Problem 1: (40 points)

Download the credit approval dataset from http://archive.ics.uci.edu/ml/datasets/Credit+Approval

The dataset file concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. This dataset is interesting because there is a good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

Attribute Information:

A1: b, a.
A2: continuous.
A3: continuous.
A4: u, y, l, t.
A5: g, p, gg.
A6: c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.
A7: v, h, bb, j, n, z, dd, ff, o.
A8: continuous.
A9: t, f.
A10: t, f.
A11: continuous.
A12: t, f.
A13: g, p, s.
A14: continuous.
A15: continuous.
A16: +,- (class attribute)

i. (5 points) Report which variables have missing values and what percentages of values are missing.

The attributes that have missing values, along with the percentages of missing values, are as follows:

Variable	Number Missing	Percentage of Values that are Missing
A1	12	1.7%
A2	12	1.7%
A4	6	0.9%
A5	6	0.9%
A6	9	1.3%
A7	9	1.3%
A14	13	1.9%

ii. (5 points) Recommend an approach to handle the missing values and apply it to your data.

To handle the missing data, we propose the following: if the data is continuous, we recommend using the attribute mean for all samples of the same class. For example, the continuous variable A2 has 12 missing values, we propose replacing the missing values with 33.72 (the attribute mean for class +), if the observations are in class +, and replace the missing values with 29.81 (the attribute mean for class -), if the observations are in class -. The hope is that this will yield a better model than just replacing the missing values with the overall attribute mean. We also replace the missing values of the variable A14 in

a similar manner; we replace missing values with 164.42 (the attribute mean for class +) or with 199.70 (the attribute mean for class -), depending in which class the missing value lies.

For the categorical values, we propose replacing the missing values with the mode of the attribute; So for variable A1, the value 'b' occurs twice as likely as value 'a', thus we replace the missing values with 'b', which is the most frequent value. (Note: we first tried to find the attribute mode for samples of the same class, similar as above; however, the results coincided with the overall mode of the attribute.) Thus, for the remaining nominal variables A4, A5, A6, and A7 we replaced the missing values with the most frequent occurring values in each attribute, which were 'u', 'g', 'c', and 'v', respectively.

iii. (10 points) Using a hold out technique (66% for training and 34% for testing), create a decision tree to classify the data. Report your best decision tree along with the other configurations (i.e. varying the depth, parent, and child parameters) you chose in order to determine the best decision tree.

- Growing Method: CRT
- Validation: Split Sample
- Impurity Measure: Gini

The best decision tree (highlighted in red below), struck a good balance between Testing Accuracy (in which it actually performed the best) and complexity (it lies somewhere in the middle compared to the other models in terms of total number of nodes and number of terminal nodes). The three most important attributes in this tree were variables A9, A3 and A11.

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy
6	3	33	17	6	94.1	80.9
10	5	35	18	7	92.5	84
13	7	15	8	6	90.3	83.8
14	5	23	12	6	91.8	83.1
14	7	21	11	7	90	86.8
14	9	19	10	7	90.9	84.6
15	7	19	10	7	90.6	85.1
18	9	17	9	4	88.5	83
22	11	17	9	6	90.3	82.6
26	13	11	6	4	87.6	82.6

iv. (10 points) Now the goal is to understand how the size of the data influences the classification results. Repeat part III for the following splits: 10% for training and 90% for testing, 20% for training and 80% for testing, up to 90% for training and 10% testing. Analyze the classification results and conclude on which data split seems optimal for the given problem.

First, the trees were constructed with the same structure as above: by the CRT method and the impurity was measured by the Gini Index. Also, for the different splits, the best models (highlighted in red) were

chosen for their high testing accuracy and relatively average complexity (when compared to their neighboring models in same split).

10% Training, 90% Testing

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy
8	4	5	3	2	79.4	75.3
9	5	5	3	3	88.5	70.6
10	5	7	4	4	96	82
14	7	3	2	2	89.7	85.1
18	9	3	2	2	86.1	85.4
22	11	3	2	2	88.2	85.2
26	13	5	3	3	81.8	73.6
30	15	5	3	3	86.8	79.5
34	17	5	3	3	83.9	77.3

20% Training, 80% Testing

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy
6	3	19	10	4	95.1	82.2
10	5	7	4	3	92.5	81.1
12	5	15	8	6	93.6	81.1
12	6	9	5	4	86.7	82.5
12	7	9	5	4	87.7	86.8
12	8	11	6	5	94.4	82.1
14	7	7	4	3	92	85.1
18	9	9	5	4	90.3	77.8
22	11	7	4	3	92	83.7

30% Training, 70% Testing

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy
6	3	17	9	4	92.6	81.6
10	5	17	9	5	91.2	79.6
14	5	13	7	3	91.7	84.6
14	7	7	4	3	89.5	85.1
14	9	7	4	3	90	83.6
18	9	7	4	3	88.2	83.5

22	11	7	4	3	90	83.4

40% Training, 60% Testing

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy
10	5	11	6	5	92	84.8
14	7	15	8	5	90.1	85
17	7	13	7	5	90.7	84.4
17	9	15	8	6	90.6	86
17	11	9	5	3	88.2	83.9
22	11	15	8	4	91.6	84.9
26	13	7	4	3	88.7	83.3

50% Training, 50% Testing

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy
14	7	19	10	5	90.3	81.9
17	9	13	7	5	89.1	84.7
19	10	7	4	3	88.9	84.6
22	7	15	8	5	89.5	83.9
22	9	17	9	5	89.5	86.9
22	11	13	7	4	89.9	85.3
26	13	13	7	4	92.2	83.4

60% Training, 40% Testing

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy
18	9	15	8	5	88.8	84.1
20	10	13	7	4	87.8	83.8
22	9	11	6	5	89.6	84.8
22	11	15	8	5	89.9	86.7
22	13	13	7	5	90.6	84.9
26	13	11	6	4	88.2	83.1

70% Training, 30% Testing

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy

18	9	23	12	5	90.5	83.1
22	9	21	11	5	89.9	81.6
22	11	15	8	5	87.6	87
22	13	17	9	5	88.1	87.6
24	11	13	7	5	89.3	84.7
26	13	11	6	3	85.9	86

80% Training, 20% Testing

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy
23	12	15	8	6	90.3	81.6
23	13	17	9	5	89.8	79.1
26	13	13	7	5	88.6	85.7
30	11	17	9	5	87.7	86
30	13	15	8	5	87.8	88.5
30	15	13	7	5	87.6	86.6
34	17	11	6	4	87.4	80.1

90% Training, 10% Testing

Minimum Cases in Parent Node	Minimum Cases in Children Node	Total Number of Nodes	Number of Terminal Nodes	Depth	Training Accuracy	Testing Accuracy
18	9	19	10	5	88.2	84.1
22	11	21	11	5	88.7	84.5
25	9	13	7	5	88.4	75
25	11	13	7	5	89	79.7
25	12	15	8	5	88.9	78.8
25	13	13	7	4	88.4	85.2
26	13	17	9	5	89	81.4

The results of the different splits, for the most part, agreed that variables A9, A11, and A10 were the three most important variables. One noticeable difference between the training data sets that were smaller (50% and below), was how it was very easy to over-fit the model to the training dataset by minimizing the cases in the parent and children nodes. In fact, at these splits, the models improved by increasing the number of cases in the parent and children nodes, which is counter-intuitive to past experience.

Now comparing the different configurations against each other, it looks like the 70/30 split performed the best. I chose it as the best split because it seems to be one where you can achieve the highest testing accuracy consistently (I also got the highest testing accuracy of all models in this split). In the 80/20 and 90/10 splits, it was not uncommon to drop into the high 70s in terms of testing accuracy. The 60/40 split also performed very closely to the 70/30 split, but the best model in the 70/30 split outperformed the best model of the 60/40 split. I should mention, I was surprised that the splits with less than 60% training data

did as well as they did; they performed comparably to the splits that had higher percentages of training data.

v. (10 points) Create an executive summary (~half a page) that outlines the problem, summarizes the data, describes the methodology, summarizes the results, and makes recommendations. When creating it, imagine that you will give this summary to someone expert in the credit card industry but not in the data mining field.

Executive Summary

This report attempts to create an optimal classification model to distinguish credit card applications from applicants who will have their credit approved and those who will not be approved. The hope is that by automating the process, time will be saved and any personal bias (by the credit agent handling the application) can be lessened. The data, provided by the UCI Machine Learning Repository, contains a mix of 15 continuous and categorical variables and an additional target (class) attribute. The data contained a relatively few amount of missing values that we were able to replace using the within-class mean (for continuous variables) or the attribute mode (for the nominal variables). To create the classification model itself, we chose the decision tree method for its following properties: it is inexpensive to construct (in terms of time and computational resources needed), the if-then rules are easy to interpret, and it is exceptionally fast at classifying new observations. In our search to find the decision tree that performed the best, we created over 80 decision trees by varying the model parameters and by varying the sizes of the data used. In the end, the optimal decision tree, which performed best in terms of accuracy, could be simplified into 9 if-then rules. Although we are more than satisfied with the decision tree created and its performance, we acknowledge that there is room for improvement. In particular, one recommendation we offer is to use k-folds validation instead of split-sample validation, especially since the number of observations is not ideally high enough for split-sample validation.