**1. Intento de Despliegue en Modal**

- \*\*Objetivo\*\*: Inicialmente, intentamos desplegar la aplicación FastAPI en Modal, un entorno serverless. El objetivo era que la aplicación pudiera procesar solicitudes del cliente utilizando un modelo de predicción basado en un archivo `random\_forest.pkl`.

- \*\*Problemas Encontrados\*\*:

- \*\*Error `modal-http: function not found`\*\*: A pesar de que el despliegue de la función en Modal parecía completarse con éxito, la aplicación no respondía correctamente a las solicitudes, generando un error `function not found`. Esto sugería problemas en cómo se configuraba la función o se manejaba el enrutamiento dentro de Modal.

- \*\*Advertencias sobre la API de Modal\*\*: Al probar diferentes configuraciones, encontramos que algunas API y decoradores de Modal no estaban disponibles en la versión que estábamos utilizando, lo que limitó las opciones de configuración.

- \*\*Diagnóstico\*\*:

- Determinamos que el problema probablemente estaba relacionado con cómo Modal gestionaba el entorno de ejecución y el montaje de archivos, específicamente con el archivo `random\_forest.pkl`.

- Decidimos cambiar de estrategia para probar si el problema era específico de Modal o si había algún problema subyacente en el código.

**2. Ejecución Local (Exitosa)**

- \*\*Estrategia Alternativa\*\*:

- Decidimos ejecutar la aplicación FastAPI localmente para verificar que el código y el modelo funcionaban correctamente fuera del entorno de Modal.

- \*\*Configuración Local\*\*:

- \*\*Estructura de Directorios\*\*: La aplicación `fastapilocal\_app.py` estaba ubicada en una carpeta llamada `app`, y el archivo `random\_forest.pkl` estaba en una carpeta llamada `model` al mismo nivel que `app`.

- \*\*Código\*\*: Ajustamos el código para que cargara el archivo `random\_forest.pkl` desde la ruta relativa `../model/random\_forest.pkl`.

- \*\*Pasos Seguidos\*\*:

1. \*\*Iniciamos el servidor FastAPI\*\* utilizando `uvicorn` desde el archivo `fastapilocal\_app.py`.

2. \*\*Probamos la API\*\*:

- Navegamos a `http://127.0.0.1:8000/` para verificar que la API estaba funcionando. Esto devolvió un mensaje de confirmación.

- Usamos `cURL` para enviar una solicitud POST al endpoint `/predict`, simulando datos de entrada para el modelo de predicción. La API respondió correctamente con una predicción (`{"predicted\_class": 0}`).

3. \*\*Revisamos advertencias menores\*\* como la deprecación de `@on\_event` en FastAPI, pero confirmamos que no afectaban el funcionamiento de la aplicación.

**Resultado:**

- La aplicación funcionó perfectamente en el entorno local, confirmando que el código y la configuración eran correctos y que el problema estaba limitado al entorno de Modal.

**Conclusión de la Etapa**

- \*\*Determinación del Problema\*\*: El éxito de la ejecución local sugirió que los problemas que enfrentábamos estaban relacionados con el entorno de Modal, posiblemente con la configuración de funciones y el manejo de archivos.

- \*\*Siguiente Paso\*\*: Decidimos proceder con un despliegue ajustado en Modal, ahora con la seguridad de que la base del código estaba funcionando correctamente.

---

Este resumen documenta cómo, tras intentar desplegar en Modal y encontrar problemas, validamos el funcionamiento del código en un entorno local. Esto nos permite avanzar con confianza en la siguiente fase del proyecto, ajustando la configuración para el despliegue en Modal.

<https://matiasbunsterraby--fastapi2-app-fastapi-app.modal.run>

<https://modal.com/apps/matiasbunsterraby/main/deployed/fastapi2_app>

**3. Desplegar en Modal**

### Documentación del Proceso de Despliegue de FastAPI en Modal

#### 1. \*\*Regeneración del Modelo\*\*

Para asegurar la compatibilidad entre el entorno local y el entorno de despliegue en Modal, se realizó la regeneración del modelo RandomForest usando la versión 1.3.2 de `scikit-learn`.

\*\*Pasos seguidos:\*\*

1. \*\*Verificación de Versiones\*\*:

- Aseguramos que `scikit-learn` en el entorno local estaba en la versión 1.3.2.

- Comando utilizado para verificar:

```bash

pip show scikit-learn

```

2. \*\*Entrenamiento y Guardado del Modelo\*\*:

- Entrenamos el modelo usando un DataFrame `df` con las características y la variable objetivo `target`.

- El modelo se entrenó y se guardó con el siguiente código:

```python

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

import joblib

X = df.drop(columns=['target'])

y = df['target']

model = RandomForestClassifier(random\_state=42, max\_depth=10)

model.fit(X, y)

joblib.dump(model, 'random\_forest.joblib')

```

3. \*\*Verificación de la Carga Local del Modelo\*\*:

- Se confirmó que el modelo se cargaba correctamente en el entorno local utilizando el siguiente código:

```python

import joblib

MODEL\_PATH = '/ruta/a/tu/modelo/random\_forest.joblib'

try:

with open(MODEL\_PATH, 'rb') as f:

rfc = joblib.load(f)

print("Modelo cargado exitosamente")

print(f"Parámetros del modelo: {rfc.get\_params()}")

except Exception as e:

print(f"Error al cargar el modelo: {e}")

```

#### 2. \*\*Configuración y Despliegue en Modal\*\*

\*\*1. Configuración de Modal:\*\*

- Se configuró la aplicación FastAPI para cargar el modelo dentro del endpoint `/predict` y se desplegó en Modal.

- Se utilizó la imagen de `debian\_slim` con Python 3.9 y las librerías necesarias (`fastapi`, `pydantic`, `joblib`, `numpy`, `uvicorn`, `scikit-learn==1.3.2`).

\*\*Código usado para la configuración:\*\*

```python

import modal

from fastapi import FastAPI

from pydantic import BaseModel

import numpy as np

import joblib

from app.predict import predecir\_viaje\_taxi

web\_app = FastAPI()

MODEL\_PATH = '/model/random\_forest.joblib'

imagen = modal.Image.debian\_slim(python\_version="3.9").pip\_install(

'fastapi', 'pydantic', 'joblib', 'numpy', 'uvicorn', 'scikit-learn==1.3.2'

).copy\_local\_file(

'/ruta/a/tu/modelo/random\_forest.joblib',

'/model/random\_forest.joblib'

)

app = modal.App('Prediccion-Propinas-Taxi')

# Definir la clase para los datos de entrada

class Item(BaseModel):

pickup\_weekday: float

pickup\_hour: float

work\_hours: float

pickup\_minute: float

passenger\_count: float

trip\_distance: float

trip\_time: float

trip\_speed: float

PULocationID: float

DOLocationID: float

RatecodeID: float

@web\_app.post('/predict')

async def manejar\_prediccion(item: Item, confianza: float = 0.5):

try:

with open(MODEL\_PATH, 'rb') as f:

rfc = joblib.load(f)

caracteristicas\_viaje = np.array([

item.pickup\_weekday, item.pickup\_hour, item.work\_hours, item.pickup\_minute, item.passenger\_count,

item.trip\_distance, item.trip\_time, item.trip\_speed, item.PULocationID, item.DOLocationID,

item.RatecodeID

])

pred = predecir\_viaje\_taxi(caracteristicas\_viaje, '/model/random\_forest.joblib', confianza)

return {'clase\_predicha': pred}

except Exception as e:

return {"error": str(e)}

@app.function(image=imagen)

@modal.asgi\_app()

def fastapi\_app():

return web\_app

if \_\_name\_\_ == '\_\_main\_\_':

app.deploy('Prediccion-Propinas-Taxi')

```

\*\*2. Despliegue:\*\*

- Se desplegó la aplicación con:

```bash

modal deploy app/fastapi2\_app.py

```

\*\*3. Pruebas del Endpoint `/predict`:\*\*

- Se utilizó el siguiente comando `curl` para probar el endpoint `/predict` y verificar que el modelo funcionara correctamente:

```bash

curl -X POST "https://matiasbunsterraby--prediccion-propinas-taxi-fastapi-app.modal.run/predict" \

-H "Content-Type: application/json" \

-d '{

"pickup\_weekday": 3,

"pickup\_hour": 14,

"work\_hours": 1,

"pickup\_minute": 30,

"passenger\_count": 2,

"trip\_distance": 5.5,

"trip\_time": 15.5,

"trip\_speed": 22.0,

"PULocationID": 130,

"DOLocationID": 150,

"RatecodeID": 1

}'

```

\*\*Resultado:\*\* Se obtuvo una respuesta exitosa con la predicción esperada.

### Consideraciones Finales

Este proceso asegura que el modelo se cargue correctamente en Modal y que el endpoint `/predict` funcione según lo esperado. Con esta base, se puede avanzar a la siguiente etapa, que consiste en la creación de una interfaz en Streamlit para interactuar con este backend. ¡Todo está listo para continuar!

**4.STREAMLIT local**

**“hemos confirmado que tanto el backend en FastAPI como la interfaz de usuario en Streamlit funcionan correctamente”**

Aquí tienes una documentación organizada en un formato adecuado para un archivo Word. La redacción se ajusta para ser clara y concisa, manteniendo todo dentro de un límite de 4 páginas.

---

## \*\*Documentación del Proyecto: Predicción de Propinas para Taxis en NYC\*\*

### \*\*1. Resumen del Proyecto\*\*

El objetivo de este proyecto es desarrollar una aplicación que prediga si un pasajero dejará una propina alta en un viaje en taxi en la ciudad de Nueva York, basada en ciertas características del viaje. El proyecto consiste en un backend desarrollado en FastAPI, desplegado en Modal, y una interfaz de usuario interactiva creada con Streamlit, ejecutada localmente.

### \*\*2. Configuración del Entorno\*\*

#### \*\*2.1. Requerimientos\*\*

- \*\*Python\*\*: 3.9

- \*\*Bibliotecas Python\*\*:

- `fastapi`

- `pydantic`

- `uvicorn`

- `joblib`

- `numpy`

- `streamlit`

- `scikit-learn==1.3.2`

- \*\*Modal CLI\*\*: Instalado y configurado para despliegues en la nube.

#### \*\*2.2. Instalación de Dependencias\*\*

Se requiere un entorno virtual para gestionar las dependencias del proyecto. A continuación, se muestra cómo configurar el entorno:

```bash

python -m venv venv

source venv/bin/activate # En Windows, usar venv\Scripts\activate

pip install -r requirements.txt

```

### \*\*3. Desarrollo del Backend con FastAPI\*\*

#### \*\*3.1. Descripción\*\*

El backend de la aplicación se desarrolló en FastAPI, encargado de recibir solicitudes de predicción y procesarlas utilizando un modelo de RandomForest entrenado.

#### \*\*3.2. Estructura del Código\*\*

```python

import modal

from fastapi import FastAPI

from pydantic import BaseModel

import numpy as np

import joblib

import os

from app.predict import predecir\_viaje\_taxi

web\_app = FastAPI()

MODEL\_PATH = '/model/random\_forest.joblib'

print(f"Ruta esperada del modelo dentro del contenedor: {MODEL\_PATH}")

imagen = modal.Image.debian\_slim(python\_version="3.9").pip\_install(

'fastapi', 'pydantic', 'joblib', 'numpy', 'uvicorn', 'scikit-learn==1.3.2'

).copy\_local\_file(

'/Users/matiasbunsterrabyhomefolder/Library/CloudStorage/OneDrive-Personal/MB\_Magister/DATA\_SCIENCE\_UDD/Tarea\_Opcional\_3/model/random\_forest.joblib',

'/model/random\_forest.joblib'

)

app = modal.App('Prediccion-Propinas-Taxi')

# Definir la clase para los datos de entrada

class Item(BaseModel):

pickup\_weekday: float

pickup\_hour: float

work\_hours: float

pickup\_minute: float

passenger\_count: float

trip\_distance: float

trip\_time: float

trip\_speed: float

PULocationID: float

DOLocationID: float

RatecodeID: float

@web\_app.get('/ping')

async def ping():

return 'pong'

@web\_app.post('/predict')

async def manejar\_prediccion(item: Item, confianza: float = 0.5):

try:

print("Intentando cargar el modelo dentro del endpoint...")

with open(MODEL\_PATH, 'rb') as f:

rfc = joblib.load(f)

print("Modelo cargado exitosamente dentro del endpoint.")

caracteristicas\_viaje = np.array([

item.pickup\_weekday, item.pickup\_hour, item.work\_hours, item.pickup\_minute, item.passenger\_count,

item.trip\_distance, item.trip\_time, item.trip\_speed, item.PULocationID, item.DOLocationID,

item.RatecodeID

])

pred = predecir\_viaje\_taxi(caracteristicas\_viaje, '/model/random\_forest.joblib', confianza)

return {'clase\_predicha': pred}

except Exception as e:

print(f"Error al cargar el modelo o durante la predicción: {e}")

return {"error": str(e)}

@app.function(image=imagen)

@modal.asgi\_app()

def fastapi\_app():

return web\_app

if \_\_name\_\_ == '\_\_main\_\_':

app.deploy('Prediccion-Propinas-Taxi')

```

### \*\*4. Despliegue en Modal\*\*

#### \*\*4.1. Proceso de Despliegue\*\*

- \*\*Paso 1\*\*: Despliega el backend FastAPI en Modal utilizando el siguiente comando:

```bash

modal deploy app/fastapi2\_app.py

```

- \*\*Paso 2\*\*: Verifica el estado del backend asegurándote de que el endpoint `/ping` responde correctamente. Usa el comando `curl` para esta prueba:

```bash

curl -X GET "https://<tu-url-modal>/ping"

```

- \*\*Paso 3\*\*: Una vez verificado que el backend funciona correctamente, procede a la ejecución y prueba de la interfaz de usuario en Streamlit.

### \*\*5. Desarrollo de la Interfaz con Streamlit\*\*

#### \*\*5.1. Ejecución Local de Streamlit\*\*

Para la interfaz de usuario, se desarrolló una aplicación en Streamlit que interactúa con el backend en Modal para realizar predicciones.

- \*\*Paso 1\*\*: Ejecuta Streamlit localmente:

```bash

streamlit run app/st\_taxi\_app.py

```

- \*\*Paso 2\*\*: Asegúrate de que la URL de FastAPI en Modal esté correctamente configurada en el código de Streamlit.

- \*\*Paso 3\*\*: Realiza pruebas ingresando diferentes valores para las variables del viaje en taxi y verifica las predicciones.

### \*\*6. Resultados y Conclusiones\*\*

Este proyecto demuestra cómo se puede integrar un backend desarrollado en FastAPI con una interfaz de usuario en Streamlit, utilizando Modal para el despliegue del backend en un entorno serverless. La configuración y despliegue han permitido predecir de manera eficiente las propinas para viajes en taxi en la ciudad de Nueva York.

### \*\*7. Próximos Pasos\*\*

1. \*\*Optimización del Modelo\*\*: Mejorar la precisión del modelo RandomForest con un reentrenamiento utilizando más datos.

2. \*\*Despliegue Completo en la Nube\*\*: Considerar el despliegue completo de la aplicación Streamlit junto con FastAPI en un entorno cloud como Heroku para una solución completamente en la nube.

3. \*\*Mejoras en la Interfaz\*\*: Implementar mejoras en la interfaz de usuario para facilitar la entrada de datos y mejorar la experiencia del usuario.

---

Este documento debería ser claro y conciso, proporcionando una guía comprensible del proyecto y los pasos realizados. Puedes copiar y pegar este contenido en tu archivo Word y formatearlo según tus preferencias.

Si necesitas hacer ajustes o agregar más detalles, estaré aquí para ayudarte. ¡Buen trabajo hasta ahora!

**5.Modal, FastApi y Streamlit**