

MÓDULO 6 - SESIÓN 5: DESPLIEGUE DE MODELOS

Desplegar un modelo de machine learning significa llevarlo desde un entorno de experimentación (notebooks, pruebas locales) a un entorno donde pueda ser utilizado por otros sistemas o usuarios finales.

R por Kibernetum Capacitación S.A. Kibernetum Capacitación S.A.



Preparación del modelo para producción

Desplegar un modelo de machine learning significa llevarlo desde un entorno de experimentación (notebooks, pruebas locales) a un entorno donde pueda ser utilizado por otros sistemas o usuarios finales.

Pasos clave:

- Validar el modelo con datos nuevos (test set)
- Optimizar hiperparámetros
- Documentar las dependencias y versiones
- Crear una API o interfaz de acceso
- Monitorear el comportamiento en producción

Revisión de pasos clave para el despliegue

Paso	Descripción
Validación	Evaluar el modelo con datos reales no vistos
Empaquetado	Guardar el modelo (ej. con joblib, pickle)
Versionamiento	Registrar versiones del modelo para trazabilidad
Documentación	Describir las variables, métricas y contexto del modelo
Seguridad	Asegurar el acceso a la API con autenticación y monitoreo

Integración en un sistema de producción

Hay muchas formas de integrar un modelo en una aplicación real:

- Crear una API REST utilizando frameworks como Flask o FastAPI.
- Contenerizar el modelo en un microservicio con Docker.
- Desplegarlo en plataformas como Heroku, AWS SageMaker o Google Vertex AI.

Métricas para modelos de regresión

Durante el curso aprendiste a usar varias métricas de evaluación. Aquí las repasamos:

Métrica	Fórmula	Interpretación
MAE (Error Absoluto Medio)	$MAE = (1/n) \sum y_i - \hat{y}_i $	Promedio de errores absolutos
MSE (Error Cuadrático Medio)	$MSE = (1/n) \sum (y_i - \hat{y}_i)^2$	Penaliza errores grandes
RMSE (Raíz del MSE)	$RMSE = \sqrt{(1/n) \sum (y_i - \hat{y}_i)^2}$	Misma unidad que la variable objetivo
R^2 (Coef. de determinación)	$R^2 = 1 - (SS_{res}/SS_{tot})$	Cuánta varianza explica el modelo

Métricas para clasificación

Métrica	Fórmula	Uso recomendado
Precisión	$TP/(TP+FP)$	Cuando importa que los positivos predichos sean correctos
Sensibilidad (Recall)	$TP/(TP+FN)$	Cuando es crítico encontrar todos los positivos reales
Especificidad	$TN/(TN+FP)$	Cuando se debe evitar falsos positivos
ROC-AUC	Área bajo la curva ROC	Comparación entre clasificadores binarios

Caso práctico final – Proyecto de despliegue real

Contexto: revisión aplicada de contenidos

Durante el módulo, aprendiste a:

- Preprocesar datos y seleccionar variables.
- Entrenar y ajustar modelos de regresión y clasificación.
- Evaluar con métricas robustas (MAE, RMSE, R^2 , precisión, recall, AUC).
- Validar modelos usando técnicas como K-Fold y LOOCV.
- Simular escenarios reales como diagnóstico médico, predicción de precios, y detección de fraude.

Ahora, integraremos todo lo aprendido en un proyecto funcional que simula el despliegue de un modelo en producción.

Actividad práctica guiada – Parte 1

Título: Despliegue de un modelo de predicción de arriendos con FastAPI

En este ejercicio, se implementa un modelo de regresión lineal para predecir precios de arriendo de propiedades en Chile, integrándolo como una API REST con FastAPI. Este ejemplo aplica los conocimientos de regresión, métricas y despliegue.

Respositorio del proyecto

Enlace: [Acá](#)

PASO 1: ENTRENAR EL MODELO EN GOOGLE COLAB

- Dataset: arriendos_chile_simulado.csv
- Algoritmo: LinearRegression con Scikit-learn
- Guardar el modelo entrenado

PASO 2: CREAR UNA API CON FASTAPI

Implementar una API que cargue el modelo y permita hacer predicciones

PASO 3: EJECUTAR LOCALMENTE

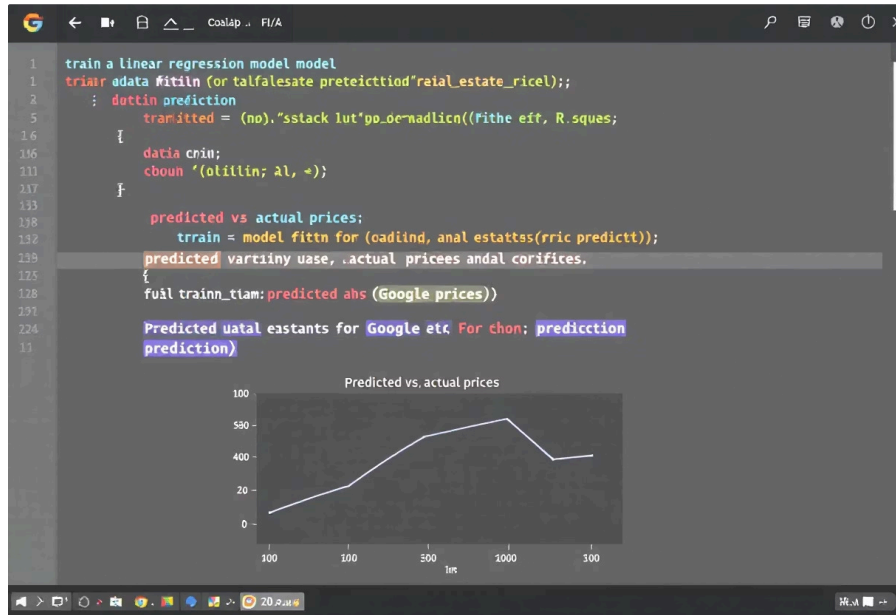
Seguir el paso a paso del repositorio para ejecutar el servidor

PASO 4: PROBAR EN NAVEGADOR

Utilizar la interfaz Swagger UI para realizar pruebas

Entrenamiento del modelo en Google Colab

Este archivo .pkl guarda el modelo ya entrenado, incluyendo sus parámetros, y permite reutilizarlo sin volver a entrenar, ejecuta el Google Colab y observa lo que sucede celda a celda.



Código para entrenar y guardar el modelo

El proceso incluye:

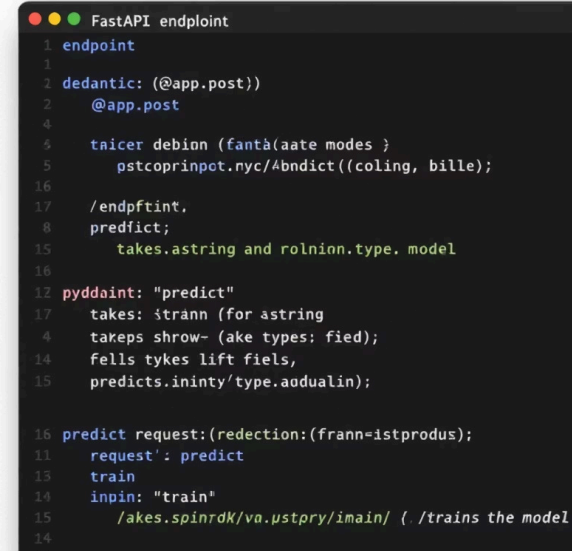
- Cargar el dataset de arriendos
- Preprocesar las variables
- Entrenar un modelo de regresión lineal
- Evaluar su rendimiento con métricas
- Guardar el modelo entrenado como archivo .pkl

El modelo entrenado se guarda en un archivo .pkl que luego será utilizado por la API para realizar predicciones sin necesidad de volver a entrenar.

Creación de la API con FastAPI

Estructura del código

- Importar bibliotecas necesarias
- Definir modelos de datos con Pydantic
- Cargar el modelo entrenado
- Crear endpoints para predicciones
- Implementar validación de datos

A screenshot of a code editor window titled "FastAPI endpoint". The code defines a FastAPI application with two endpoints. The first endpoint, decorated with @app.post, is a POST endpoint that takes a request object and returns a response. The second endpoint, decorated with @app.get, is a GET endpoint that takes a request object and returns a response. The code uses Pydantic for data validation and FastAPI for the API framework.

```
1 endpoint
2
3 dedantic: (@app.post))
4 @app.post
5
6 taicer debion (fantà(aate modes )
7   pstcoprinpot.nyc/4bndict((coling, bille);
8
9 /endpftint,
10 predict;
11 takes.astring and rolnton.type. model
12
13 pyddaint: "predict"
14 takes: itrann (for astring
15 takesps shrow~ (ake types: fied);
16 fells tykes lift fiels,
17 predicts.ininty/type.aadualin);
18
19 predict request:(redection:(frann=istprodus);
20 request': predict
21 train
22 inpin: "train"
23 /akes.spinrdK/va.ustpry/imaing ( /trains the model
24
```

Contexto realista del CSV:

El archivo arriendos_chile_simulado.csv representa los datos históricos de propiedades arrendadas en Chile. En un entorno profesional, estos datos provendrían de sistemas como SAP, bases de datos transaccionales, o un Data Warehouse corporativo.

En este ejercicio:

- El CSV simula la fuente autorizada de entrenamiento del modelo.
- Se utiliza en Colab para generar el archivo .pkl.
- Luego, el modelo guardado se reutiliza en la API para predecir nuevos valores sin necesidad de volver a entrenar.

Este enfoque refleja un flujo profesional, donde el entrenamiento y el despliegue están desacoplados, tal como ocurre en producción.

Ejecución y prueba de la API

PASO 3: EJECUTAR LOCALMENTE

Sigue el paso a paso indicado en el repositorio, puedes descargar el ZIP sugerido desde aca: Enlace: Descarga el [archivo ZIP](#) con la estructura del proyecto desde aquí, o si prefieres, accede directamente al repositorio completo.

Esto ejecuta el servidor en localhost con recarga automática.

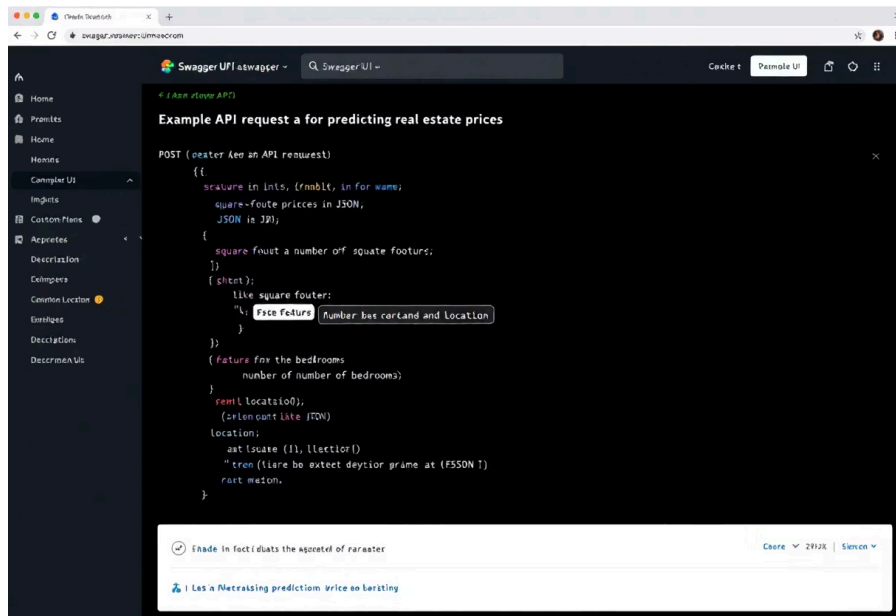
PASO 4: PROBAR EN NAVEGADOR

Abre tu navegador en: <http://localhost:8000/docs>

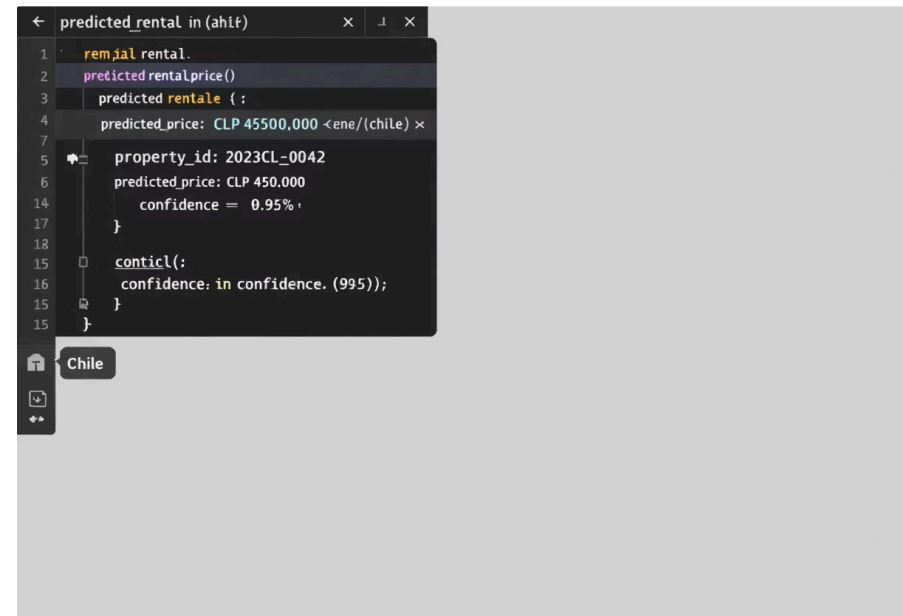
Allí encontrarás una interfaz interactiva (Swagger UI) donde puedes probar la API de forma visual.

RESULTADO ESPERADO

Al hacer una consulta POST con estos datos:



La respuesta será algo como:



Esta respuesta es generada por el modelo entrenado, ahora expuesto como un servicio de predicción.