Trabajo PEC4

Infraestructuras Computacionales para Procesamiento de Datos Masivos

Irene J. Ventura Farias

Resumen

AWS Sagemaker es un servicio en la nube que permite a los científicos de datos y desarrolladores crear, entrenar y desplegar modelos de machine learning de forma rápida y sencilla.

En este trabajo hablaremos de las características principales de este servicio, así como también de sus componentes y haremos una demostración de Sagemaker implementando el algoritmo XGBoost para la clasificación de transacciones de pago realizadas por diferentes clientes.

Características

- Altamente Escalable.
- Entrenamiento rápido y optimización de hiperparámetros.
- Alta seguridad de los datos

Componentes Principales

- **Servicio de Notebooks:** permite crear instancias para trabajar con datos desde un Jupyter o un JupyterLab

- **Servicio de Entrenamiento:** permite crear jobs que se ejecutan en la nube y monitorear el progreso.

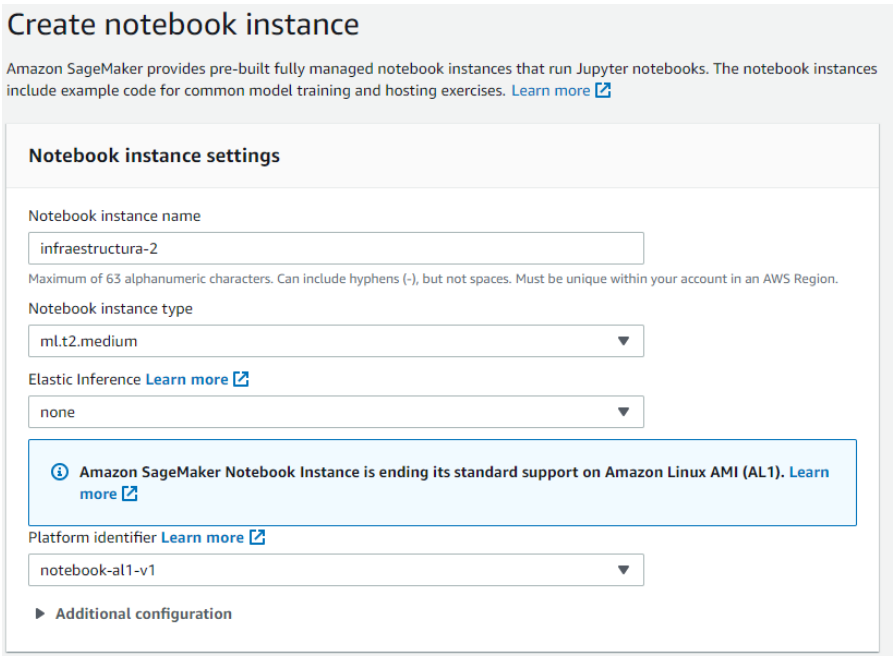
- **Servicio de Hosting:** ofrece este servicio que se encarga de exponer este servicio como una API (denominada como entrypoint) para recibir peticiones en tiempo real.

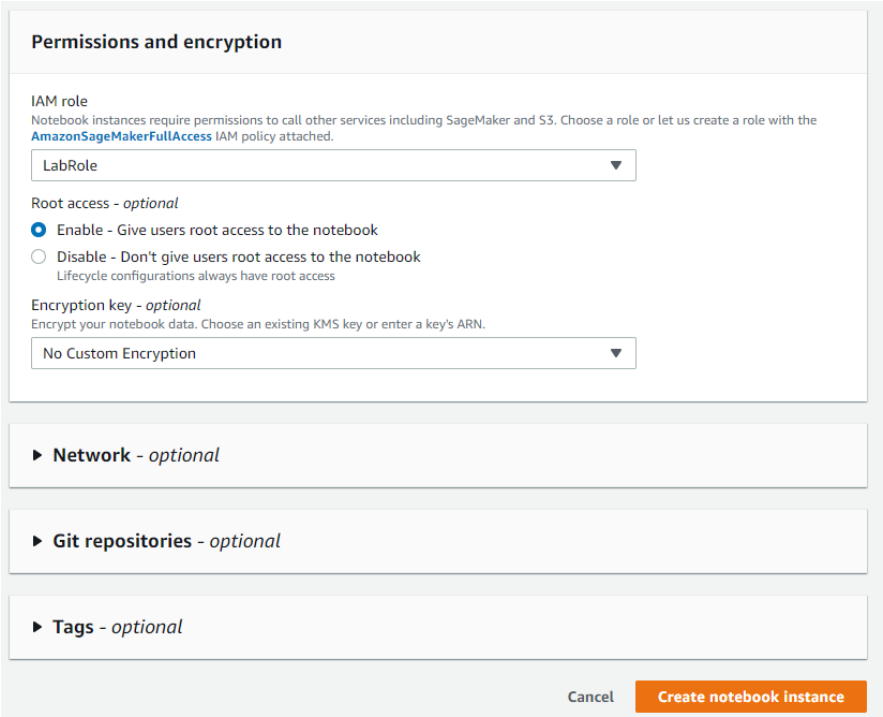
Tienes la posibilidad de usar los tres componentes o adaptarlos al flujo de trabajo del proyecto que se está desarrollando.

DEMO: Construcción, Entrenamiento y Despliegue de un modelo en Sagemaker

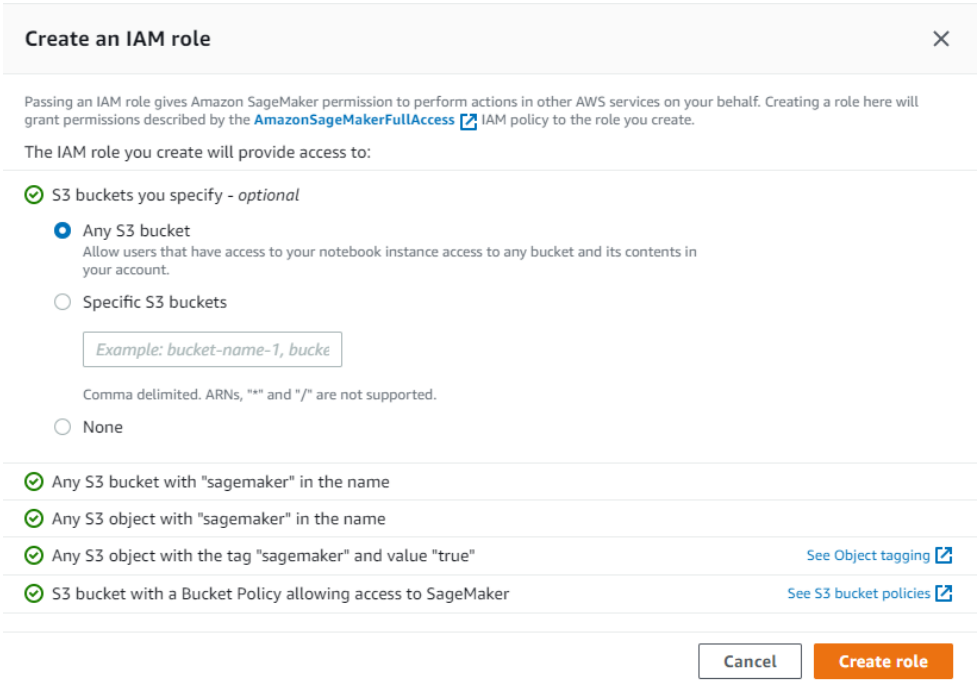
1. **Crear una instancia de notebooks (Jupyter/JupyterLab):**

   Asignamos un nombre y mantenemos el tipo de instancia en la más baja, dejamos el **elastic inference** por defecto (None) y seleccionamos un IAM role para nuestra instancia

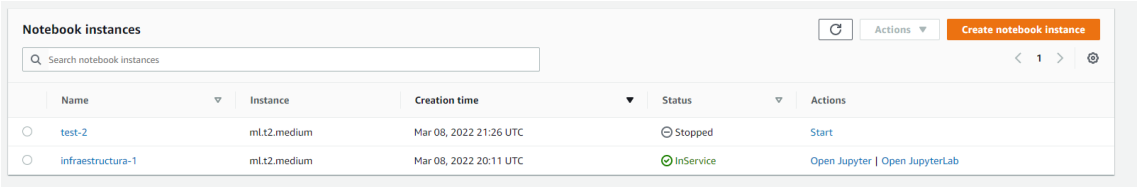




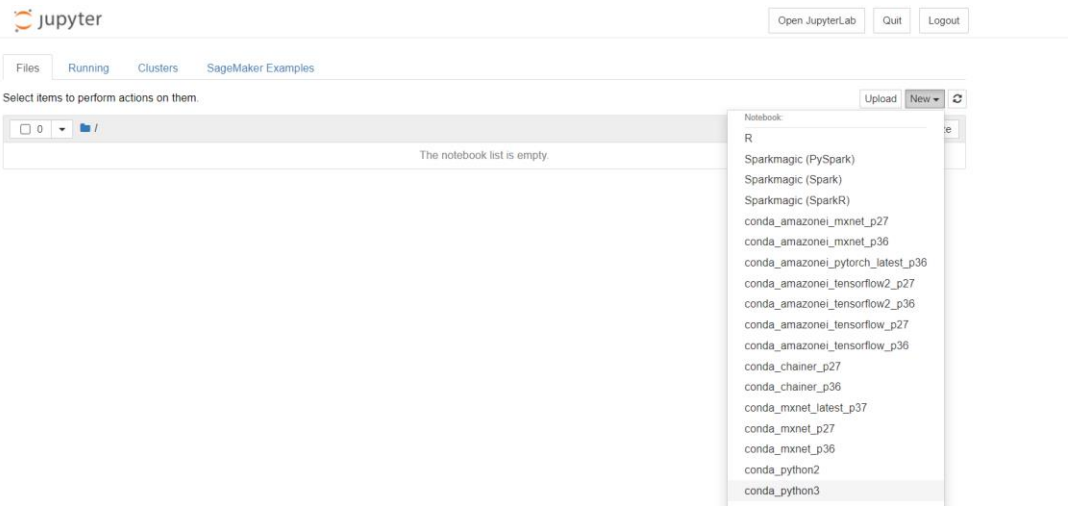
NOTA: Debemos asegurarnos de que el Role que asignemos tenga permisos para acceder a Amazon S3 para poder cargar datos de forma segura. En caso de que no se tenga un rol configurado previamente, tendremos que crear el rol:



Una vez creada la instancia se procesará la solicitud y sabremos que esta lista cuando la instancia tenga como estatus “InService”



A continuación, procederemos a abrir en Jupyter de la instancia que hemos creado para generar el notebook donde desarrollaremos el código e implementaremos las librerías necesarias. Para nuestra prueba el kernel debe ser **conda_3**



2. Crear Buckets en S3

Existen dos formas de trabajar con los buckets de AWS S3, podemos crearlos a través del servidor o podemos usar la librería boto3, el cual nos permitirá acceder a S3 a través del notebook puesto que el role que hemos asignado a la instancia nos permite trabajar con todos los buckets que tengamos creados.

In [11]:
```
import boto3
bucket_name = 'pec4'
s3 = boto3.resource('s3')
try:
    if  my_region == 'us-east-1':
        s3.create_bucket(Bucket=bucket_name)
    else:
        s3.create_bucket(Bucket=bucket_name, CreateBucketConfiguration={ 'LocationConstraint': my_region })
    print('S3 bucket created successfully')
except Exception as e:
    print('S3 error: ',e)
```

```
S3 bucket created successfully
```

En nuestro caso hemos creado un bucket llamado “pec4” y le hemos especificado la región con la que estamos trabajando. Luego procedemos a cargar los datos desde el mismo notebook y comprobaremos que estén en S3.

```
import urllib
import numpy as np
import pandas as pd

try:
    urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-train-deploy-machine-learning-model-sagemaker/bank_clean.27f
    print('Success: downloaded bank_clean.csv.')
except Exception as e:
    print('Data load error: ',e)

try:
    model_data = pd.read_csv('./bank_clean.csv',index_col=0)
    print('Success: Data loaded into dataframe.')
except Exception as e:
    print('Data load error: ',e)
```

```
Success: downloaded bank_clean.csv.
Success: Data loaded into dataframe.
```

```
model_data.sample(10)
```

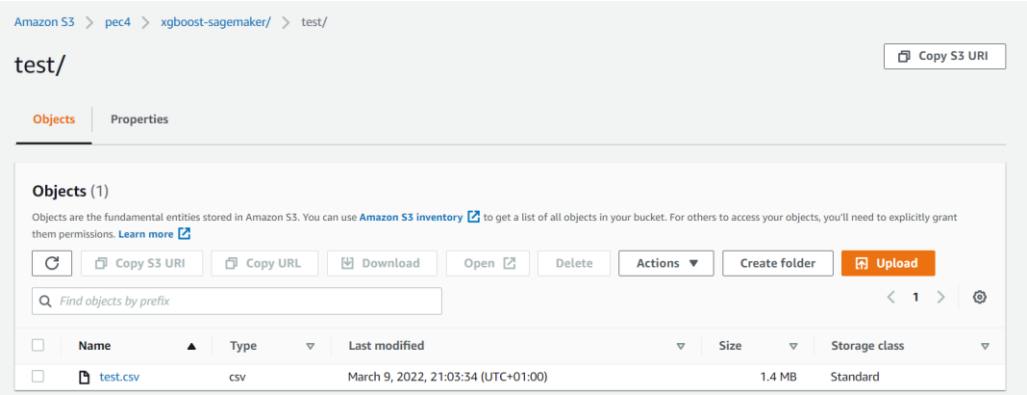
	transaction_category	receiver_id	sender_id	amount	timestamp
23953	Shopping	4726456211052484	4562609718829252	128.57	2021-01-31 07:27:30
51315	Health and Fitness	4791473909425013	4076701783821402	90.17	2021-03-09 12:29:17
40537	Shopping	4935230535165968	4570333519404674	33.32	2021-02-24 23:40:23
11023	Entertainment	4557640914814233	4638775639990000	38.09	2021-04-10 19:41:07
92398	Travel	4525200848056242	4948265227536980	176.57	2021-02-13 12:18:16
26589	Shopping	4584938333411384	4649183650822086	106.27	2021-04-20 23:12:06
86365	Auto and Transport	4983705359326078	4863225422212143	137.59	2021-02-10 16:38:56
90419	Auto and Transport	4840532524995824	4278105029572579	81.51	2021-01-28 11:15:56
48681	Shopping	4271422551678915	4165389274484492	132.56	2021-02-21 23:46:13
87508	Auto and Transport	4810621459099474	4783824010517322	90.83	2021-02-26 22:49:26

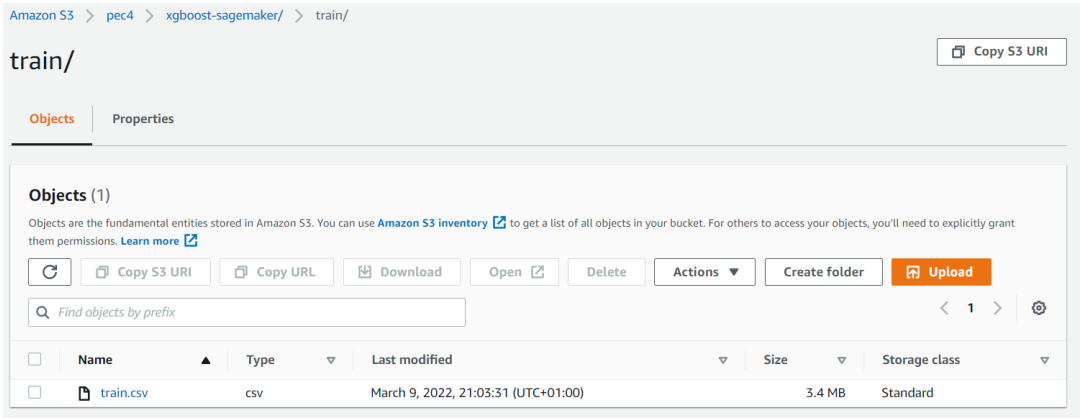
Finalmente, hemos separaremos los datos de entrenamiento (70%) y los de prueba (20%).

NOTA: es importante mencionar que desde el notebook también podemos realizar la preparación de datos. En nuestro caso como la prioridad es demostrar cómo funciona este servicio, se ha omitido este paso.

```
train_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
print(train_data.shape, test_data.shape)
```

```
(28831, 61) (12357, 61)
```





NOTA: También, debemos tener en cuenta que, si vamos a trabajar con los algoritmos que nos proporciona Sagemaker, todos esperan que la primera columna del dataset de entrenamiento sea la variable dependiente (Y)

3. Construir y Entrenar el modelo

Como vamos a hacer uso del modelo XGBoost que nos ofrece Sagemaker debemos tener en cuenta que todos los algoritmos incorporados a este servicio están en forma de contenedores y por eso debemos acceder a ellos a través de **image_uri.retrive ,** especificando como mínimo el nombre de la región.

Get the XGBoost sagemaker image

In [24]:
```
container = sagemaker.image_uris.retrieve(region=region, framework="xgboost", version="1.2-2")
```

El estimador se encarga de manejar las tareas de entrenamiento y despliegue, especificándole el algoritmo que queremos, los hiperparámetros del modelo y especificar las rutas de los datos de entrenamiento y los datos de prueba almacenados en S3

We define the XGBoost model

In [26]:
```
xgb = sagemaker.estimator.Estimator(
    container,
    role,
    instance_count=1,
    instance_type="ml.m4.xlarge",
    output_path="s3://{}/{}/output".format(s3_bucket, bucket_prefix),
    sagemaker_session=sagemaker_session,
)
```

Set the parameters

In [27]:
```
xgb.set_hyperparameters(
    max_depth=5,
    eta=0.2,
    gamma=4,
    min_child_weight=6,
    subsample=0.8,
    objective="multi:softprob",
    num_class=19,
    verbosity=0,
    num_round=100,
)
```

And train the model

In [*]:
```
xgb.fit({"train": s3_input_train, "validation": s3_input_validation})
```

```
2022-03-09 21:10:59 Starting - Starting the training job...ProfilerReport-1646860259: InProgress
```

4. Despliegue del modelo

Para levantar el modelo entrenado debemos especificar el numero de instancias y el tipo de instancias que deberían usarse para predecir

Deploy the model to an endpoint

```
xgb_predictor = xgb.deploy(
    initial_instance_count=1,
    instance_type="ml.m4.xlarge",
    serializer=sagemaker.serializers.CSVSerializer(),
)
```

5. Predicciones

Evaluaremos los resultados obtenidos en las predicciones

Run the model on our test data

```
def predict(data, predictor):
    predictions = []
    confidences = []
    for row in data:
        response = np.fromstring(predictor.predict(row).decode("utf-8")[1:], sep=",")
        pred = response.argmax()
        confidence = max(response)
        predictions.extend([pred])
        confidences.extend([confidence])

    return predictions, confidences
```

```
print(
    classification_report(
        test_data["transaction_category"].to_list(), predictions, target_names=factorize_key
    )
)
```

```
                       precision    recall  f1-score   support

         Uncategorized       1.00      0.92      0.96        51
         Entertainment       0.81      0.89      0.85      1486
             Education       1.00      0.94      0.97        80
              Shopping       0.86      0.94      0.90      3441
         Personal Care       1.00      0.98      0.99       132
    Health and Fitness       0.99      0.89      0.94       443
       Food and Dining       0.99      0.82      0.90       918
   Gifts and Donations       1.00      0.95      0.97       275
           Investments       0.99      0.97      0.98        88
   Bills and Utilities       1.00      0.99      1.00       332
    Auto and Transport       0.94      0.84      0.88      1967
                Travel       0.96      0.84      0.90       120
      Fees and Charges       1.00      0.94      0.97       106
     Business Services       1.00      0.99      1.00       146
     Personal Services       1.00      0.96      0.98        75
                 Taxes       0.98      0.94      0.96        47
              Gambling       1.00      1.00      1.00        15
                  Home       0.98      0.89      0.93       168
Pension and insurances       0.99      1.00      1.00       110

              accuracy                           0.90     10000
             macro avg       0.97      0.93      0.95     10000
          weighted avg       0.91      0.90      0.90     10000
```

6. Terminar los recursos

Se sugiere que como buena práctica terminemos los recursos que hemos usado para reducir los costos

Remove the feature group and endpoint to clean up

```
feature_group.delete()
xgb_predictor.delete_endpoint(delete_endpoint_config=True)
```