# МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э. Баумана

Факультет «Информатика и системы управления» Кафедра «Систем обработки информации и управления»

ОТЧЕТ  Рубежный контроль №2  по дисциплине «Методы машинного обучения»	
Тема: «Методы обработки текстов»	
ИСПОЛНИТЕЛЬ: группа ИУ5И-21М	<u>Ли Яцзинь</u> оио
ПРЕПОДАВАТЕЛЬ:	подпись"
Москва - 2024	

#### Заданте

## 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 моей руппы:

КСоседиКлассификатор (KNeighborsClassifier)

Логистическая регрессия (LogisticRegression)

## Загрузить набор данных

Это набор данных рецензий на фильмы, которые будут использоваться для задачи НЛП по анализу настроений. Он имеет форму предложений, где каждому предложению присваивается оценка настроения от 0 до 4 (1 = Очень Плохо 2 = Плохо 3 = Нейтрально 4 = Хорошо 5 = Очень хорошо).

#### загрузить наоор данных

```
# @title Загрузить набор данных
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Load dataset
data = pd.read_csv('/content/data/train.csv')

# Extracting features and labels
texts = data['review_text'].tolist()
labels = data['class_index'].tolist()

# Splitting the dataset into training and testing sets
train_texts, test_texts, train_labels, test_labels = train_test_split(texts, labels)
```

Рисунок 1- Код загрузить набор данных

#### Преобразовать в вектор объектов

Используйте два разных метода для преобразования текста в векторы объектов (CountVectorizer и TfidfVectorizer).

```
# @title Преобразовать в вектор объектов

# Method 1: Using CountVectorizer
count_vectorizer = CountVectorizer()
X_train_count = count_vectorizer.fit_transform(train_texts)
X_test_count = count_vectorizer.transform(test_texts)

# Method 2: Using TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(train_texts)
X_test_tfidf = tfidf_vectorizer.transform(test_texts)
```

Рисунок 2- Код преобразовать в вектор объектов

### Определить классификатор

В зависимости от параметров класса выбраны следующие классификаторы: KNeighboursClassifier и LogisticReгрессия.

```
[8] # @title Определить классификатор

# Define classifiers
knn_classifier = KNeighborsClassifier()
logistic_classifier = LogisticRegression()

↑ ↓ ⊖ ■ ┆ ☐ ⅰ
```

Рисунок 3-Код определить классификатор

# Обучение и тестирование с помощью CountVectorizer

Используйте классификатор CountVectorizer для обучения и тестирования функций, созданных KNeighborsClassifier и LogisticRegrade соответственно.

```
# @title Обучение и тестирование с помощью CountVectorizer

print("Training and testing with CountVectorizer:")
km_classifier.fit(X_train_count, *train_labels)
kmn_pred_count == kmn_classifier.predict(X_test_count)
print("KNN *Accuracy *(CountVectorizer):", *accuracy_score(test_labels, *kmn_pred_count))
print("KNN *Classification *Report *(CountVectorizer):"n, *classification_report(test_labels, *kmn_pred_count))

logistic_classifier.fit(X_train_count, *train_labels)
logistic_pred_count == logistic_classifier.predict(X_test_count)
print("Logistic *Regression *Accuracy *(CountVectorizer):", *accuracy_score(test_labels, *logistic_pred_count))
print("Logistic *Regression *Classification *Report *(CountVectorizer):", *classification_report(test_labels, *logistic_pred_count))
```

Рисунок4-Код Обучение и тестирование с помощью CountVectorizer

	Training and	testing with	CountYoot	oni zoni			
<b>→</b>	TO 11 TO 12				07007		
	KNN Accuracy				01331		
	KNN Classific	100 to 100 (100 (100 (100 (100 (100 (100 (100			7624800000000000000		
		precision	recall	f1-score	support		
	0	0.48	0.22	0.30	760		
	1	0.52	0.26	0.35	3287		
	2	0.64	0.93	0.76	11129		
	3	0.58	0.25	0.35	3982		
	4	0. 56	0.15	0. 23	981		
	accuracy			0.62	20139		
	macro avg	0.55	0.36	0.40	20139		
	weighted avg	0.60	0.62	0.57	20139		
	Logistic Regr	ession Accura	icy (Count	Vectorizer	): 0.66517	70197129947	
	Logistic Regr Logistic Regr						
	Logistic Regr Logistic Regr		fication		untVectori:		
		ession Classi	fication	Report (Co	untVectori:		
	Logistic Regr	ession Classi	fication	Report (Co	untVectori:		
	Logistic Regr 0	ression Classi precision	fication recall	Report (Con f1-score	untVectori: support		
	Logistic Regr	ression Classi precision 0.50	fication recall 0.21	Report (Con f1-score 0.30	untVectori: support 760		
	Logistic Regr 0	ression Classi precision 0.50 0.54	fication recall 0.21 0.37	Report (Con f1-score 0.30 0.44	untVectori: support 760 3287		
	Logistic Regr	pression Classi precision 0.50 0.54 0.71	0.21 0.37 0.90	Report (Con f1-score 0.30 0.44 0.79	untVectori: support 760 3287 11129		
	Logistic Regr	0.50 0.50 0.71 0.59	fication recall 0.21 0.37 0.90 0.44	Report (Cot f1-score 0.30 0.44 0.79 0.50	untVectori: support 760 3287 11129 3982		
	Logistic Regr	0.50 0.50 0.71 0.59	fication recall 0.21 0.37 0.90 0.44	Report (Con f1-score 0.30 0.44 0.79 0.50 0.41	untVectori: support 760 3287 11129 3982 981		

Рисунок5- результаты обучение и тестирование с помощью CountVectorizer

# Обучение и тестирование с помощью TfidfVectorizer

Используйте классификатор TfidfVectorizer для обучения и тестирования функций, созданных KNeighborsClassifier и LogisticRegrade соответственно.

```
▼ Обучение и тестирование с помощью TfidfVectorizer
# @title Обучение и тестирование с помощью TfidfVectorizer
#Training and testing with TfidfVectorizer:")
kmn_classifier.fit(X_train_tfidf, train_labels)
kmn_pred_tfidf = kmn_classifier.predict(X_test_tfidf)
print("KNN Accuracy (TfidfVectorizer):\n", accuracy_score(test_labels, kmn_pred_tfidf))
print("KNN Classification Report (TfidfVectorizer):\n", classification_report(test_labels, kmn_pred_tfidf))
logistic_classifier.fit(X_train_tfidf, train_labels)
logistic_pred_tfidf = logistic_classifier.predict(X_test_tfidf)
print("Logistic Regression Accuracy (TfidfVectorizer):\n", accuracy_score(test_labels, logistic_pred_tfidf))
print("Logistic Regression Classification Report (TfidfVectorizer):\n", classification_report(test_labels, logistic_pred_tfidf))
```

Рисунок6- результаты обучение и тестирование с помощью TfidfVectorizer

```
# @title Обучение и тестирование с помощью CountVectorizer

print("Training and testing with CountVectorizer:")
knn_classifier.fit(X_train_count, train_labels)
knn_pred_count = knn_classifier.predict(X_test_count)
print("KNN Accuracy (CountVectorizer):", accuracy_score(test_labels, knn_pred_count))
print("KNN Classification Report (CountVectorizer):\n", classification_report(test_labels, knn_pred_count))

logistic_classifier.fit(X_train_count, train_labels)
logistic_pred_count = logistic_classifier.predict(X_test_count)
print("Logistic Regression Accuracy (CountVectorizer):", accuracy_score(test_labels, logistic_pred_count))
print("Logistic Regression Classification Report (CountVectorizer):\n", classification_report(test_labels, logistic_pred_count))
```

Рисунок7- результаты обучение и тестирование с помощью TfidfVectorizer

→ Training and testing with CountVectorizer:

KNN Accuracy (CountVectorizer): 0.6200407170167337

KNN Classification Report (CountVectorizer):

min orappirio	in orappirioacidi nopor c		(occurrence that).		
	precision	recal1	f1-score	support	
0	0.48	0. 22	0.30	760	
1	0.52	0.26	0.35	3287	
2	0.64	0. 93	0.76	11129	
3	0.58	0.25	0.35	3982	
4	0. 56	0.15	0. 23	981	
accuracy			0.62	20139	
macro avg	0.55	0.36	0.40	20139	
weighted avg	0.60	0.62	0.57	20139	

Logistic Regression Accuracy (CountVectorizer): 0.6651770197129947 Logistic Regression Classification Report (CountVectorizer):

	precision	recal1	fl-score	support
0	0.50	0. 21	0.30	760
1	0.54	0.37	0.44	3287
2	0.71	0.90	0.79	11129
3	0.59	0.44	0.50	3982
4	0.58	0.32	0.41	981
accuracy			0.67	20139
macro avg	0.59	0.45	0.49	20139
weighted avg	0.64	0.67	0.64	20139

₹ Training and testing with CountVectorizer:

KNN Accuracy (CountVectorizer): 0.6200407170167337

KNN Classification Report (CountVectorizer):

precision recall fl-score

	precision	recal1	f1-score	support
0	0.48	0. 22	0.30	760
1	0.52	0.26	0.35	3287
2	0.64	0.93	0.76	11129
3	0.58	0.25	0.35	3982
4	0.56	0.15	0.23	981
accuracy			0.62	20139
macro avg	0.55	0.36	0.40	20139
weighted avg	0.60	0.62	0.57	20139

Logistic Regression Accuracy (CountVectorizer): 0.6651770197129947 Logistic Regression Classification Report (CountVectorizer):

	precision	recal1	fl-score	suppor
0	0.50	0. 21	0.30	760
1	0.54	0.37	0.44	3287
2	0.71	0.90	0.79	11129
3	0.59	0.44	0.50	3982
4	0.58	0.32	0.41	981
accuracy			0.67	20139
macro avg	0.59	0.45	0.49	20139
weighted avg	0.64	0.67	0.64	20139