GENERAL ASSEMBLY

PROGRESSING IN YOUR DATA SCIENCE CAREER

Mason Gallo

Data Scientist

LEARNING OBJECTIVES

- Specify common models used within different industries
- Identify the use cases for common models
- Discuss ethical considerations

OPENING

PROGRESSING IN YOUR DATA SCIENCE CAREER

OPENING

- Let’s discuss how to adapt this course to some real-world problems.
- We’ll talk about how to maintain and improve your analyses.
- We’ll also talk about what steps can be taken to make your work “production” ready.
- Lastly, we’ll focus on a few other tools and topics in the data science ecosystem that you should explore in the future!

INTRODUCTION

REAL WORLD MACHINE LEARNING SYSTEMS

INTEGRATING A MODEL INTO A DATA PRODUCT

- As you move into real world projects, it's important to remember that models and analysis are only *one part* of a larger goal or business objective.

- Typically, the model may only answer one of *many* questions that need to be addressed.

- Even within modeling itself, there are many differences between how a model runs during testing vs production.

INTEGRATING A MODEL INTO A DATA PRODUCT

- For example, in a system that will present recommendations, we may have many modeling components that come together.
- Different aspects may categorize content, extract text features, analyze popularity, etc.
- These will all tie back into the final data product.

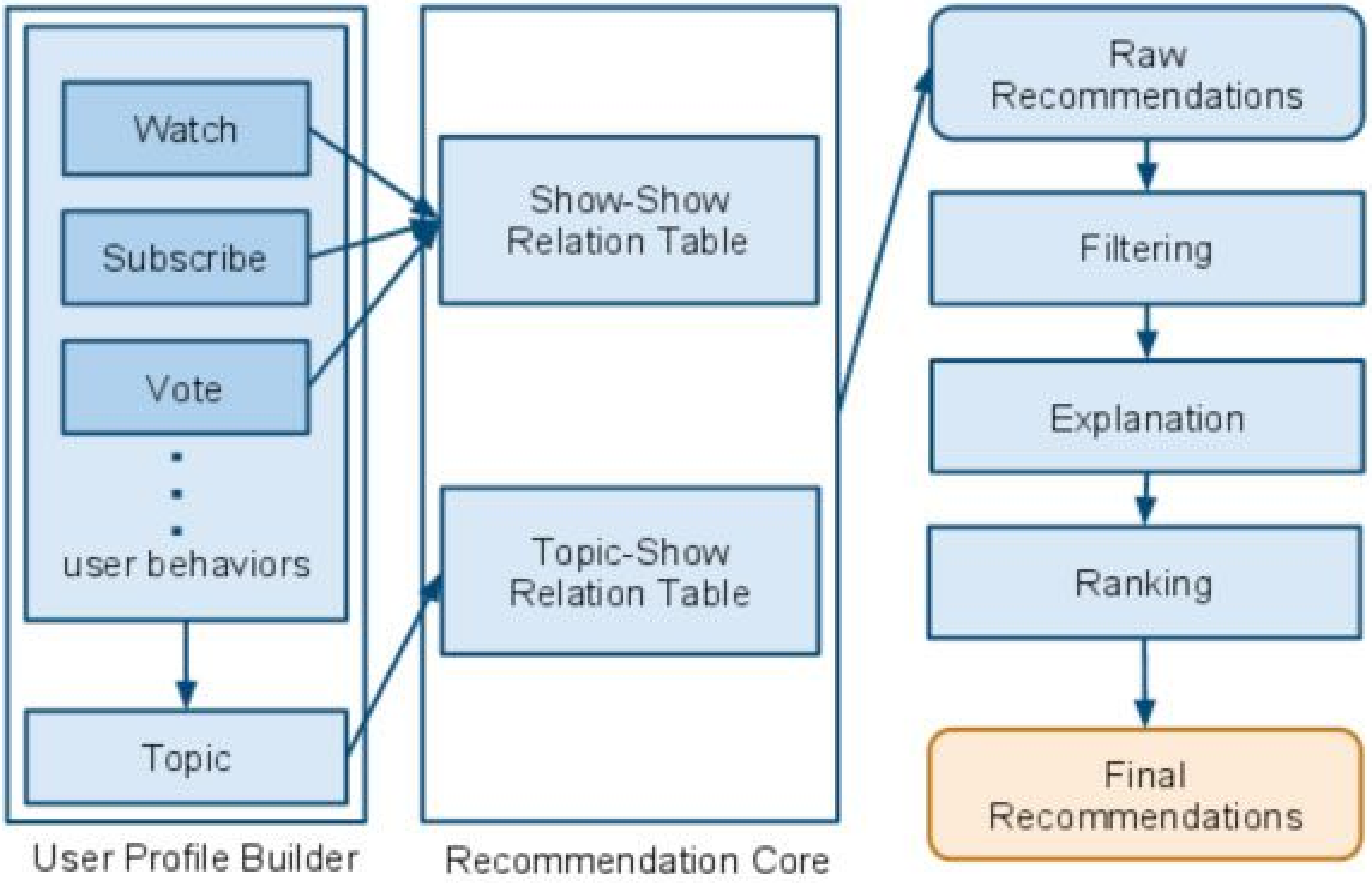
INTEGRATING A MODEL INTO A DATA PRODUCT

- For example, in Hulu's recommendation system, they:
  - Pull data from multiple sources
  - Build user profiles and summaries
  - Then apply a recommendation model

INTEGRATING A MODEL INTO A DATA PRODUCT

- However, this isn't the final step! Additional filters are placed to refine the model in order to ensure the predictions are novel and don't overlap with previous content.

INTEGRATING A MODEL INTO A DATA PRODUCT



INTEGRATING A MODEL INTO A DATA PRODUCT

- Organizing and managing the systems and data dependencies can become an important part of the job.

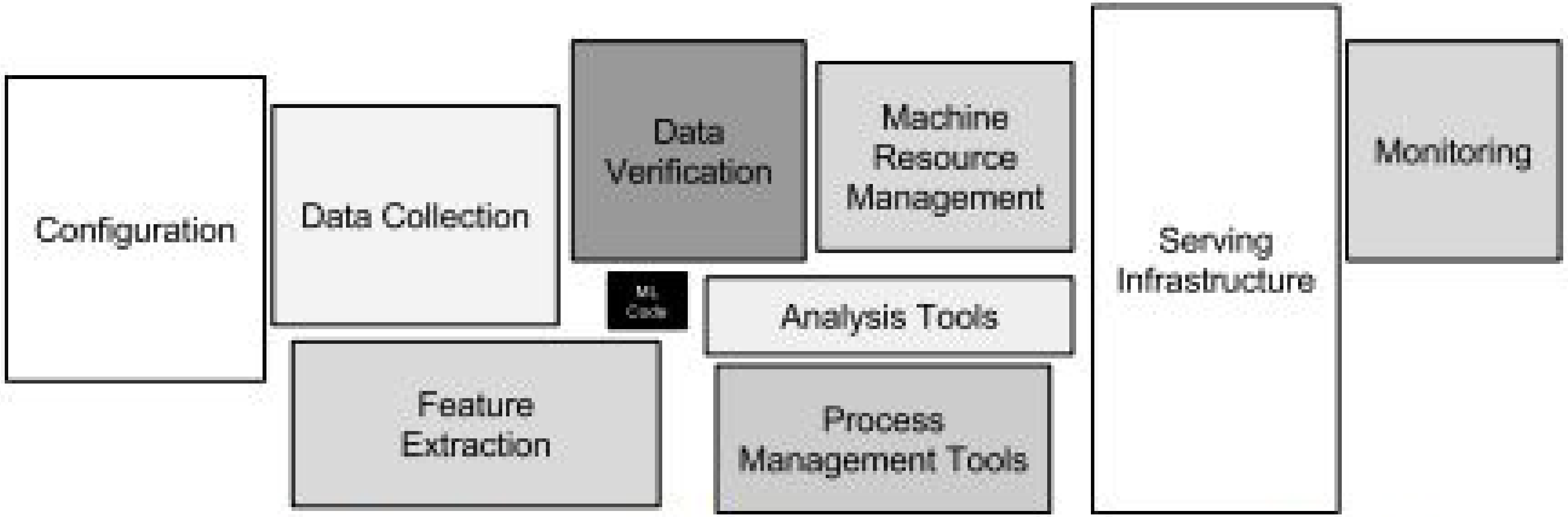


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

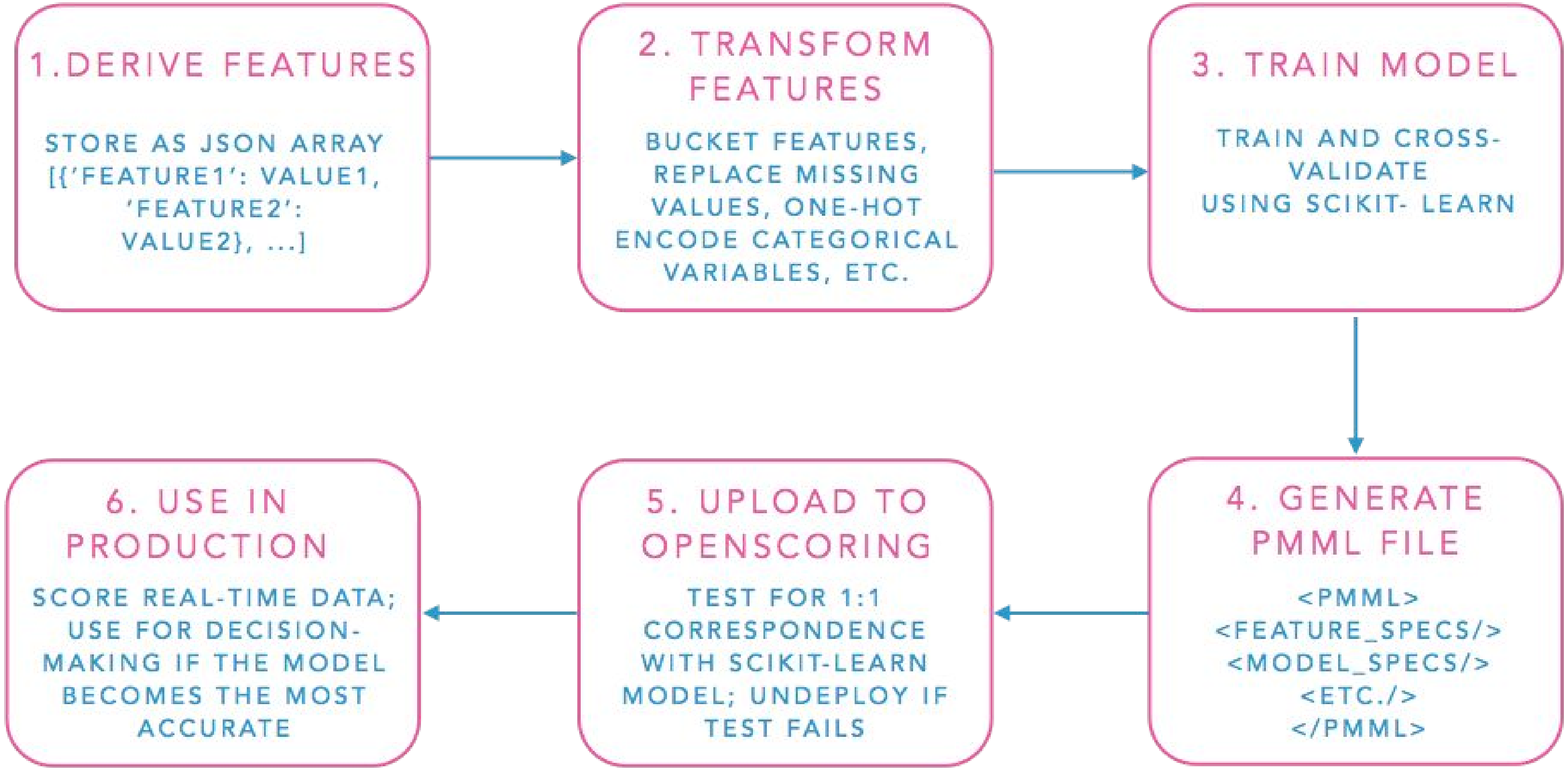
INTEGRATING A MODEL INTO A DATA PRODUCT

- Many organizations rely on data engineering teams to code these common task into pipelines.

- **Data pipelines** are a series of automated data transformations that ensure the validity of your work for routine data maintenance tasks.

INTEGRATING A MODEL INTO A DATA PRODUCT

- Below is a description of the AirBnB model building pipeline.



MODEL MAINTENANCE

- Our class has mostly focused on building an initial model and analysis
- However, once a final model is trained (and we're happy with performance), the model also needs to be **maintained**!
- Plus, as new data is gathered, the model will likely need to be **retrained**.
- Over time, previously predictive features may begin to lose their value, requiring you to investigate once more.

MODEL MAINTENANCE

- Google addresses this phenomenon, describing the “Technical Debt” of machine learning systems in a paper called: "Machine Learning: The High Interest Credit Card of Technical Debt"
- They focus on a few core issues that affect model maintenance:
    - Code complexity
    - Evolving features
    - Monitoring and testing

CODE COMPLEXITY

- Most of the code for our class has been written in notebooks.
- However, as your analysis and models develop, you are likely to revise and reuse parts of this code.
- Improving the quality of your code can simplify this process!
- This isn't always the responsibility of data scientists, but keep in mind - *more clarity in your code will lead to more clarity in your analysis*.
- This is especially true for long term or open source projects where your code has to make sense to other people (or yourself) in the future!

CODE COMPLEXITY

- One way to write better code is to create (and follow!) a *styleguide*.
- A *styleguide* is a clear set of rules for organizing your code.
- Columbia recently developed a special styleguide just for data scientists.
- Some rules are pretty straightforward:
    - Give variables, methods, and attributes descriptive names.
    - Write functions that take well-defined inputs and produce well-defined outputs.

CODE COMPLEXITY

- Another common practice is *unit testing*.
- Unit testing involves writing short statements that *test* if a piece of code or function is working correctly.
- Typically, these tests provide a few sample inputs and outputs and then check that your code can produce the same outputs.
- According to Google, “ensuring that code has been tested is vital to ensuring that your analysis results are correct.”

CODE COMPLEXITY

- Suppose we have the following function that calculates the area of a circle.

```python
def calculate_area_of_circle(radius):
    # Use value of pi
    pi = 3.14
    area = pi * radius ** 2
    return area
```

CODE COMPLEXITY

- A unit test for this may look like the following.

```python
def test_calculate_area_of_circle():
    # Various test cases
    assert calculate_area_of_circle(2) == 12.56
    assert calculate_area_of_circle(4) == 50.24
    assert calculate_area_of_circle(0) == 0.0
```

CODE COMPLEXITY

- On long term or big data projects, the code supporting a machine learning model can get complex.
- This "glue code" holds the model together, but it can get weighed down with bloat and feature creep over time.
- Thus, this code often needs to be *refactored* in order to trim the fat.

CODE COMPLEXITY

- Google describes the need to review your code, stating that:

“Only a tiny fraction of the code in many machine learning systems is actually doing "machine learning"”

“Without care, the resulting system for preparing data in an ML-friendly format may become a jungle of scrapes, joins, and sampling steps, often with intermediate files output.”

“Managing these pipelines, detecting errors, and recovering from failures are all difficult and costly."

CODE COMPLEXITY

- Creating and following a clear **styleguide** as well as **testing** and **refactoring** your code will help maintain your machine learning algorithm over time.

- Plus, reducing *technical debt* saves time and money in the long term!

- Even Google is not immune:

> "In a recent cleanup effort of an important machine learning system at Google, it was possible to rip out tens of thousands of lines of unused experimental code-paths!"

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

ANSWER THE FOLLOWING QUESTIONS

1. Take a look at the following code which parses an apartment description for the apartment's square footage. What corner cases would it fail?

```
def extract_sqft(apt_description):
    # Split the text on spaces
    words = apt_description.split(' ')
    for (i, word) in enumerate(words):
        # Look for "sqft"
        if word == 'sqft':
            # Select the word before sqft
            return int(words[i-1])
        else:
            return np.nan
```

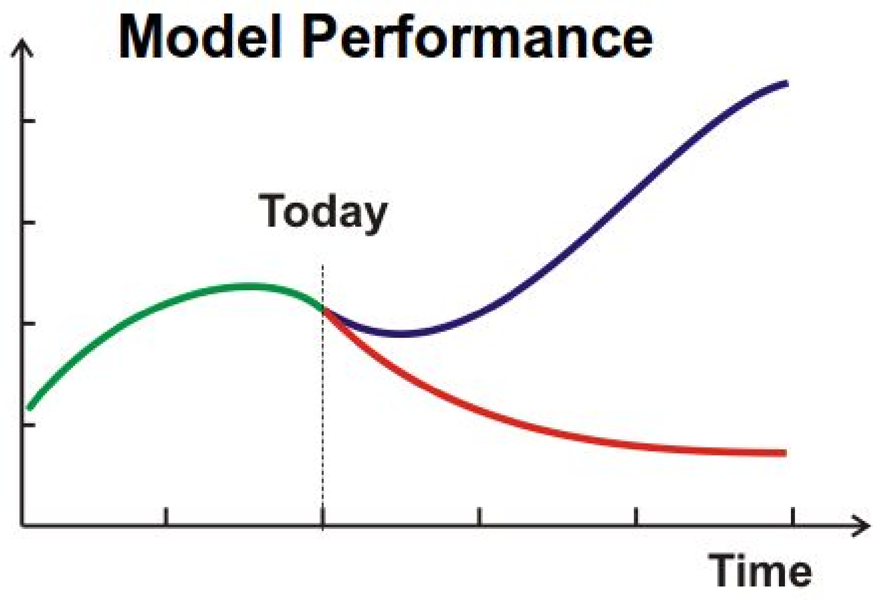
DELIVERABLE

Answers to the above questions

EVOLVING FEATURES

- Once your model is trained, it's important to track its performance over time.

- Many of the correlations found or features predicted may not remain true in a few months or years into the future.



EVOLVING FEATURES

- For example, our “evergreen” article prediction example looks for food mention to predict the popularity of certain recipes.

- However, it doesn’t know how to gauge trends in popular foods. Over time, it will return a smaller and smaller sample unless we readjust the model’s parameters.

- As one trend takes off, the model trained on old trends may not be able to pick that up.

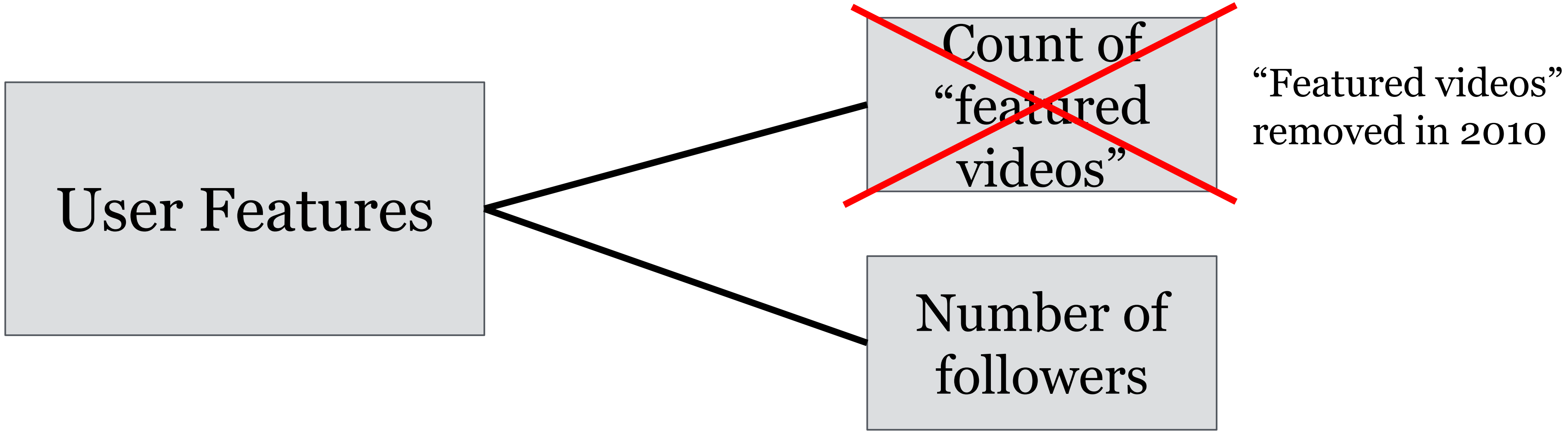
EVOLVING FEATURES

- Google's technical debt paper groups troublesome features into two groups: *legacy features* and *bundled features*.
  - **Legacy features**: When a feature "F" is included in a model early on, but later other features are added that make F redundant.
  - **Bundled features**: When a group of features are all bundled together, it can be hard to differentiate the features that aren't performing well from the ones that are.

EVOLVING FEATURES

- Features can also be "bundled" with commonly occurring variables, but those variable occurrences may change over time, making the features obsolete.

- For instance, we may have features a Youtube user that are no longer tracked or relevant. These may be bundled with other "user features".



EVOLVING FEATURES

- From Google’s paper:

"Machine learning systems often have a difficult time distinguishing the impact of correlated features.

This may not seem like a major problem: if two features are always correlated, but only one is truly causal, it may still seem okay to ascribe credit to both and rely on their observed co-occurrence.

However, if the world suddenly stops making these features co-occur, prediction behavior may change significantly."

EVOLVING FEATURES

- Changing variables is especially important for *black box models*.
- Such models rely on correlations from a wide range of features. However, in doing so, we can typically ignore one of two variables that are highly correlated.
- If these variables are no longer correlated, we may need to update this.

EVOLVING FEATURES

- Another common way for features to evolve is through *feedback loops*.
- Once you've performed an analysis and built your model, it's likely you will make decisions and take actions based on your findings.
- It's important to think about how these actions may change the data you are using for future analysis.
- Are you introducing bias to your data and model?

EVOLVING FEATURES

- For example, imagine we are investigating ways to reduce the spread of infections in hospitals:
  - We may find that whenever a doctor sees more than 5 patients in an hour, those patients have a greater risk for infection.
  - Based upon this, we implement a policy that doctors cannot see more than 5 patients in one hour.
  - If we perform our same analysis a year after this policy is enacted, the feature “saw >5 patients in an hour” won’t exist!

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

ANSWER THE FOLLOWING QUESTIONS

1. Brainstorm two correlated features from our prior work in this class that may not be correlated in the future. Preferably from your final projects or from your work!

DELIVERABLE

Answers to the above questions

MONITORING MODELS

- Once a model is deployed and making predictions, it's important to track its performance.

- From the Google again!

> "Unit testing of individual components and end-to-end tests of running systems are valuable, but in the face of a changing world such tests are not sufficient to provide evidence that a system is working as intended.
>
> Live monitoring of system behavior in real time is critical."

MONITORING MODELS

- One common monitoring technique is to compare your model's performance to a baseline.
- The baseline can be something simple, like a naive model that only predicts the average or most frequently occurring value.
- When monitoring a model, you can update your baseline as information becomes available.
- Your "better" model should always outperform the baseline!

ETHICAL CONSIDERATIONS

- Another (often overlooked) aspect of managing real world data science projects are *ethical considerations*.
- It's important to understand the biases of your data and how this influenced our analysis and models.
- Two common examples are criminal justice and financial loans applications.

ETHICAL CONSIDERATIONS

- When analyzing crime, we often want to consider what drives criminal activity and what actions might reduce it.

- However, it's important to consider how our data is collected. For example, current data is based off the current criminal justice system.

- It can be difficult to separate the biases of the current system from the correlations/predictions that you are trying to make in your model.

- If data from the current justice system overweighs specific concerns or attributes, your model will too.

ETHICAL CONSIDERATIONS

- Similarly, data from financial lenders may be biased, as their goal is to find borrowers who are most likely to pay back in a timely fashion.

- These analyses need to be strongly regulated so that protected factors (e.g. race, gender, etc) are not considered. However, they can still leak in through proxy features.

- Proxy features are not protected per se, and are strongly correlated with specific protected features.

- For example, neighborhood zip code can be used as a proxy for race.

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

ANSWER THE FOLLOWING QUESTIONS

1. In small groups, discuss other areas of possible ethical issues in Data Science (perhaps from your work domain?)
   a. How might this occur when examining health data?
   b. What about when examining educational records?

DELIVERABLE

Answers to the above questions

BEFORE NEXT CLASS

HOMEWORK

- Think about what areas in this class you enjoyed the most
- We’ll talk about specializations and tools in data science next class

LESSON

Q & A

LESSON

EXIT TICKET

DON'T FORGET TO FILL OUT YOUR EXIT TICKET