Simulation of the Design Lab Project

Importing the Different Libraries

In [2]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import sklearn
import numpy as np
import pandas as pd
import plotly as plot
import plotly.express as px
import plotly.graph_objs as go

#import cufflinks as cf
import matplotlib.pyplot as plt
#import seaborn as sns
import os
from sklearn.metrics import accuracy_score
import plotly.offline as pyo
from plotly.offline import init_notebook_mode, plot, iplot
```

Acess the data of different people and storing it in 'heart' data frame

In [3]:

```
heart = pd.read_csv(r'C:\Users\abhishek\DesignLab\Project_Design_Lab\heart.csv')
```

Let's see how our data look like in Tabular form

In [4]:

```
heart
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2
14	58	0	3	150	283	1	0	162	0	1.0	2	0	2
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2
16	58	0	2	120	340	0	1	172	0	0.0	2	0	2
17	66	0	3	150	226	0	1	114	0	2.6	0	0	2
18	43	1	0	150	247	0	1	171	0	1.5	2	0	2
19	69	0	3	140	239	0	1	151	0	1.8	2	2	2
20	59	1	0	135	234	0	1	161	0	0.5	1	0	3
21	44	1	2	130	233	0	1	179	1	0.4	2	0	2
22	42	1	0	140	226	0	1	178	0	0.0	2	0	2
23	61	1	2	150	243	1	1	137	1	1.0	1	0	2
24	40	1	3	140	199	0	1	178	1	1.4	2	0	3
25	71	0	1	160	302	0	1	162	0	0.4	2	2	2
26	59	1	2	150	212	1	1	157	0	1.6	2	0	2
27	51	1	2	110	175	0	1	123	0	0.6	2	0	2
28	65	0	2	140	417	1	0	157	0	0.8	2	1	2
29	53	1	2	130	197	1	0	152	0	1.2	0	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
273	58	1	0	100	234	0	1	156	0	0.1	2	1	3
274	47	1	0	110	275	0	0	118	1	1.0	1	1	2
275	52	1	0	125	212	0	1	168	0	1.0	2	2	3
276	58	1	0	146	218	0	1	105	0	2.0	1	1	3
277	57	1	1	124	261	0	1	141	0	0.3	2	0	3
278	58	0	1	136	319	1	0	152	0	0.0	2	2	2
279	61	1	0	138	166	0	0	125	1	3.6	1	1	2
280	42	1	0	136	315	0	1	125	1	1.8	1	0	1
281	52	1	0	128	204	1	1	156	1	1.0	1	0	0
282	59	1	2	126	218	1	1	134	0	2.2	1	1	1
283	40	1	0	152	223	0	1	181	0	0.0	2	0	3
284	61	1	0	140	207	0	0	138	1	1.9	2	1	3
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2
287	57	1	1	154	232	0	0	164	0	0.0	2	1	2
288	57	1	0	110	335	0	1	143	1	3.0	1	1	3
289	55	0	0	128	205	0	2	130	1	2.0	1	1	3
290	61	1	0	148	203	0	1	161	0	0.0	2	1	3
291	58	1	0	114	318	0	2	140	0	4.4	0	3	1
292	58	0	0	170	225	1	0	146	1	2.8	1	2	1
293	67	1	2	152	212	0	0	150	0	0.8	1	0	3
294	44	1	0	120	169	0	1	144	1	2.8	0	0	1
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2
297	59	1	0	164	176	1	0	90	0	1.0	1	2	1
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 14 columns

In [5]:

```
info = ["age", "1: male, 0: female", "chest pain type, 1: typical angina, 2: atypical angin
        "resting electrocardiographic results (values 0,1,2)", " maximum heart rate achieve
```

Dataset Description

In [6]:

```
for i in range(len(info)):
    print(heart.columns[i]+":\t\t\t"+info[i])
```

```
age:                    age
sex:                    1: male, 0: female
cp:                     chest pain type, 1: typical angina, 2: atypical angi
na, 3: non-anginal pain, 4: asymptomatic
trestbps:                       resting blood pressure
chol:                    serum cholestoral in mg/dl
fbs:                    fasting blood sugar > 120 mg/dl
restecg:                        resting electrocardiographic results (values
0,1,2)
thalach:                         maximum heart rate achieved
exang:                  exercise induced angina
oldpeak:                        oldpeak = ST depression induced by exercise
relative to rest
slope:                  the slope of the peak exercise ST segment
ca:                     number of major vessels (0-3) colored by flourosopy
thal:                   thal: 3 = normal; 6 = fixed defect; 7 = reversable d
efect
```

In [7]:

```
heart['target']
```

Out[7]:

```
0      1
1      1
2      1
3      1
4      1
5      1
6      1
7      1
8      1
9      1
10     1
11     1
12     1
13     1
14     1
15     1
16     1
17     1
18     1
19     1
20     1
21     1
22     1
23     1
24     1
25     1
26     1
27     1
28     1
29     1
      ..
273    0
274    0
275    0
276    0
277    0
278    0
279    0
280    0
281    0
282    0
283    0
284    0
285    0
286    0
287    0
288    0
289    0
290    0
291    0
292    0
293    0
294    0
295    0
```

```
296    0
297    0
298    0
299    0
300    0
301    0
302    0
Name: target, dtype: int64
```

In [8]:

```
heart.groupby('target').size()
```

Out[8]:

```
target
0    138
1    165
dtype: int64
```

In [9]:

```
heart.groupby('target').sum()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
target												
0	7811	114	66	18547	34650	22	62	19196	76	218.8	161	16
1	8662	93	227	21335	39968	23	98	26147	23	96.2	263	60

In [10]:

```
heart.shape
```

Out[10]:

```
(303, 14)
```

In [11]:

```
heart.size
```

Out[11]:

```
4242
```

Statistical Info of the data set

In [12]:

```
heart.describe()
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.52805
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.52586
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.00000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.00000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.00000

In [13]:

```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

In [14]:

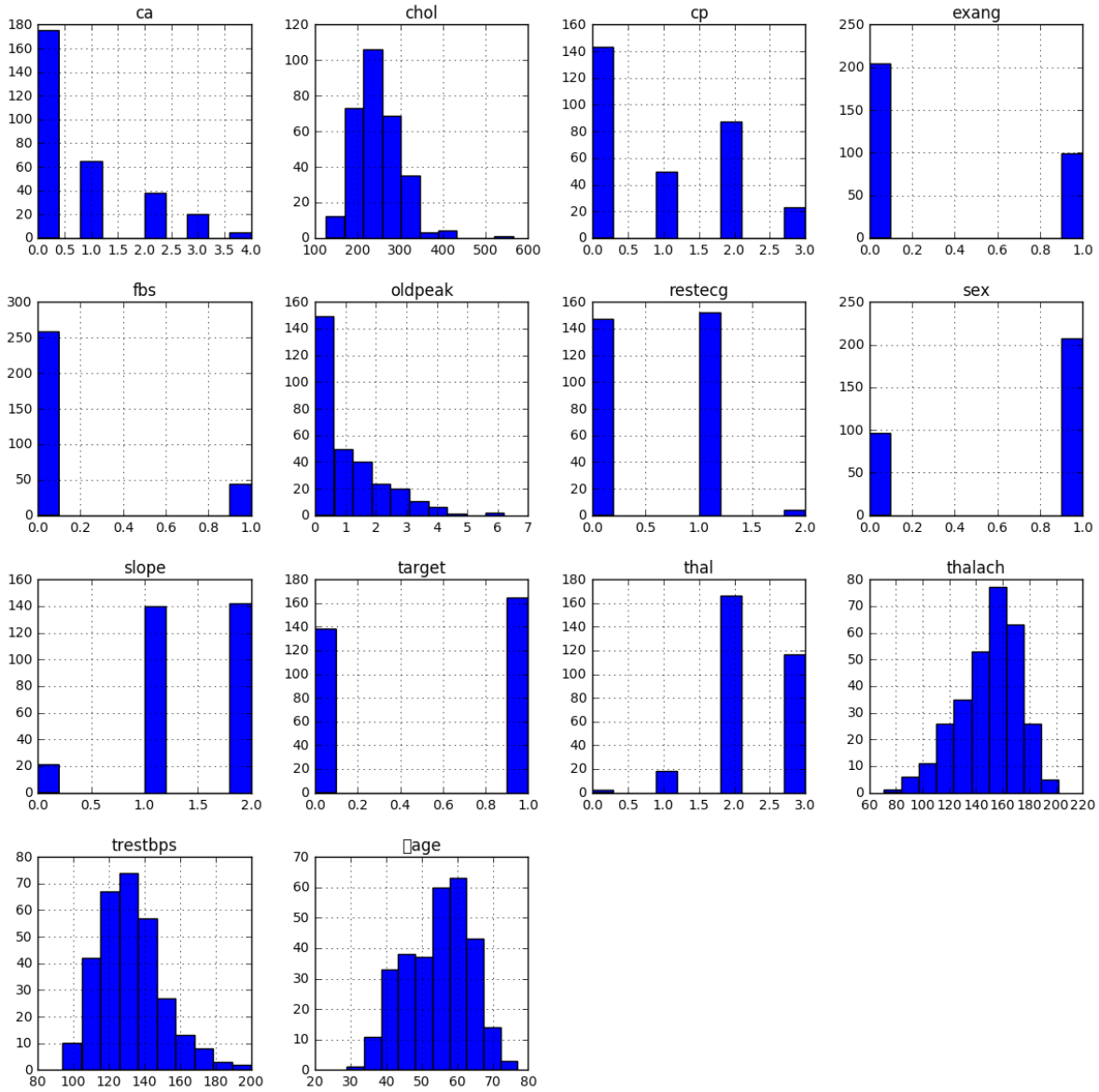
```
heart['target'].unique()
```

Out[14]:

```
array([1, 0], dtype=int64)
```

In [15]:

```
heart.hist(figsize=(14, 14))
plt.show()
```



In [17]:

```
numeric_columns = ['trestbps', 'chol', 'thalach', 'age', 'oldpeak']
heart['target']
```

Out[17]:

```
0      1
1      1
2      1
3      1
4      1
5      1
6      1
7      1
8      1
9      1
10     1
11     1
12     1
13     1
14     1
15     1
16     1
17     1
18     1
19     1
20     1
21     1
22     1
23     1
24     1
25     1
26     1
27     1
28     1
29     1
      ..
273    0
274    0
275    0
276    0
277    0
278    0
279    0
280    0
281    0
282    0
283    0
284    0
285    0
286    0
287    0
288    0
289    0
290    0
291    0
292    0
293    0
294    0
295    0
```

```
296    0
297    0
298    0
299    0
300    0
301    0
302    0
Name: target, dtype: int64
```

In [18]:

```
target_temp = heart.target.value_counts()

print(target_temp)
```

```
1    165
0    138
Name: target, dtype: int64
```

In [19]:

```
fig = plt.gcf()
fig.set_size_inches(8, 6)
plt.show()
```

```
<matplotlib.figure.Figure at 0x20640317f28>
```

Data Pre-Processing

In [20]:

```
heart['target'].value_counts()
```

Out[20]:

```
1    165
0    138
Name: target, dtype: int64
```

In [21]:

```
heart['target'].isnull()
```

Out[21]:

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
       ...
273    False
274    False
275    False
276    False
277    False
278    False
279    False
280    False
281    False
282    False
283    False
284    False
285    False
286    False
287    False
288    False
289    False
290    False
291    False
292    False
293    False
294    False
295    False
296    False
```

```
297    False
298    False
299    False
300    False
301    False
302    False
Name: target, dtype: bool
```

In [22]:

```
heart['target'].sum()
```

Out[22]:

```
165
```

In [23]:

```
heart['target'].unique()
```

Out[23]:

```
array([1, 0], dtype=int64)
```

In [24]:

```
heart.isnull().sum()
```

Out[24]:

```
age        0
sex        0
cp         0
trestbps   0
chol       0
fbs        0
restecg    0
thalach    0
exang      0
oldpeak    0
slope      0
ca         0
thal       0
target     0
dtype: int64
```

Storing the data in X and y

In [25]:

```
X, y = heart.loc[:, :'thal'], heart.loc[:, 'target']
```

In [26]:

```
X
```

Out[26]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2
14	58	0	3	150	283	1	0	162	0	1.0	2	0	2
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2
16	58	0	2	120	340	0	1	172	0	0.0	2	0	2
17	66	0	3	150	226	0	1	114	0	2.6	0	0	2
18	43	1	0	150	247	0	1	171	0	1.5	2	0	2
19	69	0	3	140	239	0	1	151	0	1.8	2	2	2
20	59	1	0	135	234	0	1	161	0	0.5	1	0	3
21	44	1	2	130	233	0	1	179	1	0.4	2	0	2
22	42	1	0	140	226	0	1	178	0	0.0	2	0	2
23	61	1	2	150	243	1	1	137	1	1.0	1	0	2
24	40	1	3	140	199	0	1	178	1	1.4	2	0	3
25	71	0	1	160	302	0	1	162	0	0.4	2	2	2
26	59	1	2	150	212	1	1	157	0	1.6	2	0	2
27	51	1	2	110	175	0	1	123	0	0.6	2	0	2
28	65	0	2	140	417	1	0	157	0	0.8	2	1	2
29	53	1	2	130	197	1	0	152	0	1.2	0	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
273	58	1	0	100	234	0	1	156	0	0.1	2	1	3
274	47	1	0	110	275	0	0	118	1	1.0	1	1	2
275	52	1	0	125	212	0	1	168	0	1.0	2	2	3
276	58	1	0	146	218	0	1	105	0	2.0	1	1	3
277	57	1	1	124	261	0	1	141	0	0.3	2	0	3
278	58	0	1	136	319	1	0	152	0	0.0	2	2	2
279	61	1	0	138	166	0	0	125	1	3.6	1	1	2
280	42	1	0	136	315	0	1	125	1	1.8	1	0	1
281	52	1	0	128	204	1	1	156	1	1.0	1	0	0
282	59	1	2	126	218	1	1	134	0	2.2	1	1	1
283	40	1	0	152	223	0	1	181	0	0.0	2	0	3
284	61	1	0	140	207	0	0	138	1	1.9	2	1	3
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2
287	57	1	1	154	232	0	0	164	0	0.0	2	1	2
288	57	1	0	110	335	0	1	143	1	3.0	1	1	3
289	55	0	0	128	205	0	2	130	1	2.0	1	1	3
290	61	1	0	148	203	0	1	161	0	0.0	2	1	3
291	58	1	0	114	318	0	2	140	0	4.4	0	3	1
292	58	0	0	170	225	1	0	146	1	2.8	1	2	1
293	67	1	2	152	212	0	0	150	0	0.8	1	0	3
294	44	1	0	120	169	0	1	144	1	2.8	0	0	1
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2
297	59	1	0	164	176	1	0	90	0	1.0	1	2	1
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

In [27]:

```
y
```

Out[27]:

```
0      1
1      1
2      1
3      1
4      1
5      1
6      1
7      1
8      1
9      1
10     1
11     1
12     1
13     1
14     1
15     1
16     1
17     1
18     1
19     1
20     1
21     1
22     1
23     1
24     1
25     1
26     1
27     1
28     1
29     1
      ..
273    0
274    0
275    0
276    0
277    0
278    0
279    0
280    0
281    0
282    0
283    0
284    0
285    0
286    0
287    0
288    0
289    0
290    0
291    0
292    0
293    0
294    0
295    0
```

```
296    0
297    0
298    0
299    0
300    0
301    0
302    0
Name: target, dtype: int64
```

In [28]:

```
X.shape
```

Out[28]:

```
(303, 13)
```

In [29]:

```
y.shape
```

Out[29]:

```
(303,)
```

In [30]:

```
X = heart.drop(['target'], axis=1)
```

In [31]:

```
X
```

Out[31]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2
14	58	0	3	150	283	1	0	162	0	1.0	2	0	2
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2
16	58	0	2	120	340	0	1	172	0	0.0	2	0	2
17	66	0	3	150	226	0	1	114	0	2.6	0	0	2
18	43	1	0	150	247	0	1	171	0	1.5	2	0	2
19	69	0	3	140	239	0	1	151	0	1.8	2	2	2
20	59	1	0	135	234	0	1	161	0	0.5	1	0	3
21	44	1	2	130	233	0	1	179	1	0.4	2	0	2
22	42	1	0	140	226	0	1	178	0	0.0	2	0	2
23	61	1	2	150	243	1	1	137	1	1.0	1	0	2
24	40	1	3	140	199	0	1	178	1	1.4	2	0	3
25	71	0	1	160	302	0	1	162	0	0.4	2	2	2
26	59	1	2	150	212	1	1	157	0	1.6	2	0	2
27	51	1	2	110	175	0	1	123	0	0.6	2	0	2
28	65	0	2	140	417	1	0	157	0	0.8	2	1	2
29	53	1	2	130	197	1	0	152	0	1.2	0	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
273	58	1	0	100	234	0	1	156	0	0.1	2	1	3
274	47	1	0	110	275	0	0	118	1	1.0	1	1	2
275	52	1	0	125	212	0	1	168	0	1.0	2	2	3
276	58	1	0	146	218	0	1	105	0	2.0	1	1	3
277	57	1	1	124	261	0	1	141	0	0.3	2	0	3
278	58	0	1	136	319	1	0	152	0	0.0	2	2	2
279	61	1	0	138	166	0	0	125	1	3.6	1	1	2
280	42	1	0	136	315	0	1	125	1	1.8	1	0	1
281	52	1	0	128	204	1	1	156	1	1.0	1	0	0
282	59	1	2	126	218	1	1	134	0	2.2	1	1	1
283	40	1	0	152	223	0	1	181	0	0.0	2	0	3
284	61	1	0	140	207	0	0	138	1	1.9	2	1	3
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2
287	57	1	1	154	232	0	0	164	0	0.0	2	1	2
288	57	1	0	110	335	0	1	143	1	3.0	1	1	3
289	55	0	0	128	205	0	2	130	1	2.0	1	1	3
290	61	1	0	148	203	0	1	161	0	0.0	2	1	3
291	58	1	0	114	318	0	2	140	0	4.4	0	3	1
292	58	0	0	170	225	1	0	146	1	2.8	1	2	1
293	67	1	2	152	212	0	0	150	0	0.8	1	0	3
294	44	1	0	120	169	0	1	144	1	2.8	0	0	1
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2
297	59	1	0	164	176	1	0	90	0	1.0	1	2	1
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

Splitting the data into train and test for training and

testing

In [77]:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=10, test_size=0.2, shuffle=True)
```

In [78]:

```
X_test
```

Out[78]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tl
246	56	0	0	134	409	0	0	150	1	1.9	1	2	3
183	58	1	2	112	230	0	0	165	0	2.5	1	1	3
229	64	1	2	125	309	0	1	131	1	1.8	1	0	3
126	47	1	0	112	204	0	1	143	0	0.1	2	0	2
184	50	1	0	150	243	0	0	128	0	2.6	1	0	3
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
59	57	0	0	128	303	0	0	159	0	0.0	2	1	2
194	60	1	2	140	185	0	0	155	0	3.0	1	0	2
132	42	1	1	120	295	0	1	162	0	0.0	2	0	2
175	40	1	0	110	167	0	0	114	1	2.0	1	0	3
162	41	1	1	120	157	0	1	182	0	0.0	2	0	2
181	65	0	0	150	225	0	0	114	0	1.0	1	3	3
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2
219	48	1	0	130	256	1	0	150	1	0.0	2	2	3
92	52	1	2	138	223	0	1	169	0	0.0	2	4	2
195	59	1	0	170	326	0	0	140	1	3.4	0	0	3
198	62	1	0	120	267	0	1	99	1	1.8	1	2	3
24	40	1	3	140	199	0	1	178	1	1.4	2	0	3
249	69	1	2	140	254	0	0	146	0	2.0	1	3	3
139	64	1	0	128	263	0	1	105	1	0.2	1	1	3
26	59	1	2	150	212	1	1	157	0	1.6	2	0	2
287	57	1	1	154	232	0	0	164	0	0.0	2	1	2
64	58	1	2	140	211	1	0	165	0	0.0	2	0	2
202	58	1	0	150	270	0	0	111	1	0.8	2	0	3
240	70	1	2	160	269	0	1	112	1	2.9	1	1	3
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3
186	60	1	0	130	253	0	1	144	1	1.4	2	1	3
127	67	0	2	152	277	0	1	172	0	0.0	2	1	2
191	58	1	0	128	216	0	0	131	1	2.2	1	3	3
...	...	...	...	...	...	...	...	...	...	...	...	...	..

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tl
52	62	1	2	130	231	0	1	146	0	1.8	1	3	3
279	61	1	0	138	166	0	0	125	1	3.6	1	1	2
138	57	1	0	110	201	0	1	126	1	1.5	1	0	1
193	60	1	0	145	282	0	0	142	1	2.8	1	2	3
207	60	0	0	150	258	0	0	157	0	2.6	1	2	3
25	71	0	1	160	302	0	1	162	0	0.4	2	2	2
270	46	1	0	120	249	0	0	144	0	0.8	2	0	3
105	68	0	2	120	211	0	0	115	0	1.5	1	0	2
264	54	1	0	110	206	0	0	108	1	0.0	1	1	2
289	55	0	0	128	205	0	2	130	1	2.0	1	1	3
169	53	1	0	140	203	1	0	155	1	3.1	0	0	3
43	53	0	0	130	264	0	0	143	0	0.4	1	0	2
188	50	1	2	140	233	0	1	163	0	0.6	1	1	3
80	41	1	2	112	250	0	1	179	0	0.0	2	0	2
273	58	1	0	100	234	0	1	156	0	0.1	2	1	3
259	38	1	3	120	231	0	1	182	1	3.8	1	0	3
56	48	1	0	122	222	0	0	186	0	0.0	2	0	2
98	43	1	2	130	315	0	1	162	0	1.9	2	1	2
106	69	1	3	160	234	1	0	131	0	0.1	1	1	2
173	58	1	2	132	224	0	0	173	0	3.2	2	2	3
87	46	1	1	101	197	1	1	156	0	0.0	2	0	3
244	56	1	0	132	184	0	0	105	1	2.1	1	1	1
213	61	0	0	145	307	0	0	146	1	1.0	1	0	3
36	54	0	2	135	304	1	1	170	0	0.0	2	0	2
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2
277	57	1	1	124	261	0	1	141	0	0.3	2	0	3
121	59	1	0	138	271	0	0	182	0	0.0	2	0	2
187	54	1	0	124	266	0	0	109	1	2.2	1	1	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
283	40	1	0	152	223	0	1	181	0	0.0	2	0	3

61 rows × 13 columns

In [79]:

```
y_test
```

Out[79]:

```
246    0
183    0
229    0
126    1
184    0
1      1
59     1
194    0
132    1
175    0
162    1
181    0
296    0
164    1
219    0
92     1
195    0
198    0
24     1
249    0
139    1
26     1
287    0
64     1
202    0
240    0
285    0
186    0
127    1
191    0
      ..
52     1
279    0
138    1
193    0
207    0
25     1
270    0
105    1
264    0
289    0
169    0
43     1
188    0
80     1
273    0
259    0
56     1
98     1
106    1
173    0
87     1
244    0
213    0
36     1
```

```
10     1
277    0
121    1
187    0
301    0
283    0
Name: target, dtype: int64
```

In [80]:

```
print("train_set_x shape: " + str(X_train.shape))
print("train_set_y shape: " + str(y_train.shape))
print("test_set_x shape: " + str(X_test.shape))
print("test_set_y shape: " + str(y_test.shape))
```

```
train_set_x shape: (242, 13)
train_set_y shape: (242,)
test_set_x shape: (61, 13)
test_set_y shape: (61,)
```

MODEL

1. Decision Tree Classifier

In [81]:

```
Catagory = ['No,You do not have Heart Disease...',
            'Yes you have Heart Disease...']
```

In [82]:

```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

Out[82]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

Finding the accuracy in Decision Tree Model

In [83]:

```
prediction = dt.predict(X_test)
accuracy_dt = accuracy_score(y_test, prediction)*100
```

In [84]:

```
accuracy_dt
```

Out[84]:

```
75.409836065573771
```

We got 75.40 % accuracy in Decision Tree model

In [85]:

```
print("Accuracy on training set: {:.3f}".format(dt.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(dt.score(X_test, y_test)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.754
```

In [86]:

```
y_test
```

Out[86]:

```
246    0
183    0
229    0
126    1
184    0
1      1
59     1
194    0
132    1
175    0
162    1
181    0
296    0
164    1
219    0
92     1
195    0
198    0
24     1
249    0
139    1
26     1
287    0
64     1
202    0
240    0
285    0
186    0
127    1
191    0
      ..
52     1
279    0
138    1
193    0
207    0
25     1
270    0
105    1
264    0
289    0
169    0
43     1
188    0
80     1
273    0
259    0
56     1
98     1
106    1
173    0
87     1
244    0
213    0
36     1
```

```
10     1
277    0
121    1
187    0
301    0
283    0
Name: target, dtype: int64
```

In [87]:

```
prediction
```

Out[87]:

```
array([1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0], dtype=int64)
```

Testing with new data to verify result

In [88]:

```
X_DT = np.array([[35, 1, 0, 126, 282, 0, 0, 156, 1, 0, 2, 0, 3]])
X_DT_prediction = dt.predict(X_DT)
```

In [89]:

```
X_DT_prediction[0]
```

Out[89]:

```
0
```

Yeah, Get The Result of new feeded data Here !!!

In [90]:

```
print(Catagory[int(X_DT_prediction[0])])
```

```
No,You do not have Heart Disease...
```

Feature Importance in Decision Trees

In [91]:

```
print("Feature importances:\n{}".format(dt.feature_importances_))
```

```
Feature importances:
[ 0.04165356  0.03389848  0.30021653  0.11653577  0.05483563  0.          0.
  0.05844875  0.04868059  0.11186649  0.07039819  0.163466    0.        ]
```

In [92]:

```
def plot_feature_importances_diabetes(model):
    plt.figure(figsize=(8, 6))
    n_features = 13
    plt.barh(range(n_features), model.feature_importances_, align='center')
    plt.yticks(np.arange(n_features), X)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)


plot_feature_importances_diabetes(dt)
plt.savefig('feature_importance')
```

2. KNN

Training with KNN

In [111]:

```
sc = StandardScaler().fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

In [112]:

```
X_test_std
```

Out[112]:

```
array([[ 0.18733254, -1.37147817, -0.96052267,  0.15197298,  3.06663801,
        -0.43159531, -1.01054031, -0.01372906,  1.50674161,  0.7957577 ,
        -0.63678177,  1.30209953,  1.19898626],
       [ 0.40633967,  0.72914029,  0.97653138, -1.06472495, -0.33003859,
        -0.43159531, -1.01054031,  0.6507574 , -0.6636838 ,  1.31881798,
        -0.63678177,  0.30492204,  1.19898626],
       [ 1.06336104,  0.72914029,  0.97653138, -0.34576709,  1.16905332,
        -0.43159531,  0.90001247, -0.85541191,  1.50674161,  0.70858098,
        -0.63678177, -0.69225545,  1.19898626],
       [-0.79819952,  0.72914029, -0.96052267, -1.06472495, -0.82341062,
        -0.43159531,  0.90001247, -0.32382274, -0.6636838 , -0.77342317,
         0.96843894, -0.69225545, -0.43109618],
       [-0.46968884,  0.72914029, -0.96052267,  1.03684419, -0.08335258,
        -0.43159531, -1.01054031, -0.9883092 , -0.6636838 ,  1.4059947 ,
        -0.63678177, -0.69225545,  1.19898626],
       [-1.89323515,  0.72914029,  0.97653138, -0.06924483,  0.04947834,
        -0.43159531,  0.90001247,  1.62533754, -0.6636838 ,  2.19058513,
        -2.24200249, -0.69225545, -0.43109618],
```

In [113]:

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_std, y_train)
```

Out[113]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

Calculating Accuracy in KNN Model

In [102]:

```
prediction_knn = knn.predict(X_test_std)
accuracy_knn = accuracy_score(y_test, prediction_knn)*100
```

In [103]:

```
accuracy_knn
```

Out[103]:

```
85.245901639344254
```

We got 85.2 % Accurcy in KNN Model

In [104]:

```
print("Accuracy on training set: {:.3f}".format(knn.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(knn.score(X_test, y_test)))
```

```
Accuracy on training set: 0.574
Accuracy on test set: 0.426
```

Let's See how Accuracy varies with the value of k in KNN Model

In [105]:

```
k_range = range(1, 26)
scores = {}
scores_list = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_std, y_train)
    prediction_knn = knn.predict(X_test_std)
    scores[k] = accuracy_score(y_test, prediction_knn)
    scores_list.append(accuracy_score(y_test, prediction_knn))
scores
```

Out[105]:

```
{1: 0.77049180327868849,
 2: 0.77049180327868849,
 3: 0.80327868852459017,
 4: 0.81967213114754101,
 5: 0.85245901639344257,
 6: 0.85245901639344257,
 7: 0.80327868852459017,
 8: 0.78688524590163933,
 9: 0.78688524590163933,
 10: 0.81967213114754101,
 11: 0.80327868852459017,
 12: 0.81967213114754101,
 13: 0.81967213114754101,
 14: 0.83606557377049184,
 15: 0.78688524590163933,
 16: 0.80327868852459017,
 17: 0.80327868852459017,
 18: 0.80327868852459017,
 19: 0.80327868852459017,
 20: 0.80327868852459017,
 21: 0.80327868852459017,
 22: 0.81967213114754101,
 23: 0.80327868852459017,
 24: 0.81967213114754101,
 25: 0.80327868852459017}
```

In the above output we can see that the accuracy is maximum at k=5

In [106]:

```
plt.plot(k_range, scores_list)
```

Out[106]:

```
[<matplotlib.lines.Line2D at 0x2064014c6a0>]
```

Input the heart data below to get the result using KNN Model

In [107]:

```
X_knn = np.array([[63, 1, 3, 145, 233, 1, 0, 150, 0, 2.3, 0, 0, 1]])
X_knn_std = sc.transform(X_knn)
X_knn_prediction = dt.predict(X_knn)
```

In [108]:

```
X_knn_std
```

Out[108]:

```
array([[ 0.95385748,  0.72914029,  1.9450584 ,  0.76032194, -0.27311105,
         2.31698534, -1.01054031, -0.01372906, -0.6636838 ,  1.14446455,
        -2.24200249, -0.69225545, -2.06117863]])
```

In [109]:

```
(X_knn_prediction[0])
```

Out[109]:

```
1
```

Result of the user's input using KNN Model

In [110]:

```
print(Catagory[int(X_knn_prediction[0])])
```

```
Yes you have Heart Disease...
```

Type *Markdown* and LaTeX: α^2