## **Name** - Nihalahmed Munir Barudwale

**Project name** - Prediction of Heart disease detection

**Batch** – Machine Learning With Python

**Certificate Code** – TCRIG02R28

**CODE** –

# -\*- coding: utf-8 -\*-  
*"""Interenship Project-2.ipynb  
  
Automatically generated by Colaboratory.  
  
Original file is located at  
 https://colab.research.google.com/drive/1Tt0h\_wIEU1yCGHoTOFgfRNu8xQueBYH7  
  
\*\*Name\*\* - Nihalahmed Munir Barudwale  
  
\*\*Project name\*\* - Prediction of Heart disease detection  
"""*#  
  
"""\*\*Import Required Libraries\*\*"""  
  
# Commented out IPython magic to ensure Python compatibility.  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
# %matplotlib inline  
import os  
import warnings  
warnings.filterwarnings('ignore')  
  
"""\*\*Import dataset\*\*"""  
  
dataset = pd.read\_csv("/content/drive/MyDrive/TCR Internship Project/heart.csv")  
  
dataset  
  
"""\*\*Shape of dataset\*\*"""  
  
dataset.shape  
  
"""\*\*Some Operations on dataset\*\*"""  
  
dataset.head()  
  
dataset.tail()  
  
type(dataset)  
  
dataset.info()  
  
dataset.describe()  
  
dataset.columns  
  
"""\*\*Checking total number of NA values\*\* """  
  
dataset.isna().sum()  
  
"""\*\*Checking total number of NULL values\*\* """  
  
dataset.isnull().sum()  
  
#  
  
"""\*\*Exploratory Data Analysis (EDA)\*\*  
  
\*\*Analysing the 'target' variable\*\*  
"""  
  
dataset.target.describe()  
  
dataset.target.unique()  
  
#Checking correlation between columns  
dataset.corr()["target"].abs().sort\_values(ascending=False)  
  
#This shows that most columns are moderately correlated with target, but 'fbs' is very weakly correlated.  
  
dataset.target.value\_counts()  
  
"""Patient without heart problems - labeled as 0  
  
Patient with heart problems - labeled as 1  
"""  
  
print("Percentage of patients without heart problems: "+str(round(138\*100/303,2)))  
print("Percentage of patients with heart problems: "+str(round(165\*100/303,2)))  
  
y = dataset["target"]  
sns.countplot(y)  
  
sns.distplot(dataset['target'])  
  
"""\*\*Analysing the 'sex' variable\*\*"""  
  
dataset.sex.value\_counts()  
  
sns.barplot(dataset["sex"],y)  
  
"""We notice that the 'sex' feature has 2 unique features."""  
  
sns.distplot(dataset['sex'])  
  
"""\*\*Analysing the 'cp' variable\*\*"""  
  
dataset.cp.value\_counts()  
  
sns.barplot(dataset["cp"],y)  
  
""" The CP feature has values from 0 to 3.We notice, that chest pain of '0', are much less likely to have heart problems"""  
  
sns.distplot(dataset['cp'])  
  
"""\*\*Analysing the 'age' variable\*\*"""  
  
dataset.age.value\_counts()  
  
sns.barplot(dataset["age"],y)  
  
"""Nothing special here."""  
  
sns.distplot(dataset['age'])  
  
"""\*\*Analysing the 'trestbps' variable\*\*"""  
  
dataset.trestbps.value\_counts()  
  
sns.barplot(dataset["trestbps"],y)  
  
"""Nothing special here."""  
  
sns.distplot(dataset['trestbps'])  
  
"""\*\*Analysing the 'chol' variable\*\*"""  
  
dataset.chol.value\_counts()  
  
sns.barplot(dataset["chol"],y)  
  
"""Nothing special here """  
  
sns.distplot(dataset['chol'])  
  
"""\*\*Analysing the 'fbs' variable\*\*"""  
  
dataset.fbs.value\_counts()  
  
sns.barplot(dataset["fbs"],y)  
  
"""Not much difference here."""  
  
sns.distplot(dataset['fbs'])  
  
"""\*\*Analysing the 'restecg' variable\*\*"""  
  
dataset.restecg.value\_counts()  
  
sns.barplot(dataset["restecg"],y)  
  
"""We realize that people with restecg '1' and '0' are much more likely to have a heart disease than with restecg '2'"""  
  
sns.distplot(dataset['restecg'])  
  
"""\*\*Analysing the 'exang' variable\*\*"""  
  
dataset.exang.value\_counts()  
  
sns.barplot(dataset["exang"],y)  
  
"""We notice here that people with exang=1, are much less likely to have heart problems."""  
  
sns.distplot(dataset['exang'])  
  
"""\*\*Analysing the 'slope' variable\*\*"""  
  
dataset.slope.value\_counts()  
  
sns.barplot(dataset["slope"],y)  
  
"""We observe, that Slope '2' causes heart pain much more than Slope '0' and '1'"""  
  
sns.distplot(dataset['slope'])  
  
"""\*\*Analysing the 'ca' variable\*\*"""  
  
dataset.ca.value\_counts()  
  
sns.barplot(dataset["ca"],y)  
  
"""We notice that ca=4 has large number of heart patients."""  
  
sns.distplot(dataset['ca'])  
  
"""\*\*Analysing the 'thal' variable\*\*"""  
  
dataset.thal.value\_counts()  
  
sns.barplot(dataset["thal"],y)  
  
"""thal=2 has large number of heart patients."""  
  
sns.distplot(dataset['thal'])  
  
"""\*\*Get an overview distribution of each column\*\*"""  
  
dataset.hist(figsize=(16, 20), xlabelsize=8, ylabelsize=8)  
  
sns.pairplot(dataset, hue='target')  
  
"""\*\*Correlation heatmap\*\*"""  
  
dataset.corr()  
  
f, ax = plt.subplots(figsize=(15, 10))  
sns.heatmap(dataset.corr(),annot=True,cmap='PiYG',linewidths=.5)  
  
"""\*\*Splitting the data -  
Train Test split\*\*  
"""  
  
from sklearn.model\_selection import train\_test\_split  
x = dataset.drop("target",axis=1)  
y= dataset["target"]  
  
X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(x,y,test\_size=0.20,random\_state=0)  
  
X\_train.shape  
  
X\_test.shape  
  
Y\_train.shape  
  
Y\_test.shape  
  
from sklearn.metrics import accuracy\_score  
  
"""\*\*Logistic Regression\*\*"""  
  
from sklearn.linear\_model import LogisticRegression  
model\_logistic\_reg = LogisticRegression()  
model\_logistic\_reg.fit(X\_train,Y\_train)  
Y\_pred\_logistic\_reg = model\_logistic\_reg.predict(X\_test)  
  
Y\_pred\_logistic\_reg.shape  
  
print("Predicted Values : ",Y\_pred\_logistic\_reg)  
  
Y\_test[0:10] #You can check accuracy by observing predicted results and test data.  
  
accuracy\_score\_logistic\_reg = round(accuracy\_score(Y\_pred\_logistic\_reg,Y\_test)\*100,2)  
print("The accuracy score achieved using Logistic Regression is: "+str(accuracy\_score\_logistic\_reg)+" %")  
  
"""\*\*SVM\*\*"""  
  
from sklearn import svm  
model\_svm = svm.SVC(kernel='linear')  
model\_svm.fit(X\_train, Y\_train)  
Y\_pred\_svm = model\_svm.predict(X\_test)  
  
Y\_pred\_svm.shape  
  
print("Predicted Values : ",Y\_pred\_svm)  
  
Y\_test[0:10] #You can check accuracy by observing predicted results and test data.  
  
accuracy\_score\_svm = round(accuracy\_score(Y\_pred\_svm,Y\_test)\*100,2)  
print("The accuracy score achieved using Linear SVM is: "+str(accuracy\_score\_svm)+" %")  
  
"""\*\*K Nearest Neighbors\*\*"""  
  
from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier(n\_neighbors=7)  
knn.fit(X\_train,Y\_train)  
Y\_pred\_knn=knn.predict(X\_test)  
  
Y\_pred\_knn.shape  
  
print("Predicted Values : ",Y\_pred\_knn)  
  
Y\_test[0:10] #You can check accuracy by observing predicted results and test data.  
  
accuracy\_score\_knn = round(accuracy\_score(Y\_pred\_knn,Y\_test)\*100,2)  
print("The accuracy score achieved using KNN is: "+str(accuracy\_score\_knn)+" %")  
  
"""\*\*Decision Tree\*\*"""  
  
from sklearn.tree import DecisionTreeClassifier  
max\_accuracy = 0  
for x in range(200):  
 dt = DecisionTreeClassifier(random\_state=x)  
 dt.fit(X\_train,Y\_train)  
 Y\_pred\_dt = dt.predict(X\_test)  
 current\_accuracy = round(accuracy\_score(Y\_pred\_dt,Y\_test)\*100,2)  
 if(current\_accuracy>max\_accuracy):  
 max\_accuracy = current\_accuracy  
 best\_x = x  
   
dt = DecisionTreeClassifier(random\_state=best\_x)  
dt.fit(X\_train,Y\_train)  
Y\_pred\_dt = dt.predict(X\_test)  
  
print(Y\_pred\_dt.shape)  
  
print("Predicted Values : ",Y\_pred\_dt)  
  
Y\_test[0:10] #You can check accuracy by observing predicted results and test data.  
  
accuracy\_score\_dt = round(accuracy\_score(Y\_pred\_dt,Y\_test)\*100,2)  
print("The accuracy score achieved using Decision Tree is: "+str(accuracy\_score\_dt)+" %")  
  
"""\*\*Random Forest\*\*"""  
  
from sklearn.ensemble import RandomForestClassifier  
max\_accuracy = 0  
for x in range(2000):  
 rf = RandomForestClassifier(random\_state=x)  
 rf.fit(X\_train,Y\_train)  
 Y\_pred\_rf = rf.predict(X\_test)  
 current\_accuracy = round(accuracy\_score(Y\_pred\_rf,Y\_test)\*100,2)  
 if(current\_accuracy>max\_accuracy):  
 max\_accuracy = current\_accuracy  
 best\_x = x  
   
rf = RandomForestClassifier(random\_state=best\_x)  
rf.fit(X\_train,Y\_train)  
Y\_pred\_rf = rf.predict(X\_test)  
  
Y\_pred\_rf.shape  
  
print("Predicted Values : ",Y\_pred\_rf)  
  
Y\_test[0:10] #You can check accuracy by observing predicted results and test data.  
  
accuracy\_score\_rf = round(accuracy\_score(Y\_pred\_rf,Y\_test)\*100,2)  
print("The accuracy score achieved using Random Forest is: "+str(accuracy\_score\_rf)+" %")  
  
"""\*\*Summary of accuracy scores\*\*"""  
  
all\_accuracy\_scores = [accuracy\_score\_logistic\_reg,accuracy\_score\_svm,accuracy\_score\_knn,accuracy\_score\_dt,accuracy\_score\_rf]  
algorithms\_used = ["Logistic Regression","Support Vector Machine","K-Nearest Neighbors","Decision Tree","Random Forest"]  
  
for i in range(len(algorithms\_used)):  
 print("\nThe accuracy score achieved using "+algorithms\_used[i]+" is: "+str(all\_accuracy\_scores[i])+" %")  
  
sns.set(rc={'figure.figsize':(15,8)})  
plt.xlabel("Algorithms")  
plt.ylabel("Accuracy score")  
  
sns.barplot(algorithms\_used,all\_accuracy\_scores)  
  
"""\*\*Here we can see that Random Forest is better than other algorithms.\*\*"""

**SCREENSHOTS** –





























































































