Development to production

### **Paths to Deployment**

There are *three* primary methods used to transfer a model from the ***modeling*** *component* to the ***deployment*** *component* of the machine learning workflow. We will be discussing them in *order* of ***least to*** ***most*** commonly used. The ***third*** *method* that's *most* similar to what’s used for *deployment* within ***Amazon’s SageMaker***.

#### **Paths to *Deployment*:**

1. Python model is *recoded* into the programming language of the production environment.
2. Model is *coded* in *Predictive Model Markup Language* (PMML) or *Portable Format Analytics* (PFA).
3. Python model is *converted* into a format that can be used in the production environment.

#### **Recoding Model into Programming Language of Production Environment**

The *first* method which involves recoding the Python model into the language of the production environment, often Java or C++. This method is *rarely used anymore* because it takes time to recode, test, and validate the model that provides the *same* predictions as the *original*.

#### **Model is Coded in PMML or PFA**

The *second* method is to code the model in Predictive Model Markup Language (PMML) or Portable Format for Analytics (PFA), which are two complementary standards that *simplify* moving predictive models to *deployment* into a *production environment*. The Data Mining Group developed both PMML and PFA to provide vendor-neutral executable model specifications for certain predictive models used by data mining and machine learning. Certain analytic software allows for the direct import of PMML including but *not limited* to IBM SPSS, R, SAS Base & Enterprise Miner, Apache Spark, Teradata Warehouse Miner, and TIBCO Spotfire.

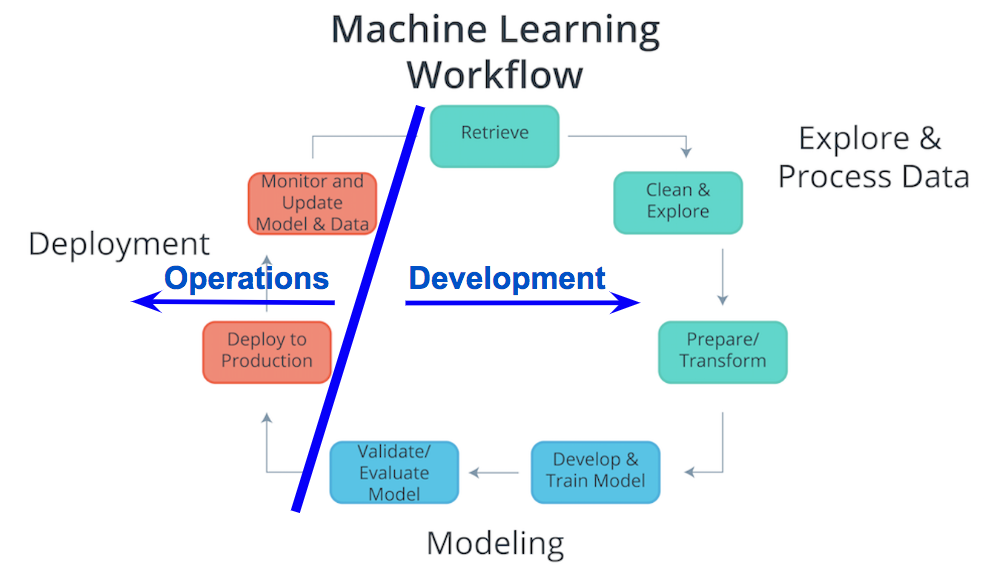
#### **Model is Converted into Format that’s used in the Production Environment**

The *third* method is to build a Python model and ***use*** *libraries* and *methods* that *convert* the model into ***code*** that can be used in the *production environment*. Specifically *most* popular machine learning software frameworks, like PyTorch, TensorFlow, SciKit-Learn, have methods that will *convert* Python models into *intermediate standard format*, like ONNX ([Open Neural Network Exchange](https://onnx.ai/) format). This *intermediate standard format* then can be ***converted*** into the software native to the *production environment*.

* This is the *easiest* and *fastest* way ***to move*** a Python model from *modeling* directly to *deployment*.
* Moving forward this is *typically* the way *models* are ***moved*** into the *production environment*.
* Technologies like *containers*, *endpoints*, and *APIs* (Application Programming Interfaces) also help ***ease*** the ***work*** required for *deploying* a model into the *production environment*.

We will discuss these *technologies* in more detail in the ***next*** sections.

### **Machine Learning Workflow and DevOps**

****

#### **Machine Learning Workflow and Deployment**

Considering the components of the *Machine Learning Workflow*, one can see how *Exploring and Processing Data* is tightly coupled with *Modeling*. The *modeling* **can’t** occur without *first* having the *data* the model will be based upon *prepared* for the modeling process.

Comparatively *deployment* is **more** tightly coupled with the *production environment* than with *modeling*or *exploring and processing the data*. Therefore, *traditionally* there’s was a separation between *Deployment* and the *other components* of the machine learning workflow. Specifically looking at the diagram ***above***, the **Process Data and Modeling** are considered ***Development***; whereas, **Deployment** is *typically* considered ***Operations***.

In the past typically, ***development*** was handled by **analysts**; whereas, ***operations*** was handled by **software developers** responsible for the *production environment*. With *recent* developments in *technology* (containers, endpoints, APIs) and the *most* common path of deployment; *this division* between ***development*** and ***operations*** *softens*. The softening of this division enables **analysts** to handle certain aspects of ***deployment*** and enables faster updates to faltering models.

#### **Deployment within Machine Learning Curriculum**

***Deployment*** is **not** commonly included in *machine learning curriculum*. This likely is associated with the *analyst's* typical focus on ***Exploring and Processing Data*** and ***Modeling***, and the *software developer's* focusing more on ***Deployment*** and the *production environment*. Advances in cloud services, like [SageMaker](https://aws.amazon.com/sagemaker/) and [ML Engine](https://cloud.google.com/ml-engine/), and deployment technologies, like Containers and REST APIs, allow for *analysts* to easily take on the responsibilities of ***deployment***.