Part 2ai

The dependent variable is "class". This is apparent for a few reasons, even with the attributes being masked from their original meaning. Being the last column is one indicator this column is dependent on the others. There are only two factors, "+" or "-", providing further indication this is a dependent yes/no indicator for credit approval. Also, the new credit csv has no values in this field, indicating this is the goal of the exercise. Based on this information we can deduce A1 through A15 are independent variables. They all have a variety of different data and types which could potentially lead to an accurate prediction of class.

Part 2 a ii

While the data attributes have been masked in the Credit Approval dataset, I can infer that the ctree function within R will recursively iterate through the provided data, both training and test sets. While doing so it will begin creating probabilities of independent variables to the defined dependent variable. This will be accomplished by calculating the information gain and Gini index of attributes to determine how to partition attributes in the tree. Ultimately, we should have a constructed decision tree which will highlight the attributes who have the greatest ability to predict the dependent variable value with a specified probability percentage (Han, Kamber, & Pei, 2011).

Part 2 c i

If we consider the nature of classification algorithms, having a training and test dataset makes sense. The training test set is needed so we can teach our algorithm, within an acceptable margin, how to classify future data points. Technically we could stop there and begin working with production data. However, any good coder will tell you testing is an absolutely important step that can not be skipped. For this reason, we should take some measurable portion of our data, with known variables, to conduct test to determine if our algorithm and training dataset were successful in forming a viable prediction model.

Part 2 d iii

The first item of note from this tree is the root, A9. A9 has two factors, "f" or "t", so we know we only have two possible directions to travel. If we followed the "f" leg the next chosen attribute is A13. At this point, the decision was made to either to have either a path for "g", which would have a weight of 195, or a path of "p" and "s", which have a weight of 29. If we had followed the A9 ="t" leg, we would find A10 as the chosen path. A10 has two factors, with "f" having a weight of 103 and "t" having a weight of 150. From these results, we can deduce A9, A13, and A10 have high enough probabilities to use as independent variables in the prediction of our class variable.

Part 2 d iv

If we decide to derive an answer to the independent variables based on the ctree model then A9, A13, and A10 would be sufficient independent variable predictors. Looking at A9 = "f" and A13 = "g" the probability is within the acceptable range and the weight of values is equal to 195, the greatest amount in the entire tree. The A13 = "p" or "s" leg only has a weight of 29, however this still needs to be included for completeness of the A13 attribute. If we followed A9 = "t" we would find the A10 attribute has two factors, "f" and "t", which respectively have 103 and 150 weight counts, showing another significant path for prediction of class. While this is the conclusion we can come to, I should point out these three attributes have the least diversity in attribute variables. A9 and A10 only have two factors, and A13 has three. If we consider how data is recursively partitioned based on information gain, it becomes clear will not necessarily provide interesting results, but rather very a generalized path. If we had a better understanding of the masked attributes, we might be able to remove some of these columns or decide on a different algorithm to use.

Part 2 d e

Looking at the plot above provides more detail for us to build on from our previous assumptions on the ctree model textual data. We can still deduce our probability predictor variables: A9, A13, and A10. From this plot, we can now see the p-value of less than 0.001 was achieved for all three. Additionally, in the terminal nodes at the bottom we have a better indication of the percentage of values of the class attribute factors within each conclusion. For example, if we follow A9 = "f" and A13 = "g", we can still see a weight count of 195. Now, however, we can also conclude that .959, or nearly 96%, of these values, are "-" and 0.041, or 4%, are "+". I determined these values by examining the order of the factors with the str() function, and cross-checking by manually filtering and sorting the Credit Approval data in Excel.

Part 2 f ii

Based on the confusion matrix we can say that 207 "-" values were correctly classified and 196 "+" values were correctly classified, for a total of 403 total values correctly identified. Since there are 477 total observations in this model, we can say the classification accuracy is 403/477 = 0.84, or 84%.

Part 2 f iii

Our confusion matrix can provide several other metrics, to include the classification error and the correctly versus incorrectly predicted value ratios. In our training model, we incorrectly calculated 17 "+" values as "-" and 57 "-" values as "+". In this case our classification error is (17+57)/477 = 0.16, or 16%. The rate of true positive prediction is 196/(17+196) = 0.92, or 92%. The rate of true negative prediction is 207/(57+207) = 0.78, or 78%. As a cross-check of these values, (78%+92%)/2 = 85%. This is nearly identical to the classification accuracy calculated in the previous step, only off by a percentage due to rounding.

Part 2 g ii

Based on the confusion matrix we can say that 99 "-" values were correctly classified and 88 "+" values were correctly classified, for a total of 187 total values correctly identified. Since there are 213 total observations in this model, we can say the classification accuracy is 187/213 = 0.88, or 88%.

Part 2 g iii

For this comparison, we will look at 5 different metrics: classification accuracy, true negative, true positive, false negative, and false positive. Our training data had an 84% classification accuracy, and our test data had an 88% classification accuracy, a 4% difference, though we should also consider the test data had 213 observations, with the training data having 477, a 264-count difference. The training data correctly identified 43% true negative, while the test data came in at 46%, showing a 3% difference with a count difference of 108. The training data had a 41% accuracy for true positive, while the test data also had a 41% accuracy, with a count difference of 108. The false positive rate for the training set was 12% and for the test data 9%, a 3% difference, and a 37-count difference. The false negative rate for the training set was 3.5%, while for the test set it was 2.8% with a count difference of 11.

Part 2 h i

The two biggest differences between the Apriori and ctree methods I used on the Credit Approval dataset I have discovered are the data visualization and scope of results. For example, with the Apriori method we created a list of rules that highlighted the correlation between data points. The ctree method, on the other hand, hinges on creating an effective visual decision tree to be traversed. Additionally, the scope of the data provided is significant, but still relevant with both methods. For example, the number one rule from our week three exercise stated A4="u", A9="t", A14="[0,55.2)" => class="+". This rule had the highest lift with Apriori but was not on the ctree at all. On the other hand, rule 17 from week 3 stated A9="t" and A10="t" => class="+" with a 30% support, 91% confidence, 2.04 lift, and 207 count which is very similar to one of our ctree paths. On that same note, rule 64 from our Apriori set stated A9="f" and A13="g" => class="-" with a support of 39%, confidence of 95%, lift of 1.72, and count of 272 which also closely matches another path of our ctree.

Part 2 h ii

Missing values can have a negative effect on the ctree model. While a small percentage might not have a detrimental impact, if a large percentage is present the recursion process will eventually decide to make a surrogate split based on the original split, vice having any meaningful data to split on. This can become problematic in our classification accuracy since the split is not based on any actual measurement that would have a meaningful prediction of the dependent variable outside of the original root node above the surrogate split. This can be managed with the ctree\_control function by limiting the number of surrogate splits which can occur in a model (Hothorn, Hornik, & Zeileis, 2015).

Part 2 h iii

Oddly one of the most challenging parts of this exercise was not technically a requirement, but I felt compelled to do it for the sake of completion. After determining the predicted values for the new instance, I decided it would make sense to merge these values into the newcredit data frame. This proved to be difficult and requires an adjustment to the factors level of the class attribute in the newcredit data frame before running a loop to append the new values.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques.* Elsevier.

Hothorn, T., Hornik, K., & Zeileis, A. (2015). ctree: Conditional Inference Trees. *The Comprehensive R Archive Network*.