**CSE422\_Lab05:**

from google.colab import files

load\_data = files.upload()

import sklearn

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import MinMaxScaler

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_score

from sklearn.tree import DecisionTreeClassifier

#loading data

dataset = pd.read\_csv('mushroom edibility classification dataset.csv')

# missing values

dataset.drop(dataset.columns[dataset.columns.str.contains('unnamed',case = False)],axis = 1, inplace = True)

dataset.head()

dataset = dataset.dropna(axis = 0, subset = ['cap-shape', 'cap-color'])

dataset.shape

print(dataset.info())

print('unique values in class ' + str(dataset['class'].unique()))

print('unique values in bruises' + str(dataset['bruises'].unique()))

#encoding categorical features

enc = LabelEncoder()

dataset['class'] = enc.fit\_transform(dataset['class'])

dataset['bruises'] = enc.fit\_transform(dataset['bruises'])

dataset[['class', 'bruises']].head()

scaler = MinMaxScaler()

scaler.fit(dataset)

dataset\_train\_scaled = scaler.transform(dataset)

print('per-feature minimum before scaling:\n{}'.format(dataset.min(axis = 0)))

print('per-feature maximum before scaling:\n{}'.format(dataset.max(axis = 0)))

print('per-feature minimum after scaling:\n{}'.format(dataset\_train\_scaled.min(axis = 0)))

print('per-feature maximum after scaling:\n{}'.format(dataset\_train\_scaled.max(axis = 0)))

#splitting the dataset into features and labels

features = dataset[['cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor', 'stalk-shape', 'stalk-root', 'stalk-surface-above-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-color', 'population', 'habitat']]

label = dataset[['class']]

stratified = pd.DataFrame(label)

#8:2 train-test split

xTrain, xTest, yTrain, yTest = train\_test\_split(features, label, test\_size = 0.20, stratify = stratified, random\_state = 0)

#using the logistic regression

perform\_logisticRegression = LogisticRegression()

perform\_logisticRegression.fit(xTrain, yTrain)

predictions = perform\_logisticRegression.predict(xTest)

accuracy\_of\_LogisticRegression = accuracy\_score(yTest, predictions)

print(accuracy\_of\_LogisticRegression)

#using the decision tree

perform\_decisionTree = DecisionTreeClassifier(criterion='entropy',random\_state=1)

perform\_decisionTree.fit(xTrain,yTrain)

yPred = perform\_decisionTree.predict(xTest)

accuracy\_of\_DecisionTreeClassifier = accuracy\_score(yPred, yTest)

print(accuracy\_of\_DecisionTreeClassifier)

# comparing the accuracy and plotting them as a bar chart using matplotlib

plt.bar(['Logistic Regression', 'Decision Tree'],[accuracy\_of\_LogisticRegression, accuracy\_of\_DecisionTreeClassifier])

plt.title('Comparison between Logistic Regression and Decision Tree' )

plt.show()