Actividad Práctica: Machine Learning como Servicio

Resumen

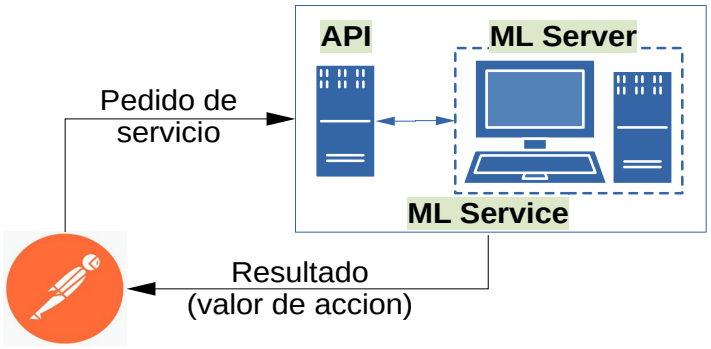
A modo de introducir al estudiante en el abordaje de la implementación de modelos predictivos, se propone en esta actividad la puesta en funcionamiento de una prueba de concepto mínima de un sistema consistente en un Servicio de Machine Learning que permite predecir el valor de la acción de YPF de mañana usando tres algoritmos regresores distintos: regresión lineal, polinomial y Support Vector Machines.

Objetivos

- Que el alumno tenga un primer abordaje a la implementación de modelos predictivos.
- Que el alumno realice su propio despliegue de un sistema virtualizado con contenedores.

Desarrollo de la Actividad

Arquitectura General



Se requiere tener instalado Docker Desktop, Visual Studio Code con la extensión Docker y (opcional) Postman (para interactuar con la API).

Preparar estructura de carpetas

```
/ml_service
├── Dockerfile
└── src/
    ├── mlscript.py
    └── data/ # Carpeta compartida
```

Crear python script

Dentro de la carpeta “src” crear el script Python que ejecutará los servicios de Machine Learning e interactuará con la API. Para ello: abrir un editor de textos, crear un archivo nuevo y llamarlo “mlscript.py”. Copiar y pegar el siguiente texto dentro del script:

Python Code:

```
from fastapi import FastAPI
import yfinance as yf
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import SVR

app = FastAPI()

# Función que realiza las predicciones
def predecir_valor_cierre():
    # Aquí incluyes tu código de obtención de datos y modelos de predicción
    # Por ejemplo:
    data = yf.download('YPF', start="2023-01-01", end="2024-07-31")
    precios = data['Close'].values.reshape(-1, 1)

    # Regresión Polinomial
    poly = PolynomialFeatures(degree=2)
    X_poly = poly.fit_transform(np.arange(len(precios)).reshape(-1, 1))
    poly_reg_model = LinearRegression().fit(X_poly, precios)
      pred_polynomial = poly_reg_model.predict(poly.transform([[len(precios)]]))
[0][0]

    # Regresión Lineal Múltiple
      lin_reg_model = LinearRegression().fit(np.arange(len(precios)).reshape(-1,
1), precios)
    pred_linear = lin_reg_model.predict([[len(precios)]])[0][0]

    # SVR
    svr_model = SVR(kernel='rbf')
    svr_model.fit(np.arange(len(precios)).reshape(-1, 1), precios.ravel())
    pred_svr = svr_model.predict([[len(precios)]])[0]

    return {
        'Regresión Polinomial': np.float64(pred_polynomial),
        'Regresión Lineal': np.float64(pred_linear),
        'SVR': np.float64(pred_svr)
    }

@app.get("/predecir")
def predecir():
    prediccion = predecir_valor_cierre()
    return prediccion
```

Guardar y cerrar

Preparar contenedor Docker

Abrir un editor de textos, crear un archivo nuevo y llamarlo Dockerfile. Guardarlo dentro de la carpeta “ml_service”. Copiar y pegar el siguiente texto dentro del Dockerfile:

Dockerfile
```
# Usa la imagen base de FastAPI con Python 3.9
FROM tiangolo/uvicorn-gunicorn-fastapi:python3.9

# Establece el directorio de trabajo en el contenedor
WORKDIR /app

# Copia solo los archivos de la carpeta src al contenedor
COPY ./src /app

# Instala las dependencias necesarias
RUN pip install --no-cache-dir yfinance pandas scikit-learn

# Crear un directorio para almacenar y compartir información
VOLUME /app/data

# Comando para iniciar la aplicación
CMD ["uvicorn", "mlscript:app", "--host", "0.0.0.0", "--port", "80"]
```

Guardar y cerrar.

Descargar Imagen y crear contenedor

Se descargará una imagen linux básica con fast-api y python 3.9. Abrir un terminal (o Powershell en Windows). Posicionarse en la carpeta “ml_service” (donde se encuentra el archivo Dockerfile). Copiar y pegar. No olvidarse el “.” que aparece al final.

Código Terminal/PS:
```
docker build -t ml_service .
```

El proceso se realiza una única vez. Dependiendo de las características de su equipo, tardará menos de un minuto.

Iniciar contenedor

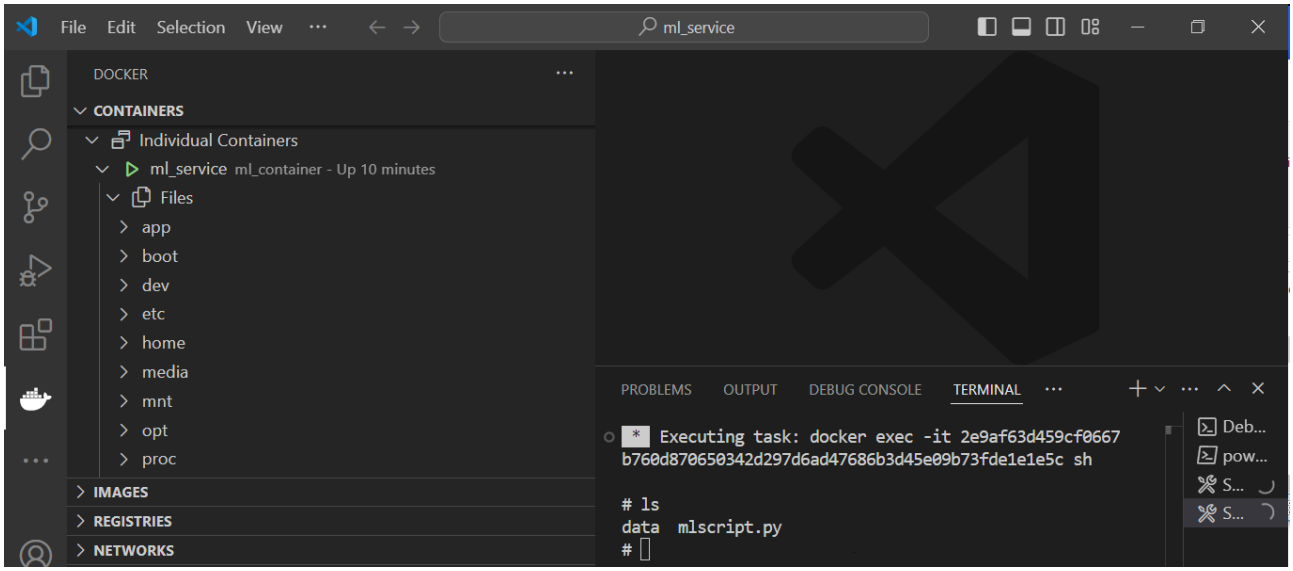
Luego de instalarse la imagen del contenedor localmente, iniciar una instancia de esa imagen. Para ello, se ejecuta el siguiente comando en el terminal.

Código Terminal/PS:
```
docker run -d --name ml_container -p 8000:80 -v "$(pwd)/src:/app" ml_service
```

Ver el contenedor encendido

En Visual Studio Code, abrir la carpeta “ml_servce”. Aparecerán los archivos que contiene esa carpeta.

En la parte de “extensiones”, elegir “Docker” y se podrá observar el contenedor iniciado (triangulo en verde). Puede observarse que se despliegan los archivos dentro del contenedor, en una distribución Linux.
Si se hace click con el botón derecho sobre el contenedor “ ▶ ml_service ml_container” y elegir “Attach Shell”, se abrirá un terminal dentro del contenedor Linux. Puede observarse la estructura de archivos dentro del contenedor.



Ejecutar la petición:

Usando Terminal y Curl:

Abriremos otra terminal desde la cual realizaremos una petición vía “Get” al puerto :80 de Localhost con el parámetro “/predecir”. Para ello:

```
Comando Terminal / PS:
curl http://localhost:8000/predecir
```

Respuesta

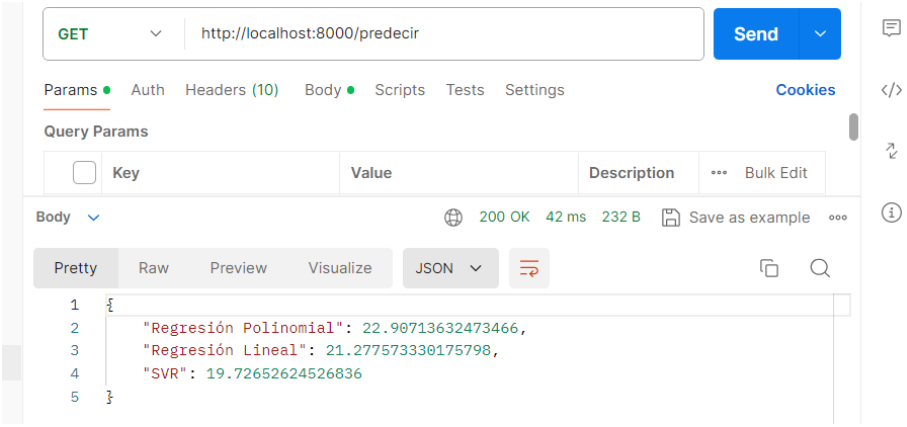
```
Salida del Terminal:
StatusCode        : 200
StatusDescription : OK
Content           : {"RegresiÃ³n Polinomial":22.90713632473466,"RegresiÃ³n
                    Lineal":21.277573330175798,"SVR":19.72652624526836}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 106
                    Content-Type: application/json
                    Date: Mon, 19 Aug 2024 00:01:26 GMT
                    Server: uvicorn

                          {"RegresiÃ³n Polinomial":22.90713632473466,"RegresiÃ³n
Lineal":21.27757333...
Forms             : {}
Headers             : {[Content-Length, 106], [Content-Type, application/json],
[Date, Mon, 19 Aug 2024 00:01:26 GMT],
```

```
                    [Server, uvicorn]}
Images              : {}
InputFields         : {}
Links               : {}
ParsedHtml          : System.__ComObject
RawContentLength    : 106
```

Usando Postman:

Postman es una herramienta para supervisar conexiones con APIs. Utilizaremos Postman para realizar la petición. Para ello: Abrimos Postman. Dentro del método se elige “Get”, se coloca la url: “http://localhost:8000/predecir“ y presionar “SEND”.

Respuesta



Conclusion:

En este demo se presentó una POC de implementación de un sistema consistente en un Servicio de Predicción del valor de acción de YPF usando tres algoritmos diferentes de Machine Learning. El sistema se compone de una API y de un servidor de Machine Learning al que se accede desde una url. El sistema responde con el valor de la acción (en json).
Se rescata también, lo simple y rápido que se puede implementar un sistema virtualizado en contenedores Docker.