“Wisdom of the Crowds” or “Let the Best Estimator Win”?

Assignment 4

# Motivation

Each traditional ML algorithm has unique strengths and weaknesses.

This report investigates whether ensembling diverse ML estimators leads to better outcomes or if using the best performer suffices.

Additionally, it explores the role of text preprocessing by studying n-gram ranges, effects of vectorization methods, and the impact of stopwords.

# Preprocessing

## Tokenization

To start, we retain basic text information through basic pre-processiing: lowercasing and tokenization by word and punctuation.

The intent is to retain as much information from the reference text, which is relatively short.

In later pipelining, we investigate the additional dimensions to tokenization, namely:

1. With Stopwords vs wo Stopwords
2. N-Gram / Bi-Gram vs Tri-Grams

## Vectorization

Two different forms of vectorization are investigated:

1. Count Vectorizer, which simply counts occurrences of each word in a document
2. TF-IDF, which highlights unique words by considering frequency of the word in the document and the corpus.

Given the small corpus size, sparsity is not considered to be a significant issue.

# Pipeline Design

## Models

We implement all three traditional ML models for this assignment:

1. KNN - sensitive to local differences, but poor with high dimensions.
2. NaiveBayes - good at handling independent features
3. Logistic Regression - robust for linearly separable classes and relatively robust at high dimensions.

## Pipeline

Six estimators were fitted using combinations of two vectorizers and three estimators.

Due to the light compute requirements, hyperparameters were tuned using a fairly comprehensive GridSearch.

The table in the appendix summarizes the pipeline components and hyperparameters.

## Metrics

We use three main metrics to compare classifier performance: (1) accuracy, (2) F1-score (macro), and (3) confusion matrix. These metrics help evaluate over- or under-fitting, balance between false positives and false negatives, and model strengths, respectively.

# Individual Fitting Observations

### Model Performance, ranked by F1 Score

## Models

1. Logistic Regression performed the best, possibly due to input features having linear correlations with the target variable. This suggests a strong relationship between certain words and specific emotions.
2. NaiveBayes tended to overfit, which could be attributed to its assumption of feature independence. To alleviate overfitting.
3. KNN's performance remained consistent across different document vectorization formats. This can be attributed to its inherent property of being sensitive to local differences and using neighboring data points for classification.

## Test vs Train F1

1. Tf-idf vectorization outperformed Count Vectorizer, which tended to overfit. Tf-idf might be superior because it captures the importance of rare words better by weighing their frequencies within and across documents.
2. The overfitting observed with Count Vectorizer could be addressed by implementing feature selection techniques or dimensionality reduction.

## F1 vs Accuracy

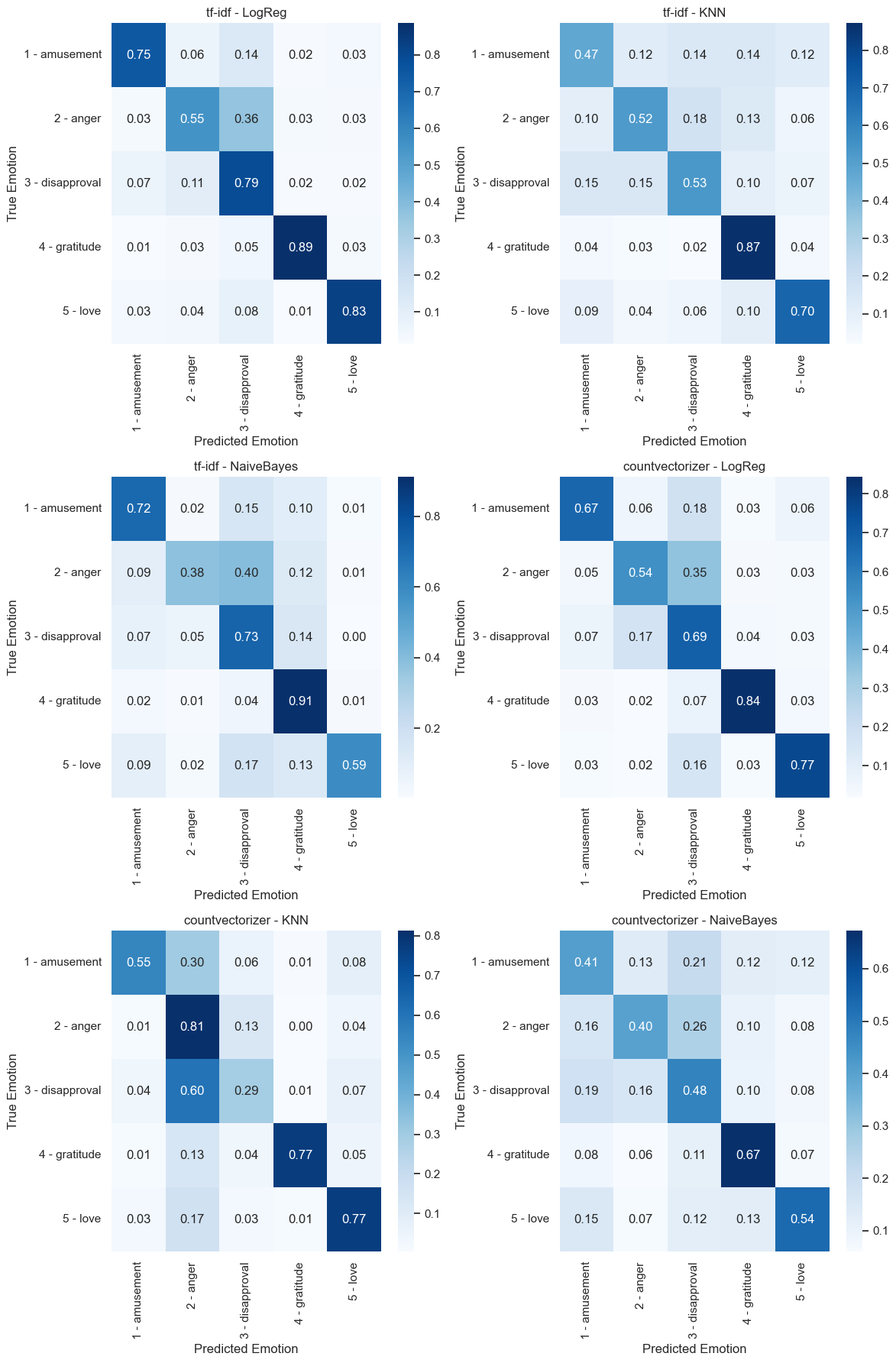
1. A general agreement between F1 score and accuracy suggests that the class distributions in our dataset are balanced. Consequently, both metrics can be reliably used for model evaluation.

## Vectorizer Hyperparams

|  | N-Grams | Stopword Removal? |
| --- | --- | --- |
| Tf-idf  + LogReg | 1~3 | N |
| Countvectorizer  + LogReg | 1~2 | Y |
| Tf-idf  + NaiveBayes | 1~3 | Y |
| CountVectorizer  + KNN | 1~3 | Y |
| Tf-idf  + KNN | 1~3 | N |
| CountVectorizer  + NaiveBayes | 1~3 | N |

1. Tf-idf models mostly retained stopwords, confirming our understanding that tf-idf would weigh out repetitive words by considering their frequencies in individual documents and the entire corpus.
2. In contrast, CountVectorizer models mostly removed stopwords, indicating that their presence might introduce noise.
3. Interestingly, LogReg with Count Vectorizer optimized for 1-2 n-grams, possibly because it captures relevant word pairs that provide more context for emotion classification.

## Confusion Matrices



### Sample Confusion Matrix (see appendix for all)

1. All classifiers had trouble distinguishing between '2 - anger' and '3 - disapproval'. Empirically, this can be attributed to these emotions having overlapping word features, making it harder for the model to distinguish between them.
2. All classifiers had relative success predicting “4 - gratitude’ and ‘5 - love’. Empirically, these emotions have unique words attached to them, which is surfaced by the method of vectorization.
3. “1- Amusement” generally sits in the middle ground of success. It is a subtle emotion, but out performs having to distinguish between negative emotions.

# Ensembling Observations

For the final validation with test.csv, 3 final models were fitted:

**6-Model Ensemble**

All six models, ensemble and cross-valided for voting-type. A “hard” vote produced a better score. (0.751 vs 0.730)

**Complimentery Ensemble**

“Tf-idf with LogReg” was paired with “CountVectorizer with LogReg” because of complimentary confusion matrices on predicting “2-anger”.

**Single Logistic Regression Model**

A single model with the best F1 and Accuracy score from GridSearch.

## Final Outcome (ranked by score)

LogReg Model: **0.776\***

6-Model Ensemble: **0.751**

Complimentary Ensemble: **0.664**

The logreg model outperformed all efforts at ensembling.

# Learnings

## Ensembling

The wisdom of crowds, or ensembling, relies on diversity.

WIthin the boundaries of this assignment and assuming a non-neural approach, the biggest area where this can be improved is in creating different feature extraction and vectorization techniques.

Aspects that can be considered include lemmatization, pos-tagging, NER tokenization and CBOW embeddings.

Diverse models are more likely to capture different aspects of the data, thus providing complementary strengths.

## Diagnostics

The confusion matrix was very useful tool in diagnostics. It clearly shows trends in strengths and weaknesses.

## For example, in this assignment, it would help to create some handcrafted features to distinguish the anger and disapproval (e.g. number of “!”s, presence of words like “sigh” or “...”.)

# 

# Appendix

Table of Hyperparameters

| **Vectorizers** | **TF-IDF**  Ngram range [(1,1)(1,2)(1,3)]  Stopword Removal:[Yes, No]  *\*wo standardization* |
| --- | --- |
| **CountVectorizer**  Ngram range: [(1,1)(1,2)(1,3)]  Stopword Removal: [Yes, No]  *\* w standardization* |
| **Estimators** | **LogReg**  C: [0.1, 1, 10]  Regularization: L2 |
| **KNN**  N: [3, 5, 8, 13]  Weights: [uniform, dist] |
| **NaiveBayes**  Alpha: [0.1, 1, 10] |

# 

# 

# Mybe Later

Three main metrics were logged to can use to compare the performance of classifiers are:

1. Accuracy: It is the ratio of correctly classified instances to the total instances in the dataset. This metric is intuitive and easy to interpret.
2. F1-score: A single metric that balances the trade-off between false positives and false negatives.
3. Confusion Matrix: This is useful to evaluate the different strengths of the models.