1. ¿Cómo se llama por convención el archivo de requisitos?

   En el ecosistema pip/Python, el nombre convencional es requirements.txt. Es un archivo de texto plano que lista dependencias y (opcionalmente) versiones/pines para reconstruir el entorno con pip install -r requirements.txt. Variantes comunes: requirements-dev.txt (deps de desarrollo), requirements-prod.txt (producción), constraints.txt (congelar versiones sin instalar por sí mismo) y requirements.in cuando se usan herramientas de resolución como pip-tools. Para paquetes instalables, las dependencias "de runtime" viven en pyproject.toml (o antes setup.cfg/setup.py), mientras que requirements.txt captura el entorno de trabajo/ejecución reproducible.

2. ¿Se puede definir un archivo de requisitos para Conda?

   Sí. En Conda el análogo se define como environment.yml (YAML). A diferencia de requirements.txt, environment.yml permite declarar el nombre del entorno, canales (p. ej. conda-forge), paquetes no-Python y versiones; además puede incluir una sección pip: para mezclar dependencias de pip. Ejemplo mínimo:

   ```yaml
   name: mlops-taller4
   channels: [conda-forge]
   dependencies:
   - python=3.10
   - numpy
   - pandas
   - scikit-learn
   - pip
   - pip:
       - joblib
   ```

   Se crea con conda env create -f environment.yml. Variaciones clave frente a requirements.txt: formato declarativo, soporte de canales/paquetes compilados, y posibilidad de exportar un "lock" reproducible (conda env export --no-builds > environment-lock.yml). En proyectos mixtos, se suele mantener ambos: environment.yml para usuarios de Conda y requirements.txt para pip/venv.

3. Conclusiones: usos de ambientes virtuales en ML

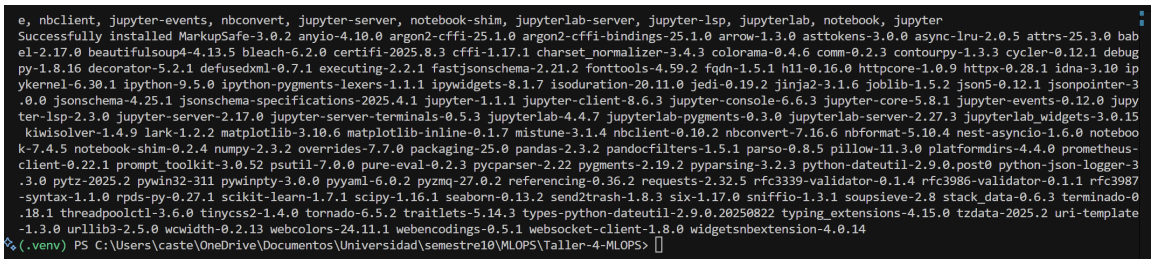
   Los ambientes virtuales son esenciales para aislar dependencias, garantizar reproducibilidad y evitar "dependency hell" en ML, donde pequeñas diferencias de versiones (ej., numpy, pandas, scikit-learn) cambian resultados o rompen pipelines. Permiten:

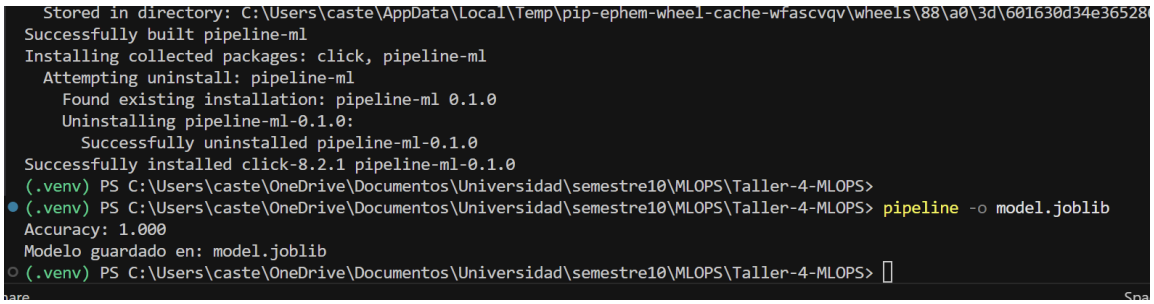
   - Onboarding rápido del equipo (un comando reconstruye el stack)
   - Integración con CI/CD y experiments tracking al versionar los requisitos
   - Coexistencia de proyectos con librerías conflictivas (incluso bindings nativos y GPU)
   - Portabilidad entre máquinas/servidores

- Empaquetar y desplegar entrenamientos/servicios con garantías de entorno. En suma, capturar el entorno con requirements.txt/environment.yml y usar venv/Conda reduce fricción, acota riesgos y acelera el ciclo de vida MLOps de datos-→modelo-→producción.

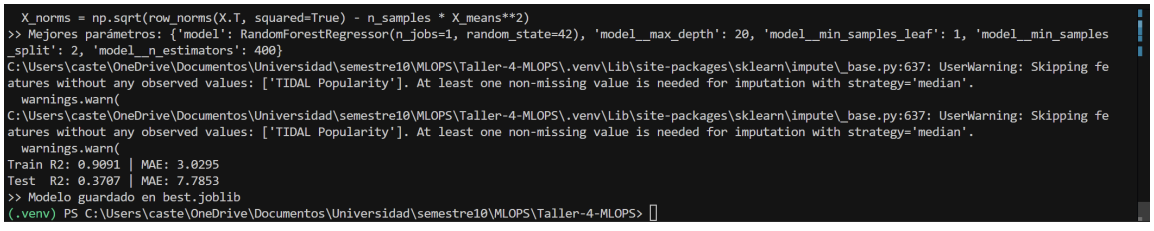
Ambiente en Python

Instalación de los requirements en el ambiente de python



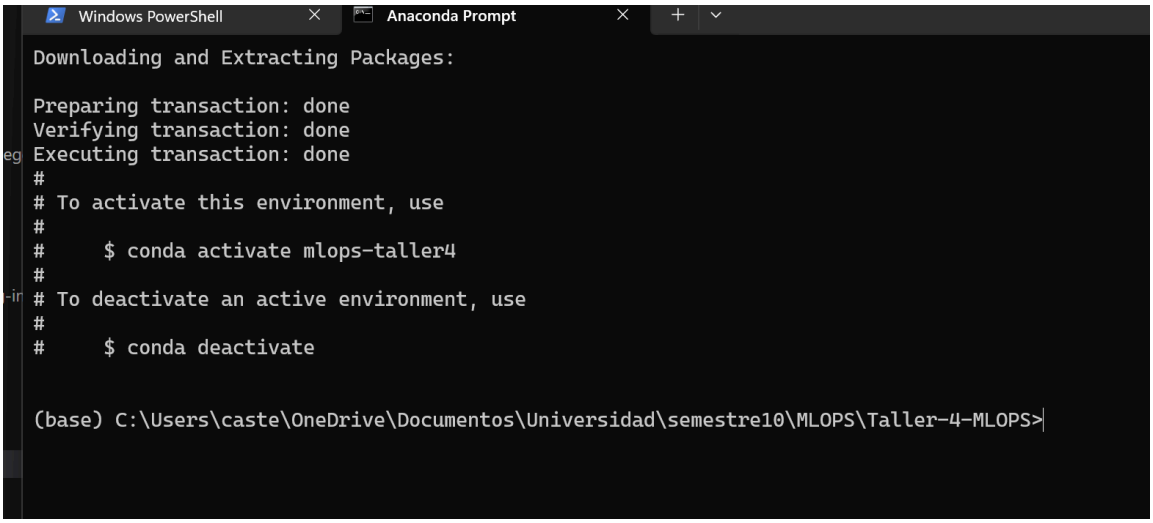


Ejecución del pipeline con el ambiente



Ambiente con Conda

Creación



Instalación

Ejecución

```
gy='median'.
  warnings.warn(
[CV] END model=StackingRegressor(estimators=[('rf',
                               RandomForestRegressor(n_jobs=1,
                                                     random_state=42)),
                              ('gb',
                               GradientBoostingRegressor(random_state=42))],
                  final_estimator=GradientBoostingRegressor(), n_jobs=1); total time=  23.6s
C:\Users\caste\anaconda3\envs\mlops-taller4\Lib\site-packages\sklearn\impute\_base.py:653: UserWarning: Skipping feature
s without any observed values: ['TIDAL Popularity']. At least one non-missing value is needed for imputation with strate
gy='median'.
  warnings.warn(
C:\Users\caste\anaconda3\envs\mlops-taller4\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:380: Ru
ntimeWarning: invalid value encountered in sqrt
  X_norms = np.sqrt(row_norms(X.T, squared=True) - n_samples * X_means**2)
>> Mejores parámetros: {'model': RandomForestRegressor(n_jobs=1, random_state=42), 'model__max_depth': 20, 'model__min_s
amples_leaf': 1, 'model__min_samples_split': 2, 'model__n_estimators': 400}
C:\Users\caste\anaconda3\envs\mlops-taller4\Lib\site-packages\sklearn\impute\_base.py:653: UserWarning: Skipping feature
s without any observed values: ['TIDAL Popularity']. At least one non-missing value is needed for imputation with strate
gy='median'.
  warnings.warn(
C:\Users\caste\anaconda3\envs\mlops-taller4\Lib\site-packages\sklearn\impute\_base.py:653: UserWarning: Skipping feature
s without any observed values: ['TIDAL Popularity']. At least one non-missing value is needed for imputation with strate
gy='median'.
  warnings.warn(
Train R2: 0.9091 | MAE: 3.0295
Test  R2: 0.3707 | MAE: 7.7853
>> Modelo guardado en best.joblib

(mlops-taller4) C:\Users\caste\OneDrive\Documentos\Universidad\semestre10\MLOPS\Taller-4-MLOPS>
```