

# Despliegue de modelos en GCP


## Links

Todo el proyecto esta en mi github por si quieres revisar el código, ya que se verá mejor que en este documento.

[https://github.com/guiller91/Modelo\\_predictivo](https://github.com/guiller91/Modelo_predictivo)

GitHub - guiller91/Modelo\_predictivo: Ejercicio de IA y big data donde se realiza un modelo predictivo usando una regresión lineal.

Ejercicio de IA y big data donde se realiza un modelo predictivo usando una regresión lineal. - GitHub - guiller91/Modelo\_predictivo: Ejercicio de IA y big data donde se realiza un modelo predictivo usando una regresión lineal.

 [https://github.com/guiller91/Modelo\\_predictivo](https://github.com/guiller91/Modelo_predictivo)

guiller91/  
**Modelo\_predictivo**

Ejercicio de IA y big data donde se realiza un modelo predictivo usando una regresión lineal.

Re 1

Com 0

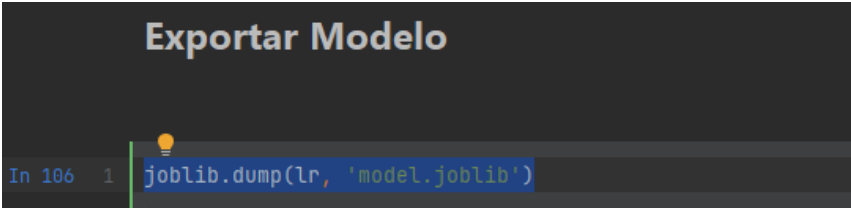
Star 0

Fork 0

## Crear y guardar el modelo en formato joblib

Para exportar el modelo ya entrenado usamos el siguiente comando.

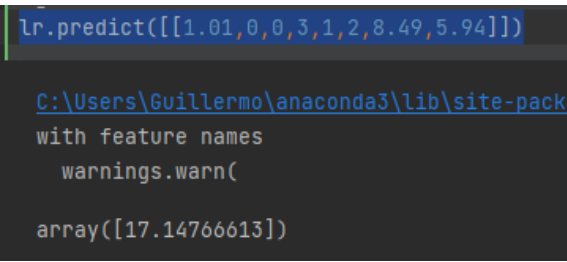
```
joblib.dump(lr, 'model.joblib')
```



También pruebo a predecir un dato para tenerlo en cuenta a la hora de hacer una llamada al endpoint que crearé.

```
lr.predict([[1.01,0,0,3,1,2,8.49,5.94]])
```

Que nos da un resultado de 17.14€.



## Creación cuenta GCP y Creación del Bucket

Una vez que hemos seguido los pasos y hemos creado la cuenta en [Google Cloud Platform](#) , que es básicamente rellenar los campos que nos pide con nuestra información, asignar un método de pago y aceptar, creamos un proyecto.

Seleccionar un proyecto

PROYECTO NUEVO

Buscar en proyectos y carpetas

RECIENTES

DESTACADOS

TODOS

Nombre	ID
<div><div>✓</div><div>☆</div><div></div><div>sklearn tip</div><div></div></div>	sklearn-tip
<div><div>☆</div><div></div><div></div><div>My First Project</div><div></div></div>	woven-answer-367108
<div><div>☆</div><div></div><div></div><div>My Project 52472</div><div></div></div>	t-osprey-354219

CANCELAR

ABRIR

Después vamos al apartado de **Cloud Storage > Buckets** y daremos al botón **CREAR**. Tendremos que asignar un nombre, elegir los datos donde almacenar, etc.

!

Los textos en negrita y cursiva tienen un link directo.

Buckets

CREAR

ACTUALIZAR

APRENDIZAJE

Filtro

Filtrar depósitos

Nombre	Fecha de creación	Tipo de ubicación	Ubicación	Default storage class	Última modificación	Acceso público	Control de acceso	Protección
sklearn tip	30 oct 2022 10:21:53	Region	us-central1	Standard	30 oct 2022 10:21:53	No público	Uniforme	Ninguna

Pincharemos en el nombre *sklearn tip* y subiremos el archivo que hemos creado con formato **joblib**.

SUBIR ARCHIVOS

SUBIR CARPETA

CREAR CARPETA

TRANSFERIR LOS DATOS

ADMINISTRAR CONSERVACIONES

DESCARGAR

BORRAR

Filtrar solo por prefijo de nombre

Filtro

Filtrar objetos y carpetas

Mostrar datos borrados

Nombre	Tamaño	Tipo	Fecha de creación	Clase de almacenamiento	Última modificación	Acceso público	Historial de versiones	Encriptación	Fecha de ve
model.joblib	1 KB	application/octet-stream	30 oct 2022 10:23:27	Standard	30 oct 2022 10:2...	No público	—	Google-managed key	—

### Importar el modelo

En el menú buscamos el apartado de INTELIGENCIA ARTIFICIAL y seleccionamos **Vertex AI>Registro de modelos**. Tendremos que activar la **Vertex AI API** dando a activar e importaremos el modelo anteriormente subido al bucket.

Nombre	Estado de implementación	Descripción	Versión predeterminada	Tipo	Origen	Actualizado	Etiquetas
sklearntotaltip	Implementado en Vertex AI	—	1	Importado	Entrenamiento personalizado	30 oct 2022 10:30:35	—

### Desplegar modelo

En la anterior imagen, a la derecha del todo, podemos dar a los 3 puntos y elegir Implementar en el extremo. Seguimos los pasos y esperamos a que se implemente.

Una vez que se haya cargado nos aparecera con estado activo y podremos hacer una prueba de petición al modelo, como abajo en la imagen.

EVALUAR

IMPLEMENTA Y PRUEBA

PREDICCIÓN POR LOTES

DETALLES DE LA VERSIÓN

Deploy your model

Endpoints are machine learning models made available for online prediction requests. Endpoints are useful for timely predictions from many users (for example, in response to an application request). You can also request batch predictions if you don't need immediate results.

IMPLEMENTAR EN EL EXTREMO

Nombre	ID	Estado	Modelos	Región	Monitoring	Trabajo de supervisión más reciente	Alertas más recientes	Última actualización ↓	API	Notificación	Etiquetas ?	Encriptació
sklearntotaltipapi	1007473708438126592	Activo	1	us-central1	Inhabilitada	—	—	30 oct 2022 10:46:12	<a href="#">Solicitud de muestra</a>			Clave administra por Google

Prueba tu modelo

VISTA PREVIA

La solicitud JSON debe contener un campo `instancias` y un campo `parámetros` opcional si usas un contenedor personalizado. No puede haber otros campos en la solicitud JSON. [Obtén más información sobre cómo dar formato a tu solicitud JSON.](#)

Solicitud JSON

```
{  "instances": [[1.01,0,0,3,1,2,8.49,5.94]]}
```

PREDECIR

Respuesta

```
{  "predictions": [    17.14766613114104  ],  "deployedModelId": "2213646760201420800",  "model": "projects/90581667415/locations/us-central1/models/83224673934272",  "modelDisplayName": "sklearntotaltip",  "modelVersionId": "1"}
```

Vemos que nos da el mismo resultado que cuando probamos en nuestro código.

## Pruebas con el modelo

Para hacer las pruebas con python tenemos que hacer unos pasos de configuración, para poder acceder al modelo y autenticarnos.

Tendremos que instalar en nuestro entorno google-cloud-aiplatform.

```
pip install virtualenv
virtualenv <your-env>
<your-env>\Scripts\activate
<your-env>\Scripts\pip.exe install google-cloud-aiplatform
```

Tendremos que instalarnos el CLI de GoogleCloud para poder crear un archivo de login. Ponemos en la terminal directamente el siguiente código y seguimos los pasos de instalación.

```
(New-Object Net.WebClient).DownloadFile("https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe", "$env:Temp\GoogleCloudSDKInstaller.exe")

& $env:Temp\GoogleCloudSDKInstaller.exe
```

Creamos el archivo de login

```
gcloud auth application-default login
```

Una vez que hemos realizado estos pasos previos, creamos el programa en python.

Podemos crear esta función para poder comprobar los buckets y comprobar que estamos logeados.

```
from google.cloud import storage

def authenticate_implicit_with_adc(project_id="your id"):
    """
    When interacting with Google Cloud Client libraries, the library can auto-detect the
    credentials to use.

    // TODO(Developer):
    // 1. Before running this sample,
    // set up ADC as described in https://cloud.google.com/docs/authentication/external/set-up-adc
    // 2. Replace the project variable.
    // 3. Make sure that the user account or service account that you are using
    // has the required permissions. For this sample, you must have "storage.buckets.list".
    Args:
        project_id: The project id of your Google Cloud project.
    """

    # This snippet demonstrates how to list buckets.
    # *NOTE*: Replace the client created below with the client required for your application.
    # Note that the credentials are not specified when constructing the client.
    # Hence, the client library will look for credentials using ADC.
    storage_client = storage.Client(project=project_id)
    buckets = storage_client.list_buckets()
    print("Buckets:")
    for bucket in buckets:
        print(bucket.name)
    print("Listed all storage buckets.")

authenticate_implicit_with_adc(project_id="90581667415")
```

Ahora necesitamos la función que haga la llamada

```
from typing import Dict, List, Union

from google.cloud import aiplatform
from google.protobuf import json_format
from google.protobuf.struct_pb2 import Value

def predict_custom_trained_model_sample(
    project: str,
    endpoint_id: str,
    instances: Union[Dict, List[Dict]],
    location: str = "us-central1",
    api_endpoint: str = "us-central1-aiplatform.googleapis.com",
):
    """
    `instances` can be either single instance of type dict or a list
    of instances.
    """
    # The AI Platform services require regional API endpoints.
    client_options = {"api_endpoint": api_endpoint}
    # Initialize client that will be used to create and send requests.
    # This client only needs to be created once, and can be reused for multiple requests.
    client = aiplatform.gapic.PredictionServiceClient(client_options=client_options)
    # The format of each instance should conform to the deployed model's prediction input schema.
    instances = instances if type(instances) == list else [instances]
    instances = [
        json_format.ParseDict(instance_dict, Value()) for instance_dict in instances
    ]
    parameters_dict = {}
    parameters = json_format.ParseDict(parameters_dict, Value())
    endpoint = client.endpoint_path(
        project=project, location=location, endpoint=endpoint_id
    )
    response = client.predict(
        endpoint=endpoint, instances=instances
    )
    print("response")
    print(" deployed_model_id:", response.deployed_model_id)
    # The predictions are a google.protobuf.Value representation of the model's predictions.
    predictions = response.predictions
    for prediction in predictions:
        print(" prediction:", prediction)
```

Podemos ver el resultado una vez llamamos a la función.

```
1
2 predict_custom_trained_model_sample(
3     project="90581667415",
4     endpoint_id="1007473708438126592",
5     location="us-central1",
6     instances= [[1.01,0,0,3,1,2,8.49,5.94]]
7 )

response
  deployed_model_id: 2213646760201420800
  prediction: 17.14766613114104
```

Nos da el mismo resultado que anteriormente.