```
import numpy as np
import pandas as pd

```

```
!unzip /content/drive/MyDrive/Colab_data/MMO/archive.zip
```

```
Archive:  /content/drive/MyDrive/Colab_data/MMO/archive.zip
  inflating: autoru_total.csv
```

```
data = pd.read_csv('/content/autoru_total.csv', sep=',')
```

```
data.shape
```

```
(36906, 17)
```

```
data.head()
```

	Model	Year	Mileage	V_engine	EngineType	HorsePower	Tax	State	
0	Subaru Forester IV	2013	83800.0	2.5	Бензин	171.0	8379.0	Не требует ремонта	вла,
1	Opel Zafira B Рестайлинг	2014	97265.0	1.8	Бензин	140.0	4900.0	Не требует ремонта	вл
2	Kia Rio IV	2017	48000.0	1.6	Бензин	123.0	3075.0	Не требует ремонта	вла,
3	Skoda Octavia II (A5) Рестайлинг	2011	292000.0	1.4	Бензин	122.0	4148.0	Не требует ремонта	вла,
4	Audi A6 IV (C7) Рестайлинг	2015	106205.0	1.8	Бензин	190.0	9500.0	Не требует ремонта	вла,

```
hcols_with_na = [c for c in data.columns if data[c].isnull().sum() > 0]
hcols_with_na
```

```
['Tax', 'State', 'Owners', 'Customs', 'Accidents']
```

```
[(c, data[c].isnull().sum()) for c in hcols_with_na]
```

```
[('Tax', 476),
```

```
 ('State', 4),
 ('Owners', 2),
 ('Customs', 1),
 ('Accidents', 28527)]
```

```
# Доля (процент) пропусков
[(c, data[c].isnull().mean()) for c in hcols_with_na]

```

```
[('Tax', 0.012897631821384056),
 ('State', 0.00010838346068389963),
 ('Owners', 5.419173034194982e-05),
 ('Customs', 2.709586517097491e-05),
 ('Accidents', 0.7729637457324012)]
```

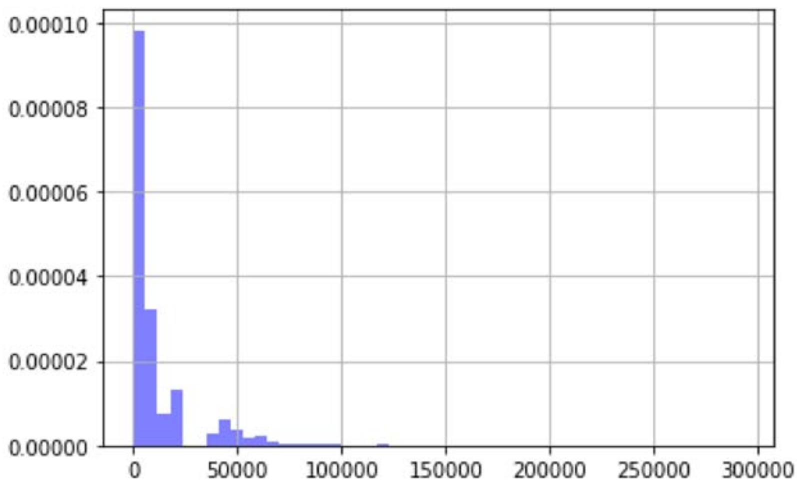
Удаление пропущенных значений

```

data_drop = data['Customs'].dropna()
data_drop.shape
```

```
(36905,)
```

Заполнение показателями центра распределения

```
import matplotlib.pyplot as plt
%matplotlib inline

fig = plt.figure()
ax = fig.add_subplot(111)
data['Tax'].hist(bins=50, ax=ax, color='blue', density=True, alpha=0.5)
plt.show()
```




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

indicator = MissingIndicator()
```

```
mask_missing_values_only = indicator.fit_transform(data[['Tax']].values)

imputer = SimpleImputer(strategy='mean')
all_data = imputer.fit_transform(data[['Tax']].values)
all_data
```

```
array([[8379.],
       [4900.],
       [3075.],
       ...,
       [4760.],
       [1176.],
       [9500.]])
```

Заполнение наиболее распространенным значением категории

```
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data[['State']].values)

imputer = SimpleImputer(strategy='most_frequent')
all_data = imputer.fit_transform(data[['State']].values)
filled_data = all_data[mask_missing_values_only]

print(filled_data)
all_data
```

```
['Не требует ремонта' 'Не требует ремонта' 'Не требует ремонта'
 'Не требует ремонта']
array([['Не требует ремонта'],
       ['Не требует ремонта'],
       ['Не требует ремонта'],
       ...,
       ['Не требует ремонта'],
       ['Не требует ремонта'],
       ['Не требует ремонта']], dtype=object)
```

введение отдельного значения категории для пропущенных значений

```
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data[['Accidents']].values)

imputer = SimpleImputer(strategy='constant', fill_value='NA')
all_data = imputer.fit_transform(data[['Accidents']].values)
all_data
```

```
array([['ДТП ненайдены'],
       ['NA'],
       ['NA'],
       ...,
       ['NA'],
       ['NA'],
       ['ДТП ненайдены']], dtype=object)
```

Добавление флагов пропусков

```
hdata_mis = data[['Tax']].copy()
indicator = MissingIndicator()
Tax_missing = indicator.fit_transform(hdata_mis[['Tax']])
Tax_missing
```

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

```
hdata_mis['flag'] = Tax_missing
hdata_mis.head()
```

	Tax	flag
0	8379.0	False
1	4900.0	False
2	3075.0	False
3	4148.0	False
4	9500.0	False

Кодирование категориальных признаков

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
cat_enc_le = le.fit_transform(data['Model'])
```

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

```
array(['Subaru Forester IV', 'Opel Zafira B Рестайлинг', 'Kia Rio IV',
       ..., 'Mitsubishi Chariot II', 'Porsche 911 Carrera VI (997)',
       'BMW 3 серии Gran Turismo 335i VI (F3x)'], dtype=object)
```

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

```
array([   0,    1,    2, ..., 3160, 3161, 3162])
```

```
le.inverse_transform(np.unique(cat_enc_le))
```

```
array(['Audi 100 III (C3)', 'Audi 100 III (C3) Рестайлинг',
```

```
'Audi 100 IV (C4)', ..., 'Volkswagen Type 1',
'Volkswagen Type 2 T2', 'Volkswagen Vento'], dtype=object)
```

Кодирование категорий наборами бинарных значений - one-hot encoding

```
from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(data[['Model']])
cat_enc_ohe
```

```
<36906x3163 sparse matrix of type '<class 'numpy.float64'>'
        with 36906 stored elements in Compressed Sparse Row format>
```

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

```
matrix([[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.]])
```

```
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.4.0-py2.py3-none-any.whl (86 kB)
     |████████████████████████████████| 86 kB 3.5 MB/s
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pandas>=0.21.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/di
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from p
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-package
Installing collected packages: category-encoders
Successfully installed category-encoders-2.4.0
```

Count (frequency) encoding

```
from category_encoders.count import CountEncoder as ce_CountEncoder

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

data_COUNT_ENC
```

	Accidents	CarBodyType	Color	Customs	Drive	EngineType	HorsePower	Mileag
0	8379	13303	3941	36905	14584	30981	171.0	83800
1	28527	946	9789	36905	20236	30981	140.0	97265
2	28527	11985	3594	36905	20236	30981	123.0	48000
3	28527	1870	3941	36905	20236	30981	122.0	292000
4	28527	11985	9789	36905	20236	30981	190.0	106205
...	...	...	...	...	...	...	...	
36901	28527	11985	7264	36905	20236	30981	102.0	133789
36902	28527	13303	9789	36905	14584	5138	249.0	103965
36903	28527	11985	7264	36905	2086	30981	136.0	108000
36904	28527	11985	9789	36905	20236	30981	98.0	259000
36905	8379	11985	5173	36905	20236	30981	190.0	265000

36906 rows × 16 columns

Target (Mean) encoding

```
from category_encoders.target_encoder import TargetEncoder as ce_TargetEncoder

ce_TargetEncoder1 = ce_TargetEncoder()
data_MEAN_ENC = ce_TargetEncoder1.fit_transform(data[data.columns.difference(['Tax'])],
data_MEAN_ENC
```

	Accidents	CarBodyType	Color	Customs	Drive	EngineT
0	12194.545894	17928.761822	7368.860160	12357.519229	22441.953945	12086.564
1	12405.048242	3726.181818	18208.519405	12357.519229	4326.840010	12086.564
2	12405.048242	9945.035446	11237.700536	12357.519229	4326.840010	12086.564
4	12405.048242	9945.035446	18208.519405	12357.519229	4326.840010	12086.564

Нормализация числовых признаков

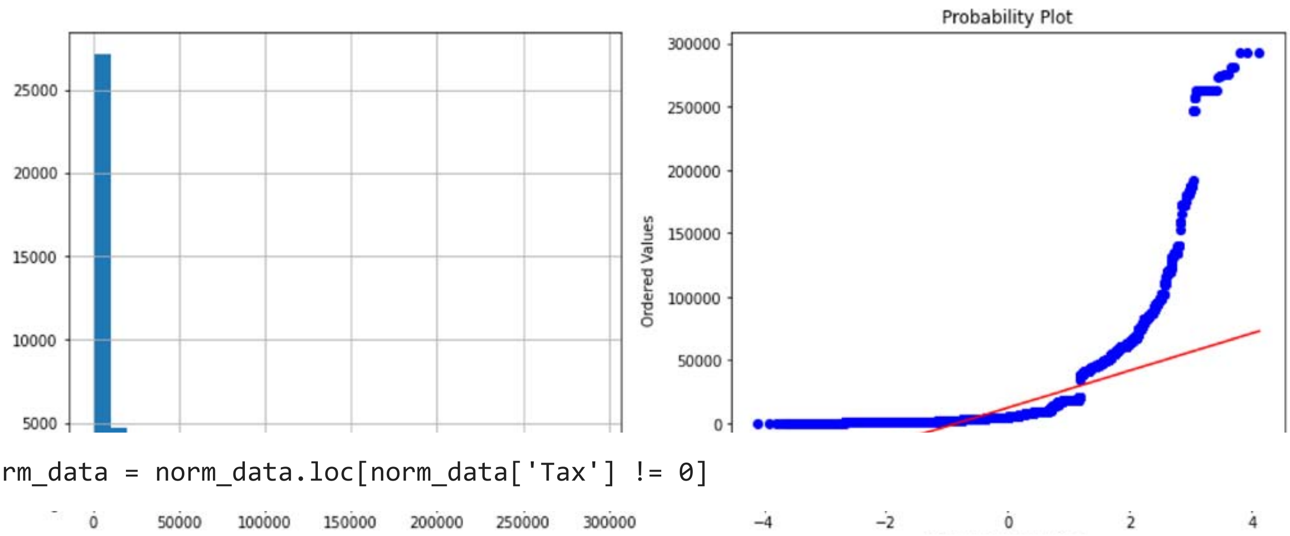
```
import scipy.stats as stats

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()
```

```
norm_data = data[['Tax']].dropna()
norm_data
```

	Tax
0	8379.0
1	4900.0
2	3075.0
3	4148.0
4	9500.0
...	...
36901	2550.0
36902	18675.0
36903	4760.0
36904	1176.0
36905	9500.0

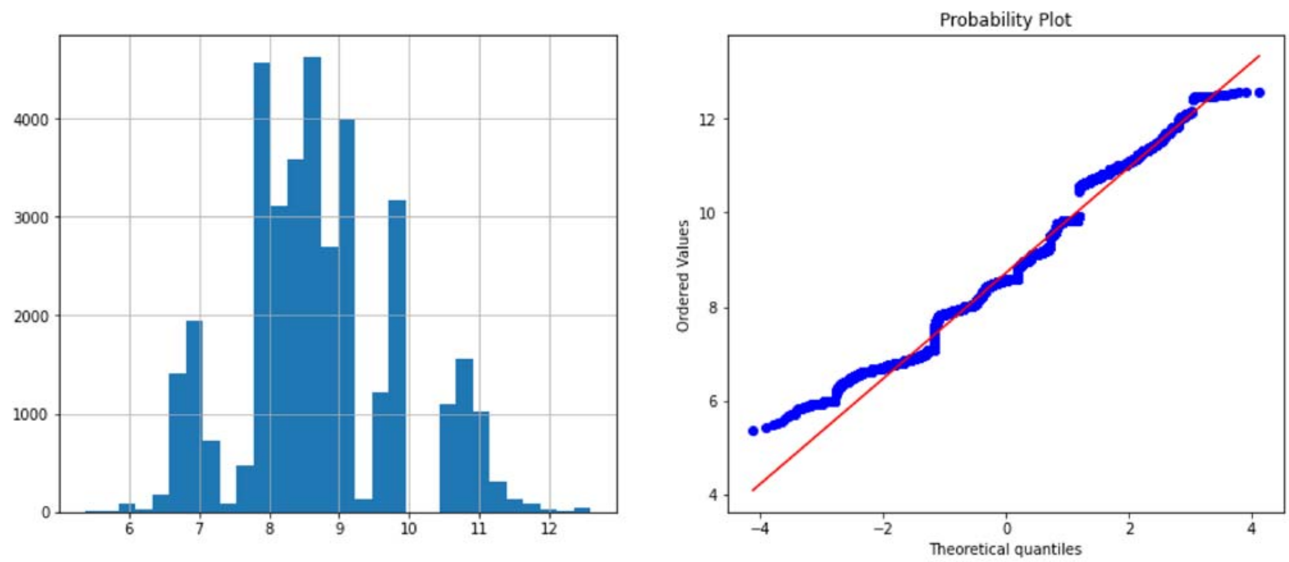
36430 rows × 1 columns

```
diagnostic_plots(norm_data, 'Tax')
```

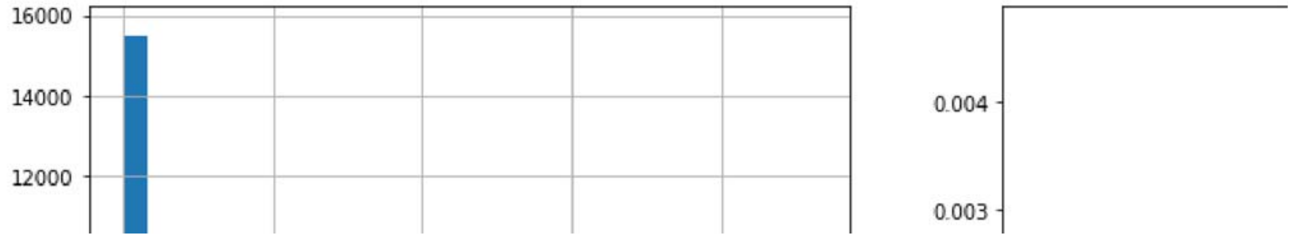



```
norm_data = norm_data.loc[norm_data['Tax'] != 0]
```

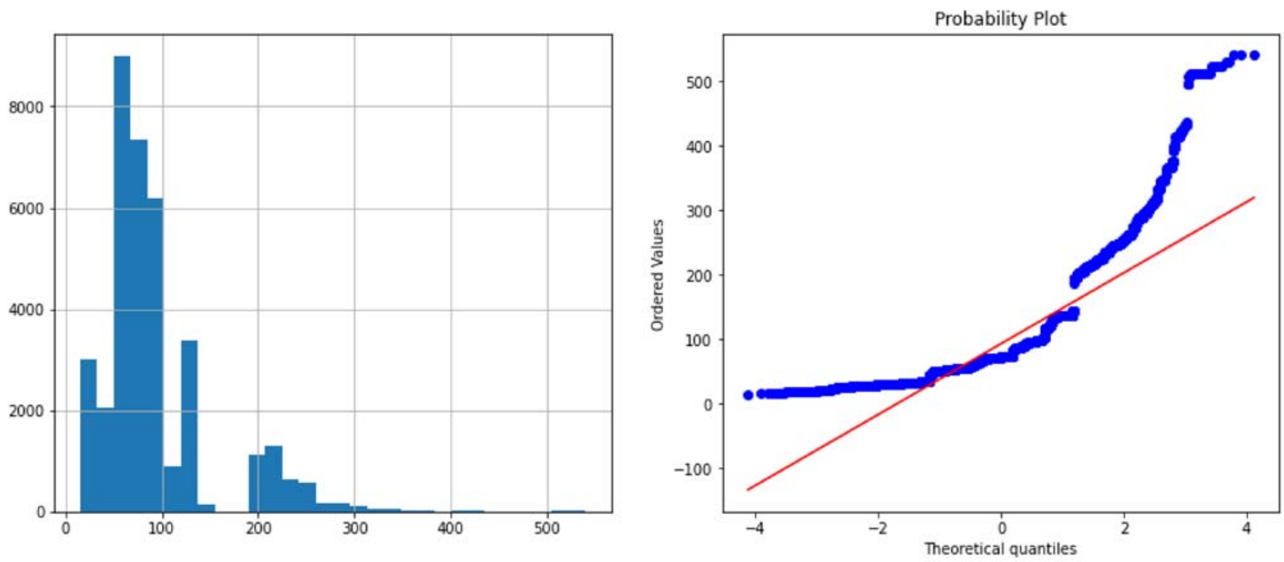
```
norm_data['Tax_log'] = np.log(norm_data['Tax'])


norm_data['Tax_log'].min()
diagnostic_plots(norm_data, 'Tax_log')
```




```
norm_data['reciprocal'] = 1 / (norm_data['Tax'])
diagnostic_plots(norm_data, 'reciprocal')
```

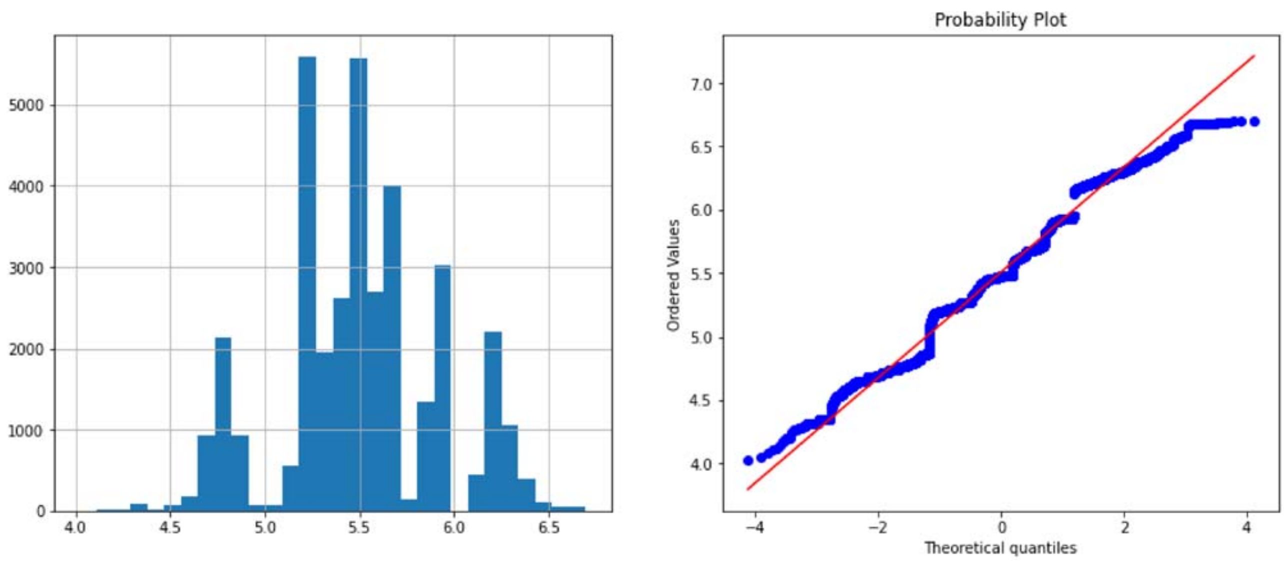


```
norm_data['sqr'] = norm_data['Tax']**(1/2)
diagnostic_plots(norm_data, 'sqr')
```

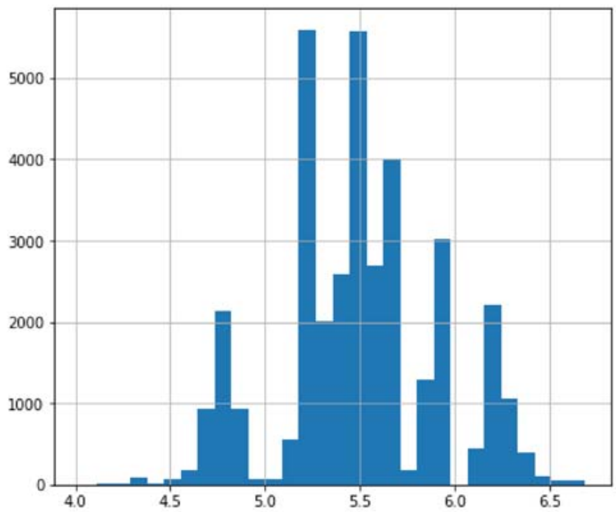


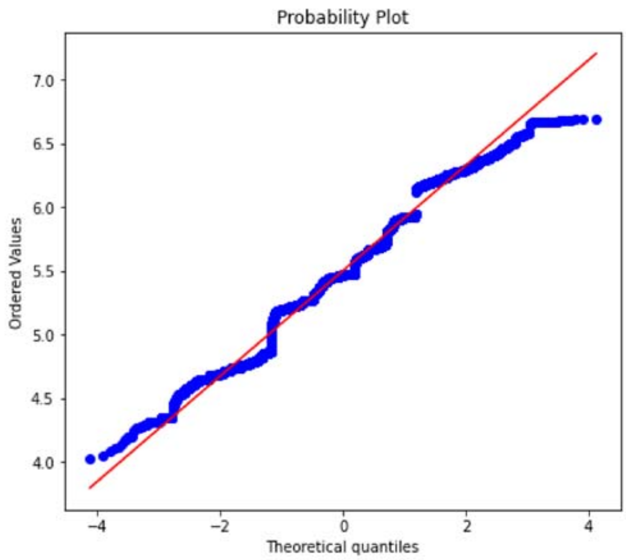

```
norm_data['boxcox'], param = stats.boxcox(norm_data['Tax'])
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(norm_data, 'boxcox')
```

```
Оптимальное значение λ = -0.11353789281093771
```




```
norm_data['Tax'] = norm_data['Tax'].astype('float')
norm_data['yeojohnson'], param = stats.yeojohnson(norm_data['Tax'])
diagnostic_plots(norm_data, 'yeojohnson')
```





1 сек. выполнено в 23:31