

In [1]: `#LOGISTIC REGRESSION`

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

In [2]: `df = pd.read_csv('DATA/heart.csv')`

In [3]: `df.head()`

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
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 [4]: `df['target'].unique()`

Out[4]: `array([1, 0])`

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

Out[6]:

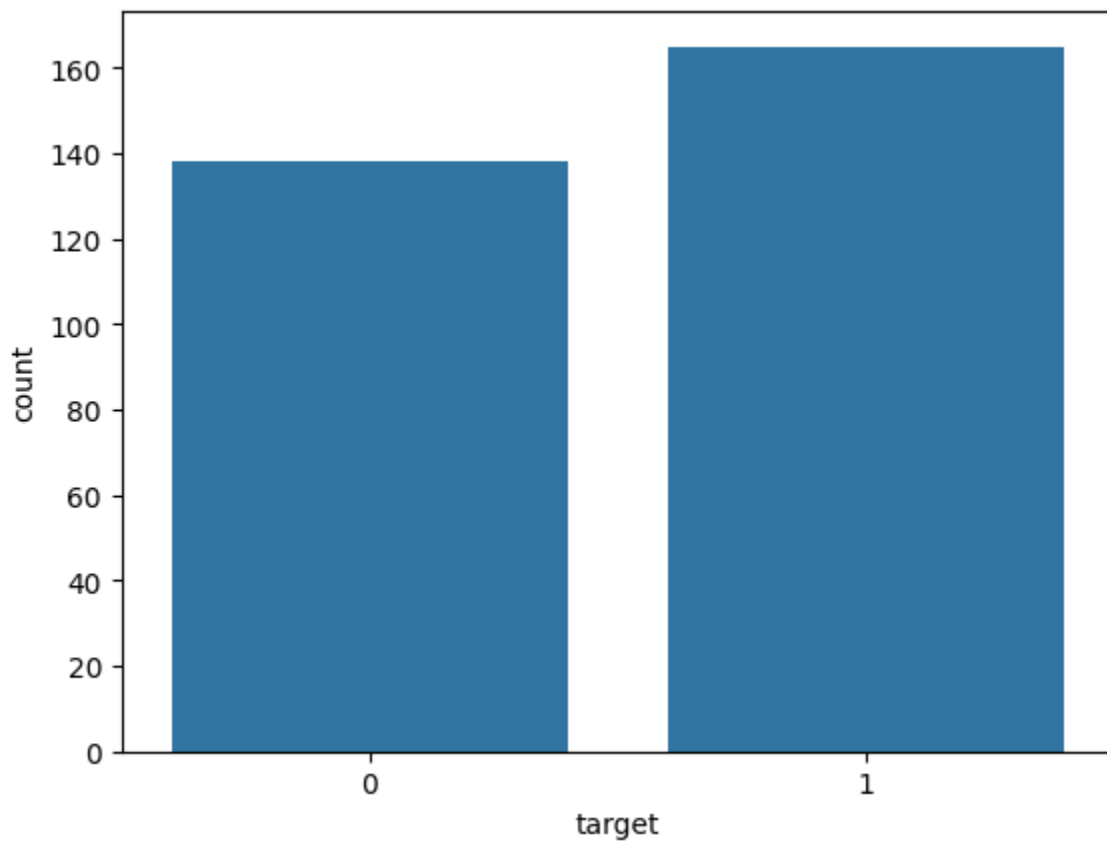
	age	sex	cp	trestbps	chol	fbs	restecg
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860
min	29.000000	0.000000	0.000000	94.000000	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

In [7]: `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   trestbps    303 non-null    int64  
4   chol        303 non-null    int64  
5   fbs         303 non-null    int64  
6   restecg     303 non-null    int64  
7   thalach     303 non-null    int64  
8   exang       303 non-null    int64  
9   oldpeak     303 non-null    float64  
10  slope       303 non-null    int64  
11  ca          303 non-null    int64  
12  thal        303 non-null    int64  
13  target      303 non-null    int64  
dtypes: float64(1), int64(13)  
memory usage: 33.3 KB
```

```
In [8]: sns.countplot(x='target',data=df)
```

```
Out[8]: <Axes: xlabel='target', ylabel='count'>
```

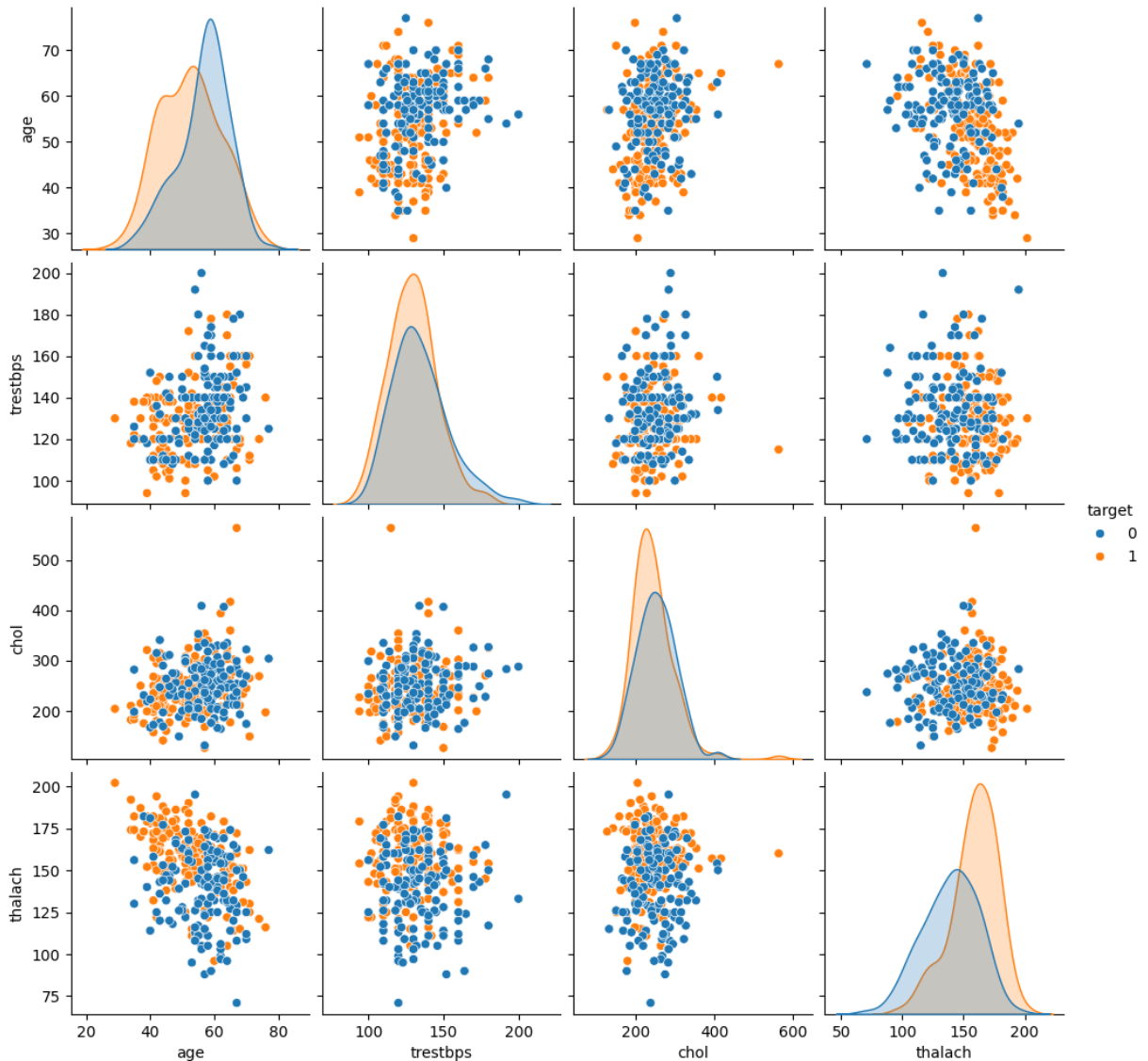


```
In [9]: df.columns
```

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

```
In [11]: sns.pairplot(df[['age', 'trestbps', 'chol', 'thalach', 'target']], hue='target')
```

```
Out[11]: <seaborn.axisgrid.PairGrid at 0x1ca01464b60>
```

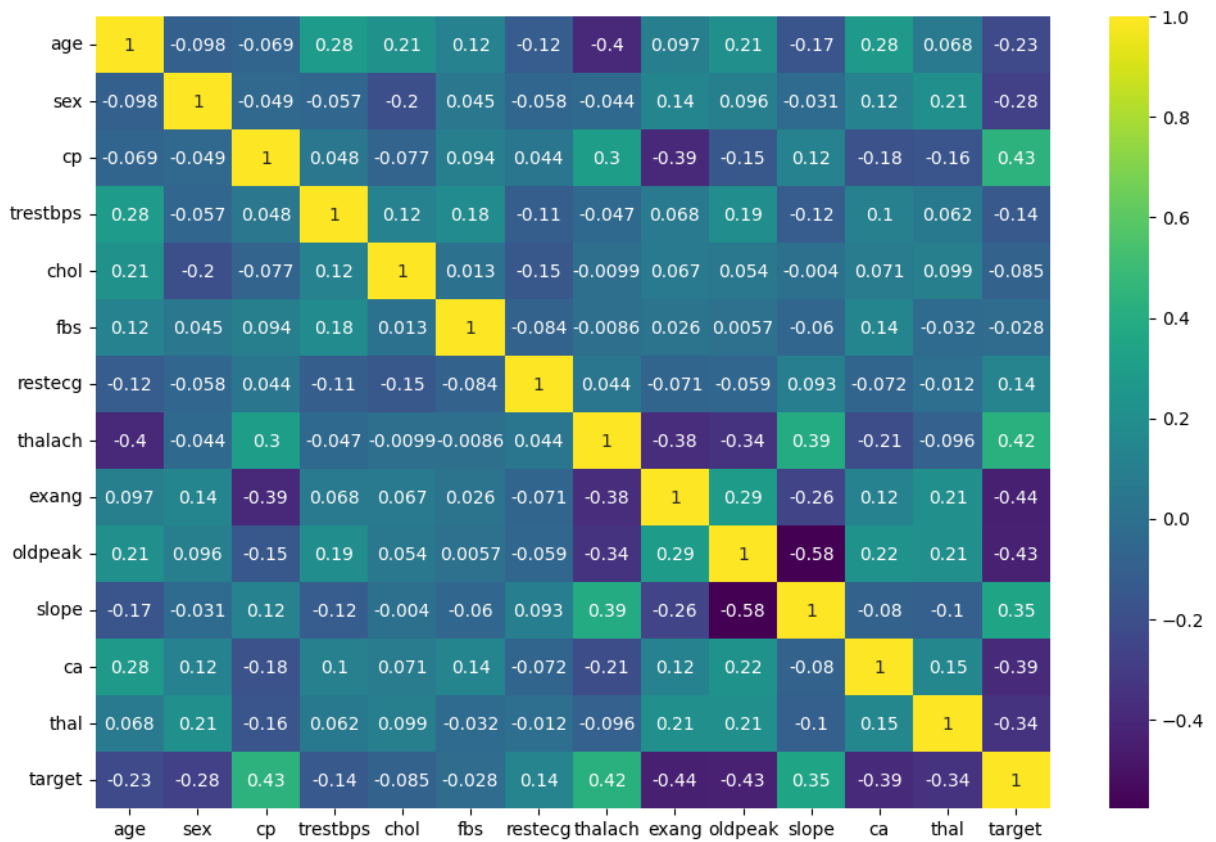


```
In [12]: plt.figure(figsize=(12,8))
```

```
Out[12]: <Figure size 1200x800 with 0 Axes>  
<Figure size 1200x800 with 0 Axes>
```

```
In [15]: plt.figure(figsize=(12,8))  
sns.heatmap(df.corr(), cmap='viridis', annot=True)
```

```
Out[15]: <Axes: >
```



```
In [17]: X=df.drop('target',axis=1)
         y=df['target']
```

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

```
In [19]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_sta
```

```
In [20]: scaler = StandardScaler()
         scaler_X_train = scaler.fit_transform(X_train)
         scaled_X_test =scaler.transform(X_test)
```

```
In [21]: scaler_X_train
```

```
Out[21]: array([[ 1.04970247,  0.69737995,  1.97993226, ..., -0.67167968,
                 -0.71422572, -0.4842146 ],
                [ 0.61644136,  0.69737995, -0.94692412, ...,  0.9451068 ,
                 1.23823052,  1.14178999],
                [-0.68334197,  0.69737995,  0.02869467, ..., -2.28846615,
                 -0.71422572,  1.14178999],
                ...,
                [-0.89997253,  0.69737995,  0.02869467, ...,  0.9451068 ,
                 -0.71422572,  1.14178999],
                [-1.44154891,  0.69737995,  0.02869467, ..., -0.67167968,
                 -0.71422572, -2.11021919],
                [-0.68334197, -1.43393855,  1.00431346, ...,  0.9451068 ,
                 -0.71422572, -0.4842146 ]], shape=(272, 13))
```

```
In [22]: scaled_X_test
```

```
Out[22]: array([[ 0.07486497,  0.69737995, -0.94692412,  0.47001573, -0.55502259,
-0.42732739,  0.89828426, -1.70935702,  1.47064295,  4.11840557,
-2.28846615, -0.71422572,  1.14178999],
 [-0.25008086,  0.69737995,  1.00431346,  2.28905391, -0.89683273,
 2.34012617,  0.89828426,  0.52722104, -0.6799747 , -0.43237682,
 0.9451068 , -0.71422572,  1.14178999],
 [ 0.3998108 , -1.43393855, -0.94692412, -1.803782 ,  0.03365043,
-0.42732739, -0.99577247, -1.22695783, -0.6799747 ,  0.01377832,
-0.67167968, -0.71422572, -0.4842146 ],
 [-1.65817947, -1.43393855,  1.00431346,  0.35632584, -0.49805423,
-0.42732739,  0.89828426,  0.08867632, -0.6799747 , -0.87853195,
-0.67167968, -0.71422572, -0.4842146 ],
 [ 0.61644136,  0.69737995, -0.94692412, -0.38265842,  0.22354495,
-0.42732739, -0.99577247, -0.39372287,  1.47064295,  1.61993681,
-0.67167968,  0.2620024 ,  1.14178999],
 [-0.57502669,  0.69737995,  0.02869467, -0.09843371,  0.37546057,
-0.42732739,  0.89828426,  0.92191129, -0.6799747 , -0.34314579,
 0.9451068 , -0.71422572, -0.4842146 ],
 [-2.09144058, -1.43393855, -0.94692412,  0.35632584, -1.20066397,
-0.42732739,  0.89828426,  1.40431048, -0.6799747 ,  0.37070243,
 0.9451068 , -0.71422572, -0.4842146 ],
 [-0.03345031,  0.69737995, -0.94692412, -0.43950337,  0.37546057,
-0.42732739, -0.99577247, -1.79706597,  1.47064295,  1.08455065,
-0.67167968,  0.2620024 ,  1.14178999],
 [ 1.69959413,  0.69737995,  1.00431346,  1.60691459,  0.43242892,
-0.42732739,  0.89828426, -1.66550255,  1.47064295,  1.70916784,
-0.67167968,  0.2620024 ,  1.14178999],
 [ 0.18318025,  0.69737995,  1.97993226, -0.66688314, -1.01076944,
-0.42732739, -0.99577247,  0.52722104, -0.6799747 ,  0.81685757,
-0.67167968, -0.71422572,  1.14178999],
 [-0.03345031,  0.69737995, -0.94692412, -1.23533257, -0.13725464,
-0.42732739,  0.89828426, -1.05153995,  1.47064295,  1.61993681,
-0.67167968,  0.2620024 ,  1.14178999],
 [ 0.18318025,  0.69737995,  0.02869467, -0.66688314, -0.11826519,
-0.42732739,  0.89828426,  0.83420234, -0.6799747 , -0.87853195,
-2.28846615, -0.71422572, -0.4842146 ],
 [ 0.50812608,  0.69737995, -0.94692412, -1.23533257, -0.13725464,
-0.42732739, -0.99577247, -0.3498684 ,  1.47064295,  0.19224037,
-0.67167968,  0.2620024 ,  1.14178999],
 [ 0.29149553,  0.69737995, -0.94692412,  0.01525618, -0.74491711,
-0.42732739,  0.89828426,  0.79034787,  1.47064295, -0.87853195,
 0.9451068 , -0.71422572,  1.14178999],
 [ 1.59127886, -1.43393855,  1.97993226,  0.47001573, -0.13725464,
-0.42732739,  0.89828426,  0.04482185, -0.6799747 ,  0.72762654,
 0.9451068 ,  1.23823052, -0.4842146 ],
 [-0.68334197,  0.69737995,  0.02869467, -0.09843371, -0.02331793,
-0.42732739, -0.99577247,  1.31660153, -0.6799747 , -0.7000699 ,
-0.67167968, -0.71422572, -0.4842146 ],
 [ 0.3998108 , -1.43393855,  1.00431346, -0.66688314,  1.78068003,
-0.42732739,  0.89828426,  0.96576576, -0.6799747 , -0.87853195,
 0.9451068 , -0.71422572, -0.4842146 ],
 [ 0.29149553, -1.43393855, -0.94692412, -0.66688314,  2.04653236,
-0.42732739,  0.89828426,  0.57107551,  1.47064295, -0.34314579,
 0.9451068 , -0.71422572, -0.4842146 ],
 [-0.68334197,  0.69737995, -0.94692412, -0.55319325, -0.46007533,
-0.42732739, -0.99577247,  1.57972836, -0.6799747 , -0.87853195,
```

```

0.9451068 , -0.71422572, -0.4842146 ],
[ 0.29149553, 0.69737995, -0.94692412, 1.8911393 , 0.81221797,
2.34012617, -0.99577247, -1.13924889, -0.6799747 , 0.01377832,
-0.67167968, 2.21445864, 1.14178999],
[ 0.83307191, 0.69737995, -0.94692412, -0.66688314, 0.39445002,
-0.42732739, 0.89828426, -2.23561068, 1.47064295, 0.72762654,
-0.67167968, 1.23823052, 1.14178999],
[ 0.61644136, 0.69737995, 1.00431346, 0.47001573, -1.16268506,
-0.42732739, -0.99577247, 0.22023974, -0.6799747 , 1.79839886,
-0.67167968, -0.71422572, -0.4842146 ],
[ 0.18318025, 0.69737995, 1.00431346, -0.09843371, 0.18556604,
2.34012617, -0.99577247, -0.3498684 , 1.47064295, -0.34314579,
-0.67167968, 0.2620024 , -2.11021919],
[ 0.50812608, 0.69737995, 1.97993226, 2.17536402, 0.79322851,
-0.42732739, -0.99577247, 0.39565762, -0.6799747 , -0.7000699 ,
-0.67167968, -0.71422572, 1.14178999],
[-0.89997253, 0.69737995, -0.94692412, -0.66688314, 0.05263988,
-0.42732739, -0.99577247, -0.26215945, -0.6799747 , -0.16468373,
0.9451068 , -0.71422572, 1.14178999],
[-0.46671142, -1.43393855, 1.00431346, -0.66688314, -0.51704369,
-0.42732739, 0.89828426, 0.35180315, -0.6799747 , 0.54916448,
-0.67167968, -0.71422572, -0.4842146 ],
[ 0.72475664, 0.69737995, -0.94692412, -0.66688314, 0.26152385,
-0.42732739, 0.89828426, -0.43757734, 1.47064295, 2.33378503,
-0.67167968, 0.2620024 , 1.14178999],
[-1.22491836, 0.69737995, 1.00431346, -0.09843371, 1.30594372,
-0.42732739, 0.89828426, 0.52722104, -0.6799747 , 0.81685757,
0.9451068 , 0.2620024 , -0.4842146 ],
[ 0.72475664, 0.69737995, -0.94692412, 0.35632584, -1.52348465,
-0.42732739, -0.99577247, -1.09539442, 1.47064295, 2.33378503,
-0.67167968, 0.2620024 , -0.4842146 ],
[ 0.3998108 , 0.69737995, 0.02869467, -0.38265842, -0.49805423,
-0.42732739, 0.89828426, -0.26215945, -0.6799747 , -0.52160784,
-0.67167968, 3.19068676, 1.14178999],
[-0.03345031, 0.69737995, -0.94692412, -0.55319325, 0.75524961,
-0.42732739, -0.99577247, -1.49008466, 1.47064295, 1.97686092,
-0.67167968, 1.23823052, -0.4842146 ]])

```

```

In [23]: from sklearn.linear_model import LogisticRegressionCV
log_model=LogisticRegressionCV()
log_model.fit(scaler_X_train,y_train)

```

```

Out[23]: ▼ LogisticRegressionCV ⓘ ?
LogisticRegressionCV()

```

```

In [24]: log_model.C_

```

```

Out[24]: array([0.04641589])

```

```

In [25]: log_model.get_params()

```

```
Out[25]: {'Cs': 10,
          'class_weight': None,
          'cv': None,
          'dual': False,
          'fit_intercept': True,
          'intercept_scaling': 1.0,
          'l1_ratios': None,
          'max_iter': 100,
          'multi_class': 'deprecated',
          'n_jobs': None,
          'penalty': 'l2',
          'random_state': None,
          'refit': True,
          'scoring': None,
          'solver': 'lbfgs',
          'tol': 0.0001,
          'verbose': 0}
```

```
In [26]: log_model.coef_
```

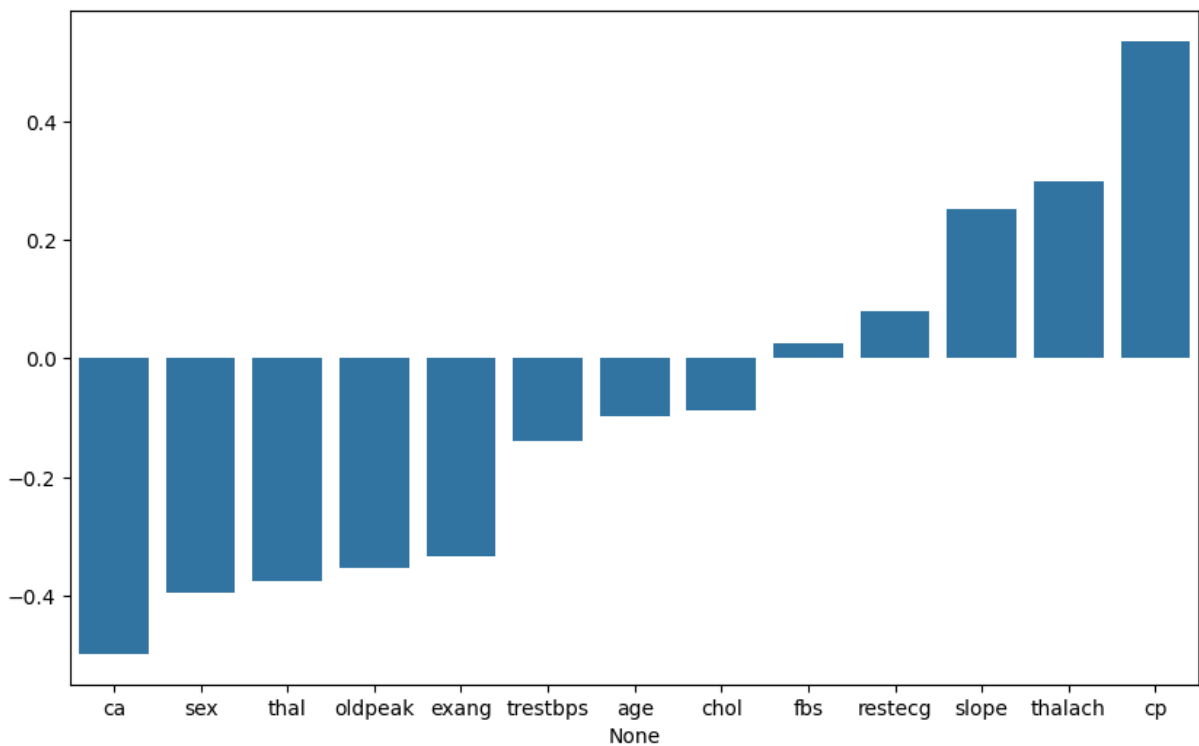
```
Out[26]: array([[ -0.09624234, -0.39455733,  0.53541263, -0.13845013, -0.0882132 ,
                   0.02495565,  0.08083019,  0.29896895, -0.33440044, -0.35252781,
                   0.25100118, -0.49732614, -0.37440968]])
```

```
In [27]: coefs=pd.Series(index=X.columns,data=log_model.coef_[0])
```

```
In [28]: coefs=coefs.sort_values()
```

```
In [31]: plt.figure(figsize=(10,6))
         sns.barplot(x=coefs.index, y=coefs.values)
```

```
Out[31]: <Axes: xlabel='None'>
```

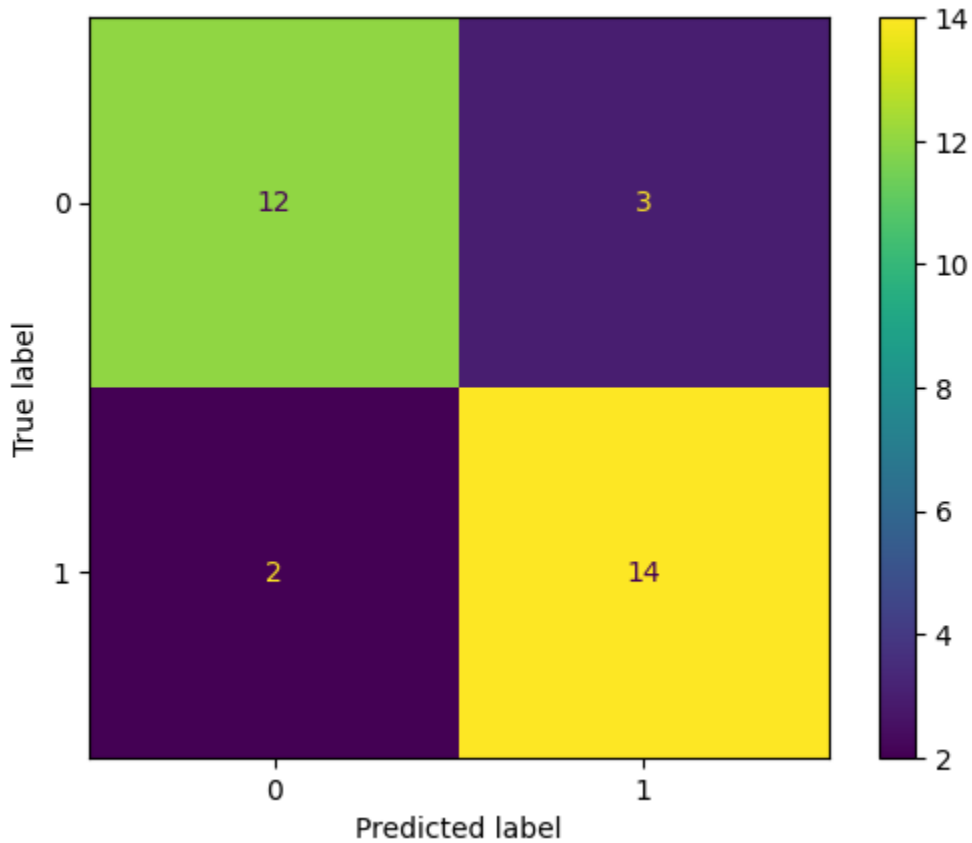


```
In [32]: from sklearn.metrics import confusion_matrix, classification_report, ConfusionMatrixDisplay
y_pred=log_model.predict(scaled_X_test)
cm=confusion_matrix(y_test,y_pred)
cm
```

```
Out[32]: array([[12,  3],
               [ 2, 14]])
```

```
In [34]: dsip=ConfusionMatrixDisplay(confusion_matrix=cm)
dsip.plot()
```

```
Out[34]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1ca0c6ba5a0>
```



```
In [35]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.86	0.80	0.83	15
1	0.82	0.88	0.85	16
accuracy			0.84	31
macro avg	0.84	0.84	0.84	31
weighted avg	0.84	0.84	0.84	31

```
In [ ]:
```