Module 2 Predictive Analytics and Data Mining

Skip the Table of Content

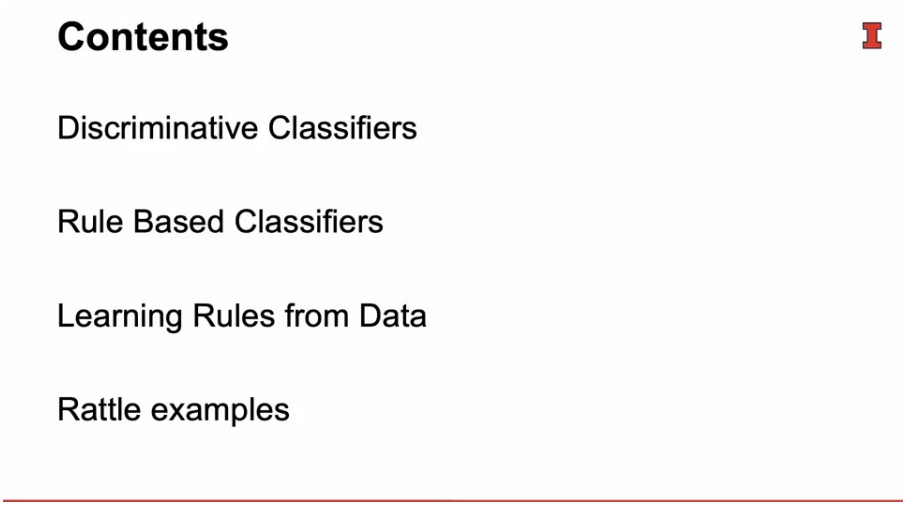
Contents

Lesson 2-1 Introduction to Discriminative Classifiers

Lesson 2-1.1 Introduction to Discriminative Classifiers

Media Player for Video

Overview - Slide 1



Discriminative Classifiers

Rule Based Classifiers

Learning Rules from Data

Rattle examples

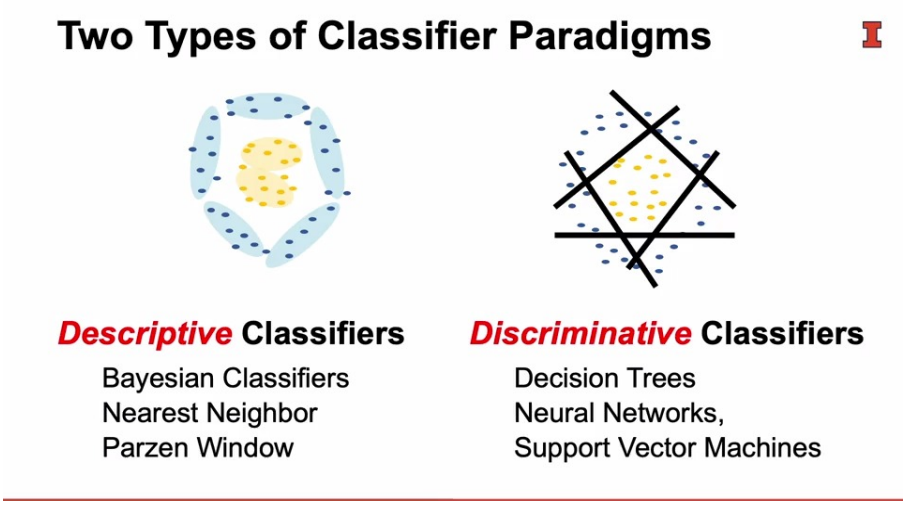
Transcript

In this module, we are going to look at another way of representing information. So, as the title says, "Knowledge can be represented in trees." What it means is that, there's a way to keep asking questions and arrive at the right answer starting from one question and saying, "Are you male or female? Are you old or young? You own a house, you don't own a house," and you can just go down the tree and then arrive at where you want to go at the answer you want to get. So, that's, we'll see that often knowledge is presented that way. As is consistent with the tools part of this course, we want to construct these trees using data rather than our experience. So, that's the meaning of this sentence, knowledge and trees, and trees constructing trees using data. Before we dive into trees, we'll briefly discuss what are the different classifiers, so partitional versus discriminative.

So, we will try, we are going into what are known as discriminative classifiers and we'll see the difference between this and what we have already seen in a previous module. Then we will come to rules, because there are different kinds of discriminative classifiers. So, we're going to look at rule-based classifiers and try to understand why rule-based classifiers actually create discrimination, discriminative

lines or surfaces to partition your data. Then we are going to then ask, how do you construct these rules from using information? So, we will look at a simple example of constructing a tree and if time permits, we will also see how to construct a forest from these trees, so which is a forest as many trees as you can see. We'll go into Rattle and start looking at these examples.

Two Types of Classifier Paradigms - Slide 2



The slide contains two images representing the two types of classifiers: Descriptive and Discriminative. Descriptive Classifiers: One cluster surrounded by several other clusters. Each clusters is contained within its own circle/elipse. Discrimiative Classifiers: A collection of dots is deliniated into clusters by several intersecting lines.

Descriptive Classifiers

Bayesian Classifiers

Nearest Neighbor

Parzen Window

Discriminative Classifiers

Decision Trees

Neural Networks

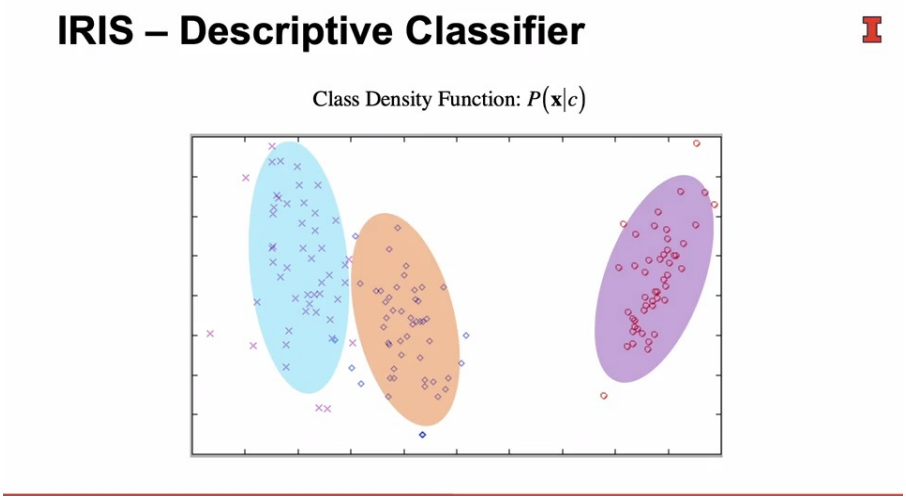
Support Vector Machines

Transcript

So, moving on, let's see the difference between the kind of classification we did so far and what we want to do in this module. So, the kind of classification we did, was by describing. So, two of them we did was the Bayesian classifier and the Nearest Neighbor classifier. So, basically what these classifiers do, is identify similar areas and say these are similar and these are similar. That's one way of doing it and we didn't do Parzen Window, but Parzen Window is another way of doing the same thing. Identifying clusters similar things and saying this is one, this is another, another. Another way of doing it, is instead of saying, "We will identify the clusters," is to be able to draw these lines in space or whatever number of dimensions in this case, they're given a two-dimensional representation.

There are all red dots are in between the four lines and outside these four lines, you've got all the blue dots. So, basically, we have partitioned it using lines, discriminated it and there are many ways to do it. We're going to see really one thing in this course, which are decision trees. Neural networks fall into this and more complex things like support vector machines also fall into this. So, the whole idea here is you want to partition the space and saying using these lines, I'm able to partition the space into regions and these regions are distinct. Let's take again the difference between one and the other. So, for example, I could, I know it looks like a sausage, but it is not, the same data on both sides and you've got a description classifier, that is a discriminative classifier. You can see visually what's the difference between one and the other.

IRIS - Descriptive Classifier (1 of 2) - Slide 3

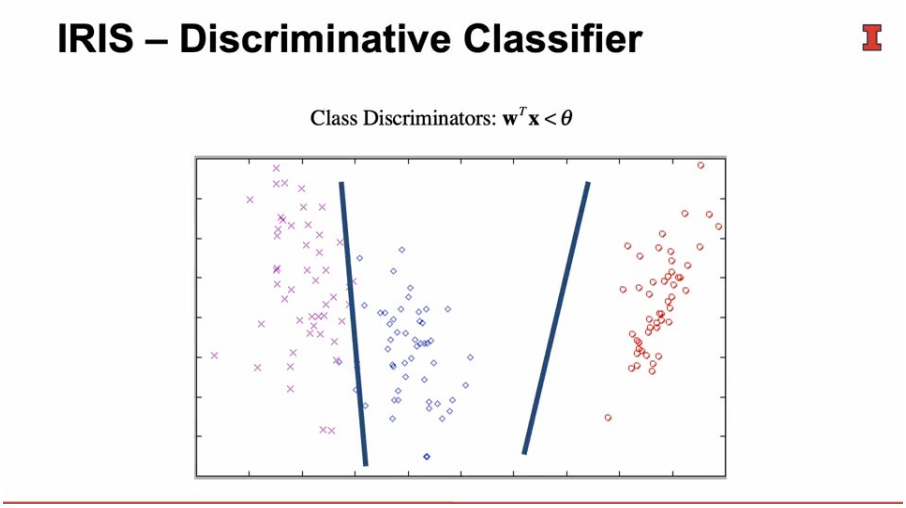


The slide contains a scatter plot called Class Density Function: P(x|c). Points in the scatter plot are grouped into three clusters using different colored elipses.

Transcript

Remember we did the IRIS dataset? And here is a descriptive classifier, one cluster represents one kind of IRIS and other, another and another.

IRIS - Descriptive Classifier (2 of 2)- Slide 4

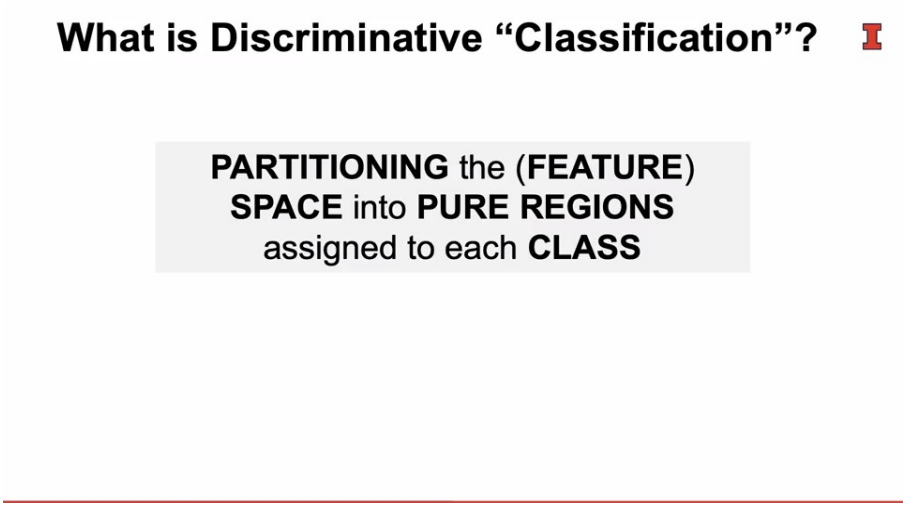


The slide contains a scatter plot called Class Discriminators: w^T x<θ. Points in the scatter plot are grouped into three clusters, separated by two verticle lines.

Transcript

You can also imagine that I could draw lines to separate them out and that becomes a discriminative classifier. So, you got two lines and they're dividing this into three places that may not always be two lines. There could be many lines, dividing it into areas and each of these areas would have a concentration of a similar type of object.

What is Discriminative "Classification"? - Slide 5

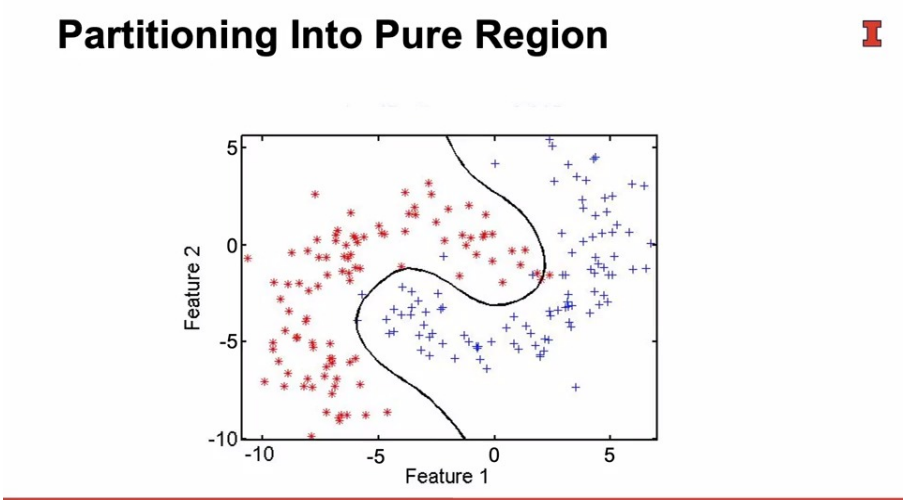


Partitioning the **(feature) space** into **pure regions** assigned to each **class**

Transcript

So, what is discriminative classification? It is partitioning, that's the formal definition, space into pure regions assigned to each class. If you like that better than the picture, fine.

Partitioning into Pure Region (1 of 3) - Slide 6

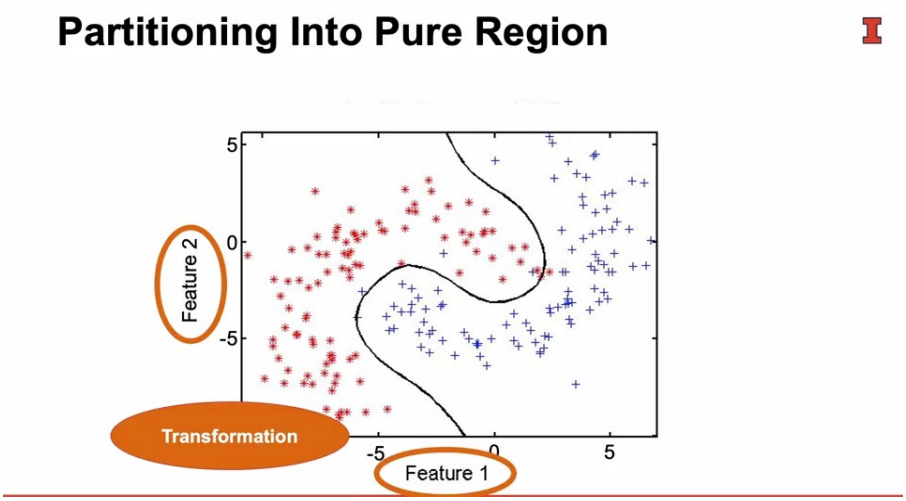


The slide contains a scatter plot. The x-axis is labeled Feature 1 and it ranges from -10 to 5 in increments of 5. The y-axis is labeled Feature 2 and it ranges from -10 to 5 in increments of 5. There is a curve classification boundary in the middle, dividing the points into two clusters differentiated by color. Some points near the boundary are misclassified (coded with the incorrect color).

Transcript

So, here's an example, and maybe it's worth animating it, showing you this way. So, here is a classification boundary and everything on the right is blue. Everything on the left is brown. You see there are some objects which had been misclassified. You can see that, right?

Partitioning into Pure Region (2 of 3) - Slide 7

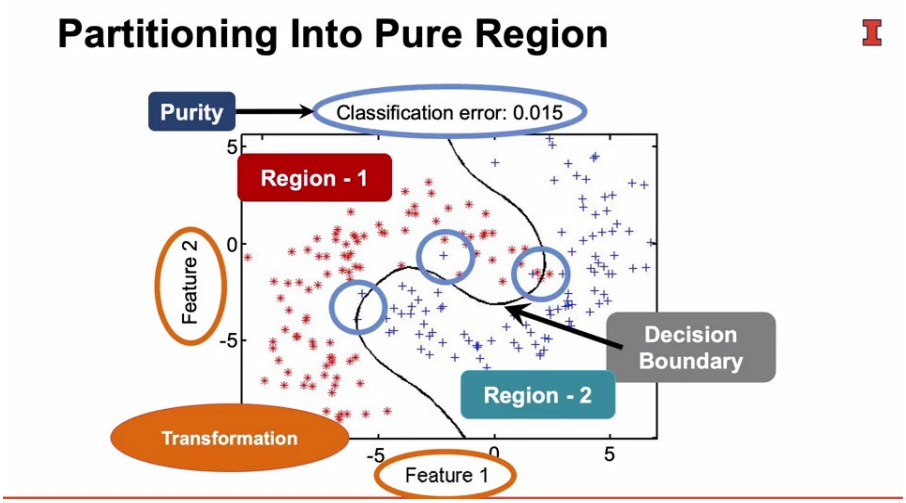


The slide contains the same graph as Slide 6 - Partitioning into Pure Region (1 of 3), with a circle on Feature 1 and Feature 2, and transformation on the left bottom of the graph.

Transcript

So, there are several steps in presenting it like this. We have transformed the data, retained the two most important features using maybe a principal component analysis, we've drawn the graph and

Partitioning into Pure Region (3 of 3) - Slide 8

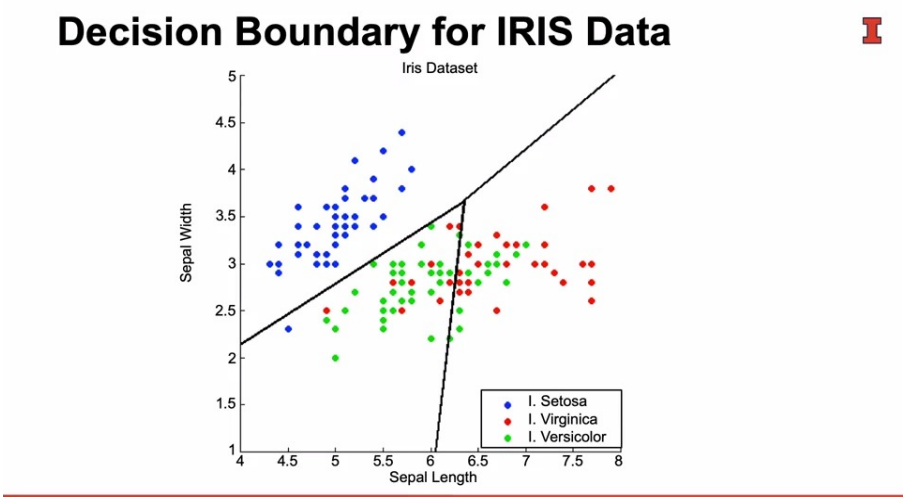


The slide contains the same graph as Slide 7 - Partitioning into Pure Region (2 of 3) but with more details labeled. Purity: Classification error: 0.015. The curve is the Decision Boundary. Some misclassified points are circled. The left is Region 1 and the right is Region 2.

Transcript

created a decision boundary and then ask, is region one in region two? Then ask, how many mistakes are there? You can see that a few of the points have been misclassified. So, the classification error, we say is 0.015, it's the number of points which have been misclassified.

Decision Boundary for IRIS Data - Slide 9

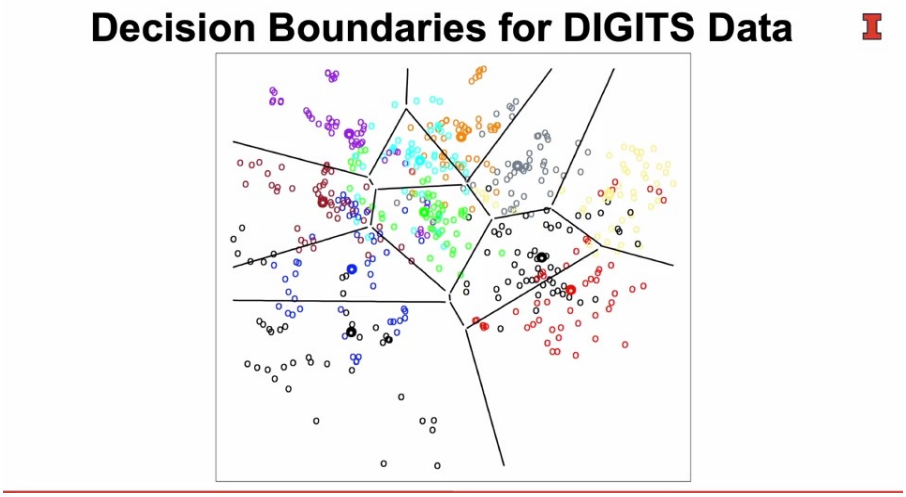


The slide contains a scatter plot using Iris Dataset. The x-axis is labeled Sepal Length and it ranges from 4 to 8 in increments of 0.5. The y-axis is labeled Sepal Width and it ranges from 1 to 5 in increments of 0.5. There are three clusters differentiated by color separated by intersecting lines. There are some misclassified points (coded with the incorrect color).

Transcript

So, here is decision boundary for the IRIS data. You will remember it. The axis is the sepal length versus width and you can see it's no longer a simple boundary, it's three different lines and it's partitioning the space into three parts and it's doing it into the three components and you have the color code, the Setosa, the Virginica and the Versicolor and they are in separate parts of this.

Decision Boundary for DIGITS Data - Slide 10



The slide contains a scatterplot called Decision Boundaries for DIGITS Data. The scatter plot has multiple intersecting lines partitioning points into 11 clusters, each representing a separate digit. Some errors exist.

Transcript

This data you have not seen but if you do a lot of analysis this kind, this is the digits data from zero, one, two, three so on to 10 I think, the 11 areas each corresponds to a digit. It's been transformed into a two-dimensional space and these lines are partitioning them into areas which correspond to each digit. It's not perfect. There is a little bit of error out there, as you can see.

Lesson 2-2 Model Complexity

Lesson 2-2.1 Model Complexity

Media Player for Video

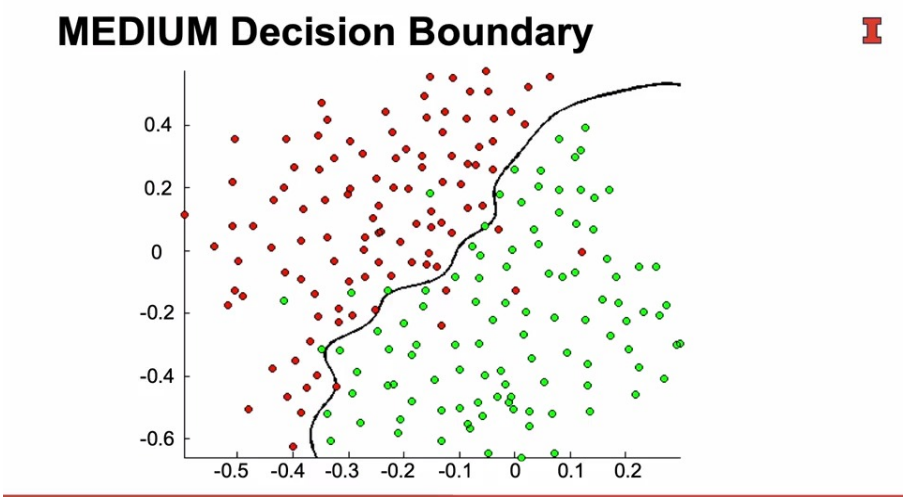
Simple Decision Boundary? - Slide 11

The slide contains a scatter plot with a straight diagonal line partitioning data. The points in the upper triangular area are red and points in the lower triangular area are green. The range of x-axis is from -0.6 to 0.3 in increments of 0.1 and the range of y-axis is from -0.8 to 0.6 in increments of 0.2. Error exists.

Transcript

Question is, how should we like our decision boundary? Should it be simple like this? It's like very simple.

Medium Decision Boundary (1 of 2)- Slide 12

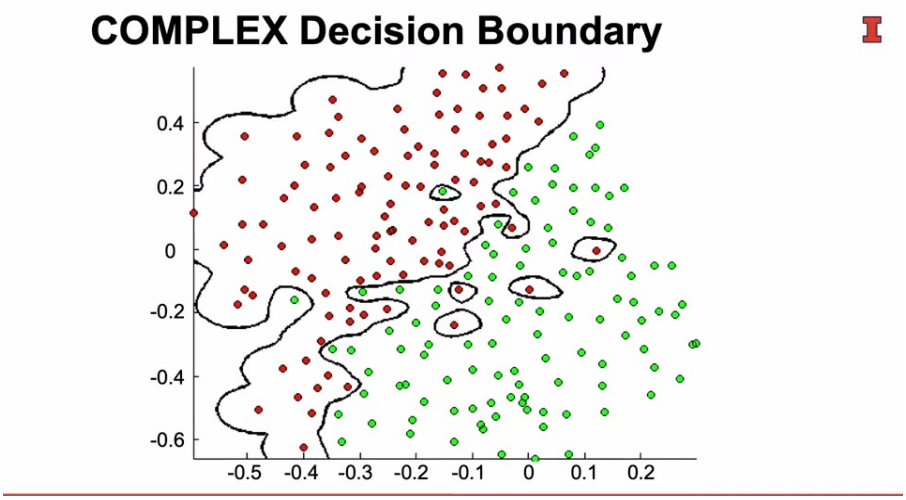


This slide contains a scatter plot that uses the same dataset as the previous slide. The range of x-axis and y-axis is the same as Slide 11 - Simple Decision Boundary? The decision boundary is a complex curve. The misclassification error is reduced.

Transcript

Second, would you like it to follow the data as closely as it can? This is doing its best trying to reduce the misclassification error.

Medium Decision Boundary (2 of 2)- Slide 13

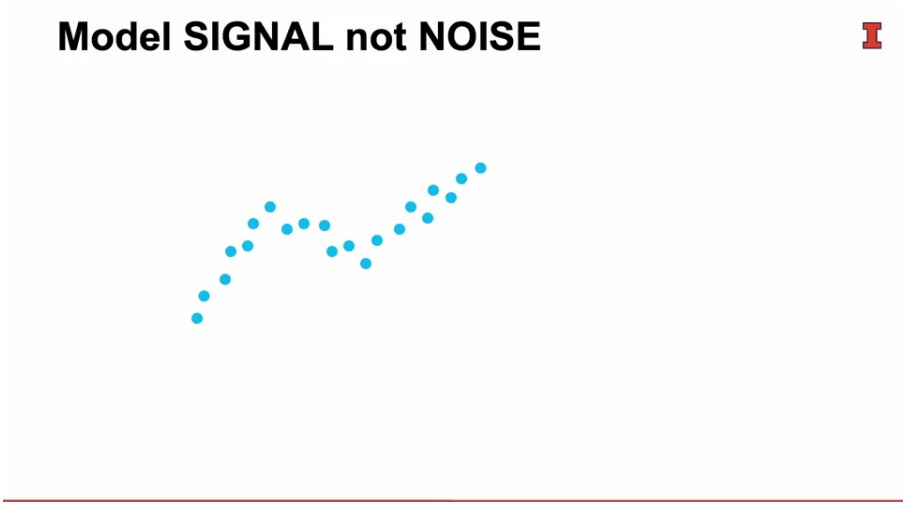


The slide contains a scatter plot which has the same data as the Slide 11 - Simple Decision Boundary?. The decision boundary is very complicated and there are circles. No error exists.

Transcript

But how about this? Now, this is actually partitioning and making it much more complex, but you have captured all the bronze correctly and all the greens correctly. So, which one do you like? Now, there is a trade-off between a simple boundary and a complex boundary. A simple boundary doesn't do well both in the training set on the validation set, but it may be preferred when you have a lot of noise in the data. A complex boundary does pretty well in the training data, it's able to fit any data you want, but the difference is that when you try to take it and predict it, predict something using these boundaries, you may find that you're making a lot of errors. So, what you really want is something which is just right, which is somewhere between a simple and a complex, somewhere in between.

Model Signal not Noise (1 of 3) - Slide 14

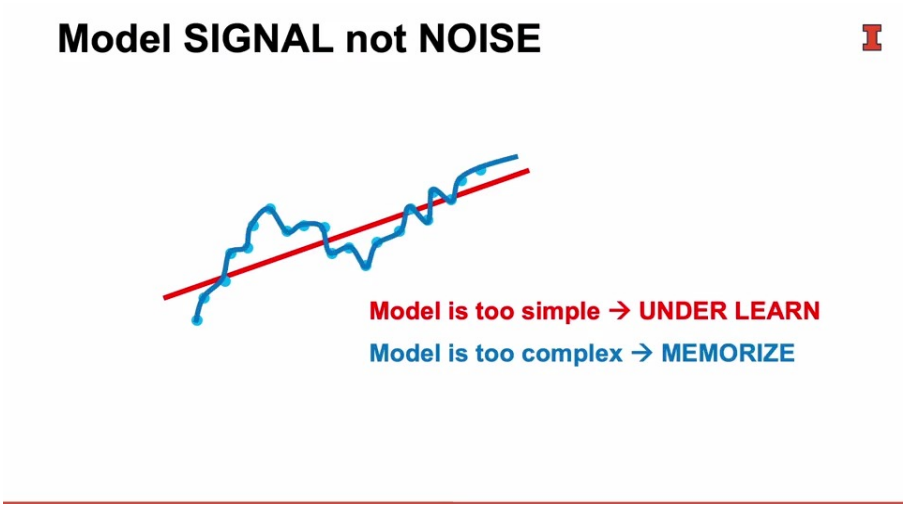


The slide contains a scatter plot shape as a cubic curve.

Transcript

So, as my friend Dr. Silage Kumar who's slides, some of them I'm using, he says model the signal not the noise. What they mean is, don't react to noise that's what the operations people say, right. If you just follow noise, you're reacting to superstition. For example, if you thought every time you touched your nose and you hit a home run, so you keep touching your noise before you go to bat, and then some people may say, it's just falling noise and it is superstitious. So, in the data tools side they say, model is too simple, it doesn't learn the pattern sufficiently.

Model Signal not Noise (2 of 3) - Slide 15



The slide contains a plot that has two lines: a stright line at 45-degree angle representing a simple model and a complex curve representing a complex model.

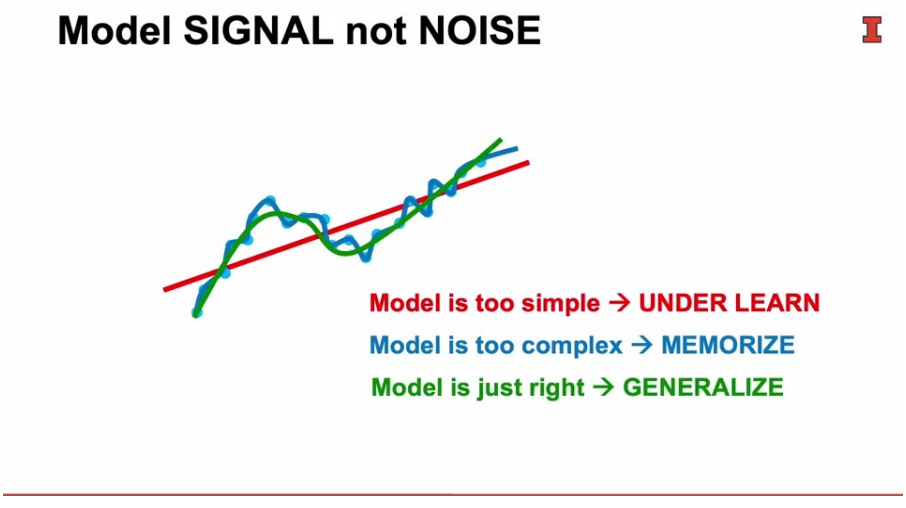
Model is too simple → Under Learn.

Model is too complex → Memorize.

Transcript

Model is too complex because it's falling every point, and you're simply memorizing by rote, but you see something new which is what we accuse our students like don't memorize, but try to learn, and what we mean by that is just don't repeat what you have seen.

Model Signal not Noise (3 of 3) - Slide 16



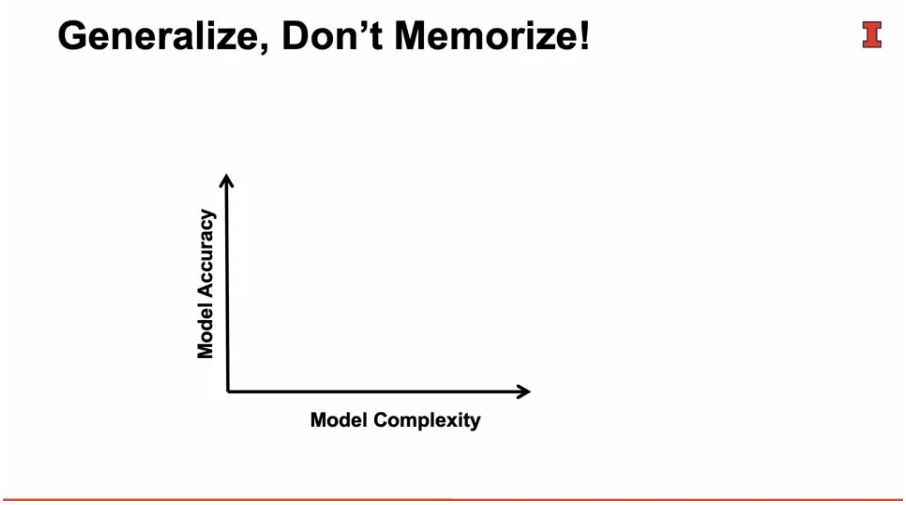
The slide contains the same information provided on Slide 15 - Model Signal not Noise (2 of 3) with a third simple curve added

Model is just right → Generalize.

Transcript

So, this model is too complex, and we say don't memorize, what we you really want you to do is find a model we'd like the green one, which is just right which allows you to generalize. In general, most classification problems will have these features from simple to complex, and just right.

Generalize, Don't Memorize! (1 of 4) - Slide 17

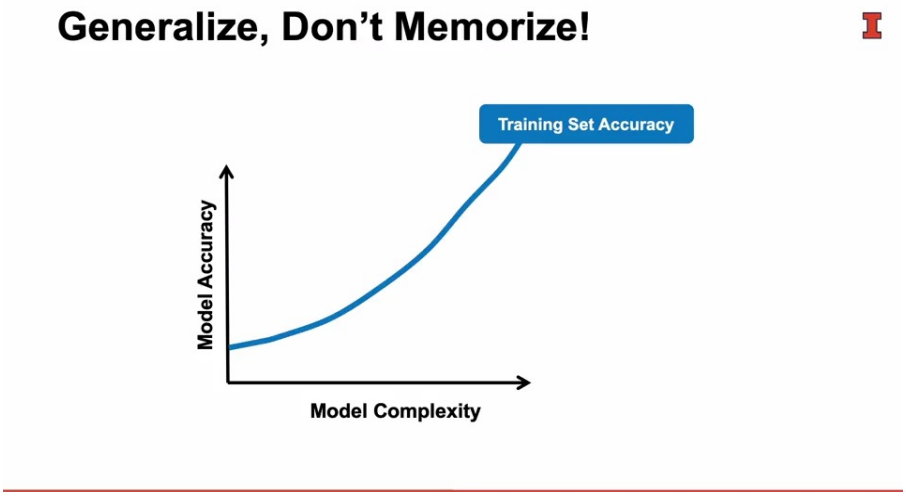


The slide contains a coordinate. The horizontal x-axis is Model Complexity and the vertical y-axis is Model Accuracy.

Transcript

So, here is the usual trade-off which we have seen, but once again it's doesn't hurt us to repeat it.

Generalize, Don't Memorize! (2 of 4) - Slide 18

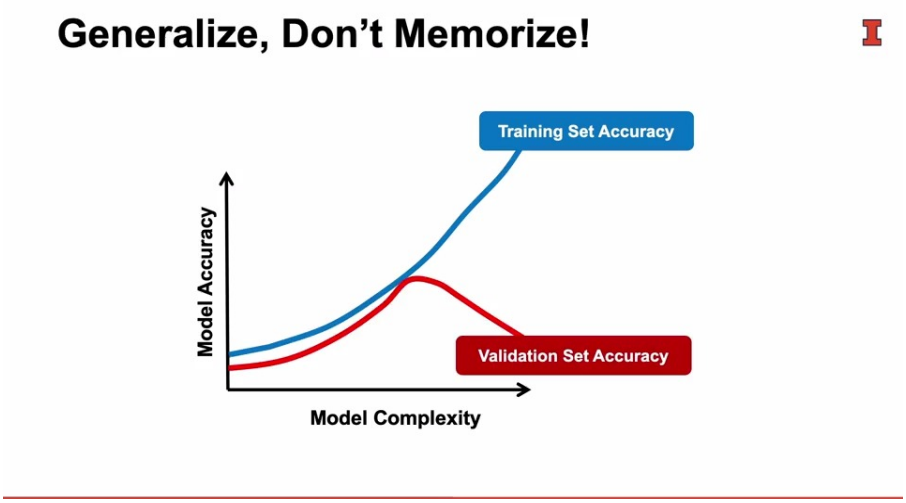


This slide is the same as Slide 17 - Generalize, Don't Memorize! (1 of 4), with a curve called Training Set Accuracy. The curve is monotonously increasing.

Transcript

As we increase the complexity in most models there will be some lever which we can use we will see, which will make it track everything in the training data, you have more larger number of gaze, more number of features, it would just track much and much better. So, it'll improve the accuracy.

Generalize, Don't Memorize! (3 of 4) - Slide 19

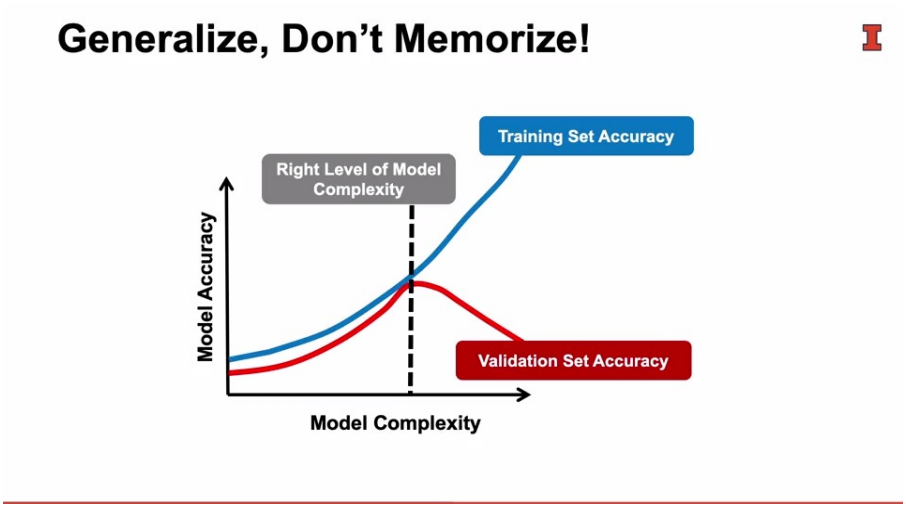


The slide is the same as Slide 18 - Generalize, Don't Memorize! (2 of 4), with second curve called Validation Set Accuracy. The curve first goes up and then goes down.

Transcript

But, as you make the model more and more complex, it's just memorizing, and after this point, it doesn't do very well in the validation set. So, you stop when the model complexity is such that the validation accuracy you're satisfied. Maybe messed up a little bit before that itself, because the other thing as you remember, we want models which are parsimonious use as little features or transform the features to get a little noise as possible.

Generalize, Don't Memorize! (4 of 4) - Slide 20

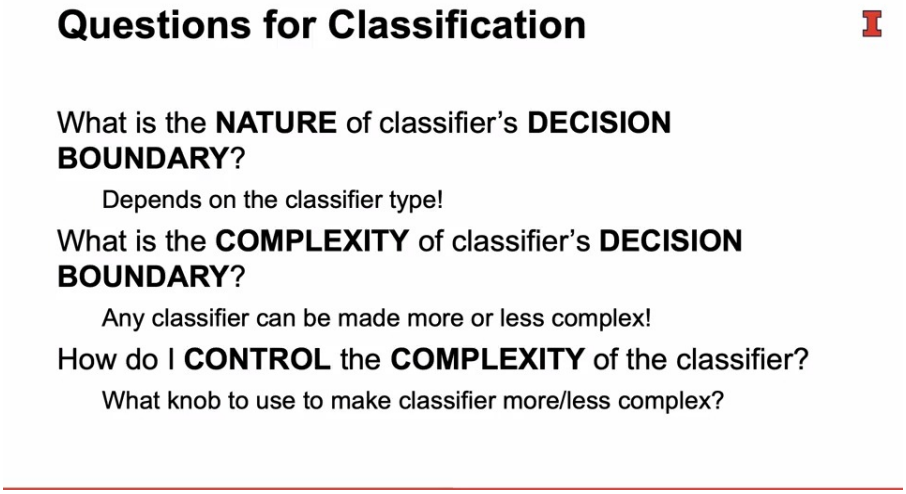


This slide is the same as Slide 19 - Generalize, Don't Memorize! (3 of 4) with a vertical straight dash line bisecting both curves where the Validation Set Accuracy reaches its peak.

Transcript

So, the main takeaway from here is the trade-off in creating these boundaries, and how complex you got to make them. Some sense there's an art but there is also a science to this, we are saying stop where the validations set accuracy is reasonably high and you are satisfied with it.

Questions for Classification - Slide 21



What is the **Nature** of classifier's **Decision Boundary**?

Depends on the classifier type!

What is the **Complexity** of classifier's **Decision Boundary**?

Any classifier can be made more or less complex!

How do I **Control** the **Complexity** of the classifier?

What knob to use to make classifier more/less complex?

Transcript

So, here are the questions. We have to ask ourselves when we develop a classifier. One, what kind of decision boundary it is? I know this is a frustrating answer, that it depends on the classifier type. Is it a straight line, is it a curved line? How complex it is? Basically, we have to understand that most classifiers can be made as complex as you want or less complex. As we saw even in K-NN if you think of it, if you use all the points it's very simple, but all you will be saying is that any point belongs to the majority class. So, it can be made very complex or very simple. You must know for each classifier through experience how to tweak to get a higher complexity. That's part of experience, part of working with the tools we have, and the new tools that you're going to pick up because there's no guarantee that the tools we talked in this course are the only ones which are going to be there in the future. How do you know it's complex enough? So, basically the trade-off is between the degree of noise versus the structure. As you make it more and more complex, you're tracking the noise much better, but we also have to make sure that you are capturing the structure in the data, and so trade-off is really there, are you capturing more structure in the data or are you just falling the noise.

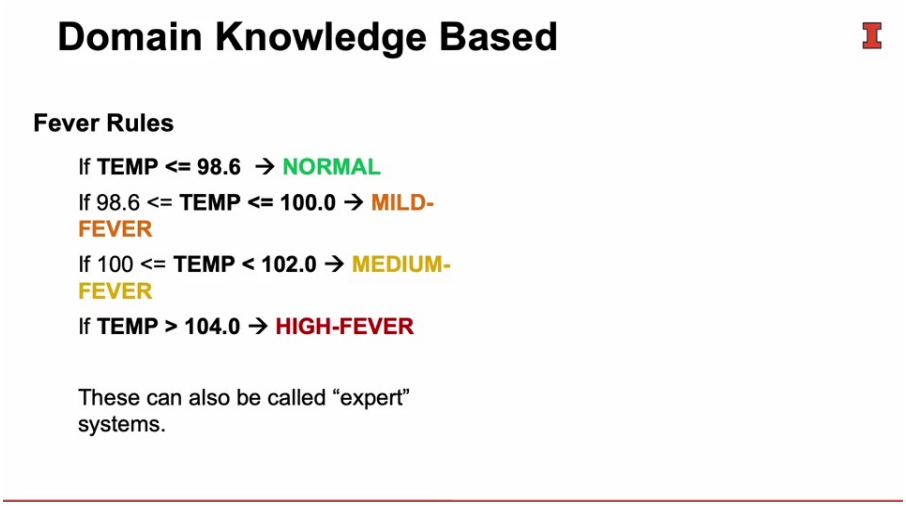
In general, if the data is very noisy, it is safer to have a less complex model. Finally, which classifier to use? It again depends, and my suggestion is, remember in the last module we give an exercise, we asked you to flip the two type of classifiers and compare. Some of them generally perform well and we'll see one of them today, and they are robust classifiers, and some ones that you should use for K-NN, I like that because it's simple and allows you to come up with a base level of classification and you can try to improve from that. So, it depends also on the nature of data. We will see today a trade-off that depending on whether the data is numerical or categorical, your classifier is going to change, whether you got labels of your data and whether you don't have labels of the data, which is structured versus unstructured, your classifier will probably change. So, we will see that.

Lesson 2-3 Rule Based Classifiers

Lesson 2-3.1 Rule Based Classifiers

Media Player for Video

Domain Knowledge Based (1 of 2) - Slide 22



If **Temp ≤ 98.6 → Normal**

If 98.6 ≤ **Temp ≤ 100.0 → Mild-Fever**

If 100 ≤ **Temp < 102.0 → Medium-Fever**

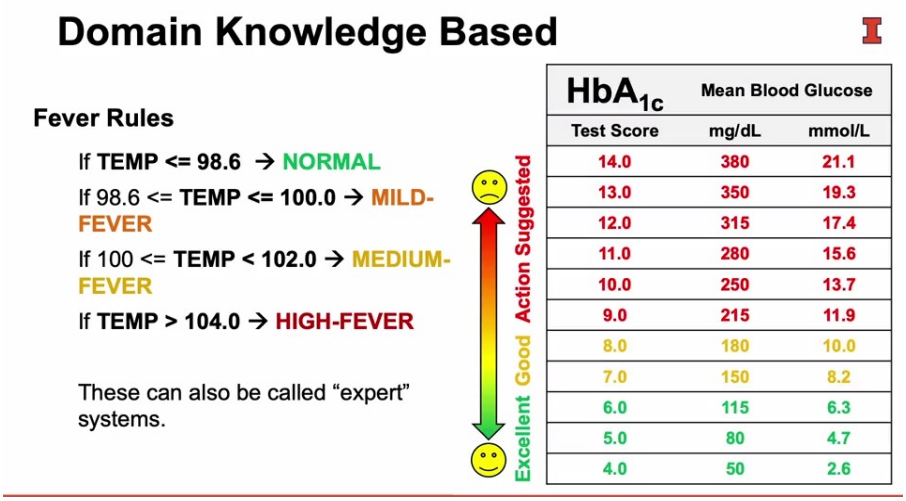
If **Temp > 104.0 → High-Fever**

These can also be called "expert" systems.

Transcript

So, we're going to talk about rule-based classifiers. You're probably familiar with a bunch of rules that people use. For example, think about this, if your temperature is less than 98.6, your normal, it's between 98.6 and 100, maybe a light fever, 100 and 102, medium fever more than 104, high fever, go and talk to your doctor.

Domain Knowledge Based (2 of 2)- Slide 23



The slide contains the same content as Slide 22 - Domain Knowledge Based (1 of 2). With an added table of test score ranging from 4.0 to 14.0. 4.0 to 6.0 is labeled as excellent, 7.0 to 8.0 is labeled as good, and 9.0 to 14.0 is labeled as action suggested. Table data is listed below:

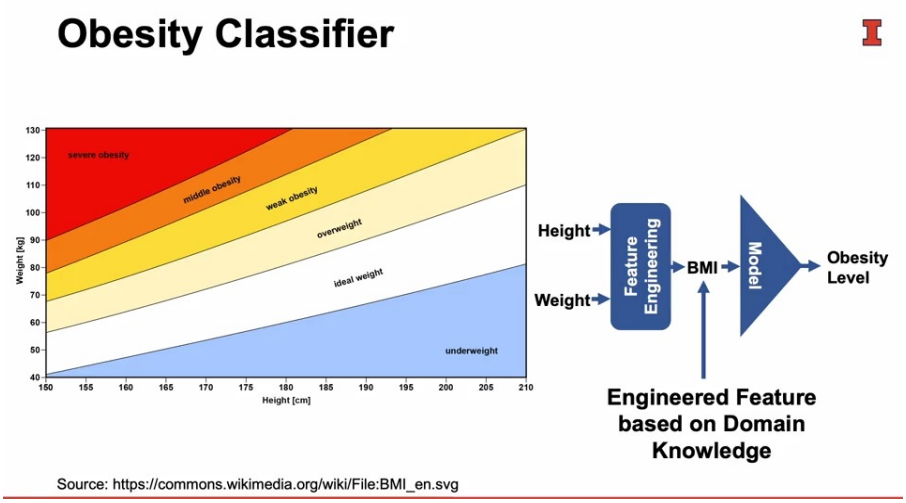
HbA1c Mean Blood Glucose

Test Score	mg/dL	mmol/L
14.0	380	21.1
13.0	350	19.3
12.0	315	17.4
11.0	280	15.6
10.0	250	13.7
9.0	215	11.9
8.0	180	10.0
7.0	150	8.2
6.0	115	6.3
5.0	80	4.7
4.0	50	2.6

Transcript

Let's say diabetic or not, on the first column, you have the HbA1c score and it's got a range. Below six is okay, between seven and eight is good, and above that is in diabetic region where some action is needed. It transforms into measurements. We will not go into the details, but here it is. How did we get this? We got this through a lot of data and people came up with these rules, and people accept these as rules to be followed, at least for adults.

Obesity Classifier - Slide 24



The slide contains an obesity classifier. On the x-axis, Height/cm. On the y-axis, Weight/kg. Five lines partition the rectangle into six areas: underweight, ideal weight, overweight, weak obesity, middle obesity, and severe obesity.

Height, Weight → Feature Engineering → BMI → Model → Obesity Level.

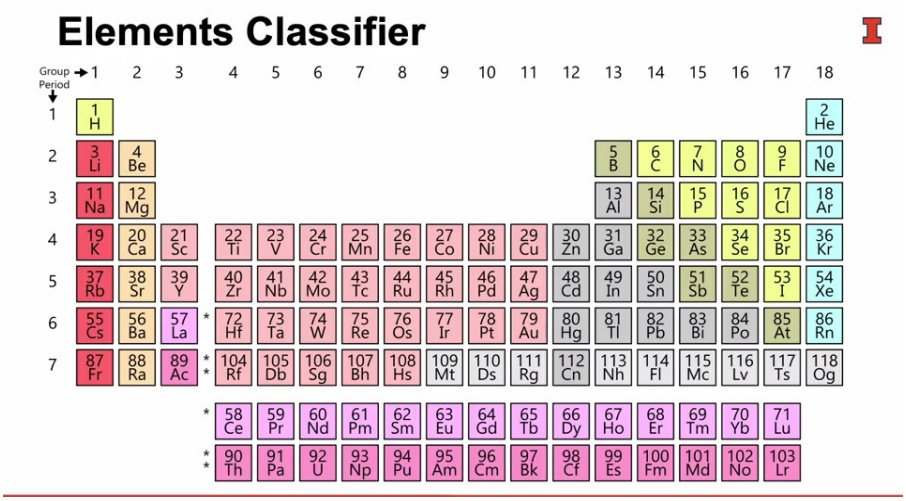
Engineered Feature based on Domain Knowledge is pointed to BMI.

Source: Wikimedia[1]

Transcript

Let's take obesity classifiers. You might have seen stuff like that. You notice again, you have clean boundaries. In the previous one, you had boundaries of temperatures. In this case, you have weight on one axis and height on the other axis, and you got boundaries. Depending on where you fall, they say you're obese or not and that comes through a way of data which they have collected. They have actually transformed these numbers into what they call as the BMI , which is the body mass index chart for adults. So, they take the height and weight, you can put it in this chart, and from this chart, you can get the BMI, and the BMI is giving you the obesity level of that. You can see the use of a discriminative classifier here. We want to develop things like that as we go on.

Elements Classifier - Slide 25

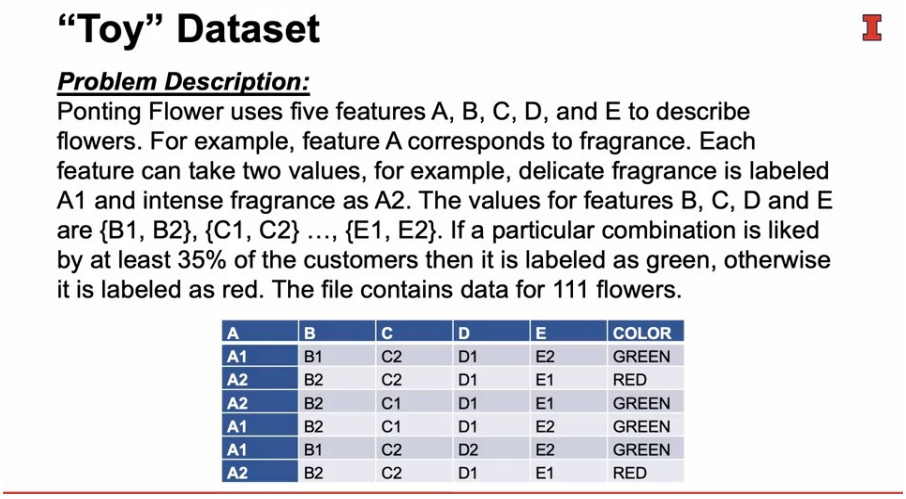


The slide contains a periodic elements chart.

Transcript

So, similarly, if you took the periodic elements chart, somebody said, okay, depending on how the atoms are structured, you can say whether you have a metal, whether you have a gas which is neutral, whether you have a radioactive material. I'm not very good at this, but as far as I'm concerned, you can see the classification boundaries are very clean depending on the structure of the atoms and how many electrons are in the outer shell. That was the theory based on which they came up with this classification. Again, a way of discriminating one group of elements from another group of elements.

"Toy" Dataset - Slide 26



Problem Description:

Ponting Flower uses five features A, B, C, D, and E to describe flowers. For example, feature A corresponds to fragrance. Each feature can take two values, for example, delicate fragrance is labeled A1

and intense fragrance as A2. The values for feature B, C, D and E are {B1, B2}, {C1, C2}..., {E1, E2}. If a particular combination is liked by at lease 35% of the customers then it is labeled as green, otherwise it is labeled as red. The file contains data for 111 flowers.

Features of Ponting Flower

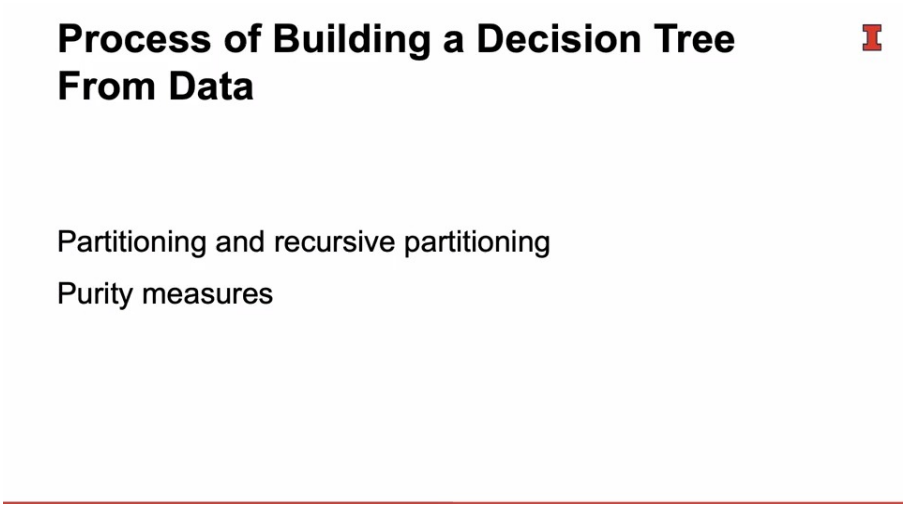
A	B	C	D	E	Color
A1	B1	C2	D1	E2	Green
A2	B2	C2	D1	E1	Red
A2	B2	C1	D1	E1	Green
A1	B2	C1	D1	E2	Green
A1	B1	C2	D2	E2	Green
A2	B2	C2	D1	E1	Red

Transcript

So, let's reverse this question. Instead of saying, okay, I have these rules, now, let's say I gave you data, I didn't give you the rules and said, tell me what rules can you infer from that? You also need to have a sense of what class this object belongs. So, there's a slight drawback here that you have data and for each of these data points we know to which class they belong to using some other method outside your system, and now you're saying, using this data can I then create rules which will allow me to classify the objects into these different classes. So, I'll try to illustrate it using first small example as we have done in the past, and then I'll take you to the software. So, here's a toy example. This example is a subset of a much bigger example. There is a company called Ponting which uses five features to describe flowers. It's trying to classify flowers into the ones that are popular and those which are not. So, it color-coded them.

Green means popular, red means not. It defines a popularity index, if at least 35 percent of the people like the flower, they classify it as popular. Each type of flower they carry, it's got five features: A, B, C, D, E. Feature A could be fragrance, and delicate fragrance versus intense fragrance, we call them A1. Feature A1 corresponds to delicate, A2 to intense and similarly, B1, B2. So, B has got B1 and B2, two categories. Feature C has got two categories: C1 and C2. I think D has got D1 and D2, and they similarly have the fifth category which they label as E1 and E2. So, we have five features and each feature has got two possible values except one which has got three. So, how do we build our tree? The idea is very simple. We break the data into maybe two or three parts, then each of these parts we break it again.

Process of Building a Decision Tree from Data - Slide 27



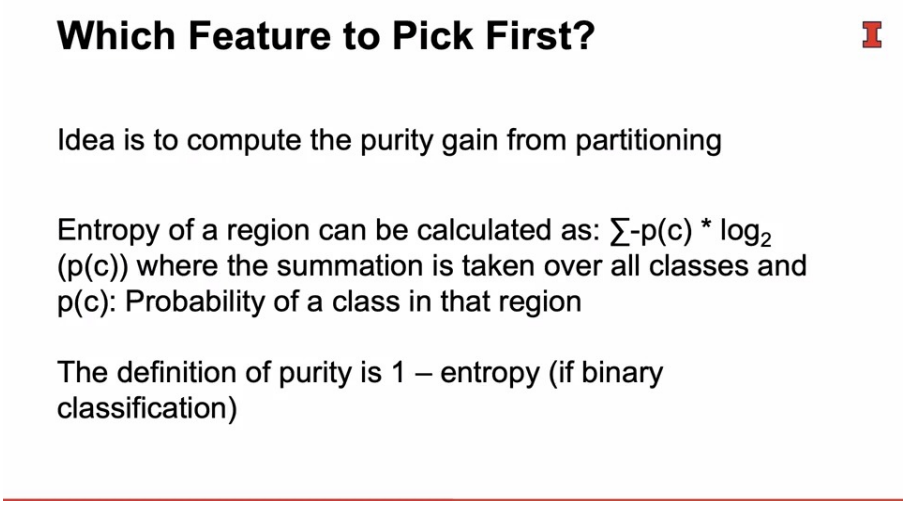
Partitioning and recursive partitioning

Purity measures

Transcript

So, we partition, then we take each partition, we recursively repartition, and what we're looking at is whether the classification is getting purer or not. So, basis for partitioning is what we call the purity measures. So, how do you compute the purity? So, which feature to pick depends on whether it improves purity, but how do you measure purity? So, I'll give you one measure. There are various other measures.

Which Feature to Pick First? (1 of 4)- Slide 28



Idea is to compute the purity gain from partitioning

Entropy of a region can be calculated as: $\sum -\mathrm{p}(\mathrm{c}) \times \log_2(p(c))$ where the summation is taken over all

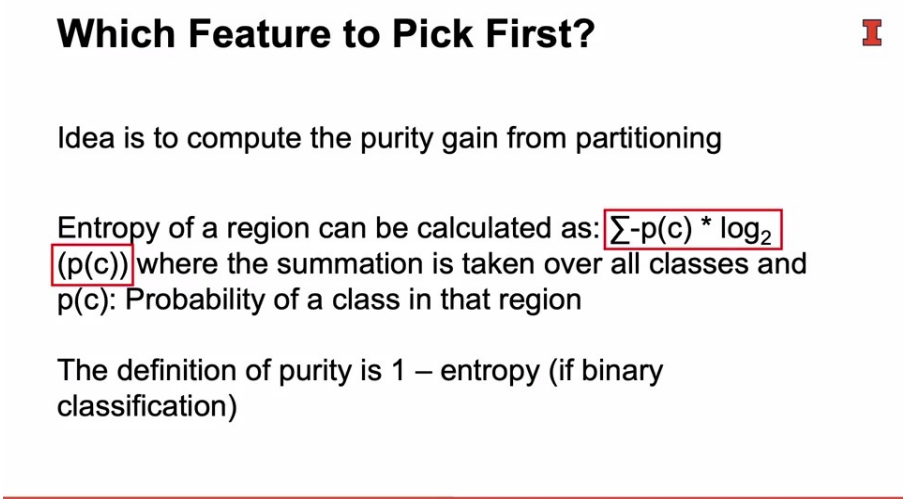
classes and p(c): Probability of a class in that region

The definition of purity is 1 - entropy (if binary classification)

Transcript

So, in this measure, we call it the entropy measure. The entropy measure is called an information measure. It comes from the theory of communications, but it's given by this formula, if you like. If you don't like it, that's fine, but you should understand what this entropy means.

Which Feature to Pick First? (2 of 4)- Slide 29

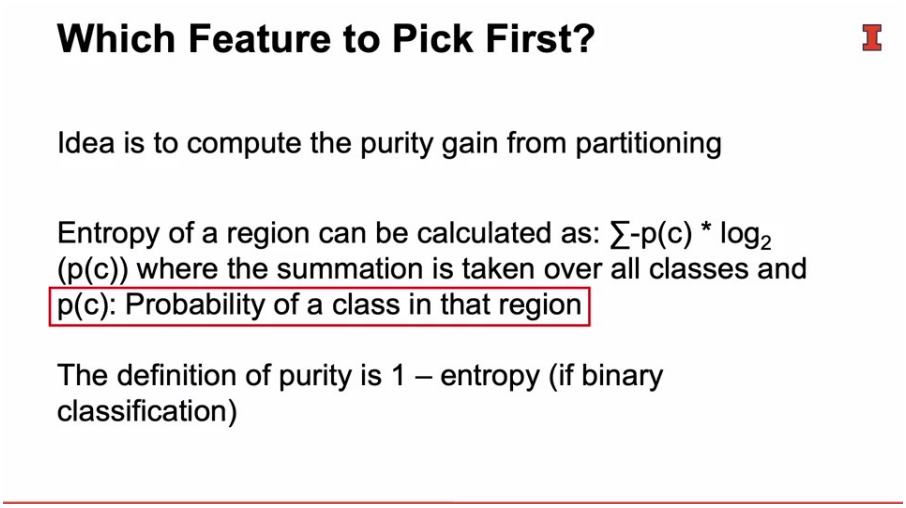


This slide is the same as Slide 28 - Which Feature to Pick First? (1 of 4), with a red circle on $\sum -\mathrm{p}(\mathrm{c}) \times \log_2(p(c))$.

Transcript

What this formula is doing is, you're taking a sum over all the classes.

Which Feature to Pick First? (3 of 4)- Slide 30

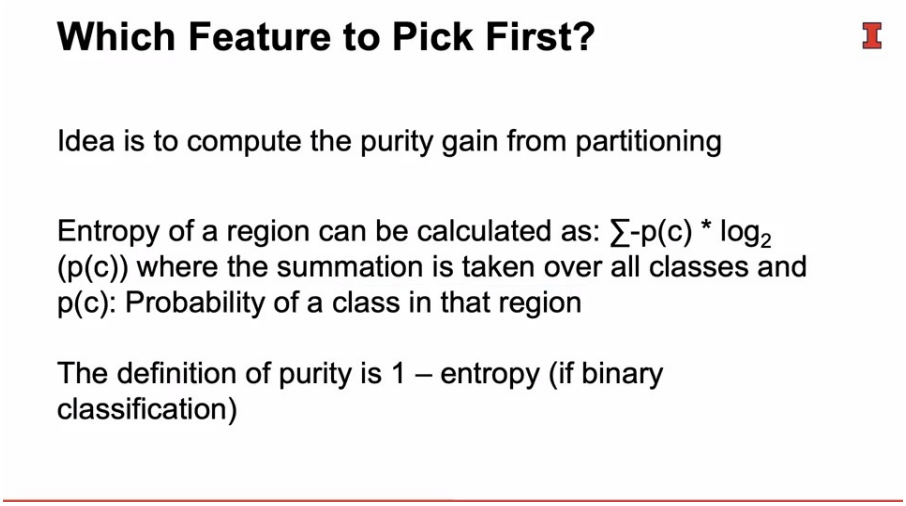


This slide is the same as Slide 28 - Which Feature to Pick First? (1 of 4), with a red circle on p(c): Probability of a class in that region.

Transcript

P(c) is the probability of the fraction of the objects that belong to class C. The sum is taken over all C. The C is the set of classes, it could be red and green or whatever it is.

Which Feature to Pick First? (4 of 4)- Slide 31



This slide is the same as Slide 28 - Which Feature to Pick First? (1 of 4).

Transcript

You're multiplying it by the log of the probability, is taken to the base 2 in this example. So, what does this formula mean? What you have to understand is if everything belongs to one class, one of these probabilities will be one and all the other properties will be zero. It's a pure classification. In that case, the entropy will be zero. If one of these probabilities is zero, then the p is zero and so that term will also be zero. So, if everything is in one category, your calculation of the entropy will be zero, the value of your entropy will be zero. So, what you really want is as small an entropy as possible. Where will it be highest? It will be highest when the number of objects in each class is the same, so what we say that distribution of object is uniform. At that point, the entropy will be the largest. That's the state in which you can say nothing clearly about whether an object belongs to class 1, or class 2, or class 3.

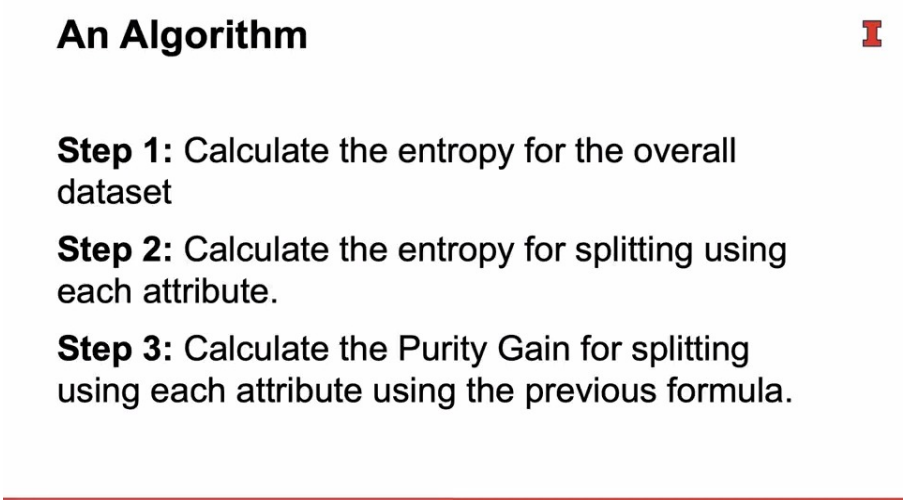
So, there are two ranges. One, when everything has got equal probability and entropy is the highest, and one when everything belongs to just one category where entropy is zero. So, you sum this up. Generally, you also have to do one thing. If it is binary classification, you can just leave it alone. That means it's classified into red or green, that's okay. That's binary. But if you have more than two categories, you also divide this by the log to the base 2 of the total number of categories, call it capital C. What it does is, it allows this value of entropy to stay in the region 0-1. That's the idea. Now, purity is the opposite of entropy. Now, entropy is between 0-1. So, one minus entropy is called purity of a region. So, what you're really trying to do is trying to improve the purity and the best you can get is one, the least you can get is zero.

Lesson 2-4 Entropy and Decision Trees

Lesson 2-4.1 Entropy and Decision Trees

Media Player for Video

An Algorithm (1 of 2) - Slide 32



Step 1: Calculate the entropy for the overall dataset.

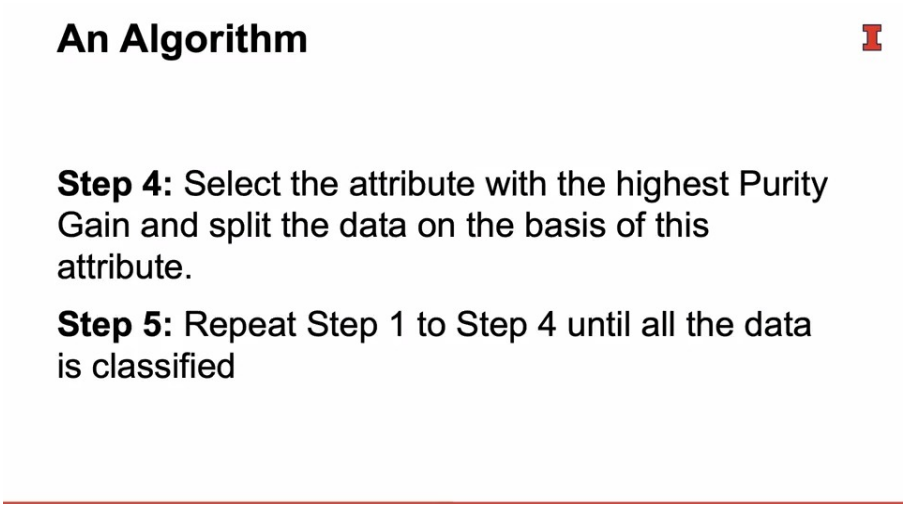
Step 2: Calculate the entropy for splitting using each attribute.

Step 3: Calculate the Purity Gain for splitting using each attribute using the previous formula.

Transcript

So, here's an algorithm, find the entropy of your data without any splits, then take each of the attributes, we can split on the A, you can split on the B, you can split on the C, you have various choices. Calculate the gain in purity you get by splitting on each of the attribute.

An Algorithm (2 of 2) - Slide 33



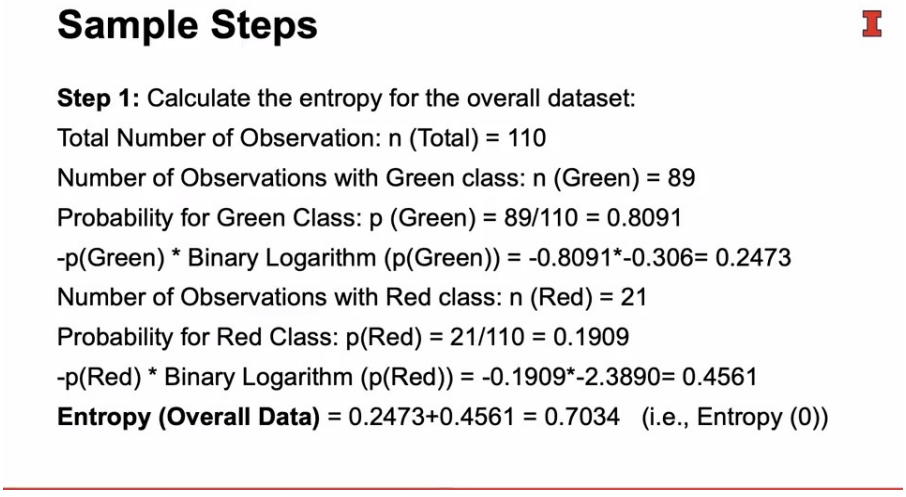
Step 4: Select the attribute with the highest Purity Gain and split the data on the basis of this attribute.

Step 5: Repeat Step 1 to Step 4 until all the data is classified.

Transcript

Choose the attribute which has the highest Purity Gain, split the data, and repeat the procedure. Typically, we can do this under all the data gets classified. You can actually use this on every data point, can get classified, and later on we will see how we can reduce. Because if all the data gets completely classified, you are probably over fitting or your model is too complex. So, you may have to reduce the depth of the tree to get the right size of the tree. Just an aside, this is not the only purity measure possible. In Rattle, this is the measure used and there are other measures which you can use like the Gini index, which R allows. But as far as we're concerned, all we need to know is there is a purity measure, and the one that Rattle uses is called the entropy measure. There are other measures which you can read about in the references given at the end of this lecture.

Sample Steps - Slide 34



Step 1: Calculate the entropy for the overall dataset:

Total Number of Observations: n(Total) = 110

Number of Observations with Green class: n(Green) = 89

Probability for Green Class: p(Green) = 89 ÷ 110 = 0.8091

-p(Green) × Binary Logarithm (p(Green)) = -0.8091 × -0.306 = 0.2473

Number of Observations with Red class: n(Red) = 21

Probability for Red Class: p(Red) = 21 ÷ 110 = 0.1909

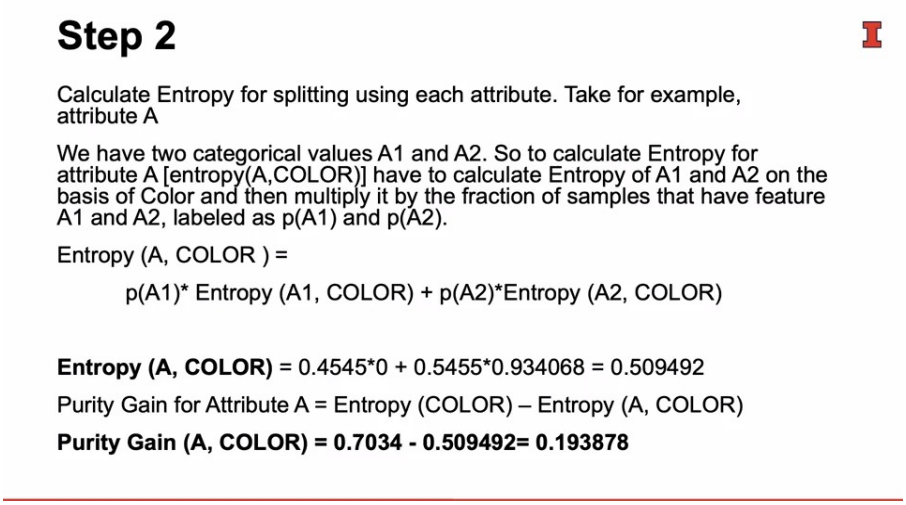
-p(Red) × Binary Logarithm (p(Red)) = -0.1909 × -2.3890 = 0.4561

Entropy (Overall Data) = 0.2473 + 0.4561 = 0.7034 (i.e., Entropy(0))

Transcript

So, let's take this particular example of flowers. You've got 110 observations out of which 89 are green, right? So, the probability of green is 80.91 percent or 0.8091, 89 divided by 110. So, probability of green times the log of the probability of green, as you can see with a negative number attached to it, because a log of a probability will always be negative and you want positive values is 0.8091 times 0.306 which is 0.2473. Number of observations which are red are 21. So, the fraction observations which are red is 0.1909. So, you multiply the probability of red which is 0.1909 with a negative sign attached to it, times log to the base two of minus 2.3890 which is 0.4561. You add up the two and you get the entropy, the data set which is 0.7034, which is high or it's closer to one than to zero. So, what it says is, this is a measure of purity of the data if you don't split.

Step 2 (1 of 3)- Slide 35



Calculate Entropy for splitting using each attribute. Take for example, attribute A

We have two categorical values A1 and A2. So to calculate Entropy for attribute A [entropy(A,

COLOR)] have to calculate Entropy of A1 and A2 on the basis of Color and then multiply it by the fraction of samples that have feature A1 and A2, labeled as p(A1) and p(A2).

Entropy(A, COLOR) = p(A1) × Entropy(A1, COLOR) + p(A2) × Entropy(A2, COLOR)

Entropy(A, COLOR) = 0.4545 × 0 + 0.5455 × 0.934068 = 0.509492

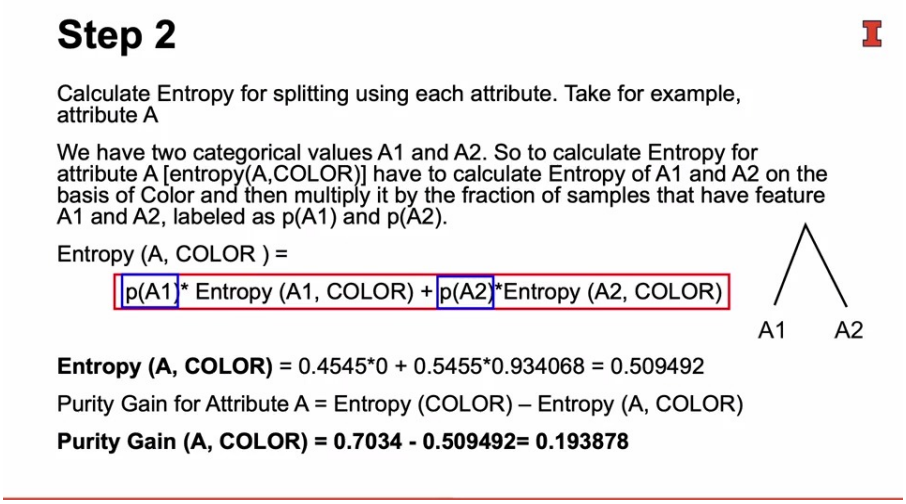
Purity Gain for Attribute A = Entropy (COLOR) - Entropy (A, COLOR)

Purity Gain (A, COLOR) = 0.7034 - 0.509492 = 0.193878

Transcript

Step two. Now, we try to split this data using each of these attributes. So, for the first categorical values A, so we could split the data into the A1's on one side and the A2's on other side. So, we write this as Entropy (A, COLOR). What it means is, I'm trying to compute the entropy or if we split using the feature A, and the basis of calculating the entropy is the color. So, that's the meaning of that function Entropy (A, COLOR). So, we break it into two pieces, A1 and A2. These are all the variables, the flowers which have the value A1 for attribute A, and these are all the flowers which have the value A2 for attribute A. We look at them again. Okay.

Step 2 (2 of 3)- Slide 36

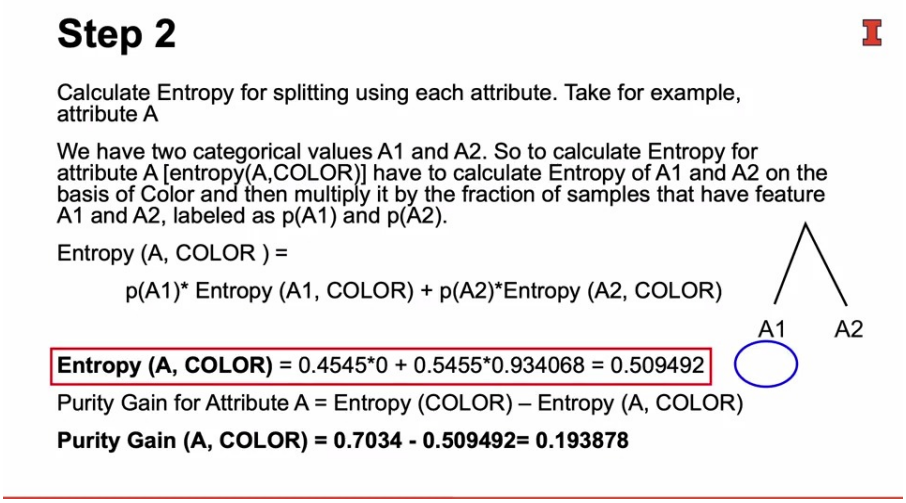


This slide the same as Slide 35 - Step 2 (1 of 3), with a red rectangle on p(A1) × Entropy(A1, Color) + p(A2) × Entropy(A2, Color) and two blue rectangle on p(A1) and p(A2). Next to this formula: A1 connecting to A2.

Transcript

So, this is the measure of the Entropy due to the split. pA1 and pA2 in this formula referred to the fraction of flowers which went to the left and went to the right. These two refer to the Entropy of flowers here and here. So, pA1 and pA2 is what, how many flows went to the left, how many went to the right. For each of these group of flowers we are recomputing their purity measures.

Step 2 (3 of 3)- Slide 37

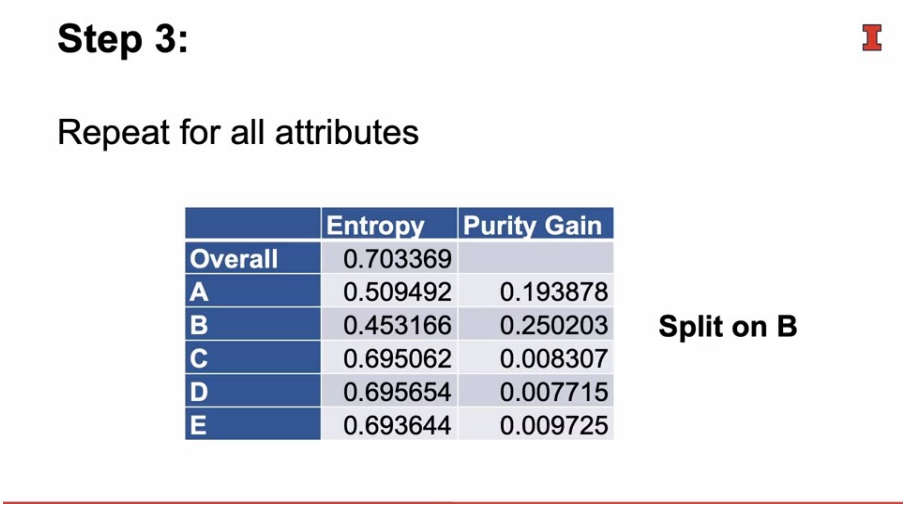


This slide the same as Slide 36 - Step 2 (2 of 3), with a red rectangle on Entropy(A, Color) = 0.4545 × 0 + 0.5455 × 0.934068 = 0.509492 and a blue circle on A1.

Transcript

So, take this example. So, what it says is, if you go on A1, means you go to the left. Your data-set is pure and because your data-set is pure, your Entropy is zero. So, the entropy on this side is zero. If you go to the right, 54.55 percent of the flowers went to the right, and 45.545 went to the left. So, 54.55, is the number of fraction of flowers which went to the right. The Entropy of the colors on the right turns out to be 0.934. That means, probably it is equally split between red and greens. I have the exact values, you can check it out. But how do you get 0.934068, is the entropy when you branch to the right. Which is the probability of green times the log of the probability of green plus the probability of red times the log of the probability of red, within all those flowers which have branched to the right. That gives you the calculation for other flowers which go to the right. If you add it up, this is your new Entropy. So, you purify the data in some sense by splitting on this variable. What is the purity gain? The purity gain you got is the original value, which is 0.7034, and the new value if you split on the variable A. This is said to be the gain due to splitting on A. We can repeat this for each of the features.

Step 3 - Slide 38



	Entropy	Purity Gain
Overall	0.703369	
A	0.509492	0.193878
B	0.453166	0.250203
C	0.695062	0.008307
D	0.695654	0.007715
E	0.693644	0.009725

Repeat for all attributes

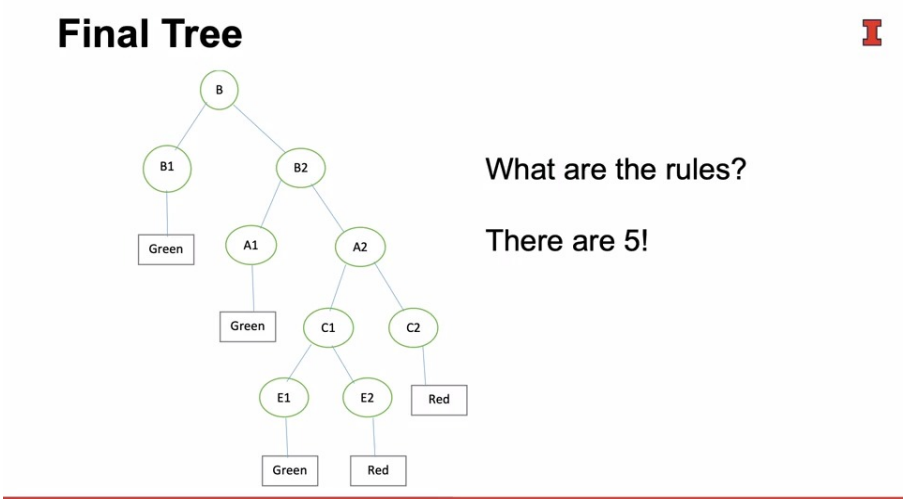
	Entropy	**Purity Gain**
Overall	0.703369	
A	0.509492	0.193878
B	0.453166	0.250203
C	0.695062	0.008307
D	0.695654	0.007715
E	0.693644	0.009725

Next to the table: **Split on B**

Transcript

Here's a table if you do. You can see you could split on A, or B, or C, or D, or E. These gain values is the delta you get, right? So, 0.25, is the gain you get from splitting on the variable B. So, the highest gain we get is splitting on B, and you split on B. So, here's the root node on the right, B1 on the left and B2. We can now apply this algorithm to B1 and B2, and try to see what to split on. There is no rule with saying, of course everything on the left is B1, everything on the right is B2. So, you can't anymore split on B in this particular example, but sometimes you may repeat the split, I'll show an example of that.

Final Tree (1 of 2)- Slide 39



Final Tree

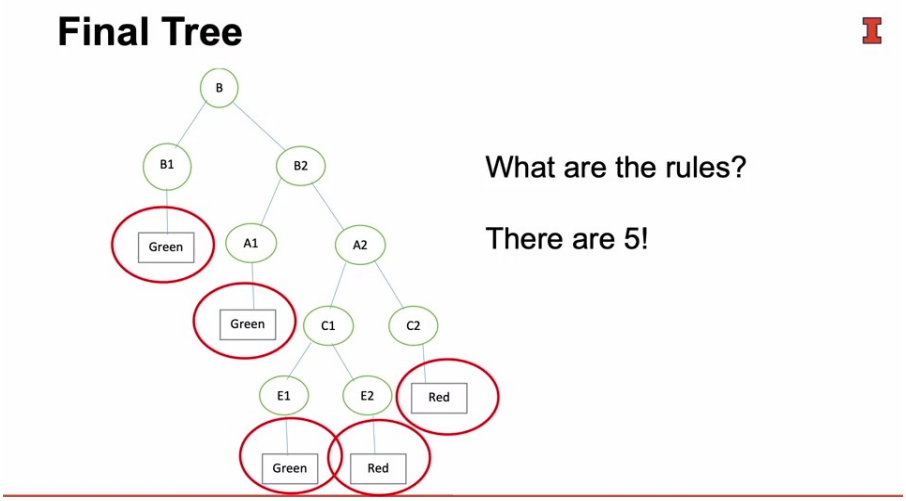
B	B1	Green			
	B2	A1	Green		
		A2	C1	E1	Green
				E2	Red
			C2	Red	

What are the rules? There are 5!

Transcript

This is the final tree. So, you first split on B1 and B2. At the moment you split on B1 you got all the greens. So, if it is B1, everybody loves those flowers. On B2, again we split and the best split was on feature A. If you split on A1, everybody loved it. Go to A2, you still have an impute area. You can split it now. The best split is between C1 and C2, and C2, if you split, you get many of the flowers that people hate, right? Don't like. Then if you split on C1, you still have a mixture of reds and greens to further refine it, we split on E1 and E2. In this particular example, we are getting a perfect classification, right? So, this is called the root.

Final Tree (2 of 2)- Slide 40

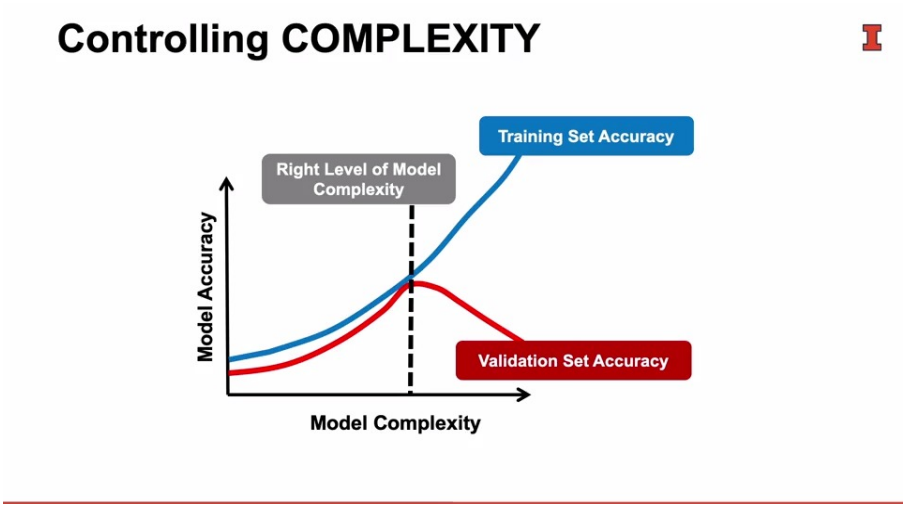


The slide contains the same graph as Slide 39 - Final Tree (1 of 2), and each of the ending nodes are circled.

Transcript

This is called the leaf. This is a leaf. This is a leaf. This is a leaf. This is a leaf. Note that, the leaf nodes are rectangular in shape. Leaf nodes are all pure. So, we have successfully managed to completely classify all the objects into whether they're red or green in this simple example. Sometimes you cannot do that, right? So, coming back to where we really were going, what are the rules? The rules are simple. There are five rules, and you can almost see it, right? If it is B1, everybody loves it. If it is B2 and A1, everybody loves it. If it is B2, A2, and C2, nobody likes it. So, you actually have rules from the data which allows you now to classify any new flower if you want probably into different categories.

Controlling Complexity - Slide 41

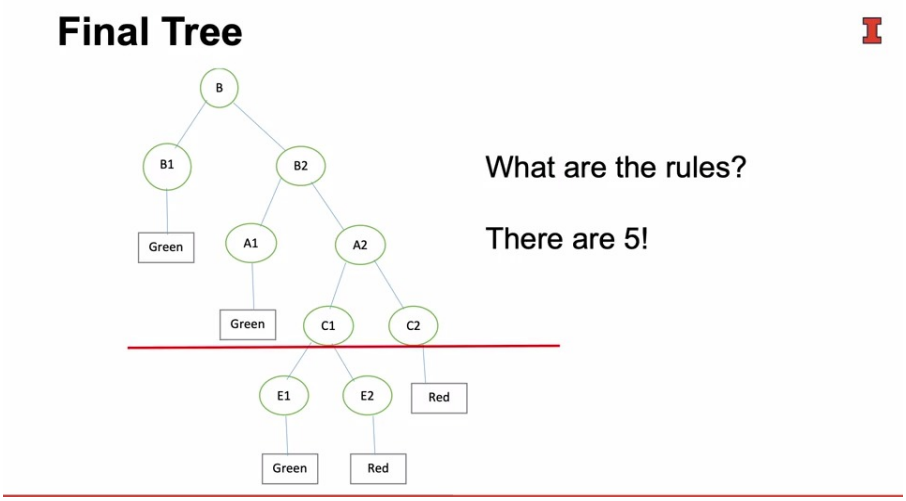


This slide is the same as Slide 20 - Generalize, Don't Memorize! (4 of 4).

Transcript

Question comes, is this too complex a model?

Final Tree - Slide 42

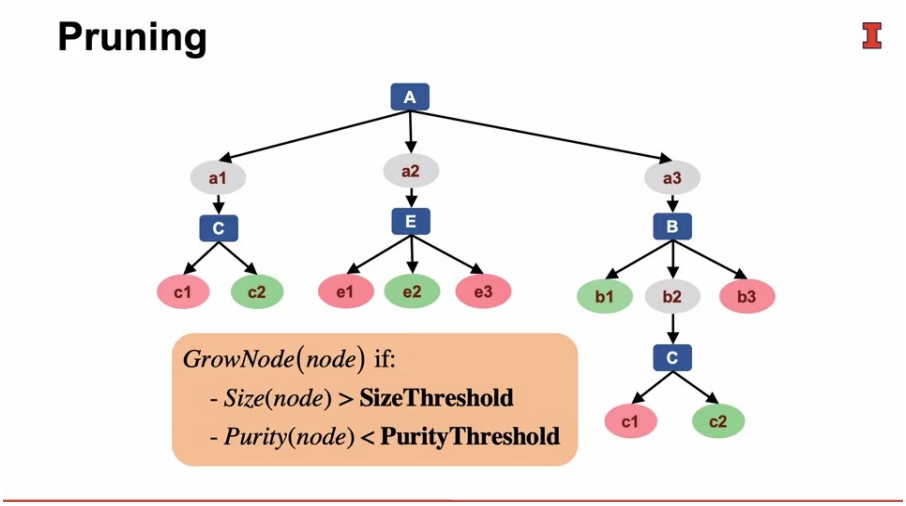


This slide is the same as Slide 39 - Final Tree (1 of 2), with a red line below Green, C1 and C2.

Transcript

Because what I could've done, I could've stopped splitting somewhere there, because it's overfitting the data. So, we will see that, how do you control the complexity? They control the complexity by many methods, but basically by the depth of the trees, by pruning the tree. We say we clear the whole tree maybe and then we keep pruning it till we are satisfied.

Pruning - Slide 43



Pruning Tree

A	a1	C	c1

		c2			
a2	E	e1			
		e2			
		e3			
a3	B	b1			
		b2	C	c1	
		b3		c2	

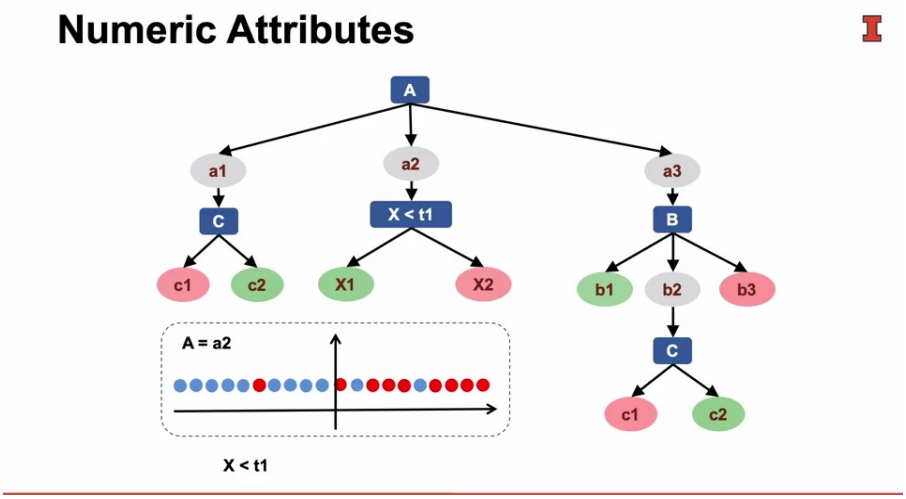
Below the table: *GrowNode(node)* if:

- *Size(node)* > **SizeThreshold**
- *Purity(node)* < **PurityThreshold**

Transcript

This is an example of overfitting. You manage to perfectly classify everything. Well, here are some rules you can use for pruning. One, you can grow a node, that means you can split on a node if there are sufficient objects in that node, right? That's called a SizeThreshold. So, split a node only if there are 15 things in that node, right? Second, that split only when the purity is not, you're not happy with that, right? If it does a purity of 90 percent, you don't split beyond that. So, here are sudden rules you can use to cut the size of the tree. In this example, we have pruned the previous tree, into a smaller tree. One last thing. All the examples we have given so far is on categorical variables. What happens if you have numerical values, okay? So, take this example, initially we are split into A1, A2, A3,

Numeric Attributes - Slide 44

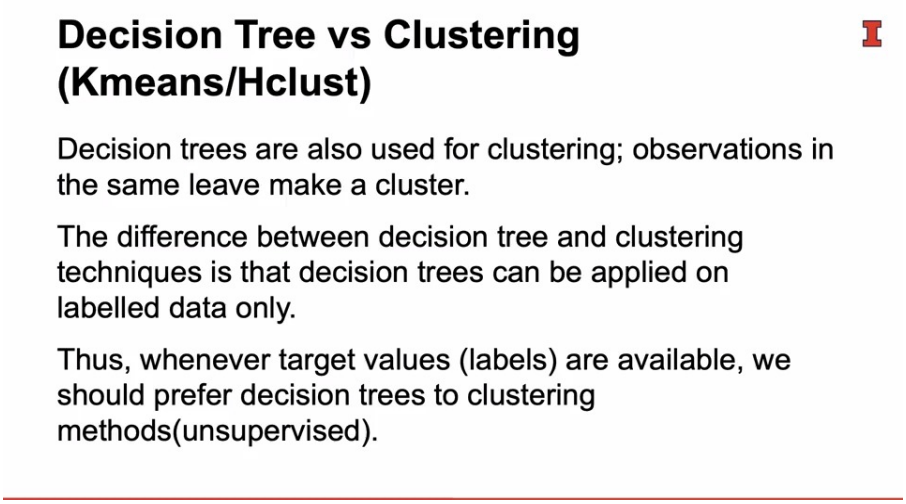


This slide is the same as Slide 43 - Pruning, with a coordinate splitting data. On the left side, X < t1, A = a2. There are nine blue points and one red point on the left side of the y-axis, and eight red points and two points on the right side of the y-axis.

Transcript

and when it came to A2, there was a numerical attribute called X. So, what we could do is split, but here the splitting criteria is you pick a number. If X is smaller than t1, you split to the left. If X is greater than t1, you split to the right. So, here you go, right? So, it's as simple as that. So, we could use numerical variables or you could use categorical variables. Numerical variables, you find a point to split in, and split it into two. One thing that may be running through your minds, is should we allow only Two-Way Splits or Multi-Way Splits. A Multi-Way Splits can be captured with Two-Way Splits. So, if that's confusing to you, always work with splitting into two doesn't matter.

Decision Tree vs Clustering (Kmeans/Hclust) - Slide 45



Decision trees are also used for clustering; observations in the same leave make a cluster.

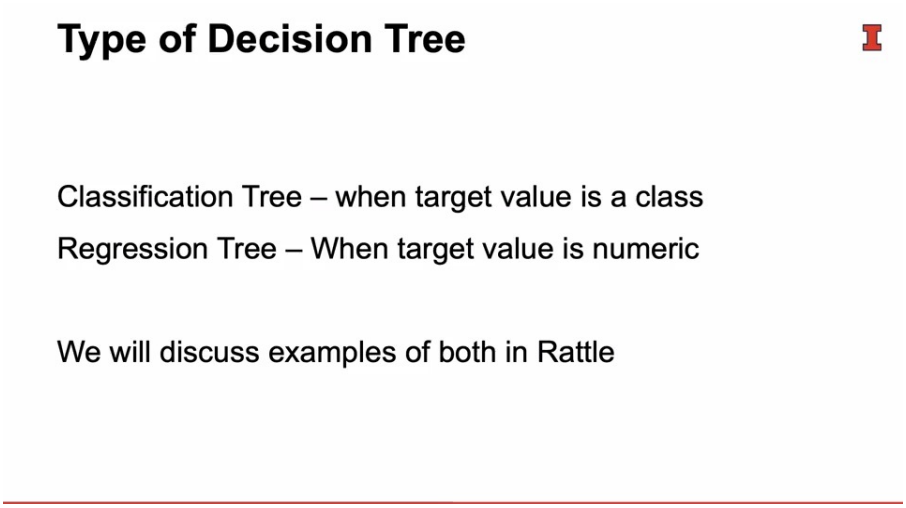
The difference between decision tree and clustering techniques is that decision trees can be applied on labelled data only.

Thus, whenever target values (labels) are available, we should prefer decision trees to clustering methods (unsupervised).

Transcript

So, if two branches come from a node, that's called the Two-Way Split. If many branches come from a node, that's called a Multi-Way Split. There are other methods we have already seen, right? K-means right. Hierarchical, the clustering algorithms for joining similar items together. That is where we used a decision tree. All right? So, what I'm trying to say here is, the leaf nodes of your tree may have many observations and we can treat those as a cluster. So, how is that different from the clustering techniques that we have already seen, okay? The answer is very simple. The difference between a decision tree and the clustering techniques that you saw, is that the decision tree needs labeled data. It needs to know what category it belongs to, okay? So, whenever the target values or labels are available, we should use decision trees rather than the other methods we saw. Which belong to what we call an unsupervised learning, okay. This is a very simple point, but please do remember that and you could probably use both. It doesn't matter, but decision trees up preferable, and there's one more reason at the end of today's module we will see.

Type of Decision Tree - Slide 46



Classification Tree - when target value is a class

Regression Tree - When target value is numeric

We will discuss examples of both in Rattle

Transcript

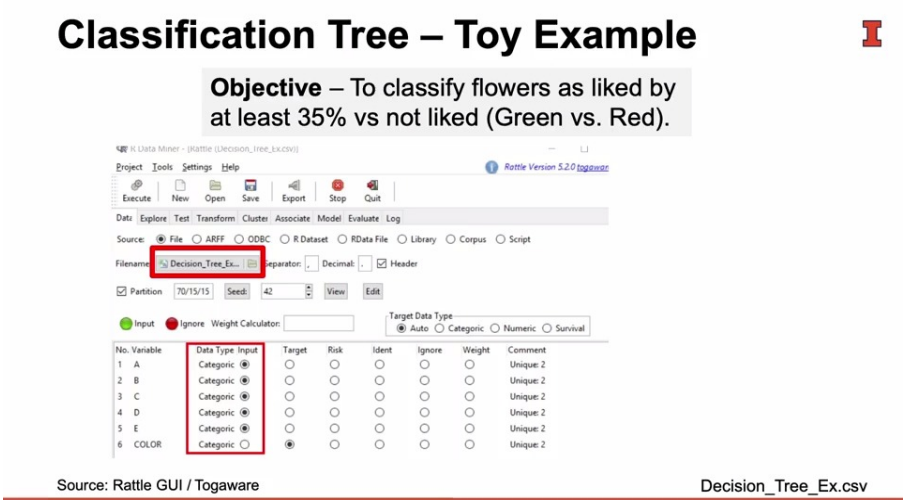
So, just to complete this, we used a decision tree to classify things into red and green. You wouldn't believe it, right. The first time I had learned this, that decision trees can also be used for regression. That means, you can actually predict the value when the target value is numeric for example pressure, or temperature, or weight, or things like that. How do you do that? Basically, you create a tree and you have leaf nodes. The leaf nodes could have one or many values. We take the average value in the leaf node as a prediction. So, basically you drop observation down the tree, it'll hit a leaf node and the leaf node there will be lots of data points. We take the average of the target value of these data points and say your value is equal to the mean of that. We will see an example of that going forward. Okay.

Lesson 2-5 Classification Tree Example

Lesson 2-5.1 Classification Tree Example

Media Player for Video

Classification Tree - Toy Example - Slide 47



Objective – To classify flowers as liked by at least 35% vs not liked (Green vs Red). The slide contains a walkthrough of running classification tree in Rattle.

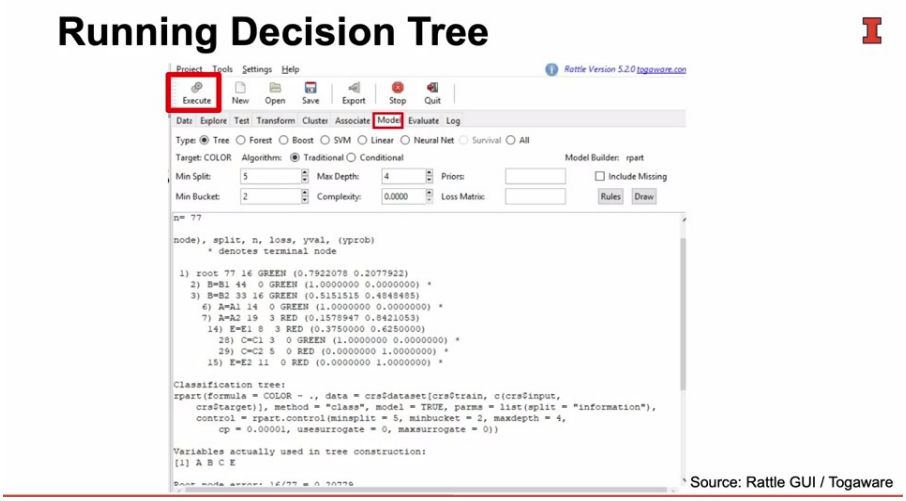
1. In the Data tab, select Decision_Tree_Example in the Filename and click Partition, set 70/15/15.
2. Select A, B, C, D, E as Input and COLOR as Target.

Source: Rattle GUI / Togaware [2]

Transcript

So, Rattle allows you to do this automatically, and I'm going to just go over the features in Rattle. I might not go through the software because it's very simple, but let's understand what the outputs are. So, the data file I'm using, and you're used to it by now, is called Decision Tree Ex. It's provided to you in your folder, your data folder. This is the same data we used in the flower example just now. So, it has got five inputs, which are categoric in nature and we are trying to figure out whether it should be green or red. One more thing I would like you to notice is the tree we're going to get is not going to be the tree we just saw. The reason is, we are partitioning the data into three sets and we're training it on 70 percent, and we're looking at the validation set of 15 percent. You could change it. We're using a seed of 42, but if you change it, you won't get the same result. So, remember, the difference in result is because my sampling is different.

Running Decision Tree (1 of 4) - Slide 48



The slide contains a walkthrough of running decision tree in Rattle.

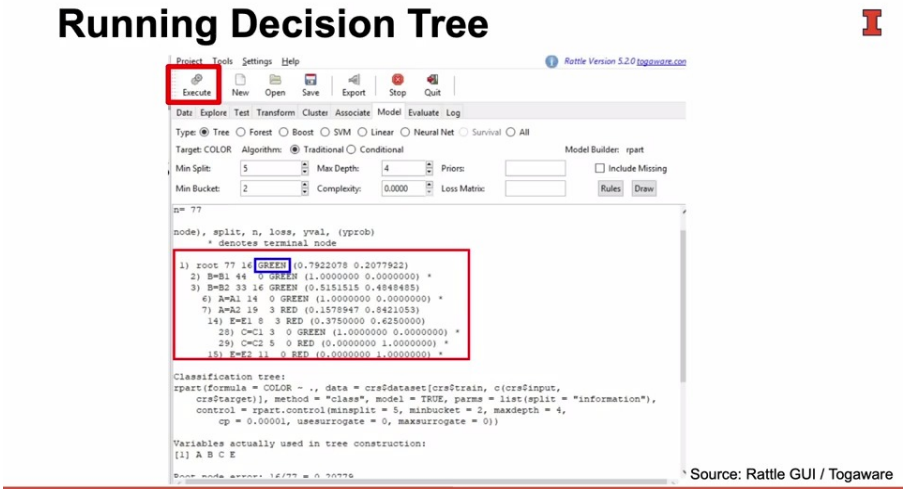
1. Click Execute, and go to Model Tab.
2. Select Tree and click Execute.

Source: Rattle GUI / Togaware

Transcript

So, you execute, and let me see what we did. You have to go to model and in the model, you would have to execute. So, when you went to model and you chose tree and hit execute, it produces this output. N is 77 because only 75 percent of the data got into that, and it produces an output. So, what does this output have? It has got various numbers. Let's quickly understand what they have.

Running Decision Tree (2 of 4) - Slide 49



The slide is the same as Slide 48 - Running Decision Tree (1 of 4). The output is in the table below.

Green is circled.

N = 77.

The output of running decision tree

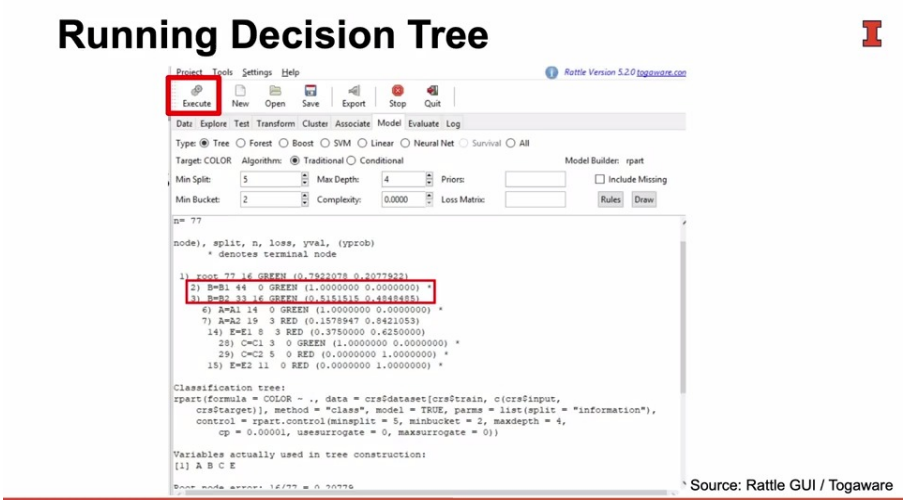
Node	Split	n	loss	yval	Yprob	* denotes the terminal node
1	Root	77	16	Green	(0.7922078 0.2077922)	
2	B = B1	44	0	Green	(1.0000000 0.0000000)	*
3	B = B2	33	16	Green	(0.5151515 0.4848485)	
6	A = A1	14	0	Green	(1.0000000 0.0000000)	*
7	A = A2	19	3	Red	(0.1578947 0.8421053)	
14	E = E1	8	3	Red	(0.3750000 0.6250000)	
28	C = C1	3	3	Green	(1.0000000 0.0000000)	*
29	C = C2	5	5	Red	(0.0000000 1.0000000)	*
15	E = E2	11	0	Red	(0.0000000 1.0000000)	*

Source: Rattle GUI / Togaware

Transcript

So, first of all, it's got some number and you see these numbers, they're not sequential because this is the best split in some way to remember. So, first, there is a root. The first line is reading the root, 77 data points, and the majority category is green. So, if you categorize this as green, you'll be right 79 percent, you'll be wrong 20 percent. You will misclassify 16 of the data points. That's what it means. Then the next best split that is found is B.

Running Decision Tree (3 of 4) - Slide 50



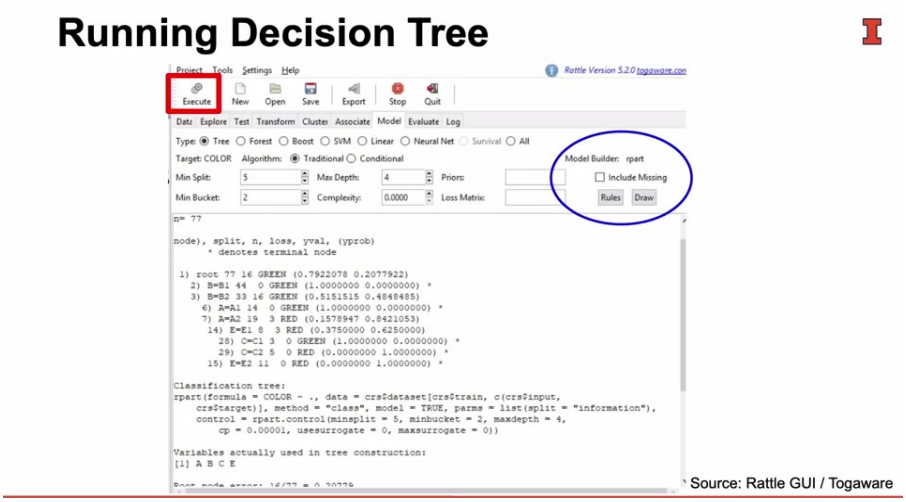
This slide is the same as Slide 49 - Running Decision Tree (2 of 4) with the red circle on the second and third row of output.

Source: Rattle GUI / Togaware

Transcript

It can be BB1, BB2. You remember, B1 is completely green. So, you can see that if you split it as green and it's got a star against it because it stops there, that's its leaf node which is saying, "I'm stopping." It's a pure node and you're labeling it as green. There are 44 points on that, you can see that, there are 44 points on that. There is no loss which means the purity is perfect. If you go B2, B1 B2, you will see that the majority there is green. So, if you classify it as green, 51 percent will be right, 48 percent will be wrong. That's what it is saying out here, and 16 will be classified wrongly and 17 would be classified rightly. That's all it means. The rest of it is just information as far as we're concerned.

Running Decision Tree (4 of 4) - Slide 51



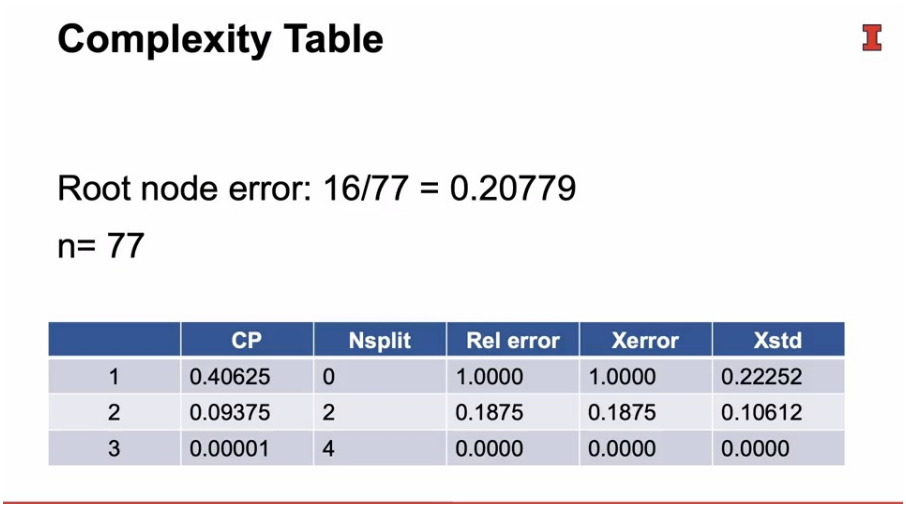
The slide is the same as Slide 50 - Running Decision Tree (3 of 4) with a blue circle on Rattle Interface, Model Builder: rpart, include Missing, Rules and Draw.

Source: Rattle GUI / Togaware

Transcript

So, once it runs the model, we can ask it to draw the tree, we can ask it to spit out the rule letters.

Complexity Table - Slide 52



Root node error: 16 ÷ 77 = 0.20779

n = 77

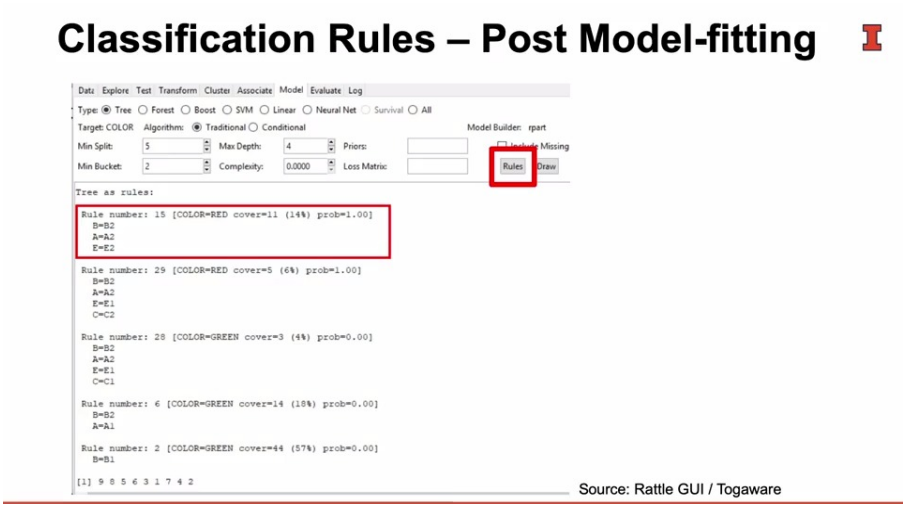
Complexity Table

	CP	Nsplit	Rel error	Xerror	Xstd
1	0.40625	0	1.0000	1.0000	0.22252
2	0.09375	2	0.1875	0.1875	0.10612
3	0.00001	4	0.0000	0.0000	0.0000

Transcript

If you do the evaluation part, it will give you the error. It says, "The root node has an error of 20 percent." Because 16 out of 77 is misclassified, and it gives you some data here. Let's try to understand what this data means. So, these are the splits it's making, and as it makes the splits, the error is coming down. So, what this means is, as I make the tree larger and larger the relative error. So, initially, the error is a 100 percent and with the first split, it has reduced to 18 percent of what it was. With the next level of the tree, the error is zero. So, this is measuring relative to the original error you had. This needs some explanation. This is called a cross-validation error. What it's really doing it's dividing data into pieces. It's fitting the tree in say, nine pieces and testing it on the 10th piece. So, in this case, it is actually testing it I think, on the validation set and splitting out the error. So, this is the actual cross-validation error that it is splitting out. There is a randomness to it because every time it does this cross-validation, you'll get a different error. So, you can find the average error on the validation set and you can find it's standard deviation. If you notice, the cross-validation error in this case is also reducing the same way as the relative error which is not true. So, what we really are going to look at is the cross-validation error.

Classification Rules - Post Model-fitting (1 of 2) - Slide 53



The slide contains a walkthrough of classification rules in Rattle.

Click the Rules after you run the model. The output is below.

Tree as rules:

Rule number: 15 [COLOR=RED cover=11 (14%) prob=1.00]

B=B2

A=A2

E=E2

This is circled.

Rule number: 29 [COLOR=RED cover=5 (6%) prob=1.00]

B=B2

A=A2

E=E1

C=C2

Rule number: 28 [COLOR=GREEN cover=3 (4%) prob=0.00]

B=B2

A=A2

E=E1

C=C1

Rule number: 6 [COLOR=GREEN cover=14 (18%) prob=0.00]

B=B2

A=A1

Rule number: 2 [COLOR=GREEN cover=44 (57%) prob=0.00]

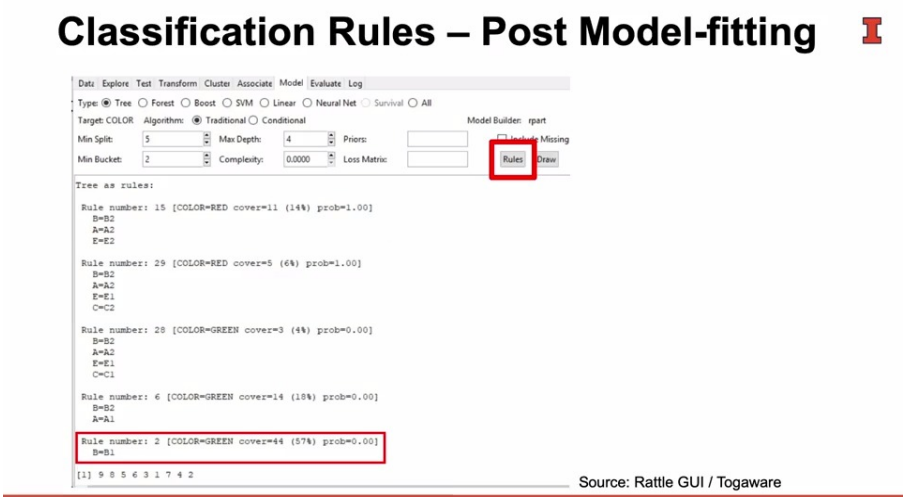
B=B1

Source: Rattle GUI / Togaware

Transcript

What are the rules? So, you can ask Rattle, you click on it after you run the model and you can look at the rules and it'll print out all the rules. These rules are different from the ones you got earlier but look at the rule 1. It says if B is B2, A is A2, E is E2, then it's red, and the probability is one it is red. Another rule says if B is B2, A is A2, E is E1, C is C2, it's red, so on.

Classification Rules - Post Model-fitting (2 of 2) - Slide 54



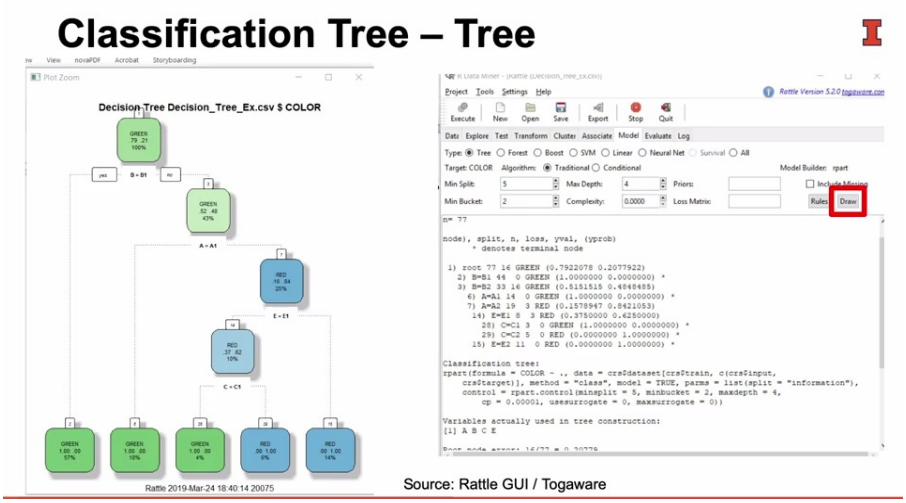
The slide is the same as Slide 53 - Classification Rules - Post Model-fitting (1 of 2) with Rule number 2 circled.

Source: Rattle GUI / Togaware

Transcript

So, if B is B1, which we know, it's green, and we know how many points it covers and what's the probability of misclassification.

Classification Tree - Tree (1 of 2) - Slide 55



The slide contains a decision tree and screenshot of the Rattle which is the same as Slide 48 Running Decision Tree (1 of 4).

Click Draw and you will get the plot in R.

Decision Tree Decision_Tree_Ex.csv $ COLOR.

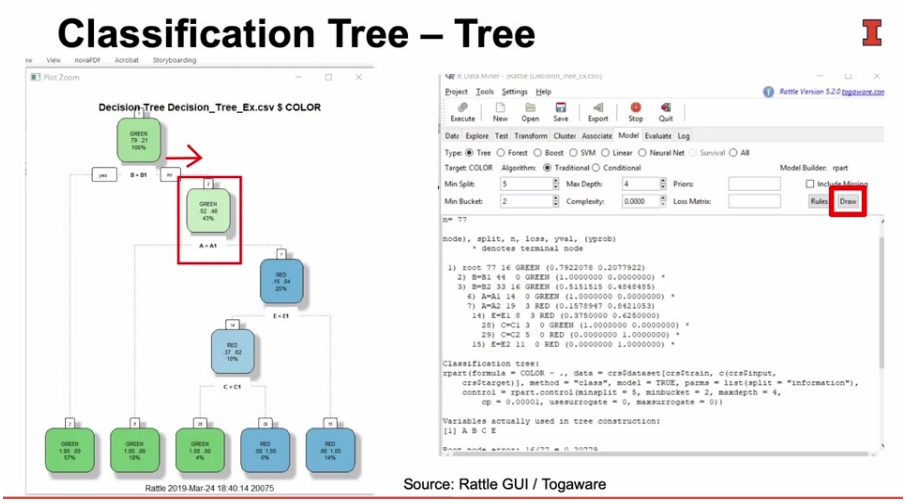
<table>
<tr><td colspan="4">B=B1 GREEN 0.79 0.21 100%</td><td>GREEN 1.00 0.00 57%</td></tr>
<tr><td rowspan="4">B!=B1</td><td colspan="3">A=A1 GREEN 0.52 0.48 43%</td><td>GREEN 1.00 0.00 18%</td></tr>
<tr><td rowspan="3">A!=A1</td><td rowspan="2">E=E1 RED 0.16 0.84 25%</td><td>C=C1 RED 0.37 0.62 10%</td><td>GREEN 1.00 0.00 4%</td></tr>
<tr><td>C!=C1</td><td>RED 0.00 1.00 6%</td></tr>
<tr><td colspan="2">E!=E1</td><td>RED 0.00 1.00 14%</td></tr>
</table>

Source: Rattle GUI / Togaware

Transcript

So, let us look at the tree. So, this is the tree. Slightly different, but it also draws a fairly neat tree up here. It corresponds to executing the command draw all of them, and you can see. So, if you answer "yes" you go to the left. If you answer "no" you go to the right, that's a convention. So, B equal to B1? Yes. You go this way, and you get green.

Classification Tree - Tree (2 of 2) - Slide 56



The slide is the same as Slide 55 - Classification Tree - Tree (1 of 2) with element GREEN 0.52 0.48 43% circled. Arrow points right to it from GREEN 0.79 0.21 100% B = B1 NO.

Source: Rattle GUI / Togaware

Transcript

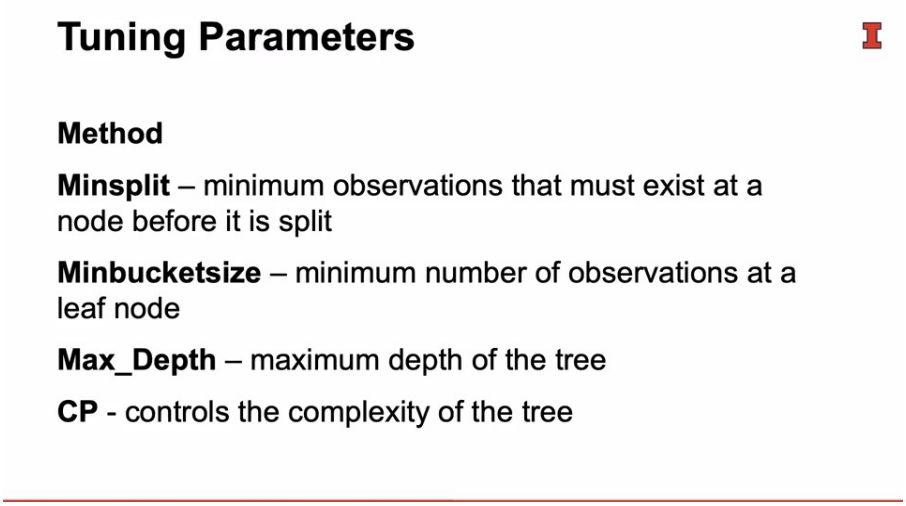
If you go "no" you go to the right. Then you ask, is A equal A1? Yes, A is equal A1, no. That's all the way the tree is drawn.

Lesson 2-6 Regression Tree Example

Lesson 2-6.1 Regression Tree Example

Media Player for Video

Tuning Parameters - Slide 57



Method

Minsplit - minimum observations that must exist at a node before it is split

Minbucketsize - minimum number of observations at a leaf node

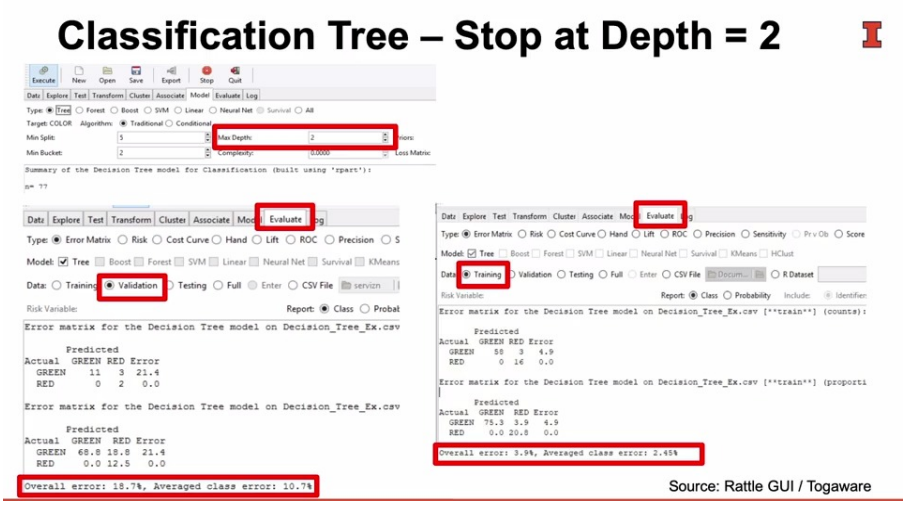
Max_Depth - maximum depth of the tree

CP - controls the complexity of the tree

Transcript

So, obviously, like in regression or like in logistics regression you can tune the parameters. So, what are the parameters based on which we can tune the performance? First of all, the method used for calculating the split. Now, that's not available in Rattle but you can use information gain or Gini if you go into R. We won't be studying it right now. Second, you can tell Rattle, how many observations must exist at a node before it splits. So, Minsplit is, under there are sufficient observations, don't split it, right? Third, it says if you split it too finely, there may be too few observations at the leaf node. So, you can set the minimum number of observations you need at a leaf node maybe 5, 6, or 7, right? The Max_Depth is the maximum depth of the tree which we will discuss further down the slide. Finally, there is a variable called CP, which controls the complexity of the tree. The complexity of a tree is the improvement in the error relative to the number of nodes we have added to it. So, we will see that in a second, what is CP. So, CP controls for the fact, it's how much benefit you're getting, relative to how many extra nodes of the depth you're adding. There are other ways of tuning, if you have very few observations of a particular category. You can actually rate it, if you have some prior information on the relative population sizes, but your data doesn't conform to the relative numbers in the population you can correct for it. So, those are certain refinements one can make to tune the creation of the tree.

Classification Tree - Stop at Depth = 2 - Slide 58



The slide contains a walkthrough of the classification tree in Rattle.

1. In the Model tab, set Max Depth as 2.
2. After executing it, go to Evaluate tab and select Tree Mode, Validation and Report Class. The output of validation data: overall error is 18.7%, Averaged class error is 10.7%.
3. In the Evaluate tab, select Data Training.

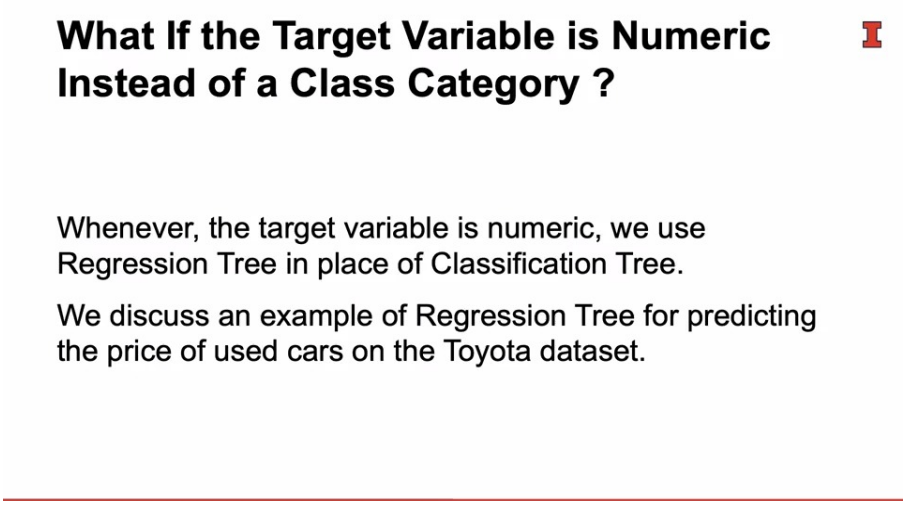
The output of training data: Overall error is 3.9%, Averaged class error is 2.45%.

Source: Rattle GUI / Togaware

Transcript

Okay. So, here's an example. Just to show you this example, what we have done here is we have stopped at a depth of two. So, we said, okay, don't go beyond that. We evaluated it, okay? As you can see, the average error is 10.7 percent, there is a 21 percent error on the greens, and there is no error on the red, okay? So, mean class error is about 10 percent. You look at the, it's on the validation set, on the trading set, also it's much better of course, right? On the training set, your average error is just 2.45 percent, okay? Which gives you the difference between one and the other.

What If the Target Variable is Numeric Instead of a Class Category? - Slide 59



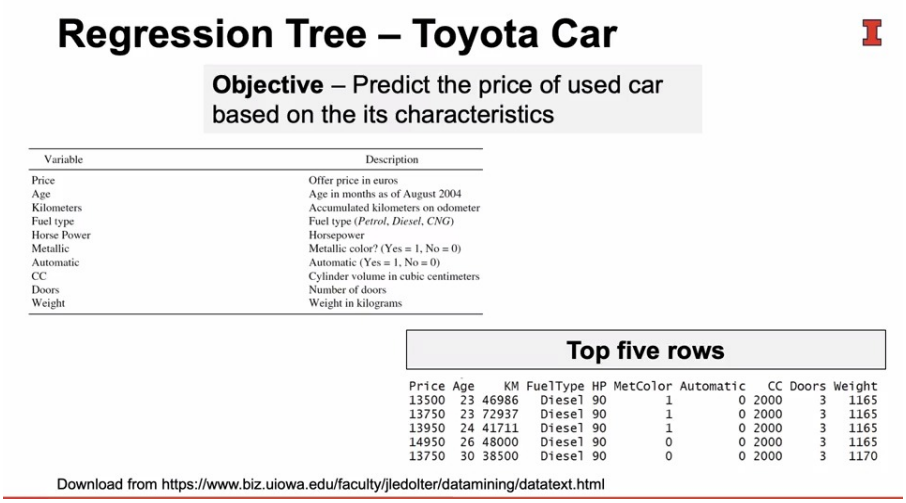
Whenever, the target variable is numeric, we use Regression Tree in place of Classification Tree.

We discuss an example of Regression Tree for predicting the price of used cars on the Toyota dataset.

Transcript

You may like this better, this is a simpler model with just two levels to it, okay? Just for sake of completeness, let us run a model where the target variable is a numeric value instead of a class, right? So, whenever the target value is numeric, we use regression tree. So, that's why these are called CART, Classification and Regression Trees (CART). Now, Rattle does it automatically for you, but we will now use it in this dataset we have studied in the past, which would be the Toyota dataset, and we will use it to predict the price of car.

Regression Tree - Toyota Car - Slide 60



Regression Tree – Toyota Car

Objective – Predict the price of used car based on the its characteristics

Variable	Description
Price	Offer price in euros
Age	Age in months as of August 2004
Kilometers	Accumulated kilometers on odometer
Fuel type	Fuel type (*Petrol, Diesel, CNG*)
Horse Power	Horsepower
Metallic	Metallic color? (Yes = 1, No = 0)
Automatic	Automatic (Yes = 1, No = 0)
CC	Cylinder volume in cubic centimeters
Doors	Number of doors
Weight	Weight in kilograms

Top five rows

Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
13500	23	46986	Diesel	90	1	0	2000	3	1165
13750	23	72937	Diesel	90	1	0	2000	3	1165
13950	24	41711	Diesel	90	1	0	2000	3	1165
14950	26	48000	Diesel	90	0	0	2000	3	1165
13750	30	38500	Diesel	90	0	0	2000	3	1170

Download from https://www.biz.uiowa.edu/faculty/jledolter/datamining/datatext.html

The slide contains the description of the Toyota dataset. Regression Tree – Toyota Car. Objective – Predict the price of a used car based on its characteristics. The description of variables and the top five rows of the dataset are in the table below.

Variables

Variable	Description
Price	Offer price in euros
Age	Age in months as of August 2004
Kilometers	Accumulated kilometers on odometer
Fuel type	Fuel type(Petrol, Diesel, CNG)
Horse Power	Horsepower
Metallic	Metallic color(Yes=1, No=0)
Automatic	Automatic(Yes=1, No=0)
CC	Cylinder volume in cubic centimeters
Doors	Number of doors
Weight	Weight in kilograms

Top five rows

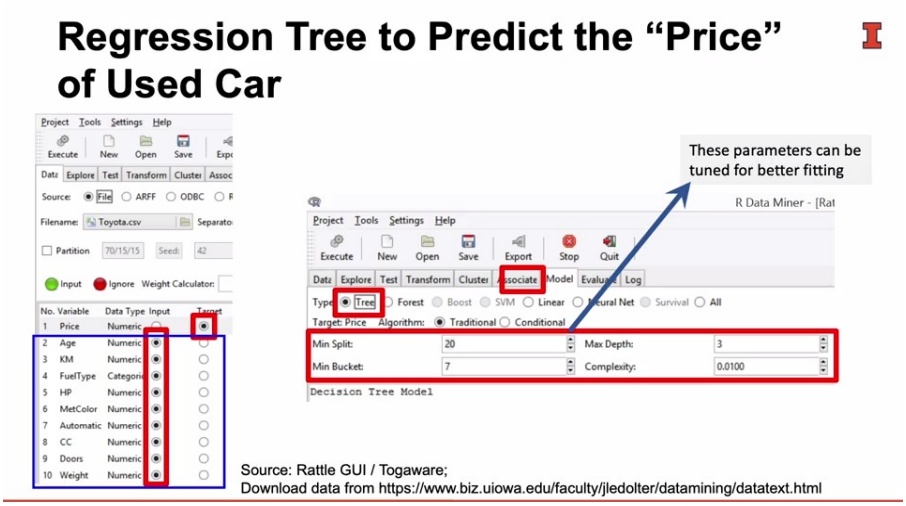
Price	Age	KM	FuelType	HP	MetColor	automatic	CC	Doors	Weight
13500	23	46986	Diesel	90	1	0	2000	3	1165
13750	23	72937	Diesel	90	1	0	2000	3	1165
13950	24	41711	Diesel	90	1	0	2000	3	1165
14950	26	48000	Diesel	90	0	0	2000	3	1165
13750	30	28500	Diesel	90	0	0	2000	3	1170

Download from: Johannes Ledolter Data Set [3]

Transcript

So, you remember this dataset I'm sure. That we are trying to predict price based on age, kilometers, horsepower, the color of the paint, and if it's automatic or not, right? Here are the top five rows of this data which you should remember, you would remember I'm sure, right?

Regression Tree to Predict the "Price" of Used Car - Slide 61



The slide contains a walkthrough of running regression tree in Rattle.

1. In the Data tab, select Toyota.csv file.
2. Select Price as Target, select Age, KM, FuelType, HP, MetColor, Automatic, CC, Doors, Weight as Input.
3. After you click execute, go to Model Tab. Select Tree type. The minimum split is 20, minimum bucket size is seven, maximum depth is three, complexity is 0.01. These parameters can be tuned for better fitting.

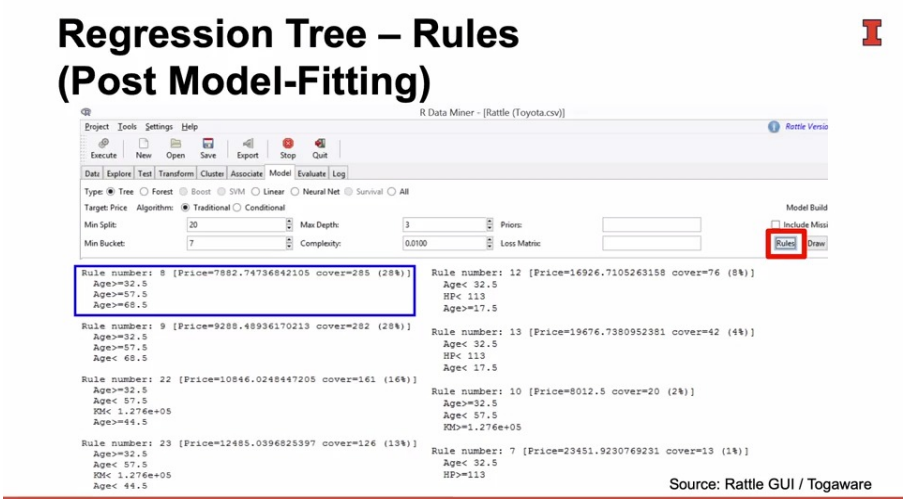
Source: Rattle GUI / Togaware

Download from: https://www.biz.uiowa.edu/faculty/jledolter/datamining/datatext.html

Transcript

So, what we do is just read the data, and you can see here, right? The price becomes the target variable, which is in Rattle. These are the variables, we take all the variables and throw it into the mix. We can tune this later, but at this moment I have just left these values on patch. Minimum split is 20, minimum bucket size is seven, maximum depth is three.

Regression Tree - Rules (Post Model-Fitting) (1 of 4) - Slide 62



The slide contains the output of Rattle.

Click Rules in the Model tab.

The circled rule is below.

Rule number: 8 (Price=7882.74736842105 cover=285 (28%))

Age >= 32.5

Age >= 57.5

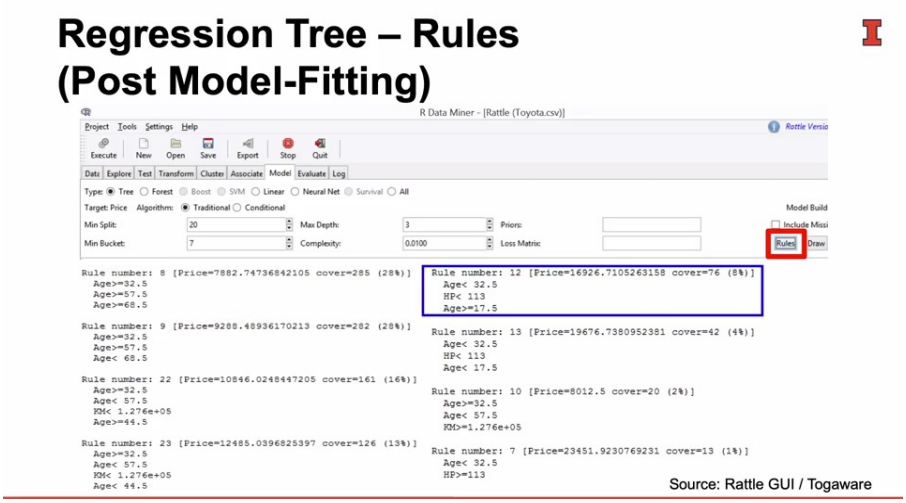
Age >= 68.5

Source: Rattle GUI / Togaware

Transcript

When we are creating a regression tree, the branching criterion is reduction in variance. So, basically it looks at the values once we split, and sees the dispersion from the mean, and it tries to minimize the weighting. So, once we fit it, we get a bunch of rules. So, what does this rule say? Look at rule number eight, if age is more than 32.5, and age is more than 57.5. So, basically its age is more than 68.5, right? Then the price is 7882.

Regression Tree - Rules (Post Model-Fitting) (2 of 4) - Slide 63



The slide is the same as Slide 62 Regression Tree - Rules (Post Model-Fitting) (1 of 4) with a different rule circled.

Rule number: 12 (Price=16926.7105263158 cover=76 (8%))

Age < 32.5

HP < 113

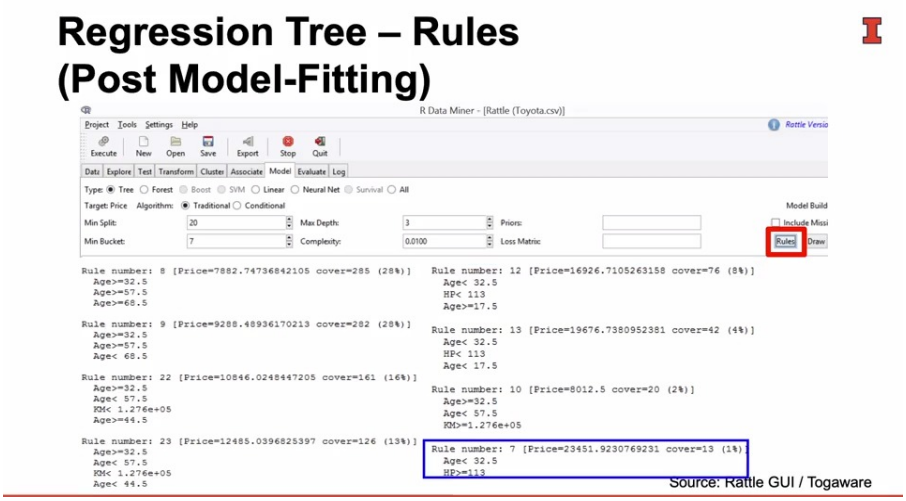
Age >= 17.5

Source: Rattle GUI / Togaware

Transcript

This is a more instructive rule. If the age is more than 17.5 and less than 32.5, and the horsepower is less than 113, the price is 16,926.

Regression Tree - Rules (Post Model-Fitting) (3 of 4) - Slide 64



The slide is the same as Slide 63 - Regression Tree - Rules (Post Model-Fitting) (2 of 4) with different rule circled.

Rule number: 7 (Price=23451.9230769231 cover=13 (1%))

Age < 32.5

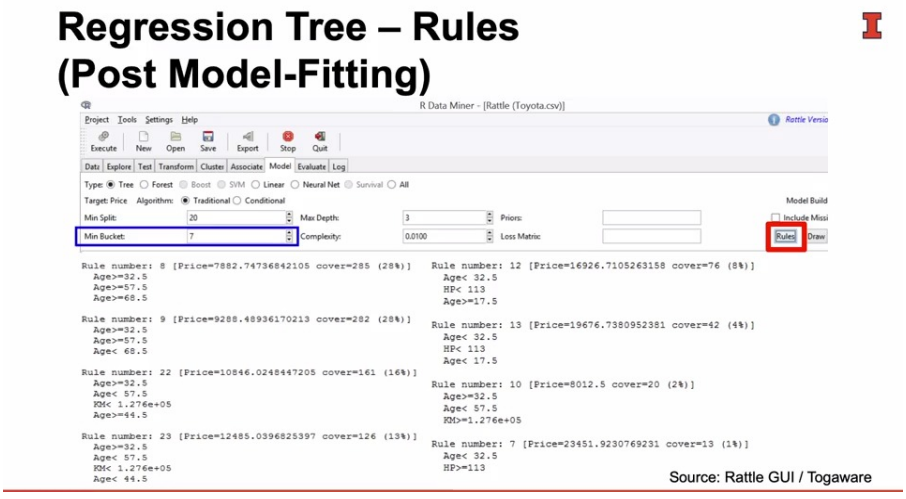
HP >= 113

Source: Rattle GUI / Togaware

Transcript

So, you can probably study this rule number seven. If the age is less than 32.5 and the horsepower is greater than 113, the price is 23,451. So, remember what it does is if it's a tree and the leaf node has a number of observations,

Regression Tree - Rules (Post Model-Fitting) (4 of 4) - Slide 65



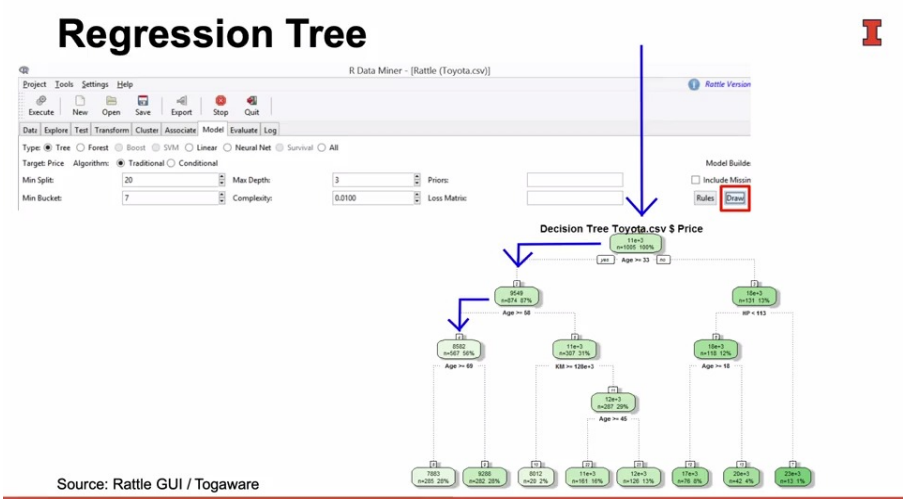
The slide is the same as Slide 64 - Regression Tree - Rules (Post Model-Fitting) (3 of 4) with minimum Bucket as 7 circled.

Source: Rattle GUI / Togaware

Transcript

and we have a minimum bucket size, so we'll at least have seven observations there. We take the average price of the cars in that bucket, and that becomes the predicted value of your car.

Regression Tree - Slide 66



The slide contains a walkthrough of plotting in Rattle.

Click Draw and get Decision Tree Toyota.csv $ Price.

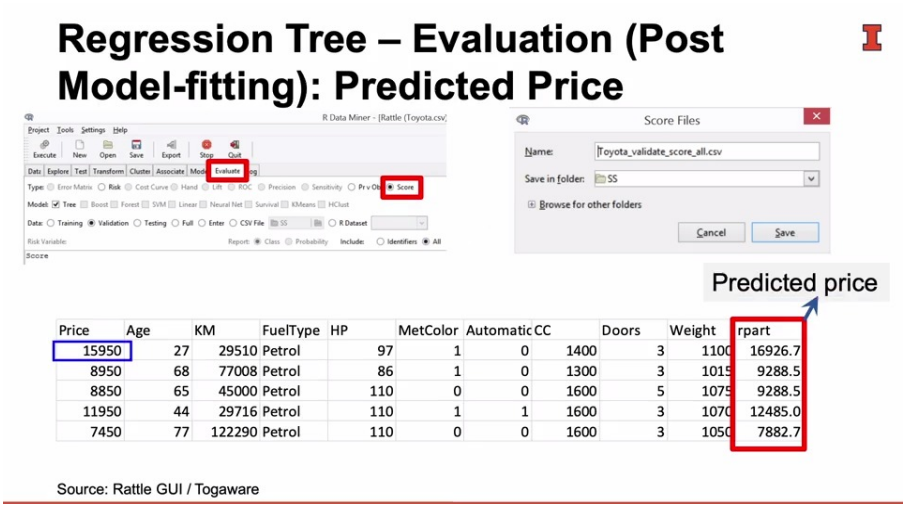
An arrow points down to the top point. If Age >= 33, an arrow points down to the next decision whether Age is greater or equal than 58. If yes, an arrow points down to the next decision whether Age is greater or equal than 69.

Source: Rattle GUI / Togaware

Transcript

How does it do? Well, here's your tree. First of all, it's a nice-looking tree. Not too many rules, right? Very different from the regression models we got in the past. Okay, but it's in the form of a tree. So, you just drop a new card here, and you put a card here and say age is greater than 33, you go left, come here, it says if age is greater than 58, and if it is yes, and you say if age is greater than 68 and you say yes, and say that's the price, 7,883. So, you can actually drop a card through this tree and see where it lands, and that becomes the price of your car.

Regression Tree - Evaluation (Post Model-fitting): Predicted Price (1 of 4) - Slide 67



The slide contains a walkthrough of exporting the predicted price in Rattle.

1. In the Evaluate tab, select score, select All in Includes.
2. Save the file in the popup window.

The review of the file is in the table below. The price of 15950 and the column of rpart are circled.

The review of the field

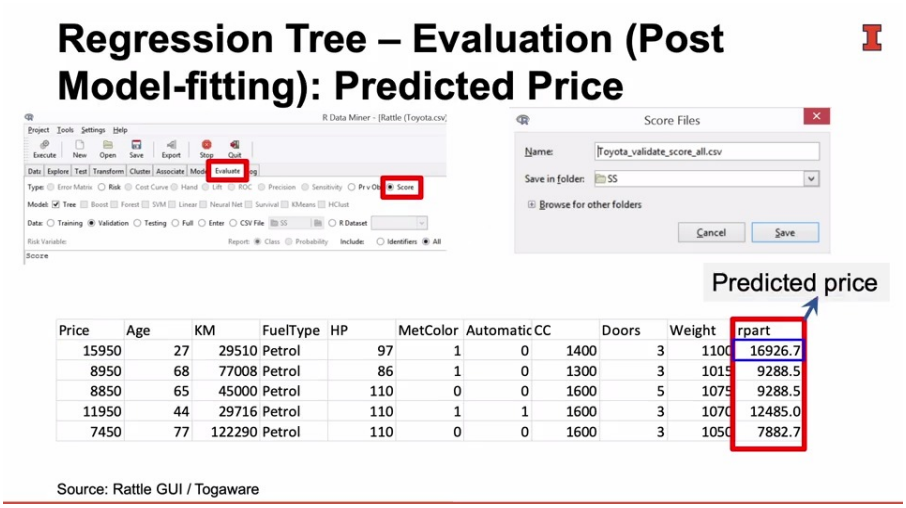
Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight	rpart
15950	27	29510	Petrol	97	1	0	1400	3	1100	16926.7
8950	68	77008	Petrol	86	1	0	1300	3	1015	9288.5
8850	65	45000	Petrol	110	0	0	1600	5	1075	9288.5
11950	44	29716	Petrol	110	1	1	1600	3	1070	12485.0
7450	77	122290	Petrol	110	0	0	1600	3	1050	7882.7

Source: Rattle GUI / Togaware

Transcript

As before, in regression we can ask it to score. So, when you say score, it actually creates a CSV file, and in the CSV file it records the predicted price, okay? Compare, so this is the price of the car

Regression Tree - Evaluation (Post Model-fitting): Predicted Price (2 of 4) - Slide 68



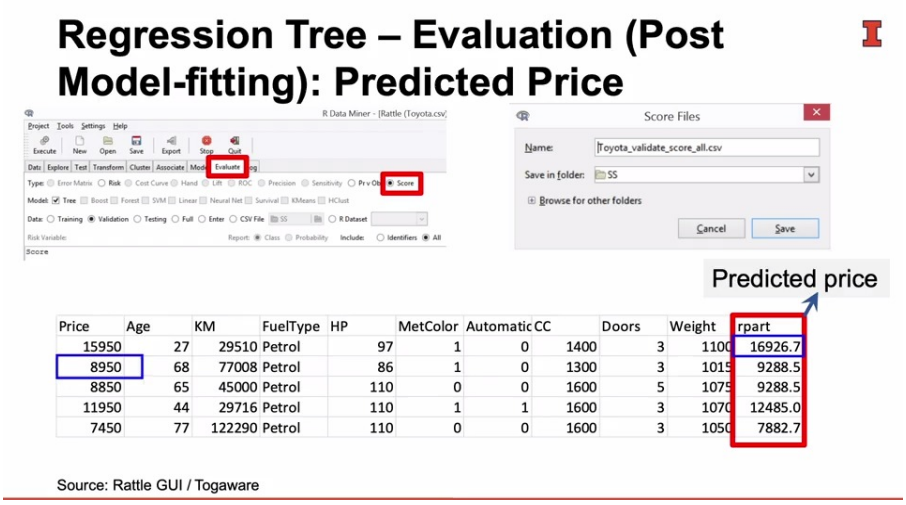
The slide is the same as Slide 67 - Regression Tree - Evaluation (Post Model-fitting): Predicted Price (1 of 4) with 16926.7 of rpart circled.

Source: Rattle GUI / Togaware

Transcript

and this is the predicted price.

Regression Tree - Evaluation (Post Model-fitting): Predicted Price (3 of 4) - Slide 69



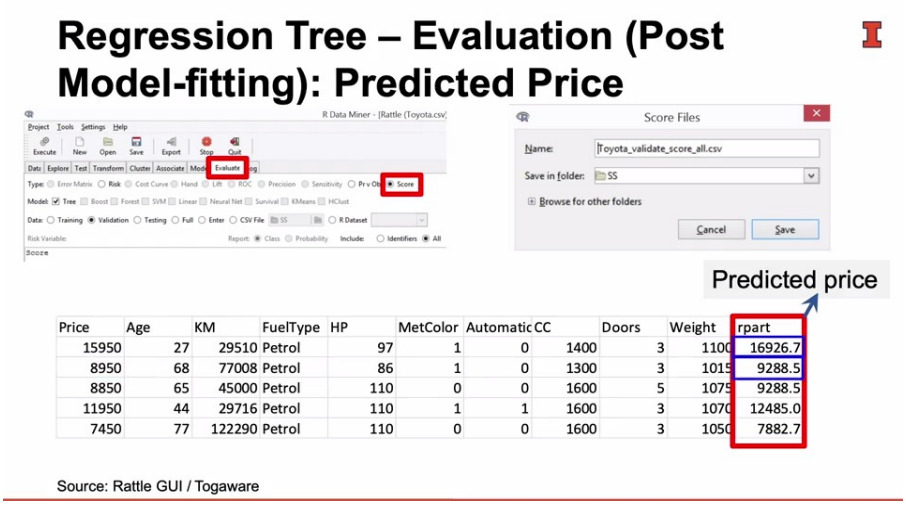
The slide is the same as Slide 68 - Regression Tree - Evaluation (Post Model-fitting): Predicted Price (2 of 4) with 8950, the second row of Price circled and 16926.7 of rpart circled.

Source: Rattle GUI / Togaware

Transcript

This is the price of the car

Regression Tree - Evaluation (Post Model-fitting): Predicted Price (4 of 4) - Slide 70



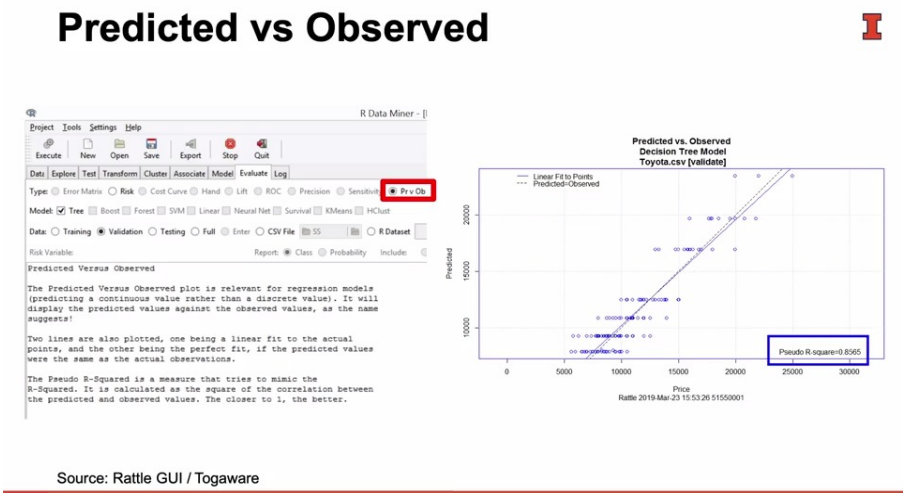
The slide is the same as Slide 69 - Regression Tree - Evaluation (Post Model-fitting): Predicted Price (3 of 4) with 16926.7 and 9288.5, the first and second row of rpart circled. Rpart is the Predicted price.

Source: Rattle GUI / Togaware

Transcript

and this is the predicted price. This is the price of the car and this is the predicted price, it's doing pretty well, right? You also know from your experience, that you can plot the predicted versus the observed value. It gives you a pseudo R-squared, and there it is.

Predicted vs Observed - Slide 71



The slide contains a walkthrough of plotting in Rattle and the graph.

1. In the Evaluate tab, select Pr v Ob and Validation data.
2. Click Execution.

The graph is titled: Predicted vs. Observed, Decision Tree Model, Toyota.csv[Validate]. There are two lines in the graph tgat are close to each other. The solid line is Linear Fit to Points, and the dotted line is Predicted=Observed. The x-axis represents Price, ranging from 0 to 30000 in increments of 5000. The vertical y-axis shows Predicted Price ranging from 0 o 25000 in increments of 5000. Pseudo R-square=0.8565 at the bottom is circled. Points are in horizontal lines along the vertical y-axis.

Source: Rattle GUI / Togaware

Transcript

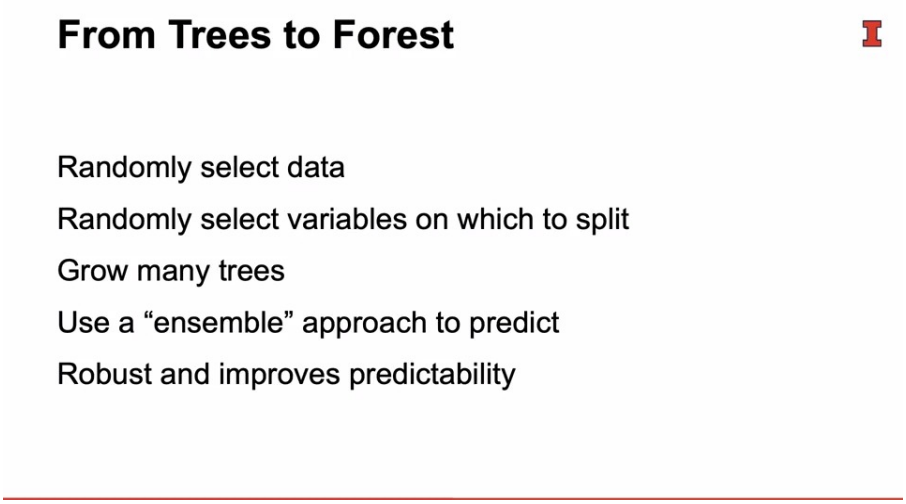
It gives a fit of 85.65 percent. Go back to your notes, and you will see that you got these two lines, they are very close to each other, it's doing pretty well, right? So, here is an example where the classifier classifies values in stock categories and you can actually use it for regression, right? It's a useful method to try when you have a regression model, you have tree model, and you can compare the two.

Lesson 2-7 Introduction to Forests and Spam Filter Exercise

Lesson 2-7.1 Introduction to Forests and Spam Filter Exercise

Media Player for Video

From Trees to Forest - Slide 72



Randomly select data

Randomly select variables on which to split

Grow many trees

Use a "ensemble" approach to predict

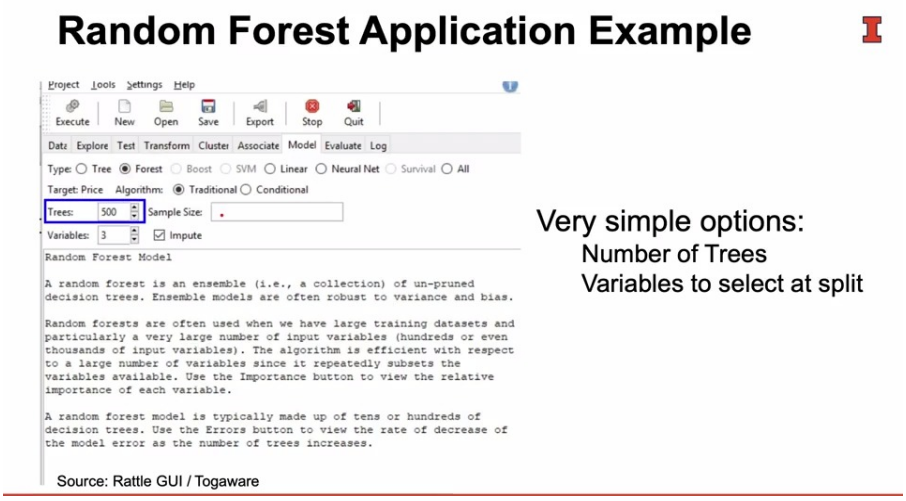
Robust and improves predictability

Transcript

So, as usual, every classifier has a problem, has got some drawbacks, all right? So, trees as you notice are sort of opportunistic, they're sort of a heuristic right? You find the easiest, the one which gives you the highest purity and splits and then finds the next highest purity of fully split. It doesn't have a backtracking mechanism saying, "Can I go back? Can I redo this?" So, it just goes one way and so it may not be the best way to split the data. It also has a problem, right? Because if there are too many variables, it might be confused and then there's too much noise in the data. So, these drawbacks of noise, confusing a tree and maybe some outliers drawing it in a particular direction to build a particular tree could be overcome and if this idea of how to overcome it gives you one of the best classifying methods at least in my opinion with this medium data which are called forests, random forests okay. So, random forest overcomes the tendency of trees to become bias due to their data or due to the noise, right? The way of selection by the initial randomly selecting their data. So, you may select maybe two-thirds of the data and keep aside a third of it. Then what you can do is at every time when you are splitting the data, you don't have to look at all the features to split upon. You can pick randomly three of them and then see which one is the best and then split on those. So, you are using randomness in two ways.

You're splitting your data and you're splitting which feature you are randomly selecting at every stage, which features to use. So, every time, so you select the data and every split gives you a different set of features so each tree you create is sort of unique. You grow many trees like this. Maybe it's 200 trees, 500 trees. Then what do you do, is instead of dropping the card down each tree, you drop the card and on all of this. Let's say this tree says it's 12,000 and this tree says it's 13,000 and this tree this is 49,000 or whatever it is right? You take maybe an average of that. Let's say if it's a category, you're predicting whether somebody would buy and not buy, if 140 of the trees you grew said they will buy and 60 said they won't buy, then you say, "Okay, maybe there's a 70 percent chance this person will buy," and then we can use that to classify this person into somebody who will buy your product or not, right? So, basically, you're using an ensemble of trees so, it's actually not an ensemble. Ensemble means slightly different than using a different method, but that's why it's in code. You're using many trees to [inaudible] and then you are taking an average of that [inaudible] and coming up with a prediction. Turns out, this method of going from trees to forests, the random forest, is extremely robust and improves predictability quite substantially. I'll just show it to you. You may like to run many of your old datasets on the random forest which is available in Rattle and you will see.

Random Forest Application Example - Slide 73



The slide contains a walkthrough of random forest application in Rattle.

Set Trees as 500.

Very simple options:

Number of Trees

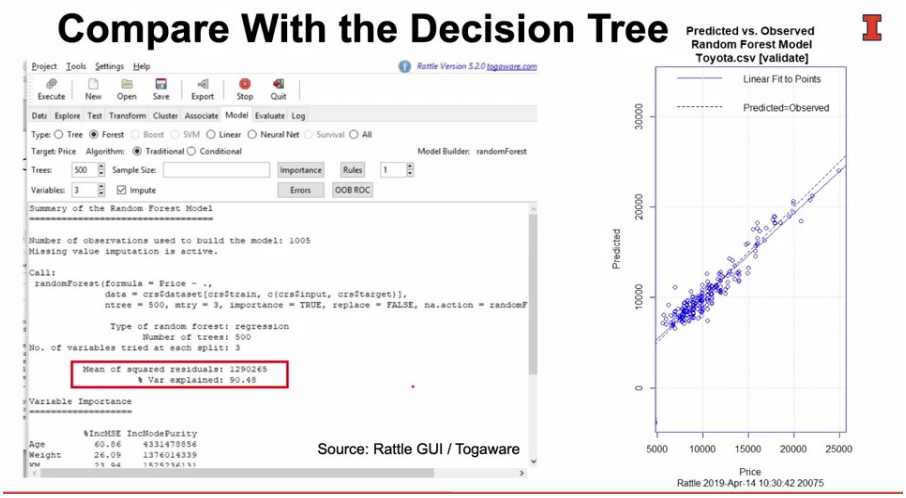
Variables to select at split

Source: Rattle GUI / Togaware

Transcript

So, I'm creating a random forest. I'm using 500 trees and every time we will have, we'll choose three features in each node and decide which one to split on. So, very simple options: number of trees to use and number of variables to select. It says very nicely, "It is an ensemble tree, it's robust, they're often used when you have large training data-sets and particularly a very large number of input variables, hundreds of thousands." The algorithm is fairly efficient, right? Because it's repeatedly sampling from these variables and it's made up of tens to hundreds of decision trees. If you look at it, you can also track as the trees are added, how does the error change?

Compare With the Decision Tree - Slide 74



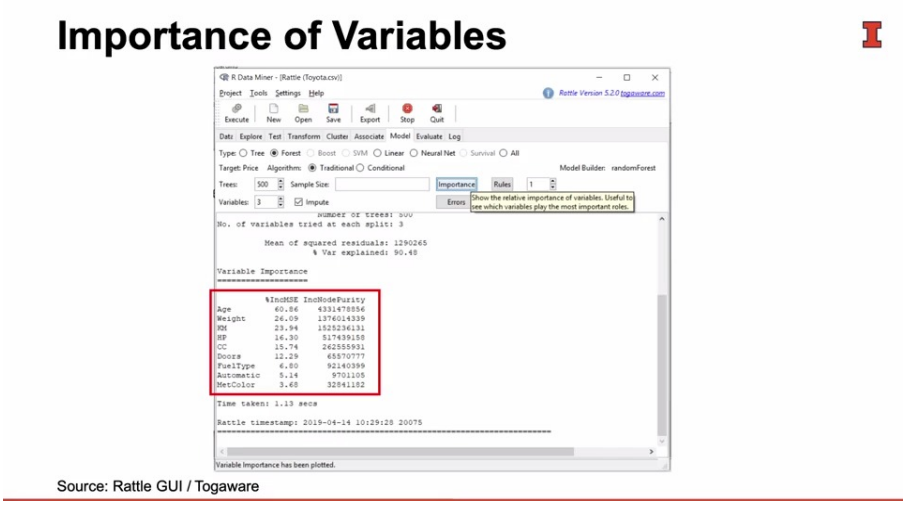
The slide contains the output from Rattle. Mean of squared residuals: 1290265. Variability explained: 90.48. The graph is on the right, titled Predicted vs. Observed Random Forest Model Toyota.csv [Validate]. The solid line is Linear Fit to Points, and the dotted line is Predicted=Observed which are close to each other. Both lines are diagonal. The horizontal x-axis represents Price, ranging from 5000 to 25000 in increments of 5000. The vertical y-axis shows Predicted Price ranging from 0 o 30000 in increments of 10000. Points are along the lines.

Source: Rattle GUI / Togaware

Transcript

So, without going too much into detail, look at what it does. It's able to improve your prediction, the same dataset which is 85 percent to 90 percent. You'd be surprised that random forest can do much better. You'd think randomness means it will not do better but actually what it's doing is, it's filtering out noise in a very nice way. So, this is the same dataset on the Toyota and you can see higher explanation and you can also see the predicted to observe further substantially better.

Importance of Variables - Slide 75



The slide contains the output of Rattle in the table below.

The output of Rattle

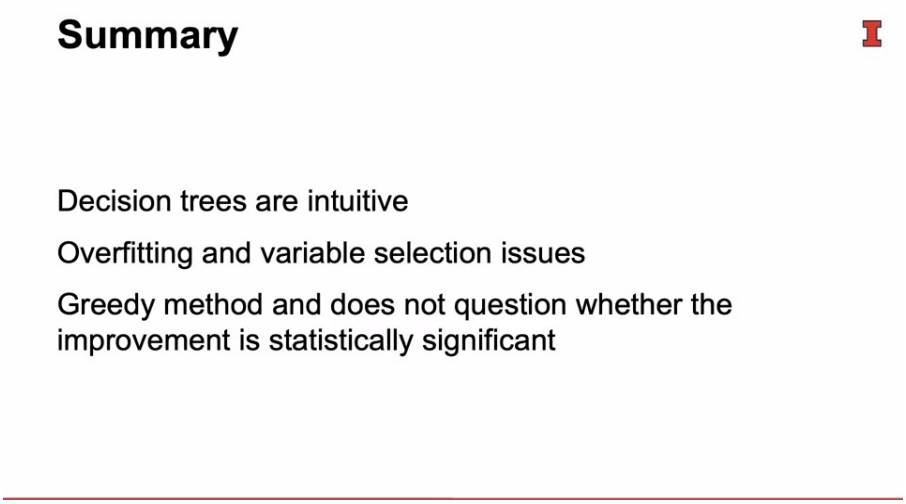
	%IncMSE	IncNodePurity
Age	60.86	4331478856
Weight	26.09	1376014339
KM	23.94	1525236131
HP	16.30	517439153
CC	15.74	262555931
Doors	12.29	65570777
FuelType	6.80	92140399
Automatic	5.14	9701105
MetColor	3.68	32841182

Source: Rattle GUI / Togaware

Transcript

So, without going too much into detail, look at what it does. It's able to improve your prediction, the same dataset which is 85 percent to 90 percent. You'd be surprised that random forest can do much better. You'd think randomness means it will not do better but actually what it's doing is, it's filtering out noise in a very nice way. So, this is the same dataset on the Toyota and you can see higher explanation and you can also see the predicted to observe further substantially better. Random Forests also, depending on which variable is came earlier to split comes up with a measure of variable importance that says the most important variable is age, then is weight, then is kilometer driven and then is harsh bar. So, it also gives you a feature. The one thing it doesn't give which you may supplement is it doesn't give you the directionality. It doesn't say, it says age is important, but it doesn't say if age increases what happens to your price, right? So, the directionality you can even glean by running a simple regression model looking at the sign of those variables if they're significant and attribute it to h. So, that's the only drawback but this method is phenomenal as a classifier.

Summary (1 of 2) - Slide 76



Decision trees are intuitive

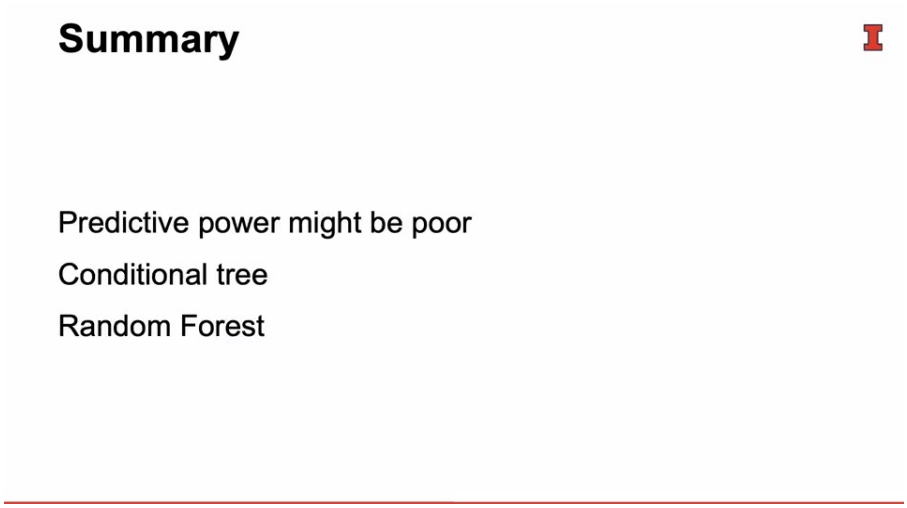
Overfitting and variable selection issues

Greedy method and does not question whether the improvement is statistically significant

Transcript

So, in summary, decision trees are very intuitive. But they have a drawback that they can overfit and if there are many variables, how will you select? So, these are problems they have. So, it is a greedy method, a heuristic method, right? It also doesn't question whether the improvement at every stage is statistically significant or not.

Summary (2 of 2) - Slide 77



Predictive power might be poor

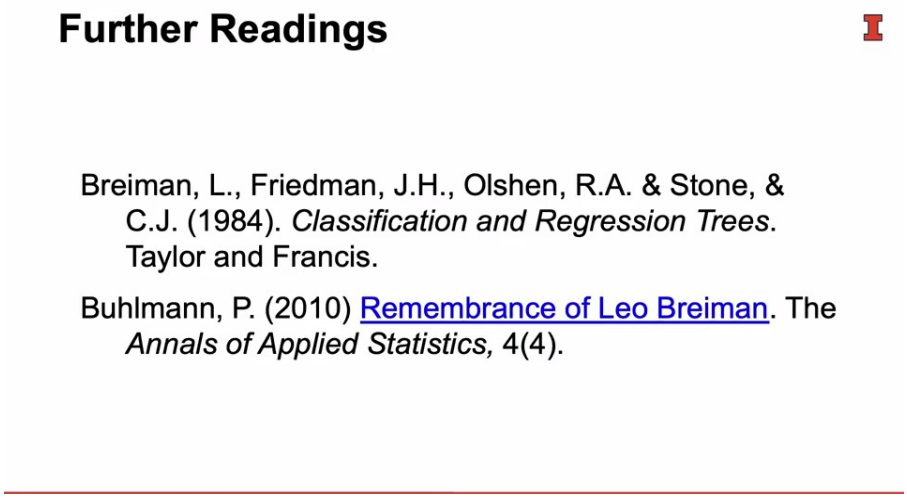
Conditional tree

Random Forest

Transcript

So, the predictive power may be poor. There are two approaches which have been proposed to solve it. We have talked about one of them the random forest is another way of controlling for which Rattle has, which is called a conditional tree. You can read about it in the references but I do want you to remember this thing, that the random forest which is just an ensemble method of predicting is very efficient and one of the better predictors available.

Further Readings - Slide 78



Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, & C.J. (1984). *Classification and Regression Trees*.

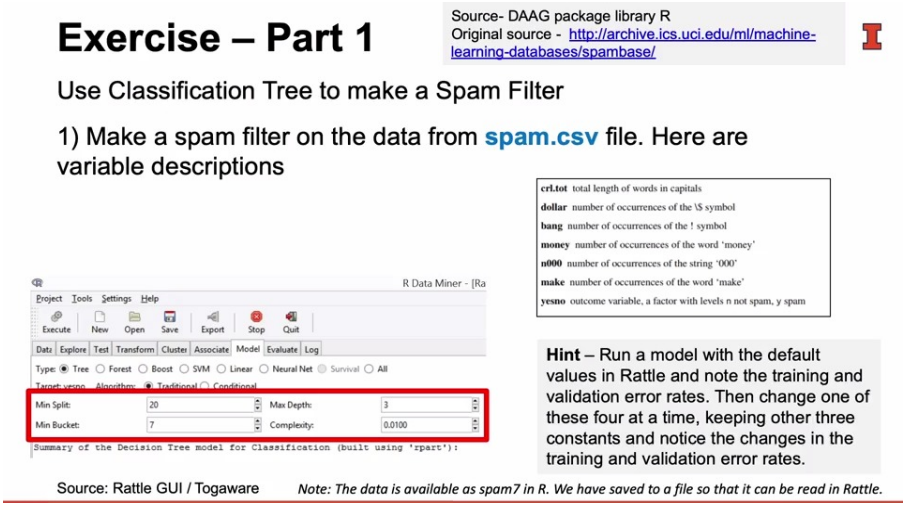
Taylor and Francis.

Buhlmann, P. (2010) Remembrance of Leo Breiman. *The Annals of Applied Statistics*, 4(4).

Transcript

The two readings I will emphasize as much of this development, is by the guru of classification machine learning, whatever it is, of statistics Leo Breiman and definitely his book which is available is something what you should read the "Remembrance of Leo Breiman" and then the contributions he has made to the subject. I want to give you a small exercise and if you can do it before the attempting the next module that would be great because I'm going to use that to understand how classification error can be computed okay?

Exercise - Part 1 - Slide 79



The slide contains a screenshot of Rattle which is the same as Slide 65 - Regression Tree - Rules (Post Model-Fitting) (4 of 4). Min Split, Max Depth, Min Bucket, Complexity are Circled.

Use Classification Tree to make a Spam Filter

1) Make a spam filter on the data from spam.cvs file. Here are variable descriptions

variable descriptions

variable	descriptions
crl.tot	total length of words in capitals
dollar	number of occurrences of the $ symbol
bang	number of occurrences of the ! symbol
money	number of occurrences of the word 'money'
n000	number of occurrences of the string '000'
make	number of occurrences of the word 'make'
yesno	outcome variable, a factor with levels n not spam, y spam

Hint – Run a model with the default values in Rattle and note the training and validation error rates. Then change one of these four at a time, keeping other three constants and notice the changes in the training and validation error rates.

Note: The data is available as spam7 in R. We have saved to a file so that it can be read in Rattle.

Source: DAAG package library R

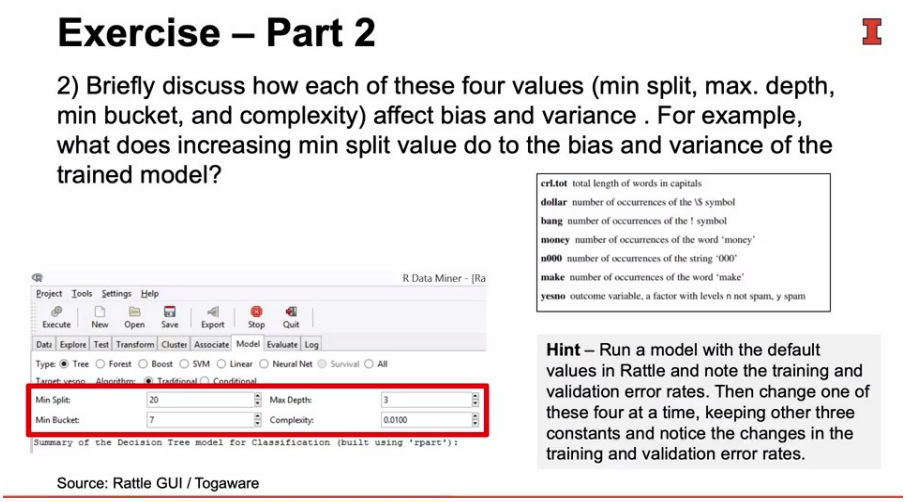
Original source: Machine Learning Database [4]

Source: Rattle GUI / Togaware

Transcript

In the exercise, I would like you to use a classification tree to develop a spam filter. The instructions are given here. We've given you that data and we are asking you to use a decision tree to classify an email into spam or no spam you just drop it.

Exercise - Part 2 - Slide 80



The slide contains the same picture, hint box and variable table as Slide 79 - Exercise - Part 1.

2) Briefly discuss how each of these four values (min split, max depth, min bucket, and complexity) affect bias and variance. For example, what does increasing min split value do to the bias and variance of the trained model?

Source: Rattle GUI / Togaware

Transcript

You may have to control some of these values to develop the spam filter, we will see it, your results in the next module.

References

1. Wikimedia Commons. (2007). *File: BMI en.svg*. Retrieved from https://commons.wikimedia.org/wiki/File:BMI_en.svg.
2. Togaware. (2006). *Rattle* [Graphical User Interface for R]. Retrieved from https://rattle.togaware.com/
3. Wiley. (2019). *Ledolter*. Retrieved from https://www.biz.uiowa.edu/faculty/jledolter/datamining/datatext.html.
4. Index of /ml/machine-learning-databases/spambase. (2019). Retrived from http://archive.ics.uci.edu/ml/machine-learning-databases/spambase/