```
from google.colab import drive
drive.mount('/content/drive/')
```

```
     Mounted at /content/drive/
```

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

```
dataset='/content/drive/MyDrive/DSBDA/heart.csv'
import pandas as pd
import numpy as np
df=pd.read_csv(dataset)
```

```
print(df.shape)
print(df.info())
```

```
     (303, 14)
     <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
     None
```

Check data type

```
df.dtypes
```

```
     age           int64
     sex           int64
     cp            int64
     trestbps      int64
     chol          int64
```

```
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

```
# to know unique values
df.nunique()
```

```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          4
target        2
dtype: int64
```

```
# change the categorical type to categorical variables
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')
df.dtypes
```

```
age           int64
sex          object
cp           object
trestbps      int64
chol          int64
fbs          object
restecg      object
thalach       int64
exang        object
oldpeak     float64
slope        object
ca           object
thal         object
target        int64
dtype: object
```

Error Correction

```
df['ca'].unique()
```

```
    array([0, 2, 1, 3, 4], dtype=object)
```

```
# to count the number in of each category decending order
print(df.ca.value_counts())

df[df['ca']==4]
```

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

```
df['ca'].unique()
```

```
    array([0, 2, 1, 3, nan], dtype=object)
```

Feature ‘thal’ ranges from 1−3, however, df.nunique() listed 0−3. There are two values of ‘0’. So lets change them to NaN

```
df.thal.value_counts()
```

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

```
df.loc[df['thal']==0,'thal']=np.NaN
```

```
df[df['thal']==0]
```

```
df['thal'].unique()
```

```
     array([1, 2, 3, nan], dtype=object)
```

Check for missing values and replace **them**

```
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          5
     thal        2
     target      0
     dtype: int64
```

```
from sklearn.impute import KNNImputer
# define imputer
imputer = KNNImputer(n_neighbors=5, weights='uniform', metric='nan_euclidean')
```

```
# fit on the dataset
imputer.fit(df)
```

```
     KNNImputer()
```

```
...
# transform the dataset
Xtrans = imputer.transform(df)
```

```
print('Missing: %d' % sum(isnan(Xtrans).flatten()))
```

```
     Missing: 0
```

```
df = df.fillna(df.median())
df.isnull().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
```

Check for duplicate rows

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

```
     Duplicated rows :1
```

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

statistical summary

```
df.describe()
```

change the labeling for better visualization and interpretation.

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

```
df.describe()
```

```
df.head(5)
```

```
features_cols=['age', 'cp', 'trestbps', 'chol', 'fbs','restecg', 'thalach', 'exang', 'oldp
x_train=df[features_cols]
print(df['target'].head())
y_train=df['target']
```

```
     0    1
     1    1
     2    1
     3    1
     4    1
     Name: target, dtype: int64
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
x_train,x_test,y_train,y_test=train_test_split(x_train,y_train,test_size=0.3,random_state=

clf=DecisionTreeClassifier()
clf=clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
Y_train_pred=clf.predict(x_train)
print("Accuracy:" , metrics.accuracy_score(y_test,y_pred))
```

```
     Accuracy: 0.7582417582417582
```

```
from sklearn import tree
tree.plot_tree(clf)
```

```
[Text(0.5371621621621622, 0.9375, 'X[1] <= 0.5\ngini = 0.497\nsamples = 212\nvalue =
 Text(0.2972972972972973, 0.8125, 'X[10] <= 0.5\ngini = 0.387\nsamples = 103\nvalue =
 Text(0.21621621621621623, 0.6875, 'X[11] <= 2.5\ngini = 0.498\nsamples = 51\nvalue =
 Text(0.13513513513513514, 0.5625, 'X[7] <= 0.5\ngini = 0.4\nsamples = 29\nvalue = [8
 Text(0.05405405405405406, 0.4375, 'X[6] <= 96.5\ngini = 0.198\nsamples = 18\nvalue =
 Text(0.02702702702702703, 0.3125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.08108108108108109, 0.3125, 'X[3] <= 300.5\ngini = 0.111\nsamples = 17\nvalue
 Text(0.05405405405405406, 0.1875, 'gini = 0.0\nsamples = 15\nvalue = [0, 15]'),
 Text(0.10810810810810811, 0.1875, 'X[0] <= 61.5\ngini = 0.5\nsamples = 2\nvalue = [1
 Text(0.08108108108108109, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.13513513513513514, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.21621621621621623, 0.4375, 'X[5] <= 0.5\ngini = 0.496\nsamples = 11\nvalue =
 Text(0.1891891891891892, 0.3125, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.24324324324324326, 0.3125, 'X[2] <= 122.0\ngini = 0.375\nsamples = 8\nvalue =
 Text(0.21621621621621623, 0.1875, 'X[3] <= 185.0\ngini = 0.444\nsamples = 3\nvalue =
 Text(0.1891891891891892, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.24324324324324326, 0.0625, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.2702702702702703, 0.1875, 'gini = 0.0\nsamples = 5\nvalue = [5, 0]'),
 Text(0.2972972972972973, 0.5625, 'X[6] <= 158.5\ngini = 0.236\nsamples = 22\nvalue =
 Text(0.2702702702702703, 0.4375, 'gini = 0.0\nsamples = 17\nvalue = [17, 0]'),
 Text(0.32432432432432434, 0.4375, 'X[3] <= 254.0\ngini = 0.48\nsamples = 5\nvalue =
 Text(0.2972972972972973, 0.3125, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.35135135135135137, 0.3125, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.3783783783783784, 0.6875, 'X[0] <= 63.5\ngini = 0.109\nsamples = 52\nvalue =
 Text(0.35135135135135137, 0.5625, 'gini = 0.0\nsamples = 39\nvalue = [39, 0]'),
 Text(0.40540540540540543, 0.5625, 'X[5] <= 0.5\ngini = 0.355\nsamples = 13\nvalue =
 Text(0.3783783783783784, 0.4375, 'gini = 0.0\nsamples = 9\nvalue = [9, 0]'),
 Text(0.43243243243243246, 0.4375, 'X[6] <= 152.5\ngini = 0.375\nsamples = 4\nvalue =
 Text(0.40540540540540543, 0.3125, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.4594594594594595, 0.3125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.777027027027027, 0.8125, 'X[10] <= 0.5\ngini = 0.322\nsamples = 109\nvalue =
 Text(0.6621621621621622, 0.6875, 'X[8] <= 2.8\ngini = 0.178\nsamples = 81\nvalue = [
 Text(0.5945945945945946, 0.5625, 'X[3] <= 272.0\ngini = 0.121\nsamples = 77\nvalue =
 Text(0.5405405405405406, 0.4375, 'X[9] <= 0.5\ngini = 0.033\nsamples = 60\nvalue = [
 Text(0.5135135135135135, 0.3125, 'X[2] <= 115.0\ngini = 0.278\nsamples = 6\nvalue =
 Text(0.4864864864864865, 0.1875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.5405405405405406, 0.1875, 'gini = 0.0\nsamples = 5\nvalue = [0, 5]'),
```

```
import graphviz
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus
```

```
 Text(0.6756756756756757, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
```

```
data1=export_graphviz(clf,out_file=None , feature_names=features_cols)
graph=pydotplus.graph_from_dot_data(data1)
graph.write_png('/content/drive/MyDrive/Heart.png')

Image(graph.create_png())
```

Outliers Detection & Handling

```
import matplotlib.pyplot as plt
import seaborn as sb
bxplt = sb.boxplot(df["target"],df["chol"])
plt.show()
```

```
sb.boxplot(x='target', y='oldpeak', data=df)
```

Drop Outliers

```
# define continuous variable & plot
continous_features = ['age','trestbps','chol','thalach','oldpeak']
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.) # 25th percentile of the data of the given f
        Q3 = np.percentile(feature_data, 75.) # 75th percentile of the data of the given f
        IQR = Q3-Q1 #Interquartile Range
        outlier_step = IQR * 1.5 #That's we were talking about above
        outliers = feature_data[~((feature_data >= Q1 - outlier_step) & (feature_data <= Q
        if not drop:
            print('For the feature {}, No of Outliers is {}'.format(each_feature, len(outl
        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 9
For the feature chol, No of Outliers is 5
For the feature thalach, No of Outliers is 1
For the feature oldpeak, No of Outliers is 5
```

```
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
```

✓ 1s completed at 3:55 PM ● ×

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.