Balanced Data 🡪

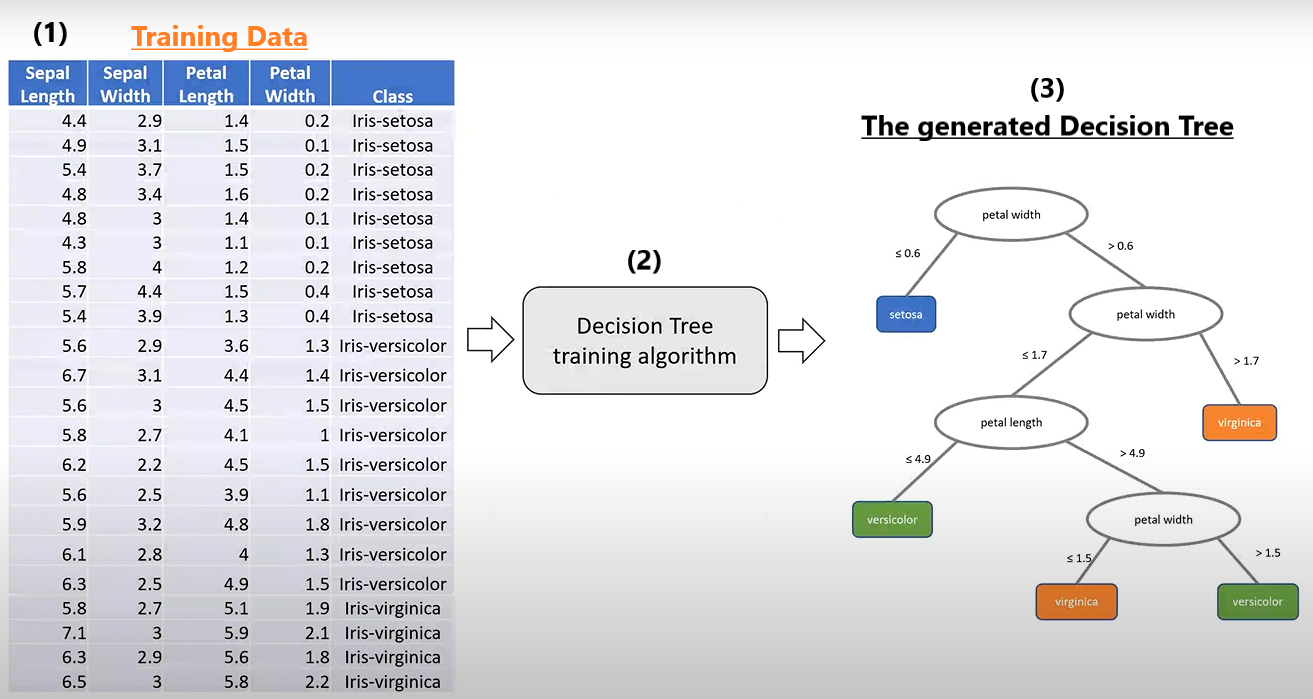
* For a dataset with n classes.
* The Training set should have an equal ratio of each class.
  + We want and even set if we can get it.
* Ex. If we have 90 observations and 3 classes:
  + 30 of the observations should be for 1 class, 30 for the second, and 30 for the 3rd.

Decision Trees Notes.

* Explain what a Decision Tree is.
* Describe how to use a trained Decision Tree to classify a new observation.
* Explain how Decision Trees compares to k-Nearest Neighbors.
* List some of the pros and cons of Decision Trees.
* Explain what **Overfitting** is and how to combat it.

Decision Trees (summary).

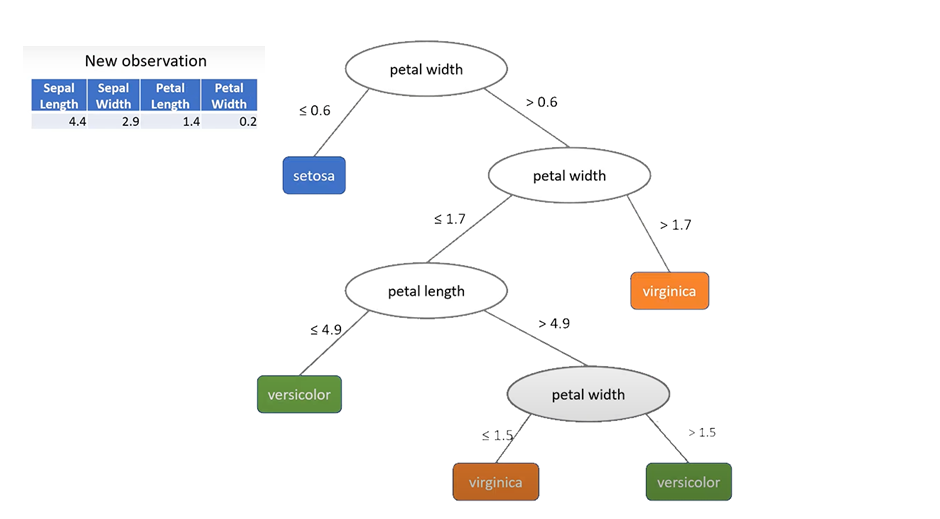
General idea behind a Decision Tree:

1. You start with your Training data. You feed it into 🡪
2. A Decision Tree Training algorithm. Which is going to actually generate 🡪
3. The Decision Tree. (Its structure is just a flowchart).

Decision Tree Structure.

* Decision Trees provide a “flow chart” over Feature values to Classify observations.
  + Each internal node specifies a Feature to split the dataset on.
    - 1 child per value for nominal Features.
    - 2 children for above/below threshold for numeric Features.
  + Leaf Nodes contain the predicted class Label.
* The Model is the Tree. (Nested rules).
  + When you train one of these Decision Trees, the Model that is outputted is the Tree. It is the nested rules.
    - It is NOT the data.
    - It will capture statistics about the Training Set (ex ‘n’). but we are not actually taking the real data. (we are not storing the real data).
  + Decision Tree’s Models are much smaller than those for kNN.
    - A kNN Model is the entire Training set.
* Feature scaling does not matter, it doesn’t affect trees.
  + And qualitative (nominal) Labels can be used.
  + Reason for which it does not matter:
    - Because we are considering each one of these Features in isolation to the other Features.
    - i.e. Each of the nodes. We are not trying to combine them in some way.
* Decision Trees can easily Overfit a Training set.
  + Extreme case (with no pruning): Every distinct observation is represented by a distinct path through the tree.
    - So, you end up with Leaf Nodes that have exactly 1 class.
    - May even have 1 observation (if every observation has some unique combination of Features).
  + Overfit Models “predict” the Training data extremely well (perfectly in a lot of cases),
    - BUT: They do NOT perform well on new observations.
    - Reason: Because the original Model was not generalized enough to handle things that don’t look exactly like the Training set.
  + Pruning is the method that we use to prevent Overfitting. (there are a couple of ways of doing it).
* Decision Trees can Bias the majority class.
  + So, you really have to Train with **Balanced Data** (if possible).
* Unlike many other Classification algorithms, the Models are Human Interpretable.
  + You can actually see the decisions that were being made during the process.
  + Thus, it is easy to identify where a misclassification went wrong.
* When we are creating the Decision Tree:
  + We are taking our Training data and we are splitting it using the threshold.
  + So, some of the data will be captured by the Subtree on the left. And the other part of the data will be captured by the Subtree on the right.
* The Root Node🡪 Has some feature that it specifies. Petal width.
* The Internal Nodes 🡪 Are Features.
* Leaf Nodes 🡪 The class that will be given to the new observation if it reaches it.
* The Branches 🡪 Which go to their children, have a Label that tells you to follow that branch based on some conditions. There are rules associated with each branch.
  + In the case of Numeric Features:
    - We use some kind of threshold to determine whether we go left or right.
  + In the case of Nominal Data: (NOT doing it)
    - Which is qualitative data. Ex: a name or a label.
    - There would be one branch per value that you could have for that data.

Classifying new observations using a Trained Decision Tree.

* If a have a New Observation I would like to Classify:
  + You take your instance and then you follow branches based on the feature that was specified and the threshold boundary that is given.
  + I look at the Root Node and see which feature it is specifying.
  + I find the value of that feature in my new observation.
  + I look at the branches of the Root Node, and depending of the value of the new observation, I follow the correct branch.
    - In this case: Based on the value 0.2, we follow the left branch.
      * Thus, classifying this instance (the new observation) as Setosa.

Overfitting.

* Decision Trees are prone to Overfitting.
* Decision Trees left on their own, would end up giving us an Overfit Tree (an Overfit Model).
* In Decision Trees, because you can just keep generating more and more rules, it really narrows things down.
  + So you end up with some very specific cases that it will try to Classify.
    - These specific cases are ones that evisted in the Training data.
  + The prolem is, that when you go to Classify a new observation 🡪 This will through things off quite a bit.
  + We do NOT want this.
    - We do NOT want it to get every single Training instance correct. Because once we go to look at new data 🡪
      * That Model is not going to generalize well to that unseen data.
* There is a way to get a Decision Tree to produce a Non-Overfitted Model.
  + The Model Missclassifies some observations, but this is OKAY.
  + What we get is these more generalized and simplified rules.
    - We want these rules if we can get them.

|  |  |
| --- | --- |
| Overfit | Not Overfit |
| These graphs represent the result if we were to plot some of the Features of the Iris Training Dataset.  Each graph represents a different Model. Both are using the same dataset.  In this case, we are looking at: sepal\_length versus sepal\_width.    Blue 🡪 Virginica.  Green 🡪 Versicolor.  Blue 🡪 Setosa. | |
|  |  |
| This Model shows us that:  We’ve got a couple of these observations that are kind of **nestled** in with a different class.  I.e. Some observations are situated **surrounded** by a different class.  These blue observations are **embedded** here with the other ones. | In this Model we can see:  We **Misclassify** a bunch of observations, and that is OKAY.  What we get are these more generalized rules.  Which say:  Everything over at (A) is going to be red (Setosa).  Everything over at (B) is going to be blue (Virginica).  Everything over at (C) is going to be green (Versicolor). |

Pruning – For Avoiding Overfit.

* Pruning 🡪 Removing unnecessary parts of the Decision Tree.
  + We are going to get rid of whole subtrees.
* Pruning makes the Model more generalizable.

|  |  |
| --- | --- |
|  | |
|  | If we look at all of the Leaf Nodes in the green selection, we see that:  Versicolor is the majority.  It’s got 49 out of 54 observations. (47+2=49).  We know this because:  The portion of the Training data that the Root Node (D.) of the yellow tree is capturing is 🡪 54 of the Training observations.  i.e. 54 of the Training observations are captured in this branch/Subtree (in the Root of yellow tree).  Virginica makes up the remainder, which is pretty small.  We only have 5 Virginica examples here.  ( K = 1, L = 3, Q = 1 ).  So, we might be okay with basically saying that K, L, and Q are Outliers, that we do not care about them.  They are too specific cases so 🡪 We want to get rid of them.  THUS, we can get rid off the whole yellow Subtree and replace it with a Versicolor Leaf Node. |
|  |  |
|  | When we are building the Decision Tree, we are going to keep track of the distribution of Labels.  So, instead of only having the majority Label specified, it will actually say:  Versicolor = 49.  Virginica = 5.  So that we actually know what the distribution is, across the Labels.  This will be useful later on, for ex: If we want to get a Confidence Score. |
| **Pruning makes our Model more generalizable**, so 🡪  Those Leaves that were very specific, we are not going to get those anymore.  Reason: Because we are just going to predict the majority class for that Subtree.  Regarding the previous graph:  For those specific Virginica color observations we will predict the majority class of the Subtree.  Which in this case was: Versicolor. | |
|  |  |
|  |  |