Рубежный контроль №2

Екатерина Максимова ИУ5-23М

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: LinearSVC, Multinomial Naive Bayes (MNB)

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

In [1]:
```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from sklearn.naive_bayes import GaussianNB, MultinomialNB, ComplementNB, Bern
from sklearn.svm import SVC, NuSVC, LinearSVC
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, classifi
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squ
from sklearn.metrics import roc_curve, roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [2]:
```
data = pd.read_csv("./spam.csv", delimiter=',')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  5572 non-null   object
 1   Message   5572 non-null   object
```

```
dtypes: object(2)
memory usage: 87.2+ KB
```

In [3]:
```
data.Category.value_counts()
```

Out[3]:
```
ham     4825
spam     747
Name: Category, dtype: int64
```

In [4]:
```
data['Category'].unique()
```

Out[4]:
```
array(['ham', 'spam'], dtype=object)
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(data['Message'], data['Ca
```

In [6]:
```
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]))

    print('Accuracy', accuracy_score(y_true, y_pred))
    print('Precision', precision_score(y_true, y_pred, average=None))
```

In [7]:
```
def spam(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)
```

In [8]:
```
spam(TfidfVectorizer(ngram_range=(1,5)), LinearSVC())
```

```
Метка     Accuracy
ham       0.9972260748959778
spam      0.9260869565217391
Accuracy 0.9874401913875598
Precision [0.98831615 0.98156682]
```

In [9]:
```
spam(CountVectorizer(), LinearSVC())
```

```
Метка     Accuracy
ham       0.9979195561719834
spam      0.9130434782608695
Accuracy 0.986244019138756
Precision [0.98629198 0.98591549]
```

In [10]:
```
spam(CountVectorizer(), MultinomialNB())
```

```
Метка     Accuracy
ham       0.9958391123439667
spam      0.9304347826086956
Accuracy 0.9868421052631579
Precision [0.98898072 0.97272727]
```

In [11]:
```
spam(TfidfVectorizer(ngram_range=(1,5)), MultinomialNB())
```

```
Метка     Accuracy
ham       1.0
spam      0.48695652173913045
Accuracy 0.9294258373205742
Precision [0.92435897 1.        ]
```

Вывод: Наилучший результат показал TfidfVectorizer(ngram_range=(1,5)) с LinearSVC()

In [ ]:
```
```