Recap Section 7

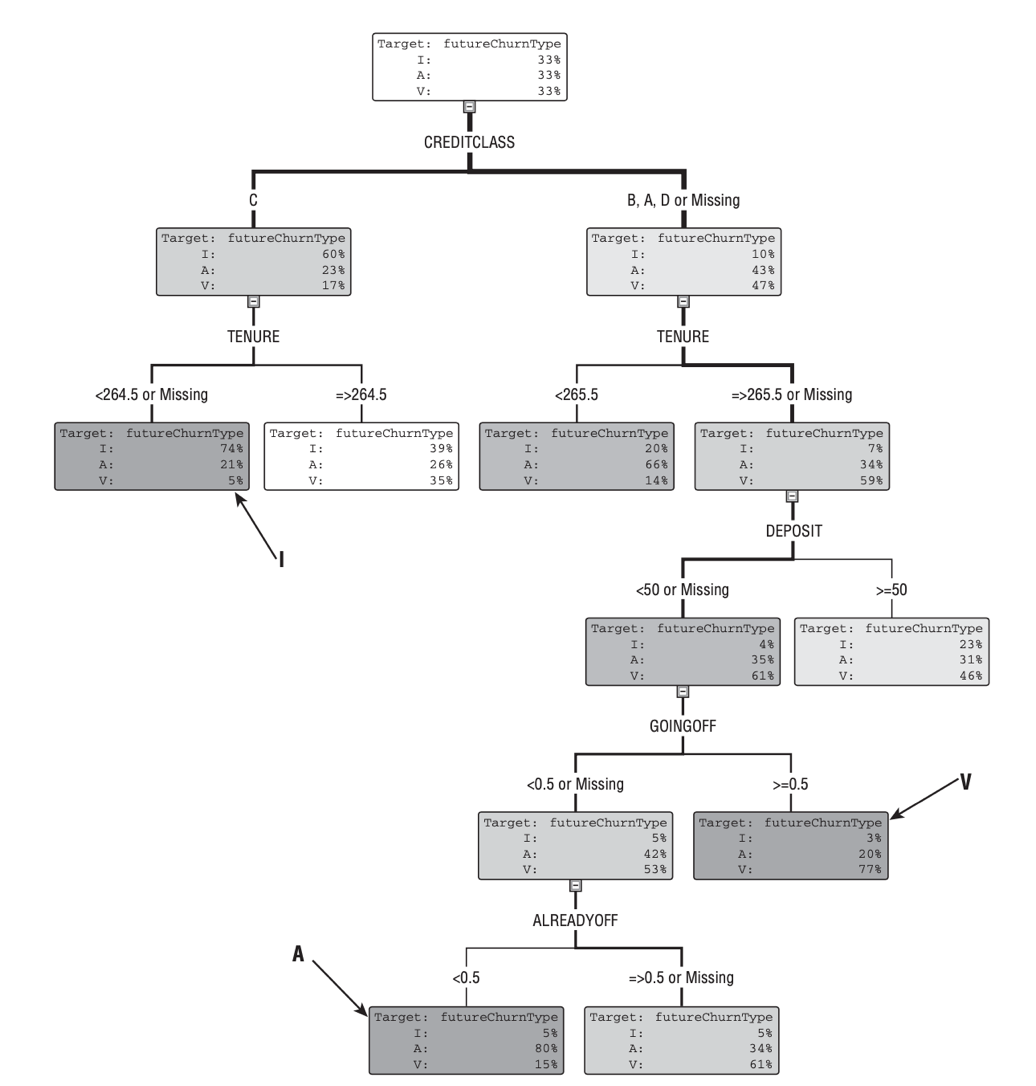
# Intro

* Recursively split data, based on chosen variables
* Classification, estimation, prediction
* All decision tree problems could be solved with a neural network; however, the decision tree is much better at showing the exact rule it used to classify the data.
* Changes in data get recognized very fast with DT since they are based on a very strict pattern.
* Not very precise overall
* Easy algorithm
* Relatively fast -> money effective (business use case)
* Easy to explain predictions

# What is a Decision Tree?

* Hierarchical collection of rules
* Divides large collection of data into successively smaller groups of records

## A Typical Decision Tree



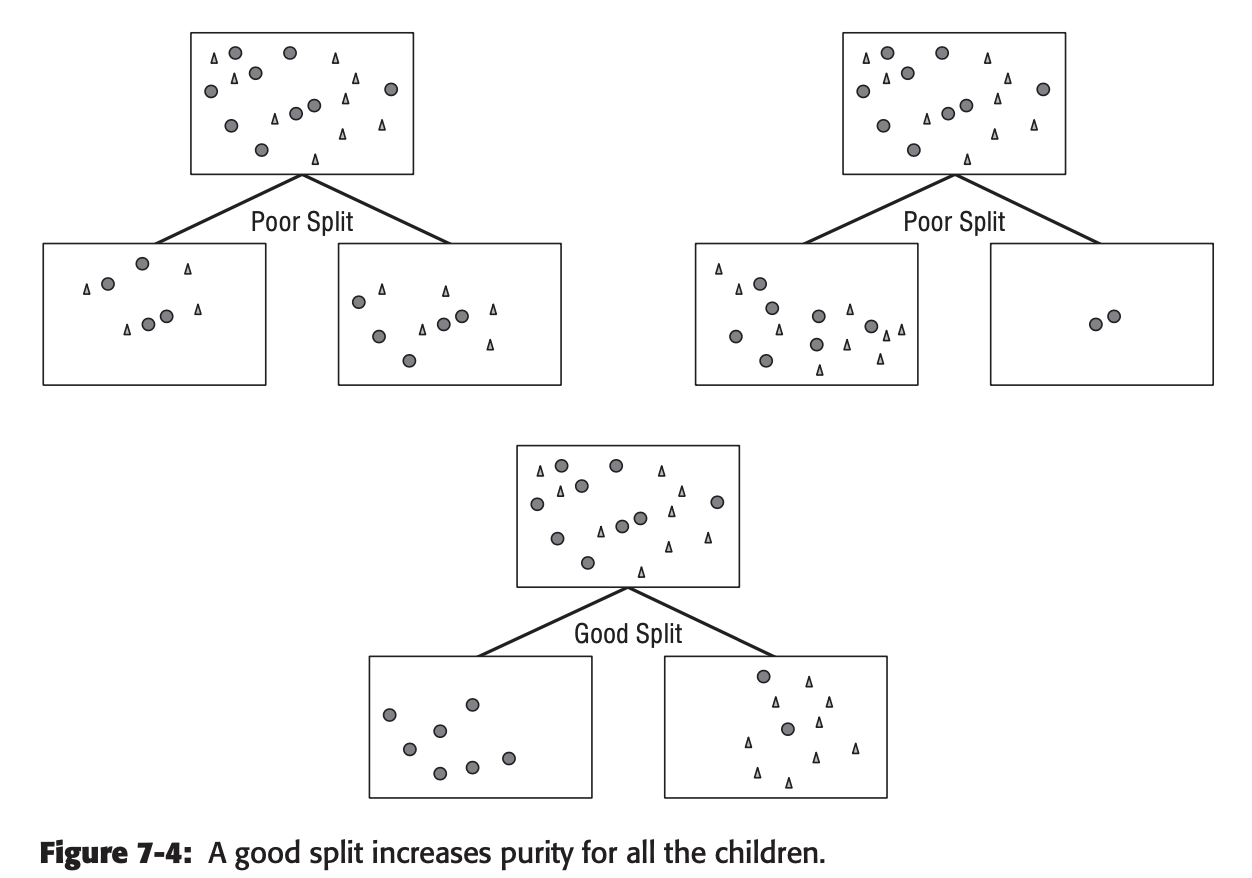
* Rote node at the top -> contains the whole dataset
* The higher the node the more important the rule
* Use DT to identify and filter a more useful subset

## Using the Tree to Produce Rankings

* Rankings are more important than the actual value (in this case curn rate)
* It’s more important to know the top 100 CURN likely customers than the actual rate of the customer

# Growing Decision Trees

* An algorithm performs a split on each of the variables an chooses the one with the purest split regarding the target value
* Always choose the current rule as the one which purifies the target values to most
* A good split also creates child nodes of similar size, or at least does not create nodes containing very few records.



* Numeric values are never weighted nor manipulated any other way in DTs
* When split on a categorical value the most common approach is to generate a child node for each category
* With smaller datasets and more categories this quickly gets a bad idea because there are too few child values left to split on further.
* Therefore, categories are often grouped in a child node.
* The best way to do it, is to calculate the distribution of a child value in a given category and group the categories into child node based on their distribution.

## Splitting with Missing or Null values

* A bad approach is to throw out all records with null values since those values are probably not random
* Another bad approach is to fill those null values with a default value, since a missing value is still an information.
* A way to handle missing values is to store multiple rules in a node and execute the next rule on missing values.

## Growing the full Tree

* For each new split all the variables are considered, even if they have already been used in further up the tree
* The algorithm builds the tree until one of the following breakpoints is reached:
  + Any further split would not significantly increase the purity of the child nodes
  + The number of records per node reached a defined lower bound.
  + The given limit of DT layers has been built. In this case the DT is fully grown. Any further growing might leave to overfitting

## Finding the best Split

* Most of the time many potential splits would lead to a similar results, even when taking a totally different approach and variables. This leads to similar performance even when the structure of the dt varies a lot.

## Defining purity

When it comes to defining purity, the algorithm to determine the purity is dependent on the type of target variable.

### Categorical type

* Gini (population diversity)
* Entropy (information gain)
* Chi-squared test
* Incremental response

### Numeric

* Reduction in variance
* F test

#### Gini

In case of a binary split this algorithm spits out a number between 0.5 (both categories are equally represented) and 1.0 (only one category is represented in the node). It is calculated by summing up the squares of the relative portion of the population (0.0 - 1.0).

Ex. A node with 80% of one category and 20% of the other the Gini algorithm would calculate: P(cat1^2) + P(cat2^2) =

A screenshot of a cell phone

Description automatically generated

Figure 1: The Gini algorithm on binary splits

#### Entropy Reduction

This algorithm gives a quite similar output as the Gini. In this algorithm the lower the score the purer the leaf, and the Range of outputs vary from 0.0 to 1.0

A close up of a piece of paper

Description automatically generated

It is calculated like this: *-1 \* (P(circle)log2P(circle)+ P(triangle)log2P(triangle) )* Where circle is one and triangle the other category. P = Proportion. The -1 is just to make the output positive as the logarithm of a ratio will give you a negative value

#### Information Gain ratio

Tbd

#### Chi-Square

Tbd

#### Reduction in Variance

#### F Test

# Pruning

When not stopped, the DT building algorithm is likely to overfit the model, because it searches for new splits in every node. To fight this, you can either set a minimum size of records per leaf, or a maximum number of layers in the tree. Another way to prevent overfitting is by pruning. With this method the tree is fully grown and the splits which tend to overfit the model are cut away later.

### The CART Pruning algorithm

This algorithm identifies subtrees inside the fully-grown tree and calculates the subtree with the best validation to misclassification ratio (average squared error for numeric target) on the validation set

