**Documentation for argumentation mining:**

Feature engineering:

Feature engineering entails combining data mining techniques with domain expertise to derive features from raw data. Feature engineering is a type of applied machine learning that is used to enhance the performance of machine learning algorithms.

The aim of feature engineering is to achieve two main objectives:

* Creating an input dataset that is consistent with the machine learning algorithm and best matches it.
* Improving machine learning models' efficiency.

We used 7 key features for feature engineering in this software. These are the follows:

1. NLP English words - Determine which natural language data is present.
2. Punctuation - It gets rid of punctuation marks.
3. Spacing - It removes all spaces from the sentence.
4. URL removal – It eliminates all URLs from the sentence.
5. Stop Words - It removes all unnecessary words from the sentence (for example, in, the, a, and so on).
6. Stemming - It can help to change the words into their root form, for example, loves becomes love.
7. Part of Speech (POS) - A POS Tagger is a software piece that reads text, allocates parts of speech to each word, such as nouns, verb, adjectives, etc.
8. Bag of Words - A bag-of-words is a text representation that defines the frequency in which words appear in a document. It entails two steps:
   * A list of terms that are well-known.
   * A metric for determining the presence of well-known terms.

Classification:

The problem of classifying text data based on its content is known as text classification. We are categorizing the data into two labels: 1 or 0, with 1 indicating a claim and 0 indicating no claim. As we can see, binary classification is required, so we had to select a classifier accordingly. Hence, Support Vector Machine (SVM) and Multinomial Naïve Bayes (MultinomialNB) were the two classifiers we tested. And we found that the SVM is very quick, gives a high preview score and has no overfitting problems. Also, for this dataset, the Naive Bayes algorithm is extremely fast. SVM produced the highest F1 score of any of the combinations we tested.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MODEL | ACCURACY | PRECISION | RECALL | F1 SCORE |
| SVM (kernel = “linear”) + FE + BOW | 0.59 | 0.4 | 0.29 | 0.33 |
| MultinomialNB + FE + BOW | 0.58 | 0.41 | 0.33 | 0.36 |
| SVM (kernel = “linear”) + FE + BOW + TF-IDF | 0.64 | 0.51 | 0.21 | 0.30 |
| MultinomialNB + FE + BOW + TF-IDF | 0.63 | 0.44 | 0.0634 | 0.11 |
| SVM (kernel = “linear”) + FE + BOW + TF-IDF + Stemming | 0.67 | 0.61 | 0.29 | 0.34 |
| MultinomialNB + FE + BOW + Stemming | 0.61 | 0.45 | 0.38 | 0.39 |
| MultinomialNB + FE + BOW + Stemming + POS | 0.62 | 0.47 | 0.37 | 0.41 |
| SVM (kernel = “linear”) + FE + BOW + Stemming + POS | 0.62 | 0.46 | 0.34 | 0.39 |
| SVM (kernel = “linear”, gamma = ”scale”, C = 100.0) + FE + BOW + Stemming + POS | 0.62 | 0.48 | 0.47 | 0.47 |

SVM = Support Vector Machine

FE = Feature Engineering such as removal of punctuation, spaces, URL and stop words

BOW = Bag of Words

TF-IDF = Term Frequency – Inverse Document Frequency

POS – Part of Speech

Generating Output File:

After classification is completed, we take the predicted values and map it to the text id and write it to *result.json* file

Steps:

Run this command:

*python arg\_mining.py -x train-data-prepared.json -y val-data-prepared.json*

Training Data: train-data-prepared.json

Validation Data: val-data-prepared.json

Output generated by the classifier in: result.json

Run this command:

*python eval.py -t val-data-prepared.json -p result.json*