**LAB - 2**

**MACHINE LEARNING**

**PRIYANSHU SHARMA**

**15BCE1282**

**DECISION TREE**

**CODE -**

import pandas as pd

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/breast.csv')

data.head()

colnames=['ID', 'RADIUS', 'TEXTURE', 'PERIMETER', 'AREA', 'SMOOTHNESS', 'COMPACTNESS', 'CONCAVITY', 'CONCAVE', 'SYMMETRY', 'FRACTAL']

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/breast.csv', names=colnames, header=None)

data.head()

print(data.columns)

data.describe()

# DECISION TREE IMPLEMENTATION

from sklearn.tree import DecisionTreeClassifier

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

print(data.columns)

columns=['RADIUS', 'TEXTURE', 'PERIMETER', 'AREA', 'SMOOTHNESS', 'COMPACTNESS', 'CONCAVITY', 'CONCAVE', 'SYMMETRY']

a=data[columns].iloc[:,:9].values #all columns in array

a

b=data[columns].iloc[:,0:1].values #label column in array (particular column selection)

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(data[columns],data['FRACTAL'],test\_size=0.4,random\_state=14)

tree = DecisionTreeClassifier(max\_depth=7,random\_state=0)

tree.fit(X\_train,Y\_train)

print("Accuracy on the training set: %.3f" % tree.score(X\_train,Y\_train))

print("Accuracy on the testing set: %.3f" % tree.score(X\_test,Y\_test))

from sklearn.tree import export\_graphviz

export\_graphviz(tree, out\_file="tree.dot", class\_names=['2','4'], impurity=False, filled=True, feature\_names=data[columns].columns)

import graphviz

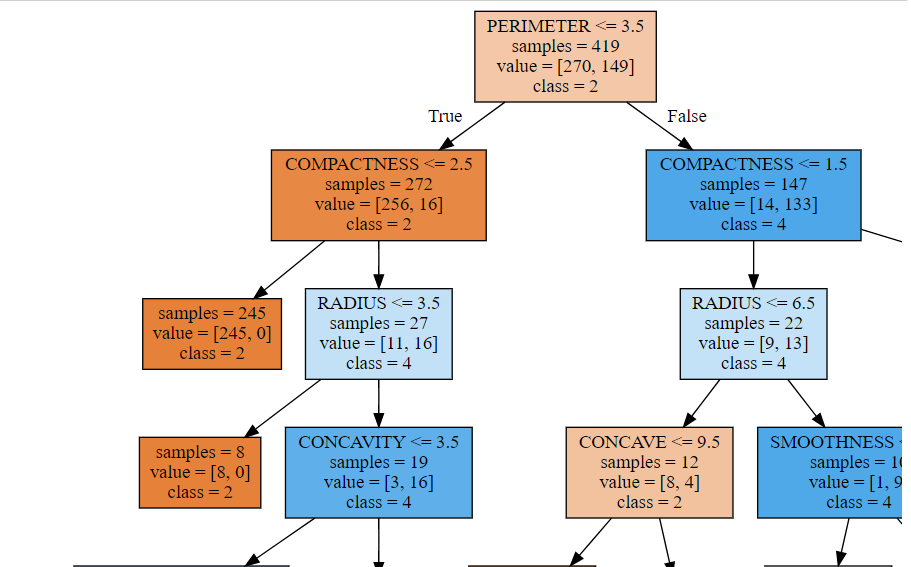
with open("tree.dot") as f:

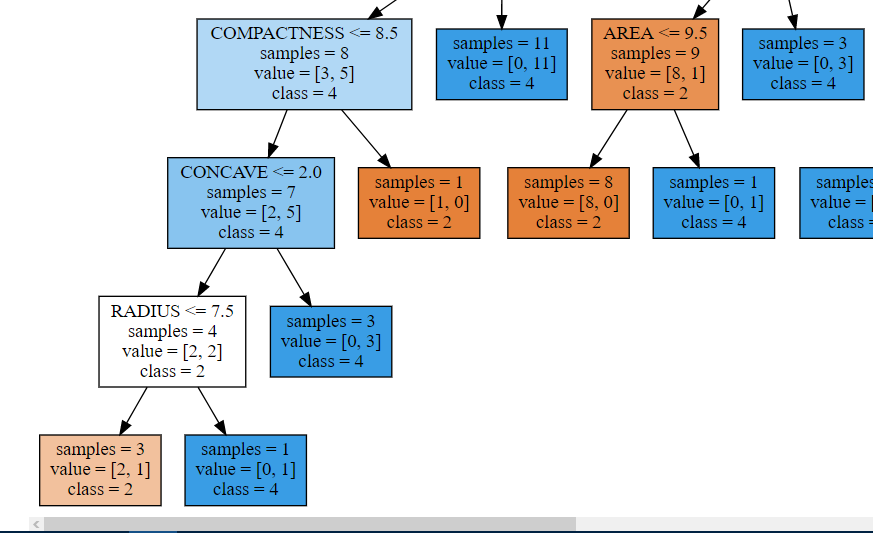
dot\_graph = f.read()

graphviz.Source(dot\_graph)

**OUTPUT**

**\*\*MY OUTPUT DECISION TREE IS BASED ON (wisconsin breast cancer datasets)**





**LINEAR REGRESSION**

**CODE -**

#LINEAR REGRESSION

import pandas as pd

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/breast.csv')

data.head()

colnames=['ID', 'RADIUS', 'TEXTURE', 'PERIMETER', 'AREA', 'SMOOTHNESS', 'COMPACTNESS', 'CONCAVITY', 'CONCAVE', 'SYMMETRY', 'FRACTAL']

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/breast.csv', names=colnames, header=None)

data.head()

import matplotlib.pyplot as plt

import seaborn as sb

sb.set(style='whitegrid', context='notebook')

cols = ['SMOOTHNESS', 'SYMMETRY', 'CONCAVE', 'PERIMETER', 'AREA']

sb.pairplot(data[cols], size=2.5);

plt.show()

import numpy as np

cm = np.corrcoef(data[cols].values.T)

sb.set(font\_scale=1.5)

hm = sb.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot\_kws={'size': 15}, yticklabels=cols, xticklabels=cols)

plt.show()

from sklearn import datasets, linear\_model

from sklearn.linear\_model import LinearRegression

X = data[['CONCAVE']].values

Y = data[['AREA']].values

regr = linear\_model.LinearRegression()

regr.fit(X, Y)

plt.scatter(X, Y, color='black')

plt.plot(X, regr.predict(X), color='blue', linewidth=3)

plt.xticks(())

plt.yticks(())

plt.show()

regr.coef\_ #slope

regr.intercept\_ #intercept

print('Slope: %.3f' % regr.coef\_[0])

print('Intercept: %.3f' % regr.intercept\_[0])

class LinearRegressionGD(object):

def \_\_init\_\_(self, eta=0.001, n\_iter=20):

self.eta = eta

self.n\_iter = n\_iter

def fit(self, X, y):

self.w\_ = np.zeros(1 + X.shape[1])

self.cost\_ = []

for i in range(self.n\_iter):

output = self.net\_input(X)

errors = (y - output)

self.w\_[1:] += self.eta \* X.T.dot(errors)

self.w\_[0] += self.eta \* errors.sum()

cost = (errors\*\*2).sum() / 2.0

self.cost\_.append(cost)

return self

def net\_input(self, X):

return np.dot(X, self.w\_[1:]) + self.w\_[0]

def predict(self, X):

return self.net\_input(X)

from sklearn.preprocessing import StandardScaler

def lin\_regplot(X, y, model):

plt.scatter(X, y, c='blue')

plt.plot(X, model.predict(X), color='red')

return None

lin\_regplot(X, Y, regr)

plt.xlabel('CONCAVE')

plt.ylabel('AREA')

plt.show()

data['CONCAVE'].unique()

data['AREA'].unique()

area\_std = regr.predict(12)

print("AREA: %.3f" %area\_std)

**OUTPUT**

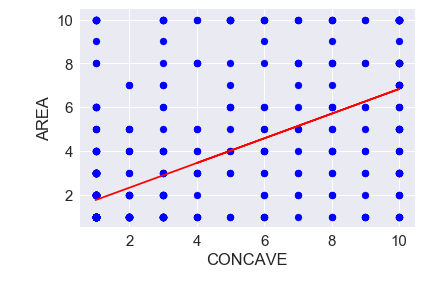
**\*\*MY OUTPUT DECISION TREE IS BASED ON (wisconsin breast cancer datasets)**

Slope: 0.564

Intercept: 1.189

**PREDICATED DATA VALUE -**

AREA: 7.960 (**FOR CONCAVE = 12**)



**QUESTION - 1 POKEMON URBAN DATASET**

**CODE -**

import pandas as pd

import matplotlib.pyplot as plt

colnames=['ITEM', 'POKEMON','URBAN']

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/pokeurban.csv', names=colnames, header=None)

data.head()

from sklearn import datasets, linear\_model

from sklearn.linear\_model import LinearRegression

X = data[['POKEMON']].values

Y = data[['URBAN']].values

regr = linear\_model.LinearRegression()

regr.fit(X, Y)

plt.scatter(X, Y, color='black')

plt.plot(X, regr.predict(X), color='blue', linewidth=3)

plt.xticks(())

plt.yticks(())

plt.show()

regr.coef\_ #slope

regr.intercept\_ #intercept

print('Slope: %.3f' % regr.coef\_[0])

print('Intercept: %.3f' % regr.intercept\_[0])

def lin\_regplot(X, y, model):

plt.scatter(X, y, c='blue')

plt.plot(X, model.predict(X), color='red')

return None

lin\_regplot(X, Y, regr)

plt.xlabel('POKEMON')

plt.ylabel('URBAN')

plt.show()

urban\_std = regr.predict(40) #when pokemon = 40

print("URBAN: %.3f" %urban\_std)

**OUTPUT**

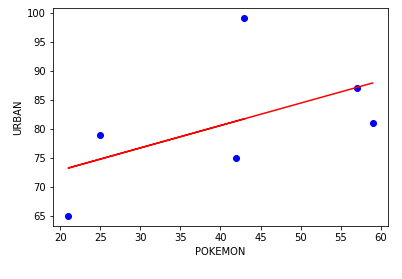
**\*\*MY OUTPUT DECISION TREE IS BASED ON (pokemon urban datasets)**

Slope: 0.385

Intercept: 65.142

**PREDICATED DATA VALUE -**

URBAN: 80.551 (**FOR POKEMON = 40**)



**QUESTION - 1 LENGTH MILEAGE DATASET**

**CODE -**

import pandas as pd

import matplotlib.pyplot as plt

colnames=['LENGTH', 'MILEAGE']

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/mileagecar.csv', names=colnames, header=None)

data.head()

from sklearn import datasets, linear\_model

from sklearn.linear\_model import LinearRegression

X = data[['LENGTH']].values

Y = data[['MILEAGE']].values

regr = linear\_model.LinearRegression()

regr.fit(X, Y)

plt.scatter(X, Y, color='black')

plt.plot(X, regr.predict(X), color='blue', linewidth=3)

plt.xticks(())

plt.yticks(())

plt.show()

regr.coef\_ #slope

regr.intercept\_ #intercept

print('Slope: %.3f' % regr.coef\_[0])

print('Intercept: %.3f' % regr.intercept\_[0])

def lin\_regplot(X, y, model):

plt.scatter(X, y, c='blue')

plt.plot(X, model.predict(X), color='red')

return None

lin\_regplot(X, Y, regr)

plt.xlabel('LENGTH')

plt.ylabel('MILEAGE')

plt.show()

mileage\_std = regr.predict(200) #when MILEAGE = 200

print("MILEAGE: %.3f" %mileage\_std)

**OUTPUT**

**\*\*MY OUTPUT DECISION TREE IS BASED ON (length mileage datasets)**

Slope: -0.232

Intercept: 64.763

**PREDICATED DATA VALUE -**

MILEAGE: 18.357 **(FOR LENGTH = 200)**

