A Gentle Introduction to Machine Learning

#VSSML16

September 2016

Outline

Outline

A Churn Problem

- You're the CEO of a mobile phone service (congratulations!)
- Some percent of your customers leave every month (churn)
- You want to reach out to the ones you think are going to leave next month (but you can't call everyone)
- And who might they be?

Begin with the End In Mind

- Currently, you use a simple targeting strategy designed by hand, which identifies the 10,000 customers most likely to churn
- For every five people you call, two are considering leaving (false positive rate of 60%)
- On these customers, your operators can convince half to stay
- Each customer saved is worth $500
- What if you could reduce that false positive rate to 40%?

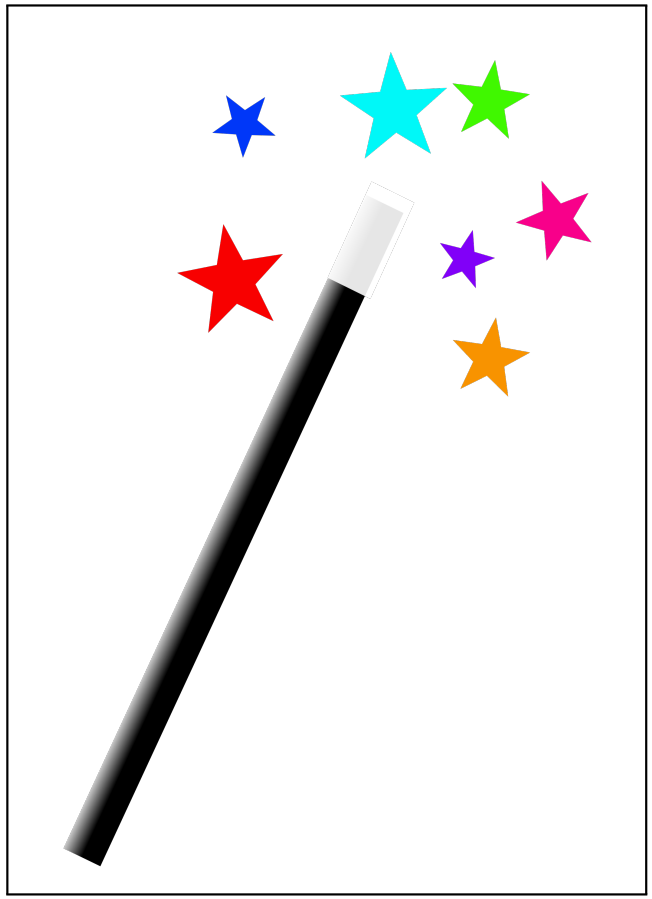
You Have The Data!

Luckily, you've been tracking customer outcomes for months, along with a number of relevant customer attributes.

Minutes Used	Last Month Bill	Support Calls	Website Visits	Churn?
104	$103.60	0	0	No
124	$56.33	1	0	No
56	$214.60	2	0	Yes
2410	$305.60	0	5	No
536	$145.70	0	0	No
234	$122.09	0	1	No
201	$185.76	1	7	Yes
111	$83.60	3	2	No

Now . . . magic!

- Can we use the data to create a better targeting strategy, with virtually no other work? (Spoiler: YES)
- Can we use the data to evaluate that strategy, confirming a better false positive rate? (Spoiler: YES)
- How? (Spoiler: MACHINE LEARNING)



Outline

In a Nutshell

- Collect row-column data about *past data* from your prediction problem (e.g., previous months of customer data)
- Feed this data to a machine learning algorithm
- The algorithm *creates a program* (alternately, “predictive model” or “classifier”) that makes predictions on *future data*

Traditional Expert System: Expert and Programmer

- Machine learning breaks the traditional paradigm of expert systems
- Experts know how the prediction is made
- Programmers know how to translate the expert's knowledge into a computer program that makes the prediction
- Constructing a system requires both (though they may be the same person)

Using Machine Learning: Data and Algorithm

- With machine learning, *data* replaces the expert
  - Is often more accurate than “expertise”

    Article: “Specialist Knowledge is Useless and Unhelpful”[1]
  - Requires no human expertise except how to collect the data
  - The data may already be there
- *Algorithms* replace programmers
  - Algorithms are faster than programmers, enabling iteration
  - Algorithms are more modular than programmers
  - Data offers the opportunity for performance measurement (which you would be doing in any case, right?)

[1] `https://www.newscientist.com/article/mg21628930-400-specialist-knowledge-is-useless-and-unhelpful/`

Outline

Today

- Morning
  - Introduction, Models, and Evaluations
  - Ensembles and Logistic Regressions
- Afternoon
  - Clusters and Anomaly Detection
  - Association Discovery and Latent Dirichlet Allocation

Tomorrow

- Morning
  - Basic Transformations
  - Feature Engineering
- Afternoon
  - REST API, Bindings, and Basic Workflows
  - Advanced Workflows: Algorithms as Workflows

Outline

First: The Data

In the simplest and most numerous cases, machine learning begins with row-column data.

Minutes Used	Last Month Bill	Support Calls	Website Visits	Churn?
104	$103.60	0	0	No
124	$56.33	1	0	No
56	$214.60	2	0	Yes
2410	$305.60	0	5	No
536	$145.70	0	0	No
234	$122.09	0	1	No
201	$185.76	1	7	Yes
111	$83.60	3	2	No

Rows: What is the Prediction *About*?

- Each row (or *instance*) in the data represents the thing *about which* you are making a prediction
- Churn prediction: Each row is a *customer*
- Medical diagnosis: Each row is a *patient*
- Credit card fraud detection: Each row is a *transaction*
- Market closing price forecast: Each row is a *day*

Columns: What information Will Help Us Predict?

- Each column (or *field*, or *feature*) represents a *piece of information* about the thing in each row
  - Churn prediction: minutes used, calls to support
  - Medical diagnosis: BMI, temperature, test results
  - Credit card fraud detection: Amount, transaction type, Geographical information (maybe difference from previous transactions)
  - Market closing price forecast: Opening price, previous day's open/close, five day trend
- How much and what type of information should you collect?
  - There's no general answer here; it's "domain-specific"
  - Question one: How much does it matter (as a function of model accuracy)?
  - Question two: How much does it cost to collect?

Objective: What are we Trying to Predict?

- One of the columns is special: The “objective” is the column in the data that contains the information we want to predict
- Churn prediction: Customer did or did not churn
- Medical diagnosis: Positive or negative for some disease
- Credit card fraud detection: Transaction is or is not fraudulent
- Market closing price forecast: The closing price on the given day

The Goal: Creating a *Prediction Machine*

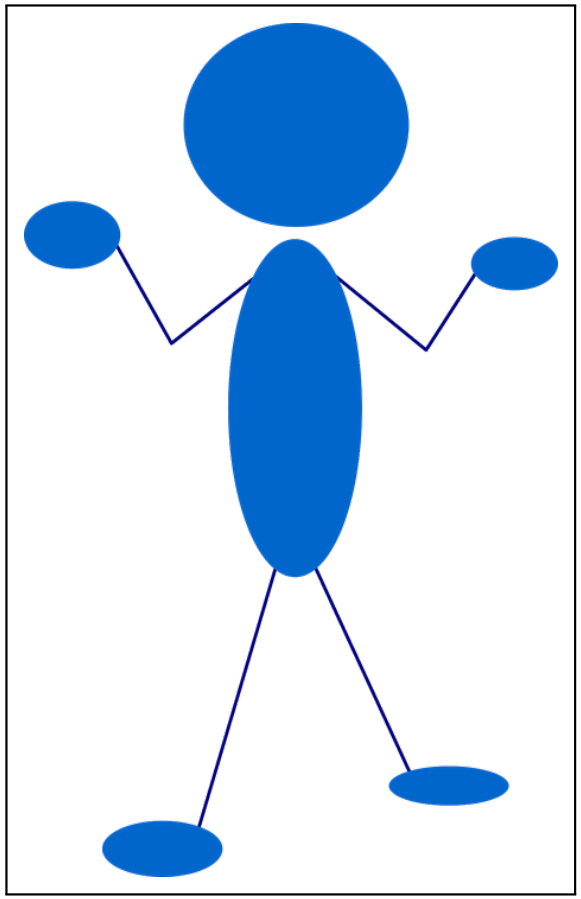
- Feed this data to a machine learning algorithm
- The algorithm constructs a program (or *model* or *classifier*):
  - Inputs: A single row of data, *excluding the objective*
  - Outputs: A prediction of the objective for that row
- Most importantly, the input row *need not come from the training data*

When is this a Good Idea?

- Remember: Machine learning replaces expert and programmer with data and algorithm
- Is it hard to find an expert?
- Can the programmer encode the knowledge (quickly enough, accurately enough)?

People Can't Tell You How They Do It

- Optical character recognition (label a written character given its image)
- Object detection (image does or does not contain an object)
- Speech recognition (label a sentence given spoken sound file)

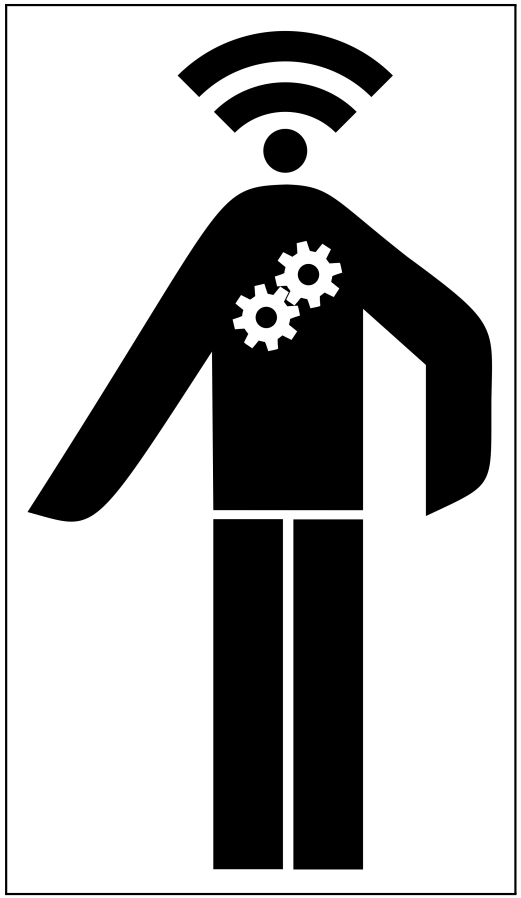


Human Experts are Rare or Expensive

- Medical diagnosis, MRI reading, etc. (predict bio-abnormalities from tests)
- Autonomous helicopter operation
- Game playing (at extremely high levels)

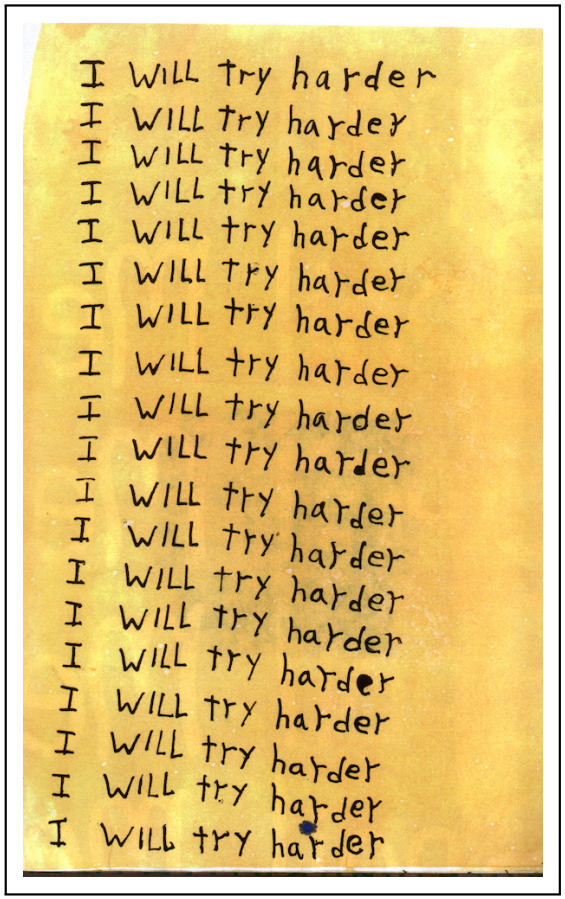
Everyone Needs Their Own Algorithm

- Spam detection (predict spam/not spam from e-mail text)
- Activity recognition (predict user activity from mobile sensors)
- Location prediction (predict future user location given current state)
- Anything that's "personalized", and generates enough data (mobiles!)
- Aside: IoT has the potential to be huge for machine learning



Every Little Bit Counts

- Market activity modeling (buy/sell at a given time)
- Recognition tasks that are “easy” (handwriting detection)
- Churn prediction?
- Any time there’s a performance critical hand-built system



When is this a Bad idea?

- Human experts are relatively cheap and easy to come by
  - Natural language processing for stock traders
  - Automatic defect detection in printer operation
- A program is easily written by hand
  - Some simple if-then rules often perform well
  - The best program is the one that's already there
- The data is difficult or expensive to acquire
  - Medical domains
  - Automatic processing of images

Outline

Searching in Hypothesis Space

- Machine learning algorithms don't exactly *create* a program
- More accurately, they *search* for a program consistent with the training data
  - Search must be very clever
  - Usually there is some sort of "Occam's razor" involved
- The space of possible solutions in which the algorithm search is called its *hypothesis space*
- The algorithm is trying to do "reverse science"!

Some Hypothesis Spaces

- Logistic regression: Sets of all possible feature coefficients and a bias term
- Neural network: Weights for all nodes in the network
- Support vector machines: Coefficients on each training point
- Note: Parametric vs. Non-parametric

A Simple Hypothesis Space

Consider thresholds on a single variable, and predict “majority class” of the resulting subsets

Minutes Used	Last Month Bill	Support Calls	Website Visits	Churn?
104	$103.60	0	0	No
124	$56.33	1	0	No
56	$214.60	2	0	Yes
2410	$305.60	0	5	No
536	$145.70	0	0	No
234	$122.09	0	1	No
201	$185.76	1	7	Yes
111	$83.60	3	2	No

A Simple Hypothesis Space

`Visits to Website > 0`

Minutes Used	Last Month Bill	Support Calls	Website Visits	Churn?
104	$103.60	0	0	No
124	$56.33	1	0	No
56	$214.60	2	0	Yes
2410	$305.60	0	5	No
536	$145.70	0	0	No
234	$122.09	0	1	No
201	$185.76	1	7	Yes
111	$83.60	3	2	No

A Simple Hypothesis Space

`Minutes > 200`

Minutes Used	Last Month Bill	Support Calls	Website Visits	Churn?
104	$103.60	0	0	No
124	$56.33	1	0	No
56	$214.60	2	0	Yes
2410	$305.60	0	5	No
536	$145.70	0	0	No
234	$122.09	0	1	No
201	$185.76	1	7	Yes
111	$83.60	3	2	No

A Simple Hypothesis Space

```
Last Bill > $180
```

Minutes Used	Last Month Bill	Support Calls	Website Visits	Churn?
104	$103.60	0	0	No
124	$56.33	1	0	No
56	$214.60	2	0	Yes
2410	$305.60	0	5	No
536	$145.70	0	0	No
234	$122.09	0	1	No
201	$185.76	1	7	Yes
111	$83.60	3	2	No

Good Question

- Subset purity
- Large difference between the *prior* and *posterior* objective distributions
- How to find the best question? Try them all!

Expanding the Space

There's no reason to stop at just one question:

`Last Bill > $180`

Minutes Used	Last Month Bill	Support Calls	Website Visits	Churn?
104	$103.60	0	0	No
124	$56.33	1	0	No
56	$214.60	2	0	Yes
2410	$305.60	0	5	No
536	$145.70	0	0	No
234	$122.09	0	1	No
201	$185.76	1	7	Yes
111	$83.60	3	2	No

Expanding the Space

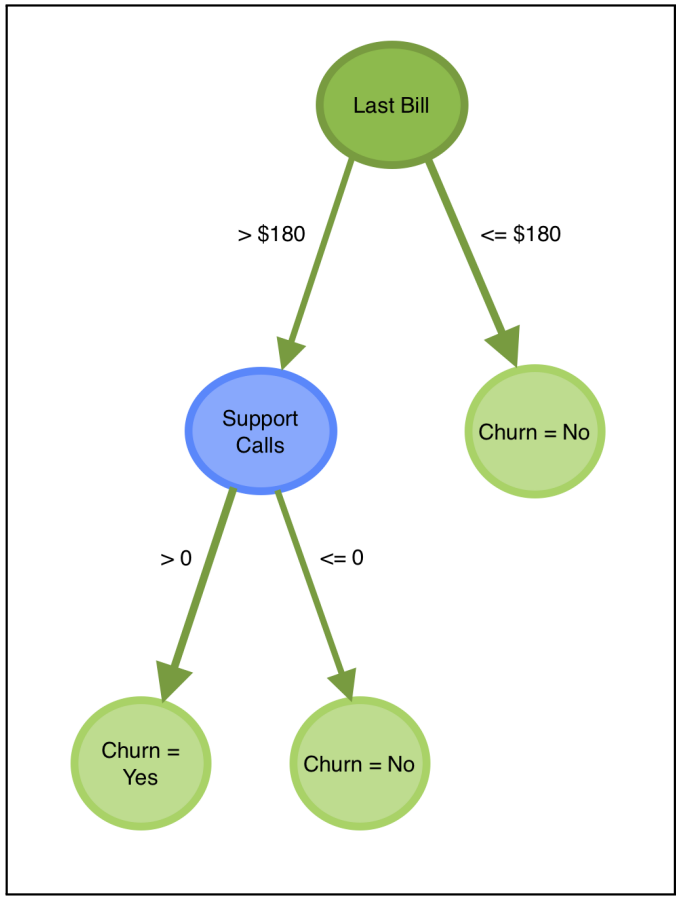
There's no reason to stop at just one question:

```
Last Bill > $180 AND Support Calls > 0
```

Minutes Used	Last Month Bill	Support Calls	Website Visits	Churn?
104	$103.60	0	0	No
124	$56.33	1	0	No
56	$214.60	2	0	Yes
2410	$305.60	0	5	No
536	$145.70	0	0	No
234	$122.09	0	1	No
201	$185.76	1	7	Yes
111	$83.60	3	2	No

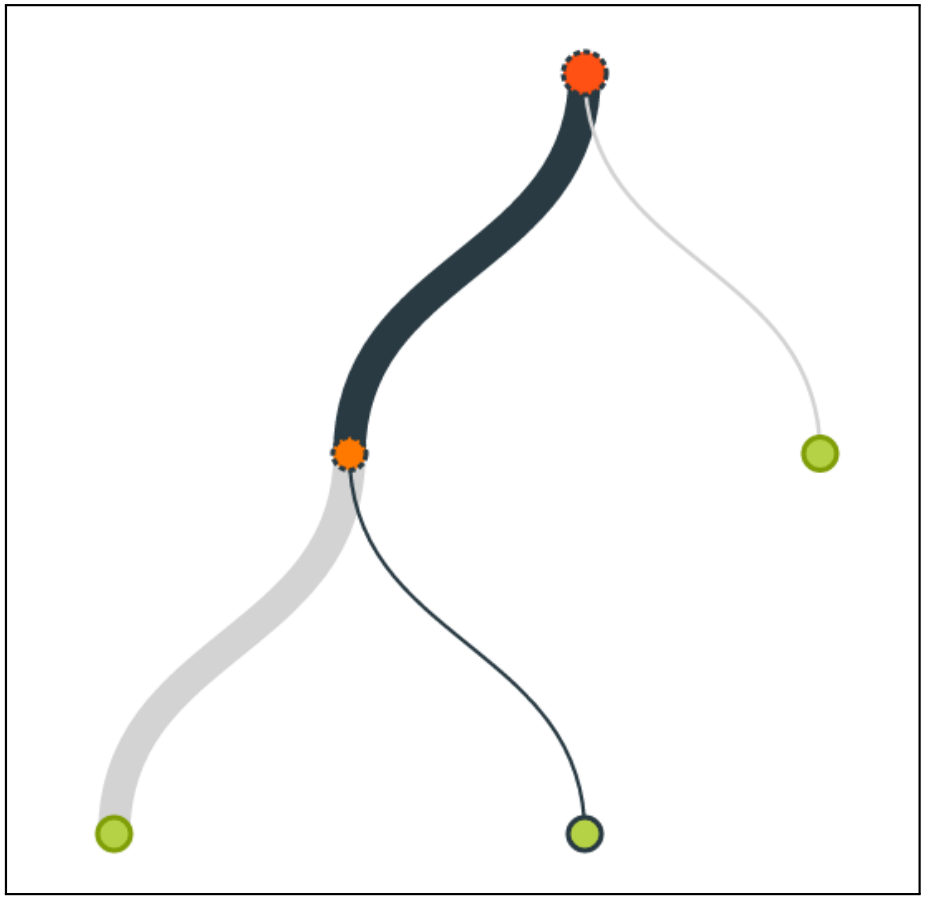
The Decision Tree

- Why stop at two?
- Suggests a recursive algorithm:
  - Choose a threshold
  - Partition the data into subsets
  - Choose a threshold for the subsets, and so on
- This is a *decision tree*



Visualizing a Decision Tree

- Each node in the tree represents a *threshold*; a question asked of the data
- Branches leading from the node indicate the two or more paths data could take
- Thickness of the branches indicates the amount of training data that took that path
- At the *leaf* or *terminal* node, we decide on a prediction; usually the majority class for the subset at that leaf.



When to Stop?

- How do we know when to stop splitting the data?
- Several possible criteria:
  - When the subset is totally pure (Note: Subsets of size 1 are trivially pure)
  - When the size reaches a predetermined minimum
  - When the number of nodes or tree depth is too large
  - When you can't get any *statistically significant* improvement
- Nodes that don't meet the latter criteria can be removed after tree construction via *pruning*

Predicting Using a Decision Tree

- The process of building the tree from the training data is called *learning*, *induction*, or *training*.
- The payoff of using the tree for prediction is *inference*, *prediction*, or *scoring*.
- To do prediction with a tree, given an instance:
  - Start at the root node, subjecting the instance to the threshold
  - "Send" the instance down the appropriate path, to the next threshold
  - Continue until you reach a terminal node, then make a prediction

Outline

Prediction Confidence - Intuition

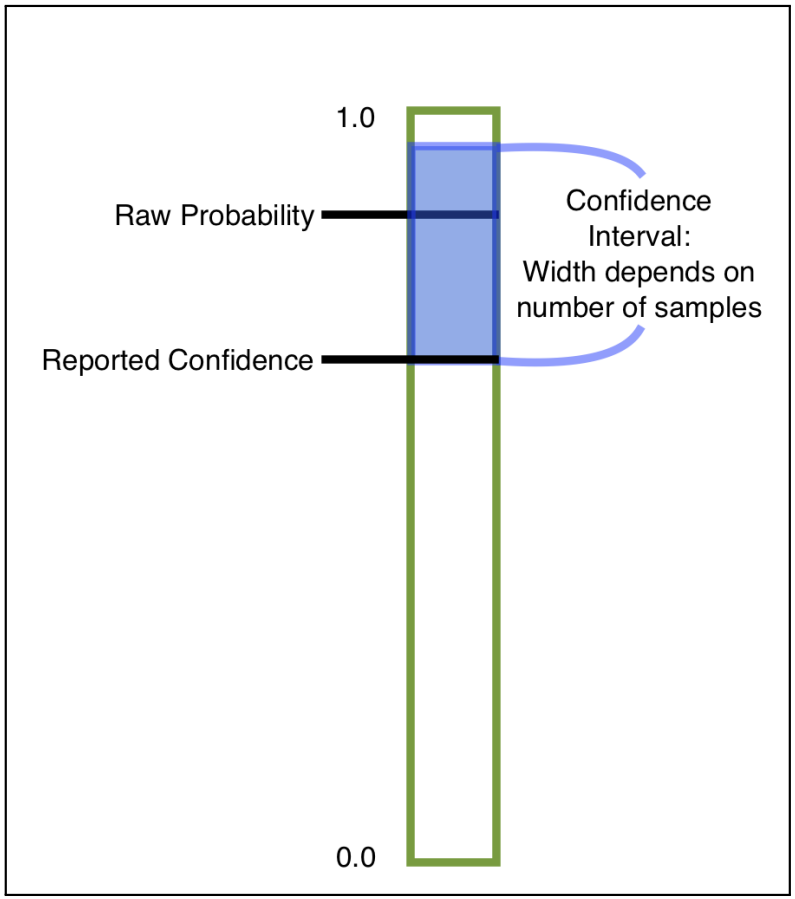
- Each terminal node return a prediction . . .
- . . . but are some terminal nodes “better” than others?
- Given a terminal node, how “confident” are we (perhaps for ranking)?
- Depends on two things:
  - The number of instances that reach that terminal node
  - The purity of that subset
- Think of each terminal node as a “mixture of experts”

Prediction Confidence - Calculation

- Compute a 95% confidence bound about the probability implied by the terminal node subset
- The Wilson score interval:

$$\frac{\hat{p} + \frac{1}{2n}z^2 \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{1}{n}z^2}$$

- Use the lower bound as the confidence
- Similarish for regression models

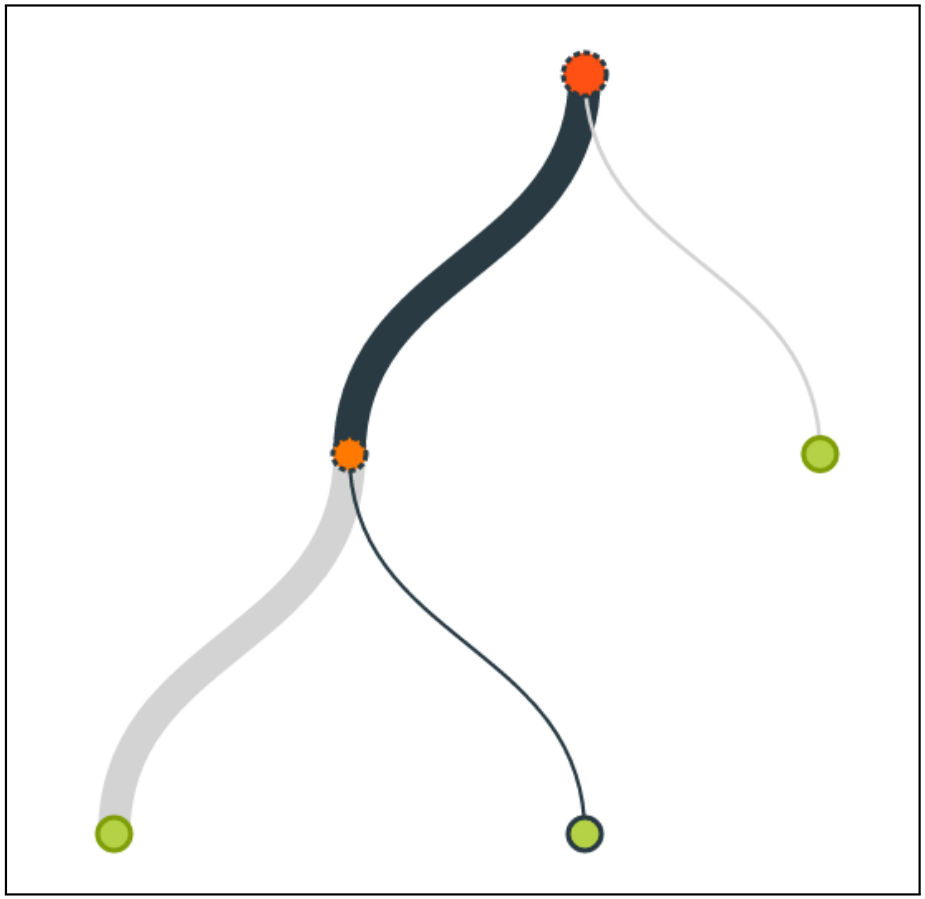


Field Importance - Intuition

- Which fields drive the model predictions the most?
- Why might we want to know?
  - Improve data collection practices
  - Reduce the number of features in your model
  - Model validation (Watch out for data leakage: objective in the training data)
- Depends on two things:
  - Number of times the node uses that feature for the threshold
  - How much it “matters” when it’s used

Field Importance - Calculation

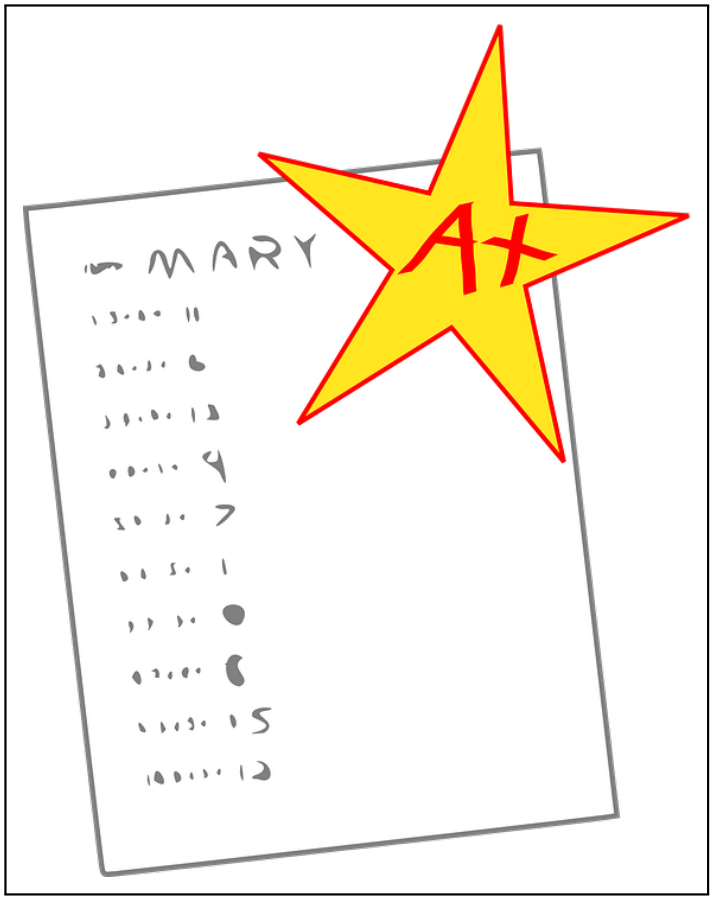
- For each node in the tree:
  - Compute the error of the prediction in the subset at that node (before)
  - Compute the error of the two child subsets (after)
  - Compute the improvement (the difference between the two)
- Sum up the improvements for all nodes corresponding to a given feature



Outline

Evaluating Your Model

- You want to know if your model is good before putting it in production
- In your data, you know the right answer!
- So use the data to test how often your model is right



Don’t Evaluate on the Training Data!

- *Never* train and test on the same data
- Many non-parametric models are able to *memorize* the training data
- This results in evaluations that are sometimes wildly optimistic (which is the worst)
- Solution: Split your data into training and testing subsets (80% and 20% is customary)

Watch out For Luck!

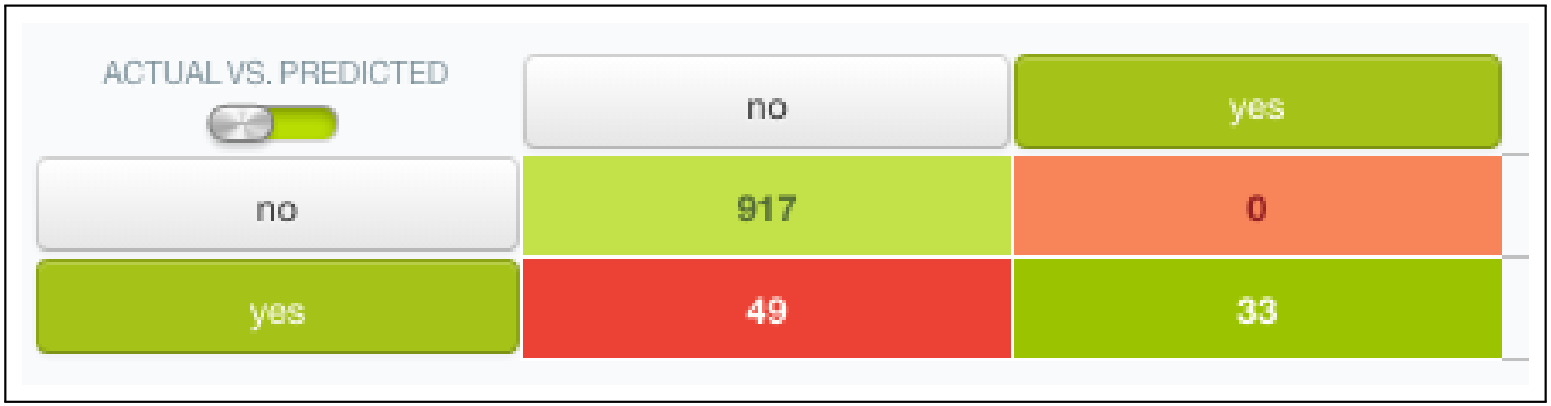
- Even splitting your data is sometimes not enough
- You may draw an “easy” subset (especially possible with small data)
- Can be disastrous if the “edge” you’re expecting is small
- Solution: Do multiple runs and average (or cross-validate)

Accuracy Can Be Misleading

- The top-line evaluation number is *accuracy*; the number of correct predictions divided by the number of total predictions
- Accuracy can be misleading!
  - Suppose only 5% of my customers churn every month
  - Suppose I just predict “no churn” for everybody
  - Ta-da! 95% accuracy!
  - But I get the same 95% accuracy if predict all “churn” instances correct with a 50% false positive rate!
- Lesson: Accuracy is broken when classes are unbalanced

The Confusion Matrix

- What to do if accuracy isn't what you want?
- Look at the specific types of mistakes the algorithm makes
- This is the *confusion matrix*



ACTUAL VS. PREDICTED	no	yes
no	917	0
yes	49	33

Aside: What If You Don't Like it?

- You see your confusion matrix and it's making "the wrong mistakes" (maybe you can even confirm by looking at the model)
- Often this is caused by *unbalanced data*; data where one class dominates the dataset
- You suspect that a trade-off could be made
- Using weights is a good way to do this
- A good place to start is often balancing the objective

But What if You Want *Just One Number*?

- For model selection, you usually want a single number for a direct comparison
- There are other functions of the confusion matrix besides accuracy
- BigML provides a few of these:
  - Φ-coefficient (common in soft sciences)
  - F1-score (information retrieval, where there's a rare positive class)
  - Balanced Accuracy (accuracy, averaged class-wise by inverse frequency)

The Right Thing is Always The Right Thing

- None of these things are better in general than any other
- In your domain, it might be best to come up with your own function
- A common form is the *cost matrix*: How much does a mistake of each type cost:

$$\text{Total cost} = c_{\text{FP}} n_{\text{FP}} + c_{\text{FN}} n_{\text{FN}}$$

  where c_x is the cost of a mistake of type x (false positive or false negative), and n_x is the number of mistakes of that type.
- This is an important part of what you, the domain expert, bring to the table! Don't ignore it!

Regression Evaluation: Generally Less Fraught

- To evaluate regression models, we first measure how much error we would have if we just predicted the mean objective for the whole dataset
- This functions as a baseline
- How much (as a fraction) of the baseline error does the model eliminate?
  - 1.0 is best
  - 0.0 is not good at all
  - And of course, we can go negative
- This is R^2
- What is "good?" Again, this is domain-specific, but at least improvement with R^2 is usually fairly monotonic

Fin!

Questions?