

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

```
In [2]: heart_df=pd.read_csv("heart.csv")
```

```
In [3]: heart_df.shape
```

```
Out[3]: (303, 15)
```

```
In [4]: heart_df.head()
```

```
Out[4]:
```

	Unnamed: 0	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
In [5]: heart_df.tail()
```

```
Out[5]:
```

	Unnamed: 0	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
298	298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

```
In [6]: heart_df.describe()
```

```
Out[6]:
```

	Unnamed: 0	age	sex	cp	trestbps	chol	fbs	restecg	thalach
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	151.000000	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865
std	87.612784	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161
min	0.000000	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000
25%	75.500000	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000
50%	151.000000	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000
75%	226.500000	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000
max	302.000000	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000

```
In [7]: heart_df.duplicated().sum()
```

```
Out[7]: 0
```

```
In [8]: heart_df.drop('Unnamed: 0',axis=1,inplace=True)
```

```
In [9]: heart_df.columns
```

```
Out[9]: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
              'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
              dtype='object')
```

```
In [12]: x=heart_df.drop('target',axis=1)
         y=heart_df['target']
```

```
In [13]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(x,y, test_size=0.2,random_state=42)
```

```
In [16]: from sklearn.linear_model import LogisticRegression
         log_clf=LogisticRegression()
         log_clf.fit(X_train,y_train)
```

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model_logistic.py:469: 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(

```
Out[16]: LogisticRegression
```

```
LogisticRegression()
```

```
In [17]: y_predict=log_clf.predict(X_test)
```

```
In [20]: from sklearn import
         metrics
         conf_mat=metrics.confusion_matrix(y_test,y_predict)
         conf_mat
```

```
Out[20]: array([[25,  4],
               [ 3, 29]], dtype=int64)
```

```
In [21]: acc=metrics.accuracy_score(y_test,y_predict)
         prec=metrics.precision_score(y_test,y_predict)
         recall=metrics.recall_score(y_test,y_predict)
         print(f'Accuracy={acc}\nPrecision={prec}\nRecall={recall}')
```

Accuracy=0.8852459016393442
Precision=0.8787878787878788
Recall=0.90625

```
In [23]: plt.title('confusion Matrix')
         sns.heatmap(pd.DataFrame(conf_mat),annot=True,cmap='YlGnBu')
         plt.show()
```

