# Assignment 3: Hands-on with Decision Trees and Linear Regression

8 points

**Abstract**

**ATTENTION: this assignment should be completed individually. And I will use tools to check your codes against your peers' submissions!** This assignment consists of two problems. For each problem, the instructions are designed to modify the data sequentially. In other words, each step should be performed on the resulting data frame from the previous step.

## Problem 1: Predicting Income using Decision Trees (5 points)

For this problem, you will be using the Adult dataset from UCI. You can get this dataset "adult.data" from Canvas. The goal is to use a decision tree model to predict the binary variable income ($> 50K$ or $<= 50K$) based on the other attributes in the dataset. Read the attribute information from the file of "adult.names" on Canvas (you can use R to open this file).

- 1. (0.3 points) Load "adult.data" into a dataframe in R. Note that "adult.data" does not have column names in the first line, so you need to set **header=FALSE** when you read the data and then manually set the column names

```
colnames(adult_data) <- c("age", "workclass", "fnlwgt", "education",
"education_num", "marital_status", "occupation", "relationship",
"race", "sex", "capital_gain", "capital_loss", "hours_per_week",
"native_country", "income")
```

  There are some missing values in this dataset represented as " ?" (Note: there is a space before ?). Make sure that all " ?" are converted to NAs. You can do so by setting "na.strings" parameter in **read.csv** to " ?".

- 2. (0.2 points) Inspect the dataset using **str** and **summary** functions.

- 3. (0.2 points) For each variable in the dataset, is it numeric or categorical? And for each categorical variable explain whether it is nominal or ordinal.

- 4. (0.2 points) Set the random seed "set.seed(2025)", and split the data to train/test. Use 80% of samples for training and the remaining 20% for testing. You can use **sample** (see what we did in decision tree code demo $train\_sample < -sample(1000, 800)$, but you need to adjust 1000 and 800 according to the size of your dataset). Alternatively, you can use **createDataPartition** method from **caret** package.

- 5. (0.5 points) **Read the section on "Handling Missing Data" in chapter 13 of the textbook "Machine Learning with R".** Find which columns/variables in the train and test set have missing values. Then decide about how you want to impute the missing values in these columns. Explain why you chose this imputation approach (Pay attention to data leakage! You do not want to inadvertently introduce information from the test set into the training process, as it can lead to overly optimistic performance estimates and poor generalization to new data.).

- 6. (0.5 points) The variable "native-country" is sparse, meaning it has too many levels, with some levels occurring infrequently. Most machine learning algorithms do not work well with sparse

data. One-hot-encoding or dummy coding of these variables will increase feature dimensions significantly and typically some preprocessing is required to reduce the number of levels. **One approach based on frequency** is to group together the levels which occur infrequently. For instance, one could combine together countries with less than 0.1% occurrence in the data to an "other" category. **Another possibility is to use domain knowledge**; for instance, combine countries based on their geographic location ( "Middle East", "East-Europe", "West-Europe", etc. **Please read the section on Making use of sparse data (remapping sparse categorical data) in chapter 13 of the textbook "Machine learning with R".** Then combine some of the infrequent levels of the native-country. You can decide whether you want to combine the levels based on frequency or domain knowledge. Either one is fine for this assignment but preference will be with a choice that would increase the cross validation performance of the ML models you will train subsequently. (You do not need to choose the better one, this last sentence is just for your knowledge on how to determine the better strategy here!)

- 7. (0.5 points) Use appropriate plots and statistic tests to find which variables in the dataset are associated with "income". Remove the variable(s) that are not associated with income ((significance level = 5%)). You can check the Bivariate analysis table at the end of this assignment.
- 8. (0.2 points) Convert the **income** attribute to factor by specifying the levels. Depending on how you read the dataset, there might be a space before " $<= 50K$" and " $> 50K$", so ensure the levels are correctly defined.
- 9. (0.5 points) Train a decision tree model on the train data (preprocessed and transformed using above steps) using the C50 package and use your trained decision tree to predict "income" for the test data.
- 10. (0.5 points) Get the cross table between the predicted labels and true labels in the test data and compute the **total error**.
- 11. (1 point) Repeat steps 8-9, but this time, use a C5.0 decision tree model with "trials = x" to apply boosting, where "x" is a user-specified number to indicate how many trees you would like to use (refer to the decision tree code demo for an example). Choose your own x and compare the boosted model's total errors on testing sets with that of the previous general decision tree and explain it.
- 12. (0.4 points) You do not need to code for this question. When using C5.0, we have various parameters to control the tree's behavior, see this link. One of them is "minCases". Describe this parameter in your own words and explain with details whether increasing or decreasing its value leads to **overfitting**. Hint: Consider how "minCases" affects tree depth (tree size) and we know large trees tend to overfit.

# Problem 2: Predicting Student Performance (3 points)

For this problem, we are going to use UCI's student performance dataset. The dataset is a recording of student grades in math and language and includes attributes related to student demographics and school related features. Go to Canvas, download and unzip "student+performance.zip". You will be using "student-mat.csv" file. The goal is to create a regression model to forecast student final grade in math "G3" based on the other attributes.

- 1. (0.1 points) Read the dataset into a dataframe. Ensure that you are using a correct delimiter to read the data correctly and set the "sep" option in **read.csv** accordingly.
- 2. (0.3 points) Explore the dataset. More specifically, answer the following questions:
  - a. Is there any missing values in the dataset?
  - b. Draw a histogram of the target variable "G3" and interpret it.

- 3. (0.1 points) Split the data into train and test. Use 80% of samples for training and 20% of samples for testing. You can use **sample** (see what we did in decision tree code demo $train\_sample <- sample(1000, 800)$, but you need to adjust 1000 and 800 according to the size of your dataset). Alternatively, you can use **createDataPartition** method from **caret** package.

- 4. (0 point) Set the random seed: set.seed(2025)

- 5. (1 point) Use caret package ("trainControl") to run 10-fold cross validation using linear regression method on **the train data** to predict the "G3" variable with all the available features. Print the resulting model to see the cross validation RMSE. In addition, take a summary of the model and interpret the coefficients. Which coefficients are statistically different from zero (significance level = 5%)? What does this mean? (Hint: see linear regression code demo, summary(ins_model), check $Pr(>|t|)$ column and comments in that part.)

- 6. (0 point) Set the random seed again. We need to do this before each training to ensure we get the same folds in cross validation. set.seed(2025) so we can compare the models using their cross validation RMSE.

- 7. (0.7 points) Use caret and leap packages to run a 10-fold cross validation using step wise linear regression method with backward selection on **the train data**. By default, the train function with method = "leapBackward" reports the best models with up to 4 features (nvmax = 4). This means that the best models are restricted to a maximum of 4 features (predictors). To allow models with any number of features up to the total available, we need to update "nvmax" setting. Specifically, in your "train" function, add the parameter "tuneGrid = data.frame(nvmax = 1:n)", where n is the total number of features you use to predict "G3". After training, which model (i.e., with how many variables or nvmax) achieves the lowest cross validation RMSE? (Hint: see "leaps" code section in linear regression code demo)

- 8. (0.3 points) Take the summary of "finalModel", which features (variables) are selected in the model with the lowest RMSE? (Hint: see "leaps" code section in linear regression code demo)

- 9. (0.5 points) Compare the model obtained in Question 5 with the best model from Question 7 in terms of cross-validation RMSE. Determine which model performs better at predicting "G3". For the better-performing model, write out its linear regression equation and use it to make predictions on the test data. Finally, compute and report the RMSE on the test data.

## What to turn in

You need to create an R notebook consisting of **your answers/analysis** to the questions outlined above for problems one and two **together with your R code** you used to answer each question.

## The submission must be in two formats

- A html file; You run all the code cells, get all the intermediate results and formalize your answers/analysis, then you click "preview" and this will create a html file in the same directory as your notebook. **You must submit this html file or your submission will not be graded. Please note that if you knit your R notebook once, then you will lose the "preview" button! So do not knit!**

- An Rmd file which contains your R notebook.

| | Numeric | Ordinal | Nominal |
|---|---|---|---|
| Numeric | Pearson;<br><br>Scatter Plots | Kendall or Spearman;<br><br>Scatter Plots | If nominal variable has two groups:<br>- Two sample t-test<br><br>If nominal variable has more than two groups:<br>- ANOVA<br><br>Side by side boxplots |
| Ordinal | Kendall or Spearman;<br><br>Scatter Plots | Kendall or Spearman;<br><br>Scatter Plots | Kruskal-Wallis test;<br><br>Side by side boxplots |
| Nominal | If nominal variable has two groups:<br>- Two sample t-test<br><br>If nominal variable has more than two groups:<br>- ANOVA<br><br>Side by side boxplots | Kruskal-Wallis test;<br><br>Side by side boxplots | Chi-Square test;<br><br>Mosaic Plots |

Table 1: Bivariate analysis: what of statistic test and plot should you use?