3D geoinformation
Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

GEO5017
Machine Learning for the Built Environment

Lecture
Decision Trees, Random Forest, Data and Features

Shenglan Du

Today’s Agenda

- Previous Lecture: Linear Classifiers

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

Today’s Agenda

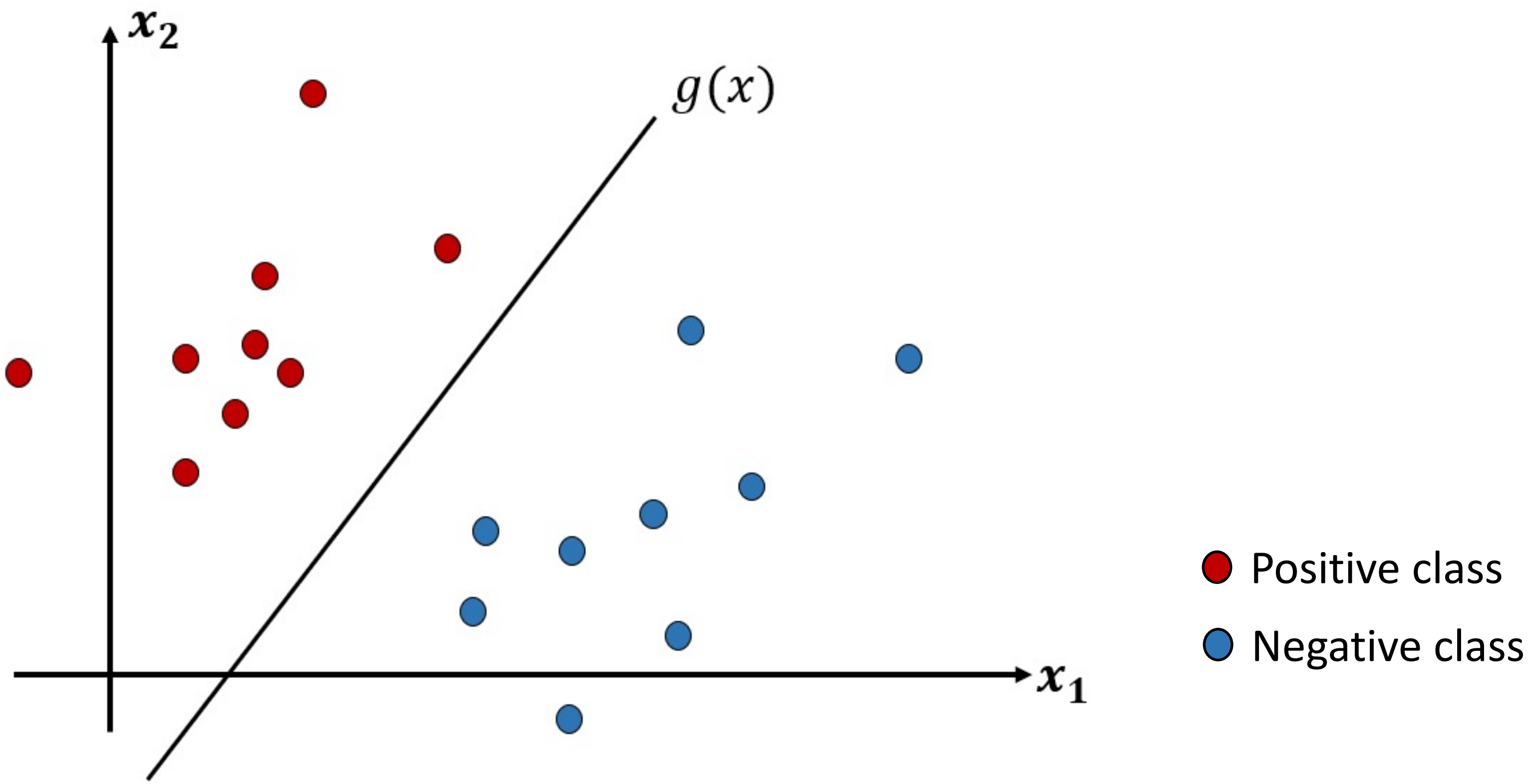
- Previous Lecture: Linear Classifiers 👈

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

Linear Classifiers

- What is the definition of linear classifiers?



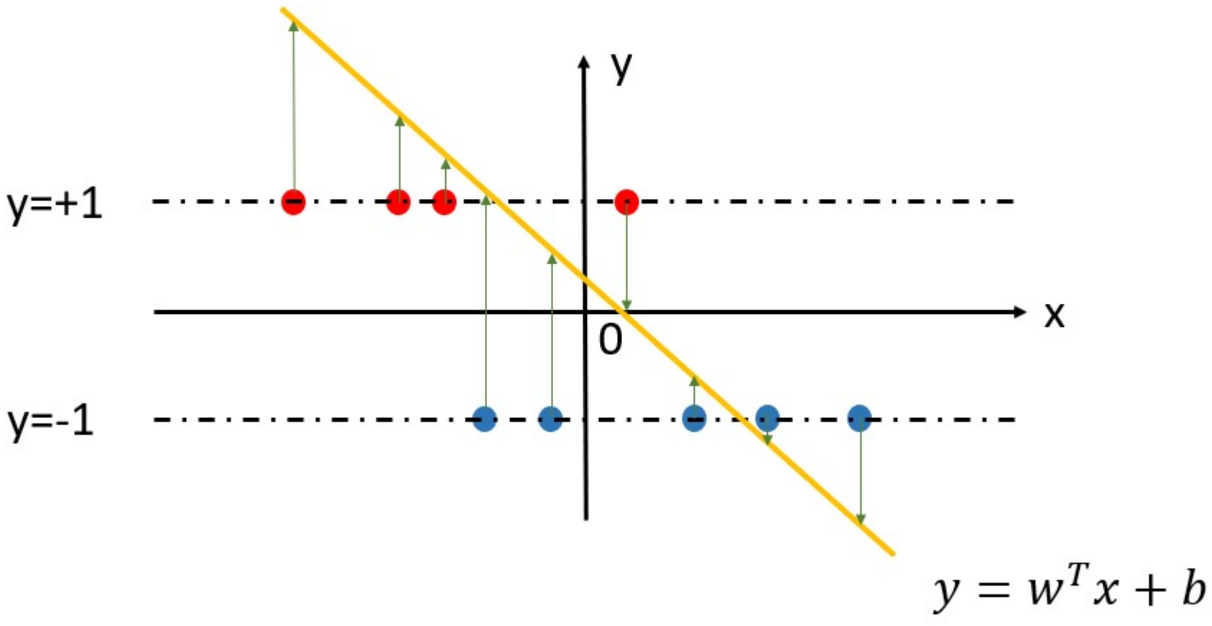
Linear Classifiers

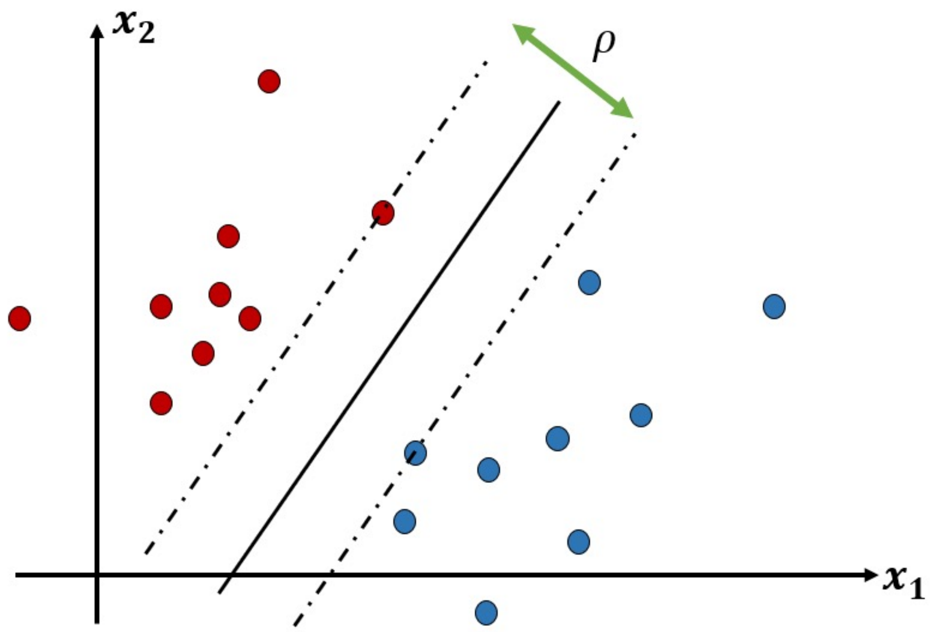
- We use a linear functions of input $\boldsymbol{x}$ to describe the decision boundary

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = 0$$

  - A decision boundary is a (D-1) dimension hyperplane of D dimension input feature space
  - If $\boldsymbol{x}$ is 1D, the decision boundary is a 0D point
  - If $\boldsymbol{x}$ is 2D, the decision boundary is a 1D line
  - ……

Linear Classifiers



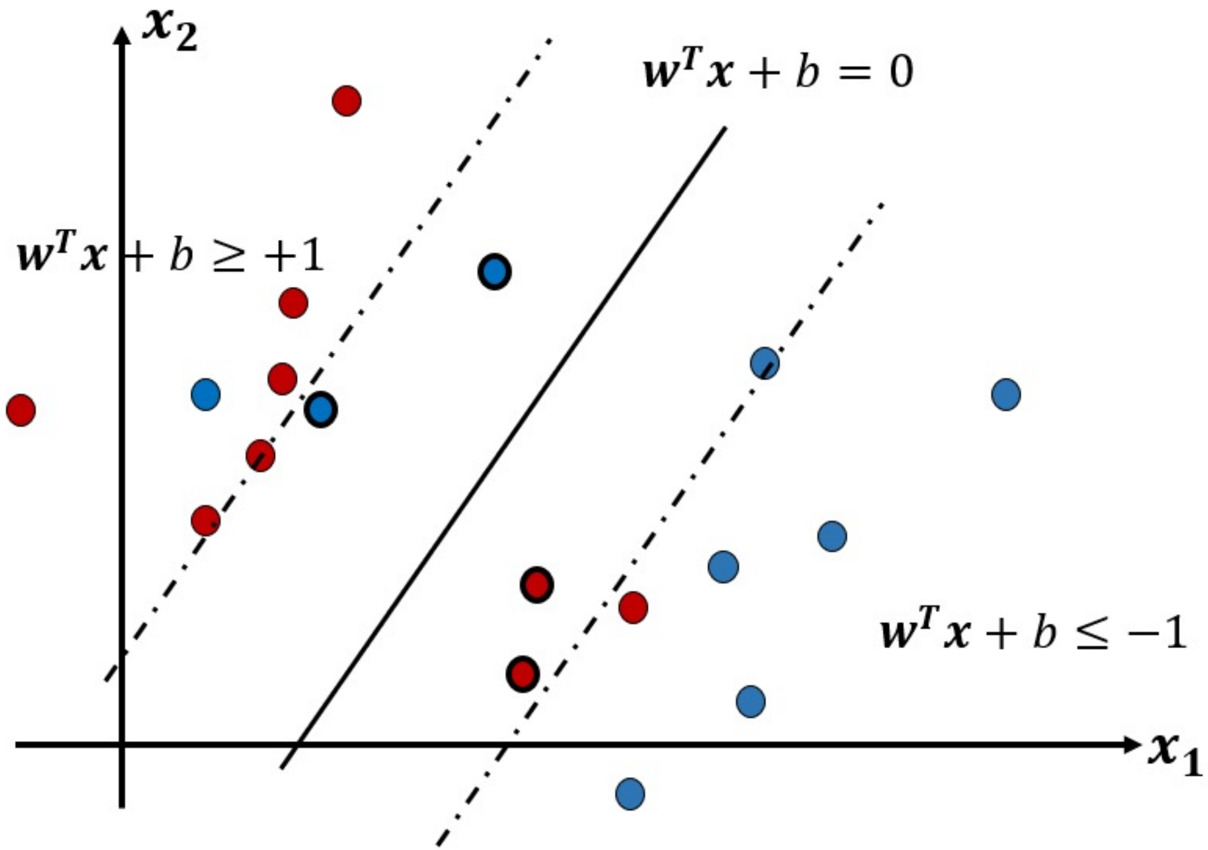


Standard Linear (Fisher) classifier
1D feature space

Standard SVM
2D feature space

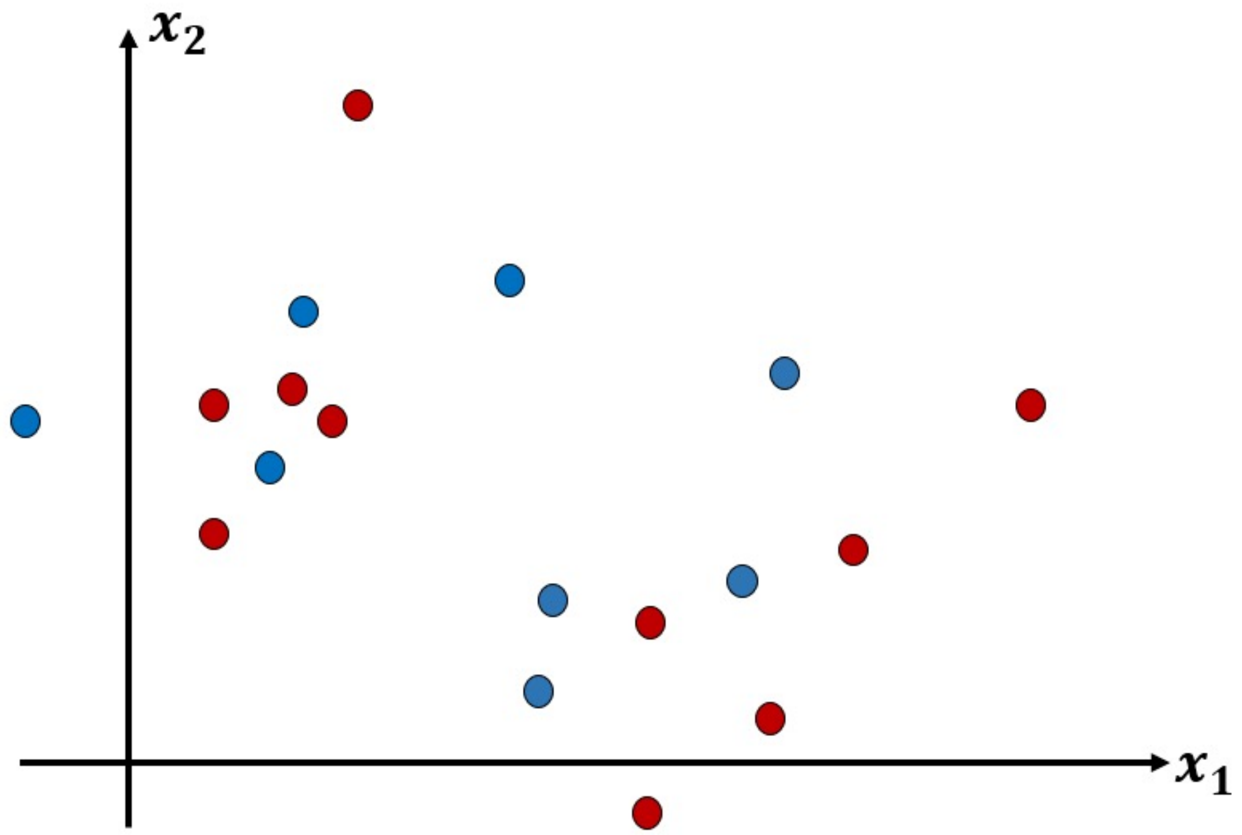
Linear Classifiers

- If there's ***slight*** data class overlap, soft-margin SVM is used



Linear Classifiers

- What if the classes highly overlap?



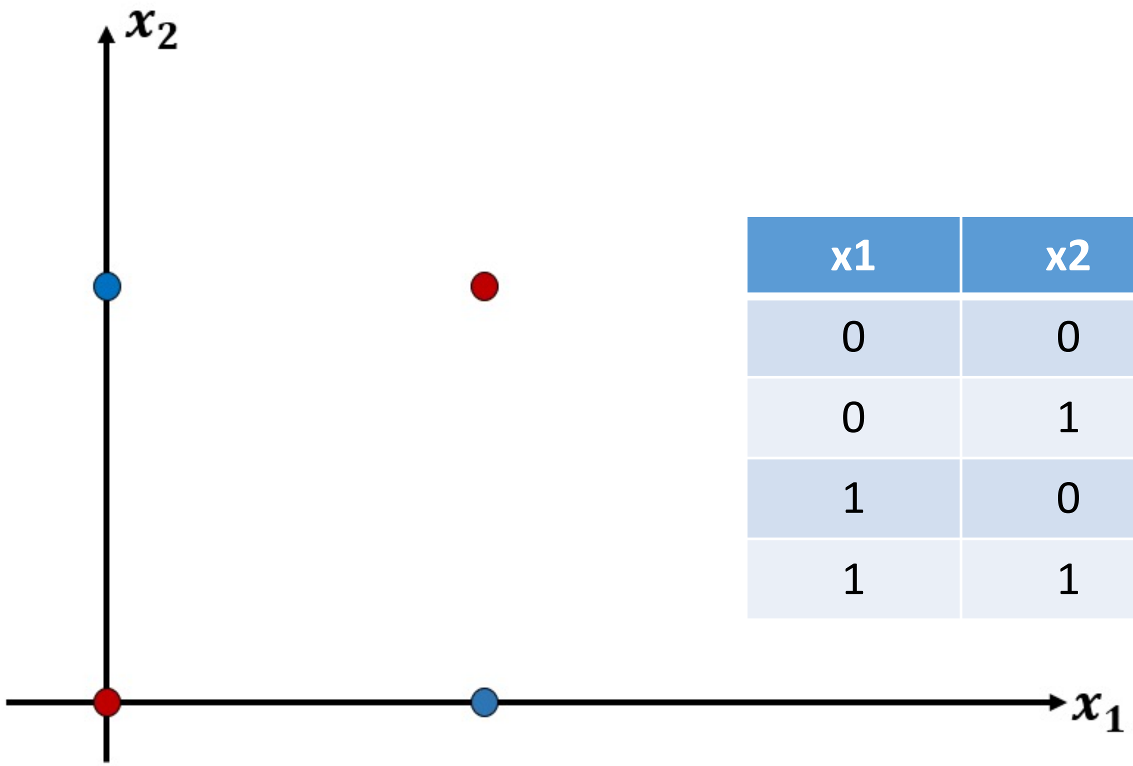
Linear Classifiers

- Example #1: Abnormalities in real world

Image source 1: https://savvygardening.com/narrow-trees-for-small-gardens/
Image source 2: https://9gag.com/gag/aOBNxmE

Linear Classifiers

- Example #2: XOR problem.



x1	x2	y
0	0	+1
0	1	-1
1	0	-1
1	1	+1

Non-Linear Classifiers

- Non-linear classifiers are designed to cope with non-linearly separable classes, which is quite common in real world
- Some popular non-linear classifiers:
  - Decision tree
  - Random forest
  - Multi-layer perceptron
  - (Deep) Neural network
  - ……

Today's Agenda

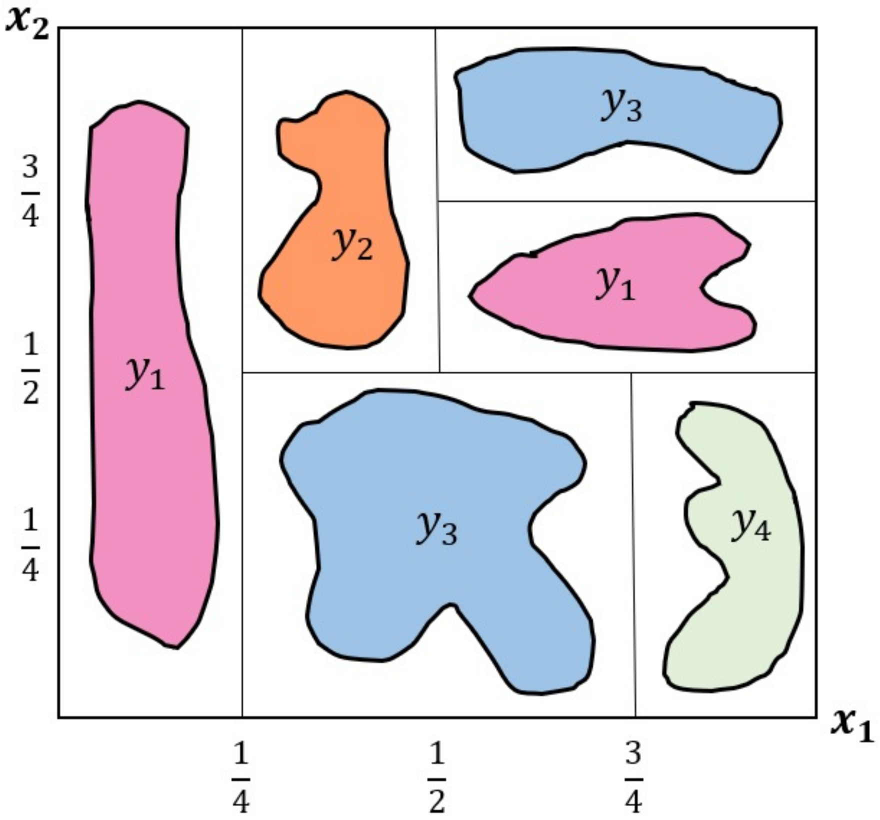
- Previous Lecture: Linear Classifiers

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

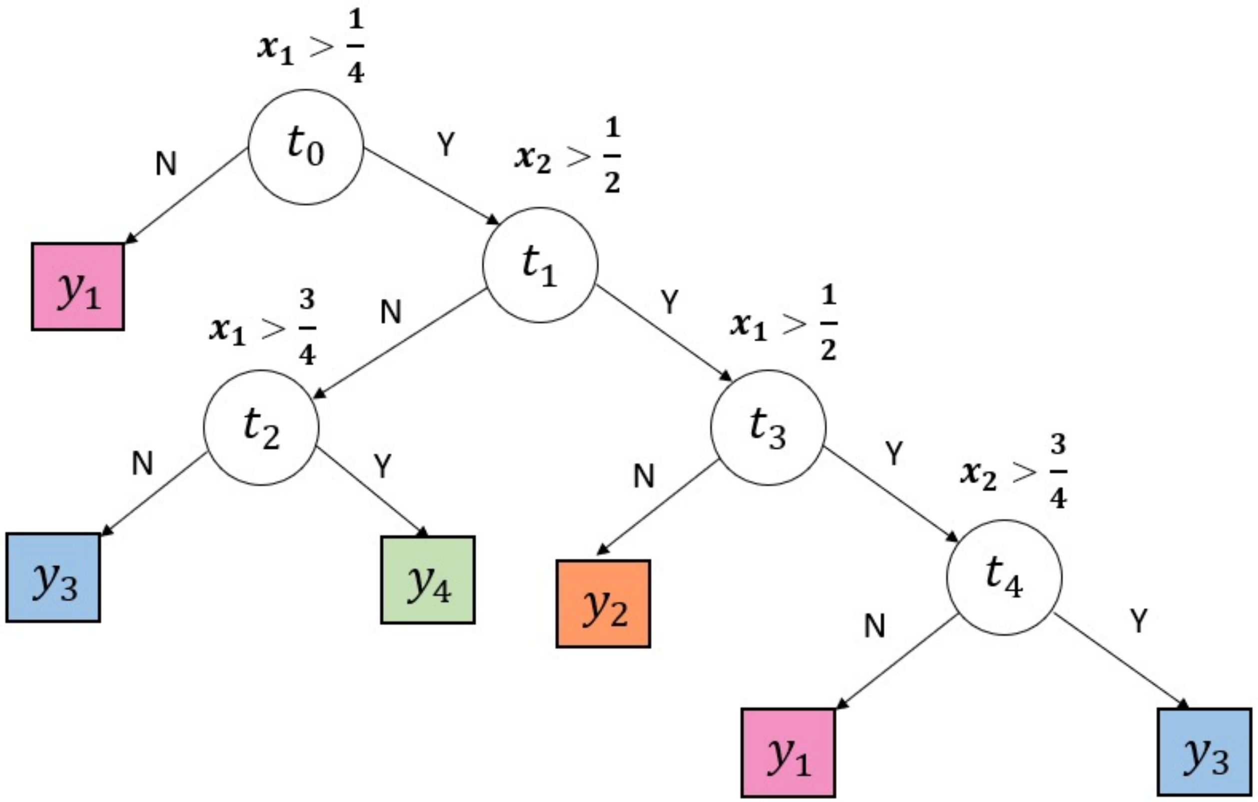
Decision Tree

- The feature space is split into unique regions, corresponding to classes, in a sequent manner



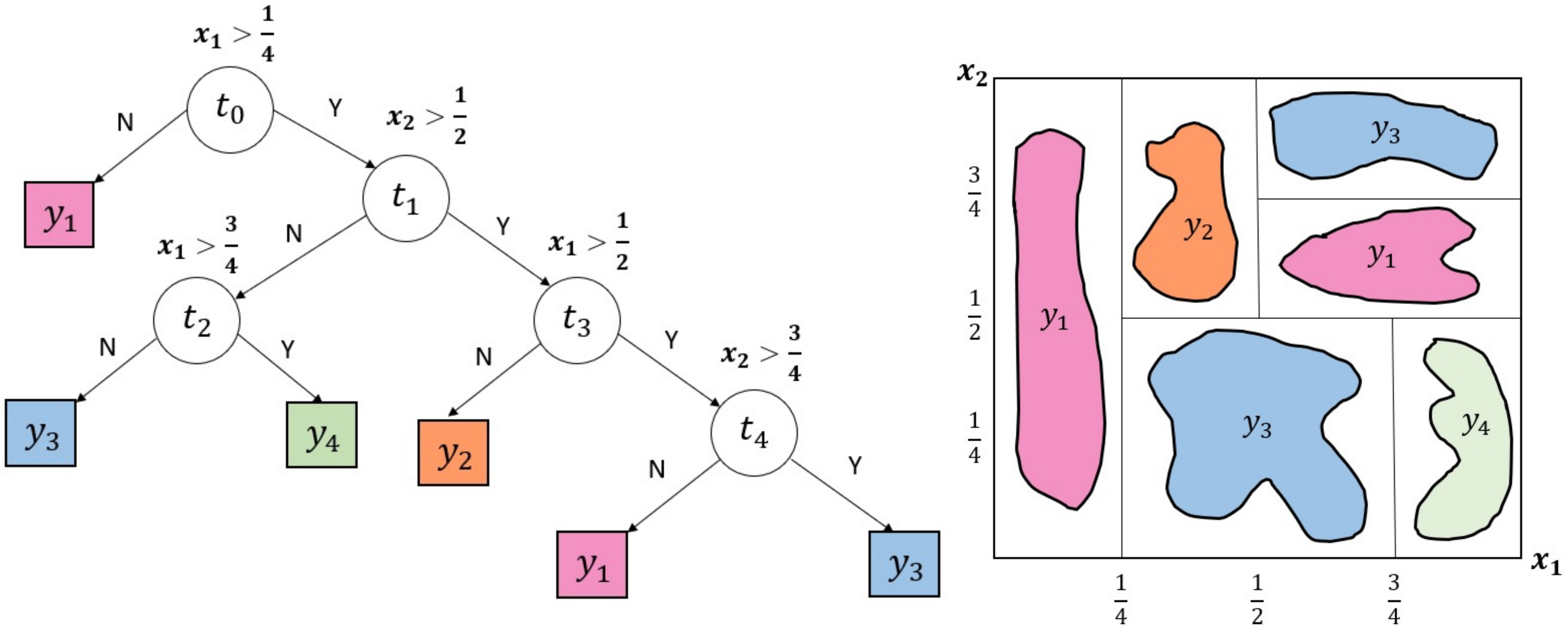
Decision Tree

- Classifying of a data sample is done by a sequence of decisions along a path of the tree



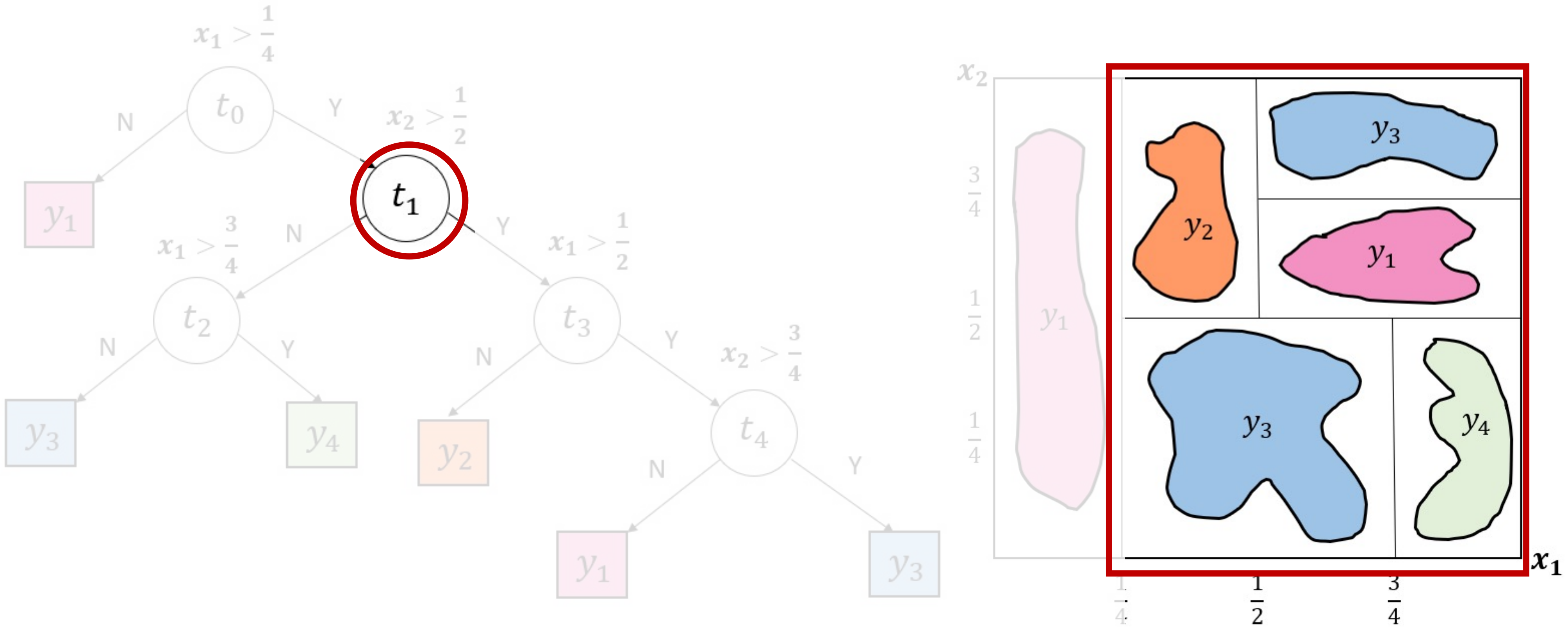
Decision Tree

- Splitting rule: every split must generate subsets that are more class homogeneous



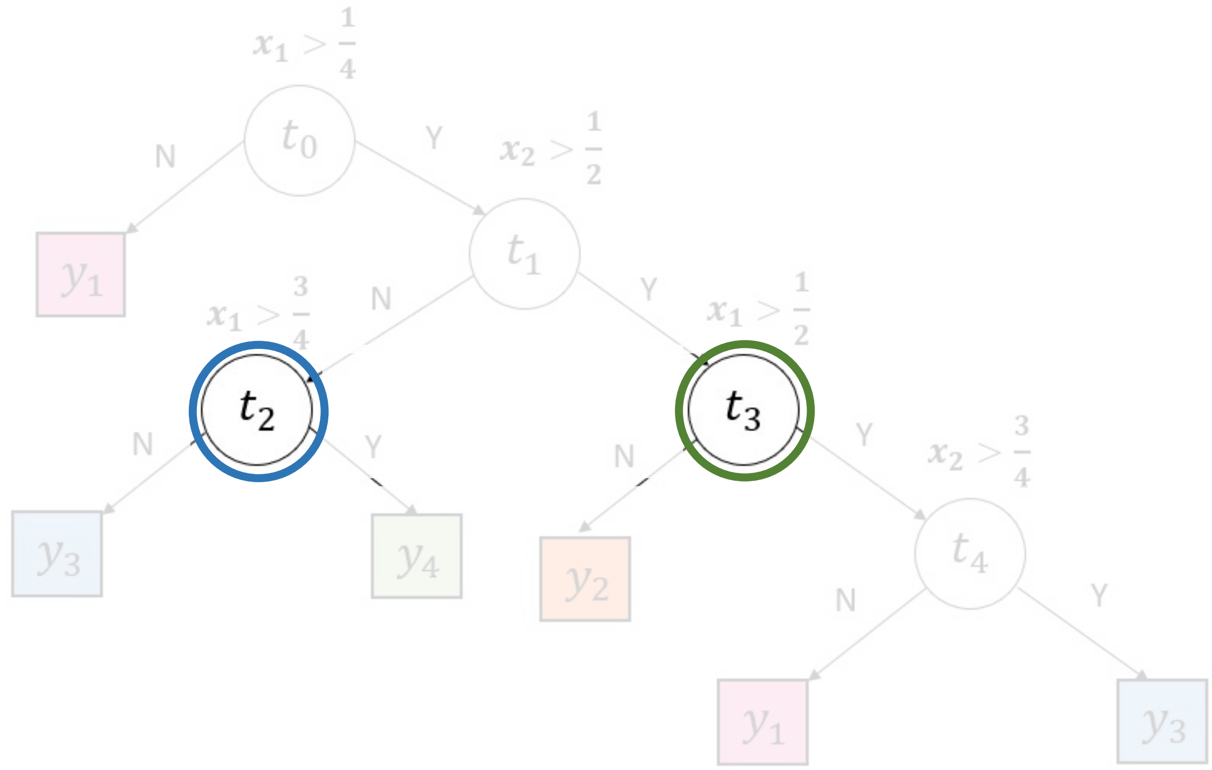
Decision Tree

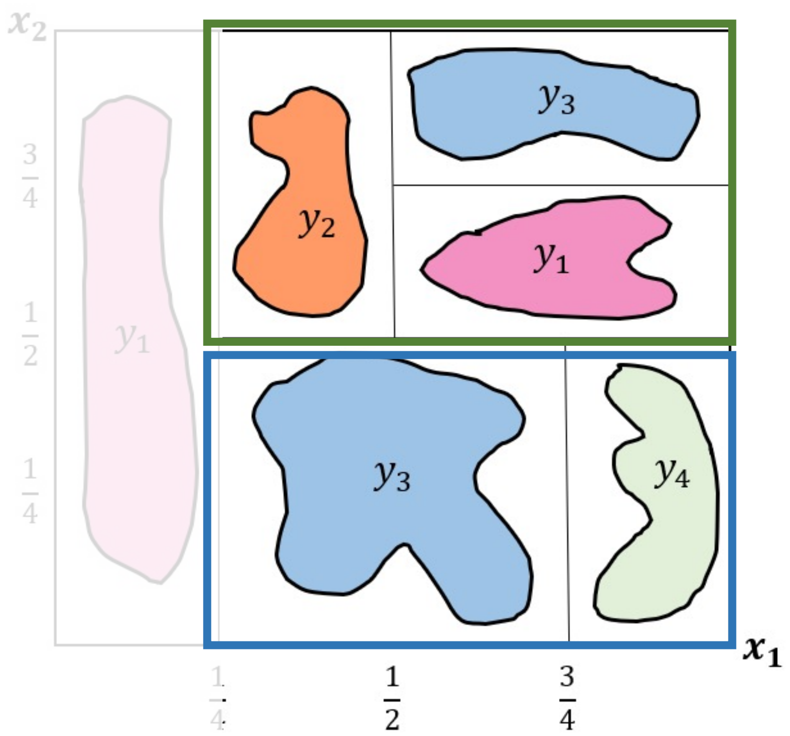
- Splitting rule: every split must generate subsets that are more class homogeneous



Decision Tree

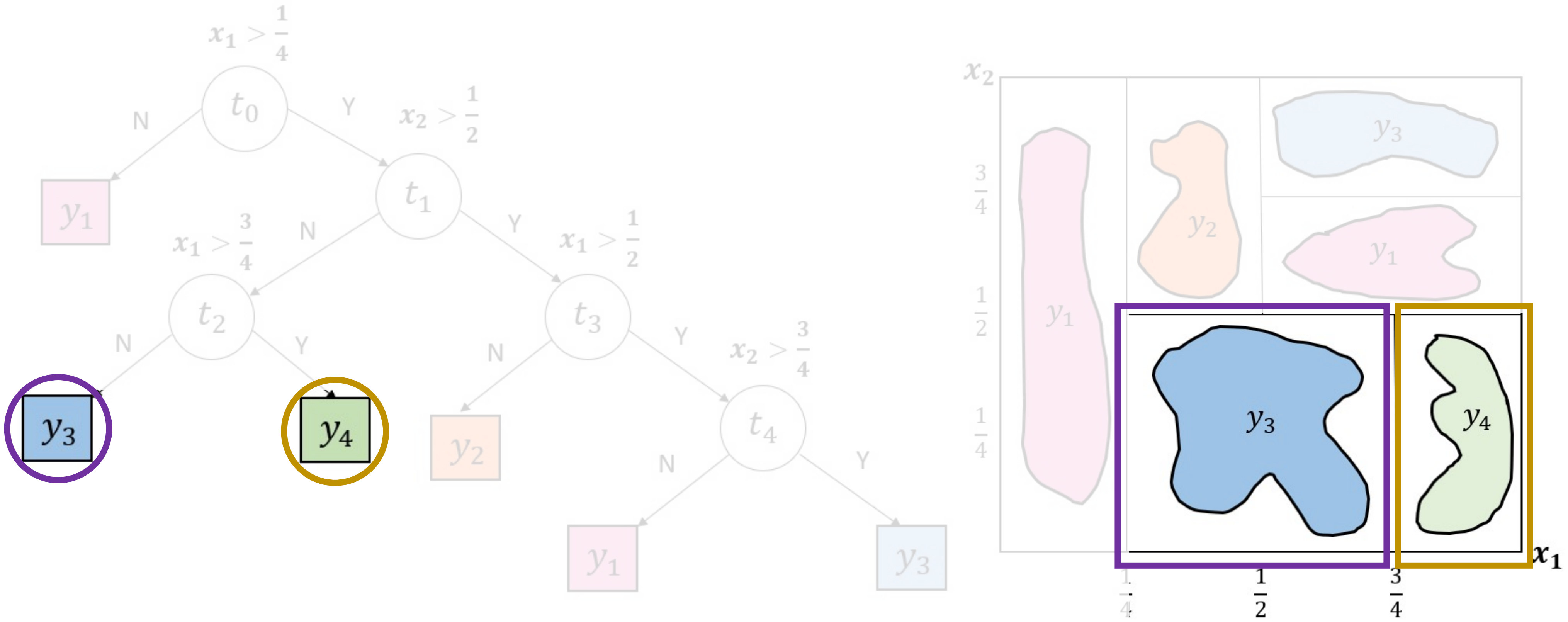
- Splitting rule: every split must generate subsets that are more class homogeneous





Decision Tree

- Splitting rule: every split must generate subsets that are more class homogeneous



Decision Tree

- Two impurity measures of a node t:
  - Gini impurity

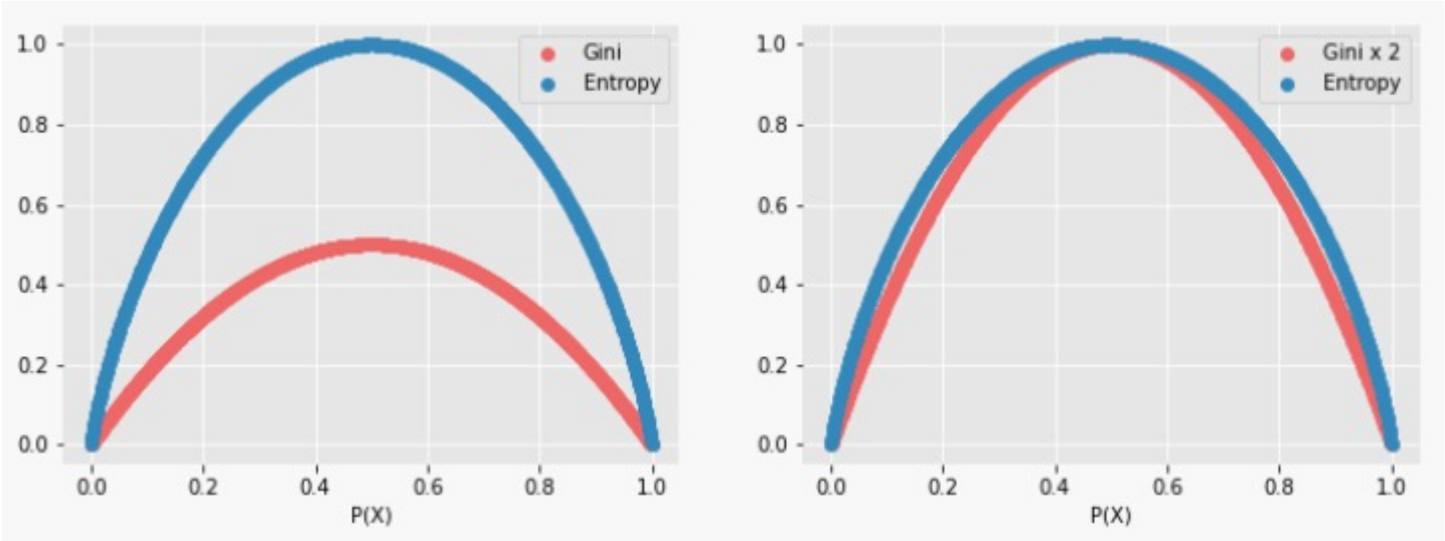
$$I(t) = 1 - \sum_{k=1}^{K} p(y_k|t)^2$$

  - Entropy impurity

$$I(t) = -\sum_{k=1}^{K} p(y_k|t) log_2 p(y_k|t)$$

Decision Tree

- Comparison between Gini and Entropy in 2-class problem



Left: original Gini compared with Entropy; Right: Gini*2 compared with Entropy

Decision Tree

- Decision tree growing steps:
  - Begin with the root node t of the original dataset $X_t = X$
  - For each feature x_i:
    - For each candidate value a_{in} (n=1,2,3,…,):
      - Divide the data into left node X_{tY} and right node X_{tN} by answering:
$$x_i < a_{in}$$
      - Compute the Impurity decrease
$$\Delta I = I(t) - \frac{N_{tY}}{N_t} I(tY) - \frac{N_{tN}}{N_t} I(tN)$$
  - Find the feature x_i and value a_{in} that lead to the most impurity decrease
  - Continue splitting……

Decision Tree

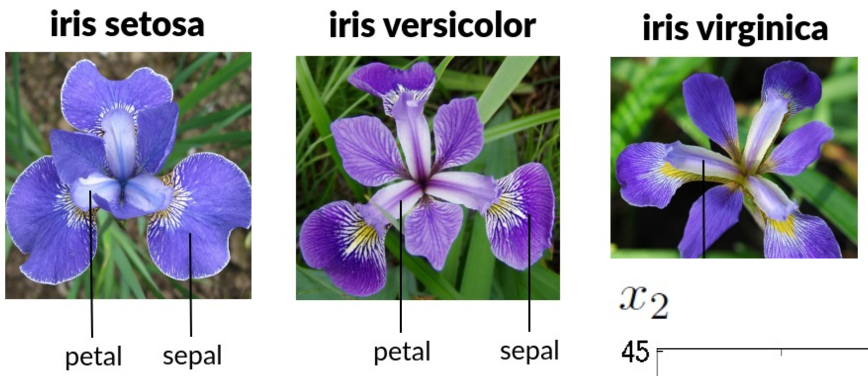
- Splitting stops until one of the following conditions meets:
  - Using all possible splitting ways, we have:

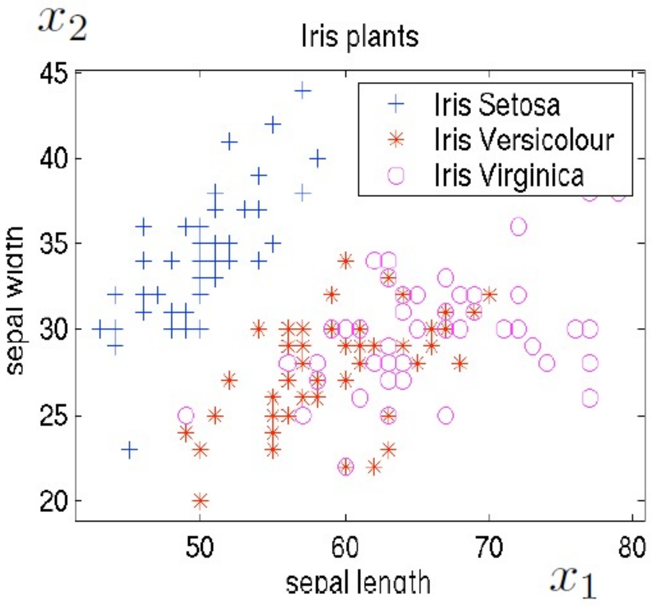
$$\Delta I < Threshold$$

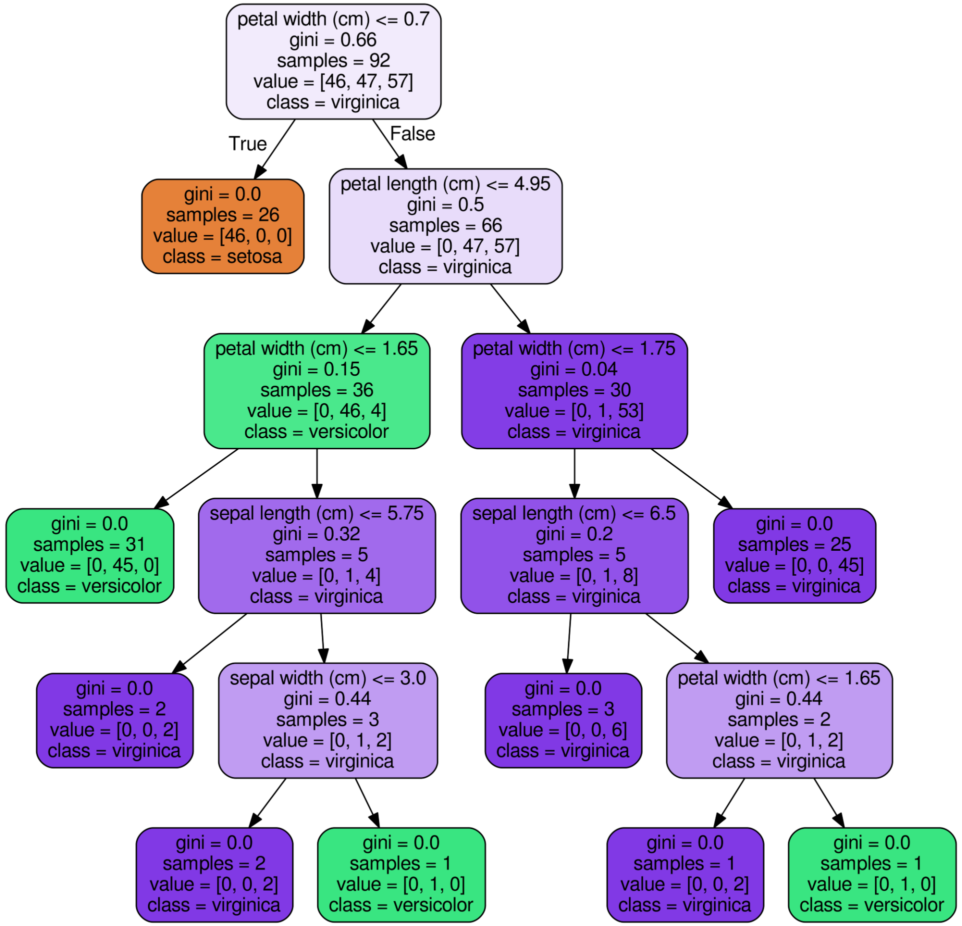
  - The data size of X_t in node t is too small
  - The data X_t in node t is pure now (i.e., contains only one class)

Decision Tree

- Visualizing a decision tree trained by iris dataset







Source code: https://gist.github.com/WillKoehrsen/ff77f5f308362819805a3defd9495ffd

Decision Tree Remarks

- Size of the tree must be large enough but too large. Otherwise, it overfits to particular data details

- Trees have high variance. A small change in data often leads to a very different tree

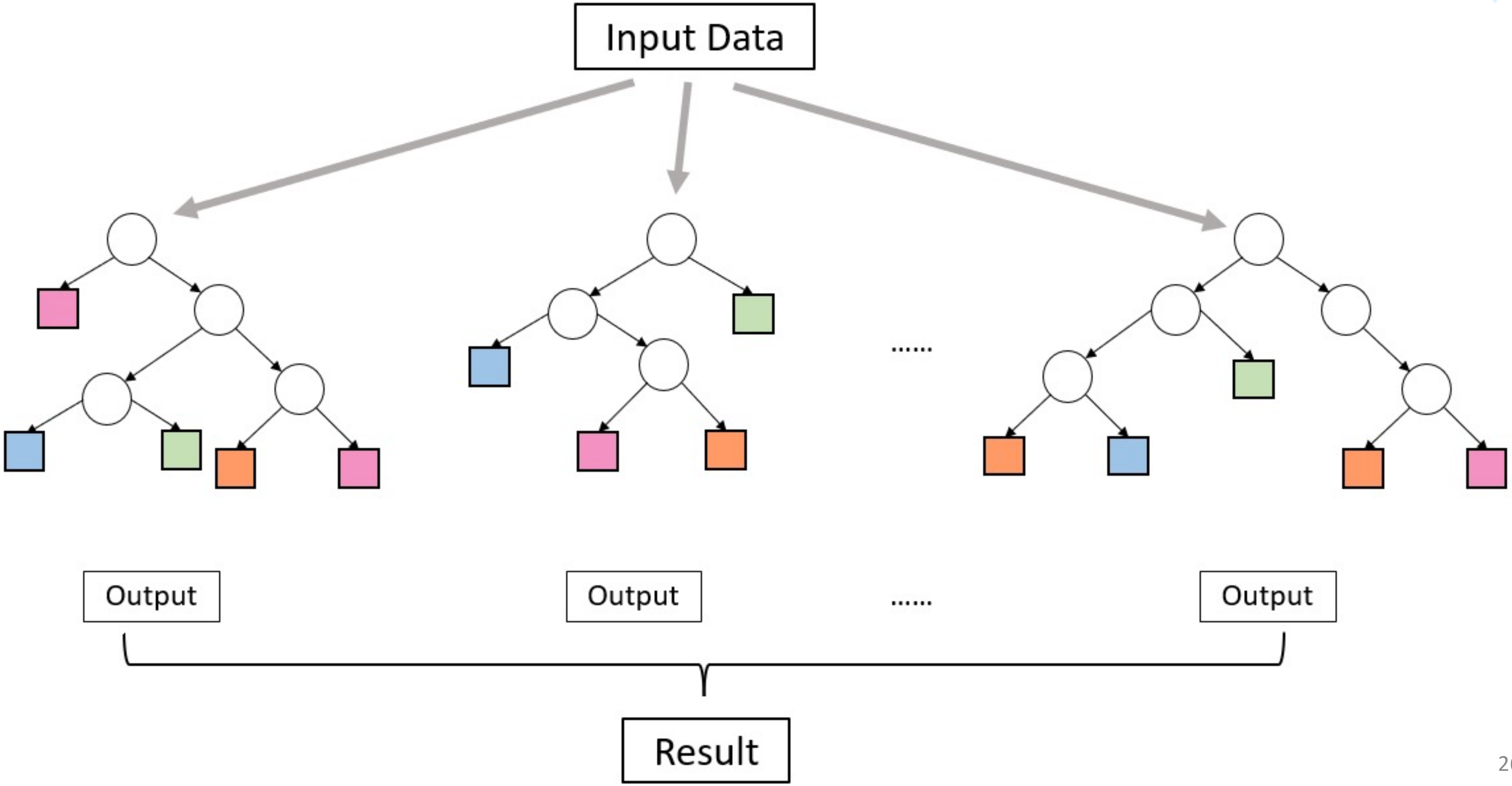
Today's Agenda

- Previous Lecture: Linear Classifiers

- Decision Trees
  - Random Forest
  - Application: SUM

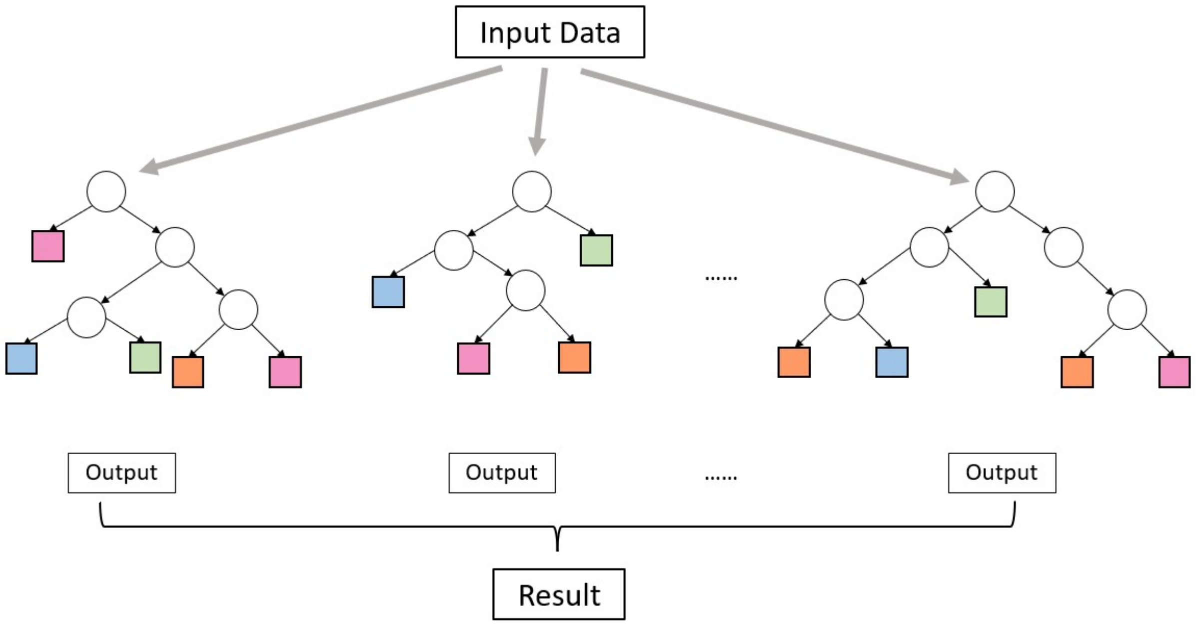
- Data and Features
  - Feature Selection
  - Classifier Evaluation

Random Forest



Random Forest

- Bagging
  - Sample the original dataset with replacement (i.e., for the original set [1,2,3,4,5], we can sample [1,3,4,4,5])
  - Create multiple tree classifiers, each with bagging. Summarize the results using majority vote.



Random Forest

- Feature Randomness
  - Each tree can pick only from a random subset of features
  - This is to further ensure the independence among various trees

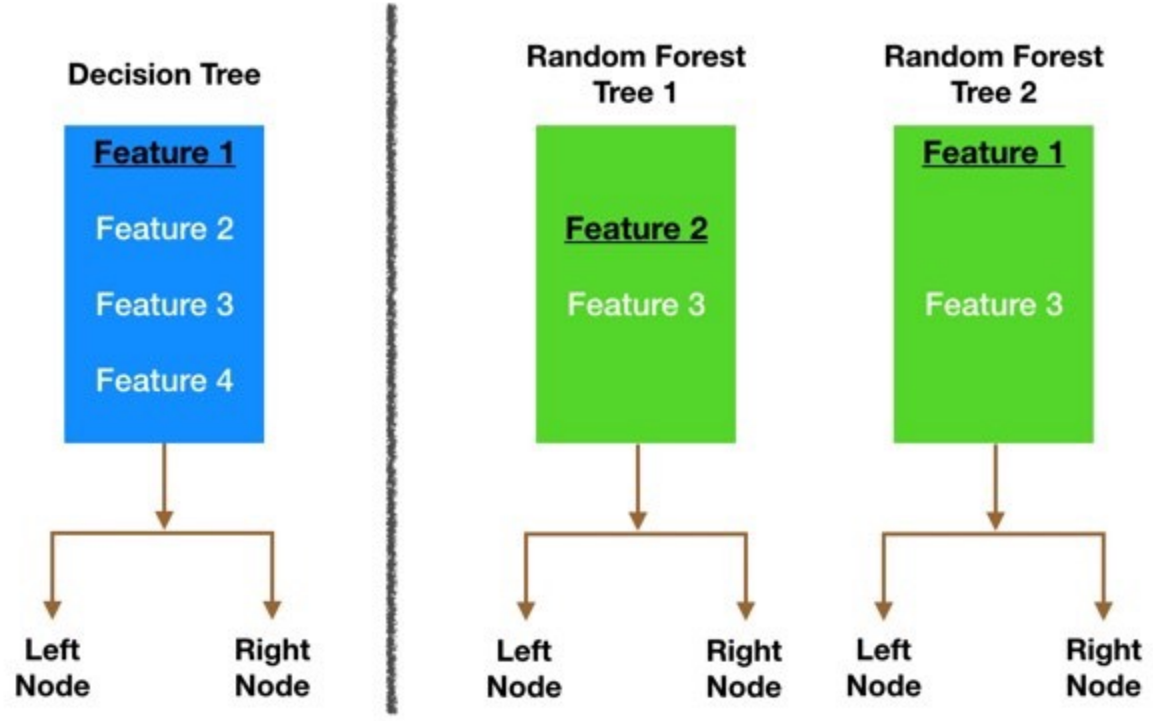


Image source: towards data science

Random Forest Remarks

- Combining relatively uncorrelated classifiers together generally outperforms a single classifier

- Combining models also helps to reduce the variance

- With sufficient amount of trees, RF can achieve comparable performance as neural networks

Today’s Agenda

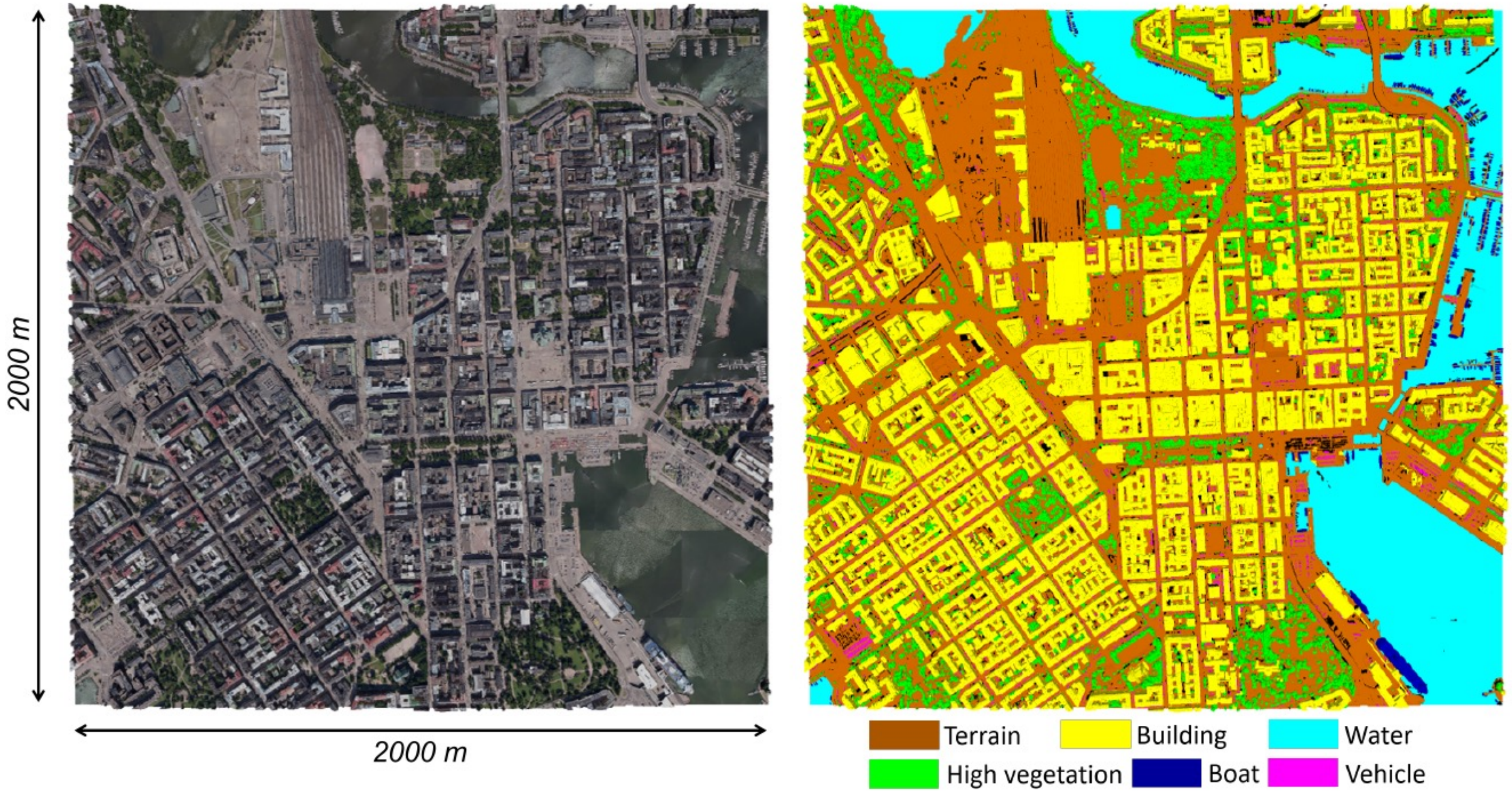
- Previous Lecture: Linear Classifiers

- Decision Trees
  - Random Forest
  - Application: SUM 👈

- Data and Features
  - Feature Selection
  - Classifier Evaluation

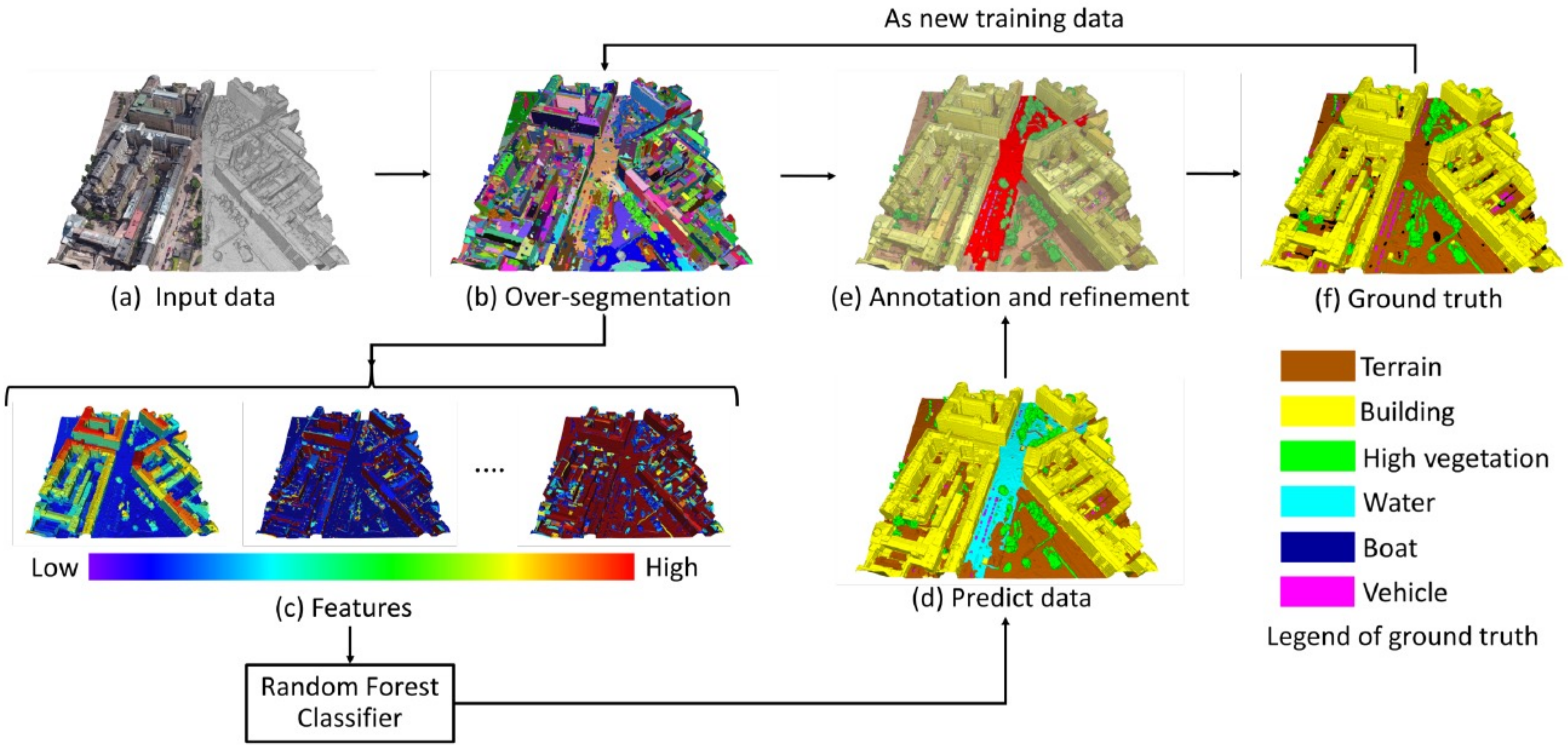
SUM: Semantic Urban Meshes

- Semantic mesh segmentation of urban environment



SUM: Semantic Urban Meshes

- Algorithm workflow



SUM: Semantic Urban Meshes

- Features to use
  - **Eigen features**

    Linearity: $\frac{\lambda_1 - \lambda_2}{\lambda_1}$

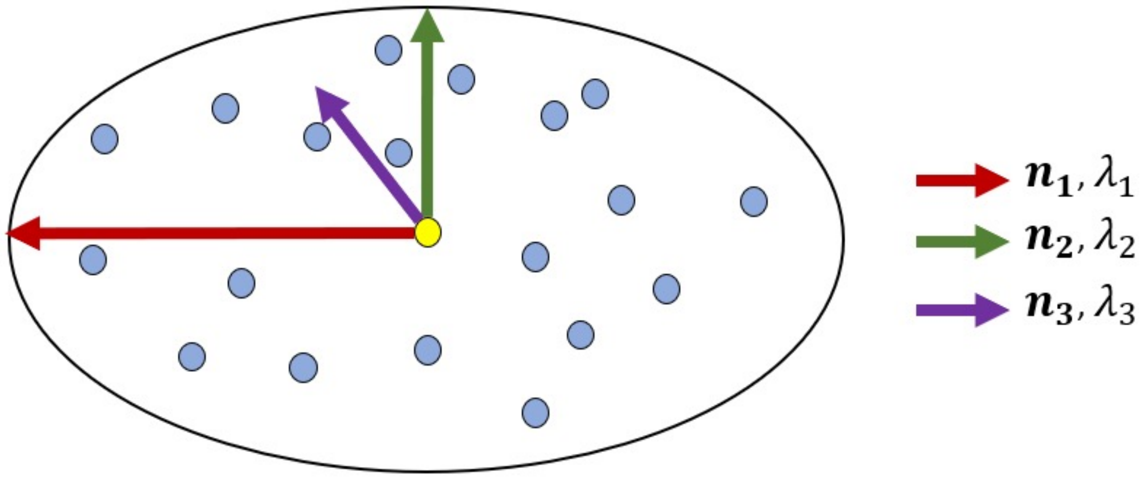
    Sphericity: $\frac{\lambda_3}{\lambda_1}$

    Curvature change: $\frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$

    Verticality: $1 - |\boldsymbol{n_3} \cdot \boldsymbol{n_z}|$

  - **Elevation features**

    Relative elevation: $z - z_{min}$



SUM: Semantic Urban Meshes

- Features to use
  - **Color features**

    RGB (HSV) colors

    Color variance within a local neighborhood

  - **Other mesh-based features**

    Mesh area

    Triangle density

    ……

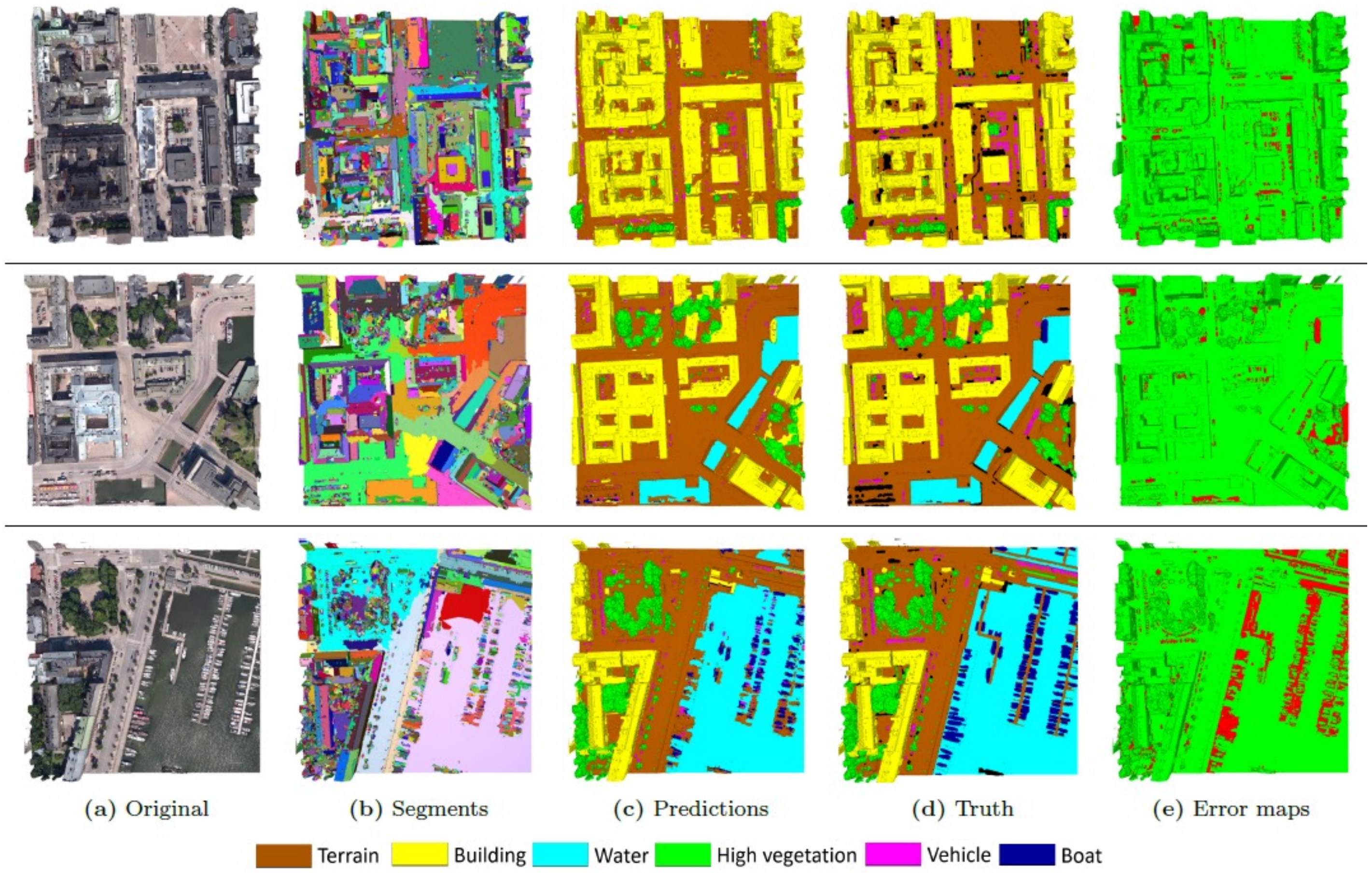
SUM: Semantic Urban Meshes

- Segmentation performance compared with deep learning methods

	Terrain	High Vegeta-tion	Building	Water	Vehicle	Boat	mIoU	OA	mAcc	mF1	t_{train}
PointNet [14]	56.3	14.9	66.7	83.8	0.0	0.0	36.9 ± 2.3	71.4 ± 2.1	46.1 ± 2.6	44.6 ± 3.2	1.8
RandLaNet [53]	38.9	59.6	81.5	27.7	22.0	2.1	38.6 ± 4.6	74.9 ± 3.2	53.3 ± 5.1	49.9 ± 4.8	10.8
SPG [15]	56.4	61.8	87.4	36.5	34.4	6.2	47.1 ± 2.4	79.0 ± 2.8	64.8 ± 1.2	59.6 ± 1.9	17.8
PointNet++ [52]	68.0	73.1	84.2	69.9	0.5	1.6	49.5 ± 2.1	85.5 ± 0.9	57.8 ± 1.8	57.1 ± 1.7	2.8
RF-MRF [43]	77.4	87.5	91.3	83.7	23.8	1.7	60.9 ± 0.0	91.2 ± 0.0	65.9 ± 0.0	68.1 ± 0.0	**1.1**
KPConv [16]	**86.5**	88.4	**92.7**	77.7	**54.3**	**13.3**	**68.8** ± 5.7	**93.3** ± 1.5	**73.7** ± 5.4	**76.7** ± 5.8	23.5
Baseline	83.3	**90.5**	92.5	**86.0**	37.3	7.4	66.2 ± 0.0	93.0 ± 0.0	70.6 ± 0.0	73.8 ± 0.0	1.2

SUM: Semantic Urban Meshes

- Visualization of the (part) result

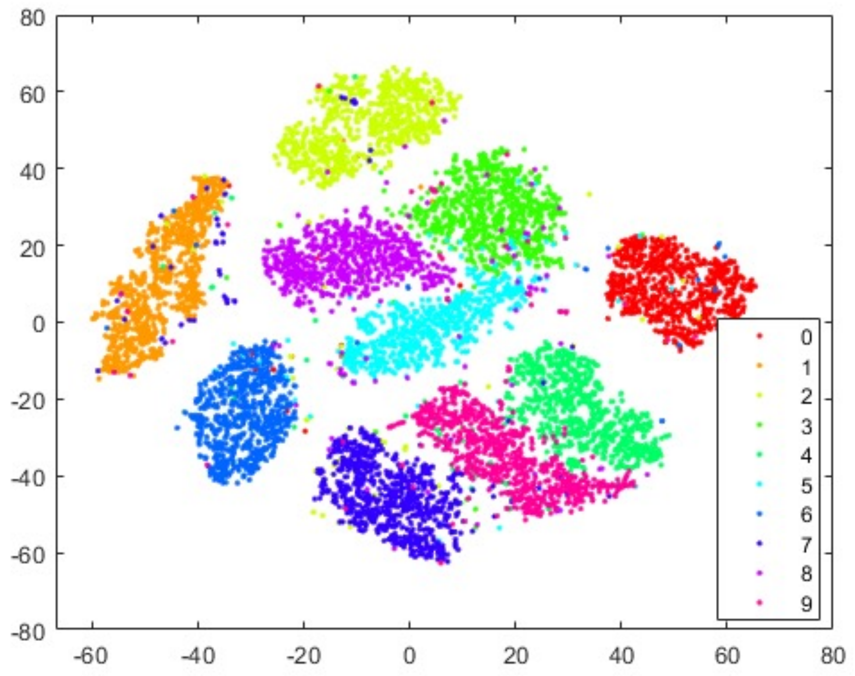


Today’s Agenda

- Previous Lecture: Linear Classifiers
- Decision Trees
    - Random Forest
    - Application: SUM
- Data and Features
    - Feature Selection
    - Classifier Evaluation

Data and Features

- Will more features lead to better performance?



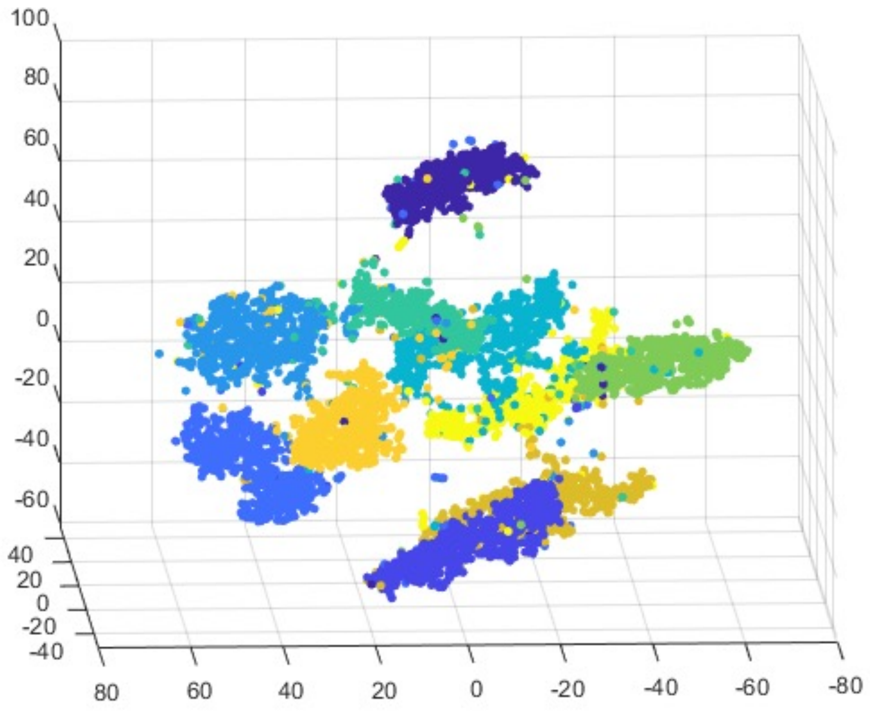


Image Source: https://www.mathworks.com/help/stats/visualize-high-dimensional-data-using-t-sne.html

Data and Features

- Curse of dimensionality
  - Too few samples in too high dimensional space
- Computation complexity
- Feature correlations
  - 1+1 is not always larger than 2

Today’s Agenda

- Previous Lecture: Linear Classifiers
- Decision Trees
  - Random Forest
  - Application: SUM
- Data and Features
  - Feature Selection
  - Classifier Evaluation

Feature Selection

- How to measure if a feature subset is good or not?
  - The best is to measure actual classification performance. However, it can be expensive

- How could we select the most important features?
  - Limit the dimensionality (i.e., number of features)
  - Retain the class discriminatory information

Feature Selection

- Scatter matrices for feature selection criterion:
  - **Within-scatter matrix**:

$$S_w = \sum_{k=1}^{K} \frac{N_k}{N} \Sigma_k$$

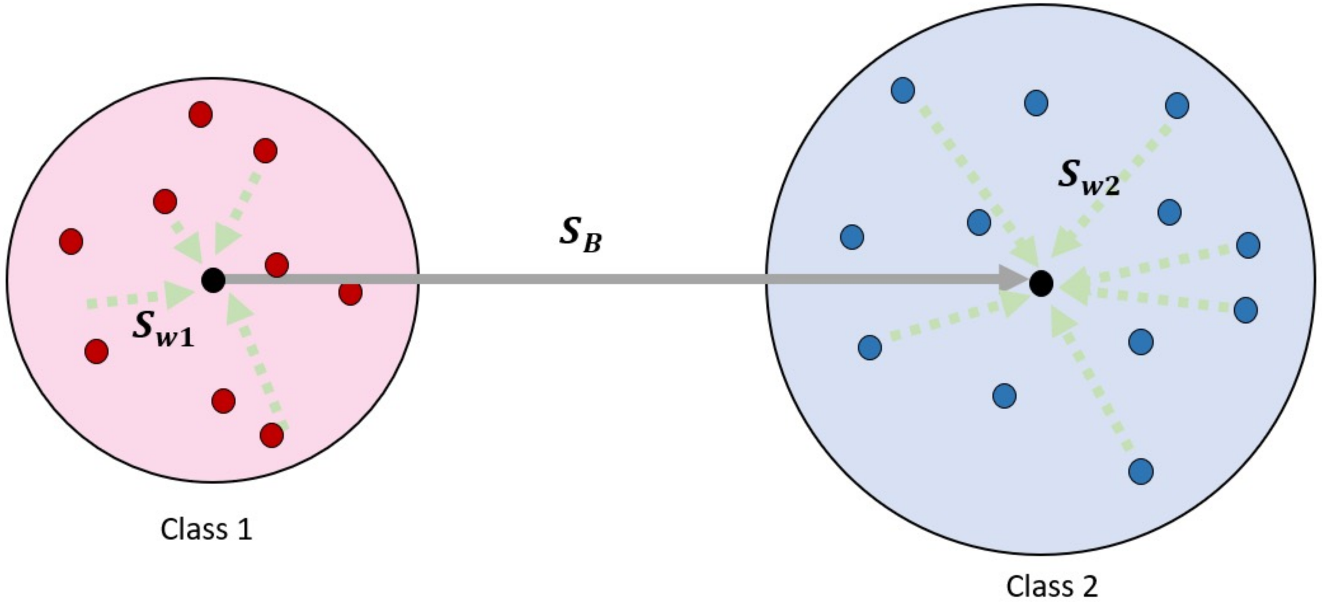
  - **Between-scatter matrix**:

$$S_B = \sum_{k=1}^{K} \frac{N_k}{N} (\boldsymbol{\mu_k} - \boldsymbol{\mu})(\boldsymbol{\mu_k} - \boldsymbol{\mu})^T$$

K: total number of classes
$\boldsymbol{\mu}$: mean of all samples
$\boldsymbol{\mu_k}$, Σ_k: mean and covariance matrix of per-class samples

Feature Selection

- S_w: the lower, the better; S_B: the higher, the better



- There're several ways of combining them, e.g.,

$$J = \frac{tr(S_B)}{tr(S_w)}$$

Feature Selection

- We want to select d out from p features, and choose the subset with optimal criterion value

- How many possible subsets in total?

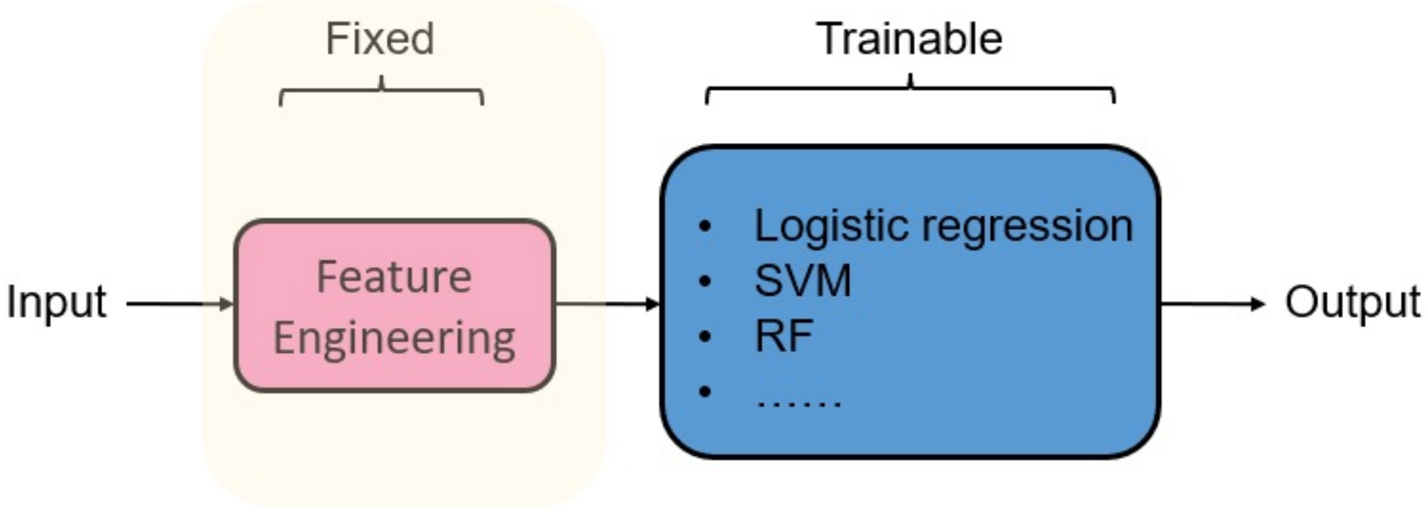
Feature Selection

- Some sub-optimal algorithms to search for the d features:
  - (1) Choose the best individual d features
  - (2) Forward search:
    - Starting with the empty set, each time add one feature that optimizes the entire chosen feature set
  - (3) Backward search:
    - Starting with the whole set, each time drop one feature that optimizes the rest of the feature set

Feature Selection

- Besides feature selection, you can also extract new features by dimension reduction methods (e.g., PCA)

- Feature engineering is the focus of most classical ML methods



Today's Agenda

- Previous Lecture: Linear Classifiers

- Decision Trees
  - Random Forest
  - Application: SUM

- Data and Features
  - Feature Selection
  - Classifier Evaluation

Classifier Evaluation

- Common evaluation metrics:
  - **OA**: overall accuracy
    - Out of 500 objects, how many are correctly classified?
  - **mAcc**: mean per-class accuracy
    - How is the accuracy of each class? Average them.
  - Confusion matrix
  - **mIoU**: mean intersection over union

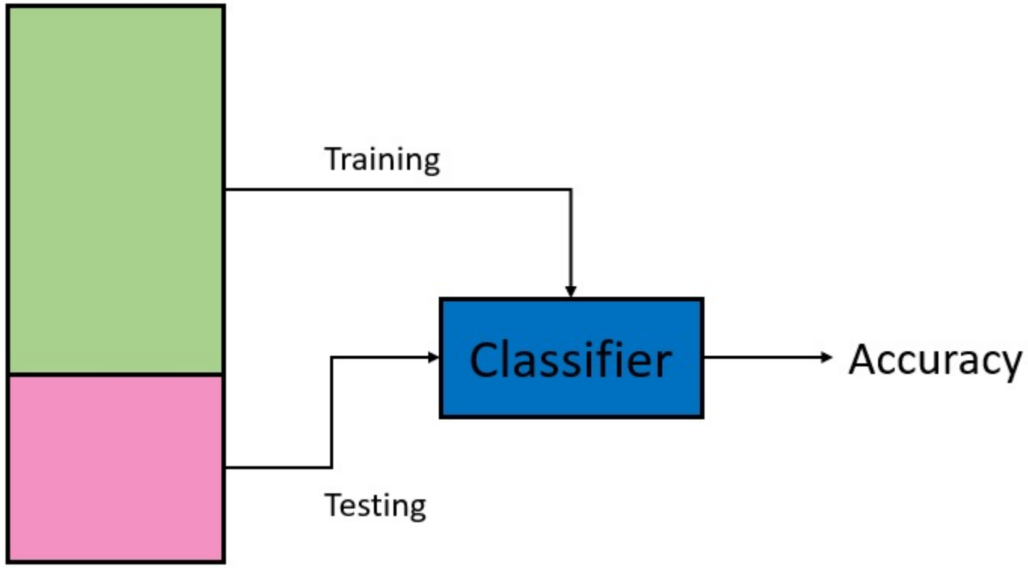
Classifier Evaluation

- Is it good to measure the performance of the classifier in the training dataset? Why?

Classifier Evaluation

- Classification accuracy over training set can be biased, and optimistically estimated

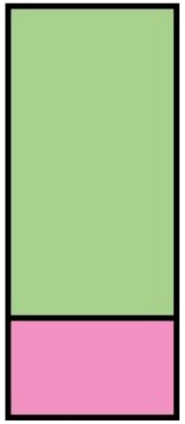
- We're interested in true accuracy of the classifier



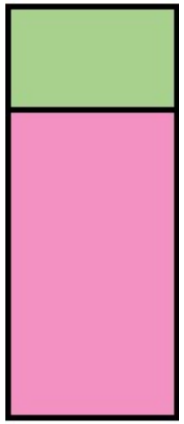
Classifier Evaluation

- Train-test split

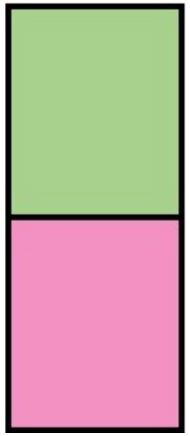
Training and testing on the same set will give a good classifier, but will yield a biased estimate of the model



A small independent test set yields an unbiased, but unreliable accuracy estimate for a well-trained classifier



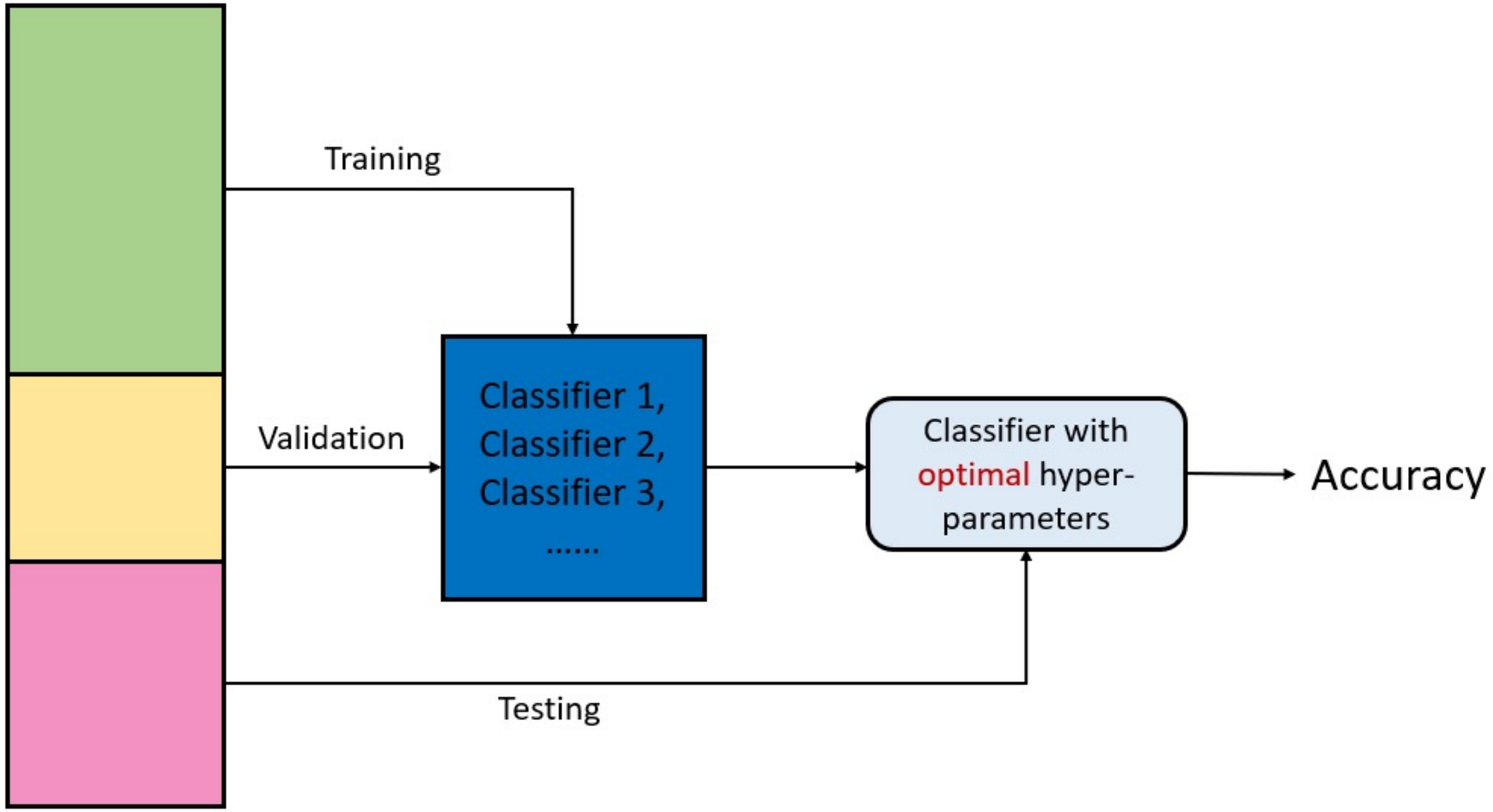
A large, independent test set yields an unbiased and reliable accuracy estimate for a badly trained classifier



7:3, 6:4, 5:5 ratios are commonly used in practice

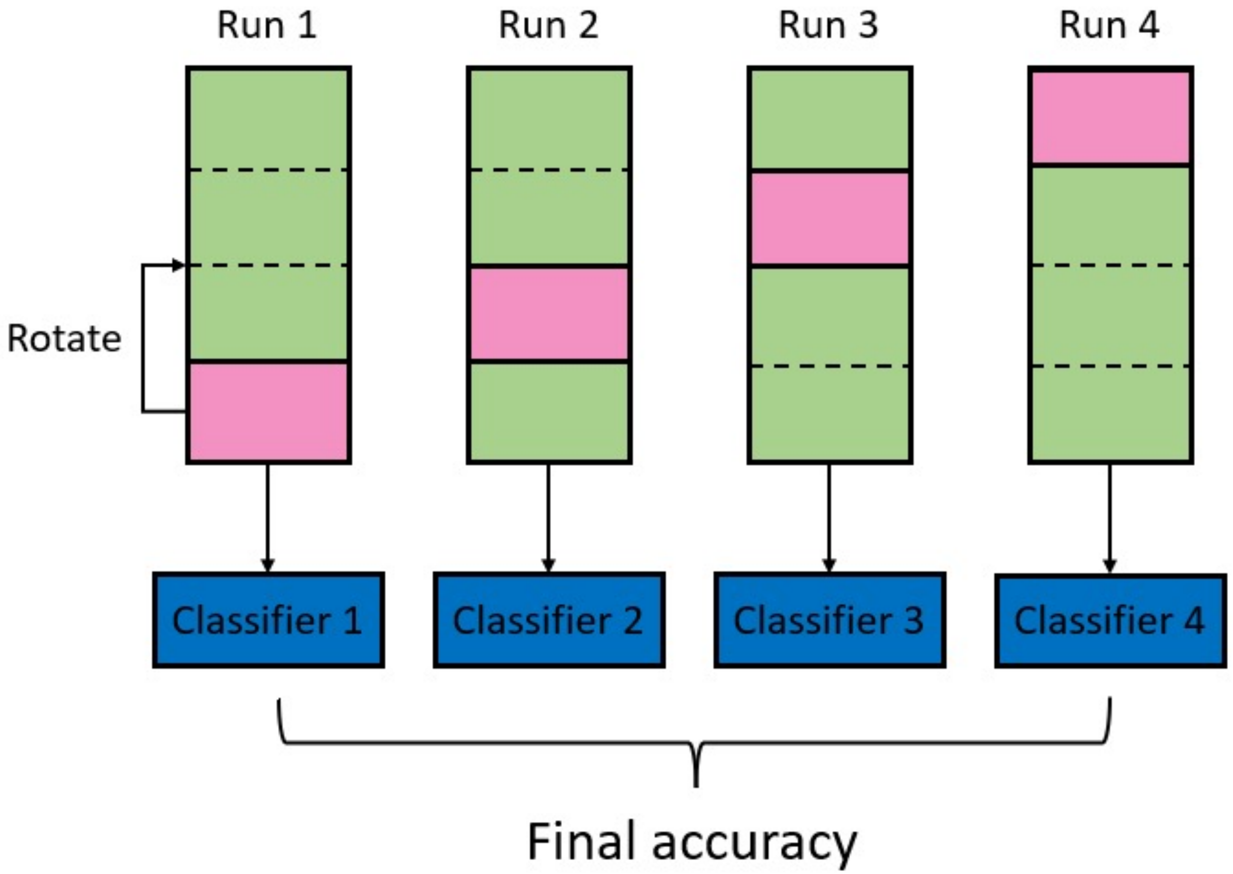
Classifier Evaluation

- Sometimes a validation set is introduced (common in deep learning)



Classifier Evaluation

- Cross Validation: making full use of data



Questions?

3D geoinformation
Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

GEO5017
Machine Learning for the Built Environment

Lab Session
RF Practice in Scikit Learn

Shenglan Du

Review: SVM in Scikit Learn

- SVM has 3 classifiers
  - **SVC**: most commonly used in practice
  - **NuSVC**: similar to SVC, has slightly different yet equivalent mathematical formulations and parameter set
  - **LinearSVC**: faster implementation of SVM, but can only adopt linear kernels

SVC: Documentation

sklearn.svm.SVC

class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None) [source]

- The most important hyper-parameters:
  - C: the coefficient introduced in soft-margin SVM
  - Kernel: a trick you can use to transform input features

SVC: Documentation

sklearn.svm.SVC

class sklearn.svm.SVC(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

- Other important hyper-parameters:
  - Class_weight: specify the weight per class. You either input a dictionary of pre-fixed weights, or use ‘balanced’.
  - Max_iter: hard limit on iterations within solver, or -1 for no limit.
  - Decision_function_shape:
    - **‘ovr’**: default, one versus the rest for multi-class
    - **‘ovo’**: one versus one for multi-class

SVC vs LinearSVC

- SVC(kernel=linear) and LinearSVC both generate linear decision boundaries

- LinearSVC is faster implementation. Also, it uses slightly different loss functions.

- Both SVC and LinearSVC involves parameter tuning. Tutorials of parameter tuning can be found here:
  https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769

RF in Scikit Learn

sklearn.ensemble.RandomForestClassifier ¶

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

- **Ensemble** means RF is a collection of individual tree classifiers
- n_estimators: number of trees in the forest
- Criterion: splitting criterion
- max_features: the number of features in each tree to start splitting

RF in Scikit Learn

sklearn.ensemble.RandomForestClassifier ¶

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

- Bootstrap: whether bagging is used for building the trees
- max_samples: if bootstrap is true, then this is to determine how many max samples to draw from the original dataset (with replacement) to building each tree

Some useful functions

- Train test split

sklearn.model_selection.train_test_split¶

sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None) [source]

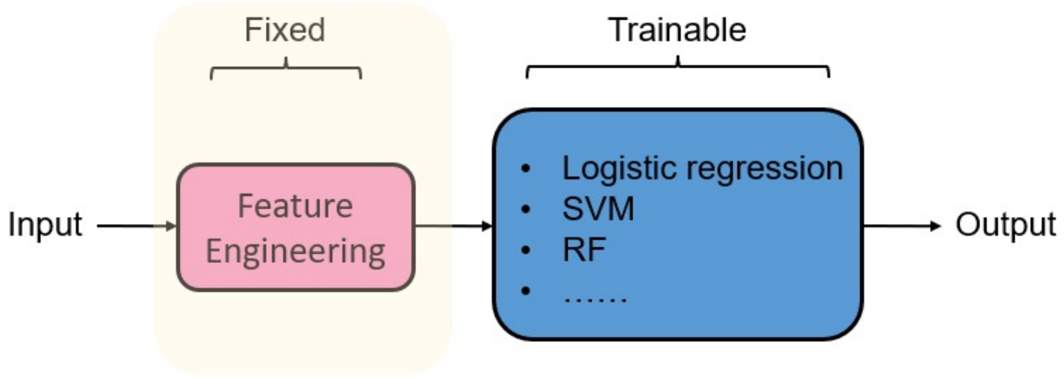
- Evaluation

sklearn.metrics.accuracy_score

sklearn.metrics.accuracy_score(y_true, y_pred, *, normalize=True, sample_weight=None) [source]

A2: Point Cloud Classification

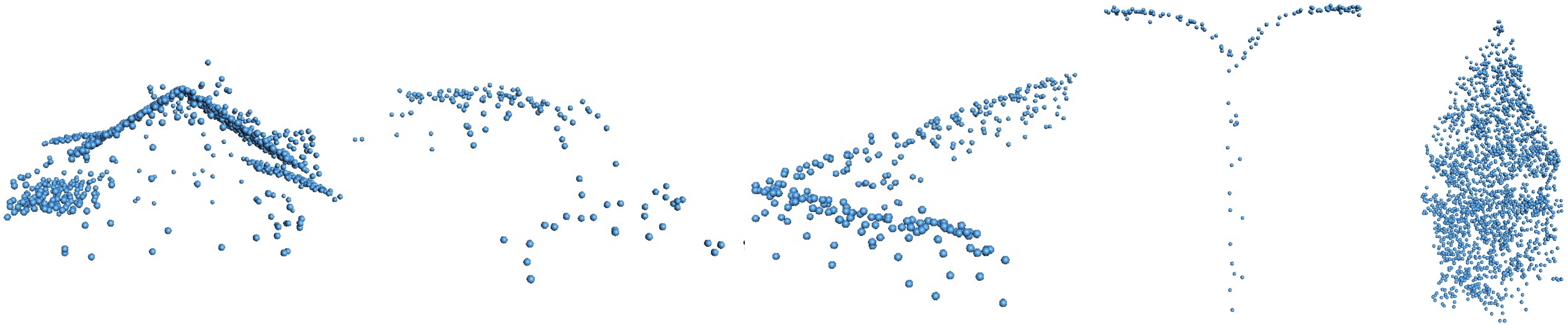
- Feature engineering is the most important part



- It's not mandatory to implement the feature selection techniques (i.e., S_W and S_B matrices), however the feature visualization should help

A2: Point Cloud Classification

- Good features should:
  - Describe the intrinsic similarity within the same class
  - Distinguish as much as possible between classes
  - With very good features, linear classifiers might work better than non-linear classifiers



A2: Point Cloud Classification

- We focus on geometrical properties of the objects
- You can use a subset of the point cloud, or a patch, to describe the object
- We don’t evaluate your work only based on accuracy, we focus more on your analysis / feedback. If your algorithm fails, it’s fine. Please provide your insights and reflections on that

Questions?