Predicting Coffee Quality

EdX Data Science Capstone Project

Evan Martin

Janurary 2021

Introduction

This report was constructed for the EdX Data Science Winter 2020-21 capstone course. The course is constructed from HarvardX materials by Professor Rafael Irizarry, including his book Introduction to Data Science which focuses on data anlysis and algorithms for the R language.

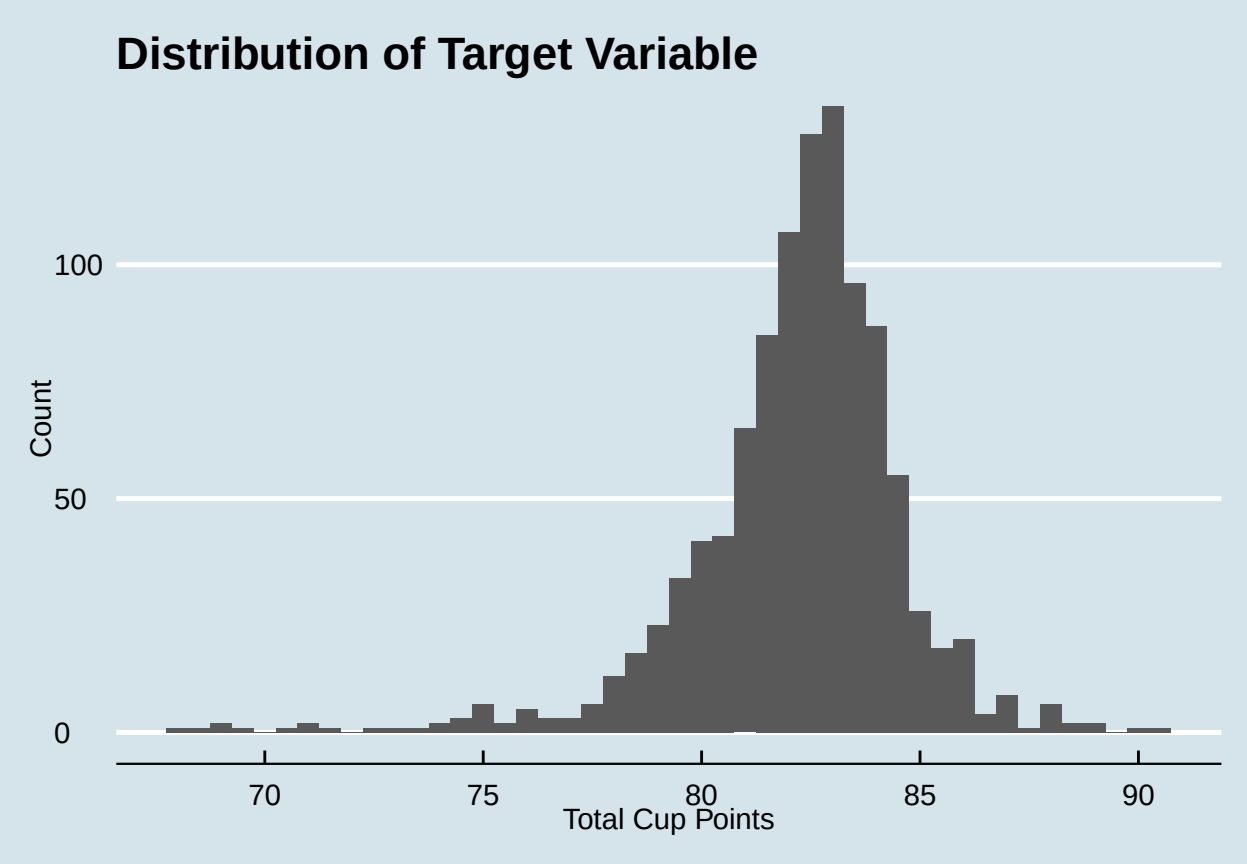
For the project, we will be analyzing a dataset published by the Coffee Quality Institute (CQI) which looks at coffee attributes and quality ratings between 2010 and 2018[1]. The CQI is an organization which offers training and certificates for coffee testers as well as a program to award a seal of quality to coffee growers. The dataset we will be looking at details attributes from samples of Arabica coffee beans and the ratings given by the testers. Ratings for a sample are based on ten criteria, each given a 0-10 score. These scores are added up to the Total Cup Points, the maximum of which can be 100. We will be examining other columns to try to find trends to see if it is possible to accurately predict the Total Cup Points.

The columns we will be looking at are Altitude, Color, Country, Harvest.Year, Moisture, Processing.Method, Variety, Category.One.Defects and Category.Two.Defects. We will examine these columns to find trends against the target variable Total.Cup.Points and build a baseline model to make predictions with a naive approach. We will then walk through several more advanced models covered in the course and compare the accuracy to the baseline. Some models will be used as touchstones to material within the course, while others are more focused on improving our predictions. We will use Root Mean Square Error (RMSE) to measure the accuracy of the predictions generated. Finally we will examine our findings and draw any conclusion.

Exploratory Analysis

The Arabica Coffee dataset[2] is from 2018, with data going back to 2010. It has been cleaned and organized, but there are still some things that need to be looked at before we begin constructing any models. 3 entries were removed for having Total.Cup.Points less than 65, which were outside the interquantile range. While it is preferable to not manipulate the incoming data, these were removed for the sake of presenting readable graphs and more accurate predictions within our models, at the cost of risking less accurate predictions if the models were used against more real-world data. The Altitude column is very messy, with ranges, different units and languages. Luckily the column altitude_mean_meters is much cleaner, allowing us to clearly see the distribution of the metric. Three entries were removed for having unrealistic measurements (higher than Mt. Everest). Lastly, NA entries have been omitted, which totalled 252 rows. If left in we would either have to omit them when creating a model and prediction, or we would have to process the data to fill them in with most likely values. Removing outliers should allow us to find more robust patterns, but it must be acknowledged that we are working within a mutated dataset, more ideal than what exists in the real world, but still similar enough to be able to draw conclusions.

Looking further at the target and predictor columns, about 85% of entries in our dataset have Total Points above 80. Our target variable is normally distributed with a long lower tail.



The Grading.Date is always one year from the Expiration Date, so the time to expiration is static across all entries, which provides no insight. Instead we want to look at predictors that have a higher degree of variability.

The country with the most entries is Mexico at 236, while other countries (Zambia, Rwanda, etc.) only have a single entry. The most samples (345) are from 2012, whereas 2018 only had 20 at the time of dataset construction, though 55 entries have no Harvest.Year information. After removing three unrealistic measurements, we can find the overal mean altitude is 1316 with a standard deviation greather than 400. It has a correlation with Total.Cup.Points of 0.10

Color, when present is either Green or Blue-Green. Entries for 'Bluish-Green' were coerced to 'Blue-Green'.

Harvest Year needed cleaning because it included extra information, some having two years (cleaned to choose the latest of the two), Quarters (1, 2, 3, 4) and months. After cleaning and standardizing, the mean Harvest Year falls between 2013 & 2014.

Moisture has complete data, mean 0.09 and SD 0.047. Where present, by far the most common Processing Method is "Washed / Wet" with the least common being "Pulped Natural / Honey" which is more of a specialty process.

For a coffee to be considered Quality by the CQI, it can contain 0 Category One Defects when examiners look at the sample of green pre-roasted beans, and up to 5 Category Two Defects. Defects that graders are looking for can include insect damage, miscoloration, frost damage and disease, to name a few. Close to 85% of all samples have 0 Category One Defects, while Categroy Two Defects are more common, where only around 25% of samples escape with 0. As expected, there is a weak negative correlation between Total.Cup.Points and Defects, -0.098 and -0.211 for Category One and Two, respectively.

There are 29 Varieties. Some, like Bourbon, make up a majority of the entries, while others, Ethiopian Yirgacheffe (one of my personal favorites) only has two entries. But in a simple linear model of Variety

against Total.Cup.Points, Yirgacheff has the strongest effect, 3.12, though with only two entries it is not a trustworthy metric.

Futhermore, with a rough visual inspection, we can see there are no dramatic trends, only some predictors with greater variance than others.

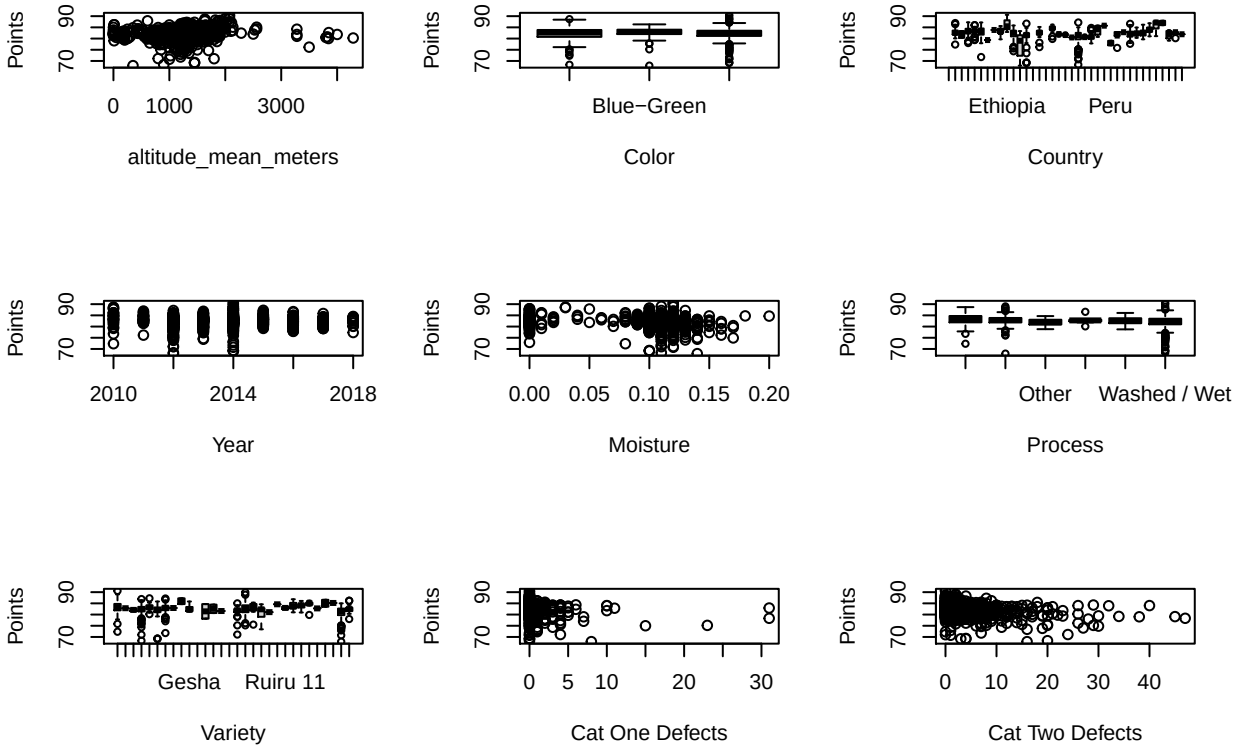


Table 1: Metrics for Numeric Predictors

	Mean	SD	Correlation with Target
Altitude	1324.51	477.75	0.16
Harvest.Year	2013.69	1.81	0.06
Moisture	0.09	0.04	-0.15
Category.One.Defects	0.38	1.89	-0.15
Category.Two.Defects	3.60	5.34	-0.29

Method & Analysis

The dataset is not overly large. We will start with splitting it into mutually exclusive train and test sets that will be used to first train a model and then test it to measure the accuracy of our predictions.

Ensure our train and test sets are relatively similar by comparing mean Total.Cup.Points.

Train_Mean	Test_Mean	Baseline_RMSE
82.23949	82.13549	2.458222

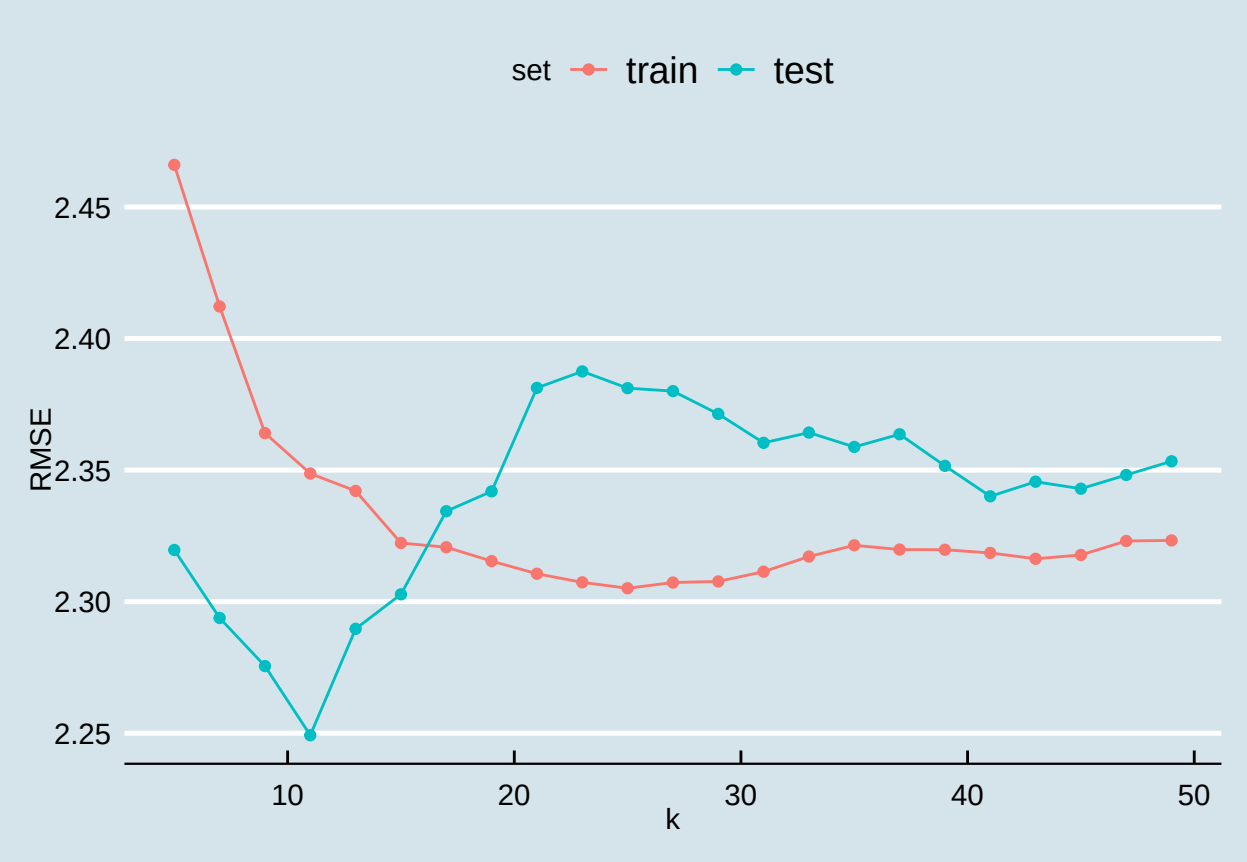
The mean Total.Cup.Points of the training set without looking at anything else is the most naive model so we will be trying to improve on that.

Linear Model The first step we can take is a straight-forward linear model on the training set with our cleaned predictors (altitude_mean_meters, Color, Country.of.Origin, Harvest.Year, Moisture, Processing.Method, Variety, Category.One.Defects and Category.Two.Defects).

method	RMSE	RSquared
Simple LM	2.235702	0.3815964

As expected, a linear model using predictors is able to outperform our baseline model though with more advanced models should be able to generate more intelligent predictions. The R-Squared indicates that our model accounts for less than half of variance which we will look at later.

KNN Just as a linear model was a simple step to improve our baseline, the next step we can take is to introduce smoothing. We can identify local trends within neighborhoods of our data, changing the size to find the fit that most reduces our errors. We will look at some numbers between 5 and 50. Since the K Nearest Neighbors model doesn't support categorical variables, it is necessary to convert categorical predictors with N entries into N individual predictors for each category. The caret train function will take on this task for us, so we will be using a large number of predictors. For this model we will use 10 fold cross validation to carve out a pieces of our training set to estimate an average true error and avoid overfitting on our training data.



After having run a tuning model, we can select the best k that was picked using only the training data to create a model and compare against the test set.

method	RMSE	Best_K
KNN	2.381158	11

While this is still an improvement over the baseline model it is not great. We should be able to do better by leveraging some of the things we noticed in our predictors during initial examination.

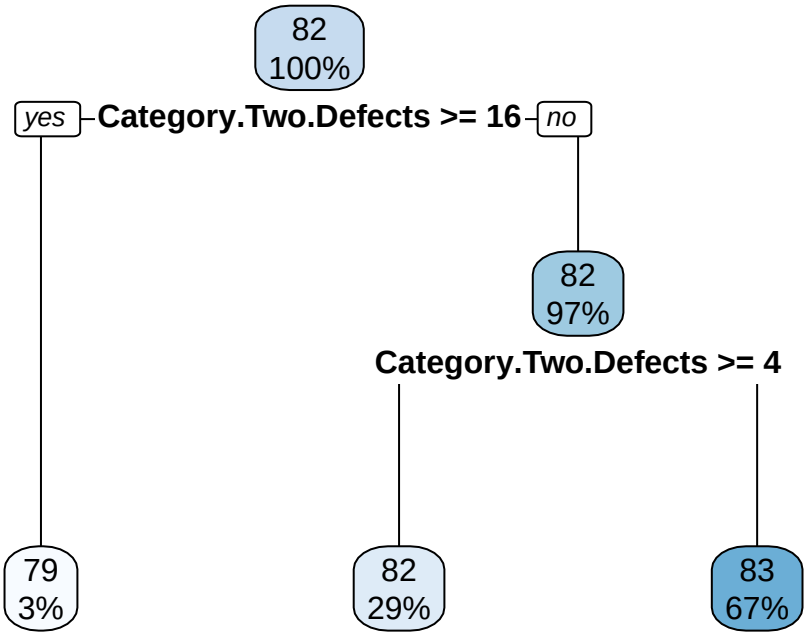
Regularized Model As noted earlier, we saw some predictors with large effects but smalls amounts of data. To navigate this, we can create a regularized model by adding a pentaly to our predictors with lambdas to minimize variance.

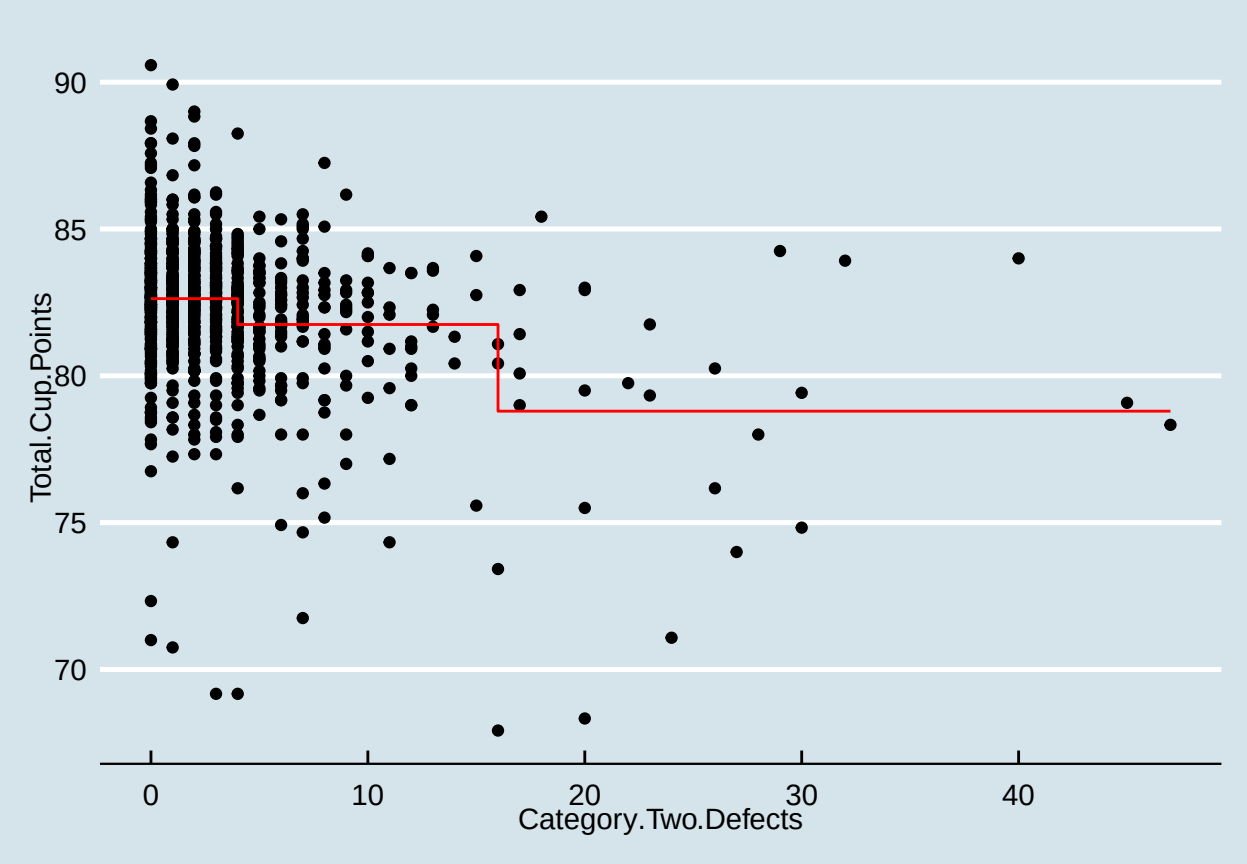
method	RMSE	Lambda
Regularized	2.0731	40

We see that by adding weights to penalize effects that come from small amounts of data we can achieve an even better model. We will use this knowledge later on once we have built up to using a random forest model.

Decision Tree Using the Recursive Partitioning and Regression Trees library, RPART, we can move into the realm of decision trees. We will use all variables when finding predictor-value pairs that minimize the Residual Sum of Squares (RSS) to form new partitions. As an example we can look at a decision tree made using only the predictor Category.Two.Defects, where we tune the complexity parameter (cp).

	method	RMSE	cp
8	Simple RPART	2.365466	0.0145833


82
100%
yes
Category.Two.Defects >= 16
no
82
97%
Category.Two.Defects >= 4
79
3%
82
29%
83
67%

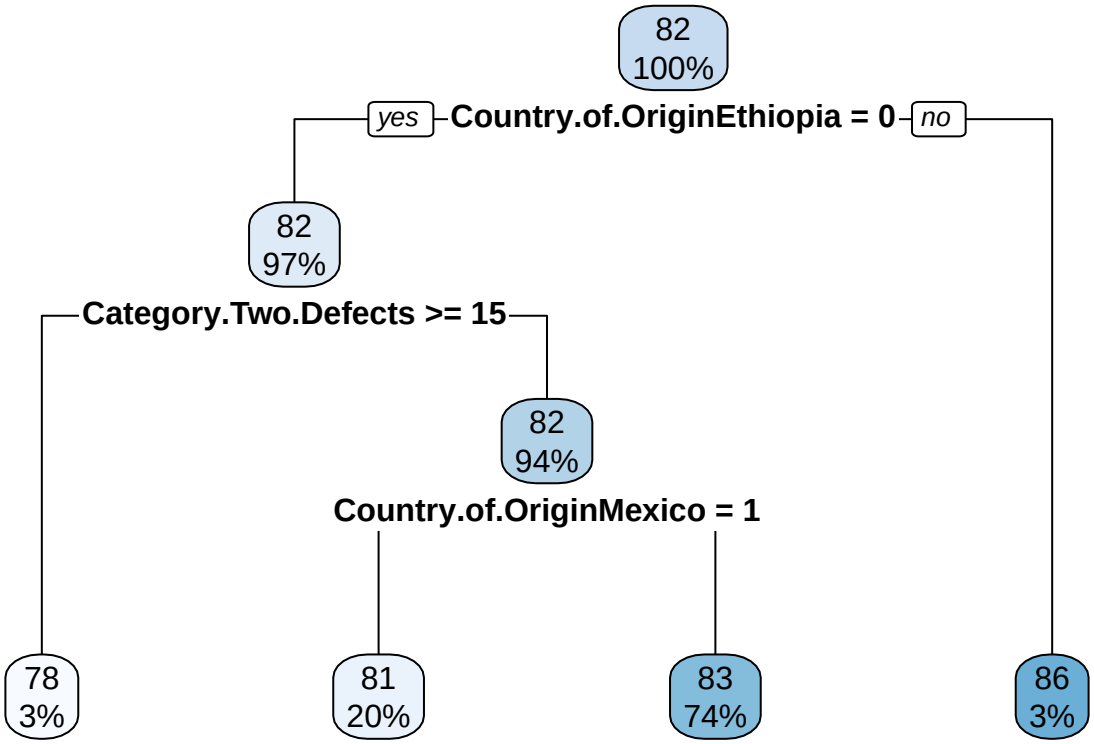


As we can see, this is a very simplistic model with only several nodes.

Now, having looked at the structure of a simple rpart model, we can expand to include all our predictors which reduces our error metric.

method	RMSE
RPART	2.27522

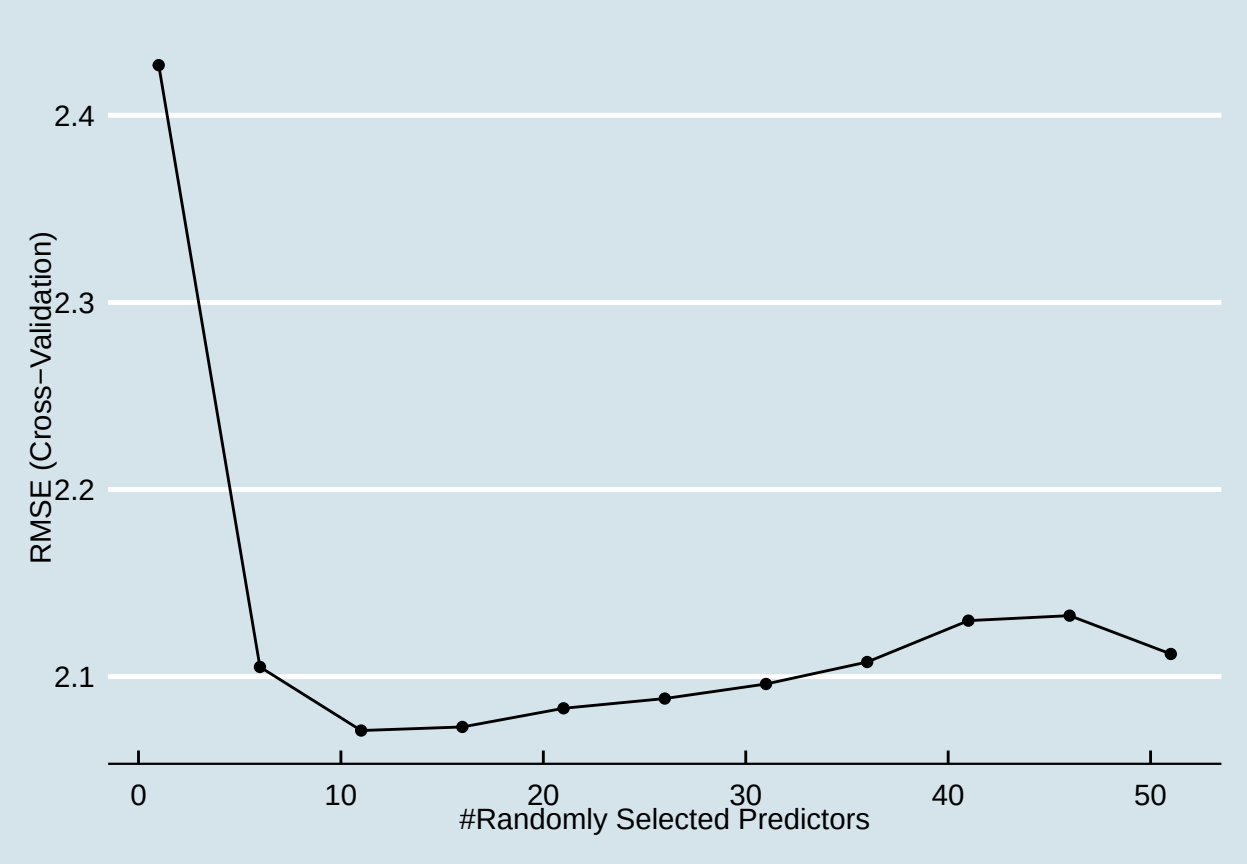
We can see the predictors which provide the greatest reductions in RSS are at and near the top root node of the inverted tree.



It should be noted that an individual decision tree can be thought of as single perspective and in order to improve our model we can take averages of many runs using a Random Forest model, the forest being an ensemble of trees, in order to smooth any extreme variances. Through this approach we hope to not only improve accuracy but also gain insight into the relative importance of our predictors.

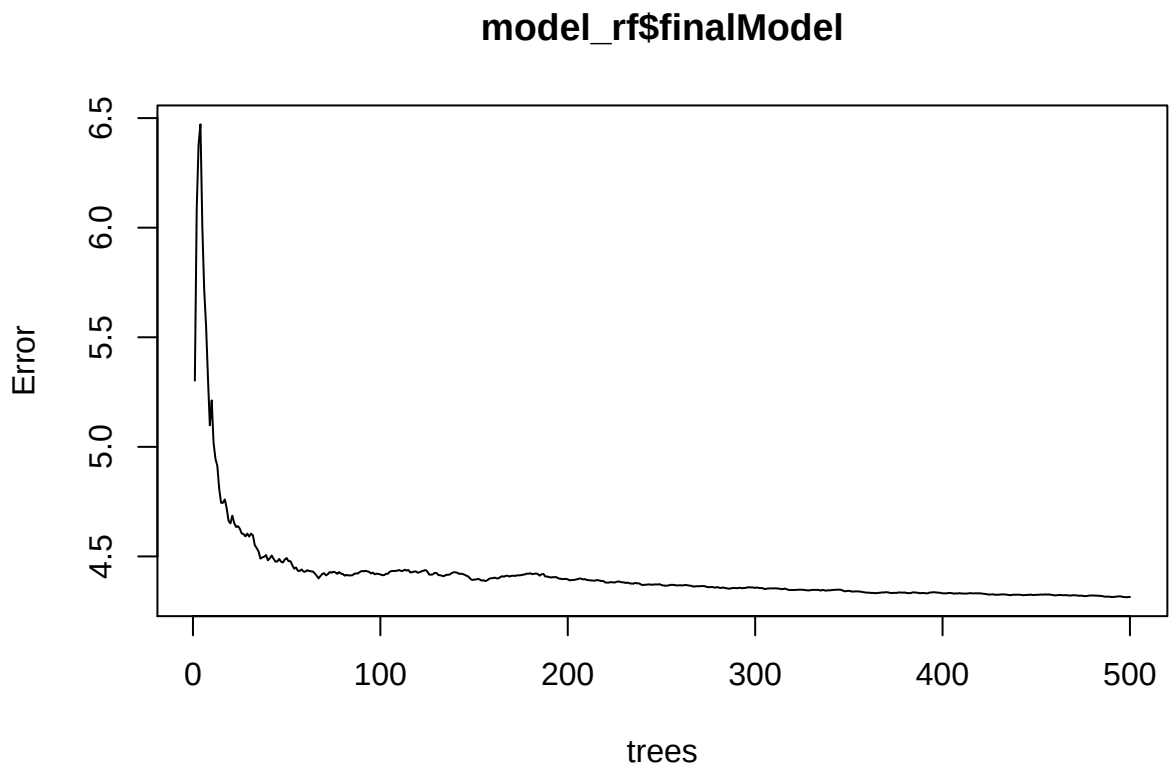
First we will find the best value for mtry, the number of variables to look at when deciding each split, or more formally, "preselected directions along which the splitting criterion is optimized"[3]. Depending on a model's signal-to-noise ratio, a random forest should be able to account for regularization by minimizing variance due to randomness introduced with the mtry parameter [4]. Given that our previous models have had low R-Squared values it is possible they have contained a high amount of noise, so we might expect a Random Forest with an opitimized mtry parameter to behave similarly to our explicitly Regularized model.

Increasing the mtry parameter increases the diversity of trees generated but also increases processing time. While our dataset has a sizeable amount of predictors, it is not so high to prohibit us using many. In order to rely only on our training set we will apply 10 fold cross validation. We will start with a small number of trees that we can increase on our final model.



Now that we have tuned our mtry, we will increase ntree from 150 to 500 and predict against our test set.

	method	RMSE	mtry
3	rf	2.033371	11



Using the tuned number of variables to be examined at each split (mtry), we can see our model stabilizes when the forest grows to around 100 trees. And as expected we have results similar to those received from the regularization model.

Results

Having walked through simple linear models to complex tree ensembles, we can compare the results and see that the explicitly penalized regularization performs nearly as well as a tuned random forest.

```
##            model     RMSE
## 1             lm 2.235702
## 2            knn 2.381158
## 3 regularization 2.073100
## 4          rpart 2.275220
## 5             rf 2.033371
```

Conclusion

The CQI dataset for Arabica coffee contained some information that needed to be cleaned and even removed before we began modeling the data, highlighting the difficulty of collecting and organizing good data, especially when it is collected from multiple countries that speak different languages. If we had left some of the bad data and NA values in we would see much higher RMSE values. While it is likely that even our cleaned data and models contain some amount of noise that masks our signal, it is also possible that our dataset is missing more powerful predictors, such as weather and soil conditions, or even information about

the individual coffee testers. It could also show that the skills of coffee testers are not so easily quantified and automated, thus showing a continued need for them as a skilled profession.

References

1. Coffee Quality Institute https://www.coffeeinstitute.org/
2. Coffee Quality Database https://github.com/jldbc/coffee-quality-database
3. ESAIM: PROCEEDINGS AND SURVEYS, 2018, Vol. 60, p. 144-162 - Jean-François Coeurjolly & Adeline Leclercq-Samson https://www.esaim-proc.org/articles/proc/pdf/2017/05/proc186008.pdf
4. Randomization as Regularization: A Degrees of Freedom Explanation for Random Forest Success - Lucas Mentch & Siyu Zhou https://deepai.org/publication/randomization-as-regularization-a-degrees-of-freedom-explanation-for-random-forest-success