Cardiovascular diseases: risk factors and attempts of prediction

Olga Ivanova

Abstract

Cardiovascular diseases (CVDs) are the number 1 cause of death globally: more people die annually from CVDs than from any other cause. This project contains 2 parts: the first part is dedicated to exploration of mortality due to CVDs and its correlation with risk factors (raised level of cholesterol and overelevated blood pressure); it includes an attempt of mortality prediction based on the percentage of people with those risk factors; the second part is devoted to prediction of CVD presence for given patients, knowing the values of a number of CVD attributes for these patients.

For the part I I used datasets containing summary tables of mortality due to different causes in 2000-2015, a table of percentage of people with raised total cholesterol and a table of percentage of people with raised blood pressure; for the part II I used the dataset consisting of CVD attributes and observations of patients. I used 2 regression analyses to predict the values of mortality due to CVD in part I and 3 classification analyses to predict the presence of CVD for a given patient in part II.

Part I of this project allowed me to conclude that the prediction of the mortality due to cardiovascular diseases is not enough effective in these circumstances, while part II provided the predictions of the presence of cardiovascular disease which were feasible using all 3 classifiers. Part II also suggested the preferred classifier for disease prediction basing on estimation of the type of errors.

Motivation

Cardiovascular diseases (CVDs) are a group of disorders of the heart and blood vessels, associated with heart attacks and strokes, which are mainly caused by a blockage of blood flow in the heart or in the brain. The most common reason for this is a build-up of fatty deposits on the inner walls of the blood vessels that supply the heart or the brain. CVDs are the number 1 cause of death globally: more people die annually from CVDs than from any other cause (http://www.who.int). The death due to CVD is terrible in its unpredictability and rapidity. It looks like a normal healthy person walks or does his regular job, then suddenly he falls and dies. Unexpectedly, without visible causes.

Of course, in fact there is a cause. The causes of heart attacks and strokes are usually a combination of risk factors like raised blood pressure, raised blood glucose levels, raised blood lipids, overweight and obesity. In this project I would like, first of all, to estimate the correlation between some risk factors and mortality due to CVDs and to predict the mortality due to CVDs using these risk factors. Secondly, which is even more important, I would like to make a prognosis of CVD presence for given patients, knowing the values of a number of risk factors. This could be useful to promptly diagnose CVD, to prevent its consequences and to save lives.

Datasets

1. Summary tables of mortality, estimates by cause, age and sex, by country, 2000–2015. 4 files .xls 16MB (sheet "Deaths All ages" – table 627*187) for each year: 2000, 2005, 2010, 2015 from http://www.who.int/healthinfo/global_burden_disease/estimates/en/

2. Table of percentage of people with raised total cholesterol (≥6.2 mmol/L) in 2008, estimates by country. File .csv 27KB (table 194*9) from http://apps.who.int/gho/data/node.main.A887?lang=en)

3. Table of percentage of people with raised blood pressure (SBP ≥ 140 OR DBP ≥ 90), age-standardized (%), estimates by country. File .csv 457KB (table 198*124) from http://apps.who.int/gho/data/node.main.A875STANDARD?lang=en

4. Dataset consisting of 14 attributes and 303 observations that were used for the purpose of heart disease classification of a given patient. File .csv 12KB (table 303*14) from https://www.kaggle.com/danimal/heartdiseaseensembleclassifier

Data Preparation and Cleaning - 1

This work consists of 2 parts: in part I I used the datasets #1, #2 and #3 described above (causes of death, cholesterol and blood pressure), in part II I used the dataset #4 (heart disease classification).

Part I.

- I read the sheet "Deaths All ages" of the file .xls containing the causes of death in 2000, using pd.read_excel(). It was very noisy and contained a lot of missing data and unnecessary information. Using slices, function dropna() and Series.str.contains('text') I extracted the data concerning all causes of death, the cause of death – cardiovascular diseases and the population in 2000. I performed the same manipulations with files, containing the causes of death in 2005, 2010 and 2015 and merged them, using the common feature – the column 'countries'.
- I read the file .csv containing the dataset #2 using pd.read_csv(). From this dataset I needed only 2 columns: the percentage of defined population of both sexes with total cholesterol ≥ 6.2 mmol/l (age-standardized) and the countries to merge this dataset with dataset #1. I encountered a problem that the values for percentage were not digital and contained additional useless values in brackets. I imported the module "re" providing regular expression matching operations and used re.sub to delete text with brackets. Then I used the pd.to_numeric to transform the type of data to float. I merged datasets #1 and #2 using the common feature 'countries'.
- I performed the same manipulations with the file containing percent of defined population with raised blood pressure (systolic blood pressure ≥ 140 OR diastolic blood pressure ≥ 90). The difference was that this dataset contained the values since 1975 and until 2015 and for further analysis I extracted the data for 2000, 2005, 2010 and 2015.

Data Preparation and Cleaning - 2

Part II

- I read the file .csv containing the dataset #4 using pd.read_csv(). It was not very noisy, but it contained some missing values, which were marked as '?'. So I set the parameter na_values="?“ in pd.read_csv() and after exploration I cleaned these values by means of dropna().
- The attributes to predict were:

age	(age in years)
sex	(1 = male, 0 = female)
cp	(chest pain type: 1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic
trestbps	(resting blood pressure (in mm Hg on admission to the hospital))
chol	(serum cholesterol in mg/dl)
fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
restecg	(resting electrocardiographic results: 0 = normal, 1 = having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria
thalach	(maximum heart rate achieved)
exang	(exercise induced angina: 1 = yes, 0 = no)
oldpeak	(ST depression induced by exercise relative to rest)
slope	(the slope of the peak exercise ST segment: 1 = upsloping, 2 = flat, 3 = downsloping
ca	(number of major vessels (0-3) colored by fluoroscopy)
thal	(3 = normal, 6 = fixed defect, 7 = reversable defect)
pred_attribute	(the predicted attribute: diagnosis of heart disease (angiographic disease status): 0 = < 50% diameter narrowing (Healthy), 1 = > 50% diameter narrowing (Sick))

Research Questions

Part 1. Global mortality, deaths due to cardiovascular diseases and risk factors.

1. How the mortality due to cardiovascular diseases is correlated with risk factors: high cholesterol level and raised blood pressure?

2. Can we predict the mortality due to cardiovascular diseases, if we know the percentage of population with the above-mentioned risk factors in each country and the related mortality in some countries?

Part 2. Cardiovascular disease presence prediction for a given patient.

3. Can we predict the presence of cardiovascular disease using different classification analyses, if we have the information about the number of patients, having or not this disease depending on some attributes?

Methods

Part I

1. To analyse the link between mortality due to cardiovascular diseases (CVD) and risk factors I used pandas.DataFrame.corr() and visualized the results with seaborn.pairplot.
2. In attempt to predict the values of mortality due to CVD I used, first, Linear Regression and, second, Decision Tree Regressor. The input variables were: the percentage of population with raised level of cholesterol and the percentage of population with raised blood pressure in each country. The target variable was the percentage of deaths due to CVD out of all global deaths (numeric value) in each country. The quality of regression analyses was estimated by Root Mean Square Error (RMSE).

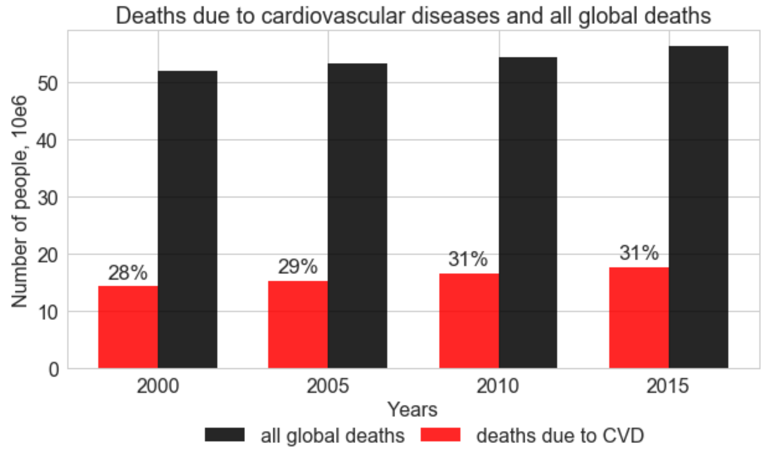
Part II

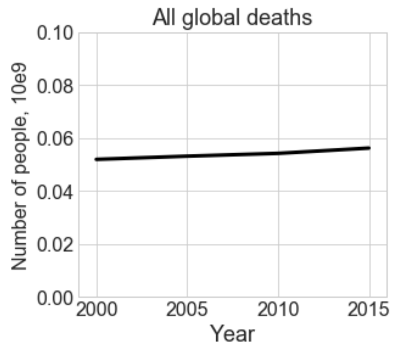
3. To predict the presence of CVD for a given patient I used 3 classification analyses: Decision Tree Classifier, Stochastic Gradient Descent Classifier (SGDClassifier), Support Vector Classification (SVC). The input variables were 14 attributes of CVD for each patient. The target variable was the diagnosis of heart disease (healthy or sick - categorical variable) for each patient. I compared the analyses quality by means of accuracy score and confusion matrices.

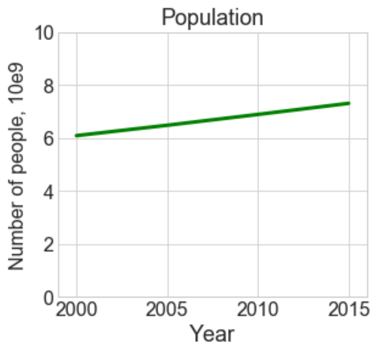
Findings - Part I. Global mortality, deaths due to cardiovascular diseases and risk factors

Findings – part I – 1

- The total number of humans currently living in the world increased from 6.1 billion people in 2000 to 7.3 billion people in 2015
- The global number of deaths in the world increased from 52.0 million in 2000 to 56.2 million in 2015
- The number of deaths due to CVD in the world increased from 14.4 million in 2000 to 17.6 million in 2015
- The mortality due to CVD in 2015 represented 31% of all global deaths (in 2000 it was 28%). It means that approximately each third die in the world is due to CVD

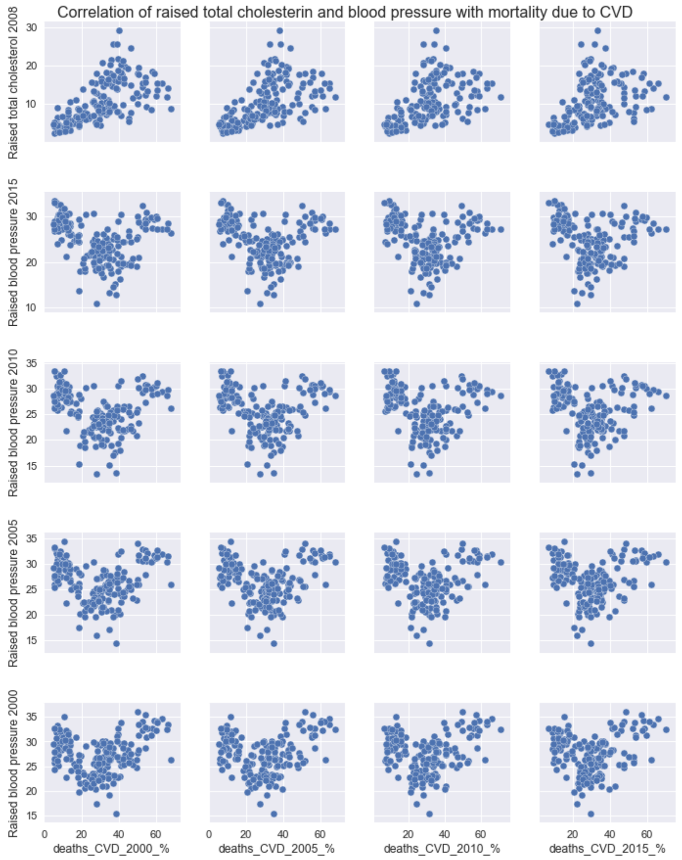






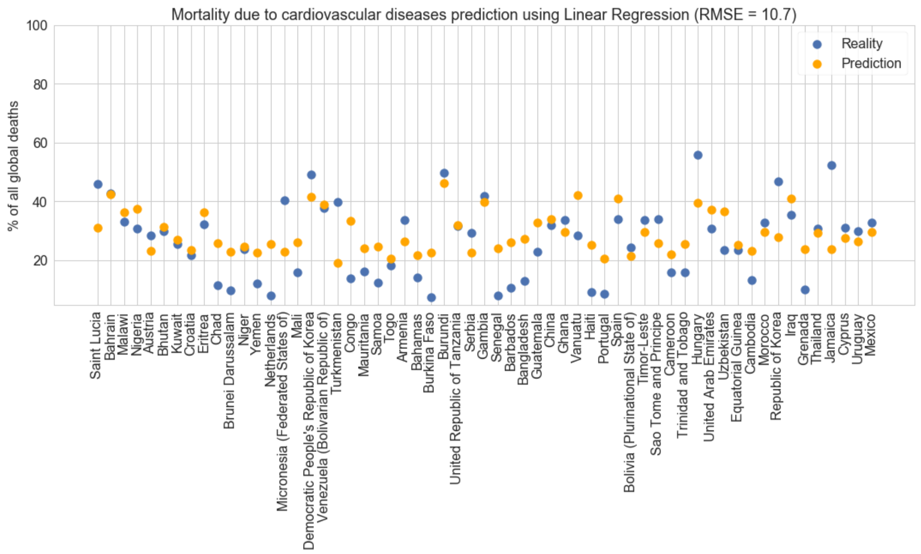
Findings – part I – 2

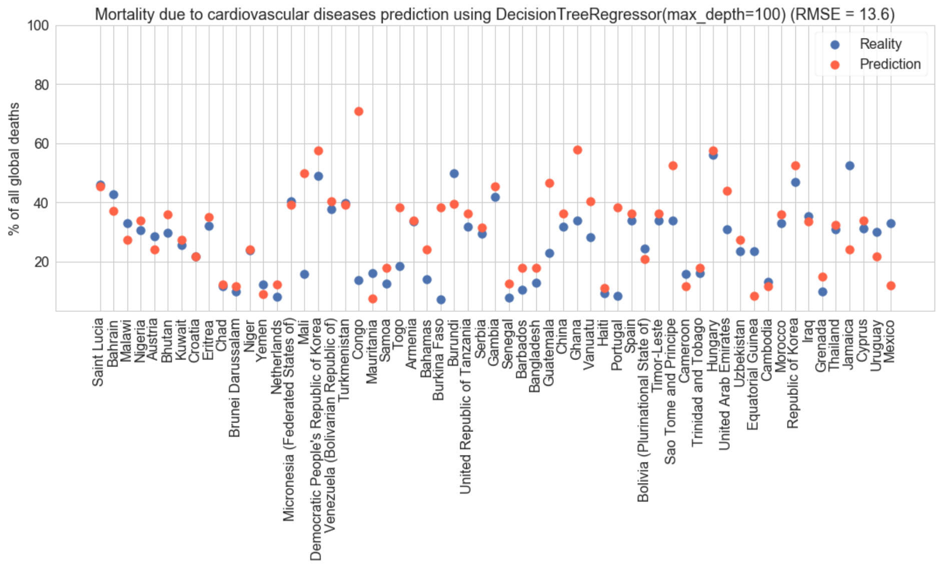
- The mortality due to CVD in 2010 is correlated with the percentage of people with raised total cholesterol (coeff = 0.54)
- The mortality due to CVD is slightly correlated with the percentage of people with raised blood pressure (coeff 2000 = 0.12, coeff 2005 = -0.02, coeff 2010 = -0.08, coeff 2015 = -0.09)
- The graphs of the mortality due to CVD dependence from the percentage of people with raised blood pressure in different years have a similar form



Findings – part I – 3

The prediction of the values of mortality due to CVD using regression analyses was not appropriate (neither by means of Linear Regression and nor by means of Decision Tree Regressor)

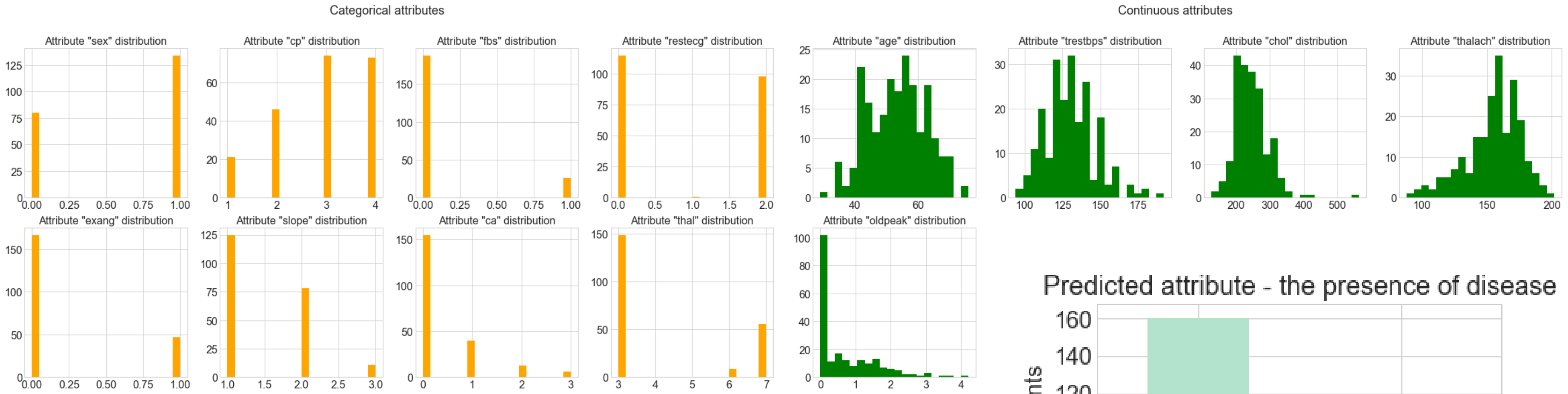


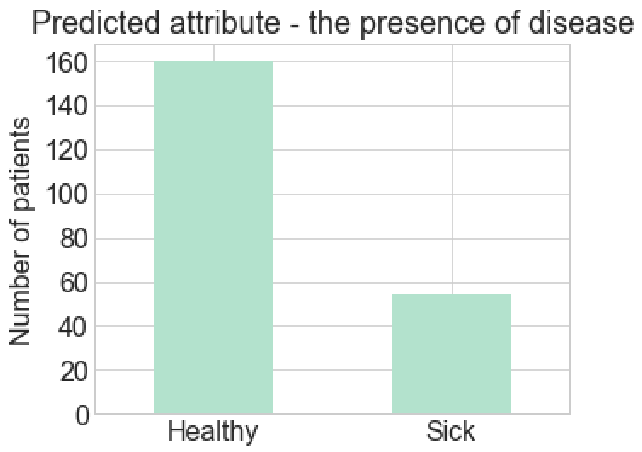


RMSE for Linear Regression = 10.7, RMSE for Decision Tree Regressor = 13.7, which both are commensurable with the mean of test data of target variable (26.7)

Findings - Part II. Cardiovascular disease presence prediction for a given patient

Findings – part II – 1

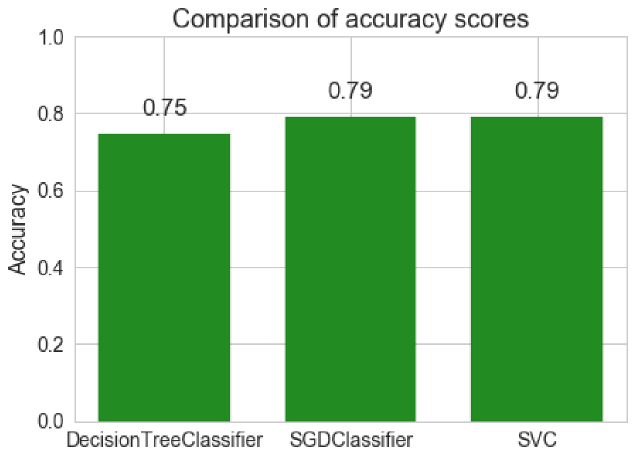


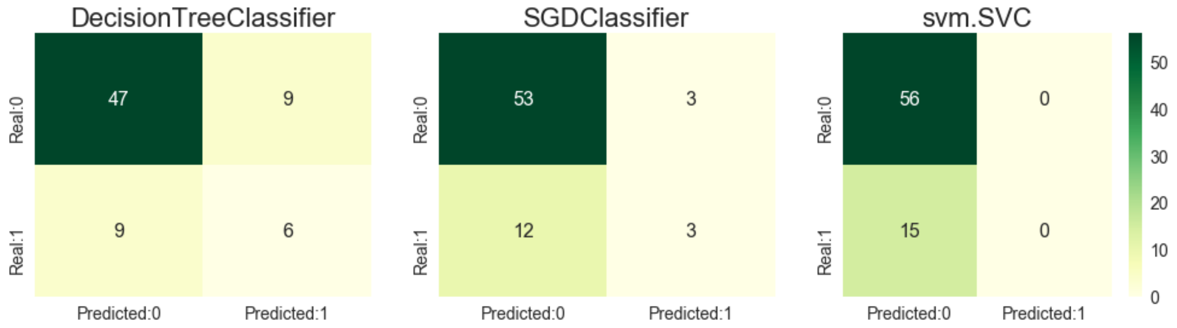


- In this dataset there are 9 categorical attributes and 5 continuous attributes
- The predicted attribute is categorical. There are 160 patients with no heart disease and 54 sick patients

Findings – part II – 2

- The prediction of CVD presence was appropriate
- The classification by SVC or by SGDClassifier have higher accuracy score than classification by Decision Tree Classifier
- The Decision Tree Clasiffier made more errors type I (False Positives: 9), than others (SGDClassifier has 3 False Positives, SVC has 0 False Positives)
- The Decision Tree Clasiffier made less errors type II (False Negatives: 9), than others (SGDClassifier has 12 False Negatives, SVC has 15 False Negatives)





DecisionTreeClassifier

	Predicted:0	Predicted:1
Real:0	47	9
Real:1	9	6

SGDClassifier

	Predicted:0	Predicted:1
Real:0	53	3
Real:1	12	3

svm.SVC

	Predicted:0	Predicted:1
Real:0	56	0
Real:1	15	0

For the model predicting a disease a False Negative (ignoring the probability of disease when there actually is one) is more dangerous than a False Positive. So, the Decision Tree Classifier is preferred in this case.

Limitations

Part I

- I chose only 2 “intermediate” risk factors, while there are a lot of behavioral risk factors of heart disease and stroke, like unhealthy diet, physical inactivity, tobacco use and harmful use of alcohol, which could be more important.
- I used the data for people with raised cholesterol in 2008, because I have not found other years, but it could be different in 2010.
- From all data for risk factors I chose the overall data for both sexes, but the values could differ in men and women

Part II

- In this dataset after cleaning there was data only for 214 patients. More samples could give better results for prognosis.
- The prediction is very rough – only yes or no, it does not allow to identify the concrete type of cardiovascular disease that patient has.

Conclusions

Part 1. Global mortality, deaths due to cardiovascular diseases and risk factors.

1. The mortality due to cardiovascular diseases is correlated with high level of cholesterol and is slightly correlated with raised blood pressure

2. The prediction of the mortality due to cardiovascular diseases is not enough appropriate, if we know only the percentage of population with mentioned above risk factors in each country and related mortality in some countries

Part 2. Cardiovascular disease presence prediction for a given patient.

3. The predictions of the presence of cardiovascular disease using the Decision Tree Classifier, SGDClassifier and SVC were feasible. The Decision Tree Classifier is preferred in this case, because it made less errors type II

Acknowledgements

I would like to thank

- the websites http://www.who.int and https://www.kaggle.com for the data I used and the information about cardiovascular diseases

- the courses Mathematics and Python for Data Science and Supervised learning (MIPT and Yandex) from Coursera.org for my first meeting with Python and its libraries, which helped me in this project realization

- the websites https://stackoverflow.com, https://habr.com, and certainly Google.com and Wikipedia.org, where I found a lot of answers for my questions

References

1. http://www.who.int
2. https://www.kaggle.com

FinalProject. Cardiovascular diseases: risk factors and attempts of prediction.

In [1]:

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import seaborn as sns
```

Part I. Global mortality, deaths due to cardiovascular diseases and risk factors.

Summary tables of mortality estimates by cause, age and sex, by country, 2000–2015

http://www.who.int/healthinfo/global_burden_disease/estimates/en/

Data Preparation and Cleaning - Part I - 1

In [2]:

```
Causes_of_death2000 =
pd.read_excel('C:\\Users\\Olga\\edx_Python_for_Data_science\\FinalProject\\
datasets\\GHE2015_Deaths-2000-country.xls', sheet_name=1)
```

In [3]:

```
Causes_of_death2000.head(20)
```

Out[3]:

			World Health Organization	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unna
NaN	**NaN**	**NaN**	Department of Information, Evidence and Research	NaN	NaN	NaN	NaN	NaN
		NaN	March 2017	NaN	NaN	NaN	NaN	NaN
		NaN	NaN	NaN	NaN	NaN	NaN	NaN
		NaN	Estimated deaths ('000) by cause, sex	NaN	NaN	NaN	NaN	NaN
		NaN	and WHO Member State	NaN	NaN	NaN	NaN	NaN

		NaN	Member State (1), 2000	NaN	NaN	NaN	NaN	NaN
			World Health Organization	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unna
Sex	**GHE code**	**NaN**	GHE cause	NaN	NaN	Member State\n(See Notes for explanation of co...	Afghanistan	Albar
NaN	**NaN**	**NaN**	NaN	NaN	NaN	ISO-3 Code	AFG	ALB
		NaN	NaN	NaN	NaN	NaN	4	3
Persons	**NaN**	**NaN**	Population ('000) (2)	NaN	NaN	NaN	19702	3122
	0	**NaN**	All Causes	NaN	NaN	NaN	249.757	21.21
	10	**I.**	Communicable, maternal, perinatal and nutritio...	NaN	NaN	NaN	148.887	1.573
	20	**NaN**	A.	Infectious and parasitic diseases	NaN	NaN	69.0322	0.266
	30	**NaN**	NaN	1.	Tuberculosis	NaN	13.5765	0.025
	40	**NaN**	NaN	2.	STDs excluding HIV	NaN	0.62938	0.019
	50	**NaN**	NaN	NaN	a.	Syphilis	0.625165	0.019
	60	**NaN**	NaN	NaN	b.	Chlamydia	0.00059323	1.177
	70	**NaN**	NaN	NaN	c.	Gonorrhoea	0.00231667	0.000
	80	**NaN**	NaN	NaN	d.	Trichomoniasis	.	.
	85	**NaN**	NaN	NaN	e.	Genital herpes	.	.
	90	**NaN**	NaN	NaN	f.	Other STDs	0.00130523	5.851

20 rows × 187 columns

In [4]:

```
Causes_of_death2000.shape
```

Out[4]:

```
(627, 187)
```

In [5]:

```python
print(Causes_of_death2000['World Health Organization'].unique())
print(Causes_of_death2000['Unnamed: 1'].unique())
print(Causes_of_death2000['Unnamed: 2'].unique())
```

```
['Department of Information, Evidence and Research' 'March 2017' nan
 "Estimated deaths ('000) by cause, sex" 'and WHO Member State (1), 2000'
 'GHE cause' "Population ('000) (2)" 'All Causes'
 'Communicable, maternal, perinatal and nutritional conditions' 'A.' 'B.'
 'C.' 'D.' 'E.' 'Noncommunicable diseases' 'F.' 'G.' 'H.' 'I.' 'J.' 'K.'
 'L.' 'M.' 'N.' 'O.' 'P.' 'Injuries']
[nan 'Infectious and parasitic diseases' '1.' '2.' '3.' '4.' '5.' '6.'
 '7.' '8.' '9.' '10.' '11.' '12.' 'Respiratory Infectious '
 'Maternal conditions' 'Neonatal conditions' 'Nutritional deficiencies'
 'Malignant neoplasms' '13.' '14.' '15.' '16.' '17.' '18.' '19.' '20.'
 '21.' '22.' '23.' '24.' 'Other neoplasms' 'Diabetes mellitus'
 'Endocrine, blood, immune disorders' 'Mental and substance use disorders'
 'Neurological conditions' 'Sense organ diseases'
 'Cardiovascular diseases' 'Respiratory diseases' 'Digestive diseases'
 'Genitourinary diseases' 'Skin diseases' 'Musculoskeletal diseases'
 'Congenital anomalies' 'Oral conditions' 'Sudden infant death syndrome'
 'Unintentional injuries' 'Intentional injuries']
[nan 'Tuberculosis' 'STDs excluding HIV' 'a.' 'b.' 'c.' 'd.' 'e.' 'f.'
 'HIV/AIDS' 'Diarrhoeal diseases' 'Childhood-cluster diseases'
 'Meningitis' 'Encephalitis' 'Hepatitis' 'Parasitic and vector diseases'
 'g.' 'h.' 'i.' 'j.' 'k.' 'l.' 'm.' 'Intestinal nematode infections'
 'Leprosy' 'Other infectious diseases' 'Lower respiratory infections'
 'Upper respiratory infections' 'Otitis media'
 'Preterm birth complications' 'Birth asphyxia and birth trauma'
 'Neonatal sepsis and infections' 'Other neonatal conditions'
 'Protein-energy malnutrition' 'Iodine deficiency' 'Vitamin A deficiency'
 'Iron-deficiency anaemia' 'Other nutritional deficiencies'
 'Mouth and oropharynx cancers' 'Oesophagus cancer' 'Stomach cancer'
 'Colon and rectum cancers' 'Liver cancer' 'Pancreas cancer'
 'Trachea, bronchus, lung cancers' 'Melanoma and other skin cancers'
 'Breast cancer' 'Cervix uteri cancer' 'Corpus uteri cancer'
 'Ovary cancer' 'Prostate cancer' 'Testicular cancer' 'Kidney cancer'
 'Bladder cancer' 'Brain and nervous system cancers'
 'Gallbladder and biliary tract cancer' 'Larynx cancer' 'Thyroid cancer'
 'Mesothelioma' 'Lymphomas, multiple myeloma' 'Leukaemia'
 'Other malignant neoplasms' 'Thalassaemias'
 'Sickle cell disorders and trait'
 'Other haemoglobinopathies and haemolytic anaemias'
 'Other endocrine, blood and immune disorders' 'Depressive disorders'
 'Bipolar disorder' 'Schizophrenia' 'Alcohol use disorders'
 'Drug use disorders' 'Anxiety disorders' 'Eating disorders'
 'Autism and Asperger syndrome' 'Childhood behavioural disorders'
 'Idiopathic intellectual disability'
 'Other mental and behavioural disorders'
 'Alzheimer disease and other dementias' 'Parkinson disease' 'Epilepsy'
 'Multiple sclerosis' 'Migraine' 'Non-migraine headache'
 'Other neurological conditions' 'Rheumatic heart disease'
 'Hypertensive heart disease' 'Ischaemic heart disease' 'Stroke'
 'Cardiomyopathy, myocarditis, endocarditis' 'Other circulatory diseases'
 'Chronic obstructive pulmonary disease' 'Asthma'
 'Other respiratory diseases' 'Peptic ulcer disease'
 'Cirrhosis of the liver' 'Appendicitis' 'Gastritis and duodenitis'
 'Paralytic ileus and intestinal obstruction' 'Inflammatory bowel disease'
 'Gallbladder and biliary diseases' 'Pancreatitis'
 'Other digestive diseases' 'Kidney diseases'
 'Benign prostatic hyperplasia' 'Urolithiasis' 'Other urinary diseases'
 'Infertility' 'Gynecological diseases' 'Rheumatoid arthritis'
 'Osteoarthritis' 'Gout' 'Back and neck pain'
 'Other musculoskeletal disorders' 'Neural tube defects'
 'Cleft lip and cleft palate' 'Down syndrome' 'Congenital heart anomalies'
 'Other chromosomal anomalies' 'Other congenital anomalies' 'Road injury'
 'Poisonings' 'Falls' 'Fire, heat and hot substances' 'Drowning'
 'Exposure to mechanical forces' 'Natural disasters'
 'Other unintentional injuries' 'Self-harm' 'Interpersonal violence'
 'Collective violence and legal intervention']
```

In [6]:

```
all_causes = Causes_of_death2000.iloc[9].values[4:]
all_causes
```

Out[6]:

```
array([249.75736333, 21.2187077709, 146.427147145, 290.077680216,
       0.522158227599, 282.77542193, 24.9690091875, 128.501707317,
       76.6868652117, 55.0923000742, 1.66727006096, 1.89647043305,
       917.350454481, 2.82909086863, 148.079020096, 104.618856595,
       1.5125812757, 90.3156857051, 5.08251776403, 76.2088283493,
       31.1378358794, 27.0103536542, 1034.84372847, 0.947324959773,
       118.184430114, 189.674130754, 98.7781013816, 115.615997775,
       230.570335084, 217.609404902, 2.55113861327, 69.3402256634,
       146.207840453, 79.3736470304, 8053.5478388, 208.640683853,
       5.12327210679, 41.6399594309, 15.7363862497, 266.91075308,
       50.2453391137, 77.1057900405, 6.56568561346, 108.880224869,
       205.194741623, 710.000339861, 56.8587455814, 7.31146127888,
       46.5554808405, 66.0047438094, 442.0800418, 38.4728680619,
       7.53698062212, 57.8758074986, 18.1558166703, 979.204344657,
       4.82869961393, 49.1389635208, 529.830441627, 13.6595146797,
       14.3693715111, 48.3699261437, 838.621009122, 199.784804514,
       105.169449077, 0.853030058756, 72.8678206826, 130.548946999,
       19.4568565433, 5.50393926094, 87.8697470729, 31.4384784808,
       135.191159028, 1.80050263175, 9184.59433826, 1542.96357921,
       329.420615266, 123.606085987, 31.367321373, 37.6344887509,
       559.648866865, 18.7292827145, 960.843727487, 19.7727808166,
       161.705878053, 378.959165566, 0.621938922185, 5.1794427238,
       37.1145594459, 53.9865764479, 33.0322092052, 18.9388017361,
       28.0534808624, 46.7314021791, 25.1582277383, 40.2865167832,
       3.59441992714, 169.330906015, 204.773950487, 106.476101959,
       1.3213533221, 189.092738931, 2.99013150101, 26.1267312023,
       7.99959289093, 454.655795369, 0.685255573678, 18.5217324698,
       6.0511647281, 181.60071101, 296.813877244, 426.561757744,
       18.9227357942, 203.486672282, 140.166995729, 26.6004051454,
       25.3521174562, 194.011215585, 2216.06831053, 43.8724773687,
       7.65638828831, 1203.21259895, 13.833781847, 49.3316882298,
       28.7412825676, 140.944812926, 473.150036967, 374.181474704,
       105.687780974, 1.25162826298, 244.755087062, 47.614503242,
       256.64295246, 2301.56004333, 129.486922775, 1.15831463315,
       0.776410061175, 0.973374787872, 1.22806835544, 78.3932254136,
       109.324031325, 109.290585496, 0.602351220512, 101.701804391,
       18.4762160179, 52.6151743895, 18.549280401, 2.41599624106,
       135.253574183, 493.497218865, 109.391570243, 360.528702989,
       125.171767422, 288.284950847, 3.63023615255, 14.9514821343,
       93.2926065802, 62.3336111444, 63.6865101874, 46.3840121665,
       421.001097693, 16.9544675541, 8.62889114026, 60.0021554059,
       0.572637112881, 10.1940918938, 54.344378487, 396.467269826,
       33.164331932, 393.189653288, 767.358316538, 5.78670773812,
       607.55332179, 479.940090639, 2398.82839913, 30.4260193733,
       162.917789413, 1.01592860041, 116.290539445, 436.796719792,
       157.800188045, 193.233969009, 192.51438584], dtype=object)
```

Extracting cause of death - **Cardiovascular diseases (CVD)**

In [7]:

```
def clean_data(data):
    qq = data.copy()
    countries = data.iloc[5].values[4:]
    population = data.iloc[8].values[4:]
    all_causes = data.iloc[9].values[4:]
    qq.dropna()
    mask1 = qq['Unnamed: 1'].str.contains('Cardiovascular diseases', na = F
alse)
    qqc = qq[mask1]
    cause_cardio = qqc.iloc[0].values[4:]
    d = {'countries': countries, 'all causes': all_causes, 'cause cardio': c
ause_cardio, 'population': population}
    return pd.DataFrame(d)
```

In [8]:

```
data2000 = clean_data(Causes_of_death2000)
```

In [9]:

```
Causes_of_death2005 =
pd.read_excel('C:\\Users\\Olga\\edx_Python_for_Data_science\\FinalProject\\
datasets\\GHE2015_Deaths-2005-country.xls', sheet_name=1)
Causes_of_death2010 =
pd.read_excel('C:\\Users\\Olga\\edx_Python_for_Data_science\\FinalProject\\
datasets\\GHE2015_Deaths-2010-country.xls', sheet_name=1)
Causes_of_death2015 =
pd.read_excel('C:\\Users\\Olga\\edx_Python_for_Data_science\\FinalProject\\
datasets\\GHE2015_Deaths-2015-country.xls', sheet_name=1)
```

In [10]:

```
print(Causes_of_death2005.shape,
      Causes_of_death2010.shape,
      Causes_of_death2015.shape)
```

```
(627, 187) (627, 187) (627, 187)
```

In [11]:

```
data2005 = clean_data(Causes_of_death2005)
data2010 = clean_data(Causes_of_death2010)
data2015 = clean_data(Causes_of_death2015)
```

In [12]:

```
data2000.rename(columns={'all causes': 'all2000', 'cause cardio':
'cardio2000', 'population': 'population2000'}, inplace = True)
data2005.rename(columns={'all causes': 'all2005', 'cause cardio':
'cardio2005', 'population': 'population2005'}, inplace = True)
```

```
data2010.rename(columns={'all causes': 'all2010', 'cause cardio':
'cardio2010', 'population': 'population2010'}, inplace = True)
data2015.rename(columns={'all causes': 'all2015', 'cause cardio':
'cardio2015', 'population': 'population2015'}, inplace = True)
```

In [13]:

```
data2005.loc[data2005['countries'] == 'Czech Republic']
data2005['countries'][43] = 'Czechia'
data2005.countries.values
```

Out[13]:

```
array(['Afghanistan', 'Albania', 'Algeria', 'Angola',
       'Antigua and Barbuda', 'Argentina', 'Armenia', 'Australia',
       'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh',
       'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
       'Bolivia (Plurinational State of)', 'Bosnia and Herzegovina',
       'Botswana', 'Brazil', 'Brunei Darussalam', 'Bulgaria',
       'Burkina Faso', 'Burundi', 'Cambodia', 'Cameroon', 'Canada',
       'Cape Verde', 'Central African Republic', 'Chad', 'Chile', 'China',
       'Colombia', 'Comoros', 'Congo', 'Costa Rica', "Côte d'Ivoire",
       'Croatia', 'Cuba', 'Cyprus', 'Czechia',
       "Democratic People's Republic of Korea",
       'Democratic Republic of the Congo', 'Denmark', 'Djibouti',
       'Dominican Republic', 'Ecuador', 'Egypt', 'El Salvador',
       'Equatorial Guinea', 'Eritrea', 'Estonia', 'Ethiopia', 'Fiji',
       'Finland', 'France', 'Gabon', 'Gambia', 'Georgia', 'Germany',
       'Ghana', 'Greece', 'Grenada', 'Guatemala', 'Guinea',
       'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras', 'Hungary',
       'Iceland', 'India', 'Indonesia', 'Iran (Islamic Republic of)',
       'Iraq', 'Ireland', 'Israel', 'Italy', 'Jamaica', 'Japan', 'Jordan',
       'Kazakhstan', 'Kenya', 'Kiribati', 'Kuwait', 'Kyrgyzstan',
       "Lao People's Democratic Republic", 'Latvia', 'Lebanon', 'Lesotho',
       'Liberia', 'Libyan Arab Jamahiriya', 'Lithuania', 'Luxembourg',
       'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali', 'Malta',
       'Mauritania', 'Mauritius', 'Mexico',
       'Micronesia (Federated States of)', 'Mongolia', 'Montenegro',
       'Morocco', 'Mozambique', 'Myanmar', 'Namibia', 'Nepal',
       'Netherlands', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria',
       'Norway', 'Oman', 'Pakistan', 'Panama', 'Papua New Guinea',
       'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal', 'Qatar',
       'Republic of Korea', 'Republic of Moldova', 'Romania',
       'Russian Federation', 'Rwanda', 'Saint Lucia',
       'Saint Vincent and the Grenadines', 'Samoa',
       'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
       'Seychelles', 'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia',
       'Solomon Islands', 'Somalia', 'South Africa', 'South Sudan',
       'Spain', 'Sri Lanka', 'Sudan', 'Suriname', 'Swaziland', 'Sweden',
       'Switzerland', 'Syrian Arab Republic', 'Tajikistan', 'Thailand',
       'The former Yugoslav Republic of Macedonia', 'Timor-Leste', 'Togo',
       'Tonga', 'Trinidad and Tobago', 'Tunisia', 'Turkey',
       'Turkmenistan', 'Uganda', 'Ukraine', 'United Arab Emirates',
       'United Kingdom', 'United Republic of Tanzania',
       'United States of America', 'Uruguay', 'Uzbekistan', 'Vanuatu',
       'Venezuela (Bolivarian Republic of)', 'Viet Nam', 'Yemen',
       'Zambia', 'Zimbabwe'], dtype=object)
```

In [14]:

```
data = data2000.merge(data2005, on='countries', how='inner')
data = data.merge(data2010, on='countries', how='inner')
data = data.merge(data2015, on='countries', how='inner')
data.head()
```

Out[14]:

	all2000	cardio2000	countries	population2000	all2005	cardio2005	population2005	a
0	249.757	33.5378	Afghanistan	19702	253.554	41.6761	24400	26
1	21.2187	10.7373	Albania	3122	22.7499	12.5973	3082	20.
2	146.427	48.3884	Algeria	31184	158.492	53.6709	33268	17
3	290.078	20.8035	Angola	15059	314.963	24.4419	17913	33
4	0.522158	0.180695	Antigua and Barbuda	78	0.53936	0.19641	83	0.5

Findings - Part I - 1

In [15]:

```
data_sum = data.copy()
data_sum = data_sum.sum(axis = 0)
del data_sum['countries']
data_sum
```

Out[15]:

```
all2000            51964.8
cardio2000         14374.9
population2000     6093454
all2005            53176.9
cardio2005           15338
population2005     6485002
all2010            54233.7
cardio2010         16570.2
population2010     6893891
all2015              56229
cardio2015         17630.9
population2015     7312631
dtype: object
```

In [16]:

```
data_sum.values.reshape((4,3))
```

Out[16]:

```
array([[51964.7534413778, 14374.850582479909, 6093454],
       [53176.86840660747, 15337.987182532857, 6485002],
       [54233.74232256859, 16570.217707141364, 6893891],
       [56228.95059410143, 17630.939554373985, 7312631]], dtype=object)
```

In [17]:

```
data_sum_years = pd.DataFrame(data = data_sum.values.reshape((4,3)))
data_sum_years.rename(columns = {0: 'deaths_all_causes', 1: 'deaths_CVD', 2:
'population'}, inplace = True)
```

```
data_sum_years['years'] = [2000, 2005, 2010, 2015]
data_sum_years.rename(index = data_sum_years['years'], inplace = True)
del data_sum_years['years']
data_sum_years
```

Out[17]:

	deaths_all_causes	deaths_CVD	population
2000	51964.8	14374.9	6093454
2005	53176.9	15338	6485002
2010	54233.7	16570.2	6893891
2015	56229	17630.9	7312631

In [18]:

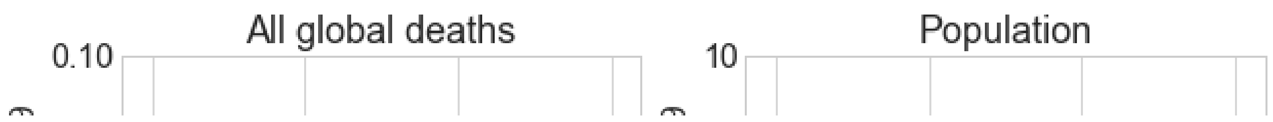
```
data_sum_years['deaths_CVD_%'] =
data_sum_years['deaths_CVD']/data_sum_years['deaths_all_causes'] *100
data_sum_years
```

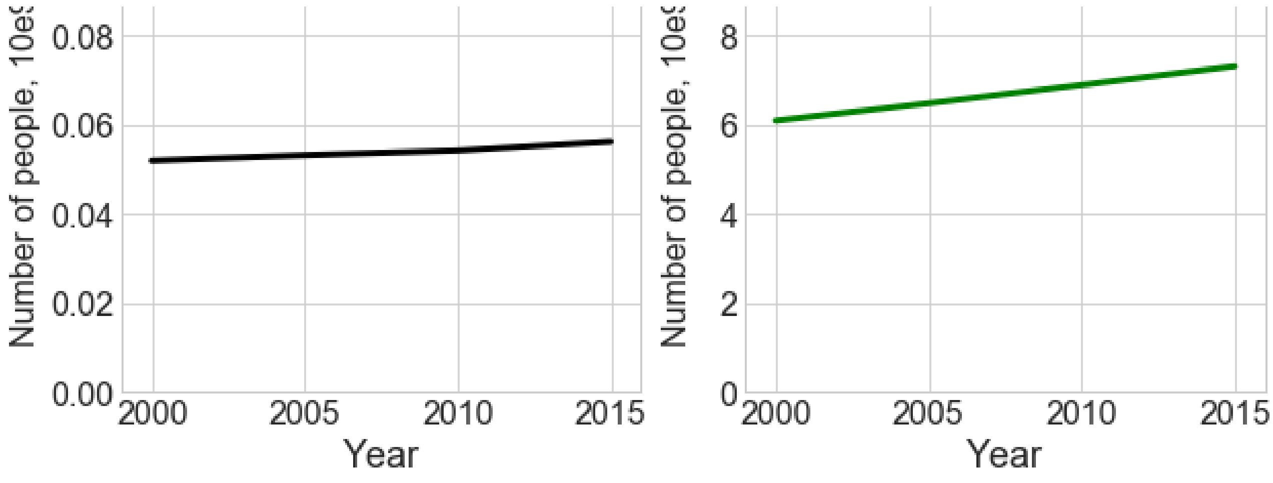
Out[18]:

	deaths_all_causes	deaths_CVD	population	deaths_CVD_%
2000	51964.8	14374.9	6093454	27.6627
2005	53176.9	15338	6485002	28.8433
2010	54233.7	16570.2	6893891	30.5533
2015	56229	17630.9	7312631	31.3556

In [22]:

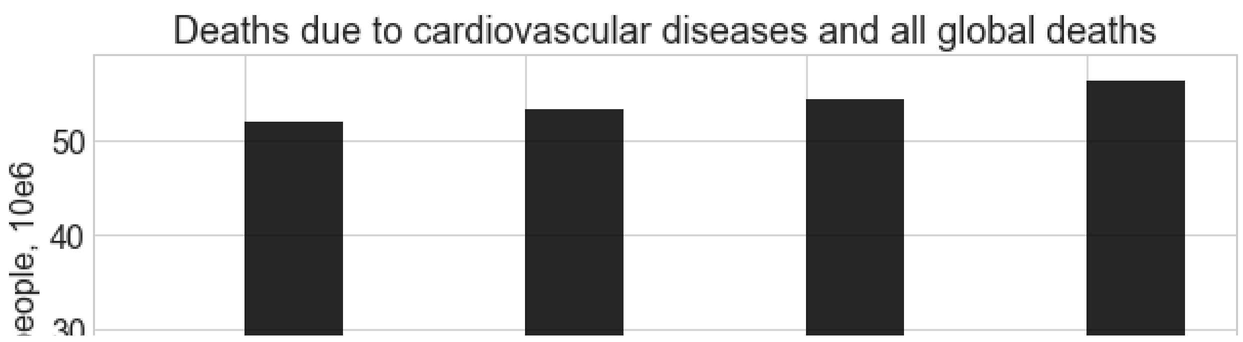
```
sns.set_style("whitegrid")
fig0 = plt.figure(figsize = (10, 4))

plt.subplot(1, 2, 1)
(data_sum_years['deaths_all_causes']/1000000).plot(lw=3, color = 'black')
plt.axis([1999, 2016, 0, 0.1])
plt.xlabel('Year', size = 18)
plt.ylabel('Number of people, 10e9', size = 16)
plt.title('All global deaths', fontsize = 18)
plt.tick_params(labelsize = 16)

plt.subplot(1, 2, 2)
(data_sum_years['population']/1000000).plot(lw=3, color = 'green')
plt.axis([1999, 2016, 0, 10])
plt.xlabel('Year', size = 18)
plt.ylabel('Number of people, 10e9', size = 16)
plt.title('Population', fontsize = 18)

plt.tick_params(labelsize = 16)
fig0.savefig('fig0', bbox_inches = 'tight')
```

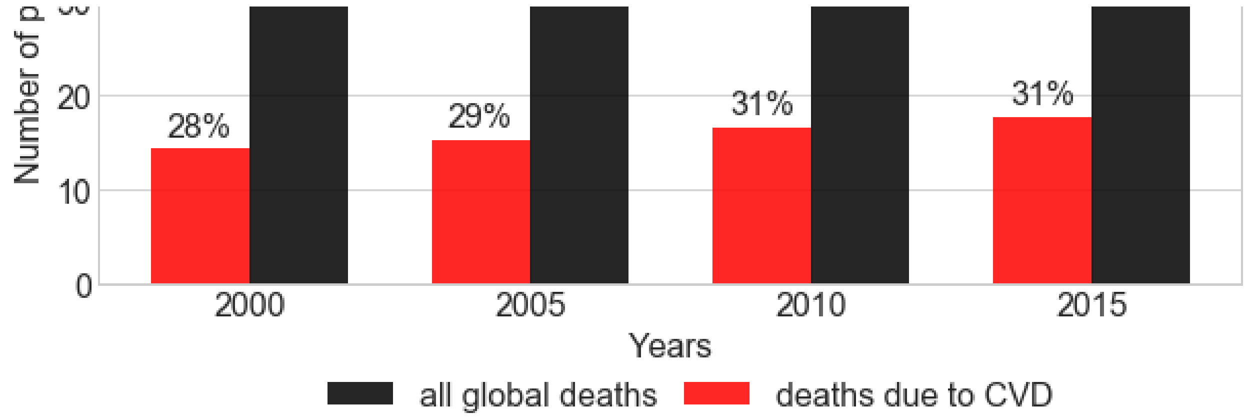




In [23]:

```
sns.set_style("whitegrid")
fig1, ax = plt.subplots(figsize = (10,5))

index = np.arange(4)
bar_width = 0.35


rects1 = ax.bar(index, data_sum_years['deaths_CVD']/1000, bar_width,
                alpha = 0.85, color='r',
                label='deaths due to CVD')

rects2 = ax.bar(index + bar_width, data_sum_years['deaths_all_causes']/1000
, bar_width,
                alpha = 0.85, color='black',
                label='all global deaths')

for i, rect1 in enumerate(rects1):
    height = str(round(data_sum_years['deaths_CVD_%'].values[i])) + '%'
    ax.text(rect1.get_x() + rect1.get_width()/2., 1.05*rect1.get_height(),
'%s' % height, ha='center', va='bottom', fontsize =16)

ax.set_xlabel('Years', fontsize =16)
ax.set_ylabel('Number of people, 10e6', fontsize =16)
ax.set_title('Deaths due to cardiovascular diseases and all global deaths'
, fontsize =18)
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(data_sum_years.index)
ax.legend()
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=2,
mode="expand", borderaxespad=0., fontsize =16)
plt.tick_params(labelsize = 16)

fig1.savefig('fig1', bbox_inches = 'tight')
```





In [23]:

```
data_percent_cardio = data.copy()
data_percent_cardio['deaths_CVD_2000_%'] = data_percent_cardio['cardio2000'
]/data_percent_cardio['all2000']*100
data_percent_cardio['deaths_CVD_2005_%'] = data_percent_cardio['cardio2005'
]/data_percent_cardio['all2005']*100
data_percent_cardio['deaths_CVD_2010_%'] = data_percent_cardio['cardio2010'
]/data_percent_cardio['all2010']*100
data_percent_cardio['deaths_CVD_2015_%'] = data_percent_cardio['cardio2015'
]/data_percent_cardio['all2015']*100
del data_percent_cardio['cardio2000']
del data_percent_cardio['cardio2005']
del data_percent_cardio['cardio2010']
del data_percent_cardio['cardio2015']
del data_percent_cardio['all2000']
del data_percent_cardio['all2005']
del data_percent_cardio['all2010']
del data_percent_cardio['all2015']
```

In [24]:

```
data_percent_cardio.head()
```

Out[24]:

	countries	population2000	population2005	population2010	population2015	deaths_CVD_
0	Afghanistan	19702	24400	27962	32527	13.4281
1	Albania	3122	3082	2902	2897	50.6032
2	Algeria	31184	33268	36036	39667	33.0461
3	Angola	15059	17913	21220	25022	7.17169
4	Antigua and Barbuda	78	83	87	92	34.6055

Cholesterol

http://apps.who.int/gho/data/node.main.A887?lang=en Raised total cholesterol (≥6.2 mmol/L). Data by country

Data Preparation and Cleaning - Part I - 2

In [25]:

```
Cholesterol = pd.read_csv('C:\\Users\\Olga\\edx_Python_for_Data_science\\Fi
nalProject\\datasets\\heart\\Cholesterol6.csv', sep=',')
Cholesterol.head()
```

Out[25]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Raised total cholesterol (&amp;#8805, 6.2 mmol/L) (age-standardized estimate)	Raised total cholesterol (&amp;#8805, 6.2 mmol/L) (age-standardized estimate).1	Raised total cholesterol (&amp;#8805, 6.2 mmol/L) (age-standardized estimate).2	Raised tot cholester (&amp;#880 6.2 mmol/ (cruc estimat
0	Country	Year	Age Group	Both sexes	Male	Female	Both sexes
1	Afghanistan	2008	25+ years	4.0 [1.7-8.1]	3.5 [1.0-8.3]	4.5 [1.2-11.6]	3.6 [1.6-7.1]
2	Albania	2008	25+ years	12.0 [5.9-21.1]	11.3 [4.2-23.5]	12.6 [4.0-27.4]	12.4 [6.0-21.
3	Algeria	2008	25+ years	8.6 [4.7-14.5]	7.6 [3.3-14.4]	9.6 [3.7-20.0]	8.2 [4.6-13.5
4	Andorra	2008	25+ years	25.5 [14.2-39.9]	26.7 [11.3-48.5]	24.1 [9.3-43.3]	27.0 [15.0-42.3]

In [26]:

```
Cholesterol.shape
```

Out[26]:

```
(194, 9)
```

In [27]:

```
del Cholesterol['Raised total cholesterol (&amp;#8805, 6.2 mmol/L) (crude e
stimate)']
Cholesterol.rename(columns=Cholesterol.iloc[0], inplace = True)
Cholesterol = Cholesterol[1:]
```

In [28]:

```
Cholesterol2008 = Cholesterol[['Country', ' Both sexes']]
Cholesterol2008.rename(columns = {' Both sexes': 'Raised total cholesterol
2008', 'Country': 'countries'}, inplace = True)
Cholesterol2008.head()
```

```
C:\Users\Olga\AppData\Local\conda\conda\envs\py36\lib\site-
packages\pandas\core\frame.py:3027: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy
  return super(DataFrame, self).rename(**kwargs)
```

Out[28]:

	countries	Raised total cholesterol 2008
1	Afghanistan	4.0 [1.7-8.1]
2	Albania	12.0 [5.9-21.1]
3	Algeria	8.6 [4.7-14.5]
4	Andorra	25.5 [14.2-39.9]
5	Angola	6.6 [2.4-14.6]

In [29]:

```
import re
```

In [30]:

```
def delete_brackets(text):
    return re.sub(r'\[[^()]*\]', '', text)
```

In [31]:

```
Cholesterol2008 = Cholesterol2008.applymap(lambda x: delete_brackets(x))

Cholesterol2008.head()
```

Out[31]:

	countries	Raised total cholesterol 2008
1	Afghanistan	4.0
2	Albania	12.0
3	Algeria	8.6
4	Andorra	25.5
5	Angola	6.6

In [32]:

```
Cholesterol2008.loc[Cholesterol2008['Raised total cholesterol 2008'] == '..
.']
Cholesterol2008.drop([112, 126, 148, 179], inplace = True)
```

In [33]:

```
Cholesterol2008['Raised total cholesterol 2008'] = pd.to_numeric(Cholestero
l2008['Raised total cholesterol 2008'])
```

In [34]:

```
data = data_percent_cardio.merge(Cholesterol2008, on='countries', how='inne
r')
data.head()
```

Out[34]:

	countries	population2000	population2005	population2010	population2015	deaths_CVD_
0	Afghanistan	19702	24400	27962	32527	13.4281
1	Albania	3122	3082	2902	2897	50.6032
2	Algeria	31184	33268	36036	39667	33.0461
3	Angola	15059	17913	21220	25022	7.17169
4	Antigua and Barbuda	78	83	87	92	34.6055

In [35]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 178 entries, 0 to 177
Data columns (total 10 columns):
countries                        178 non-null object
population2000                   178 non-null object
population2005                   178 non-null object
population2010                   178 non-null object
population2015                   178 non-null object
deaths_CVD_2000_%                178 non-null object
deaths_CVD_2005_%                178 non-null object
deaths_CVD_2010_%                178 non-null object
deaths_CVD_2015_%                178 non-null object
Raised total cholesterol 2008    178 non-null float64
dtypes: float64(1), object(9)
memory usage: 15.3+ KB
```

Blood pressure

http://apps.who.int/gho/data/node.main.A875STANDARD?lang=en Raised blood pressure (SBP ≥ 140 OR DBP ≥ 90), age-standardized (%). Estimates by country

Data Preparation and Cleaning - Part I - 3

In [36]:

```
Blood_pressure = pd.read_csv('C:\\Users\\Olga\\edx_Python_for_Data_science\
\FinalProject\\datasets\\heart\\BloodPressure.csv', sep=',')
Blood_pressure.head()
```

Out[36]:

	Unnamed: 0	2015	2015.1	2015.2	2014	2014.1	

	Unnamed: 0	2015	2015.1	2015.2	2014	2014.1	
0	NaN	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised pressu (SBP& OR DBP&g
1	NaN	18+ years	18+ years	18+ years	18+ years	18+ years	18+ ye
2	Country	Both sexes	Male	Female	Both sexes	Male	Femal
3	Afghanistan	30.6 [23.6-38.3]	30.4 [20.4-41.6]	30.7 [21.2-41.3]	30.6 [23.9-37.8]	30.4 [21.0-41.0]	30.6 [2 40.7]
4	Albania	29.0 [22.4-36.2]	33.0 [23.0-44.3]	25.0 [16.8-34.7]	29.2 [23.0-36.0]	33.2 [23.9-43.7]	25.3 [1 34.5]

5 rows × 124 columns

In [37]:

```
Blood_pressure.shape
```

Out[37]:

```
(198, 124)
```

In [38]:

```
Blood_pressure.columns
```

Out[38]:

```
Index(['Unnamed: 0', '2015', '2015.1', '2015.2', '2014', '2014.1', '2014.2'
,
        '2013', '2013.1', '2013.2',
        ...
        '1978.2', '1977', '1977.1', '1977.2', '1976', '1976.1', '1976.2',
        '1975', '1975.1', '1975.2'],
       dtype='object', length=124)
```

In [39]:

```
for c in Blood_pressure.columns:
    if (Blood_pressure[c].str.contains(" Male").any()):
        del Blood_pressure[c]
for c in Blood_pressure.columns:
    if (Blood_pressure[c].str.contains(" Female").any()):
        del Blood_pressure[c]
```

In [40]:

```
Blood_pressure.head()
```

Out[40]:

	Unnamed: 0	2015	2014	2013	2012	2011	
		Raised blood pressure	Raised blood pressure	Raised blood pressure	Raised blood pressure	Raised blood pressure	Raised pressu

	Unnamed: 0	2015	2014	2013	2012	2011	
0	NaN	(SBP&gt;=140 OR DBP&gt;...	(SBP&gt;=140 OR DBP&gt;...	(SBP&gt;=140 OR DBP&gt;...	(SBP&gt;=140 OR DBP&gt;...	(SBP&gt;=140 OR DBP&gt;...	(SBP& OR DBP&g
1	NaN	18+ years	18+ years	18+ years	18+ years	18+ years	18+ ye
2	Country	Both sexes	Both sexes	Both sexes	Both sexes	Both sexes	Both se
3	Afghanistan	30.6 [23.6-38.3]	30.6 [23.9-37.8]	30.5 [24.1-37.3]	30.5 [24.3-36.9]	30.4 [24.5-36.6]	30.4 [2 36.3]
4	Albania	29.0 [22.4-36.2]	29.2 [23.0-36.0]	29.5 [23.6-35.8]	29.7 [24.1-35.6]	29.9 [24.6-35.4]	30.1 [2 35.4]

5 rows × 42 columns

In [41]:

```
Blood_pressure.rename(columns={'Unnamed: 0': 'countries'}, inplace = True)
Blood_pressure.head()
```

Out[41]:

	countries	2015	2014	2013	2012	2011	
0	NaN	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised blood pressure (SBP&gt;=140 OR DBP&gt;...	Raised pressu (SBP& OR DBP&g
1	NaN	18+ years	18+ years	18+ years	18+ years	18+ years	18+ ye
2	Country	Both sexes	Both sexes	Both sexes	Both sexes	Both sexes	Both se
3	Afghanistan	30.6 [23.6-38.3]	30.6 [23.9-37.8]	30.5 [24.1-37.3]	30.5 [24.3-36.9]	30.4 [24.5-36.6]	30.4 [2 36.3]
4	Albania	29.0 [22.4-36.2]	29.2 [23.0-36.0]	29.5 [23.6-35.8]	29.7 [24.1-35.6]	29.9 [24.6-35.4]	30.1 [2 35.4]

5 rows × 42 columns

In [42]:

```
Blood_pressure.drop([0,1,2], inplace = True)
Blood_pressure.head()
```

Out[42]:

	countries	2015	2014	2013	2012	2011	2010	2009	2008	2007	...	1984	1983	19
3	Afghanistan	30.6 [23.6-38.3]	30.6 [23.9-37.8]	30.5 [24.1-37.3]	30.5 [24.3-36.9]	30.4 [24.5-36.6]	30.4 [24.6-36.3]	30.3 [24.6-36.0]	30.2 [24.7-35.9]	30.1 [24.7-35.6]	...	26.2 [20.0-32.6]	25.9 [19.7-32.6]	25 [19 32
4		29.0	29.2	29.5	29.7	29.9	30.1	30.2	30.4	30.5		32.1	32.2	32

	countries	2015	2014	2013	2012	2011	2010	2009	2008	2007	...	1984	1983	19
4	Albania	[22.4-36.2]	[23.0-36.0]	[23.6-35.8]	[24.1-35.6]	[24.6-35.4]	[25.0-35.4]	[25.4-35.4]	[25.6-35.5]	[25.8-35.4]	...	[26.0-38.4]	[25.9-38.6]	[25 39
5	Algeria	25.1 [19.4-31.5]	25.4 [20.0-31.5]	25.8 [20.6-31.5]	26.2 [21.2-31.5]	26.5 [21.8-31.6]	26.9 [22.3-31.8]	27.2 [22.8-32.0]	27.6 [23.2-32.2]	27.9 [23.6-32.5]	...	31.7 [25.6-38.0]	31.6 [25.5-38.1]	31 [25 38
6	Andorra	18.7 [13.3-24.8]	19.2 [14.0-24.9]	19.6 [14.6-25.1]	20.1 [15.3-25.4]	20.6 [16.0-25.7]	21.1 [16.6-26.0]	21.7 [17.3-26.4]	22.2 [17.9-26.9]	22.8 [18.5-27.4]	...	35.7 [29.7-41.9]	36.1 [30.0-42.5]	36 [30 43
7	Angola	29.7 [22.1-38.3]	29.8 [22.6-38.0]	30.0 [22.9-37.7]	30.1 [23.4-37.5]	30.2 [23.7-37.4]	30.4 [24.0-37.3]	30.5 [24.2-37.2]	30.6 [24.4-37.3]	30.6 [24.5-37.2]	...	28.9 [21.9-36.5]	28.7 [21.6-36.4]	28 [21 36

5 rows × 42 columns

In [43]:

```
Blood_pressure = Blood_pressure.applymap(lambda x: delete_brackets(x))
Blood_pressure.tail()
```

Out[43]:

	countries	2015	2014	2013	2012	2011	2010	2009	2008	2007	...	1984	1983	1982	198
193	Venezuela (Bolivarian Republic of)	18.6	19.0	19.4	19.8	20.2	20.6	21.0	21.5	21.9	...	32.8	33.1	33.4	33.6
194	Viet Nam	23.4	23.4	23.3	23.2	23.2	23.1	23.0	22.9	22.9	...	20.6	20.6	20.5	20.5
195	Yemen	30.7	30.7	30.7	30.6	30.6	30.6	30.5	30.5	30.5	...	27.4	27.2	26.9	26.7
196	Zambia	27.1	27.2	27.3	27.3	27.4	27.5	27.5	27.6	27.7	...	29.3	29.4	29.4	29.4
197	Zimbabwe	28.2	28.3	28.4	28.5	28.6	28.7	28.7	28.8	28.9	...	29.2	29.1	29.0	28.9

5 rows × 42 columns

In [44]:

```
def clean_numbers(text):
    return re.sub("\D", "", text)
```

In [45]:

```
Blood_pressure2000_2015 = Blood_pressure[['2015', '2010', '2005', '2000']]
Blood_pressure2000_2015 = Blood_pressure2000_2015.applymap(lambda x: clean_
numbers(x))
Blood_pressure2000_2015.head()
```

Out[45]:

	2015	2010	2005	2000
3	306	304	299	291

	2015	2010	2005	2000
4	290	301	308	312
5	251	269	286	300
6	187	211	239	267
7	297	304	307	307

In [46]:

```
for c in Blood_pressure2000_2015.columns:
    Blood_pressure2000_2015[c] = pd.to_numeric(Blood_pressure2000_2015[c])
    Blood_pressure2000_2015[c] = Blood_pressure2000_2015[c]/10
Blood_pressure2000_2015['countries'] = Blood_pressure['countries']
Blood_pressure2000_2015.head()
```

Out[46]:

	2015	2010	2005	2000	countries
3	30.6	30.4	29.9	29.1	Afghanistan
4	29.0	30.1	30.8	31.2	Albania
5	25.1	26.9	28.6	30.0	Algeria
6	18.7	21.1	23.9	26.7	Andorra
7	29.7	30.4	30.7	30.7	Angola

In [47]:

```
data = data.merge(Blood_pressure2000_2015, on='countries', how='inner')
data.rename(columns = {'2015': 'Raised blood pressure 2015',
                       '2010': 'Raised blood pressure 2010',
                       '2005': 'Raised blood pressure 2005',
                      '2000': 'Raised blood pressure 2000'}, inplace = True
data.head()
```

Out[47]:

	countries	population2000	population2005	population2010	population2015	deaths_CVD_
0	Afghanistan	19702	24400	27962	32527	13.4281
1	Albania	3122	3082	2902	2897	50.6032
2	Algeria	31184	33268	36036	39667	33.0461
3	Angola	15059	17913	21220	25022	7.17169
4	Antigua and Barbuda	78	83	87	92	34.6055

In [64]:

```
data['deaths_CVD_2000_%'] = pd.to_numeric(data['deaths_CVD_2000_%'])
data['deaths_CVD_2005_%'] = pd.to_numeric(data['deaths_CVD_2005_%'])
data['deaths_CVD_2010_%'] = pd.to_numeric(data['deaths_CVD_2010_%'])
data['deaths_CVD_2015_%'] = pd.to_numeric(data['deaths_CVD_2015_%'])
```

Findings - Part I - 2

In [65]:

```
correlation = data.corr()
correlation
```

Out[65]:

	deaths_CVD_2000_%	deaths_CVD_2005_%	deaths_CVD_2010_%	deat
deaths_CVD_2000_%	1.000000	0.982884	0.959266	0.92
deaths_CVD_2005_%	0.982884	1.000000	0.987547	0.95
deaths_CVD_2010_%	0.959266	0.987547	1.000000	0.97
deaths_CVD_2015_%	0.922854	0.958494	0.971850	1.00
Raised total cholesterol 2008	0.661859	0.595760	0.535489	0.47
Raised blood pressure 2015	-0.226977	-0.185131	-0.133603	-0.09
Raised blood pressure 2010	-0.155737	-0.123160	-0.080383	-0.04
Raised blood pressure 2005	-0.037148	-0.017837	0.012380	0.02
Raised blood pressure 2000	0.120877	0.124949	0.139854	0.13

In [67]:

```
corr_table = [correlation.loc['Raised blood pressure 2000',
'deaths_CVD_2000_%'],
              correlation.loc['Raised blood pressure 2005', 'deaths_CVD_200
_%'],
              correlation.loc['Raised blood pressure 2010', 'deaths_CVD_201
_%'],
              correlation.loc['Raised blood pressure 2015', 'deaths_CVD_201
_%']]
```

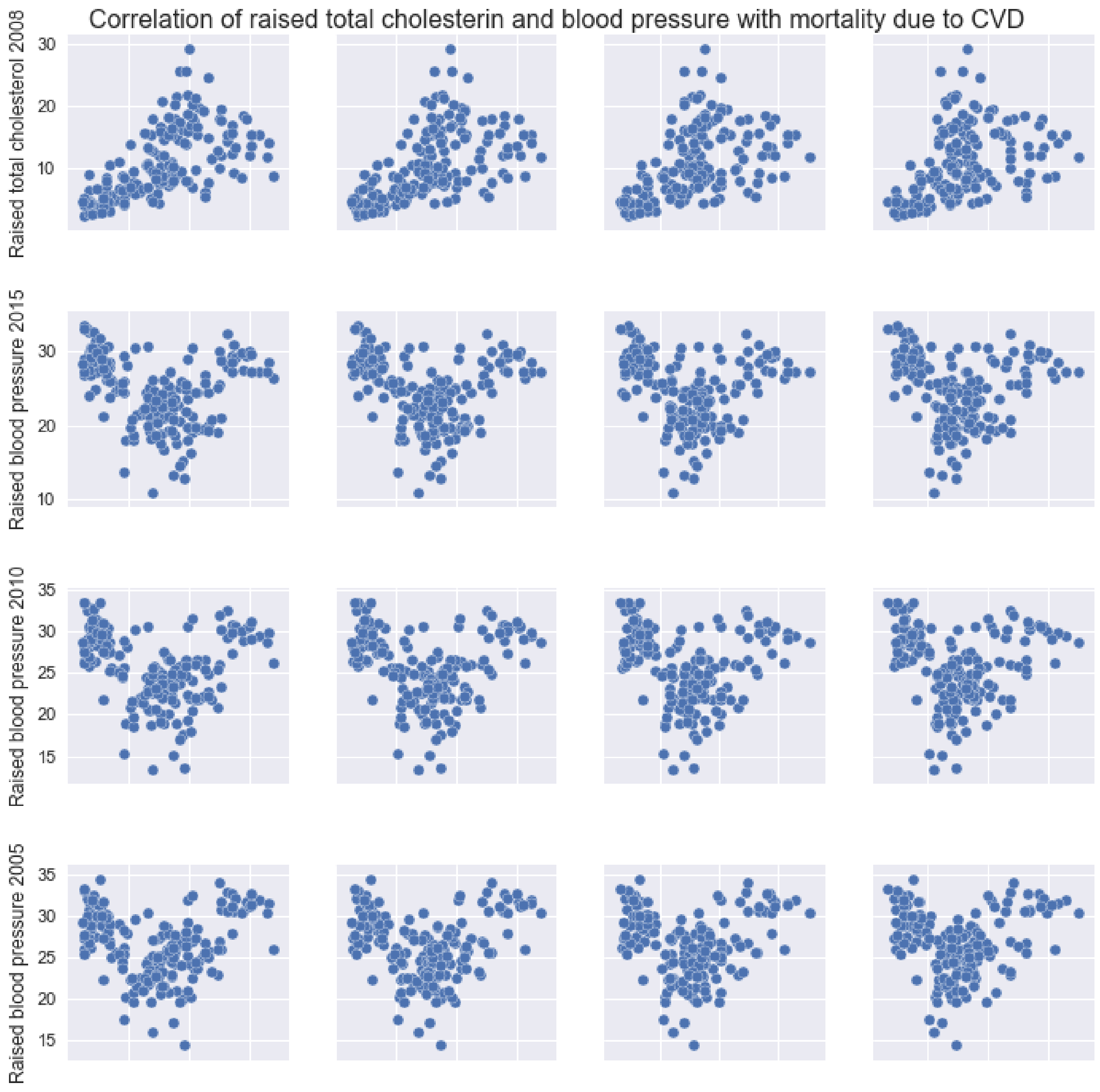
```
corr_table
```

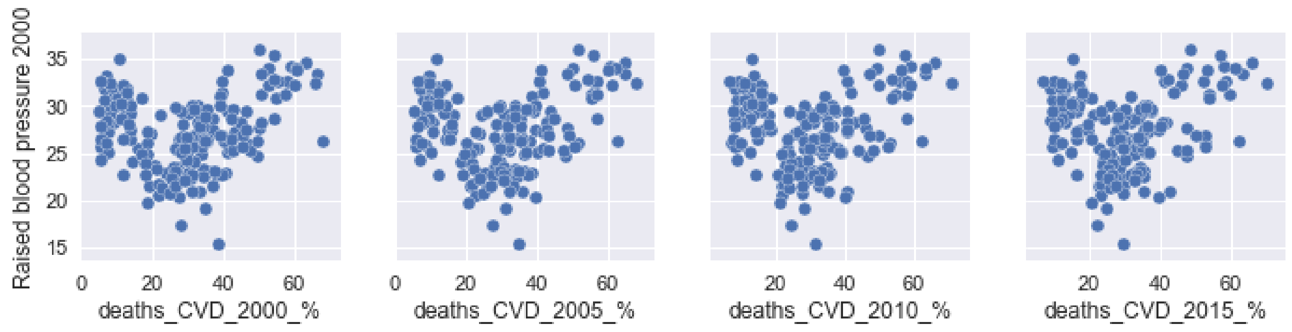
Out[67]:

```
[0.12087711815826553,
 -0.017836763395666973,
 -0.0803830134350187,
 -0.09401785294434403]
```

In [204]:

```
sns.set(font_scale=1.1)
fig2 = sns.pairplot(data, x_vars=['deaths_CVD_2000_%', 'deaths_CVD_2005_%',
'deaths_CVD_2010_%', 'deaths_CVD_2015_%'],
                             y_vars=['Raised total cholesterol 2008', 'Raised bl
ood pressure 2015', 'Raised blood pressure 2010',
                                      'Raised blood pressure 2005', 'Raised bloo
pressure 2000'])
plt.suptitle('Correlation of raised total cholesterin and blood pressure wi
th mortality due to CVD', fontsize = 16)
fig2.savefig('fig2', bbox_inches = 'tight')
```





Findings - Part I -3

In [75]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [76]:

```
data1 = data.copy()
```

In [77]:

```
data1.isnull().any()
```

Out[77]:

```
countries                        False
population2000                   False
population2005                   False
population2010                   False
population2015                   False
deaths_CVD_2000_%                False
deaths_CVD_2005_%                False
deaths_CVD_2010_%                False
deaths_CVD_2015_%                False
Raised total cholesterol 2008    False
Raised blood pressure 2015        True
Raised blood pressure 2010        True
Raised blood pressure 2005        True
Raised blood pressure 2000        True
dtype: bool
```

In [78]:

```
data1.dropna(inplace = True)
```

In [79]:

```
data1.isnull().any()
```

Out[79]:

```
countries                        False
population2000                   False
population2005                   False
```

```
population2010                False
population2015                False
deaths_CVD_2000_%             False
deaths_CVD_2005_%             False
deaths_CVD_2010_%             False
deaths_CVD_2015_%             False
Raised total cholesterol 2008    False
Raised blood pressure 2015    False
Raised blood pressure 2010    False
Raised blood pressure 2005    False
Raised blood pressure 2000    False
dtype: bool
```

In [80]:

```
data1.shape
```

Out[80]:

```
(177, 14)
```

In [81]:

```
data1.rename(index = data1['countries'], inplace = True)
del data1['countries']
data1.head()
```

Out[81]:

	population2000	population2005	population2010	population2015	deaths_CVD_2
Afghanistan	19702	24400	27962	32527	13.428143
Albania	3122	3082	2902	2897	50.603197
Algeria	31184	33268	36036	39667	33.046056
Angola	15059	17913	21220	25022	7.171688
Antigua and Barbuda	78	83	87	92	34.605471

Prediction of mortality due to CVD in 2010 depending on percentage of people with raised blood pressure and raised total cholesterol using linear regression

In [297]:

```
X = data1[['Raised blood pressure 2010', 'Raised total cholesterol 2008']]
```

In [298]:

```
y = data1['deaths_CVD_2010_%']
```

In [299]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=324)
```

In [300]:

```
regressor1 = LinearRegression()
regressor1.fit(X_train, y_train)
```

Out[300]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1,
normalize=False)
```

In [301]:

```
y_prediction = regressor1.predict(X_test)
y_prediction
```

Out[301]:

```
array([29.67782929, 37.45946249, 24.00168163, 20.48622191, 42.39387652,
       26.85821133, 39.68028344, 41.46813487, 23.91876788, 25.40245541,
       36.25682773, 25.23662792, 29.58582528, 42.08051221, 33.9487131 ,
       26.11198763, 19.14525074, 26.24517318, 27.6966593 , 26.16247971,
       27.16708557, 25.82600426, 23.11252132, 30.97706826, 36.26613827,
       25.67890798, 22.78995661, 23.66991652, 39.357939  , 24.59609871,
       25.5774833 , 31.22120929, 23.14001238, 21.57801132, 22.7574248 ,
       20.42630917, 27.38789519, 22.41667956, 40.81829512, 37.00327169,
       23.49915843, 25.15809411, 22.01591144, 22.4719921 , 29.43357812,
       46.19520948, 40.88729813, 23.84954461, 33.34531589, 24.48996369,
       29.65942849, 21.48096657, 31.89393991, 26.3743092 , 36.48727829,
       22.62850905, 38.860016  , 29.2538399 , 32.87970443])
```

In [302]:

```
y_test
```

Out[302]:

```
Saint Lucia                              33.686022
Bahrain                                  30.683816
Malawi                                    7.883203
Nigeria                                   8.474171
Austria                                  42.786440
Bhutan                                   25.529955
Kuwait                                   41.808753
Croatia                                  48.958651
Eritrea                                  16.051473
Chad                                      8.091067
Brunei Darussalam                        32.106465
Niger                                     9.144352
Yemen                                    32.818790
Netherlands                              28.323574
Micronesia (Federated States of)         31.796013
Mali                                     10.538545
Democratic People's Republic of Korea    39.718613
Venezuela (Bolivarian Republic of)       29.935618
Turkmenistan                             46.901109
Congo                                    15.770136
Mauritania                               12.835200
Samoa                                    33.954355
Togo                                     13.127303
```

```
Togo
Armenia                                  45.963406
Bahamas                                  32.983075
Burkina Faso                             11.584694
Burundi                                   9.748419
United Republic of Tanzania               9.892451
Serbia                                   55.925850
Gambia                                   12.378053
Senegal                                  15.934719
Barbados                                 29.737987
Bangladesh                               28.364915
Guatemala                                14.066819
China                                    40.475282
Ghana                                    18.313218
Vanuatu                                  31.060847
Haiti                                     7.304141
Portugal                                 33.935380
Spain                                    30.839557
Bolivia (Plurinational State of)         21.613810
Timor-Leste                              23.509421
Sao Tome and Principe                    15.832305
Cameroon                                 12.184027
Trinidad and Tobago                      32.898380
Hungary                                  49.818855
United Arab Emirates                     35.262539
Uzbekistan                               52.333770
Equatorial Guinea                        13.791899
Cambodia                                 23.707218
Morocco                                  33.759503
Republic of Korea                        24.444687
Iraq                                     31.643114
Grenada                                  33.590380
Thailand                                 23.529917
Jamaica                                  29.369281
Cyprus                                   37.588810
Uruguay                                  30.815485
Mexico                                   22.813135
Name: deaths_CVD_2010_%, dtype: float64
```

In [303]:

```
X_test.index
```

Out[303]:

```
Index(['Saint Lucia', 'Bahrain', 'Malawi', 'Nigeria', 'Austria', 'Bhutan',
       'Kuwait', 'Croatia', 'Eritrea', 'Chad', 'Brunei Darussalam',
'Niger',
       'Yemen', 'Netherlands', 'Micronesia (Federated States of)', 'Mali',
       'Democratic People's Republic of Korea',
       'Venezuela (Bolivarian Republic of)', 'Turkmenistan', 'Congo',
       'Mauritania', 'Samoa', 'Togo', 'Armenia', 'Bahamas', 'Burkina Faso',
       'Burundi', 'United Republic of Tanzania', 'Serbia', 'Gambia',
'Senegal',
       'Barbados', 'Bangladesh', 'Guatemala', 'China', 'Ghana', 'Vanuatu',
       'Haiti', 'Portugal', 'Spain', 'Bolivia (Plurinational State of)',
       'Timor-Leste', 'Sao Tome and Principe', 'Cameroon',
       'Trinidad and Tobago', 'Hungary', 'United Arab Emirates',
'Uzbekistan',
       'Equatorial Guinea', 'Cambodia', 'Morocco', 'Republic of Korea', 'Ir
aq',
       'Grenada', 'Thailand', 'Jamaica', 'Cyprus', 'Uruguay', 'Mexico'],
       'Grenada', 'Thailand', 'Jamaica', 'Cyprus', 'Uruguay', 'Mexico'],
      dtype='object')
```

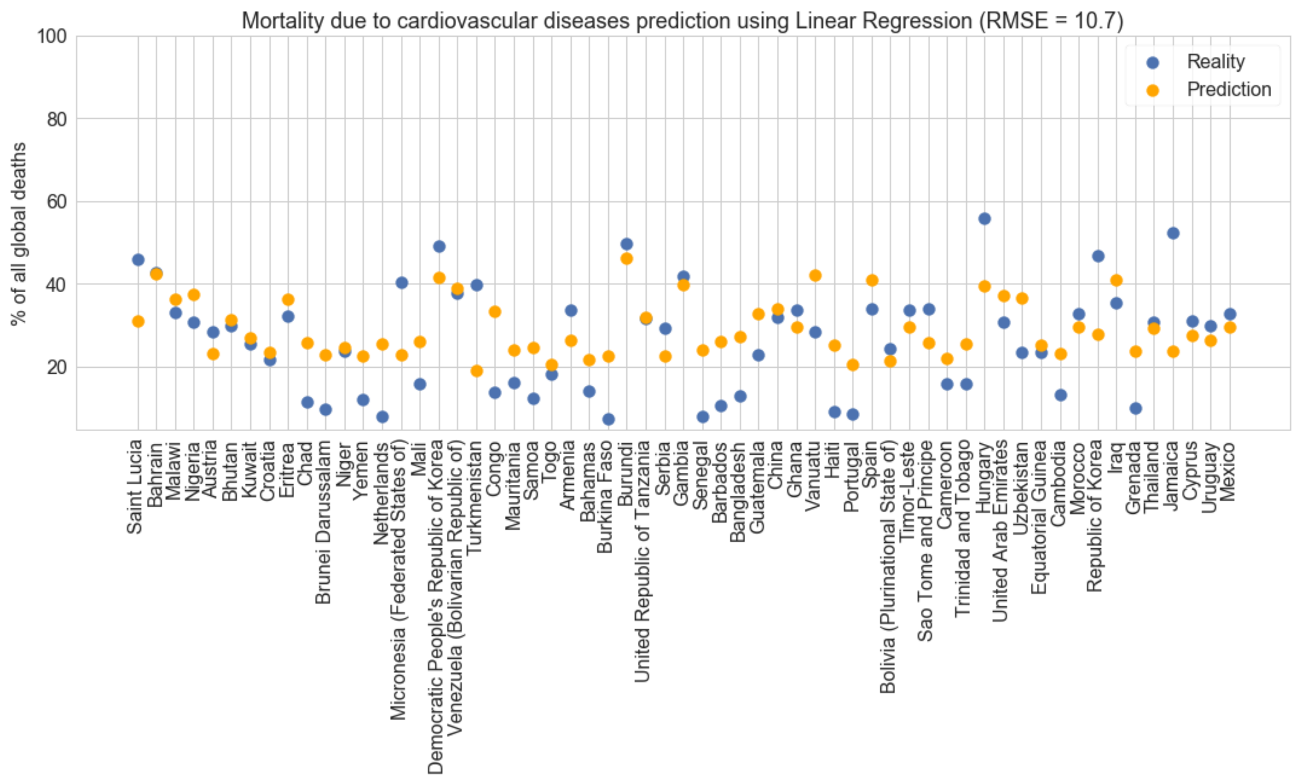
In [304]:

```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
print(RMSE)
```

```
10.686786364853736
```

In [305]:

```
sns.set_style("whitegrid")
fig1_2, ax = plt.subplots(figsize=(18, 6))

ax.scatter(X_test.index, y_test, label = 'Reality', s = 100)
ax.scatter(X_test.index, y_prediction, label = 'Prediction', s = 100, c = 'o
range')

ax.set_title('Mortality due to cardiovascular diseases prediction using Lin
ear Regression (RMSE = 10.7)', size =18)
ax.set_xticklabels(X_test.index, rotation = 'vertical', fontsize =16)
ax.set_ylabel('% of all global deaths', size =16)

ax.legend(fontsize =16, frameon = True, facecolor = 'white')
plt.tick_params(labelsize = 16)
ax.set_ylim(top = 100)

plt.show()
fig1_2.savefig('fig1_2', bbox_inches = 'tight')
```



In [306]:

```
y_test.describe()
```

Out[306]:

```
count     59.000000
mean      26.744728
```

```
std       12.740387
min        7.304141
25%       14.918478
50%       29.369281
75%       33.722763
max       55.925850
Name: deaths_CVD_2010_%, dtype: float64
```

Prediction using Decision Tree Regressor

In [307]:

```
regressor2 = DecisionTreeRegressor(max_depth=100)
regressor2.fit(X_train, y_train)
```

Out[307]:

```
DecisionTreeRegressor(criterion='mse', max_depth=100, 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')
```

In [308]:

```
y_prediction = regressor2.predict(X_test)
y_prediction
```

Out[308]:

```
array([36.09703025, 33.88870064, 12.62777865, 38.31092546, 37.19007263,
       27.23657715, 45.31326048, 57.55403092,  7.48527149, 12.21137985,
       35.00264993, 10.87922152, 11.94936827, 40.31895426, 36.2155165 ,
       17.95923816, 39.09580736, 21.57317677, 52.61774448, 49.84305125,
       17.95923816, 52.61774448, 11.7357726 , 45.31326048, 27.32939234,
       12.21137985, 11.7357726 , 14.9089725 , 57.55403092, 17.95923816,
       17.95923816, 35.96866142, 23.93441422, 23.93349326, 39.09580736,
       38.31092546, 33.9410438 , 38.31092546, 36.09703025, 43.91274209,
       21.57317677,  8.39604112, 11.7357726 ,  9.00885344, 35.96866142,
       39.31341937, 33.44875641, 23.93441422, 70.74088217, 23.93441422,
       57.91477291, 20.89107361, 36.09703025, 33.93894714, 27.32939234,
       31.36546626, 40.31895426, 32.20579995, 46.66596609])
```

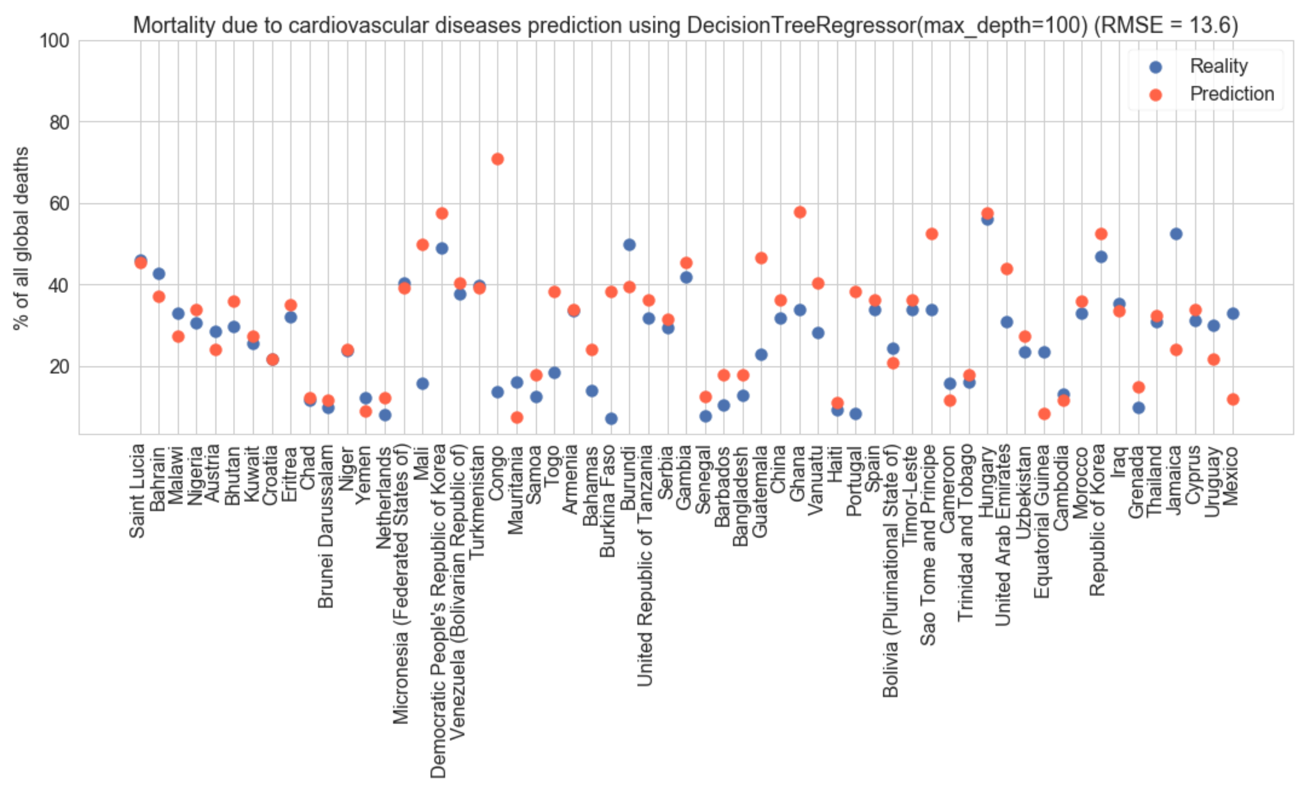
In [309]:

```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
print(RMSE)
```

```
13.58534043507599
```

In [311]:

```
sns.set_style("whitegrid")
fig1_3, ax = plt.subplots(figsize=(18, 6))

ax.scatter(X_test.index, y_test, label = 'Reality', s = 100)
ax.scatter(X_test.index, y_prediction, label = 'Prediction', s = 100, c = 't
omato')

ax.set_title('Mortality due to cardiovascular diseases prediction using Dec
isionTreeRegressor(max_depth=100) (RMSE = 13.6)', size = 18)
isionTreeRegressor(max_depth=100) (RMSE = 13.6)', size =18)
ax.set_xticklabels(X_test.index, rotation = 'vertical', fontsize =16)
ax.set_ylabel('% of all global deaths', size =16)

ax.legend(fontsize =16, frameon = True, facecolor = 'white')
plt.tick_params(labelsize = 16)
ax.set_ylim(top = 100)

plt.show()
fig1_3.savefig('fig1_3', bbox_inches = 'tight')
```



RMSE is very high => these data are not enough for prediction.

Part II. Cardiovascular disease presence prediction for a given patient.

https://www.kaggle.com/danimal/heartdiseaseensembleclassifier

In [2]:

```
heart_disease = pd.read_csv('C:\\Users\\Olga\\edx_Python_for_Data_science\\
FinalProject\\datasets\\heart\\heart_disease_ensemble_classifier\\Heart_Dis
ease_Data.csv', sep=',', na_values="?")
```

In [3]:

```
heart_disease.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slop	ca	thal	pred_attri
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slop	ca	thal	pred_attri
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0

Attributes:	
age	(age in years)
sex	(1 = male, 0 = female)
cp	(chest pain type: 1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic
trestbps	(resting blood pressure (in mm Hg on admission to the hospital))
chol	(serum cholesterol in mg/dl)
fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
restecg	(resting electrocardiographic results: 0 = normal, 1 = having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria
thalach	(maximum heart rate achieved)
exang	(exercise induced angina: 1 = yes, 0 = no)
oldpeak	(ST depression induced by exercise relative to rest)
slope	(the slope of the peak exercise ST segment: 1 = upsloping, 2 = flat, 3 = downsloping
ca	(number of major vessels (0-3) colored by fluoroscopy)
thal	(3 = normal, 6 = fixed defect, 7 = reversable defect)
pred_attribute	(the predicted attribute: diagnosis of heart disease (angiographic disease status): 0 = < 50% diameter narrowing (Healthy), 1 = > 50% diameter narrowing (Sick))

In [4]:

```
heart_disease.shape
```

Out[4]:

```
(303, 14)
```

Data Preparation and Cleaning - Part II

In [5]:

```
heart_disease.dropna(inplace=True)
```

In [6]:

```
heart_disease.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 297 entries, 0 to 301
Data columns (total 14 columns):
age                 297 non-null int64
sex                 297 non-null int64
cp                  297 non-null int64
trestbps            297 non-null int64
chol                297 non-null int64
fbs                 297 non-null int64
restecg             297 non-null int64
thalach             297 non-null int64
exang               297 non-null int64
oldpeak             297 non-null float64
slop                297 non-null int64
ca                  297 non-null float64
thal                297 non-null float64
pred_attribute      297 non-null int64
dtypes: float64(3), int64(11)
memory usage: 34.8 KB
```

In [7]:

```python
heart_disease.isnull().any()
```

Out[7]:

```
age                 False
sex                 False
cp                  False
trestbps            False
chol                False
fbs                 False
restecg             False
thalach             False
exang               False
oldpeak             False
slop                False
ca                  False
thal                False
pred_attribute      False
dtype: bool
```

In [8]:

```python
heart_disease.rename(columns = {'slop': 'slope'}, inplace = True)
```

In [9]:

```python
heart_disease.corr()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
age	1.000000	-0.092399	0.110471	0.290476	0.202644	0.132062	0.149917	-0.394563
sex	-0.092399	1.000000	0.008908	-0.066340	-0.198089	0.038850	0.033897	-0.060496
				-		-		-

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
cp	0.110471	0.008908	1.000000	0.036980	0.072088	0.057663	0.063905	0.339308
trestbps	0.290476	-0.066340	-0.036980	1.000000	0.131536	0.180860	0.149242	-0.049108
chol	0.202644	-0.198089	0.072088	0.131536	1.000000	0.012708	0.165046	-0.000075
fbs	0.132062	0.038850	-0.057663	0.180860	0.012708	1.000000	0.068831	-0.007842
restecg	0.149917	0.033897	0.063905	0.149242	0.165046	0.068831	1.000000	-0.072290
thalach	-0.394563	-0.060496	-0.339308	-0.049108	-0.000075	-0.007842	-0.072290	1.000000
exang	0.096489	0.143581	0.377525	0.066691	0.059339	-0.000893	0.081874	-0.384368
oldpeak	0.197123	0.106567	0.203244	0.191243	0.038596	0.008311	0.113726	-0.347640
slope	0.159405	0.033345	0.151079	0.121172	-0.009215	0.047819	0.135141	-0.389307
ca	0.362210	0.091925	0.235644	0.097954	0.115945	0.152086	0.129021	-0.268727
thal	0.126586	0.383652	0.268500	0.138183	0.010859	0.062209	0.018795	-0.274831
pred_attribute	0.222156	0.226797	0.404248	0.159620	0.066448	0.049040	0.184136	-0.420639

In [10]:

```
heart_disease = heart_disease[heart_disease['pred_attribute'] <= 1]
```

In [11]:

```
heart_disease.shape
```

Out[11]:

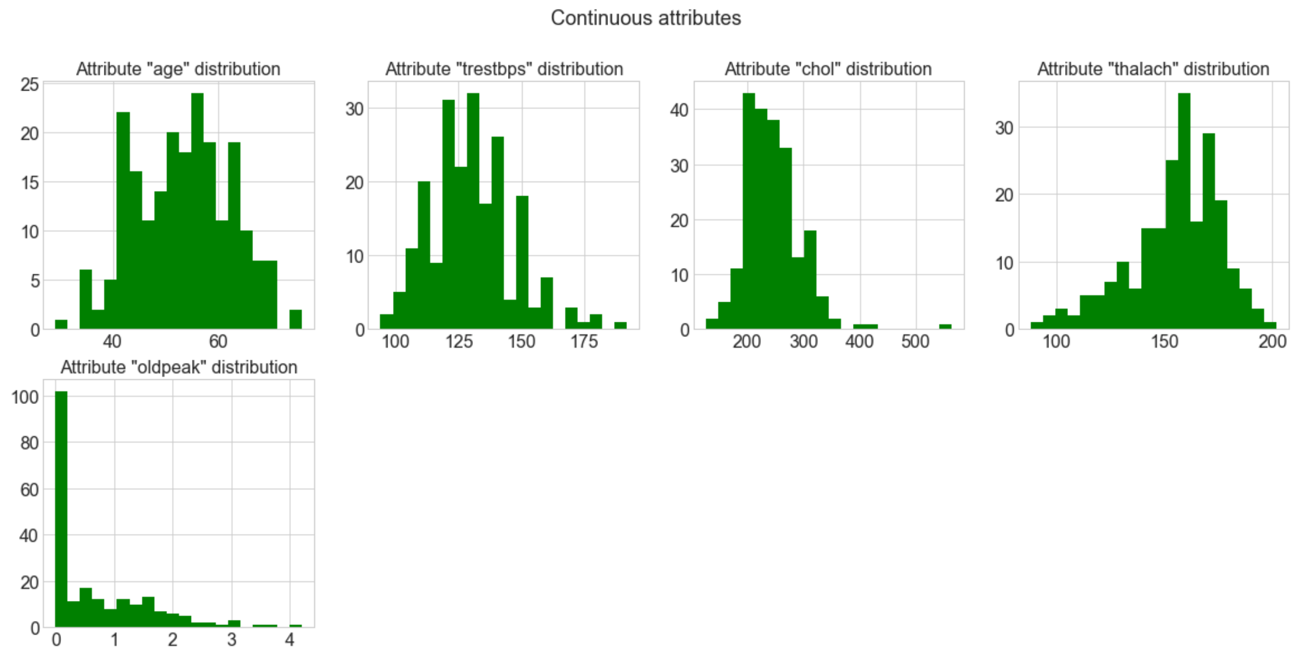
```
(214, 14)
```

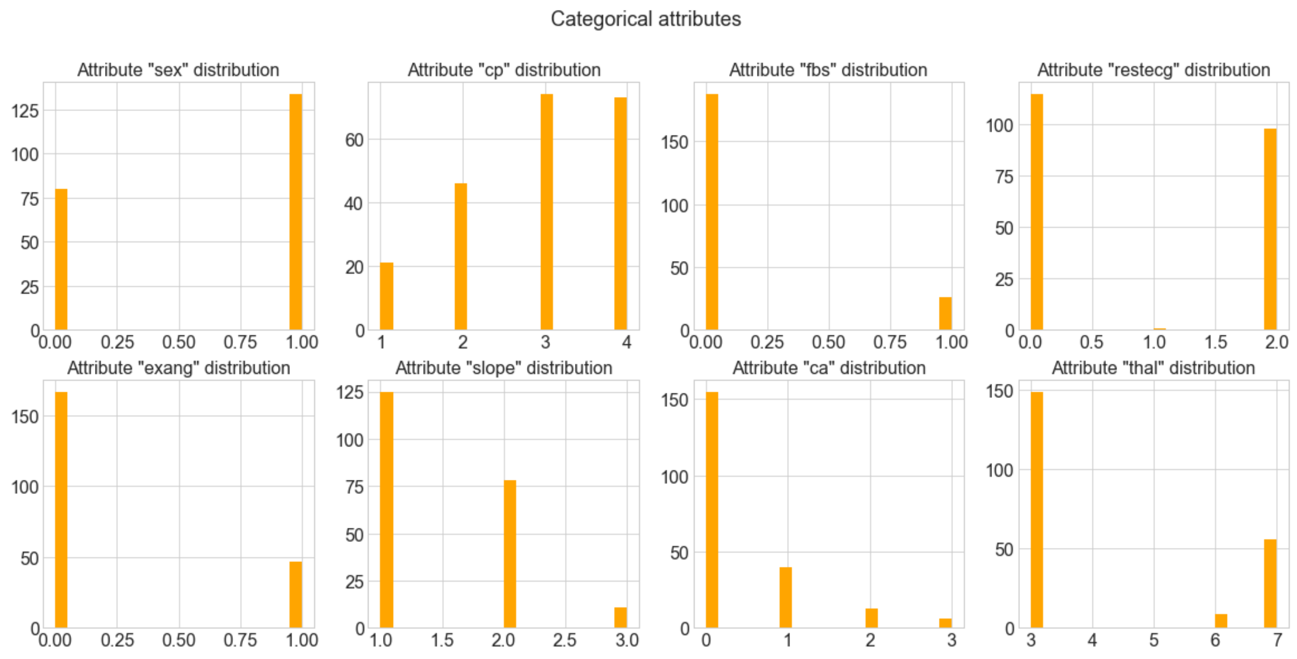
Findings - Part II - 1

In [12]:

```
sns.set_style('whitegrid')
fig3=plt.figure(figsize=(20,9))
for i, feature in enumerate(heart_disease.columns[[0, 3, 4, 7, 9]]):
    ax=fig3.add_subplot(2, 4, i+1)
    heart_disease[feature].hist(bins=20, ax=ax, facecolor='green')
    ax.set_title('Attribute "' + feature + '" distribution', fontsize = 16)
    plt.tick_params(labelsize = 16)
```

```
fig3.suptitle('Continuous attributes', fontsize = 18)
fig3.savefig('fig3', bbox_inches = 'tight')

fig4=plt.figure(figsize=(20,9))
for i, feature in enumerate(heart_disease.columns[[1, 2, 5, 6, 8, 10, 11, 12
]]):
    ax=fig4.add_subplot(2, 4, i+1)
    heart_disease[feature].hist(bins=20, ax=ax, facecolor='orange')
    ax.set_title('Attribute "' + feature + '" distribution', fontsize = 16)
    plt.tick_params(labelsize = 16)

fig4.suptitle('Categorical attributes', fontsize = 18)
fig4.savefig('fig4', bbox_inches = 'tight')
plt.show()
```

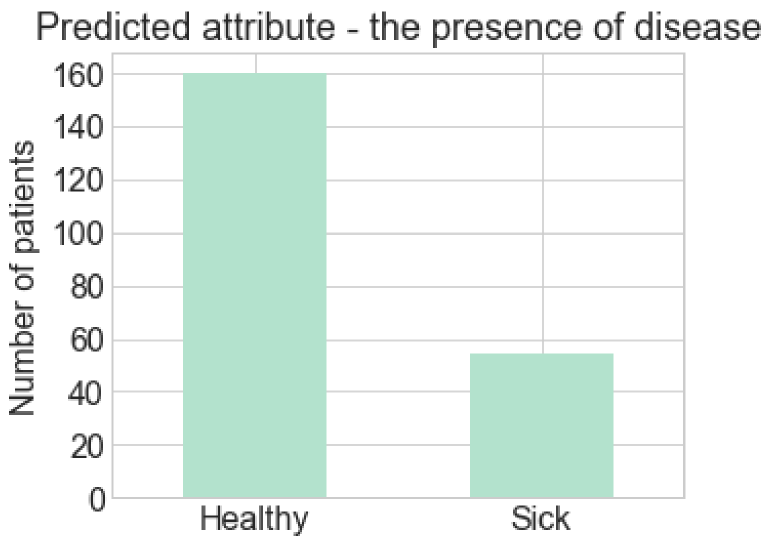




In [13]:

```
qq = heart_disease['pred_attribute'].copy()
qq[qq==0] = 'Healthy'
qq[qq==1] = 'Sick'
fig5 = plt.figure(figsize = (5,4))
```

```
qq.value_counts().plot(kind = 'bar',  colormap = 'Pastel2', fontsize = 16)
plt.ylabel('Number of patients', size = 16)
plt.xticks([0,1], ['Healthy', 'Sick'], rotation='horizontal')
plt.title('Predicted attribute - the presence of disease', fontsize = 18)

fig5.savefig('fig5', bbox_inches = 'tight')
```



In [14]:

```
qq.value_counts()
```

Out[14]:

```
Healthy    160
Sick        54
Name: pred_attribute, dtype: int64
```

In [222]:

```
heart_disease.describe()
```

Out[222]:

	age	sex	cp	trestbps	chol	fbs	restecg	
count	214.000000	214.000000	214.000000	214.00000	214.000000	214.000000	214.000000	21
mean	53.392523	0.626168	2.929907	130.21028	244.920561	0.121495	0.920561	15
std	9.233799	0.484954	0.973712	16.90314	50.835815	0.327468	0.996825	20
min	29.000000	0.000000	1.000000	94.00000	126.000000	0.000000	0.000000	88
25%	46.000000	0.000000	2.000000	120.00000	211.000000	0.000000	0.000000	14
50%	54.000000	1.000000	3.000000	130.00000	240.000000	0.000000	0.000000	15
75%	60.000000	1.000000	4.000000	140.00000	269.750000	0.000000	2.000000	17
max	76.000000	1.000000	4.000000	192.00000	564.000000	1.000000	2.000000	20

In [223]:

```
heart_disease.columns[[0, 3, 4, 7, 9]]
```

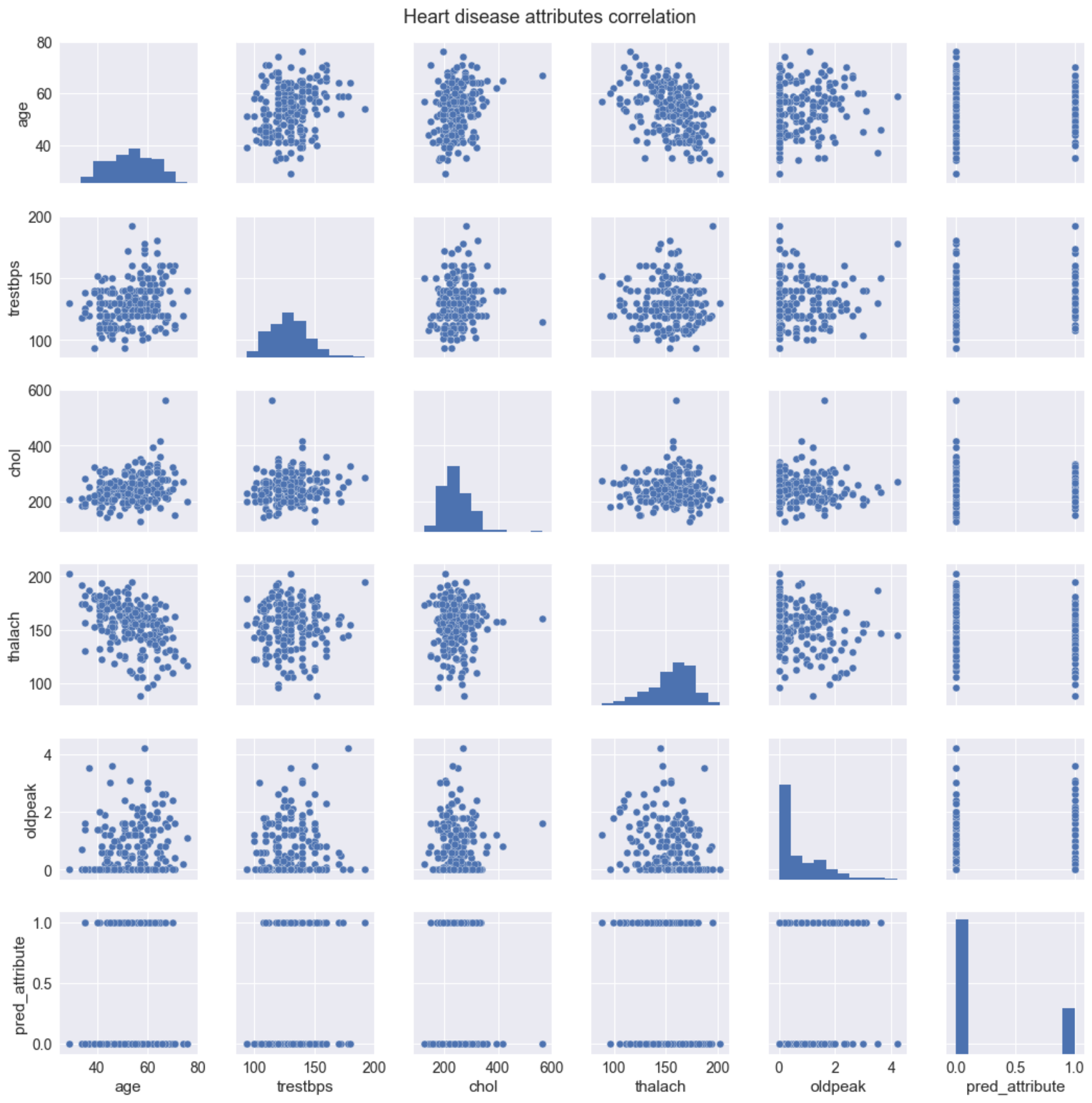
Out[223]:

Out[223]:

```
Index(['age', 'trestbps', 'chol', 'thalach', 'oldpeak'], dtype='object')
```

In [224]:

```
sns.set(font_scale=1.4)
fig6 = sns.pairplot(heart_disease[['age', 'trestbps', 'chol', 'thalach', 'o
ldpeak', 'pred_attribute']])
plt.subplots_adjust(top=0.95)
plt.suptitle('Heart disease attributes correlation', fontsize = 18)
plt.show()

fig6.savefig('fig6', bbox_inches = 'tight')
```



Findings - Part II - 2

Classification

In [225]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.linear_model import SGDClassifier
from sklearn import linear_model, metrics, model_selection
from sklearn.metrics import accuracy_score
```

In [226]:

```
heart_disease.columns
```

Out[226]:

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'pred_attribute'],
      dtype='object')
```

In [227]:

```
X = heart_disease[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg'
, 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal']]
y = heart_disease['pred_attribute']
```

In [231]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33,
random_state = 324)
```

In [232]:

```
def Classification(classifier):
    classifier.fit(X_train, y_train)
    return classifier.predict(X_test)
```

In [233]:

```
y_pred1 = Classification(DecisionTreeClassifier(max_leaf_nodes=50, random_s
tate=0))
y_pred2 = Classification(SGDClassifier(max_iter = 10000)) #Stochastic Gradi
ent Descent Classifier
y_pred3 = Classification(svm.SVC()) #Support Vector Classification
```

Accuracy: the fraction of correct predictions (true positive + true negative / all)

In [234]:

```
A1 = accuracy_score(y_test, y_pred1)
A2 = accuracy_score(y_test, y_pred2)
A3 = accuracy_score(y_test, y_pred3)
print(A1, A2, A3)
```

```
0.7464788732394366 0.7887323943661971 0.7887323943661971
```

In [256]:

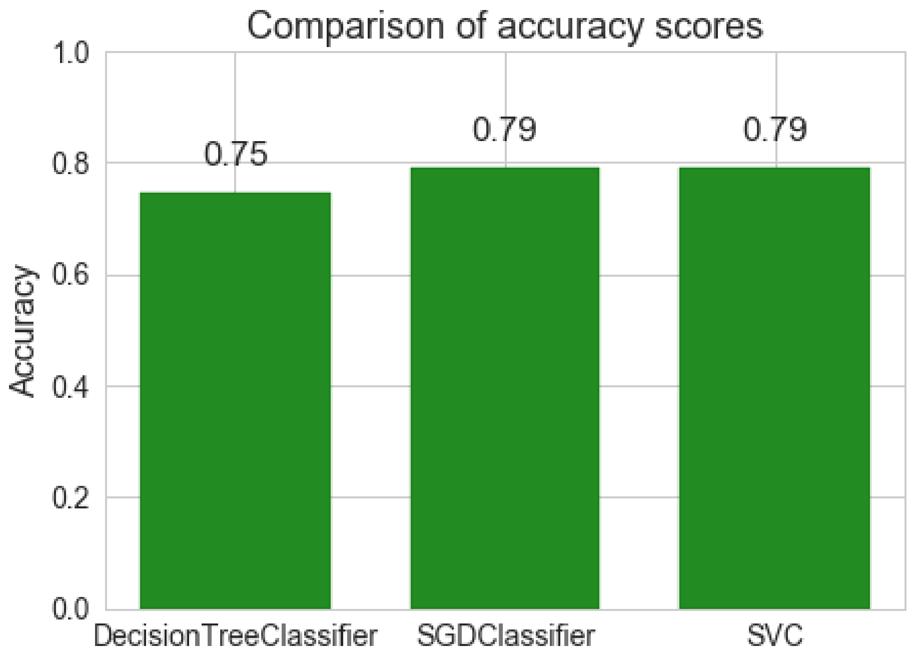
```
round(A1,2)
```

Out[256]:

```
0.75
```

In [263]:

```
acc = [A1, A2, A3]
names = ['DecisionTreeClassifier', 'SGDClassifier', 'SVC']

sns.set_style('whitegrid')
fig_acc, ax = plt.subplots(figsize = (7,5))
plt.ylabel('Accuracy', fontsize = 16)
ax.set_ylim(top = 1)
plt.title('Comparison of accuracy scores', fontsize = 18)

rects = ax.bar(names, acc, width = 0.7, color = 'forestgreen', edgecolor = '
darkgreen')

for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.05*height,
                '%.2f' % height,
                ha='center', va='bottom')

plt.show()
fig_acc.savefig('fig_acc', bbox_inches = 'tight')
```



Confusion matrix: columns(0,1) - number of real values y_test, rows(0,1) - number of predicted values y_pred

In [264]:

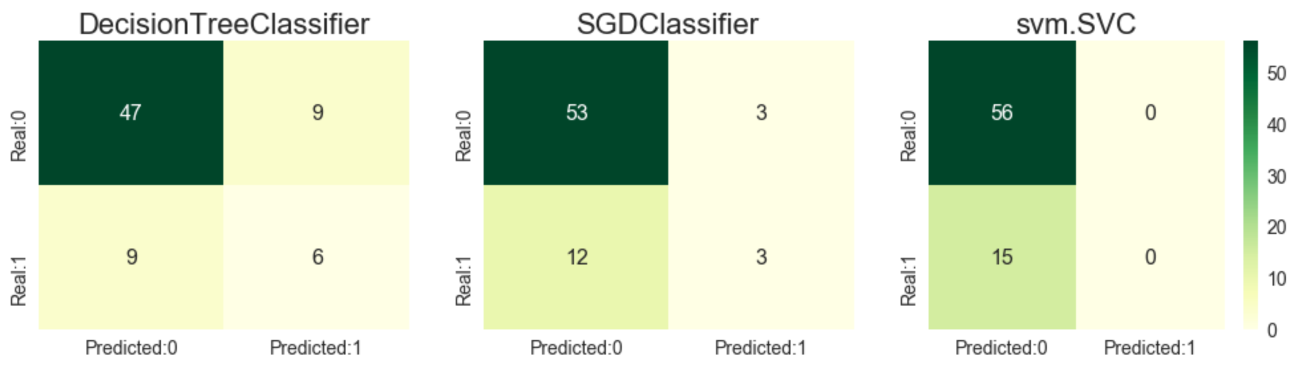
```
matrix1 = metrics.confusion_matrix(y_test, y_pred1)
matrix2 = metrics.confusion_matrix(y_test, y_pred2)
matrix3 = metrics.confusion_matrix(y_test, y_pred3)
cm1=pd.DataFrame(data=matrix1, columns=['Predicted:0','Predicted:1'],index=
['Real:0','Real:1'])
cm2=pd.DataFrame(data=matrix2, columns=['Predicted:0','Predicted:1'],index=
['Real:0','Real:1'])
cm3=pd.DataFrame(data=matrix3, columns=['Predicted:0','Predicted:1'],index=
['Real:0','Real:1'])
```

```
['Real:0', 'Real:1'])
```

In [265]:

```
fig7 = plt.figure(figsize = (17,4))
sns.set(font_scale=1.4)

plt.subplot(1,3,1)
plt.title('DecisionTreeClassifier', fontsize = 22)
sns.heatmap(cm1, annot=True, fmt='d',cmap="YlGn", annot_kws={"size": 16}, c
bar=False)

plt.subplot(1,3,2)
plt.title('SGDClassifier', fontsize = 22)
sns.heatmap(cm2, annot=True, fmt='d', cmap="YlGn", annot_kws={"size": 16},
cbar=False)

plt.subplot(1,3,3)
plt.title('svm.SVC', fontsize = 22)
sns.heatmap(cm3, annot=True, fmt='d', cmap="YlGn", annot_kws={"size": 16})
print (matrix1, '\n', matrix2, '\n', matrix3)

fig7.savefig('fig7', bbox_inches = 'tight')
```

```
[[47  9]
 [ 9  6]]
 [[53  3]
 [12  3]]
 [[56  0]
 [15  0]]
```



DecisionTreeClassifier:

True Positives: 6

True Negatives: 47

False Positives: 9 (Type I error)

False Negatives: 9 (Type II error)

In [237]:

```
print (sum(matrix1), '\n',sum(matrix2), '\n', sum(matrix3))
```

```
[56 15]
 [65  6]
 [71  0]
```