

Spark Notebooks in Cloud

Databricks, AWS, Google

Comparing PySpark cloud offerings

- AWS & Microsoft Azure
 - Free tier doesn't provide cluster for PySpark
- Google Cloud Platform
 - Free tier Dataproc does provide cluster for PySpark
- Simplest
 - Databricks
- Best, inexpensive managed cluster for PySpark
 - GCP Dataproc

Managed and unmanaged infrastructure

- Managed infrastructure
 - Underlying computing resources (CPUs, cluster, network) managed by technology, not by user (data scientist)
 - E.g., Databricks, Google Colab, AWS SageMaker, Azure studio
- Unmanaged infrastructure
 - User must create cluster and install libraries (e.g., Spark)
 - E.g.,
 - Google Colab, Azure simple notebook (for PySpark setup)
 - AWS SageMaker, Azure studio for connecting to clusters defined elsewhere in technology stack

Development phases

- Common to all environments
 - Build your model
 - Train your model
 - Deploy your model
 - AWS supports deployment into Docker images, which run in their clusters

Summary of environments for PySpark notebooks

- Google Colab
 - Free, simple environment
 - See also Dataproc, AI Platform
- Amazon SageMaker
 - Can be free and simple
 - Many associated tools for more complex usage
 - Define your own cluster
 - Use Amazon Spark libraries
 - Deploy models to Docker images as endpoints on network API
- Microsoft Azure
 - Simple notebook
 - Azure Databricks
 - Cluster + notebook
 - Studio
 - Drag & drop interface
 - Design for non-data scientist

Google Colab

- <https://colab.research.google.com/>
- Jupyter notebook stored in Google Drive
- Simple for Python
- Managed infrastructure
 - Linux with 12G memory
- Setup required for PySpark
- Tips
 - <https://www.kdnuggets.com/2018/02/essential-google-colaboratory-tips-tricks.html>
 - <https://www.geeksforgeeks.org/how-to-use-google-colab/>
 - <https://dev.to/kriyeng/8-tips-for-google-colab-notebooks-to-take-advantage-of-their-free-of-charge-12gb-ram-gpu-be4>

Google Colab setup required

```
!wget -q http://apache.cs.utah.edu/spark/spark-2.4.4/spark-2.4.4-bin-hadoop2.7.tgz
!tar xf spark-2.4.4-bin-hadoop2.7.tgz
!pip install -q findspark
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.6/dist-packages (2.4.0)
Requirement already satisfied: py4j==0.10.7 in /usr/local/lib/python3.6/dist-packages (0.10.7)
```

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-2.4.4-bin-hadoop2.7"
```

```
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
import findspark
#findspark.init()
```

```
APP_NAME = "Housing"
SPARK_URL = "local[*]"
```

```
spark = SparkSession\
    .builder\
    .appName(APP_NAME)\
    .master(SPARK_URL)\
    .getOrCreate()
```

```
sc = spark.sparkContext
sqlContext = SQLContext(sc)
```

```
!ls sample_data
```

```
anscombe.json          mnist_test.csv
california_housing_test.csv  mnist_train_small.csv
california_housing_train.csv  README.md
```

```
df = spark.read.csv('sample_data/california_housing_test.csv',
df.printSchema())
```

Amazon SageMaker

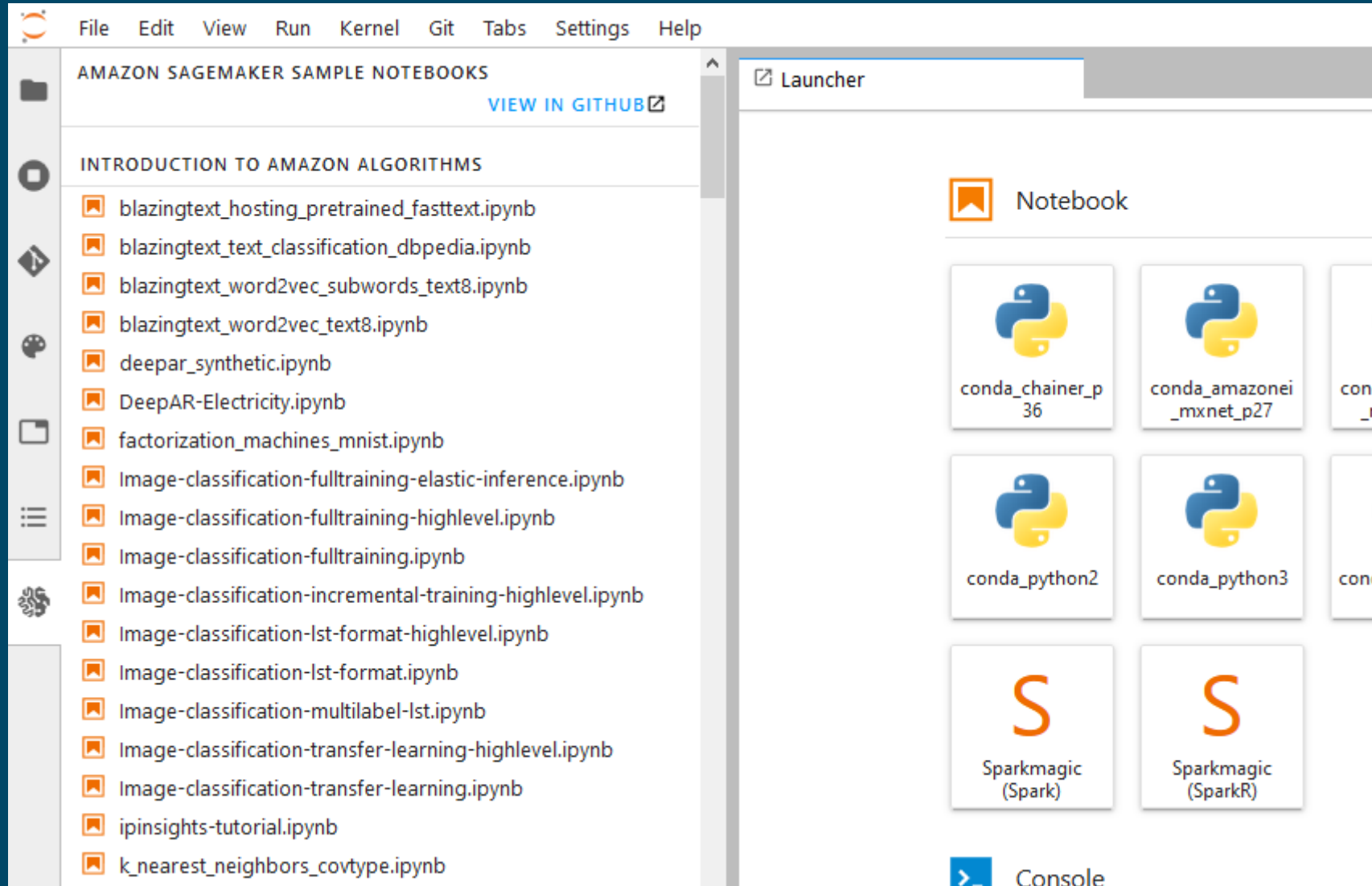
- <https://aws.amazon.com/sagemaker/>
- Managed infrastructure
 - Select from different size computing environments
 - Can also create EC2 spark cluster for environment
- Works with deployment
 - Using Amazon models (e.g., XGBoost), can use pre-built Docker image to create Docker image with model to create an endpoint where others can access your model via a standard API (over the network)
 - Can also use your own custom algorithms, create your own custom Docker image to deploy your model

Create a notebook instance

The screenshot displays the Amazon SageMaker console interface. On the left, a sidebar menu includes 'Dashboard', 'Search', 'Ground Truth' (with sub-items 'Labeling jobs', 'Labeling datasets', 'Labeling workforces'), 'Notebook' (with sub-items 'Notebook instances', 'Lifecycle configurations', 'Git repositories'), and 'Training' (with sub-items 'Algorithms', 'Training jobs', 'Hyperparameter tuning jobs'). The main panel is titled 'Amazon SageMaker > Notebook instances'. It features a 'Notebook instances' header with an 'Actions' dropdown and a prominent orange 'Create notebook instance' button. Below this is a search bar labeled 'Search notebook instances' and pagination controls showing '1' of 1 items. A table lists the notebook instances with columns: Name, Instance, Creation time, Status, and Actions. One instance, 'demo', is listed with type 'ml.t2.medium', created on 'Nov 17, 2019 15:40 UTC', and status 'InService' (indicated by a green checkmark). The Actions column for 'demo' provides links to 'Open Jupyter' and 'Open JupyterLab'.

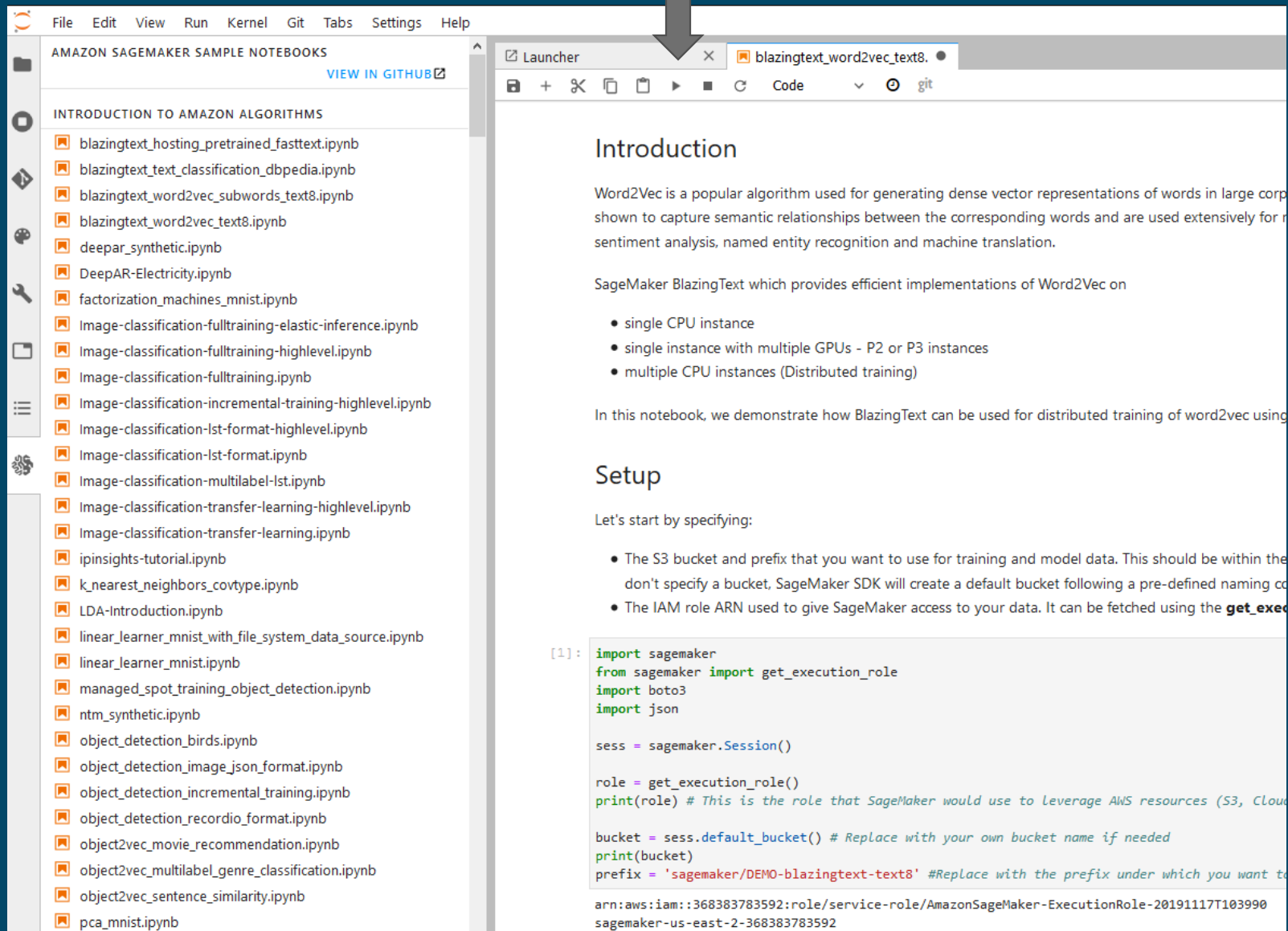
	Name ▼	Instance	Creation time ▼	Status ▼	Actions
<input type="radio"/>	demo	ml.t2.medium	Nov 17, 2019 15:40 UTC	InService	Open Jupyter Open JupyterLab

Open notebook (Jupyter lab tab) select from examples



Word2Vec example

run each cell



The screenshot displays the Amazon SageMaker console interface. On the left, a sidebar lists various sample notebooks under the heading 'AMAZON SAGEMAKER SAMPLE NOTEBOOKS'. The notebook 'blazingtext_word2vec_text8.ipynb' is selected. A large grey arrow points from the title 'Word2Vec example' to the notebook title. The main panel shows the notebook content, which includes an 'Introduction' section, a 'Setup' section, and a code cell. The 'Introduction' section describes Word2Vec as a popular algorithm for generating dense vector representations of words. The 'Setup' section provides instructions on how to use SageMaker BlazingText for distributed training. The code cell shows the following Python code:

```
[1]: import sagemaker
from sagemaker import get_execution_role
import boto3
import json

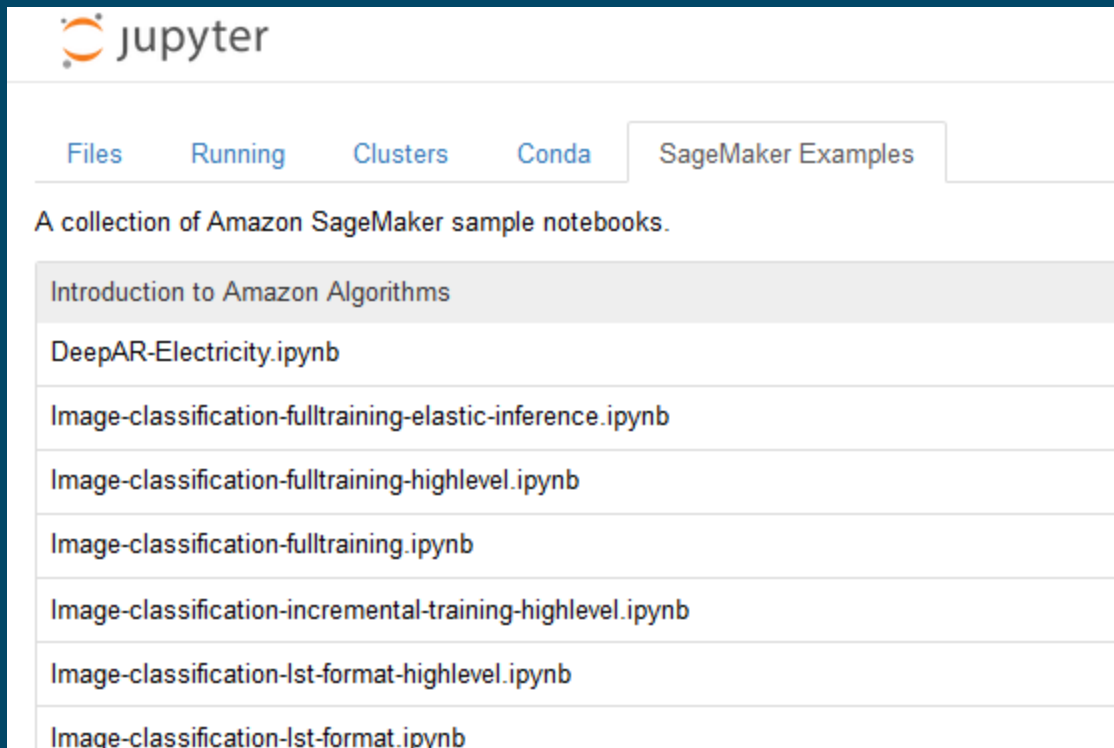
sess = sagemaker.Session()

role = get_execution_role()
print(role) # This is the role that SageMaker would use to Leverage AWS resources (S3, Cloud

bucket = sess.default_bucket() # Replace with your own bucket name if needed
print(bucket)
prefix = 'sagemaker/DEMO-blazingtext-text8' # Replace with the prefix under which you want to
```

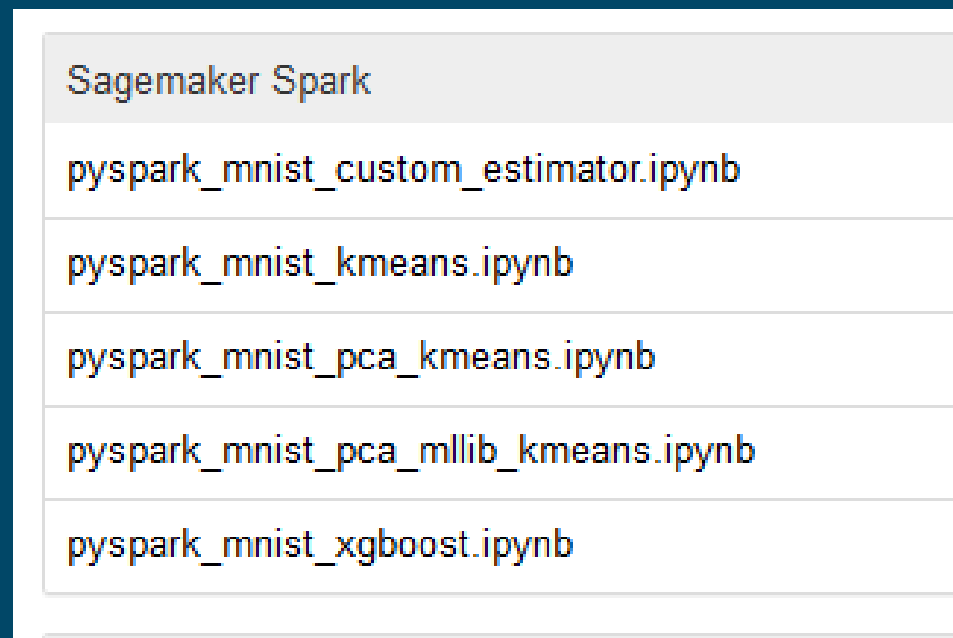
The output of the code cell shows the IAM role ARN: `arn:aws:iam::368383783592:role/service-role/AmazonSageMaker-ExecutionRole-20191117T103990` and the S3 bucket name: `sagemaker-us-east-2-368383783592`.

Open from Jupyter **notebook tab**



The screenshot shows the Jupyter web interface. At the top, there's a navigation bar with tabs: Files, Running, Clusters, Conda, and SageMaker Examples (which is selected). Below the tabs, a text line reads "A collection of Amazon SageMaker sample notebooks." Below this is a list of notebook titles. The first item, "Introduction to Amazon Algorithms", is highlighted with a light gray background. The other items are listed in plain text.

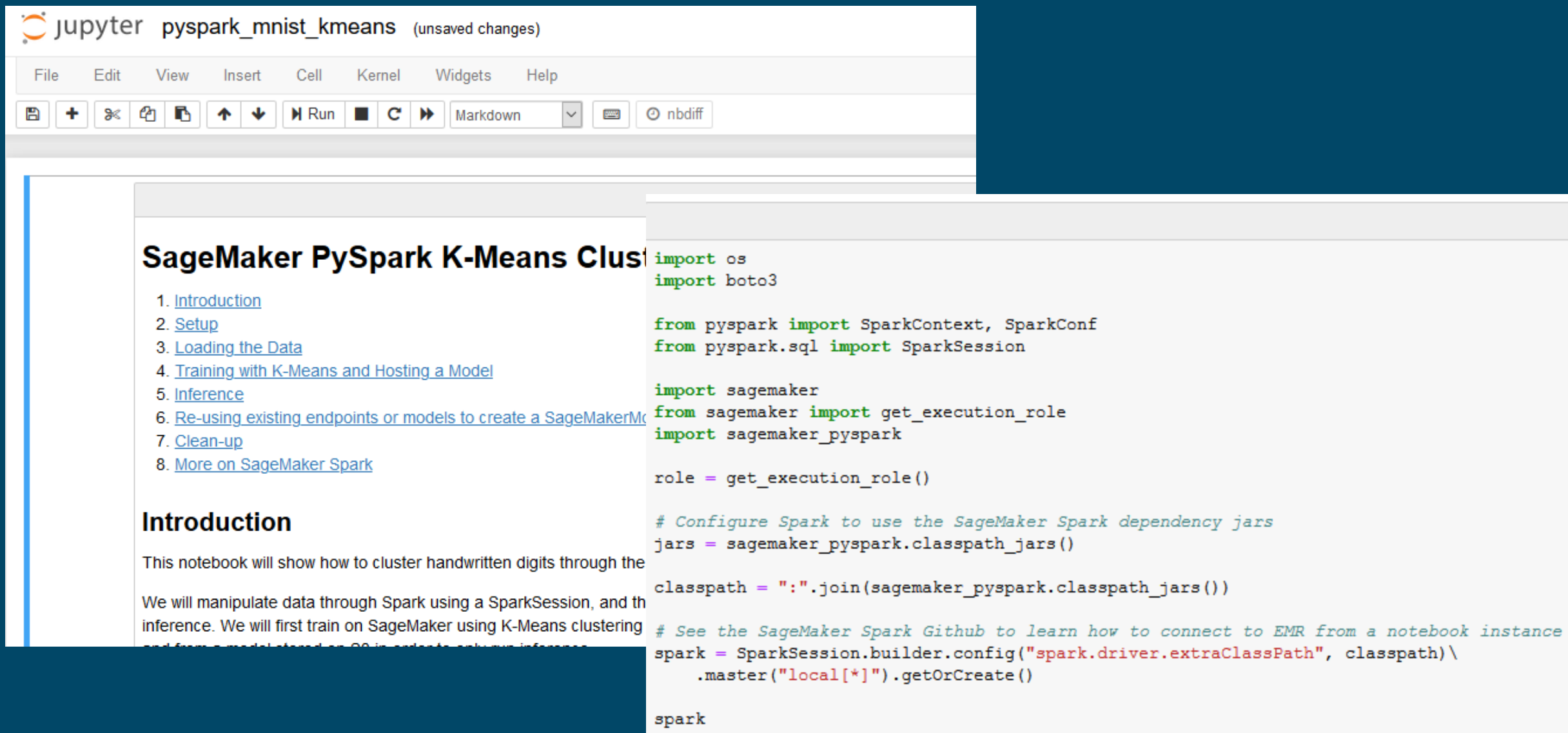
Introduction to Amazon Algorithms
DeepAR-Electricity.ipynb
Image-classification-fulltraining-elastic-inference.ipynb
Image-classification-fulltraining-highlevel.ipynb
Image-classification-fulltraining.ipynb
Image-classification-incremental-training-highlevel.ipynb
Image-classification-lst-format-highlevel.ipynb
Image-classification-lst-format.ipynb



The screenshot shows a list of notebooks under the heading "Sagemaker Spark". The heading is in a light gray box. Below it, a list of notebook titles is shown. The first item, "pyspark_mnist_custom_estimator.ipynb", is highlighted with a light gray background. The other items are listed in plain text.

Sagemaker Spark
pyspark_mnist_custom_estimator.ipynb
pyspark_mnist_kmeans.ipynb
pyspark_mnist_pca_kmeans.ipynb
pyspark_mnist_pca_mllib_kmeans.ipynb
pyspark_mnist_xgboost.ipynb

Running PySpark in Amazon SageMaker



The screenshot shows a Jupyter notebook interface with the title 'pyspark_mnist_kmeans' and '(unsaved changes)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar contains icons for saving, adding cells, undo, redo, and running code. The notebook content is divided into two main sections: a table of contents on the left and a code cell on the right.

SageMaker PySpark K-Means Clustering

- [1. Introduction](#)
- [2. Setup](#)
- [3. Loading the Data](#)
- [4. Training with K-Means and Hosting a Model](#)
- [5. Inference](#)
- [6. Re-using existing endpoints or models to create a SageMaker Model](#)
- [7. Clean-up](#)
- [8. More on SageMaker Spark](#)

Introduction

This notebook will show how to cluster handwritten digits through the Spark MLlib library. We will manipulate data through Spark using a SparkSession, and then use the trained model for inference.

```
import os
import boto3

from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession

import sagemaker
from sagemaker import get_execution_role
import sagemaker_pyspark

role = get_execution_role()

# Configure Spark to use the SageMaker Spark dependency jars
jars = sagemaker_pyspark.classpath_jars()

classpath = ":".join(sagemaker_pyspark.classpath_jars())

# See the SageMaker Spark Github to learn how to connect to EMR from a notebook instance
spark = SparkSession.builder.config("spark.driver.extraClassPath", classpath)\
    .master("local[*]").getOrCreate()

spark
```

Amazon SageMaker

- Specialized managed environment
 - Supports creation and deployment of containers from PySpark notebooks
 - Depends on AWS API
- Similar to
 - Google AI platform
 - Microsoft Azure ML studio

Microsoft Azure notebook environments

- <https://notebooks.azure.com/>
- Managed infrastructure
 - 4-core CPU having a 17 GB of RAM
 - 2-core CPU having a 14 GB of RAM plus a GPU
- Notebook environments
 - Basic, like Colab, <https://notebooks.azure.com/>
 - Azure Databricks, <https://azure.microsoft.com/en-us/services/databricks/>
 - Studio, like SageMaker, <https://studio.azureml.net/>

Basic notebook

```
!pip install pyspark
```

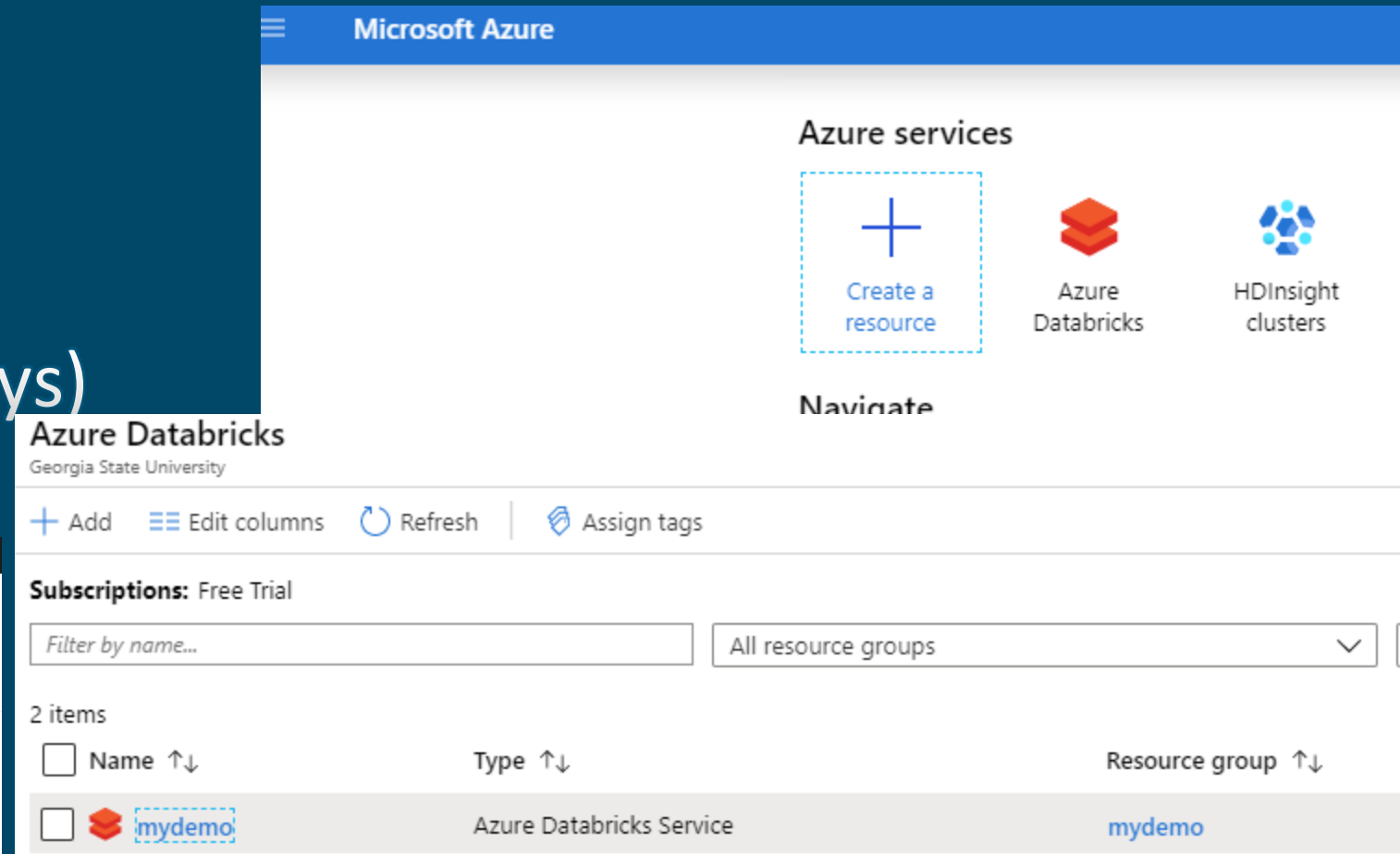
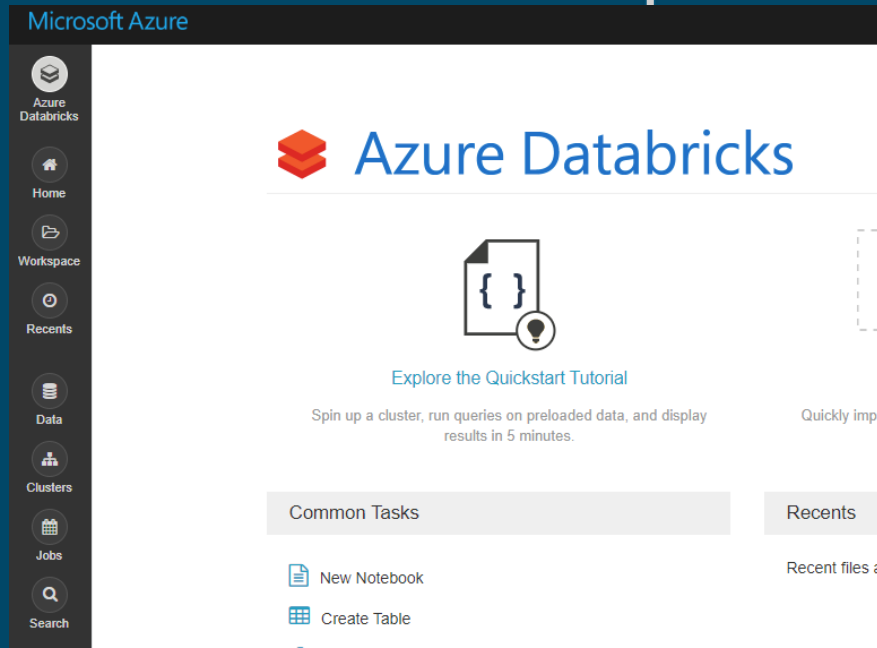
```
Requirement already satisfied: pyspark in .  
Requirement already satisfied: py4j==0.10.
```

```
spark = SparkSession.builder.appName(name="PySpark Intro").master("local[*]").getOrCreate().newSession()  
# .config("spark.jars", "hadoop-aws-2.7.3.jar")  
spark
```

<https://notebooks.azure.com/loldja/projects/pyspark-intro>

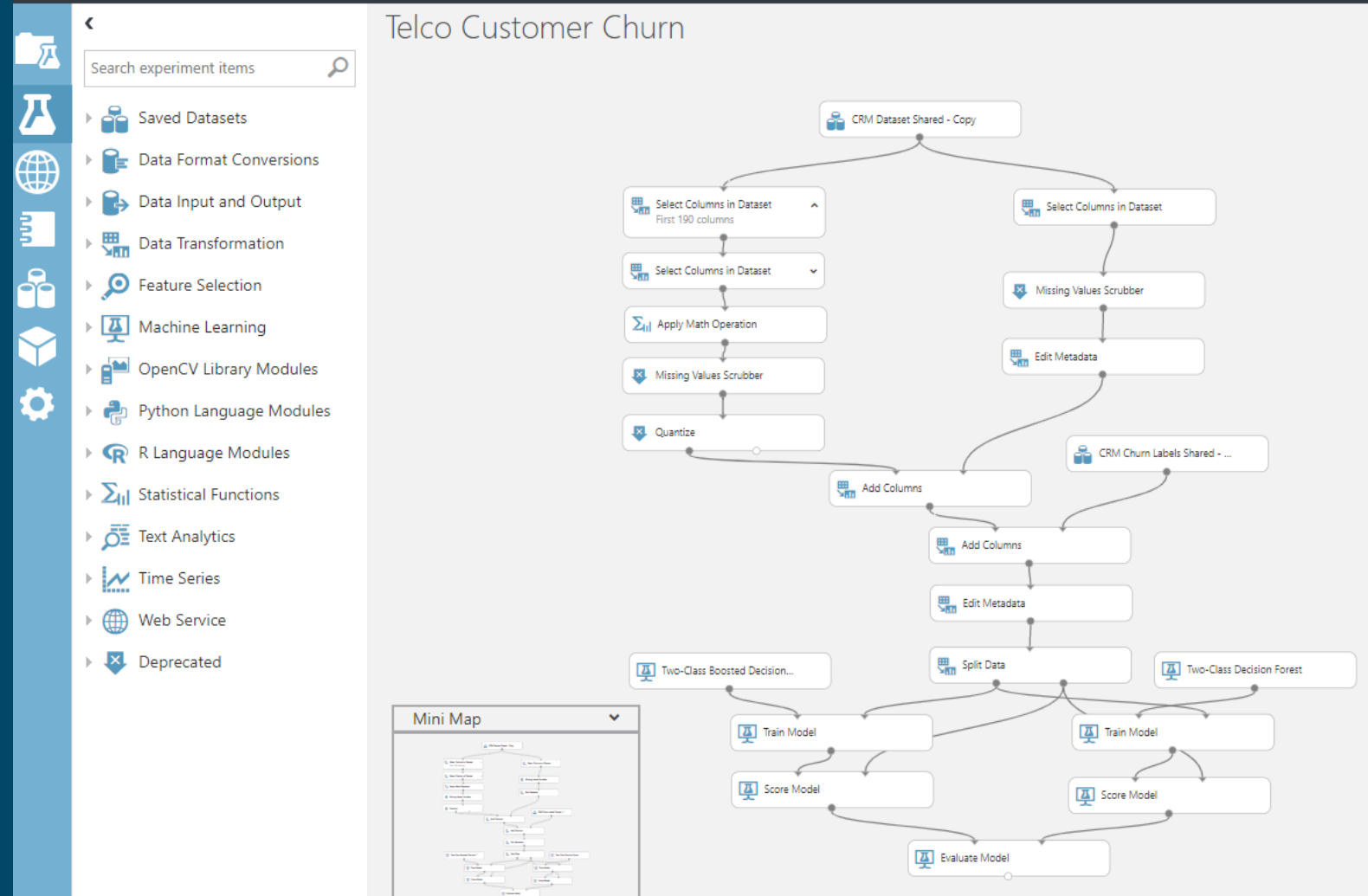
Azure Databricks notebook

1. Create an Azure Databricks service
2. Select (after it deploys)
3. Launch workspace



Azure ML studio

- <http://studio.azureml.net>
- Drag & drop



Important to remember

- Providers offer managed & unmanaged clusters
 - Managed has no need to specify node details, such as number, memory, CPU, pre-installed software etc.
 - Unmanaged requires that user configure cluster