МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

Лабораторная работа №6
по курсу «Методы машинного обучения»

ИСПОЛНИТЕЛЬНИЦА: Абросимова Н.Г.
ИУ5-24М

____
подпись

"__"_________2021 г.

ПРЕПОДАВАТЕЛЬ: ____________________
ФИО

____
подпись

"__"_________2021 г.

Москва - 2021

Задание

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

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

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

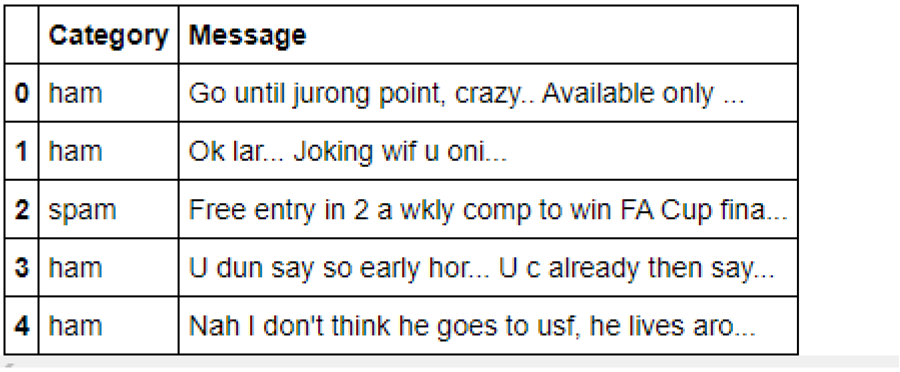
Текст программы и экранные формы

```
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.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.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")

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]))
```

```
# Загрузка данных
df = pd.read_csv('SPAM.csv')
df.head()
```



	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
df.shape
(5572, 2)

# Сформируем общий словарь для обучения моделей из обучающей и тестовой выбор
ки
vocab_list = df['Message'].tolist()
vocab_list[1:10]
Out[22]:
['Ok lar... Joking wif u oni...',
 "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA
to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18
's",
 'U dun say so early hor... U c already then say...',
 "Nah I don't think he goes to usf, he lives around here though",
 "FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like
some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv",
 'Even my brother is not like to speak with me. They treat me like aids paten
t.',
 "As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has
been set as your callertune for all Callers. Press *9 to copy your friends Ca
llertune",
 'WINNER!! As a valued network customer you have been selected to receivea £9
00 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours
only.',
 'Had your mobile 11 months or more? U R entitled to Update to the latest col
our mobiles with camera for Free! Call The Mobile Update Co FREE on 080029860
30']
```

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

```
Количество сформированных признаков - 8709
```

In [24]:

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

```
until=8080
jurong=4370
point=5954
crazy=2334
available=1313
only=5567
in=4110
bugis=1763
great=3651
```

CountVectorizer

In [25]:

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

In [26]:

```
test_features
```

Out[26]:

```
<5572x8709 sparse matrix of type '<class 'numpy.int64'>'
        with 74098 stored elements in Compressed Sparse Row format>
```

In [27]:

```
test_features.todense()
```

Out[27]:

```
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)
```

```
In [28]:
# Размер нулевой строки
len(test_features.todense()[0].getA1())
Out[28]:
8709
In [29]:
# Непустые значения нулевой строки
[i for i in test_features.todense()[0].getA1() if i>0]
Out[29]:
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
In [30]:
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, df['Message'], df['Category'],
scoring='accuracy', cv=3).mean()
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('===========================')
In [31]:
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab)]
classifiers_list = [LogisticRegression(C=3.0), LinearSVC(), KNeighborsClassif
ier()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '000pes': 2, '0
08704050406': 3,
                            '0089': 4, '0121': 5, '01223585236': 6,
                            '01223585334': 7, '0125698789': 8, '02': 9,
                            '0207': 10, '02072069400': 11, '02073162414': 12,
                            '02085076972': 13, '021': 14, '03': 15, '04': 16,
                            '0430': 17, '05': 18, '050703': 19, '0578': 20,
                            '06': 21, '07': 22, '07008009200': 23,
                            '07046744435': 24, '07090201529': 25,
                            '07090298926': 26, '07099833605': 27,
                            '07123456789': 28, '0721072': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.982591785578825
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '000pes': 2, '0
08704050406': 3,
                            '0089': 4, '0121': 5, '01223585236': 6,
                            '01223585334': 7, '0125698789': 8, '02': 9,
                            '0207': 10, '02072069400': 11, '02073162414': 12,
                            '02085076972': 13, '021': 14, '03': 15, '04': 16,
                            '0430': 17, '05': 18, '050703': 19, '0578': 20,
                            '06': 21, '07': 22, '07008009200': 23,
                            '07046744435': 24, '07090201529': 25,
                            '07090298926': 26, '07099833605': 27,
                            '07123456789': 28, '0721072': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.9834887108563705
============================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '000pes': 2, '0
08704050406': 3,
                            '0089': 4, '0121': 5, '01223585236': 6,
                            '01223585334': 7, '0125698789': 8, '02': 9,
                            '0207': 10, '02072069400': 11, '02073162414': 12,
                            '02085076972': 13, '021': 14, '03': 15, '04': 16,
                            '0430': 17, '05': 18, '050703': 19, '0578': 20,
                            '06': 21, '07': 22, '07008009200': 23,
                            '07046744435': 24, '07090201529': 25,
                            '07090298926': 26, '07099833605': 27,
                            '07123456789': 28, '0721072': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.9122387985297536
============================
```

In [32]:

```
X_train, X_test, y_train, y_test = train_test_split(df['Message'], df['Catego
ry'], test_size=0.5, random_state=1)
```

In [ ]:

```
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)
```

In [33]:

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

```
Метка    Accuracy
ham      0.9975031210986267
spam     0.8616187989556136
```

word2vec

In [1]:

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

```
c:\users\надя\appdata\local\programs\python\python37\lib\site-packages\gensim
\similarities\__init__.py:15: UserWarning: The gensim.similarities.levenshtei
n submodule is disabled, because the optional Levenshtein package <https://py
pi.org/project/python-Levenshtein/> is unavailable. Install Levenhstein (e.g.
`pip install python-Levenshtein`) to suppress this warning.
  warnings.warn(msg)
```

In [2]:

```
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]     C:\Users\Надя\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[2]:

```
True
```

In [5]:

```
# Подготовим корпус
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in df['Message'].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)
```

In [6]:

```
corpus[:5]
```

Out[6]:

```
[['go',
  'jurong',
  'point',
  'crazy',
  'available',
  'bugis',
  'n',
  'great',
  'world',
  'la',
  'e',
  'buffet',
  'cine',
  'got',
  'amore',
  'wat'],
 ['ok', 'lar', 'joking', 'wif', 'u', 'oni'],
 ['free',
  'entry',
  'wkly',
  'comp',
  'win',
  'fa',
  'cup',
  'final',
  'tkts',
  'st',
  'may',
  'text',
  'fa',
  'receive',
  'entry',
  'question',
  'std',
  'txt',
  'rate',
  'c',
  'apply'],
 ['u', 'dun', 'say', 'early', 'hor', 'u', 'c', 'already', 'say'],
 ['nah', 'think', 'goes', 'usf', 'lives', 'around', 'though']]
```

In [7]:

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

```
Wall time: 415 ms
```

In [8]:

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

```
[('keep', 0.9995377659797668), ('someone', 0.9994912147521973), ('yet', 0.999
4739890098572), ('next', 0.9994291663169861), ('special', 0.9994221329689026)
]
```

In [9]:

```
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)
```

In [10]:

```
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])
```

In [11]:

```
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]))
```

In [15]:

```
# Обучающая и тестовая выборки
boundary = 700
X_train = corpus[:boundary]
X_test = corpus[boundary:]
y_train = df['Category'][:boundary]
y_test = df['Category'][boundary:]
```

In [17]:

```
sentiment(EmbeddingVectorizer(model_imdb.wv), LogisticRegression(C=5.0))
```

```
Метка    Accuracy
ham      0.9933727810650888
spam     0.36476043276661513
```