Visual Analytics

Session 5: Image Classification with Scikit-Learn

Ross Deans Kristensen-McLachlan

rdkm@cas.au.dk

Course outline

- 1. Introducing Visual Analytics
- 2. Basic image processing
- 3. More image processing
- 4. Convolutional kernels
- **5. Image classification 1**
- 6. Image classification 2
- 7. From shallow to deep learning
- 8. Convolutional neural networks
- 9. Pretrained CNNs and transfer learning
- 10. Image embeddings
- 11. Project presentations
- 12. Text-to-image models
- 13. Project development

Plan for today

- Catch-up

- 1. What is classification?
  - Thinking more about the task
  - Thinking about classifying images

- Coding session
  - Image classification with Logistic Regression
  - Python scripting
  - Assignment work

What is classification?

- Classification is a task which falls under the umbrella of *machine learning*

- Machine learning is the practice of using algorithms to learn patterns in data
  - Data agnostic: can be applied to text, images, audio data, spreadsheets of customer consumption, etc

- The patterns learned by the algorithm constitute a *model* which can be used to study new data unseen by the model

What is classification?

- Machine learning approaches tend to fall under two broad categories
- *Supervised*: The algorithm is presented with data and each data point has some kind of (usually human) label from a predefined list of labels. It creates a model which maps data to labels
- *Unsupervised*: The algorithm is fed raw data without labels and a set of parameters. It is then left to find ’structure’ in the data

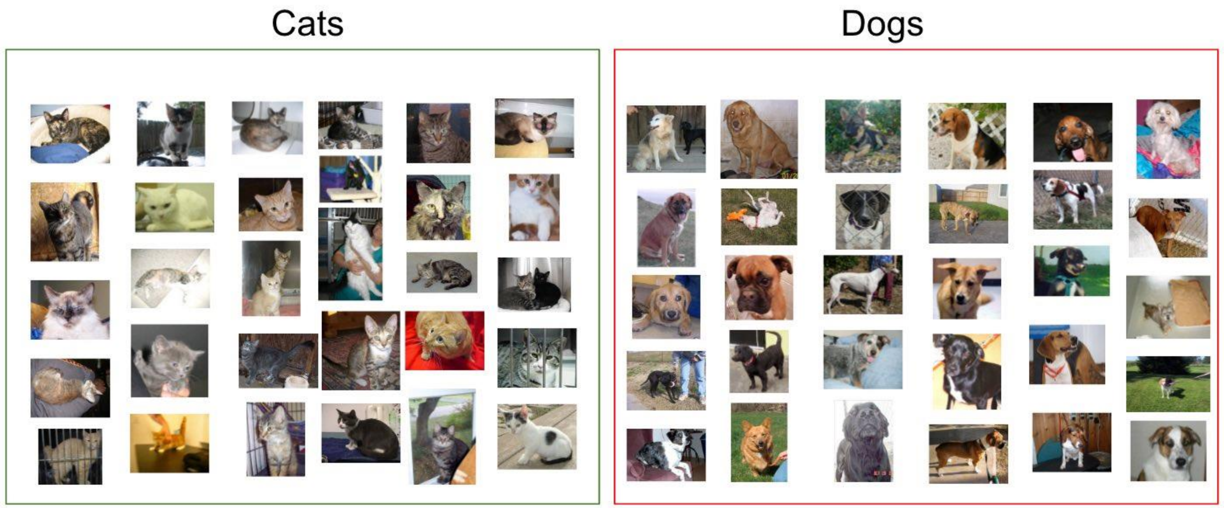
What is classification?

- On these definitions, classification is a kind of supervised machine learning problem

- Given some set of data which has been pre-labelled into different classes by a human, can we learn a model which can *predict* the class of some unseen data?

Data and labels

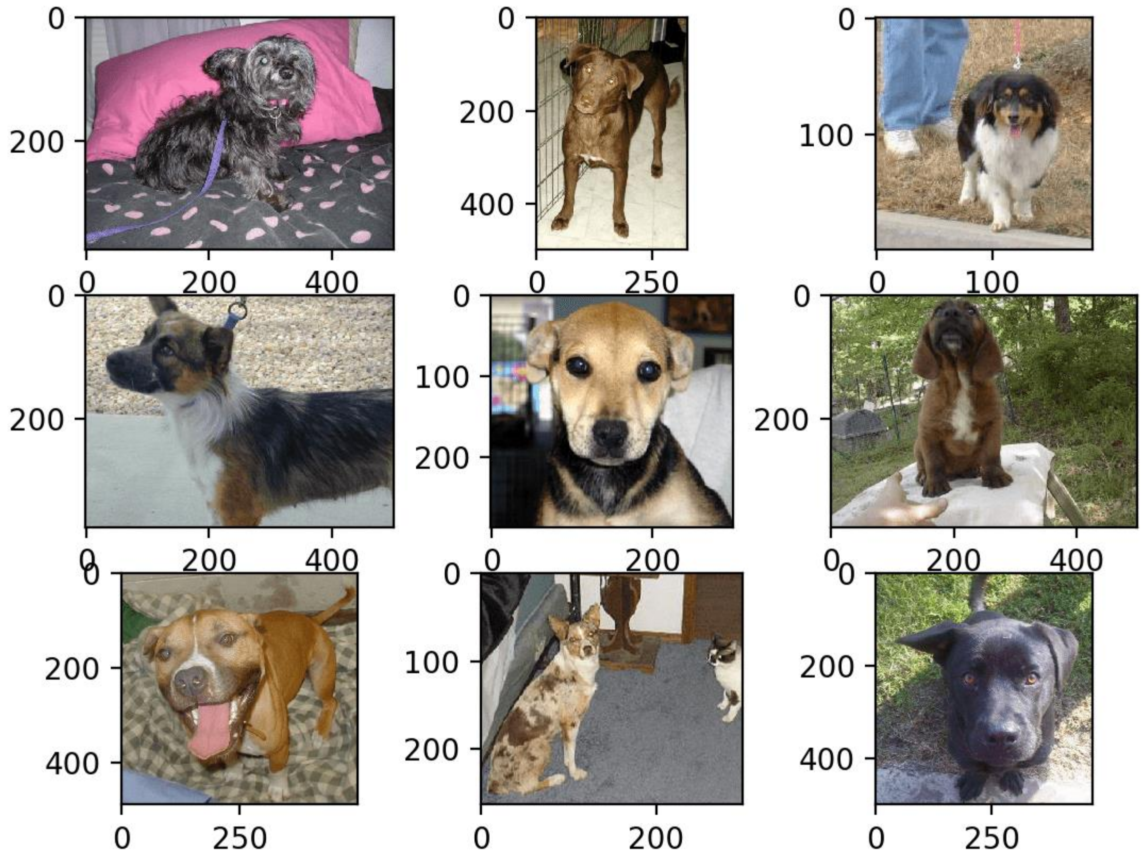
- Labels are created by human annotators who assign one or more labels to each data point
  - Labels can be binary or multiclass
  - Labels need to be drawn from a set of clearly defined options



Sample of cats & dogs images from Kaggle Dataset

Data and labels

- Look at these dogs 👉
- What do they have in common?
- How do *you* classify them as dogs?
- What other classification labels might you want to give them?



Data and labels

- Generally speaking, when we perform machine learning tasks, we do not want to take the full, raw data

- Raw data has too much noise, too much unnecessary information

- Instead, we want to train our model on specific *features* which appear in the data

- This pre-processing step is known as *feature extraction* or feature engineering

Some questions

- What kind of things do you think we use to perform image classification? What features of the image?
- Can you think of other topics which can be framed as a classification problem using computer vision?
- Relatedly, can you see any applications of this in your own discipline?

Break

Some answers

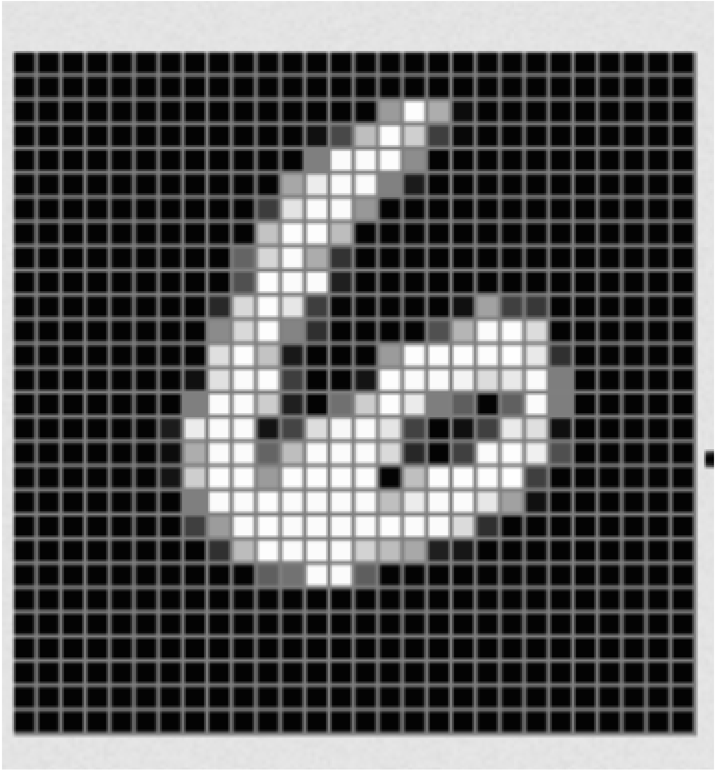
- As with network analysis, many problems can be framed as or reduced to *classification problems*

- Visual: object identification; handwriting recognition; etc

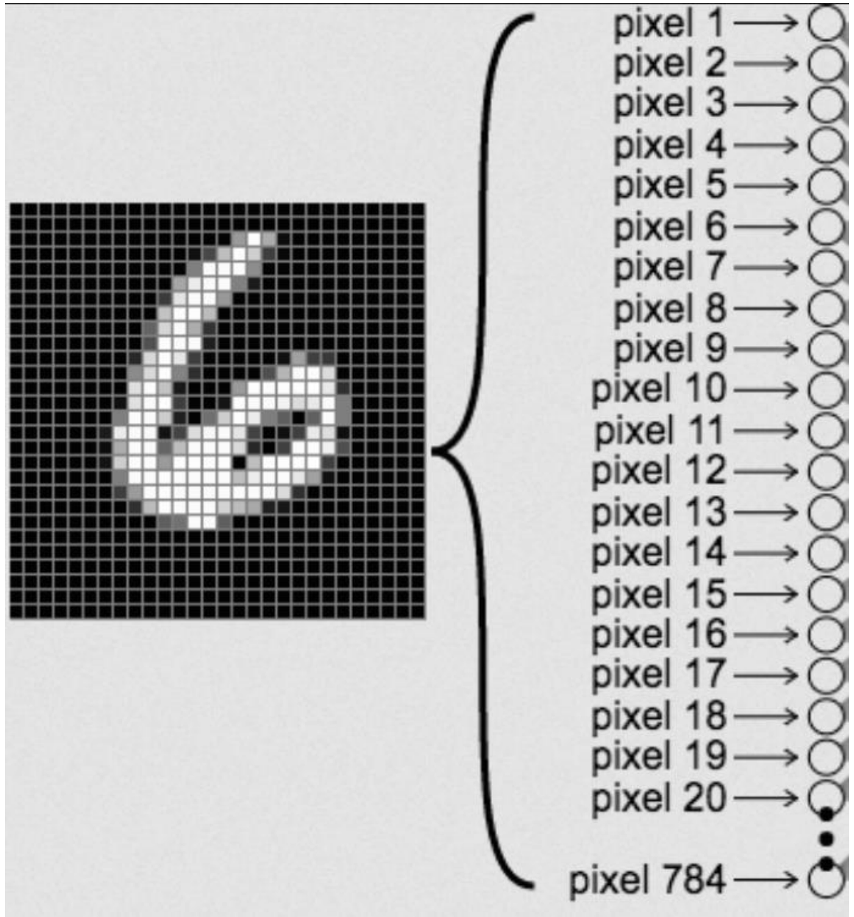
- Features: pixel intensities; contours; masks; all things we’ve covered
    - *Feature engineering*

- Machine learning give us a flexible way of approach a wide range of problems from similar perspectives

Feature engineering



Feature engineering



Training and testing

- Imagine we have a labelled dataset of images, with one label per image

- We've performed some kind feature engineering on the data to reduce noise and increase the strength of the signal

- Now what?

Training and testing

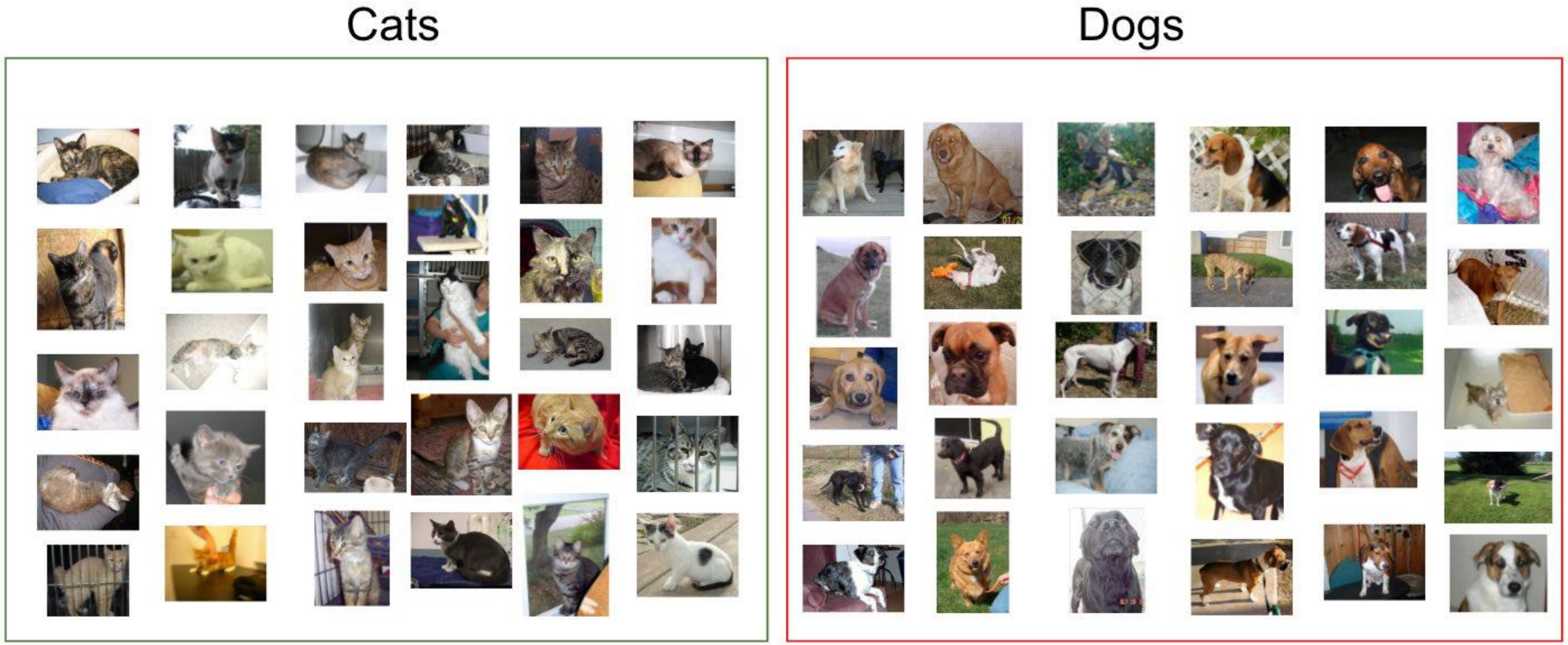
- Assuming the dataset is sufficiently large, we split it into two subsets - a *training* set and a *test* set
  - Usually split something like 80%/20% or 75%/25% - i.e. 80% training, 20% test

- The *training* set is used to learn the model which maps features to labels

- The *test* set is used to estimate the performance on data not used to train the model and is hence *unseen* by the model

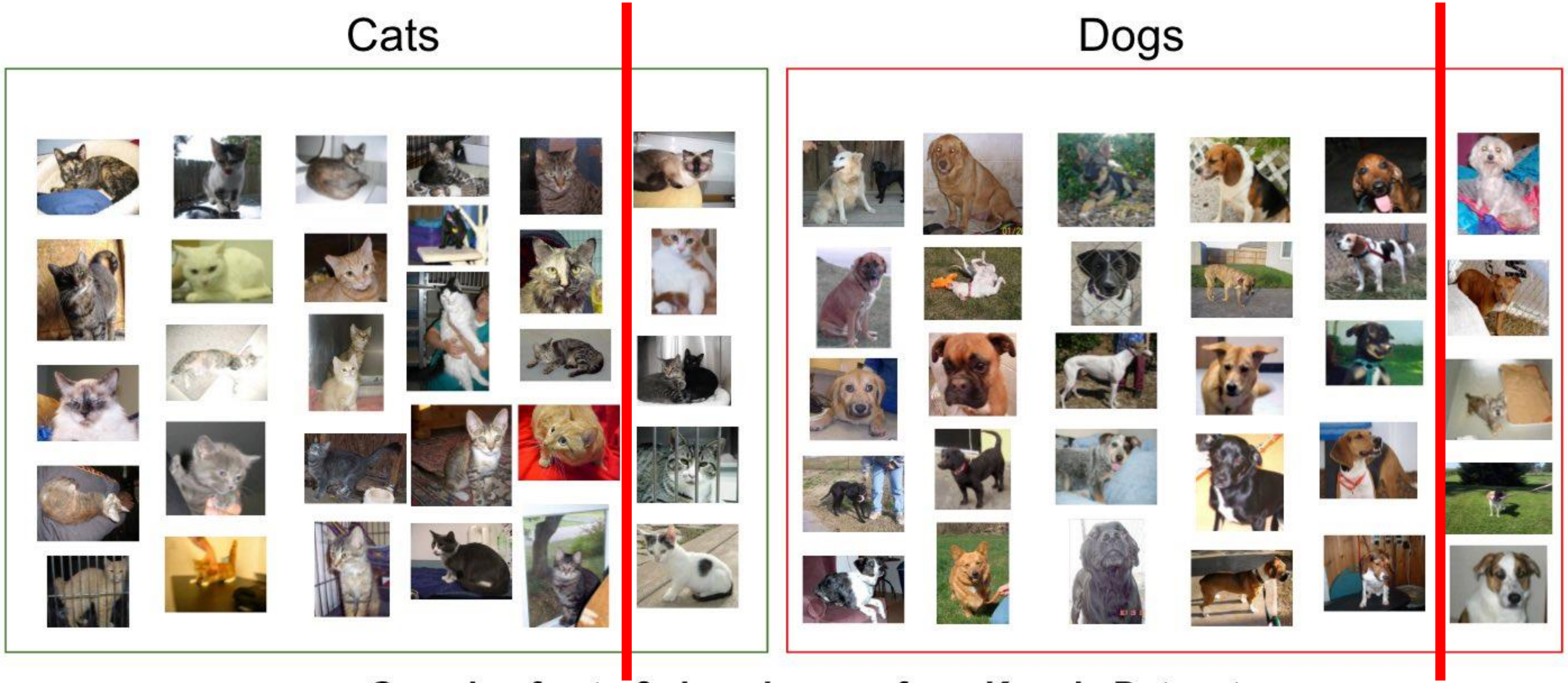
- If a model performs roughly as well on the test data, this suggests that it might generalize well

Training and testing



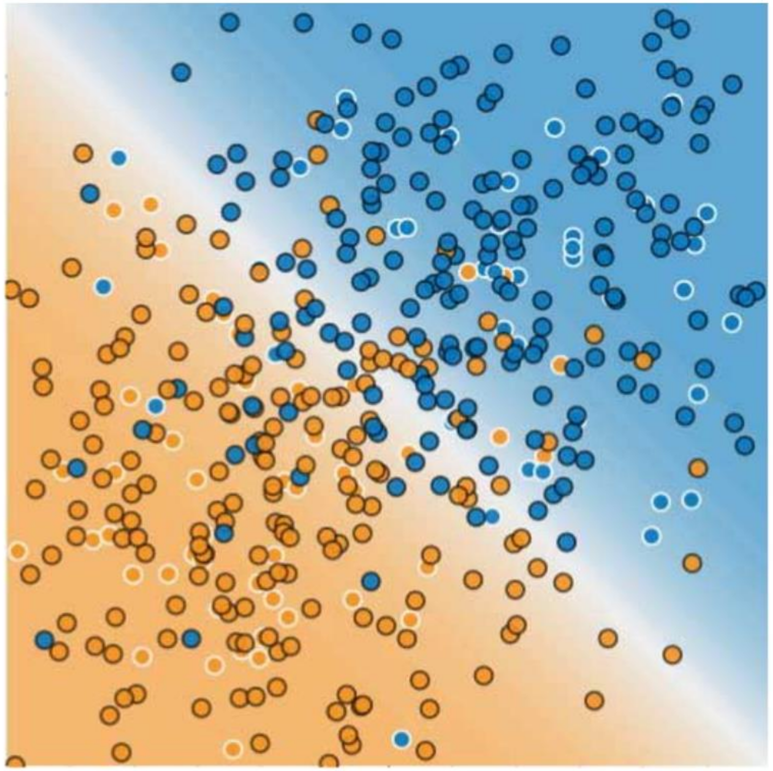
Sample of cats & dogs images from Kaggle Dataset

Training and testing

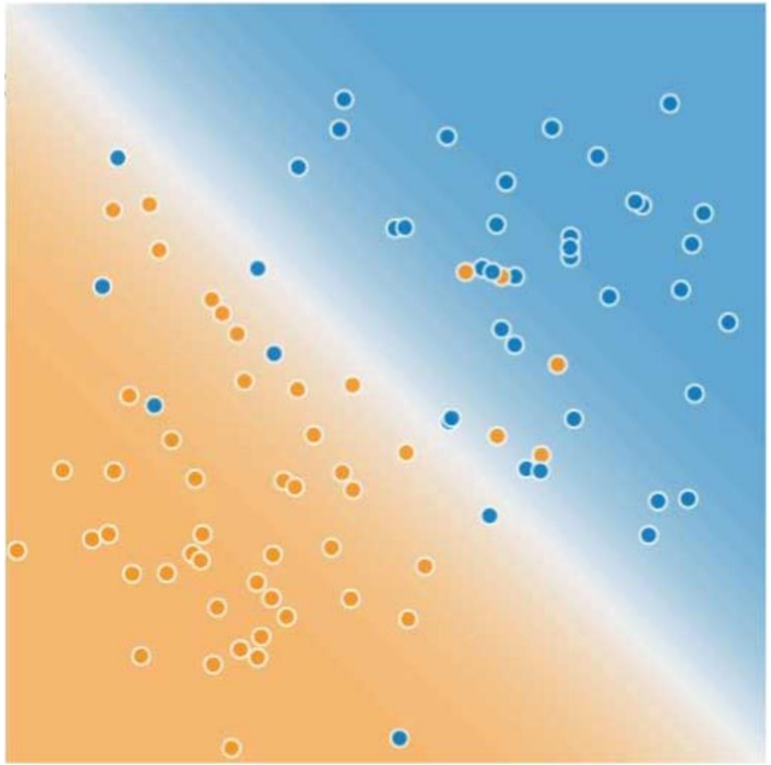


Sample of cats & dogs images from Kaggle Dataset

Training and testing



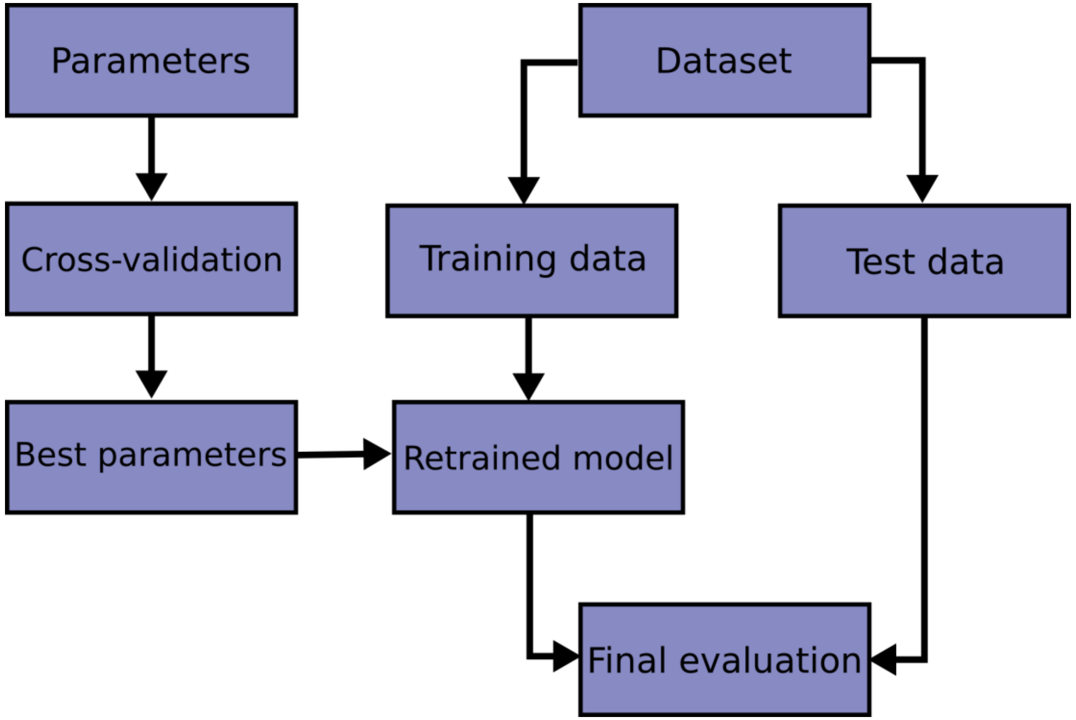
Training Data



Test Data

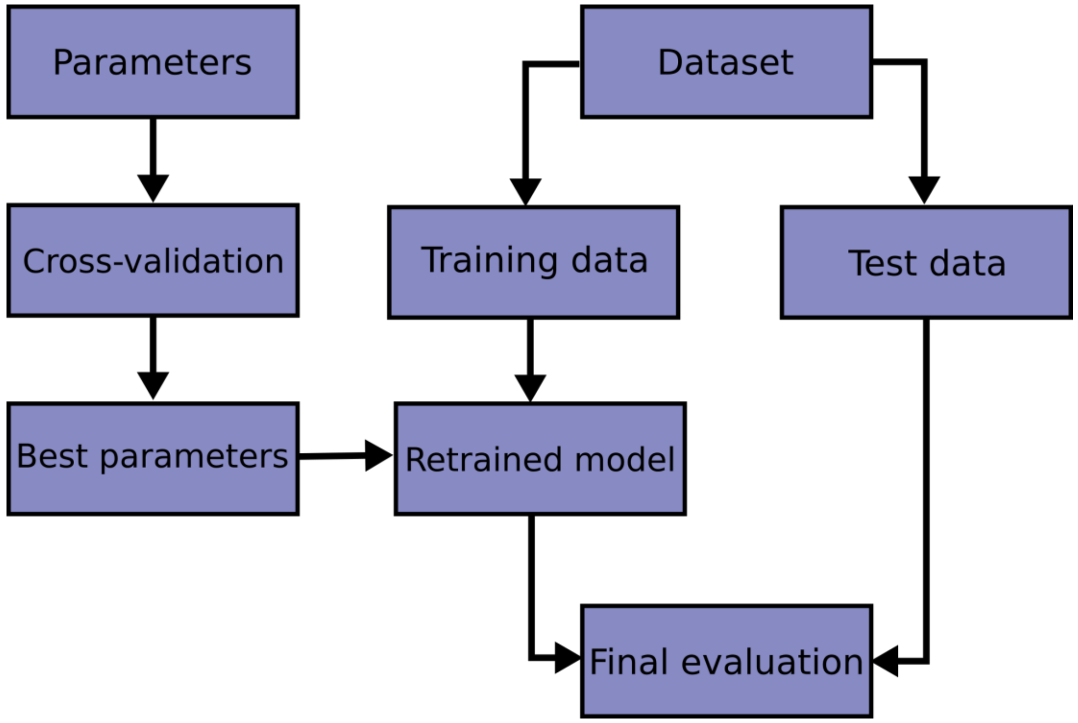
How do we train a model?

- In classical machine learning terms, training a model is comparatively simple
- The whole pipeline is outlined schematically here
- We have a labelled dataset of features which is split into training and testing sets
- The training data is fed into a classification algorithm with some pre-defined parameters
- The trained model is then evaluated on the testing data
- This is often performed iteratively until an optimal model is achieved



How do we train a model?

- Defining and optimizing model parameters can be partially automated by training a number of different models and choosing the 'best'
- So in practice, the main bottleneck is at the start of the pipeline
- If they don't already exist, creating labelled datasets can time-consuming
- Similarly, feature engineering can be time consuming
- Especially in the field of computer vision, some feature engineering does not generalise particularly well - think of the recent assignment!



Evaluating a model

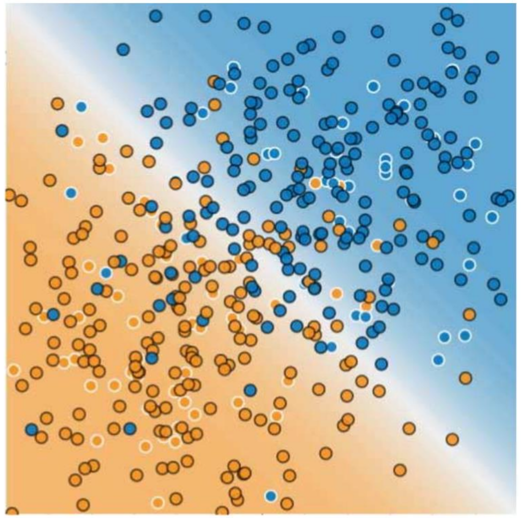
- So, we had a labelled dataset of images, with one label per image
- We performed some kind feature engineering on the data to reduce noise and increase the strength of the signal
- We used these features to train a model on the data, based on some chosen algorithm with pre-defined parameters
- Now what?

Evaluating a model

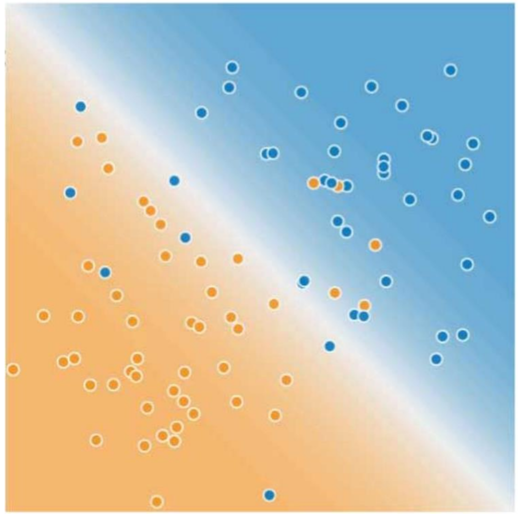
- We saw earlier that we can visually evaluate a hypothetical model and saw that it seemed to perform equally well on training and test data

- What we want, though, is a way of quantifying how well a model performs

- Thankfully, we can do that!



Training Data



Test Data

Evaluating a model

		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

A confusion matrix

Evaluating a model

- True positive (TP)
  A test says you have coronavirus and you actually have it
- True negative (TN)
  A test says you do not have coronavirus and you actually do not
- False positive (FP, Type 1 error)
  A test says you have coronavirus but you actually do not have it
- False negative (FN, Type 2 error)
  A test says you do not have coronavirus but you actually do have it

Predicted Class

		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Evaluating a model

- Sensitivity (True positive rate, recall)
  Proportion of the positive class correctly classified - i.e number of sick people correctly identified

- Specificity (True negative rate)
  Proportion of the negative class correctly classified - i.e. number of healthy people who were correctly identified

- Precision
  Patients correctly identified having COVID out of all the patients actually having it - ie. ratio of true positives to all positives

- Accuracy
  Ratio of correct classifications, relative to whole dataset

		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	**Negative Predictive Value** $\frac{TN}{(TN + FN)}$	**Accuracy** $\frac{TP + TN}{(TP + TN + FP + FN)}$

Evaluating a model

- Very often with classification problems, there is a trade-off between *precision* and *recall*
  - Increasing sensitivity model reduces precision; a more precise model has lower recall

- We can get around this by calculating F1 score, defined as the harmonic mean of precision and recall

$$F_1\text{–score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

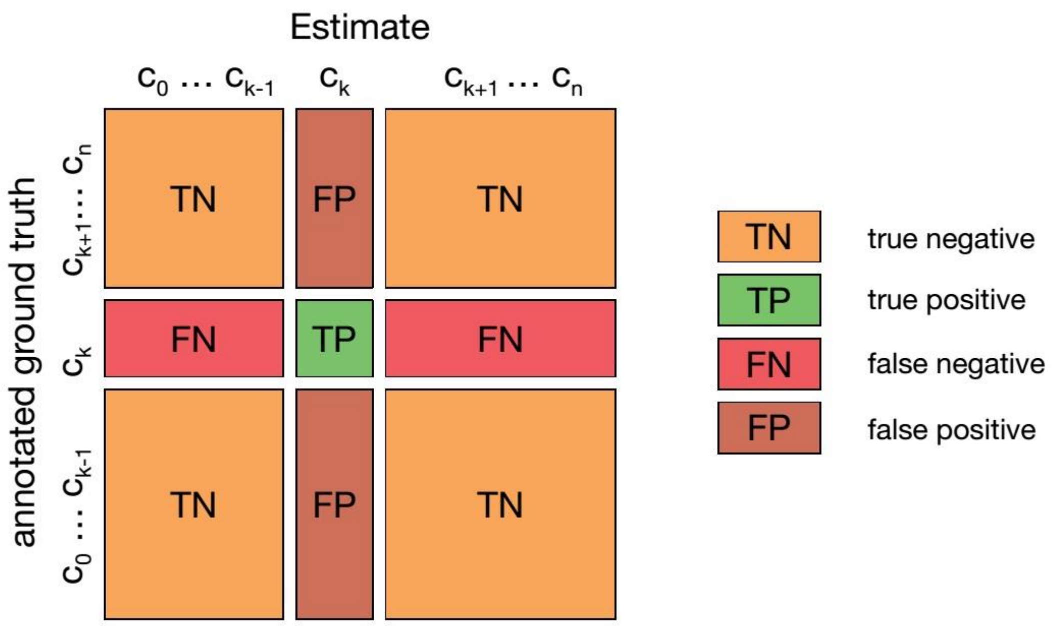
Evaluating a model

- In some cases, we will have multiple different classes, not just a binary classification system

- In these cases, we can calculate the exact same value using a similar confusion matrix

- Macro-F1
  Calculates F1-score for each class individually and then calculates an unweighted mean

- Weighted-F1
  Calculates F1-score for each class individually then calculates a weighted mean based on how often each class appears in the data



Evaluating a model

	precision	recall	f1-score	support
0	0.96	0.98	0.97	244
1	0.91	0.97	0.94	287
2	0.89	0.90	0.89	235
3	0.89	0.88	0.89	281
4	0.90	0.94	0.92	213
5	0.88	0.85	0.87	215
6	0.95	0.92	0.94	225
7	0.92	0.91	0.92	257
8	0.84	0.83	0.84	253
9	0.90	0.87	0.89	290
accuracy			0.91	2500
macro avg	0.91	0.91	0.91	2500
weighted avg	0.91	0.91	0.90	2500

Summary

- Machine learning is the practice of using algorithms to learn from data
- The data has usually undergone some pre-processing and feature engineering to reduce the amount of noise
- Classification is a kind of supervised machine learning task, where the algorithm learns a model which maps from features in the data to labels
- Models can be evaluated statistically and can be used to classify unseen documents
- Many problems can be reduced to some kind of binary or multiclass classification task

Break

And head over to UCloud