In [6]:
```python
import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.metrics import confusion_matrix
import matplotlib.mlab as mlab
```

In [7]:
```python
df = pd.read_csv('E:\heart.csv')
```

In [8]:
```python
df
```

Out[8]:

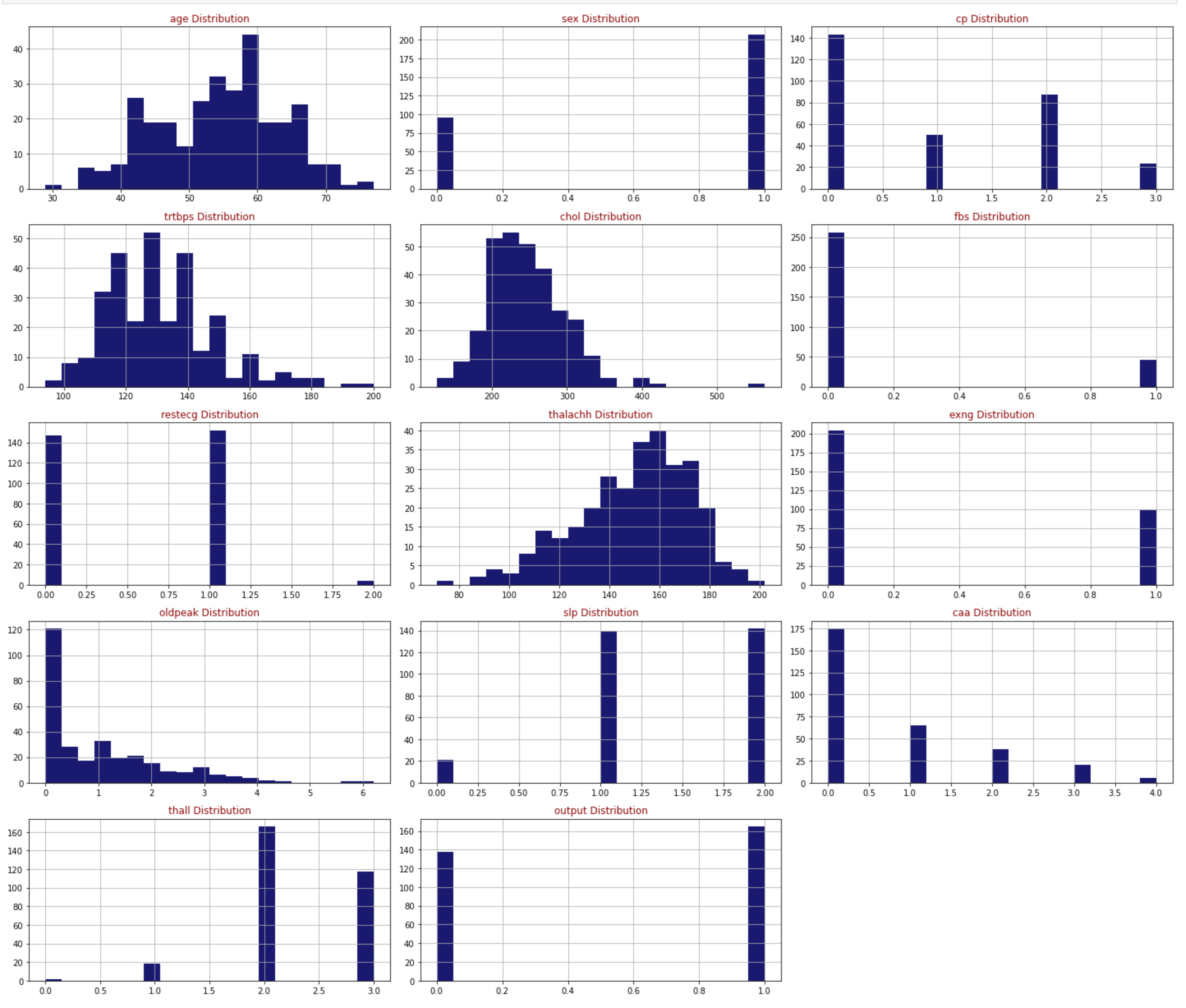
	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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 [9]:
```python
df.isnull().sum()
```

Out[9]:
```
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 [10]:
```python
def draw_histograms(dataframe, features, rows, cols):
    fig=plt.figure(figsize=(20,20))
    for i, feature in enumerate(features):
        ax=fig.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=20,ax=ax,facecolor='midnightblue')
        ax.set_title(feature+" Distribution",color='DarkRed')

    fig.tight_layout()
    plt.show()
draw_histograms(df,df.columns,6,3)
```

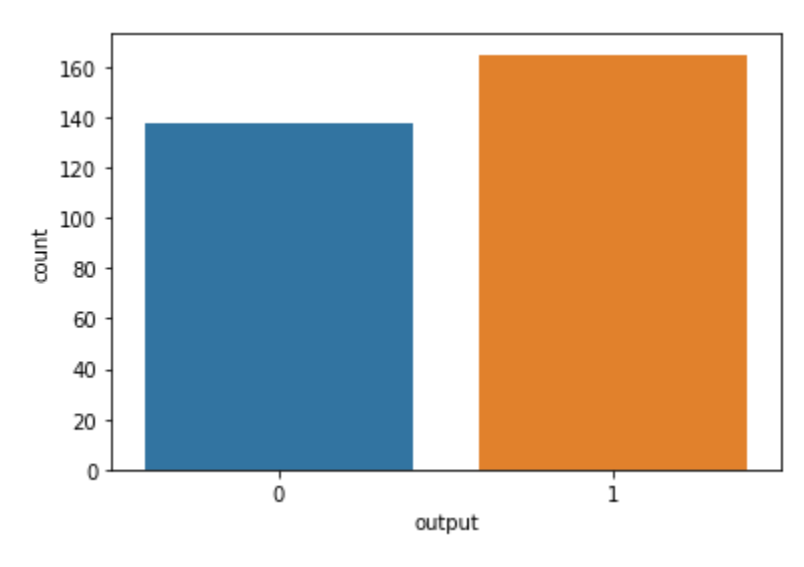


In [11]:
```python
df.output.value_counts()
```

Out[11]:
```
1    165
0    138
Name: output, dtype: int64
```

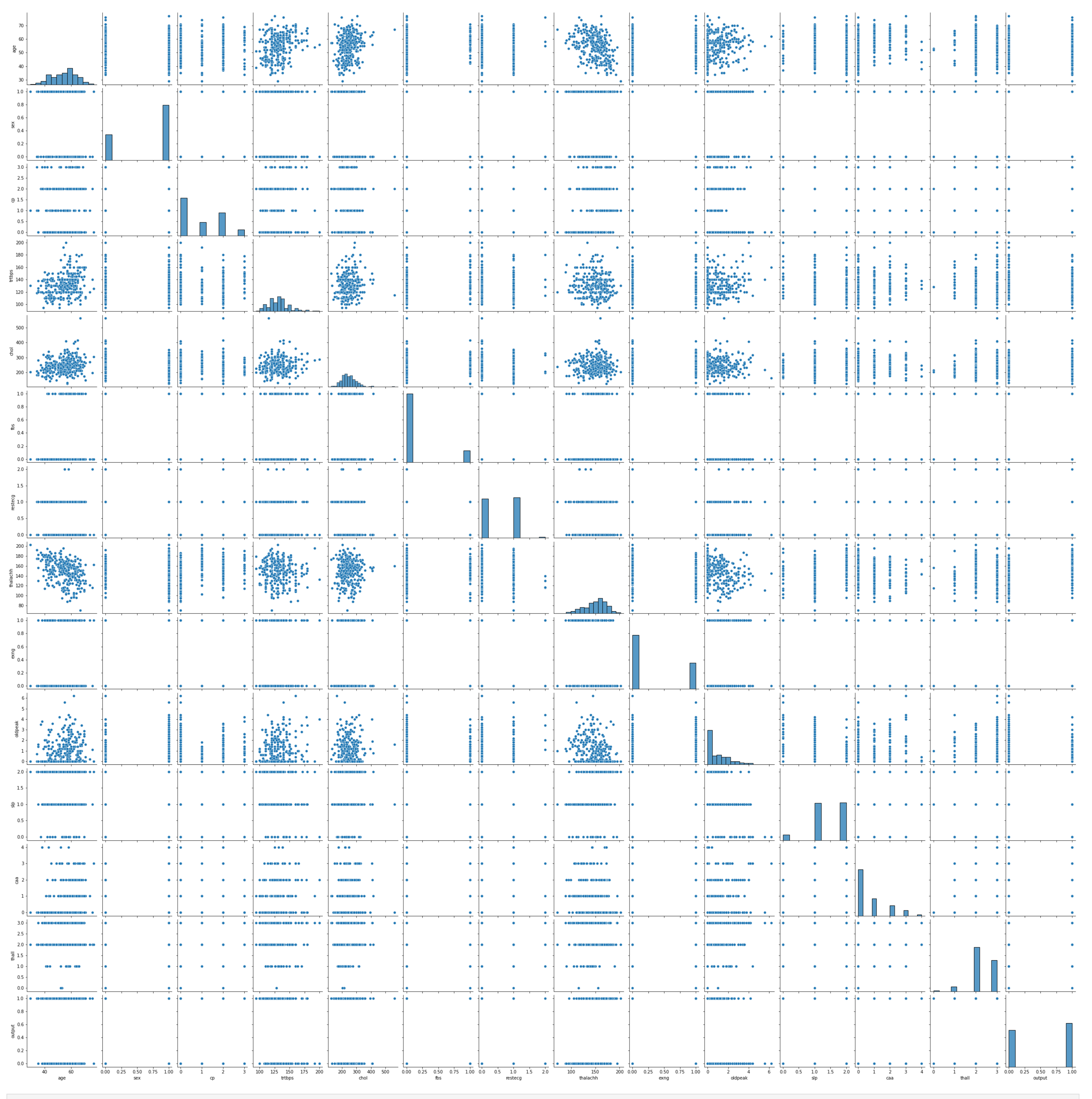
In [13]:
```python
sn.countplot(x='output',data=df)
```

Out[13]: <AxesSubplot:xlabel='output', ylabel='count'>



In [14]:
```python
sn.pairplot(data=df)
```

Out[14]: <seaborn.axisgrid.PairGrid at 0x1bcbe83a520>



In [15]:
```python
df.describe()
```

Out[15]:

	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 [16]:
```python
#machine learning
from statsmodels.tools import add_constant as add_constant
df_constant = add_constant(df)
df_constant.head()
```

Out[16]:

	const	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	1.0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	1.0	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	1.0	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	1.0	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	1.0	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [17]:
```python
st.chisqprob = lambda chisq, df: st.chi2.sf(chisq, df)
cols=df_constant.columns[:-1]
model=sm.Logit(df.output,df_constant[cols])
result=model.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.348904
         Iterations 7
```

Out[17]:

Logit Regression Results

Dep. Variable:	output	**No. Observations:**	303
Model:	Logit	**Df Residuals:**	289
Method:	MLE	**Df Model:**	13
Date:	Tue, 01 Nov 2022	**Pseudo R-squ.:**	0.4937
Time:	21:48:08	**Log-Likelihood:**	-105.72
converged:	True	**LL-Null:**	-208.82
Covariance Type:	nonrobust	**LLR p-value:**	7.262e-37

	coef	std err	z	P>|z|	[0.025	0.975]
const	3.4505	2.571	1.342	0.180	-1.590	8.490
age	-0.0049	0.023	-0.212	0.832	-0.050	0.041
sex	-1.7582	0.469	-3.751	0.000	-2.677	-0.839
cp	0.8599	0.185	4.638	0.000	0.496	1.223
trtbps	-0.0195	0.010	-1.884	0.060	-0.040	0.001
chol	-0.0046	0.004	-1.224	0.221	-0.012	0.003
fbs	0.0349	0.529	0.066	0.947	-1.003	1.073
restecg	0.4663	0.348	1.339	0.181	-0.216	1.149
thalachh	0.0232	0.010	2.219	0.026	0.003	0.044
exng	-0.9800	0.410	-2.391	0.017	-1.783	-0.177
oldpeak	-0.5403	0.214	-2.526	0.012	-0.959	-0.121
slp	0.5793	0.350	1.656	0.098	-0.106	1.265
caa	-0.7733	0.191	-4.051	0.000	-1.147	-0.399
thall	-0.9004	0.290	-3.104	0.002	-1.469	-0.332

In [19]:
```python
def back_feature_elem (data_frame,dep_var,col_list):

    while len(col_list)>0 :
        model=sm.Logit(dep_var,data_frame[col_list])
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,3).nlargest(1)
        if largest_pvalue[0]<(0.05):
            return result
            break
        else:
            col_list=col_list.drop(largest_pvalue.index)

result=back_feature_elem(df_constant,df.output,cols)
```

In [20]:
```python
result.summary()
```

Out[20]:

Logit Regression Results

Dep. Variable:	output	**No. Observations:**	303
Model:	Logit	**Df Residuals:**	296
Method:	MLE	**Df Model:**	6
Date:	Tue, 01 Nov 2022	**Pseudo R-squ.:**	0.4651
Time:	21:51:36	**Log-Likelihood:**	-111.71
converged:	True	**LL-Null:**	-208.82
Covariance Type:	nonrobust	**LLR p-value:**	3.209e-39

	coef	std err	z	P>|z|	[0.025	0.975]
sex	-1.3898	0.405	-3.431	0.001	-2.184	-0.596
cp	0.7861	0.174	4.509	0.000	0.444	1.128
thalachh	0.0261	0.004	5.905	0.000	0.017	0.035
exng	-1.0130	0.376	-2.695	0.007	-1.750	-0.276
oldpeak	-0.7262	0.176	-4.130	0.000	-1.071	-0.382
caa	-0.7053	0.173	-4.087	0.000	-1.043	-0.367
thall	-0.8674	0.259	-3.351	0.001	-1.375	-0.360

In [21]:
```python
params = np.exp(result.params)
conf = np.exp(result.conf_int())
conf['OR'] = params
pvalue=round(result.pvalues,3)
conf['pvalue']=pvalue
conf.columns = ['CI 95%(2.5%)', 'CI 95%(97.5%)', 'Odds Ratio','pvalue']
print ((conf))
```

```
          CI 95%(2.5%)  CI 95%(97.5%)  Odds Ratio  pvalue
sex           0.112623       0.551073    0.249126   0.001
cp            1.559575       3.088655    2.194764   0.000
thalachh      1.017567       1.035326    1.026408   0.000
exng          0.173839       0.758508    0.363123   0.007
oldpeak       0.342750       0.682775    0.483757   0.000
caa           0.352232       0.692750    0.493973   0.000
thall         0.252918       0.697612    0.420046   0.001
```

In [27]:
```python
import sklearn
new_features=df[['age','sex','cp','thalachh','exng','oldpeak','caa','thall','output']]
x=new_features.iloc[:,:-1]
y=new_features.iloc[:,-1]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=5)
```

In [28]:
```python
x_train
```

Out[28]:

	age	sex	cp	thalachh	exng	oldpeak	caa	thall
266	55	0	0	117	1	3.4	0	2
215	43	0	0	136	1	3.0	0	3
99	53	1	2	173	0	0.0	3	2
119	46	0	0	152	1	0.0	0	2
11	48	0	2	139	0	0.2	0	2
...	...	...	...	...	...	...	...	...
8	52	1	2	162	0	0.5	0	3
73	51	1	0	186	1	0.0	0	2
118	46	0	1	172	0	0.0	0	2
189	41	1	0	158	0	0.0	0	3
206	59	1	0	142	1	1.2	1	3

242 rows × 8 columns

In [29]:
```python
y_train
```

Out[29]:
```
266    0
215    0
99     1
119    1
11     1
      ..
8      1
73     1
118    1
189    0
206    0
Name: output, Length: 242, dtype: int64
```

In [30]:
```python
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(x_train,y_train)
y_pred=logreg.predict(x_test)
```

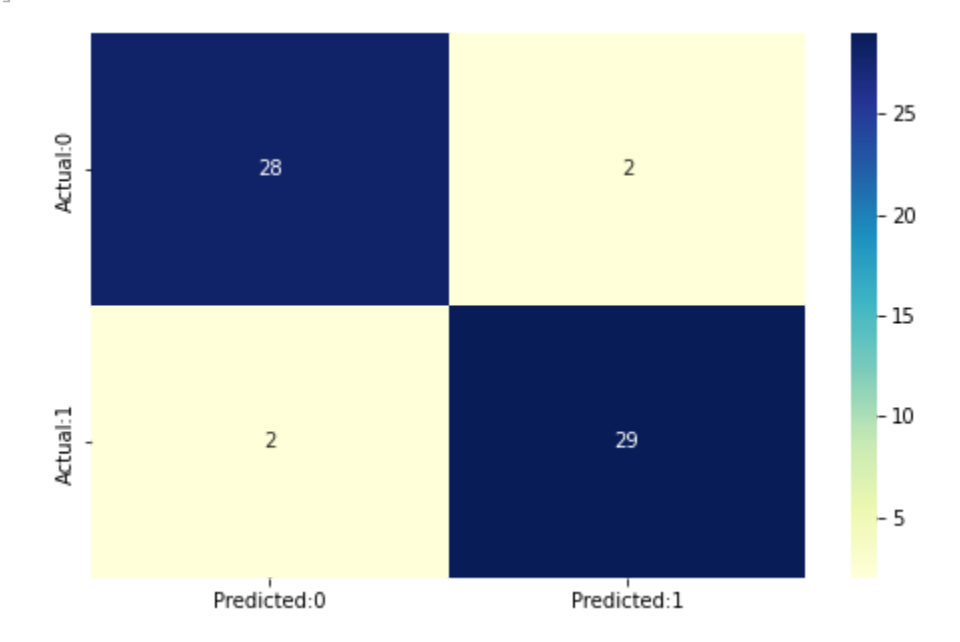
```
E:\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown 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(
```

In [31]:
```python
sklearn.metrics.accuracy_score(y_test,y_pred)
```

Out[31]: 0.9344262295081968

In [32]:
```python
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sn.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")
```

Out[32]: <AxesSubplot:>



In [33]:
```python
y_pred_prob=logreg.predict_proba(x_test)[:,:]
y_pred_prob_df=pd.DataFrame(data=y_pred_prob, columns=['Prob of no heart disease (0)','Prob of Heart Disease (1)'])
y_pred_prob_df.head()
```

Out[33]:

	Prob of no heart disease (0)	Prob of Heart Disease (1)
0	0.063587	0.936413
1	0.984729	0.015271
2	0.987555	0.012445
3	0.634431	0.365569
4	0.285850	0.714150

In [ ]: