Machine Learning for Developers

Mai Nguyen

https://github.com/mjnguyennz/ml_workshop_kiwiruby
mjnguyen@gmail.com
@mjnguyen on RubyNZ Slack

About me

Ruby/Rails development 2006 - 2010 in Washington DC

Took a break to travel, study, and work in Winemaking

Worked in Australia, France, California, New Zealand

Missed learning new things everyday

2016 - Moved to Wellington, Senior Developer at Loyalty NZ

What will happen in this workshop

Learn what machine learning is and what it entails

Understand the Machine Learning workflow

Understand the importance of data

Exercises to reinforce concepts, and try ML tools and libraries in Ruby

Emphasise practical application, not algorithm details, statistics, linear algebra, etc.

Workshop materials

https://github.com/mjnguyennz/ml_workshop_kiwiruby

Outline

Lecture about the Machine Learning and the basic workflow and data preparation
Data exercises
Evaluating models

Tea Break

BigML demonstration, and try it for yourself!
PyCall examples, and try it out!
Wrap up

Typical Dev tries out Machine Learning

Read some blog post about a gem/library that does ML

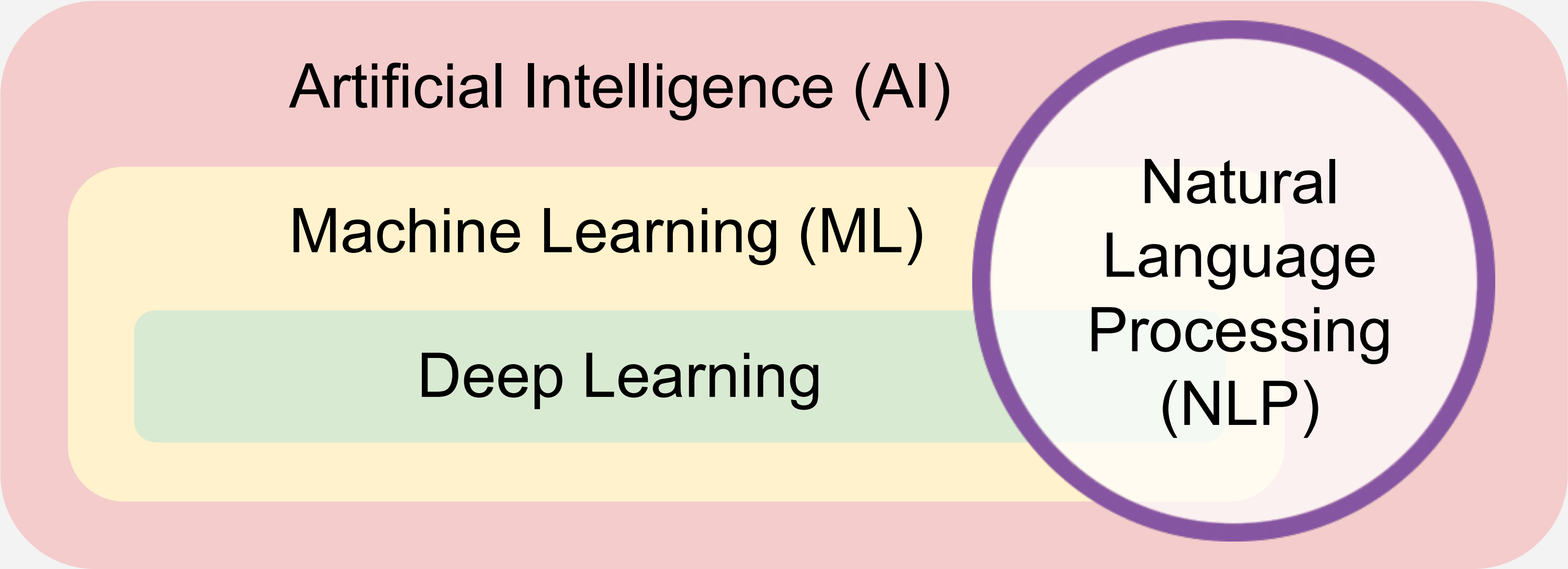
Read the README with very trivial example. Easy!

Try to out with more complicated data, and get unsatisfactory results.

Declare it not useful and that Machine Learning is not for your project

Any tool is not useful until you know how to use it.

What is Machine Learning?



What is Machine Learning?

Does not rely on coded rules

Creates its own model based on training data

Can be supervised or unsupervised

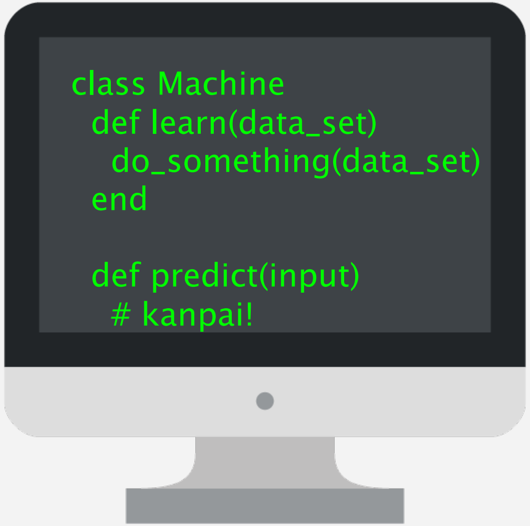
- Supervised - where data examples have known outputs to train upon
- Unsupervised - no outputs defined, finds hidden structure in unlabeled data

Many types of algorithms for different problems

What can machine learning do for me?

Infinite Combination of Inputs

∞



Finite Output

(Boolean, number, etc)

Example: Can it fly?

Animal	Has Wings?	Has Feathers?	Height/Length	Weight	Can it fly?
Emperor Penguin	Y	Y	100cm	30kg	N
Kea	Y	Y	40cm	1kg	Y
Honeybee	Y	N	1cm	0.3g	Y
Grasshopper	Y	N	2cm	0.5g	Y
Chicken	Y	Y	45cm	3kg	N
Kiwi 	Y	Y	25cm	1.3kg	N

Usecases for machine learning

Classification - spam filtering, sentiment, fraud detection, ad targeting & personalisation, medical diagnosis

Recommendations - products, job recruiting, dating, content

Predictions - stock-market, demand forecasting, weather, sports results, asset management

Imputation - infer missing values of input to make complete datasets

When you wouldn’t use machine learning

Rules are known, well defined and finite

High accuracy

Data is unavailable / difficult to obtain

Features of Machine Learning

Accuracy improves as you collect more data

Can be automated - learn automatically as answers are validated (online learning)

Can be fast

Customisable - built from your own data

Scalable

Machine learning challenges

Mistakes/fallacies in training data can be hard to spot

100% accuracy is near impossible

Testing is difficult - edge cases

Future data may not resemble past data

Biases in your training data can be magnified

Determining successful outcome

"Correlation doesn't equal causation"

Why aren't more Rubyists using it?

I like Ruby, I don't want to write Python.

I don't have time to learn these algorithms.

I am not a data scientist.

Ruby resources

Algorithms and tools ported to Ruby
http://sciruby.com/

PyCall - call Python from Ruby

Machine Learning gems:
https://github.com/arbox/machine-learning-with-ruby

Natural Language Processing gems:
https://github.com/arbox/nlp-with-ruby

Popular ML APIs and services

BigML

Amazon Machine Learning APIs (only in N. Va and Ireland)

Microsoft Azure Machine Learning APIs (not all regions)

Google Cloud Machine Learning Engine - TensorFlow
https://github.com/somaticio/tensorflow.rb

Popular NLP APIs and services

Wit.ai

Microsoft Language Understanding Intelligent Service (LUIS) API

MonkeyLearn

Google Cloud Speech and Natural Language API, api.ai

Amazon Alexa (N. Va and Oregon) and Lex (N. Va only)

IBM Watson API

Let's focus the scope of this workshop

There are so many types of machine learning problems. With our limited time, we will focus on two of the most popular:

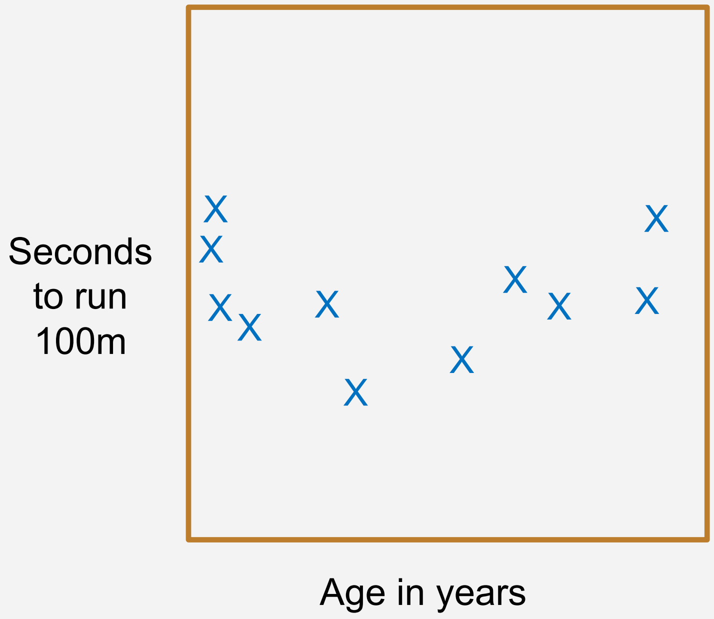
Regression – Supervised learning, predicting numerical (continuous) values

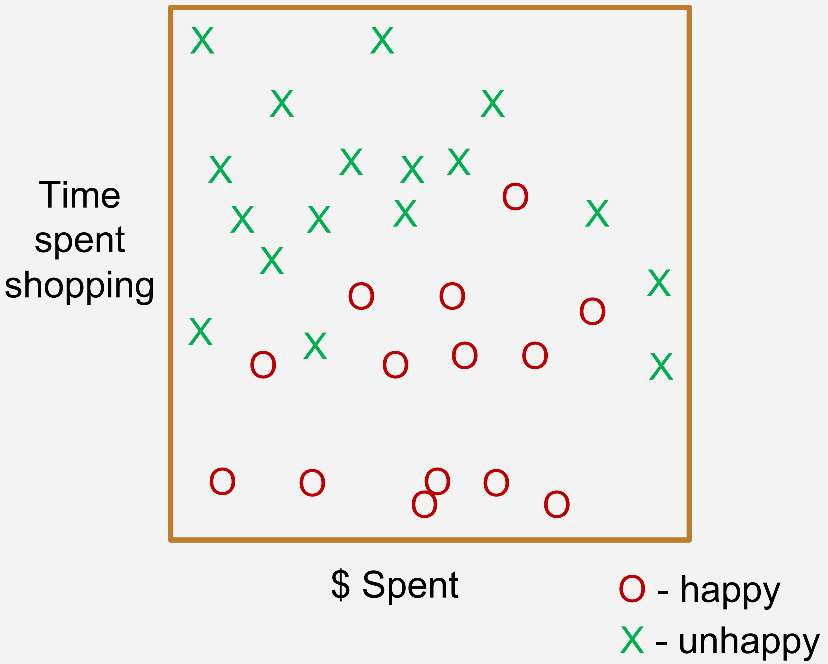
Find the line/curve that best fits the data

Logistic Regression / Classification – Supervised learning, predicting classification categories/labels (discrete values)

Fine the line/curve that best separates the data by the category

Example: regression and classification





Definitions

Example or **instance** – row of data i.e. row in your csv

Feature – a column in that data

Feature engineering - transforming inputs into suitably formatted features

Target variable or **objective field** – the value you are seeking/predicting

Model – the pattern/decision making that ML has derived from data for predicting

First step - Ask yourself this:

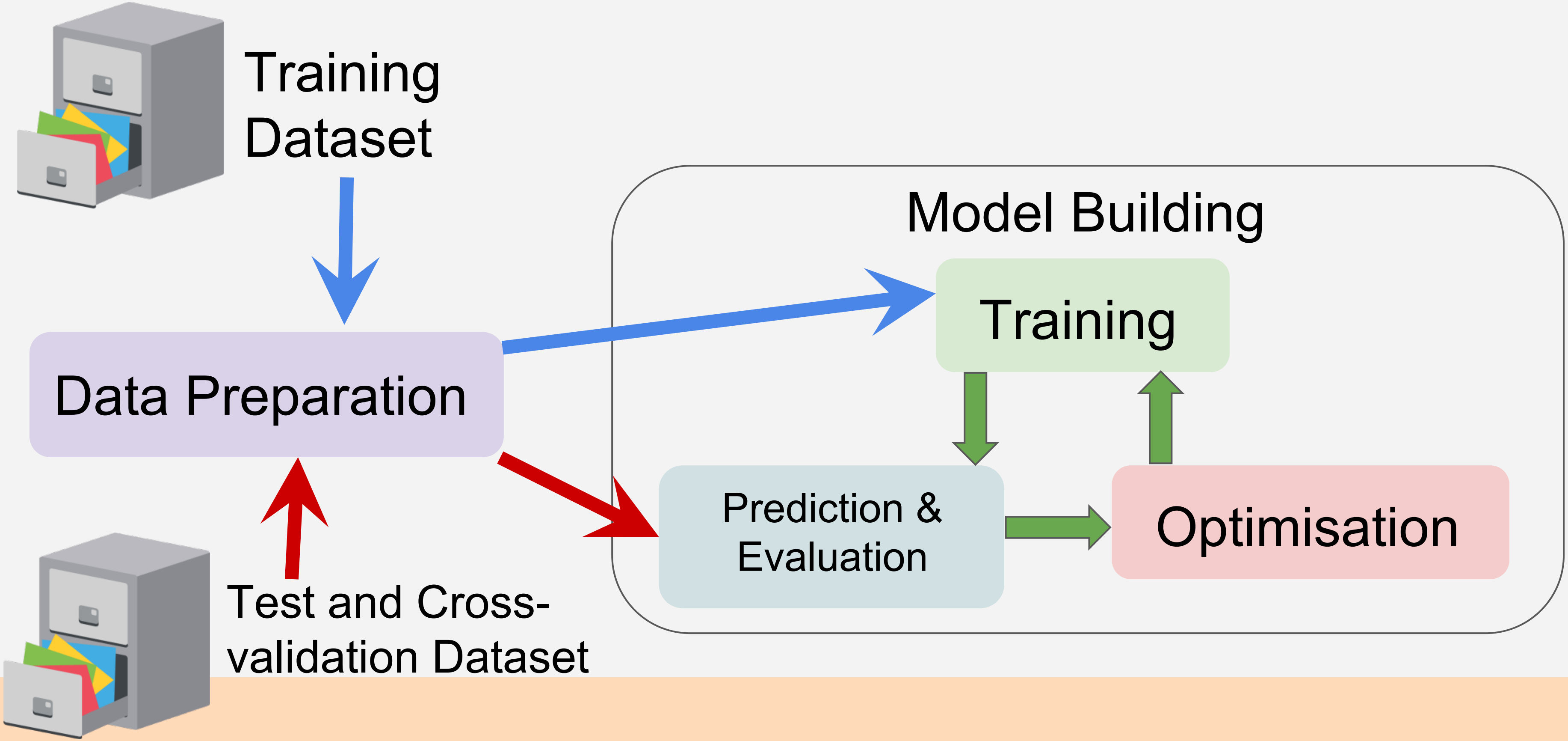
What is the question you want to answer?

- Well-defined target

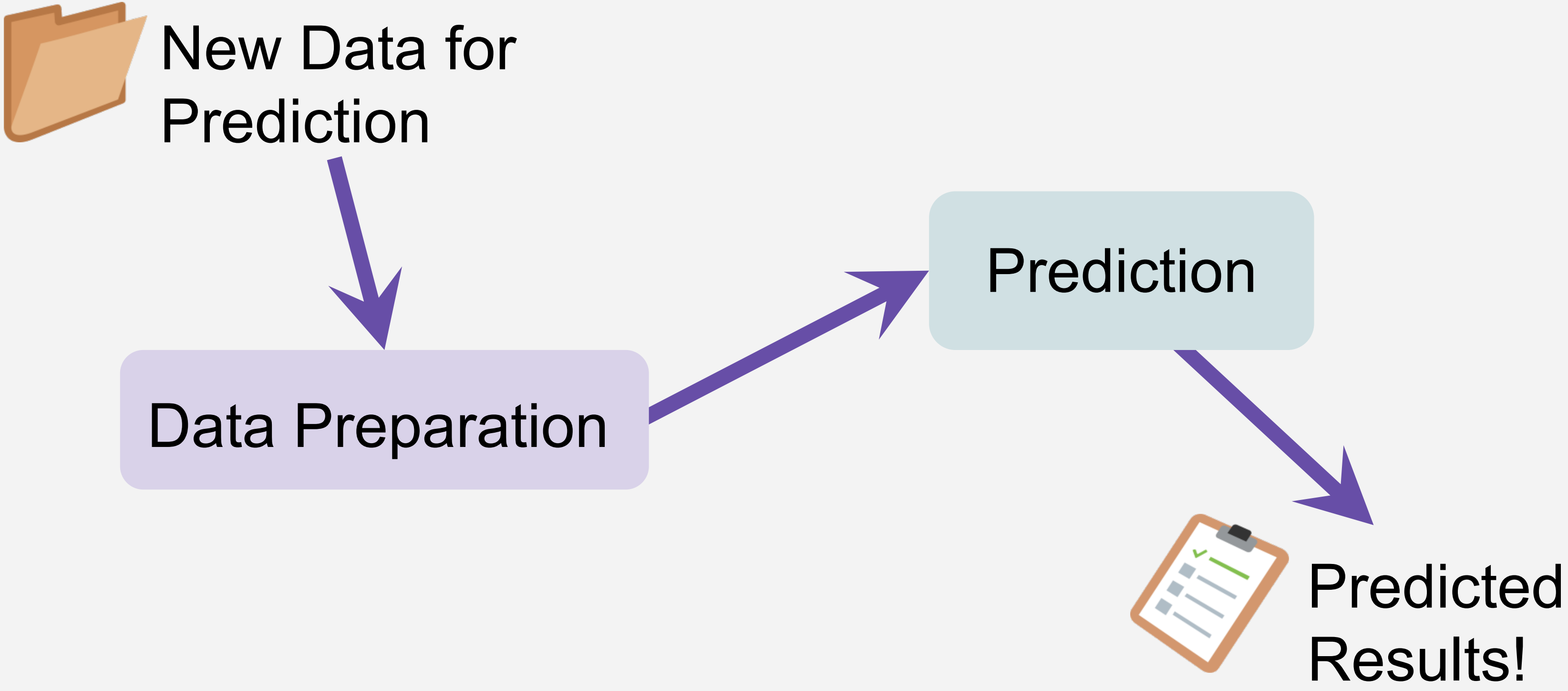
What data do you have access to?

- Custom data
- Free public data - UC Irvine Machine Learning Repository, Kaggle.com, etc

The Machine Learning Modeling Process



The Machine Learning Prediction Process



Splitting the data

Standard Practice

- Training Data – used to build your model
- Test Data – used to assess performance of your model

Better Practice

- Training Data – used to build your model
- Cross-validation Data – used to find the best tuning parameters
- Test Data – used to measure accuracy performance of your final, tuned model

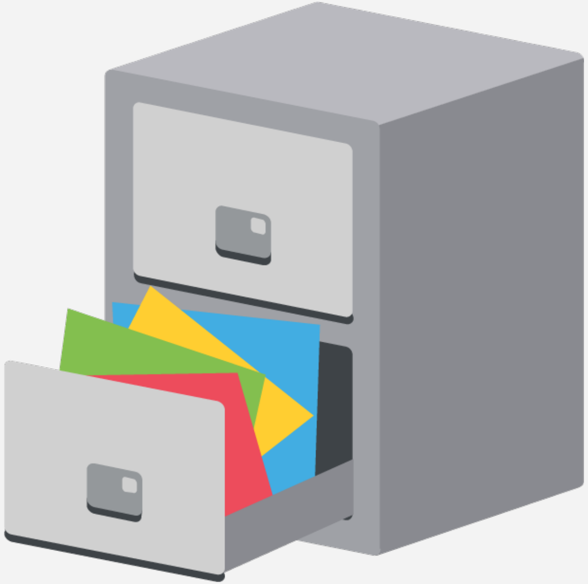
What makes good quality data?

Representative of future data

Complete

Relevant features - minimal noise

Lots of it - the more the better!



Types of data

Numerical

Categorical

Boolean

Text

Date/Time

Images/Video/Sound

Making your data ML ready

Feature engineering: transforming inputs into predictive features

“Data munging”

Handling missing data

Feature normalisation

Use only the inputs that are relevant*

Feature Engineering

Can boost the accuracy and computational efficiency of your ML models.

- Computational transformations
- Data joins with another table, external data, etc
- Turn variable length text into fixed length features
- Images - represent characteristics of the image with numeric features

Data munging

Date and time pre-processing – turn into categories

- DOB – age range categories
- Day of week or general time of day may be useful

Location – lat/long/addresses may be too specific

Categorical data – transform into Boolean feature per category (required for many algorithms, but not all)

Standardise units

Example

Example #	DOB	Marital Status	Income
1	01/04/2000	Married	0
2	05/12/1992	Single	150,000
3	22/06/1976	null	40,000
4	08/08/1956	Divorced	80,000

Example #	< 20	20-30	> 30	Single	Income
1	1	0	0	0	0
2	0	1	0	1	150,000
3	0	0	1	null	40,000
4	0	0	1	1	80,000

Handling missing data

When the fact that it’s missing can carry meaningful information

Numerical data – assign a number at end of the spectrum, like -1

Categorical data – assign a new category like “None”, “Missing”, etc.

Handling missing data (cont.)

When the unavailability of the information is not meaningful

- If missing data is small, easiest to drop the data
- If you can't drop the data, fill in the missing data
  - Impute with adjacent data
  - Mean/median
  - Machine learning to make an educated guess

Example

Example #	DOB	Marital Status	Income
1	01/04/2000	Married	0
2	05/12/1992	Single	150,000
3	22/06/1976	null	40,000
4	08/08/1956	Divorced	80,000

Example #	< 20	20-30	> 30	Single	No Marital Status	Income
1	1	0	0	0	0	0
2	0	1	0	1	0	150,000
3	0	0	1	0	1	40,000
4	0	0	1	1	0	80,000

Feature normalisation

Making sure the feature values are at the same scale

Allows each feature to be weighted by ML algorithm, not the data (many classifiers use Euclidean distance)

Speeds up building model when you have many features

Ideally from -1 to 1

$$\text{Value}_{(\text{normalised})} = (\text{Value} - \text{Value}_{(\text{mean})}) / (\text{Value}_{(\text{max})} - \text{Value}_{(\text{min})})$$

Example

Example #	DOB	Marital Status	Income
1	01/04/2000	Married	0
2	05/12/1992	Single	150,000
3	22/06/1976	null	40,000
4	08/08/1956	Divorced	80,000

Example #	< 20	20-30	> 30	Single	No Marital Status	Normalised Income
1	1	0	0	0	0	-0.45
2	0	1	0	1	0	0.55
3	0	0	1	0	1	-0.18333
4	0	0	1	1	0	0.083333

What is a relevant feature?

Email addresses, user ID’s, Names likely not relevant

ML can help you figure it out – some algorithms have built-in feature selection like random forest

Some algorithms can handle more noise than others.

Forward selection/Backward elimination - start from no features and iteratively find the best features to add, or start from all features and iteratively remove the worst

Questions around gathering data

How do I obtain known values of my target variable / objective field?

- Dedicated analysts
- Crowd sourcing
- Interviews, surveys, controlled experiments, etc

How much training data do I need?

- More, if adding more data makes a difference

How do I know if my training data is good enough?

Visualising your data

Help to determine what features are most relevant

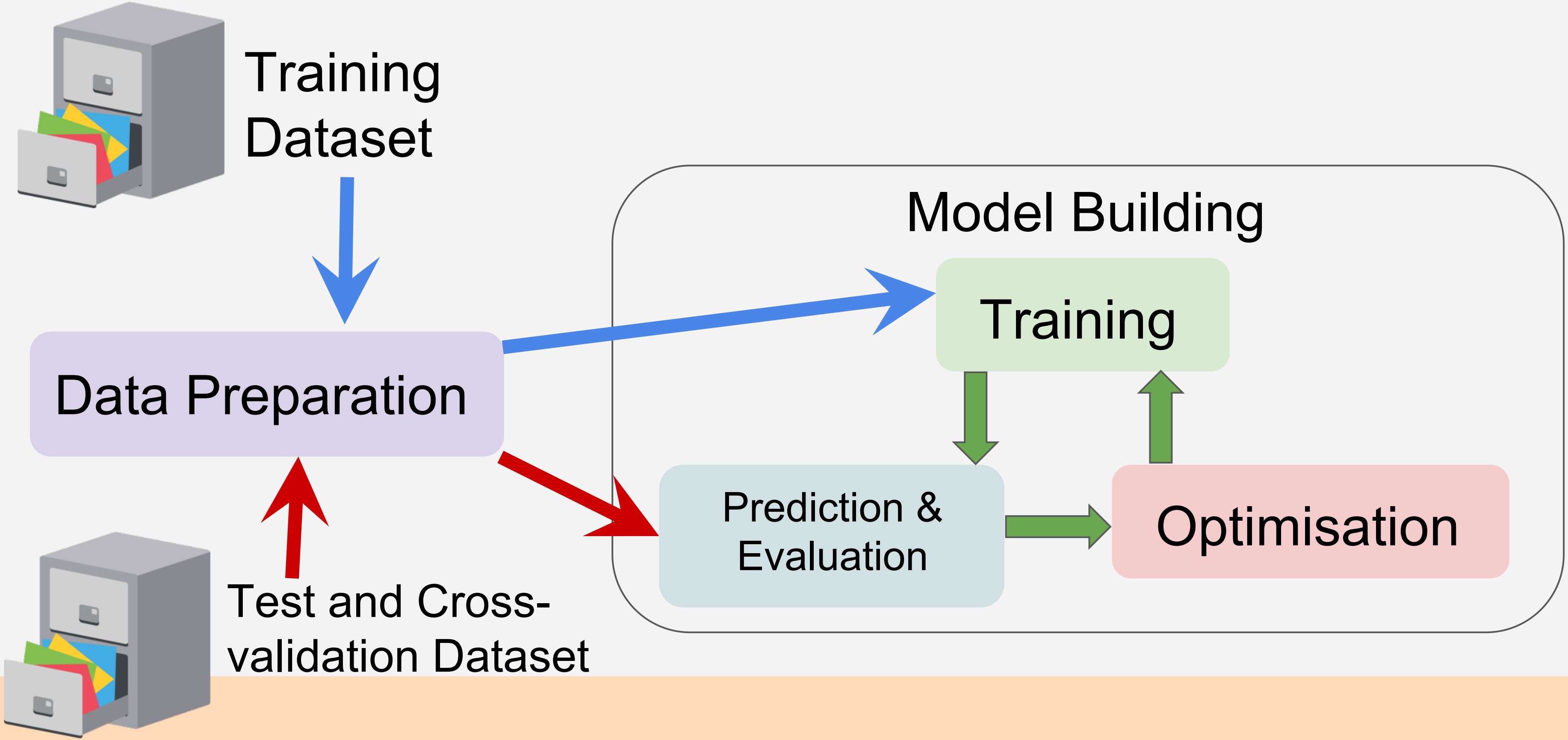
Spot anomalies or unusual data you may want to exclude

Help you choose a more suitable learning algorithm

Data exercises

https://github.com/mjnguyennz/ml_workshop_kiwiruby/blob/master/Data_Exercises.md

The Machine Learning Modeling Process



Evaluating your model

Predict on test data and evaluating the results

How do you measure performance?

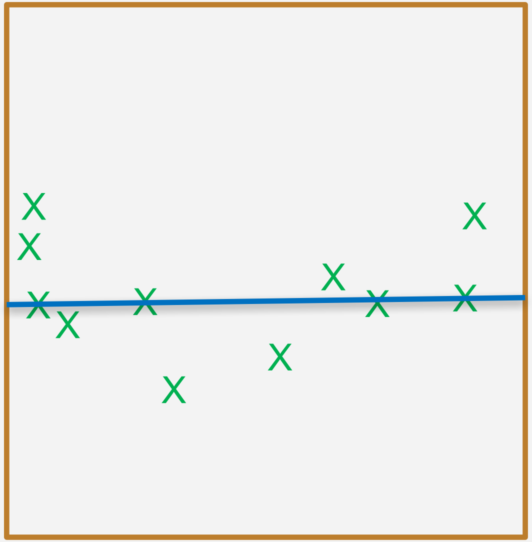
Accuracy

Precision

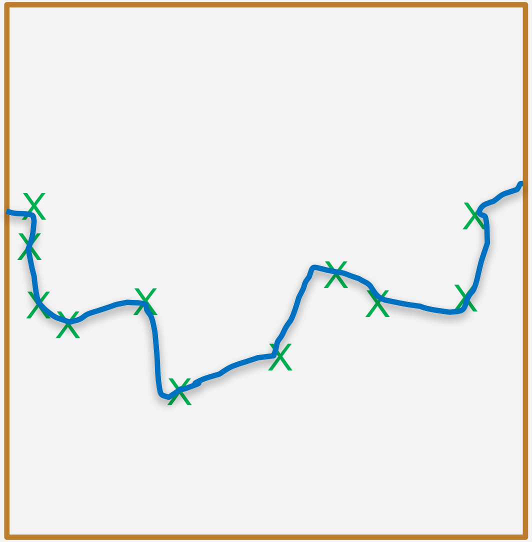
Regression – Mean Squared Error, Root Mean Squared Error, or R^2

Classification – Mean Accuracy (not ideal), F-score better

Underfitting and Overfitting

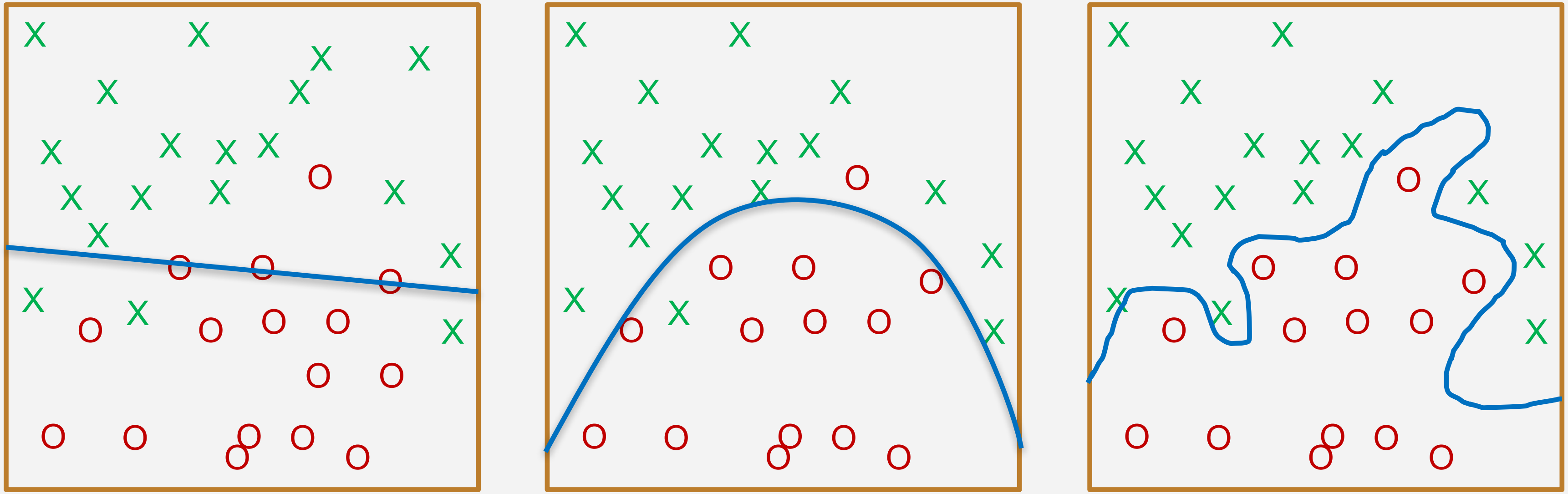


Underfitting



Overfitting

Underfitting and Overfitting



Underfitting

Overfitting

How to recognise and help underfitting

Underfitting

- Predicting on your training data performs poorly
- Predicting on test data performs poorly

How to improve?

- Adjust tuning parameters
- Try adding more features
- Try a more flexible ML algorithm

How to recognise and help overfitting

Overfitting

- Predicting on your training data performs very well
- Predicting on test data performs poorly

How to improve?

- Adjust tuning parameters
- Get more data for training
- Consider reducing features
- Try ML algorithm less prone to overfitting

Other algorithm considerations

Consider what is more important to you:

- Accuracy vs speed
- Faster predictions vs Faster training
- Memory and computational limitations

Examples of some tuning parameters

K-nearest neighbors—Number of nearest neighbors to average

Decision trees—Splitting criterion, max depth of tree, minimum samples needed to make a split

Kernel SVM—Kernel type, kernel coefficient (gamma), penalty parameter

Random forest—Number of trees, number of features to split in each node, splitting criterion, minimum samples needed to make a split

Different algorithms for different situations – some examples

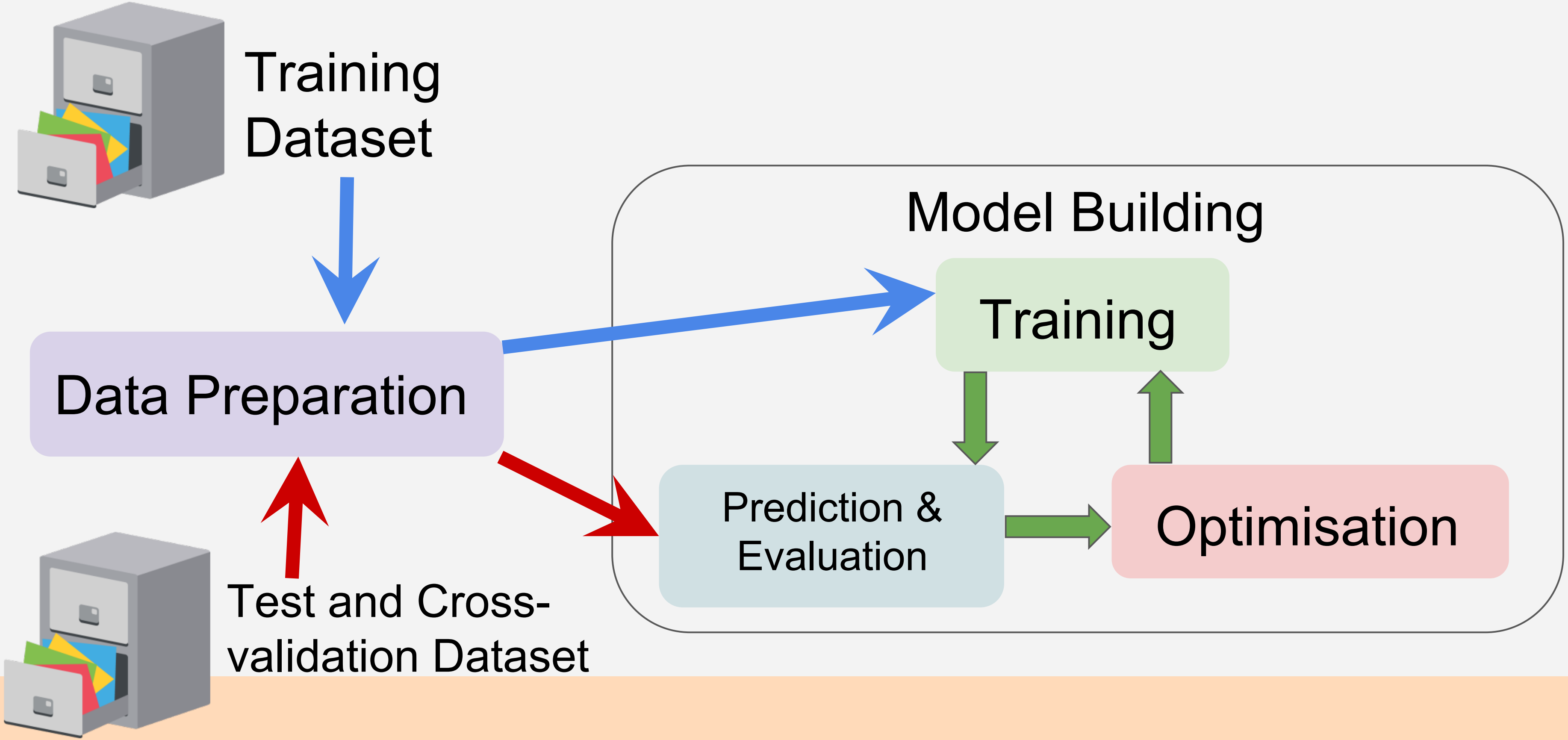
Linear regressions : scalable, computationally simple, risk underfitting

Non-linear regressions : not as computationally simple to train, risk of overfitting

K nearest neighbors : training is fast, but predictions slow

Random forest : collection of decision trees – slow to train, computationally more complex, but handles imperfect data better

The Machine Learning Modeling Process



Explore a MLaaS - BigML

Use the Quickstart guide to practice the full Supervised Learning workflow

Try the process for a Regression problem and/or Logistic Regression (classification) problem.

Quickstart Guide
https://github.com/mjnguyennz/ml_workshop_kiwiruby/blob/master/ML_with_BigML.md

Explore PyCall

PyCall - https://github.com/mrkn/pycall.rb

Written by Kenta Murata
Inspired by Julia’s PyCall package

This workskop’s exercises:
https://github.com/mjnguyennz/ml_workshop_kiwiruby/blob/master/ML_with_PyCall.md

Learn more about PyCall from Kenta:
https://github.com/RubyData/rubykaigi2017/blob/master/pycall_lecture.ipynb

More about Python Libraries

Tutorials for Python's libraries:

https://github.com/amueller/scipy-2017-sklearn

https://github.com/matplotlib/AnatomyOfMatplotlib

https://github.com/enthought/Numpy-Tutorial-SciPyConf-2017

https://github.com/jonathanrocher/pandas_tutorial

Now that you know a bit more about ML…

Next time you read some blog post about a gem/library that does ML.

Evaluate whether this will be useful for your ML problem.

Try to out with your data, and get results which you can iterate on.

Ethics around Machine Learning

Privacy, consent from users around data gathering and usage

ML model can become biased and discriminating on age, race, etc

- price discrimination
- financial
- employment

ML model reinforces the status quo because of training on past data

Many things I didn’t go over

Ensembles - An ensemble is a collection of models which are combined together to create a stronger model with better predictive performance.

Unsupervised Learning

Automated feature selection

Deep Learning

And so much more! ….

Further ML resources – less math

Real-World Machine Learning by H. Brink, J. W. Richards, M. Fetherolf (coding examples in Python)

Andrew Ng's Machine Learning Coursera course – implementing basic algorithms in Octave/Matlab (some math)

Further resources – more math

An Introduction to Statistical Learning by Gareth James et al. (coding examples in R)

http://www-bcf.usc.edu/~gareth/ISL/

The Elements of Statistical Learning: Data Mining, Inference, and Prediction by Trevor Hastie et al. (Springer, 2009).

https://web.stanford.edu/~hastie/ElemStatLearn/download.html

Pattern Recognition and Machine Learning by Christopher Bishop (Springer, 2007).

Takeaways

Many options to use Machine Learning in your Ruby stack

The workflow process is the same, just mix and match the tools

Quality and quantity of data is very important

This was just a taste, but I hope it inspires you to continue exploring ML

Cheers!

Github - @mjnguyennz
mjnguyen@gmail.com
@mjnguyen on RubyNZ Slack