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»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчёт по лабораторной работе
по дисциплине «Методы Машинного Обучения»

Выполнил:
студент группы № ИУ5-21М
Кучеренко Михаил Александрович
_____________, _________

Проверил:
к.т.н., доц., Ю.Е. Гапанюк
_____________, _________

2021 г.

Лабораторная работа №3

Обработка признаков (часть 2).

Рассмотрим исторические данные по выходу и продажам видеоигр из на 2019 год

Задание:

Оригинал:

- Выбрать один или несколько наборов данных (датасетов) для решения следующих задач. Каждая задача может быть решена на отдельном датасете, или несколько задач могут быть решены на одном датасете. Просьба не использовать датасет, на котором данная задача решалась в лекции.
- Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
  - масштабирование признаков (не менее чем тремя способами);
  - обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
  - обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);
  - отбор признаков:
    - один метод из группы методов фильтрации (filter methods);
    - один метод из группы методов обертывания (wrapper methods);
    - один метод из группы методов вложений (embedded methods).

```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MaxAbsScaler
```

```
save_path = 'video_games_s2.csv'
df = pd.read_csv(save_path)
print(f'Loaded {len(df)} games')
```

```
Loaded 219 games
```

```
df.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Rank	Name	Genre	ESRB_Rating	Platform	Publisher	Developer	Critic_Score	User_Score	Year	Sales	Critic
0	3	Mario Kart Wii	9	3	13	27	83	8.2	9.1	2008.0	37.14	2.056
1	5	Wii Sports Resort	13	3	13	27	83	8.0	8.8	2009.0	33.09	2.032
2	7	New Super Mario Bros.	7	3	1	27	83	9.1	8.1	2006.0	30.80	2.155
3	9	New Super Mario Bros. Wii	7	3	13	27	83	8.6	9.2	2009.0	30.22	2.102
4	12	Wii Play	5	3	13	27	83	5.9	4.5	2007.0	28.02	1.741

```
# Построение плотности распределения
def draw_kde(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(
        ncols=2, figsize=(12, 5))
    # первый график
    ax1.set_title(label1)
    for col in col_list:
        sns.distplot(df1[col], ax=ax1)
    # второй график
    ax2.set_title(label2)
    for col in col_list:
        sns.distplot(df2[col], ax=ax2)
    plt.show()
```

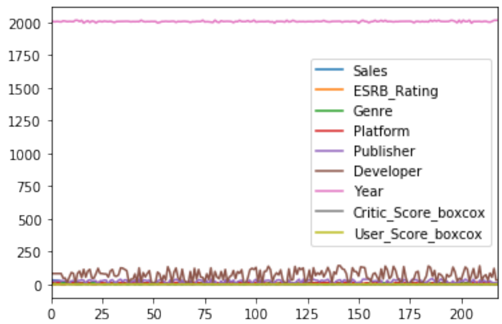
```
def draw_data(col_list, df, df_scaled):
    df[col_list].plot()
    df_scaled[col_list].plot()
    plt.show()
```

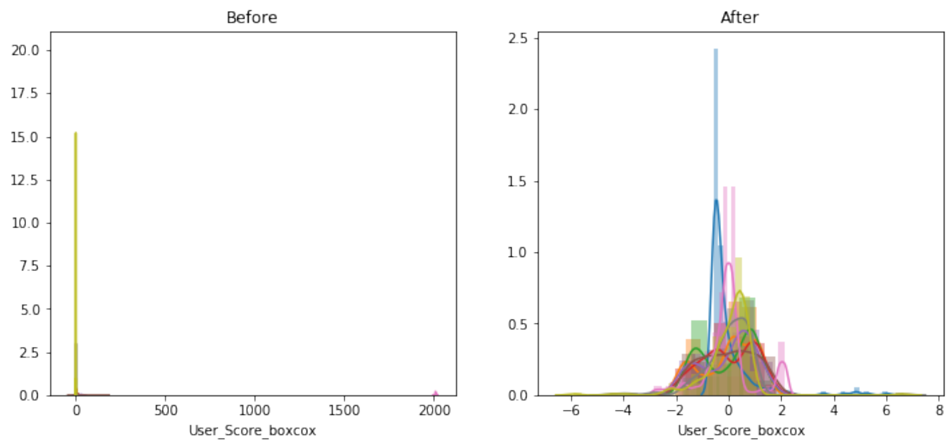
```
def get_scaled(df, columns, scaler=StandardScaler()):
    data_scaled = scaler.fit_transform(df[columns])
    df_scaled = pd.DataFrame(data_scaled, columns=columns)
    draw_data(columns, df, df_scaled)
    draw_kde(columns, df, df_scaled, 'Before', 'After')
    return df_scaled
```

```
def apply_scaled(df, df_scaled, columns):
    for col in columns:
        df[f'{col}_scaled'] = df_scaled[col]
    return df
```

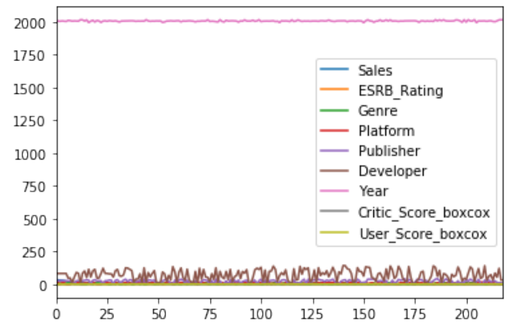
```
columns = ['Sales', 'ESRB_Rating', 'Genre', 'Platform', 'Publisher', 'Developer', 'Year', 'Critic_Score_boxcox', 'User_Score_boxcox']
```

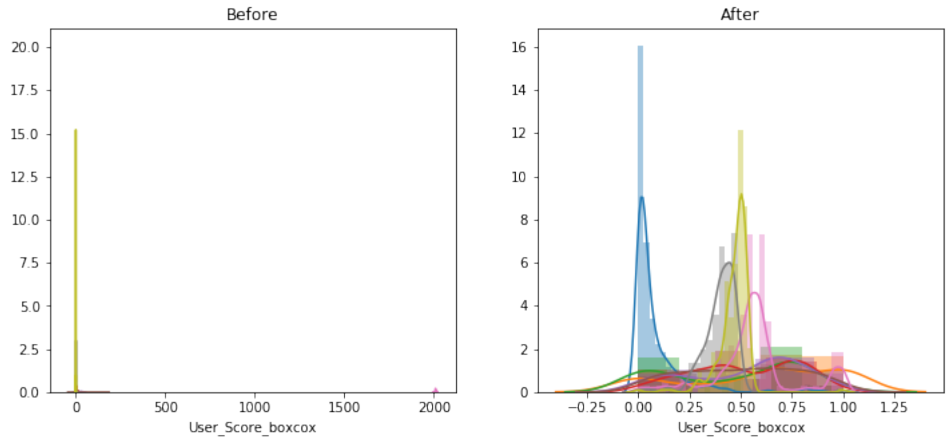
```
df_scaled_std = get_scaled(df, columns, StandardScaler())
```



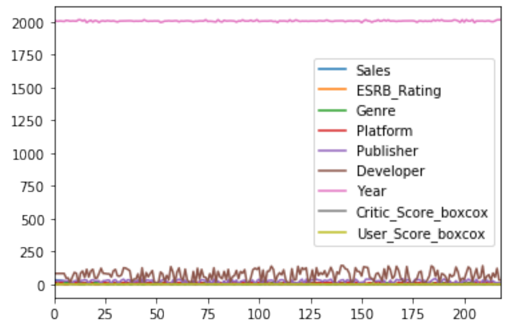


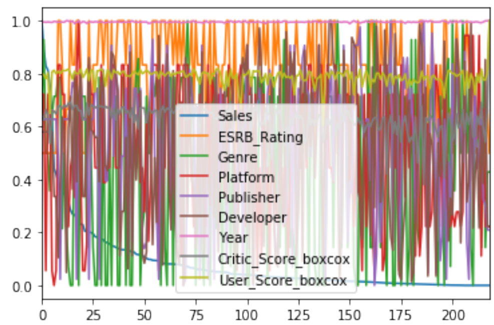

```
df_scaled_maxmin = get_scaled(df, columns, MinMaxScaler())
```

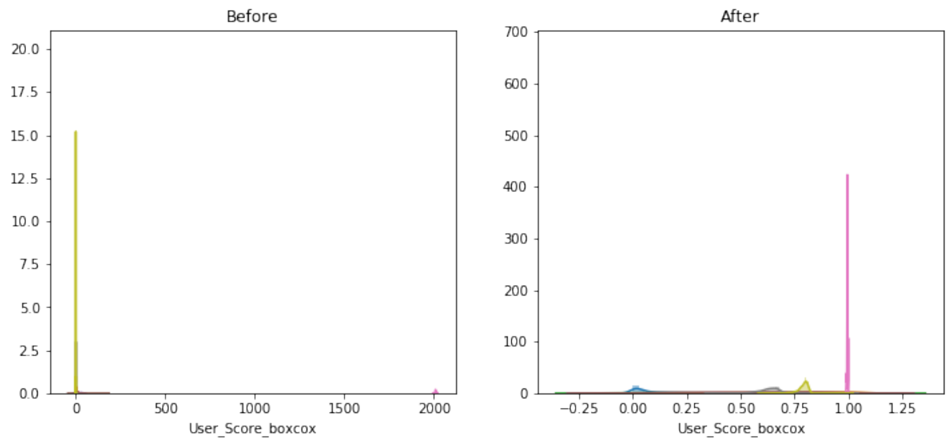





```
df_scaled_maxabs = get_scaled(df, columns, MaxAbsScaler())
```








```
df = apply_scaled(df, df_scaled_maxmin, columns)
df.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Rank	Name	Genre	ESRB_Rating	Platform	Publisher	Developer	Critic_Score	User_Score	Year	...	User_Score_boxcox	Sales_sc
0	3	Mario Kart Wii	9	3	13	27	83	8.2	9.1	2008.0	...	1.218568	1.00000
1	5	Wii Sports Resort	13	3	13	27	83	8.0	8.8	2009.0	...	1.209662	0.89095
2	7	New Super Mario Bros.	7	3	1	27	83	9.1	8.1	2006.0	...	1.186848	0.82929
3	9	New Super Mario Bros. Wii	7	3	13	27	83	8.6	9.2	2009.0	...	1.221433	0.81367
4	12	Wii Play	5	3	13	27	83	5.9	4.5	2007.0	...	0.987667	0.75444

5 rows × 22 columns

```
df.tail(2)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Rank	Name	Genre	ESRB_Rating	Platform	Publisher	Developer	Critic_Score	User_Score	Year	...	User_Score_boxcox	S
217	55424	Thumper	6	4	4	9	32	9.0	9.3	2017.0	...	1.224249	0
218	66666	FakeGame	6	3	4	9	32	29.0	64.0	2018.0	...	1.519776	0

2 rows × 22 columns

Поиск и устранение выбросов

```python
from enum import Enum
import scipy.stats as stats
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

```python
class OutlierBoundaryType(Enum):
    SIGMA = 1
    QUANTILE = 2
    IRQ = 3

# Функция вычисления верхней и нижней границы выбросов
```

```
def get_outlier_boundaries(df, col, outlier_boundary_type: OutlierBoundaryType):
    if outlier_boundary_type == OutlierBoundaryType.SIGMA:
        K1 = 3
        lower_boundary = df[col].mean() - (K1 * df[col].std())
        upper_boundary = df[col].mean() + (K1 * df[col].std())

    elif outlier_boundary_type == OutlierBoundaryType.QUANTILE:
        lower_boundary = df[col].quantile(0.05)
        upper_boundary = df[col].quantile(0.95)

    elif outlier_boundary_type == OutlierBoundaryType.IRQ:
        K2 = 1.5
        IQR = df[col].quantile(0.75) - df[col].quantile(0.25)
        lower_boundary = df[col].quantile(0.25) - (K2 * IQR)
        upper_boundary = df[col].quantile(0.75) + (K2 * IQR)

    else:
        raise NameError('Unknown Outlier Boundary Type')

    return lower_boundary, upper_boundary
```

```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # ящик с усами
    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])
    # ящик с усами
    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

```
scaled_columns = [f'{x}_scaled' for x in columns]
```

```
method_list = ['Original']
df_changed = [df]
```

```
for obt in OutlierBoundaryType:

    df2 = df.copy()

    for col in columns:
        # Вычисление верхней и нижней границы
        lower_boundary, upper_boundary = get_outlier_boundaries(df2, col, obt)
        # Изменение данных
        df2[col] = np.where(df2[col] > upper_boundary, upper_boundary,
                            np.where(df2[col] < lower_boundary, lower_boundary, df2[col]))

    for col in scaled_columns:
        # Вычисление верхней и нижней границы
        lower_boundary, upper_boundary = get_outlier_boundaries(df2, col, obt)
        # Изменение данных
        df2[col] = np.where(df2[col] > upper_boundary, upper_boundary,
                            np.where(df2[col] < lower_boundary, lower_boundary, df2[col]))

    title = '{}-updated'.format(obt)
    # Сохранение в списки
    method_list.append(title)
    df_changed.append(df2)
```

```
df_changed[1].tail(2)
```

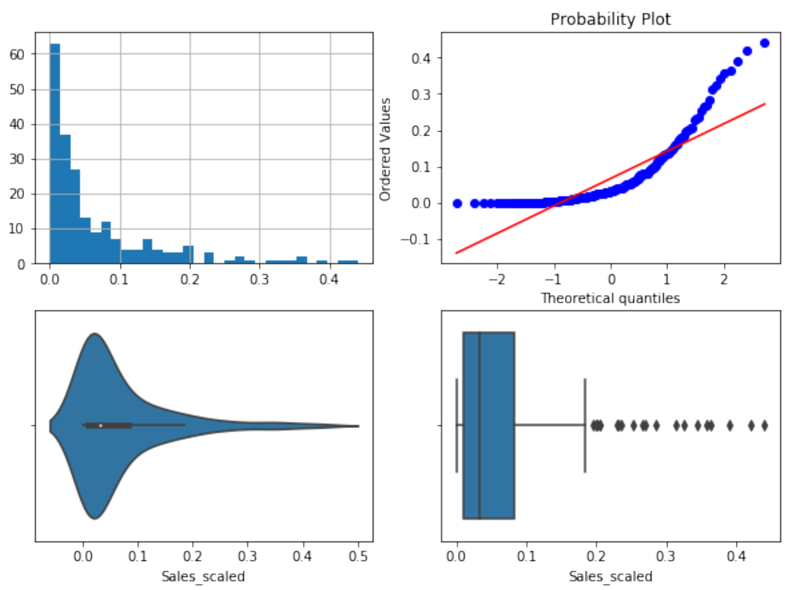
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Rank	Name	Genre	ESRB_Rating	Platform	Publisher	Developer	Critic_Score	User_Score	Year	...	User_Score_boxcox	S
217	55424	Thumper	6.0	4.0	4.0	9.0	32.0	9.0	9.3	2017.0	...	1.224249	0
218	66666	FakeGame	6.0	3.0	4.0	9.0	32.0	29.0	64.0	2018.0	...	1.344304	0

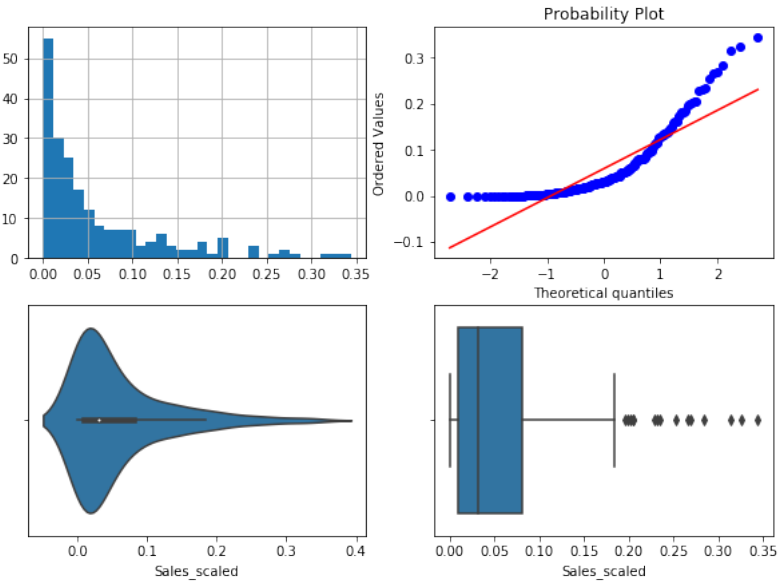
2 rows × 22 columns

```
for col in scaled_columns:
    for obt in OutlierBoundaryType:
        # Вычисление верхней и нижней границы
        lower_boundary, upper_boundary = get_outlier_boundaries(df, col, obt)
        # Флаги для удаления выбросов
        outliers_temp = np.where(df[col] > upper_boundary, True,
                                 np.where(df[col] < lower_boundary, True, False))
        # Удаление данных на основе флага
        df_trimmed = df.loc[~(outliers_temp), ]
        title = 'Поле-{}, метод-{}, строк-{}'.format(col, obt, df_trimmed.shape[0])
        diagnostic_plots(df_trimmed, col, title)
```

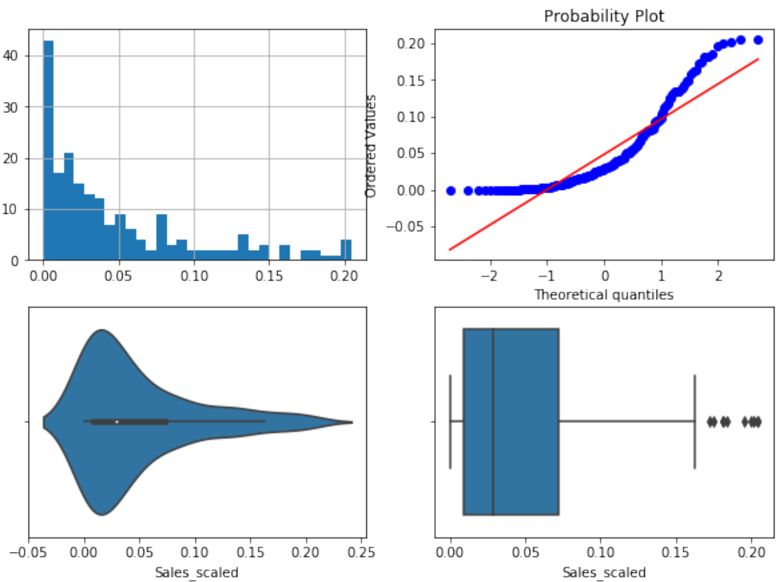
Поле-Sales_scaled, метод-OutlierBoundaryType.SIGMA, строк-213



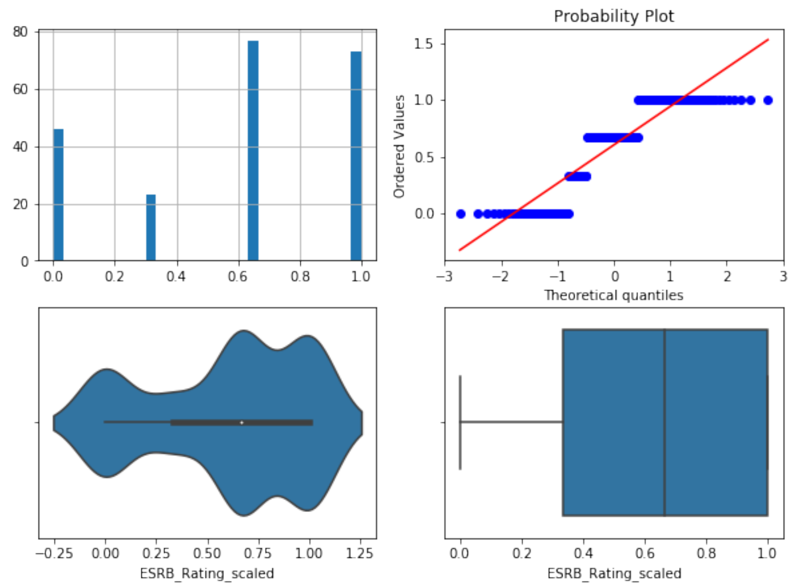
Поле-Sales_scaled, метод-OutlierBoundaryType.QUANTILE, строк-208



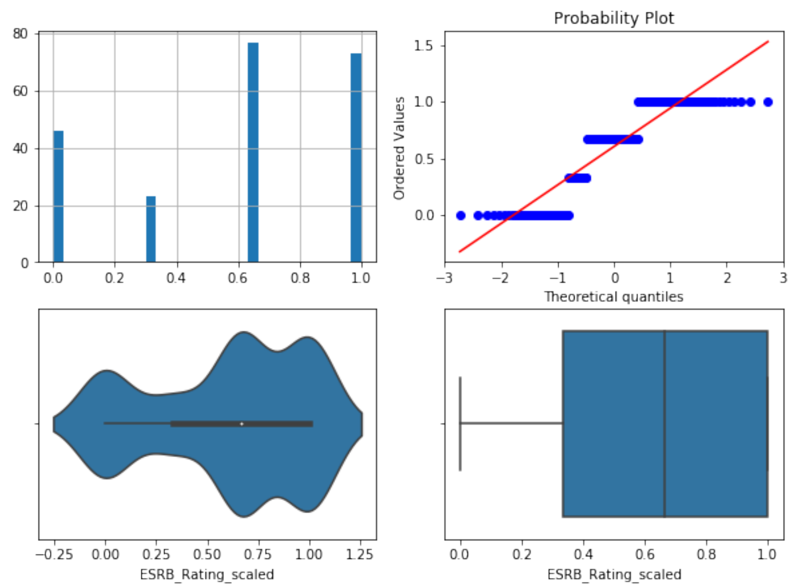
Поле-Sales_scaled, метод-OutlierBoundaryType.IRQ, строк-198



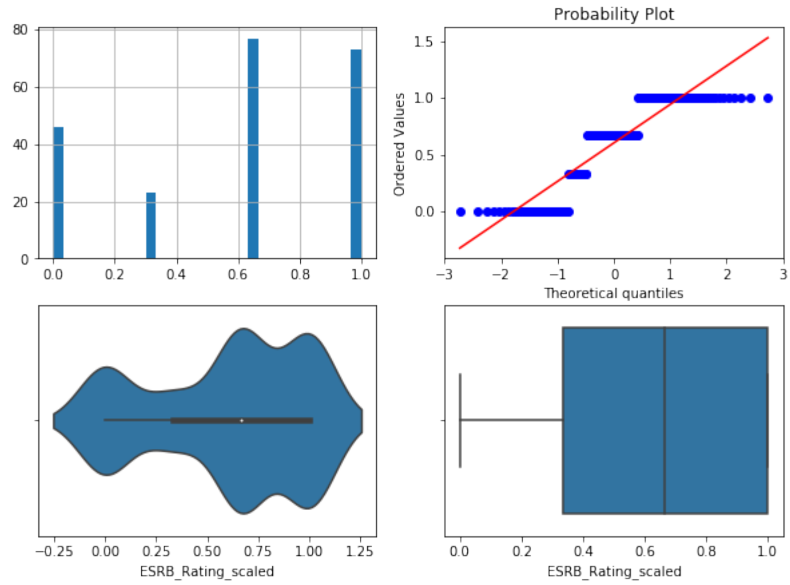
Поле-ESRB_Rating_scaled, метод-OutlierBoundaryType.SIGMA, строк-219



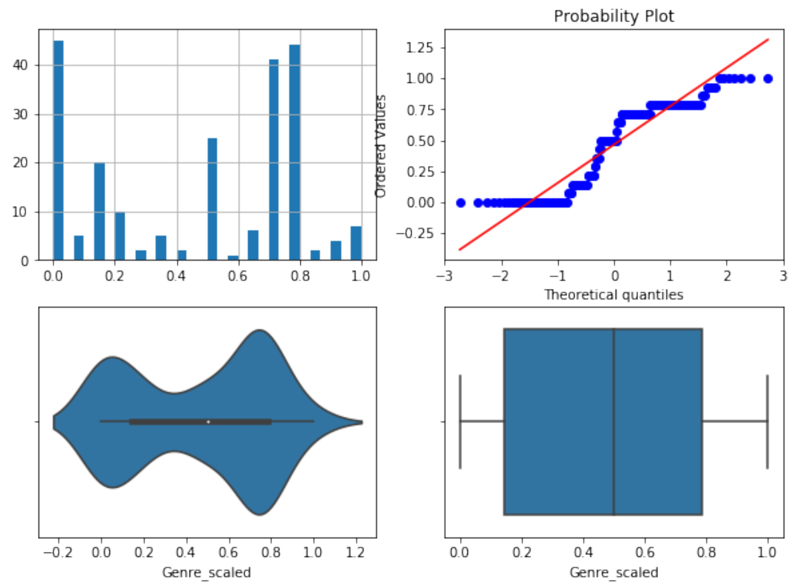
Поле-ESRB_Rating_scaled, метод-OutlierBoundaryType.QUANTILE, строк-219



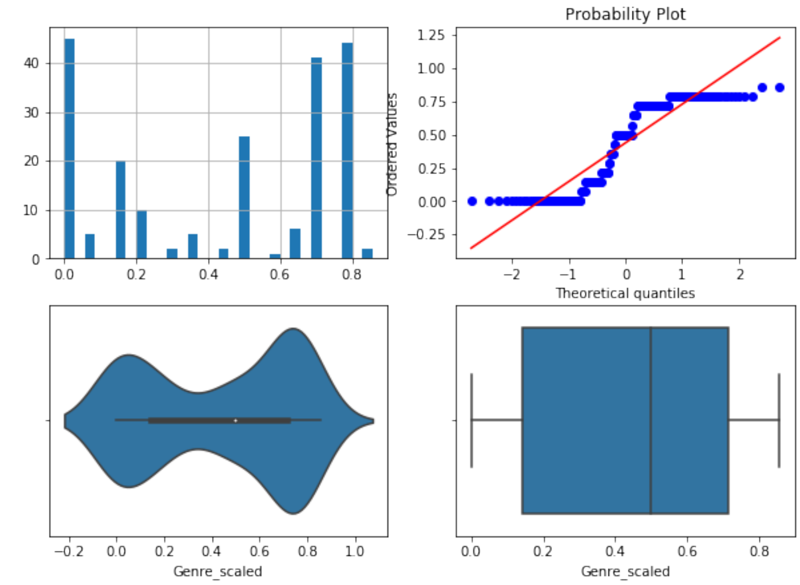
Поле-ESRB_Rating_scaled, метод-OutlierBoundaryType.IRQ, строк-219



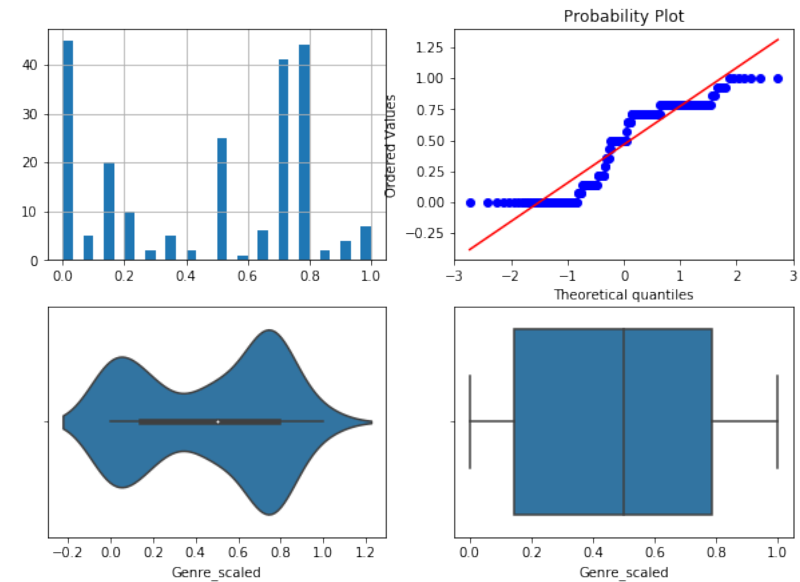
Поле-Genre_scaled, метод-OutlierBoundaryType.SIGMA, строк-219



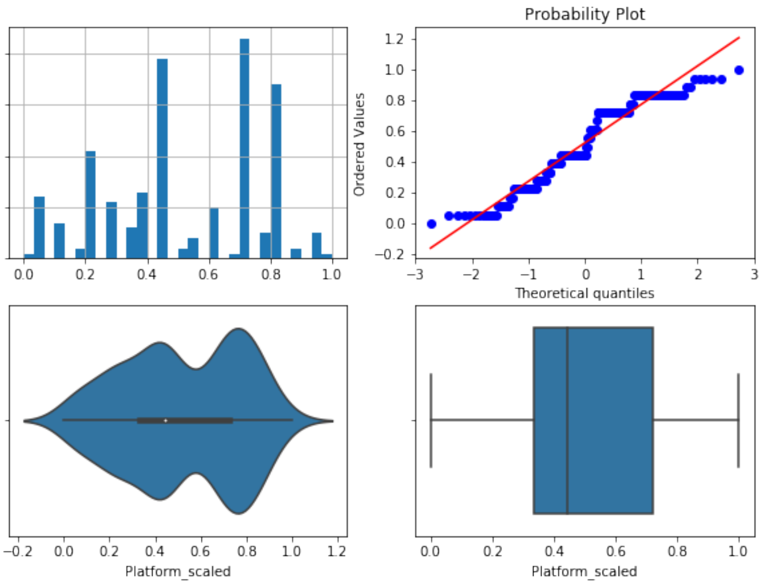
Поле-Genre_scaled, метод-OutlierBoundaryType.QUANTILE, строк-208



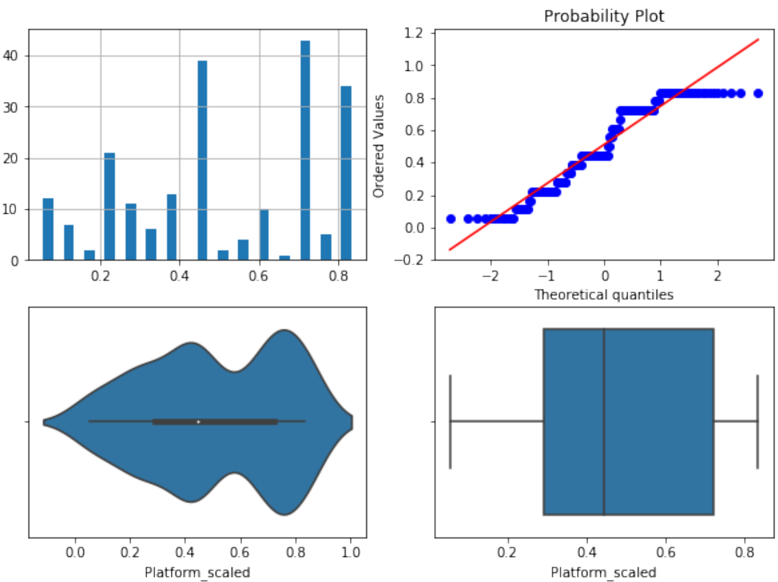
Поле-Genre_scaled, метод-OutlierBoundaryType.IRQ, строк-219



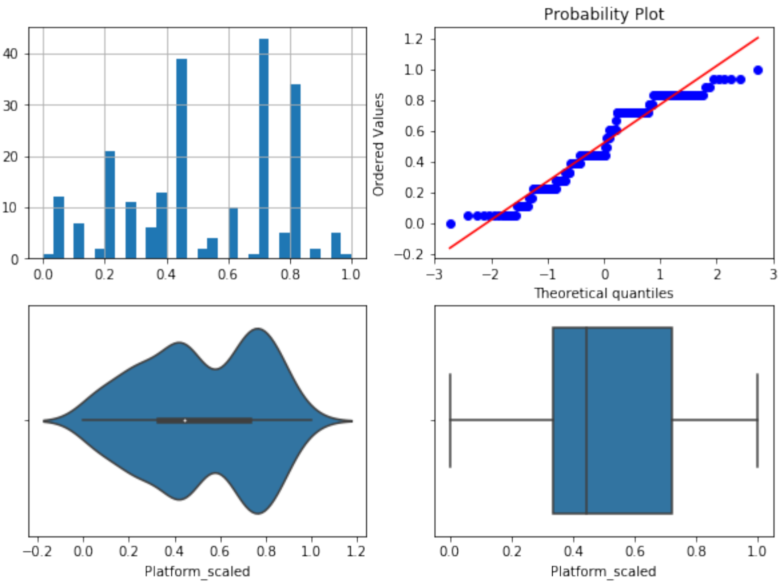
Поле-Platform_scaled, метод-OutlierBoundaryType.SIGMA, строк-219



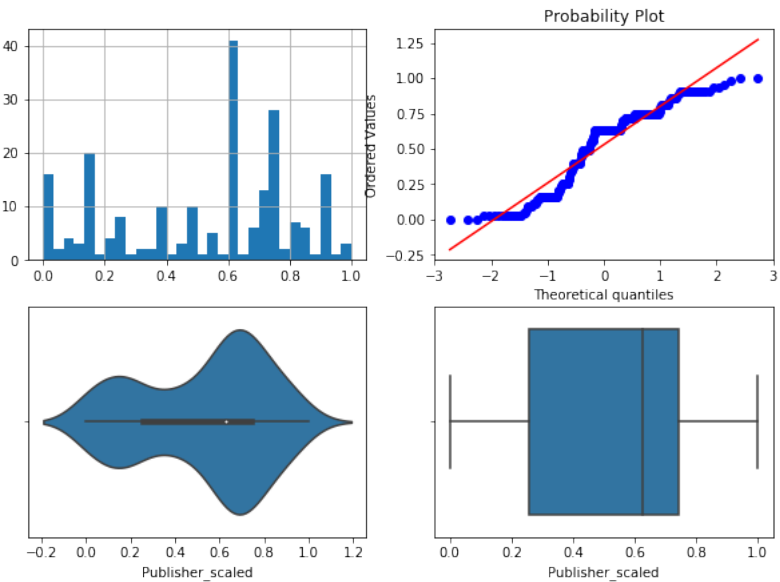
Поле-Platform_scaled, метод-OutlierBoundaryType.QUANTILE, строк-210



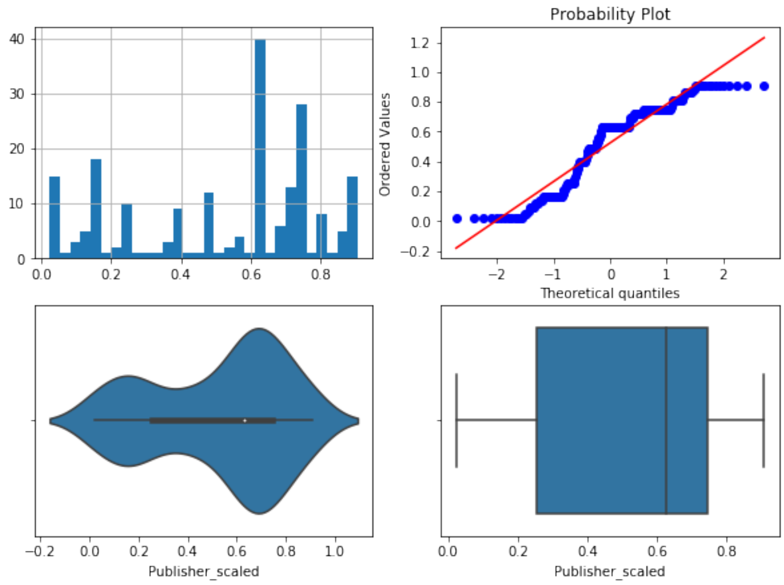
Поле-Platform_scaled, метод-OutlierBoundaryType.IRQ, строк-219



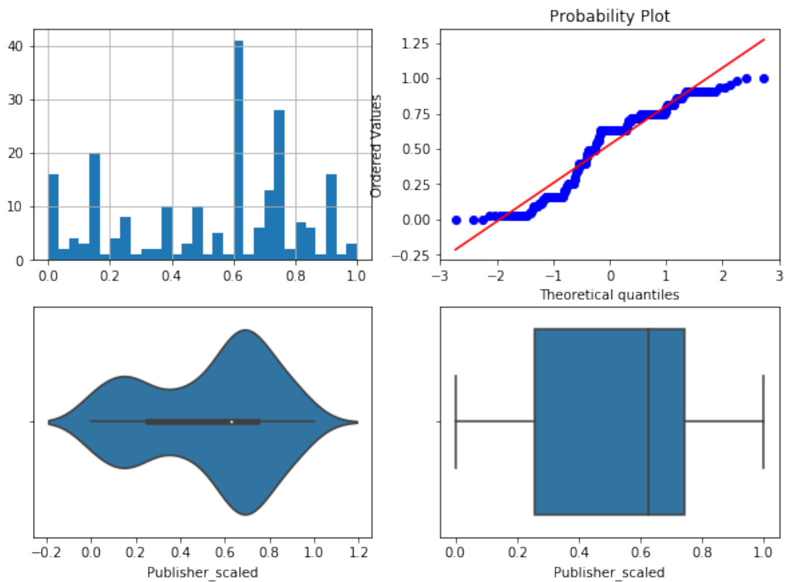
Поле-Publisher_scaled, метод-OutlierBoundaryType.SIGMA, строк-219



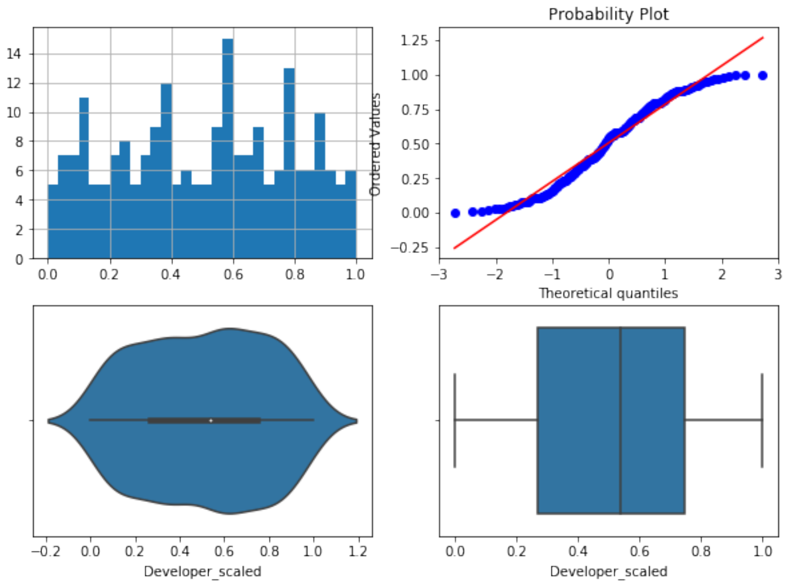
Поле-Publisher_scaled, метод-OutlierBoundaryType.QUANTILE, строк-210



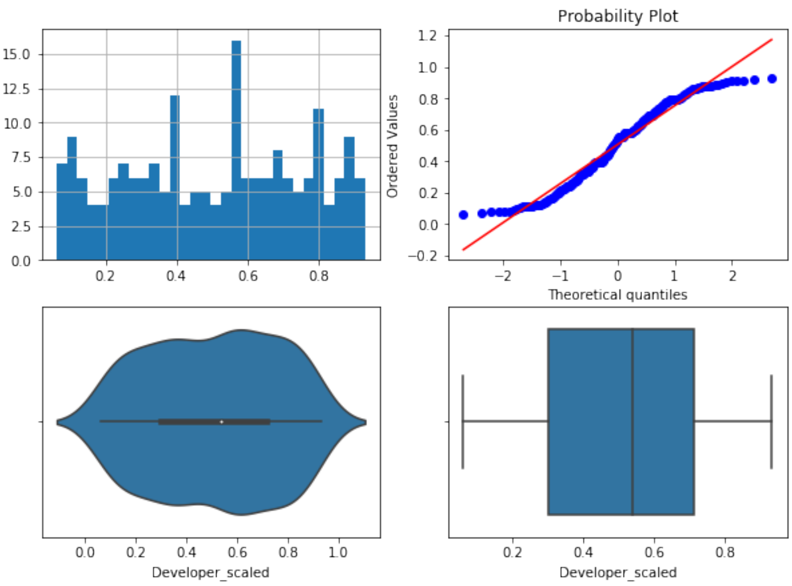
Поле-Publisher_scaled, метод-OutlierBoundaryType.IRQ, строк-219



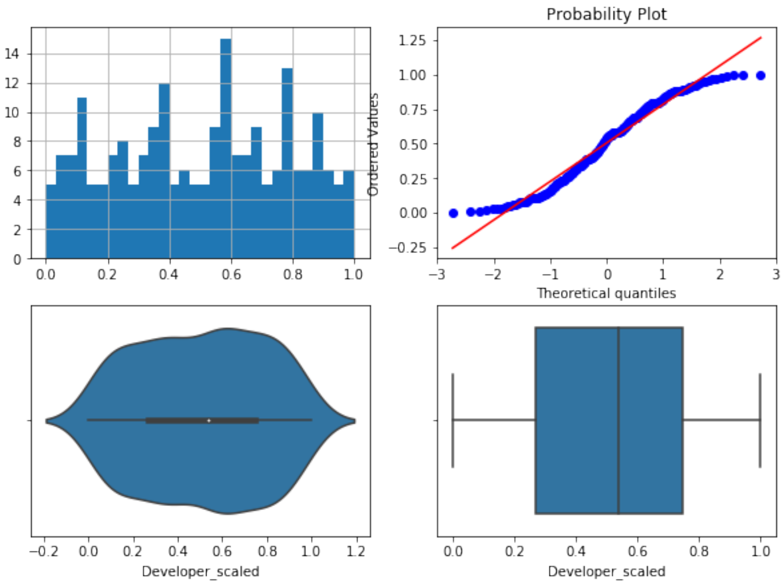
Поле-Developer_scaled, метод-OutlierBoundaryType.SIGMA, строк-219



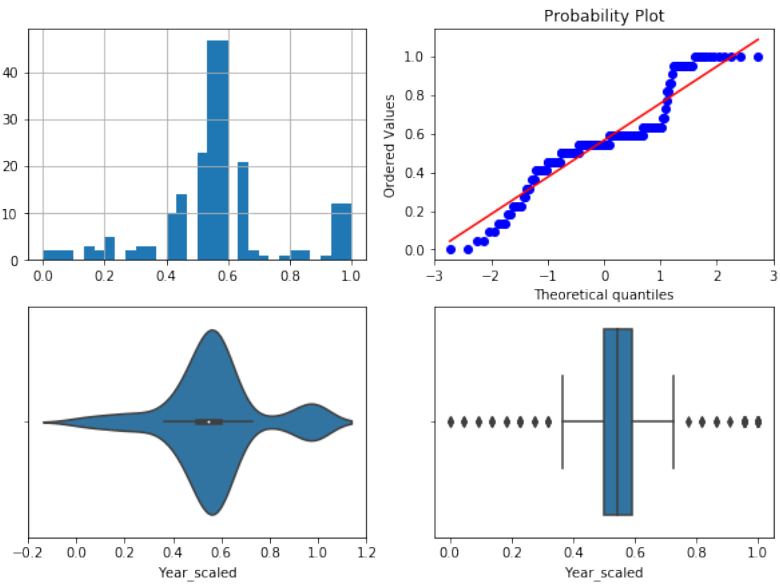
Поле-Developer_scaled, метод-OutlierBoundaryType.QUANTILE, строк-197



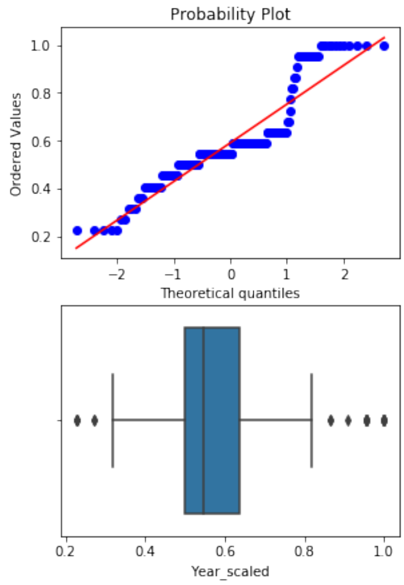
Поле-Developer_scaled, метод-OutlierBoundaryType.IRQ, строк-219



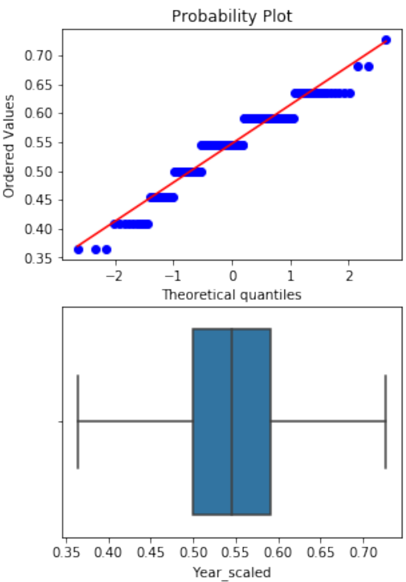
Поле-Year_scaled, метод-OutlierBoundaryType.SIGMA, строк-219



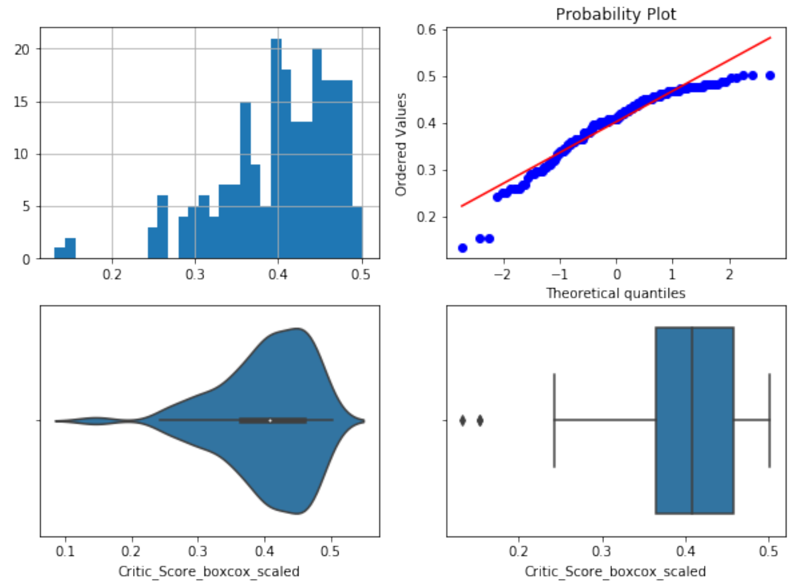
Поле-Year_scaled, метод-OutlierBoundaryType.QUANTILE, строк-208



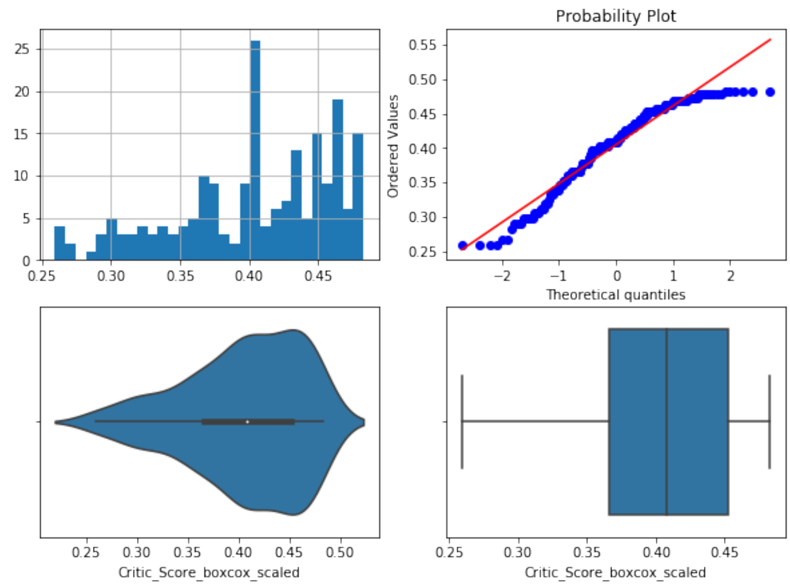
Поле-Year_scaled, метод-OutlierBoundaryType.IRQ, строк-168



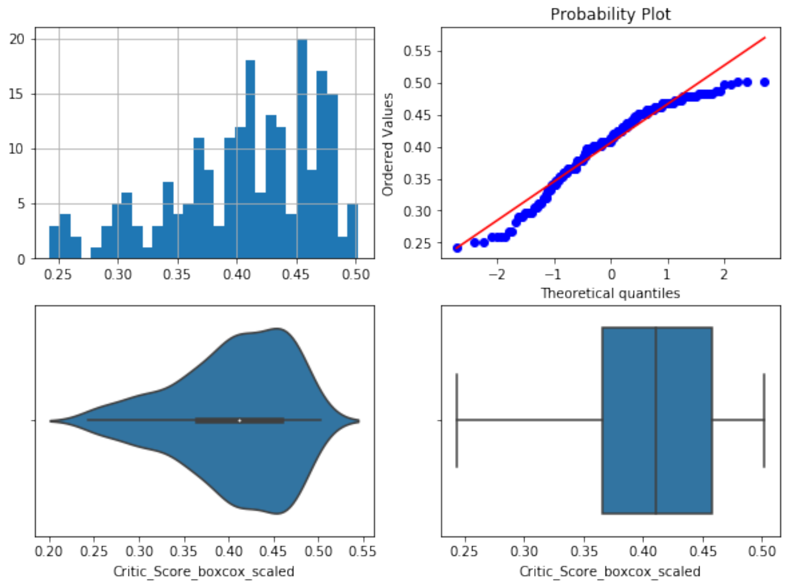
Поле-Critic_Score_boxcox_scaled, метод-OutlierBoundaryType.SIGMA, строк-215



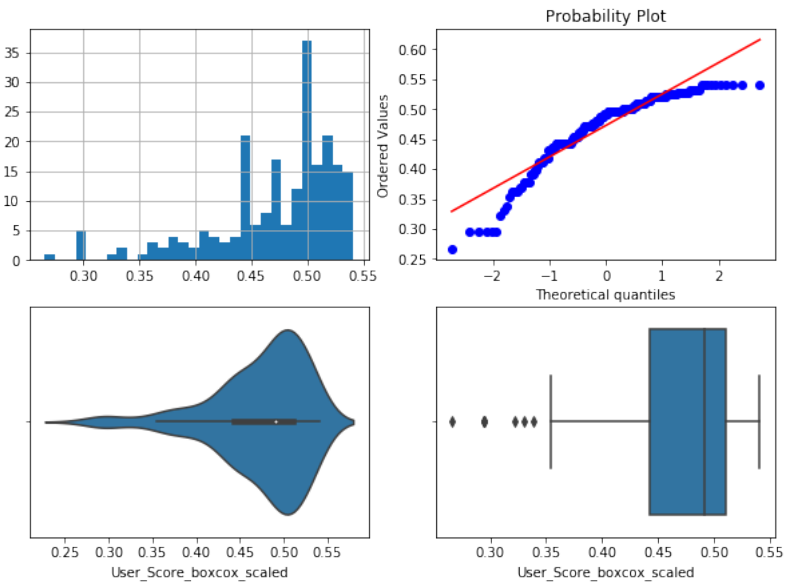
Поле-Critic_Score_boxcox_scaled, метод-OutlierBoundaryType.QUANTILE, строк-202

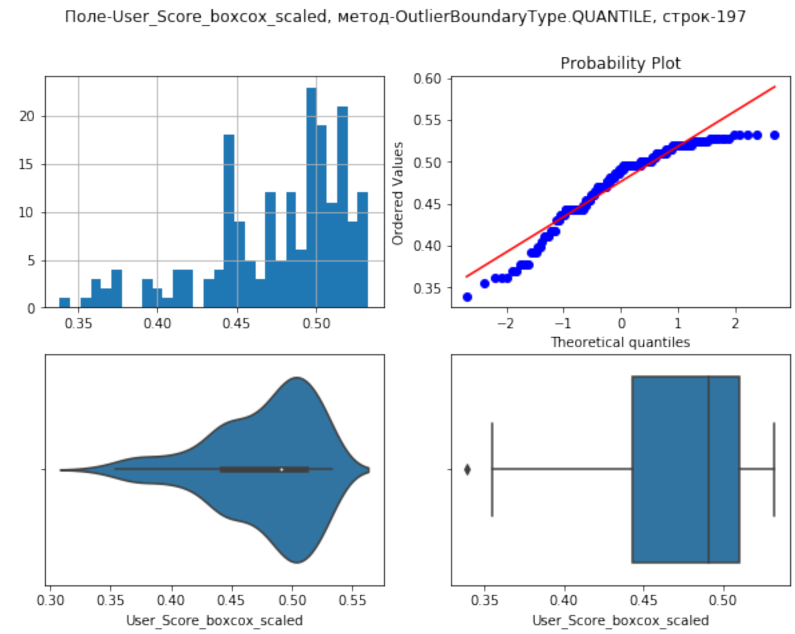


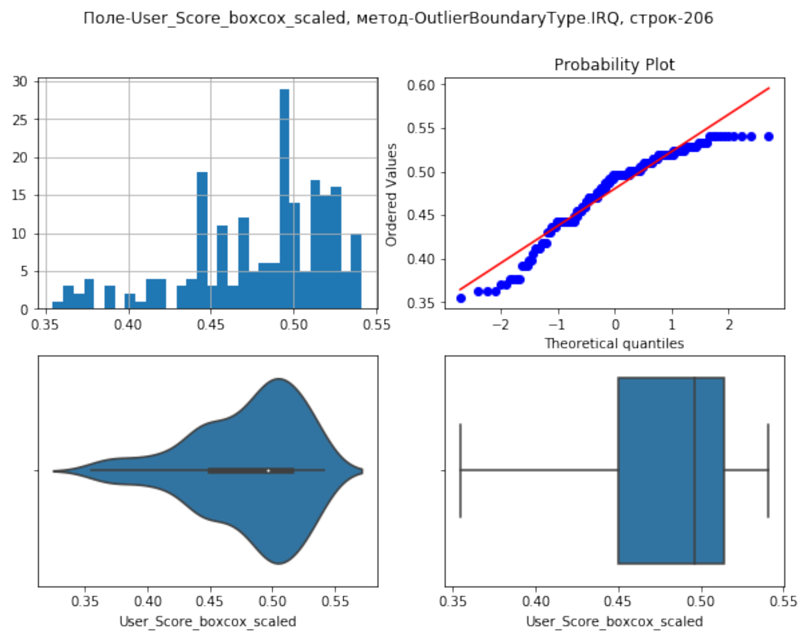
Поле-Critic_Score_boxcox_scaled, метод-OutlierBoundaryType.IRQ, строк-212



Поле-User_Score_boxcox_scaled, метод-OutlierBoundaryType.SIGMA, строк-215

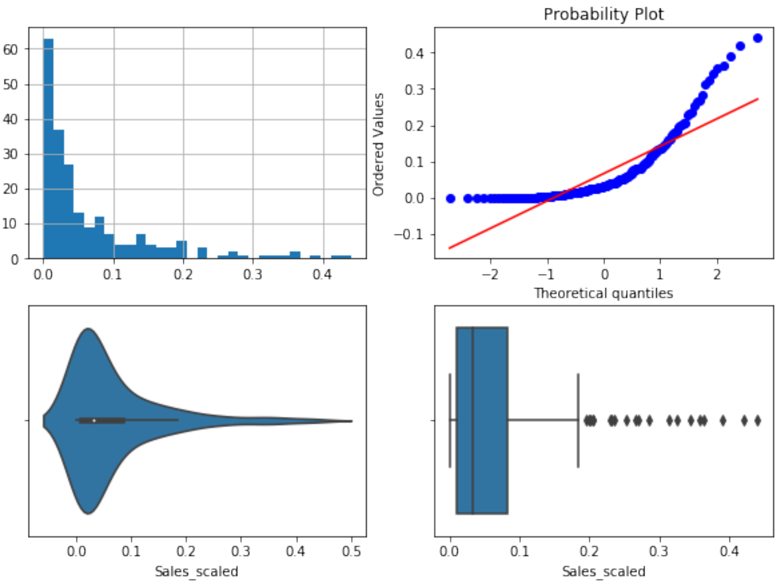




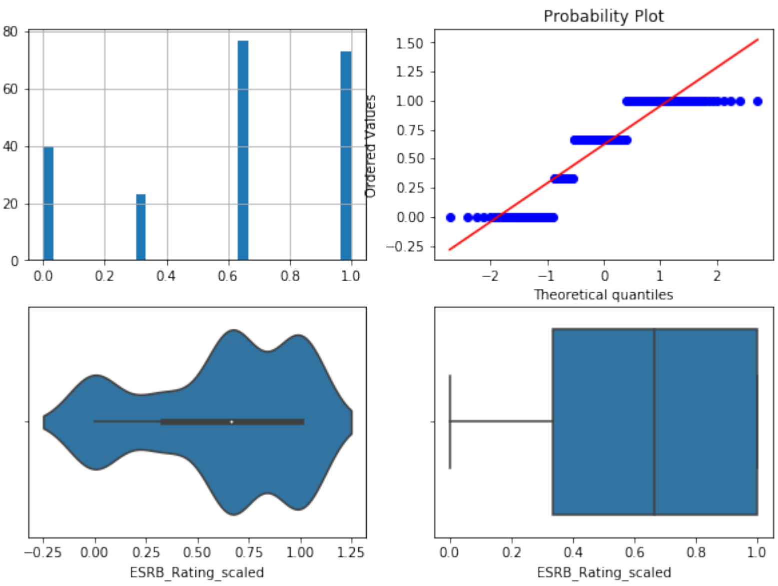



```
for col in scaled_columns:
    obt = OutlierBoundaryType.SIGMA
    # Вычисление верхней и нижней границы
    lower_boundary, upper_boundary = get_outlier_boundaries(df, col, obt)
    # Флаги для удаления выбросов
    outliers_temp = np.where(df[col] > upper_boundary, True,
                             np.where(df[col] < lower_boundary, True, False))
    # Удаление данных на основе флага
    df_trimmed = df.loc[~(outliers_temp), ]
    title = 'Поле-{}, метод-{}, строк-{}'.format(col, obt, df_trimmed.shape[0])
    diagnostic_plots(df_trimmed, col, title)
    df = df_trimmed
```

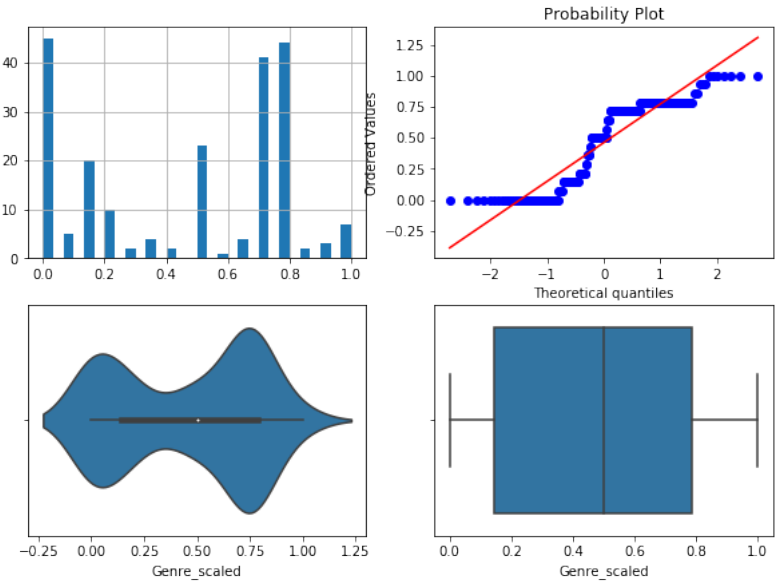
Поле-Sales_scaled, метод-OutlierBoundaryType.SIGMA, строк-213



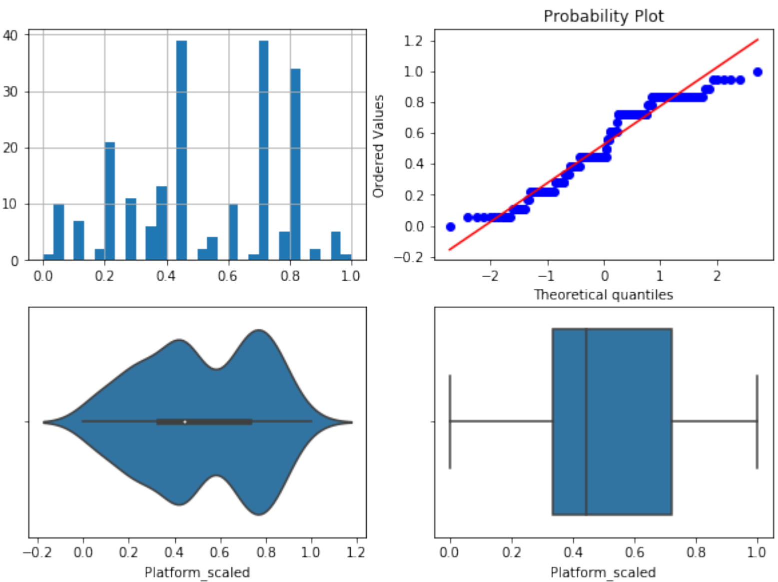
Поле-ESRB_Rating_scaled, метод-OutlierBoundaryType.SIGMA, строк-213



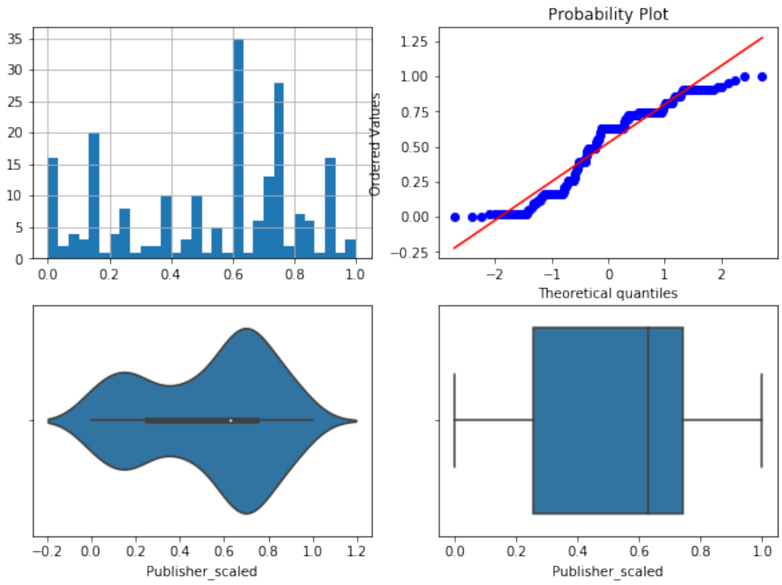
Поле-Genre_scaled, метод-OutlierBoundaryType.SIGMA, строк-213



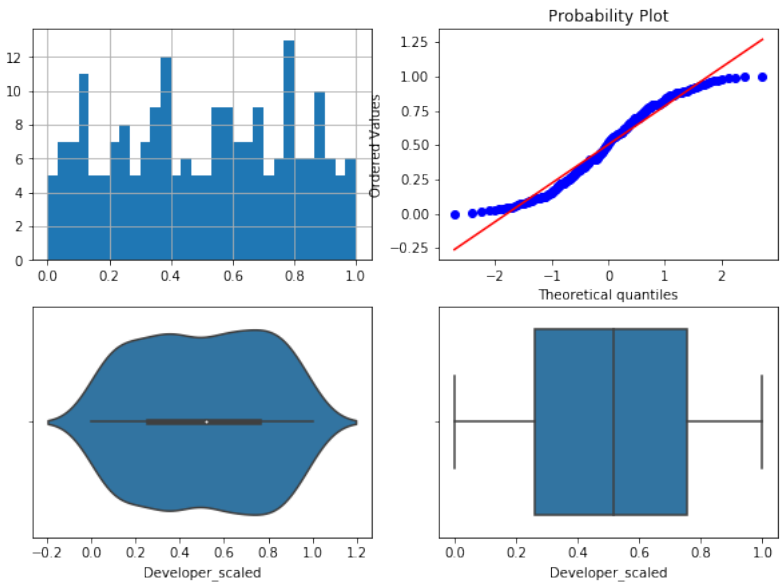
Поле-Platform_scaled, метод-OutlierBoundaryType.SIGMA, строк-213



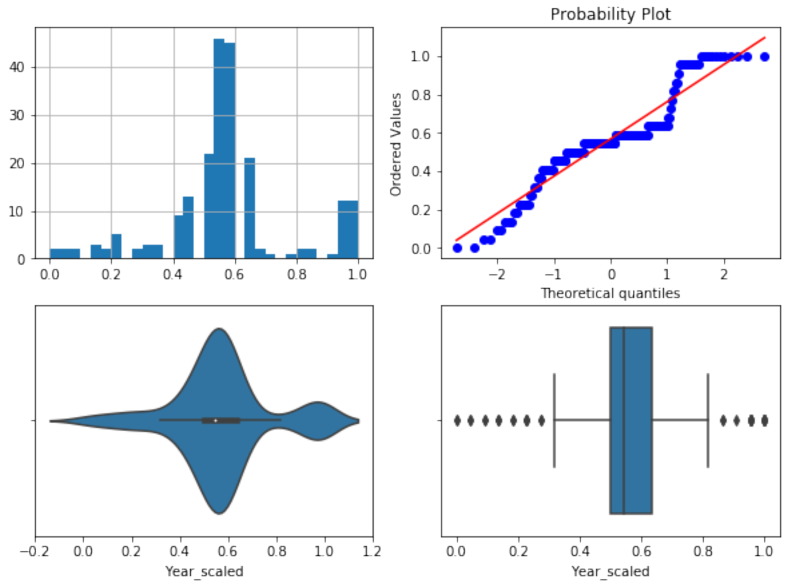
Поле-Publisher_scaled, метод-OutlierBoundaryType.SIGMA, строк-213



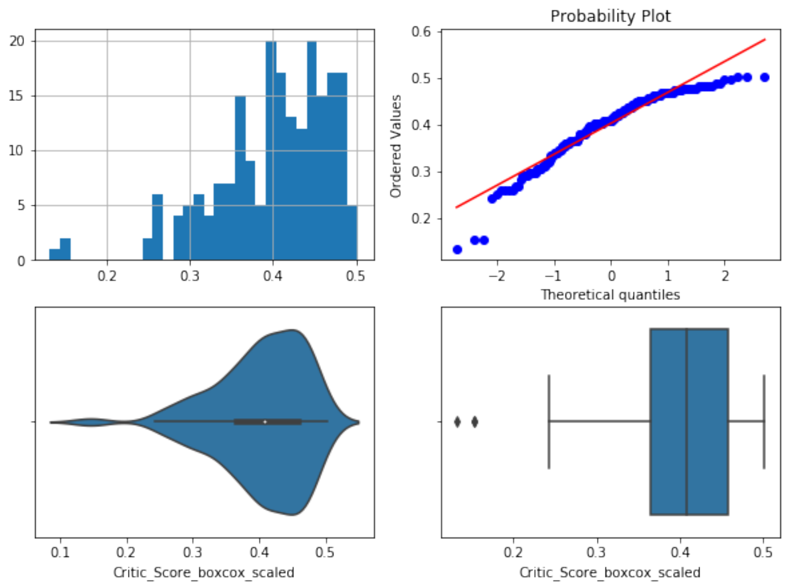
Поле-Developer_scaled, метод-OutlierBoundaryType.SIGMA, строк-213

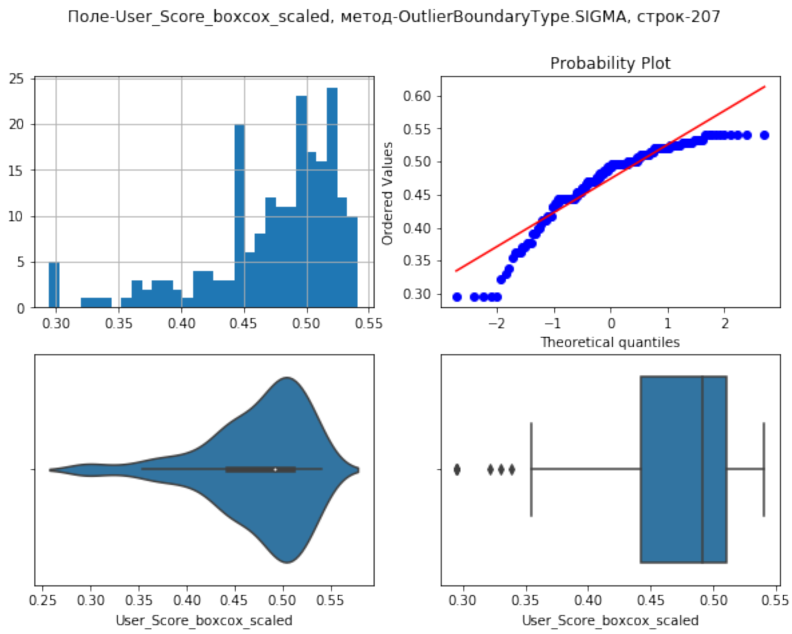


Поле-Year_scaled, метод-OutlierBoundaryType.SIGMA, строк-213



Поле-Critic_Score_boxcox_scaled, метод-OutlierBoundaryType.SIGMA, строк-209






```python
df.tail(2)
```

```css
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Rank	Name	Genre	ESRB_Rating	Platform	Publisher	Developer	Critic_Score	User_Score	Year	...	User_Score_boxcox	S
216	54538	Perception	2	3	4	13	122	6.0	7.9	2017.0	...	1.179739	0
217	55424	Thumper	6	4	4	9	32	9.0	9.3	2017.0	...	1.224249	0

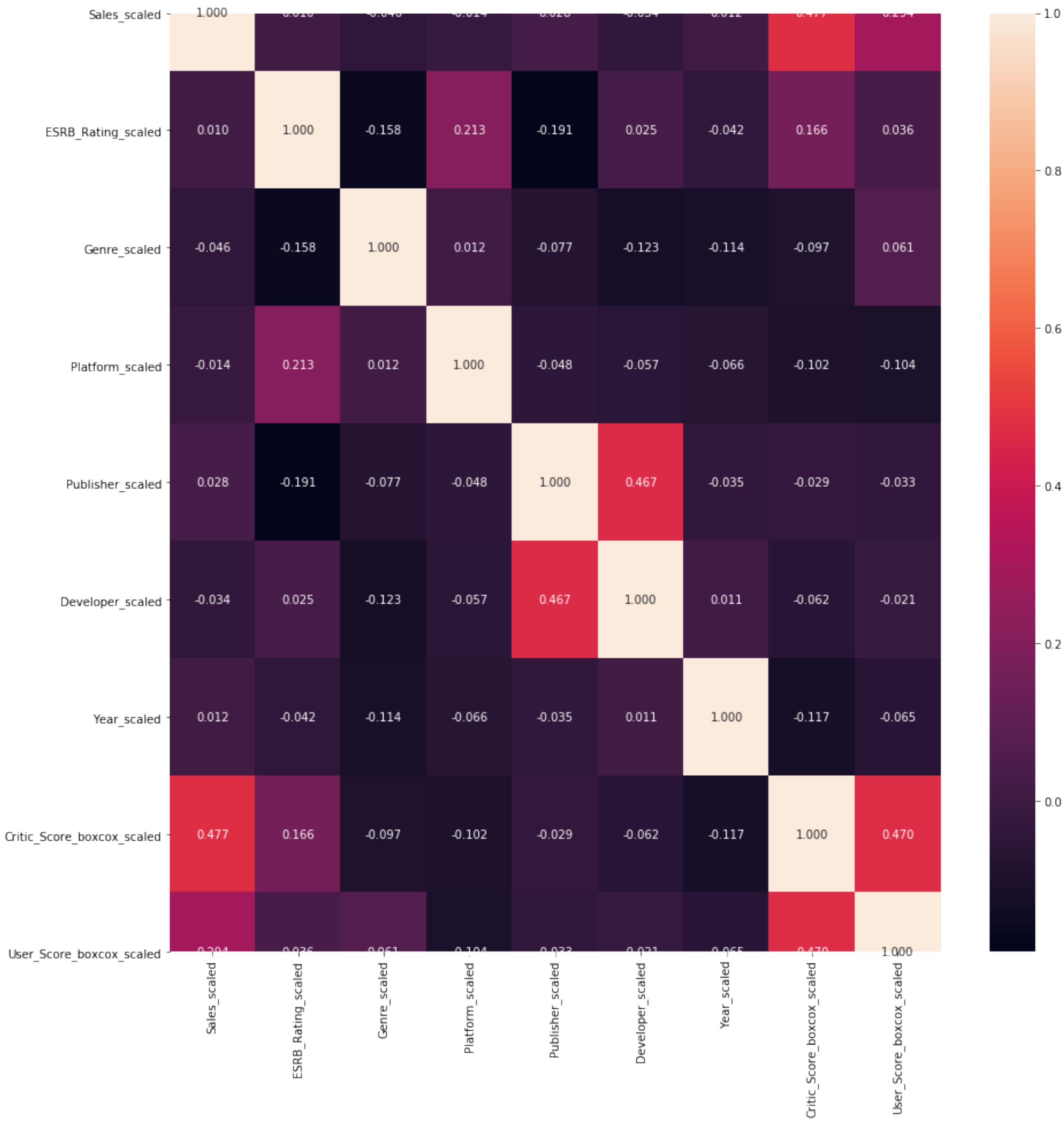
2 rows × 22 columns

Отбор признаков

Методы фильтрации (filter methods)

На основе оценки корреляции

```python
plt.figure(figsize=(15,15))
sns.heatmap(df[scaled_columns].corr(), annot=True, fmt='.3f')
plt.show()
```




```
def make_corr_df(df, tr=0.6):
    cr = df.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= tr]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
    cr.columns = ['f1', 'f2', 'corr']
    return cr
```

```
make_corr_df(df)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	f1	f2	corr

```python
# Обнаружение групп коррелирующих признаков
def corr_groups(cr):
    grouped_feature_list = []
    correlated_groups = []

    for feature in cr['f1'].unique():
        if feature not in grouped_feature_list:
            # находим коррелирующие признаки
            correlated_block = cr[cr['f1'] == feature]
            cur_dups = list(correlated_block['f2'].unique()) + [feature]
            grouped_feature_list = grouped_feature_list + cur_dups
            correlated_groups.append(cur_dups)
    return correlated_groups
```

```python
# Группы коррелирующих признаков
drop_cols = []
for g in corr_groups(make_corr_df(df)):
    for f in g:
        if '_scaled' not in f:
            drop_cols.append(f)

print(drop_cols)
```

```
[]
```

Методы обертывания (wrapper methods)

На основе алгоритма полного перебора

```python
from sklearn.svm import SVR
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import Lasso
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import mutual_info_classif, mutual_info_regression
from sklearn.feature_selection import SelectKBest, SelectPercentile
```

```python
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS

lr = LinearRegression()
```

```python
y_column = 'Sales'

trash_cols = {y_column, 'Sales_scaled', 'Developer', 'Name', 'Platform', 'Rank', 'Year', 'Publisher'}
x_columns = set(df.columns) - trash_cols

print(y_column, x_columns)
```

```
Sales {'Platform_scaled', 'Year_scaled', 'Developer_scaled', 'User_Score_boxcox_scaled', 'Genre_scaled', 'Publisher_scaled',
'Critic_Score_boxcox_scaled', 'ESRB_Rating_scaled'}
```

```python
def train_efs(df, x_cols, y_col, min_f=2, max_f=4, cv=5, model=None, scoring='neg_mean_squared_error'):
    if model is None:
        model = LinearRegression()

    efs = EFS(model,
              min_features=min_f,
              max_features=max_f,
```

```
                scoring=scoring,
                print_progress=True,
                cv=cv)
    efs = efs.fit(df[x_cols], pd.DataFrame(df[y_col]))

    print('Best accuracy score: %.2f' % efs.best_score_)
    print('Best subset (indices):', efs.best_idx_)
    print('Best subset (corresponding names):', efs.best_feature_names_)
    return efs1
```

```
efs1 = train_efs(df, x_columns, y_column, 4, 8, 5, lr)
```

```
Features: 163/163

Best accuracy score: -13.71
Best subset (indices): (0, 2, 3, 6)
Best subset (corresponding names): ('Platform_scaled', 'Developer_scaled', 'User_Score_boxcox_scaled', 'Critic_Score_boxcox_scaled')
```

```
efs2 = train_efs(df, x_columns, y_column, 2, 8, 5, lr)
```

```
Features: 247/247

Best accuracy score: -13.61
Best subset (indices): (3, 6)
Best subset (corresponding names): ('User_Score_boxcox_scaled', 'Critic_Score_boxcox_scaled')
```

Методы вложений (embedded methods)

На основе линейной регрессии

```
e_ls1 = LinearRegression()
e_ls1.fit(df[x_columns], df[y_column])
# Коэффициенты регрессии
list(zip(x_columns, e_ls1.coef_))
```

```
[('Platform_scaled', 0.8135289455413431),
 ('Year_scaled', 1.1786807196107814),
 ('Developer_scaled', -0.25420000858681396),
 ('User_Score_boxcox_scaled', 5.662058101682488),
 ('Genre_scaled', -0.10089104471518615),
 ('Publisher_scaled', 0.5227810641436722),
 ('Critic_Score_boxcox_scaled', 22.695110016228906),
 ('ESRB_Rating_scaled', -0.6470729147251617)]
```

```
sel_e_ls1 = SelectFromModel(e_ls1)
sel_e_ls1.fit(df[x_columns], df[y_column])
list(zip(x_columns, sel_e_ls1.get_support()))
```

```
[('Platform_scaled', False),
 ('Year_scaled', False),
 ('Developer_scaled', False),
 ('User_Score_boxcox_scaled', True),
 ('Genre_scaled', False),
 ('Publisher_scaled', False),
 ('Critic_Score_boxcox_scaled', True),
 ('ESRB_Rating_scaled', False)]
```

Очистка фич

```
df = df.drop(columns=drop_cols)
df.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Rank	Name	Platform	Publisher	Developer	Year	Sales	Sales_scaled	ESRB_Rating_scaled	Genre_scaled	Platform_scaled	Pub
6	29	Pokemon X/Y	0	27	48	2013.0	16.37	0.440765	0.000000	0.714286	0.000000	0.6
7	34	Pokemon Black / White Version	1	27	48	2011.0	15.64	0.421109	0.000000	0.714286	0.055556	0.6
8	44	Halo 3	15	21	15	2007.0	14.50	0.390415	1.000000	0.785714	0.833333	0.4
9	50	Call of Duty: Modern Warfare 2	15	1	56	2009.0	13.53	0.364297	1.000000	0.785714	0.833333	0.0
10	53	Super Smash Bros. Brawl	13	27	91	2008.0	13.29	0.357835	0.666667	0.214286	0.722222	0.6

```python
# df.drop(columns=['Platform', 'Publisher', 'Developer', 'Year', 'Sales_scaled'])
```

Сохраняем

```python
save_path = 'video_games_s3.csv'
df.to_csv(save_path, index=False)
```

```python
check_df = pd.read_csv(save_path)
check_df.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Rank	Name	Platform	Publisher	Developer	Year	Sales	Sales_scaled	ESRB_Rating_scaled	Genre_scaled	Platform_scaled	Publ
0	29	Pokemon X/Y	0	27	48	2013.0	16.37	0.440765	0.000000	0.714286	0.000000	0.62
1	34	Pokemon Black / White Version	1	27	48	2011.0	15.64	0.421109	0.000000	0.714286	0.055556	0.62
2	44	Halo 3	15	21	15	2007.0	14.50	0.390415	1.000000	0.785714	0.833333	0.48
3	50	Call of Duty: Modern Warfare 2	15	1	56	2009.0	13.53	0.364297	1.000000	0.785714	0.833333	0.02
4	53	Super Smash Bros. Brawl	13	27	91	2008.0	13.29	0.357835	0.666667	0.214286	0.722222	0.62