HARVARD | BUSINESS | SCHOOL

9-119-021
REV: SEPTEMBER 24, 2018

SRIKANT M. DATAR
CAITLIN N. BOWLER

LendingClub (B): Decision Trees & Random Forests

After exploring the LendingClub data and working through the model building process, Figel turned to a modeling technique that was very popular among practicing data scientists: *decision trees*.

Decision Trees

Decision trees were popular in part because they were easy to build, the most transparent, and therefore easiest to interpret. As industry analyst Dean Abbot wrote, "All trees can be read as a series of 'if-then-else' rules that ultimately generate a predicted value…[which are] much more accessible to decision-makers than mathematical formulas."[1]

The decision tree emerged from a process of subdividing the data along descriptive features, such as *income* value or *credit score* value in the LendingClub context, to create increasingly homogenous (pure) subsets of data in terms of default/repay classifications. *Homogeneity* (*purity)* referred to a scenario where one subset contained only points that represented repaid loans and the other subset contained only points that represented defaults. There were different measures of impurity. *Appendix A* discusses the Gini impurity measure, which Figel used in her models.

Several technical characteristics of this process contributed to its popularity. The decision tree algorithm automatically selected the variable by which to segment data points, based on how much *impurity* that segmentation eliminated in the resulting groups. Once a decision tree was fully *grown*, one could extract a set of rules by which to classify a new point. Also, the technique was extremely flexible. It did not assume a particular functional form (e.g. linear, logistic, etc.). Rather, it relied on the data to segment different loans into defaulters and repayers based on which predictor values performed best in doing the "sorting." Moreover, once a cut was made, the decision tree algorithm used different predictive features and values to make cuts for the resulting subsets. To better understand the decision trees technique, Figel worked through a simple example. See *Appendix A*.

Building a predictive model

Figel used the same data set she had prepared to train the logistic regression models. The set contained approximately 413,000 records, of which approximately 75,000 were borrowers who had defaulted and approximately 338,000 were borrowers who had repaid. About 100 descriptive features

Professor Srikant M. Datar and Research Associate Caitlin N. Bowler prepared the original verson of this case, "Lending Club: Predicting Default," HBS No. 518-010, which is being replaced by this version prepared by the same authors with the assistance of Rashmi Banthia. This case was developed from published sources. Funding for the development of this case was provided by Harvard Business School and not by the company. Emily Figel is a fictional character. HBS cases are developed solely as the basis for class discussion. Cases are not intended to serve as endorsements, sources of primary data, or illustrations of effective or ineffective management.

described each record and the *target* feature was, again, *loan status* with two classes: *default* and *repay*. As before, Figel cleaned the data, which left her with nearly 70 descriptive features. She then divided it into training, cross-validation, and holdout folds. **Exhibit 1** summarizes these groupings of data.

For fully-grown decision trees, with pure terminal nodes, the model either returned a prediction of default ($y = 1$) with probability 0 or 1, or a prediction of repay ($y = 0$) with probability 0 or 1. When $y = 1$ and $p = 1$ or $y = 0$ and $p = 0$ the log loss was zero, that is, the tree correctly classified the record. When $y = 1$ and $p = 0$ or when $y = 0$ and $p = 1$, the log loss would be a large positive number, indicating a misclassification. The decision tree produced a log loss measure of 0.4393. The log loss indicated the decision tree model was doing a reasonably good job identifying defaulters from repayers.

Feature Impact

The *feature impact* measured how important a particular feature was in predicting the target feature, according to that model. In the case of decision trees, feature impact recognized how often a feature was chosen as the splitting variable, weighted by the relative improvement in the measure (e.g. Gini impurity) as a result of the split.

The decision tree seemed to be driven heavily by the features *sub-grade* and *debt-to-income*.[a] Remarkably, in a dataset with nearly 70 features, the decision tree model drew on only a few features to make classifications, and some of those registered at minimal levels (3% or below). See **Table 1** below.

Table 1 Feature Impact (percent)

Feature	Decision Tree
Sub-grade	100
Debt-to-income	58
Loan amount	15
Grade	9
Mortgage account	5
Employment length	4
FICO range high	3
Months since oldest revolving account opened	2.5
FICO range low	2

Source: Casewriter.

Figel considered what to make of the feature impact table. She was not surprised that sub-grade was the most important feature that the decision trees used to distinguish repayers from defaulters. After all, LendingClub had done some analysis when classifying loans into different sub-grades. She was a little confused about why *other* features such as debt-to-income and loan amount were so impactful. Were these, and other, variables picking up information about repayers and defaulters beyond what sub-grade was indicating?

[a] LendingClub assigned each loan a *grade* (A-G) and *sub-grade* (1-5).

Thinking about Overfitting

The LendingClub data was a record of prior events. The decision tree model learned from the way events occurred in the past, as encoded in the data, to recognize patterns that would be generalizable to new data. This was the beauty of predictive modeling.[2]

However, this was also the source of some danger. Because the model derived directly from the data, and was very flexible in how it fit the data, it was susceptible to *induction bias*. In other words, the model would detect and formalize whatever assumptions were latent in that *particular* sample of data. The danger was that those assumptions might not be representative of a larger population. This phenomenon, where the model was too closely tuned to a particular sample, was known as *overfitting*. The key to mitigating the problem was to adjust the model after building it so that it was indeed generalizable.

Pruning the Tree

Figel "pruned" the decision tree by cutting back layers to reduce the overall depth. Her concern was that a very deep tree was likely to be overfitting the training data. Intuitively, the deeper branches and splits were simply capturing noise in the training set rather than reflecting patterns that would generally occur in the data, such as in a holdout set. To see why pruning might be helpful, Figel returned to her simple example in *Appendix A*. See *Appendix B* for a discussion of pruned trees.

A decision tree model created from Figel's approximately 70 feature data set could reach a *tree depth* of over 70 if fully built out. As Siegel wrote, "Pruning is guided not by the training data that determined the tree's growth, but by the validation data that now reveals where that growth went awry."[3] Figel chose to prune her trained model to several depths and then compare the log loss value for each tree to determine the optimal depth to which to prune. She wondered if this was a good approach to balancing the tradeoff of model accuracy against overfitting on unrepresentative training data.

Tree depth was a *hyperparameter* in the decision tree model. In the model-building process, a *parameter* emerged directly from the data. Parameters included the specific feature and value to use at each node in the decision tree or, as in the case of regression, the values of the βs. In contrast, hyperparameters could not be learned. To optimize the hyperparameter, Figel would have to train her model several times using different tree depths. She would then use cross-validation to determine which hyperparameter worked best.

Figel began by pruning the decision tree to depth 3. This tree had better predictive accuracy than the full tree. She wondered if she was throwing away information and if this was a good way to choose a decision-tree model. Because the tree was not being fully built out, pruning meant that the decision tree would have several impure nodes containing both borrowers and repayers. For example, a node of 20 borrowers might have 12 repayers and 8 defaulters. At this node, Figel would only be able to conclude that borrowers with these characteristics would have a 60% chance of repaying (12/20). She wondered what this meant. Moreover, she wondered whether she should be concerned about expressing these as probabilities when there were so few data points at this node (12 borrowers who repaid and 8 borrowers who defaulted). She thought she would have felt differently if she had 500 borrowers at this node of whom 300 repaid and 200 defaulted.

Figel then pruned the decision tree to depths of 5, 10, and 20. She then calculated the log loss on the different validation samples, each time using different folds for training and validation. She averaged the log loss over the five cross-validation samples. **Table 2** shows the log loss for these different models.

Table 2 Comparative logloss values, decision tree (fully grown) and pruned trees

Decision Tree	DT Depth 3	DT Depth 5	DT Depth 10	DT Depth 20
0.4393	0.4350	0.4323	0.4343	0.4387

Source: Casewriter.

The decision tree pruned to depth 5 performed better than the rest of Figel's models in terms of the log loss measure. That made sense to her, given what she knew about overfitting. She was intrigued by the trajectory of the log loss functions across her decision trees. These decreased from the fully grown tree (log loss .4393) until the tree at depth 5 (log loss .4323). However, for the tree at depth 10 log loss increased again to .4343 and even more at depth 20 to .4387.

Pruning had reduced the log loss from 0.4393 to 0.4323 but not significantly. She looked at the feature impact table for the decision tree pruned to depth 5. It was very similar to feature impacts for the decision tree model—sub-grade was still the most dominant feature and the other features, such as debt-to-income ratio and loan amount were still showing as important features to distinguish defaulters from repayers.

Ensemble Models

Figel had successfully trained a simple decision tree model and compared the log loss values for trees pruned to various depths. Pruning to depth 5 worked best, but she was aware that there were other ways to build predictive models that were even more powerful. As she built decision trees, her concern was whether a particular feature might be overly influencing the results in all the models she trained. She looked into another modeling technique that combined the output of many decision trees into an *ensemble*. In ensemble modeling a data scientist built multiple individual models, each with its own probability predictions, and combined them into one aggregate model. This "ensemble" often made more accurate probabilistic predictions on new data than a single model alone.

Authors Provost and Fawcett used the analogy of experts to explain the concept. Imagine each model as an "expert" on a target prediction task: Will this borrower default? A collection of models was simply a group of experts each with a slightly different perspective on the task. If each expert had the exact same knowledge of the task, then each expert would predict the same outcome and additional models would add no value. But because each expert had a *slightly* different perspective on the probability of a borrower defaulting, then collectively they could provide more information than any single expert on her own.[4]

Random Forest

Figel looked into one form of ensemble modelling that used a collection of decision trees to build a more effective model: *random forest*. The foundational assumption of the random forest approach was that a collection of decision trees with variations in the variables used to make the next partition would produce better predictions than a single decision tree such as Figel had developed earlier. There were two key elements to building a random forest model: *bootstrapping* and *random variable suppression.*

Bootstrapping

Bootstrapping was a method for creating variation within sub-samples when all observations came from the same sample. Figel referred back to the first cross-validation sample comprising folds 1 through 4 and comprising 216,000 repayers and 48,000 defaulters.

She began with the pool of *defaulters*. She picked one defaulter *at random* from that pool—say, #3,000—recorded that number i.d. and then *replaced* #3,000 into the pool. She then picked a second defaulter at random—#17,000—and again replaced it. She picked a third defaulter at random—it happened to be #3,000, again—and then replaced it into the pool. She repeated this process until she had a sample of 48,000 defaulters, some of whom were selected multiple times as she selected, noted, and replaced each defaulter. Several defaulters from the 48,000 defaulters were not included in this sample.

She repeated this process for the repayers. She now had bootstrapped sample #1. She would use this sample to build one decision tree. She would bootstrap a *new* sample using this same process for every decision tree she trained.

Random feature selection

Random feature selection was the second key element to the random forest approach. She imagined a two-step process at each node of the decision tree. The algorithm (a) randomly selected features on which to make the cut, and (b) made the cut on one of those features to minimize Gini impurity.

Figel wondered how many random features should be chosen at each node. She was aware of research that revealed that a good rule-of-thumb for the number of features to include in an individual decision tree node when doing random forest was the square root of the number of features in the model ($\sqrt{p}$). Figel had nearly 70 features to choose from. Her random forest algorithm would randomly select 8 from the 70 features ($\sqrt{70} \approx 8$). For example, for node 1 in her first decision tree the algorithm might look at the 8 features listed in **Table 3**.

Table 3 Randomly selected features for node 1

8 Features ($\sqrt{70} \approx 8$)			
Purpose	Total credit lines	Employment length	Avg. current balance
Fico range-high	Credit inquiries last 12 months	Revolving balance	Verification status

Source: Casewriter

From among these 8 features the algorithm would select the feature and value for the cut that would minimize the Gini impurity at the first cut. At the very next node the algorithm would be cutting from among 8 different, randomly selected features, again, selecting to cut on the feature that minimized Gini impurity at that point in the tree. It would continue this process of selecting a feature from among a fresh set of randomly selected features and splitting at a value that minimized Gini impurity until the tree was fully-grown.

Other than the fact that the decision tree was created using a bootstrapped sample and random feature selection, the tree that Figel built was *exactly* like any other decision tree. Because trees were grown sufficiently deep, they specified a set of rules that classified any loan as a defaulter or repayer.

Figel had now developed multiple trees, each with its own decision rule. Each observation in the validation sample was classified as a repayer or defaulter by each bootstrapped decision tree. The loan was assigned a class, default or repay, based on how the majority of trees voted.

Figel understood that, counterintuitively, there could be value to randomly suppressing different features when determining the value and feature that minimized Gini impurity at each node. This would be particularly important if there were one or two variables that were overly influencing how the decision tree formed for a specific sample. At the same time, randomly selecting features on which to build out a tree hardly seemed like the best way to minimize Gini impurity, based on the decision tree models she had built previously. How could this possibly result in better models than the very flexible (pruned) decision trees she had built?

She was also uncertain about how many features to use at each node. What if there were a large number of irrelevant features? Selecting too few would result in trees being built that missed important features. What if important features were highly correlated? Then, random selection of features might still result in including only those few features that had the greatest influence in this particular dataset.

By repeating the process of bootstrapping and random feature selection Figel built hundreds of trees. An important question was how many trees should Figel build in her random forest model? At the same time, she puzzled over why she should create so many trees. After all, the same data was being used over and over again to create the bootstrapped samples.

Hyperparameter

In random forest, the choice of the number of bootstrapped sample trees to build was a *hyperparameter*. Figel had to pick that value before training the model. If she wanted, she could *optimize* this hyperparameter by training her model several times using different number of bootstrapped samples (150, 200, 250, 300). At each node, she needed to choose another hyperparameter: the number of features she should randomly selected (10%, 20%, 30%, or 40% of all the features). She could then use cross validation to determine which hyperparameter worked best.

She tried different hyperparameters. The random forest model predicted default versus repay best with 250 bootstrapped samples and by choosing from among 40% of randomly selected features at each node. She contemplated why she needed to choose from among so many features at each node. She had thought that the random forest model deliberately limited the number of features to choose from at each node so that a few features would not dominate the development of the tree.

Comparing the models

Table 4 shows the log loss for the random forest model compared to the log losses calculated for the other decision tree models. The random forest had the lowest log loss.

Table 4 Comparative logloss values, decision tree (fully grown), pruned trees, and random forest

Decision Tree	DT Depth 3	DT Depth 5	DT Depth 10	DT Depth 20	Random Forest
0.4393	0.4350	0.4323	0.4343	0.4387	0.4267

Source: Casewriter.

Table 5 shows the features with the greatest impacts in the random forest model compared to the decision tree model. Figel observed that while the features *sub-grade* and *debt-to-income* were still very important to the random forest model's ability to discriminate between default and repayer loans, many more features appeared to have an impact in the random forest model compared to the decision tree and regression models. It was these features that contributed to the better performance of the random forest model.

Table 5 Feature Impact (percent)

Feature	Decision Tree	Random Forest
Sub-grade	100	100
Debt-to-income	58	63
Loan amount	15	54
Interest rate		49
Grade	9	48
Months since oldest revolving account opened	2.5	24
Total credit balance excluding mortgage		22
Zipcode		21
Revolving balance	1.5	21
Mortgage account	5	17
Employment length	4	15
FICO range low	2	16
FICO range high	3	15

Source: Casewriter.

See **Exhibit 2** for a visual comparison of the feature impact measures for these models.

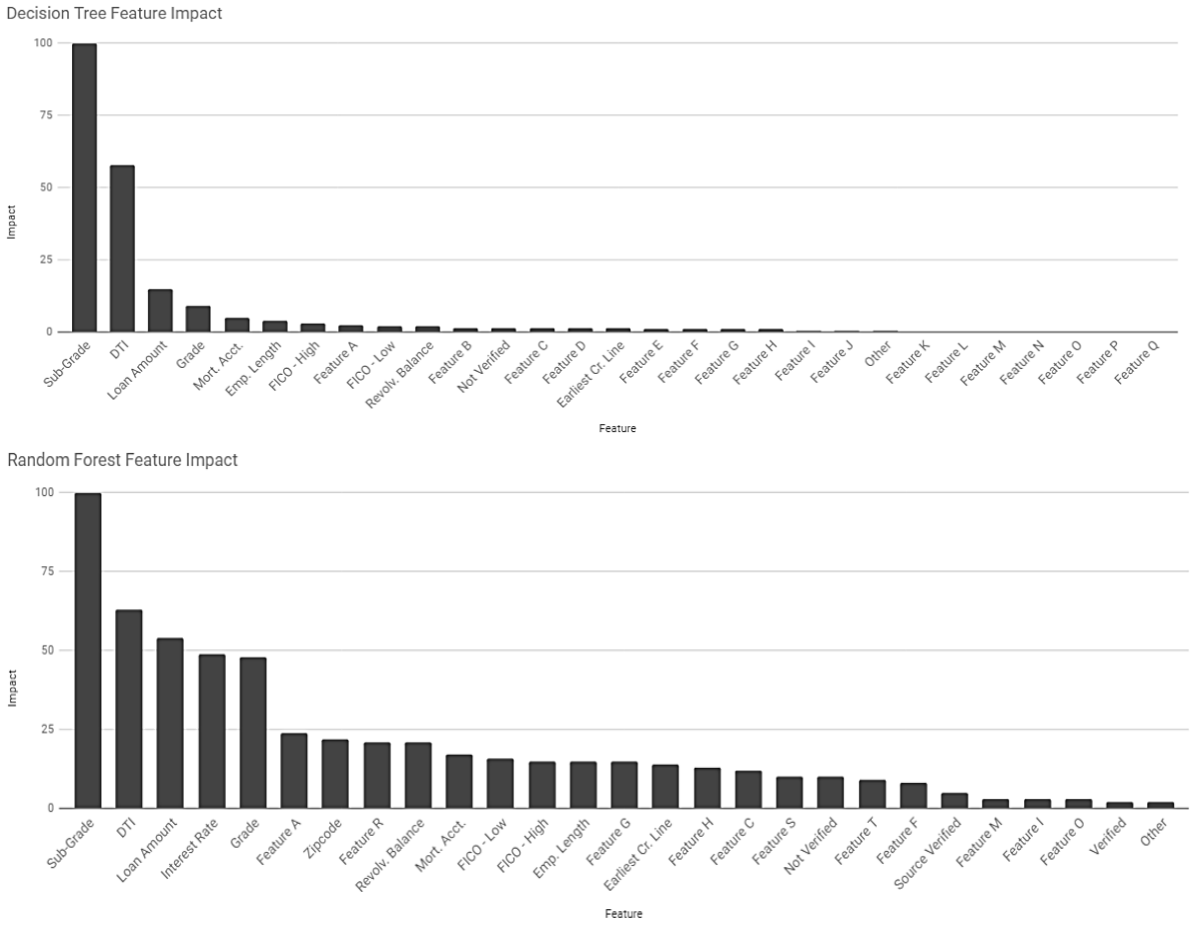
Figel began to think about whether these features made sense, given everything she knew about consumer lending and factors that contributed to borrowers' defaults.

Exhibit 1 Summary of data sub-division—Training, validation, holdout

Sub-set	Total	Loan is Bad Default (1)	Loan is Good Repay (0)
Total	413,000	75,000	338,000
Holdout Set (20%)	83,000	15,000	68,000
Training Set (80%)	330,000	60,000	270,000
Four folds used for estimating the model	264,000	48,000	216,000
Cross-validation fold	66,000	12,000	54,000

Source: Casewriter.

Exhibit 2 Comparison of feature impact measures for decision tree (top) and random forest (bottom) models



Descriptions of Abbreviated Feature Names

A	Months since oldest revolving account opened	K	Purpose: Major purchase
B	Months since oldest bank installment account opened	L	Employment: Service industry
C	Number of accounts opened in past 12 months	M	Purpose: Debt consolidation
D	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.	N	Employment: Business owner
E	Percentage of all bankcard accounts > 75% of limit.	O	Number of accounts ever 120 or more days past due
F	Number of satisfactory bankcard accounts	P	Employment: Transportation
G	Months since oldest revolving account opened	Q	Employment: Engineering/Technology
H	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER	R	Total credit balance excluding mortgage
I	Total collection amounts ever owed	S	Number of bankcard accounts
J	Total credit balance excluding mortgage	T	Number of currently active bankcard accounts

Source: Casewriter.

Note: In the random forest model only 5 features (of 55) have a feature impact *below* 5%, while in the decision tree model only 4 features (of 55) have a feature impact *above* 5%.

Appendix A: Training a Decision Tree

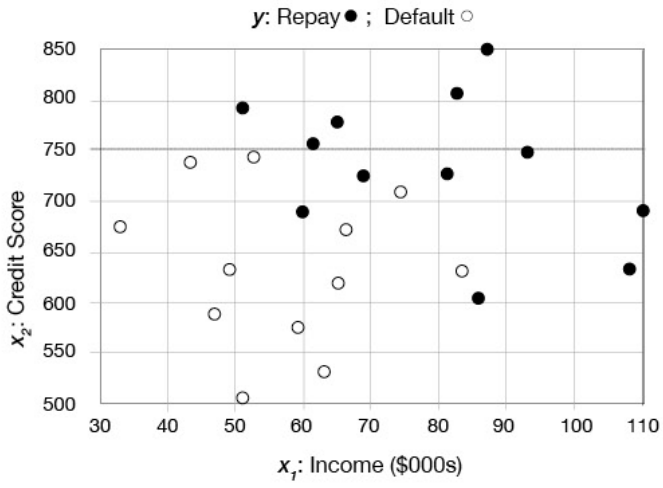
To understand the mechanics of the decision tree algorithm Figel assembled a random sample of 24 borrowers: 12 borrowers who had re-paid their loans and 12 borrowers who had defaulted. This was her training data. For each sample borrower she chose two *descriptive* features: income (x_1) and credit score (x_2). The *target* feature (y) was *loan status*, with two class values, default and repay. **Table A-1** shows the feature values for the 24 records. **Figure A-1** shows the plot of this data.

Table A- 1 Dataset feature values

Income	Credit Score	Default (1)	Income	Credit Score	Repay (0)
$51,000	524	1	$85,000	620	0
$63,000	548	1	$108,000	648	0
$59,000	596	1	$60,000	682	0
$47,000	602	1	$110,000	701	0
$64,000	627	1	$69,000	731	0
$84,000	640	1	$81,000	716	0
$49,000	638	1	$93,000	749	0
$66,000	667	1	$61,000	752	0
$33,000	680	1	$65,000	767	0
$75,000	708	1	$52,000	788	0
$43,000	730	1	$83,000	796	0
$53,000	748	1	$87,000	840	0

Source: Casewriter.

Figure A-1 Original scatter plot



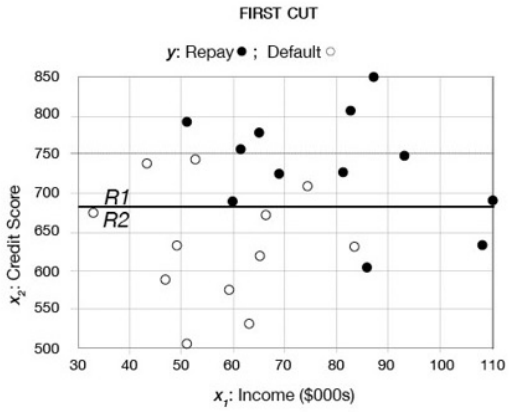
Source: Casewriter.

To place the first *split* that would segment the data, the algorithm tested all possible values for vertical and horizontal cuts. It then selected the feature that, when split on a particular value, resulted in two rectangles (R1, R2) that were as homogenous as possible. **Figure A-2** shows the split in the partitioned scatter plot, the distribution of repay/default across R1/R2 in the observation tally, and the translation from partition to a decision tree. Homogeneity means that R1 contains more repayers and R2 contains more defaulters.

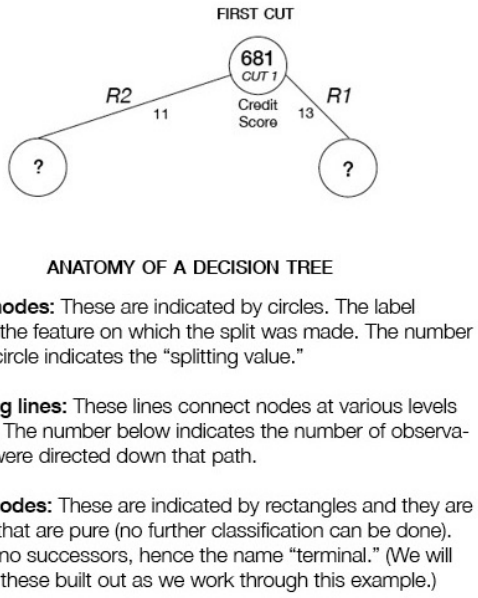
In this case, the first split occurred at a credit score of 681. These first two rectangles (R1, R2) were more homogenous than the rectangles created by a split at any point on *either* feature would have been. In this way the classification algorithm determined the feature with the most explanatory power; this feature and split value became the first node in the decision tree.

Compared to 12 defaulters and 12 repayers in the full rectangle, rectangle R1, which represents borrowers with a credit score greater than 681, has 10 repayers and 3 defaulters. Rectangle R2, which represents borrowers with a credit score less than 681, has 2 repayers and 9 defaulters. Rectangle R1 comprises mostly repayers and rectangle R2 comprises mostly defaulters. But both rectangles are still impure—rectangle R1 contains a few defaulters and rectangle R2 contains a few repayers.

Figure A-2 Partition, first cut



	R1	R2
Repay (black)	10	2
Default (white)	3	9



Source: Casewriter.

The algorithm had evaluated all possible options for splitting the observations into R1 and R2 using both credit score (horizontal cut) and income levels (vertical cut) and stopped at the point on the descriptive feature (credit score) that maximized homogeneity (purity) in the resulting rectangles. This process of *recursive partitioning* was the foundational mechanism for building a decision tree from the data. *Recursion* simply meant to apply a rule, definition, or procedure again and again to successive results.[5]

How to Determine Whether a Split Maximizes Homogeneity

To determine the particular split that maximized homogeneity the algorithm used a purity measure, *Gini impurity*.[b] If a set was very mixed it was "impure" and the Gini impurity would be high. As the proportion of re-payers to defaulters increased and the rectangle became more pure, the Gini impurity would decrease. The difference in Gini impurity from, for example, the original data set to the contents of R1 and R2, was *information gain*. Information gain and Gini impurity were inversely related.

The Gini impurity calculation was based on *expected error*. If Figel knew the class probabilities of defaulters and repayers from her sample, and she chose an applicant at random, she could calculate the probability of misclassifying the applicant. For any point on either the credit score or income axis she could calculate the *Gini impurity* (or misclassification error):

$$probability\ of\ picking\ a\ repayer\ (black\ dot)\ \times (1 - probability\ of\ picking\ a\ repayer)$$

$$+\ probability\ of\ picking\ defaulter\ (white) \times (1 - probability\ of\ picking\ defaulter)$$

The process for calculating Gini impurity was as follows.

1. **Establish the baseline Gini impurity.** This calculation was based on the number of classes in the starting population and would be at its highest at this moment. In this example, Gini impurity would equal *P*(*repaid*) + *P*(*default*). (However, imagine a third category of borrower, one who was *late* on payments, but had not yet defaulted. In that case the Gini impurity would equal *P*(*repaid*) + *P*(*late*) + *P*(*default*).) In the current example in ***Figure A-1***, the 12 defaulters and 12 repayers are all in the same pool, so the calculation was:

$$\frac{12}{24} \times \left(1 - \frac{12}{24}\right) + \frac{12}{24} \times \left(1 - \frac{12}{24}\right) = \left(\frac{1}{2} \times \frac{1}{2}\right) + \left(\frac{1}{2} \times \frac{1}{2}\right) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

   The Gini impurity was ½, which in fact was the maximum value for the measure, given only two classes.

2. **Compare new Gini impurities for any possible cut.** With the baseline established, a software program could then test each possible placement of a cut (on either the credit score or income axis), calculate the new Gini impurity values for each possible partition, R1 and R2, calculate the difference between baseline and new values, and then place the cut to optimize reduction of the Gini impurity.

---

[b] *Entropy* was another purity measure. If a set was very mixed, for instance it had ½ re-payers and ½ defaulters, then it was most "impure" and the entropy measure was at its highest, 1. If a group has only one type, either re-payers or defaulters, it is pure and entropy = 0. Like Gini impurity, entropy was inversely related to information gain.

"Shannon" Entropy is named after one of the first information theorists. Claude Shannon entropy is defined as $A = \sum_{k=1}^{m} - p_k log_2(p_k)$ where p_k is the different class proportions of response variable $k = 1 \cdots m$. This measure ranges between 0 (most pure, all observations belong to the same class) and $log2\ (m)$ when all m classes are represented equally. (Note: There are other entropy measures depending on the log function used. Shannon entropy used 2.)

Note that in two dimensions a line separates the data points. In three dimensions the algorithm will calculate a plane to separate points. In *n* dimensions, the algorithm will calculate an (*n* - 1) dimensional separating plane.

At the credit score equal to 681 cut, the Gini impurity for R1 and R2 are:

$$R1: \frac{10}{13} \times \frac{3}{13} + \frac{3}{13} \times \frac{10}{13} = \frac{60}{169} = 0.356$$

$$R2: \frac{2}{11} \times \frac{9}{11} + \frac{9}{11} \times \frac{2}{11} = \frac{36}{121} = 0.298$$

Gini impurity for R1/R2 =

$$proportion\ of\ observations\ in\ R1 \times Gini\ impurity\ for\ R1 + proportion\ of\ observations\ in\ R2 \times Gini\ impurity\ for\ R2 =$$

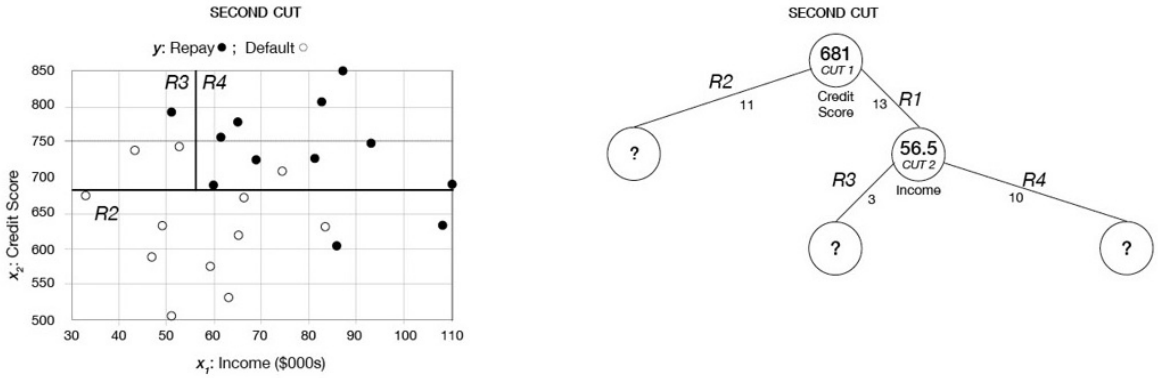
$$\frac{13}{24} \times 0.356 + \frac{11}{24} \times 0.298 = 0.3286$$

The Gini impurity was reduced from 0.5 to 0.3286 as a result of this segmentation.

Continuing to Build the Decision Tree

The decision tree algorithm made the second split on the feature of income at a value of $56,500, which put 3 observations to the left of the split; this group had incomes < $56,500. (See **Figure A-3.**) The 10 observations that split to the right had incomes >$56,500. Again, neither of these new rectangles (R3, R4) were fully pure. R3 consisted of 3 observations (1 repayer and 2 defaulters), while R4 contained 10 observations (9 repayers and 1 defaulter).

Figure A-3 Partitioning, second cut



	R1	R2	R3	R4
Repay (black)	n/a	2	1	9
Default (white)	n/a	9	2	1

Source: Casewriter.

The Gini impurity for the full decision tree at this stage is 0.2584.

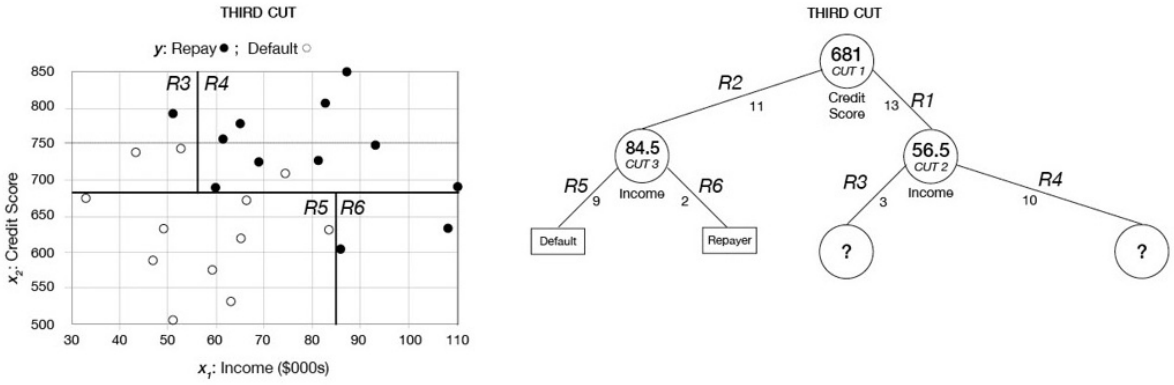
On its third iteration the decision tree algorithm made the partition on the feature of income at a value of $84,500. (See **Figure A-4**.) The observations that split to the left (R5) had incomes < $84,500, while those that split to the right (R6) had incomes > $84,500. Because both new rectangles were composed of observations from a single class (R5 all defaulters, white dots; R6 all repayers, black dots) neither could be split further. Figel did the formal calculations for Gini impurity for R5 and R6 anyway.

$$R5 = P(B) + P(W) = \frac{0}{9} \times \left(1 - \frac{0}{9}\right) + \frac{9}{9} \times \left(1 - \frac{9}{9}\right) = (0 \times 1) + (1 \times 0) = 0$$

$$R6 = P(B) + P(W) = \frac{2}{2} \times \left(1 - \frac{2}{2}\right) + \frac{0}{2} \times \left(1 - \frac{2}{2}\right) = (1 \times 0) + (0 \times 1) = 0$$

At complete purity, R5 and R6 were classified as *terminal nodes*. This *terminal* status was indicated on the decision tree itself by a rectangle (see R5, R6) as shown in **Figure A-4**.

Figure A-4 Partitioning, third cut



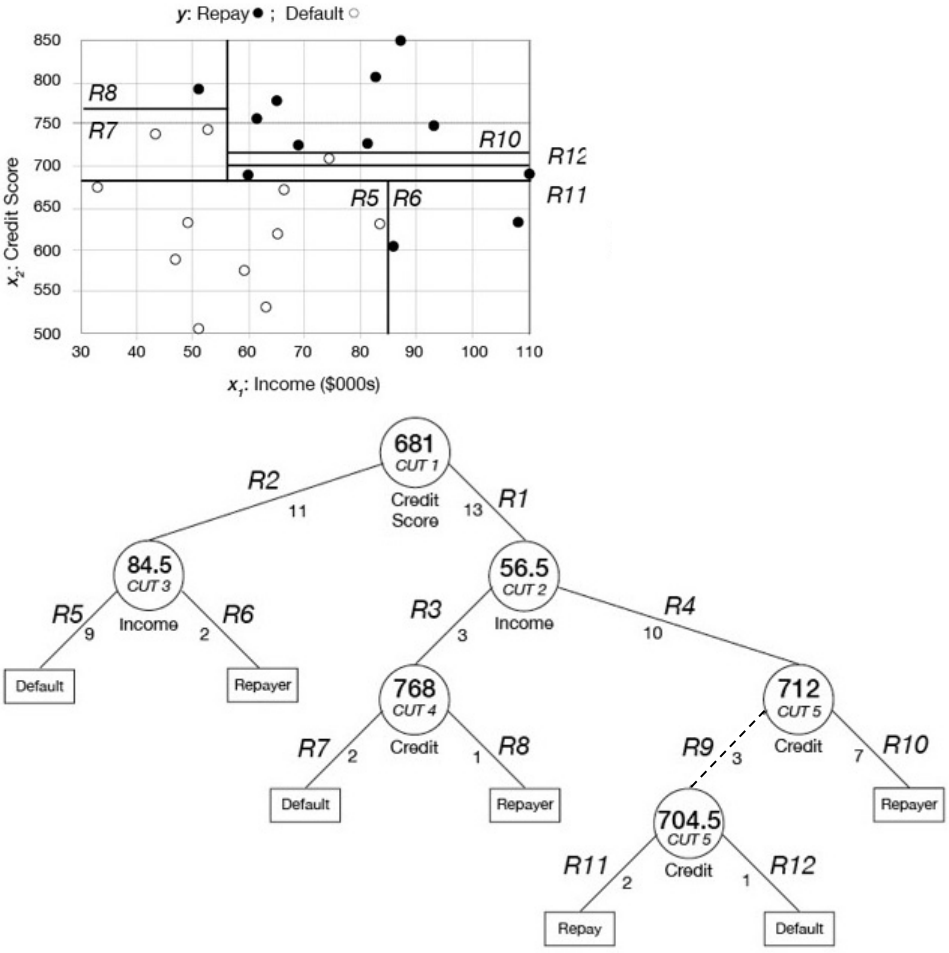
	R1	R2	R3	R4	R5	R6
Repay (black)	n/a	n/a	1	9	0	2
Default (white)	n/a	n/a	2	1	9	0

Source: Casewriter.

The Gini impurity for the full decision tree at this stage is 0.1306.

The algorithm would proceed in this fashion, determining how best to split subsequent rectangles such that purity of the rectangles was maximized until all individual data points were accounted for. **Figure A-5** shows the full decision tree.

Figure A-5 LendingClub, Fully Grown Decision Tree



Source: Casewriter.

Note: The line that connects the node at tree depth 3 (credit = 712) and the node at depth 4 (credit = 704.5) is dotted to indicate what Figel would not know about node 4 if she pruned the tree at tree depth 3.

The full set of classification rules for the decision tree in **Figure A-5** are:

- If 681 < credit score < 768 and income < $56,500 then = 1 (default)
- If credit score < 681 and income < $84,500 then = 1 (default)
- If 704.5 < credit score < 712 and income > $56,500 then = 1 (default)
- If the borrower's features do not satisfy any of these conditions then classify as = 1 (repay)

Visually, anytime Default is to the right of Repay at a terminal node, it is a potential problem because it suggests that for the same level of income, an individual with a *higher* credit score would be classified as a *defaulter* while an individual with a *lower* credit score would be classified as a *repayer*.

Appendix B: Pruning the Tree

The last several nodes of the decision tree in *Appendix A* suggested the following decisions for repayers:

If income > $56,500 and 712 > credit score > 704.5 then probability of default = 1

If income > $56,500 and credit score < 704.5 then probability of default = 0

This rule would suggest that a borrower with a credit score *greater* than 704.5 but less than 712 and income greater than $56,500 would be expected to default (= 1), while another borrower with the same income (greater than $56,500) but a *lower* credit score (less than 704.5) would be expected to repay.

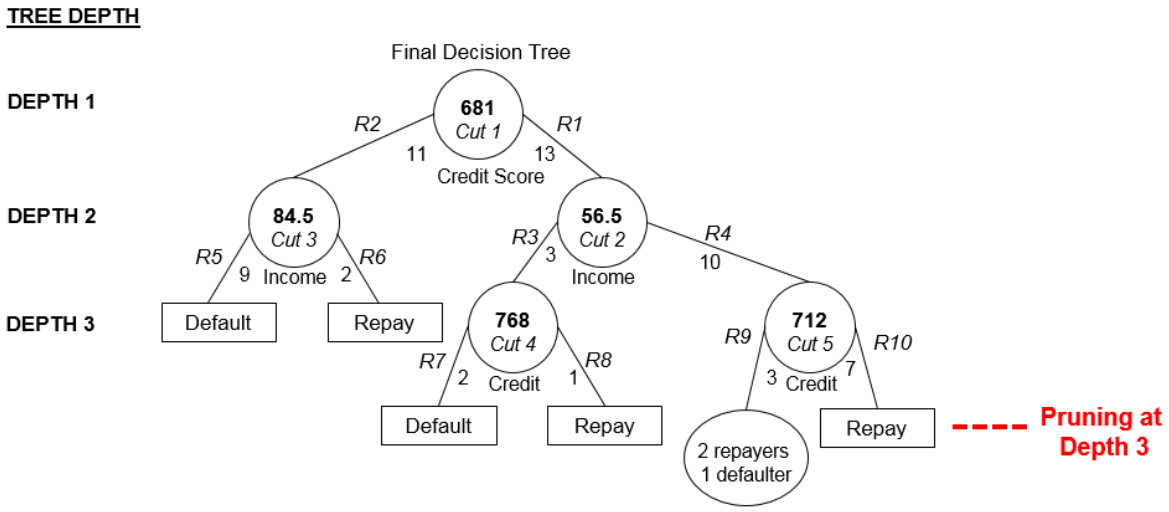
This did not accord with Figel's intuition. She wondered if this might have happened because of an error in recording the data or some idiosyncrasy in the data. She wondered if she should cut out those decision rules by pruning the tree earlier. (See **Figure B-1**.) For example, pruning the tree after three layers would give the following tree and rule for defaulters:

If 681 < credit score > 768 and income < $56,500 then = 1 (default)

If credit score < 681 and income < $84,500 then = 1 (default)

If the borrower's features do not satisfy any of these conditions then classify as = 0 (repay)

Figure B-1 Tree pruned to depth 3



Source: Casewriter.

Figel noted that this decision tree would leave an impure node at one of the ending points of the tree with one borrower with those characteristics being a defaulter and two borrowers with those characteristics being repayers. Without a pure node, Figel could only express the decision in *probabilistic* terms when she reached that node. In this case when 712 > credit score > 681, income > $56,500 there is only a 1/3 probability of the loan defaulting. On reflection this seemed reasonable. That is, borrowers who met this criteria had a 66.67% chance of being a repayer.

Although intuitively appealing, Figel would only know if pruning was a good idea if it yielded better predictions in the validation sample. By pruning, she was using a less sophisticated model, but also perhaps reducing overfitting the model to details and random factors in the training data that were not representative of the overall data.

Figel decided to prune the tree and apply both rules on a new data set (5 borrowers). She reasoned that she would choose whichever model did a better job of predicting repayers and defaulters. See **Table B-1** below.

Table B-1 Misclassification Rates for Full vs. Pruned Decision Trees on New Data

Income	Credit	Actual	Full Tree Prediction	Pruned Tree Prediction	
$60,000	680	Repay	**Default***	**Default***	(with probability of 1)
$67,000	710	Repay	**Default***	Repay	(with probability of 2/3)
$56,000	770	Default	**Repay***	**Repay***	(with probability of 1)
$61,000	702	Repay	Repay	Repay	(with probability of 2/3)
$58,000	715	Repay	Repay	Repay	(with probability of 1)
Correct Classification Rate			2/5	3/5	

Source: Casewriter.

Note: *Misclassified.

Endnotes

[1] Dean Abbott, *Applied Predictive Analytics* (Indianapolis: Wiley, 2014), p. 214.

[2] Eric Siegel, *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, Or Die* (Hoboken: Wiley, 2013), p. 160.

[3] Eric Siegel, *Predictive Analytics,* p. 176.

[4] Forest Provost and Tom Fawcett, *Data Science for Business* (Sebastopol: O'Reilly, 2013), p. 306-308.

[5] Galit Shmueli, Nitin R. Patel, and Peter C. Bruce, *Data Mining for Business Intelligence*, 2nd Ed. (Hoboken, NJ: Wiley Publishing, 2010), p. 166.