Для выполнения задачи построения моделей классификации был представлен набор данных heart disease dataset

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from typing import Dict
```

In [ ]:
```
data = pd.read_csv('../datasets/heart.csv')
data = data.dropna()
```

Был создан датафрейм, содержащий 13 нецелевых признаков и 1 целевой - сердечное заболевание

In [ ]:
```
data
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	52.0	1	0	125.0	212.0	0.0	1	168.0	0.0	1.0	2	2
3	61.0	1	0	148.0	203.0	0.0	1	161.0	0.0	0.0	2	1
5	58.0	0	0	100.0	248.0	0.0	0	122.0	0.0	1.0	1	0
6	58.0	1	0	114.0	318.0	0.0	2	140.0	0.0	4.4	0	3
7	55.0	1	0	160.0	289.0	0.0	0	145.0	1.0	0.8	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59.0	1	1	140.0	221.0	0.0	1	164.0	1.0	0.0	2	0
1021	60.0	1	0	125.0	258.0	0.0	0	141.0	1.0	2.8	1	1
1022	47.0	1	0	110.0	275.0	0.0	0	118.0	1.0	1.0	1	1
1023	50.0	0	0	110.0	254.0	0.0	0	159.0	0.0	0.0	2	0
1024	54.0	1	0	120.0	188.0	0.0	1	113.0	0.0	1.4	1	1

970 rows × 14 columns

In [ ]:
```
data_y = data['target']
data_y
```

Out[ ]:
```
0       0.0
3       0.0
5       1.0
6       0.0
7       0.0
       ...
1020    1.0
1021    0.0
1022    0.0
1023    1.0
1024    0.0
Name: target, Length: 970, dtype: float64
```

Типы данных всех полей являются числовыми

In [ ]:
```
data.dtypes
```

Out[ ]:
```
age         float64
sex           int64
cp            int64
trestbps    float64
chol        float64
fbs         float64
restecg       int64
thalach     float64
exang       float64
oldpeak     float64
slope         int64
ca            int64
thal        float64
target      float64
dtype: object
```

В наборе данных есть дубликаты, но так как набор данных о сердечном заболевании и идентичные записи вероятны, удалять дубликаты не будем

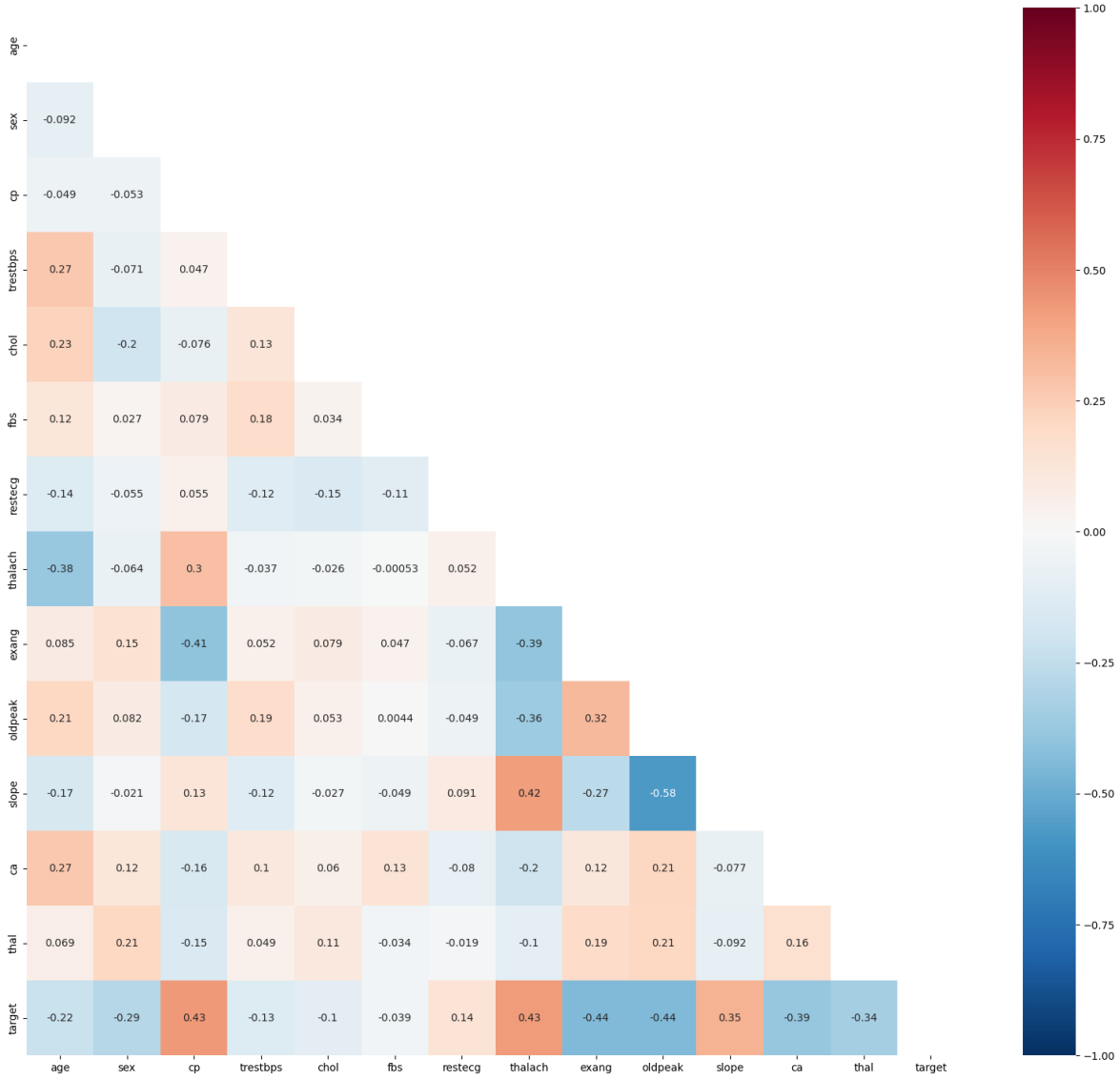
In [ ]:
```
data.duplicated().sum()
```

Out[ ]:
```
667
```

Проведем корреляционный анализ, чтобы оценить вклад признаков для построения моделей классификации. Для визуализации корреляционной матрицы была использована тепловая карта

In [ ]:
```
plt.figure(figsize=(20, 18))
mask=np.triu(np.ones_like(data.corr(), dtype=bool))
sns.heatmap(data.corr(), mask=mask, annot=True, vmin=-1.0, vmax=1, center=0, cma
```

Out[ ]:
```
<AxesSubplot: >
```



С целевым признаком наиболее сильную корреляцию имеют признаки 'exang' (-0.44), 'oldpeak' (-0.44), 'cp' (0.43), 'thalach' (0.43). Эти признаки будут наиболее информативны при построении моделей машинного обучения. Целевой признак также коррелирует с признаками 'ca' (0.39), 'slope' (0.35), 'trial' (-0.34), 'sex' (-0.29). Эти признаки также стоит использовать при обучении модели. Признаки 'fbs' (-0.039), 'chol' (-0.1), 'trestbps' (-0.13), 'restecg' (0.14), 'age' (-0.22) слабо коррелируют с целевым признаком и могут негативно сказаться на модели машинного обучения, поэтому их стоит исключить из модели. Однако не все признаки, которые имеют сильную корреляцию, стоит использовать для построения модели машинного обучения. Между признаками 'oldpeak' и 'slope' наблюдается очень высокая корреляция (-0.58). Поэтому из этих двух признаков стоит оставить тот, который имеет наибольшую корреляцию с целевым признаком - 'oldpeak'. Остальные нецелвые признаки не коррелируют друг с другом так сильно. Таким образом, на основе признаков 'exang', 'oldpeak', 'cp', 'thalach', 'ca', 'trial', 'sex' могут быть построены модели машинного обучения, первые четыре признака могут иметь наиболее весомый вклад в их обучение.

Выборка сердечного заболевания сбалансирована

In [ ]:
```
data_y.value_counts()
```

Out[ ]:
```
1.0    503
0.0    467
Name: target, dtype: int64
```

Разобьем исходную выборку на обучающую и тестовую

In [ ]:
```
data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(data[['a
```

Было произведено MinMax масштабирование данных

In [ ]:
```
mms = MinMaxScaler()
```

In [ ]:
```
data_X_train_scaled = mms.fit_transform(data_X_train)
data_X_test_scaled = mms.fit_transform(data_X_test)
```

Была обучена модель логической регрессии

In [ ]:
```
cl=LogisticRegression(multi_class='multinomial')
```

In [ ]:
```
cl.fit(data_X_train_scaled, data_y_train)
```

Out[ ]:
```
LogisticRegression
LogisticRegression(multi_class='multinomial')
```

Результаты классификации с использованием модели логической регрессии

In [ ]:
```
pred_data_y_test = cl.predict(data_X_test_scaled)
pred_data_y_test
```

Out[ ]:
```
array([1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 1.,
       0., 1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0.,
       0., 0., 1., 1., 0., 0., 1., 0., 0., 0., 1., 1., 1., 1., 1., 0., 1.,
       0., 1., 0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 1.,
       1., 0., 1., 1., 1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
       1., 0., 0., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0.,
       1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0., 1., 1., 1.,
       1., 1., 1., 0., 0., 1., 0., 1., 0., 0., 1., 1., 1., 0., 0., 0., 1.,
       1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1., 1., 0.,
       0., 0., 1., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1.,
       0., 1., 1., 0., 0., 1., 1., 0., 0., 1., 1., 0., 0., 1., 1., 0., 0.,
       0., 1., 0., 0., 0., 1., 0.])
```

Для оценки качества моделей машинного обучения были использованы метрики accuracy и F1-мера. Метриа accuracy подходит для оценки качества моделей классификации для заданного набора данных, тк классификация производится по двум равноценным классам и нет необходимости в более точном определении того или иного класса. Метрика F1-мера подходит для оценки качества моделей классификации для заданного набора данных, тк в случае классификации по двум равноценным классам precision и recall имеют равноценное значение, поэтому их оценку можно совместить в метрику F1-мера.

Значение метрики accuracy

In [ ]:
```
accuracy_score(data_y_test, pred_data_y_test)
```

Out[ ]:
```
0.9072164948453608
```

Функции вывода занчения метрики accuracy для каждого класса

In [ ]:
```
def accuracy_score_for_classes(y_true:np.ndarray, y_pred:np.ndarray) -> Dict[int
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    classes = np.unique(y_true)
    res = dict()
    for c in classes:
        temp_data_flt = df[df['t']==c]
        temp_acc = accuracy_score(
            temp_data_flt['t'].values, temp_data_flt['p'].values
        )
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(y_true: np.ndarray, y_pred: np.ndarray):
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

Значение метрики accuracy для каждого класса

In [ ]:
```
print_accuracy_score_for_classes(data_y_test, pred_data_y_test)
```

```
Метка    Accuracy
0.0      0.9222222222222223
1.0      0.8942307692307693
```

Значение метрики F1-мера

In [ ]:
```
f1_score(data_y_test, pred_data_y_test, average=None)
```

Out[ ]:
```
array([0.90217391, 0.91176471])
```

Была обучена модель случайного леса

In [ ]:
```
data_rl_cf = RandomForestClassifier(random_state=2)
data_rl_cf.fit(data_X_train_scaled, data_y_train)
```

Out[ ]:
```
RandomForestClassifier(random_state=2)
```

Результаты классификации с использованием модели случайного леса

In [ ]:
```
pred_data_rf_y_test = data_rl_cf.predict(data_X_test_scaled)
```

```
pred_data_rf_y_test
```

Out[ ]:
```
array([1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1.,
       0., 1., 1., 1., 0., 1., 1., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0.,
       0., 0., 1., 1., 0., 0., 1., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 1.,
       1., 0., 1., 1., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
       1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1.,
       1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 0., 0., 0., 0., 1., 1., 1., 0., 0., 0., 1.,
       1., 1., 0., 1., 0., 1., 0., 0., 0., 1., 0., 1., 1., 0., 1., 0., 0.,
       0., 0., 1., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1.,
       0., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0.,
       0., 1., 0., 0., 0., 1., 0.])
```

Значение метрики accuracy для модели случайного леса

In [ ]:
```
accuracy_score(data_y_test, pred_data_rf_y_test)
```

Out[ ]:
```
0.9742268041237113
```

Значение метрики accuracy для каждого класса

In [ ]:
```
print_accuracy_score_for_classes(data_y_test, pred_data_rf_y_test)
```

```
Метка    Accuracy
0.0      0.9888888888888889
1.0      0.9615384615384616
```

Значение метрики F1-мера для модели случайного леса для каждого класса

In [ ]:
```
f1_score(data_y_test, pred_data_rf_y_test, average=None)
```

Out[ ]:
```
array([0.9726776 , 0.97560976])
```

Таким образом, каждая из моделей машинного обучения классифицирует вино с высокой точностью. Обе модели практически безошибочно определяют наличие сердчного заболевания. Модель случайного леса производит классификацию лучше модели логистической регрессии.