ENGG*6500: Introduction to Machine Learning

Assignment #2

Spencer Walls

1. Write a Python code to implement a decision tree classifier according to the data points. The file heart.csv can be downloaded from CourseLink.

```
# This program takes as input the "heart.csv" dataset.
# A decision tree is then implemented with respect to
# this dataset, and the output is simply the accuracy
# of the algorithm's predictions. The data is first
# separated into a training set and a testing set,
# which collectively comprise a 70/30 train/test split.

# Step 1 - import necessary libraries
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

# Step 2 - import the dataset
data = pd.read_csv("heart.csv").values

# Step 3 - split the dataset into X and y
X = data[:,0:5]
y = data[:,-1]

# Step 4 - split the dataset into a training set and a testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Step 5 - create in instance of the decision tree algorithm
decision_tree = DecisionTreeClassifier()

# Step 6 - fit the algorithm to the dataset
decision_tree = decision_tree.fit(X_train, y_train)

# Step 7 - make predictions for y_test using the testing set (X_test) as inputs
predict = decision_tree.predict(X_test)

# Step 8 - store predicted values with ground truth values in a table and print it
table = np.zeros((91,2), dtype=np.int64)
table[:,0] = y_test
table[:,1] = predict
print("\nactual values vs. predicted values\n")
print(table)

# Step 8 - print the accuracy of the decision tree algorithm
print("\nAccuracy of decision tree:", metrics.accuracy_score(y_test, predict))
```

2. A parking lot needs to improve its performance. Write a Python code to obtain the regression lines. The file Regression_data_sets.csv can be downloaded from CourseLink.

```
# This program takes as input 5 different datasets, i.e. one for
# each parking lot. Therefore, the original dataset was divided
# into five different datasets, and the program  thus takes
# as input "lot1.csv", "lot2.csv", "lot3.csv", "lot4.csv", and
# "lot5.csv", which were all submitted with the assignment
# submission. This data is processed through a regression
# algorithm, the output of which is a scatter plot with all of
# the data points for each of the 5 parking lots, in a addition
# to a regression line for each parking lot.

# import the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# this function estimates the w0 and w1 coefficients for each
# dataset, i.e. the y-intercept and the slope, respectively
def coeff_est(x, y):

    # get number of observations
    n = np.size(x)

    # calculate mean of x and y
    x_mean = np.mean(x)
    y_mean = np.mean(y)

    # calculate sum of squares for (x - x_mean) and sum of (x - x_mean)*(y - y_mean)
    SS_xy = np.sum(y*x) - n*y_mean*x_mean
    SS_xx = np.sum(x*x) - n*x_mean*x_mean

    # calculate the coefficients
    w1 = SS_xy / SS_xx
    w0 = y_mean - w1*x_mean

    return(w0, w1)

# this function plots the data points for all 5 parking lots on a
# single graph, and then plots the regression line for each of
# the parking lots
def plot(x, y, w, x1, y1, w1, x2, y2, w2, x3, y3, w3, x4, y4, w4):

    # plot the data points for all of the parking lots
    plt.scatter(x, y, color = "b", marker = "o", s = 5)
    plt.scatter(x1, y1, color = "g", marker = "o", s = 5)
    plt.scatter(x2, y2, color = "c", marker = "o", s = 5)
    plt.scatter(x3, y3, color = "m", marker = "o", s = 5)
    plt.scatter(x4, y4, color = "y", marker = "o", s = 5)

    # compute the parking lot occupancy predictions
    y_pred = w[0] + w[1]*x
    y_pred1 = w1[0] + w1[1]*x1
    y_pred2 = w2[0] + w2[1]*x2
    y_pred3 = w3[0] + w3[1]*x3
    y_pred4 = w4[0] + w4[1]*x4

    # plot the regression lines
    plt.plot(x, y_pred, color = "r")
    plt.plot(x1, y_pred1, color = "r")
    plt.plot(x2, y_pred2, color = "r")
    plt.plot(x3, y_pred3, color = "r")
    plt.plot(x4, y_pred4, color = "r")
    plt.xlabel('Days')
    plt.ylabel('Occupancy')
    plt.suptitle('Regression lines for 5 parking lots')
    plt.show()

# main function
def main():

    # import the datasets (note: this involves 5 different datsets, one for each parking lot)
    data = pd.read_csv("lot1.csv").values
    data1 = pd.read_csv("lot2.csv").values
    data2 = pd.read_csv("lot3.csv").values
    data3 = pd.read_csv("lot4.csv").values
    data4 = pd.read_csv("lot5.csv").values

    # split the 5 datasets into x and y (independent and dependent variables)
    x = data[:,0]
    y = data[:,1]
    x1 = data1[:,0]
    y1 = data1[:,1]
    x2 = data2[:,0]
    y2 = data2[:,1]
    x3 = data3[:,0]
    y3 = data3[:,1]
    x4 = data4[:,0]
    y4 = data4[:,1]

    # estimate the coefficients
    b = coeff_est(x, y)
    b1 = coeff_est(x1, y1)
    b2 = coeff_est(x2, y2)
    b3 = coeff_est(x3, y3)
    b4 = coeff_est(x4, y4)

    # plot the 5 regression lines (one for each parking lot)
    plot(x, y, b, x1, y1, b1, x2, y2, b2, x3, y3, b3, x4, y4, b4)

if __name__ == "__main__":
    main()
```

3. The file Clustering_Data_Sets.csv contains the data points. Write a Python code to group the data points in the file into 20 clusters using K – means clustering.

```
# This program reads in the data file "Clustering_Data_Sets.csv"
# and inputs this data into the k-means clustering algorithm.
# The output of the program is a scatter plot with all of the
# data points and the centroids for the 20 different clusters
# that are created. The coordinates of the 20 centroids are
# also printed.

# Step 1 - import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Step 2 - import the dataset
data = pd.read_csv("Clustering_Data_Sets.csv").values

# Step 3 - create an instance of the k-means algorithm with the number
# of clusters set to 20 and the intial centroids set to random
k_mean = KMeans(n_clusters=20, init='random')

# Step 4 - fit the algorithm to the dataset
k_mean.fit(data)

# Step 5 - get the centroids of the clusters and print them
centroids = k_mean.cluster_centers_
print("\nCentroids of clusters: \n\n", centroids)
```

```
# Step 6 - get the labels of the clusters (in case you want to look at these)
labels = k_mean.labels_

# Step 7 - create a plot of the 20 clusters with their centroids and show it
plt.scatter(data[:,0], data[:,1], s = 20)
plt.scatter(centroids[:,0], centroids[:,1], c = 'r', s = 20)
plt.xlabel("x")
plt.ylabel("y")
plt.suptitle("20 clusters with centroids using k-means")
plt.show()
```