In [2]:

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

In [9]:

```
df = pd.read_csv("Heart.csv")
```

In [10]:

```
df
```

...

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [12]:

```
df['sex'] = df['sex'].astype('object')
df['cp'] = df['cp'].astype('object')
df['fbs'] = df['fbs'].astype('object')
df['restecg'] = df['restecg'].astype('object')
df['exang'] = df['exang'].astype('object')
df['slope'] = df['slope'].astype('object')
df['ca'] = df['ca'].astype('object')
df['thal'] = df['thal'].astype('object')
```

In [13]:

```
df.dtypes
```

...

In [14]:

```
df.isnull().sum()
```

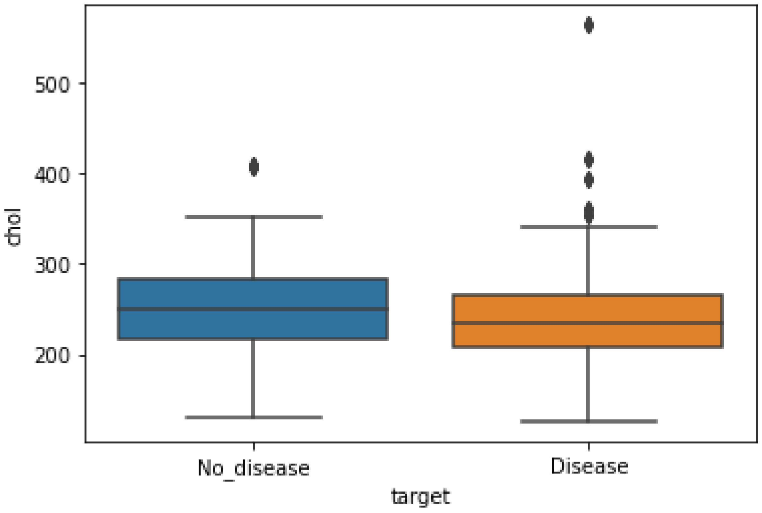
...

```
In [15]: df['target'] = df.target.replace({1: "Disease", 0: "No_disease"})
         df['sex'] = df.sex.replace({1: "Male", 0: "Female"})
         df['cp'] = df.cp.replace({0: "typical_angina",1: "atypical_angina",2:"non-anginal
         df['exang'] = df.exang.replace({1: "Yes", 0: "No"})
         df['fbs'] = df.fbs.replace({1: "True", 0: "False"})
         df['slope'] = df.slope.replace({0: "upsloping", 1:"flat",2:"downsloping"})
         df['thal'] = df.thal.replace({1: "fixed_defect", 2: "reversable_defect",3:"normal
```

```
In [16]: bxplt = sns.boxplot(df['target'],df['chol'])
         plt.show
```

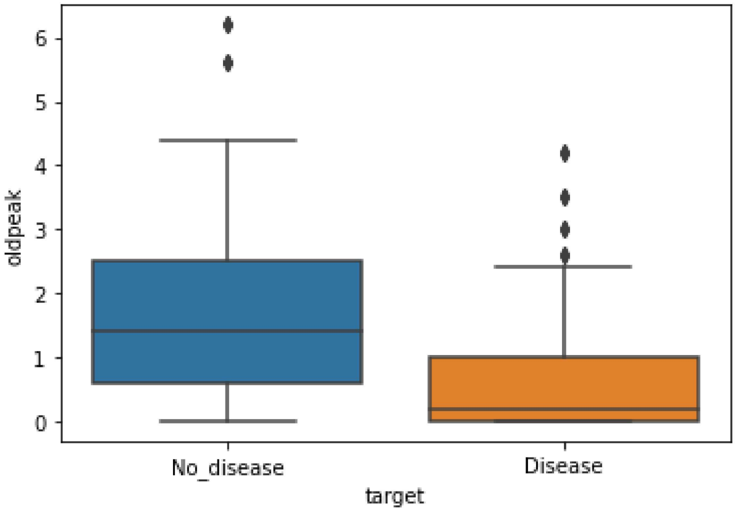
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
Out[16]: <function matplotlib.pyplot.show(close=None, block=None)>
```



```
In [19]: sns.boxplot(x="target",y="oldpeak", data=df)
```

```
Out[19]: <AxesSubplot:xlabel='target', ylabel='oldpeak'>
```



In [20]:

```python

```

In [23]:

```python
continous_features = ['age','trestbps','chol','thalach','oldpeak']
```

In [24]:

```python

```

Out[24]:

```
ModeResult(mode=array(['age'], dtype='<U8'), count=array([1]))
```

In [26]:

```python
def outliers(df_out, drop = False):
    for each_feature in df_out.columns:
        feature_data = df_out[each_feature]
        Q1 = np.percentile(feature_data, 25.)
        Q3 = np.percentile(feature_data, 75.)
        IQR = Q3-Q1
        outlier_step = IQR * 1.5
        outliers = feature_data[~((feature_data >= Q1 - outlier_step) & (feature_

        if not drop:
            print('For the feature {}, No of Outliers is {}'.format(each_feature,
        if drop:
            df.drop(outliers, inplace = True, errors = 'ignore')
            print('Outliers from {} feature removed'.format(each_feature))
outliers(df[continous_features])
```

```
For the feature age, No of Outliers is 0
For the feature trestbps, No of Outliers is 30
For the feature chol, No of Outliers is 16
For the feature thalach, No of Outliers is 4
For the feature oldpeak, No of Outliers is 7
```

In [27]:

```python
outliers(df[continous_features],drop=True)
```

```
Outliers from age feature removed
Outliers from trestbps feature removed
Outliers from chol feature removed
Outliers from thalach feature removed
Outliers from oldpeak feature removed
```

In [28]:

```python
duplicated=df.duplicated().sum()
```

In [31]:

```python
duplicated
```

Out[31]:

```
683
```

In [32]:

```python
if duplicated:
    print("Duplicated rows :{}".format(duplicated))
else:
    print("No duplicates")
```

```
Duplicated rows :683
```

In [33]:

```
duplicates=df[df.duplicated(keep=False)]
duplicates.head()
```

Out[33]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	52	Male	typical_angina	125	212	False	1	168	No	1.0	downsloping
1	53	Male	typical_angina	140	203	True	0	155	Yes	3.1	upsloping
2	70	Male	typical_angina	145	174	False	1	125	Yes	2.6	upsloping
3	61	Male	typical_angina	148	203	False	1	161	No	0.0	downsloping
4	62	Female	typical_angina	138	294	True	1	106	No	1.9	flat

In [34]:

```
df.drop_duplicates()
```

Out[34]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slo
0	52	Male	typical_angina	125	212	False	1	168	No	1.0	downslop
1	53	Male	typical_angina	140	203	True	0	155	Yes	3.1	upslop
2	70	Male	typical_angina	145	174	False	1	125	Yes	2.6	upslop
3	61	Male	typical_angina	148	203	False	1	161	No	0.0	downslop
4	62	Female	typical_angina	138	294	True	1	106	No	1.9	
...	...	...	...	...	...	...	...	...	...	...	
723	68	Female	non-anginal pain	120	211	False	0	115	No	1.5	
733	44	Female	non-anginal pain	108	141	False	1	175	No	0.6	
739	52	Male	typical_angina	128	255	False	1	161	Yes	0.0	downslop
843	59	Male	asymtomatic	160	273	False	0	125	No	0.0	downslop
878	54	Male	typical_angina	120	188	False	1	113	No	1.4	

285 rows × 14 columns

In [35]:
```
duplicates.head()
```

Out[35]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	52	Male	typical_angina	125	212	False	1	168	No	1.0	downsloping
1	53	Male	typical_angina	140	203	True	0	155	Yes	3.1	upsloping
2	70	Male	typical_angina	145	174	False	1	125	Yes	2.6	upsloping
3	61	Male	typical_angina	148	203	False	1	161	No	0.0	downsloping
4	62	Female	typical_angina	138	294	True	1	106	No	1.9	flat

In [36]:
```
df.drop_duplicates()
```

Out[36]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slo
0	52	Male	typical_angina	125	212	False	1	168	No	1.0	downslop
1	53	Male	typical_angina	140	203	True	0	155	Yes	3.1	upslop
2	70	Male	typical_angina	145	174	False	1	125	Yes	2.6	upslop
3	61	Male	typical_angina	148	203	False	1	161	No	0.0	downslop
4	62	Female	typical_angina	138	294	True	1	106	No	1.9	
...	...	...	...	...	...	...	...	...	...	...	
723	68	Female	non-anginal pain	120	211	False	0	115	No	1.5	
733	44	Female	non-anginal pain	108	141	False	1	175	No	0.6	
739	52	Male	typical_angina	128	255	False	1	161	Yes	0.0	downslop
843	59	Male	asymtomatic	160	273	False	0	125	No	0.0	downslop
878	54	Male	typical_angina	120	188	False	1	113	No	1.4	

285 rows × 14 columns

In [37]:
```
duplicated=df.duplicated().sum()
```

In [38]:
```
duplicated
```

Out[38]:
```
683
```

In [39]:
```
df1 = pd.read_csv("student.csv", header = 0)
df2 = pd.read_csv("mark.csv", header = 0)
```

In [40]:
```
df1.info()
```

...

In [41]:
```
df2.info()
```
...

In [42]:
```
df1.head()
```
...

In [43]:
```
df2.head()
```
...

In [44]:
```
df = pd.merge(df1, df2, on = 'Student_id')
df.head(10)
```
...

In [46]:
```
df = pd.read_csv("Heart.csv")
df
```
...

In [47]:
```
ddf =pd.read_csv("data.csv",encoding='cp1252')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:344
4: DtypeWarning: Columns (0) have mixed types.Specify dtype option on import or
set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [80]:
```
ddf.head()
```

Out[80]:

	state	location	type	so2	no2	rspm	spm	pm2_5	date
0	0	Hyderabad	Residential, Rural and other Areas	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01
1	0	Hyderabad	Industrial Area	3.1	7.0	108.833091	220.78348	40.791467	1990-02-01
2	0	Hyderabad	Residential, Rural and other Areas	6.2	28.5	108.833091	220.78348	40.791467	1990-02-01
3	0	Hyderabad	Residential, Rural and other Areas	6.3	14.7	108.833091	220.78348	40.791467	1990-03-01
4	0	Hyderabad	Industrial Area	4.7	7.5	108.833091	220.78348	40.791467	1990-03-01

In [50]:
```
ddf=ddf.drop(['stn_code','agency','sampling_date','location_monitoring_station'],
```

In [51]:
```
ddf=ddf.dropna(subset=['date'])
```

In [52]:
```
COLS = ['so2', 'no2', 'rspm', 'spm', 'pm2_5']
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
ddf[COLS] = imputer.fit_transform(ddf[COLS])
```

In [53]:
```
ddf[COLS]
```

...

In [54]:
```
ddf['type'].value_counts()
```

Out[54]:
```
Residential, Rural and other Areas    179013
Industrial Area                        96089
Residential and others                 86791
Industrial Areas                       51747
Sensitive Area                          8979
Sensitive Areas                         5536
RIRUO                                   1304
Sensitive                                495
Industrial                               233
Residential                              158
Name: type, dtype: int64
```

In [55]:
```
ddf['type']
```

Out[55]:
```
0         Residential, Rural and other Areas
1                            Industrial Area
2         Residential, Rural and other Areas
3         Residential, Rural and other Areas
4                            Industrial Area
                         ...                
435734                                 RIRUO
435735                                 RIRUO
435736                                 RIRUO
435737                                 RIRUO
435738                                 RIRUO
Name: type, Length: 435735, dtype: object
```

In [56]:
```
ddf['state'].value_counts()
```

...

In [57]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
ddf["state"]=labelencoder.fit_transform(ddf["state"])
ddf.head(5)
```

Out[57]:

	state	location	type	so2	no2	rspm	spm	pm2_5	date
0	0	Hyderabad	Residential, Rural and other Areas	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01
1	0	Hyderabad	Industrial Area	3.1	7.0	108.833091	220.78348	40.791467	1990-02-01
2	0	Hyderabad	Residential, Rural and other Areas	6.2	28.5	108.833091	220.78348	40.791467	1990-02-01
3	0	Hyderabad	Residential, Rural and other Areas	6.3	14.7	108.833091	220.78348	40.791467	1990-03-01
4	0	Hyderabad	Industrial Area	4.7	7.5	108.833091	220.78348	40.791467	1990-03-01

In [58]:
```
dfAndhra=ddf[(ddf['state']==0)]
dfAndhra
```

...

In [59]:
```
dfAndhra['location'].value_counts()
```

...

In [60]:
```
dfAndhra=ddf[(ddf['state']==0)]
dfAndhra
```

...

In [61]:
```
dfAndhra['location'].value_counts()
```

Out[61]:
```
Hyderabad         7764
Visakhapatnam     7108
Vijayawada        2093
Chittoor          1003
Tirupati           986
Kurnool            857
Patancheru         698
Guntur             629
Nalgonda           618
Ramagundam         554
Nellore            408
Khammam            385
Warangal           336
Ananthapur         324
Ongole             317
Kadapa             316
Srikakulam         315
Rajahmundry        311
Eluru              300
Vishakhapatnam     297
```

In [62]:
```
from sklearn.preprocessing import OneHotEncoder
onehotencoder=OneHotEncoder(sparse=False,handle_unknown='error',drop='first')
pd.DataFrame(onehotencoder.fit_transform(dfAndhra[["location"]]))
```

...

In [63]:
```
df['ca'].unique()
```

Out[63]:
```
array([2, 0, 1, 3, 4], dtype=int64)
```

In [64]:
```
df[df['ca']==4]
```

...

In [65]:
```
df.loc[df['ca']==4,'ca']=np.NaN
```

In [66]:
```
df['thal'].nunique()
```
Out[66]:
```
4
```
In [67]:
```
df['thal'].unique()
```
Out[67]:
```
array([3, 2, 1, 0], dtype=int64)
```
In [68]:
```
df[df['thal']==3]
```
...

In [69]:
```
df.loc[df['thal']==3,'thal']=np.NaN
```
In [70]:
```
df.isna().sum()
```
...

In [71]:
```
df = df.fillna(df.median())
df.isnull().sum()
```
...

In [72]:
```
X = df.drop('target', axis=1)
```
In [73]:
```
X.head()
```
Out[73]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	1	0	125	212	0	1	168	0	1.0	2	2.0	2.0
1	53	1	0	140	203	1	0	155	1	3.1	0	0.0	2.0
2	70	1	0	145	174	0	1	125	1	2.6	0	0.0	2.0
3	61	1	0	148	203	0	1	161	0	0.0	2	1.0	2.0
4	62	0	0	138	294	1	1	106	0	1.9	1	3.0	2.0

In [74]:
```
X.shape
```
Out[74]:
```
(1025, 13)
```
In [75]:
```
y = df['target']
y.head(10)
```
...

In [76]:
```
from sklearn import preprocessing
df=df.apply(preprocessing.LabelEncoder().fit_transform)
```
In [77]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_st
```

In [78]:

```
print("X_train : ",X_train.shape)
print("X_test : ",X_test.shape)
print("y_train : ",y_train.shape)
print("y_test : ",y_test.shape)
```

```
X_train :  (820, 13)
X_test :  (205, 13)
y_train :  (820,)
y_test :  (205,)
```

In [79]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
clf = DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(y_pred)
```

```
[1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0 0 1 0 0 0
 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 0
 0 0 0 1 0 0 0 0 1 0 1 0 1 0 1 0 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 1 0 1 0 1 1
 0 1 0 1 0 1 0 0 1 0 1 1 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 0 1 0 0 1 1 1
 1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 0 1
 0 1 0 1 0 0 1 0 1 0 1 1 1 0 0 1 1 1 0 1]
```

In [ ]: