CSE 482 Exercise 6 (Date: November 9, 2020, 11:59PM)

The purpose of this exercise is to help you learn how to apply clustering methods, predictive modeling methods, as well as association rule mining to your dataset. Follow the instructions below to complete the exercise.

1. Download the data set *animals.csv* from the class Web page (http://cse.msu.edu/~karimiha/cse482/slides/lecture13/animals.csv):
   a. Use pandas to load the animals.csv file into a DataFrame object named "data". After the file is successfully loaded, type data.head() to display the first 5 rows of the table.
   b. Extract the names of the animals into a Series object named "names":
      names = data['Name']
   c. Extract the class (type of animal):
      Y = data['Class']
   d. Extract the features of each animal into a DataFrame object named X by dropping the Name and Class columns from the data object.
   e. Apply k-means clustering to create 5 clusters. Extract the centroids of each cluster.
   f. Create a DataFrame object named clusters. The object has 2 columns: name and cluster of each animal:
      ```
      from pandas import Series
      clusters = pd.DataFrame(names)
      clusters['cluster'] = k_means.labels_
      clusters
      ```
   g. Calculate the Adjusted Rand Index of the k-means clustering solution.

2. Next you need to apply hierarchical clustering to the *animals.csv* dataset.
   a. Use scipy hierarchy module to apply complete link clustering to the algorithm (see example shown in lecture notes). First, you need to use the linkage function to generate the merged clusters. Next, you need to plot the resulting dendrogram. Make sure the dendrogram also shows the names of the animals.
   b. Extract the "flat clusters" from the hierarchy using fcluster command using a threshold of 1.1. Calculate the Adjusted Rand Index of the solution.
   c. Repeat parts (a) and (b) using Ward's hierarchical clustering. Find the appropriate threshold to "cut" the dendrogram in order to generate 5 clusters. Plot the resulting dendrogram and calculate the corresponding Adjusted Rand Index (make sure the threshold you use is for 5 clusters).

3. Download the data sets *train.csv* and *test.csv* from the class Web page. The data contains information about the date, temperature, humidity, and other environmental factors for the office. The goal is to predict whether the office room is occupied (1) or not (0). Implement the following steps in an ipython notebook:

a. Use pandas to load the train.csv file into a DataFrame object named "data". After the file is successfully loaded, type data.head() to display the first 5 rows of the table. Remove the "date" column from the table using the drop() function since the column is not predictive of the class (unless you extract the time of day and day of week information). Use the last column, Occupancy, as your target (class) variable.

b. Train the following classifiers (using 5-fold cross validation) on the train.csv file and calculate the average accuracy of the cross-validation for each method given below. Vary the hyperparameters of the classifier and draw a plot that shows the average cross-validation accuracy versus hyperparameter values for each classification method shown below:

   i. Decision tree (maxdepth = 1, 5, 10, 50, 100)
   ii. K-nearest neighbor (k = 1, 2, 3, 4, 5, 10,15)
   iii. Logistic regression (C = 0.001, 0.01, 0.1, 0.5, 1)

   Your code would look something like this:

```
hyperparams = [ .... ]
accuracy = []
for param in hyperparams:
    clf = CLASSIFIER_DEFINITION (  set param here )
    scores = cross_val_score(clf, X, Y, cv=5)
    acc.append(scores.mean())
```

   Use the plot to choose the hyperparameter that gives the highest average accuracy. Note: the hyperparameter chosen may not be the same with different runs due to the randomization used by the cross-validation procedure.

c. For each method above, train a model using the entire train.csv dataset (with the hyperparameter you had chosen from the previous step). Apply the model to the test.csv data. Draw a bar chart to compare their test accuracies using the 3 methods.

4. Download the apriori software from the following website http://www.borgelt.net/apriori.html. Download the data set *Titanic.csv* from the class Web page. Generate the association rules from the data with the following thresholds: minimum support =1% and minimum conference = 90%. Save the rules generated in a file called titanic.rules. Execute the program from ipython notebook.

Deliveries: Save your IPython notebook as **exercise6.ipynb** and submit it via D2L. Submit titanic.rules too.