Уткин Д.Ю. ИУ5-23М

Вариант 14

> ***Задача №14***.

Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции "квадратный корень".

> ***Задача №34***

Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод вложений (embedded method). Используйте подход на основе линейной или логистической регрессии (в зависимости от того, на решение какой задачи ориентирован выбранный Вами набор данных - задачи регрессии или задачи классификации).

> Дополнительные требования

Для студентов групп ИУ5-23М, ИУ5И-23М - для произвольной колонки данных построить график "Ящик с усами (boxplot)".

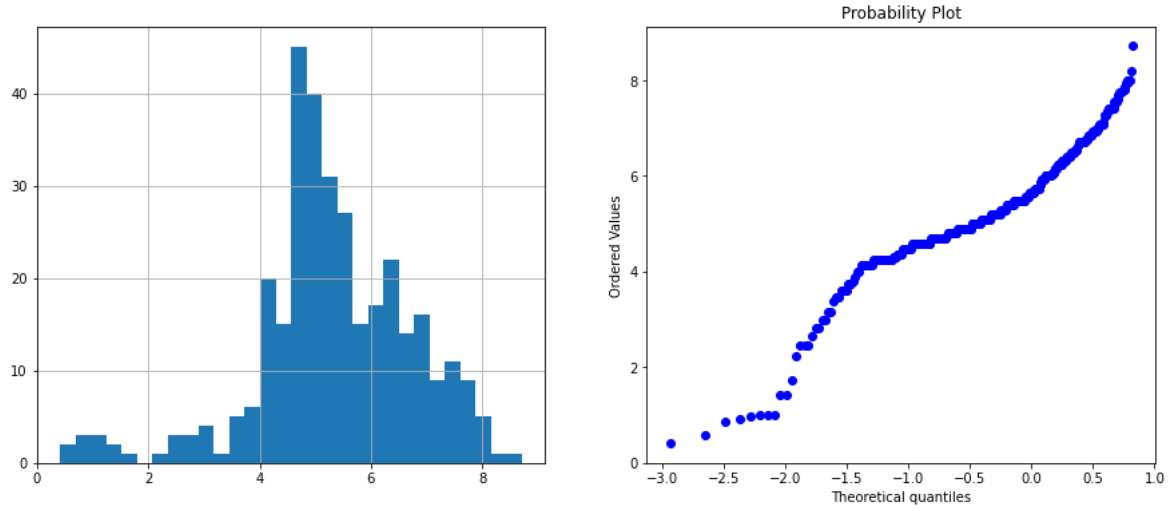
Задача 14

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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()
```

```
data = pd.read_csv('tested.csv', sep=",")
```

```
data['Age_sqr'] = data['Age']**(1/2)
diagnostic_plots(data, 'Age_sqr')
```



Задача 34

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
import scipy.stats as stats
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 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
from IPython.display import Image
%matplotlib inline
sns.set(style="ticks")
cancer = load_breast_cancer()
X = cancer.data
y = cancer.target
```

Использование линейных моделей

```
# Используем L1-регуляризацию
e_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1', max_iter=500, random_state=1)
e_lr1.fit(X, y)
# Коэффициенты регрессии
e_lr1.coef_
```

```
array([[ 1.74582528e+00, -1.14032407e-02,  2.42531641e-01,
         9.68430706e-04, -4.63739886e+01,  1.04718073e+02,
        -6.90696671e+01, -1.40084043e+02,  4.28511895e+01,
        -4.06669723e+01, -1.14319188e+01,  2.53858364e+00,
         1.32274835e+00, -1.65993422e-01,  0.00000000e+00,
         0.00000000e+00,  1.57800789e+02, -8.19875799e+02,
         9.12984724e+01,  6.99399379e+02,  3.60585749e-01,
        -5.85051521e-01, -3.75583010e-02, -4.12134482e-02,
        -5.65041836e+00,  2.93273026e+00, -1.46252144e+01,
         0.00000000e+00, -4.17321012e+01, -1.17433685e+02]])
```

```
# Все 4 признака являются "хорошими"
sel_e_lr1 = SelectFromModel(e_lr1)
sel_e_lr1.fit(X, y)
sel_e_lr1.get_support()
```

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

Линейный классификатор на основе SVM

```
e_lr2 = LinearSVC(C=0.01, penalty="l1", max_iter=2000, dual=False)
e_lr2.fit(X, y)
# Коэффициенты регрессии
e_lr2.coef_
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the nu
  warnings.warn(
array([[ 0.        ,  0.        ,  0.07168777,  0.00106387,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        , -0.00965706,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        , -0.02612847, -0.00148275, -0.00742943,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ]])
```

```
# Признак 4 исключен
sel_e_lr2 = SelectFromModel(e_lr2)
sel_e_lr2.fit(X, y)
sel_e_lr2.get_support()
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the nu
  warnings.warn(
array([False, False,  True,  True, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
```

```
       False, False, False,  True,  True,  True, False, False, False,
       False, False, False])
```

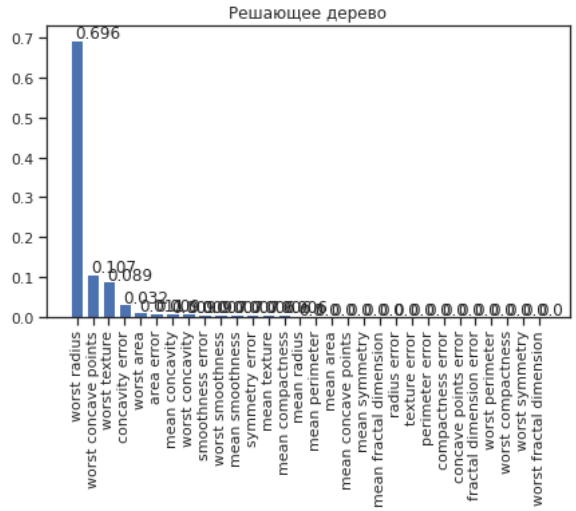
Использование моделей на основе решающего дерева

> Задача классификации

```
dtc1 = DecisionTreeClassifier()
rfc1 = RandomForestClassifier()
gbc1 = GradientBoostingClassifier()
dtc1.fit(X, y)
rfc1.fit(X, y)
gbc1.fit(X, y)

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

```
(array([0.        , 0.00563858, 0.        , 0.        , 0.00701689,
        0.00563858, 0.00877112, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.00936121, 0.00744793,
        0.        , 0.03184137, 0.        , 0.00583135, 0.        ,
        0.69559352, 0.08856128, 0.        , 0.0110859 , 0.00738798,
        0.        , 0.00872741, 0.10709688, 0.        , 0.        ]), 1.0)
```

```
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

cancer_feature_names = cancer['feature_names']
cancer_x_df = pd.DataFrame(data=cancer['data'], columns=cancer['feature_names'])
_,_=draw_feature_importances(dtc1, cancer_x_df, 'Решающее дерево')
```

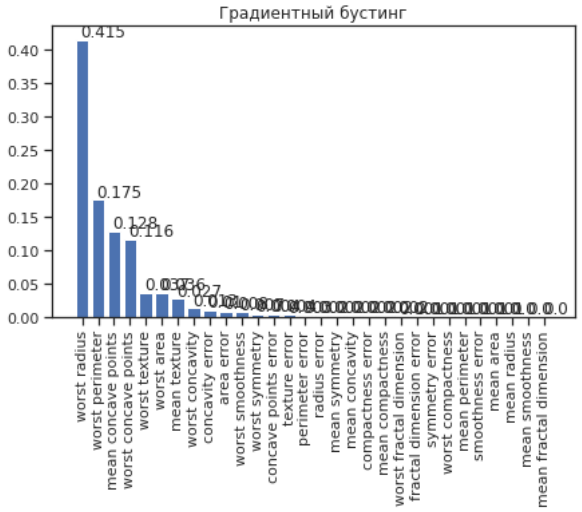



```
list(zip(cancer_feature_names, SelectFromModel(dtc1).fit(X, y).get_support()))
```

```
[('mean radius', False),
 ('mean texture', True),
```

```
 ('mean perimeter', False),
 ('mean area', False),
 ('mean smoothness', False),
 ('mean compactness', False),
 ('mean concavity', False),
 ('mean concave points', False),
 ('mean symmetry', False),
 ('mean fractal dimension', False),
 ('radius error', False),
 ('texture error', False),
 ('perimeter error', False),
 ('area error', False),
 ('smoothness error', False),
 ('compactness error', False),
 ('concavity error', False),
 ('concave points error', False),
 ('symmetry error', False),
 ('fractal dimension error', False),
 ('worst radius', True),
 ('worst texture', True),
 ('worst perimeter', False),
 ('worst area', False),
 ('worst smoothness', False),
 ('worst compactness', False),
 ('worst concavity', False),
 ('worst concave points', True),
 ('worst symmetry', False),
 ('worst fractal dimension', False)]
```

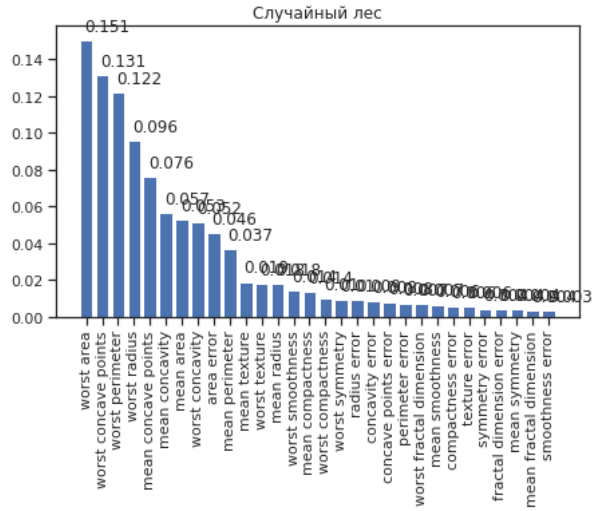
```
_,_=draw_feature_importances(gbc1, cancer_x_df, 'Градиентный бустинг')
```




```
list(zip(cancer_feature_names, SelectFromModel(gbc1).fit(X, y).get_support()))
```

```
[('mean radius', False),
 ('mean texture', False),
 ('mean perimeter', False),
 ('mean area', False),
 ('mean smoothness', False),
 ('mean compactness', False),
 ('mean concavity', False),
 ('mean concave points', True),
 ('mean symmetry', False),
 ('mean fractal dimension', False),
 ('radius error', False),
 ('texture error', False),
 ('perimeter error', False),
 ('area error', False),
 ('smoothness error', False),
 ('compactness error', False),
 ('concavity error', False),
 ('concave points error', False),
 ('symmetry error', False),
 ('fractal dimension error', False),
 ('worst radius', True),
 ('worst texture', True),
 ('worst perimeter', True),
 ('worst area', False),
 ('worst smoothness', False),
 ('worst compactness', False),
 ('worst concavity', False),
 ('worst concave points', True),
 ('worst symmetry', False),
 ('worst fractal dimension', False)]
```

```
_,_=draw_feature_importances(rfc1, cancer_x_df, 'Случайный лес')
```




```
list(zip(cancer_feature_names, SelectFromModel(rfc1).fit(X, y).get_support()))
```

```
[('mean radius', True),
 ('mean texture', False),
 ('mean perimeter', True),
 ('mean area', True),
 ('mean smoothness', False),
 ('mean compactness', False),
 ('mean concavity', True),
 ('mean concave points', True),
 ('mean symmetry', False),
 ('mean fractal dimension', False),
 ('radius error', False),
 ('texture error', False),
 ('perimeter error', False),
 ('area error', True),
 ('smoothness error', False),
 ('compactness error', False),
 ('concavity error', False),
 ('concave points error', False),
 ('symmetry error', False),
 ('fractal dimension error', False),
 ('worst radius', True),
 ('worst texture', False),
 ('worst perimeter', True),
 ('worst area', True),
 ('worst smoothness', False),
 ('worst compactness', False),
 ('worst concavity', True),
 ('worst concave points', True),
 ('worst symmetry', False),
 ('worst fractal dimension', False)]
```

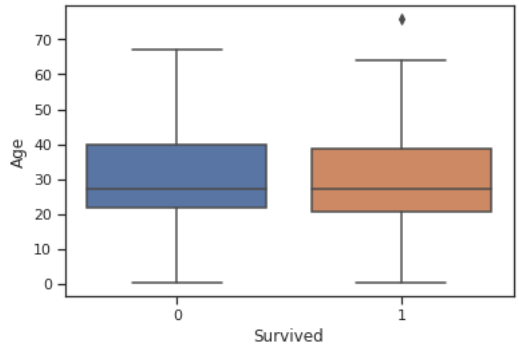
Дополнительные требования

```
# подключение библиотек
import pandas as pd
import random
import matplotlib.pyplot as plt
import pandas.plotting
import seaborn as sb
import math as math
%matplotlib inline
my_file = open("tested.csv","r")
dataset = pd.read_csv(my_file)
dataset
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...

```
#График показывает прямоугольную диаграмму распределения возраста для двух групп.
sb.boxplot( x=dataset["Survived"], y=dataset["Age"] )
```

```
<Axes: xlabel='Survived', ylabel='Age'>
```



✓ 0 сек. выполнено в 17:53