**PROJECT 2**

**Heart Disease Prediction**

RIOS LEON, FERNANDO C.

EISENMANN, WILLIAM

VASHISTHA , AYUSH

JIAXING

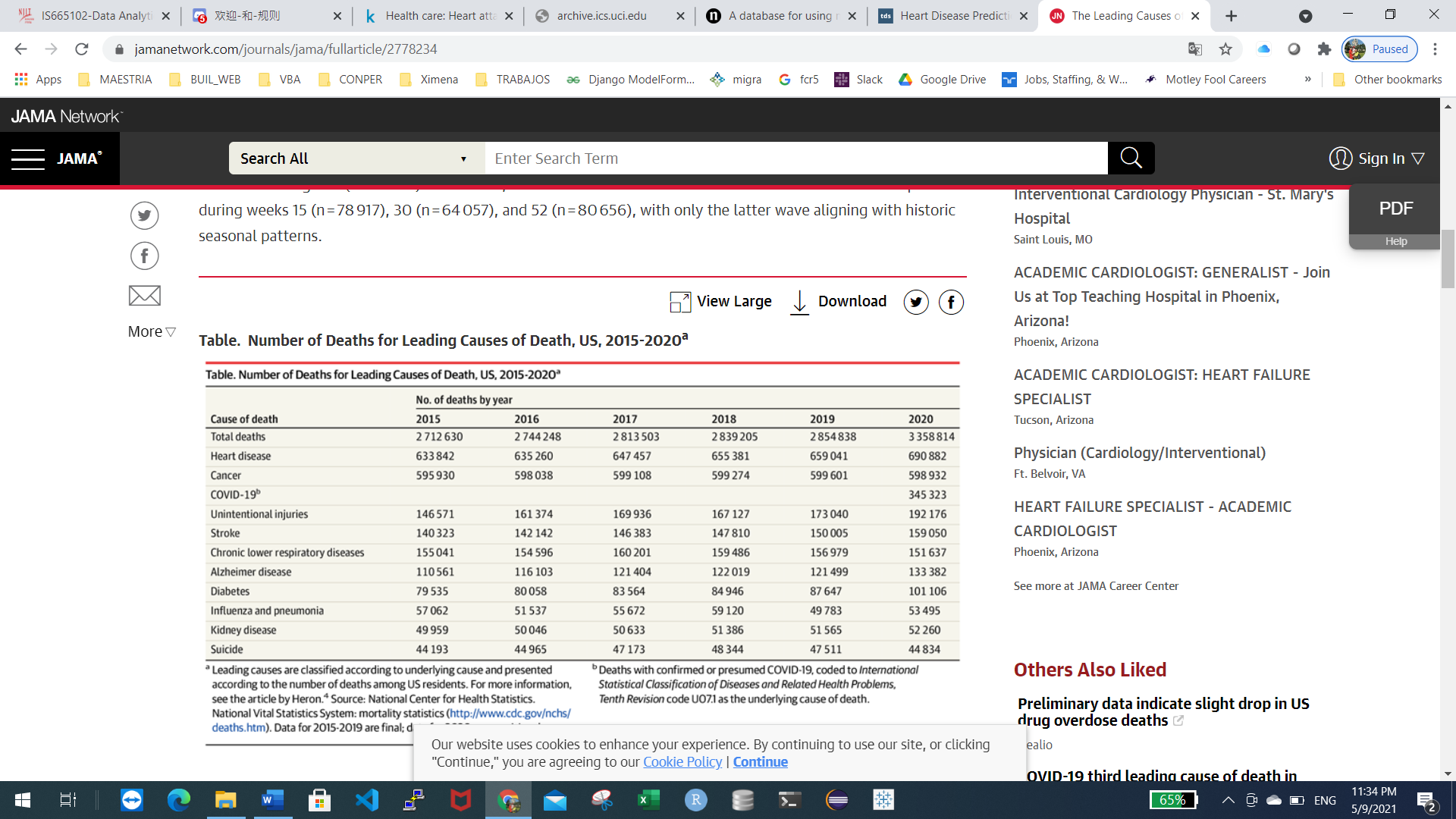
**PART I. Introduction**

Heart disease describes a range of conditions that affect your heart. Diseases under the heart disease umbrella include blood vessel diseases, such as coronary artery disease, heart rhythm problems (arrhythmias) and heart defects you’re born with (congenital heart defects), among others.

The term “heart disease” is often used interchangeably with the term “cardiovascular disease”. Cardiovascular disease generally refers to conditions that involve narrowed or blocked blood vessels that can lead to a heart attack, chest pain (angina) or stroke. Other heart conditions, such as those that affect your heart’s muscle, valves or rhythm, also are considered forms of heart disease.

Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of clinical data analysis. The amount of data in the healthcare industry is huge. Data mining turns the large collection of raw healthcare data into information that can help to make informed decisions and predictions.

According to a news article, heart disease proves to be the leading cause of death for both women and men. The article states the following :



About 650,000 people die of heart disease in the United States every year–that’s 1 in every 5 deaths.

Coronary Heart Disease(CHD) is the most common type of heart disease, killing over 370,000 people annually.

Every year about 735,000 Americans have a heart attack. Of these, 525,000 are a first heart attack and 210,000 happen in people who have already had a heart attack.

This makes heart disease a major concern to be dealt with. But it is difficult to identify heart disease because of several contributory risk factors such as diabetes, high blood pressure, high cholesterol, abnormal pulse rate, and many other factors. Due to such constraints, scientists have turned towards modern approaches like Data Mining and Machine Learning for predicting the disease.

Machine learning (ML) proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry.

In this project, we will be applying Machine Learning approaches(and eventually comparing them) for classifying whether a person is suffering from heart disease or not, using one of the most used dataset — Cleveland Heart Disease dataset from the UCI Repository.

**PART II. Data**

(to be completed as a team): Summary statistics of the data shall be presented here, including general measurements such as number of records, number of features, the label (if you have any), mean, standard deviation, etc.

The dataset consists of 303 individuals data. There are 14 columns in the dataset (features), which are described below.

* **Age (age):** displays the age of the individual.
* **Sex (sex):** displays the gender of the individual using the following format :

1 = male ; 0 = female

* **Chest-pain type (cp):** displays the type of chest-pain experienced by the individual using the following format :

0 = typical angina; 1 = atypical angina; 2 = non — anginal pain ; 3 = asymptotic

* **Resting Blood Pressure (trestbps):** displays the resting blood pressure value of an individual in mmHg (unit)
* **Serum Cholestrol (chol):** displays the serum cholesterol in mg/dl (unit)
* **Fasting Blood Sugar (fbs):** compares the fasting blood sugar value of an individual with 120mg/dl.

If fasting blood sugar > 120mg/dl then : 1 (true) else : 0 (false)

* **Resting ECG** **(restecg):** displays resting electrocardiographic results

0 = normal; 1 = having ST-T wave abnormality; 2 = left ventricular hyperthrophy

* **Max heart rate achieved** **(thalach)**: displays the max heart rate achieved by an individual.
* **Exercise induced angina (exang):**

1 = yes; 0 = no

* **ST depression induced by exercise relative to rest (oldpeak):** displays the value which is an integer or float.
* **Peak exercise ST segment (slope):**

1 = upsloping; 2 = flat; 3 = downsloping

* **Number of major vessels (0–3) colored by flourosopy (ca):** displays the value as integer or float.
* **Thal (thal):** displays the thalassemia :

**LABEL**

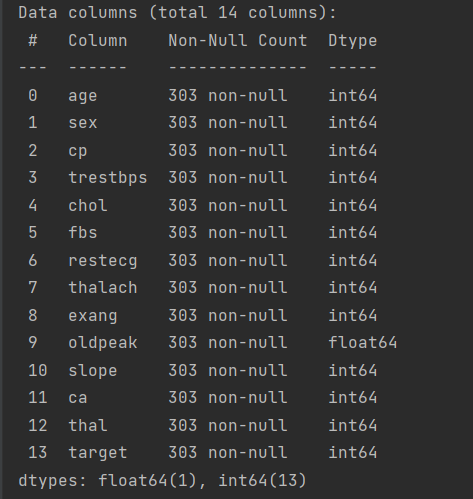
* **Diagnosis of heart disease (target):** Displays whether the individual is suffering from heart disease or not :

0 **=** absence**;** 1= present.

**DATA PREPROCESSING USING PYTHON**

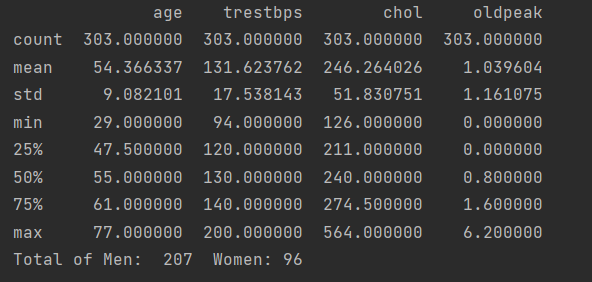
List of Columns : ***data.info()***

Total 14 columns, the target included. All features contain data not nulls.



**Summary Statistics**

***print(data[["age", "trestbps", "chol", "oldpeak"]].describe())***

****

**Number of records:** 303

**Age min:**29 - max: 77

**The resting blood pressure:** min: 94 - max: 200

**Cholesterol:** min: 51 - 564

**Depression induced by exercise relative to rest (oldpeak):** min: 0 - max: 6.2

**Number of men and women**

***n = data["sex"].value\_counts()***

***print("Total of Men: ", n[1], " Women:" , n[0])***



**Correlation**

***# Function to Generate Correlation Heatmap***

***def correlation\_heatmap(df):***

***correlations = df.corr()***

***fig, ax = plt.subplots(figsize=(10, 10))***

***cmap = sns.diverging\_palette(230, 20, as\_cmap=True)***

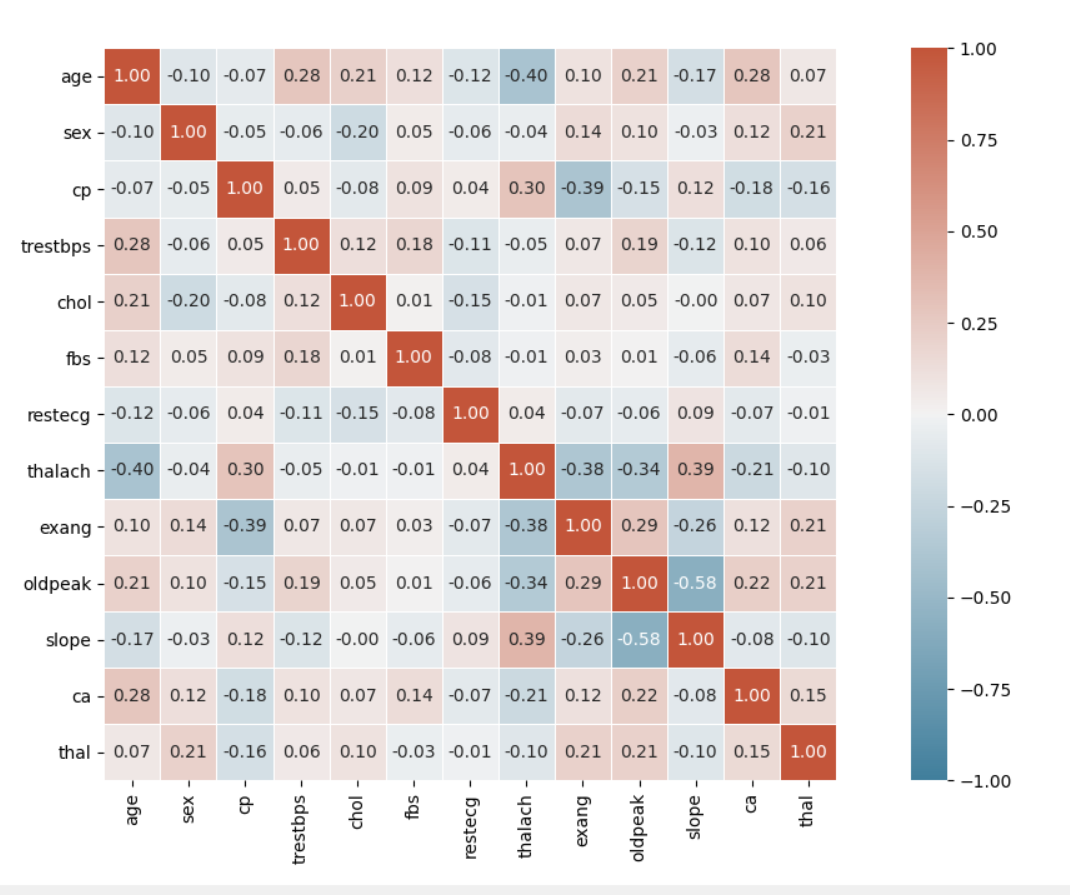
***sns.heatmap(correlations, vmax=1.0, vmin=-1, center=0, fmt='.2f',***

***square=True, linewidths=.5, annot=True, cmap=cmap)***

***plt.show()***

***# Create Heatmap of correlation***

***correlation\_heatmap(data)***

****

The attributes of the data hasn’t high correlation, so there is no chance that the performance of the model will be impacted by multicollinearity.

**Distribution of Age**

***# Creating histogram***

***fig, ax = plt.subplots(1, 1)***

***ax.hist(data['age'], bins)***

***plt.title("Distribution by Age")***

***plt.xlabel("Age")***

***plt.ylabel("Frequency")***

***# Printing Values on the bars.***

***rects = ax.patches***

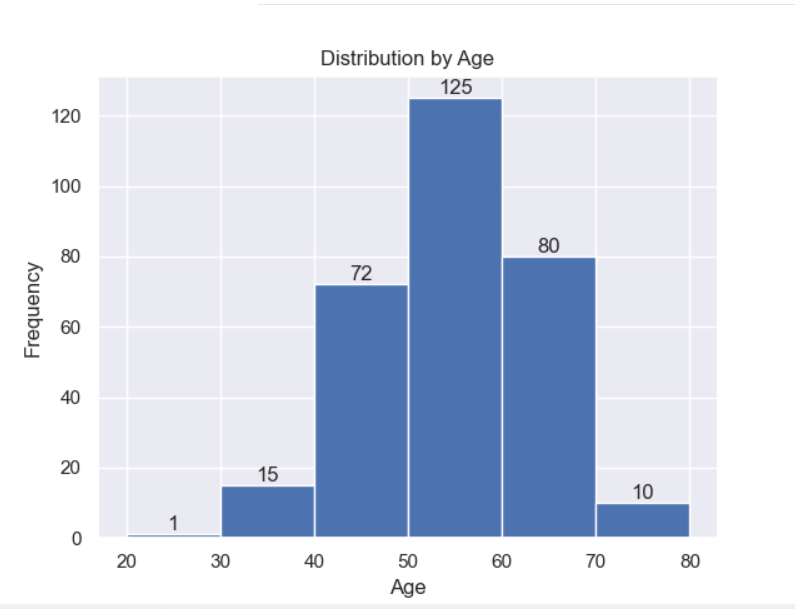
***for rect in rects:***

***height = rect.get\_height()***

***ax.text(rect.get\_x() + rect.get\_width() / 2, height + 0.01, int(height),***

***ha='center', va='bottom')***

***plt.show()***

****

In the graphic shown above we can see that age has a distribution normal.

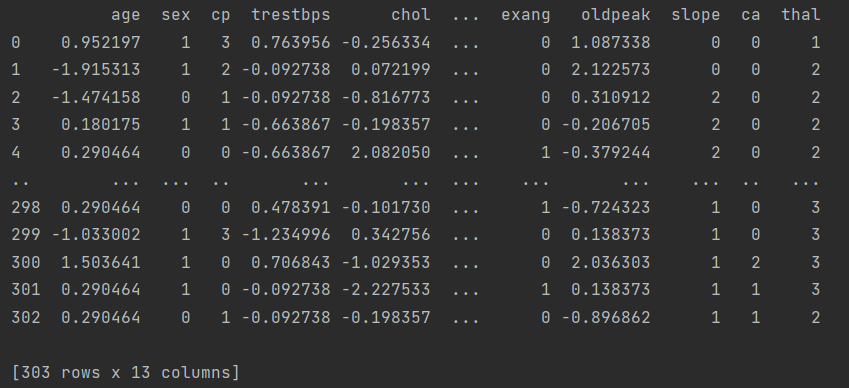
**Features onto the same scale**

Standardization.- Using standardization, we center the feature columns at mean zero with standard deviation 1 so that the feature columns have the same parameters as a standard normal distribution (zero mean and unit variance), which makes it easier to learn the weights.

***standardScaler = StandardScaler()***

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

***X[columns\_to\_scale] = standardScaler.fit\_transform(X[columns\_to\_scale])***

******

**Partitioning dataset into separate training and test datasets**

***X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=0)***

We used the train\_test\_split function to randomly split X and y into separate training and test datasets. By setting test\_size=0.3, we assigned 30 percent of the examples to X\_test and y\_test, and the remaining 70 percent of the examples were assigned to X\_train and y\_train, respectively. The parameter stratify ensures that both training and test datasets have the same class proportions as the original dataset.

**PART III. Individual Modeling by each**

In part three, each team member will choose ONE data mining algorithm to work on the data set and report the result. Please report the type of algorithm chosen, a screenshot of your workflow and screenshots of the parameter setup if you are using RapidMiner, or your code if you are using python, R, or any other language. If it is a supervised learning problem, please include the confusion matrix and discuss your findings and performance. If you are working on a unsupervised problem such as clustering or association rules, please present the clustering or rules that you found and discuss the implications. Each team member will be evaluated based on his / her performance.

**Fernando C Rios ( Support Vector Machine SVM)**

#Function for storing model scores using various kernels

svc\_scores = []

kernel\_type = ['linear', 'poly', 'rbf', 'sigmoid']

for type in kernel\_type:

svc\_classifier = SVC(kernel = type)

svc\_classifier.fit(X\_train, y\_train)

svc\_scores.append(svc\_classifier.score(X\_test, y\_test))

#Plotting the accuracy

for i in range(len(kernel\_type)):

label = round(svc\_scores[i], 5)

plt.text(i, svc\_scores[i], label)

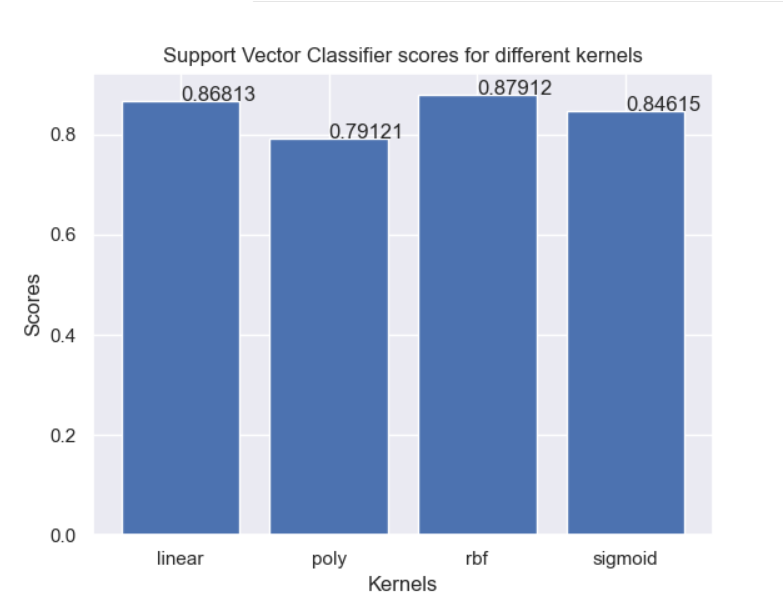
plt.xlabel('Kernels')

plt.ylabel('Scores')

plt.title('Support Vector Classifier scores for different kernels')

plt.bar(kernel\_type, svc\_scores)

plt.show()



#Training the model on 'rbf' Kernel

svc = SVC(kernel='rbf')

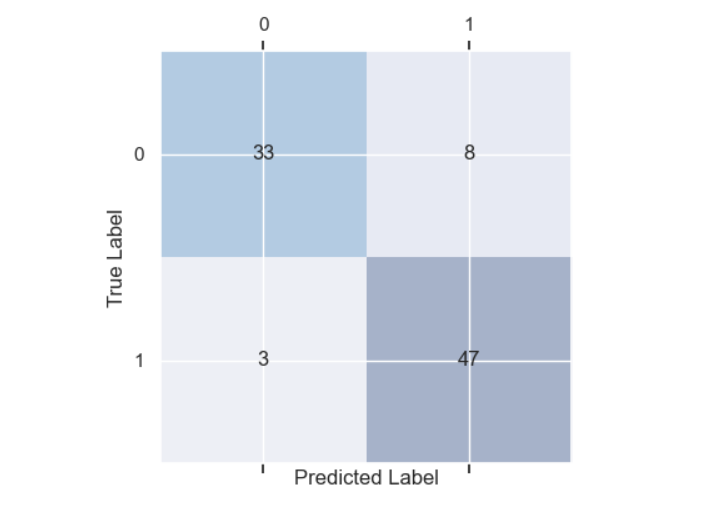
svc.fit(X\_train, y\_train)

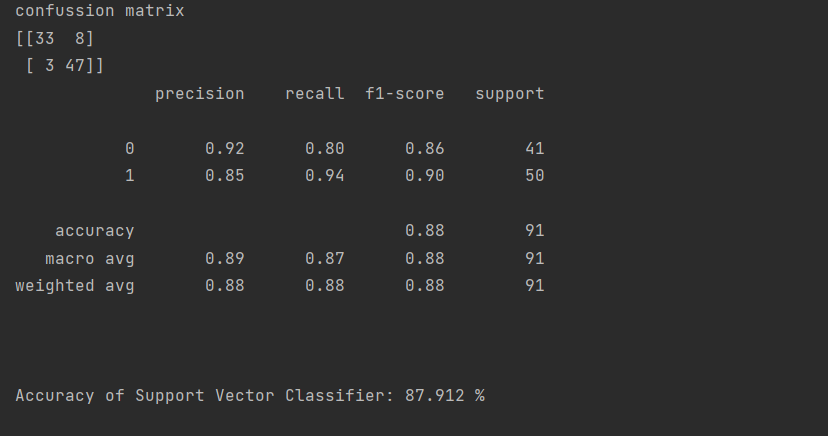
svc\_predicted = svc.predict(X\_test)

svc\_conf\_matrix = confusion\_matrix(y\_test, svc\_predicted)

svc\_acc\_score = accuracy\_score(y\_test, svc\_predicted)

**Confusion Matrix**





This model correctly classified 33 of the 41 examples that belong to class “0” absence (TN) and 47 examples that belong to class 1 present (TP), respectively. However, the model also incorrectly misclassified 3 examples from class “1” as class “0”, and it predicted that 8 examples belong to class “0” although it belongs to class “1”.

**William (Random Forest)**

from sklearn.ensemble import RandomForestClassifier

model=RandomForestClassifier(n\_estimators=500,random\_state=1)

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

rfpred=model.predict(X\_test)

RF\_conf\_matrix = confusion\_matrix(y\_test, rfpred)

rf\_acc\_score = accuracy\_score(y\_test, rfpred)

#Printing the confusion matrix and accuracy scores

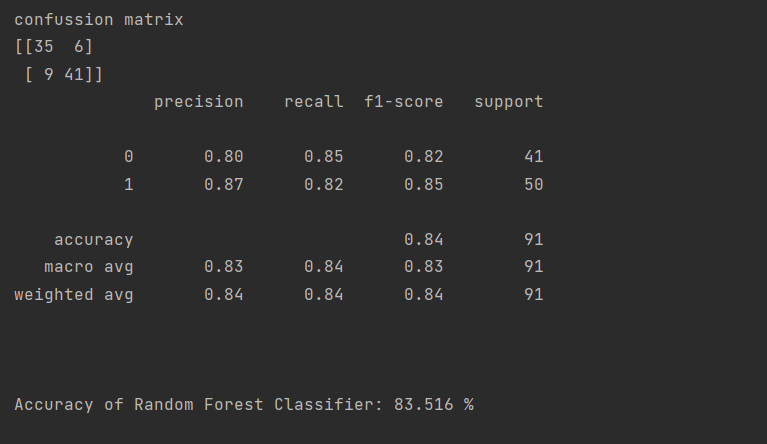
print("confusion matrix")

print(RF\_conf\_matrix)

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

print("\n")

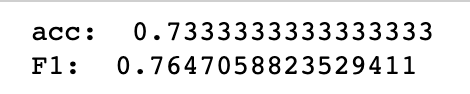
print("Accuracy of Random Forest Classifier: {:.3f}".format(rf\_acc\_score\*100),'%\n')



This model correctly classified 35 of the 41 examples that belong to class “0” absence (TN) and 41 examples that belong to class 1 present (TP), respectively. However, the model also incorrectly misclassified six examples from class “1” as class “0”, and it predicted that 9 examples belong to class “0” althought it belongs to class “1”.

**Jiaxing: (KNN) , cluster and predict the risk by training knn with data**

The code and explanation are attached in knn.ipynb



**Ayush Vashistha - (Logistic Regression )**

#CODE -

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(10)

df.describe()

df.isnull().sum()

from sklearn.metrics import classification\_report

from sklearn.linear\_model import LogisticRegression

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

from sklearn import preprocessing

from sklearn import metrics

from sklearn.metrics import accuracy\_score

X = df.drop(["target"],1)

y = df["target"]

standardized\_X = preprocessing.scale(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(standardized\_X,y,test\_size=0.3,random\_state=0)

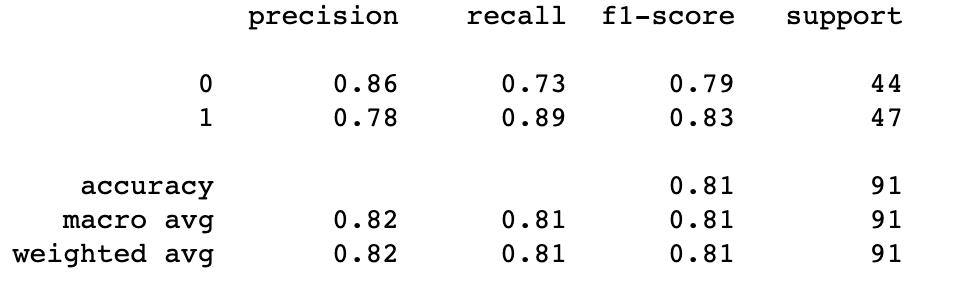
logmodel = LogisticRegression()

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

predictions = logmodel.predict(X\_test)

from sklearn.metrics import classification\_report

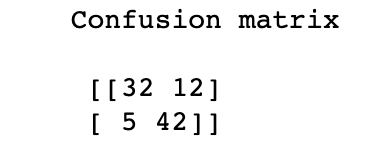
print(classification\_report(y\_test, predictions))



from sklearn.metrics import confusion\_matrix

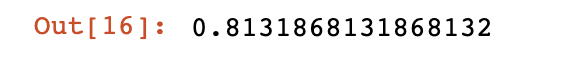
cm = confusion\_matrix(y\_test, predictions)

print('Confusion matrix\n\n', cm)



score = accuracy\_score(y\_test, predictions)

score



probs\_y=logmodel.predict\_proba(X\_test)

fpr, tpr, thresholds = metrics.roc\_curve(y\_test,probs\_y[:,1])

plt.plot(fpr, tpr)

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.0])

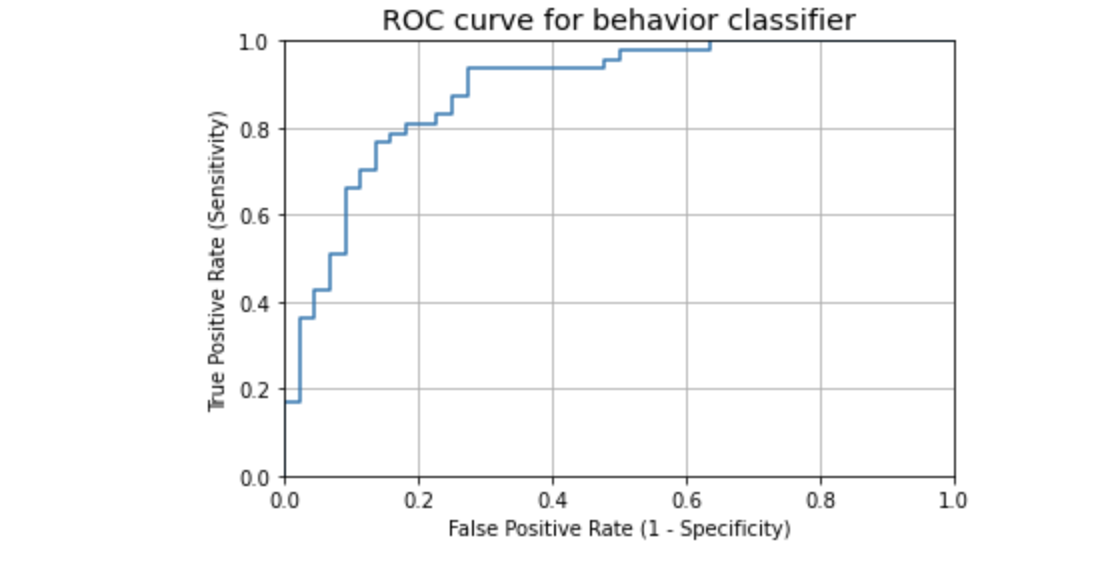
plt.rcParams['font.size'] = 12

plt.title('ROC curve for behavior classifier')

plt.xlabel('False Positive Rate (1 - Specificity)')

plt.ylabel('True Positive Rate (Sensitivity)')

plt.grid(True)



This model correctly classified 32 examples that belong to class (TN) and 42 examples that belong to class (TP), respectively. However, the model also incorrectly misclassified 12 examples from class “0” as class “1” (FP) , and it predicted that 5 examples to class “0” although it belongs to class “1 (FN)”.

TN=32

TP=42

FP=12

FN=5

**PART IV. Conclusion** (to be completed as a team): This is where you put everyone’s model together and compare their results. Please discuss:

* **Which evaluation metrics should be used (Recall or precision) and why**

Recall is preferred in this situation, because of its ability to respect false negatives. If this was any other case, precision would be a measure, but in relation to healthcare, we needed a measuring algorithm that will take into account false negatives. This is relevant even today's terms with Covid. False negatives must be minimized.

In our model selected SVM we have

* **Which algorithm works best, at what parameter setup**

After, we applied the algorithms: Support Vector Machine (SVM), Random Forest, KNN and Logistic Regression, we found the follow results:

SVM: Accuracy: 87.912 %

Random Forest: Accuracy: 83.516 %

Logistic Regression Accuracy: 81.32 %

KNN: Accuracy: 73.33%

According to the results shown above, SVM has the best prediction accuracy 87.912 %.

To achieve this value, we revised all the kernels that support this algorithm ('linear', 'poly', 'rbf', 'sigmoid') and we found that the kernel = ‘rbf’ gives the best result, then we used it to predict the values.

In addition, we prefer this algorithm because the number of False Negatives is lowest, just 3 examples.

* **What does the finding tell you?**

With respect to the dataset we found that is a small dataset, it has just 303 examples, 13 features and 1 target feature.

We found that the dataset hasn’t null or empty values.

The correlation matrix that we did show us that the attributes of the data hasn’t high correlation, so there is no chance that the performance of the model will be impacted by multicollinearity.

The conversion of features onto the same scale helped to get a better prediction.

SVM algorithm applied with parameter Kernel = ‘rbf’ improved the performance of the model.

* **Can you take action to improve the performance and solve a problem based on the findings? Do you have any recommended action?**

**K-Fold Cross Validation.-** As there is enough data to train the model, just (303 examples), if we remove a part of it for validation poses a problem of underfitting. By reducing the training data, we risk losing important patterns/ trends in the data set, which in turn increases error induced by bias. So, what we require is a method that provides ample data for training the model and also leaves ample data for validation. K Fold cross validation does exactly that. Then, Cross Validation could improve the model.