Sentiment Analysis & Predictions

Classification models

Adam Kacprzycki – Warsaw University of Technology - Data Science

Project purpose:

Using classification models to predict the sentiment from a given text.

Intro:

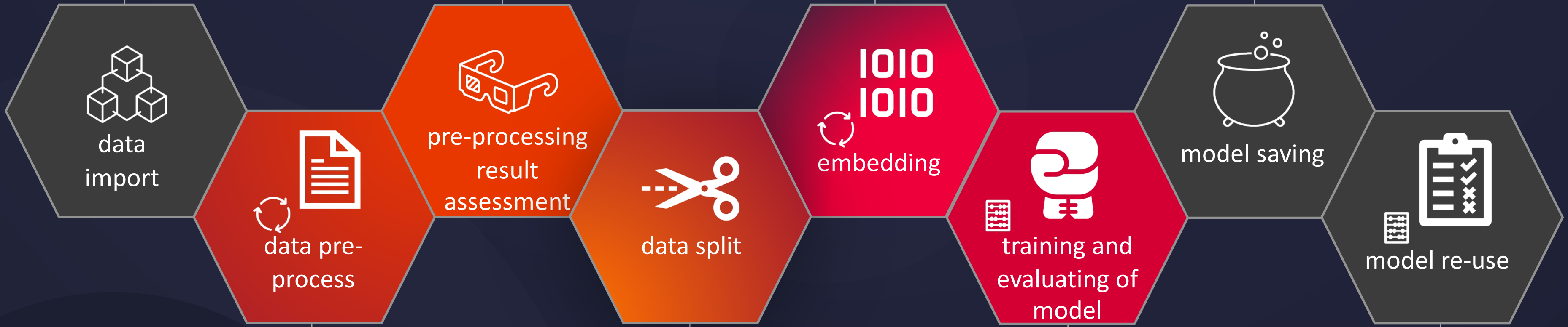
Sentiment analysis (*or opinion mining)* is a natural language processing (NLP) technique used to determine the emotional state of a given author and the impact that author's text may have on the emotions of others.

In practice, we can come across ready-made algorithms or solutions to determine whether a given text has a positive or negative tone (eg. VADER - a lexicon for Social Media or Textblob).

Areas of use: e-commerce, PR (politicians, political parties), etc.

SENTIMENT ANALYSIS & PREDICTION

Project workflow:



IMDB Reviews

Balanced data,
50k objects (opinions),
feature: opinion (str) and
sentiment (str, binary flag).

Data pre-procesing

Analysis of input data,
determining the scope of data
to be „cleaned"

Ocena preprocessingu

Word frequency analysis +
word cloud and evaluating of
pre-processing results

Data split into:

Training dataset: 80 %
Test dataset: 20 %

Embedding

TF-IDF Vectoriser to vectorise
data

Model training

K-NeighboursClassifier
MultinomialNB
BernoulliNB
LinearSVC
LogisticRegression

Model saving

Model saving with pickles lib
for future re-use

Scoring on new data

["text", …] Model 0, 1, 0, 0

Data pre-procesing:

Data pre-processing include the following steps:

1. Expanding contractions: Replacing contractions eg. it's -> it is, ive -> i have
2. Removing HTML: tag Replacing all items included <>.
3. Replacing Emojis: Replace emojis from the analyzed text.
4. Lower Casing: Each text is converted to lowercase.
5. Tokenization: Word tokenizer split each individual word into a token.
6. Removing punctuation: Punctuation serves little value for modeling.
7. Removing stopwords: stopwords are English words which does not add much meaning to a sentence. Can be safely ignored without sacrificing the meaning of the sentence. (eg: "the", "he", "have")
8. Stemming Lemmatizing: Lemmatization is the process of converting a word to its base form. (e.g: “better” to “good”)
9. Preparing universal pre-processing function.

```
# Defining preprocessing data function to cover all above requiremens:

def prep_engine(l: list):

    #Defining punctation
    punctuation = string.punctuation
    #Set WordNetLemmatizer as var
    wordNetLem = WordNetLemmatizer()

    ## Defining set containing all stopwords in english from sklearn.feature_extracti
    stopwordlist = txt.ENGLISH_STOP_WORDS

    newlist = []

    for i in l:

        i = contractions.fix(i)

        i = re.sub('<[^>]*>', '', i)

        emoticons = re.findall('(?::|;|=)(?:-)?(?:\)|\(|D|P)', i)

        i = re.sub('[\W]+', ' ', i.lower()) + ' '.join(emoticons).replace('-', '')

        newlist.append(i)

    newSeries = pd.Series(newlist)

    newSeries = newSeries.apply(word_tokenize)

    newSeries = newSeries.apply(lambda token: [punc for punc in token if punc not in

    newSeries = newSeries.apply(lambda token: [stop for stop in token if stop not in
```

Intro model selection

The choice of models was determined by the following reasons:

- **The analyzed case concerns classification problem – determination of binary target variable (labels: negative, positive),**
- **Checking effectiveness of different classification models:**
  - **to choose the model that returns the best results (evaluation metric: accuracy – balanced dataset),**
  - **to observe the difference between training times.**

Model Evaluation:

1. KNeighborsClassifier

	precision	recall	f1-score	support
0	0.82	0.72	0.77	5035
1	0.75	0.84	0.79	4965
accuracy			0.78	10000
macro avg	0.78	0.78	0.78	10000
weighted avg	0.78	0.78	0.78	10000

Model was trained within: 29 seconds

2. MultinomialNB (Naive Bayes)

	precision	recall	f1-score	support
0	0.87	0.88	0.87	5035
1	0.87	0.87	0.87	4965
accuracy			0.87	10000
macro avg	0.87	0.87	0.87	10000
weighted avg	0.87	0.87	0.87	10000

Model was build within: 1 seconds

3. BernoulliNB (Naive Bayes)

	precision	recall	f1-score	support
0	0.86	0.88	0.87	5035
1	0.88	0.85	0.86	4965
accuracy			0.87	10000
macro avg	0.87	0.87	0.87	10000
weighted avg	0.87	0.87	0.87	10000

Model was build within: 0 seconds

4. LinearSVC Model

	precision	recall	f1-score	support
0	0.91	0.89	0.90	5035
1	0.89	0.91	0.90	4965
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

Model was build within: 1 seconds

5. Logistic Regression Model

	precision	recall	f1-score	support
0	0.90	0.88	0.89	5035
1	0.88	0.90	0.89	4965
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

Model was build within: 10 seconds

Conclusions:

The model selection was based on the accuracy metric because the data was balanced. The best model in terms of accuracy as well as the speed of "learning" was the Linear Support Vector Classification (LinearSVC) model.

Model scoring:

```
# Loading the models.
vectoriser_Tfidf, SVCLmodel = load_models()

# Text to classify should be in a list.
random_text = ["Everything in the kids section of IKEA is so cute.\
                Shame I'm nearly 19 in 2 months :(",
                "@Hegelbon That heart sliding into the waste basket.
                "I love chocolate",
                "Dumb bi***",
                "It was a pleasure!",
                "Thank you for your attention!!"]

df = predict_sentiment(vectoriser_Tfidf, SVCLmodel, random_text)
print(df)
```

```
                                               text sentiment
Everything in the kids section of IKEA is so c...  Negative
@Hegelbon That heart sliding into the waste ba...  Negative
                                  I love chocolate  Positive
                                        Dumb bi***  Negative
                                It was a pleasure!  Positive
                    Thank you for your attention!!  Positive
```

What next … ?

- model training on a larger dataset.
- changes in pre-processing function – regex, optimization of pre-processing process etc.
- changing the embedding method,
- feature engineering/hyperparameter tuning for an aspiring model, choosing a new model
- scoring on non-IMDB dat

Challenges:

Mainly related to pre-processing process:

- regex expressions ☹,
- the complexity of pre-processing function,
- 50k observation – training time 5 min (in case of 1.5 mln observation it may take over 2h*

*with linear growth and training on the usual machine ☺

THANK YOU ☺