import modules

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import sklearn as sl
from sklearn.metrics import confusion_matrix
```

importing dataset

In [3]:

```
data = pd.read_csv("C:/Users/Shilpi Rani/Downloads/heart.csv")
```

Description about dataset

In [4]:

```
data.shape
```

Out[4]:

```
(303, 14)
```

In [5]:

```
data.size
```

Out[5]:

```
4242
```

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.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
```

In [9]:

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

Out[9]:

	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 [10]:

```
data.columns
```

Out[10]:

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

checking missing values in dataset

In [11]:

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

Out[11]:

```
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 [12]:

```
data.isnull()
```

Out[12]:

	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

Checking the duplicate values in dataset

In [13]:

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

In [14]:

```
data_dup
```

Out[14]:

```
True
```

In [15]:

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

In [16]:

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

In [17]:

```
data_dup
```

Out[17]:

```
False
```

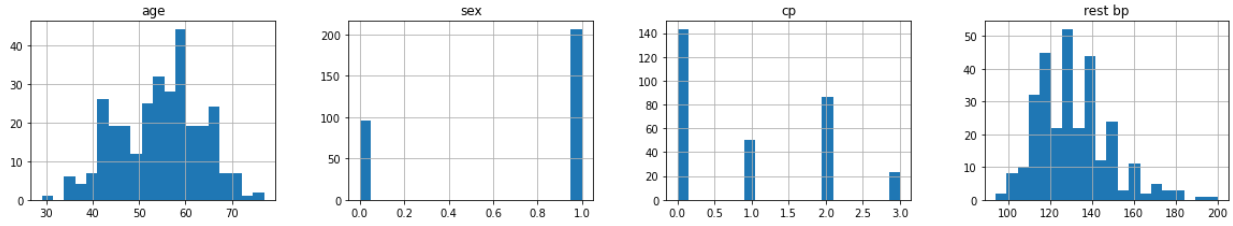
Exploratory data analysis

Histogram graph

In [18]:

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

```
       <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)
```



Correlation matrix

In [19]:

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

In [20]:

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

Out[20]:

```
age         1.000000
M.V no.     0.302261
rest bp     0.283121
chol        0.207216
oldpeak     0.206040
fbs         0.119492
exng        0.093216
thall       0.065317
cp         -0.063107
sex        -0.094962
restecg    -0.111590
slp        -0.164124
output     -0.221476
max H.R    -0.395235
Name: age, dtype: float64
```

In [21]:

```
data.corr()
```

Out[21]:

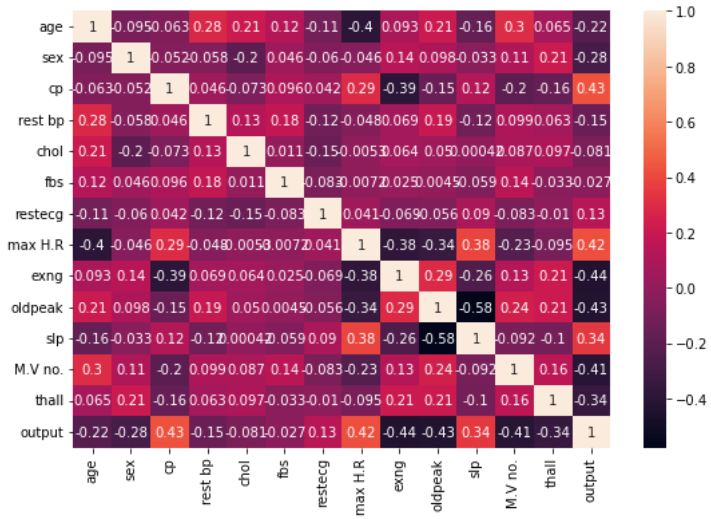
	age	sex	cp	rest bp	chol	fbs	restecg	max H.R	exng	oldpeak	slp	M.V no.	thall	out
age	1.000000	-0.094962	-0.063107	0.283121	0.207216	0.119492	-0.111590	-0.395235	0.093216	0.206040	-0.164124	0.302261	0.065317	-0.221
sex	-0.094962	1.000000	-0.051740	-0.057647	-0.195571	0.046022	-0.060351	-0.046439	0.143460	0.098322	-0.032990	0.113060	0.211452	-0.283
cp	-0.063107	-0.051740	1.000000	0.046486	-0.072682	0.096018	0.041561	0.293367	-0.392937	-0.146692	0.116854	-0.195356	-0.160370	0.432
rest bp	0.283121	-0.057647	0.046486	1.000000	0.125256	0.178125	-0.115367	-0.048023	0.068526	0.194600	-0.122873	0.099248	0.062870	-0.146
chol	0.207216	-0.195571	-0.072682	0.125256	1.000000	0.011428	-0.147602	-0.005308	0.064099	0.050086	0.000417	0.086878	0.096810	-0.081
fbs	0.119492	0.046022	0.096018	0.178125	0.011428	1.000000	-0.083081	-0.007169	0.024729	0.004514	-0.058654	0.144935	-0.032752	-0.026
restecg	-0.111590	-0.060351	0.041561	-0.115367	-0.147602	-0.083081	1.000000	0.041210	-0.068807	-0.056251	0.090402	-0.083112	-0.010473	0.134
max H.R	-0.395235	-0.046439	0.293367	-0.048023	-0.005308	-0.007169	0.041210	1.000000	-0.377411	-0.342201	0.384754	-0.228311	-0.094910	0.419
exng	0.093216	0.143460	-0.392937	0.068526	0.064099	0.024729	-0.068807	-0.377411	1.000000	0.286766	-0.256106	0.125377	0.205826	-0.435
oldpeak	0.206040	0.098322	-0.146692	0.194600	0.050086	0.004514	-0.056251	-0.342201	0.286766	1.000000	-0.576314	0.236560	0.209090	-0.429
slp	-0.164124	-0.032990	0.116854	-0.122873	0.000417	-0.058654	0.090402	0.384754	-0.256106	-0.576314	1.000000	-0.092236	-0.103314	0.343
M.V no.	0.302261	0.113060	-0.195356	0.099248	0.086878	0.144935	-0.083112	-0.228311	0.125377	0.236560	-0.092236	1.000000	0.160085	-0.408
thall	0.065317	0.211452	-0.160370	0.062870	0.096810	-0.032752	-0.010473	-0.094910	0.205826	0.209090	-0.103314	0.160085	1.000000	-0.343
output	-0.221476	-0.283609	0.432080	-0.146269	-0.081437	-0.026826	0.134874	0.419955	-0.435601	-0.429146	0.343940	-0.408992	-0.343101	1.000

In [22]:

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

Out[22]:

```
<AxesSubplot:>
```



Data processing

In [23]:

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

In [24]:

```
cate_val
```

Out[24]:

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

In [25]:

```
cont_val
```

Out[25]:

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

Encoding categorical data

In [26]:

```
cate_val
```

Out[26]:

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

In [27]:

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

Out[27]:

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

In [28]:

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

In [29]:

```
data.head()
```

Out[29]:

	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 [30]:

```
data.head()
```

Out[30]:

	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 [31]:

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

In [32]:

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

In [33]:

```
data.head()
```

Out[33]:

	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 dataset

In [34]:

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

In [35]:

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

In [36]:

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

In [45]:

```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [46]:

```
X_train
```

Out[46]:

	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 [47]:

```
X_test
```

Out[47]:

	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
180	0.064151	1	0.022661	2.061248	-0.768384	0.135360	0	0	0	0	...	1	1	0	1	0	0	0	0	
229	1.060500	1	-0.376556	1.209652	-0.812118	0.652812	0	1	0	0	...	1	1	0	0	0	0	0	0	
111	0.285561	1	1.049221	-2.332210	1.024705	-0.727060	0	1	0	1	...	0	0	1	1	0	0	0	0	
247	1.281911	1	1.619532	-0.009677	-1.293190	-0.899544	1	0	0	0	...	1	1	0	0	0	1	0	1	
60	1.835438	0	-1.232023	0.358057	-0.855851	-0.899544	0	1	0	1	...	0	0	1	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
250	-0.378671	1	0.478910	0.996754	-1.205722	2.722620	0	0	0	0	...	1	1	0	0	0	1	0	0	
104	-0.489377	1	-0.148432	-0.977399	0.587366	-0.899544	0	1	0	0	...	0	0	1	0	0	0	0	0	
300	1.503322	1	0.707035	-1.035462	-0.374779	2.032684	0	0	0	1	...	0	1	0	0	1	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	
185	-1.153610	1	-1.117961	0.841918	0.150027	-0.899544	0	0	0	0	...	0	0	1	1	0	0	0	0	

61 rows × 22 columns

In [40]:

```
y_train
```

Out[40]:

```
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 [41]:

```
y_test
```

Out[41]:

```
180    0
229    0
111    1
247    0
60     1
      ..
250    0
104    1
300    0
194    0
185    0
Name: output, Length: 61, dtype: int64
```

Logistic regression

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

In [43]:

```
from sklearn.linear_model import LogisticRegression
```

In [48]:

```
log = LogisticRegression()
log.fit(X_train,y_train)
```

Out[48]:

```
LogisticRegression()
```

In [49]:

```
y_pred1 = log.predict(X_test)
```

In [50]:

```
from sklearn.metrics import accuracy_score
```

In [51]:

```
accuracy_score(y_test,y_pred1)
```

Out[51]:

```
0.9016393442622951
```

SVC

In [52]:

```
from sklearn import svm
```

In [53]:

```
svm = svm.SVC()
```

In [54]:

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

Out[54]:

```
SVC()
```

In [55]:

```
y_pred2 = svm.predict(x_test)
```

In [56]:

```
accuracy_score(y_test,y_pred2)
```

Out[56]:

```
0.8688524590163934
```

Non-linear ML Algorithms

In [57]:

```
data = pd.read_csv("C:/Users/Shilpi Rani/Downloads/heart.csv")
```

In [58]:

```
data.head()
```

Out[58]:

	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 [59]:

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

In [60]:

```
data.shape
```

Out[60]:

```
(302, 14)
```

KNN

In [61]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [62]:

```
knn = KNeighborsClassifier()
```

In [63]:

```
knn.fit(X_train, y_train)
```

Out[63]:

```
KNeighborsClassifier()
```

In [64]:

```
y_pred3 = knn.predict(X_test)
```

In [65]:

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

Out[65]:

```
0.8688524590163934
```

In [66]:

```
score = []
```

In [67]:

```
knn = KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
(accuracy_score(y_test, y_pred))
```

Out[67]:

```
0.8688524590163934
```

Decision Tree Classifier

In [68]:

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

In [69]:

```
dt=DecisionTreeClassifier()
```

In [70]:

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

Out[70]:

```
DecisionTreeClassifier()
```

In [71]:

```
y_pred4 = dt.predict(X_test)
```

In [72]:

```
accuracy_score(y_test, y_pred4)
```

Out[72]:

```
0.7704918032786885
```

Comparison

In [73]:

```
final_data = pd.DataFrame({'Models':['LR','SVM','KNN','DT'], 'ACC':[accuracy_score(y_test, y_pred1),
                                                                    accuracy_score(y_test, y_pred2),
                                                                    accuracy_score(y_test, y_pred3),
                                                                    accuracy_score(y_test, y_pred4)]})
```

In [74]:

```
final_data
```

Out[74]:

	Models	ACC
0	LR	0.901639
1	SVM	0.868852
2	KNN	0.868852
3	DT	0.770492

In [75]:

```
import seaborn as sns
```

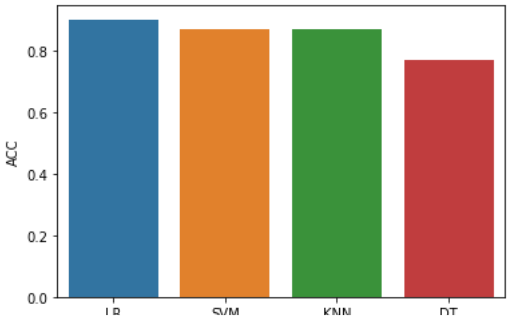
In [76]:

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

```
  warnings.warn(
```

Out[76]:

```
<AxesSubplot:xlabel='Models', ylabel='ACC'>
```



In [77]:

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

In [78]:

```
X.shape
```

Out[78]:

```
(302, 13)
```

In [83]:

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

In [85]:

```
dt=DecisionTreeClassifier()
dt.fit(X,y)
```

Out[85]:

```
DecisionTreeClassifier()
```

Prediction on New Data

In [79]:

```
import pandas as pd
```

In [80]:

```
new_data = pd.DataFrame({
  'age': 52,
  'sex': 1,
  'cp': 0,
  'trestbps': 125,
  'chol': 212,
  'fbs': 0,
  'restecg': 1,
  'thalach': 168,
  'exang': 0,
  'oldpeak': 1.0,
  'slope': 2,
  'ca': 2,
  'thal': 3,
}, index=[0])
```

In [81]:

```
new_data
```

Out[81]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3

In [86]:

```
p= dt.predict(new_data)
if p[0]==0:
    print("No Disease")
else:
    print("Yes, you have heart disease")
```

```
No Disease

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 unseen at fit time:
- ca
- exang
- slope
- thal
- thalach
- ...
Feature names seen at fit time, yet now missing:
- M.V no.
- exng
- max H.R
- rest bp
- slp
- ...

  warnings.warn(message, FutureWarning)
```

Save Model using Joblib

In [87]:

```
import joblib
```

In [88]:

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

Out[88]:

```
['model_joblib_heart']
```

In [89]:

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

In [90]:

```
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 unseen at fit time:
- ca
- exang
- slope
- thal
- thalach
- ...
Feature names seen at fit time, yet now missing:
- M.V no.
- exng
- max H.R
- rest bp
- slp
- ...

  warnings.warn(message, FutureWarning)
```

Out[90]:

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

THANKS!!