2023-01-20

Optuna + MLflow

MLOps Optuna MLflow

Introduction

This article presents an example of experiment management using a combination of Optuna and MLflow Tracking.

LightGBM example

Install the libraries.

```
$ pip install mlflow
$ pip install optuna
```

```
$ pip install scikit-learn
$ pip install lightgbm
```

Write the following LightGBM code in `mlflow_lightgbm.py` .

mlflow_lightgbm.py

```
import lightgbm as lgb
import numpy as np
import sklearn.datasets
import sklearn.metrics
from sklearn.model_selection import train_test_split
import optuna
import mlflow


def mlflow_callback(study: optuna.Study, trial: optuna.Trial):
    mlflow.set_experiment("LightGBM")
    trial_value = trial.value if trial.value is not None else float("nan")
    with mlflow.start_run(run_name=str(trial.number)):
        mlflow.log_params(trial.params)
        mlflow.log_metrics({"accuracy": trial_value})

def objective(trial: optuna.Trial):
    data, target = sklearn.datasets.load_breast_cancer(return_X_y=True)
    train_x, test_x, train_y, test_y = train_test_split(data, target, test_size=0.25)
    dtrain = lgb.Dataset(train_x, label=train_y)

    param = {
        "objective": "binary",
```

```
        "metric": "binary_logloss",
        "verbosity": -1,
        "boosting_type": "gbdt",
        "lambda_l1": trial.suggest_loguniform("lambda_l1", 1e-8, 10.0),
        "lambda_l2": trial.suggest_loguniform("lambda_l2", 1e-8, 10.0),
        "num_leaves": trial.suggest_int("num_leaves", 2, 256),
        "feature_fraction": trial.suggest_uniform("feature_fraction", 0.4, 1.0),
        "bagging_fraction": trial.suggest_uniform("bagging_fraction", 0.4, 1.0),
        "bagging_freq": trial.suggest_int("bagging_freq", 1, 7),
        "min_child_samples": trial.suggest_int("min_child_samples", 5, 100),
    }

    gbm = lgb.train(param, dtrain)
    preds = gbm.predict(test_x)
    pred_labels = np.rint(preds)
    accuracy = sklearn.metrics.accuracy_score(test_y, pred_labels)
    return accuracy

if __name__ == "__main__":
    study = optuna.create_study(
        direction="maximize",
    )
    study.optimize(
        objective,
        n_trials=100,
        callbacks=[mlflow_callback]
    )

    print("Number of finished trials: {}".format(len(study.trials)))

    print("Best trial:")
```

```
    trial = study.best_trial

    print("  Value: {}".format(trial.value))

    print("  Params: ")
    for key, value in trial.params.items():
        print("    {}: {}".format(key, value))
```

Execute the following command.

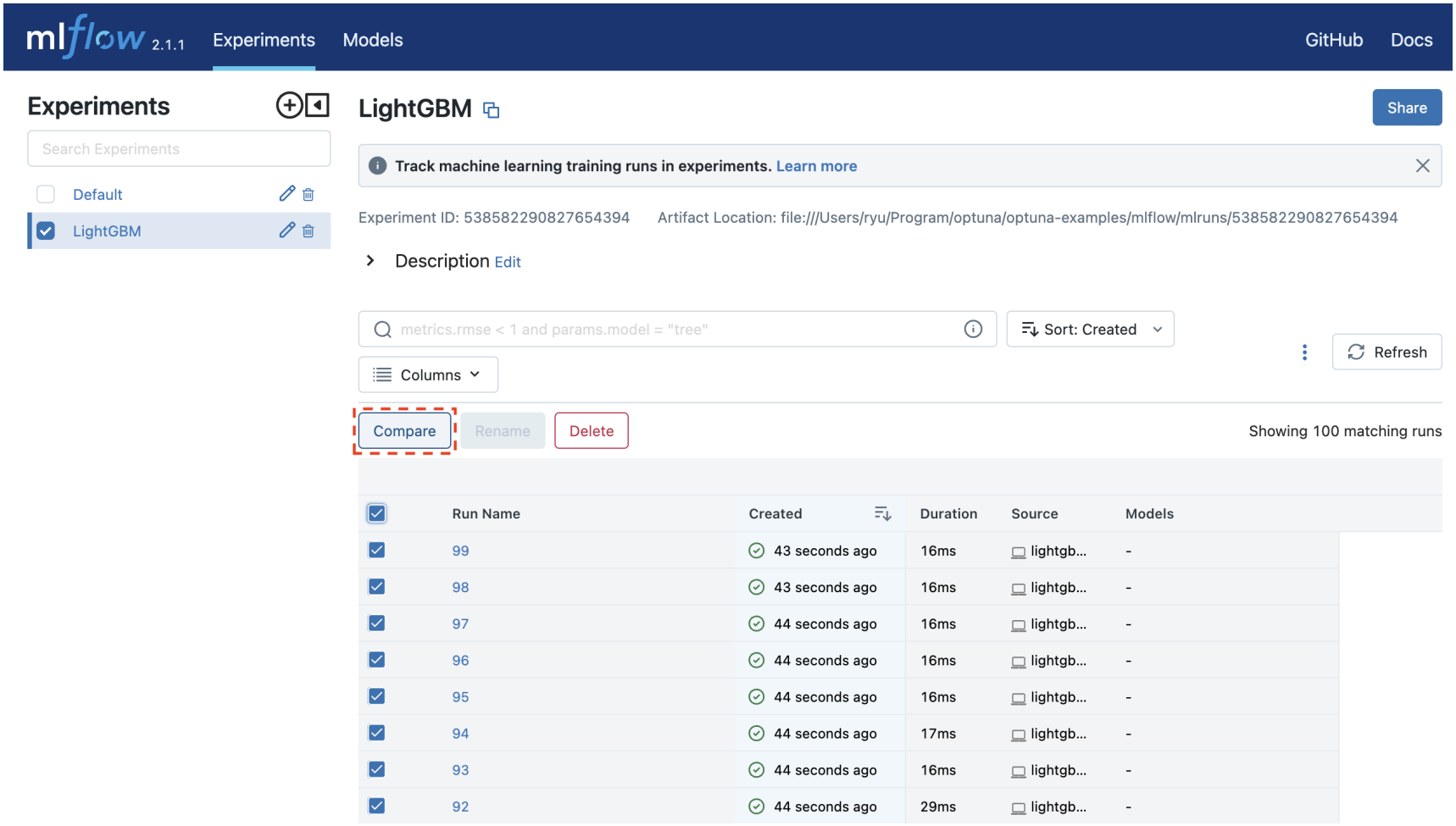
```
$ python mlflow_lightgbm.py
```

The results of the Optuna trials will be returned.

```
Number of finished trials: 100
Best trial:
  Value: 0.993006993006993
  Params:
    lambda_l1: 0.06612656999057859
    lambda_l2: 2.052197523772018
    num_leaves: 119
    feature_fraction: 0.4295597947409407
    bagging_fraction: 0.8875131492710178
    bagging_freq: 2
    min_child_samples: 24
```

Run `mlflow ui` and access `http://127.0.0.1:5000` to see the results of the experiment in your browser.

```
$ mlflow ui
```


mlflow 2.1.1
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
LightGBM
LightGBM
Share
Track machine learning training runs in experiments. Learn more
Experiment ID: 538582290827654394
Artifact Location: file:///Users/ryu/Program/optuna/optuna-examples/mlflow/mlruns/538582290827654394
Description Edit
metrics.rmse < 1 and params.model = "tree"
Sort: Created
Columns
Refresh
Compare
Rename
Delete
Showing 100 matching runs
Run Name
Created
Duration
Source
Models
99 43 seconds ago 16ms lightgb... -
98 43 seconds ago 16ms lightgb... -
97 44 seconds ago 16ms lightgb... -
96 44 seconds ago 16ms lightgb... -
95 44 seconds ago 16ms lightgb... -
94 44 seconds ago 17ms lightgb... -
93 44 seconds ago 16ms lightgb... -
92 44 seconds ago 29ms lightgb... -

mlflow 2.1.1 Experiments Models GitHub Docs

LightGBM ›

Comparing 100 Runs from 1 Experiment

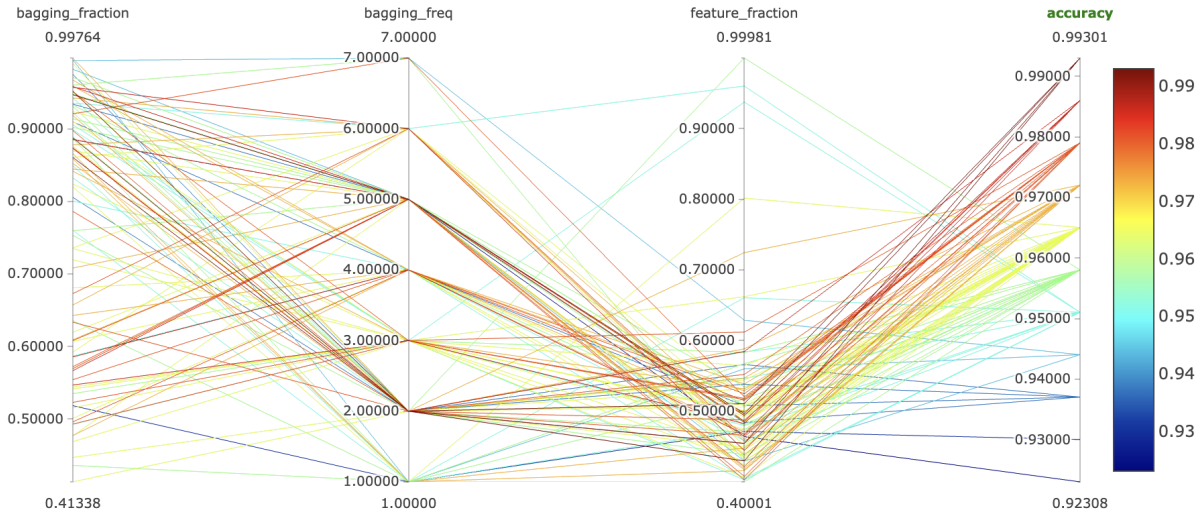
Visualizations

Parallel Coordinates Plot | Scatter Plot | Box Plot | Contour Plot

Parameters: bagging_fraction × bagging_freq × feature_fraction ×

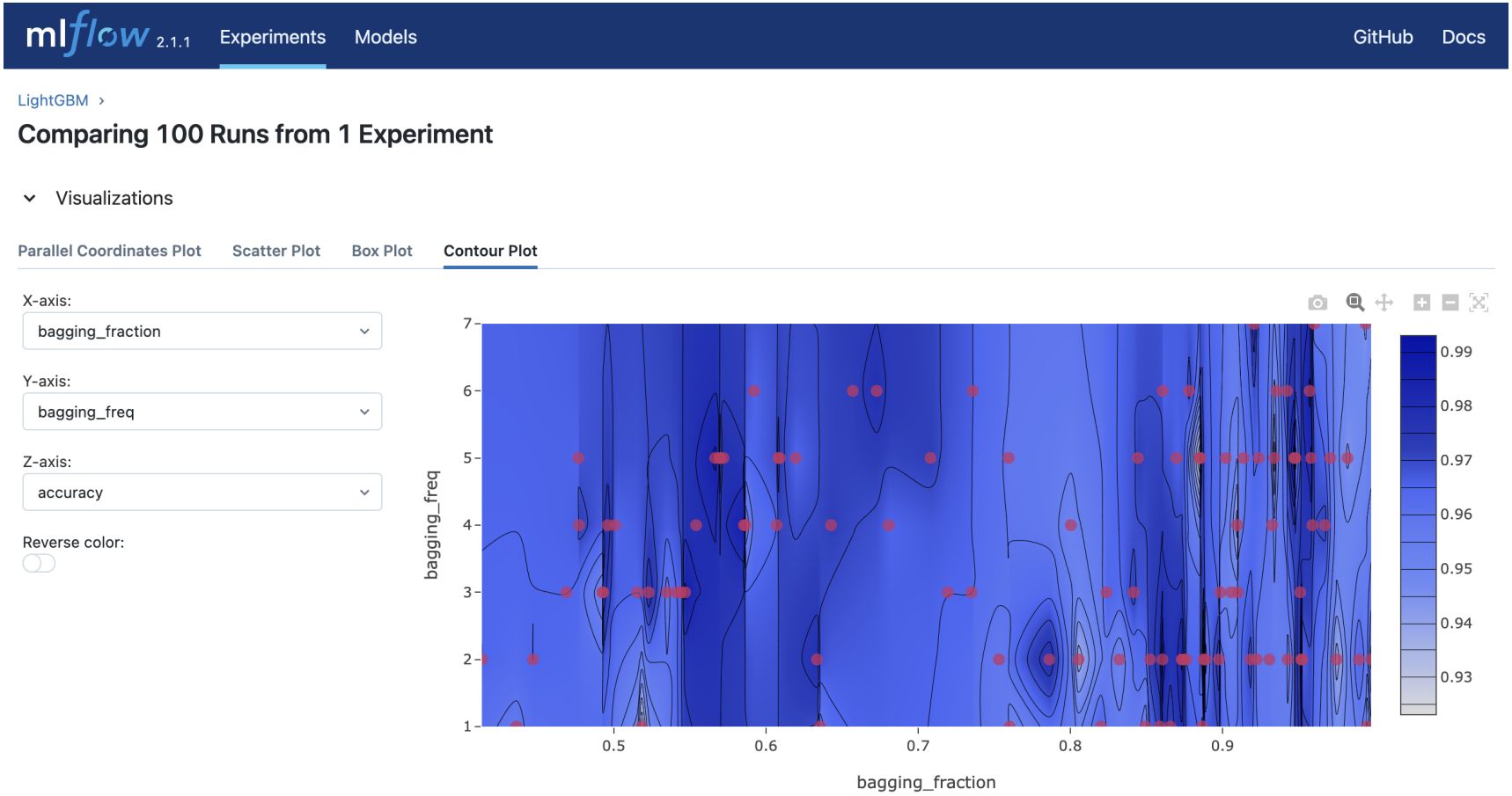
Metrics: accuracy ×

Clear All



Run details

Run ID:	d237c7ca80de4973a783...	e8ccba3d3ed7415aa25e...	61b6157d8a5a4b9e9b1...	63eab9f852d04ad5b6d1...	07014315a8d24916be0...	2c64b
Run Name:	99	98	97	96	95	94
Start Time:	2023-01-21 09:35:48	2023-01-21 09:35:48	2023-01-21 09:35:47	2023-01-21 09:35:47	2023-01-21 09:35:47	2023-
End Time:	2023-01-21 09:35:48	2023-01-21 09:35:48	2023-01-21 09:35:47	2023-01-21 09:35:47	2023-01-21 09:35:47	2023-
Duration:	16ms	16ms	16ms	16ms	16ms	17ms



Keras example

Install the libraries.

```
$ pip install mlflow
$ pip install optuna
```

```
$ pip install scikit-learn
$ pip install tensorflow
```

Write the following Keras code in `mlflow_keras.py` Optuna provides an integration with MLflow called `optuna.integration.MLflowCallback`. Below is the code for using `optuna.integration.MLflowCallback`.

mlflow_keras.py

```
import optuna
from optuna.integration.mlflow import MLflowCallback

from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.backend import clear_session
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import SGD


TEST_SIZE = 0.25
BATCHSIZE = 16
EPOCHS = 100


def standardize(data):
    return StandardScaler().fit_transform(data)


def create_model(num_features: int, trial: optuna.Trial):
```

```
    model = Sequential()
    model.add(
        Dense(
            num_features,
            activation="relu",
            kernel_initializer="normal",
            input_shape=(num_features,),
        )
    ),
    model.add(Dense(16, activation="relu", kernel_initializer="normal"))
    model.add(Dense(16, activation="relu", kernel_initializer="normal"))
    model.add(Dense(1, kernel_initializer="normal", activation="linear"))

    optimizer = SGD(
        learning_rate=trial.suggest_float("learning_rate", 1e-5, 1e-1, log=True),
        momentum=trial.suggest_float("momentum", 0.0, 1.0),
    )
    model.compile(loss="mean_squared_error", optimizer=optimizer)
    return model


def objective(trial: optuna.Trial):
    # Clear clutter from previous Keras session graphs.
    clear_session()

    X, y = load_wine(return_X_y=True)
    X = standardize(X)
    X_train, X_valid, y_train, y_valid = train_test_split(
        X, y, test_size=TEST_SIZE, random_state=42
    )
```

```
    model = create_model(X.shape[1], trial)
    model.fit(X_train, y_train, shuffle=True, batch_size=BATCHSIZE, epochs=EPOCHS, verbose=False)

    return model.evaluate(X_valid, y_valid, verbose=0)


if __name__ == "__main__":
    mlflc = MLflowCallback(metric_name="mean_squared_error")
    study = optuna.create_study(
        study_name="Keras",
    )
    study.optimize(objective, n_trials=100, timeout=600, callbacks=[mlflc])

    print("Number of finished trials: {}".format(len(study.trials)))

    print("Best trial:")
    trial = study.best_trial

    print("  Value: {}".format(trial.value))

    print("  Params: ")
    for key, value in trial.params.items():
        print("    {}: {}".format(key, value))
```

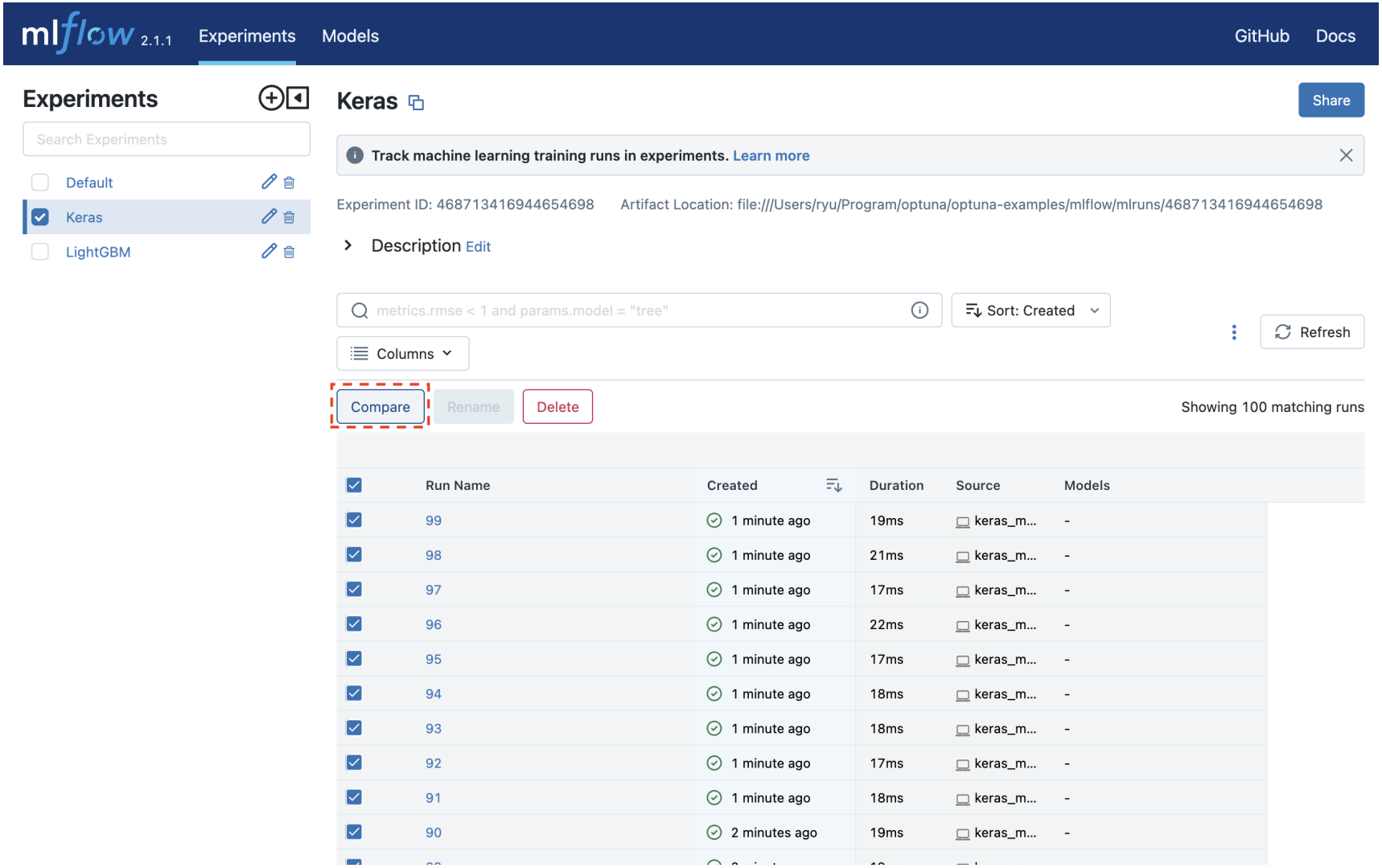
Execute the following command.

```
$ python mlflow_keras.py
```

The results of the Optuna trials will be returned.

```
Number of finished trials: 100
Best trial:
  Value: 0.0008772228611633182
  Params:
    learning_rate: 0.046622562292222114
    momentum: 0.9013368782978584
```

Run `mlflow ui` and access `http://127.0.0.1:5000` to see the results of the experiment in your browser.


mlflow 2.1.1
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
Keras
LightGBM
Keras
Share
Track machine learning training runs in experiments. Learn more
Experiment ID: 468713416944654698
Artifact Location: file:///Users/ryu/Program/optuna/optuna-examples/mlflow/mlruns/468713416944654698
Description Edit
metrics.rmse < 1 and params.model = "tree"
Sort: Created
Columns
Refresh
Compare
Rename
Delete
Showing 100 matching runs
Run Name
Created
Duration
Source
Models
99 1 minute ago 19ms keras_m... -
98 1 minute ago 21ms keras_m... -
97 1 minute ago 17ms keras_m... -
96 1 minute ago 22ms keras_m... -
95 1 minute ago 17ms keras_m... -
94 1 minute ago 18ms keras_m... -
93 1 minute ago 18ms keras_m... -
92 1 minute ago 17ms keras_m... -
91 1 minute ago 18ms keras_m... -
90 2 minutes ago 19ms keras_m... -

mlflow 2.1.1 Experiments Models GitHub Docs

Keras ›

Comparing 100 Runs from 1 Experiment

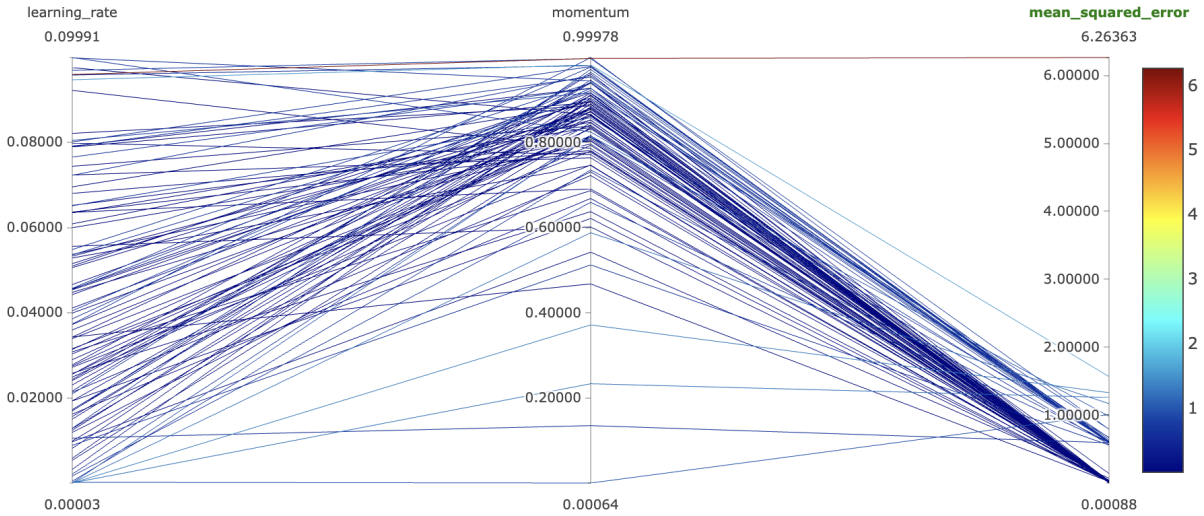
Visualizations

Parallel Coordinates Plot | Scatter Plot | Box Plot | Contour Plot

Parameters: learning_rate × momentum ×

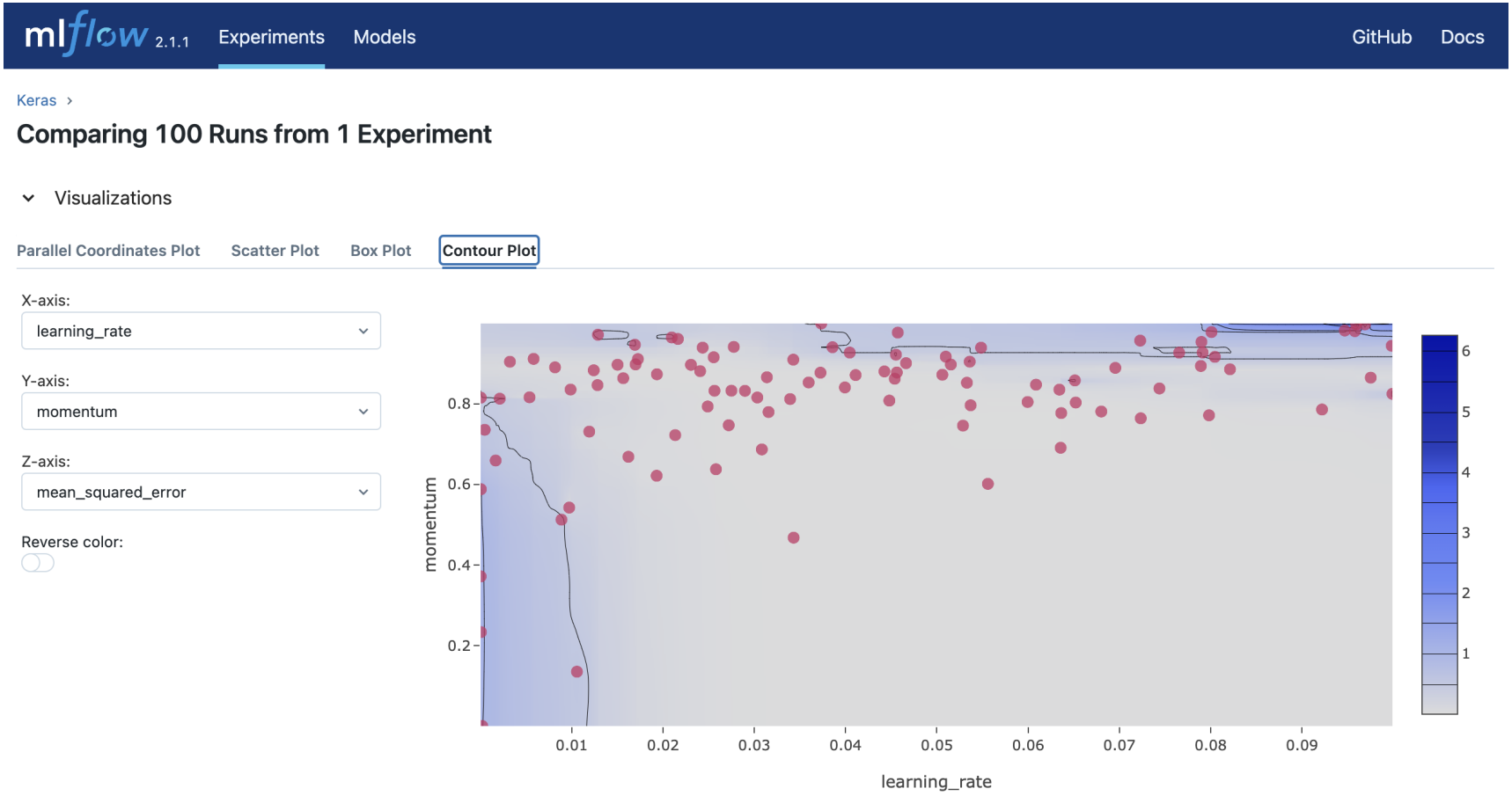
Metrics: mean_squared_error ×

Clear All



Run details

Run ID:	69350bac37b642a6839...	f5efc241dc6c4511adb6...	f7e297cf1d9e4c9e9bf7c...	d6e1afeaf04047c68edba...	6cfa4bb6538d49ecb37d...	1804
Run Name:	99	98	97	96	95	94
Start Time:	2023-01-21 09:42:17	2023-01-21 09:42:15	2023-01-21 09:42:14	2023-01-21 09:42:12	2023-01-21 09:42:11	2023-
End Time:	2023-01-21 09:42:17	2023-01-21 09:42:15	2023-01-21 09:42:14	2023-01-21 09:42:12	2023-01-21 09:42:11	2023-
Duration:	19ms	21ms	17ms	22ms	17ms	18ms



References

optuna.integration.MLflowCallback — Optuna 3.1.0 documentation

optuna.readthedocs.io

optuna / optuna-examples

Examples for https://github.com/optuna/optuna

411 133 Python

Optuna

CPU and GPU

Ryusei Kakujo

Data science, data engineering, transportation engineering

Bench Press 100kg!

Amazon Cognito Amazon QuickSight Amazon RDS Amazon Redshift Amazon S3 API AWS
AWS API Gateway AWS Chalice AWS Control Tower AWS Lambda BERT Chrome Extension CircleCI

Classification Cloud Functions Cloud IAM Cloud Run Cloud Storage Clustering Data Modeling
Data Platform dbt Decision Tree Deep Learning Descriptive Statistics Dimensionality Reduction
Discrete Choice Model Docker Economics FastAPI Firebase git GitHub GitHub Actions
Google Cloud Hugging Face Hypothesis Testing Inferential Statistics Interval Estimation JavaScript Jinja
Kedro LightGBM Linux LLM Machine Learning MLflow MLOps MySQL NLP Nodejs ONNX
Optuna Pandas Pinecone Probability Distribution Python PyTorch R Regression Singer sklearn
Slack Snowflake Software Development SQL Statistical Model Statistics Tabular TensorFlow
Terraform Transportation TypeScript Vector DB Vertex AI XGBoost