Building a Data Science App

February 22
Park University
Micah Melling

Micah Melling

Current: Chief Data Scientist at Americo Financial

Former: Senior Director of Data Science at Spring Venture Group

Bachelors in Economics from UCM

Professional Certificate in Data Science from Georgetown

Masters in Data Science from Rockhurst

UMKC Analytics Advisory Board

Park University Adjunct Faculty

Why We Are Here

Build a production-ready machine learning application, deploy it on AWS, and learn data science tips and tricks along the way.

Agenda

1. Get set up.
2. Walk through the data science pipeline with a real project.
3. Deploy!

Learning Outcomes

1. Understand the components of a production ML application.
2. Get experience with putting a model in the cloud and making it accessible.
3. Learn data science tips and tricks along the way.

Is this “Real World”?

This work will be close to a real-world example, with two exceptions.

1. To fit into a workshop timeframe, the data and modeling problem are fairly “friendly”.
2. We would want more security around our AWS deployment.

That said, everything else about this effort is pretty “real world”.

Why Can’t We Just Let GenAI Do All This for Us?

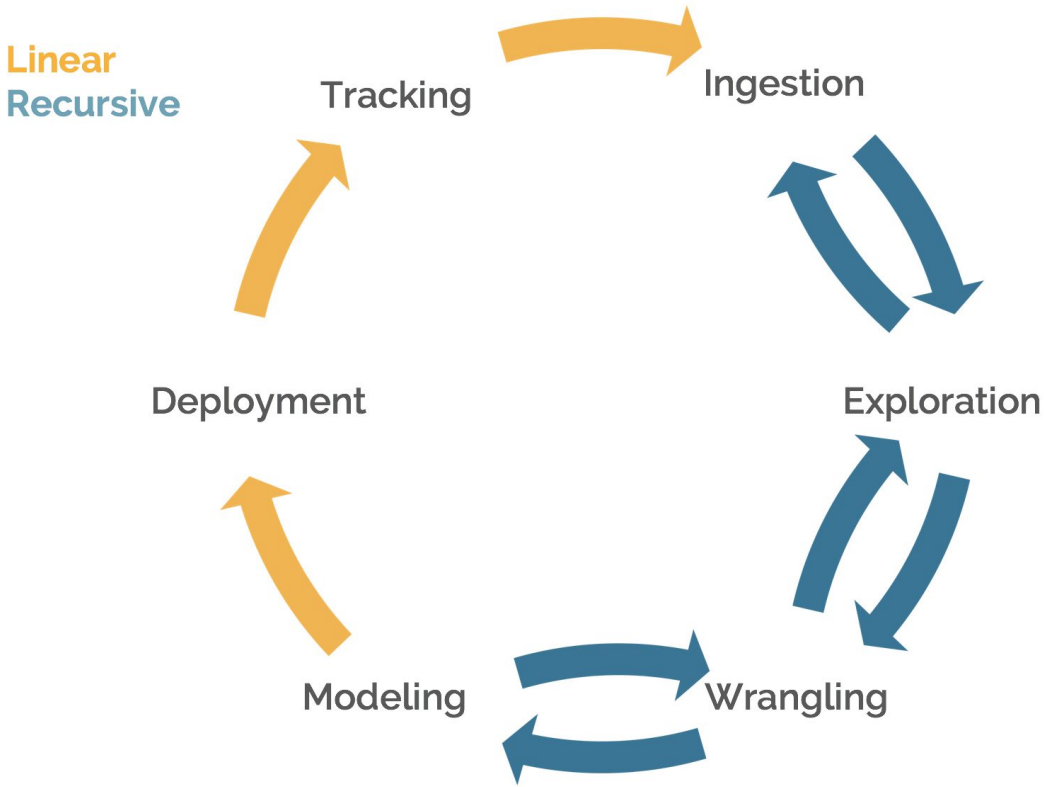
1. We must know the correct and possible questions to ask.
2. We must have the ability to evaluate and debug answers.
3. GenAI models produce a rough average of their training data - I want to strive for deep excellence at specific problems.
4. We still need to be able to think critically and be wary of cognitive offloading.
5. The following tips are less about knowledge and more about leveraging context and implementing a bespoke strategic approach (humans have an edge here).

GenAI can potentially help us on the way (e.g., serve as a better Stack Overflow) as long as we heed the above points.

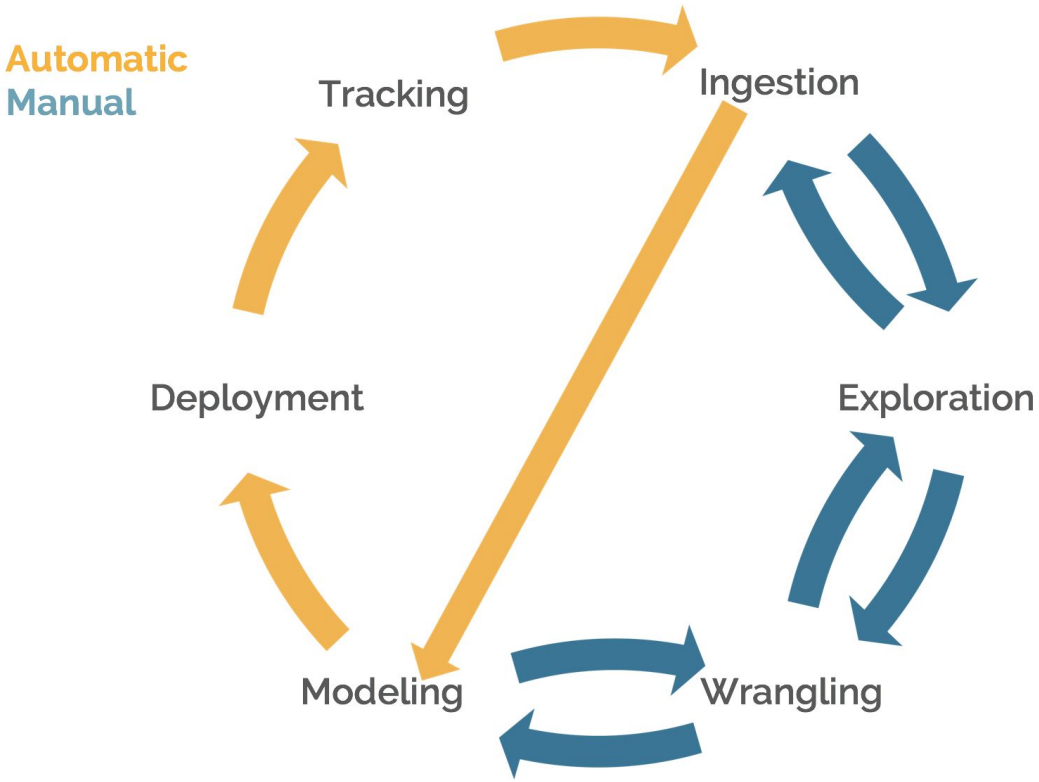
Data Science Pipeline

https://www.linkedin.com/pulse/understanding-data-science-pipeline-micah-melling/

Data Science Pipeline



Data Science Pipeline



Getting Started - GitHub

Clone the repo:

$ git clone https://github.com/micahmelling/prod-app-workshop.git

The repo includes all the code, data, and configuration files you need!

If you need to install git, see here.

(Optionally, you can create a GitHub account, which will allow you to store your own code updates).

Create and Activate Your Virtual Environment

Linux:

$ python -m venv venv

$ source venv/bin/activate

$ pip install -r requirements.txt

Windows (Powershell):

> python -m venv venv

> venv\Scripts\Activate

> pip install -r requirements.txt

(Using a virtual environment is optional but encouraged. Regardless, you will need to install the necessary packages).

In Case of Installation Errors…

You can remove the package versions from requirements.txt and let the pip resolver “figure it out”.

To note, the repo was created using Python 3.9. Different versions of Python can produce subtle differences. (Python 3.9 was chosen for ease of this workshop).

(Running code in Docker or using a tool like Poetry are also good ideas but not ones we will explore in this workshop).

An Aside: Using Copier for Project Templates (Not Necessary for this Project)

Templates are good - they give us a repeatable starting place!

Copier is a useful solution.

https://copier.readthedocs.io/en/stable/

$ python3 -m copier path/to/project/template path/to/destination

Getting Started - AWS Steps Overview

AWS can incur charges! That said, our deployment will be serverless and should be de facto free (and can be taken down quickly and easily).

Create an AWS account

Create an Admin user with programmatic access

Set access keys as environment variables

Create an AWS Account

Start at this link: https://signin.aws.amazon.com/signup?request_type=register

This setup requires a credit card, but following the deployment steps exactly should incur zero cost.

Create a Programmatic Admin User

Create an admin user and grant programmatic access.

Be sure to retrieve your access keys, which will be needed in the next step!

Beware! This is a powerful account. For security reasons, we will clean up all our work at the end of the tutorial.

Set Environment Variables

With more time, we would opt for a mechanism to grant temporary, one-time access keys (which is more secure). For now, we will set programmatic access keys as local environment variables. This action must be done in an environment where your Python script can access them (one option could be in the “terminal” section of your IDE).

```
$ export AWS_ACCESS_KEY_ID=...
$ export AWS_SECRET_ACCESS_KEY=...
$ export AWS_DEFAULT_REGION=us-west-2
```

On Windows, use “setx” or “Env” and not “export”.

The above steps will set the environment variables in your current terminal session. Once you exit the session, the keys will no longer be there. (There are ways we can set keys permanently).

Data Ingestion

In this case, it’s as simple as reading a csv!

Data documentation:
https://www.kaggle.com/datasets/rabieelkharoua/students-performance-dataset

(In our code repo, we make some slight adjustments to our data to make it more interesting).

Project Goal

Predict if a student will have a GPA of at least a B given data about their school involvement and study habits.

Our model will predict the probability of a student having a GPA of at least a B given a set of input data.

Example Input and Output of Our System - What ML Looks like in the “Real World”

Pass in something like this


```
{
        "StudentID": 1001,
        "Age": 18,
        "Gender": "cat_1",
        "ParentalEducation": "cat_2",
        "StudyTimeWeekly": 19.83,
        "Absences": 7,
        "Tutoring": "cat_2",
        "ParentalSupport": "cat_2",
        "Extracurricular": "cat_0",
        "Sports": "cat_0",
        "Music": "cat_1",
        "Volunteering": "cat_0",
        "counselor": "a"
}
```


And get back something like this


```
{
        "prediction": 0.448
}
```

Data Wrangling

The dataset is fairly clean, but we still need to perform some basic wrangling:

- Fill in missing values
- Handle outliers
- Drop unwanted columns

Data Wrangling Pipeline

Ideally, we want to wrap our wrangling code into a single pipeline that is coupled with our model.

Useful Videos (for later)

https://youtu.be/4dGv_6QT2Xw?si=gkDjqnBaOog0hvbI

https://youtu.be/frqcuPwgOl8?si=PcmNu33aVSzjxg1T

Optimizing Data Wrangling and Feature Engineering

Likewise, we can tune our wrangling / engineering in concert with our model's hyperparameters.

Useful Video and Example (for later)

https://youtu.be/8rT4PM3w6ME?si=A4gfU3GN2vpk1jt6

https://github.com/micahmelling/data-science-tips/blob/main/custom_transformer.py

Model Calibration and Prediction Intervals

In machine learning, we always want to quantify uncertainty.

In classification, we want a calibrated model. That is, we want our predicted probabilities to map to real-world probabilities.

In regression, we want a prediction interval. That is, we predict the value will be between X and Y 90% of the time, etc.

More on Model Calibration…

https://endtoenddatascience.com/chapter11-machine-learning-calibration

https://youtu.be/bbvZffubbIQ?si=57WKQstTpH6U14PJ

More on Prediction Intervals…

https://mapie.readthedocs.io/en/latest/

https://youtu.be/RTBmBZtBtuE?si=HISYe2Zywf5yUvzR

Model Optimization

Better options than grid search and randomized search exist.

https://endtoenddatascience.com/chapter10-machine-learning

https://youtu.be/_z8Ri_LwD5E?si=xJ7PyNoRUdKX0FxL

Model Evaluation

We want to evaluate our model on a suite of metrics. One metric does not rule them all.

Model Explanation

We can use a nifty wrapper around SHAP.

$ pip install auto-shap

https://youtu.be/1D_EaiyMwuI?si=Nz2VTTgom4AnbEPZ

Be Sure to Train the Models!

Run the following command to actually train models that we can deploy:

$ python3 modeling/train.py

(Ideally, we want to run the script from the project directory root and using the command line, like the above. If the script is run from modeling/ in an IDE, everything will work, and a slight adjustment can be made to the model path we deploy in app.py. Additionally, the top of modeling/train.py tries to proactively solve any path issues that might be encountered.)

Zappa

Zappa is an easy way to deploy Python Flask apps on AWS Lambda, a “serverless” architecture.

Prepare app.py

The script app.py is what gets deployed to Lambda.

We will need to change out the MODEL_ID global for the model ID we want to deploy. This will be the directory name of a model we produced by running modeling/train.py (we will all get different IDs).

Update the Zappa Settings

We will leverage the existing the zappa_settings.json in the repo.

You will need to fill in the name of the S3 bucket (make something up that is globally unique and avoid underscores; Zappa will create it for you).

The repo's readme includes some additional context that is not strictly necessary for your deployment.

Deploy!

We can simply run the following command to deploy our app.

$ zappa deploy

This process will take a few minutes and will return a URL we can interact with! (Everyone will have a different URL). You can start by clicking on the provided link to do a “health check”.

To start from scratch, if needed or desired, you can instead run the below command. However, you will need to make adjustments to the default settings file; see the repo’s readme.

$ zappa init

Test the App

To interact with the app, download Postman.

You can then send a POST request to the URL returned by running zappa deploy.

You can copy the contents of sample_payload.json into the body of the POST request.

We deployed our model to the /predict endpoint, so you will need to add that to the end of the URL in your POST request. It will look something like: https://<unique>.execute-api.us-west-2.amazonaws.com/dev/predict

Clean Up

First, take down everything you deployed.

$ zappa undeploy --remove-logs

Second, go to IAM in the AWS console and deactivate and / or delete the programmatic access keys for the admin user.

Summary of Technical Steps

- Ensure git is set up
- Clone the repo
- Set up virtual environment
- Create an AWS admin user
- Set AWS environment variables
- Run model training script
- Make updates to zappa_settings.json (adding a unique S3 name)
- Use zappa to deploy to AWS (you'll get an endpoint to interact with)
- Download Postman and use it interact with the provided endpoint
- Use zappa to undeploy the cloud infrastructure
- Deactivate the AWS user

Building a Data Science App

February 22
Park University
Micah Melling