STOCKHOLM UNIVERSITY
Department of Computer and Systems Sciences
October 27, 2023
Assignment results

Assignment 4

Principles and Foundations of Artificial Intelligence

Machine learning: Decision trees

version 1.0

Name N. Chentre, N. Boekelman
Username nech2600, nobo0291

Graders
Unknown

Question 4:

We get very unprecise predictions (Accuracy: 0.7609, F1-score: 0.8353)

After exploring the code, we noticed that in the 'predict' method of the random forest, instead of picking the most predicted label, it was picking one of the predictions randomly. After modifying this code for it to work properly and after splitting the training set into a training and a test set in order to evaluate the model's generalization capabilities correctly, the following score was achieved: (Accuracy: 0.7306, F1-score: 0.7975). The loss in precision can be explained by the separation of the training and the testing set.

Question 5:

After a grid search ran on the following parameters space:

- `bias` in [.01, .05, .1, .2, .25, .3, .35, .4, .5, .7, .9]
- `max_depth` in [2, 3, 4, 5, 6, 7, 8, 9, 10]

We found that the best parameters for the unseen test set are with a `bias` of 0.4 and a `max_depth` of 10: (Accuracy: 0.7333, F1-score: 0.8040)

Question 6: NLP

For the preprocessing of the text, we decided to implement a preprocessing function that will make it easier to tune the preprocessing stage as we try different methods. You can find below the different parameters and models we used for some of the attempts we made (Attempt 0 being the one before any modification of the script) as well as the score they yeild (see Table 1).

- Attempt 0: Tokenization (whitespace split) -> Bag of words (token count).
- Attempt 1: Tokenization (*nltk TreeBankTokenizer()*) -> Stopwords removal (*nltk corpus StopWords*) -> Punctuation removal (List of symbols) -> Bag of words (token frequency).
- Attempt 2: Tokenization (*nltk WordPunctTokenizer()*) -> Stopwords removal (*nltk corpus StopWords*) -> Punctuation removal (List of symbols) -> Bag of words (token frequency).
- Attempt 3: Tokenization (*nltk WordPunctTokenizer()*) -> Stemming (*nltk SnowBallStemmer()*) -> Bag of words (token frequency).
- Attempt 4: Tokenization (*nltk WordPunctTokenizer()*) -> Stemming (*nltk SnowBallStemmer()*) -> Stopwords removal (*nltk corpus StopWords*) -> Punctuation removal (List of symbols) -> Bag of words (token frequency).
- Attempt 5: Tokenization (*nltk WordPunctTokenizer()*) -> Lemmatization (*nltk pos_tag() -> nltk WordNetLemmatizer()*) -> Stopwords removal (*nltk corpus StopWords*) -> Punctuation removal (List of symbols) -> Bag of words (token frequency).
- Attempt 6: Tokenization (*nltk TreeBankTokenizer()*) -> Lemmatization (*nltk pos_tag() -> nltk WordNetLemmatizer()*) -> Stopwords removal (*nltk corpus StopWords*) -> Punctuation removal (List of symbols) -> Bag of words (token frequency).

	Unique Tokens	Accuracy	F1-Score
Attempt 0	50264	0.6500	0.6698
Attempt 1	41197	0.6650	0.6683
Attempt 2	36274	0.6375	0.6437
Attempt 3	23726	0.6475	0.6466
Attempt 4	23565	0.6825	0.6833
Attempt 5	29714	0.7050	0.7150
Attempt 6	35442	0.7300	0.7379
Attempt 7	35267	0.7350	0.7440
Attempt 8	29242	0.7125	0.7229

Table 1: Results of various attempts

- Attempt 7: Tokenization (*nltk word_tokenizer()*) -> Lemmatization (*nltk pos_tag()* -> *nltk WordNetLemmatizer()*) -> Stopwords removal (*nltk corpus StopWords*) -> Punctuation removal (List of symbols) -> Bag of words (token frequency).

We decided to modify the way the bag of words model works in order to avoid the size of the text having an effect on the results: before, the number of times the token appeared in the text was the value used in the bag of words, and we now use the frequency of the said word in the text. Therefore the size of the text does not matter compared to how often the said word appears. (for example, a review of 500 words containing 4 times the word 'bad' could be less negative than a review of 50 words containing it 3 times).

Tokenizer: Our final choice went to `word_tokenize()` from `NLTK`. This tokenizer is perfect as it does not perform too many operations, is not too specific, and works great with different kinds of problems (contractions, decimal numbers, currencies symbols, composed names...) which lets us build more specific preprocessing afterward.

StopWords: We opted for the removal of the stopwords after POS tagging as this would only add irrelevant information that the model might be tempted to use as a decision point when those tokens/words convey little to no meaning.

Punctuation: We decided to remove all punctuation after POS-tagging as punctuation only gives part of speech or contextual information and we wish to get rid of contextual information.

Term normalization: We decided to go for lemmatization as this option offers more types (types are the distinct tokens that can appear when tokens are occurrences of a type) and preserves the meaning of these. It is more expensive to put in place as we need to do Part-Of-Speech tagging in order to do lemmatization but as our dataset is not extensive, the cost can be disregarded. We got the best results with the `WordNet` lemmatizer. We tried stemming but none gave results as good as `WordNet`, even the `SnowBall` stemmer.

With all abovementioned modifications, we raised accuracy from **0.6500** to **0.7350** and F1-score from **0.6698** to **0.7440**.