Fine-tuning an **XGBRegressor** (from the scikit-learn API) typically means either:

1. **Continuing training from a previously trained model**, or
2. **Tuning hyperparameters** using methods like GridSearchCV or RandomizedSearchCV.

Here’s how you can do both:

**✅ 1. Load a Saved XGBRegressor Model**

Assuming it was saved using joblib or pickle:

import joblib

model = joblib.load("xgb\_regressor\_model.pkl")

**🔁 2. Continue Training (Fine-Tune on More Data)**

You can use the fit() method again to continue training, **but you must set xgb\_model=model**:

model.fit(X\_new, y\_new, xgb\_model=model.get\_booster())

⚠️ Make sure:

* New data (X\_new, y\_new) is from the same distribution.
* Preprocessing is consistent.

**🔧 3. Hyperparameter Tuning (GridSearchCV Example)**

You can use GridSearchCV to search for better hyperparameters:

from sklearn.model\_selection import GridSearchCV

from xgboost import XGBRegressor

param\_grid = {

'n\_estimators': [100, 200],

'learning\_rate': [0.01, 0.1],

'max\_depth': [3, 5, 7],

'subsample': [0.8, 1.0]

}

xgb = XGBRegressor(objective='reg:squarederror', random\_state=42)

grid = GridSearchCV(estimator=xgb, param\_grid=param\_grid,

cv=3, scoring='neg\_mean\_squared\_error', verbose=1)

grid.fit(X\_train, y\_train)

# Get the best model

best\_model = grid.best\_estimator\_

**💾 Save Fine-Tuned Model**

joblib.dump(best\_model, "xgb\_regressor\_finetuned.pkl")

Would you like help choosing good hyperparameter ranges for your dataset or task?