Heart Attack Analysis & Prediction Dataset

Data Resource : https://www.kaggle.com/rashikrahmanpritom/heart-attack-analysis-prediction-dataset (https://www.kaggle.com/rashikrahmanpritom/heart-attack-analysis-prediction-dataset)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Import Data

In [2]:
```
Heart = pd.read_csv('heart.csv')
```

In [3]:
```
Heart.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 [4]:
```
Heart.describe()
```

Out[4]:

	age	sex	cp	trtbps	chol	fbs	restecg	th
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.

In [5]:
```
Heart.nunique()
```

Out[5]:
```
age          41
sex           2
cp            4
trtbps       49
chol        152
fbs           2
restecg       3
thalachh     91
exng          2
oldpeak      40
slp           3
caa           5
thall         4
output        2
dtype: int64
```

Exploratory Data Analysis

In [6]:
```
Heart.head(10)
```

Out[6]:

	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
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

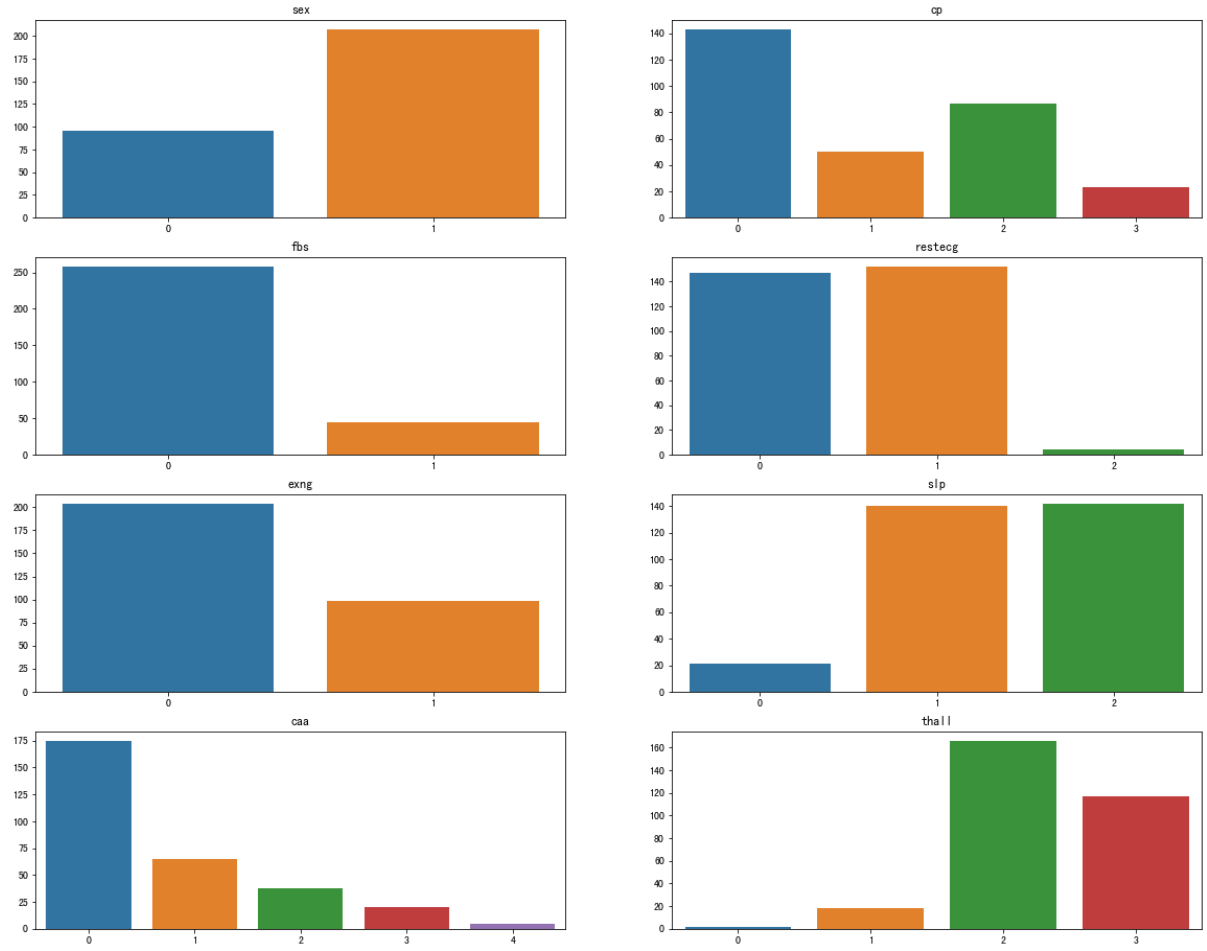
In [7]:
```
cat = ['sex','cp','fbs','restecg','exng','slp','caa','thall']
len(cat)
```

Out[7]:
```
8
```

In [8]:
```
Heart['sex'].value_counts().index
```

Out[8]:
```
Int64Index([1, 0], dtype='int64')
```

In [9]:
```
fig,  ax = plt.subplots(nrows=4, ncols=2, figsize=(20,16))
r = 0
for i in range(4):
    for j in range(2):
        category = [Heart[i].value_counts() for i in cat]
        axes = ax[i][j]
        sns.barplot(x=category[r].index,y=category[r].values,ax=axes)
        axes.set_title(cat[r])
        r+=1
plt.show()
```



About this dataset

· sex : sex of the patient (0 = Female; 1 = Male)

· exang: exercise induced angina (1 = yes; 0 = no)

· ca: number of major vessels (0-3)

· cp : Chest Pain type chest pain type

```
Value 1: typical angina
Value 2: atypical angina
Value 3: non-anginal pain
Value 4: asymptomatic
```

· fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

· rest_ecg : resting electrocardiographic results

```
Value 0: normal
Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation
or depression of > 0.05 mV)
Value 2: showing probable or definite left ventricular hypertrophy by Estes'
criteria
```

Analysis

· ex : 68% Man (207 / (207 + 96))

· cp : 47% of typical angina (143 / (143 + 87 + 50 + 23))

· fbs : 85% of people have fasting blood sugar <= 120 mg/dl (45 / (258 + 45))

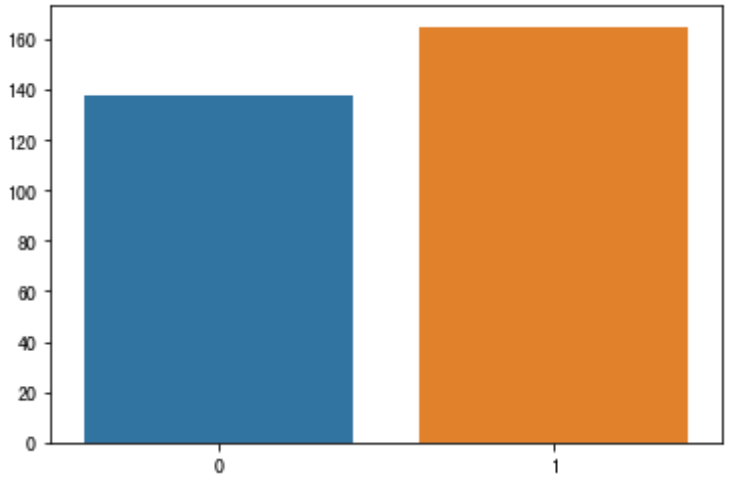
· rest_ecg : normal & ST-T wave abnormality are 98% (152 + 147) / (152 + 147 + 4)

· exang: 67% people exercise "without" induced angina (204 / (204 + 99))

As we know 55% of the people have high risk of Heart Attack

In [10]:
```
target = Heart.output.value_counts()
sns.barplot(x=target.index,y=target.values)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcc97002dc0>



Correlation

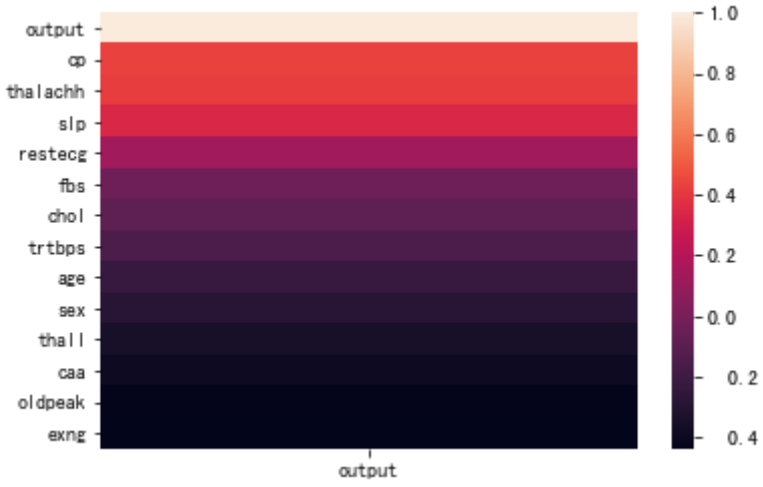
In [11]:
```
Output = pd.DataFrame(Heart.corr()['output'].sort_values(ascending=False
))
Heart.corr()['output'].sort_values(ascending=False)
```

Out[11]:
```
output      1.000000
cp          0.433798
thalachh    0.421741
slp         0.345877
restecg     0.137230
fbs        -0.028046
chol       -0.085239
trtbps     -0.144931
age        -0.225439
sex        -0.280937
thall      -0.344029
caa        -0.391724
oldpeak    -0.430696
exng       -0.436757
Name: output, dtype: float64
```

In [12]:
```
sns.heatmap(Output)
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-pa
ckages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyph 87
22 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-pa
ckages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyph 87
22 missing from current font.
  font.set_text(s, 0, flags=flags)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcc97eeeb80>



Data Preprocessing

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
train = Heart.iloc[:,:-1]
test = Heart.iloc[:,-1:]
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(train,test, test_siz
e=0.3, random_state=3)
```

In [16]:
```
X_train
```

Out[16]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
197	67	1	0	125	254	1	1	163	0	0.2	1	2	3
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2
227	35	1	0	120	198	0	1	130	1	1.6	1	0	3
146	44	0	2	118	242	0	1	149	0	0.3	1	1	2
115	37	0	2	120	215	0	1	170	0	0.0	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
277	57	1	1	124	261	0	1	141	0	0.3	2	0	3
256	58	1	0	128	259	0	0	130	1	3.0	1	2	3
131	49	0	1	134	271	0	1	162	0	0.0	1	0	2
249	69	1	2	140	254	0	0	146	0	2.0	1	3	3
152	64	1	3	170	227	0	0	155	0	0.6	1	0	3

212 rows × 13 columns

In [17]:
```
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

```
(212, 13) (91, 13)
(212, 1) (91, 1)
```

In [18]:
```
from sklearn.preprocessing import StandardScaler
```

In [19]:
```
scaler = StandardScaler()
X_train_raw = scaler.fit_transform(X_train)
X_test_raw = scaler.transform(X_test)
```

In [20]:
```
X_train = pd.DataFrame(X_train_raw, columns=X_train.columns, index=X_tra
in.index)
X_test = pd.DataFrame(X_test_raw, columns=X_test.columns, index=X_test.i
ndex)
```

In [21]:
```
X_train
```

Out[21]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	e
197	1.397056	0.650245	-0.901439	-0.356084	0.154583	2.617604	0.878509	0.652301	-0.717
15	-0.480271	-1.537881	1.068716	-0.656362	-0.492710	-0.382029	0.878509	0.429184	-0.717
227	-2.136735	0.650245	-0.901439	-0.656362	-0.881086	-0.382029	0.878509	-0.820270	1.394
146	-1.142857	-1.537881	1.068716	-0.776473	-0.067346	-0.382029	0.878509	0.027574	-0.717
115	-1.915873	-1.537881	1.068716	-0.656362	-0.566687	-0.382029	0.878509	0.964665	-0.717
...	...	...	...	...	...	...	...	...	
277	0.292746	0.650245	0.083639	-0.416140	0.284041	-0.382029	0.878509	-0.329413	-0.717
256	0.403177	0.650245	-0.901439	-0.175918	0.247053	-0.382029	-1.021939	-0.820270	1.394
131	-0.590702	-1.537881	0.083639	0.184416	0.468982	-0.382029	0.878509	0.607678	-0.717
249	1.617918	0.650245	1.068716	0.544749	0.154583	-0.382029	-1.021939	-0.106296	-0.717
152	1.065763	0.650245	2.053793	2.346417	-0.344758	-0.382029	-1.021939	0.295314	-0.717

212 rows × 13 columns

Modeling Using LogisticRegression

In [22]:
```
from sklearn.linear_model import LogisticRegression
```

In [23]:
```
lr = LogisticRegression()
```

In [24]:
```
lr.fit(X_train,y_train.values.ravel())
y_pred = lr.predict(X_test)
```

In [25]:
```
(y_pred == y_test.values.ravel()).sum() / len(y_pred)
```

Out[25]: 0.8791208791208791

In [26]:
```
from sklearn.metrics import accuracy_score
```

In [27]:
```
accuracy_score(y_pred, y_test)
```

Out[27]: 0.8791208791208791