

# **5. Feature Engineering**


## **Machine Learning Operations**

# Previously...

# Previously...


### Training Data is the bottleneck

**Data**




Data is messy, complex, unpredictable, and a key differentiator.

**Algorithms**



Increasingly commoditized.

**ML Model**



```
openai/gpt-oss  
git clone https://github.com/openai/gpt-oss  
cd gpt-oss  
from transformers import AutoTokenizer, AutoModelForCausalLM  
tokenizer = AutoTokenizer.from_pretrained("openai/gpt-oss")  
model = AutoModelForCausalLM.from_pretrained("openai/gpt-oss")
```

ie UNIVERSITY

### Labeling


**Hand-Labeling: Still the Backbone**

Label multiplicity: experts not always aligned!

More data isn't always better if the labeling is not accurate

**Hand Labels**

- Natural Labels
- Programmatic Labels
- Handling the Lack of Labels



Labeling Instructions: "Draw bounding boxes around the iguanas"

ie UNIVERSITY

### Sampling

**Random sampling**

- Simple random sample
- Systematic sample
- Stratified sample
- Cluster sample

Non-probability sampling

▶ Random sampling

Machine learning data is chosen with equal probability for all items.

Ensures fair representation

May miss rare categories

amazon

Survey customer satisfaction:


- Divide customers into groups by region (e.g., North America, Europe, Asia) and sample equally from each.
- Ensures representation of all regions.

ie UNIVERSITY

### Class Imbalance

The Solutions: Resampling Techniques

**Over sampling**



Balances the dataset by increasing the number of examples in the minority class.

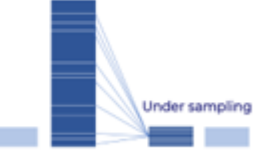
- Duplicating existing rare samples
- Creating new synthetic similar samples

Best for small datasets where every point matters.

Example: Rare disease detection with few patients.

Risk: Overfitting & longer training.

**Under sampling**



Balances the data by reducing the number of examples in the majority class.

- Simpler, BUT this approach risks removing important information that the model could have learned from the discarded samples


Best for large datasets to cut training time & cost.

Example: Fraud detection in millions of transactions.

Risk: Information loss.

ie UNIVERSITY

### Data Augmentation



ie UNIVERSITY





# AGENDA

**1. Features**

**2. Categorical Encoding**

 **3. Hands-on 2: Data Sampling & Feature Engineering**

# Features

**Once we have the data... how do we turn it into useful features that make our model smart?**



# Features



# Is this the same person?



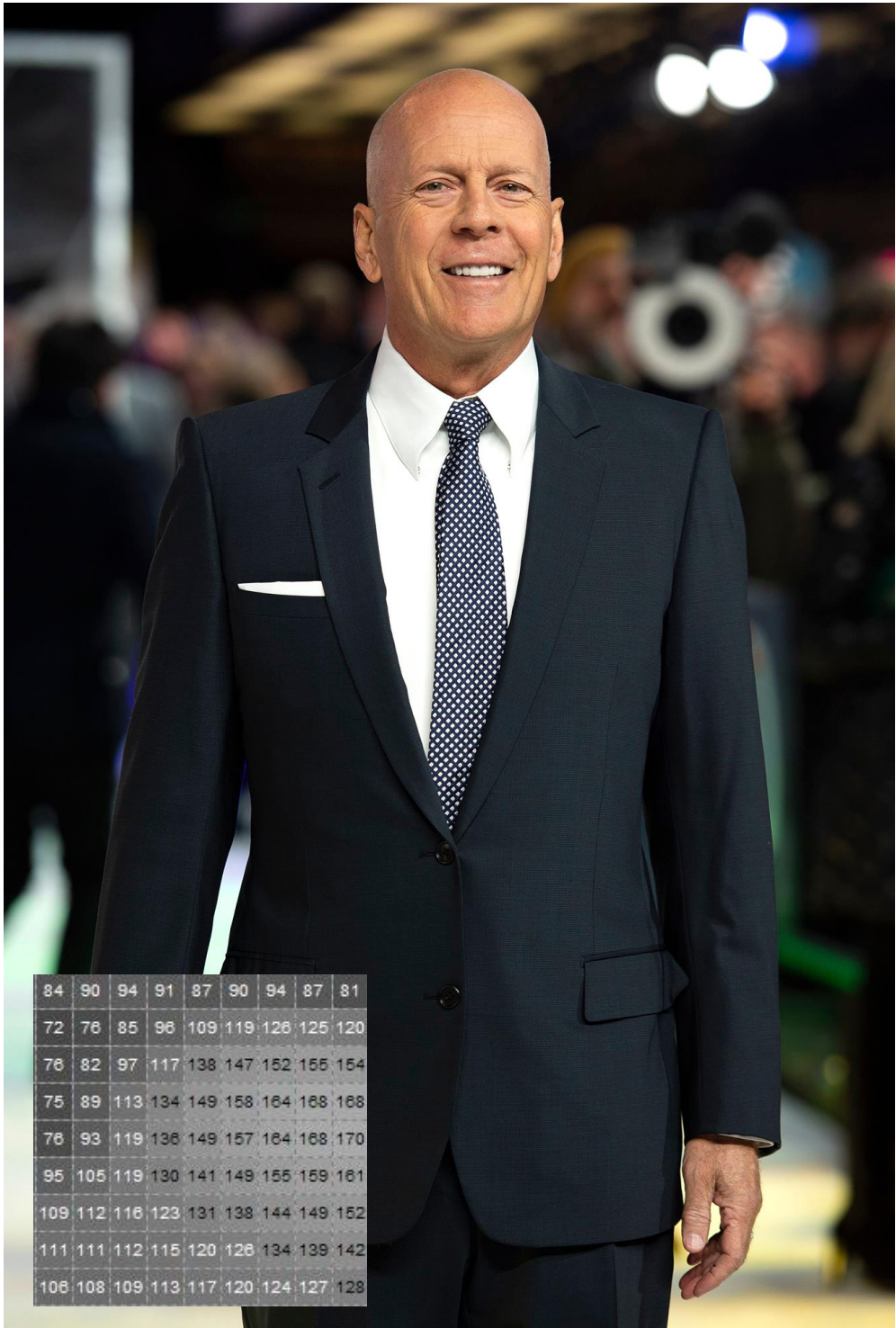


# Is this the same person?





# Is this the same person?

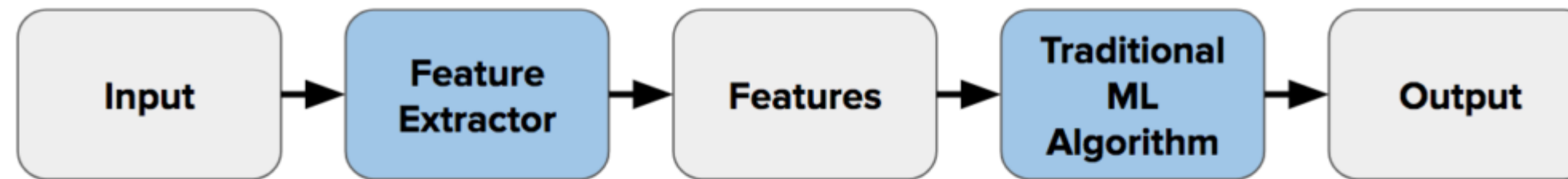


# Features

## Raw data

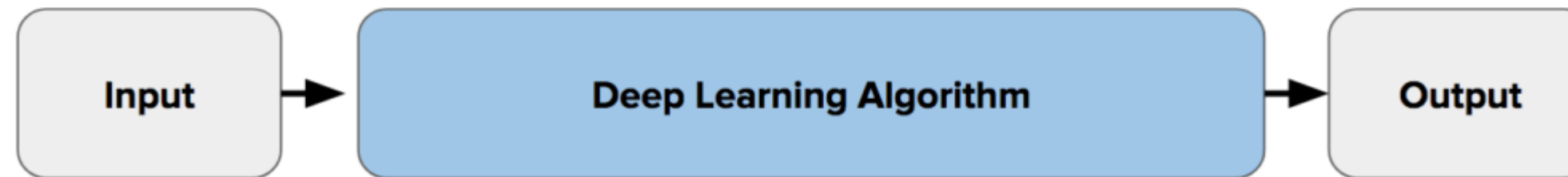


- Hand-crafted using domain knowledge: hair? 0/1
- More interpretable
- Part “art,” part “engineering”



Traditional Machine Learning Flow

**Engineered features**  
(by humans)



Deep Learning Flow

**Learned features**  
(by the model)

- Automatically extracted representations from data
- Often less interpretable; data and compute hungry
- Great when you have lots of labeled data



# Categorical Encoding

# Features

What do you think the model infers when it sees  $3 > 2 > 1$ ?



**Linear/logistic models:**

The weight for “3” is forced to be ‘bigger’ than “1”.

**K-NN:**

$\text{distance}(3,2) < \text{distance}(3,1) \rightarrow$  ‘chocolate’ is ‘closer’ to ‘strawberry’.

**Trees:**

splits like  $\leq 2$  imply vanilla+strawberry vs chocolate.



# Fix: One-hot encoding



Independent binary flags -> No fake orders

flavor	vainilla	chocolate	strawberry
vanilla	1	0	0
chocolate	0	1	0
strawberry	0	0	1

What if a new flavor appears at inference, like 'Mint'?"



Send it to **UNKNOWN** (an extra bucket) and always log unseen categories.

# One-hot encoding

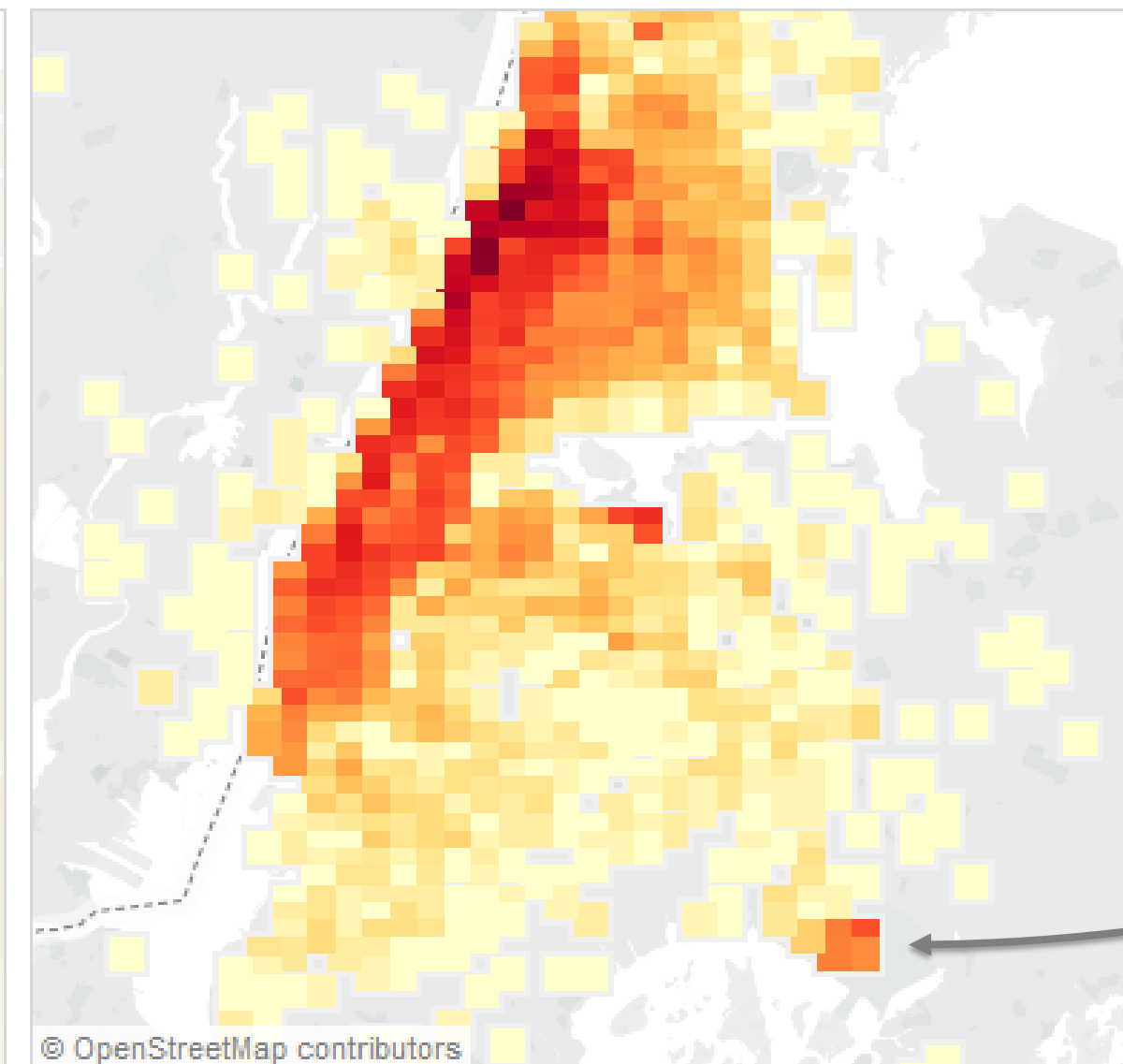
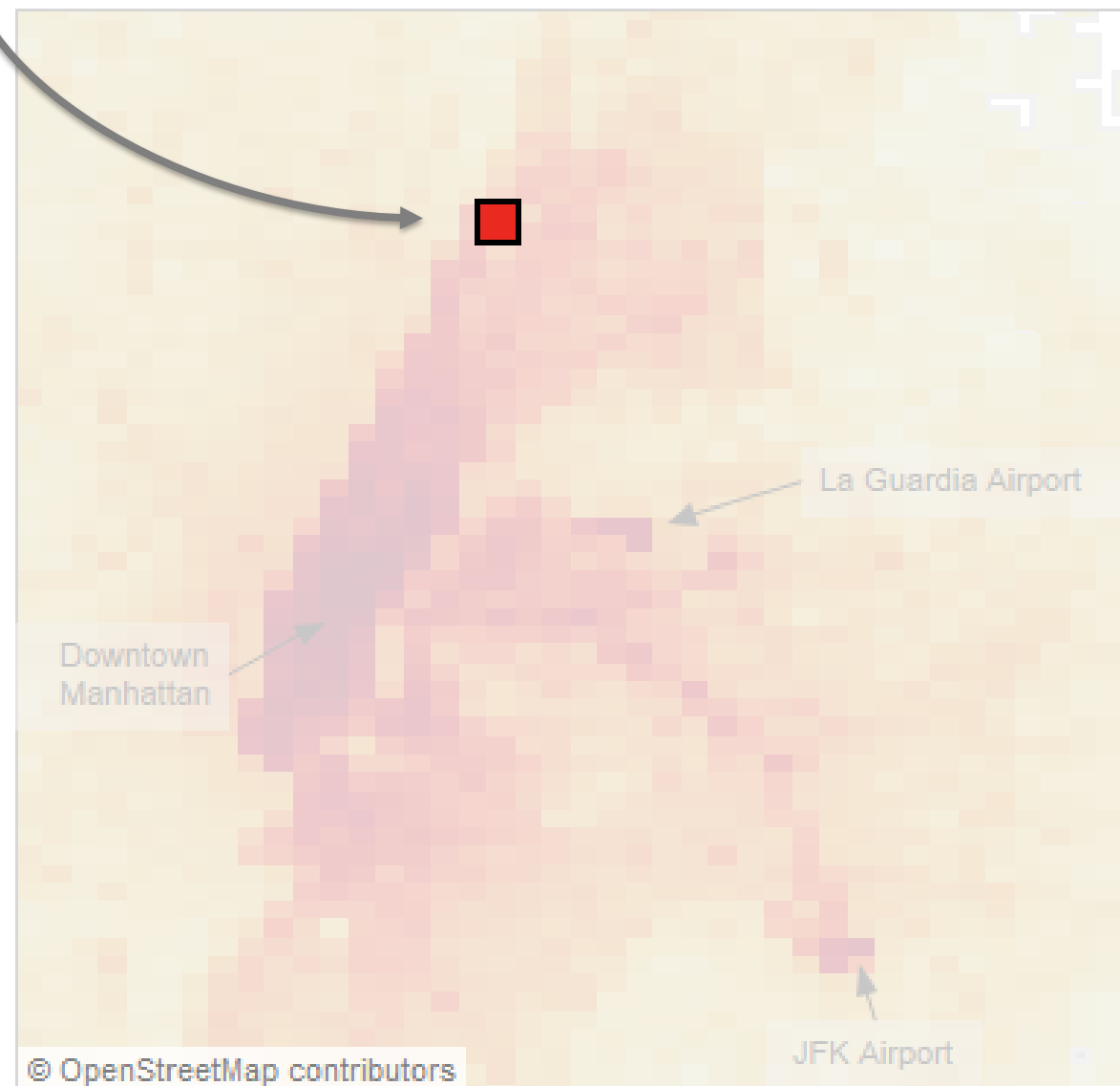


Pick Up Location  
Zone 236

## Where do people go from where in NYC?

Click on pickup locations on the left to show

the corresponding dropoff locations on the right



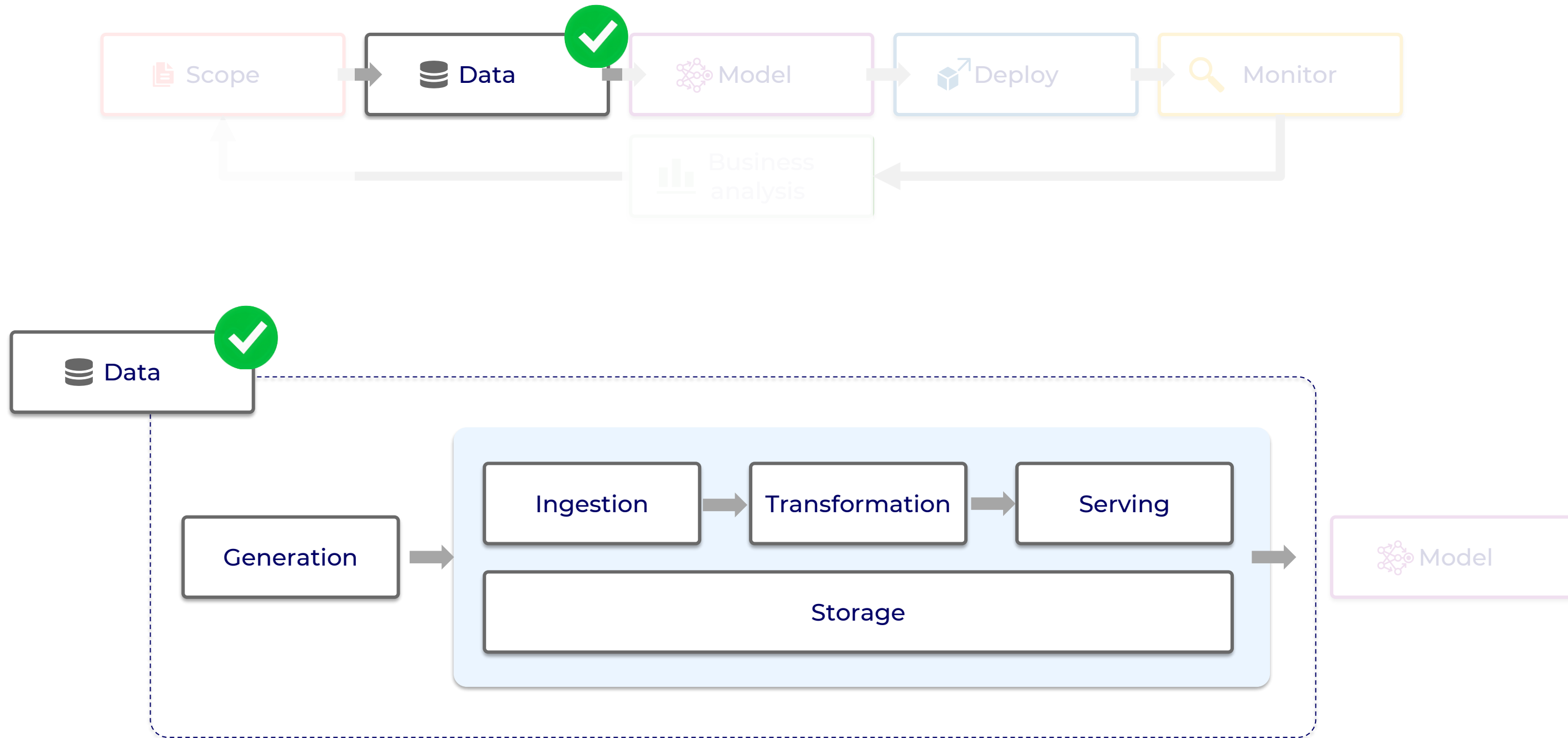
Drop-off Location  
Zone 142

Whenever a category has **no natural order**, giving it numbers **lies to the model**;  
**one-hot** tells the truth.



# Data Engineering → Training Data

The infrastructure that moves and transforms our data.





# Hands-on 2: Data Sampling & Feature Engineering



# Hands-On Roadmap



1. Initial Notebook

3. Experiment Tracking

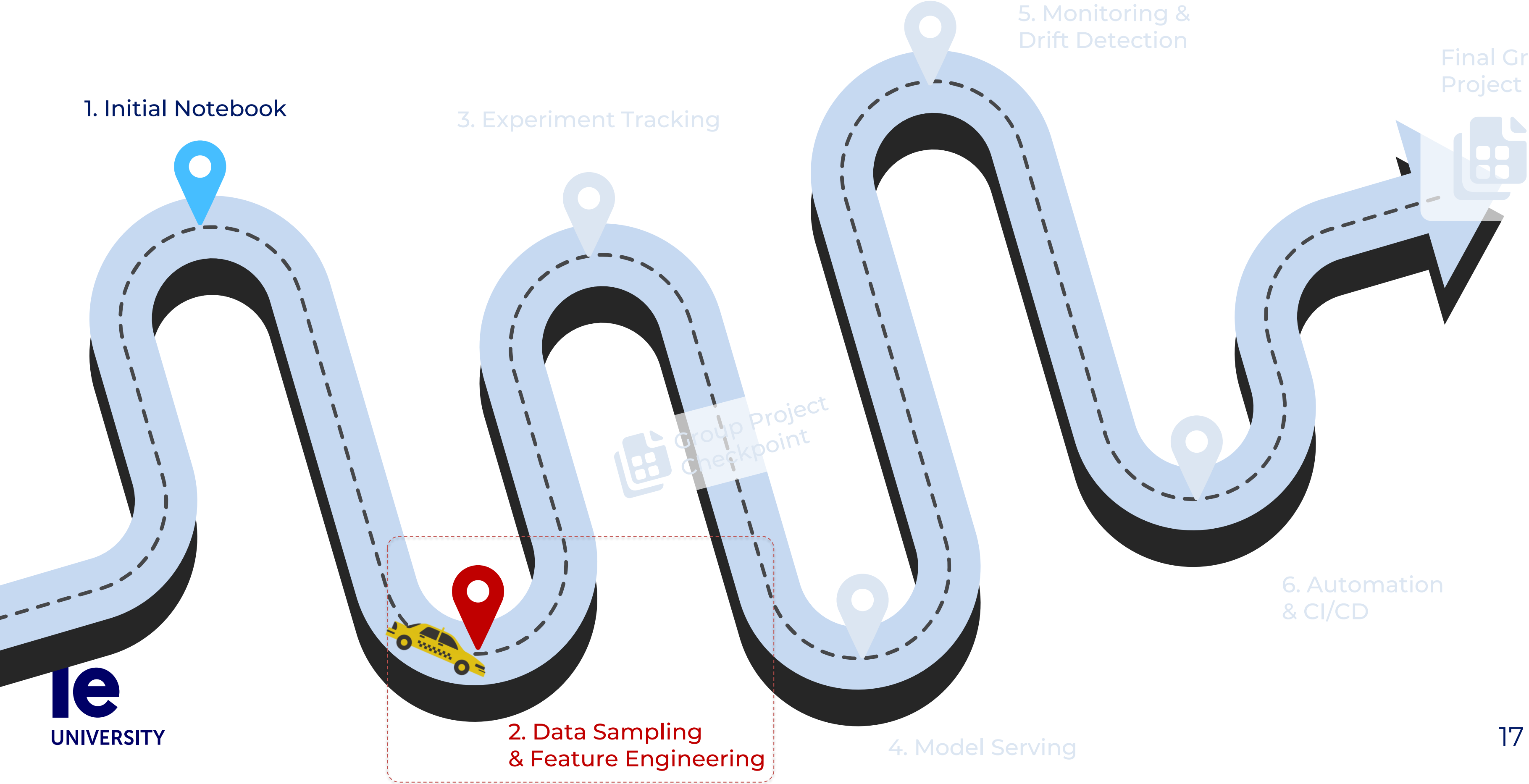
5. Monitoring & Drift Detection

Final Group Project

2. Data Sampling & Feature Engineering

4. Model Serving

6. Automation & CI/CD



# What's what in the repo


## Special folders

**.github/** – GitHub Actions workflows (CI). Anything here runs on each push/PR. 

**.pytest\_cache/** – local cache created by pytest (shouldn't be committed).

**.venv/** – your local Python virtual environment (private to your machine; never commit).

## Top-level folders:

Each folder = one topic/module of the course. Keeps notebooks, scripts, and small datasets scoped so teams don't step on each other. 

## Key files

**requirements.txt** – the list of Python packages pinned to exact versions so everyone installs the same stack.

**.gitignore** – tells Git what not to track (e.g., .venv, data dumps, caches).

**README.md** – the front door of your project: what it does, how to run it, how to contribute.

**render.yaml** – infra-as-config for your chosen host. 

**.flake8** – linter rules so the Python style is consistent.


## ▼ IE-MLOPS-NYC-TAXIS

> .github


> .pytest\_cache

> .venv

▼ 01-initial-notebook

 nyc\_taxi\_duration\_prediction.ipynb

▼ 02-data-sampling-features

 nyc\_taxi\_duration\_prediction\_data\_fe.ipynb


> 03-experiment-tracking


> 04-deployment

> 05-monitoring

> 06-cicd

≡ .flake8

 .gitignore

 README.md

! render.yaml

≡ requirements.txt



# Hands-on 2

## Data Sampling & Feature Engineering



### Goals:

1. Speed up iteration with a small 10% reproducible **sample**.
2. Make **categorical features** model-ready without fake order.
3. **Compare** metrics vs Hands-on 1 baseline.





# Hands-on 2



LAB



# Creating our GitHub Repo

**Step 1: Create your project folder**  
ie-mlops-nyc-taxis

**Step 2: Create and activate a virtual environment**

macOS / Linux

```
bash

python3 -m venv .venv
source .venv/bin/activate
```

Windows (PowerShell)

```
powershell

py -3 -m venv .venv
.\.venv\Scripts\Activate.ps1
```

👉 You should see (.venv) in your prompt.

**Step 3: Download requirements.txt from blackboard**

# Creating our GitHub Repo

## Step 4: Install dependencies

```
pip install -r requirements.txt
```

## Step 5: Create a new empty repo on github.com ie-mlops-nyc-taxi

## Step 6: Initialize Git and commit

```
git init  
git add .  
git commit -m "Initial commit"
```

Optional `.gitignore`:

```
markdown  
  
.venv/  
__pycache__/  
.ipynb_checkpoints/
```



# Creating our GitHub Repo

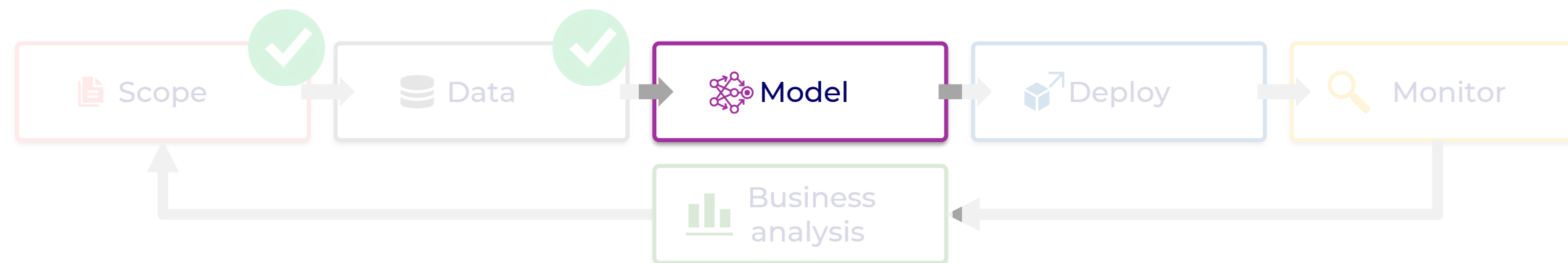
## Step 7: Push to GitHub

```
git branch -M main  
git remote add origin https://github.com/<user>/ie-mlops-nyc-taxis.git  
git push -u origin main
```

 Your first project is live — check GitHub!

# ML Iterative Loop

Model: Quick tour of data plumbing



# Next session

**How do we build and track models using these features?**



To be continued...