Рубежный контроль №2

ИУ5-23 Зорин Арсений

Решение задачи классификации текстов.

Вариант №3: LinearSVC и Multinomial Naive Bayes (MNB)

In [15]:
```
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 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_er
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
import seaborn as sns
import tensorflow as tf
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
from gensim.models import word2vec
from nltk.corpus import stopwords
import re
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/a.zorin/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[15]:
```
True
```

In [2]:
```
categories = ['comp.graphics', 'misc.forsale', 'talk.politics.misc', 'rec.sport.hockey']
groups = fetch_20newsgroups(subset='train', categories=categories)
data = groups['data']
```

In [3]:
```
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 [4]:
```
vectorized = CountVectorizer()
vectorized.fit(data)
vocabulary = vectorized.vocabulary_
print('Количество сформированных признаков - {}'.format(len(vocabulary)))
```

```
Количество сформированных признаков - 34701
```

In [5]:
```
for i in list(vocabulary)[1:10]:
    print('{}={}'.format(i, vocabulary[i]))
```

```
dwarf=12688
bcarh601=6807
bnr=7381
ca=8258
jim=18501
jordan=18615
subject=30225
re=26291
truly=31992
```

In [6]:
```
test_features = vectorized.transform(data)
test_features
```

Out[6]:
```
<2234x34701 sparse matrix of type '<class 'numpy.int64'>'
        with 317800 stored elements in Compressed Sparse Row format>
```

In [7]:
```
# Размер нулевой строки
len(test_features.todense()[0].getA1())
```

Out[7]:
```
34701
```

In [13]:
```
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, groups['data'], groups['target'], scoring='accuracy', cv=3).mean
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

In [16]:
```
vectorizers_list = [CountVectorizer(vocabulary = vocabulary), TfidfVectorizer(vocabulary = vocabulary)]
classifiers_list = [LinearSVC(), MultinomialNB()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '000005102000': 2, '000007': 3,
                            '000100255pixel': 4, '000256': 5, '0004': 6,
                            '0007': 7, '000k': 8, '000usd': 9, '001': 10,
                            '0010': 11, '0010580b': 12, '001116': 13,
                            '001200201pixel': 14, '001323': 15, '001338': 16,
                            '00196': 17, '002': 18, '002302': 19, '002339': 20,
                            '0028': 21, '00309': 22, '003221': 23, '0038': 24,
                            '003848': 25, '0039': 26, '004253agrgb': 27,
                            '004325': 28, '004808': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.9480779870582858
============================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '000005102000': 2, '000007': 3,
                            '000100255pixel': 4, '000256': 5, '0004': 6,
                            '0007': 7, '000k': 8, '000usd': 9, '001': 10,
                            '0010': 11, '0010580b': 12, '001116': 13,
                            '001200201pixel': 14, '001323': 15, '001338': 16,
                            '00196': 17, '002': 18, '002302': 19, '002339': 20,
                            '0028': 21, '00309': 22, '003221': 23, '0038': 24,
                            '003848': 25, '0039': 26, '004253agrgb': 27,
                            '004325': 28, '004808': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.9552362223665537
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '000005102000': 2, '000007': 3,
                            '000100255pixel': 4, '000256': 5, '0004': 6,
                            '0007': 7, '000k': 8, '000usd': 9, '001': 10,
                            '0010': 11, '0010580b': 12, '001116': 13,
                            '001200201pixel': 14, '001323': 15, '001338': 16,
                            '00196': 17, '002': 18, '002302': 19, '002339': 20,
                            '0028': 21, '00309': 22, '003221': 23, '0038': 24,
                            '003848': 25, '0039': 26, '004253agrgb': 27,
                            '004325': 28, '004808': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.9677726059031536
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '000005102000': 2, '000007': 3,
                            '000100255pixel': 4, '000256': 5, '0004': 6,
                            '0007': 7, '000k': 8, '000usd': 9, '001': 10,
                            '0010': 11, '0010580b': 12, '001116': 13,
                            '001200201pixel': 14, '001323': 15, '001338': 16,
                            '00196': 17, '002': 18, '002302': 19, '002339': 20,
                            '0028': 21, '00309': 22, '003221': 23, '0038': 24,
                            '003848': 25, '0039': 26, '004253agrgb': 27,
                            '004325': 28, '004808': 29, ...})
Модель для классификации - MultinomialNB()
Accuracy = 0.9431478675037887
============================
```

Результаты:

- LinearSVC - 0.967
- Multinomial Naive Bayes (MNB) - 0.943

Лучшая точность у LinearSVC.