Рубежный контроль №2

Выполнил: Ма Линь ¶

Студент группы:ИУ5И-21М

Тема: Методы обработки текстов.

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 классификатора по варианту для Вашей группы:

Мой вариант: LogisticRegression & Multinomial Naive Bayes - MNB

Список Библитека:

In [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.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
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")
```

In [2]:

```
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
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('М е т к а  \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

Загрузка данных

scotch_review.csv

In [3]:

```
ML_rev = pd.read_csv("scotch_review.csv")
ML_rev.head()
#WC_df = pd.read_csv("WClothing_Reviews.csv", delimiter='\t', header=None, names=['text', 'value'])
#WC_df.head()
```

Out[3]:

	Unnamed: 0	name	category	review.point	price	currency	description
0	1	Johnnie Walker Blue Label, 40%	Blended Scotch Whisky	97	225	$	Magnificently powerful and intense. Caramels, ...
1	2	Black Bowmore, 1964 vintage, 42 year old, 40.5%	Single Malt Scotch	97	4500.00	$	What impresses me most is how this whisky evol...
2	3	Bowmore 46 year old (distilled 1964), 42.9%	Single Malt Scotch	97	13500.00	$	There have been some legendary Bowmores from t...
3	4	Compass Box The General, 53.4%	Blended Malt Scotch Whisky	96	325	$	With a name inspired by a 1926 Buster Keaton m...
4	5	Chivas Regal Ultis, 40%	Blended Malt Scotch Whisky	96	160	$	Captivating, enticing, and wonderfully charmin...

In [4]:

```
ML_rev.shape
```

Out[4]:

```
(2247, 7)
```

Держать колонки "Review Text" и "Recommended IND".

In [5]:

```
ML_df = pd.DataFrame(ML_rev,columns=['description','review.point'])
ML_df.columns = ['text','value']
ML_df.head()
```

Out[5]:

	text	value
0	Magnificently powerful and intense. Caramels, ...	97
1	What impresses me most is how this whisky evol...	97
2	There have been some legendary Bowmores from t...	97
3	With a name inspired by a 1926 Buster Keaton m...	96
4	Captivating, enticing, and wonderfully charmin...	96

In [6]:

```
# Сформируем общий словарь для обучения моделей из обу
ML_df['text'].astype('U')
vocab_list = ML_df['text'].tolist()
vocab_list[1:10]
```

Out[6]:

```
["What impresses me most is how this whisky evolves; it's incredibly complex. On the
nose and palate, this is a thick, viscous, whisky with notes of sticky toffee, earth
y oak, fig cake, roasted nuts, fallen fruit, pancake batter, black cherry, ripe peac
h, dark chocolate-covered espresso bean, polished leather, tobacco, a hint of wild g
ame, and lingering, leafy damp kiln smoke. Flavors continue on the palate long after
swallowing. This is what we all hope for (and dream of) in an older whisky!",
 "There have been some legendary Bowmores from the mid-60s and this is every bit the
ir equal. All of them share a remarkable aroma of tropical fruit, which here moves i
nto hallucinatory intensity: guava, mango, peach, pineapple, grapefruit. There's a
very light touch of peat smoke, more a memory of Islay than the reality. Concentrate
d; even at low strength the palate is silky, heady, and haunting, and lasts forever
in the dry glass. A legend is born. (Eight bottles only for the U.S.) Editor's Choic
e.",
 "With a name inspired by a 1926 Buster Keaton movie, only 1,698 bottles produced, a
nd the news that one of the two batches is more than 30 years old, the clues were th
ere that this blend was never going to be cheap. It isn't, but it's superb, rich in
flavor that screams dusty old oak office, fresh polish, and Sunday church, with spic
es, oak dried fruits, squiggly raisins, and a surprising melting fruit-and-nut dairy
chocolate back story.",
 "Captivating, enticing, and wonderfully charming, this first blended malt from Chiv
as Regal contains selections of five Speyside malts: Strathisla, Longmorn, Tormore,
Allt-a-Bhainne, and Braeval. Red apple, cherry, raspberry fudge, peach and mango fru
it salad, dusting of cinnamon, and dry heather sprigs. In essence, it's rich and sa
tisfying, with dark vanilla, apricot, Bourneville-covered Brazil nuts, and tangerin
e, smoothed over by caramel and wood spices, maltiness, and gingersnap biscuits. Qui
te heavenly. Editor's Choice",
 'Powerful, muscular, well-textured, and invigorating. Even within the realm of Ardb
eg, this one stands out. The more aggressive notes of coal tar, damp kiln, anise, an
d smoked seaweed are supported by an array of fruit (black raspberry, black cherry,
plum), dark chocolate, espresso, molasses, bacon fat, kalamata olive, and warming ci
nnamon on the finish. Quite stunning!',
 'Deep gold color. Surprisingly lively on the nose for its age. A complex array of f
ruit (tangerine, sultana, pink grapefruit, papaya, and the general overall citrus DN
A that you'll find in old Bowmores), with balancing notes of honey and vanilla. A h
int of damp smoke and coconut. Just like with Black Bowmore, this is a texturally so
othing whisky on the palate, which continues to evolve in waves -- first the sweet h
oney, coating vanilla, and lively fruit, then turning quite visceral, with juicy oa
k, damp earth, deep peat smoke, and charcoal, followed by another wave of fruit (thi
s time, dried fruit), finishing off with subtle charred oak and roasted nuts. This w
hisky is better than White Bowmore, and it falls just short of Black Bowmore (which
I rated 97), because it's just a bit softer and less vibrant on the palate.',
 "Definitely showing its age, but not in a bad way — the distillery character is st
ill there. Solid foundation of thick, chewy toffee, old pot still rum, and fig cake.
Fruity too, with notes of golden raisin and nectarine. Soft, seductive peat smoke, j
uicy oak, cinnamon, and brine round out the palate. Excellent balance! One of the fi
nest Bowmore whiskies I've ever tasted (and, at this price, will probably never tas
te again.) (Editor's Pick) ",
 'The Dalmore is one of a handful of whiskies that seem to be able to age in the cas
k for many decades and still improve. This one is incredibly viscous on the nose and
palate (and very heavy on the tongue), with chewy toffee and old pot still rum. The
classic Dalmore marmalade note shines throughout, along with vanilla cream, an array
```

```
of dried spices (especially cinnamon and evergreen), juicy oak, forest bedding, ranc
io, old armagnac, polished leather, tobacco, maple syrup, dark chocolate, almond mac
aroon, and subtle espresso. Long, mouth-coating finish. The flavors evolve like wave
s lapping on the palate -- especially the interplay with the oak. I can't drink thi
s whisky slowly enough. A rare experience for the lucky few who can afford it. (Pric
e is per 100ml).',
 'A rich amber color and elegantly oxidized notes greet you. There are luscious old
fruits—pineapple, dried peach, apricot—and puffs of coal-like smokiness. In time,
sweet spices (cumin especially) emerge. Superbly balanced. The palate, while fragil
e, still has real sweetness alongside a lick of treacle. It can take a drop of wate
r, allowing richer, darker fruits to emerge. The finish is powerful, long, and reson
ant. Superb, not over-wooded, and a fair price for such a rarity.\xa0£1,995']
```

In [7]:

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

```
Количество сформированных признаков - 9086
```

In [8]:

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

```
powerful=6255
and=747
intense=4413
caramels=1621
dried=2783
peats=5974
elegant=2936
cigar=1836
smoke=7474
```

Векторизация текста на основе модели "мешка слов"

Векторизация текста поддерживается библиотекой scikit-learn.

Использование класса CountVectorizer

Подсчитывает количество слов словаря, входящих в данный текст.

In [9]:

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

In [10]:

```
test_features
```

Out[10]:

```
<2247x9086 sparse matrix of type '<class 'numpy.int64'>'
        with 127980 stored elements in Compressed Sparse Row format>
```

In [11]:

```
test_features.todense()
```

Out[11]:

```
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 [12]:

```
# Размер нулевой строки 无效行的尺寸
len(test_features.todense()[0].getA1())
```

Out[12]:

```
9086
```

In [13]:

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

Out[13]:

```
[1,
 1,
 4,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 2,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 3,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 2,
 1,
 3,
 1,
 1,
 1,
 1,
 1,
 1,
```

```
 1,
 1]
```

In [14]:

```
vocabVect.get_feature_names()[100:120]
```

Out[14]:

```
['1882',
 '1890',
 '1898',
 '18th',
 '19',
 '190',
 '1905',
 '191',
 '1911',
 '1913',
 '192',
 '1920s',
 '1926',
 '1946',
 '195',
 '1952',
 '1957',
 '1958',
 '1959',
 '196']
```

Использование класса TfidfVectorizer

Вычисляет специфичность текста в корпусе текстов на основе метрики TF-IDF.

In [15]:

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

Out[15]:

```
<2247x198463 sparse matrix of type '<class 'numpy.float64'>'
        with 434149 stored elements in Compressed Sparse Row format>
```

In [16]:

```
tfidf_ngram_features.todense()
```

Out[16]:

```
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.]])
```

In [17]:

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

Out[17]:

```
198463
```

In [18]:

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

Out[18]:

```
[0.08350257433898073,
 0.08350257433898073,
 0.08350257433898073,
 0.0255003110670377,
 0.08350257433898073,
 0.08350257433898073,
 0.04175285254572138,
 0.07628985092230442,
 0.07628985092230442,
 0.07046662317041068,
 0.08350257433898073,
 0.08350257433898073,
 0.08350257433898073,
 0.07207067819547541,
 0.07207067819547541,
 0.046069805214705065,
 0.08350257433898073,
 0.08350257433898073,
```

Решение задачи анализа тональности текста на основе модели "мешка слов"

С использованием кросс-валидации попробуем применить к корпусу текстов различные варианты векторизации и классификации.

In [19]:

```
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, ML_df['text'], ML_df['value'], scoring='accuracy', cv
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

По варианту используем классификаторы "LogisticRegression" и "Multinomial Naive Bayes - MNB"

In [20]:

```
from sklearn.model_selection import ShuffleSplit
splits = ShuffleSplit(n_splits=5, test_size=0.2, random_state=42)
```

In [21]:

```
from sklearn.naive_bayes import MultinomialNB
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(vocabulary = corpusVo
classifiers_list = [LogisticRegression(C=3.0), MultinomialNB()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
E:\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:666: UserWarning:
The least populated class in y has only 1 members, which is less than n_splits=3.
  warnings.warn(("The least populated class in y has only %d"
E:\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceW
arning: 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 (https://scikit-lea
rn.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
 (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
E:\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceW
arning: 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:
```

Разделим выборку на обучающую и тестовую и проверим решение для лучшей модели

In [22]:

```
X_train, X_test, y_train, y_test = train_test_split(ML_df['text'], ML_df['value'], test_size=0.5, ra
```

In [23]:

```
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 [24]:

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

```
Метка       Accuracy
70          0.0
72          0.0
73          0.0
74          0.0
76          0.0
77          0.0
78          0.0
79          0.0
80          0.05
81          0.0
82          0.0
83          0.014285714285714285
84          0.15853658536585366
85          0.1782178217821782
86          0.12396694214876033
87          0.18181818181818182
88          0.20833333333333334
89          0.05660377358490566
90          0.0989010989010989
91          0.018867924528301886
92          0.0
93          0.0
94          0.0
95          0.0
96          0.0
97          0.0

E:\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWar
ning: 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 (https://scikit-lear
n.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 (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [25]:

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

```
Метка       Accuracy
70          0.0
72          0.0
73          0.0
74          0.0
76          0.0
77          0.0
78          0.0
79          0.058823529411764705
80          0.05
81          0.027777777777777776
82          0.06382978723404255
83          0.08571428571428572
84          0.0975609756097561
85          0.1485148514851485
86          0.14049586776859505
87          0.13636363636363635
88          0.14583333333333334
89          0.09433962264150944
90          0.12087912087912088
91          0.05660377358490566
92          0.018867924528301886
93          0.04878048780487805
94          0.0
95          0.0
96          0.0
97          0.0

E:\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWar
ning: 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 (https://scikit-lear
n.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 (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [26]:

```
sentiment(TfidfVectorizer(), MultinomialNB(alpha=0.5))
```

Метка	Accuracy
70	0.0
72	0.0
73	0.0
74	0.0
76	0.0
77	0.0
78	0.0
79	0.0
80	0.0
81	0.0
82	0.0
83	0.0
84	0.012195121951219513
85	0.12871287128712872
86	0.0743801652892562
87	0.4818181818181818
88	0.3541666666666667
89	0.009433962264150943
90	0.04395604395604396
91	0.0
92	0.0
93	0.0
94	0.0
95	0.0
96	0.0
97	0.0

In [27]:

```
sentiment(CountVectorizer(), MultinomialNB(alpha=0.5))
```

Метка	Accuracy
70	0.0
72	0.0
73	0.0
74	0.0
76	0.0
77	0.0
78	0.0
79	0.0
80	0.025
81	0.0
82	0.0
83	0.014285714285714285
84	0.12195121951219512
85	0.2277277227722773
86	0.15702479338842976
87	0.2
88	0.28125
89	0.05660377358490566
90	0.08791208791208792
91	0.018867924528301886
92	0.018867924528301886
93	0.0
94	0.0
95	0.0
96	0.0
97	0.0

In [ ]:

```

```