Assignment 2:

1) Possible to do slightly better than decision trees for tumor, but not by much

2) Overfitting with small K, maybe (depends on noise, intrinsic dimensionality of data)

3) KNN is quite slow if distance evaluation is slow

4) In large dimensions, vectors start to look very orthogonal to one another.

5) Lower dimensionality makes it harder to overfit

6) Maybe, if you are concerned about overfitting or noise

How do the answers to 2) and 4) connect to one another? What about 2) and 6), and 4) and 5)

Agaricus-Lepiota has higher dimensionality than primary-tumor, but higher accuracy as well ?!

Quiz

1) Why is leave-one-out cross validation practical for K-NN classifiers and impractical for other settings?

2) For any bootstrap sample, what's the expected number of times any element of the original sample will be chosen?

In the limit of an infinitely large sample, what is the probability that any given element of the sample is selected?

Probability that a sample isn't selected in one of the n draws: $\frac{n-1}{n}$

Probability it isn't selected in any:

$$\lim_{n\to\infty} \left(\frac{n-1}{n}\right)^n = \left(1-\frac{1}{n}\right)^n = e^{-1}$$

Probability it is selected at least once: $1-\frac{1}{e} \approx 0.63$

Precision, Recall, Sensitivity, Oh my!

$$Acc = \frac{A+D}{A+B+C+D}$$

Confusion matrix

	$\hat{y}=0$	$\hat{y}=1$
$y=0$	A	B
$y=1$	C	D

$T = C+D$

$S = B+D$

$I = D$

$\neg I = A$

$\neg S = A+C$

$$\text{Precision} = \frac{I}{S} = \frac{D}{B+D}$$

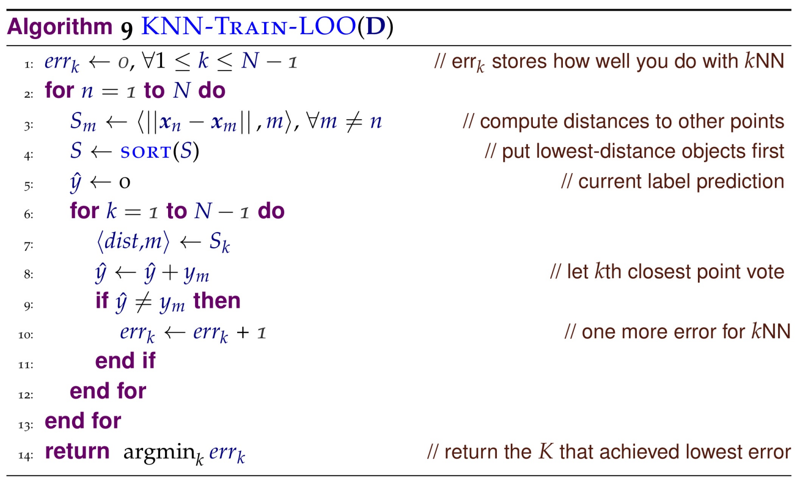
$$\text{Recall} = \frac{I}{T} = \frac{D}{C+D}$$

$$\text{Sensitivity} = \text{Recall} = \frac{D}{C+D}$$

$$\text{Specificity} = \frac{\neg I}{\neg S} = \frac{A}{A+C}$$

Cross Validation

- Validation data is used inefficiently
  - Let's partition training data into K subsets, then create K training-validation splits
  - The average of the validation error is a better estimate of test error
  - "K-fold cross validation"
  - Use this to decide on hyperparameters
- What if K increases to ∞?
  - leave-one-out CV
  - Only useful when not horribly inefficient
    - KNN



Algorithm 9 KNN-Train-LOO(**D**)

```
err_k ← 0, ∀1 ≤ k ≤ N − 1                        // err_k stores how well you do with kNN
for n = 1 to N do
    S_m ← ⟨||x_n − x_m||, m⟩, ∀m ≠ n               // compute distances to other points
    S ← SORT(S)                                    // put lowest-distance objects first
    ŷ ← 0                                          // current label prediction
    for k = 1 to N − 1 do
        ⟨dist,m⟩ ← S_k
        ŷ ← ŷ + y_m                                // let kth closest point vote
        if ŷ ≠ y_m then
            err_k ← err_k + 1                      // one more error for kNN
        end if
    end for
end for
return argmin_k err_k                              // return the K that achieved lowest error
```

Bootstrap Resampling (Statistical Testing)

- When is 8% really greater than 7.9% ?
  - fundamental new idea : error estimation matters

  - Null hypothesis testing asks :

    - How likely is it that we saw this diffeence because of chance ?

    - If we could repeat the experiment infinitel many times, how often would we have made a mistake ?

- Instead of reporting "8% on our 'population'", we report (an estimate of) the distribtion by rerunning our experiment on (many estimates of, resamples of) the population

```
Algorithm 10 BootstrapEvaluate($y$, $\hat{y}$, 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 $\oplus$ $y_m$                    // add on the truth
        pred ← pred $\oplus$ $\hat{y}_m$                // add on our prediction
    end for
    scores ← scores $\oplus$ F-SCORE(truth, pred)       // evaluate
end for
return (MEAN(scores), STDDEV(scores))
```