Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное 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 и искусственный интеллект

КАФЕДРА Системы обработки информации и управления

Рубежный контроль №2 по курсу «Методы машинного обучения в автоматизированных системах обработки информации и управления»

Подготовили:

Чжан Чжиси

ИУ5И-25М

09.05.2024

Проверил:

Гапанюк Ю. Е.

2024 г.

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:

Группа	Классификатор №1	Классификатор №2
ИУ5-21М, ИУ5И-21М, ИУ5Ц-21М	KNeighborsClassifier	LogisticRegression
ИУ5-22М, ИУ5И-22М	RandomForestClassifier	LogisticRegression
ИУ5-23М, ИУ5И-23М	LinearSVC	LogisticRegression
ИУ5-24М, ИУ5И-24М	GradientBoostingClassifier	LogisticRegression
ИУ5-25М, ИУ5И-25М, ИУ5И-26М	SVC	LogisticRegression

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

Группа	Классификатор №1	Классификатор №2
ИУ5И-25М	SVC	LogisticRegression

Набор данных Spam SMS Collection содержит ряд текстовых сообщений, которые разделены на два типа: спам и обычные сообщения (не спам). Этот набор данных является классическим примером задачи бинарной классификации, где цель состоит в том, чтобы предсказать, является ли сообщение спамом или обычным.

Сообщения в наборе данных могут касаться различных тем, включая рекламу, мошенничество, акции и т. д. Тексты сообщений обычно имеют разную длину и стиль языка, что увеличивает сложность задачи классификации.

1 Загрузка данных:

```
import pandas as pd

# Загрузка данных
df = pd.read_csv(r"C:\Users\12974\Desktop\sms+spam+collection\SMSSpamCollection", sep='\t', header=None, names=["label", "text"])

# Отображение первых нескольких строк набора данных
print(df.head())
```

```
  label                                               text
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
```

Рис.1- Загрузка данных

2 Предварительная обработка данных:

```
import re
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer

# Инициализация стеммера
stemmer = SnowballStemmer("russian")

# Загрузка списка стоп-слов
stop_words = set(stopwords.words("russian"))

# Функция предварительной обработки текста
def preprocess_text(text):
    # Очистка текста от ненужных символов и разбиение на слова
    words = re.findall(r'\b\w+\b', text.lower())
    # Удаление стоп-слов и применение стемминга
    words = [stemmer.stem(word) for word in words if word not in stop_words]
    # Сборка токенов обратно в текст
    return " ".join(words)

# Применение функции предварительной обработки к столбцу с текстом
df["text"] = df["text"].apply(preprocess_text)

# Вывод нескольких строк обновленного набора данных
for index, row in df.iterrows():
    print("label:", row["label"])
    print("text:", row["text"])
    print()
```

```
label: ham
text: go until jurong point crazy available only in bugis n great world la e buffet cine there got amore wat

label: ham
text: ok lar joking wif u oni

label: spam
text: free entry in 2 a wkly comp to win fa cup final tkts 21st may 2005 text fa to 87121 to receive entry question std txt rate t c s app
ly 08452810075over18 s

label: ham
text: u dun say so early hor u c already then say

label: ham
text: nah i don t think he goes to usf he lives around here though

label: spam
text: freemsg hey there darling it s been 3 week s now and no word back i d like some fun you up for it still tb ok xxx std chgs to send 1
50 to rcv
```

```
label: ham
text: even my brother is not like to speak with me they treat me like aids patent

label: ham
text: as per your request melle melle oru minnaminunginte nurungu vettam has been set as your callertune for all callers press 9 to copy y
our friends callertune

label: spam
text: winner as a valued network customer you have been selected to receivea 900 prize reward to claim call 09061701461 claim code kl341 v
alid 12 hours only

label: spam
text: had your mobile 11 months or more u r entitled to update to the latest colour mobiles with camera for free call the mobile update co
free on 08002986030

label: ham
text: i m gonna be home soon and i don t want to talk about this stuff anymore tonight k i ve cried enough today
```

Рис.2- Предварительная обработка данных

3 Разделите данные на обучающий и тестовый наборы:

```
from sklearn.model_selection import train_test_split

# Разделение данных на признаки (X) и целевую переменную (y)
X = df['text']
y = df['label']

# Разделение данных на тренировочный и тестовый наборы (80% - тренировочный, 20% -
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Проверка размеров наборов данных
print("Размер тренировочного набора данных:", X_train.shape[0])
print("Размер тестового набора данных:", X_test.shape[0])
```

```
Размер тренировочного набора данных: 4457
Размер тестового набора данных: 1115
```

Рис.3- Разделите данные на обучающий и тестовый наборы

4 Векторизация признаков с помощью CountVectorizer

```
from sklearn.model_selection import train_test_split

# Разделение данных на признаки (X) и целевую переменную (y)
X = df['text']
y = df['label']

# Разделение данных на тренировочный и тестовый наборы (80% - тренировочный, 20% -
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Проверка размеров наборов данных
print("Размер тренировочного набора данных:", X_train.shape[0])
print("Размер тестового набора данных:", X_test.shape[0])
```

```
Размер тренировочного набора данных: 4457
Размер тестового набора данных: 1115
```

Рис.4- Векторизация признаков с помощью CountVectorizer

5 Векторизация признаков с помощью TfidfVectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Создание экземпляра TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

# Преобразование тренировочного и тестового наборов данных с помощью TfidfVectorizer
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Вывод размеров преобразованных наборов данных
print("Размер матрицы признаков тренировочного набора (TfidfVectorizer):", X_train_tfidf.shape)
print("Размер матрицы признаков тестового набора (TfidfVectorizer):", X_test_tfidf.shape)
```

```
Размер матрицы признаков тренировочного набора (TfidfVectorizer): (4457, 7702)
Размер матрицы признаков тестового набора (TfidfVectorizer): (1115, 7702)
```

Рис.5- Векторизация признаков с помощью TfidfVectorizer

Из результатов видно, что размер матриц признаков для тренировочного и тестового наборов данных одинаков, как и при использовании CountVectorizer, что вполне ожидаемо, поскольку количество уникальных маркеров остается неизменным.

6 Решение задач классификации текстов с помощью классификаторов SVC

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Инициализация и обучение классификатора SVC на матрице признаков от CountVectorizer
svc_count = SVC()
svc_count.fit(X_train_count, y_train)

# Предсказание меток для тестового набора данных
y_pred_svc_count = svc_count.predict(X_test_count)

# Оценка точности классификации
accuracy_svc_count = accuracy_score(y_test, y_pred_svc_count)
print("Точность классификации с использованием CountVectorizer и SVC:", accuracy_svc_count)
```

```
Точность классификации с использованием CountVectorizer и SVC: 0.9865470852017937
```

Рис.6- Решение задач классификации текстов с помощью классификаторов SVC

7 Решение задач классификации текстов с помощью классификаторов LogisticRegression

```
from sklearn.linear_model import LogisticRegression

# Инициализация и обучение классификатора LogisticRegression на матрице признаков от Co
lr_count = LogisticRegression()
lr_count.fit(X_train_count, y_train)

# Предсказание меток для тестового набора данных
y_pred_lr_count = lr_count.predict(X_test_count)

# Оценка точности классификации
accuracy_lr_count = accuracy_score(y_test, y_pred_lr_count)
print("Точность классификации с использованием CountVectorizer и LogisticRegression:", accuracy_lr_count)
```

```
Точность классификации с использованием CountVectorizer и LogisticRegression: 0.9883408071748879
```

Рис.7- Решение задач классификации текстов с помощью классификаторов LogisticRegression

Теперь мы можем сравнить результаты классификации с использованием CountVectorizer и двух разных классификаторов: SVC и LogisticRegression. Оба метода векторизации показали высокую точность, но в данном случае LogisticRegression показал немного лучший результат.

8 Использование TfidfVectorizer с классификаторами SVC и LogisticRegression

```
# Инициализация и обучение классификатора SVC на матрице признаков от TfidfVectorizer
svc_tfidf = SVC()
svc_tfidf.fit(X_train_tfidf, y_train)

# Предсказание меток для тестового набора данных
y_pred_svc_tfidf = svc_tfidf.predict(X_test_tfidf)

# Оценка точности классификации
accuracy_svc_tfidf = accuracy_score(y_test, y_pred_svc_tfidf)
print("Точность классификации с использованием TfidfVectorizer и SVC:", accuracy_svc_tfidf)

# Инициализация и обучение классификатора LogisticRegression на матрице признаков от Ti
lr_tfidf = LogisticRegression()
lr_tfidf.fit(X_train_tfidf, y_train)

# Предсказание меток для тестового набора данных
y_pred_lr_tfidf = lr_tfidf.predict(X_test_tfidf)

# Оценка точности классификации
accuracy_lr_tfidf = accuracy_score(y_test, y_pred_lr_tfidf)
print("Точность классификации с использованием TfidfVectorizer и LogisticRegression:", accuracy_lr_tfidf)
```

```
Точность классификации с использованием TfidfVectorizer и SVC: 0.989237668161435
Точность классификации с использованием TfidfVectorizer и LogisticRegression: 0.9766816143497757
```

Рис.8- Использование TfidfVectorizer с классификаторами SVC и LogisticRegression

результаты классификации с использованием TfidfVectorizer и двух разных классификаторов показывают следующие точности:

Для классификатора SVC с использованием TfidfVectorizer точность составляет около 98.9%.
Для классификатора LogisticRegression с использованием TfidfVectorizer точность составляет около 97.7%.
Таким образом, на этот раз классификатор SVC показал немного лучший результат по сравнению с LogisticRegression при использовании TfidfVectorizer.

ВЫВОДЫ

Классификатор SVC:
CountVectorizer: точность классификации составила около 98.7%.
TfidfVectorizer: точность классификации составила около 98.9%.
Классификатор LogisticRegression:
CountVectorizer: точность классификации составила около 98.8%.
TfidfVectorizer: точность классификации составила около 97.7%.

Исходя из этих результатов, можно сделать следующие выводы:
Для классификатора SVC, использование TfidfVectorizer показало немного лучшее качество по сравнению с CountVectorizer.
Для классификатора LogisticRegression, использование CountVectorizer показало немного лучшее качество по сравнению с TfidfVectorizer.
Таким образом, выбор оптимального метода векторизации признаков зависит от конкретного классификатора и характеристик задачи. В данном случае, если

наибольшее внимание уделяется точности классификации, то выбором может быть TfidfVectorizer с классификатором SVC. Однако, стоит также учитывать другие факторы, такие как интерпретируемость модели, скорость обучения и предсказания, а также специфика задачи.