Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика, искусственный интеллект и системы управления»

Кафедра «Системы обработки информации и управления»

Рубежный контроль № 2

по дисциплине «Методы машинного обучения»

Методы обработки текстов

Вариант 8

ИСПОЛНИТЕЛЬ:

студентка ИУ5-23М

Морозевич М.А.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

___ "_____" 2024 г.

Москва, 2024

1 Вариант и задание

Необходимо решить задачу классификации текстов на основе датасета, классификация может быть бинарной или многоклассовой.

Необходимо сформировать два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer.

В качестве классификаторов необходимо использовать два классификатора:

– LinearSVC;

– LogisticRegression.

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

2 Описание набора данных

Для набора данных проведите устранение пропусков для одного (произвольного) числового признака с использованием метода заполнения модой.

Будет использован набор данных для классификации текстовых документов, который содержит 2225 текстовых данных и пять категорий документов.

Данные содержат 2 столбца:

- Текст: Содержит различные категории текстовых данных
- Метка: Содержит метки для пяти различных категорий (0,1,2,3,4)
    1. Политика = 0
    2. Спорт = 1
    3. Технологии = 2
    4. Развлечения = 3
    5. Бизнес = 4

Добавим данные и разделим данные на обучающую и тестовую выборки.

```
data = pd.read_csv('df_file.csv')
```

```
[3] data.head()
```

	Text	Label
0	Budget to set scene for election\n \n Gordon B...	0
1	Army chiefs in regiments decision\n \n Militar...	0
2	Howard denies split over ID cards\n \n Michael...	0
3	Observers to monitor UK election\n \n Minister...	0
4	Kilroy names election seat target\n \n Ex-chat...	0

Далее: Посмотреть рекомендованные графики

```
[4] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2225 entries, 0 to 2224
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Text    2225 non-null   object
 1   Label   2225 non-null   int64
dtypes: int64(1), object(1)
memory usage: 34.9+ KB
```

```
X, Y = data['Text'], data['Label']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
```

3 Векторизация текстовых данных

Преобразуем текстовые данные в векторы действительных чисел, которые понятны моделям машинного обучения. Она является шагом в извлечении признаков.

Используем 2 метода:

- Мешок слов: включает в себя токенизацию, создание словаря и создание вектора
- TF-IDF: числовой статистический показатель, который отражает важность слова для документа.

Векторизация текстовых данных

```
[7] count_vect = CountVectorizer()
    X_train_counts = count_vect.fit_transform(X_train)
    X_test_counts = count_vect.transform(X_test)
```

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

Используем их в дальнейшем при обучении различных классификаторов.

4 Обучение и сравнение классификаторов

Для классификации текстов будут использованы LinearSVC и LogisticRegression.

Обучим классификаторы для данных, векторизованных каждым из способов.

```
[20] gbc = LinearSVC(max_iter = 1000)
     start_time = time.time()
     gbc.fit(X_train_counts, y_train)
     train_time = time.time() - start_time
     time_arr.append(train_time)
     pred_gbc_counts = gbc.predict(X_test_counts)
     print("Точность CountVectorizer и LinearSVC: ", accuracy_score(y_test, pred_gbc_counts))
```

```
Точность CountVectorizer и LinearSVC:  0.9640449438202248
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn(
```

Точность CountVectorizer и LinearSVC: 0.9640449438202248

```
gbc = LinearSVC()
start_time = time.time()
gbc.fit(X_train_tfidf, y_train)
train_time = time.time() - start_time
time_arr.append(train_time)
pred_gbc_tfidf = gbc.predict(X_test_tfidf)
print("Точность TfidfVectorizer и LinearSVC: ", accuracy_score(y_test, pred_gbc_tfidf))
```

```
Точность TfidfVectorizer и LinearSVC:  0.9797752808988764
```

Точность TfidfVectorizer и LinearSVC: 0.9797752808988764

```
[22] lr = LogisticRegression(max_iter = 1000)
     start_time = time.time()
     lr.fit(X_train_counts, y_train)
     train_time = time.time() - start_time
     time_arr.append(train_time)
     pred_lr_counts = lr.predict(X_test_counts)
     print("Точность CountVectorizer и Logistic Regression: ", accuracy_score(y_test, pred_lr_counts))
```

```
Точность CountVectorizer и Logistic Regression:  0.9640449438202248
```

Точность CountVectorizer и Logistic Regression: 0.9640449438202248

```
[23] lr = LogisticRegression(max_iter = 1000)
     start_time = time.time()
     lr.fit(X_train_tfidf, y_train)
     train_time = time.time() - start_time
     time_arr.append(train_time)
     pred_lr_tfidf = lr.predict(X_test_tfidf)
     print("Точность TfidfVectorizer и Logistic Regression: ", accuracy_score(y_test, pred_lr_tfidf))
```

```
Точность TfidfVectorizer и Logistic Regression:  0.9595505617977528
```

Точность TfidfVectorizer и Logistic Regression: 0.9595505617977528

Сведем результаты обучения в одну таблицу для сравнения.

Сравнение результатов

```
from tabulate import tabulate

data = [
    ["(CountVectorizer и LogisticRegression)", accuracy_score(y_test, pred_lr_counts), time_arr[2]],
    ["(CountVectorizer и LinearSVC)", accuracy_score(y_test, pred_gbc_counts), time_arr[0]],
    ["(TfidfVectorizer и LogisticRegression)", accuracy_score(y_test, pred_lr_tfidf), time_arr[3]],
    ["(TfidfVectorizer и LinearSVC)", accuracy_score(y_test, pred_gbc_tfidf), time_arr[1]]
]

sorted_data = sorted(data, key=lambda x: x[1], reverse=True)

print(tabulate(sorted_data,['Комбинация','Точность','Время обучения'], tablefmt="grid"))
```

```
+---------------------------------------+------------+------------------+
| Комбинация                            |   Точность |   Время обучения |
+=======================================+============+==================+
| (TfidfVectorizer и LinearSVC)         |   0.979775 |         0.151996 |
+---------------------------------------+------------+------------------+
| (CountVectorizer и LogisticRegression) |   0.964045 |        25.957    |
+---------------------------------------+------------+------------------+
| (CountVectorizer и LinearSVC)         |   0.964045 |         1.2834   |
+---------------------------------------+------------+------------------+
| (TfidfVectorizer и LogisticRegression) |   0.959551 |         3.33244  |
+---------------------------------------+------------+------------------+
```

Из таблицы можно отметить, что лучший результат показала комбинация «TfidfVectorizer+LinearSVC» (97,9%), а наихудший – «TfidfVectorizer+LinearSVC» (95,9%). При этом результаты обучения всех комбинаций в целом достаточно высокие, выше 95%.

С другой стороны, наименьшее время обучения продемонстрировала комбинация для «TfidfVectorizer+LinearSVC» (0,15 с), а наибольшее – «CountVectorizer+LogisticRegression» (25,96 с).

В итоге, наилучший результат у комбинации – «TfidfVectorizer+LinearSVC». Более того, оба метода с использованием CountVectorizer выполняются дольше, чем с TfidfVectorizer.