```
from google.colab import files
uploaded = files.upload()
```

Choose Files heart.csv

- **heart.csv**(text/csv) - 38114 bytes, last modified: 24/4/2024 - 100% done

```
Saving heart.csv to heart.csv
```

```
from sklearn import datasets
import pandas as pd
df=pd.read_csv("/content/heart.csv")
print(df.head())
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
```

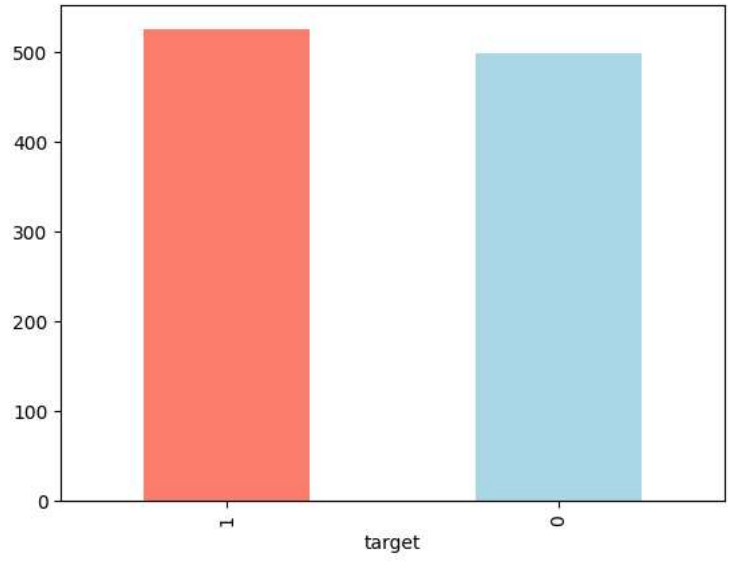
Default title text

```
# @title Default title text
pd.set_option("display.float", "{:.2f}".format)
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang
count	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00	1025.00
mean	54.43	0.70	0.94	131.61	246.00	0.15	0.53	149.11	0.34
std	9.07	0.46	1.03	17.52	51.59	0.36	0.53	23.01	0.47
min	29.00	0.00	0.00	94.00	126.00	0.00	0.00	71.00	0.00
25%	48.00	0.00	0.00	120.00	211.00	0.00	0.00	132.00	0.00
50%	56.00	1.00	1.00	130.00	240.00	0.00	1.00	152.00	0.00
75%	61.00	1.00	2.00	140.00	275.00	0.00	1.00	166.00	1.00

```
df.target.value_counts().plot(kind="bar", color=["salmon", "lightblue"])
```

```
<Axes: xlabel='target'>
```



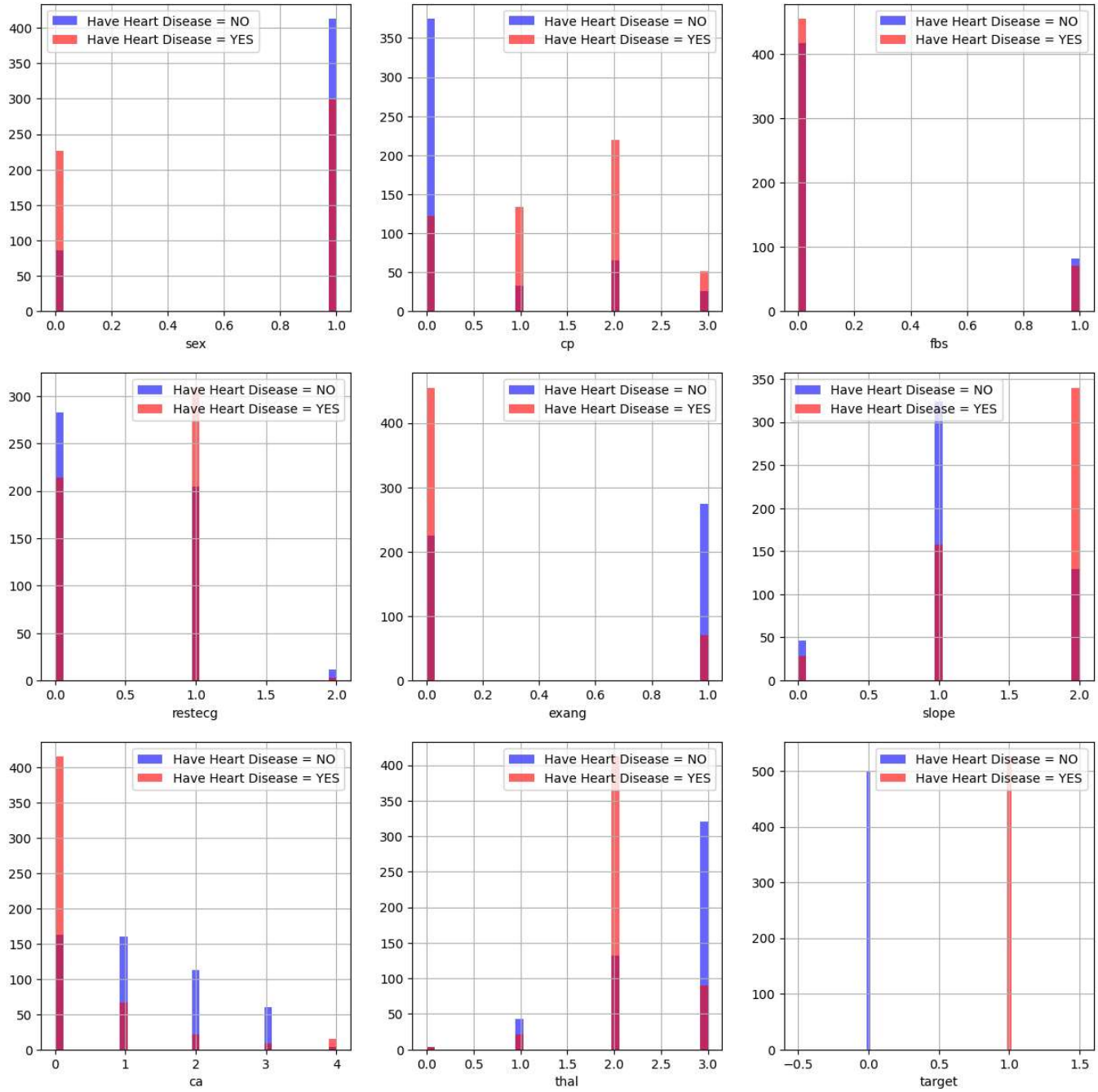

```
# Checking for messing values
df.isna().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

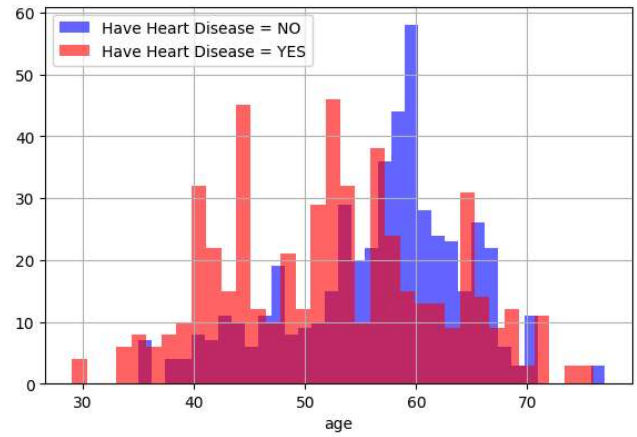
```
categorical_val = []
continous_val = []
for column in df.columns:
    print('==============================')
    print(f"{column} : {df[column].unique()}")
    if len(df[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
```

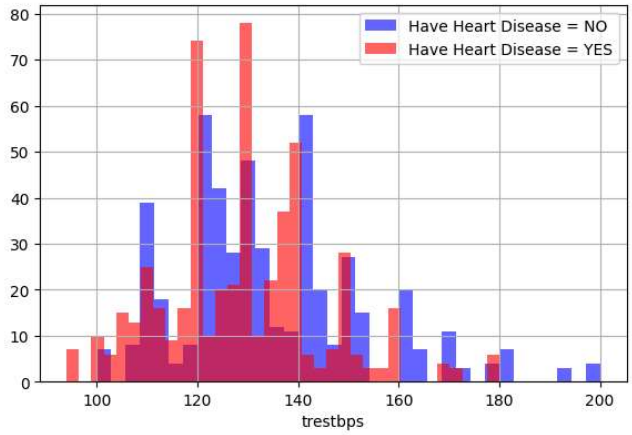
```
==============================
age : [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 63 42 44 56 57 59 64
 65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
==============================
sex : [1 0]
==============================
cp : [0 1 2 3]
==============================
trestbps : [125 140 145 148 138 100 114 160 120 122 112 132 118 128 124 106 104 135
 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]
```

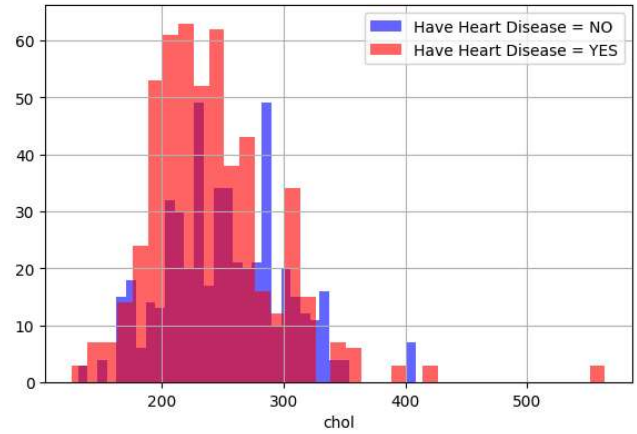
```
import matplotlib.pyplot as plt
plt.figure(figsize=(15, 15))
for i, column in enumerate(categorical_val, 1):
    plt.subplot(3, 3, i)
    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```

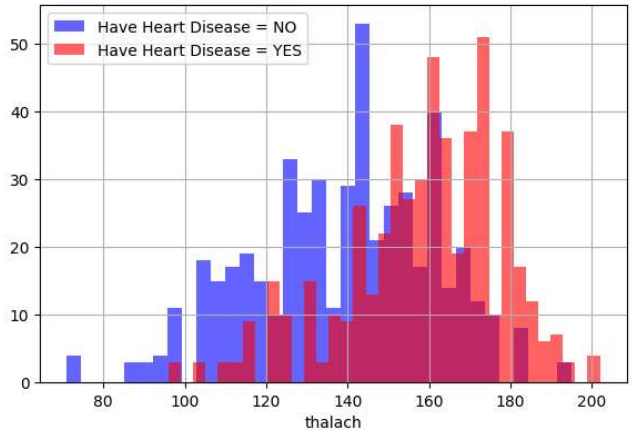


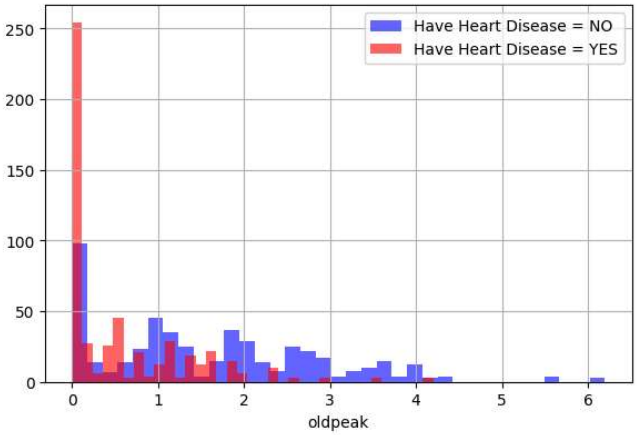

```
plt.figure(figsize=(15, 15))
for i, column in enumerate(continous_val, 1):
    plt.subplot(3, 2, i)
    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```



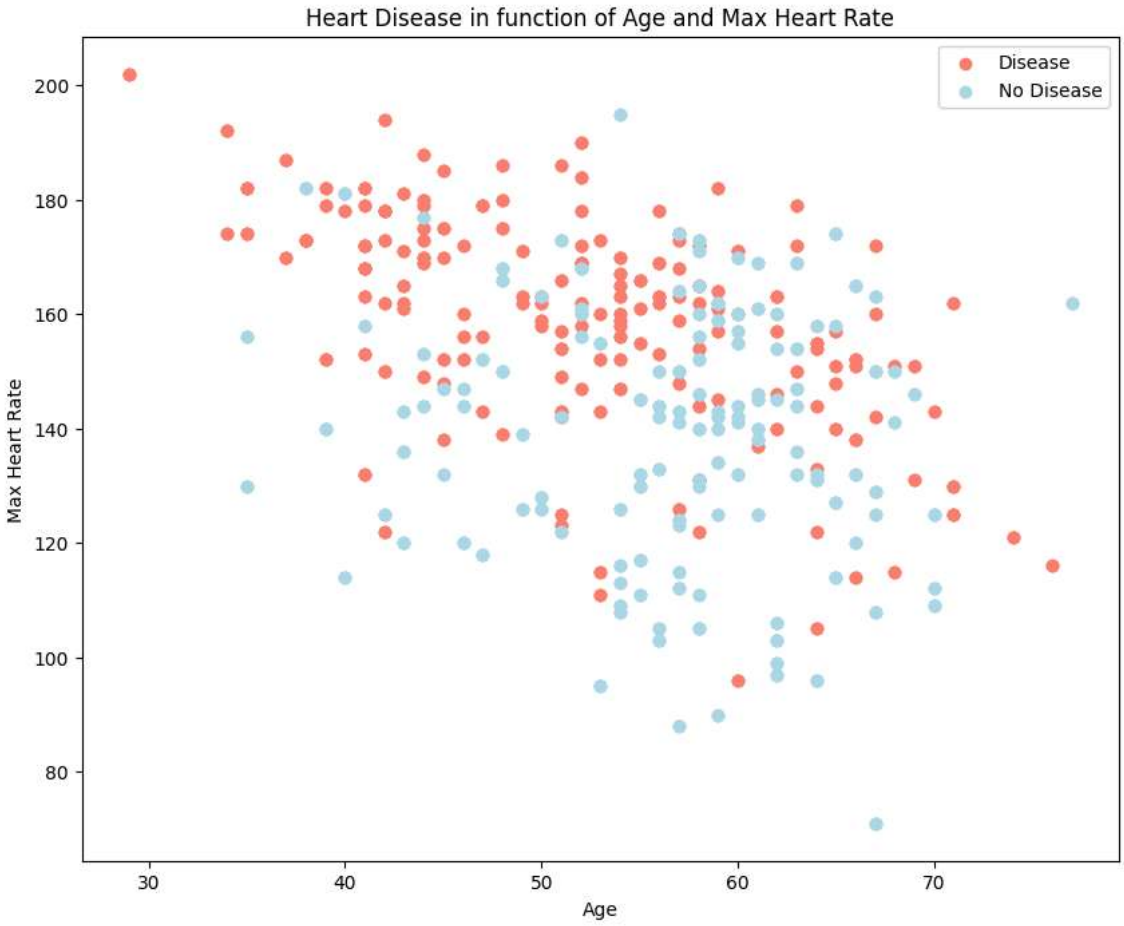






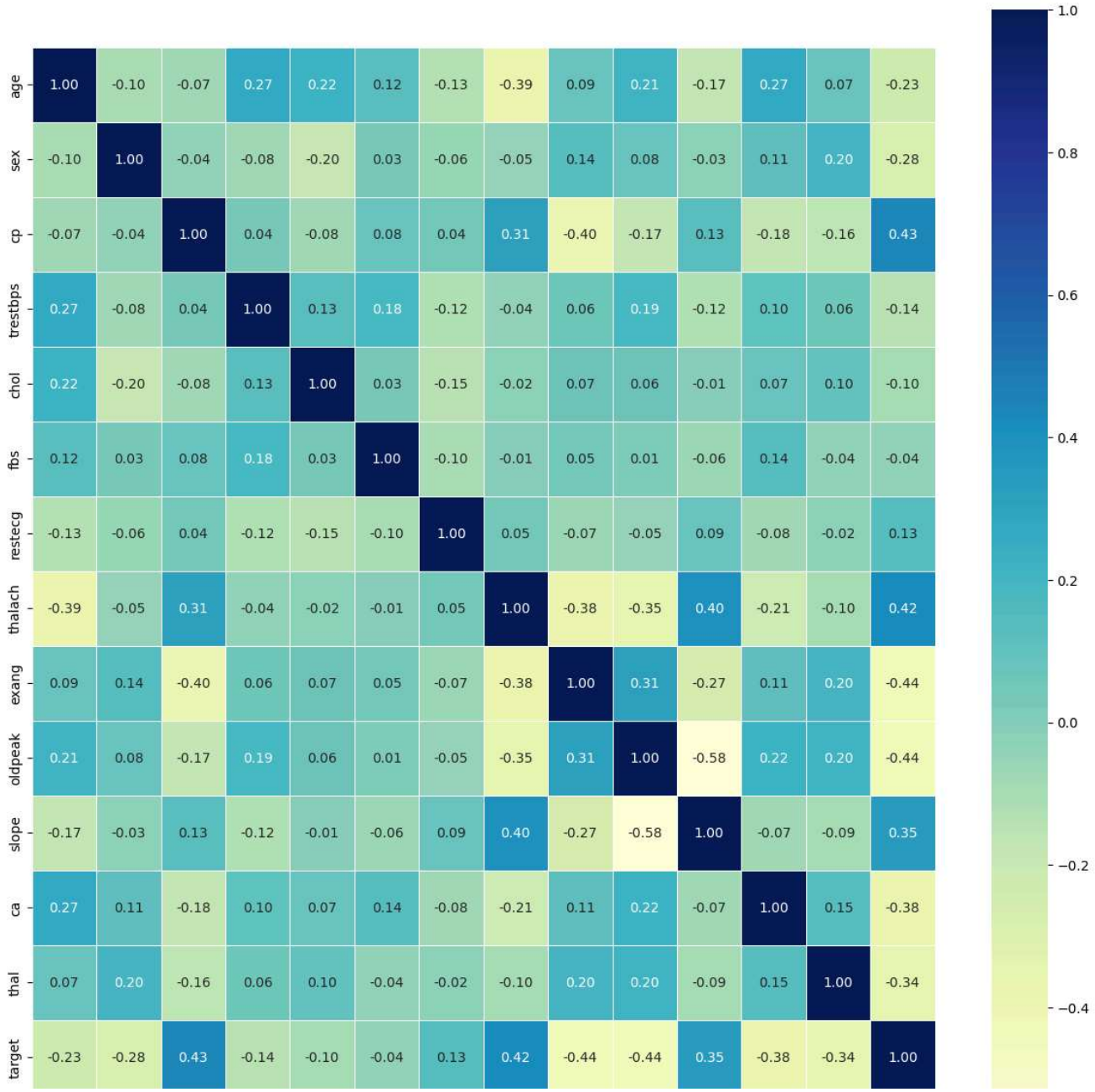



```
# Create another figure
plt.figure(figsize=(10, 8))
# Scatter with postivie examples
plt.scatter(df.age[df.target==1],
            df.thalach[df.target==1],
            c="salmon")
# Scatter with negative examples
plt.scatter(df.age[df.target==0],
            df.thalach[df.target==0],
            c="lightblue")
# Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease", "No Disease"]);
```



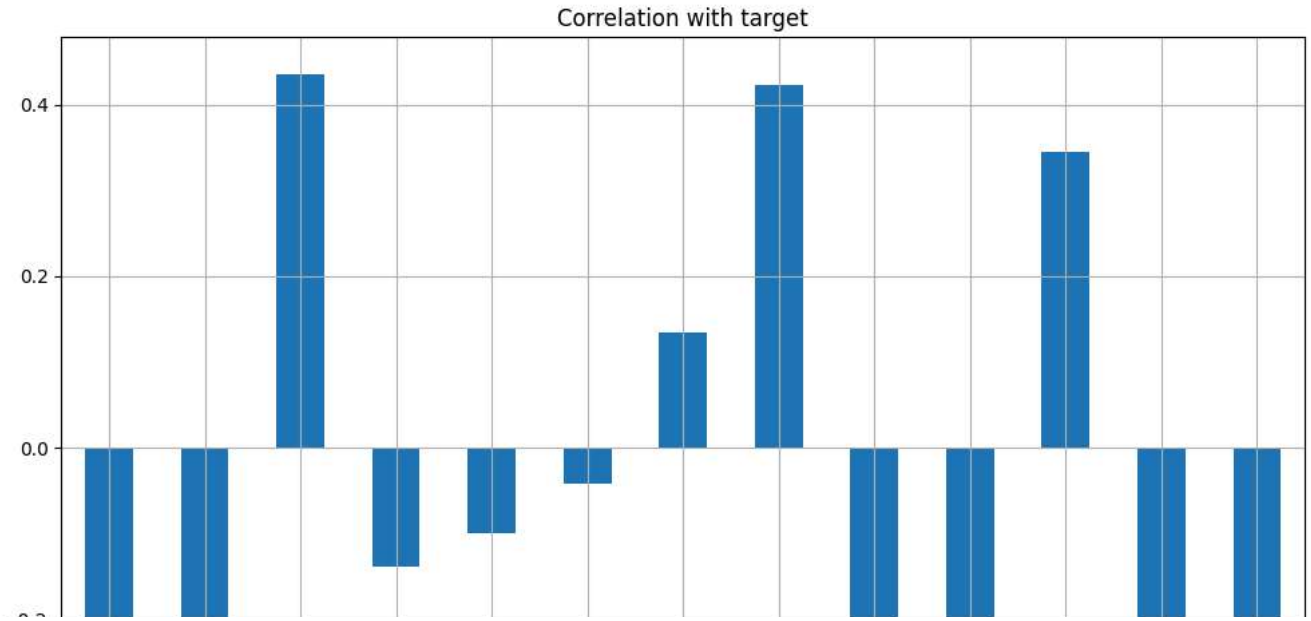

```
import seaborn as sns
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

```
(14.5, -0.5)
```




```
df.drop('target', axis=1).corrwith(df.target).plot(kind='bar', grid=True, figsize=(12, 8), title="Correlation with target")
```

```
<Axes: title={'center': 'Correlation with target'}>
```




```
import numpy as np
```

```
categorical_val = np.array(['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']).tolist()
```

```
dataset = pd.get_dummies(df, columns = categorical_val)
from sklearn.preprocessing import StandardScaler
s_sc = StandardScaler()
col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[col_to_scale] = s_sc.fit_transform(dataset[col_to_scale])
```

```
from sklearn.model_selection import train_test_split
X = dataset.drop('target', axis=1)
y = dataset.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression
lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train, y_train)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 89.54%

_______________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision    0.91   0.89      0.90       0.90          0.90
recall       0.87   0.92      0.90       0.89          0.90
f1-score     0.89   0.90      0.90       0.89          0.90
support    340.00 377.00      0.90     717.00        717.00

_______________________________________________
Confusion Matrix:
 [[295  45]
 [ 30 347]]

Test Result:
================================================
Accuracy Score: 81.82%
```