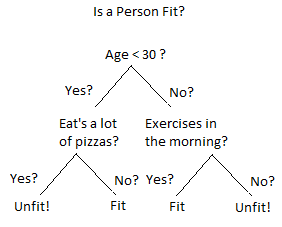
**Theory:**

A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter.

**Decision Tree consists of :**

1. **Nodes** : Test for the value of a certain attribute.
2. **Edges/ Branch** : Correspond to the outcome of a test and connect to the next node or leaf.
3. **Leaf nodes** : Terminal nodes that predict the outcome (represent class labels or class distribution).



To understand the concept of Decision Tree consider the above example. Let’s say you want to predict whether a person is fit or unfit, given their information like age, eating habits, physical activity, etc. The decision nodes are the questions like ‘What’s the age?’, ‘Does he exercise?’, ‘Does he eat a lot of pizzas’? And the leaves represent outcomes like either ‘fit’, or ‘unfit’.

## ****Decision Tree Classifier****

* Using the decision algorithm, we start at the tree root and split the data on the feature that results in the **largest information gain (IG)** (reduction in uncertainty towards the final decision).
* In an iterative process, we can then repeat this splitting procedure at each child node **until the leaves are pure**. This means that the samples at each leaf node all belong to the same class.
* In practice, we may set a **limit on the depth of the tree to prevent overfitting**. We compromise on purity here somewhat as the final leaves may still have some impurity.

**Bayesian Classifier:**

In numerous applications, the connection between the attribute set and the class variable is non- deterministic. In other words, we can say the class label of a test record cant be assumed with certainty even though its attribute set is the same as some of the training examples. These circumstances may emerge due to the noisy data or the presence of certain confusing factors that influence classification, but it is not included in the analysis. For example, consider the task of predicting the occurrence of whether an individual is at risk for liver illness based on individuals eating habits and working efficiency. Although most people who eat healthly and exercise consistently having less probability of occurrence of liver disease, they may still do so due to other factors. For example, due to consumption of the high-calorie street foods and alcohol abuse. Determining whether an individual's eating routine is healthy or the workout efficiency is sufficient is also subject to analysis, which in turn may introduce vulnerabilities into the leaning issue.

Bayesian classification uses Bayes theorem to predict the occurrence of any event. Bayesian classifiers are the statistical classifiers with the Bayesian probability understandings. The theory expresses how a level of belief, expressed as a probability.

Bayes theorem came into existence after Thomas Bayes, who first utilized conditional probability to provide an algorithm that uses evidence to calculate limits on an unknown parameter.

Bayes's theorem is expressed mathematically by the following equation that is given below.

Data Mining Bayesian Classifiers

Where X and Y are the events and P (Y) ≠ 0

P(X/Y) is a **conditional probability** that describes the occurrence of event **X** is given that **Y** is true.

P(Y/X) is a **conditional probability** that describes the occurrence of event **Y** is given that **X** is true.

P(X) and P(Y) are the probabilities of observing X and Y independently of each other. This is known as the **marginal probability**.

**Program:**

**Decision Tree Classifier:**

import pandas as pd

from sklearn.metrics import confusion\_matrix

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

# Function importing Dataset

def importdata():

    balance\_data = pd.read\_csv(

    'https://archive.ics.uci.edu/ml/machine-learning-' +

    'databases/balance-scale/balance-scale.data',

    sep=',', header=None)

    # Printing the dataswet shape

    print("Dataset Length: ", len(balance\_data))

    print("Dataset Shape: ", balance\_data.shape)

    # Printing the dataset obseravtions

    print("Dataset: ", balance\_data.head())

    return balance\_data

# Function to split the dataset

def splitdataset(balance\_data):

    # Seperating the target variable

    X = balance\_data.values[:, 1:5]

    Y = balance\_data.values[:, 0]

    # Spliting the dataset into train and test

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, Y, test\_size=0.3, random\_state=100)

    return X, Y, X\_train, X\_test, y\_train, y\_test

# Function to perform training with giniIndex.

def train\_using\_gini(X\_train, X\_test, y\_train):

    # Creating the classifier object

    clf\_gini = DecisionTreeClassifier(criterion="gini",

    random\_state=100, max\_depth=3, min\_samples\_leaf=5)

    # Performing training

    clf\_gini.fit(X\_train, y\_train)

    return clf\_gini

# Function to perform training with entropy.

def tarin\_using\_entropy(X\_train, X\_test, y\_train):

    # Decision tree with entropy

    clf\_entropy = DecisionTreeClassifier(

    criterion="entropy", random\_state=100,

    max\_depth=3, min\_samples\_leaf=5)

    # Performing training

    clf\_entropy.fit(X\_train, y\_train)

    return clf\_entropy

# Function to make predictions

def prediction(X\_test, clf\_object):

    # Predicton on test with giniIndex

    y\_pred = clf\_object.predict(X\_test)

    print("Predicted values:")

    print(y\_pred)

    return y\_pred

# Function to calculate accuracy

def cal\_accuracy(y\_test, y\_pred):

    print("Confusion Matrix: ",

    confusion\_matrix(y\_test, y\_pred))

    print("Accuracy : ",

    accuracy\_score(y\_test, y\_pred) \* 100)

    print("Report : ",

    classification\_report(y\_test, y\_pred))

# Driver code

def main():

    # Building Phase

    data = importdata()

    X, Y, X\_train, X\_test, y\_train, y\_test = splitdataset(data)

    clf\_gini = train\_using\_gini(X\_train, X\_test, y\_train)

    clf\_entropy = tarin\_using\_entropy(X\_train, X\_test, y\_train)

    # Operational Phase

    print("Results Using Gini Index:")

    # Prediction using gini

    y\_pred\_gini = prediction(X\_test, clf\_gini)

    cal\_accuracy(y\_test, y\_pred\_gini)

    print("Results Using Entropy:")

# Prediction using entropy

    y\_pred\_entropy = prediction(X\_test, clf\_entropy)

    cal\_accuracy(y\_test, y\_pred\_entropy)

# Calling main function

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Bayesian Classifier:**

from sklearn.metrics import confusion\_matrix

from sklearn.naive\_bayes import GaussianNB

from sklearn import tree

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

from sklearn import metrics

from sklearn import datasets

import pandas as pd

import numpy as np

wine = datasets.load\_wine()

print("Features: ", wine.feature\_names)

print("Labels: ", wine.target\_names)

X = pd.DataFrame(wine['data'])

print(X.head())

print(wine.data.shape)

y = print(wine.target)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(wine.data, wine.target,test\_size=0.30, random\_state=109)

gnb = GaussianNB()

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

y\_pred = gnb.predict(X\_test)

print(y\_pred)

print("Accuracy:", metrics.accuracy\_score(y\_test, y\_pred))

cm = np.array(confusion\_matrix(y\_test, y\_pred))

print("Confusion matrix:")

print(cm)

dataset = datasets.load\_wine()

X = dataset.data

y = dataset.target

print("Features: ", wine.feature\_names)

# print the label type of wine

print("Labels: ", wine.target\_names)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25)

model = tree.DecisionTreeClassifier()

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

print(model)

expected\_y = y\_test

predicted\_y = model.predict(X\_test)

print(metrics.classification\_report(expected\_y, predicted\_y,

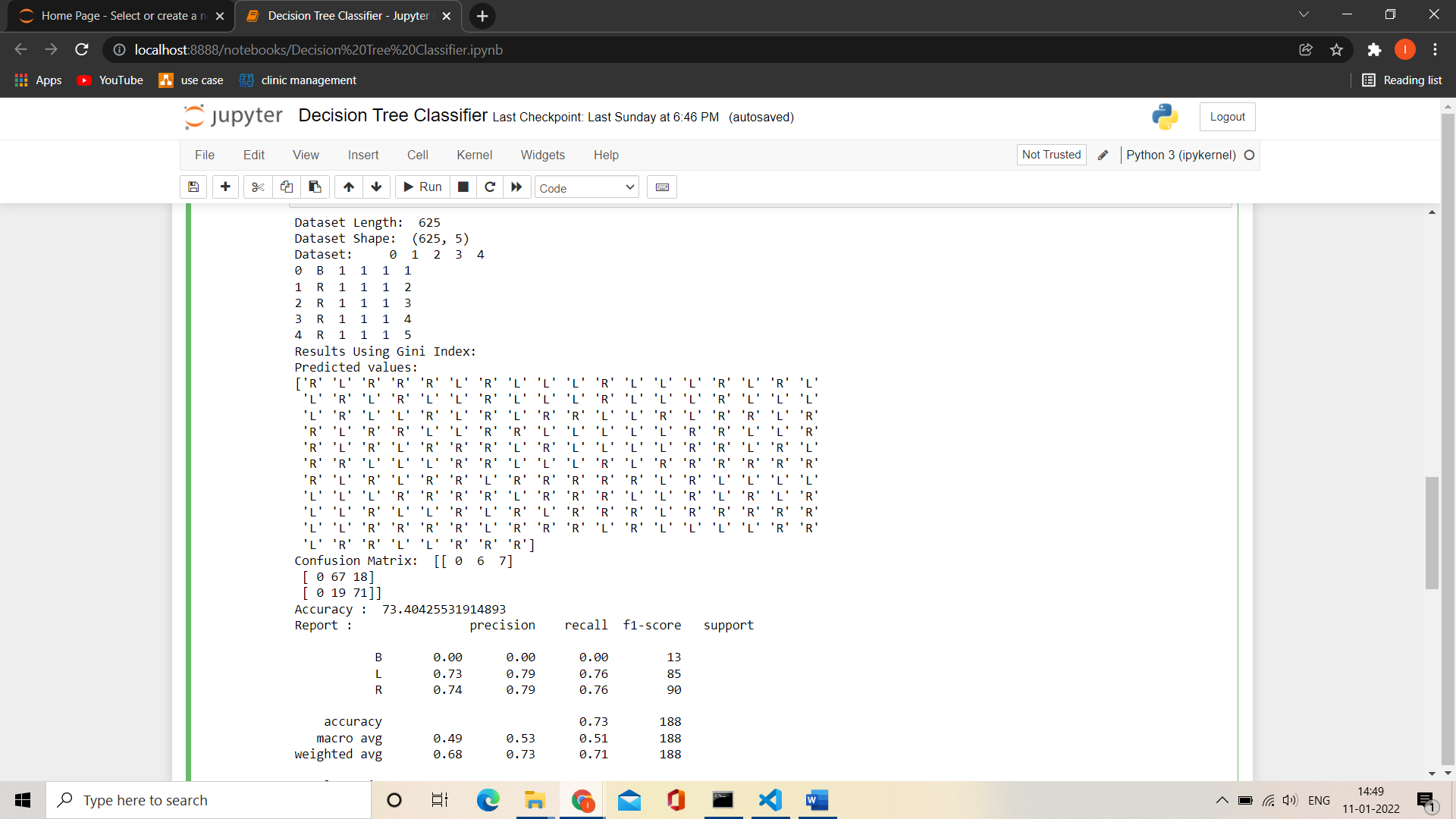
                                    target\_names=dataset.target\_names))

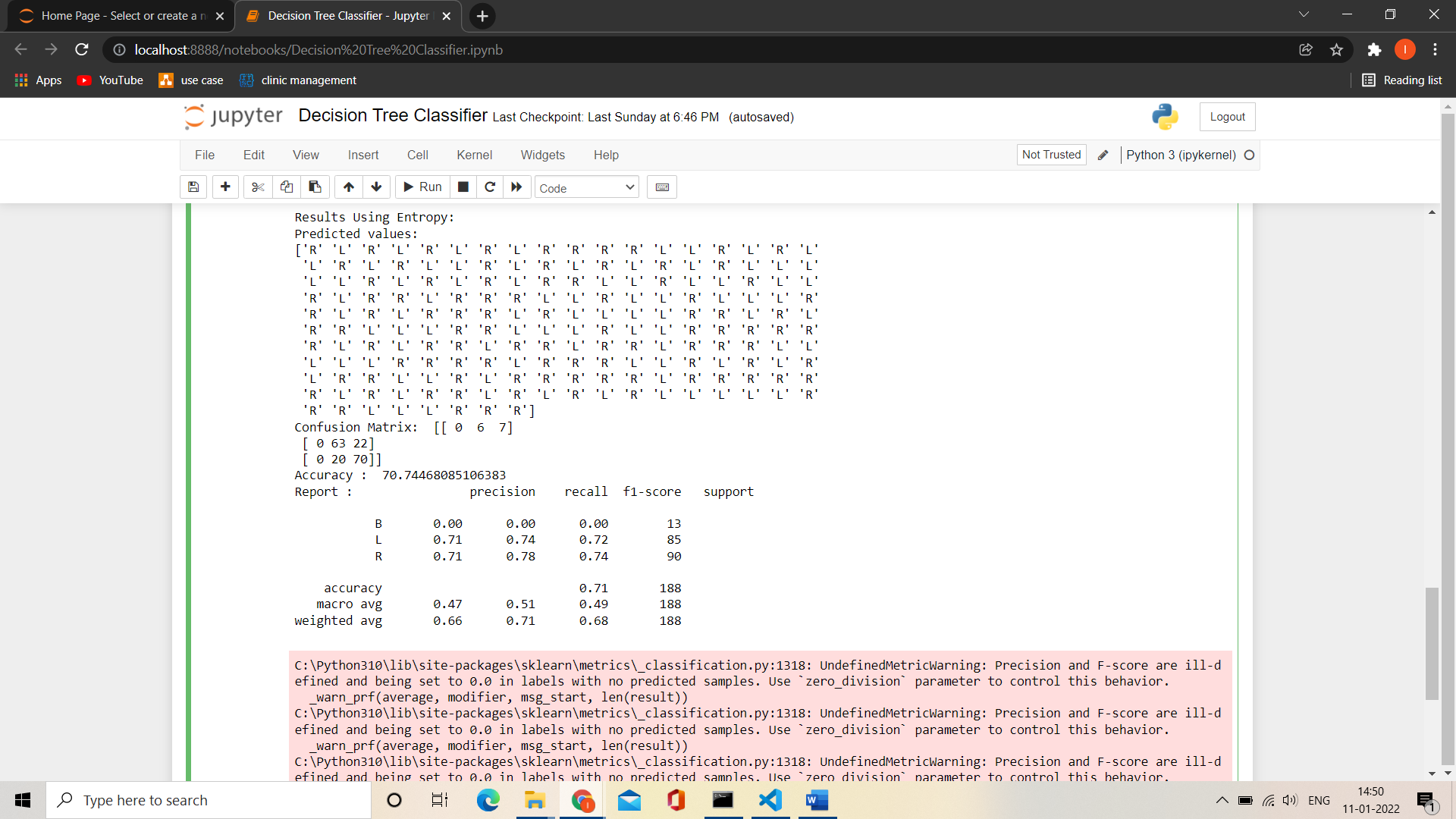
print("Confusion Matrix:")

print(metrics.confusion\_matrix(expected\_y, predicted\_y))

**Screenshots:**

**Decision Tree Classifier:**





**Bayesian Classifier:**

