Report on Assignment №2

«Naïve Bayes Text classification»

1 Proposed text classification method

One of the best Python libraries to work with Naïve Bayes Classifier is an NLTK because it has various methods that significantly ease building a classification model and applying some other NLP techniques.

For example, the main methods of NLTK applied in my programme are enlisted in the table below.

Table 1 – Main methods of NLTK library

The name of a method	The purpose of a method
nltk.FreqDist(all_words)	Creates a distribution of words from movie reviews
nltk.NaiveBayesClassifier.train(training_feature_set)	Conducts training of Naïve Bayes Classifier model on a training feature set
classifier.classify_many(testing_set_content)	Conducts classifying reviews from the testing set
nltk.ConfusionMatrix(golden_label, tested_label)	Creates a confusion matrix with different classes of predictions
nltk.classify.accuracy(classifier, testing_feature_set)	Calculates a final accuracy of classification
classifier.show_most_informative_features(50)	Outputs words that had the biggest difference in frequency in opposite classes of reviews and promoted efficient classification

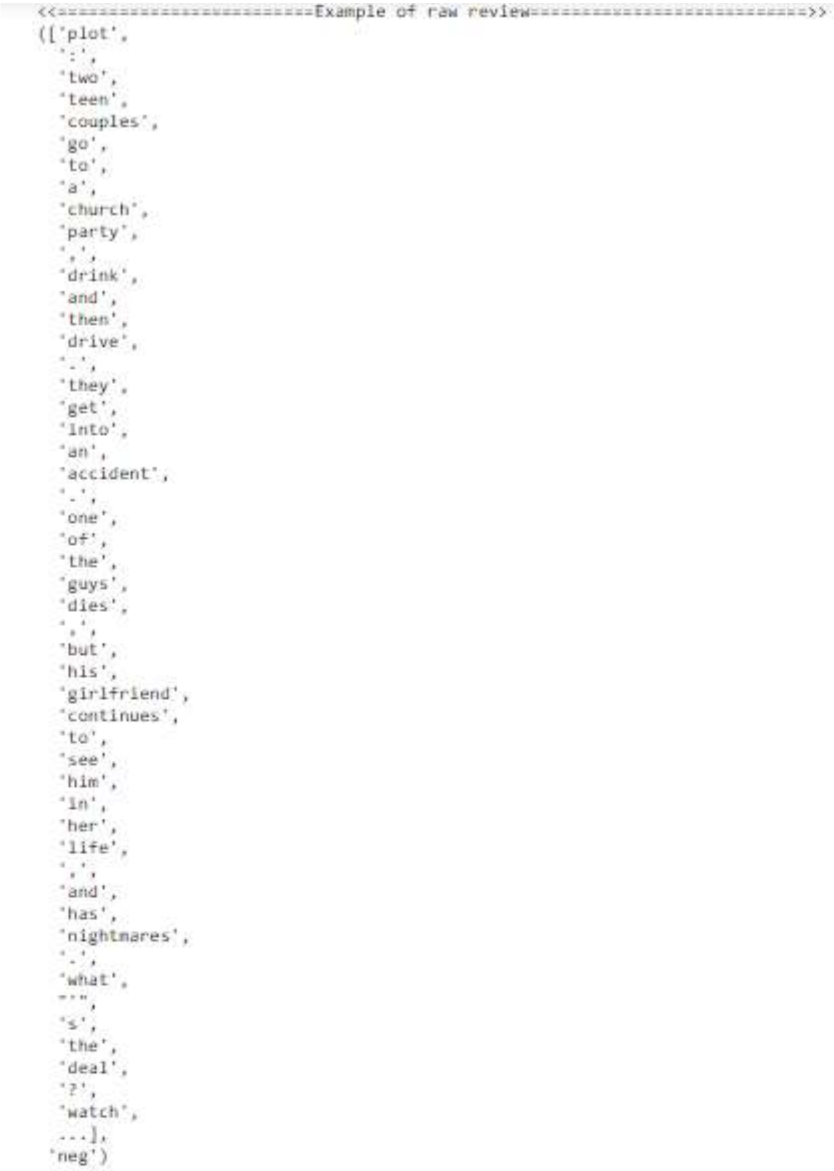
2 Pre-processing methods such as removing stop words, punctuations.

The first step of conducting pre-processing of movie reviews' corpus is changing all words' case to a lower one in order to apply a lemmatisation.

Lemmatisation helped to decrease the amount of different words but with the same lemma, thereby it preserved their common sense.

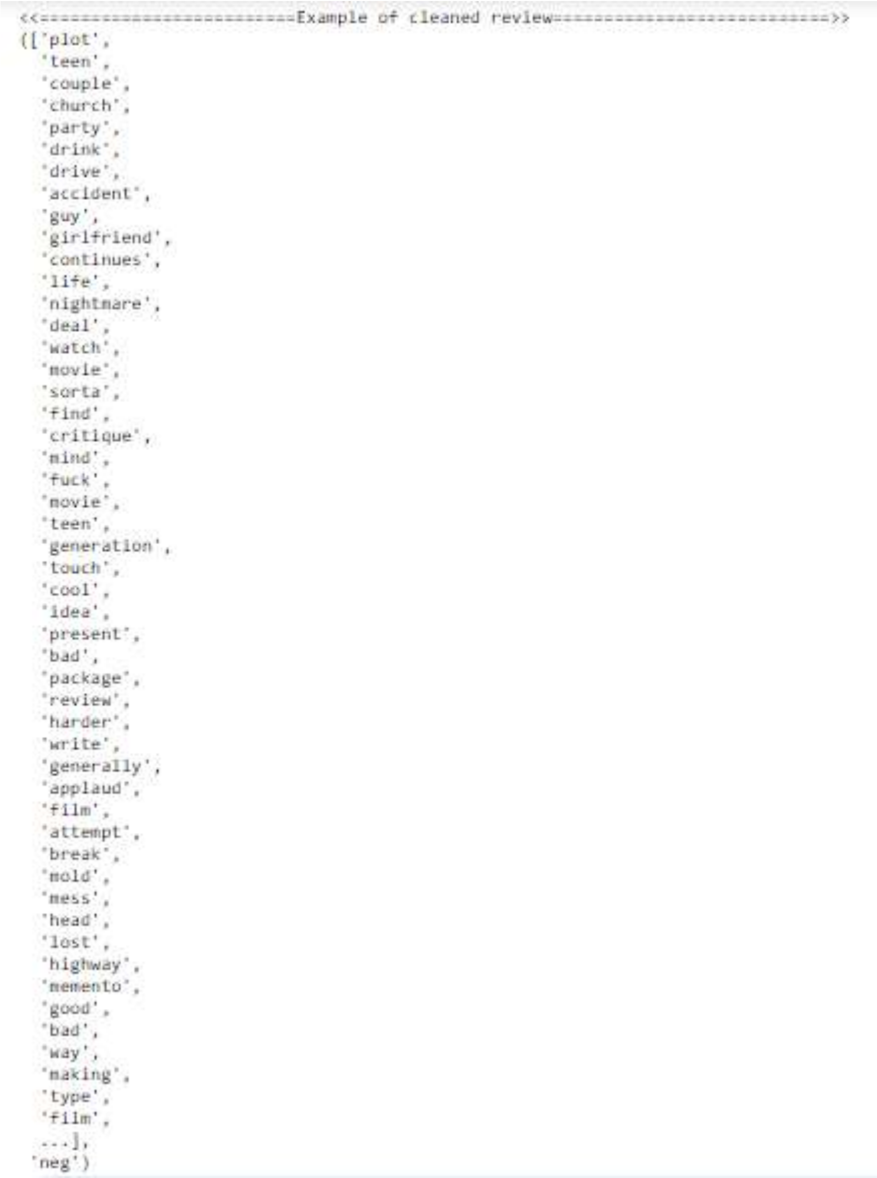
For this task the WordNetLemmatiser from NLTK was applied, because of its main feature - it is based on a huge dataset of words connected with each other by semantic relationships. As the consequence this tool managed to simplify a larger share of words in reviews.

After that, all stop-words were eliminated using a list of stop-words from Spacy (another rather useful Python NLP library) Along with that, all words that contain digits (0-9) or have a length less than two letters were removed as well. By the way, considering that initially all reviews in the corpus had been divided into single elements, this also allowed to delete even some sequential punctuation signs using that condition above. In general, the whole result may be seen in figure 1 and figure 2 below.



```
<<==========================Example of raw review============================>>
(['plot',
  ':',
  'two',
  'teen',
  'couples',
  'go',
  'to',
  'a',
  'church',
  'party',
  ',',
  'drink',
  'and',
  'then',
  'drive',
  '.',
  'they',
  'get',
  'into',
  'an',
  'accident',
  '.',
  'one',
  'of',
  'the',
  'guys',
  'dies',
  ',',
  'but',
  'his',
  'girlfriend',
  'continues',
  'to',
  'see',
  'him',
  'in',
  'her',
  'life',
  ',',
  'and',
  'has',
  'nightmares',
  '.',
  'what',
  "'",
  's',
  'the',
  'deal',
  '?',
  'watch',
  ...],
 'neg')
```

Figure 1 – The example of a raw review



```
<<=========================Example of cleaned review===========================>>
(['plot',
  'teen',
  'couple',
  'church',
  'party',
  'drink',
  'drive',
  'accident',
  'guy',
  'girlfriend',
  'continues',
  'life',
  'nightmare',
  'deal',
  'watch',
  'movie',
  'sorta',
  'find',
  'critique',
  'mind',
  'fuck',
  'movie',
  'teen',
  'generation',
  'touch',
  'cool',
  'idea',
  'present',
  'bad',
  'package',
  'review',
  'harder',
  'write',
  'generally',
  'applaud',
  'film',
  'attempt',
  'break',
  'mold',
  'mess',
  'head',
  'lost',
  'highway',
  'memento',
  'good',
  'bad',
  'way',
  'making',
  'type',
  'film',
  ...],
 'neg')
```

Figure 2 – The example of a cleaned review

Considering the figures above, the general quality and value of information have been significantly enhanced. Therefore, it would become easier for the classifier to distinguish positive and negative classes because words that occur in both classes have been removed.

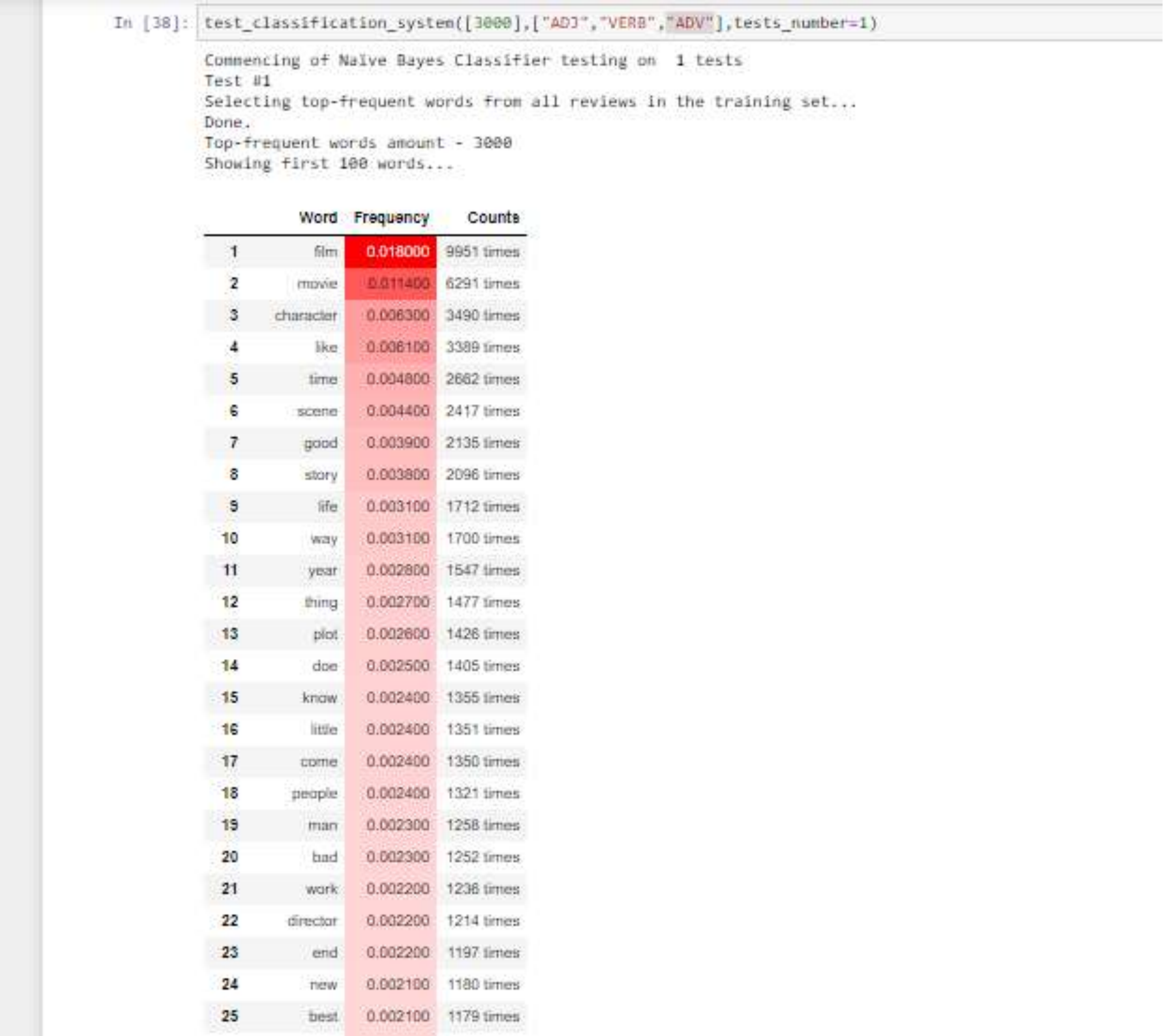
3 Feature selection methods such as the selection 3000 most important words.

There are two feature selection methods implemented in this assignment:

1. Selecting top-frequent words
2. Selecting word features using Part-Of-Speech Tagging

The first method is based on a theory that the most important words are those that have the biggest frequency in the training set of movie reviews. Thus, if we train a classifier with these words we could obtain a rather high prediction accuracy because it would be easier for a classifier to deal with top-frequent words.

To find the top-frequent words we should apply an “nltk.FreqDist(all_words)” NLTK method and take from it a “most_common(word_features_number)” property. The example of the top - 25 words from the 3000 most frequent words is in figure 3 below.



```
In [38]: test_classification_system([3000],["ADJ","VERB","ADV"],tests_number=1)

Commencing of Naive Bayes Classifier testing on  1 tests
Test #1
Selecting top-frequent words from all reviews in the training set...
Done.
Top-frequent words amount - 3000
Showing first 100 words...
```

	Word	Frequency	Counts
1	film	0.018000	9951 times
2	movie	0.011400	6291 times
3	character	0.006300	3490 times
4	like	0.006100	3389 times
5	time	0.004800	2662 times
6	scene	0.004400	2417 times
7	good	0.003900	2135 times
8	story	0.003800	2096 times
9	life	0.003100	1712 times
10	way	0.003100	1700 times
11	year	0.002800	1547 times
12	thing	0.002700	1477 times
13	plot	0.002600	1426 times
14	doe	0.002500	1405 times
15	know	0.002400	1355 times
16	little	0.002400	1351 times
17	come	0.002400	1350 times
18	people	0.002400	1321 times
19	man	0.002300	1258 times
20	bad	0.002300	1252 times
21	work	0.002200	1236 times
22	director	0.002200	1214 times
23	end	0.002200	1197 times
24	new	0.002100	1180 times
25	best	0.002100	1179 times

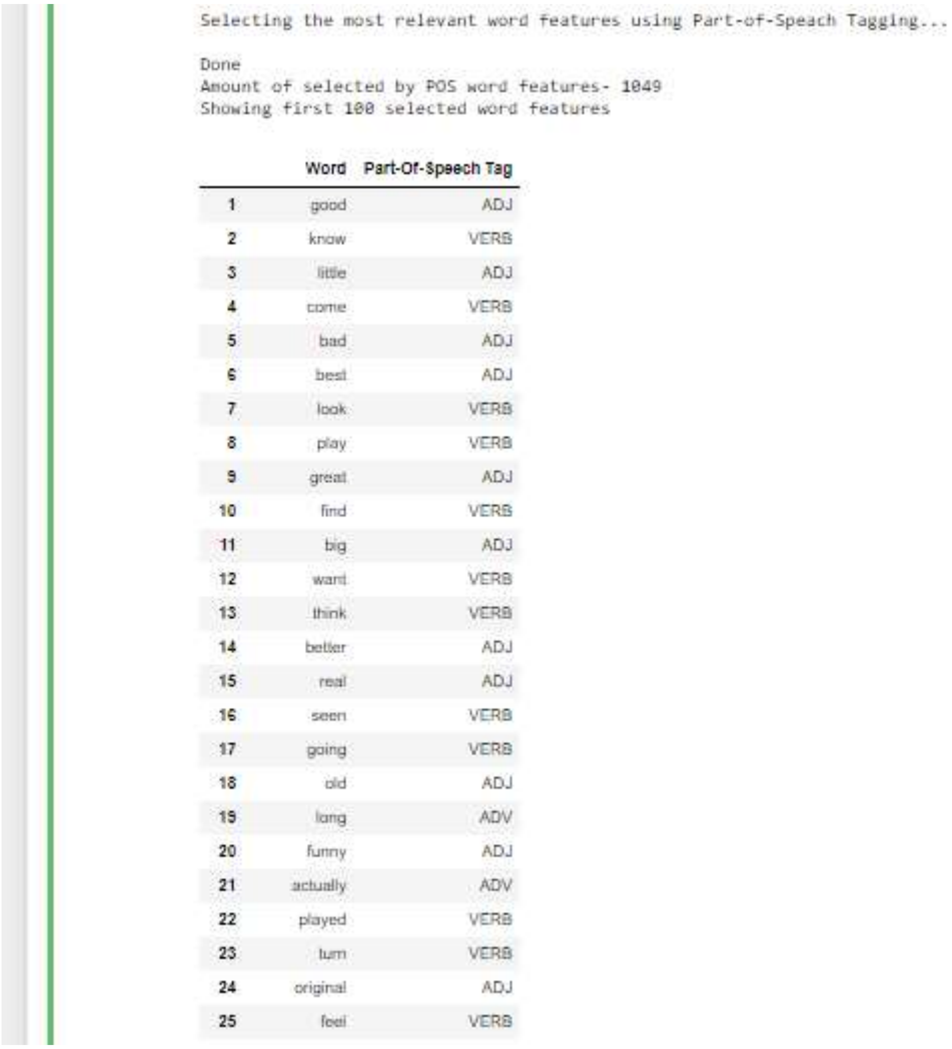
Figure 3 – The example of top-25 words among the most frequent 3000 words.

As we can see from figure 3, the most frequent word in the entire training dataset is the word “film”.

The second method is mostly based on the first one. We use Part-of-Speech Tagger to select some specific words from the top-frequent ones for our needs. For example, we may consider that the most influential words are adjectives because they are regularly used to describe something.

In this assignment, we used a Spacy library to load its “en_core_web_sm” English language model and to apply its model to determine what a certain word is using command “docs = nlp(word[0])”, where “word[0]” is a word itself.

After this, we should use the following command “docs[0].pos_” to retrieve a certain tag for that word. Finally, we select only those words that we need from a predefined list of required tags such as “["ADJ","VERB","ADV"]”. Eventually, we may obtain the next data table that can be seen in figure 4 below.



```
Selecting the most relevant word features using Part-of-Speach Tagging...

Done
Amount of selected by POS word features- 1049
Showing first 100 selected word features
```

	Word	Part-Of-Speech Tag
1	good	ADJ
2	know	VERB
3	little	ADJ
4	come	VERB
5	bad	ADJ
6	best	ADJ
7	look	VERB
8	play	VERB
9	great	ADJ
10	find	VERB
11	big	ADJ
12	want	VERB
13	think	VERB
14	better	ADJ
15	real	ADJ
16	seen	VERB
17	going	VERB
18	old	ADJ
19	long	ADV
20	funny	ADJ
21	actually	ADV
22	played	VERB
23	turn	VERB
24	original	ADJ
25	feel	VERB

Figure 4 – The example of a data table with words and their tags.

4 Model evaluation

The model evaluation is mainly conducted by methods “train”, “classify_many”, “nltk.ConfusionMatrix”,“nltk.classify.accuracy”, “classifier.show_most_informative_features” mentioned in table 1 and one another method – “calculate_metrics(cm)”, which is purposed to calculate Recall, Precision, Accuracy and F1 Score of the built model.

Besides that, the model evaluation is divided into carrying out multiple tests with different amounts of top-frequent words and various POS-tags. As the result, we test our model several times on a certain set of input parameters to define the average values of all metrics. Thus, we can estimate our model with better accuracy.

In this project, the Naïve Bayes Classifier was tested on the following sets of input parameters and returned the following results represented in table 2.

Table 2 – Table of tests and obtained results

Single launch of the Naïve Bayes Classifier with 3000 top-frequent words	
[3000], ["ADJ","VERB","ADV"], tests_number=1	{'3000 top-frequent words': { 'Average recall': '83.908%', 'Average Precision': '73.000%', 'Average accuracy': '79.500%', 'Average F1 score': '78.075%'}, 'Tests quantity': 1, 'POS-tags': ['ADJ', 'VERB', 'ADV']}
Experiments with different amount of top-frequent words for feature selection and training the Naïve Bayes Classifier	

([1000,2000,3000,4000,8000, 10000,12000, 15000, 18000, 20000],
["ADJ","VERB","ADV"],
tests_number=3
	{'1000 top-frequent words':
	{
	'Average recall': '79.682%',
	'Average Precision': '79.333%',
	'Average accuracy': '79.500%',
	'Average F1 score': '79.381%'},
	'Tests quantity': 3,
	'POS-tags': ['ADJ', 'VERB', 'ADV']
	}

```
{'2000 top-frequent words':
{
'Average recall': '83.451%',
'Average Precision': '76.000
%',
 'Average accuracy': '80.333
%',
'Average F1 score': '79.330%
'},
'Tests quantity': 3,
'POS-tags': ['ADJ', 'VERB',
'ADV']
}
{'3000 top-frequent words':
{
'Average recall': '84.634%',
'Average Precision': '78.000
%',
'Average accuracy': '81.833%
',
'Average F1 score': '81.141%
'},
'Tests quantity': 3,
'POS-tags': ['ADJ', 'VERB',
'ADV']
}
{'4000 top-frequent words':
{
'Average recall': '83.044%',
'Average Precision': '75.000
%',
 'Average accuracy': '79.833
%',
'Average F1 score': '78.697%
'},
'Tests quantity': 3,
'POS-tags': ['ADJ', 'VERB',
'ADV']
}
{'8000 top-frequent words':
{
'Average recall': '82.439%',
 'Average Precision': '71.66
7%',
 'Average accuracy': '78.167
%',
'Average F1 score': '76.598%
'},
'Tests quantity': 3,
'POS-tags': ['ADJ', 'VERB',
'ADV']
}
{'10000 top-frequent words':
{
'Average recall': '82.970%',
'Average Precision': '73.333
%',
'Average accuracy': '79.167%
',
'Average F1 score': '77.761%
'},
 'Tests quantity': 3,
 'POS-tags': ['ADJ', 'VERB',
'ADV']
}
{'12000 top-frequent words':
{
'Average recall': '80.448%',
'Average Precision': '73.333
%',
'Average accuracy': '77.667%
',
 'Average F1 score': '76.661
%'},
'Tests quantity': 3,
 'POS-tags': ['ADJ', 'VERB',
'ADV']
}
{'15000 top-frequent words':
{
'Average recall': '82.337%',
'Average Precision': '72.667
%',
'Average accuracy': '78.500%
',
'Average F1 score': '77.174%
'},
 'Tests quantity': 3,
'POS-tags': ['ADJ', 'VERB',
'ADV']
}
{'18000 top-frequent words':
{
'Average recall': '82.937%',
 'Average Precision': '71.33
3%',
 'Average accuracy': '78.333
%',
 'Average F1 score': '76.678
%'},
 'Tests quantity': 3,
 'POS-tags': ['ADJ', 'VERB',
'ADV']}
{'20000 top-frequent words':
{
'Average recall': '82.900%',
'Average Precision': '73.667
%',
'Average accuracy': '79.333%
',
```

	'Average F1 score': '77.957%'}, 'Tests quantity': 3, 'POS-tags': ['ADJ', 'VERB', 'ADV']}
Experiments with some combinations of Part-Of-Speach tags	
[8000],None,tests_number=1	{'8000 top-frequent words': { 'Average recall': '82.353%', 'Average Precision': '70.000%', 'Average accuracy': '77.500%', 'Average F1 score': '75.676%'}, 'Tests quantity': 1, 'POS-tags': None}
[8000],None,tests_number=5	{'8000 top-frequent words': {'Average recall': '82.281%', 'Average Precision': '69.600%', 'Average accuracy': '77.400%', 'Average F1 score': '75.266%'}, 'Tests quantity': 5, 'POS-tags': None}
[8000],["ADJ"],tests_number=1	{'8000 top-frequent words': {'Average recall': '87.640%', 'Average Precision': '78.000%', 'Average accuracy': '83.500%', 'Average F1 score': '82.540%'}, 'Tests quantity': 1, 'POS-tags': ['ADJ']
[8000],["ADJ"],tests_number=5	{'8000 top-frequent words': {'Average recall': '80.111%', 'Average Precision': '73.600%', 'Average accuracy': '77.600%', 'Average F1 score': '76.544%'}, 'Tests quantity': 5, 'POS-tags': ['ADJ']}
[8000],["ADJ","NOUN",],tests_number=1	{'8000 top-frequent words': {'Average recall': '86.420%', 'Average Precision': '70.000%', 'Average accuracy': '79.500%', 'Average F1 score': '77.348%'}, 'Tests quantity': 1, 'POS-tags': ['ADJ', 'NOUN']}
[8000],["ADJ","NOUN",],tests_number=5	{'8000 top-frequent words': {'Average recall': '82.666%', 'Average Precision': '69.400%', 'Average accuracy': '77.400%', 'Average F1 score': '75.368%'}, 'Tests quantity'

	: 5, 'POS-tags': ['ADJ', 'NOUN']}
[8000],["ADJ","NOUN","VERB"],tests_number=1	{'8000 top-frequent words': {'Average recall': '82.895%', 'Average Precision': '63.000%', 'Average accuracy': '75.000%', 'Average F1 score': '71.591%'}, 'Tests quantity': 1, 'POS-tags': ['ADJ', 'NOUN', 'VERB']
[8000],["ADJ","NOUN","VERB"],tests_number=5	{'8000 top-frequent words': {'Average recall': '82.189%', 'Average Precision': '73.400%', 'Average accuracy': '78.700%', 'Average F1 score': '77.518%'}, 'Tests quantity': 5, 'POS-tags': ['ADJ', 'NOUN', 'VERB']}
[8000],["ADJ","NOUN","VERB","ADV"],tests_number=1	{'8000 top-frequent words': {'Average recall': '81.818%', 'Average Precision': '72.000%', 'Average accuracy': '78.000%', 'Average F1 score': '76.596%'}, 'Tests quantity': 1, 'POS-tags': ['ADJ', 'NOUN', 'VERB', 'ADV']
[8000],["ADJ","NOUN","VERB","ADV"],tests_number=5	{'8000 top-frequent words': {'Average recall': '83.813%', 'Average Precision': '74.200%', 'Average accuracy': '79.900%', 'Average F1 score': '78.665%'}, 'Tests quantity': 5, 'POS-tags': ['ADJ', 'NOUN', 'VERB', 'ADV']

As we can see from the second experiment with different amounts of word features, the best result is reached with 3000 words features: all its metrics (except the precision which is less than a precision from the first test) have the highest value among values from other tests. Thus, it should be concluded that an increase in the number of word features does not always mean an increase in classification accuracy.

Besides this, if we take a look at the last set of experiments with POS tags we can notice that the following combination of tags such as "["ADJ","NOUN","VERB","ADV"]," has provided the highest values of all metrics on 5 tests. This means that these particular parts of speech have the greatest influence on a text’s sentiment.

All in all, in this assignment four stages of conducting classification of movie reviews using the Naïve Bayes Classifier, were represented. Also, we conducted some fascinating experiments with various amounts of the most frequent words or with different combinations of POS-tags. The retrieved results will help us to understand the applied techniques and continue further research of such field of Data Science as NLE.