## **Name** - Nihalahmed Munir Barudwale

**Project name** - Prediction of Breast Cancer for a given patient.

**Batch** – Machine Learning With Python

**Certificate Code** – TCRIG02R28

**CODE** –

# -\*- coding: utf-8 -\*-  
*"""Interenship Project.ipynb  
  
Automatically generated by Colaboratory.  
  
Original file is located at  
 https://colab.research.google.com/drive/1sETev9W0SDcC9Du9Qeor-O\_uGQCtou\_-  
  
\*\*Name\*\* - Nihalahmed Munir Barudwale  
  
\*\*Project name\*\* - Prediction of Breast cancer for a given patient  
"""*#  
  
from google.colab import drive  
drive.mount('/content/drive')  
  
"""\*\*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')  
from scipy.stats import norm  
  
"""\*\*Import dataset\*\*"""  
  
dataset = pd.read\_csv("/content/drive/MyDrive/TCR Internship Project/data.csv")  
  
dataset  
  
"""\*\*Shape of dataset\*\*"""  
  
dataset.shape  
  
"""\*\*Some Operations on dataset\*\*"""  
  
dataset.head()  
  
dataset.tail()  
  
type(dataset)  
  
dataset.info()  
  
dataset.describe()  
  
dataset.columns  
  
dataset.isna().sum()  
  
dataset.isnull().sum()  
  
"""\*\*Data Preprocessing\*\*  
  
Deleting 'Unnamed:32' column which is of no use to us and it contains only null values.  
"""  
  
del dataset['Unnamed: 32']  
  
dataset.shape  
  
dataset.isna().sum()  
  
dataset.diagnosis.unique()  
  
"""From the results above, diagnosis is a categorical variable, because it represents a fix number of possible values (i.e, Malignant, of Benign. The machine learning algorithms wants numbers, and not strings, as their inputs so we need to convert them into numbers."""  
  
dataset['diagnosis'] = dataset['diagnosis'].apply(lambda x : '1' if x == 'M' else '0')  
dataset = dataset.set\_index('id')  
  
dataset.skew()  
  
"""The skew result show a positive (right) or negative (left) skew. Values closer to zero show less skew. From the graphs, we can see that radius\_mean, perimeter\_mean, area\_mean, concavity\_mean and concave\_points\_mean are useful in predicting cancer type due to the distinct grouping between malignant and benign cancer types in these features. We can also see that area\_worst and perimeter\_worst are also quite useful."""  
  
#  
  
"""\*\*Exploratory Data Analysis (EDA)\*\*  
  
\*\*Analysing the 'diagnosis' variable\*\*  
"""  
  
dataset.diagnosis.describe()  
  
dataset.diagnosis.unique()  
  
dataset.diagnosis.value\_counts()  
  
"""1= Malignant (Cancerous) - Present  
  
0= Benign (Not Cancerous) -Absent  
"""  
  
print("Percentage of patients which have breast cancer: "+str(round(212\*100/569,2)))  
print("Percentage of patient which does not have breast cancer: "+str(round(357\*100/569,2)))  
  
y = dataset["diagnosis"]  
sns.countplot(y)  
  
"""\*\*Get an overview distribution of each column\*\*"""  
  
dataset.hist(figsize=(16, 20), xlabelsize=8, ylabelsize=8)  
  
"""\*\*Separate columns into smaller dataframes to perform visualization\*\*"""  
  
data\_id\_diag=dataset.reindex(columns=['id', 'diagnosis'])  
data\_diag=dataset.reindex(columns=['diagnosis'])  
  
#For a merge + slice:  
data\_mean=dataset.iloc[:,1:11]  
data\_se=dataset.iloc[:,11:22]  
data\_worst=dataset.iloc[:,23:]  
  
"""\*\*Visualise distribution of data via histograms\*\*  
  
\*\*1.Histogram the "\_mean" suffix designition\*\*  
"""  
  
hist\_mean=data\_mean.hist(bins=10, figsize=(15, 10),grid=False,)  
  
#  
  
"""\*\*\*2.Histogram for the "\_se" suffix designition\*\*\*"""  
  
hist\_se=data\_se.hist(bins=10, figsize=(15, 10),grid=False,)  
  
#  
  
"""\*\*3.Histogram "\_worst" suffix designition\*\*"""  
  
hist\_worst=data\_worst.hist(bins=10, figsize=(15, 10),grid=False,)  
  
#  
  
"""\*\*Visualise distribution of data via Density plots\*\*  
  
\*\*1.Density plots "\_mean" suffix designition\*\*  
"""  
  
plt = data\_mean.plot(kind= 'density', subplots=True, layout=(4,3), sharex=False,   
 sharey=False,fontsize=12, figsize=(15,10))  
  
"""\*\*2.Density plots "\_se" suffix designition\*\*"""  
  
plt = data\_se.plot(kind= 'density', subplots=True, layout=(4,3), sharex=False,   
 sharey=False,fontsize=12, figsize=(15,10))  
  
"""\*\*3.Density plot "\_worst" suffix designition\*\*"""  
  
plt = data\_worst.plot(kind= 'kde', subplots=True, layout=(4,3), sharex=False, sharey=False,fontsize=5,   
 figsize=(15,10))  
  
"""\*\*Correlation heatmap\*\*"""  
  
dataset.corr()  
  
data\_mean.corr()  
  
import pandas as pd  
import numpy as np  
import seaborn as sns  
from matplotlib import pyplot as plt  
f, ax = plt.subplots(figsize=(15, 10))  
plt.title('Breast Cancer Feature Correlation')  
sns.heatmap(data\_mean.corr(),annot=True,cmap='PiYG',linewidths=.5)  
  
"""\*\*Splitting the data -  
Train Test split\*\*  
  
\*\*Label encoding\*\*  
"""  
  
#Here, I assign the 30 features to a NumPy array x, and transform the class labels from their original string representation (M and B) into integers.  
array = dataset.values  
x = array[:,1:31]  
y = array[:,0]  
  
from sklearn.model\_selection import train\_test\_split  
array = dataset.values  
x = array[:,1:31]  
y = array[:,0]  
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.\*\*  
  
  
---  
  
\*\*Name\*\* - Nihalahmed Munir Barudwale  
  
\*\*Project name\*\* - Prediction of Breast cancer for a given patient  
"""

**SCREENSHOTS** –





















































