ML Modeling with Amazon SageMaker
Mico Ellerich Comia
May 22, 2021 (Afternoon Session)

---

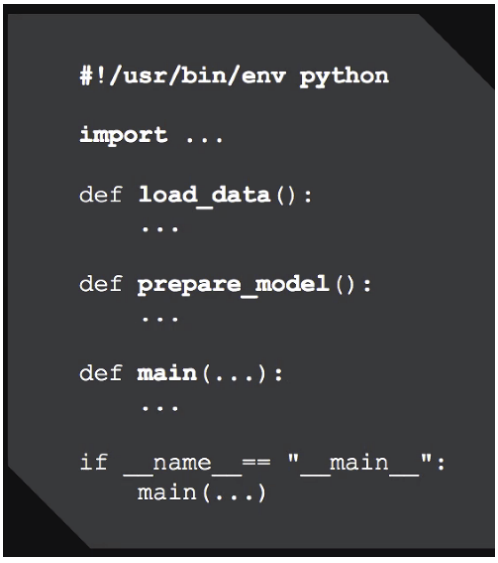
1. **SageMaker Studio**
    - Overview:
        - An interface to help you perform straightforward and basic tasks
        - Shows options/features available with SageMaker (lineage, pipeline, autopilot)
        - Similar to Jupyter lab with integrations.
        - Takes a while to create a notebook:
            1. Configuration of Studio.
            2. Kernel is not ready instantaneously,
            3. Creates instances unlike Notebook instances where EC2 instances are already created.
            4. Internally, SageMaker Studio uses containers to process logic/kernel for notebooks.
        - Can't use local mode.
    - Standard Setup
        - Configuration:
            1. IAM Option
            2. Create a new role:
                - Any S3 Bucket (For prod bucket, better to use Specific S3 buckets)
            3. Default for remaining options:
                - Options similar to EC2
    - Parts:
        - Sidebar:
            1. File - create folders/files
            2. GIT Settings - initialize a repository, clone a repository, etc.
            3. Instances - check running instances
            4. Components and registries - check created resources
    - Images:

        - For custom container images (e.g. you want to use R)
        - If built-in environments are not sufficient.
    - Features:
        - Experiments help you track input, hyperparameters, input artifacts, model output. Tool to help organize things (for auditing of models/lineage of models).
        - Data Wrangler performs data wrangling.
        - Feature Store - stores features and data. If you want features to be clean across notebooks.
        - Model Registry - can contain models.

2. **AutoML (Automated Machine Learning) with Amazon SageMaker Autopilot**
    - Create experiment
        - Name
        - Specify where input data is located in S3 (e.g. s3://studio-test-bucket-12345/)
        - Specify the name of target column
        - Specify where output will be located
        - Select ML problem type:
            1. Auto
            2. Regression
            3. Classification
            4. Etc.
        - Do you want to run a complete experiment?
            1. Candidate definition notebook: provides different options of models/algorithms for the training job
        - Auto deploy deploys an endpoint
        - Configure:
            1. Max trial runtime
            2. Max job runtime
            3. Max candidate
    - Explainability Report
        - Utilizes Clarify to explain what columns are important

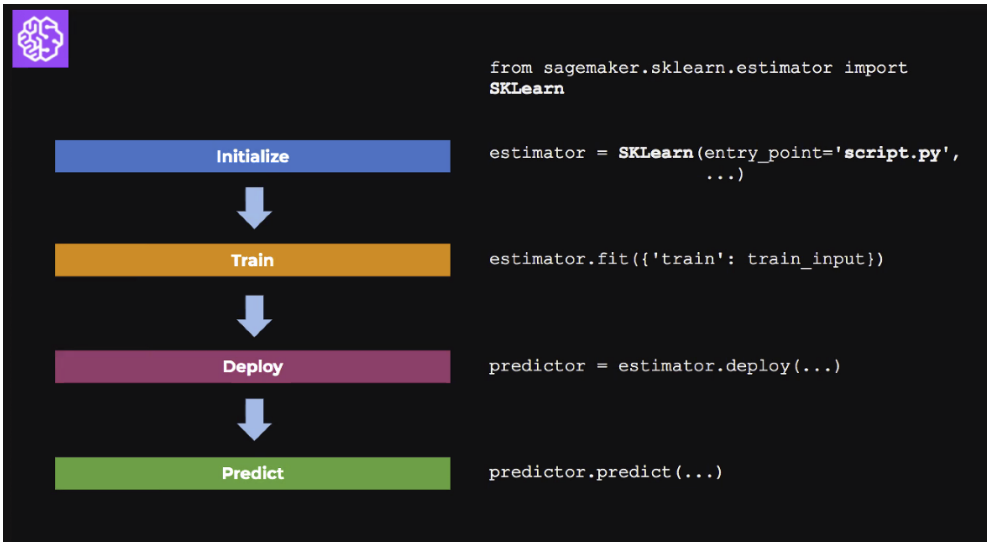
- Requirements:
  - CSV file with 500 data points

3. **ML Libraries and Frameworks with SageMaker**
   - SageMaker procedure:
     - Preprocessing
     - SageMaker Training
     - SageMaker Deployment
     - Inference
   - Steps (should work without SageMaker):
     - Prepare the “train” script



```
#!/usr/bin/env python

import ...

def load_data():
    ...

def prepare_model():
    ...

def main(...):
    ...

if __name__== "__main__":
    main(...)
```

     - Test if the train script works locally
       1. Loads data
       2. Loads hyperparameters
       3. Performs training
       4. Outputs model and model artifacts
     - Prepare the SageMaker Estimator and use the train script during training and deployment
       1. Define where training script is in “entry_point”
       2. PyTorch requires an inference script
       3. Similar steps for TensorFlow / Keras



- Local mode
  - Importing scripts to SageMaker needs a bit of trial and error
  - Allows you to create estimators and deploy them to your local environment
  - Useful when you're using ML and Deep Learning frameworks
  - Supported for frameworks images (TensorFlow, MXNet, Chainer, PyTorch, and Scikit-Learn) and images you supply yourself