Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Отчет

Рубежный контроль № 2

Вариант 17

По курсу «Технологии машинного обучения»

ИСПОЛНИТЕЛЬ:
Молева Анастасия
Группа ИУ5-61Б
_____

"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.
_____

"__"_____2020 г.

Москва 2020

Задача 1. Классификация текстов на основе методов наивного Байеса.

ИУ5-61Б, Молева А.А.

Задача 1. Классификация текстов на основе методов наивного Байеса.

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 признаки на основе CountVectorizer или TfidfVectorizer.

В качестве классификаторов необходимо использовать два классификатора, не относящихся к наивным Байесовским методам (например, LogisticRegression, LinearSVC), а также Multinomial Naive Bayes (MNB), Complement Naive Bayes (CNB), Bernoulli Naive Bayes.

Для каждого метода необходимо оценить качество классификации с помощью хотя бы одной метрики качества классификации (например, Accuracy).

Сделате выводы о том, какой классификатор осуществляет более качественную классификацию на Вашем наборе данных.

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

Загрузка данных

In [2]:
```
data = pd.read_csv("datasets/sentiment labelled sentences/yelp_labelled.txt",
                   delimiter='\t', header=None, names=['Text', 'Value'])
data.head()
```

Out[2]:

Загрузка данных

In [2]:
```
data = pd.read_csv("datasets/sentiment labelled sentences/yelp_labelled.txt",
                   delimiter='\t', header=None, names=['Text', 'Value'])
data.head()
```

Out[2]:

	Text	Value
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1

In [3]:
```
data.shape
```

Out[3]:
```
(1000, 2)
```

Общий словарь для обучения моделей

In [4]:
```
vocab_list = data.Text.tolist()
vocab_list[:10]
```

Out[4]:
```
['Wow... Loved this place.',
 'Crust is not good.',
 'Not tasty and the texture was just nasty.',
 'Stopped by during the late May bank holiday off Rick Steve recommendation and loved it.',
 'The selection on the menu was great and so were the prices.',
 'Now I am getting angry and I want my damn pho.',
 "Honeslty it didn't taste THAT fresh.)",
 'The potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer.',
 'The fries were great too.',
 'A great touch.']
```

In [5]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

In [6]:
```
vocabVect = CountVectorizer()
vocabVect.fit_transform(vocab_list)
```

Out[6]:
```
<1000x2035 sparse matrix of type '<class 'numpy.int64'>'
        with 9782 stored elements in Compressed Sparse Row format>
```

Количество признаков = 2035

In [7]:
```
len(vocabVect.get_feature_names())
```

Out[7]:
```
2035
```

In [8]:
```
corpusVocab = vocabVect.vocabulary_
```

Признак и его индекс в словаре

In [9]:
```
for i in list(corpusVocab)[:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
wow=2012
loved=1046
this=1798
place=1330
crust=427
is=943
not=1195
good=764
tasty=1761
and=64
```

Векторизация текста

In [10]:
```
test_features = vocabVect.transform(vocab_list)
test_features
```

Out[10]:
```
<1000x2035 sparse matrix of type '<class 'numpy.int64'>'
        with 9782 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)
```

1000 строк - 1000 предложений в документе

2035 столбцов - 2035 уникальных значений в документе

N-граммы

In [12]:
```
ncv = CountVectorizer(ngram_range=(1, 3))
ngram_features = ncv.fit_transform(vocab_list)
ngram_features
```

Out[12]:
```
<1000x16553 sparse matrix of type '<class 'numpy.int64'>'
        with 27378 stored elements in Compressed Sparse Row format>
```

In [13]:
```
ncv.get_feature_names()[100:120]
```

Out[13]:
```
['about taste like',
 'about the',
 'about the place',
 'about this',
 'about this place',
 'about two',
 'about two bites',
 'about was',
 'about was their',
 'about working',
 'about working eating',
 'above',
 'above and',
 'above and beyond',
 'above average',
 'above average or',
 'absolute',
 'absolute must',
 'absolute must visit',
 'absolutely']
```

Векторизация TfidVectorizer

In [14]:
```
tfidfv = TfidfVectorizer(ngram_range=(1,3))
tfidf_ngram_features = tfidfv.fit_transform(vocab_list)
tfidf_ngram_features
```

Out[14]:
```
<1000x16553 sparse matrix of type '<class 'numpy.float64'>'
        with 27378 stored elements in Compressed Sparse Row format>
```

In [15]:
```
tfidf_ngram_features.todense()
```

Out[15]:
```
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 [16]:
```
# Непустые значения нулевой строки
[i for i in tfidf_ngram_features.todense()[0].getA1() if i>0][:10]
```

Out[16]:
```
[0.3110249863005478,
 0.407238517312785,
 0.407238517312785,
 0.1842360034601564,
 0.16951209244619436,
 0.20421138601952366,
 0.36811828714125194,
 0.407238517312785,
 0.407238517312785]
```

Решение задачи

In [17]:
```
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, data['Text'], data['Value'], scoring='accuracy', cv=3).mean()
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

In [18]:
```
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
```

In [19]:
```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(vocabulary = corpusVocab)]
classifiers_list = [LogisticRegression(C=3.0), LinearSVC(), KNeighborsClassifier()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '10': 1, '100': 2, '11': 3, '12': 4,
                               '15': 5, '17': 6, '1979': 7, '20': 8, '2007': 9,
                               '23': 10, '30': 11, '30s': 12, '35': 13, '40': 14,
                               '40min': 15, '45': 16, '4ths': 17, '51b': 18,
                               '70': 19, '85': 20, '90': 21, '99': 22, 'about': 23,
                               'above': 24, 'absolute': 25, 'absolutely': 26,
                               'absolutley': 27, 'accident': 28,
                               'accommodations': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.7770045494596394
============================
Векторизация - CountVectorizer(vocabulary={'00': 0, '10': 1, '100': 2, '11': 3, '12': 4,
                               '15': 5, '17': 6, '1979': 7, '20': 8, '2007': 9,
                               '23': 10, '30': 11, '30s': 12, '35': 13, '40': 14,
                               '40min': 15, '45': 16, '4ths': 17, '51b': 18,
                               '70': 19, '85': 20, '90': 21, '99': 22, 'about': 23,
                               'above': 24, 'absolute': 25, 'absolutely': 26,
                               'absolutley': 27, 'accident': 28,
                               'accommodations': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.7749965534396672
============================
Векторизация - CountVectorizer(vocabulary={'00': 0, '10': 1, '100': 2, '11': 3, '12': 4,
                               '15': 5, '17': 6, '1979': 7, '20': 8, '2007': 9,
                               '23': 10, '30': 11, '30s': 12, '35': 13, '40': 14,
                               '40min': 15, '45': 16, '4ths': 17, '51b': 18,
                               '70': 19, '85': 20, '90': 21, '99': 22, 'about': 23,
                               'above': 24, 'absolute': 25, 'absolutely': 26,
                               'absolutley': 27, 'accident': 28,
                               'accommodations': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.6339962717208226
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '10': 1, '100': 2, '11': 3, '12': 4,
                               '15': 5, '17': 6, '1979': 7, '20': 8, '2007': 9,
                               '23': 10, '30': 11, '30s': 12, '35': 13, '40': 14,
                               '40min': 15, '45': 16, '4ths': 17, '51b': 18,
                               '70': 19, '85': 20, '90': 21, '99': 22, 'about': 23,
                               'above': 24, 'absolute': 25, 'absolutely': 26,
                               'absolutley': 27, 'accident': 28,
                               'accommodations': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.7989975844107581
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '10': 1, '100': 2, '11': 3, '12': 4,
                               '15': 5, '17': 6, '1979': 7, '20': 8, '2007': 9,
                               '23': 10, '30': 11, '30s': 12, '35': 13, '40': 14,
                               '40min': 15, '45': 16, '4ths': 17, '51b': 18,
                               '70': 19, '85': 20, '90': 21, '99': 22, 'about': 23,
                               'above': 24, 'absolute': 25, 'absolutely': 26,
                               'absolutley': 27, 'accident': 28,
                               'accommodations': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.7910065754377132
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '10': 1, '100': 2, '11': 3, '12': 4,
                               '15': 5, '17': 6, '1979': 7, '20': 8, '2007': 9,
                               '23': 10, '30': 11, '30s': 12, '35': 13, '40': 14,
                               '40min': 15, '45': 16, '4ths': 17, '51b': 18,
                               '70': 19, '85': 20, '90': 21, '99': 22, 'about': 23,
                               'above': 24, 'absolute': 25, 'absolutely': 26,
                               'absolutley': 27, 'accident': 28,
                               'accommodations': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.7529655403906981
============================
```

Разделение выборки

In [20]:
```
from sklearn.model_selection import train_test_split
```

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(data['Text'], data['Value'], test_size=0.5, random_state=42)
```

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]:
```
types = [[TfidfVectorizer(), LogisticRegression(C=5.0)],
        [TfidfVectorizer(ngram_range=(1,3)), LogisticRegression(C=5.0)],
        [TfidfVectorizer(ngram_range=(2,3)), LogisticRegression(C=5.0)],
        [TfidfVectorizer(ngram_range=(1,4)), LogisticRegression(C=5.0)],
        [TfidfVectorizer(ngram_range=(2,4)), LogisticRegression(C=5.0)]]
for type_ in types:
    sentiment(*type_)
    print("============================")
```

```
Метка    Accuracy
0        0.80078125
1        0.8114754098360656
============================
Метка    Accuracy
0        0.77734375
1        0.8401639344262295
============================
Метка    Accuracy
0        0.49609375
1        0.8278688524590164
============================
Метка    Accuracy
0        0.76171875
1        0.8401639344262295
============================
Метка    Accuracy
0        0.48828125
1        0.8319672131147541
============================
```