Machine Learning for Cities

From Key Concepts to Smart City Applications

Smart Cities KSB – November 20, 2018

Machine Learning for Cities workshop

Objectives:

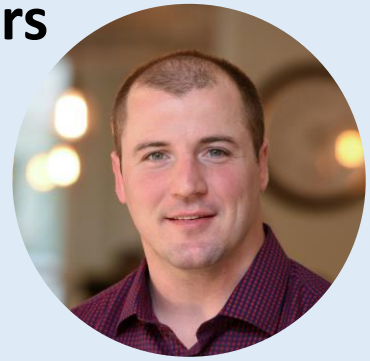
(1) Understand the steps to build and deploy a machine learning model for city authorities.

- **12.30pm-1.30pm**: Families of ML algorithms. Five steps to model fitting.

(2) Identify untapped datasets and use cases where ML can help your clients.

- **1.30pm-2.00pm:** Brainstorm city cases, identify training data.

Presenters

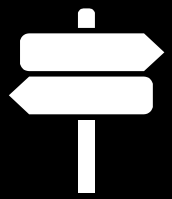


Jon Kastelan

Machine Learning specialist

Nick Jones

DRM Specialist, GFDRR



1. Introduction

What can Machine Learning do for city authorities?

What is Machine Learning?

Arthur Samuel (1959).	“Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.”
Andrew Ng (2017).	"Just as electricity transformed almost everything 100 years ago, today I have a hard time thinking of an industry that AI won't transform in the next several years"
Pedro Domingos (2015).	“People worry that computers will get too smart and take over the world, but the real problem is they’re too stupid and they’ve already taken over the world.”

What is Machine Learning?

Formal definition:

A computer program is said to learn from experience *E* with regard to task *T* and performance measure *P*, if its performance at task *T* as measured by *P* improves with experience.

#	Task	Experience	Performance Measure
1	Predict stock prices	History of stock prices	Average prediction accuracy
2	Recognize handwritten digits	Set of digits with labels	Percent of correct recognitions
3	Recommend Netflix shows	Viewing histories	# users viewing show

Why Machine Learning for Cities?

1. Urgent need to address urban challenges: disaster resilience, service delivery, poverty, environment.

2. New technologies have made cities increasingly data-rich environments, with large and complex datasets.

3. Expansion of processing power, and scalable data analysis methods to extract actionable information from this data.

Machine Learning techniques have become increasingly essential for urban policy analysis, and for developing new technologies that city authorities can use to allocate resources and serve their citizens.

Acknowledgment: D. Neill, Machine Learning for Cities, CUSP NYU

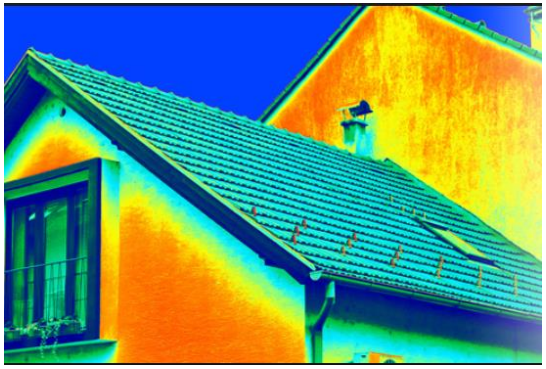
Some motivating examples

Early detection of disease outbreaks

Identifying vulnerable buildings for retrofit

Predicting transport demand

Preventing violent crime



Reducing CO2 emissions

Targeting fire risk inspections

Acknowledgment: D. Neill, Machine Learning for Cities, CUSP NYU

Supervised Learning

Supervised learning is a category of algorithm that works by generalizing from known examples.

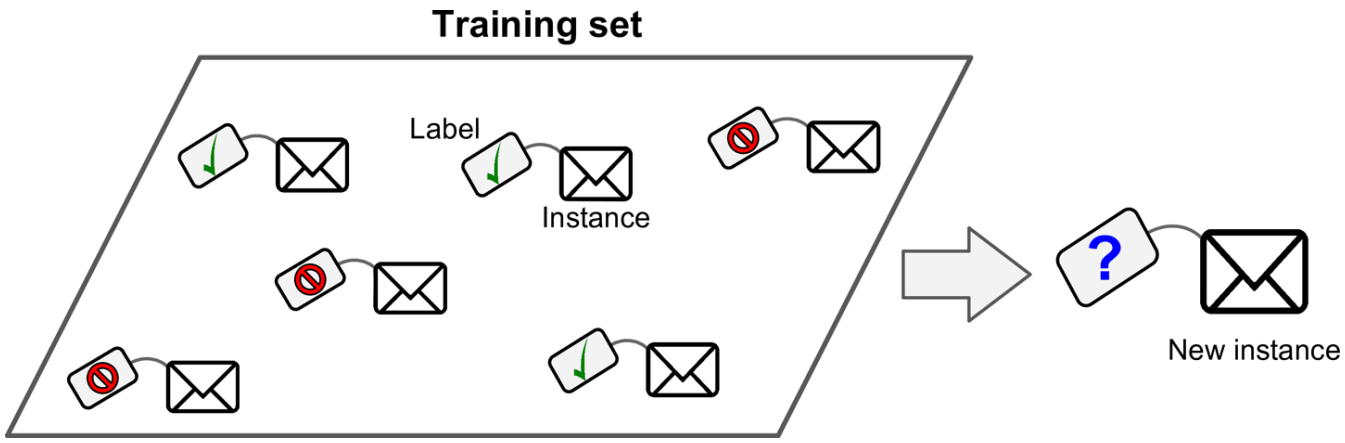


Figure: A labeled training set for spam classification[1]

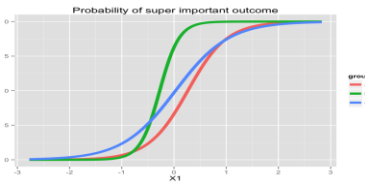
Unsupervised learning

We have data but no output labels. Example: Classify YouTube videos or segment website customers.

Other variants of learning include: Semi-supervised, Active and Reinforcement Learning

[1] Source: Géron, Hands-On Machine Learning

Families of Supervised Learning algorithms



Linear regression

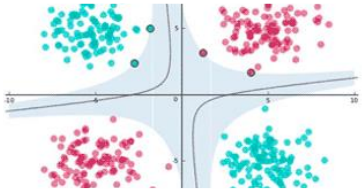
- Models output as linear combination of inputs

➢ Fast to train, effective on high-dimensional data.

Decision trees and Random Forest

- Builds flow-chart style rules that maximize information gain

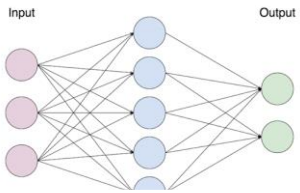
➢ High predictive power, requires less data preparation.



Support Vector Machines

- Learns a decision boundary (linear or non-linear)

➢ Good for complex, medium-size datasets



Neural networks and deep learning

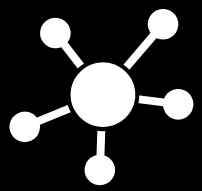
- Algorithms inspired by structure and function of the brain.
- Scalable, highly accurate on complex tasks like image recognition.

An approach to model fitting

Fitting a machine learning model involves five main steps

Use Case & Data	Model training	Tune (calibrate)	Predict	Evaluate
Determine the use-case you are interested in and source data.	Split the data into training and test sets. Fit model to training set.	Tune model parameters to moderate complexity & interpretability	Use the model to make predictions about test set	Compare the predictions with the actual values

Variables of interest are typically either **categorical**, supported by *classification* OR **numerical**, supported by *regression*



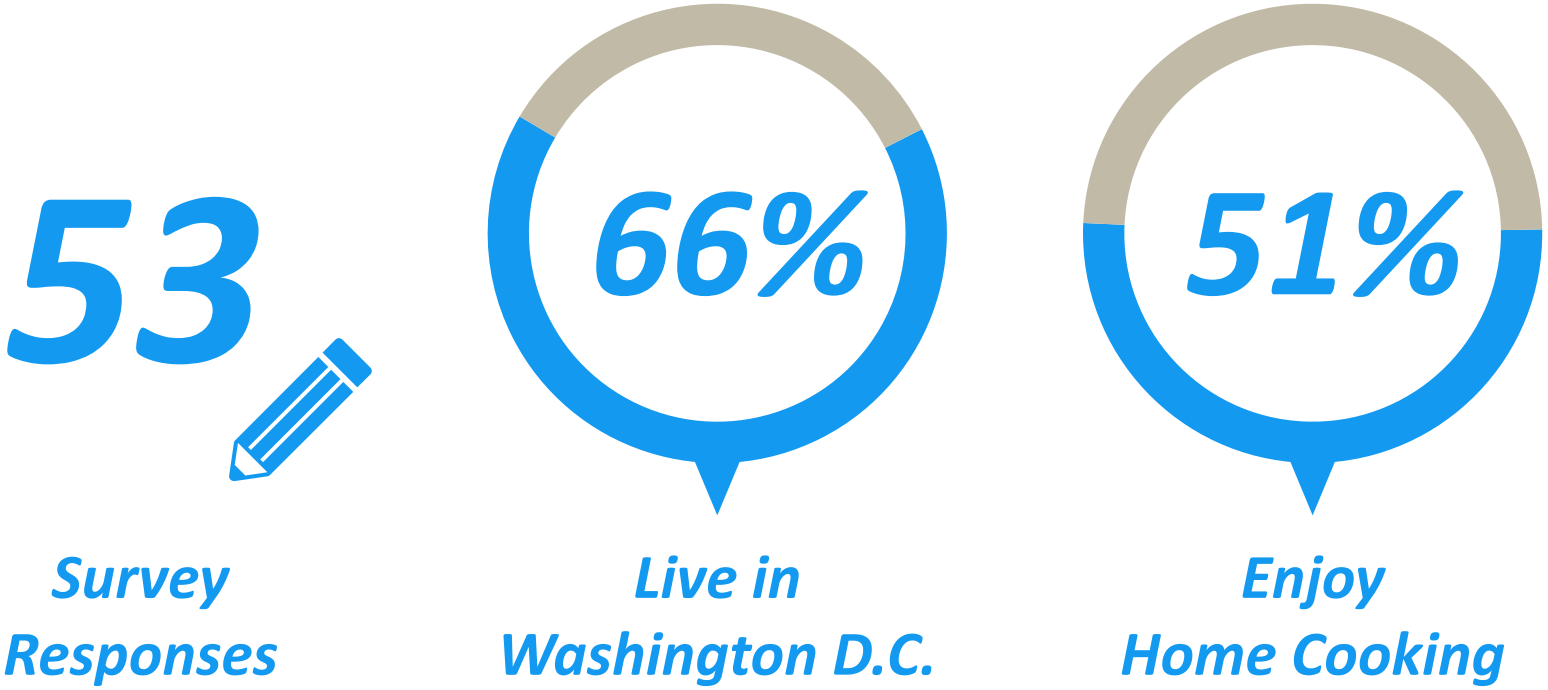
2. Building a model

The mechanics of training a ML algorithm

EXAMPLE:
Predicting mode of transport and music taste

Decision tree model for transport planning

Scenario[1]: The World Bank has hired a talented cohort of 100 new staff, who start after Thanksgiving. GSD needs to decide how many bike racks or parking spaces to build for them.



[1] Fictional scenario for teaching purposes

Decision tree model for transport planning

Scenario[1] : The World Bank has hired a talented cohort of 100 new staff, who start after Thanksgiving. GSD needs to decide how many bike racks or parking spaces to build for them.

Attributes (X_1 … X_N) **Target variable (y)**

gender	state	favorite_food	exercise_regime	age	mode_of_travel
Female	DC	Vegetarian	Keep fit through jogging or gym.	25-34	Walk or bike
Female	Virginia	Sushi	Netflix is my most strenuous exercise.	35-44	Train
Male	Maryland	Home cooking	Keep fit through jogging or gym.	35-44	Train
Male	DC	Home cooking	Keep fit through jogging or gym.	55+	Train
Male	Virginia	Home cooking	Keep fit through jogging or gym.	45-54	Train

Let's build a classifier!

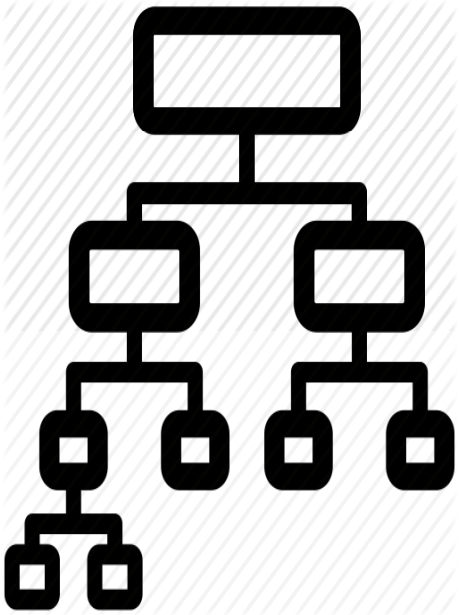
[1] Fictional scenario for teaching purposes

Building a decision tree (CART)

1. Start with all the samples (this is the *'root node'*).

2. Split the samples into two new nodes by asking a 'true/false' question.
   - Is feature k greater than threshold t *(example: "age > 35?")*
   - Choose the (k, t) combination that produce the purest subsets (measure: Gini impurity).

3. Keep creating new splits until each node is pure (contains only one class).



Gini impurity

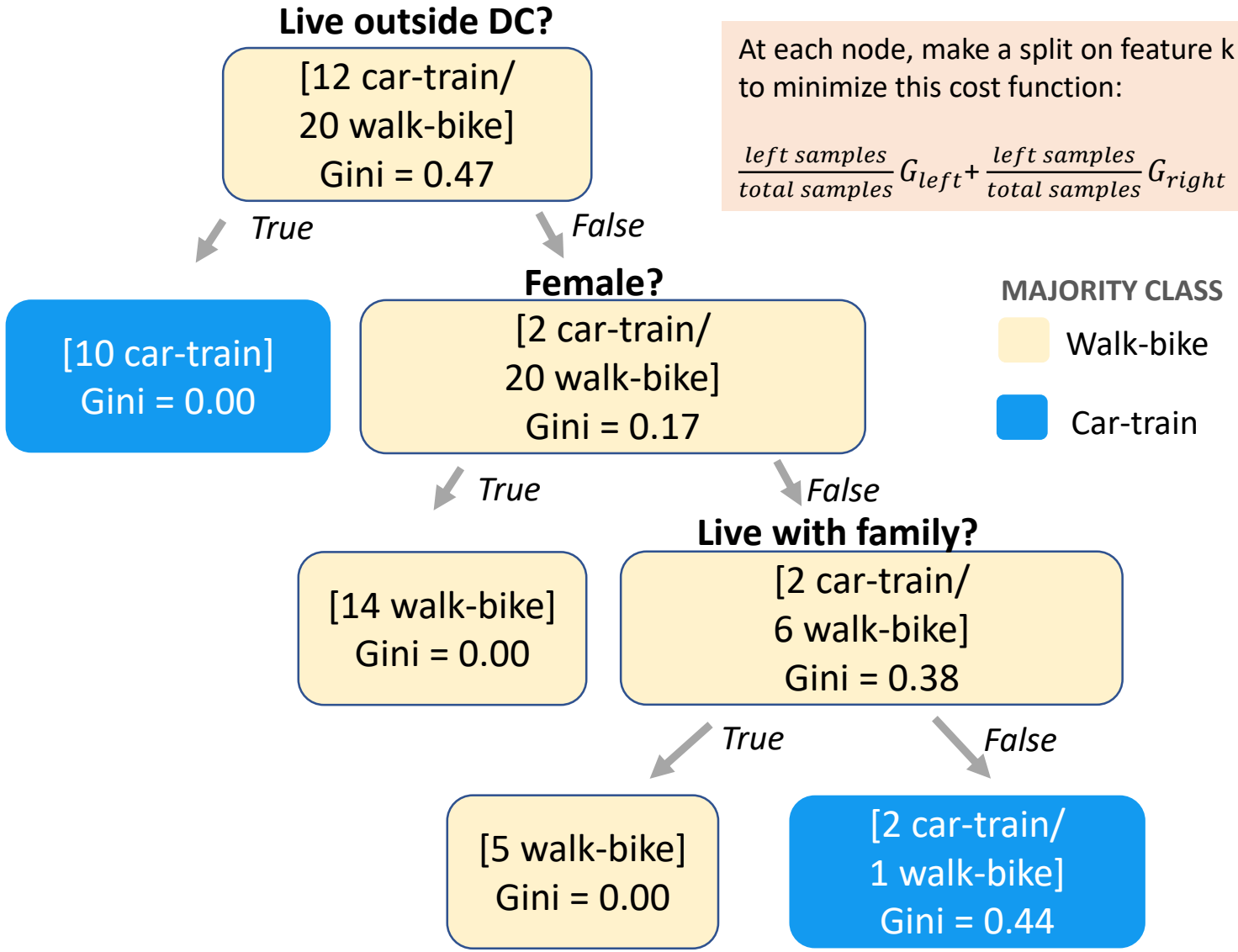
$$= 1 - \left(\frac{\boldsymbol{class\ A}}{\boldsymbol{n}}\right)^2 - \left(\frac{\boldsymbol{class\ B}}{\boldsymbol{n}}\right)^2$$

Where n represents number of samples in the node.

[2 walk-bike; 6 car-train]
Gini = …

$$1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 = 0.38$$

Decision tree: Transport



At each node, make a split on feature k to minimize this cost function:

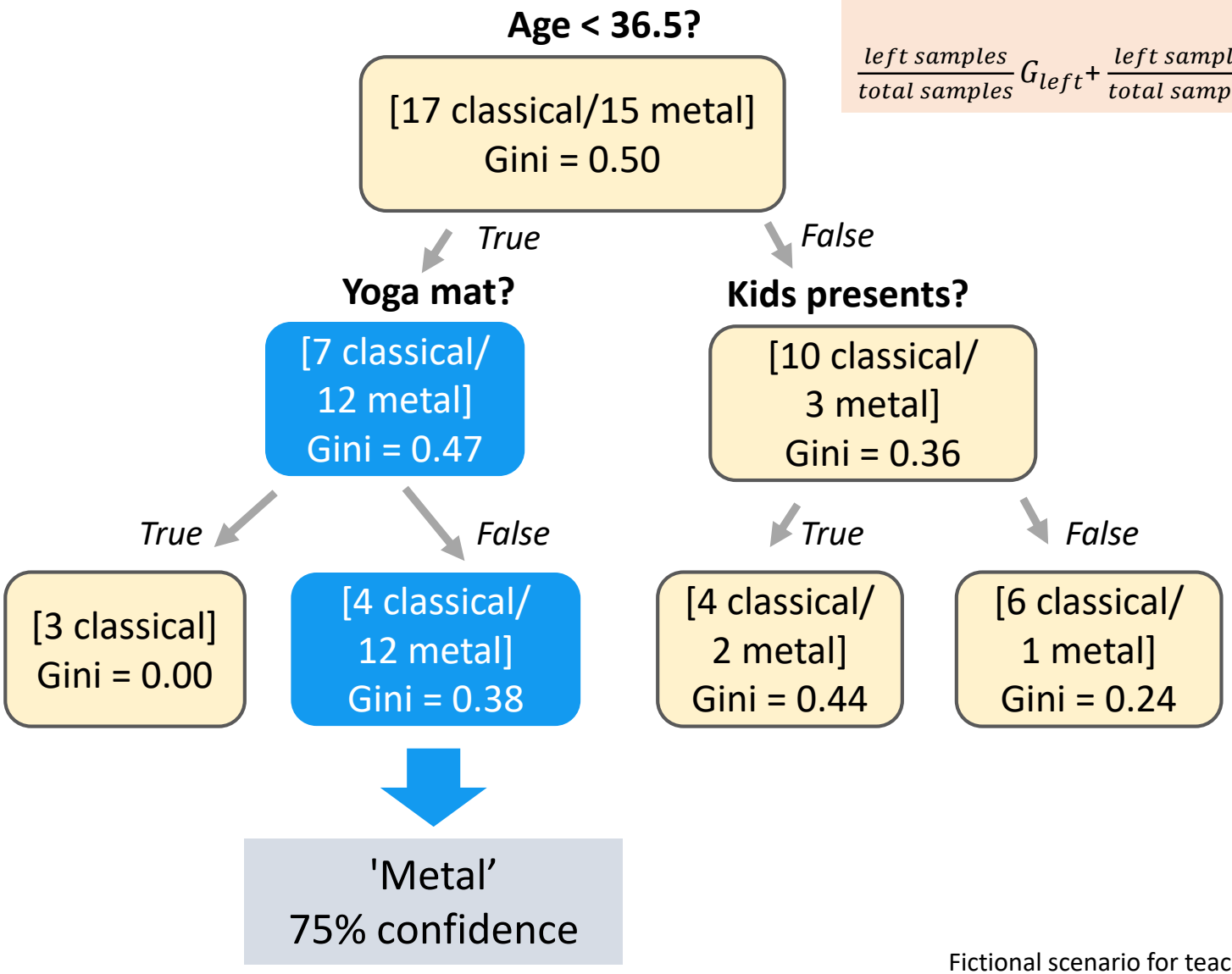
$$\frac{left\ samples}{total\ samples} G_{left} + \frac{left\ samples}{total\ samples} G_{right}$$

Fictional scenario for teaching purposes

Decision tree: Music

At each node, make a split on feature k to minimize this cost function:

$$\frac{left\ samples}{total\ samples} G_{left} + \frac{left\ samples}{total\ samples} G_{right}$$



Fictional scenario for teaching purposes

Complexity, Accuracy & Interpretability

Previous examples illustrate flexible and straightforward approach to classification.
Splits are interpretable, even if not intuitive

Better computing & algorithms

Increasing complexity of models

Higher levels of accuracy

Occam's razor
Principle of parsimony

With competing hypotheses to solve a problem, select the solution with the fewest assumptions

Modern ML techniques can trade-off between:

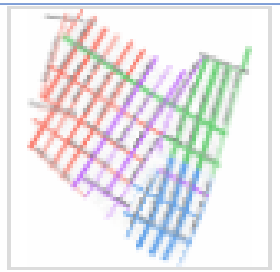
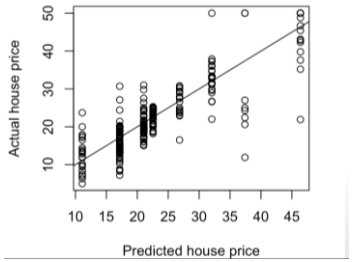
Accuracy and **Interpretability**

Ideally, you have both. Although maybe one is needed more than the other.


Use Case & Data
Model training
Tune (calibrate)
Predict
Evaluate


Use Case & Data
Model training
Tune (calibrate)
Predict
Evaluate

Build labelled datasets for question of interest

Question	Attributes (X_1 … X_N)	Target variable (y)	Example
How much solid waste will building X produce? [1]	Floor space Building type Weather Neighborhood characteristics	Tons of waste per week	
Predict median house price by zip code? [2]	building_age avg_rooms property_tax_rate employment_dist	Median price of owner-occupied homes	
Where are the soft-story buildings in Guatemala City [3]	Building imagery Estimated building height Estimated roof material	Soft story building (0/1)	

[1] Using machine learning and small area estimation to predict building-level municipal solid waste generation in cities – Kontokosta et al, (July 2018)

[2] Decision trees to predict house prices - James Gammerman (April 2017)

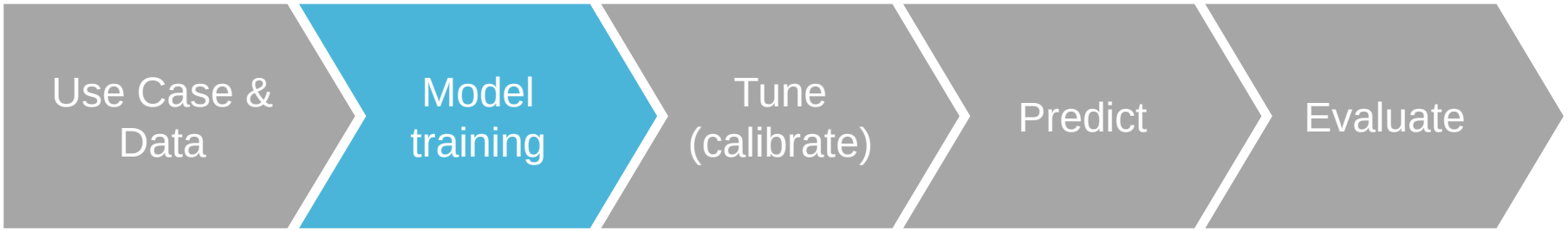
[3] Resilient Housing Joins the Machine Learning Revolution – Sarah Antos, Luis Triveno (November 2018)

Create a good set of features

Data collection and cleaning can represent two-thirds of the time of an urban analytics project:

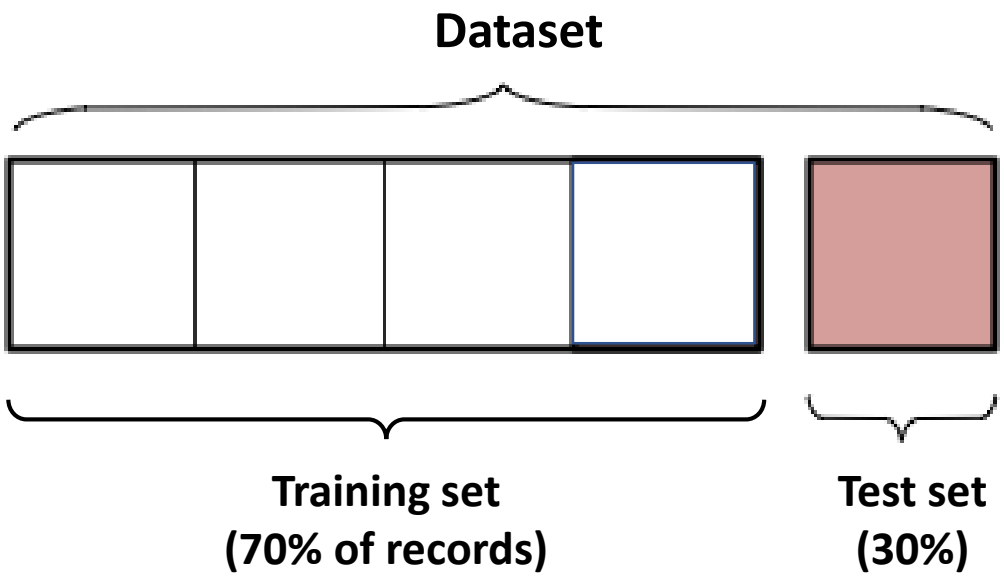
Collection and aggregation	Data sharing, privacy	Missing data	Merging across spatial granularities

- Building training data is hard work: be creative.
- For model fitting: Garbage In → Garbage Out


Use Case & Data
Model training
Tune (calibrate)
Predict
Evaluate

Split training and test data

When fitting ML algorithms, it is common to separate data into training and test sets*



Split the dataset (e.g. 70/30 ratio)

Build model on the *training set*

Evaluate model on the *test set*

Seems easy, and for the most part – hopefully, it is. A few considerations:

- Time dependence of observations (e.g. with time series)
- Rare events – use up-sampling or down-sampling as required
- Bias / representativeness of training set

*A *validation set* may also be split if needed

Image credit: D. Ziganto "Standard Deviations" blog

Train the model

There are many different ML techniques which could be applied, the ‘right’ one is problem dependent

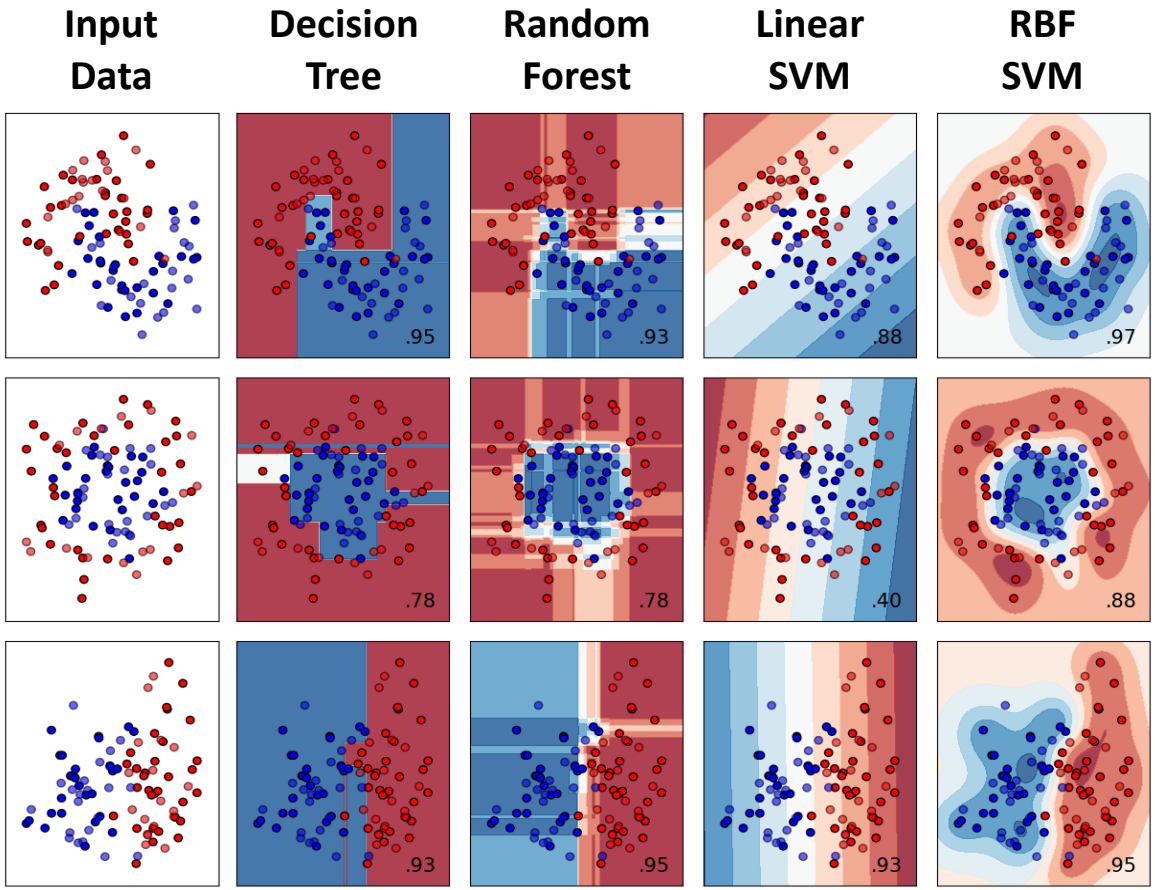


Figure: The plots show two classes (red and blue), separated using different techniques. Classification accuracy is reported on the lower right of each panel.

Which model is best?

Easy to ‘try out’ many different classifiers

Try a few, and compare their performance

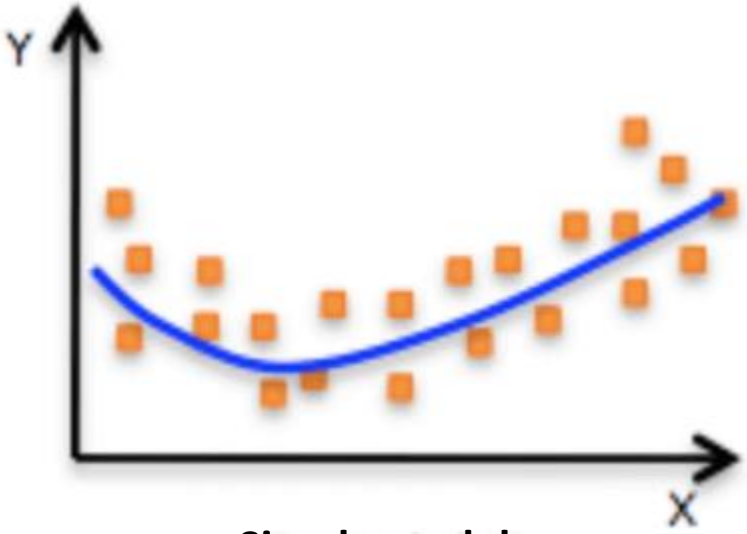
Image credit: Classifier comparison – scikit learn documentation


Use Case & Data
Model training
Tune (calibrate)
Predict
Evaluate

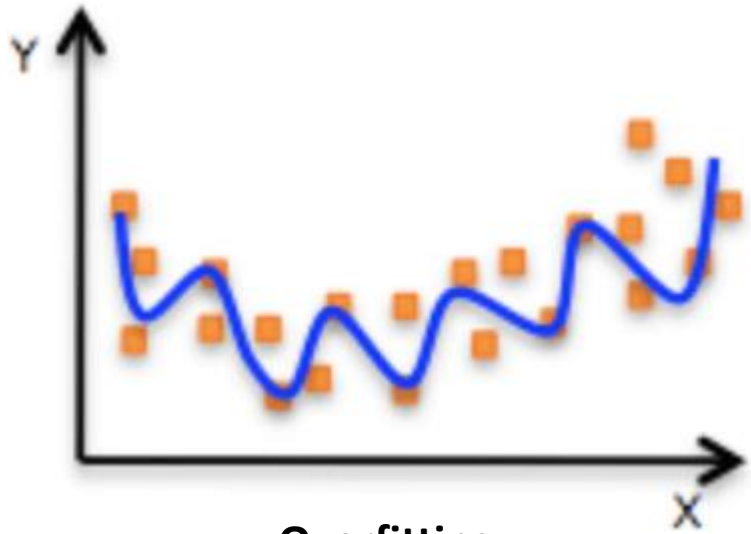
Complexity vs. Accuracy

We can build models of lower or higher complexity by changing their parameters.

Aim for the ‘sweet spot’ that maximizes performance but avoids overfitting*.



Simple model
(some errors)



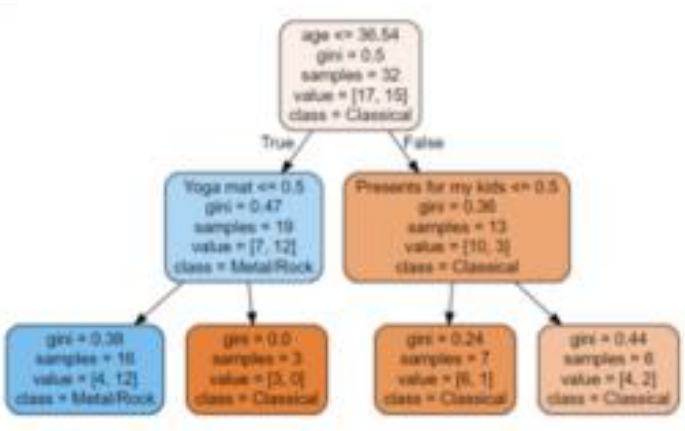
Overfitting
(will fail on unseen examples)

*__Overfitting__: a complex model that memorizes the test set (including noise in it) but fails to generalize to new data.

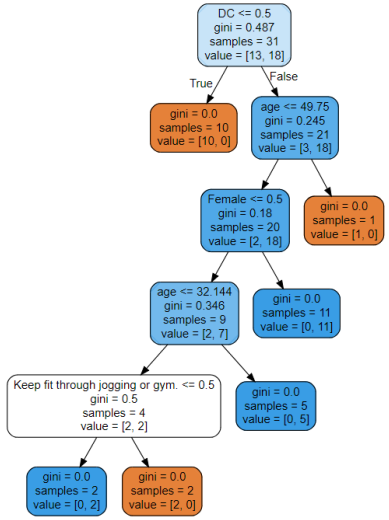
Complexity vs. Accuracy

We can build models of lower or higher complexity by changing their parameters.

Aim for the ‘sweet spot’ that maximizes performance but avoids overfitting*.



Simple model
(some errors)



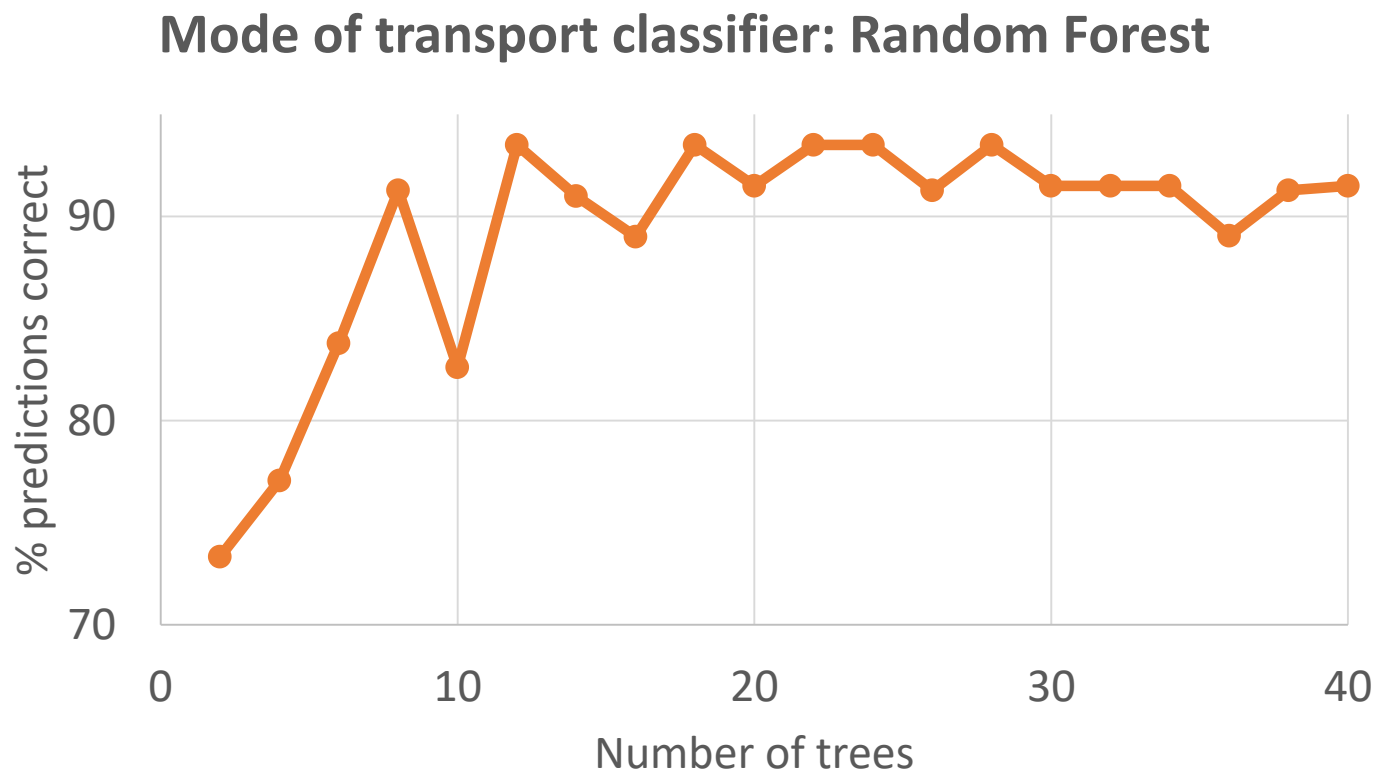
Overfitting
(will fail on unseen examples)

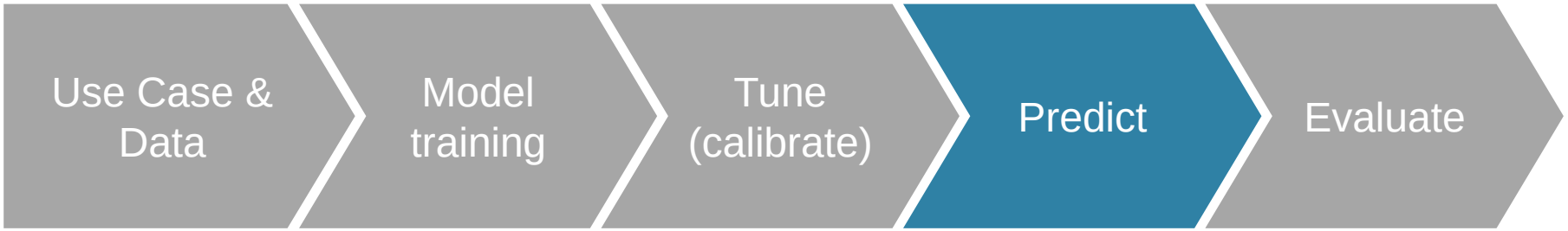
*__Overfitting__: a complex model that memorizes the test set (including noise in it) but fails to generalize to new data.

Tune model parameters

Decision tree parameters include maximum tree depth, minimum samples for a split, and (for Random Forest) number of trees.

- Choose the **parameter combination that maximizes prediction accuracy** on unseen data.




Use Case & Data
Model training
Tune (calibrate)
Predict
Evaluate

Make predictions

With the model tuned and fitted to training data, we can predict outcomes for test set

We have learnt a target function (f) that best maps input variables (X) to an output variable (Y): Y = f(X)

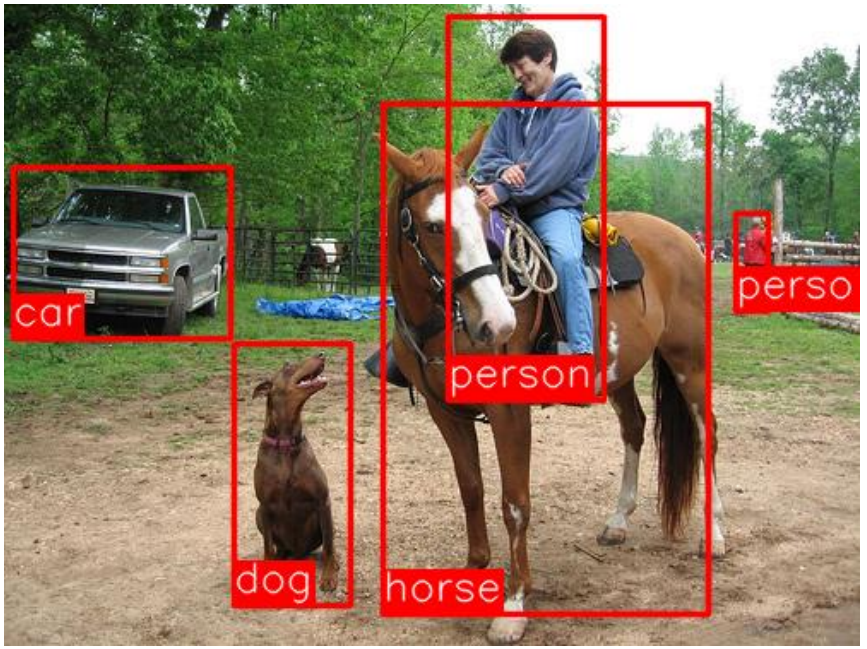
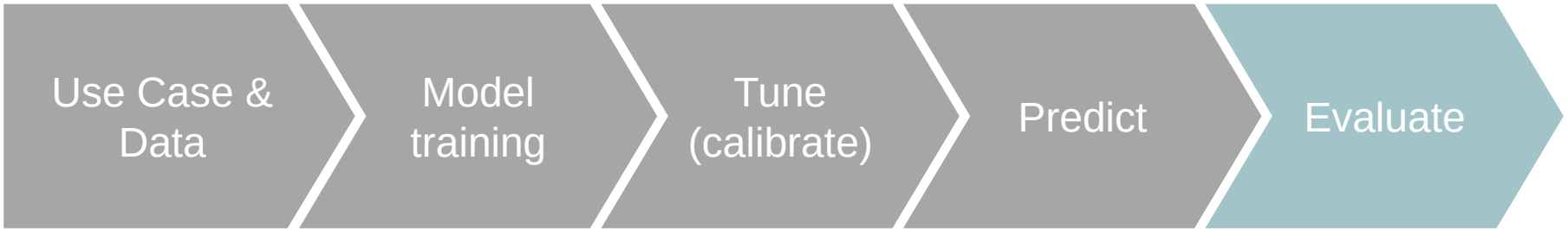


Figure: Object detection in images

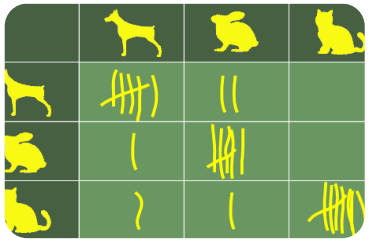
Image: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (2016)

3. Evaluating a model

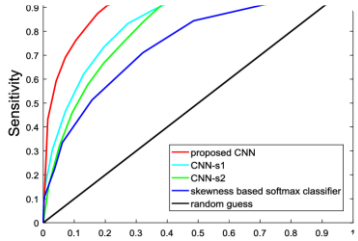
How do I know which model to use, and which provides the best results?


Use Case & Data
Model training
Tune (calibrate)
Predict
Evaluate

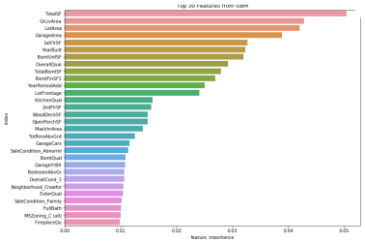
Techniques for model evaluation



Confusion matrix



ROC-curve



Feature importance

Confusion matrix

A **confusion matrix** is a common summary used in ML (and statistics) to assess the effectiveness of a model. It is used to compare:

Predicted vs. **Actual**

results

Let's illustrate with a simple classifier example

Confusion matrix - example

Question: Can we develop a fire detection algorithm to alert the fire department when a building is on fire? Consider this algorithm applied to over 10,000 buildings

Confusion matrix - example

Question: Can we develop a fire detection algorithm to alert the fire department when a building is on fire? Consider this algorithm applied to over 10,000 buildings

		The fire alarm goes off (model predicted)	
		Yes	**No**
There is a fire (actual)	**Yes**		
	No		

Confusion matrix - example

Question: Can we develop a fire detection algorithm to alert the fire department when a building is on fire? Consider this algorithm applied to over 10,000 buildings

		The fire alarm goes off (model predicted)	
		Yes	**No**
There is a fire (actual)	**Yes**	*Cases of detected fires:* 10  **TRUE POSITIVE**	
	No		

There is a fire in the building. An alarm goes off, and fire department attend.

Whilst fire is not ideal, it is good the detection algorithm identified it and evidence the model is effective

Confusion matrix - example

Question: Can we develop a fire detection algorithm to alert the fire department when a building is on fire? Consider this algorithm applied to over 10,000 buildings

		The fire alarm goes off (model predicted)	
		Yes	**No**
There is a fire (actual)	**Yes**	*Cases of detected fires:* 10 TRUE POSITIVE	*One fire **not** detected:* 1  **FALSE NEGATIVE**
	No	Alternate scenario where model fails to detect an actual fire. Costly failure, potentially resulting in property damage and risk to residents living there	

Confusion matrix - example

Question: Can we develop a fire detection algorithm to alert the fire department when a building is on fire? Consider this algorithm applied to over 10,000 buildings

		The fire alarm goes off (model predicted)	
		Yes	**No**
There is a fire (actual)	**Yes**	*Cases of detected fires:* 10 TRUE POSITIVE	Alarm is triggered however there is no actual fire. Not as costly as an undetected burning building, it is annoying as we may send resources like the fire department to the building. Want to limit this if possible.
	No	*About 1% false alarms:* 100  **FALSE POSITIVE**	

Confusion matrix - example

Question: Can we develop a fire detection algorithm to alert the fire department when a building is on fire? Consider this algorithm applied to over 10,000 buildings

		The fire alarm goes off (model predicted)	
		Yes	**No**
There is a fire (actual)	**Yes**		
	No	*About 1% false* 	*Non-fires reported:* 10,000 
		FALSE POSITIVE	**TRUE NEGATIVE**

Sub-case with well functioning fire detection algorithm, where alarm isn't triggered when without a fire.

i.e. the ideal algorithm gets both all the positives and negatives correct

Confusion matrix - example

Question: Can we develop a fire detection algorithm to alert the fire department when a building is on fire? Consider this algorithm applied to over 10,000 buildings

		The fire alarm goes off (model predicted)	
		Yes	**No**
There is a fire (actual)	**Yes**	*Cases of detected fires:* 10 	*One fire **not** detected:* 1 
		TRUE POSITIVE	**FALSE NEGATIVE**
	No	*About 1% false alarms:* 100 	*Non-fires reported:* 10,000 
		FALSE POSITIVE	**TRUE NEGATIVE**

Confusion matrix - discussion

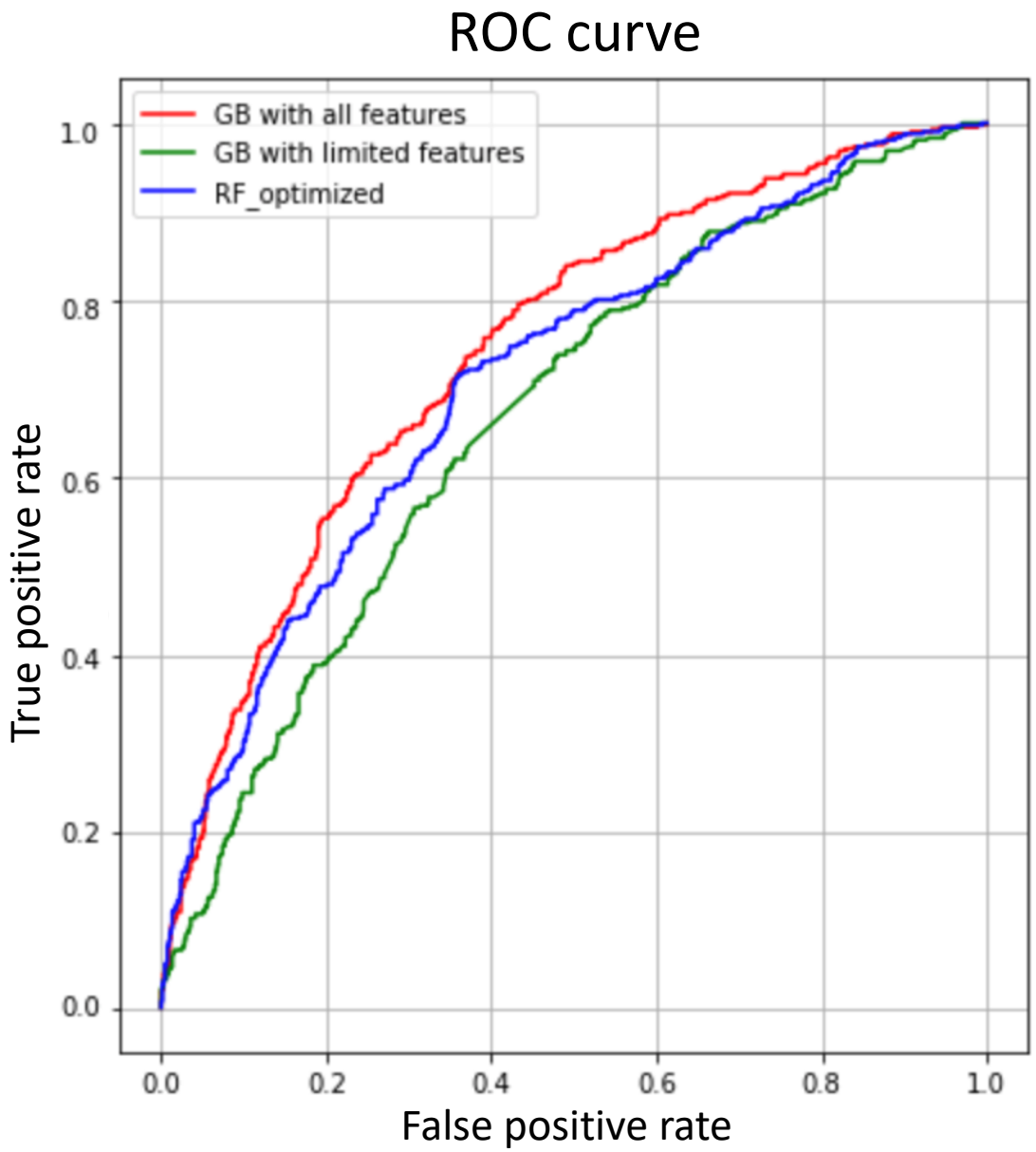
In the above case, we examined a *fire detection algorithm*. What about some other examples:

- Spam email filter
- Allocation of workforce to inspect buildings
- Medical test

In each of these cases, the 'cost' of falsely classifying either a Positive or Negative 'Actual' has different implications.

It is common to trade-off the costs associated with False Positives and False Negatives using statistical **Precision** and/or **Recall** metrics

Model selection



The **ROC curve** is a commonly used technique to *compare models* and *classification of classes* within in a model.

Aim for the top left. Other information such as model complexity, speed to response may be considered

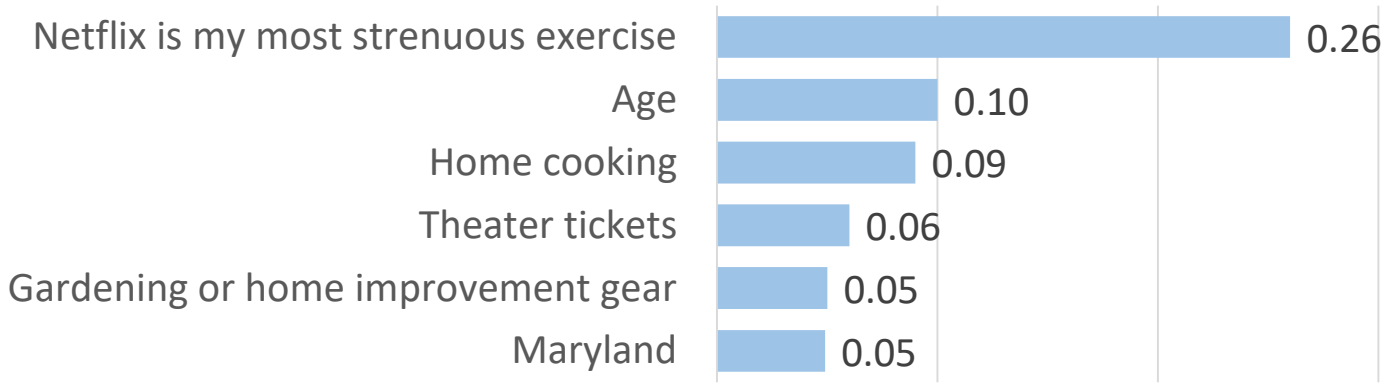
Feature importance

For some questions, we are not so interested in predicting the outcome; rather **what factors are important at determining an outcome?**

Feature importance provides us some insight into the factors which improve model accuracy.

e.g. What factors are important in explaining student outcomes and education performance?

Returning to regression tree example, to explain ***"Arrival time to work"***



4. Imagining urban use cases

How could WB apply machine learning techniques to better support cities?

Case studies

Three cases, then over to you:

1. Prioritize building inspections
2. Target land-bank interventions
3. Detect polluting plumes

Case 1: Prioritize building inspections

Question: Could an algorithm reduce risk in the built environment by sending building inspectors to the most dangerous cases first?

Challenges:

Buildings data can be fragmented across agencies.
Time-consuming to build training set.

Management was initially hesitant about ML. Deploying as an interactive dashboard helped make outputs intuitive.

Data used:

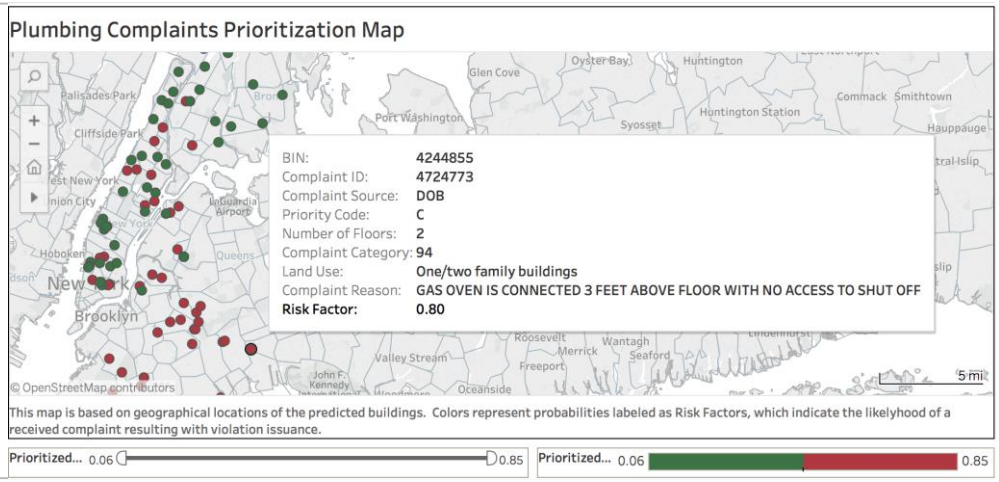
1. Building characteristics (year built, number of floors, retail/residential)
2. Neighborhood demographics (median income, percent homes owner-occupied)
3. City records of prior code violations and construction filings.
4. Source of the complaint or referral.

Outputs / visualization:

Best model is a Gradient Boosting Classifier. Trees: 750; max depth 9.

Trained on data from 2015-2017.

Achieves 70% accuracy on the unseen test data (1,200 plumbing complaints from 2017-2018).



Case 2: Target land bank interventions

Question: Can we train a classification algorithm to identify vacant homes for possible intervention by the Detroit Land Bank Authority, without sending officers to look?

Challenges:

Targeting properties for buyback or demolition required detailed lot-level data.

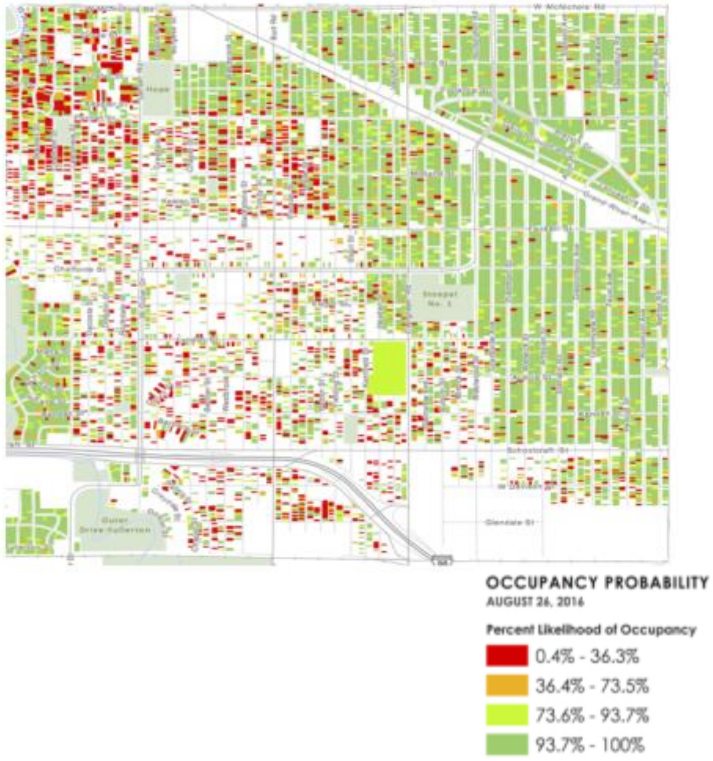
Negotiating agreements with utilities and postal service took time.

Data used:

- Voter registration data
- Fire records
- Postal delivery
- Utility bill payment
- 'Blexters' (blight texters) labeled buildings

Outputs / visualization

Brightmoor / Minock Park / Rosedale Park Occupancy Map,



Case 3: Plume detection (images)

Question: Can we develop an algorithm to automatically detect polluting plumes (ash clouds) from images of buildings

Challenges:

No readily available plume dataset for model training

Requirement to build the training set by reviewing **lots** of images

Biases such as weather and lighting conditions impact ability to see plumes

Data used:

- Images of New York City, Eastern Manhattan skyline
- Continuously sampled every 10 sec
- Approx. 1,000 buildings in the field of view

Visualization

Figures: Plume detection examples

Reference: CUSP Urban Observatory, New York University

Over to you..

Case exercise – Part A (15 minutes)

In your table groups, work together to populate a similar case study:

Question: *[devise your response]*

Develop a use-case which you would like to address using Machine Learning

- Choose a city for which this would be useful?
- Consider the:
  - Impact,
  - Users, and
  - Main beneficiaries

Data: *[your response here]*

What data will support the development of this algorithm?

- What data is available?
- What are your data 'wishes' and 'dreams'?

Challenges: *[your response here]*

What challenges may be involved in developing this ML application, and how might you resolve them? For example:

- Privacy
- Stakeholder sensitivity
- Partnerships
- Data granularity

Report out

Thanks!

Nick Jones
njones@worldbank.org

Jon Kastelan
jlk635@nyu.edu