Introduction to Machine Learning

Evaluating Model Performance

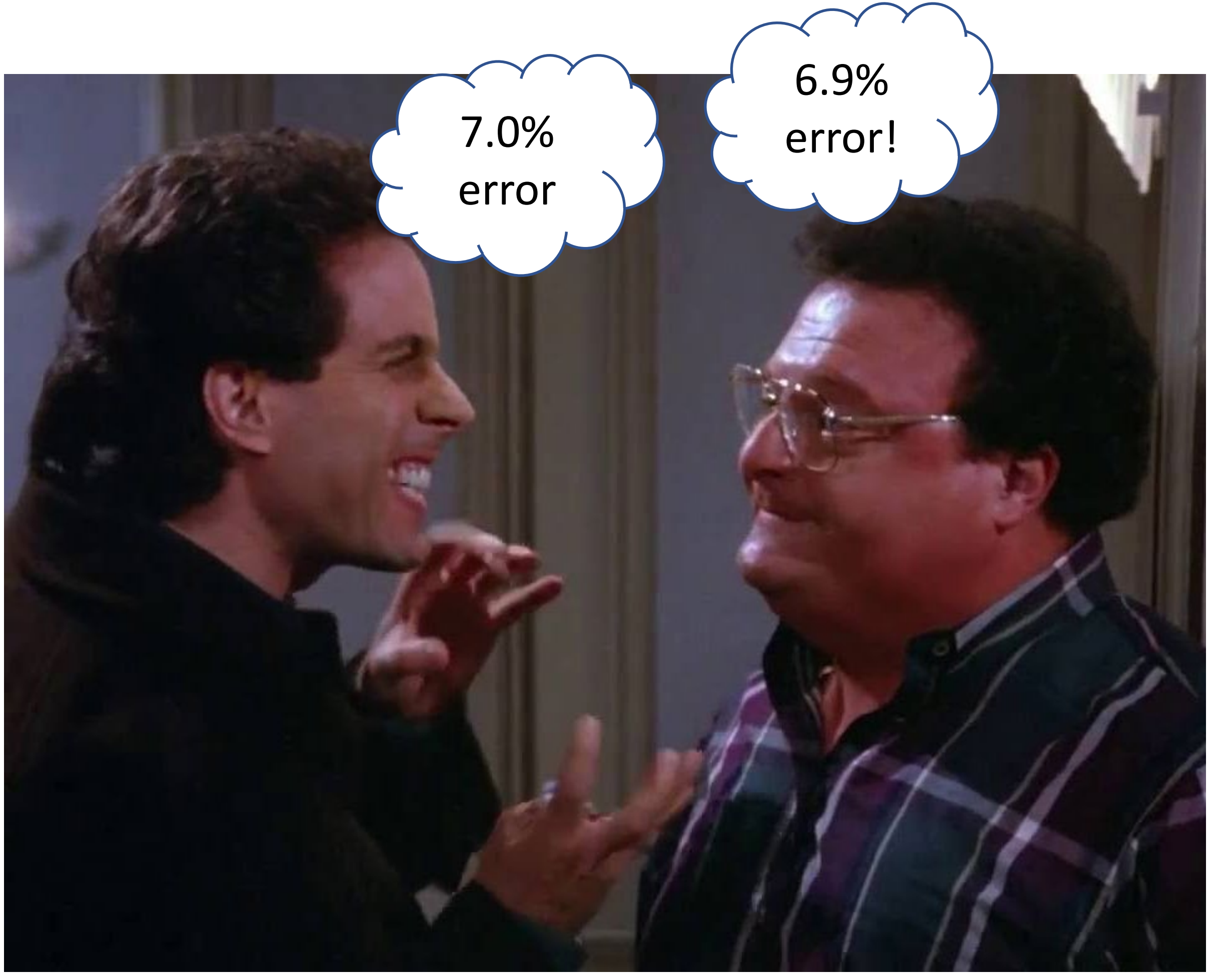
(Part 2)

Confusion matrix

predicted → real ↓	Class_pos	Class_neg	
Class_pos	TP	FN	P = TP+FN
Class_neg	FP	TN	N = TN+FP

- Accuracy is % correct (fraction correct)
- Accuracy = (TP + TN)/(P+N)

x1	√
x2	√
x3	√
x4	X
x5	√
x6	X
x8	X
x9	X
x10	√
x11	√
…	…
x1000	√



x1	√
x2	√
x3	√
x4	X
x5	√
x6	X
x8	X
x9	***√***
x10	√
x11	√
…	…
x1000	√

x1	√
x2	√
x3	√
x4	X
x5	√
x6	X
x8	X
x9	X
x10	√
x11	√
…	…
x1000	√

x1	√
x2	X
x3	√
x4	√
x5	√
x6	√
x8	X
x9	√
x10	X
x11	√
…	…
x1000	√

Hypothesis: Newman’s algorithm is better than mine

- Null hypothesis: Newman’s algorithm is not better than mine

Determine whether difference is statistically significant
(not just due to random luck)

T-test

- Compute p-value
- Probability that observed difference was luck

“There is a 95% chance this difference was not by chance”

T-test

- Calculate sample mean (population mean)
- Degrees of freedom = n-1

T-test

- Error of algorithm A is $a_1, .., a_N$
- Error of algorithm B is $b_1, .., b_N$
- Center data points around means μ_a and μ_b
  - Each a_i is now $\hat{a}_i = a - \mu_a$

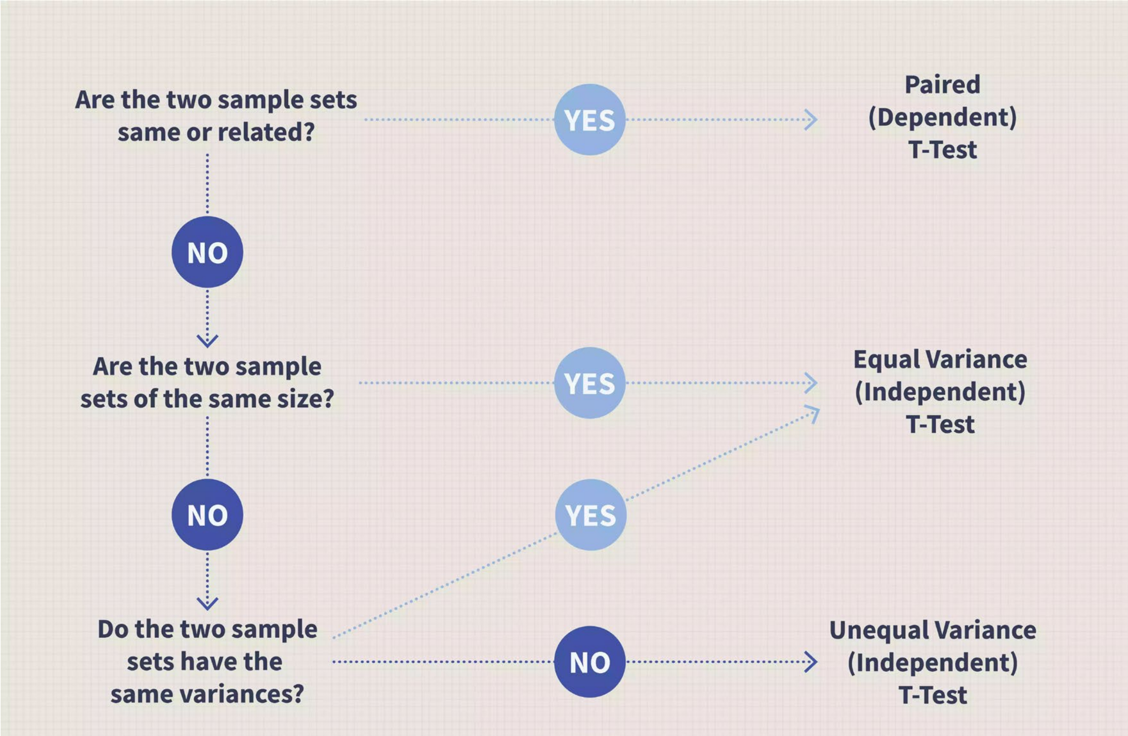
T-test

- Error of algorithm A is $a_1, .., a_N$
- Error of algorithm B is $b_1, .., b_N$
- Center data points around means μ_a and μ_b
    - Each a_i is now $\hat{a}_i = a - \mu_a$
    - Each b_i is now $\hat{b}_i = b - \mu_b$

$$t = (\mu_a - \mu_b)\sqrt{\frac{N(N-1)}{\sum_n(\hat{a}_n - \hat{b}_n)^2}}$$

t	significance
≥ 1.28	90.0%
≥ 1.64	95.0%
≥ 1.96	97.5%
≥ 2.58	99.5%

One of many such tests



Calculating p-value from t statistic

- t distribution
- We care about values away from the mean
  - Mean +/- t
- Look up p value based on t statistic and degrees of freedom (= N-1)
- t table
  - Here for two tailed
  - https://www.medcalc.org/manual/t-distribution.php

Example

Data	A	B
1	3	20
2	3	13
3	3	13
4	12	20
5	15	29
6	16	32
7	17	23
8	19	20
9	23	25
10	24	15
11	32	30

$$t = (\mu_a - \mu_b)\sqrt{\frac{N(N-1)}{\sum_n (\hat{a}_n - \hat{b}_n)^2}}$$

Generate p value

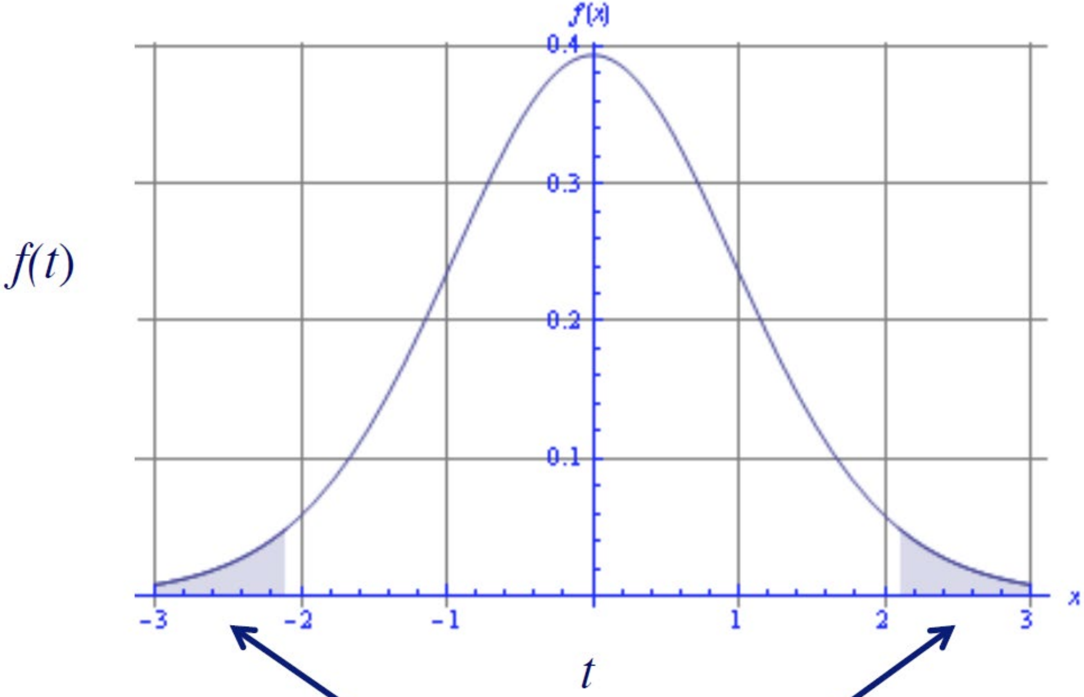
Data	A	B
1	3	20
2	3	13
3	3	13
4	12	20
5	15	29
6	16	32
7	17	23
8	19	20
9	23	25
10	24	15
11	32	30

DF	A = 0.2	0.10	0.05	0.02	0.01	0.002	0.001
∞	t_a = 1.282	1.645	1.960	2.326	2.576	3.091	3.291
1	3.078	6.314	12.706	31.821	63.656	318.289	636.578
2	1.886	2.920	4.303	6.965	9.925	22.328	31.600
3	1.638	2.353	3.182	4.541	5.841	10.214	12.924
4	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	1.476	2.015	2.571	3.365	4.032	5.894	6.869
6	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	1.363	1.796	2.201	2.718	3.106	4.025	4.437

df = N-1

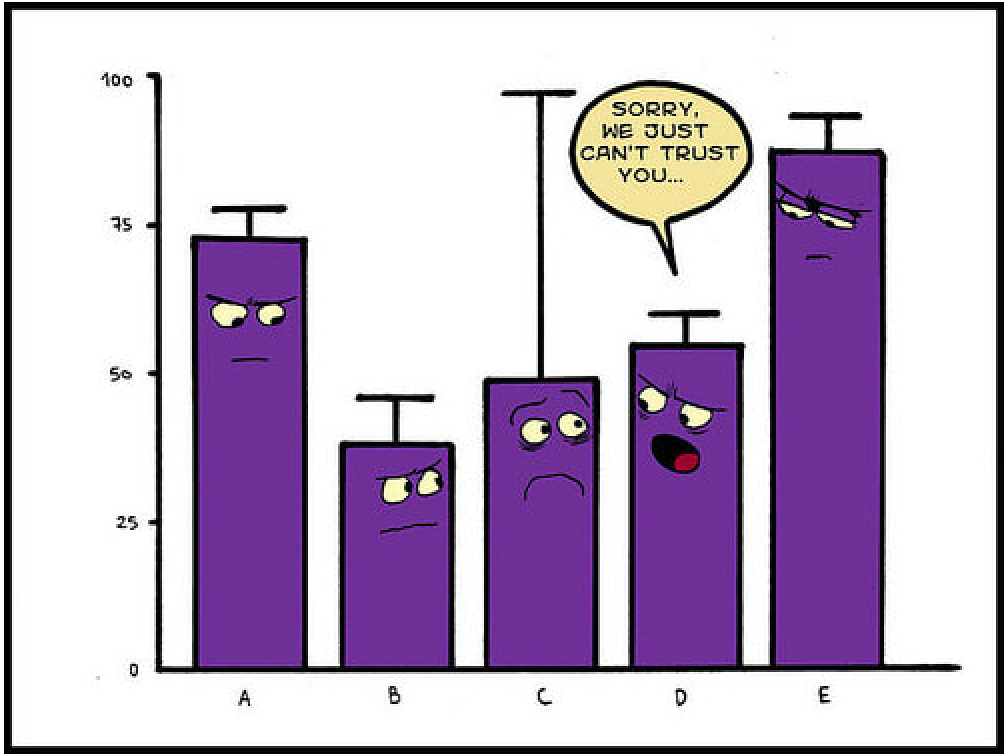
Calculated $|t|$ = 2.737

$p<.05$



for a two-tailed test, the p-value represents the probability mass in these two regions

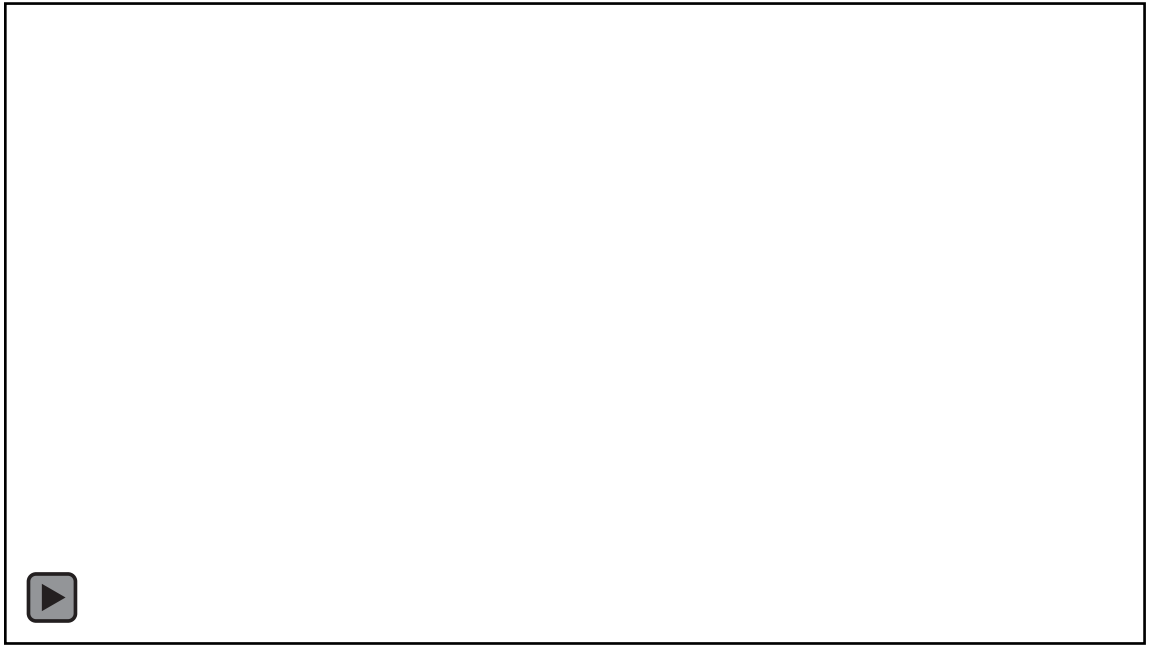

LOOK AT THIS!!
"STUDY FINDS 50% OF PEOPLE BORED BY STATISTICS."


SORRY, WE JUST CAN'T TRUST YOU...
100
75
50
25
0
A
B
C
D
E

Confidence intervals

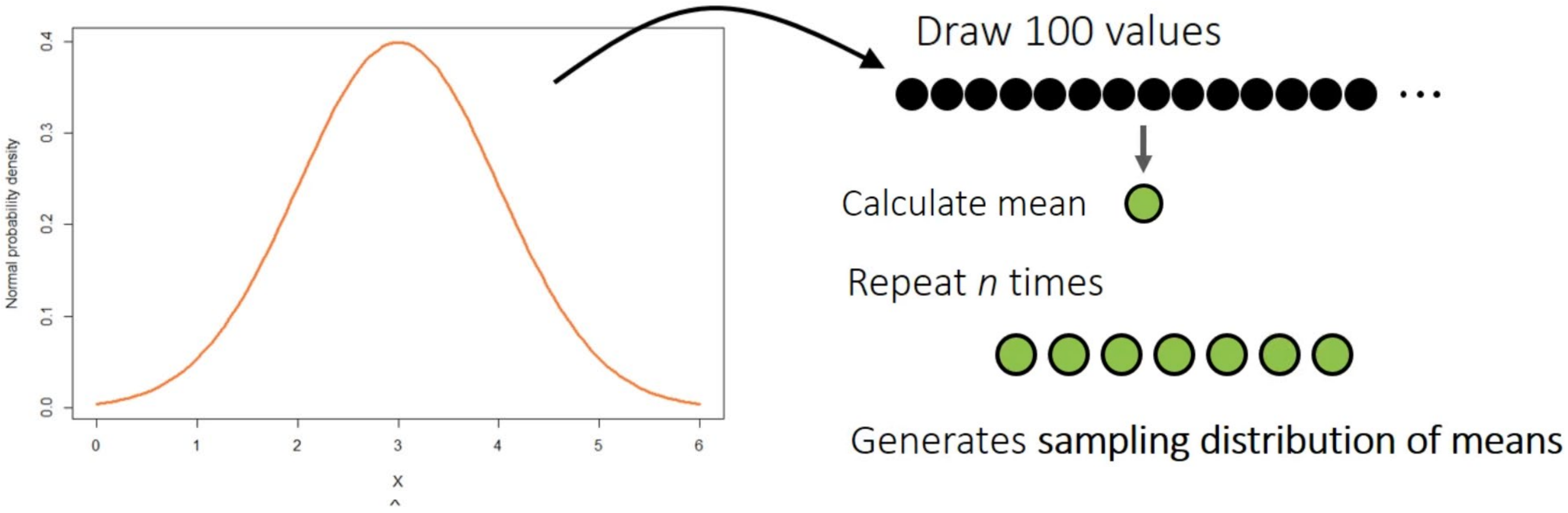
The **median**
for the population
lies between 

Confidence intervals



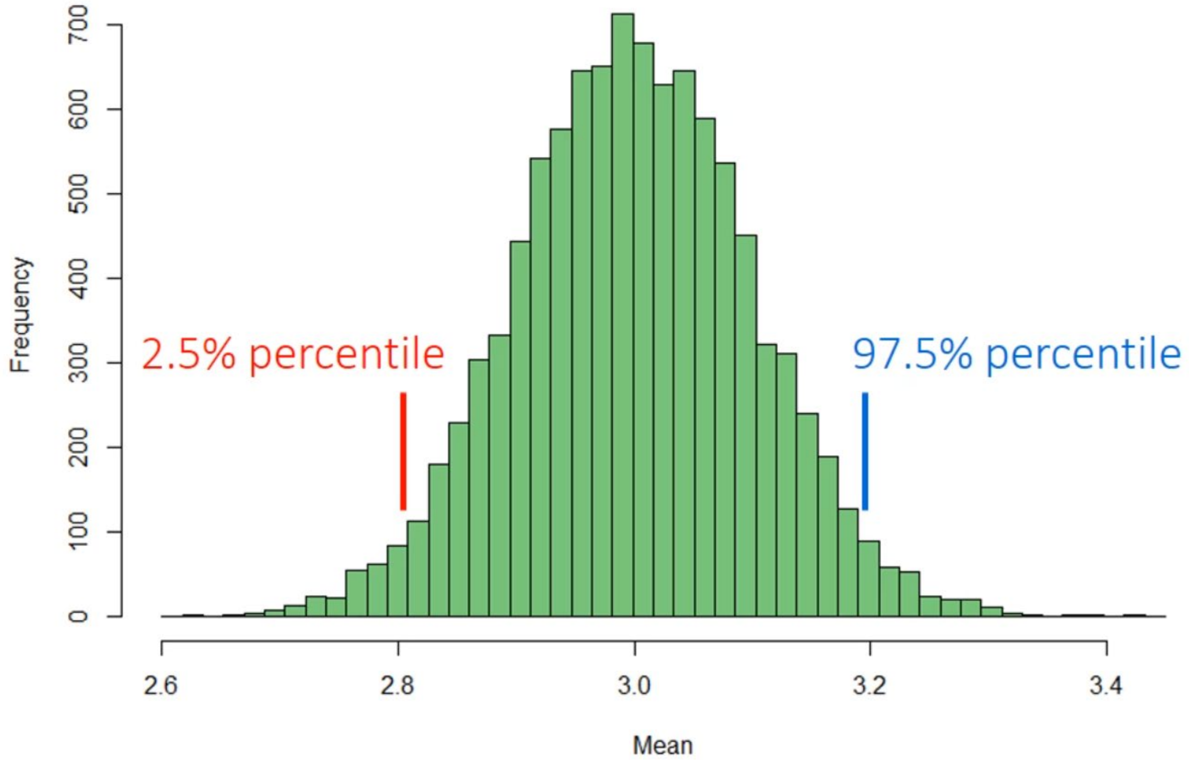
Confidence intervals

If you took many random samples from the population, 95% of the confidence intervals on those samples would include μ

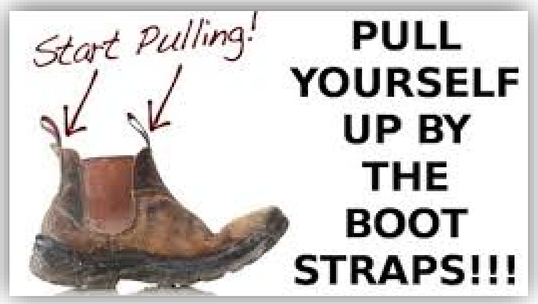


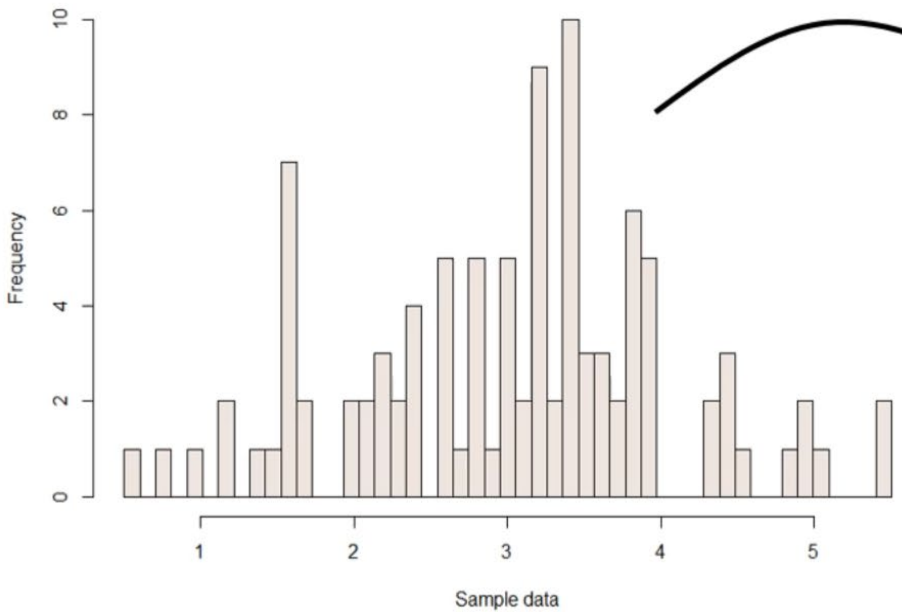
What if I do not have enough data?

Resampling distribution



Bootstrapping





Sample values **with replacement**

...

Calculate mean

Repeat *n* times

Generates **sampling distribution of means**

Algorithm 10 BootstrapEvaluate($\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, *NumFolds*)

```
scores ← [ ]
for k = 1 to NumFolds do
    truth ← [ ]                                          // list of values we want to predict
    pred ← [ ]                                           // list of values we actually predicted
    for n = 1 to N do
        m ← uniform random value from 1 to N             // sample a test point
        truth ← truth ⊕ y_m                              // add on the truth
        pred ← pred ⊕ ŷ_m                                // add on our prediction
    end for
    scores ← scores ⊕ F-SCORE(truth, pred)               // evaluate
end for
return (MEAN(scores), STDDEV(scores))
```

Why is the learning algorithm performing poorly?