Machine Learning Engineering

Software Engineering Practices

- Clean & Modular Code
- Refactoring Code
- Efficient Code
- Documentation
  - In-line comments
  - Docstrings
  - Readme
- Version Control
  - https://algorithmia.com/blog/how-to-version-control-your-production-machine-learning-models
  - https://nvie.com/posts/a-successful-git-branching-model/
- Testing Code
    - Bad encoding
    - Inappropriate features
    - Unexpected features
  - Unit Tests
    - pytest
      - e.g. test a function
  - Test Driven Development
- Logging
- Code Review
  - Using a Python code linter like **pylint** can automatically check for coding standards and PEP 8 guidelines for you.
  - https://github.com/lyst/MakingLyst/tree/master/code-reviews
  - https://www.kevinlondon.com/2015/05/05/code-review-best-practices.html

Object-Oriented Programming

- class - a blueprint consisting of methods and attributes
- object - an instance of a class. It can help to think of objects as something in the real world like a yellow pencil, a small dog, a blue shirt, etc. However, as you'll see later in the lesson, objects can be more abstract.
- attribute - a descriptor or characteristic. Examples would be colour, length, size, etc. These attributes can take on specific values like blue, 3 inches, large, etc.
- method - an action that a class or object could take
- OOP - a commonly used abbreviation for object-oriented programming

- encapsulation - one of the fundamental ideas behind object-oriented programming is called encapsulation: you can combine functions and data all into a single entity. In object-oriented programming, this single entity is called a class. Encapsulation allows you to hide implementation details much like how the scikit-learn package hides the implementation of machine learning algorithms.

Contributing to a GitHub project

Here are a few links about how to contribute to a github project:

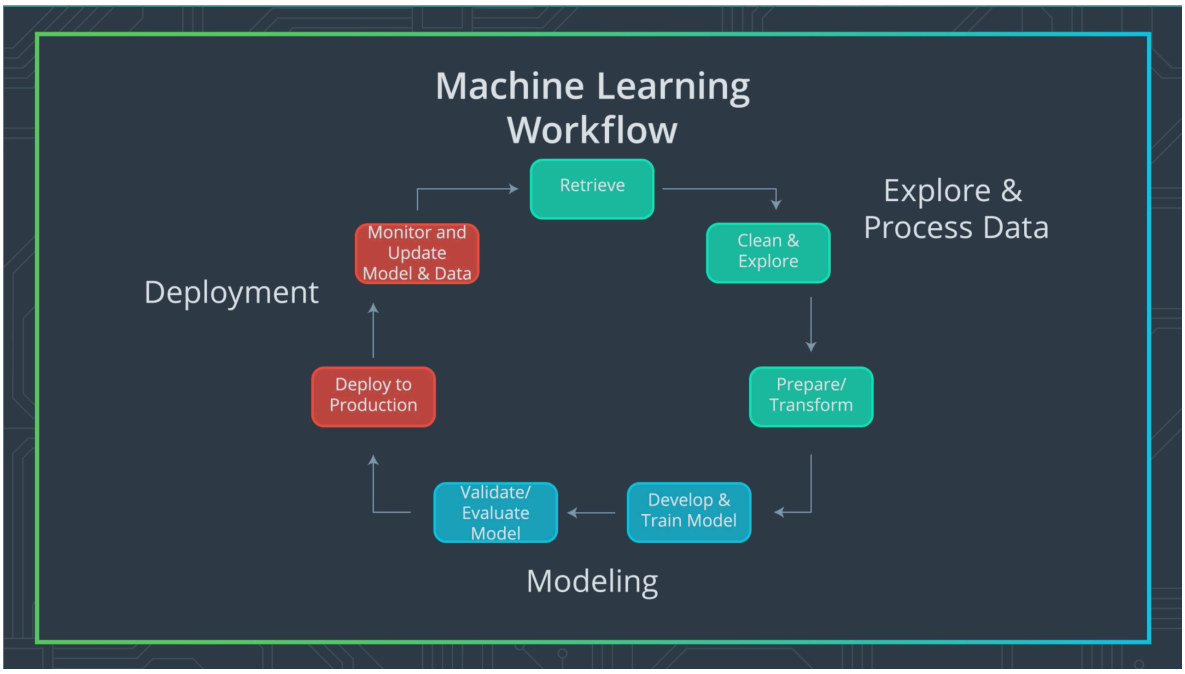
- **Beginner's Guide to Contributing to a Github Project**
- **Contributing to a Github Project**

Advanced Python OOP Topics

Here are a few links to more advanced OOP topics that appear in the Scikit-learn package:

- **Decorators**
- **Mixins**

Deployment



AWS Sagemaker

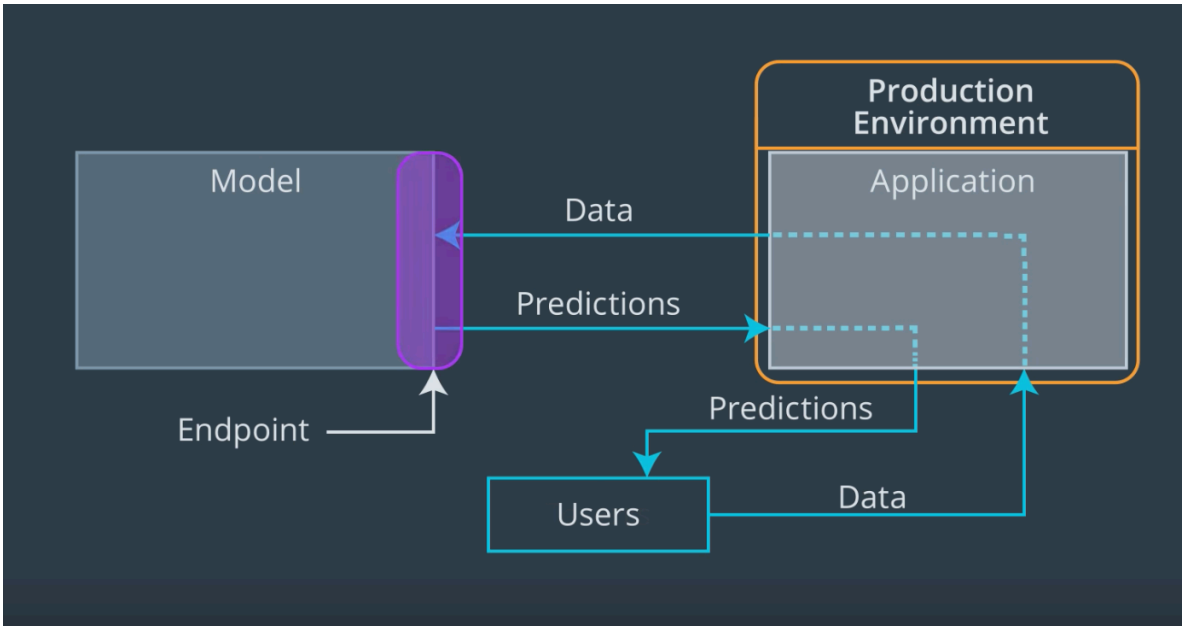
- Link directly to a Github repository or
- Open terminal from a Notebook instance and type:
  - cd SageMaker
  - git clone https://github.com/udacity/sagemaker-deployment.git
  - exit
- Training Jobs
  - View Logs to take a look at the output that was created by a training job.
  - The errors appear here.
- Models
  - Contains the model artefacts

Concept Drift

- Sometimes a model may not work as well as it once did due to changes in the underlying data. In this resource, you can read more about how a model's predictions and accuracy may degrade as a result of something called *concept drift*, which is a change in the underlying data distribution over time. When this happens we might want to update a deployed model, however, our model may be in use so we don't want to shut it down. SageMaker allows us to solve this problem without there being any loss of service.

Deploying a Model in SageMaker

- Create an endpoint that will be used as a way to send data to our model.
- An endpoint is basically a way to allow a model and an application to communicate.



- For our purposes **an endpoint is just a URL**. Instead of returning a web page, like a typical url, this endpoint URL returns the results of performing inference. In addition, we are able to send data to this URL so that our model knows what to perform inference on.
- if you aren't using an endpoint you really need to **shut it down**.
- End Point Configurations
  - Contain the configuration blueprints.
- Endpoints
  - We can shutdown the service from here too just clicking on Delete.

Deploy Model in a Simple Web App

- Create a new endpoint using Lambda to avoid using the Sagemaker API directly with authentication.
  - Create an IAM role for the Lambda function to call Sagemaker endpoint.
  - Next, set a new Lambda function.
  - Test it
    - Create a test event “API Gateway AWS Proxy” and just replace the “body” value.
    - Test
- Create our endpoint using API Gateway
  - AWS Console
  - Networking and Content Delivery
  - API Gateway
  - New API
  - Give a name
  - Add Action —> Create a POST Method
  - Integrate with Lambda
  - Use also Lambda Proxy integration
  - Save
  - Actions —> Deploy API
  - New Stage —> give a name: prod (for production)
  - Get the URL to invoke the API from Stages
  - Add this URL to the index.html in Sagemaker
  - If we stop use it, delete endpoint from Notebook or Console.
  - In addition, delete the Lambda function and the API.

SageMaker Hyperparameter Tuning

- High or Low level

SageMaker provides functionality to update a model

- Update a model without interrupting the web app that using it
- Create an endpoint that sends data to multiple different models (useful for e.g. an A/B test).
- Low level offers more flexibility, but high level is more convenient.
- The model is not created unless you create a transformation object or deploy the model.