MACHINE LEARNING

Course-End Project: Healthcare

PROBLEM STATEMENT:

To develop the model and predict the heart attacks that are causing the deaths globally using the given relevant dataset variables

TASK- 1 :

- ☐ Perform preliminary data inspection and report the findings on the structure of the data, missing values, duplicates, etc.
- ☐ Based on these findings, remove duplicates (if any) and treat missing values using an appropriate strategy

CODE:

```
heathcare_df = pd.read_excel("Healthcare_cep1_dataset.xlsx")
heathcare_df.head()
heathcare_df.tail()
heathcare_df.shape
```

Screenshot:

In [7]:
```
#Displays the top 5 rows of the dataset

heathcare_df.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]:
```
#Displays the bottom 5 rows of the dataset

heathcare_df.tail()
```

Out[8]:

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

In [9]:
```
# To find the total number of rows and columns

heathcare_df.shape
```

Out[9]: (303, 14)

Code

heathcare_df.info()

heathcare_df.isnull().sum()

Screenshot:

```
heathcare_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
```

```
[12]: # To find the missing values in each column of the entire dataframe(ALL COLUMNS

heathcare_df.isnull().sum()
```

```
t[12]: age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
```

CONCLUDING FROM ABOVE INFO THAT THE DATASET HAS NO MISSING VALUES

<u>TASK -2</u>

- ☐ **EXPLAINING THE DATA DISTRIBUTION AND THE RELATED FACTORS¶**

<u>Code:</u>

heathcare_df["target"].value_counts()

sns.countplot(data = heathcare_df , x ="target", width=0.3)
plt.xlabel("One and zeros of CVDS")
plt.ylabel("Count of ones and zeros")
plt.title('Countplot of CVD')
plt.xticks(rotation=90)
#Adjust the figure size (optional)
plt.figure(figsize=(10, 10))
plt.show()

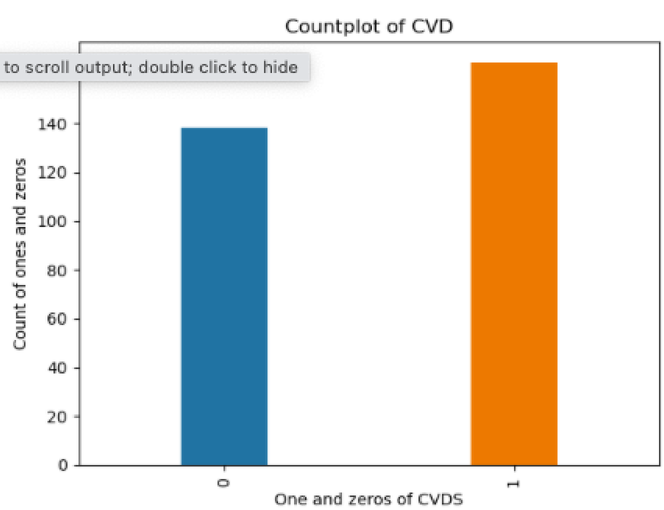
<u>Screenshot</u>

In [14]:
```
# Finding the number of positive CVD'S and negative CVD'S
heathcare_df["target"].value_counts()
```

Out[14]:
```
1    165
0    138
Name: target, dtype: int64
```

In [40]:
```
# FINDING THE COUNT OF CVD'S USING COUNTPLOT

sns.countplot(data = heathcare_df , x ="target", width=0.3)
plt.xlabel("One and zeros of CVDS")
plt.ylabel("Count of ones and zeros")
plt.title('Countplot of CVD')
plt.xticks(rotation=90)
#Adjust the figure size (optional)
plt.figure(figsize=(10, 10))
plt.show()
```

click to scroll output; double click to hide




```
<Figure size 1000x1000 with 0 Axes>
```

Code:
heathcare_df.describe()

print(heathcare_df.corr()['target'].abs().sort_values(ascending=False))

Screenshot

```
In [41]: # Displaying the summary of the statistical analysis of the dataset

         heathcare_df.describe()
```

Out[41]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	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	149.646865	0.326733	1.039604	1.399340	0.729373
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000

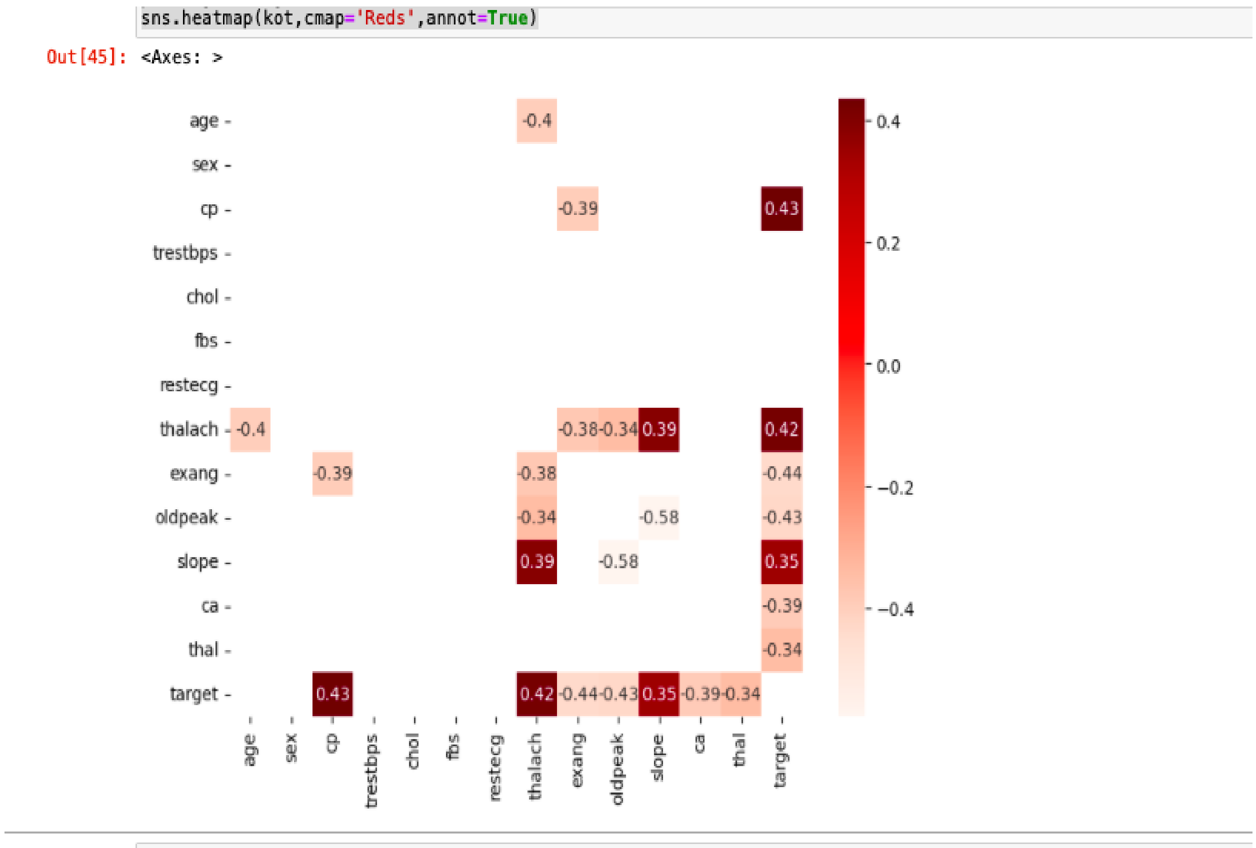
```
In [42]: #checking the correlation of the variables with respect to target variable

         print(heathcare_df.corr()['target'].abs().sort_values(ascending=False))
```

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

Code:

```
corr_diagram = heathcare_df.corr()
thresh=0.3
kot=corr_diagram[((corr_diagram>=thresh)|(corr_diagram<=-thresh))&(corr_diagram!=1)]
plt.figure(figsize=(8,6))
sns.heatmap(kot,cmap='Reds',annot=True)
```

Screenshot:



CODE:

heathcare_df["sex"].value_counts()

pd.crosstab(heathcare_df["target"],heathcare_df["sex"])

pd.crosstab(heathcare_df.target,heathcare_df.sex).plot(kind='bar',figsize=(10,6),color=['yellow','green'])

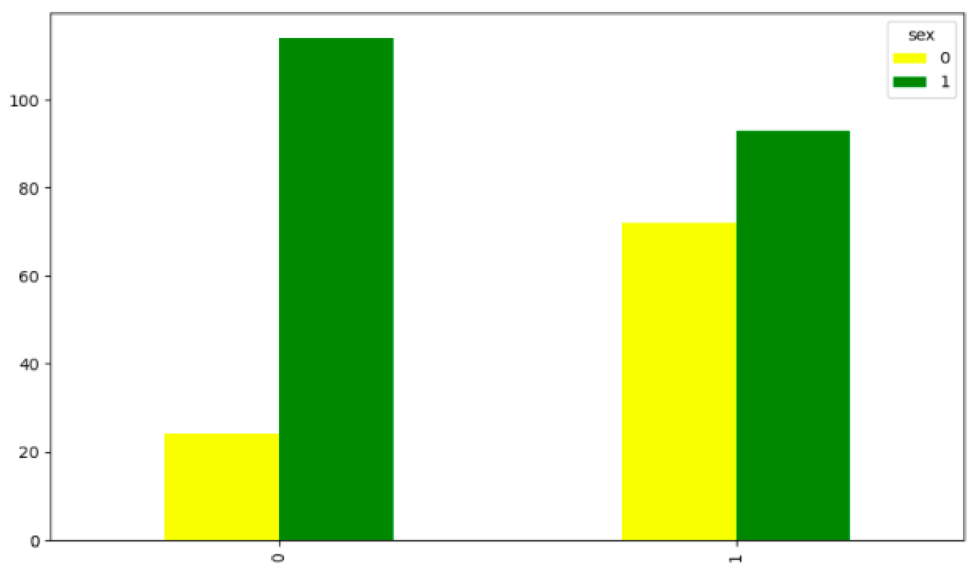
Screenshot:

```
In [46]: heathcare_df["sex"].value_counts()
Out[46]: 1    207
         0     96
         Name: sex, dtype: int64
```

```
In [48]: #creating contingency table to compare sex with target
         pd.crosstab(heathcare_df["target"],heathcare_df["sex"])
Out[48]:
```

sex target	0	1
0	24	114
1	72	93

```
In [72]: #creating plot of CVD against sex
         pd.crosstab(heathcare_df.target,heathcare_df.sex).plot(kind='bar',figsize=(10,6),color=['yellow','green'])
Out[72]: <Axes: xlabel='target'>
```



Inference from above graph
93 males as compared to 72 females are detected with CVD. So males are at a higher risk of CVD

CODE:

pd.crosstab(heathcare_df.cp,heathcare_df.target)

heathcare_df['cp'].unique()

import warnings
warnings.filterwarnings('ignore')
sns.barplot(x=heathcare_df['cp'],y=heathcare_df['target'],data=heathcare_df)

plt.show()

Screenshot:

In [54]:
```
#creating a crosstab for chest pain and target(cp vs target)
pd.crosstab(heathcare_df.cp,heathcare_df.target)
```

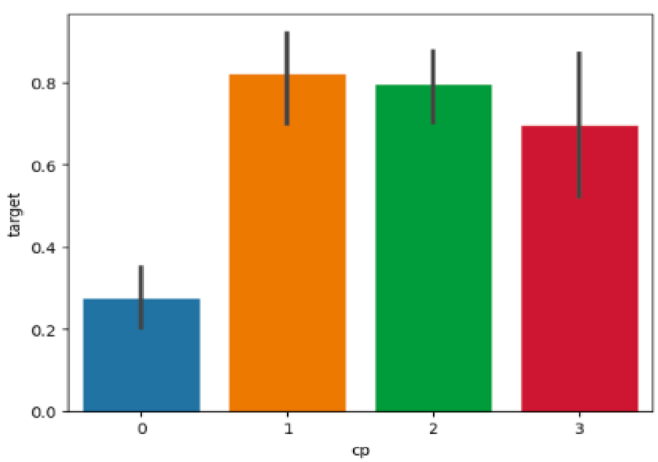
Out[54]:

target	0	1
cp		
0	104	39
1	9	41
2	18	69
3	7	16

In [61]:
```
heathcare_df['cp'].unique()
```

Out[61]:
```
array([3, 2, 1, 0])
```

In [62]:
```
import warnings
warnings.filterwarnings('ignore')
sns.barplot(x=heathcare_df['cp'],y=heathcare_df['target'],data=heathcare_df)
plt.show()
```



Inference from above graph
Maximum people(69) who are suffering from non-anginal pain gets detected with CVD.
Asymptomatic people(104) are least likely to suffer from heart diseases.

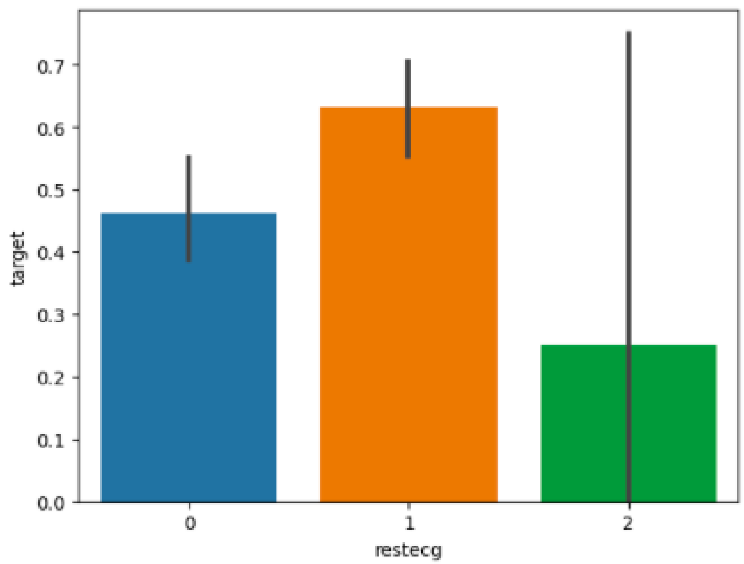
CODE:

heathcare_df['restecg'].unique()

sns.barplot(x='restecg',y='target',data=heathcare_df)
plt.show()

Screenshot:

```
In [ ]:

In [69]: # Analysing the restecg feature
         heathcare_df['restecg'].unique()

Out[69]: array([0, 1, 2])

In [70]: #RESTECG NOTATIONS ARE TAKEN AS BELOW

         #0- 'resting_ecg']  = 'normal'
         #1-'resting_ecg'] = 'abnormal'
         #2- 'resting_ecg'] = 'hyper'

In [68]: sns.barplot(x='restecg',y='target',data=heathcare_df)
         plt.show()
```



#Inference from above graph
#category 1- Abnormal Resting electrocardiographic results show maximum occurences of a CVD

CODE:

#Study if one can detect heart attacks based on anomalies in the resting blood pressure (trestbps) of a patient

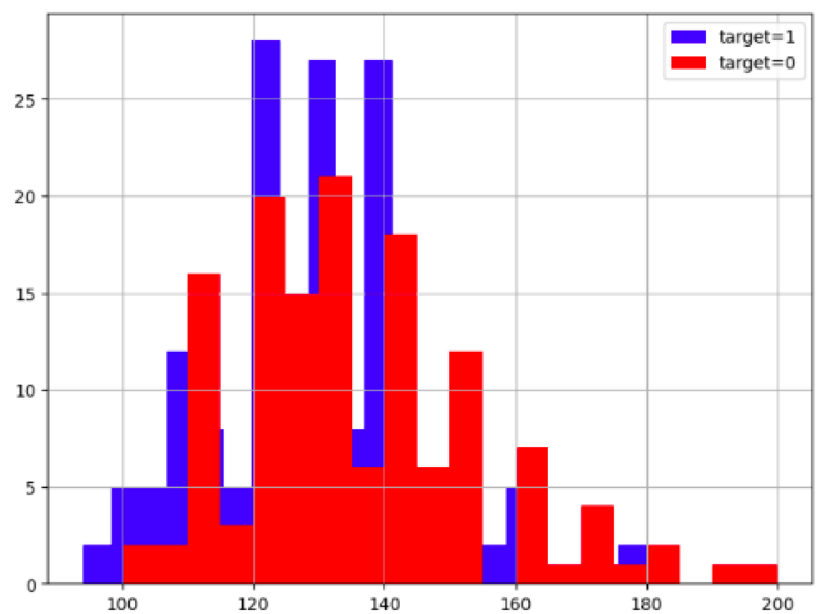
plt.figure(figsize=(8,6))
heathcare_df[heathcare_df['target']==1]['trestbps'].hist(color='blue',bins=20,label='target =1')

heathcare_df[heathcare_df['target']==0]['trestbps'].hist(color='red',bins=20,label='target= 0')

plt.legend()
plt.show()

Screenshot:

```
In [79]: # Study if one can detect heart attacks based on anomalies in the resting blood pressure (trestbps) of
         plt.figure(figsize=(8,6))
         heathcare_df[heathcare_df['target']==1]['trestbps'].hist(color='blue',bins=20,label='target=1')
         heathcare_df[heathcare_df['target']==0]['trestbps'].hist(color='red',bins=20,label='target=0')
         plt.legend()
         plt.show()
```

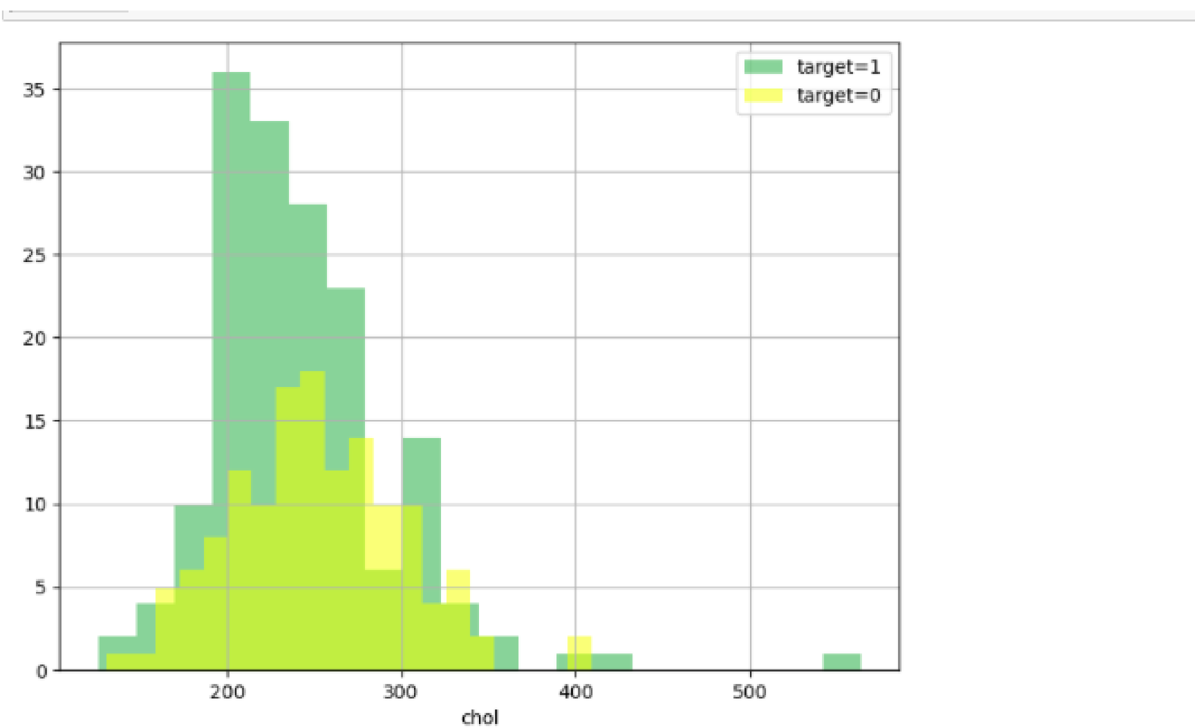


Inference from above graph
If trestbps is between 120 to 140 have higher chances of CVD

CODE:

```
# Describe the relationship between cholesterol levels and a target variable
plt.figure(figsize=(8,6))

heathcare_df[heathcare_df['target']==1]['chol'].hist(alpha=0.4,color='green',bins=20,label
='target=1')

heathcare_df[heathcare_df['target']==0]['chol'].hist(alpha=0.5,color='yellow',bins=20,labe
l='target=0')

plt.legend()
plt.xlabel('chol')
plt.show()
```

Screenshot:

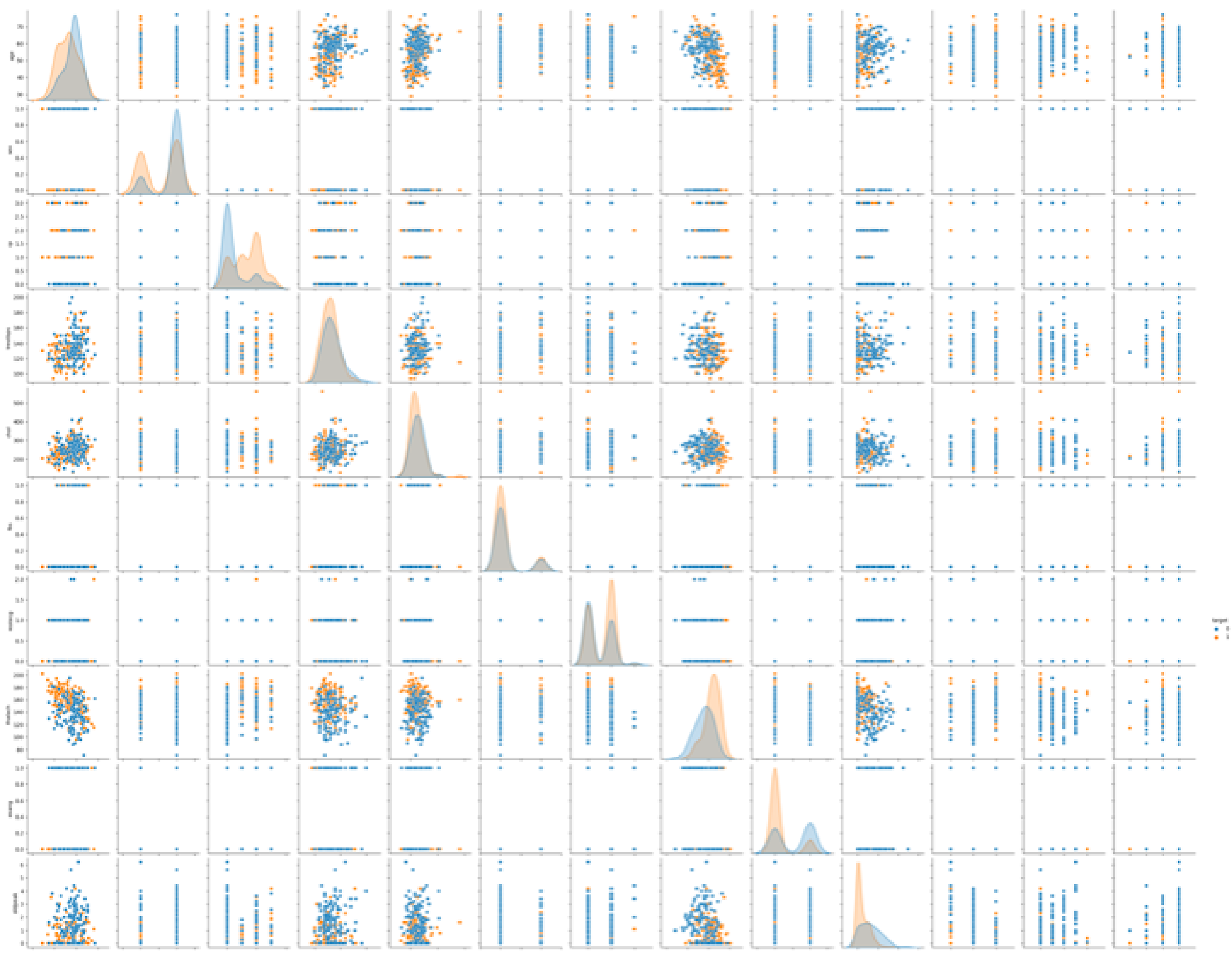


CODE:

```
# Use a pair plot to understand the relationship between all the given variables

plt.figure(figsize=(8,6))
sns.pairplot(data = heathcare_df,hue='target')
plt.title('CVD')
plt.show()
```

Screenshot:

```
<Figure size 800x600 with 0 Axes>
```



TASK-3
MODELLING THE DATASET

Code:

#Splitting the dataset

```
from sklearn.model_selection import train_test_split
predictors= heathcare_df.drop('target',axis=1)
target = heathcare_df['target']
X_train,X_test,y_train,y_test=train_test_split(predictors,target,test_size=0.25,random_st
ate=0)
```

#building the logistic regression model

```
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression()
lr.fit(X_train,y_train)

y_pred = lr.predict(X_test)
```

#Calculating the accuracy

```
from sklearn.metrics import accuracy_score
score_lr = round(accuracy_score(y_pred,y_test)*100,2)
print("The accuracy score achieved using logistic regression is: "+ str(score_lr)+" %")
```

#fitting a stats logistic regression model

```
import statsmodels.api as sm
log_reg=sm.Logit(y_train,X_train).fit()
```

#printing the summary reports

```
print(log_reg.summary())
```

<u>Screenshot:</u>

In [83]:
```
from sklearn.model_selection import train_test_split
predictors= heathcare_df.drop('target',axis=1)
target = heathcare_df['target']
X_train,X_test,y_train,y_test=train_test_split(predictors,target,test_size=0.25,random_state=0)
```

In [84]:
```
#building the logistic regression model
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression()
lr.fit(X_train,y_train)
```

Out[84]:
```
▾ LogisticRegression
LogisticRegression()
```

In [85]:
```
y_pred = lr.predict(X_test)
```

In [86]:
```
from sklearn.metrics import accuracy_score
score_lr = round(accuracy_score(y_pred,y_test)*100,2)
print("The accuracy score achieved using logistic regression is: "+ str(score_lr)+" %")
```

```
The accuracy score achieved using logistic regression is: 84.21 %
```

In [87]:
```
#fitting a stats logistic regression model
import statsmodels.api as sm
log_reg=sm.Logit(y_train,X_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.343229
         Iterations 7
```

In [89]:
```
#printing the summary reports
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:                  227
Model:                          Logit   Df Residuals:                      214
Method:                           MLE   Df Model:                           12
Date:                Wed, 20 Sep 2023   Pseudo R-squ.:                  0.5028
Time:                        14:03:53   Log-Likelihood:                -77.913
converged:                       True   LL-Null:                       -156.71
Covariance Type:            nonrobust   LLR p-value:                 1.627e-27
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
age            0.0176      0.022      0.803      0.422      -0.025       0.060
sex           -2.0263      0.531     -3.815      0.000      -3.067      -0.985
cp             0.8986      0.223      4.027      0.000       0.461       1.336
```

#BUILDING THE RANDOMFOREST AND CALCULATING THE ACCURACY

SCREENSHOT:

```
n [90]: # Building the Randomforest model

        from sklearn.ensemble import RandomForestClassifier
        clf=RandomForestClassifier(criterion='gini',
                                   max_depth=7,
                                   n_estimators=200,
                                   #min_samples_split=10,
                                   random_state=5)

n [91]: #fitting the model
        clf.fit(X_train,y_train)

ut[91]: ▾ RandomForestClassifier
        RandomForestClassifier(max_depth=7, n_estimators=200, random_state=5)

n [92]: y_predt=clf.predict(X_test)

n [93]: clf.feature_importances_

ut[93]: array([0.07794457, 0.05003741, 0.15391964, 0.06958547, 0.07236738,
               0.01105043, 0.0165406 , 0.11611831, 0.05549952, 0.11698825,
               0.04239796, 0.11501138, 0.1025391 ])

n [95]: heathcare_df.columns

ut[95]: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
               'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
              dtype='object')

n [96]: from sklearn.metrics import confusion_matrix
        confusion_matrix(y_test,y_predt)

ut[96]: array([[25,  8],
               [ 3, 40]])

n [99]: accuracy_score(y_test,y_predt)*100

ut[99]: 85.52631578947368
```