Week3 Assignments

Please do the assignments using the `mlops_eng` environment.

This week's assignments will give you some hands-on experience with Optuna (and Ray Tune). Similar to the tutorial of the first week, the red wine dataset will be used in this week's assignments.

Guidelines for submitting assignments:

- For each assignment, a code skeleton is provided. Please put your solutions between the `### START CODE HERE` and `### END CODE HERE` code comments. Please **do not change any code other than those between the `### START CODE HERE` and `### END CODE HERE` comments**. Otherwise your notebook may not pass the tests used in grading.
- Some assignments also require you to answer questions (in text) or capture screenshots in order to earn points. Please put your text answers and screenshots in a single PDF file. For each answer and screenshot, please clearly indicate which assignment it corresponds to in your PDF file. Please include the PDF file in your submission.
- In Assignments 1 and 2, you'll be asked to save your Optuna Studies in an SQLite database "optuna.sqlite3" (the database will be created when you proceed with the assignments). Please also include this database in your submission (**do not change the file name, just keep it as "optuna.sqlite3"**).

```
# import packages
import os
import pandas as pd
import numpy as np
import plotly
import platform
import time
import random
from typing import List, Dict
import logging
import warnings

import optuna
from optuna.integration.mlflow import MLflowCallback
from optuna.samplers import TPESampler
import mlflow
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
import lightgbm as lgb
import xgboost as xgb
from ray import tune
from ray.tune.search.optuna import OptunaSearch
```

```
2025-11-21 15:59:43,332 INFO util.py:154 -- Outdated packages:
  ipywidgets==7.8.1 found, needs ipywidgets>=8
Run `pip install -U ipywidgets`, then restart the notebook server for rich notebook output.
2025-11-21 15:59:44,772 INFO util.py:154 -- Outdated packages:
  ipywidgets==7.8.1 found, needs ipywidgets>=8
Run `pip install -U ipywidgets`, then restart the notebook server for rich notebook output.
```

```
import ray
ray.__version__
```

```
'2.9.3'
```

```
# Make sure you've installed the right version of lightgbm and xgboost
current_platform = platform.system()
print(f"Current platform: {current_platform}")
if current_platform == "Darwin" or current_platform == "Linux":
    if current_platform == "Darwin":  # macOS
        assert lgb.__version__.startswith("4.5.0"), f"Wrong version of lightgbm for platform: {current_platform}"
        assert xgb.__version__.startswith("2.0.3"), f"Wrong version of xgb for platform: {current_platform}"
    elif current_platform == "Linux": # Ubuntu
        assert lgb.__version__ == "4.0.0", f"Wrong version of lightgbm for platform: {current_platform}"
        assert xgb.__version__ == "2.0.3", f"Wrong version of xgboost for platform: {current_platform}"
    else:
        assert False, f"Unknown platform: {current_platform}"
else:
    assert False, f"Unexpected platform: {current_platform}"
```

```
Current platform: Linux
```

```python
# This is just for the grading purpose
def is_being_graded():
    """
    Returns True if the notebook is being executed by the auto-grading tool.
    """
    env = os.environ.get("NBGRADER_EXECUTION")
    return env == "autograde" or env == "validate"


# Suppress loggings and warnings when grading the notebook
if is_being_graded():
    loggers = [logging.getLogger(name) for name in logging.root.manager.loggerDict]
    for logger in loggers:
        logger.setLevel(logging.ERROR)
    mlflow.utils.logging_utils.disable_logging()
    warnings.filterwarnings("ignore")
```

```python
# Random seed for making the assignments reproducible
RANDOM_SEED = 42

# MLflow service URI
mlflow_tracking_uri = "http://mlflow-server.local"

# Configure MLflow
os.environ["MLFLOW_S3_ENDPOINT_URL"] = "http://mlflow-minio.local"
os.environ["AWS_ACCESS_KEY_ID"] = "minioadmin"
os.environ["AWS_SECRET_ACCESS_KEY"] = "minioadmin"

mlflow.set_tracking_uri(mlflow_tracking_uri)
```

```python
# Prepare training and testing data
data = pd.read_csv("winequality-red.csv", delimiter=";")

X = data.drop("quality", axis=1)
y = data["quality"]

train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.25, random_state=RANDOM_SEED)
```

```python
# An overview of the original dataset
data.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

```python
print(f"The dimension of train_x is {train_x.shape}")
print(f"The dimension of train_y is {train_y.shape}")
print(f"The dimension of test_x is {test_x.shape}")
print(f"The dimension of test_y is {test_y.shape}")
```

```
The dimension of train_x is (1199, 11)
The dimension of train_y is (1199,)
The dimension of test_x is (400, 11)
The dimension of test_y is (400,)
```

Assignment 1: The basic use of Optuna (2 points)

Your task is to use Optuna to find the optimal hyperparameter combination for the LightGBM regression model used for predicting red wine quality. **Please use the sklearn API as you did in the first week's assignments.**

This assignment has the following requirements:

1. Define the objective function (`objective_func`). The target of the optimization is to minimize the MAE (mean absolute error) of the model when evaluating the model against the testing dataset. The hyperparameters to be tuned and their search ranges are shown below. Some of the hyperparameter values are fixed. The hyperparameter values should be sampled in a linear domain if not separately specified. In your objective function, please specify the

hyperparameters in the same order as presented in the table.

Hyperparameter	Explanation	type	range
n_estimators	The number of decision trees.	integer	1000 (fixed value)
learning_rate	The step size of the gradient descent. It controls how quickly the model fits and then overfits the training data.	float	[0.001, 0.1] (sampled from the logarithmic domain)
subsample	The percentage of training samples to be used to train each tree. `subsample*100%` of the training samples will be randomly selected for training.	float	[0.05, 0.5]
subsample_freq	Subsampling frequency. The subsampling will be performed again after `subsample_freq` trees have been trained.	integer	1 (fixed value)
colsample_bytree	The percentage of features to use when training each tree.	float	[0.05, 0.5]
min_child_samples	A leaf node should have `min_child_samples` data points to be further splitted.	integer	[20, 100]
num_leaves	Max number of nodes in a single tree.	integer	[2, 2^10]
random_state	The seed for random number generation for reproducibility.	integer	RANDOM_SEED (fixed value, RANDOM_SEED has been defined as a variable in a previous cell)

2. Define another function named `run_study` that creates and runs a study. Detailed requirements are listed below:
    - Use TPESampler as the sampler for the hyperparameter sampling and use `RANDOM_SEED` as the seed of the sampler.
    - The Optuna study should have a name specified by the `study_name` argument, use the objective function given as the `objective_func` argument and perform `n_trials` trials.
    - The study history should be persisted in a relational database specified by the `storage` argument so that the study can be loaded and analyzed later.
    - The function should finally return the study.

Hints:

- How to sample hyperparameter values in the logarithmic domain?
- How to configure a study to use a specific sampler and persist study history in a specific database?
- You'll probably see LightGBM throw a bunch of warnings of "No further splits with positive gain, best gain: -inf". This warning basically means LightGBM can't find a split that would improve the model's performance at a particular node. You can suppress these warning by setting `verbose=-1` when defining your model, e.g., `lgb.LGBMRegressor(verbose=-1, ...)`

Notes:

- When define the search space for the hyperparameters, please use the names given in the "Hyperparameter" column in the table above.
- Please **do not** use deprecated methods (e.g, suggest_uniform and suggest_loguniform) when define the search space.

More reading material: If you are interested, the LightGBM documentation explains the use of each hyperparameter in more details.

```
# Define the objective function
def objective_func(trial):
    # TODO:
    # Define the hyperparameters to be tuned and their search ranges
    # Train a model using the sampled hyperparameters
    # Evaluate the model using the test dataset
    # Return the evaluation metric
    ### START CODE HERE
    params = {
        'n_estimators': 1000,
        'learning_rate': trial.suggest_float('learning_rate', 0.001, 0.1, log=True),
        'subsample': trial.suggest_float('subsample', 0.05, 0.5),
        'subsample_freq': 1,
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.05, 0.5),
        'min_child_samples': trial.suggest_int('min_child_samples', 20, 100),
        'num_leaves': trial.suggest_int('num_leaves', 2, 1024),
        'random_state': RANDOM_SEED,
        'verbose': -1
    }
    model = lgb.LGBMRegressor(**params)
    model.fit(train_x,train_y)
```

```python
    pred = model.predict(test_x)
    mae = mean_absolute_error(test_y,pred)
    return mae
    ### END CODE HERE

def run_study(study_name: str, storage: str, objective_func: callable, n_trials: int) -> optuna.study.Study:
    """
    Create and run an Optuna study.
    Args:
        study_name: The name of the study.
        storage: The URI of the storage used to save the study history.
        objective_func: The objective function to be optimized.
        n_trials: The number of trials the study should perform.
    Returns:
        A Study object.
    """
    # Delete the study if it already exists
    if study_name in optuna.get_all_study_names(storage=storage):
        optuna.delete_study(study_name=study_name, storage=storage)

    # TODO: Create (and run) the study and record the history in the storage
    ### START CODE HERE
    sampler = TPESampler(seed = RANDOM_SEED)
    study = optuna.create_study(
        study_name=study_name,
        storage=storage,
        sampler=sampler,
        direction='minimize'
    )
    study.optimize(objective_func,n_trials=n_trials,show_progress_bar=False)

    return study
    ### END CODE HERE
```

```python
# Assign "lgbm-wine-1" as the study_name to the Optuna study in the first assignment.
study_name_1 = "lgbm-wine-1"

# For this assignment, it is enough to use a simple sqlite3 database for persisting study history
storage = "sqlite:///optuna.sqlite3"

# When grading the notebook, the study will be loaded from the submitted database
study_1 = (
    run_study(
        study_name=study_name_1,
        storage=storage,
        objective_func=objective_func,
        n_trials=100,
    )
    if not is_being_graded()
    else optuna.load_study(study_name=study_name_1, storage=storage)
)

print("Best MAE", study_1.best_value)
print("Best params:", study_1.best_trial.params)
```

```
[I 2025-11-21 16:01:25,266] A new study created in RDB with name: lgbm-wine-1
[I 2025-11-21 16:01:26,097] Trial 0 finished with value: 0.48305757514051956 and parameters: {'learning_rate': 0
.005611516415334507, 'subsample': 0.4778214378844623, 'colsample_bytree': 0.3793972738151323, 'min_child_samples
': 68, 'num_leaves': 161}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:26,452] Trial 1 finished with value: 0.6685091743119267 and parameters: {'learning_rate': 0.
002051110418843397, 'subsample': 0.07613762547568977, 'colsample_bytree': 0.4397792655987208, 'min_child_samples
': 68, 'num_leaves': 726}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:27,229] Trial 2 finished with value: 0.5330860406894041 and parameters: {'learning_rate': 0.
0010994335574766201, 'subsample': 0.48645943347289744, 'colsample_bytree': 0.4245991883601898, 'min_child_sample
s': 37, 'num_leaves': 188}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:27,683] Trial 3 finished with value: 0.5338040960963436 and parameters: {'learning_rate': 0.
002327067708383781, 'subsample': 0.186909009331792, 'colsample_bytree': 0.28614039423450705, 'min_child_samples'
: 54, 'num_leaves': 299}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:28,290] Trial 4 finished with value: 0.5076269460531386 and parameters: {'learning_rate': 0.
01673808578875214, 'subsample': 0.11277223729341883, 'colsample_bytree': 0.1814650918408482, 'min_child_samples'
: 49, 'num_leaves': 468}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:28,690] Trial 5 finished with value: 0.5076748898380093 and parameters: {'learning_rate': 0.
037183641805732096, 'subsample': 0.1398532019712619, 'colsample_bytree': 0.28140549728612524, 'min_child_samples
': 67, 'num_leaves': 49}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:29,059] Trial 6 finished with value: 0.6685091743119267 and parameters: {'learning_rate': 0.
016409286730647923, 'subsample': 0.1267358556592812, 'colsample_bytree': 0.0792732168433758, 'min_child_samples'
: 96, 'num_leaves': 989}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:29,722] Trial 7 finished with value: 0.5051750796616936 and parameters: {'learning_rate': 0.
041380401125610165, 'subsample': 0.1870761961280168, 'colsample_bytree': 0.09395245130287275, 'min_child_samples
': 75, 'num_leaves': 452}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:30,141] Trial 8 finished with value: 0.602185209067021 and parameters: {'learning_rate': 0.0
017541893487450805, 'subsample': 0.2728296095500716, 'colsample_bytree': 0.06547483450184828, 'min_child_samples
': 93, 'num_leaves': 266}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:30,615] Trial 9 finished with value: 0.49454634175435186 and parameters: {'learning_rate': 0
.02113705944064573, 'subsample': 0.19026998424023495, 'colsample_bytree': 0.28403060953001485, 'min_child_sample
s': 64, 'num_leaves': 191}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 16:01:31,446] Trial 10 finished with value: 0.46421846512762593 and parameters: {'learning_rate':
0.005609875978923096, 'subsample': 0.46978789997535003, 'colsample_bytree': 0.36325301151124445, 'min_child_samp
les': 21, 'num_leaves': 17}. Best is trial 10 with value: 0.46421846512762593.
[I 2025-11-21 16:01:31,940] Trial 11 finished with value: 0.4839260622308465 and parameters: {'learning_rate': 0
.005581031669443425, 'subsample': 0.4925415013251626, 'colsample_bytree': 0.3706448031058781, 'min_child_sample
s': 26, 'num_leaves': 5}. Best is trial 10 with value: 0.46421846512762593.
[I 2025-11-21 16:01:32,692] Trial 12 finished with value: 0.46846727958527695 and parameters: {'learning_rate':
0.005633225713234431, 'subsample': 0.3990732670799304, 'colsample_bytree': 0.3557251924593569, 'min_child_sample
s': 24, 'num_leaves': 19}. Best is trial 10 with value: 0.46421846512762593.
[I 2025-11-21 16:01:33,780] Trial 13 finished with value: 0.4490225094819387 and parameters: {'learning_rate': 0
.006726317178272483, 'subsample': 0.39331551661530145, 'colsample_bytree': 0.4982190144284359, 'min_child_sample
s': 20, 'num_leaves': 662}. Best is trial 13 with value: 0.4490225094819387.
[I 2025-11-21 16:01:34,532] Trial 14 finished with value: 0.4623174947161377 and parameters: {'learning_rate': 0
.0091300828477436, 'subsample': 0.3602914705042889, 'colsample_bytree': 0.4984916911753534, 'min_child_samples':
36, 'num_leaves': 687}. Best is trial 13 with value: 0.4490225094819387.
[I 2025-11-21 16:01:35,215] Trial 15 finished with value: 0.44139549474694817 and parameters: {'learning_rate':
0.08538648795388477, 'subsample': 0.3625443334866371, 'colsample_bytree': 0.498207439286346, 'min_child_samples'
: 38, 'num_leaves': 684}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:35,878] Trial 16 finished with value: 0.45298158513185255 and parameters: {'learning_rate':
0.09970682275036434, 'subsample': 0.3436754646846178, 'colsample_bytree': 0.4683173574944437, 'min_child_samples
': 39, 'num_leaves': 714}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:36,475] Trial 17 finished with value: 0.495701894159936 and parameters: {'learning_rate': 0.
09909565063638189, 'subsample': 0.29572259322277433, 'colsample_bytree': 0.18655083539985107, 'min_child_samples
': 47, 'num_leaves': 885}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:37,194] Trial 18 finished with value: 0.4434038694428212 and parameters: {'learning_rate': 0
.03671556184809921, 'subsample': 0.4187086614418443, 'colsample_bytree': 0.48538282007418293, 'min_child_samples
': 30, 'num_leaves': 570}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:37,947] Trial 19 finished with value: 0.44335870966120255 and parameters: {'learning_rate':
0.053731799997625895, 'subsample': 0.43017554203830133, 'colsample_bytree': 0.4184832504175784, 'min_child_sampl
es': 33, 'num_leaves': 578}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:38,528] Trial 20 finished with value: 0.4625134893789983 and parameters: {'learning_rate': 0
.06319576078452216, 'subsample': 0.29178069551740093, 'colsample_bytree': 0.41731449937912957, 'min_child_sample
s': 44, 'num_leaves': 831}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:39,278] Trial 21 finished with value: 0.44448560452947283 and parameters: {'learning_rate':
0.04524540905246321, 'subsample': 0.4277926712672274, 'colsample_bytree': 0.44279227737290805, 'min_child_sample
s': 31, 'num_leaves': 558}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:39,927] Trial 22 finished with value: 0.4449876430487491 and parameters: {'learning_rate': 0
.027183592230793398, 'subsample': 0.42495777429581916, 'colsample_bytree': 0.4731338608597703, 'min_child_sample
s': 30, 'num_leaves': 537}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:40,512] Trial 23 finished with value: 0.47278352746529306 and parameters: {'learning_rate':
0.06634213926914975, 'subsample': 0.33888282632930455, 'colsample_bytree': 0.3319155110982273, 'min_child_sample
s': 55, 'num_leaves': 396}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:41,268] Trial 24 finished with value: 0.45156570004074614 and parameters: {'learning_rate':
0.059317013345016885, 'subsample': 0.4363746618457771, 'colsample_bytree': 0.39533671032494305, 'min_child_sampl
es': 31, 'num_leaves': 602}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:42,103] Trial 25 finished with value: 0.4548851002749927 and parameters: {'learning_rate': 0
.028653412461781675, 'subsample': 0.3606533936413034, 'colsample_bytree': 0.46890414257666063, 'min_child_sample
s': 41, 'num_leaves': 833}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:42,895] Trial 26 finished with value: 0.4647769208431393 and parameters: {'learning_rate': 0
.07673007352022662, 'subsample': 0.3852050523585826, 'colsample_bytree': 0.32285913841337865, 'min_child_samples
': 33, 'num_leaves': 613}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:43,370] Trial 27 finished with value: 0.48160079751638873 and parameters: {'learning_rate':
0.04793447045409202, 'subsample': 0.3239866908012082, 'colsample_bytree': 0.4073387573921768, 'min_child_samples
': 78, 'num_leaves': 359}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:43,917] Trial 28 finished with value: 0.48375889068571815 and parameters: {'learning_rate':
0.030847828892393674, 'subsample': 0.22398310957504441, 'colsample_bytree': 0.44939243060058803, 'min_child_samp
les': 53, 'num_leaves': 753}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:44,691] Trial 29 finished with value: 0.46912939971036527 and parameters: {'learning_rate':
0.012970437950994895, 'subsample': 0.4566181691646891, 'colsample_bytree': 0.21312225481011907, 'min_child_sampl
es': 27, 'num_leaves': 478}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:45,380] Trial 30 finished with value: 0.4551162002470042 and parameters: {'learning_rate': 0
.0794962164313919, 'subsample': 0.406847659791187, 'colsample_bytree': 0.49450564293479493, 'min_child_samples':
44, 'num_leaves': 619}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:46,191] Trial 31 finished with value: 0.4421583837166304 and parameters: {'learning_rate': 0
.04759114776546995, 'subsample': 0.4370055967845491, 'colsample_bytree': 0.44538215296236494, 'min_child_samples
': 33, 'num_leaves': 542}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:46,915] Trial 32 finished with value: 0.4491740558976645 and parameters: {'learning_rate': 0
.05252390675183853, 'subsample': 0.4609139238380871, 'colsample_bytree': 0.4417879397417666, 'min_child_samples'
: 37, 'num_leaves': 788}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:47,643] Trial 33 finished with value: 0.44595479283243106 and parameters: {'learning_rate':
0.03404578668311577, 'subsample': 0.44255113377054534, 'colsample_bytree': 0.46439197644990027, 'min_child_sampl
es': 34, 'num_leaves': 547}. Best is trial 15 with value: 0.44139549474694817.
```

```
[I 2025-11-21 16:01:48,422] Trial 34 finished with value: 0.453706253490258 and parameters: {'learning_rate': 0.
023309644660997556, 'subsample': 0.3778085828117565, 'colsample_bytree': 0.40380754958162035, 'min_child_samples
': 26, 'num_leaves': 396}. Best is trial 15 with value: 0.44139549474694817.
[I 2025-11-21 16:01:49,262] Trial 35 finished with value: 0.4377161778888277 and parameters: {'learning_rate': 0
.07877623013299838, 'subsample': 0.49214080229606594, 'colsample_bytree': 0.41886526176548317, 'min_child_sample
s': 42, 'num_leaves': 640}. Best is trial 35 with value: 0.4377161778888277.
[I 2025-11-21 16:01:50,015] Trial 36 finished with value: 0.4407666942697611 and parameters: {'learning_rate': 0
.08097967547298585, 'subsample': 0.49001305105479465, 'colsample_bytree': 0.4276260557279081, 'min_child_samples
': 49, 'num_leaves': 654}. Best is trial 35 with value: 0.4377161778888277.
[I 2025-11-21 16:01:50,723] Trial 37 finished with value: 0.4696496215679975 and parameters: {'learning_rate': 0
.07913576300786627, 'subsample': 0.4908714010652557, 'colsample_bytree': 0.33462124859324527, 'min_child_samples
': 59, 'num_leaves': 654}. Best is trial 35 with value: 0.4377161778888277.
[I 2025-11-21 16:01:51,177] Trial 38 finished with value: 0.6685091743119267 and parameters: {'learning_rate': 0
.09990946357932352, 'subsample': 0.05625390118055745, 'colsample_bytree': 0.23880172327603824, 'min_child_sample
s': 51, 'num_leaves': 912}. Best is trial 35 with value: 0.4377161778888277.
[I 2025-11-21 16:01:51,858] Trial 39 finished with value: 0.45057707385935303 and parameters: {'learning_rate':
0.07692220877479171, 'subsample': 0.4931263223454522, 'colsample_bytree': 0.38450948301304494, 'min_child_sample
s': 43, 'num_leaves': 761}. Best is trial 35 with value: 0.4377161778888277.
[I 2025-11-21 16:01:52,501] Trial 40 finished with value: 0.499769629399527 and parameters: {'learning_rate': 0.
003014274727230161, 'subsample': 0.460642860715639, 'colsample_bytree': 0.3067698679263518, 'min_child_samples':
48, 'num_leaves': 501}. Best is trial 35 with value: 0.4377161778888277.
[I 2025-11-21 16:01:53,238] Trial 41 finished with value: 0.4350363678628525 and parameters: {'learning_rate': 0
.053592095481482166, 'subsample': 0.47526669170357255, 'colsample_bytree': 0.42220385944882993, 'min_child_sampl
es': 39, 'num_leaves': 711}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:53,813] Trial 42 finished with value: 0.4570860935401379 and parameters: {'learning_rate': 0
.06273439297188611, 'subsample': 0.4743738247970897, 'colsample_bytree': 0.4313996046290791, 'min_child_samples'
: 58, 'num_leaves': 705}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:54,506] Trial 43 finished with value: 0.5127803359955099 and parameters: {'learning_rate': 0
.04622925368367998, 'subsample': 0.4479455160957899, 'colsample_bytree': 0.12988133261251145, 'min_child_samples
': 39, 'num_leaves': 653}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:55,348] Trial 44 finished with value: 0.4520216371501099 and parameters: {'learning_rate': 0
.039923542401704225, 'subsample': 0.4735474167684242, 'colsample_bytree': 0.45702245838486066, 'min_child_sample
s': 47, 'num_leaves': 808}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:56,248] Trial 45 finished with value: 0.4585698306925107 and parameters: {'learning_rate': 0
.01862919735728007, 'subsample': 0.49541169144530306, 'colsample_bytree': 0.37714123053854537, 'min_child_sample
s': 41, 'num_leaves': 996}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:56,750] Trial 46 finished with value: 0.46723986030915937 and parameters: {'learning_rate':
0.08143598912281705, 'subsample': 0.4066576531053914, 'colsample_bytree': 0.4271632147716544, 'min_child_samples
': 72, 'num_leaves': 432}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:57,312] Trial 47 finished with value: 0.4744042019838017 and parameters: {'learning_rate': 0
.06692279023681458, 'subsample': 0.4696025476030517, 'colsample_bytree': 0.3451959185331103, 'min_child_samples'
: 88, 'num_leaves': 736}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:57,900] Trial 48 finished with value: 0.4700499967336279 and parameters: {'learning_rate': 0
.05211354781900806, 'subsample': 0.44517254939779666, 'colsample_bytree': 0.3922725445735883, 'min_child_samples
': 64, 'num_leaves': 870}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:58,612] Trial 49 finished with value: 0.46747174556821686 and parameters: {'learning_rate':
0.013537015144585336, 'subsample': 0.4973120081461907, 'colsample_bytree': 0.2582447508186682, 'min_child_sample
s': 50, 'num_leaves': 681}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:59,258] Trial 50 finished with value: 0.5351653910637607 and parameters: {'learning_rate': 0
.001113717572032855, 'subsample': 0.37272577569360094, 'colsample_bytree': 0.4760627894157579, 'min_child_sample
s': 37, 'num_leaves': 511}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:01:59,926] Trial 51 finished with value: 0.44462653361882304 and parameters: {'learning_rate':
0.05386181933986789, 'subsample': 0.4150123887681665, 'colsample_bytree': 0.4227892201080663, 'min_child_samples
': 35, 'num_leaves': 589}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:00,600] Trial 52 finished with value: 0.4541022446729782 and parameters: {'learning_rate': 0
.039698329877918936, 'subsample': 0.23973845003841482, 'colsample_bytree': 0.4461247811375621, 'min_child_sample
s': 23, 'num_leaves': 637}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:01,277] Trial 53 finished with value: 0.4536196777856762 and parameters: {'learning_rate': 0
.08841022084971306, 'subsample': 0.43499723380940003, 'colsample_bytree': 0.4131866559335381, 'min_child_samples
': 44, 'num_leaves': 710}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:01,993] Trial 54 finished with value: 0.4559363429239926 and parameters: {'learning_rate': 0
.06888172558661687, 'subsample': 0.4739443280425542, 'colsample_bytree': 0.36101012251582254, 'min_child_samples
': 39, 'num_leaves': 573}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:02,859] Trial 55 finished with value: 0.43635166888938753 and parameters: {'learning_rate':
0.024699173414371754, 'subsample': 0.4532245935509078, 'colsample_bytree': 0.479552754185874, 'min_child_samples
': 28, 'num_leaves': 521}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:03,684] Trial 56 finished with value: 0.43583269346521675 and parameters: {'learning_rate':
0.02372028072370049, 'subsample': 0.4800941293102379, 'colsample_bytree': 0.4842767841825664, 'min_child_samples
': 28, 'num_leaves': 297}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:04,449] Trial 57 finished with value: 0.43637220358923573 and parameters: {'learning_rate':
0.02267891948732951, 'subsample': 0.48134848703791305, 'colsample_bytree': 0.485577737257115, 'min_child_samples
': 27, 'num_leaves': 126}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:05,297] Trial 58 finished with value: 0.4379977624180333 and parameters: {'learning_rate': 0
.02365981202805777, 'subsample': 0.48240164138235786, 'colsample_bytree': 0.4876154065023363, 'min_child_samples
': 28, 'num_leaves': 178}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:05,751] Trial 59 finished with value: 0.49003793198500206 and parameters: {'learning_rate':
0.021341870885220104, 'subsample': 0.09777835082231806, 'colsample_bytree': 0.49101263934987976, 'min_child_samp
les': 28, 'num_leaves': 111}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:06,234] Trial 60 finished with value: 0.4666835604982235 and parameters: {'learning_rate': 0
```

```
.011373324413229677, 'subsample': 0.16351439780768762, 'colsample_bytree': 0.4798039957160988, 'min_child_sample
s': 22, 'num_leaves': 234}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:07,090] Trial 61 finished with value: 0.43538731094259264 and parameters: {'learning_rate':
0.025136148701684312, 'subsample': 0.48226100995738247, 'colsample_bytree': 0.4609592343485887, 'min_child_sampl
es': 28, 'num_leaves': 130}. Best is trial 41 with value: 0.4350363678628525.
[I 2025-11-21 16:02:08,325] Trial 62 finished with value: 0.42014872405310155 and parameters: {'learning_rate':
0.025076400520261857, 'subsample': 0.48136029577521056, 'colsample_bytree': 0.4581941479625377, 'min_child_sampl
es': 20, 'num_leaves': 135}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:09,181] Trial 63 finished with value: 0.42938443310297053 and parameters: {'learning_rate':
0.016426087682754262, 'subsample': 0.4566488347352516, 'colsample_bytree': 0.4556767954225494, 'min_child_sample
s': 20, 'num_leaves': 109}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:09,996] Trial 64 finished with value: 0.4433318825854681 and parameters: {'learning_rate': 0
.008334029082619964, 'subsample': 0.45699862213521586, 'colsample_bytree': 0.47587196033862356, 'min_child_sampl
es': 20, 'num_leaves': 122}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:10,730] Trial 65 finished with value: 0.43775066142426694 and parameters: {'learning_rate':
0.016371900471575485, 'subsample': 0.4583044601986897, 'colsample_bytree': 0.4629397937643391, 'min_child_sample
s': 24, 'num_leaves': 75}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:11,459] Trial 66 finished with value: 0.4361187221859765 and parameters: {'learning_rate': 0
.017043568309508164, 'subsample': 0.47777591969012995, 'colsample_bytree': 0.45580890293737925, 'min_child_sampl
es': 25, 'num_leaves': 278}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:12,296] Trial 67 finished with value: 0.4361945343078869 and parameters: {'learning_rate': 0
.017644849560939396, 'subsample': 0.4225445771223221, 'colsample_bytree': 0.4500527643810497, 'min_child_samples
': 20, 'num_leaves': 281}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:13,125] Trial 68 finished with value: 0.4303228927744931 and parameters: {'learning_rate': 0
.017121397816868614, 'subsample': 0.4286125059322883, 'colsample_bytree': 0.45737939264061844, 'min_child_sample
s': 20, 'num_leaves': 291}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:13,955] Trial 69 finished with value: 0.44115638698345633 and parameters: {'learning_rate':
0.013332015113264453, 'subsample': 0.4703398233082029, 'colsample_bytree': 0.4598346301458862, 'min_child_sample
s': 24, 'num_leaves': 330}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:14,739] Trial 70 finished with value: 0.44308644838500955 and parameters: {'learning_rate':
0.014898301077388128, 'subsample': 0.3963988487201868, 'colsample_bytree': 0.4323017834011807, 'min_child_sample
s': 23, 'num_leaves': 216}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:15,520] Trial 71 finished with value: 0.4287063288415011 and parameters: {'learning_rate': 0
.018446524034688293, 'subsample': 0.4197871737945318, 'colsample_bytree': 0.446217420824492, 'min_child_samples'
: 20, 'num_leaves': 271}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:16,334] Trial 72 finished with value: 0.4445428802407598 and parameters: {'learning_rate': 0
.010728026489663202, 'subsample': 0.49998188506312835, 'colsample_bytree': 0.45431402793774645, 'min_child_sampl
es': 25, 'num_leaves': 240}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:16,965] Trial 73 finished with value: 0.45491205909010934 and parameters: {'learning_rate':
0.03332040360600946, 'subsample': 0.41628174326825323, 'colsample_bytree': 0.4039630076307149, 'min_child_sample
s': 31, 'num_leaves': 314}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:17,870] Trial 74 finished with value: 0.4215266214137319 and parameters: {'learning_rate': 0
.028348827409211578, 'subsample': 0.44441853575068074, 'colsample_bytree': 0.4373882131020728, 'min_child_sample
s': 20, 'num_leaves': 45}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:18,804] Trial 75 finished with value: 0.43198045481428815 and parameters: {'learning_rate':
0.027091712684707964, 'subsample': 0.44589156552932935, 'colsample_bytree': 0.43401242514947463, 'min_child_samp
les': 21, 'num_leaves': 51}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:19,624] Trial 76 finished with value: 0.42873525042280974 and parameters: {'learning_rate':
0.019859130336736645, 'subsample': 0.4321082716742556, 'colsample_bytree': 0.4363796082598203, 'min_child_sample
s': 21, 'num_leaves': 39}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:20,588] Trial 77 finished with value: 0.4404805468416916 and parameters: {'learning_rate': 0
.028625992963205827, 'subsample': 0.42991112492694045, 'colsample_bytree': 0.4007412853225355, 'min_child_sample
s': 20, 'num_leaves': 51}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:21,285] Trial 78 finished with value: 0.44670018300473785 and parameters: {'learning_rate':
0.01996673211822791, 'subsample': 0.3920744922999444, 'colsample_bytree': 0.3866982203898095, 'min_child_samples
': 22, 'num_leaves': 77}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:22,073] Trial 79 finished with value: 0.43999474963830454 and parameters: {'learning_rate':
0.015176456324180451, 'subsample': 0.4029133584333072, 'colsample_bytree': 0.44075717146489335, 'min_child_sampl
es': 22, 'num_leaves': 27}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:22,826] Trial 80 finished with value: 0.45858981297338525 and parameters: {'learning_rate':
0.00842640260722713, 'subsample': 0.3451828489974794, 'colsample_bytree': 0.3709843257753698, 'min_child_samples
': 20, 'num_leaves': 152}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:23,568] Trial 81 finished with value: 0.43617714805439556 and parameters: {'learning_rate':
0.026740288270578217, 'subsample': 0.4368389368521033, 'colsample_bytree': 0.43167765360072924, 'min_child_sampl
es': 25, 'num_leaves': 84}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:24,321] Trial 82 finished with value: 0.4338117329136626 and parameters: {'learning_rate': 0
.03172335618898072, 'subsample': 0.44587949391710274, 'colsample_bytree': 0.466668944766839, 'min_child_samples'
: 22, 'num_leaves': 44}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:25,023] Trial 83 finished with value: 0.43259003445332467 and parameters: {'learning_rate':
0.03118773035968867, 'subsample': 0.4461095103602265, 'colsample_bytree': 0.4378720271594881, 'min_child_samples
': 22, 'num_leaves': 44}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:25,589] Trial 84 finished with value: 0.45729210924348224 and parameters: {'learning_rate':
0.0314855519301292, 'subsample': 0.41034324379360004, 'colsample_bytree': 0.4387191122243962, 'min_child_samples
': 23, 'num_leaves': 5}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:26,318] Trial 85 finished with value: 0.43395979747969293 and parameters: {'learning_rate':
0.020678683991104212, 'subsample': 0.4456676625406879, 'colsample_bytree': 0.49926769417347716, 'min_child_sampl
es': 22, 'num_leaves': 35}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:27,051] Trial 86 finished with value: 0.4418402805576459 and parameters: {'learning_rate': 0
.03760905684274078, 'subsample': 0.4266627124366876, 'colsample_bytree': 0.46803081102718974, 'min_child_samples
```

```
': 30, 'num_leaves': 62}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:27,761] Trial 87 finished with value: 0.44991832408785853 and parameters: {'learning_rate':
0.01965167847945125, 'subsample': 0.3788864022379647, 'colsample_bytree': 0.4165630981384861, 'min_child_samples
': 26, 'num_leaves': 154}. Best is trial 62 with value: 0.42014872405310155.
[I 2025-11-21 16:02:28,562] Trial 88 finished with value: 0.418099151103134 and parameters: {'learning_rate': 0.
02967900426626908, 'subsample': 0.46364572682407335, 'colsample_bytree': 0.4107961679403244, 'min_child_samples'
: 20, 'num_leaves': 181}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:29,357] Trial 89 finished with value: 0.42210505320880665 and parameters: {'learning_rate':
0.027979011013985228, 'subsample': 0.42087749342053765, 'colsample_bytree': 0.4095763971310802, 'min_child_sampl
es': 20, 'num_leaves': 193}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:30,143] Trial 90 finished with value: 0.4428766435364821 and parameters: {'learning_rate': 0
.012259682419929096, 'subsample': 0.4614787028223032, 'colsample_bytree': 0.4090802273573943, 'min_child_samples
': 20, 'num_leaves': 199}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:30,884] Trial 91 finished with value: 0.4349726583633044 and parameters: {'learning_rate': 0
.029797373946091915, 'subsample': 0.4215867689535292, 'colsample_bytree': 0.43861694403588025, 'min_child_sample
s': 25, 'num_leaves': 103}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:31,767] Trial 92 finished with value: 0.42892617342403627 and parameters: {'learning_rate':
0.027078851342445504, 'subsample': 0.43507291247496416, 'colsample_bytree': 0.43350184514903894, 'min_child_samp
les': 21, 'num_leaves': 169}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:32,691] Trial 93 finished with value: 0.4387265958428051 and parameters: {'learning_rate': 0
.025743618214524613, 'subsample': 0.4638011570764096, 'colsample_bytree': 0.3963371092574062, 'min_child_samples
': 20, 'num_leaves': 163}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:33,469] Trial 94 finished with value: 0.4301751253834087 and parameters: {'learning_rate': 0
.03549373724618361, 'subsample': 0.4369847851580642, 'colsample_bytree': 0.42335878331673354, 'min_child_samples
': 24, 'num_leaves': 249}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:34,174] Trial 95 finished with value: 0.4416876158008614 and parameters: {'learning_rate': 0
.041851178014467964, 'subsample': 0.43463906929703844, 'colsample_bytree': 0.41513131992359087, 'min_child_sampl
es': 30, 'num_leaves': 199}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:34,878] Trial 96 finished with value: 0.509805118434688 and parameters: {'learning_rate': 0.
0153524684847997, 'subsample': 0.3898263184266103, 'colsample_bytree': 0.11479400014345628, 'min_child_samples':
24, 'num_leaves': 260}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:35,621] Trial 97 finished with value: 0.4447261746905309 and parameters: {'learning_rate': 0
.03530761895483818, 'subsample': 0.40262589376625196, 'colsample_bytree': 0.4503849846998152, 'min_child_samples
': 26, 'num_leaves': 170}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:36,121] Trial 98 finished with value: 0.48350813420006644 and parameters: {'learning_rate':
0.02193815881261892, 'subsample': 0.36948998299484515, 'colsample_bytree': 0.37714126615986726, 'min_child_sampl
es': 85, 'num_leaves': 352}. Best is trial 88 with value: 0.418099151103134.
[I 2025-11-21 16:02:36,794] Trial 99 finished with value: 0.4574962027341036 and parameters: {'learning_rate': 0
.01764278488119714, 'subsample': 0.4126330794837564, 'colsample_bytree': 0.2957732251789356, 'min_child_samples'
: 32, 'num_leaves': 240}. Best is trial 88 with value: 0.418099151103134.
```

```
Best MAE 0.418099151103134
Best params: {'learning_rate': 0.02967900426626908, 'subsample': 0.46364572682407335, 'colsample_bytree': 0.4107
961679403244, 'min_child_samples': 20, 'num_leaves': 181}
```

Example output:

```
Best MAE 0.418099151103134
Best params: {'learning_rate': 0.02967900426626908, 'subsample': 0.46364572682407335,
'colsample_bytree': 0.4107961679403244, 'min_child_samples': 20, 'num_leaves': 181}
```

```python
# Check the optimization results

# The required hyperparameters should be optimized
for param in ["learning_rate", "subsample", "colsample_bytree", "min_child_samples", "num_leaves"]:
    assert param in study_1.best_trial.params, f"Missing hyperparameter {param} from the objective function"

# The searching ranges should be correct
assert study_1.best_trial.distributions.get("learning_rate").log is True, "learning_rate should be searched in a
for param in ["learning_rate", "subsample", "colsample_bytree"]:
    assert isinstance(study_1.best_trial.distributions.get(param), optuna.distributions.FloatDistribution), f"{pa

for param in ["min_child_samples", "num_leaves"]:
    assert isinstance(study_1.best_trial.distributions.get(param), optuna.distributions.IntDistribution), f"{para

# The study should perform 100 trials
assert len(study_1.trials) == 100, "Wrong number of trials"
```

```python
# Test the Optuna study results
assert study_1.best_value < 0.42, "Too large MAE"
```

Assignment 2: Analyzing an Optuna study (2 points)

Optuna offers utility functions for visualizing the optimization process (i.e., study history). For example, it can plot the hyperparameter importance and the relationship between a hyperparameter and the objective. In this assignment, you need to analyze the study created in Assignment 1, adjust the search ranges of some hyperparameters to obtain better MAE. Detailed instructions will be provided later.

```python
import plotly.io as pio

# Configure Jupyter Notebook to render plotly figures drawn by Optuna
pio.renderers.default = "notebook"
```

2a) Hyperparameter importance

Implement a function `show_param_importances` that loads an Optuna study from a storage and return a plot showing importance of the hyperparameters in the study. Similar to the tutorial, use the FanovaImportanceEvaluator as the importance evaluator and set `RANDOM_STATE` as the seed for the evaluator for reproducibility.

```python
# Plot the hyperparameter importance
def show_param_importances(study_name: str, storage: str) -> plotly.graph_objs.Figure:
    """
    Plot the hyperparameter importance of a study.
    Args:
        study_name: The name of the study.
        storage: The URI of the storage used to save the study history.
    Returns:
        A plotly Figure object.
    """
    ### START CODE HERE
    study = optuna.load_study(study_name=study_name,storage=storage)
    importance_evaluator = optuna.importance.FanovaImportanceEvaluator()
    res = optuna.visualization.plot_param_importances(
        study,
        evaluator=importance_evaluator
    )
    return res
    ### END CODE HERE

param_importance_fig_wine = show_param_importances(study_name=study_name_1, storage=storage)
param_importance_fig_wine.show()
```

Next, put the names of the three most important hyperparameters in to a list named `important_hyperparams`

```python
# TODO: important_hyperparams = ...
### START CODE HERE
important_hyperparams = ['learning_rate', 'num_leaves', 'colsample_bytree']
### END CODE HERE
```

```python
assert isinstance(param_importance_fig_wine, plotly.graph_objs.Figure), "Incorrect importance plot"
assert len(important_hyperparams) == 3, "Incorrect number of important hyperparameters"
```

2b) The impact of hyperparameters

Complete the `plot_relationship_between_hyperparams_and_obj` function that loads an Optuna study from a storage and return a plot showing the relationships of the most three important hyperparameters and the objective in a slice plot.

Hint: How to plot the relationship between a hyperparameter and the objective?

```python
def show_relationship_between_hyperparams_and_obj(study_name: str, storage: str, important_hyperparams: List[str]
    """
    Plot the relationship between the hyperparameters and the objective function.
    Args:
        study_name: The name of the study.
        storage: The URI of the storage used to save the study history.
        important_hyperparams: A list of hyperparameters that are considered important.
    Returns:
        A plotly Figure object.
    """
    ### START CODE HERE
    study = optuna.load_study(study_name=study_name, storage=storage)
    fig = optuna.visualization.plot_slice(
        study,
        params=important_hyperparams
    )
    return fig
    ### END CODE HERE

slice_fig_wine = show_relationship_between_hyperparams_and_obj(study_name=study_name_1, storage=storage, importan
slice_fig_wine.show()
```

```python
assert isinstance(slice_fig_wine, plotly.graph_objs.Figure), "Incorrect slice plot"
```

Question for Assignment 2b

Now you know the most three important hyperparameters that affect the objective value. Looking at the positions of the points in the slice plot, which area the points that resulted in better MAE are concentrated on? How would you adjust the search ranges of these three hyperparameters in the next study? Please put your answer in your PDF file.

Three most important hyperparameters are usually: learning_rate, num_leaves, and colsample_bytree.we should narrow the search ranges of these most important hyperparameters to the optimal areas observed in the Slice Plot.

2c) An improved Optuna study

In this assignment, you have two tasks. First, complete another objective function `objective_func_2`, where you need to adjust the search ranges of the most three important hyperparameters to improve MAE (keep values/search ranges of other hyperparameters as the same as in Assignment 1).

Then complete the `run_improved_study` that starts and runs another Optuna study. Detailed requirements are listed below:

- Use TPESampler as the sampler for the hyperparameter sampling and use `RANDOM_SEED` as the seed of the sampler.
- The Optuna study should have a name specified by the `new_study_name` argument, use the objective function given as the `objective_func` argument and perform `n_trials` trials.
- The study history should be persisted in a relational database specified by the `storage` argument so that the study can be loaded and analyzed later.
- You should start the study using the best hyperparameter combination of a previous study. The previous study should be loaded from the given storage using the study name given as the `prev_study_name` argument.
- The trials should be tracked in an MLflow experiment.
- The function should finally return the study.

Hints:

- Now you know the most three important hyperparameters that affect the objective value. Looking at the positions of the points in the slice plot, which areas the points that resulted in better MAE are concentrated on?
- How to log Optuna trials to MLflow?
- To start the new study using the best hyperparameter combination of the previous study, you can 1) load the previous study whose name is given as the `prev_study_name` argument, 2) retrieve the optimal hyperparameters combination of the previous study, 3) create a new study, 4) insert a trial with the best hyperparameter values of the previous study into the new study, and then start the optimization of the new study.

You may also find the following links helpful:

- optuna.load_study
- study.best_params
- enqueue_trial.

Note: If you want to delete all the trials in MLflow, don't delete the experiment, as deleting the experiment using the UI will not really delete the experiment in the PostgreSQL database used by MLflow, which will cause problems when recording trials under an experiment with the same name as the deleted experiment. Instead, you can keep the experiment and delete the trials, as shown in the image below. (If you feel like permanently deleting MLflow experiments from the PostgreSQL database, please check the `delete_from_mlflow.md` file in Week1 tutorials.

▶ If callback and decorator are new to you...

```python
# Define the objective function
def improved_objective_func(trial):
    ### START CODE HERE
    params = {
        'n_estimators': 1000,
        'learning_rate': trial.suggest_float('learning_rate', 0.005, 0.08, log=True),
        'subsample': trial.suggest_float('subsample', 0.05, 0.5),
        'subsample_freq': 1,
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.3, 0.9),
        'min_child_samples': trial.suggest_int('min_child_samples', 10, 150),
        'num_leaves': trial.suggest_int('num_leaves', 100, 800),
        'random_state': RANDOM_SEED,
        'verbose': -1
    }
    model = lgb.LGBMRegressor(**params)
    model.fit(train_x, train_y)

    predictions = model.predict(test_x)
    mae = mean_absolute_error(test_y, predictions)

    return mae
    ### END CODE HERE

def run_improved_study(prev_study_name: str, new_study_name: str, storage: str, objective_func: callable, n_trial
    """
    Create and run an Optuna study based on a previous study.
    Args:
        prev_study_name: The name of the previous study.
        new_study_name: The name of the new study.
        storage: The URI of the storage used to save the study history.
        objective_func: The objective function to be optimized.
        n_trials: The number of trials the study should perform.
    Returns:
        An Optuna Study object.
    """
    # Some cleanup before starting the new study
    # Delete the study if it already exists
    if new_study_name in optuna.get_all_study_names(storage=storage):
        optuna.delete_study(study_name=new_study_name, storage=storage)

    mlflow_exp = mlflow.get_experiment_by_name(new_study_name)
    if mlflow_exp is not None:
        # Delete all old trials in the experiment
        for run in mlflow.search_runs(mlflow_exp.experiment_id, output_format="list"):
            mlflow.delete_run(run.info.run_id)

    ### START CODE HERE
    prev_study = optuna.load_study(study_name=prev_study_name,storage=storage)
    prev_study_best_params = prev_study.best_params

    sampler = TPESampler(seed=RANDOM_SEED)
    study = optuna.create_study(
        study_name = new_study_name,
        storage=storage,
        sampler = sampler,
        direction='minimize'
    )
    study.enqueue_trial(prev_study_best_params)
    mlflow_callback = MLflowCallback(
        tracking_uri = mlflow_tracking_uri,
        metric_name = "mae"
    )
    study.optimize(
        objective_func,
        n_trials=n_trials,
        callbacks=[mlflow_callback],
        show_progress_bar=False
    )
    return study
    ### END CODE HERE
```

```
# Name of the new study
new_study_name = "lgbm-wine-2"

study_2 = (
    run_improved_study(
        prev_study_name=study_name_1,
        new_study_name=new_study_name,
        storage=storage,
        objective_func=improved_objective_func,
        n_trials=20,
    )
    if not is_being_graded()
    else optuna.load_study(study_name=new_study_name, storage=storage)
)
print("Best MAE", study_2.best_value)
print("Best params:", study_2.best_trial.params)
```

```
[I 2025-11-21 17:33:39,963] A new study created in RDB with name: lgbm-wine-2
/tmp/ipykernel_4744/1070365039.py:59: ExperimentalWarning:

MLflowCallback is experimental (supported from v1.4.0). The interface can change in the future.

[I 2025-11-21 17:33:40,915] Trial 0 finished with value: 0.418099151103134 and parameters: {'learning_rate': 0.0
296790042662690 8, 'subsample': 0.46364572682407335, 'colsample_bytree': 0.4107961679403244, 'min_child_samples':
20, 'num_leaves': 181}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:41,659] Trial 1 finished with value: 0.4751654415962778 and parameters: {'learning_rate': 0.
014124115025060347, 'subsample': 0.4778214378844623, 'colsample_bytree': 0.7391963650868432, 'min_child_samples'
: 94, 'num_leaves': 209}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:42,200] Trial 2 finished with value: 0.6685091743119267 and parameters: {'learning_rate': 0.
0077055940127295905, 'subsample': 0.07613762547568977, 'colsample_bytree': 0.8197056874649611, 'min_child_sample
s': 94, 'num_leaves': 596}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:43,067] Trial 3 finished with value: 0.4610050848661109 and parameters: {'learning_rate': 0.
005293661964364217, 'subsample': 0.48645943347289744, 'colsample_bytree': 0.7994655844802532, 'min_child_samples
': 39, 'num_leaves': 227}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:43,656] Trial 4 finished with value: 0.49917427306117207 and parameters: {'learning_rate': 0
.008314019514903102, 'subsample': 0.186909009331792, 'colsample_bytree': 0.6148538589793427, 'min_child_samples'
: 70, 'num_leaves': 304}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:44,561] Trial 5 finished with value: 0.5115058310013029 and parameters: {'learning_rate': 0.
02727183127344505, 'subsample': 0.11277223729341883, 'colsample_bytree': 0.47528678912113087, 'min_child_samples
': 61, 'num_leaves': 419}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:45,287] Trial 6 finished with value: 0.5489132508382163 and parameters: {'learning_rate': 0.
04409771327642757, 'subsample': 0.1398532019712619, 'colsample_bytree': 0.608540663048167, 'min_child_samples':
93, 'num_leaves': 132}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:45,821] Trial 7 finished with value: 0.6685091743119267 and parameters: {'learning_rate': 0.
026948022473899044, 'subsample': 0.1267358556592812, 'colsample_bytree': 0.3390309557911677, 'min_child_samples'
: 143, 'num_leaves': 776}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:46,532] Trial 8 finished with value: 0.5101429875313362 and parameters: {'learning_rate': 0.
04703026275515195, 'subsample': 0.187076196128016 8, 'colsample_bytree': 0.35860326840383033, 'min_child_samples'
: 106, 'num_leaves': 408}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:47,916] Trial 9 finished with value: 0.5130105312528986 and parameters: {'learning_rate': 0.
007013239683834226, 'subsample': 0.2728296095500716, 'colsample_bytree': 0.320633112669131, 'min_child_samples':
138, 'num_leaves': 281}. Best is trial 0 with value: 0.418099151103134.
[I 2025-11-21 17:33:49,302] Trial 10 finished with value: 0.4139426194377201 and parameters: {'learning_rate': 0
.07162543108574987, 'subsample': 0.39339130814726386, 'colsample_bytree': 0.48461652687015744, 'min_child_sample
s': 12, 'num_leaves': 556}. Best is trial 10 with value: 0.4139426194377201.
[I 2025-11-21 17:33:50,574] Trial 11 finished with value: 0.41930897382936166 and parameters: {'learning_rate':
0.06345691406358016, 'subsample': 0.3933700555179675, 'colsample_bytree': 0.4685338318254222, 'min_child_samples
': 14, 'num_leaves': 565}. Best is trial 10 with value: 0.4139426194377201.
[I 2025-11-21 17:33:51,773] Trial 12 finished with value: 0.4241990561025514 and parameters: {'learning_rate': 0
.07876775180209496, 'subsample': 0.38630580317911456, 'colsample_bytree': 0.4715125850670594, 'min_child_samples
': 14, 'num_leaves': 572}. Best is trial 10 with value: 0.4139426194377201.
[I 2025-11-21 17:33:52,512] Trial 13 finished with value: 0.45196185763965346 and parameters: {'learning_rate':
0.01896827079002305, 'subsample': 0.39216052817180147, 'colsample_bytree': 0.5268174298914856, 'min_child_sample
s': 40, 'num_leaves': 706}. Best is trial 10 with value: 0.4139426194377201.
[I 2025-11-21 17:33:53,368] Trial 14 finished with value: 0.45613184916949506 and parameters: {'learning_rate':
0.03842607712827143, 'subsample': 0.3269790393217251, 'colsample_bytree': 0.38696836364387566, 'min_child_sample
s': 34, 'num_leaves': 478}. Best is trial 10 with value: 0.4139426194377201.
[I 2025-11-21 17:33:54,186] Trial 15 finished with value: 0.4587917080467418 and parameters: {'learning_rate': 0
.013204339310547044, 'subsample': 0.4444135400172214, 'colsample_bytree': 0.6868767216042064, 'min_child_samples
': 53, 'num_leaves': 102}. Best is trial 10 with value: 0.4139426194377201.
[I 2025-11-21 17:33:55,247] Trial 16 finished with value: 0.4424104686723544 and parameters: {'learning_rate': 0
.03190703659919232, 'subsample': 0.3143032666707738, 'colsample_bytree': 0.5354395381865809, 'min_child_samples'
: 24, 'num_leaves': 667}. Best is trial 10 with value: 0.4139426194377201.
[I 2025-11-21 17:33:56,544] Trial 17 finished with value: 0.4009620843773491 and parameters: {'learning_rate': 0
.06126895717482317, 'subsample': 0.4366417186922958, 'colsample_bytree': 0.4114910633255542, 'min_child_samples'
: 12, 'num_leaves': 494}. Best is trial 17 with value: 0.4009620843773491.
[I 2025-11-21 17:33:57,267] Trial 18 finished with value: 0.491691248532112 and parameters: {'learning_rate': 0.
06144100971856799, 'subsample': 0.34305256896903474, 'colsample_bytree': 0.54912396981798, 'min_child_samples':
121, 'num_leaves': 505}. Best is trial 17 with value: 0.4009620843773491.
[I 2025-11-21 17:33:58,251] Trial 19 finished with value: 0.45021708119280673 and parameters: {'learning_rate':
0.07818564189938625, 'subsample': 0.43017554203830133, 'colsample_bytree': 0.8815619112292309, 'min_child_sample
s': 51, 'num_leaves': 346}. Best is trial 17 with value: 0.4009620843773491.
Best MAE 0.4009620843773491
Best params: {'learning_rate': 0.06126895717482317, 'subsample': 0.4366417186922958, 'colsample_bytree': 0.41149
10633255542, 'min_child_samples': 12, 'num_leaves': 494}
```

Example output:

```
Best MAE 0.386448234091213
Best params: {'learning_rate': 0.020120960308154467, 'subsample': 0.9215993238868946,
'colsample_bytree': 0.825439643398257, 'min_child_samples': 10, 'num_leaves': 855}
```

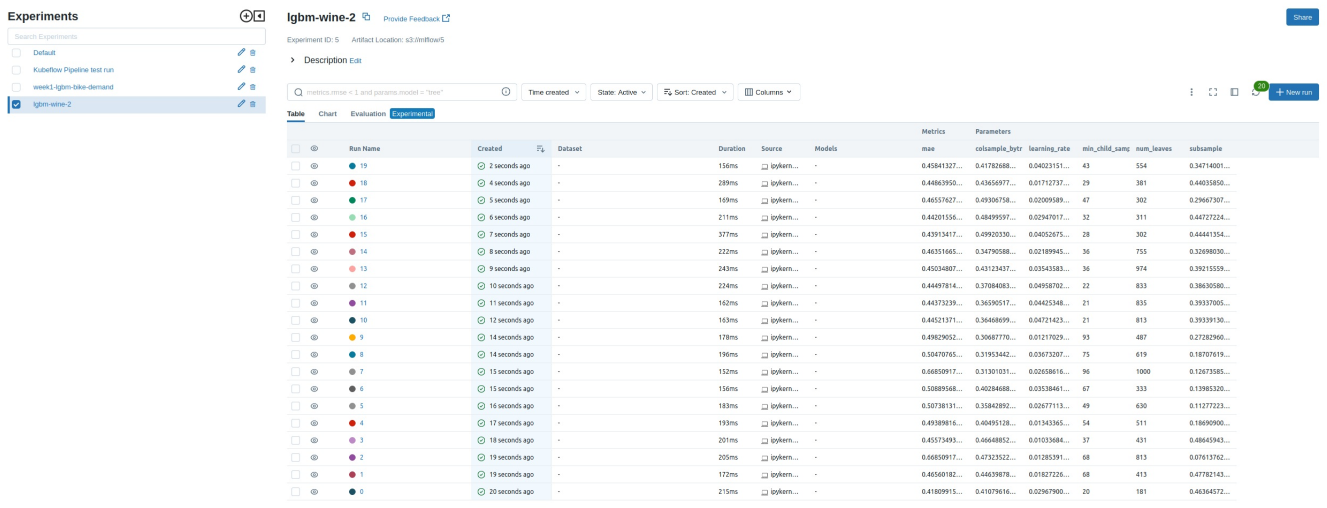
```python
# New MAE should be smaller than the previous one
assert study_2.best_value < study_1.best_value, "The new MAE should be smaller than the previous one"

# The new study should have 20 trials
```

```python
assert len(study_2.trials) == 20, "Wrong number of trials"

# The first trial should be the same as the best trial of the previous study
assert study_2.trials[0].values[0] == study_1.best_value

# The ranges of at least one of the three most important hyperparameters should be changed
param_changed = False
for param in important_hyperparams:
    new_distribution = study_2.best_trial.distributions.get(param)
    old_distribution = study_1.best_trial.distributions.get(param)
    if new_distribution.low != old_distribution.low or new_distribution.high != old_distribution.high:
        param_changed = True
        break
assert param_changed, "The ranges of at least one of the three most important hyperparameters should be changed"
```

Screenshot for Assignment 2c

Capture a screenshot of the trails of the "lgbm-wine-2" study in MLflow UI. Please include the metrics and parameters of each trial in your screenshot.

▶ Example:



lgbm-wine-2

Experiment ID: 5 Artifact Location: s3://mlflow/5

metrics.rmse < 1 and params.model = "tree"

					Metrics	Parameters				
Run Name	Created	Dataset	Duration	Source	Models	mae	colsample_bytr	learning_rate	min_child_samp	num_leaves	subsample
19	2 seconds ago	-	156ms	ipykern...	-	0.45841327...	0.41782688...	0.04023151...	43	554	0.34714001...
18	4 seconds ago	-	289ms	ipykern...	-	0.44863950...	0.43656977...	0.01712737...	29	381	0.44035850...
17	5 seconds ago	-	169ms	ipykern...	-	0.46557627...	0.49306758...	0.02009589...	47	302	0.29667307...
16	6 seconds ago	-	211ms	ipykern...	-	0.44201556...	0.48499597...	0.02947017...	32	311	0.44727224...
15	7 seconds ago	-	377ms	ipykern...	-	0.43913417...	0.49920330...	0.04052675...	28	302	0.44441354...
14	8 seconds ago	-	222ms	ipykern...	-	0.46351665...	0.34790588...	0.02189945...	36	755	0.32698030...
13	9 seconds ago	-	243ms	ipykern...	-	0.45034807...	0.43123437...	0.03543583...	36	974	0.39215559...
12	10 seconds ago	-	224ms	ipykern...	-	0.44497814...	0.37084083...	0.04958702...	22	833	0.38630580...
11	11 seconds ago	-	162ms	ipykern...	-	0.44373239...	0.36590517...	0.04425348...	21	835	0.39337005...
10	12 seconds ago	-	163ms	ipykern...	-	0.44521371...	0.36468699...	0.04721423...	21	813	0.39339130...
9	14 seconds ago	-	178ms	ipykern...	-	0.49829052...	0.30687770...	0.01217029...	93	487	0.27282960...
8	14 seconds ago	-	196ms	ipykern...	-	0.50470765...	0.31953442...	0.03673207...	75	619	0.18707619...
7	15 seconds ago	-	152ms	ipykern...	-	0.66850917...	0.31301031...	0.02658616...	96	1000	0.12673585...
6	15 seconds ago	-	156ms	ipykern...	-	0.50889568...	0.40284688...	0.03538461...	67	333	0.13985320...
5	16 seconds ago	-	183ms	ipykern...	-	0.50738131...	0.35842892...	0.02677113...	49	630	0.11277223...
4	17 seconds ago	-	193ms	ipykern...	-	0.49389816...	0.40495128...	0.01343365...	54	511	0.18690900...
3	18 seconds ago	-	201ms	ipykern...	-	0.45573493...	0.46648852...	0.01033684...	37	431	0.48645943...
2	19 seconds ago	-	205ms	ipykern...	-	0.66850917...	0.47323522...	0.01285391...	68	813	0.07613762...
1	19 seconds ago	-	172ms	ipykern...	-	0.46560182...	0.44639878...	0.01827226...	68	413	0.47782143...
0	20 seconds ago	-	215ms	ipykern...	-	0.41809915...	0.41079616...	0.02967900...	20	181	0.46364572...

Assignment 3: More about MLflow (2 points)

3a) Find the MLflow run with the best hyperparameter combination

In assignment 2c), you found the best hyperparameter combination for your model. Now, your task is to complete the `find_best_run_id` function. The function receives an MLflow Experiment name as the argument and returns the best MLflow Run ID that resulted in the best hyperparameter combination. In other words, this function should find the MLflow Run with the smallest MAE.

You may find the following MLflow docs useful:

- How to retrieve an MLflow Experiment given an Experiment name?
- How to search MLflow Runs inside an MLflow Experiment?

```python
def find_best_run_id(mlflow_experiment_name: str) -> str:
    """
    Find the ID of the MLflow run with the smallest MAE
    Args:
        mlflow_experiment_name: The name of the MLflow experiment where the run should be found
    Return:
        An MLflow run ID
    """
    ### START CODE HERE
    experiment = mlflow.get_experiment_by_name(mlflow_experiment_name)
    best = mlflow.search_runs(
        experiment_ids = [experiment.experiment_id],
        order_by=["metrics.mae ASC"],
        max_results = 1,
        output_format="list"
    )
    return best[0].info.run_id
    ### END CODE HERE
```

```
# From your MLflow UI you can check whether the printed MLflow Run ID really produced the smallest MAE
if not is_being_graded():
    best_run_id = find_best_run_id(mlflow_experiment_name=new_study_name)
    print(f"The best MLflow Run ID is: {best_run_id}")
```

```
The best MLflow Run ID is: 96ab8244d92243c59789376f3acbb905
```

3b) Train a model using the best hyperparameter combination

Your task is to train a model using the best hyperparameter combination found in Assignment 2c). You also need to upload and register the model to MLflow, the model needs to be associated with the MLflow Run where the optimal hyperparameter combination was found.

For example, suppose running the "lgbm-wine-2" study of Assignment 2 created an MLflow Run 14 where the best hyperparameter combination was found, the model created in this assignment needs to be associated with MLflow Run 14, as shown below:

No description has been provided for this image

No description has been provided for this image

Hints:

- You may find the following function helpful: mlflow.start_run (Pay attention to the use of the `run_id` parameter).
- It would probably be more convenient to retrieve the best hyperparameter combination using `optuna.load_study` rather than from the Mlflow Run.

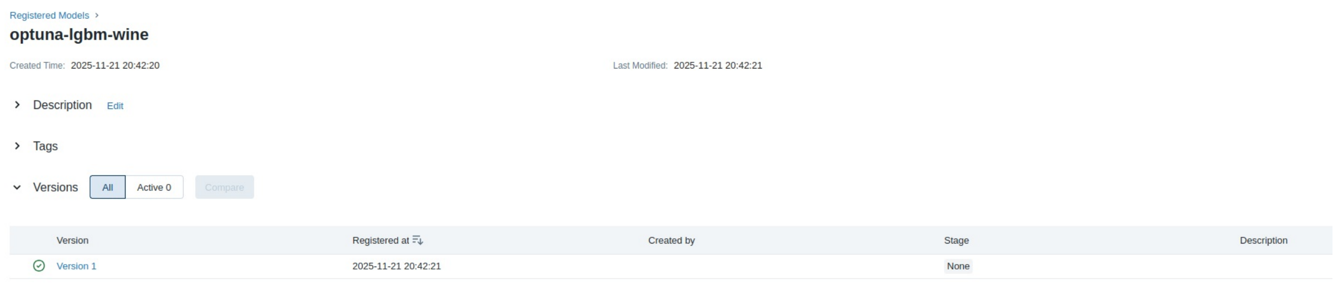
```
def train_optimized_model(study_name: str, storage: str, model_name: str):
    """
    Train a model with the best hyperparameters found by Optuna.
    Args:
        study_name: The name of the Optuna study where the best hyperparameters have been found. This is also the
        storage: The URI of the storage used to save the study history.
        model_name: The name of the registered MLflow model.
    """
    ### START CODE HERE
    study = optuna.load_study(study_name=study_name, storage=storage)
    best_params = study.best_params
    params = {
        'n_estimators': 1000,
        'subsample_freq': 1,
        'random_state': RANDOM_SEED,
        'verbose': -1,
        **best_params # Overwrites any common keys if present, but here it's fine
    }
    best_run_id = find_best_run_id(mlflow_experiment_name=study_name)
    with mlflow.start_run(run_id = best_run_id):
        optimized_model = lgb.LGBMRegressor(**params)
        optimized_model.fit(train_x,train_y)
        predictions = optimized_model.predict(test_x)
        mae = mean_absolute_error(test_y,predictions)

        mlflow.lightgbm.log_model(
            lgb_model = optimized_model,
            artifact_path="model",
            registered_model_name=model_name
        )
    ### END CODE HERE
```

```
# From your MLflow UI you can make sure that the registered model is associated with the
if not is_being_graded():
    train_optimized_model(study_name=new_study_name, storage=storage, model_name="optuna-lgbm-wine")
```

```
Successfully registered model 'optuna-lgbm-wine'.
2025/11/21 20:42:21 INFO mlflow.store.model_registry.abstract_store: Waiting up to 300 seconds for model version
to finish creation. Model name: optuna-lgbm-wine, version 1
Created version '1' of model 'optuna-lgbm-wine'.
```

Screenshots for Assignment 3b)

Submit the screenshots of the registered model version info and the corresponding MLflow run. You can take the images in the assignment instructions above as an example.


Registered Models ›

optuna-lgbm-wine

Created Time: 2025-11-21 20:42:20 Last Modified: 2025-11-21 20:42:21

› Description Edit

› Tags

⌄ Versions All Active 0 Compare

Version	Registered at	Created by	Stage	Description
Version 1	2025-11-21 20:42:21		None	

Assignment 4: Conditional search space with Optuna (2 points)

Complete the objective function `objective_func_multimodel`. The target is to optimize the hyperparameters of a LightGBM and an XGBoost regression model for the red wine quality prediction use case and see which model type is better in the use case. Similar to the previous assignments, the objective is also a smaller MAE. The hyperparameters to be tuned and their search ranges are given below. Use TPESampler with `RANDOM_SEED` as the random seed.

LightGBM:

Hyperparameter	Explanation	type	range
n_estimators	The number of decision trees.	integer	1000 (fixed value)
learning_rate	The step size of the gradient descent. It controls how quickly the model fits and then overfits the training data.	float	[0.001, 0.1] (sampled from the logarithmic domain)
subsample	The percentage of training samples to be used to train each tree. `subsample*100%` of the training samples will be randomly selected for training.	float	[0.05, 1.0]
subsample_freq	Subsampling frequency. The subsampling will be performed again after `subsample_freq` trees have been trained.	integer	1 (fixed value)
colsample_bytree	The percentage of features to use when training each tree.	float	[0.05, 1.0]
min_child_samples	A leaf node should have at least `min_child_samples` data points to be further splitted.	integer	[1, 100]
num_leaves	Max number of nodes in a single tree.	integer	[2, 2^10]

| random_state | The seed for random number generation for reproducibility. | integer | RANDOM_SEED (fixed value)

XGBoost

Hyperparameter	Explanation	type	range
n_estimators	Same as LightGBM.	integer	1000 (fixed value)

learning_rate	Same as LightGBM.	float	[0.001, 0.1] (sampled from the logarithmic domain)
subsample	Same as LightGBM.	float	[0.05, 1.0]
colsample_bytree	Same as LightGBM.	float	[0.05, 1.0]
min_child_weight	Minimum sum of instance weight needed for a leaf mode to be further splitted (similar to min_child_samples in LightGBN).	integer	[1, 100]
max_depth	Max depth of a single tree	integer	[1, 10]

| random_state | Same as LightGBM. | integer | RANDOM_SEED (fixed value)

Notes:

- When specifying the search ranges for the model type, please use **"model_type"** as the parameter name that indicates the model type and `["lgbm", "xgb"]` as the value candidates.
- Some of the hyperparameters to be optimized used by the LightGBM and XGBoost models share the same names. In the assignment, these hyperparameters are `learning_rate, subsample, colsample_bytree`. You need to give these hyperparameters a unique name when defining their search ranges using `trial.suggest_*` so that Optuna can properly optimize the correct hyperparameters for each different model. Please rename the hyperparameters as `<hyperparameter-name>_lgbm` for the LightGBM model and `<hyperparameter-name>_xgb` for the XGBoost model, such as `learning_rate_lgbm` and `learning_rate_xgb`.

```python
def objective_func_multimodel(trial):
    ### START CODE HERE
    model_type = trial.suggest_categorical('model_type', ['lgbm', 'xgb'])

    if model_type == 'lgbm':
        params = {
            'n_estimators': 1000,
            'random_state': RANDOM_SEED,
            'learning_rate': trial.suggest_float('learning_rate_lgbm', 0.001, 0.1, log=True),
            'subsample': trial.suggest_float('subsample_lgbm', 0.05, 1.0),
            'subsample_freq': 1,
            'colsample_bytree': trial.suggest_float('colsample_bytree_lgbm', 0.05, 1.0),
            'min_child_samples': trial.suggest_int('min_child_samples', 1, 100),
            'num_leaves': trial.suggest_int('num_leaves', 2, 1024),
            'verbose': -1
        }
        model = lgb.LGBMRegressor(**params)
        model.fit(train_x, train_y)
    elif model_type == 'xgb':
        params = {
            'n_estimators': 1000,
            'random_state': RANDOM_SEED,
            'learning_rate': trial.suggest_float('learning_rate_xgb', 0.001, 0.1, log=True),
            'subsample': trial.suggest_float('subsample_xgb', 0.05, 1.0),
            'colsample_bytree': trial.suggest_float('colsample_bytree_xgb', 0.05, 1.0),
            'min_child_weight': trial.suggest_int('min_child_weight', 1, 100),
            'max_depth': trial.suggest_int('max_depth', 1, 10),
            'objective': 'reg:squarederror'
        }
        model = xgb.XGBRegressor(**params)
        model.fit(train_x, train_y)

    pred = model.predict(test_x)
    mae = mean_absolute_error(test_y, pred)

    return mae
    ### END CODE HERE
```

```python
multimodel_study_name = "multimodel-wine"
multimodel_study = run_study(
    study_name=multimodel_study_name,
    storage=None,
    objective_func=objective_func_multimodel,
    n_trials=10,
)

print("Best MAE", multimodel_study.best_value)
print("Best params:", multimodel_study.best_trial.params)
```

```
[I 2025-11-21 20:51:24,024] A new study created in memory with name: multimodel-wine
[I 2025-11-21 20:51:24,587] Trial 0 finished with value: 0.49022297143936155 and parameters: {'model_type': 'xgb
', 'learning_rate_xgb': 0.029106359131330698, 'subsample_xgb': 0.6187255599871848, 'colsample_bytree_xgb': 0.198
21770842031466, 'min_child_weight': 16, 'max_depth': 1}. Best is trial 0 with value: 0.49022297143936155.
[I 2025-11-21 20:51:24,680] Trial 1 finished with value: 0.6685091743119267 and parameters: {'model_type': 'lgbm
', 'learning_rate_lgbm': 0.02607024758370768, 'subsample_lgbm': 0.06955526958101232, 'colsample_bytree_lgbm': 0.
9714143595538947, 'min_child_samples': 84, 'num_leaves': 219}. Best is trial 0 with value: 0.49022297143936155.
[I 2025-11-21 20:51:25,362] Trial 2 finished with value: 0.46709711909294127 and parameters: {'model_type': 'xgb
', 'learning_rate_xgb': 0.0040596116104843075, 'subsample_xgb': 0.548518610050626, 'colsample_bytree_xgb': 0.460
34776771000996, 'min_child_weight': 30, 'max_depth': 7}. Best is trial 2 with value: 0.46709711909294127.
[I 2025-11-21 20:51:25,968] Trial 3 finished with value: 0.4546022093296051 and parameters: {'model_type': 'xgb'
, 'learning_rate_xgb': 0.005404103854647328, 'subsample_xgb': 0.4832664850061841, 'colsample_bytree_xgb': 0.7959
171633233629, 'min_child_weight': 20, 'max_depth': 6}. Best is trial 3 with value: 0.4546022093296051.
[I 2025-11-21 20:51:26,543] Trial 4 finished with value: 0.5096934247165656 and parameters: {'model_type': 'lgbm
', 'learning_rate_lgbm': 0.016409286730647923, 'subsample_lgbm': 0.21199791750292696, 'colsample_bytree_lgbm': 0
.11179901333601554, 'min_child_samples': 95, 'num_leaves': 989}. Best is trial 3 with value: 0.4546022093296051.
[I 2025-11-21 20:51:27,810] Trial 5 finished with value: 0.4859459510725019 and parameters: {'model_type': 'lgbm
', 'learning_rate_lgbm': 0.0015679933916723015, 'subsample_lgbm': 0.700021375186549, 'colsample_bytree_lgbm': 0.
4681448690526212, 'min_child_samples': 13, 'num_leaves': 508}. Best is trial 3 with value: 0.4546022093296051.
[I 2025-11-21 20:51:28,413] Trial 6 finished with value: 0.49382792234420775 and parameters: {'model_type': 'xgb
', 'learning_rate_xgb': 0.0032927591344236173, 'subsample_xgb': 0.6793961701362828, 'colsample_bytree_xgb': 0.34
612552228494037, 'min_child_weight': 53, 'max_depth': 6}. Best is trial 3 with value: 0.4546022093296051.
[I 2025-11-21 20:51:29,271] Trial 7 finished with value: 0.43266252517700193 and parameters: {'model_type': 'xgb
', 'learning_rate_xgb': 0.035503048581283086, 'subsample_xgb': 0.9425239944859797, 'colsample_bytree_xgb': 0.900
0859829062664, 'min_child_weight': 60, 'max_depth': 10}. Best is trial 7 with value: 0.43266252517700193.
[I 2025-11-21 20:51:30,110] Trial 8 finished with value: 0.5370481526851654 and parameters: {'model_type': 'xgb'
, 'learning_rate_xgb': 0.0012315571723666018, 'subsample_xgb': 0.3590638142251011, 'colsample_bytree_xgb': 0.419
2434252050079, 'min_child_weight': 28, 'max_depth': 9}. Best is trial 7 with value: 0.43266252517700193.
[I 2025-11-21 20:51:30,918] Trial 9 finished with value: 0.42899067751612513 and parameters: {'model_type': 'lgb
m', 'learning_rate_lgbm': 0.012172847081122434, 'subsample_lgbm': 0.18387801372602453, 'colsample_bytree_lgbm':
0.8120871317163377, 'min_child_samples': 8, 'num_leaves': 1011}. Best is trial 9 with value: 0.42899067751612513
.
Best MAE 0.42899067751612513
Best params: {'model_type': 'lgbm', 'learning_rate_lgbm': 0.012172847081122434, 'subsample_lgbm': 0.183878013726
02453, 'colsample_bytree_lgbm': 0.8120871317163377, 'min_child_samples': 8, 'num_leaves': 1011}
```

Example output:

```
Best MAE 0.42899067751612513
Best params: {'model_type': 'lgbm', 'learning_rate_lgbm': 0.012172847081122434, 'subsample_lgbm':
0.18387801372602453, 'colsample_bytree_lgbm': 0.8120871317163377, 'min_child_samples': 8,
'num_leaves': 1011}
```

Assignment 5: Ensemble averaging (2 points)

Complete the objective function `objective_func_ensemble`. The purpose is to Optuna to find the optimal weight combination to combine the predictions of an Sklearn's RandomForest, a XGBoost, and a LightGBM model to obtain better predictions for the red wine quality us case. Detailed instructions are given below.

Before going to the assignment, first train three models for the red wine quality use case using Sklearn's RandomForest, XGBoost, and LightGBM and evaluate MAE of each model. For simplicity, the default configurations of the hyperparameters are used except for "random_state" which is set as `RANDOM_SEED` as in the previous assignments.

```python
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(random_state=RANDOM_SEED)
xgb_model = xgb.XGBRegressor(random_state=RANDOM_SEED)
lgbm_model = lgb.LGBMRegressor(random_state=RANDOM_SEED, verbose=-1)

model_names = ["rf_model", "xgb_model", "lgbm_model"]

for name in model_names:
    model = eval(name)
    model.fit(train_x, train_y)
    predictions = model.predict(test_x)
    print(f"{name}: {mean_absolute_error(test_y, predictions)}")
```

```
rf_model: 0.42284999999999995
xgb_model: 0.42444213390350344
lgbm_model: 0.43465539993381924
```

You can see the best performing model is the random forest one, with an MAE of 0.4228.

Now, it is time to combine the predictions of these models to improve the final prediction. In this assignment, your task is to complete the objective function `objective_func_ensemble`. The target is to find the best weight combination for the three models that have been trained in the cell above to obtain smaller MAE. Use 3 as the step of discretization when searching weight for each model. The search range of the weight should be [1, 100]. Use TPESampler (with `RANDOM_SEED` as the seed) in the study.

Note: Please use "rf_model", "xgb_model", and "lgbm_model" as the names of the parameters that specify the weights of the random forest, XGBoost, and LightGBM models, respectively. E.g.,

```
study = run_study(study_name="ensemble-wine", storage=None, objective_func=objective_func_ensemble,
n_trials=10)
print(study.best_trial.params)
# Example_output:
# {'rf_model': 46, 'xgb_model': 79, 'lgbm_model': 19}
```

```
all_predictions = {name: (eval(name)).predict(test_x) for name in model_names}

# Define the objective function
def objective_func_ensemble(trial):
    ### START CODE HERE
    rf_weight = trial.suggest_int("rf_model", 1, 100, step=3)
    xgb_weight = trial.suggest_int("xgb_model", 1, 100, step=3)
    lgbm_weight = trial.suggest_int("lgbm_model", 1, 100, step=3)

    weight = rf_weight + xgb_weight + lgbm_weight

    predictions = np.zeros_like(test_y,dtype=float)

    predictions += rf_weight * all_predictions['rf_model']
    predictions += xgb_weight * all_predictions['xgb_model']
    predictions += lgbm_weight * all_predictions['lgbm_model']

    predictions /= weight
    mae = mean_absolute_error(test_y,predictions)
    return mae
    ### END CODE HERE
```

```
study_ensemble = run_study(study_name="ensemble-wine", storage=None, objective_func=objective_func_ensemble, n_tr

print("Best MAE", study_ensemble.best_value)
print("Best params:", study_ensemble.best_trial.params)
```

```
[I 2025-11-21 21:11:19,219] A new study created in memory with name: ensemble-wine
[I 2025-11-21 21:11:19,235] Trial 0 finished with value: 0.41652429555863707 and parameters: {'rf_model': 37, 'x
gb_model': 97, 'lgbm_model': 73}. Best is trial 0 with value: 0.41652429555863707.
[I 2025-11-21 21:11:19,241] Trial 1 finished with value: 0.4167632784253199 and parameters: {'rf_model': 61, 'xg
b_model': 16, 'lgbm_model': 16}. Best is trial 0 with value: 0.41652429555863707.
[I 2025-11-21 21:11:19,251] Trial 2 finished with value: 0.41925149465179073 and parameters: {'rf_model': 4, 'xg
b_model': 88, 'lgbm_model': 61}. Best is trial 0 with value: 0.41652429555863707.
[I 2025-11-21 21:11:19,263] Trial 3 finished with value: 0.41989033789428 and parameters: {'rf_model': 73, 'xgb_
model': 1, 'lgbm_model': 97}. Best is trial 0 with value: 0.41652429555863707.
[I 2025-11-21 21:11:19,273] Trial 4 finished with value: 0.41691708988720605 and parameters: {'rf_model': 85, 'x
gb_model': 22, 'lgbm_model': 19}. Best is trial 0 with value: 0.41652429555863707.
[I 2025-11-21 21:11:19,278] Trial 5 finished with value: 0.4183727940734564 and parameters: {'rf_model': 19, 'xg
b_model': 31, 'lgbm_model': 52}. Best is trial 0 with value: 0.41652429555863707.
[I 2025-11-21 21:11:19,284] Trial 6 finished with value: 0.41669817924382635 and parameters: {'rf_model': 43, 'x
gb_model': 28, 'lgbm_model': 61}. Best is trial 0 with value: 0.41652429555863707.
[I 2025-11-21 21:11:19,287] Trial 7 finished with value: 0.4180165095492431 and parameters: {'rf_model': 13, 'xg
b_model': 28, 'lgbm_model': 37}. Best is trial 0 with value: 0.41652429555863707.
[I 2025-11-21 21:11:19,292] Trial 8 finished with value: 0.41523636202081865 and parameters: {'rf_model': 46, 'x
gb_model': 79, 'lgbm_model': 19}. Best is trial 8 with value: 0.41523636202081865.
[I 2025-11-21 21:11:19,299] Trial 9 finished with value: 0.4154842183867582 and parameters: {'rf_model': 52, 'xg
b_model': 61, 'lgbm_model': 4}. Best is trial 8 with value: 0.41523636202081865.
Best MAE 0.41523636202081865
Best params: {'rf_model': 46, 'xgb_model': 79, 'lgbm_model': 19}
```

Example output:

```
Best MAE 0.41523636202081865
Best params: {'rf_model': 46, 'xgb_model': 79, 'lgbm_model': 19}
```

```
assert (
    study_ensemble.best_value < 0.4228
), "The MAE should be smaller than 0.4228 (the MAE of the best single model)"

assert set(study_ensemble.best_trial.params.keys()) == set(["rf_model", "xgb_model", "lgbm_model"]), "Incorret pa
```

Assignment 6: Scale up Optuna using Ray Tune

In this assignment, you need to parallelize the Optuna hyperparameter optimization process using Ray Tune. Specifically, you will do this in three steps:

1. Complete the `trainable` function to define a Trainable. You need to specify the model, the model training process, and the metric to be optimized. Similar to Assignment 1, the model is an LightGBM regressor, and the metric to be optimized is MAE.
2. Complete the `create_search_algo` function to define search spaces and the search algorithm. Use OptunaSearch here. The search spaces should be the same as Assignment 1. Use the TPESampler as you did in Assignment 1.
3. Complete the `create_ray_tuner` function to define a Tuner that launches hyperparameter optimization trials. This function receives a Boolean argument named `parallel`. If `parallel` is set to False, the Tuner should perform the hyperparameter optimization trials one by one. Otherwise, the Tuner should use all available CPUs on your machine to run the trials concurrently. You'll find the details of the arguments passed to the function in the function docs.

Notes:

- Remember to specify the random seed (`RANDOM_SEED`) for your model and sampler.
- When define the Trainable, please use **"mae"** as the metric name when report the metric to the Tuner.
- The actual number of the concurrent jobs doesn't matter (as long as it's larger than 1) as it depends on the number of CPUs available on your machine.

```python
def trainable(config: Dict):
    """
    Defines a model using the given configuration, trains it, and report the metric.
    Args:
        config: A dictionary containing the hyperparameters to be tuned
    """
    ### START CODE HERE
    params = {
        'n_estimators': 1000,
        'learning_rate': config['learning_rate'],
        'subsample': config['subsample'],
        'subsample_freq': 1,
        'colsample_bytree': config['colsample_bytree'],
        'min_child_samples': config['min_child_samples'],
        'num_leaves': config['num_leaves'],
        'random_state': RANDOM_SEED,
        'verbose': -1
    }
    model = lgb.LGBMRegressor(**params)
    model.fit(train_x, train_y)
    predictions = model.predict(test_x)
    mae = mean_absolute_error(test_y, predictions)
    tune.report(mae=mae)
    ### END CODE HERE


def create_search_algo() -> OptunaSearch:
    """
    Create an Optuna search algorithm.
    """
    ### START CODE HERE
    sampler = TPESampler(seed=RANDOM_SEED)

    algo = OptunaSearch(
        sampler=sampler,
        metric="mae",
        mode="min",
    )
    return algo
    ### END CODE HERE


def create_ray_tuner(
    trainable: callable, algo: OptunaSearch, n_trials: int, parallel: bool = False
) -> tune.Tuner:
    """
    Create a Ray Tune Tuner
    Args:
        trainable: The Trainable that specifies the objective.
        algo: The search algorithm.
        n_trials: The number of trials.
        parallel: Whether to run the trials in parallel.
    """
    ### START CODE HERE
    ray.init(ignore_reinit_error=True)
    search_space = {
        'learning_rate': tune.loguniform(0.001, 0.1),
        'subsample': tune.uniform(0.05, 0.5),
        'colsample_bytree': tune.uniform(0.05, 0.5),
        'min_child_samples': tune.randint(20, 101),
        'num_leaves': tune.randint(2, 1025),
    }

    # parallel=False → 串行：一次只能跑 1 个 trial
    # parallel=True  → 并行：使用所有 CPU
    max_concurrent = 1 if not parallel else None

    tune_config = tune.TuneConfig(
        num_samples=n_trials,
        search_alg=algo,
        metric="mae",
        mode="min",
        max_concurrent_trials=max_concurrent,
    )

    tuner = tune.Tuner(
        trainable,
        param_space=search_space,
        tune_config=tune_config,
    )

    return tuner
    ### END CODE HERE
```

```
# When the Tuner is running trials sequentially, the best hyperparameters should be the same as the ones found by
tuner = create_ray_tuner(trainable=trainable, algo=create_search_algo(), n_trials=10, parallel=False)
results = tuner.fit()
original_optuna_study = run_study(study_name="original", storage=None, objective_func=objective_func, n_trials=10
assert results.get_best_result(metric="mae", mode="min").metrics.get("mae") == original_optuna_study.best_value,
```

Tune Status

Current time:	2025-11-21 21:52:31
Running for:	00:01:30.49
Memory:	3.3/8.4 GiB

System Info

Using FIFO scheduling algorithm.
Logical resource usage: 1.0/4 CPUs, 0/0 GPUs

Messages

Number of errored trials: 10

Trial name	# failures	
trainable_b3f956ea	1	/home/user/ray_results/trainable_2
trainable_e95a96f6	1	/home/user/ray_results/trainable_2
trainable_0edab112	1	/home/user/ray_results/trainable_20
trainable_ae48cefc	1	/home/user/ray_results/trainable_2
trainable_7129cf42	1	/home/user/ray_results/trainable_2
trainable_9ca96cc1	1	/home/user/ray_results/trainable_
trainable_a868c1de	1	/home/user/ray_results/trainable_20
trainable_5a99d9a0	1	/home/user/ray_results/trainable_20
trainable_616fff36	1	/home/user/ray_results/trainable_
trainable_374ab9b2	1	/home/user/ray_results/trainable_202

Trial Status

Trial name	status	loc	colsample_bytree	learning_rate	min_child_samples	num_leaves	subsamp
trainable_b3f956ea	ERROR	10.0.2.15:23360	0.379397	0.00561152	68	161	0.4778
trainable_e95a96f6	ERROR	10.0.2.15:23413	0.439779	0.00205111	68	726	0.07613
trainable_0edab112	ERROR	10.0.2.15:23457	0.424599	0.00109943	37	188	0.4864
trainable_ae48cefc	ERROR	10.0.2.15:23498	0.28614	0.00232707	54	299	0.1869
trainable_7129cf42	ERROR	10.0.2.15:23544	0.181465	0.0167381	49	468	0.1127
trainable_9ca96cc1	ERROR	10.0.2.15:23584	0.281405	0.0371836	67	49	0.1398
trainable_a868c1de	ERROR	10.0.2.15:23623	0.0792732	0.0164093	96	989	0.1267
trainable_5a99d9a0	ERROR	10.0.2.15:23669	0.0939525	0.0413804	75	452	0.1870
trainable_616fff36	ERROR	10.0.2.15:23705	0.0654748	0.00175419	93	266	0.272
trainable_374ab9b2	ERROR	10.0.2.15:23743	0.284031	0.0211371	64	191	0.190

```
2025-11-21 21:51:20,187 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_b3f956ea
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23360, ip=10.0.2.15, actor_id=44
82e0b3a71bb760d86c77a001000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
```

```
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:51:30,391 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_e95a96f6
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23413, ip=10.0.2.15, actor_id=f8
8af5dcde7577b6d104c75001000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:51:40,024 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_0edab112
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23457, ip=10.0.2.15, actor_id=0d
5e3dbb4c1e5312c69bdbaa01000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:51:50,432 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_ae48cefc
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23498, ip=10.0.2.15, actor_id=06
e6155de388bdeac2a96faf01000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:51:59,665 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_7129cf42
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23544, ip=10.0.2.15, actor_id=f2
db854ca25fdb208cecb6e001000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:52:07,590 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_9ca96cc1
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23584, ip=10.0.2.15, actor_id=4f
6c79a28fa6d26d2f88276b01000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:52:17,290 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_a868c1de
```

```
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23623, ip=10.0.2.15, actor_id=13
005daac6d596cd0e9a069b01000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:52:21,269 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_5a99d9a0
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23669, ip=10.0.2.15, actor_id=36
fcfb99387d8f0d70b71dfa01000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:52:25,159 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_616fff36
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23705, ip=10.0.2.15, actor_id=2b
d7c5e2bf3b3d87cb6162e601000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:52:31,199 ERROR tune_controller.py:1374 -- Trial task failed for trial trainable_374ab9b2
Traceback (most recent call last):
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/execution/_internal/event_manager.p
y", line 110, in resolve_future
    result = ray.get(future)
             ^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/auto_init_hook.py", line 22, i
n auto_init_wrapper
    return fn(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/client_mode_hook.py", line 103
, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/_private/worker.py", line 2624, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(DeprecationWarning): ray::ImplicitFunc.train() (pid=23743, ip=10.0.2.15, actor_id=f8
6da9b7c704a86ecd6cd9a001000000, repr=trainable)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/trainable.py", line 342,
in train
    raise skipped from exception_cause(skipped)
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/air/_internal/util.py", line 88, in run
    self._ret = self._target(*self._args, **self._kwargs)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 115, in <lambda>
    training_func=lambda: self._trainable_func(self.config),
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/function_trainable.py",
line 332, in _trainable_func
    output = fn()
             ^^^^
  File "/tmp/ipykernel_4744/3555021276.py", line 23, in trainable
  File "/home/user/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/trainable/session.py", line 121, i
n report
    raise DeprecationWarning(_TUNE_REPORT_DEPRECATION_MSG)
DeprecationWarning: `tune.report` is deprecated.
Use `ray.train.report` instead -- see the example below:

from ray import tune     ->     from ray import train
tune.report(metric=1)    ->     train.report({'metric': 1})
2025-11-21 21:52:31,522 ERROR tune.py:1038 -- Trials did not complete: [trainable_b3f956ea, trainable_e95a96f6,
trainable_0edab112, trainable_ae48cefc, trainable_7129cf42, trainable_9ca96cc1, trainable_a868c1de, trainable_5a
99d9a0, trainable_616fff36, trainable_374ab9b2]
2025-11-21 21:52:31,546 INFO tune.py:1042 -- Total run time: 91.15 seconds (90.46 seconds for the tuning loop).
[I 2025-11-21 21:52:31,728] A new study created in memory with name: original
[I 2025-11-21 21:52:34,773] Trial 0 finished with value: 0.48305757514051956 and parameters: {'learning_rate': 0
.005611516415334507, 'subsample': 0.4778214378844623, 'colsample_bytree': 0.3793972738151323, 'min_child_samples
': 68, 'num_leaves': 161}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 21:52:35,631] Trial 1 finished with value: 0.6685091743119267 and parameters: {'learning_rate': 0.
002051110418843397, 'subsample': 0.07613762547568977, 'colsample_bytree': 0.4397792655987208, 'min_child_samples
': 68, 'num_leaves': 726}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 21:52:36,699] Trial 2 finished with value: 0.5330860406894041 and parameters: {'learning_rate': 0.
0010994335574766201, 'subsample': 0.48645943347289744, 'colsample_bytree': 0.4245991883601898, 'min_child_sample
s': 37, 'num_leaves': 188}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 21:52:37,476] Trial 3 finished with value: 0.5338040960963436 and parameters: {'learning_rate': 0.
002327067708383781, 'subsample': 0.186909009331792, 'colsample_bytree': 0.28614039423450705, 'min_child_samples'
: 54, 'num_leaves': 299}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 21:52:38,695] Trial 4 finished with value: 0.5076269460531386 and parameters: {'learning_rate': 0.
01673808578875214, 'subsample': 0.11277223729341883, 'colsample_bytree': 0.1814650918408482, 'min_child_samples'
: 49, 'num_leaves': 468}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 21:52:39,053] Trial 5 finished with value: 0.5076748898380093 and parameters: {'learning_rate': 0.
037183641805732096, 'subsample': 0.1398532019712619, 'colsample_bytree': 0.28140549728612524, 'min_child_samples
': 67, 'num_leaves': 49}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 21:52:39,469] Trial 6 finished with value: 0.6685091743119267 and parameters: {'learning_rate': 0.
016409286730647923, 'subsample': 0.1267358556592812, 'colsample_bytree': 0.0792732168433758, 'min_child_samples'
: 96, 'num_leaves': 989}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 21:52:39,676] Trial 7 finished with value: 0.5051750796616936 and parameters: {'learning_rate': 0.
041380401125610165, 'subsample': 0.1870761961280168, 'colsample_bytree': 0.09395245130287275, 'min_child_samples
': 75, 'num_leaves': 452}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 21:52:39,928] Trial 8 finished with value: 0.602185209067021 and parameters: {'learning_rate': 0.0
017541893487450805, 'subsample': 0.2728296095500716, 'colsample_bytree': 0.06547483450184828, 'min_child_samples
': 93, 'num_leaves': 266}. Best is trial 0 with value: 0.48305757514051956.
[I 2025-11-21 21:52:40,184] Trial 9 finished with value: 0.49454634175435186 and parameters: {'learning_rate': 0
.02113705944064573, 'subsample': 0.19026998424023495, 'colsample_bytree': 0.28403060953001485, 'min_child_sample
s': 64, 'num_leaves': 191}. Best is trial 0 with value: 0.48305757514051956.
2025-11-21 21:52:40,190 WARNING experiment_analysis.py:584 -- Could not find best trial. Did you pass the correc
t `metric` parameter?
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[61], line 5
      3 results = tuner.fit()
      4 original_optuna_study = run_study(study_name="original", storage=None, objective_func=objective_func, n_
trials=10)
----> 5 assert results.get_best_result(metric="mae", mode="min").metrics.get("mae") == original_optuna_study.bes
t_value, "Incorrect best hyperparameters"

File ~/anaconda3/envs/mlops/lib/python3.11/site-packages/ray/tune/result_grid.py:162, in ResultGrid.get_best_res
ult(self, metric, mode, scope, filter_nan_and_inf)
    151     error_msg = (
    152         "No best trial found for the given metric: "
    153         f"{metric or self._experiment_analysis.default_metric}. "
    154         "This means that no trial has reported this metric"
    155     )
    156     error_msg += (
    157         ", or all values reported for this metric are NaN. To not ignore NaN "
    158         "values, you can set the `filter_nan_and_inf` arg to False."
    159         if filter_nan_and_inf
    160         else "."
    161     )
--> 162     raise RuntimeError(error_msg)
    164 return self._trial_to_result(best_trial)

RuntimeError: No best trial found for the given metric: mae. This means that no trial has reported this metric,
or all values reported for this metric are NaN. To not ignore NaN values, you can set the `filter_nan_and_inf` a
rg to False.
```

```python
# Mock a slow training process
def slow_trainable(config):
    time.sleep(10)
    return ({"mae": random.random()})

tuner_parallelized = create_ray_tuner(trainable=slow_trainable, algo=create_search_algo(), n_trials=5, parallel=T
start_time = time.time()
results = tuner_parallelized.fit()
end_time = time.time()

assert end_time - start_time < 50, "It should take less than 50s to complete the optimization in the parallel mod
```

Tune Status

Current time:	2025-11-21 22:03:50
Running for:	00:00:36.88
Memory:	3.2/8.4 GiB

System Info

Using FIFO scheduling algorithm.
Logical resource usage: 1.0/4 CPUs, 0/0 GPUs

Trial Status

Trial name	status	loc	colsample_bytree	learning_rate	min_child_samples	num_leaves
slow_trainable_d2252c0f	TERMINATED	10.0.2.15:24316	0.379397	0.00561152	68	161
slow_trainable_0a05fbea	TERMINATED	10.0.2.15:24349	0.439779	0.00205111	68	726
slow_trainable_96d42ab0	TERMINATED	10.0.2.15:24316	0.424599	0.00109943	37	188
slow_trainable_3e28e8f3	TERMINATED	10.0.2.15:24349	0.28614	0.00232707	54	299
slow_trainable_d096c43f	TERMINATED	10.0.2.15:24316	0.181465	0.0167381	49	468

```
2025-11-21 22:03:50,491 INFO tune.py:1042 -- Total run time: 36.98 seconds (36.88 seconds for the tuning loop).
```

Wrap-up

Please include the following files in your submission:

- This Jupyter notebook (`week3_assignments.ipynb`)
- The "optuna.sqlite3" database file
- The PDF file containing your answers for Assignment 2b and screenshots for Assignments 2c and 3b

N.B. Before making your submission, please check that your notebook and database files are named **exactly** as specified here.