Manual de Instalación de Optuna en Google Colab

Edilfonso Muñoz Anccori

February 13, 2025

1 Introducción

Optuna es una biblioteca de optimización automática de hiperparámetros utilizada en machine learning. Su objetivo es proporcionar una forma eficiente y flexible de optimizar funciones de costo en diversas aplicaciones, especialmente en aprendizaje automático y redes neuronales. Optuna permite la búsqueda automática de hiperparámetros mediante técnicas avanzadas como búsqueda bayesiana, pruning y ensayos paralelos, lo que la convierte en una herramienta poderosa para la optimización de modelos. Su integración con frameworks populares como TensorFlow, PyTorch y Scikit-Learn facilita su uso en entornos de investigación y producción.

Este manual explica cómo instalar y configurar Optuna en Google Colab.

2 Requisitos Previos

Para seguir este manual, es necesario:

- Tener una cuenta en Google.
- Acceder a Google Colab.
- Conexión a Internet.

3 Pasos de Instalación

3.1 Paso 1: Abrir Google Colab

1. Inicie sesión en su cuenta de Google. 2. Acceda a Google Colab. 3. Cree un nuevo cuaderno (File → New Notebook).

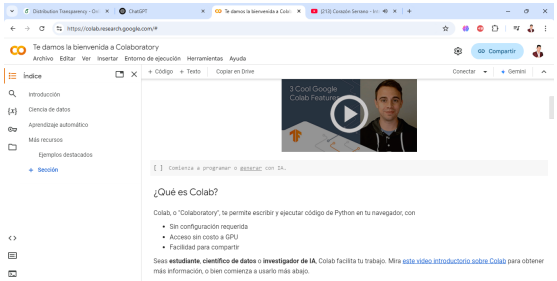


Figure 1: Cuenta de Colab

3.2 Paso 2: Instalar Optuna

En una celda de código, ejecute el siguiente comando para instalar Optuna:

```
!pip install optuna
!pip install pandas openpyxl
!pip install matplotlib-venn
!apt-get -qq install -y libfluidsynth1
!apt-get -qq install -y libarchive-dev && pip install -U libarchive
import libarchive
!apt-get -qq install -y graphviz && pip install pydot
import pydot
!pip install cartopy
import cartopy
```

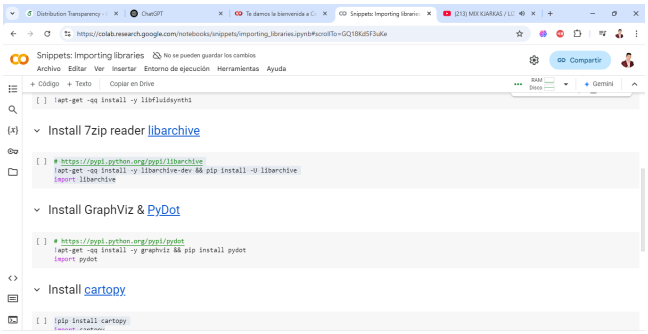


Figure 2: Librerías para Instalar

Esto descargará e instalará la biblioteca Optuna y sus dependencias.

3.3 Paso 3: Verificar la Instalación

Para comprobar que Optuna se instaló correctamente, ejecute:

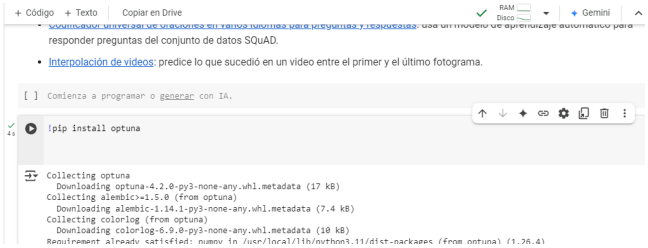


Figure 3: Instalar Optuna

```
import optuna
print(optuna.__version__)
```

Si no se muestra ningún error y aparece la versión de Optuna, la instalación fue exitosa.

Figure 4: Verificar la Instalación

4 Ejemplo de Uso

Para probar Optuna, ejecute el siguiente código en una celda de Google Colab:

```
def objective(trial):
    x = trial.suggest_float("x", -10, 10)
    return (x - 2) ** 2

study = optuna.create_study()
study.optimize(objective, n_trials=100)

print("Mejor resultado:", study.best_value)
print("Mejor parámetro:", study.best_params)
```

Código en Python para Procesamiento de Datos y Optimización

El siguiente código carga un archivo Excel, lo procesa y optimiza un modelo de regresión usando Optuna:

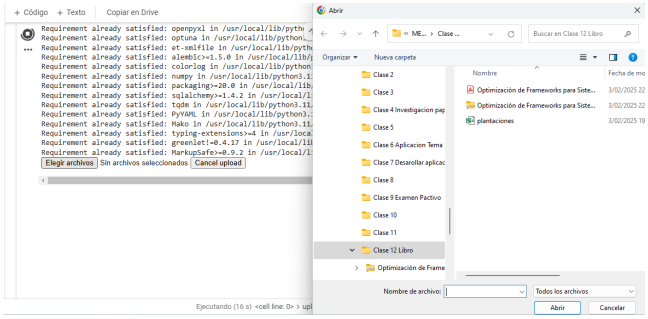


Figure 5: Subir Archivo

```python
# Instalar librer as necesarias
!pip install openpyxl optuna

# Importar librer as
import pandas as pd
import optuna
from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

# Subir el archivo
uploaded = files.upload()

# Obtener el nombre del archivo subido
file_name = next(iter(uploaded.keys()))

# Leer el archivo Excel correctamente
try:
    df = pd.read_excel(file_name, engine='openpyxl')  # Forzar openpyxl
    print("Archivo cargado correctamente:")
    print(df.head())
except Exception as e:
    print(f"Error al leer el archivo: {e}")

# Verificar valores nulos
df.fillna(df.mean(numeric_only=True), inplace=True)

# Codificar variables categ ricas
label_encoders = {}
for col in df.select_dtypes(include=['object']).columns:
    label_encoders[col] = LabelEncoder()
    df[col] = label_encoders[col].fit_transform(df[col])

# Separar en variables predictoras y objetivo
X = df.drop(columns=['SUPERFICIE_PLANTACION'])
y = df['SUPERFICIE_PLANTACION']

```

```python
# Divisi n en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Definir funci n de optimizaci n con Optuna
def objective(trial):
    n_estimators = trial.suggest_int('n_estimators', 50, 200)
    max_depth = trial.suggest_int('max_depth', 5, 20)

    model = RandomForestRegressor(n_estimators=n_estimators, max_depth=max_depth, random_state=42)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    return mean_absolute_error(y_test, y_pred)

# Ejecutar la optimizaci n
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=20)

# Mostrar mejores par metros
print("\nMejores par metros:", study.best_params)

# Entrenar modelo final con mejores par metros
best_model = RandomForestRegressor(**study.best_params, random_state=42)
best_model.fit(X_train, y_train)

# Evaluar el modelo
y_pred = best_model.predict(X_test)
print("\nError final (MAE):", mean_absolute_error(y_test, y_pred))
```

5 Resultado

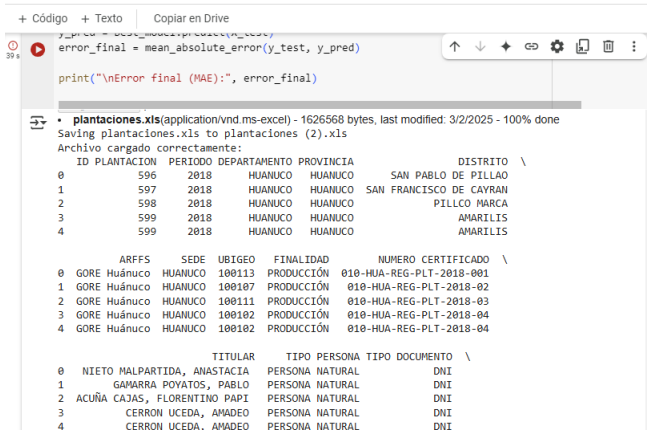
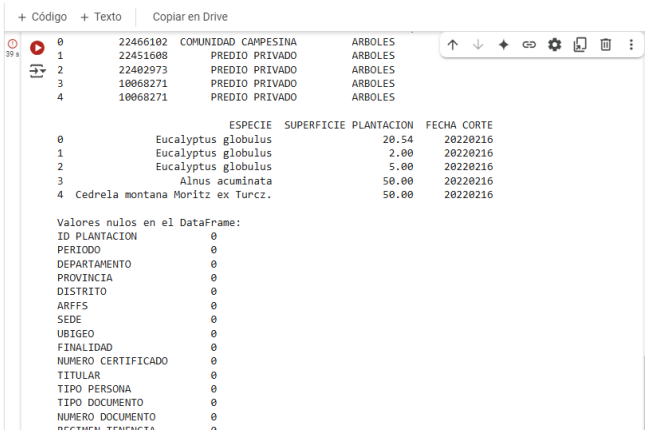


Figure 6: Subir Archivo



```
+ Código  + Texto  Copiar en Drive

0        22466102  COMUNIDAD CAMPESINA        ARBOLES
1        22451608       PREDIO PRIVADO        ARBOLES
2        22402973       PREDIO PRIVADO        ARBOLES
3        10068271       PREDIO PRIVADO        ARBOLES
4        10068271       PREDIO PRIVADO        ARBOLES

                            ESPECIE  SUPERFICIE PLANTACION  FECHA CORTE
0              Eucalyptus globulus                  20.54     20220216
1              Eucalyptus globulus                   2.00     20220216
2              Eucalyptus globulus                   5.00     20220216
3                  Alnus acuminata                  50.00     20220216
4  Cedrela montana Moritz ex Turcz.                 50.00     20220216

Valores nulos en el DataFrame:
ID PLANTACION            0
PERIODO                  0
DEPARTAMENTO             0
PROVINCIA                0
DISTRITO                 0
ARFFS                    0
SEDE                     0
UBIGEO                   0
FINALIDAD                0
NUMERO CERTIFICADO       0
TITULAR                  0
TIPO PERSONA             0
TIPO DOCUMENTO           0
NUMERO DOCUMENTO         0
```

Figure 7: Subir Archivo

6 Conclusión

Ahora está listo para usar Optuna en sus proyectos de optimización de hiperparámetros. Optuna es una herramienta poderosa para la optimización automática de hiperparámetros, facilitando la búsqueda de configuraciones óptimas en modelos de machine learning de manera eficiente. Su capacidad para implementar estrategias avanzadas como la búsqueda bayesiana, pruning dinámico y paralelización lo convierte en una opción ideal para investigadores y profesionales del área.

En este manual, hemos aprendido a instalar y configurar Optuna en Google Colab, además de verificar su funcionamiento con un ejemplo práctico. La optimización de modelos es un proceso clave en machine learning, ya que permite mejorar el rendimiento sin necesidad de ajustes manuales extensivos. Con Optuna, este proceso se vuelve más rápido y efectivo, permitiendo a los usuarios centrarse en la interpretación de resultados y la mejora de sus modelos.

El uso de técnicas automatizadas de optimización no solo ahorra tiempo, sino que también ayuda a descubrir combinaciones de hiperparámetros que podrían ser difíciles de encontrar mediante métodos tradicionales. Gracias a su integración con bibliotecas populares como Scikit-Learn, TensorFlow y PyTorch, Optuna se convierte en una herramienta esencial en el desarrollo de modelos predictivos robustos y precisos