Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Отчет по рубежному контролю № 2

«Классификация текстов на основе методов наивного Байеса»
по курсу
“Методы машинного обучения”

Выполнил:
Али Диб А.Ж.
Студент группы ИУ5-22М

Москва, 2020

Цель работы

Классификация текстов на арабском языке с использованием разных видов классификат
классификаторов.

Набор данных

Набор данных представляет собой коллекцию арабских текстов, которая охватывает твит

Набор данных состоит из 9694 твитов и 246 872 слов, структурированных в текстовые фай

Документы в наборе данных делятся на 4 класса: объективные, положительные, отрицател

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from IPython.display import Image
from sklearn.datasets import load_iris, load_boston
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classificatio
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_l
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.naive_bayes import GaussianNB, MultinomialNB, ComplementNB, BernoulliN
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
import warnings
warnings.filterwarnings("ignore")
```



```
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)
        temp_f1=f1_score(temp_data_flt['t'].values,
            temp_data_flt['p'].values,average='weighted')
        # сохранение результата в словарь
        res[c] = (temp_acc,temp_f1)
    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 \t  Accuracy \t \t \t f1_score')
    for i in accs:
        if i != 'NEUTRAL':
          print('{} \t \t {} \t \t {}'.format(i, accs[i][0],accs[i][1]))
        else:
          print('NEUT \t \t {} \t \t {}'.format(accs[i][0],accs[i][1]))


# Загрузка данных
data = pd.read_csv('/content/gdrive/My Drive/Tweets.txt', delimiter='\t',header=Nor
```

	text	target
0	...بعد استقالة رئيس #المحكمة_الدستورية ننتظر استق	OBJ

Количество строки в нашем наборе данных:

```
data.shape
```

```
(9694, 2)
```

```
data.count()
```

```
text      9694
target    9694
dtype: int64
```

```
data.target.value_counts()
```

```
OBJ        6470
NEG        1642
NEUTRAL     805
POS         777
Name: target, dtype: int64
```

```
data.head()
```

	text	target
0	...بعد استقالة رئيس #المحكمة_الدستورية ننتظر استق	OBJ
1	...أهنئ الدكتور أحمد جمال الدين، القيادي بحزب مصر	POS
2	...البرادعي يستقوى بامريكا مرةاخرى و يرسل عصام ال	NEG
3	... الحرية_والعدالة	شاهد الآن: #ليلة_الاتحادية#	OBJ
4	...الوالدة لو اقولها بخاطري حشيشة تضحك بس من اقول	NEUTRAL

```
# Сформируем общий словарь для обучения моделей из обучающей и тестовой выборки
vocab_list = data['text'].tolist()
vocab_list[1:10]
```

```
['أهنئ الدكتور أحمد جمال الدين، القيادي بحزب مصر، بمناسبة صدور أولى روايته',
 'البرادعي يستقوى بامريكا مرة اخرى و يرسل عصام العريان الي واشنطن شئ مقرف',
 'يلة_الاتحادية أول فيلم استقصائي يتناول أسرار و كواليس تعرض لأول مرة حول حقيقة#'
 ل
 حد الفتيات كان بينهم حب كبير ولكن حدثت غلطة واحدة؟ فهل ستستمر هذه القصة ويتم'
 'أدعوكم لحضور الندوة الثقافية الأربعاء مركز اعداد القادة التفاصيل',
 سية بنفس وضعها السابق مستحيلة والطرمخة على جرائم الماضي لن تجعلنا نتقدم شبرا'
```


Количество уникальных слов в наборе данных.

```
vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 37561
```

Примеры некоторых слов, взятых из набора данных с их частотами.

```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
3491=استقالة
18456=رئيس
9298=المحكمة_الدستورية
30067=ننتظر
18457=رئيس_القضاء
7272=السودان
1942=أهنئ
6675=الدكتور
977=أحمد
```

```
test_features = vocabVect.transform(vocab_list)
```

```
vocabVect.get_feature_names()[100:120]
```

```
['14آذار',
 '15',
 '150',
 '151',
 '1515',
 '153',
 '1540',
 '155',
 '157',
 '15سنة',
 '15على',
 '16',
 '1615445',
 '1620995',
 '168',
 '16طالبا',
 '17',
 '172',
 '173',
 '1735']
```



```
ncv = CountVectorizer(ngram_range=(1,3))
ngram_features = ncv.fit_transform(vocab_list)
ngram_features
```

```
<9694x246872 sparse matrix of type '<class 'numpy.int64'>'
        with 369945 stored elements in Compressed Sparse Row format>
```

```
            score2=cross_val_score(pipeline1, data['text'], data['target'], scorinc
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('F1 = {}'.format(score2))
            print('============================')


vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(voca
classifiers_list = [LogisticRegression(C=3.0), LinearSVC(), KNeighborsClassifier()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
len(ncv.get_feature_names())
```

```
246872
```

```
# Теперь признаками являются N-граммы
ncv.get_feature_names()[1:30]
```

```
['00 00',
 '00 00 الاجتماع',
 '00 00 للإستفسار',
 '00 10',
 '00 10 00',
 '00 11',
 '00 11 00',
 '00 12',
 '00 12 00',
 '00 22',
 '00 22 00',
 '00 الاجتماع',
 '00 الاجتماع الأول',
 '00 القناة',
 '00 القناة عصام',
 '00 المنسق',
 '00 المنسق كريم',
 '00 بتوقيت',
 '00 بتوقيت القاهرة',
 '00 عصرا',
 '00 عصرا مع',
 '00 للإستفسار',
 '00 للإستفسار والتواصل',
 '00 مساء',
 '00 مساء على',
 '00 يتشرف',
 '00 يتشرف الدكتور',
 '000',
 '000 000']
```

```
tfidfv = TfidfVectorizer(ngram_range=(1,3))
tfidf_ngram_features = tfidfv.fit_transform(vocab_list)
tfidf_ngram_features
```

```
<9694x246872 sparse matrix of type '<class 'numpy.float64'>'
        with 369945 stored elements in Compressed Sparse Row format>
```

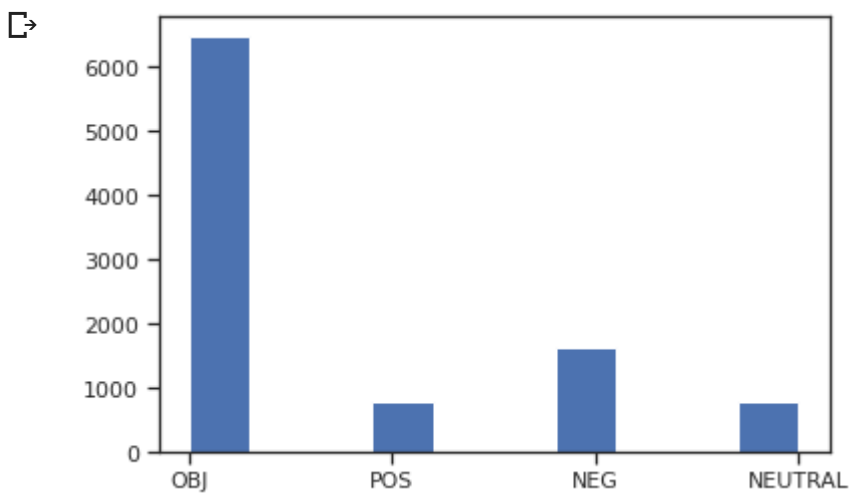
Подгонка и прогнозирование данных обучения и испыта

Your URL should open in a new window. If it does not, make sure that pop ups are not blocked and reopen the link.

```
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for v in vectorizers_list:
        for c in classifiers_list:
            pipeline1 = Pipeline([("vectorizer", v), ("classifier", c)])
            score = cross_val_score(pipeline1, data['text'], data['target'], scorin
```

```
                                '0165541351': 22, '0190555044': 23, '02': 24,
                                '029': 25, '03': 26, '033': 27, '04': 28,
                                '04681953': 29, ...})
Модель для классификации - LogisticRegression(C=3.0, class_weight=None, dual=Fals
                    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)
Accuracy = 0.6780478448262653
F1 = 0.5864210910123421
============================
Векторизация - TfidfVectorizer(analyzer='word', binary=False, decode_error='stri
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features=None
                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...
                vocabulary={'00': 0, '000': 1, '00202': 2, '005': 3, '009': 4,
                            '01': 5, '01001135875': 6, '01002118193': 7,
                            '01008880090': 8, '0101004800': 9, '0103424388': 1
                            '0105064942': 11, '01069883354': 12,
                            '0107381486': 13, '0109333999': 14, '011': 15,
                            '0111650008': 16, '0112': 17, '01201968': 18,
                            '01280423836': 19, '01289954776': 20, '0155': 21,
                            '0165541351': 22, '0190555044': 23, '02': 24,
                            '029': 25, '03': 26, '033': 27, '04': 28,
                            '04681953': 29, ...})
Модель для классификации - LinearSVC(C=1.0, class_weight=None, dual=True, fit_int
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
Accuracy = 0.6729924269121433
F1 = 0.6003665807484148
============================
Векторизация - TfidfVectorizer(analyzer='word', binary=False, decode_error='stri
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features=None
                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...
                vocabulary={'00': 0, '000': 1, '00202': 2, '005': 3, '009': 4,
                            '01': 5, '01001135875': 6, '01002118193': 7,
                            '01008880090': 8, '0101004800': 9, '0103424388': 1
                            '0105064942': 11, '01069883354': 12,
                            '0107381486': 13, '0109333999': 14, '011': 15,
                            '0111650008': 16, '0112': 17, '01201968': 18,
                            '01280423836': 19, '01289954776': 20, '0155': 21,
                            '0165541351': 22, '0190555044': 23, '02': 24,
                            '029': 25, '03': 26, '033': 27, '04': 28,
                            '04681953': 29, ...})
                                                                          m
                    weights='uniform')
Accuracy = 0.6544244315332183
F1 = 0.5972908213361873
============================
```

```
plt.hist(data['target'])
plt.show()
```




```
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['target'], t
```

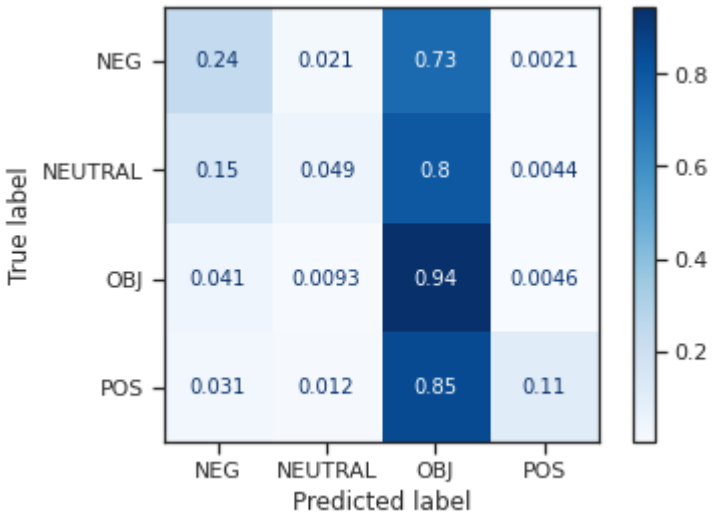
```
def sentiment(v, c):
    model = Pipeline(
        [("vectorizer", v),
```

```
        ("classifier", c)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print_accuracy_score_for_classes(y_test, y_pred)
    labels=['NEG', 'NEUTRAL', 'OBJ', 'POS']
    plot_confusion_matrix(model,X_test, y_test,display_labels=['NEG', 'NEUTRAL', '(
```

Использование Tfid Vectorizer и логистической регрессии

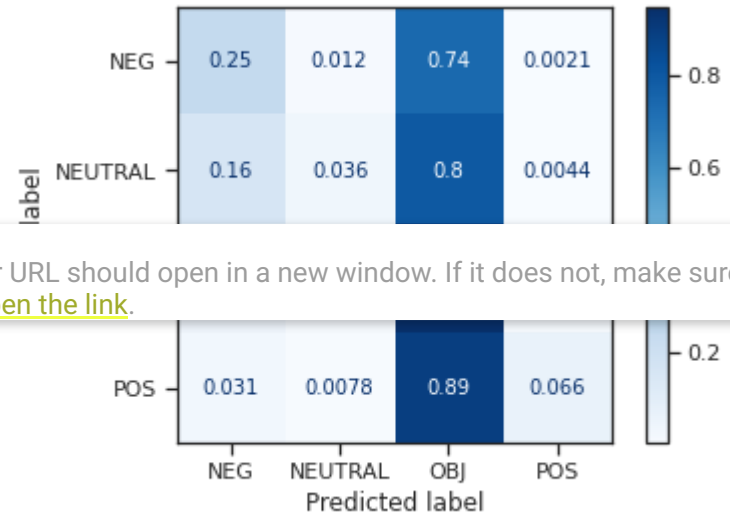
```
sentiment(TfidfVectorizer(), LogisticRegression(C=3.0))
```

Метка	Accuracy	f1_score
NEG	0.24329896907216494	0.3913764510779436
NEUT	0.04888888888888889	0.09322033898305083
OBJ	0.9449021627188465	0.9716706380725444
POS	0.10894941634241245	0.19649122807017544



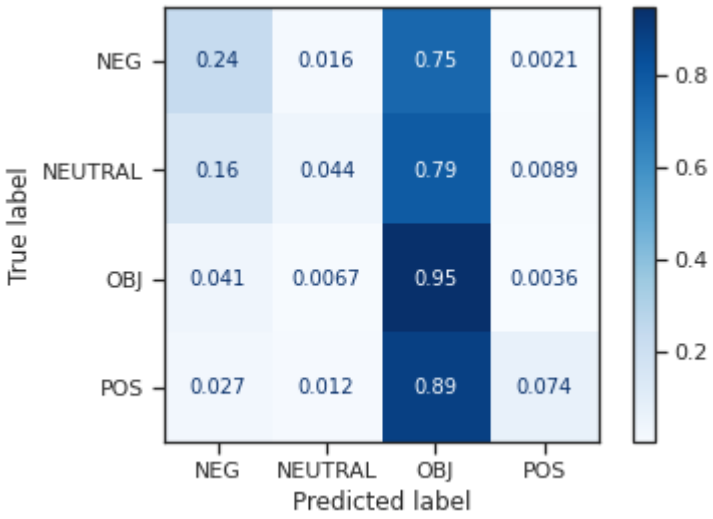

```
sentiment(TfidfVectorizer(ngram_range=(1,3)), LogisticRegression(C=3.0))
```

Метка	Accuracy	f1_score
NEG	0.24536082474226803	0.3940397350993377
NEUT	0.035555555555555556	0.06866952789699571
OBJ	0.9490216271884655	0.9738441215323645
POS	0.06614785992217899	0.12408759124087591



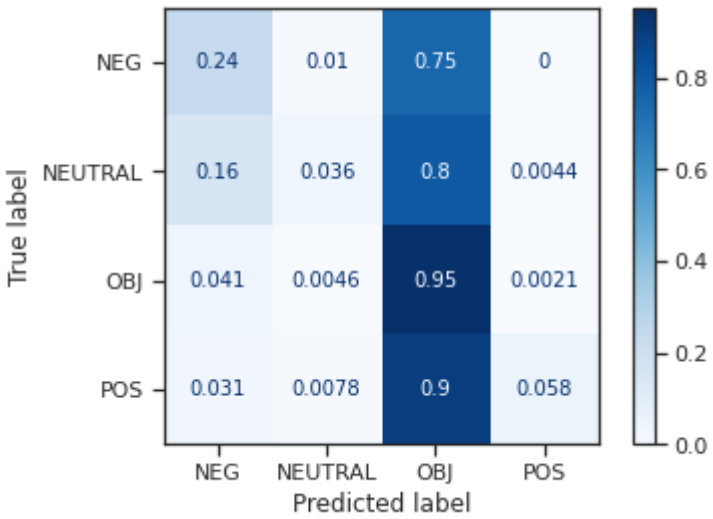
```
sentiment(TfidfVectorizer(ngram_range=(1,2)), LogisticRegression(C=3.0))
```

Метка	Accuracy	f1_score
NEG	0.23505154639175257	0.3806343906510851
NEUT	0.044444444444444446	0.08510638297872339
OBJ	0.9485066941297632	0.9735729386892178
POS	0.07392996108949416	0.13768115942028986



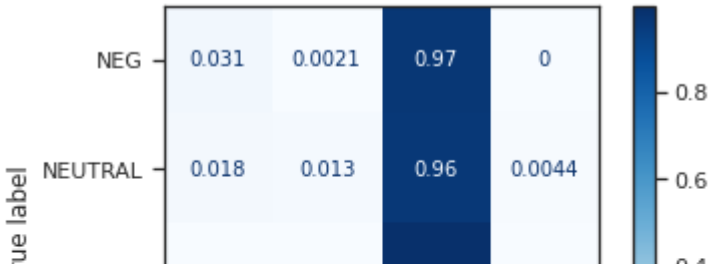

```
sentiment(TfidfVectorizer(ngram_range=(1,4)), LogisticRegression(C=3.0))
```

Метка	Accuracy	f1_score
NEG	0.23711340206185566	0.3833333333333333
NEUT	0.035555555555555556	0.06866952789699571
OBJ	0.9521112255406797	0.9754682141915062
POS	0.058365758754863814	0.11029411764705882




```
sentiment(TfidfVectorizer(ngram_range=(2,4)), LogisticRegression(C=3.0))
```

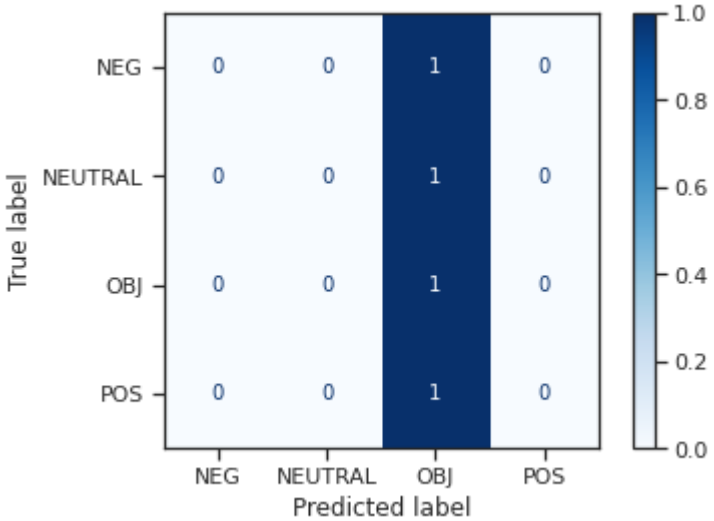
Метка	Accuracy	f1_score
NEG	0.030927835051546393	0.06000000000000001
NEUT	0.013333333333333334	0.02631578947368421
OBJ	0.9969104016477858	0.9984528107271788
POS	0.0038910505836575876	0.007751937984496124



Использование Tfid Vectorizer и полиномиального наивного бай

```
sentiment(TfidfVectorizer(ngram_range=(1,2)), MultinomialNB())
```

Метка	Accuracy	f1_score
NEG	0.0	0.0
NEUT	0.0	0.0
OBJ	1.0	1.0
POS	0.0	0.0



Использование Tfid Vectorizer и наивного байесовского Бернулл

```
sentiment(TfidfVectorizer(ngram_range=(1,2)), BernoulliNB())
```

Your URL should open in a new window. If it does not, make sure that pop ups are not blocked and reopen the link.