Source:

---

1 | Validation

We have visualized out models and used human judgment to, well, judge them. We have *not* done this algorithmically or mathematically.

Why?

Some things are blind to the human eye. Eg. underfitting and overfitting.

Not enough data, the algorithm was buggy (can't we see these though? maybe just not as easily?)

Underfitting

Wrong algorithm, buggy, or the data just sucks / there isn't actually a correlation.

Overfitting

Training *to well* to our dateset, making it not applicable to the real world / other data.

Bias-Variance Tradeoff

Bias - off Variance - inconsistent

We want low bias low variance (doih).

Holdout? nah, let's cross validate!

Like holdout, but you do it multiple times with different chunks of data 'held out'

Validation?

What do? - Accuracy - Easy, but not super effective / informative. - Precision, Recall, F-measure - True positive, false negative, and all the permutations. - Precision =

$$\frac{TP}{TP + FP}$$

- Recall

```
NUM_INPUTS = 50 # inputs per class
PLANT_A_AVG_HEIGHT = 60.0
PLANT_A_AVG_WIDTH = 8.0
PLANT_B_AVG_HEIGHT = 59.0
PLANT_B_AVG_WIDTH = 10.0
PLANT_C_AVG_HEIGHT = 70.0
PLANT_C_AVG_WIDTH = 15.0
```

```
# Pick numbers randomly with a normal distribution centered around the averages

plant_a_heights = numpy.random.normal(loc=PLANT_A_AVG_HEIGHT, size=NUM_INPUTS)
plant_a_widths = numpy.random.normal(loc=PLANT_A_AVG_WIDTH, size=NUM_INPUTS)

plant_b_heights = numpy.random.normal(loc=PLANT_B_AVG_HEIGHT, size=NUM_INPUTS)
plant_b_widths = numpy.random.normal(loc=PLANT_B_AVG_WIDTH, size=NUM_INPUTS)

plant_c_heights = numpy.random.normal(loc=PLANT_C_AVG_HEIGHT, size=NUM_INPUTS)
plant_c_widths = numpy.random.normal(loc=PLANT_C_AVG_WIDTH, size=NUM_INPUTS)

# this creates a 2-dimensional matrix, with heights in the first column and widths in the second
# the first half of rows are all plants of type a and the second half are type b
plant_inputs = list(zip(numpy.append(plant_a_heights, plant_b_heights),
                        numpy.append(plant_a_widths, plant_b_widths),
                        numpy.append(plant_a_heights, plant_b_heights)))

# this is a list where the first half are 0s (representing plants of type a) and the second half are 1s
classes =  +  +

# Generate some new random values for two plants, one of each class
new_a_height = numpy.random.normal(loc=PLANT_A_AVG_HEIGHT)
new_a_width = numpy.random.normal(loc=PLANT_A_AVG_WIDTH)
new_b_height = numpy.random.normal(loc=PLANT_B_AVG_HEIGHT)
new_b_width = numpy.random.normal(loc=PLANT_B_AVG_WIDTH)

# Pull the values into a matrix, because that is what the predict function wants
inputs = , new_a_width], , new_b_width]]

# Print out the outputs for these new inputs
print('Plant A: {0} {1}'.format(new_a_height, new_a_width))
print('Plant B: {0} {1}'.format(new_b_height, new_b_width))
print('Class predictions: {0}'.format(model.predict(inputs))) # guess which class
print('Probabilities:\n{0}'.format(model.predict_proba(inputs))) # give probability of each class
```