Heart Disease Prediction System

SEM 3 Mini Project for Terna Engineering College

Contributor:
Denish Sharma
Sahil Chipkar
Atish Malusare

Table of contents

1. Introduction

The diagnosis of heart disease in most cases depends on a complex combination of clinical and pathological data. Because of this complexity, there exists a significant amount of interest among clinical professionals and researchers regarding the efficient and accurate prediction of heart disease. In this paper, we develop a heart disease prediction system that can assist medical professionals in predicting heart disease status based on clinical data of patients. Our approaches include three steps. Firstly, we select important features based on feature selection technique and according to clinical studies. Secondly, we develop artificial neural network algorithms for classifying heart disease based on these extracted clinical features. The accuracy of prediction is near 87%. Finally, we develop a user-friendly heart disease prediction system (HDPS). The HDPS system will consist of multiple features, including input clinical data section and prediction performance display section with execution time, accuracy and predicted result. Our approaches are effective in predicting the heart disease of patients. The HDPS system developed in this study is a novel approach that can be used in the classification of heart disease.

2. Overview

Heart disease describes a range of conditions that affect your heart. Diseases under the heart disease umbrella include blood vessel diseases, such as coronary artery disease, heart rhythm problems (arrhythmias) and heart defects you're born with (congenital heart defects), among others.

The term “**heart disease**” is often used interchangeably with the term “cardiovascular disease”. Cardiovascular disease generally refers to conditions that involve narrowed or blocked blood vessels that can lead to a heart attack, chest pain (angina) or stroke. Other heart conditions, such as those that affect your heart’s muscle, waves or rhythm, also are considered forms of heart disease.

Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of clinical data analysis. The amount of data in the healthcare industry is huge. Data mining turns the large collection of raw healthcare data into information that can help to make informed decisions and predictions.

Many forms of heart disease can be prevented or treated with healthy lifestyle choices.

3. Symptoms

Heart disease symptoms depend on what type of heart disease you have. Symptoms of heart disease in your blood vessels (atherosclerotic disease). Cardiovascular disease symptoms may be different for men and women. For instance, men are more likely to have chest pain; women are more likely to have other symptoms along with chest discomfort, such as shortness of breath, nausea and extreme fatigue.

Symptoms of heart disease can include:

- Fainting
- Slow or fast heartbeat
- Chest tightness
- Chest pain
- Chest pressure
- Chest discomfort (angina)
- Shortness of breath
- Sudden swelling in your legs, feet, ankles, or abdomen
- Pain in the neck, jaw, throat, or back

You might not be diagnosed with cardiovascular disease until you have a heart attack, angina, stroke or heart failure. It's important to watch for cardiovascular symptoms and discuss concern with your doctor. Cardiovascular disease can sometimes be found early with regular evaluations.

3.1 Abnormal heartbeats (Arrhythmias)

There are cases where heart disease symptoms caused by abnormal heartbeats (**heart arrhythmias**)

The heart arrhythmia is an abnormal heartbeat. Your heart may beat too quickly, too slow, or irregularly. Heart arrhythmia symptoms can include:

- Fluttering in your chest
- Racing heartbeat (tachycardia)
- Slow heartbeat (bradycardia)
- Chest pain or discomfort
- Shortness of breath
- Lightheadedness

- Dizziness
- Fainting (syncope) or near fainting

3.2 Heart defects

Serious congenital heart defects — defects you're born with — usually become evident soon after birth. Heart defect symptoms in children could include:

- Pale gray or blue skin color (cyanosis)
- Swelling in the legs, abdomen or areas around the eyes
- In an infant, shortness of breath during feedings, leading to poor weight gain.

Less serious congenital heart disease defects are often not diagnosed until later in childhood or during adulthood. Signs and symptoms of congenital heart defects that usually aren’t immediately life-threatening include:

- Easily getting short of breath during exercise or activity
- Easily tiring during exercise or activity
- Swelling in the hands, ankles, or feet

3.3 Weak heart muscle (Dilated Cardiomyopathy)

In early stages of cardiomyopathy, you may have no symptoms. As the condition worsens, symptoms may include:

- Breathlessness with exertion or at rest
- Swelling of the legs, ankles, and feet
- Fatigue
- Irregular heartbeats that feel rapid, pounding or fluttering
- Dizziness, lightheadedness and fainting

4. Diagnosing Heart Disease

Various tests are used to diagnose heart disease. Diagnoses are done by taking a patient's profile which includes personal and family medical history, recording current and past symptoms, and doing laboratory tests and an electrocardiogram. Based on these results of the assessment and tests, further tests are conducted.

Some of these tests are non-invasive, which means no instruments are inserted into the body. Other tests are invasive and require inserting instruments into the body.

4.1 Laboratory Tests

Laboratory tests include blood tests to determine your risk of heart disease as well as evaluate other systems of the body that can affect cardiovascular health.

Blood is drawn from the vein of the arm. Some tests require fasting for 12 hours, but mod=st do not have any dietary restriction.

4.1.1 Blood tests for Heart Disease

1. Lipid profile

The lipid profile includes:

- Total cholesterol
- LDL (low-density lipoprotein), termed as “bad” cholesterol
- HDL (high-density lipoprotein), termed as “good” cholesterol
- Triglycerides

2. Lipoprotein (a) or Lp (a)

Lipoprotein (a) is a special type of lipid-containing protein. Your genes, not diet or exercise, play the main role in determining your level of Lp (a).

3. C-reactive protein (CRP)

Your liver produces C-reactive protein (CRP) as part of your body's response to injury or infection. Inflammation plays a central role in the process of atherosclerosis, in which fatty deposits clog your arteries. CRP test results combined with other blood test results and risk factors for heart disease help create an overall picture of your heart health.

4. Homocysteine

Your body uses homocysteine to make protein and to build and maintain tissue. However, too much homocysteine may increase your risk of heart disease and stroke. Homocysteine is usually ordered for people who have a high risk for developing heart disease or have a known history of heart disease. It is also used for people with a family history of heart disease but no other known risk factors.

4.1.2 Blood tests for Other Body Systems

1. Complete blood count (CBC)

CBC is a series of tests that measures your red blood cells, white blood cells and platelets.

2. Sodium and Potassium levels

Sodium and potassium levels are measured to detect a problem with electrolytes in the body fluids.

3. Blood urea nitrogen and creatinine

Blood urea nitrogen and creatinine are measured to check kidney function.

4. Fasting glucose

Fasting glucose is performed to diagnose diabetes or pre-diabetes.

5. ALT and AST

The alanine transaminase (ALT) blood test measures the level of the enzyme ALT in the blood. The aspartate aminotransferase (AST) blood test measures the level of the enzyme AST in the blood. ALT and AST is performed to detect liver inflammation or damage.

6. TSH

A TSH test measures the amount of thyroid stimulating hormone (TSH) in your blood. TSH is produced by the pituitary gland. It prompts the thyroid gland to make and release thyroid hormones into the blood.

4.2 Non-Invasive Tests

4.2.1 Electrocardiogram

An electrocardiogram (EKG or ECG) is a graphic measure of the electrical activity in your heart. There are specific patterns on the EKG that your doctor looks for to determine whether there are abnormalities such as atrial fibrillation (an abnormal rhythm), or new or old heart attack.

During the test, you will lie on an exam table while an electrocardiograph machine records the information. You will be attached to the electrocardiograph by stickers on your chest that are connected to wires leading to the machine. The test takes less than five minutes.

4.2.2 Echocardiogram

An echocardiogram ("echo") is an ultrasound of the heart. A small probe like a microphone, called a transducer, is placed on the chest in various places. The ultrasound waves sent by the transducer bounce off the various parts of the heart. A computer in the machine determines the time it takes for the sound wave to return to the transducer and generates a picture with the data.

During the test, you will lie on your back or left side on a stretcher for about 45 minutes while the pictures are being recorded. The echocardiographer will review the pictures before sending you home to be sure all the necessary information has been obtained.

4.2.3 Stress ECG or Echocardiogram

Stress tests are performed to see how the heart performs under physical stress. The heart can be stressed with exercise on a treadmill or in a few instances, a bicycle. If a person cannot exercise on a treadmill or bicycle, medications can be used to cause the heart rate to increase, simulating normal reactions of the heart to exercise.

During the stress test, you will wear ECG leads and wires while exercising so that the electrical signals of your heart can be recorded at the same time. Your blood pressure is monitored throughout the test. The stress test can be performed together with the echocardiogram, described above.

4.2.4 Nuclear Stress Test

Nuclear stress tests have two components to them: a treadmill (or chemical) stress test and scanning of the heart after injection of a radionuclide material. This material has been used safely for many years to determine the amount of blood the heart muscle is receiving during rest and stress. The scanning is done with a nuclear camera.

4.2.5 Carotid Ultrasound

Carotid ultrasound is done to evaluate your risk of stroke. The sonographer presses the transducer gently against the sides of your neck, which sends images of your arteries to a computer screen for the technician to see. The technician monitors your blood flow through the carotid arteries on both sides of your neck to check for stenosis.

During the exam, you lie on your back on an examination table and a small amount of warm gel is applied to your skin. The test usually takes about 15 to 30 minutes.

4.2.6 Abdominal Ultrasound

Your doctor may also want you to have an abdominal ultrasound to screen for potential abdominal aortic aneurysm. The sonographer presses the transducer against the skin over your abdomen, moving from one area to another. The transducer sends images to a computer screen that the technician monitors. The technician monitors blood flow through your abdominal aorta to check for an aneurysm.

During the exam, you lie on your back on an examination table and a small amount of warm gel is applied to your abdomen. The test usually takes between 20 minutes to an hour.

4.2.7 Holter Monitor

A Holter monitor is a small, portable machine that you wear for 24 to 48 hours. It enables continuous recording of your EKG as you go about your daily activities. You will be asked to keep a diary log of your activities and symptoms. This monitor can detect arrhythmias that might not show up on a resting EKG that only records for a few seconds.

4.2.8 Event Recorder

An event recorder (loop recorder) is a small, portable transtelephonic monitor that may be worn for several weeks. This type of recorder is good for patients whose symptoms are infrequent.

The monitor 'loops' a two- to five-minute recording into its memory which is continually overwritten. When you experience symptoms you press a 'record' button on the monitor, which stores a correlating strip of EKG. The recordings are telephoned through to a 24-hour monitoring station and faxed directly to your doctor.

4.3 Invasive Tests

4.3.1 Cardiac Catheterization and Coronary Angiography

Cardiac catheterization is a common procedure that can help diagnose heart disease. In some cases, catheterization is also used to treat heart disease by opening blocked arteries with balloon angioplasty and stent placement.

Cardiac catheterization can show:

- If the blood vessels in your heart have narrowed.
- If your heart is pumping normally and blood is flowing correctly.
- If the valves in your heart are functioning normally.
- If you were born with any heart abnormalities.

- If the pressures in the heart and lung are normal. If not, catheterization can help assess the problem.

During catheterization, the cardiologist inserts a long, flexible tube called a catheter into a blood vessel, either through the wrist artery or the groin artery, and gently guides it towards your heart under X-ray guidance. Once the catheter is in place, X-rays and other tests are done to help your doctor evaluate whether your coronary arteries are blocked and how well your heart is working.

At times, it might also be necessary to insert a small catheter into a vein to allow measurement of specific pressures in the heart and lung. This procedure can be done either through a neck vein, arm vein or groin vein.

4.3.2 Electrophysiology Study

An electrophysiology study (EP) is a recording of the electrical activity of the heart. This test helps your doctor determine the cause of your rhythm disturbance (arrhythmia) and the best treatment. During the test, the doctor may safely reproduce the arrhythmia, and then give certain medications to see which one controls it best.

An EP study is performed in the Electrophysiology Laboratory, where you will lie on an X-ray table. As with a cardiac catheterization, the doctor inserts a long, flexible tube — an electrode catheter — into a blood vessel, usually in the groin.

There are stages in an EP study:

- Recording the heart's electrical signals to assess electrical function.
- Pacing the heart to bring on certain abnormal rhythms for observation under controlled conditions.
- In some cases, an ablation procedure is performed at the same time to destroy abnormal tissue which may be causing the arrhythmia.

5. Dataset

About 610,000 people die of heart attack in the United States every year, that's 1 in every 4 deaths. Coronary Heart Disease (CHD) is the most common type of heart disease, killing over 370,000 people annually. Every year about 735,000 Americans have a heart attack. Of these, 525,000 are a first heart attack and 210,000 happen in people who have already had a heart attack.

This makes heart disease a major concern to be dealt with. But it is difficult to identify heart disease because of several contributory risk factors such as diabetes, high blood pressure, high cholesterol, abnormal pulse rate, and many other factors. Due to such constraints, scientists have turners toward modern approaches like Data Mining and Machine Learning for predicting the disease.

Machine Learning (ML) proves to be effective in assisting in making decisions and predictions from the large quantity of the data produced by the healthcare industry.

The dataset used in this project is the Cleveland Heart Disease dataset taken from the UCI repository.

The dataset contains 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 has been used by ML researchers to this date. The "target" field refers to the presence of heart disease in the patient. It is integer value from 0 (no presence) to 4 (presence).

The names and personal profiles (social security number) of the patients were removed from the database in order to maintain privacy of the profiles.

5.1 Attribute Information

According to clinical trials and studies only 14 factors plays a major role in detecting heart disease, which are:

- **Age**: Displays the age of an individual
- **Sex**: Displays the gender of an individual using the following format:
  1 = Male; 0 = Female
- **Chest Pain Type**: Displays the type of chest pain experienced by the individual using the following format:
  1 = Typical angina; 2 = Atypical angina; 3 = Non-anginal pain; 4 = Asymptotic
- **Resting Blood Pressure**: Displays the resting blood pressure value of an individual in mmHg (unit)

- **Serum Cholesterol**: Displays the serum cholesterol in mg/dl (unit)
- **Fasting Blood Sugar**: Compares the fasting blood sugar value of an individual with 120mg/dl.
  If fasting blood sugar > 120mg/dl then: 1 (true) else: 0 (false)
- **Resting ECG**: Displays resting electrocardiographic results:
  0 = Normal; 1 = Having ST-T wave abnormality; 2 = Left ventricular hypertrophy
- **Maximum Heart Rate Achieved**: Displays the max heart rate achieved by an individual
- **Exercise Induced Angina**: 1 = Yes; 0 = No
- **ST depression induced by exercise relative to rest**: Displays the value which is an integer or float.
- **Peak exercise ST segment**: 1 = Upsloping; 2 = Flat; 3 = Downsloping
- **Number of major vessels colored by fluoroscopy**: Displays the value as integer or float.
- **Thalassemia**: Displays the thalassemia:
  3 = Normal; 6 = Fixed defect; 7 = Reversible defect
- **Diagnosis of heart disease**: Displays whether the individual is suffering from heart disease or not:
  0 = Absence; 1, 2, 3, 4 = Present

Why these parameters?

In the actual dataset, there are 76 features but only 14 are needed for clinical studies and to determine the heart disease.

1. Age

Age is the most important risk factor in developing cardiovascular or heart diseases, with approximately a tripling of risk with each decade of life. Coronary fatty streaks can begin to form in adolescence. It is estimated that 82 percent of people who die of coronary heart disease are 65 and older. Simultaneously, the risk of stroke doubles every decade after age 55.

2. Sex

Men are at greater risk of heart disease than premenopausal women. Once past menopause, it has been argued that a woman’s risk is similar to a man’s although more recent data from the WHO and UN disputes this. If a female has diabetes, she is more likely to develop heart disease than a male with diabetes.

3. Angina (Chest Paint)

Angina is chest pain or discomfort caused when your heart muscle doesn't get enough oxygen-rich blood. It may feel like pressure or squeezing in your chest. The discomfort also can occur in your shoulders, arms, neck, jaw, or back. Angina pain may even feel like indigestion.

4. Resting Blood Pressure

Over time, high blood pressure can damage arteries that feed your heart. High blood pressure that occurs with other conditions, such as obesity, high cholesterol or diabetes, increases your risk even more.

5. Serum Cholesterol

A high level of low-density lipoprotein (LDL) cholesterol (the "bad" cholesterol) is most likely to narrow arteries. A high level of triglycerides, a type of blood fat related to your diet, also ups your risk of a heart attack. However, a high level of high-density lipoprotein (HDL) cholesterol (the "good" cholesterol) lowers your risk of a heart attack.

6. Fasting Blood Sugar

Not producing enough of a hormone secreted by your pancreas (insulin) or not responding to insulin properly causes your body's blood sugar levels to rise, increasing your risk of a heart attack.

7. Resting ECG

For people at low risk of cardiovascular disease, the USPSTF concludes with moderate certainty that the potential harms of screening with resting or exercise ECG equal or exceed the potential benefits. For people at intermediate to high risk, current evidence is insufficient to assess the balance of benefits and harms of screening.

8. Maximum Heart Rate Achieved

The increase in cardiovascular risk, associated with the acceleration of heart rate, was comparable to the increase in risk observed with high blood pressure. It has been shown that an increase in heart rate by 10 beats per minute was associated with an increase in the risk of cardiac death by at least 20%, and this increase in the risk is similar to the one observed with an increase in systolic blood pressure by 10 mm Hg.

9. Exercise Induced Angina

The pain or discomfort associated with angina usually feels tight, gripping or squeezing, and can vary from mild to severe. Angina is usually felt in the center of your chest but may spread to either or both of your shoulders, or your back, neck, jaw or arm. It can even be felt in your hands. o Types of Angina a. Stable Angina / Angina Pectoris b. Unstable Angina c. Variant (Prinzmetal) Angina d. Microvascular Angina.

10. Peak exercise ST segment

A treadmill ECG stress test is considered abnormal when there is a horizontal or down-sloping ST-segment depression ≥ 1 mm at 60–80 ms after the J point. Exercise ECGs with up-sloping ST-segment depressions are typically reported as an 'equivocal' test. In general, the occurrence of horizontal or down-sloping ST-segment depression at a lower workload (calculated in METs) or heart rate indicates a worse prognosis and higher likelihood of multi-vessel disease. The duration of ST-segment depression is also important, as prolonged recovery after peak stress is consistent with a positive treadmill ECG stress test. Another finding that is highly indicative of significant CAD is the occurrence of ST-segment elevation > 1 mm (often suggesting transmural ischemia); these patients are frequently referred urgently for coronary angiography.

6. The Approach

6.1 Strategy

- Short Data Visualization. This task details the visualization aspect of the data analysis.
- Data cleaning. This task relates to the data cleaning aspect of the analysis.
- Insights generation. Analysis might lead to insights generation to generate unique and visually understandable insights.
- Feature engineering. This task related to the feature engineering aspect of the analysis.
- Start analysis with Random Forest model. Compare them to two simple models - decision tree and random forest.
- Continue AdaBoost model with selected features. Use Cross validation and explore partial effects of features on models.
- Compare models results during grid Search process. Find the best performed model with best model parameters.
- Conclusions

6.2 Evaluation Method

Measurement of any model ability to generalize data requires a clear metric.

Here will use well-studied several metrics at the same time. In conclusion we may select the one that might be more important to some observer.

Since the dataset is limited with data, a part of it will be used to train the model (80%), and the rest of the data (20%) will be used solely to evaluate with mentioned metrics.

7. Solution

```
##### Importing required libraries

import pandas as pd
import numpy as np
import time
import gc
from tqdm import tqdm
import warnings
warnings.filterwarnings("ignore")

# https://matplotlib.org/
import matplotlib.pyplot as plt

# https://seaborn.pydata.org/
import seaborn as sns

# https://scikit-learn.org/stable/about.html#citing-scikit-learn
from sklearn.model_selection import train_test_split, GridSearchCV,
StratifiedKFold, KFold, cross_val_score, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, AdaBoostClassifier, ExtraTreesClassifier
from sklearn.metrics import roc_curve, roc_auc_score, mean_squared_error
from sklearn.pipeline import Pipeline
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.inspection import plot_partial_dependence,
partial_dependence
from sklearn.decomposition import PCA
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
```

7.1 Load dataset

```
# Original dataset
df = pd.read_csv('datasets/dataset.csv')
display(df.head(2), 'Dataset is small with only {}
observation'.format(df.shape[0]))
```

	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

```
'Dataset is small with only 303 observation'
```

7.1.1 Check data types and missing values

```
assert df.dtypes.any() != object

# Check if there are any missing values
df.isnull().values.any()
```

```
False
```

There are only numerical features and no NaN and no missing values.

7.2 Short Data Visualization

7.2.1 Features correlation

```
corr = df.corr()
corr.style.background_gradient(cmap='PRGn', low=0.15)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326	0.068001	-0.225439
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280937
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144931
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803	-0.085239
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137230
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439	0.421741
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

Highly correlated features with target variable can increase error on new (unseen data), so their influence has to be reduced.

7.2.2 Plot distribution of correlated features

```
fig = plt.figure()
sns.pairplot(df, vars=['age', 'trestbps', 'chol', 'thalach', 'oldpeak'],
hue='target').fig.suptitle('Distribution of highly correlated features
over target variable')
plt.subplots_adjust(top=0.9)
plt.show()
```

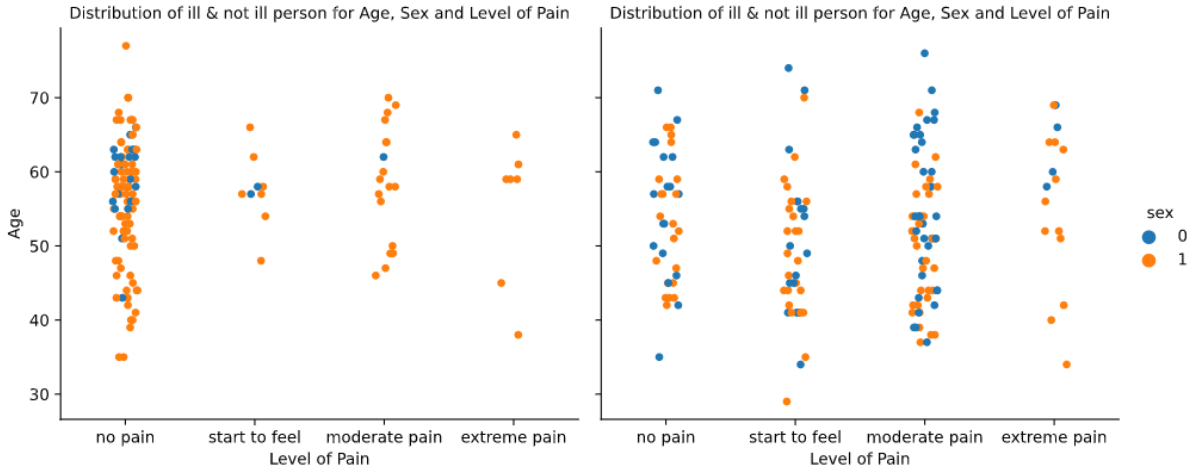
```
<Figure size 432x288 with 0 Axes>
```

From the above chart, it is to be noticed that some data can be separated quite well (different colors for the target variable on the chart). Here you can also see what distribution of the correlating variables have.

7.2.3 Explore distribution of target variable (ill & not ill) with Age, Sex, and Level of Pain features

```
# Plot
fig = plt.figure(figsize=(10, 10))
g = sns.catplot(x='cp', y='age', hue='sex', col='target', data=df)

# Build plot graph
(g.set_axis_labels("Level of Pain", "Age").set_xticklabels(["no pain",
"start to feel", "moderate pain", "extreme
pain"]).set_titles('Distribution of ill & not ill person for Age, Sex
and Level of Pain'))
plt.subplots_adjust(top=0.8)
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```



The above plot graph gives several ideas about the ***distribution of healthy and sick people, depending on age, gender and level of pain***.

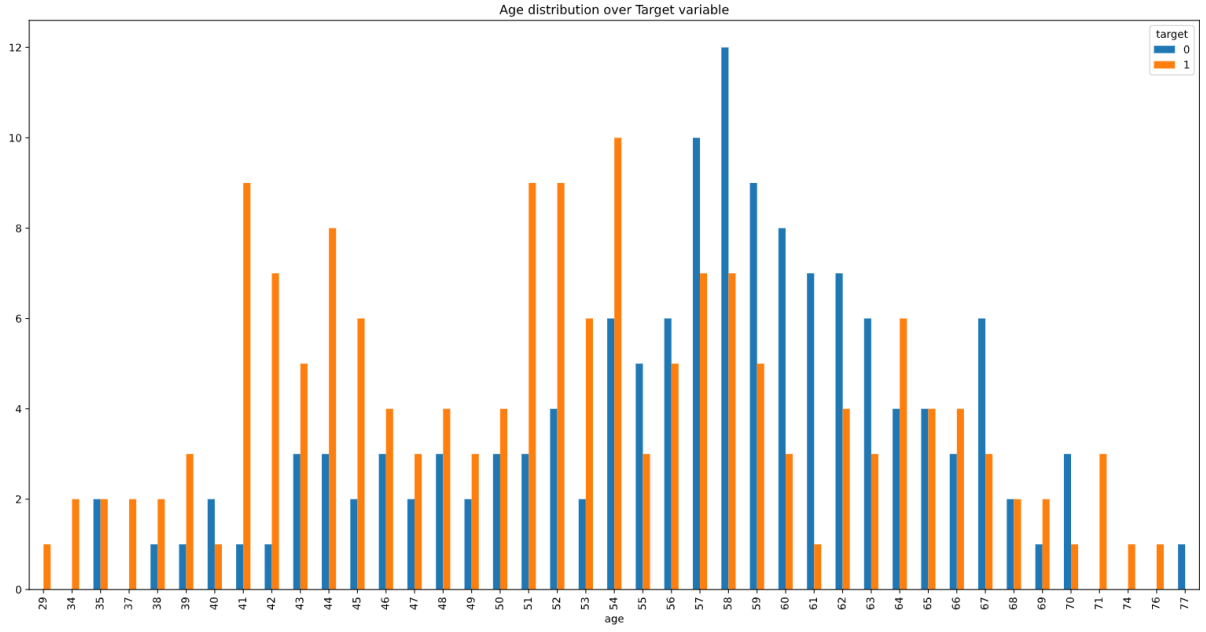
We see that the ill group is more normally distributed by gender and density.

Medium pain is the most clear feature, which, oddly enough, works for all ages.

We observe some less dense zones, but not many outlays. And it is obvious that some groups contain very little data - for example, healthy people with severe pain.

7.2.4 Age distribution

```
pd.crosstab(df.age, df.target).plot(kind='bar', figsize=(20, 10))
plt.title('Age distribution over Target variable')
plt.show()
```

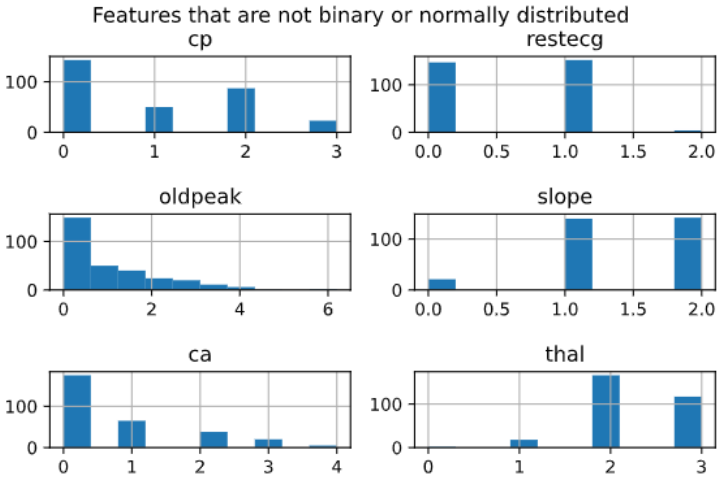


Two peaks of the disease: 41-46 yeas=rs old and 51-54 years old, it can be used to create categorical variables.

7.3 Data cleaning and transforming

Check features that are not binary and not normally distributed

```
fig, axarr = plt.subplots(3, 2)
fig.suptitle("Features that are not binary or normally distributed",
fontsize=12)

plt.subplot(3, 2, 1, title='cp')
df.cp.hist()

plt.subplot(3, 2, 2, title='restecg')
df.restecg.hist()

plt.subplot(3, 2, 3, title='oldpeak')
df.oldpeak.hist()

plt.subplot(3, 2, 4, title='slope')
df.slope.hist()

plt.subplot(3, 2, 5, title='ca')
df.ca.hist()

plt.subplot(3, 2, 6, title='thal')
df.thal.hist()

plt.subplots_adjust(top=1.5)
fig.tight_layout()

fig.subplots_adjust(top=0.88)
plt.show();
```



Now we transform cp, restecg, oldpeak, slope, ca & thal features.

```
# Function for One Hot for transformation
def encode(data, col):
    return pd.concat([data, pd.get_dummies(col, prefix=col.name)],
axis=1)

df = encode(df, df.cp)
df = encode(df, df.thal)
df = encode(df, df.slope)
df = encode(df, df.restecg)
df = encode(df, df.ca)

# Dropping old columns after transformation
df.drop(['cp', 'thal', 'slope', 'restecg', 'ca'], axis=1, inplace=True)
df.head()
```

	age	sex	trestbps	chol	fbs	thalach	exang	oldpeak	target	cp_0	...	slope_1	slope_2	restecg_0	restecg_1	restecg_2	ca_0	ca_1	ca_2	ca_3	ca_4
0	63	1	145	233	1	150	0	2.3	1	0	...	0	0	1	0	0	1	0	0	0	0
1	37	1	130	250	0	187	0	3.5	1	0	...	0	0	0	1	0	1	0	0	0	0
2	41	0	130	204	0	172	0	1.4	1	0	...	0	1	1	0	0	1	0	0	0	0
3	56	1	120	236	0	178	0	0.8	1	0	...	0	1	0	1	0	1	0	0	0	0
4	57	0	120	354	0	163	1	0.6	1	1	...	0	1	0	1	0	1	0	0	0	0

5 rows × 28 columns

7.3.1 Check the numeric values

```
df[['age', 'trestbps', 'chol', 'thalach']].describe()
```

	age	trestbps	chol	thalach
count	303.000000	303.000000	303.000000	303.000000
mean	54.366337	131.623762	246.264026	149.646865
std	9.082101	17.538143	51.830751	22.905161
min	29.000000	94.000000	126.000000	71.000000
25%	47.500000	120.000000	211.000000	133.500000
50%	55.000000	130.000000	240.000000	153.000000
75%	61.000000	140.000000	274.500000	166.000000
max	77.000000	200.000000	564.000000	202.000000

7.3.2 Scale these numeric values

```
scale = StandardScaler()
se = pd.DataFrame(scale.fit_transform(df[['age', 'trestbps', 'chol',
'thalach']]))
```

7.3.3 Prepare data for modeling

```
df = pd.concat([df, se], axis=1)
y = df.target

df.drop(['target', 'age', 'trestbps', 'chol', 'thalach'], axis=1,
inplace=True)
df.rename(columns={0: 'target', 1: 'age', 2: 'trestbps', 3: 'chol', 4:
'thalach'}, inplace=True)
df.head()
```

	sex	fbs	exang	oldpeak	cp_0	cp_1	cp_2	cp_3	thal_0	thal_1	...	restecg_2	ca_0	ca_1	ca_2	ca_3	ca_4	target	age	trestbps	chol
0	1	1	0	2.3	0	0	0	1	0	1	...	0	1	0	0	0	0	0.952197	0.763956	-0.256334	0.015443
1	1	0	0	3.5	0	0	1	0	0	0	...	0	1	0	0	0	0	-1.915313	-0.092738	0.072199	1.633471
2	0	0	0	1.4	0	1	0	0	0	0	...	0	1	0	0	0	0	-1.474158	-0.092738	-0.816773	0.977514
3	1	0	0	0.8	0	1	0	0	0	0	...	0	1	0	0	0	0	0.180175	-0.663867	-0.198357	1.239897
4	0	0	1	0.6	1	0	0	0	0	0	...	0	1	0	0	0	0	0.290464	-0.663867	2.082050	0.583939

5 rows × 27 columns

7.4 Feature Engineering

Lets get rid of multicollinearity with target variable with PCA

```
pca = PCA(n_components=10)
df_pca = pca.fit_transform(df)
```

7.4.1 Check the optimal number of components for PCA

```
total = 0
for i, component in enumerate(pca.components_):
    print("{} component: {}% of initial variance".format(i+1,
round(100*pca.explained_variance_ratio_[i], 2)))
    total += pca.explained_variance_ratio_[i]

print()
print("Total % of initial variance {}".format(total))
```

```
1 component: 26.74% of initial variance
2 component: 13.81% of initial variance
3 component: 10.91% of initial variance
4 component: 9.81% of initial variance
5 component: 6.36% of initial variance
6 component: 5.24% of initial variance
7 component: 5.18% of initial variance
8 component: 3.55% of initial variance
9 component: 3.28% of initial variance
10 component: 2.88% of initial variance

Total % of initial variance 0.8775800114335137
```

To explain data variance 10 PCA components were selected

7.4.2 What is the cross validation score **without** 10 PCA components?

```
# Check the RandomForestClassifier
cross_val_score(RandomForestClassifier(n_estimators=50,
random_state=13), df, y, scoring='roc_auc').mean()
```

```
0.907243065576399
```

```
# Check the LogisticRegression
cross_val_score(LogisticRegression(random_state=13), df, y,
scoring='roc_auc').mean()
```

```
0.9181417348084014
```

7.4.3 What is the cross validation score **with** 10 PCA components?

```
# Check the RandomForestClassifier
cross_val_score(RandomForestClassifier(n_estimators=10,
random_state=13), df_pca, y, scoring='roc_auc').mean()
```

```
0.8662417829084494
```

```
# Check the LogisticRegression
cross_val_score(LogisticRegression(random_state=13), df_pca, y,
scoring='roc_auc').mean()
```

```
0.9201298701298702
```

Random Forest Classifier gives us a little worse result. Logistic Regression has small improvement.

Since there is not much data, it is likely that the transformation of the PCA will negatively affect the final results.

Let's find other ways to select (or delete) features. Especially collinear ones.

7.4.4 Check the variance of features without PCA

```
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectKBest, f_classif
```

Use threshold (75%) to reduce number of feature variance.

```
df_variance = VarianceThreshold(0.75).fit_transform(df)
display('From {} features we have {}'.format(df.shape[1],
df_variance.shape[1]))
```

```
'From 27 features we have 5'
```

We can also use other SelectKBest with AVONA stat classifier and top 5 features.

```
df_best_5 = SelectKBest(f_classif, k=5).fit_transform(df, y)
```

Let's check how PCA will change the performance.

```
df_pca_variance = VarianceThreshold(0.75).fit_transform(df_pca)
```

7.4.5 Results

```
cv_df = cross_val_score(RandomForestClassifier(n_estimators=10,
random_state=13), df, y, scoring='roc_auc').mean()

cv_var_5 = cross_val_score(RandomForestClassifier(n_estimators=10,
random_state=13), df_best_5, y, scoring='roc_auc').mean()

cv_var_75 = cross_val_score(RandomForestClassifier(n_estimators=10,
random_state=13), df_variance, y, scoring='roc_auc').mean()

cv_pca = cross_val_score(RandomForestClassifier(n_estimators=10,
random_state=13), df_pca_variance, y, scoring='roc_auc').mean()
```

```
print('------------------------')
print('Initial CV score {}\n'.format(cv_df))
print('CV score with 5 best features with SelectKBest sklearn class
{}\n'.format(cv_var_5))
print('CV score with Threshold (75%) using VarianceThreshold sklearn
class {}\n'.format(cv_var_75))
print('CV score on PCA transformed data with Threshold (75%) using
VarianceThreshold sklearn class {}'.format(cv_pca))
print('------------------------')
```

```
------------------------
Initial CV score 0.8807359307359308

CV score with 5 best features with SelectKBest sklearn class
0.9048140131473463

CV score with Threshold (75%) using VarianceThreshold sklearn class
0.7226671476671477

CV score on PCA transformed data with Threshold (75%) using
VarianceThreshold sklearn class 0.766333974667308
------------------------
```

The initial value of 1 is due to the small amount of data. And we can conclude that PSA and a feature reduction by the threshold of 0.75 does not increase or even worsens the results.

Select seems to do better. But a small amount of data allows us to use it fully.
So we are going to work with a full set of features.

And in the final test of our hypothesis we use one more solution.

7.4.6 Use Feature selection to check our hypothesis

```
# http://rasbt.github.io/mlxtend/
from mlxtend.feature_selection import SequentialFeatureSelector

# Split data
X_train, X_test, y_train, y_test = train_test_split(df, y,
train_size=0.2)
print('Total {} features now'.format(X_train.shape[1]))
```

```
Total 27 features now
```

Start selector RandomForestClassifier. You can change verbose to 2, just to see how roc_auc changes

```
selector =
SequentialFeatureSelector(RandomForestClassifier(random_state=13),
scoring='roc_auc', verbose=0,  k_features=10, forward=True, n_jobs=-1)

# Train selector
selector.fit(X_train, y_train)
```

```
SequentialFeatureSelector(estimator=RandomForestClassifier(random_state=
13), k_features=10, n_jobs=-1, scoring='roc_auc')
```

```
print('features idx {}, features name {}, max score
{}'.format(selector.k_feature_idx_, selector.k_feature_names_,
selector.k_score_))
```

```
features idx (0, 2, 4, 5, 6, 7, 9, 10, 13, 23), features name ('sex',
'exang', 'cp_0', 'cp_1', 'cp_2', 'cp_3', 'thal_1', 'thal_2', 'slope_1',
'target'), max score 0.928125
```

Here is the result. We can use just 9 features (out of 14) to get value of roc_auc - 0.93

Once again - 303 observations is not enough to use such an approach to reduce noise (via feature reduction or transformation). So I propose to **use the whole data set**. Even feature engineering in such a case will affect the accuracy of classification on new data.

But for accurate reliability and applying a scientific approach to experiments - we will save the data set with a reduced number of features (10 out of 14)

```
df_less_features = df[list(selector.k_feature_names_)]
df_less_features.shape
```

```
(303, 10)
```

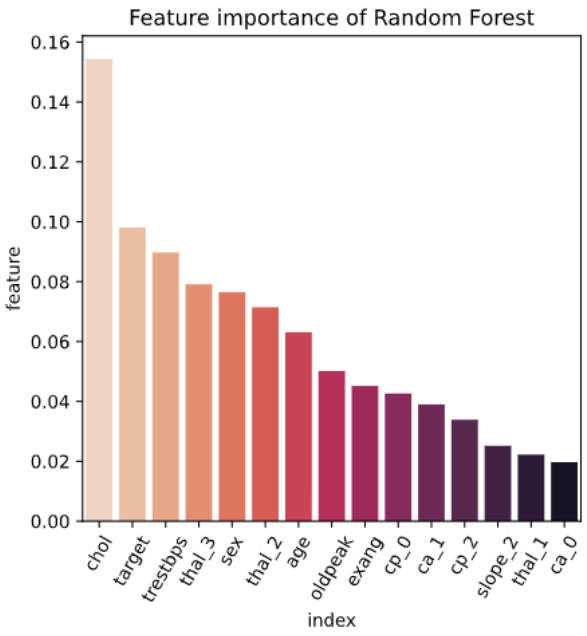
7.5 Use Random Forest and Logistic Regression for Feature Importance

Let's first look at the feature importance rank.

```
random_forest = RandomForestClassifier(n_estimators=10, max_features=2,
random_state=13, verbose=False).fit(X_train, y_train)
```

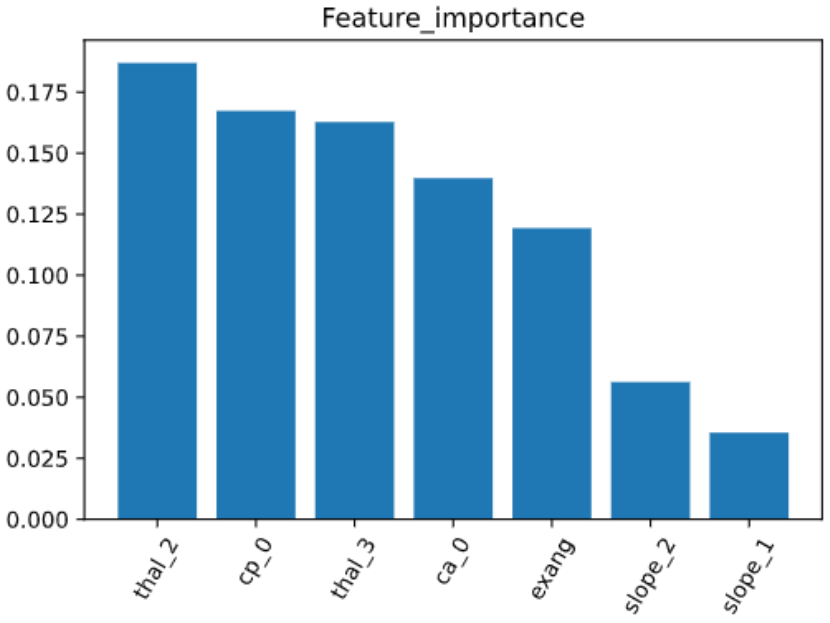
7.5.1 Plot graph of Feature Importance

```
def plot_features_importance(model, X_train, X_test, y_test):
    ddd = pd.DataFrame(model.feature_importances_, X_train.columns,
columns=["feature"])
    ddd = ddd.sort_values(by='feature', ascending=False).reset_index()
    plt.figure(figsize=[5,5])
    sns.barplot(x='index', y='feature', data=ddd[:15],
palette="rocket_r")
    plt.title('Feature importance of Random Forest')
    plt.xticks(rotation=60)
    plt.show();
    display('ROC_AUC is {}'.format(roc_auc_score(y_test,
random_forest.predict_proba(X_test)[:,1])))

# Run function
plot_features_importance(random_forest, X_train, X_test, y_test)
```




```
'ROC_AUC is 0.8478305084745763'
```

```
# Plot
plt.figure(figsize=(10,6))
plt.bar(X_train.columns, logreg.coef_[0])
plt.title('Feature importance of LogisticRegression')
plt.xticks(rotation=60)
plt.show();
```



We can use LogisticRegression coeffs to see 'influence' of features.

Random_forest and LogisticRegression see data differently. LogisticRegression is more about linear relations, while random_forest catches non-linear as well.

For example, the cp_0 feature in the random_forest model is 2th, while for logreg it is negative! At the same time ca_0 in both models are number 1. Check age as well.

Such insights help to understand which features have more linear or nonlinear processes. (in certain point it is mixture of them)

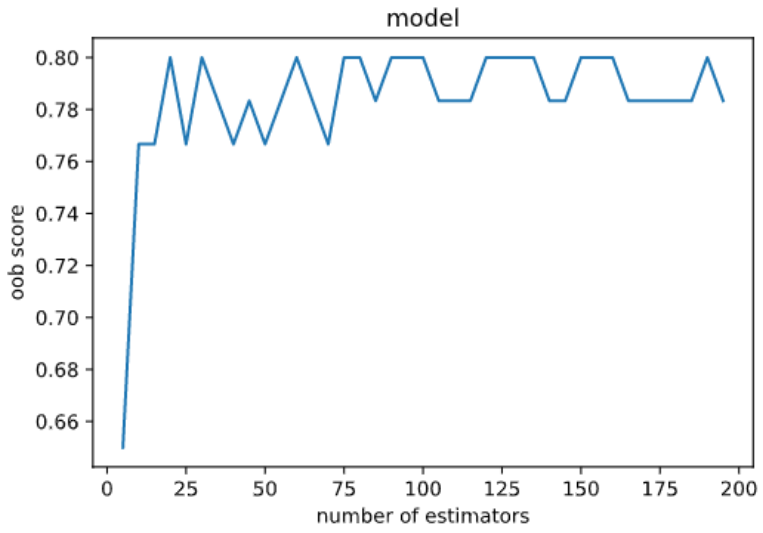
Although, for me all possible ways to use these differences are not yet available - I'm sure they can be useful for analyzing more complex data.

This can give confidence, for example, in the **choice of transformation methods at the date cleaning stage**.

7.5.2 Estimation of estimators random_forest needs for best performance

```
def graph(model, X_train, y_train):
    obb = []
    est = list(range(5, 200, 5))
    for i in tqdm(est):
        random_forest = model(n_estimators=i, random_state=13,
max_features=2, verbose=0, oob_score=True, n_jobs=-1)
        random_forest.fit(X_train, y_train)
        obb.append(random_forest.oob_score_)

    print('Max oob {} and number of estimators {}'.format(max(obb),
est[np.argmax(obb)]))
    plt.plot(est, obb)
    plt.title('model')
    plt.xlabel('number of estimators')
    plt.ylabel('oob score')
    plt.show();

graph(RandomForestClassifier, X_train, y_train)
```

```
100%|██████████| 39/39 [00:12<00:00,  3.20it/s]
Max oob 0.8 and number of estimators 20
```



OOB score (out-of-bag) is used here as a more stable statistic for trees that better track Trees estimator performance.

We see that **20 estimators** is enough for the best score.

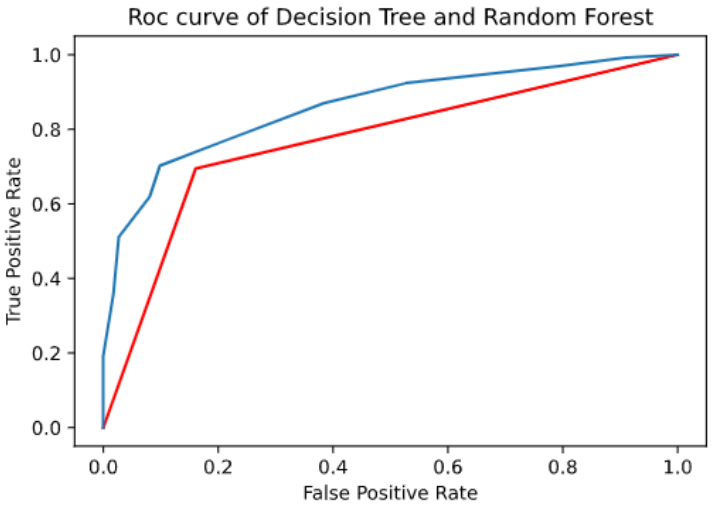
7.5.3 Comparison of Decision Tree and Random Forest by checking performances ROC metrics

```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
roc_auc_score(y_test, dt.predict_proba(X_test)[:,1])
```

```
0.7677627118644067
```

```
f_dt, t_dt, tt_dt = roc_curve(y_test, dt.predict_proba(X_test)[:,1])
f_rf, t_rf, tt_rf = roc_curve(y_test,
random_forest.predict_proba(X_test)[:,1])
```

```
plt.plot(f_dt, t_dt, c='r', label='Decision Tree')
plt.plot(f_rf, t_rf, label='Random Forest')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Roc curve of Decision Tree and Random Forest')
plt.show();
```



Here you can clearly see how the random forest more accurately describes (line is more flexible with bigger total square) the data, with more **than 0.90 ROC_AUC score**.

7.6 Testing with more advanced models - AdaBoost.

We consider the variables Thal, Sex, ChestPain and Chol. We want to compare how such algorithms perform with limited data compared to more simple one with complete dataset.

```
X_t, Xt, y_t, yt = train_test_split(df[['thal_0', 'thal_1', 'thal_2',
'thal_3', 'sex', 'cp_0', 'cp_1', 'cp_2', 'cp_3']], y)
```

```
adaboost = AdaBoostClassifier(n_estimators=10, random_state=13)
adaboost.fit(X_t, y_t)

roc_auc_score(yt, adaboost.predict_proba(Xt)[:,1])
```

```
0.898263888888889
```

7.6.1 Using GridSearch to find better performance

```
# Parameters
param_grid = [{'n_estimators' : list(range(25, 150, 25))}]
cv = StratifiedKFold(random_state=13)
```

```
adaboost_1 = AdaBoostClassifier(random_state=11)
grid = GridSearchCV(adaboost_1, param_grid=param_grid,
scoring='roc_auc', cv=cv)
```

```
grid.fit(X_t, y_t)
```

```
GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=13,
shuffle=False), estimator=AdaBoostClassifier(random_state=11),
param_grid=[{'n_estimators': [25, 50, 75, 100, 125]}],
scoring='roc_auc')
```

```
grid.best_estimator_, grid.best_params_, grid.best_score_
```

```
(AdaBoostClassifier(n_estimators=25, random_state=11), {'n_estimators':
25}, 0.822095238095238)
```

```
cv = cross_val_score(grid.best_estimator_, X_t, y_t, scoring='roc_auc',
cv=5)
print('CV score is {}'.format(np.mean(cv)))
```

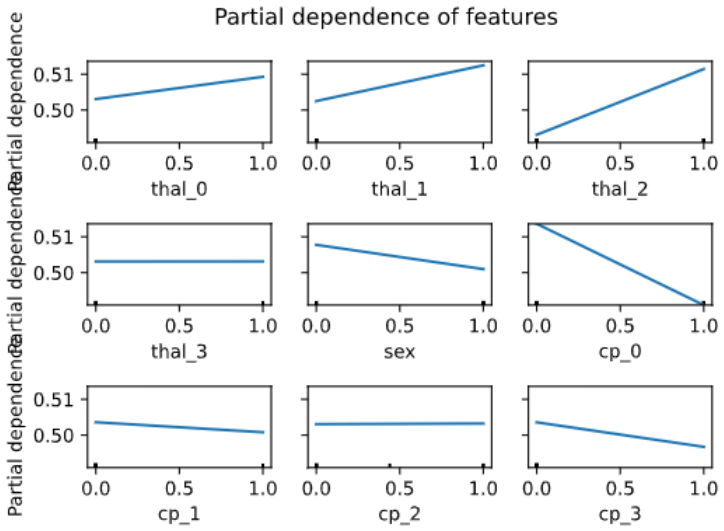
```
CV score is 0.822095238095238
```

The results are a bit better!

But complex algorithms on small data can behave unstable.

Lets calculate the partial effects for the previous subset of variables (thal_0', 'thal_1', 'thal_2', 'thal_3', 'sex', 'cp_0', 'cp_1', 'cp_2', 'cp_3)

```
# Save features
features = list(X_t.columns)

# Plot
plot_partial_dependence(grid.best_estimator_, X_t, features, n_jobs=3,
grid_resolution=30)
fig = plt.gcf()
fig.suptitle('Partial dependence of features ')
fig.subplots_adjust(hspace=1)
plt.show();
```



Partial dependence plots show the dependence between the target function 2 and a set of ‘target’ features, marginalizing over the values of all other features (the complement features).

Here we see two strong dependence (a linear relationship). People get ill with thal_2 feature (top left). cp_0 (no chest pain) says that they are not ill at all.

thal_0', 'thal_1' - weak dependencies but with small positive rate. all others are not dependent.

7.7 Calculate and compare models (Linear, boosting and others) with and without feature reduction

Metrics are:

- Mean squared error
- Prediction accuracy
- ROC_AUC

```
X_train, X_test, y_train, y_test = train_test_split(df, y)
df_less_features.shape, df.shape
```

```
((303, 10), (303, 27))
```

```
n_jobs = -1
random_state = 13

##### Pipeline

pipe_rf = Pipeline([('rf',
RandomForestClassifier(random_state=random_state, n_jobs=n_jobs))])

pipe_ada = Pipeline([('ada',
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
random_state=random_state))])

pipe_gbc = Pipeline([('gbc',
GradientBoostingClassifier(random_state=random_state))])

pipe_ext = Pipeline([('ext',
ExtraTreesClassifier(random_state=random_state, n_jobs=n_jobs))])

pipe_bag = Pipeline([('bag', BaggingClassifier(n_jobs=n_jobs,
random_state=random_state))])

pipe_svc = Pipeline([('svc', SVC(random_state=random_state,
probability=True))])


##### Params
```

```
cv = StratifiedKFold(shuffle=False, n_splits=5, random_state=13)

grid_params_rf = {
                'rf__criterion': ['entropy', 'gini'],
                'rf__min_samples_leaf': [10, 15, 20],
                'rf__max_depth': [10, 15, 20],
                'rf__n_estimators' : range(40, 120, 20)
                }


grid_params_ada = {
                'ada__learning_rate' : [0.5, 0.8, 1],
                'ada__n_estimators': [25, 50]
                }

grid_params_gbc = {
                'gbc__learning_rate' : list(np.linspace(0.01, 0.5, 5))
                }

grid_params_ext = {
                'ext__min_samples_leaf': [15, 18, 20],
                'ext__max_depth': [10, 15, 16],
                'ext__n_estimators' : [50, 100, 150]
                }

grid_params_bag = {'bag__base_estimator': [KNeighborsClassifier()],
                'bag__n_estimators': [50, 70, 80] #list(range(10, 80,
10))
                    }

grid_params_svc = {
                    'svc__C': [1, 3, 4]
                    }

### RandomizedSearchCV

gs_rf = RandomizedSearchCV(pipe_rf, param_distributions=grid_params_rf,
                     scoring='roc_auc', cv=cv, n_iter=15)

gs_ada = RandomizedSearchCV(pipe_ada,
param_distributions=grid_params_ada,
                     scoring='roc_auc', cv=cv, n_iter=15)

gs_gbc = RandomizedSearchCV(pipe_gbc,
param_distributions=grid_params_gbc,
```

```
                    scoring='roc_auc', cv=cv, n_iter=15)

gs_ext = RandomizedSearchCV(pipe_ext,
param_distributions=grid_params_ext,
                    scoring='roc_auc', cv=cv, n_iter=15)

gs_bag = RandomizedSearchCV(pipe_bag,
param_distributions=grid_params_bag,
                    scoring='roc_auc', cv=cv, n_iter=15)

gs_svc = RandomizedSearchCV(pipe_svc,
param_distributions=grid_params_svc,
                    scoring='roc_auc', cv=cv, n_iter=15)

### Dict of models
look_for = [gs_rf, gs_ada, gs_gbc, gs_ext, gs_bag, gs_svc]

model_dict = {0:'RandomForest', 1: 'Adaboost', 2: "Gradient boost",
3:'ExtraTreeBoost', 4:'Bagging', 5:'SupportVEctorClass'}
```

```
''' Function to iterate over models and obtain results'''
# Set empty dicts and list
result_acc = {}
result_auc = {}
mse = {}
models = []

# Iterate over models
for index, model in enumerate(look_for):
        start = time.time()
        print()
        print('########## Start New Model ####################')
        print('Estimator is {}'.format(model_dict[index]))
        model.fit(X_train, y_train)
        print('----------------------------------------------')
        print('best params {}'.format(model.best_params_))
        print('best score is {}'.format(model.best_score_))
        auc = roc_auc_score(y_test, model.predict_proba(X_test)[:,1])
        print('----------------------------------------------')
        print('ROC_AUC is {} and accuracy rate is {}'.format(auc,
model.score(X_test, y_test)))
        end = time.time()
        print('It lasted for {} sec'.format(round(end - start, 3)))
        print('########## End Model ##########################')
```

```
        print()
        print()
        models.append(model.best_estimator_)
        result_acc[index] = model.best_score_
        result_auc[index] = auc
        mse[index] = mean_squared_error(y_test, model.predict(X_test))
```

```
########## Start New Model ####################
Estimator is RandomForest
-----------------------------------------
best params {'rf__n_estimators': 60, 'rf__min_samples_leaf': 10,
'rf__max_depth': 10, 'rf__criterion': 'entropy'}
best score is 0.9293238095238096
-----------------------------------------
ROC_AUC is 0.8857142857142857 and accuracy rate is 0.8857142857142857
It lasted for 9.406 sec
########## End Model ##########################



########## Start New Model ####################
Estimator is Adaboost
-----------------------------------------
best params {'ada__n_estimators': 25, 'ada__learning_rate': 0.5}
best score is 0.7189285714285714
-----------------------------------------
ROC_AUC is 0.6885017421602788 and accuracy rate is 0.6885017421602788
It lasted for 0.438 sec
########## End Model ##########################



########## Start New Model ####################
Estimator is Gradient boost
-----------------------------------------
best params {'gbc__learning_rate': 0.5}
best score is 0.8868857142857143
-----------------------------------------
ROC_AUC is 0.8397212543554007 and accuracy rate is 0.8397212543554007
It lasted for 3.084 sec
########## End Model ##########################
```

```
########## Start New Model ####################
Estimator is ExtraTreeBoost
-----------------------------------------
best params {'ext__n_estimators': 100, 'ext__min_samples_leaf': 18,
'ext__max_depth': 10}
best score is 0.9264285714285714
-----------------------------------------
ROC_AUC is 0.8878048780487806 and accuracy rate is 0.8878048780487806
It lasted for 12.466 sec
########## End Model ##########################



########## Start New Model ####################
Estimator is Bagging
-----------------------------------------
best params {'bag__n_estimators': 80, 'bag__base_estimator':
KNeighborsClassifier()}
best score is 0.8941095238095238
-----------------------------------------
ROC_AUC is 0.8738675958188153 and accuracy rate is 0.8738675958188153
It lasted for 3.594 sec
########## End Model ##########################



########## Start New Model ####################
Estimator is SupportVEctorClass
-----------------------------------------
best params {'svc__C': 1}
best score is 0.9240507936507937
-----------------------------------------
ROC_AUC is 0.8738675958188153 and accuracy rate is 0.8738675958188153
It lasted for 0.291 sec
########## End Model ##########################
```

```
# Function to show results

def model_performance(df, result_acc, result_auc, mse, model_dict):
    result = pd.DataFrame([result_acc, result_auc, mse])
    result.columns = list(model_dict.values())
    result.index = ['accuracy', 'roc_auc', 'MSE']
    display(result.head(), 'shape of data is {}'.format(df.shape))
    plt.figure(figsize=(8,8))
    plt.plot(result.columns.values, result.iloc[0,:].values,
```

```
label=['accuracy'])
    plt.plot(result.columns.values, result.iloc[1,:].values,
label=['roc_auc'])
    plt.plot(result.columns.values, result.iloc[2,:].values,
label=['mse'])

    plt.xticks(rotation=60)
    plt.title('Compare performance')
    plt.legend()
    plt.show()

def plot_ExtraTreesClassifier():
    extra_tree = ExtraTreesClassifier(n_estimators=100,
min_samples_leaf=15, max_depth=15).fit(df, y)

    temp_df = pd.DataFrame(df.columns, extra_tree.feature_importances_,
columns=['feature']).reset_index().sort_values(by='index',
ascending=False)
    temp_df.columns = ['Feature_importance', 'Feature']

    plt.bar(temp_df[:7].Feature, temp_df[:7].Feature_importance)
    plt.xticks(rotation=60)
    plt.title('Feature_importance')
    plt.show();
```

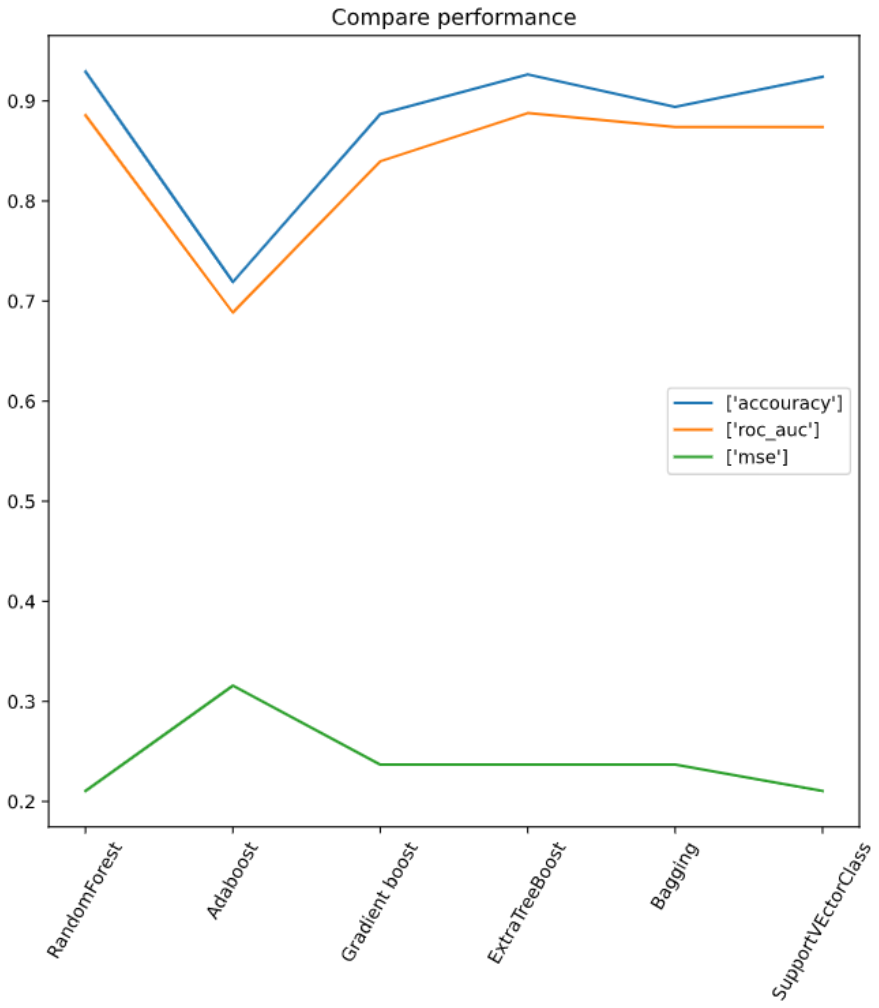
7.7.1 Results **without** feature reduction

Here we use 27 features.

```
model_performance(df, result_acc, result_auc, mse, model_dict)
```

	RandomForest	Adaboost	Gradient boost	ExtraTreeBoost	Bagging	SupportVEctorClass
accouracy	0.929324	0.718929	0.886886	0.926429	0.894110	0.924051
roc_auc	0.885714	0.688502	0.839721	0.887805	0.873868	0.873868
MSE	0.210526	0.315789	0.236842	0.236842	0.236842	0.210526

```
'shape of data is (303, 27)'
```



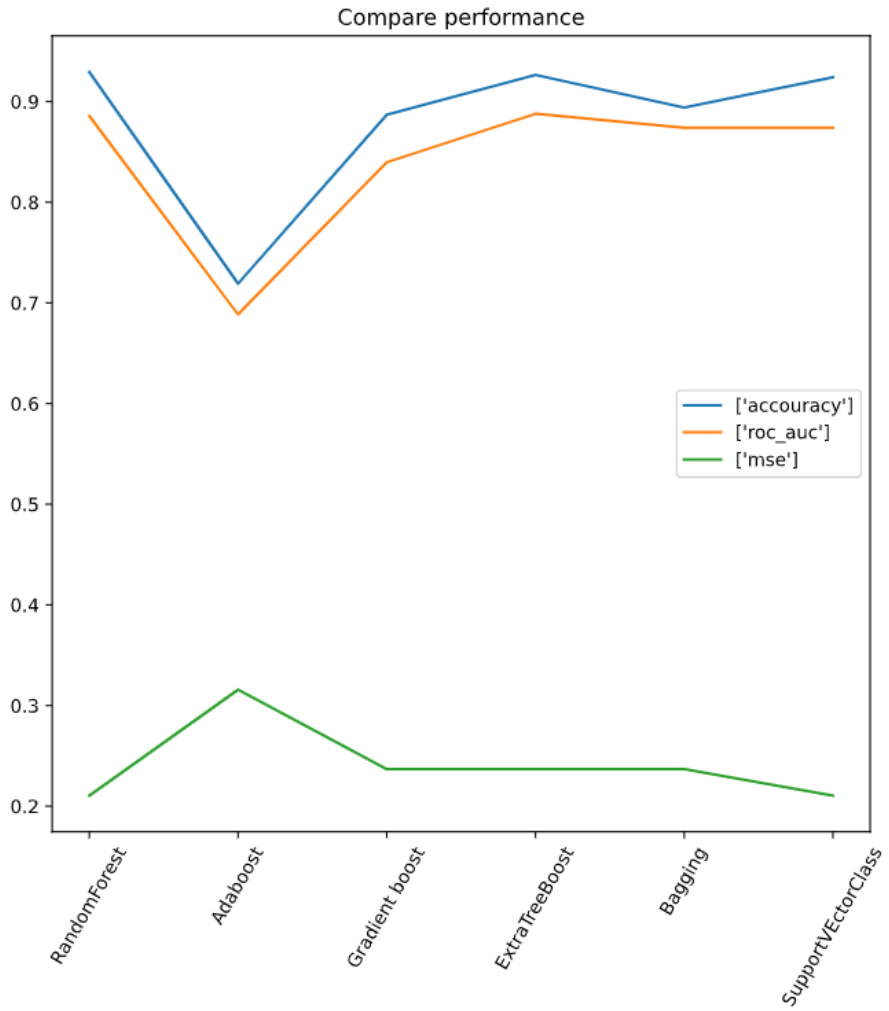
7.7.2 Results **with** feature reduction.

Here we use 10 features.

```
model_performance(df_less_features, result_acc, result_auc, mse,
model_dict)
```

	RandomForest	Adaboost	Gradient boost	ExtraTreeBoost	Bagging	SupportVEctorClass
accouracy	0.929324	0.718929	0.886886	0.926429	0.894110	0.924051
roc_auc	0.885714	0.688502	0.839721	0.887805	0.873868	0.873868
MSE	0.210526	0.315789	0.236842	0.236842	0.236842	0.210526

```
'shape of data is (303, 10)'
```

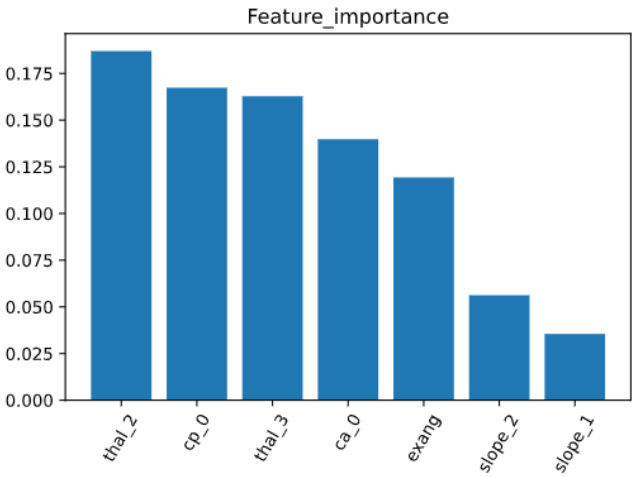


8. Conclusion

1. Here are the most important factors that influence heart disease of any age or affect any gender.

- Chest Pain
- Fixed defect or heart without defect
- Exercise induced angina
- Slopes of the peak exercise ST segment

```
plot_ExtraTreesClassifier()
```



2. With additional data records (personal profile data) it is possible to predict illness with more than 90% accuracy.

Almost all models showed quite good results.

- As expected, models built on ensembles were able to accurately determine nonlinear relationships.
- Processing speed is fast enough to use it on personal computers.
- Recugin the variables from 27 to 10 did not significantly affect the result, which may reduce the cost (time) of diagnosing patients.
- Conducted EDA allowed to clearly articulate a risk group by age, sex and other characteristics.

3. Metrics and evaluation models.

The presented metrics evaluate different approaches of models in the process of data evaluation.

The choice among them, accordingly, ***will depend on the priorities of management***, which sets priorities.

For example, if we assume that errors in classification for a healthy person is less significant (Type I error) than errors in classification of a patient, then the appropriate metric would be ***Precision***.

4. How to use results and models?

Such studies can help with early diagnosis of diseases, helping to reduce the risk of complications.

Also modeling can be used by other specialists in the medical field without the high cost of IT infrastructure with acceptable accuracy and speed.

Naturally, this modeling can be used by a wider circle: patients, insurance companies, researchers etc.