МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

**Лабораторная работа №\_\_5\_\_**

по дисциплине«Методы машинного обучения»

ИСПОЛНИТЕЛЬ: \_Морозенков О.Н\_\_\_

ФИО

группа ИУ5-23М \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2022 г.

ПРЕПОДАВАТЕЛЬ: \_Гапанюк Ю.Е.\_\_\_\_\_

ФИО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2022 г.

Москва - 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Цель лабораторной работы:** изучение методов предобработки и классификации текстовых данных.

**Задание:**

1. Для произвольного предложения или текста решите следующие задачи:

* Токенизация.
* Частеречная разметка.
* Лемматизация.
* Выделение (распознавание) именованных сущностей.
* Разбор предложения.

1. Для произвольного набора данных, предназначенного для классификации текстов, решите задачу классификации текста двумя способами:

* Способ 1. На основе CountVectorizer или TfidfVectorizer.
* Способ 2. На основе моделей word2vec или Glove или fastText.
* Сравните качество полученных моделей.

Для поиска наборов данных в поисковой системе можно использовать ключевые слова "datasets for text classification".

text = '''С другой стороны социально-экономическое развитие влечет за собой процесс внедрения и модернизации модели развития.   
Разнообразный и богатый опыт начало повседневной работы по формированию позиции представляет собой интересный эксперимент проверки направлений прогрессивного развития.   
Повседневная практика показывает, что новая модель организационной деятельности играет важную роль в формировании модели развития.'''  
text2 = 'Россия или Российская Федерация — государство в Восточной Европе и Северной Азии со столицей в городе Москва.'

# Задача токенизации

from razdel import tokenize, sentenize

n\_tok\_text = list(tokenize(text))  
n\_tok\_text

[Substring(0, 1, 'С'),  
 Substring(2, 8, 'другой'),  
 Substring(9, 16, 'стороны'),  
 Substring(17, 40, 'социально-экономическое'),  
 Substring(41, 49, 'развитие'),  
 Substring(50, 56, 'влечет'),  
 Substring(57, 59, 'за'),  
 Substring(60, 65, 'собой'),  
 Substring(66, 73, 'процесс'),  
 Substring(74, 83, 'внедрения'),  
 Substring(84, 85, 'и'),  
 Substring(86, 98, 'модернизации'),  
 Substring(99, 105, 'модели'),  
 Substring(106, 114, 'развития'),  
 Substring(114, 115, '.'),  
 Substring(117, 130, 'Разнообразный'),  
 Substring(131, 132, 'и'),  
 Substring(133, 140, 'богатый'),  
 Substring(141, 145, 'опыт'),  
 Substring(146, 152, 'начало'),  
 Substring(153, 165, 'повседневной'),  
 Substring(166, 172, 'работы'),  
 Substring(173, 175, 'по'),  
 Substring(176, 188, 'формированию'),  
 Substring(189, 196, 'позиции'),  
 Substring(197, 209, 'представляет'),  
 Substring(210, 215, 'собой'),  
 Substring(216, 226, 'интересный'),  
 Substring(227, 238, 'эксперимент'),  
 Substring(239, 247, 'проверки'),  
 Substring(248, 259, 'направлений'),  
 Substring(260, 274, 'прогрессивного'),  
 Substring(275, 283, 'развития'),  
 Substring(283, 284, '.'),  
 Substring(286, 298, 'Повседневная'),  
 Substring(299, 307, 'практика'),  
 Substring(308, 318, 'показывает'),  
 Substring(318, 319, ','),  
 Substring(320, 323, 'что'),  
 Substring(324, 329, 'новая'),  
 Substring(330, 336, 'модель'),  
 Substring(337, 352, 'организационной'),  
 Substring(353, 365, 'деятельности'),  
 Substring(366, 372, 'играет'),  
 Substring(373, 379, 'важную'),  
 Substring(380, 384, 'роль'),  
 Substring(385, 386, 'в'),  
 Substring(387, 399, 'формировании'),  
 Substring(400, 406, 'модели'),  
 Substring(407, 415, 'развития'),  
 Substring(415, 416, '.')]

[\_.text for \_ in n\_tok\_text]

['С',  
 'другой',  
 'стороны',  
 'социально-экономическое',  
 'развитие',  
 'влечет',  
 'за',  
 'собой',  
 'процесс',  
 'внедрения',  
 'и',  
 'модернизации',  
 'модели',  
 'развития',  
 '.',  
 'Разнообразный',  
 'и',  
 'богатый',  
 'опыт',  
 'начало',  
 'повседневной',  
 'работы',  
 'по',  
 'формированию',  
 'позиции',  
 'представляет',  
 'собой',  
 'интересный',  
 'эксперимент',  
 'проверки',  
 'направлений',  
 'прогрессивного',  
 'развития',  
 '.',  
 'Повседневная',  
 'практика',  
 'показывает',  
 ',',  
 'что',  
 'новая',  
 'модель',  
 'организационной',  
 'деятельности',  
 'играет',  
 'важную',  
 'роль',  
 'в',  
 'формировании',  
 'модели',  
 'развития',  
 '.']

n\_sen\_text = list(sentenize(text))  
n\_sen\_text

[Substring(0,  
 115,  
 'С другой стороны социально-экономическое развитие влечет за собой процесс внедрения и модернизации модели развития.'),  
 Substring(117,  
 284,  
 'Разнообразный и богатый опыт начало повседневной работы по формированию позиции представляет собой интересный эксперимент проверки направлений прогрессивного развития.'),  
 Substring(286,  
 416,  
 'Повседневная практика показывает, что новая модель организационной деятельности играет важную роль в формировании модели развития.')]

[\_.text for \_ in n\_sen\_text], len([\_.text for \_ in n\_sen\_text])

(['С другой стороны социально-экономическое развитие влечет за собой процесс внедрения и модернизации модели развития.',  
 'Разнообразный и богатый опыт начало повседневной работы по формированию позиции представляет собой интересный эксперимент проверки направлений прогрессивного развития.',  
 'Повседневная практика показывает, что новая модель организационной деятельности играет важную роль в формировании модели развития.'],  
 3)

# Этот вариант токенизации нужен для последующей обработки  
def n\_sentenize(text):  
 n\_sen\_chunk = []  
 for sent in sentenize(text):  
 tokens = [\_.text for \_ in tokenize(sent.text)]  
 n\_sen\_chunk.append(tokens)  
 return n\_sen\_chunk

n\_sen\_chunk = n\_sentenize(text)  
n\_sen\_chunk

[['С',  
 'другой',  
 'стороны',  
 'социально-экономическое',  
 'развитие',  
 'влечет',  
 'за',  
 'собой',  
 'процесс',  
 'внедрения',  
 'и',  
 'модернизации',  
 'модели',  
 'развития',  
 '.'],  
 ['Разнообразный',  
 'и',  
 'богатый',  
 'опыт',  
 'начало',  
 'повседневной',  
 'работы',  
 'по',  
 'формированию',  
 'позиции',  
 'представляет',  
 'собой',  
 'интересный',  
 'эксперимент',  
 'проверки',  
 'направлений',  
 'прогрессивного',  
 'развития',  
 '.'],  
 ['Повседневная',  
 'практика',  
 'показывает',  
 ',',  
 'что',  
 'новая',  
 'модель',  
 'организационной',  
 'деятельности',  
 'играет',  
 'важную',  
 'роль',  
 'в',  
 'формировании',  
 'модели',  
 'развития',  
 '.']]

n\_sen\_chunk\_2 = n\_sentenize(text2)  
n\_sen\_chunk\_2

[['Россия',  
 'или',  
 'Российская',  
 'Федерация',  
 '—',  
 'государство',  
 'в',  
 'Восточной',  
 'Европе',  
 'и',  
 'Северной',  
 'Азии',  
 'со',  
 'столицей',  
 'в',  
 'городе',  
 'Москва',  
 '.']]

# Частеречная разметка

from navec import Navec  
from slovnet import Morph

# Файл необходимо скачать по ссылке https://github.com/natasha/navec#downloads  
navec = Navec.load('navec\_news\_v1\_1B\_250K\_300d\_100q.tar')

# Файл необходимо скачать по ссылке https://github.com/natasha/slovnet#downloads  
n\_morph = Morph.load('slovnet\_morph\_news\_v1.tar', batch\_size=4)

morph\_res = n\_morph.navec(navec)

def print\_pos(markup):  
 for token in markup.tokens:  
 print('{} - {}'.format(token.text, token.tag))

n\_text\_markup = list(\_ for \_ in n\_morph.map(n\_sen\_chunk))  
[print\_pos(x) for x in n\_text\_markup]

С - ADP  
другой - ADJ|Case=Gen|Degree=Pos|Gender=Fem|Number=Sing  
стороны - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing  
социально-экономическое - ADJ|Case=Nom|Degree=Pos|Gender=Neut|Number=Sing  
развитие - NOUN|Animacy=Inan|Case=Nom|Gender=Neut|Number=Sing  
влечет - VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act  
за - ADP  
собой - PRON|Case=Ins  
процесс - NOUN|Animacy=Inan|Case=Acc|Gender=Masc|Number=Sing  
внедрения - NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing  
и - CCONJ  
модернизации - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing  
модели - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing  
развития - NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing  
. - PUNCT  
Разнообразный - ADJ|Case=Nom|Degree=Pos|Gender=Masc|Number=Sing  
и - CCONJ  
богатый - ADJ|Case=Nom|Degree=Pos|Gender=Masc|Number=Sing  
опыт - NOUN|Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing  
начало - NOUN|Animacy=Inan|Case=Nom|Gender=Neut|Number=Sing  
повседневной - ADJ|Case=Gen|Degree=Pos|Gender=Fem|Number=Sing  
работы - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing  
по - ADP  
формированию - NOUN|Animacy=Inan|Case=Dat|Gender=Neut|Number=Sing  
позиции - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing  
представляет - VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act  
собой - PRON|Case=Ins  
интересный - ADJ|Animacy=Inan|Case=Acc|Degree=Pos|Gender=Masc|Number=Sing  
эксперимент - NOUN|Animacy=Inan|Case=Acc|Gender=Masc|Number=Sing  
проверки - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing  
направлений - NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Plur  
прогрессивного - ADJ|Case=Gen|Degree=Pos|Gender=Neut|Number=Sing  
развития - NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing  
. - PUNCT  
Повседневная - ADJ|Case=Nom|Degree=Pos|Gender=Fem|Number=Sing  
практика - NOUN|Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing  
показывает - VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act  
, - PUNCT  
что - SCONJ  
новая - ADJ|Case=Nom|Degree=Pos|Gender=Fem|Number=Sing  
модель - NOUN|Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing  
организационной - ADJ|Case=Gen|Degree=Pos|Gender=Fem|Number=Sing  
деятельности - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing  
играет - VERB|Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act  
важную - ADJ|Case=Acc|Degree=Pos|Gender=Fem|Number=Sing  
роль - NOUN|Animacy=Inan|Case=Acc|Gender=Fem|Number=Sing  
в - ADP  
формировании - NOUN|Animacy=Inan|Case=Loc|Gender=Neut|Number=Sing  
модели - NOUN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing  
развития - NOUN|Animacy=Inan|Case=Gen|Gender=Neut|Number=Sing  
. - PUNCT

[None, None, None]

n\_text2\_markup = list(n\_morph.map(n\_sen\_chunk\_2))  
[print\_pos(x) for x in n\_text2\_markup]

Россия - PROPN|Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing  
или - CCONJ  
Российская - ADJ|Case=Nom|Degree=Pos|Gender=Fem|Number=Sing  
Федерация - PROPN|Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing  
— - PUNCT  
государство - NOUN|Animacy=Inan|Case=Nom|Gender=Neut|Number=Sing  
в - ADP  
Восточной - ADJ|Case=Loc|Degree=Pos|Gender=Fem|Number=Sing  
Европе - PROPN|Animacy=Inan|Case=Loc|Gender=Fem|Number=Sing  
и - CCONJ  
Северной - ADJ|Case=Gen|Degree=Pos|Gender=Fem|Number=Sing  
Азии - PROPN|Animacy=Inan|Case=Gen|Gender=Fem|Number=Sing  
со - ADP  
столицей - NOUN|Animacy=Inan|Case=Ins|Gender=Fem|Number=Sing  
в - ADP  
городе - NOUN|Animacy=Inan|Case=Loc|Gender=Masc|Number=Sing  
Москва - PROPN|Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing  
. - PUNCT

[None]

# Лемматизация

from natasha import Doc, Segmenter, NewsEmbedding, NewsMorphTagger, MorphVocab

def n\_lemmatize(text):  
 emb = NewsEmbedding()  
 morph\_tagger = NewsMorphTagger(emb)  
 segmenter = Segmenter()  
 morph\_vocab = MorphVocab()  
 doc = Doc(text)  
 doc.segment(segmenter)  
 doc.tag\_morph(morph\_tagger)  
 for token in doc.tokens:  
 token.lemmatize(morph\_vocab)  
 return doc

n\_doc = n\_lemmatize(text)  
{\_.text: \_.lemma for \_ in n\_doc.tokens}

{'С': 'с',  
 'другой': 'другой',  
 'стороны': 'сторона',  
 'социально-экономическое': 'социально-экономический',  
 'развитие': 'развитие',  
 'влечет': 'влечь',  
 'за': 'за',  
 'собой': 'себя',  
 'процесс': 'процесс',  
 'внедрения': 'внедрение',  
 'и': 'и',  
 'модернизации': 'модернизация',  
 'модели': 'модель',  
 'развития': 'развитие',  
 '.': '.',  
 'Разнообразный': 'разнообразный',  
 'богатый': 'богатый',  
 'опыт': 'опыт',  
 'начало': 'начало',  
 'повседневной': 'повседневный',  
 'работы': 'работа',  
 'по': 'по',  
 'формированию': 'формирование',  
 'позиции': 'позиция',  
 'представляет': 'представлять',  
 'интересный': 'интересный',  
 'эксперимент': 'эксперимент',  
 'проверки': 'проверка',  
 'направлений': 'направление',  
 'прогрессивного': 'прогрессивный',  
 'Повседневная': 'повседневный',  
 'практика': 'практика',  
 'показывает': 'показывать',  
 ',': ',',  
 'что': 'что',  
 'новая': 'новый',  
 'модель': 'модель',  
 'организационной': 'организационный',  
 'деятельности': 'деятельность',  
 'играет': 'играть',  
 'важную': 'важный',  
 'роль': 'роль',  
 'в': 'в',  
 'формировании': 'формирование'}

n\_doc2 = n\_lemmatize(text2)  
{\_.text: \_.lemma for \_ in n\_doc2.tokens}

{'Россия': 'россия',  
 'или': 'или',  
 'Российская': 'российский',  
 'Федерация': 'федерация',  
 '—': '—',  
 'государство': 'государство',  
 'в': 'в',  
 'Восточной': 'восточный',  
 'Европе': 'европа',  
 'и': 'и',  
 'Северной': 'северный',  
 'Азии': 'азия',  
 'со': 'с',  
 'столицей': 'столица',  
 'городе': 'город',  
 'Москва': 'москва',  
 '.': '.'}

# Выделение (распознавание) именованных сущностей

from slovnet import NER  
from ipymarkup import show\_span\_ascii\_markup as show\_markup

ner = NER.load('slovnet\_ner\_news\_v1.tar')

ner\_res = ner.navec(navec)

markup\_ner = ner(text2)  
markup\_ner

SpanMarkup(  
 text='Россия или Российская Федерация — государство в Восточной Европе и Северной Азии со столицей в городе Москва.',  
 spans=[Span(  
 start=0,  
 stop=6,  
 type='LOC'  
 ),  
 Span(  
 start=11,  
 stop=31,  
 type='LOC'  
 ),  
 Span(  
 start=48,  
 stop=64,  
 type='LOC'  
 ),  
 Span(  
 start=67,  
 stop=80,  
 type='LOC'  
 ),  
 Span(  
 start=102,  
 stop=108,  
 type='LOC'  
 )]  
)

show\_markup(markup\_ner.text, markup\_ner.spans)

Россия или Российская Федерация — государство в Восточной Европе и   
LOC─── LOC───────────────── LOC─────────────   
Северной Азии со столицей в городе Москва.  
LOC────────── LOC───

# Разбор предложения

from natasha import NewsSyntaxParser

emb = NewsEmbedding()  
syntax\_parser = NewsSyntaxParser(emb)

n\_doc.parse\_syntax(syntax\_parser)  
n\_doc.sents[0].syntax.print()

┌──► С case  
 │ ┌► другой amod  
 ┌►└─└─ стороны obl  
 │ ┌► социально-экономическое amod  
 │ ┌►└─ развитие nsubj  
┌─┌─└─└─── влечет   
│ │ │ ┌► за case  
│ │ └──►└─ собой obl  
│ └────►┌─ процесс obj  
│ ┌─┌─└► внедрения nmod  
│ │ │ ┌► и cc  
│ │ └►└─ модернизации conj  
│ └──►┌─ модели nmod  
│ └► развития nmod  
└────────► . punct

n\_doc.parse\_syntax(syntax\_parser)  
n\_doc.sents[1].syntax.print()

┌──► Разнообразный amod  
 │ ┌► и cc  
 │ └─ богатый   
┌──────►└─── опыт conj  
│ ┌──►┌─── начало obj  
│ │ │ ┌► повседневной amod  
│ │ ┌─└►└─ работы nmod  
│ │ │ ┌► по case  
│ │ └►┌─└─ формированию nmod  
│ │ └──► позиции nmod  
│ ┌─└───┌─┌─ представляет   
│ │ │ └► собой fixed  
│ │ │ ┌► интересный amod  
└─│ └►└─ эксперимент obj  
 │ └►┌─ проверки nmod  
 │ ┌─└► направлений nmod  
 │ │ ┌► прогрессивного amod  
 │ └►└─ развития nmod  
 └────────► . punct

n\_doc.parse\_syntax(syntax\_parser)  
n\_doc.sents[2].syntax.print()

┌► Повседневная amod  
 ┌►└─ практика nsubj  
┌─┌───────└─── показывает   
│ │ ┌────────► , punct  
│ │ │ ┌──────► что mark  
│ │ │ │ ┌► новая amod  
│ │ │ │ ┌►┌─└─ модель nsubj  
│ │ │ │ │ │ ┌► организационной amod  
│ │ │ │ │ └►└─ деятельности nmod  
│ └►└─└─└─┌─── играет ccomp  
│ │ ┌► важную amod  
│ ┌─└►└─ роль obj  
│ │ ┌► в case  
│ └►┌─└─ формировании nmod  
│ └►┌─ модели nmod  
│ └► развития nmod  
└────────────► . punct

n\_doc2.parse\_syntax(syntax\_parser)  
n\_doc2.sents[0].syntax.print()

┌────► Россия nsubj  
 │ ┌──► или cc  
 │ │ ┌► Российская amod  
 │ └─└─ Федерация   
 │ ┌► — punct  
┌─┌───└───└─ государство   
│ │ │ ┌──► в case  
│ │ │ │ ┌► Восточной amod  
│ │ ┌─└►└─└─ Европе nmod  
│ │ │ ┌──► и cc  
│ │ │ │ ┌► Северной amod  
│ │ └──►└─└─ Азии conj  
│ │ ┌► со case  
│ └────►┌─└─ столицей nmod  
│ │ ┌► в case  
│ └►└─ городе nmod  
│ └──► Москва appos  
└──────────► . punct

import numpy as np  
import pandas as pd  
from typing import Dict, Tuple  
from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer  
from sklearn.model\_selection import GridSearchCV, RandomizedSearchCV  
from sklearn.metrics import accuracy\_score, balanced\_accuracy\_score  
from sklearn.metrics import precision\_score, recall\_score, f1\_score, classification\_report  
from sklearn.metrics import confusion\_matrix  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.pipeline import Pipeline  
from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, mean\_squared\_log\_error, median\_absolute\_error, r2\_score   
from sklearn.metrics import roc\_curve, roc\_auc\_score  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.linear\_model import LogisticRegression  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.svm import LinearSVC  
from sklearn.model\_selection import train\_test\_split  
import seaborn as sns  
from collections import Counter  
from sklearn.datasets import fetch\_20newsgroups  
import matplotlib.pyplot as plt  
  
%matplotlib inline   
sns.set(style="ticks")

# Векторизация текста на основе модели "мешка слов"

categories = ["rec.motorcycles", "rec.sport.baseball", "sci.electronics","sci.med"]  
newsgroups = fetch\_20newsgroups(subset='train', categories=categories)  
data = newsgroups['data']

def accuracy\_score\_for\_classes(  
 y\_true: np.ndarray,   
 y\_pred: np.ndarray) -> Dict[int, float]:  
 """  
 Вычисление метрики accuracy для каждого класса  
 y\_true - истинные значения классов  
 y\_pred - предсказанные значения классов  
 Возвращает словарь: ключ - метка класса,   
 значение - Accuracy для данного класса  
 """  
 # Для удобства фильтрации сформируем Pandas DataFrame   
 d = {'t': y\_true, 'p': y\_pred}  
 df = pd.DataFrame(data=d)  
 # Метки классов  
 classes = np.unique(y\_true)  
 # Результирующий словарь  
 res = dict()  
 # Перебор меток классов  
 for c in classes:  
 # отфильтруем данные, которые соответствуют   
 # текущей метке класса в истинных значениях  
 temp\_data\_flt = df[df['t']==c]  
 # расчет accuracy для заданной метки класса  
 temp\_acc = accuracy\_score(  
 temp\_data\_flt['t'].values,   
 temp\_data\_flt['p'].values)  
 # сохранение результата в словарь  
 res[c] = temp\_acc  
 return res  
  
def print\_accuracy\_score\_for\_classes(  
 y\_true: np.ndarray,   
 y\_pred: np.ndarray):  
 """  
 Вывод метрики accuracy для каждого класса  
 """  
 accs = accuracy\_score\_for\_classes(y\_true, y\_pred)  
 if len(accs)>0:  
 print('Метка \t Accuracy')  
 for i in accs:  
 print('{} \t {}'.format(i, accs[i]))

vocabVect = CountVectorizer()  
vocabVect.fit(data)  
corpusVocab = vocabVect.vocabulary\_  
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))

Количество сформированных признаков - 33448

for i in list(corpusVocab)[1:10]:  
 print('{}={}'.format(i, corpusVocab[i]))

nrmendel=22213  
unix=31462  
amherst=5287  
edu=12444  
nathaniel=21624  
mendell=20477  
subject=29220  
re=25369  
bike=6898

# Использование класса CountVectorizer

test\_features = vocabVect.transform(data)  
test\_features

<2380x33448 sparse matrix of type '<class 'numpy.int64'>'  
 with 335176 stored elements in Compressed Sparse Row format>

test\_features.todense()

matrix([[0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0],  
 ...,  
 [2, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0]])

# Размер нулевой строки  
len(test\_features.todense()[0].getA1())

33448

# Непустые значения нулевой строки  
print([i for i in test\_features.todense()[0].getA1() if i>0])

[1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 2]

vocabVect.get\_feature\_names()[0:10]

/home/dnikolskiy/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get\_feature\_names is deprecated; get\_feature\_names is deprecated in 1.0 and will be removed in 1.2. Please use get\_feature\_names\_out instead.  
 warnings.warn(msg, category=FutureWarning)

['00',  
 '000',  
 '0000',  
 '0000000004',  
 '0000000005',  
 '0000000667',  
 '0000001200',  
 '0001',  
 '00014',  
 '0002']

# Решение задачи анализа тональности текста на основе модели "мешка слов"

def VectorizeAndClassify(vectorizers\_list, classifiers\_list):  
 for v in vectorizers\_list:  
 for c in classifiers\_list:  
 pipeline1 = Pipeline([("vectorizer", v), ("classifier", c)])  
 score = cross\_val\_score(pipeline1, newsgroups['data'], newsgroups['target'], scoring='accuracy', cv=3).mean()  
 print('Векторизация - {}'.format(v))  
 print('Модель для классификации - {}'.format(c))  
 print('Accuracy = {}'.format(score))  
 print('===========================')

vectorizers\_list = [CountVectorizer(vocabulary = corpusVocab)]  
classifiers\_list = [LogisticRegression(C=3.0), LinearSVC(), KNeighborsClassifier()]  
VectorizeAndClassify(vectorizers\_list, classifiers\_list)

/home/dnikolskiy/anaconda3/lib/python3.9/site-packages/sklearn/linear\_model/\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.  
  
Increase the number of iterations (max\_iter) or scale the data as shown in:  
 https://scikit-learn.org/stable/modules/preprocessing.html  
Please also refer to the documentation for alternative solver options:  
 https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression  
 n\_iter\_i = \_check\_optimize\_result(  
/home/dnikolskiy/anaconda3/lib/python3.9/site-packages/sklearn/linear\_model/\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.  
  
Increase the number of iterations (max\_iter) or scale the data as shown in:  
 https://scikit-learn.org/stable/modules/preprocessing.html  
Please also refer to the documentation for alternative solver options:  
 https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression  
 n\_iter\_i = \_check\_optimize\_result(  
/home/dnikolskiy/anaconda3/lib/python3.9/site-packages/sklearn/linear\_model/\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.  
  
Increase the number of iterations (max\_iter) or scale the data as shown in:  
 https://scikit-learn.org/stable/modules/preprocessing.html  
Please also refer to the documentation for alternative solver options:  
 https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression  
 n\_iter\_i = \_check\_optimize\_result(

Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,  
 '0000000005': 4, '0000000667': 5, '0000001200': 6,  
 '0001': 7, '00014': 8, '0002': 9, '0003': 10,  
 '0005111312': 11, '0005111312na1em': 12,  
 '00072': 13, '000851': 14, '000rpm': 15,  
 '000th': 16, '001': 17, '0010': 18, '001004': 19,  
 '0011': 20, '001211': 21, '0013': 22, '001642': 23,  
 '001813': 24, '002': 25, '002222': 26,  
 '002251w': 27, '0023': 28, '002937': 29, ...})  
Модель для классификации - LogisticRegression(C=3.0)  
Accuracy = 0.9382336841146768  
===========================  
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,  
 '0000000005': 4, '0000000667': 5, '0000001200': 6,  
 '0001': 7, '00014': 8, '0002': 9, '0003': 10,  
 '0005111312': 11, '0005111312na1em': 12,  
 '00072': 13, '000851': 14, '000rpm': 15,  
 '000th': 16, '001': 17, '0010': 18, '001004': 19,  
 '0011': 20, '001211': 21, '0013': 22, '001642': 23,  
 '001813': 24, '002': 25, '002222': 26,  
 '002251w': 27, '0023': 28, '002937': 29, ...})  
Модель для классификации - LinearSVC()  
Accuracy = 0.9453742497059174  
===========================  
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000000004': 3,  
 '0000000005': 4, '0000000667': 5, '0000001200': 6,  
 '0001': 7, '00014': 8, '0002': 9, '0003': 10,  
 '0005111312': 11, '0005111312na1em': 12,  
 '00072': 13, '000851': 14, '000rpm': 15,  
 '000th': 16, '001': 17, '0010': 18, '001004': 19,  
 '0011': 20, '001211': 21, '0013': 22, '001642': 23,  
 '001813': 24, '002': 25, '002222': 26,  
 '002251w': 27, '0023': 28, '002937': 29, ...})  
Модель для классификации - KNeighborsClassifier()  
Accuracy = 0.6655358653541747  
===========================

#Разделим выборку на обучающую и тестовую и проверим решение для лучшей модели

X\_train, X\_test, y\_train, y\_test = train\_test\_split(newsgroups['data'], newsgroups['target'], test\_size=0.5, random\_state=1)

def sentiment(v, c):  
 model = Pipeline(  
 [("vectorizer", v),   
 ("classifier", c)])  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
 print\_accuracy\_score\_for\_classes(y\_test, y\_pred)

sentiment(CountVectorizer(), LinearSVC())

Метка Accuracy  
0 0.9290322580645162  
1 0.9675090252707581  
2 0.9026845637583892  
3 0.9245901639344263

#Работа с векторными представлениями слов с использованием word2vec

import gensim  
from gensim.models import word2vec

model\_path = 'ruscorpora\_mystem\_cbow\_300\_2\_2015.bin.gz'

model = gensim.models.KeyedVectors.load\_word2vec\_format(model\_path, binary=True)

words = ['холод\_S', 'мороз\_S', 'береза\_S', 'сосна\_S']

for word in words:  
 if word in model:  
 print('\nСЛОВО - {}'.format(word))  
 print('5 ближайших соседей слова:')  
 for word, sim in model.most\_similar(positive=[word], topn=5):  
 print('{} => {}'.format(word, sim))  
 else:  
 print('Слово "{}" не найдено в модели'.format(word))

СЛОВО - холод\_S  
5 ближайших соседей слова:  
стужа\_S => 0.7676383852958679  
сырость\_S => 0.6338975429534912  
жара\_S => 0.6089427471160889  
мороз\_S => 0.589036762714386  
озноб\_S => 0.5776054859161377  
  
СЛОВО - мороз\_S  
5 ближайших соседей слова:  
стужа\_S => 0.6425478458404541  
морозец\_S => 0.5947279930114746  
холод\_S => 0.589036762714386  
жара\_S => 0.5522176623344421  
снегопад\_S => 0.5083199143409729  
  
СЛОВО - береза\_S  
5 ближайших соседей слова:  
сосна\_S => 0.7943246960639954  
тополь\_S => 0.7562226057052612  
дуб\_S => 0.7440178394317627  
дерево\_S => 0.7373415231704712  
клен\_S => 0.7105200886726379  
  
СЛОВО - сосна\_S  
5 ближайших соседей слова:  
береза\_S => 0.7943247556686401  
дерево\_S => 0.758143424987793  
лиственница\_S => 0.7478148937225342  
дуб\_S => 0.7412480711936951  
ель\_S => 0.7363824844360352

#Находим близость между словами и строим аналогии

print(model.similarity('сосна\_S', 'береза\_S'))

0.79432476

print(model.most\_similar(positive=['холод\_S', 'стужа\_S'], negative=['мороз\_S']))

[('сырость\_S', 0.5040210485458374), ('стылость\_S', 0.46336129307746887), ('голод\_S', 0.4604816436767578), ('зной\_S', 0.45904630422592163), ('скука\_S', 0.4489358067512512), ('жара\_S', 0.44645124673843384), ('усталость\_S', 0.4218570291996002), ('озноб\_S', 0.41469815373420715), ('духота\_S', 0.4099087715148926), ('неуют\_S', 0.40298786759376526)]

#Обучим word2vec на наборе данных "fetch\_20newsgroups"

import re  
import pandas as pd  
import numpy as np  
from typing import Dict, Tuple  
from sklearn.metrics import accuracy\_score, balanced\_accuracy\_score  
from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer  
from sklearn.linear\_model import LogisticRegression  
from sklearn.pipeline import Pipeline  
from nltk import WordPunctTokenizer  
from nltk.corpus import stopwords  
import nltk  
nltk.download('stopwords')

[nltk\_data] Downloading package stopwords to  
[nltk\_data] /home/dnikolskiy/nltk\_data...  
[nltk\_data] Unzipping corpora/stopwords.zip.

True

categories = ["rec.motorcycles", "rec.sport.baseball", "sci.electronics","sci.med"]  
newsgroups = fetch\_20newsgroups(subset='train', categories=categories)  
data = newsgroups['data']

# Подготовим корпус  
corpus = []  
stop\_words = stopwords.words('english')  
tok = WordPunctTokenizer()  
for line in newsgroups['data']:  
 line1 = line.strip().lower()  
 line1 = re.sub("[^a-zA-Z]"," ", line1)  
 text\_tok = tok.tokenize(line1)  
 text\_tok1 = [w for w in text\_tok if not w in stop\_words]  
 corpus.append(text\_tok1)

corpus[:5]

[['nrmendel',  
 'unix',  
 'amherst',  
 'edu',  
 'nathaniel',  
 'mendell',  
 'subject',  
 'bike',  
 'advice',  
 'organization',  
 'amherst',  
 'college',  
 'x',  
 'newsreader',  
 'tin',  
 'version',  
 'pl',  
 'lines',  
 'ummm',  
 'bikes',  
 'kx',  
 'suggest',  
 'look',  
 'zx',  
 'since',  
 'horsepower',  
 'whereas',  
 'might',  
 'bit',  
 'much',  
 'sincerely',  
 'nathaniel',  
 'zx',  
 'dod',  
 'ama'],  
 ['grante',  
 'aquarius',  
 'rosemount',  
 'com',  
 'grant',  
 'edwards',  
 'subject',  
 'krillean',  
 'photography',  
 'reply',  
 'grante',  
 'aquarius',  
 'rosemount',  
 'com',  
 'grant',  
 'edwards',  
 'organization',  
 'rosemount',  
 'inc',  
 'lines',  
 'nntp',  
 'posting',  
 'host',  
 'aquarius',  
 'stgprao',  
 'st',  
 'unocal',  
 'com',  
 'richard',  
 'ottolini',  
 'writes',  
 'living',  
 'things',  
 'maintain',  
 'small',  
 'electric',  
 'fields',  
 'enhance',  
 'certain',  
 'chemical',  
 'reactions',  
 'promote',  
 'communication',  
 'states',  
 'cell',  
 'communicate',  
 'cells',  
 'nervous',  
 'system',  
 'specialized',  
 'example',  
 'perhaps',  
 'uses',  
 'true',  
 'electric',  
 'fields',  
 'change',  
 'location',  
 'time',  
 'large',  
 'organism',  
 'also',  
 'true',  
 'special',  
 'photographic',  
 'techniques',  
 'applying',  
 'external',  
 'fields',  
 'kirillian',  
 'photography',  
 'interact',  
 'fields',  
 'resistances',  
 'caused',  
 'fields',  
 'make',  
 'interesting',  
 'pictures',  
 'really',  
 'kirlian',  
 'photography',  
 'taking',  
 'pictures',  
 'corona',  
 'discharge',  
 'objects',  
 'animate',  
 'inanimate',  
 'fields',  
 'applied',  
 'objects',  
 'millions',  
 'times',  
 'larger',  
 'biologically',  
 'created',  
 'fields',  
 'want',  
 'record',  
 'biologically',  
 'created',  
 'electric',  
 'fields',  
 'got',  
 'use',  
 'low',  
 'noise',  
 'high',  
 'gain',  
 'sensors',  
 'typical',  
 'eegs',  
 'ekgs',  
 'kirlian',  
 'photography',  
 'phun',  
 'physics',  
 'type',  
 'stuff',  
 'right',  
 'soaking',  
 'chunks',  
 'extra',  
 'fine',  
 'steel',  
 'wool',  
 'liquid',  
 'oxygen',  
 'hitting',  
 'hammer',  
 'like',  
 'kirlean',  
 'setup',  
 'fun',  
 'possibly',  
 'dangerous',  
 'perhaps',  
 'pictures',  
 'diagonistic',  
 'disease',  
 'problems',  
 'organisms',  
 'better',  
 'understood',  
 'perhaps',  
 'probably',  
 'grant',  
 'edwards',  
 'yow',  
 'vote',  
 'rosemount',  
 'inc',  
 'well',  
 'tapered',  
 'half',  
 'cocked',  
 'ill',  
 'conceived',  
 'grante',  
 'aquarius',  
 'rosemount',  
 'com',  
 'tax',  
 'deferred'],  
 ['liny',  
 'sun',  
 'scri',  
 'fsu',  
 'edu',  
 'nemo',  
 'subject',  
 'bates',  
 'method',  
 'myopia',  
 'reply',  
 'lin',  
 'ray',  
 'met',  
 'fsu',  
 'edu',  
 'distribution',  
 'na',  
 'organization',  
 'scri',  
 'florida',  
 'state',  
 'university',  
 'lines',  
 'bates',  
 'method',  
 'work',  
 'first',  
 'heard',  
 'newsgroup',  
 'several',  
 'years',  
 'ago',  
 'got',  
 'hold',  
 'book',  
 'improve',  
 'sight',  
 'simple',  
 'daily',  
 'drills',  
 'relaxation',  
 'margaret',  
 'corbett',  
 'authorized',  
 'instructor',  
 'bates',  
 'method',  
 'published',  
 'talks',  
 'vision',  
 'improvement',  
 'relaxation',  
 'exercise',  
 'study',  
 'whether',  
 'method',  
 'actually',  
 'works',  
 'works',  
 'actually',  
 'shortening',  
 'previously',  
 'elongated',  
 'eyeball',  
 'increasing',  
 'lens',  
 'ability',  
 'flatten',  
 'order',  
 'compensate',  
 'long',  
 'eyeball',  
 'since',  
 'myopia',  
 'result',  
 'eyeball',  
 'elongation',  
 'seems',  
 'logical',  
 'approach',  
 'correction',  
 'find',  
 'way',  
 'reverse',  
 'process',  
 'e',  
 'shorten',  
 'somehow',  
 'preferably',  
 'non',  
 'surgically',  
 'recent',  
 'studies',  
 'find',  
 'know',  
 'rk',  
 'works',  
 'changing',  
 'curvature',  
 'cornea',  
 'compensate',  
 'shape',  
 'eyeball',  
 'way',  
 'train',  
 'muscles',  
 'shorten',  
 'eyeball',  
 'back',  
 'correct',  
 'length',  
 'would',  
 'even',  
 'better',  
 'bates',  
 'idea',  
 'right',  
 'thanks',  
 'information'],  
 ['mcovingt',  
 'aisun',  
 'ai',  
 'uga',  
 'edu',  
 'michael',  
 'covington',  
 'subject',  
 'buy',  
 'parts',  
 'time',  
 'nntp',  
 'posting',  
 'host',  
 'aisun',  
 'ai',  
 'uga',  
 'edu',  
 'organization',  
 'ai',  
 'programs',  
 'university',  
 'georgia',  
 'athens',  
 'lines',  
 'pricing',  
 'parts',  
 'reminds',  
 'something',  
 'chemist',  
 'said',  
 'gram',  
 'dye',  
 'costs',  
 'dollar',  
 'comes',  
 'liter',  
 'jar',  
 'also',  
 'costs',  
 'dollar',  
 'want',  
 'whole',  
 'barrel',  
 'also',  
 'costs',  
 'dollar',  
 'e',  
 'charge',  
 'almost',  
 'exclusively',  
 'packaging',  
 'delivering',  
 'chemical',  
 'particular',  
 'case',  
 'byproduct',  
 'cost',  
 'almost',  
 'nothing',  
 'intrinsically',  
 'michael',  
 'covington',  
 'associate',  
 'research',  
 'scientist',  
 'artificial',  
 'intelligence',  
 'programs',  
 'mcovingt',  
 'ai',  
 'uga',  
 'edu',  
 'university',  
 'georgia',  
 'phone',  
 'athens',  
 'georgia',  
 'u',  
 'amateur',  
 'radio',  
 'n',  
 'tmi'],  
 ['tammy',  
 'vandenboom',  
 'launchpad',  
 'unc',  
 'edu',  
 'tammy',  
 'vandenboom',  
 'subject',  
 'sore',  
 'spot',  
 'testicles',  
 'nntp',  
 'posting',  
 'host',  
 'lambada',  
 'oit',  
 'unc',  
 'edu',  
 'organization',  
 'university',  
 'north',  
 'carolina',  
 'extended',  
 'bulletin',  
 'board',  
 'service',  
 'distribution',  
 'na',  
 'lines',  
 'husband',  
 'woke',  
 'three',  
 'days',  
 'ago',  
 'small',  
 'sore',  
 'spot',  
 'spot',  
 'size',  
 'nickel',  
 'one',  
 'testicles',  
 'bottom',  
 'side',  
 'knots',  
 'lumps',  
 'little',  
 'sore',  
 'spot',  
 'says',  
 'reminds',  
 'bruise',  
 'feels',  
 'recollection',  
 'hitting',  
 'anything',  
 'like',  
 'would',  
 'cause',  
 'bruise',  
 'asssures',  
 'remember',  
 'something',  
 'like',  
 'clues',  
 'might',  
 'somewhat',  
 'hypochondriac',  
 'sp',  
 'sure',  
 'gonna',  
 'die',  
 'thanks',  
 'opinions',  
 'expressed',  
 'necessarily',  
 'university',  
 'north',  
 'carolina',  
 'chapel',  
 'hill',  
 'campus',  
 'office',  
 'information',  
 'technology',  
 'experimental',  
 'bulletin',  
 'board',  
 'service',  
 'internet',  
 'launchpad',  
 'unc',  
 'edu']]

%time model\_imdb = word2vec.Word2Vec(corpus, workers=4, min\_count=10, window=10, sample=1e-3)

CPU times: user 3.76 s, sys: 19.1 ms, total: 3.78 s  
Wall time: 1.16 s

# Проверим, что модель обучилась  
print(model\_imdb.wv.most\_similar(positive=['find'], topn=5))

[('etc', 0.9759538173675537), ('voltage', 0.9697504639625549), ('buy', 0.9697079658508301), ('circuit', 0.9695694446563721), ('work', 0.9691796898841858)]

def sentiment\_2(v, c):  
 model = Pipeline(  
 [("vectorizer", v),   
 ("classifier", c)])  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
 print\_accuracy\_score\_for\_classes(y\_test, y\_pred)

#Проверка качества работы модели word2vec

class EmbeddingVectorizer(object):  
 '''  
 Для текста усредним вектора входящих в него слов  
 '''  
 def \_\_init\_\_(self, model):  
 self.model = model  
 self.size = model.vector\_size  
  
 def fit(self, X, y):  
 return self  
  
 def transform(self, X):  
 return np.array([np.mean(  
 [self.model[w] for w in words if w in self.model]   
 or [np.zeros(self.size)], axis=0)  
 for words in X])

def accuracy\_score\_for\_classes(  
 y\_true: np.ndarray,   
 y\_pred: np.ndarray) -> Dict[int, float]:  
 """  
 Вычисление метрики accuracy для каждого класса  
 y\_true - истинные значения классов  
 y\_pred - предсказанные значения классов  
 Возвращает словарь: ключ - метка класса,   
 значение - Accuracy для данного класса  
 """  
 # Для удобства фильтрации сформируем Pandas DataFrame   
 d = {'t': y\_true, 'p': y\_pred}  
 df = pd.DataFrame(data=d)  
 # Метки классов  
 classes = np.unique(y\_true)  
 # Результирующий словарь  
 res = dict()  
 # Перебор меток классов  
 for c in classes:  
 # отфильтруем данные, которые соответствуют   
 # текущей метке класса в истинных значениях  
 temp\_data\_flt = df[df['t']==c]  
 # расчет accuracy для заданной метки класса  
 temp\_acc = accuracy\_score(  
 temp\_data\_flt['t'].values,   
 temp\_data\_flt['p'].values)  
 # сохранение результата в словарь  
 res[c] = temp\_acc  
 return res  
  
def print\_accuracy\_score\_for\_classes(  
 y\_true: np.ndarray,   
 y\_pred: np.ndarray):  
 """  
 Вывод метрики accuracy для каждого класса  
 """  
 accs = accuracy\_score\_for\_classes(y\_true, y\_pred)  
 if len(accs)>0:  
 print('Метка \t Accuracy')  
 for i in accs:  
 print('{} \t {}'.format(i, accs[i]))

# Обучающая и тестовая выборки  
boundary = 1500  
X\_train = corpus[:boundary]   
X\_test = corpus[boundary:]  
y\_train = newsgroups['target'][:boundary]  
y\_test = newsgroups['target'][boundary:]

sentiment\_2(EmbeddingVectorizer(model\_imdb.wv), LogisticRegression(C=5.0))

/home/dnikolskiy/anaconda3/lib/python3.9/site-packages/sklearn/linear\_model/\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.  
  
Increase the number of iterations (max\_iter) or scale the data as shown in:  
 https://scikit-learn.org/stable/modules/preprocessing.html  
Please also refer to the documentation for alternative solver options:  
 https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression  
 n\_iter\_i = \_check\_optimize\_result(

Метка Accuracy  
0 0.868421052631579  
1 0.9368932038834952  
2 0.8073394495412844  
3 0.7719298245614035

### *Как видно из результатов проверки качества моделей, лучшее качество показал CountVectorizer*