In [20]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [21]:

```
data = pd.read_csv('heart.csv')
data
```

Out[21]:

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

303 rows × 14 columns

In [22]:

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

Out[22]:

```
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
```

In [23]:

```
data.describe()
```

Out[23]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

In [24]:

```
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
```

In [25]:

```
data.duplicated().sum()
```

Out[25]:

```
1
```

In [26]:

```
data.drop_duplicates(inplace=True)
```

In [27]:

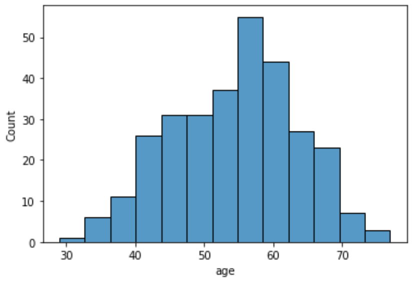
```
data.duplicated().sum()
```

Out[27]:

```
0
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
);
```



```
In [29]: s=data["sex"].value_counts().reset_index()
         px.pie(s,names="index",values="sex",title="%AGE OF MALE AND FEMALE PATIENTS:")
```

```
In [30]: from sklearn.model_selection import train_test_split
```

```
In [31]: x=data.drop("output",axis=1).values
         y=data["output"].values
         x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.5)
```

```
In [32]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         x_train = sc.fit_transform(x_train)
         x_test = sc.fit_transform(x_test)
```

```
In [33]: from sklearn.linear_model import LogisticRegression
         reg = LogisticRegression()
         reg.fit(x_train, y_train)
```

```
Out[33]: LogisticRegression()
```

```
In [34]: reg.score(x_train,y_train)
```

```
Out[34]: 0.8675496688741722
```

```
In [35]: from xgboost import XGBClassifier
         from sklearn.metrics import r2_score

         xgb = XGBClassifier(colsample_bylevel= 0.9,
                             colsample_bytree = 0.8,
                             gamma=0.99,
                             max_depth= 5,
                             min_child_weight= 1,
                             n_estimators= 8,
                             nthread= 5,
                             random_state= 0,
                             )
         xgb.fit(x_train,y_train)
```

```
[20:06:53] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
c:\users\user\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning:

The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_l
abel_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
```

```
Out[35]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=0.9,
                       colsample_bynode=1, colsample_bytree=0.8, gamma=0.99, gpu_id=-1,
                       importance_type='gain', interaction_constraints='',
                       learning_rate=0.300000012, max_delta_step=0, max_depth=5,
                       min_child_weight=1, missing=nan, monotone_constraints='()',
                       n_estimators=8, n_jobs=5, nthread=5, num_parallel_tree=1,
                                         eg_alpha=0, reg_lambda=1, scale_pos_weight=1,
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
              subsample=1, tree_method='exact', validate_parameters=1,
              verbosity=None)
```

In [36]:

```
print('Accuracy of XGBoost classifier on training set: {:.2f}'
     .format(xgb.score(x_train, y_train)))
print('Accuracy of XGBoost classifier on test set: {:.2f}'
     .format(xgb.score(x_test, y_test)))
```

```
Accuracy of XGBoost classifier on training set: 0.97
Accuracy of XGBoost classifier on test set: 0.79
```

In [37]:

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

In [38]:

```
y_pred=xgb.predict(x_test)
print("Accuracy of XG Boost model is:",
accuracy_score(y_test, y_pred)*100)
```

```
Accuracy of XG Boost model is: 79.47019867549669
```

In [ ]:

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js