Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

Рубежный контроль №2

По курсу
Методы машинного обучения в АСОИУ

Студент ИУ5-21М _____ Я.С. Стельмах
(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель _____ **Ю.Е. Гапанюк**
(Подпись, дата) (И.О.Фамилия)

2025 г.

Рубежный контроль 2.

датасет https://www.kaggle.com/datasets/pashupatigupta/emotion-detection-from-text

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 ИУ5-21М

KNeighborsClassifier
LogisticRegression

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 scipy import stats
from IPython.display import Image
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, classific
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_squa
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, Linear
```

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [2]:
```
# Функция для вычисления accuracy по классам (из примера)
def accuracy_score_for_classes(y_true: np.ndarray, y_pred: np.ndarray) -> dict
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    classes = np.unique(y_true)
    res = {}
    for c in classes:
        temp_data_flt = df[df['t'] == c]
        temp_acc = accuracy_score(temp_data_flt['t'].values, temp_data_flt['p'
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(y_true: np.ndarray, y_pred: np.ndarray):
    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 [3]:
```
data = pd.read_csv('/Users/Iana/Desktop/2M/MMO/data/tweet_emotions.csv', encoc
```

In [4]:
```
data.head()
```

Out[4]:

	tweet_id	sentiment	content
0	1956967341	empty	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	wants to hang out with friends SOON!
4	1956968416	neutral	@dannycastillo We want to trade with someone w...

In [5]:
```
data.shape
```

Out[5]:
```
(40000, 3)
```

Пропуски в данных:

In [6]:
```
print(data.isnull().sum())
```

```
tweet_id     0
sentiment    0
content      0
dtype: int64
```

Удаление строк с пропусками

In [7]:
```
data = data.dropna(subset=['content', 'sentiment'])
```

Оставляем только нужные столбцы

In [8]:
```
data = data[['content', 'sentiment']].reset_index(drop=True)
```

Переименуем столбцы для удобства

In [9]:
```
data.columns = ['text', 'label']
```

Создание словаря

In [11]:
```
# Сформируем общий словарь для обучения моделей из обучающей и тестовой выборк
vocab_list = data['text'].tolist()
vocab_list[1:10]
```

Out[11]:
```
['Layin n bed with a headache  ughhhh...waitin on your call...',
 'Funeral ceremony...gloomy friday...',
 'wants to hang out with friends SOON!',
 '@dannycastillo We want to trade with someone who has Houston tickets, but n
o one will.',
 "Re-pinging @ghostridah14: why didn't you go to prom? BC my bf didn't like m
y friends",
 "I should be sleep, but im not! thinking about an old friend who I want. but
he's married now. damn, &amp;amp; he wants me 2! scandalous!",
 'Hmmm. http://www.djhero.com/ is down',
 '@charviray Charlene my love. I miss you',
 "@kelcouch I'm sorry  at least it's Friday?"]
```

In [12]:
```
vocabVect = CountVectorizer()
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
```

In [14]:
```
print(f"\nРазмер словаря: {len(corpusVocab)}")
```

```
Размер словаря: 48212
```

In [15]:
```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
know=24017
was=45848
listenin=25463
to=43187
bad=4919
habit=18430
earlier=13732
and=3374
started=40270
```

In [16]:
```
test_features = vocabVect.transform(vocab_list)
```

In [19]:
```
# Функция для векторизации и классификации
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for v in vectorizers_list:
        for c in classifiers_list:
            pipeline = Pipeline([("vectorizer", v), ("classifier", c)])
            score = cross_val_score(pipeline, data['text'], data['label'],
                                    scoring='accuracy', cv=3).mean()
            print(f"Векторизация: {v}")
            print(f"Классификатор: {c}")
            print(f"Accuracy: {score:.4f}")
            print("============================")
```

In [18]:
```
# Определение векторизаторов и классификаторов
vectorizers_list = [CountVectorizer(vocabulary=corpusVocab),
                    TfidfVectorizer(vocabulary=corpusVocab)]
classifiers_list = [KNeighborsClassifier(), LogisticRegression(C=3.0, max_iter
```

Выполнение оценки

In [20]:
```
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
Векторизация: CountVectorizer(vocabulary={'00': 0, '000': 1, '000th': 2, '006':
3, '00am': 4,
                               '00pm': 5, '01': 6, '01theone': 7, '02': 8,
                               '023': 9, '024': 10, '0255': 11, '02mxjj': 12,
                               '03': 13, '04': 14, '04182012154': 15, '05': 16,
                               '053agj': 17, '05ixbj': 18, '06': 19, '060': 20,
                               '0600': 21, '06am': 22, '07': 23, '0783l': 24,
                               '07am': 25, '07jzs': 26, '07k6e': 27, '07k6x': 28,
                               '07kbp': 29, ...})
Классификатор: KNeighborsClassifier()
Accuracy: 0.2398
============================
Векторизация: CountVectorizer(vocabulary={'00': 0, '000': 1, '000th': 2, '006':
3, '00am': 4,
                               '00pm': 5, '01': 6, '01theone': 7, '02': 8,
                               '023': 9, '024': 10, '0255': 11, '02mxjj': 12,
                               '03': 13, '04': 14, '04182012154': 15, '05': 16,
                               '053agj': 17, '05ixbj': 18, '06': 19, '060': 20,
                               '0600': 21, '06am': 22, '07': 23, '0783l': 24,
                               '07am': 25, '07jzs': 26, '07k6e': 27, '07k6x': 28,
                               '07kbp': 29, ...})
Классификатор: LogisticRegression(C=3.0, max_iter=1000)
Accuracy: 0.3077
============================
Векторизация: TfidfVectorizer(vocabulary={'00': 0, '000': 1, '000th': 2, '006':
3, '00am': 4,
                               '00pm': 5, '01': 6, '01theone': 7, '02': 8,
                               '023': 9, '024': 10, '0255': 11, '02mxjj': 12,
                               '03': 13, '04': 14, '04182012154': 15, '05': 16,
                               '053agj': 17, '05ixbj': 18, '06': 19, '060': 20,
                               '0600': 21, '06am': 22, '07': 23, '0783l': 24,
                               '07am': 25, '07jzs': 26, '07k6e': 27, '07k6x': 28,
                               '07kbp': 29, ...})
Классификатор: KNeighborsClassifier()
Accuracy: 0.2177
============================
Векторизация: TfidfVectorizer(vocabulary={'00': 0, '000': 1, '000th': 2, '006':
3, '00am': 4,
                               '00pm': 5, '01': 6, '01theone': 7, '02': 8,
                               '023': 9, '024': 10, '0255': 11, '02mxjj': 12,
                               '03': 13, '04': 14, '04182012154': 15, '05': 16,
                               '053agj': 17, '05ixbj': 18, '06': 19, '060': 20,
                               '0600': 21, '06am': 22, '07': 23, '0783l': 24,
                               '07am': 25, '07jzs': 26, '07k6e': 27, '07k6x': 28,
                               '07kbp': 29, ...})
Классификатор: LogisticRegression(C=3.0, max_iter=1000)
Accuracy: 0.3290
============================
```

In [21]:

```
# Дополнительная оценка для лучшей модели (LogisticRegression с TfidfVectorize
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['label'
                                                    test_size=0.2, random_stat
pipeline_best = Pipeline([("vectorizer", TfidfVectorizer(vocabulary=corpusVoca
                          ("classifier", LogisticRegression(C=3.0, max_iter=100
```

```
pipeline_best.fit(X_train, y_train)
y_pred = pipeline_best.predict(X_test)
```

In [22]:
```
# Вывод подробных метрик
print("\nОтчет по лучшей модели (LogisticRegression с TfidfVectorizer):")
print(classification_report(y_test, y_pred))
print("\nAccuracy по классам:")
print_accuracy_score_for_classes(y_test, y_pred)
```

```
Отчет по лучшей модели (LogisticRegression с TfidfVectorizer):
              precision    recall  f1-score   support

       anger       0.00      0.00      0.00        19
     boredom       0.00      0.00      0.00        31
       empty       0.14      0.01      0.01       162
  enthusiasm       0.00      0.00      0.00       163
         fun       0.09      0.02      0.03       338
   happiness       0.35      0.35      0.35      1028
        hate       0.48      0.16      0.24       268
        love       0.50      0.37      0.42       762
     neutral       0.34      0.57      0.42      1740
      relief       0.33      0.04      0.07       352
     sadness       0.32      0.26      0.29      1046
    surprise       0.27      0.06      0.10       425
       worry       0.33      0.46      0.38      1666

    accuracy                           0.34      8000
   macro avg       0.24      0.18      0.18      8000
weighted avg       0.33      0.34      0.31      8000


Accuracy по классам:
Метка          Accuracy
anger          0.0
boredom          0.0
empty          0.006172839506172839
enthusiasm          0.0
fun          0.01775147928994083
happiness          0.3472762645914397
hate          0.16044776119402984
love          0.3700787401574803
neutral          0.5678160919540229
relief          0.03977272727272727
sadness          0.26003824091778205
surprise          0.06352941176470588
worry          0.4567827130852341
```

```
/Users/Iana/Desktop/2M/MMO/.venv/lib/python3.11/site-packages/sklearn/metric
s/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/Iana/Desktop/2M/MMO/.venv/lib/python3.11/site-packages/sklearn/metric
s/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/Iana/Desktop/2M/MMO/.venv/lib/python3.11/site-packages/sklearn/metric
s/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Лучшая комбинация — LogisticRegression с TfidfVectorizer