**NAME:** DEEPJYOTI DEKA

**CERTIFICATION CODE:** TCRIG02R51

**EMAIL:** deepjyotideka8@gmail.com

**BATCH:** MACHINE LEARNING WITH PYTHON

**PROJECT NAME-** PREDICTION OF HEART DISEASE DETECTION

**GROUP :** OWN

# TCR Final Project

# Heart Problems detection using K Nearest Neighbors Algorithm

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

df = pd.read\_csv('heart.csv' )

df.head()

**Output :**

|  | **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 |

df.describe()

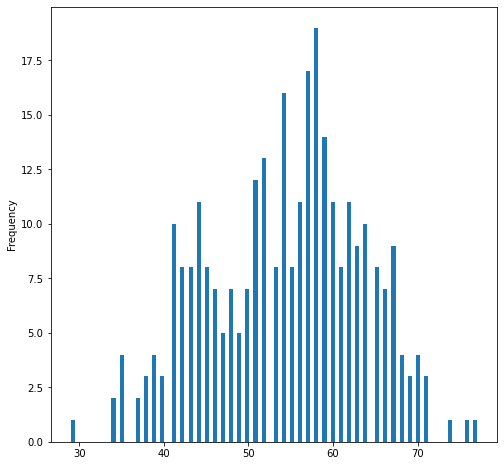
**Output :**

|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **thalach** | **exang** | **oldpeak** | **slope** | **ca** | **thal** | **target** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 | 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 | 2.313531 | 0.544554 |
| **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 | 0.612277 | 0.498835 |
| **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 | 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 | 2.000000 | 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 | 2.000000 | 1.000000 |
| **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 | 3.000000 | 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 | 3.000000 | 1.000000 |

df.age.plot(kind = 'hist', bins = 100, figsize = (8,8))

plt.show()

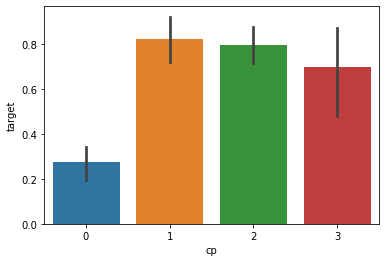
**Output :**



sns.barplot(df["cp"],df['target'])

**Output :**

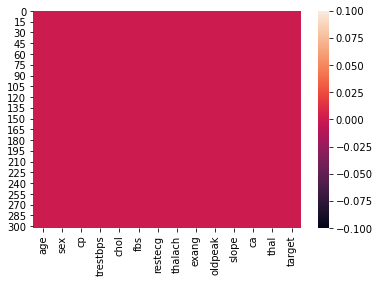
<AxesSubplot:xlabel='cp', ylabel='target'>



sns.heatmap(df.isnull())

**Output :**

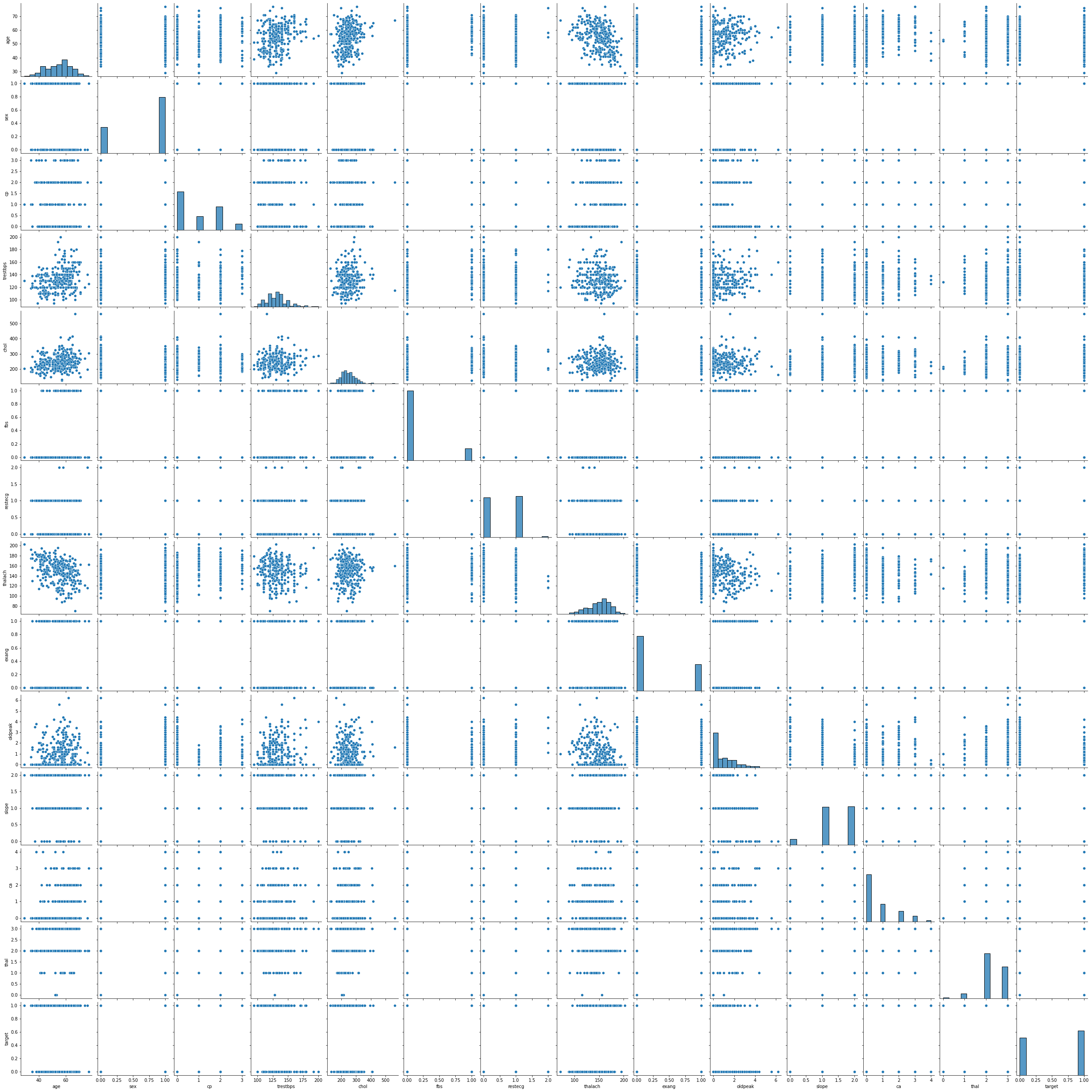
<AxesSubplot:>



sns.pairplot(df)

<seaborn.axisgrid.PairGrid at 0x7f94cfa131c0>

**Output :**



from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df.drop('target' , axis =1))

StandardScaler()

scaler\_featured = scaler.transform(df.drop("target" , axis= 1))

df\_feat = pd.DataFrame(scaler\_featured , columns = df.columns[:-1])

df\_feat

**Output :**

|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **thalach** | **exang** | **oldpeak** | **slope** | **ca** | **thal** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.952197 | 0.681005 | 1.973123 | 0.763956 | -0.256334 | 2.394438 | -1.005832 | 0.015443 | -0.696631 | 1.087338 | -2.274579 | -0.714429 | -2.148873 |
| **1** | -1.915313 | 0.681005 | 1.002577 | -0.092738 | 0.072199 | -0.417635 | 0.898962 | 1.633471 | -0.696631 | 2.122573 | -2.274579 | -0.714429 | -0.512922 |
| **2** | -1.474158 | -1.468418 | 0.032031 | -0.092738 | -0.816773 | -0.417635 | -1.005832 | 0.977514 | -0.696631 | 0.310912 | 0.976352 | -0.714429 | -0.512922 |
| **3** | 0.180175 | 0.681005 | 0.032031 | -0.663867 | -0.198357 | -0.417635 | 0.898962 | 1.239897 | -0.696631 | -0.206705 | 0.976352 | -0.714429 | -0.512922 |
| **4** | 0.290464 | -1.468418 | -0.938515 | -0.663867 | 2.082050 | -0.417635 | 0.898962 | 0.583939 | 1.435481 | -0.379244 | 0.976352 | -0.714429 | -0.512922 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **298** | 0.290464 | -1.468418 | -0.938515 | 0.478391 | -0.101730 | -0.417635 | 0.898962 | -1.165281 | 1.435481 | -0.724323 | -0.649113 | -0.714429 | 1.123029 |
| **299** | -1.033002 | 0.681005 | 1.973123 | -1.234996 | 0.342756 | -0.417635 | 0.898962 | -0.771706 | -0.696631 | 0.138373 | -0.649113 | -0.714429 | 1.123029 |
| **300** | 1.503641 | 0.681005 | -0.938515 | 0.706843 | -1.029353 | 2.394438 | 0.898962 | -0.378132 | -0.696631 | 2.036303 | -0.649113 | 1.244593 | 1.123029 |
| **301** | 0.290464 | 0.681005 | -0.938515 | -0.092738 | -2.227533 | -0.417635 | 0.898962 | -1.515125 | 1.435481 | 0.138373 | -0.649113 | 0.265082 | 1.123029 |
| **302** | 0.290464 | -1.468418 | 0.032031 | -0.092738 | -0.198357 | -0.417635 | -1.005832 | 1.064975 | -0.696631 | -0.896862 | -0.649113 | 0.265082 | -0.512922 |

303 rows × 13 columns

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df\_feat, df['target'], test\_size=0.33, random\_state=101)

from sklearn.neighbors import KNeighborsClassifier

error\_rate = []

for i in range(1,40):

knn = KNeighborsClassifier(n\_neighbors=i)

knn.fit(X\_train,y\_train)

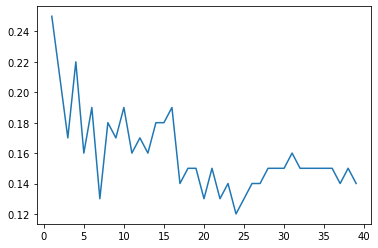
pred\_i = knn.predict(X\_test)

error\_rate.append(np.mean(pred\_i != y\_test))

plt.plot(range(1,40) , error\_rate)

**Output :**

[<matplotlib.lines.Line2D at 0x7fe086019f70>]



knn = KNeighborsClassifier(n\_neighbors = 24)

knn.fit(X\_train , y\_train)

KNeighborsClassifier(n\_neighbors=24)

pred = knn.predict(X\_test)

from sklearn.metrics import classification\_report,confusion\_matrix

print(confusion\_matrix(y\_test,pred))

**Output :**

[[38 10]

[ 2 50]]

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

**Output :**

precision recall f1-score support

0 0.95 0.79 0.86 48

1 0.83 0.96 0.89 52

accuracy 0.88 100

macro avg 0.89 0.88 0.88 100

weighted avg 0.89 0.88 0.88 100

# The accuracy of the algorithm is 88%