In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [ ]:

```
#Data collection and processing
#This model will predict whether the person has Heart disease or Not.
#Step 1:Creating the dataframe from dataset.
data = pd.read_csv('/content/heart_disease_data.csv')
data
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
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 [ ]:

```
data.head()
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
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 [ ]:

```
data.tail()
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
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	

In [ ]:

```
data.shape
```

Out[ ]:

```
(303, 14)
```

In [ ]:

```
#Step 2:Preprocessing of Dataframes.
#getting info
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   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [ ]:

```
#checking for missing values
data.isnull().sum()
```

Out[ ]:

```
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
```

In [ ]:

```
#statstical measure
data.describe()
```

Out[ ]:

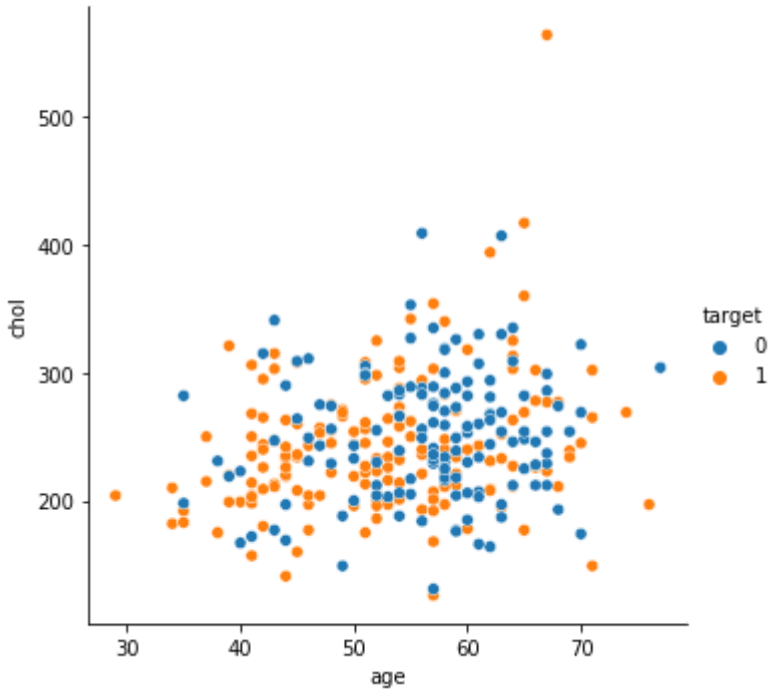
age	sex	cp	trestbps	chol	fbs	restecg	thalach	
000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.
337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.
101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.
000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.
000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.
000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.
000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.
000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.

In [ ]:

```
#step 3:Data Visualization
sns.relplot(x='age',y='chol', hue = 'target', data=data) #Relating age with chol column
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7fee598c07d0>
```

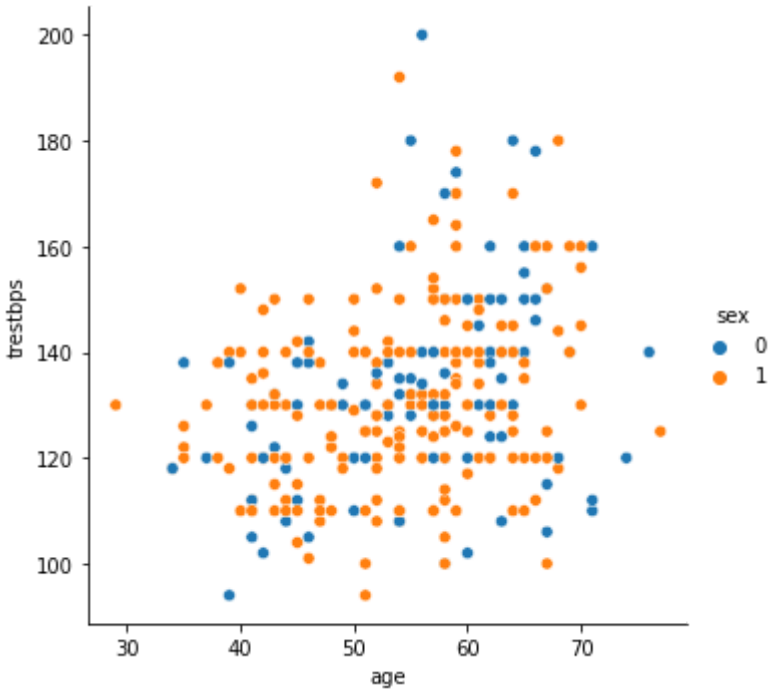


In [ ]:

```
sns.relplot(x='age',y='trestbps',hue = 'sex',data=data)
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7fee4d9e2a90>
```



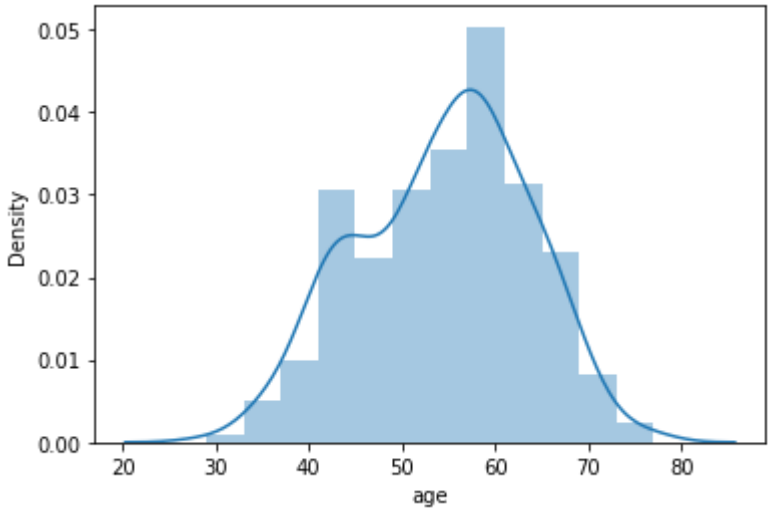
In [ ]:

```
sns.distplot(data['age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Futu
reWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee518fb3d0>
```

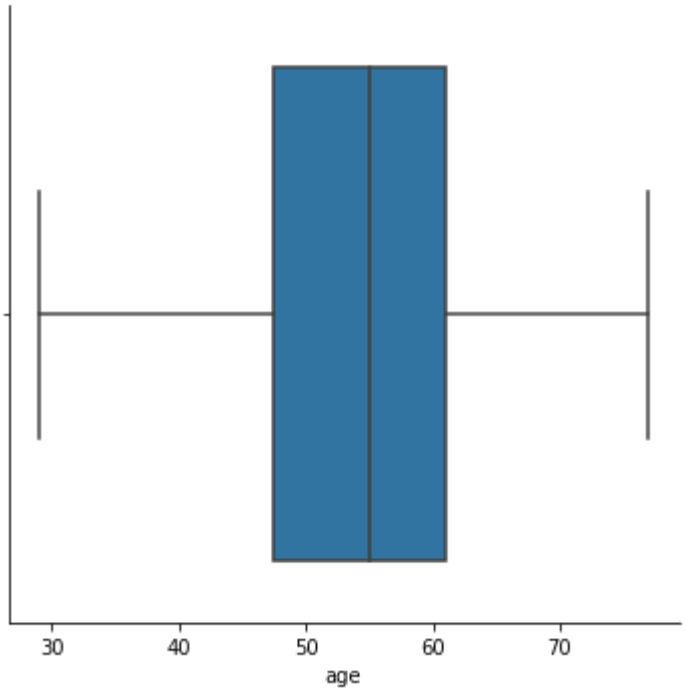


In [ ]:

```
sns.catplot(x='age',kind='box',data=data) #The persons belongs in the range of this bo
x.
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7fee4db77290>
```

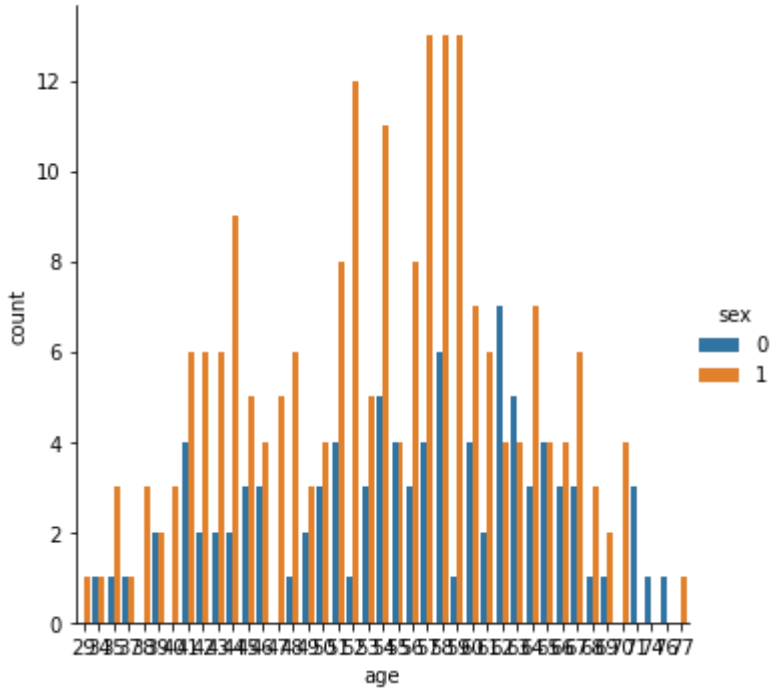


In [ ]:

```
sns.catplot(x='age',hue = 'sex', kind = 'count',data= data)
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7fee513efa90>
```



In [ ]:

```
#checking the distribution of target data
data['target'].value_counts()
```

Out[ ]:

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

In [ ]:

```
#Step 4:Dividing the input and Output
#1 - defective heart
#0 - Healthy heart
#splitting the features and Target
x = data.drop(columns = 'target',axis = -1)
y = data['target']
```

In [ ]:

```
print(x)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
\
0     63    1   3       145   233    1        0      150      0      2.3
1     37    1   2       130   250    0        1      187      0      3.5
2     41    0   1       130   204    0        0      172      0      1.4
3     56    1   1       120   236    0        1      178      0      0.8
4     57    0   0       120   354    0        1      163      1      0.6
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298   57    0   0       140   241    0        1      123      1      0.2
299   45    1   3       110   264    0        1      132      0      1.2
300   68    1   0       144   193    1        1      141      0      3.4
301   57    1   0       130   131    0        1      115      1      1.2
302   57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     ...  ..   ...
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[303 rows x 13 columns]
```

In [ ]:

```
print(y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In [ ]:

```
#Step 5: Train and Test variables.
#splitting the data into training data into testing data
x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.2, stratify = y,rand
om_state = 2)
```

In [ ]:

```
print(x.shape,x_train.shape,x_test.shape)
```

```
(303, 13) (242, 13) (61, 13)
```

In [ ]:

```
print(y.shape,y_train,y_test.shape)
```

```
(303,) 61     1
238    0
160    1
158    1
289    0
      ..
100    1
49     1
300    0
194    0
131    1
Name: target, Length: 242, dtype: int64 (61,)
```

In [ ]:

```
#Step 7 : Running the Regressor.
#model training
model = LogisticRegression()
```

In [ ]:

```
#Step 8:Fitting the model(Mapping the inputs with outputs)
#training the Logisticmodel with training data
model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:8
18: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

Out[ ]:

```
LogisticRegression()
```

In [ ]:

```
#model evaluation
#accuracyScore
#accuract on training data
```

In [ ]:

```
#Step 9 : Predicting the output.
#Building a predictive System
input_data = (41,0,1,130,204,0,0,172,0,1.4,2,0,2)

#change the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

#reshape the numpy array as we are predicting for only one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if(prediction[0] == 0):
  print("The Person has a Heart Disease")
else:
  print("The Person has Heart Disease")
```

```
[1]
The Person has Heart Disease

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  "X does not have valid feature names, but"
```

In [ ]:

```
#Step 10 : Evaluation :Accuracy score.
#accuracy on training data
X_train_prediction = model.predict(x_train)
training_data_accuracy = accuracy_score(X_train_prediction,y_train)
```

In [ ]:

```
print('Accuracy on Training data : ',training_data_accuracy)
```

```
Accuracy on Training data :  0.8512396694214877
```

In [ ]:

```
#accuracy on training data
X_test_prediction = model.predict(x_test)
test_data_accuracy = accuracy_score(X_test_prediction,y_test)
```

In [ ]:

```
print('Accuracy on Test data : ',test_data_accuracy)
```

```
Accuracy on Test data :  0.819672131147541
```