

# IBM Cloud

## Hands-on Workshop

IBM Developer Advocacy

### Módulo: Data Science – Jupyter Notebook con Machine Learning y App.

---

#### Agenda

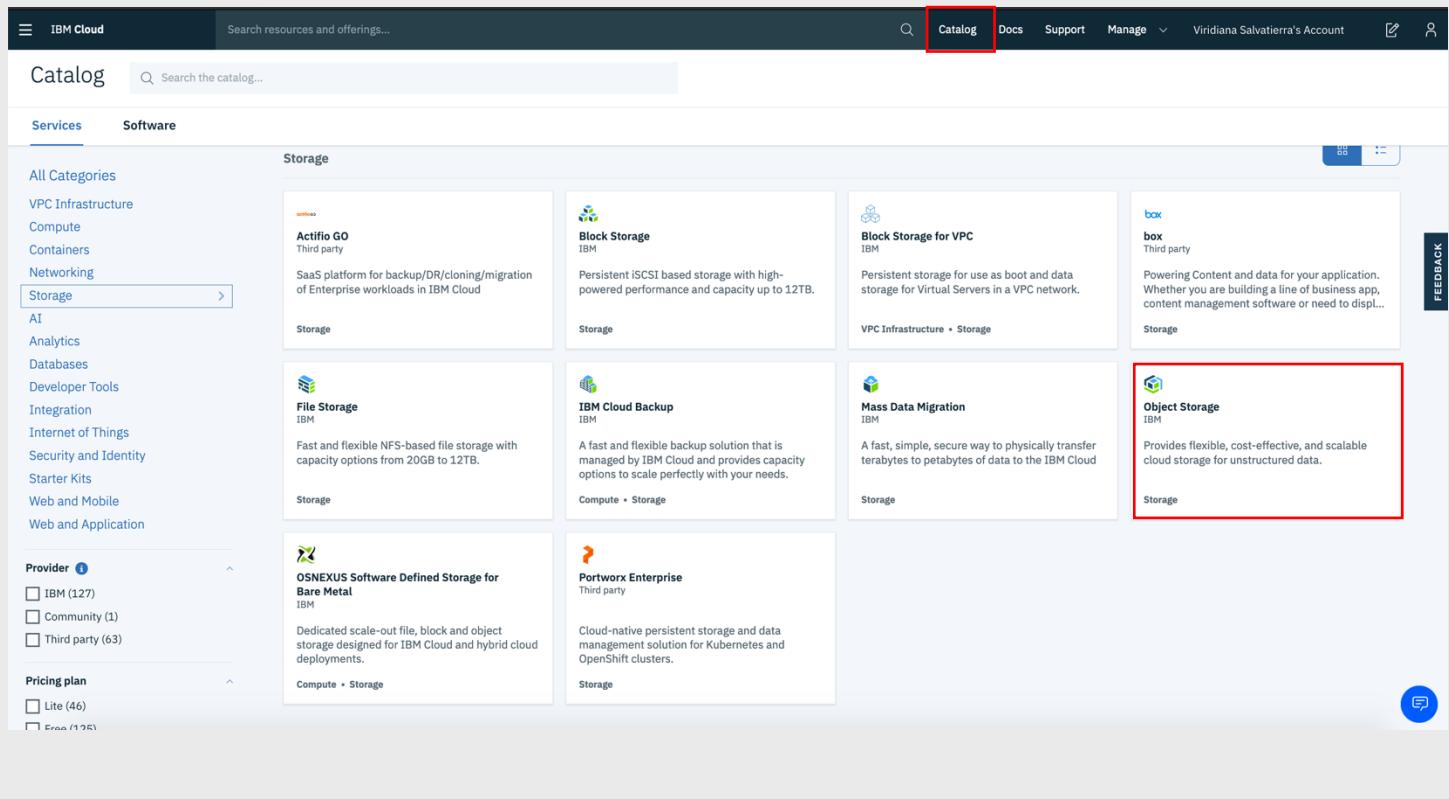
0.	Prework.....	p. 2
1.	Crear una instancia de Cloud Object Storage.....	p. 2
2.	Crear una instancia de Watson Studio.....	p. 3
3.	Crear un proyecto .....	p. 6
4.	Cargar un <i>Data asset</i> .....	p. 7
5.	Importar una Jupyter Notebook .....	p. 8
6.	Cargar un data asset a nuestra Notebook .....	p. 10
7.	Crear un modelo.....	p. 12
8.	Guardar el modelo y probar los datos .....	p. 13
9.	Desplegar el modelo .....	p. 16
10.	Probar el modelo .....	p. 18
11.	Desplegar la app .....	p. 23
12.	Siguientes pasos .....	p. 25

# 0. Prework:

- Tener instalado [github Desktop](#) o [git cli](#).
- Tener instalado el [CLI de IBM Cloud](#) (Si tienes problemas con este paso, da clic [aquí para descargar la versión “STANDALONE”](#)).
- Descargar el repositorio del siguiente enlace:  
<https://github.com/ibmdevelopermx/CHURN-app-Jupyter-Notebook-y-WS>.

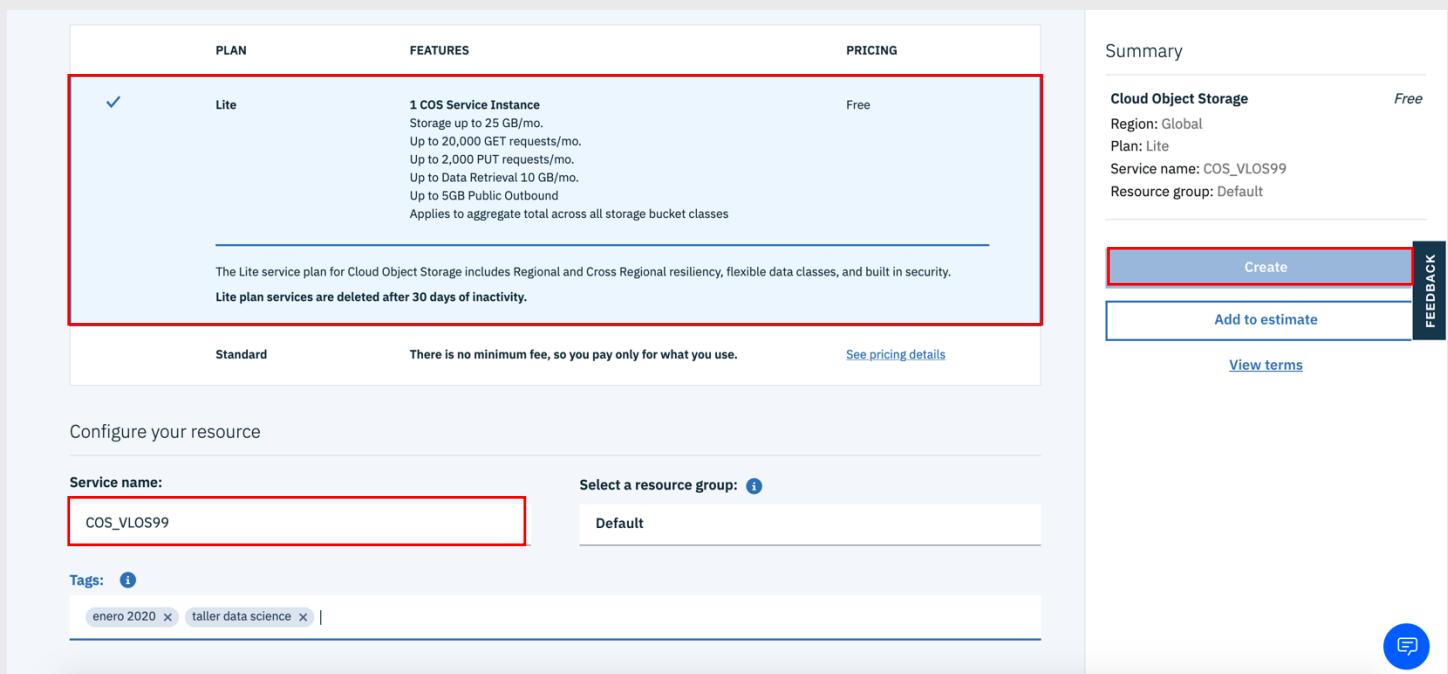
# 1. Crear una instancia de Cloud Object Storage.

1. Realizamos login a IBM Cloud  
<http://cloud.ibm.com>
2. Seleccionamos “Catalog” (Catálogo) en el menú superior.
3. En las categorías de la izquierda, seleccionamos “Storage”.
4. Seleccionamos el servicio de “Object Storage”.



The screenshot shows the IBM Cloud Catalog interface. The top navigation bar includes the IBM Cloud logo, a search bar, and a menu with 'Catalog' highlighted in red. The left sidebar lists categories like All Categories, VPC Infrastructure, Compute, Containers, Networking, Storage (selected), AI, Analytics, Databases, Developer Tools, Integration, Internet of Things, Security and Identity, Starter Kits, Web and Mobile, and Web and Application. Below this is a 'Provider' filter with options for IBM (127), Community (1), and Third party (63). A 'Pricing plan' filter shows Lite (46) and Free (125). The main content area is titled 'Storage' and displays several service cards. One card for 'Object Storage' by IBM is highlighted with a red box. Other visible cards include Actifio GO, Block Storage, Block Storage for VPC, box, File Storage, IBM Cloud Backup, Mass Data Migration, and Portworx Enterprise.

5. Seleccionamos el plan “**Lite**”, le damos un nombre a nuestra instancia de servicio y damos clic en “**Create**”.



PLAN	FEATURES	PRICING
<input checked="" type="checkbox"/> Lite	<b>1 COS Service Instance</b> Storage up to 25 GB/mo. Up to 20,000 GET requests/mo. Up to 2,000 PUT requests/mo. Up to Data Retrieval 10 GB/mo. Up to 5GB Public Outbound Applies to aggregate total across all storage bucket classes	Free
The Lite service plan for Cloud Object Storage includes Regional and Cross Regional resiliency, flexible data classes, and built in security. <b>Lite plan services are deleted after 30 days of inactivity.</b>		
Standard	There is no minimum fee, so you pay only for what you use.	<a href="#">See pricing details</a>

Configure your resource

Service name:

Select a resource group:

Tags:

[Create](#) [Add to estimate](#) [View terms](#) [FEEDBACK](#)

## 2. Crear una instancia de Watson Studio:

1. Realizamos login a IBM Cloud  
<http://cloud.ibm.com>
2. Seleccionamos “**Catalog**” (Catálogo) en el menú superior.
3. En las categorías de la izquierda, seleccionamos “**AI**”.
4. Seleccionamos el servicio de “**Watson Studio**”

IBM Cloud Catalog

Search resources and offerings...

Catalog Docs Support Manage IBM

All Categories Services Software Private

**Services**

VPC Infrastructure Compute Containers Networking Storage

**AI**

Watson Assistant IBM Watson Studio IBM Annotator for Clinical Data IBM

Watson Assistant lets you build conversational interfaces into any application, device, or channel.

Embed AI and machine learning into your business. Create custom models using your own data.

Analyze text to extract medical codes and concepts such as diseases, lab values, medications, procedures and more.

Watson Studio IBM Discovery Insights for Medical Literature IBM Language Translator

Compare and Comply IBM Process governing documents to convert, identify, classify, and compare important elements.

Add a cognitive search and content analytics engine to applications.

Discovery IBM Insights for Medical Literature IBM Language Translator

Internet of Things Security and Identity Starter Kits Web and Mobile Web and Application

Provider IBM Community Third party

Feedback

5. Una vez dentro del servicio, seleccionamos la región “Dallas”.

6. Seleccionamos el plan “Lite”.

IBM Cloud Watson Studio

Search resources and offerings...

Catalog Docs Support Manage IBM

Lite IBM Service IAM-enabled

Author: IBM • Date of last update: 07/18/2019

Need Help? Contact Support View docs

Create About

Select a region

Dallas

Select a pricing plan  
Displayed prices do not include tax. Monthly prices shown are for country or region: [United States](#)

PLAN	FEATURES	PRICING
Lite	<b>1 authorized user</b> 50 capacity unit-hours monthly limit 1 free small compute environment with 1 vCPU and 4 GB RAM (does not require capacity unit-hours)	Free

The Lite plan for Watson Studio offers everything you need to become a better data scientist or domain expert in a collaborative environment.  
Lite plan services are deleted after 30 days of inactivity.

Standard v1 1 authorized user + unlimited viewer collaborators \$99.00 USD/Instance

Summary

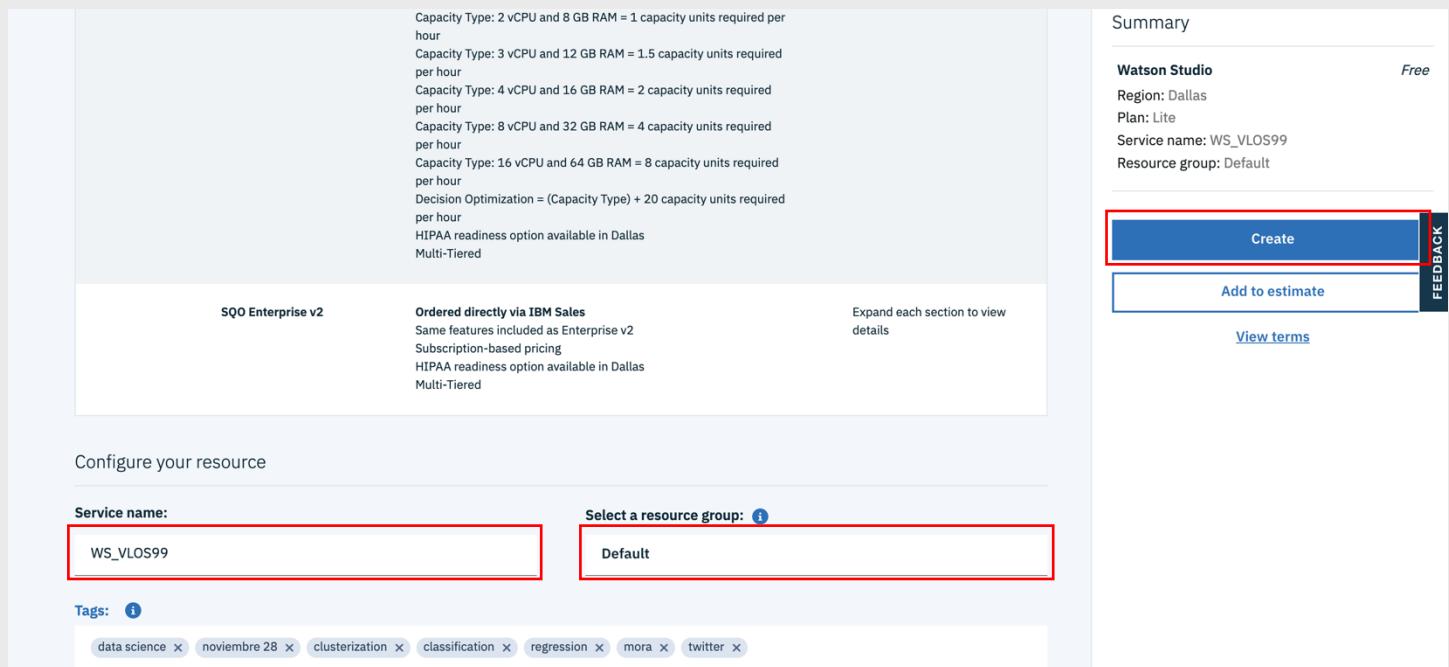
**Watson Studio** Free

Region: Dallas  
Plan: Lite  
Service name: Watson Studio-sx  
Resource group: Default

Create Add to estimate View terms

Feedback

7. En la parte inferior de la página, escribir el “Nombre de la instancia del Servicio”. (Es recomendable agregar etiquetas o “tags”)
8. En grupo de recursos, dejamos el valor “Default”.
9. Dar clic en “Create”.



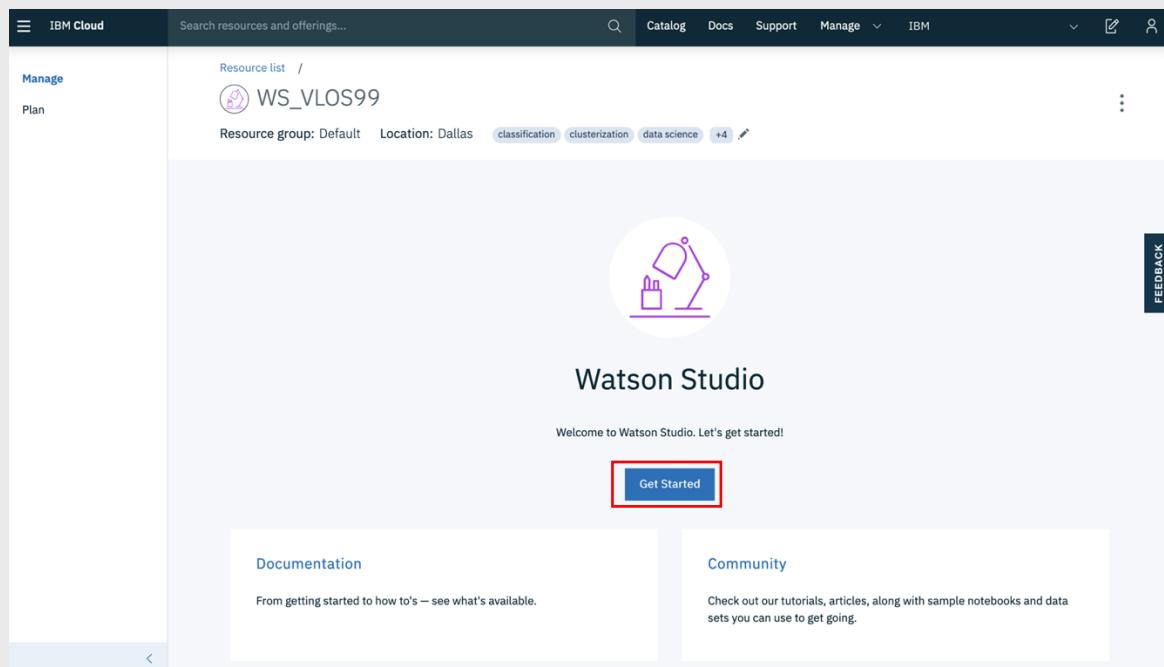
Capacity Type: 2 vCPU and 8 GB RAM = 1 capacity units required per hour  
 Capacity Type: 3 vCPU and 12 GB RAM = 1.5 capacity units required per hour  
 Capacity Type: 4 vCPU and 16 GB RAM = 2 capacity units required per hour  
 Capacity Type: 8 vCPU and 32 GB RAM = 4 capacity units required per hour  
 Capacity Type: 16 vCPU and 64 GB RAM = 8 capacity units required per hour  
 Decision Optimization = (Capacity Type) + 20 capacity units required per hour  
 HIPAA readiness option available in Dallas  
 Multi-Tiered

SQO Enterprise v2      Ordered directly via IBM Sales      Expand each section to view details

Service name: WS\_VLOS99      Select a resource group: Default

Tags: data science, noviembre 28, clusterization, classification, regression, mora, twitter

10. En la pestaña que se despliegue, damos clic en “Get Started” de la pestaña “Manage”.



Resource list / WS\_VLOS99

Resource group: Default Location: Dallas classification clusterization data science +4

Watson Studio

Welcome to Watson Studio. Let's get started!

Get Started

Documentation      Community

From getting started to how to's – see what's available.

Check out our tutorials, articles, along with sample notebooks and data sets you can use to get going.

### 3. Crear un proyecto:

- Una vez en la página principal de nuestra instancia, damos clic en “Create a project”.

Welcome Viridiana!

Watson Studio • Try out other IBM Watson Studio apps

**Start by creating a project**

A project is how you organize your resources to work with data and collaborate with team members

**Create a project**  
Create a project, then add the tools and assets you need.

- Seleccionamos la opción “Create an empty project”.

[← Back](#)

## Create a project

Choose whether to create an empty project or to preload your project with data and analytical assets. Add collaborators and data, and then choose the right tools to accomplish your goals. Add services as necessary.

**Create an empty project**

Add the data you want to prepare, analyze, or model. Choose tools based on how you want to work: write code, create a flow on a graphical canvas, or automatically build models.

**NEW** AutoAI experiment tool: Fully automated approach to building a classification or re...

**USE TO**

- Prepare and visualize data
- Analyze data in notebooks
- Train models

**Create a project from a sample or file**

Get started fast by loading existing assets. Choose a project file from your system, or choose a curated sample project.

**USE TO**

- Learn by example
- Build on existing work
- Run tutorials

- Ahora, le damos un nombre al proyecto, comprobamos que nuestra instancia de “Cloud Object Storage” sea correcta y damos clic en “Create”.

## New project

## Define project details

## Name

JN\_VLOS99

## Description

Manual Data Science.  
Uso de Jupyter Notebook con Machine Learning con App para predecir el CHURN de una TelCo.

## Choose project options

 Restrict who can be a collaborator i

Project will include integration with Cloud Object Storage for storing project assets.

## Storage

COS\_VLOS99

Cancel

**Create**

## 4. Cargar un *data asset*:

1. Una vez dentro del proyecto que acabamos de crear, damos clic en “**Assets**”. Del lado derecho, en el apartado de “**Find and add data**”, seleccionamos “**Browse**” o arrastramos y soltamos el archivo “**Telco-Customer-Churn.csv**” de la carpeta “**Data**” del repositorio que recién clonamos.

The screenshot shows the IBM Watson Studio interface with the 'Assets' tab selected. On the right, there's a 'Find and add data' section with a 'Browse' button and a file upload area. A red box highlights the 'Assets' tab and the 'Load' button. Another red box highlights the 'Drop files here or browse for files to upload' area. Below, a table lists data assets, and a red box highlights the list of assets.

NAME	TYPE	CREATED BY	LAST MODIFIED	ACTIONS
You don't have any Data assets yet.				

**Data assets**

NAME	TYPE	CREATED BY	LAST MODIFIED	ACTIONS
CSV Telco-Customer-Churn.csv	Data Asset	Viridiana Salvatierra	20 Jan 2020, 9:26:06 am	<input type="checkbox"/>

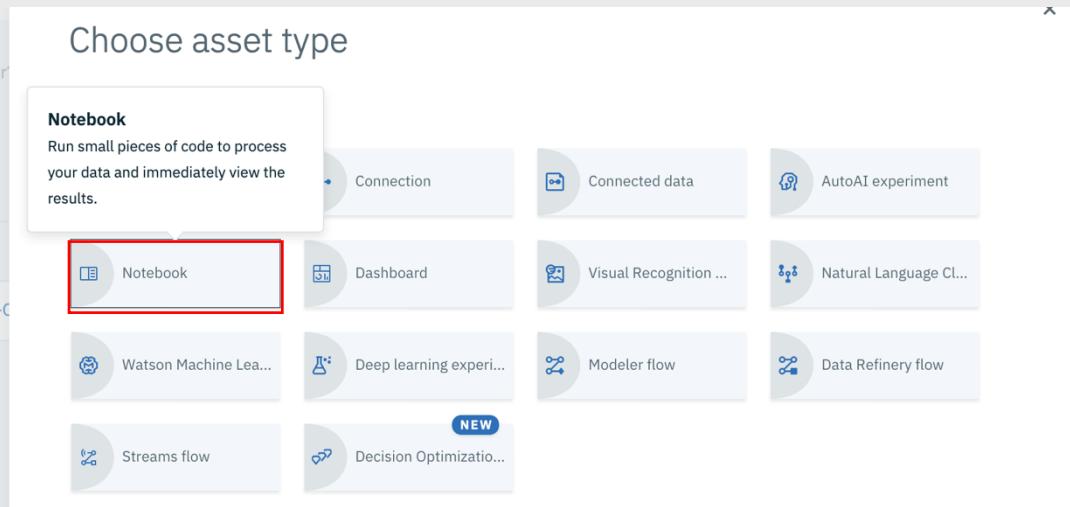
Stay on the page until upload completes.  
Incomplete uploads are cancelled

# 5. Importar una Jupyter Notebook:

1. Damos clic en “Add to project”.

The screenshot shows the IBM Watson Studio interface. At the top, there's a navigation bar with 'IBM Watson Studio', 'Upgrade', 'Bell', 'Viridiana Salvatierra's Account', and a 'VS' icon. Below the bar, the 'My Projects' section shows 'JN\_VLOS99'. The main menu includes 'Overview', 'Assets' (which is selected), 'Environments', 'Jobs', 'Deployments', 'Access Control', and 'Settings'. A toolbar below the menu has icons for 'Load', 'Files', and 'Catalog'. A prominent red box highlights the 'Add to project' button, which is located in the top right corner of the main workspace area.

2. Elegimos “Notebook” como “Asset Type”.



3. Ahora seleccionamos la pestaña “From URL”. Le damos un nombre original a nuestra Jupyter Notebook, elegimos la opción “Python 3.6” en “Enviroment” y en URL pegamos la siguiente dirección: <https://raw.githubusercontent.com/ibmdevelopermx/CHURN-app-Jupyter-Notebook-y-WS/master/notebooks/Telco-customer-churn-ICP4D.ipynb>

Y damos clic en “Create Notebook”.

This is a 'New notebook' creation dialog. At the top, there are three tabs: 'Blank', 'From file', and 'From URL', with 'From URL' selected and highlighted with a red box. The left panel contains fields for 'Name' (set to 'Telco-customer-CHURN-VLOS99') and 'Description (optional)' (containing a note about predicting churn in a Telco customer dataset). The right panel contains a 'Select runtime' dropdown set to 'Default Python 3.6 XS (2 vCPU and 8 GB RAM)', a note about capacity unit hours, and a 'Notebook URL' field containing the URL from step 3. At the bottom right, there are 'Cancel' and 'Create Notebook' buttons, with 'Create Notebook' highlighted with a red box.

4. Una vez que cargue nuestra Jupyter Notebook, podemos empezar a correr las celdas.

```
In [1]: !pip install --user watson-machine-learning-client --upgrade | tail -n 1
!pip install --user pyspark==2.3.3 --upgrade|tail -n 1

Requirement already satisfied, skipping upgrade: docutils>=0.10 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from ibm-cos-sdk-core==2.*,>=2.0.0->ibm-cos-sdk->watson-machine-learning-client) (0.14)
Requirement already satisfied, skipping upgrade: py4j==0.10.7 in /home/dsxuser/.local/lib/python3.6/site-packages (from pyspark==2.3.3) (0.10.7)
```

5. Tras correr la primera, regresamos a nuestro proyecto. Seleccionamos la pestaña “Assets” y en el apartado “Notebooks”, buscamos nuestro notebook. Damos clic en el candado “Unlock notebook” y en la ventana emergente, damos clic en “Unlock”.

NAME	SHARED	SCHEDULED	STATUS	LANGUAGE	LAST EDITOR	LAST MODIFIED	ACTIONS
Telco-Customer-Churn.csv				Data Asset	Viridiana Salvatierra	20 Jan 2020, 9:26:07 am	

NAME	SHARED	SCHEDULED	STATUS	LANGUAGE	LAST EDITOR	LAST MODIFIED	ACTIONS
Telco-customer-CHURN-VLOS99					Viridiana Salvatierra	20 Jan 2020	

6. Ahora damos clic en el menú de “hamburguesa” (aparece al pasar el puntero sobre él). Y seleccionamos la opción “**Stop Kernel**”. En la ventana emergente, damos clic en “**Stop**”. (Esto es para no gastar los recursos de memoria asignados a esta NOTEBOOK).

The screenshot shows the IBM Cloud Notebooks interface. A table lists notebooks, with 'Telco-customer-CHURN-VLOS99' selected. A context menu is open, and the 'Stop Kernel' option is highlighted with a red box. Below, a confirmation dialog asks if you want to stop the kernel for the selected notebook, with 'Stop' highlighted.

## 6. Cargar un *data asset* a nuestra Notebook:

1. Damos clic en nuestra “Notebook”. Una vez dentro de ésta, damos clic en el lápiz para editar.

The screenshot shows the IBM Watson Studio interface with a notebook titled 'Telco Customer Churn para ICP4D'. The top toolbar includes a pencil icon, which is highlighted with a red box, indicating the edit mode.

2. Damos clic en la celda del paso 2. Ahora seleccionamos el ícono “01/00” “Find and add data” del lado superior derecho de nuestra Notebook. Da clic en la pestaña “Files”, damos clic en “Insert to code” y seleccionamos la opción “Insert pandas DataFrame”.

The screenshot shows a Jupyter Notebook interface within the IBM Cloud environment. The top navigation bar includes icons for file operations (upload, download, share, etc.), a help icon, a refresh icon, a message icon, and a 'Files' icon (which is highlighted with a red box). Below the bar, the notebook title is 'Trusted | Python 3.6'. A sidebar on the right is titled 'Connections' and contains a section for uploading files with a 'Drop files here or browse' area. The main workspace displays a portion of a CSV file named 'Telco-Customer-Churn.csv'. In the code cell, the command `df\_data\_2 = pd.read\_csv(body)` is shown. To the right of the code cell, a dropdown menu is open, with the 'Insert to code' option and its sub-option 'Insert pandas DataFrame' both highlighted with red boxes.

3. Hecho esto, corremos la segunda celda. Nos daremos cuenta de que se despliega una tabla según los parámetros de la celda.

The screenshot shows the execution results of the second code cell. The code `df\_data\_2 = pd.read\_csv(body)` was run, followed by `df\_data\_2.head()`. The output displays the first five rows of a Pandas DataFrame. The columns are labeled: customerID, gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService, MultipleLines, InternetService, OnlineSecurity, ... DeviceProtection. The data shows five rows of customer information, including gender (Female/Male), senior citizen status, partner status, dependents count, tenure in months, service types, and internet security/Device Protection options. At the bottom, it indicates "5 rows x 21 columns".

4. Antes de correr la tercera celda, debemos asegurarnos de que la variable “n” en “df\_data\_n” sea igual en la segunda y tercera celdas. Ejemplo: en mi caso, el resultado

del proceso de la celda 2, dio como resultado “df\_data\_2”, por lo que en la tercera celda, “df” debe tener asignado el valor “df\_data\_2”.

```
df_data_2 = pd.read_csv(body)
df_data_2.head()
```

Out[1]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	T
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDÉ	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows × 21 columns

Usaremos la convención de nombramiento de Pandas 'df' para nuestro marco de datos (DataFrame). Asegúrate de que la celda inferior use el mismo nombre para el marco de datos(DataFrame) usado en la celda superior. Para el archivo cargado de forma local, debe verse como: 'df\_data\_1' o 'df\_data\_2' o 'df\_data\_x'. Para el caso de los datos virtualizados, debe verse como: data\_df\_1 o data\_df\_2 o data\_df\_x.

```
In [3]: # Para datos virtualizados:
# df = data_df_1

# Para carga local:
df = df_data_2
```

5. A continuación, seguimos ejecutando las celdas hasta llegar al apartado 3.0 “Crear un modelo”.

## 7. Crear un modelo:

1. En esta sección, se realiza la separación de datos de prueba y de entrenamiento.

### 3.0 Crear un modelo

```
In [ ]: from pyspark.sql import SparkSession
import pandas as pd
import json

spark = SparkSession.builder.getOrCreate()
df_data = spark.createDataFrame(df)
df_data.show()
```

#### 3.1 Dividir los datos en conjuntos de entrenamiento y de prueba.

```
In [ ]: spark_df = df_data
(train_data, test_data) = spark_df.randomSplit([0.8, 0.2], 24)

print("Número de registros para entrenamiento: " + str(train_data.count()))
print("Número de registros para evaluación: " + str(test_data.count()))
```

2. En el apartado 3.5, se crea la “pipeline” y el modelo utilizando el algoritmo clasificador “Random Forest”.

### 3.5 Crear una 'pipeline', y ajustar un modelo usando 'RandomForestClassifier'.

Montar todas las etapas a una 'pipeline'. No esperamos una regresión lineal limpia, así que usaremos 'RandomForestClassifier' para encontrar el mejor árbol de decisión para nuestros datos.

```
In [44]: classifier = RandomForestClassifier(featuresCol="features")
pipeline = Pipeline(stages=[si_gender, si_Partner, si_Dependents, si_PhoneService, si_MultipleLines, si_InternetService,
                           si_TechSupport, si_StreamingTV, si_StreamingMovies, si_Contract, si_PaperlessBilling,
                           classifier, label_converter])
model = pipeline.fit(train_data)

In [45]: predictions = model.transform(test_data)
evaluatorDT = BinaryClassificationEvaluator(rawPredictionCol="prediction")
area_under_curve = evaluatorDT.evaluate(predictions)

#La evaluación predeterminada es 'areaUnderROC'
print("areaUnderROC = %g" % area_under_curve)

areaUnderROC = 0.694581
```

## 8. Guardar el modelo y probar los datos:

1. Lo primero que debemos hacer, es darle un nombre fácilmente identificable a nuestro modelo. Podemos usar nuestras iniciales y un número o la fecha.

Agrega un nombre único para el MODEL\_NAME.

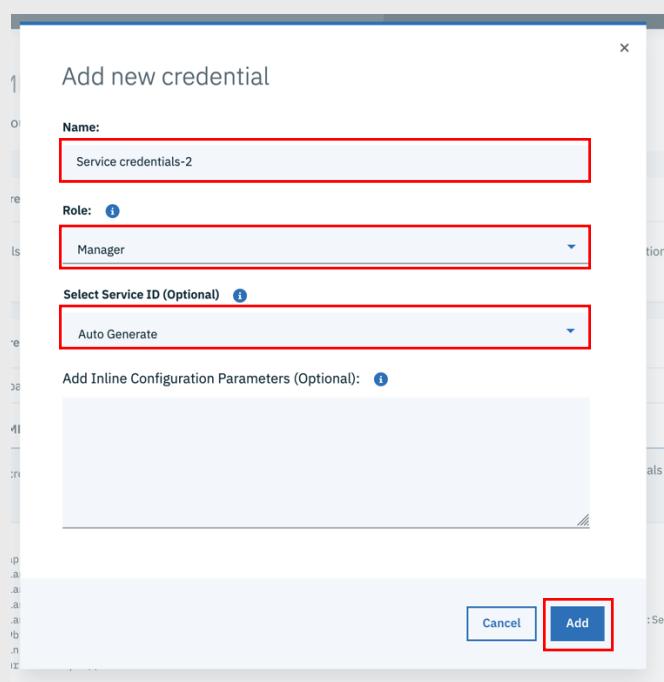
```
In [47]: MODEL_NAME = "JN_VLOS_99-Enero-21-2020"
```

2. En las credenciales, cambiamos el “url”, la “apikey” y la “instance\_id” por tus propias credenciales, para encontrarlas, en la página del servicio de “Watson Machine Learning”, damos clic en el apartado de “Service Credentials” y desplegamos la que tengamos generada o creamos una nueva credencial.

The screenshot shows the IBM Cloud Service Credentials page for the WML\_VLOS99 service. The left sidebar has 'Service credentials' highlighted. The main area shows a table of existing service credentials. A blue callout bubble points to the 'View credentials' button for an existing credential, with the text 'Visualizar credencial existente'. Another blue callout bubble points to the 'New credential +' button, with the text 'Crear una nueva credencial'. The table includes columns for KEY NAME, DATE CREATED, and actions. Below the table is a JSON snippet of a credential entry.

KEY NAME	DATE CREATED	
Service credentials-1	JAN 28, 2020 - 11:59:35 AM	<a href="#">View credentials</a>

```
{
  "apikey": "REDACTED",
  "iam_apikey_description": "Auto-generated for key",
  "iam_apikey_name": "Service credentials-1",
  "iam_role_crn": "cn:v1:bluemix:public:iam::::serviceRole:Manager",
  "iam_serviceid_crn": "cn:v1:bluemix:public:iam:identity:serviceId:REDACTED",
  "instance_id": "REDACTED",
  "url": "https://us-south.ml.cloud.ibm.com"
}
```



#### 4.1 Guardar el modelo en 'ICP4D local Watson Machine Learning'.

```
In [115]: from watson_machine_learning_client import WatsonMachineLearningAPIClient

wml_credentials = wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "REDACTED",
    "instance_id": "REDACTED"
}
client = WatsonMachineLearningAPIClient(wml_credentials)
```

3. Corremos la celda siguiente.
4. En las siguientes dos celdas (4.2 y 4.3), cambiamos los valores “**project\_id**” y “**Access\_Token**” por nuestros valores. Para obtener el “**Project ID**”, simplemente vamos al “**Overview**” de nuestro proyecto y copiamos, del “**URL**” dicho ID.

5. Ahora, para obtener el “**Access Token**”, vamos a la pestaña de “**Settings**”, buscamos el apartado de “**Access Token**” y damos clic en “**New Token**”.

My Projects / JN\_VLOS99

Overview Assets Environments Jobs Deployments Access Control **Settings**

**Project information**

Project name JN\_VLOS99

Description

Manual Data Science.  
Uso de Jupyter Notebook con Machine Learning con App para predecir el CHURN de una TelCo.  
Enero 2020

Cancel Save

**Storage**Cloud Object Storage i  
1.92 MB used[Manage in IBM Cloud](#)**Associated services**[Add service ▾ +](#)

NAME	SERVICE TYPE	PLAN	ACTIONS
WML_VLOS99	Watson - Machine Learning		

**Access tokens**

NAME	ROLE	CREATED	LAST USED	ACTIONS
Token1	Editor	29 Jan 2020, 11:08:42 am	never used	⋮

New token +

6. Para crear nuestro “Token”, le vamos a dar un nombre que identifiquemos fácilmente, y en el apartado de “Access role for project”, seleccionamos la opción de “Editor” y damos clic en “Create”.

New Token

Name

Access role for project

Cancel Create

7. Hecho esto, vamos a dar clic en el menú de “hamburguesa” (aparece al pasar el mouse sobre él) y seleccionamos la opción “view”.

## Access tokens

New token +

NAME	ROLE	CREATED	LAST USED	ACTIONS
AT-JN_VLOS99	Editor	29 Jan 2020, 12:34:44 pm	never used	⋮
Token1	Editor	29 Jan 2020, 11:08:42 am	never used	<div style="border: 2px solid red; padding: 2px;">View</div> Revoke

## Integrations

8. Ahora copiamos el valor del token y lo reemplazamos en nuestra Jupyter Notebook, así como con el valor de nuestro “**project ID**” en las celdas de los apartados **4.2** y **4.3** de nuestra “Jupyter Notebook”

## 4.2 Escribe los datos de prueba sin ninguna etiqueta a un .csv, para usarlo después como puntuación por lotes.

```
In [83]: from project_lib import Project
project = Project("", "Project ID", "Access Token", "4")
write_score_CSV=test_data.toPandas().drop(['Churn'], axis=1)

project.save_data(file_name = "TelcoCustomerSparkMLBatchScore.csv", data = write_score_CSV.to_csv(index=False))

Out[83]: {'file_name': 'TelcoCustomerSparkMLBatchScore.csv',
'message': 'File saved to project storage.',
'bucket_name': 'jnvlos99-donotdelete-pr-zjytexkjpbsd11',
'asset_id': '7'}
```

## 4.3 Escribe los datos de prueba a un .csv para usarlos posteriormente para la evaluación.

```
In [84]: from project_lib import Project
project = Project("", "Project ID", "Access Token", "4")
write_eval_CSV=test_data.toPandas()

project.save_data(file_name = "TelcoCustomerSparkMLEval.csv", data = write_eval_CSV.to_csv(index=False))

Out[84]: {'file_name': 'TelcoCustomerSparkMLEval.csv',
'message': 'File saved to project storage.',
'bucket_name': 'jnvlos99-donotdelete-pr-zjytexkjpbsd11',
'asset_id': 'd'}
```

9. Hecho esto ya habrás terminado de crear tu modelo.

## 9. Desplegar el modelo:

1. Para realizar nuestro despliegue, vamos a nuestro proyecto en Watson Studio, y en la pestaña “**Assets**”, buscamos el apartado “**Models**”. Ahí debemos poder ver el modelo que recién creamos. Damos clic en el menú de “hamburguesa” (aparece al pasar el “mouse” sobre él) y seleccionamos la opción. “**Deploy**”.

The screenshot shows the IBM Watson Studio interface. At the top, there's a dark header bar with the IBM Watson Studio logo, a 'Upgrade' button, a notification bell, and a dropdown for 'Viridiana Salvatierra's Account'. Below the header is a navigation bar with 'My Projects / JN\_VLOS99' and icons for Launch IDE, Add to project, and others. The main content area has tabs for Overview, Assets (which is highlighted with a red box), Environments, Jobs, Deployments, Access Control, and Settings. The 'Assets' tab likely leads to the 'Models' section where the deployed model would be listed.

## Notebooks

[New notebook +](#)

NAME	SHARED	SCHEDULED	STATUS	LANGUAGE	LAST EDITOR	LAST MODIFIED	ACTIONS
Telco-customer-CHURN-VLOS99			○	Python 3.6	Viridiana Salvatierra	30 Jan 2020	 

## Deep learning experiments

[New deep learning experiment +](#)

NAME	LAST MODIFIED ▾	ACTIONS
You don't have any Deep learning experiments yet.		

## Models

### Watson Machine Learning models

[Import model +](#)

NAME	TYPE	RUNTIME	LAST MODIFIED ▾	ACTIONS
JN_VLOS_99-Enero-30-2020	mllib-2.3	spark-2.3	30 Jan 2020	
Clasificación logística - Mora.	pmml-4.2	java-1.8	7 Jan 2020	
Modelo de clusterización con K-Means Twitter.	pmml-4.2	java-1.8	7 Jan 2020	
Clasificación-CHAID-Mora	pmml-4.2	java-1.8	6 Jan 2020	

2. Una vez que hemos sido redirigidos a la página del modelo, en la pestaña “Deployments”, damos clic en “Add Deployment”.

MODEL  
JN\_VLOS\_99-Enero-30-2020

Overview   Evaluation   **Deployments**   Lineage



NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Your model is not deployed.			

3. Ahora, le damos un nombre a nuestro despliegue, después, en “Deployment type”, seleccionamos la opción “Web service” y damos clic en “Save”.

IBM Watson Studio   [Upgrade](#)      Viridiana Salvatierra's Account   

Create Deployment

Define deployment details

Name

Description

Deployment type  Web service  Batch prediction



4. Ahora podemos visualizar nuestro despliegue dentro de la página de nuestro modelo, en el apartado “**Deployments**”. Cuando el “status” cambie a “**DEPLOY\_SUCCESS**”, nuestro modelo habrá sido desplegado exitosamente. (Si no cambia el “status” en un par de minutos, te recomendamos refrescar la página.)

My Projects / JN\_VLOS99 / JN\_VLOS\_99-Enero-30-2020

MODEL  
JN\_VLOS\_99-Enero-30-2020

Overview Evaluation Deployments Lineage

Add Deployment +

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Telco-CHURN_VLOS99	DEPLOY_SUCCESS	Web Service	⋮

## 10. Probar el modelo:

1. Una vez que desplegamos el modelo, damos clic en el nombre de nuestro despliegue.

Overview Evaluation Deployments Lineage

Add Deployment +

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Telco-CHURN_VLOS99	DEPLOY_SUCCESS	Web Service	⋮

2. Una vez dentro de la página de nuestro despliegue, damos clic en la pestaña “**Test**” y seleccionamos la opción “**Provide input data as JSON**”(⌘).

Telco-CHURN\_VLOS99

Overview Implementation Test

Enter input data ⌘

Paste the request payload here

Predict

3. Ahora, en el apartado “**Paste the request payload here**”, pegamos lo siguiente:

```
1.      {
2.          "fields": [
3.              "gender",
4.              "SeniorCitizen",
5.              "Partner",
6.              "Dependents",
7.              "tenure",
8.              "PhoneService",
9.              "MultipleLines",
10.                 "InternetService",
11.                 "OnlineSecurity",
12.                 "OnlineBackup",
13.                 "DeviceProtection",
14.                 "TechSupport",
15.                 "StreamingTV",
16.                 "StreamingMovies",
17.                 "Contract",
18.                 "PaperlessBilling",
19.                 "PaymentMethod",
20.                 "MonthlyCharges",
21.                 "TotalCharges"
22.             ],
23.             "values": [
24.                 [
25.                     "Female",
26.                     0,
27.                     "No",
28.                     "No",
29.                     1,
30.                     "No",
31.                     "No phone service",
32.                     "DSL",
33.                     "No",
34.                     "No",
35.                     "No",
36.                     "No",
37.                     "No",
38.                     "No",
39.                     "Month-to-month",
40.                     "No",
41.                     "Bank transfer (automatic)",
42.                     25.25,
43.                     25.25
44.                 ]
45.             ]
46.         }
```

# Telco-CHURN\_VLOS99

Y damos clic en “Predict”.

```
{"fields":["gender","SeniorCitizen",  
"Partner","Dependents","tenure","  
PhoneService","MultipleLines","Int  
ernetService","OnlineSecurity","On  
lineBackup","DeviceProtection","T  
echSupport","StreamingTV","Strea  
mingMovies","Contract","Paperless  
Billing","PaymentMethod","Monthly  
Charges","TotalCharges"],"values":  
[["Female",0,"No","No",1,"No","No  
phone  
service","DSL","No","No","No","No",  
"No","No","Month-to-  
month","No","Bank transfer  
(automatic)".25.25.25.25]]}
```

4. Ahora, podemos visualizar el resultado de nuestra predicción en formato JSON. Al final, debemos poder ver un “**Yes**” o un “**No**” como valor correspondiente al “**CHURN**”.

## Telco-CHURN\_VLOS99

Overview   Implementation   Test

---

**Enter input data**

<pre>"Month-to-month", "No", "Bank transfer (automatic)", 25.25, 25.25 } ]</pre>	<pre>25.25, 25.25 ] }, [ 9.85797807968493, 10.14202192031507 ], [ 0.4928989039842465, 0.5071010960157535 ], 1, 1, "Yes" ] }</pre>
--	---

**Predict**

5. Para poder probarlo realizando un “**Post**” desde la CMD, Terminal, PowerShell, etc., primero debemos obtener la apikey de nuestro servicio de “Watson Machine Learning” desde la página de nuestra instancia. En el apartado de “**Service Credentials**”, desplegamos las credenciales creadas anteriormente y copiamos la apikey.

WML\_VLOS99

Resource group: Default Location: Dallas Add Tags

Service credentials

New credential +

Key Name	Date Created	Actions
Service credentials-1	JAN 28, 2020 - 11:59:35 AM	<a href="#">View credentials</a> <a href="#">Delete</a>

{ "apikey": "REDACTED", "refresh\_token": "REDACTED" }

6. Ahora debemos obtener un Token pegando lo siguiente en la CMD, la Terminal, PowerShell (modo administrador).

```
curl -k -X POST \
--header "Content-Type: application/x-www-form-urlencoded" \
--header "Accept: application/json" \
--data-urlencode "grant_type=urn:ibm:params:oauth:grant-type:apikey" \
--data-urlencode "apikey=(escribe aquí la clave deapikey sin espacios ni paréntesis)" \
"https://iam.bluemix.net/identity/token"
```

```
{"access_token": "REDACTED", "refresh_token": "REDACTED"}:3600,"expiration":1580857563,"scope":"ibm openid"}Viridianas-MBP:- viridianasalvatierra$
```

7. Ahora obtenemos nuestro URL desde la página del despliegue de nuestro proyecto. Dentro de la página de nuestro proyecto, en la pestaña de “Assets”, damos clic en el nombre de nuestro modelo (desde el apartado “Models”).

IBM Watson Studio

My Projects / JN\_VLOS99

Upgrade [+](#) [Bell](#) Viridiana Salvatierra's Account [VS](#)

Overview [Assets](#) Environments Jobs Deployments Access Control Settings

**Models**

Watson Machine Learning models [Import model +](#)

NAME	TYPE	RUNTIME	LAST MODIFIED	ACTIONS
JN_VLOS_99-Enero-30-2020	mllib-2.3	spark-2.3	30 Jan 2020	

8. Una vez dentro de nuestro modelo, en la pestaña “Deployments”, damos clic en el nombre de nuestro despliegue.

Overview   Evaluation   **Deployments**   Lineage

Add Deployment +

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Telco-CHURN_VLOS99	DEPLOY_SUCCESS	Web Service	<span style="color: blue;">⋮</span>

## 9. Una vez dentro de nuestro despliegue, copiamos la dirección de URL.

Overview   **Implementation**   Test

**Implementation**   View API Specification

Scoring End-point	<a href="https://us-south.ml.cloud.ibm.com/v3/wml_instances/c38d5677-f4bc-4e98-aeb8-788d82252e9b/deployments/7a69641d-ab55-4e56-b25b-23febe7369c6/online">https://us-south.ml.cloud.ibm.com/v3/wml_instances/c38d5677-f4bc-4e98-aeb8-788d82252e9b/deployments/7a69641d-ab55-4e56-b25b-23febe7369c6/online</a>
Authorization: Bearer <token>	Review the <a href="#">WML authentication</a> documentation for details about generating IAM tokens.
ML-Instance-ID	The "ML-Instance-ID" HTTP header must be populated with the WML instance id, which can be obtained as <a href="#">described here</a> .
Content-type: application/json	Required if the request body is sent in JSON format.

## 10. Ahora, reemplazamos el access token resultante y la URL en donde corresponde e introducimos los comandos siguientes en la Terminal, CMD, etc.:

```
export WML_AUTH_TOKEN=<sustituir por "access token" sin espacios ni flechas>
export URL=https://<sustituir por el URL de WML sin espacios ni flechas>
```

## 11. Una vez que hemos exportado estos valores, realizamos la consulta introduciendo lo siguiente en nuestra Terminal, CMD, etc.:

```
curl -k -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header "Authorization: Bearer $WML_AUTH_TOKEN" -d '{"fields": ["gender", "SeniorCitizen", "Partner", "Dependents", "tenure", "PhoneService", "MultipleLines", "InternetService", "OnlineSecurity", "OnlineBackup", "DeviceProtection", "TechSupport", "StreamingTV", "StreamingMovies", "Contract", "PaperlessBilling", "PaymentMethod", "MonthlyCharges", "TotalCharges"], "values": [[{"Female", 0, "No", "No", 1, "No", "No phone service", "DSL", "No", "No", "No", "Month-to-month", "No", "Bank transfer (automatic)", 25.25, 25.25}]]}' $URL
```

## 12. Hecho esto, obtendremos el resultado de nuestra consulta como un “YES” o “NO” al final de nuestros resultados.

```
viridianas-mbp:~ viridianasalvatierra$ curl -k -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header "Authorization: Bearer $WML_AUTH_TOKEN" -d '{"fields": ["gender", "SeniorCitizen", "Partner", "Dependents", "tenure", "PhoneService", "MultipleLines", "InternetService", "OnlineSecurity", "OnlineBackup", "DeviceProtection", "TechSupport", "StreamingTV", "StreamingMovies", "Contract", "PaperlessBilling", "PaymentMethod", "MonthlyCharges", "TotalCharges"], "values": [{"Female", 0, "No", "No", 1, "No", "No phone service", "DSL", "No", "No", "No", "Month-to-month", "No", "Bank transfer (automatic)", 25.25, 25.25}]]}' $URL
{
  "fields": ["gender", "SeniorCitizen", "Partner", "Dependents", "tenure", "PhoneService", "MultipleLines", "InternetService", "OnlineSecurity", "OnlineBackup", "DeviceProtection", "TechSupport", "StreamingTV", "StreamingMovies", "Contract", "PaperlessBilling", "PaymentMethod", "MonthlyCharges", "TotalCharges"], "values": [{"Female", 0, "No", "No", 1, "No", "No phone service", "DSL", "No", "No", "No", "Month-to-month", "No", "Bank transfer (automatic)", 25.25, 25.25}], "rawPrediction": [{"label": "No", "probability": 0.0, "prediction": "predictedLabel"}, {"label": "Yes", "probability": 1.0, "prediction": "predictedLabel"}], "features": [{"label": "Female", "value": 0}, {"label": "SeniorCitizen", "value": 0}, {"label": "Partner", "value": 0}, {"label": "Dependents", "value": 1}, {"label": "tenure", "value": "Month-to-month"}, {"label": "PhoneService", "value": "No"}, {"label": "MultipleLines", "value": "No"}, {"label": "InternetService", "value": "DSL"}, {"label": "OnlineSecurity", "value": "No"}, {"label": "OnlineBackup", "value": "No"}, {"label": "DeviceProtection", "value": "No"}, {"label": "TechSupport", "value": "No"}, {"label": "StreamingTV", "value": "No"}, {"label": "StreamingMovies", "value": "No"}, {"label": "Contract", "value": "Month-to-month"}, {"label": "PaperlessBilling", "value": "No"}, {"label": "PaymentMethod", "value": "No"}, {"label": "MonthlyCharges", "value": 25.25}, {"label": "TotalCharges", "value": 25.25}], "predictedLabel": "Yes", "predictedProbability": 1.0, "rawPrediction": [{"label": "No", "probability": 0.0}, {"label": "Yes", "probability": 1.0}], "label": "Yes", "probability": 1.0}
viridianas-mbp:~ viridianasalvatierra$
```

# 11. Desplegar la App:

1. Antes de empezar, es muy importante haber clonado el repositorio requerido en los [prerrequisitos](#), además de tener instalado Python 3.
2. Accedemos a la carpeta (descomprimida) en la Terminal, CMD, etc.

```
[viridianas-mbp:GitHub viridianasalvatierra$ cd CHURN-app-Jupyter-Notebook-y-WS
[viridianas-mbp:CHURN-app-Jupyter-Notebook-y-WS viridianasalvatierra$ ls
CONTRIBUTING.md LICENSE          MAINTAINERS.md README.md      data           doc           examples        flaskapp       notebooks
viridianas-mbp:CHURN-app-Jupyter-Notebook-y-WS viridianasalvatierra$ ]
```

3. Ahora, creamos nuestro ambiente (“enviroment”).

```
python -m venv venv
source venv/bin/activate  (Mac / Linux)
./venv/Scripts/activate  (Windows)
```

4. Instalamos los requerimientos de Python.

```
cd flaskapp
pip install -r requirements.txt
```

5. Ahora, editamos el archivo “env.sample”, le añadimos nuestras credenciales y lo guardamos como “.env” (**Si llegas a tener errores al correr la app, vuelve a sacar el token y cámbialo en el archivo “.env”, pues es temporal**).

```
# Copy this file to .env.
# Edit the .env file with the required settings before starting the app.

# Required: Provide your web service URL for scoring.
# E.g., URL=https://9.10.222.3:31843/dmodel/v1/project/pyscript/tag/score

URL=(aquí pegamos el URL que obtuvimos en el paso 8.7 sin paréntesis)

# Required: Provide your web service deployment access token.
#           This TOKEN will be the part after `accessToken`. So, your
#           json string will look like:
#
>{"username":"scottda","role":"Admin","permissions":["administrator","can_provision","manage_catalog","virtualize_transform","access_catalog"],"sub":"scottda","iss":"KNOXSSO","aud":"DSX","uid":"1000331001","authenticator":"default","accessToken":"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2 <snip>
neQ","_messageCode_":"success","message":"success"}
# The value for `TOKEN=` below will be:
#     TOKEN=eyJhbGciOi <snip> neQ

TOKEN=(aquí pegamos el token que obtuvimos en el paso 8.6 sin paréntesis)

# Optional: You can override the server's host and port here.

HOST=0.0.0.0
PORT=5000
```

6. Una vez hecho esto, corremos nuestra app con el siguiente comando:

```
python telcochurn.py
```

7. Ahora podemos acceder a nuestra app de forma local, pegando en el navegador lo siguiente:

<http://0.0.0.0:5000/>

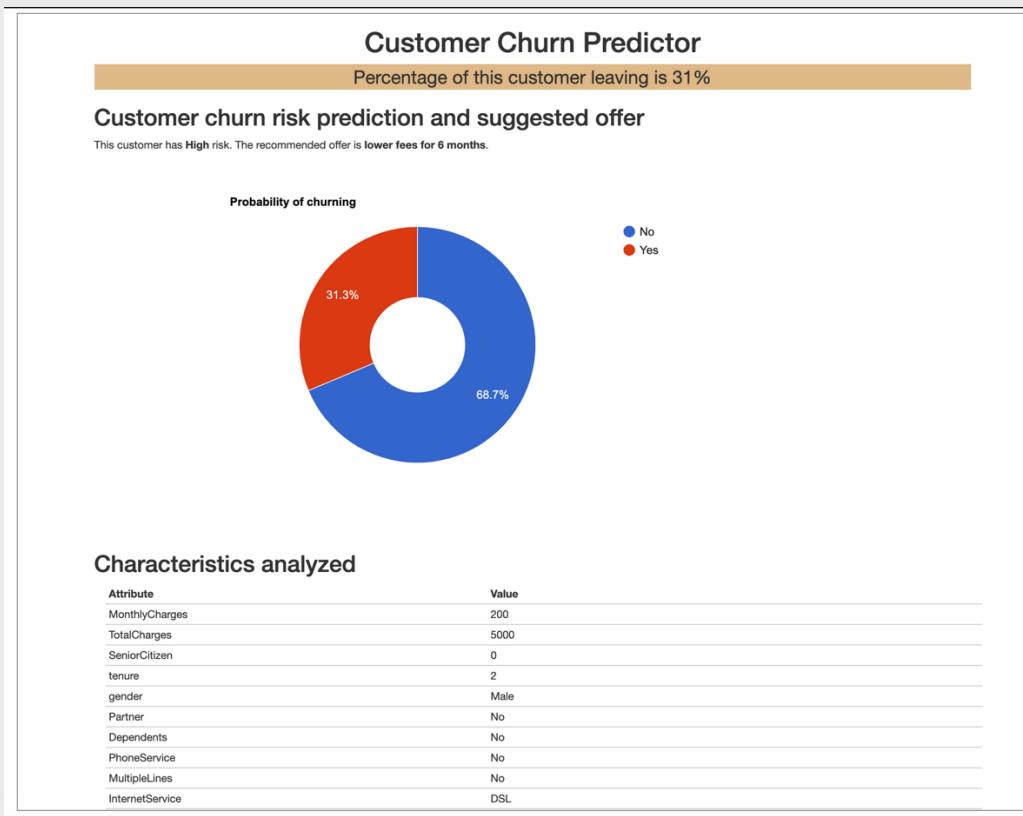
### Customer Churn Predictor

**Will a customer stay or leave?**

Input the following information and click submit to make a prediction.

MonthlyCharges 200	TotalCharges 5000
SeniorCitizen 0	tenure 2
gender Male	Partner No
Dependents No	PhoneService No
MultipleLines No	InternetService DSL
OnlineSecurity No	OnlineBackup No
DeviceProtection No	TechSupport No
StreamingTV No	StreamingMovies No
Contract Month-to-month	PaperlessBilling No
PaymentMethod Credit card (automatic)	

**Submit**



## 12. Siguientes Pasos

---

Te invitamos a explorar otros talleres y manuales en el siguiente enlace de Github.  
<https://github.com/ibmdevelopermx>