LAB - 2
MACHINE LEARNING

PRIYANSHU SHARMA
15BCE1282

DECISION TREE

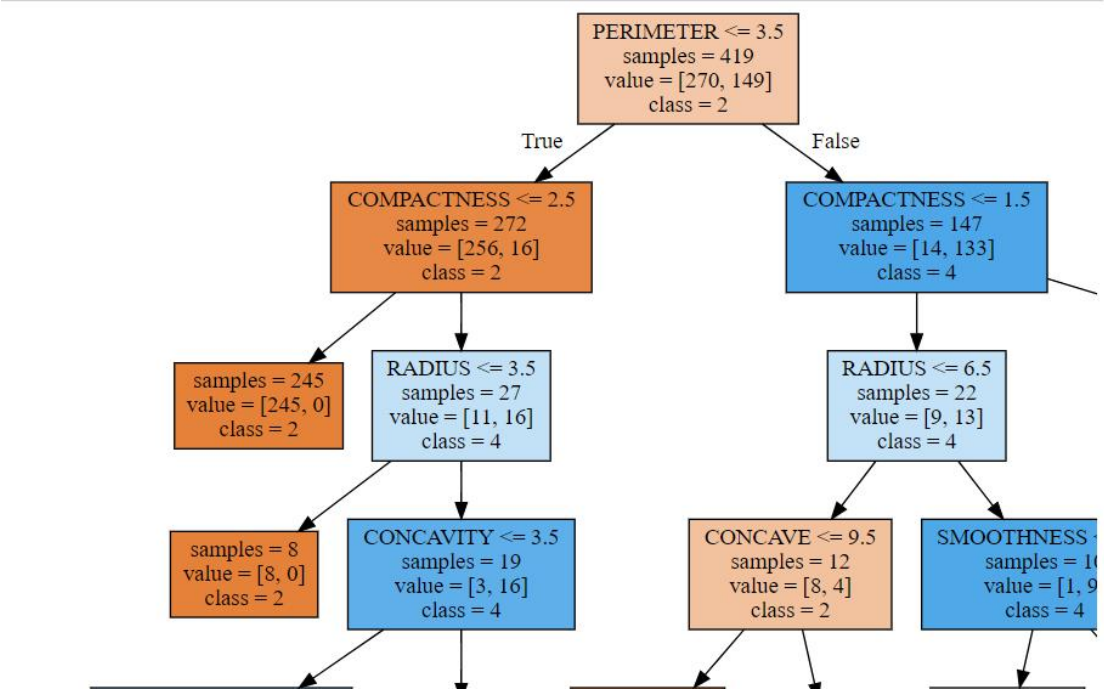
CODE -

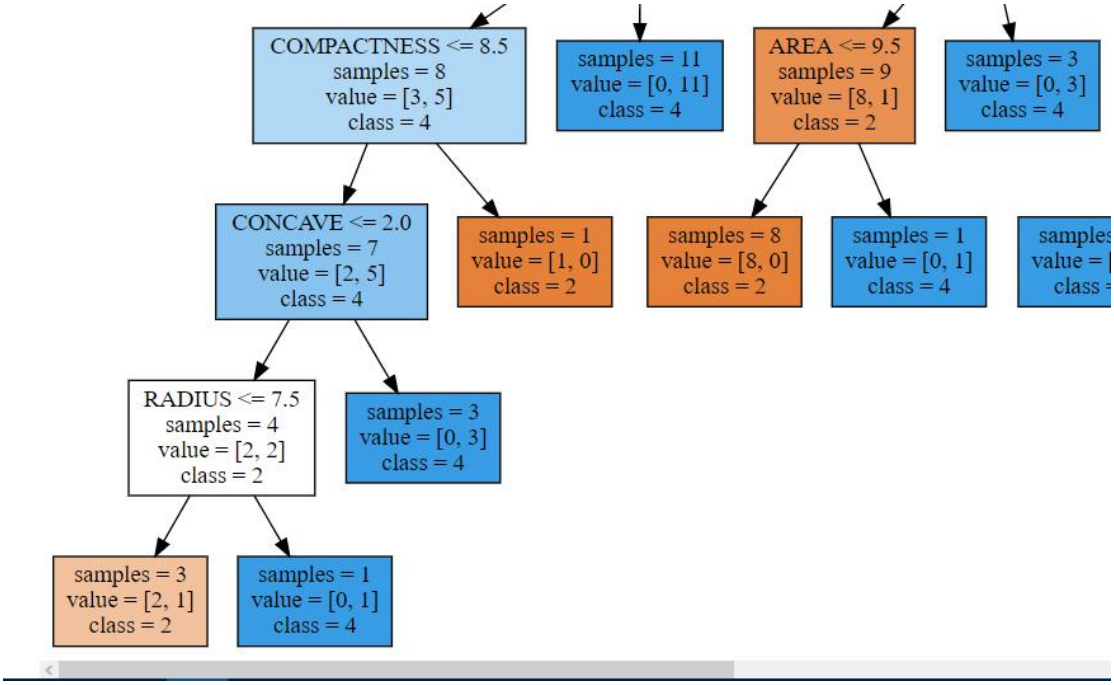
```
import pandas as pd

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)
```

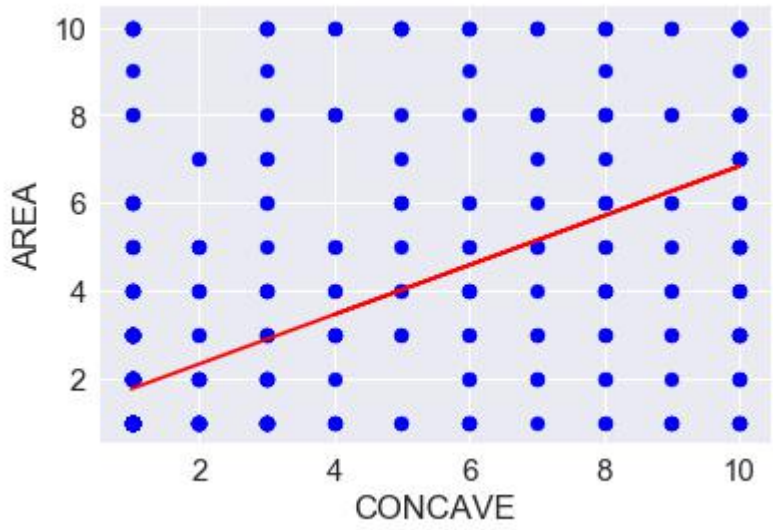
OUTPUT

**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)
```

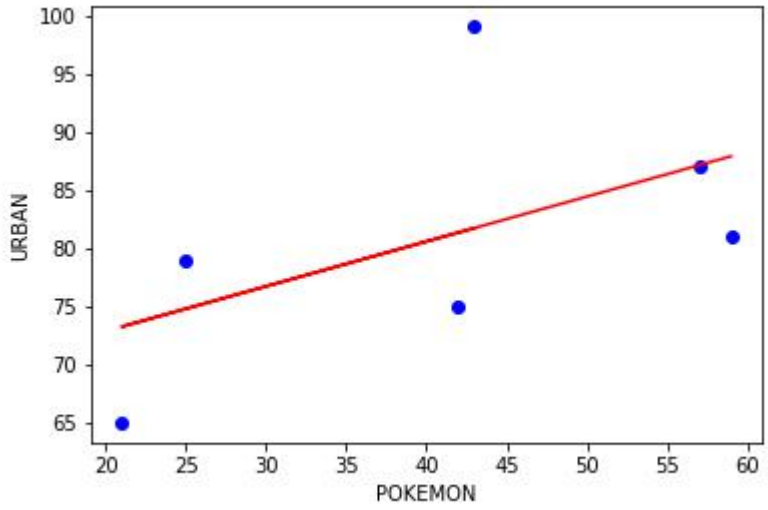
OUTPUT

**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)**