SVKM'S NMIM'S Nilkamal School of Mathematics, Applied Statistics & Analytics

Master of Science (Data Science)

Practical-09 ML Model implementation and deployment using Sage Maker

Date: -17/04/2024 **Submission Date: - 18/04/2024**

Writeup: -

- **AWS Sage maker**
- **Features of Sage maker**
- **Components of Sage maker**

AWS Sage Maker:

Definition: AWS Sage Maker is a fully managed service provided by Amazon Web Services (AWS) that simplifies the process of building, training, deploying, and managing machine learning (ML) models at scale. It offers a comprehensive set of tools and services to support the entire machine learning lifecycle, from data preparation and model training to deployment and monitoring. Sage Maker abstracts away the complexities of managing infrastructure, allowing developers and data scientists to focus on building and deploying ML models without worrying about the underlying infrastructure setup and maintenance.

Features of Sage maker

a. End-to-End Machine Learning Workflow: Sage Maker provides a seamless end-to-end workflow for machine learning projects, encompassing data processing, model training, model optimization, deployment, and monitoring. This integrated approach streamlines the machine learning lifecycle and improves productivity.

b. Fully Managed Service: As a fully managed service, Sage Maker handles infrastructure provisioning, scaling, and maintenance automatically. Users can access Sage Maker's capabilities through the AWS Management Console or APIs without the need to manage servers or infrastructure manually.

c. Broad Range of Algorithms and Frameworks: Sage Maker offers support for a wide range of built-in algorithms and popular machine learning frameworks, including TensorFlow, PyTorch, MXNet, and scikit-learn. This flexibility allows users to choose the tools and technologies that best suit their requirements.

d. AutoML Capabilities: Sage Maker simplifies the process of model development with AutoML capabilities, including automatic model tuning. AutoML automates the optimization of model hyperparameters to improve model performance, reducing the need for manual tuning.

e. Notebook Instances: SageMaker provides managed Jupyter notebook instances for interactive data exploration, experimentation, and model development. Users can create, edit, and run Jupyter notebooks directly within Sage Maker, facilitating collaboration and iteration

Components of Sage Maker

a. Notebook Instances: Managed Jupyter notebook instances for interactive data exploration, experimentation, and model development. Users can create, edit, and run Jupyter notebooks within Sage Maker's managed environment.

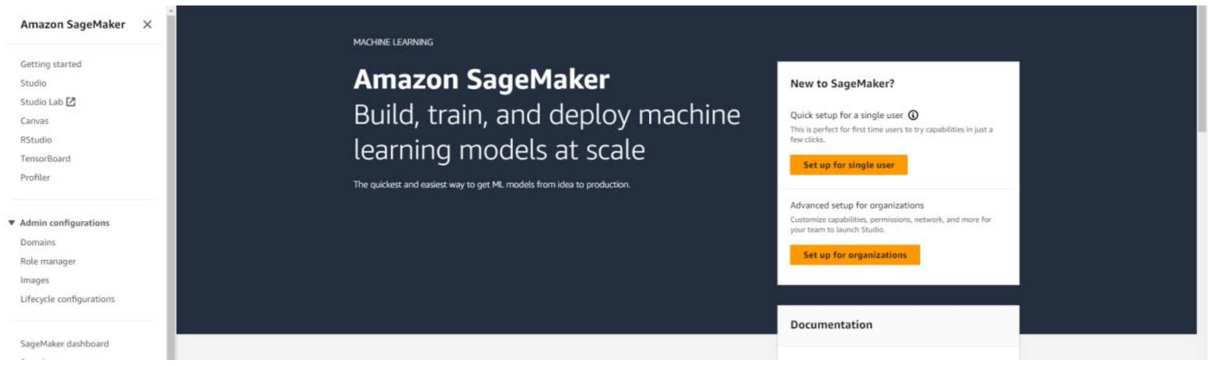
b. Training Jobs: Infrastructure and resources for training machine learning models using data stored in Amazon S3 or other data sources. Sage Maker manages the provisioning and scaling of resources to execute training jobs efficiently.

c. Model Hosting: Built-in model hosting and deployment capabilities for deploying trained models as RESTful endpoints. Users can deploy models into production environments with ease, enabling real-time predictions and inference.

d. Endpoint Configurations: Configuration settings for deploying models as endpoints, including instance type, scaling behaviour, and encryption options. Users can customize endpoint configurations to meet specific performance and security requirements

Implement and deploy ML Model using Sage maker

Step 1 :login to the amazon console and go on the amazon sage maker.



Amazon SageMaker

Getting started
Studio
Studio Lab
Canvas
RStudio
TensorBoard
Profiler

Admin configurations
Domains
Role manager
Images
Lifecycle configurations

SageMaker dashboard

MACHINE LEARNING

Amazon SageMaker
Build, train, and deploy machine learning models at scale

The quickest and easiest way to get ML models from idea to production.

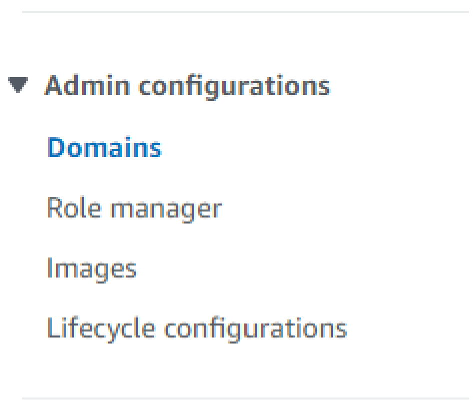
New to SageMaker?

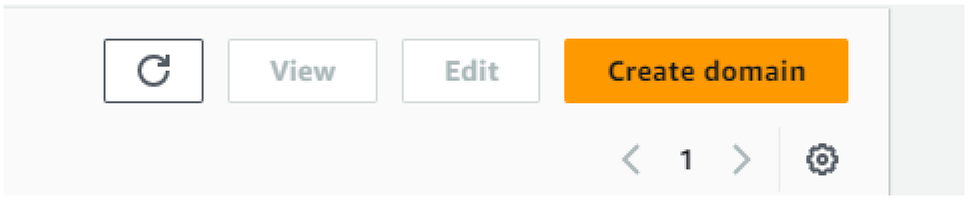
Quick setup for a single user
This is perfect for first time users to try capabilities in just a few clicks.
Set up for single user

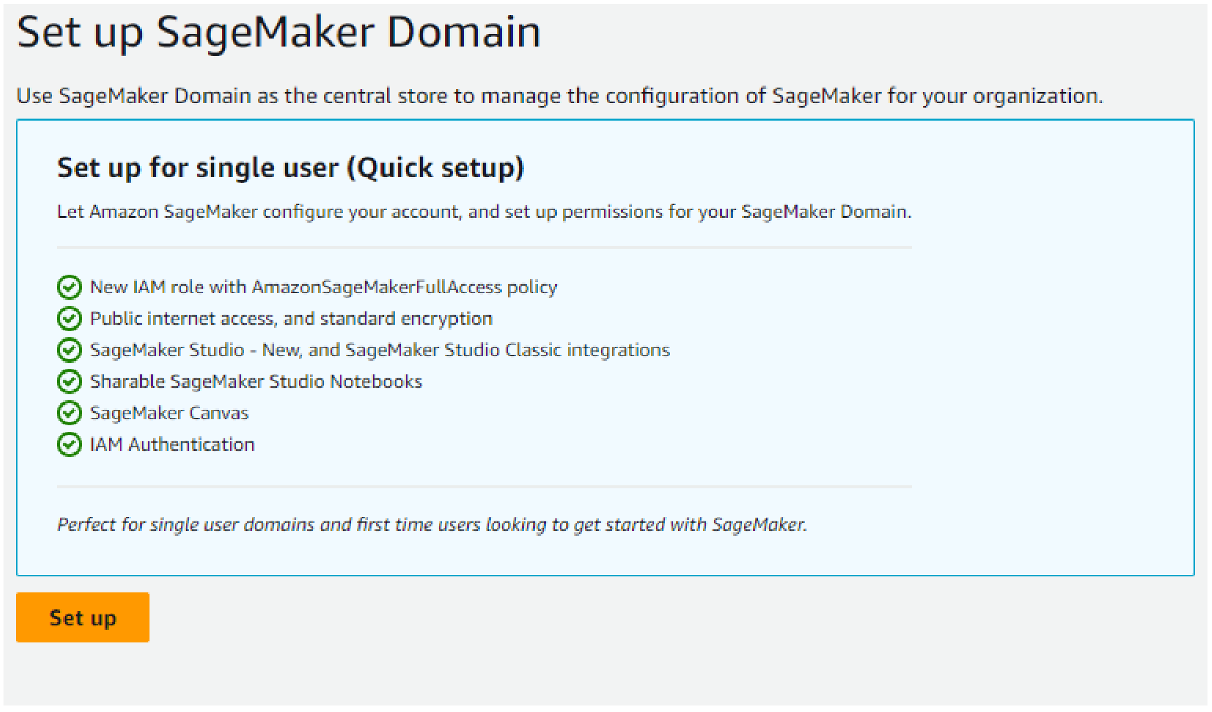
Advanced setup for organizations
Customize capabilities, permissions, network, and more for your team to launch Studio.
Set up for organizations

Documentation

Step 2 : Then select domain and create a new domain and click on the set up.






Set up SageMaker Domain

Use SageMaker Domain as the central store to manage the configuration of SageMaker for your organization.

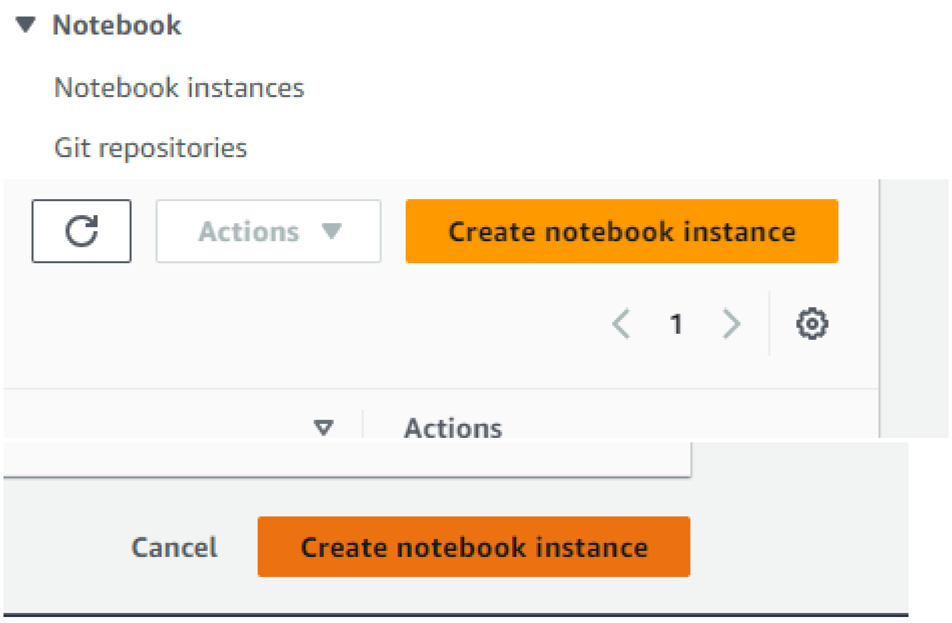
Set up for single user (Quick setup)

Let Amazon SageMaker configure your account, and set up permissions for your SageMaker Domain.

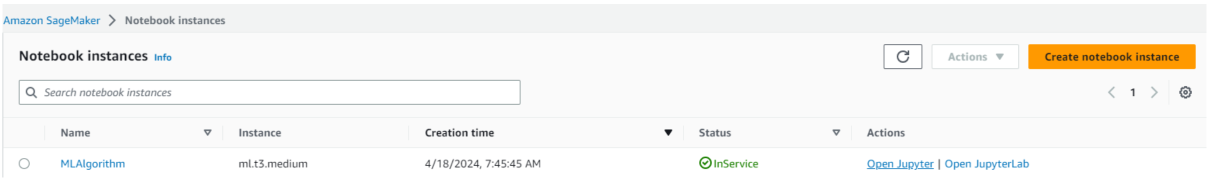
- ✓ New IAM role with AmazonSageMakerFullAccess policy
- ✓ Public internet access, and standard encryption
- ✓ SageMaker Studio - New, and SageMaker Studio Classic integrations
- ✓ Sharable SageMaker Studio Notebooks
- ✓ SageMaker Canvas
- ✓ IAM Authentication

Perfect for single user domains and first time users looking to get started with SageMaker.

Step 3 : click on the notebook and create a new notebook instance.



Notebook is successfully created.



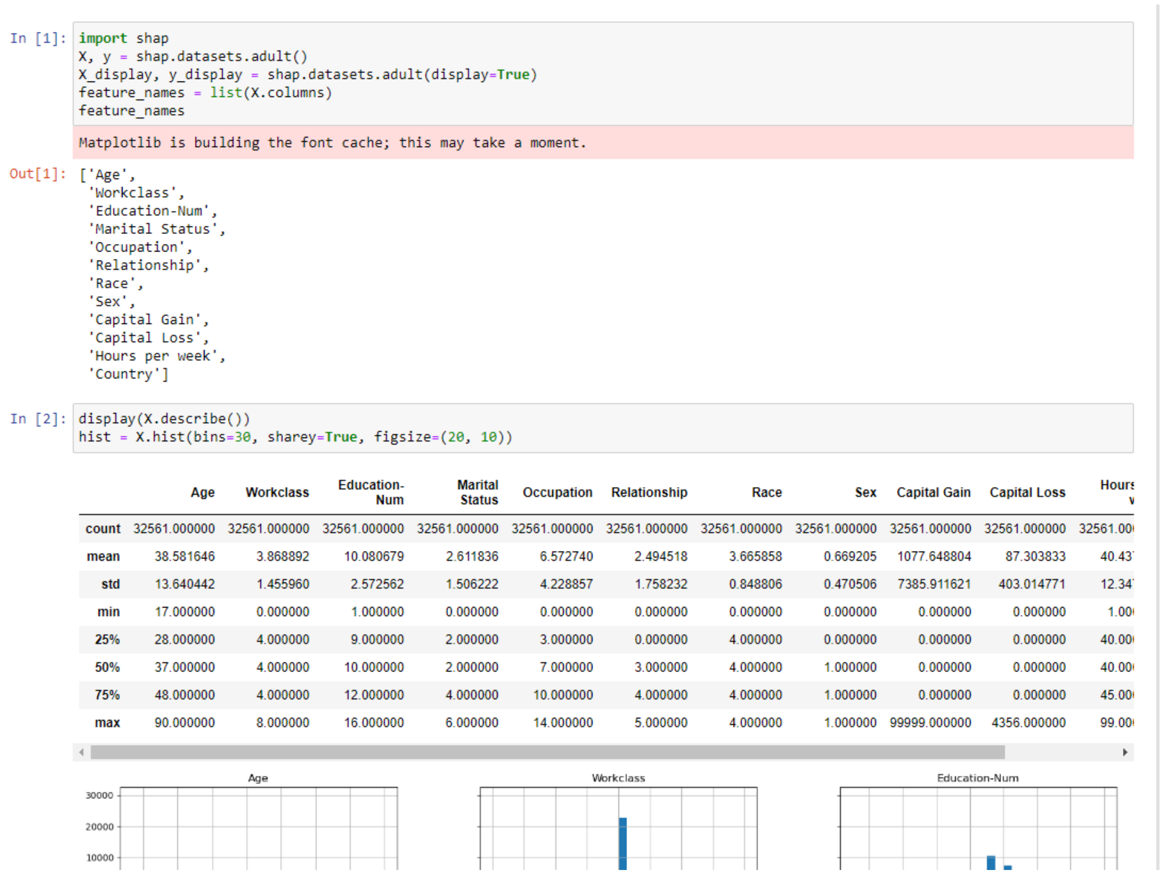
Amazon SageMaker > Notebook instances

Notebook instances Info

Search notebook instances

	Name	Instance	Creation time	Status	Actions
○	MLAlgorithm	ml.t3.medium	4/18/2024, 7:45:45 AM	InService	Open Jupyter | Open JupyterLab

Step 4 : after succesfully creating a notebook instance open the python and train the model.



```
In [1]: import shap
        X, y = shap.datasets.adult()
        X_display, y_display = shap.datasets.adult(display=True)
        feature_names = list(X.columns)
        feature_names
```

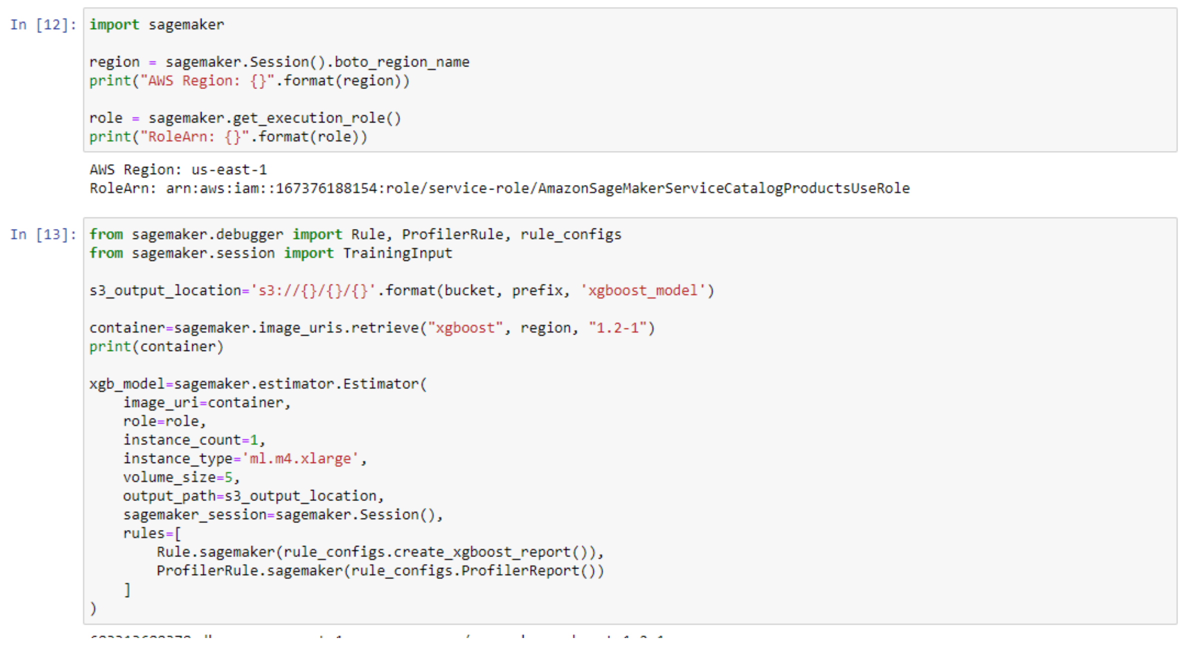
```
Matplotlib is building the font cache; this may take a moment.
```

```
Out[1]: ['Age',
         'Workclass',
         'Education-Num',
         'Marital Status',
         'Occupation',
         'Relationship',
         'Race',
         'Sex',
         'Capital Gain',
         'Capital Loss',
         'Hours per week',
         'Country']
```

```
In [2]: display(X.describe())
        hist = X.hist(bins=30, sharey=True, figsize=(20, 10))
```

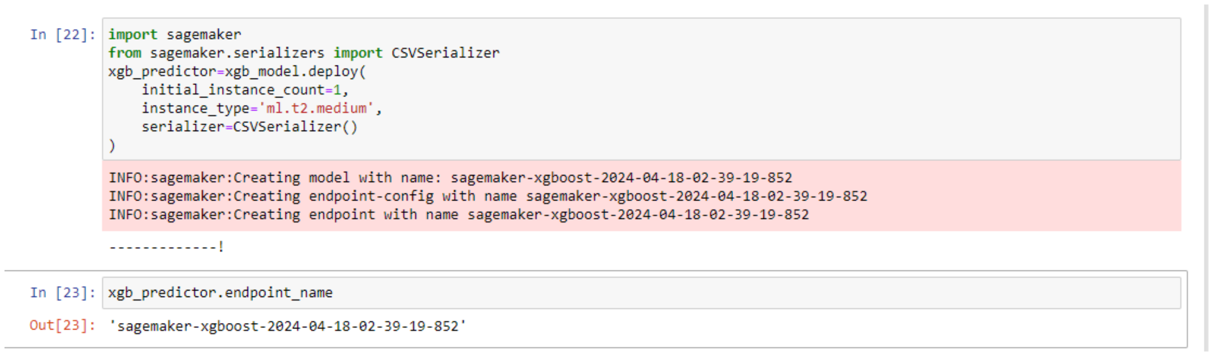
	Age	Workclass	Education-Num	Marital Status	Occupation	Relationship	Race	Sex	Capital Gain	Capital Loss	Hours
count	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.00
mean	38.581646	3.868892	10.080679	2.611836	6.572740	2.494518	3.665858	0.669205	1077.648804	87.303833	40.43
std	13.640442	1.455960	2.572562	1.506222	4.228857	1.758232	0.848806	0.470506	7385.911621	403.014771	12.34
min	17.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.00
25%	28.000000	4.000000	9.000000	2.000000	3.000000	0.000000	4.000000	0.000000	0.000000	0.000000	40.00
50%	37.000000	4.000000	10.000000	2.000000	7.000000	3.000000	4.000000	1.000000	0.000000	0.000000	40.00
75%	48.000000	4.000000	12.000000	4.000000	10.000000	4.000000	4.000000	1.000000	0.000000	0.000000	45.00
max	90.000000	8.000000	16.000000	6.000000	14.000000	5.000000	4.000000	1.000000	99999.000000	4356.000000	99.00

Step 5 : Deploy the model on amazon ec2.



```
In [12]: import sagemaker

         region = sagemaker.Session().boto_region_name
         print("AWS Region: {}".format(region))

         role = sagemaker.get_execution_role()
         print("RoleArn: {}".format(role))

         AWS Region: us-east-1
         RoleArn: arn:aws:iam::167376188154:role/service-role/AmazonSageMakerServiceCatalogProductsUseRole

In [13]: from sagemaker.debugger import Rule, ProfilerRule, rule_configs
         from sagemaker.session import TrainingInput

         s3_output_location='s3://{}/{}/{}'.format(bucket, prefix, 'xgboost_model')

         container=sagemaker.image_uris.retrieve("xgboost", region, "1.2-1")
         print(container)

         xgb_model=sagemaker.estimator.Estimator(
             image_uri=container,
             role=role,
             instance_count=1,
             instance_type='ml.m4.xlarge',
             volume_size=5,
             output_path=s3_output_location,
             sagemaker_session=sagemaker.Session(),
             rules=[
                 Rule.sagemaker(rule_configs.create_xgboost_report()),
                 ProfilerRule.sagemaker(rule_configs.ProfilerReport())
             ]
         )
```

Step 6 : after the model is successfully deployed.



```
In [22]: import sagemaker
         from sagemaker.serializers import CSVSerializer
         xgb_predictor=xgb_model.deploy(
             initial_instance_count=1,
             instance_type='ml.t2.medium',
             serializer=CSVSerializer()
         )

         INFO:sagemaker:Creating model with name: sagemaker-xgboost-2024-04-18-02-39-19-852
         INFO:sagemaker:Creating endpoint-config with name sagemaker-xgboost-2024-04-18-02-39-19-852
         INFO:sagemaker:Creating endpoint with name sagemaker-xgboost-2024-04-18-02-39-19-852

         -------------!

In [23]: xgb_predictor.endpoint_name

Out[23]: 'sagemaker-xgboost-2024-04-18-02-39-19-852'
```