Machine Learning for Cities
From Key Concepts to Smart City Applications

Smart Cities KSB – November 20, 2018
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?

<table>
<thead>
<tr>
<th>Name</th>
<th>Quote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arthur Samuel (1959).</td>
<td>“Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.”</td>
</tr>
<tr>
<td>Andrew Ng (2017).</td>
<td>&quot;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&quot;</td>
</tr>
<tr>
<td>Pedro Domingos (2015).</td>
<td>“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.”</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>#</th>
<th>Task</th>
<th>Experience</th>
<th>Performance Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Predict stock prices</td>
<td>History of stock prices</td>
<td>Average prediction accuracy</td>
</tr>
<tr>
<td>2</td>
<td>Recognize handwritten digits</td>
<td>Set of digits with labels</td>
<td>Percent of correct recognitions</td>
</tr>
<tr>
<td>3</td>
<td>Recommend Netflix shows</td>
<td>Viewing histories</td>
<td># users viewing show</td>
</tr>
</tbody>
</table>
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 is a category of algorithm that works by generalizing from known examples.

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.
Fitting a machine learning model involves five main steps

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

Survey Responses: 53

- 66% Live in Washington D.C.
- 51% Enjoy Home Cooking

1 Fictional scenario for teaching purposes
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 ($X_1 \ldots X_N$)  

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

¹ Fictional scenario for teaching purposes

Let’s build a classifier!
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**

\[
\text{Gini impurity} = 1 - \left(\frac{\text{class A}}{n}\right)^2 - \left(\frac{\text{class B}}{n}\right)^2
\]

Where n represents number of samples in the node.

Example:

- [2 walk-bike; 6 car-train]  
  Gini = ...

\[
1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 = 0.38
\]
At each node, make a split on feature k to minimize this cost function:

\[
\frac{\text{left samples}}{\text{total samples}} G_{\text{left}} + \frac{\text{left samples}}{\text{total samples}} G_{\text{right}}
\]

**Live outside DC?**

- **True**
  - 12 car-train/20 walk-bike
    - Gini = 0.47

- **False**
  - 10 car-train
    - Gini = 0.00

**Female?**

- **True**
  - 2 car-train/20 walk-bike
    - Gini = 0.17

- **False**
  - 14 walk-bike
    - Gini = 0.00

**Live with family?**

- **True**
  - 5 walk-bike
    - Gini = 0.00

- **False**
  - 12 car-train/6 walk-bike
    - Gini = 0.38

**MAJORITY CLASS**
- Walk-bike
- Car-train

Fictional scenario for teaching purposes
At each node, make a split on feature k to minimize this cost function:

$$
G = \frac{\text{left samples}}{\text{total samples}} G_{\text{left}} + \frac{\text{left samples}}{\text{total samples}} G_{\text{right}}
$$

Fictional scenario for teaching purposes
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
### Build labelled datasets for question of interest

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

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

- Building training data is hard work: be creative.
- For model fitting: Garbage In → Garbage Out
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
There are many different ML techniques which could be applied, the ‘right’ one is problem dependent.

Train the model

Which model is best?
Easy to ‘try out’ many different classifiers
Try a few, and compare their performance

**Input Data**
- Decision Tree
- Random Forest
- Linear SVM
- RBF SVM

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

Image credit: Classifier comparison – scikit learn documentation
Use Case & Data
Model training
Tune (calibrate)
Predict
Evaluate
We can build models of lower or higher complexity by changing their parameters.

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

*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*.

*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.
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*
3. Evaluating a model

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

Confusion matrix

ROC-curve

Feature importance
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
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.
**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.

<table>
<thead>
<tr>
<th>There is a fire (actual)</th>
<th>The fire alarm goes off (model predicted)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>False Positive</td>
</tr>
<tr>
<td>False Positive</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>False Negative</td>
</tr>
</tbody>
</table>

**Some cases of detected fires:**
- 10

**One fire not detected:**
- 1

**About 1% false alarms:**
- 100

**Mostly non-fires reported:**
- 10,000
**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.

<table>
<thead>
<tr>
<th>There is a fire (actual)</th>
<th>The fire alarm goes off (model predicted)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td><strong>Cases of detected fires: 10</strong></td>
</tr>
<tr>
<td></td>
<td><strong>TRUE POSITIVE</strong></td>
</tr>
<tr>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

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.
**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.

<table>
<thead>
<tr>
<th>There is a fire (actual)</th>
<th>The fire alarm goes off (model predicted)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Cases of detected fires: 10</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>One fire not detected: 1</td>
<td>No</td>
</tr>
</tbody>
</table>

Alternate scenario where model fails to detect an actual fire.

- Costly failure, potentially resulting in property damage and risk to residents living there.
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.

<table>
<thead>
<tr>
<th>There is a fire (actual)</th>
<th>The fire alarm goes off (model predicted)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Cases of detected fires: 10</td>
<td>Alarm is triggered however there is no actual fire.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>About 1% false alarms: 100</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>FALSE POSITIVE</td>
<td></td>
</tr>
</tbody>
</table>

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.
### 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.

<table>
<thead>
<tr>
<th>The fire alarm goes off (model predicted)</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Yes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>No</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>There is a fire (actual)</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Yes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-case with well functioning fire detection algorithm, where alarm isn’t triggered when without a fire.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i.e. the ideal algorithm gets both all the positives and negatives correct</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>No</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>About 1% false alarms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-fires reported: 10,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FALSE POSITIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUE NEGATIVE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>There is a fire (actual)</th>
<th>The fire alarm goes off (model predicted)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Yes</strong></td>
<td><strong>Yes</strong></td>
<td><strong>No</strong></td>
</tr>
<tr>
<td><strong>Cases of detected fires:</strong> 10</td>
<td>![Fire truck with sad face]</td>
<td>![Fire truck and fire]</td>
</tr>
<tr>
<td><strong>TRUE POSITIVE</strong></td>
<td>![Fire truck and sad face]</td>
<td>![Fire truck and cooling face]</td>
</tr>
<tr>
<td><strong>One fire not detected:</strong> 1</td>
<td>![Fire truck and sad face]</td>
<td>![Fire truck and cooling face]</td>
</tr>
<tr>
<td><strong>FALSE NEGATIVE</strong></td>
<td>![Fire truck and sad face]</td>
<td>![Fire truck and cooling face]</td>
</tr>
<tr>
<td><strong>About 1% false alarms:</strong> 100</td>
<td>![Fire truck and confused face]</td>
<td>![Fire truck and smiling face]</td>
</tr>
<tr>
<td><strong>FALSE POSITIVE</strong></td>
<td>![Fire truck and confused face]</td>
<td>![Fire truck and smiling face]</td>
</tr>
<tr>
<td><strong>Non-fires reported:</strong> 10,000</td>
<td>![Fire truck and smiling face]</td>
<td>![Fire truck and smiling face]</td>
</tr>
<tr>
<td><strong>TRUE NEGATIVE</strong></td>
<td>![Fire truck and smiling face]</td>
<td>![Fire truck and smiling face]</td>
</tr>
</tbody>
</table>
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.
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 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”

<table>
<thead>
<tr>
<th>Factor</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netflix is my most strenuous exercise</td>
<td>0.26</td>
</tr>
<tr>
<td>Age</td>
<td>0.10</td>
</tr>
<tr>
<td>Home cooking</td>
<td>0.09</td>
</tr>
<tr>
<td>Theater tickets</td>
<td>0.06</td>
</tr>
<tr>
<td>Gardening or home improvement gear</td>
<td>0.05</td>
</tr>
<tr>
<td>Maryland</td>
<td>0.05</td>
</tr>
</tbody>
</table>
4. Imagining urban use cases

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


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**

**Reference:** CUSP Urban Observatory, New York University
Over to you..
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