上海科技大学
ShanghaiTech University

CS182: Introduction to Machine Learning – k-Nearest Neighbors & Model Selection

Yujiao Shi
SIST, ShanghaiTech
Spring, 2025

Slides Courtesy of Matt Gormley & Henry Chai, CMU

Recall...

Decision Tree: Pseudocode

def train($\mathcal{D}$):

  store root = tree_recurse($\mathcal{D}$)

def tree_recurse($\mathcal{D}'$):

  q = new node()

  base case – if (SOME CONDITION):

  recursion – else:

    find best attribute to split on, x_d

    q.split = x_d

    for v in $V(x_d)$, all possible values of x_d:

$$\mathcal{D}_v = \left\{\left(x^{(n)}, y^{(n)}\right) \in \mathcal{D} \mid x_d^{(n)} = v\right\}$$

      q.children(v) = tree_recurse($\mathcal{D}_v$)

  return q

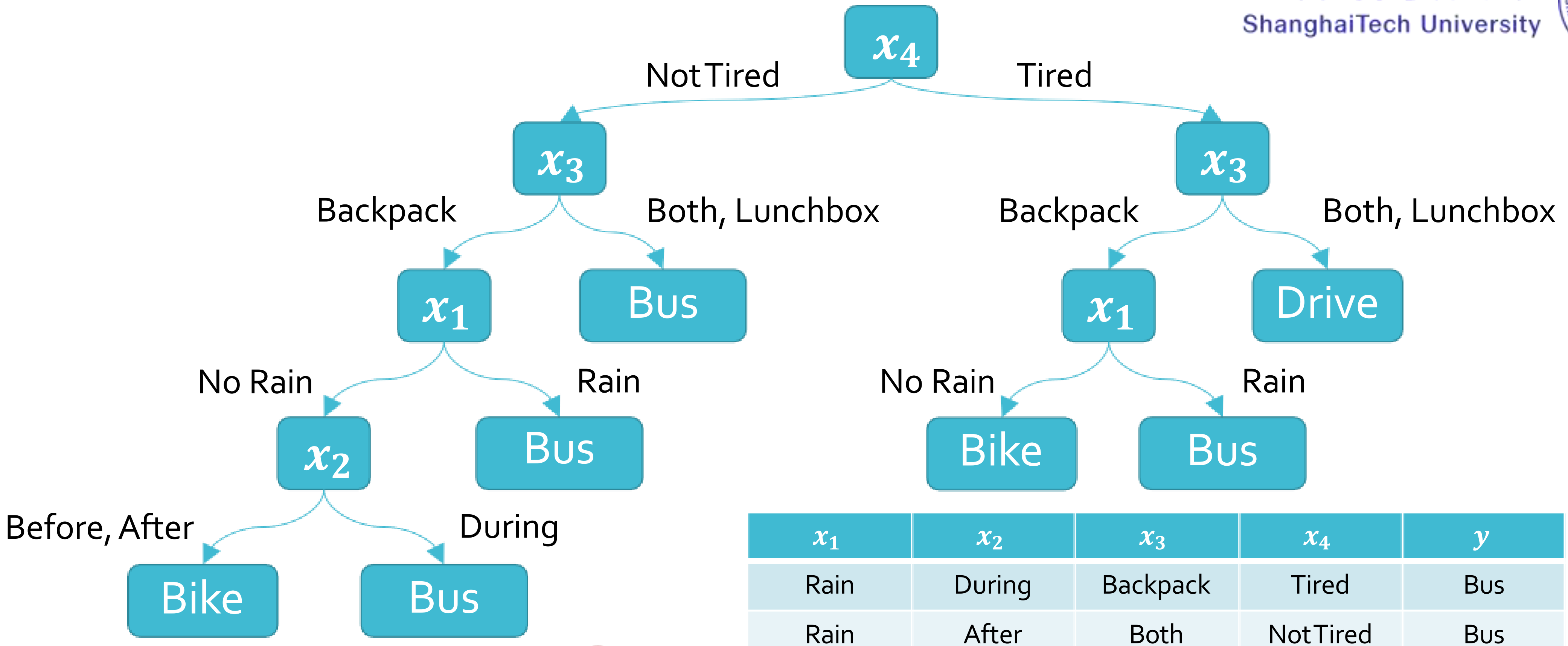
Recall...

上海科技大学 ShanghaiTech University

Decision Tree: Pseudocode

```
def train(𝒟):
    store root = tree_recurse(𝒟)
def tree_recurse(𝒟′):
    q = new node()
    base case – if (𝒟′ is empty OR
       all labels in 𝒟′ are the same OR
       all features in 𝒟′ are identical OR
       some other stopping criterion):
          q.label = majority_vote(𝒟′)

    recursion – else:
    return q
```

上海科技大学
ShanghaiTech University



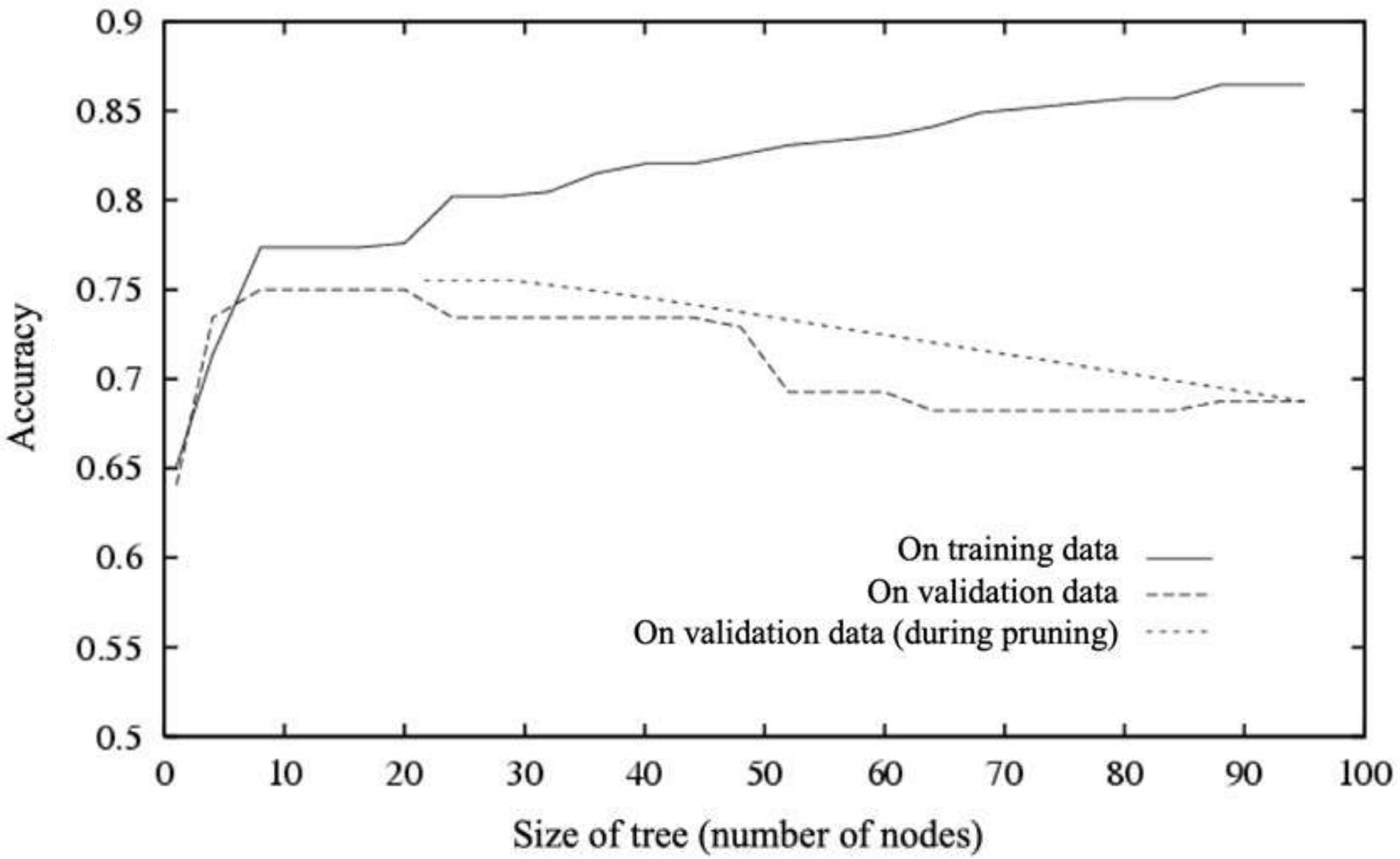
$\mathcal{D}_{val} =$

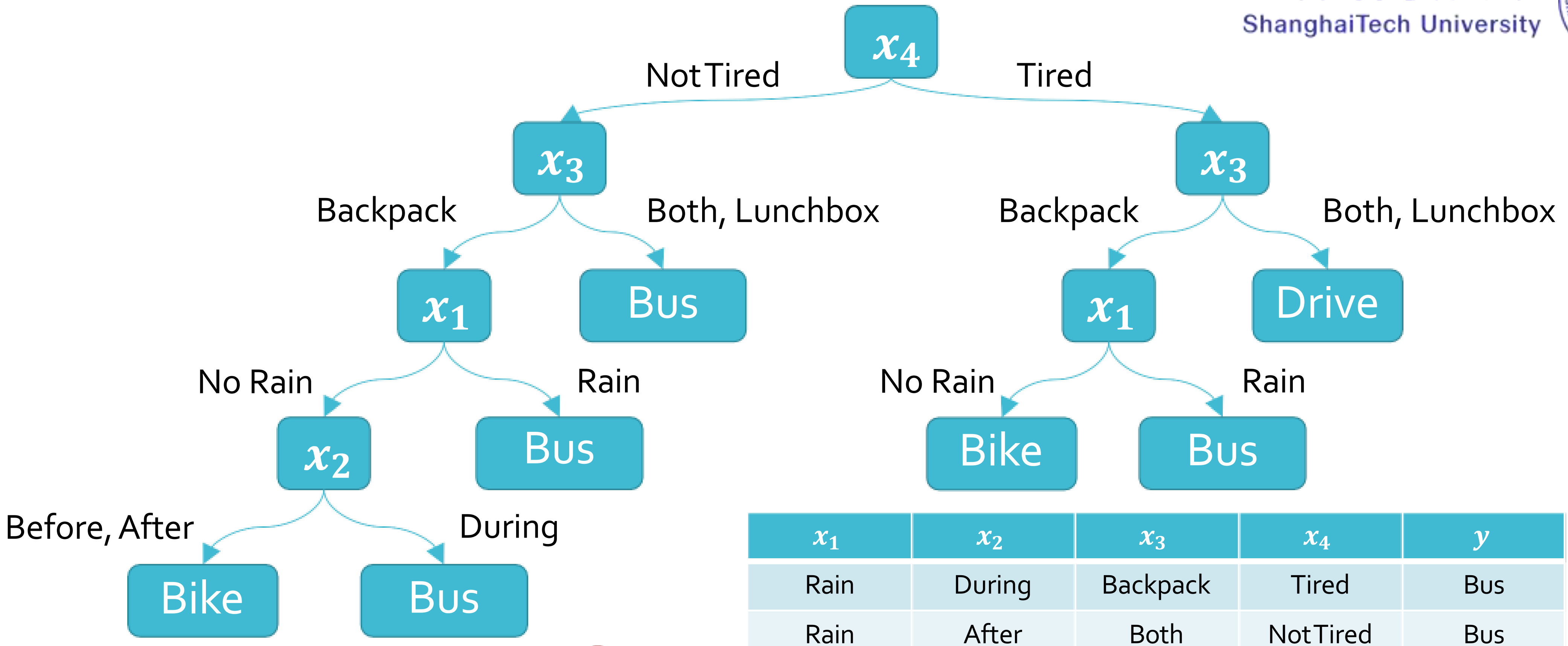
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	NotTired	Bus
No Rain	Before	Backpack	NotTired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

Combatting Overfitting in Decision Trees

Recall...

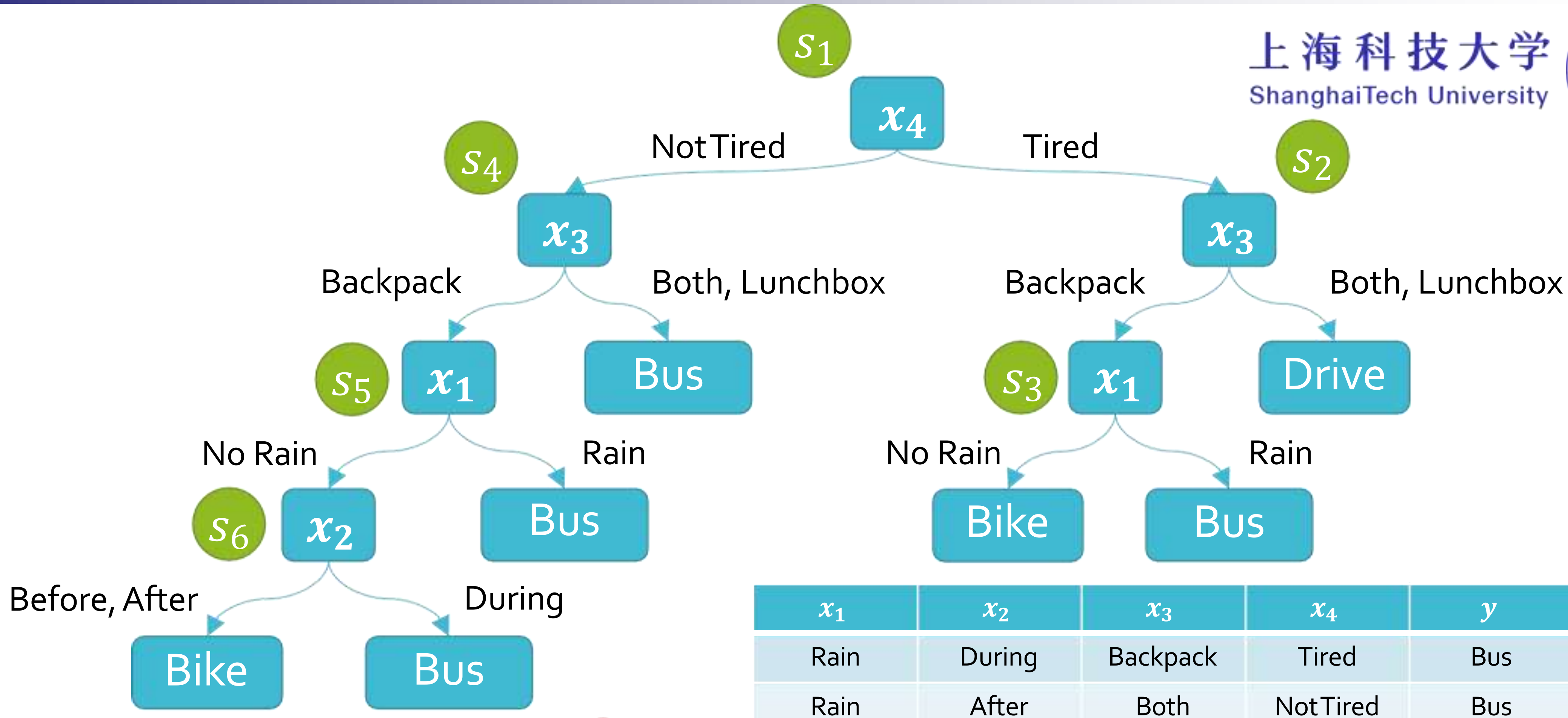
1. Split data in two: *training* dataset and *validation* dataset
2. Grow the full tree using the *training* dataset
3. Repeatedly prune the tree:
   - Evaluate each split using a *validation* dataset by comparing the validation error rate **with and without** that split
   - (Greedily) remove the split that most decreases the validation error rate
   - Stop if no split improves validation error, otherwise repeat





$\mathcal{D}_{val} =$

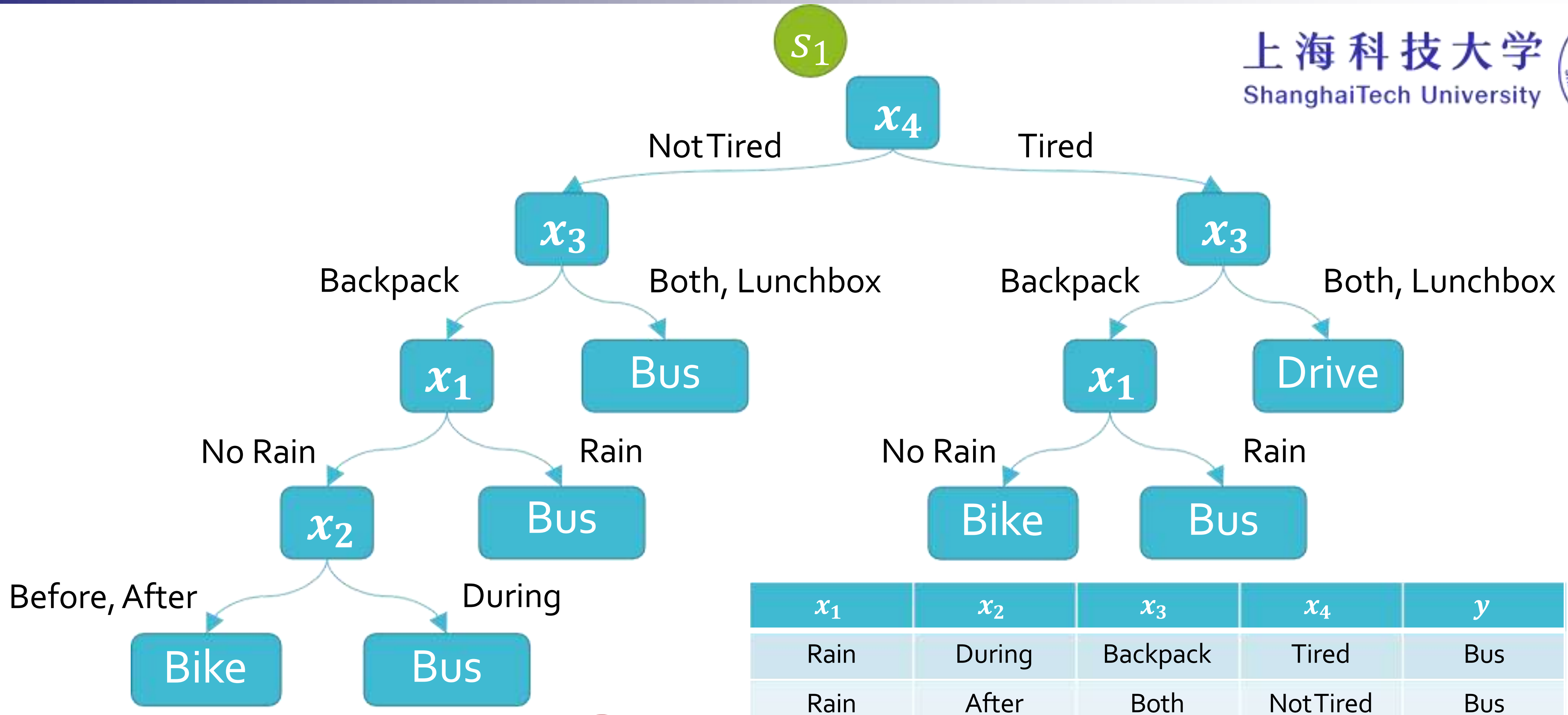
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive



$$\mathcal{D}_{val} =$$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

$$err(h, \mathcal{D}_{val}) = 0.2$$



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

$err(h - s_1, \mathcal{D}_{val})$

$$\mathcal{D}_{val} =$$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

$$err(h - s_1, \mathcal{D}_{val})$$

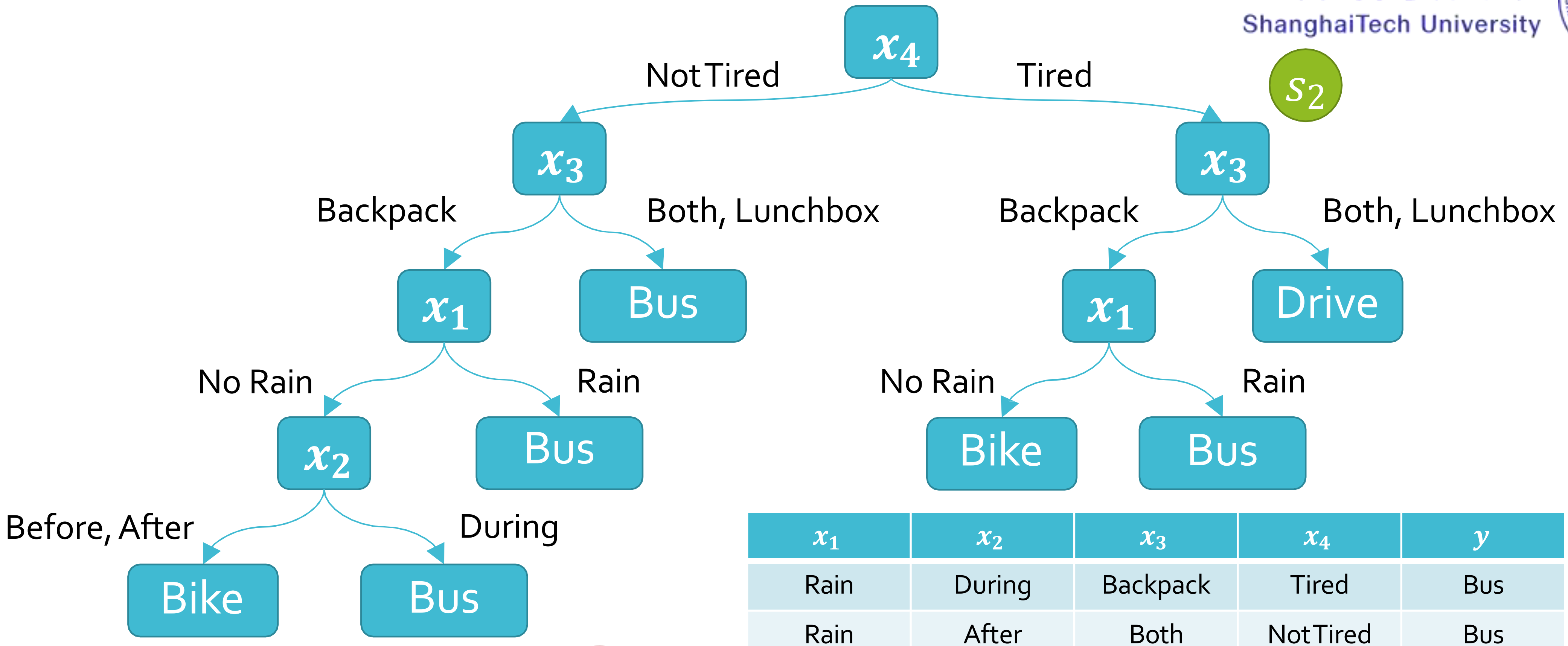
Slide from Henry Chai

$$\mathcal{D}_{val} =$$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

$$err(h - s_1, \mathcal{D}_{val}) = 0.4$$

Slide from Henry Chai

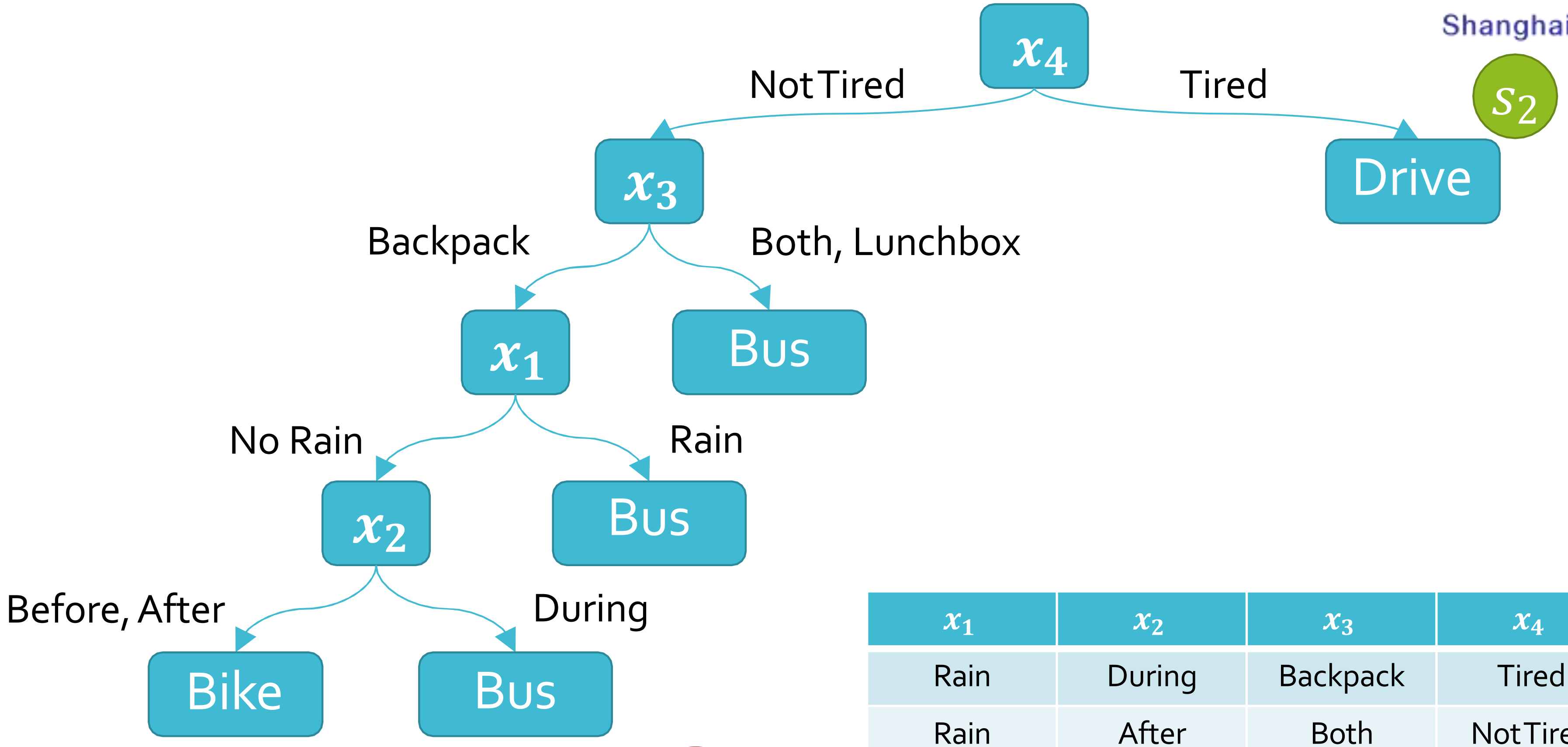


$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

$err(h - s_2, \mathcal{D}_{val})$

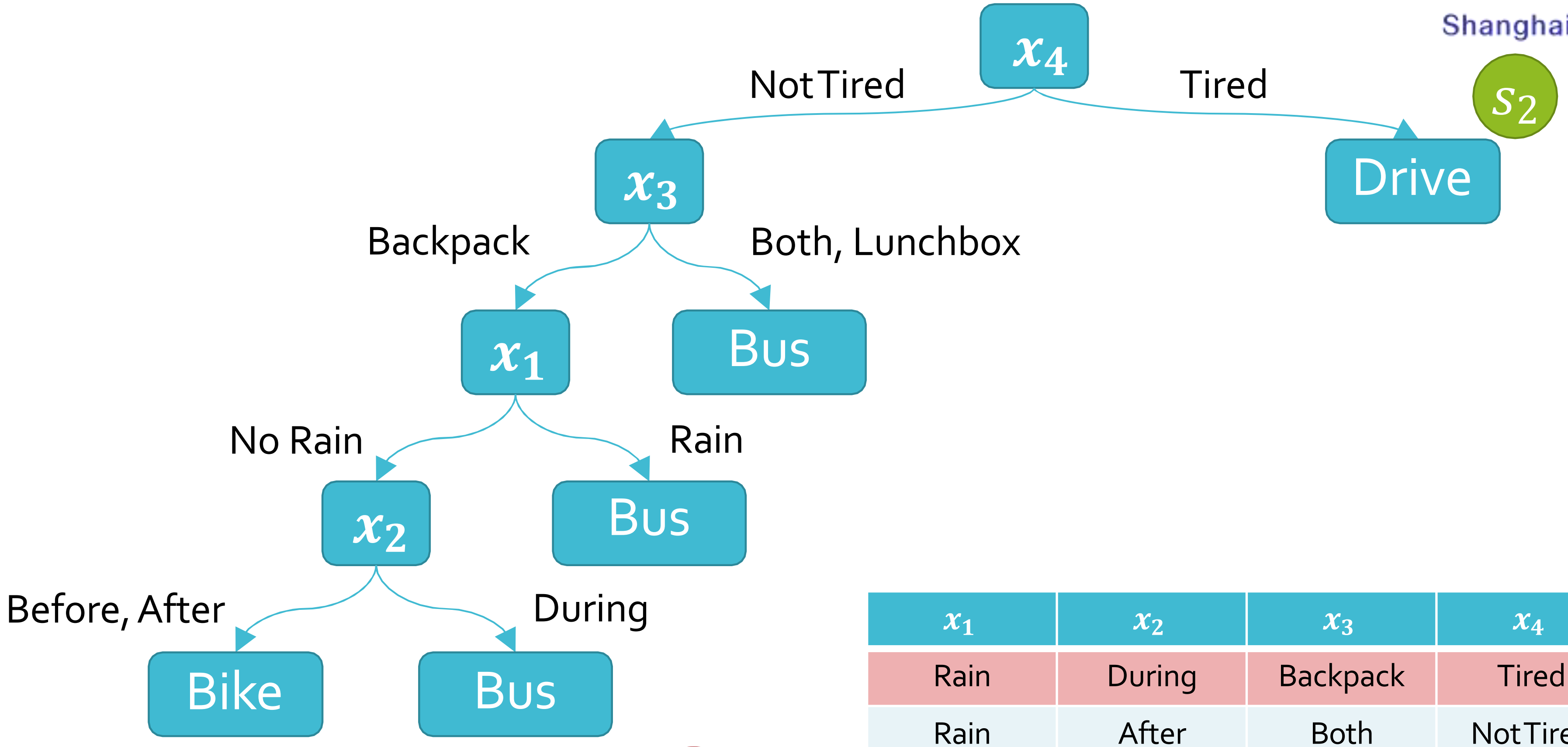
Slide from Henry Chai



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

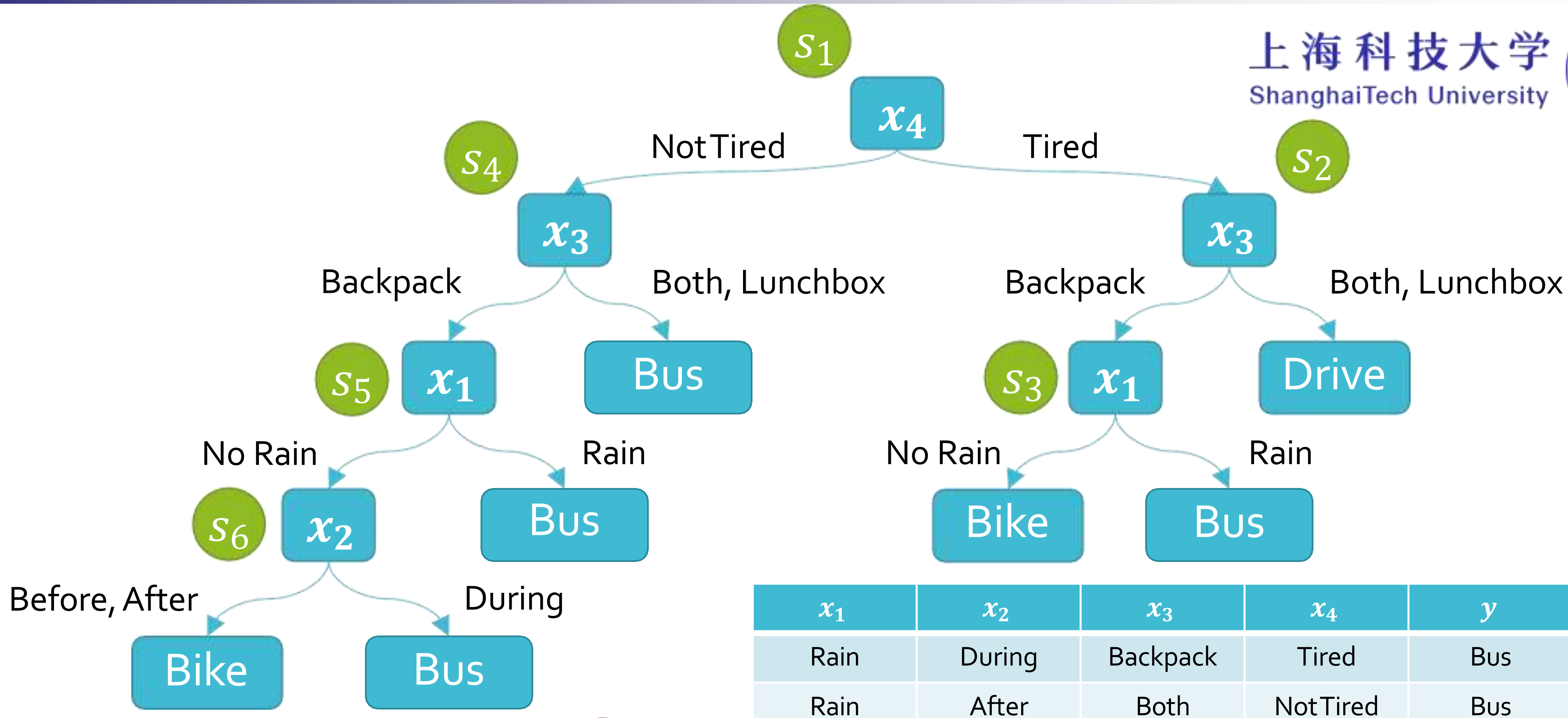
$err(h - s_2, \mathcal{D}_{val})$



$$\mathcal{D}_{val} =$$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

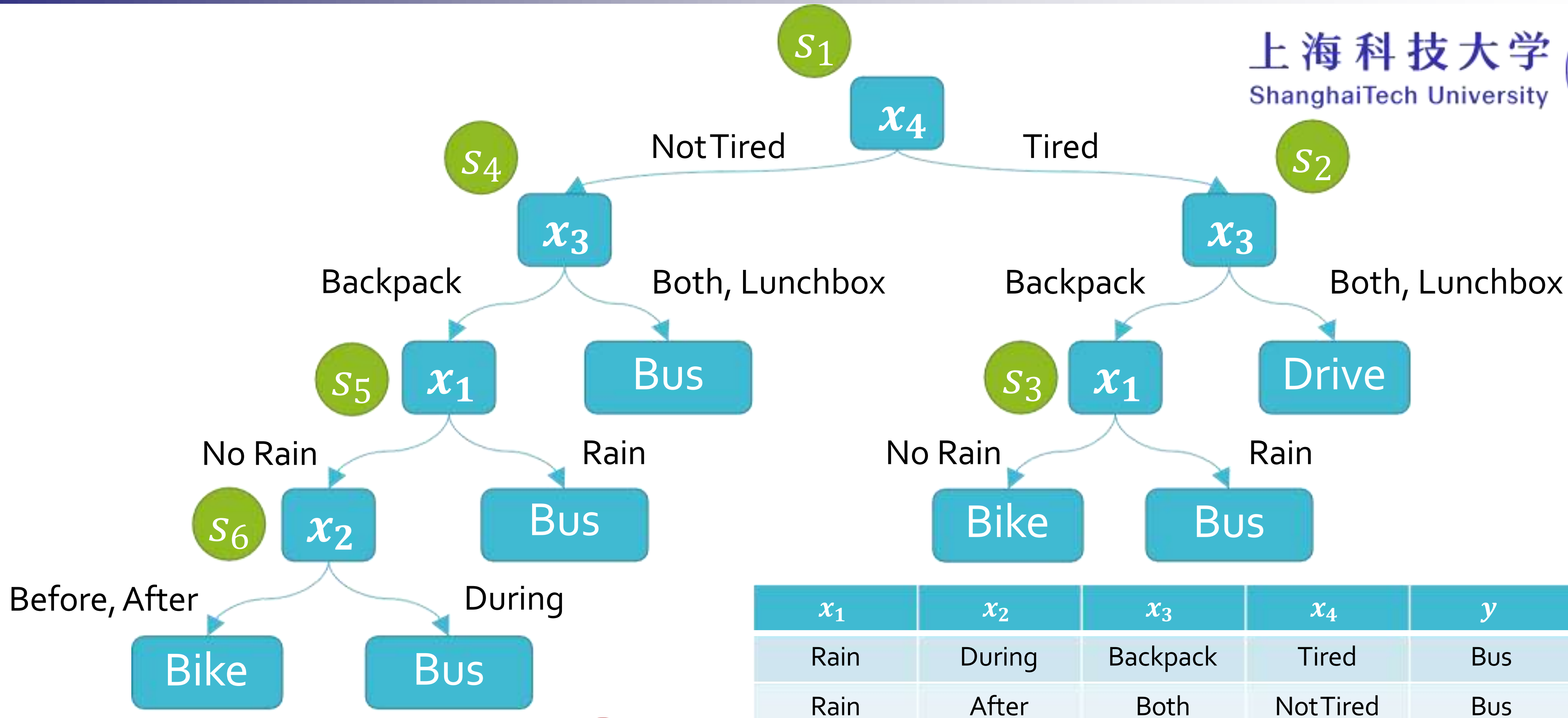
$$err(h - s_2, \mathcal{D}_{val}) = 0.4$$



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

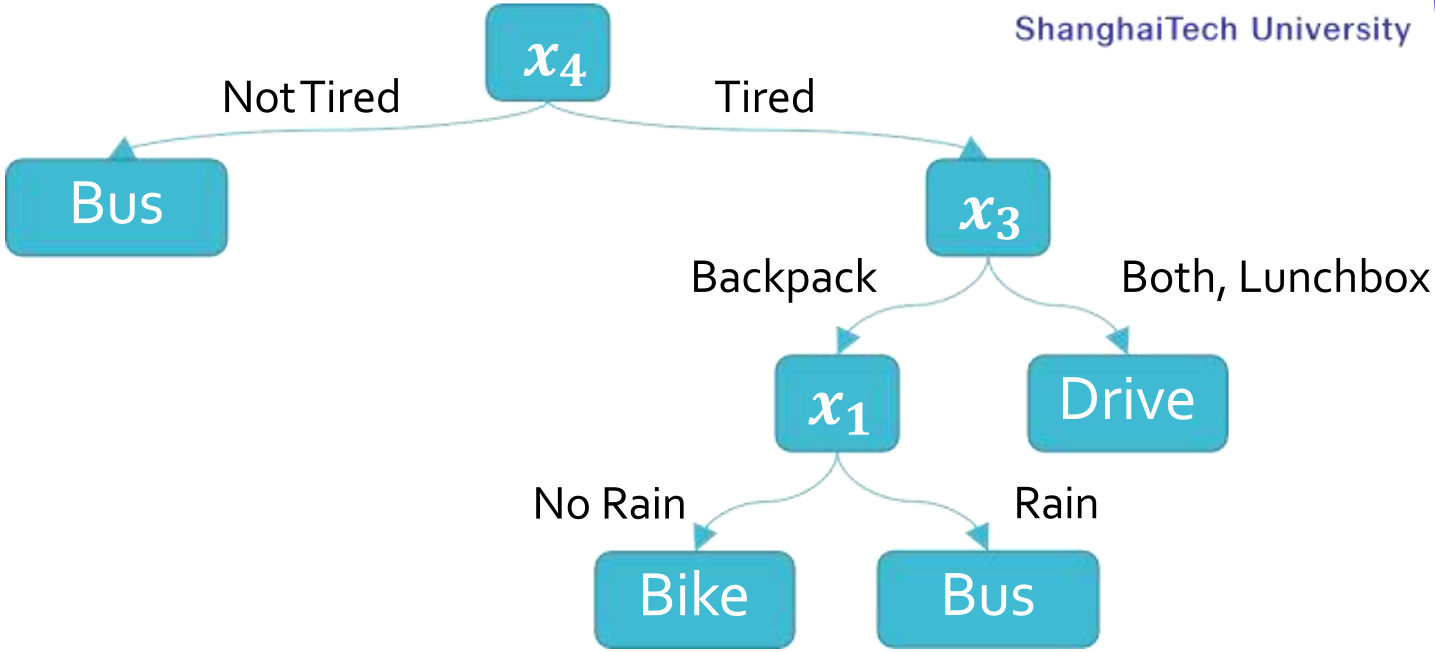
s	s_1	s_2	s_3	s_4	s_5	s_6
$err(h - s, \mathcal{D}_{val})$	0.4	0.4	0.4	0	0	0



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

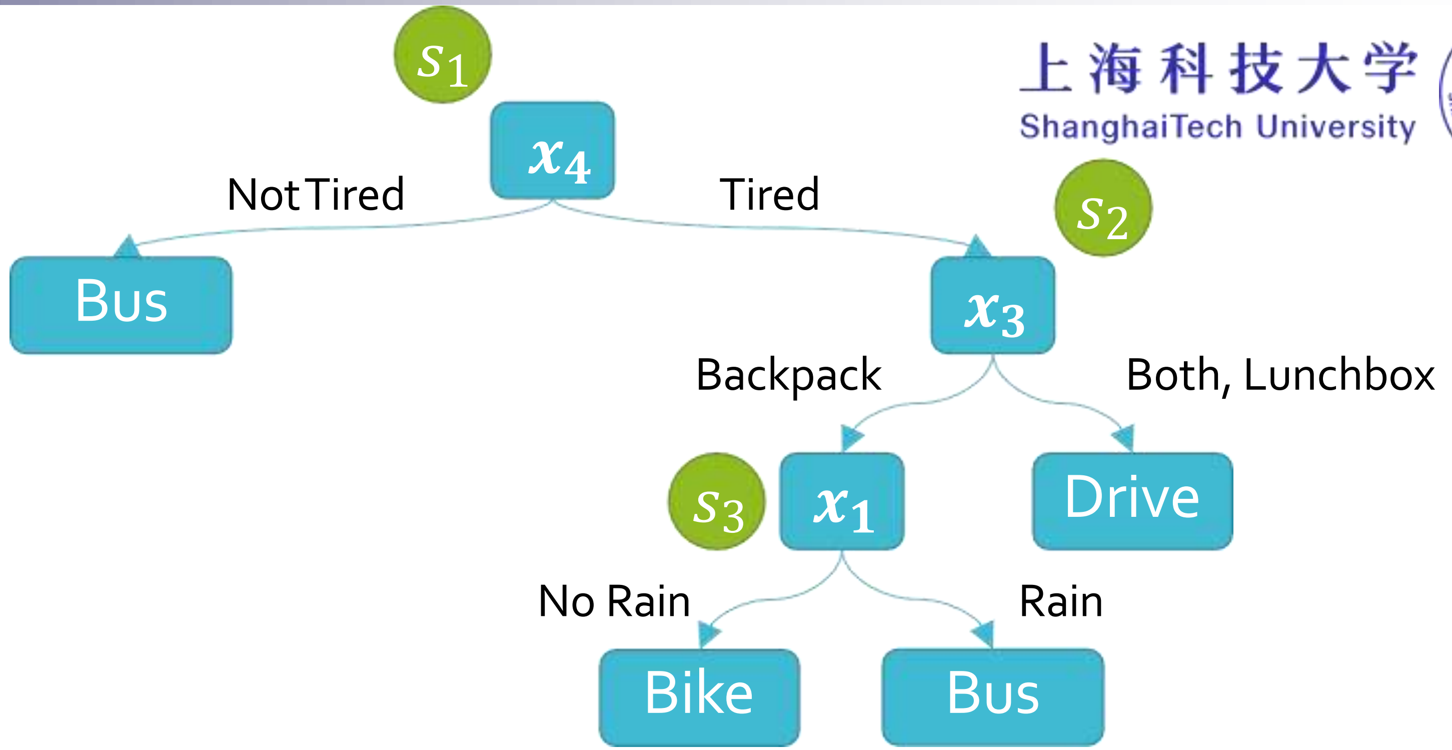
s	s_1	s_2	s_3	s_4	s_5	s_6
$err(h - s, \mathcal{D}_{val})$	0.4	0.4	0.4	0	0	0



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

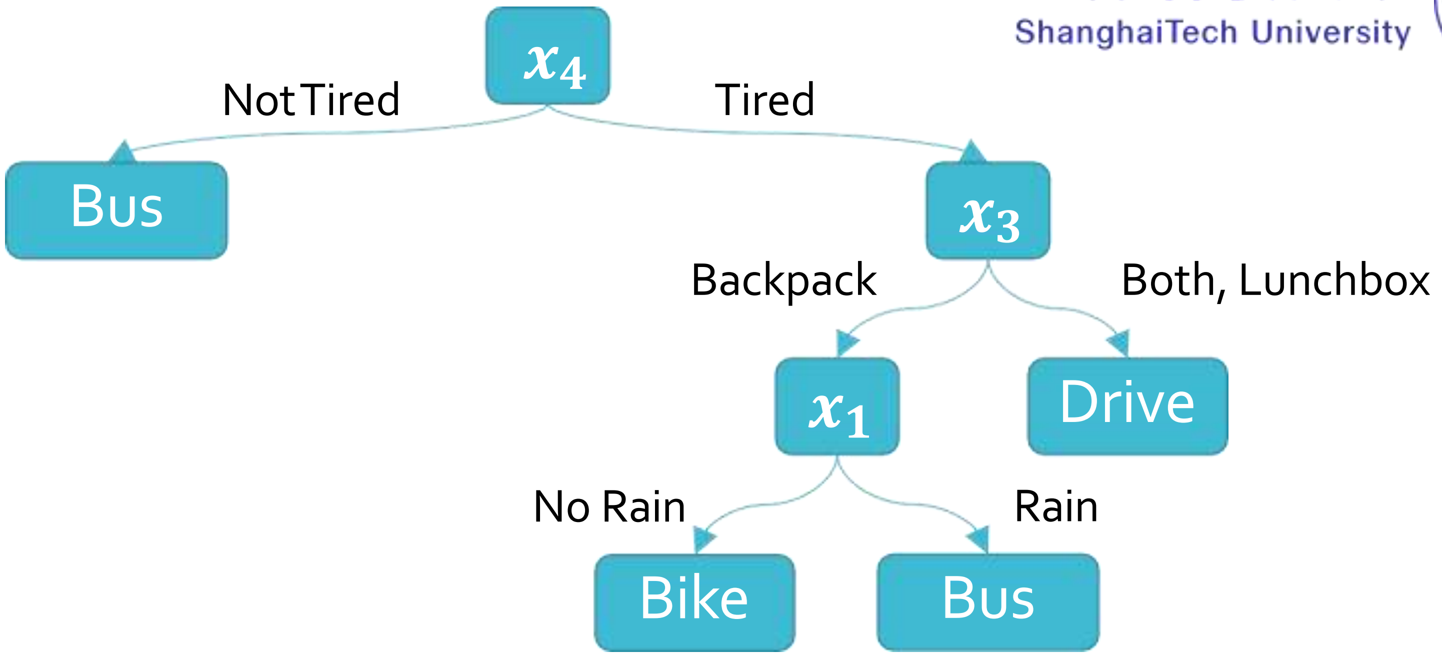
$err(h, \mathcal{D}_{val}) = 0$



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

s	s_1	s_2	s_3
$err(h - s, \mathcal{D}_{val})$	0.4	0.2	0.2


x_4
Not Tired
Tired
Bus
x_3
Backpack
Both, Lunchbox
x_1
Drive
No Rain
Rain
Bike
Bus

Recall…

Q & A:

Wait, how could we end up calling `tree_recurse` with an empty dataset in the first place?

- Given some subset of our dataset, it could be the case that we choose to split on some feature where not every value that the feature could take on appears in the subset
    - This could happen if we know something about the feature *a priori* or we observe that the feature takes on more values in a different subset/the entire training dataset.
- In this case we would still want to make a branch for it in our decision tree because at inference time, some new data point might come along that goes down that branch

Recall...

Q & A:

Okay, so what should we predict in leaf nodes with no training data?

- Well, there isn't really a majority label, so we could return a random label or a majority vote over the entire training dataset.
- This is related to the question of "what should we predict if some feature in our test dataset takes on a value we didn't observe in the training dataset?"
  - Going down a branch corresponding to an unseen feature value is like hitting a leaf node with no training data.

Real-valued Features

Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
0	4.3	3.0	1.1	0.1
0	4.9	3.6	1.4	0.1
0	5.3	3.7	1.5	0.2
1	4.9	2.4	3.3	1.0
1	5.7	2.8	4.1	1.3
1	6.3	3.3	4.7	1.6
1	6.7	3.0	5.0	1.7

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

上海科技大学 ShanghaiTech University

Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width
0	4.3	3.0
0	4.9	3.6
0	5.3	3.7
1	4.9	2.4
1	5.7	2.8
1	6.3	3.3
1	6.7	3.0

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

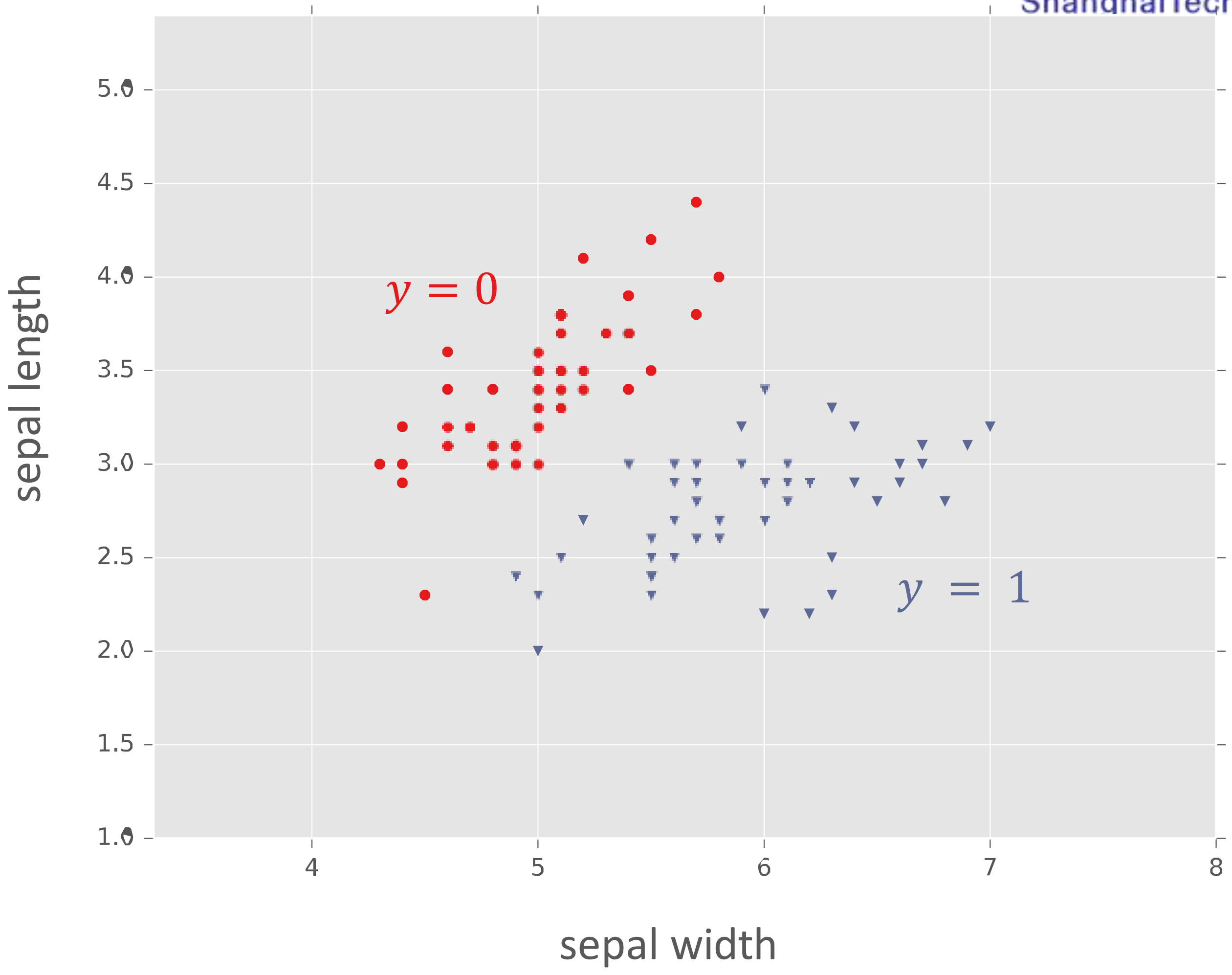


Figure courtesy of Matt Gormley

The Duck Test

Source: https://en.wikipedia.org/wiki/Duck_test

The Duck Test for Machine Learning

- Classify a point as the label of the "most similar" training point
- Idea: given real-valued features, we can use a distance metric to determine how similar two data points are
- A common choice is Euclidean distance:

$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2 = \sqrt{\sum_{d=1}^{D} (x_d - x'_d)^2}$$

- An alternative is the Manhattan distance:

$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_1 = \sum_{d=1}^{D} |x_d - x'_d|$$

K-NEAREST NEIGHBORS

Nearest Neighbor: Algorithm

def train($\mathcal{D}$):

Store $\mathcal{D}$

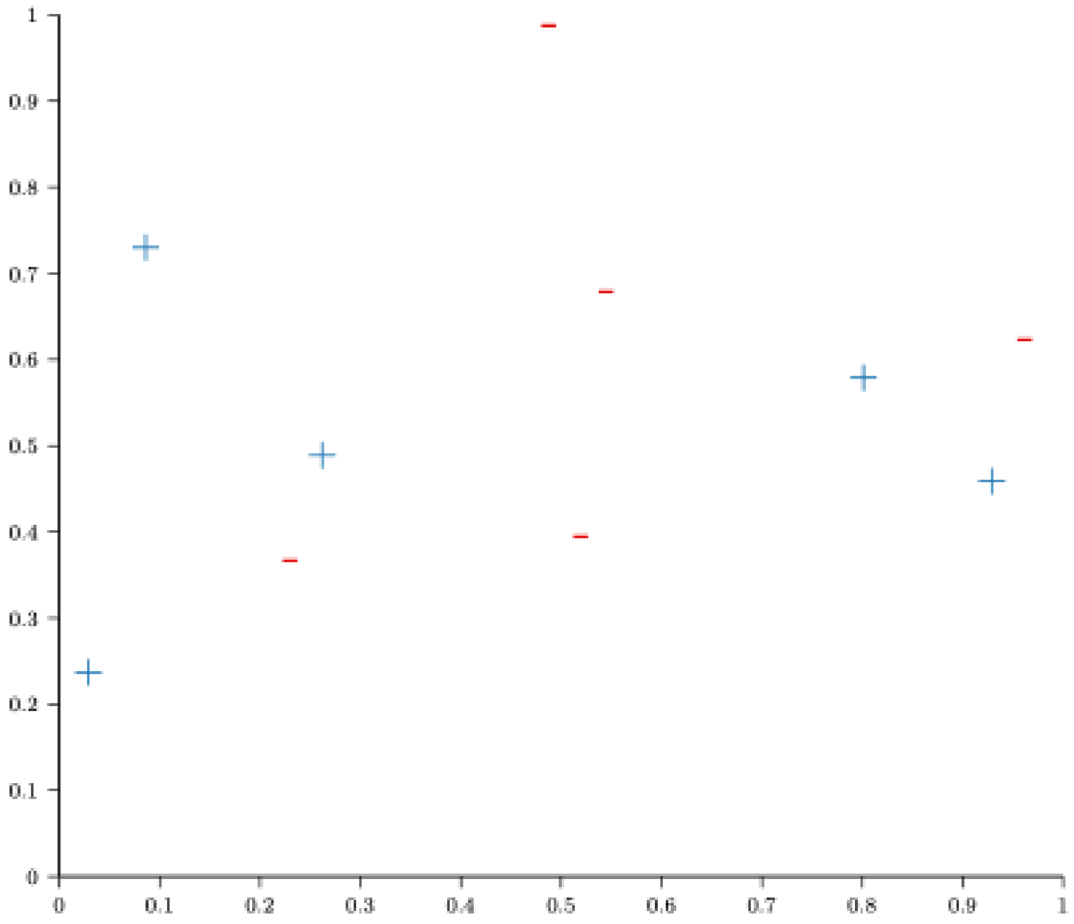
def h($\boldsymbol{x}'$):

Let $\boldsymbol{x}^{(i)}$ = the point in $\mathcal{D}$ that is nearest to $\boldsymbol{x}'$

return $y^{(i)}$

Nearest Neighbor: Example

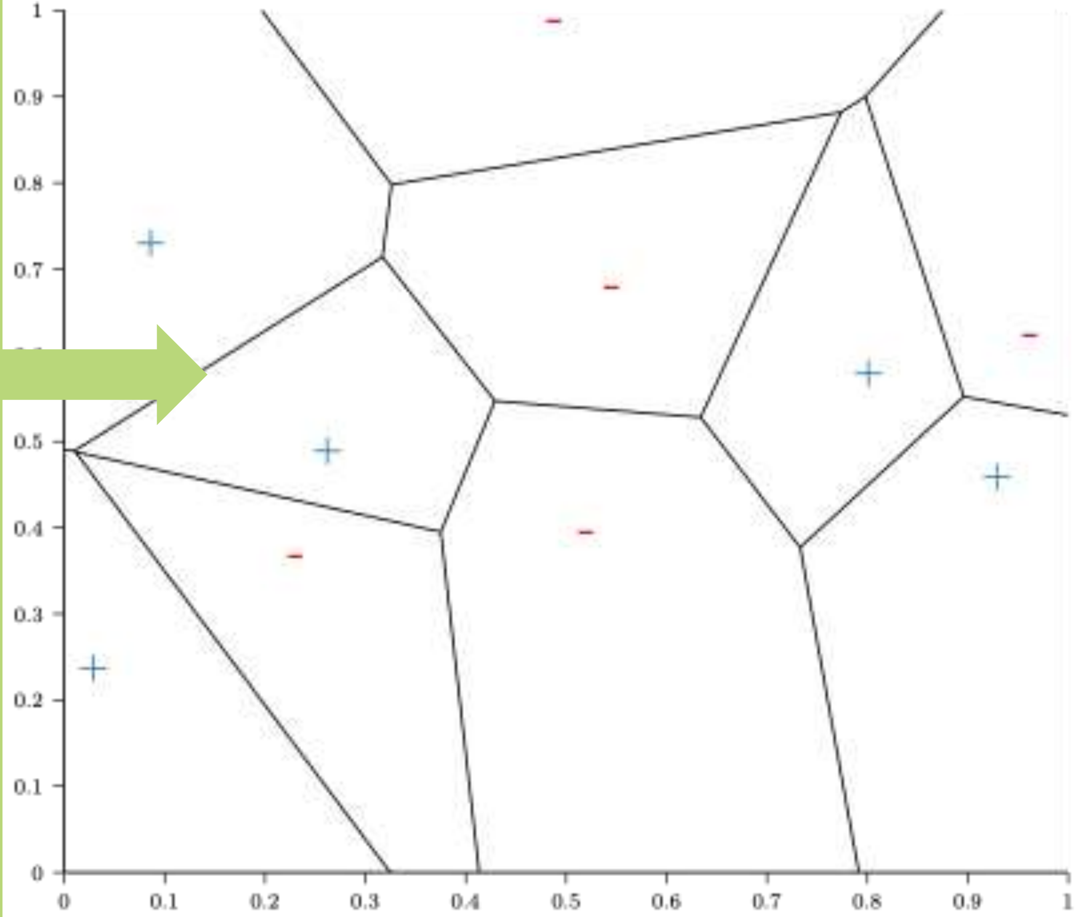
上海科技大学
ShanghaiTech University



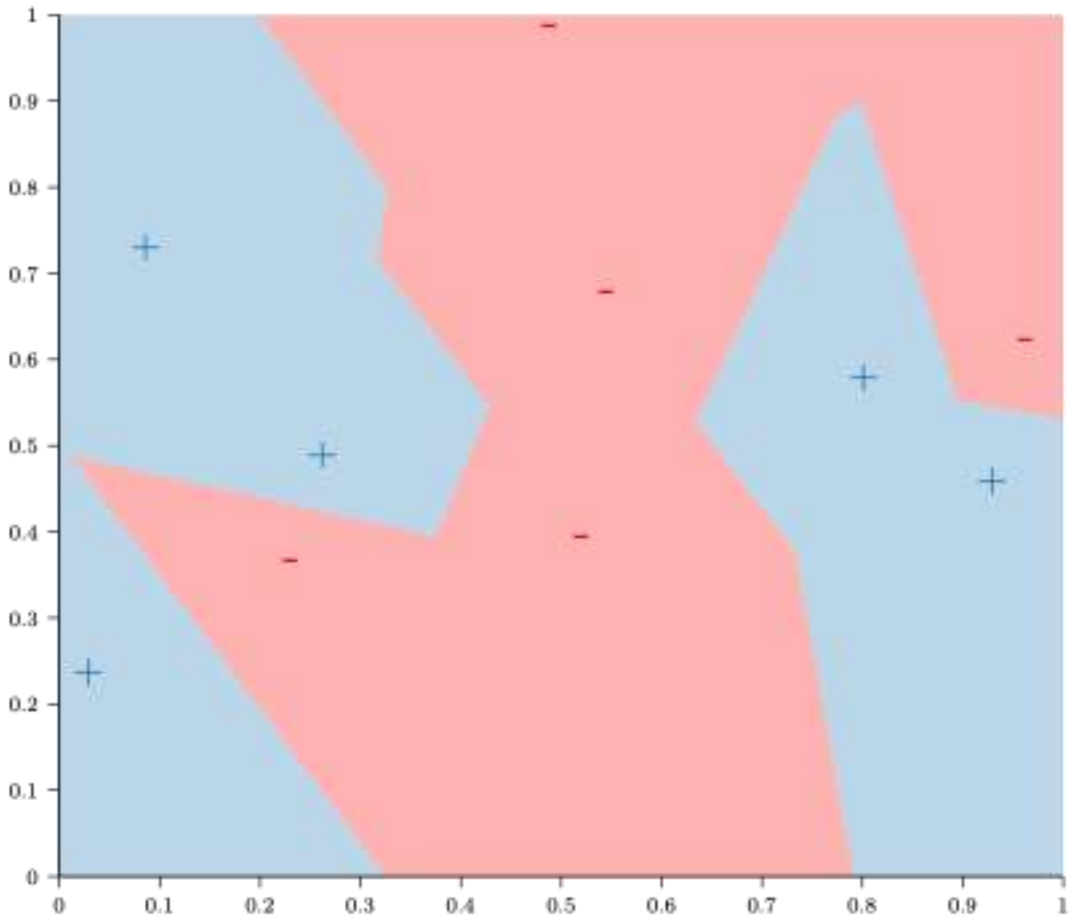
Nearest Neighbor: Example

上海科技大学 ShanghaiTech University

- This is a **Voronoi diagram**
- Each **cell** contain one of our training examples
- **All points within a cell** are **closer** to that training example, than to any other training example
- **Points on the Voronoi line segments** are **equidistant** to one or more training examples



Nearest Neighbor: Example



The Nearest Neighbor Model

- Requires no training!
- Always has zero training error!
  - ***A data point is always its own nearest neighbor***

k-Nearest Neighbors: Algorithm

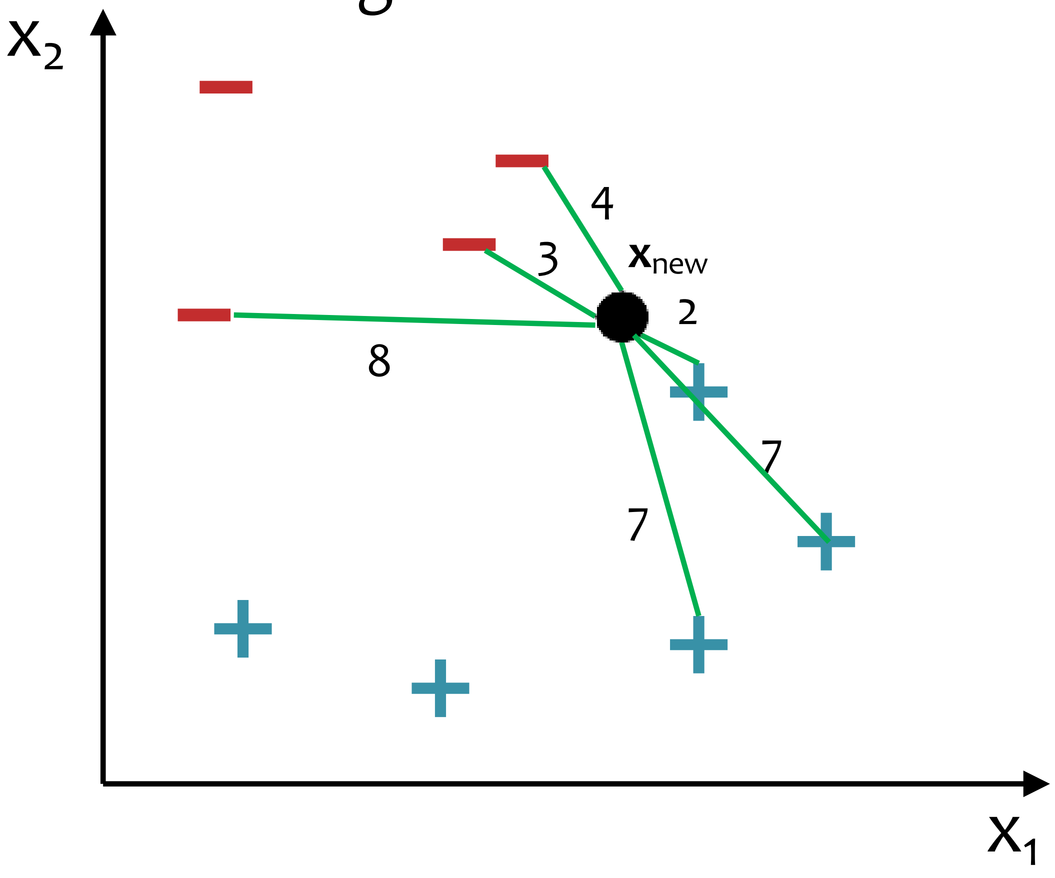
上海科技大学 ShanghaiTech University

```
def set_hyperparameters(k, d):
        Store k
        Store d(·, ·)

def train(𝒟):
        Store 𝒟

def h(x′):
        Let S = the set of k points in 𝒟 nearest to x′
                according to distance function
                d(u, v)
        Let v = majority_vote(S)
        return v
```

k-Nearest Neighbors

上海科技大学 ShanghaiTech University

Suppose we have the training dataset below.



How should we label the new point?

It depends on k:

if k=1, $h(\mathbf{x}_{new}) = +1$

if k=3, $h(\mathbf{x}_{new}) = -1$

if k=5, $h(\mathbf{x}_{new}) = +1$

+

−

KNN: Remarks

Distance Functions:

- KNN requires a **distance function**

$$d : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

- The most common choice is **Euclidean distance**

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{\sum_{m=1}^{M} (u_m - v_m)^2}$$

- But there are other choices (e.g. **Manhattan distance**)

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{m=1}^{M} |u_m - v_m|$$

KNN: Computational Efficiency

上海科技大学 ShanghaiTech University

- Suppose we have N training examples and each one has M features
- Computational complexity when k=1:

Task	Naive	k-d Tree
Train	O(1)	~ O(M N log N)
Predict (one test example)	O(MN)	~ O(2^M log N) on average

Problem: Very fast for small M, but very slow for large M

In practice: use stochastic approximations (very fast, and empirically often as good)

KNN: Theoretical Guarantees

上海科技大学 ShanghaiTech University

Cover & Hart (1967)

Let h(x) be a Nearest Neighbor (k=1) binary classifier. As the number of training examples N goes to infinity…

$$\text{error}_{\text{true}}(h) < 2 \times \text{Bayes Error Rate}$$

"In this sense, it may be said that half the classification information in an infinite sample set is contained in the nearest neighbor."

very informally, Bayes Error Rate can be thought of as:

'the best you could possibly do'

KNN: Remarks

In-Class Exercises

How can we handle ties for even values of k?

Answer(s) Here:

KNN: Remarks

In-Class Exercises

How can we handle ties for even values of k?

Answer(s) Here:

- Consider another point
- Remove farthest of k points
- Weight votes by distance
- Consider another distance metric

Recall…

Decision Trees: Inductive Bias

- The **inductive bias** of a machine learning algorithm is the principle by which it generalizes to unseen examples
- What is the inductive bias of the ID3 algorithm?
  - Try to find the **_smallest decision_** tree that achieves a **_low/zero training error_** with **_high mutual information_** features at the top
- Occam’s razor: try to find the “simplest” (e.g., smallest decision tree) classifier that explains the training dataset

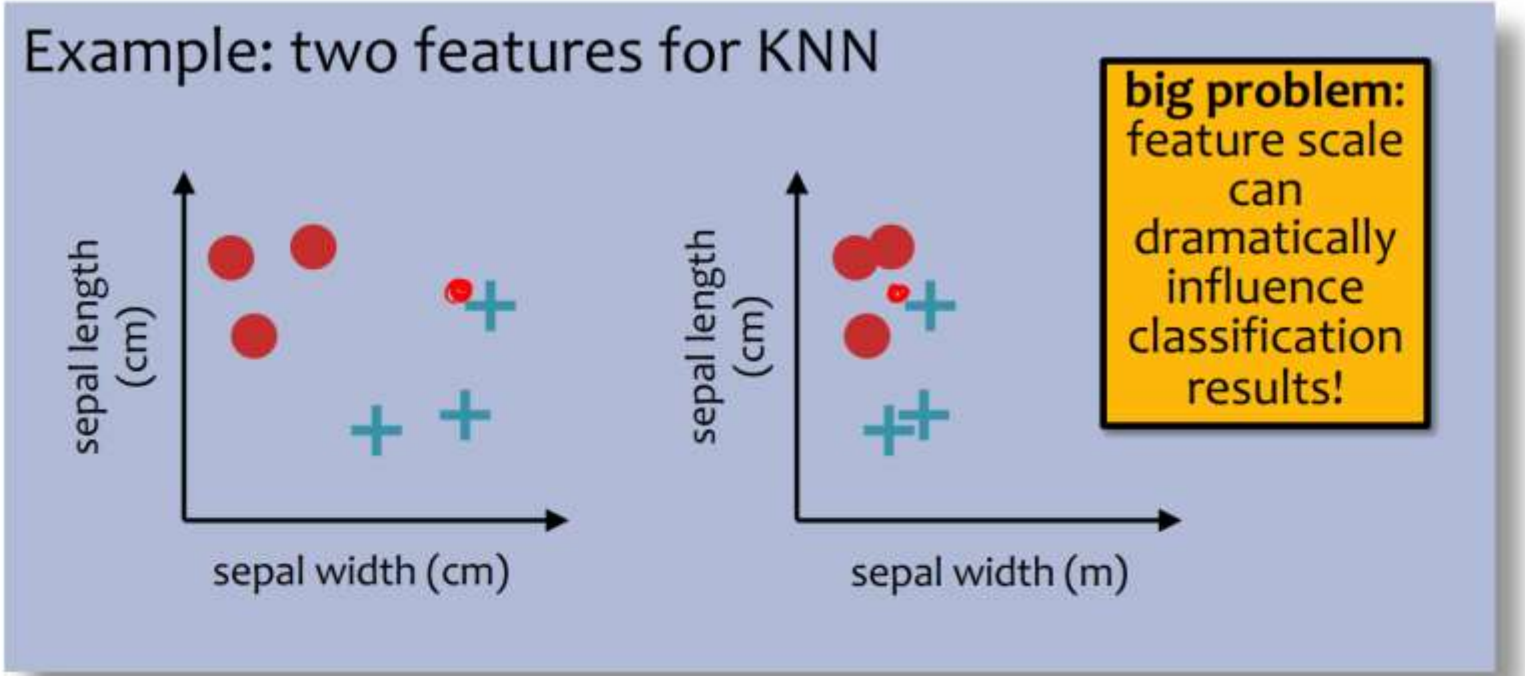
KNN: Inductive Bias

In-Class Exercise What is the inductive bias of KNN?

KNN: Inductive Bias

In-Class Exercise What is the inductive bias of KNN?

1. Similar points should have similar labels
2. All dimensions are created equally!



Classification & Real-Valued Features

Def: Classification

$$D = \left\{ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{N}$$

$\forall i,\ \mathbf{x}^{(i)} \in \mathbb{R}^M$ (features/instance)

$\forall i,\ y^{(i)} \in \{1, 2, \ldots, L\}$ (label/class)

$M =$ # of features (dimensionality of $\mathbf{x}$)

$N =$ # of training examples $= |D|$

Def: Binary Classification

Classification where $|\mathcal{Y}| = 2$

$\forall i,\ y^{(i)} \in \{+, -\}$

$\in$ {red, blue}

$\in$ {cat, dog}

Classification & Real-Valued Features

Def: Hypothesis (aka. Decision Rule) for Binary Classification

$h : \mathbb{R}^M \rightarrow \{+, -\}$

Train time:	Learn h
Test time:	Given $\hat{\mathbf{x}}$, predict $\hat{y} = h(\hat{\mathbf{x}})$
	Evaluate h

Ex: Decision Boundaries (2D Binary Classification)

Decision Boundary Example

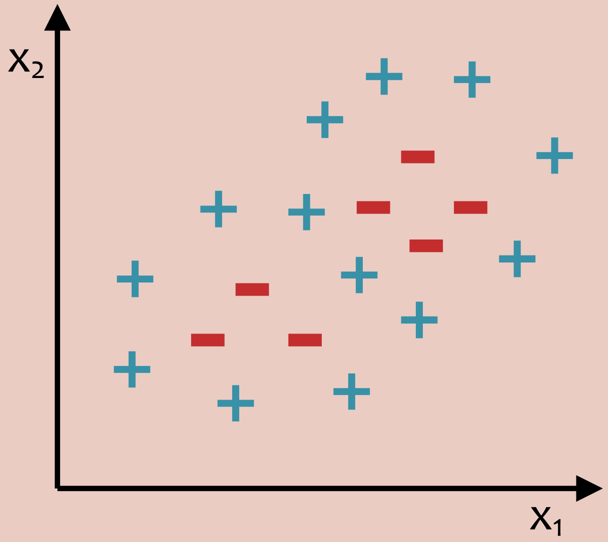
Dataset: Outputs {+,-}; Features x_1 and x_2

In-Class Exercise

Question:

A. Can a **k-Nearest Neighbor classifier with k=1** achieve **zero training error** on this dataset?

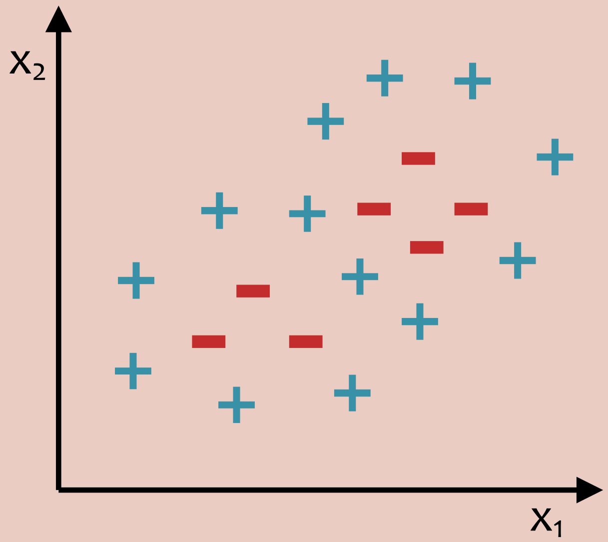
B. If 'Yes', draw the learned decision boundary. **If 'No'**, why not?



Question:

A. Can a **Decision Tree classifier** achieve **zero training error** on this dataset?

B. If 'Yes', draw the learned decision boundary. **If 'No'**, why not?



KNN ON FISHER IRIS DATA


sepal
petal


Fisher Iris Dataset

上海科技大学 ShanghaiTech University

- Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
0	4.3	3.0	1.1	0.1
0	4.9	3.6	1.4	0.1
0	5.3	3.7	1.5	0.2
1	4.9	2.4	3.3	1.0
1	5.7	2.8	4.1	1.3
1	6.3	3.3	4.7	1.6
1	6.7	3.0	5.0	1.7

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

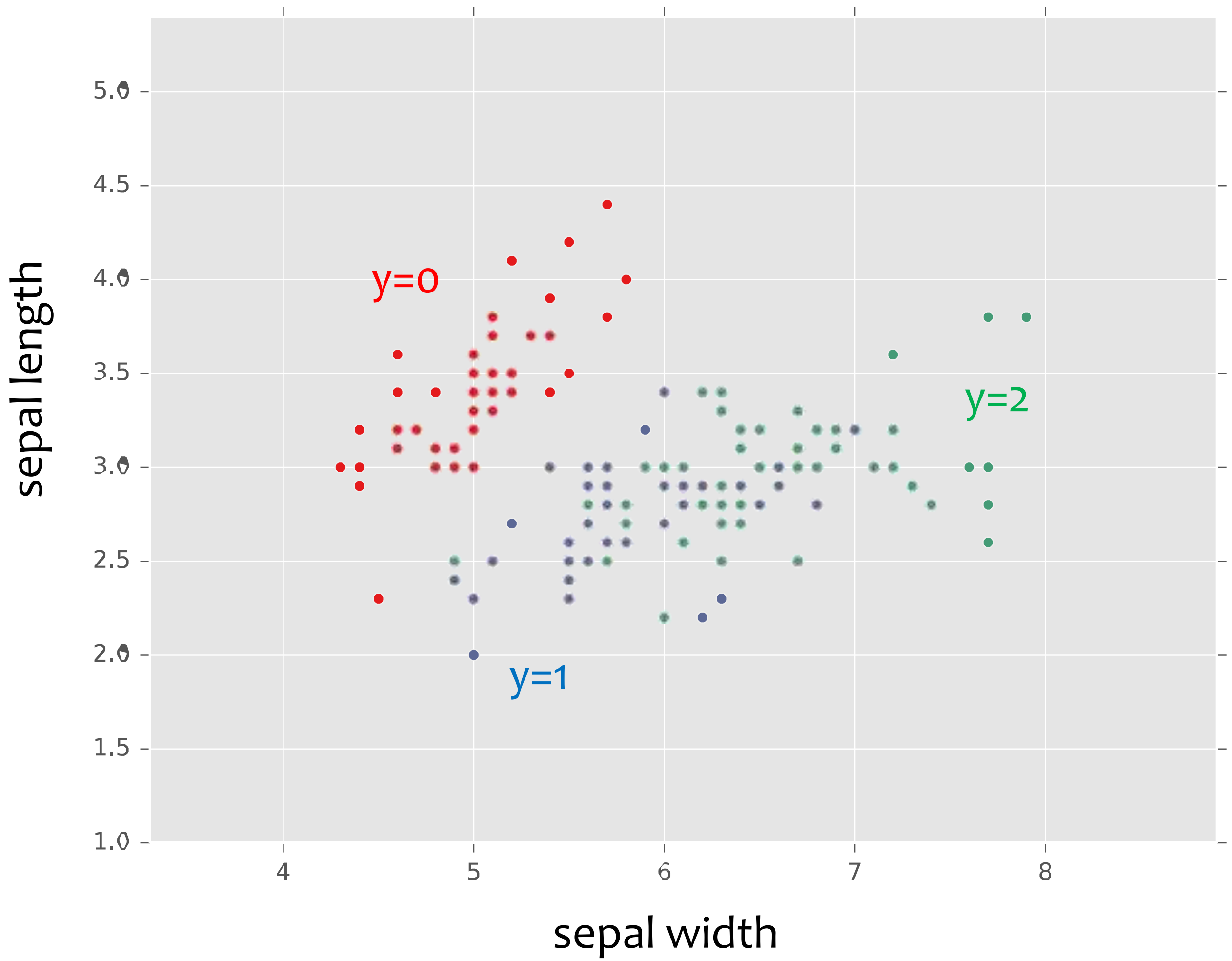
- Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width
0	4.3	3.0
0	4.9	3.6
0	5.3	3.7
1	4.9	2.4
1	5.7	2.8
1	6.3	3.3
1	6.7	3.0

Deleted two of the four features, so that input space is 2D

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

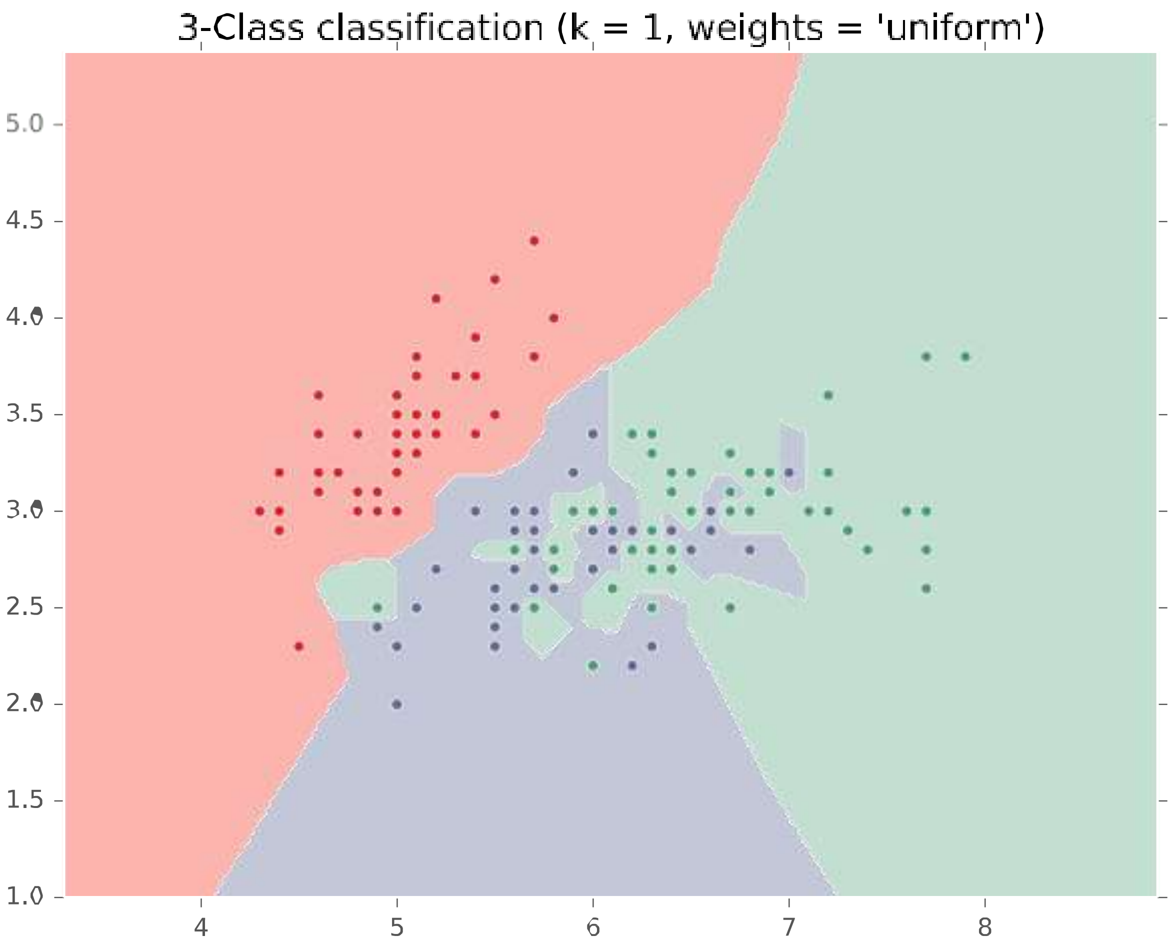
KNN on Fisher Iris Data



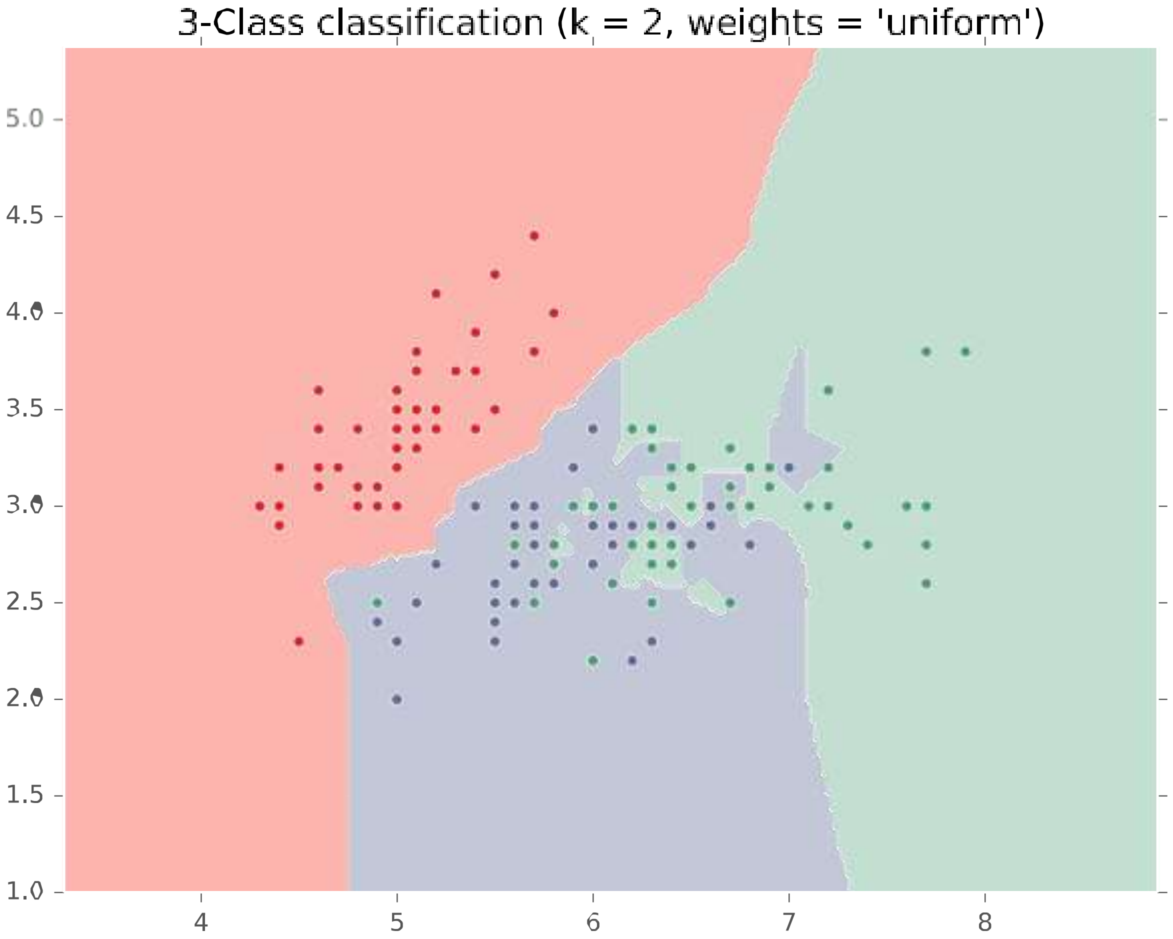
KNN on Fisher Iris Data

上海科技大学 ShanghaiTech University

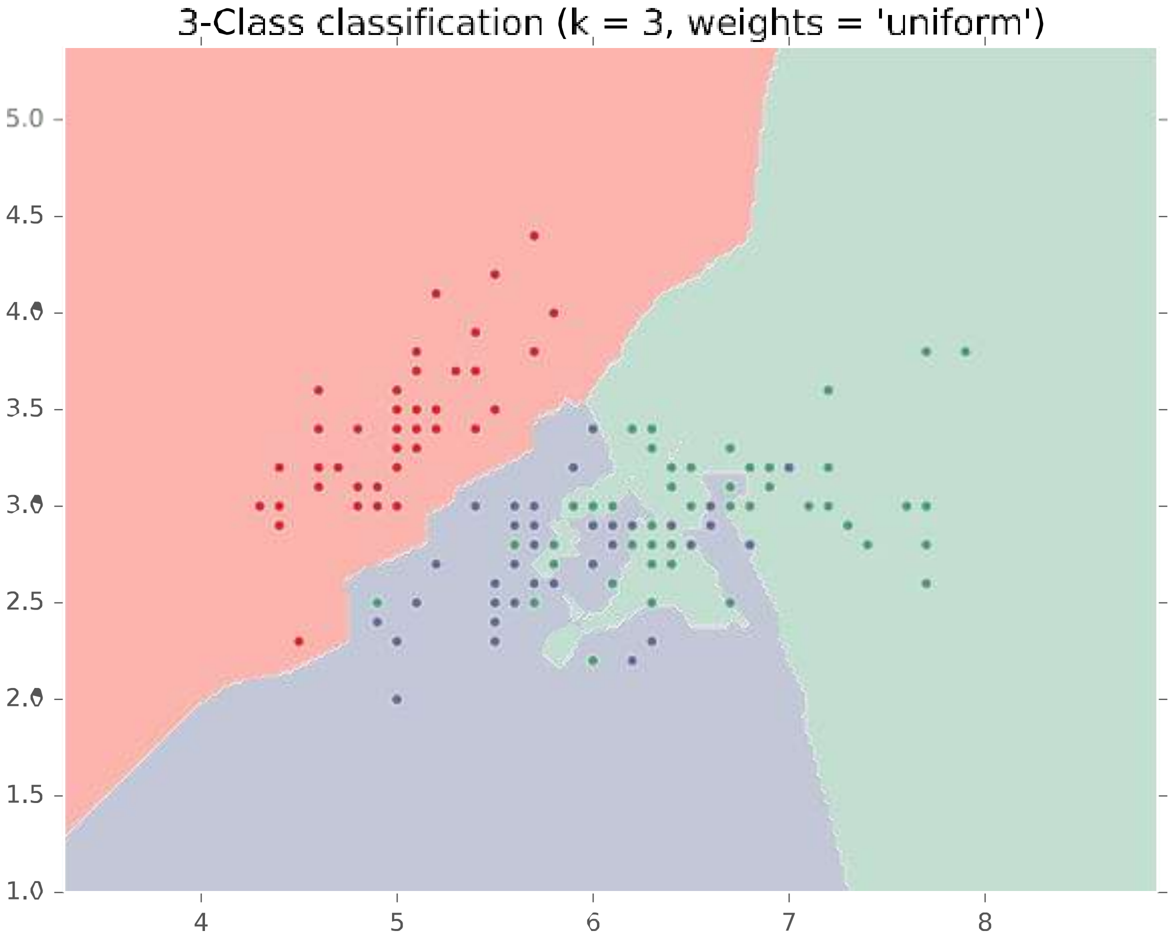
Special Case: Nearest Neighbor



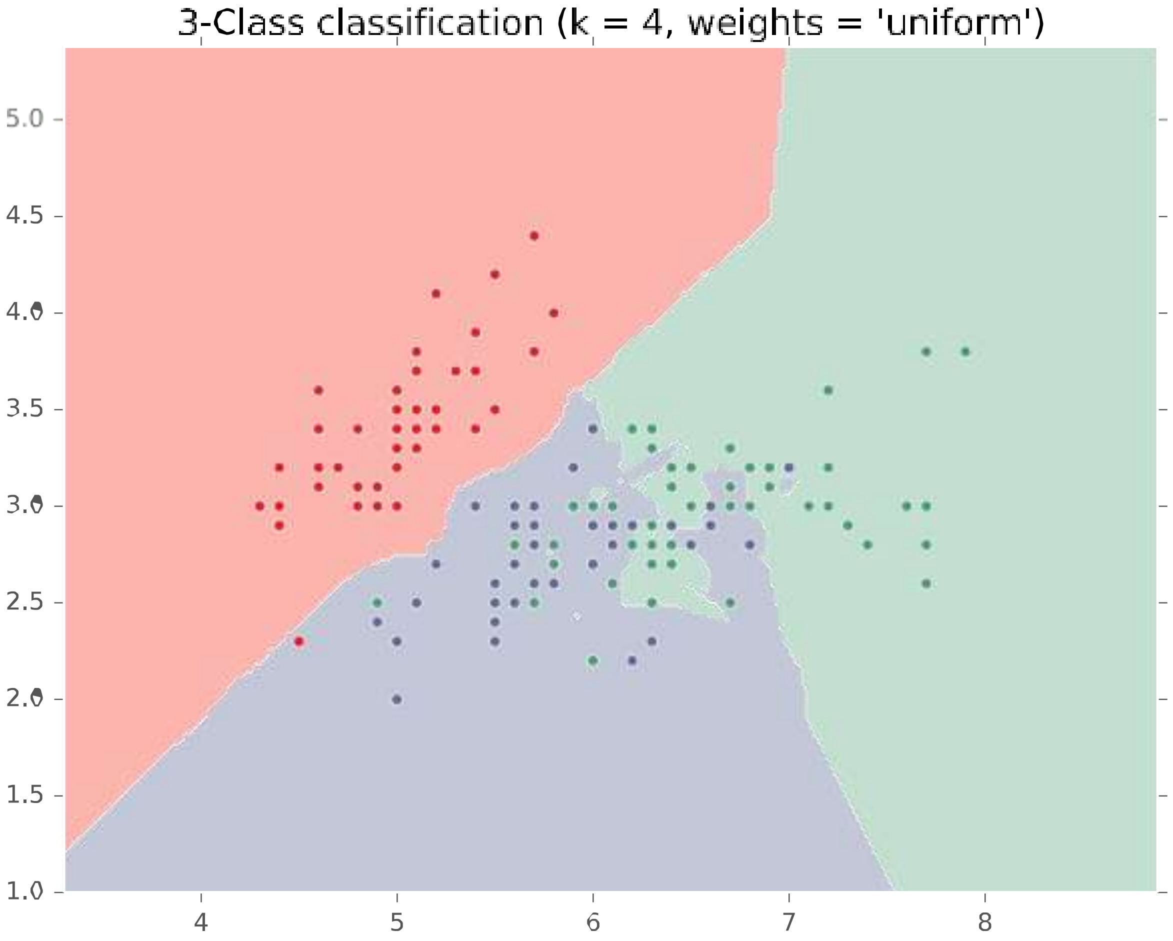
KNN on Fisher Iris Data



KNN on Fisher Iris Data

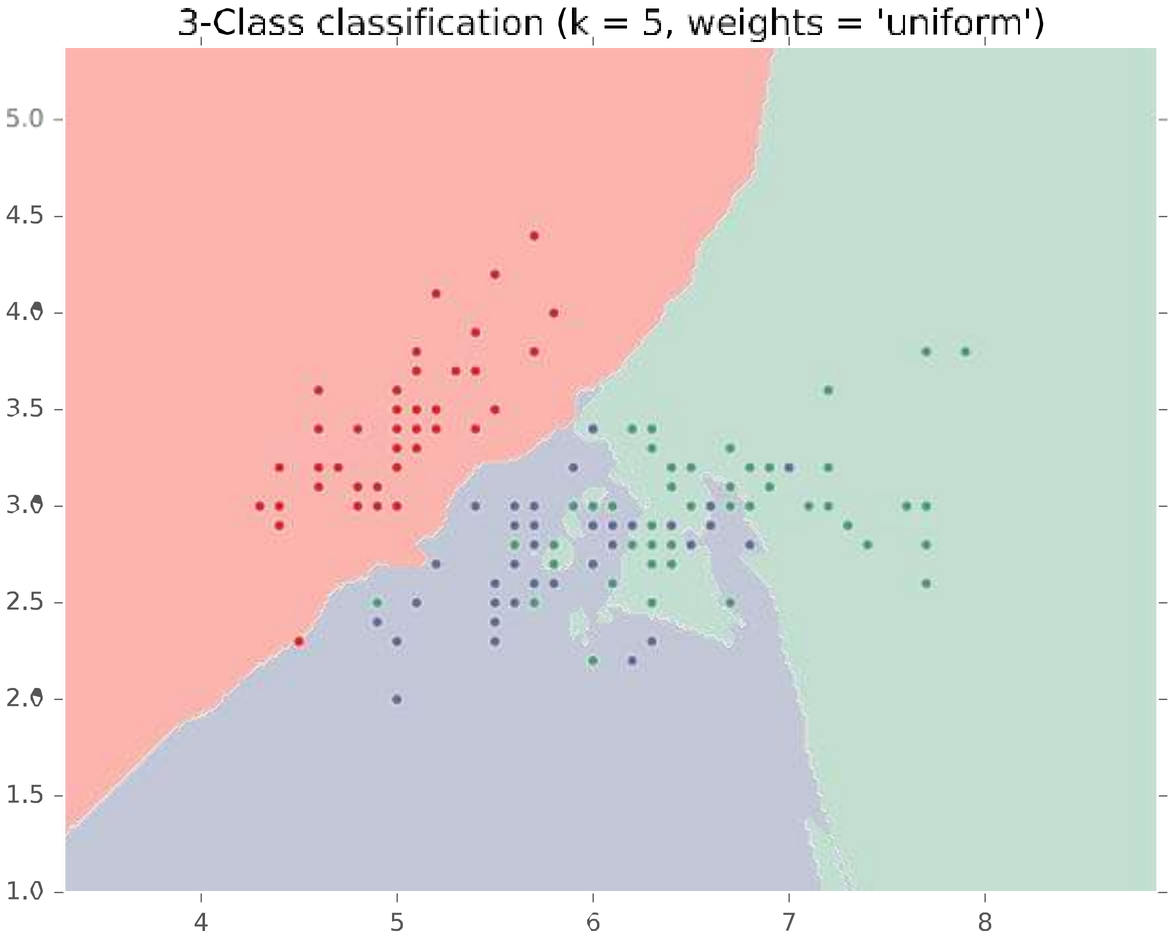


KNN on Fisher Iris Data



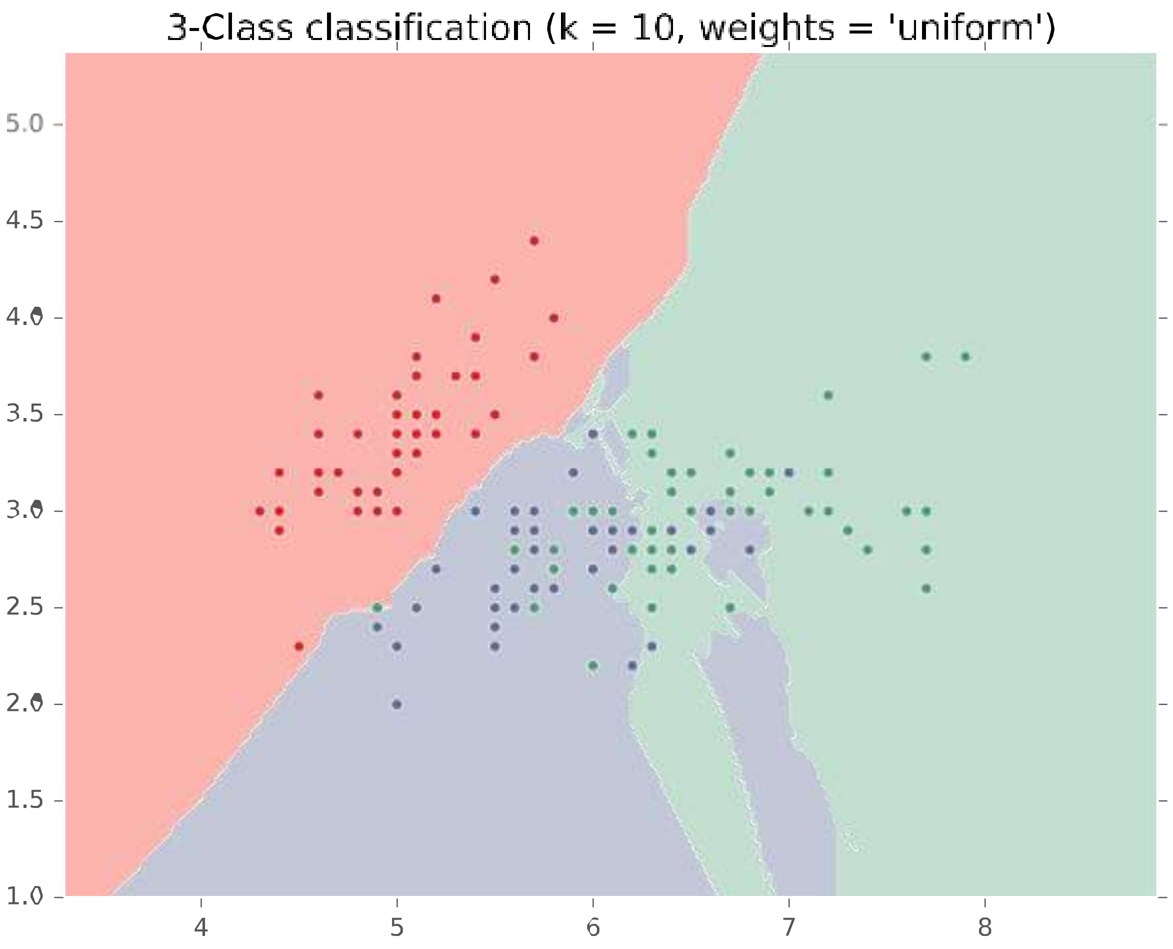
KNN on Fisher Iris Data

上海科技大学 ShanghaiTech University



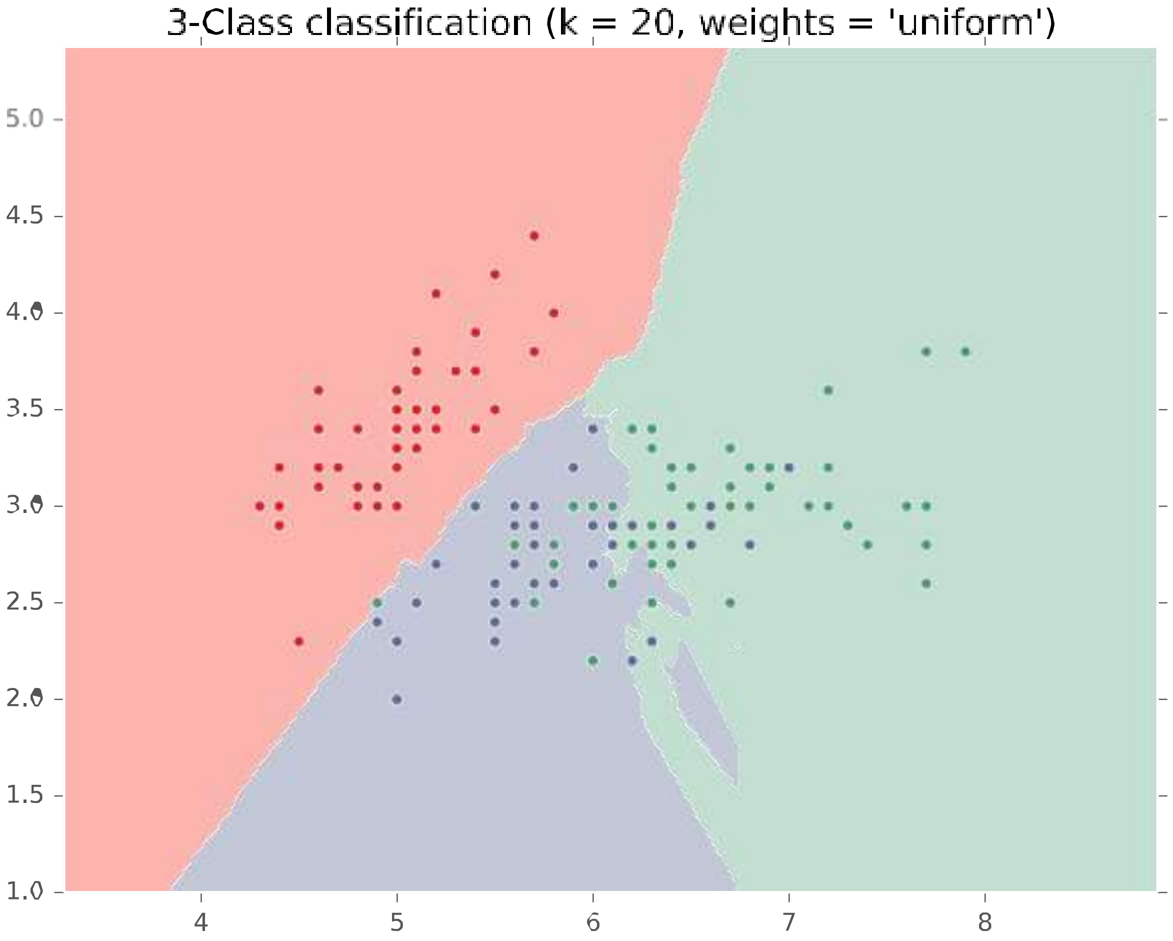
KNN on Fisher Iris Data

上海科技大学 ShanghaiTech University



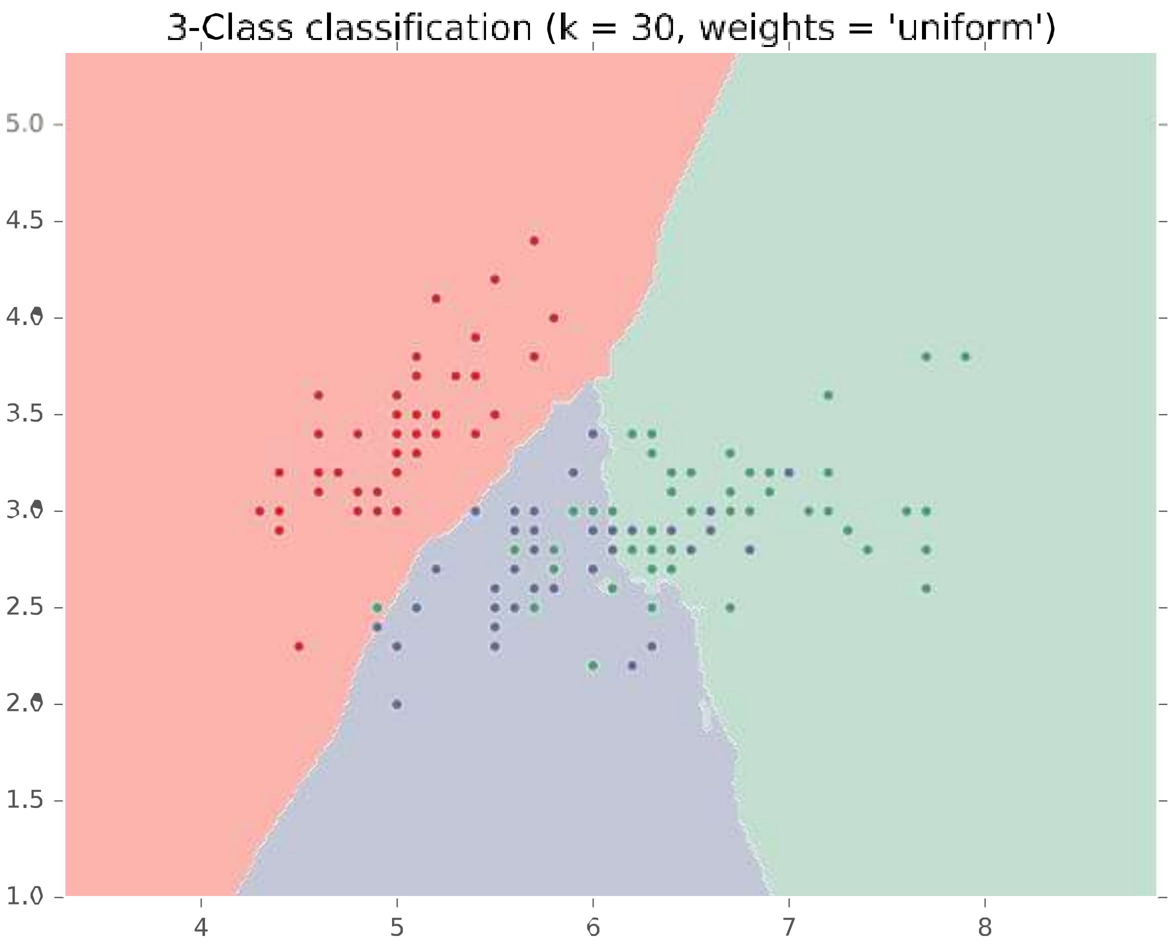
KNN on Fisher Iris Data

上海科技大学
ShanghaiTech University

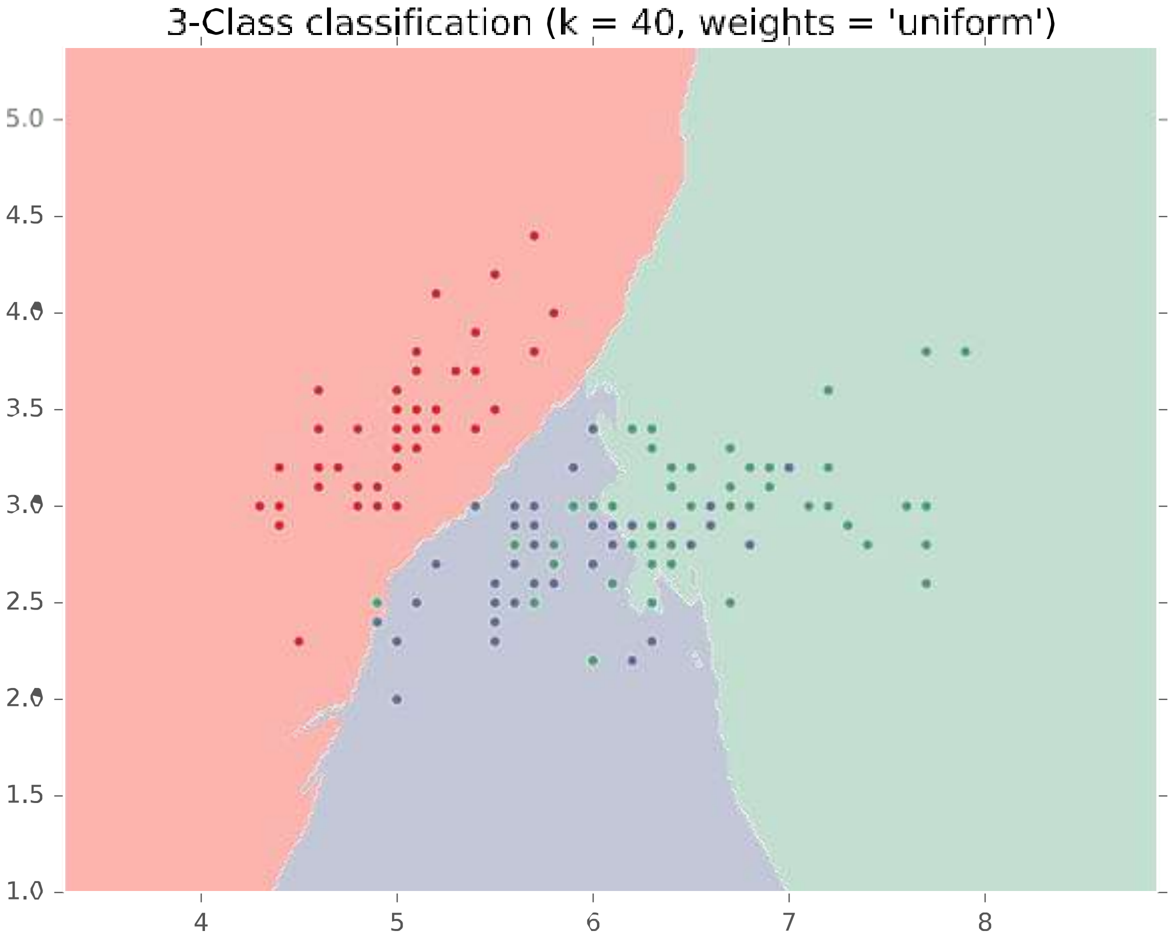


KNN on Fisher Iris Data

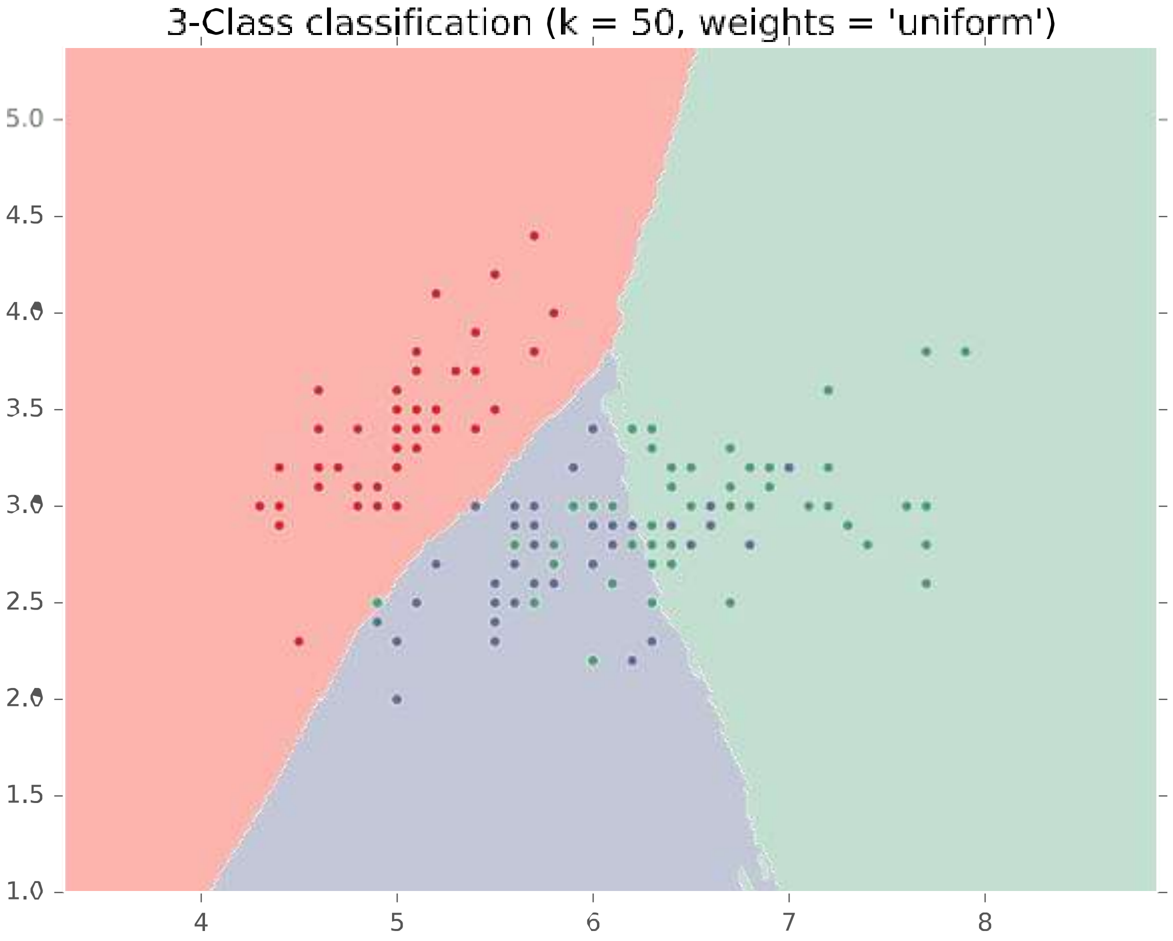
上海科技大学 ShanghaiTech University



KNN on Fisher Iris Data

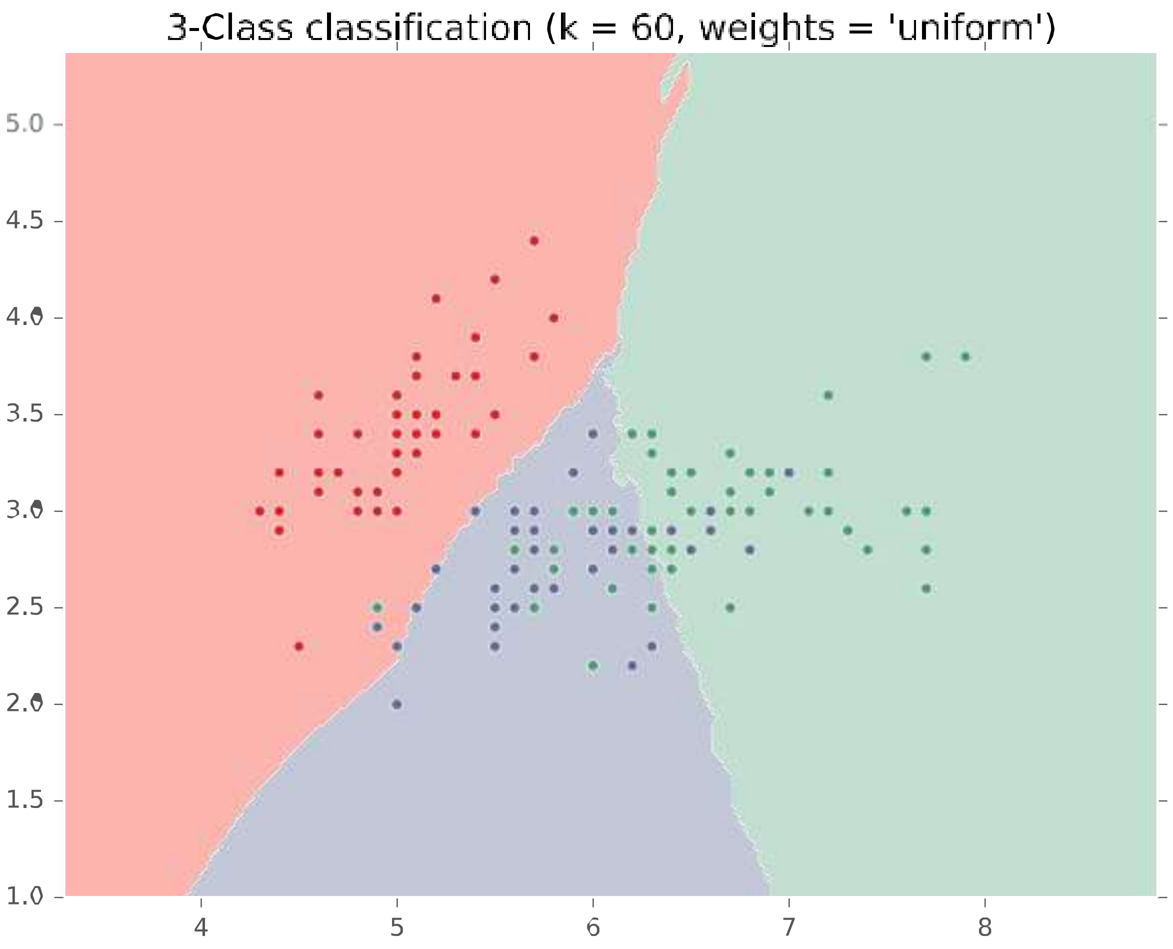


KNN on Fisher Iris Data

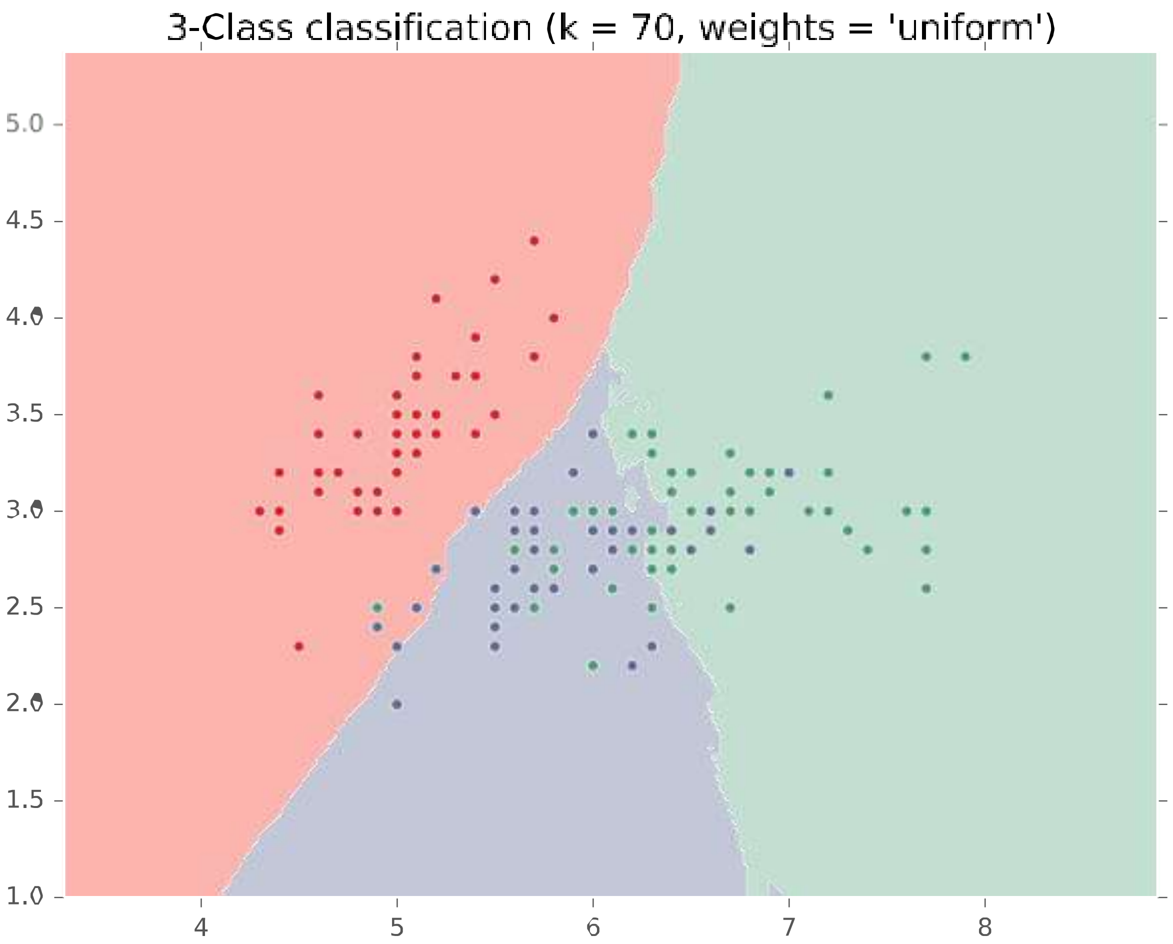


KNN on Fisher Iris Data

上海科技大学 ShanghaiTech University

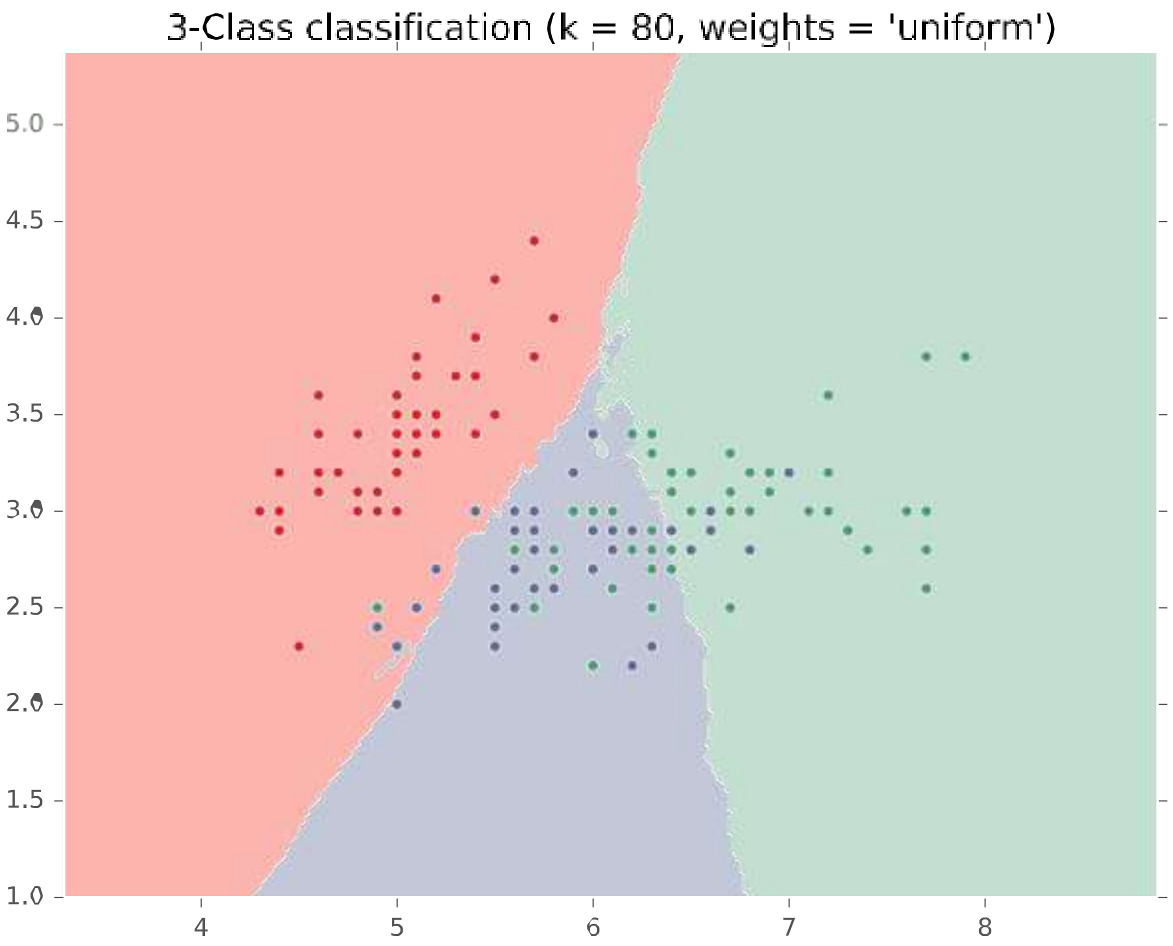


KNN on Fisher Iris Data

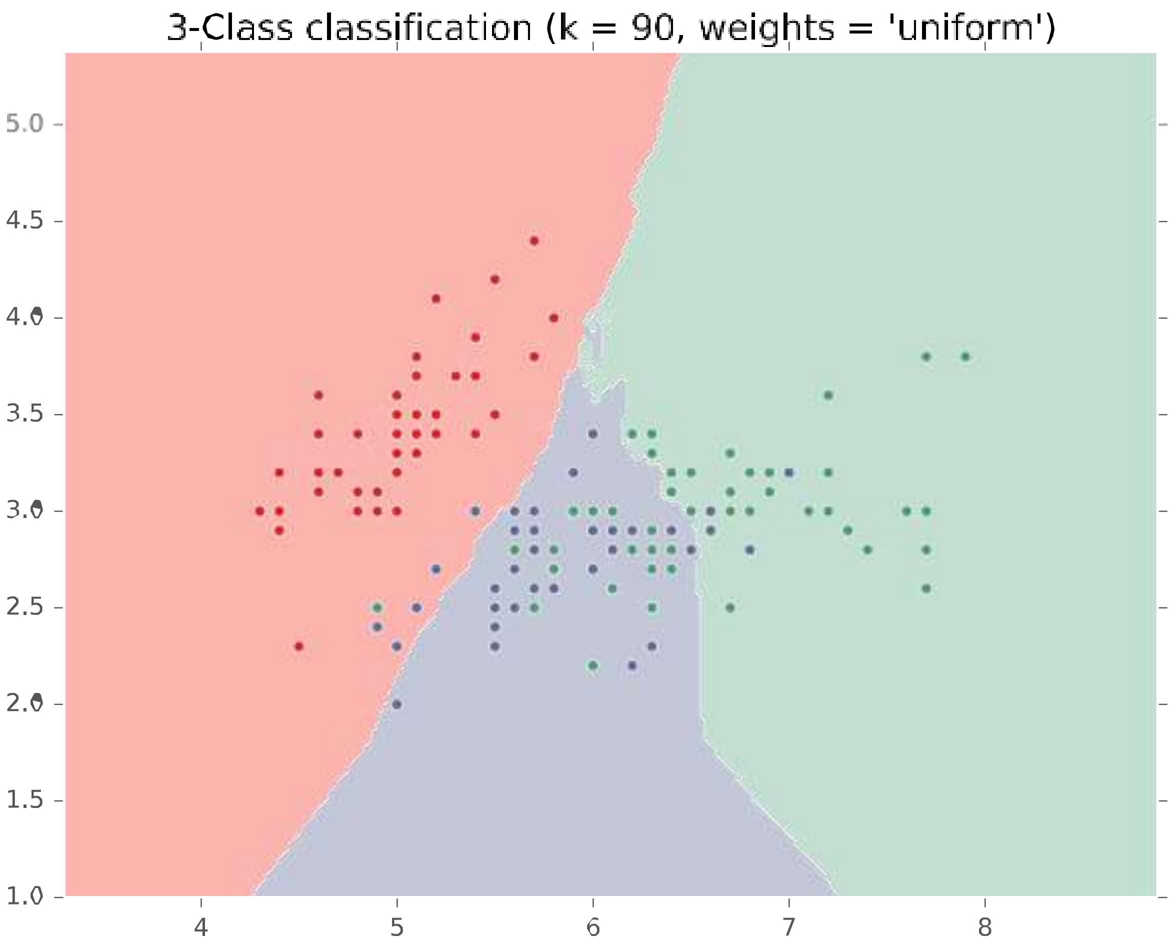


KNN on Fisher Iris Data

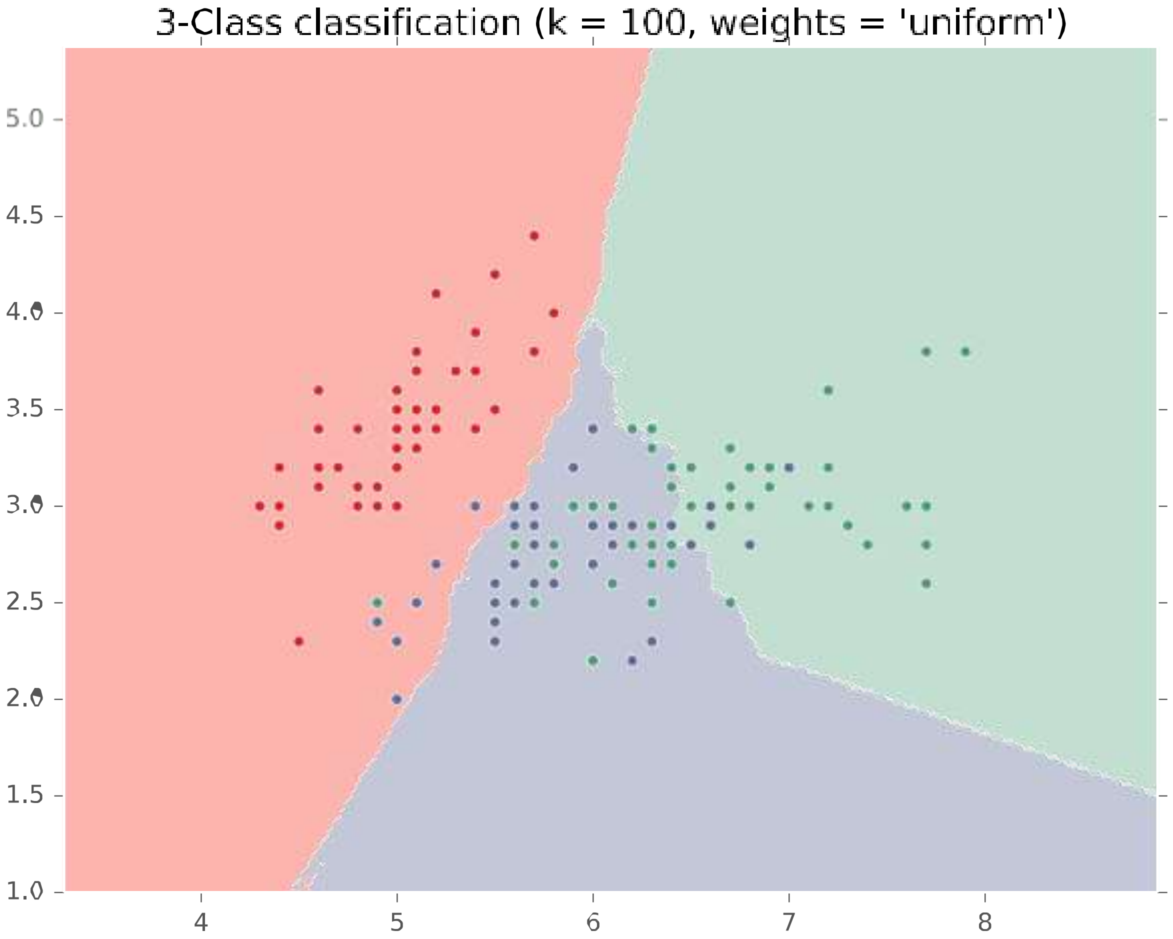
上海科技大学 ShanghaiTech University



KNN on Fisher Iris Data

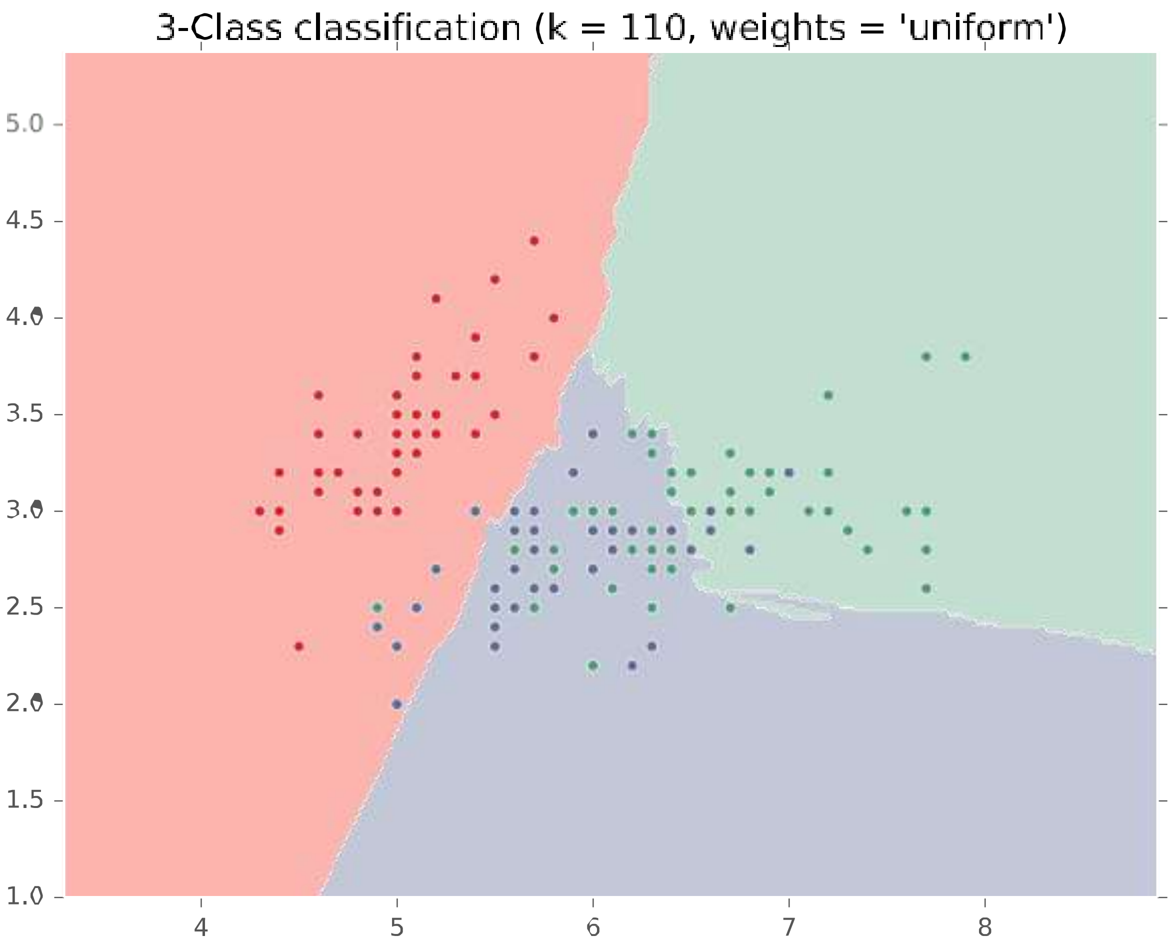


KNN on Fisher Iris Data



KNN on Fisher Iris Data

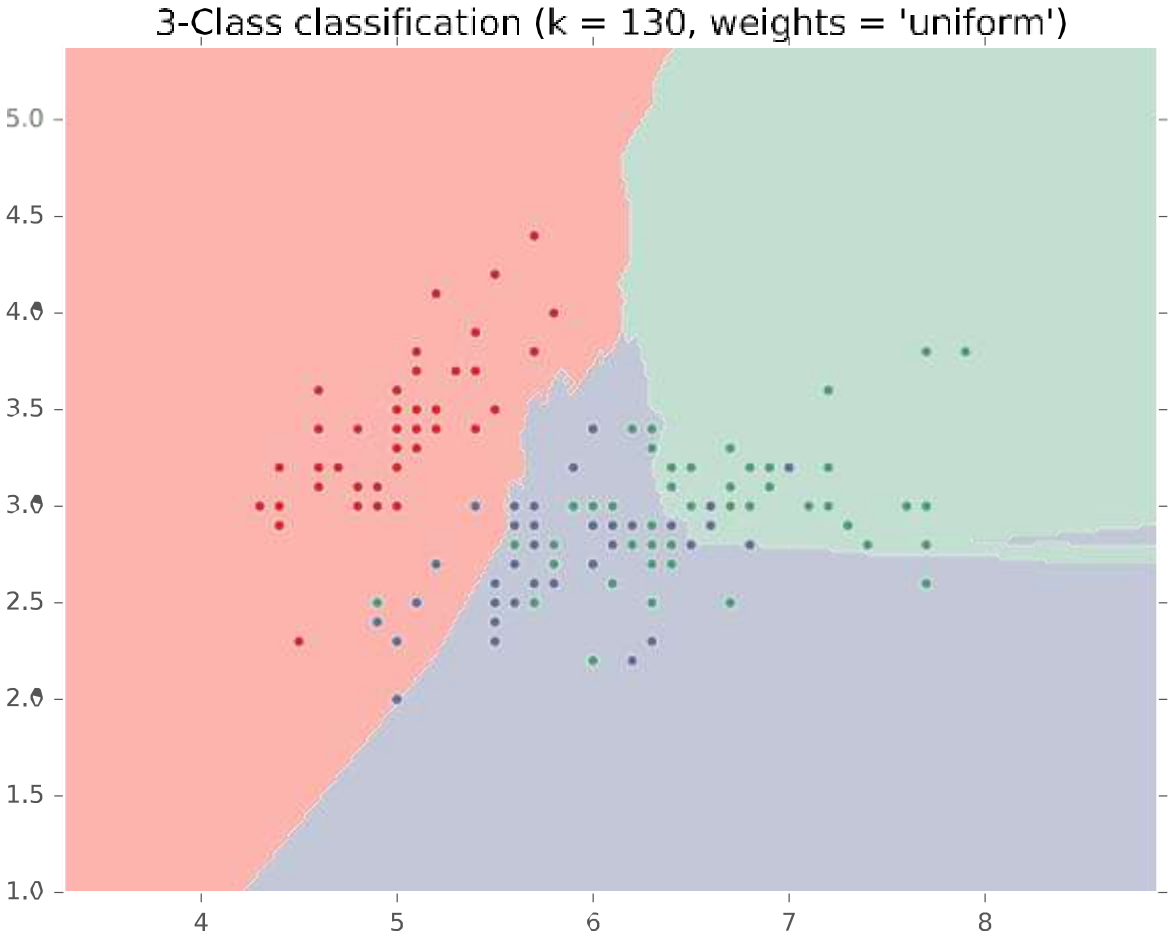
上海科技大学 ShanghaiTech University



KNN on Fisher Iris Data

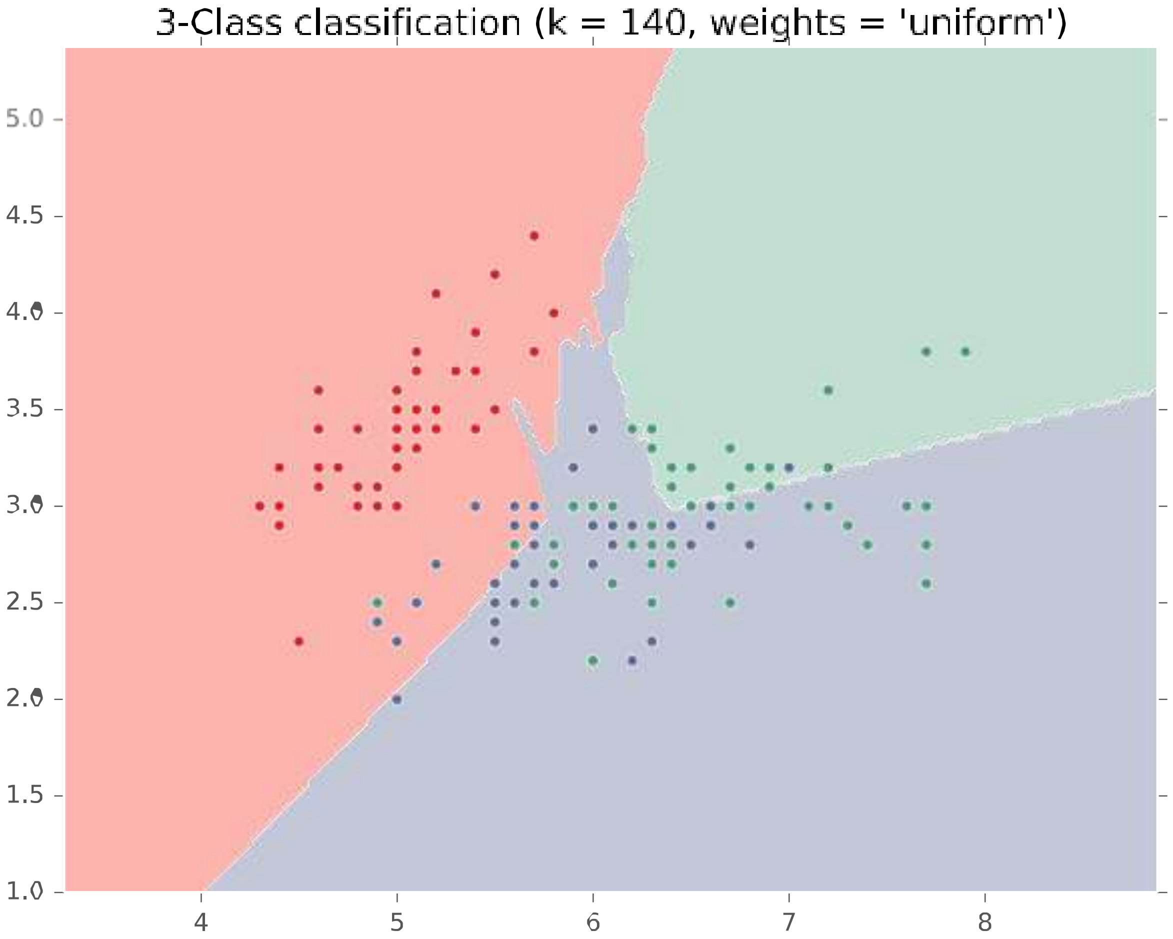
上海科技大学
ShanghaiTech University

KNN on Fisher Iris Data



KNN on Fisher Iris Data

上海科技大学
ShanghaiTech University



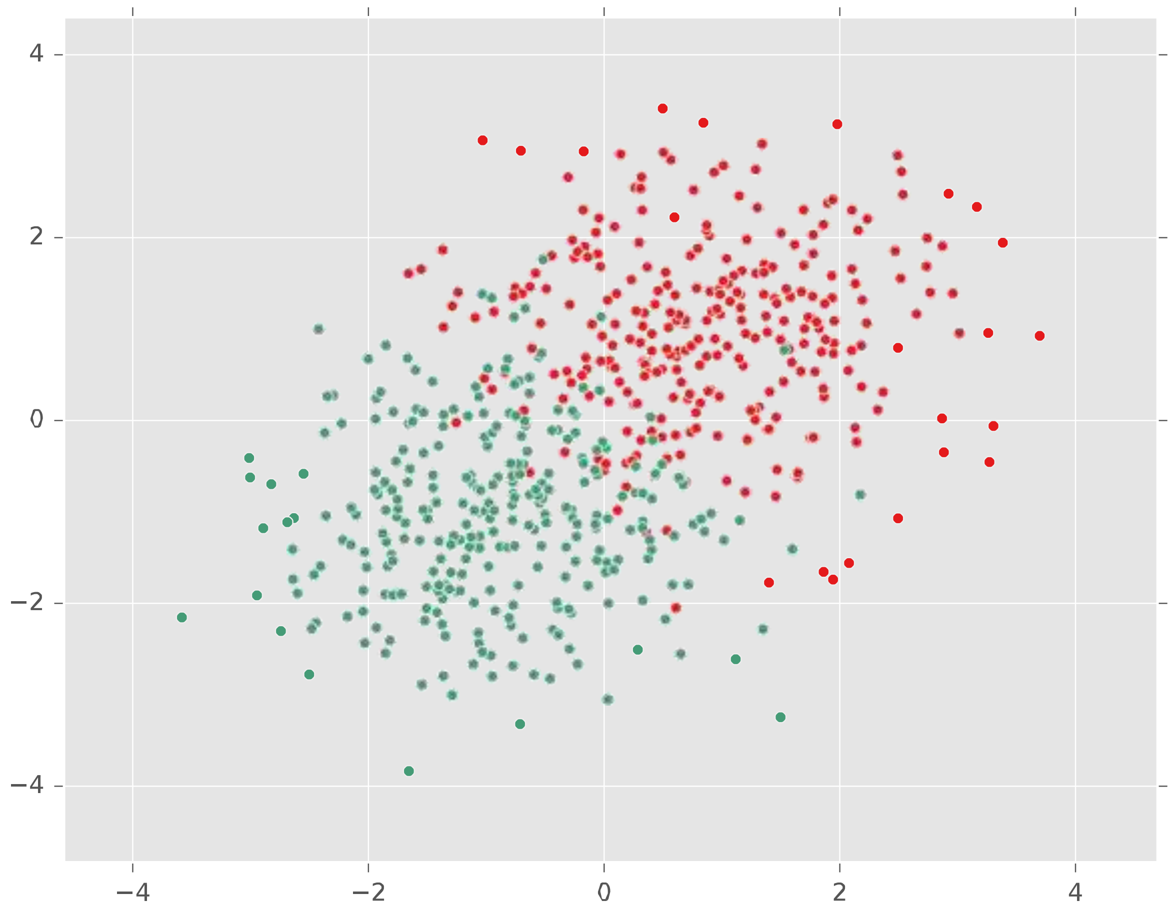
KNN on Fisher Iris Data

Special Case: Majority Vote

KNN ON GAUSSIAN DATA

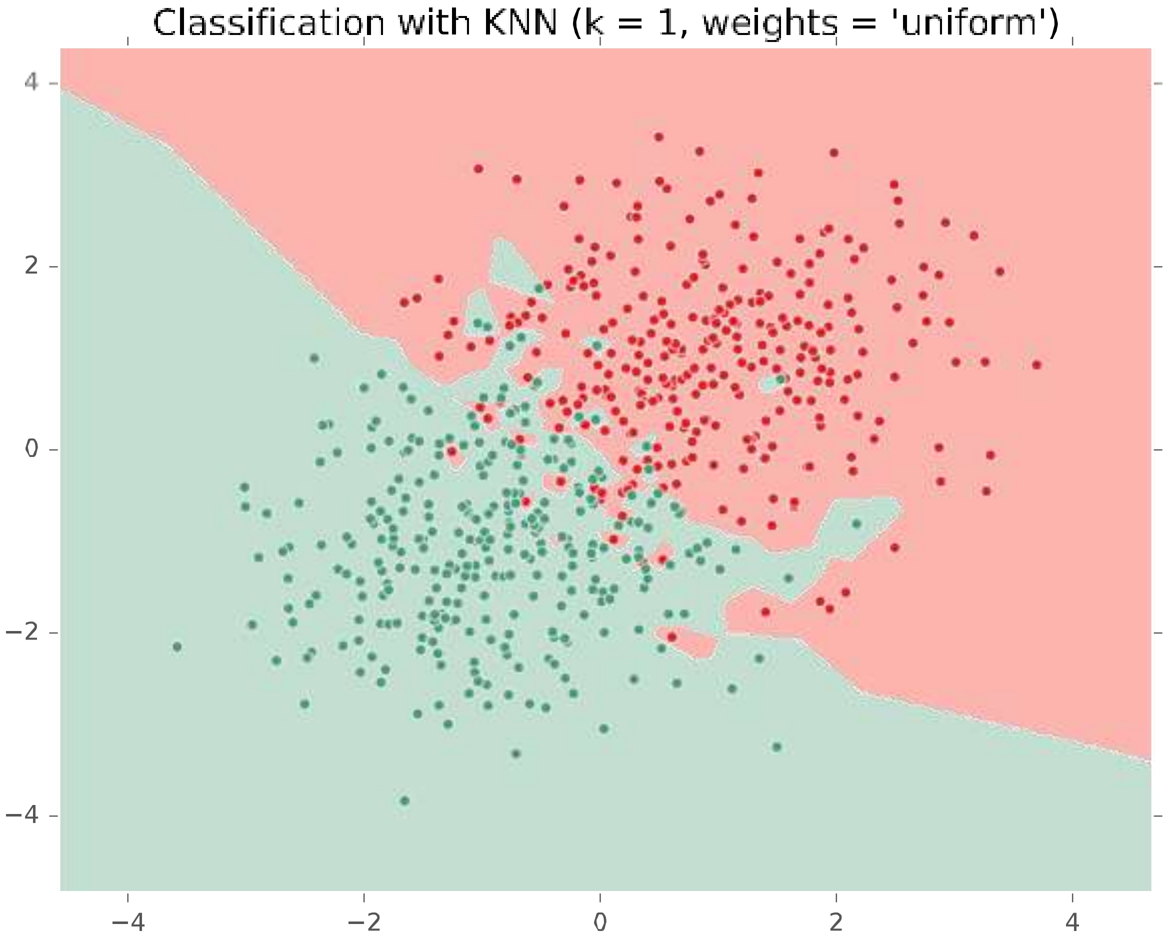
KNN on Gaussian Data

上海科技大学
ShanghaiTech University

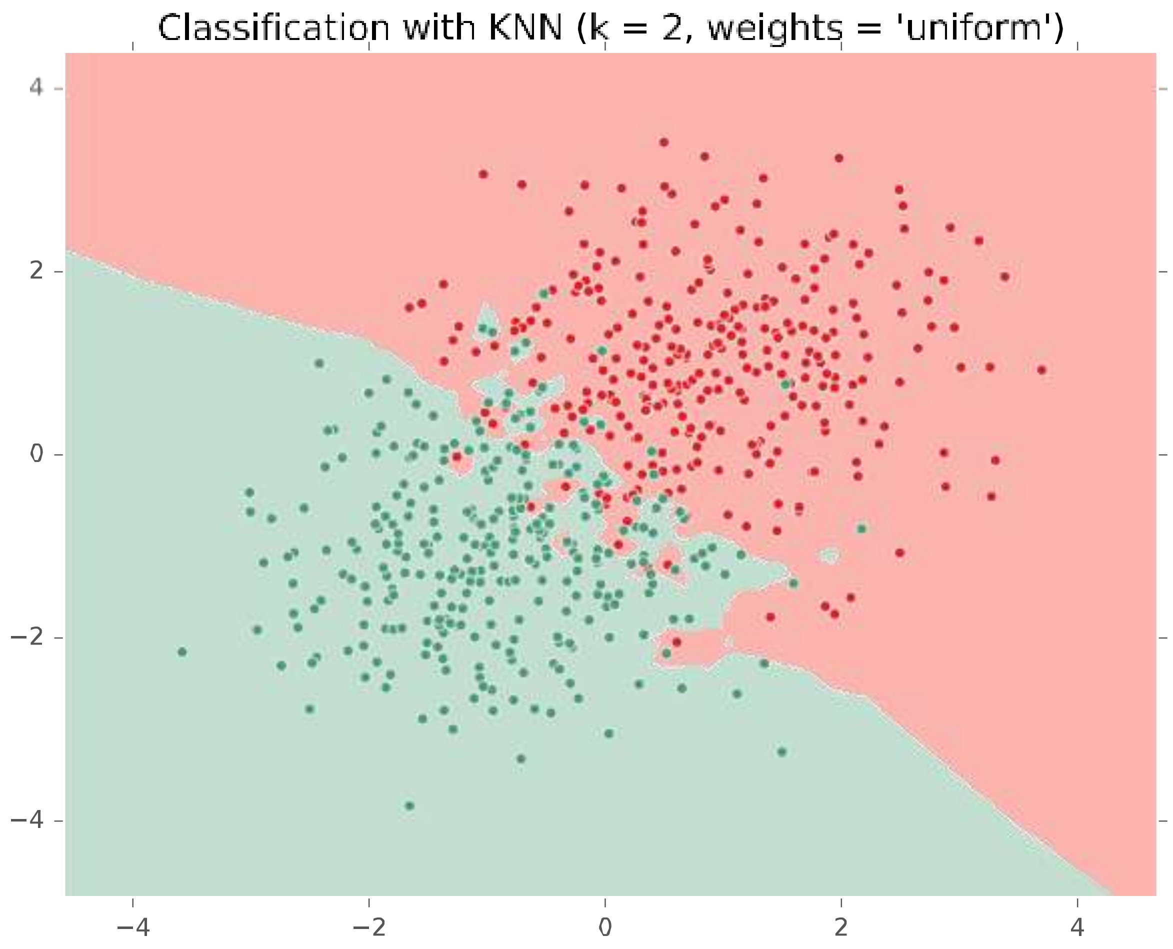


KNN on Gaussian Data

上海科技大学 ShanghaiTech University

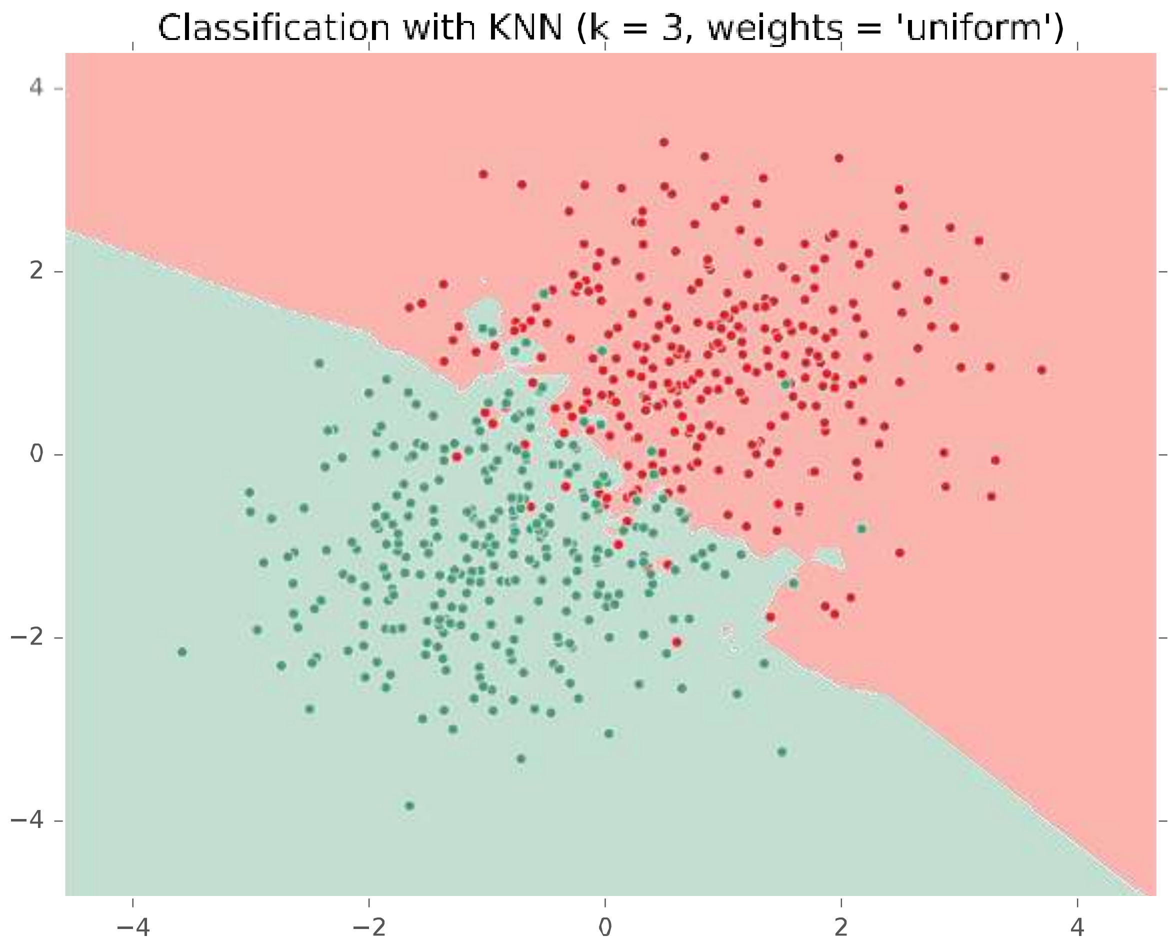


KNN on Gaussian Data



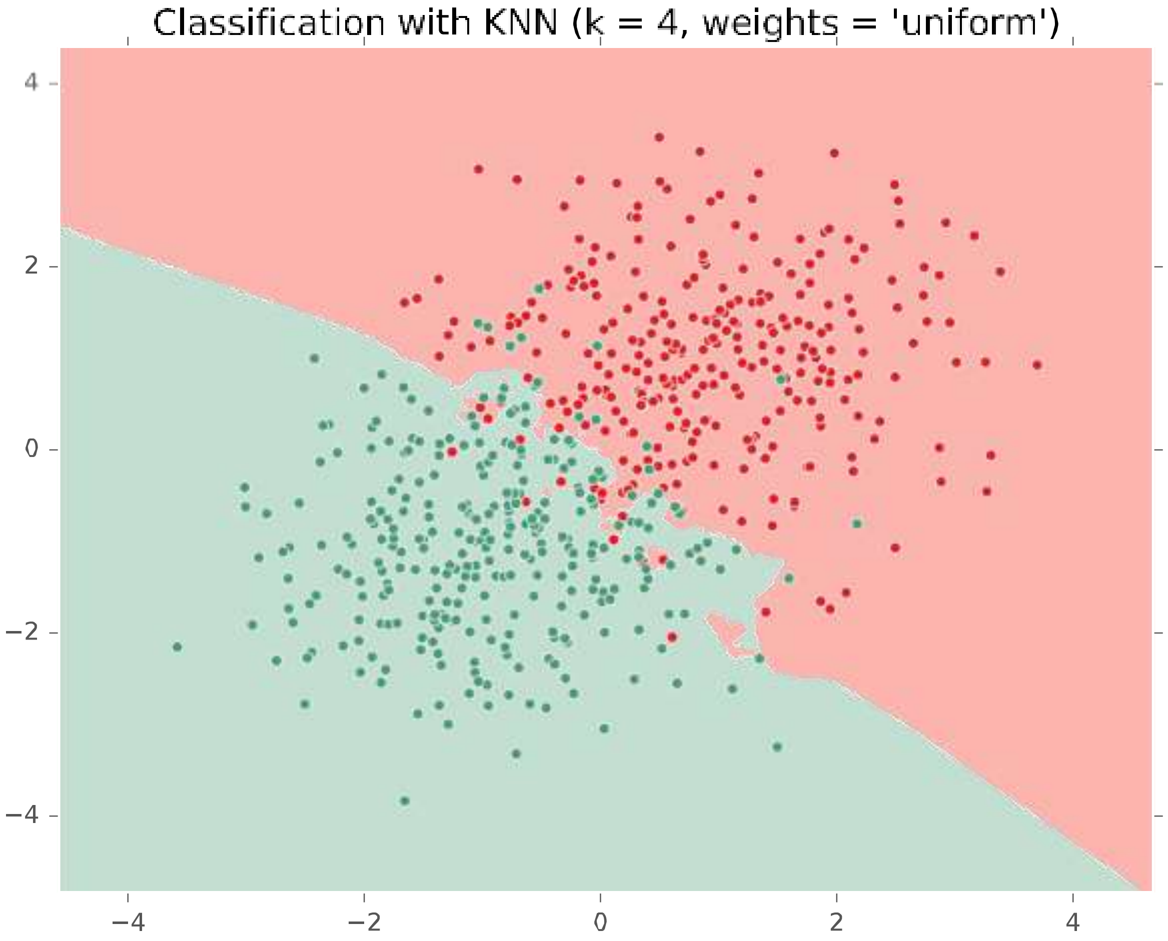
KNN on Gaussian Data

上海科技大学
ShanghaiTech University

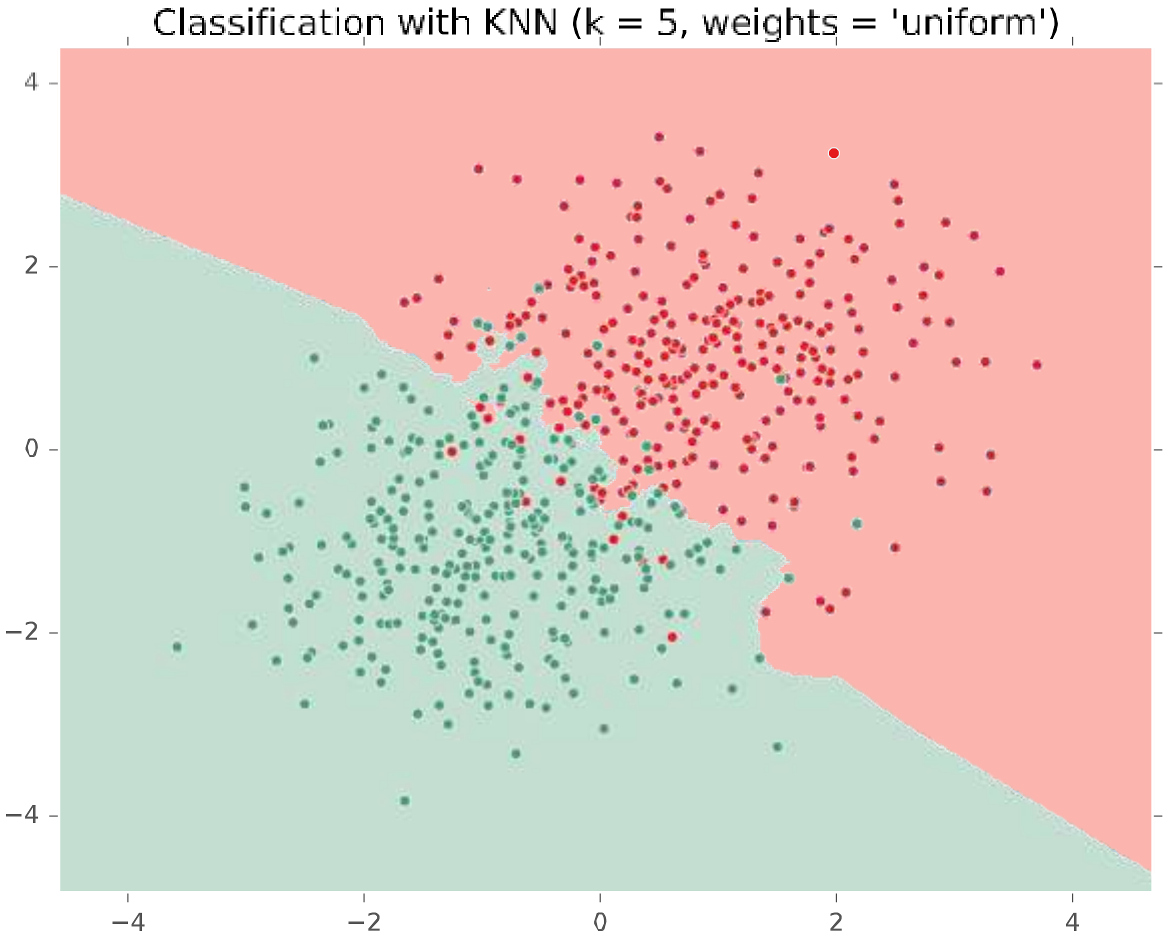


KNN on Gaussian Data

上海科技大学 ShanghaiTech University

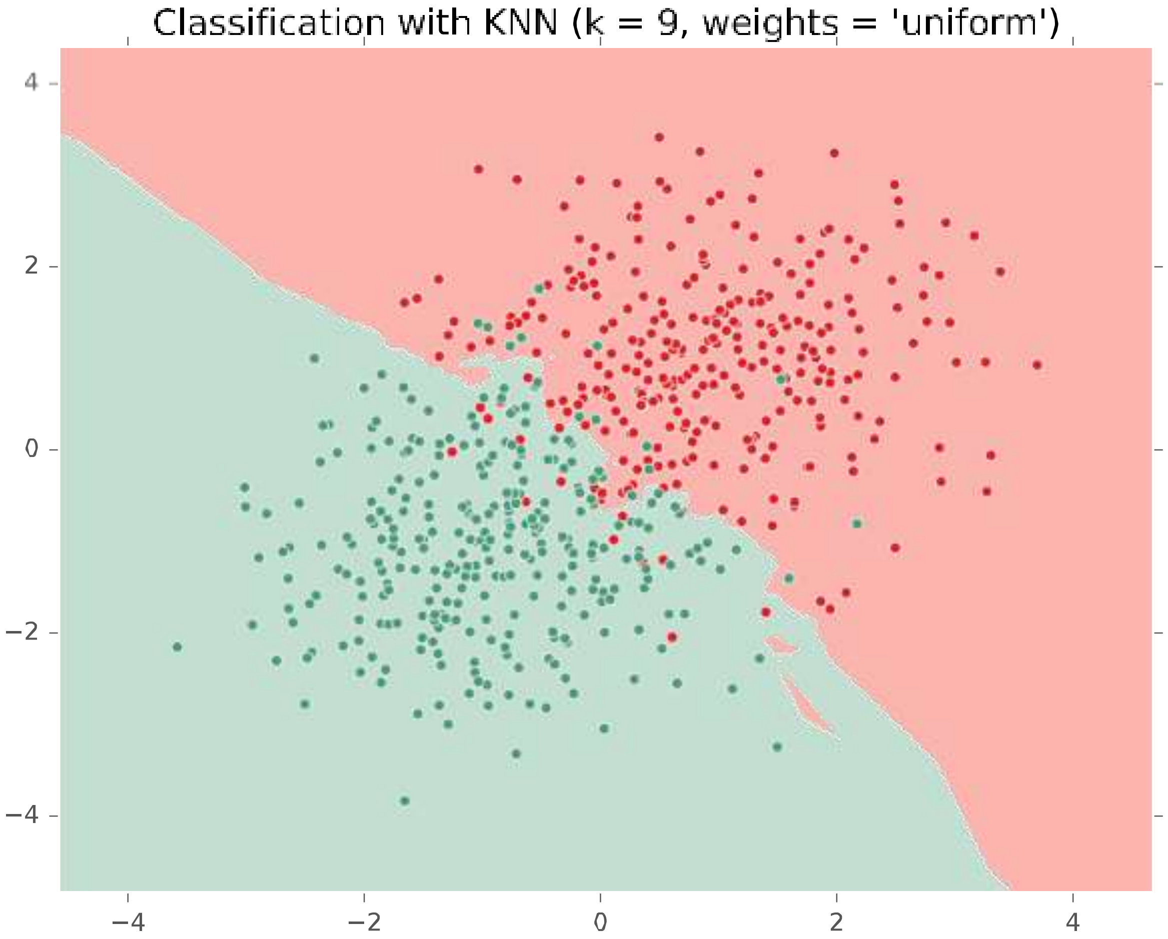


KNN on Gaussian Data



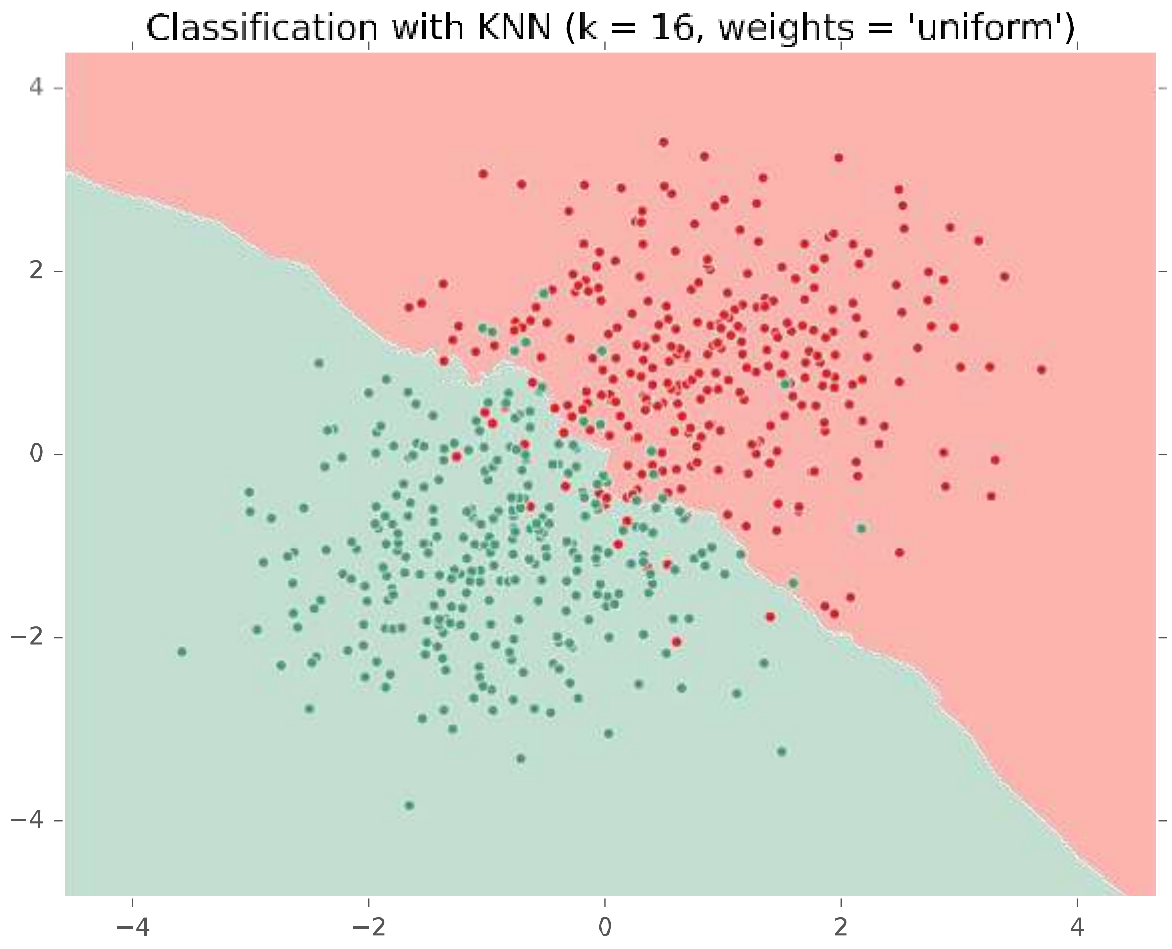
KNN on Gaussian Data

上海科技大学
ShanghaiTech University



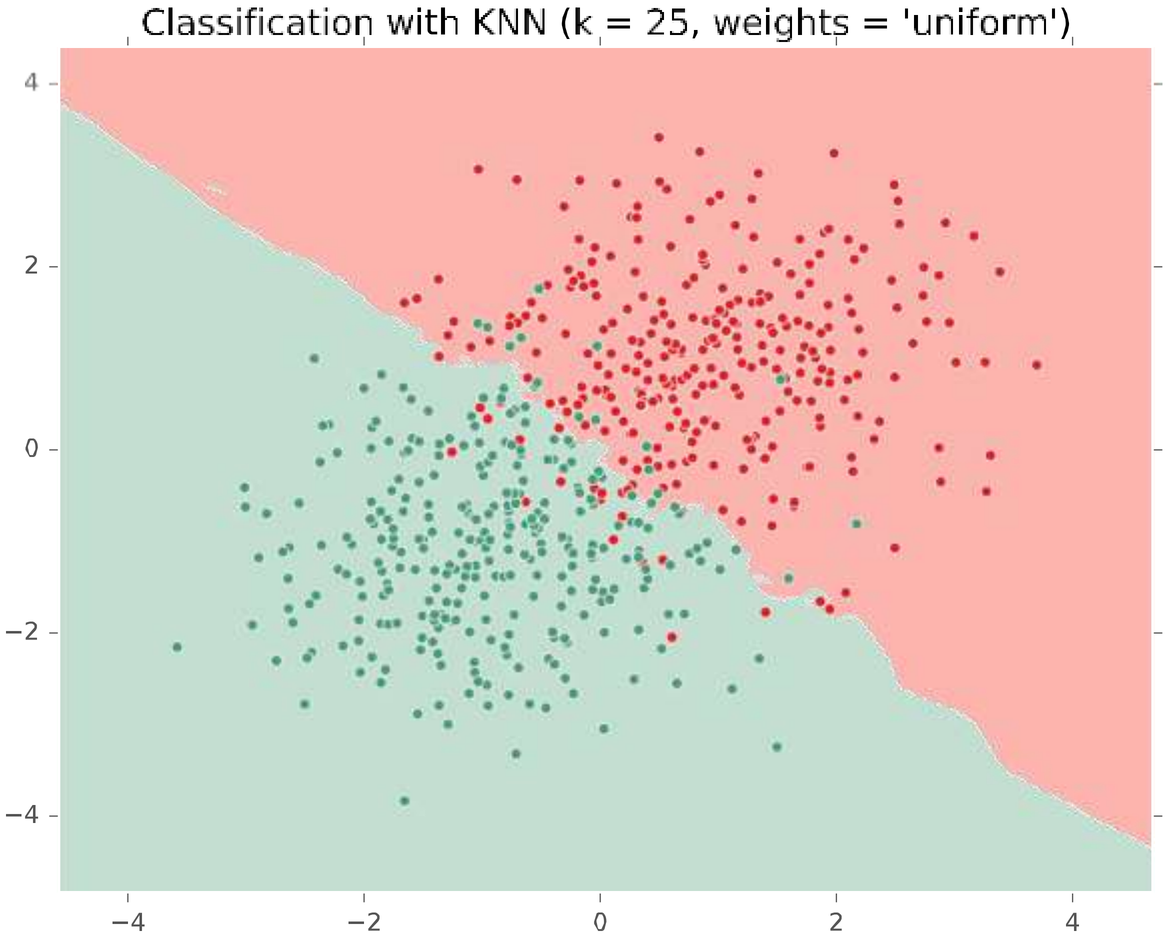
KNN on Gaussian Data

上海科技大学
ShanghaiTech University



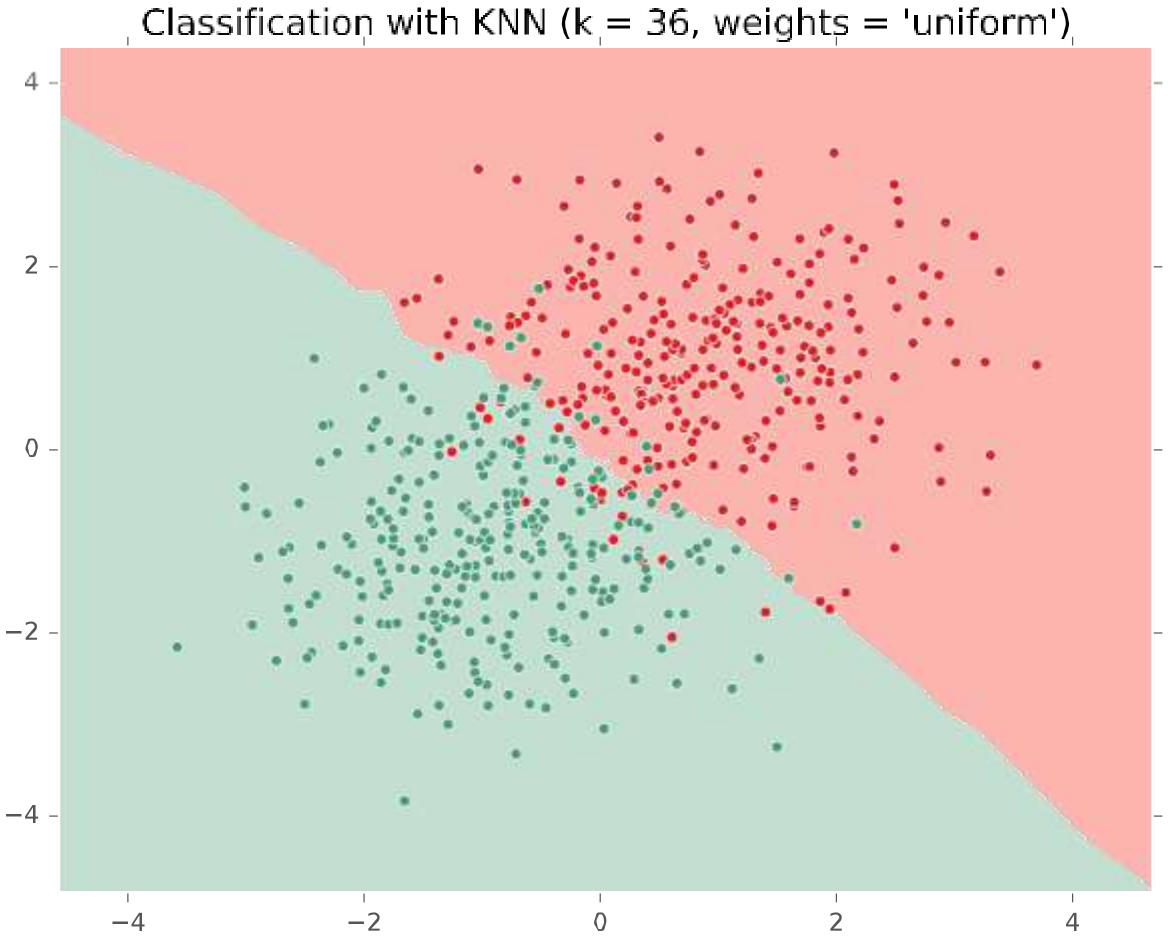
KNN on Gaussian Data

上海科技大学 ShanghaiTech University



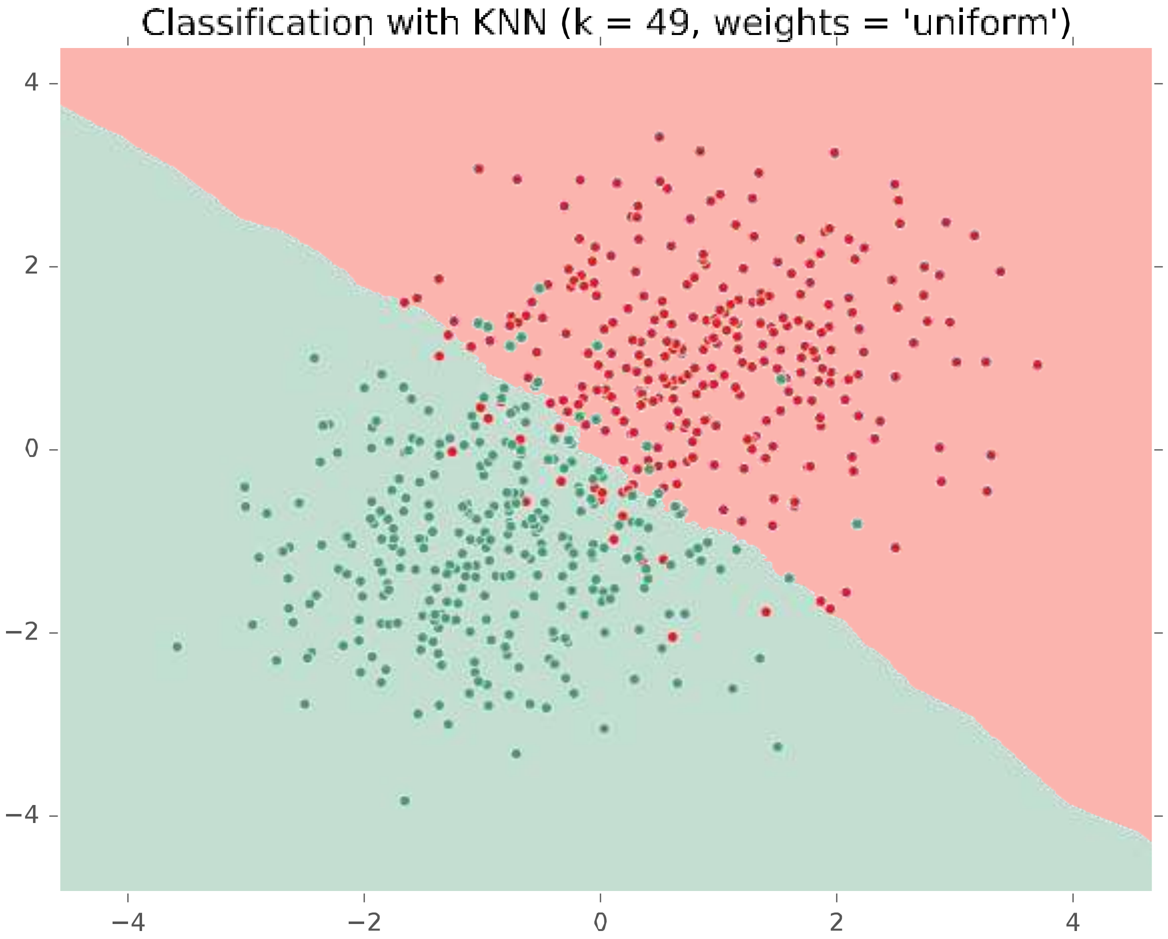
KNN on Gaussian Data

上海科技大学 ShanghaiTech University

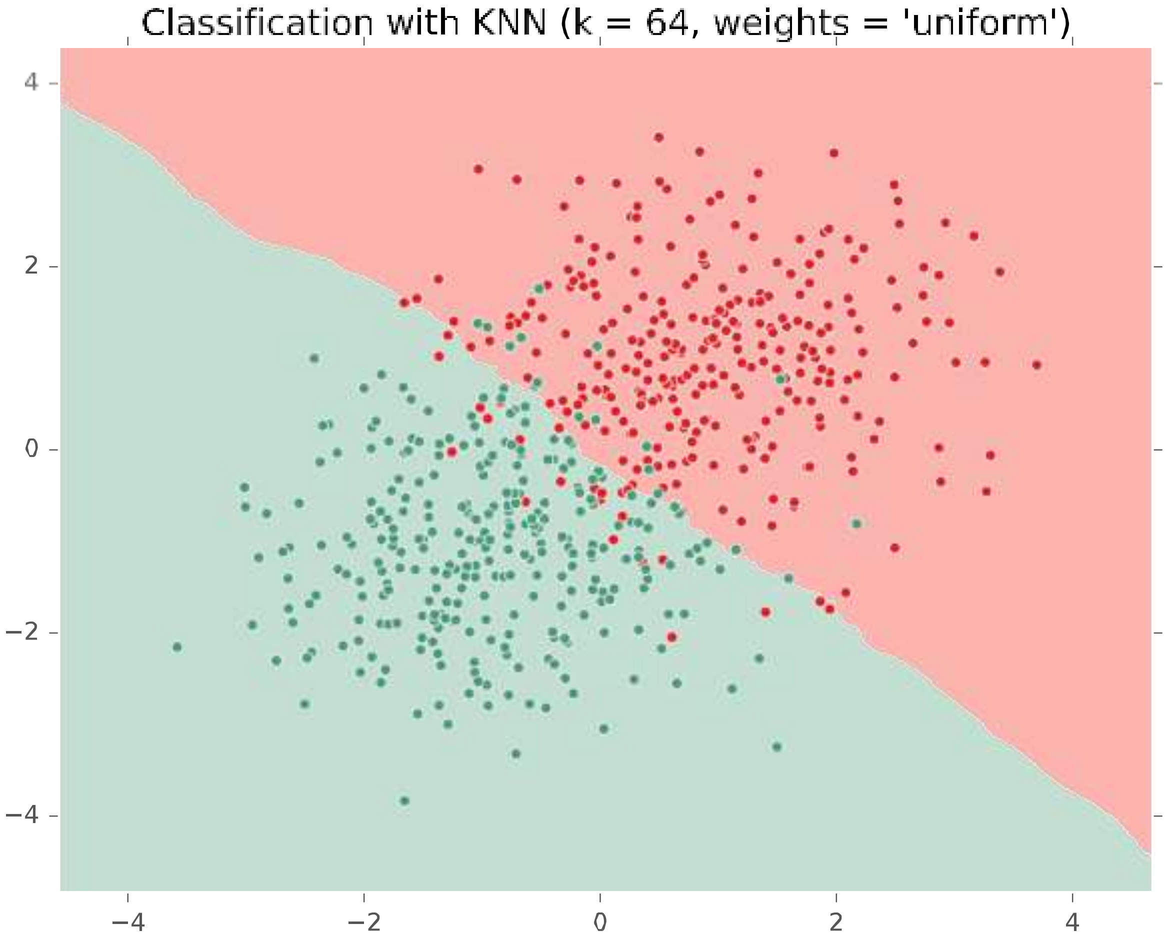


KNN on Gaussian Data

上海科技大学 ShanghaiTech University

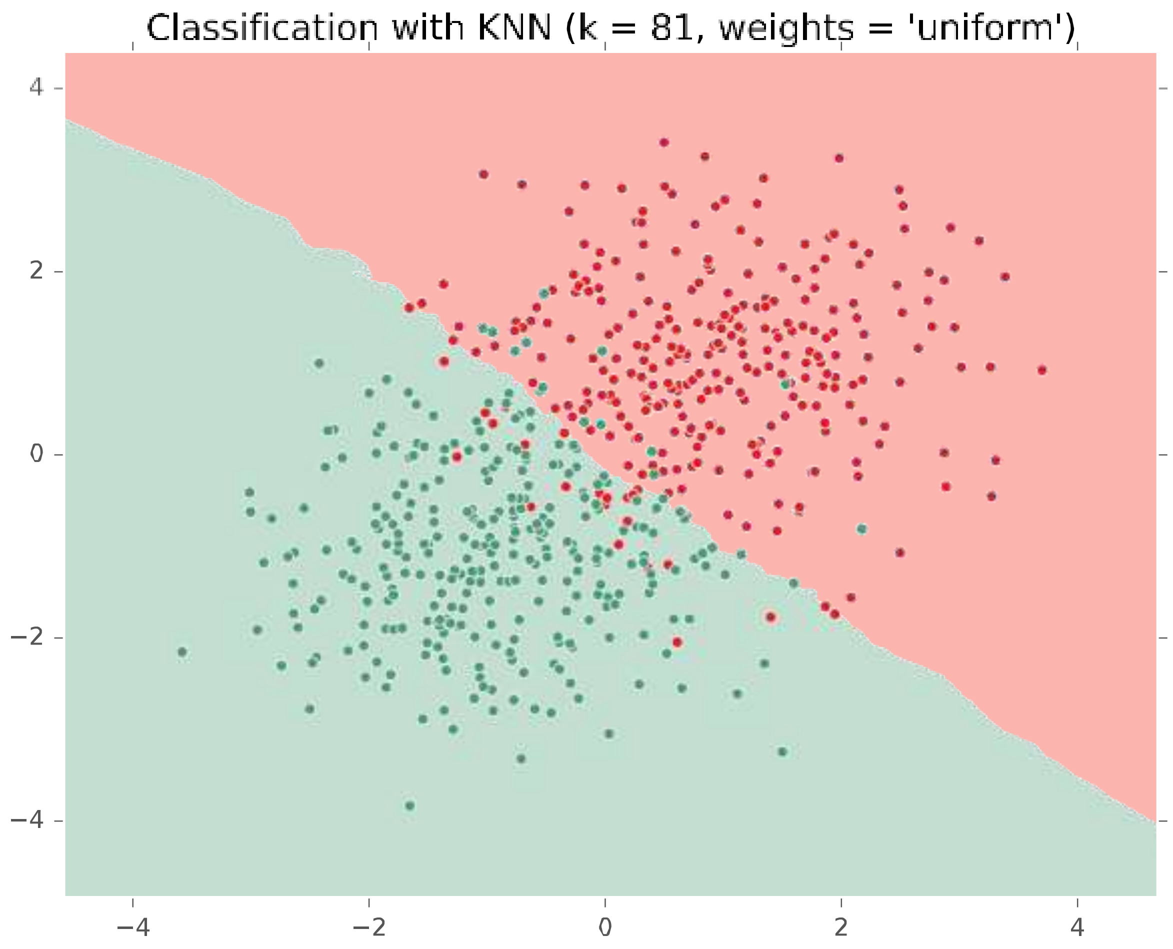


KNN on Gaussian Data



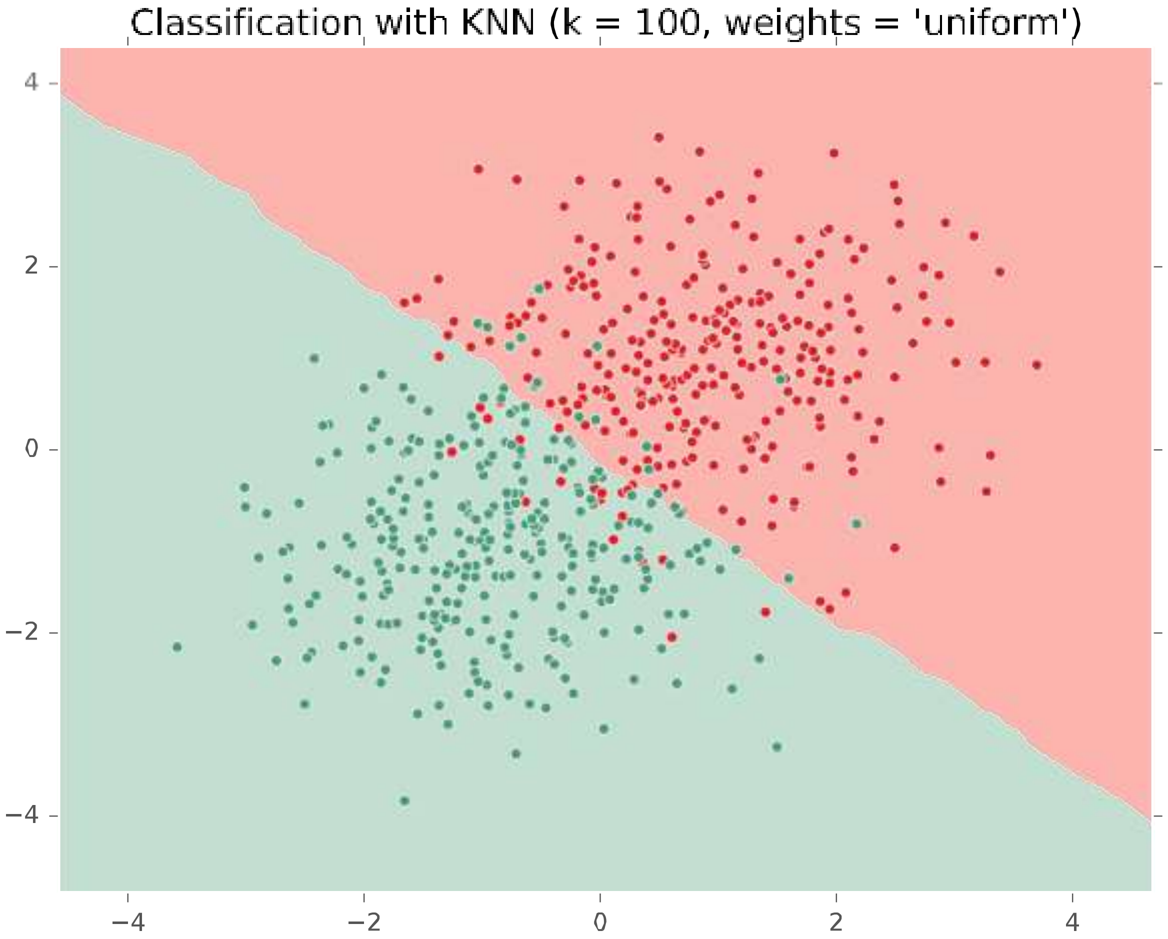
KNN on Gaussian Data

上海科技大学
ShanghaiTech University

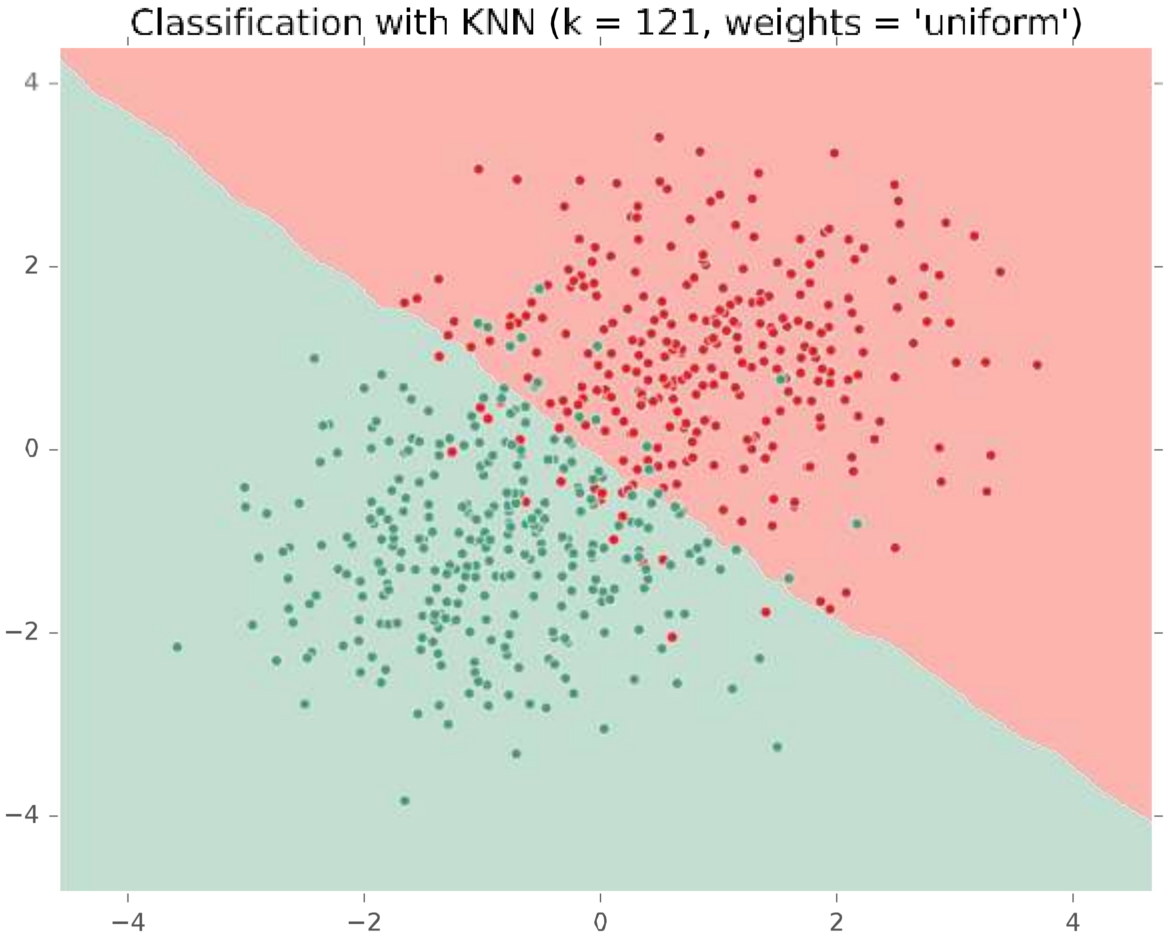


KNN on Gaussian Data

上海科技大学
ShanghaiTech University

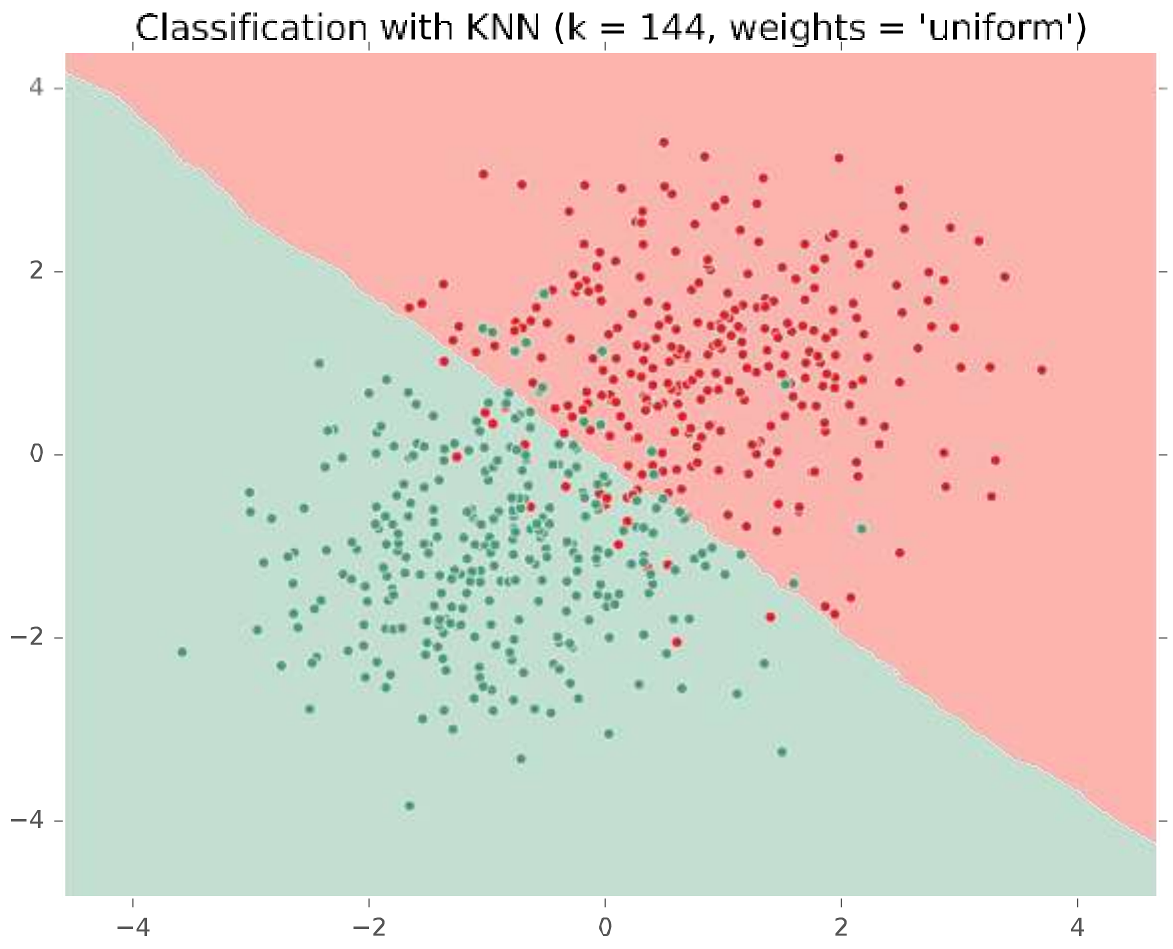


KNN on Gaussian Data



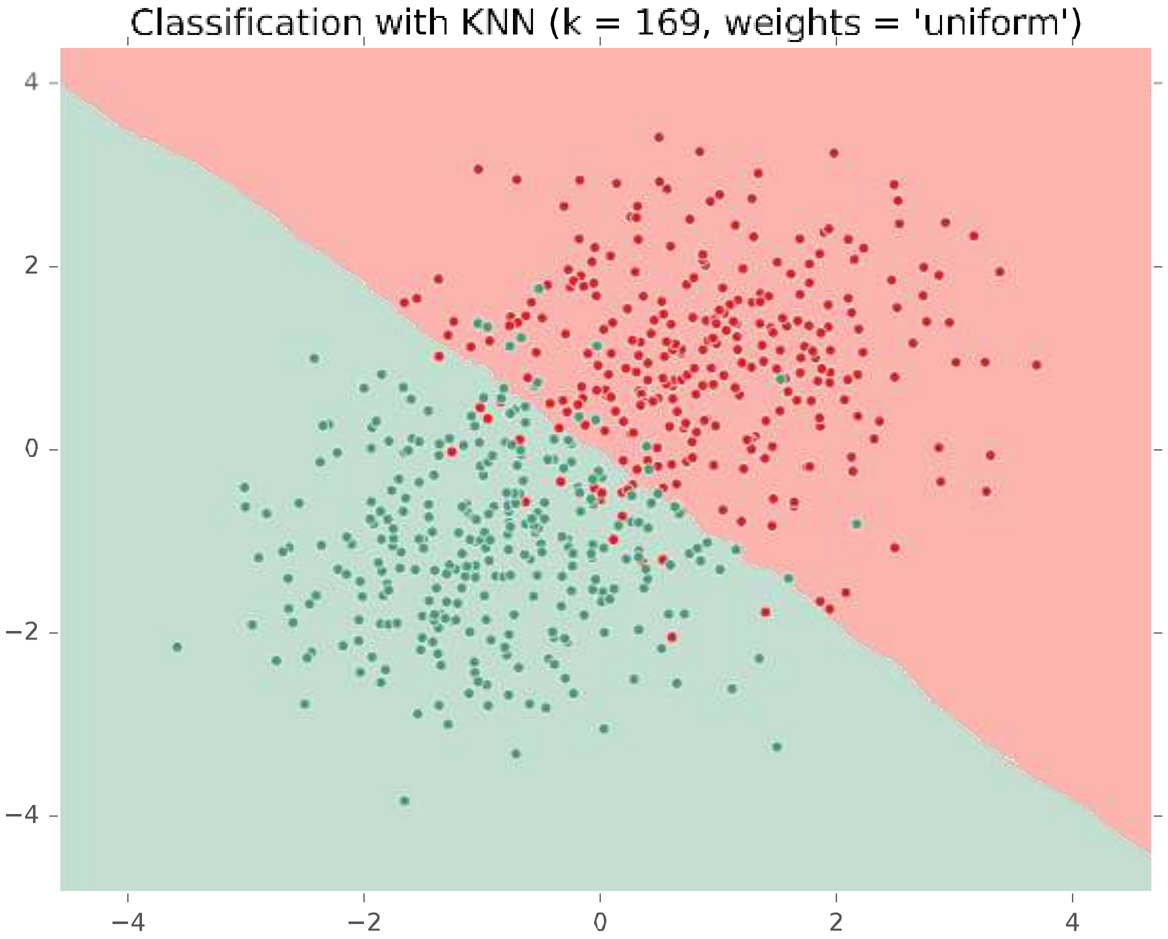
KNN on Gaussian Data

上海科技大学
ShanghaiTech University



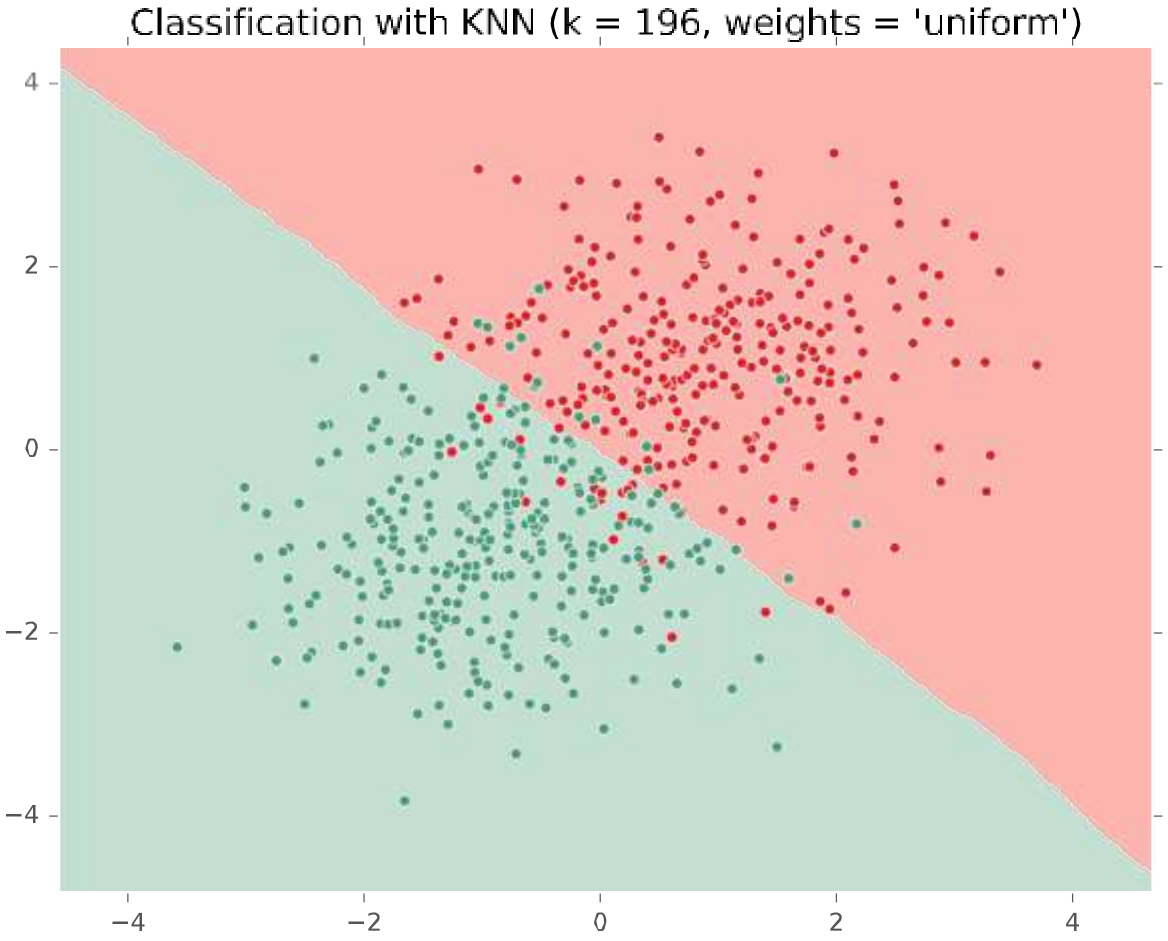
KNN on Gaussian Data

上海科技大学 ShanghaiTech University



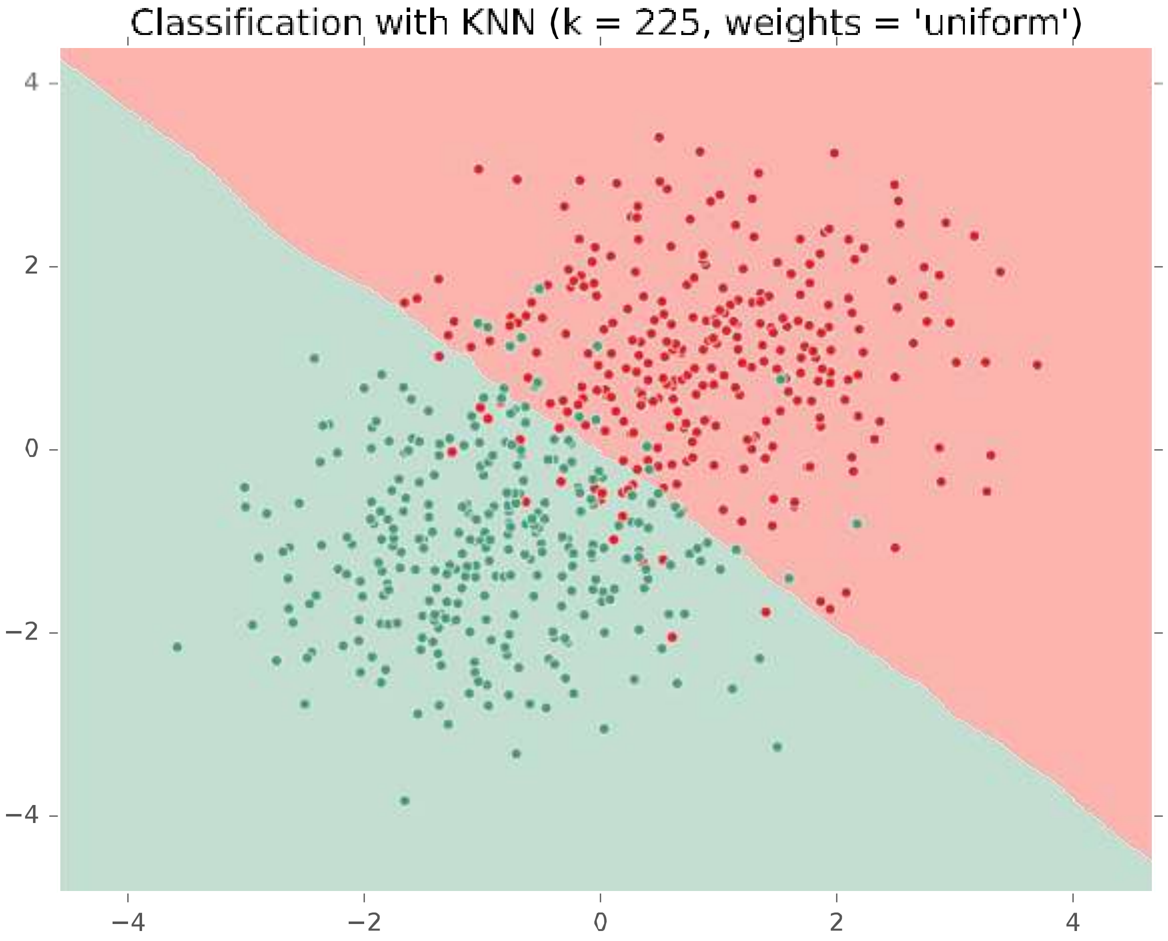
KNN on Gaussian Data

上海科技大学 ShanghaiTech University



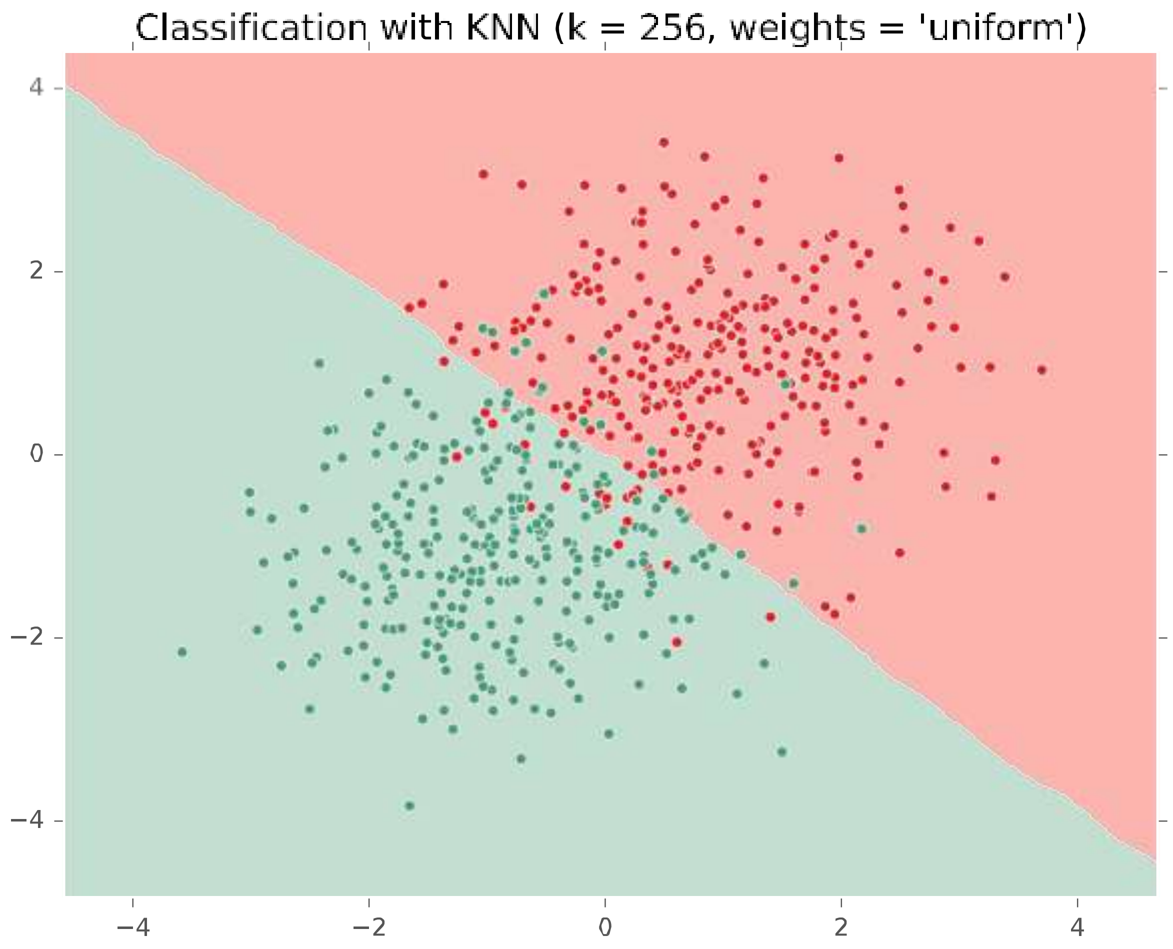
KNN on Gaussian Data

上海科技大学 ShanghaiTech University



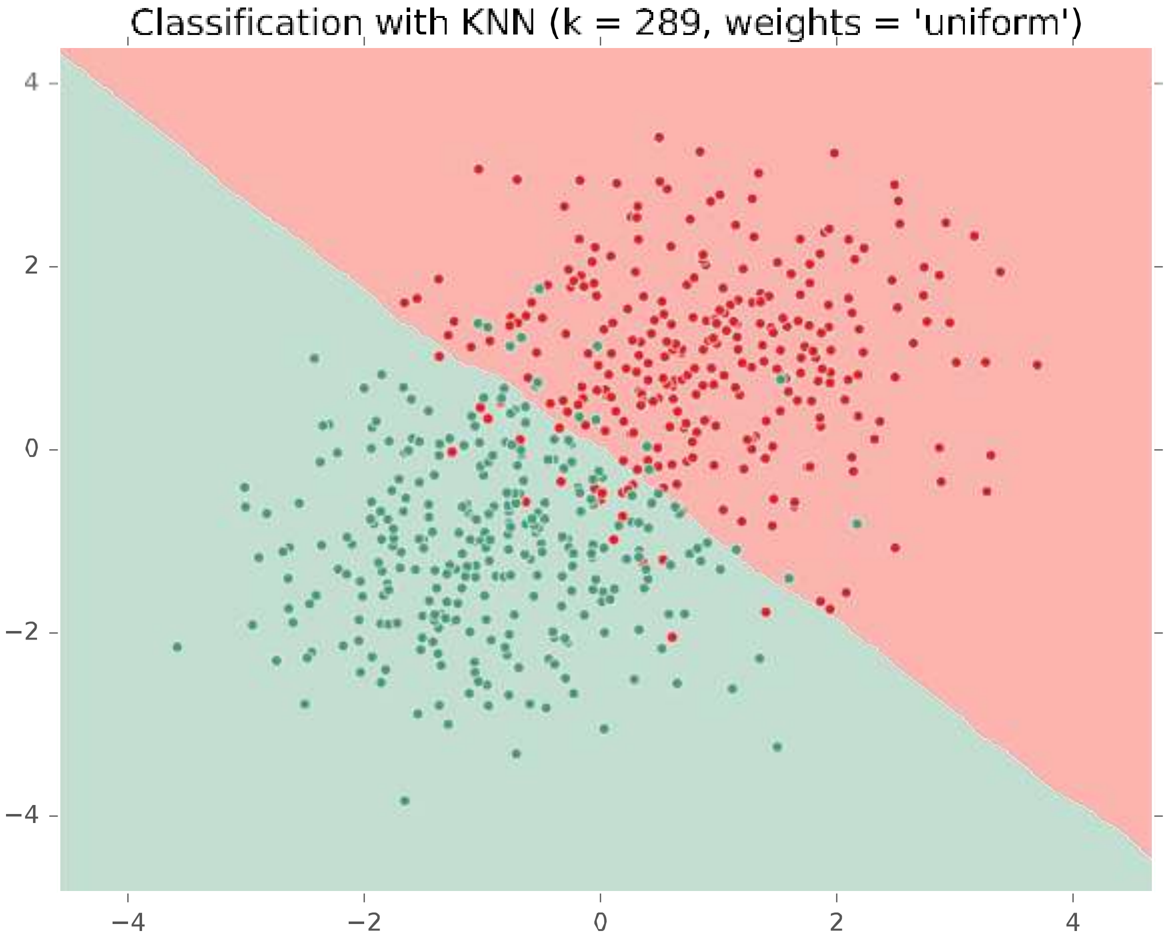
KNN on Gaussian Data

上海科技大学 ShanghaiTech University

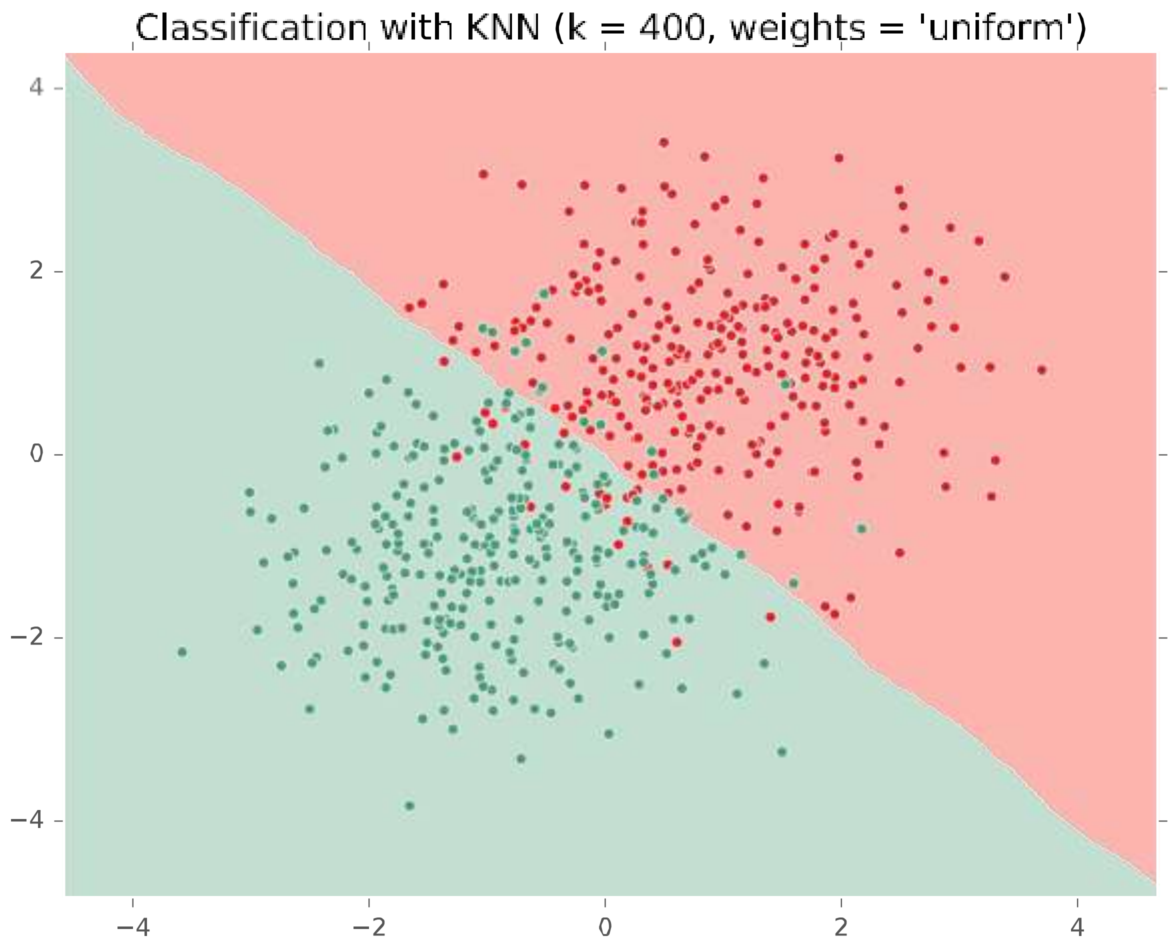


KNN on Gaussian Data

上海科技大学
ShanghaiTech University

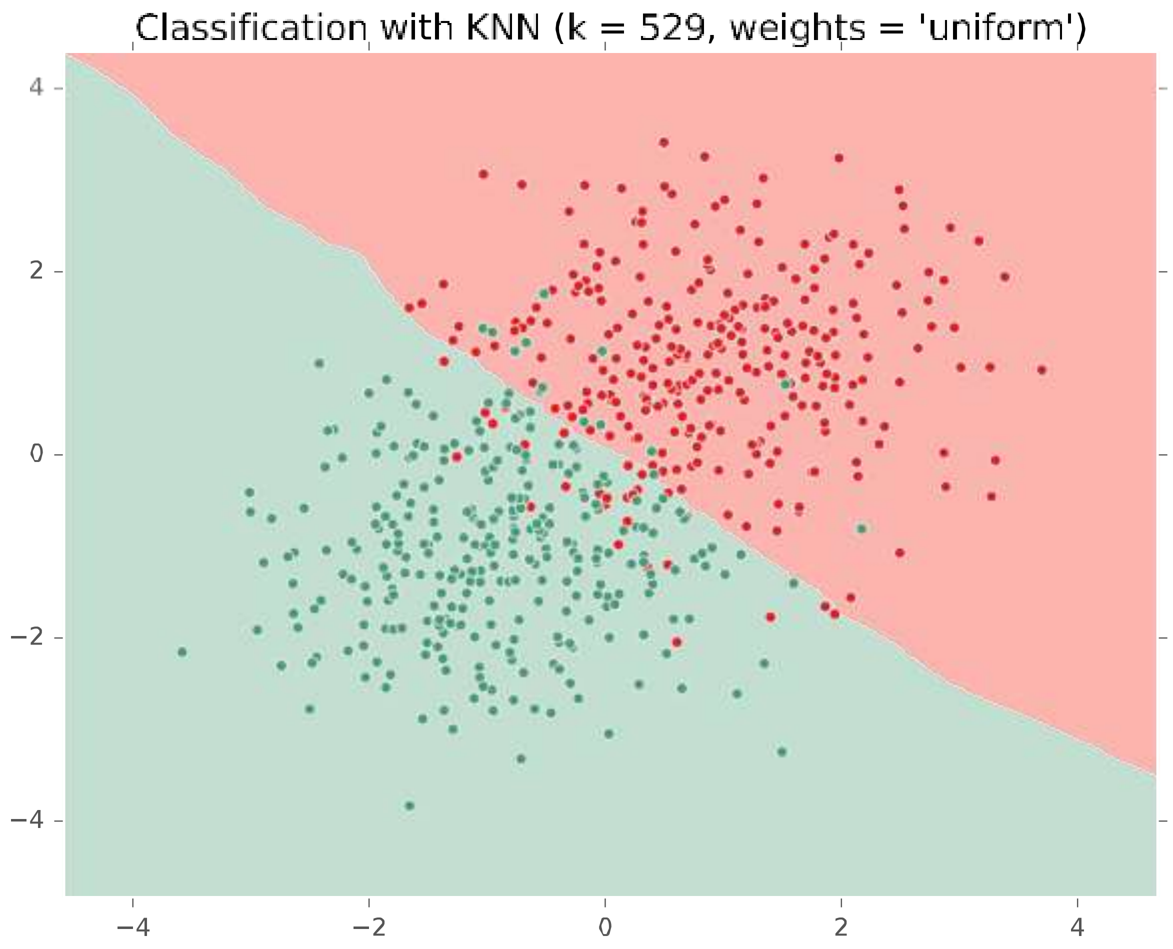


KNN on Gaussian Data



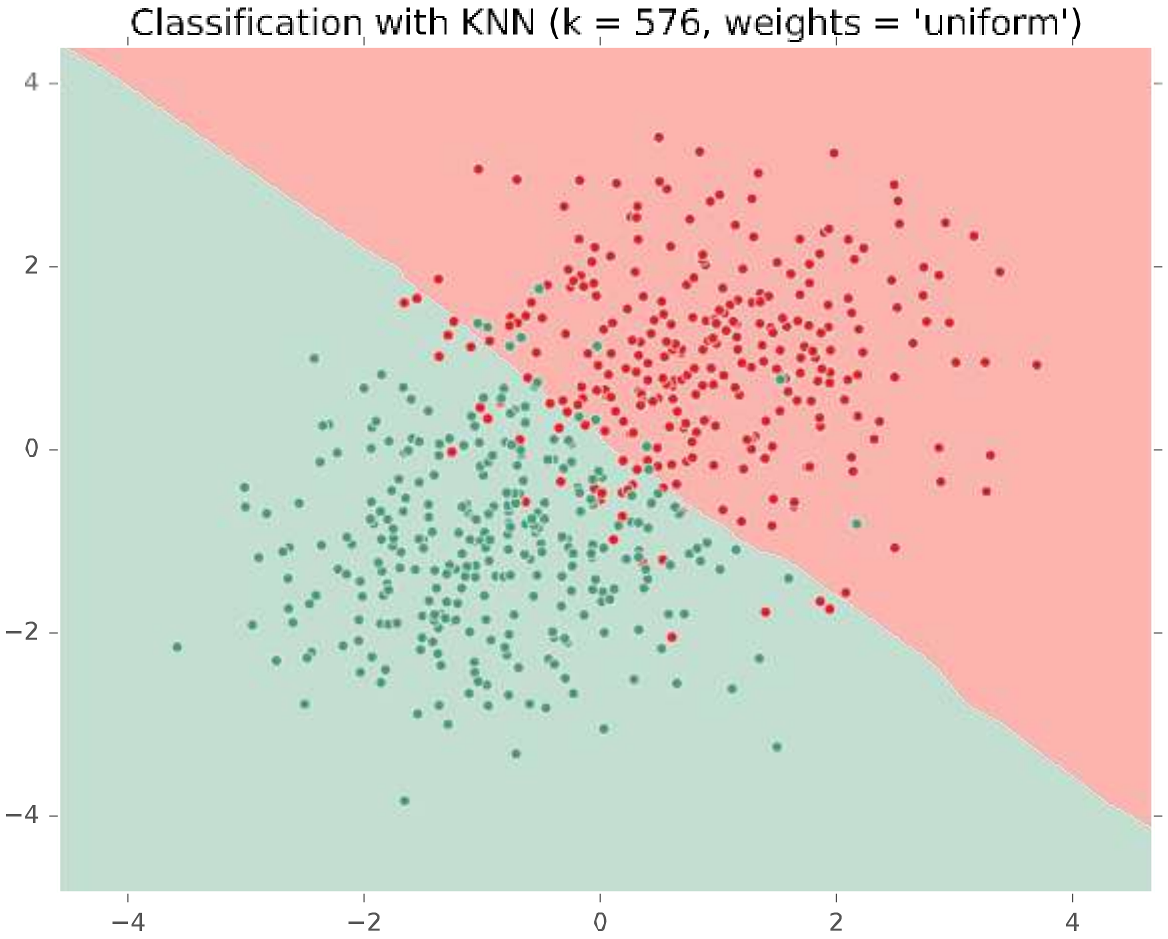
KNN on Gaussian Data

上海科技大学 ShanghaiTech University



KNN on Gaussian Data

上海科技大学
ShanghaiTech University



KNN Learning Objectives

You should be able to…

- Describe a dataset as points in a high dimensional space [CIML]
- Implement k-Nearest Neighbors with O(N) prediction
- Describe the inductive bias of a k-NN classifier and relate it to feature scale [a la. CIML]
- Sketch the decision boundary for a learning algorithm (compare k-NN and DT)
- State Cover & Hart (1967)'s large sample analysis of a nearest neighbor classifier
- Invent "new" k-NN learning algorithms capable of dealing with even k

MODEL SELECTION

Model Selection

WARNING:

- In some sense, our discussion of model selection is premature.
- The models we have considered thus far are fairly simple.
- The models and the many decisions available to the data scientist wielding them will grow to be much more complex than what we've seen so far.

Model Selection

上海科技大学 ShanghaiTech University

Example: Decision Tree

- model = set of all possible trees, possibly restricted by some hyperparameters (e.g. max depth)
- parameters = structure of a specific decision tree
- learning algorithm = ID3, CART, etc.
- hyperparameters = max-depth, threshold for splitting criterion, etc.

Machine Learning

- *Def*: (loosely) a **model** defines the hypothesis space over which learning performs its search
- *Def*: **model parameters** are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
- *Def*: the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- *Def*: **hyperparameters** are the tunable aspects of the model, that the learning algorithm does *not* select

Model Selection

上海科技大学
ShanghaiTech University

Example: k-Nearest Neighbors

- model = set of all possible nearest neighbors classifiers
- parameters = none
(KNN is an instance-based or non-parametric method)
- learning algorithm = for naïve setting, just storing the data
- hyperparameters = *k*, the number of neighbors to consider

Machine Learning

- *Def*: (loosely) a **model** defines the hypothesis space over which learning performs its search
- *Def*: **model parameters** are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
- *Def*: the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- *Def*: **hyperparameters** are the tunable aspects of the model, that the learning algorithm does *not* select

Model Selection

Example: Perceptron

- model = set of all linear separators
- parameters = vector of weights (one for each feature)
- learning algorithm = mistake based updates to the parameters
- hyperparameters = none
(unless using some variant such as averaged perceptron)

Machine Learning

- *Def*: (loosely) a **model** defines the hypothesis space over which learning performs its search
- *Def*: **model parameters** are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
- *Def*: the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- *Def*: **hyperparameters** are the tunable aspects of the model, that the learning algorithm does *not* select

Model Selection

上海科技大学
ShanghaiTech University

Statistics

- *Def*: a **model** defines the data generation process (i.e. a set or family of parametric probability distributions)
- *Def*: **model parameters** are the values that give rise to a particular probability distribution in the model family
- *Def*: **learning** (aka. estimation) is the process of finding the parameters that best fit the data
- *Def*: **hyperparameters** are the parameters of a prior distribution over parameters

Machine Learning

- *Def*: (loosely) a **model** defines the hypothesis space over which learning performs its search
- *Def*: **model parameters** are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
- *Def*: the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- *Def*: **hyperparameters** are the tunable aspects of the model, that the learning algorithm does *not* select

Model Selection

上海科技大学 ShanghaiTech University

Statistics

- *Def*: a **model** defines the data generation process (i.e. a set or family probability distributions)
- *Def*: **model parameters** are give rise to a particular pro distribution in the model fa
- *Def*: **learning** (aka. estimation) is the process of finding the parameters that best fit the data
- *Def*: **hyperparameters** are the parameters of a prior distribution over parameters

If "learning" is all about picking the best **parameters** how do we pick the best **hyperparameters?**

Machine Learning

- *Def*: (loosely) a **model** defines the hypothesis which learning performs its
- **parameters** are the numeric ructure selected by the learning hat give rise to a hypothesis
- *Def*: the learning algorithm defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- *Def*: **hyperparameters** are the tunable aspects of the model, that the learning algorithm does *not* select

Model Selection

上海科技大学
ShanghaiTech University

- Two *very* similar definitions:
  - *Def*: **model selection** is the process by which we choose the "best" model from among a set of candidates
  - *Def*: **hyperparameter optimization** is the process by which we choose the "best" hyperparameters from among a set of candidates **(could be called a special case of model selection)**
- **Both** assume access to a function capable of measuring the quality of a model
- **Both** are typically done "outside" the main training algorithm --- typically training is treated as a black box

EXPERIMENTAL DESIGN

Experimental Design

上海科技大学 ShanghaiTech University

	Input	Output	Notes
Training	• training dataset • hyperparameters	• best model parameters	We pick the best model parameters by learning on the training dataset for a fixed set of hyperparameters
Hyperparameter Optimization	• training dataset • validation dataset	• best hyperparameters	We pick the best hyperparameters by learning on the training data and evaluating error on the validation error
Testing	• test dataset • hypothesis (i.e. fixed model parameters)	• test error	We evaluate a hypothesis corresponding to a decision rule with fixed model parameters on a test dataset to obtain test error

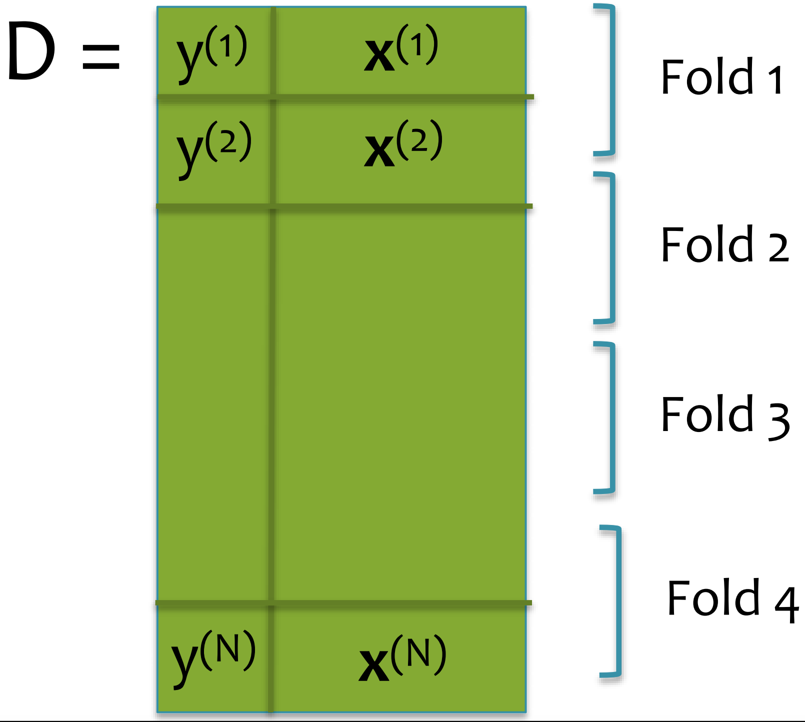
Choosing K for KNN

Cross-Validation

Cross validation is a method of estimating loss on held out data

- **Input:** training data, learning algorithm, loss function (e.g. 0/1 error)
- **Output:** an estimate of loss function on held-out data

Key idea: rather than just a single "validation" set, use many!
(Error is more stable. Slower computation.)



Algorithm:
Divide data into folds (e.g. 4)

1. Train on folds {1,2,3} and predict on {4}
2. Train on folds {1,2,4} and predict on {3}
3. Train on folds {1,3,4} and predict on {2}
4. Train on folds {2,3,4} and predict on {1}

Concatenate all the predictions and evaluate loss (*almost* equivalent to averaging loss over the folds)

Definition:
N-fold cross validation = cross validation with N folds

Experimental Design

上海科技大学
ShanghaiTech University

	Input	Output	Notes
Training	• training dataset • hyperparameters	• best model parameters	We pick the best model parameters by learning on the training dataset for a fixed set of hyperparameters
Hyperparameter Optimization	• training dataset • validation dataset	• best hyperparameters	We pick the best hyperparameters by learning on the training data and evaluating error on the validation error
Cross-Validation	• training dataset • validation dataset	• cross-validation error	We estimate the error on held out data by repeatedly training on N-1 folds and predicting on the held-out fold
Testing	• test dataset • hypothesis (i.e. fixed model parameters)	• test error	We evaluate a hypothesis corresponding to a decision rule with fixed model parameters on a test dataset to obtain test error

Experimental Design

上海科技大学 ShanghaiTech University

Q: We pick the best hyperparameters by learning on the training data and evaluating error on the validation data. For our final model, should we also learn from just the training data?

A: No!

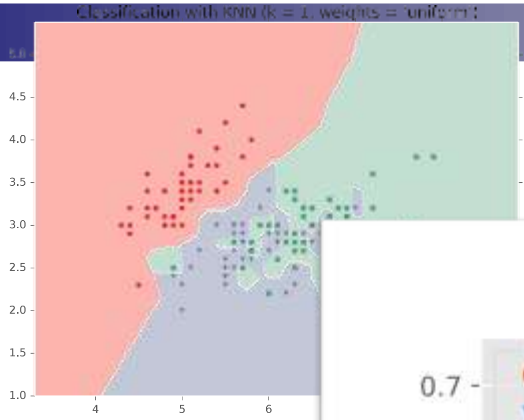
Let's assume that {train-original} is the original training data and {test} is the provided test dataset.

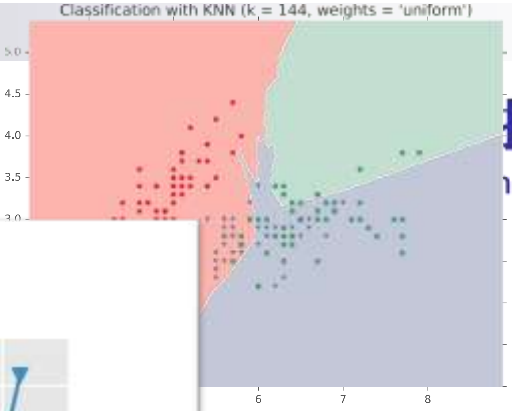
1. Split {train-original} into {train-subset} and {validation}.
2. Pick the hyperparameters that when training on {train-subset} give the lowest error on {validation}. Call these hyperparameters {best-hyper}.
3. Retrain a new model using {best-hyper} on {train-original} = {train-subset} ∪ {validation}.
4. Report test error by evaluating on {test}.

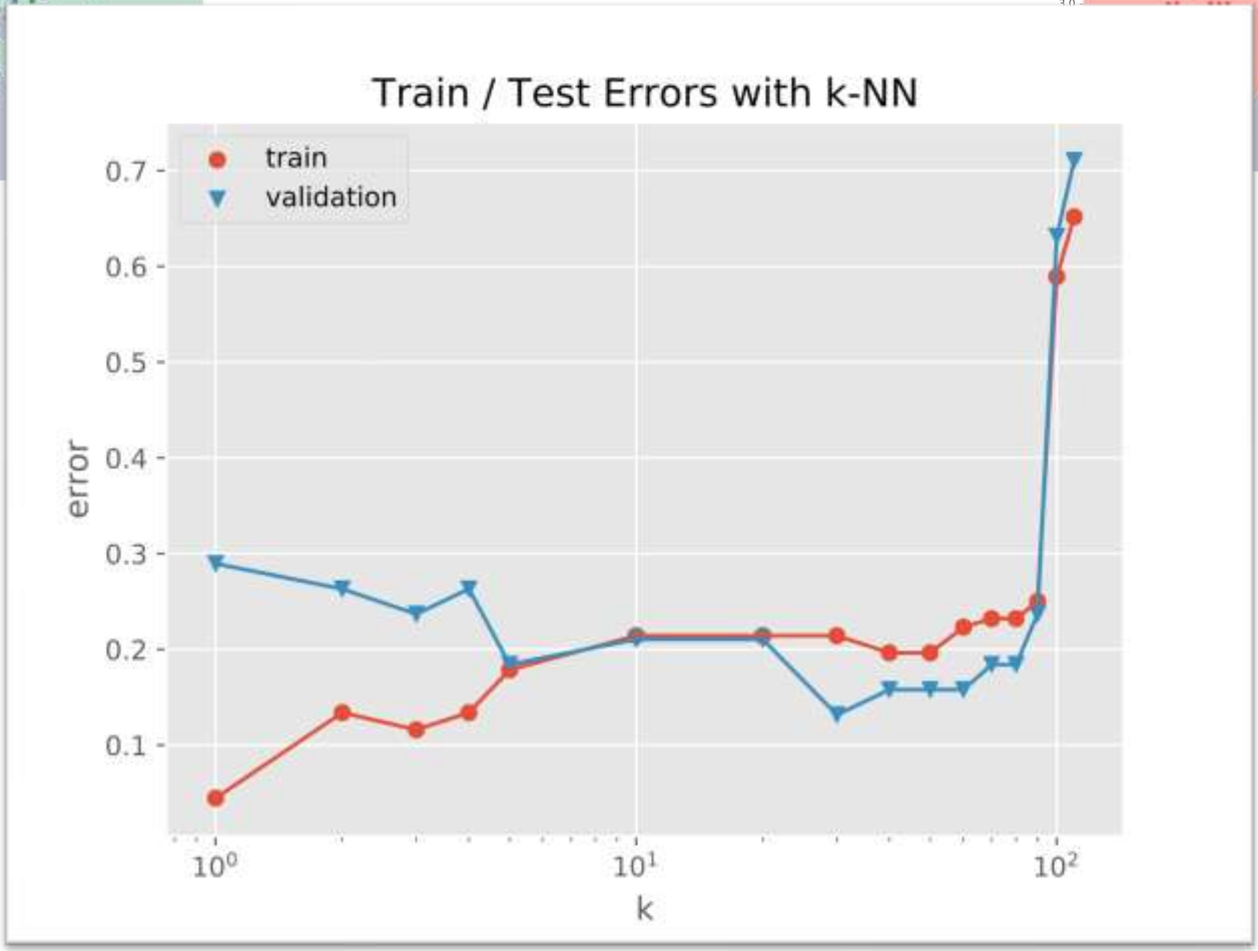
Alternatively, you could replace Steps 1-2 with the following:

1. Pick the hyperparameters that give the lowest cross-validation error on {train-original}. Call these hyperparameters {best-hyper}.

k-NN: Choosing k

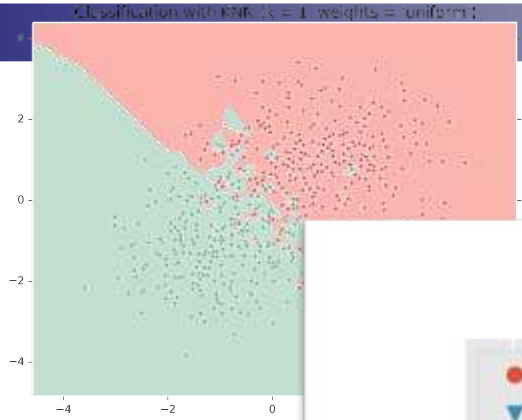


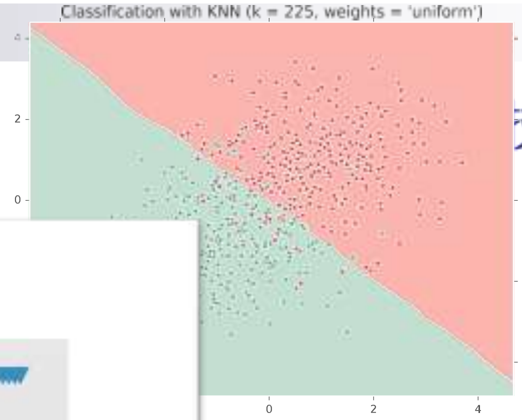


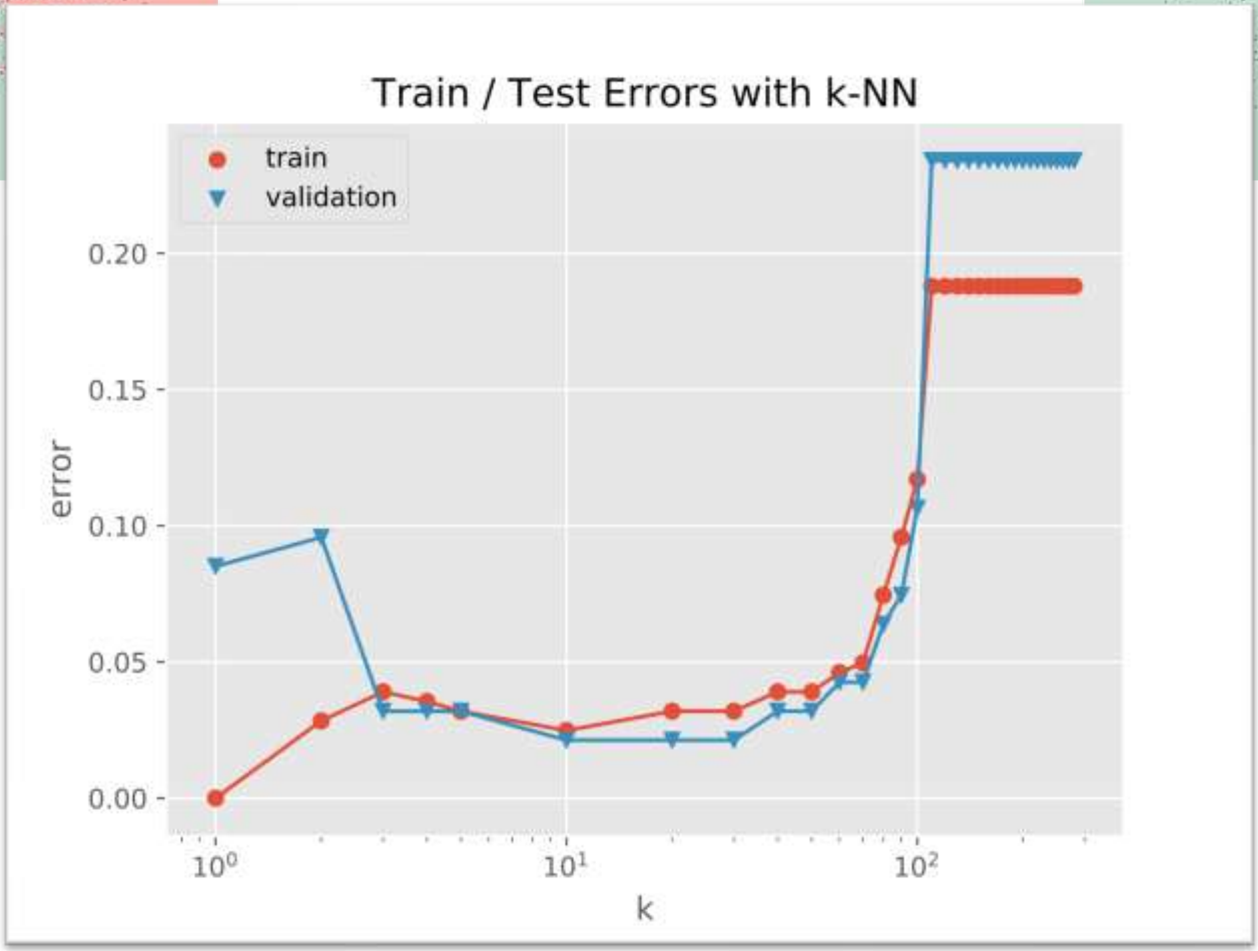


Fisher Iris Data: varying the value of k

k-NN: Choosing k







Gaussian Data: varying the value of k

HYPERPARAMETER OPTIMIZATION

WARNING (again):

– This section is only scratching the surface!
– Lots of methods for hyperparameter optimization: (to talk about later)
    - Grid search
    - Random search
    - Bayesian optimization
    - Graduate-student descent
    - …

Main Takeaway:

– Model selection / hyperparameter optimization is just another form of learning

Hyperparameter Optimization

上海科技大学 ShanghaiTech University

Setting: suppose we have hyperparameters α, β, and χ and we wish to pick the "best" values for each one

Algorithm 1: Grid Search

- Pick a set of values for each hyperparameter
  $\alpha \in \{a_1, a_2, \dots, a_n\}$, $\beta \in \{b_1, b_2, \dots, b_n\}$, and $\chi \in \{c_1, c_2, \dots, c_n\}$
- Run a grid search

```
for α ∈ {a₁, a₂, … , aₙ}:
    for β ∈ {b₁, b₂, … , bₙ}:
        for χ ∈ {c₁, c₂, … , cₙ}:
            θ = train(D_train; α, β, χ)
             error = predict(D_validation; θ)
```

- return α, β, and χ with lowest validation error

Hyperparameter Optimization

Setting: suppose we have hyperparameters α, β, and χ and we wish to pick the "best" values for each one

Algorithm 2: Random Search

- Pick a range of values for each parameter
  $\alpha \in \{a_1, a_2, \dots, a_n\}$, $\beta \in \{b_1, b_2, \dots, b_n\}$, and $\chi \in \{c_1, c_2, \dots, c_n\}$
- Run a random search

  for $t = 1, 2, \dots, T$:
  sample α uniformly from $\{a_1, a_2, \dots, a_n\}$
  sample β uniformly from $\{b_1, b_2, \dots, b_n\}$
  sample χ uniformly from $\{c_1, c_2, \dots, c_n\}$
  $\theta = \text{train}(D_{train}; \alpha, \beta, \chi)$
  $\text{error} = \text{predict}(D_{validation}; \theta)$

- return α, β, and χ with lowest validation error

Hyperparameter Optimization

上海科技大学
ShanghaiTech University

Question:

True or False: given a finite amount of computation time, grid search is more likely to find good values for hyperparameters than random search.

Hyperparameter Optimization

Question:

True or False: given a finite amount of computation time, grid search is more likely to find good values for hyperparameters than random search.

Answer:

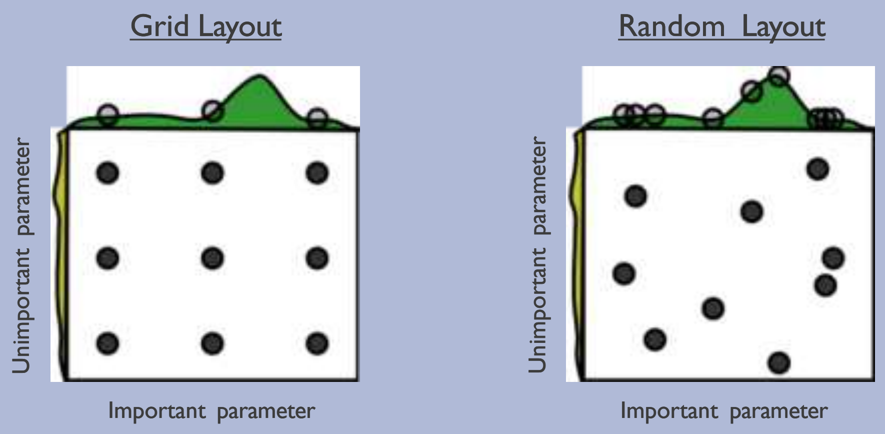


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Figure from Bergstra & Bengio (2012)

Model Selection Learning Objectives

You should be able to…

- Plan an experiment that uses training, validation, and test datasets to predict the performance of a classifier on unseen data (without cheating)
- Explain the difference between (1) training error, (2) validation error, (3) cross-validation error, (4) test error, and (5) true error
- For a given learning technique, identify the model, learning algorithm, parameters, and hyperparamters
- Define "instance-based learning" or "nonparametric methods"
- Select an appropriate algorithm for optimizing (aka. learning) hyperparameters

Reminders

- Homework 1: Decision Trees
  - Out: Thu, Feb. 20
  - Due: Fri, Mar. 7 at 11:59pm
- Homework 2: Decision Trees
  - Out: Tue, Feb. 25
  - Due: Fri, Mar. 14 at 11:59pm
- HW Recitation
  - 20:00~21:30pm Wed
  - Weekly