# Decision Trees vs Neural Networks for Classification of Emotions from Text

Ettore Ricci

September 13, 2024

## 1 Introduction

In today's digital world, the ability to interpret emotions from written text has gained importance for a variety of applications, including customer feedback analysis and mental health monitoring. Emotion classification, a challenging task within natural language processing, focuses on identifying the emotional tone within text.

This work compares the performance of three different models for emotion classification: Decision Trees, Random Forests and Neural Networks. The models were trained on a dataset of online messages labeled with 7 different emotions: Happiness, Sadness, Anger, Worry, Love, Surprise and Neutral.

## 2 Data

The dataset used was a mix of the CrowdFlower dataset[1] and the Emotions dataset[2]. The CrowdFlower dataset contains around 40,000 tweets labeled with 13 different emotions. The Emotions dataset contains around 400,000 tweets labeled with 6 different emotions.

### 2.1 Exploratory data analysis and preparation

Some emotions that are present in the CrowdFlower dataset were remapped because they were not present in the Emotions dataset. The emotions and relative mappings are shown in Table 1.

| Emotion | Mapping |
|---|---|
| Happiness | |
| Sadness | |
| Anger | |
| Worry | |
| Love | |
| Surprise | |
| Neutral | |
| Fun | Happiness |
| Relief | Happiness |
| Hate | Anger |
| Empty | Neutral |
| Enthusiasm | Happiness |
| Boredom | Neutral |

Table 1: Emotion mappings, when the mapping is empty the emotion was not remapped.

Some instances were removed because they were duplicates. There were 23,163 duplicates and the dataset was reduced to 433,646 instances.

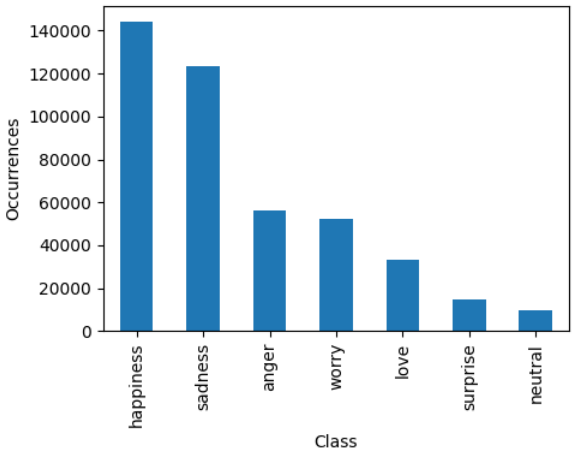


Figure 1: Class distribution of the full dataset

As shown in Figure 1, the dataset is imbalanced.

The most common emotion is Happiness, and the least common is Neutral. This is because the larger dataset (Emotions) does not contain the Neutral class.

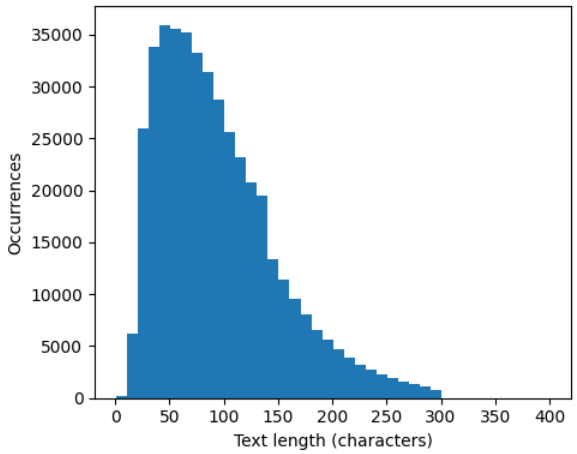


Figure 2: Text length distribution of the full dataset

The dataset was split into a training set and a test set with a 80% - 20% ratio.

### 2.2 Preprocessing

Multiple kinds of preprocessing were tested:

- *Bag of words (BoW)*
- *Word embeddings (WE)*
- *GloVe embeddings (GE)*

For the BoW and WE preprocessing, the text was first cleaned with a regular expression that removes urls, mentions and symbols. Then the text was tokenized and stemmed with the Snowball stemmer. Both **TFIDF** and **binary** encodings were tested for the BoW preprocessing. The GE preprocessing was done by applying the same cleaning regular expression but without stemming. The vectors used were the 6B 50d vectors from GloVe[3]. The main difference between the WE and GE preprocessing is that the WE vectors were trained on the dataset, while the GE vectors were pre-trained on a large corpus of text.

## 3 Models

Three different categories of models were considered:

- *Decision Trees*
- *Random Forests*
- *Neural Networks*
  - *Feedforward Networks*
  - *LSTM Networks*

Decision Trees and Random Forests both use the BoW preprocessing with **binary** encoding. Feedforward Neural Networks use the BoW preprocessing with **TFIDF** encoding. LSTM Networks were tested with both the WE and GE preprocessing.

### 3.1 Imbalanced Dataset

To handle the imbalanced dataset, a class weight was assigned to each class. The weight for a class $c$ was calculated as $W_c = \frac{|D|}{|C||D_c|}$ where $|D|$ is the size of the dataset, $|C|$ is the number of classes and $|D_c|$ is the number of samples in class $c$. This weight was used in the criterion of the Decision Trees and Random Forests models, and in the loss function of the Neural Networks models.

### 3.2 Model Selection

To find a good hyperparameter configuration for each model, a random search was performed. The search was done with 10 iterations for each model (Decision Trees, Random Forests and Neural Networks). Each configuration was evaluated with a 6-fold cross-validation on the training set. Two subsequent random searches were performed, the second one narrowing the search space around the best configuration found in the first search. The search spaces for each model are shown in Table 2, Table 3 and Table 4 for the first search, and in Table 5, Table 6 and Table 7 for the second search.

| Hyperparameter | Search Space |
|---|---|
| `criterion` | {gini, entropy, log_loss} |
| `splitter` | {best} |
| `max_depth` | {10, 100, None} |
| `min_impurity_decrease` | {0.01, 0.0001, 1e-06, 1e-08, 0} |
| `class_weight` | {balanced} |

Table 2: Decision Trees hyperparameter search space for the first search.

| Hyperparameter | Search Space |
|---|---|
| n_estimators | {10, 50, 100} |
| criterion | {gini, entropy, log_loss} |
| max_depth | {10, 100, None} |
| min_impurity_decrease | {0.01, 0.0001, 1e-06, 0} |
| n_jobs | {-1} |
| class_weight | {balanced} |

Table 3: Random Forest hyperparameter search space for the first search.

| Hyperparameter | Search Space |
|---|---|
| network | {ff_tfidf, lstm_embeddings} |
| base_size | {8, 16, 32} |
| depth | {1, 2, 3, 4} |
| epochs | {10, 15} |
| dropout | {0.5} |
| batchnorm | {True, False} |
| batch_size | {32} |
| lr | {0.01, 0.001} |
| optimizer | {adam} |

Table 4: Neural Networks hyperparameter search space for the first search. While using the `ff_tfidf` network, if `batchnorm` is set to `True`, the `dropout` hyperparameter is set to 0. When using the `lstm_embeddings` and `lstm_glove` networks, the `batchnorm` hyperparameter is set to `False`.

| Hyperparameter | Search Space |
|---|---|
| criterion | {gini, log_loss} |
| splitter | {best} |
| max_depth | {1000, None} |
| min_impurity_decrease | {1e-05, 1e-06, 1e-07} |
| class_weight | {balanced} |

Table 5: Decision Trees hyperparameter search space for the second search.

| Hyperparameter | Search Space |
|---|---|
| n_estimators | {100, 125, 150} |
| criterion | {gini, log_loss} |
| max_depth | {1000, None} |
| min_impurity_decrease | {1e-05, 1e-06, 1e-07} |
| n_jobs | {-1} |
| class_weight | {balanced} |

Table 6: Random Forest hyperparameter search space for the second search.

| Hyperparameter | Search Space |
|---|---|
| network | {ff_tfidf, lstm_embeddings} |
| base_size | {16, 32} |
| depth | {2, 3} |
| epochs | {10, 15} |
| dropout | {0.5} |
| batchnorm | {False} |
| batch_size | {32} |
| lr | {0.001} |
| optimizer | {adam} |

Table 7: Neural Networks hyperparameter search space for the second search. The same rules described in Table 4 apply.

Accuracy was used as the metric to evaluate the models and we can see the results in Figure 3.

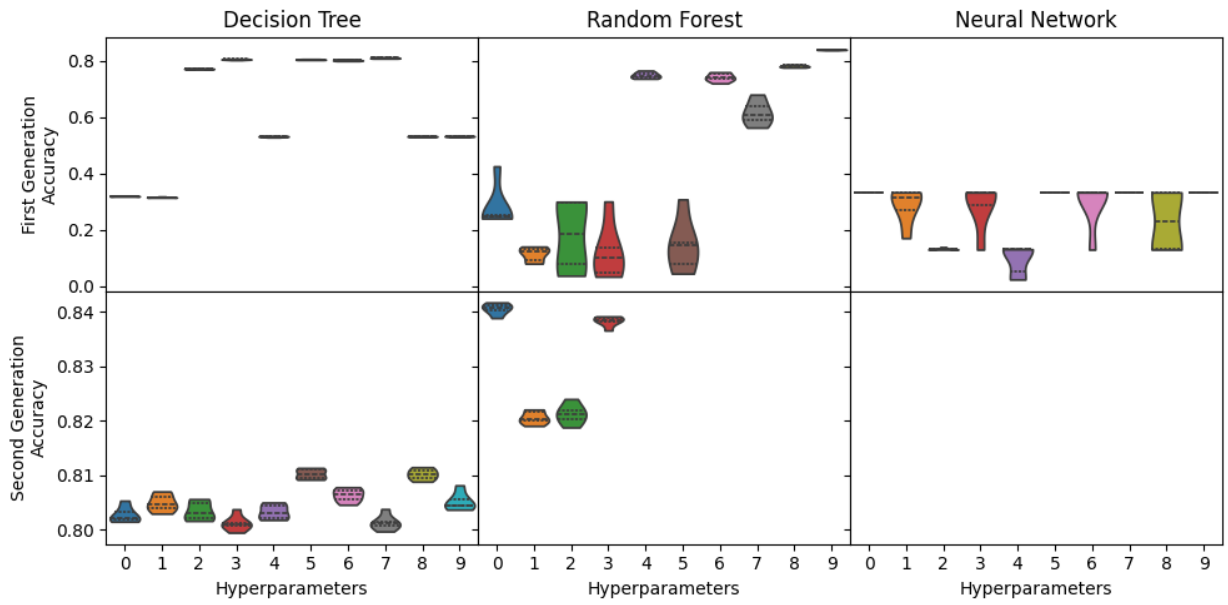


Figure 3: Accuracy distribution of each model configuration tested during the model selection process.

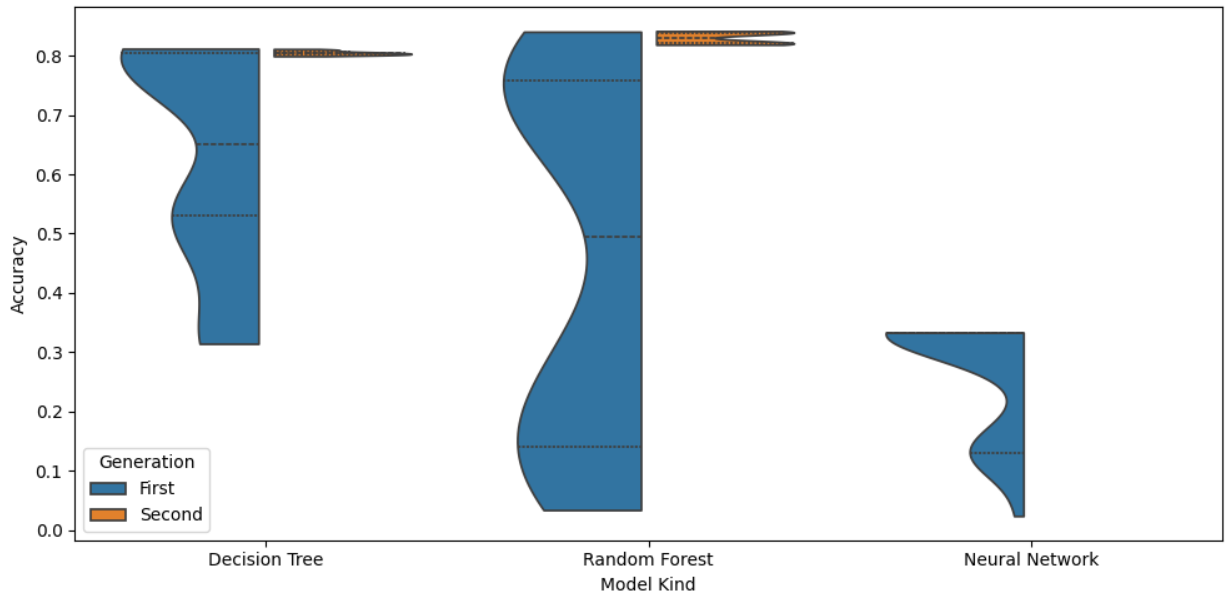


Figure 4: Accuracy distribution of each model category.

The best model was selected based on the average accuracy obtained in the cross-validation. The accuracy distribution of this model was compared to the other models with a Wilcoxon signed-rank test, some models were not significantly different ($p > 0.05$) from the best model and they are listed in Table 8.

| Model | Accuracy |
|---|---|
| `random_forest-G1-0` | 84.06% |

Table 8: Best model (first row) and models not significantly different from the best model.

# 4 Results [4]

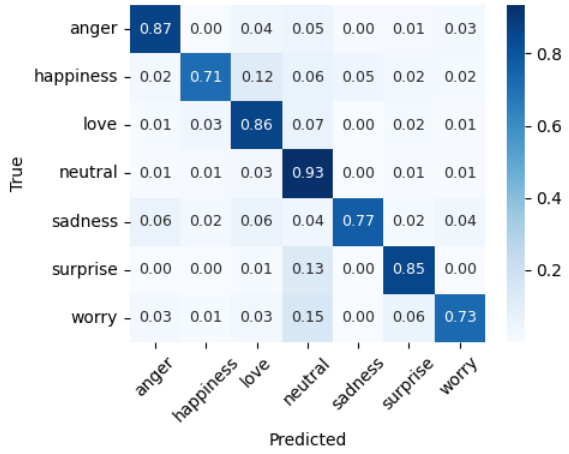


Figure 5: Confusion matrix of the best decision tree model (decision_tree-G0-3).

# References


[1] C. Van Pelt and A. Sorokin, "Designing a scalable crowdsourcing platform," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 765–766, 2012.

[2] N. Elgiriyewithana, "Emotions." https://www.kaggle.com/dsv/7563141, 2024.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[4] E. Batbaatar, M. Li, and K. H. Ryu, "Semantic-emotion neural network for emotion recognition from text," *IEEE Access*, vol. 7, pp. 111866–111878, 2019.