NEAREST NEIGHBOR CLASSIFICATION

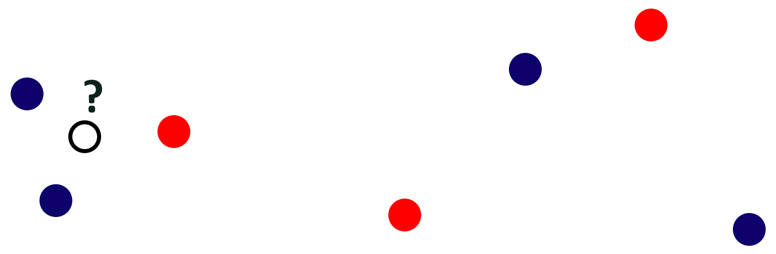
Classification

- Definition: assign an object to one of several predefined categories

- Given:
  - A set of predefined classes
  - A number of attributes
  - A learning set
- Goal:
  - Predict the class of unclassified data

Nearest Neighbor Classification

- Find training examples that are relatively similar to the test example
- The k-nearest neighbor of a data point refers to the k neighbors closest to it.
- The data point is classified based on these neighbors.
- If different neighbors have different labels, the majority label is picked. In case of a tie, a class may be chosen randomly



Algorithm

D: Training set

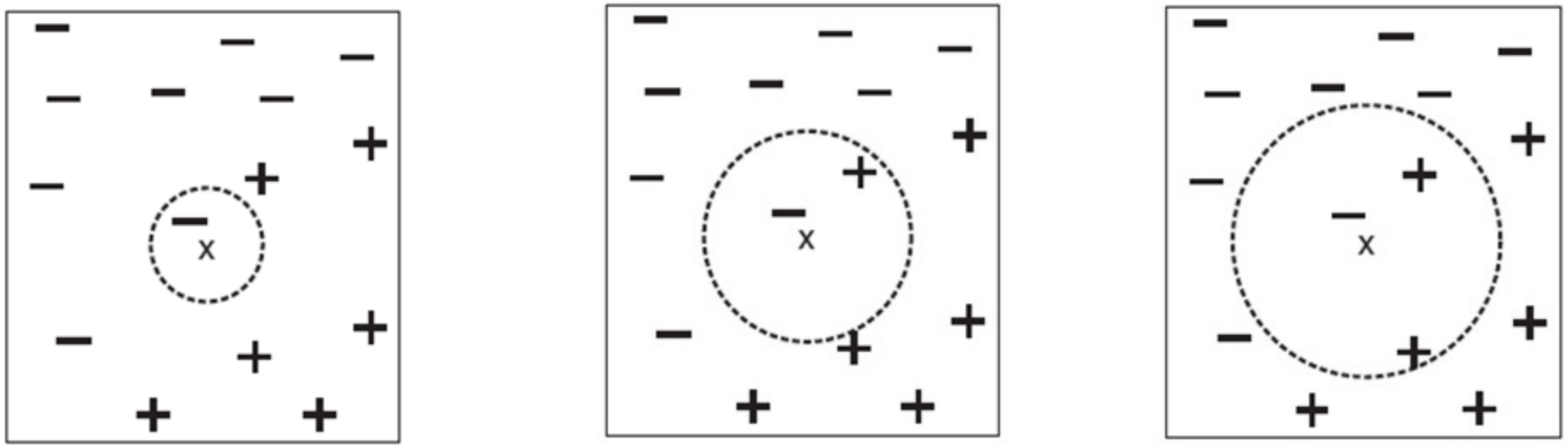
For each test example z(x', y')

- $\boxed{\text{Compute } d(x', x)}$: the distance between x' and every x in D
- Select D_z subset of D containing $\boxed{k}$ closest example
- Select the class label with the largest vote count in D_z

Algorithm

- Need a distance measure:
  - Normalize
  - Choose distance measure? Categorical attributes?
  - Missing values: numeric/categorical?
    - Assume maximum possible distance
- Need to choose k

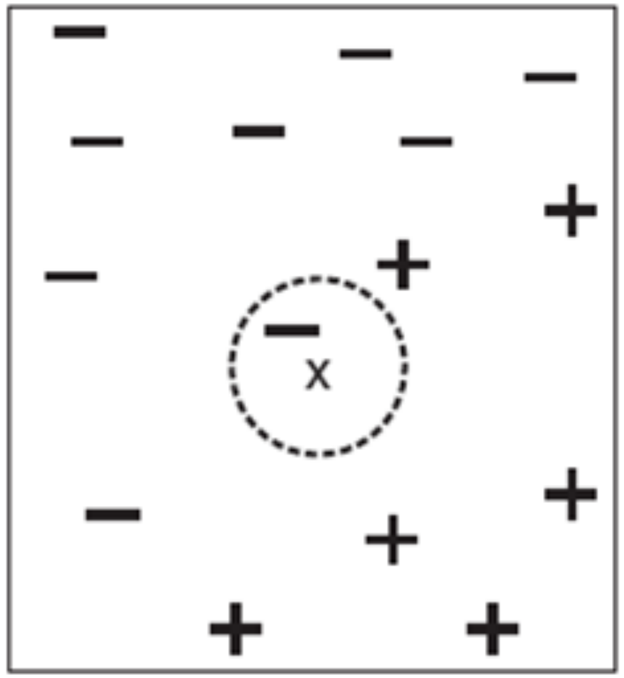
Nearest Neighbor Classification



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

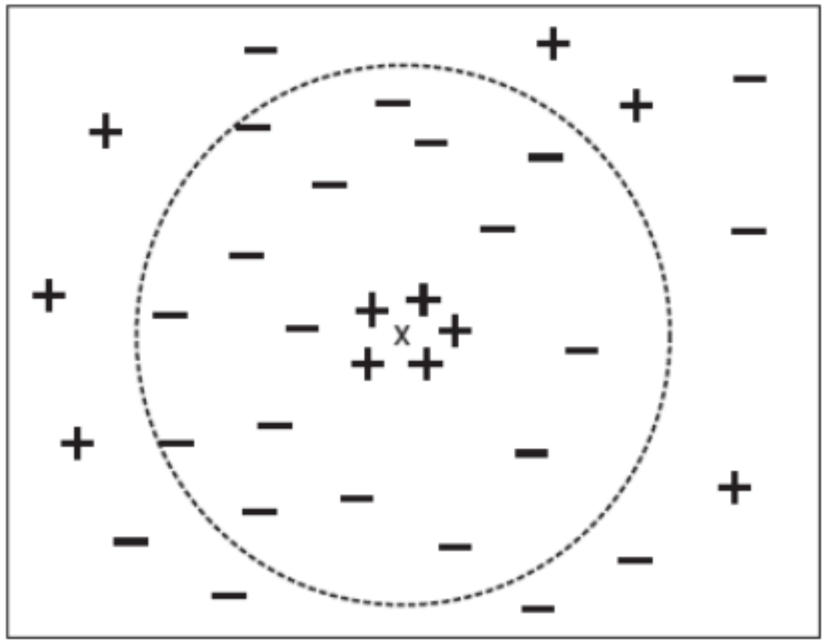
What value of k is good?

- If k is too small: risk of overfitting increases, susceptible to noise



What value of k is good?

- If k is too large: risk of wrong classification since neighbors list may contain unrelated data



What value of k is good?

What happens if k = N?

Common Approaches:

k = sqrt(N)

Cross validation

Label Selection

- Majority Voting:

$$y' = \arg\max_{v} \sum_{(x_i, y_i) \in D} I(v = y_i)$$

Same impact from all points

- Distance Weighted Voting:

$$y' = \arg\max_{v} \sum_{(x_i, y_i) \in D_z} w_i \times I(v = y_i)$$

$$w_i = \frac{1}{d(x', x_i)^2}$$

Nearby nodes have higher impact than farther nodes

Characteristics

Advantages

- Simple and intuitive
- Make predictions without maintaining a model, lazy learners
- Make prediction based on local information

Disadvantages

- It can be **expensive to classify** a data point, distances must be computed between the test data point and every training point
- May produce arbitrary decision boundaries
- Dependent on proximity measure
- Choosing best k may be difficult

Example: Riding lawn mower owners

Owners

Household Number	Income (in thousands)	Thousands Sq. Feet
1	60.0	18.4
2	85.5	16.8
3	64.8	21.6
4	61.5	20.8
5	87.0	23.6
6	110.1	19.2
7	108.0	17.6
8	82.8	22.4
9	69.0	20.0
10	93.0	20.8
11	51.0	22.0
12	81.0	20.0

Nonowners

Household Number	Income (in thousands)	Thousands Sq. Feet
13	75.0	19.6
14	52.8	20.8
15	64.8	17.2
16	43.2	20.4
17	84.0	17.6
18	49.2	17.6
19	59.4	16.0
20	66.0	18.4
21	47.4	16.4
22	33.0	18.8
23	51.0	14.0
24	63.0	14.8

Example: Riding lawn mower owners

- New household
  - $60,000 income
  - 20,000 sq. feet
- Normalize?
- Without normalization:
  - 1-NN:
    - #1 (owner)
  - 3-NN:
    - #1, #4, #19 (Majority is owner)
  - 5-NN:
    - #1, #4, #19, #3, #15 (Majority is owner)
  - 7-NN:
    - #1, #4, #19, #3, #15,#24,#20 (Majority is no-owner)

Household	Distance	Household	Distance
1	1.60	13	15.01
2	25.70	14	7.24
3	5.06	15	5.56
4	1.70	16	16.80
5	27.24	17	24.12
6	50.11	18	11.06
7	48.06	19	4.04
8	22.93	20	6.21
9	9.00	21	13.10
10	33.01	22	27.03
11	9.22	23	10.82
12	21.00	24	6.00

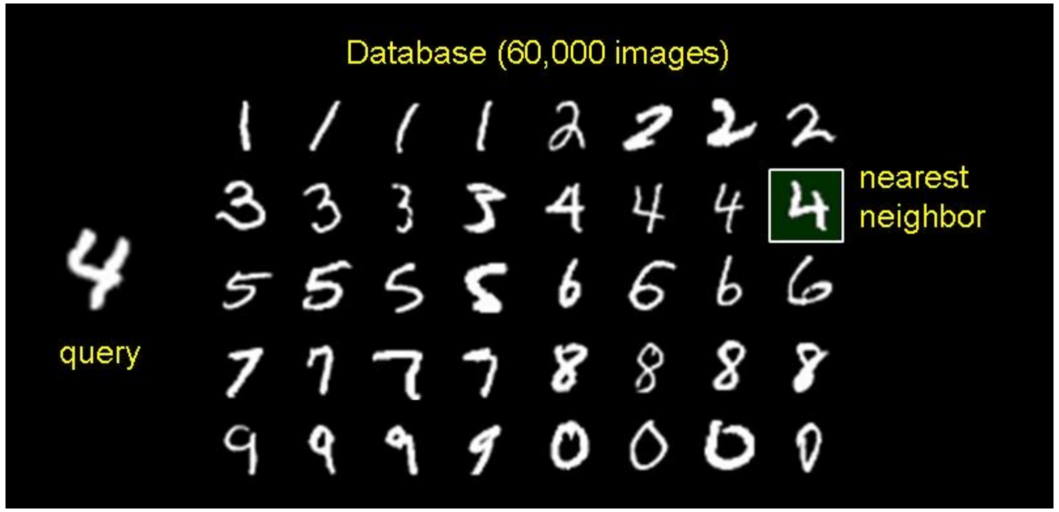
Example: Riding lawn mower owners

- New household
  - $60,000 income
  - 20,000 sq. feet
- With normalization:
  - 1-NN:
    - #4 (owner)
  - 3-NN:
    - #4, #9, #14
    - Majority is owner
  - 5-NN:
    - #4, #9, #14, #1, #3
    - Majority is owner
  - 7-NN:
    - #4, #9, #14, #1, #3, #20, #13
    - Majority is owner

Household	Distance		Household	Distance
1	0.67		13	0.79
2	1.88		14	0.50
3	0.72		15	1.20
4	0.35		16	0.88
5	2.06		17	1.60
6	2.61		18	1.15
7	2.68		19	1.68
8	1.55		20	0.74
9	0.46		21	1.65
10	1.74		22	1.48
11	0.96		23	2.57
12	1.08		24	2.19

Applications

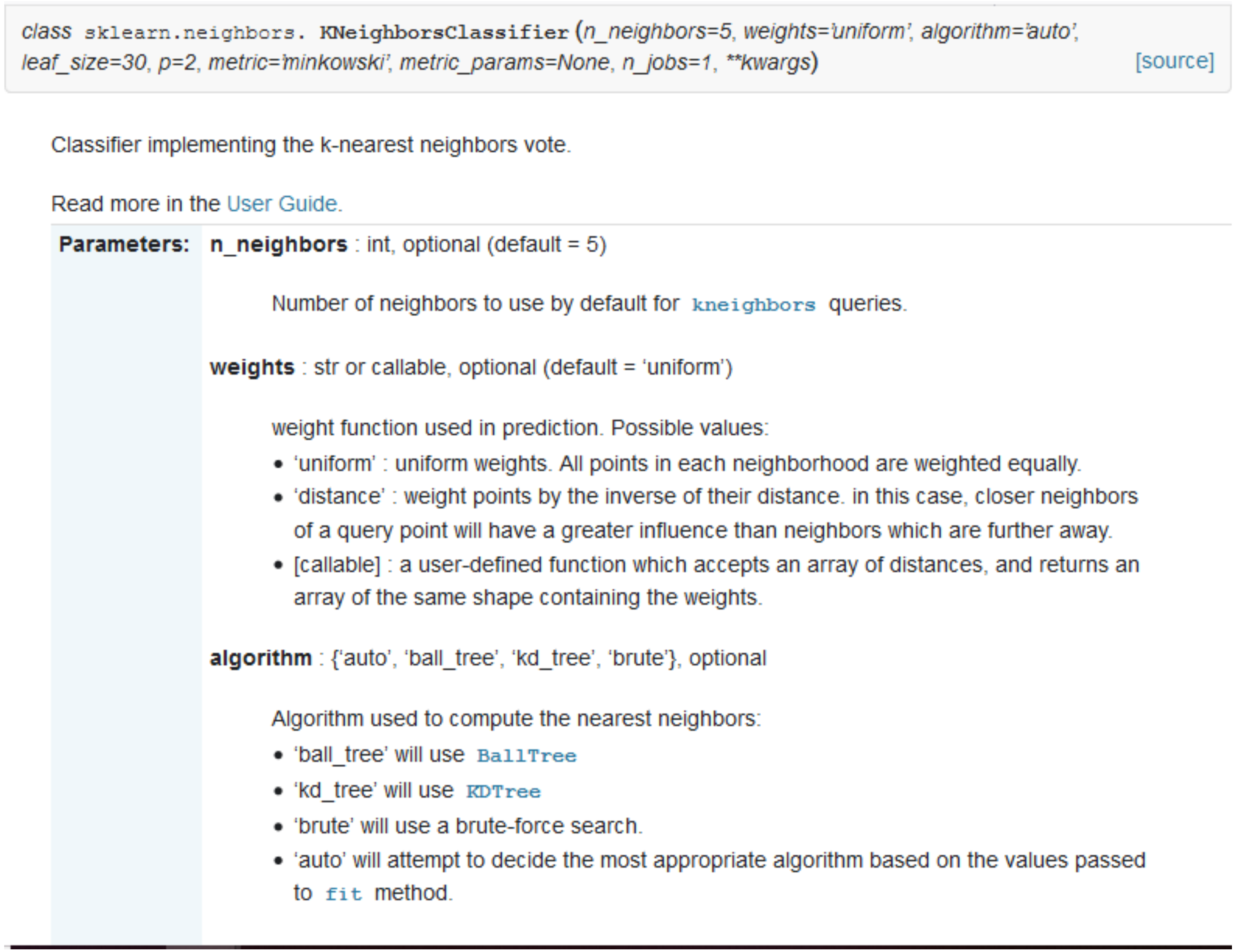
- Pattern recognition



- Predicting churn
  (if a customer will renew their contract)
- Predicting if a potential customer will respond to an offer

Implementation

- from sklearn import neighbors
- clf = neighbors.KNeighborsClassifier()
- Variables:
  - **n_neighbors** - number of neighbors (odd, often prime integer)
  - **weights** – how to weight influence of each neighbor (uniform or distance)
    - distant neighbors have less weight on prediction
  - **algorithm** – brute force, kd_tree, ball_tree, auto
    - brute – find neighbors by comparing distance with all items
    - kd_tree and ball_tree are more efficient ways for finding neighbors (build tree-like structure)
    - **use default value - auto**



class sklearn.neighbors.**KNeighborsClassifier**(*n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=1, **kwargs*) [source]

Classifier implementing the k-nearest neighbors vote.

Read more in the User Guide.

Parameters:	**n_neighbors** : int, optional (default = 5) Number of neighbors to use by default for `kneighbors` queries. **weights** : str or callable, optional (default = 'uniform') weight function used in prediction. Possible values: • 'uniform' : uniform weights. All points in each neighborhood are weighted equally. • 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away. • [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights. **algorithm** : {'auto', 'ball_tree', 'kd_tree', 'brute'}, optional Algorithm used to compute the nearest neighbors: • 'ball_tree' will use `BallTree` • 'kd_tree' will use `KDTree` • 'brute' will use a brute-force search. • 'auto' will attempt to decide the most appropriate algorithm based on the values passed to `fit` method.

http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

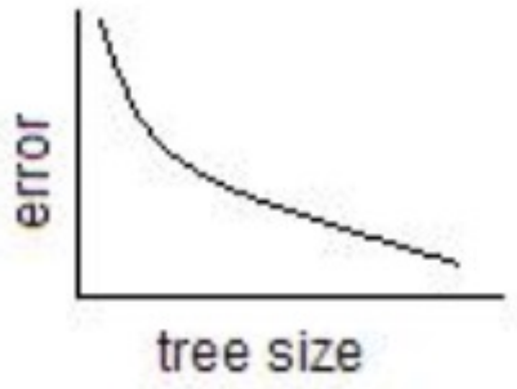
REFRESHING CLASSIFICATION

Performance evaluation

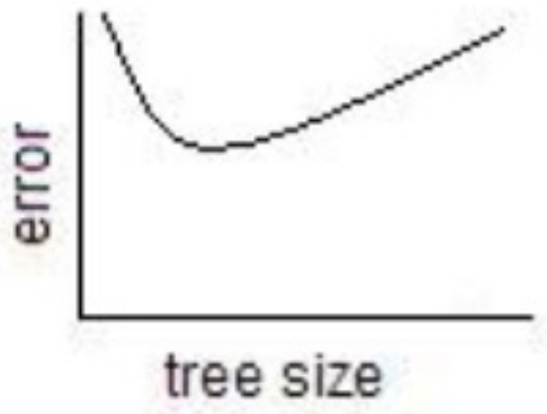
- Training errors: number of misclassified records in the training set
- Generalization errors: the expected error of the model on previously unseen records

- Goal: reduce both training errors AND generalization errors

Decision Tree Example



tree size vs. training error



tree size vs. testing error

Performance Evaluation

- Confusion Matrix:

		Predicted Class	
		Class = 1	Class = 0
Actual Class	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

- Accuracy: fraction of correct predictions

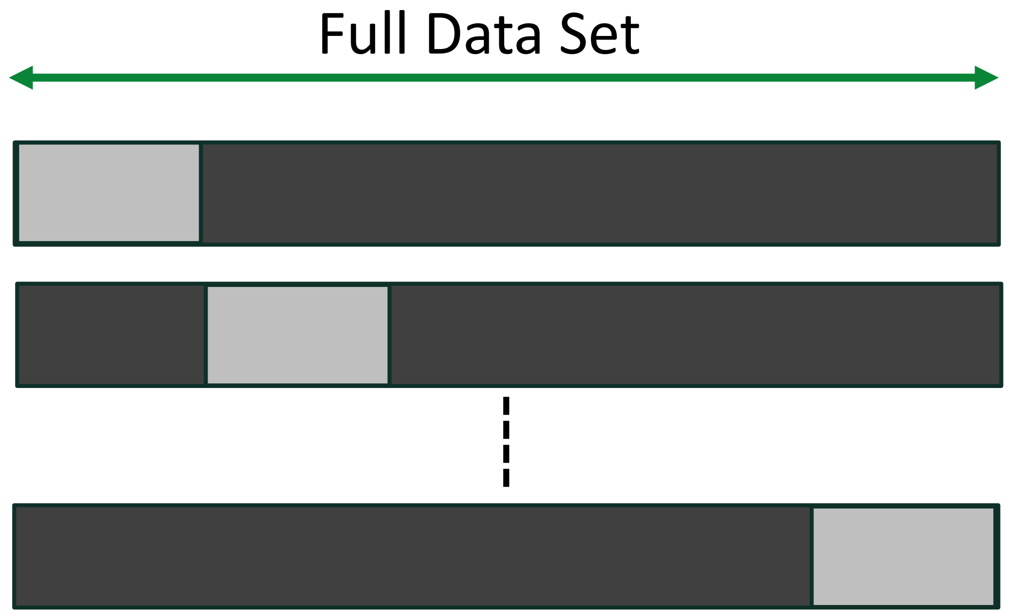
$$accuracy = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

- Error rate: fraction of wrong predictions

$$error\_rate = \frac{f_{01} + f_{10}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

K-Fold Cross Validation

- Splits the data into k disjoint sets
- In each iteration, one set is used for testing and K-1 for training



- Advantage: all records are used for both training and test

Comparing Classifiers

- Consider two models M_A and M_B
  M_A: 85% accuracy on a test set of 30 records
  M_B: 75% accuracy on a test set of 5000 records
- Which one is better?
- How much confidence does the accuracy have?
- Can we explain the difference in accuracy as a result of variations in the test sets?

Confidence Interval of Accuracy

- Task of predicting label: a binomial experiment with probability of success *p*
- If the test set contains N records, X is the number of records correctly predicted
  - X has a binomial distribution with mean: *Np* and variance: *Np(1-p)*
  - *Example:*
    - *Obtaining heads on coin toss: p = 0.5*
    - *Obtaining heads 20 times on 50 tosses:*
    - *Mean: 50*0.5 = 25*
    - *Variance: 50*0.5*0.5 = 12.5*

$$P(X = v) = \binom{N}{v} p^{v} (1-p)^{N-v}$$

$$P(X = 20) = \binom{50}{20} 0.5^{20} (1-0.5)^{50-20} = 0.0419$$

Confidence Interval of Accuracy

- Task of predicting label: a binomial experiment with probability of success p
- If the test set contains N records, X is the number of records correctly predicted
  - X has a binomial distribution with mean: Np and variance: $Np(1-p)$
  - Accuracy X/N has a <u>binomial distribution</u>

    mean = p

    var = $p(1-p)/N$
  - For large N, X/N has a <u>normal distribution</u> with mean = p and var = $p(1-p)/N$

Confidence Interval for Accuracy

Confidence intervals are constructed for given confidence levels

Example:

Confidence interval is between 82% and 88% at confidence level 95%

Meaning:

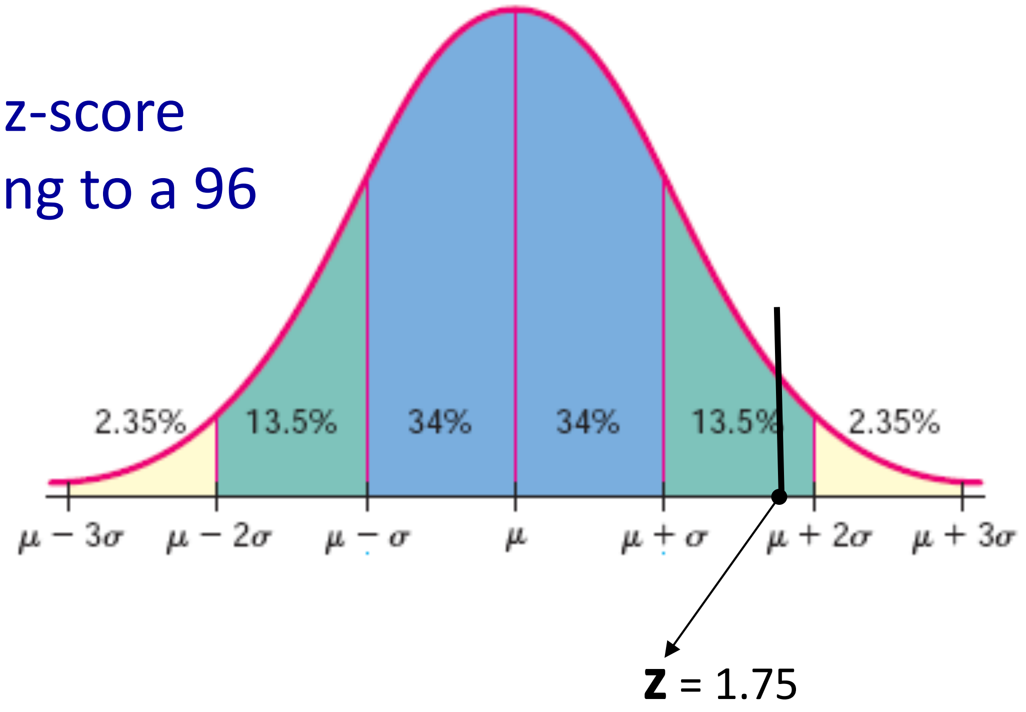
If population sample multiple times:

the result (of experiment) is within the confidence interval (82% and 88%) 95% of the times.

Z-Score

- A measure of how many standard deviations an element is below or above the mean

What is the z-score corresponding to a 96 percentile?

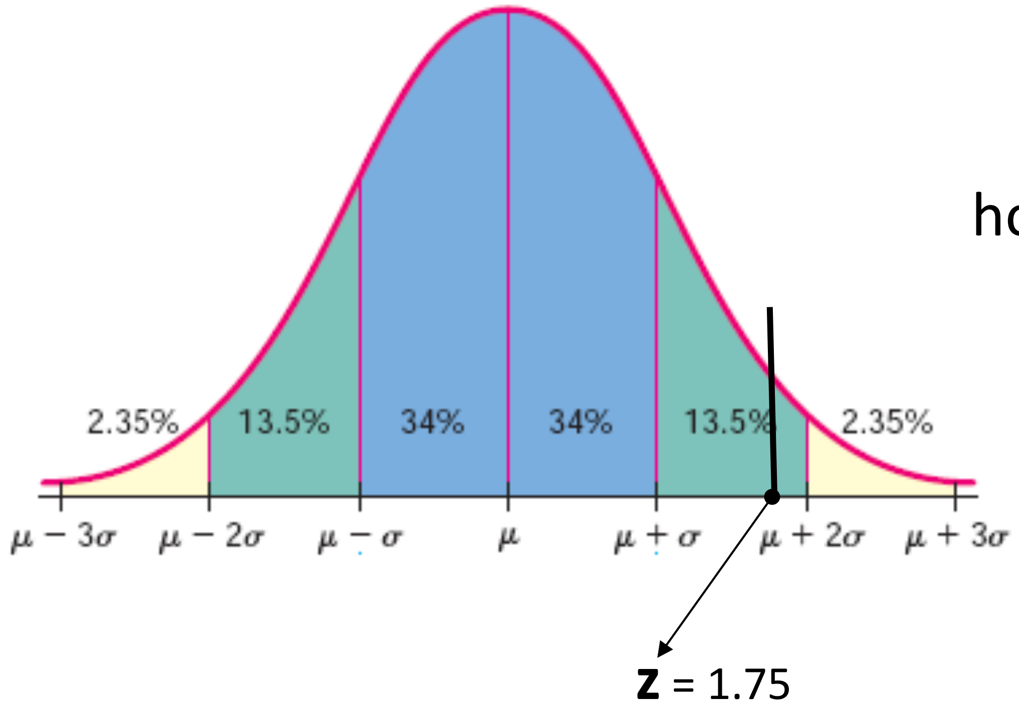


Explain: if you go 1.75 standard deviations above the mean in a normal distribution, you will have covered 96% of the data

```
from scipy.stats import norm
z_score = norm.ppf(0.96)
```

Z-Score

- A measure of how many standard deviations an element is below or above the mean



Suppose z-score is given,

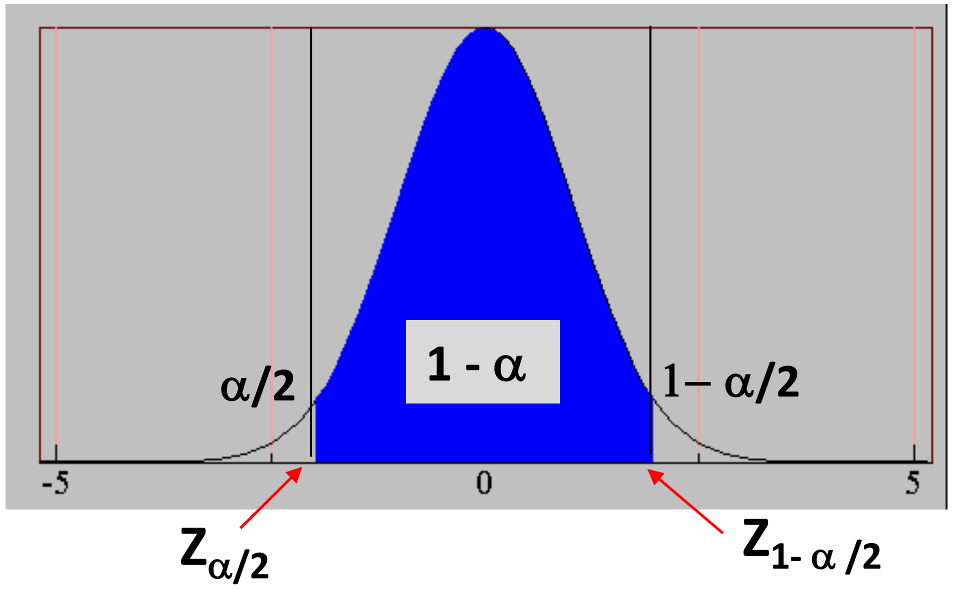
how to compute the element's value?

$$\mu + 1.75 * \sigma$$

Confidence Interval for Accuracy

- 1−α represents the confidence level. For example, if α=0.05, the confidence level is 95%.

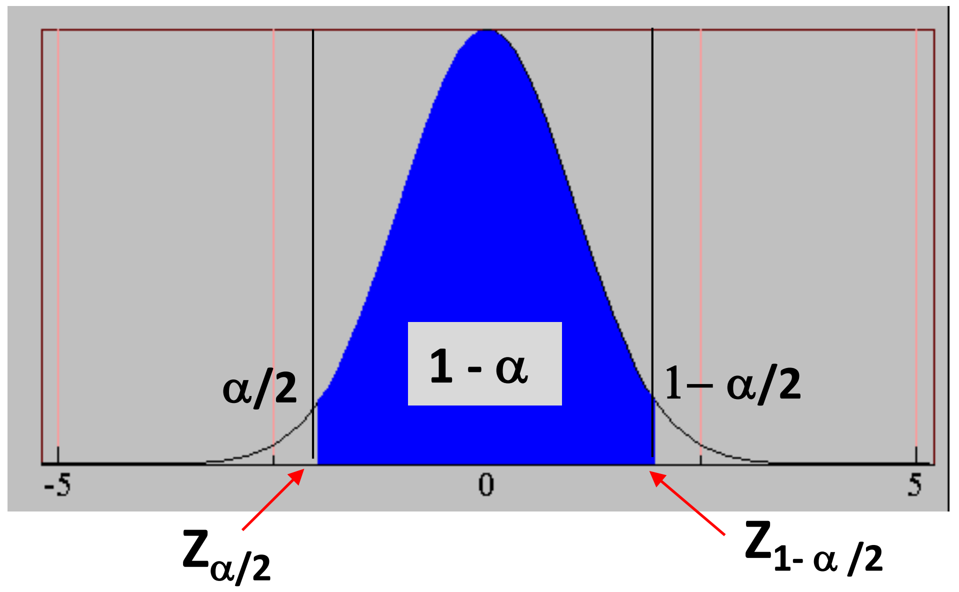
$$P\left(-Z_{\alpha/2} < \frac{acc - p}{\sqrt{p(1-p)/N}} < Z_{1-\alpha/2}\right)$$
$$= 1 - \alpha$$



$\mathbf{acc<Z_{1-\alpha/2}} \Rightarrow \text{acc}<\mu + \mathbf{Z_{1-\alpha/2}} * \sigma \Rightarrow \text{acc} < \text{p} + \mathbf{Z_{1-\alpha/2}} * \sqrt{p(1-p)/N}$

$\mathbf{acc>Z_{\alpha/2}} \Rightarrow \text{acc}>\mu - \mathbf{Z_{1-\alpha/2}} * \sigma \Rightarrow \text{acc}>\text{p} - \mathbf{Z_{1-\alpha/2}} * \sqrt{p(1-p)/N}$

Confidence Interval for Accuracy



- Given the accuracy, and an α
- Confidence Interval for acc:

$$\frac{2 \times N \times acc + Z_{\alpha/2}^2 \pm Z_{\alpha/2}\sqrt{Z_{\alpha/2}^2 + 4 \times N \times acc - 4 \times N \times acc^2}}{2(N + Z_{\alpha/2}^2)}$$

Confidence Interval for Accuracy

- Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:
  - N = 100, acc = 0.8
  - Let 1 - α = 0.95 (95% confidence)
  - From probability table, $Z\alpha/2 = 1.96$

1-a	0.99	0.98	0.95	0.9	0.8	0.7	0.5
$Z_{a/2}$	2.58	2.33	1.96	1.65	1.28	1.04	0.67

N	50	100	1000
Confidence Interval of true accuracy	67% - 88%	71% - 86%	77.4%-82.4%

Comparing performance of two models

- Given:
  - Two models M_1 and M_2
  - Evaluated on two sets D_1 and D_2 of size n_1 and n_2
  - Resulting in error rates e_1 and e_2 respectively
- Is the error difference $d = e_1 - e_2$ statistically significant?

- *Assuming n_1 and n_2 are sufficiently large, the error rates can be approximated using normal distributions*

Comparing performance of two models

- The variance of the difference can be approximated by:

$$\sigma_d^2 = e_1(1-e_1)/n_1 + e_2(1-e_2)/n_2$$

- At (1-α)%, the true difference:

$$d_t = d \pm Z_{\alpha/2}\sigma_d$$

- If the interval spans zero, then the observed difference is not significant
  - For example, d=0.1, while the interval is [−0.05,0.25]. With 0 included in the interval, the observed difference (d=0.1) is not significant.

Comparing performance of two classifiers

- Given: two classifiers L1 and L2
  - L1 generates models $M_{11}, \ldots, M_{1k}$ with errors $e_{11}, \ldots, e_{1k}$
  - L2 generates models $M_{21}, \ldots, M_{2k}$ with errors $e_{21}, \ldots, e_{2k}$
- Goal: compare performance using k-fold validation
- The jth fold error difference: $d_j = e_{1j} - e_{2j}$
- The overall variance:

$$\hat{\sigma}_t^2 = \frac{\sum_{j=1}^{k}(d_j - \overline{d})^2}{k(k-1)}$$

$$d_t = d \pm t_{1-\alpha, k-1}\hat{\sigma}_t$$

Is obtained from the t-distribution with two parameters,: the confidence level (1-a) and the degree of freedom which is k-1