

Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Рубежный контроль 2

Вариант №2

Выполнила:

студентка группы ИУ5-22М

Миронова Александра

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 RandomForestClassifier и LogisticRegression.

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


```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import pandas as pd
import time
```

```
from google.colab import drive
drive.mount('/content/drive')
```

 Mounted at /content/drive

```
# Загрузка данных
df = pd.read_csv('/content/drive/MyDrive/train_40k.csv')
```

```
df.head(10)
```



| | productId | Title | userId | Helpfulness | Score | Time | Text | Cat1 | Cat2 | Cat3 |
|---|------------|-------------------------------------|----------------|-------------|-------|-----------|---|----------------------|--------------|----------------|
| 0 | B000E46LYG | Golden Valley Natural Buffalo Jerky | A3MQDNGHDJU4MK | 0/0 | 3.0 | -1 | The description and photo on this product need... | grocery gourmet food | meat poultry | jerky |
| 1 | B000GRA6N8 | Westing Game | unknown | 0/0 | 5.0 | 860630400 | This was a great book!!!! It is well thought t... | toys games | games | unknown |
| 2 | B000GRA6N8 | Westing Game | unknown | 0/0 | 5.0 | 883008000 | I am a first year teacher, teaching 5th grade.... | toys games | games | unknown |
| 3 | B000GRA6N8 | Westing Game | unknown | 0/0 | 5.0 | 897696000 | I got the book at my bookfair at school lookin... | toys games | games | unknown |
| 4 | B00000DMDQ | I SPY A is For Jigsaw Puzzle 63pc | unknown | 2/4 | 5.0 | 911865600 | Hi! I'm Martine Redman and I created this puzz... | toys games | puzzles | jigsaw puzzles |
| 5 | B00000DMER | ThinkFun Rush Hour | unknown | 2/2 | 5.0 | 912816000 | My eight year old loves this game, whenever he... | toys games | games | board games |
| 6 | B00004RYGX | Beetle Juice (1988) | unknown | 1/1 | 4.0 | 918000000 | The real joy of this movie doesn't lie in its ... Okav Tim | grocery gourmet food | beverages | juices |

Далее:

[Посмотреть рекомендованные графики](#)

df.info()

```
>>> <class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   productId       40000 non-null  object
 1   Title           39984 non-null  object
 2   userId          40000 non-null  object
 3   Helpfulness     40000 non-null  object
 4   Score           40000 non-null  float64
 5   Time            40000 non-null  int64
 6   Text            40000 non-null  object
 7   Cat1            40000 non-null  object
 8   Cat2            40000 non-null  object
 9   Cat3            40000 non-null  object
dtypes: float64(1), int64(1), object(8)
memory usage: 3.1+ MB
```

проверим пропуски в данных и устроим их

na_mask = df.isna()

na_counts = na_mask.sum()

na_counts

```
>>> productId      0
Title             16
userId            0
Helpfulness       0
Score             0
Time              0
Text              0
Cat1              0
Cat2              0
Cat3              0
dtype: int64
```

```
df.dropna(inplace=True)
na_mask = df.isna()
na_counts = na_mask.sum()
na_counts
```

```
➞ productId      0
   Title          0
   userId        0
   Helpfulness    0
   Score          0
   Time           0
   Text           0
   Cat1           0
   Cat2           0
   Cat3           0
   dtype: int64
```

```
# Разделим набор данных на обучающую и тестовую выборки
X, Y = df['Text'], df['Cat2']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

```
time_arr = []
```

```
# векторизация признаков с помощью CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_test_counts = count_vect.transform(X_test)
```

```
# векторизация признаков с помощью TfidfVectorizer
tfidf_vect = TfidfVectorizer()
X_train_tfidf = tfidf_vect.fit_transform(X_train)
X_test_tfidf = tfidf_vect.transform(X_test)
```

```
# Произведем обучения двух классификаторов (по варианту) для CountVectorizer

# RandomForestClassifier
gbc = RandomForestClassifier()
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 + RandomForestClassifier):", accuracy_score(y_test, pred_gbc_counts))

# Logistic Regression
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 + LogisticRegression):", accuracy_score(y_test, pred_lr_counts))
```

⇒ Точность (CountVectorizer + RandomForestClassifier): 0.5410779042140803
Точность (CountVectorizer + LogisticRegression): 0.5997248968363136

```
# Произведем обучения двух классификаторов (по варианту) для TfidfVectorizer


# RandomForestClassifier
gbc = RandomForestClassifier()
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)

from tabulate import tabulate

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

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

# Вывод отсортированных данных в виде таблицы
print(tabulate(sorted_data, ['Модели', 'Точность валидации', 'Время обучения'], tablefmt="grid"))
```



| Модели | Точность валидации | Время обучения |
|--|--------------------|----------------|
| (TfidfVectorizer + LogisticRegression) | 0.622233 | 196.294 |
| (CountVectorizer + LogisticRegression) | 0.599725 | 183.254 |
| (TfidfVectorizer + LinearSVC) | 0.547205 | 164.485 |
| (CountVectorizer + LinearSVC) | 0.541078 | 532.346 |

Лучше всего показал себя TFIDF векторайзер в паре с логистической регрессией