Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Тульский государственный университет»

Институт прикладной математики и компьютерных наук
Кафедра прикладной математики и информатики

Дискриминантный анализ. Решение задачи бинарной классификации при помощи логистической регрессии и нейронной сети.

Отчёт по лабораторной работе
по курсу «Теория вероятностей и математическая статистика»

Выполнила: студентка группы 221201 _____ Холичева А.А.
Проверил: профессор каф. ПМиИ _____ Кочетыгов А.А.

Тула 2023

Постановка задачи

Задача классификации в машинном обучении – это задача отнесения объекта к одному из заранее определённых классов на основании его формализованных признаков. Отдельный объект в данной задаче представляется в виде вектора в N-мерном пространстве, каждое измерение в котором представляет собой описание одного из признаков объекта.

Отдельно стоит выделить задачу бинарной классификации. Как видно из названия, в этом случае имеется всего 2 возможных класса, к одному из которых может принадлежать объект. К примерам задач такого вида можно отнести задачу фильтрации электронной почты: необходимо определить, является ли письмо спамом или нет.

Рассмотрим задачу бинарной классификации на примере датасета, содержащего данные о симптомах пациентов и наличии у них заболеваний сердечно-сосудистой системы. Набор данных представляет собой таблицу из 918 строк (наблюдений) и 12 показателей (рис. 1):

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Рис. 1. Первые 5 строк датасета, выведенные средствами библиотеки Pandas

Опишем имеющиеся показатели:

1. Age – возраст пациента в годах, количественный признак. Диапазон значений: [28, 77].
2. Sex – пол пациента, качественный бинарный признак. Множество значений: [‘M’, ‘F’], где ‘M’ - мужской, ‘F’ – женский.
3. ChestPainType – тип боли в грудной клетке, качественный признак. Множество значений: [‘TA’, ‘ATA’, ‘NAP’, ‘ASY’], где ‘TA’ – Typical Angina – типичная стенокардия, ‘ATA’ – Atypical Angina – атипичная стенокардия, ‘NAP’ – Non-Anginal Pain – неангинальная боль, ‘ASY’ – Asymptomatic – бессимптомная.
4. RestingBP – кровяное давление в состоянии покоя в мм. рт. ст., количественный признак. Диапазон значений: [0, 200].
5. Cholesterol – общий холестерин в сыворотке крови в мг/дл, количественный признак. Диапазон значений: [0, 603].

6. FastingBS – уровень сахара в крови натощак, качественный бинарный признак. Множество значений: [0, 1], где 1 – уровень сахара превышает 120 мг/дл, 0 – в противном случае.

7. RestingECG – ЭКГ в состоянии покоя, качественный признак. Множество значений: [‘Normal’, ‘ST’, ‘LVH’], где ‘Normal’ – нормальные показатели, ‘ST’ – наличие аномалии зубца ST-T, ‘LVH’ – наличие вероятной или определенной гипертрофии левого желудочка по критериям Эстеса.

8. MaxHR – достигнутая максимальная частота сердечных сокращений, количественный признак. Множество значений: [60, 202].

9. ExerciseAngina – стенокардия, вызванная физической нагрузкой, качественный бинарный признак. Множество значений: [‘Y’, ‘N’], где ‘Y’ – да, ‘N’ – нет.

10. Oldpeak – подавление сегмента ST, вызванное упражнением относительно отдыха, количественный признак. Диапазон значений: [-2.6, 6.2].

11. ST_Slope - наклон максимального сегмента ST упражнения, качественный признак. Множество значений: [‘Up’, ‘Flat’, ‘Down’], где ‘Up’ – наклон вверх, ‘Flat’ – наклон отсутствует, ‘Down’ – наклон вниз.

12. HeartDisease – наличие заболевания сердечно-сосудистой системы, качественный бинарный признак.

Будем рассматривать 11 показателей для прогнозирования наличия у пациента сердечно-сосудистого заболевания.

Разобьём задачу на несколько этапов:

1. Разведочный анализ данных
2. Поиск выбросов и их удаление
3. Выделение значимых признаков и подготовка данных
4. Построение моделей

Разведочный анализ данных

Получим общее описание имеющихся данных, их основные статистические характеристики, коэффициенты корреляции между количественными переменными, гистограммы распределений.

Используя библиотеку Pandas для языка программирования Python, разделим все переменные на 3 типа: числовой, категориальный, целевой.

Листинг 1

```
target = ['HeartDisease']

categorical = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope',
'FastingBS']

numerical = dataframe.drop(labels=target + categorical, axis=1).columns.to_list()

print(f"Numerical: {numerical}\nCategorical: {categorical}\n\nTarget: {target}")
```

Перекодируем переменные Sex и ExerciseAngina в числа 0 и 1, так как это бинарные признаки.

Листинг 2

```
dataframe = dataframe.replace({'Sex': {'M': 1, 'F': 0}, 'ExerciseAngina': {'N': 0, 'Y':
1}})
```

Командой dataframe.describe() выведем основные статистические показатели данных (рис. 2).

	Age	Sex	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	0.789760	132.396514	198.799564	0.233115	136.809368	0.404139	0.887364	0.553377
std	9.432617	0.407701	18.514154	109.384145	0.423046	25.460334	0.490992	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	0.000000	60.000000	0.000000	-2.600000	0.000000
25%	47.000000	1.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000	0.000000
50%	54.000000	1.000000	130.000000	223.000000	0.000000	138.000000	0.000000	0.600000	1.000000
75%	60.000000	1.000000	140.000000	267.000000	0.000000	156.000000	1.000000	1.500000	1.000000
max	77.000000	1.000000	200.000000	603.000000	1.000000	202.000000	1.000000	6.200000	1.000000

Рис. 2. Статистические показатели данных, выраженных числами

При помощи строчки кода dataframe.corr().style.background_gradient() получим корреляционную матрицу (рис. 3).

	Age	Sex	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	HeartDisease
Age	1.000000	0.055750	0.254399	-0.095282	0.198039	-0.382045	0.215793	0.258612	0.282039
Sex	0.055750	1.000000	0.005133	-0.200092	0.120076	-0.189186	0.190664	0.105734	0.305445
RestingBP	0.254399	0.005133	1.000000	0.100893	0.070193	-0.112135	0.155101	0.164803	0.107589
Cholesterol	-0.095282	-0.200092	0.100893	1.000000	-0.260974	0.235792	-0.034166	0.050148	-0.232741
FastingBS	0.198039	0.120076	0.070193	-0.260974	1.000000	-0.131438	0.060451	0.052698	0.267291
MaxHR	-0.382045	-0.189186	-0.112135	0.235792	-0.131438	1.000000	-0.370425	-0.160691	-0.400421
ExerciseAngina	0.215793	0.190664	0.155101	-0.034166	0.060451	-0.370425	1.000000	0.408752	0.494282
Oldpeak	0.258612	0.105734	0.164803	0.050148	0.052698	-0.160691	0.408752	1.000000	0.403951
HeartDisease	0.282039	0.305445	0.107589	-0.232741	0.267291	-0.400421	0.494282	0.403951	1.000000

Рис. 3. Матрица коэффициентов корреляции Пирсона

Средствами библиотеки matplotlib построим гистограммы распределения числовых и качественных признаков (рис. 4, 5).

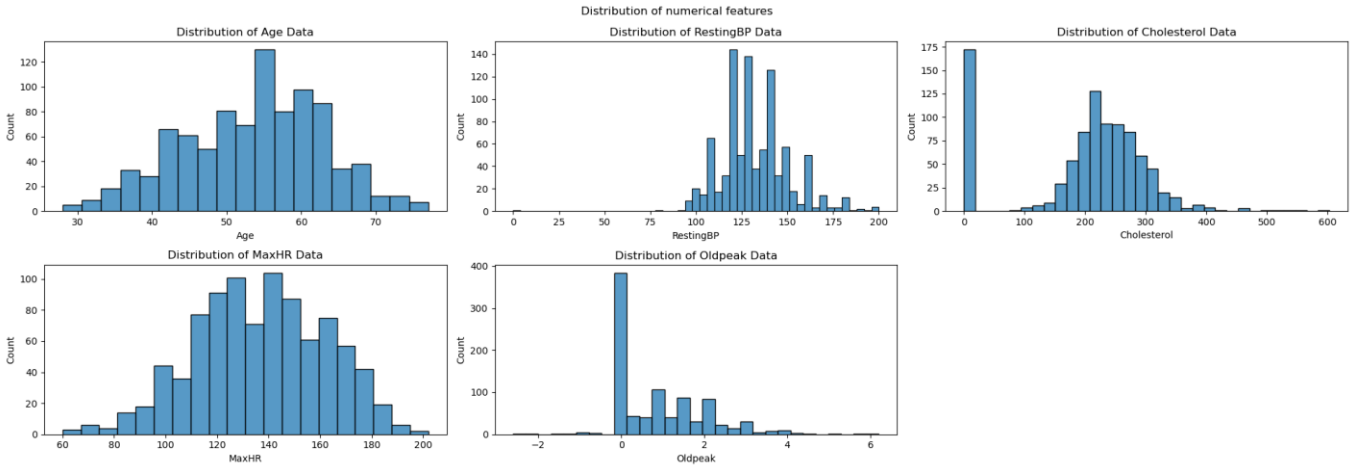

Рис. 4. Распределение количественных признаков

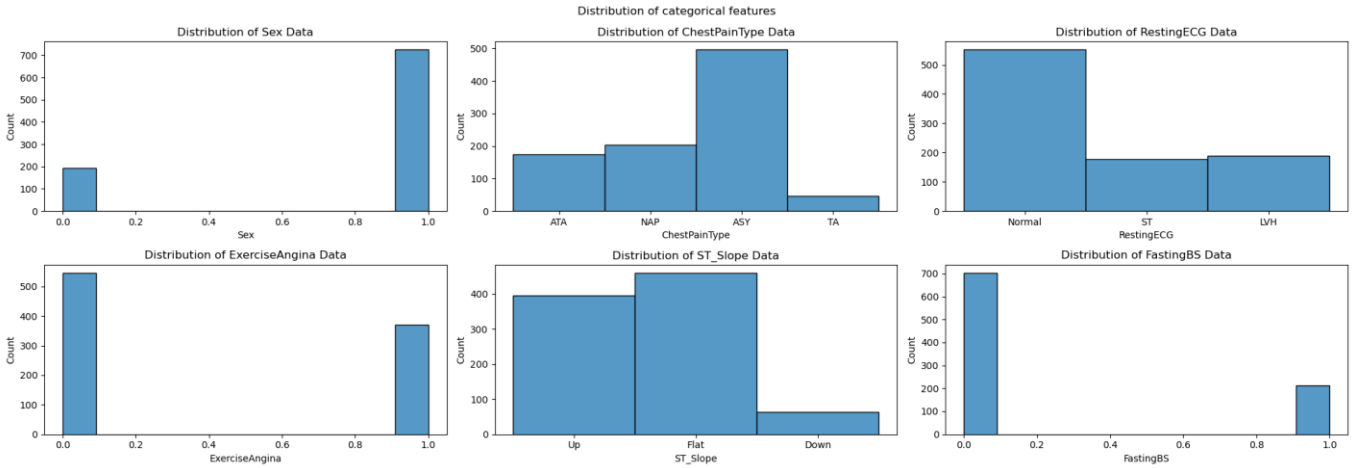

Рис. 5. Распределение качественных признаков

Построим также гистограмму, в которой распределение качественных признаков сгруппировано по наличию или отсутствию у человека сердечно-сосудистого заболевания (рис. 6).

Листинг 3

```
plt.figure(figsize=(25, 4))

for i, col in enumerate(categorical, 1):
    plt.subplot(1, 5, i)
    ax = sns.countplot(data=dataframe, x='HeartDisease', hue=col)

    for label in ax.containers:
        ax.bar_label(label)
```

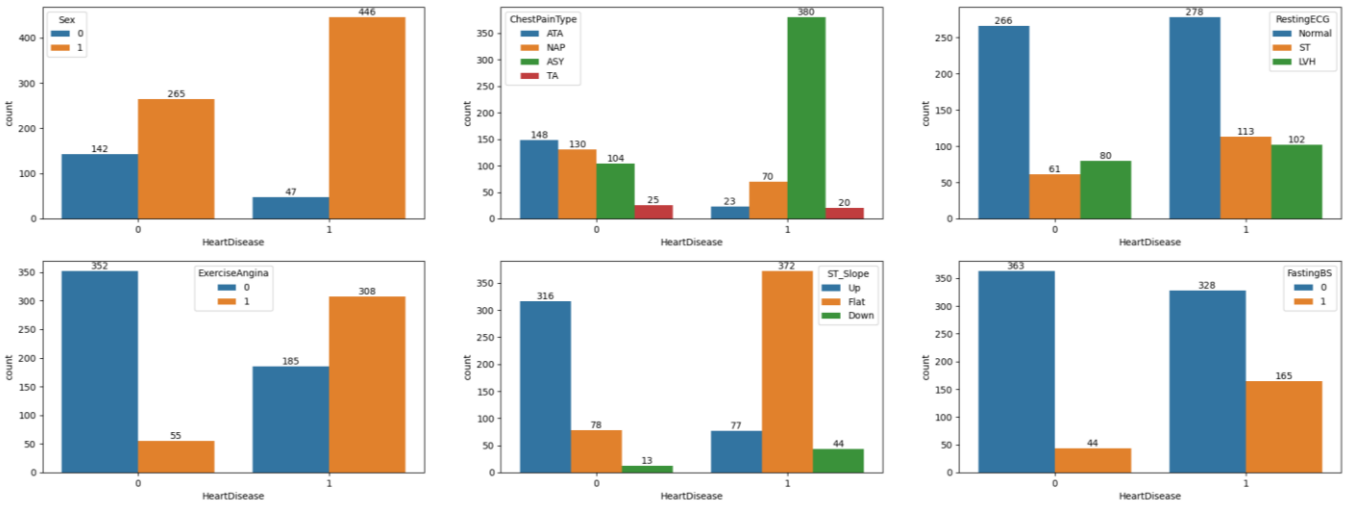

Рис. 6. Распределение качественных признаков в зависимости от наличия или отсутствия заболевания

Поиск выбросов и их удаление

Из рис. 4 вполне можно заключить, что числовые признаки RestingBP, Cholesterol, Oldpeak могут содержать выбросы. От этого явления необходимо избавляться, так как выбросы и аномалии вносят в данные искажения и впоследствии мешают нормальной работе моделей классификации. Действительно, вряд ли значение кровяного давления 0 мм. рт. ст. можно считать адекватным, а ведь такое наблюдение присутствует в датасете.

Построим специальные графики, позволяющие визуально оценить разброс данных (рис. 7).

Листинг 4

```
plt.figure(figsize=(15, 4))
cols = ['RestingBP', 'Oldpeak', 'Cholesterol']
for i, col in enumerate(cols, 1):
    plt.subplot(1, 3, i)
    plt.title(col)
    sns.boxplot(dataframe[col])
```

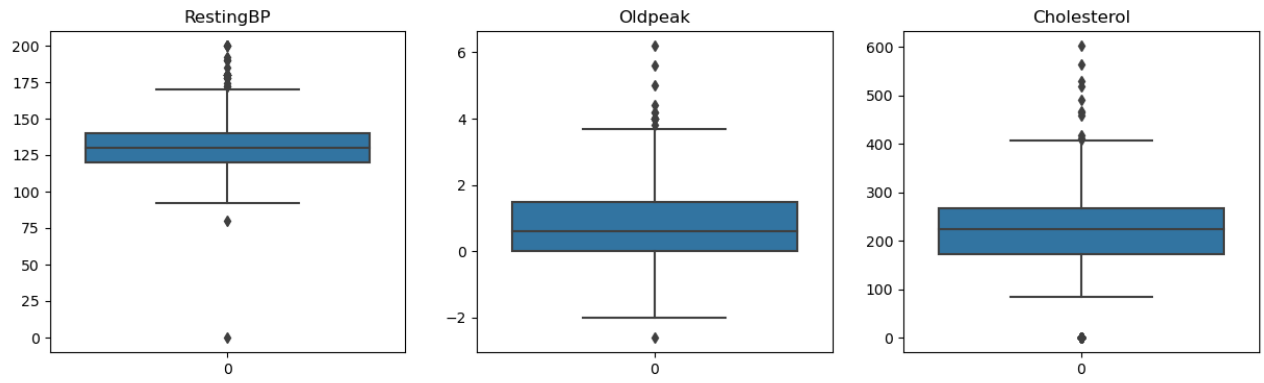

Рис. 7. Признаки, предположительно содержащие выбросы

Будем искать выбросы и удалять их, основываясь на правиле трёх сигм. Напишем соответствующую функцию, которая возвращает индексы строк таблицы, для которых по данному признаку не выполнено правило трёх сигм.

Листинг 5

```
# Поиск выбросов по конкретному числовому признаку
def outliers_indicies(df: pd.DataFrame, feature: str, print_info=False):
    mean = df[feature].mean()
    std = df[feature].std()
    if print_info:
        print(f'Признак {feature}, диапазон 3-сигм: [{mean - 3*std :.3f}, {mean + 3*std
:.3f}]')

    # условие того, что точка данных - выброс
    condition = (df[feature] < mean - 3*std) | (df[feature] > mean + 3*std)
    return df[condition].index
```

Произведём поиск выбросов по интересующим нас признакам и удалим наблюдения, содержащие выбросы (рис. 8).

Листинг 6

```
outliers_restingBP = outliers_indicies(dataframe, 'RestingBP', print_info=True)
outliers_cholesterol = outliers_indicies(dataframe, 'Cholesterol', print_info=True)
outliers_oldpeak = outliers_indicies(dataframe, 'Oldpeak', print_info=True)

all_outliers = set(outliers_restingBP) | set(outliers_cholesterol) |
set(outliers_oldpeak)
print(f'Всего выбросов: {len(all_outliers)}')
dataframe.drop(index=all_outliers, inplace=True)
```

```
Признак RestingBP, диапазон 3-сигм: [76.854, 187.939]
Признак Cholesterol, диапазон 3-сигм: [-129.353, 526.952]
Признак Oldpeak, диапазон 3-сигм: [-2.312, 4.087]
Всего выбросов: 18
```

Рис. 8. Результат поиска выбросов

Построим гистограммы распределения количественных признаков после удаления выбросов (рис. 9). Видим, что диаграммы действительно изменились и, скорее всего, стали ближе отражать реальные данные.

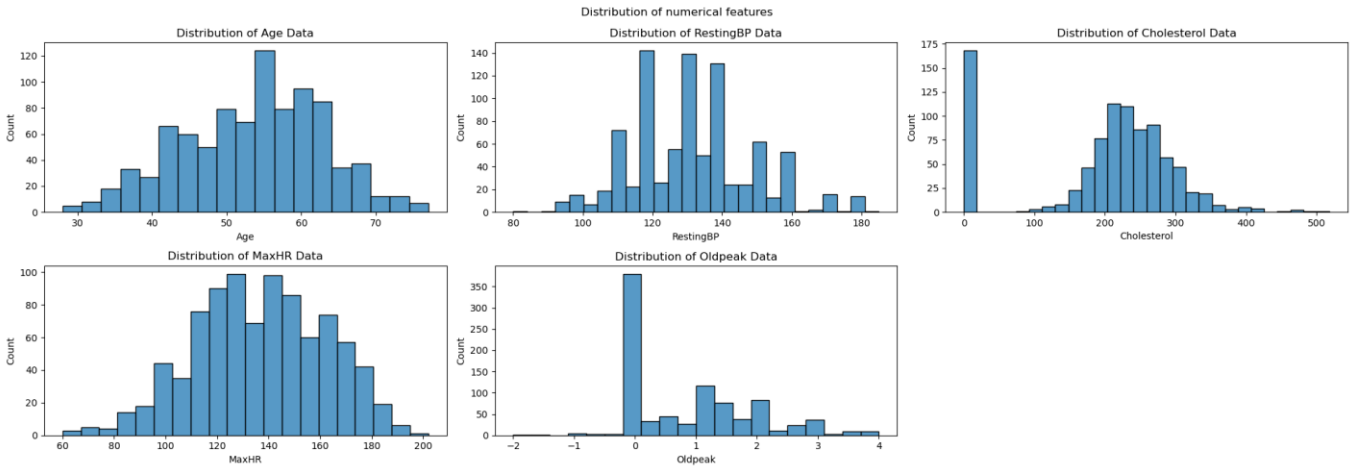

Рис. 9. Распределение числовых признаков после очистки от выбросов

Пользуясь данными, очищенными от выбросов, обновим корреляционную матрицу (рис. 10). Коэффициенты корреляции между переменными также скорректировались.

	Age	Sex	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	HeartDisease
Age	1.000000	0.065975	0.269629	-0.094785	0.199923	-0.392104	0.217126	0.261989	0.286263
Sex	0.065975	1.000000	0.018011	-0.207404	0.122038	-0.189205	0.195907	0.125558	0.309838
RestingBP	0.269629	0.018011	1.000000	0.105095	0.064822	-0.113307	0.151319	0.156439	0.110673
Cholesterol	-0.094785	-0.207404	0.105095	1.000000	-0.271042	0.239178	-0.038991	0.034192	-0.240357
FastingBS	0.199923	0.122038	0.064822	-0.271042	1.000000	-0.127130	0.057413	0.071720	0.267070
MaxHR	-0.392104	-0.189205	-0.113307	0.239178	-0.127130	1.000000	-0.375724	-0.174021	-0.406914
ExerciseAngina	0.217126	0.195907	0.151319	-0.038991	0.057413	-0.375724	1.000000	0.423603	0.496743
Oldpeak	0.261989	0.125558	0.156439	0.034192	0.071720	-0.174021	0.423603	1.000000	0.418470
HeartDisease	0.286263	0.309838	0.110673	-0.240357	0.267070	-0.406914	0.496743	0.418470	1.000000

Рис. 10. Новая корреляционная матрица

Выделение значимых признаков

Займёмся анализом значимости категориальных признаков. Из библиотеки SciPy импортируем функции для расчёта критерия Фишера и критерия хи-квадрат.

Построим матрицы сопряженности между каждым качественным признаком и целевой переменной и посчитаем критерий хи-квадрат, чтобы выяснить, есть ли корреляция с целевой переменной HeartDisease (рис. 11). Заметим, что данный критерий можно рассчитать для таблиц сопряжённости любого размера.

Листинг 7

```
for feature in categorical:
    confusion_matrix = pd.crosstab(dataframe[feature], dataframe['HeartDisease'])
    chi2, pvalue = chi2_contingency(confusion_matrix)[:2]
    print(confusion_matrix)
    print(f'chi2 = {chi2}, pvalue = {pvalue}\n')
```

```
HeartDisease    0    1
Sex
0             142   47
1             265  446
chi2 = 84.87817632809544, pvalue = 3.173263739498517e-20

HeartDisease     0    1
ChestPainType
ASY            104  380
ATA            148   23
NAP            130   70
TA              25   20
chi2 = 261.48808876346294, pvalue = 2.1423993796374233e-56

HeartDisease    0    1
RestingECG
LVH            80  102
Normal        266  278
ST             61  113
chi2 = 10.34092018687718, pvalue = 0.005681953987984134
```

```
HeartDisease      0    1
ExerciseAngina
0               352  185
1                55  308
chi2 = 220.04831361042943, pvalue = 8.827651221002139e-50

HeartDisease    0    1
ST_Slope
Down           13   44
Flat           78  372
Up            316   77
chi2 = 349.2569450600609, pvalue = 1.444834146570383e-76

HeartDisease    0    1
FastingBS
0             363  328
1              44  165
chi2 = 62.92928023655695, pvalue = 2.1426320877937236e-15
```

Рис. 11. Результат расчёта критерия хи-квадрат

По результатам расчёта видим, что pvalue во всех случаях не более 0.05, следовательно, нулевую гипотезу об отсутствии корреляции между каждой из категориальных переменных и целевой отвергаем.

Также проверим критерий Фишера для признаков Sex, ExerciseAngina и FastingBS, так как их таблицы сопряженности с целевой переменной имеют размер 2x2.

Листинг 8

```
for feature in ['Sex', 'ExerciseAngina', 'FastingBS']:
    crosstab = pd.crosstab(dataframe[feature], dataframe['HeartDisease'])
    print(crosstab)
    print(fisher_exact(crosstab), end='\n\n')
```

```
HeartDisease    0    1
Sex
0             142   47
1             265  446
SignificanceResult(statistic=5.084865515857086, pvalue=8.123345004279794e-21)

HeartDisease      0    1
ExerciseAngina
0               352  185
1                55  308
SignificanceResult(statistic=10.655135135135135, pvalue=1.2400210736753246e-53)

HeartDisease    0    1
FastingBS
0             363  328
1              44  165
SignificanceResult(statistic=4.1501524390243905, pvalue=2.1563232433789856e-16)
```

Рис. 12. Результат расчёта критерия Фишера

Здесь все значения pvalue также меньше 0.05, следовательно, нулевую гипотезу отвергаем, между переменными есть взаимосвязь.

На основании разведочного анализа данных и расчёта критерия хи-квадрат исключим переменную RestingECG из числа признаков.

Распределим датасет по двум переменным: X – матрица признаков, y – целевой вектор.

Построение моделей. Логистическая регрессия

Разобьём датасет на тренировочную и валидационную выборки и отмасштабируем данные.

Листинг 9

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, stratify=y)
# scaler = StandardScaler()
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
```

Используя библиотеку scikit-learn, создадим модель логистической регрессии с параметрами penalty='l2', solver='liblinear' и обучим её на тренировочном наборе. На валидационном наборе рассчитаем метрики качества модели (accuracy, ROC AUC), построим матрицу ошибок (рис. 13).

Листинг 10

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score

clf = LogisticRegression(penalty='l2', solver='liblinear')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_val)

print("Accuracy: ", accuracy_score(y_val, y_pred))
print("ROC AUC: ", roc_auc_score(y_val, y_pred))

conf_matrix = confusion_matrix(y_val, y_pred)
ax = sns.heatmap(conf_matrix, annot=True)
ax.set(xlabel='Predicted', ylabel='Actual')
plt.show()
```

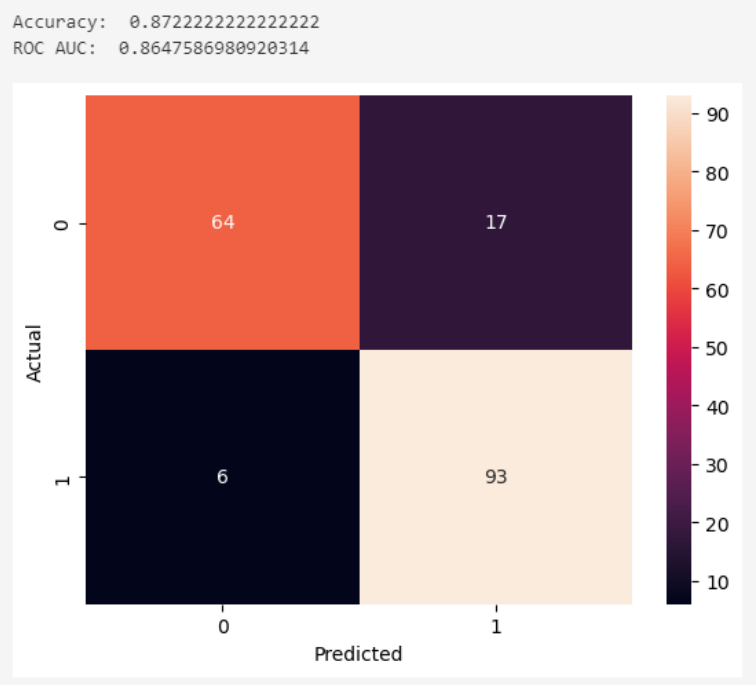

Рис. 13. Матрица ошибок для логистической регрессии

Построение модели. Нейронная сеть

При помощи фреймворка PyTorch напишем класс для создания датасета, архитектуру нейросети-классификатора, а также функцию обучения train:

Листинг 11

```
import torch
import torch.nn as nn
from torch.utils.data import Dataset
import numpy as np
```

```
class HeartDataset(Dataset):
    def __init__(self, X, y):
        if isinstance(X, np.ndarray):
            self.X = torch.tensor(X, dtype=torch.float32)
        else:
            self.X = torch.tensor(X.to_numpy(), dtype=torch.float32)

        if isinstance(y, np.ndarray):
            self.y = torch.tensor(y, dtype=torch.float32)
        else:
            self.y = torch.tensor(y.to_numpy(), dtype=torch.float32)

        self.length = len(y)

    def __getitem__(self, index):
        return self.X[index], self.y[index]

    def __len__(self):
        return self.length


class NeuralNetwork(nn.Module):
    def __init__(self, n_features):
        super().__init__()

        self.main = nn.Sequential(
            nn.Linear(n_features, 100),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(100, 50),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(50, 1),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.main(x)


def train(model: nn.Module,
          num_epochs: int,
          train_loader,
          optimizer,
          loss_fn,
          valid_loader,
          epoch_logging_interval=10):

    log_dict = {'losses_per_epoch': [], 'accuracy_per_epoch': []}
```

```
    for epoch in range(num_epochs):

        model.train()
        loss_curr_epoch = 0
        for batch, (x, y) in enumerate(train_loader):
            y = y.view(-1, 1)
            y_hat = model(x)

            # Усреднённый лосс по батчам
            loss = loss_fn(y_hat, y)
            loss_curr_epoch += loss.item()

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()


        model.eval()
        accuracy_curr_epoch = 0
        for batch, (x, y) in enumerate(valid_loader):
            y_pred = model(x).view(-1)

            # Среднее accuracy по батчам
            accuracy_curr_epoch += torch.eq(y_pred.round(), y).detach().numpy().mean()

        log_dict['losses_per_epoch'].append(loss_curr_epoch / len(train_loader))
        log_dict['accuracy_per_epoch'].append(accuracy_curr_epoch / len(valid_loader))

        if (epoch + 1) % epoch_logging_interval == 0:
            print(f'Epoch {epoch + 1}:\tLoss {log_dict["losses_per_epoch"][epoch]},
Accuracy {log_dict["accuracy_per_epoch"][epoch]}')

    return log_dict
```

Произведём с данными те же манипуляции, что и в случае первой модели: разобьём на тренировочную и валидационную выборки, нормализуем. Зададим исходные гиперпараметры для обучения.

Листинг 12

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)

train_dataset = HeartDataset(X_train, y_train)
val_dataset = HeartDataset(X_val, y_val)

train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=16, shuffle=True)
```

```
n_features = X_train.shape[1]
num_epochs = 200
learning_rate = 0.01
```

Инициализируем модель, выберем оптимизатор и лосс-функцию. В частности, для задачи бинарной классификации в качестве лосс-функции используется бинарная кросс-энтропия. Запустим процесс обучения и сохраним в словарь данные о лоссе и метрике по эпохам обучения (рис. 14).

Листинг 13

```
model = NeuralNetwork(n_features)
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
loss_fn = nn.BCELoss()
print(model)

log_dict = network.train(model, num_epochs, train_loader, optimizer, loss_fn,
val_loader)
```

Построим графики изменения лосса и точности (рис. 14):

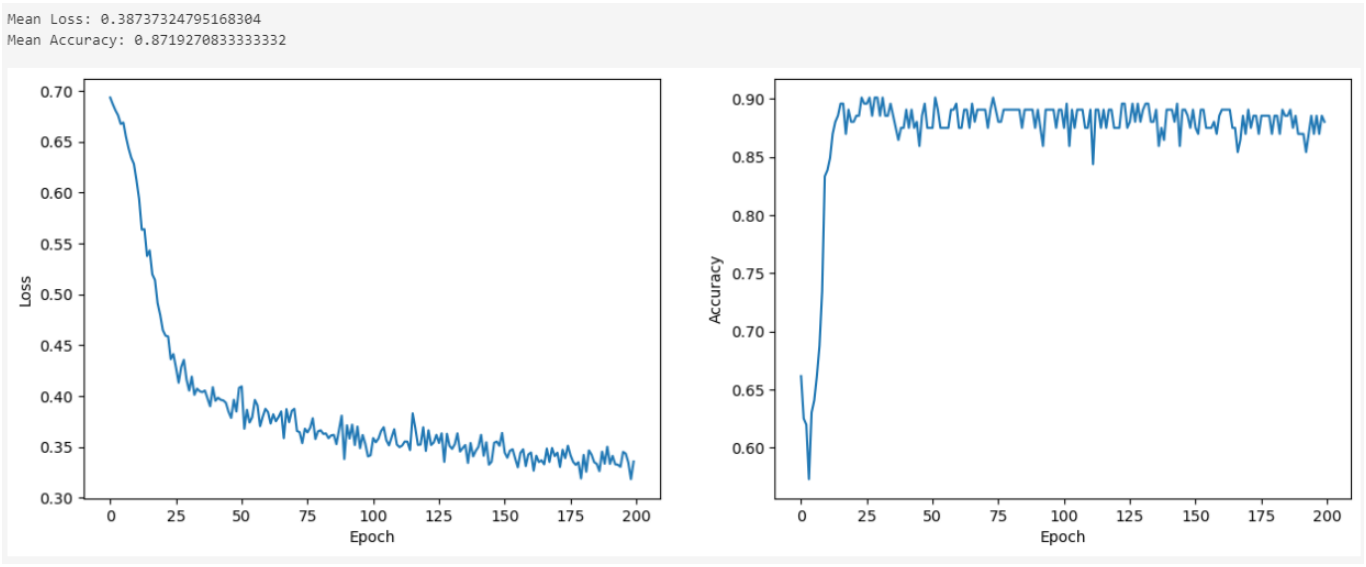

Рис. 14

Наконец, построим матрицу ошибок для обученной нейронной сети:

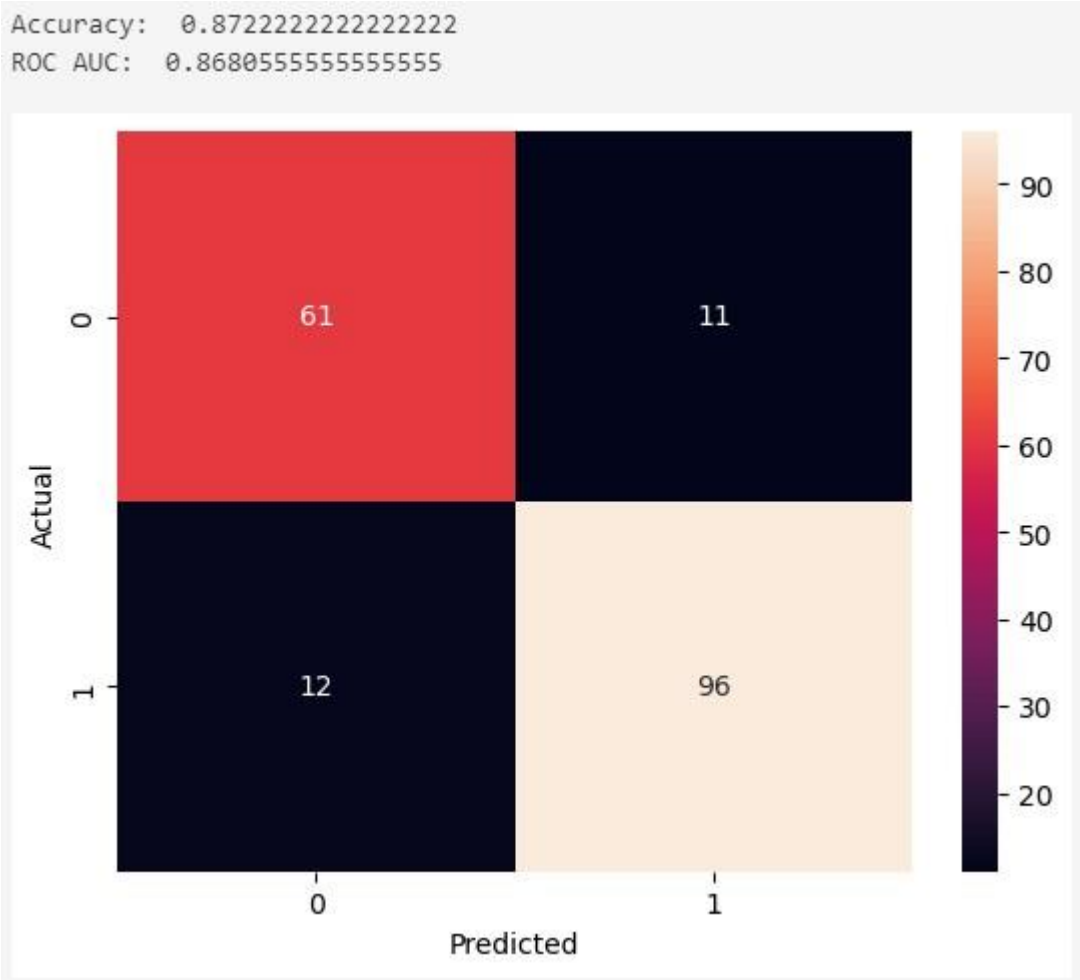

Рис. 15

Выводы

Сравнивая метрики, полученные при использовании двух разных моделей (логистической регрессии и нейронной сети) видим, что обе модели с достаточной точностью обучились предсказывать наличие заболевания сердечно-сосудистой системы по имеющимся показателям. Заметим, что в рамках этой конкретной задачи важно минимизировать число пациентов, для которых модель предсказала отсутствие заболевания, в то время как на самом деле оно есть. Иными словами, нужно минимизировать процент FalseNegative (левая нижняя клетка матрицы ошибок).