Kingdom of Saudi Arabia
Ministry of Education
Princess Nourah bint Abdulrahman University
Faculty of Computer and Information Sciences

Academic Year: 2021/2022
2nd Semester
Module: DCS 873-DIS 826
Program: MSc in computing
(Data Analytics)

Coronary Heart Disease Prediction - Codes File

It is a project that combines the basic concepts of Data Management and Visualisation and Machine learning.

Maram Alqahtani – Hissah AlKanhal
Supervised by: Prof. Souham Meshoul, Dr. Romana Aziz

Data Understanding

In [1]:
```
# hide warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# import libraries
import numpy as np
import pandas as pd
import statistics
import matplotlib.pyplot as plt
%matplotlib inline
import pylab as plt
import seaborn as sns
```

In [3]:
```
# loading data
df = pd.read_csv("Disease.csv" , sep=';')
del df['ind']
df.head()
```

Out[3]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160.0	12.00	5.73	23.11	Present	49	25.30	97.20	52	1
1	144.0	0.01	4.41	28.61	Absent	55	28.87	2.06	63	1
2	118.0	0.08	3.48	32.28	Present	52	29.14	3.81	46	0
3	170.0	7.50	6.41	38.03	Present	51	31.99	24.26	58	1
4	134.0	13.60	3.50	27.78	Present	60	25.99	57.34	49	1

In [4]:
```
df.tail()
```

Out[4]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
457	214.0	0.4	5.98	31.72	Absent	64	28.45	0.00	58	0
458	182.0	4.2	4.41	32.10	Absent	52	28.61	18.72	52	1
459	108.0	3.0	1.59	15.23	Absent	40	20.09	26.64	55	0
460	118.0	5.4	11.61	30.79	Absent	64	27.35	23.97	40	0
461	132.0	0.0	4.82	33.41	Present	62	14.70	0.00	46	1

Describe data

In [5]:
```
# checking dataset structure
df.shape
```

Out[5]:
```
(462, 10)
```

In [6]:
```
# print columns names
df.columns.values
```

Out[6]:
```
array(['sbp', 'tobacco', 'ldl', 'adiposity', 'famhist', 'typea',
       'obesity', 'alcohol', 'age', 'chd'], dtype=object)
```

In [7]:
```
# checking datatypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 462 entries, 0 to 461
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   sbp        461 non-null    float64
 1   tobacco    462 non-null    float64
 2   ldl        462 non-null    float64
 3   adiposity  461 non-null    float64
 4   famhist    462 non-null    object 
 5   typea      462 non-null    int64  
 6   obesity    462 non-null    float64
 7   alcohol    461 non-null    float64
 8   age        462 non-null    int64  
 9   chd        462 non-null    int64  
dtypes: float64(6), int64(3), object(1)
memory usage: 36.2+ KB
```

In [8]:
```
df.index
```

Out[8]:
```
RangeIndex(start=0, stop=462, step=1)
```

Verify Data Quality

1. Accuracy

In [9]:
```
# sbp
print(df['sbp'].max())
print(df['sbp'].min())
# tobacco
print(df['tobacco'].max())
print(df['tobacco'].min())
# LDL
print(df['ldl'].max())
print(df['ldl'].min())
# adiposity
print(df['adiposity'].max())
print(df['adiposity'].min())
# typea
print(df['typea'].max())
print(df['typea'].min())
# obesity
print(df['obesity'].max())
print(df['obesity'].min())
# obesity
print(df['alcohol'].max())
print(df['alcohol'].min())
# age
print(df['age'].max())
print(df['age'].min())
```

```
218.0
101.0
31.2
0.0
15.33
0.98
42.49
6.74
78
13
46.58
14.7
147.19
0.0
64
15
```

2. Completeness

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
sbp           1
tobacco       0
ldl           0
adiposity     1
famhist       0
typea         0
obesity       0
alcohol       1
age           0
chd           0
dtype: int64
```

3. Consistency

In [11]:
```
print(df['famhist'].unique())
print(df['chd'].unique())
```

```
['Present' 'Absent']
[1 0]
```

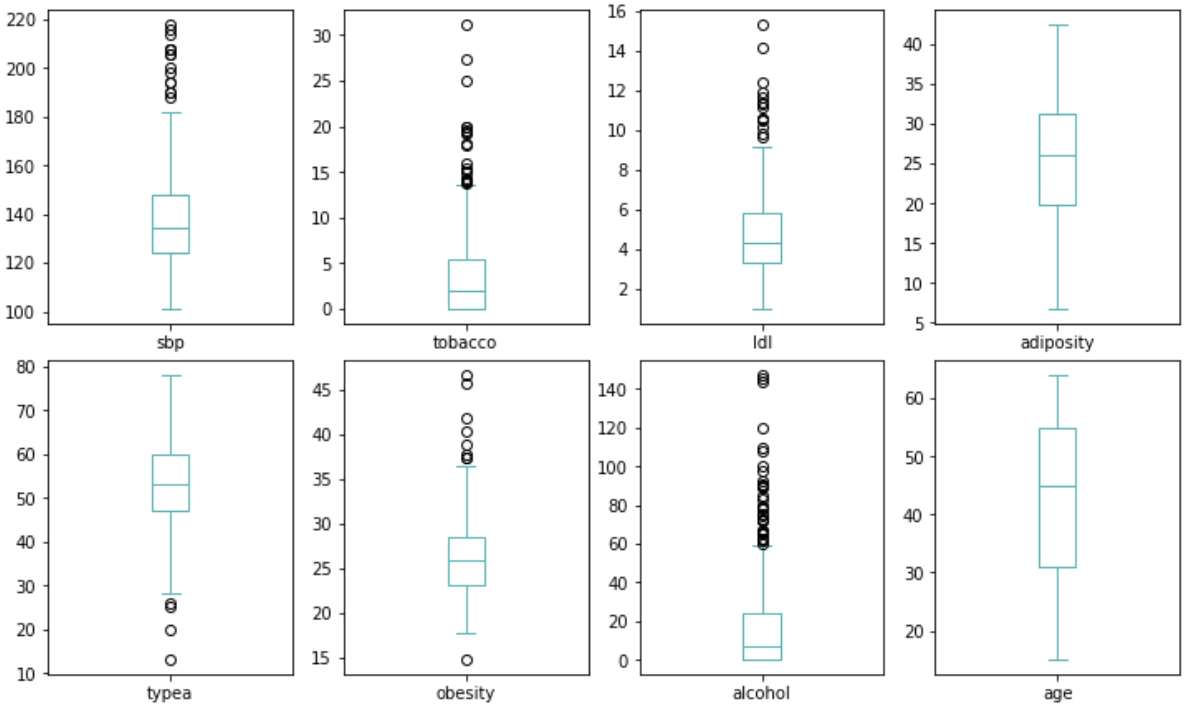
4. Validity

In [12]:
```
# desribe data, to show features details
df.describe()
```

Out[12]:

	sbp	tobacco	ldl	adiposity	typea	obesity	alco
count	461.000000	462.000000	462.000000	461.000000	462.000000	462.000000	461.000(
mean	138.396963	3.635649	4.740325	25.435119	53.103896	26.044113	17.052:
std	20.463029	4.593024	2.070909	7.765163	9.817534	4.213680	24.507(
min	101.000000	0.000000	0.980000	6.740000	13.000000	14.700000	0.000(
25%	124.000000	0.052500	3.282500	19.850000	47.000000	22.985000	0.510(
50%	134.000000	2.000000	4.340000	26.130000	53.000000	25.805000	7.410(
75%	148.000000	5.500000	5.790000	31.290000	60.000000	28.497500	23.970(
max	218.000000	31.200000	15.330000	42.490000	78.000000	46.580000	147.190(

In [13]:
```
plt.subplot(241)
df['sbp'].plot.box(figsize=(10,6), color ='#5ab3b4' )
plt.tight_layout(pad=0.5)
plt.subplot(242)
df['tobacco'].plot.box( color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(243)
df['ldl'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(244)
df['adiposity'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)

plt.subplot(245)
df['typea'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(246)
df['obesity'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(247)
df['alcohol'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(248)
df['age'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
```



In [14]:
```
#create a function to find outliers using IQR

def find_outliers_IQR(df):

   q1=df.quantile(0.25)

   q3=df.quantile(0.75)

   IQR=q3-q1

   outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]

   return outliers
#sbp
```

```
outliers = find_outliers_IQR(df['sbp'])
print('number of sbp outliers: '+ str(len(outliers)))
print('max outlier value: '+ str(outliers.max()))
print('min outlier value: '+ str(outliers.min()))
#tobacco
outliers = find_outliers_IQR(df['tobacco'])
print('number of tobacco outliers: '+ str(len(outliers)))
print('max outlier value: '+ str(outliers.max()))
print('min outlier value: '+ str(outliers.min()))
#ldl
outliers = find_outliers_IQR(df['ldl'])
print('number of ldl outliers: '+ str(len(outliers)))
print('max outlier value: '+ str(outliers.max()))
print('min outlier value: '+ str(outliers.min()))
#adiposity
outliers = find_outliers_IQR(df['adiposity'])
print('number of adiposity outliers: '+ str(len(outliers)))
print('max outlier value: '+ str(outliers.max()))
print('min outlier value: '+ str(outliers.min()))
#typea
outliers = find_outliers_IQR(df['typea'])
print('number of typea outliers: '+ str(len(outliers)))
print('max outlier value: '+ str(outliers.max()))
print('min outlier value: '+ str(outliers.min()))
#obesity
outliers = find_outliers_IQR(df['obesity'])
print('number of obesity outliers: '+ str(len(outliers)))
print('max outlier value: '+ str(outliers.max()))
print('min outlier value: '+ str(outliers.min()))
#alcohol
outliers = find_outliers_IQR(df['alcohol'])
print('number of alcohol outliers: '+ str(len(outliers)))
print('max outlier value: '+ str(outliers.max()))
print('min outlier value: '+ str(outliers.min()))
#age
outliers = find_outliers_IQR(df['age'])
print('number of age outliers: '+ str(len(outliers)))
print('max outlier value: '+ str(outliers.max()))
print('min outlier value: '+ str(outliers.min()))
```

```
number of sbp outliers: 15
max outlier value: 218.0
min outlier value: 188.0
number of tobacco outliers: 19
max outlier value: 31.2
min outlier value: 13.8
number of ldl outliers: 14
max outlier value: 15.33
min outlier value: 9.65
number of adiposity outliers: 0
max outlier value: nan
min outlier value: nan
number of typea outliers: 4
max outlier value: 26
min outlier value: 13
number of obesity outliers: 9
max outlier value: 46.58
min outlier value: 14.7
number of alcohol outliers: 32
max outlier value: 147.19
min outlier value: 59.79
number of age outliers: 0
max outlier value: nan
min outlier value: nan
```

In [15]:

```
#sbp
Q1 = df['sbp'].quantile(0.25)
Q3 = df['sbp'].quantile(0.75)
IQR = Q3 - Q1
print('sbp IQR= '+str(IQR))
Lower_Fence = Q1 - (1.5 * IQR)
Upper_Fence = Q3 + (1.5 * IQR)
print('Lower inner fence= '+str(Lower_Fence))
print('Upeer inner fence = '+str(Upper_Fence))
Lower_Fence = Q1 - (3 * IQR)
Upper_Fence = Q3 + (3 * IQR)
print('Lower outer fence= '+str(Lower_Fence))
print('Upper outer fence='+str(Upper_Fence))
```

```
sbp IQR= 24.0
Lower inner fence= 88.0
Upeer inner fence = 184.0
Lower outer fence= 52.0
Upper outer fence=220.0
```

In [16]:

```
#tobacco
Q1 = df['tobacco'].quantile(0.25)
Q3 = df['tobacco'].quantile(0.75)
IQR = Q3 - Q1
print('tobacco IQR= '+str(IQR))
Lower_Fence = Q1 - (1.5 * IQR)
Upper_Fence = Q3 + (1.5 * IQR)
print('Lower inner fence= '+str(Lower_Fence))
print('Upeer inner fence = '+str(Upper_Fence))
Lower_Fence = Q1 - (3 * IQR)
Upper_Fence = Q3 + (3 * IQR)
print('Lower outer fence= '+str(Lower_Fence))
print('Upper outer fence= '+str(Upper_Fence))
```

```
tobacco IQR= 5.4475
Lower inner fence= -8.11875
Upeer inner fence = 13.67125
Lower outer fence= -16.290000000000003
Upper outer fence= 21.8425
```

In [17]:

```
#ldl
Q1 = df['ldl'].quantile(0.25)
Q3 = df['ldl'].quantile(0.75)
IQR = Q3 - Q1
print('ldl IQR= '+str(IQR))
Lower_Fence = Q1 - (1.5 * IQR)
Upper_Fence = Q3 + (1.5 * IQR)
print('Lower inner fence= '+str(Lower_Fence))
print('Upeer inner fence = '+str(Upper_Fence))
Lower_Fence = Q1 - (3 * IQR)
Upper_Fence = Q3 + (3 * IQR)
print('Lower outer fence= '+str(Lower_Fence))
print('Upper outer fence= '+str(Upper_Fence))
```

```
ldl IQR= 2.5075000000000003
Lower inner fence= -0.4787500000000007
Upeer inner fence = 9.55125
Lower outer fence= -4.240000000000001
Upper outer fence= 13.3125
```

In [18]:

```
#adiposity
```

```
Q1 = df['adiposity'].quantile(0.25)
Q3 = df['adiposity'].quantile(0.75)
IQR = Q3 - Q1
print('adiposity IQR= '+str(IQR))
Lower_Fence = Q1 - (1.5 * IQR)
Upper_Fence = Q3 + (1.5 * IQR)
print('Lower inner fence= '+str(Lower_Fence))
print('Upeer inner fence = '+str(Upper_Fence))
Lower_Fence = Q1 - (3 * IQR)
Upper_Fence = Q3 + (3 * IQR)
print('Lower outer fence= '+str(Lower_Fence))
print('Upper outer fence= '+str(Upper_Fence))
```

```
adiposity IQR= 11.439999999999998
Lower inner fence= 2.690000000000005
Upeer inner fence = 48.449999999999996
Lower outer fence= -14.469999999999992
Upper outer fence= 65.60999999999999
```

In [19]:

```
#typea
Q1 = df['typea'].quantile(0.25)
Q3 = df['typea'].quantile(0.75)
IQR = Q3 - Q1
print('typea IQR= '+str(IQR))
Lower_Fence = Q1 - (1.5 * IQR)
Upper_Fence = Q3 + (1.5 * IQR)
print('Lower inner fence= '+str(Lower_Fence))
print('Upeer inner fence = '+str(Upper_Fence))
Lower_Fence = Q1 - (3 * IQR)
Upper_Fence = Q3 + (3 * IQR)
print('Lower outer fence= '+str(Lower_Fence))
print('Upper outer fence= '+str(Upper_Fence))
```

```
typea IQR= 13.0
Lower inner fence= 27.5
Upeer inner fence = 79.5
Lower outer fence= 8.0
Upper outer fence= 99.0
```

In [20]:

```
#obesity
Q1 = df['obesity'].quantile(0.25)
Q3 = df['obesity'].quantile(0.75)
IQR = Q3 - Q1
print('obesity IQR= '+str(IQR))
Lower_Fence = Q1 - (1.5 * IQR)
Upper_Fence = Q3 + (1.5 * IQR)
print('Lower inner fence= '+str(Lower_Fence))
print('Upeer inner fence = '+str(Upper_Fence))
Lower_Fence = Q1 - (3 * IQR)
Upper_Fence = Q3 + (3 * IQR)
print('Lower outer fence= '+str(Lower_Fence))
print('Upper outer fence= '+str(Upper_Fence))
```

```
obesity IQR= 5.512499999999999
Lower inner fence= 14.71625
Upeer inner fence = 36.76625
Lower outer fence= 6.447500000000002
Upper outer fence= 45.035
```

In [21]:

```
#alcohol
Q1 = df['alcohol'].quantile(0.25)
Q3 = df['alcohol'].quantile(0.75)
```

```
IQR = Q3 - Q1
print('alcohol IQR= '+str(IQR))
Lower_Fence = Q1 - (1.5 * IQR)
Upper_Fence = Q3 + (1.5 * IQR)
print('Lower inner fence= '+str(Lower_Fence))
print('Upeer inner fence = '+str(Upper_Fence))
Lower_Fence = Q1 - (3 * IQR)
Upper_Fence = Q3 + (3 * IQR)
print('Lower outer fence= '+str(Lower_Fence))
print('Upper outer fence= '+str(Upper_Fence))
```

```
alcohol IQR= 23.459999999999997
Lower inner fence= -34.68
Upeer inner fence = 59.16
Lower outer fence= -69.86999999999999
Upper outer fence= 94.35
```

In [22]:

```
#age
Q1 = df['age'].quantile(0.25)
Q3 = df['age'].quantile(0.75)
IQR = Q3 - Q1
print('age IQR= '+str(IQR))
Lower_Fence = Q1 - (1.5 * IQR)
Upper_Fence = Q3 + (1.5 * IQR)
print('Lower inner fence= '+str(Lower_Fence))
print('Upeer inner fence = '+str(Upper_Fence))
Lower_Fence = Q1 - (3 * IQR)
Upper_Fence = Q3 + (3 * IQR)
print('Lower outer fence= '+str(Lower_Fence))
print('Upper outer fence= '+str(Upper_Fence))
```

```
age IQR= 24.0
Lower inner fence= -5.0
Upeer inner fence = 91.0
Lower outer fence= -41.0
Upper outer fence= 127.0
```

5. Duplication

In [23]:

```
df.duplicated().sum()
```

Out[23]:

```
0
```

Data Cleaning

In [1]:
```
# hide warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# import libraries
import numpy as np
import pandas as pd
import statistics
import matplotlib.pyplot as plt
%matplotlib inline
import pylab as plt
import seaborn as sns
```

In [3]:
```
# loading data
df = pd.read_csv("Disease.csv" , sep=';')
del df['ind']
df.head()
```

Out[3]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160.0	12.00	5.73	23.11	Present	49	25.30	97.20	52	1
1	144.0	0.01	4.41	28.61	Absent	55	28.87	2.06	63	1
2	118.0	0.08	3.48	32.28	Present	52	29.14	3.81	46	0
3	170.0	7.50	6.41	38.03	Present	51	31.99	24.26	58	1
4	134.0	13.60	3.50	27.78	Present	60	25.99	57.34	49	1

1. Dealing with missing values

In [4]:
```
missing_count = df.isnull().sum()
percent_missing = df.isnull().sum() * 100 / len(df)
missing_value_df = pd.DataFrame({'missing count:' : missing_count ,
                                 'missing percent': percent_missing})
missing_value_df
```

Out[4]:

	missing count:	missing percent
sbp	1	0.21645
tobacco	0	0.00000
ldl	0	0.00000
adiposity	1	0.21645
famhist	0	0.00000
typea	0	0.00000
obesity	0	0.00000
alcohol	1	0.21645
age	0	0.00000
chd	0	0.00000

In [5]:
```
df = df.dropna(axis=0)
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
sbp           0
tobacco       0
ldl           0
adiposity     0
famhist       0
typea         0
obesity       0
alcohol       0
age           0
chd           0
dtype: int64
```

2. Dealing with Irregular Data (Outliers)

In [7]:
```
#sbp
df["sbp"] = df["sbp"].map(
    lambda x: 184.0 if x > 184.0 else x
)
```

In [8]:
```
#tobacco
df["tobacco"] = df["tobacco"].map(
    lambda x: 13.67125 if x > 13.67125 else x
)
```

In [9]:
```
#LDL
df["ldl"] = df["ldl"].map(
    lambda x: 9.55125 if x > 9.55125 else x
)
```

In [10]:
```
#typea
df["typea"] = df["typea"].map(
    lambda x: 27.5 if x < 27.5 else x
)
```

In [11]:

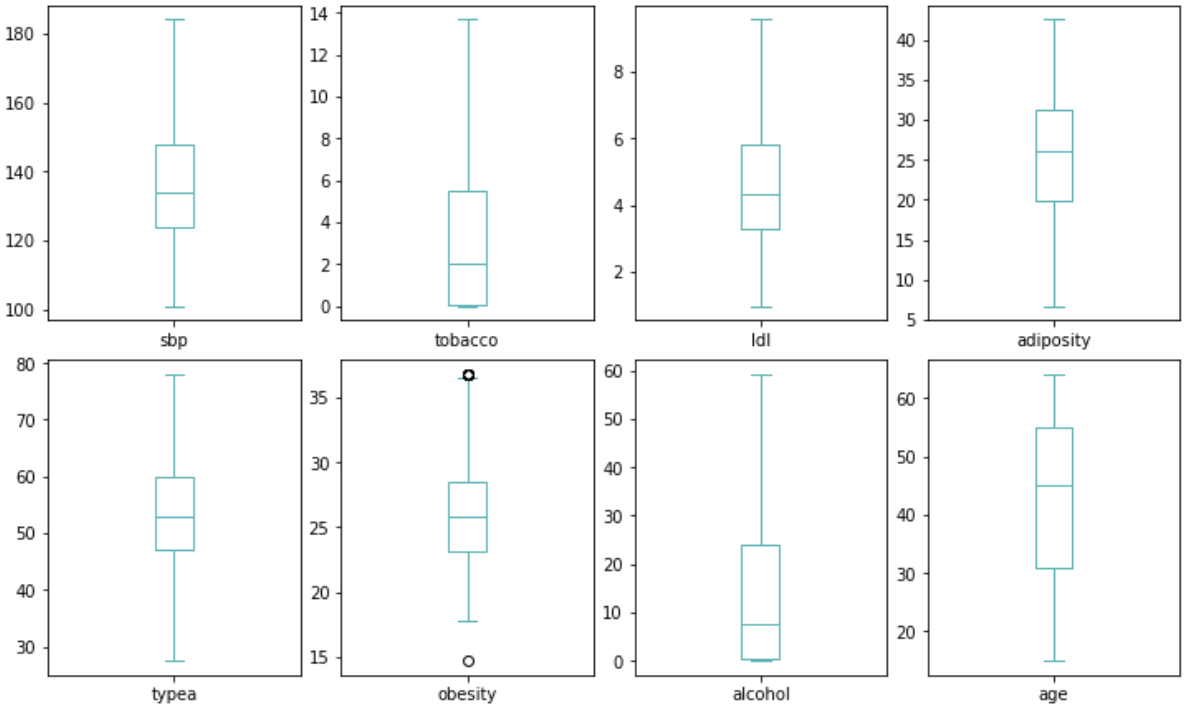
```
#obesity
df["obesity"] = df["obesity"].map(
    lambda x: 36.76625 if x > 36.76625 else x
)

df["obesity"] = df["obesity"].map(
    lambda x: 14.71625 if x < 14.71625 else x
)
```

In [12]:

```
#alcohol
df["alcohol"] = df["alcohol"].map(
    lambda x: 59.16 if x > 59.16 else x
)
```

In [13]:

```
fig = plt.figure(figsize =(10, 7))

plt.subplot(241)
df['sbp'].plot.box(figsize=(10,6), color ='#5ab3b4' )
plt.tight_layout(pad=0.5)
plt.subplot(242)
df['tobacco'].plot.box( color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(243)
df['ldl'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(244)
df['adiposity'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)

plt.subplot(245)
df['typea'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(246)
df['obesity'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(247)
df['alcohol'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
plt.subplot(248)
df['age'].plot.box(color ='#5ab3b4')
plt.tight_layout(pad=0.5)
```



In [14]:

```
df.to_csv('cleaned_disease.csv',index=False)
```

Univarite Analysis

In [1]:
```
# hide warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# import libraries
import numpy as np
import pandas as pd
import statistics
import matplotlib.pyplot as plt
%matplotlib inline
import pylab as plt
import seaborn as sns
```

In [3]:
```
# loading data
df = pd.read_csv("cleaned_disease.csv")
df.head()
```

Out[3]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160.0	12.00	5.73	23.11	Present	49.0	25.30	59.16	52	1
1	144.0	0.01	4.41	28.61	Absent	55.0	28.87	2.06	63	1
2	118.0	0.08	3.48	32.28	Present	52.0	29.14	3.81	46	0
3	170.0	7.50	6.41	38.03	Present	51.0	31.99	24.26	58	1
4	134.0	13.60	3.50	27.78	Present	60.0	25.99	57.34	49	1

In [4]:
```
# checking dataset structure
df.shape
```

Out[4]:
```
(461, 10)
```

Continuous Features

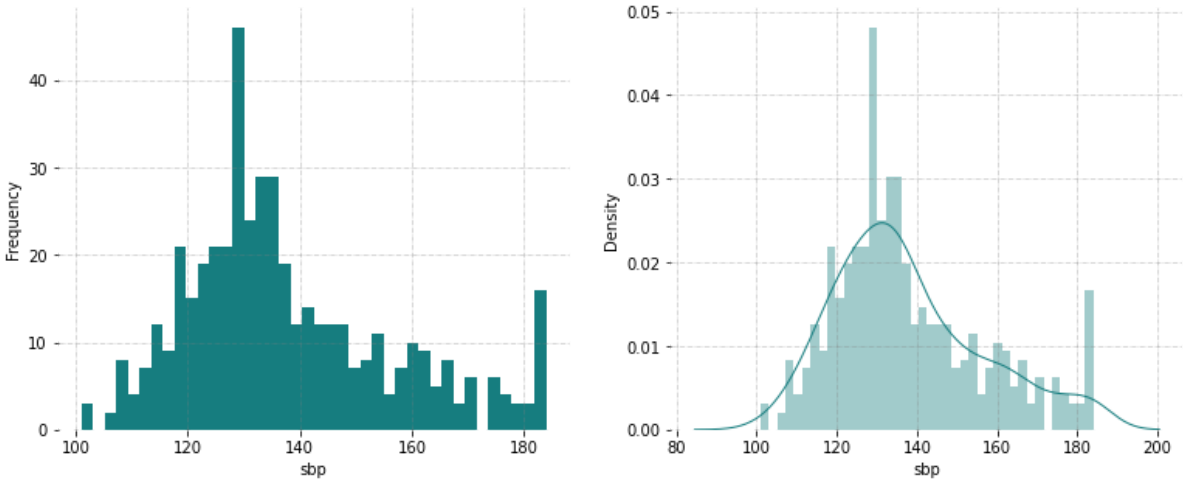
1. sbp Feature

In [5]:
```
# calculating summary statistics for sbp
summary_statistics = df['sbp'].describe()
summary_statistics['mode'] = df['sbp'].mode()[0]
summary_statistics['variance'] = df['sbp'].var()
summary_statistics['skewness'] = df['sbp'].skew()
summary_statistics['range'] =  np.ptp(df['sbp'])
summary_statistics['sum'] = df['sbp'].sum()
print('Summary Statistics\n',summary_statistics)
```

```
Summary Statistics
 count         461.000000
mean           137.793926
std             18.683835
min            101.000000
25%            124.000000
50%            134.000000
75%            148.000000
max            184.000000
mode           134.000000
variance       349.085702
skewness         0.720674
range           83.000000
sum          63523.000000
Name: sbp, dtype: float64
```

In [6]:

```
fig = plt.figure(figsize =(13, 5))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
n, bins, patches = ax1.hist(df['sbp'], bins=40, color='#167D7F')
ax1.set_xlabel('sbp')
ax1.set_ylabel('Frequency')

ax2.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
ax2 = sns.distplot(df['sbp'], hist=True, kde=True, bins=40, color='#167D7F'
ax2.set_xlabel('sbp')
ax2.set_ylabel('Density')
plt.savefig('sbp.jpg')
plt.show()
```



2.Tobacco Feature

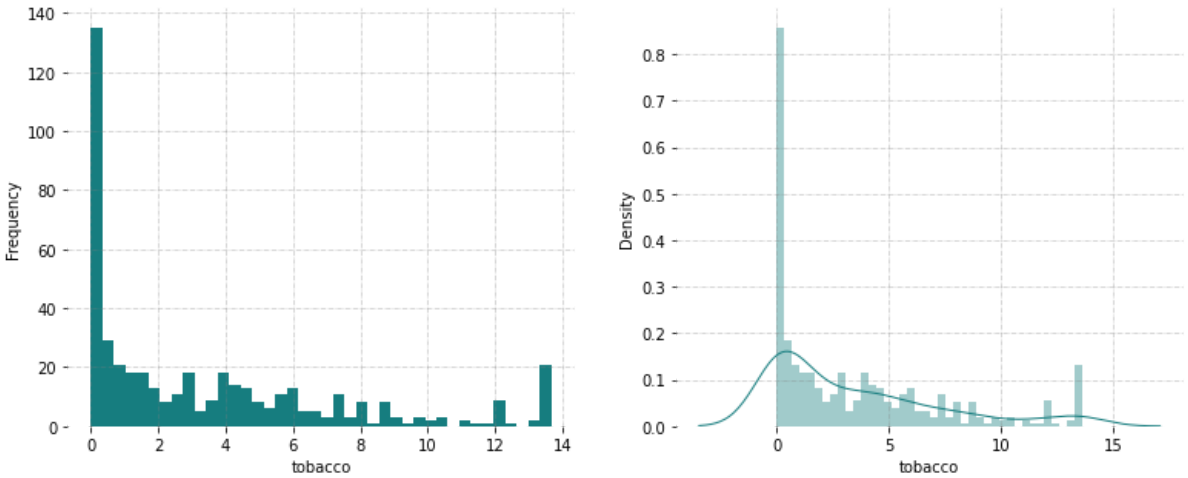
In [7]:

```
# calculating summary statistics for tobacco
summary_statistics = df['tobacco'].describe()
summary_statistics['mode'] = df['tobacco'].mode()[0]
summary_statistics['variance'] = df['tobacco'].var()
summary_statistics['skewness'] = df['tobacco'].skew()
summary_statistics['range'] =  np.ptp(df['tobacco'])
summary_statistics['sum'] = df['tobacco'].sum()
print('Summary Statistics\n',summary_statistics)
```

```
Summary Statistics
 count         461.000000
mean            3.444585
std             3.911996
min             0.000000
25%             0.050000
50%             2.000000
75%             5.500000
max            13.671250
mode            0.000000
variance       15.303716
skewness        1.196790
range          13.671250
sum          1587.953750
Name: tobacco, dtype: float64
```

In [8]:

```
fig = plt.figure(figsize =(13, 5))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
n, bins, patches = ax1.hist(df['tobacco'], bins=40, color='#167D7F')
ax1.set_xlabel('tobacco')
ax1.set_ylabel('Frequency')

ax2.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
ax2 = sns.distplot(df['tobacco'], hist=True, kde=True, bins=40, color='#167
ax2.set_xlabel('tobacco')
ax2.set_ylabel('Density')
plt.savefig('tobacco.jpg')
plt.show()
```



3. ldl Feature

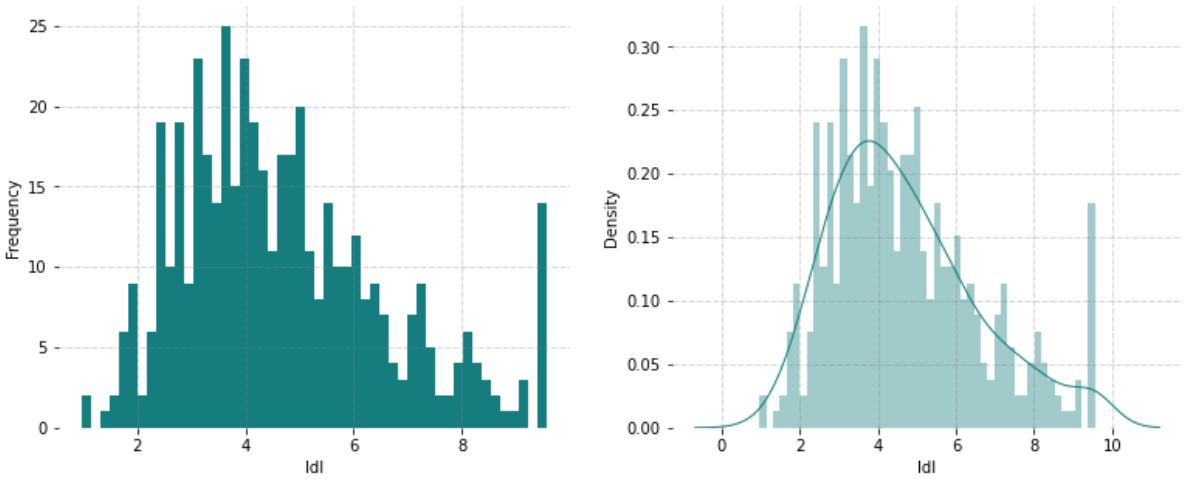
In [9]:

```
# calculating summary statistics for ldl
summary_statistics = df['ldl'].describe()
summary_statistics['mode'] = df['ldl'].mode()[0]
summary_statistics['variance'] = df['ldl'].var()
summary_statistics['skewness'] = df['ldl'].skew()
summary_statistics['range'] =  np.ptp(df['ldl'])
summary_statistics['sum'] = df['ldl'].sum()
print('Summary Statistics\n',summary_statistics)
```

```
Summary Statistics
 count         461.000000
mean            4.688671
std             1.878779
min             0.980000
25%             3.290000
50%             4.340000
75%             5.800000
max             9.551250
mode            9.551250
variance        3.529810
skewness        0.719637
range           8.571250
sum          2161.477500
Name: ldl, dtype: float64
```

In [10]:

```
fig = plt.figure(figsize =(13, 5))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
n, bins, patches = ax1.hist(df['ldl'], bins=50, color='#167D7F')
ax1.set_xlabel('ldl')
ax1.set_ylabel('Frequency')

ax2.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
ax2 = sns.distplot(df['ldl'], hist=True, kde=True, bins=50, color='#167D7F'
ax2.set_xlabel('ldl')
ax2.set_ylabel('Density')
plt.savefig('ldl.jpg')
plt.show()
```



4. Adiposity Feature

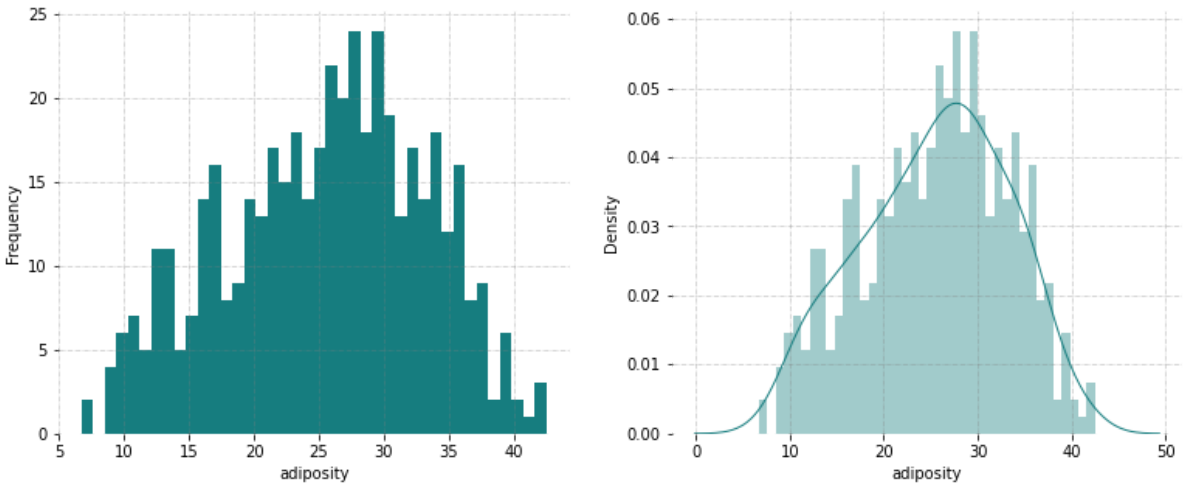
In [11]:

```
# calculating summary statistics for adiposity
summary_statistics = df['adiposity'].describe()
summary_statistics['mode'] = df['adiposity'].mode()[0]
summary_statistics['variance'] = df['adiposity'].var()
summary_statistics['skewness'] = df['adiposity'].skew()
summary_statistics['range'] =  np.ptp(df['adiposity'])
summary_statistics['sum'] = df['adiposity'].sum()
print('Summary Statistics\n',summary_statistics)
```

```
Summary Statistics
 count         461.000000
mean           25.435119
std             7.765163
min             6.740000
25%            19.850000
50%            26.130000
75%            31.290000
max            42.490000
mode           21.100000
variance       60.297760
skewness       -0.216972
range          35.750000
sum         11725.590000
Name: adiposity, dtype: float64
```

In [12]:

```
fig = plt.figure(figsize =(13, 5))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
n, bins, patches = ax1.hist(df['adiposity'], bins=40, color='#167D7F')
ax1.set_xlabel('adiposity')
ax1.set_ylabel('Frequency')

ax2.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
ax2 = sns.distplot(df['adiposity'], hist=True, kde=True, bins=40, color='#1
ax2.set_xlabel('adiposity')
ax2.set_ylabel('Density')
plt.savefig('adiposity.jpg')
plt.show()
```



5. Typea Feature

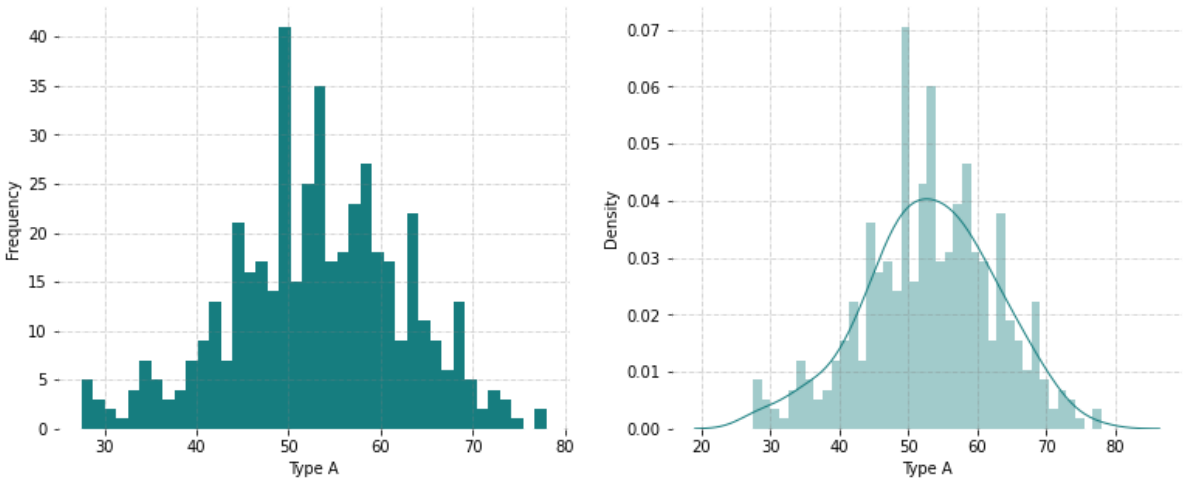
In [13]:

```
# calculating summary statistics for typea
summary_statistics = df['typea'].describe()
summary_statistics['mode'] = df['typea'].mode()[0]
summary_statistics['variance'] = df['typea'].var()
summary_statistics['skewness'] = df['typea'].skew()
summary_statistics['range'] =  np.ptp(df['typea'])
summary_statistics['sum'] =  df['typea'].sum()
print('Summary Statistics\n',summary_statistics)
```

```
Summary Statistics
 count          461.000000
mean            53.114967
std              9.599582
min             27.500000
25%             47.000000
50%             53.000000
75%             60.000000
max             78.000000
mode            52.000000
variance        92.151971
skewness        -0.210424
range           50.500000
sum          24486.000000
Name: typea, dtype: float64
```

In [14]:

```
fig = plt.figure(figsize =(13, 5))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
n, bins, patches = ax1.hist(df['typea'], bins=40, color='#167D7F')
ax1.set_xlabel('Type A')
ax1.set_ylabel('Frequency')

ax2.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
ax2 = sns.distplot(df['typea'], hist=True, kde=True, bins=40, color='#167D7
ax2.set_xlabel('Type A')
ax2.set_ylabel('Density')

plt.show()
```



6. Obesity Feature

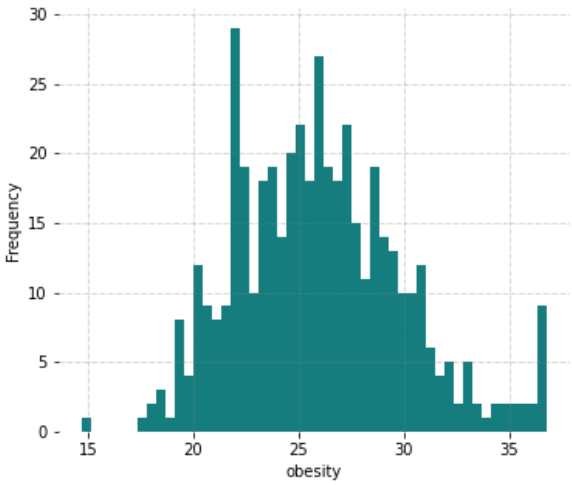
In [15]:

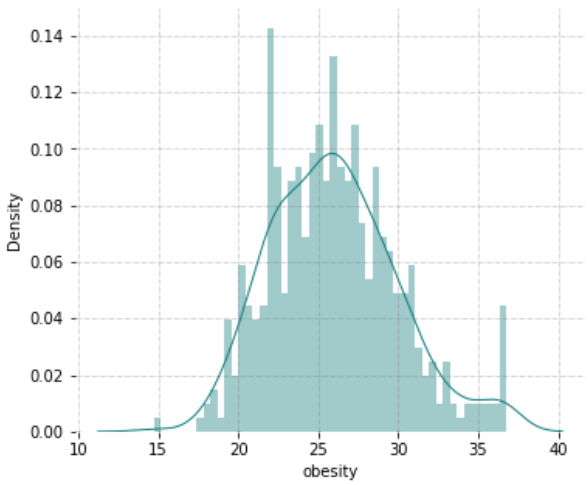
```
# calculating summary statistics for typea
summary_obesity = df['obesity'].describe()
summary_obesity['mode'] = df['obesity'].mode()[0]
summary_obesity['variance'] = df['obesity'].var()
summary_obesity['skewness'] = df['obesity'].skew()
summary_obesity['range'] =  np.ptp(df['obesity'])
summary_obesity['sum'] =  df['obesity'].sum()
print('Summary Statistics\n',summary_obesity)
```

```
Summary Statistics
 count          461.000000
mean            25.987085
std              3.973589
min             14.716250
25%             23.090000
50%             25.810000
75%             28.500000
max             36.766250
mode            36.766250
variance        15.789412
skewness         0.442454
range           22.050000
sum          11980.046250
Name: obesity, dtype: float64
```

In [16]:

```
fig = plt.figure(figsize =(13, 5))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
n, bins, patches = ax1.hist(df['obesity'], bins=50, color='#167D7F')
ax1.set_xlabel('obesity')
ax1.set_ylabel('Frequency')

ax2.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
ax2 = sns.distplot(df['obesity'], hist=True, kde=True, bins=50, color='#167
ax2.set_xlabel('obesity')
ax2.set_ylabel('Density')

plt.show()
```





7. Alcohol

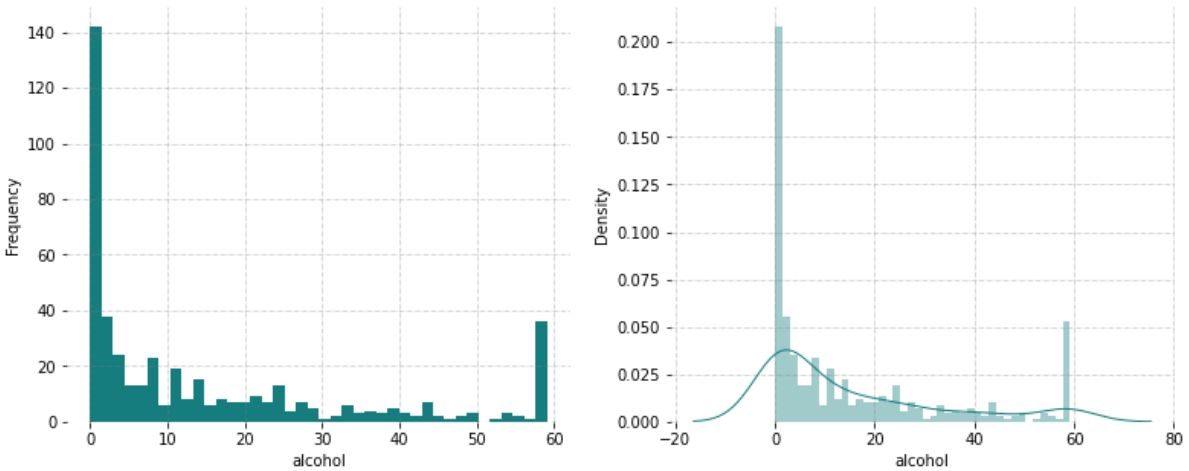
In [17]:

```
# calculating summary statistics for typea
summary_alcohol = df['alcohol'].describe()
summary_alcohol['mode'] = df['alcohol'].mode()[0]
summary_alcohol['variance'] = df['alcohol'].var()
summary_alcohol['skewness'] = df['alcohol'].skew()
summary_alcohol['range'] =  df['alcohol'].max() - df['alcohol'].min()
summary_alcohol['sum'] =  df['alcohol'].sum()
print('Summary Statistics\n',summary_alcohol)
```

```
Summary Statistics
 count        461.000000
mean          15.144165
std           18.517020
min            0.000000
25%            0.510000
50%            7.410000
75%           23.970000
max           59.160000
mode           0.000000
variance     342.880042
skewness       1.250455
range         59.160000
sum         6981.460000
Name: alcohol, dtype: float64
```

In [18]:

```
fig = plt.figure(figsize =(13, 5))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
n, bins, patches = ax1.hist(df['alcohol'], bins=40, color='#167D7F')
ax1.set_xlabel('alcohol')
ax1.set_ylabel('Frequency')

ax2.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
ax2 = sns.distplot(df['alcohol'], hist=True, kde=True, bins=40, color='#167
ax2.set_xlabel('alcohol')
ax2.set_ylabel('Density')

plt.show()
```



8. Age

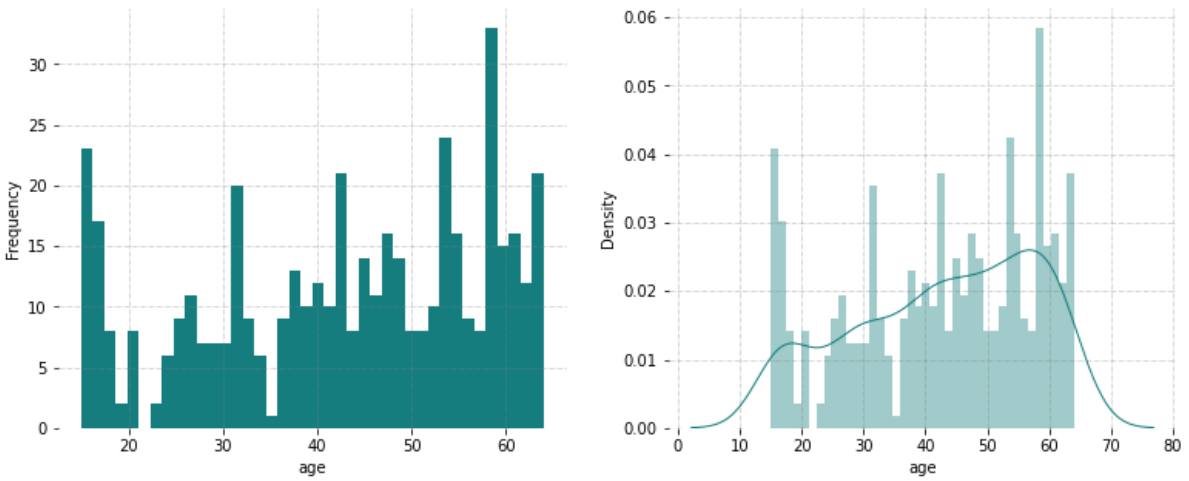
In [19]:

```
# calculating summary statistics for age
summary_age = df['age'].describe()
summary_age['mode'] = df['age'].mode()[0]
summary_age['variance'] = df['age'].var()
summary_age['skewness'] = df['age'].skew()
summary_age['range'] =  df['age'].max() - df['age'].min()
summary_age['sum'] =  df['age'].sum()
print('Summary Statistics\n',summary_age)
```

```
Summary Statistics
 count          461.000000
mean            42.865510
std             14.586001
min             15.000000
25%             31.000000
50%             45.000000
75%             55.000000
max             64.000000
mode            16.000000
variance       212.751438
skewness        -0.386244
range           49.000000
sum          19761.000000
Name: age, dtype: float64
```

In [20]:

```
fig = plt.figure(figsize =(13, 5))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)
    ax2.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
n, bins, patches = ax1.hist(df['age'], bins=40, color='#167D7F')
ax1.set_xlabel('age')
ax1.set_ylabel('Frequency')

ax2.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
ax2 = sns.distplot(df['age'], hist=True, kde=True, bins=40, color='#167D7F'
ax2.set_xlabel('age')
ax2.set_ylabel('Density')

plt.show()
```



Categorical Features

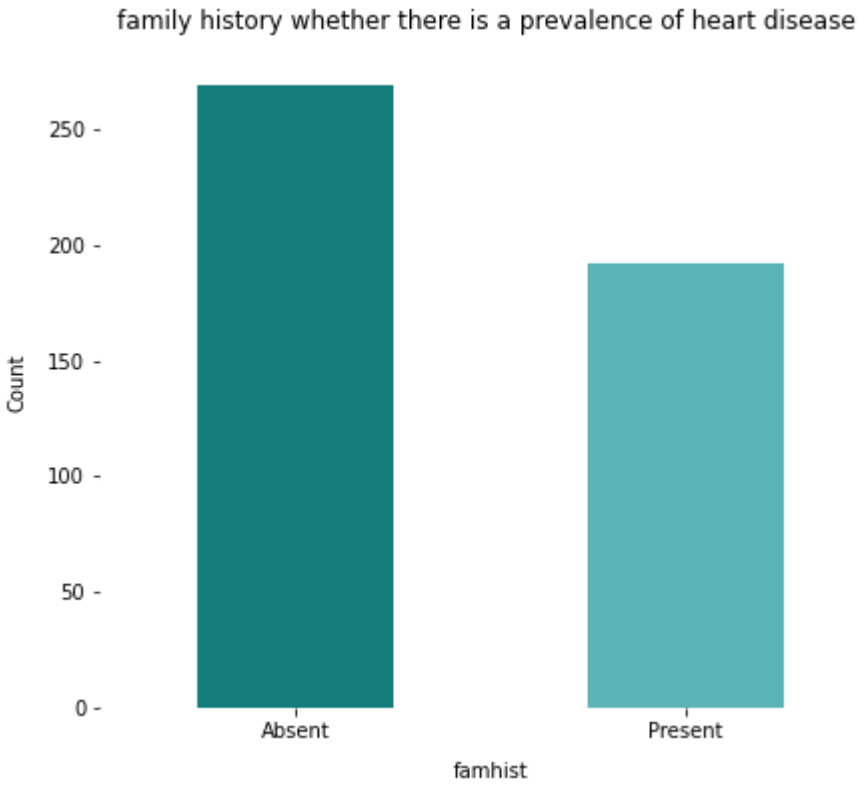
9. Famhist Feature

In [21]:

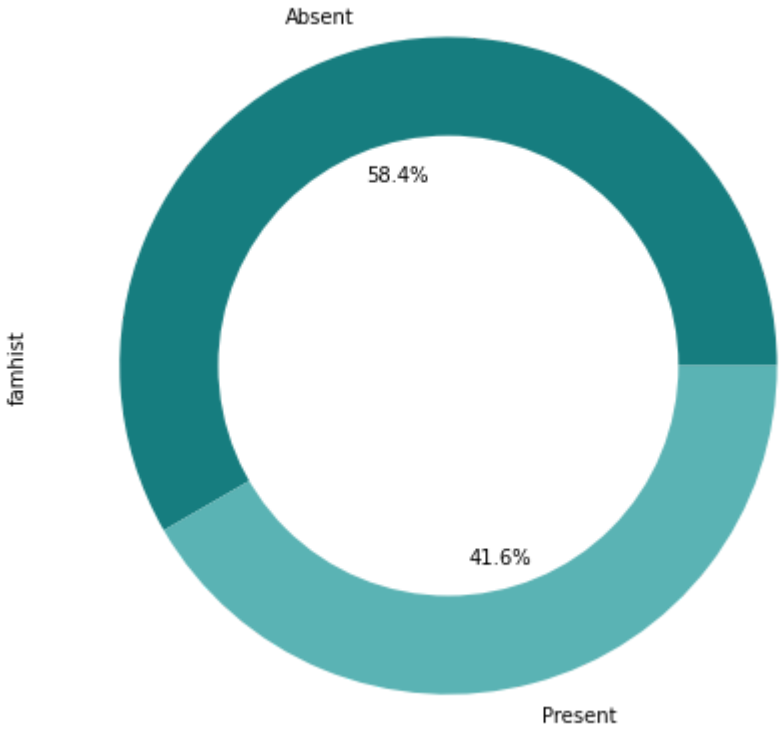
```
# calculating summary statistics
df['famhist'].value_counts()
```

Out[21]:
```
Absent     269
Present    192
Name: famhist, dtype: int64
```

In [22]:
```
fig = plt.figure(figsize =(13, 5))
ax1 = fig.add_subplot(1, 1, 1)
for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.7, alpha =
ax1 = df['famhist'].value_counts().plot(kind='bar', figsize=(7, 6), rot=0,c
plt.xlabel("famhist", labelpad=10)
plt.ylabel("Count", labelpad=10)
plt.title("family history whether there is a prevalence of heart disease ",
plt.savefig('famhist.jpg')
```



In [23]:
```
df['famhist'].value_counts().plot(kind='pie', figsize=(7, 6),colors=['#167D
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
ax1.axis('equal')
plt.tight_layout()
plt.savefig('famhist2.jpg')
plt.show()
```



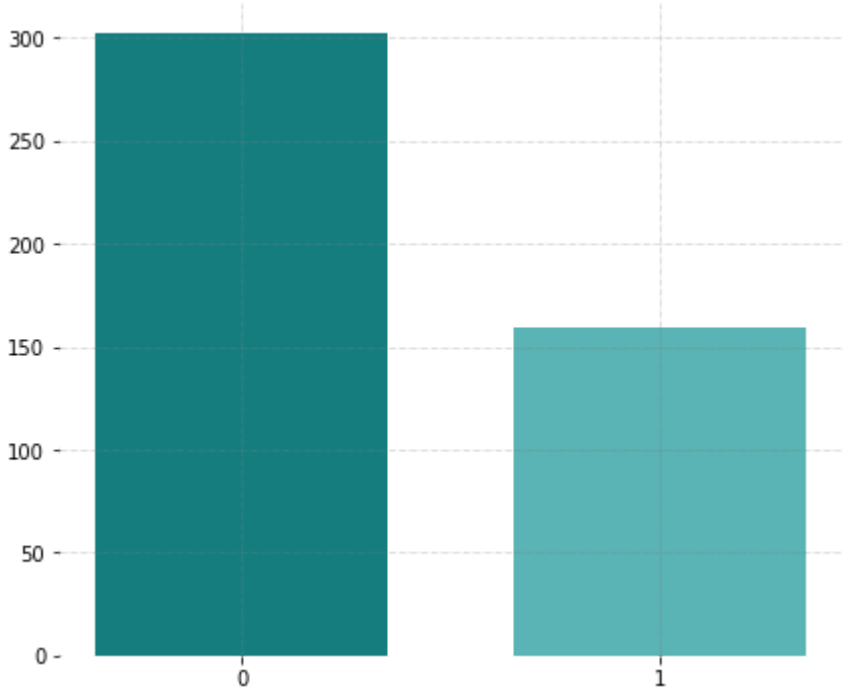
Target Feature

10. CHD

In [24]:
```
# calculating summary statistics for age
df['chd'].value_counts()
```

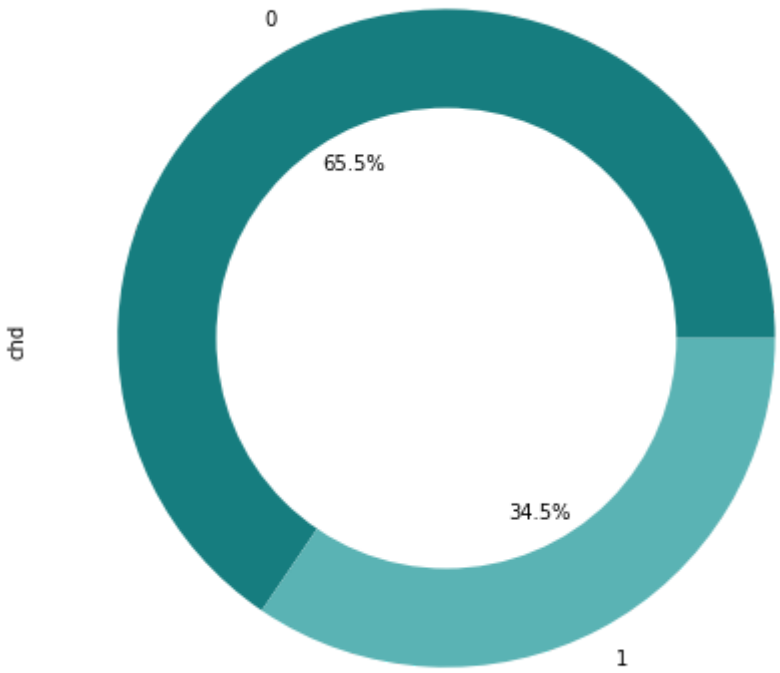
Out[24]:
```
0    302
1    159
Name: chd, dtype: int64
```

In [25]:
```
height = list(df['chd'].value_counts())
bars = [0,1]

fig = plt.figure(figsize = (7, 6))
ax1 = fig.add_subplot(1, 1, 1)

for s in ['top', 'bottom', 'left', 'right']:
    ax1.spines[s].set_visible(False)

ax1.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
patches = plt.bar(bars, height, color='#167D7F', width=0.7, align='center')
patches[1].set_facecolor('#5ab3b4')
plt.xticks(bars, bars)

plt.show()
```



In [26]:

```
df['chd'].value_counts().plot(kind='pie', figsize=(7, 6),colors=['#167D7F',
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
ax1.axis('equal')
plt.tight_layout()
plt.show()
```



Baivarite Analysis

In [1]:
```
# hide warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# import libraries
import numpy as np
import pandas as pd
import statistics
import matplotlib.pyplot as plt
%matplotlib inline
import pylab as plt
import seaborn as sns
from scipy.stats import chi2_contingency
```

In [3]:
```
# loading data
df = pd.read_csv("cleaned_disease.csv")
df.head()
```

Out[3]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160.0	12.00	5.73	23.11	Present	49.0	25.30	59.16	52	1
1	144.0	0.01	4.41	28.61	Absent	55.0	28.87	2.06	63	1
2	118.0	0.08	3.48	32.28	Present	52.0	29.14	3.81	46	0
3	170.0	7.50	6.41	38.03	Present	51.0	31.99	24.26	58	1
4	134.0	13.60	3.50	27.78	Present	60.0	25.99	57.34	49	1

1. Impact of numerical features on target feature

In [4]:
```
numerical_features = [col for col in df.columns \
                      if np.issubdtype(df[col].dtype, np.number)]

print(numerical_features)
numerical_features.pop(8)
print(numerical_features)
```

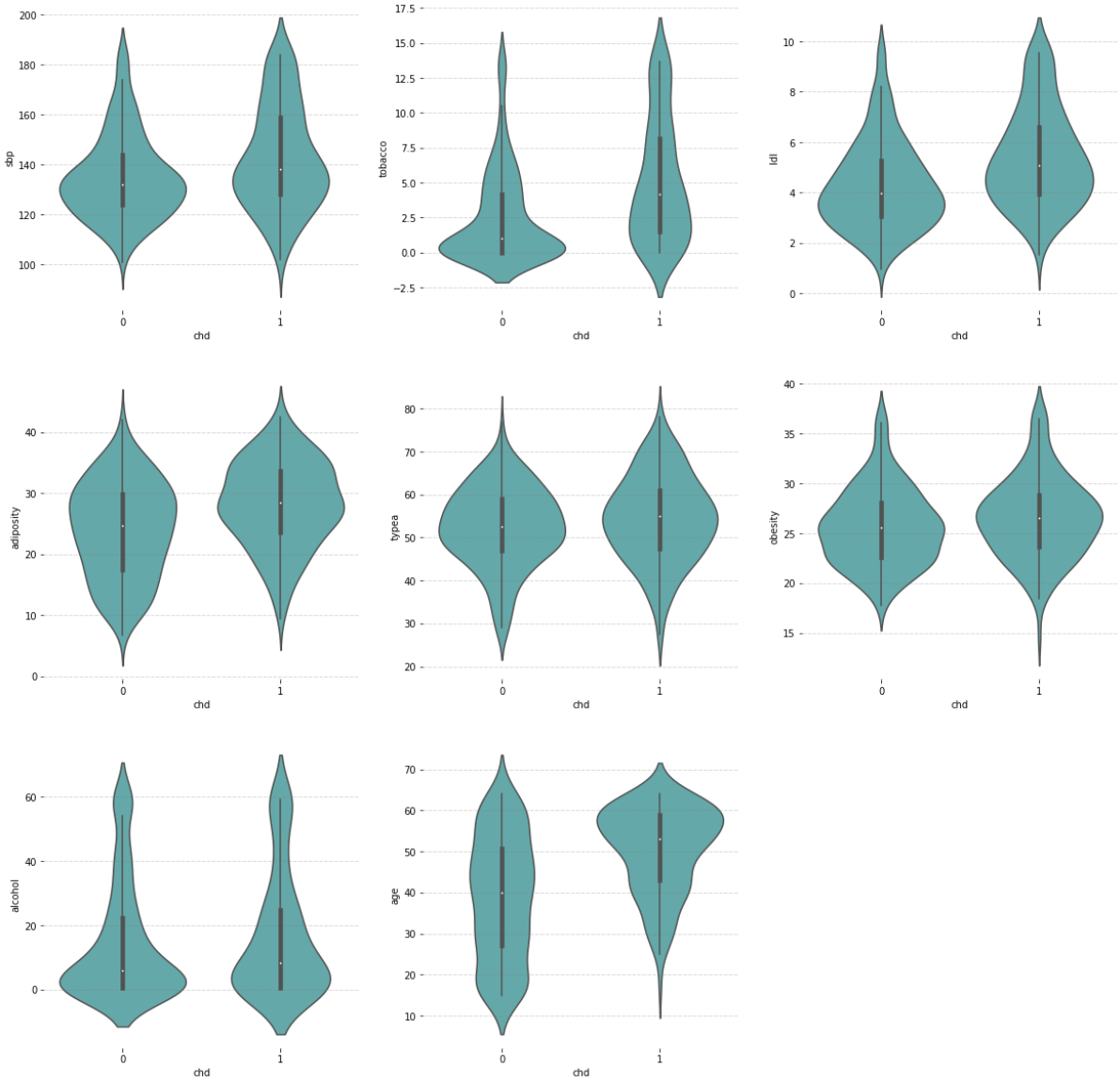
```
['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesity', 'alcohol', 'age',
'chd']
['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesity', 'alcohol', 'age']
```

In [5]:
```
# define categorical features
categorical_features = [col for col in df.columns \
                      if pd.api.types.is_string_dtype(df[col])]
print(categorical_features)
```

```
['famhist']
```

In [6]:
```
fig = plt.figure(figsize=(20,20))
for index, col in enumerate(numerical_features):
    ax = plt.subplot(3, 3, index+1)
    for s in ['top', 'bottom', 'left', 'right']:
        ax.spines[s].set_visible(False)
```

```
        ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5,
    sns.violinplot(x='chd', y=col, data=df, color='#5ab3b4')
```



In [7]:

```
from scipy.stats import ttest_ind

# define function for computing mean of column for 0 and 1 cases,
def test_means(data, col):
    disease = data["chd"] == 1
    values_disease = data[col][disease]
    values_no_disease = data[col][~disease]
    mean_disease = values_disease.mean()

    mean_no_disease = values_no_disease.mean()


    return [col, mean_disease, mean_no_disease]

test_df = pd.DataFrame(columns=["column", "mean disease","mean no disease"]

for index, col in enumerate(numerical_features):
    test_df.loc[index] = test_means(df, col)

test_df
```

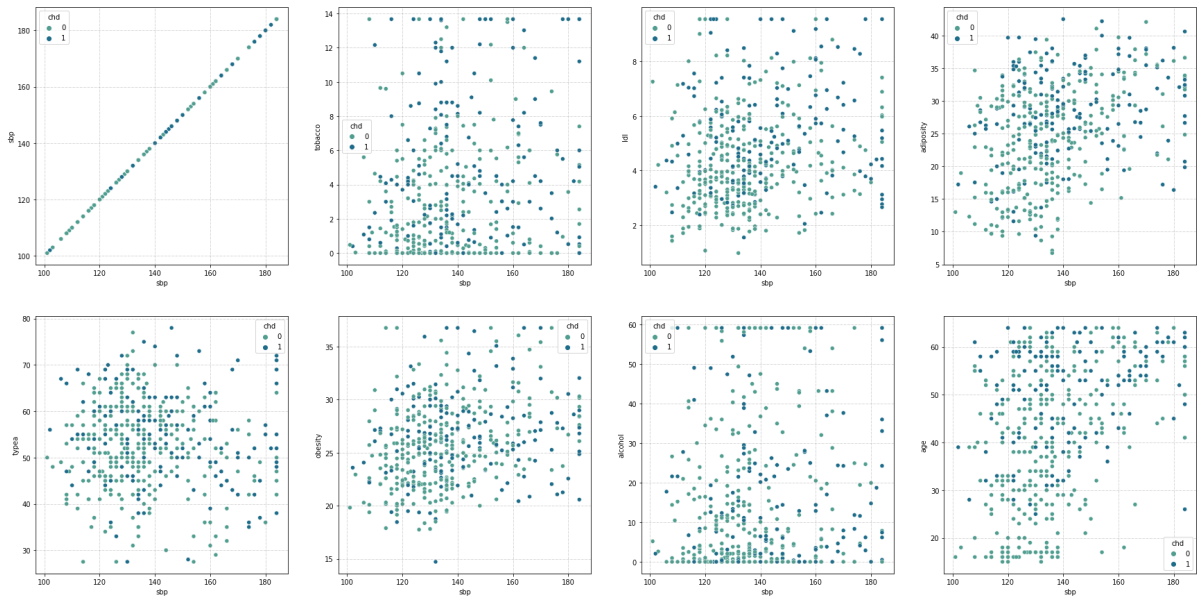
Out[7]:

	column	mean disease	mean no disease
0	sbp	142.767296	135.175497
1	tobacco	5.115951	2.564627
2	ldl	5.403278	4.312438
3	adiposity	28.219623	23.969106
4	typea	54.418239	52.428808
5	obesity	26.564159	25.683262
6	alcohol	16.752830	14.297219
7	age	50.484277	38.854305

2. Relationship between numerical features

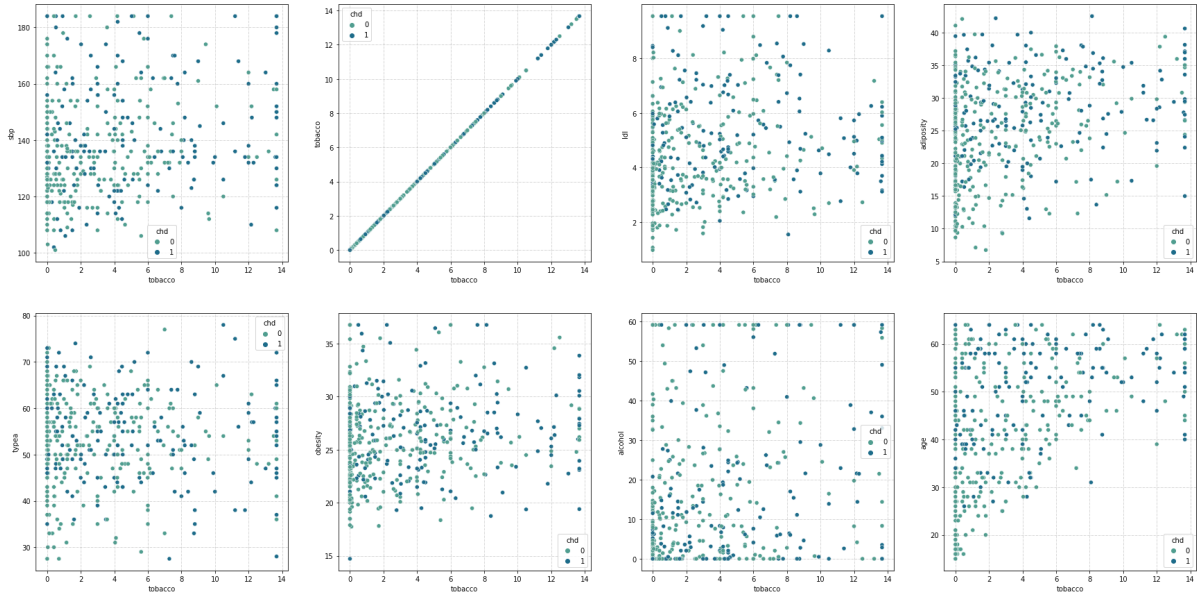
In [8]:

```
#spb
fig = plt.figure(figsize=(30,15))
for index, col in enumerate(numerical_features):
    ax = plt.subplot(2, 4, index+1)
    ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alph
    sns.scatterplot(x='sbp', y=col, hue='chd', data=df, palette='crest')
```



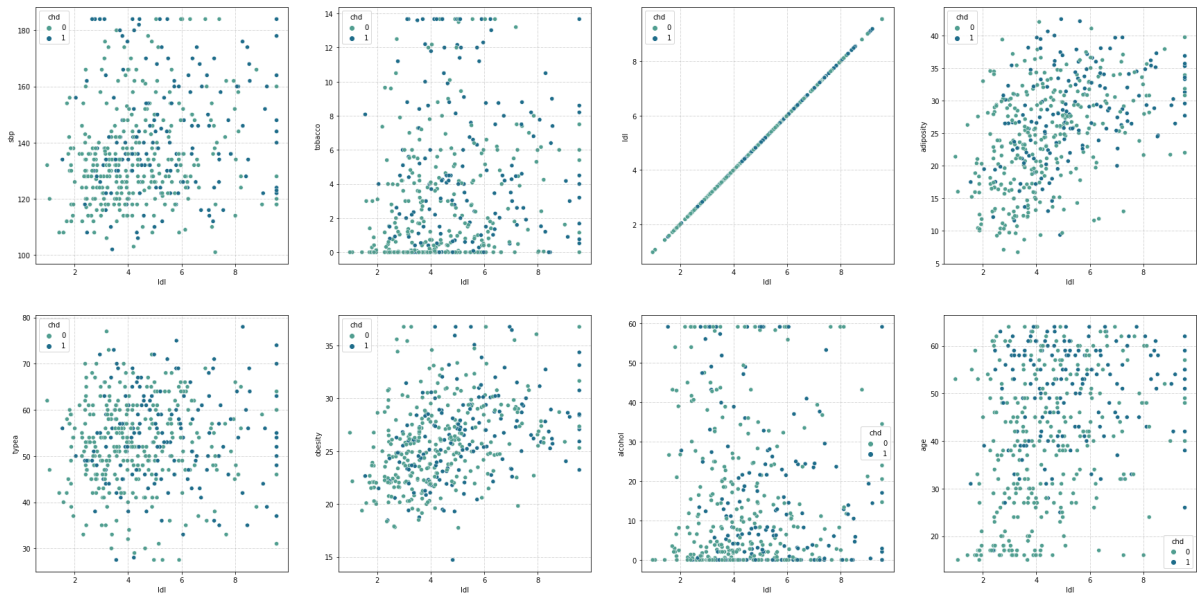
In [9]:

```
#tobacco
fig = plt.figure(figsize=(30,15))
for index, col in enumerate(numerical_features):
    ax = plt.subplot(2, 4, index+1)
    ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alph
    sns.scatterplot(x='tobacco', y=col, hue='chd', data=df, palette='crest'
```



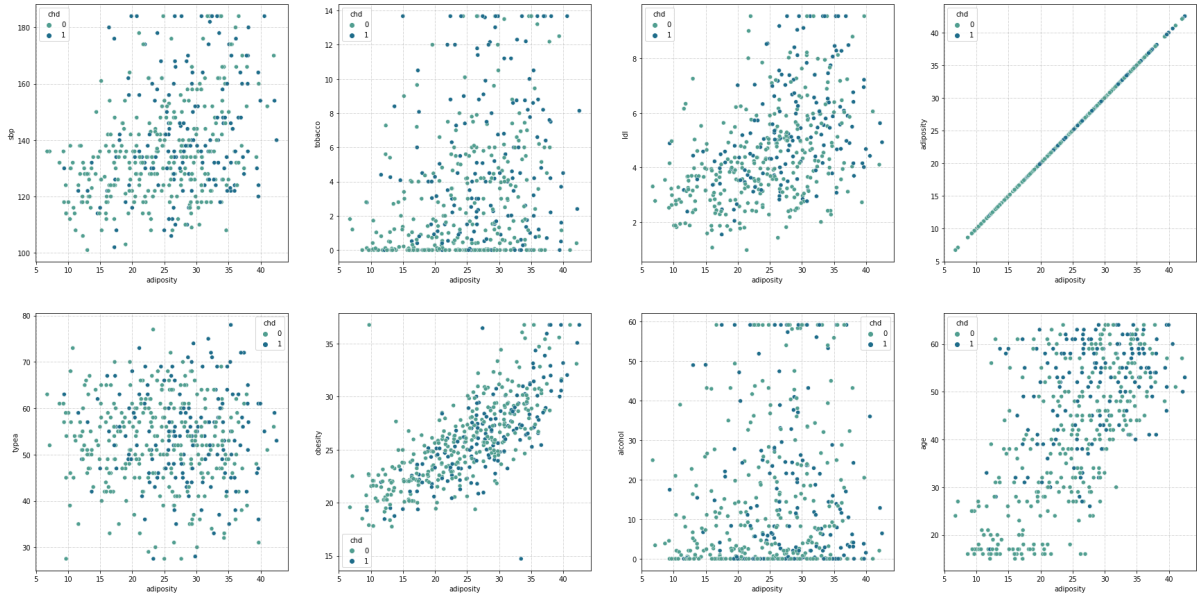
In [10]:

```
#ldl
fig = plt.figure(figsize=(30,15))
for index, col in enumerate(numerical_features):
    ax = plt.subplot(2, 4, index+1)
    ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alph
    sns.scatterplot(x='ldl', y=col, hue='chd', data=df, palette='crest')
```



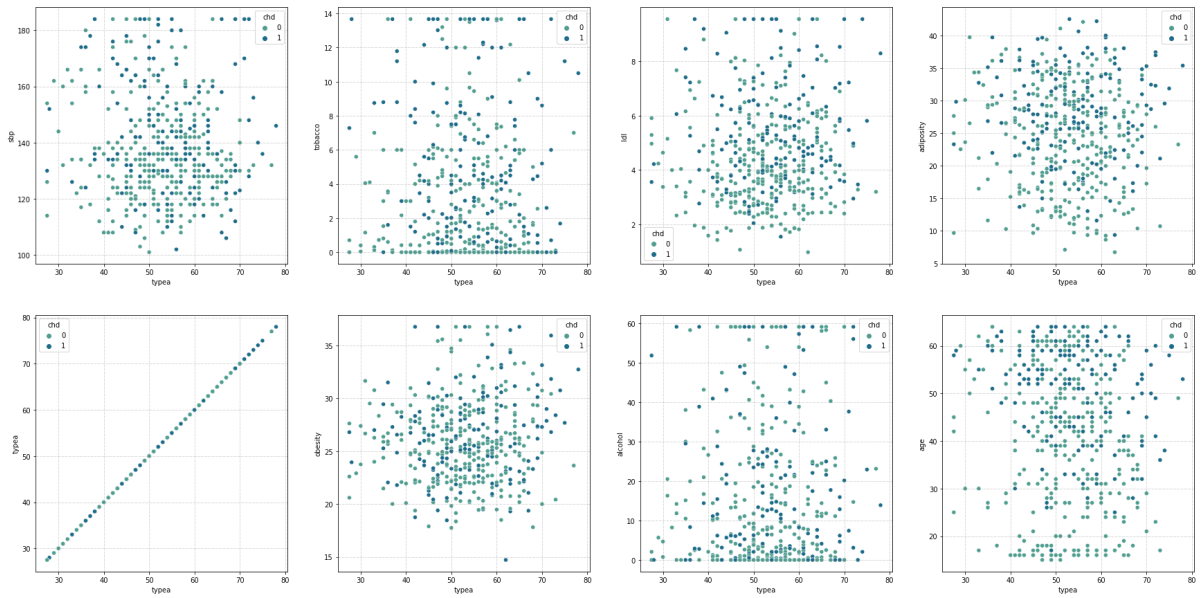
In [11]:

```
#adiposity
fig = plt.figure(figsize=(30,15))
for index, col in enumerate(numerical_features):
    ax = plt.subplot(2, 4, index+1)
    ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alph
    sns.scatterplot(x='adiposity', y=col, hue='chd', data=df, palette='cres
```



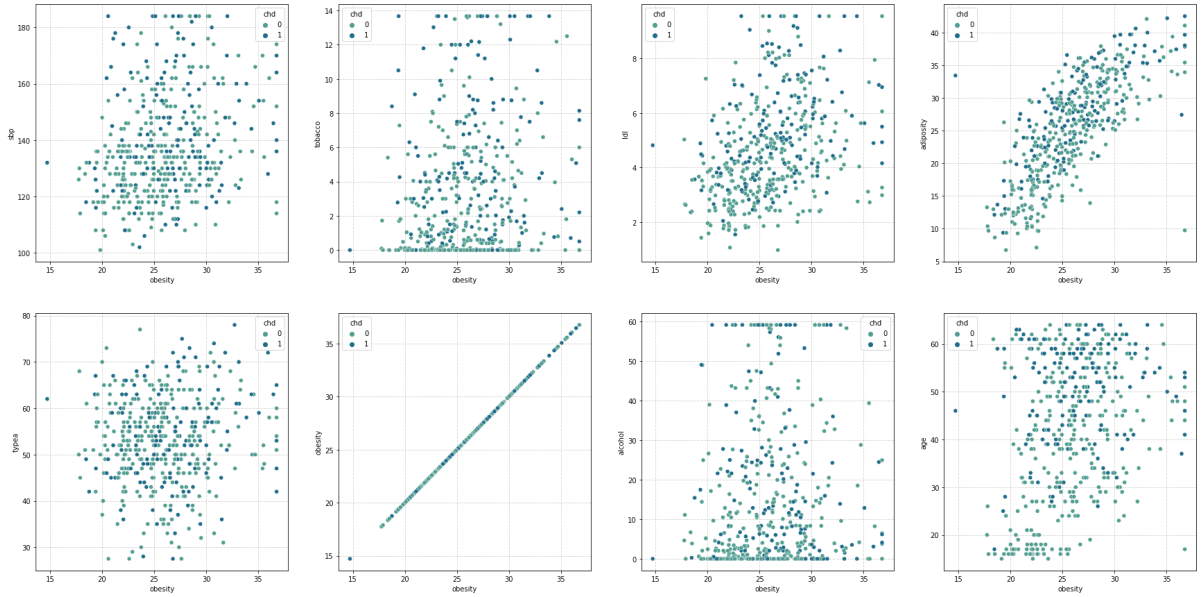
In [12]:

```
#typea
fig = plt.figure(figsize=(30,15))
for index, col in enumerate(numerical_features):
    ax = plt.subplot(2, 4, index+1)
    ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alph
    sns.scatterplot(x='typea', y=col, hue='chd', data=df, palette='crest')
```



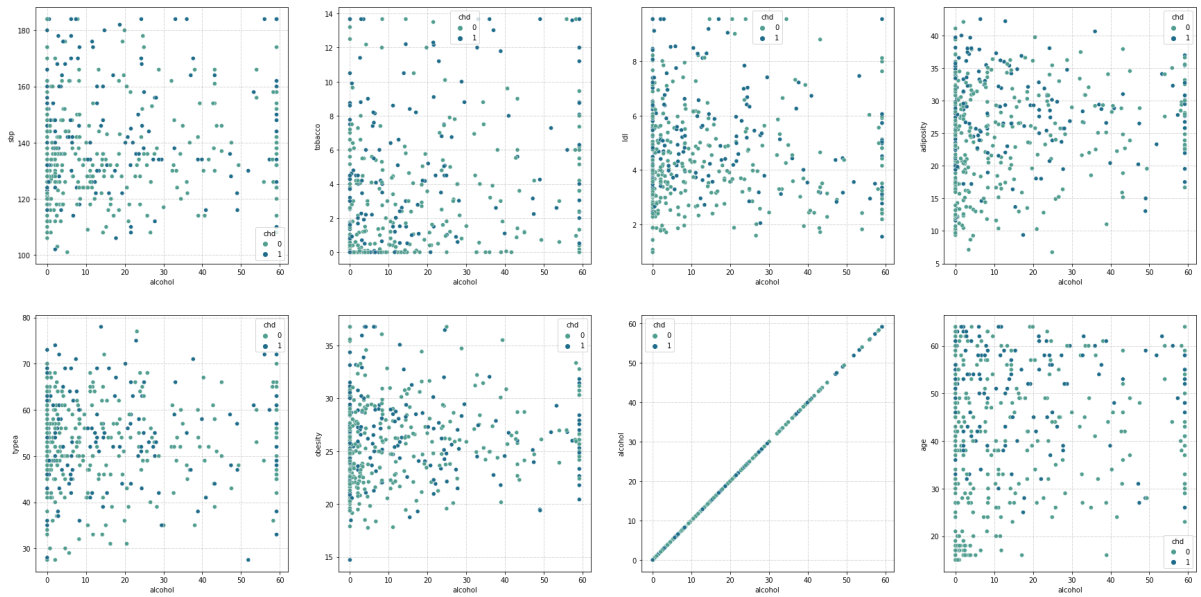
In [13]:

```
#Obesity
fig = plt.figure(figsize=(30,15))
for index, col in enumerate(numerical_features):
    ax = plt.subplot(2, 4, index+1)
    ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alph
    sns.scatterplot(x='obesity', y=col, hue='chd', data=df, palette='crest'
```



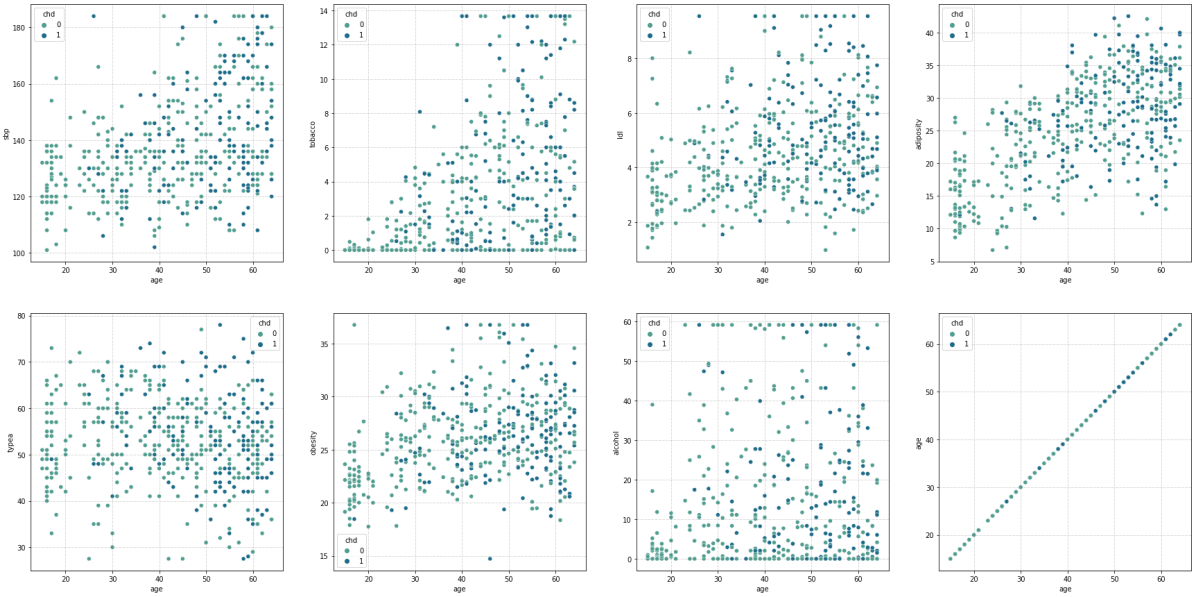
In [14]:

```
#alcohol
fig = plt.figure(figsize=(30,15))
for index, col in enumerate(numerical_features):
    ax = plt.subplot(2, 4, index+1)
    ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alph
    sns.scatterplot(x='alcohol', y=col, hue='chd', data=df, palette='crest'
```



In [15]:

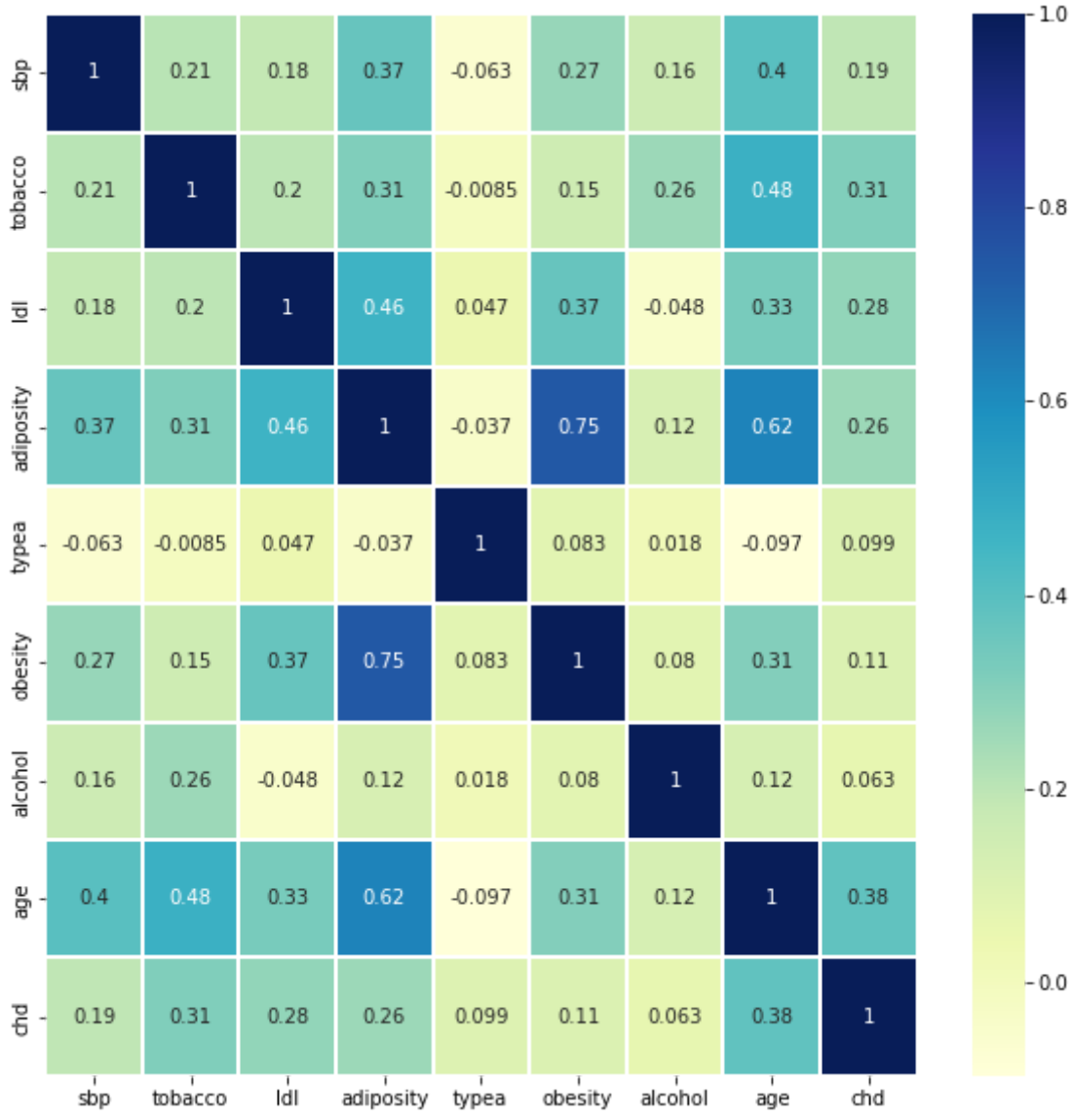
```
#age
fig = plt.figure(figsize=(30,15))
for index, col in enumerate(numerical_features):
    ax = plt.subplot(2, 4, index+1)
    ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alph
    sns.scatterplot(x='age', y=col, hue='chd', data=df, palette='crest')
```



Correlation Coefficients

In [16]:

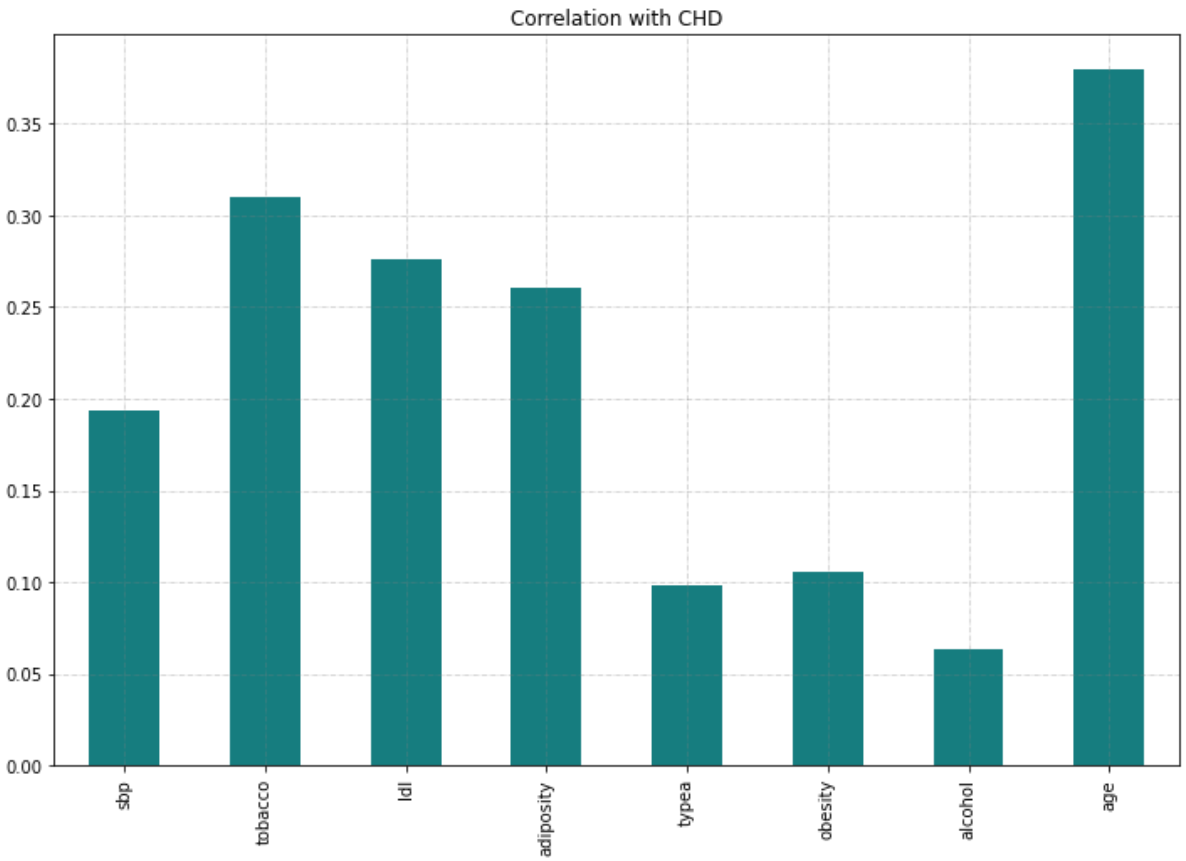
```
dfcorr = df.corr()
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(dfcorr, annot=True,linewidths=0.3,cmap="YlGnBu")
plt.show()
```



In [17]:
```
df.corr()["chd"]
```

Out[17]:
```
sbp          0.193353
tobacco      0.310341
ldl          0.276286
adiposity    0.260474
typea        0.098616
obesity      0.105491
alcohol      0.063105
age          0.379416
chd          1.000000
Name: chd, dtype: float64
```

In [18]:
```
df.drop('chd', axis=1).corrwith(df.chd).plot(kind='bar',
                                             figsize=(12, 8),title="Correla
plt.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
```



2. Relationship Between Categorical Variables

In [19]:
```
cross_table = pd.crosstab(df.chd , df.famhist)
cross_table
```

Out[19]:

famhist	Absent	Present
chd		
0	206	96
1	63	96

In [20]:
```
results = chi2_contingency(cross_table)
print('The P-Value of the ChiSq Test is:', results[1])
```

```
The P-Value of the ChiSq Test is: 5.907439103427825e-09
```

Feature Engineering

In [1]:
```
# hide warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# import libraries
import numpy as np
import pandas as pd
import statistics
import matplotlib.pyplot as plt
%matplotlib inline
import pylab as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
```

In [3]:
```
# loading data
df = pd.read_csv("cleaned_disease.csv")
df.head()
```

Out[3]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160.0	12.00	5.73	23.11	Present	49.0	25.30	59.16	52	1
1	144.0	0.01	4.41	28.61	Absent	55.0	28.87	2.06	63	1
2	118.0	0.08	3.48	32.28	Present	52.0	29.14	3.81	46	0
3	170.0	7.50	6.41	38.03	Present	51.0	31.99	24.26	58	1
4	134.0	13.60	3.50	27.78	Present	60.0	25.99	57.34	49	1

1. Categorical Encoding

In [4]:
```
dummy = pd.get_dummies(df['famhist'])
df=pd.concat ((df,dummy), axis = 1)
df = df.drop(['famhist'], axis = 1)
df = df.drop(['Absent'], axis = 1)
df = df.rename(columns ={"Present":"famhist"})
titles = list(df.columns)
titles[4],titles[5],titles[6],titles[7],titles[8],titles[9]= titles[9],titl
df=df[titles]
df
```

Out[4]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160.0	12.00	5.73000	23.11	1	49.0	25.30000	59.16	52	1
1	144.0	0.01	4.41000	28.61	0	55.0	28.87000	2.06	63	1
2	118.0	0.08	3.48000	32.28	1	52.0	29.14000	3.81	46	0
3	170.0	7.50	6.41000	38.03	1	51.0	31.99000	24.26	58	1
4	134.0	13.60	3.50000	27.78	1	60.0	25.99000	57.34	49	1
...	...	...	...	...	...	...	...	...	...	...
456	184.0	0.40	5.98000	31.72	0	64.0	28.45000	0.00	58	0
457	182.0	4.20	4.41000	32.10	0	52.0	28.61000	18.72	52	1
458	108.0	3.00	1.59000	15.23	0	40.0	20.09000	26.64	55	0
459	118.0	5.40	9.55125	30.79	0	64.0	27.35000	23.97	40	0
460	132.0	0.00	4.82000	33.41	1	62.0	14.71625	0.00	46	1

461 rows × 10 columns

2. Feature Scaling

In [5]:

```
X_before = df.iloc[:, 0:8]
y = df["chd"]

X_before.head()
```

Out[5]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol
0	160.0	12.00	5.73	23.11	1	49.0	25.30	59.16
1	144.0	0.01	4.41	28.61	0	55.0	28.87	2.06
2	118.0	0.08	3.48	32.28	1	52.0	29.14	3.81
3	170.0	7.50	6.41	38.03	1	51.0	31.99	24.26
4	134.0	13.60	3.50	27.78	1	60.0	25.99	57.34

In [6]:

```
scaler = MinMaxScaler()
X = pd.DataFrame(scaler.fit_transform(X_before.values),
                 columns=X_before.columns, index=X_before.index)
X.head()
```

Out[6]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol
0	0.710843	0.877754	0.554178	0.457902	1.0	0.425743	0.479989	1.000000
1	0.518072	0.000731	0.400175	0.611748	0.0	0.544554	0.641893	0.034821
2	0.204819	0.005852	0.291673	0.714406	1.0	0.485149	0.654138	0.064402
3	0.831325	0.548597	0.633513	0.875245	1.0	0.465347	0.783390	0.410074
4	0.397590	0.994788	0.294006	0.588531	1.0	0.643564	0.511281	0.969236

3. Handling imbalanced data

In [7]:

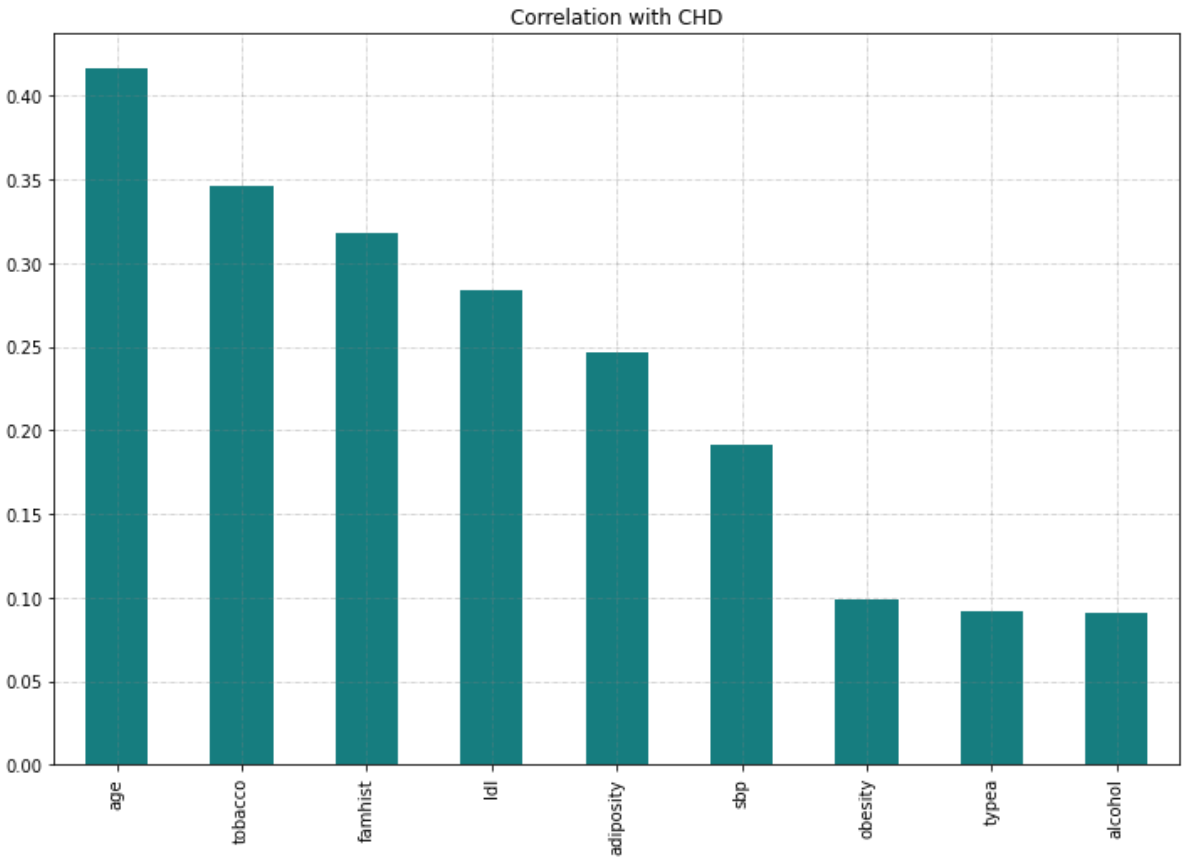
```
msk = df['chd'] == 1
```

```
num_to_oversample = len(df) - 2*msk.sum()
df_positive_oversample = df[msk].sample(n=num_to_oversample, replace=True,
df = pd.concat([df, df_positive_oversample])
df['chd'].value_counts()
```

Out[7]:
```
1    302
0    302
Name: chd, dtype: int64
```

4. Feature Selection

In [8]:
```
df.drop('chd', axis=1).corrwith(df.chd).sort_values(ascending=False).plot(k
                                                    figsize=(12, 8),title="Correla
plt.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
```



In [9]:
```
del df['obesity']
```

In [10]:
```
df.head()
```

Out[10]:

	sbp	tobacco	ldl	adiposity	famhist	typea	alcohol	age	chd
0	160.0	12.00	5.73	23.11	1	49.0	59.16	52	1
1	144.0	0.01	4.41	28.61	0	55.0	2.06	63	1
2	118.0	0.08	3.48	32.28	1	52.0	3.81	46	0
3	170.0	7.50	6.41	38.03	1	51.0	24.26	58	1
4	134.0	13.60	3.50	27.78	1	60.0	57.34	49	1

In [12]:
```
df.to_csv('modeling.csv',index=False)
```

Modeling and Evaluation

In [1]:
```
# hide warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# import libraries
import numpy as np
import pandas as pd

import statistics
import matplotlib.pyplot as plt
%matplotlib inline
import pylab as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix, classificatio
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.tree import DecisionTreeClassifier
import six
import sys
sys.modules['sklearn.externals.six']=six
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
from sklearn.tree import export_text
```

In [3]:
```
# loading data
df = pd.read_csv("modeling.csv")
df.head()
```

Out[3]:

	sbp	tobacco	ldl	adiposity	famhist	typea	alcohol	age	chd
0	160.0	12.00	5.73	23.11	1	49.0	59.16	52	1
1	144.0	0.01	4.41	28.61	0	55.0	2.06	63	1
2	118.0	0.08	3.48	32.28	1	52.0	3.81	46	0
3	170.0	7.50	6.41	38.03	1	51.0	24.26	58	1
4	134.0	13.60	3.50	27.78	1	60.0	57.34	49	1

In [4]:
```
X_before = df.iloc[:, 0:8]
y = df["chd"]

scaler = MinMaxScaler()
X = pd.DataFrame(scaler.fit_transform(X_before.values),
                 columns=X_before.columns, index=X_before.index)
X.head()
```

Out[4]:

	sbp	tobacco	ldl	adiposity	famhist	typea	alcohol	age
0	0.710843	0.877754	0.554178	0.457902	1.0	0.425743	1.000000	0.755102
1	0.518072	0.000731	0.400175	0.611748	0.0	0.544554	0.034821	0.979592
2	0.204819	0.005852	0.291673	0.714406	1.0	0.485149	0.064402	0.632653
3	0.831325	0.548597	0.633513	0.875245	1.0	0.465347	0.410074	0.877551
4	0.397590	0.994788	0.294006	0.588531	1.0	0.643564	0.969236	0.693878

In [5]:
```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, ran
```

Modeling

1. Support Vector Machines

1.1. Hypertuning model parameters using GridSearchCV

In [6]:
```
param_grid = {'C': [1,0.1,10], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf'
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=True)
grid.fit(X_train,y_train)
```

Out[6]:
```
Fitting 5 folds for each of 252 candidates, totalling 1260 fits
GridSearchCV(estimator=SVC(),
             param_grid={'C': [1, 0.1, 10], 'degree': [2, 3, 4, 5, 6, 7, 8],
                         'gamma': [1, 0.1, 0.01, 0.001],
                         'kernel': ['rbf', 'poly', 'sigmoid']},
             verbose=True)
```

In [7]:
```
print(grid.best_estimator_)
print(grid.best_params_)
```

```
SVC(C=1, degree=7, gamma=1, kernel='poly')
{'C': 1, 'degree': 7, 'gamma': 1, 'kernel': 'poly'}
```

1.2. Building SVM Classifier

In [8]:
```
#create new a SVM model
model = SVC(kernel='poly', C=1, degree=7, gamma=1, probability=True)

# Fit the classifier to the data
model.fit(X_train,y_train)
svm_pred = model.predict(X_test)
```

1.3. Assess Model

In [9]:
```
cr = classification_report(y_test, svm_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.84      0.79      0.81        89
           1       0.81      0.86      0.83        93

    accuracy                           0.82       182
   macro avg       0.83      0.82      0.82       182
weighted avg       0.83      0.82      0.82       182
```

In [10]:

```
cf_matrix = confusion_matrix(y_test,svm_pred, labels=[1,0])
print(cf_matrix)
```

```
[[80 13]
 [19 70]]
```

In [11]:

```
tn, fp, fn, tp = confusion_matrix(y_test,svm_pred, labels=[0, 1]).ravel()

print('True Positive', tp)
print('True Negative', tn)
print('False Positive', fp)
print('False Negative', fn)
```

```
True Positive 80
True Negative 70
False Positive 19
False Negative 13
```

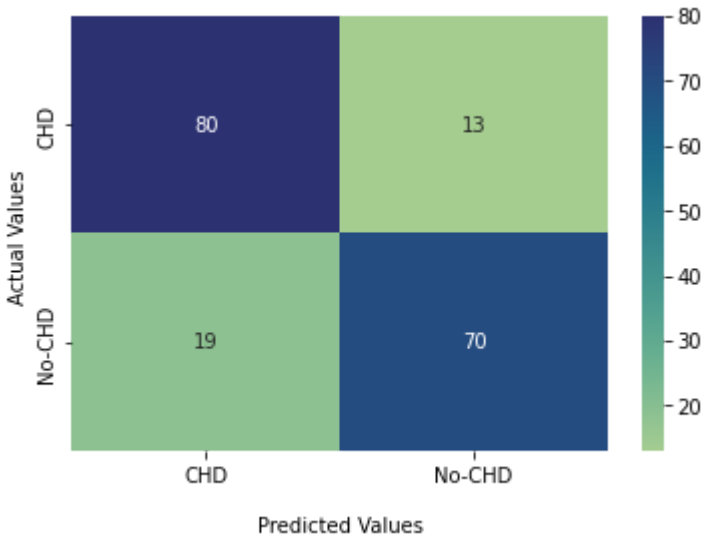
In [12]:

```
total1=sum(sum(cf_matrix))
#####from confusion matrix calculate accuracy
accuracy1=(cf_matrix[0,0]+cf_matrix[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cf_matrix[0,0]/(cf_matrix[0,0]+cf_matrix[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cf_matrix[1,1]/(cf_matrix[1,0]+cf_matrix[1,1])
print('Specificity : ', specificity1)
```

```
Accuracy :  0.8241758241758241
Sensitivity :  0.8602150537634409
Specificity :  0.7865168539325843
```

In [13]:

```
ax = sns.heatmap(cf_matrix, annot=True, cmap='crest')

ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['CHD','No-CHD'])
ax.yaxis.set_ticklabels(['CHD','No-CHD'])

## Display the visualization of the Confusion Matrix.
plt.show()
```



In [14]:

```
# worset case, when the model predict all wrong
r_probs = [0 for _ in range(len(y_test))]
# predict probabilities
svm_probs = model.predict_proba(X_test)
# keep probabilities for the positive outcome only
svm_probs = svm_probs[:, 1]
```

In [15]:

```
# calculate scores
r_auc = roc_auc_score(y_test, r_probs)
print(r_auc)
svm_auc = roc_auc_score(y_test, svm_probs)
print(svm_auc)
```

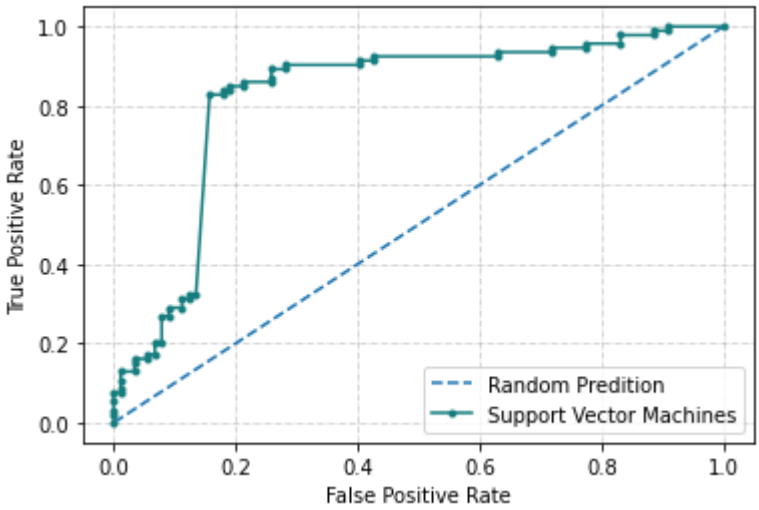
```
0.5
0.824332487616286
```

In [16]:

```
# calculate roc curves
r_fpr, r_tpr, _ = roc_curve(y_test, r_probs)
svm_fpr, svm_tpr, _ = roc_curve(y_test, svm_probs)
```

In [17]:

```
# plot the roc curve for the model
plt.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
plt.plot(r_fpr, r_tpr, linestyle='--', label='Random Predition',)
plt.plot(svm_fpr, svm_tpr, marker='.', label='Support Vector Machines', col
# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()
# show the plot
plt.show()
```



2. Decision Tree

2.1. Hypertuning model parameters using GridSearchCV

In [18]:

```
tuned_parameters = [{'criterion': ['gini','entropy'],'max_depth': [1,2,3,4,
                     'min_samples_split': [2,4,6,8]}]
scores = ['recall']
for score in scores:

    print()
    print(f"Tuning hyperparameters for {score}")
    print()

    grid = GridSearchCV(
        DecisionTreeClassifier(), tuned_parameters,
        scoring = f'{score}_macro'
    )
    grid.fit(X_train, y_train)

    print("Best parameters set found on development set:")
    print()
    print(grid.best_params_)
    print()
```

```
Tuning hyperparameters for recall

Best parameters set found on development set:

{'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 4}
```

2.2. Build Classifier

In [19]:

```
# create Decision Tree Model
model = DecisionTreeClassifier(criterion = 'gini', max_depth =4,min_samples_
# Fit the classifier to the data
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

In [20]:

```
# importance feature
importance = pd.DataFrame({'feature': X_train.columns,
'importance' : np.round(model.feature_importances_, 3)})
importance.sort_values('importance', ascending=False, inplace = True)
print(importance)
```

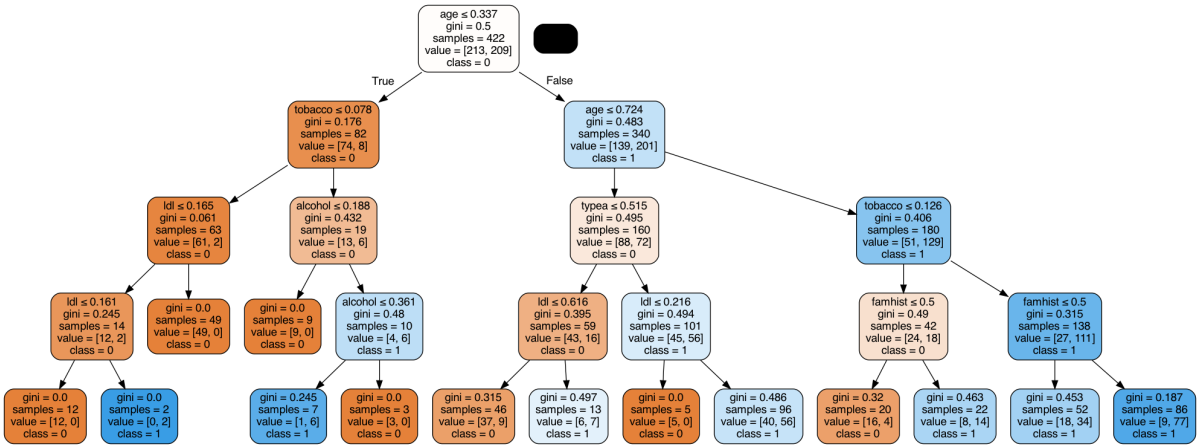
```
      feature  importance
7         age       0.518
1     tobacco       0.134
2         ldl       0.111
4     famhist       0.091
6     alcohol       0.076
5       typea       0.070
0         sbp       0.000
3   adiposity       0.000
```

2.3. Assess the model

In [21]:

```
dot_data = StringIO()
export_graphviz(model, out_file=dot_data,
                filled= True, rounded=True,
                special_characters= True, feature_names= X.columns,class_nam
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('DecisionTreeModel.png')
Image(graph.create_png())
```

Out[21]:



In [22]:

```
feature_names = X.columns
tree_rules = export_text(model,
                         feature_names = list(X))

print(tree_rules)
```

```
|--- age <= 0.34
|   |--- tobacco <= 0.08
|   |   |--- ldl <= 0.17
|   |   |   |--- ldl <= 0.16
|   |   |   |   |--- class: 0
|   |   |   |--- ldl >  0.16
|   |   |   |   |--- class: 1
|   |   |--- ldl >  0.17
|   |   |   |--- class: 0
|   |--- tobacco >  0.08
|   |   |--- alcohol <= 0.19
|   |   |   |--- class: 0
|   |   |--- alcohol >  0.19
|   |   |   |--- alcohol <= 0.36
|   |   |   |   |--- class: 1
|   |   |   |--- alcohol >  0.36
|   |   |   |   |--- class: 0
|--- age >  0.34
|   |--- age <= 0.72
|   |   |--- typea <= 0.51
|   |   |   |--- ldl <= 0.62
|   |   |   |   |--- class: 0
|   |   |   |--- ldl >  0.62
|   |   |   |   |--- class: 1
|   |   |--- typea >  0.51
|   |   |   |--- ldl <= 0.22
|   |   |   |   |--- class: 0
|   |   |   |--- ldl >  0.22
|   |   |   |   |--- class: 1
|   |--- age >  0.72
|   |   |--- tobacco <= 0.13
|   |   |   |--- famhist <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- famhist >  0.50
|   |   |   |   |--- class: 1
|   |   |--- tobacco >  0.13
|   |   |   |--- famhist <= 0.50
|   |   |   |   |--- class: 1
|   |   |   |--- famhist >  0.50
|   |   |   |   |--- class: 1
```

In [23]:

```
cr = classification_report(y_test, y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.86      0.63      0.73        89
           1       0.72      0.90      0.80        93

    accuracy                           0.77       182
   macro avg       0.79      0.77      0.76       182
weighted avg       0.79      0.77      0.76       182
```

In [24]:

```
cf_matrix = confusion_matrix(y_test,y_pred, labels=[1,0])
print(cf_matrix)
```

```
[[84  9]
 [33 56]]
```

In [25]:

```
tn, fp, fn, tp = confusion_matrix(y_test,y_pred, labels=[0, 1]).ravel()
```

```
print('True Positive', tp)
print('True Negative', tn)
print('False Positive', fp)
print('False Negative', fn)
```

```
True Positive 84
True Negative 56
False Positive 33
False Negative 9
```

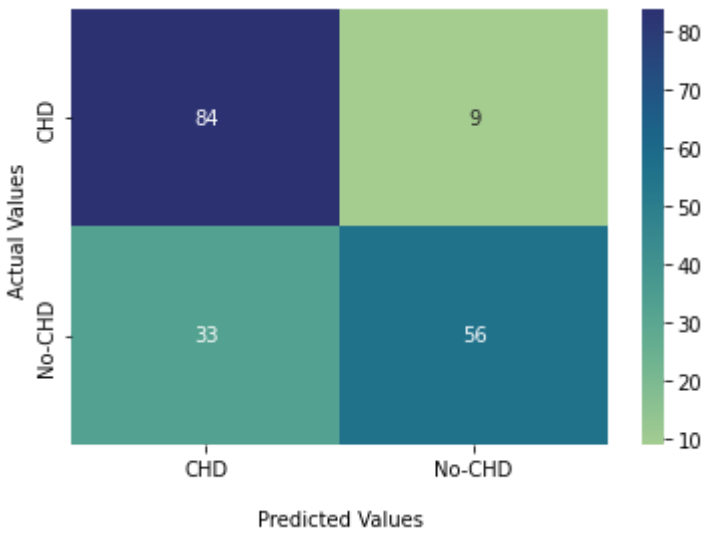
In [26]:

```
total1=sum(sum(cf_matrix))
#####from confusion matrix calculate accuracy
accuracy1=(cf_matrix[0,0]+cf_matrix[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cf_matrix[0,0]/(cf_matrix[0,0]+cf_matrix[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cf_matrix[1,1]/(cf_matrix[1,0]+cf_matrix[1,1])
print('Specificity : ', specificity1)
```

```
Accuracy :  0.7692307692307693
Sensitivity :  0.9032258064516129
Specificity :  0.6292134831460674
```

In [27]:

```
ax = sns.heatmap(cf_matrix, annot=True, cmap='crest')

ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['CHD','No-CHD'])
ax.yaxis.set_ticklabels(['CHD','No-CHD'])

## Display the visualization of the Confusion Matrix.
plt.show()
```



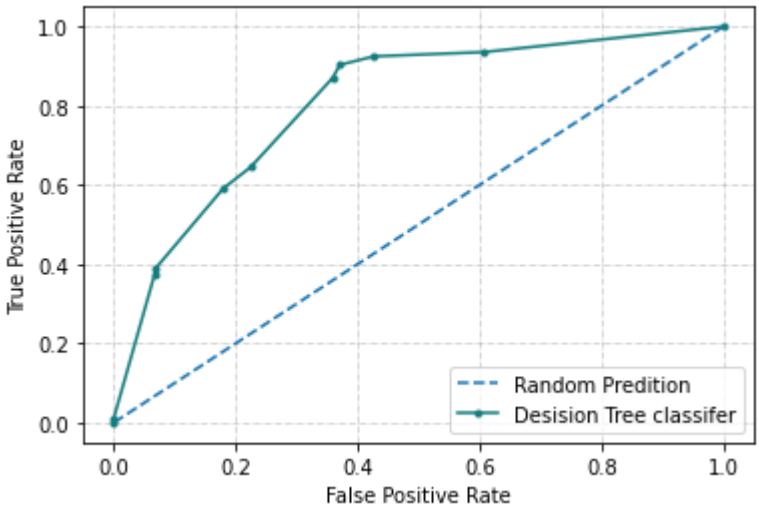
In [28]:

```
# predict probabilities
tree_probs = model.predict_proba(X_test)
# keep probabilities for the positive outcome only
tree_probs = tree_probs[:, 1]
tree_auc = roc_auc_score(y_test, tree_probs)
print(tree_auc)
```

```
0.8071161048689138
```

In [29]:
```
r_fpr, r_tpr, _ = roc_curve(y_test, r_probs)
tree_fpr, tree_tpr, _ = roc_curve(y_test, tree_probs)
```

In [30]:
```
# plot the roc curve for the model
plt.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
plt.plot(r_fpr, r_tpr, linestyle='--', label='Random Predition',)
plt.plot(tree_fpr, tree_tpr, marker='.', label='Desision Tree classifer', c
# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()
# show the plot
plt.show()
```



3. Logistic Regression

3.1. Hypertuning model parameters using GridSearchCV

In [31]:
```
parameters = {'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
    'C' : np.logspace(-4, 4, 20),
    'solver' : ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'max_iter' : [100, 1000,2500, 5000]
    }
clf = GridSearchCV(LogisticRegression(), param_grid = parameters, verbose=T
clf.fit(X_train,y_train)
```

Out[31]:
```
Fitting 5 folds for each of 1600 candidates, totalling 8000 fits
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': array([1.00000000e-04, 2.63665090e-04, 6.95192
796e-04, 1.83298071e-03,
       4.83293024e-03, 1.27427499e-02, 3.35981829e-02, 8.85866790e-02,
       2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240e+00,
       1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808e+02,
       5.45559478e+02, 1.43844989e+03, 3.79269019e+03, 1.00000000e+04]),
                         'max_iter': [100, 1000, 2500, 5000],
                         'penalty': ['l1', 'l2', 'elasticnet', 'none'],
                         'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sa
g',
                                    'saga']},
             verbose=True)
```

In [32]:
```
print("Tuned Hyperparameters :", clf.best_params_)
print("Best Estimator :",clf.best_estimator_)
```

```
Tuned Hyperparameters : {'C': 0.23357214690901212, 'max_iter': 100, 'penalt
y': 'l2', 'solver': 'newton-cg'}
Best Estimator : LogisticRegression(C=0.23357214690901212, solver='newton-c
g')
```

3.2. Building Logistic Regression Classifier

In [33]:
```
# build the model
log_reg = LogisticRegression(solver='newton-cg', penalty='l2', C=0.23357214

# fit the model with data
log_reg.fit(X_train,y_train)

# predict the testing set
log_pred = log_reg.predict(X_test)
```

In [34]:
```
print('intercept ', log_reg.intercept_[0])
print('classes', log_reg.classes_)
pd.DataFrame({'coeff': log_reg.coef_[0]},
             index=X.columns)
```

```
intercept  -2.251568648251529
classes [0 1]
```

Out[34]:

	coeff
sbp	0.224334
tobacco	0.988312
ldl	0.606026
adiposity	0.105982
famhist	0.842028
typea	0.362228
alcohol	0.072231
age	1.470739

3.3. Assess Model

In [35]:
```
cnf_matrix = confusion_matrix(y_test, log_pred, labels=[1,0])
cnf_matrix
```

Out[35]:
```
array([[74, 19],
       [29, 60]])
```

In [36]:
```
tn, fp, fn, tp = confusion_matrix(y_test,log_pred, labels=[0, 1]).ravel()

print('True Positive', tp)
print('True Negative', tn)
print('False Positive', fp)
print('False Negative', fn)
```

```
True Positive 74
True Negative 60
False Positive 29
False Negative 19
```

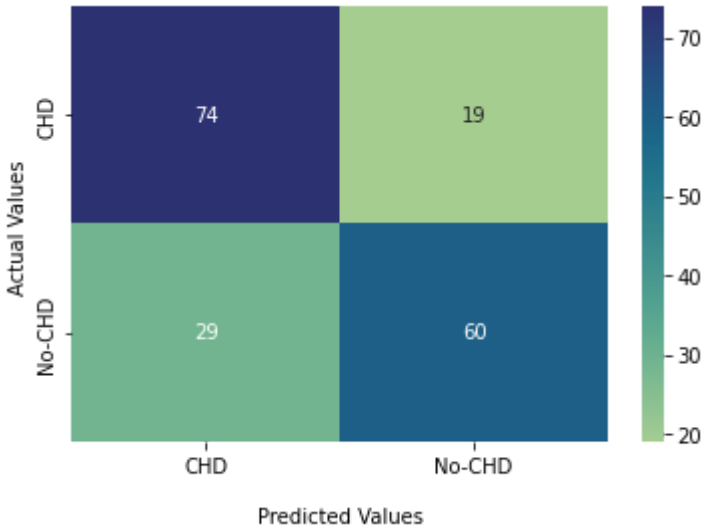
In [37]:

```
total1=sum(sum(cnf_matrix))
#####from confusion matrix calculate accuracy
accuracy1=(cnf_matrix[0,0]+cnf_matrix[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cnf_matrix[0,0]/(cnf_matrix[0,0]+cnf_matrix[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cnf_matrix[1,1]/(cnf_matrix[1,0]+cnf_matrix[1,1])
print('Specificity : ', specificity1)
```

```
Accuracy :  0.7362637362637363
Sensitivity :  0.7956989247311828
Specificity :  0.6741573033707865
```

In [38]:

```
ax = sns.heatmap(cnf_matrix, annot=True, cmap='crest')

ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['CHD','No-CHD'])
ax.yaxis.set_ticklabels(['CHD','No-CHD'])

## Display the visualization of the Confusion Matrix.
plt.show()
```



In [39]:

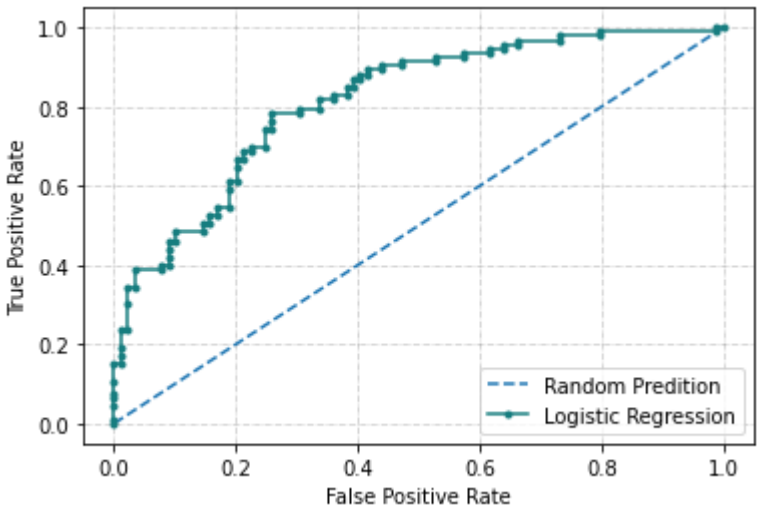
```
cr_lo = classification_report(y_test, log_pred)
print(cr_lo)
```

```
              precision    recall  f1-score   support

           0       0.76      0.67      0.71        89
           1       0.72      0.80      0.76        93

    accuracy                           0.74       182
   macro avg       0.74      0.73      0.73       182
weighted avg       0.74      0.74      0.74       182
```

In [40]:

```
p_probs = log_reg.predict_proba(X_test)
r_probs = [0 for _ in range(len(y_test))]

p_probs = p_probs[:, 1]

r_auc = roc_auc_score(y_test, r_probs)
print(r_auc)
logreg_auc = roc_auc_score(y_test, p_probs)
print(logreg_auc)


r_fpr, r_tpr, _ = roc_curve(y_test, r_probs)
logreg_fpr, logreg_tpr, _ = roc_curve(y_test, p_probs)

# plot the roc curve for the model
plt.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
plt.plot(r_fpr, r_tpr, linestyle='--', label='Random Predition',)
plt.plot(logreg_fpr, logreg_tpr, marker='.', label='Logistic Regression', c
# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()
# show the plot
plt.show()
```

```
0.5
0.8150296000966534
```



In [41]:

```
log_loss(y_test, p_probs)
```

Out[41]:

```
0.543281570543671
```

Evalution

In [42]:

```
# plot the roc curve for the model
plt.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha =
plt.plot(r_fpr, r_tpr, linestyle='--', label='Random Predition')
plt.plot(svm_fpr, svm_tpr, marker='.', label='Support Vector Machines', col
plt.plot(logreg_fpr, logreg_tpr, marker='.', label='Logistic Regression', c
plt.plot(tree_fpr, tree_tpr, marker='.', label='Desision Tree classifer', c

# axis labels
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()
# show the plot
plt.show()
```