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

Рубежный контроль №2

по дисциплине

«Методы машинного обучения»

Выполнил:
студент группы ИУ5-21М
Хтет Мин Паинг Вин

Москва — 2020 г.

Хтет Мин Паинг Вин, ИУ5И-21М

Indented block

Задача № 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 [0]:
```
import pandas as pd

df = pd.read_csv("/content/Reviews.csv", sep = ",")
```

In [2]:
```
df.head(1)
```

Out[2]:

Userld	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...

In [3]:
```
del df['ProductId']
del df['UserId']
del df['HelpfulnessNumerator']
del df['HelpfulnessDenominator']
del df['Time']
del df['ProfileName']
del df['Id']
df.head(3)
```

Out[3]:

	Score	Summary	Text
0	5	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	1	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	4	"Delight" says it all	This is a confection that has been around a fe...

In [4]:
```
df.dtypes
```

Out[4]:
```
Score      int64
Summary    object
Text       object
dtype: object
```

In [5]:
```
#Проверка на пустые значения
df.isnull().sum()
```

Out[5]:
```
Score      0
Summary    27
Text       0
dtype: int64
```

In [0]:
```
df = df.dropna(axis=0, how='any')
```

In [7]:
```
df.shape
```

Out[7]:
```
(568427, 3)
```

In [0]:
```
# df3_ = df3.dropna(axis=0, how='any')
```

In [0]:
```
%matplotlib inline
```

Обработка данных

In [0]:
```
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
```

In [11]:
```
from sklearn.model_selection import train_test_split

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

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

```
Training dataset:  341056
Test dataset:  227371
```

In [0]:
```
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 [0]:
```
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]:
```
import warnings
warnings.filterwarnings('ignore')

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

```
Метка      Accuracy
1       0.6906712663504384
2       0.21756015196285353
3       0.335618444952247
4       0.26556055202356954
5       0.9465058516750032
```

In [15]:
```
sentiment(CountVectorizer(), MultinomialNB())
```

```
Метка      Accuracy
1       0.6422308466292942
2       0.21578725200506543
3       0.3129431065799496
4       0.37500387657001083
5       0.8706852067179018
```

In [16]:
```
sentiment(TfidfVectorizer(), MultinomialNB())
```

```
Метка      Accuracy
1       0.07843419098270327
2       0.0010130856901646263
3       0.00041014823929220133
4       0.007163901380058924
5       0.9991881273951962
```

In [17]:
```
sentiment(CountVectorizer(), ComplementNB())
```

Метка	Accuracy
1	0.7998179291840353
2	0.13718868720979316
3	0.2668893185679967
4	0.347123585051946
5	0.862614642604047

In [18]:
```
sentiment(TfidfVectorizer(), ComplementNB())
```

Метка	Accuracy
1	0.7649369939150017
2	0.12553820177289995
3	0.20929278725024902
4	0.23153977360831138
5	0.9176706136518442

In [19]:
```
sentiment(CountVectorizer(binary=True), BernoulliNB())
```

Метка	Accuracy
1	0.503521632887739
2	0.1400590966652596
3	0.26642057772309136
4	0.3097844627073965
5	0.8253373055461908

In [20]:
```
sentiment(TfidfVectorizer(binary=True), BernoulliNB())
```

Метка	Accuracy
1	0.503521632887739
2	0.1400590966652596
3	0.26642057772309136
4	0.3097844627073965
5	0.8253373055461908

Вывод:

Методы классификации текстов, основанные на "наивном" Байесе работают не хуже чем логистическая регрессия. Логистическая регрессия - точность достигает даже 95 процентов для метки 5,70%-для 1, для остальных случаев результаты не очень хорошие. Во всех методах для метки 5 были достигнуты хорошие результаты - выше 82 процентов. Логистическая регрессия работает более плавно. Все методы в чем-то показывают лучше результат, а в чем-то хуже. Закономерности не наблюдается.