0.1 ММО Сысойкин Егор ИУ5-21М РК1

0.1.1 Вариант 15

Задача 15

Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции “возведение в степень”.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
```

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

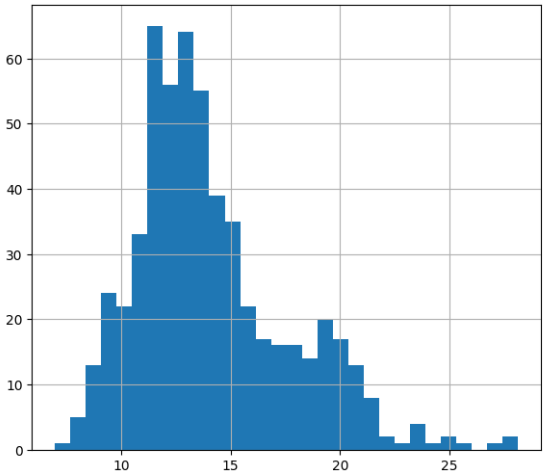
[3]:
```
data.head()
```

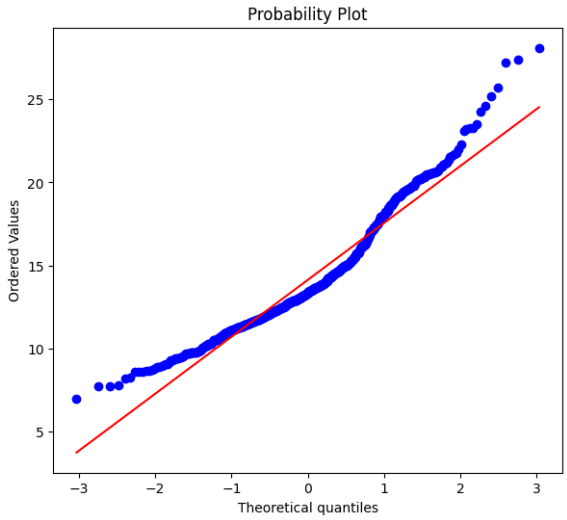
[3]:
```
         Id Diagnosis  Radius (mean)  Texture (mean)  Perimeter (mean) 
↪\
0   8510426         B         13.540           14.36             87.46
1   8510653         B         13.080           15.71             85.63
2   8510824         B          9.504           12.44             60.34
3    854941         B         13.030           18.42             82.61
4  85713702         B          8.196           16.84             51.71

   Area (mean)  Smoothness (mean)  Compactness (mean)  Concavity (mean)
↪ \
0        566.3            0.09779             0.08129           0.06664
1        520.0            0.10750             0.12700           0.04568
2        273.9            0.10240             0.06492           0.02956
3        523.8            0.08983             0.03766           0.02562
4        201.9            0.08600             0.05943           0.01588

   Concave points (mean)  …  Radius (worst)  Texture (worst)  \
0               0.047810  …          15.110            19.26
1               0.031100  …          14.500            20.49
2               0.020760  …          10.230            15.66
3               0.029230  …          13.300            22.81
4               0.005917  …           8.964            21.96

   Perimeter (worst)  Area (worst)  Smoothness (worst)  Compactness
↪(worst)  \
0              99.70         711.2             0.14400              0.
↪17730
```

```
1              96.09         630.5              0.13120              0.
↪27760
2              65.13         314.9              0.13240              0.
↪11480
3              84.46         545.9              0.09701              0.
↪04619
4              57.26         242.2              0.12970              0.
↪13570

   Concavity (worst)  Concave points (worst)  Symmetry (worst)  \
0            0.23900                 0.12880            0.2977
1            0.18900                 0.07283            0.3184
2            0.08867                 0.06227            0.2450
3            0.04833                 0.05013            0.1987
4            0.06880                 0.02564            0.3105

   Fractal dimension (worst)
0                    0.07259
1                    0.08183
2                    0.07773
3                    0.06169
4                    0.07409

[5 rows x 32 columns]
```

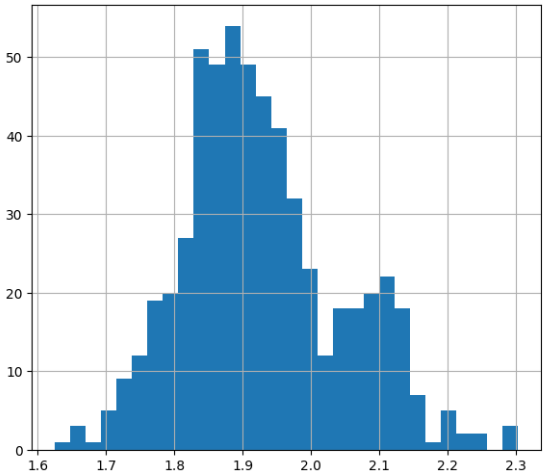
[4]:
```
def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    # гистограмма
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

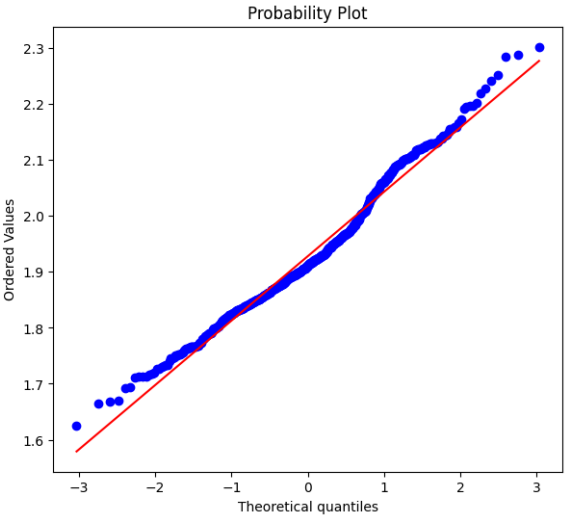
[5]:
```
diagnostic_plots(data,"Radius (mean)")
```





[6]:
```
data['Radius_my'] = data['Radius (mean)']**(1/4)
diagnostic_plots(data, 'Radius_my')
```





Задача 35

Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод вложений (embedded method). Используйте подход на основе дерева решений.

[19]:
```
data = pd.read_csv('insurance.csv', sep=",")
data.head()
```

[19]:
```
   age     sex     bmi  children smoker     region      charges
0   19  female  27.900         0    yes  southwest  16884.92400
1   18    male  33.770         1     no  southeast   1725.55230
2   28    male  33.000         3     no  southeast   4449.46200
3   33    male  22.705         0     no  northwest  21984.47061
4   32    male  28.880         0     no  northwest   3866.85520
```

[20]:
```
data["smoker_enc"] = data.apply(lambda x: 1 if x['smoker'] == 'yes'
  ↪else 0, axis=1)
data["sex_enc"] = data.apply(lambda x: 1 if x['sex'] == 'male' else 0,
  ↪axis=1)
```

[21]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['region_enc'] = le.fit_transform(data['region'])
data.head()
```

[21]:
```
   age     sex     bmi  children smoker     region      charges
  ↪smoker_enc  \
0   19  female  27.900         0    yes  southwest  16884.92400
  ↪  1
1   18    male  33.770         1     no  southeast   1725.55230
  ↪  0
2   28    male  33.000         3     no  southeast   4449.46200
  ↪  0
3   33    male  22.705         0     no  northwest  21984.47061
  ↪  0
4   32    male  28.880         0     no  northwest   3866.85520
  ↪  0

   sex_enc  region_enc
0        0           3
1        1           2
2        1           2
3        1           1
4        1           1
```

[22]:
```
data = data.drop(columns=['sex', 'smoker', 'region'])
```

[23]:
```
data.head()
```

[23]:
```
   age     bmi  children      charges  smoker_enc  sex_enc  region_enc
0   19  27.900         0  16884.92400           1        0           3
1   18  33.770         1   1725.55230           0        1           2
2   28  33.000         3   4449.46200           0        1           2
3   33  22.705         0  21984.47061           0        1           1
```

```
4   32  28.880         0   3866.85520           0        1           1
```

[24]:
```
data.dtypes
```

[24]:
```
age             int64
bmi           float64
children        int64
charges       float64
smoker_enc      int64
sex_enc         int64
region_enc      int64
dtype: object
```

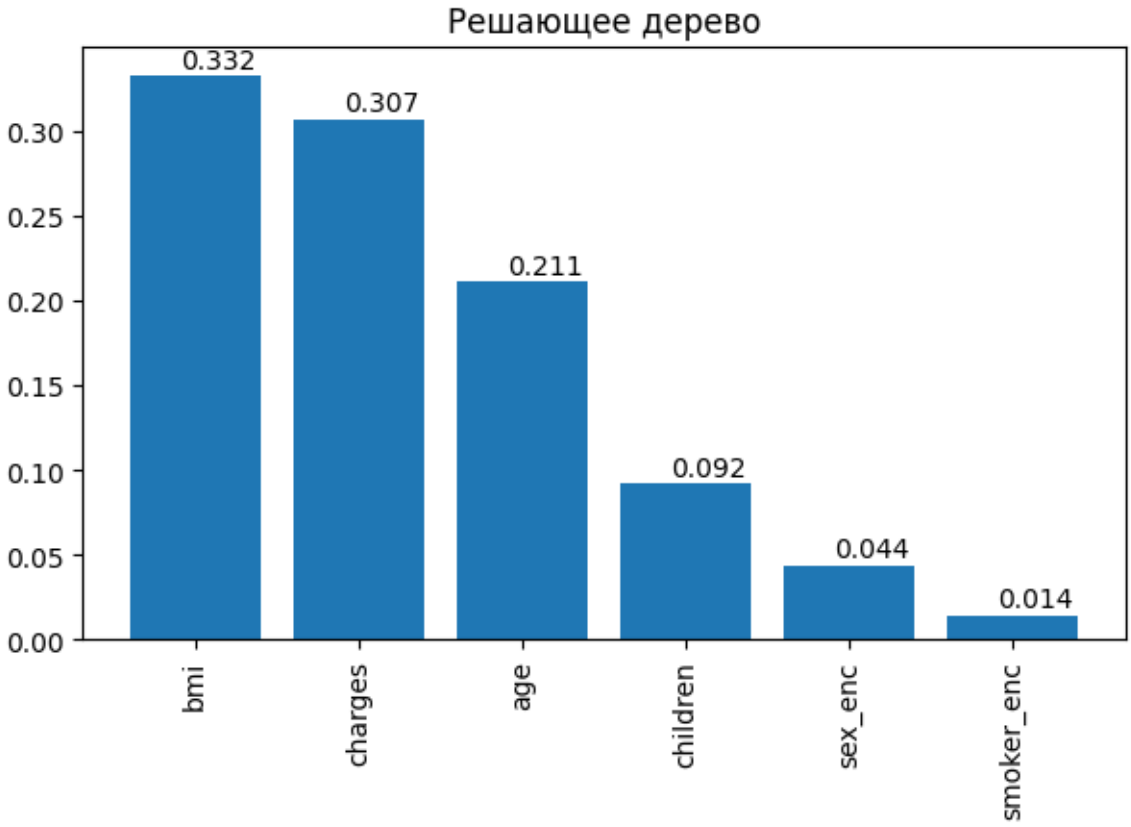
[25]:
```
dataX = data[['age', 'bmi', 'children', 'charges', 'smoker_enc',
  ↪'sex_enc']]
dataY = data[['region_enc']]
```

[26]:
```
dtc1 = DecisionTreeClassifier()
dtc1.fit(dataX, dataY)

# Важность признаков
dtc1.feature_importances_, sum(dtc1.feature_importances_)
```

[26]:
```
(array([0.21080699, 0.33234935, 0.09214304, 0.30677273, 0.01436694,
        0.04356094]),
 0.9999999999999999)
```

[27]:
```
from operator import itemgetter

def draw_feature_importances(tree_model, X_dataset, title,
  ↪figsize=(7,4)):
    """
    Вывод важности признаков в виде графика
    """
    # Сортировка значений важности признаков по убыванию
    list_to_sort = list(zip(X_dataset.columns.values, tree_model.
  ↪feature_importances_))
    sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse =
  ↪True)
    # Названия признаков
    labels = [x for x,_ in sorted_list]
    # Важности признаков
    data = [x for _,x in sorted_list]
    # Вывод графика
    fig, ax = plt.subplots(figsize=figsize)
    ax.set_title(title)
    ind = np.arange(len(labels))
```

```
    plt.bar(ind, data)
    plt.xticks(ind, labels, rotation='vertical')
    # Вывод значений
    for a,b in zip(ind, data):
        plt.text(a-0.1, b+0.005, str(round(b,3)))
    plt.show()
    return labels, data

_,_=draw_feature_importances(dtc1, dataX, 'Решающее дерево')
```

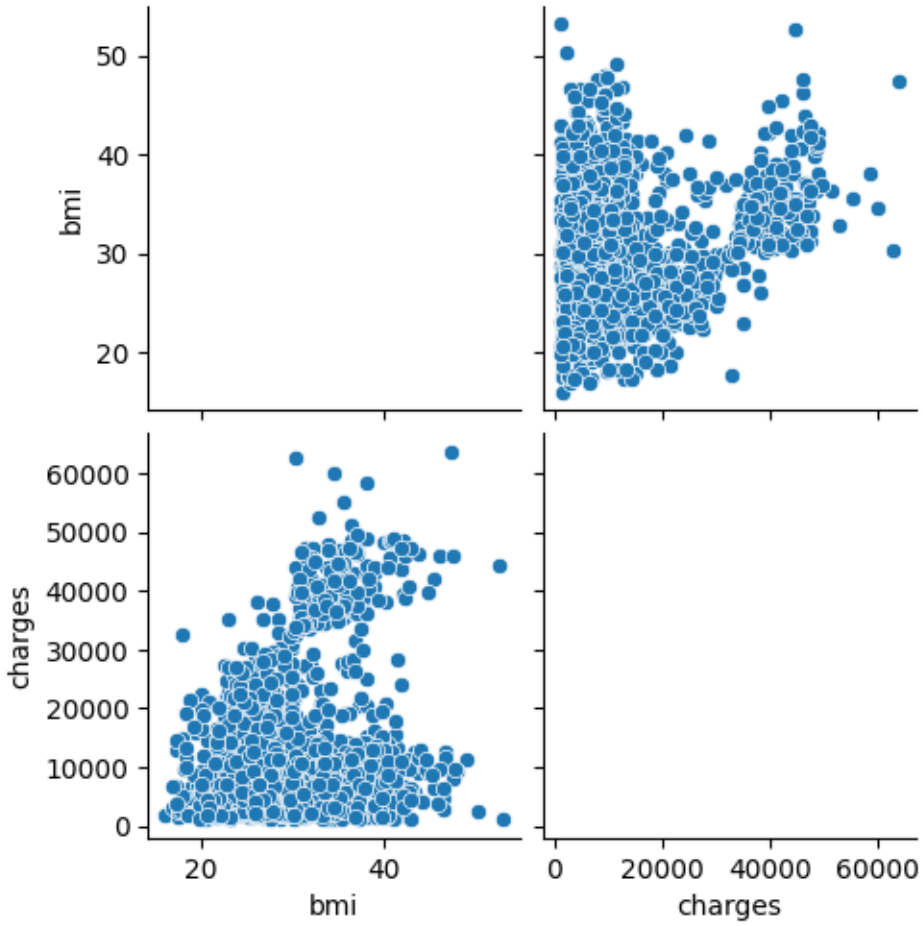


Дополнительное задание

для пары произвольных колонок данных построить график “Диаграмма рассеяния”.

[29]:
```
sns.pairplot(data=data, vars=['bmi', 'charges'], diag_kind="scatter")
```

[29]:
```
<seaborn.axisgrid.PairGrid at 0x7fca5445e6d0>
```


bmi
50
40
30
20
charges
60000
50000
40000
30000
20000
10000
0
20
40
bmi
0
20000
40000
60000
charges

[ ]: