Уралова Е.А., ИУ5-24М

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

Загружаем данные:

```
df = pd.read_csv('https://raw.githubusercontent.com/reisanar/datasets/master/IMDB_movies.csv')
print(df.head())
print(df.info())
```

```
   Rank                    Title                     Genre  \
0     1  Guardians of the Galaxy   Action,Adventure,Sci-Fi
1     2               Prometheus  Adventure,Mystery,Sci-Fi
2     3                    Split           Horror,Thriller
3     4                     Sing   Animation,Comedy,Family
4     5            Suicide Squad  Action,Adventure,Fantasy

                                         Description             Director  \
0  A group of intergalactic criminals are forced ...           James Gunn
1  Following clues to the origin of mankind, a te...         Ridley Scott
2  Three girls are kidnapped by a man with a diag...  M. Night Shyamalan
3  In a city of humanoid animals, a hustling thea...  Christophe Lourdelet
4  A secret government agency recruits some of th...           David Ayer

                                        Actors  Year  Runtime (Minutes)  \
0  Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...  2014                121
1  Noomi Rapace, Logan Marshall-Green, Michael Fa...  2012                124
2  James McAvoy, Anya Taylor-Joy, Haley Lu Richar...  2016                117
3  Matthew McConaughey,Reese Witherspoon, Seth Ma...  2016                108
4  Will Smith, Jared Leto, Margot Robbie, Viola D...  2016                123

   Rating   Votes  Revenue (Millions)  Metascore
0     8.1  757074              333.13       76.0
1     7.0  485820              126.46       65.0
2     7.3  157606              138.12       62.0
3     7.2   60545              270.32       59.0
4     6.2  393727              325.02       40.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Rank                1000 non-null   int64
 1   Title               1000 non-null   object
 2   Genre               1000 non-null   object
 3   Description         1000 non-null   object
 4   Director            1000 non-null   object
 5   Actors              1000 non-null   object
 6   Year                1000 non-null   int64
 7   Runtime (Minutes)   1000 non-null   int64
 8   Rating              1000 non-null   float64
 9   Votes               1000 non-null   int64
 10  Revenue (Millions)  872 non-null    float64
 11  Metascore           936 non-null    float64
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
None
```

Разделим данные на обучающую и тестовую выборки:

```
X = df['Description']
y = df['Genre']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Векторизация текстовых данных с помощью CountVectorizer:

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

GradientBoostingClassifier + CountVectorizer:

```
gb_clf = GradientBoostingClassifier()
gb_clf.fit(X_train_counts, y_train)
y_pred_gb = gb_clf.predict(X_test_counts)
accuracy_gb = accuracy_score(y_test, y_pred_gb)
print("Accuracy using GradientBoostingClassifier and CountVectorizer:", accuracy_gb)
```

```
Accuracy using GradientBoostingClassifier and CountVectorizer: 0.075
```

LogisticRegression + CountVectorizer:

```
lr_clf = LogisticRegression(max_iter=1000)
lr_clf.fit(X_train_counts, y_train)
y_pred_lr = lr_clf.predict(X_test_counts)
accuracy_lr = accuracy_score(y_test, y_pred_lr)
print("Accuracy using LogisticRegression and CountVectorizer:", accuracy_lr)
```

```
Accuracy using LogisticRegression and CountVectorizer: 0.06
```

Векторизация текстовых данных с помощью TfidfVectorizer:

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

GradientBoostingClassifier + TfidfVectorizer:

```
gb_clf_tfidf = GradientBoostingClassifier()
gb_clf_tfidf.fit(X_train_tfidf, y_train)
y_pred_gb_tfidf = gb_clf_tfidf.predict(X_test_tfidf)
accuracy_gb_tfidf = accuracy_score(y_test, y_pred_gb_tfidf)
print("Accuracy using GradientBoostingClassifier and TfidfVectorizer:", accuracy_gb_tfidf)
```

```
Accuracy using GradientBoostingClassifier and TfidfVectorizer: 0.06
```

LogisticRegression + TfidfVectorizer:

```
lr_clf_tfidf = LogisticRegression(max_iter=1000)
lr_clf_tfidf.fit(X_train_tfidf, y_train)
y_pred_lr_tfidf = lr_clf_tfidf.predict(X_test_tfidf)
accuracy_lr_tfidf = accuracy_score(y_test, y_pred_lr_tfidf)
print("Accuracy using LogisticRegression and TfidfVectorizer:", accuracy_lr_tfidf)
```

```
Accuracy using LogisticRegression and TfidfVectorizer: 0.06
```

Вывод результатов:

```
results_summary = pd.DataFrame(columns=['Classifier/Vectorizer', 'CountVectorizer', 'TfidfVectorizer'])
results_summary.loc[0] = ['GradientBoostingClassifier', accuracy_gb, accuracy_gb_tfidf]
results_summary.loc[1] = ['LogisticRegression', accuracy_lr, accuracy_lr_tfidf]

print(results_summary)
```

```
             Classifier/Vectorizer  CountVectorizer  TfidfVectorizer
     0  GradientBoostingClassifier            0.075             0.06
     1          LogisticRegression            0.060             0.06
```

Получили очень низкие значения для всех четырех вариантов. Самым же лучшим из них оказался вариант векторизации данных CountVectorizer в паре с классификатором GradientBoostingClassifier.