Logistic Regression

Logistic Regression was used in the biological sciences in early twentieth century.
It was then used in many social science applications.
Logistic Regression is used when the dependent variable(target) is categorical.
For example,
To predict whether an email is spam (1) or (0)
Whether the tumor is malignant (1) or not (0)

Types of Logistic Regression

1. Binary Logistic Regression
The categorical response has only two 2 possible outcomes. Example: Spam or Not

2. Multinomial Logistic Regression
Three or more categories without ordering. Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan)

3. Ordinal Logistic Regression
Three or more categories with ordering. Example: Movie rating from 1 to 5

In this notebook we are talking only about **Binary Logistic Regression.**

In [1]:

```
%load_ext nb_black
```

In [2]:

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

In [3]:

```
# %matplotlib notebook
```

Heart Disease Dataset :

Attribute Information:

1. age
2. sex
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholestoral in mg/dl
6. fasting blood sugar > 120 mg/dl

7. resting electrocardiographic results (values 0,1,2),
8. maximum heart rate achieved
9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

[Heart Disease Dataset Link (https://www.kaggle.com/ronitf/heart-disease-uci)](https://www.kaggle.com/ronitf/heart-disease-uci)

S.No	Attribute	Value	Description
1	age	29 – 62	age in years
2	sex	0 – male, 1- female	gender
3	cp	1-typical angina; 2-atypical angina 3-non-anginal pain; 4-asymptomatic	chest pain type
4	trestbps	Numeric value(140mm/Hg)	resting blood pressure in mm/Hg
5	chol	Numeric value(289mg/dl)	serum cholesterol in mg/dl
6	fbs	1-true, 0-false	fasting blood pressure>120mg/dl
7	restecg	0-normal, 1-having ST-T, 2-hypertrophy	resting electrocardiographic results
8	thalach	140,173	maximum heart rate achieved
9	exang	1-yes, 0-no	exercise induced angina
10	oldpeak	Numeric value	ST depression induced by exercise relative to rest
11	slope	1-upsloping, 2-flat, 3-downsloping	the slope of the peak exercise ST segment
12	ca	0-3 vessels	number of major vessels colored by flourosopy
13	thal	3-normal, 6-fixed defect, 7-reversable defect	thalassemia
14	num	0: < 50% diameter narrowing 1: > 50% diameter narrowing	diagnosis of heart disease (angiographic disease status)

Read Full Paper [Here (https://www.ijrte.org/wp-content/uploads/papers/v8i2S3/B11630782S319.pdf)](https://www.ijrte.org/wp-content/uploads/papers/v8i2S3/B11630782S319.pdf)

Why you want to apply classification on selected dataset? Discuss full story behind dataset.

In this dataset we have some parameters like age , sex , chest pain type and etc. and finally we have a target which is tell us weather a person has a heart disease or not.From all this parameters we have to predict weather person has a heart disease or not i.e we have to predict 0 (No heart Disease) or 1 (Yes heart Disease). So it is a classification problem we have to classify the categoris and that's why we apply classification algorithms on this dataset.

Generally for two categories we called it **Binary classification.**
For more than two categories we called it **Multiclass classification**

In [6]:

```
dataset = pd.read_csv("heart.csv")
```

In [7]:

```
dataset.head()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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

In [8]:

```
dataset.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
```

How many total observations in data?

There are total 303 non-null values in dataset.

How many independent variables?

There are 12 independent variable in this dataset.

- age
- sex
- cp
- trestbps
- chol

- fbs
- restecg
- thalach
- exang
- oldpeak
- slope
- ca
- thal

Which is dependent variable?

' target ' is dependent variable and that we have to predict.

Heart Disease Dataset :

Attribute Information:

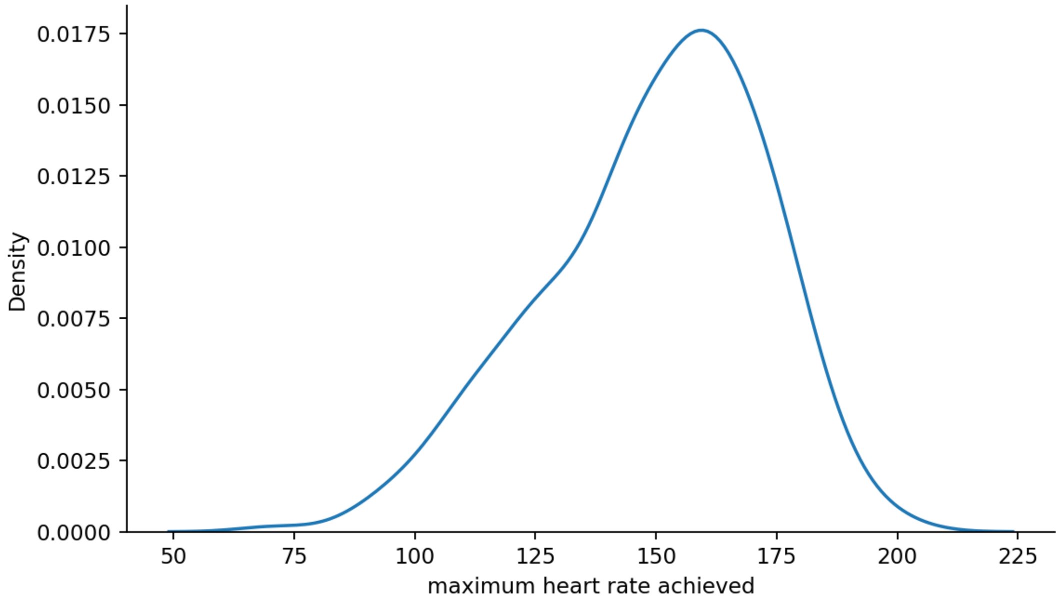
1. age
2. sex
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholestoral in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2),
8. maximum heart rate achieved
9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

In [4]:

```
import seaborn as sns
```

In [186]:

```
sns.displot(dataset["thalach"], kind="kde")
plt.xlabel("maximum heart rate achieved")
```



Out[186]:

```
Text(0.5, 9.444444444444438, 'maximum heart rate achieved')
```

In [9]:

```
dataset["thal"].value_counts()
```

Out[9]:

```
2    166
3    117
1     18
0      2
Name: thal, dtype: int64
```

Categorical Values

sex
cp
exang
restecg
slope
ca
thal

In [10]:

```
dataset.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 [11]:

```
dataset.head(10)
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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 [12]:

```
categorized_sex = pd.get_dummies(dataset["sex"])  # caterogies encoder
```

In [13]:

```
categorized_sex.columns = ["female", "male"]
```

In [14]:

```
categorized_data = pd.get_dummies(
    data=dataset,
    columns=["sex", "cp", "fbs", "restecg", "exang", "slope", "ca", "thal", "target"],
    drop_first=True,
)
```

Model prepration

for this time we take all the variables and check accuracy. And We make this model as the base model

In [15]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

In [16]:

```
sc = StandardScaler()
targets = categorized_data["target_1"]
```

In [17]:

```
X = categorized_data.drop(columns=["target_1"])
```

In [18]:

```
new_X = sc.fit_transform(X)
```

In [19]:

```
X_train, X_test, y_train, y_test = train_test_split(
    new_X, targets, test_size=0.2, random_state=0
)
```

In [20]:

```
model = LogisticRegression()
```

In [21]:

```
model.fit(X_train, y_train)
```

Out[21]:

```
LogisticRegression()
```

In [22]:

```
test_predictions = model.predict(X_test)
```

In [23]:

```
print(classification_report(y_test, test_predictions))
```

```
              precision    recall  f1-score   support

           0       0.88      0.85      0.87        27
           1       0.89      0.91      0.90        34

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
weighted avg       0.89      0.89      0.88        61
```

In [24]:

```
print(classification_report(y_train, model.predict(X_train)))
```

```
              precision    recall  f1-score   support

           0       0.88      0.82      0.85       111
           1       0.86      0.91      0.88       131

    accuracy                           0.87       242
   macro avg       0.87      0.86      0.87       242
weighted avg       0.87      0.87      0.87       242
```

In [25]:

```
model.score(X_test, y_test)
```

Out[25]:

```
0.8852459016393442
```

In [26]:

```
model.score(X_train, y_train)
```

Out[26]:

```
0.8677685950413223
```

Test Accuracy : 88%

Train Accuracy : 86%

Corelation

In [27]:

```
dataset.corr()["target"]
```

Out[27]:

```
age        -0.225439
sex        -0.280937
cp          0.433798
trestbps   -0.144931
chol       -0.085239
fbs        -0.028046
restecg     0.137230
thalach     0.421741
exang      -0.436757
oldpeak    -0.430696
slope       0.345877
ca         -0.391724
thal       -0.344029
target      1.000000
Name: target, dtype: float64
```

We take cp , restecg , thalach , slope variables in count

In [28]:

```
categorized_data.corr()["target_1"]
```

Out[28]:

```
age          -0.225439
trestbps     -0.144931
chol         -0.085239
thalach       0.421741
oldpeak      -0.430696
sex_1        -0.280937
cp_1          0.245879
cp_2          0.316742
cp_3          0.086957
fbs_1        -0.028046
restecg_1     0.175322
restecg_2    -0.068410
exang_1      -0.436757
slope_1      -0.362053
slope_2       0.394066
ca_1         -0.232412
ca_2         -0.273998
ca_3         -0.210615
ca_4          0.066441
thal_1       -0.106589
thal_2        0.527334
thal_3       -0.486112
target_1      1.000000
Name: target_1, dtype: float64
```

Modal Improvement I

First time we take all the variables and check how our model performs.
As you can see above most of the variables are negative corelated and that thing is bad for model.
So for this time we include most of the positive corelated variables and check on that data how our model performes.

Highly Corelated Variables

thalach cp restecg slope thal

In [29]:

```
X = categorized_data.drop(columns=["target_1"])
```

In [30]:

```
X = X[["thalach", "cp_1", "cp_2", "cp_3", "restecg_1", "slope_2", "thal_2"]]
```

In [31]:

```
new_x = sc.fit_transform(X)
```

In [32]:

```
X_train, X_test, y_train, y_test = train_test_split(
    new_x, targets, test_size=0.2, random_state=2
)
```

In [33]:

```
model2 = LogisticRegression()
```

In [34]:

```
model2.fit(X_train, y_train)
```

Out[34]:

```
LogisticRegression()
```

In [35]:

```
model2.score(X_test, y_test)
```

Out[35]:

```
0.8688524590163934
```

In [36]:

```
model2.score(X_train, y_train)
```

Out[36]:

```
0.7768595041322314
```

Test Accuracy : 86%

Train Accuracy : 77%

In [37]:

```
test_predictions = model2.predict(X_test)
```

In [38]:

```
print(classification_report(y_test, test_predictions))
```

```
              precision    recall  f1-score   support

           0       0.93      0.81      0.87        32
           1       0.82      0.93      0.87        29

    accuracy                           0.87        61
   macro avg       0.87      0.87      0.87        61
weighted avg       0.88      0.87      0.87        61
```

In [39]:

```
print(classification_report(y_train, model2.predict(X_train)))
```

```
              precision    recall  f1-score   support

           0       0.74      0.76      0.75       106
           1       0.81      0.79      0.80       136

    accuracy                           0.78       242
   macro avg       0.77      0.78      0.77       242
weighted avg       0.78      0.78      0.78       242
```

Now we can say our model1 is best..!!

Becuase model1 has accuracy of 88% and model2 has accuracy of 86%

model1 has train accuracy is 87% which near to its test accuracy on otherside model2 has train accuracy 77% which is very far from its test accuracy.

That means our model1 can perform very good on train data as well as test data.

Cross Validation

In [40]:

```
from sklearn.model_selection import cross_val_score
```

In [41]:

```
scores_model1 = cross_val_score(model, new_X, targets, cv=6, n_jobs=-1)
scores_model2 = cross_val_score(model2, new_x, targets, cv=6, n_jobs=-1)
```

In [42]:

```
scores_model1.mean()
```

Out[42]:

```
0.8611764705882353
```

In [43]:

```
scores_model2.mean()
```

Out[43]:

```
0.8016993464052287
```

Now, We can prove that model1 is best by using K-Fold Cross Validation.

In [44]:

```
dataset.head(10)
```

Out[44]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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

1. age
2. sex
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholestoral in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2),
8. maximum heart rate achieved

9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

In [106]:

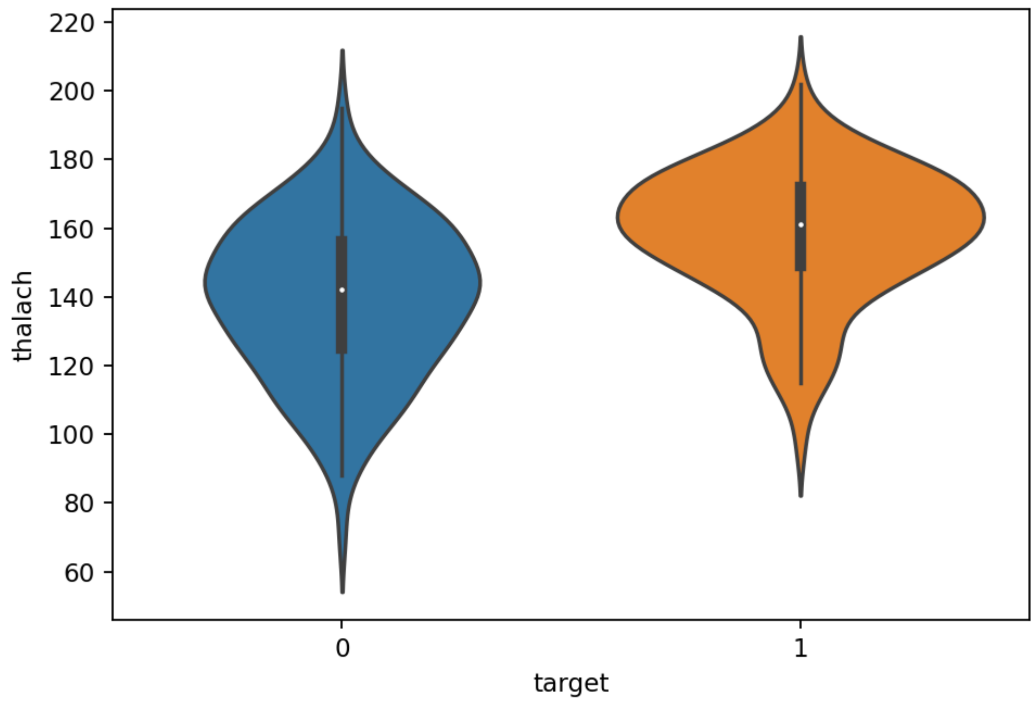
```
%matplotlib notebook
```

Model Improvment II

now we do analysis on all the variables and after graphical representation we select the variable and then make a model and then check K-Fold Cross Validation Score.

In [105]:

```
sns.violinplot(x="target", y="thalach", data=dataset)  # 0 female and 1 male
```



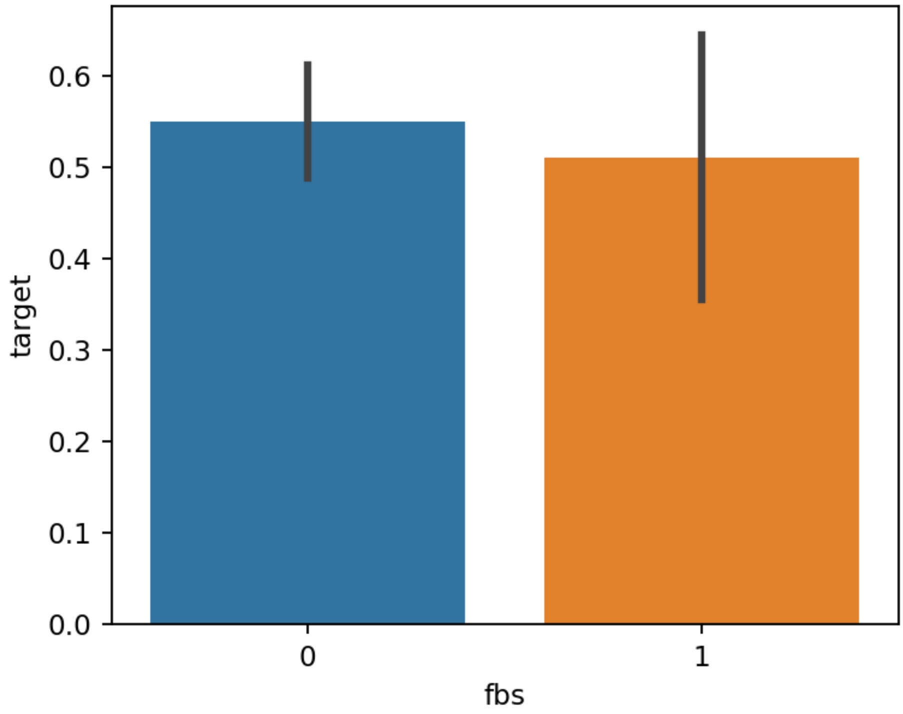
Out[105]:

```
<AxesSubplot:xlabel='target', ylabel='thalach'>
```

Observation:
As you can see a person with high heart rate have a little chance of heart disease. In the heart disease positive you can see very high desntiy at mean value.A person with aproximate heart rate of 160 have e very good chance of heart disease.That's why include this field in model prepration.

In [117]:

```
sns.barplot(x="fbs", y="target", data=dataset)  # 0 female and 1 male
```



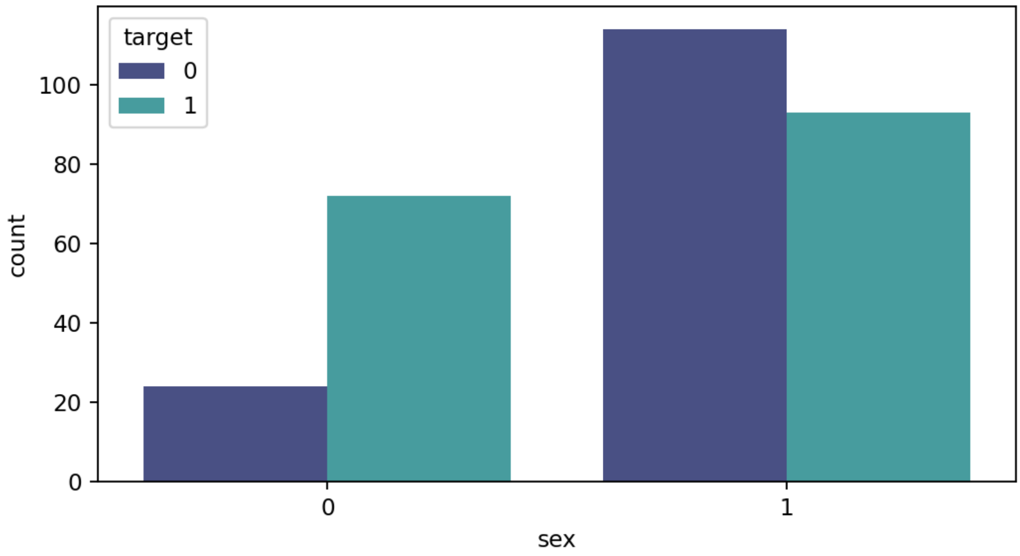
Out[117]:

```
<AxesSubplot:xlabel='fbs', ylabel='target'>
```

Observation:
For both cases there is not a large difference fbs (fasting blood pressure). So this field is not that much userful. so we are not including this field.

In [118]:

```
sns.countplot(x="sex", hue="target", data=dataset, palette="mako")
```



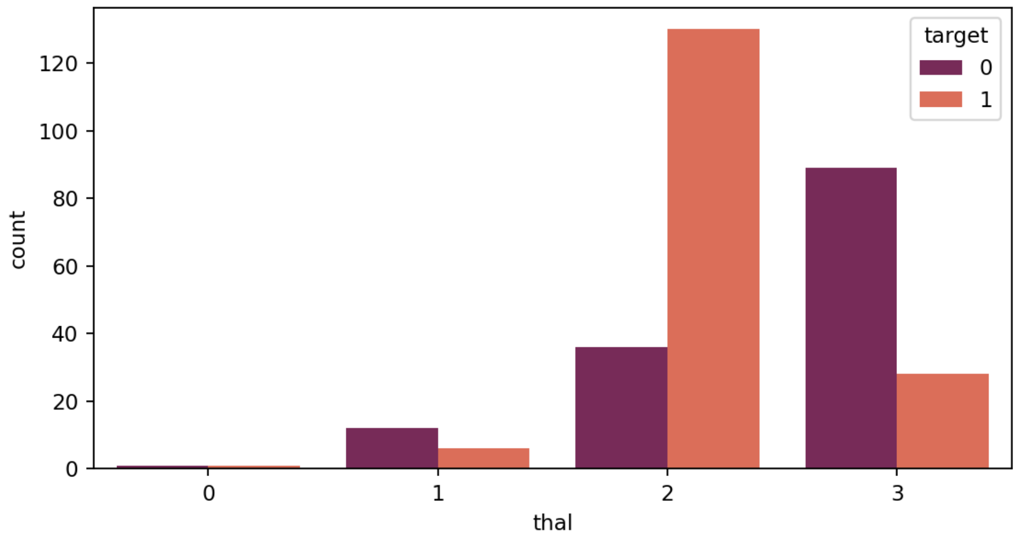
Out[118]:

<AxesSubplot:xlabel='sex', ylabel='count'>

Observation:
In this plot we can see women have low chance of heart disease and men have high chance of high disease.But women with heart disease bar also near to men with heart disease.so we are including that field in model prepration

In [119]:

```
sns.countplot(x="thal", hue="target", data=dataset, palette="rocket")
```



Out[119]:

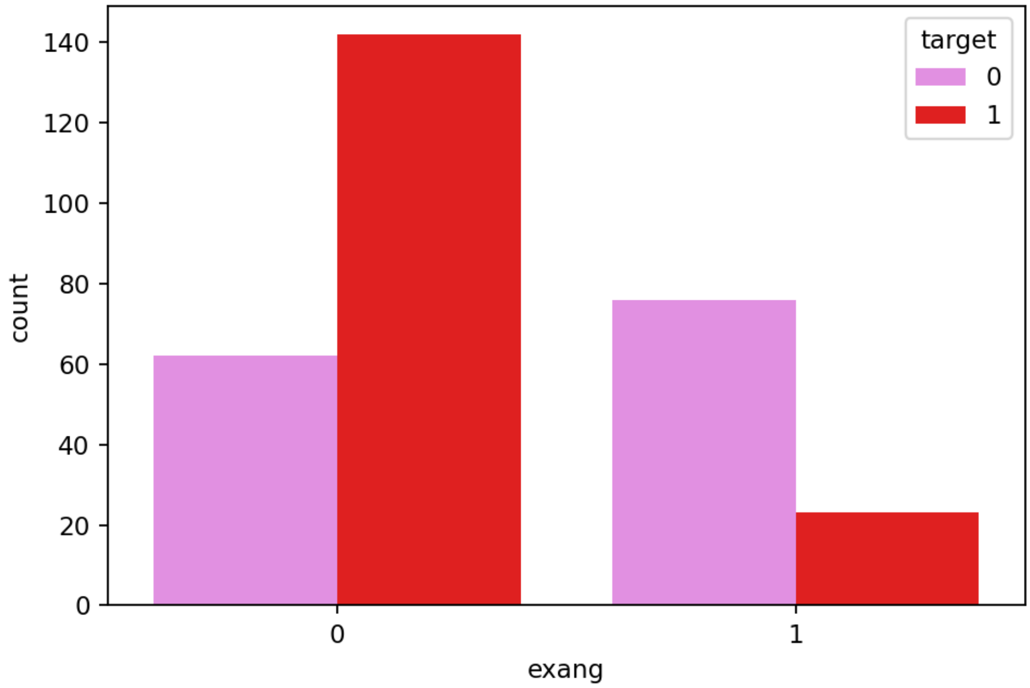
<AxesSubplot:xlabel='thal', ylabel='count'>

Observation:

As we can see people with thal type-2 have very high chance of heart disease and people with thal type-3 have very good chance not have a heart disease. So we use this field in the model prepration.

In [120]:

```
sns.countplot(x="exang", hue="target", data=dataset, palette=["violet", "red"])
```



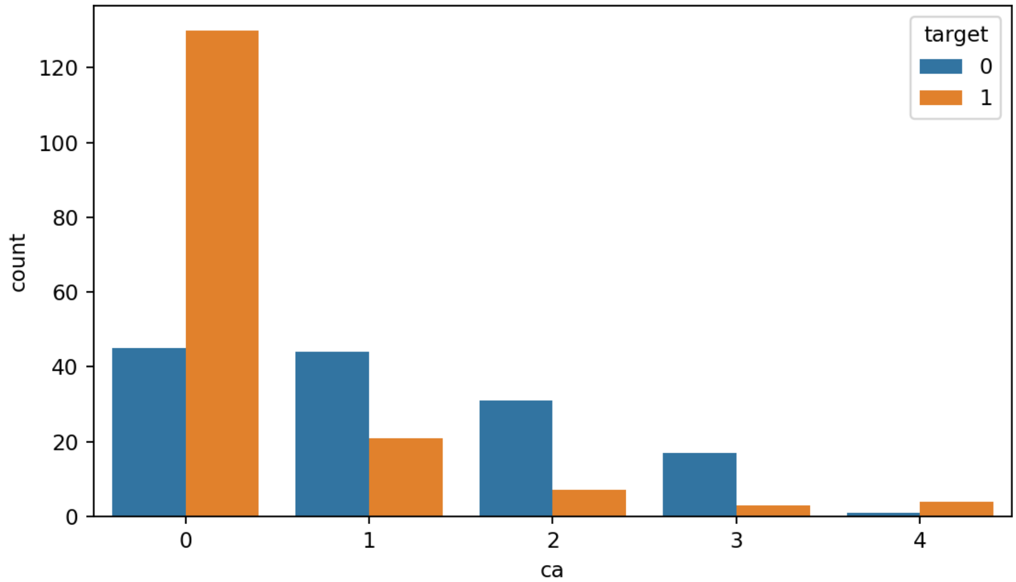
Out[120]:

```
<AxesSubplot:xlabel='exang', ylabel='count'>
```

Observation:
As we can see people who don't excersice have a very good chance of heart disease and people who do excersice have ver less chance of heart disease.

In [121]:

```
sns.countplot(x="ca", hue="target", data=dataset)
```



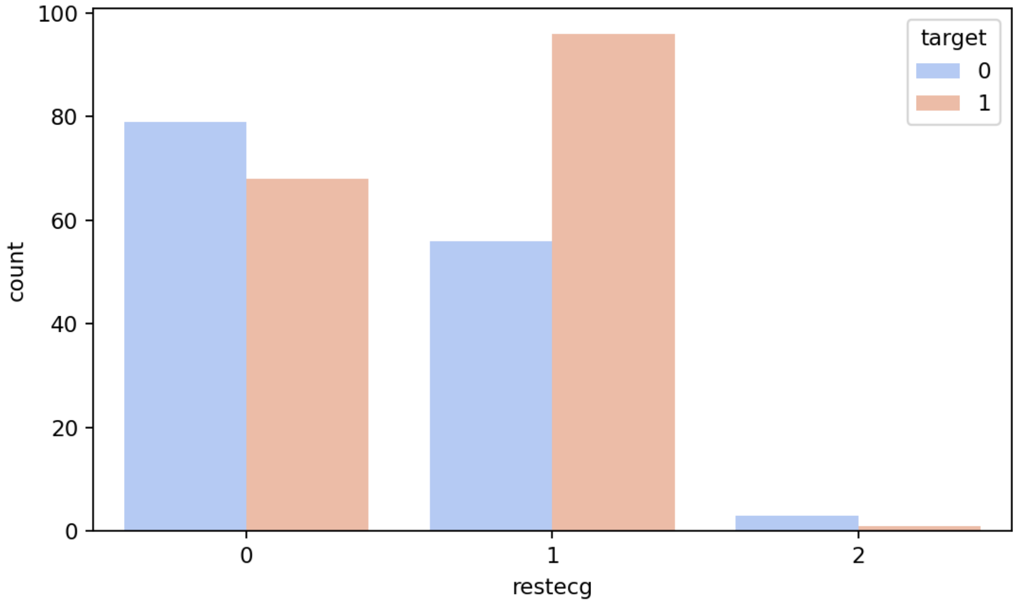
Out[121]:

```
<AxesSubplot:xlabel='ca', ylabel='count'>
```

Observation:
people with ca type-0 have a very good chance of having a heart disease and ca type-3 have less chance to have a heart disease. So that why we include only ca type-0 in model prepration.

In [122]:

```
sns.countplot(x="restecg", hue="target", data=dataset, palette="coolwarm")
```



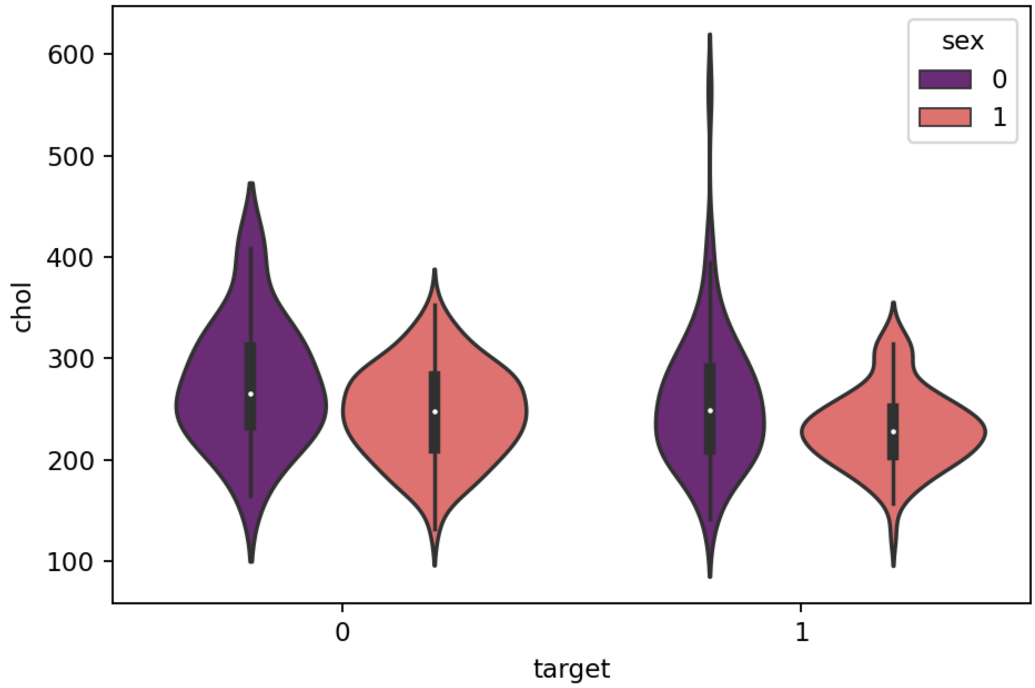
Out[122]:

```
<AxesSubplot:xlabel='restecg', ylabel='count'>
```

Observation:
People with ST-T (type - 1) as electrocariographic result have a good chance to having a heart disease. People with hypertrophy (type-2) as electrocardiographic result have not that much enough data. so that we can't predict.It has a good corlation with target so that we include this field.

In [125]:

```
sns.violinplot(x="target", y="chol", hue="sex", data=dataset, palette="magma")
```



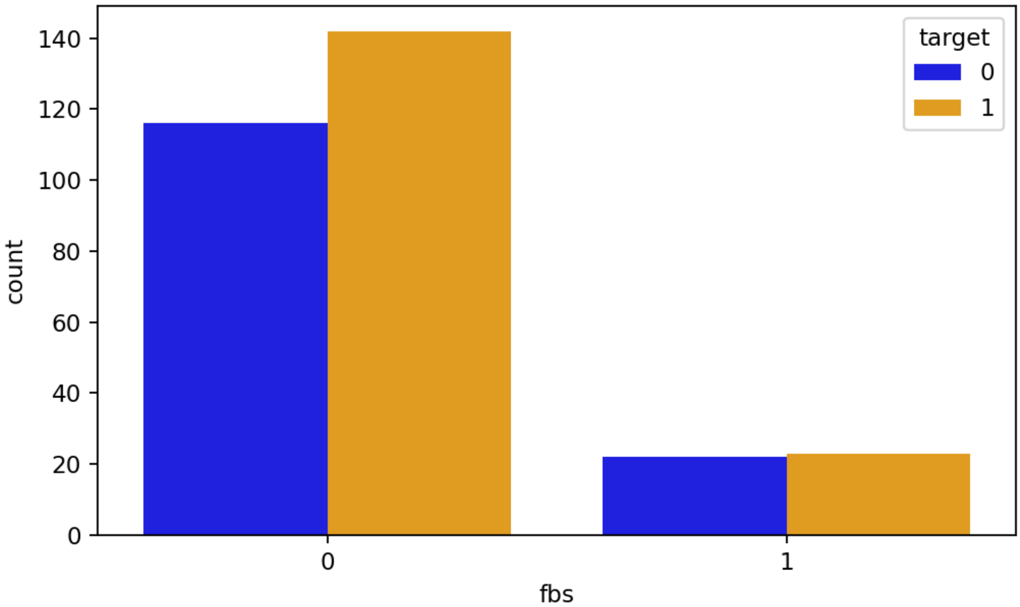
Out[125]:

```
<AxesSubplot:xlabel='target', ylabel='chol'>
```

Observation:
we can see for all the results we have near equal mean. we can not classify the data by using cholestrol field. It is also negative correlated with target so that's why we are not include this field.

In [124]:

```
sns.countplot(x="fbs", hue="target", data=dataset, palette=["blue", "orange"])
```



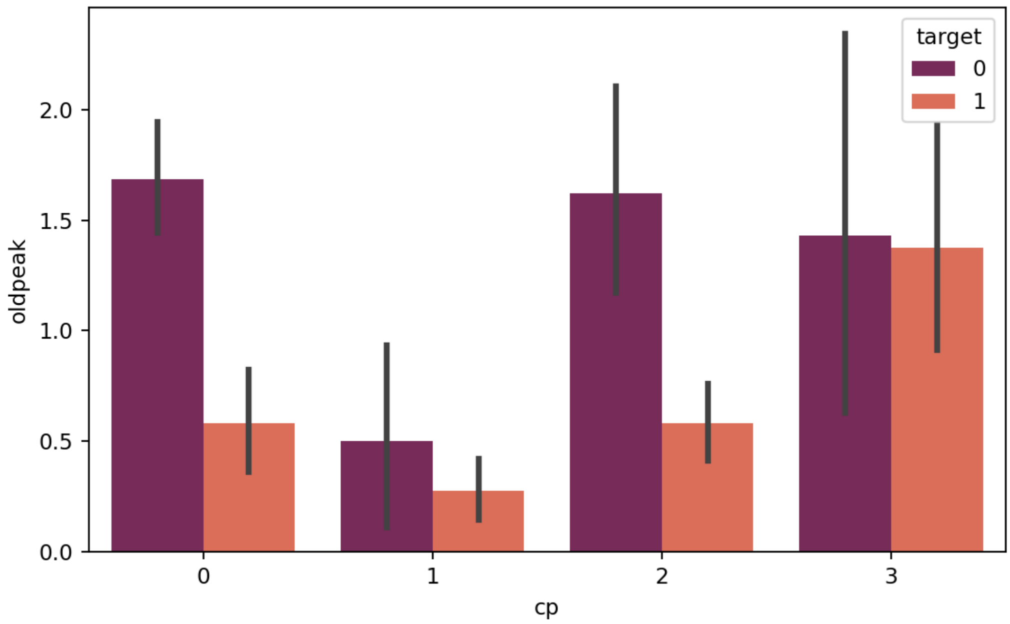
Out[124]:

```
<AxesSubplot:xlabel='fbs', ylabel='count'>
```

Observation:
Surprisingly...!!!
Person with blood presure < 120 have a high chance of high diseases. For person with high blood presure we can't say anything. So it is unpredicteble we are not include this field.

In [128]:

```
sns.barplot(x="cp", y="oldpeak", hue="target", data=dataset, palette="rocket")
# plt.legend()
```



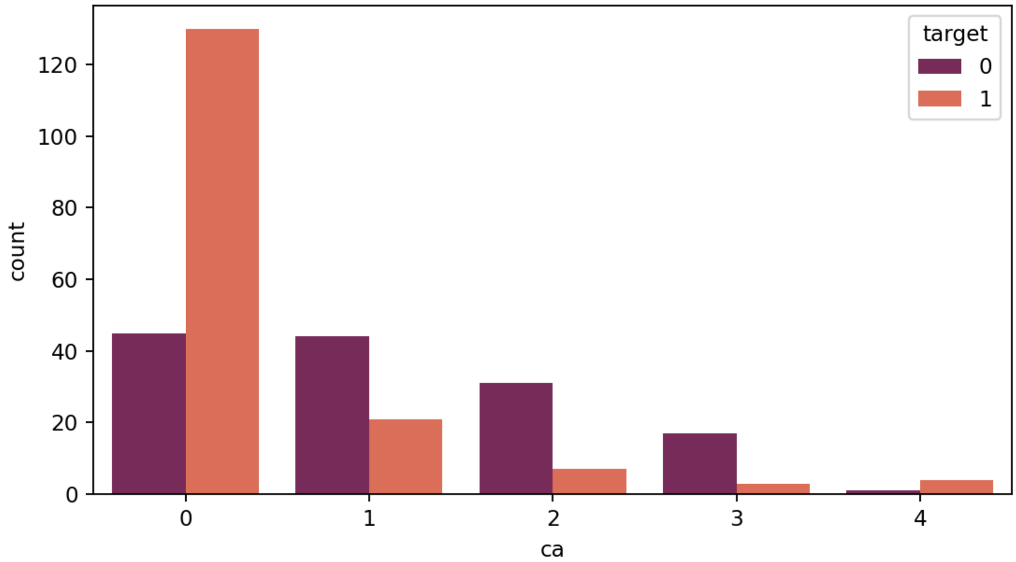
Out[128]:

```
<AxesSubplot:xlabel='cp', ylabel='oldpeak'>
```

Observation:
person with chest pain type-0 and type-2 and high oldpeak have a very good chance of having heart disease. for type-3 and type-2 we can say neutral. We include both field in model prepration.

In [132]:

```
sns.countplot(x="ca", hue="target", data=dataset, palette="rocket")
```



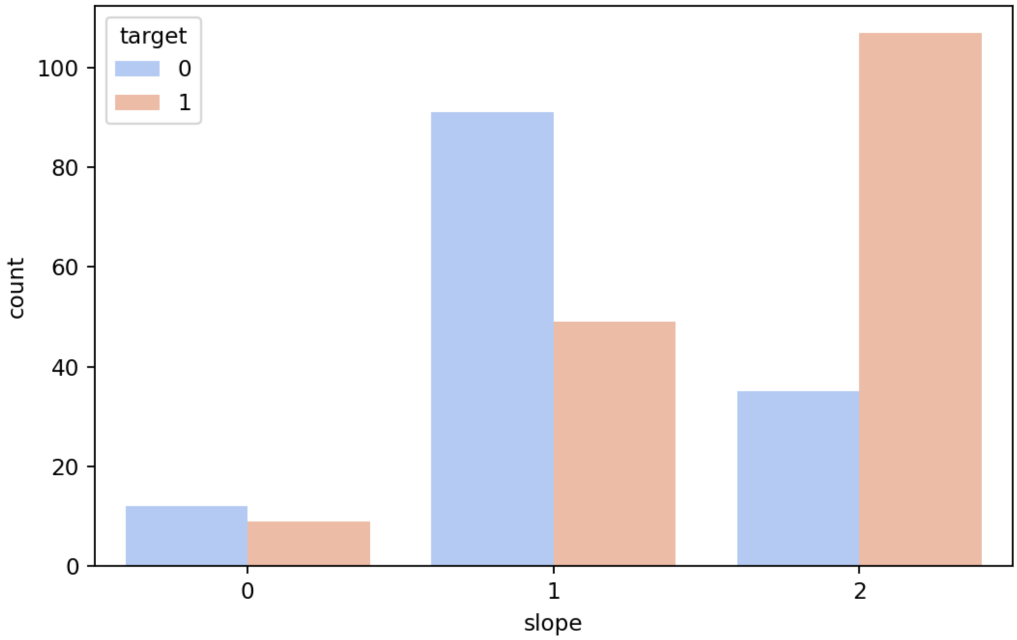
Out[132]:

```
<AxesSubplot:xlabel='ca', ylabel='count'>
```

Observation:
We can see that ca type-0 have a very good chance of having heart disease then other types. we include this field and check how it performs on model.

In [134]:

```
sns.countplot(x="slope", hue="target", data=dataset, palette="coolwarm")
```



Out[134]:

```
<AxesSubplot:xlabel='slope', ylabel='count'>
```

Observation:
We can clearly see that person with slop type-2 (downsloping) have a very good chance of having heart diseas and person with slope type-1 (flat) have less chance of having heart disease. we use this field in model preparation.

Include variables

- thalach
- ca
- oldpeak
- cp
- restecg
- ca
- exang
- thal
- trestbps
- slope
- age
- sex

Not Included variables

- fbs
- chol

In [45]:

```
categorized_data.head()
```

Out[45]:

	age	trestbps	chol	thalach	oldpeak	sex_1	cp_1	cp_2	cp_3	fbs_1	...	slope_1	slope_
0	63	145	233	150	2.3	1	0	0	1	1	...	0	
1	37	130	250	187	3.5	1	0	1	0	0	...	0	
2	41	130	204	172	1.4	0	1	0	0	0	...	0	
3	56	120	236	178	0.8	1	1	0	0	0	...	0	
4	57	120	354	163	0.6	0	0	0	0	0	...	0	

5 rows × 23 columns

In [78]:

```
X = categorized_data.drop(columns=["chol", "fbs_1", "target_1"])
```

In [79]:

```
Y = categorized_data["target_1"]
```

In [80]:

```
new_X1 = sc.fit_transform(X)
```

In [81]:

```
X_train, X_test, y_train, y_test = train_test_split(
    new_X1, Y, test_size=0.2, random_state=0
)
```

In [82]:

```
model3 = LogisticRegression()
```

In [83]:

```
model3.fit(X_train, y_train)
# model3_b.fit(X_train, y_train)
```

Out[83]:

```
LogisticRegression()
```

In [84]:

```
model3.score(X_test, y_test)
```

Out[84]:

```
0.8524590163934426
```

In [85]:

```
model3.score(X_train, y_train)
```

Out[85]:

```
0.8760330578512396
```

Test Accuracy : 85.24%

Train Accuracy : 85.95%

In [86]:

```
print(classification_report(y_test, model3.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84        27
           1       0.88      0.85      0.87        34

    accuracy                           0.85        61
   macro avg       0.85      0.85      0.85        61
weighted avg       0.85      0.85      0.85        61
```

In [87]:

```
print(classification_report(y_train, model3.predict(X_train)))
```

```
              precision    recall  f1-score   support

           0       0.89      0.84      0.86       111
           1       0.87      0.91      0.89       131

    accuracy                           0.88       242
   macro avg       0.88      0.87      0.87       242
weighted avg       0.88      0.88      0.88       242
```

In [88]:

```
scores_model3 = cross_val_score(model3, new_X1, Y, cv=6, n_jobs=-1)
```

In [89]:

```
scores_model1.mean(), scores_model2.mean(), scores_model3.mean()
```

Out[89]:

```
(0.8611764705882353, 0.8016993464052287, 0.8643137254901961)
```

Conclusion :

Model 3 performs slight better then Model 1.
It gives 86.43 % accuracy by using Cross Validation while Model 1 Gives 86.11% accuracy.
Model 2 Perform worst in all three models because it is not include negative corelated features and only include positive corelated features. That's why Model2 not perform well.

In [ ]:

```

```