import necessary libraries

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df=pd.read_csv("I:\AIML\FeatureSelection.csv")
```

In [3]:
```
df
```

Out[3]:

	age	weight	height	cholestrol	sugar	Target
0	35	70	150	233	250	1
1	56	75	100	250	300	0
2	67	68	180	204	260	0
3	72	60	170	236	450	1
4	39	77	190	354	220	1
5	48	68	110	192	360	1
6	55	62	200	294	190	0
7	42	59	175	263	350	0
8	30	58	198	199	280	1
9	41	57	100	168	100	1

In [4]:
```
df.shape
```

Out[4]:
```
(10, 6)
```

In [6]:
```
df.describe()
```

Out[6]:

	age	weight	height	cholestrol	sugar	Target
count	10.000000	10.000000	10.000000	10.00000	10.000000	10.000000
mean	48.500000	65.400000	157.300000	239.30000	276.000000	0.600000
std	13.769935	7.214184	39.994583	54.83926	97.661547	0.516398
min	30.000000	57.000000	100.000000	168.00000	100.000000	0.000000
25%	39.500000	59.250000	120.000000	200.25000	227.500000	0.000000
50%	45.000000	65.000000	172.500000	234.50000	270.000000	1.000000
75%	55.750000	69.500000	187.500000	259.75000	337.500000	1.000000
max	72.000000	77.000000	200.000000	354.00000	450.000000	1.000000

feature scaling using min max method

new value=(old value-min)/(max-min)(new_max-new_min)+new_min

new_max =1, new_min=0

new_value=(old_value - min of the column)/(comlum_max - column_min)(1-0)+0

In [7]:
```
df1=df.copy()
```

In [8]:
```
df1
```

Out[8]:

	age	weight	height	cholestrol	sugar	Target
0	35	70	150	233	250	1
1	56	75	100	250	300	0
2	67	68	180	204	260	0
3	72	60	170	236	450	1
4	39	77	190	354	220	1
5	48	68	110	192	360	1
6	55	62	200	294	190	0
7	42	59	175	263	350	0
8	30	58	198	199	280	1
9	41	57	100	168	100	1

In [12]:
```
for column in df1.columns:
    df1[column]=(df1[column]-df1[column].min())/(df1[column].max()-df1[column].min(
```

In [13]:
```
df1.head()
```

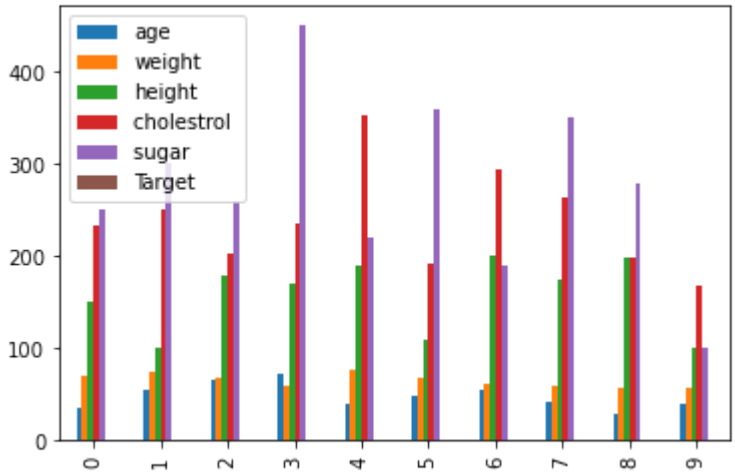
Out[13]:

	age	weight	height	cholestrol	sugar	Target
0	0.119048	0.65	0.5	0.349462	0.428571	1.0
1	0.619048	0.90	0.0	0.440860	0.571429	0.0
2	0.880952	0.55	0.8	0.193548	0.457143	0.0
3	1.000000	0.15	0.7	0.365591	1.000000	1.0
4	0.214286	1.00	0.9	1.000000	0.342857	1.0

In [14]:
```
import matplotlib.pyplot as plt
```

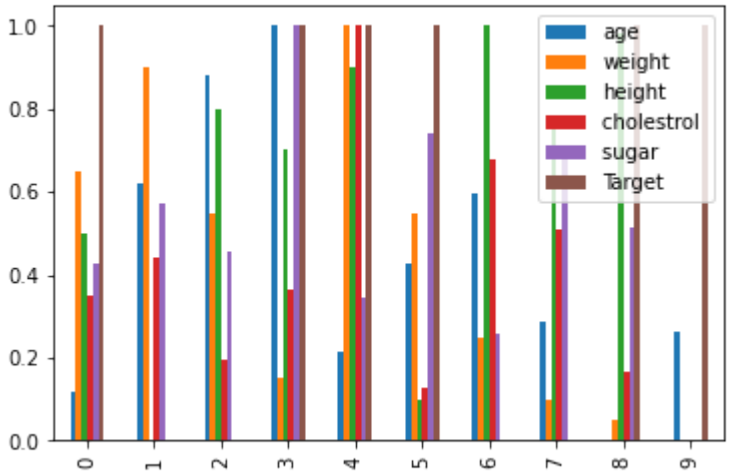
In [15]:
```
df.plot(kind='bar')
```

Out[15]:
```
<AxesSubplot:>
```



In [16]:
```
df1.plot(kind='bar')
```

Out[16]:
```
<AxesSubplot:>
```



z score normalization

new_value=(old_value - mean of the column) / (standard deviation of column)

In [17]:
```
df2 = df.copy()
```

In [18]:
```
for column in df2.columns:
    df2[column]=(df2[column]-df2[column].mean())/df2[column].std()
```

In [19]:
```
df2
```

Out[19]:

	age	weight	height	cholestrol	sugar	Target
0	-0.980397	0.637633	-0.182525	-0.114881	-0.266226	0.774597
1	0.544665	1.330712	-1.432694	0.195116	0.245747	-1.161895
2	1.343507	0.360401	0.567577	-0.643699	-0.163831	-1.161895
3	1.706617	-0.748525	0.317543	-0.060176	1.781663	0.774597
4	-0.689909	1.607944	0.817611	2.091567	-0.573409	0.774597
5	-0.036311	0.360401	-1.182660	-0.862521	0.860113	0.774597
6	0.472043	-0.471294	1.067645	0.997461	-0.880592	-1.161895
7	-0.472043	-0.887141	0.442560	0.432172	0.757719	-1.161895
8	-1.343507	-1.025757	1.017638	-0.734875	0.040958	0.774597
9	-0.544665	-1.164373	-1.432694	-1.300163	-1.802142	0.774597

PCA

In [20]:
```
import pandas as pd
```

In [21]:
```
df=pd.read_csv("I:\AIML\heart.csv")
```

In [22]:
```
df
```

Out[22]:

	age	gender	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	outpu
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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 [24]:
```
df.shape
```

Out[24]:
```
(303, 14)
```

In [25]:
```
inputs = df.drop('output',axis='columns')
```

In [26]:
```
target =df['output']
```

In [27]:
```
inputs.head()
```

Out[27]:

	age	gender	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
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

In [28]:
```
target.head()
```

Out[28]:
```
0    1
1    1
2    1
3    1
4    1
Name: output, dtype: int64
```

In [29]:
```
x=inputs
y=target
```

scale the input data

In [30]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [31]:
```
x_scaled=scaler.fit_transform(x)
```

In [32]:
```
x_scaled
```

Out[32]:
```
array([[ 0.9521966 ,  0.68100522,  1.97312292, ..., -2.27457861,
        -0.71442887, -2.14887271],
       [-1.91531289,  0.68100522,  1.00257707, ..., -2.27457861,
        -0.71442887, -0.51292188],
       [-1.47415758, -1.46841752,  0.03203122, ...,  0.97635214,
        -0.71442887, -0.51292188],
       ...,
       [ 1.50364073,  0.68100522, -0.93851463, ..., -0.64911323,
         1.24459328,  1.12302895],
       [ 0.29046364,  0.68100522, -0.93851463, ..., -0.64911323,
         0.26508221,  1.12302895],
       [ 0.29046364, -1.46841752,  0.03203122, ..., -0.64911323,
         0.26508221, -0.51292188]])
```

In [33]:
```
from sklearn.model_selection import train_test_split
```

In [34]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Apply any classification method without PCA : logistic regression

In [36]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(max_iter=1000)
model.fit(x_train,y_train)
```

Out[36]:
```
LogisticRegression(max_iter=1000)
```

In [37]:
```
model.score(x_test,y_test)
```

Out[37]:
```
0.8688524590163934
```

use PCA to reduce dimensions

In [38]:
```
x
```

Out[38]:

	age	gender	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
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

import libraries for PCA

In [39]:
```
from sklearn.decomposition import PCA
```

use components such that 99% similarities retrived

In [40]:
```
pca=PCA(0.99)
```

In [41]:
```
x_pca=pca.fit_transform(x)
```

In [42]:
```
x.shape
```

Out[42]:
```
(303, 13)
```

In [43]:
```
x_pca.shape
```

Out[43]:
```
(303, 4)
```

In [44]:
```
pca.explained_variance_ratio_
```

Out[44]:
```
array([0.7475642 , 0.15037022, 0.08459685, 0.01621596])
```

In [45]:
```
x_pca
```

Out[45]:
```
array([[ -12.26734484,    2.87383781,   14.96987876,    6.8929401 ],
       [   2.69013712,  -39.87137362,    0.8778823 ,  -10.58359849],
       [ -42.95021407,  -23.63681988,    1.75944589,   -7.7866551 ],
       ...,
       [ -51.96381148,   13.32379836,   15.48684358,   11.6374553 ],
       [-114.75598084,   36.43518423,    0.12777095,    0.51138633],
       [ -10.39614198,  -23.30240081,    2.39130354,    7.18326419]])
```

how many PC created

In [46]:
```
pca.n_components_
```

Out[46]:
```
4
```

In [47]:
```
x_train_pca,x_test_pca,y_train,y_test=train_test_split(x_pca,y,test_size=0.2)
```

In [48]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(max_iter=1000)
model.fit(x_train_pca,y_train)
```

Out[48]:
```
LogisticRegression(max_iter=1000)
```

In [49]:
```
model.score(x_test_pca,y_test)
```

Out[49]:
```
0.5901639344262295
```

let's cosider 7 componets

In [50]:
```
pca=PCA(n_components=7)
```

In [51]:
```
x_pca=pca.fit_transform(x)
```

In [52]:
```
x_pca.shape
```

Out[52]:
```
(303, 7)
```

In [53]:
```
x_train_pca,x_test_pca,y_train,y_test=train_test_split(x_pca,y,test_size=0.2)
```

In [54]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(max_iter=1000)
model.fit(x_train_pca,y_train)
```

Out[54]:
```
LogisticRegression(max_iter=1000)
```

In [55]:
```
model.score(x_test_pca,y_test)
```

Out[55]:
```
0.7049180327868853
```

In [ ]:
```

```