “anger”, “fear”, and “surprise”. Each emotion corresponds to an integer label from 0 to 5.

I experimented with three different classifiers: NaiveBayes, RandomForestClassifier, and LogisticRegression in hopes of training a model that could accurately predict the emotions of new Tweets. I also seeked a deeper understanding on the subject by analyzing trends and patterns in the data.

Methodology

The project begins with the preprocessing of the data. After importing the dataset as a PySpark DataFrame, preprocessed the raw Tweet text by removing words under three characters long, removed punctuations and numbers, and converted the text to lowercase.

Next, I used the TF-IDF approach to vectorize the text data. To do this, I created a pipeline containing a tokenizer, a stopword remover, a blank string remover, a count vectorizer, and an IDF transformer. I also calculated weights for each class to deal with the class imbalance. This improves scalability and robustness as the dataset grows bigger and potentially more or less imbalanced.

The data was split into a training and testing set to ensure our results and evaluation metrics were trustworthy. My three models were trained on the training set with tested hyperparameters. Models were evaluated using several criteria: precision, recall, F1, accuracy, and a confusion matrix.

Results

Out of the three models I tried, Logistic Regression gave the best results. The second was Naive Bayes, and the third best was Random Forest. I believe the Random Forest performed the worst because it doesn’t handle sparse vectors well. My Tweets were short in length, resulting in sparse vectors as features. The Logistic Regression model after training for 20 iterations with a regulation parameter of 0.1 had an accuracy of 0.8661, a precision of 0.8149, a recall of 0.8753, and an f1 score of 0.8440. I found that increasing the number of iterations and adjusting the regularization didn’t improve the performance.

Discussion

I was happy with how well my model performed. It’s far from perfect, but an F1 score of about 85% is pretty good. I tried adjusting the hyperparameters of each model, but this was the best

performance I could get. Human emotions can be hard to understand, even for humans, so it's impressive that the model could predict emotions this well.

I did further research to see which words most related to each emotion. Sadness was heavily indicated by words like 'homesick', "unimportant", "troubled", and "sentimental". Joy had words like "triumphant", "resolved", "vital", "reassured", and "smug". Fear's top word was "distressed", anger's top word was "resentful", love's top word was "romantic", and surprised's top word was "stunned". I believe the model performed wonderfully in this aspect, because these results are very easy to interpret and make total sense.

I also saw which emotions were being falsely mistaken for other emotions. For instance, if a Tweet is joyful, but the model got it wrong, what emotion did the model most often think it was? For this, sadness was most confused with anger, joy was most confused with love (and visa-versa), anger was most confused with joy, fear was most confused with surprise, and surprise was most confused with joy. These all do make sense as well because emotions are more of a spectrum, and some emotions can be similar to others in how they're expressed.

Conclusion

Overall, I am very satisfied and learned a lot from performing this analysis. Emotions are extremely complex, and when you throw them onto an already complex task like text classification, I wondered how well it would work. I hoped for a model that could reach an F1 score of over 80%-90%, and I was close to hitting that mark.