by Maria

Predicting Heart Disease Using Logistic Regression

In [1]:
```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV, Shuff

from sklearn.linear_model import LogisticRegression, LinearRegression, La
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score, confusion_matrix, accuracy_sc
import warnings
warnings.filterwarnings("ignore")
```

In [51]:
```
from ucimlrepo import fetch_ucirepo

# fetch dataset
heart_disease = fetch_ucirepo(id=45)

# data (as pandas dataframes)
X = heart_disease.data.features
y = heart_disease.data.targets

# metadata
print(heart_disease.metadata)

# variable information
print(heart_disease.variables)
```

{'uci_id': 45, 'name': 'Heart Disease', 'repository_url': 'https://archiv
e.ics.uci.edu/dataset/45/heart+disease', 'data_url': 'https://archive.ics.
uci.edu/static/public/45/data.csv', 'abstract': '4 databases: Cleveland, H
ungary, Switzerland, and the VA Long Beach', 'area': 'Health and Medicin
e', 'tasks': ['Classification'], 'characteristics': ['Multivariate'], 'num
_instances': 303, 'num_features': 13, 'feature_types': ['Categorical', 'In
teger', 'Real'], 'demographics': ['Age', 'Sex'], 'target_col': ['num'], 'i
ndex_col': None, 'has_missing_values': 'yes', 'missing_values_symbol': 'Na
N', 'year_of_dataset_creation': 1989, 'last_updated': 'Fri Nov 03 2023',
'dataset_doi': '10.24432/C52P4X', 'creators': ['Andras Janosi', 'William S
teinbrunn', 'Matthias Pfisterer', 'Robert Detrano'], 'intro_paper': {'ID':
231, 'type': 'NATIVE', 'title': 'International application of a new probab
ility algorithm for the diagnosis of coronary artery disease.', 'authors':
'R. Detrano, A. Jánosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu,
K. Guppy, S. Lee, V. Froelicher', 'venue': 'American Journal of Cardiolog
y', 'year': 1989, 'journal': None, 'DOI': None, 'URL': 'https://www.semant
icscholar.org/paper/a7d714f8f87bfc41351eb5ae1e5472f0ebbe0574', 'sha': Non
e, 'corpus': None, 'arxiv': None, 'mag': None, 'acl': None, 'pmid': '27568
73', 'pmcid': None}, 'additional_info': {'summary': 'This database contain
s 76 attributes, but all published experiments refer to using a subset of
14 of them. In particular, the Cleveland database is the only one that ha
s been used by ML researchers to date. The "goal" field refers to the pre
sence of heart disease in the patient. It is integer valued from 0 (no pr
esence) to 4. Experiments with the Cleveland database have concentrated on
simply attempting to distinguish presence (values 1,2,3,4) from absence (v
alue 0). \n   \nThe names and social security numbers of the patients wer
e recently removed from the database, replaced with dummy values.\n\nOne f
ile has been "processed", that one containing the Cleveland database. All
four unprocessed files also exist in this directory.\n\nTo see Test Costs
(donated by Peter Turney), please see the folder "Costs" ', 'purpose': Non
e, 'funded_by': None, 'instances_represent': None, 'recommended_data_split
s': None, 'sensitive_data': None, 'preprocessing_description': None, 'vari
able_info': 'Only 14 attributes used:\r\n   1. #3 (age)   \r\n
2. #4 (sex)   \r\n   3. #9 (cp)   \r\n   4. #10 (trestbp
s) \r\n   5. #12 (chol)   \r\n   6. #16 (fbs)   \r\n   7.
#19 (restecg)   \r\n   8. #32 (thalach)   \r\n   9. #38 (exang)
\r\n   10. #40 (oldpeak)   \r\n   11. #41 (slope)   \r\n   12.
#44 (ca)   \r\n   13. #51 (thal)   \r\n   14. #58 (num)
(the predicted attribute)\r\n\r\nComplete attribute documentation:\r\n
1 id: patient identification number\r\n   2 ccf: social security number
(I replaced this with a dummy value of 0)\r\n   3 age: age in years\r\n
4 sex: sex (1 = male; 0 = female)\r\n   5 painloc: chest pain location
(1 = substernal; 0 = otherwise)\r\n   6 painexer (1 = provoked by exert
ion; 0 = otherwise)\r\n   7 relrest (1 = relieved after rest; 0 = other
wise)\r\n   8 pncaden (sum of 5, 6, and 7)\r\n   9 cp: chest pain ty
pe\r\n   -- Value 1: typical angina\r\n   -- Value 2: atypical a
ngina\r\n   -- Value 3: non-anginal pain\r\n   -- Value 4: asymp
tomatic\r\n   10 trestbps: resting blood pressure (in mm Hg on admission
to the hospital)\r\n   11 htn\r\n   12 chol: serum cholestoral in mg/d
l\r\n   13 smoke: I believe this is 1 = yes; 0 = no (is or is not a smok
er)\r\n   14 cigs (cigarettes per day)\r\n   15 years (number of years
as a smoker)\r\n   16 fbs: (fasting blood sugar > 120 mg/dl) (1 = true;
0 = false)\r\n   17 dm (1 = history of diabetes; 0 = no such history)\r
\n   18 famhist: family history of coronary artery disease (1 = yes; 0 =
no)\r\n   19 restecg: resting electrocardiographic results\r\n   --
Value 0: normal\r\n   -- Value 1: having ST-T wave abnormality (T wav
e inversions and/or ST elevation or depression of > 0.05 mV)\r\n   --
Value 2: showing probable or definite left ventricular hypertrophy by Este
s\' criteria\r\n   20 ekgmo (month of exercise ECG reading)\r\n   21 e
kgday(day of exercise ECG reading)\r\n   22 ekgyr (year of exercise ECG

reading)\r\n 23 dig (digitalis used furing exercise ECG: 1 = yes; 0 = no)\r\n 24 prop (Beta blocker used during exercise ECG: 1 = yes; 0 = no)\r\n 25 nitr (nitrates used during exercise ECG: 1 = yes; 0 = no)\r\n 26 pro (calcium channel blocker used during exercise ECG: 1 = yes; 0 = no)\r\n 27 diuretic (diuretic used used during exercise ECG: 1 = yes; 0 = no)\r\n 28 proto: exercise protocol\r\n 1 = Bruce \r\n 2 = Kottus\r\n 3 = McHenry\r\n 4 = fast Balke\r\n 5 = Balke\r\n 6 = Noughton \r\n 7 = bike 150 kpa min/min (Not sure if "kpa min/min" is what was written!)\r\n 8 = bike 125 kpa min/min \r\n 9 = bike 100 kpa min/min\r\n 10 = bike 75 kpa min/min\r\n 11 = bike 50 kpa min/min\r\n 12 = arm ergometer\r\n 29 thaldur: duration of exercise test in minutes\r\n 30 thaltime: time when ST measure depression was noted\r\n 31 met: mets achieved\r\n 32 thalach: maximum heart rate achieved\r\n 33 thalrest: resting heart rate\r\n 34 tpeakbps: peak exercise blood pressure (first of 2 parts)\r\n 35 tpeakbpd: peak exercise blood pressure (second of 2 parts)\r\n 36 dummy\r\n 37 trestbpd: resting blood pressure\r\n 38 exang: exercise induced angina (1 = yes; 0 = no)\r\n 39 xhypo: (1 = yes; 0 = no)\r\n 40 oldpeak = ST depression induced by exercise relative to rest\r\n 41 slope: the slope of the peak exercise ST segment\r\n -- Value 1: upsloping\r\n -- Value 2: flat\r\n -- Value 3: downsloping\r\n 42 rldv5: height at rest\r\n 43 rldv5e: height at peak exercise\r\n 44 ca: number of major vessels (0-3) colored by flourosopy\r\n 45 restckm: irrelevant\r\n 46 exerckm: irrelevant\r\n 47 restef: rest raidonuclid (sp?) ejection fraction\r\n 48 restwm: rest wall (sp?) motion abnormality\r\n 0 = none\r\n 1 = mild or moderate\r\n 2 = moderate or severe\r\n 3 = akinesis or dyskmem (sp?)\r\n 49 exeref: exercise radinalid (sp?) ejection fraction\r\n 50 exerwm: exercise wall (sp?) motion \r\n 51 thal: 3 = normal; 6 = fixed defect; 7 = reversable defect\r\n 52 thalsev: not used\r\n 53 thalpul: not used\r\n 54 earlobe: not used\r\n 55 cmo: month of cardiac cath (sp?) (perhaps "call")\r\n 56 cday: day of cardiac cath (sp?)\r\n 57 cyr: year of cardiac cath (sp?)\r\n 58 num: diagnosis of heart disease (angiographic disease status)\r\n -- Value 0: < 50% diameter narrowing\r\n -- Value 1: > 50% diameter narrowing\r\n (in any major vessel: attributes 59 through 68 are vessels)\r\n 59 lmt\r\n 60 ladprox\r\n 61 laddist\r\n 62 diag\r\n 63 cxmain\r\n 64 ramus\r\n 65 om1\r\n 66 om2\r\n 67 rcaprox\r\n 68 rcadist\r\n 69 lvx1: not used\r\n 70 lvx2: not used\r\n 71 lvx3: not used\r\n 72 lvx4: not used\r\n 73 lvf: not used\r\n 74 cathef: not used\r\n 75 junk: not used\r\n 76 name: last name of patient (I replaced this with the dummy string "name")', 'citation': None}}

	name	role	type	demographic	\
0	age	Feature	Integer	Age	
1	sex	Feature	Categorical	Sex	
2	cp	Feature	Categorical	None	
3	trestbps	Feature	Integer	None	
4	chol	Feature	Integer	None	
5	fbs	Feature	Categorical	None	
6	restecg	Feature	Categorical	None	
7	thalach	Feature	Integer	None	
8	exang	Feature	Categorical	None	
9	oldpeak	Feature	Integer	None	
10	slope	Feature	Categorical	None	
11	ca	Feature	Integer	None	
12	thal	Feature	Categorical	None	
13	num	Target	Integer	None	

	description	units	\

```
0                                                    None  years
1                                                    None   None
2                                                    None   None
3   resting blood pressure (on admission to the hospital)  mm Hg
4                                        serum cholestoral  mg/dl
5                          fasting blood sugar > 120 mg/dl   None
6                                                    None   None
7                              maximum heart rate achieved   None
8                                  exercise induced angina   None
9       ST depression induced by exercise relative to rest   None
10                                                   None   None
11    number of major vessels (0-3) colored by flourosopy   None
12                                                   None   None
13                             diagnosis of heart disease   None

   missing_values
0              no
1              no
2              no
3              no
4              no
5              no
6              no
7              no
8              no
9              no
10             no
11            yes
12            yes
13             no
```

0. Исходные данные

Предлагаю решить следующую задачу:

1. У вас есть датасет с различными жизненно важными показателями ~300 пациентов: https://archive.ics.uci.edu/dataset/45/heart+disease. Вам необходимо натренировать модель, которая будет предсказывать, есть у пациента болезнь сердца или нет.
2. Необходимо использовать лишь только данные для Cleveland. Поскольку в оригинальном датасете наличие болезни имеет 5 возможных значений (0,1,2,3,4), предлагаю модифицировать данные в последней колонке и оставить только две категории - здоровые (0) и больные (1,2,3,4). Соответственно все значения больше 0 вам нужно заменить на 1.
3. Натренировать модель логистической регрессии для задачи бинарной классификации.
4. Оценить модель с помощью следующих метрик и проанализировать результаты:
5. Confusion Matrix
6. Accuracy
7. Precision
8. Recall

9. Precision-Recall Curve
10. F1 metric
11. ROC/AUC Curve
12. Как себя ведут эти метрики для задачи бинарной классификации? Какие из них лучше всего подходят? Какие не подходят? Почему?
13. Как улучшить модель? Возможно ли использовать регуляризацию?

---

7. После того, как справитесь с этим заданием, попробуйте поработать с оригинальным датасетом (то есть у нас пациент может быть здоровым - 0, а также больным с разной степенью тяжести сердечного заболевания - 1,2,3,4)
8. Повторите процесс выше и проверьте качество вашей модели на тех же самых метриках уже для задачи классификации с 5 классами.
9. И вновь вопрос - как улучшить модель?

1. EDA (Exploratory Data Analysis)

In [4]:
```
data = X
data['disease_flag'] = y
data
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
298	45	1	1	110	264	0	0	132	0	1.2	2	0.0
299	68	1	4	144	193	1	0	141	0	3.4	2	2.0
300	57	1	4	130	131	0	0	115	1	1.2	2	1.0
301	57	0	2	130	236	0	2	174	0	0.0	2	1.0
302	38	1	3	138	175	0	0	173	0	0.0	1	NaN

303 rows × 14 columns

In [5]:
```
def check_spaces_in_column_names(df):
    columns_with_space = [col for col in df.columns if col.endswith(' ')]
    if columns_with_space:
        print("Столбцы с пробелом в конце названия:", columns_with_space)
    else:
        print("Пробелы в конце названий столбцов отсутствуют.")
```

In [6]:
```
check_spaces_in_column_names(data)
```

Пробелы в конце названий столбцов отсутствуют.

Сразу также добавим столбец с бинарными таргетами для бинарной кластеризации
(Две категории - здоровые (0) и больные (1,2,3,4).
Соответственно все значения больше 0 заменяем на 1.)

In [7]:
```
data['bin_flag'] = (data['disease_flag'] > 0).astype(int)
data
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0
...	...	...	...	...	...	...	...	...	...	...	...	..
298	45	1	1	110	264	0	0	132	0	1.2	2	0.0
299	68	1	4	144	193	1	0	141	0	3.4	2	2.0
300	57	1	4	130	131	0	0	115	1	1.2	2	1.0
301	57	0	2	130	236	0	2	174	0	0.0	2	1.0
302	38	1	3	138	175	0	0	173	0	0.0	1	NaN

303 rows × 15 columns

1.1. Анализ признакового пространства

In [8]:
```
info_data = {
    'Признак': [],
    'Описание на русском': [],
    'Тип признака' : []
}

cleveland_about = {
    'age' : 'Возраст,  (1 = мужчина; 0 = женщина)',
    'sex' : 'Пол',
    'cp' : 'Тип боли в груди (1: типичная ангина, 2: атипичная стенокарди
    'trestbps' : 'Артериальное давление в состоянии покоя (в мм рт. ст. п
    'chol' : 'Холестерин в сыворотке крови в мг/дл',
    'fbs' : 'Сахар натощак > 120 мг/дл (1 = истина; 0 = ложь)',
    'restecg' : 'Результаты электрокардиографии в состоянии покоя (0: нор
    'thalach' : 'Максимальная частота сердечных сокращений',
    'exang' : 'Стенокардия, вызванная физической нагрузкой (1 = да; 0 = н
    'oldpeak' : 'Угнетение сегмента ST, вызванное физической нагрузкой по
    'slope' : 'Наклон пикового сегмента ST при физической нагрузке 1: вос
    'ca' : 'Количество крупных сосудов (0-3), окрашенных при флоуроскопии
    'thal' : 'Тип дефекта 3 = норма; 6 = фиксированный дефект; 7 = обрати
    'disease_flag' : 'Диагностика заболеваний сердца'
```

```
}

types = {
    'age' : 'Integer',
    'sex' : 'Categorical',
    'cp' : 'Categorical',
    'trestbps' : 'Integer',
    'chol' : 'Integer',
    'fbs' : 'Categorical',
    'restecg' : 'Categorical',
    'thalach' : 'Integer',
    'exang' : 'Categorical',
    'oldpeak' : 'Integer',
    'slope' : 'Categorical',
    'ca' : 'Integer',
    'thal' : 'Categorical',
    'disease_flag' : 'Categorical/Integer'

}

for column in data.columns:
    info_data['Признак'].append(column)
    info_data['Описание на русском'].append(cleveland_about.get(column, '
    info_data['Тип признака'].append(types.get(column, ''))


info_df = pd.DataFrame(info_data)
```

Основные выводы

In [9]:
```
#def make_clickable(val):
#    return f'<a target="_blank" href="{val}">{val}</a>'

#info_df.style.format({'Подробнее': make_clickable})
```

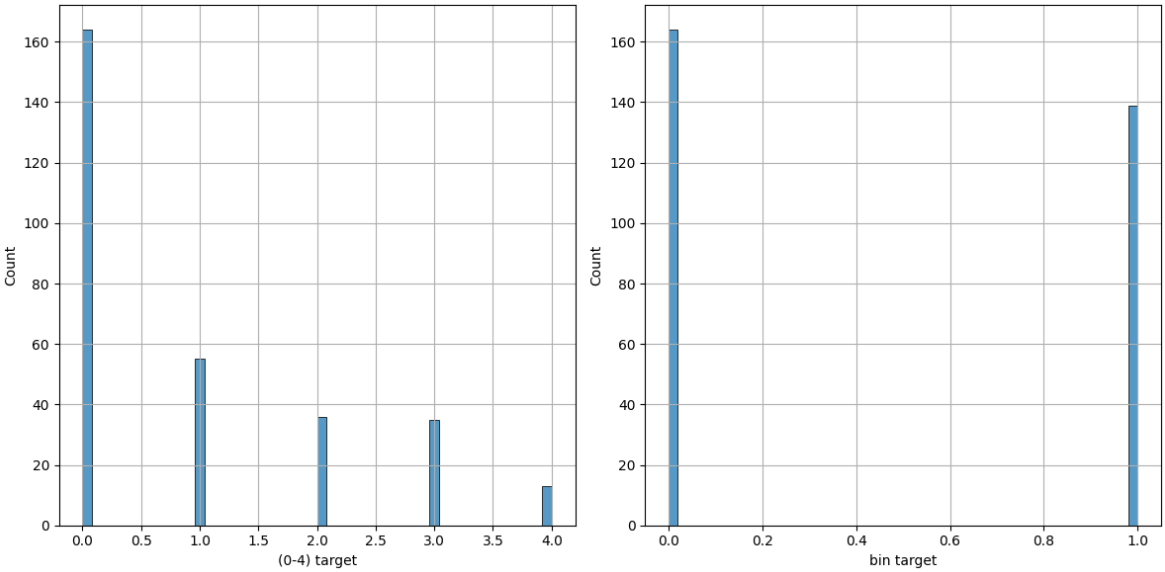
In [10]:
```
pd.set_option('display.max_colwidth', None)
info_df
```

Out[10]:

	Признак	Описание на русском	Тип признака
0	age	Возраст, (1 = мужчина; 0 = женщина)	Integer
1	sex	Пол	Categorical
2	cp	Тип боли в груди (1: типичная ангина, 2: атипичная стенокардия, 3: неангинальная боль, 4: бессимптомная)	Categorical
3	trestbps	Артериальное давление в состоянии покоя (в мм рт. ст. при поступлении в больницу)	Integer
4	chol	Холестерин в сыворотке крови в мг/дл	Integer
5	fbs	Сахар натощак > 120 мг/дл (1 = истина; 0 = ложь)	Categorical
6	restecg	Результаты электрокардиографии в состоянии покоя (0: норма, 1: наличие аномалии ST-T (инверсии T-волны и/или повышение или понижение ST > 0,05 мВ), 2: вероятная или определенная гипертрофия левого желудочка по критериям Эстеса	Categorical
7	thalach	Максимальная частота сердечных сокращений	Integer
8	exang	Стенокардия, вызванная физической нагрузкой (1 = да; 0 = нет)	Categorical
9	oldpeak	Угнетение сегмента ST, вызванное физической нагрузкой по сравнению с покоем	Integer
10	slope	Наклон пикового сегмента ST при физической нагрузке 1: восходящий, 2: плоский, 3: нисходящий	Categorical
11	ca	Количество крупных сосудов (0-3), окрашенных при флоуроскопии	Integer
12	thal	Тип дефекта 3 = норма; 6 = фиксированный дефект; 7 = обратимый дефект	Categorical
13	disease_flag	Диагностика заболеваний сердца	Categorical/Integer
14	bin_flag		

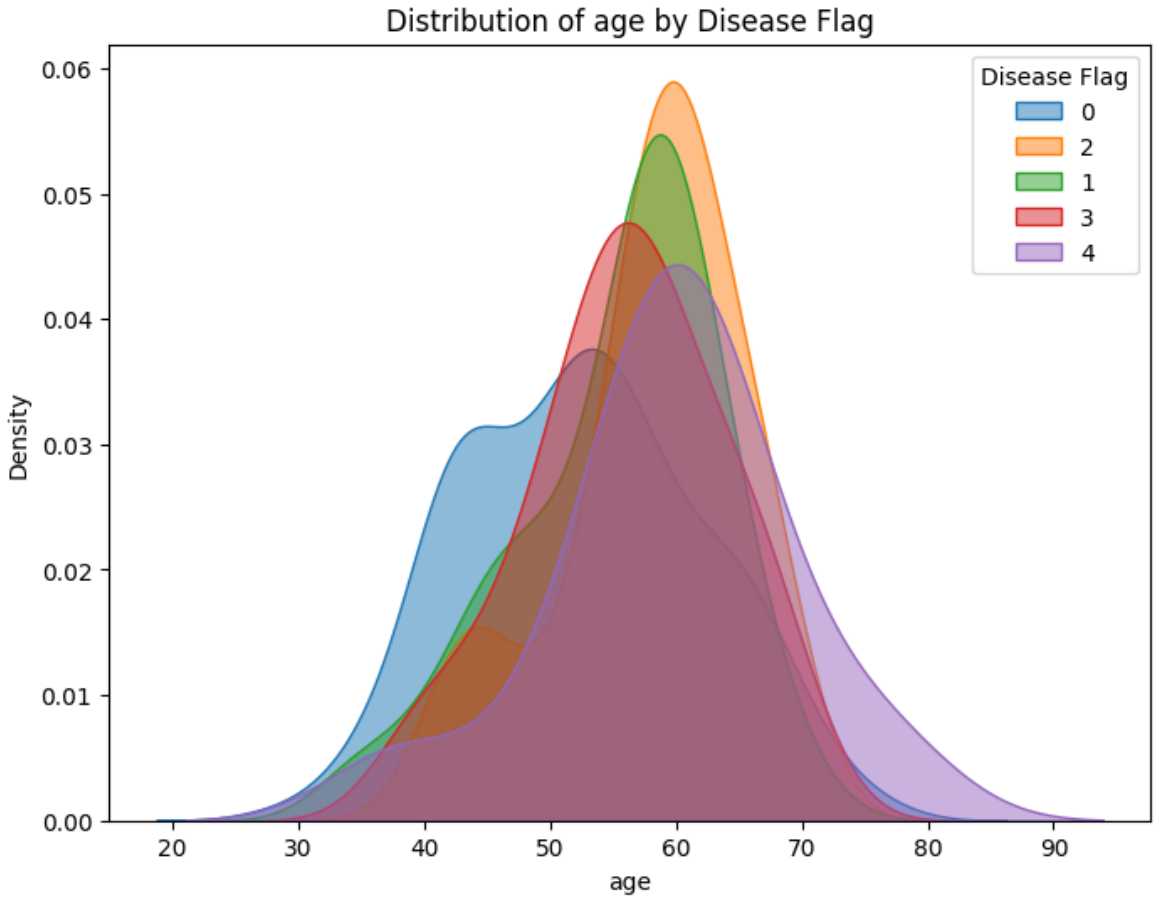
1.2. Визуальный анализ признаков

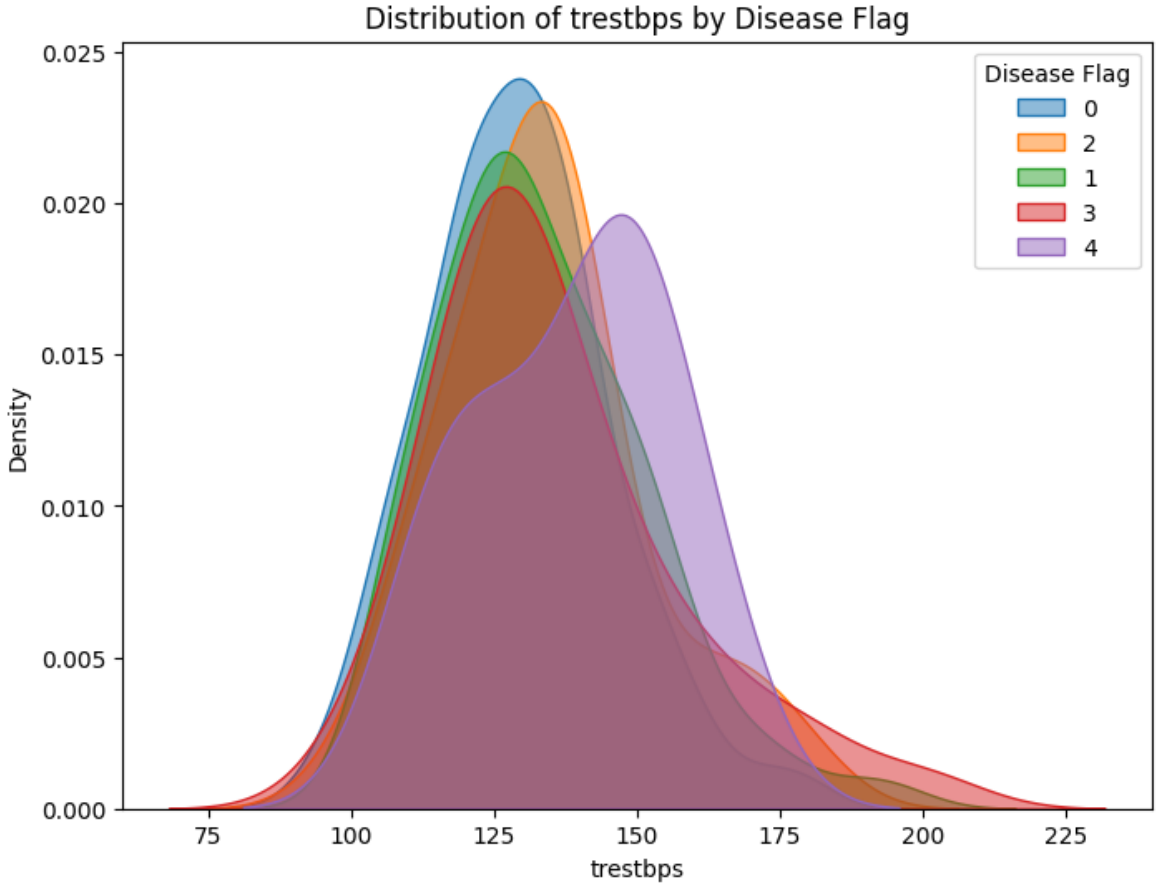
In [11]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

sns.histplot(x='disease_flag', data=data, bins=50, kde=False, ax=axes[0])
axes[0].set_xlabel('(0-4) target')
axes[0].grid()

sns.histplot(x='bin_flag', data=data, bins=50, kde=False, ax=axes[1])
axes[1].set_xlabel('bin target')
axes[1].grid()

plt.tight_layout()
plt.show()
```

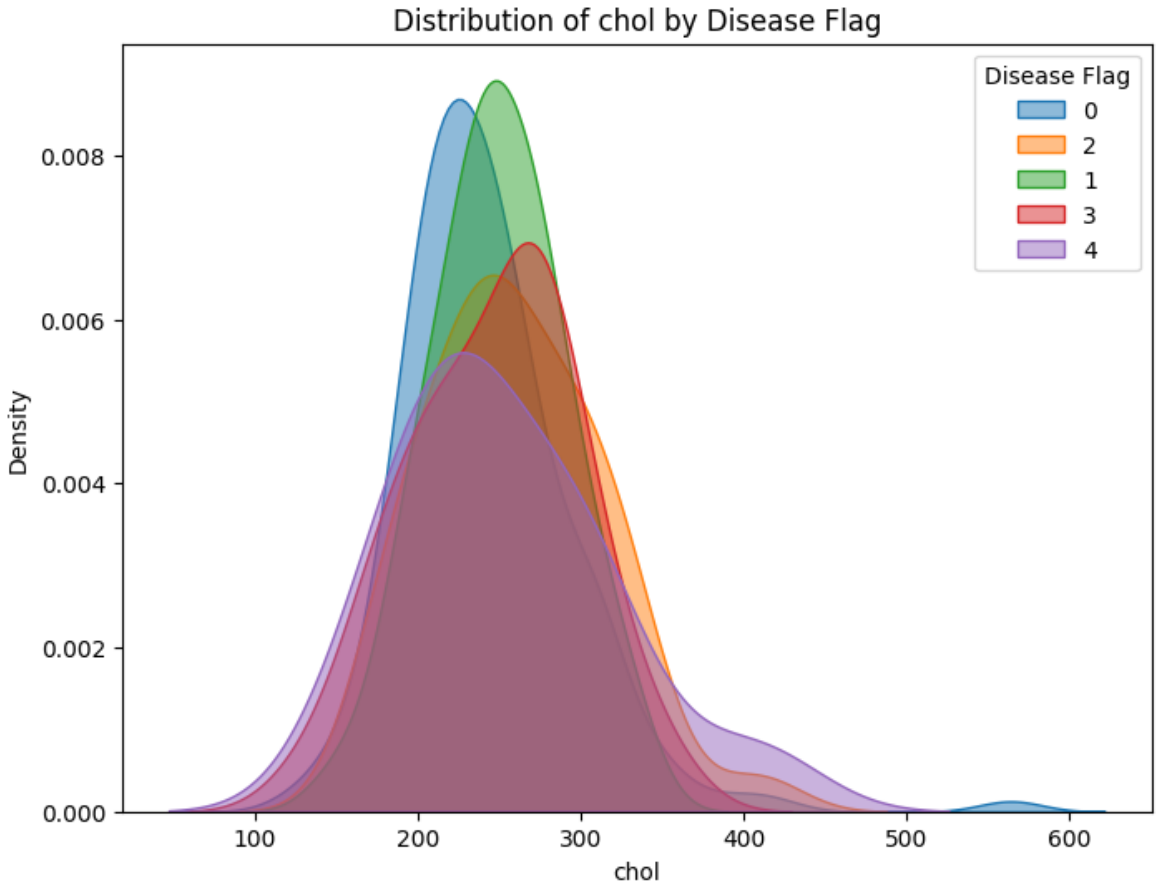


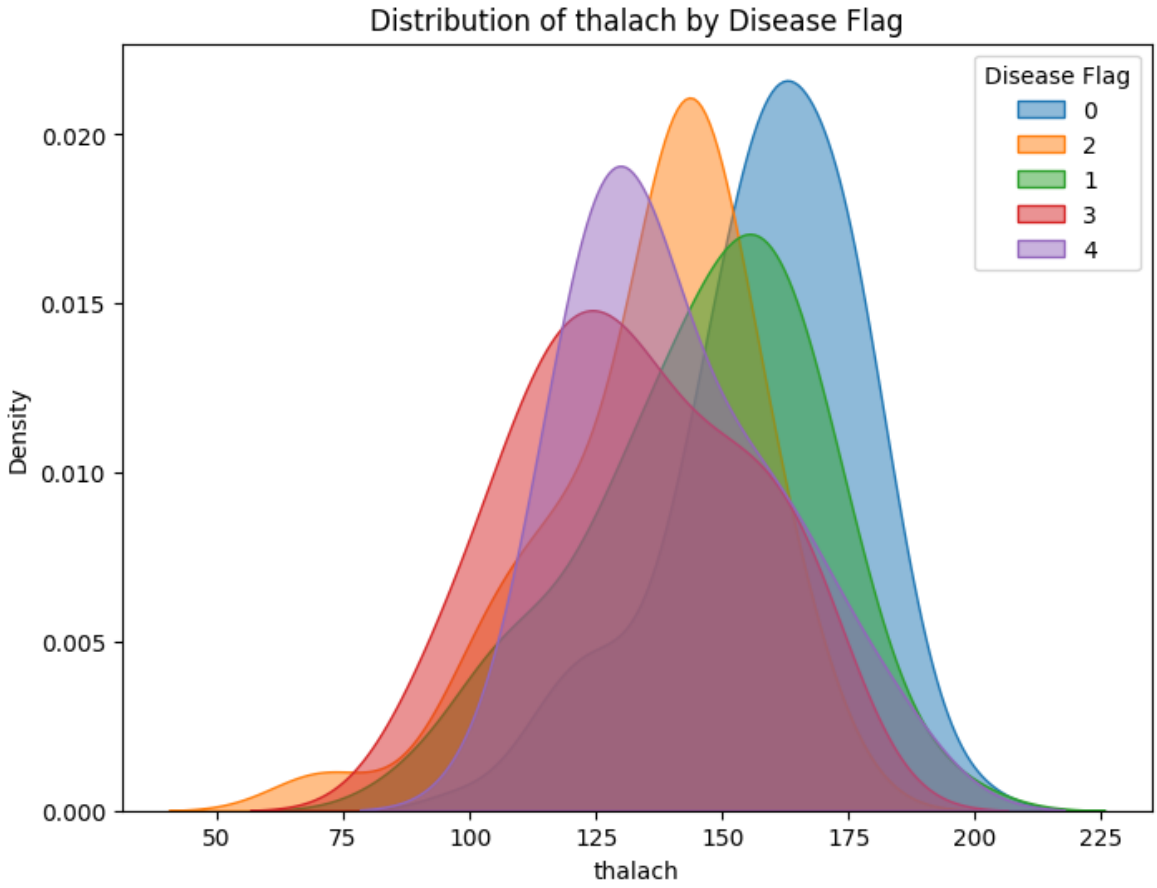
Понаблюдаем расппределение числовых признаков для разных значений таргета

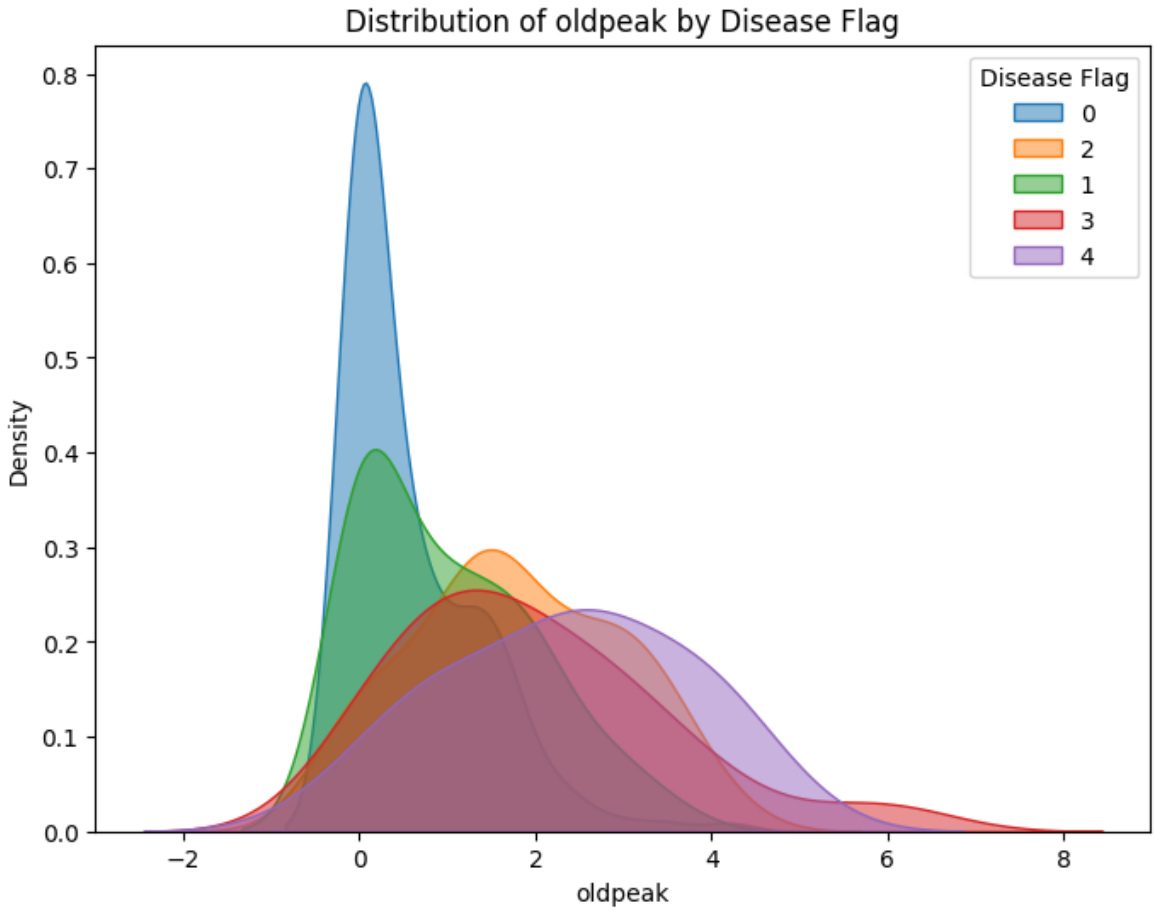
In [12]:
```
numeric_cols_indices = [0, 3, 4, 7, 9, 11]
numeric_cols = data.columns[numeric_cols_indices]

#numeric_cols = data.columns[:-2]

for col in numeric_cols:
    plt.figure(figsize=(8, 6))
    for label in data['disease_flag'].unique():
        subset = data[data['disease_flag'] == label]
        sns.kdeplot(subset[col], label=label, shade=True, alpha=0.5)
    plt.title(f'Distribution of {col} by Disease Flag')
    plt.xlabel(col)
    plt.ylabel('Density')
    plt.legend(title='Disease Flag')
    plt.show()
```

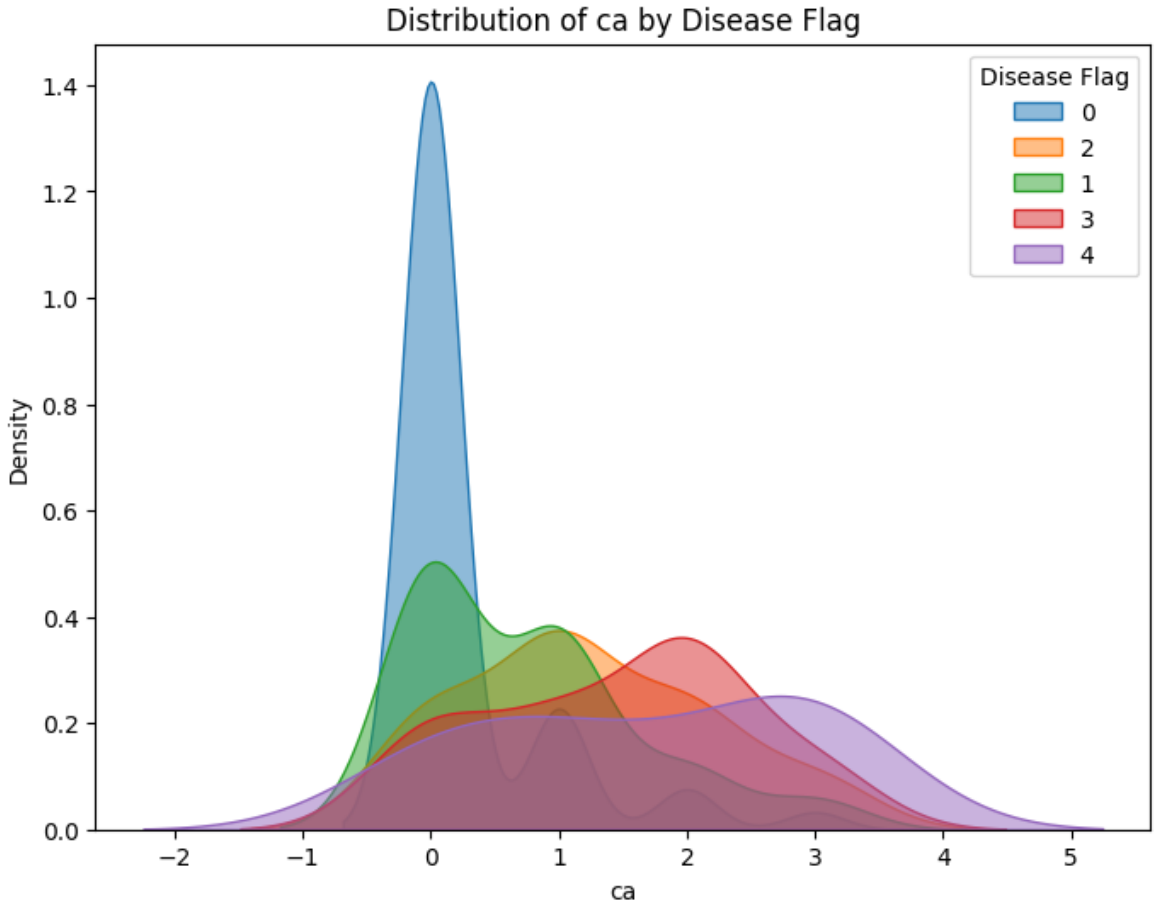

Distribution of age by Disease Flag
Disease Flag
0
2
1
3
4
Density
0.06
0.05
0.04
0.03
0.02
0.01
0.00
20
30
40
50
60
70
80
90
age


Distribution of trestbps by Disease Flag
Disease Flag
0
2
1
3
4
Density
0.025
0.020
0.015
0.010
0.005
0.000
75
100
125
150
175
200
225
trestbps

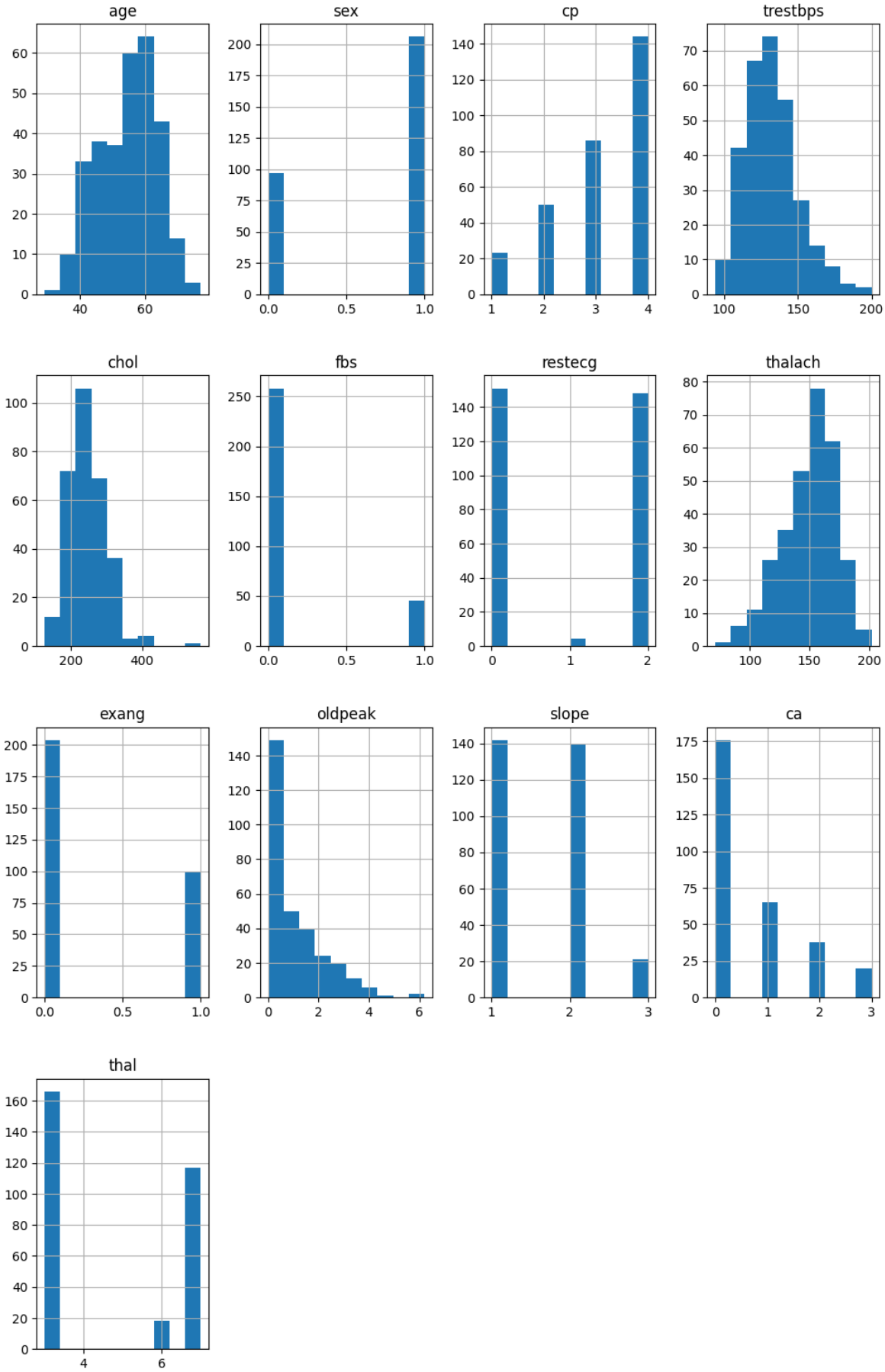

Distribution of chol by Disease Flag
Disease Flag
0
2
1
3
4
Density
0.000
0.002
0.004
0.006
0.008
100
200
300
400
500
600
chol


Distribution of thalach by Disease Flag
Disease Flag
0
2
1
3
4
Density
0.000
0.005
0.010
0.015
0.020
50
75
100
125
150
175
200
225
thalach





In [13]:
```
data.iloc[:, :-2].hist(figsize=(12, 19), sharex=False);
```

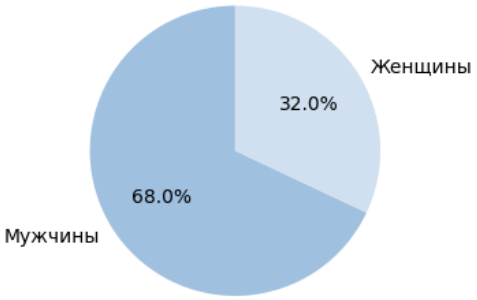


In [14]:
```
men = data[(data['sex'] == 1)].shape[0]
women = data[(data['sex'] == 0)].shape[0]

cp_1 = data[(data['cp'] == 1)].shape[0]
cp_2 = data[(data['cp'] == 2)].shape[0]
```

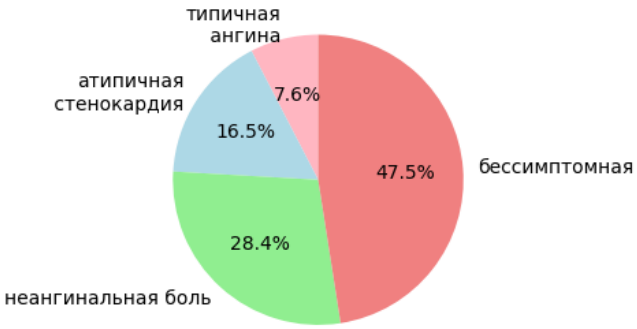
```
cp_3 = data[(data['cp'] == 3)].shape[0]
cp_4 = data[(data['cp'] == 4)].shape[0]

exang_0 = data[(data['exang'] == 0)].shape[0]
exang_1 = data[(data['exang'] == 1)].shape[0]

slope_1 = data[(data['slope'] == 1)].shape[0]
slope_2 = data[(data['slope'] == 2)].shape[0]
slope_3 = data[(data['slope'] == 3)].shape[0]

ca_0 = data[(data['ca'] == 0)].shape[0]
ca_1 = data[(data['ca'] == 1)].shape[0]
ca_2 = data[(data['ca'] == 2)].shape[0]
ca_3 = data[(data['ca'] == 3)].shape[0]

thal_3 = data[(data['thal'] == 3)].shape[0]
thal_6 = data[(data['thal'] == 6)].shape[0]
thal_7 = data[(data['thal'] == 7)].shape[0]

target_0 = data[(data['disease_flag'] == 0)].shape[0]
target_1 = data[(data['disease_flag'] == 1)].shape[0]
target_2 = data[(data['disease_flag'] == 2)].shape[0]
target_3 = data[(data['disease_flag'] == 3)].shape[0]
target_4 = data[(data['disease_flag'] == 4)].shape[0]

bt_0 = data[(data['bin_flag'] == 0)].shape[0]
bt_1 = data[(data['bin_flag'] == 1)].shape[0]

fbs_0 = data[(data['fbs'] == 0)].shape[0]
fbs_1 = data[(data['fbs'] == 1)].shape[0]

res_0 = data[(data['restecg'] == 0)].shape[0]
res_1 = data[(data['restecg'] == 1)].shape[0]
res_2 = data[(data['restecg'] == 2)].shape[0]

fig, axes = plt.subplots(5, 2, figsize=(10, 20))

men_colors = ['#A0C0E0', '#D0E0F0']
res_colors = ['ivory', 'lightslategray', 'darkred']
cp_colors = ["lightpink", "lightblue", "lightgreen", "lightcoral"]
exang_colors = ["palegoldenrod", "lightskyblue"]
slope_colors = ["lightpink", "lightblue", "lightgreen"]
ca_colors = ["lightcyan", "lightcoral", "palegoldenrod", "thistle"]
thal_colors = ["thistle", "lightseagreen", "lavender"]
target_colors = ["lightpink", "lightblue", "lightgreen", "lightcoral", "l
bt_colors = ['lightblue', 'lavender']
fbs_colors = ["lightpink", "palegoldenrod"]

axes[0, 0].pie([men, women], labels=['Мужчины', 'Женщины'], autopct='%1.1
axes[0, 0].set_title('Распределение по полу\n (sex)')

axes[0, 1].pie([cp_1, cp_2, cp_3, cp_4], labels=['типичная\n ангина', 'ат
axes[0, 1].set_title('Распределение по типу\n боли в груди (cp)')

axes[1, 0].pie([exang_0, exang_1], labels=['да', 'нет'], autopct='%1.1f%%
axes[1, 0].set_title('Распределение по стенокардии,\n вызванной физическо

axes[1, 1].pie([slope_1, slope_2, slope_3], labels=['восходящий', 'плоски
axes[1, 1].set_title('Распределение по наклону пикового\n сегмента ST при
```

```
axes[2, 0].pie([thal_3, thal_6, thal_7], labels=['норма', 'фиксированный\
axes[2, 0].set_title('Распределение по типу дефекта\n (thal)')

axes[2, 1].pie([fbs_0, fbs_1], labels=['0 (<120 мг/дл)', '1 (>120 мг/дл)'
axes[2, 1].set_title('Распределение по уровню сахара\n в крови (fbs)')

axes[3, 0].pie([res_0, res_1, res_2], labels=['0 (норма)', '1 (наличие ан
axes[3, 0].set_title('Распределение по ЭКГ (restecg)')

axes[3, 1].pie([target_0, target_1, target_2, target_3, target_4], labels
axes[3, 1].set_title('Распределение по диагнозу\n (disease_flag)')

axes[4, 0].pie([bt_0, bt_1], labels=['0', '1'], autopct='%1.1f%%', starta
axes[4, 0].set_title('Распределение по бин диагнозу\n (bin_flag)')

fig.delaxes(axes[4, 1])

plt.show()
plt.tight_layout()
```

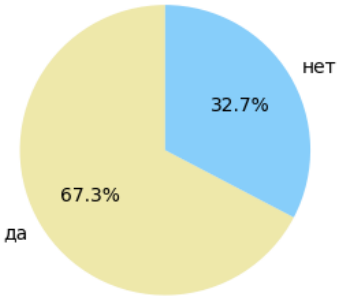
Распределение по полу (sex)



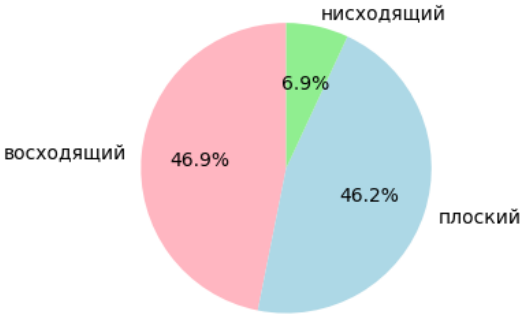
Распределение по типу боли в груди (cp)



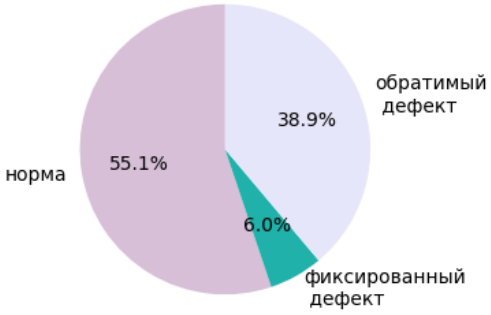
Распределение по стенокардии, вызванной физической нагрузкой (exang)



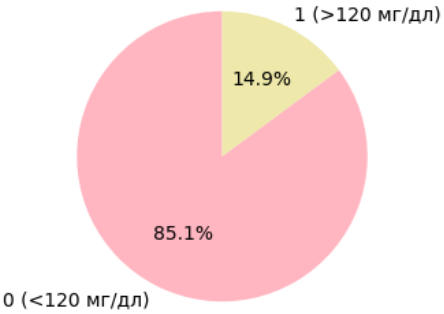
Распределение по наклону пикового сегмента ST при физической нагрузке (slope)



Распределение по типу дефекта (thal)

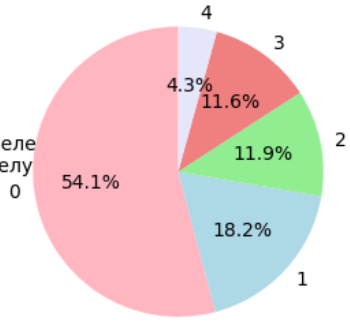


Распределение по уровню сахара в крови (fbs)

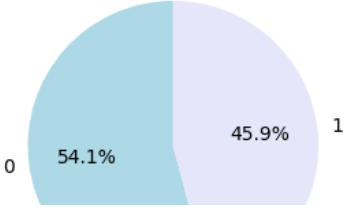


Распределение по ЭКГ (restecg)

Распределение по диагнозу (disease_flag)



Распределение по бин диагнозу (bin_flag)



```
<Figure size 640x480 with 0 Axes>
```

Основные выводы:

После наблюдения распределений и размерностей всех признаков, можно заметить следующее:

```
age  -     Integer     - нормируем - std scaler
sex  -     Categorical - OHE (ухудшает, оставляем 0 и 1)
cp   -     Categorical - OHE
trestbps - Integer     - нормируем - minmax
chol  -    Integer     - нормируем - minmax
fbs  -     Categorical - упорядочиваем
restecg  - Categorical - OHE
thalach -  Integer     - нормируем - minmax
exang  -   Categorical - OHE (ухудшает, оставляем 0 и 1)
oldpeak -  Integer     - нормируем делением на max
slope -    Categorical - упорядочиваем
ca   -     Integer     - делим на max
thal  -    Categorical - OHE
```

На основании масштабов данных и их распределений далее будем провдить нормировки

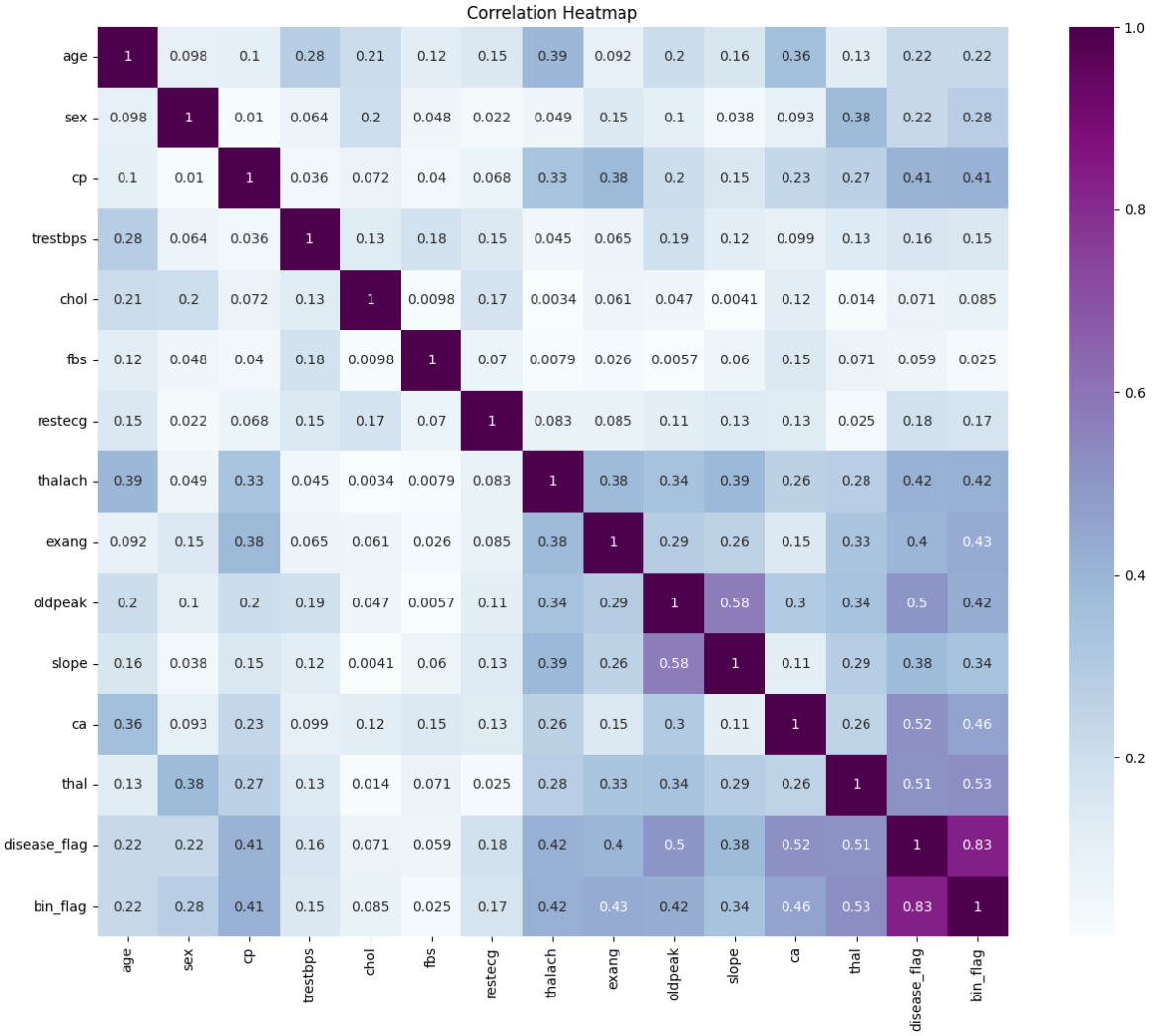
1.3. Посмотрим на скоррелированность данных

In [15]:
```
corr = abs(data.corr())
corr
```

Out[15]:

	age	sex	cp	trestbps	chol	fbs	restecg	
age	1.000000	0.097542	0.104139	0.284946	0.208950	0.118530	0.148868	0
sex	0.097542	1.000000	0.010084	0.064456	0.199915	0.047862	0.021647	0
cp	0.104139	0.010084	1.000000	0.036077	0.072319	0.039975	0.067505	0
trestbps	0.284946	0.064456	0.036077	1.000000	0.130120	0.175340	0.146560	0
chol	0.208950	0.199915	0.072319	0.130120	1.000000	0.009841	0.171043	0
fbs	0.118530	0.047862	0.039975	0.175340	0.009841	1.000000	0.069564	0
restecg	0.148868	0.021647	0.067505	0.146560	0.171043	0.069564	1.000000	0
thalach	0.393806	0.048663	0.334422	0.045351	0.003432	0.007854	0.083389	1
exang	0.091661	0.146201	0.384060	0.064762	0.061310	0.025665	0.084867	0
oldpeak	0.203805	0.102173	0.202277	0.189171	0.046564	0.005747	0.114133	0
slope	0.161770	0.037533	0.152050	0.117382	0.004062	0.059894	0.133946	0
ca	0.362605	0.093185	0.233214	0.098773	0.119000	0.145478	0.128343	0
thal	0.127389	0.380936	0.265246	0.133554	0.014214	0.071358	0.024531	0
disease_flag	0.222853	0.224469	0.407075	0.157754	0.070909	0.059186	0.183696	0
bin_flag	0.223120	0.276816	0.414446	0.150825	0.085164	0.025264	0.169202	0

In [16]:
```
plt.figure(figsize=(16, 12))
sns.heatmap(corr, square=True, annot=True, cmap='BuPu')
plt.title('Correlation Heatmap')
plt.show()
```



Основные выводы:

Есть небольшая корреляция признаков, но не более 0.6. Удалять признаки не целесообразно, будем использовать регуляризацию в дальнейшем и кодировать категориальные признаки

In [17]:
```
data['age'].describe()
```

Out[17]:
```
count    303.000000
mean      54.438944
std        9.038662
min       29.000000
25%       48.000000
50%       56.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

In [18]:
```
data['sex'].describe()
```

Out[18]:
```
count    303.000000
mean       0.679868
std        0.467299
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: sex, dtype: float64
```

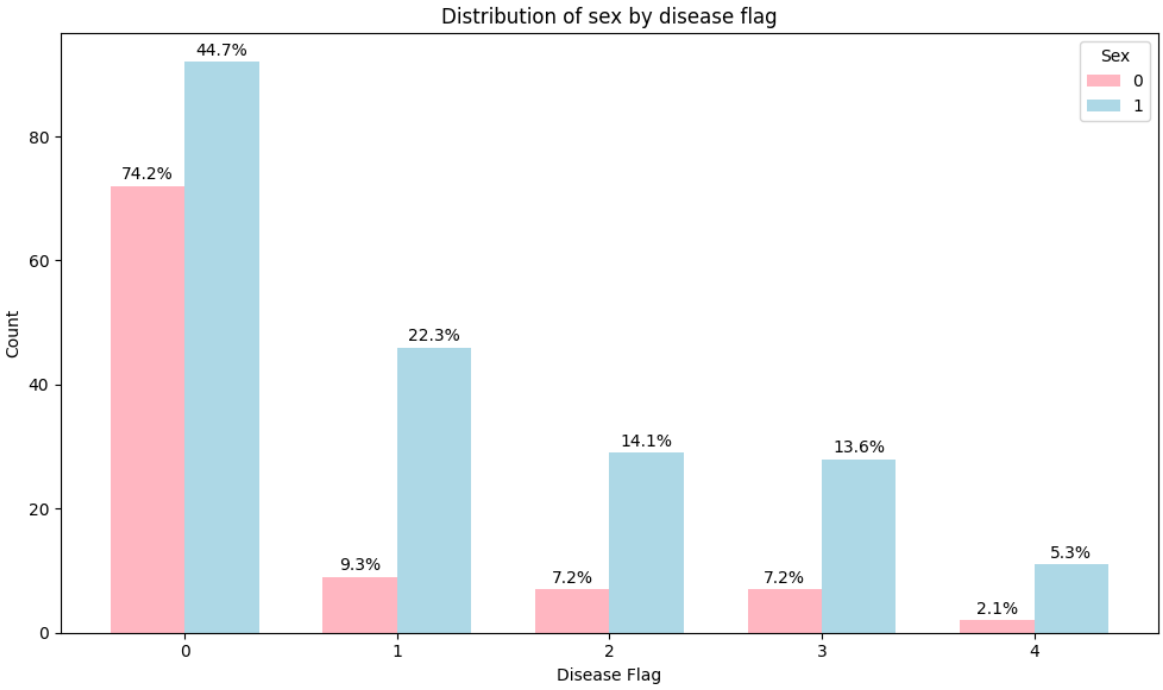
1.4. Посмотрим совместное распределение признаков и таргета

Смотрим, влияет ли конкретный ПОЛ на степень болезни

In [19]:
```
data[data['sex'] == 0]['sex'].count()
```

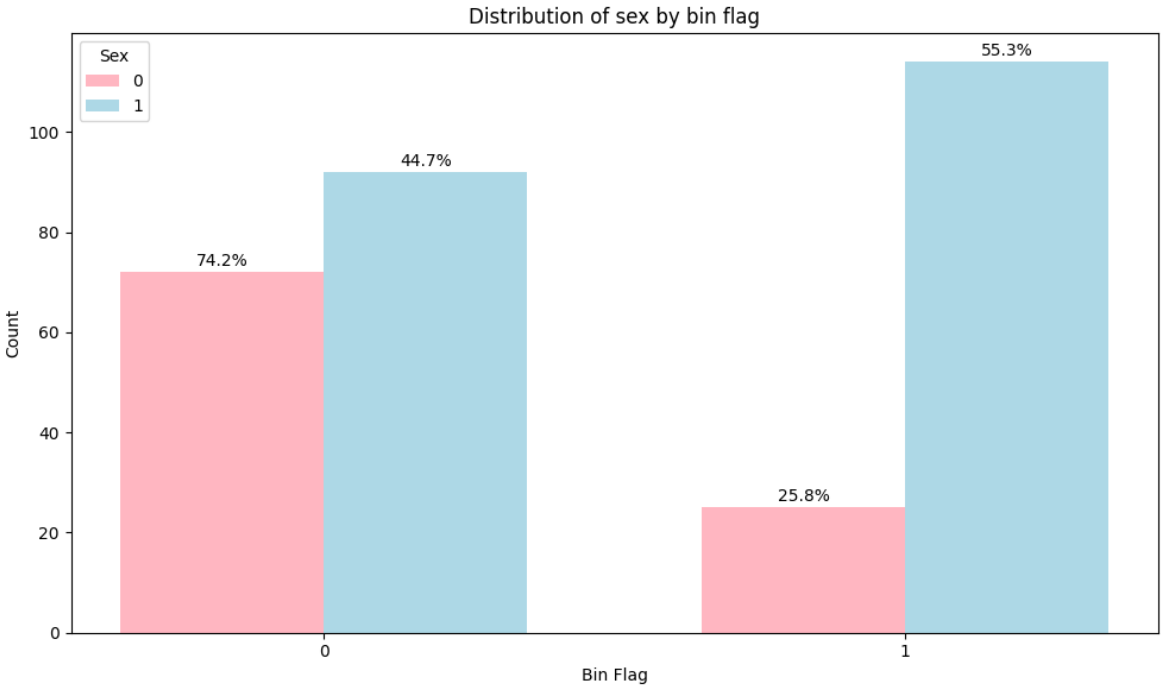
Out[19]:
```
97
```

In [20]:
```
grouped = data.groupby(['disease_flag', 'sex']).size().unstack(fill_value

fig, ax = plt.subplots(figsize=(10, 6))
colors = ['lightpink', 'lightblue']

width = 0.35
indices = range(len(grouped))

total_counts = grouped.sum(axis=0)

for i, sex in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[sex], width, la
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[sex] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Disease Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of sex by disease flag')
ax.set_xticks([x + width/2 for x in indices])
ax.set_xticklabels(grouped.index)
ax.legend(title='Sex')

plt.tight_layout()
plt.show()
```



Примечание: здесь процент - от общего кол-ва мужчин для голубых столцов, а для розовых - от общего кол-ва женщин

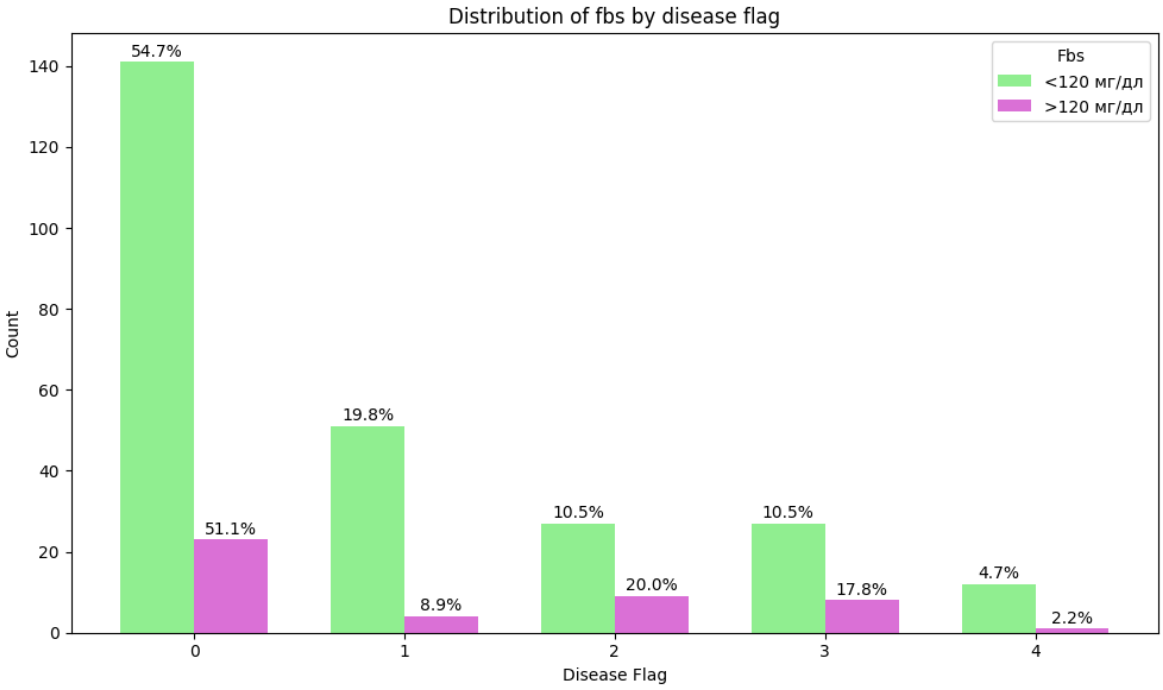
In [21]:
```
grouped = data.groupby(['bin_flag', 'sex']).size().unstack(fill_value=0)

fig, ax = plt.subplots(figsize=(10, 6))
indices = range(len(grouped))

for i, sex in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[sex], width, la
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[sex] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Bin Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of sex by bin flag')
ax.set_xticks([x + width/2 for x in indices])
ax.set_xticklabels(grouped.index)
ax.legend(title='Sex')

plt.tight_layout()
plt.show()
```



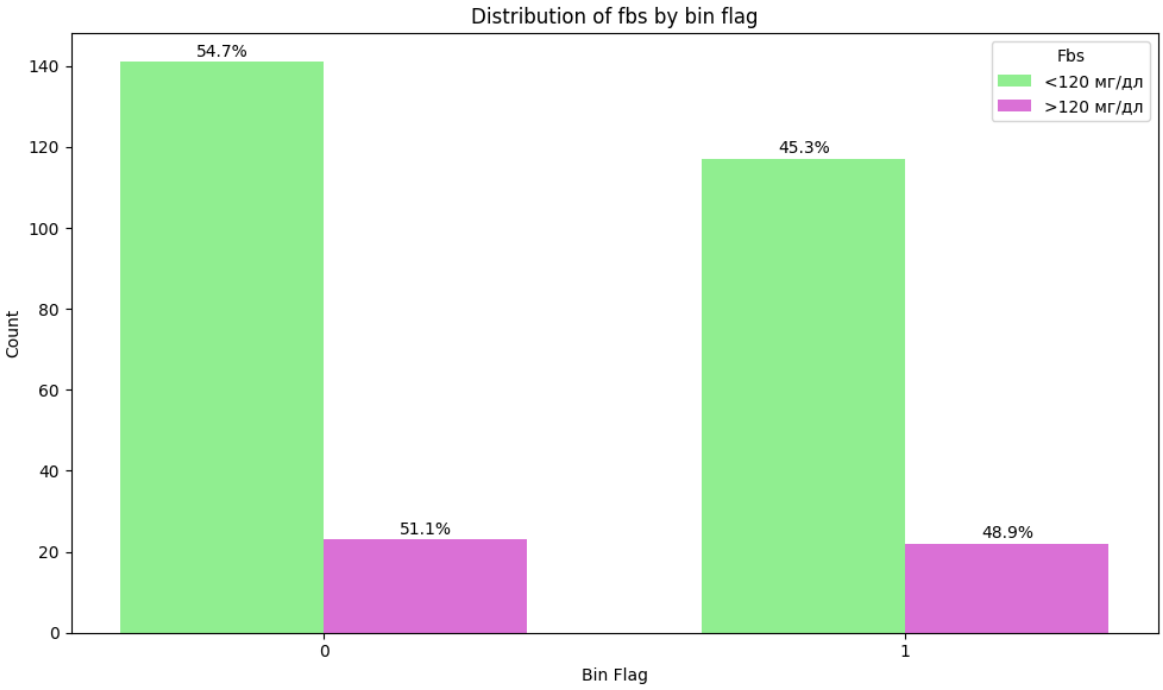
Как видим, влияет - женское сердце более живучее :) Нужно будет сохранить распределение при делении на трейн и тест

Аналогично оценим влияние 'fbs' : 'Сахар натощак > 120 мг/дл (1 = истина; 0 = ложь)',

In [22]:
```
grouped = data.groupby(['disease_flag', 'fbs']).size().unstack(fill_value

fig, ax = plt.subplots(figsize=(10, 6))
colors = ['lightgreen', 'orchid']
fbs_lab = ['<120 мг/дл', '>120 мг/дл']

width = 0.35
indices = range(len(grouped))

total_counts = grouped.sum(axis=0)

for i, fbs in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[fbs], width, la
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[fbs] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Disease Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of fbs by disease flag')
ax.set_xticks([x + width/2 for x in indices])
ax.set_xticklabels(grouped.index)
ax.legend(title='Fbs')

plt.tight_layout()
plt.show()
```

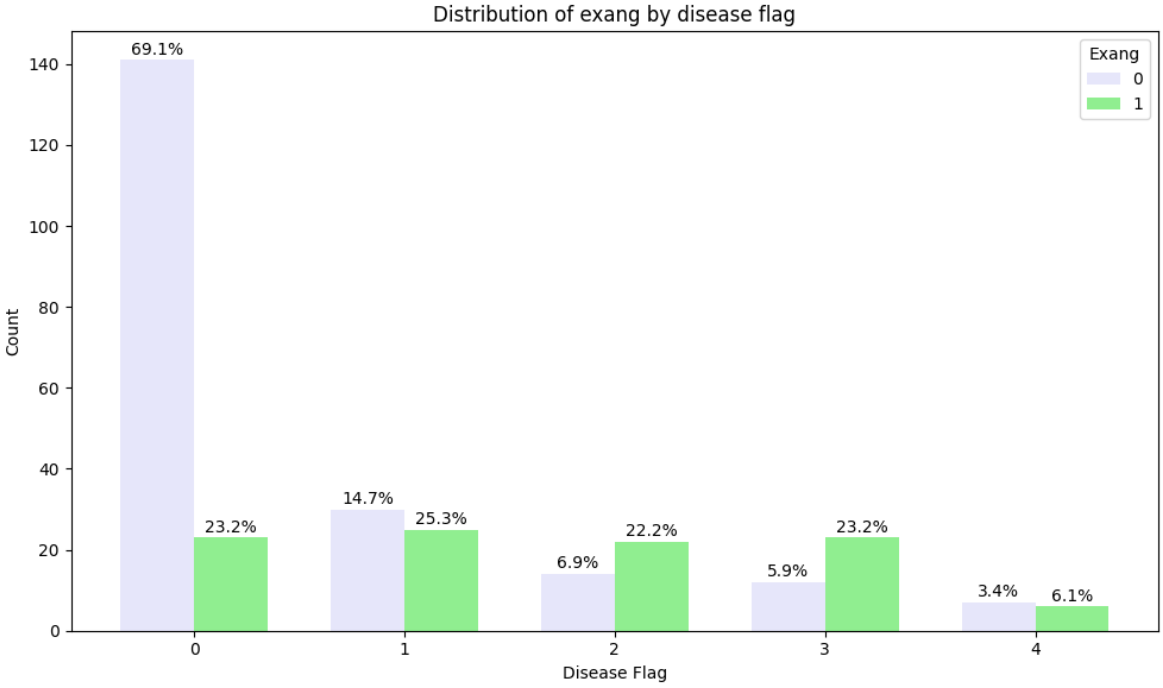


In [23]:
```
grouped = data.groupby(['bin_flag', 'fbs']).size().unstack(fill_value=0)

fig, ax = plt.subplots(figsize=(10, 6))
indices = range(len(grouped))

for i, fbs in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[fbs], width, la
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[fbs] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Bin Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of fbs by bin flag')
ax.set_xticks([x + width/2 for x in indices])
ax.set_xticklabels(grouped.index)
ax.legend(title='Fbs')

plt.tight_layout()
plt.show()
```

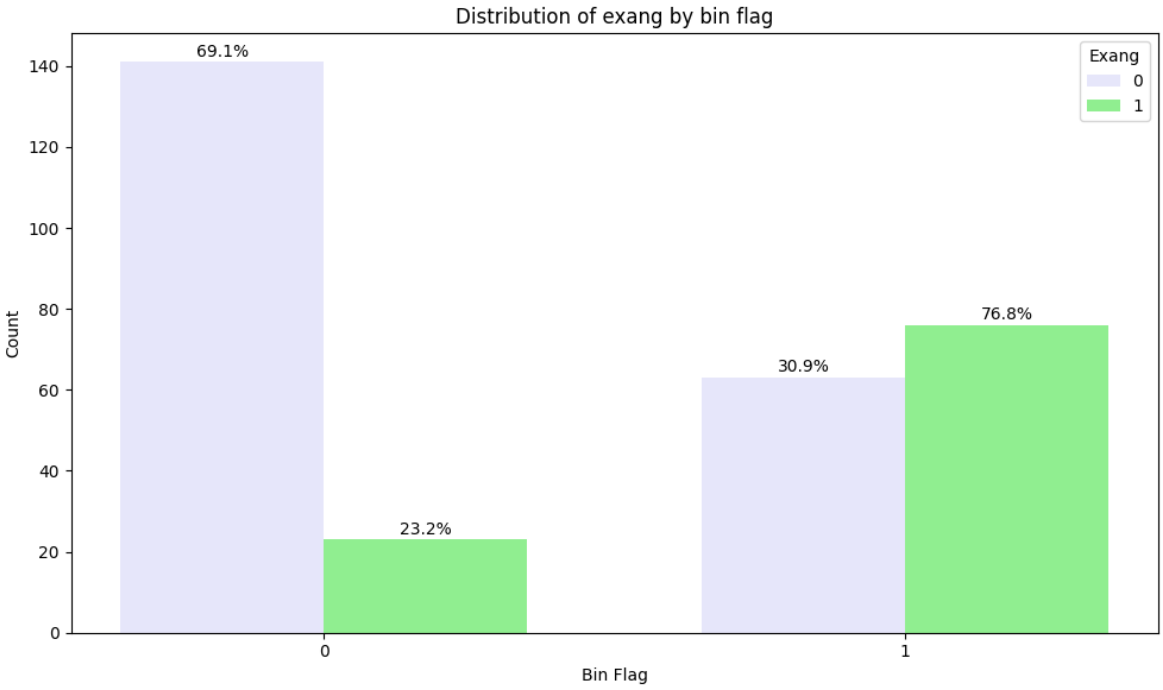


Аналогично понаблюдаем за exang - Стенокардия, вызванная физической нагрузкой (1 = да; 0 = нет)

In [24]:
```
grouped = data.groupby(['disease_flag', 'exang']).size().unstack(fill_val

fig, ax = plt.subplots(figsize=(10, 6))
colors = ['lavender', 'lightgreen']
indices = range(len(grouped))

total_counts = grouped.sum(axis=0)

for i, exang in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[exang], width,
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[exang] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Disease Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of exang by disease flag')
ax.set_xticks([x + width/2 for x in indices])
ax.set_xticklabels(grouped.index)
ax.legend(title='Exang')

plt.tight_layout()
plt.show()
```



In [25]:
```
grouped = data.groupby(['bin_flag', 'exang']).size().unstack(fill_value=0

fig, ax = plt.subplots(figsize=(10, 6))
indices = range(len(grouped))

for i, exang in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[exang], width,
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[exang] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Bin Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of exang by bin flag')
ax.set_xticks([x + width/2 for x in indices])
ax.set_xticklabels(grouped.index)
ax.legend(title='Exang')

plt.tight_layout()
plt.show()
```

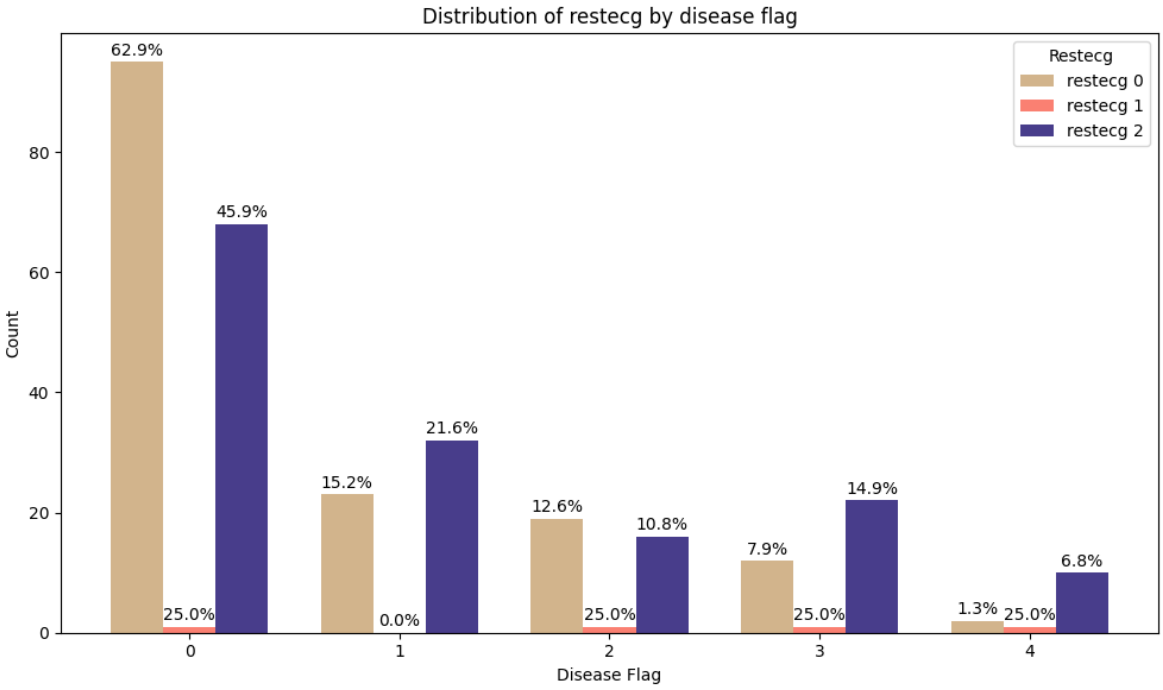


Будем нормировать

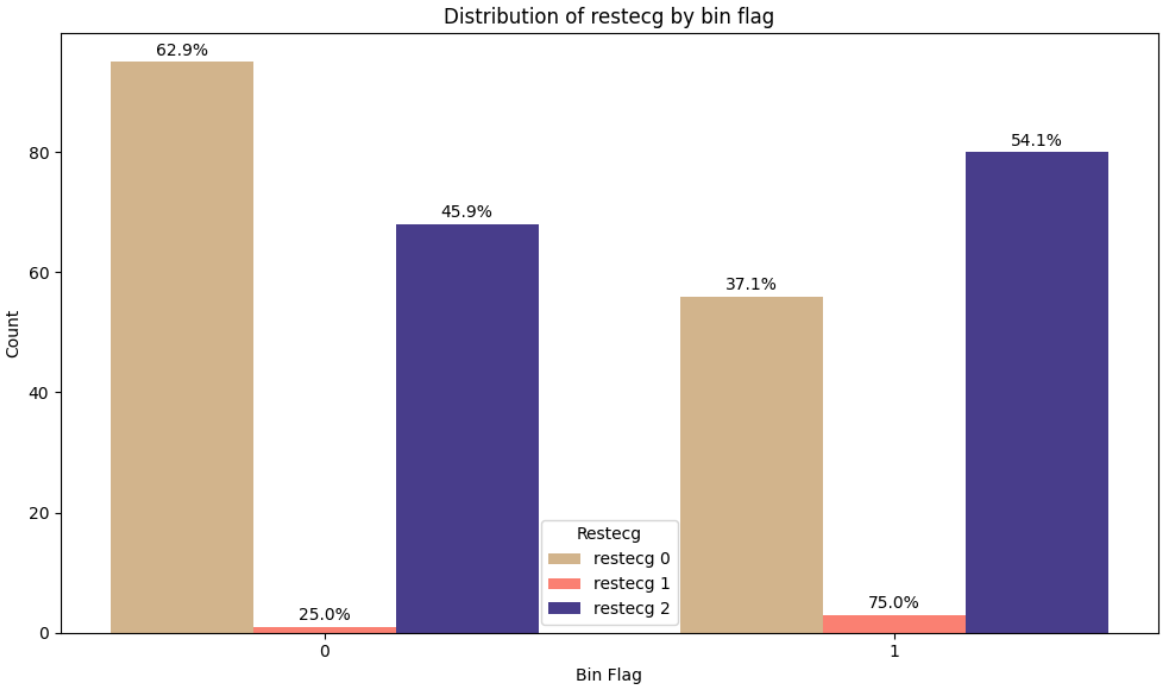
Видим, что влияет на степень значительно

Аналогично понаблюдаем за restecg - 'Результаты электрокардиографии в состоянии покоя (0: норма, 1: наличие аномалии ST-T (инверсии T-волны и/или повышение или понижение ST > 0,05 мВ), 2: вероятная или определенная гипертрофия левого желудочка по критериям Эстеса'

In [26]:
```
grouped = data.groupby(['disease_flag', 'restecg']).size().unstack(fill_v

fig, ax = plt.subplots(figsize=(10, 6))
colors = ['tan', 'salmon', 'darkslateblue']
width = 0.25

indices = range(len(grouped))

total_counts = grouped.sum(axis=0)

for i, restecg_value in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[restecg_value],
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[restecg_value] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Disease Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of restecg by disease flag')
ax.set_xticks([x + width for x in indices])
ax.set_xticklabels(grouped.index)
ax.legend(title='Restecg')

plt.tight_layout()
plt.show()
```



In [27]:
```
grouped = data.groupby(['bin_flag', 'restecg']).size().unstack(fill_value

fig, ax = plt.subplots(figsize=(10, 6))
indices = range(len(grouped))

for i, restecg_value in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[restecg_value],
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[restecg_value] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Bin Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of restecg by bin flag')
ax.set_xticks([x + width for x in indices])
ax.set_xticklabels(grouped.index)
ax.legend(title='Restecg')

plt.tight_layout()
plt.show()
```

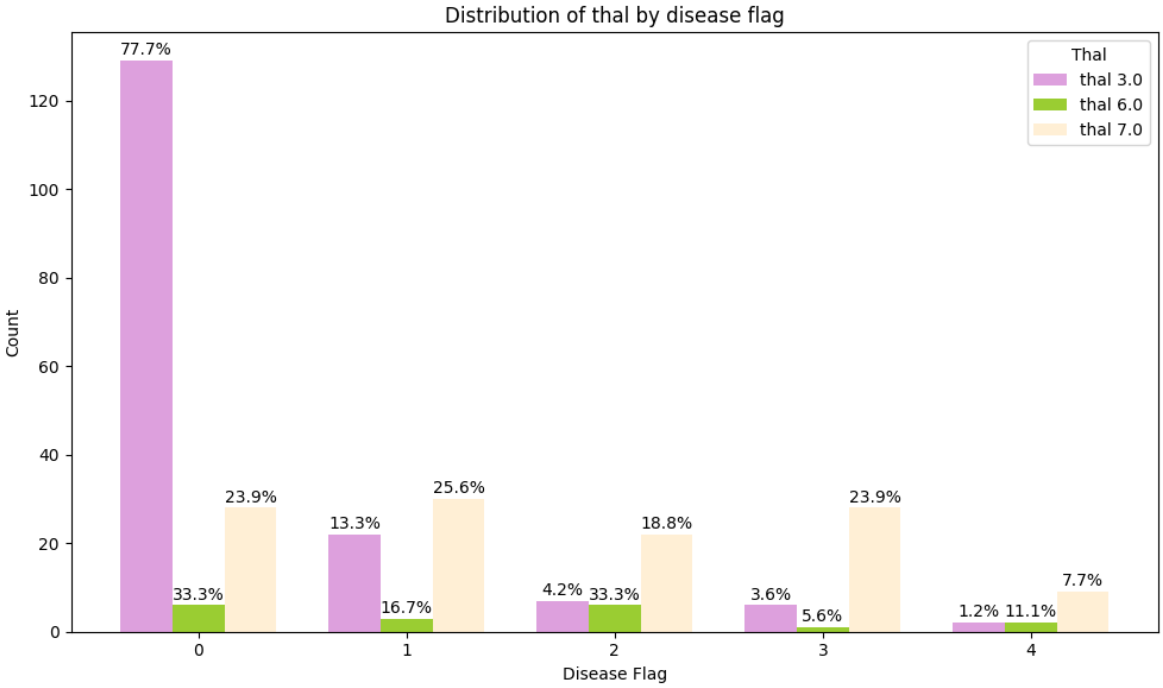


Видим, что при обоих вариантах классификации значения restecg распределены вполне равномерно, разумно объединить 1 и 2 значения для большей сбалансированности, у нас как раз будет 0 - норма и 1 - не норма ('restecg' : 'Результаты электрокардиографии в состоянии покоя (0: норма, 1: наличие аномалии ST-T (инверсии Т-волны и/или повышение или понижение ST > 0,05 мВ), 2: вероятная или определенная гипертрофия левого желудочка по критериям Эстеса')

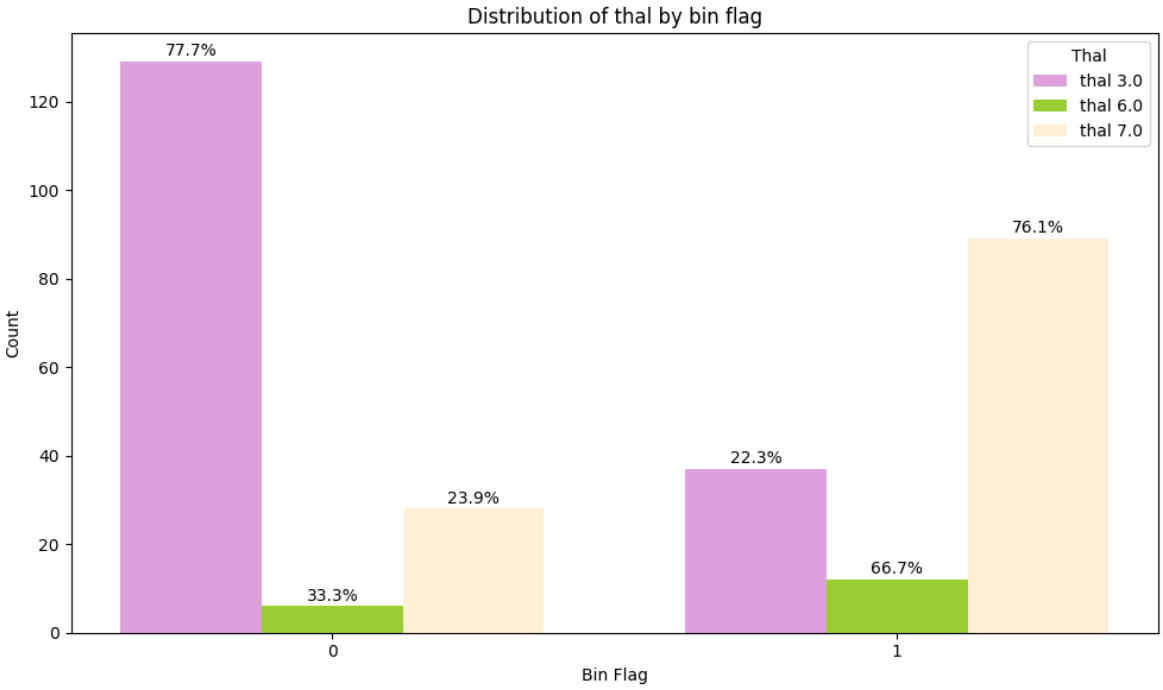
Аналогично понаблюдаем за thal Тип дефекта 3 = норма; 6 = фиксированный дефект; 7 = обратимый дефект

In [28]:
```
grouped = data.groupby(['disease_flag', 'thal']).size().unstack(fill_valu

fig, ax = plt.subplots(figsize=(10, 6))

colors = ['plum', 'yellowgreen', 'papayawhip']

width = 0.25

indices = range(len(grouped))

total_counts = grouped.sum(axis=0)

for i, thal in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[thal], width, l
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[thal] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Disease Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of thal by disease flag')
ax.set_xticks([x + width for x in indices])
```

```
ax.set_xticklabels(grouped.index)
ax.legend(title='Thal')

plt.tight_layout()
plt.show()
```



In [29]:
```
grouped = data.groupby(['bin_flag', 'thal']).size().unstack(fill_value=0)

fig, ax = plt.subplots(figsize=(10, 6))
indices = range(len(grouped))

for i, thal in enumerate(grouped.columns):
    bars = ax.bar([x + i*width for x in indices], grouped[thal], width, l
    for bar in bars:
        yval = bar.get_height()
        percent = yval / total_counts[thal] * 100
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.5, f'{percent:.

ax.set_xlabel('Bin Flag')
ax.set_ylabel('Count')
ax.set_title('Distribution of thal by bin flag')
ax.set_xticks([x + width for x in indices])
ax.set_xticklabels(grouped.index)
ax.legend(title='Thal')

plt.tight_layout()
plt.show()
```



Заметим, что slope ('Наклон пикового сегмента ST при физической нагрузке 1: восходящий, 2: плоский, 3: нисходящий') неплохо бы упорядочить к -1 0 1

Основные выводы

После наблюдения распределений и размерностей всех признаков, можно заметить следующее:

```
 age  -     Integer     - нормируем - std scaler
 sex  -     Categorical - OHE
 cp   -     Categorical - OHE
 trestbps - Integer     - нормируем - minmax
 chol  -    Integer     - нормируем - minmax
 fbs  -     Categorical - упорядочен, оставляем 0 и 1
 restecg  - Categorical - объединяем 1 и 2, получили
признак с бианрными значениями 0 и 1 - OHE
 thalach -  Integer     - нормируем - minmax
 exang  -   Categorical - OHE
 oldpeak -  Integer     - нормируем делением на max
 slope -    Categorical - упорядочили от 1 2 3 к -1 0 1
 ca   -     Integer     - делим на max
 thal  -    Categorical - к 0, 1, 2 - OHE
```

На основании масштабов данных и их распределений далее будем провдить нормировки

Примечание:

1. Age - надо нормировать
2. Sex - несбалансирован, 68% мужчин и всего 32% женщин, возможно нужно будет дать меньшинству больший вес При делении на train и test надо учесть,

что распределение должно сохраняться, т е делить будем соответствующим образом

3. Cp - тип боли в груди - категориальный, будем кодировать One Hot Encoding
4. Trestbps (Артериальное давление в состоянии покоя (в мм рт. ст. при поступлении в больницу)) - нужно нормировать
5. Chol - нужно нормировать
6. Fbs - несбалансирован при небинарной классификации, возможно нужно будет дать меньшинству больший вес
7. Restecg - сбалансировали, объединив 1 и 2, получили признак с бинарными значениями 0 и 1
8. Thalach - нужно нормировать
9. Exang - значительно влияет на степень болезни. Несбалансирован, возможно нужно будет дать меньшинству больший вес
10. Oldpeak - нужно нормировать
11. Slope - упорядочили от 1 2 3 к -1 0 1
12. Ca - нужно нормировать
13. Thal - влияет на степень болезни, будем использовать One Hot Encoding

1.5. Проверка наличия пропусков в данных

In [30]:
```
missing_data = data.isnull().sum()
print("Пропуски в данных по каждому признаку:")
print(missing_data[missing_data > 0])
```

```
Пропуски в данных по каждому признаку:
ca      4
thal    2
dtype: int64
```

Анализ обоснованности пропусков

In [31]:
```
ca_nan_rows = data[data['ca'].isna()]
ca_nan_rows
```

Out[31]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
166	52	1	3	138	223	0	0	169	0	0.0	1	NaN
192	43	1	4	132	247	1	2	143	1	0.1	2	NaN
287	58	1	2	125	220	0	0	144	0	0.4	2	NaN
302	38	1	3	138	175	0	0	173	0	0.0	1	NaN

In [32]:
```
thal_nan_rows = data[data['thal'].isna()]
thal_nan_rows
```

Out[32]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
87	53	0	3	128	216	0	2	115	0	0.0	1	0.0
266	52	1	4	128	204	1	0	156	1	1.0	2	0.0

Видимо, у этих людей просто не измерили данные показатели, либо данные были утеряны

Заполняем медианой, а можно регрессию попробовать(но недостоверно и мало данных , нельзя отбр данные)

In [33]:
```
data['ca'].fillna(data['ca'].median(), inplace=True)
data['thal'].fillna(data['thal'].median(), inplace=True)
```

In [34]:
```
data.isnull().sum()
```

Out[34]:
```
age             0
sex             0
cp              0
trestbps        0
chol            0
fbs             0
restecg         0
thalach         0
exang           0
oldpeak         0
slope           0
ca              0
thal            0
disease_flag    0
bin_flag        0
dtype: int64
```

2. Подготовка данных

In [35]:
```
data.head(5)
```

Out[35]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	t
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	

2.1. Кодировка категориальных признаков

In [36]:
```
data['slope'] = data['slope'].apply(lambda x: x-2)
```

In [37]:
```
data['restecg'] = data['restecg'].apply(lambda x: 1 if x > 0 else 0)
```

In [38]:
```
data['thal'] = data['thal'].apply(lambda x: math.floor((x-3)/2))
data
```

Out[38]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	1	1	145	233	1	1	150	0	2.3	1	0.0
1	67	1	4	160	286	0	1	108	1	1.5	0	3.0
2	67	1	4	120	229	0	1	129	1	2.6	0	2.0
3	37	1	3	130	250	0	0	187	0	3.5	1	0.0
4	41	0	2	130	204	0	1	172	0	1.4	-1	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
298	45	1	1	110	264	0	0	132	0	1.2	0	0.0
299	68	1	4	144	193	1	0	141	0	3.4	0	2.0
300	57	1	4	130	131	0	0	115	1	1.2	0	1.0
301	57	0	2	130	236	0	1	174	0	0.0	0	1.0
302	38	1	3	138	175	0	0	173	0	0.0	-1	0.0

303 rows × 15 columns

Кодируем категориальные признаки

In [39]:
```
data = pd.get_dummies(data, columns = ['cp', 'thal'])
```

2.2. Нормировка числовых признаков

In [40]:
```
def prepare_input(x_train, x_test):

    STDscalerAge = StandardScaler()
    MMscalerTre = MinMaxScaler()
    MMscalerChol = MinMaxScaler()
    MMscalerThalach = MinMaxScaler()
    OPMax = 0
    CAMax = 0

    x_train.loc[:, 'age'] = STDscalerAge.fit_transform(x_train[['age']])
    x_train.loc[:, 'trestbps'] = MMscalerTre.fit_transform(x_train[['tres
    x_train.loc[:, 'chol'] = MMscalerChol.fit_transform(x_train[['chol']]
    x_train.loc[:, 'thalach'] = MMscalerThalach.fit_transform(x_train[['t
    OPMax = np.max(x_train['oldpeak'])
    x_train.loc[:, 'oldpeak'] = x_train['oldpeak'] / OPMax
    CAMax = np.max(x_train['ca'])
    x_train.loc[:, 'ca'] = x_train['ca'] / CAMax

    x_test.loc[:, 'age'] = STDscalerAge.transform(x_test[['age']])
    x_test.loc[:, 'trestbps'] = MMscalerTre.transform(x_test[['trestbps']
    x_test.loc[:, 'chol'] = MMscalerChol.transform(x_test[['chol']])
```

```
    x_test.loc[:, 'thalach'] = MMscalerThalach.transform(x_test[['thalach
    x_test.loc[:, 'oldpeak'] = x_test['oldpeak'] / OPMax
    x_test.loc[:, 'ca'] = x_test['ca'] / CAMax

    return x_train, x_test
```

2.3. Подготовка метрик

In [41]:
```
def bin_metrics(y_test, y_pred, y_pred_prob):
    cf_matrix = confusion_matrix(y_test, y_pred)
    ac_score = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    RA_score = roc_auc_score(y_test, y_pred_prob)
    return cf_matrix, ac_score, prec, rec, f1, RA_score
```

In [42]:
```
def metrics(y_test, y_pred):
    cf_matrix = confusion_matrix(y_test, y_pred, labels=[0, 1, 2, 3, 4])
    ac_score = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, average='weighted')
    rec = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
    return cf_matrix, ac_score, prec, rec, f1
```

3. Обучение и тестирование модели

Поскольку данных очень мало (всего 300 примеров), а также есть несбалансированные признаки и малопредставленные признаки, которые существенно влияют на целевую переменную, будем использовать кросс-валидацию для оценки качества модели

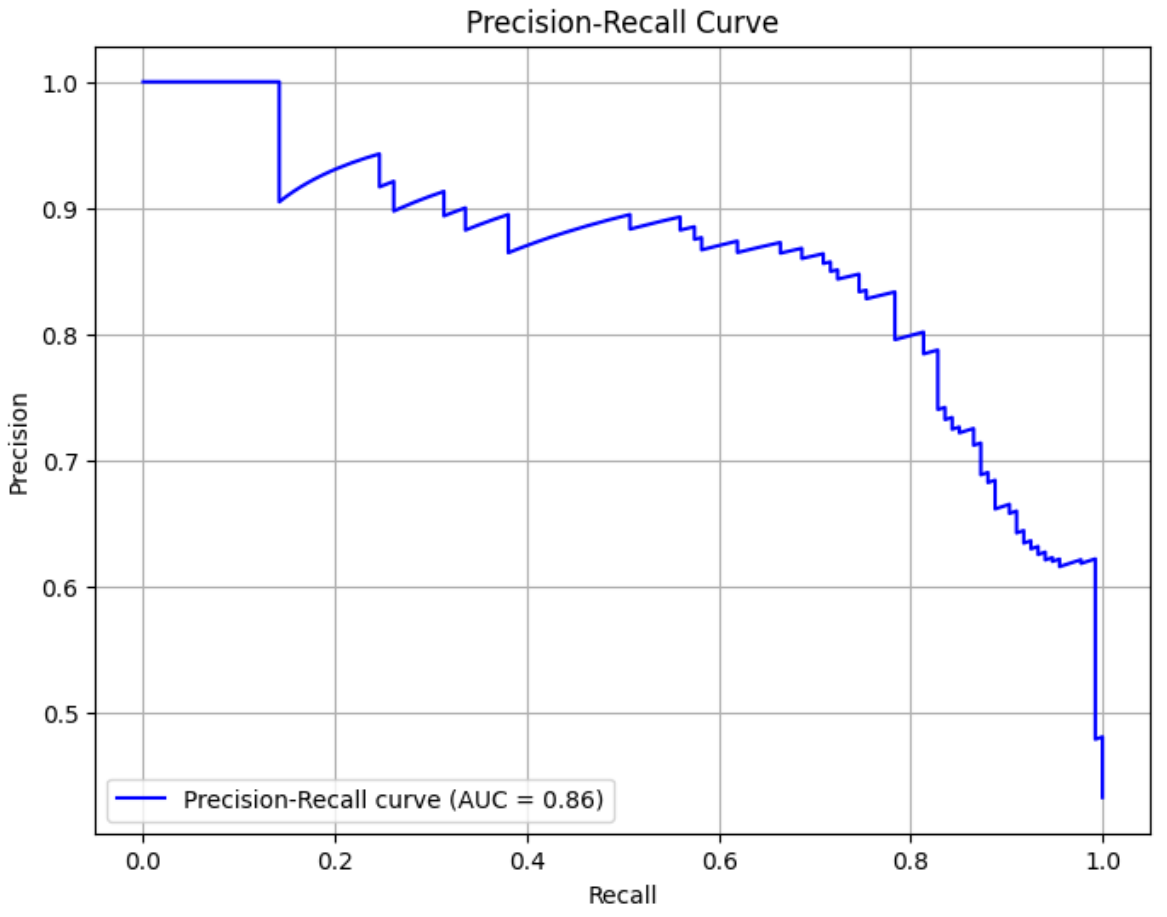
3.1. Бинарная классификация

In [43]:
```
BM = np.zeros(5)
Y_PRED_PROB = np.array([])
Y_TEST = np.array([])
CONFUSION_MATRIX = np.zeros((2, 2))
CV = ShuffleSplit(random_state=0)

for i, (train_index, test_index) in enumerate(CV.split(data)):
    y = data['bin_flag']
    x = data.drop(['bin_flag', 'disease_flag'], axis=1)
    x_train, x_test = prepare_input(x.loc[train_index], x.loc[test_index]
    y_train = y.loc[train_index]
    y_test = y.loc[test_index]

    model = LogisticRegression()
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    y_pred_prob = model.predict_proba(x_test)[:,1]

    binm = bin_metrics(y_test, y_pred, y_pred_prob)
```

```
    CONFUSION_MATRIX += binm[0]
    BM += binm[1:]
    Y_PRED_PROB = np.concatenate((Y_PRED_PROB, y_pred_prob))
    Y_TEST = np.concatenate((Y_TEST, y_test))

BM /= 10
precision, recall, thresholds = precision_recall_curve(Y_TEST, Y_PRED_PRO
fpr, tpr, _ = roc_curve(Y_TEST, Y_PRED_PROB)


print(f' Accuracy_score = {BM[0]}\n Precision = {BM[1]}, Recall = {BM[2]}
```

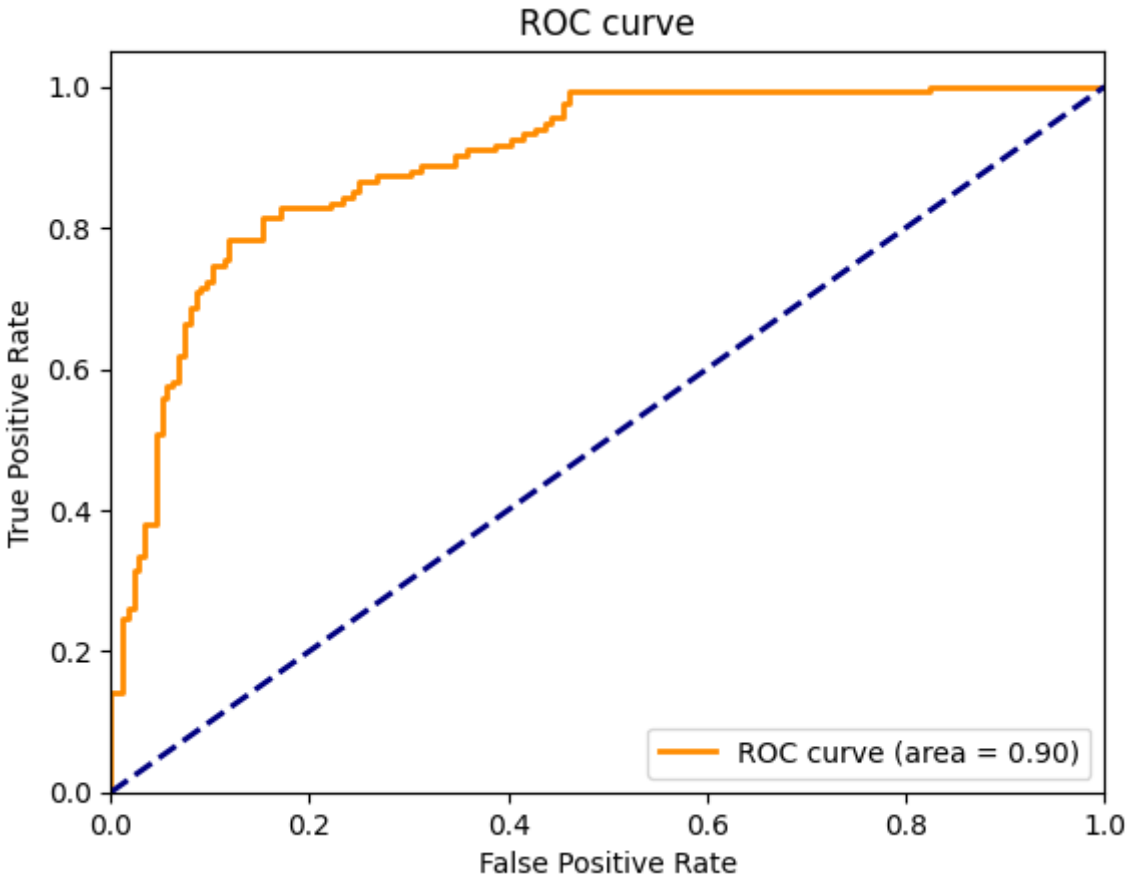
```
Accuracy_score = 0.8225806451612904
Precision = 0.8514208014208015, Recall = 0.7290981567452157
F1_score = 0.7765277046236567,
ROC/AUC_score = 0.9007577553165789
```

In [44]:
```
auc_score = auc(recall, precision)

plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='b', label=f'Precision-Recall curve (AU
#plt.fill_between(recall, precision, step='post', alpha=0.2, color='b')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='lower left')
plt.grid(True)
plt.show()
```

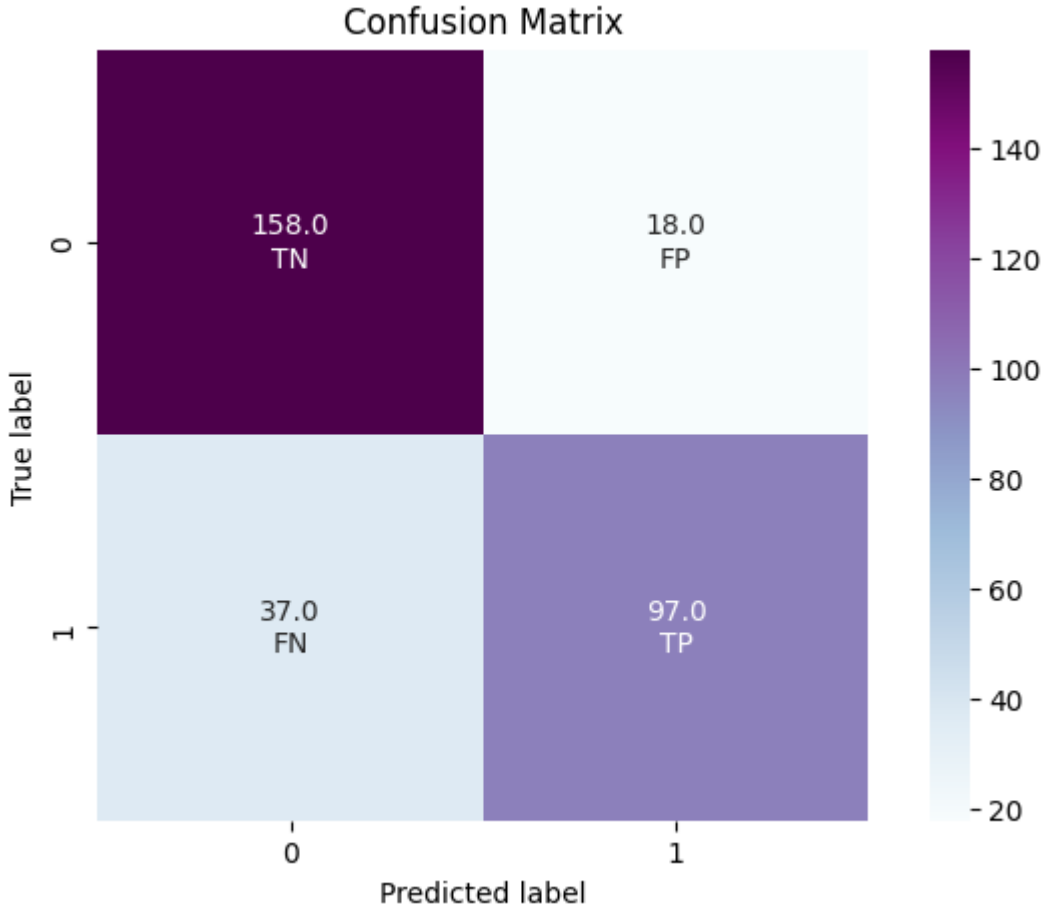


In [45]:
```
roc_auc = auc(fpr, tpr)

plt.figure()
```

```
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve')
plt.legend(loc="lower right")
plt.show()
```



In [46]:
```
labels = np.array([['TN', 'FP'], ['FN', 'TP']])
stats = np.array([[f"{CONFUSION_MATRIX[i, j]}\n{labels[i, j]}" for j in r

plt.figure(figsize=(8, 5))
sns.heatmap(CONFUSION_MATRIX, annot=stats, fmt='', cmap='BuPu', square=Tr
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
```



Вывод

Модель бинарной классификации отработала довольно хорошо, классы сбалансированны

1. Accuracy_score - 84% предсказаний верные(с учётом сбалансированных классов целевой переменной - хороший результат)
2. Precision = 0.8552503052503052 Recall = 0.7654764843000138 Достаточно низкий recall говорит о том, что мы достаточно плохо предсказываем больным людям, что они больны(находим мало объектов положительного класса из всех положительных) В контретной рассматриваемой задаче нам важнее показать больным, что они больны (recall), для этого можно дать большие веса при оптимизации модели именно положительному классу
3. F1_score = 80% - учитывая, что нам recall важнее, лучше считать Fβ-score β > 1
4. ROC/AUC_score = 0.89% - в целом, при варьировании трешхолда(порога), достаточно неплохо
5. Precision recall curve - если нам важно предсказывать, что человек болен, с вероятностью, например, 0.9, то мы получим много ложных предсказаний(precision = 0.6)
6. Confusion matrix - многим людям говорим, что они здоровы, хотя они больны

7. Какие из них лучше всего подходят? Precision recall curve, т.к. можно подобрать параметр трешхолда, чтобы подобрать precision и recall, которые нам действительно нужны

8. Какие не подходят? Почему? Нам не подходят Accuracy_score и F1_score, поскольку они не говорят нам об ошибках первого и второго рода. ROC/AUC_score - аналогично, но зато говорит, насколько в целом модель хорошо работает

9. Как улучшить модель? Возможно ли использовать регуляризацию? Есть небольшая корреляция, используем регуляризацию l2

10. Добавление кодировки категориальных признов незначительно снижает все метрики, помимо ROC/AUC_score и Precision recall curve, однако принято решение, что именно эти метрики ключевые.

Поскольку с кодировкой значение метрики recall достигает 0.9 при precision 0.65, а не 0.6, как без кодировки. Принято решение оставить кодировку

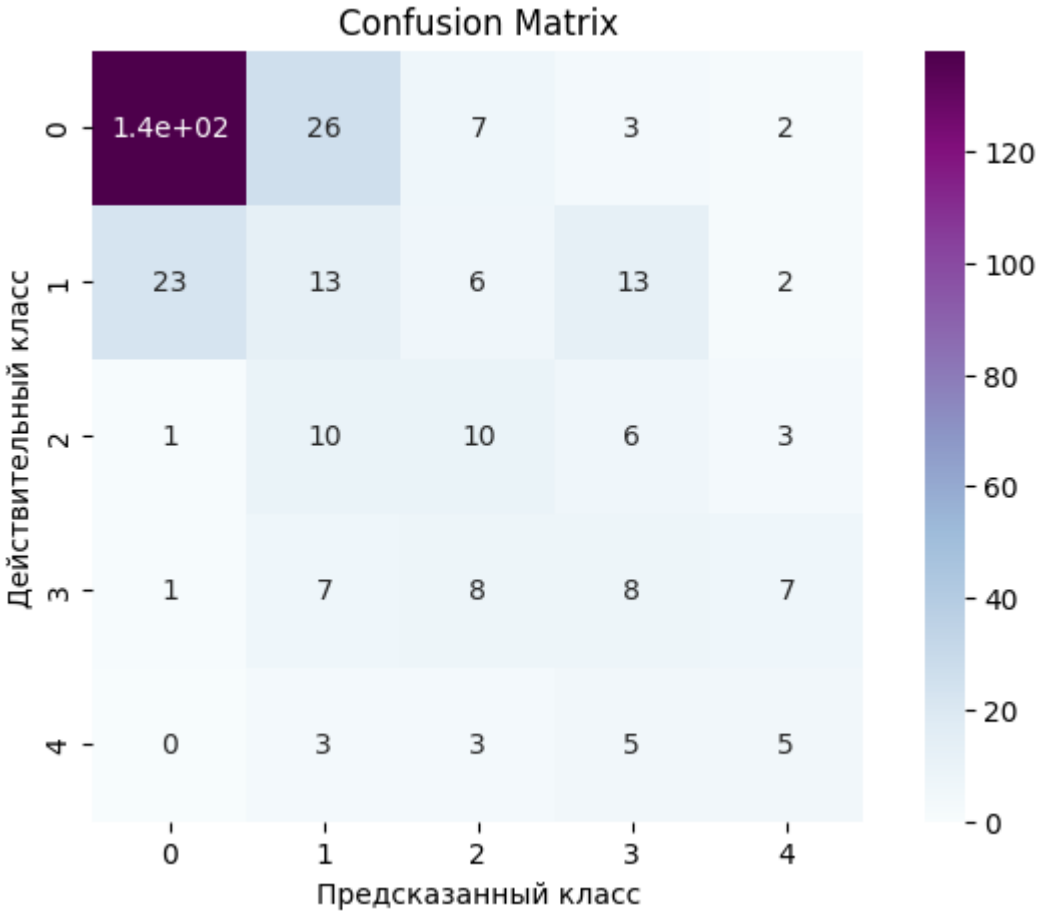
3.2. Многоклассовая классификация

In [47]:
```
BM = np.zeros(4)
Y_PRED_PROB = np.array([])
Y_TEST = np.array([])
CONFUSION_MATRIX = np.zeros((5, 5))
CV = ShuffleSplit(random_state=0)

for i, (train_index, test_index) in enumerate(CV.split(data)):
    y = data['disease_flag']
    x = data.drop(['bin_flag', 'disease_flag'], axis=1)
    x_train, x_test = prepare_input(x.loc[train_index], x.loc[test_index]
    y_train = y.loc[train_index]
    y_test = y.loc[test_index]

    model = LogisticRegression(class_weight='balanced')
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)

    binm = metrics(y_test, y_pred)
    CONFUSION_MATRIX += binm[0]
    BM += binm[1:]

BM /= 10


print(f' Accuracy_score = {BM[0]}\n Precision = {BM[1]}, Recall = {BM[2]}
```

```
Accuracy_score = 0.5612903225806452
Precision = 0.5937019111806887, Recall = 0.5612903225806452
F1_score = 0.5672138709705484
```

In [48]:
```
plt.figure(figsize=(8, 5))
sns.heatmap(CONFUSION_MATRIX, square=True, annot=True, cmap='BuPu')
plt.title('Confusion Matrix')
```

```
plt.ylabel('Действительный класс')
plt.xlabel('Предсказанный класс')
plt.show()
```



Вывод: модель не работает, и данных для пяти классов недостаточно(по 20 примеров для некоторых классов), и модель слабая: классы схожи и модель неспособна их различить

Даже балансировка весов классов не помогает

3.3. Линейная регрессия

In [49]:
```
BM = np.zeros(4)
Y_PRED_PROB = np.array([])
Y_TEST = np.array([])
CONFUSION_MATRIX = np.zeros((5, 5))
CV = ShuffleSplit(random_state=0)

for i, (train_index, test_index) in enumerate(CV.split(data)):
    y = data['disease_flag']
    x = data.drop(['bin_flag', 'disease_flag'], axis=1)
    x_train, x_test = prepare_input(x.loc[train_index], x.loc[test_index]
    y_train = y.loc[train_index]
    y_test = y.loc[test_index]

    model = LinearRegression()
    model.fit(x_train, y_train)
    y_pred = np.trunc(model.predict(x_test))
```

```
    binm = metrics(y_test, y_pred)
    CONFUSION_MATRIX += binm[0]
    BM += binm[1:]

BM /= 10


print(f' Accuracy_score = {BM[0]}\n Precision = {BM[1]}, Recall = {BM[2]}
```

```
Accuracy_score = 0.6387096774193549
Precision = 0.5781691153592681, Recall = 0.6387096774193549
F1_score = 0.5981024675035936
```

Поскольку таргет упорядоченый 0 - 4 (степень заболевания), можно попробовать задач задачу регресии, и потом уже округлять результат к классу для того, чтобы дать модели информацию о том, что мы предсказываем не абсолютно разные классы, а просто степени одного и того же

но не особо помогло :(

9. И вновь вопрос - как улучшить модель?

Судя по графикам п. 1.4 классы существенно пересекаются. Линейная модель с таким справиться не может. Следовательно, никак Однако, конечно, можно поискать преобразование над данными, которое позволит перейти в пространство, где классы будут разделимы, но проще взять нелинейную модель