**HEART DISEASE PREDICTION**

**Using : Anaconda (jupyter notebook)**

Step1 : import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

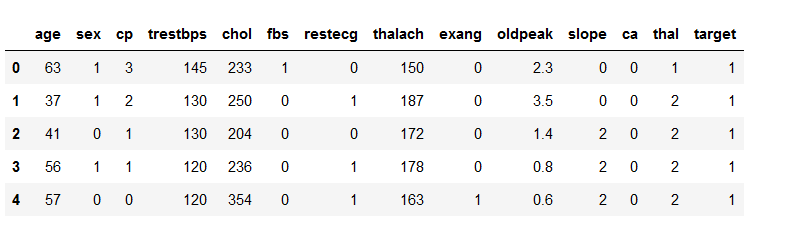
import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

Step2 : df=pd.read\_csv('downloads/heart.csv')

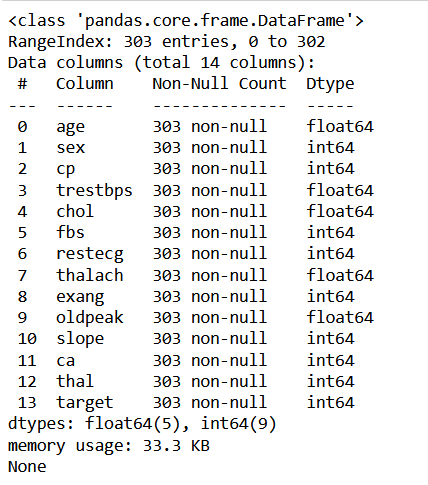
Step3 : df.head()



Step4 : df.isnull().sum()

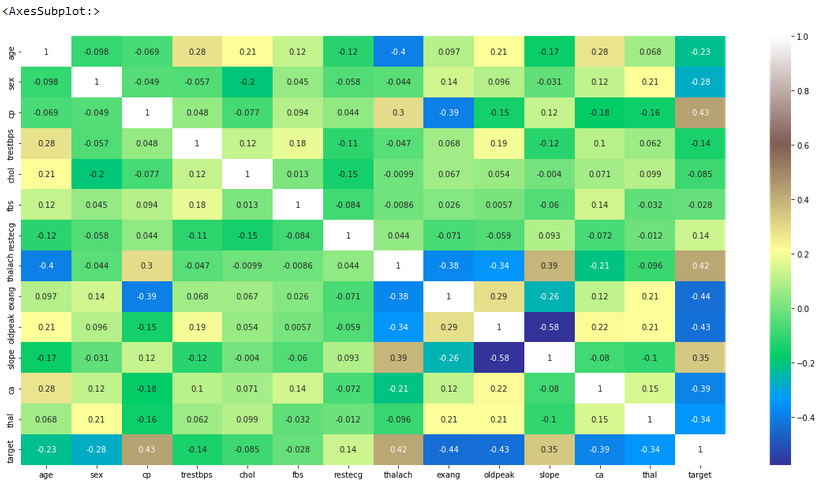


Step5:print(df.info())

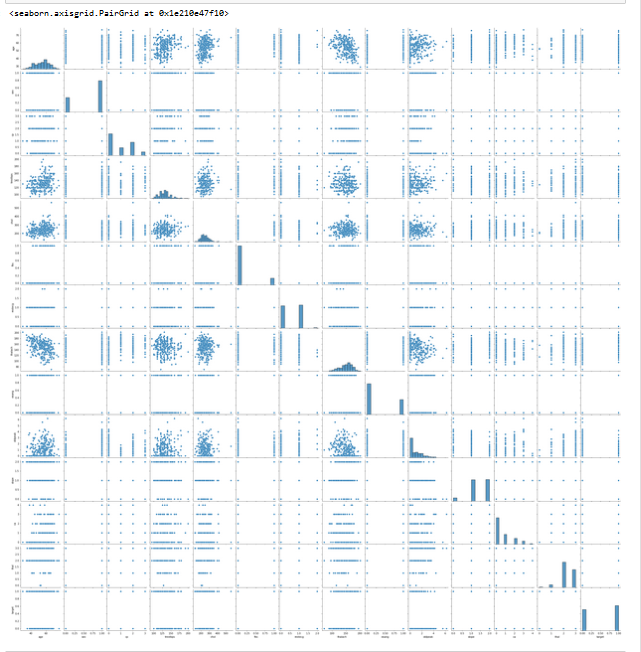


Step6 : plt.figure(figsize=(20,10))

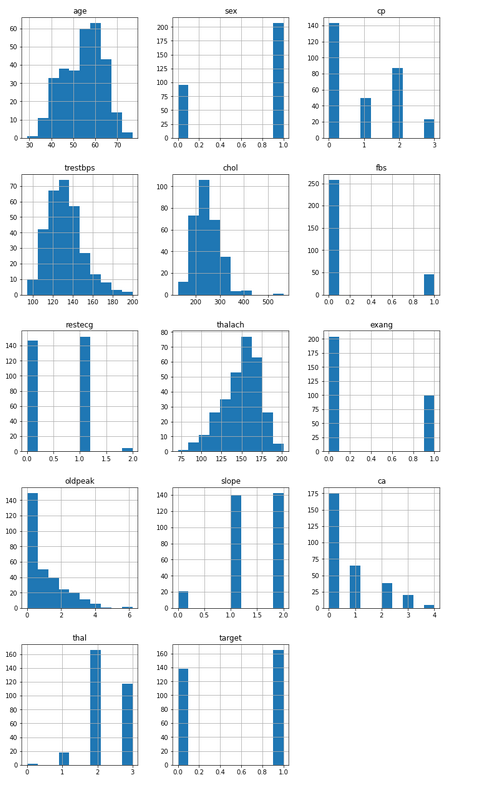
sns.heatmap(df.corr(),annot=True, cmap='terrain')



Step7 : sns.pairplot(data=df)



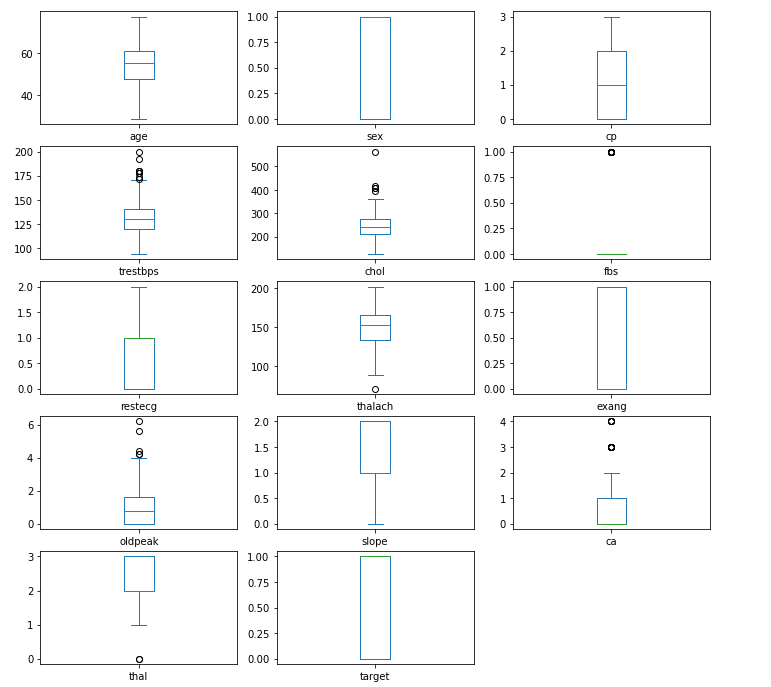
Step8 : df.hist(figsize=(12,22),layout=(5,3));



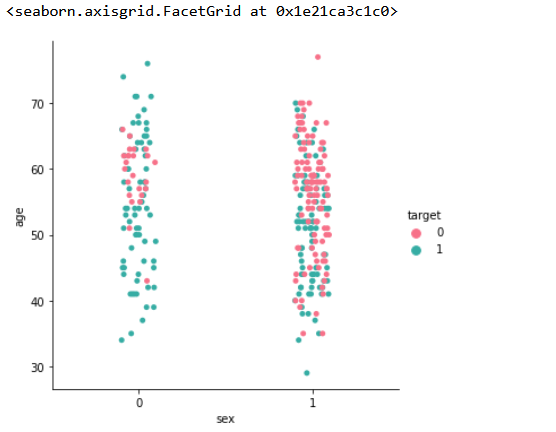
Step9 : #box and whiskers plot

df.plot(kind='box',subplots=True,layout=(5,3),figsize=(12,12))

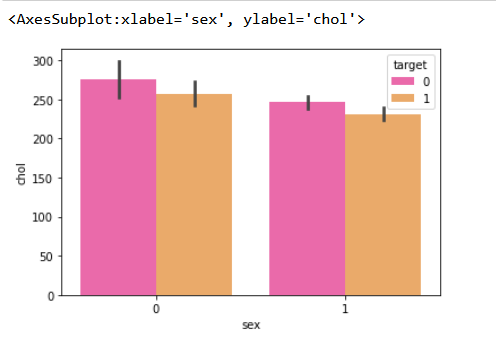
plt.show()



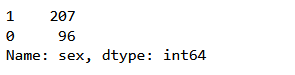
Step10 : sns.catplot(data=df,x='sex',y='age',hue='target',palette='husl')



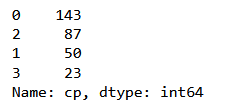
Step11: sns.barplot(data=df,x='sex',y='chol',hue='target',palette='spring')



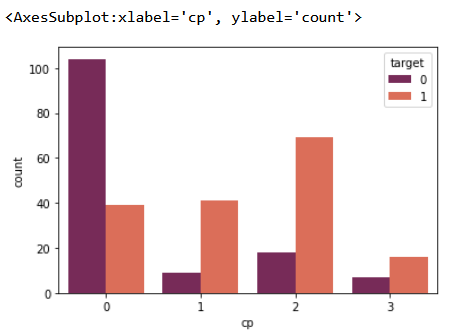
Step12 : df['sex'].value\_counts()



Step13 : df['cp'].value\_counts()

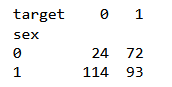


Step14 : sns.countplot(x='cp',hue='target',data=df,palette='rocket')

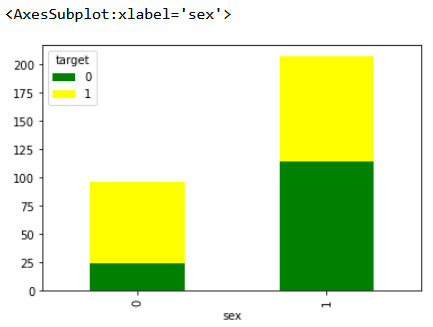


Step15 : gen=pd.crosstab(df['sex'],df['target'])

print(gen)

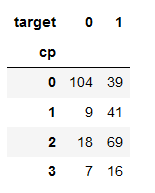


Step16 : gen.plot(kind='bar',stacked=True,color=['green','yellow'],grid=False)



Step17 : chest\_pain = pd.crosstab(df['cp'],df['target'])

chest\_pain



Step18 : from sklearn.model\_selection import train\_test\_split

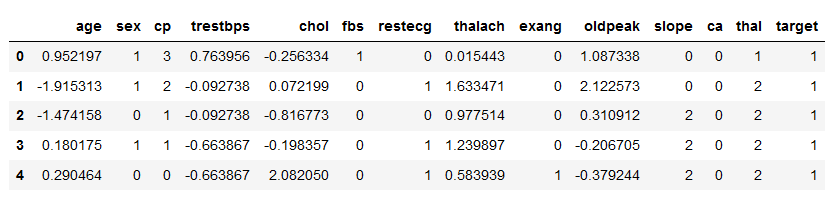
from sklearn.preprocessing import StandardScaler

StandardScaler =StandardScaler()

columns\_to\_scale=['age','trestbps','chol','thalach','oldpeak']

df[columns\_to\_scale]=StandardScaler.fit\_transform(df[columns\_to\_scale])

Step19 : df.head()



Step20 : x=df.drop(['target'],axis=1)

y=df['target']

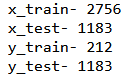
Step21 : x\_train, x\_test, y\_train ,y\_test=train\_test\_split(x,y,test\_size=0.3,random\_state=40)

Step22 : print('x\_train-',x\_train.size)

print('x\_test-',x\_test.size)

print('y\_train-',y\_train.size)

print('y\_test-',x\_test.size)



Step23 : from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier().fit(x\_train,y\_train)

knn.score(x\_test,y\_test)



Step24 : from sklearn.ensemble import RandomForestClassifier

np.random.seed(41)

rf=RandomForestClassifier().fit(x\_train,y\_train)

rf.score(x\_test,y\_test)



Step25 : from sklearn.linear\_model import LogisticRegression

lr=LogisticRegression()

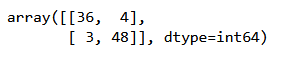
model1=lr.fit(x\_train,y\_train)

prediction1=model1.predict(x\_test)

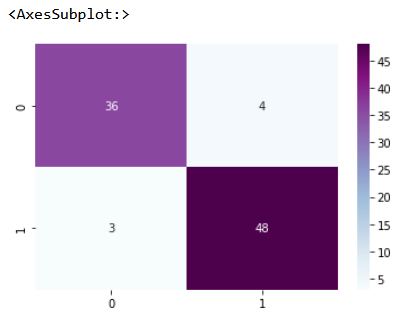
Step26 : from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(y\_test,prediction1)

cm



Step27 : sns.heatmap(cm,annot=True,cmap='BuPu')



Step28 : TP=cm[0][0]

TN=cm[1][1]

FN=cm[1][0]

FP=cm[0][1]

print('Testing Accuracy: ',(TP+TN)/(TP+TN+FN+FP))



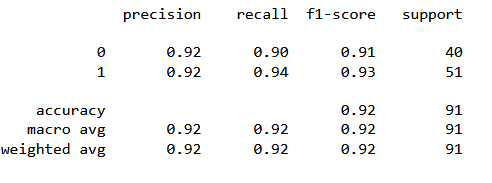
Step29 : from sklearn.metrics import accuracy\_score

accuracy\_score(y\_test,prediction1)



Step30 : from sklearn.metrics import classification\_report

print(classification\_report(y\_test,prediction1))



Step31 : print('lr :',accuracy\_score(y\_test,prediction1))



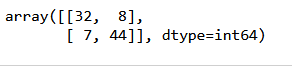
Step32 : from sklearn.metrics import confusion\_matrix

prediction=knn.predict(x\_test)

prediction

confusion\_matrix=confusion\_matrix(y\_test,prediction)

confusion\_matrix



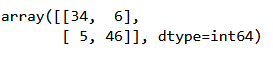
Step33 : from sklearn.metrics import confusion\_matrix

prediction=rf.predict(x\_test)

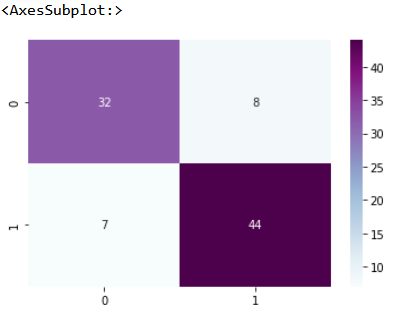
prediction

confusion\_matrix1=confusion\_matrix(y\_test,prediction)

confusion\_matrix1



Step34 : sns.heatmap(confusion\_matrix,annot=True,cmap='BuPu')



Step35 : TP=confusion\_matrix[0][0]

TN=confusion\_matrix[1][1]

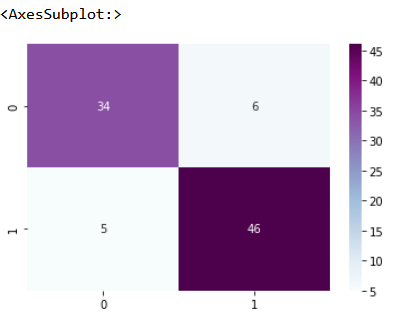
FN=confusion\_matrix[1][0]

FP=confusion\_matrix[0][1]

Print('Testing Accuracy: ',(TP+TN)/(TP+TN+FN+FP))



Step36 : sns.heatmap(confusion\_matrix1,annot=True,cmap='BuPu')



Step37 : TP=confusion\_matrix1[0][0]

TN=confusion\_matrix1[1][1]

FN=confusion\_matrix1[1][0]

FP=confusion\_matrix1[0][1]

print('Testing Accuracy: ',(TP+TN)/(TP+TN+FN+FP)

