Text mining project

Data Set

In our project we decided to analyze The Blog Authorship Corpus which consists of the collected posts of 19,320 bloggers gathered from blogger.com in August 2004. The corpus incorporates a total of 681,288 posts and over 140 million words - or approximately 35 posts and 7250 words per person.

Each blog is presented as a separate file, the name of which indicates a blogger id and the blogger's self-provided gender, age, industry, and astrological sign. (All are labeled for gender and age but for many, industry and sign is marked as unknown.)

We know that each of the bloggers in the corpus can fall into one of three age categories:

- ages 13-17
- ages 23-27
- ages 33-47

As a classification task we decided to try predicting the age of the blogger based on their way of writing.

Text extraction

We use the BeautifulSoup library to parse XML files containing blog data. It extracts text and age information from the XML files, saving post's text and age in two lists. At the and by using function in pickle library we serialize those data to do not need to execute this part of code each time

Then we change age into age groups - categorize them into broader ranges. For example, if someone is between 13 and 17 years old, their new label becomes '13-17'. Similarly, there are ranges for 23-27 and 33-47. If someone's age doesn't fall into any of these ranges, they get labeled as 'Other'. We did this to simplify the data and make classification easier to perform. The new age ranges provide a more general view of age distribution, which can be handy for finding features typical for a given age group.

Text pre-processing

We used the Natural Language Toolkit (nltk) library to perform text preprocessing on a list of blog posts. The preprocessing steps include cleaning the text, tokenizing it, removing stop words and lemmatizing the remaining words. Now the code preprocesses the raw text data by removing HTML tags, special characters and stop words. The final output is a list of preprocessed blog post texts ready for further analysis or modeling.

Additionally, we leveraged scikit-learn which is a machine learning library in Python. The fastText library was used for text embedding. We decided to use this technique by using subword information to capture more nuances of language needed to distinguish labels. After embedding, each word in every element of the list we calculated an average embedding for each post. We employed it to convert text data into numerical representations known as word embeddings. These embeddings capture semantic relationships between words and allow us to represent textual information in a format suitable for machine learning tasks.

Techniques that we used

- Stopword removal: we elimitaded common words (stopwords) that often carry little semantic meaning and improving computational efficiency.
- Lemmatization: we reduced words to their base or root form, capturing the essential meaning of the word.
- Tokenization: we broke down the text into individual words (tokens)
- Removing Special Characters and Numbers: we reduced noise in the text data by excluding non-alphabetic characters and digits.
- Removing HTML Tags: we eliminated HTML tags is crucial to focus on the actual text.
- Lowercasing: we converted text to lowercase ensures uniformity and prevents duplication of words based on case differences.

Collectively, these techniques helped us to prepare the text for analysis by eliminating unnecessary elements such as HTML tags and special characters. The text became more normalized, facilitating further analysis in the context of our task, specifically: Support Vector Classifier, Random Forest, Neural Network and Decision Tree.

Classification and results

We have decided to employ variety a variety of classification techniques co compare them and to find one that works the best, specifically: Support Vector Classifier (SVC), Random Forest, Neural Network (MLP Classifier), and Decision Tree. We chose accuracy, precision, recall, and F1-score as evaluation metrics to comprehensively assess each model's performance, ensuring a balance between overall accuracy and the ability to correctly identify specific classes. Here are results that we achieved:

Metrics for SVC:

Accuracy: 0.7054865424430642

Precision: 0.622960111484981

Recall: 0.7054865424430642

F1 Score: 0.6522651349941868

Metrics for Random forest classifier:

Accuracy: 0.6801242236024845

Precision: 0.675247011083086

Recall: 0.6801242236024845

F1 Score: 0.6535456589595977

Metrics for Multi-Layer Perceptron Classifier:

Accuracy: 0.7631987577639752

Precision: 0.7561030284626843

Recall: 0.7631987577639752

F1 Score: 0.755503704607101

Metrics for decision tree classifier:

Accuracy: 0.5654761904761905

Precision: 0.570682676300026

Recall: 0.5654761904761905

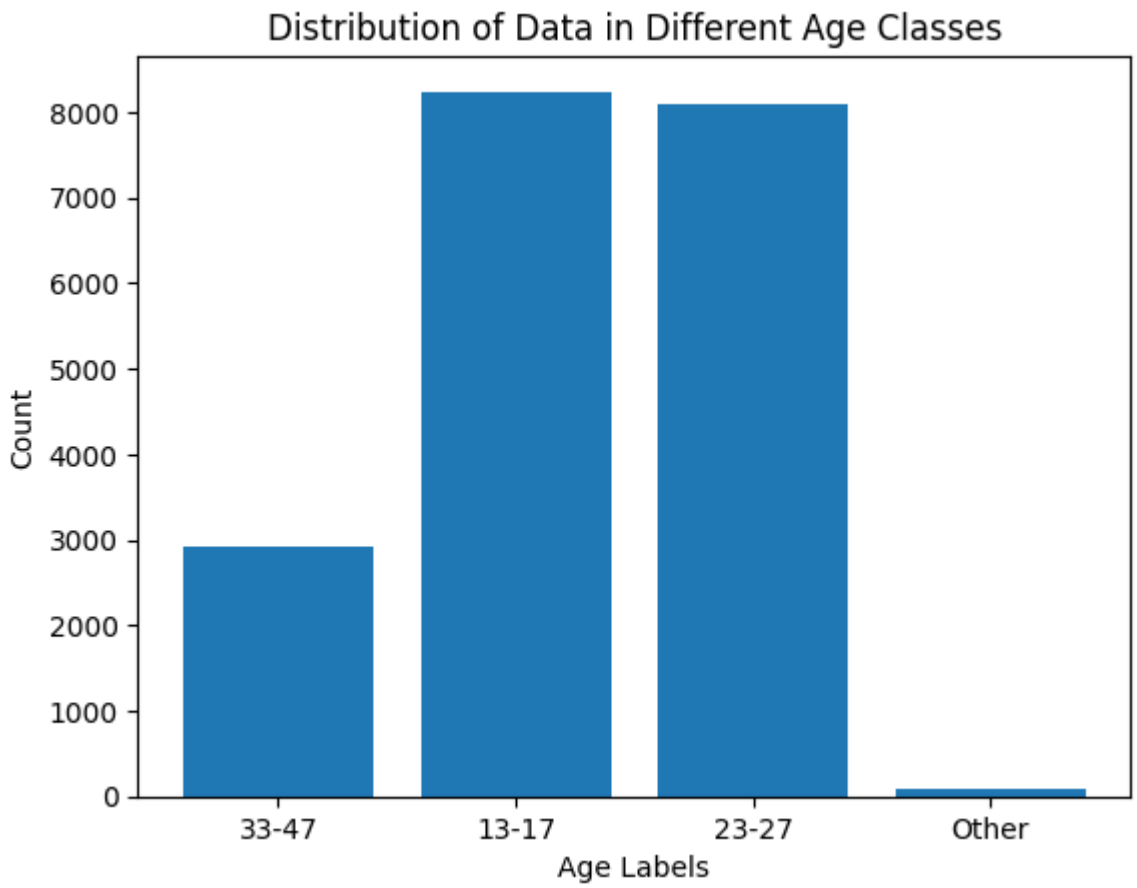
F1 Score: 0.5678800818397249

Analysis of results:

We can see that the Multi-Layer Perceptron Classifier performed the best. We can assume that is because it's well-suited for complex patterns in the data. This type of neural network is good at finding subtle relationships and trends, which is probably why it scored highest in accuracy, precision, recall, and F1 score.

The Support Vector Classifier (SVC) and Random Forest Classifier had moderate results. SVC is great for high-dimensional data, but it might not have been flexible enough to capture all the nuances in our dataset. Random Forest is strong in preventing overfitting and can handle various data types, but it may not have been as effective in pinpointing the specific patterns present in our data.

The Decision Tree Classifier didn't perform as well, likely because it's a simpler model. While easy to understand and interpret, Decision Trees can struggle with more complex or nuanced data patterns, leading to lower overall performance in our case. This suggests that for our specific task, which may involve intricate relationships within the data, a more sophisticated approach like the Multi-Layer Perceptron is needed.


Distribution of Data in Different Age Classes
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
33-47
13-17
23-27
Other
Age Labels