Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Лабораторная работа №2
по курсу «Методы машинного обучения»

«Обработка признаков (часть 1)»

ИСПОЛНИТЕЛЬ:

Андронов Д.О.

Группа ИУ5-24М

_____

"__"_____2022 г.

Москва 2022

Цель работы:

Изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

Задание:

Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.) Просьба не использовать датасет, на котором данная задача решалась в лекции. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

- устранение пропусков в данных;
- кодирование категориальных признаков;
- нормализацию числовых признаков.

Выполнение:

Устранение пропусков в данных

In [2]:
```
data = pd.read_csv('top_100_movies_by_genres.csv', sep=",")
```

In [3]:
```
data.shape
```

Out[3]:
```
(1612, 5)
```

In [4]:
```
data.dtypes
```

Out[4]:
```
Genre                object
Rank                float64
RatingTomatometer    object
Title                object
No. of Reviews        int64
dtype: object
```

In [5]:
```
data.head()
```

Out[5]:

	Genre	Rank	RatingTomatometer	Title	No. of Reviews
0	Action & Adventure	1.0	96%	Black Panther (2018)	525
1	Action & Adventure	2.0	94%	Avengers: Endgame (2019)	547
2	Action & Adventure	3.0	97%	Mission: Impossible - Fallout (2018)	437
3	Action & Adventure	4.0	97%	Mad Max: Fury Road (2015)	434
4	Action & Adventure	5.0	97%	Spider-Man: Into the Spider-Verse (2018)	393

In [6]:
```
def get_loss(some_data):
    for col in some_data.columns:
        null_counter = some_data[some_data[col].isnull()].shape[0]
        print("{} : {}".format(col,null_counter))
```

In [39]:
```
# % пропусков
[(c, data2[c].isnull().mean()) for c in data2]
```

Out[39]:
```
[('Genre', 0.17990074441687345),
 ('Rank', 0.184863523573201),
 ('RatingTomatometer', 0.07568238213399504),
 ('Title', 0.19044665012406947),
 ('No. of Reviews', 0.15384615384615385)]
```

In [40]:
```
array_to_del = ['Genre','Title']
```

In [41]:
```
new_data = data2.dropna(subset = array_to_del)
```

In [42]:
```
[(c, new_data[c].isnull().mean()) for c in new_data]
```

Out[42]:
```
[('Genre', 0.0),
 ('Rank', 0.1746031746031746),
 ('RatingTomatometer', 0.07936507936507936),
 ('Title', 0.0),
 ('No. of Reviews', 0.16339869281045752)]
```

In [46]:
```
def impute_na(df, variable, value):
    df[variable].fillna(value, inplace=True)
impute_na(new_data, 'RatingTomatometer', new_data['RatingTomatometer'].mean())
impute_na(new_data, 'No. of Reviews', new_data['No. of Reviews'].mean())
impute_na(new_data, 'Rank', new_data['Rank'].mean())
```

```
c:\users\s4lly\pycharmprojects\lab1\venv\lib\site-packages\pandas\core\series.py:4536: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  downcast=downcast,
```

In [47]:
```
get_loss(new_data)
```

```
Genre : 0
Rank : 0
RatingTomatometer : 0
Title : 0
No. of Reviews : 0
```

Кодирование категориальных признаков

In [51]:
```
# one-hot
pd.get_dummies(new_data2[['Genre']]).head()
```

Out[51]:

	Genre_Action & Adventure	Genre_Animation	Genre_Art House & International	Genre_Classics	Genre_Comedy	Genre_Documentary	Genre_Drama	Genre_Horror	G
0	1	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	0	
4	1	0	0	0	0	0	0	0	
5	1	0	0	0	0	0	0	0	

In [62]:

```
# Count (frequency) encoding
import category_encoders as ce
new_data3 = new_data.copy()
data_OHE = new_data.copy()
ce_CountEncoder1 = ce.CountEncoder()
data_OHE[['Genre']] = ce_CountEncoder1.fit_transform(new_data3['Genre'])
data_OHE
```

Out[62]:

	Genre	Rank	RatingTomatometer	Title	No. of Reviews
0	75	1.000000	96.000000	Black Panther (2018)	525.000000
1	75	2.000000	94.000000	Avengers: Endgame (2019)	547.000000
3	75	4.000000	97.000000	Mad Max: Fury Road (2015)	434.000000
4	75	5.000000	97.000000	Spider-Man: Into the Spider-Verse (2018)	393.000000
5	75	6.000000	92.196755	Wonder Woman (2017)	148.551339
...	...	...	...	...	...
1606	65	81.000000	17.000000	Wild Wild West (1999)	132.000000
1608	65	48.680995	16.000000	September Dawn (2007)	148.551339
1609	65	84.000000	14.000000	American Outlaws (2001)	103.000000
1610	65	85.000000	12.000000	Jonah Hex (2010)	153.000000
1611	65	86.000000	2.000000	Texas Rangers (2001)	51.000000

1071 rows × 5 columns

In [63]:

```
data_OHE['Genre'].unique()
```

Out[63]:

```
array([75, 69, 62, 60, 64, 65, 68, 55, 52, 66, 73, 72, 49, 38],
      dtype=int64)
```

In [64]:

```
ce_CountEncoder1 = ce.CountEncoder(normalize = True)
data_OHE[['Genre']] = ce_CountEncoder1.fit_transform(new_data3['Genre'])
data_OHE
```

Out[64]:

	Genre	Rank	RatingTomatometer	Title	No. of Reviews
0	0.070028	1.000000	96.000000	Black Panther (2018)	525.000000
1	0.070028	2.000000	94.000000	Avengers: Endgame (2019)	547.000000
3	0.070028	4.000000	97.000000	Mad Max: Fury Road (2015)	434.000000
4	0.070028	5.000000	97.000000	Spider-Man: Into the Spider-Verse (2018)	393.000000
5	0.070028	6.000000	92.196755	Wonder Woman (2017)	148.551339
...	...	...	...	...	...
1606	0.060691	81.000000	17.000000	Wild Wild West (1999)	132.000000
1608	0.060691	48.680995	16.000000	September Dawn (2007)	148.551339
1609	0.060691	84.000000	14.000000	American Outlaws (2001)	103.000000
1610	0.060691	85.000000	12.000000	Jonah Hex (2010)	153.000000
1611	0.060691	86.000000	2.000000	Texas Rangers (2001)	51.000000

1071 rows × 5 columns

In [70]:

```
# Target (Mean) encoding
# На самом деле этот метод реализует Mean encoding
from category_encoders import TargetEncoder
data_MEAN_ENC = new_data.copy()
ce_TargetEncoder1 = TargetEncoder()
data_MEAN_ENC['Genre'] = ce_TargetEncoder1.fit_transform(new_data['Genre'], new_data['RatingTomatometer'])
data_MEAN_ENC
```

Out[70]:

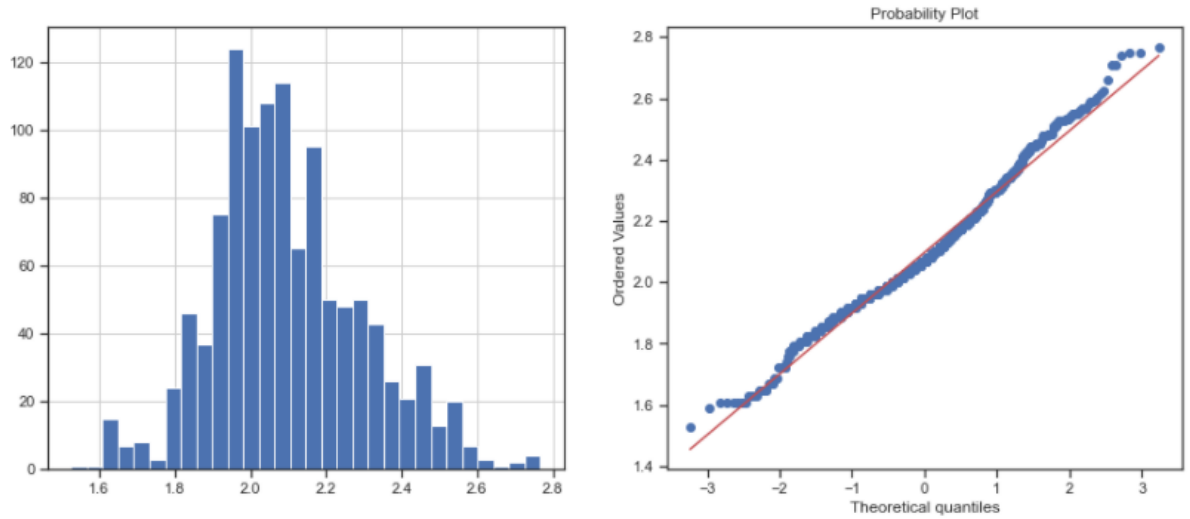
	Genre	Rank	RatingTomatometer	Title	No. of Reviews
0	94.338364	1.000000	96.000000	Black Panther (2018)	525.000000
1	94.338364	2.000000	94.000000	Avengers: Endgame (2019)	547.000000
3	94.338364	4.000000	97.000000	Mad Max: Fury Road (2015)	434.000000
4	94.338364	5.000000	97.000000	Spider-Man: Into the Spider-Verse (2018)	393.000000
5	94.338364	6.000000	92.196755	Wonder Woman (2017)	148.551339
...	...	...	...	...	...
1606	67.892058	81.000000	17.000000	Wild Wild West (1999)	132.000000
1608	67.892058	48.680995	16.000000	September Dawn (2007)	148.551339
1609	67.892058	84.000000	14.000000	American Outlaws (2001)	103.000000
1610	67.892058	85.000000	12.000000	Jonah Hex (2010)	153.000000
1611	67.892058	86.000000	2.000000	Texas Rangers (2001)	51.000000

Нормализацию числовых признаков

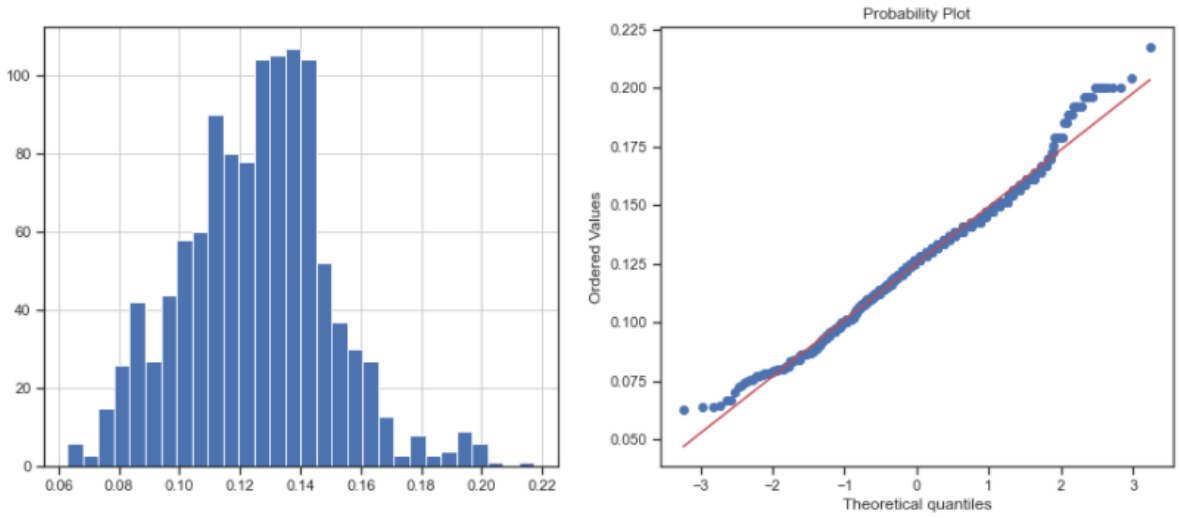
In [31]:

```
def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    # гистограмма
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

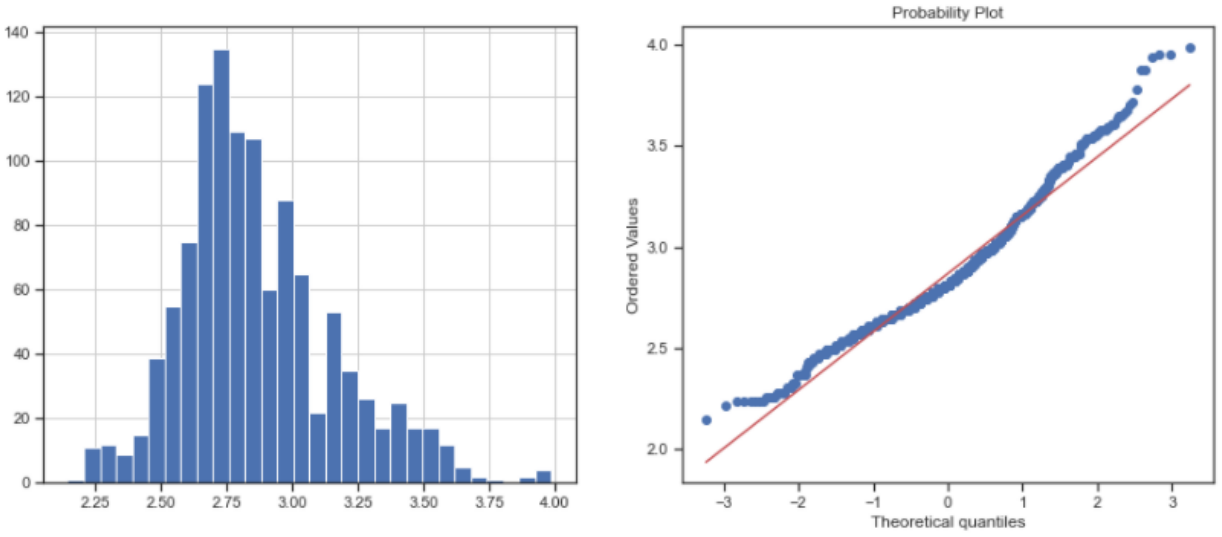
```
# логарифмическое
data['fixed acidity_log'] = np.log(data['fixed acidity'])
diagnostic_plots(data, 'fixed acidity_log')
```



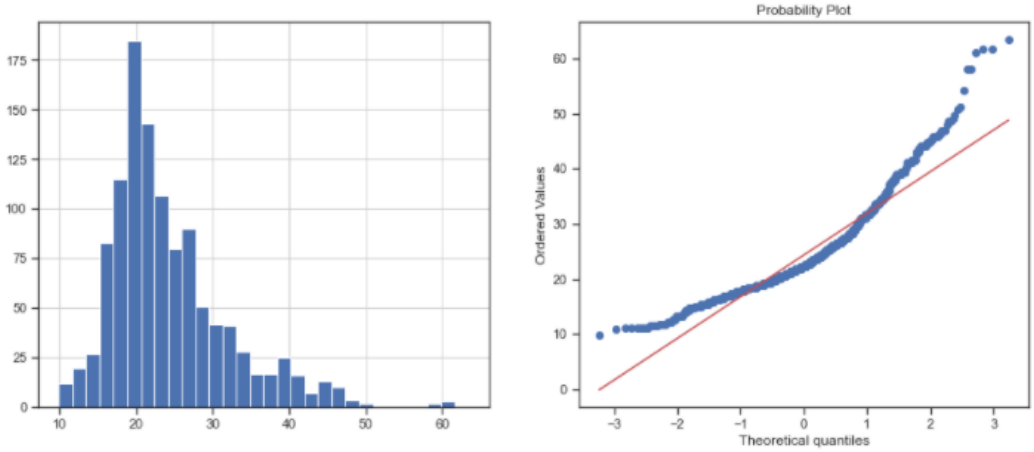
```
# обратное
data['fixed acidity_reciprocal'] = 1 / (data['fixed acidity'])
diagnostic_plots(data, 'fixed acidity_reciprocal')
```



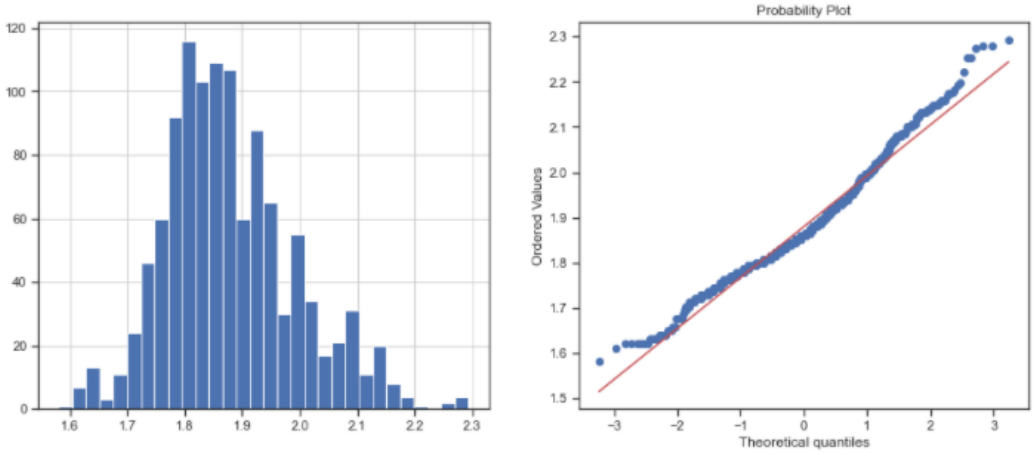

```
# корень вадратный
data['fixed acidity_sqr'] = data['fixed acidity']**(1/2)
diagnostic_plots(data, 'fixed acidity_sqr')
```



```
# возведение в степень
data['fixed acidity_exp1'] = data['fixed acidity']**(1.5)
diagnostic_plots(data, 'fixed acidity_exp1')
```

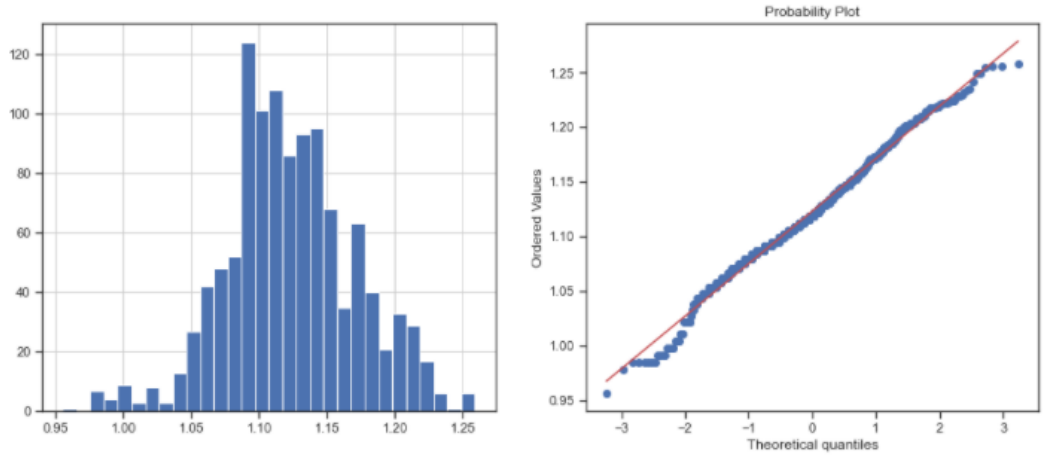



```
data['fixed acidity_exp2'] = data['fixed acidity']**(0.3)
diagnostic_plots(data, 'fixed acidity_exp2')
```



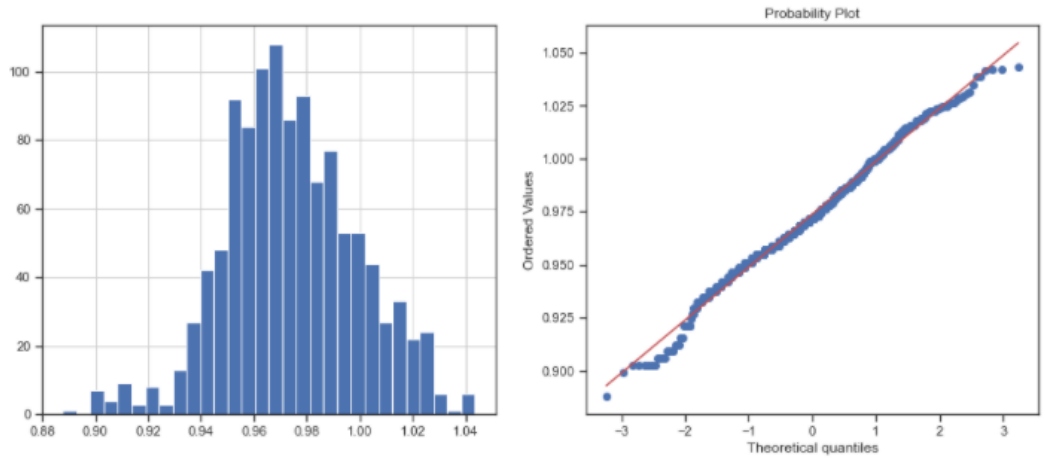
```
# Бокс-Кокс
data['fixed acidity_boxcox'], param = stats.boxcox(data['fixed acidity'])
print('a = {}'.format(param))
diagnostic_plots(data, 'fixed acidity_boxcox')
```

```
a = -0.6701020048823834
```




```
#Йео-Джонсон
data['fixed acidity_yeojohnson'], param = stats.yeojohnson(data['fixed acidity'])
print('a = {}'.format(param))
diagnostic_plots(data, 'fixed acidity_yeojohnson')
```

```
a = -0.8784004627239638
```



Вывод:

В результате работы были изучены способы предварительной обработки данных для дальнейшего формирования моделей.