1. Write a program in python to classify the “Iris” dataset available with Scikit Learn to perform Decision Tree Classification using the CART Algorithm.

**Solution:**

# Load libraries

from sklearn.tree import DecisionTreeClassifier

from sklearn import datasets

from IPython.display import Image

from sklearn import tree

import graphviz

from graphviz import Source

# Load data

iris = datasets.load\_iris()

X = iris.data

y = iris.target

# Create decision tree classifer object

clf = DecisionTreeClassifier(random\_state=0)

# Train model

model = clf.fit(X, y)

# Create DOT data

dot\_data = tree.export\_graphviz(clf, out\_file=None, feature\_names=iris.feature\_names, class\_names=iris.target\_names)

graph = graphviz.Source(dot\_data)

graph.render("newtree",view = True)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Write a program in python to perform regression using Decision tree on the following data of different Games (Cloumn\_1) on the basis of Production Cost (Cloumn\_2) and Profit (Cloumn\_3):

['Asset Flip', 100, 1000],

['Text Based', 500, 3000],

['Visual Novel', 1500, 5000],

['2D Pixel Art', 3500, 8000],

['2D Vector Art', 5000, 6500],

['Strategy', 6000, 7000],

['First Person Shooter', 8000, 15000],

['Simulator', 9500, 20000],

['Racing', 12000, 21000],

['RPG', 14000, 25000],

['Sandbox', 15500, 27000],

['Open-World', 16500, 30000],

['MMOFPS', 25000, 52000],

['MMORPG', 30000, 80000]

**Solution:**

'''Step 1: Import the required libraries.'''

# import numpy package for arrays and stuff

import numpy as np

# import matplotlib.pyplot for plotting our result

import matplotlib.pyplot as plt

# import pandas for importing csv files

import pandas as pd

'''Step 2: Initialize and print the Dataset.'''

# import dataset

# dataset = pd.read\_csv('Data.csv')

# alternatively open up .csv file to read data

dataset = np.array(

[['Asset Flip', 100, 1000],

['Text Based', 500, 3000],

['Visual Novel', 1500, 5000],

['2D Pixel Art', 3500, 8000],

['2D Vector Art', 5000, 6500],

['Strategy', 6000, 7000],

['First Person Shooter', 8000, 15000],

['Simulator', 9500, 20000],

['Racing', 12000, 21000],

['RPG', 14000, 25000],

['Sandbox', 15500, 27000],

['Open-World', 16500, 30000],

['MMOFPS', 25000, 52000],

['MMORPG', 30000, 80000]

])

# print the dataset

print(dataset)

'''Step 3: Select all the rows and column 1 from dataset to “X”.'''

# select all rows by : and column 1

# by 1:2 representing features

X = dataset[:, 1:2].astype(int)

# print X

print(X)

'''Step 4: Select all of the rows and column 2 from dataset to “y”.'''

# select all rows by : and column 2

# by 2 to Y representing labels

y = dataset[:, 2].astype(int)

# print y

print(y)

'''Step 5: Fit decision tree regressor to the dataset'''

# import the regressor

from sklearn.tree import DecisionTreeRegressor

# create a regressor object

regressor = DecisionTreeRegressor(random\_state = 0)

# fit the regressor with X and Y data

regressor.fit(X, y)

'''Step 6: Predicting a new value'''

# predicting a new value

# test the output by changing values, like 3750

y\_pred = regressor.predict([[3750]])

# print the predicted price

print("Predicted price: % d\n"% y\_pred)

'''Step 7: Visualising the result'''

# arange for creating a range of values

# from min value of X to max value of X

# with a difference of 0.01 between two

# consecutive values

X\_grid = np.arange(min(X), max(X), 0.01)

# reshape for reshaping the data into

# a len(X\_grid)\*1 array, i.e. to make

# a column out of the X\_grid values

X\_grid = X\_grid.reshape((len(X\_grid), 1))

# scatter plot for original data

plt.scatter(X, y, color = 'red')

# plot predicted data

plt.plot(X\_grid, regressor.predict(X\_grid), color = 'blue')

# specify title

plt.title('Profit to Production Cost (Decision Tree Regression)')

# specify X axis label

plt.xlabel('Production Cost')

# specify Y axis label

plt.ylabel('Profit')

# show the plot

plt.show()

'''Step 8: The tree is finally exported and shown in the TREE STRUCTURE below,

automatically visualized in .pdf format with name 'tree.pdf' '''

## import export\_graphviz

from sklearn import tree

import graphviz

from graphviz import Source

dot\_data = tree.export\_graphviz(regressor, out\_file =None, feature\_names =['Production Cost'])

graph = graphviz.Source(dot\_data)

graph.render("tree",view = True)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Write a program in python to perform Classification on the “Iris” dataset available with Scikit Learn using Gaussian Naive Bayes Classifier. Also generate the Confusion Matrix. Also separate out the data into training and testing and find out the number of mislabeled points.

**Solution:**

# Gaussian Naive Bayes

from sklearn import datasets

from sklearn import metrics

from sklearn.naive\_bayes import GaussianNB

# load the iris datasets

dataset = datasets.load\_iris()

# fit a Naive Bayes model to the data

model = GaussianNB()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

expected = dataset.target

predicted = model.predict(dataset.data)

# summarize the fit of the model

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

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

from sklearn.datasets import load\_iris

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

X, y = load\_iris(return\_X\_y=True)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=0)

gnb = GaussianNB()

y\_pred = gnb.fit(X\_train, y\_train).predict(X\_test)

print("Number of mislabeled points out of a total %d points : %d"% (X\_test.shape[0], (y\_test != y\_pred).sum()))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Write a program in python to perform Multi-label Classification on the “Wine” dataset available with Scikit Learn using Gaussian Naive Bayes Classifier. Also find the Accuracy of the classification model.

**Solution:**

#Import scikit-learn dataset library

from sklearn import datasets

#Load dataset

wine = datasets.load\_wine()

# print the names of the 13 features

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

# print the label type of wine(class\_0, class\_1, class\_2)

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

# print data(feature)shape

wine.data.shape

# print the wine data features (top 5 records)

print (wine.data[0:5])

# print the wine labels (0:Class\_0, 1:class\_2, 2:class\_2)

print (wine.target)

# Import train\_test\_split function

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

# Split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(wine.data, wine.target, test\_size=0.3,random\_state=109)

# 70% training and 30% test

#Import Gaussian Naive Bayes model

from sklearn.naive\_bayes import GaussianNB

#Create a Gaussian Classifier

gnb = GaussianNB()

#Train the model using the training sets

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

#Predict the response for test dataset

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

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

# Model Accuracy, how often is the classifier correct?

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*