COL 774: Assignment 3 (Part A)
Sem II, 2019-20

Due Date: Mar 31, 11:50 pm. Total Points: 30 (Part A) +
Notes:

- This assignment has two implementation questions.
- You should submit all your code (including any pre-processing scripts written by you) and any graphs that you might plot.
- Do not submit the datasets. Do not submit any code that we have provided to you for processing.
- Include a **single write-up (pdf) file** which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- You should use Python as your programming language. For Decision tree question (only), we may allow using C/C++/Java - but you need to confirm with us first.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you have submitted as well as your ability to explain your code.
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. We will also include submissions from previous years since some of the questions may be repeated. Any cheating will result in a zero on the assignment, a penalty of -10 points and possibly much stricter penalties (including a **fail grade** and/or a **DISCO**).

1. **(30 points) Decision Trees (and Random Forests):**
Machine learning has been deployed in various domains of computer science as well as other engineering fields. In this problem you will work on detecting files infected with virus on your system. You will work with VirusShare dataset available for download from the UCI repository. Read about the dataset in detail from the link given above. You have been provided with a pre-defined set of test, train and validation of dataset to work with (available for download from the course website). In this dataset, for any given example, a large number attribute values are missing (they can be thought of having as a 'default' value, *i.e.* 0). Correspondingly, you have also been provided sparse files and you can use **pyxclib** for reading and writing the sparse files. You have to implement the decision tree algorithm for predicting the virus infected files based on a variety of features. You will also experiment with Random Forests in the last part of this problem.

   (a) **(10 points) Decision Tree Construction** Construct a decision tree using the given data to predict which files are infected. You should use mutual information as the criteria for selecting the attribute to split on. At each node, you should select the attribute which results in maximum decrease in the entropy of the class variable (i.e. has the highest mutual information with respect to the class variable). This problem has all its attributes as integer (continuous) valued. For handling continuous attributes, you should use the following procedure. At any given internal node of the tree, a numerical attribute is considered for a two way split by calculating the median attribute value from the data instances coming to that node, and then computing the information gain if the data was split based on whether the numerical value of the attribute is greater than the median or not. For example, if

you have have 10 instances coming to a node, with values of an attribute being (0,0,0,1,1,2,2,2,3,4) in the 10 instances, then we will split on value 1 of the attribute (median). Note that in this setting, a numerical attribute can be considered for splitting multiple number of times. At any step, choose the attribute which results in highest mutual information by splitting on its median value as described above. Note that a large number of attribute values are missing for any given instance, and for this problem, it is safe to treat them as having a default value of '0' [1]. Plot the train, validation and test set accuracies against the number of nodes in the tree as you grow the tree. On X-axis you should plot the number of nodes in the tree and Y-axis should represent the accuracy. Comment on your observations.

(b) **(6 points) Decision Tree Post Pruning** One of the ways to reduce overfitting in decision trees is to grow the tree fully and then use post-pruning based on a validation set. In post-pruning, we greedily prune the nodes of the tree (and sub-tree below them) by iteratively picking a node to prune so that resultant tree gives maximum increase in accuracy on the validation set. In other words, among all the nodes in the tree, we prune the node such that pruning it(and sub-tree below it) results in maximum increase in accuracy over the validation set. This is repeated until any further pruning leads to decrease in accuracy over the validation set. Read the following notes on pruning decision trees to avoid overfitting (also available from the course website). Post prune the tree obtained in step (a) above using the validation set. Again plot the training, validation and test set accuracies against the number of nodes in the tree as you successively prune the tree. Comment on your findings.

(c) **(10 points) Random Forests:** As discussed in class, Random Forests are extensions are decision trees, where we grow multiple decision trees in parallel on bootstrapped samples constructed from the original training data. A number of libraries are available for learning Random Forests over a given training data. In this particular question you will use the scikit-learn library of Python to grow a Random Forest. Click here to read the documentation and the details of various parameter options. Try growing different forests by playing around with various parameter values. Especially, you should experiment with the following parameter values (in the given range): (a) $n\_estimators$ (50 to 450 in range of 100). (b) $max\_features$ (0.1 to 1.0 in range of 0.2) (c) $min\_samples\_split$ (2 to 10 in range of 2). You are free to try out non-default settings of other parameters too. Use the out-of-bag accuracy (as explained in the class) to tune to the optimal values for these parameters. You should perform a grid search over the space of parameters (read the description at the link provided for performing grid search). Report training, out-of-bag, validation and test set accuracies for the optimal set of parameters obtained. How do your numbers, i.e., train, validation and test set accuracies compare with those you obtained in part (b) above (obtained after pruning)?

(d) **(4 points) Random Forests - Parameter Sensitivity Analysis:** Once you obtain the optimal set of parameters for Random Forests (part (c) above), vary one of the parameters (in a range) while fixing others to their optimum. Plot the validation and test Repeat this for each of the parameters considered above. What do you observe? How sensitive is the model to the value of each parameter? Comment.

(e) **Extra Fun: No Credits!**: Read about the XG-boost algorithm which is an extension of decision trees to Gradient Boosted Trees. You can read about gradient boosted trees here (link 1) and here (link 2). Try out using XG-boost on the above dataset. Try out different parameter settings, and find the one which does best on the validation set. Report the corresponding test accuracies. How do these compare with those reported for Random Forests?

2. **Neural Networks:**
Coming Soon.

[1] think about why this makes sense. Are there any other ways that you can deal with these missing attribute values? Feel free to try alternate methods