ЛР №6 по курсу "Методы машинного обучения"

Классификация текста.

Чжан Чжибо ИУ5И-21М

Цель лабораторной работы: изучение методов классификации текстов.

Задание:

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.

```
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,
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
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")


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)
```

```
    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 соответствую
        # текущей метке класса в истинных значения
        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:

```
%cd /content/drive/MyDrive/dataset/spam/
```

```
/content/drive/MyDrive/dataset/spam
```

```
dataset = pd.read_csv("enron_spam_data.csv")
dataset.head()
```

	Unnamed: 0	Subject	Message	Sp
0	0	christmas tree farm pictures	NaN	
1	1	vastar resources , inc .	gary , production from the high island larger ...	
2	2	calpine daily gas nomination	- calpine daily gas nomination 1 . doc	

```
dataset=dataset.drop(['Unnamed: 0','Subject','Date'],axis=1)
dataset.head()
```

	Message	Spam/Ham
0	NaN	ham
1	gary , production from the high island larger ...	ham
2	- calpine daily gas nomination 1 . doc	ham
3	fyi - see note below - already done .\nstella\...	ham
4	fyi .\n- - - - - - - - - - - - - - - - - - -...	ham

```
dataset['Spam/Ham']=dataset['Spam/Ham'].replace(['ham','spam'],[0,1])
dataset.head()
```

	Message	Spam/Ham
0	NaN	0
1	gary , production from the high island larger ...	0
2	- calpine daily gas nomination 1 . doc	0
3	fyi - see note below - already done .\nstella\...	0
4	fyi .\n- - - - - - - - - - - - - - - - - - -...	0

```
dataset=dataset.dropna()
dataset.head()
```

	Message	Spam/Ham
1	gary , production from the high island larger ...	0
2	- calpine daily gas nomination 1 . doc	0
3	fyi - see note below - already done .\nstella\...	0
4	fyi .\n- - - - - - - - - - - - - - - - - - -...	0
5	jackie ,\nsince the inlet to 3 river plant is ...	0

```
dataset=dataset.sample(frac=1)
dataset.head()
```

	Message	Spam/Ham
7683	gentleman ,\nkevin presto concurred on the pur...	0
434	daren or stacey : could you please extend deal...	0
19561	start date : 2 / 6 / 02 ; hourahead hour : 24 ...	1
31175	fyi , kim .\n- - - - - original message - - - ...	1
27668	attached is the latest version of the cost cen...	1

```
dataset.describe()
```

	Spam/Ham
count	33664.000000
mean	0.510070
std	0.499906
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

```
train_df=(dataset.iloc[0:26664,:])
test_df=(dataset.iloc[26664:33664,:])
```

```
train_df.describe()
```

	Spam/Ham
count	26664.000000
mean	0.509338
std	0.499922
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

```
test_df.describe()
```

	Spam/Ham
count	7000.000000
mean	0.512857
std	0.499870
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

Способ 1. На основе CountVectorizer или TfidfVectorizer.

Сформируем общий словарь для обучения моделей из обучающей и тестовой выборки

```
vocab_list = train_df['Message'].tolist()
vocab_list[1:10]
```

```
['daren or stacey : could you please extend deal # 169625 for meter 1520 for\nfeb . 2000 ?\n
 "start date : 2 / 6 / 02 ; hourahead hour : 24 ; no ancillary schedules awarded . no varian
 'fyi , kim .\n- - - - - original message - - - - -\nfrom : frazier , perry\nsent : thursday
 "attached is the latest version of the cost center assignments for the transfers out of ees
 'fyi , kim .\n- - - - - original message - - - - -\nfrom : frazier , perry\nsent : thursday
 "start date : 12 / 31 / 01 ; hourahead hour : 6 ; no ancillary schedules awarded . no varia
 "rick ,\naram is coming to houston , in my view , to explore the possibility of coming\nbac
 'dear ut team :\ni wanted to let each of you know that the following ( 8 : 13 ) summer 2000
 'executive summary :\nsb - 7 x gives dept . of water and resources given legislative author
```

```
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, train_df['Message'], train_df['Spam/
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

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

```
Количество сформированных признаков - 54038
```

```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(vocabulary =
classifiers_list = [LogisticRegression(), MultinomialNB()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Convergence
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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Convergence
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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceW
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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
Векторизация - CountVectorizer(analyzer='word', binary=False, decode_error='s
               dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
               lowercase=True, max_df=1.0, max_features=None, min_df=1,
               ngram_range=(1, 1), preprocessor=None, stop_words=None,
               strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
               tokenizer=None,
               vocabulary={'00': 0, '000': 1, '0000': 2, '000000': 3,
                           '00...
                           '000000000005413': 12, '000000000005820': 13,
                           '000000000006238': 14, '000000000007399': 15,
                           '000000000007494': 16, '000000000007498': 17,
                           '000000000007588': 18, '000000000007589': 19,
                           '000000000007590': 20, '000000000007591': 21,
                           '000000000007592': 22, '000000000007593': 23,
                           '000000000007874': 24, '000000000007876': 25,
                           '000000000008254': 26, '000000000008348': 27,
                           '000000000008544': 28, '000000000008582': 29, ...})
Модель для классификации - LogisticRegression(C=1.0, class_weight=N
                  intercept_scaling=1, l1_ratio=None, max_iter=100,
                  multi_class='auto', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False)
Accuracy = 0.9993999399939995
==========================
Векторизация - CountVectorizer(analyzer='word', binary=False, decode_error='s
               dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
               lowercase=True, max_df=1.0, max_features=None, min_df=1,
               ngram_range=(1, 1), preprocessor=None, stop_words=None,
               strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
               tokenizer=None,
               vocabulary={'00': 0, '000': 1, '0000': 2, '000000': 3,
                           '00...
                           '000000000005413': 12, '000000000005820': 13,
                           '000000000006238': 14, '000000000007399': 15,
```

Лучшую точность показал CountVectorizer и LogisticRegression (99,93%)

```
X_train=train_df['Message']
y_train=train_df['Spam/Ham']
X_test=test_df['Message']
y_test=test_df['Spam/Ham']


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(),  LogisticRegression(C=5.0))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWar
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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
Метка       Accuracy
0       0.9994134897360704
1       1.0
```

Способ 2. На основе моделей word2vec

```
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 /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

Подготовим корпус

```
corpus  =  []
stop_words  =  stopwords.words('english')
tok  =  WordPunctTokenizer()
for  line  in  dataset['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)
```

```
corpus[:5]
```

```
[['gentleman',
  'kevin',
  'presto',
  'concurred',
  'purchase',
  'site',
  'license',
  'recommended',
  'vince',
  'thoughts',
  'others',
  'available',
  'demo',
  'package',
  'others',
  'would',
  'like',
  'see',
  'thanks',
  'lance',
  'forwarded',
  'lance',
  'cunningham',
  'na',
  'enron',
  'pm',
  'vince',
  'j',
  'kaminski',
  'ect',
  'vince',
  'j',
  'kaminski',
  'hou',
  'ect',
  'ect',
  'richard',
  'lewis',
  'lon',
  'ect',
  'ect',
  'tim',
  'belden',
  'hou',
  'ect',
  'ect',
  'tim',
  'heizenrader',
  'enron',
  'com',
  'kevin',
  'presto',
  'hou',
  'ect',
  'ect',
  'george',
  'hopley',
  'hou',
```

```
        'ect',
        'eat'
```

Количество текстов в корпусе не изменилось и соответствует целевому признаку

```
assert  dataset.shape[0]==len(corpus)
```

```
import  gensim
from  gensim.models  import  word2vec
%time  model  =  word2vec.Word2Vec(corpus,  workers=4,  min_count=10,  window=10,  sample=1e-3)
```

```
CPU times: user 1min 18s, sys: 442 ms, total: 1min 18s
Wall time: 42.3 s
```

Проверим, что модель обучилась

```
print(model.wv.most_similar(positive=['find'],  topn=5))
```

```
[('contacts', 0.4571227431297302), ('complete', 0.4518755376338959), ('internally', 0.446609
```

```
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)
```

```
class  EmbeddingVectorizer(object):
        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])
```

```
boundary  =  26664
X_train  =  corpus[:boundary]
X_test  =  corpus[boundary:]
y_train  =  dataset['Spam/Ham'][:boundary]
y_test  =  dataset['Spam/Ham'][boundary:]
```

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

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWar
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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Метка	Accuracy
0	0.9982404692082112
1	1.0

Лучшую точность показал CountVectorizer и LogisticRegression