NAME: SYED MUZZAMIL WASEEM

SID: 11067

CID: 113083

ASSIGNMENT # 02 (MACHINE LEARNING)

PART - A (SVM)

1. Why are SVMs fast?

   Support Vector Machines are fast because they use a clever trick called the "kernel trick." This trick lets them work in a space with lots of dimensions without calculating the exact positions of data in that space. Instead of doing that, they use special functions (kernels) that measure how close data points are to each other in this space. These measurements are like finding the positions, but it's faster and needs less memory.

   SVM are also fast because they only care about a few of the training points, which they call "support vectors." They use these support vectors to make decisions about new data. Since they don't need all the training data, they work quicker and save memory space too.

2. Why are SVMs often more accurate than logistic regression?

   Support Vector Machines are often more accurate than logistic regression because they focus on creating a bigger gap between the closest points of different classes. This gap makes the decision boundary clearer. On the other hand, logistic regression aims to figure out the likelihood of a point belonging to a certain class. SVM are also more decisive in their predictions, but we can adjust them to give probability scores like logistic regression does.

   Additionally, SVM work well in situations where the data is not easily separated in a straight line. They use a technique called the "kernel trick" to work in a different space where separation might be easier. This trick makes SVM faster than logistic regression when dealing with complex data.

   SVM focuses on creating a bigger gap between classes, are more determined in their predictions, and can handle complex data patterns efficiently using the kernel trick, making them often more accurate in various scenarios compared to logistic regression.

3. True or False? A support vector could be inside the margin.

   Support vectors are the data points that sit right on the lines that define the gap between different groups in a Support Vector Machine. They're like the key points that help decide where to draw the lines. Points that are inside this gap are not support vectors because they don't directly help set the lines.

So, the statement "A support vector could be inside the margin" is false. Support vectors are the ones right on the lines, not inside the gap.

4. Explain the slack variable in the below figure 2.

In the SVM with soft margin, slack variables are introduced to allow some misclassification of training examples. The optimization problem is modified to minimize the sum of the slack variables subject to the constraints that the classification margin is at least 1 for each training example. The slack variables are positive and represent the degree of misclassification of each training example. The parameter C controls the trade-off between maximizing the margin and minimizing the degree of misclassification.
In this case, ξ = [ξ1, ξ2, …, ξN] is a set of slack variables. For the variables which are in the safety margin, then ξi = 0 (x_1, x_2). For the variables which are not located in the safety margin but still on the right side of their class, then 0 < ξi < 1 (x_3). For the variables which are located on the wrong side of their class, then ξi > 1 (x_4, x_5).

This is the link of that research paper where I have studied about slack variables in detail: https://www.researchgate.net/publication/327015448_AUTOMATIC_HEART_DISEASE_PREDICTION_USING_FEATURE_SELECTION_AND_DATA_MINING_TECHNIQUE.

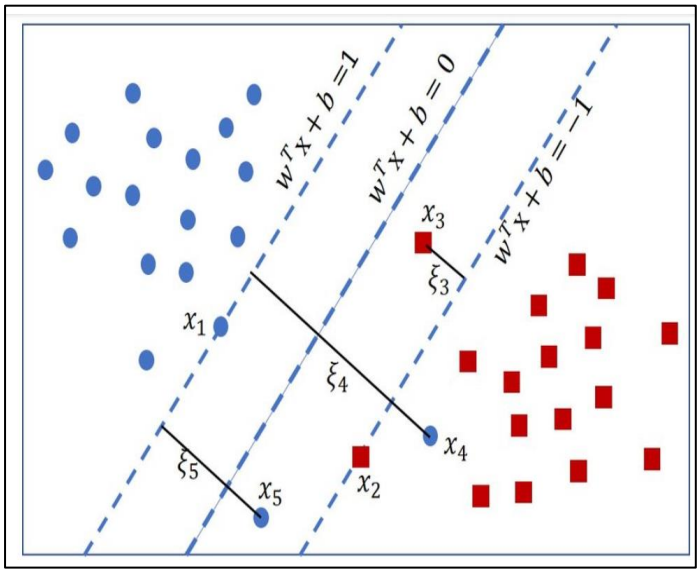


Figure 2: slack in svm

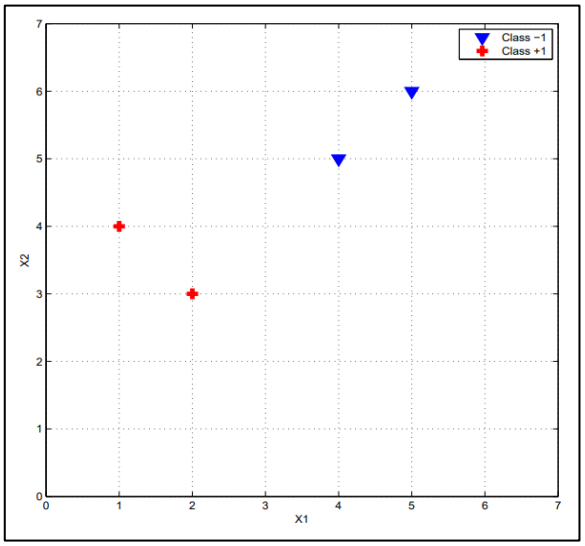


Figure 1: support vectors

5. Explain kernel trick and support vectors in SVM.

Kernel Trick:

The kernel trick is a smart method used by Support Vector Machines to handle complex data. Imagine you have data that's not easy to separate with a straight line or a plane. The kernel trick allows SVM to deal with this by transforming the data into a higher-dimensional space without explicitly calculating all the new coordinates.

Working Flow:

i. Higher-Dimensional Space: Think of the kernel trick as a magic lens that takes your data and puts it in a higher-dimensional space where it's easier to separate.

ii. Inner Products: Instead of finding the actual coordinates in this higher-dimensional space, the trick uses kernel functions to find the inner products between pairs of data points.

iii. Equivalence: These inner products are like secret hints about where the data points would be in the higher-dimensional space, without having to know the exact coordinates.

iv. Cheaper Computation: Calculating inner products is computationally cheaper than figuring out the coordinates explicitly, especially when the new space is high-dimensional.

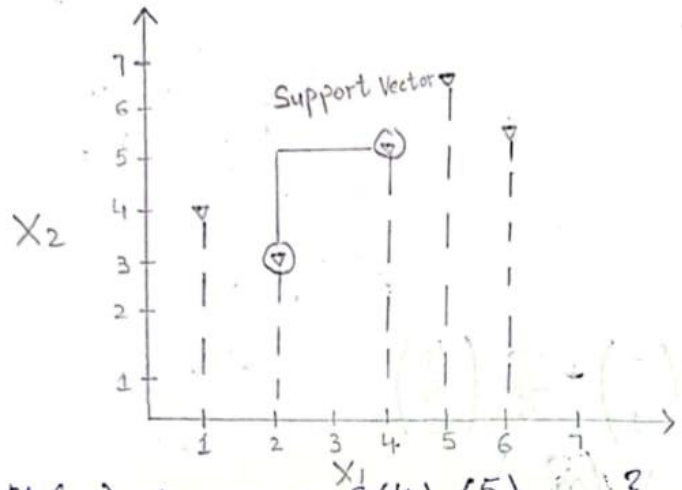
So, the kernel trick lets SVM play in a fancy space without revealing all the details, making it easier to find good separation lines or planes for complicated data.

6. Support vector machines learn a decision boundary leading to the largest margin from both classes. You are training SVM on a tiny dataset with 4 points shown in Figure 1. This dataset consists of two examples with class label -1 (denoted with plus), and two examples with class label +1 (denoted with triangles).
   i. Find the weight vector w and bias b. What's the equation corresponding to the decision boundary?

NAME: SYED MUZZAMIL WASEEM
CID: 113083
SID: 11067

ASSIGNMENT # 02 MACHINE LEARNING

ANSWER NO 6(a):



⇒ diamond (+1) class points: $\left\{\begin{pmatrix}4\\5\end{pmatrix}, \begin{pmatrix}5\\6\end{pmatrix}\right\}$

⇒ Triangle (-1) class points: $\left\{\begin{pmatrix}1\\4\end{pmatrix}, \begin{pmatrix}2\\3\end{pmatrix}\right\}$

⇒ Support vectors are: $\left\{\begin{pmatrix}2\\3\end{pmatrix}, \begin{pmatrix}4\\5\end{pmatrix}\right\}$

$S_1 = \begin{pmatrix}2\\3\end{pmatrix}$, $\overline{S_1}$ (-1 side) $= \begin{pmatrix}2\\3\\1\end{pmatrix}$ (1 ← bias), $S_2 = \begin{pmatrix}4\\5\end{pmatrix}$ $\overline{S_2}$ (+1 side) $= \begin{pmatrix}4\\5\\1\end{pmatrix}$ ← bias

$\Rightarrow \alpha_1 \overline{S_1}\cdot\overline{S_2} + \alpha_2 \overline{S_1}\cdot\overline{S_2} = -1$

$\Rightarrow \alpha_1 \overline{S_2}\cdot\overline{S_1} + \alpha_2 \overline{S_2}\cdot\overline{S_2} = +1$

$\Rightarrow \alpha_1 \begin{pmatrix}2\\3\\1\end{pmatrix}\cdot\begin{pmatrix}2\\3\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}2\\3\\1\end{pmatrix}\cdot\begin{pmatrix}4\\5\\1\end{pmatrix} = -1$

$\Rightarrow \alpha_1 (4+9+1) + \alpha_2 (8+15+1) = -1$

$\Rightarrow 14\alpha_1 + 24\alpha_2 = -1 \longrightarrow (A)$

$\alpha_1 \begin{pmatrix}4\\5\\1\end{pmatrix}\cdot\begin{pmatrix}2\\3\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}4\\5\\1\end{pmatrix}\cdot\begin{pmatrix}4\\5\\1\end{pmatrix} = 1$

$\alpha_1 (8+15+1) + \alpha_2 (16+25+1) = 1$

$24\alpha_1 + 42\alpha_2 = 1$

$\Rightarrow 24\alpha_1 + 42\alpha_2 = 1 \times 4$

$14\alpha_1 + 24\alpha_2 = -1 \times 7$

$\Rightarrow 26\alpha_1 + 168\alpha_2 = 4 \longrightarrow (1)$

$98\alpha_1 + 168\alpha_2 = -7 \longrightarrow (2)$

From eq ① & eq ② we get:

$\Rightarrow 2X_1 = -11$

$$\boxed{X_1 = -\frac{11}{2}}$$

Put X_1 in eq (A) we get:

$\Rightarrow 14\left(-\frac{11}{2}\right) + 24X_2 = -1$

$\Rightarrow 24X_2 = 76$

$\Rightarrow X_2 = \frac{76}{24}$

$\Rightarrow \boxed{X_2 = \frac{19}{6}}$

$$\overline{W} = \sum_i d_i \overline{S_i} = -\frac{11}{2}\begin{pmatrix}2\\3\\1\end{pmatrix} + \frac{19}{6}\begin{pmatrix}4\\5\\1\end{pmatrix}$$

$$= \begin{pmatrix}-11\\-33/2\\-11/2\end{pmatrix} + \begin{pmatrix}38/4\\95/6\\19/6\end{pmatrix}$$

$$= \begin{pmatrix}-11+38/3\\-33/2+95/6\\-11/2+19/6\end{pmatrix}$$

$$\overline{W} = \begin{pmatrix}5/3\\-4/6\\-14/6\end{pmatrix} = \begin{pmatrix}5/3\\-2/3\\-7/3\end{pmatrix}$$

$$\boxed{W = \begin{pmatrix}5/3\\-2/3\end{pmatrix},\ b = -7/3}$$

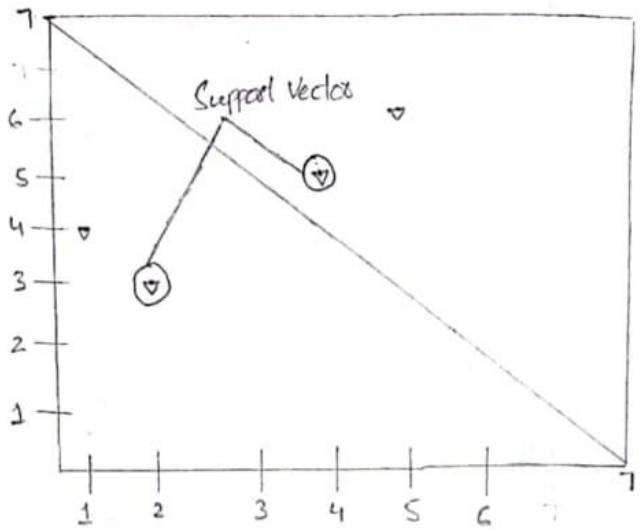
Weight Vector Bias

$$y = wx + b = \begin{pmatrix}5/3\\-2/3\end{pmatrix}\begin{pmatrix}X_1\\X_2\end{pmatrix} + b$$

Weight vector value and bias value depend on the support vectors not on the other points.

ii. Circle the support vectors and draw the decision boundary.

ANSWER NO 6(b):



PART - B

1. Implement DT and RF on any dataset with at least 10 feature.

I've already applied Decision Trees and Random Forests in my project, focusing on heart disease prediction. These techniques help make predictions based on data. The dataset I used has at least 12 features that describe heart health. After implementing DT and RF, I measured their performance using key metrics.

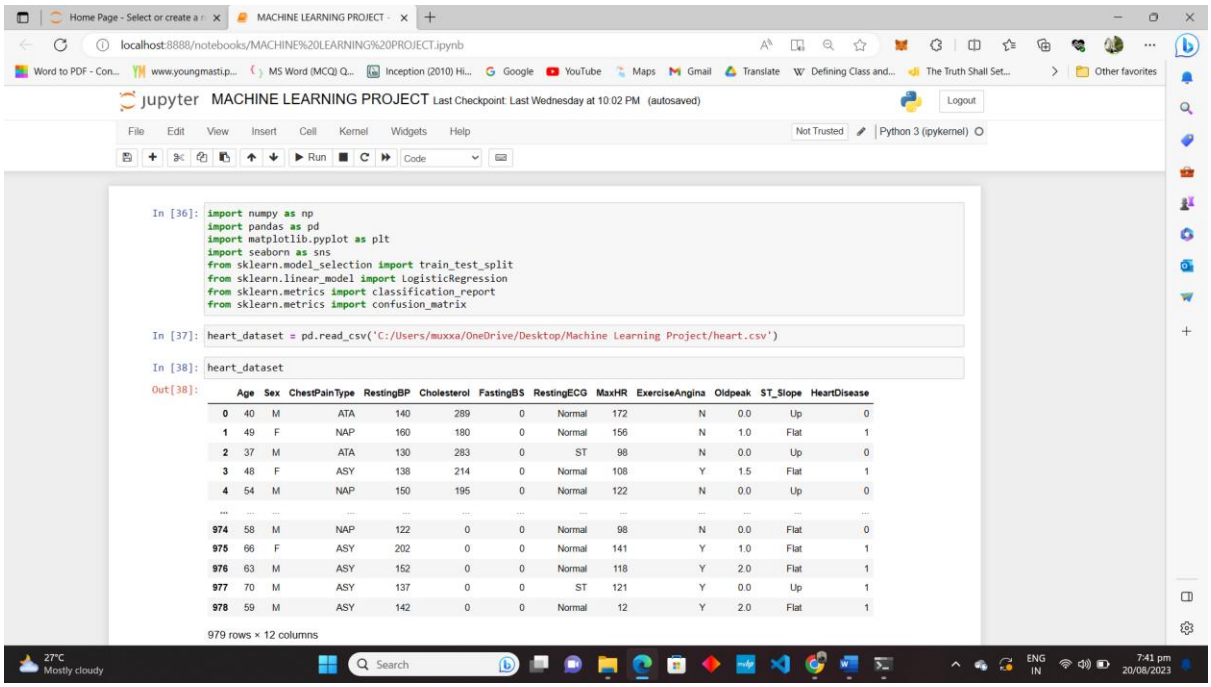
Decision Tree

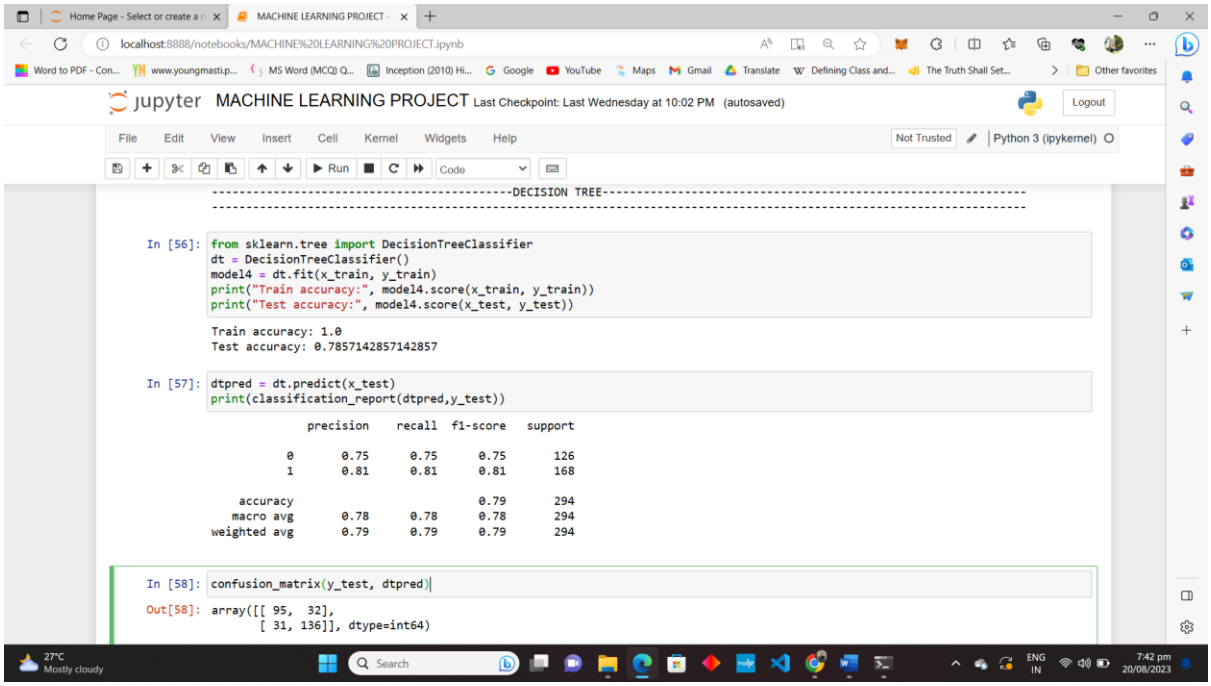
CODE:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report
```

```
from sklearn.metrics import confusion_matrix

heart_dataset = pd.read_csv('C:/Users/muxxa/OneDrive/Desktop/Machine Learning Project/heart.csv')

heart_dataset

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()

model4 = dt.fit(x_train, y_train)

print("Train accuracy:", model4.score(x_train, y_train))

print("Test accuracy:", model4.score(x_test, y_test))

dtpred = dt.predict(x_test)

print(classification_report(dtpred,y_test))

confusion_matrix(y_test, dtpred)
```

CODE + OUTPUTS:





RANDOM FOREST

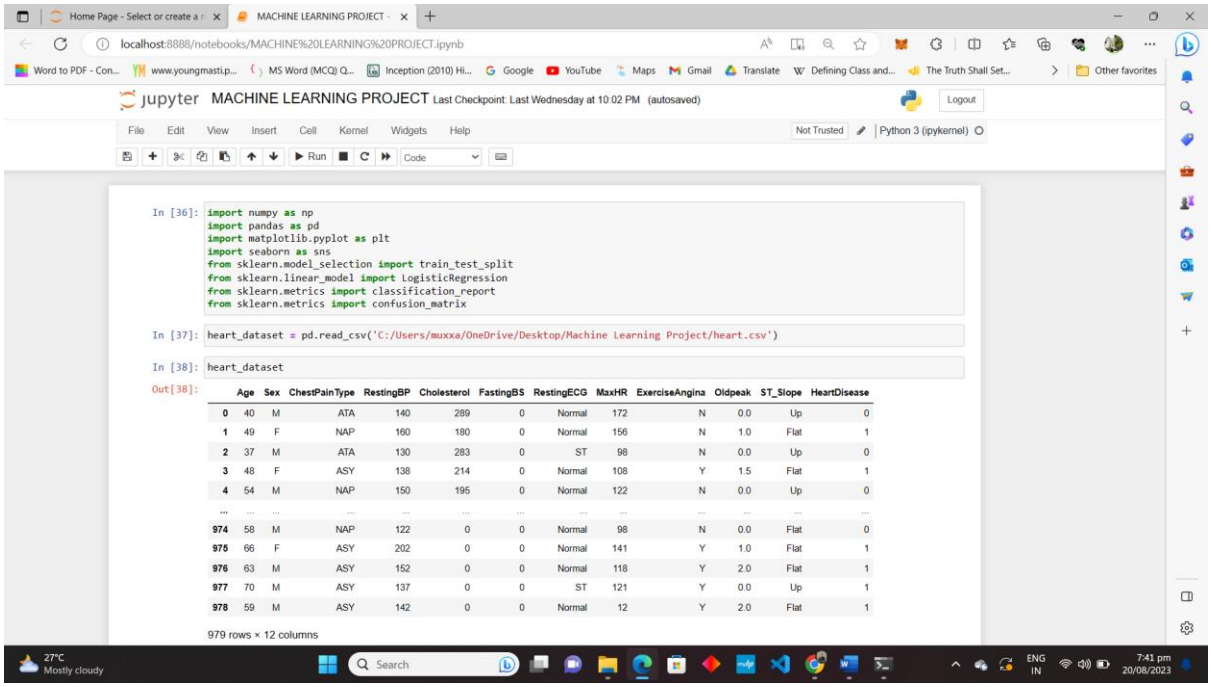
CODE:

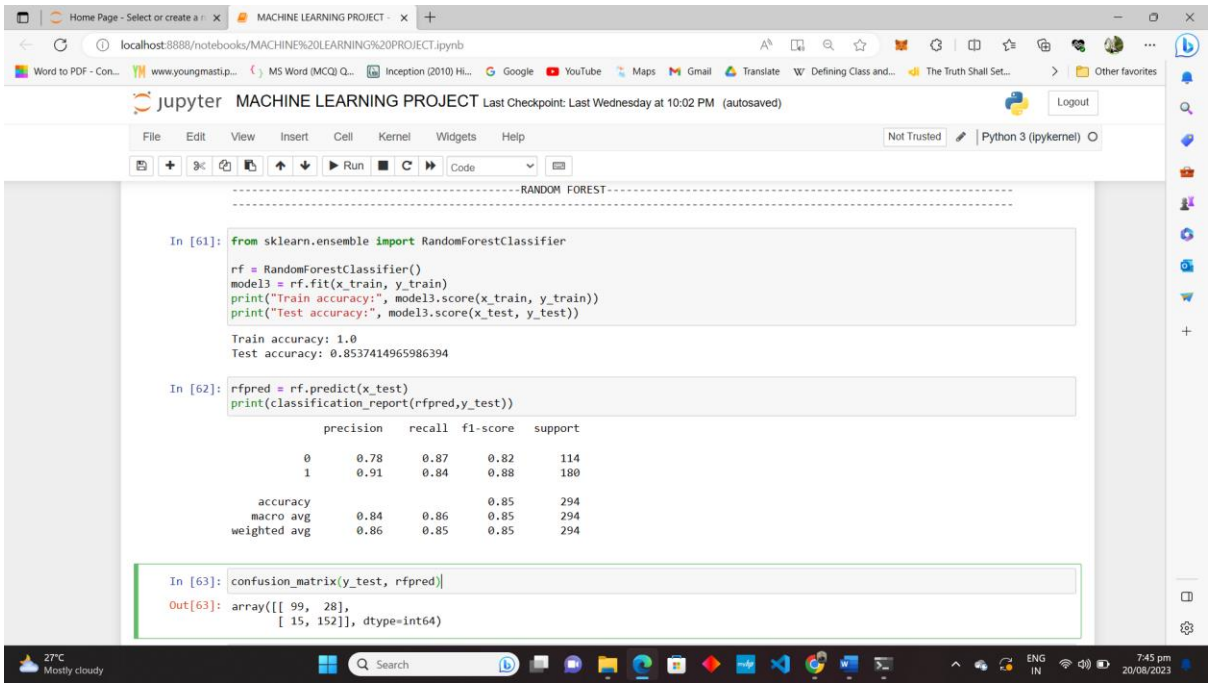
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

heart_dataset = pd.read_csv('C:/Users/muxxa/OneDrive/Desktop/Machine Learning Project/heart.csv')

heart_dataset

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

model3 = rf.fit(x_train, y_train)

print("Train accuracy:", model3.score(x_train, y_train))

print("Test accuracy:", model3.score(x_test, y_test))

rfpred = rf.predict(x_test)
```

print(classification_report(rfpred,y_test))

confusion_matrix(y_test, rfpred)

CODE + OUTPUTS:





2. Reinforcement learning and describe the structure of neural networks.

Reinforcement Learning (RL):

Reinforcement Learning is like teaching a computer to learn from trial and error, just like how we learn through experiences. Imagine you're training a pet. When it does something good, you reward it; when it does something not-so-good, you guide it towards a better action. Over time, your pet learns to maximize the good actions to get more rewards.

For instance, think about a self-driving car. It learns to navigate the roads by getting rewards (like arriving at the destination) and punishments (like accidents). Through these experiences, the car improves its driving skills, making it better at getting us where we want to go while avoiding problems.

Neural Networks Structure:

Neural networks are like a digital version of the human brain's intricate connections. They're made up of interconnected "neurons" that process information. Let's break down their structure:

i. Input Layer: This is where data enters the network. It's like the first step where the network gets a taste of what's happening.

ii. Hidden Layers: Think of these as the network's inner thoughts. It processes the input through a series of calculations, trying to understand the underlying patterns.

iii. Output Layer: This is where the network gives its final answer. It's like the network's conclusion after analyzing the data.

Each neuron talks to others by passing information along weighted connections. These weights are like the strength of opinions among friends they change as the network learns from data. Activation functions are like the excitement level of each neuron. They determine whether a neuron should "fire up" and pass on its message or stay quiet. Imagine a painter mixing colors. Neurons calculate, combine, and activate, creating complex representations of the input data. This "painting" is the network's prediction. Training the network is like teaching an artist to paint better. We show them paintings and tell them what's wrong and right. They adjust their technique until their paintings look amazing. In summarize manner, neural networks are like creative minds that learn from examples. They're used in things like recognizing faces, translating languages, and even playing games like chess.