Рубежный контроль №2

Необходимо подготовить отчет по рубежному контролю и разместить его в Вашем репозитории. Вы можете использовать титульный лист, или в начале ноутбука в текстовой ячейке указать Ваши Ф.И.О. и группу.

Тема: Методы обработки текстов. ¶

Решение задачи классификации 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:

ИУ5-22М, ИУ5И-22М RandomForestClassifier 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.

In [3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import numpy as np
import pandas as pd
from typing import Dict, Tuple
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import ComplementNB
import seaborn as sns
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
import matplotlib.pyplot as plt
from sklearn.svm import SVC, NuSVC, OneClassSVM, SVR, NuSVR, LinearSVR

%matplotlib inline
sns.set(style="ticks")

# !pip install category_encoders
```

In [4]:

```
categories = ['alt.atheism', 'talk.religion.misc', 'comp.graphics', 'sci.space']
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
# data
```

Анализируем датасет и готовим категориальный признак

In [5]:

```
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 [6]:

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

```
Количество сформированных признаков - 34118
```

In [7]:

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

```
rych=27180
festival=13876
ed=12325
ac=4289
uk=31720
hawkes=15898
subject=29644
3ds=2362
where=33286
```

In [8]:

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

Out[8]:

```
<2034x34118 sparse matrix of type '<class 'numpy.int64'>'
        with 323433 stored elements in Compressed Sparse Row format>
```

In [9]:

```
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, newsgroups['data'], newsgroups['target'], scoring='ac
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('============================')
```

In [10]:

```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab), TfidfVectorizer(vocabulary = corpusVo
classifiers_list = [RandomForestClassifier(), ComplementNB()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000':
2, '00000': 3,
                               '000000': 4, '000005102000': 5, '000021': 6,
                               '000062david42': 7, '0000vec': 8, '0001': 9,
                               '000100255pixel': 10, '000406': 11, '00041032': 12,
                               '0004136': 13, '0004246': 14, '0004422': 15,
                               '00044513': 16, '0004847546': 17, '0005': 18,
                               '0007': 19, '00090711': 20, '000usd': 21,
                               '0010580b': 22, '001125': 23, '0012': 24,
                               '001200201pixel': 25, '001428': 26, '001555': 27,
                               '001718': 28, '001757': 29, ...})
Модель для классификации - RandomForestClassifier()
Accuracy = 0.8795476892822025
============================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000':
2, '00000': 3,
                               '000000': 4, '000005102000': 5, '000021': 6,
                               '000062david42': 7, '0000vec': 8, '0001': 9,
                               '000100255pixel': 10, '000406': 11, '00041032': 12,
                               '0004136': 13, '0004246': 14, '0004422': 15,
                               '00044513': 16, '0004847546': 17, '0005': 18,
                               '0007': 19, '00090711': 20, '000usd': 21,
                               '0010580b': 22, '001125': 23, '0012': 24,
                               '001200201pixel': 25, '001428': 26, '001555': 27,
                               '001718': 28, '001757': 29, ...})
Модель для классификации - ComplementNB()
Accuracy = 0.9523107177974435
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000':
2, '00000': 3,
                               '000000': 4, '000005102000': 5, '000021': 6,
                               '000062david42': 7, '0000vec': 8, '0001': 9,
                               '000100255pixel': 10, '000406': 11, '00041032': 12,
                               '0004136': 13, '0004246': 14, '0004422': 15,
                               '00044513': 16, '0004847546': 17, '0005': 18,
                               '0007': 19, '00090711': 20, '000usd': 21,
                               '0010580b': 22, '001125': 23, '0012': 24,
                               '001200201pixel': 25, '001428': 26, '001555': 27,
                               '001718': 28, '001757': 29, ...})
Модель для классификации - RandomForestClassifier()
Accuracy = 0.8716814159292037
============================
Векторизация - TfidfVectorizer(vocabulary={'00': 0, '000': 1, '0000':
2, '00000': 3,
                               '000000': 4, '000005102000': 5, '000021': 6,
                               '000062david42': 7, '0000vec': 8, '0001': 9,
                               '000100255pixel': 10, '000406': 11, '00041032': 12,
                               '0004136': 13, '0004246': 14, '0004422': 15,
                               '00044513': 16, '0004847546': 17, '0005': 18,
                               '0007': 19, '00090711': 20, '000usd': 21,
                               '0010580b': 22, '001125': 23, '0012': 24,
                               '001200201pixel': 25, '001428': 26, '001555': 27,
                               '001718': 28, '001757': 29, ...})
Модель для классификации - ComplementNB()
```

```
Accuracy = 0.9341199606686331
==========================
```

Лучшая точность была у CountVectorizer с ComplementNB - 0.952

In [ ]: