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

Master of Science (Data Science)

Practical-09 ML Model implementation and deployment using Sagemaker

Writeup:-

- **AWS Sagemaker**
- **Features of Sagemaker**
- **Components of Sagemaker**

implement and deploy ML Model using Sagemaker

Purva – A018

Amazon SageMaker

Amazon SageMaker is a fully managed service that provides developers and data scientists with the tools to build, train, and deploy machine learning models at scale. It offers a variety of features and capabilities to streamline the machine learning workflow, including data labeling, model training, hyperparameter optimization, and hosting for model deployment.

With SageMaker, users can choose from pre-built algorithms and frameworks, such as TensorFlow, PyTorch, and Apache MXNet, or bring their own custom algorithms. It also integrates with other AWS services like S3 for data storage, AWS Glue for data preparation, and AWS Lambda for serverless computing.

One of the key advantages of SageMaker is its scalability and flexibility, allowing users to easily scale resources up or down based on demand and pay only for what they use. This makes it suitable for a wide range of use cases, from experimentation and prototyping to production-grade machine learning applications.

Features of Sagemaker:

Amazon SageMaker offers a comprehensive set of features designed to streamline the end-to-end machine learning workflow. Here are some of the key features:

Notebook Instances: SageMaker provides fully managed Jupyter notebook instances that allow data scientists and developers to easily explore, preprocess, and visualize data.

Data Labeling: SageMaker includes built-in data labeling tools that enable the annotation and labeling of datasets for supervised learning tasks. This feature helps in creating high-quality labeled datasets for training machine learning models.

Built-in Algorithms: SageMaker offers a library of built-in algorithms for common machine learning tasks such as classification, regression, clustering, and anomaly detection. These algorithms are optimized for performance and can be easily used with minimal configuration.

Custom Algorithms: Users can bring their own custom algorithms and frameworks to SageMaker and train models using their preferred libraries such as TensorFlow, PyTorch, and scikit-learn.

Model Training: SageMaker provides managed training environments with distributed training capabilities, allowing users to train models at scale across multiple instances with ease. It also supports automatic model tuning for hyperparameter optimization.

Model Deployment: Once a model is trained, SageMaker makes it easy to deploy it to production using managed hosting services. Models can be deployed as real-time endpoints or batch transform jobs for offline inference.

Monitoring and Logging: SageMaker offers built-in monitoring and logging capabilities to track model performance, detect drift, and troubleshoot issues in real-time.

Security and Compliance: SageMaker provides robust security features such as encryption at rest and in transit, VPC support, and fine-grained access control using IAM roles. It also supports compliance with regulatory requirements such as GDPR and HIPAA.

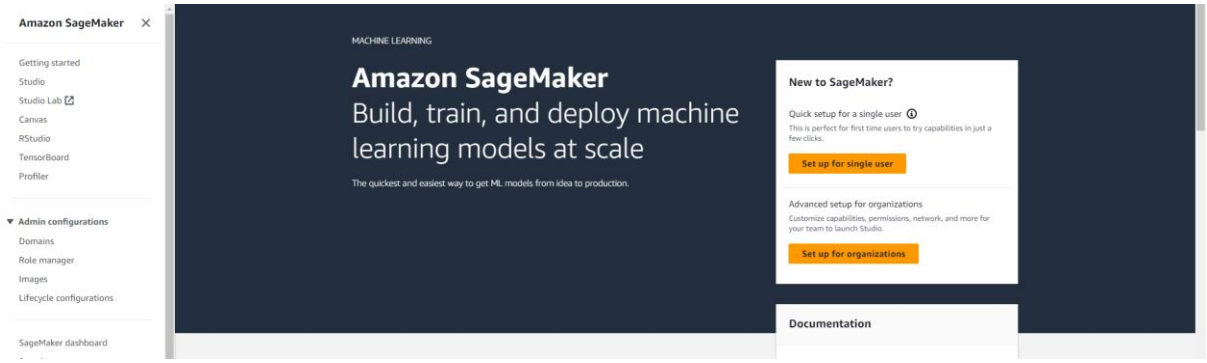
Integration with AWS Services: SageMaker seamlessly integrates with other AWS services such as S3 for data storage, AWS Glue for data preparation, AWS Lambda for serverless computing, and AWS Step Functions for orchestrating machine learning workflows.

Cost Optimization: SageMaker offers flexible pricing options, including pay-as-you-go pricing and savings plans, to help users optimize costs based on their usage patterns and requirements.

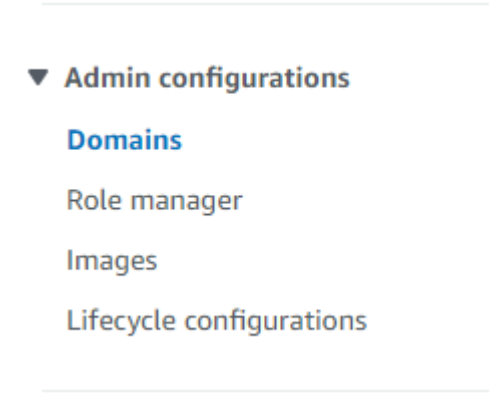
Components of sagemaker:

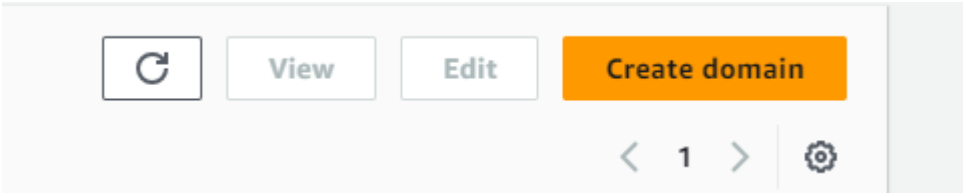
Amazon SageMaker is a comprehensive platform for building, training, and deploying machine learning models on AWS. It comprises several key components designed to streamline the end-to-end machine learning workflow. First, it offers fully managed Jupyter notebook instances and SageMaker Studio, an integrated development environment (IDE), for data exploration, experimentation, and model development. SageMaker Data Wrangler simplifies data preparation tasks by providing a visual interface for data cleaning and transformation. Users can leverage built-in algorithms or bring their own custom models packaged in Docker containers for training. SageMaker provides managed training environments with distributed training capabilities and supports automatic hyperparameter optimization. Once trained, models can be deployed as real-time endpoints or batch transform jobs using managed hosting services. SageMaker Model Monitor continuously monitors model performance and detects concept drift and data quality issues. Experiments and SageMaker Feature Store facilitate experiment tracking and feature management, while SageMaker Pipelines allow for the automation of end-to-end machine learning workflows. Together, these components offer a powerful and flexible platform for machine learning practitioners to develop and deploy models at scale.

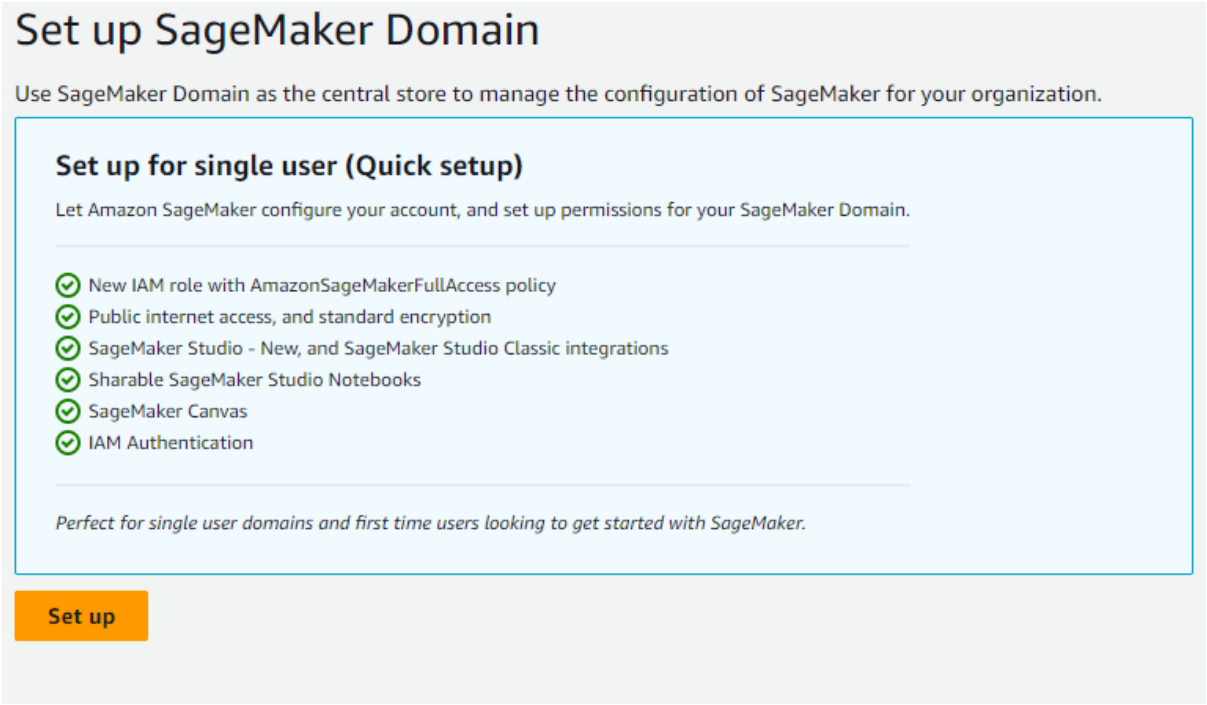
Step 1 :login to the amazon console and go on the amazon sagemaker.



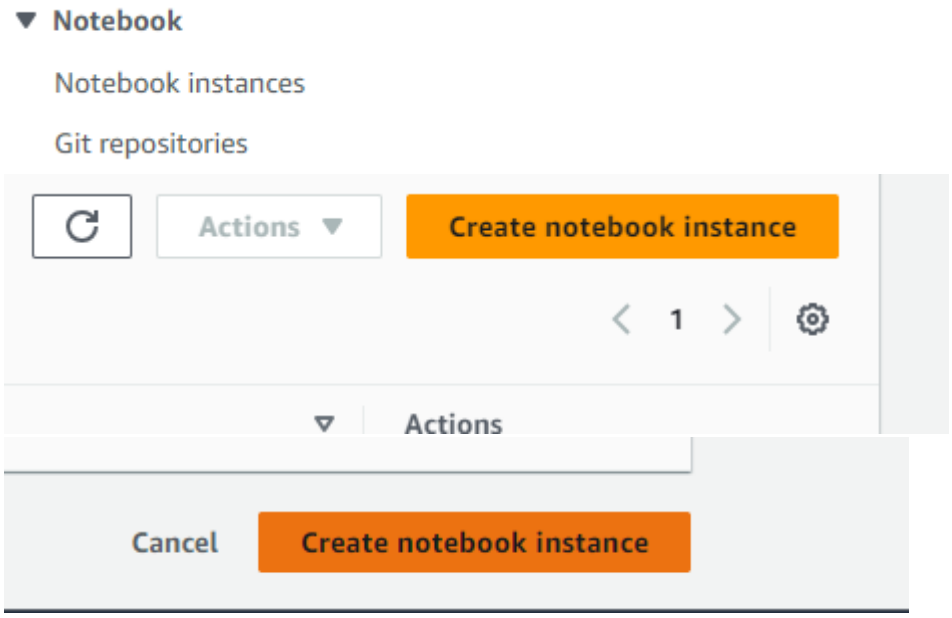
Step 2 : Then select domain and create a new domain and click on the set up.



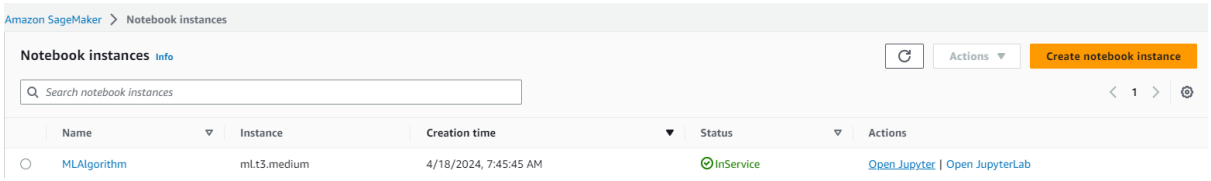




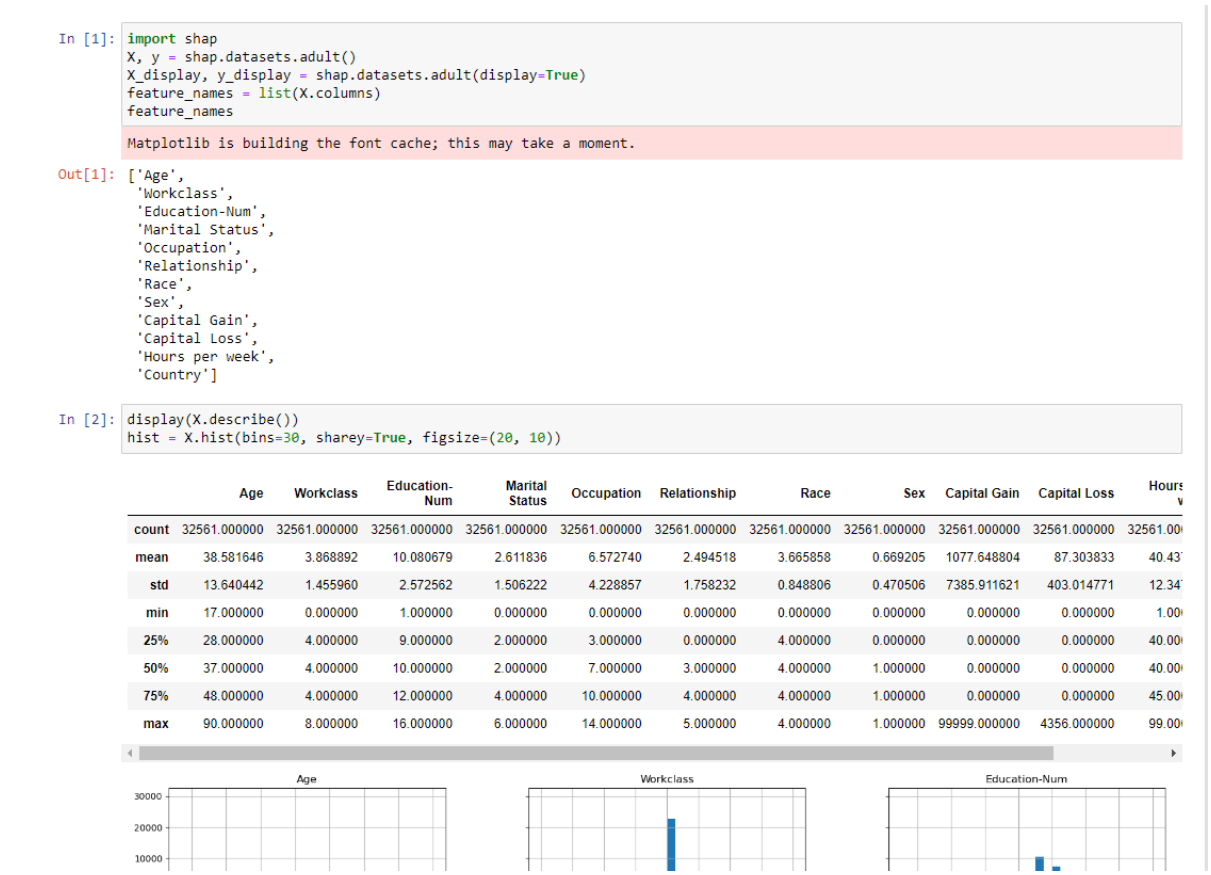
Step 3 : click on the notebook and create a new notebook instance.



Notebook is successfully created.



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'
```