Heart Disease Dataset Analysis

Import module

In [1]:

```
import pandas as pd
import numpy as np
import os
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn as sl
```

Importing dataset

In [2]:

```
data=pd.read_csv("C:/Users/priya/OneDrive/Desktop/heart.csv")
```

shape & size of dataset

In [3]:

```
data.shape
```

Out[3]:

```
(303, 14)
```

In [4]:

```
data.size
```

Out[4]:

```
4242
```

Description about dataset

In [5]:

```
data.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
age	303.0	54.366337	9.082101	29.0	47.5	55.0	61.0	77.0
sex	303.0	0.683168	0.466011	0.0	0.0	1.0	1.0	1.0
cp	303.0	0.966997	1.032052	0.0	0.0	1.0	2.0	3.0
rest bp	303.0	131.623762	17.538143	94.0	120.0	130.0	140.0	200.0
chol	303.0	246.264026	51.830751	126.0	211.0	240.0	274.5	564.0
fbs	303.0	0.148515	0.356198	0.0	0.0	0.0	0.0	1.0
restecg	303.0	0.528053	0.525860	0.0	0.0	1.0	1.0	2.0
max H.R	303.0	149.646865	22.905161	71.0	133.5	153.0	166.0	202.0
exng	303.0	0.326733	0.469794	0.0	0.0	0.0	1.0	1.0
oldpeak	303.0	1.039604	1.161075	0.0	0.0	0.8	1.6	6.2
slp	303.0	1.399340	0.616226	0.0	1.0	1.0	2.0	2.0
M.V no.	303.0	0.729373	1.022606	0.0	0.0	0.0	1.0	4.0
thall	303.0	2.313531	0.612277	0.0	2.0	2.0	3.0	3.0
output	303.0	0.544554	0.498835	0.0	0.0	1.0	1.0	1.0

In [6]:

```
data.head()
```

Out[6]:

	age	sex	cp	rest bp	chol	fbs	restecg	max H.R	exng	oldpeak	slp	M.V no.	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [7]:

```
data.tail()
```

Out[7]:

	age	sex	cp	rest bp	chol	fbs	restecg	max H.R	exng	oldpeak	slp	M.V no.	thall	output
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

In [8]:

```
data.columns
```

Out[8]:

```
Index(['age', 'sex', 'cp', 'rest bp', 'chol', 'fbs', 'restecg', 'max H.R',
       'exng', 'oldpeak', 'slp', 'M.V no.', 'thall', 'output'],
      dtype='object')
```

In [9]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   rest bp   303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   max H.R   303 non-null    int64
 8   exng      303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64
 11  M.V no.   303 non-null    int64
 12  thall     303 non-null    int64
 13  output    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Checking of missing value of dataset

In [10]:

```
data.isnull().sum()
```

Out[10]:

```
age         0
sex         0
cp          0
rest bp     0
chol        0
fbs         0
restecg     0
max H.R     0
exng        0
oldpeak     0
slp         0
M.V no.     0
thall       0
output      0
dtype: int64
```

In [11]:

```
data.isnull()
```

Out[11]:

	age	sex	cp	rest bp	chol	fbs	restecg	max H.R	exng	oldpeak	slp	M.V no.	thall	output
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	False	False	False	False	False	False	False	False	False	False	False	False	False	False
299	False	False	False	False	False	False	False	False	False	False	False	False	False	False
300	False	False	False	False	False	False	False	False	False	False	False	False	False	False
301	False	False	False	False	False	False	False	False	False	False	False	False	False	False
302	False	False	False	False	False	False	False	False	False	False	False	False	False	False

303 rows × 14 columns

Attributes counts

In [12]:

```
data['fbs'].value_counts()
```

Out[12]:

```
0    258
1     45
Name: fbs, dtype: int64
```

In [13]:

```
data['sex'].value_counts()
```

Out[13]:

```
1    207
0     96
Name: sex, dtype: int64
```

In [14]:

```
data['exng'].value_counts()
```

Out[14]:

```
0    204
1     99
Name: exng, dtype: int64
```

Target function count

In [15]:

```
data['output'].value_counts()
```

Out[15]:

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

Exploratory dataset analysis

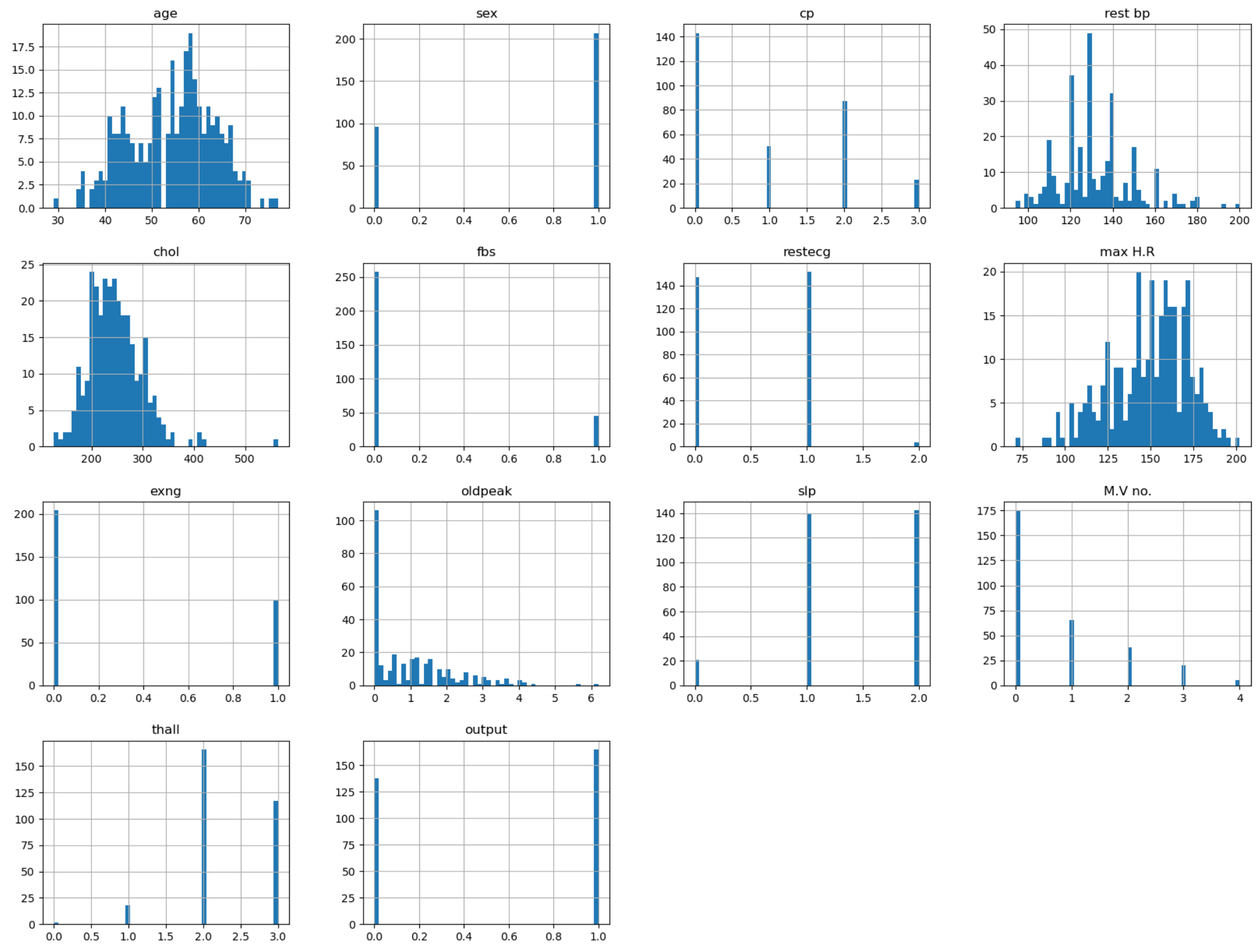
Histogram plotting of attributes

In [16]:

```
data.hist(bins=50, figsize=(20,15))
```

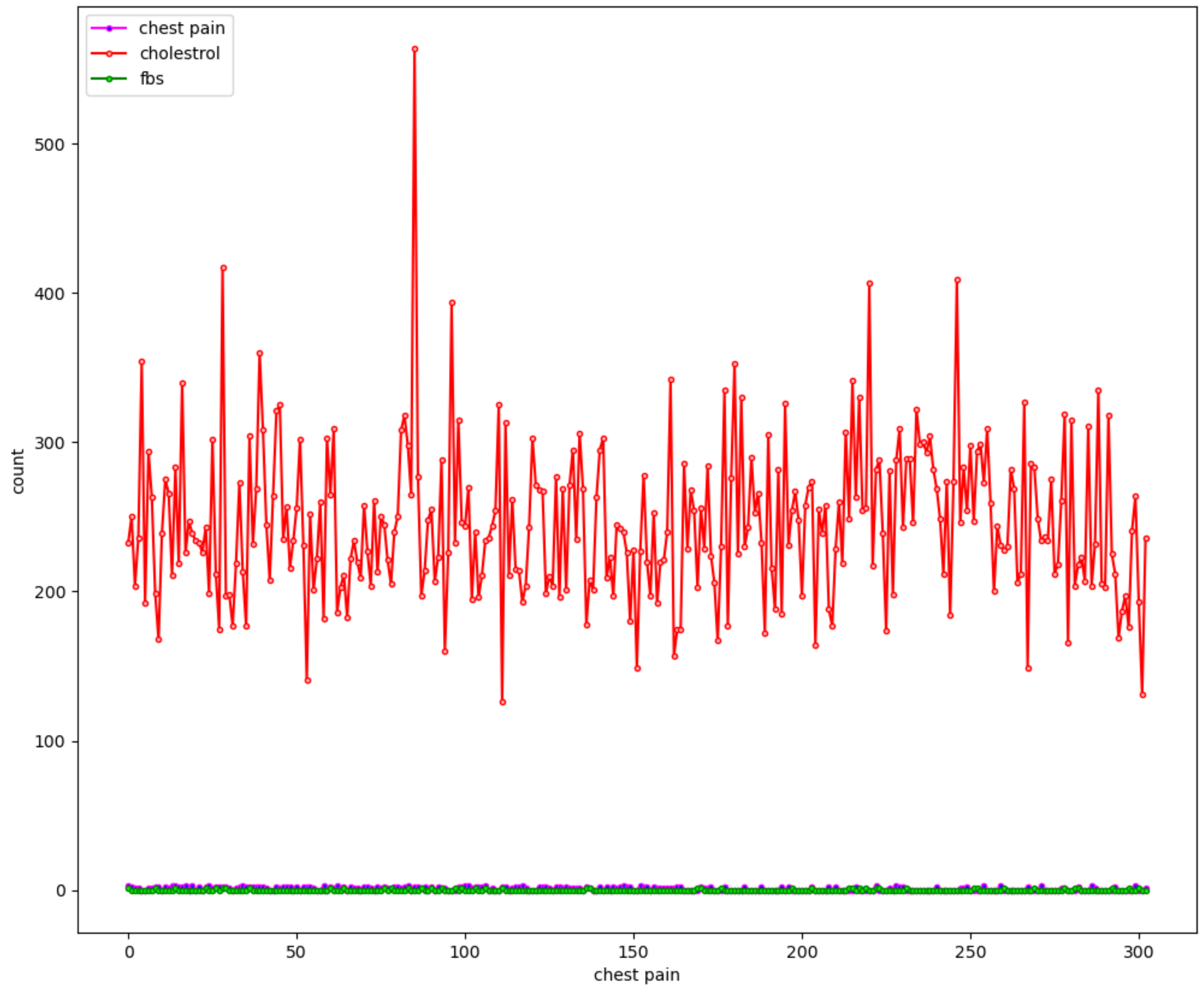
Out[16]:

```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sex'}>,
        <AxesSubplot:title={'center':'cp'}>,
        <AxesSubplot:title={'center':'rest bp'}>],
       [<AxesSubplot:title={'center':'chol'}>,
        <AxesSubplot:title={'center':'fbs'}>,
        <AxesSubplot:title={'center':'restecg'}>,
        <AxesSubplot:title={'center':'max H.R'}>],
       [<AxesSubplot:title={'center':'exng'}>,
        <AxesSubplot:title={'center':'oldpeak'}>,
        <AxesSubplot:title={'center':'slp'}>,
        <AxesSubplot:title={'center':'M.V no.'}>],
       [<AxesSubplot:title={'center':'thall'}>,
        <AxesSubplot:title={'center':'output'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



In [17]:

```
plt.figure(figsize=(12,10))
plt.plot(data.cp,label='chest pain',color='magenta',marker='.',markerfacecolor='blue')
plt.plot(data.chol,label='cholestrol',color='red',marker='.',markerfacecolor='pink')
plt.plot(data.fbs,label='fbs',color='green',marker='.',markerfacecolor='lime')
#plt.plot(data.restbp,label='rest bp',color='red',marker='.',markerfacecolor='yellow')
#plt.plot(data.maxH.R,label='max H.R',color='green',marker='.',markerfacecolor='brown')
plt.xlabel('chest pain')
plt.ylabel('count')
plt.legend(loc=2);
```



In [ ]:

```

```

Correlation matrix

In [18]:

```
corr_matrix = data.corr()
```

In [19]:

```
corr_matrix['age'].sort_values(ascending = False)
```

Out[19]:

```
age         1.000000
rest bp     0.279351
M.V no.     0.276326
chol        0.213678
oldpeak     0.210013
fbs         0.121308
exng        0.096801
thall       0.068001
cp         -0.068653
sex        -0.098447
restecg    -0.116211
slp        -0.168814
output     -0.225439
max H.R    -0.398522
Name: age, dtype: float64
```

In [20]:

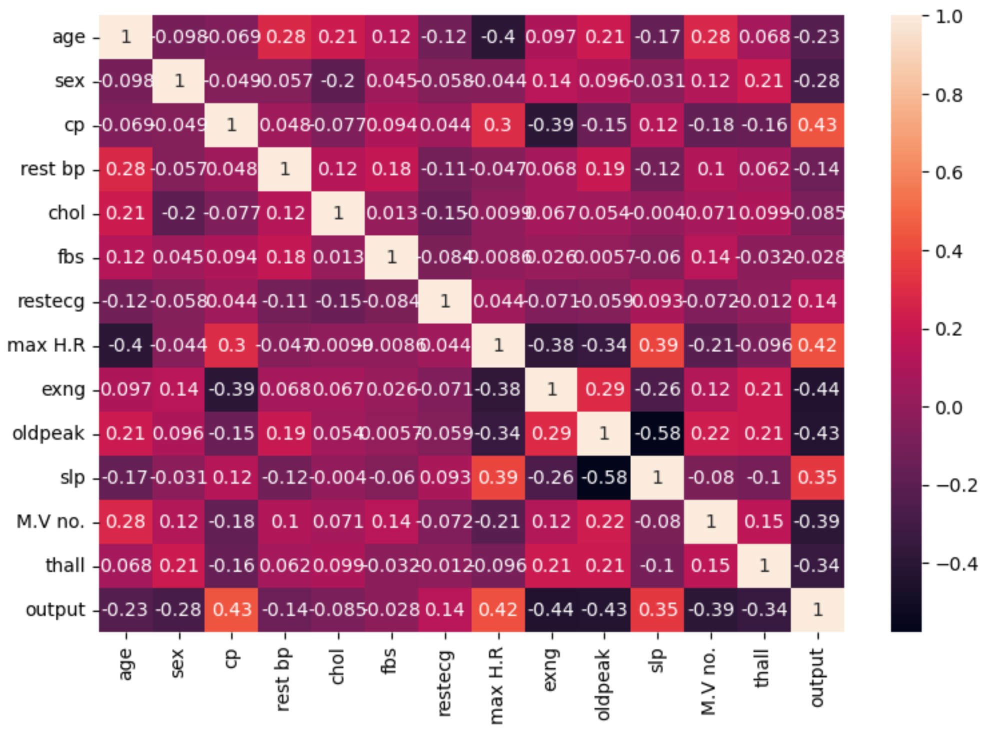
```
data.corr()
```

Out[20]:

	age	sex	cp	rest bp	chol	fbs	restecg	max H.R	exng	oldpeak	slp	M.V no.	thall	out
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326	0.068001	-0.225
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433
rest bp	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803	-0.085
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137
max H.R	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439	0.421
exng	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430
slp	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345
M.V no.	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391
thall	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344
output	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000

In [21]:

```
corr=data.corr()
plt.subplots(figsize=(9,6))
sns.heatmap(corr,annot=True);
```



Checking of duplicate values in dataset

In [22]:

```
data_dup = data.duplicated().any()
```

In [23]:

```
data_dup
```

Out[23]:

```
True
```

In [24]:

```
data = data.drop_duplicates()
```

In [25]:

```
data_dup = data.duplicated().any()
```

In [26]:

```
data_dup
```

Out[26]:

```
False
```

In [27]:

```
x = data.drop('output',axis=1)
```

In [28]:

```
y = data['output']
```

In [29]:

```
from sklearn.model_selection import train_test_split
```

In [30]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [31]:

```
x_train
```

Out[31]:

	age	sex	cp	rest bp	chol	fbs	restecg	max H.R	exng	oldpeak	slp	M.V no.	thall
132	42	1	1	120	295	0	1	162	0	0.0	2	0	2
203	68	1	2	180	274	1	0	150	1	1.6	1	0	3
197	67	1	0	125	254	1	1	163	0	0.2	1	2	3
75	55	0	1	135	250	0	0	161	0	1.4	1	0	2
177	64	1	2	140	335	0	1	158	0	0.0	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
189	41	1	0	110	172	0	0	158	0	0.0	2	0	3
71	51	1	2	94	227	0	1	154	1	0.0	2	1	3
106	69	1	3	160	234	1	0	131	0	0.1	1	1	2
271	61	1	3	134	234	0	1	145	0	2.6	1	2	2
102	63	0	1	140	195	0	1	179	0	0.0	2	2	2

241 rows × 13 columns

Data processing

In [32]:

```
cate_val = []
cont_val = []
for column in data.columns:
    if data[column].nunique()<=10:
        cate_val.append(column)
    else:
        cont_val.append(column)
```

In [33]:

```
cate_val
```

Out[33]:

```
['sex', 'cp', 'fbs', 'restecg', 'exng', 'slp', 'M.V no.', 'thall', 'output']
```

In [34]:

```
cont_val
```

Out[34]:

```
['age', 'rest bp', 'chol', 'max H.R', 'oldpeak']
```

Encoding categorical data

In [35]:

```
cate_val
```

Out[35]:

```
['sex', 'cp', 'fbs', 'restecg', 'exng', 'slp', 'M.V no.', 'thall', 'output']
```

In [36]:

```
data['cp'].unique()
```

Out[36]:

```
array([3, 2, 1, 0], dtype=int64)
```

In [37]:

```
cate_val.remove('sex')
cate_val.remove('output')
data = pd.get_dummies(data,columns = cate_val,drop_first = True)
```

In [38]:

```
data.head()
```

Out[38]:

	age	sex	rest bp	chol	max H.R	oldpeak	output	cp_1	cp_2	cp_3	...	exng_1	slp_1	slp_2	M.V no._1	M.V no._2	M.V no._3	M.V no._4	thall_1	thall_2	thall_3
0	63	1	145	233	150	2.3	1	0	0	1	...	0	0	0	0	0	0	0	1	0	0
1	37	1	130	250	187	3.5	1	0	1	0	...	0	0	0	0	0	0	0	0	1	0
2	41	0	130	204	172	1.4	1	1	0	0	...	0	0	1	0	0	0	0	0	1	0
3	56	1	120	236	178	0.8	1	1	0	0	...	0	0	1	0	0	0	0	0	1	0
4	57	0	120	354	163	0.6	1	0	0	0	...	1	0	1	0	0	0	0	0	1	0

5 rows × 23 columns

Feature Scaling

In [39]:

```
data.head()
```

Out[39]:

	age	sex	rest bp	chol	max H.R	oldpeak	output	cp_1	cp_2	cp_3	...	exng_1	slp_1	slp_2	M.V no._1	M.V no._2	M.V no._3	M.V no._4	thall_1	thall_2	thall_3
0	63	1	145	233	150	2.3	1	0	0	1	...	0	0	0	0	0	0	0	1	0	0
1	37	1	130	250	187	3.5	1	0	1	0	...	0	0	0	0	0	0	0	0	1	0
2	41	0	130	204	172	1.4	1	1	0	0	...	0	0	1	0	0	0	0	0	1	0
3	56	1	120	236	178	0.8	1	1	0	0	...	0	0	1	0	0	0	0	0	1	0
4	57	0	120	354	163	0.6	1	0	0	0	...	1	0	1	0	0	0	0	0	1	0

5 rows × 23 columns

In [40]:

```
from sklearn.preprocessing import StandardScaler
```

In [41]:

```
st = StandardScaler()
data[cont_val] = st.fit_transform(data[cont_val])
```

In [42]:

```
data.head()
```

Out[42]:

	age	sex	rest bp	chol	max H.R	oldpeak	output	cp_1	cp_2	cp_3	...	exng_1	slp_1	slp_2	M.V no._1	M.V no._2	M.V no._3	M.V no._4	thall_1	thal
0	0.949794	1	0.764066	-0.261285	0.018826	1.084022	1	0	0	1	...	0	0	0	0	0	0	0	1	
1	-1.928548	1	-0.091401	0.067741	1.636979	2.118926	1	0	1	0	...	0	0	0	0	0	0	0	0	
2	-1.485726	0	-0.091401	-0.822564	0.980971	0.307844	1	1	0	0	...	0	0	1	0	0	0	0	0	
3	0.174856	1	-0.661712	-0.203222	1.243374	-0.209608	1	1	0	0	...	0	0	1	0	0	0	0	0	
4	0.285561	0	-0.661712	2.080602	0.587366	-0.382092	1	0	0	0	...	1	0	1	0	0	0	0	0	

5 rows × 23 columns

Splitting training and testing data

In [43]:

```
def split_train_test(data, test_ratio):
    np.random.seed(42)
    shuffled = np.random.permutation(len(data))
    print(shuffled)
    test_set_size = int(len(data)*test_ratio)
    test_indices = shuffled[:test_set_size]
    train_indices = shuffled[test_set_size:]
    return data.iloc[train_indices],data.iloc[test_indices]
```

In [44]:

```
train_set,test_set = split_train_test(data,0.2)
```

```
[179 228 111 246  60   9 119 223 267  33   5 101  45 175 118  46 125 192
 284 278 152 268 271  25 146 282 254  73 231 109 139 283 198  42  17 168
  76  90  24  57  92  77 137 116   7 251 280  78 291 232 219 255  63  82
 236 204 249 104 299 193 184 132 202 196  75 176  59  93   6 177  30  22
 258  56 242 114 286 281 197 158 164 244  84  66 113 167 250  19 143  79
 144 124  72  15  10 163 155  97  68 229  37  16 126 290 272  67 108  69
  31 178 154 230 294  18 185  96 183 148  86 253 288 206 287 170 234 211
  55 186 297 210 129  38 239 173 140 112 172 117 279 273 165 180 182   2
 115 147 181 120 215 262 127  74  29  83 248 107 157 208 133 194 221  65
 203  85 218 159  12  35  28 142 195 131 226  51  95 213 225  41  89 222
 136  26 295 141 238   0 285 274 100 261 103 171  98  36  61 150 264 233
 247  11 298 200 269  27 224   4 122  32 209 162 237 259 138  62 135 128
 292   8  70 266  64  44 240 156  40 123 277 216 153  23 263 110  81 207
 212  39 245 293 260 199  14  47  94 265 227 275 201 161  43 217 145 190
 220 256   3 105  53   1  49  80 205  34  91  52 241  13  88 166 296 134
 289 243  54  50 174 189 300 187 169  58  48 235 252  21 160 276 191 257
 149 130 151  99  87 214 121 301  20 188  71 106 270 102]
```

In [45]:

```
#print(f"Rows in train set:{len(train_set)}\nRows in test set:{len(test_set)}\n")
```

In [46]:

```
from sklearn.model_selection import train_test_split
train_set,test_set = train_test_split(data,test_size=0.2,random_state=42)
print(f"Rows in train set:{len(train_set)}\nRows in test set:{len(test_set)}\n")
```

```
Rows in train set:241
Rows in test set:61
```

In [47]:

```
data.columns
```

Out[47]:

```
Index(['age', 'sex', 'rest bp', 'chol', 'max H.R', 'oldpeak', 'output', 'cp_1',
       'cp_2', 'cp_3', 'fbs_1', 'restecg_1', 'restecg_2', 'exng_1', 'slp_1',
       'slp_2', 'M.V no._1', 'M.V no._2', 'M.V no._3', 'M.V no._4', 'thall_1',
       'thall_2', 'thall_3'],
      dtype='object')
```

In [48]:

```
X = data.drop('output',axis=1)
Y = data['output']
```

In [49]:

```
from sklearn.model_selection import train_test_split
```

In [50]:

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=42)
```

In [51]:

```
X_train
```

Out[51]:

	age	sex	rest bp	chol	max H.R	oldpeak	cp_1	cp_2	cp_3	fbs_1	...	exng_1	slp_1	slp_2	M.V no._1	M.V no._2	M.V no._3	M.V no._4	thall_1	th
132	-1.375021	1	-0.661712	0.938690	0.543632	-0.899544	1	0	0	0	...	0	0	1	0	0	0	0	0	
203	1.503322	1	2.760154	0.532247	0.018826	0.480328	0	1	0	1	...	1	1	0	0	0	0	0	0	
197	1.392616	1	-0.376556	0.145158	0.587366	-0.727060	0	0	0	1	...	0	1	0	0	1	0	0	0	
75	0.064151	0	0.193755	0.067741	0.499898	0.307844	1	0	0	0	...	0	1	0	0	0	0	0	0	
177	1.060500	1	0.478910	1.712868	0.368697	-0.899544	0	1	0	0	...	0	0	1	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
189	-1.485726	1	-1.232023	-1.441906	0.368697	-0.899544	0	0	0	0	...	0	0	1	0	0	0	0	0	
71	-0.378671	1	-2.144521	-0.377412	0.193761	-0.899544	0	1	0	0	...	1	0	1	1	0	0	0	0	
106	1.614027	1	1.619532	-0.241930	-0.812118	-0.813302	0	0	1	1	...	0	1	0	1	0	0	0	0	
271	0.728383	1	0.136724	-0.241930	-0.199843	1.342748	0	0	1	0	...	0	1	0	0	1	0	0	0	
102	0.949794	0	0.478910	-0.996754	1.287108	-0.899544	1	0	0	0	...	0	0	1	0	1	0	0	0	

241 rows × 22 columns

In [52]:

```
X_train.shape
```

Out[52]:

```
(241, 22)
```

In [53]:

```
Y_train
```

Out[53]:

```
132    1
203    0
197    0
75     1
177    0
      ..
189    0
71     1
106    1
271    0
102    1
Name: output, Length: 241, dtype: int64
```

In [54]:

```
Y_train.value_counts(normalize=True)*100
```

Out[54]:

```
1    54.771784
0    45.228216
Name: output, dtype: float64
```

In [55]:

```
Y_test.value_counts(normalize=True)*100
```

Out[55]:

```
1    52.459016
0    47.540984
Name: output, dtype: float64
```

In [56]:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.5,random_state=1,stratify=Y)
```

In [57]:

```
Y_train.value_counts(normalize=True)*100
```

Out[57]:

```
1    54.304636
0    45.695364
Name: output, dtype: float64
```

In [58]:

```
Y_test.value_counts(normalize=True)*100
```

Out[58]:

```
1    54.304636
0    45.695364
Name: output, dtype: float64
```

In [59]:

```
Y_train.value_counts()
```

Out[59]:

```
1    82
0    69
Name: output, dtype: int64
```

In [60]:

```
Y_test.value_counts()
```

Out[60]:

```
1    82
0    69
Name: output, dtype: int64
```

In [61]:

```
data=Y_train.copy()
```

In [62]:

```
X_train
```

Out[62]:

	age	sex	rest bp	chol	max H.R	oldpeak	cp_1	cp_2	cp_3	fbs_1	...	exng_1	slp_1	slp_2	M.V no._1	M.V no._2	M.V no._3	M.V no._4	thall_1	th
285	-0.932199	1	0.478910	1.248361	-1.293190	0.652812	0	0	0	0	...	1	1	0	0	1	0	0	0	
42	-1.042904	1	-1.574210	-0.745146	-0.068642	1.687716	0	0	0	0	...	1	1	0	0	0	0	0	0	
194	0.617678	1	0.478910	-1.190298	0.237495	1.687716	0	1	0	0	...	0	1	0	0	0	0	0	0	
169	-0.157260	1	0.478910	-0.841918	0.237495	1.773958	0	0	0	1	...	1	0	0	0	0	0	0	0	
183	0.396267	1	-1.117961	-0.319348	0.674834	1.256506	0	1	0	0	...	0	1	0	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
107	-1.042904	0	0.364848	-0.203222	0.106294	-0.727060	0	0	0	0	...	1	1	0	0	0	0	0	0	
275	-0.267966	1	-0.376556	-0.667728	0.806035	-0.037124	0	0	0	0	...	0	0	1	0	1	0	0	0	
103	-1.375021	1	-0.661712	-0.125804	1.943116	-0.209608	0	1	0	1	...	0	0	0	0	0	0	0	0	
65	-2.149959	0	0.364848	-1.229007	1.418309	0.307844	0	0	0	0	...	0	0	1	0	0	0	0	0	
92	-0.267966	1	0.364848	-0.454829	0.849769	-0.899544	0	1	0	0	...	0	0	1	0	0	0	1	0	

151 rows × 22 columns

In [63]:

```
X_test
```

Out[63]:

	age	sex	rest bp	chol	max H.R	oldpeak	cp_1	cp_2	cp_3	fbs_1	...	exng_1	slp_1	slp_2	M.V no._1	M.V no._2	M.V no._3	M.V no._4	thall_1	th
97	-0.267966	1	-1.346085	-0.261285	-0.112376	-0.813302	0	0	0	1	...	0	0	1	0	0	1	0	0	
202	0.396267	1	1.049221	0.454829	-1.686795	-0.209608	0	0	0	0	...	1	0	1	0	0	0	0	0	
233	1.060500	1	-0.661712	-0.009677	-2.342803	0.997780	0	0	0	0	...	1	0	0	1	0	0	0	0	
191	0.396267	1	-0.205463	-0.590310	-0.812118	0.997780	0	0	0	0	...	1	1	0	0	0	1	0	0	
28	1.171205	0	0.478910	3.299932	0.324963	-0.209608	0	1	0	1	...	0	0	1	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
185	-1.153610	1	-1.117961	0.841918	0.150027	-0.899544	0	0	0	0	...	0	0	1	1	0	0	0	0	
54	0.949794	0	0.193755	0.106449	0.980971	-0.899544	0	1	0	0	...	0	0	1	0	0	0	0	0	
151	1.835438	0	-1.117961	-1.887058	-1.074521	0.480328	0	0	0	0	...	0	1	0	0	0	0	0	0	
64	0.396267	1	0.478910	-0.687083	0.674834	-0.899544	0	1	0	1	...	0	0	1	0	0	0	0	0	
154	-1.707137	0	0.364848	-0.512893	0.106294	-0.899544	0	1	0	0	...	0	1	0	0	0	0	0	0	

151 rows × 22 columns

In [64]:

```
Y_train
```

Out[64]:

```
285    0
42     1
194    0
169    0
183    0
      ..
107    1
275    0
103    1
65     1
92     1
Name: output, Length: 151, dtype: int64
```

In [65]:

```
Y_test
```

Out[65]:

```
97     1
202    0
233    0
191    0
28     1
      ..
185    0
54     1
151    1
64     1
154    1
Name: output, Length: 151, dtype: int64
```

Logistic Regression

In [66]:

```
data.head()
```

Out[66]:

```
285    0
42     1
194    0
169    0
183    0
Name: output, dtype: int64
```

In [67]:
```
from sklearn.linear_model import LogisticRegression
```

In [68]:
```
log = LogisticRegression()
log.fit(X_train,Y_train)
```

Out[68]:
```
LogisticRegression()
```

In [69]:
```
Y_pred = log.predict(X_test)
```

In [70]:
```
from sklearn.metrics import accuracy_score
```

In [71]:
```
accuracy_score(Y_test,Y_pred)
```

Out[71]:
```
0.8211920529801324
```

SVC(Support Vector Machine)

In [72]:
```
from sklearn import svm
```

In [73]:
```
svm = svm.SVC()
```

In [74]:
```
svm.fit(X_train,Y_train)
```

Out[74]:
```
SVC()
```

In [75]:
```
Y_pred1= svm.predict(X_test)
```

In [76]:
```
accuracy_score(Y_test,Y_pred1)
```

Out[76]:
```
0.8079470198675497
```

KNN(K Nearest Neighbour)

In [77]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [78]:
```
knn = KNeighborsClassifier()
```

In [79]:
```
knn.fit(X_train,Y_train)
```

Out[79]:
```
KNeighborsClassifier()
```

In [80]:

```
Y_pred2 = knn.predict(X_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction
functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.
11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is
taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warn
ing.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [81]:

```
accuracy_score(Y_test,Y_pred2)
```

Out[81]:

```
0.8145695364238411
```

In [82]:

```
score = []
for k in range(1,40):
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train,Y_train)
    Y_pred = knn.predict(X_test)
    score.append(accuracy_score(Y_test,Y_pred2))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reducti
on functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In Sc
iPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the stat
istic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avo
id this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reducti
on functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In Sc
iPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the stat
istic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avo
id this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reducti
on functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In Sc
iPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the stat
istic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avo
id this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reducti
```

In [83]:

```
score
```

Out[83]:

```
[0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411,
 0.8145695364238411]
```

In [84]:

```
knn = KNeighborsClassifier(n_neighbors = 2)
knn.fit(X_train,Y_train)
Y_pred = knn.predict(X_test)
accuracy_score(Y_test,Y_pred2)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction
functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.
11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is
taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warn
ing.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[84]:

```
0.8145695364238411
```

Non=Linear ML Algorithms

In [85]:

```
data=pd.read_csv("C:/Users/priya/OneDrive/Desktop/heart.csv")
```

In [86]:

```
data.head()
```

Out[86]:

	age	sex	cp	rest bp	chol	fbs	restecg	max H.R	exng	oldpeak	slp	M.V no.	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [87]:

```
data = data.drop_duplicates()
```

In [88]:

```
data.shape
```

Out[88]:

```
(302, 14)
```

In [89]:

```
x = data.drop('output',axis=1)
y = data['output']
```

In [90]:

```
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.2,random_state=42)
```

In [91]:

```
x_train.head()
```

Out[91]:

	age	sex	cp	rest bp	chol	fbs	restecg	max H.R	exng	oldpeak	slp	M.V no.	thall
132	42	1	1	120	295	0	1	162	0	0.0	2	0	2
203	68	1	2	180	274	1	0	150	1	1.6	1	0	3
197	67	1	0	125	254	1	1	163	0	0.2	1	2	3
75	55	0	1	135	250	0	0	161	0	1.4	1	0	2
177	64	1	2	140	335	0	1	158	0	0.0	2	0	2

In [92]:

```
y_train.tail()
```

Out[92]:

```
189    0
71     1
106    1
271    0
102    1
Name: output, dtype: int64
```

Decision Tree Classifier

In [93]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [94]:

```
dt = DecisionTreeClassifier()
```

In [95]:

```
dt.fit(x_train,y_train)
```

Out[95]:

```
DecisionTreeClassifier()
```

In [96]:

```
y_pred3 = dt.predict(x_test)
```

In [97]:

```
accuracy_score(y_test,y_pred3)
```

Out[97]:

```
0.7868852459016393
```

Random Forest Classifier

In [98]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [99]:
```
rf = RandomForestClassifier()
```

In [100]:
```
rf.fit(x_train,y_train)
```

Out[100]:
```
RandomForestClassifier()
```

In [101]:
```
y_pred4 = rf.predict(x_test)
```

In [102]:
```
accuracy_score(y_test,y_pred4)
```

Out[102]:
```
0.8688524590163934
```

Gradient Boosting Classifier

In [103]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [104]:
```
gb = GradientBoostingClassifier()
```

In [105]:
```
gb.fit(x_train,y_train)
```

Out[105]:
```
GradientBoostingClassifier()
```

In [106]:
```
y_pred5 = gb.predict(x_test)
```

In [107]:
```
accuracy_score(y_test,y_pred5)
```

Out[107]:
```
0.8524590163934426
```

Data visualization

In [108]:
```
final_data = pd.DataFrame({'models':['LR','SVM','KNN','DT','RF','GB'],'ACC':[accuracy_score(Y_test,Y_pred),
                                                                            accuracy_score(Y_test,Y_pred1),
                                                                            accuracy_score(Y_test,Y_pred2),
                                                                            accuracy_score(y_test,y_pred3),
                                                                            accuracy_score(y_test,y_pred4),
                                                                            accuracy_score(y_test,y_pred5)]})
```

In [109]:

```
final_data
```

Out[109]:

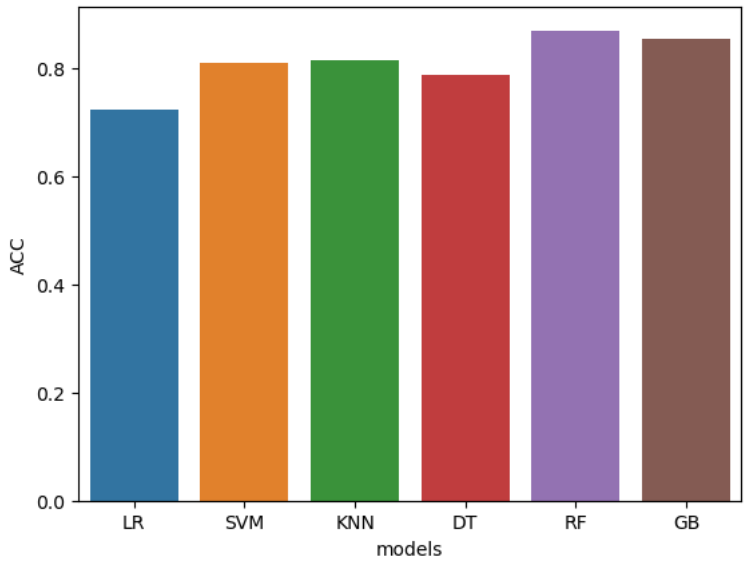
	models	ACC
0	LR	0.721854
1	SVM	0.807947
2	KNN	0.814570
3	DT	0.786885
4	RF	0.868852
5	GB	0.852459

Bar plot

In [110]:

```
sns.barplot(final_data['models'],final_data['ACC']);
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keywor
d args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Here, we can see that the model random forest classifier has is more accurate on dataset.so,we use Random forest classifier on entire dataset in our production of result.

using module random forest on entire data

In [111]:

```
x = data.drop('output',axis=1)
y = data['output']
```

In [112]:

```
x.shape
```

Out[112]:

```
(302, 13)
```

In [113]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [114]:

```
rf = RandomForestClassifier()
```

In [115]:

```
rf.fit(x,y)
```

Out[115]:

```
RandomForestClassifier()
```

Prediction of new data

In [116]:

```
new_data = pd.DataFrame({'age':50,'sex':1,'cp':0,'fbs':0,'exng':0,'oldpeak':1,'thall':3,'max H.R':5,'rest bp':125,
                        'M.V no.':2,'chol':212,'restecg':1,'slp':2},index=[0])
```

In [117]:

```
new_data
```

Out[117]:

	age	sex	cp	fbs	exng	oldpeak	thall	max H.R	rest bp	M.V no.	chol	restecg	slp
0	50	1	0	0	0	1	3	5	125	2	212	1	2

In [118]:

```
p = rf.predict(new_data)
if p[0]==0:
    print("No Heart Disease.....congratulation:)")
else:
    print("yes you have heart disease. Be careful")
```

```
No Heart Disease.....congratulation:)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should match those that we
re passed during fit. Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.

  warnings.warn(message, FutureWarning)
```

sava model using joblib

In [119]:

```
import joblib
```

In [120]:

```
joblib.dump(rf,'model_joblib_heart')
```

Out[120]:

```
['model_joblib_heart']
```

In [121]:

```
model = joblib.load('model_joblib_heart')
```

In [122]:

```
model.predict(new_data)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should match those that we
re passed during fit. Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.

  warnings.warn(message, FutureWarning)
```

Out[122]:

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

THANKS:)