РК №2 по ММО Цапий Вадим, ИУ5-23М

Вариант №1. Классификация текстов на основе методов наивного Байеса.

Задание:

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета. 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 один из классификаторов, не относящихся к наивным Байесовским методам (например, LogisticRegression), 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]:

```
from typing import Dict, Tuple
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.naive_bayes import MultinomialNB, ComplementNB, BernoulliNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
import numpy as np
import string
from sklearn.model_selection import train_test_split
import warnings
import pandas as pd
%matplotlib inline
```

In [4]:

```
data = pd.read_csv('C:/Users/VTsapiy/Desktop/data/Food_Reviews.csv')
```

In [5]:

```
data.head()
```

Out[5]:

	Text	Score
0	I have bought several of the Vitality canned d...	3
1	Product arrived labeled as Jumbo Salted Peanut...	1
2	This is a confection that has been around a fe...	3
3	If you are looking for the secret ingredient i...	1
4	Great taffy at a great price. There was a wid...	3

In [7]:

```
data.dtypes
```

Out[7]:

```
Text     object
Score     int64
dtype: object
```

In [9]:

```
data.isnull().sum()
```

Out[9]:

```
Text     0
Score    0
dtype: int64
```

Делим датасет на тестовую и обучающую выборки

In [11]:

```
X_train, X_test, y_train, y_test = train_test_split(
    data['Text'],
    data['Score'],
     test_size=0.4,
    random_state = 1
)

print("Training dataset: ", X_train.shape[0])
print("Test dataset: ", X_test.shape[0])
```

```
Training dataset:  5999
Test dataset:  4000
```

In [12]:

```
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 [13]:

```
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 [14]:

```
classificators = [LogisticRegression(C=5.0), MultinomialNB(), ComplementNB(), BernoulliNB()
vectorizers = [TfidfVectorizer(), CountVectorizer()]
```

In [15]:

```
warnings.filterwarnings('ignore')

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

```
Метка    Accuracy
1        0.50177304964539
2        0.14893617021276595
3        0.9707112970711297
```

In [16]:

```
sentiment(CountVectorizer(), MultinomialNB())
```

```
Метка    Accuracy
1        0.34397163120567376
2        0.11246200607902736
3        0.9639523656260058
```

In [17]:

```
sentiment(TfidfVectorizer(), MultinomialNB())
```

```
Метка    Accuracy
1        0.0
2        0.0
3        1.0
```

In [18]:

```
sentiment(CountVectorizer(), ComplementNB())
```

```
Метка    Accuracy
1        0.5177304964539007
2        0.182370820668693
3        0.9327325394271001
```

In [19]:

```
sentiment(TfidfVectorizer(), ComplementNB())
```

```
Метка    Accuracy
1        0.028368794326241134
2        0.0
3        0.9983907306083039
```

In [20]:

```
sentiment(CountVectorizer(binary=True), BernoulliNB())
```

```
Метка    Accuracy
1        0.21453900709219859
2        0.1702127659574468
3        0.9266173157386547
```

In [21]:

```
sentiment(TfidfVectorizer(binary=True), BernoulliNB())
```

```
Метка    Accuracy
1        0.21453900709219859
2        0.1702127659574468
3        0.9266173157386547
```

В нашемм случае наиболее высокие результаты были для метки 3(>90%), для остальных меток результаты очень маленькие. Все результаты примерно одинаковые по результатам, нельзя выделить какой-то один метод классификации

In [ ]:

```

```