COL 774: Assignment 3 (Part A)
Semester I, 2023-24

[Part A: Decision Trees]. Total Points: 44
[Part B: Neural Networks]. Will be released soon. Total Points:
Due Date (for both parts): Sunday Oct 29th, 11:50 pm.

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 write-up (pdf) file**, one (consolidated) for each part, 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 (one for each of the parts A and B).
- 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. **(44 points) Decision Trees (and Random Forests):**
   Imagine you've been hired by the BCCI as a data analyst due to your machine-learning expertise. Your initial assignment is to construct a decision tree-based classifier(s) to forecast the result of a white ball cricket match using various attributes obtained after the first innings is over, e.g., score, as well as meta-information about the match, such us countries playing, who won the toss etc. Your work will revolve around data obtained from the StatsGuru repository which has been scraped for this purpose. You've been provided with predefined sets of test, train, and validation datasets available for download from the course website. To gain a comprehensive understanding of the dataset, be sure to read the detailed information in the accompanying readme file. Please refrain from making the data publicly accessible and solely utilize it for the purposes of this assignment.
   You've also been given a starter code that facilitates file reading and basic tree data structure definition. However, it's worth noting that you have the flexibility to create your code from scratch, as the use of the starter code is not obligatory. Your primary task is to implement the decision tree algorithm for predicting cricket match outcomes based on a variety of features. Additionally, you'll explore the application of Random Forests in the later stages of this problem.

(a) **(15 points) Decision Tree Construction** Construct a decision tree using the given data to predict whether the team will win or lose the match. 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 the maximum decrease in the entropy of the class variable (i.e. has the highest mutual information with respect to the class variable). Remember some attributes are categorical and some are continuous. 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 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 times. At any step, choose the attribute which results in the highest mutual information by splitting on its median value as described above.
For categorical attributes, you have to perform k way split where k is a number of unique attribute values. For example, if the following instance of training data arrives at a node ($India, India, England, Kenya, Australia, England$) then splitting on this attribute will result in 4 children nodes. While testing if an example coming at this node have a different attribute value (For eg: $Pakistan$) then consider the node as a leaf node and return the prediction.
Experiment with different maximum depths specifically for $\{5, 10, 15, 20, 25\}$ and report the accuracy on train and test datasets. Feel free to experiment with other depths for performance gain. Plot the train and test set accuracies against the maximum depth in the tree. On the X-axis you should plot the maximum depth in the tree and the Y-axis should represent the train and test accuracies. Comment on your observations. Also, report the test accuracies for only win prediction and only loss prediction. Comment on your observation.

(b) **(5 points) Decision Tree One Hot Encoding:** Categorical attributes having more than 2 categories can be encoded using one hot encoding. In one hot encoding, each category of attribute is represented by a new attribute which has $0, 1$ value depending on the attribute value. Replace the above categorical attributes having more than 2 categories with respective one hot encoding attributes. For example, the attribute $team$ has the following categories $\{India, Australia, England\}$ then after replacement the attribute $team$ will be replaced with these 3 attributes $team\_India, team\_Australia, team\_England$ each having 2 categories $0, 1$. Repeat part (a) for this transformed dataset with maximum depths for $\{15, 25, 35, 45\}$. Compare the results with part (a) and comment on your observations. Feel free to experiment with other depths for performance gain. We will extend this version of the Decision Tree for the part (c) below.

(c) **(8 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 the sub-tree below them) by iteratively picking a node to prune so that the resultant tree gives a 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 the sub-tree below it) results in a maximum increase in accuracy over the validation set. This is repeated until any further pruning leads to a 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 trees of different maximum depths obtained in part (b) above using the validation set. Again for each tree 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.

(d) **(8 points) Decision Tree sci-kit learn:** As there are number of libraries which provide Decision tree implementation, we would like to compare our own implementation against these. Use the scikit learn library to build the decision tree on the dataset provided to you. Change the value of the *criterion* parameter to *entropy*. Next, (i) Vary max_depth in range $\{15, 25, 35, 45\}$. Keep all other arguments default. Report the train and test accuracies and plot the accuracies against maximum depth. Use the validation set to determine the best value of the depth parameter for your final model. (ii) Next, choose the default value of the depth parameter (which grows the tree fully), and vary value of the pruning parameters ccp_alpha, in the range $\{0.001, 0.01, 0.1, 0.2\}$. Use default value of all other parameters. Again, plot train and test accuracies against the value of the pruning parameter. Choose the value of the ccp_alpha parameter [1] which gives the maximum accuracy on the validation set, and

[1] You should read about the semantics of this parameter from Scikit's documentation.

use this as your final model. Now, compare the results obtained using the best model obtained in sub-parts (i) and (ii) above, i.e., with varying *max_depth*, and *ccp_alpha* parameters, with part (b) and (c) and comment on your observations. Note: Sci-kit's internal implementation uses a one-hot encoding to split multi-valued discrete attributes, so in some sense, results here are comparable to your implementations in parts (b) and (c), respectively.

(e) **(8 points) Random Forests:** Random Forests are extensions of 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 sci-kit 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 350 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 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 (c) and part (d) above?

(f) **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?