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

ОТЧЕТ

ПО РУБЕЖНЫЙ КОНТРОЛЬ №2

ПО ДИСЦИПЛИНЕ «МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ»

Тема: Методы обработки текстов

Студент ИУ5И-25М _____ **Ши Чжань**
(Группа) (Подпись, дата) (И.О.Фамилия)

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

2025 г.

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

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: ИУ5-25М, ИУ5И-25М, ИУ5-25МВ: SVC и LogisticRegression

Датасет - https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset

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

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Загрузка данных

```
# Загрузка train и test
train_df = pd.read_csv("/content/train.csv", header=None)
test_df = pd.read_csv("/content/test.csv", header=None)

# Названия колонок
train_df.columns = ['label', 'title', 'description']
test_df.columns = ['label', 'title', 'description']

# Объединяем заголовок и описание
X_train = train_df['title'] + " " + train_df['description']
y_train = train_df['label'].astype(str)

X_test = test_df['title'] + " " + test_df['description']
y_test = test_df['label'].astype(str)

# Просмотр классов
print("Уникальные классы:", sorted(y_train.unique()))
```

Уникальные классы: ['1', '2', '3', '4', 'Class Index']

Векторизация

```
# CountVectorizer
count_vectorizer = CountVectorizer()
X_train_count = count_vectorizer.fit_transform(X_train)
X_test_count = count_vectorizer.transform(X_test)

# TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Logistic Regression

```
# Count
log_count = LogisticRegression(max_iter=1000)
log_count.fit(X_train_count, y_train)
y_pred_log_count = log_count.predict(X_test_count)

# TF-IDF
log_tfidf = LogisticRegression(max_iter=1000)
log_tfidf.fit(X_train_tfidf, y_train)
y_pred_log_tfidf = log_tfidf.predict(X_test_tfidf)
```

SVC

```
# Count
svc_count = SVC()
svc_count.fit(X_train_count[:10000], y_train[:10000])
y_pred_svc_count = svc_count.predict(X_test_count[:10000])

# TF-IDF
svc_tfidf = SVC()
svc_tfidf.fit(X_train_tfidf[:10000], y_train[:10000])
y_pred_svc_tfidf = svc_tfidf.predict(X_test_tfidf[:10000])
```

Оценка

```
def evaluate_model(name, y_true, y_pred):
    print(f"\n==== {name} ====")
    print("Accuracy:", accuracy_score(y_true, y_pred))
    print(classification_report(y_true, y_pred))

evaluate_model("Logistic Regression (Count)", y_test, y_pred_log_count)
evaluate_model("Logistic Regression (TF-IDF)", y_test, y_pred_log_tfidf)
evaluate_model("SVC (Count)", y_test, y_pred_svc_count)
evaluate_model("SVC (TF-IDF)", y_test, y_pred_svc_tfidf)
```

==== Logistic Regression (Count) ====

Accuracy: 0.9088277858176556

	precision	recall	f1-score	support
1	0.92	0.90	0.91	1900
2	0.96	0.97	0.96	1900
3	0.88	0.87	0.88	1900
4	0.89	0.89	0.89	1900
Class Index	0.00	0.00	0.00	1
accuracy			0.91	7601
macro avg	0.73	0.73	0.73	7601
weighted avg	0.91	0.91	0.91	7601

==== Logistic Regression (TF-IDF) ====

Accuracy: 0.9193527167477964

	precision	recall	f1-score	support

1	0.93	0.91	0.92	1900
2	0.96	0.98	0.97	1900
3	0.89	0.88	0.89	1900
4	0.90	0.91	0.90	1900
Class Index	0.00	0.00	0.00	1
accuracy			0.92	7601
macro avg	0.74	0.74	0.74	7601
weighted avg	0.92	0.92	0.92	7601

==== SVC (Count) ====

Accuracy: 0.842389159321142

	precision	recall	f1-score	support
1	0.86	0.84	0.85	1900
2	0.91	0.90	0.91	1900
3	0.86	0.76	0.81	1900
4	0.76	0.87	0.81	1900
Class Index	0.00	0.00	0.00	1
accuracy			0.84	7601
macro avg	0.68	0.67	0.67	7601
weighted avg	0.85	0.84	0.84	7601

==== SVC (TF-IDF) ====

Accuracy: 0.8571240626233391

	precision	recall	f1-score	support
1	0.88	0.87	0.87	1900
2	0.93	0.91	0.92	1900
3	0.88	0.75	0.81	1900
4	0.76	0.90	0.82	1900
Class Index	0.00	0.00	0.00	1
accuracy			0.86	7601
macro avg	0.69	0.69	0.69	7601
weighted avg	0.86	0.86	0.86	7601

Confusion Matrix

```
def plot_conf_matrix(y_true, y_pred, title):
    labels = sorted(y_test.unique())
    cm = confusion_matrix(y_true, y_pred, labels=labels)

    plt.figure(figsize=(8,6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=labels, yticklabels=labels)
    plt.title(title)
    plt.xlabel("Predicted")
    plt.ylabel("True")
    plt.show()

plot_conf_matrix(y_test, y_pred_log_tfidf, "Logistic Regression (TF-IDF)")
```