HEART DISEASE PREDICTION

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Submitted By:
Vishalakshi
102017189
Twesha Arvind
102017195

Submitted To:
Ms. Kudratdeep Aulakh

July 2022 – December 2022

HEART DISEASE PREDICTION

Python Libraries

- Import pandas:

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

- Import NumPy:

NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning.

- Import Matplotlib.pyplot:

Matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

- Import seaborn:

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Overview of the Project

In this project, we worked on heart disease prediction and for that, we looked into the heart disease dataset.

From that dataset we derived various insights that helped us know about the weightage of each feature and how they are interrelated to each other.

But our sole aim was to detect the probability of a person that will be affected by a savior heart problem or not.

So we used KNN(K-Nearest Neighbour) machine learning algorithm to train our model and predict our results.

This model gives us an accuracy of 84%.

Code and Output

```
[ ] #Importing Necessary Libraries
```

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

```
[35] #Metrics for Classification technique
```

```
from sklearn.metrics import accuracy_score
```

```
[37] #Scaler
```

```
[38] from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
```

```
[39] # Model building
```

```
[40] from sklearn.neighbors import KNeighborsClassifier
```

```
[19] #Importing Data
```

```
data = pd.read_csv("heart.csv")
data.head(6)
```

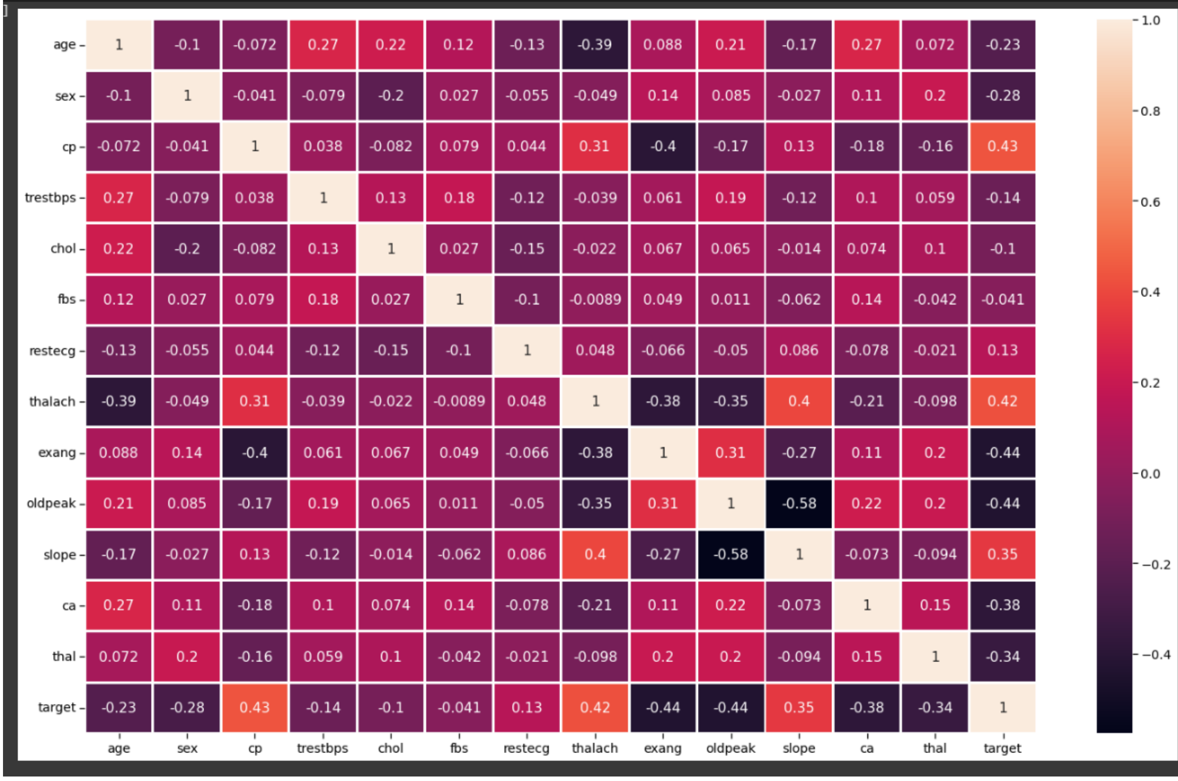
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	1

```
[21] data.shape
     data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512	1.385366	0.754146	2.323902	0.513171
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.030798	0.620660	0.500070
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

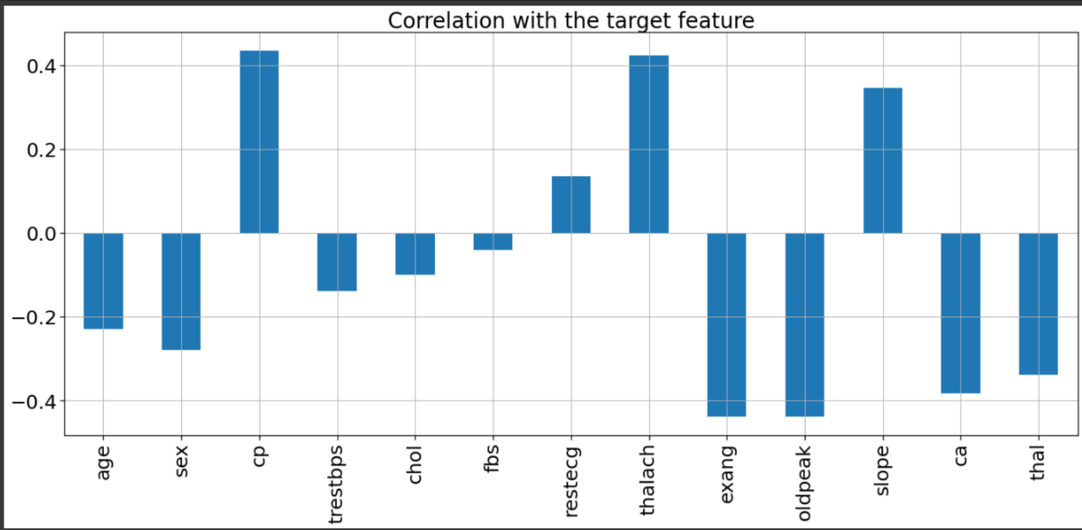
```
#correlation between features
```

```
[ ] plt.figure(figsize=(20,12))
    sns.set_context('notebook',font_scale = 1.3)
    sns.heatmap(data.corr(),annot=True,linewidth =2)
    plt.tight_layout()
```



```
[24] #correlation of target variable

[25] sns.set_context('notebook',font_scale = 2.3)
     data.drop('target',axis=1).corrwith(data.target).plot(kind='bar', grid=True, figsize=(20, 10),title="Correlation with the target feature")
     plt.tight_layout()
```

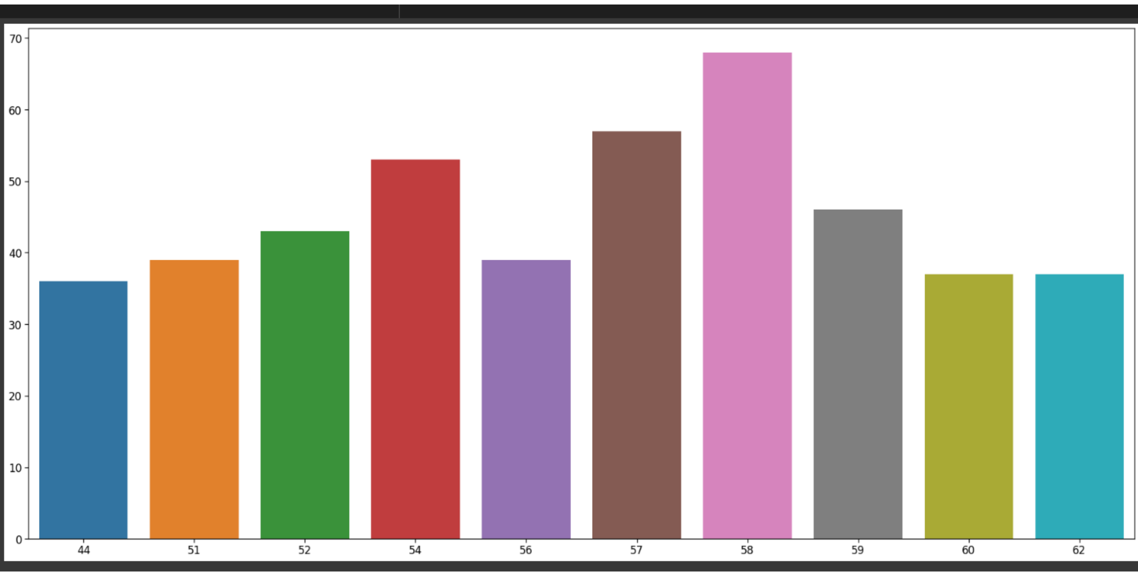


Inference: Insights from the above graph are:

- Four features (“cp”, “restecg”, “thalach”, “slope”) are positively correlated with the target feature.
- Other features are negatively correlated with the target feature.

```
[26] #Age(“age”) Analysis

[27] plt.figure(figsize=(25,12))
     sns.set_context('notebook',font_scale = 1.5)
     sns.barplot(x=data.age.value_counts()[:10].index,y=data.age.value_counts()[:10].values)
     plt.tight_layout()
```



Inference: Here we can see that the 58 age column has the highest frequency.

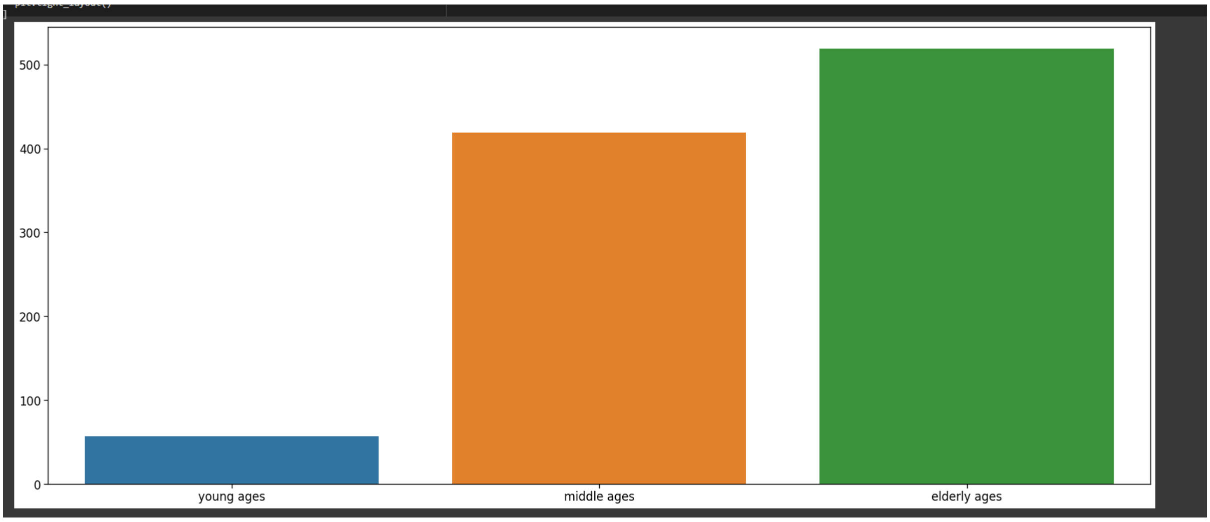
```
[28] #checking range
     minAge=min(data.age)
     maxAge=max(data.age)
     meanAge=data.age.mean()
     print('Min Age :',minAge)
     print('Max Age :',maxAge)
     print('Mean Age :',meanAge)

     Min Age : 29
     Max Age : 77
     Mean Age : 54.43414634146342
```

```
[29] #divide the Age feature into three parts - “Young”, “Middle” and “Elder”
```

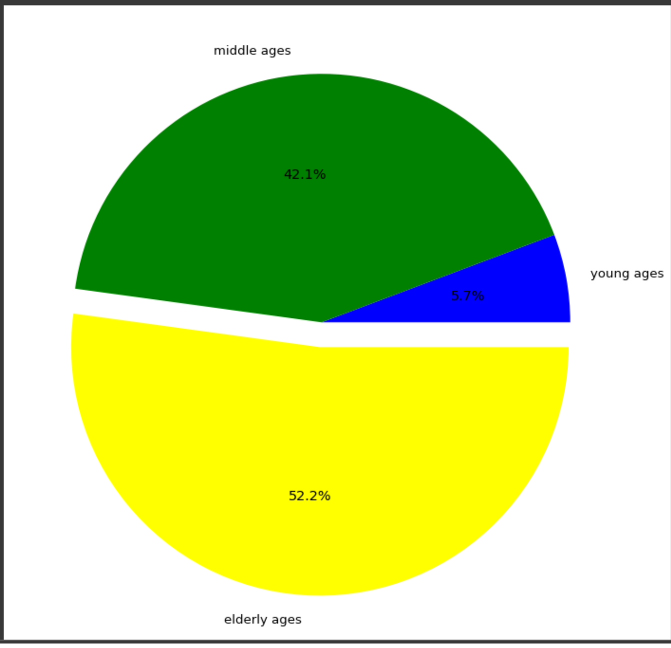
```
[30] Young = data[(data.age>=29)&(data.age<40)]
     Middle = data[(data.age>=40)&(data.age<55)]
     Elder = data[(data.age>55)]

     plt.figure(figsize=(23,10))
     sns.set_context('notebook',font_scale = 1.5)
     sns.barplot(x=['young ages','middle ages','elderly ages'],y=[len(Young),len(Middle),len(Elder)])
     plt.tight_layout()
```



Inference: Here we can see that elder people are the most affected by heart disease and young ones are the least affected.

```
[31] colors = ['blue','green','yellow']
     explode = [0,0,0.1]
     plt.figure(figsize=(10,10))
     sns.set_context('notebook',font_scale = 1.2)
     plt.pie([len(Young),len(Middle),len(Elder)],labels=['young ages','middle ages','elderly ages'],explode=explode,colors=colors, autopct='%1.1f%%')
     plt.tight_layout()
```

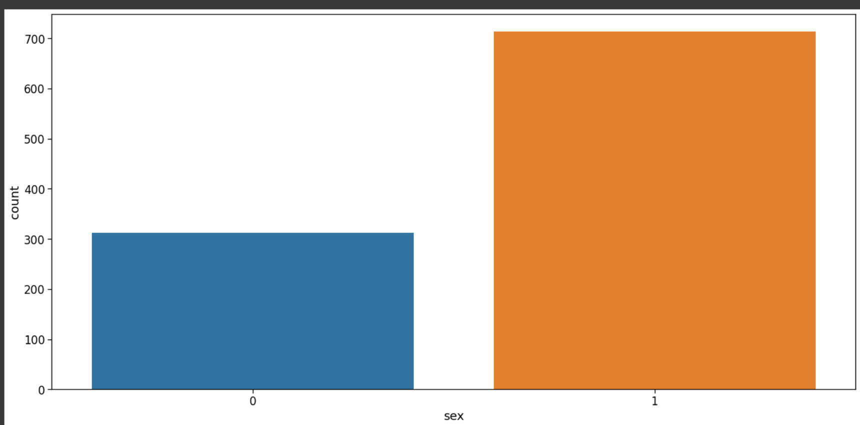



```
[32] #Sex("sex") Feature Analysis
```

```
[33] plt.figure(figsize=(18,9))
     sns.set_context('notebook',font_scale = 1.5)
     sns.countplot(data['sex'])
     plt.tight_layout()
```

```
[33] /usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

     Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```

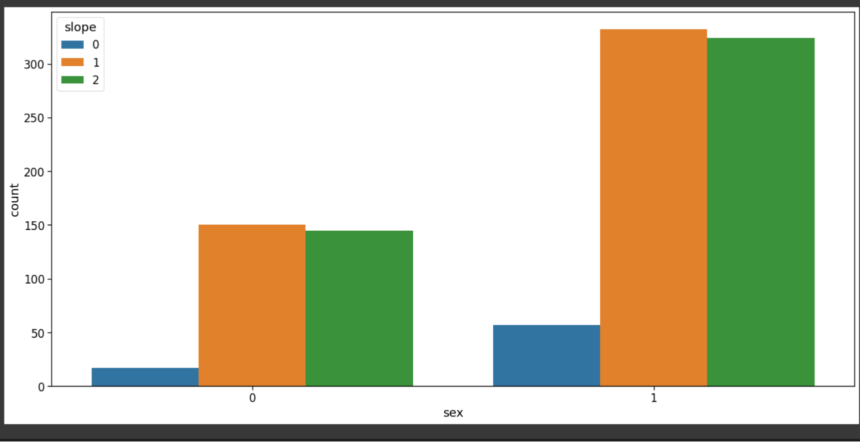


Inference: Here it is clearly visible that, Ratio of Male to Female is approx 2:1.

```
[34] #plot the relation between sex and slope.

     plt.figure(figsize=(18,9))
     sns.set_context('notebook',font_scale = 1.5)
     sns.countplot(data['sex'],hue=data["slope"])
     plt.tight_layout()
```

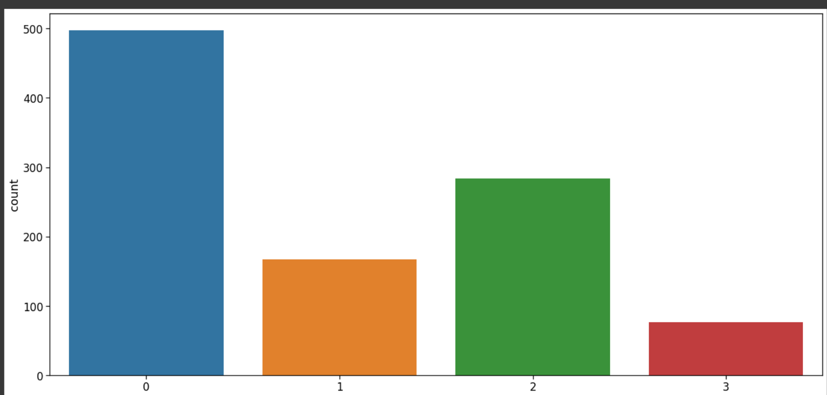
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```



Inference: Here it is clearly visible that the slope value is higher in the case of males.

```
[35] #Chest Pain Type("cp") Analysis
     plt.figure(figsize=(18,9))
     sns.set_context('notebook',font_scale = 1.5)
     sns.countplot(data['cp'])
     plt.tight_layout()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```



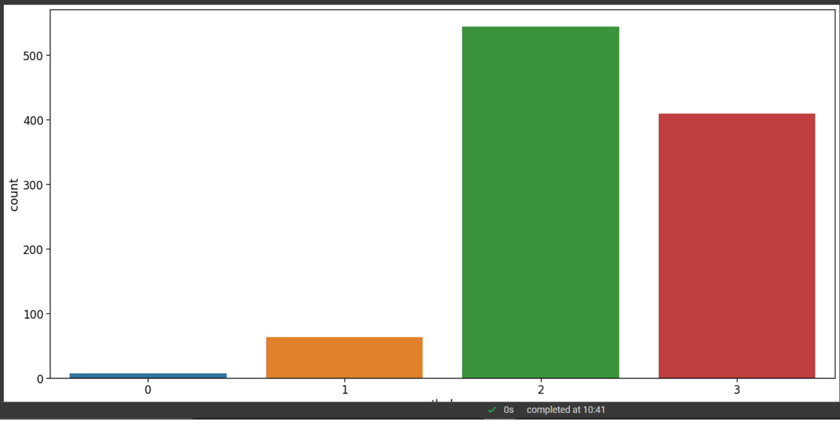
Inference: As seen, there are 4 types of chest pain

1. status at least
2. condition slightly distressed
3. condition medium problem
4. condition too bad

```
[36] #Thal Analysis
     plt.figure(figsize=(18,9))
     sns.set_context('notebook',font_scale = 1.5)
     sns.countplot(data['thal'])
     plt.tight_layout()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```



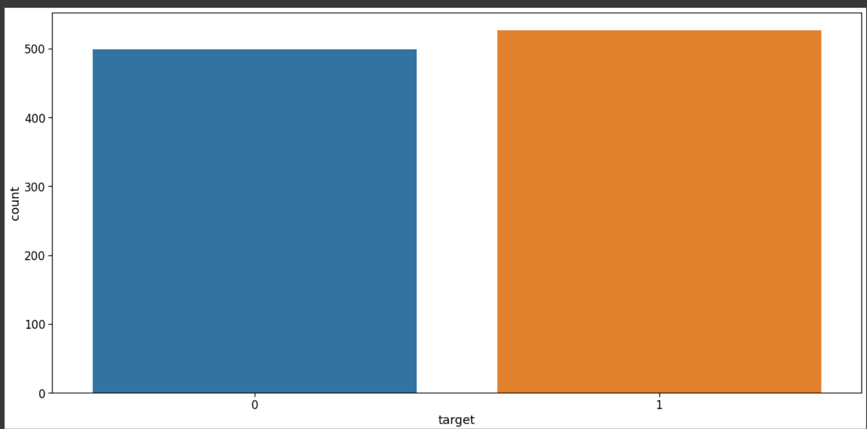
0s completed at 10:41

+ Code + Text

```
[37] #Target
     plt.figure(figsize=(18,9))
     sns.set_context('notebook',font_scale = 1.5)
     sns.countplot(data['target'])
     plt.tight_layout()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```



0s completed at 10:41

Inference: The ratio between 1 and 0 is much less than 1.5 which indicates that the target feature is not imbalanced. So for a balanced dataset, we can use accuracy_score as evaluation metrics for our model.

```python
#Feature Engineering
#complete description of the continuous data as well as the categorical data

categorical_val = []
continous_val = []
for column in data.columns:
    print("--------------------")
    print(f"{column} : {data[column].unique()}")
    if len(data[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
```

```
--------------------
age : [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 63 42 44 56 57 59 64
 65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
--------------------
sex : [1 0]
--------------------
cp : [0 1 2 3]
--------------------
trestbps : [125 140 145 148 138 100 114 160 120 122 112 132 118 128 124 106 104 135
 130 136 180 129 150 178 146 117 152 154 170 134 174 144 108 123 110 142
 126 192 115  94 200 165 102 105 155 172 164 156 101]
--------------------
chol : [212 203 174 294 248 318 289 249 286 149 341 210 298 204 308 266 244 211
 185 223 208 252 209 307 233 319 256 327 169 131 269 196 231 213 271 263
 229 360 258 330 342 226 228 278 230 283 241 175 188 217 193 245 232 299
 288 197 315 215 164 326 207 177 257 255 187 201 220 268 267 236 303 282
 126 309 186 275 281 206 335 218 254 295 417 260 240 302 192 225 325 235
 274 234 182 167 172 321 300 199 564 157 304 222 184 354 160 247 239 246
 409 293 180 250 221 200 227 243 311 261 242 205 306 219 353 198 394 183
 237 224 265 313 340 259 270 216 264 276 322 214 273 253 176 284 305 168
 407 290 277 262 195 166 178 141]
--------------------
fbs : [0 1]
--------------------
restecg : [1 0 2]
--------------------
thalach : [168 155 125 161 106 122 140 145 144 116 136 192 156 142 109 162 165 148
 172 173 146 179 152 117 115 112 163 147 182 105 150 151 169 166 178 132
 160 123 139 111 180 164 202 157 159 170 138 175 158 126 143 141 167  95
 190 118 103 181 108 177 134 120 171 149 154 153  88 174 114 195 133  96
 124 131 185 194 128 127 186 184 188 130  71 137  99 121 187  97  90 129
 113]
--------------------
exang : [0 1]
--------------------
oldpeak : [1.  3.1 2.6 0.  1.9 4.4 0.8 3.2 1.6 3.  0.7 4.2 1.5 2.2 1.1 0.3 0.4 0.6
 3.4 2.8 1.2 2.9 3.6 1.4 0.2 2.  5.6 0.9 1.8 6.2 4.  2.5 0.5 0.1 2.1 2.4
 3.8 2.3 1.3 3.5]
--------------------
slope : [2 0 1]
--------------------
ca : [2 0 1 3 4]
--------------------
thal : [3 2 1 0]
--------------------
target : [0 1]
```

[39]
```python
categorical_val.remove('target')
dfs = pd.get_dummies(data, columns = categorical_val)
dfs.head(6)
```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	52	125	212	168	1.0	0	0	1	1	0	...	1	0	0	1	0	0	0	0	0	1
1	53	140	203	155	3.1	0	0	1	1	0	...	0	1	0	0	0	0	0	0	0	1
2	70	145	174	125	2.6	0	0	1	1	0	...	0	1	0	0	0	0	0	0	0	1
3	61	148	203	161	0.0	0	0	1	1	0	...	1	0	1	0	0	0	0	0	0	1
4	62	138	294	106	1.9	0	1	0	1	0	...	0	0	0	0	1	0	0	0	1	0
5	58	100	248	122	1.0	1	1	0	1	0	...	0	1	0	0	0	0	0	0	1	0

6 rows × 31 columns

```
[40] sc = StandardScaler()
     col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
     dfs[col_to_scale] = sc.fit_transform(dfs[col_to_scale])
     dfs.head(6)
```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	-0.268437	-0.377636	-0.659332	0.821321	-0.060888	0	0	1	1	0	...	1	0	0	1	0	0	0	0	0	1
1	-0.158157	0.479107	-0.833861	0.255968	1.727137	0	0	1	1	0	...	0	1	0	0	0	0	0	0	0	1
2	1.716595	0.764688	-1.396233	-1.048692	1.301417	0	0	1	1	0	...	0	1	0	0	0	0	0	0	0	1
3	0.724079	0.936037	-0.833861	0.516900	-0.912329	0	0	1	1	0	...	1	0	1	0	0	0	0	0	0	1
4	0.834359	0.364875	0.930822	-1.874977	0.705408	0	1	0	1	0	...	0	0	0	0	1	0	0	0	1	0
5	0.393241	-1.805540	0.038784	-1.179158	-0.060888	1	1	0	1	0	...	0	1	0	0	0	0	0	0	1	0

6 rows × 31 columns

```
[31] #Modeling
     #Splitting our Dataset

     X = dfs.drop('target', axis=1)
     y = dfs.target

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
#The KNN Machine Learning Algorithm
knn = KNeighborsClassifier(n_neighbors = 20)
knn.fit(X_train,y_train)
y_pred1 = knn.predict(X_test)
print(accuracy_score(y_test,y_pred1))
```

```
0.8409090909090909
```