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

Рубежный контроль №2
«Методы обработки текстов»

ИСПОЛНИТЕЛЬ:

Федорова Антонина Алексеевна
Группа ИУ5-24М

_____

"__"_____2021 г.

Москва 2021

Целью работы является: решение задачи классификации текстов на основе любого датасета.

Задание:

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 два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer. В качестве классификаторов необходимо использовать два классификатора по варианту для Вашей группы:

Группа: ИУ5-24М

Классификатор 1: KNeighborsClassifier

Классификатор 2: Complement Naive Bayes (CNB)

Для каждого метода необходимо оценить качество классификации. Сделайте вывод о том, какой вариант векторизации признаков в паре с каким классификатором показал лучшее качество.

Для выполнения данной работы взят датасет с данными о спаме в тексте.

Рубежный контроль №2

Тема: Методы обработки текстов

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 два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer. В качестве классификаторов необходимо использовать два классификатора по варианту для Вашей группы:

Группа: ИУ5-24М

- Классификатор 1: KNeighborsClassifier
- Классификатор 2: Complement Naive Bayes (CNB)

Для каждого метода необходимо оценить качество классификации. Сделайте вывод о том, какой вариант векторизации признаков в паре с каким классификатором показал лучшее качество.

Для данной работы был выбран [датасет (https://www.kaggle.com/team-ai/spam-text-message-classification?select=SPAM+text+message+20170820+-+Data.csv)](https://www.kaggle.com/team-ai/spam-text-message-classification?select=SPAM+text+message+20170820+-+Data.csv)

In [17]:
```
port numpy as np
port pandas as pd
m typing import Dict, Tuple
m scipy import stats
m IPython.display import Image
m sklearn.datasets import load_iris, load_boston
m sklearn.feature_extraction.text import CountVectorizer, TfidfVect
m sklearn.model_selection import train_test_split
m sklearn.neighbors import KNeighborsClassifier
m sklearn.linear_model import LogisticRegression
m sklearn.model_selection import GridSearchCV, RandomizedSearchCV
m sklearn.metrics import accuracy_score, balanced_accuracy_score
m sklearn.metrics import precision_score, recall_score, f1_score, c
m sklearn.metrics import confusion_matrix
m sklearn.model_selection import cross_val_score
m sklearn.pipeline import Pipeline
m sklearn.metrics import mean_absolute_error, mean_squared_error, m
m sklearn.metrics import roc_curve, roc_auc_score
m sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR
port seaborn as sns
port matplotlib.pyplot as plt
atplotlib inline
s.set(style="ticks")
m sklearn.naive_bayes import ComplementNB
```

In [18]:
```
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 [19]:
```
# Загрузка данных
df = pd.read_csv('/Users/a.fedorova/Desktop/учеба/Великолепная мага
df.head()
```

Out[19]:

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

In [20]:
```
# Сформируем общий словарь для обучения моделей из обучающей и тест
vocab_list = df['Message'].tolist()
vocab_list[1:10]
```

Out[20]:
```
['Ok lar... Joking wif u oni...',
 "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 20
05. Text FA to 87121 to receive entry question(std txt rate)T&C's
apply 08452810075over18's",
 'U dun say so early hor... U c already then say...',
 "Nah I don't think he goes to usf, he lives around here though",
 "FreeMsg Hey there darling it's been 3 week's now and no word bac
k! I'd like some fun you up for it still? Tb ok! XxX std chgs to s
end, £1.50 to rcv",
 'Even my brother is not like to speak with me. They treat me like
aids patent.',
 "As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Ve
ttam)' has been set as your callertune for all Callers. Press *9 t
o copy your friends Callertune",
 'WINNER!! As a valued network customer you have been selected to
receivea £900 prize reward! To claim call 09061701461. Claim code
KL341. Valid 12 hours only.',
 'Had your mobile 11 months or more? U R entitled to Update to the
latest colour mobiles with camera for Free! Call The Mobile Update
Co FREE on 08002986030']
```

In [21]:
```
vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusV
```

```
Количество сформированных признаков - 8709
```

In [22]:
```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
until=8080
jurong=4370
point=5954
crazy=2334
available=1313
only=5567
in=4110
bugis=1763
great=3651
```

Использование класса CountVectorizer

In [23]:
```
test_features = vocabVect.transform(vocab_list)
test_features
test_features.todense()
```

Out[23]:
```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]])
```

Использование класса TfidfVectorizer

In [24]:
```
tfidfv = TfidfVectorizer(ngram_range=(1,3))
tfidf_ngram_features = tfidfv.fit_transform(vocab_list)
tfidf_ngram_features
```

Out[24]:
```
<5572x104934 sparse matrix of type '<class 'numpy.float64'>'
        with 217339 stored elements in Compressed Sparse Row forma
t>
```

In [25]:
```
tfidf_ngram_features.todense()
```

Out[25]:
```
matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

In [26]:
```
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for v in vectorizers_list:
        for c in classifiers_list:
            pipeline1 = Pipeline([("vectorizer", v), ("classifier",
            score = cross_val_score(pipeline1, df['Message'], df['C
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

Выполнение с классификаторами: ComplementNB, KNeighborsClassifier

In [27]:
```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), Tfid
classifiers_list = [ComplementNB(), KNeighborsClassifier()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(analyzer='word', binary=False, deco
de_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', inp
ut='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={'00': 0, '000': 1, '000pes': 2, '00870
4050406': 3,
                            '0089': 4, '0121': 5, '01223585236': 6
,
                            '01223585334': 7, '0125698789': 8, '02
': 9,
                            '0207': 10, '02072069400': 11, '020731
62414': 12,
                            '02085076972': 13, '021': 14, '03': 15
, '04': 16,
                            '0430': 17, '05': 18, '050703': 19, '0
578': 20,
                            '06': 21, '07': 22, '07008009200': 23,
                            '07046744435': 24, '07090201529': 25,
                            '07090298926': 26, '07099833605': 27,
                            '07123456789': 28, '0721072': 29, ...}
)
Модель для классификации - ComplementNB(alpha=1.0, class_prior=Non
e, fit_prior=True, norm=False)
Accuracy = 0.9664390153607633
============================
Векторизация - CountVectorizer(analyzer='word', binary=False, deco
de_error='strict',
```

```
                dtype=<class 'numpy.int64'>, encoding='utf-8', inp
ut='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={'00': 0, '000': 1, '000pes': 2, '00870
4050406': 3,
                            '0089': 4, '0121': 5, '01223585236': 6
,
                            '01223585334': 7, '0125698789': 8, '02
': 9,
                            '0207': 10, '02072069400': 11, '020731
62414': 12,
                            '02085076972': 13, '021': 14, '03': 15
, '04': 16,
                            '0430': 17, '05': 18, '050703': 19, '0
578': 20,
                            '06': 21, '07': 22, '07008009200': 23,
                            '07046744435': 24, '07090201529': 25,
                            '07090298926': 26, '07099833605': 27,
                            '07123456789': 28, '0721072': 29, ...}
)
Модель для классификации - KNeighborsClassifier(algorithm='auto',
leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=
5, p=2,
                     weights='uniform')
Accuracy = 0.9122387985297536
============================
Векторизация - TfidfVectorizer(analyzer='word', binary=False, deco
de_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_f
eatures=None,
                min_df=1, ngram_range=(1, 1), norm='l2', preproces
sor=None,
                smooth_idf=True, stop_words=None, strip_accents=No
ne,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\
\b',
                tokenizer=None, use...
                vocabulary={'00': 0, '000': 1, '000pes': 2, '00870
4050406': 3,
                            '0089': 4, '0121': 5, '01223585236': 6
,
                            '01223585334': 7, '0125698789': 8, '02
': 9,
                            '0207': 10, '02072069400': 11, '020731
62414': 12,
```

```
                                '02085076972': 13, '021': 14, '03': 15
, '04': 16,
                                '0430': 17, '05': 18, '050703': 19, '0
578': 20,
                                '06': 21, '07': 22, '07008009200': 23,
                                '07046744435': 24, '07090201529': 25,
                                '07090298926': 26, '07099833605': 27,
                                '07123456789': 28, '0721072': 29, ...}
)
Модель для классификации - ComplementNB(alpha=1.0, class_prior=Non
e, fit_prior=True, norm=False)
Accuracy = 0.9675155382353525
============================
Векторизация - TfidfVectorizer(analyzer='word', binary=False, deco
de_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_f
eatures=None,
                min_df=1, ngram_range=(1, 1), norm='l2', preproces
sor=None,
                smooth_idf=True, stop_words=None, strip_accents=No
ne,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\
\b',
                tokenizer=None, use...
                vocabulary={'00': 0, '000': 1, '000pes': 2, '00870
4050406': 3,
                                '0089': 4, '0121': 5, '01223585236': 6
,
                                '01223585334': 7, '0125698789': 8, '02
': 9,
                                '0207': 10, '02072069400': 11, '020731
62414': 12,
                                '02085076972': 13, '021': 14, '03': 15
, '04': 16,
                                '0430': 17, '05': 18, '050703': 19, '0
578': 20,
                                '06': 21, '07': 22, '07008009200': 23,
                                '07046744435': 24, '07090201529': 25,
                                '07090298926': 26, '07099833605': 27,
                                '07123456789': 28, '0721072': 29, ...}
)
Модель для классификации - KNeighborsClassifier(algorithm='auto',
leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=
5, p=2,
                     weights='uniform')
Accuracy = 0.924806089662772
============================
```

In [ ]: