## Part II :   Problems R coding

4. (3+6+7+7+7+8+8 = 46 points)  (*Analysis of a Credit Score Data Base*)  In this problem, we will be applying some different ML algorithms with the aim of better understanding what is important when your credit score gets computed.    Credit scores get used to assess the creditworthyness of customers for an assortment of purposes.  For example, landlords will often ask for credit scores/reports of potential tenants.   Banks and car dealerships will use your credit score to help determine whether to offer you a loan, and higher credit scores typically result in lower interest rates.  You can download the data frame (in R Format) using the following link:

`https://stanoyevitch.net/gateways/MAT495ML/CreditScores.RData`

directly to your R session.   This dataset contains 75,780 observations with 21 variables.  Each row represents a person in the database.  For most parts of this problem the variable, `Credit_Score`, whose value is either Poor, Standard, or Good, will be the target variable.

NOTE:  This data set was downloaded from Kaggle, and cleaned up a bit.  Please see the home Kaggle page for this data set to obtain more details about it:  `https://www.kaggle.com/datasets/parisrohan/credit-score-classification`

For each part below, make sure to include ALL of the relevant R code you use to get your answers.

(a)   Of the 21 variables of `CreditScores`, how many of these are factors (=categorical variables)?
(b)   Create a subset training set of row indices of 25,000 subjects from the `CreditScores` set by running exactly the following commands:

```
   set.seed(2);  train = sample(1:nrow(CreditScores), 25e3)
```

Using just the training subset of data, use an appropriate R function from the `randomForest` package (that you will need to install) on the `CreditScores` data frame to create a random forest model to predict the variable `Credit_Score`. Name your model as `rf.CreditScores` Use the default setting in your R function that creates the model using all of the predictors. Right before you create the model, enter the command `set.seed(2)` to ensure reproducibility of the results. Compute the test error (on the data held out of the training data).

(c) Create a modified data frame `Credit_Scorev2` from the original data frame by doing the following two things: delete all categorical predictor variables (except for the target variable `Credit_Score`), then scale all of the numerical variables using R's built-in scaling utility. Use all of the non-target variables of this data frame and the 5-nearest neighbor algorithm with the training rows of part (b) to make predictions for all of the test cases. (Make sure to enter the command `set.seed(2)` right before doing the 5-nearest neighbor algorithm.) What is the resulting test error? Repeat this with the 201-nearest neighbor algorithm.

(d) Create a single "big tree" model for the target variable `Credit_Score` and using all other variables of `CreditScores` as predictor variables. Call your model as `tree.CreditScores.big`. In the rpart algorithm, use "cp = 0.001" but all other settings as default. Make sure to enter `set.seed(2)` right before you build the big tree.
(i) What is the test error for this tree (corresponding to the test and training sets from part (b))?
(ii) If we wanted to prune this tree, what would be a good cp value, according to the method explained in class?

(e) Build a linear regression model to predict the variable: `Monthly_Balance`
using the variables: `Monthly_Inhand_Salary, Num_Credit_Card, Interest_Rate`
and using the training set of indices that you created in part (b).

- (i) Compute the test error for this model as the square root of the MSE
- (ii) Which of the three input variables is the most important predictor for the model?

(f) (*Cleaning the data set*) As we had mentioned, we partially cleaned the data set as it originally appeared in Kaggle, but there are still some things that need fixing. One of them is the variable: `Payment_Behaviour`
(i) One of the values of this variable does not make any sense. Create a new data frame `CreditScoresv3` from the original data frame that has all rows with this strange value of being deleted, but everything else is the same.
(ii) Next, for each of the "high spent" levels of the categorical variable `Payment_Behaviour` in `CreditScoresv3` find the percent of people in that category that have a Good credit score, and put these values in a table that looks like this (you can paste the table into your solution and paste the R outputs into the appropriate cells of the table):

| CATEGORY OF PAYMENT BEHAVIOR | PERCENT GOOD CREDIT RATING |
|---|---|
| High_spent_Large_value_payments | |
| High_spent_Medium_value_payments | |
| High_spent_Small_value_payments | |

NOTE: It is not necessary to do part (i) in order to do part (ii).

(g) (*Unsupervised Approach*) From the original data frame `CreditScores`, create a new one here by deleting all of the factor predictors as well as the target column. Call your new data frame as `CreditScores _USL`. Apply the `hclust` (hierarchical clustering) algorithm to this dataframe by using complete linkage, but make sure to enter the command "set.seed(2)" right before applying this program. Plot the corresponding dendrogram. For each of the three clusters, use the target column of `CreditScores`, to determine the percent of points in the cluster that correspond to a customer with a good credit rating and paste the R outputs into the appropriate cells of the table:

| CLUSTER # | PERCENT OF PEOPLE WITH "GOOD" CREDIT RATING |
|---|---|
| 1 | |
| 2 | |
| 3 | |

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

(x) (*Extra Credit, Up to 10 points*) During the semester, we have studied several ML methods and techniques for improving their performance. Experiment with several of these methods, using any that we have learned using the training set of part (b) with the goal of minimizing the test error of your algorithm to predict `Credit_Score`. You may not use any new ML algorithms that have not been introduced in this class (e.g., neural networks). But you can do things like: scaling variables, select any subset of variables as well as creating new variables in terms of existing variables in `CreditScores` data frame. For example, you might construct a new variable like: `Outstanding_Debt` divided by `Monthly_Inhand_Salary`.
If you obtain a test error < 22%, you will get 5 points EC.
If you obtain a test error < 18%, you will get 10 points EC.
Make sure to include ALL input and relevant output code for your final version (no need to include your experimental versions, there will most probably be several).
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~