

**GUIA PASO A PASO  
IMPLEMENTACIÓN DE MACHINE LEARNING  
CON AMAZON SAGEMAKER**

**GRUPO 4**

**ROBERTO CARLOS ISAJAR MORALES**

**CÓDIGO: 2171191**

**JUAN CAMILO ESPINOSA MORALES**

**CÓDIGO: 2175473**

**ALEJANDRO OSORIO ECHEVERRI**

**CÓDIGO: 2161629**

**LINO SANTIAGO ZAMORA MENDEZ**

**CÓDIGO: 2171252**

**DOCENTE:**

**OSCAR HERNAN MONDRAGON MARTINEZ**

**ASIGNATURA:**

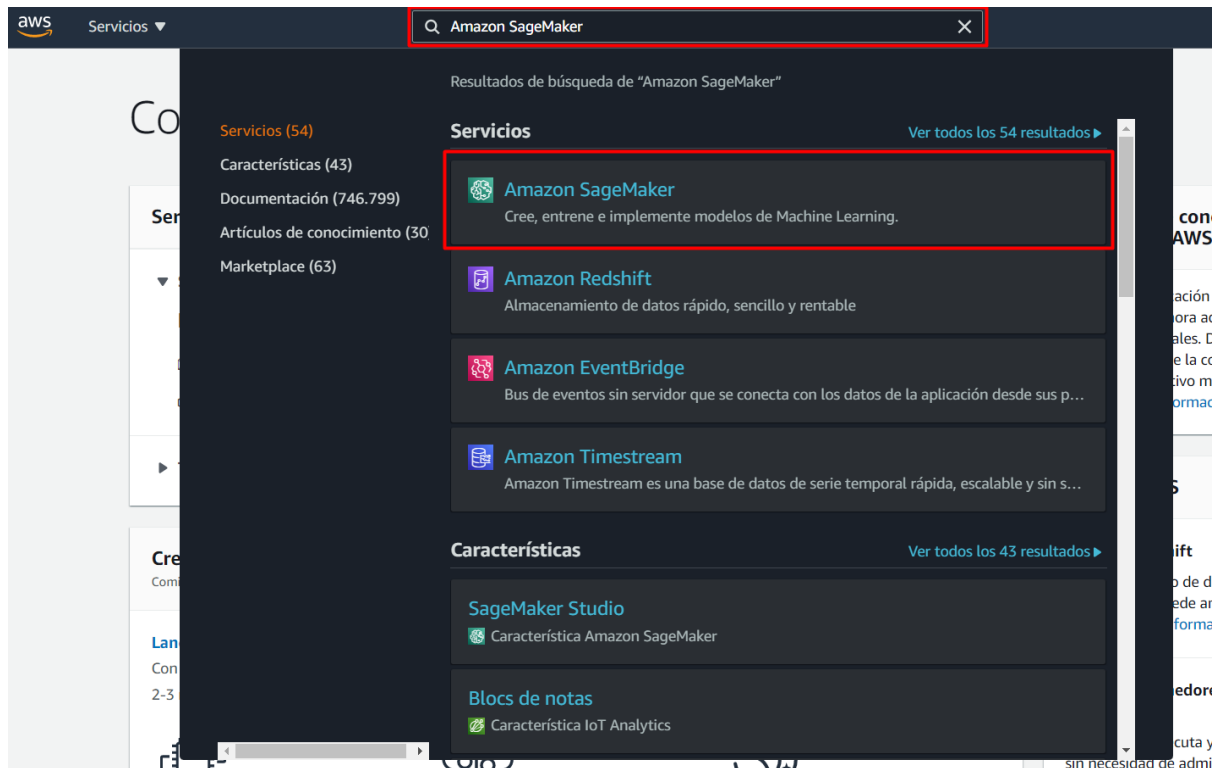
**COMPUTACIÓN EN LA NUBE**

**UNIVERSIDAD AUTÓNOMA DE OCCIDENTE  
FACULTAD DE INGENIERÍA  
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA  
SANTIAGO DE CALI, VALLE DEL CAUCA  
2021**

Para seguir esta guía paso a paso es necesario tener creada una cuenta en **AWS** [haga clic aquí](#).

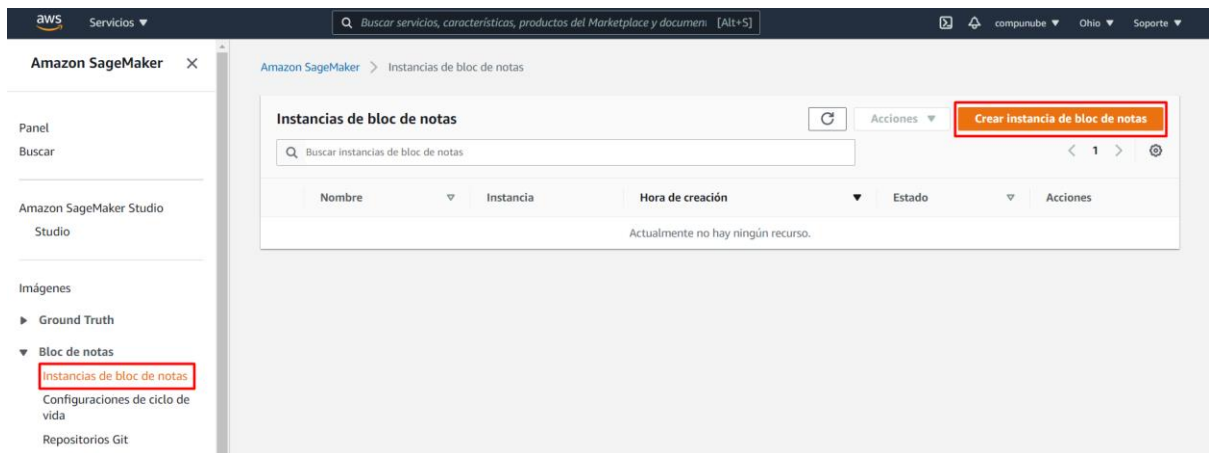
## Paso 1: Abra la consola de Amazon SageMaker

[Haga clic aquí](#), para abrir una consola de administración de AWS, estando ahí utilice la barra de búsqueda y escriba **Amazon SageMaker**, y posteriormente seleccionarlo para abrir el servicio.



## Paso 2: Cree una instancia de bloc de notas de Amazon SageMaker

Cree una instancia de **Instancia de bloc de notas**.



Durante la creación de la **Instancia de bloc de notas**, escriba el **Nombre de instancia del bloc de notas** este puede ser cualquier nombre, las demás opciones se pueden dejar por defecto. Se debe **Crear un nuevo rol** para Amazon SageMaker, para permitir que la instancia del bloc de notas acceda a Amazon S3 y pueda cargar datos de manera segura en este servicio.

Amazon SageMaker > Instancias de bloc de notas > Crear instancia de bloc de notas

## Crear instancia de bloc de notas

Amazon SageMaker proporciona instancias de bloc de notas completamente administradas e integradas que ejecutan blocs de notas Jupyter. Las instancias de bloc de notas incluyen código de ejemplo para ejercicios de alojamiento y entrenamiento de modelos comunes. [Obtenga más información](#)

### Configuración de la instancia del bloc de notas

Nombre de instancia del bloc de notas

Proyecto-Final-CN

Máximo de 63 caracteres alfanuméricos. Puede incluir guiones (-), pero no espacios. Debe ser único dentro de la cuenta en una región de AWS.

Tipo de instancia de bloc de notas

ml.t2.medium

Inferencia elástica [Obtenga más información](#)

ninguno

**i** La instancia de notebook de Amazon SageMaker está finalizando su soporte estándar en la AMI de Amazon Linux (AL1). [Obtenga más información](#)

Identificador de plataforma [Obtenga más información](#)

notebook-al1-v1

► Configuración adicional

Permiso a utilizar

Crear un nuevo rol

Especifique el ARN de una función de IAM personalizada

Usar rol existente

Elegir un rol de IAM

3. Elija una función o permítanos

Estando en **Crear una función de IAM**, se debe seleccionar **Cualquier bucket de S3**, para permitir que los usuarios tengan acceso a su instancia del bloc de notas, a cualquier bucket y a su contenido en su cuenta. Posteriormente seleccioné **Crear función**.

Crear una función de IAM

Transferir una función de IAM otorga permiso a Amazon SageMaker para realizar acciones en otros servicios de AWS en su nombre. La creación de una función aquí concederá los permisos descritos por la [AmazonSageMakerFullAccess](#) política de IAM a la función que cree.

La función de IAM que cree proporcionará acceso a:

☒ Los buckets de S3 que especifique - *opcional*

☒ Cualquier bucket de S3

Permita que los usuarios tengan acceso a su instancia del bloc de notas, a cualquier bucket y a su contenido en su cuenta.

☐ Buckets de S3 específicos

Ejemplo: nombre-bucket-1, nc

Delimitado por comas. No se admiten ARN, "\*" o "/"

☐ Ninguno

☒ Cualquier bucket de S3 con "sagemaker" en el nombre

☒ Cualquier objeto de S3 con "sagemaker" en el nombre

☒ Cualquier objeto de S3 con la etiqueta "sagemaker" y el valor "true" [Ver etiquetado de objetos](#)

☒ El bucket de S3 con una política de bucket que permite el acceso a SageMaker [Ver políticas de bucket de S3](#)

Cancelar



Crear función

Para esta guía paso a paso, se utilizarán las demás opciones como predeterminadas, seleccione **Crear instancia de bloc de notas**.

### Permisos y cifrado

**Rol de IAM**  
Las instancias de bloc de notas requieren permisos para llamar a otros servicios, incluidos SageMaker y S3. Elija una función o permítanos crear una función con la [AmazonSageMakerFullAccess](#) política de IAM asociada.

AmazonSageMaker-ExecutionRole-20211025T085403 ▼

 **Ha creado una función de IAM correctamente.** [AmazonSageMaker-ExecutionRole-20211025T085403](#) 

**Acceso raíz - opcional**  
☒ **Habilitar:** proporcionar a los usuarios acceso raíz al bloc de notas  
☐ **Deshabilitar:** no conceda a los usuarios acceso raíz al bloc de notas  
Las configuraciones de ciclo de vida siempre tienen acceso raíz

**Clave de cifrado - opcional**  
Cifre sus datos de bloc de notas. Elija una clave de KMS existente o escriba el ARN de una clave.

Sin cifrado personalizado ▼

► **Red - opcional**




► **Repositorios Git - opcional**

► **Etiquetas - opcional**

Cancelar **Crear instancia de bloc de notas**

Estando en el apartado **Instancias de bloc de notas**, se le mostrará su nueva instancia de bloc de notas con el estado **Pending**, esta debería pasar en unos minutos al estado de **InService**.

Amazon SageMaker


  **La instancia de bloc de notas se está creando correctamente.** [Ver detalles](#)   
Abra la instancia del bloc de notas cuando su estado sea En servicio y abra un bloc de notas de plantilla para empezar.


Panel  
Buscar

Amazon SageMaker Studio  
Studio

Imágenes

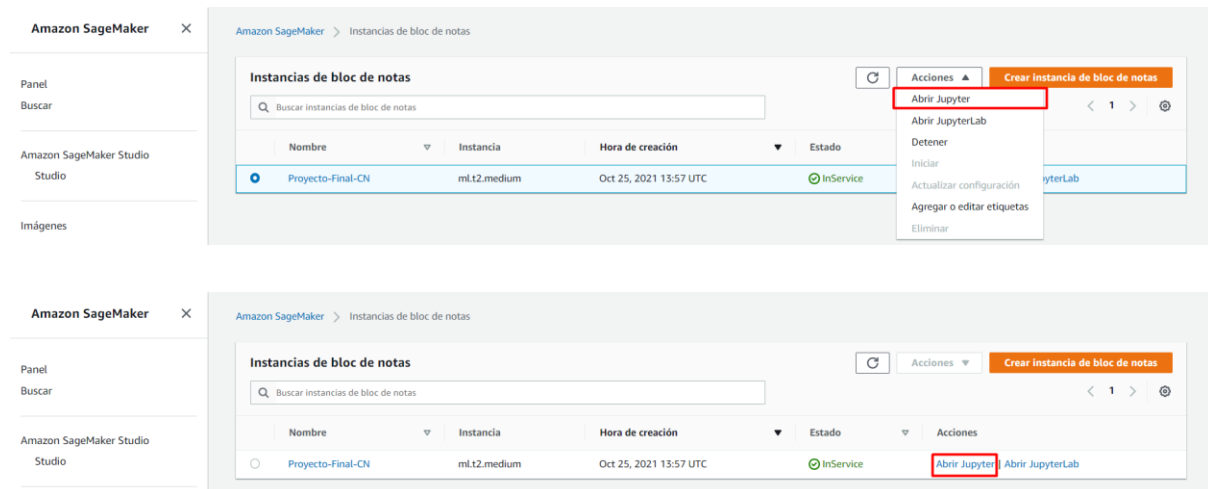
Amazon SageMaker > Instancias de bloc de notas

**Instancias de bloc de notas**  Acciones ▼ **Crear instancia de bloc de notas**

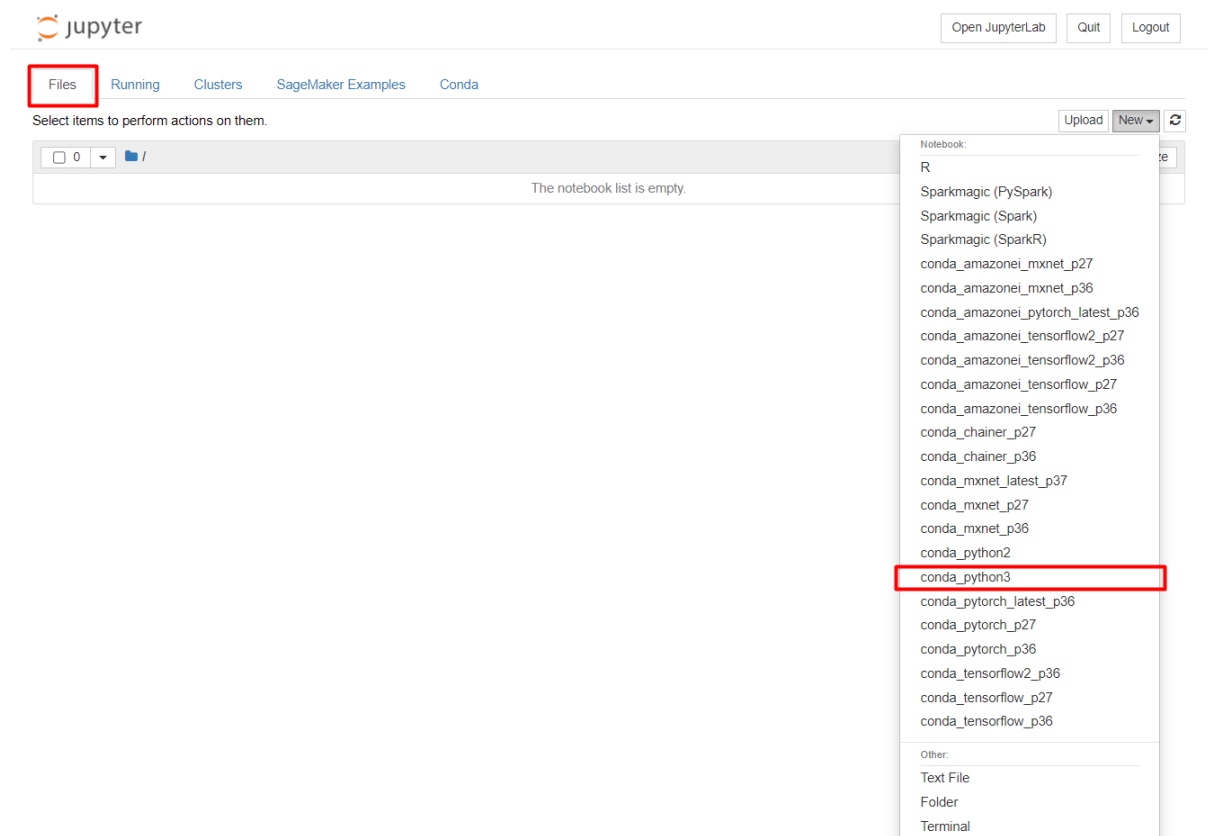
Nombre	Instancia	Hora de creación	Estado	Acciones
<input type="radio"/> Proyecto-Final-CN	ml.t2.medium	Oct 25, 2021 13:57 UTC	 Pending	-

### Paso 3: Prepare los datos

Cuando el estado de la instancia del bloc de notas este en **InService**, diríjase al menú desplegable **Acciones** y seleccione la opción **Abrir Jupyter** o en la columna **Acciones** que se encuentra al lado de **InService**, seleccione la opción **Abrir Jupyter**.



Estando en **Jupyter** en la pestaña **Files**, seleccione **New**, posteriormente **conda\_python3**.



Se debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**. Con esto se importaron algunas bibliotecas y definieron algunas

variables del entorno en su entorno de bloc de notas de **Jupyter**. Con esto se prepararon los datos, necesarios para entrenar el modelo de aprendizaje automático e implementarlo.

Durante la ejecución del código, aparecerá el símbolo \* entre corchetes, Luego de completarse la ejecución del código, el símbolo \* se reemplazará por el número 1.

```
# import libraries
import boto3, re, sys, math, json, os, sagemaker,
urllib.request
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer

# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
              'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'} # each region has its
XGBoost container
my_region = boto3.session.Session().region_name # set the
region of the instance
print("Success - the MySageMakerInstance is in the " +
my_region + " region. You will use the " +
containers[my_region] + " container for your SageMaker
endpoint.")
```



```
File Edit View Insert Cell Kernel Widgets Help Trusted | conda_python3
In [*]: # import libraries
import boto3, re, sys, math, json, os, sagemaker, urllib.request
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer

# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
              'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'} # each region has its XGBoost container

my_region = boto3.session.Session().region_name # set the region of the instance
print("Success - the MySageMakerInstance is in the " + my_region + " region. You will use the " + containers[my_region] + " container for your SageMaker endpoint.")
```

```
File Edit View Insert Cell Kernel Widgets Help Trusted | conda_python3
In [1]: # import libraries
import boto3, re, sys, math, json, os, sagemaker, urllib.request
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer

# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
              'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'} # each region has its XGBoost container

my_region = boto3.session.Session().region_name # set the region of the instance
print("Success - the MySageMakerInstance is in the " + my_region + " region. You will use the " + containers[my_region] + " container for your SageMaker endpoint.")

Success - the MySageMakerInstance is in the us-east-2 region. You will use the [REDACTED].dkr.ecr.us-east-2.amazonaws.com/xgboost:latest container for your SageMaker endpoint.
```

Se debe copiar el siguiente código en una celda de **Jupyter**, cambiando el nombre del bucket de S3 (Este debe de ser único), posteriormente seleccioné **Run**. De no recibir un mensaje de **successfully**, deberás cambiar el nombre de bucket de S3 y volver a intentarlo.

```
bucket_name = 'compunube' # <--- CAMBIE ESTA VARIABLE POR UN NOMBRE ÚNICO PARA SU BUCKET
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)

```

```

In [2]: bucket_name = 'compunube' # <--- CHANGE THIS VARIABLE TO A UNIQUE NAME FOR YOUR BUCKET
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

```

Se procede a descargar los datos en su instancia de **Amazon SageMaker** y cargarlos en un marco de datos. Se debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**.

```

try:
    urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-
train-deploy-machine-learning-model-
sagemaker/bank_clean.27f01fbbdf43271788427f3682996ae29ceca05d.csv"
, "bank_clean.csv")
    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)

```

```
In [3]: try:
urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-train-deploy-machine-learning-model-sagemaker/bank_clean.27f01:
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.

Se procederá a mezclar los datos y los dividiremos en datos de entrenamiento y de prueba.

Se debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**. Con esto se mezclarán y dividirán los datos.

```
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)
```

```
In [4]: 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)

#### Paso 4: Entrene el modelo con los datos

Se debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**. Con esto se cambiará el formato y se cargarán los datos.

```
pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'],
axis=1)], axis=1).to_csv('train.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.
join(prefix, 'train/train.csv')).upload_file('train.csv')
s3_input_train =
sagemaker.inputs.TrainingInput(s3_data='s3://{}/{}/train'.format(b
ucket_name, prefix), content_type='csv')
```

```
In [5]: pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'], axis=1)], axis=1).to_csv('train.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'train/train.csv')).upload_file('train.csv')
s3_input_train = sagemaker.inputs.TrainingInput(s3_data='s3://{}/{}/train'.format(bucket_name, prefix), content_type='csv')
```

Se procede a configurar la sesión de **Amazon SageMaker**, crear una instancia del modelo XGBoost (un estimador) y definir los hiperparámetros del modelo. Se debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**.

```
sess = sagemaker.Session()
xgb = sagemaker.estimator.Estimator(containers[my_region],role,
train_instance_count=1,
train_instance_type='ml.m4.xlarge',output_path='s3://{}/{}/output'
.format(bucket_name, prefix),sagemaker_session=sess)
xgb.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,subsample=0.8,silent=0,objective='binary:logistic',num_round=100)
```

```
In [6]: sess = sagemaker.Session()
xgb = sagemaker.estimator.Estimator(containers[my_region],role, train_instance_count=1, train_instance_type='ml.m4.xlarge',output
xgb.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,subsample=0.8,silent=0,objective='binary:logistic',num_rou

train_instance_count has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
train_instance_type has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

Ya con los datos cargados y el estimador XGBoost configurado, entrene el modelo. Se debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**.

Pasado algunos minutos, se debería ver los registros de entrenamiento que se han generado.

```
xgb.fit({'train': s3_input_train})
```

```
In [7]: xgb.fit({'train': s3_input_train})
[92]#011train-error:0.094239
[14:38:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 16 extra nodes, 32 pruned nodes, max_depth=5
[93]#011train-error:0.094169
[14:38:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14 extra nodes, 28 pruned nodes, max_depth=5
[94]#011train-error:0.094169
[14:38:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10 extra nodes, 14 pruned nodes, max_depth=5
[95]#011train-error:0.094204
[14:38:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 28 pruned nodes, max_depth=0
[96]#011train-error:0.094204
[14:38:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 26 extra nodes, 20 pruned nodes, max_depth=5
[97]#011train-error:0.093927
[14:38:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 38 pruned nodes, max_depth=0
[98]#011train-error:0.093927
[14:38:36] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 32 pruned nodes, max_depth=0
[99]#011train-error:0.093892

2021-10-25 14:38:50 Completed - Training job completed
Training seconds: 53
Billable seconds: 53
```

## Paso 5: Implemente el modelo

Para la implementación de modelo en un servidor y crear un punto de enlace al que pueda acceder, se debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**.

```
xgb_predictor =  
xgb.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')
```

```
In [8]: xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')  
-----!
```

Para predecir si los clientes de los datos de prueba se inscribieron o no en el producto del banco, se debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**.

```
test_data_array = test_data.drop(['y_no', 'y_yes'], axis=1).values  
#Load the data into an array  
xgb_predictor.serializer = csv_serializer # set the serializer  
type  
predictions = xgb_predictor.predict(test_data_array).decode('utf-  
8') # predict!  
predictions_array = np.fromstring(predictions[1:], sep=',') # and  
turn the prediction into an array  
print(predictions_array.shape)
```

```
In [9]: test_data_array = test_data.drop(['y_no', 'y_yes'], axis=1).values #Load the data into an array  
xgb_predictor.serializer = csv_serializer # set the serializer type  
predictions = xgb_predictor.predict(test_data_array).decode('utf-8') # predict!  
predictions_array = np.fromstring(predictions[1:], sep=',') # and turn the prediction into an array  
print(predictions_array.shape)
```

The csv\_serializer has been renamed in sagemaker>=2.  
See: <https://sagemaker.readthedocs.io/en/stable/v2.html> for details.

```
(12357,)
```

## Paso 6. Evalúe el rendimiento del modelo

Se debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**. Con esto se comparan los valores reales con los valores predichos en una tabla denominada **matriz de confusión**.

Con base en los pronósticos, es factible anticipar que un cliente se inscribirá en un certificado de depósito para exactamente el 90% de los consumidores en los datos de prueba, con una precisión del 63% (278/429) para los registrados y del 90%. (10 785/11928) para aquellos que no están registrados.

```

cm = pd.crosstab(index=test_data['y_yes'],
columns=np.round(predictions_array), rownames=['Observed'],
colnames=['Predicted'])
tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp =
cm.iloc[0,1]; p = (tp+tn)/(tp+tn+fp+fn)*100
print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate:
", p))
print("{0:<15}{1:<15}{2:>8}".format("Predicted", "No Purchase",
"Purchase"))
print("Observed")
print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("No
Purchase", tn/(tn+fn)*100,tn, fp/(tp+fp)*100, fp))
print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<})
\n".format("Purchase", fn/(tn+fn)*100,fn, tp/(tp+fp)*100, tp))

```

```

In [10]: cm = pd.crosstab(index=test_data['y_yes'], columns=np.round(predictions_array), rownames=['Observed'], colnames=['Predicted'])
tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp = cm.iloc[0,1]; p = (tp+tn)/(tp+tn+fp+fn)*100
print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate: ", p))
print("{0:<15}{1:<15}{2:>8}".format("Predicted", "No Purchase", "Purchase"))
print("Observed")
print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("No Purchase", tn/(tn+fn)*100,tn, fp/(tp+fp)*100, fp))
print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Purchase", fn/(tn+fn)*100,fn, tp/(tp+fp)*100, tp))

```

Overall Classification Rate: 89.5%

Predicted	No Purchase	Purchase
Observed		
No Purchase	90% (10769)	37% (167)
Purchase	10% (1133)	63% (288)

## Paso 7: Termine los recursos

Se elimina el punto de enlace de Amazon SageMaker y los objetos de su bucket de S3, para esto debe copiar el siguiente código en una celda de **Jupyter**, posteriormente seleccione **Run**.

```

sagemaker.Session().delete_endpoint(xgb_predictor.endpoint)
bucket_to_delete = boto3.resource('s3').Bucket(bucket_name)
bucket_to_delete.objects.all().delete()

```

## Referencia

- [1] Amazon Web Services. "Cómo crear, entrenar e implementar un modelo de aprendizaje automático con Amazon SageMaker | AWS". Amazon Web Services, Inc. <https://aws.amazon.com/es/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/> (accedido el 29 de octubre de 2021).
- [2] Amazon Web Services. "What Is Amazon SageMaker? - Amazon SageMaker". Amazon Web Services, Inc. <https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html> (accedido el 29 de octubre de 2021).
- [3] Amazon Web Services. "What Is Amazon SageMaker? - Amazon SageMaker". Readthedocs. <https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html> (accedido el 29 de octubre de 2021).
- [4] P. P. Nayak. "AWS sagemaker : A brief introduction". Medium. <https://medium.com/@partha.pratimnayak/aws-sagemaker-a-brief-introduction-6647348f387f> (accedido el 29 de octubre de 2021).