PROBLEM STATEMENT - 1

NAME : P.KANISHK

REGISTER NO. : 1833020

HEART FAILURE PREDICTION

The code is implemented in Python and different classification models are applied.

To Predict Heart Failure, Here I will be using the following classification models for classification :

* SVM
* Naive Bayes
* Logistic Regression
* Random Forest
* Decision Trees
* K-NN

**1. IMPORT LIBRARIES**

I imported several libraries for the project:

1. numpy: To work with arrays
2. pandas: To work with csv files and dataframes
3. matplotlib: To create charts using pyplot, define parameters using rcParams
4. warnings: To ignore all warnings which might be showing up in the notebook due to past/future depreciation of a feature
5. train\_test\_split: To split the dataset into training and testing data
6. StandardScaler: To scale all the features, so that the Machine Learning model better adapts to the dataset

Next, I imported all the necessary Machine Learning algorithms.

FEATURES & PREDICTOR :

My **Predictor** (Y, Positive or Negative diagnosis of Heart Disease) is determined by 11 **features** (X).

2. DATA WRANGLING :

import numpy as np

import pandas as pd

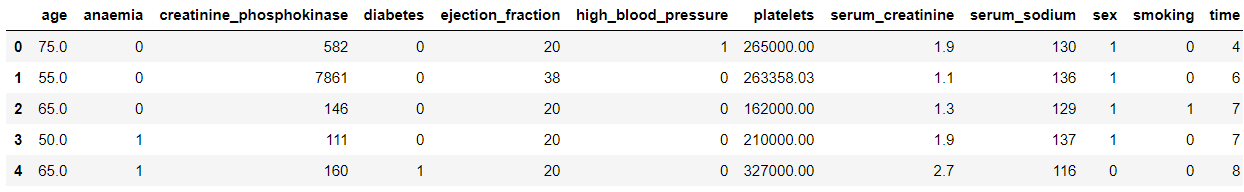
import matplotlib as plt

import seaborn as sns

import matplotlib.pyplot as plt

data = pd.read\_csv('heart\_failure\_clinical\_records\_dataset.csv')

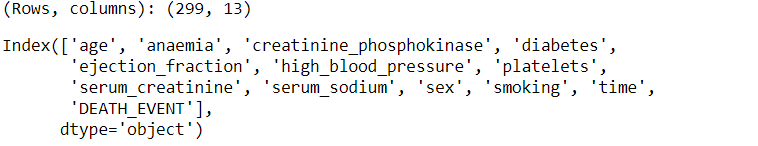
data.head(5)



Displaying Number of Rows & Columns. As well as the Column names

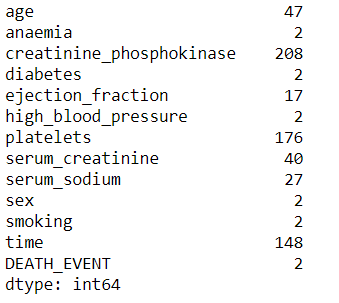
print("(Rows, columns): " + str(data.shape))

data.head(5)



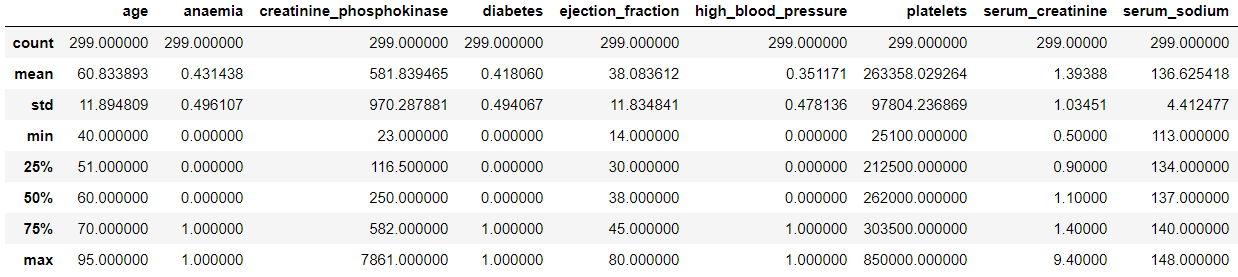
Returning the Number of Unique Values for each Variable

data.nunique(axis=0)# returns the number of unique values for each variable.



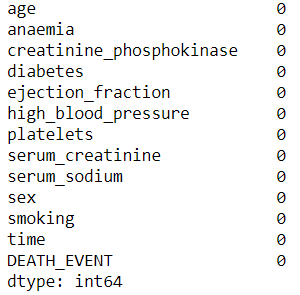
Summarizing the count, mean, standard deviation, min, and max for numeric variables

data.describe()



Displaying the Number of Missing Values for each column.

print(data.isna().sum())



**3. EXPLORATORY DATA ANALYSIS :**

**CORRELATIONS –** correlations between all variables

# calculate correlation matrix

corr = data.corr()

plt.subplots(figsize=(15,10))

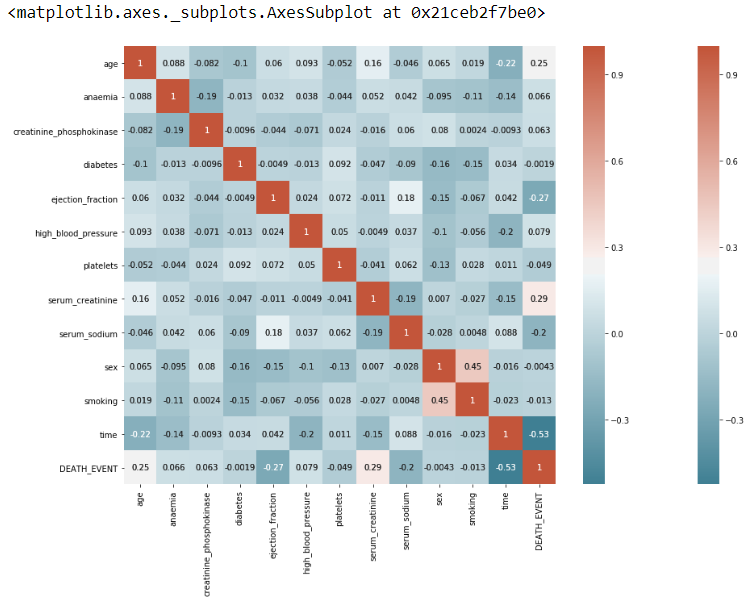
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, cmap=sns.diverging\_palette(220, 20, as\_cmap=True))

sns.heatmap(corr, xticklabels=corr.columns,

yticklabels=corr.columns,

annot=True,

cmap=sns.diverging\_palette(220, 20, as\_cmap=True))



**4. MACHINE LEARNING MODELS :**

Preparing Data for Modeling

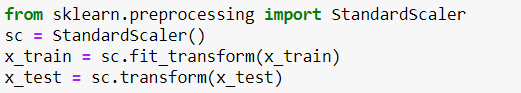
To prepare data for modeling, I’ve Assigned the 12 features to X, & the last column to our classification predictor, y



Spliting the data set into the Training set and Test set



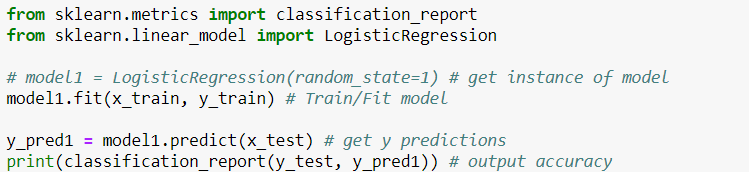
Normalization : Standardizing the data will transform the data so that its distribution will have a mean of 0 and a standard deviation of 1.

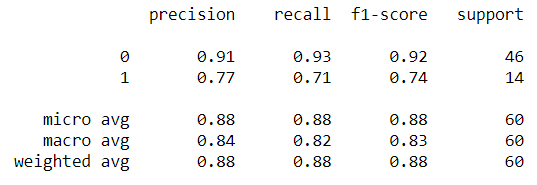


**MODELING /TRAINING :**

Now I’ve Trained various **Classification Models** on the Training set & I analyzed which **yields the highest accuracy**. I will compare the accuracy of Logistic Regression, K-NN (k-Nearest Neighbours), SVM (Support Vector Machine), Naives Bayes Classifier, Decision Trees and Random Forest.

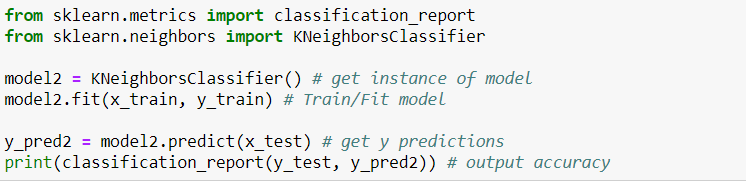
**MODEL 1: LOGISTIC REGRESSION**

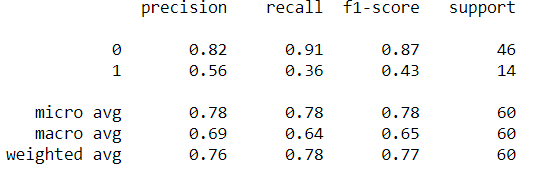




ACCURACY FOR LOGISTIC REGRESSION = 88.33%

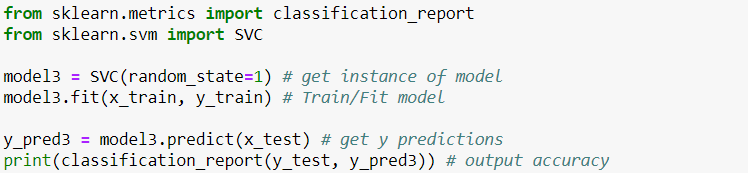
**MODEL 2 : K-NN (K-NEAREST NEIGHBORS)**

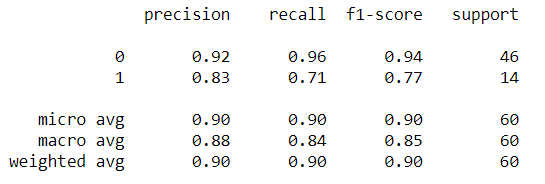




ACCURACY FOR K-NN (K-NEAREST NEIGHBORS) = 78.33%

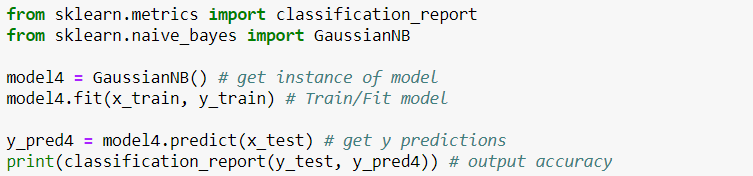
**MODEL 3 : SVM (SUPPORT VECTOR MACHINE)**

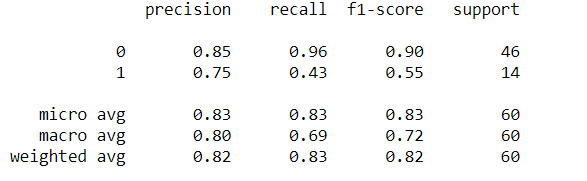




ACCURACY FOR SVM (SUPPORT VECTOR MACHINE) = 90.00%

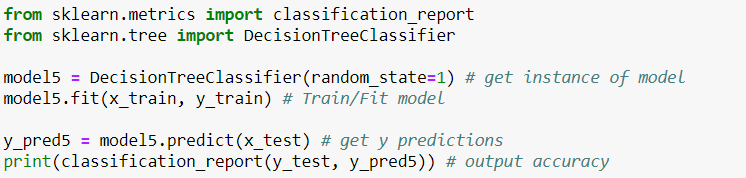
**MODEL 4 : NAIVE BAYES CLASSIFIER**

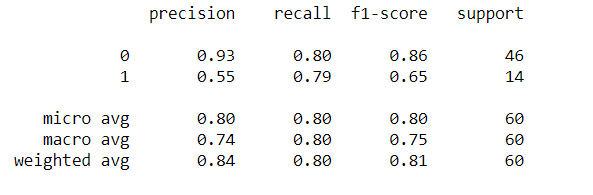




ACCURACY FOR NAIVE BAYES CLASSIFIER = 83.33%

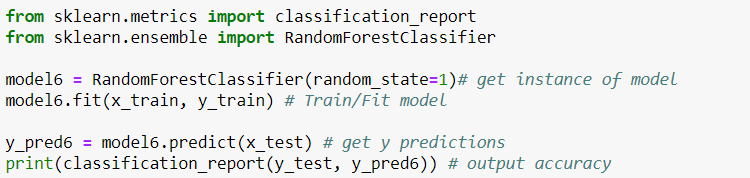
**MODEL 5 : DECISION TREES**

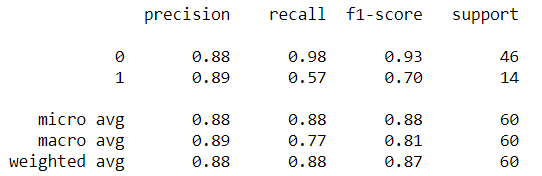




ACCURACY FOR DECISION TREES = 80.00%

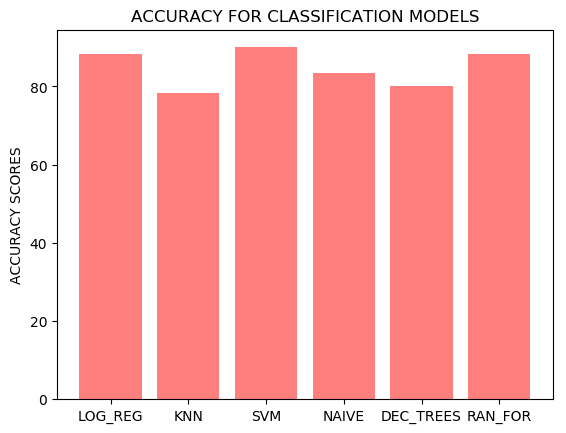
**MODEL 6 : RANDOM FOREST**





ACCURACY FOR RANDOM FOREST = 88.33%

**CONCLUSION :**

****

**Support Vector Machine yields the highest accuracy, 90%. Any accuracy above 70% is considered good, but if accuracy is**extremely**high, it may be too good to be true (an example of Over fitting). Thus, 90% is the ideal accuracy!**