# Text classification

Luka van der Plas

Centre for Digital Humanities, Utrecht University

# This lecture

- Classification basics
- Algorithms
- Evaluating classifiers
- Responsible classification

# Classification

# Classification

Classification is a form of **supervised machine learning**:

- Try to predict some output based on input
- Compare the output to a **ground truth**
- Optimise your algorithm so it best predicts the ground truth

In classification, we try to predict *labels* / *categories*

# Classification in text mining

Select which newspaper articles are about a specific topic

Detect the author of a text

Moderate hate speech in online spaces

Investigate what language characterises positive or negative book reviews

# Terminology

**Features**: information which is used to separate data into classes

**Prediction***:* the output of the model, which can be compared to the correct labels (our ground truth)

**Parameters***:* the data in the model; training = fitting parameters to the data

**Hyperparameters***:* the “settings” of the model; these are choices made by you

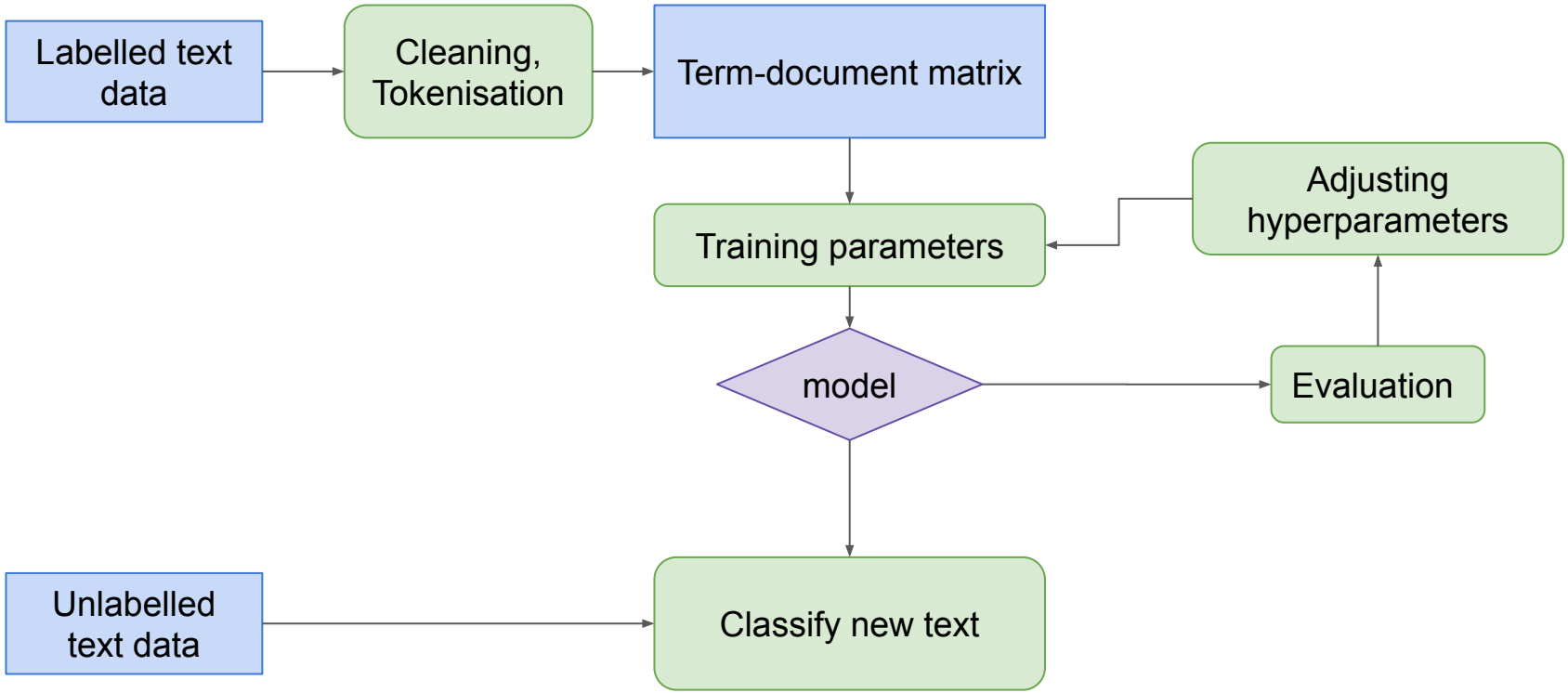
# Classification

**Binary** classification: two categories, true/false

**Multiclass** classification: multiple categories

# Text classification algorithms

# Classification workflow



# Today: bag-of-words classifiers

We will discuss algorithms that can work on a term-document matrix

Deep learning models iterate over sentences instead - we will see those later in the course

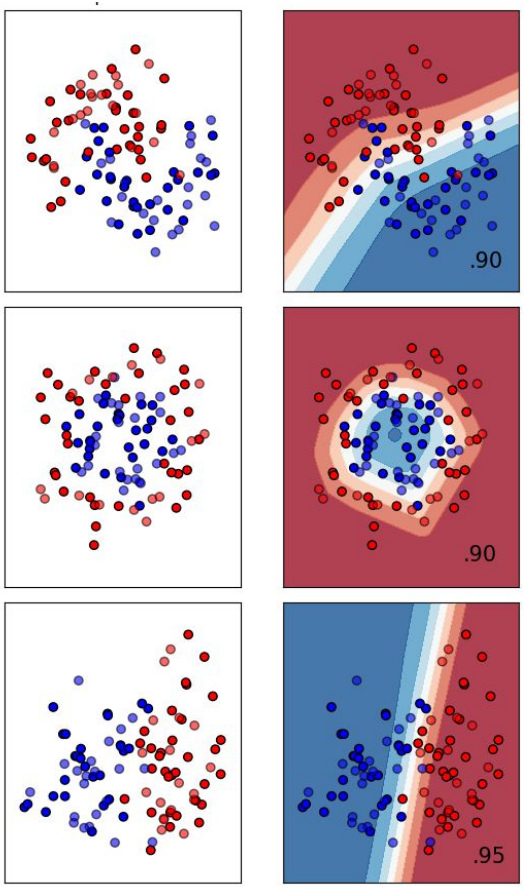
Many of the principles here apply for both cases

# Logistic regression

Assigns positive/negative weights to input features

- Start with random weights
- Iteratively adjust weights and re-evaluate to optimise accuracy

Weights are normalised to give a probability distribution as output



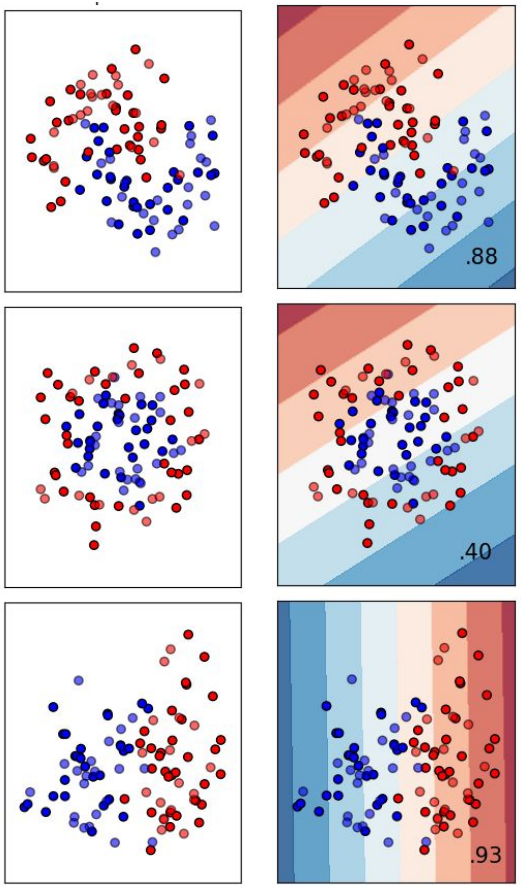
# Support vector machine

Input consists of many dimensions (features)

SVM tries to find a cross-section that provides a clean separation between the classes

Less “fickle” than some other methods 👍

Requires linear separation 👎



# K-nearest neighbour (KNN)

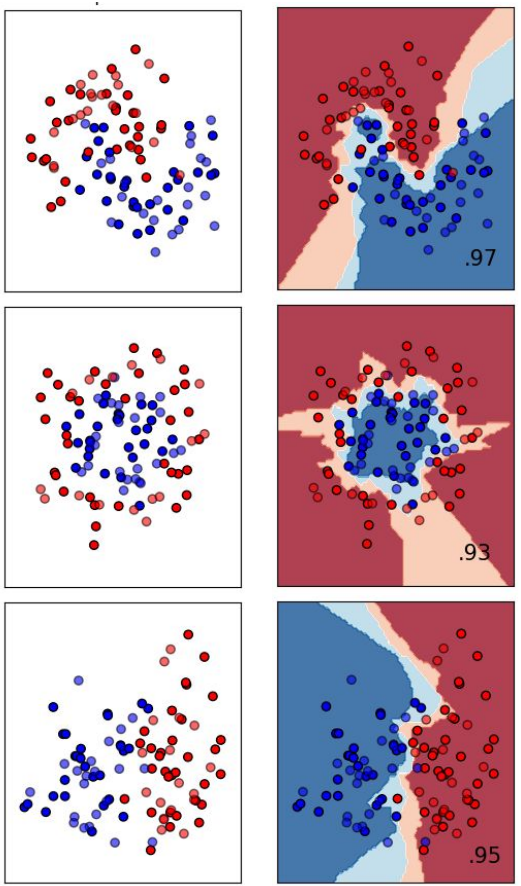
To classify a text, find its “nearest neighbour” in the training data and look up the label of that text

Proximity is based on the term-document matrix

Take the average of *k* nearest neighbours

No assumptions about linearity or independence 👍

For large data: quick to train, slow to run 👎



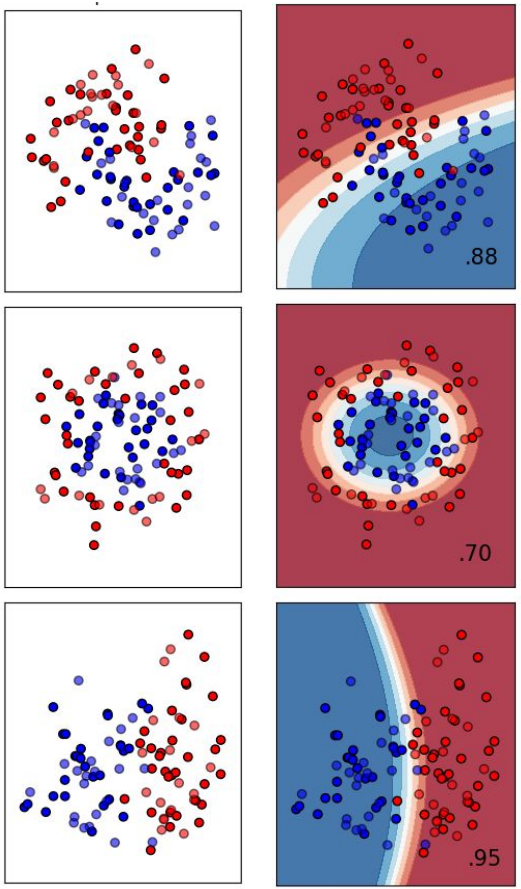
# Naive Bayes

Assumes that each feature (i.e. word) is independent

For each feature, estimate the probability distribution of the word given the label

Use this to estimate the likelihood of a label given the words in the text

Scales well to many input features 👍



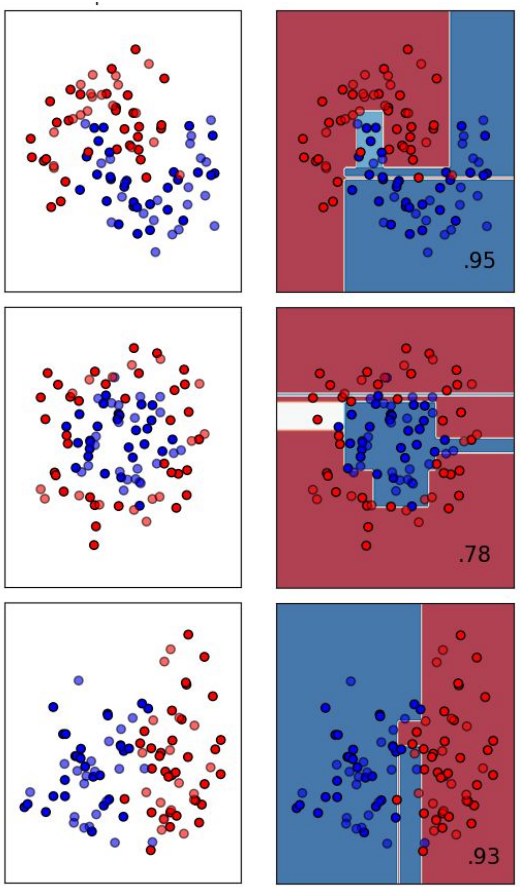
# Decision tree

Generate a decision tree: at each branch, pick the single feature that is most informative (i.e. gives the cleanest separation of the data)

Hyperparameter: *maximum depth* of the tree. More levels means higher granularity

Can capture complex relationships 👍

Sensitive to single words 👎



# Ensemble classifiers

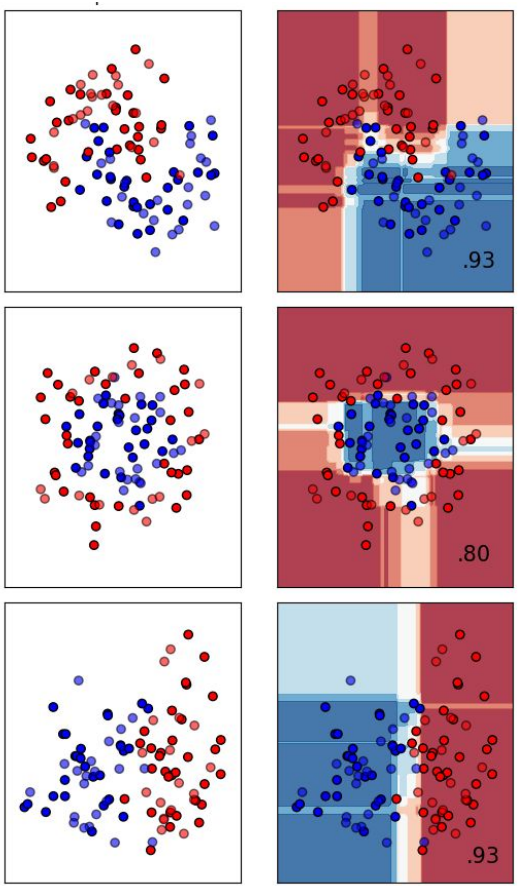
We can combine multiple classifiers

**Random forest classifier**

- fit multiple decision trees, average over predictions
- Less “volatile” than a single decision tree

**Voting classifier:**

- Fit multiple classifiers and let them “vote” on the result
- Classifiers may be of different classes!



# Selecting hyperparameters

A big part of training is selecting *hyperparameters*

This can be automated with a **grid search**: systematically go through your data

# Evaluating classifiers

# Evaluation metrics

**Loss:**

- value that is minimised during training.
- Usually includes a degree of “how far off” each prediction was.
- Highly dependent on implementation.

**Accuracy:** percentage of correct labels

Accuracy may be misleading!

# Evaluation metrics: looking deeper

For each label, we can look at:

**Recall:** of all positive cases, how many are labelled as positive by the model?

**Precision:** of all cases labelled as positive, how many are *actually* positive?

**F-score:** combines precision and recall. **F1** is most commonly used:

*(precision * recall) / (precision + recall)*

**Matthew's correlation coefficient:** correlation between true and predicted labels

# Evaluation metrics: multiclass classification

Evaluation is less straightforward with multiple classes

Precision/recall/F1 per class

But what if we want a single score?

- **Macro F1**: average of F1 score for each class
- **Micro F1***:* uses total number of true/false positives/negatives

# What does low performance mean?

Some explanations:

- Noisy data / missing features
- Not enough data
- Underfitting
- Overfitting

# Noisy data / missing features

A classifier can only learn from what you provide in the input: *garbage in, garbage out*

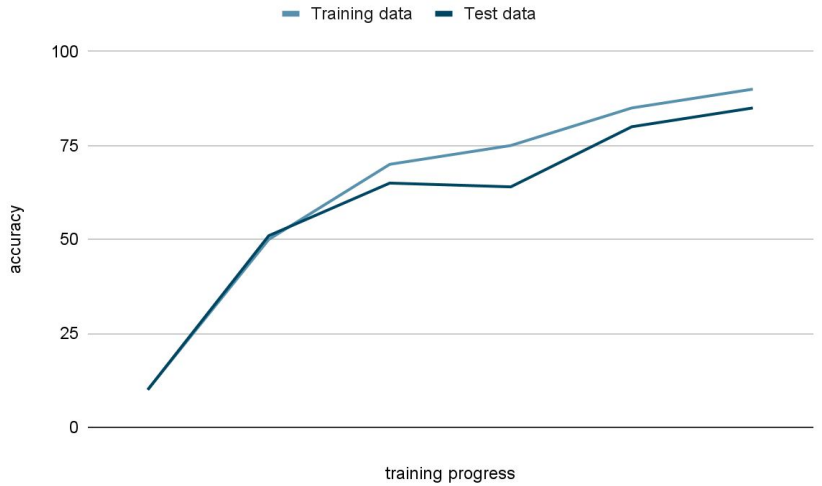
- Training labels have mistakes
- Training input has mistakes
- Training features lack essential information

In many cases, the information needed for perfect accuracy is simply not there

# Not enough data

Quality of the data is fine, but our classifier has not seen enough training data!

The model is still improving

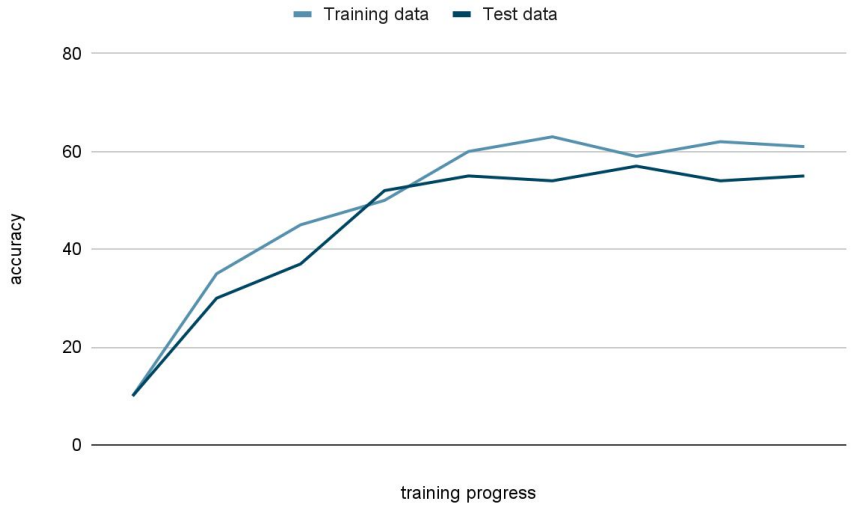


# Underfitting

A classifier is **underfitting** when it is not capturing enough complexity of the data

An underfitting is not improving with more data

A more complex model would give better performance



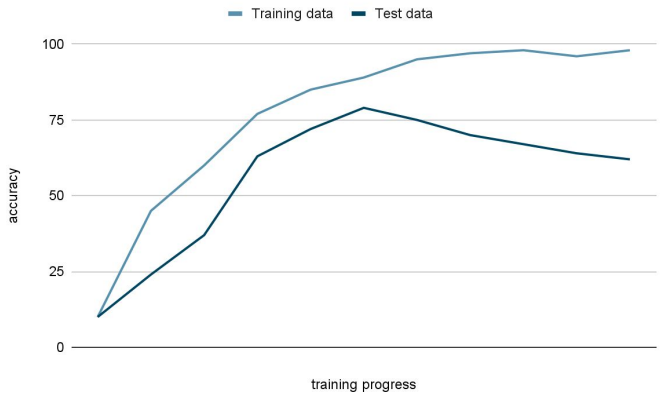
# Overfitting

A classifier is **overfitting** when it is matching the training data, rather than general trends

Performance on the training data is good, performance on new data not so much

Happens when

- The complexity of the model outweighs the amount of training data
- The complexity of the model outweighs the complexity of the problem



# Avoiding overfitting

Splitting data into train set / validation set / test set

- **Train set:** used to optimise the model
- **Validation set:** used to select the best model (tuning hyperparameters)
- **Test set:** used to evaluate the model

Notes:

- Validation set may be rotated (cross-validation)
- Validation/test distinction is less relevant in practicals

# Variable naming conventions

*X*: feature matrix

*y:* vector of labels

*X_train, y_train*: training data

*X_val, y_val:* validation data

*X_test, y_test*: test data

# Responsible classification

# External validity

Let’s say we are trying to predict thunderstorms:

- We have trained our model on data from June-August in 2010-2020
- We used a train/test split and found no significant overfitting

Great! Can we use our model to predict thunderstorms in November 2023?

# External validity

Even if you avoid overfitting, your model is still trained to a specific training set

It will make good predictions when the new data *is sufficiently similar to the training data*

For text:

- Applying a model trained on newspapers to classify social media messages
- Applying a model trained on American data to British data
- …

# Keep in mind

Text data is very rich

Your categorisation may obscure more meaningful differences along other dimensions

Dividing data into categories is often reductive

Classifiers are *always* inaccurate