|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_” Информатика и системы управления”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”Системы обработки информации и управления”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***\_\_\_\_\_\_\_\_\_\_”Бинарная классификация”\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_\_\_ИУ5-63Б\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_М.А. Попов\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Ю.Е. Гапанюк\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Ю.Е. Гапанюк\_\_**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_ИУ5\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.М. Черненький

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине \_\_\_\_\_”Технологии машинного обучения”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_\_\_\_ИУ5-63Б\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Попов Максим Алексеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсовой работы \_\_\_”Решение задача классификации \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_учебная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения работы: 25% к \_3\_ нед., 50% к \_9\_ нед., 75% к 12 нед., 100% к 16 нед.

***Задание*** \_\_Решение задачи машинного обучения. Результатом курсового проекта является

отчет, содержащий описания моделей, тексты программ и результаты экспериментов. \_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление курсовой работы:***

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_7\_ » \_\_февраля\_\_ 2020 г.

**Руководитель курсовой работы**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ **Ю.Е. Гапанюк**

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ М.А. Попов**

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

[Задание 3](#_bookmark0)

[Введение 4](#_bookmark1)

[Основная часть 5](#_bookmark2)

[Постановка задачи 5](#_bookmark3)

[Описание выбранного датасета 5](#_bookmark4)

[Решение поставленной задачи 5](#_bookmark5)

[Заключение 30](#_bookmark6)

[Список литературы 31](#_bookmark7)

# Задание

##### В данной курсовой работе необходимо предпринять следующие шаги:

##### Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.

##### Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.

##### Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.

##### Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения.

##### В зависимости от набора данных, порядок выполнения пунктов 2, 3, 4 может быть изменен.

##### Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.

##### Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.

##### Формирование обучающей и тестовой выборок на основе исходного набора данных.

##### Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.

##### Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.

##### Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.

##### Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется постройение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

##### Приведенная схема исследования является рекомендуемой. Возможно выполнение курсовой работы на нестандартную тему, которая должна быть предварительно согласована с ответственным за прием курсовой работы.

# Введение

##### Курсовая работа – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного обучения.

##### Результатом курсовой работы является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

##### Курсовая работа опирается на знания, умения и владения, полученные студентом в рамках лекций и лабораторных работ по дисциплине.

##### В рамках данной курсовой работы необходимо применить навыки, полученные в течение курса «Технологии машинного обучения», и обосновать полученные результаты.

# Основная часть

## Постановка задачи

##### В данной курсовой работе ставится задача определения наличия заболевания сердца у пациента по различным параметрам с помощью методов машинного обучения: "Stochastic gradient descent", "Support vector machine", "Decision tree", "Gradient boosting", "Random forest". С помощью различных метрик выбор метода, который наиболее эффективно и качественно определяет заболевание.

## Описание выбранного датасета

##### Этот датасет содержит 76 атрибутов, но во всех опубликованных экспериментах упоминается подмножество из 14 из них. В частности, база данных Кливленда является единственной, которая использовалась исследователями ОМ до настоящего времени. Поле «target» относится к наличию болезни сердца у пациента.

##### Age - возраст в годах

##### Sex (1 = male; 0 = female)-пол

##### CP (chest pain type) - тип боли в груди

##### TRESTBPS (resting blood pressure (in mm Hg on admission to the hospital))- артериальное давление в покое (в мм рт. ст. при поступлении в больницу)

##### CHOL (serum cholestoral in mg/dl)-холестерин в мг / дл

##### FPS (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false) -уровень сахара в крови натощак> 120 мг / дл

##### RESTECH (resting electrocardiographic results) - результаты электрокардиографии в покое

##### THALACH (maximum heart rate achieved) - достигнута максимальная частота сердечных сокращений

##### EXANG (exercise induced angina (1 = yes; 0 = no)) - стенокардия, вызванная физической нагрузкой

##### OLDPEAK (ST depression induced by exercise relative to rest) - Депрессия ST, вызванная физическими упражнениями относительно отдыха

##### SLOPE (the slope of the peak exercise ST segment) - наклон пика упражнений сегмента ST

##### CA (number of major vessels (0-3) colored by flourosopy) - количество крупных сосудов (0-3), окрашенных по цвету

##### THAL (3 = normal; 6 = fixed defect; 7 = reversable defect)

##### TARGET (1 or 0) - заболевание

## Решение поставленной задачи

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 SGDClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_score, recall\_score from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import GridSearchCV from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, mean\_squared

\_log\_error, median\_absolute\_error, r2\_score

from sklearn.ensemble import GradientBoostingClassifier from sklearn.metrics import confusion\_matrix

from sklearn.model\_selection import KFold, RepeatedKFold, ShuffleSplit, Stratifie dKFold

from sklearn.model\_selection import learning\_curve, validation\_curve

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

###### Загружаем данные и выводим первые 5 строк

###### In [3]:

data=pd.read\_csv("/Users/user/Desktop/data2.csv") data.head()

###### Out[3]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **thalach** | **exang** | **oldpeak** | **slope** | **ca** | **thal** | **target** |
| **0** | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | 1 | 1 |
| **1** | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | 2 | 1 |
| **2** | 41 | 0 | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 | 2 | 0 | 2 | 1 |
| **3** | 56 | 1 | 1 | 120 | 236 | 0 | 1 | 178 | 0 | 0.8 | 2 | 0 | 2 | 1 |
| **4** | 57 | 0 | 0 | 120 | 354 | 0 | 1 | 163 | 1 | 0.6 | 2 | 0 | 2 | 1 |

###### Информация о данных

###### In [4]:

data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 303 entries, 0 to 302 Data columns (total 14 columns):

|  |  |  |
| --- | --- | --- |
| age | 303 | non-null int64 |
| sex | 303 | non-null int64 |
| cp | 303 | non-null int64 |
| trestbps | 303 | non-null int64 |
| chol | 303 | non-null int64 |
| fbs | 303 | non-null int64 |
| restecg | 303 | non-null int64 |
| thalach | 303 | non-null int64 |
| exang | 303 | non-null int64 |

|  |  |  |
| --- | --- | --- |
| oldpeak | 303 | non-null float64 |
| slope | 303 | non-null int64 |
| ca | 303 | non-null int64 |
| thal | 303 | non-null int64 |
| target | 303 | non-null int64 |

dtypes: float64(1), int64(13) memory usage: 33.2 KB

###### Посмотрим заполненность датасета. Возможно есть пропуски.

###### In [5]:

data.isnull().sum()

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

target 0

dtype: int64

###### Out[5]:

###### In [6]:

data.describe()

###### Out[6]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **thalach** | **exang** | **oldpeak** | **slope** | **ca** | **thal** | **target** |
| **coun**  **t** | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 | 303.00000  0 |
| **mea**  **n** | 54.366337 | 0.683168 | 0.966997 | 131.62376  2 | 246.26402  6 | 0.148515 | 0.528053 | 149.64686  5 | 0.326733 | 1.039604 | 1.399340 | 0.729373 | 2.313531 | 0.544554 |
| **std** | 9.082101 | 0.466011 | 1.032052 | 17.538143 | 51.830751 | 0.356198 | 0.525860 | 22.905161 | 0.469794 | 1.161075 | 0.616226 | 1.022606 | 0.612277 | 0.498835 |
| **min** | 29.000000 | 0.000000 | 0.000000 | 94.000000 | 126.00000  0 | 0.000000 | 0.000000 | 71.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 47.500000 | 0.000000 | 0.000000 | 120.00000  0 | 211.00000  0 | 0.000000 | 0.000000 | 133.50000  0 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 2.000000 | 0.000000 |
| **50%** | 55.000000 | 1.000000 | 1.000000 | 130.00000  0 | 240.00000  0 | 0.000000 | 1.000000 | 153.00000  0 | 0.000000 | 0.800000 | 1.000000 | 0.000000 | 2.000000 | 1.000000 |
| **75%** | 61.000000 | 1.000000 | 2.000000 | 140.00000  0 | 274.50000  0 | 0.000000 | 1.000000 | 166.00000  0 | 1.000000 | 1.600000 | 2.000000 | 1.000000 | 3.000000 | 1.000000 |
| **max** | 77.000000 | 1.000000 | 3.000000 | 200.00000  0 | 564.00000  0 | 1.000000 | 2.000000 | 202.00000  0 | 1.000000 | 6.200000 | 2.000000 | 4.000000 | 3.000000 | 1.000000 |

###### Функции, описанные в приведенном выше наборе данных:

###### Count сообщает нам количество пустых строк в объекте.

###### Среднее говорит нам о среднем значении этой функции.

###### Стандарт сообщает нам значение стандартного отклонения этой функции.

###### Мин говорит нам минимальное значение этой функции.

###### 25%, 50% и 75% - процентиль / квартиль каждой характеристики.

###### Макс сообщает нам максимальное значение этой функции.

##### Рассмотрим фичи¶

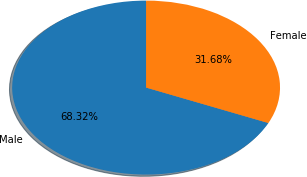
###### Sex

###### In [7]:

male = len(data[data.sex == 1]) female = len(data[data.sex == 0])

plt.pie(x=[male, female], explode=(0, 0), labels=['Male', 'Female'], autopct='%1. 2f%%', shadow=True, startangle=90)

plt.show()



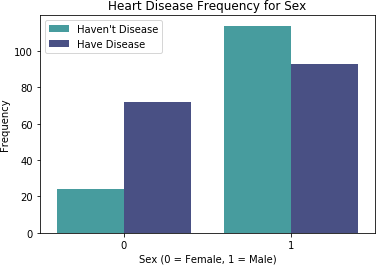
###### In [8]:

sns.countplot('sex',hue='target', data=data, palette='mako\_r') plt.title('Heart Disease Frequency for Sex')

plt.xlabel('Sex (0 = Female, 1 = Male)') plt.xticks(rotation=0)

plt.legend(["Haven't Disease", "Have Disease"]) plt.ylabel('Frequency')

plt.show()



###### CP (chest pain type)

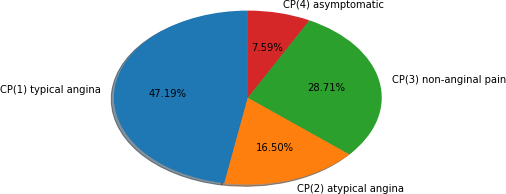
###### In [9]:

x = [len(data[data['cp'] == 0]),len(data[data['cp'] == 1]), len(data[data['cp'] =

= 2]), len(data[data['cp'] == 3])]

plt.pie(x, data=data, labels=['CP(1) typical angina', 'CP(2) atypical angina', 'C P(3) non-anginal pain', 'CP(4) asymptomatic'], autopct='%1.2f%%', shadow=True,sta rtangle=90)

plt.show()

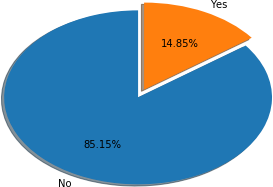


###### In [10]:

sizes = [len(data[data.fbs == 0]), len(data[data.fbs==1])] labels = ['No', 'Yes']

plt.pie(x=sizes, labels=labels, explode=(0.1, 0), autopct="%1.2f%%", startangle=9 0,shadow=True)

plt.show()



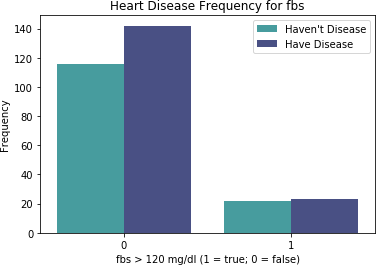
###### FPS (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

###### In [11]:

sns.countplot('fbs', hue='target', data=data, palette='mako\_r') plt.title('Heart Disease Frequency for fbs') plt.xticks(rotation=0)

plt.xlabel('fbs > 120 mg/dl (1 = true; 0 = false)') plt.legend(["Haven't Disease", "Have Disease"]) plt.ylabel('Frequency')

plt.show()



###### restecg (resting electrocardiographic results)

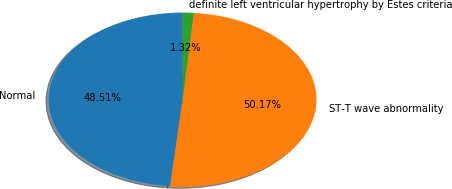
###### In [12]:

sizes = [len(data[data.restecg == 0]), len(data[data.restecg==1]), len(data[data. restecg==2])]

labels = ['Normal', 'ST-T wave abnormality', 'definite left ventricular hypertrop hy by Estes criteria']

plt.pie(x=sizes, labels=labels, explode=(0, 0, 0), autopct="%1.2f%%", startangle= 90,shadow=True)

plt.show()

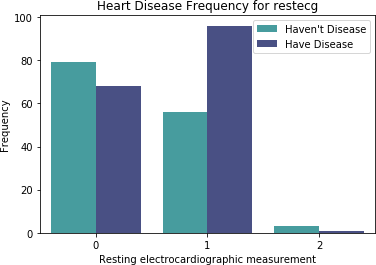


###### In [13]:

sns.countplot('restecg', hue='target', data=data, palette='mako\_r') plt.title('Heart Disease Frequency for restecg') plt.xlabel('Resting electrocardiographic measurement') plt.xticks(rotation=0)

plt.legend(["Haven't Disease", "Have Disease"]) plt.ylabel('Frequency')

plt.show()



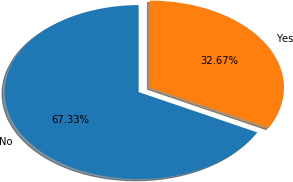
###### exang (exercise induced angina)

###### In [14]:

sizes = [len(data[data.exang == 0]), len(data[data.exang==1])] labels = ['No', 'Yes']

plt.pie(x=sizes, labels=labels, explode=(0.1, 0), autopct="%1.2f%%", startangle=9 0,shadow=True)

plt.show()

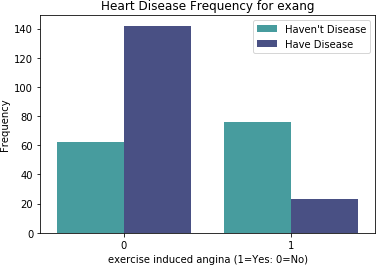


###### In [15]:

sns.countplot('exang', hue='target', data=data, palette='mako\_r') plt.title('Heart Disease Frequency for exang') plt.xlabel('exercise induced angina (1=Yes: 0=No)') plt.xticks(rotation=0)

plt.legend(["Haven't Disease", "Have Disease"]) plt.ylabel('Frequency')

plt.show()



###### Slope : The slope of the peak exercise ST segment

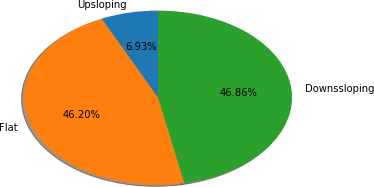
###### In [16]:

sizes = [len(data[data.slope == 0]), len(data[data.slope==1]), len(data[data.slop e==2])]

labels = ['Upsloping', 'Flat', 'Downssloping']

plt.pie(x=sizes, labels=labels, explode=(0, 0, 0), autopct="%1.2f%%", startangle= 90,shadow=True)

plt.show()



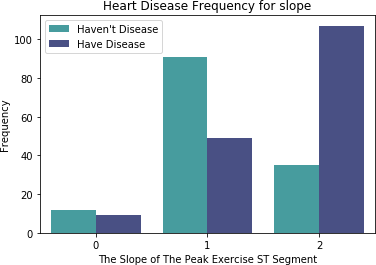
###### In [17]:

sns.countplot('slope', hue='target', data=data, palette='mako\_r') plt.title('Heart Disease Frequency for slope')

plt.xlabel('The Slope of The Peak Exercise ST Segment') plt.xticks(rotation=0)

plt.legend(["Haven't Disease", "Have Disease"]) plt.ylabel('Frequency')

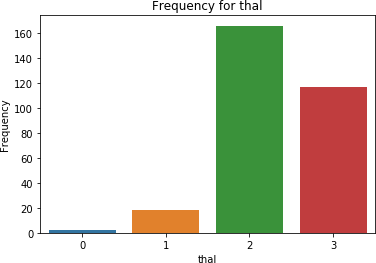
plt.show()



###### thal : A blood disorder called thalassemia

###### In [18]:

sns.countplot('thal', data=data) plt.title('Frequency for thal') plt.ylabel('Frequency') plt.show()

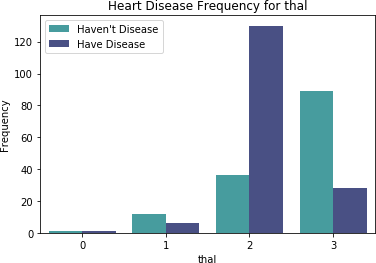


###### In [19]:

sns.countplot('thal', hue='target', data=data, palette='mako\_r') plt.title('Heart Disease Frequency for thal') plt.xticks(rotation=0)

plt.legend(["Haven't Disease", "Have Disease"]) plt.ylabel('Frequency')

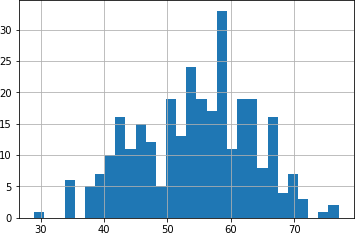
plt.show()



###### Age

###### In [20]:

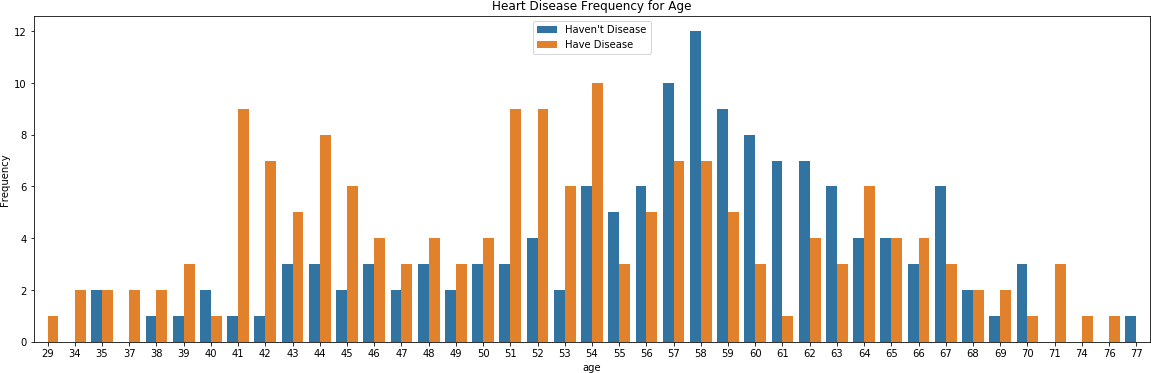
data.age.hist(bins=30) plt.show()



###### In [21]:

plt.figure(figsize=(20, 6)) sns.countplot('age', hue='target', data=data) plt.title('Heart Disease Frequency for Age') plt.legend(["Haven't Disease", "Have Disease"]) plt.ylabel('Frequency')

plt.show()



###### Chol : serum cholestoral in mg/dl

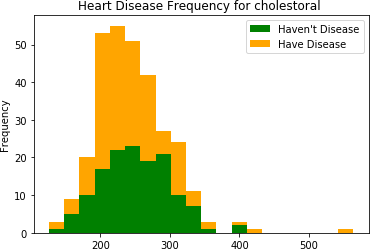
###### In [22]:

plt.hist([data.chol[data.target==0], data.chol[data.target==1]], bins=20,color=[' green', 'orange'], stacked=True)

plt.legend(["Haven't Disease", "Have Disease"]) plt.title('Heart Disease Frequency for cholestoral ') plt.ylabel('Frequency')

plt.plot()

[]



###### thalach : maximum heart rate achieved

###### Out[22]:

###### In [23]:

plt.hist([data.thalach[data.target==0], data.thalach[data.target==1]], bins=20,co lor=['green', 'orange'], stacked=True)

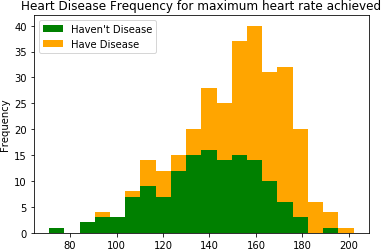
plt.legend(["Haven't Disease", "Have Disease"])

plt.title('Heart Disease Frequency for maximum heart rate achieved') plt.ylabel('Frequency')

plt.plot()

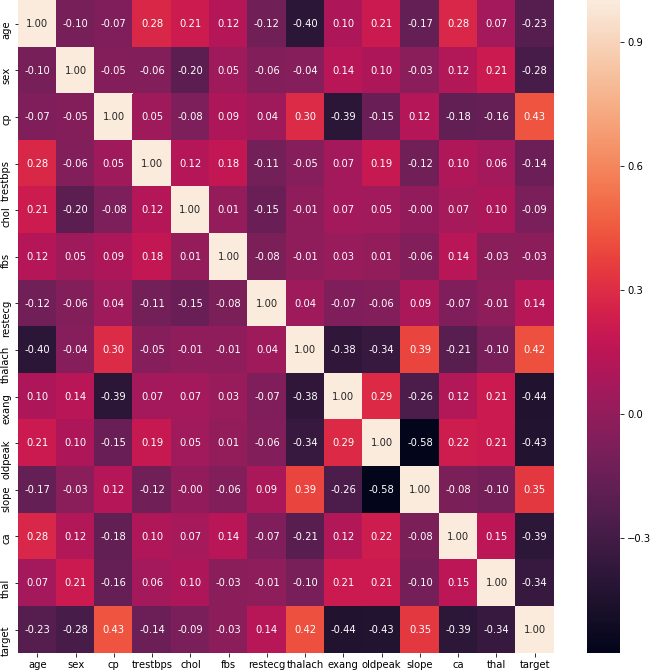
###### Out[23]:

[]



###### In [24]:

plt.figure(figsize=(12, 12)) sns.heatmap(data.corr(), annot=True, fmt='.2f') plt.show()



###### In [25]:

target = data['target']

data = data.drop('target', axis = 1)

###### In [26]:

*#деление на тестовую и обучающую выборку*

X\_train, X\_test, y\_train, y\_test = train\_test\_split( data, target, test\_size=0.2, random\_state=1)

###### In [27]:

#*Стохастический градиентный спуск*

sgd = SGDClassifier().fit(X\_train, y\_train)

###### In [28]:

#SVM

svm\_svc = SVC(gamma='auto').fit(X\_train, y\_train)

###### In [29]:

#Decision tree

decision\_tree = DecisionTreeClassifier(random\_state=1, max\_depth=0.75).fit(X\_trai n, y\_train)

###### In [33]:

target\_sgd = sgd.predict(X\_test) sgd\_inf = []

sgd\_inf = [accuracy\_score(y\_test, target\_sgd), precision\_score(y\_test, target\_sgd

), recall\_score(y\_test, target\_sgd)] accuracy\_score(y\_test, target\_sgd), \ precision\_score(y\_test, target\_sgd), \ recall\_score(y\_test, target\_sgd)

(0.6065573770491803, 0.6129032258064516, 0.6129032258064516)

###### Out[33]:

###### In [46]:

target\_svm\_svc = svm\_svc.predict(X\_test) svm\_inf = []

svm\_inf = [accuracy\_score(y\_test, target\_svm\_svc), precision\_score(y\_test, target

\_svm\_svc), recall\_score(y\_test, target\_svm\_svc)] accuracy\_score(y\_test, target\_svm\_svc), \ precision\_score(y\_test, target\_svm\_svc), \ recall\_score(y\_test, target\_svm\_svc)

(0.5081967213114754, 0.5081967213114754, 1.0)

###### Out[46]:

###### In [35]:

target\_decision\_tree = decision\_tree.predict(X\_test) decision\_tree\_inf = []

decision\_tree\_inf = [accuracy\_score(y\_test, target\_decision\_tree), precision\_scor e(y\_test, target\_decision\_tree), recall\_score(y\_test, target\_decision\_tree)]

accuracy\_score(y\_test, target\_decision\_tree), \ precision\_score(y\_test, target\_decision\_tree), \ recall\_score(y\_test, target\_decision\_tree)

(0.5081967213114754, 0.5081967213114754, 1.0)

###### Out[35]:

###### In [98]:

*#Подбор гиперпараметра с помошью* GridSearchCV *и кроссвалидации* scores\_sgd = cross\_val\_score(SGDClassifier(),

X\_train, y\_train, cv=2)

scores\_sgd

array([0.61157025, 0.45454545])

###### Out[98]:

###### In [99]:

scores\_svm\_svc = cross\_val\_score(SVC(gamma='auto'),

X\_train, y\_train, cv=2)

scores\_svm\_svc

array([0.55371901, 0.55371901])

###### Out[99]:

###### In [100]:

scores\_decision\_tree = cross\_val\_score(DecisionTreeClassifier(),

X\_train, y\_train, cv=2)

scores\_decision\_tree

array([0.80991736, 0.67768595])

###### Out[100]:

###### In [101]:

parameters = {'alpha':[0.5,0.4,0.3,0.2,0.1]}

clf\_gs\_sgd = GridSearchCV(SGDClassifier(), parameters, cv=2, scoring='accuracy') clf\_gs\_sgd.fit(X\_train, y\_train)

GridSearchCV(cv=2, error\_score='raise-deprecating',

###### Out[101]:

estimator=SGDClassifier(alpha=0.0001, average=False, class\_weight=None, early\_stopping=False, epsilon=0.1, eta0=0.0, fit\_intercept=True, l1\_ratio=0.15, learning\_rate='optimal', loss='hinge', max\_iter=None, n\_iter=None, n\_iter\_no\_change=5, n\_jobs=None, penalty='l2', power\_t=0.5, random\_state=None, shuffle=True, tol=None, validation\_fraction=0.1, verbose=0, warm\_start=False),

fit\_params=None, iid='warn', n\_jobs=None, param\_grid={'alpha': [0.5, 0.4, 0.3, 0.2, 0.1]},

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score='warn', scoring='accuracy', verbose=0)

###### In [102]:

parameters = {'gamma':[0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1]}

clf\_gs\_svm\_svc = GridSearchCV(SVC(), parameters, cv=2, scoring='accuracy')

clf\_gs\_svm\_svc.fit(X\_train, y\_train)

GridSearchCV(cv=2, error\_score='raise-deprecating',

###### Out[102]:

estimator=SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0, decision\_function\_shape='ovr', degree=3, gamma='auto\_deprecated', kernel='rbf', max\_iter=-1, probability=False, random\_state=None, shrinking=True, tol=0.001, verbose=False),

fit\_params=None, iid='warn', n\_jobs=None,

param\_grid={'gamma': [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]},

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score='warn', scoring='accuracy', verbose=0)

###### In [103]:

parameters = {'min\_impurity\_decrease':[0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1]}

clf\_gs\_decision\_tree = GridSearchCV(DecisionTreeClassifier(), parameters, cv=2, s coring='accuracy')

clf\_gs\_decision\_tree.fit(X\_train, y\_train)

GridSearchCV(cv=2, error\_score='raise-deprecating',

###### Out[103]:

estimator=DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_ depth=None,

max\_features=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None, splitter='best'),

fit\_params=None, iid='warn', n\_jobs=None, param\_grid={'min\_impurity\_decrease': [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0

.2, 0.1]},

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score='warn', scoring='accuracy', verbose=0)

###### In [104]:

sgd\_new = SGDClassifier(alpha=0.5).fit(X\_train, y\_train) svm\_svc\_new = SVC(gamma=0.1).fit(X\_train, y\_train)

decision\_tree\_new = DecisionTreeClassifier(random\_state=1, min\_impurity\_decrease= 0.4, max\_depth=0.75).fit(X\_train, y\_train)

###### In [164]:

target\_sgd\_new = sgd\_new.predict(X\_test) accuracy\_score(y\_test, target\_sgd\_new), \ precision\_score(y\_test, target\_sgd\_new), \ recall\_score(y\_test, target\_sgd\_new)

(0.6721311475409836, 0.6486486486486487, 0.7741935483870968)

###### Out[164]:

###### In [165]:

cnf\_matrix = confusion\_matrix(y\_test,target\_sgd\_new) cnf\_matrix

array([[17, 13],

[ 7, 24]])

###### Out[165]:

###### In [166]:

class\_names = [0,1] fig,ax = plt.subplots()

tick\_marks = np.arange(len(class\_names)) plt.xticks(tick\_marks,class\_names) plt.yticks(tick\_marks,class\_names)

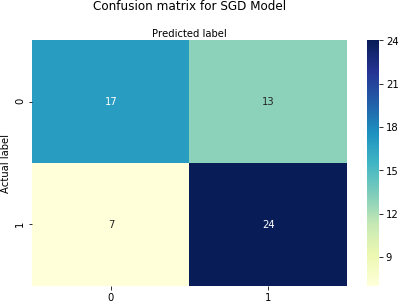
#create a heat map

sns.heatmap(pd.DataFrame(cnf\_matrix), annot = True, cmap = 'YlGnBu', fmt = 'g')

ax.xaxis.set\_label\_position('top') plt.tight\_layout()

plt.title('Confusion matrix for SGD Model', y = 1.1) plt.ylabel('Actual label')

plt.xlabel('Predicted label') plt.show()



###### In [167]:

target\_svm\_svc\_new = svm\_svc\_new.predict(X\_test) accuracy\_score(y\_test, target\_svm\_svc\_new), \ precision\_score(y\_test, target\_svm\_svc\_new), \ recall\_score(y\_test, target\_svm\_svc\_new)

(0.5081967213114754, 0.5081967213114754, 1.0)

###### Out[167]:

###### In [168]:

cnf\_matrix = confusion\_matrix(y\_test,target\_svm\_svc\_new) cnf\_matrix

array([[ 0, 30],

[ 0, 31]])

###### Out[168]:

###### In [169]:

class\_names = [0,1]

fig,ax = plt.subplots()

tick\_marks = np.arange(len(class\_names)) plt.xticks(tick\_marks,class\_names) plt.yticks(tick\_marks,class\_names)

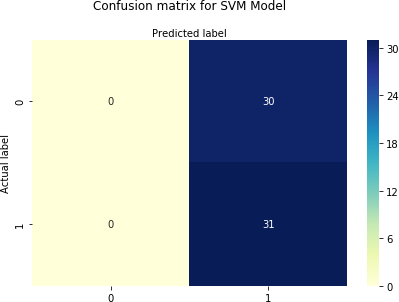
#create a heat map

sns.heatmap(pd.DataFrame(cnf\_matrix), annot = True, cmap = 'YlGnBu', fmt = 'g')

ax.xaxis.set\_label\_position('top') plt.tight\_layout()

plt.title('Confusion matrix for SVM Model', y = 1.1) plt.ylabel('Actual label')

plt.xlabel('Predicted label') plt.show()



###### In [173]:

target\_decision\_tree\_new = decision\_tree\_new.predict(X\_test) accuracy\_score(y\_test, target\_decision\_tree\_new), \ precision\_score(y\_test, target\_decision\_tree\_new), \ recall\_score(y\_test, target\_decision\_tree\_new)

(0.5081967213114754, 0.5081967213114754, 1.0)

###### Out[173]:

###### In [174]:

cnf\_matrix = confusion\_matrix(y\_test,target\_decision\_tree\_new) cnf\_matrix

array([[ 0, 30],

[ 0, 31]])

###### Out[174]:

###### In [175]:

class\_names = [0,1] fig,ax = plt.subplots()

tick\_marks = np.arange(len(class\_names)) plt.xticks(tick\_marks,class\_names) plt.yticks(tick\_marks,class\_names)

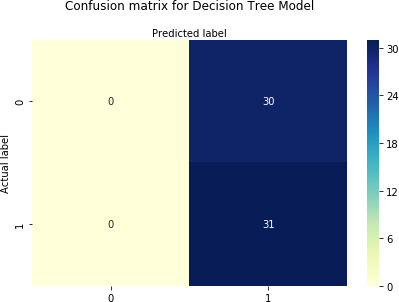
#create a heat map

sns.heatmap(pd.DataFrame(cnf\_matrix), annot = True, cmap = 'YlGnBu', fmt = 'g')

ax.xaxis.set\_label\_position('top') plt.tight\_layout()

plt.title('Confusion matrix for Decision Tree Model', y = 1.1) plt.ylabel('Actual label')

plt.xlabel('Predicted label') plt.show()



### Ассамблевые модели¶

##### Алгоритм случайного леса¶

###### In [176]:

random\_forest = RandomForestClassifier(n\_estimators=10, max\_depth=1, random\_state

=0).fit(X\_train, y\_train)

###### In [178]:

target\_random\_forest = random\_forest.predict(X\_test)

###### In [180]:

accuracy\_score(y\_test, target\_random\_forest), \ precision\_score(y\_test, target\_random\_forest), \ recall\_score(y\_test, target\_random\_forest)

###### Out[180]:

(0.8032786885245902, 0.7567567567567568, 0.9032258064516129)

###### In [182]:

#*Подбираем гиперпараметры*

best\_random\_forest = GridSearchCV(RandomForestClassifier(), parameters\_random\_for est, cv=3, scoring='accuracy')

best\_random\_forest.fit(X\_train, y\_train)

|  |  |  |  |
| --- | --- | --- | --- |
| parameters\_random\_forest = {'n\_estimators':[1, | 3, | 5, | 7, 10], |
| 'max\_depth':[1, 3, | 5, | 7, | 10], |
| 'random\_state':[0, | 2, | 4, | 6, 8, 10]} |

/anaconda3/lib/python3.6/site-packages/sklearn/model\_selection/\_search.py:841: De precationWarning: The default of the `iid` parameter will change from True to Fal se in version 0.22 and will be removed in 0.24. This will change numeric results when test-set sizes are unequal.

DeprecationWarning)

GridSearchCV(cv=3, error\_score='raise-deprecating',

###### Out[182]:

estimator=RandomForestClassifier(bootstrap=True, class\_weight=None, criter ion='gini',

max\_depth=None, max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators='warn', n\_jobs=None, oob\_score=False, random\_state=None, verbose=0, warm\_start=False),

fit\_params=None, iid='warn', n\_jobs=None,

param\_grid={'n\_estimators': [1, 3, 5, 7, 10], 'max\_depth': [1, 3, 5, 7, 10

], 'random\_state': [0, 2, 4, 6, 8, 10]},

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score='warn', scoring='accuracy', verbose=0)

###### In [183]:

GridSearchCV(cv=3, error\_score='raise-deprecating', estimator=RandomForestClassifier(bootstrap=True, class\_weight=None, criter

ion='gini',

max\_depth=None, max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators='warn', n\_jobs=None, oob\_score=False, random\_state=None, verbose=0, warm\_start=False),

fit\_params=None, iid='warn', n\_jobs=None,

param\_grid={'n\_estimators': [1, 3, 5, 7, 10], 'max\_depth': [1, 3, 5, 7, 10

], 'random\_state': [0, 2, 4, 6, 8, 10]},

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score='warn', scoring='accuracy', verbose=0)

GridSearchCV(cv=3, error\_score='raise-deprecating',

###### Out[183]:

estimator=RandomForestClassifier(bootstrap=True, class\_weight=None, criter ion='gini',

max\_depth=None, max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators='warn', n\_jobs=None,

oob\_score=False, random\_state=None, verbose=0, warm\_start=False),

fit\_params=None, iid='warn', n\_jobs=None,

param\_grid={'n\_estimators': [1, 3, 5, 7, 10], 'max\_depth': [1, 3, 5, 7, 10

], 'random\_state': [0, 2, 4, 6, 8, 10]},

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score='warn', scoring='accuracy', verbose=0)

###### In [185]:

best\_random\_forest.best\_params\_

{'max\_depth': 3, 'n\_estimators': 5, 'random\_state': 10}

###### Out[185]:

###### In [187]:

new\_random\_forest = RandomForestClassifier(n\_estimators=5, max\_depth=3, random\_st ate=10).fit(X\_train, y\_train)

###### In [189]:

new\_target\_random\_forest = new\_random\_forest.predict(X\_test)

###### In [191]:

accuracy\_score(y\_test, new\_target\_random\_forest), \ precision\_score(y\_test, new\_target\_random\_forest), \ recall\_score(y\_test, new\_target\_random\_forest)

(0.7377049180327869, 0.7027027027027027, 0.8387096774193549)

###### Out[191]:

###### In [193]:

cnf\_matrix = confusion\_matrix(y\_test,new\_target\_random\_forest) cnf\_matrix

array([[19, 11],

[ 5, 26]])

###### Out[193]:

###### In [194]:

class\_names = [0,1] fig,ax = plt.subplots()

tick\_marks = np.arange(len(class\_names)) plt.xticks(tick\_marks,class\_names) plt.yticks(tick\_marks,class\_names)

#create a heat map

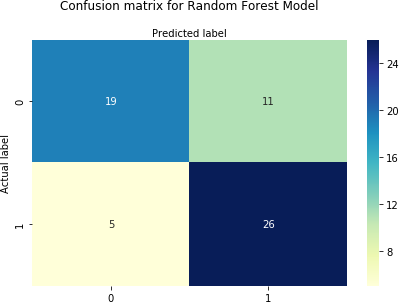
sns.heatmap(pd.DataFrame(cnf\_matrix), annot = True, cmap = 'YlGnBu', fmt = 'g')

ax.xaxis.set\_label\_position('top') plt.tight\_layout()

plt.title('Confusion matrix for Random Forest Model', y = 1.1) plt.ylabel('Actual label')

plt.xlabel('Predicted label')

plt.show()



##### Градиентный бустинг¶

###### In [177]:

gradient\_boosting = GradientBoostingClassifier(n\_estimators=10, max\_depth=10, lea rning\_rate=0.01).fit(X\_train, y\_train)

###### In [179]:

target\_gradient\_boosting = gradient\_boosting.predict(X\_test)

###### In [181]:

accuracy\_score(y\_test, target\_gradient\_boosting), \ precision\_score(y\_test, target\_gradient\_boosting), \ recall\_score(y\_test, target\_gradient\_boosting)

(0.5081967213114754, 0.5081967213114754, 1.0)

###### Out[181]:

###### In [184]:

parameters\_gradient\_boosting = {'n\_estimators':[1, 3, 5, 7, 10],

'max\_depth':[1, 3, 5, 7, 10],

'learning\_rate':[0.001, 0.0025, 0.005, 0.0075, 0.01,

0.025]}

best\_gradient\_boosting = GridSearchCV(GradientBoostingClassifier(), parameters\_gr adient\_boosting, cv=3, scoring='accuracy')

best\_gradient\_boosting.fit(X\_train, y\_train)

/anaconda3/lib/python3.6/site-packages/sklearn/model\_selection/\_search.py:841: De precationWarning: The default of the `iid` parameter will change from True to Fal se in version 0.22 and will be removed in 0.24. This will change numeric results when test-set sizes are unequal.

DeprecationWarning)

GridSearchCV(cv=3, error\_score='raise-deprecating',

###### Out[184]:

estimator=GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance', max\_depth=3, max\_features=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_sampl... subsample=1.0, tol=0.0001, va lidation\_fraction=0.1,

verbose=0, warm\_start=False), fit\_params=None, iid='warn', n\_jobs=None,

param\_grid={'n\_estimators': [1, 3, 5, 7, 10], 'max\_depth': [1, 3, 5, 7, 10

], 'learning\_rate': [0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025]},

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score='warn', scoring='accuracy', verbose=0)

###### In [186]:

best\_gradient\_boosting.best\_params\_

{'learning\_rate': 0.025, 'max\_depth': 5, 'n\_estimators': 7}

###### Out[186]:

###### In [188]:

new\_gradient\_boosting = GradientBoostingClassifier(n\_estimators=10, max\_depth=3, learning\_rate=0.025).fit(X\_train, y\_train)

###### In [190]:

new\_target\_gradient\_boosting = new\_gradient\_boosting.predict(X\_test)

###### In [192]:

accuracy\_score(y\_test, new\_target\_gradient\_boosting), \ precision\_score(y\_test, new\_target\_gradient\_boosting), \ recall\_score(y\_test, new\_target\_gradient\_boosting)

(0.7377049180327869, 0.6744186046511628, 0.9354838709677419)

###### Out[192]:

###### In [195]:

cnf\_matrix = confusion\_matrix(y\_test,new\_target\_gradient\_boosting) cnf\_matrix

array([[16, 14],

[ 2, 29]])

###### Out[195]:

###### In [197]:

class\_names = [0,1] fig,ax = plt.subplots()

tick\_marks = np.arange(len(class\_names)) plt.xticks(tick\_marks,class\_names) plt.yticks(tick\_marks,class\_names)

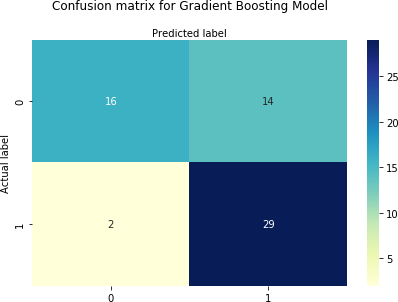
#create a heat map

sns.heatmap(pd.DataFrame(cnf\_matrix), annot = True, cmap = 'YlGnBu', fmt = 'g')

ax.xaxis.set\_label\_position('top') plt.tight\_layout()

plt.title('Confusion matrix for Gradient Boosting Model', y = 1.1)

plt.ylabel('Actual label') plt.xlabel('Predicted label') plt.show()



###### In [202]:

def plot\_learning\_curve(estimator, title, X, y, ylim=None, cv=None,

n\_jobs=None, train\_sizes=np.linspace(.1, 1.0, 5)):

plt.figure() plt.title(title)

if ylim is not None: plt.ylim(\*ylim)

plt.xlabel("Training examples") plt.ylabel("Score")

train\_sizes, train\_scores, test\_scores = learning\_curve(

estimator, X, y, cv=cv, n\_jobs=n\_jobs, train\_sizes=train\_sizes) train\_scores\_mean = np.mean(train\_scores, axis=1)

train\_scores\_std = np.std(train\_scores, axis=1) test\_scores\_mean = np.mean(test\_scores, axis=1) test\_scores\_std = np.std(test\_scores, axis=1) plt.grid()

plt.fill\_between(train\_sizes, train\_scores\_mean - train\_scores\_std, train\_scores\_mean + train\_scores\_std, alpha=0.1, color="r")

plt.fill\_between(train\_sizes, test\_scores\_mean - test\_scores\_std, test\_scores\_mean + test\_scores\_std, alpha=0.1, color="g")

plt.plot(train\_sizes, train\_scores\_mean, 'o-', color="r", label="Training score")

plt.plot(train\_sizes, test\_scores\_mean, 'o-', color="g", label="Cross-validation score")

plt.legend(loc="best") return plt

###### In [206]:

plot\_learning\_curve(GradientBoostingClassifier(n\_estimators=10, max\_depth=10, lea rning\_rate=0.01), '',

X\_train, y\_train, cv=StratifiedKFold(n\_splits=5))

###### Out[206]:

<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-packages/matplotl ib/pyplot.py'>



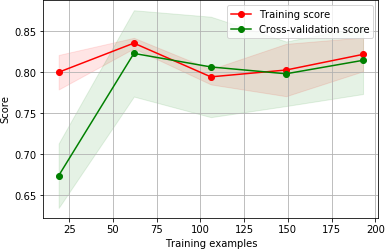
###### In [207]:

plot\_learning\_curve(RandomForestClassifier(n\_estimators=10, max\_depth=1, random\_s tate=0), '',

X\_train, y\_train, cv=StratifiedKFold(n\_splits=5))

###### Out[207]:

<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-packages/matplotl ib/pyplot.py'>



###### In [208]:

plot\_learning\_curve(SGDClassifier(), '', X\_train, y\_train, cv=StratifiedKFold(n\_s plits=5))

###### Out[208]:

<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-packages/matplotl ib/pyplot.py'>



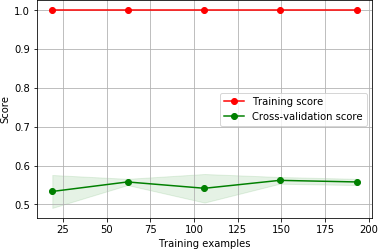
###### In [209]:

plot\_learning\_curve(SVC(gamma='auto'), '', X\_train, y\_train, cv=StratifiedKFold(n

\_splits=5))

###### Out[209]:

<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-packages/matplotl ib/pyplot.py'>



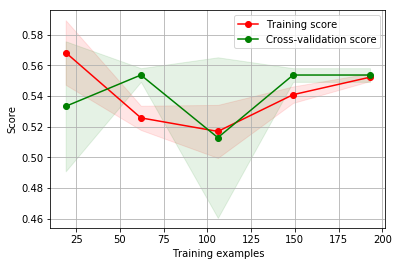
###### In [210]:

plot\_learning\_curve(DecisionTreeClassifier(random\_state=1, max\_depth=0.75), '', X

\_train, y\_train, cv=StratifiedKFold(n\_splits=5))

###### Out[210]:

<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-packages/matplotl ib/pyplot.py'>



# Заключение

##### Из всех рассмотренных алгоритмов: "Stochastic gradient descent", "Support vector machine", "Decision tree", "Gradient boosting", "Random forest". Для модели классификации сердечных заболеваний наиболее эффективным является алгоритм случайного леса, т.е. "Random forest". Как известно Random forest борется с переобучением модели, следовательно можно сделать вывод о том, что датасет является довольно простым и разрозненным, другие методы переобучались, а "Random forest" успешно обошёл эту проблему.

# Список литературы

##### Лекции Гапанюка Ю.Е. [Электронный ресурс]. – Электрон. дан. - URL: https://github.com/ugapanyuk/ml\_course/wiki/COURSE\_TMO (дата обращения: 25.05.2019)

##### Heart Disease UCI [Электронный ресурс]. – Электрон. дан. - URL: <https://www.kaggle.com/ronitf/heart-disease-uci>(дата обращения: 25.05.2019)

##### Руководство для начинающих [Электронный ресурс]. – Электрон. дан. - URL: https://mlbootcamp.ru/article/tutorial/ (дата обращения: 25.05.2019)

##### Scikit learn[Электронный ресурс]. – Электрон. дан. - URL: [https://scikit-](https://scikit-learn.org/stable/index.html) [learn.org/stable/index.html](https://scikit-learn.org/stable/index.html) (дата обращения: 25.05.2019)