# Homework 5 Report

***Spam Dataset***

a) I did not use any additional features for spam.
b) My Decision Tree gave me 83% and Random Forests gave 84%. My Kaggle score was around 74%.
c) (29) <= 0.
(20) <= 0.
(30) <= 0.
(27) <= 0.
(4) <= 0.
(1) <= 0.
(10) <= 0.
(14) <= 0.
(13) <= 0.
(26) <= 4.
(18) <= 0.
(32) <= 1.
(17) <= 0.
(30) <= 4.
(25) <= 1.
(16) <= 0.
(3) <= 0.
(25) <= 0.
(28) <= 0.
(19) <= 0.
(30) <= 3.
(26) <= 0.
(30) <= 2.
(1) <= 0.
(1) <= 0.

d) (29) <= 0. (20 trees)
(20) <= 0. (20 trees)
(30) <= 0. (10 trees)
(17) <= 0. (7 trees)
(32) <= 0. (8 trees)
(7) <= 0. (15 trees)
(1) <= 0. (20 trees)
(26) <= 1. (2 trees)

(26) <= 0. (4 trees)
(26) <= 2. (3 trees)
(14) <= 0. (2 trees)
(11) <= 0. (1 trees)
(27) <= 2. (1 trees)
(31) <= 0. (1 trees)

***Census Dataset:***

a) I used some external code for the pre-processing step to handle the extra/missing features and their values. Otherwise, no extra features added.
b) My Decision tree gives me about 85% while my random forest did slightly better with 86.3%. My Kaggle score came to 82%.

c) (relationship) <= 1.
(education-num) <= 11.
(capital-gain) <= 5013.
(capital-loss) <= 1740.
(hours-per-week) <= 30.
(age) <= 33.
(education) <= 8.
(age) <= 27.
(native-country) <= 37.
(capital-loss) <= 0. =
(occupation) <= 4.
(capital-gain) <= 3103.
(capital-gain) <= 2407.
(age) <= 28.
(occupation) <= 5.
(hours-per-week) <= 40.
(age) <= 30.
(fnlwgt) <= 55291.
(fnlwgt) <= 105229.
(workclass) <= 5.
(fnlwgt) <= 167319.
(fnlwgt) <= 348152.
(fnlwgt) <= 185216.

d) (relationship) <= 1. (50 trees)
(education-num) <= 12. (24 trees)
(education-num) <= 13. (26 trees)
(fnlwgt) <= 210013. (1 tree)
(fnlwgt) <= 83064. (12 trees)
(age) <= 59. (6 trees)
(age) <= 39. (2 trees)
(occupation) <= 4. (20 trees)

***Pruning/ Additional Implementation(s) (PART 5)***

- My Decision Tree class is implemented in a way that each node of this class is either a list of the split arguments in the form [index to split on, threshold for split].
- I further have a isLeaf indicator that turns on/off based on where I am in the decision tree.
- Missing values are replaced with {} or 'NaN' in Matlab (external code implementation)
- For bagging I take out 30% of the overall data randomly.
- I speed my my segmentation process by only considering unique values of a column as appropriate thresholds. This sped up the process from an initial run-time of 3 minutes to a mere 2 seconds.
- Impurity criteria is just the information gain function implemented in lecture.
- Whenever I have a confusion, classifier predicts the optimal label to be the mode of the remaining label and makes the decision.
- I make sure that my tree wont classify until 2 criteria are met:
    - More 99% of the remaining labels are the same.
    - The tree will classify regardless of label at a certain depth (around 25 for the tree and 12 for a tree in the forest). It will find the mode of the remaining labels and make a guess.

Random forest techniques are described above. The only modification that I included was considering the depth of the tree as a hyper parameter and tuning to find both optimal depth as well optimal number of bags.