Importing The Libraries

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

Importing The Datasets

In [2]:
```
data=pd.read_csv('heart1.csv')
```

Taking Care Of Missing Values

In [3]:
```
data.isnull().sum()
```

Out[3]:
```
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
Unnamed: 14    1025
dtype: int64
```

Taking Care Of Duplicate Values

In [4]:
```
data_dup = data.duplicated().any()
```

In [5]:
```
data_dup
```

Out[5]: True

In [6]:
```
data = data.drop_duplicates()
```

In [7]:
```
data_dup = data.duplicated().any()
```

In [8]:
```
data_dup
```

Out[8]: False

Data processing

In [9]:
```
cate_val=[]
cont_val=[]

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

In [10]:
```
cate_val
```

Out[10]:
```
['sex',
 'cp',
 'fbs',
 'restecg',
 'exang',
 'slope',
 'ca',
 'thal',
 'target',
 'Unnamed: 14']
```

In [11]:
```
cont_val
```

Out[11]:
```
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

Encoding Categorical Data

In [12]:
```
cate_val
```

Out[12]:
```
['sex',
 'cp',
 'fbs',
 'restecg',
 'exang',
 'slope',
 'ca',
 'thal',
 'target',
 'Unnamed: 14']
```

In [13]:
```
data['cp'].unique()
```

Out[13]:
```
array([0, 1, 2, 3], dtype=int64)
```

In [14]:
```
cate_val.remove('sex')
```

In [15]:
```
cate_val.remove('target')
```

In [16]:
```
data=pd.get_dummies(data,columns=cate_val,drop_first=True)
```

In [17]:
```
data.head()
```

Out[17]:

	age	sex	trestbps	chol	thalach	oldpeak	target	cp_1	cp_2	cp_3	...	exang_1	slope_1	slope_2	ca_1	ca_2	ca_3	ca_4	thal_1	thal_2	thal_3
0	52	1	125	212	168	1.0	0	0	0	0	...	0	0	1	0	1	0	0	0	0	1
1	53	1	140	203	155	3.1	0	0	0	0	...	1	0	0	0	0	0	0	0	0	1
2	70	1	145	174	125	2.6	0	0	0	0	...	1	0	0	0	0	0	0	0	0	1
3	61	1	148	203	161	0.0	0	0	0	0	...	0	0	1	1	0	0	0	0	0	1
4	62	0	138	294	106	1.9	0	0	0	0	...	0	1	0	0	0	1	0	0	1	0

5 rows × 23 columns

Feature Scaling

In [18]:
```
data.head()
```

Out[18]:

	age	sex	trestbps	chol	thalach	oldpeak	target	cp_1	cp_2	cp_3	...	exang_1	slope_1	slope_2	ca_1	ca_2	ca_3	ca_4	thal_1	thal_2	thal_3
0	52	1	125	212	168	1.0	0	0	0	0	...	0	0	1	0	1	0	0	0	0	1
1	53	1	140	203	155	3.1	0	0	0	0	...	1	0	0	0	0	0	0	0	0	1
2	70	1	145	174	125	2.6	0	0	0	0	...	1	0	0	0	0	0	0	0	0	1
3	61	1	148	203	161	0.0	0	0	0	0	...	0	0	1	1	0	0	0	0	0	1
4	62	0	138	294	106	1.9	0	0	0	0	...	0	1	0	0	0	1	0	0	1	0

5 rows × 23 columns

In [19]:
```
from sklearn.preprocessing import StandardScaler
```

In [20]:
```
st=StandardScaler()
data[cont_val]=st.fit_transform(data[cont_val])
```

In [21]:
```
data.head()
```

Out[21]:

	age	sex	trestbps	chol	thalach	oldpeak	target	cp_1	cp_2	cp_3	...	exang_1	slope_1	slope_2	ca_1	ca_2	ca_3	ca_4	thal_1	thal_2	thal_3
0	-0.267966	1	-0.376556	-0.667728	0.806035	-0.037124	0	0	0	0	...	0	0	1	0	1	0	0	0	0	1
1	-0.157260	1	0.478910	-0.841918	0.237495	1.773958	0	0	0	0	...	1	0	0	0	0	0	0	0	0	1
2	1.724733	1	0.764066	-1.403197	-1.074521	1.342748	0	0	0	0	...	1	0	0	0	0	0	0	0	0	1
3	0.728383	1	0.935159	-0.841918	0.499898	-0.899544	0	0	0	0	...	0	0	1	1	0	0	0	0	0	1
4	0.839089	0	0.364848	0.919336	-1.905464	0.739054	0	0	0	0	...	0	1	0	0	0	1	0	0	1	0

5 rows × 23 columns

Splitting The Dataset Into The Training Set And Test Set

In [22]:
```
x = data.drop('target',axis=1)
```

In [23]:
```
y =data['target']
```

In [24]:
```
from sklearn.model_selection import train_test_split
```

In [25]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

Logistic Regression

In [26]:
```
data.head()
```

Out[26]:

	age	sex	trestbps	chol	thalach	oldpeak	target	cp_1	cp_2	cp_3	...	exang_1	slope_1	slope_2	ca_1	ca_2	ca_3	ca_4	thal_1	thal_2	thal_3
0	-0.267966	1	-0.376556	-0.667728	0.806035	-0.037124	0	0	0	0	...	0	0	1	0	1	0	0	0	0	1
1	-0.157260	1	0.478910	-0.841918	0.237495	1.773958	0	0	0	0	...	1	0	0	0	0	0	0	0	0	1
2	1.724733	1	0.764066	-1.403197	-1.074521	1.342748	0	0	0	0	...	1	0	0	0	0	0	0	0	0	1
3	0.728383	1	0.935159	-0.841918	0.499898	-0.899544	0	0	0	0	...	0	0	1	1	0	0	0	0	0	1
4	0.839089	0	0.364848	0.919336	-1.905464	0.739054	0	0	0	0	...	0	1	0	0	0	1	0	0	1	0

5 rows × 23 columns

In [27]:
```
from sklearn.linear_model import LogisticRegression
```

In [28]:
```
log=LogisticRegression()
log.fit(x_train,y_train)
```

Out[28]:
```
▾ LogisticRegression
LogisticRegression()
```

In [29]:
```
y_pred1=log.predict(x_test)
```

In [30]:
```
from sklearn.metrics import accuracy_score
```

In [31]:
```
accuracy_score(y_test, y_pred1)
```

Out[31]: 0.7868852459016393

SVC(Support Vector classifier)

In [32]:
```
from sklearn import svm
```

In [33]:
```
svm=svm.SVC()
```

In [34]:
```
svm.fit(x_train,y_train)
```

Out[34]:
```
▾ SVC
SVC()
```

In [35]:
```
y_pred2=svm.predict(x_test)
```

In [36]:
```
accuracy_score(y_test,y_pred2)
```

Out[36]: 0.8032786885245902

KNeighbours Classifiers

In [37]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [38]:
```
knn = KNeighborsClassifier()
```

In [39]:
```
knn.fit(x_train,y_train)
```

Out[39]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [40]:
```
y_pred3=knn.predict(x_test)
```

In [41]:
```
accuracy_score(y_test,y_pred3)
```

Out[41]: 0.7377049180327869

In [42]:
```
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_pred))
```

In [43]:
```
score
```

Out[43]:
```
[0.7213114754098361,
 0.8032786885245902,
 0.7049180327868853,
 0.7049180327868853,
 0.7377049180327869,
 0.8032786885245902,
 0.7868852459016393,
 0.8032786885245902,
 0.7704918032786885,
 0.7540983606557377,
 0.7704918032786885,
 0.7540983606557377,
 0.7377049180327869,
 0.7377049180327869,
 0.7540983606557377,
 0.7704918032786885,
 0.7540983606557377,
 0.7540983606557377,
 0.7377049180327869,
 0.7540983606557377,
 0.7377049180327869,
 0.7213114754098361,
 0.7377049180327869,
 0.7377049180327869,
 0.7213114754098361,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869,
 0.7377049180327869]
```

In [44]:
```
knn=KNeighborsClassifier(n_neighbors=2)
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
accuracy_score(y_test,y_pred)
```

Out[44]: 0.8032786885245902

Non-Linear ML Algorithms

In [45]:
```
data=pd.read_csv('heart1.csv')
```

In [46]:
```
data.head()
```

Out[46]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	Unnamed: 14
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0	NaN
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0	NaN
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0	NaN
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0	NaN
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0	NaN

In [47]:
```
data = data.drop_duplicates()
del data[data. columns[-1]]
```

In [48]:
```
data.shape
```

Out[48]: (302, 14)

In [49]:
```
x=data.drop('target',axis=1)
y=data['target']
```

In [50]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

Decision Tree Classifier

In [51]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [52]:
```
dt=DecisionTreeClassifier()
```

In [53]:
```
dt.fit(x_train,y_train)
```

Out[53]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [54]:
```
y_pred4=dt.predict(x_test)
```

In [55]:
```
accuracy_score(y_test,y_pred4)
```

Out[55]: 0.7704918032786885

Random Forest Classifier

In [56]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [57]:
```
rf = RandomForestClassifier()
```

In [58]:
```
rf.fit(x_train,y_train)
```

Out[58]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [59]:
```
y_pred5=rf.predict(x_test)
```

In [60]:
```
accuracy_score(y_test,y_pred5)
```

Out[60]: 0.8524590163934426

Gradient Boosting Classifier

In [61]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [62]:
```
gbc = GradientBoostingClassifier()
```

In [63]:
```
gbc.fit(x_train,y_train)
```

Out[63]:
```
▾ GradientBoostingClassifier
GradientBoostingClassifier()
```

In [64]:
```
y_pred6=gbc.predict(x_test)
```

In [65]:
```
accuracy_score(y_test,y_pred6)
```

Out[65]: 0.8032786885245902

In [66]:
```
final_data = pd.DataFrame({'Models':['LR','SVM','KNN','DT','RF','GB'],
                           '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),
                                  accuracy_score(y_test,y_pred5),
                                  accuracy_score(y_test,y_pred6)]})
```

In [67]:
```
final_data
```

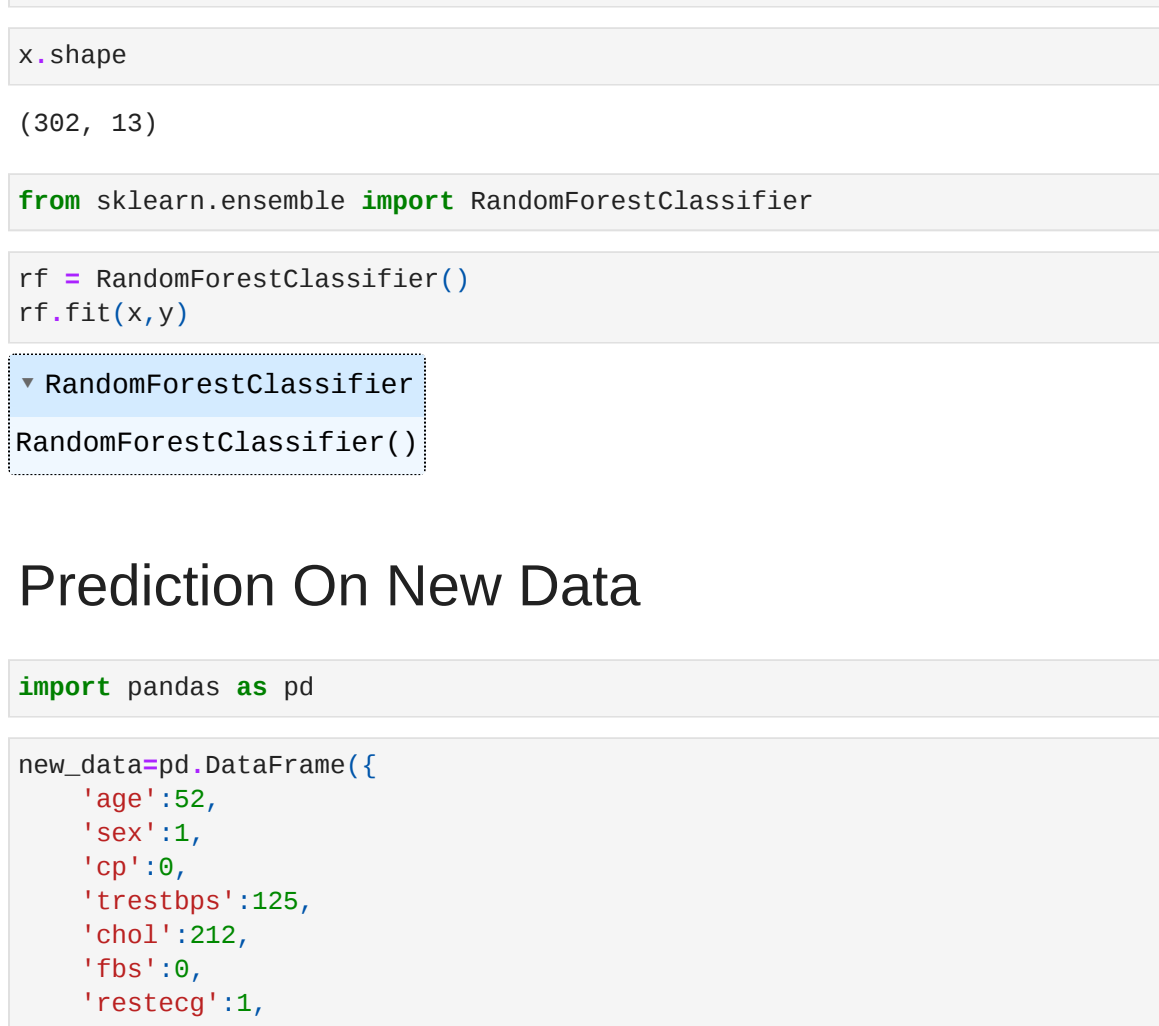
Out[67]:

	Models	ACC
0	LR	0.786885
1	SVM	0.803279
2	KNN	0.737705
3	DT	0.770492
4	RF	0.852459
5	GB	0.803279

In [68]:
```
import seaborn as sns
```

In [69]:
```
sns.barplot(x=final_data['Models'],y=final_data['ACC'])
```

Out[69]: <AxesSubplot: xlabel='Models', ylabel='ACC'>



In [70]:
```
x=data.drop('target',axis=1)
```

In [71]:
```
y=data['target']
```

In [72]:
```
x.shape
```

Out[72]: (302, 13)

In [73]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [74]:
```
rf = RandomForestClassifier()
rf.fit(x,y)
```

Out[74]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

Prediction On New Data

In [75]:
```
import pandas as pd
```

In [76]:
```
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 [77]:
```
new_data
```

Out[77]:

	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 [78]:
```
p = rf.predict(new_data)
if p[0]==0:
    print("Patient has no Disease")
else:
    print("Patient has a heart Disease")
```

```
Patient has no Disease
```

Save Model Using Joblib

In [79]:
```
import joblib
```

In [80]:
```
joblib.dump(rf,'model_joblib_heart')
```

Out[80]:
```
['model_joblib_heart']
```

In [81]:
```
model=joblib.load('model_joblib_heart')
```

In [82]:
```
model.predict(new_data)
```

Out[82]:
```
array([0], dtype=int64)
```

Creating GUI

In [83]:
```
from tkinter import *
import joblib
```

In [84]:
```
from tkinter import *
import joblib
import numpy as np
from sklearn import *
def show_entry_fields():
    p1=int(e1.get())
    p2=int(e2.get())
    p3=int(e3.get())
    p4=int(e4.get())
    p5=int(e5.get())
    p6=int(e6.get())
    p7=int(e7.get())
    p8=int(e8.get())
    p9=int(e9.get())
    p10=float(e10.get())
    p11=int(e11.get())
    p12=int(e12.get())
    p13=int(e13.get())
    model = joblib.load('model_joblib_heart')
    result=model.predict([[p1,p2,p3,p4,p5,p6,p7,p8,p8,p10,p11,p12,p13]])

    if result == 0:
        Label(master, text="NO HEART DISEASE").grid(row=31)
    else:
        Label(master, text="POSSIBILITY OF HEART DISEASE").grid(row=31)


master = Tk()
master.title("Heart Disease Prediction System")


label = Label(master, text = "Heart Disease Prediction System"
                          , bg = "white", fg = "blue"). \
                               grid(row=0,columnspan=2)


Label(master, text="Enter Your Age").grid(row=1)
Label(master, text="Male Or Female [1/0]").grid(row=2)
Label(master, text="Enter Value of CP").grid(row=3)
Label(master, text="Enter Value of trestbps").grid(row=4)
Label(master, text="Enter Value of chol").grid(row=5)
Label(master, text="Enter Value of fbs").grid(row=6)
Label(master, text="Enter Value of restecg").grid(row=7)
Label(master, text="Enter Value of thalach").grid(row=8)
Label(master, text="Enter Value of exang").grid(row=9)
Label(master, text="Enter Value of oldpeak").grid(row=10)
Label(master, text="Enter Value of slope").grid(row=11)
Label(master, text="Enter Value of ca").grid(row=12)
Label(master, text="Enter Value of thal").grid(row=13)



e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)
e8 = Entry(master)
e9 = Entry(master)
e10 = Entry(master)
e11 = Entry(master)
e12 = Entry(master)
e13 = Entry(master)

e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)
e6.grid(row=6, column=1)
e7.grid(row=7, column=1)
e8.grid(row=8, column=1)
e9.grid(row=9, column=1)
e10.grid(row=10, column=1)
e11.grid(row=11, column=1)
e12.grid(row=12, column=1)
e13.grid(row=13, column=1)



Button(master, text='Predict',  bg = "white", fg = "green",command=show_entry_fields).grid()

mainloop()
```

```
C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but Ra
ndomForestClassifier was fitted with feature names
  warnings.warn(
```

In [ ]:

In [ ]: