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

Рубежный контроль №2
по дисциплине
«Методы машинного обучения»
на тему
«Методы построения моделей машинного обучения»

Выполнл:
студент группы ИУ5-23М
Богомолов Д.Н.

Москва — 2020 г.

РК №2 по дисциплине Методы машинного обучения

Богомолов Д.Н. ИУ5-23М

Вариант №1. Классификация текстов на основе методов наивного Байеса

Задание 1:

Данный вариант выполняется на основе материалов лекции часть 1 и часть 2.

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 двух метрик качества классификации (например, Accuracy, ROC-AUC).

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

Датасет: https://www.kaggle.com/grikomsn/amazon-cell-phones-reviews?select=20191226-reviews.csv

Будут использованы значения колонок body, где хранится текст, и rating. Рейтинг выставлен от 1 до 5, т.е. количество классов равно 5.

Загрузка и подготовка датасета

```
1  import pandas as pd
2
3  df = pd.read_csv("/content/drive/My Drive/MMO_Datasets/20191226-reviews.csv", sep = "
```

```
1  from google.colab import drive
2  drive.mount('/content/drive')
```

```
   Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
1  df.head(5)
```

	asin	name	rating	date	verified	
0	B0000SX2UC	Janet	3	October 11, 2005	False	Def not best, but n
1	B0000SX2UC	Luke Wyatt	1	January 7, 2004	False	Text Messaging Does
2	B0000SX2UC	Brooke	5	December 30, 2003	False	Love Thi
3	B0000SX2UC	amy m. teague	3	March 18, 2004	False	Love the Phone

```
del df['asin']
del df['name']
del df['date']
del df['verified']
del df['helpfulVotes']
df.head(3)
```

	rating	title	body
0	3	Def not best, but not worst	I had the Samsung A600 for awhile which is abs...
1	1	Text Messaging Doesn't Work	Due to a software issue between Nokia and Spri...
2	5	Love This Phone	This is a great, reliable phone. I also purcha...

```
df.dtypes
```

```
rating     int64
title     object
body      object
dtype: object
```

```
# Проверка на пустые значения
df.isnull().sum()
```

```
rating     0
title     14
body      21
dtype: int64
```

```
df = df.dropna(axis=0, how='any')
```

```
df.shape
```

```
(67956, 3)
```

Обработка данных

```
from typing import Dict, Tuple
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, balanced_accuracy_score, f1_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
```

```
X = df.body
y = df.rating
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Создание трёх моделей

```
from sklearn.pipeline import Pipeline

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Lasso
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.calibration import CalibratedClassifierCV
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier

tfidf = TfidfVectorizer(sublinear_tf=True, min_df=1, norm='l2',
                                                ngram_range=(1, 2),
                                                stop_words='english')


classif = Pipeline([('tfidf', tfidf),
('MnNB', OneVsRestClassifier(MultinomialNB())), 19      ])

classif2 = Pipeline([('tfidf', tfidf),
('lSVC', OneVsRestClassifier(LinearSVC())), 23    ])

classif3 = Pipeline([('tfidf', tfidf),
('LR', LogisticRegression(C=5.0)), 27      ])

result_df = pd.DataFrame()
```

```
classif.fit(X_train, y_train);
```

```
from sklearn.metrics import accuracy_score as AS, precision_score as PS, recall_score
result_df.loc['MultinomialNB', 'AS train'] = AS(y_train, classif.predict(X_train))
result_df.loc['MultinomialNB', 'PS train'] = PS(y_train, classif.predict(X_train), av
result_df.loc['MultinomialNB', 'RS train'] = RS(y_train, classif.predict(X_train), av
result_df.loc['MultinomialNB', 'AS test']  = AS(y_test , classif.predict(X_test))
result_df.loc['MultinomialNB', 'PS test']  = PS(y_test , classif.predict(X_test), ave
result_df.loc['MultinomialNB', 'RS test']  = RS(y_test , classif.predict(X_test), ave
```

```
classif2.fit(X_train, y_train);
```

```
result_df.loc['LinearSVC', 'AS train'] = AS(y_train, classif2.predict(X_train))
result_df.loc['LinearSVC', 'PS train'] = PS(y_train, classif2.predict(X_train), avera
result_df.loc['LinearSVC', 'RS train'] = RS(y_train, classif2.predict(X_train), avera
result_df.loc['LinearSVC', 'AS test']  = AS(y_test , classif2.predict(X_test))
result_df.loc['LinearSVC', 'PS test']  = PS(y_test , classif2.predict(X_test), averag
result_df.loc['LinearSVC', 'RS test']  = RS(y_test , classif2.predict(X_test), averag
```

```
classif3.fit(X_train, y_train)
```

```
Pipeline(memory=None,
         steps=[('tfidf',
                 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,
                                 min_df=1, ngram_range=(1, 2), norm='l2',
                                 preprocessor=None, smooth_idf=True,
                                 stop_words='english', strip_accents=None,
                                 sublinear_tf=True,
                                 token_pattern='(?u)\\b\\w\\w+\\b',
                                 tokenizer=None, use_idf=True,
                                 vocabulary=None)),
                ('LR',
                 LogisticRegression(C=5.0, class_weight=None, dual=False,
                                    fit_intercept=True, intercept_scaling=1,
                                    l1_ratio=None, max_iter=100,
                                    multi_class='auto', n_jobs=None,
                                    penalty='l2', random_state=None,
                                    solver='lbfgs', tol=0.0001, verbose=0,
                                    warm_start=False))],
         verbose=False)
```

```
result_df.loc['LogisticRegression', 'AS train'] = AS(y_train, classif3.predict(X_trai
result_df.loc['LogisticRegression', 'PS train'] = PS(y_train, classif3.predict(X_trai
result_df.loc['LogisticRegression', 'RS train'] = RS(y_train, classif3.predict(X_trai
result_df.loc['LogisticRegression', 'AS test']  = AS(y_test , classif3.predict(X_test
result_df.loc['LogisticRegression', 'PS test']  = PS(y_test , classif3.predict(X_test
result_df.loc['LogisticRegression', 'RS test']  = RS(y_test , classif3.predict(X_test
```

Результаты сравнения по метрикам accuracy и F1

```
print('F1 for MultiNB: ', f1 score(y train, classif.predict(X train), average = 'micr
```

```
F1 for MultiNB:  0.6533456662952764
```

```
print('F1 for LSVC: ', f1_score(y_train, classif2.predict(X_train), average = 'micro'
```

```
F1 for LSVC:  0.9758876579284829
```

```
print('F1 for LR: ', f1_score(y_train, classif3.predict(X_train), average = 'micro'))
```

```
F1 for LR:  0.9467300132439194
```

```
result_df
```

	AS train	PS train	RS train	AS test	PS test	RS test
MultinomialNB	0.653346	0.653346	0.653346	0.607397	0.607397	0.607397
LinearSVC	0.975888	0.975888	0.975888	0.722765	0.722765	0.722765
LogisticRegression	0.946730	0.946730	0.946730	0.722470	0.722470	0.722470

Вывод:

Методы классификации текстов на основе метода наивного Байеса работают хуже логистической регрессии. Возможно, это связано с тем, что количество классов 5, а не 2. При логистической регрессии точность (accuracy) достигает 94%. Лучше всего с задачей справилась модель LinearSVC - точность порядка 97%.