Ввод [1]:

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from IPython.display import Image
from sklearn.datasets import load_iris, load_boston
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
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.naive_bayes import ComplementNB
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
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Ввод [2]:

```
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]))
```

Ввод [37]:

```
# Загрузка данных
df = pd.read_csv('D:\\Ботва\\Магистратура\\2сем\\ММО\\лаб6\\imdb_sup.csv')
text_df=df.head(500).append(df.tail(500))
text_df.drop('Rating', axis=1, inplace=True)
text_df.tail(15)
```

Out[37]:

	Review	Sentiment
49985	I had great expectations surrounding this movi...	0
49986	It is playing on SHOWTIME right now but is goi...	0
49987	I love the so-called "blaxploitation" films an...	0
49988	OK, here is the deal. I love action movies and...	0
49989	Grim instead of amusing, mean-spirited instead...	0
49990	This movie did not give Mr. Bachchan justice. ...	0
49991	Oh dear! The BBC is not about to be knocked of...	0
49992	Ridiculous thriller in which a group of studen...	0
49993	If you like poor SE, (some) bad acting and a t...	0
49994	Some Plot Spoilers Ahead.<br /><br />The Nashv...	0
49995	(spoiler) it could be the one the worst movie ...	0
49996	So, you've seen the Romero movies, yes? And yo...	0
49997	Just listen to the Broadway cast album and to ...	0
49998	I have been a fan of the Carpenters for a long...	0
49999	Set in 1945, Skenbart follows a failed Swedish...	0

Ввод [98]:

```
#Кодирование целевого признаков
text_df.loc[text_df['Sentiment'] == 1, 'Sentiment'] = 'pol'
text_df.loc[text_df['Sentiment'] == 0, 'Sentiment'] = 'otr'
text_df.sort_values(by=['Review'], inplace=True)
text_df
```

Out[98]:

	Review	Sentiment
49906	'Bloody Birthday' is an odd and, at times, hum...	otr
181	'Renaissance (2006)' was created over a period...	pol
49867	(SMALL SPOILERS) I just bought the DVD of this...	otr
49995	(spoiler) it could be the one the worst movie ...	otr
49834	**********POSSIBLE SPOILER********** Madonna p...	otr
...	...	...
483	this one is out there. Not much to say about i...	pol
283	very few chess movies have been made over the ...	pol
492	well, i said it all in the summary, i simpley ...	pol
381	when i first heard about this movie i thought ...	pol
49888	wow, i just got one watching this.<br /><br />...	otr

1000 rows × 2 columns

Ввод [99]:

```
text_df.shape
```

Out[99]:

```
(1000, 2)
```

Ввод [100]:

```
# Сформируем общий словарь для обучения моделей из обучающей и тестовой выборки
vocab_list = text_df['Review'].tolist()
vocab_list[1:10]
```

```
 '(SMALL SPOILERS) I just bought the DVD of this movie yesterday. I saw it
with my friends and I couldn\'t believe what had happened.<br /><br />In t
he first 3 movies, the critters at least had a sense of humor (especially
the 3rd movie), but not only did the critters barely ever make an appearan
ce, they weren\'t funny! They never made me laugh. I must admit that the s
tory did start off nicely. After an hour had gone by I remembered that the
Critters movies were always very short. So I thought to myself, "Where the
$^%#$ are the critters?!?!" They were barely in this movie! If that didn
\'t make me mad enough, the boy named Ethan was sitting on his bed after C
harlie had "murdered the ship" and he knew that the critters were still on
board! In the first movie the Brown family was scared out of their minds.
But here, Ethan didn\'t even care! It was as if the critters weren\'t even
a threat!<br /><br />Now what I\'m about to say next may ruin the ending,
but I\'m going to say it anyways. In the first movie, at the end, they had
to face the giant critter for a final battle. In the second one, there was
the great ball of critter. In the third movie, the critter with his fave b
urned did a spindash (from Sonic the Hedgehog) and was going to attack the
little kid. But at the end of the fourth one (which is what made me the an
griest) the bald critter charges toward Ethan, and Ethan kills it as if it
were nothing.<br /><br />Now something that I really don\'t understand was
```

Ввод [101]:

```
vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 17896
```

Ввод [102]:

```
for i in list(corpusVocab)[0:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
bloody=1874
birthday=1772
is=8498
an=781
odd=11070
and=798
at=1158
times=16139
humorous=7838
low=9560
```

Векторизация признаков на основе CountVectorizer

Подсчитывает количество слов словаря, входящих в данный текст

Ввод [103]:

```
test_features = vocabVect.transform(vocab_list)
```

Ввод [104]:

```
test_features
```

Out[104]:

```
<1000x17896 sparse matrix of type '<class 'numpy.int64'>'
	with 137926 stored elements in Compressed Sparse Row format>
```

Ввод [105]:

```
test_features.todense()
```

Out[105]:

```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Ввод [106]:

```
# Размер нулевой строки
len(test_features.todense()[0].getA1())
```

Out[106]:

```
17896
```

Ввод [107]:

```
# Непустые значения нулевой строки
[i for i in test_features.todense()[0].getA1() if i>0]
```

Out[107]:

```
[1,
 1,
 2,
 1,
 1,
 1,
 1,
 2,
 7,
 1,
 3,
 1,
 5,
 3,
 1,
 2,
 1,
 2,
```

Ввод [108]:

```
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, text_df['Review'], text_df['Sentiment'], sco
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('===========================')
```

Ввод [109]:

```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab)]
classifiers_list = [LogisticRegression(C=3.0), LinearSVC(), KNeighborsClassifier()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\sklearn\li
near_model\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (s
tatus=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 (https://scik
it-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-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\sklearn\li
near_model\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (s
tatus=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 (https://scik
it-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-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\sklearn\li
near_model\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (s
tatus=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 (https://scik
it-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-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0069': 2,
'007': 3, '01': 4,
                            '06th': 5, '08': 6, '0f': 7, '10': 8, '100':
9,
                            '100th': 10, '101': 11, '102': 12, '10th': 13,
                            '11': 14, '112': 15, '11th': 16, '12': 17, '1
3': 18,
                            '13th': 19, '14': 20, '14th': 21, '15': 22,
                            '150': 23, '16': 24, '1600s': 25, '16éme': 26,
                            '17': 27, '1710': 28, '18': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.7819855783927641
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0069': 2,
'007': 3, '01': 4,
                            '06th': 5, '08': 6, '0f': 7, '10': 8, '100':
9,
                            '100th': 10, '101': 11, '102': 12, '10th': 13,
                            '11': 14, '112': 15, '11th': 16, '12': 17, '1
3': 18,
                            '13th': 19, '14': 20, '14th': 21, '15': 22,
                            '150': 23, '16': 24, '1600s': 25, '16éme': 26,
                            '17': 27, '1710': 28, '18': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.779977582372792
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0069': 2,
'007': 3, '01': 4,
                            '06th': 5, '08': 6, '0f': 7, '10': 8, '100':
9,
                            '100th': 10, '101': 11, '102': 12, '10th': 13,
                            '11': 14, '112': 15, '11th': 16, '12': 17, '1
3': 18,
                            '13th': 19, '14': 20, '14th': 21, '15': 22,
                            '150': 23, '16': 24, '1600s': 25, '16éme': 26,
                            '17': 27, '1710': 28, '18': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.5729981478484473
===========================
```

Разделим на обучающую и тестовую выборки

Ввод [110]:

```
X_train, X_test, y_train, y_test = train_test_split(text_df['Review'], text_df['Sentiment']
```

Ввод [113]:

```
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)
```

Ввод [114]:

```
sentiment(CountVectorizer(), LogisticRegression(C=3.0))
```

```
Метка    Accuracy
otr      0.7765151515151515
pol      0.7669491525423728

c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\sklearn\li
near_model\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (s
tatus=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 (https://scik
it-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-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Классификация текста на основе моделей word2vec

Ввод [115]:

```
import gensim
from gensim.models import word2vec
```

Ввод [116]:

```
import re
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]     C:\Users\sveta\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[116]:

```
True
```

Ввод [117]:

```
# Подготовим корпус
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in text_df['Review'].values:
    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)
```

Ввод [118]:

```
corpus[:10]
```

```
  'humorous',
  'low',
  'budget',
  'horror',
  'flick',
  'along',
  'lines',
  'mikey',
  'less',
  'intelligent',
  'version',
  'good',
  'son',
  'br',
  'br',
  'set',
  'small',
  'californian',
  'town',
  'three'
```

Ввод [119]:

```
%time
model_imdb = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=10, sample=1e-3)
```

```
Wall time: 0 ns
```

Ввод [120]:

```
# Проверим, что модель обучилась
print(model_imdb.wv.most_similar(positive=['find'], topn=5))
```

```
[('someone', 0.9996957778930664), ('audience', 0.9996758103370667), ('far',
0.9996746182441711), ('everything', 0.9996718168258667), ('nothing', 0.99967
10419654846)]
```

Ввод [121]:

```
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)
```

Ввод [122]:

```
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])
```

Ввод [123]:

```
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]))
```

Ввод [128]:

```
# Обучающая и тестовая выборки
boundary = 900
X_train = corpus[:boundary]
X_test = corpus[boundary:]
y_train = text_df['Sentiment'][:boundary]
y_test = text_df['Sentiment'][boundary:]
```

Ввод [129]:

```
sentiment(EmbeddingVectorizer(model_imdb.wv), LogisticRegression(C=3.0))
```

```
Метка    Accuracy
otr      0.5319148936170213
pol      0.5094339622641509
```

Наибольшая точность получилась при использовании CountVectorizer и LogisticRegression