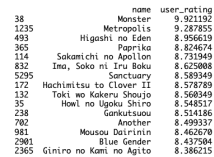


# Система рекомендаций Аниме



```
1 Downloading anime.csv to /Users/snipghost/Desktop  
2 100% ██████████ 915k/915k [00:00<00:00, 4.41MB/s]  
3 100% ██████████ 915k/915k [00:00<00:00, 4.37MB/s]
```

```

1 | import numpy as np
2 | import pandas as pd
3 | import matplotlib.pyplot as plt
4 | from sklearn.preprocessing import MultiLabelBinarizer, LabelBinarizer
5 | from sklearn import preprocessing

```

```

1 | df = pd.read_csv('anime.csv')
2 | print(df.shape)
3 | df.head()

```

```

1 | (12294, 7)

```

```

1 | .dataframe tbody tr th {
2 |     vertical-align: top;
3 | }
4 |
5 | .dataframe thead th {
6 |     text-align: right;
7 | }

```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama&#039;	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

## Anime.csv

```

1 | anime_id - myanimelist.net's unique id identifying an anime.
2 | name - full name of anime.
3 | genre - comma separated list of genres for this anime.
4 | type - movie, TV, OVA, etc.
5 | episodes - how many episodes in this show. (1 if movie).
6 | rating - average rating out of 10 for this anime.
7 | members - number of community members that are in this anime's "group".

```

```

1 | df1 = df[df.isna().any(axis=1)]
2 | df1.shape

```

```

1 | (277, 7)

```

```

1 | print(df.shape)
2 | df = df.dropna()
3 | print(df.shape)
4 | df.reset_index(drop=True, inplace=True)
5 | df.head()

```

```

1 | (12294, 7)
2 | (12017, 7)

```

```
1 | .dataframe tbody tr th {
2 |     vertical-align: top;
3 | }
4 |
5 | .dataframe thead th {
6 |     text-align: right;
7 | }
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama&#039;	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

```
1 | def normalize(df, cols):
2 |     x = df[cols]
3 |     print(x.shape)
4 |     min_max_scaler = preprocessing.MinMaxScaler()
5 |     x_scaled = min_max_scaler.fit_transform(x)
6 |     print(x_scaled)
7 |     print(x_scaled.shape)
8 |     dataset = pd.DataFrame(x_scaled, columns = cols)
9 |     print(dataset.shape)
10 |    print(dataset.tail())
11 |    #     origin_columns = df.columns
12 |    #     df = df.drop(columns=cols)
13 |    #     df = pd.concat([df, dataset], ignore_index=True, axis=1)
14 |    #     df.columns = origin_columns
15 |    for col in cols:
16 |        df[col] = dataset[col]
17 |    return df
```

```
1 | df['episodes'] = pd.to_numeric(df['episodes'], errors='coerce')
2 | df = df.fillna(df.mean())
3 | df.head()
```

```
1 | .dataframe tbody tr th {
2 |     vertical-align: top;
3 | }
4 |
5 | .dataframe thead th {
6 |     text-align: right;
7 | }
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1.0	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64.0	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51.0	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24.0	9.17	673572
4	9969	Gintama&#039;	Action, Comedy, Historical, Parody, Samurai, S...	TV	51.0	9.16	151266

```
1 | df = normalize(df, ['rating', 'members', 'episodes'])
2 | df.head()
```

```
1 | (12017, 3)
2 | [[9.24369748e-01 1.97866664e-01 0.00000000e+00]
3 |  [9.11164466e-01 7.82768603e-01 3.46725371e-02]
4 |  [9.09963986e-01 1.12683141e-01 2.75178866e-02]
5 |  ...
6 |  [3.85354142e-01 2.04161139e-04 1.65107320e-03]
7 |  [3.97358944e-01 1.60764569e-04 0.00000000e+00]
```

```

8      [4.54981993e-01 1.28217141e-04 0.00000000e+00]]
9      (12017, 3)
10     (12017, 3)
11         rating    members    episodes
12 12012  0.297719   0.000196  0.000000
13 12013  0.313325   0.000169  0.000000
14 12014  0.385354   0.000204  0.001651
15 12015  0.397359   0.000161  0.000000
16 12016  0.454982   0.000128  0.000000

```

```

1      .dataframe tbody tr th {
2          vertical-align: top;
3      }
4
5      .dataframe thead th {
6          text-align: right;
7      }

```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	0.000000	0.924370	0.197867
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	0.034673	0.911164	0.782769
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	0.027518	0.909964	0.112683
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	0.012658	0.900360	0.664323
4	9969	Gintama&#039;	Action, Comedy, Historical, Parody, Samurai, S...	TV	0.027518	0.899160	0.149180

```

1  def one_hot_encoding(df, cols):
2      for col, col_type in cols:
3          if col_type == 'list':
4              col_data = df[col].str.split(' ')
5              mlb = MultiLabelBinarizer()
6          else:
7              col_data = df[col]
8              mlb = LabelBinarizer()
9          mlb_data = mlb.fit_transform(col_data)
10         encoded_data = pd.DataFrame(mlb_data, columns=mlb.classes_, index=col_data.index)
11
12         for cls in mlb.classes_:
13             if cls in df.columns.values:
14                 df = df.rename(columns={cls: '{}_{}'.format(cls, col)})
15
16         print('Changing column: {} to: {}'.format(col, mlb.classes_))
17         df = df.drop(columns=[col])
18         df = pd.concat([df, encoded_data], axis=1)
19     return df

```

```

1  df = df.drop(columns=['type'])

```

```

1  # df = one_hot_encoding(df, [('type', str), ('genre', list)])
2  df = one_hot_encoding(df, [('genre', list)])
3  # df = df.drop(columns=['OVA'])
4  df.head()

```

```

1  Changing column: genre to: ['Action' 'Adventure' 'Cars' 'Comedy' 'Dementia' 'Demons' 'Drama' 'Ecchi'
2  'Fantasy' 'Game' 'Harem' 'Hentai' 'Historical' 'Horror' 'Josei' 'Kids'
3  'Magic' 'Martial Arts' 'Mecha' 'Military' 'Music' 'Mystery' 'Parody'
4  'Police' 'Psychological' 'Romance' 'Samurai' 'School' 'Sci-Fi' 'Seinen'
5  'Shoujo' 'Shoujo Ai' 'Shounen' 'Shounen Ai' 'Slice of Life' 'Space'
6  'Sports' 'Super Power' 'Supernatural' 'Thriller' 'Vampire' 'Yaoi' 'Yuri']

```

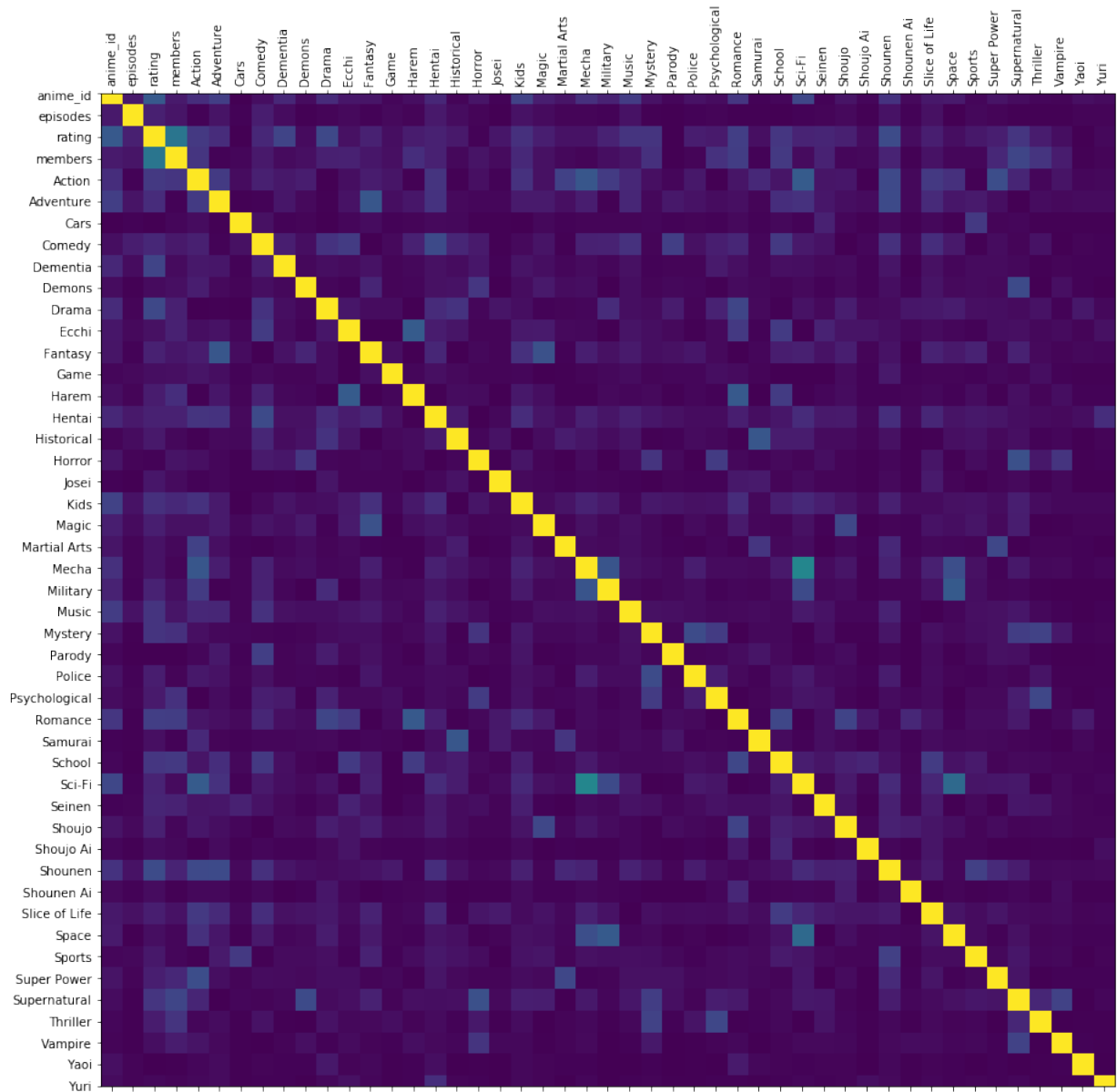
```
1 .dataframe tbody tr th {
2     vertical-align: top;
3 }
4
5 .dataframe thead th {
6     text-align: right;
7 }
```

	anime_id	name	episodes	rating	members	Action	Adventure	Cars	Comedy	Dementia	...	Shounen Ai	Slice of Life	Space
0	32281	Kimi no Na wa.	0.000000	0.924370	0.197867	0	0	0	0	0	...	0	0	0
1	5114	Fullmetal Alchemist: Brotherhood	0.034673	0.911164	0.782769	1	1	0	0	0	...	0	0	0
2	28977	Gintama°	0.027518	0.909964	0.112683	1	0	0	1	0	...	0	0	0
3	9253	Steins;Gate	0.012658	0.900360	0.664323	0	0	0	0	0	...	0	0	0
4	9969	Gintama&#039;	0.027518	0.899160	0.149180	1	0	0	1	0	...	0	0	0

5 rows × 48 columns

```
1 def plot_corr_matrix(df,size=15):
2     corr = df.corr().abs()
3     fig, ax = plt.subplots(figsize=(size, size))
4     ax.matshow(corr)
5     plt.xticks(range(len(corr.columns)), corr.columns)
6     plt.yticks(range(len(corr.columns)), corr.columns)
7     plt.xticks(rotation=90)
8     return corr

1 corr = plot_corr_matrix(df)
```



```

1 def drop_correlated(df, corr_matrix, threshold=0.5):
2     upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
3     to_drop = [column for column in upper.columns if any(upper[column] > 0.5)]
4     df = df.drop(columns=to_drop)
5     return df

```

```

1 df = drop_correlated(df, corr)
2 df.head()

```

```

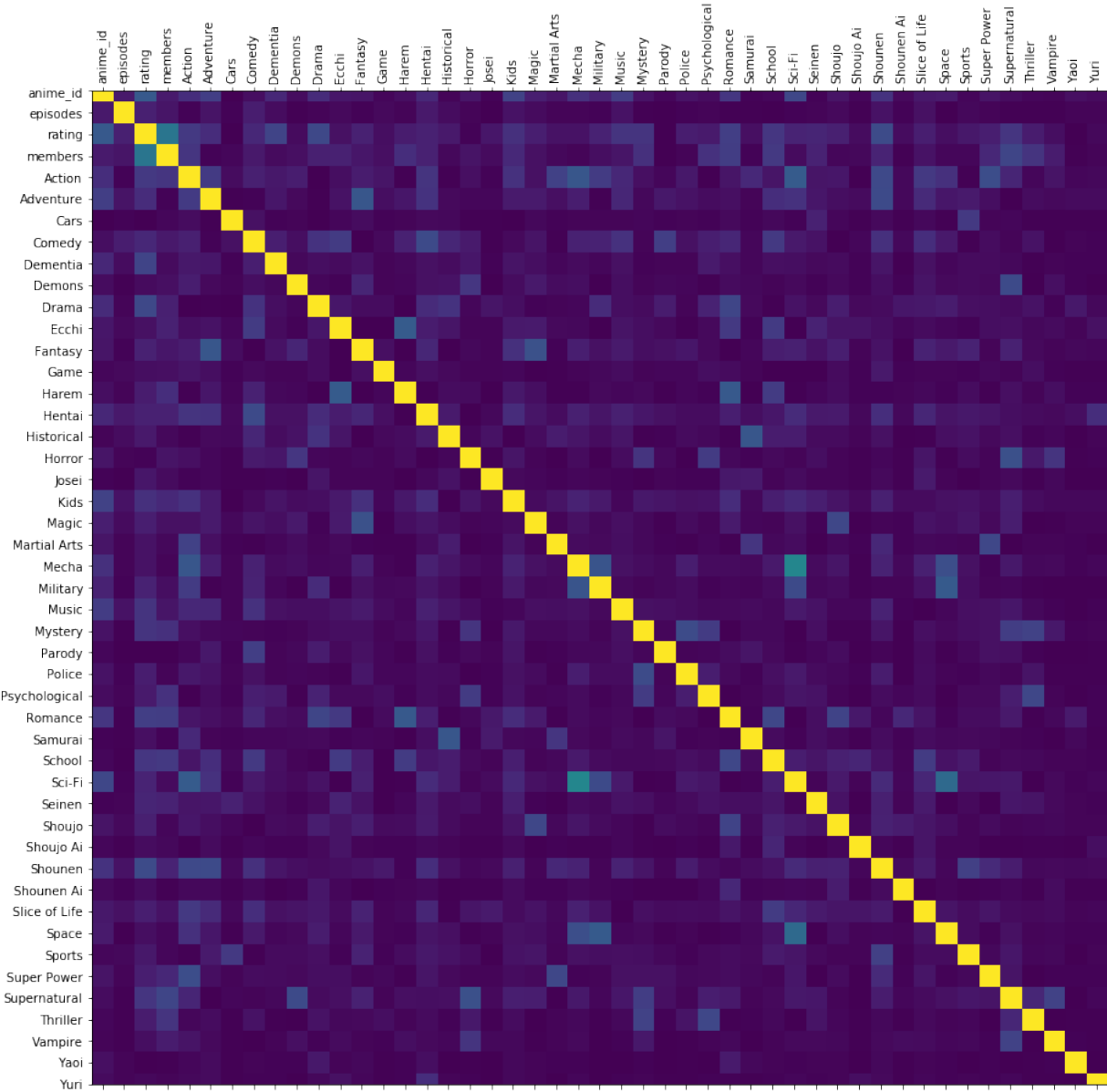
1 .dataframe tbody tr th {
2     vertical-align: top;
3 }
4
5 .dataframe thead th {
6     text-align: right;
7 }

```

	anime_id	name	episodes	rating	members	Action	Adventure	Cars	Comedy	Dementia	...	Shounen Ai	Slice of Life	Space
0	32281	Kimi no Na wa.	0.000000	0.924370	0.197867	0	0	0	0	0	...	0	0	0
1	5114	Fullmetal Alchemist: Brotherhood	0.034673	0.911164	0.782769	1	1	0	0	0	...	0	0	0
2	28977	Gintama°	0.027518	0.909964	0.112683	1	0	0	1	0	...	0	0	0
3	9253	Steins;Gate	0.012658	0.900360	0.664323	0	0	0	0	0	...	0	0	0
4	9969	Gintama&#039;	0.027518	0.899160	0.149180	1	0	0	1	0	...	0	0	0

5 rows × 48 columns

```
1 | corr = plot_corr_matrix(df)
```



```
1 | df['user_rating'] = pd.Series(np.zeros(df.shape[0]))
2 | df.head()
3 | df.tail()
```

```
1 | .dataframe tbody tr th {
2 |     vertical-align: top;
3 | }
4 |
5 | .dataframe thead th {
6 |     text-align: right;
7 | }
```

	anime_id	name	episodes	rating	members	Action	Adventure	Cars	Comedy	Dementia	...	Slice of Life	Space	Sports
12012	9316	Toushindai My Lover: Minami tai Mecha-Minami	0.000000	0.297719	0.000196	0	0	0	0	0	...	0	0	0
12013	5543	Under World	0.000000	0.313325	0.000169	0	0	0	0	0	...	0	0	0
12014	5621	Violence Gekiga David no Hoshi	0.001651	0.385354	0.000204	0	0	0	0	0	...	0	0	0
12015	6133	Violence Gekiga Shin David no Hoshi: Inma Dens...	0.000000	0.397359	0.000161	0	0	0	0	0	...	0	0	0
12016	26081	Yasuji no Pornorama: Yacchimae!!	0.000000	0.454982	0.000128	0	0	0	0	0	...	0	0	0

5 rows × 49 columns

```
1 | import xml.etree.ElementTree as ET
2 |
3 | tree = ET.parse('my_anime.xml')
4 | root = tree.getroot()
5 | anime_list = root.findall('anime')
6 |
7 | for anime in anime_list:
8 |     idx = anime.find('series_animedb_id')
9 |     title = anime.find('series_title')
10 |    score = anime.find('my_score')
11 |    list_has = df['anime_id'] == float(idx.text)
12 |    if not list_has.empty and float(score.text) != 0.0:
13 |        df.loc[list_has, 'user_rating'] = float(score.text)
14 |
15 | df.head()
```

```
1 | .dataframe tbody tr th {
2 |     vertical-align: top;
3 | }
4 |
5 | .dataframe thead th {
6 |     text-align: right;
7 | }
```



	anime_id	name	episodes	rating	members	Action	Adventure	Cars	Comedy	Dementia	...	Slice of Life	Space	Sports
0	32281	Kimi no Na wa.	0.000000	0.924370	0.197867	0	0	0	0	0	...	0	0	0
1	5114	Fullmetal Alchemist: Brotherhood	0.034673	0.911164	0.782769	1	1	0	0	0	...	0	0	0
2	28977	Gintama°	0.027518	0.909964	0.112683	1	0	0	1	0	...	0	0	0
3	9253	Steins;Gate	0.012658	0.900360	0.664323	0	0	0	0	0	...	0	0	0
4	9969	Gintama&#039;	0.027518	0.899160	0.149180	1	0	0	1	0	...	0	0	0

5 rows × 49 columns

```
1 df.corr().abs()[ 'user_rating' ][:].sort_values(ascending=False)
```

```
1 user_rating      1.000000
2 members          0.398669
3 rating           0.169579
4 School           0.124686
5 Romance           0.091002
6 Thriller          0.077548
7 Military          0.063860
8 Harem             0.061067
9 Ecchi             0.060635
10 Space            0.052915
11 Kids             0.050718
12 Seinen           0.049713
13 Sci-Fi           0.048391
14 Psychological    0.048230
15 Game             0.044487
16 Shounen          0.039878
17 Drama            0.038032
18 anime_id         0.036035
19 Comedy           0.035542
20 Historical        0.032808
21 Hentai           0.029237
22 Action           0.027912
23 Martial Arts     0.019517
24 Adventure         0.017990
25 Josei            0.017673
26 Music            0.016990
27 Samurai          0.014442
28 Demons           0.014098
29 Police           0.013589
30 Mystery          0.012704
31 Slice of Life     0.012673
32 Supernatural      0.011336
33 Dementia          0.010951
34 Cars             0.010110
35 Shounen Ai       0.009378
36 Sports           0.008986
37 Shoujo Ai        0.008749
38 Super Power      0.008040
39 Yuri             0.007620
40 Magic            0.007353
41 Yaoi             0.007335
42 Shoujo           0.005842
43 episodes         0.004312
44 Parody           0.003515
45 Vampire          0.002828
46 Fantasy          0.002147
47 Mecha            0.001743
48 Horror           0.001663
49 Name: user_rating, dtype: float64
```

```
1 pred_val = 'user_rating'
2 c = list(df.columns)
3 c.remove('anime_id')
```

```

4 c.remove('name')
5 c.remove(pred_val)
6
7 e = df[pred_val] != 0.0
8 X_train = pd.DataFrame(df[e][c])
9 Y_train = pd.DataFrame(df[e][pred_val])
10 print(X_train.shape)
11
12 e = df[pred_val] == 0.0
13 X = pd.DataFrame(df[e][c])
14 print(X.shape)

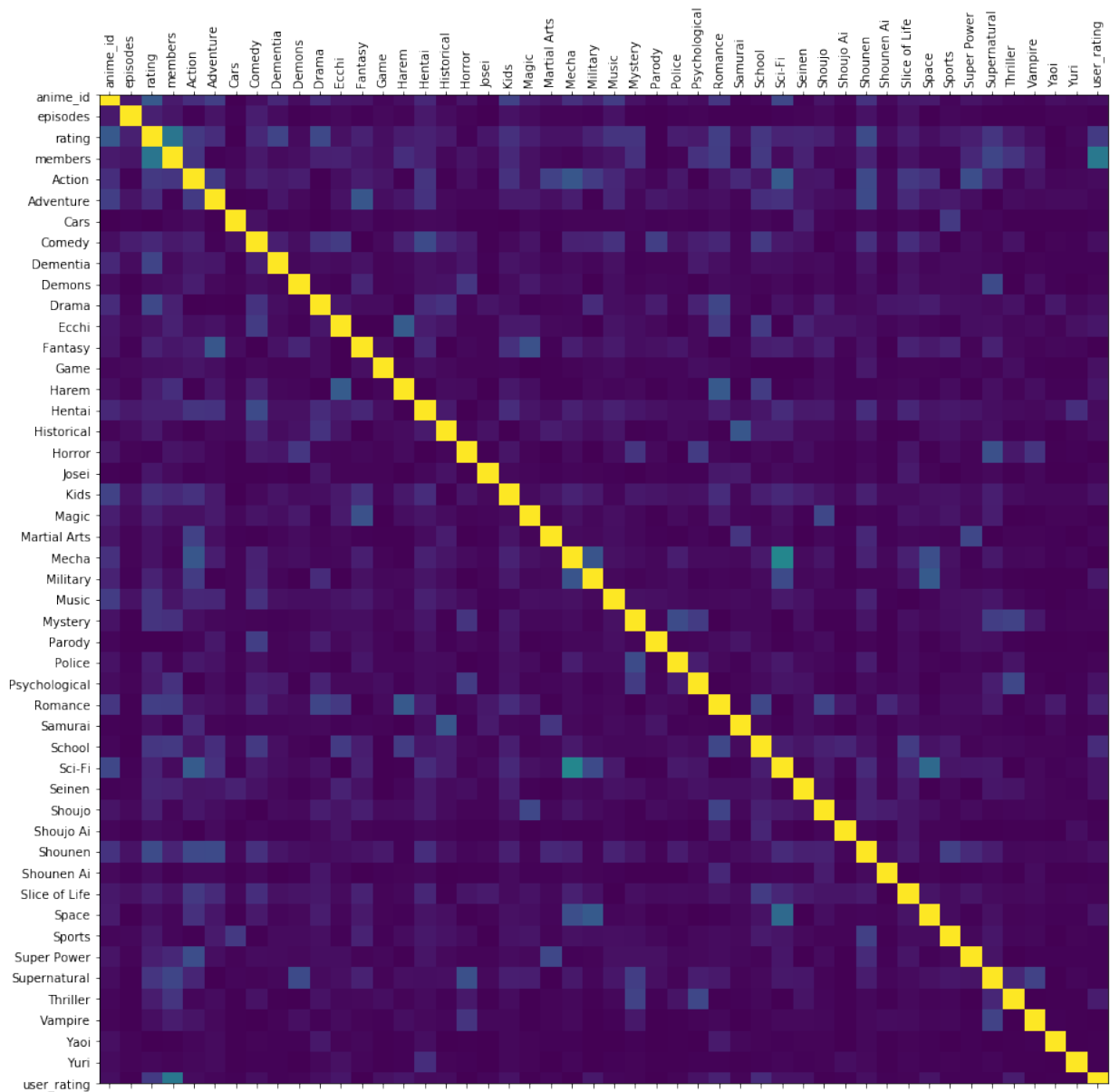
```

```

1 (225, 46)
2 (11792, 46)

```

```
1 corr = plot_corr_matrix(df)
```



```

1 from sklearn.model_selection import KFold
2 from sklearn.metrics import mean_squared_error, mean_absolute_error
3
4 from sklearn.linear_model import LinearRegression, BayesianRidge, LogisticRegression
5 from sklearn.neural_network import MLPClassifier
6
7 from sklearn.ensemble import RandomForestRegressor
8 from sklearn.ensemble import GradientBoostingRegressor

```

```

1 def calc_scores(model, X, Y, test=None):
2     if test is not None:
3         Xr = X.iloc[test,:]
4         Yr = Y.iloc[test,:]
5     else:
6         Xr = X
7         Yr = Y
8     Y_pred = model.predict(Xr)
9     score_r2 = model.score(Xr, Yr.values.ravel())
10    score_rmse = mean_squared_error(Yr, Y_pred)
11    score_abs = mean_absolute_error(Yr, Y_pred)
12    return (score_r2, score_rmse, score_abs)

```

```

1 def fit_predict(model, X_train, Y_train, X, rst=21, print_size=15):
2     scores_1 = []
3     scores_2 = []
4     scores_3 = []
5
6     kfold = KFold(n_splits=5, shuffle=True, random_state=rst)
7     for i, (train, test) in enumerate(kfold.split(X_train, Y_train)):
8         model.fit(X_train.iloc[train,:], Y_train.iloc[train,:].values.ravel())
9         scores = calc_scores(model, X_train, Y_train, test)
10        scores_1.append(scores[0])
11        scores_2.append(scores[1])
12        scores_3.append(scores[2])
13
14    model.fit(X_train, Y_train.values.ravel())
15
16    scores = calc_scores(model, X_train, Y_train)
17    scores_1.append(scores[0])
18    scores_2.append(scores[1])
19    scores_3.append(scores[2])
20
21    x = np.arange(0, 6, 1)
22    plt.plot(x, scores_1)
23    plt.plot(x, scores_2)
24    plt.plot(x, scores_3)
25    labels = ['R2', 'RMSE', 'MAE']
26    plt.legend(labels)
27    plt.show()
28
29    # Условие по которому вы выбираете TEST выборку
30    # это может быть просто df.iloc[test,:] (среэ)
31    e = df[pred_val] == 0.0
32
33    Y = pd.DataFrame(df[e])
34    Y[pred_val] = model.predict(X)
35    print(Y.sort_values(by=[pred_val], ascending=False)[['name', pred_val]].head(print_size))

```

```

1 models = [
2     BayesianRidge(),
3     LinearRegression(),
4     RandomForestRegressor(),
5     GradientBoostingRegressor(),
6     # Classification methods
7     # LogisticRegression(),
8     # MLPClassifier(),
9 ]

```

```

1 def solve(models, X_train, Y_train, X):
2     for model in models:
3         print('Model:', type(model).__name__)
4         fit_predict(model, X_train, Y_train, X)
5         print('\n\n\n')

```

```

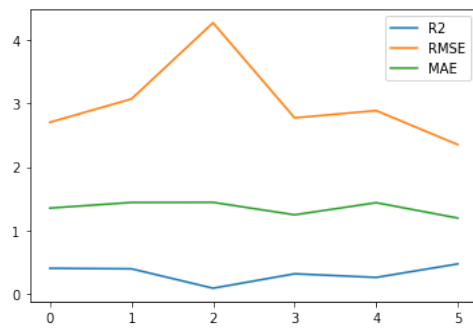
1 solve(models, X_train, Y_train, X)

```

```

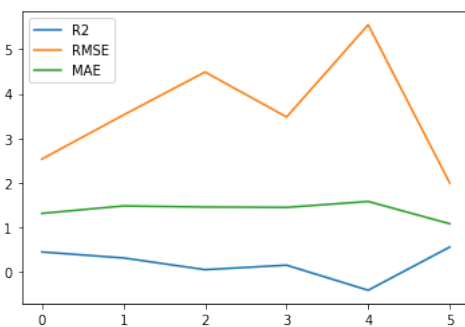
1 Model: BayesianRidge

```



	name	user_rating
38	Monster	9.921192
1235	Metropolis	9.287855
493	Higashi no Eden	8.956619
365	Paprika	8.824674
114	Sakamichi no Apollon	8.731949
832	Ima, Soko ni Iru Boku	8.625008
5295	Sanctuary	8.589349
172	Hachimitsu to Clover II	8.578789
132	Toki wo Kakeru Shoujo	8.560349
35	Howl no Ugoku Shiro	8.548517
238	Gankutsuou	8.514186
702	Another	8.499337
981	Mousou Dairinin	8.462670
2901	Blue Gender	8.437504
2365	Giniro no Kami no Agito	8.386215

1 Model: LinearRegression



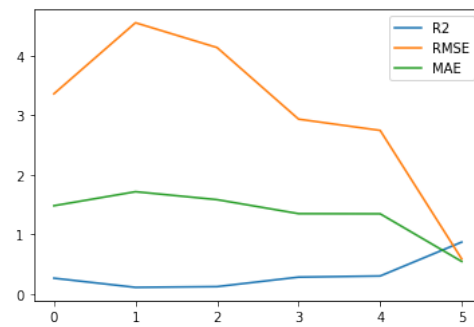
	name	user_rating
38	Monster	12.283960
172	Hachimitsu to Clover II	10.572720
1235	Metropolis	10.544546
114	Sakamichi no Apollon	10.361105
365	Paprika	10.340557
325	Hachimitsu to Clover	10.086614
596	Psycho-Pass Movie	9.805784
68	Shouwa Genroku Rakugo Shinjuu	9.743177
493	Higashi no Eden	9.624178
91	Shinsekai yori	9.561978
75	Ghost in the Shell: Stand Alone Complex 2nd GIG	9.502509
137	Detective Conan Movie 06: The Phantom of Baker...	9.491569
633	Paradise Kiss	9.471116
601	Vampire Hunter D (2000)	9.439421
93	Chihayafuru 2	9.426317

1 Model: RandomForestRegressor

```

1 /Users/snipghost/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n_estimators will
  change from 10 in version 0.20 to 100 in 0.22.
2 "10 in version 0.20 to 100 in 0.22.", FutureWarning)

```

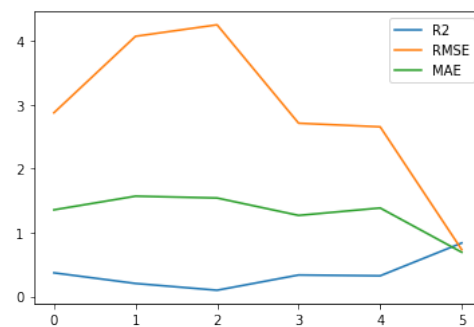


		name	user_rating
2	50	Yojouhan Shinwa Taikei	9.3
3	21	Rurouni Kenshin: Meiji Kenkaku Romantan - Tsui...	9.2
4	26	Monogatari Series: Second Season	9.2
5	10	Clannad: After Story	9.1
6	10739	Yakusoku: Africa Mizu to Midori	9.1
7	60	Hotarubi no Mori e	9.1
8	9046	Kahei no Umi	9.1
9	8	Gintama Movie: Kanketsu-hen - Yorozuya yo Eien...	9.0
10	45	Kara no Kyoukai 5: Mujun Rasen	9.0
11	19	Code Geass: Hangyaku no Lelouch	9.0
12	38	Monster	9.0
13	6	Hunter x Hunter (2011)	9.0
14	9	Gintama#039;: Enchousen	9.0
15	13	Code Geass: Hangyaku no Lelouch R2	8.9
16	53	Rainbow: Nisha Rokubou no Shichinin	8.9

```

1 Model: GradientBoostingRegressor

```



		name	user_rating
2	45	Kara no Kyoukai 5: Mujun Rasen	9.875052
3	21	Rurouni Kenshin: Meiji Kenkaku Romantan - Tsui...	9.663214
4	53	Rainbow: Nisha Rokubou no Shichinin	9.337465
5	60	Hotarubi no Mori e	9.232013
6	38	Monster	9.226272
7	36	Fate/Zero 2nd Season	9.157793
8	68	Shouwa Genroku Rakugo Shinjuu	9.060116
9	93	Chihayafuru 2	8.926881
10	9538	Mirai ni Mukete: Bousai wo Kangaeru	8.876312
11	77	Kara no Kyoukai 7: Satsujin Kousatsu (Kou)	8.825971
12	10	Clannad: After Story	8.798712
13	75	Ghost in the Shell: Stand Alone Complex 2nd GIG	8.744925
14	35	Howl no Ugoku Shiro	8.721789
15	50	Yojouhan Shinwa Taikei	8.668339
16	269	Mobile Suit Gundam 00	8.657001

## Выводы:

Эмпирически выявлено, что лучший результат дают регрессионные модели. Задача оказалось довольно простой и предобработка данных почти не влияет на результат, для получения высокой точности предсказаний достаточно было провести one-hot-encoding жанров, преобразовать их в тэги (может быть несколько на каждый сериал), и нормализовать рейтинг.