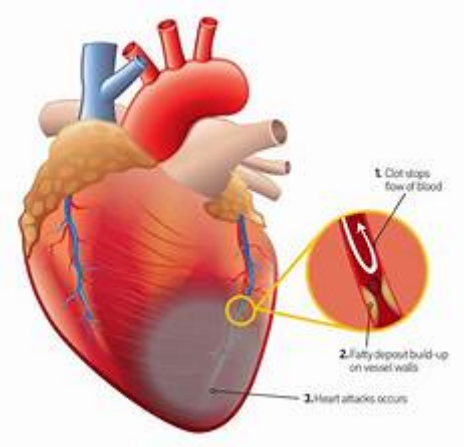
Heart Attack Analysis & Prediction



AIM:- The Purpose of this Project is to create a Machine learning model to predict a person's chance of a heart attack.

Column details

Categorical

```
sex - Gender of person
cp - Chest pain type
caa - number of major vessels (0-3)

fbs - fasting blood sugar (fbs > 120 mg/dl) (1 = true, 0 = false)

restecg - resting electrocardiographic results

(0:normal, 1:ST-T wave abnormality, 2:showing probable or definite left ventricula
r hypertrophy by Estes' criteria)

exng - exercise induced angina (1= yes, 0 = no)

slp - slope
thall - thal rate
```

Continuous

```
trtbps - Resting blood pressure (mm Hg)
chol - cholesterol in mg/dl fetched via BMI sensor (1: typical angina, 2: atypical
angina, 3: non-anginal pain, 4: asymptomatic)

Age - Age of person

thalachh - maximum heart rate achieved
oldpeak - previous peak
```

In [1]:

```
# First Of all Import required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
```

Dataset Link;-[https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset (https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset)](https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset)

In [2]:

```
heart_df = pd.read_csv('heart.csv')
header = ['Saturation Level']
```

In [3]:

```
heart_df.head()
```

Out[3]:

	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

Shape of dataset

In [4]:

```
heart_df.shape
```

Out[4]:

```
(303, 14)
```

Missing Values

In [5]:

```
heart_df.isnull().any()
```

Out[5]:

```
age         False
sex         False
cp          False
trtbps      False
chol        False
fbs         False
restecg     False
thalachh    False
exng        False
oldpeak     False
slp         False
caa         False
thall       False
output      False
dtype: bool
```

You Can See that there is nothing any missing records or values in dataset

In [6]:

```
heart_df.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
```

Statistical Summary

In [7]:

```
1 heart_df.describe()
```

Out[7]:

	age	sex	cp	trtbps	chol	fbs	restecg
count	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
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000

Target

output - target variable (0 = less chance of heart attack, 1 = more chance of heart attack)

In [8]:

```
1 heart_df['output'].value_counts()
```

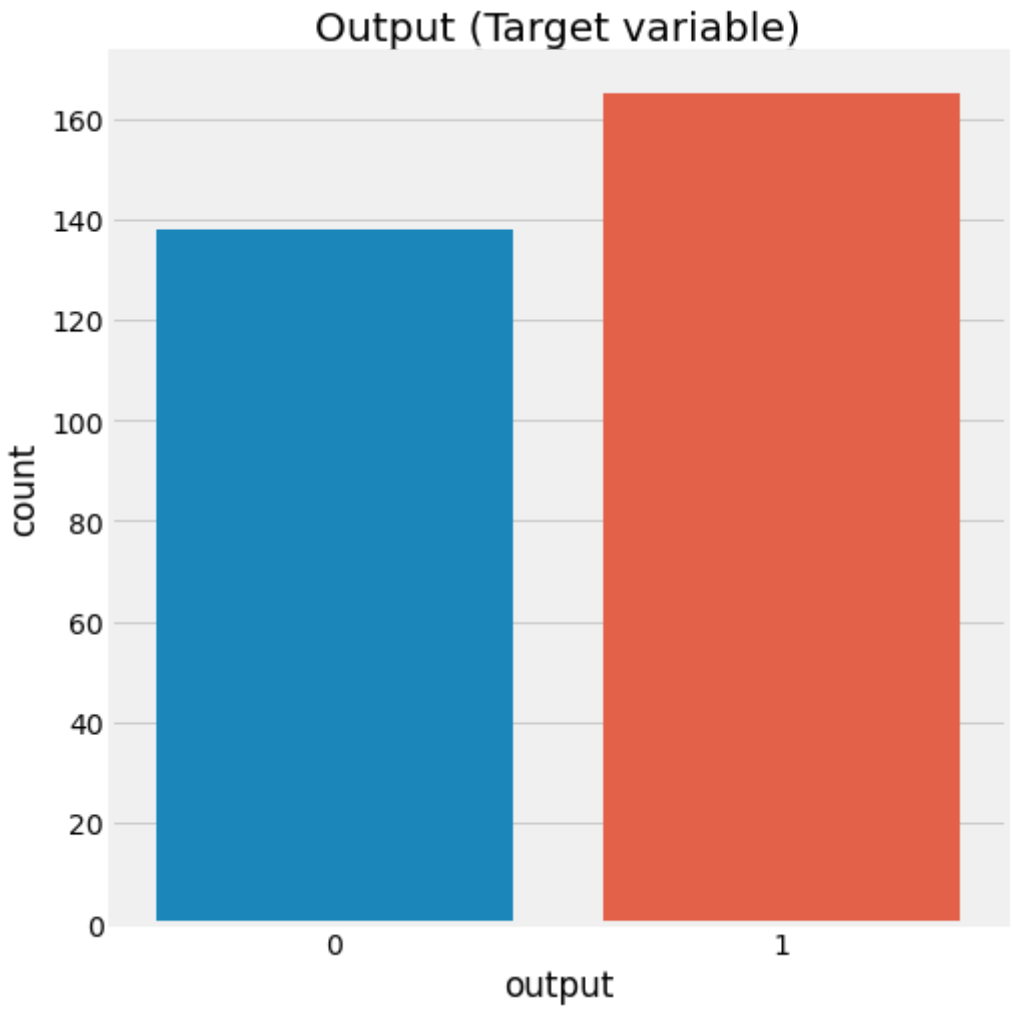
Out[8]:

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

In [9]:

```
plt.figure(figsize = (8,8))
plt.style.use("fivethirtyeight")
plt.title("Output (Target variable)")
sns.countplot(heart_df["output"])
plt.show()
```

```
C:\pythonnew\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only va
lid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Conclusion:

Values are as following: (0 contains: 138) (1 contains:165) in the case of output 0 means a lower chance of heart attack whereas 1 means a higher chance of heart attack (54.45% of patients have a higher chance of heart attack).

Sex

In [10]:

```
1 heart_df.sex.value_counts()
```

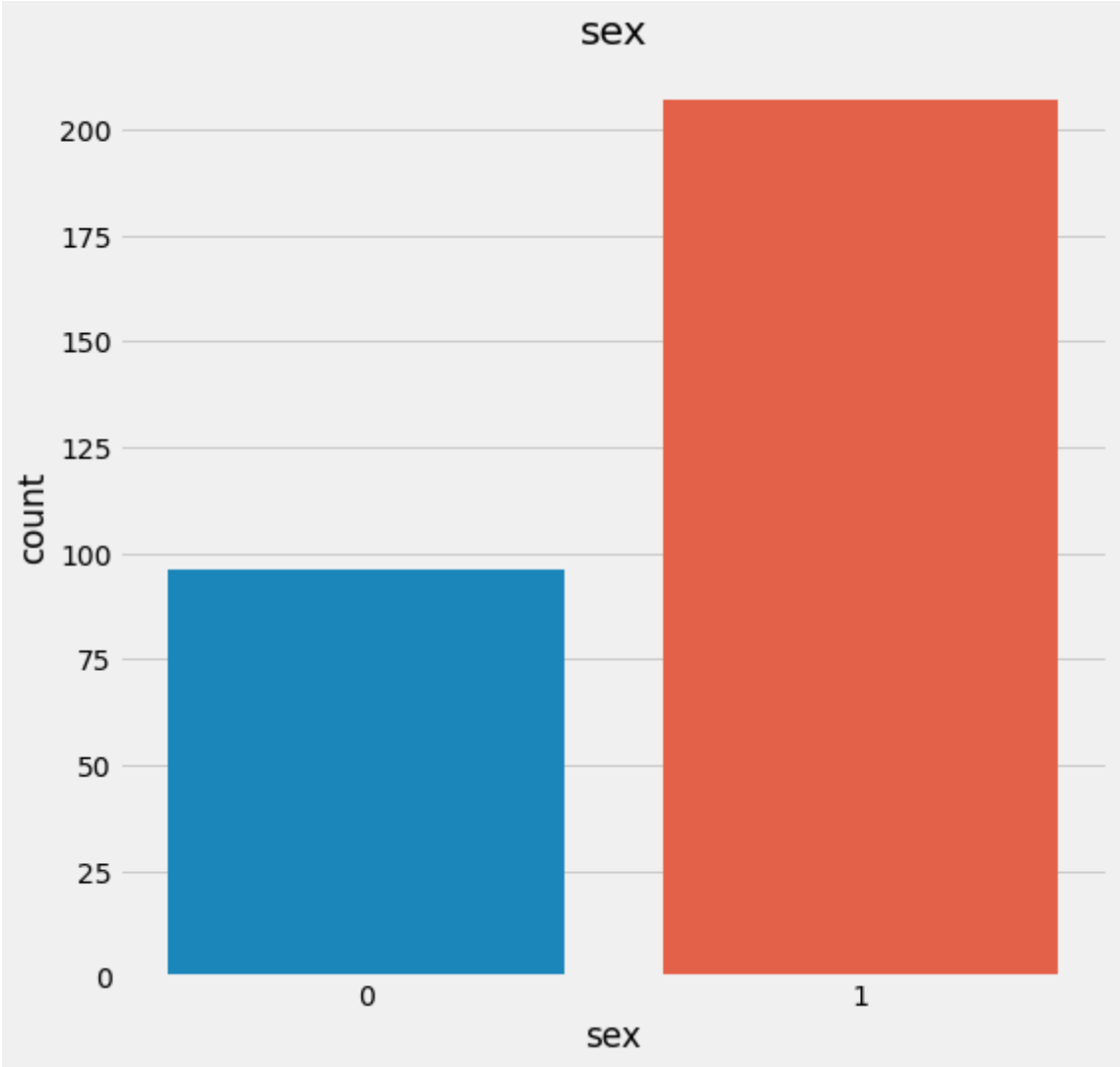
Out[10]:

```
1    207
0     96
Name: sex, dtype: int64
```

In [11]:

```
plt.figure(figsize=(8,8))
plt.title('sex')
sns.countplot(heart_df['sex'])
plt.show()
```

```
C:\pythonnew\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only va
lid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Conclusion:

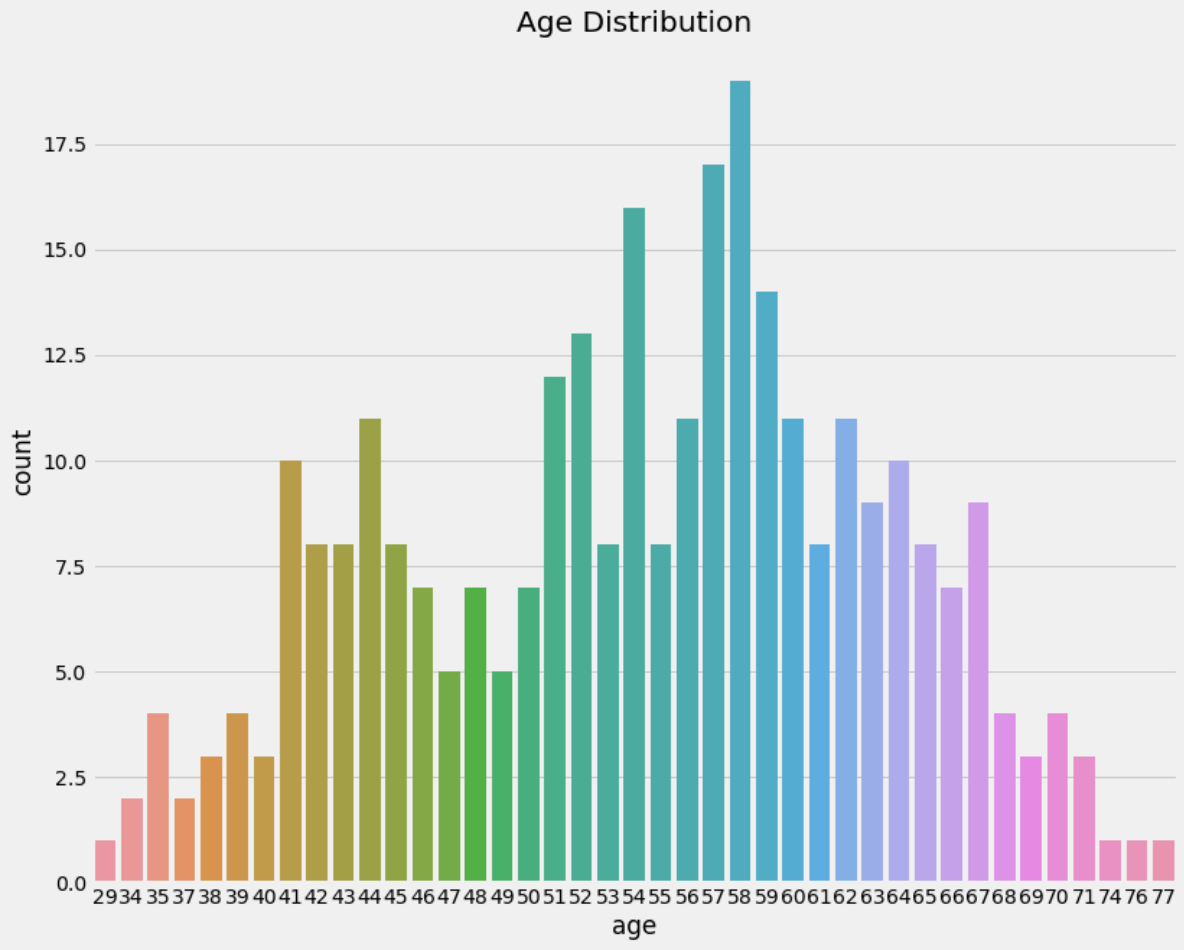
Looking at the difference in gender, a total of (68% are 1) whereas (31% is 0)

Age

In [12]:

```
plt.figure(figsize=(12,10))
plt.title('Age Distribution')
sns.countplot(heart_df['age'])
plt.show()
```

```
C:\pythonnew\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only va
lid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



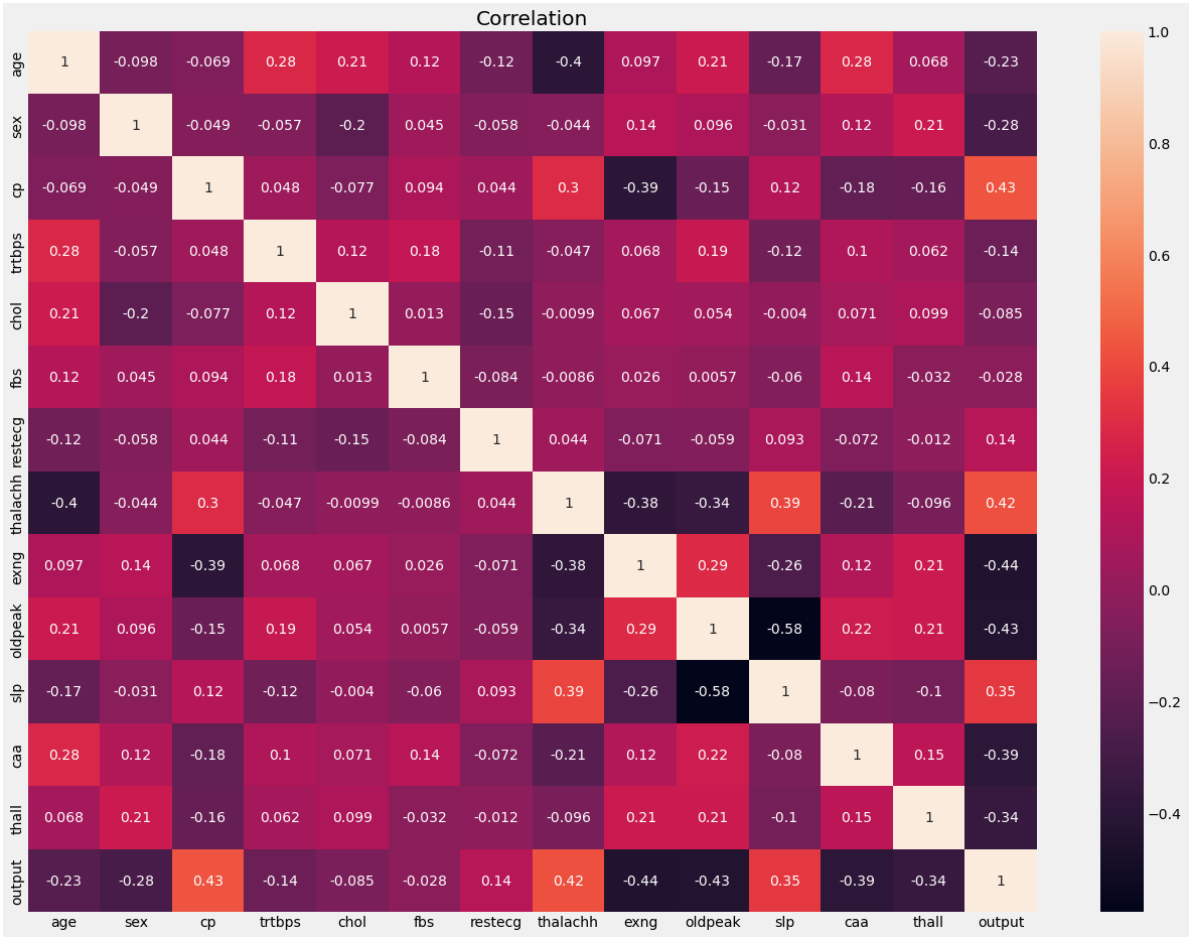
Conclusion:

```
The biggest age group is 58 years old, followed by 57 and 54.
```

Correlation

In [13]:

```
plt.figure(figsize=(20,15))
plt.title('Correlation')
sns.heatmap(heart_df.corr(),annot=True)
plt.show()
heart_df.corr()
```



Out[13]:

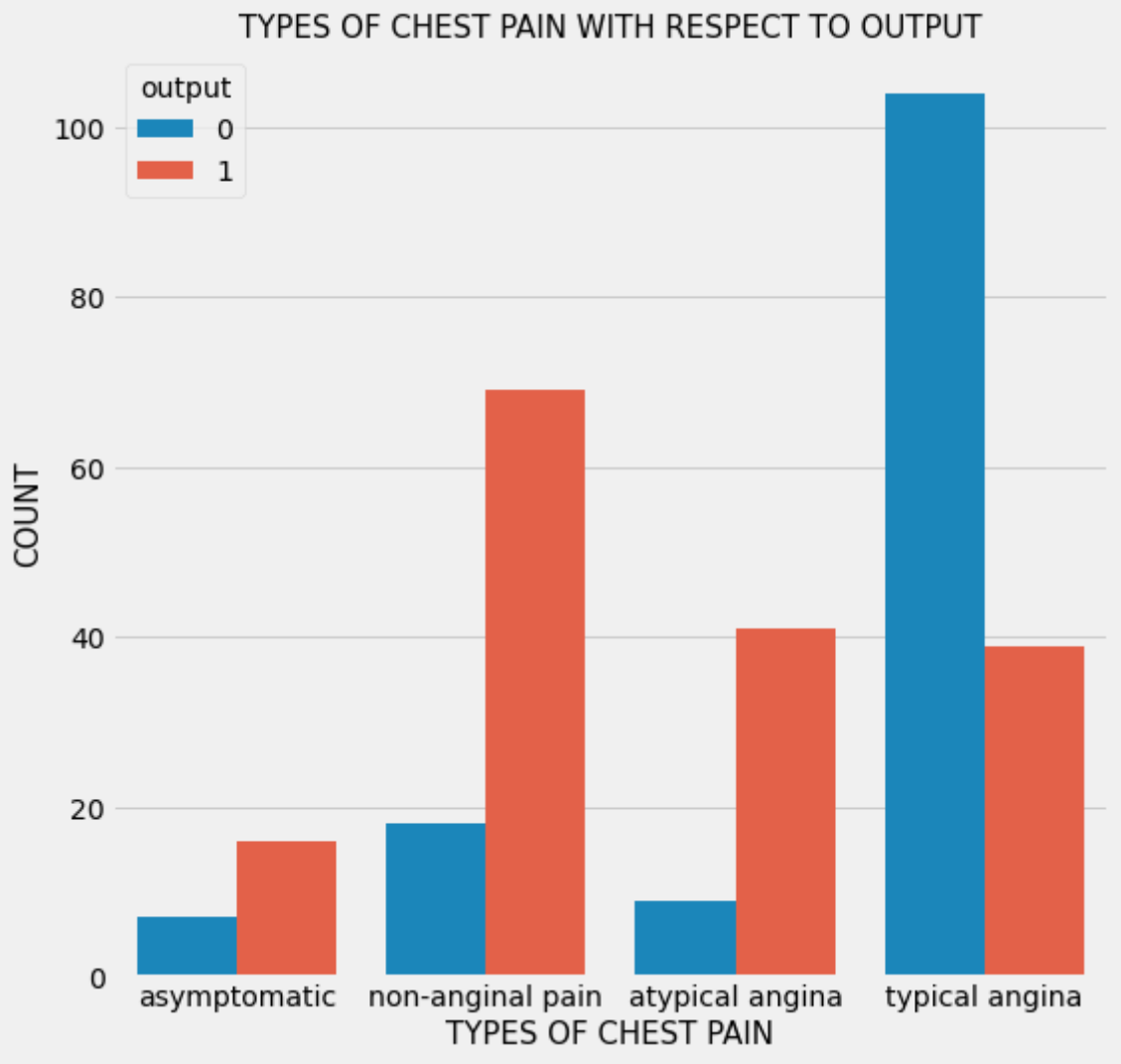
	age	sex	cp	trtbps	chol	fbs	restecg	thalac
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.3985
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.0440
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.2957
trtbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.0466
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.0099
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.0085
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.0441
thalachh	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.0000
exng	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.3788
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.3441
slp	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.3867
caa	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.2131
thall	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.0964

	age	sex	cp	trtbps	chol	fbs	restecg	thalac
output	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.4217

Chest Pain Vs. Output

In [14]:

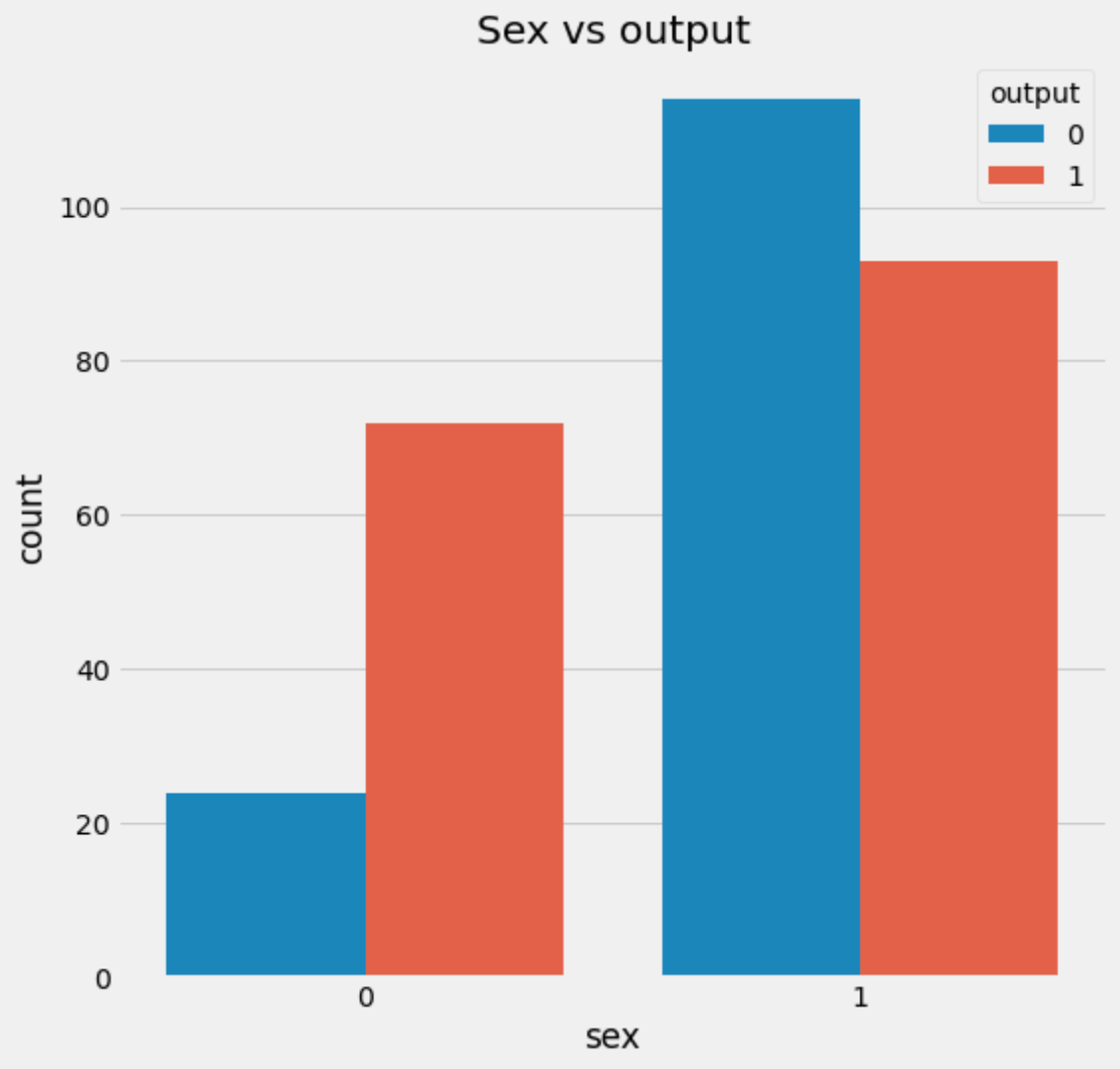
```
plt.figure(figsize=(8,8))
chest_pain=heart_df.cp.map({0:'typical angina',1:'atypical angina',2:'non-anginal pain'
sns.countplot(x=chest_pain,hue='output',data=heart_df)
plt.xlabel('TYPES OF CHEST PAIN',fontsize=15)
plt.ylabel('COUNT',fontsize=15)
plt.title('TYPES OF CHEST PAIN WITH RESPECT TO OUTPUT',fontsize=15)
plt.show()
```



Sex vs. Output

In [15]:

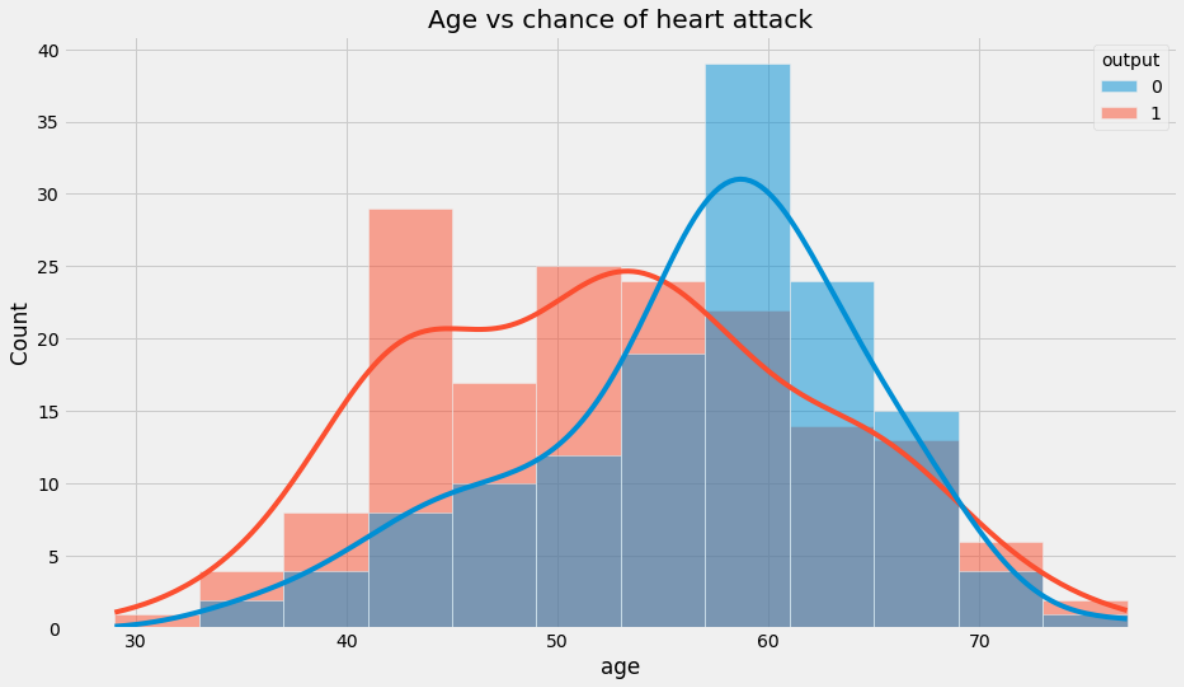
```
plt.figure(figsize = (8,8))
sns.countplot(x=heart_df['sex'], hue=heart_df['output'])
plt.title("Sex vs output")
plt.show()
```



Age vs. chance of heart attack

In [16]:

```
plt.figure(figsize = (14,8))
sns.histplot(x=heart_df['age'],hue=heart_df['output'],kde=True)
plt.title("Age vs chance of heart attack")
plt.show()
```



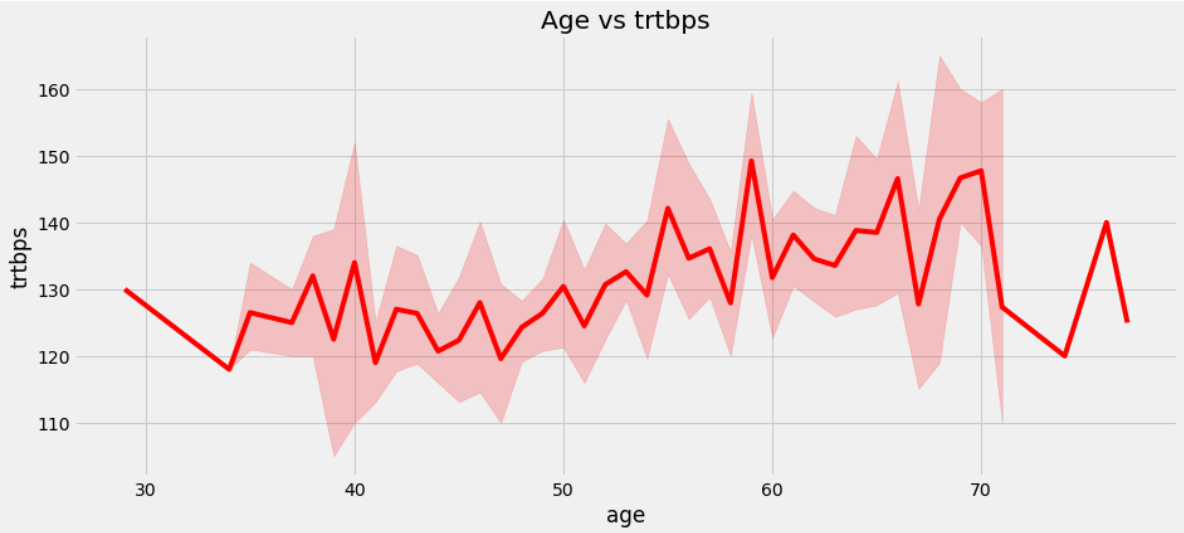
Conclusion:

Heart attacks are more likely between the age of 40 and 55. (output 0 = lower chance, 1 = higher chance)

Age vs. trtbps

In [17]:

```
plt.figure(figsize=(14,6))
plt.style.use
sns.lineplot(x=heart_df['age'],y=heart_df['trtbps'], color='red')
plt.title("Age vs trtbps")
plt.show()
```

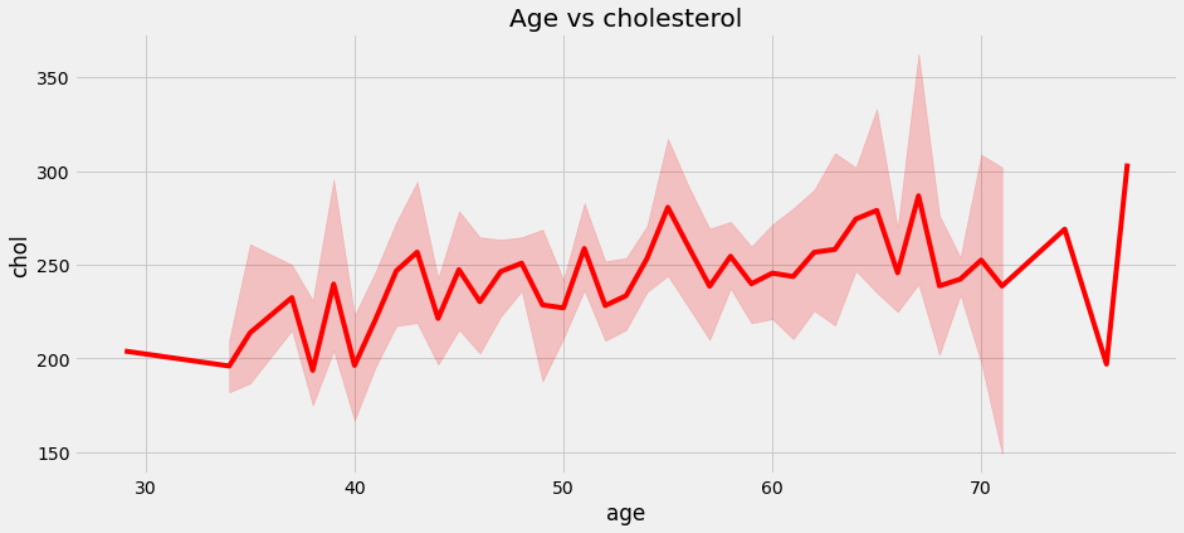


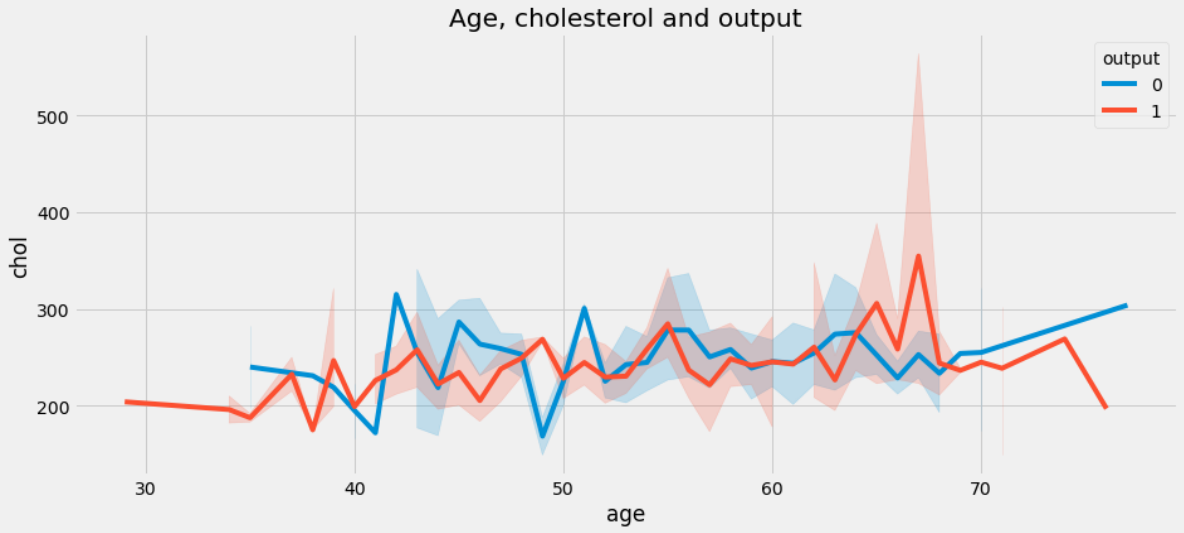
Conclusion:

Theres an increase in resting blood pressure the older you get.

In [18]:

```
plt.figure(figsize=(14,6))
sns.lineplot(x=heart_df['age'],y=heart_df['chol'], color='red')
plt.title("Age vs cholesterol")
plt.figure(figsize=(14,6))
sns.lineplot(x=heart_df['age'],y=heart_df['chol'], hue=heart_df['output'])
plt.title("Age, cholesterol and output")
plt.show()
```





Conclusion:

Theres an increase in cholesterol the older you get but there doesn't seem to be a relation in a higher chance of heart attack with an increase in cholesterol and age

Spliting the dataset

In [19]:

```
# Independent
X=heart_df.drop(['output'],axis=1)
```

In [20]:

```
1 X.head()
```

Out[20]:

	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

In [21]:

```
1 # Dependent
2 y=heart_df.iloc[:,-1]
```

In [22]:

```
1 y.head()
```

Out[22]:

```
0    1
1    1
2    1
3    1
4    1
Name: output, dtype: int64
```

In [23]:

```
1 from sklearn.model_selection import train_test_split
2 X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
```

Feature Scaling

In [24]:

```
1 from sklearn.preprocessing import StandardScaler
2 sc = StandardScaler()
3 X_train = sc.fit_transform(X_train)
4 X_test = sc.transform(X_test)
```

Logistic Regression

In [25]:

```
from sklearn.linear_model import LogisticRegression
classifier1=LogisticRegression(random_state=42)
classifier1.fit(X_train,y_train)
```

Out[25]:

```
LogisticRegression(random_state=42)
```

In [26]:

```
y_pred=classifier1.predict(X_test)
```

In [27]:

```
classifier1.predict(X_test)
```

Out[27]:

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

In [28]:

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

In [29]:

```
acc_score=accuracy_score(y_test, y_pred)
print('accuracy score of LogisticRegression is',acc_score)
```

```
accuracy score of LogisticRegression is 0.8131868131868132
```

KNN

In [30]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier2=KNeighborsClassifier(n_neighbors= 5, p = 2)
classifier2.fit(X_train,y_train)
```

Out[30]:

```
KNeighborsClassifier()
```

In [31]:

```
y_predict=classifier2.predict(X_test)
```

In [32]:

```
classifier2.predict(X_test)
```

Out[32]:

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

In [33]:

```
acc_score=accuracy_score(y_test,y_predict)
print('accuracy Score of KNN is',acc_score)
```

```
accuracy Score of KNN is 0.8681318681318682
```

Random Forest

In [34]:

```
from sklearn.ensemble import RandomForestClassifier
classifier3=RandomForestClassifier(n_estimators = 100, random_state = 42)
classifier3.fit(X_train,y_train)
```

Out[34]:

```
RandomForestClassifier(random_state=42)
```

In [35]:

```
y_predict=classifier3.predict(X_test)
```

In [36]:

```
classifier3.predict(X_test)
```

Out[36]:

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

In [37]:

```
acc_score=accuracy_score(y_test,y_predict)
print('accuracy of RandomForest is',acc_score)
```

```
accuracy of RandomForest is 0.8351648351648352
```

Final

KNN looks the most promising with 86% accuracy