```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
heart_data = pd.read_csv('/content/heart.csv')
```

```
heart_data.head()
```

	age	sex	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

```
heart_data.tail()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thal
298	57	0	0	140	241	0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	

```
heart_data.shape
```

```
(303, 14)
```

```
heart_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   trtbps    303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       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
```

```
heart_data.isnull().sum()
```

```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
```

```
caa         0
thall       0
output      0
dtype: int64
```

```
heart_data.describe()
```

	age	sex	cp	trtbps	chol	fbs	restec
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.52805
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.52586
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.00000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.00000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.00000

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

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

1 --> Defective heart 0 --> Healthy heart

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

```
print(x)
```

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

     caa  thall
0      0      1
1      0      2
2      0      2
3      0      2
4      0      2
..   ...    ...
298    0      3
299    0      3
300    2      3
301    1      3
302    1      2

[303 rows x 13 columns]
```

```
print(y)
```

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

```
from IPython.testing import test
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,stratify=y,random_state=2)
```

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

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

Logistic Regression

```
model = LogisticRegression()
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Conver
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    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-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
x_train_prediction = model.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction,y_train)
```

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

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

```
x_test_prediction = model.predict(x_test)
test_data_accuracy = accuracy_score(x_test_prediction,y_test)
```

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

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

```
input_data=(41,0,1,130,204,0,172,0,1,4,2,0,2)

input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]==0):
  print('Person doesnot have heart disease')
else:
  print('Person has a heart disease')
```

```
[1]
Person has a heart disease
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticRegressi
  warnings.warn(
```

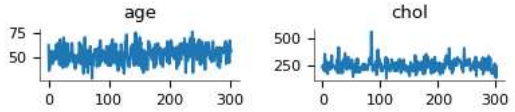
Kmeans clustering

```
X=heart_data[["age","chol"]]
X
```

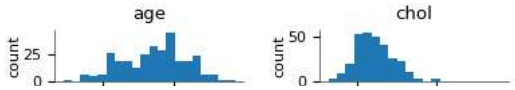
	age	chol
0	63	233
1	37	250
2	41	204
3	56	236
4	57	354
...	...	...
298	57	241
299	45	264
300	68	193
301	57	131
302	57	236

303 rows × 2 columns

Values

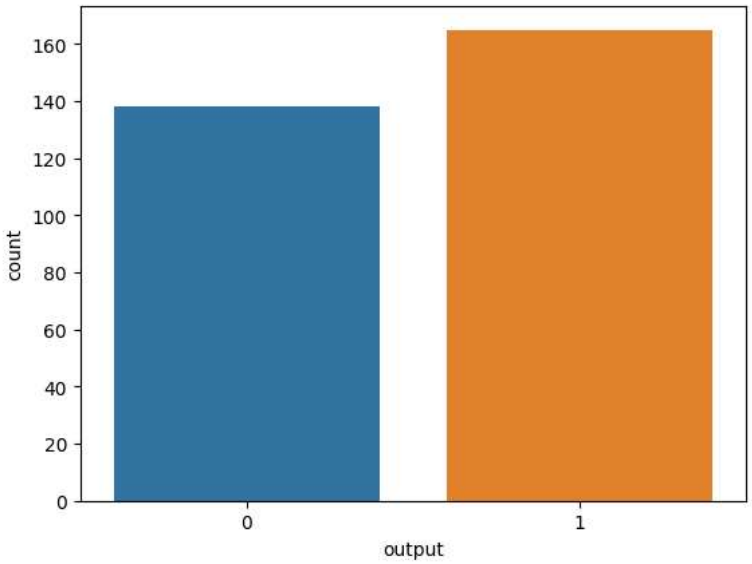


Distributions




```
sns.countplot(x='output',data=heart_data)
```

```
<Axes: xlabel='output', ylabel='count'>
```




```
sns.heatmap(heart_data.isnull(),yticklabels=False)
```

```
<Axes: >
```