

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 ИУ5И-21М
Гунькин Михаил

Москва — 2020 г.

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).

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

Датасет: <https://www.kaggle.com/snap/amazon-fine-food-reviews>

```
In [0]: import pandas as pd

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

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

```
Out[2]:
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	130386240

```
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    1  
Text       0  
dtype: int64
```

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

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

```
Out[7]: (36304, 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 [10]: 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: 21782  
Test dataset: 14522
```

```

In [0]: def accuracy_score_for_classes(
        y_true: np.ndarray,
        y_pred: np.ndarray) -> Dict[int, float]:
        """
        Вычисление метрики ассигасы для каждого класса
        y_true - истинные значения классов
        y_pred - предсказанные значения классов
        Возвращает словарь: ключ - метка класса,
        значение - Ассигасу 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_fit = df[df['t']==c]
            # расчет ассигасы для заданной метки класса
            temp_acc = accuracy_score(
                temp_data_fit['t'].values,
                temp_data_fit['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 ассигасы 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 [0]: classifiers = [LogisticRegression(C=5.0), MultinomialNB(), ComplementNB(), BernoulliNB()]
        vectorizers = [TfidfVectorizer(), CountVectorizer()]

```

```

In [14]: import warnings
         warnings.filterwarnings('ignore')

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

```

Метка	Accuracy
1	0.5958549222797928
2	0.23564593301435408
3	0.32096635030198445
4	0.2800959232613909
5	0.9129908700912991

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

Метка	Accuracy
1	0.5062916358253146
2	0.03708133971291866
3	0.14926660914581535
4	0.24988009592326138
5	0.9347706522934771

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

Метка	Accuracy
1	0.0014803849000740192
2	0.0
3	0.0
4	0.0
5	0.999780002199978

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

Метка	Accuracy
1	0.6787564766839378
2	0.1339712918660287
3	0.23382226056945643
4	0.26091127098321343
5	0.8838411615883841

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

Метка	Accuracy
1	0.3545521835677276
2	0.03708133971291866
3	0.04400345125107852
4	0.04364508393285372
5	0.982180178198218

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

Метка	Accuracy
1	0.2908956328645448
2	0.02631578947368421
3	0.14150129421915444
4	0.23932853717026378
5	0.8691013089869102

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

Метка	Accuracy
1	0.2908956328645448
2	0.02631578947368421
3	0.14150129421915444
4	0.23932853717026378
5	0.8691013089869102

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.