Classification Algorithms

Week 10 Day 02

DS 3000 – Foundations of Data Science

1

Reminders

HW 6

Thursday, November 7

FP3

Thursday, November 14

2

Outline

Case Study: Digits Dataset

Classification Algorithms

Model Evaluation

Cross Validation

3

Steps in a ML Case Study

1. Load the dataset
2. Explore the data with pandas and visualizations
3. Transform your data (variable coding, normalization, etc.)
4. Split the data for training and testing
5. Create the model
6. Train and test the model
7. Tune the model and evaluate its accuracy
8. Make predictions on live data that the model hasn’t seen before

4

k-Nearest Neighbors

5

Birds of a feather flock together

6

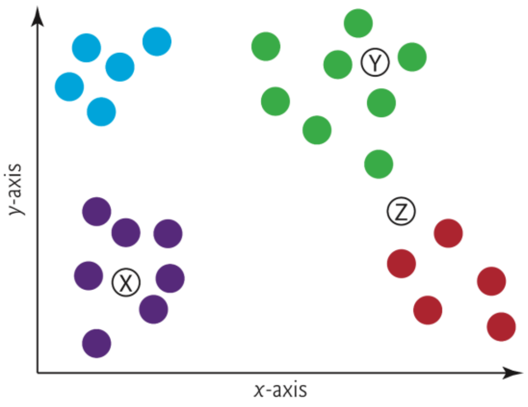
k-Nearest Neighbors

Predict a sample’s class by looking at the ***k* training samples nearest in "distance"** to the **sample**

Filled dots represent four distinct classes A (blue), B (green), C (red) and D (purple)

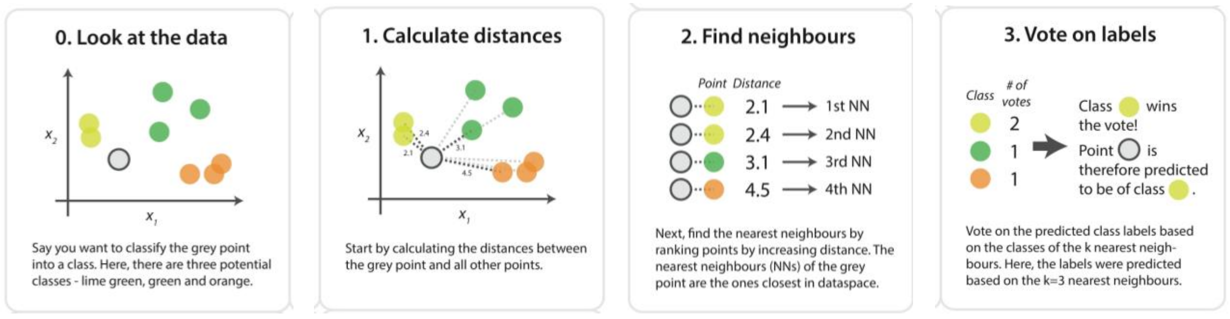
Class with the most “votes” wins

Odd *k* value avoids ties — there’s never an equal number of votes



7

k-Nearest Neighbors Algorithm



Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

Start by calculating the distances between the grey point and all other points.

Point	Distance	
○··●	2.1	⟶ 1st NN
○··●	2.4	⟶ 2nd NN
○··●	3.1	⟶ 3rd NN
○··●	4.5	⟶ 4th NN

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

Class	# of votes
●	2
●	1
●	1

Class ● wins the vote! Point ○ is therefore predicted to be of class ●.

Vote on the predicted class labels based on the classes of the k nearest neigh-bours. Here, the labels were predicted based on the k=3 nearest neighbours.

8

k-Nearest Neighbors Algorithm

Given a training set with features and labels, and given a new instance to be classified:

1. Find the **k-**most similar instances to the new sample in the training set
   - based on distance between the new sample and instances

2. Get the labels of the **k-**most similar instances in the training set

3. Predict the label for the new sample by combining the labels of the **k-**most similar instances
   e.g. simple majority vote

9

k-Nearest Neighbors

Pros	**Cons**
Easy to understand	With large training sets (# of features or samples), prediction can be slow.
Reasonable performance without a lot of adjustments	Not ideal for datasets with many features (hundreds or more) or with sparse data
Model is built really fast	

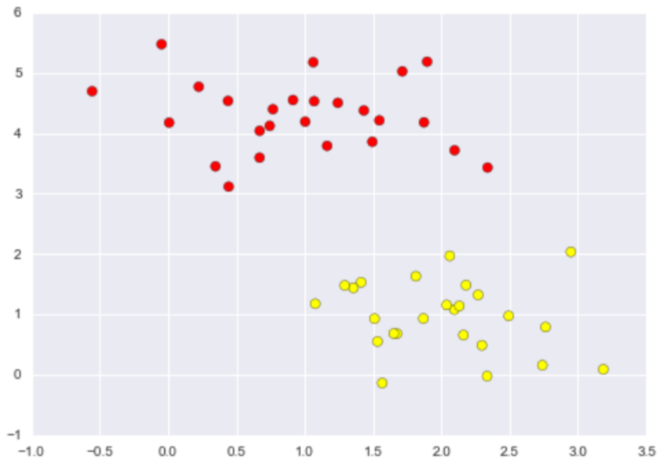
10

Support Vector Machines

11

Support Vector Machines

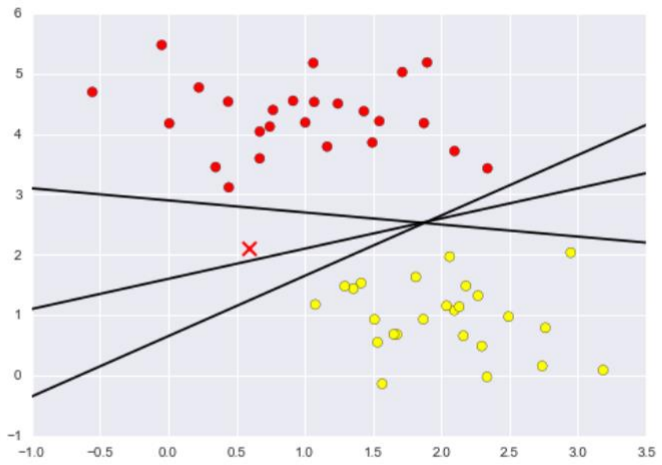
Goal of classification: make sure classes are well separated



12

Support Vector Machines

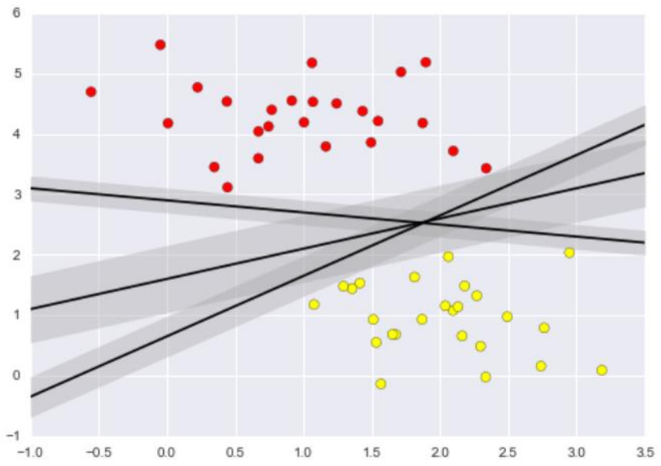
Possible to draw many dividing lines



13

Support Vector Machines: Maximum Margin

SVM finds and uses the line that maximizes the margin between the line and nearest data points

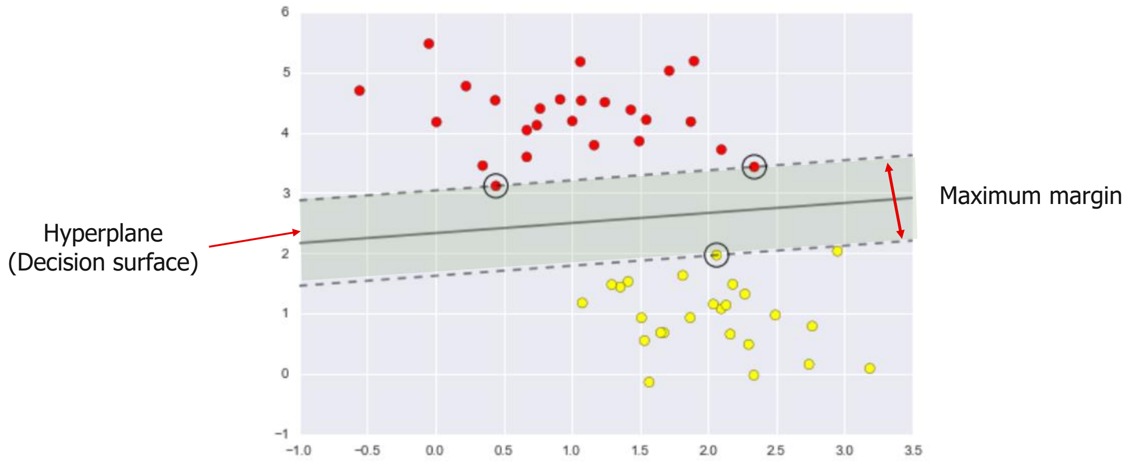


14

Support Vector Machines: Model Fit

The points touching the margins are called Support Vectors

- Most difficult to classify (and thus influence the location of hyperplane)



15

Support Vector Machines: Model Fit

The strength of regularization is determined by C

Larger values of C: less regularization

- Fit the training data as well as possible
- Each individual data point is important to classify correctly

Smaller values of C: more regularization

- More tolerant of errors on individual data points

16

Support Vector Machines

Pros	Cons
Simple and easy to train	Not ideal for datasets where features are not linearly separable (can use kernels to transform the data)
Fast prediction	
Scales well to very large datasets	For low-dimensional datasets, other classifiers may yield better performance
Works well with sparse data (0s)	

17

Naïve-Bayes

19

Naïve-Bayes

A probabilistic classification algorithm

$$P(L \mid \text{features}) = \frac{P(\text{features} \mid L)P(L)}{P(\text{features})}$$

Two assumptions:

Features are equally important a priori

Features are statistically independent

i.e., knowing the value of one attribute says nothing about the value of another (if the class is known)

Independence assumption is not usually met

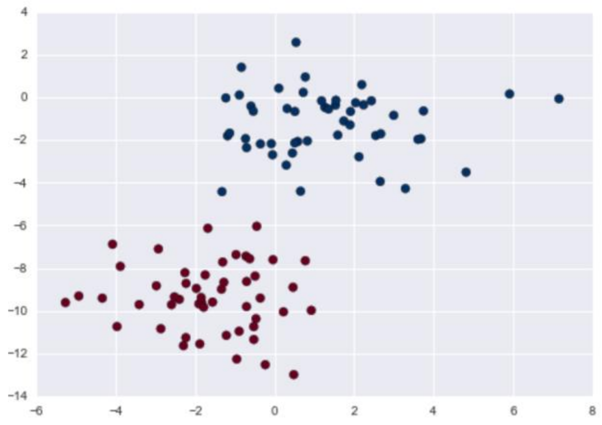
Often works well in practice though

20

Gaussian Naïve-Bayes

Assumption

Data from each label is drawn from a simple Gaussian distribution

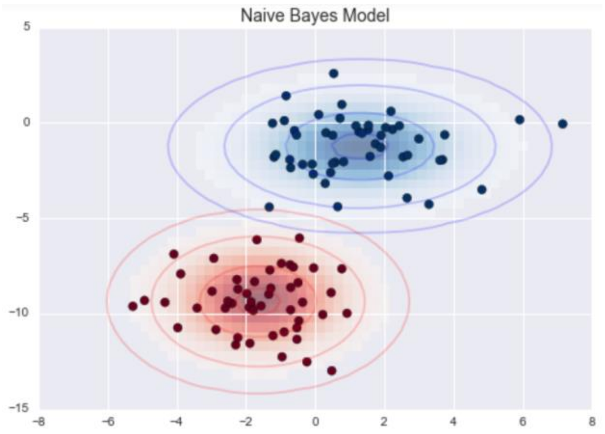


21

Gaussian Naïve-Bayes

Ellipses represent the Gaussian generative model for each label

Larger probability toward the center of the ellipses



22

Gaussian Naïve-Bayes

Pros

Easy to understand

Simple, efficient parameter estimation

Works well with high-dimensional data

Often useful as a baseline comparison against more sophisticated methods

Cons

Assumption that features are conditionally independent is naïve

Other classifiers often generalize better

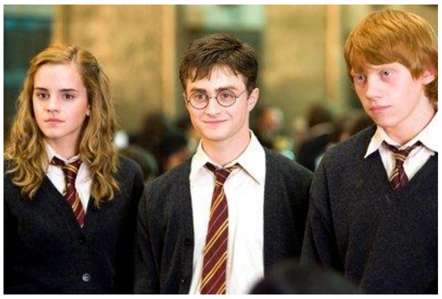
23

Decision Trees

24

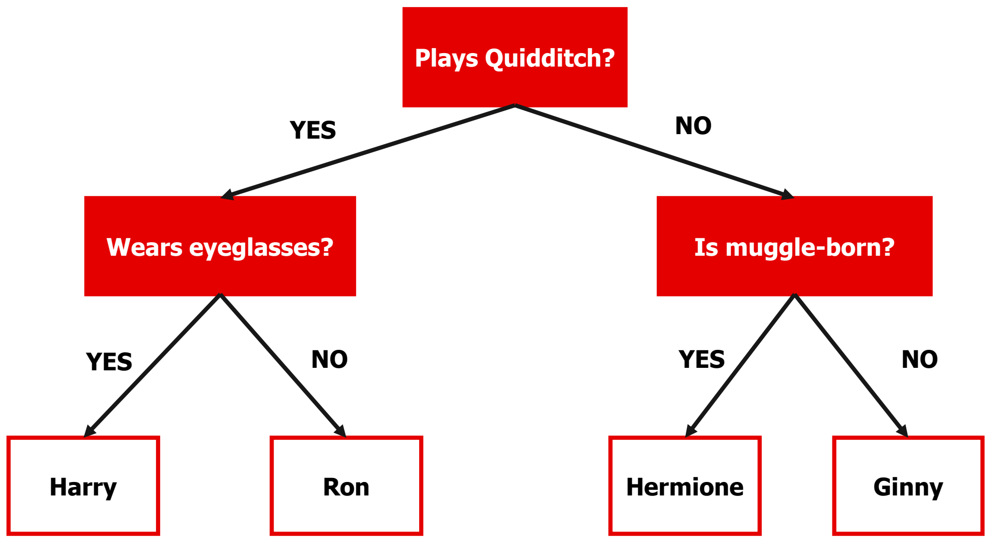
Decision Trees

Imagine you want to distinguish among the following four characters:

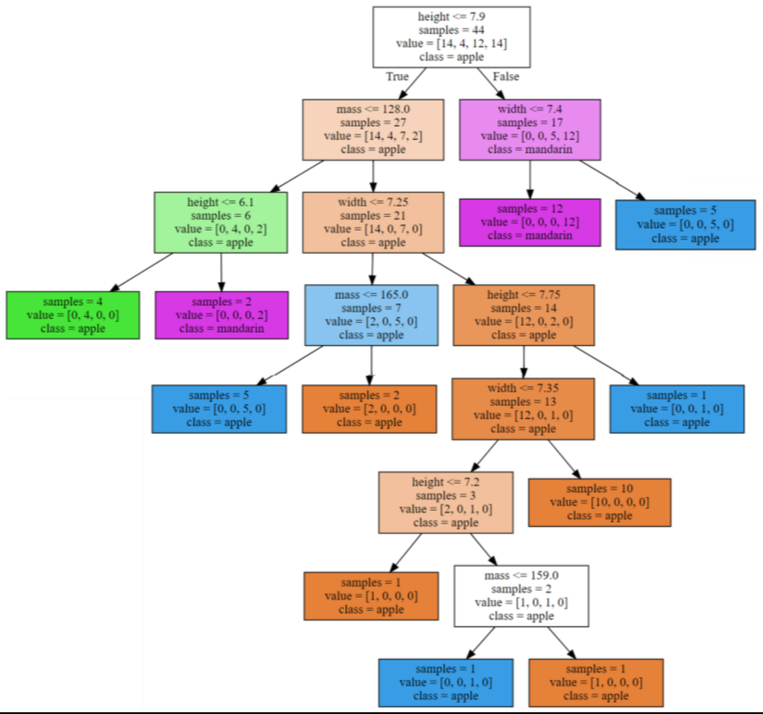


25

Decision Trees



26



27

Decision Trees

Decision trees learn a hierarchy of if/else questions, leading to a decision

Learning a decision tree means learning the sequence of if/else questions that gets us to the true answer most quickly

The algorithm searches over all possible questions and finds the one that is most informative about the target variable

With each question, a cutoff value is applied and the leaf is split

Is feature *i* larger than value *a*?

28

Decision Trees

Decision trees learn a hierarchy of if/else questions, leading to a decision

Learning a decision tree means learning the sequence of if/else questions that gets us to the true answer most quickly

The algorithm searches over all possible questions and finds the one that is most informative about the target variable

With each question, a cutoff value is applied and the leaf is split

Is feature *i* larger than value *a*?

29

Decision Trees

Pros

Easily visualized and interpreted

No feature normalization or scaling typically needed.

Work well with datasets using a mixture of feature types (continuous, categorical, binary)

Cons

Even after tuning, decision trees may still fail to generalize

Usually need an ensemble of trees for better generalization performance

30

Cross-Validation

31

Training and Testing

Ideally, use two different data sets (one for training and one for testing)

Training and test sets are assumed to have been sampled independently from an infinite population

Never test your algorithm on your training data

What if you have just one dataset?

- Percentage-split
- Cross-validation

32

Cross-validation

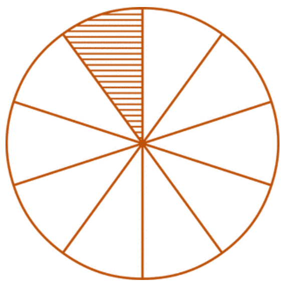
Splits the dataset into *k* equal-size folds

- Typically *k* = 10

Repeatedly **trains** your model with *k* – 1 folds and **test** the model with the remaining fold

Uses **all of your data** for **training and testing**

- Averages the accuracy rates



33

Training and Testing

Cross-validation usually reveals better, more reliable results than percentage-split, especially with small datasets

Which one to use?

If you have a lot of data, use percentage-split

If not, use cross-validation

34