Overfitting and Cross Validation

1

Dr. Yi Long (Neal)

Most contents (text or images) of course slides are from the following textbook

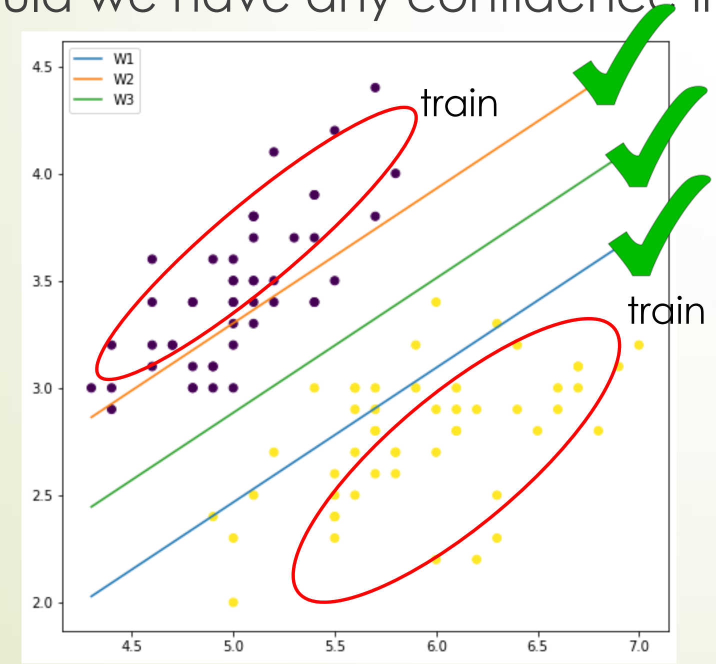
Provost, Foster, and Tom Fawcett. Data Science for Business: What you need to know about data mining and data-analytic thinking. " O'Reilly Media, Inc.", 2013

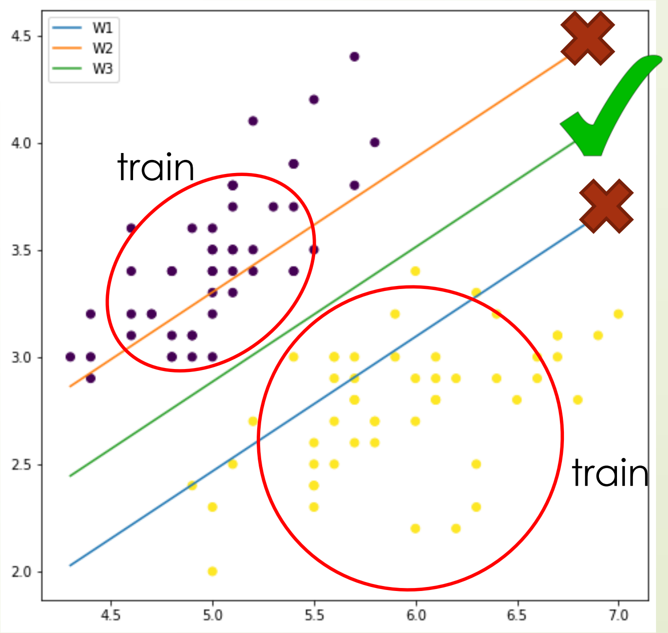
Outline

- Cross Validation
- Overfitting Avoidance
- Similarity and Distance

Drawbacks of Holdout Evaluation

- While a holdout set will indeed give us an estimate of generalization performance, it is just a single estimate.
  - ✓ A single estimate of model accuracy might have just been a single particularly lucky (or unlucky) choice of training and test data
  - ✓ Should we have any confidence in our estimation?



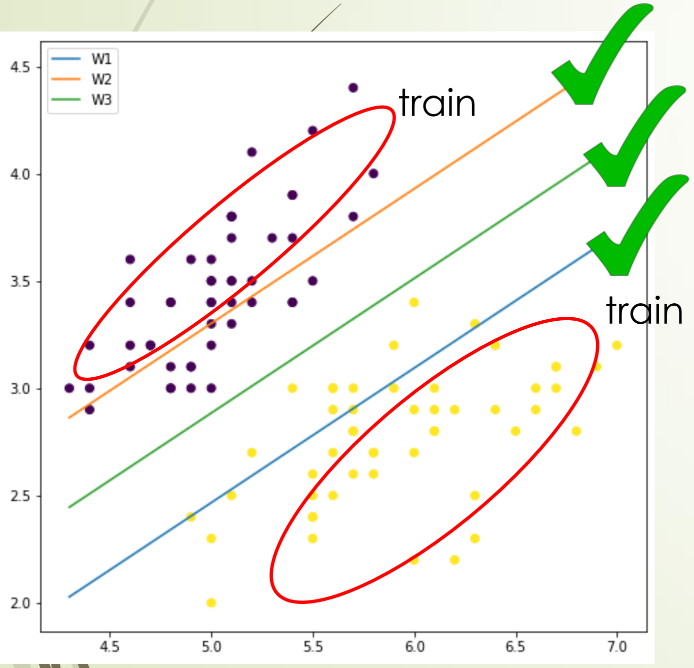


Repeated Holdout Evaluation with Randomness

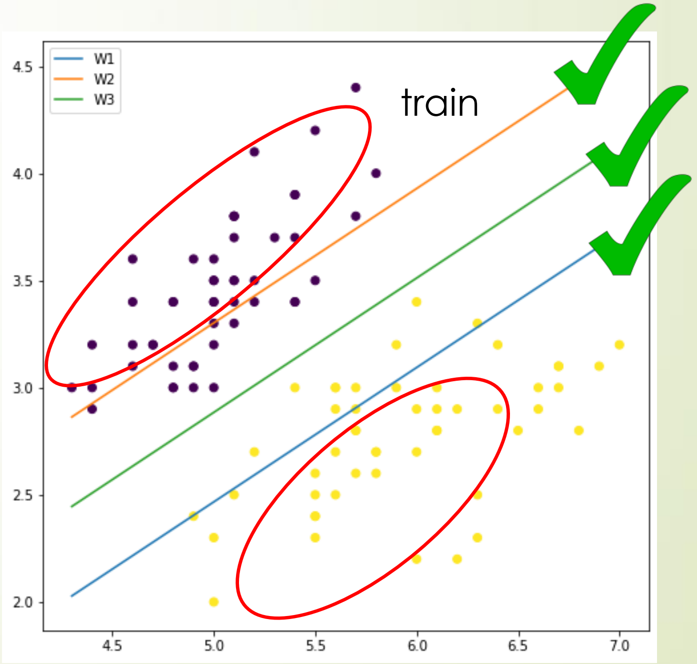
- Random splits do NOT guarantee that all splits will be different, although this is still very likely for sizeable datasets and many runs.

train_test_split():

shuffle : *boolean, optional (default=True)*
Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None.



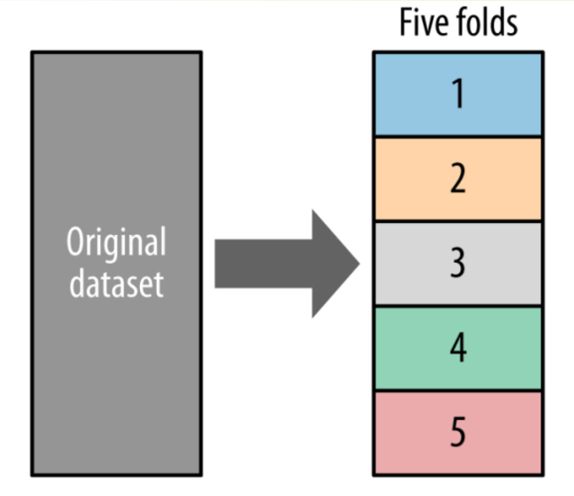
1st Run



n-th Run

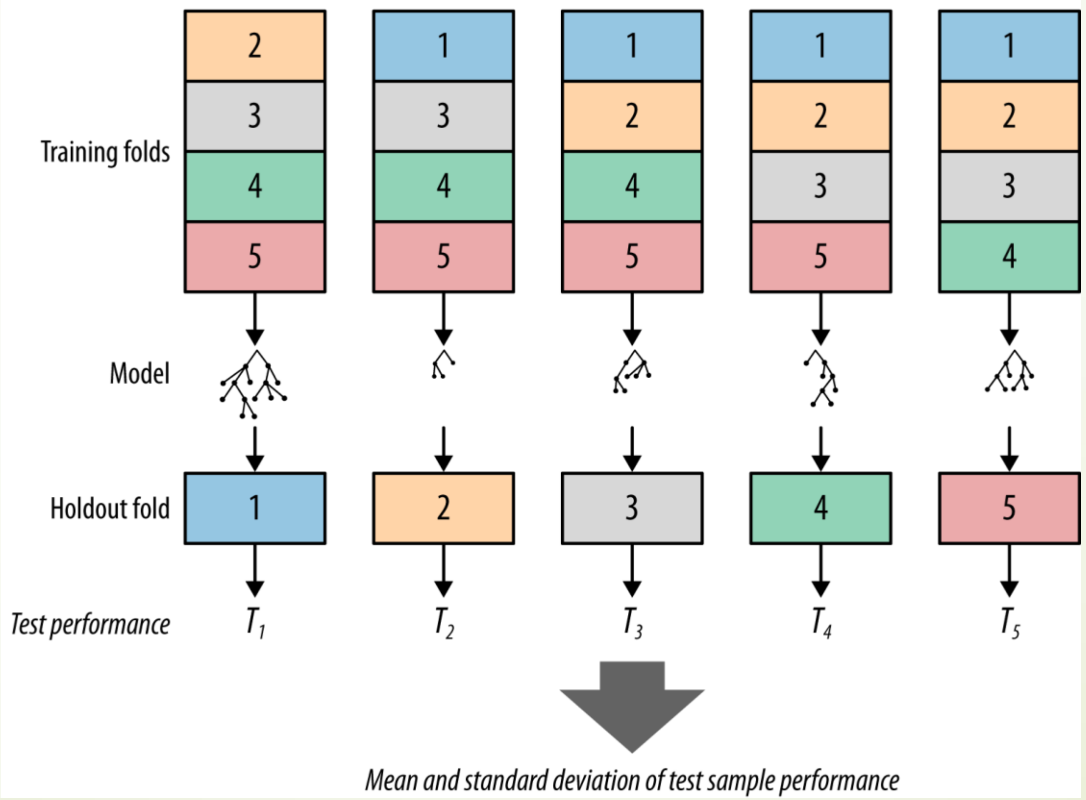
Cross-validation

- Cross-validation (CV) computes the estimation of out_performance over the whole training data by performing multiple splits (training and holdout subgroups)
  - ✓ Conduct multiple times of independent holdout test
  - ✓ Not only a simple estimate of the generalization performance, but also some statistics on the estimated performance (mean, variance)
  - ✓ Different implementations: **k-fold**, leave-*p*/1-out, Monte Carlo cross-validation
- *k*-fold cross-validation split whole training data into k partitions called *folds*
  - ✓ Iterate training and testing k times
  - ✓ In each iteration, a different fold is chosen as the holdout(test)data; and the other $k-1$ folds are combined to form the training data



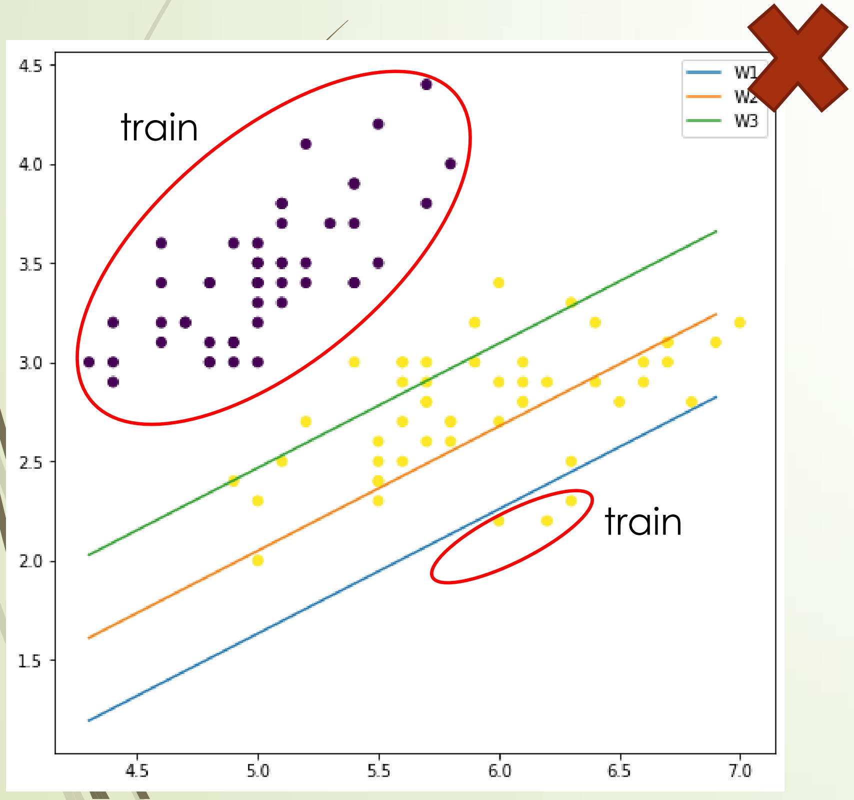
K-fold Cross-validation

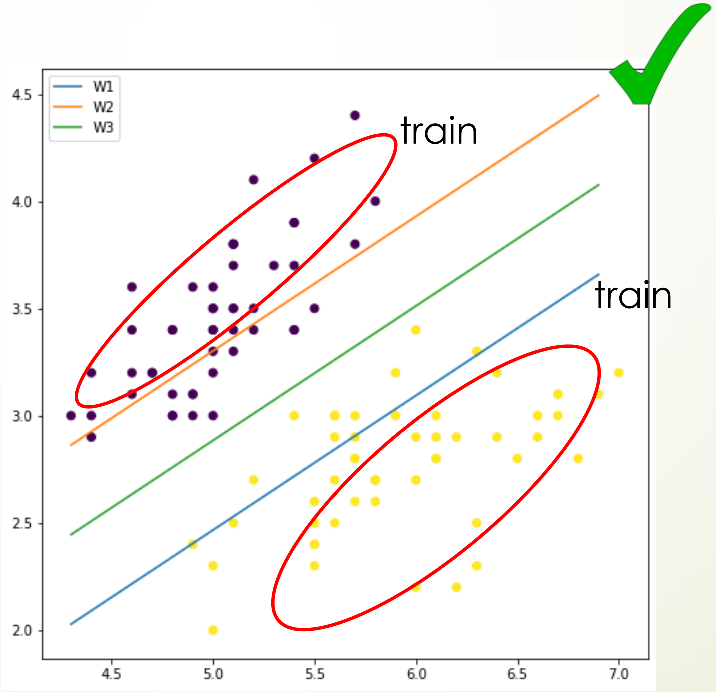
- Every example will be used once for testing but $k-1$ times for training
- Compute average and standard deviation from the evaluation results of k folds

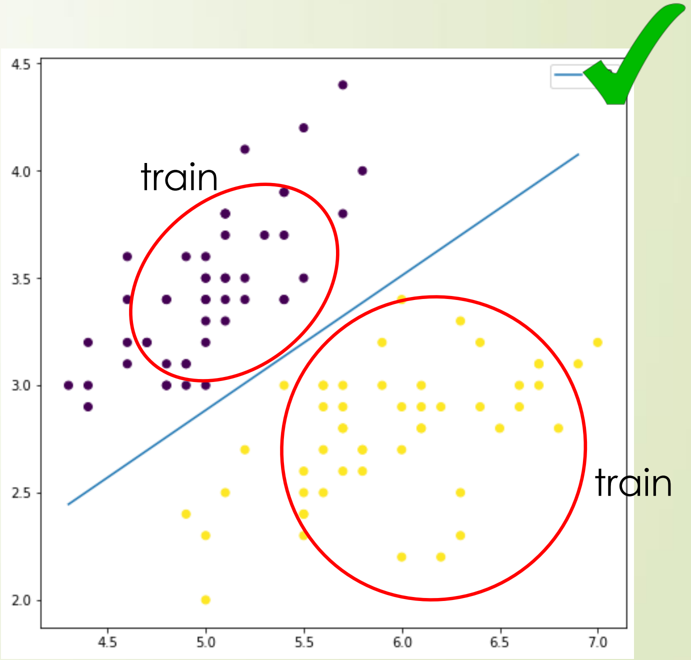


Stratified CV

- Different proportion of labels in different train-test splits
  - Stratify =True parameter makes a split preserving the percentage of samples for each class.

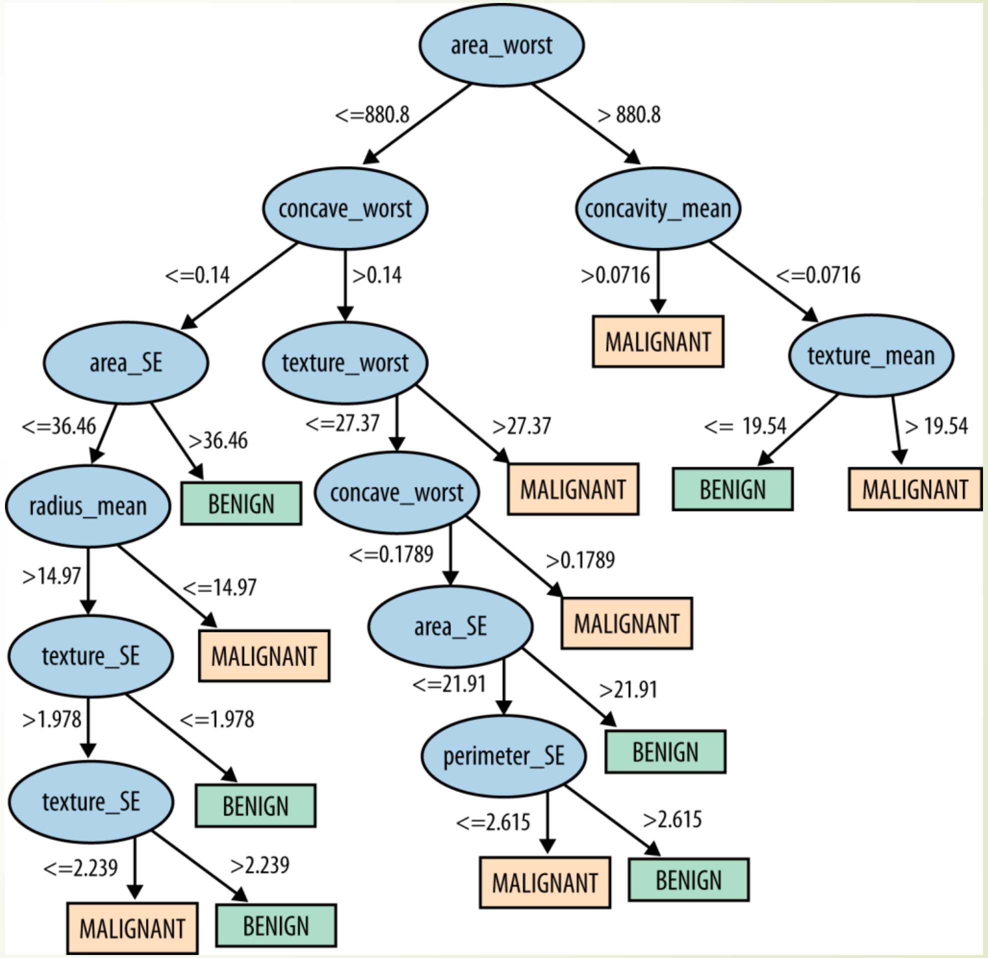






Revisit "Logistic Regression vs. Decision Tree"

Attribute	Weight (learned parameter)
SMOOTHNESS_worst	22.3
CONCAVE_mean	19.47
CONCAVE_worst	11.68
SYMMETRY_worst	4.99
CONCAVITY_worst	2.86
CONCAVITY_mean	2.34
RADIUS_worst	0.25
TEXTURE_worst	0.13
AREA_SE	0.06
TEXTURE_mean	0.03
TEXTURE_SE	−0.29
COMPACTNESS_mean	−7.1
COMPACTNESS_SE	−27.87
w_0 *(intercept)*	−17.7

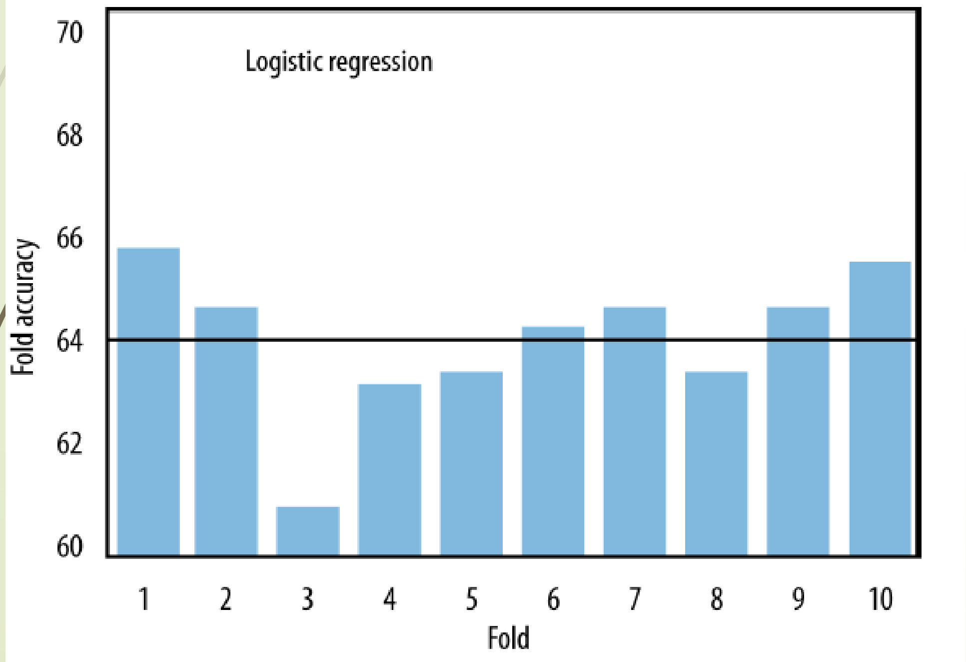


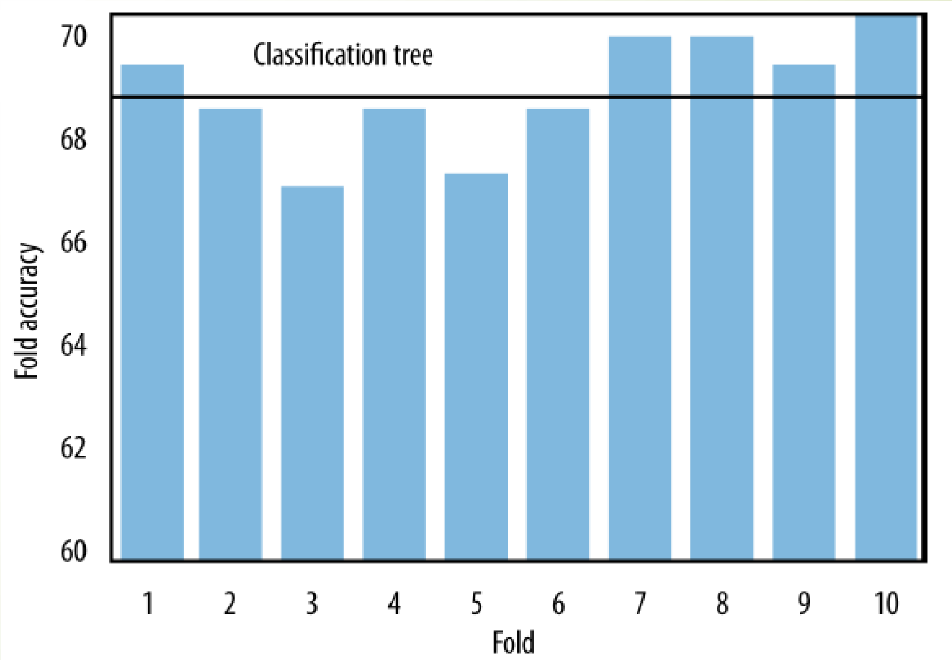
Linear equation learned by Logistic regression
acc. = 98.9% (ln_performance)

Deducted decision tree(J48)
acc. = 99.1% (ln_performance)

Logistic Regression vs. Decision Tree

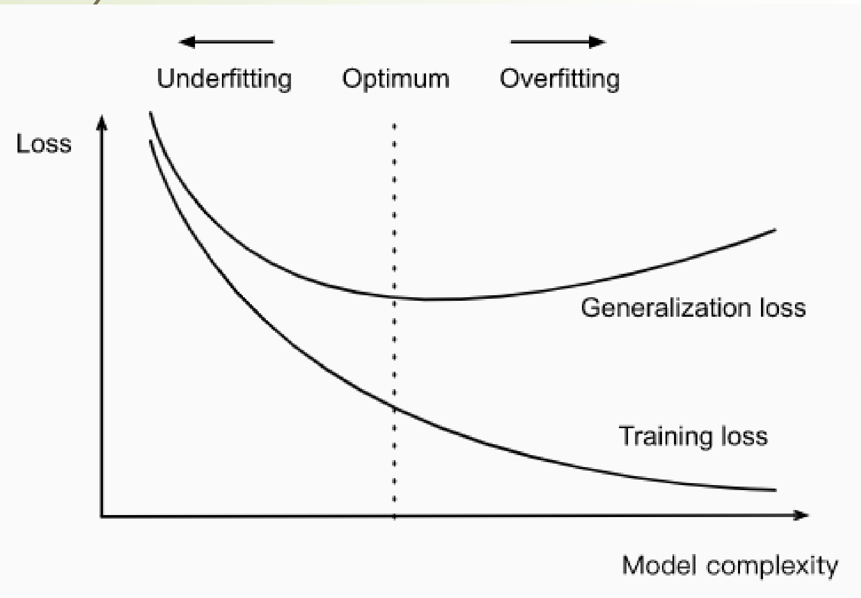
- Logistic regression have slightly lower average accuracy (64.1%) and with higher variation (standard deviation of 1.3) ➡ Decision tree are more stable and accurate





Interpreting Fitting Graph

- Focusing on the performance on the holdout data among cross validation
  - Validation curve in sklearn : training and test scores for varying parameter values
- In-performance may help deterring the direction of model complexity
  - More complex models will offer more **higher flexibility** to pick up complex but effective patterns(good in_performance)



$$f(\mathbf{x}) = w'_0 + w'_1 * x_1 + w'_2 * x_2 + w'_3 * x_3 + w'_4 * x_4$$

$$\begin{aligned} w'_0 &= w_0 \\ w'_1 &= w_1 \\ w'_2 &= w_2 \\ w'_3 &= 0 \\ w'_4 &= 0 \end{aligned}$$

$$f(\mathbf{x}) = w_0 + w_1 * x_1 + w_2 * x_2$$

Outline

- Cross Validation
- Overfitting Avoidance
- Similarity and Distance
- Quiz

Generals Method for Avoiding Overfitting

- **1-Control the model complexity** (assuming that each model has a parameter C to control complexity)
  - ✓ Complexity parameter C**:** decision tree - number of nodes; mathematical function-weight of regularization (larger weight of regularization ➧ lower model complexity)
  - ✓ Occam's razor: choose the simplest model/hypothesis that can fit the data well
  - ✓ How to set parameter C?
- **2-Feature Selection** – use fewer attributes
  - ✓ What is the subset of features should be retained?
- **3-Collect more data**
  - ✓ Using the learning curve

Avoiding Overfitting in Tree Induction

- **Pre-pruning:** stop growing the tree before it gets too complex
  - ✓ Stop criteria: the information gain/nodes in leaf is lower than a threshold
  - ✓ Decide the threshold by experience, or using hypothesis test
  - ✓ Computational efficient, but easy to be under-fitted
- **Post-pruning:** takes a fully-grown decision tree and discards unreliable parts
  - ✓ Estimate whether replacing a set of leaves or a branch with a leaf would reduce accuracy ? If not, do the replacement
  - ✓ Continue process iteratively, until any removal or replacement would reduce accuracy

Complexity Control in Mathematical Functions

- For models described as a mathematical function, its loss on the whole training data can also be represented as a mathematical function ***TrainLoss(w)***. Then we try to find the best model parameter ***w*** that generates least loss on training data

$$\min_{\mathbf{w}} TrainlLoss(\mathbf{w})$$

- Occam's razor: choose the simplest model/hypothesis that can fit the data well
- Regularization: Add the penalty of model complexity to the objectives of optimizations
  - ✓ Weight C determines how much importance the optimization procedure should place on the penalty, compared to *TrainLoss*
  - ✓ Trade-off between model simplicity and its in_performance

$$\min_{\mathbf{w}} TrainlLoss(\mathbf{w}) + C \cdot penalty(\mathbf{w})$$

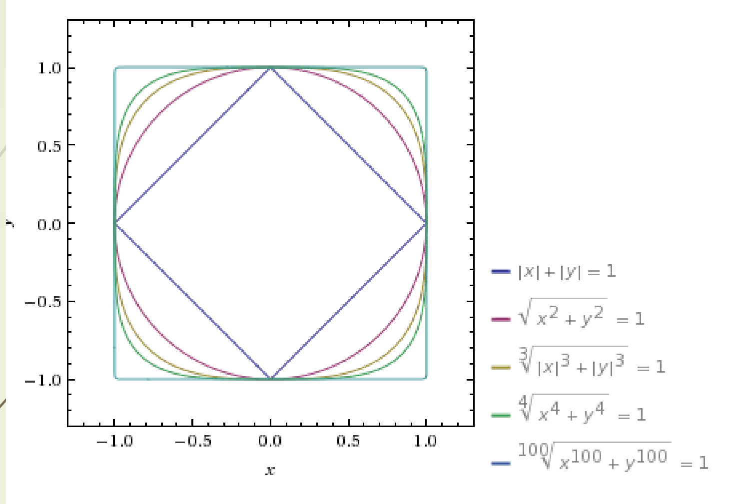
Popular Complexity Penalty

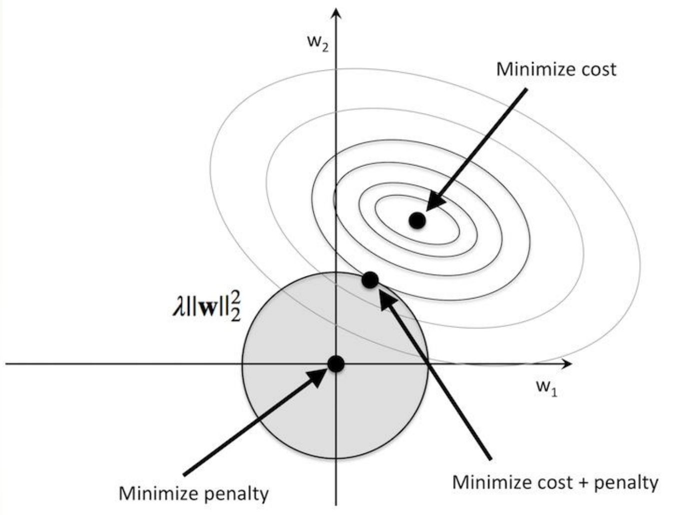
- We try to find optimal ***w*** by optimizing the following regularized objective

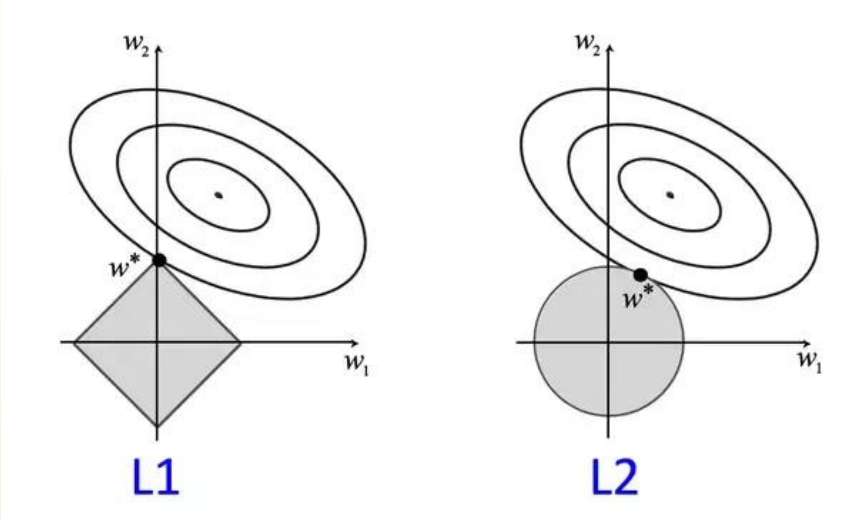
$$\min_{\mathbf{w}} TrainlLoss(\mathbf{w}) + C \cdot penalty(\mathbf{w})$$

- L2-norm penalty: $penalty(\mathbf{w}) = \|\mathbf{w}\|_2 = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$
  - ✓ The sum of the *squares* of the weights (avoid large negative/positve weights)
  - ✓ L2-norm + standard least-squares linear regression = ridge regression
- L1-norm penalty: $penalty(\mathbf{w}) = \|\mathbf{w}\|_1 = |w_1| + |w_2| + \ldots + |w_n|$
  - ✓ The sum of the absolute values of the weights
  - ✓ L1-norm + standard least-squares linear regression = LASSO regression
  - ✓ Zero out entries of weights (drop corresponding feature)

Intuition of L1/L2-norm Penalty*







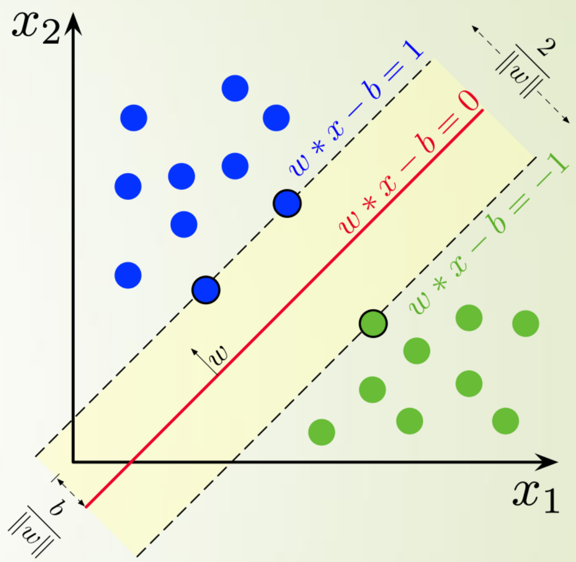
SVM as Regularized Linear Model

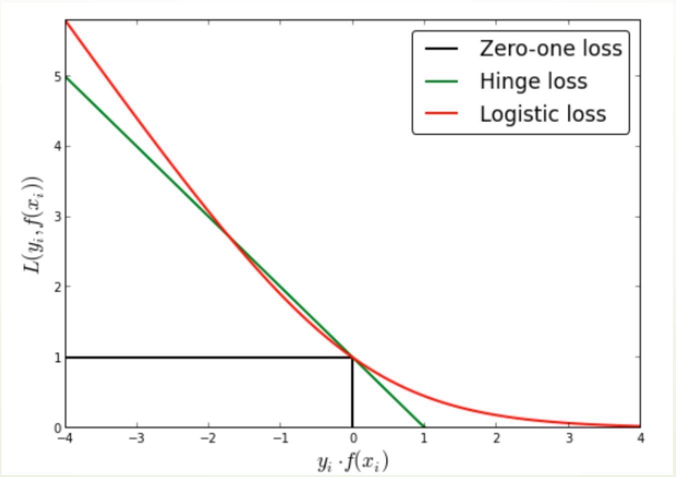
- SVM tries to optimize

$$\min_{\mathbf{w}} HingeLoss(\mathbf{w}) + C \cdot \|\mathbf{w}\|_2$$

- Linear SVM learning is almost equivalent to L2-regularized Logistic regression as follows

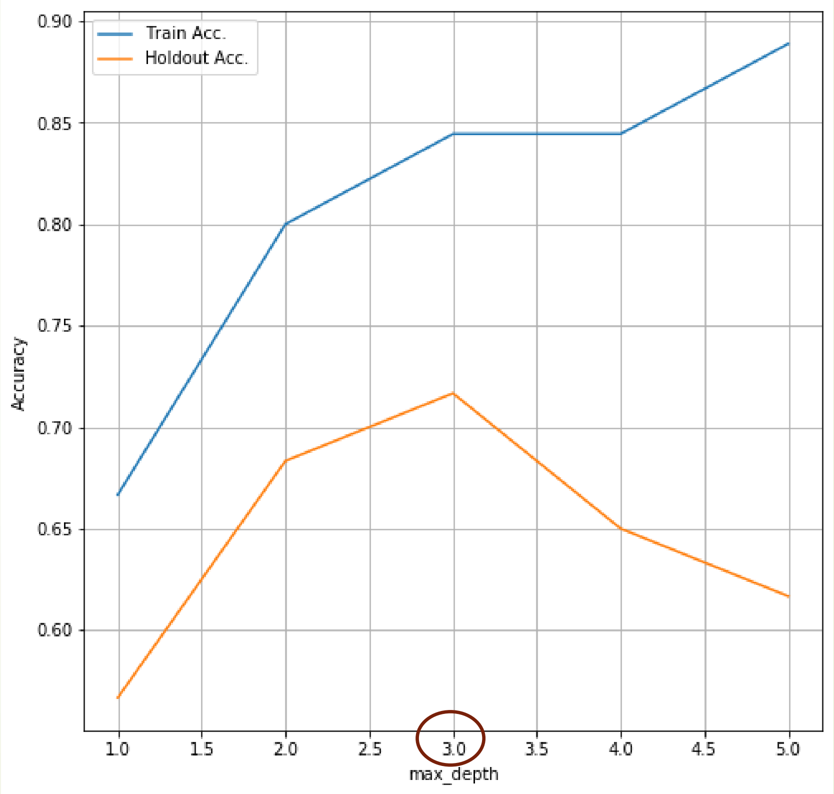
$$\min_{\mathbf{w}} LogisticLoss(\mathbf{w}) + C \cdot \|\mathbf{w}\|_2$$





Parameter Selection with Cross-validation

- Treat models with different complexity parameter C or with different subset of features as different models
  - ✓ Examples of C: trade coefficient of SVM, max_depth for decision tree …
  - ✓ Use cross-validation to estimate their generalization performance respectively and select the best model accordingly.
  - ✓ The corresponding parameters of the best model are the optimal parameter C_{best}
- Train the model with parameter C_{best} on the whole training data to get the final trained model (during CV we only trained on k-1 folds of training data)



GridSearchCV for Multiple Parameters

- SVM with kernel = 'poly'

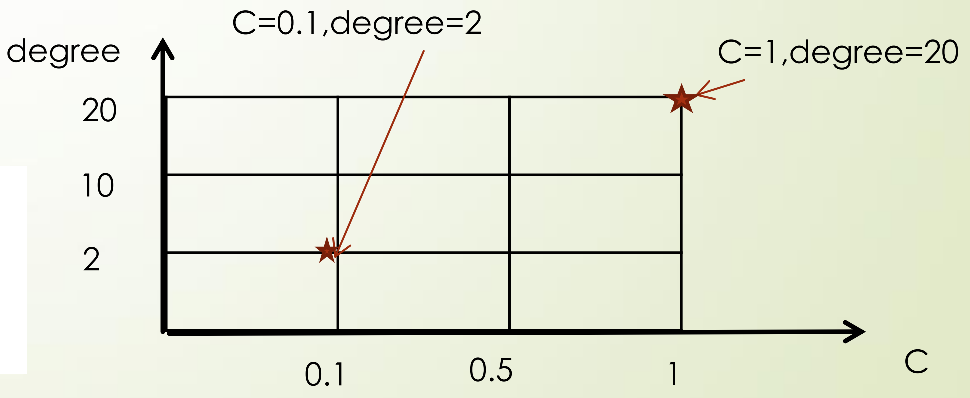
C : *float, optional (default=1.0)*
Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel : *string, optional (default='rbf')*
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, optional (default=3)*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

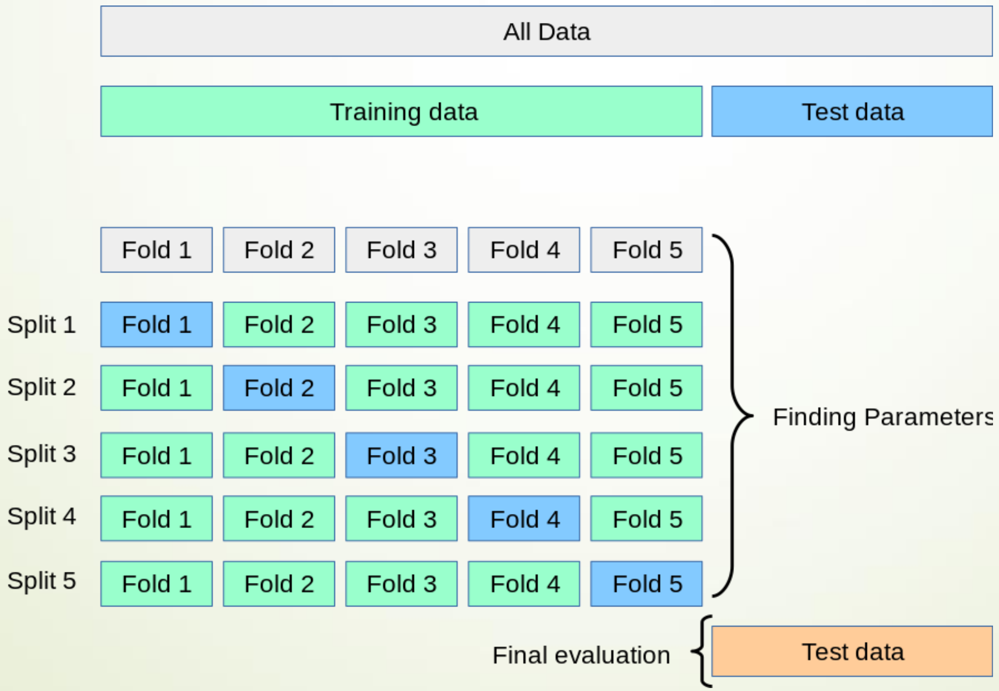
refit : *boolean, string, or callable, default=True*
Refit an estimator using the best found parameters on the whole dataset.

For multiple metric evaluation, this needs to be a string denoting the scorer that would be used to find the best parameters for refitting the estimator at the end.



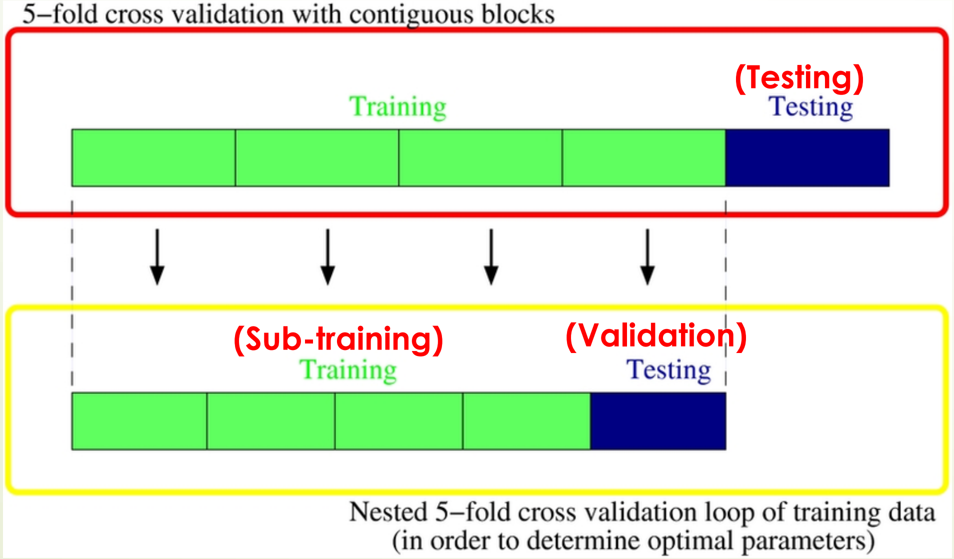
Estimated Generalization Performance

- Report the estimated generalization performance by simple cross-validation is <u>overly optimistic</u>
  - ✓ We choose the best model by estimated generalization performance, and then report the generalization performance directly



Nested Cross Validation*

- Nested cross validation
  - ✓ (Outer) Repeatedly split whole training data into **training/testing** data. (Fit selected model to training data, and report performance on testing data)
  - ✓ (Inner) Do cross-validation on the training data by further repeatedly splitting the **training into sub-training/validation**(select the best model/parameter)



2-Feature Selection

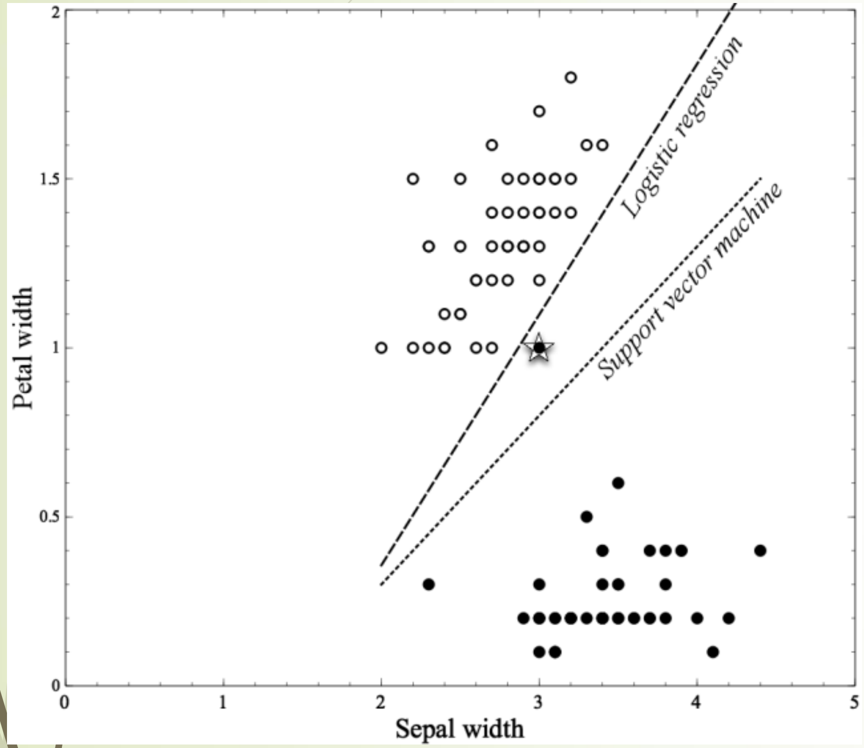
- Sequential forward selection (SFS) can help us to select features based on nested cross-validation
  - ✓ Pick the best individual feature by looking at all models built with just one feature
  - ✓ Then, test all models that add a second feature to first chosen feature
  - ✓ Proceed similarly with three, four, … features
- SFS is computationally expensive by evaluating all combinations of features
- Sequential backward elimination (Using all features initially, but delete feature one by one based on nested cross-validation)
- L1-norm for feature selection (drop the features associated with **zero weights**)

$$f(\mathbf{x}) = w'_0 + w'_1 * x_1 + w'_2 * x_2 \boxed{+ w'_3 * x_3 + w'_4 * x_4}$$

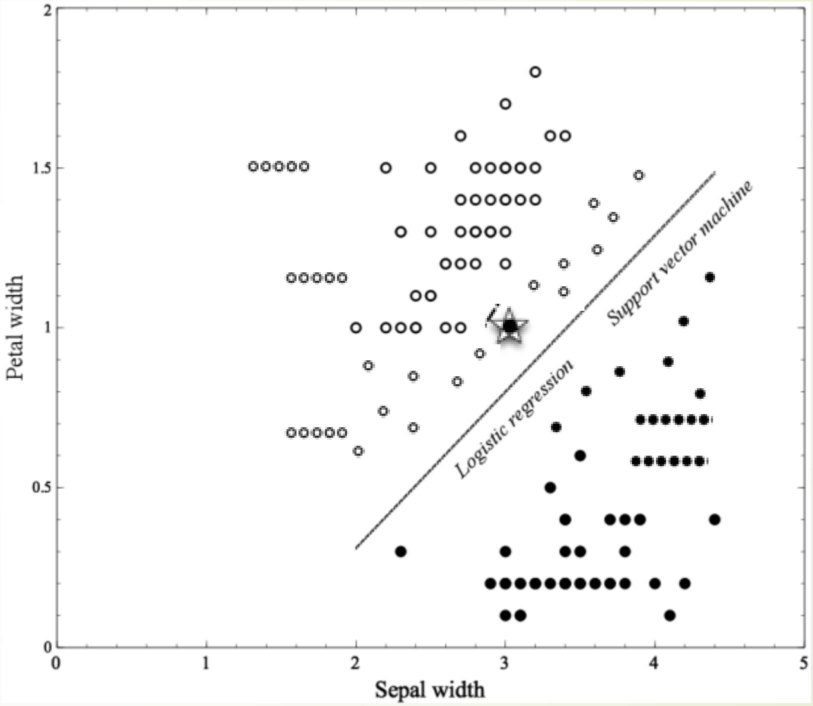
$$w'_0 = w_0$$
$$w'_1 = w_1$$
$$w'_2 = w_2$$
$$\boxed{\begin{array}{l} w'_3 = 0 \\ w'_4 = 0 \end{array}}$$

$$f(\mathbf{x}) = w_0 + w_1 * x_1 + w_2 * x_2$$

3-Collect More Data: Intuition



Training data with one outliers

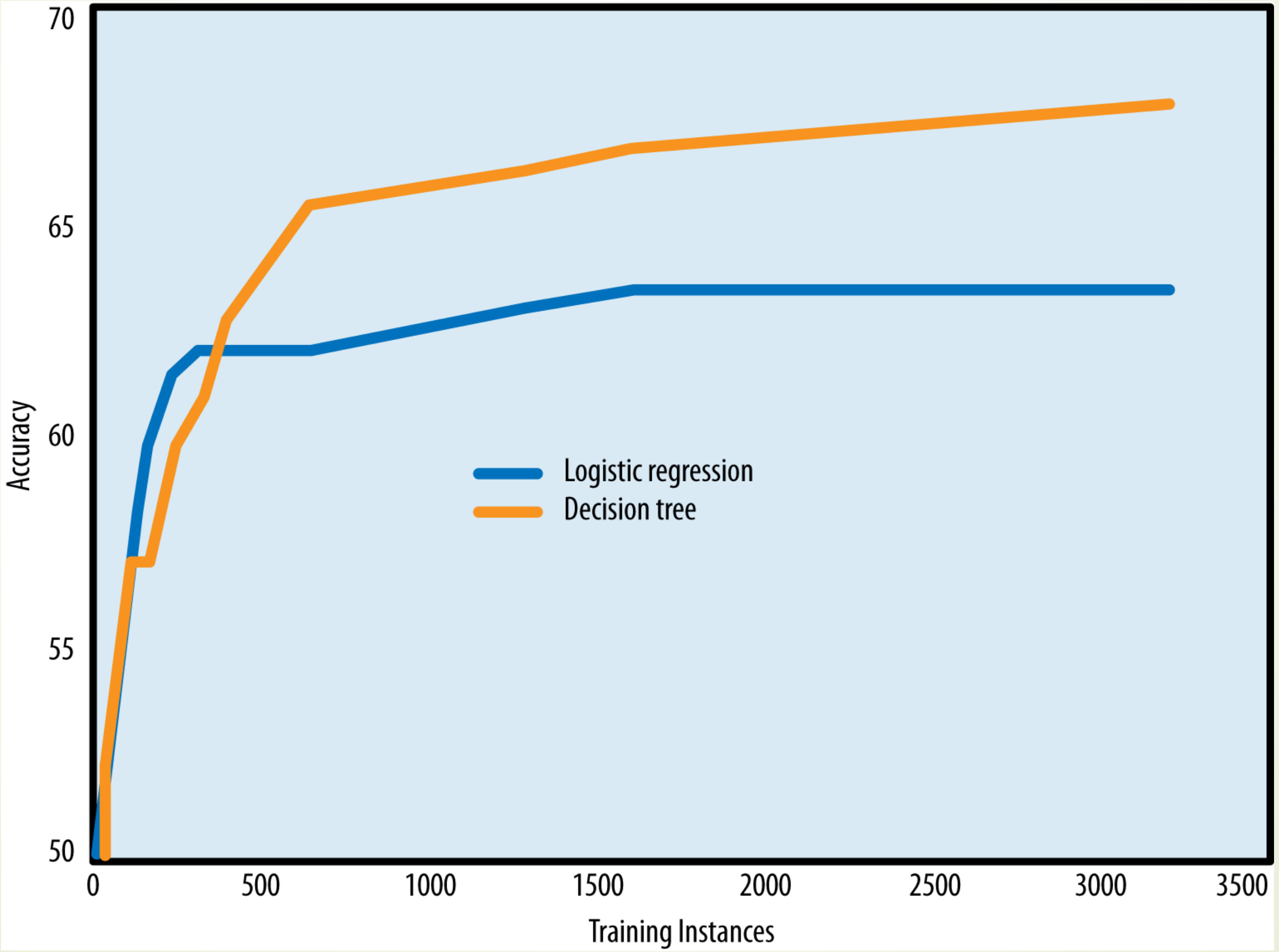


Collecting more data

3-Collect More Data: Learning Curve

- The learning curve is a plot of the estimated generalization performance against the amount of training data
  - ✓ Usually steep initially, but then marginal advantage of more data decreases
  - ✓ Generalization performance improves as more training data are available
  - ✓ Fitting curve shows the performance on the training and the holdout data against model complexity (for a fixed amount of training data)
- Learning curve may give recommendations on **how much to** invest in training data
  - ✓ Different modeling procedures may have different performance on the same data
  - ✓ Different size of training data may result in different estimated generation performance of the same model

Example of Learning Curve



Summary of CV in Sklearn

https://scikit-learn.org/stable/modules/cross_validation.html

- CV split:
  - train_test_split : random holdout split
  - StratifiedKFold: stratified k-fold CV
  - Kfold: k-fold CV
- CV evaluation
  - cross_validate: only score
  - cross_val_score: detailed information
- CV for model selection
  - validation_curve: Single parameter
  - gridSearchCV
- CV for learning curve
  - learning_curve

Outline

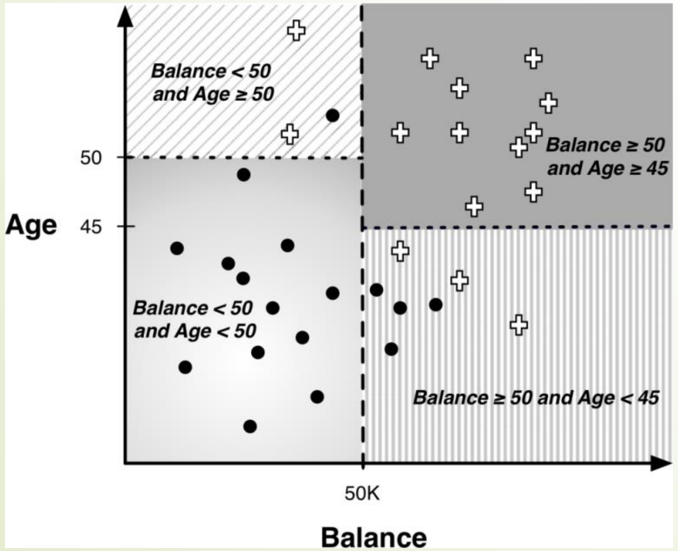
- Cross Validation
- Overfitting Avoidance
- Similarity and Distance
- Quiz

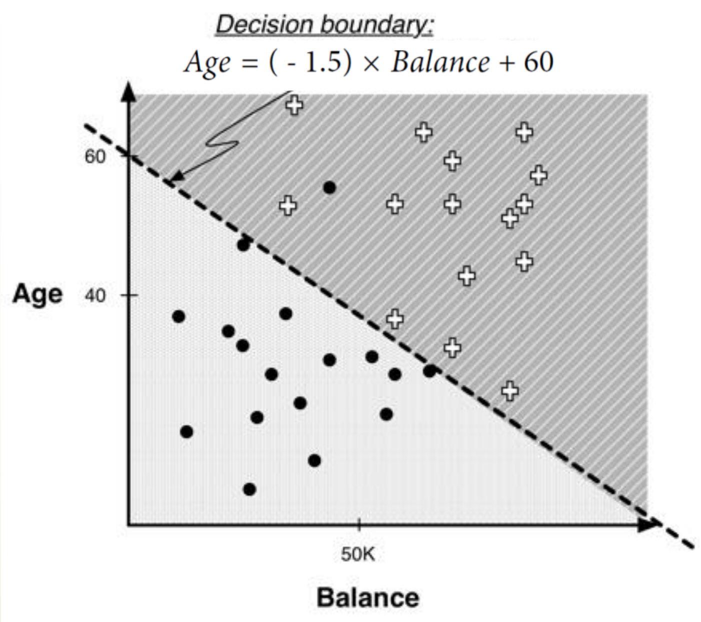
Applications of Similarity

- Similarity underlies many data science methods and solutions to business problems
- If two things (people, companies, products) are similar in some ways, they often share other characteristics as well
- Different sorts of business tasks involve reasoning from similar examples
  - ✓ Retrieve similar things: search engine
  - ✓ Classification or regression: K-Nearest-Neighbor
  - ✓ Clustering: grouping similar things tougher
  - ✓ Recommender system: Amazon --"People who like X also like Y"
  - ✓ Reasoning from similar cases beyond business applications: lawyers and doctors

Similarity under Segmentations

- Both classification trees and linear classifiers establish boundaries between regions of differing classifications.
- They have in common the view that instances sharing a common region in space should be similar.
- What differs between classification trees and linear classifiers is how the regions/boundaries are represented and discovered.





Similarity and Distance

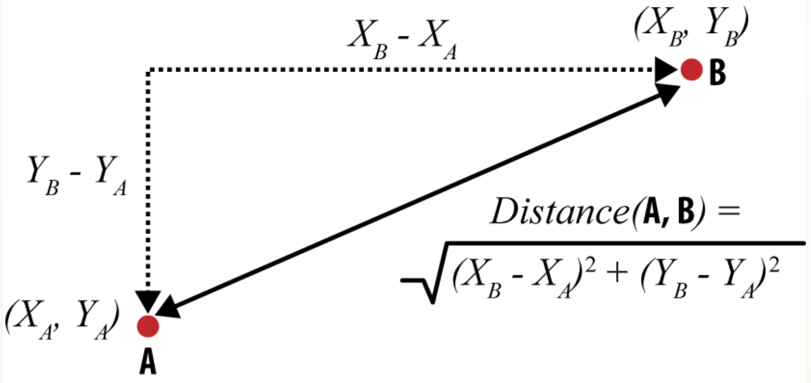
- Why not reason about the similarity or distance between objects directly?
- If we represent each object as a feature vector, then the closer two objects are in the space defined by the features, the more similar they are.
- For example :

Attribute	Person A	Person B
Age	23	40
Years at current address	2	10
Residential status (1=Owner, 2=Renter, 3=Other)	2	1

How to measure the similarity or distance between Person A and Person B ?
What does it mean that two companies or two consumers are similar?

Euclidean distance

- Euclidean distance is the "ordinary" straight-line distance between two points in Euclidean space.
  - ✓ When each sample have two features, then each object is a point in a two-dimensional space. (Pythagorean theorem)



  - ✓ When an object is described by *n*-dimensions features *(d_1, d_2, …, d_n)*, the general equation for Euclidean distance in *n* dimensions is shown as

$$\sqrt{(d_{1,A} - d_{1,B})^2 + (d_{2,A} - d_{2,B})^2 + \ldots + (d_{n,A} - d_{n,B})^2}$$

Euclidean distance (Example)

- Euclidean distance for Person A and Person B

$$d(A, B) = \sqrt{(23 - 40)^2 + (2 - 10)^2 + (2 - 1)^2} \approx 18.8$$

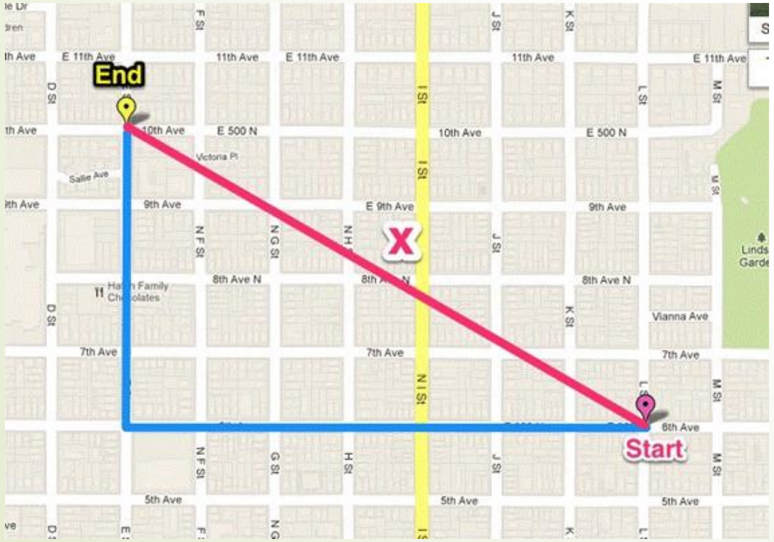
Attribute	Person A	Person B
Age	23	40
Years at current address	2	10
Residential status (1=Owner, 2=Renter, 3=Other)	2	1

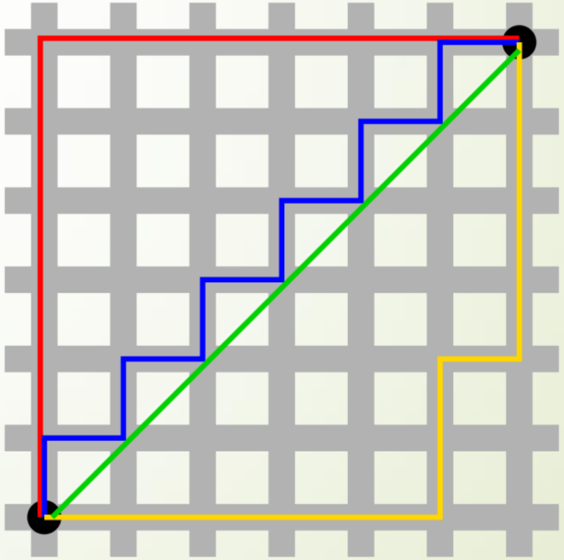
- ✓ This distance is just a number—it has no units, and no meaningful interpretation.
- ✓ Distance is only really useful for comparing the similarity of one pair of instances to that of another pair, such as *d(A,B)*<>= *d(A,C)*

Other Distance Measure (1)

- Manhattan distance: L1-norm distance
  - ✓ Application: Compressed sensing, compressed sensing …

$$d_{\text{Manhattan}}(\mathbf{X}, \mathbf{Y}) = \| \mathbf{X} - \mathbf{Y} \|_1 = | x_1 - y_1 | + | x_2 - y_2 | + \cdots$$

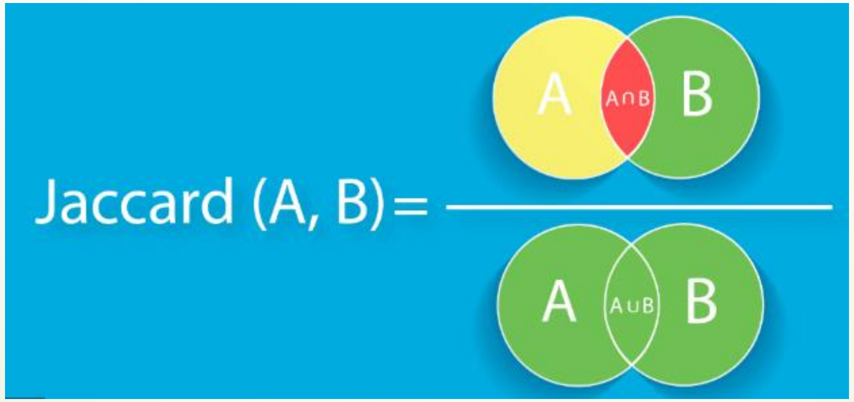




Other Distance Measure (2)

- Jaccard distance measure treats the two objects as *sets* of characteristics
  - ✓ The possession of a <u>common characteristic</u> between two item sets is important, but the <u>common absence</u> of a characteristic is not

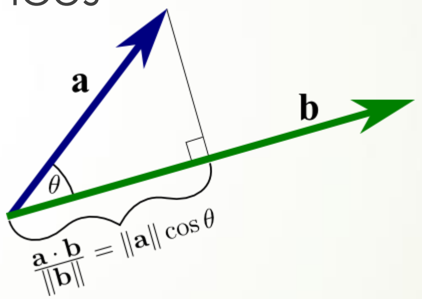
$$d_{\text{Jaccard}}(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$$



Other Distance Measure (3)

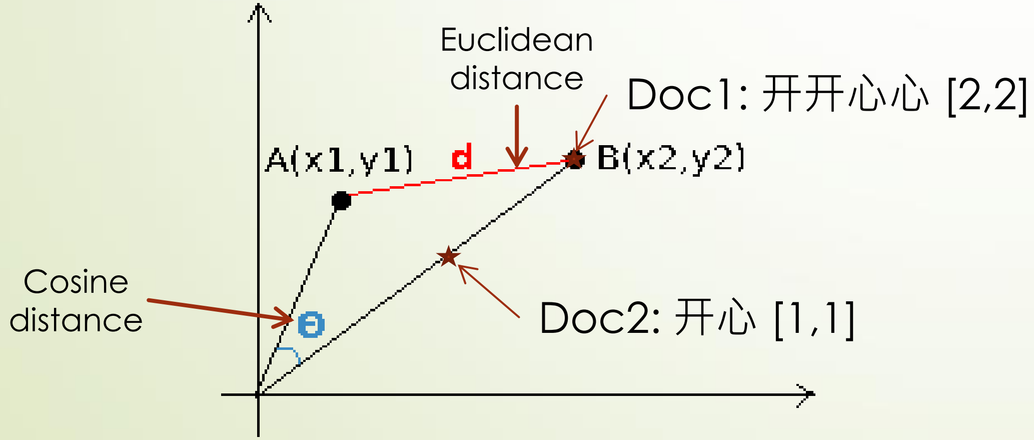
- Cosine distance: often used in text classification to measure the similarity of two documents
  - ✓ Often used in text classification to measure the similarity of two documents
  - ✓ Ignore differences in scale across instances

$$d_{cosine}(\mathbf{X}, \mathbf{Y}) = 1 - \frac{\mathbf{X} \cdot \mathbf{Y}}{\| \mathbf{X} \|_2 \cdot \| \mathbf{Y} \|_2}$$



$A = <7,3,2>$ and $B = <2,3,0>$

$$\begin{aligned} d_{\text{cosine}}(A, B) &= 1 - \frac{\langle 7, 3, 2\rangle \cdot \langle 2, 3, 0\rangle}{\| \langle 7, 3, 2\rangle \|_2 \cdot \| \langle 2, 3, 0\rangle \|_2} \\ &= 1 - \frac{7 \cdot 2 + 3 \cdot 3 + 2 \cdot 0}{\sqrt{49+9+4} \cdot \sqrt{4+9}} \\ &= 1 - \frac{23}{28.4} \approx 0.19 \end{aligned}$$



Other Distance/Similarity Measure (4)

- Hamming distance (equal length)
- Edit distance
- Longest common substring
- Longest common subsequence

Example: Similar Whiskey

- How can we can find the most similar whiskey of a given type of whiskey as a data scientist
  - ✓ Construct 5 attributes that can describe the general whiskey as follows

Color: *yellow, very pale, pale, pale gold, gold, old gold, full gold, amber, etc.*	(14 values)
Nose: *aromatic, peaty, sweet, light, fresh, dry, grassy, etc.*	(12 values)
Body: *soft, medium, full, round, smooth, light, firm, oily.*	(8 values)
Palate: *full, dry, sherry, big, fruity, grassy, smoky, salty, etc.*	(15 values)
Finish: *full, dry, warm, light, smooth, clean, fruity, grassy, smoky, etc.*	(19 values)

  - ✓ The values of attributes are **NOT** mutually exclusive (e.g., Aberlour's palate is described as medium, full, soft, round and smooth).
  - ✓ Use a feature vector of 68 (=14+12+8+15+19) binary (0/1) attributes to reprint a type of whiskey as [0,1,1,0,…,1] (with 68 entries of 0/1)

Example: Similar Whiskey by Euclidean Distance

- Given a special type of whiskey(Bunnahabhain), we could compute the Euclidean distance between it and other whiskey, respectively
- We could rank the other whiskey by the distance in descending order as follows

Whiskey	Distance	Descriptors
Bunnahabhain	—	*gold; firm,med,light; sweet,fruit,clean; fresh,sea; full*
Glenglassaugh	0.643	gold; firm,light,smooth; sweet,grass; fresh,grass
Tullibardine	0.647	gold; firm,med,smooth; sweet,fruit,full,grass,clean; sweet; big,arome,sweet
Ardbeg	0.667	sherry; firm,med,full,light; sweet; dry,peat,sea;salt
Bruichladdich	0.667	pale; firm,light,smooth; dry,sweet,smoke,clean; light; full
Glenmorangie	0.667	p.gold; med,oily,light; sweet,grass,spice; sweet,spicy,grass,sea,fresh; full,long

Outline

- Cross-validation & Learning Curve
- Overfitting Avoidance
- Similarity and Distance
- Quiz

Lab Quiz

- **Deadline**: 17:59 p.m., Mar. 27, 2020
- Two questions accounting for **5%** of overall score
- **Upload** the **answer worksheet** and the accomplished **Python files** to the **Blackboard**
- You may submit **unlimited times** but only the **LAST** submission will be considered
- **Only the answers in answer sheet** will be referred for grading
- Note： **MUST attach ALL** the required files in every submission/resubmission, otherwise other files will be missing.