

# Welcome to Machine Learning!

---

COMP 4630 | Winter 2026

Charlotte Curtis

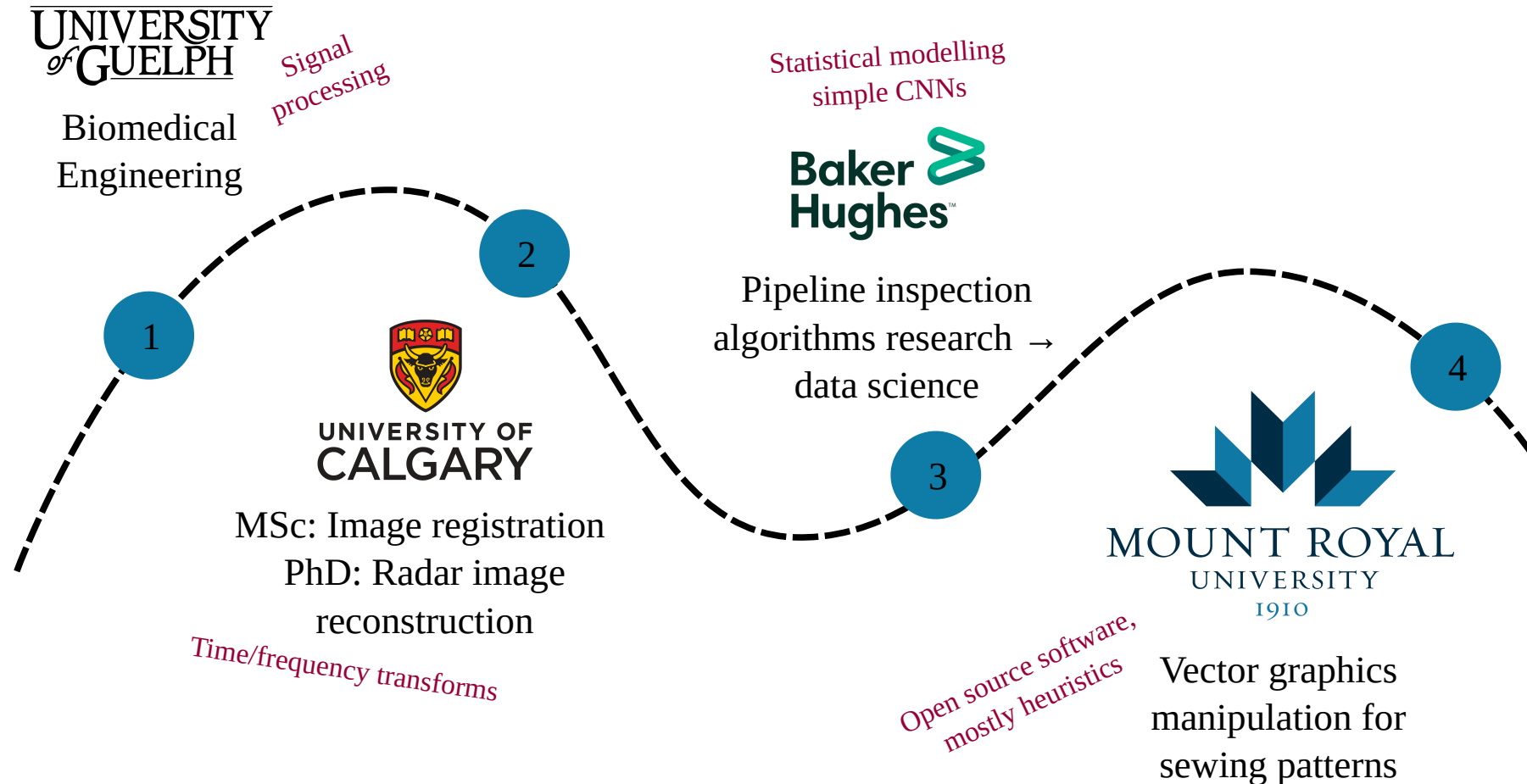
# What is this course about?

---

- Continuing the supervised/unsupervised learning algorithms from COMP 3652, with a focus on **Neural Networks**
- First half: the history, theory, and math behind neural networks
- Second half: applications of NNs in computer vision, natural language processing, and more

*This is not (just) a course on building models using libraries like TensorFlow or PyTorch, it is a course on understanding the theory*

# How did I get involved with ML?



# What do you want to learn about ML?

---



# Grade Assessment

---

Component	Weight
Assignments	3 x 10%
Midterm (theory) exam	20%
Journal club	10%
Final project	40%

Bonus marks may be awarded for *substantial* corrections to materials, submitted as pull requests

**Course materials repo:** <https://github.com/mru-comp4630/w26>

**Rendered at:** <https://mru-comp4630.github.io/w26/>

# Textbooks and other readings

---

Primary Textbook:

- [Hands on Machine Learning with Scikit-Learn and \[Tensorflow/PyTorch\]](#)
- [Associated GitHub repo \(Tensorflow\)](#)
- [Associated GitHub repo \(PyTorch\)](#)

More mathy details:

- [Deep Learning](#)

Journal club list: on D2L under "Course Info" (requires MRU library login)

# Generative AI policy

---

- Yes, AI can do a lot of what I'm asking for in this course
- No, I do not want to read about what AI "thinks"
- ? What do you think is an appropriate use?

# Machine Learning Project Checklist

---

[Appendix A of the hands-on textbook](#)

1. **Frame the problem** and look at the big picture.
2. **Get the data.**
3. **Explore the data** to gain insights.
4. **Prepare the data** to better expose the underlying data patterns to Machine Learning algorithms.
5. Explore many different models and short-list the best ones.
6. Fine-tune your models and combine them into a great solution.
7. Present your solution.
8. Launch, monitor, and maintain your system.



# 1. Look at the big picture

---

Example Dataset: California housing prices (1990)

? Discussion questions:

- How does the company expect to use and benefit from this model?
- What is the current solution?
- What kind of ML task is this?
- What kind of performance measure should we use?

## 2. Get the data

---

For this class, we'll use readily available datasets. Some sources are:

- [UCI Machine Learning Repository](#)
- [Kaggle](#)
- [Google Dataset Search](#)
- Various Government open data portals (e.g. [Calgary](#), [Alberta](#), [Canada](#))

After fetching the data, set aside a test set and **don't look at it**.

*"Get the data" can often be a huge task in itself!*

## 2a. Set aside a test set

---

? Discussion questions:

- Why do we need an independent test set?
  - Avoid data snooping bias
  - [Relevant XKCD](#)
- Why would we use a random seed?
- What is naive about simply selecting a random sample?
- What else could we do?
- What is stratified sampling?

# Side tangent: Sampling bias

---

- Simple example: assume 80% of population likes cilantro
- Goal: ensure our sample is representative of the population,  $\pm 5\%$

The [binomial distribution](#) can be used to model the probability of choosing  $k$  people who like cilantro from  $n$  total participants:

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}, \text{ where } \binom{n}{k} = \frac{n!}{k!(n - k)!}$$

# Side tangent: Sampling bias continued

---

$P(X = k)$  is the probability mass function, and the corresponding cumulative distribution function is just the sum up to  $k$ :

$$P(X \leq k) = \sum_{i=0}^k \binom{n}{i} p^i (1-p)^{n-i}$$

Suppose we **randomly** sample 100 people. What is the probability of fewer than 75 or more than 85 cilantro lovers?

*This is also my excuse to review some probability theory and notation*

# 3. Explore the data

---

? Discussion questions:

- What do you notice about the data?
- Do the values make sense for the labels?
- Is the scale of the features comparable? Does this matter?
- What possible biases might be present in the data?

## 3a. Look for correlations

---

The **Pearson correlation coefficient** is a measure of the linear correlation between two variables  $X$  and  $Y$  (commonly denoted as  $r$ ):

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

where  $\bar{x}$  and  $\bar{y}$  are the sample means of  $X$  and  $Y$ , respectively.

- What do correlations of 0, 1, and -1 mean?
- What are some limitations of Pearson correlation?

# 4. Prepare the data

---

General goals:

- Handle missing data, and maybe outliers
- Drop irrelevant features
- Combine features using domain knowledge
- Apply various transformations (e.g. scaling, encoding)
- Apply scaling when necessary



## 4a. Handling missing data

---

In the book 3 options are listed to handle the NaN values:

```
housing.dropna(subset=["total_bedrooms"], inplace=True) ## option 1
housing.drop("total_bedrooms", axis=1)                  ## option 2
median = housing["total_bedrooms"].median()             ## option 3
housing["total_bedrooms"].fillna(median, inplace=True)
```

? Discussion questions:

- What is each option doing?
- What are the pros and cons of each option?
- Which one should we choose?

## 4b. Handling non-numeric data

---

Most of the math in ML algorithms is based on numbers, so we need to convert text and categorical attributes to numbers. This is called **encoding**.

? Discussion questions:

- Which columns of our data are categorical?
- What methods could we use to convert them to numbers?
- What are the assumptions about the various encoding methods?

## 4c. Scaling the data

---

Many ML algorithms don't like features with vastly different scales. Common scaling methods are **min-max scaling** and **standardization**.

*Important: scaling is **computed** on the training set and **applied** to the validation and test sets - they are not scaled independently!*

? Discussions questions:

- What are the bounds of each method?
- Which method is more affected by outliers?
- How would you decide which method to use?

## 4e. Standardization details

---

A general Gaussian distribution is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

where  $\mu$  is the mean and  $\sigma$  is the standard deviation. The standard normal distribution is a special case where  $\mu = 0$  and  $\sigma = 1$ , reducing the equation to:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

## 4f. Other transformations

---

- **Log transformation:** useful for data that is heavily skewed
- Also **square root, squaring, etc.:** try to remove heavy tails
- **Feature engineering:** combining features to create new ones
- **Binning:** turning continuous data into discrete categories
  - Possibly using K-means clustering
  - Relies on domain knowledge
- Best to create a **transformation pipeline** and apply it to the data rather than saving the transformed data