Georgia Tech Scheller College of Business

MGT 6503 – Spring 2025

Group Assignment #1
Decision Trees and Random Forests

Points: 160

Primary contact for questions related to the assignment – Yan Wu (TA - contact info in the syllabus). You can also CC Diego and the instructor. Use email (not MS Teams or Discussions in Canvas).

Assignment due date: Sun, Mar. 30, 2025 - 11:59 pm (1 min. before midnight).

Submission:

- All submissions must be made **electronically** through the **"Assignments" tab in Canvas**.
- **No late submissions accepted**. Submission in Canvas will close automatically at the deadline.
- This is a **group** assignment. You are free to work solo or you can work in teams of up to 4 people (but only with peers from your **OWN** section). You have to email the TA (Yan Wu) by Mon, Mar. 24, 2025 with your team info (one email per team).
- Submission – **3** files (main write-up, **one** R script file for the problem, and E1_ind.csv).
- Make **only one** submission **per team.** One team member will submit in Canvas from his/her account - the others do not submit. All team members get the grade.
- Make sure that your **team member names are included in your answer files**.

Format:

- Main write-up should be in **doc/docx** or **pdf** format (text and printscreens)

Collaboration rules:

- Collaboration among members of different teams is **NOT permitted** beyond general discussions of methods. Sharing your assignment work (files) with other groups is not allowed under any circumstance. Copying answers from other teams is considered academic cheating and will be sanctioned according to GT rules. You have to write your own answers (as a team).

Exercise: Decision Trees and Random Forests [160 points]

Dataset significantly altered and expanded from:

https://www.kaggle.com/datasets/bhavikjikadara/loan-status-prediction

Background (fictitious):

General Premise - Consider the EU retail banking sector, in an alternate reality. Many consumers would love to secure a mortgage loan (especially given the meteoric rise of real estate prices – fear of missing out is real!!!). Suppose that financial institutions have a way of sharing across the entire industry whether or not consumers have been previously declined mortgage loan applications (similar to the way US home insurance operates when there are claims – think of a water damage claim – all insurers in the industry will know you filed that claim with your specific insurer).

- Due to a tough economic landscape, a loan application decline makes the consumer “radioactive” to other institutions – once a bank declined a loan application from a consumer, that consumer cannot secure a loan from any other bank in the market.
- From the beginning, (potentially unfairly), banks collude to decline mortgage loan applications from unemployed consumers

EU regulators are growing increasingly concerned that the loan offering process is becoming unfair and potentially life-changing to some consumers. Hence, they are trying to create prediction models to better understand the loan application acceptance process at various banks, prior to initiating legal action and scrutinizing deployed algorithms in detail. The regulators use decision trees and random forests for that preliminary stage of the investigation.

Zoom in:

In this exercise, we focus on GT Europe Bank (GTEB), the European Union subsidiary of GT World Fund (GTWF), a global financial conglomerate. Among many products and services, GTEB offers mortgage loans as well.

The regulators have access to a dataset containing recent loan application and acceptance decisions at GTEB. The regulators want to investigate what factors influence the acceptance process at GTEB.

*** Note that the GTEB dataset does not include any loan applications from unemployed consumers (since, as mentioned above, it is a coordinated practice in this fictitious industry to automatically decline mortgage loan applications for such consumers).

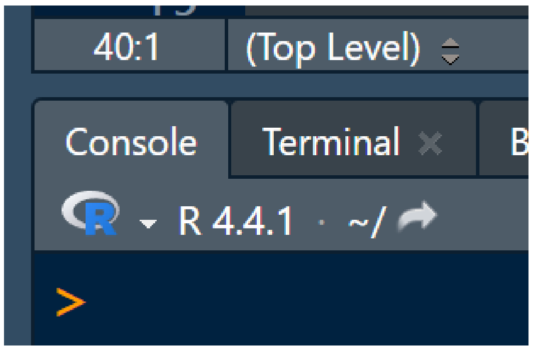
(a) Open RStudio but do NOT create yet a new file. **BEFORE** you create the R code to build your decision trees and random forests simulation, you need to first create the index of positions in the training data that will correspond to the training set. For this exercise, we are changing the sequence of commands compared to the in-class demo.

- Note – this index of positions is just a list of positions (i.e., integer numbers) for the records in the dataset that will be assigned to the training set. For example, something like [2 57 320 23 134 …]. This index list does NOT contain data from the training set. We will take the records from the dataset at the positions in the index list and put them in the training set soon (after a few more steps).

In the in-class demo, in file decision_tree_gt_mgt.R, we created this index of positions on line 51.

We want you to create a similar index of positions for this problem. Your dataset in this assignment contains 400 records. So basically consider the command in the in-class demo but replace "nrow(titanic)" with 400.

Task – Go to the console window (bottom left window)



and run command

```
ind = sample(1:400, floor(400*0.70))
```

This will create a vector of 70% of the numbers from 1 to 400, chosen randomly.

Resources:

Lecture 2s – file decision_tree_gt_mgt.R

Lecture 2 - R Tutorial – Decision Trees and Random Forests - Step 5

(b) Save that list on your system

You will use **write.csv** function with parameter setting "row.names =FALSE" (no counter column) to save the "ind" indices (the list that you created in step (a)) into a file called "E1_ind.csv". That file should contain ONLY one column - in most cases with header "x". You are required to submit this file (E1_ind.csv) with the rest of your code. Your command line should look something like this:

```
write.csv(ind, row.names = FALSE, "[here put your own file path on your system]/E1_ind.csv")
```

*** The file path represents the folder destination where you want the file written. The last part (E1_ind.csv) is the file name and extension for the file to be created in that folder.

*** If you do not specify the file path (i.e., in which directory on your system you want the file to be written, R will create the file in the working directory).

i. First, learn how to use the R command "write.csv". Open ChatGPT (or equivalent gen AI tool), and ask it to **explain** to explain the R command line write.csv(ind, row.names = FALSE, "E1_ind.csv"). Note that for the explanation from ChatGPT, I am not asking you to write the complete file path. Grab a printscreen of both your prompt and the answer from ChatGPT and past it into your main write-up.

ii. Go to the console window and run command:

```
write.csv(ind, row.names = FALSE, [here put your own file path on your system]/E1_ind.csv")
```

Run this command only once. You want to store the index list but you do not want to keep overwriting it by mistake. In the R script file, you will import this index list and will create a training set based on it. So, every time you will run your R script file, you will generate the same tree (since you will be importing that same index list from E1_ind.csv) and we (instructor and TAs) can recreate that tree as well.

Make sure you know where the file E1_ind.csv is:

- You will need to submit it along with the rest of your assignment
- You may need to specify the path to it on your machine in order to import the data from it

Submit "E1_ind.csv" as part of the assignment.

(c) Download **loan_application.csv** from Canvas. You can see the data description below. Read this info carefully as it will explain each variable.

Data Description

Column ID	**Variable Description**
loan_id	loan_id – consecutive positive integers capturing the unique position in the database
gender	Gender of the applicant (Male/Female)
married	Marital status of main loan applicant– Yes/No
dependents	How many dependents (for tax purposes) does the main loan applicant has (non-negative integers – 0,1,2,3,…)
education	Highest education level for the main loan applicant: • 1 – highschool or lower • 2 – undergraduate university degree • 3 – masters degree • 4 – graduate degree other than masters (doctorate or degree from graduate schools such as law or medical school)
employment	Type of employment (remember that in this problem, unemployed applicants are screened immediately and declined the loan – thus, this dataset includes only the employed applicants that make it to subsequent evaluation rounds) • 1 – self-employed • 2 – employed by an institution based on a contract
coapplicant	Does the loan include a co-applicant? • 0 – no other co-applicant on the loan • 1 – there are other co-applicants on the loan
total_income	Income of all applicants (main applicant and co-applicant, if any) put together Unit is 1,000 EUR (e.g., 434 corresponds to 434,000 EUR)
dti	Debt-to-income ratio for main applicant
total_loan	Total amount of the loan (as requested by the consumer) Unit is 1,000 EUR (e.g., 275 corresponds to 275,000 EUR)
loan_term	The loan term in months (for example, a 20-yr loan has length 240).
credit_score	A credit score (equivalent to the one in US but computed according to a very secretive formula – scaled to match values in US)
property_type	Type of the property for which the mortgage loan is taken 1 – rural / 2 – semiurban / 3 - urban
loan_status	Loan decision: • 0 – loan application denied • 1 – loan application approved

- All data points are reported as current at the time of the loan application

***Note – your instructor added some correlation between some variables – so when you build the decision trees, keep in mind that some variables might capture to some extent similar patterns. As such, you may want to play with which variables to use for your decision tree. You may not need/want all 14.

Now it is time to build the R script file for the decision tree and random forest analysis, and run that analysis

(d) Open RStudio and create a new R Script file (File-> New File->R Script).
Add the names of the members of your team in a comment (starting with #) at the top of the file.

Keep/save ALL of your code (that you will create based on the tasks below) in this script file. Essentially (apart from moving upfront the creation on the index list of positions for the training set), we want script files similar to those used for demo in class.

(e) If you have not done so already - install the required libraries "**rpart**", "**rpart.plot**", and "**tidyverse"** using **install.packages()**. You can run these commands directly from the console (bottom left) or also put them in the script file and run them from there.

(f) Once you have the above libraries on your computer (based on the in-class demo) just use **library()** command to load the libraries. Write the code in the R script file.

Resources:
Lecture 2 – file decision_tree_gt_mgt.R
Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 2

(g) Import data from the file **loan_application.csv** in R studio using **read.csv()** function into a data object. You choose how to name that data object. Write the code in the R script file.

*** Note – if your file is not in the working directory for R, then you will need to specify the file path.

Resources:
Lecture 2 – file decision_tree_gt_mgt.R
Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 3

(h) Run the **summary()** function to get descriptive statistics for your data and report those stats.

Write the code in the R script file.

[main write-up] Report the following:

i. Minimum, maximum, and mean value of *total income*.

ii. Median value of the *debt-to-income* ratio

iii. Get the frequency count of the *loan status* (declines vs acceptances as coded 0 / 1).

Hint: Convert loan_status into a factor variable as follows (replacing "yourdataname" with the name of the data object created in part g):

```
summary(as.factor(yourdataname$loan_status))
```

iv. Comment on these statistics you calculated for each of the subsections i-iii

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 4

(i) Import the list of indices (created and saved during steps a-b) for the records that will be in the training set.

i. Import from E1_ind.csv (which you created for Problem 1) into a list of indices E1_ind (not a whole data object) using command read.csv but selecting the specific column. If your header was "x" in the file, you need to add "$x" at the end. Here is an example of the command:

```
E1_ind <-read.csv("E1_ind.csv")$x
```

Technically your initial "ind" and "E1_ind" will be identical. Use E1_ind list of indices in the code. As mentioned above - we (TAs or instructor) will rerun your code reading this file and will recreate a decision tree identical to yours to check your answers.

*** The reason we did NOT want for parts (a) and (b) to be part of the script file is that we do not want you to re-sample the training set every time you re-run the code (imagine the 4 members of the same team working on different parts of the assignment, all getting different trees because they re-ran the code on their separate computers).

(j) Partition the dataset into the **training** and **testing** subsets using using E1_ind list of indices (which, as a reminder, has 70% of the positions from 1 to 400, chosen randomly). In particular, look at lines 52-53 in the lecture demo file. Write the code in the R script file.

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 5

(k) Create a decision tree using **rpart()** and **rpart.plot()** functions.

i. Run the recursive portioning algorithm **rpart()** on the **training** subset to try to predict the loan approval decision process (i.e. loan_status variable) at GTEB**.** Review the Titanic example (lines 59 and 68 in the in-class R script demo file). Write the code in the R script file. See the hints below.

Requirements:

- **1. Use at least 6 different predicting variables (at least 2 binary and at least 3 non-binary).** Do NOT use loan_id (is is just an integer record keeping counter).
- **2. You are required to use "credit_score" variable**
- **3. Your tree has to have at least 5 levels but no more than 7 levels of depth on at least one of the branches (in counting depth, we include the top node – that is level 1).**
    - The depth of the tree is the number of nodes along the longest path/branch (counting by number of nodes, going only downwards) between top node and a leaf. Note that some branches (left vs. right) may have different depths. Rpart.plot tends to put all leaves (regardless of depth along that path) at the bottom so you need to count path length along the splits.
    - See the hint and instructions below on how to generate deeper or shallower trees

Hint + additional instructions:

This will be an **iterative** process, trying different combinations of variables, different "cp" values and creating a plot of the tree. If your tree is too deep or too shallow (in terms of the allowed levels of depth), one of the ways to adjust depth is to play with argument "**cp**" for the **rpart** function. *See the R script file for the in-class demo for the Titanic data analysis using decision trees for an example of the "cp" argument – line 68*. This argument dictates that, if the improvement in purity is not above a certain threshold, the group in question should not be split further. So, the larger cp value is, the fewer leaves and shorter branches the tree will have. But, if you set cp too high, then the tree will not get rid of impurities well. Therefore, use cp to prune but be careful not to overdo it.

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 6

ii. Make a **deliberate mistake** in the syntax for the generation of the decision tree (the equivalent of line 68 in the demo file – when using rpart function together with cp parameter). Keep the mistake simple (like removing a comma or a quotation mark or a plus sign). Prompt ChatGPT (or equivalent gen AI tool) to help you fix (debug) the syntax error. Attach a screenshot of the prompt to ChatGPT and the solution offered. After you are done, restore the code.

iii. Visualize the decision tree. Write the code in the R script file. Attach the screenshot of the tree graph in the main write-up.

Note that in the R demo script in class for decision trees (Titanic example) I presented you with several options to plot trees, by changing the rpart.plot argument "**extra**" for the analysis related to the Titanic. In particular, look at lines 61, 70 – argument "**extra=104**". That will render a graph where each label contains on the second row two numbers (indicating the probability of each outcome within that group of customers). This is one of the better approaches for readability. Use that one.

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 7

iv. In your tree, examine the **importance** of the predictor variables.

1. Use code similar to lines 90-91 in the R demo file to retrieve importance of the predictors. Write your code in the R script file. Grab a printscreen of the importance output and add it to the main write-up file.

   I> Which predictor is among the least important but still plays a "visible" role in the tree structure (shows up as part of a splitting logical test in the tree)?

   II> Are some of your predictors that you used in rpart() *not* showing up in the output tree? If yes, why do you think that is the case?

2. Comment on the results of the classification, referring to the statistics displayed on the nodes, in conjunction of the derived variable importance. What do you see in the ordering/magnitude of importance scores of predictors that corroborates the "story" told by the tree structure?

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 9

(l) Adjust your visualization.

i. Increase the size of the text in the nodes using the **tweak** argument for rpart.plot function (make tweak equal to a value between 1 and 1.5 – you choose the tweak level so that your tree is best rendered). Write the code in the R script file.

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 7

ii. Add a title to the decision tree visualization using **main** argument for rpart.plot function. Write the code in the R script file.

Resource:

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 7

iii. Add the screenshot of your visualization in the main write-up.

(m) Compute the improvement in purity (**reduction in impurity**) for the top node

i. Rerun rpart(…) with argument **"extra = 1"** which will give you the number of observations that fall in each node.

ii. [main write-up] Paste a screenshot of the tree with "extra=1" in the main answer file.

iii. [main write-up] Calculate the improvement (reduction) in impurity achieved during the first split in your tree (i.e., the split at the top/root node). Describe the details of the computation (impurity at root node and at each child node, and weights for each node). Write the complete argument and result in your main file. **Hint: fill out the following table first to help you organize the calculation process.**

	Count for class 0	**Count for class 1**	**Impurity**	**Weight**
Top (root) node				
Left child of root node				
Right child of root node				

Resource:

Lecture 2— lecture slides

(n) Use **predict()** function to predict the loan application outcome in the **testing** subset using the model you created based on the **training** subset. Use function **table()** to create the corresponding **confusion matrix**.

Write the code in the R script file[main write-up]

In the main write-up, provide the screenshot of the confusion matrix (the predicted versus actual outcomes – that should be in your console once you run the code). Discuss the outcome. How many of the actual loan application approvals did we misclassify as declines? Based on your opinion, were we able to predict well the GTEB loan approvals using our model?

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 8

(o) As it happens, in EU, using ***credit_score*** variable for automation of approval decisions is highly controversial and, more recently, prohibited (apart from a few exceptions) under GDPR legislation (https://www.loc.gov/item/global-legal-monitor/2024-01-10/european-union-court-of-justice-rules-credit-scoring-constitutes-automated-individual-decision-making-under-gdpr/).

Thus, GTEB can no longer use that variable (*credit_score*) for the loan approval decision making process. In turn, the regulators need to adjust their prediction model accordingly as well to try to replicate GTEB's updated model.

Tasks:

i. Prompt ChatGPT (or equivalent gen AI tool) to **summarize** the article at the web link above. Instruct ChatGPT to give you the main takeaway. Add a screenshot of the prompt and output to the main writeup.

ii. Repeat steps (k.i and k.iii) but WITHOUT using **credit_score** variable (use other variables). Write the code in the R script file.
Take a screenshot of the tree and paste into the main write-up.

iii. Repeat step (n). Write the code in the R script file and add the confusion matrix in the main write-up.

iv. Comment on the findings – at a high level, how are the tree structure and predicting outcomes changing based on whether or not you include *credit_score* among the considered predictors?

v. By looking at your dataset, and at the two trees (with and without *credit_score* variable) that you generated – can you identify a loan record in the data (either in the training or in the test set – does not matter) that is classified differently (one tree predicts that that loan would be declined while the other tree predicts that it would be approved)? If you cannot find such a record in the current dataset, can you generate (i.e., make up) a record (with all 14 columns), that would be classified differently by the two trees? Include in the main writeup the record and how it would be classified under the two trees.

(p) Create a **random forest** to predict loan_status (lines 97-128 in the R demo file)

i. Load the proper libraries (caTools, randomForest – make sure to have them installed first on your system). Write your code in the R script file.

ii. Remove all records with missing data from the training and test sets so that you can calibrate a random forest. Write your code in the R script file.

iii. Set the seed for the random forest code to 2024. Write your code in the R script file.

iv. Using the same training set as in step (o), same decision parameters (without *credit_score*) as in step (o), and adding an extra variable you did not use before in generating your initial tree (again, cannot be *credit_score*), create a **random forest** with 400 trees for the same classification purpose. See command line 115 in the demo file. Set parameter mtry of the randomForest function, to allow only three variables to be considered randomly at each split. Write your code in the R script file. Use ChatGPT (or equivalent gen AI tool) to **understand** how parameter "mtry" works. Take a screenshot of your prompt(s) and the answer(s) and include it(them) in the main write-up.

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Steps 10 and 11

(q) Examine the **importance** of variables in your random forest. How does it compare to the importance of the variables from your initial tree from step (o)? Write your code in the R script file. Also add a screenshot of the importance table and write your answer in the main write-up file.

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 12

(r) OOB error analysis:

i. Check how the OOB error stabilizes with the number of trees by plotting the random forest (line 121 in the demo file). How many trees roughly (by eyeballing the plot) are needed for the OOB error to stabilize? Write your code in the R script file. Also attach a screenshot of the plot and write your answer in the main write-up file.

ii. Prompt ChatGPT to explain the significance of the OOB error stabilizing beyond a certain number of trees in the random forest. Add a screenshot of the prompt and output in the main writeup.

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 13

(s) Create a confusion matrix for the random forest (lines 124-128 in the demo file). Describe what you observe and how the confusion matrix associated with the random forest differs from or is similar to the confusion matrix for the initial tree you derived in step (o). Write your code in the R script file. Also write your answer in the main write-up file.

Resources:

Lecture 2 – file decision_tree_gt_mgt.R

Lecture 2 – R Tutorial – Decision Trees and Random Forests - Step 14

(t) Make sure all the codes are in the script file in the above sequence of tasks and then save the file (if you did not do that already). Name the script “Assignment1.R”. Submit it together with the main write-up and other required files.