

Abstract geometric lines in the top left corner, consisting of several overlapping, irregular polygons and lines in a light beige color.

PRÁCTICA 1 DE SIE II

Paloma Pérez de Madrid

ENUNCIADO

Modelo: nombre del modelo. Por ejemplo: MODELO_1_CROSSELL o XSELL o MODELIT

Periodo: string en formato YYYYMM31

Publico Objetivo: En este caso, existen dos posibles valores, a saber, EMPRESAS O RESIDENCIAL

ID_CLIENTE: un identificador unívoco del cliente

Propension: probabilidad de que este cliente (el definido por id_cliente) haga o no cross sell. Por tanto, es un número entre 0 y 1

El ejercicio trata de simular un proceso de scoring de modelos de aprendizaje automático.

Para ello, vais a definir una función de pega que se llame scoring. Esta función va a simular el resultado del resultado de vuestro modelo. Tiene que devolver un fichero de 100 registros mínimo que siga la estructura anterior.

ENUNCIADO

Sobre este fichero:

Realiza un análisis del dataframe usando visualizaciones y llamadas de pandas

Sube los escorings a GCS

Define una Cloud Función que cargue los resultados a una tabla de BQ. El nombre de la tabla tiene que ser `scoring_mensual_xsell_periodo`, donde `periodo` es el periodo definido en el campo de la tabla, por ejemplo, `20221231`.

Haz una consulta a la tabla anterior que incluya únicamente a los clientes de empresas y genera una nueva tabla que se denomine `scoring_mensual_empresas`.

Descarga los datos. En BQ, cuando se excede un cupo de capacidad al exportar se generan diferentes particiones de los datos para almacenar como ficheros. Normalmente, esto se hace también poniendo un `*` en el nombre del archivo a exportar, por ejemplo, `scoringmensual*.csv`. Asume que se han generado esos ficheros en cloud storage y haz una función que los descargue todos a un único dataframe.

Define una función que añada los registros de la tabla anterior (`scoring_mensual_xsell_periodo`) a una nueva tabla que contenga todos los registros históricos denominada `scoring_historico_xsell`



ÍNDICE

1. Crear una función llamada `scoring` que genere un archivo con 100 registros mínimos con la estructura especificada.
2. Realizar un análisis del dataframe generado por la función `scoring` utilizando visualizaciones y funciones de Pandas.
3. Subir los resultados del scoring a Google Cloud Storage (GCS).
4. Definir una Cloud Function que cargue los resultados a una tabla de BigQuery (BQ) con el nombre `scoring_mensual_xsell_periodo`.
5. Realizar una consulta a la tabla `scoring_mensual_xsell_periodo` para filtrar únicamente los clientes de empresas y generar una nueva tabla llamada `scoring_mensual_empresas`.
6. Descargar los datos de la tabla `scoring_mensual_empresas` almacenados en diferentes particiones en Cloud Storage y cargarlos en un único DataFrame.
7. Definir una función que añada los registros de la tabla `scoring_mensual_xsell_periodo` a una nueva tabla que contenga todos los registros históricos denominada `scoring_historico_xsell`.



ÍNDICE

1. **Crear una función llamada `scoring` que genere un archivo con 100 registros mínimos con la estructura especificada.**
2. Realizar un análisis del dataframe generado por la función `scoring` utilizando visualizaciones y funciones de Pandas.
3. Subir los resultados del scoring a Google Cloud Storage (GCS).
4. Definir una Cloud Function que cargue los resultados a una tabla de BigQuery (BQ) con el nombre `scoring_mensual_xsell_periodo`.
5. Realizar una consulta a la tabla `scoring_mensual_xsell_periodo` para filtrar únicamente los clientes de empresas y generar una nueva tabla llamada `scoring_mensual_empresas`.
6. Descargar los datos de la tabla `scoring_mensual_empresas` almacenados en diferentes particiones en Cloud Storage y cargarlos en un único DataFrame.
7. Definir una función que añada los registros de la tabla `scoring_mensual_xsell_periodo` a una nueva tabla que contenga todos los registros históricos denominada `scoring_historico_xsell`

1. CREAR UNA FUNCIÓN LLAMADA `SCORING` QUE GENERE UN ARCHIVO CON 100 REGISTROS MÍNIMOS CON LA ESTRUCTURA ESPECIFICADA.

```
# 1) Creamos función scoring que genere un archivo con 100 registros mínimos con la estructura especificada.
def scoring():
    np.random.seed(0) # Para reproducibilidad
    df = pd.DataFrame({
        'Modelo': np.random.choice(['MODELO_1_CROSSELL', 'MODELO_2_CROSSELL', 'MODELO_3_PALOMA'], 100),
        'Periodo': np.random.choice(['16022024', '17022024', '18022024', '19022024', '20022024'], 100),
        'Publico Objetivo': np.random.choice(['EMPRESAS', 'RESIDENCIAL'], 100),
        'ID_CLIENTE': np.random.randint(1000, 2000, size=100),
        'Propension': np.random.rand(100)
    })

    df.to_csv('scoring_result.csv', index=False)
    print(df)

scoring()
```

	Modelo	Periodo	Publico Objetivo	ID_CLIENTE	Propension
0	MODELO_1_CROSSELL	18022024	EMPRESAS	1971	0.488056
1	MODELO_2_CROSSELL	16022024	EMPRESAS	1662	0.355613
2	MODELO_1_CROSSELL	17022024	RESIDENCIAL	1414	0.940432
3	MODELO_2_CROSSELL	16022024	RESIDENCIAL	1657	0.765325
4	MODELO_2_CROSSELL	18022024	RESIDENCIAL	1710	0.748664
..
95	MODELO_1_CROSSELL	19022024	RESIDENCIAL	1786	0.799796
96	MODELO_3_PALOMA	16022024	RESIDENCIAL	1373	0.076956
97	MODELO_1_CROSSELL	19022024	RESIDENCIAL	1290	0.518835
98	MODELO_2_CROSSELL	16022024	RESIDENCIAL	1563	0.306810
99	MODELO_3_PALOMA	20022024	RESIDENCIAL	1670	0.577543

[100 rows x 5 columns]



ÍNDICE

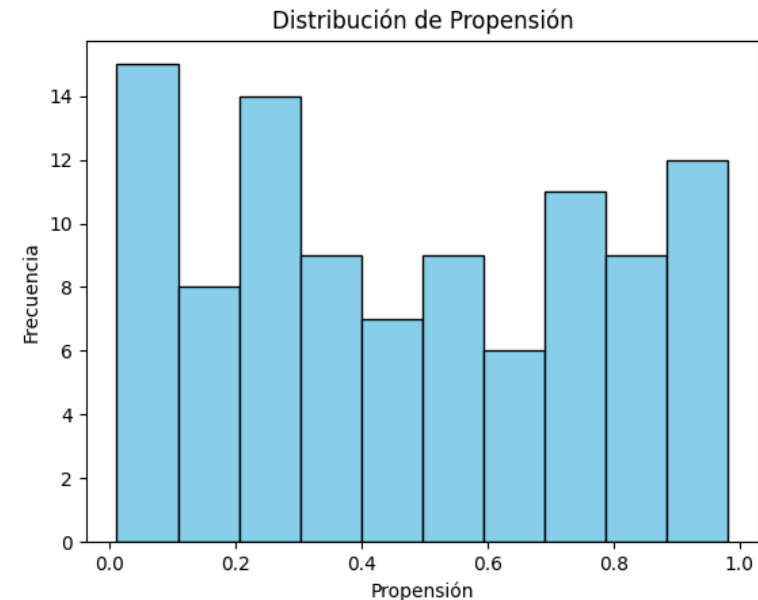
1. Crear una función llamada `scoring` que genere un archivo con 100 registros mínimos con la estructura especificada.
2. **Realizar un análisis del dataframe generado por la función `scoring` utilizando visualizaciones y funciones de Pandas.**
3. Subir los resultados del scoring a Google Cloud Storage (GCS).
4. Definir una Cloud Function que cargue los resultados a una tabla de BigQuery (BQ) con el nombre `scoring_mensual_xsell_periodo`.
5. Realizar una consulta a la tabla `scoring_mensual_xsell_periodo` para filtrar únicamente los clientes de empresas y generar una nueva tabla llamada `scoring_mensual_empresas`.
6. Descargar los datos de la tabla `scoring_mensual_empresas` almacenados en diferentes particiones en Cloud Storage y cargarlos en un único DataFrame.
7. Definir una función que añada los registros de la tabla `scoring_mensual_xsell_periodo` a una nueva tabla que contenga todos los registros históricos denominada `scoring_historico_xsell`

2. REALIZAR UN ANÁLISIS DEL DATAFRAME GENERADO POR LA FUNCIÓN `SCORING` UTILIZANDO VISUALIZACIONES Y FUNCIONES DE PANDAS.

Vamos a realizar los siguientes análisis estadísticos

- Histograma de la propensión
- Boxplot: comparar propensión entre Empresa y Residencial
- Boxplot: comparar por modelo
- Countplot: Análisis de frecuencia por modelo y público objetivo

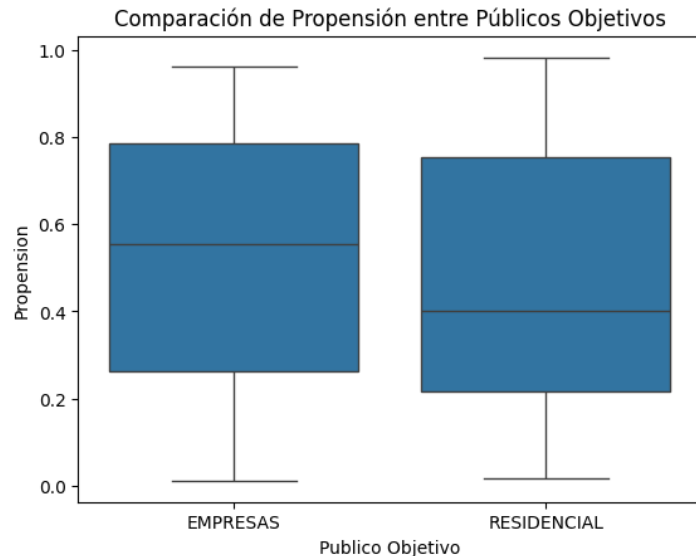
```
# Histograma --> distribución de la propensión
print("\n=====HISTOGRAMA=====")
plt.hist(df['Propension'], bins=10, color='skyblue', edgecolor='black')
plt.xlabel('Propensión')
plt.ylabel('Frecuencia')
plt.title('Distribución de Propensión')
plt.show()
```



(Observamos que no hay ningún tipo de sesgo de propensión pues ocurren

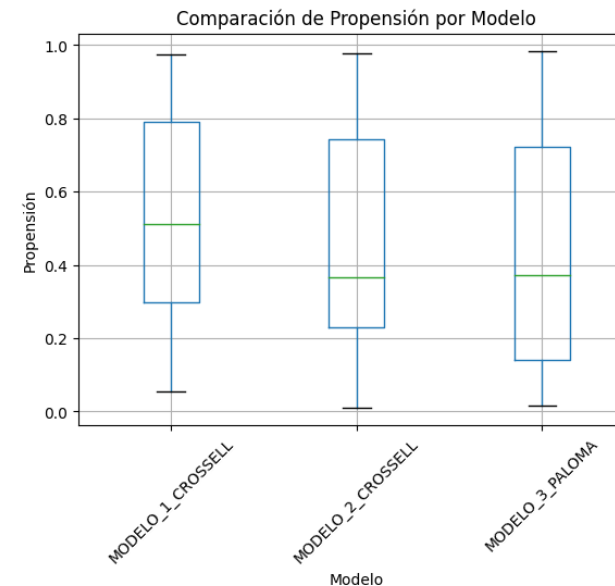
2. REALIZAR UN ANÁLISIS DEL DATAFRAME GENERADO POR LA FUNCIÓN 'SCORING' UTILIZANDO VISUALIZACIONES Y FUNCIONES DE PANDAS.

```
# Comparar propensión entre EMPRESA y RESIDENCIAL
# usaremos boxplot --> seaborn
print("\n=====BOXPLOT=====\\n")
sns.boxplot(x='Publico Objetivo', y='Propension', data=df)
plt.title('Comparación de Propensión entre Públicos Objetivos')
plt.show()
```



Se observa que el residencial tiene una propensión ligeramente menor, a igual que su mediana.

```
# Comparación de propensión por modelo
plt.figure(figsize=(10, 6))
df.boxplot(column='Propension', by='Modelo')
plt.title('Comparación de Propensión por Modelo')
plt.ylabel('Propensión')
plt.xlabel('Modelo')
plt.suptitle('')
plt.xticks(rotation=45)
plt.show()
```

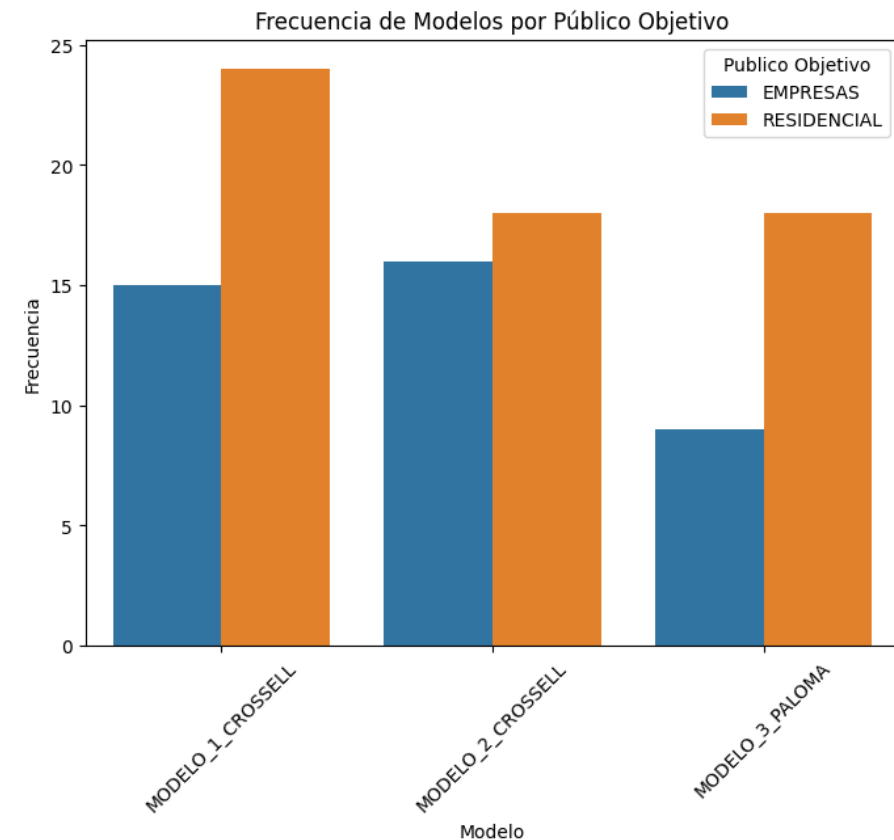


Al igual que el anterior modelo, podemos observar que la mediana del modelo 3 y 2 son similares, siendo la del modelo 1 significativamente mayor a estos dos últimos modelos. Se puede observar que el modelo 3 es el modelo con mayor variación de propensión.

2. REALIZAR UN ANÁLISIS DEL DATAFRAME GENERADO POR LA FUNCIÓN `SCORING` UTILIZANDO VISUALIZACIONES Y FUNCIONES DE PANDAS.

```
# Análisis de frecuencia por modelo y público objetivo
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Modelo', hue='Publico Objetivo')
plt.title('Frecuencia de Modelos por Público Objetivo')
plt.xlabel('Modelo')
plt.ylabel('Frecuencia')
plt.xticks(rotation=45)
plt.show()
```

Vemos que el modelo 1 destaca por tener una mayor cantidad de residencial respecto al resto de modelos. A su vez, el modelo 3 destaca por tener una menor cantidad de Empresa.





ÍNDICE

1. Crear una función llamada `scoring` que genere un archivo con 100 registros mínimos con la estructura especificada.
2. Realizar un análisis del dataframe generado por la función `scoring` utilizando visualizaciones y funciones de Pandas.
3. **Subir los resultados del scoring a Google Cloud Storage (GCS).**
4. Definir una Cloud Function que cargue los resultados a una tabla de BigQuery (BQ) con el nombre `scoring_mensual_xsell_periodo`.
5. Realizar una consulta a la tabla `scoring_mensual_xsell_periodo` para filtrar únicamente los clientes de empresas y generar una nueva tabla llamada `scoring_mensual_empresas`.
6. Descargar los datos de la tabla `scoring_mensual_empresas` almacenados en diferentes particiones en Cloud Storage y cargarlos en un único DataFrame.
7. Definir una función que añada los registros de la tabla `scoring_mensual_xsell_periodo` a una nueva tabla que contenga todos los registros históricos denominada `scoring_historico_xsell`

3. SUBIR LOS RESULTADOS A GOOGLE CLOUD STORAGE

```
"""  
3. Subir la función a GCS  
"""
```

```
nombre_bucket = 'ejemplo_sieii_bucket'  
!gsutil cp scoring_result.csv gs://ejemplo_sieii_bucket/
```

```
Copying file://scoring_result.csv [Content-Type=text/csv]...  
/ [1 files][ 6.1 KiB/ 6.1 KiB]  
Operation completed over 1 objects/6.1 KiB.
```

```
# Comprobación del bucket  
!gsutil ls gs://ejemplo_sieii_bucket/
```

```
gs://ejemplo_sieii_bucket/Registro_archivos_exámenes.csv  
gs://ejemplo_sieii_bucket/ejemplito.csv  
gs://ejemplo_sieii_bucket/scoring_result.csv  
gs://ejemplo_sieii_bucket/trips-1.csv  
gs://ejemplo_sieii_bucket/trips-2.csv  
gs://ejemplo_sieii_bucket/Datasets/
```



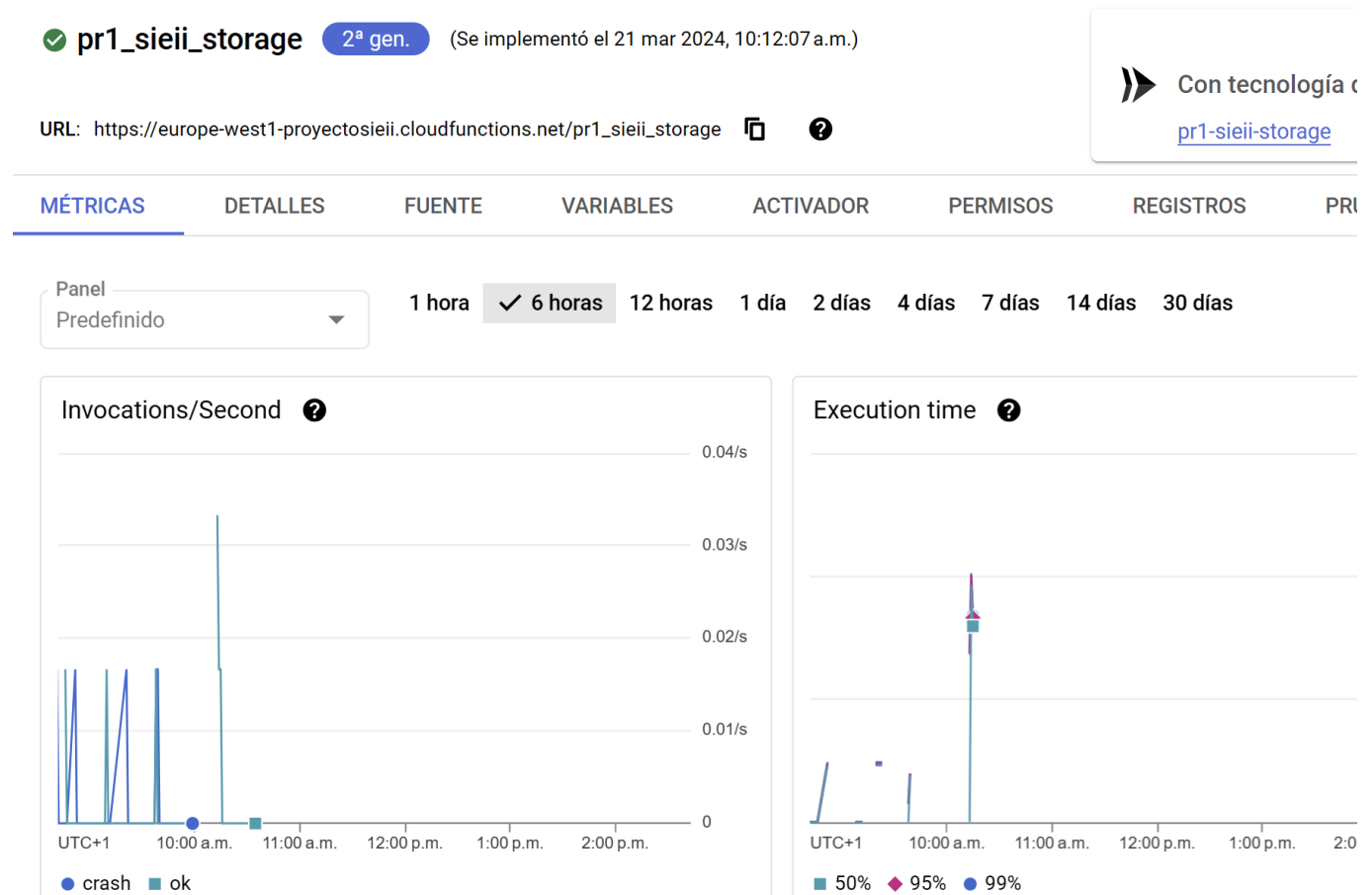
ÍNDICE

1. Crear una función llamada `scoring` que genere un archivo con 100 registros mínimos con la estructura especificada.
2. Realizar un análisis del dataframe generado por la función `scoring` utilizando visualizaciones y funciones de Pandas.
3. Subir los resultados del scoring a Google Cloud Storage (GCS).
4. **Definir una Cloud Function que cargue los resultados a una tabla de BigQuery (BQ) con el nombre `scoring_mensual_xsell_periodo`.**
5. Realizar una consulta a la tabla `scoring_mensual_xsell_periodo` para filtrar únicamente los clientes de empresas y generar una nueva tabla llamada `scoring_mensual_empresas`.
6. Descargar los datos de la tabla `scoring_mensual_empresas` almacenados en diferentes particiones en Cloud Storage y cargarlos en un único DataFrame.
7. Definir una función que añada los registros de la tabla `scoring_mensual_xsell_periodo` a una nueva tabla que contenga todos los registros históricos denominada `scoring_historico_xsell`.

4. DEFINIR UNA CLOUD FUNCTION QUE CARGUE LOS RESULTADOS A UNA TABLA DE BIGQUERY (BQ) CON EL NOMBRE `SCORING_MENSUAL_XSELL_PERIODO`.

Se debe activar cuando subamos 'scoring_result.csv' al bucket

- Cloud Function
- Cloud storage
- Archived



4. DEFINIR UNA CLOUD FUNCTION QUE CARGUE LOS RESULTADOS A UNA TABLA DE BIGQUERY (BQ) CON EL NOMBRE `SCORING_MENSUAL_XSELL_PERIODO`.

```
import functions_framework
from google.cloud import bigquery, storage
import pandas as pd
import numpy as np
# Triggered by a change in a storage bucket
@functions_framework.cloud_event
def cargar_datos_a_bq(cloud_event):
    data = cloud_event.data

    event_id = cloud_event["id"]
    event_type = cloud_event["type"]

    bucket = data["bucket"]
    name = data["name"]
    metageneration = data["metageneration"]
    timeCreated = data["timeCreated"]
    updated = data["updated"]

    print(f"Event ID: {event_id}")
    print(f"Event type: {event_type}")
    print(f"Bucket: {bucket}")
    print(f"File: {name}")
```

```
bq_client = bigquery.Client()
cliente_gcs = storage.Client()

# 1. job_config
job_config = bigquery.LoadJobConfig()

## 2. Autodetect = True. Especificando el esquema
job_config.autodetect = True

## 3. Como es csv: decimos que obvie la primera fila y que el formato es CSV
(por defecto, es CSV).

job_config.skip_leading_rows=1
job_config.source_format=bigquery.SourceFormat.CSV

## 4. Tipo de escritura
job_config.write_disposition = 'WRITE_TRUNCATE'
uri = 'gs://' + bucket + '/' + name
table_ref = 'ceu_dataset1.' + name.replace('.csv', '')
##Dataset.nombre_de_la_clase

load_job = bq_client.load_table_from_uri(uri, table_ref,
job_config=job_config)\
```

4. DEFINIR UNA CLOUD FUNCTION QUE CARGUE LOS RESULTADOS A UNA TABLA DE BIGQUERY (BQ) CON EL NOMBRE `SCORING_MENSUAL_XSELL_PERIODO`.

```
print("Comenzando job {}".format(load_job.job_id))
load_job.result() # Espera a que la tabla esté subida
print("Job terminado")
print('Errores?')
print(load_job.errors)

"""
Requirements.txt:
    functions_framework
    pandas
    google.cloud.bigquery
    google.cloud.storage
"""
```

Para empezar con BigQuery se debe volver a subir el archivo al Bucket y comprobar el funcionamiento de la cloud functions (entrar en Registros o Logs)



ÍNDICE

1. Crear una función llamada `scoring` que genere un archivo con 100 registros mínimos con la estructura especificada.
2. Realizar un análisis del dataframe generado por la función `scoring` utilizando visualizaciones y funciones de Pandas.
3. Subir los resultados del scoring a Google Cloud Storage (GCS).
4. Definir una Cloud Function que cargue los resultados a una tabla de BigQuery (BQ) con el nombre `scoring_mensual_xsell_periodo`.
5. **Realizar una consulta a la tabla `scoring_mensual_xsell_periodo` para filtrar únicamente los clientes de empresas y generar una nueva tabla llamada `scoring_mensual_empresas`.**
6. Descargar los datos de la tabla `scoring_mensual_empresas` almacenados en diferentes particiones en Cloud Storage y cargarlos en un único DataFrame.
7. Definir una función que añada los registros de la tabla `scoring_mensual_xsell_periodo` a una nueva tabla que contenga todos los registros históricos denominada `scoring_historico_xsell`

5. REALIZAR UNA CONSULTA A LA TABLA `SCORING_MENSUAL_XSELL_PERIODO` PARA FILTRAR ÚNICAMENTE LOS CLIENTES DE EMPRESAS Y GENERAR UNA NUEVA TABLA LLAMADA `SCORING_MENSUAL_EMPRESAS`.

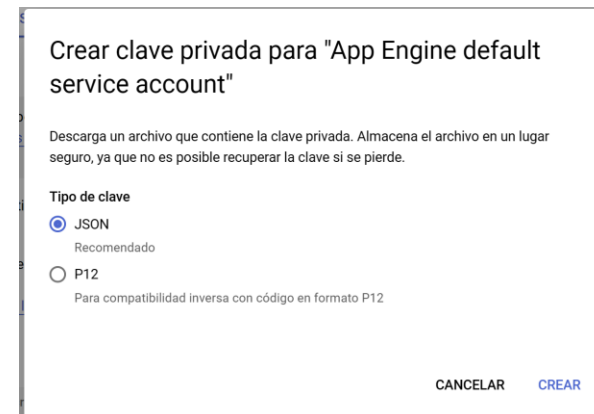
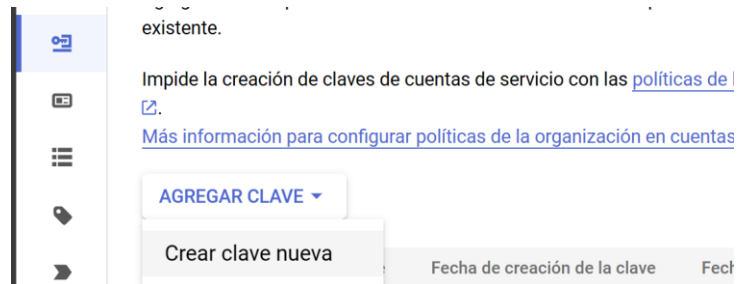
Prerrequisitos para hacerlo desde Google Colab:

```
!pip install google-cloud-bigquery
```

```
import os  
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "/content/proyectosieii-f9fb9c456410.json"
```

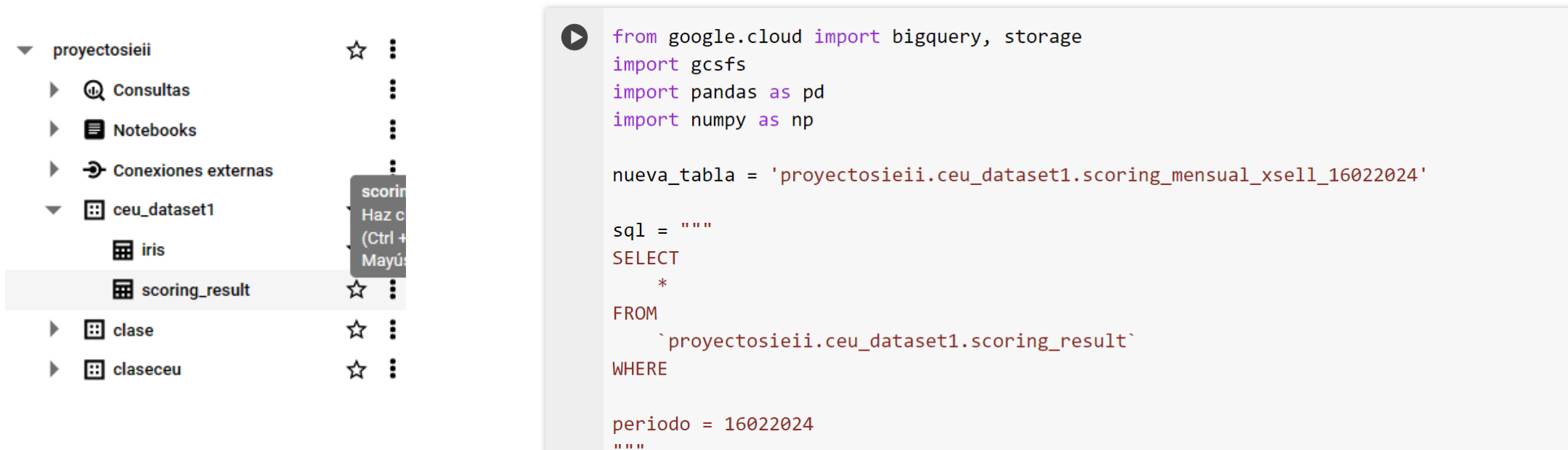
Para descargarse el Json con las credenciales:

Cuentas de servicio >> Elegir la cuenta correspondiente >> tres puntitos >> Administrar claves



5. REALIZAR UNA CONSULTA A LA TABLA `SCORING_MENSUAL_XSELL_PERIODO` PARA FILTRAR ÚNICAMENTE LOS CLIENTES DE EMPRESAS Y GENERAR UNA NUEVA TABLA LLAMADA `SCORING_MENSUAL_EMPRESAS`.

BigQuery



The screenshot displays the BigQuery interface. On the left, the project hierarchy is shown: 'proyectosieii' contains 'Consultas', 'Notebooks', 'Conexiones externas', and 'ceu_dataset1'. Under 'ceu_dataset1', there are tables 'iris' and 'scoring_result'. The 'scoring_result' table is selected, and a context menu is open with options like 'scoring_result', 'Haz clic aquí para crear una nueva consulta (Ctrl + N)', and 'Mayúscula'. On the right, a SQL query is shown in a code editor:

```
from google.cloud import bigquery, storage
import gcsfs
import pandas as pd
import numpy as np

nueva_tabla = 'proyectosieii.ceu_dataset1.scoring_mensual_xsell_16022024'

sql = """
SELECT
    *
FROM
    `proyectosieii.ceu_dataset1.scoring_result`
WHERE

periodo = 16022024
"""
```

5. REALIZAR UNA CONSULTA A LA TABLA `SCORING_MENSUAL_XSELL_PERIODO` PARA FILTRAR ÚNICAMENTE LOS CLIENTES DE EMPRESAS Y GENERAR UNA NUEVA TABLA LLAMADA `SCORING_MENSUAL_EMPRESAS`.

```
job_config = bigquery.QueryJobConfig(destination=nueva_tabla)
query_job = bq_client.query(sql, job_config=job_config) # Make an API request.
query_job.result() # Wait for the job to complete.
```

```
print("Query results loaded to the table {}".format(nueva_tabla))
```

Query results loaded to the table proyectosieii.ceu_dataset1.scoring_mensual_xsell_16022024

5. REALIZAR UNA CONSULTA A LA TABLA `SCORING_MENSUAL_XSELL_PERIODO` PARA FILTRAR ÚNICAMENTE LOS CLIENTES DE EMPRESAS Y GENERAR UNA NUEVA TABLA LLAMADA `SCORING_MENSUAL_EMPRESAS`.

```
bq_client = bigquery.Client()  
bigquery_df = bq_client.query(sql).to_dataframe()  
bigquery_df .head()
```

No es necesario, pero lo he pasado a dataframe para visualizarlo

	Modelo	Periodo	Publico_Objetivo	ID_CLIENTE	Propension
0	MODELO_3_PALOMA	16022024	EMPRESAS	1575	0.941378
1	MODELO_3_PALOMA	16022024	RESIDENCIAL	1606	0.293020
2	MODELO_3_PALOMA	16022024	RESIDENCIAL	1136	0.981829
3	MODELO_3_PALOMA	16022024	RESIDENCIAL	1859	0.497391
4	MODELO_3_PALOMA	16022024	RESIDENCIAL	1093	0.639473





ÍNDICE

1. Crear una función llamada `scoring` que genere un archivo con 100 registros mínimos con la estructura especificada.
2. Realizar un análisis del dataframe generado por la función `scoring` utilizando visualizaciones y funciones de Pandas.
3. Subir los resultados del scoring a Google Cloud Storage (GCS).
4. Definir una Cloud Function que cargue los resultados a una tabla de BigQuery (BQ) con el nombre `scoring_mensual_xsell_periodo`.
5. Realizar una consulta a la tabla `scoring_mensual_xsell_periodo` para filtrar únicamente los clientes de empresas y generar una nueva tabla llamada `scoring_mensual_empresas`.
6. **Descargar los datos de la tabla `scoring_mensual_empresas` almacenados en diferentes particiones en Cloud Storage y cargarlos en un único DataFrame.**
7. Definir una función que añada los registros de la tabla `scoring_mensual_xsell_periodo` a una nueva tabla que contenga todos los registros históricos denominada `scoring_historico_xsell`

6. REALIZAR UNA CONSULTA A LA TABLA `SCORING_MENSUAL_XSELL_PERIODO` PARA FILTRAR ÚNICAMENTE LOS CLIENTES DE EMPRESAS Y GENERAR UNA NUEVA TABLA LLAMADA `SCORING_MENSUAL_EMPRESAS`.

```
sql = """
SELECT
    *
FROM
    `proyectosieii.ceu_dataset1.scoring_mensual_xsell_16022024`
WHERE

Publico_Objetivo = 'EMPRESAS'
"""

bq_client = bigquery.Client()
empresas_df = bq_client.query(sql).to_dataframe()
empresas_df.head()
```

	Modelo	Periodo	Publico_Objetivo	ID_CLIENTE	Propension
0	MODELO_3_PALOMA	16022024	EMPRESAS	1575	0.941378
1	MODELO_1_CROSSELL	16022024	EMPRESAS	1284	0.089603
2	MODELO_1_CROSSELL	16022024	EMPRESAS	1136	0.695625
3	MODELO_1_CROSSELL	16022024	EMPRESAS	1147	0.511319
4	MODELO_2_CROSSELL	16022024	EMPRESAS	1282	0.259423





ÍNDICE

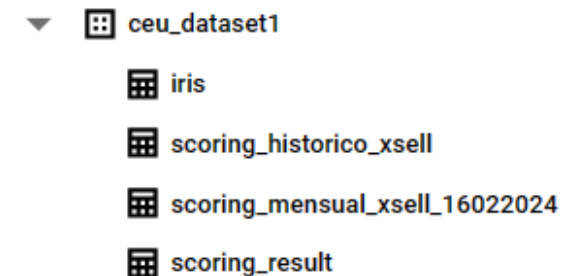
1. Crear una función llamada `scoring` que genere un archivo con 100 registros mínimos con la estructura especificada.
2. Realizar un análisis del dataframe generado por la función `scoring` utilizando visualizaciones y funciones de Pandas.
3. Subir los resultados del scoring a Google Cloud Storage (GCS).
4. Definir una Cloud Function que cargue los resultados a una tabla de BigQuery (BQ) con el nombre `scoring_mensual_xsell_periodo`.
5. Realizar una consulta a la tabla `scoring_mensual_xsell_periodo` para filtrar únicamente los clientes de empresas y generar una nueva tabla llamada `scoring_mensual_empresas`.
6. Descargar los datos de la tabla `scoring_mensual_empresas` almacenados en diferentes particiones en Cloud Storage y cargarlos en un único DataFrame.
7. **Definir una función que añada los registros de la tabla `scoring_mensual_xsell_periodo` a una nueva tabla que contenga todos los registros históricos denominada `scoring_historico_xsell`**

7. DEFINIR UNA FUNCIÓN QUE AÑADA LOS REGISTROS DE LA TABLA `SCORING_MENSUAL_XSELL_PERIODO` A UNA NUEVA TABLA QUE CONTENGA TODOS LOS REGISTROS HISTÓRICOS DENOMINADA `SCORING_HISTORICO_XSELL`

```
# Función que añade los registros de una tabla a 'scoring_historico_xsell'

def add_tabla_historico(tabla):
    tabla_historicos = 'proyectosieii.ceu_dataset1.scoring_historico_xsell'
    sql = f'SELECT * FROM `proyectosieii.ceu_dataset1.{tabla}`'
    bq_client = bigquery.Client()
    job_config = bigquery.QueryJobConfig(destination=tabla_historicos)
    query_job = bq_client.query(sql, job_config=job_config) # Make an API request.
    query_job.result() # Wait for the job to complete.
    print("Query results loaded to the table {}".format(nueva_tabla))

add_tabla_historico("scoring_mensual_xsell_16022024")
```



→ No es nueva_tabla sino “tabla_historicos”

Si intentas ejecutar otra vez la función te dará error, ya has agregado la tabla a scoring_mensual_xsell