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

Лабораторная работа №2
по дисциплине «Методы машинного обучения»
«Обработка признаков (часть 1)»

ИСПОЛНИТЕЛЬ:

Цветкова Алена
Группа ИУ5-21М

____

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

____

Москва, 2021

Цель лабораторной работы: изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

In [87]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
import scipy.stats as stats
```

In [69]:

```
data = pd.read_csv('archive/winemag-data-130k-v2.csv')
data.head()
```

Out[69]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	
0	0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	20
1	1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Qu A A
2	2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Ra 20 (Wi
3	3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	S I Ri
4	4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	V I

In [5]:

```
data.dtypes
```

Out[5]:

```
Unnamed: 0                 int64
country                   object
description               object
designation               object
points                     int64
price                    float64
province                  object
region_1                  object
```

```
region_2                  object
taster_name               object
taster_twitter_handle     object
title                     object
variety                   object
winery                    object
dtype: object
```

In [6]:

```
data.shape
```

Out[6]:

```
(129971, 14)
```

Колонки designation, region_1, region_2, taster_name, taster_twitter_handle содержат много пропущенных данных, заполнение пропусков в этих колонках может нарушить распределение исходных данных (поэтому лучше их удалить)

In [7]:

```
data.isnull().sum()
```

Out[7]:

```
Unnamed: 0                   0
country                     63
description                  0
designation              37465
points                       0
price                     8996
province                    63
region_1                 21247
region_2                 79460
taster_name              26244
taster_twitter_handle    31213
title                        0
variety                      1
winery                       0
dtype: int64
```

1. Обработка пропусков в данных

1.1 Удаление

In [8]:

```
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

Out[8]:

```
((129971, 14), (129971, 5))
```

In [9]:

```
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

Out[9]:

```
((129971, 14), (22387, 14))
```

In [10]:

```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

1.2 Заполнение значений для одного признака

1.2.1 Числовой признак

In [11]:

```
total_count = data.shape[0]
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_nul
l_count, temp_perc))
```

```
Колонка price. Тип данных float64. Количество пустых значений 8996, 6.92%.
```

In [12]:

```
data_price = data[['price']]
data_price.head()
```

Out[12]:

	price
0	NaN
1	15.0
2	14.0
3	13.0
4	65.0

In [13]:

```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_price)
mask_missing_values_only
```

Out[13]:

```
array([[ True],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

In [14]:

```
strategies=['mean', 'median','most_frequent']
```

```
strategies=['mean', 'median', 'most_frequent']
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(data_price)
    return data_num_imp[mask_missing_values_only]
```

In [15]:

```
strategies[0], test_num_impute(strategies[0])
```

Out[15]:

```
('mean', array([35.36338913, 35.36338913, 35.36338913, ..., 35.36338913,
        35.36338913, 35.36338913]))
```

In [16]:

```
strategies[1], test_num_impute(strategies[1])
```

Out[16]:

```
('median', array([25., 25., 25., ..., 25., 25., 25.]))
```

In [17]:

```
strategies[2], test_num_impute(strategies[2])
```

Out[17]:

```
('most_frequent', array([20., 20., 20., ..., 20., 20., 20.]))
```

In [18]:

```
def impute_column(dataset, column, strategy_param, fill_value_param=None):
    """
    Заполнение пропусков в одном признаке
    """
    temp_data = dataset[[column]].values
    size = temp_data.shape[0]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imputer = SimpleImputer(strategy=strategy_param,
                            fill_value=fill_value_param)
    all_data = imputer.fit_transform(temp_data)

    missed_data = temp_data[mask_missing_values_only]
    filled_data = all_data[mask_missing_values_only]

    return all_data.reshape((size,)), filled_data, missed_data
```

In [19]:

```
all_data, filled_data, missed_data = impute_column(data, 'price', 'median')
```

In [20]:

```
all_data
```

Out[20]:

```
array([25., 15., 14., ..., 30., 32., 21.])
```

In [21]:

```
filled_data
```

Out[21]:

```
array([25., 25., 25., ..., 25., 25., 25.])
```

In [22]:

```
missed_data
```

Out[22]:

```
array([nan, nan, nan, ..., nan, nan, nan])
```

1.2.2 Категориальный признак

In [23]:

```
cat_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='object'):
        cat_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 4)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_nul
l_count, temp_perc))
```

```
Колонка country. Тип данных object. Количество пустых значений 63, 0.0485%.
Колонка designation. Тип данных object. Количество пустых значений 37465, 28.8257%.
Колонка province. Тип данных object. Количество пустых значений 63, 0.0485%.
Колонка region_1. Тип данных object. Количество пустых значений 21247, 16.3475%.
Колонка region_2. Тип данных object. Количество пустых значений 79460, 61.1367%.
Колонка taster_name. Тип данных object. Количество пустых значений 26244, 20.1922%.
Колонка taster_twitter_handle. Тип данных object. Количество пустых значений 31213, 24.0154%.
Колонка variety. Тип данных object. Количество пустых значений 1, 0.0008%.
```

In [24]:

```
cat_temp_data = data[['country']]
cat_temp_data.head()
```

Out[24]:

	country
0	Italy
1	Portugal
2	US
3	US
4	US

In [25]:

```
cat_temp_data['country'].unique()
```

Out[25]:

```
array(['Italy', 'Portugal', 'US', 'Spain', 'France', 'Germany',
       'Argentina', 'Chile', 'Australia', 'Austria', 'South Africa',
       'New Zealand', 'Israel', 'Hungary', 'Greece', 'Romania', 'Mexico',
       'Canada', nan, 'Turkey', 'Czech Republic', 'Slovenia',
       'Luxembourg', 'Croatia', 'Georgia', 'Uruguay', 'England',
       'Lebanon', 'Serbia', 'Brazil', 'Moldova', 'Morocco', 'Peru',
       'India', 'Bulgaria', 'Cyprus', 'Armenia', 'Switzerland',
       'Bosnia and Herzegovina', 'Ukraine', 'Slovakia', 'Macedonia',
       'China', 'Egypt'], dtype=object)
```

In [26]:

```
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

Out[26]:

```
array([['Italy'],
       ['Portugal'],
       ['US'],
       ...,
       ['France'],
       ['France'],
       ['France']], dtype=object)
```

In [27]:

```
np.unique(data_imp2)
```

Out[27]:

```
array(['Argentina', 'Armenia', 'Australia', 'Austria',
       'Bosnia and Herzegovina', 'Brazil', 'Bulgaria', 'Canada', 'Chile',
       'China', 'Croatia', 'Cyprus', 'Czech Republic', 'Egypt', 'England',
       'France', 'Georgia', 'Germany', 'Greece', 'Hungary', 'India',
       'Israel', 'Italy', 'Lebanon', 'Luxembourg', 'Macedonia', 'Mexico',
       'Moldova', 'Morocco', 'New Zealand', 'Peru', 'Portugal', 'Romania',
       'Serbia', 'Slovakia', 'Slovenia', 'South Africa', 'Spain',
       'Switzerland', 'Turkey', 'US', 'Ukraine', 'Uruguay'], dtype=object)
```

In [28]:

```
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='Unknown')
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

Out[28]:

```
array([['Italy'],
       ['Portugal'],
       ['US'],
       ...,
       ['France'],
       ['France'],
       ['France']], dtype=object)
```

In [29]:

```
np.unique(data_imp3)
```

Out[29]:

```
array(['Argentina', 'Armenia', 'Australia', 'Austria',
       'Bosnia and Herzegovina', 'Brazil', 'Bulgaria', 'Canada', 'Chile',
       'China', 'Croatia', 'Cyprus', 'Czech Republic', 'Egypt', 'England',
       'France', 'Georgia', 'Germany', 'Greece', 'Hungary', 'India',
```

```
       'France', 'Georgia', 'Germany', 'Greece', 'Hungary', 'India',
       'Israel', 'Italy', 'Lebanon', 'Luxembourg', 'Macedonia', 'Mexico',
       'Moldova', 'Morocco', 'New Zealand', 'Peru', 'Portugal', 'Romania',
       'Serbia', 'Slovakia', 'Slovenia', 'South Africa', 'Spain',
       'Switzerland', 'Turkey', 'US', 'Ukraine', 'Unknown', 'Uruguay'],
      dtype=object)
```

In [30]:

```
data[data['country'].isnull()].shape[0]
```

Out[30]:

```
63
```

In [31]:

```
data_imp3[data_imp3=='Unknown'].size
```

Out[31]:

```
63
```

2. Кодирование категориальных признаков

In [32]:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [33]:

```
cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
cat_enc.head()
```

Out[33]:

	c1
0	Italy
1	Portugal
2	US
3	US
4	US

2.1 Кодирование категорий целочисленными значениями

In [34]:

```
le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
```

In [35]:

```
cat_enc['c1'].unique()
```

Out[35]:

```
array(['Italy', 'Portugal', 'US', 'Spain', 'France', 'Germany',
```

```
       'Argentina', 'Chile', 'Australia', 'Austria', 'South Africa',
       'New Zealand', 'Israel', 'Hungary', 'Greece', 'Romania', 'Mexico',
       'Canada', 'Turkey', 'Czech Republic', 'Slovenia', 'Luxembourg',
       'Croatia', 'Georgia', 'Uruguay', 'England', 'Lebanon', 'Serbia',
       'Brazil', 'Moldova', 'Morocco', 'Peru', 'India', 'Bulgaria',
       'Cyprus', 'Armenia', 'Switzerland', 'Bosnia and Herzegovina',
       'Ukraine', 'Slovakia', 'Macedonia', 'China', 'Egypt'], dtype=object)
```

In [36]:

```
np.unique(cat_enc_le)
```

Out[36]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42])
```

In [37]:

```
le.inverse_transform([0, 1, 2, 3, 4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42])
```

Out[37]:

```
array(['Argentina', 'Armenia', 'Australia', 'Austria',
       'Bosnia and Herzegovina', 'Brazil', 'Bulgaria', 'Canada', 'Chile',
       'China', 'Croatia', 'Cyprus', 'Czech Republic', 'Egypt', 'England',
       'France', 'Georgia', 'Germany', 'Greece', 'Hungary', 'India',
       'Israel', 'Italy', 'Lebanon', 'Luxembourg', 'Macedonia', 'Mexico',
       'Moldova', 'Morocco', 'New Zealand', 'Peru', 'Portugal', 'Romania',
       'Serbia', 'Slovakia', 'Slovenia', 'South Africa', 'Spain',
       'Switzerland', 'Turkey', 'US', 'Ukraine', 'Uruguay'], dtype=object)
```

2.2 Кодирование категорий наборами бинарных значений

In [38]:

```
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])
```

In [39]:

```
cat_enc.shape
```

Out[39]:

```
(129971, 1)
```

In [40]:

```
cat_enc_ohe.shape
```

Out[40]:

```
(129971, 43)
```

In [41]:

```
cat_enc_ohe.todense()[0:10]
```

Out[41]:

```
matrix([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

In [42]:

```
pd.get_dummies(cat_enc, dummy_na=True).head()
```

Out[42]:

	c1_Argentina	c1_Armenia	c1_Australia	c1_Austria	c1_Bosnia and Herzegovina	c1_Brazil	c1_Bulgaria	c1_Canada	c1_Chile	c1_China	...	c1_Sl
0	0	0	0	0	0	0	0	0	0	0	...	
1	0	0	0	0	0	0	0	0	0	0	...	
2	0	0	0	0	0	0	0	0	0	0	...	
3	0	0	0	0	0	0	0	0	0	0	...	
4	0	0	0	0	0	0	0	0	0	0	...	

5 rows × 44 columns

2.3 Count (frequency) encoding

In [43]:

```
from category_encoders.count import CountEncoder as ce_CountEncoder
```

In [49]:

```
data_imp2_df = pd.DataFrame(data_imp2)
```

In [74]:

```
data['new_country'] = data_imp2_df[0]
```

In [75]:

```
data.head()
```

Out[75]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	
0	0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	20
1	1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Qu A A
2	2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Ra 20 (Wi
3	3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	S I Ri
4	4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	V I

In [76]:

```
ce_CountEncoder1 = ce_CountEncoder()
data_COUNT_ENC = ce_CountEncoder1.fit_transform(data[data.columns.difference(['new_country'])])
```

In [77]:

```
data_COUNT_ENC.head()
```

Out[77]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	v
0	0	19540	1	1	87	NaN	1797	268	79460	10776	10776	1	
1	1	5691	1	2	87	15.0	1281	21247	79460	25514	25514	1	
2	2	54567	1	37465	87	14.0	5373	2301	3423	9532	9532	1	
3	3	54567	1	8	87	13.0	114	27	79460	415	31213	1	
4	4	54567	1	1	87	65.0	5373	2301	3423	9532	9532	1	

In [78]:

```
data_COUNT_ENC['country'].head()
```

Out[78]:

Out[78]:

```
0    19540
1     5691
2    54567
3    54567
4    54567
Name: country, dtype: int64
```

In [79]:

```
data['country'].unique()
```

Out[79]:

```
array(['Italy', 'Portugal', 'US', 'Spain', 'France', 'Germany',
       'Argentina', 'Chile', 'Australia', 'Austria', 'South Africa',
       'New Zealand', 'Israel', 'Hungary', 'Greece', 'Romania', 'Mexico',
       'Canada', 'Turkey', 'Czech Republic', 'Slovenia', 'Luxembourg',
       'Croatia', 'Georgia', 'Uruguay', 'England', 'Lebanon', 'Serbia',
       'Brazil', 'Moldova', 'Morocco', 'Peru', 'India', 'Bulgaria',
       'Cyprus', 'Armenia', 'Switzerland', 'Bosnia and Herzegovina',
       'Ukraine', 'Slovakia', 'Macedonia', 'China', 'Egypt'], dtype=object)
```

In [80]:

```
data_COUNT_ENC['country'].unique()
```

Out[80]:

```
array([19540,  5691, 54567,  6645, 22093,  2165,  3800,  4472,  2329,
        3345,  1401,  1419,   505,   146,   466,   120,    70,   257,
          90,    12,    87,     6,    73,    86,   109,    74,    35,
          52,    59,    28,    16,     9,   141,    11,     2,     7,
          14,     1])
```

In [81]:

```
ce_CountEncoder2 = ce_CountEncoder(normalize=True)
data_FREQ_ENC = ce_CountEncoder2.fit_transform(data[data.columns.difference(['new_country'])])
```

In [83]:

```
data_FREQ_ENC.head()
```

Out[83]:

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	
0	0	0.150341	0.000008	0.000008	87	NaN	0.013826	0.002062	0.611367	0.082911	0.082911	0.0
1	1	0.043787	0.000008	0.000015	87	15.0	0.009856	0.163475	0.611367	0.196305	0.196305	0.0
2	2	0.419840	0.000008	0.288257	87	14.0	0.041340	0.017704	0.026337	0.073339	0.073339	0.0
3	3	0.419840	0.000008	0.000062	87	13.0	0.000877	0.000208	0.611367	0.003193	0.240154	0.0
4	4	0.419840	0.000008	0.000008	87	65.0	0.041340	0.017704	0.026337	0.073339	0.073339	0.0

In [84]:

```
data_FREQ_ENC['country'].unique()
```

Out[84]:

```
array([1.50341230e-01, 4.37866909e-02, 4.19839810e-01, 5.11267898e-02,
```

```
1.69984073e-01, 1.66575621e-02, 2.92372914e-02, 3.44076756e-02,
1.79193820e-02, 2.57365105e-02, 1.07793277e-02, 1.09178201e-02,
3.88548215e-03, 1.12332751e-03, 3.58541521e-03, 9.23282886e-04,
5.38581684e-04, 1.97736418e-03, 6.92462165e-04, 9.23282886e-05,
6.69380092e-04, 4.61641443e-05, 5.61663756e-04, 6.61686068e-04,
8.38648622e-04, 5.69357780e-04, 2.69290842e-04, 4.00089251e-04,
4.53947419e-04, 2.15432673e-04, 1.23104385e-04, 6.92462165e-05,
1.08485739e-03, 8.46342646e-05, 1.53880481e-05, 5.38581684e-05,
1.07716337e-04, 7.69402405e-06])
```

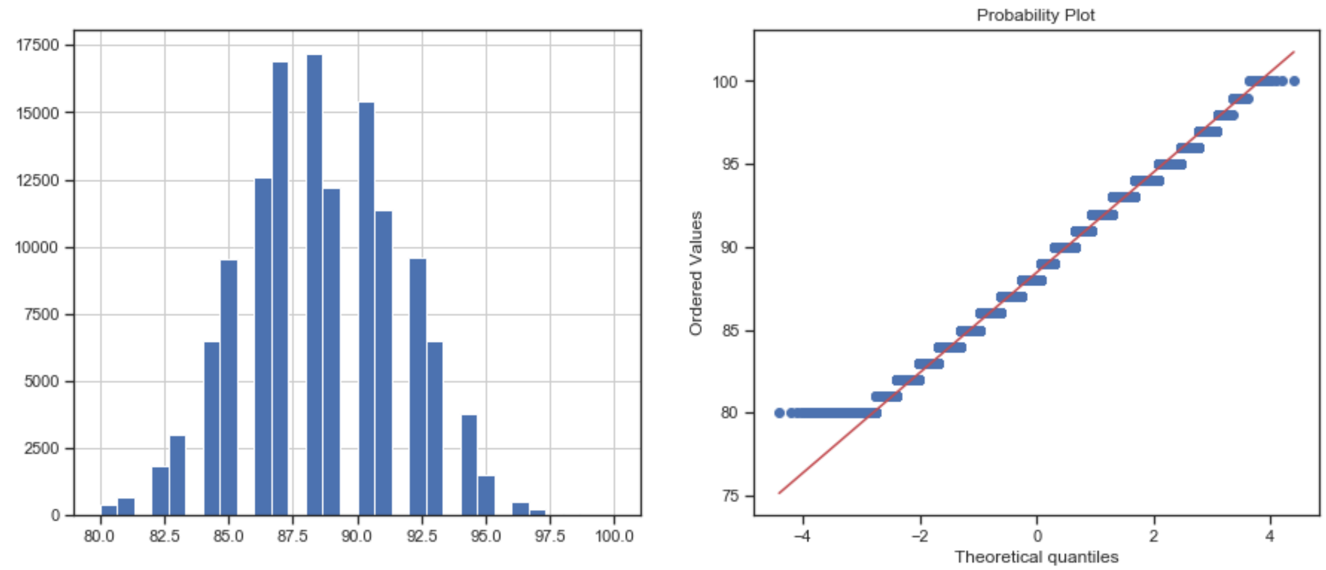
3. Нормализация числовых признаков

In [85]:

```python
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()
```

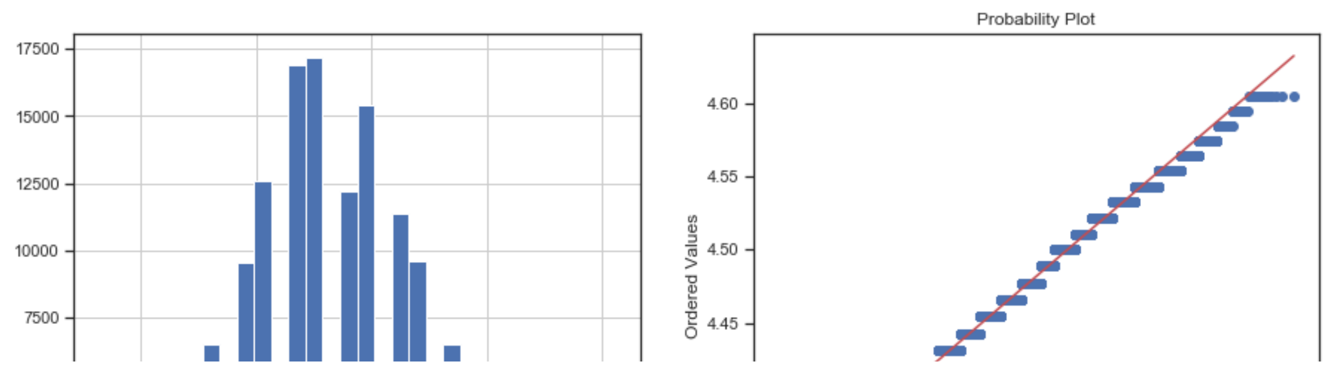
In [89]:

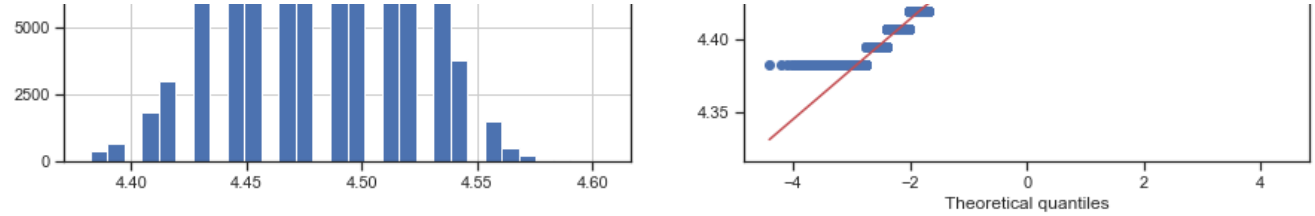
```python
diagnostic_plots(data, 'points')
```



In [90]:

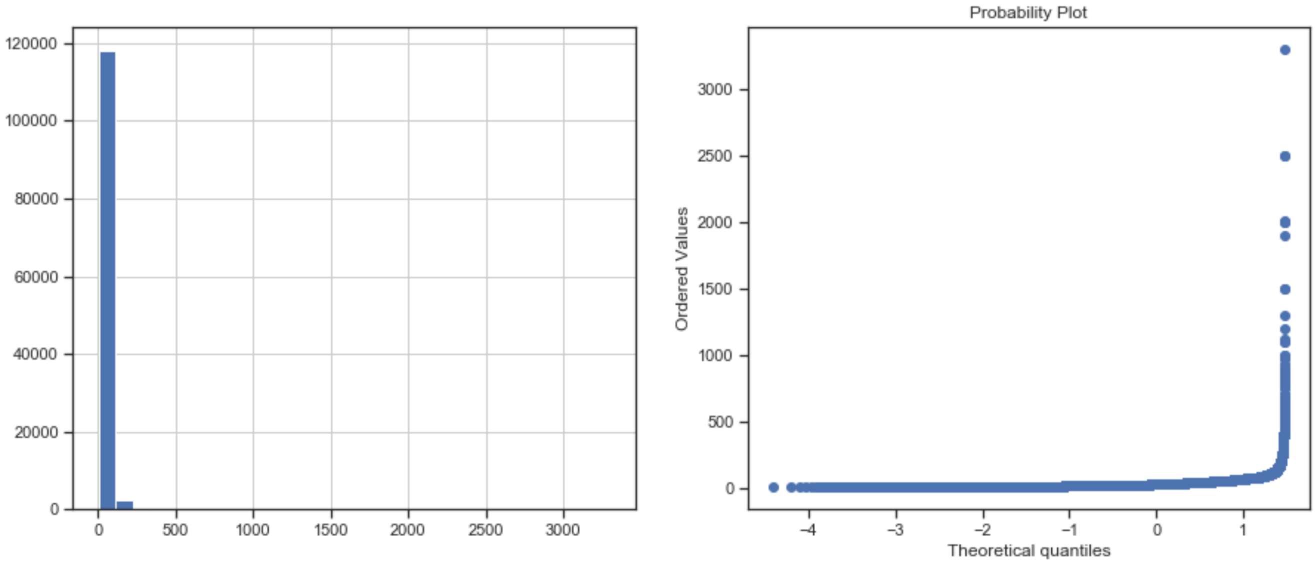
```python
data['points_log'] = np.log(data['points'])
diagnostic_plots(data, 'points_log')
```





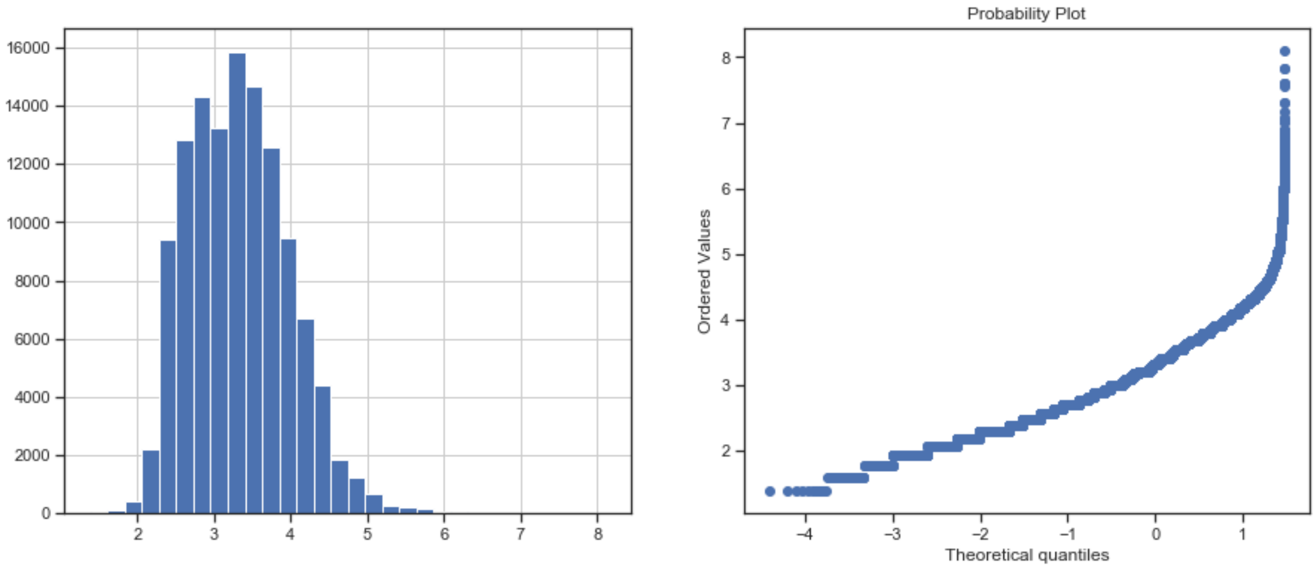
In [91]:

```
diagnostic_plots(data, 'price')
```



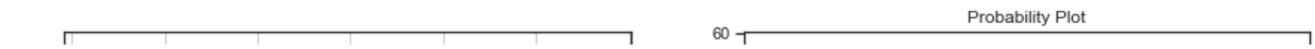
In [92]:

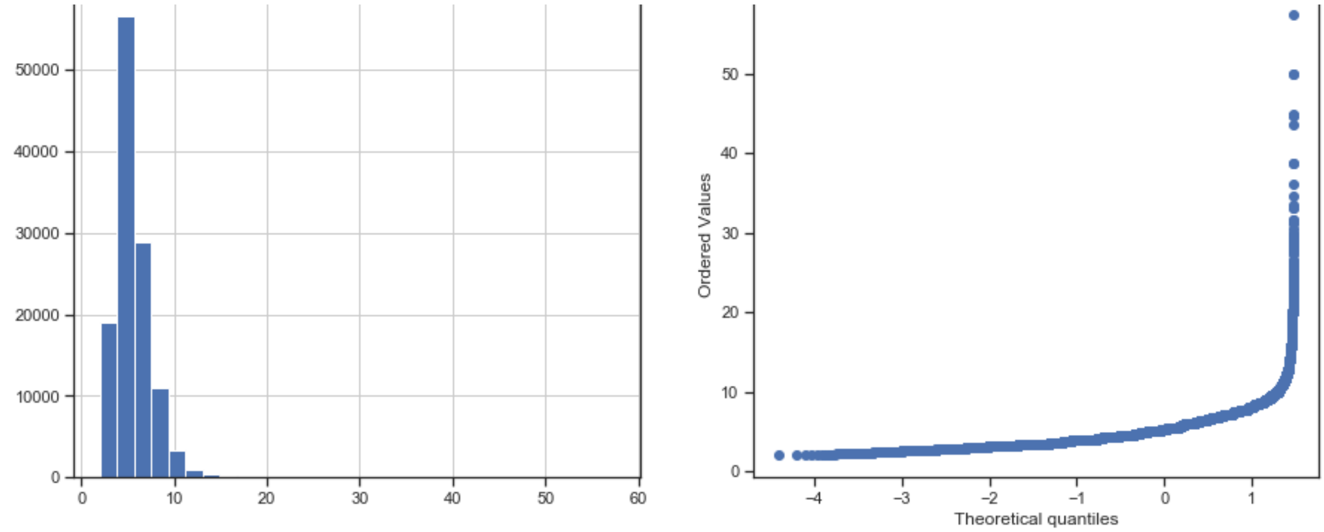
```
data['price_log'] = np.log(data['price'])
diagnostic_plots(data, 'price_log')
```



In [95]:

```
data['price_sqr'] = data['price']**(1/2)
diagnostic_plots(data, 'price_sqr')
```



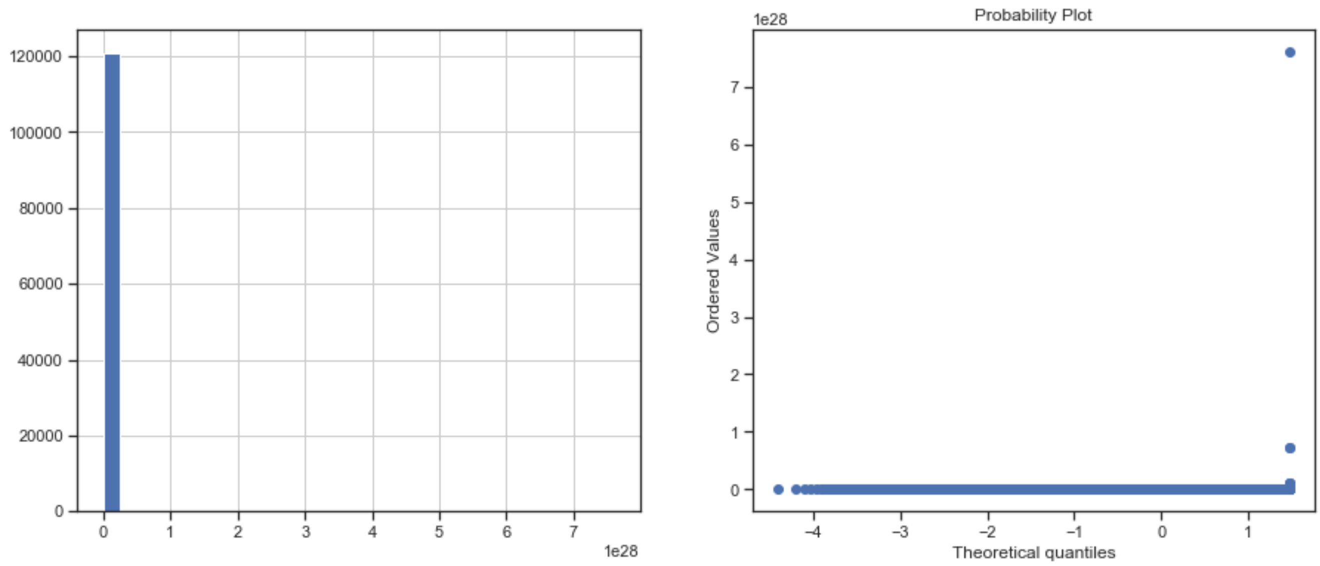


In [93]:

```
data['price_boxcox'], param = stats.boxcox(data['price'])
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(data, 'price_boxcox')
```

```
//anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1044: RuntimeWarning: invalid value en
countered in less_equal
  if any(x <= 0):
```

```
Оптимальное значение λ = 8.472135811722177
```

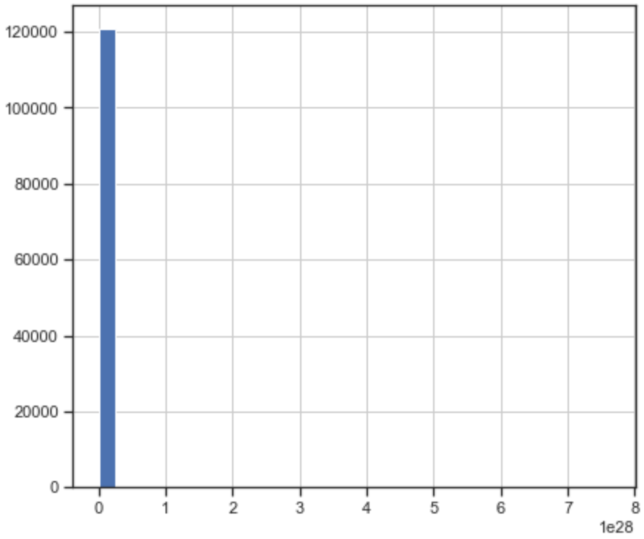


In [94]:

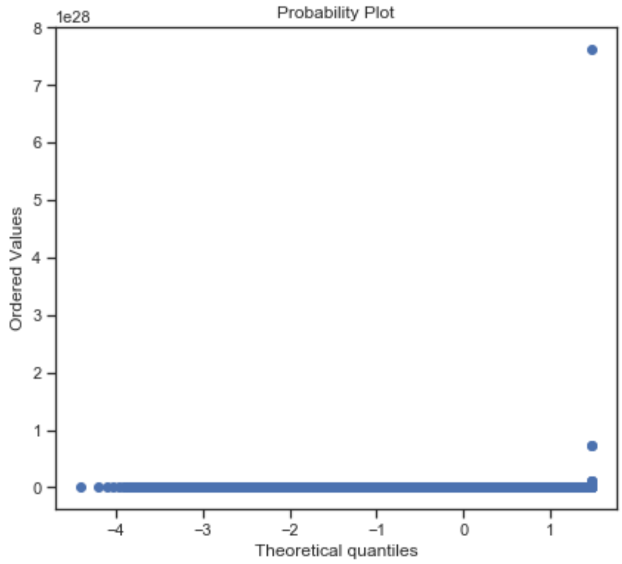
```
data['price'] = data['price'].astype('float')
data['price_yeojohnson'], param = stats.yeojohnson(data['price'])
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(data, 'price_yeojohnson')
```

```
//anaconda3/lib/python3.7/site-packages/scipy/stats/morestats.py:1371: RuntimeWarning: invalid value en
countered in greater_equal
  pos = x >= 0  # binary mask
```

```
Оптимальное значение λ = 8.472135811722177
```



120000
100000
80000
60000
40000
20000
0
0
1
2
3
4
5
6
7
8
1e28



Probability Plot
1e28
Ordered Values
Theoretical quantiles
8
7
6
5
4
3
2
1
0
−4
−3
−2
−1
0
1


```
In [ ]:
```