

# Рубежный контроль №2

Выполнила: Сукач Елизавета, группа ИУ5-23М

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

1. 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;
2. 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;
3. Для каждого метода необходимо оценить качество классификации с помощью хотя бы одной метрики качества классификации (например, Accuracy);
4. Сделайте выводы о том, какой классификатор осуществляет более качественную классификацию на Вашем наборе данных.

## Загрузка и предобработка данных

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
os.listdir()
data = pd.read_csv('/Users/elizavetasukach/Desktop/MachineLearning/rk2/hotel_
reviews.csv', sep=",")
```

In [4]:

```
reviews = data[['reviews.text', 'reviews.rating']]
reviews.head()
```

Out[4]:

	reviews.text	reviews.rating
0	Pleasant 10 min walk along the sea front to th...	4.0

	<b>reviews.text</b>	<b>reviews.rating</b>
1	Really lovely hotel. Stayed on the very top fl...	5.0
2	Ett mycket bra hotell. Det som drog ner betyge...	5.0
3	We stayed here for four nights in October. The...	5.0
4	We stayed here for four nights in October. The...	5.0

In [5]:

```
reviews_cleaned = reviews.dropna(axis=0, how='any')
float_rating = reviews_cleaned['reviews.rating']
(reviews.shape, reviews_cleaned.shape)
```

Out[5]:

```
((35912, 2), (35028, 2))
```

In [7]:

```
reviews_cleaned['reviews.rating'] = reviews_cleaned['reviews.rating'].astype(
int)
reviews_cleaned.head()
```

Out[7]:

	<b>reviews.text</b>	<b>reviews.rating</b>
0	Pleasant 10 min walk along the sea front to th...	4
1	Really lovely hotel. Stayed on the very top fl...	5

	reviews.text	reviews.rating
2	Ett mycket bra hotell. Det som drog ner betyge...	5
3	We stayed here for four nights in October. The...	5
4	We stayed here for four nights in October. The...	5

## Обучение на различных классификаторах

In [9]:

```
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
```

In [10]:

```
X_train, X_test, y_train, y_test = train_test_split(reviews_cleaned['reviews.
text'], reviews_cleaned['reviews.rating'], test_size=0.4, random_state=1)
```

In [11]:

```
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_dataflt = df[df['t']==c]
    # расчет accuracy для заданной метки класса
    temp_acc = accuracy_score(
        temp_dataflt['t'].values,
        temp_dataflt['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, v, c):
    """
    Вывод метрики accuracy для каждого класса
    """
    print("Признаки сформированы на\n{}".format(v))
    print("\nКлассификатор\n{}".format(c))
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        if i > 5:
            pass
        else:
            print('{} \t {:.2%}'.format(i, accs[i]))
    print('\n\n')

```

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, v, c)

```

In [14]:

```

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

```

In [15]:

```

for classifier in classifiers:
    for vectorizer in vectorizers:
        sentiment(vectorizer, classifier)
Признаки сформированы на
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',

```

```
dtype=<class 'numpy.float64'>, encoding='utf-8',
input='content', lowercase=True, max_df=1.0, max_features=None,
e,
min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None,
smooth_idf=True, stop_words=None, strip_accents=None,
sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
tokenizer=None, use_idf=True, vocabulary=None)
```

Классификатор

```
LogisticRegression(C=5.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='warn', n_jobs=None, penalty='l2',
random_state=None, solver='warn', tol=0.0001, verbose=0,
warm_start=False)
```

Метка	Accuracy
0	90.08%
1	55.61%
2	19.68%
3	31.93%
4	43.03%
5	74.08%

Признаки сформированы на

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
lowercase=True, max_df=1.0, max_features=None, min_df=1,
ngram_range=(1, 1), preprocessor=None, stop_words=None,
strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
tokenizer=None, vocabulary=None)
```

Классификатор

```
LogisticRegression(C=5.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='warn', n_jobs=None, penalty='l2',
random_state=None, solver='warn', tol=0.0001, verbose=0,
warm_start=False)
```

Метка	Accuracy
0	90.84%
1	49.20%
2	22.85%
3	31.84%
4	41.58%
5	70.66%

Признаки сформированы на

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
dtype=<class 'numpy.float64'>, encoding='utf-8',
input='content', lowercase=True, max_df=1.0, max_features=Non
e,
min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None,
smooth_idf=True, stop_words=None, strip_accents=None,
sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
tokenizer=None, use_idf=True, vocabulary=None)
```

Классификатор

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Метка	Accuracy
-------	----------

0	90.08%
1	9.54%
2	0.25%
3	3.28%
4	28.36%
5	91.26%

Признаки сформированы на

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
dtype=<class 'numpy.int64'>, encoding='utf-8', input='content
',
lowercase=True, max_df=1.0, max_features=None, min_df=1,
ngram_range=(1, 1), preprocessor=None, stop_words=None,
strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
tokenizer=None, vocabulary=None)
```

Классификатор

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Метка	Accuracy
-------	----------

0	90.08%
1	63.29%
2	10.18%
3	31.97%
4	40.77%
5	77.37%

Признаки сформированы на

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
dtype=<class 'numpy.float64'>, encoding='utf-8',
input='content', lowercase=True, max_df=1.0, max_features=Non
e,
min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None,
smooth_idf=True, stop_words=None, strip_accents=None,
sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
```

```
tokenizer=None, use_idf=True, vocabulary=None)
```

Классификатор

```
ComplementNB(alpha=1.0, class_prior=None, fit_prior=True, norm=False)
```

Метка	Accuracy
-------	----------

0	90.08%
1	69.11%
2	8.09%
3	24.60%
4	27.52%
5	83.99%

Признаки сформированы на

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content
',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

Классификатор

```
ComplementNB(alpha=1.0, class_prior=None, fit_prior=True, norm=False)
```

Метка	Accuracy
-------	----------

0	90.08%
1	79.32%
2	10.59%
3	25.36%
4	25.86%
5	81.09%

Признаки сформированы на

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features=Non
e,
                min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None,
                smooth_idf=True, stop_words=None, strip_accents=None,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, use_idf=True, vocabulary=None)
```

Классификатор

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Метка	Accuracy
-------	----------

0	0.00%
1	39.41%

2	7.76%
3	23.74%
4	38.68%
5	80.62%

Признаки сформированы на

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

Классификатор

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Метка	Accuracy
0	0.00%
1	39.41%
2	7.76%
3	23.74%
4	38.68%
5	80.62%

## Вывод

На основе полученного можно сделать вывод, что лучшим методом в данной ситуации является CountVectorizer с ComplementNB, где удалось правильно оценить отзывы с оценками 0, 1 и 5 в 80% случаев