Emotion classification – ML project

Michał Wietecki

Barbara Seweryn

Elizaveta Mokhova

Karolina Buceneka

1. INTRODUCTION

In this project we focused on developing a machine learning model that would classify short texts (for example comments on social media) as various emotions (happiness, sadness, surprise, neutral etc.).

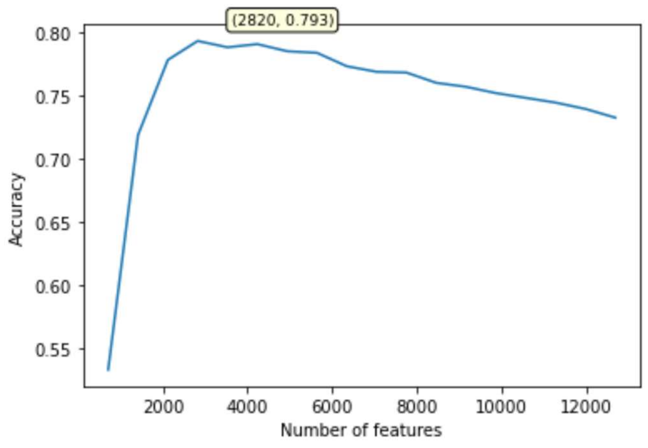
2. DATA PREPROCESSING AND EXPERIMENTATION WITH VECTORIZERS

Our dataset in the beginning only consisted of two columns: the text column and the target class (emotion) column. To actually be able to work with this dataset and for it to be understandable for a ML model, we had to transform it into numeric values. For this we used two text vectorization techniques:

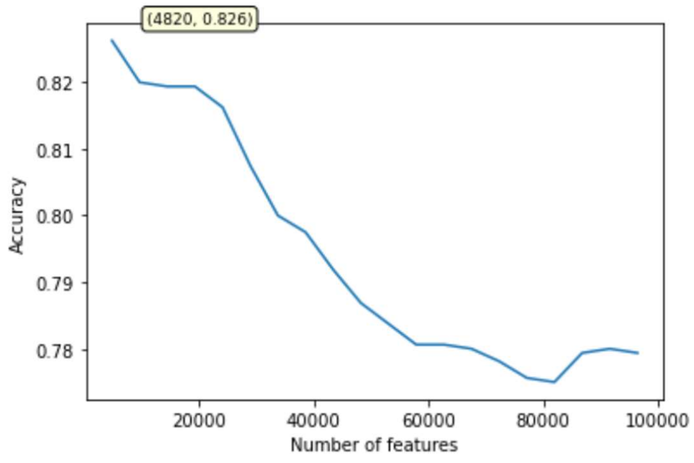
- Count Vectorizer:
    a. BOW (bag of words) – Count Vectorizer for single words. This methods creates a column for every word found in the text column and for each row (each text instance) it counts for many times a word (for which the column is dedicated to) appeared. So for example for "I have a dog and a cat", the values would be 1 for "I", "have", "dog", "and", "cat, 2 for "a" and 0 for everything else.
    b. N-gram – same thing, but depending on the ngram_range = (a,b) parameter, this vectorizer will consider phrases, that include from a to b words (not just single words like in BOW). The rest of the process is similar to BOW.
- Tfidf Vectorizer - weighs word frequency by how common it is across all document. Reduces the impact of common but less informative words. Multiplies two values:
    - TF (Term Frequency) = Frequency of a term in a document.
    - IDF (Inverse Document Frequency) = Inverse of how many documents contain the term.

We first experimented with these vectorizers, using a simple MultinomialNB model, just to see how they would perform. We experimented with different ngram_range parameters, stop_words parameter and max_features parameter (to reduce dataset dimensions). The best performance was found for :

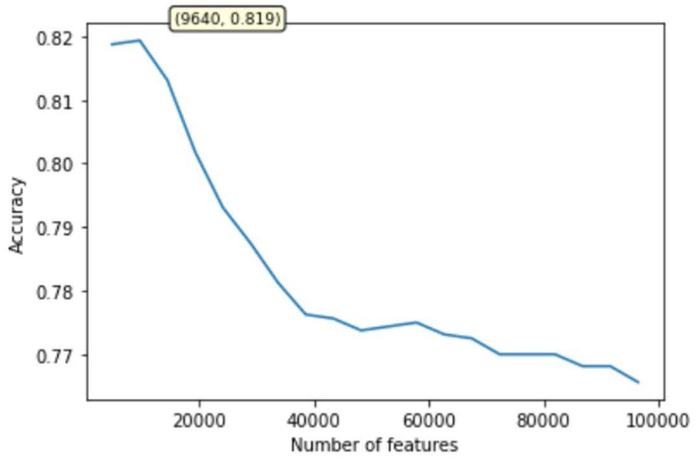
- a simple single-word BOW for 2820 out of 14000 features (about 20%), with accuracy with the simple MultinomialNB model reaching 0,793.



- CountVectorizer with ngram_range = (1,2), about 5% of the features and stop_words = ‘english’, with accuracy with the MultinomialNB reaching 0.826



- Tfidf Vectorizer with ngram_range = (1,2),about 5% of the features and stop_word = ‘english’, sub_tf =True, with accuracy with the MultinomialNB reaching 0.819



3. MODEL SELECTION AND EXPERIMENTATION

We experimented with the following models:

1. MultinomialNB
2. SVM
3. XGB

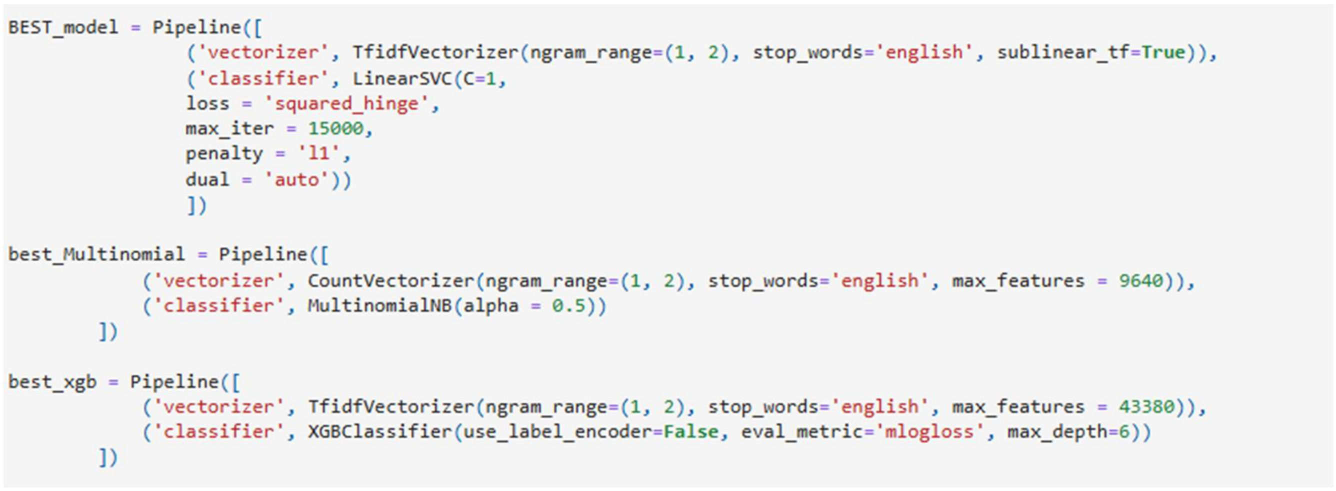
For those models we did a lot of experimenting, looking for different parameters will influence their performance. The most important parameter that we focused on was the percentage of features the model should use. This experimentation was extremely important, since we wanted to focus on strong dimensionality reduction, but as the same time not decrease the model performance.

In most cases a small percentage of features used (about 5%) was getting us the best results, which was a very good information, since it provided a huge reduction of dimensions.

For each model we also tried to experiment with parameters characteristic for this model especially (for example max_depth or XGB or different loss functions for SVM).

All this experimenting can be found in the classification.ipynb file.

Finally the best performance was found when working with the following pipelines (combinations of different vectorizers and models):



```
BEST_model = Pipeline([
                ('vectorizer', TfidfVectorizer(ngram_range=(1, 2), stop_words='english', sublinear_tf=True)),
                ('classifier', LinearSVC(C=1,
                loss = 'squared_hinge',
                max_iter = 15000,
                penalty = 'l1',
                dual = 'auto'))
                ])

best_Multinomial = Pipeline([
            ('vectorizer', CountVectorizer(ngram_range=(1, 2), stop_words='english', max_features = 9640)),
            ('classifier', MultinomialNB(alpha = 0.5))
        ])

best_xgb = Pipeline([
            ('vectorizer', TfidfVectorizer(ngram_range=(1, 2), stop_words='english', max_features = 43380)),
            ('classifier', XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', max_depth=6))
        ])
```

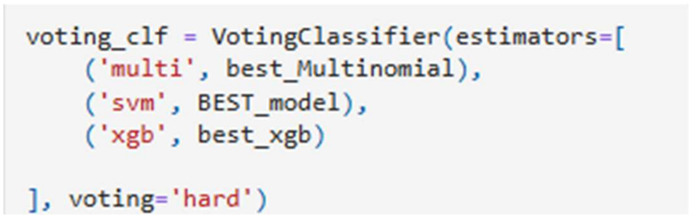
The accuracies for these models were the following:

VECTORIZER + MODEL	ACCURACY
TfidfVectorizer + SVM	0.924
CountVectorizer + MultinomialNB	0.832
TfidfVectorizer + XGB	0.909

4. FINAL MODEL

In order to improve the accuracy and stability of the model we decided to go even further and create an ensemble model, using these three models and have them vote on the final class label.

To do this we created the final model, which used hard voting (every model has the same weight) with these 3 models.



```
voting_clf = VotingClassifier(estimators=[
    ('multi', best_Multinomial),
    ('svm', BEST_model),
    ('xgb', best_xgb)

], voting='hard')
```

This is the final model we decided to go with. The accuracy we managed to achieve was **0.9175.**

We tested the model accuracy using cross validation with 5 folds and got the following results:

Cross-validated accuracies: [0.91484375 0.90546875 0.91953125 0.896875   0.89375   ]

Mean accuracy: 0.9061