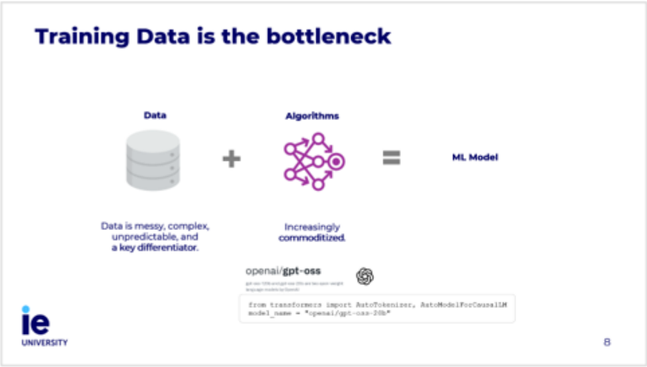
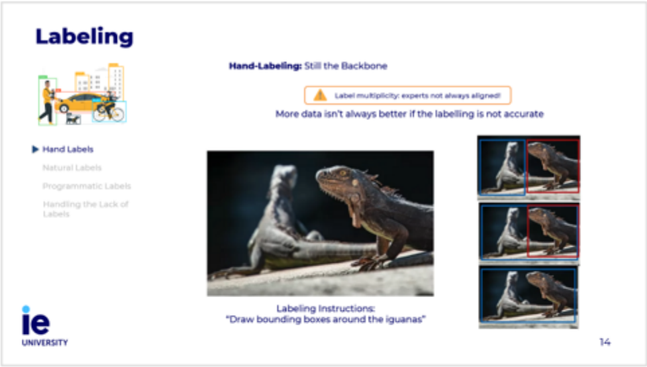
5. Feature Engineering

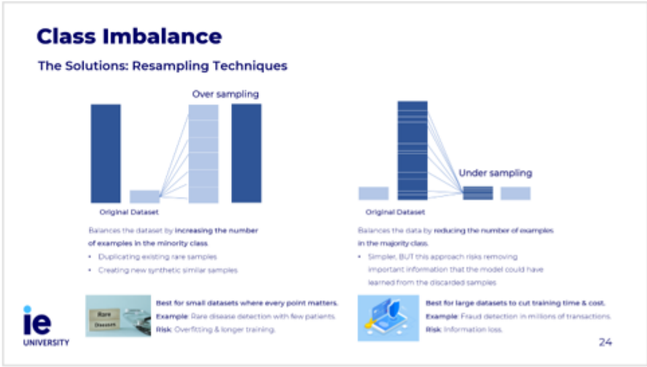
Machine Learning Operations

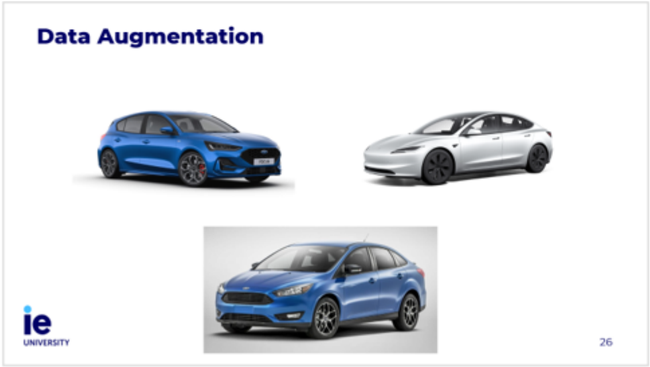
Previously…

Previously...









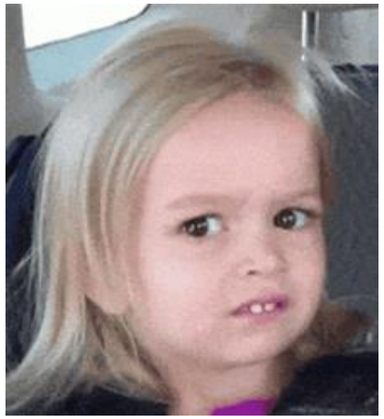
ie UNIVERSITY

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?

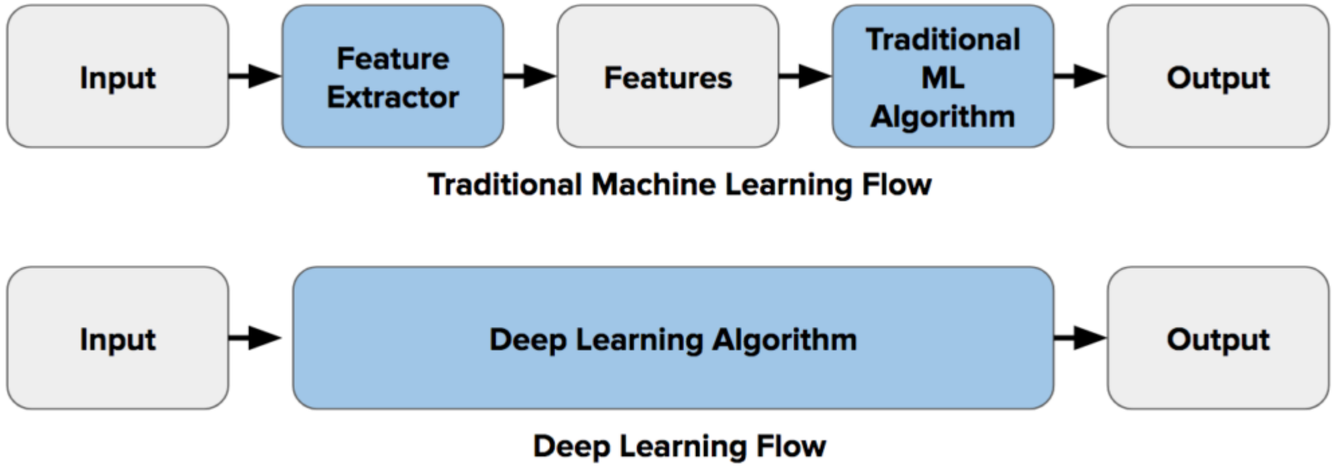
Is this the same person?

Is this the same person?

Features

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

Raw data



Engineered features
(by humans)

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?

Vanilla 1

Chocolate 2

Strawberry 3

Linear/logistic models:
The weight for “3” is forced to be ‘bigger’ than “1”.

K-NN:
distance(3,2) < distance(3,1) → ‘chocolate’ is ‘closer’ to ‘strawberry’.

Trees:
splits like <= 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

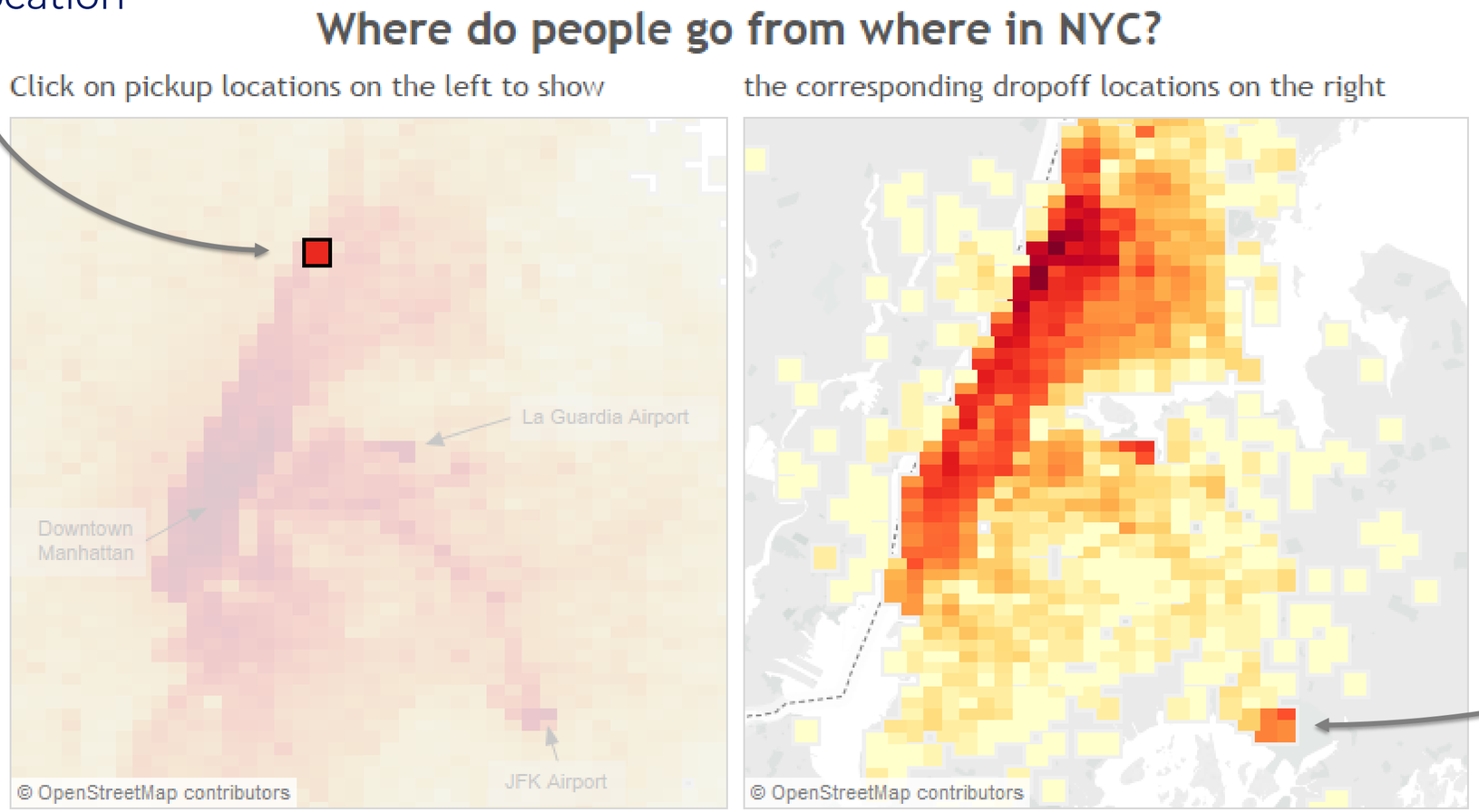
What if a new flavor appears at inference, like '**Mint**'?"

➡

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

One-hot encoding

Pick Up Location
Zone 236

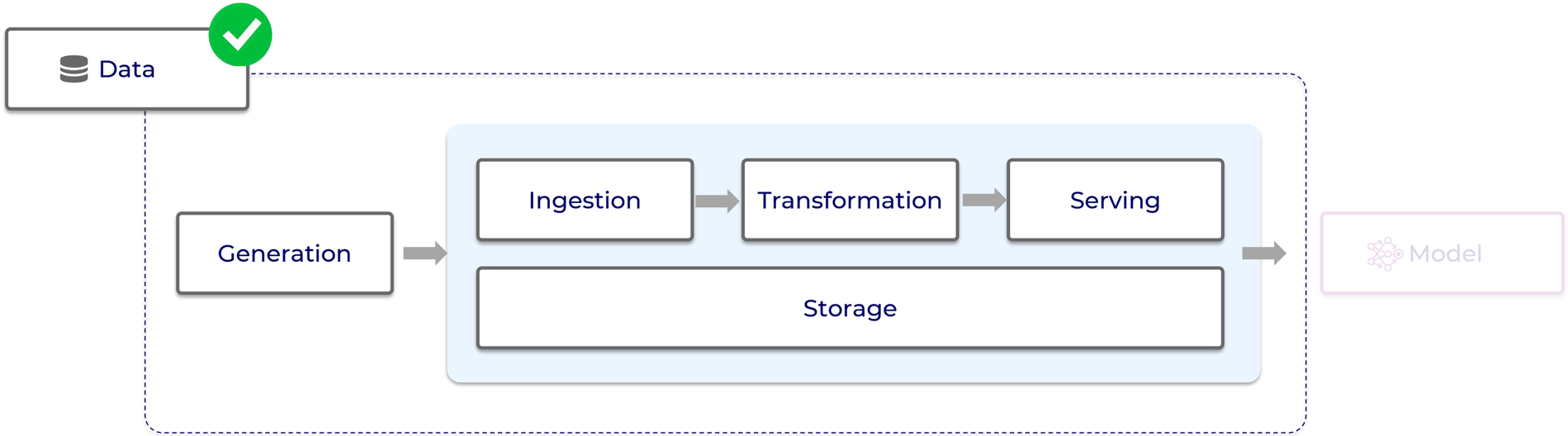


Drof-off Location
Zone 142

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

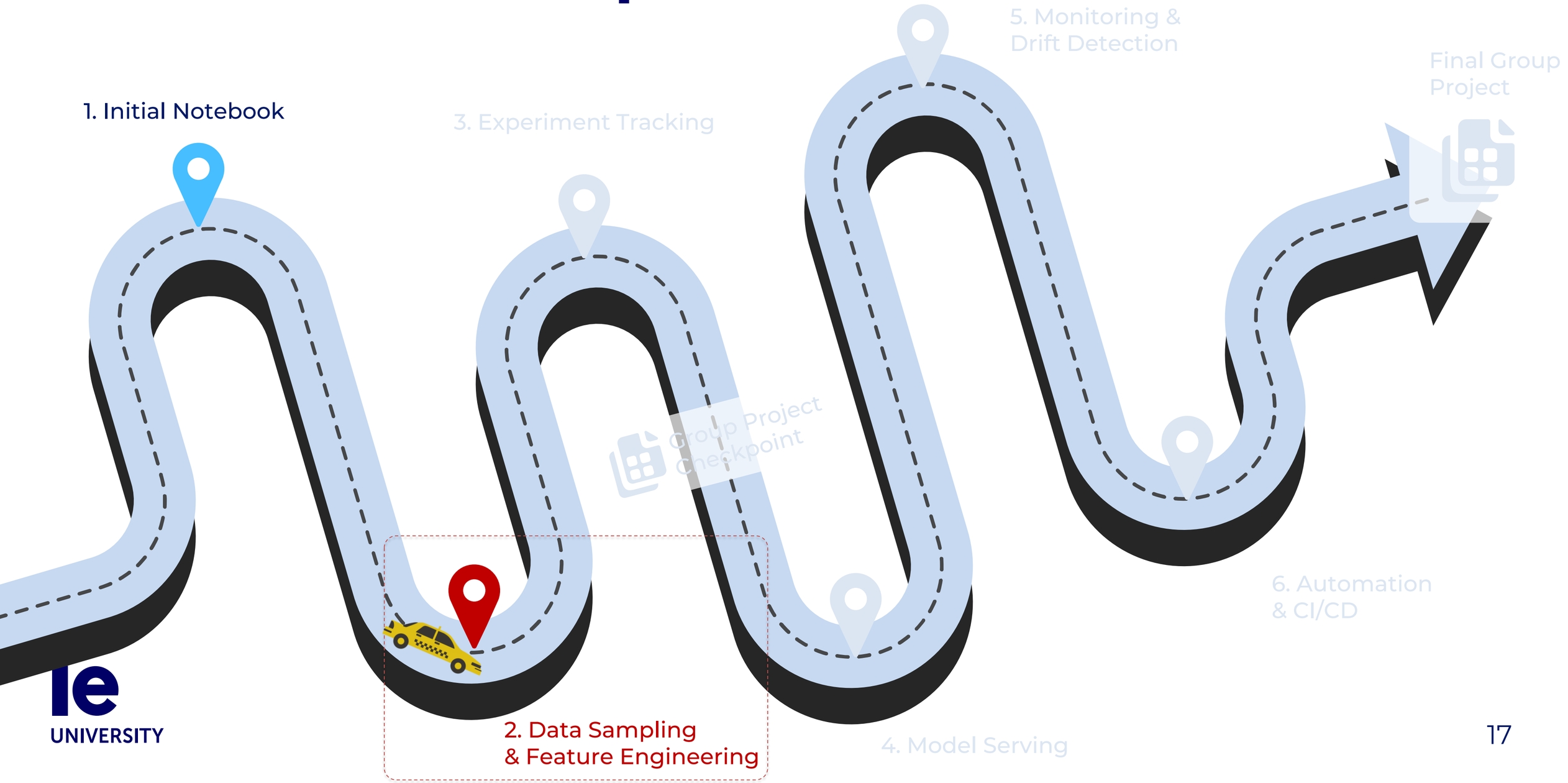
Data Engineering → Training Data

The infrastructure that moves and transforms our data.



Hands-on 2: Data Sampling & Feature Engineering

Hands-On Roadmap

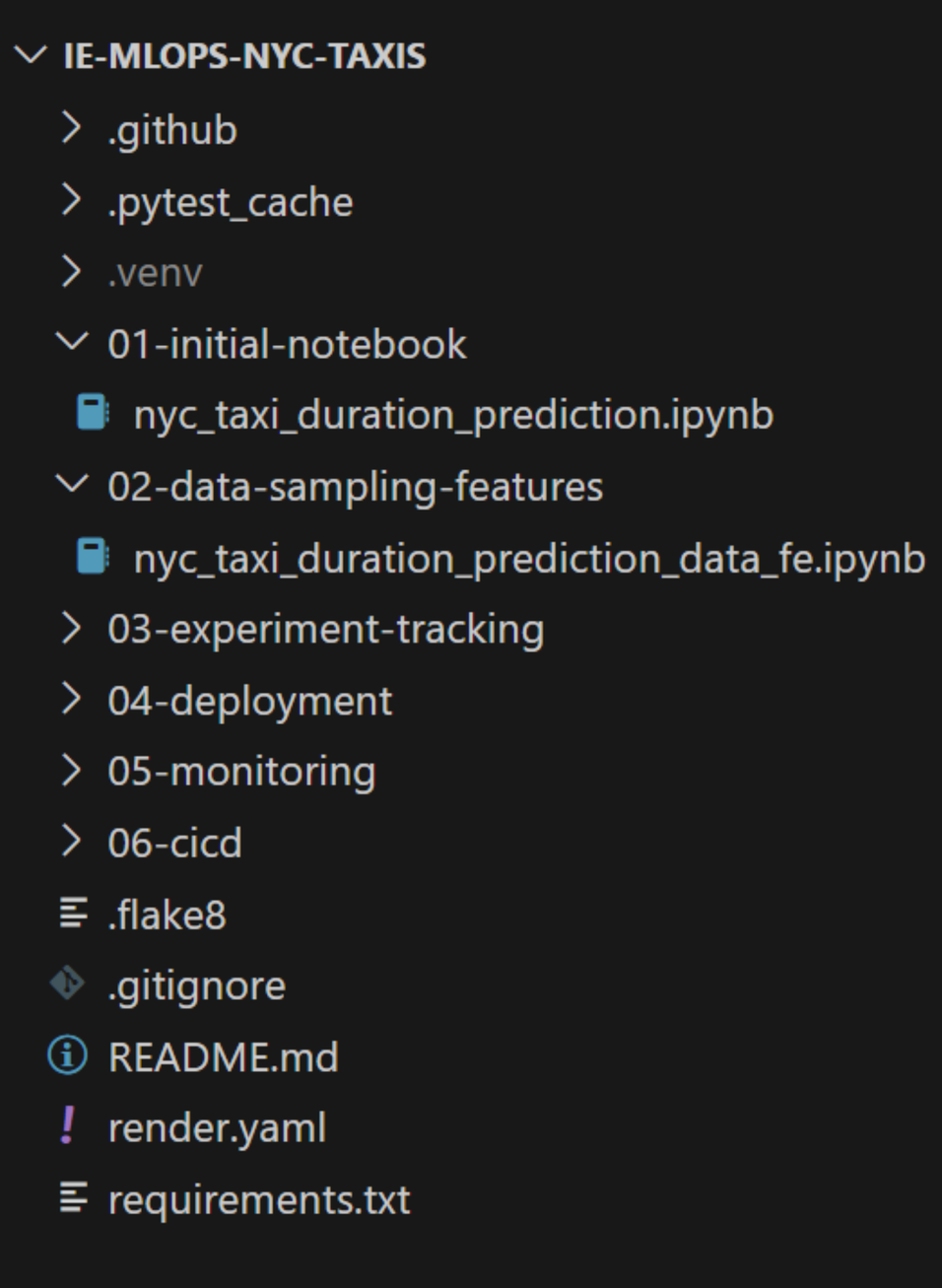


What's what in the repo

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 UNIVERSITY

Hands-on 2

Data Sampling & Feature Engineering

jupyter

scikit learn

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.

ie UNIVERSITY

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-taxis

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

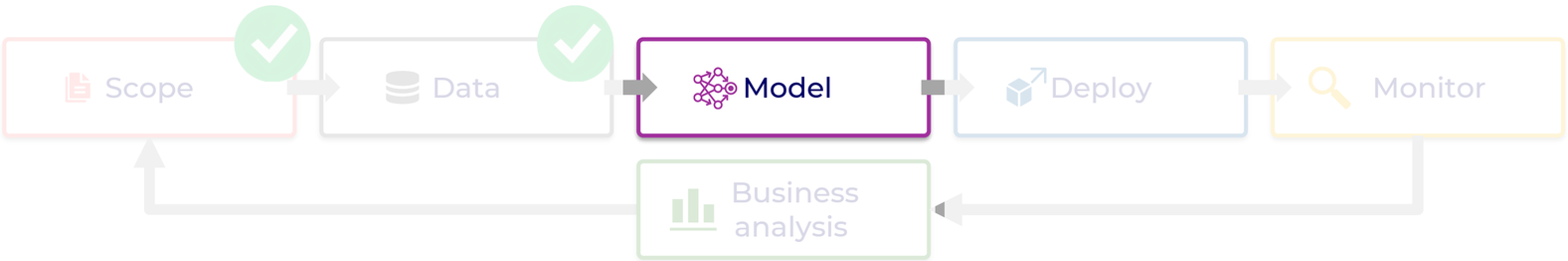
Step 7: Push to GitHub

```
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...