Лабораторная работа №9. Решение задач машинного обучения с помощью логических методов классификации.

Используемый набор данных: [Breast Cancer Wisconsin (Prognostic) (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Prognostic%29)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Prognostic%29)

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, roc_curve, roc_auc_score
import os
import requests

%matplotlib inline

pd.options.display.max_columns = None
```

In [2]:

```
def downloadFile(url, filePath):
    if not os.path.exists(filePath):
        req = requests.get(url)
        f = open(filePath, "wb")
        f.write(req.content)
        f.close

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsi
n"
downloadFile(url + "/wpbc.data", "dataset/wpbc.data")
downloadFile(url + "/wpbc.names", "dataset/wpbc.names")
```

In [3]:

```
headers = ["ID", "Outcome", "Time", "Radius Mean", "Texture Mean", "Perimeter Mean", "A
rea Mean", "Smoothness Mean",
           "Compactness Mean", "Concavity Mean", "Concave points Mean", "Symmetry Mean"
, "Fractal dimension Mean",
           "Radius SE", "Texture SE", "Perimeter SE", "Area SE", "Smoothness SE", "Comp
actness SE", "Concavity SE",
           "Concave points SE", "Symmetry SE", "Fractal dimension SE", "Radius Worst",
"Texture Worst", "Perimeter Worst",
           "Area Worst", "Smoothness Worst", "Compactness Worst", "Concavity Worst", "C
oncave points Worst",
           "Symmetry Worst", "Fractal dimension Worst", "Tumor size", "Lymph node statu
s"]
data = pd.read_csv("dataset/wpbc.data", names=headers)
data = data.astype({"Outcome": "category"})
data.sample(40)
```

Out[3]:

	ID	Outcome	Time	Radius Mean	Texture Mean	Perimeter Mean	Area Mean	Smoothness Mean	Compactness Mean
64	868826	N	36	14.95	17.57	96.85	678.1	0.11670	0.1305
34	855625	R	9	19.07	24.81	128.30	1104.0	0.09081	0.2190
133	901088	N	61	20.44	21.78	133.80	1293.0	0.09150	0.1131
108	887256	N	27	15.53	33.56	103.70	744.9	0.10630	0.1639
94	881190	N	74	19.53	32.47	128.00	1223.0	0.08420	0.1130
88	879523	R	17	15.12	16.68	98.78	716.6	0.08876	0.0958
26	854002	N	53	19.27	26.47	127.90	1162.0	0.09401	0.1719
41	857793	N	62	14.71	21.59	95.55	656.9	0.11370	0.1365
8	844981	N	119	13.00	21.82	87.50	519.8	0.12730	0.1932
5	843786	R	77	12.75	15.29	84.60	502.7	0.11890	0.1569
109	887549	R	39	20.31	27.06	132.90	1288.0	0.10000	0.1088
119	892189	N	1	11.76	18.14	75.00	431.1	0.09968	0.0591
184	937664	N	17	17.98	23.96	120.00	995.0	0.11570	0.1739
63	868202	N	10	12.77	22.47	81.72	506.3	0.09055	0.0576
185	937897	N	17	13.63	24.70	89.65	569.2	0.10550	0.1312
59	866203	R	73	19.00	18.91	123.40	1138.0	0.08217	0.0802
107	887181	N	64	15.66	23.20	110.20	773.5	0.11090	0.3114
44	859283	N	106	14.78	23.94	97.40	668.3	0.11720	0.1479
157	914062	R	12	18.01	20.56	118.40	1007.0	0.10010	0.1289
135	9012000	R	2	22.01	21.90	147.20	1482.0	0.10630	0.1954
155	913505	R	14	19.44	18.82	128.10	1167.0	0.10890	0.1448
190	939426	N	8	19.96	27.41	130.80	1238.0	0.09075	0.1167
31	855138	N	76	13.48	20.82	88.40	559.2	0.10160	0.1255
47	8610637	N	97	19.55	15.49	128.00	1156.0	0.10790	0.1747
118	8912280	N	8	16.24	18.77	108.80	805.1	0.10660	0.1802
101	884180	N	62	19.40	23.50	129.10	1155.0	0.10270	0.1558
189	939095	N	6	19.80	20.46	130.20	1235.0	0.09652	0.1077
176	929684	R	14	17.53	25.28	114.00	966.6	0.09278	0.0917
104	886452	N	58	13.96	17.05	91.43	602.4	0.10960	0.1279
111	889403	N	62	15.61	19.38	100.00	758.6	0.07840	0.0561
17	851509	R	10	21.16	23.04	137.20	1404.0	0.09428	0.1022
10	845636	N	123	16.02	23.24	102.70	797.8	0.08206	0.0666
39	857438	R	48	15.10	22.02	97.26	712.8	0.09056	0.0708
66	869691	N	43	11.80	16.58	78.99	432.0	0.10910	0.1700
50	86208	R	10	20.77	22.83	137.40	1336.0	0.10330	0.1515
14	848406	N	123	14.68	20.13	94.74	684.5	0.09867	0.0720

	ID	Outcome	Time	Radius Mean	Texture Mean	Perimeter Mean	Area Mean	Smoothness Mean	Compactnes Mea
178	931678	N	24	24.29	25.48	161.80	1715.0	0.09374	0.228
75	873592	R	17	27.22	21.87	182.10	2250.0	0.10940	0.191
96	881861	N	77	12.83	22.33	85.26	503.2	0.10880	0.179
125	89539	R	78	16.27	20.71	106.90	813.7	0.11690	0.131

In [4]:

```
data.dtypes
```

Out[4]:

```
ID                           int64
Outcome                   category
Time                         int64
Radius Mean                float64
Texture Mean               float64
Perimeter Mean             float64
Area Mean                  float64
Smoothness Mean            float64
Compactness Mean           float64
Concavity Mean             float64
Concave points Mean        float64
Symmetry Mean              float64
Fractal dimension Mean     float64
Radius SE                  float64
Texture SE                 float64
Perimeter SE               float64
Area SE                    float64
Smoothness SE              float64
Compactness SE             float64
Concavity SE               float64
Concave points SE          float64
Symmetry SE                float64
Fractal dimension SE       float64
Radius Worst               float64
Texture Worst              float64
Perimeter Worst            float64
Area Worst                 float64
Smoothness Worst           float64
Compactness Worst          float64
Concavity Worst            float64
Concave points Worst       float64
Symmetry Worst             float64
Fractal dimension Worst    float64
Tumor size                 float64
Lymph node status           object
dtype: object
```

Колонка *Lymph node status* имеет тип *object*. Проверим, имеются ли в ней пропуски.

In [5]:

```
data['Lymph node status'].replace({'?': np.nan}, inplace=True)
display(data.isna().sum())
```

```
ID                         0
Outcome                    0
Time                       0
Radius Mean                0
Texture Mean               0
Perimeter Mean             0
Area Mean                  0
Smoothness Mean            0
Compactness Mean           0
Concavity Mean             0
Concave points Mean        0
Symmetry Mean              0
Fractal dimension Mean     0
Radius SE                  0
Texture SE                 0
Perimeter SE               0
Area SE                    0
Smoothness SE              0
Compactness SE             0
Concavity SE               0
Concave points SE          0
Symmetry SE                0
Fractal dimension SE       0
Radius Worst               0
Texture Worst              0
Perimeter Worst            0
Area Worst                 0
Smoothness Worst           0
Compactness Worst          0
Concavity Worst            0
Concave points Worst       0
Symmetry Worst             0
Fractal dimension Worst    0
Tumor size                 0
Lymph node status          4
dtype: int64
```

Имеется 4 пропуска в колонке *Lymph node status*. [Заполним (https://basegroup.ru/community/articles/missing)](https://basegroup.ru/community/articles/missing) их модой. Малое количество пропусков не должно привести к существенному искажению распределения.

In [6]:

```
data["Lymph node status"].fillna(data["Lymph node status"].mode()[0], inplace=True)
data = data.astype({"Lymph node status": "float64"})
data.drop(columns=["ID"], inplace=True)

display(data.dtypes["Lymph node status"])
display(data.describe())
```

```
dtype('float64')
```

	Time	Radius Mean	Texture Mean	Perimeter Mean	Area Mean	Smoothness Mean	Compactne Me
count	198.000000	198.000000	198.00000	198.000000	198.000000	198.000000	198.0000
mean	46.732323	17.412323	22.27601	114.856566	970.040909	0.102681	0.1426
std	34.462870	3.161676	4.29829	21.383402	352.149215	0.012522	0.0498
min	1.000000	10.950000	10.38000	71.900000	361.600000	0.074970	0.0460
25%	14.000000	15.052500	19.41250	98.160000	702.525000	0.093900	0.1102
50%	39.500000	17.290000	21.75000	113.700000	929.100000	0.101900	0.1317
75%	72.750000	19.580000	24.65500	129.650000	1193.500000	0.110975	0.1722
max	125.000000	27.220000	39.28000	182.100000	2250.000000	0.144700	0.3114

Подготовим тренировочные и тестовые выборки.

In [26]:

```
X = data.drop(columns=["Outcome"]).copy()
y = data["Outcome"].replace({'N': -1, 'R': 1})

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state=
27)
```

Создадим классификатор, обучим его, а затем выполним классификацию.

In [27]:

```
y_pred = DecisionTreeClassifier(max_depth=25).fit(X_train, y_train).predict(X_test)
```

Оценим получившуюся классификацию.

In [28]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

          -1       0.88      0.79      0.83        57
           1       0.37      0.54      0.44        13

    accuracy                           0.74        70
   macro avg       0.63      0.66      0.64        70
weighted avg       0.79      0.74      0.76        70
```

In [29]:

```
fpr, tpr, _ = roc_curve(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred)
```

In [30]:

```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = "AUC = %0.2f"%auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```