Журавлев Н.В., ИУ5-24М, Вариант 5

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета (кроме примера, который рассматривался в лекции). Классификация может быть бинарной или многоклассовой. Целевой признак из выбранного Вами датасета может иметь любой физический смысл, примером является задача анализа тональности текста.

Необходимо сформировать два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer.

В качестве классификаторов необходимо использовать два классификатор - GradientBoostingClassifier LogisticRegression Для каждого метода необходимо оценить качество классификации. Сделайте вывод о том, какой вариант векторизации признаков в паре с каким классификатором показал лучшее качество.

Был выбран датасет анализа эмоции по тексту

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from IPython.display import Image
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
from sklearn.ensemble import GradientBoostingClassifier
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
imdb_df = pd.read_csv("emotion_sentimen_dataset.csv", delimiter=',')
imdb_df = imdb_df.drop("n", axis=1)
imdb_df = imdb_df.sample(frac=0.05)
imdb_df
```


```
{"summary":"{\n  \"name\": \"imdb_df\",\n  \"rows\": 41978,\n
\"fields\": [\n    {\n      \"column\": \"text\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 40635,\n        \"samples\": [\n          \"i
could really embarrass him but i m feeling gracious tonight and he
could probably embarrass me even worse\",\n          \"i feel very
```

```
passionate about and hope to continue doing for many years\",\n
\"i feel like ive become so boring category a href http world music\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Emotion\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 13,\n
\"samples\": [\n          \"relief\",\n          \"surprise\",\n
\"love\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"imdb_df"}

# Сформируем общий словарь для обучения моделей из обучающей и
тестовой выборки
vocab_list = imdb_df['text'].tolist()

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

ncv = CountVectorizer(ngram_range=(1,3))
ngram_features = ncv.fit_transform(vocab_list)

tfidfv = TfidfVectorizer(ngram_range=(1,3))
tfidf_ngram_features = tfidfv.fit_transform(vocab_list)

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, imdb_df['text'],
imdb_df['Emotion'], 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),
TfidfVectorizer(vocabulary = corpusVocab)]
classifiers_list = [LogisticRegression(C=3.0, solver='lbfgs',
max_iter=100), GradientBoostingClassifier()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Количество сформированных признаков - 24668
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/
_split.py:700: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=3.
```

```
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: 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(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: 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(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: 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={'aa': 0, 'aaa': 1,
'aardvark': 2,
                            'aardvarkartglass': 3, 'aaron': 4,
'aarons': 5,
                            'aatp': 6, 'ab': 7, 'aback': 8, 'abandon':
9,
                            'abandoned': 10, 'abandoning': 11,
                            'abandonment': 12, 'abba': 13, 'abbey':
14,
                            'abbott': 15, 'abbreviations': 16, 'abby':
17,
                            'abc': 18, 'abdicate': 19, 'abdomen': 20,
```

```
                                'abducted': 21, 'abe': 22, 'abed': 23,
'abedin': 24,
                                'abelard': 25, 'abeyance': 26,
                                'abfcbrizlzconhq': 27, 'abhor': 28,
'abhyasa': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.9810615134951831
============================

/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/
_split.py:700: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=3.
  warnings.warn(

Векторизация - CountVectorizer(vocabulary={'aa': 0, 'aaa': 1,
'aardvark': 2,
                                'aardvarkartglass': 3, 'aaron': 4,
'aarons': 5,
                                'aatp': 6, 'ab': 7, 'aback': 8, 'abandon':
9,
                                'abandoned': 10, 'abandoning': 11,
                                'abandonment': 12, 'abba': 13, 'abbey':
14,
                                'abbott': 15, 'abbreviations': 16, 'abby':
17,
                                'abc': 18, 'abdicate': 19, 'abdomen': 20,
                                'abducted': 21, 'abe': 22, 'abed': 23,
'abedin': 24,
                                'abelard': 25, 'abeyance': 26,
                                'abfcbrizlzconhq': 27, 'abhor': 28,
'abhyasa': 29, ...})
Модель для классификации - GradientBoostingClassifier()
Accuracy = 0.9918528770088985
============================

/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/
_split.py:700: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=3.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: 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(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: 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(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: 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 - TfidfVectorizer(vocabulary={'aa': 0, 'aaa': 1,
'aardvark': 2,
                               'aardvarkartglass': 3, 'aaron': 4,
'aarons': 5,
                               'aatp': 6, 'ab': 7, 'aback': 8, 'abandon':
9,
                               'abandoned': 10, 'abandoning': 11,
                               'abandonment': 12, 'abba': 13, 'abbey':
14,
                               'abbott': 15, 'abbreviations': 16, 'abby':
17,
                               'abc': 18, 'abdicate': 19, 'abdomen': 20,
                               'abducted': 21, 'abe': 22, 'abed': 23,
'abedin': 24,
                               'abelard': 25, 'abeyance': 26,
                               'abfcbrizlzconhq': 27, 'abhor': 28,
'abhyasa': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.9490923771259694
============================

/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/
_split.py:700: UserWarning: The least populated class in y has only 1
```

```
members, which is less than n_splits=3.
  warnings.warn(

Векторизация - TfidfVectorizer(vocabulary={'aa': 0, 'aaa': 1,
'aardvark': 2,
                               'aardvarkartglass': 3, 'aaron': 4,
'aarons': 5,
                               'aatp': 6, 'ab': 7, 'aback': 8, 'abandon':
9,
                               'abandoned': 10, 'abandoning': 11,
                               'abandonment': 12, 'abba': 13, 'abbey':
14,
                               'abbott': 15, 'abbreviations': 16, 'abby':
17,
                               'abc': 18, 'abdicate': 19, 'abdomen': 20,
                               'abducted': 21, 'abe': 22, 'abed': 23,
'abedin': 24,
                               'abelard': 25, 'abeyance': 26,
                               'abfcbrizlzconhq': 27, 'abhor': 28,
'abhyasa': 29, ...})
Модель для классификации - GradientBoostingClassifier()
Accuracy = 0.9897088917188589
============================
```

Из полученных результатов видно, что лучший GradientBoostingClassifier и CountVectoriz. Это может быть из-за того, что выделять важные слова не сильно влияет в выбранном датасете. А GradientBoostingClassifier лучше LogisticRegression, т.к. может создавать более сложные модели, способные улавливать более сложные зависимости в данных, в то время как LogisticRegression ограничена линейной гипотезой, так же GradientBoostingClassifier может лучше адаптироваться к различным типам данных и шумам, в то время как LogisticRegression может быть менее гибкой в этом отношении.