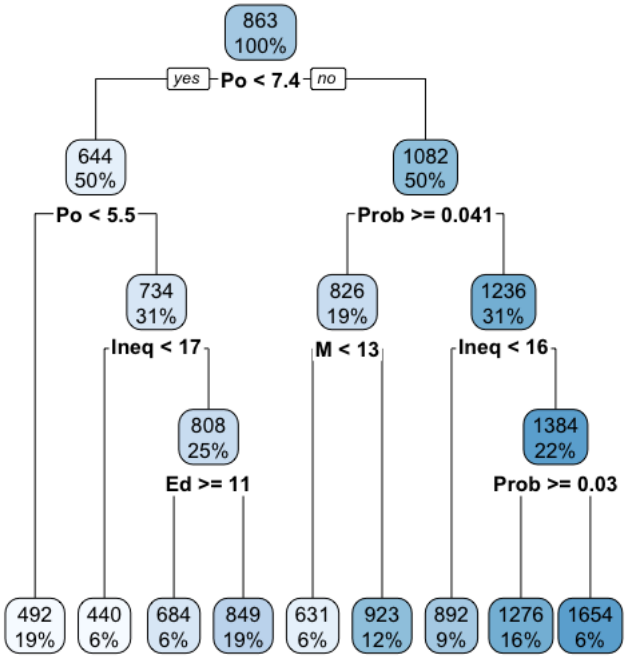
10.1 – Please review the included 10.1.R file along with this submission
First I loaded and inspected the data as we have in the last few weeks. I again decided not to remove outliers based on the week three evaluation. By plotting all of the predictors I saw that Po1 and Po2 have very similar values, which makes sense given that they are recording the same observation taken in subsequent years. Since we're working with tree-based models, I decided to combine Po1 and Po2 into a single variable Po that is the average of the two years. This way the model won't be splitting on two similar variables. Other than removing Po1 and Po2 and combining them into a single variable Po, I used the data as is.

I then split the data into training and test sets, using 70% for training and 30% for testing. The rpart package uses 10 fold cross validation by default so I didn't create a validation set.

I then built the regression tree and random forest models, trying a variety of factor combinations and minimum leaf sizes. For both the regression tree model I found the best fit with the following combination of predictors:

M + Ed + Po + Ineq + Prob

The highest R2 on the training data (.88) for this model was with a minimum leaf size of two, which is the smallest we can go while still including at least 5% of the data in each leaf. However, because we have such a small dataset this still means only 3 data points will be included on the smallest leaf, likely leading to overfitting. A plot of the regression tree is below:



When predicting crime rates on test data with this model, the R2 is much lower (.34) showing the overfitting due to a small amount of data. When predicting the crime rate in the

hypothetical test city from the previous assignment, this model predicts a crime rate of 1276, not far off from the 1304 in my linear model from a few weeks back.

The random forest model had the best results with the following factors:

M + Ed + Po + Wealth + Ineq + Prob

The R2 for this model on the training data was .84. Po was the most important factor, followed by Prob, then Wealth. Interestingly, Po and Prob were also the first and second splitting points in the regression tree model. However, Wealth did not factor in in the other model, but was important here.

On the test data the R2 of this model was .34, again much lower than the training results. On the hypothetical test city, the model predicted a crime rate of 1142.

In analyzing both models a few things stand out. It's interesting that both models have a similar R2 on the test data, suggesting that it is possibly the best we can do when predicting this data with a tree-based approach. This seems reasonable given the limited number of data points we're working with. This leads to two potential issues with a tree-based approach, causing the model to both over fit the training data and have a lack of specificity when predicting on test data. Because each leaf has only 3-6 data points, the model is overfitting it's branch points. However, because there are only 9 terminal leaves, the predicted values are the same over an overly broad range of predictor values.

10.2

Logistic Regression works best for classifying categorical data. Logistic regression would be useful for classifying whether an individual is a likely voter or not. Potential predictors are age, race, income, education level or political party affiliation.

10.3 - Please review the included 10.3.R file along with this submission

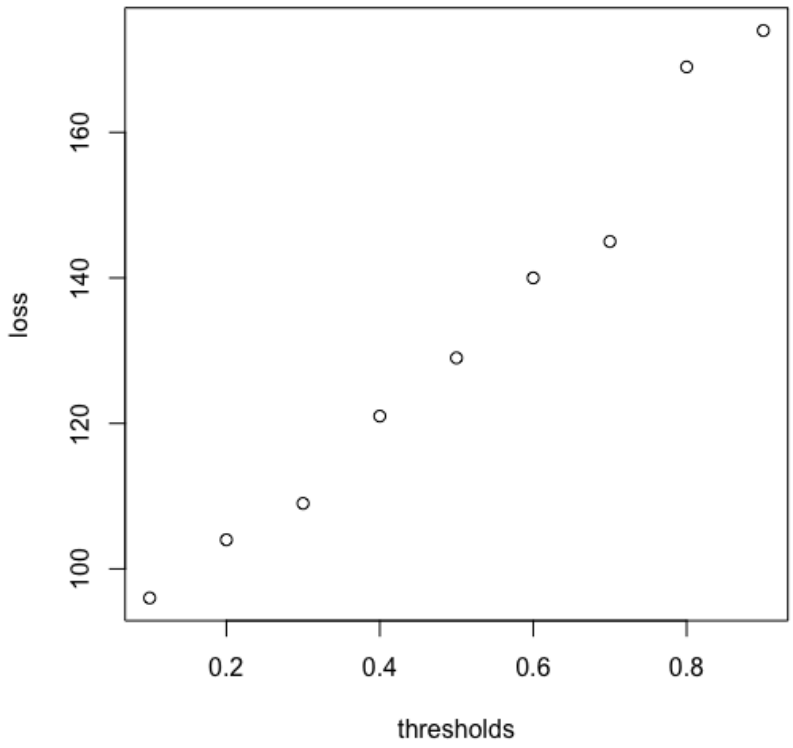
After loading the data, it appears that Column 21 contains the response for each person in the dataset, with 1 representing "Good" and 2 representing "Bad". Because we need 0 and 1 values for logistic regression, I wrote a quick function to convert each 2 in column 21 to 0. Therefore, for my analysis 0 represents a bad outcome and 1 is good.

I then split the data into training, validation, and test sets, using 70% for training, 15% for validation, and 15% for testing. I trained the model using the glm function and tried a variety of predictor combinations. The best result I could find was using a combination of V1+V2+V3+V4+V5+V6+V8+V14, which produced the lowest AIC (705.5) of all models on both the training and validation data sets. The coefficients for that model are below:

Coefficients:	
	Estimate
(Intercept)	-8.60E-01
V1A12	1.98E-01
V1A13	1.15E+00
V1A14	1.58E+00
V2	-3.15E-02
V3A31	4.00E-01
V3A32	1.08E+00
V3A33	1.22E+00
V3A34	1.71E+00
V4A41	1.43E+00
V4A410	1.44E+00
V4A42	5.51E-01
V4A43	9.43E-01
V4A44	-5.01E-02
V4A45	-6.10E-02
V4A46	-2.17E-01
V4A48	1.55E+01
V4A49	1.04E+00
V5	-1.19E-04
V6A62	4.11E-01
V6A63	5.97E-01
V6A64	1.22E+00
V6A65	9.88E-01
V8	-2.19E-01
V14A142	5.49E-01
V14A143	9.38E-01
V8	-2.19E-01
V14A142	5.49E-01
V14A143	9.38E-01

I then used the validation data set to determine the best threshold value, based on the standard that incorrectly identifying a bad customer as good is 5 times worse than incorrectly

classifying a good customer as bad. I iterated through 9 threshold values from .1 to .9. A plot of the loss function for each threshold is below:



The plot shows the loss function steadily increasing as the threshold increases. The lowest loss is at a threshold of .1. A confusion matrix for the .1 threshold on the validation data is below:

	0	1
0	17 True Negative	15 False Positive
1	21 False Negative	97 True Positive

And on the test data:

	0	1
0	13 True Negative	19 False Positive
1	35 False Negative	83 True Positive

As expected, with this threshold the model favors more false negatives over false positives as they are less costly, and is more accurate on the validation data than the test data. While the .1 threshold minimizes the loss function, it may not be the best threshold to use in production depending on what the actual business needs are. Because of the high false negative rate many good applicants are turned away. If the business needs to accept a certain number of applicants

to remain viable, it may make sense to accept a higher overall loss function to lower the number of false negatives and increase the number of true positives.