CS 4232 / 5232 – Fall 2020 – Homework 2
Assigned: 02/19/2020
Due: 03/04/2020 at 11:59 p.m.
Submit your file(s) to Canvas and a hardcopy in class on 03/06/2020
Maximum Grade: 100 pts.
Attendance is graded separately

Objectives: The objectives of this homework are the following:

- Learn how to compute simple similarity and distance measures (SMC, Jaccard, Euclidean, Mahalanobis)
- Learn how to design and implement similarity measures.
- Learn how to build decision trees using Hunt's Algorithm.
- Learn how to use scikit-learn and caret to build decision trees for hyperparameter tuning.
- Learn how to use k-fold cross validation to estimate the generalization error.

Notes:

- This homework is to be done in groups of 2 people. There must be just 1 submission per group.
- Submit a single PDF file named: yourlastnames_hw2.pdf with the answers to your lab. To generate this PDF file, you **must use R markdown**.

Activity 1: (20 pts.) (Proof of Completion Due 02/05) Given the vectors X and Y as below, compute by hand the indicated similarity or distance measure:

a) (4 pts.) X = (0,1,0,1,0), Y = (0,1,1,0,0), SMC and Jaccard coefficient
b) (6 pts.) X = (-1, 9, 3, 2), Y = (10, 6, 2, 4), Euclidean distance, cosine similarity, correlation
c) (10 pts.) X = (3,-5),Y = (-3,5), Mahalanobis distance, where the covariance matrix is:

$$\Sigma = \begin{pmatrix} 30 & 2 \\ 2 & 10 \end{pmatrix}$$

You must show your work step by step.

Activity 2: (20 pts.) (Similarity Functions and Missing Values) Using R, do the following:

a) Download the Adult dataset from the UCI website. https://archive.ics.uci.edu/ml/machine-learning-databases/adult/
b) (8 pts.) Design (not implement) a similarity measure that can assess the similarity between two adults in the dataset. This similarity measure must only take into account the attributes: age, workclass, fnlwgt, education, marital-status, capital-gain, and native-country. This measure must return a number close to 1 if the two adults are very similar, and if they are not similar, it must return a number close to 0.

Remember that attributes of different types will have different similarity measurements and that in the end you want your function to output a single number. This means that your similarity function will be a (weighted) average of different similarity functions on individual attributes.

For this question, you must formally write down how you compute the similarity between any two rows. You must specify how you average the individual similarity functions. Carefully justify each aspect of your similarity measure.

c) (8 pts.) Using R, write a function called *adult_sim*, which implements the similarity function that you designed in part (b).

d) (4 pts.) Write a function called *top_k*, which receives as input an adult *A* (with the same attributes listed in part (b)), a positive integer *K*, and that returns the K rows of adults most similar to *A* along with their similarities. The rows in the result set of this query cannot have missing values.

Activity 3: (20 pts.) (Decision trees) Consider the following dataset:

id	Work Class	Education	Capital Gain	Earns ≤ 50 K?
1	Private	Bachelors	12,000	**N**
2	Private	Some College	49,000	**N**
3	Private	Bachelors	67,000	**N**
4	Public	Masters	2,000	**N**
5	Private	Masters	49,000	**Y**
6	Public	Bachelors	49,000	**Y**
7	Public	Some College	2,000	**Y**
8	Public	Some College	12,000	**Y**
9	Private	Masters	67,000	**N**
10	Public	Masters	67,000	**N**

a) (16 pts.) Use the dataset above and Hunt's Algorithm to train by hand a decision tree to predict if an "adult" earns less than 50K a year. Show your work step-by-step. For the impurity measure use entropy. To deal with continuous attributes, do as we did in class: sort the values of the continuous attribute in increasing order and then consider splits in between every value in that sorted array. *Don't forget to consider all possible (valid splits) for all attributes*.

b) (2 pts.) Compute the training error for your tree of Activity 3a.

c) (2 pts.) Compute the generalization error for each tree using a pessimistic estimate with $\Omega = 0.8$.

Activity 4: (20 pts.) (Fitting decision trees with R) (20 pts.) Using the Caret package/library and the Adult dataset used in Activity 2, do the following:

a) (1 pt.) Pre-process the adult dataset by removing all rows with missing values. You can do this with the *drop_na* function of the Caret package.

b) (6 pts.) Using the Adult *training set* and the CART algorithm of Caret (*rpart2*), build a decision tree to predict if a person earns less than 50K. In order to accomplish this, you will first use grid-search CV (use *train* and *trainControl*) with k = 5,

to find the "maximum depth" hyperparameter value in the range [1, 2, …, 8] such that it leads to the highest accuracy. Then, build a decision tree using this best value for the maximum depth on the training set. Print out the table with the accuracies for all the different hyperparameter values and then generate plot that R automatically creates with these values.

c) (1 pt.) "Pretty print" the tree obtained in Activity 4b using fancyRPartPlot and based on the tree that you found, write conclusions about the nature of dataset.
d) (2 pt.) Run the tree that you obtained in Activity 4b on the Adult test set and find its error rate/ accuracy.
e) (6 pts.) Repeat activity 1b, but using the C5.0 algorithm of the Caret package. This might require you to install the C50 package.
f) Run the C5.0 tree that you obtained in Activity 4e on the Adult test set and find its error rate/ accuracy.
g) (2 pt.) Compute the confidence interval with a significance level of 95% for the difference between the accuracy of CART (obtained in the test set in Activity 4d) and that of the C5.0 algorithm (obtained in the test set in Activity 4f).
h) (2 pts.) What do you conclude from the confidence interval of Activity 4g? Explain.

Activity 5: (20 pts.) (Fitting decision trees with scikit-learn) Using the scikit-learn package and the Adult dataset used in Activity 2, do the following:

a) (1 pt.) Pre-process the adult dataset by removing all rows with missing values. If you use Reticulate, you can reuse the same code that you wrote for activity 4a.
b) (14 pts.) Using the Adult *training set*, the CART algorithm of scikit-learn (*DecisionTreeClassifier*), and grid-search CV (use GridSearchCV) with k = 5, find the best value for the maximum depth hyperparameter in terms of the accuracy. Then, build a decision tree on the training set using this best value.
   Remember that scikit-learn might require you to create a Dummy Variable encoding (a.k.a. a one-hot encoding) of the categorical attributes. You can do this encoding in R with the *dummyVars* function of Caret and then pass it to scikit-learn, or you can do it in Python directly using *OneHotEncoder*.
c) (1 pt.) "Pretty print" the tree you obtained in part 5b. For this you can use *export_graphviz*. Is this tree different from the one obtained in Activity 4c? If so, what could be the reasons for the difference?
d) (4 pts.) Run the tree obtained in part 5b on the Adult test set and find the error rate/ accuracy. Compare this with the results of the final tree obtained in Activity 4.

Note: For Activity 5, you can either call scikit-learn from within R using the reticulate package, or use python directly, whichever you prefer.