



Boost productivity with generative AI and scalable development tools using Jupyter notebooks from anywhere

Giuseppe Angelo Porcelli

Principal ML Solutions Architect at
Amazon Web Services (AWS)

Agenda

Jupyter and AWS – our vision and innovations

New extensions to use from anywhere Jupyter runs

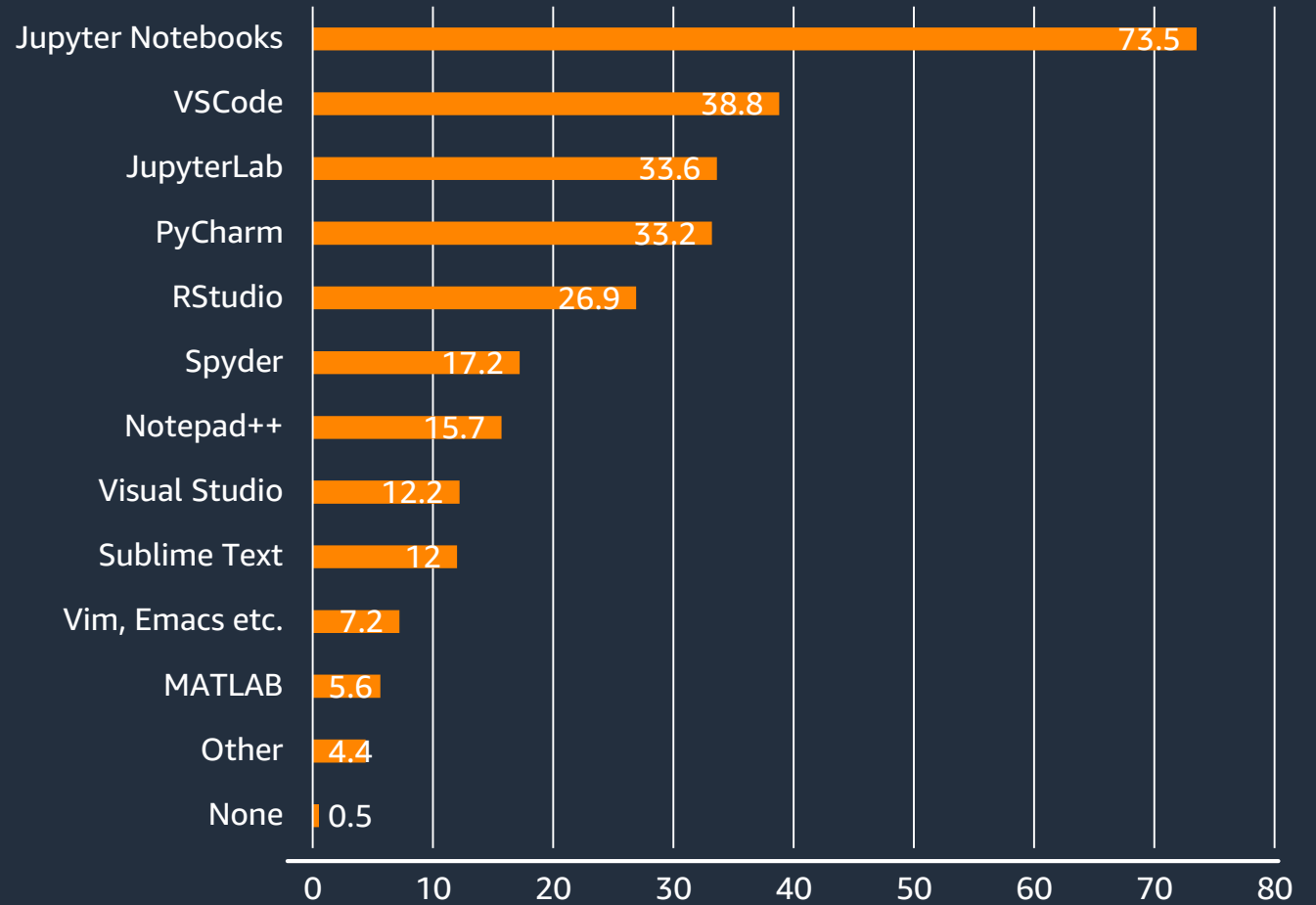
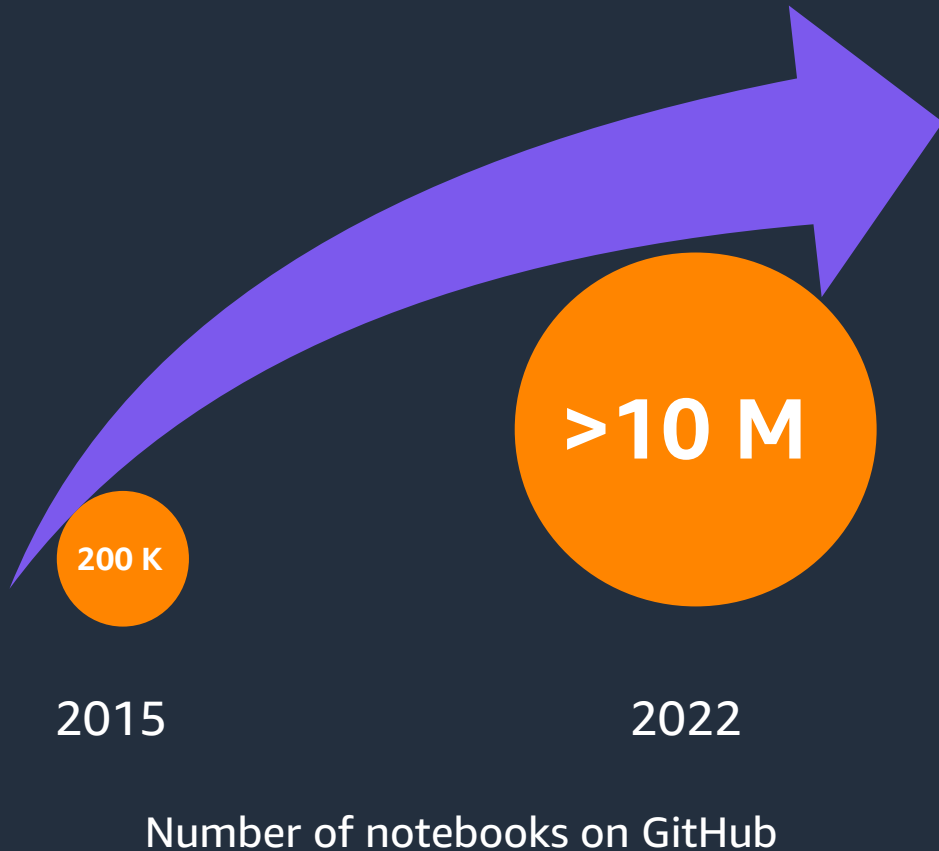
Learn and experiment ML for free

Advance your ML development with Amazon SageMaker

Demo

Q&A

Jupyter Notebooks for data science and ML

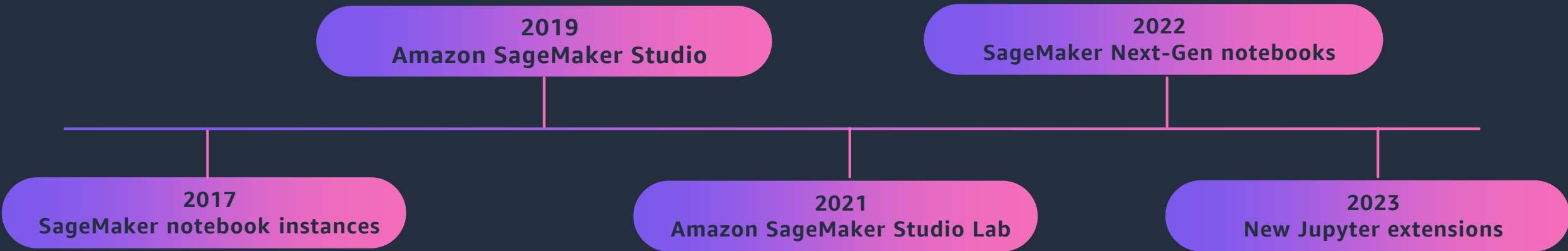


State of Machine Learning and Data Science - 2021 by Kaggle

Our vision

Make machine learning and data science accessible to every learners, developers, and data scientists

AWS's notebooks innovations



New tools for data science, ML, and scientific computing – install today and use from anywhere Jupyter runs



Generative AI



Scale
development



Amazon
CodeWhisperer
Jupyter extension



Jupyter AI



Notebook
scheduling



SageMaker
distribution

New!

Jupyter AI

An open source project to bring advanced generative AI features to Jupyter Notebooks

Join the session
5/11 10:30-11:00am Louis Armand 2
Using generative AI models to help Jupyter users perform tasks with code, data, and more

```
pip install jupyter_ai
```



The screenshot shows a Jupyter Notebook window titled 'code.ipynb' running on 'Python 3 (ipykernel)'. The notebook contains five code cells:

- [1]: `%reload_ext jupyter_ai`
- [2]: `%ai chatgpt --format code` with the prompt: "A program that asks me for my name and then greets me by my name, in Polish". The output is: "AI generated code inserted below".
- [3]: `name = input("Jak masz na imię? ")` and `print("Cześć " + name + "!")`. The output shows the user input "foo" and the printed output "Cześć foo!".
- [4]: `%ai chatgpt --format code` with the prompt: "A function that computes the lowest common multiples of two integers, and a function that runs 5 test cases of".
- [5]: A Python function `def lcm(x, y):` that finds the least common multiple, followed by a test function `def test_lcm():` with assertions for various inputs and a call to `test_lcm()`.

On the right side of the notebook, there is a chat interface with two messages:

- Anonymous Europa (37 seconds ago): "Hello world"
- Jupyter AI (36 seconds ago): "Hello! Is there anything you would like to discuss or ask me?"

At the bottom of the chat, there is a text input field with the text "What does this code do?", a send button, and two checkboxes: "Include selection" (checked) and "Replace selection" (unchecked).

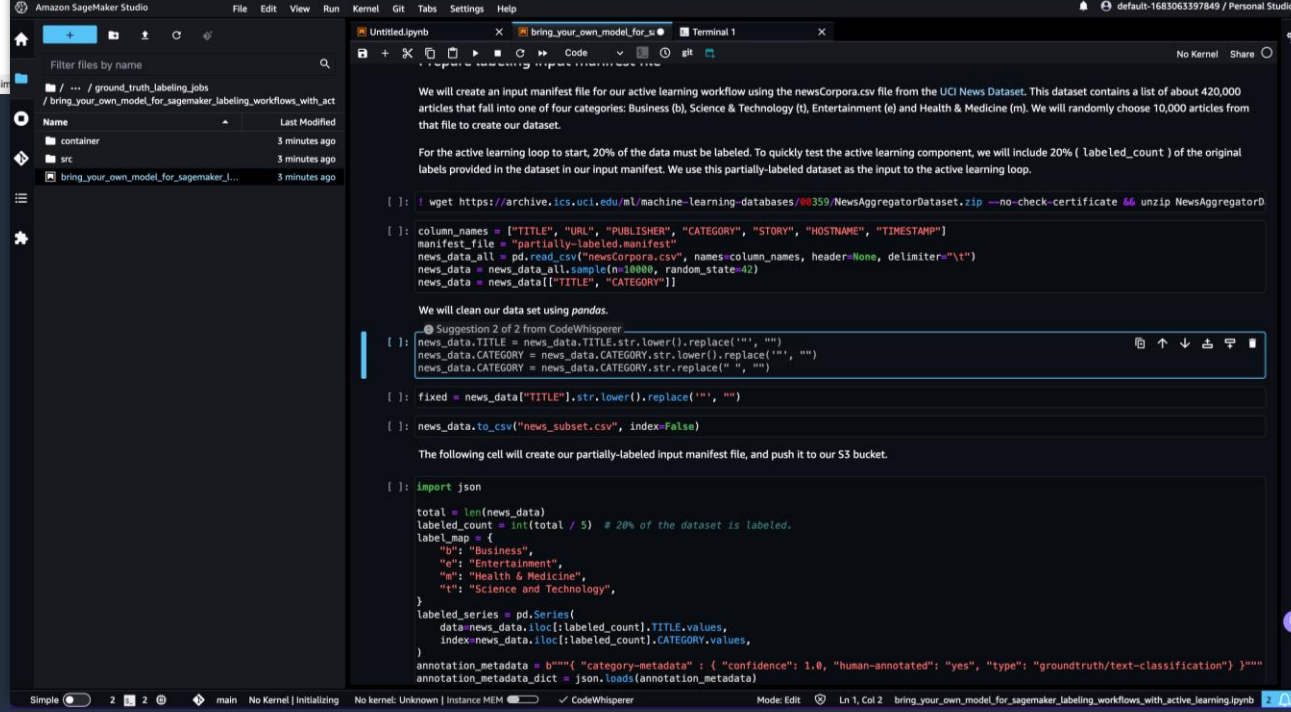
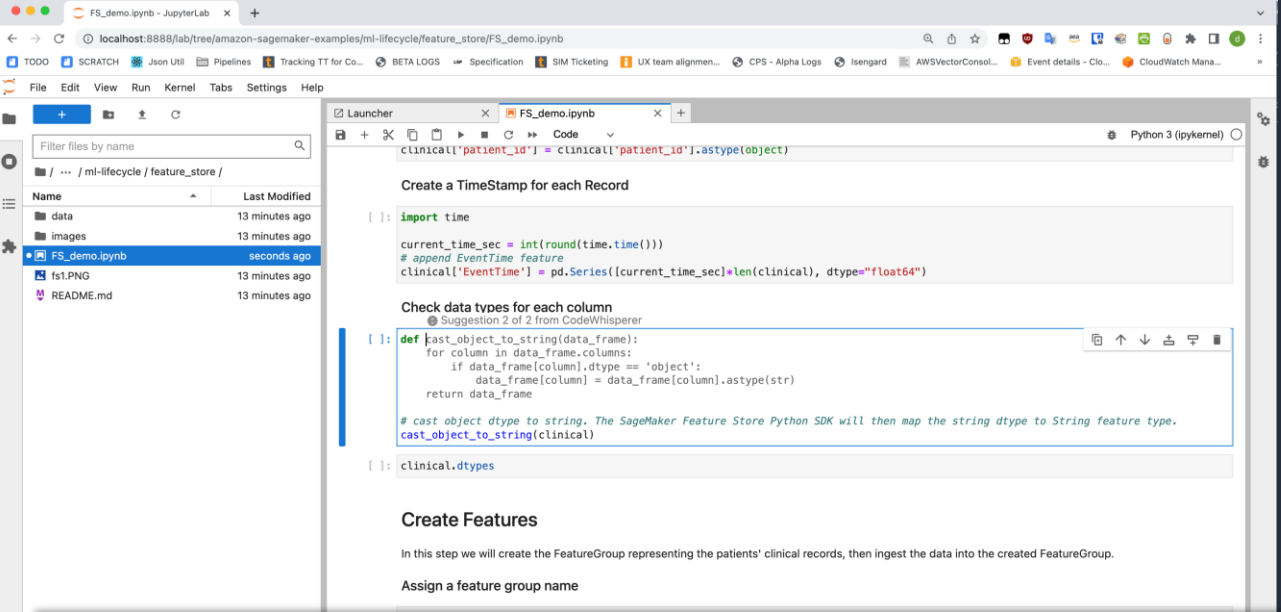




Amazon CodeWhisperer Jupyter extension

Generate real-time, single-line or full-function code suggestions for Python notebooks in JupyterLab and Amazon SageMaker Studio

`pip install amazon-codewhisperer-jupyterlab-ext`



New!

Notebook scheduling Jupyter extension

Select a notebook and automate it as a job that can run in a production environment via a simple yet powerful user interface

Join the session

5/11 15:20-15:30pm Poster

Introducing Jupyter Scheduler: native support in Jupyter for running and scheduling Jupyter notebooks as jobs

```
pip install  
amazon_sagemaker_jupyter_scheduler
```



The screenshot shows a web interface for creating a job. At the top, there are two tabs: 'sample-notebook.ipynb' and 'Notebook Jobs'. The main content area is titled 'Create Job' and contains several sections:

- Job name:** A text input field containing 'sample-notebook.ipynb'.
- Input file:** A text input field containing '/ Desktop/sample notebooks/sample-notebook.ipynb'.
- Compute type:** A dropdown menu with 'ml.m5.large' selected.
- Parameters:** A section with a '+' icon and a dropdown menu labeled 'Additional options'.
- Schedule:** Two radio button options: 'Run now' (which is selected) and 'Run on a schedule'.

At the bottom right, there are two buttons: 'Cancel' and 'Create'.



New!

SageMaker Distribution

A new open source distribution that supports most popular libraries and mutually compatible packages for machine learning, data science, and visualization to initialize runtimes on their local machines

```
docker pull public.ecr.aws/sagemaker/sagemaker-distribution:latest-cpu
```



The screenshot shows the GitHub repository for `aws/sagemaker-distribution`. The repository is private and has 12 commits. The main branch is selected. The repository contains several files and folders, including `build_artifacts/v0/v0.0`, `src`, `test`, `.gitignore`, `CODE_OF_CONDUCT.md`, `CONTRIBUTING.md`, `DEVELOPMENT.md`, `LICENSE`, `NOTICE`, `README.md`, `RELEASE.md`, `environment.yml`, and `pytest.ini`. The repository also has a README, Apache-2.0 license, Code of conduct, Security policy, Activity, 3 stars, 2 watching, and 3 forks.

Example use cases

Here are some examples on how you can try out one of our images.

Local environment, such as your laptop

The easiest way to get it running on your laptop is through the Docker CLI:

```
export ECR_IMAGE_ID='INSERT_IMAGE_YOU_WANT_TO_USE'
docker run -it \
  -p 8888:8888 \
  --user `id -u`:`id -g` \
  -v `pwd`/sample-notebooks:/home/sagemaker-user/sample-notebooks \
  $ECR_IMAGE_ID jupyter-lab --no-browser --ip=0.0.0.0
```

(If you have access to GPUs, you can pass `--gpus=all` to the Docker command.)

In the console output, you'll then see a URL similar to `http://127.0.0.1:8888/lab?token=foo`. Just open that URL in your browser, create a Jupyter Lab notebook or open a terminal, and start hacking.

Note that the sample command above bind mounts a directory in `pwd` inside the container. That way, if you were to re-create the container (say, to use a different version or CPU/GPU variant), any files you created within that directory (such as Jupyter Lab notebooks) will persist.



Innovate faster with managed infrastructure and the broadest and most complete set of ML capabilities



Amazon SageMaker

Quick start your development with **fully managed**, secure Jupyter Notebooks in the cloud.

Scale your compute resources up or down with the broadest selection of compute-optimized and GPU-accelerated instances in the cloud.

Efficiently collaborate with teams across all steps of your ML lifecycle by editing the same notebooks together in real time.

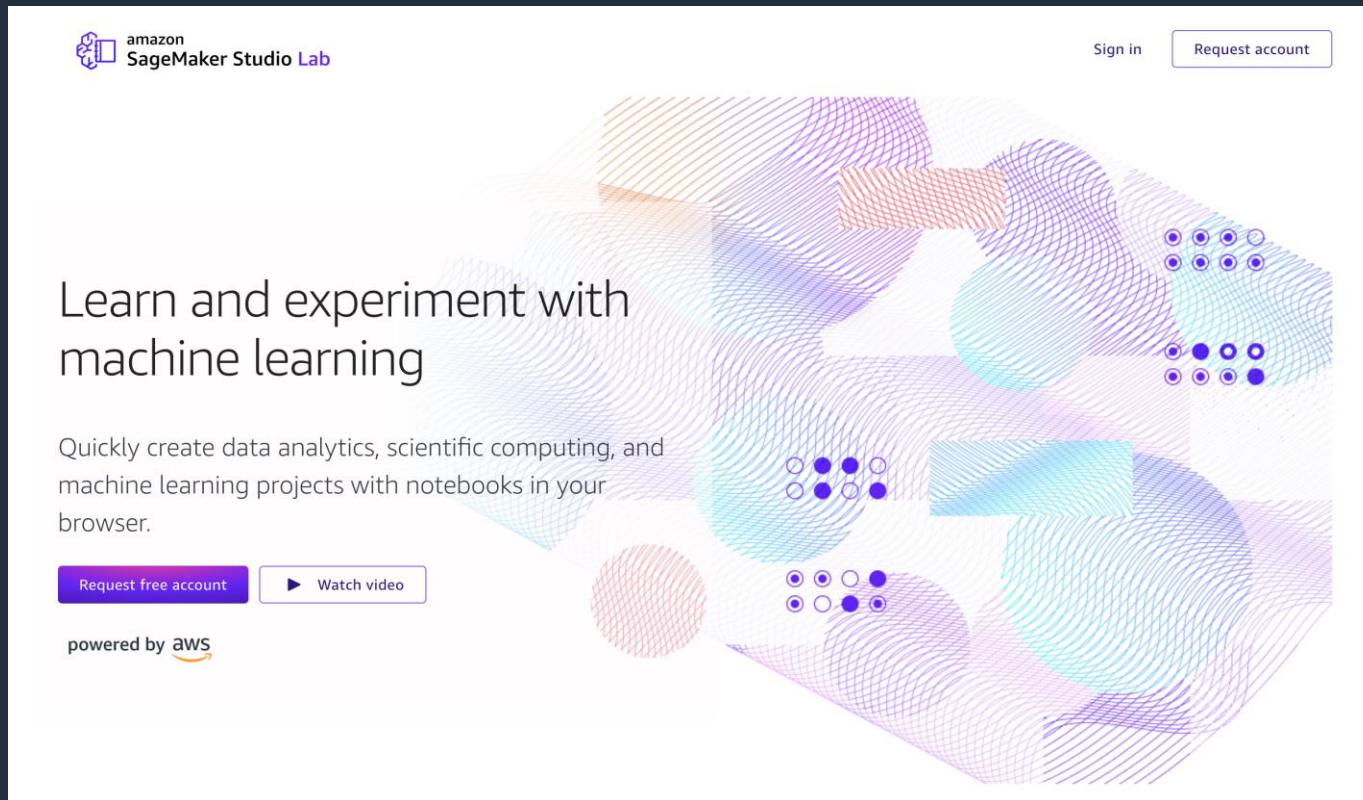
Go from data to insights up to 2X faster with optimizations for popular frameworks and packages such as Spark, NumPy and Scikit-learn.

Access, label, and process large amounts of structured data (tabular data) and unstructured data (photo, video, geospatial, and audio) for ML.

Reduce training/processing time from hours to minutes with optimized infrastructure.

Amazon SageMaker Studio Lab

A NO-CHARGE, NO-CONFIGURATION SERVICE THAT ENABLES DATA SCIENTISTS TO LEARN AND EXPERIMENT WITH MACHINE LEARNING



amazon SageMaker Studio Lab

Sign in Request account

Learn and experiment with machine learning

Quickly create data analytics, scientific computing, and machine learning projects with notebooks in your browser.

Request free account Watch video

powered by AWS

Create an account with an email address – **free**

No setup or configuration required

15 GBs to save your work projects

As many compute sessions as you need –
CPU (12 hrs) / GPU (4 hrs)

Access any notebook on GitHub

Migrate to SageMaker Studio when ready

Use the referral code **JupyterCon-13519**
to get immediate access



Amazon SageMaker Studio



Amazon SageMaker Studio is a web-based, integrated development environment (IDE) to **prepare data, build, train, deploy, and monitor** your machine learning models.

Amazon SageMaker Studio Notebooks

FAST-START COLLABORATIVE NOTEBOOKS FOR DATA SCIENCE, ML AND ANALYTICS

The screenshot displays the Amazon SageMaker Studio Notebook interface. The main editor shows a Python notebook titled 'linear-regression-concise.ipynb'. The notebook content includes a 'Training' section with explanatory text and a list of steps: 'Generate predictions by calling net(X) and calculate the loss l (the forward propagation)', 'Calculate gradients by running the backpropagation', and 'Update the model parameters by invoking our optimizer'. Below this is a code cell with a training loop. A 'Summary' section follows, mentioning 'Using TensorFlow's In TensorFlow, the layers and common'. The interface also features a 'VARIABLES' panel on the right, a 'CALLSTACK' panel, and a 'BREAKPOINTS' panel. A data insights panel is overlaid on the bottom right, showing a table of data with columns 'survived', 'pclass', 'name', and 'sex'. The table lists various passengers, including 'Lieskes, Mr. Rene Aimo' and 'Lester, Mr. James'. The insights panel includes a 'Data quality' section with a 'Target Column Insights' table showing 'Too few instances per class' as 'High'. It also has 'SUGGESTED TRANSFORMS' like 'Replace rare target values' and 'Drop rare target values', and 'Feature Column Insights' showing 'Disguised missing values' as 'Medium'. The AWS logo is visible in the bottom left corner.

Quick start with Single Sign-On (SSO) or IAM
Access your notebooks in seconds via pre-signed URLs

Elastic
Dial up or down compute resources on the flight

Collaborative
Attach Git and AWS CodeCommit repos or simply share notebooks with a single click. Create shared Spaces for real-time collaboration on notebooks

Perform data engineering, ML, and analytics workflows in same notebook
Native integrations with Amazon EMR, Glue, S3, and more. Interactively prepare data at scale using cluster-based and serverless compute environments for Spark and Ray

Customizable
Bring your own docker image with custom packages, and libraries. Use lifecycle hooks to customize development environments.

New!

Accelerate local ML code conversion to training jobs

Execute ML code authored in your local notebooks along with the associated runtime dependencies as large-scale ML model training jobs with minimal code changes

```
pip install sagemaker
```



Python

```
import boto3
import sagemaker
from sagemaker.remote_function import remote

sm_session = sagemaker.Session(boto_session=boto3.session.Session(region_name="us-west-2"))
settings = dict(
    sagemaker_session=sm_session,
    role=<IAM_ROLE_NAME>
    instance_type="ml.m5.xlarge",
)
@remote(**settings)
def divide(x, y):
    return x / y
if __name__ == "__main__":
    print(divide(2, 3.0))
```

Python

```
from sagemaker.remote_function import remote

@remote(instance_type="ml.g4dn.xlarge",dependencies = "./environment.yml")
def train_hf_model(
    train_input_path,test_input_path,s3_output_path = None,
    *,epochs = 1, train_batch_size = 32, eval_batch_size = 64,
    warmup_steps = 500,learning_rate = 5e-5
):
    model_name = "distilbert-base-uncased"
    model = AutoModelForSequenceClassification.from_pretrained(model_name)
    ... <TRUCNATED>
    return os.path.join(s3_output_path, model_dir), eval_result
```


Demo





Thank you!

See you at the AWS Booth
5/10 – 5/12 8:00am-4:30pm Level S2

Giuseppe Angelo Porcelli
gianpo@amazon.com