homework-duration-prediction

May 26, 2023

1 Homework

The goal of this homework is to get familiar with tools like MLflow for experiment tracking and model management.

[1]:
```
import os
import mlflow
from pathlib import Path
```

[2]:
```
MLFLOW_TRACKING_URI = os.environ["MLFLOW_TRACKING_URI"]
DATA_DIR = Path(os.environ["DATA_DIR"])
MODELS_DIR = Path(os.environ["MODELS_DIR"])
```

1.1 Q1. Install the package

To get started with MLflow you'll need to install the appropriate Python package.

For this we recommend creating a separate Python environment, for example, you can use conda environments, and then install the package there with pip or conda.

Once you installed the package, run the command `mlflow --version` and check the output.

What's the version that you have?

[3]:
```
mlflow.__version__
```

[3]:
```
'2.3.2'
```

1.2 Q2. Download and preprocess the data

We'll use the Green Taxi Trip Records dataset to predict the amount of tips for each trip.

Download the data for January, February and March 2022 in parquet format from here.

Use the script `preprocess_data.py` located in the folder homework to preprocess the data.

The script will:

- load the data from the folder (the folder where you have downloaded the data),
- fit a DictVectorizer on the training set (January 2022 data),
- save the preprocessed datasets and the DictVectorizer to disk.

Your task is to download the datasets and then execute this command:

```
python preprocess_data.py --raw_data_path <TAXI_DATA_FOLDER> --dest_path ./output
```

[4]:
```
!python preprocess_data.py --raw_data_path $DATA_DIR --dest_path ./output
```

Tip: go to 02-experiment-tracking/homework/ folder before executing the command and change the value of to the location where you saved the data.

So what's the size of the saved DictVectorizer file?

[5]:
```
size_in_kb = Path("./output/dv.pkl").stat().st_size / 1000
size_in_kb
```

[5]:
```
153.66
```

1.3 Q3. Train a model with autolog

We will train a `RandomForestRegressor` (from Scikit-Learn) on the taxi dataset.

We have prepared the training script `train.py` for this exercise, which can be also found in the folder `homework`.

The script will:

- load the datasets produced by the previous step,
- train the model on the training set,
- calculate the RMSE score on the validation set.

Your task is to modify the script to enable **autologging** with MLflow, execute the script and then launch the MLflow UI to check that the experiment run was properly tracked.

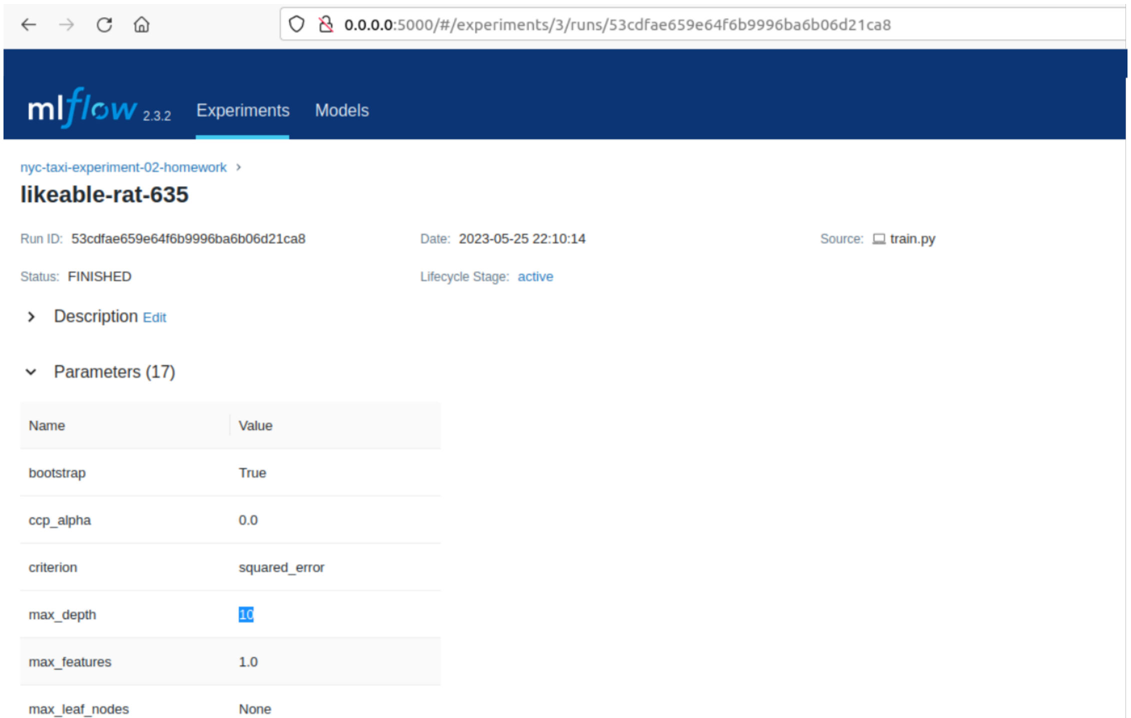
[6]:
```
mlflow.set_tracking_uri(MLFLOW_TRACKING_URI)
mlflow.set_experiment("nyc-taxi-experiment-02-homework")

!python train.py --data_path ./output
```

```
2023/05/26 08:19:57 WARNING mlflow.utils.autologging_utils: MLflow autologging
encountered a warning: "/opt/conda/lib/python3.10/site-
packages/_distutils_hack/__init__.py:33: UserWarning: Setuptools is replacing
distutils."
```

Tip 1: don't forget to wrap the training code with a with `mlflow.start_run():` statement as we showed in the videos.

Tip 2: don't modify the hyperparameters of the model to make sure that the training will finish quickly.

What is the value of the `max_depth` parameter:



1.4 Q4. Tune model hyperparameters

Now let's try to reduce the validation error by tuning the hyperparameters of the `RandomForestRegressor` using `optuna`. We have prepared the script `hpo.py` for this exercise.

Your task is to modify the script hpo.py and make sure that the validation RMSE is logged to the tracking server for each run of the hyperparameter optimization (you will need to add a few lines of code to the `objective` function) and run the script without passing any parameters.

After that, open UI and explore the runs from the experiment called `random-forest-hyperopt` to answer the question below.

Note: Don't use autologging for this exercise.

The idea is to just log the information that you need to answer the question below, including:

- the list of hyperparameters that are passed to the `objective` function during the optimization,
- the RMSE obtained on the validation set (February 2022 data).

What's the best validation RMSE that you got?

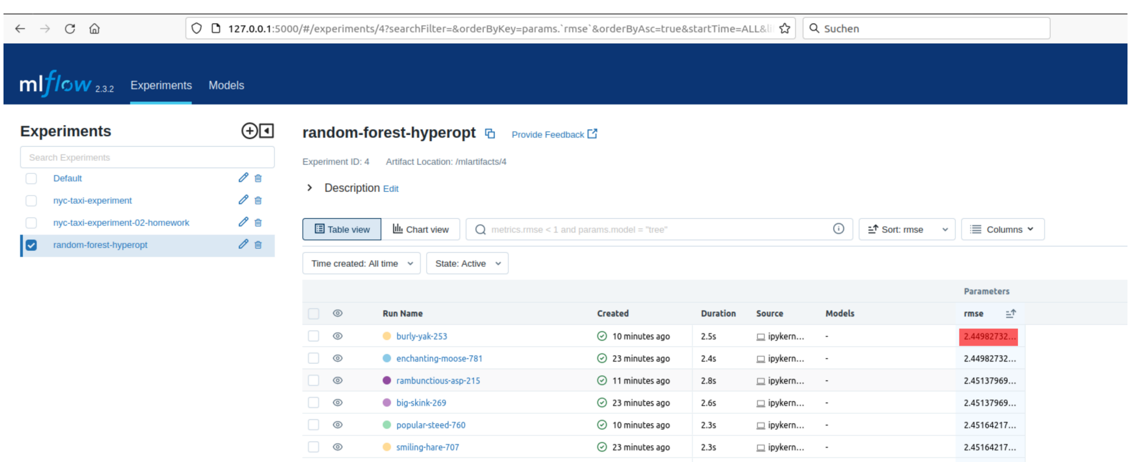
Comment Logging did not work in the function, instead through an api error, so I isolated the function

[7]:
```
import optuna
from hpo import load_pickle
from optuna.samplers import TPESampler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

[8]:
```
mlflow.sklearn.autolog(disable=True)
mlflow.set_experiment("random-forest-hyperopt")

data_path = "./output"

X_train, y_train = load_pickle(os.path.join(data_path, "train.pkl"))
X_val, y_val = load_pickle(os.path.join(data_path, "val.pkl"))

def objective(trial):

    with mlflow.start_run():
        params = {
            'n_estimators': trial.suggest_int('n_estimators', 10, 50, 1),
            'max_depth': trial.suggest_int('max_depth', 1, 20, 1),
            'min_samples_split': trial.suggest_int('min_samples_split', 2, 10,␣
 ↪1),
            'min_samples_leaf': trial.suggest_int('min_samples_leaf', 1, 4, 1),
            'random_state': 42,
            'n_jobs': -1
        }

        rf = RandomForestRegressor(**params)
        rf.fit(X_train, y_train)
        y_pred = rf.predict(X_val)
        rmse = mean_squared_error(y_val, y_pred, squared=False)

        mlflow.log_metric("rmse", rmse)
        mlflow.log_params(params)

        return rmse

sampler = TPESampler(seed=42)
study = optuna.create_study(direction="minimize", sampler=sampler)
study.optimize(objective, n_trials=10)
```

```
[I 2023-05-26 08:20:00,234] A new study created in memory with name:
no-name-6aa8298d-403f-448f-97ed-9e748abd4a42
[I 2023-05-26 08:20:03,015] Trial 0 finished with value:
2.451379690825458 and parameters: {'n_estimators': 25, 'max_depth': 20,
'min_samples_split': 8, 'min_samples_leaf': 3}. Best is trial 0 with value:
2.451379690825458.
[I 2023-05-26 08:20:03,900] Trial 1 finished with value:
2.4667366020368333 and parameters: {'n_estimators': 16, 'max_depth': 4,
'min_samples_split': 2, 'min_samples_leaf': 4}. Best is trial 0 with value:
2.451379690825458.
[I 2023-05-26 08:20:06,459] Trial 2 finished with value:
2.449827329704216 and parameters: {'n_estimators': 34, 'max_depth': 15,
```

```
'min_samples_split': 2, 'min_samples_leaf': 4}. Best is trial 2 with value:
2.449827329704216.
[I 2023-05-26 08:20:08,097] Trial 3 finished with value:
2.460983516558473 and parameters: {'n_estimators': 44, 'max_depth': 5,
'min_samples_split': 3, 'min_samples_leaf': 1}. Best is trial 2 with value:
2.449827329704216.
[I 2023-05-26 08:20:09,614] Trial 4 finished with value:
2.453877262701052 and parameters: {'n_estimators': 22, 'max_depth': 11,
'min_samples_split': 5, 'min_samples_leaf': 2}. Best is trial 2 with value:
2.449827329704216.
[I 2023-05-26 08:20:10,830] Trial 5 finished with value:
2.4720122094960733 and parameters: {'n_estimators': 35, 'max_depth': 3,
'min_samples_split': 4, 'min_samples_leaf': 2}. Best is trial 2 with value:
2.449827329704216.
[I 2023-05-26 08:20:13,254] Trial 6 finished with value:
2.4516421799356767 and parameters: {'n_estimators': 28, 'max_depth': 16,
'min_samples_split': 3, 'min_samples_leaf': 3}. Best is trial 2 with value:
2.449827329704216.
[I 2023-05-26 08:20:14,195] Trial 7 finished with value:
2.5374040268274087 and parameters: {'n_estimators': 34, 'max_depth': 1,
'min_samples_split': 7, 'min_samples_leaf': 1}. Best is trial 2 with value:
2.449827329704216.
[I 2023-05-26 08:20:15,844] Trial 8 finished with value:
2.455971238567075 and parameters: {'n_estimators': 12, 'max_depth': 19,
'min_samples_split': 10, 'min_samples_leaf': 4}. Best is trial 2 with value:
2.449827329704216.
[I 2023-05-26 08:20:16,753] Trial 9 finished with value:
2.486106021576535 and parameters: {'n_estimators': 22, 'max_depth': 2,
'min_samples_split': 8, 'min_samples_leaf': 2}. Best is trial 2 with value:
2.449827329704216.
```



1.5 Q5. Promote the best model to the model registry

The results from the hyperparameter optimization are quite good. So, we can assume that we are ready to test some of these models in production. In this exercise, you'll promote the best model to the model registry. We have prepared a script called `register_model.py`, which will check the results from the previous step and select the top 5 runs. After that, it will calculate the RMSE of those models on the test set (March 2022 data) and save the results to a new experiment called `random-forest-best-models`.

Your task is to update the script `register_model.py` so that it selects the model with the lowest RMSE on the test set and registers it to the model registry.

Tips for MLflow:

- you can use the method `search_runs` from the `MlflowClient` to get the model with the lowest RMSE,
- to register the model you can use the method `mlflow.register_model` and you will need to pass the right model_uri in the form of a string that looks like this: `"runs:/<RUN_ID>/model"`, and the name of the model (make sure to choose a good one!).

Comment Logging did not work in the function, instead through an api error, so I isolated the function

[9]:
```
import os
import pickle

from mlflow.entities import ViewType
from mlflow.tracking import MlflowClient
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

from register_model import train_and_log_model

HPO_EXPERIMENT_NAME = "random-forest-hyperopt"
EXPERIMENT_NAME = "random-forest-best-models"
RF_PARAMS = ['max_depth', 'n_estimators', 'min_samples_split',␣
 ↪'min_samples_leaf', 'random_state', 'n_jobs']

mlflow.set_tracking_uri(MLFLOW_TRACKING_URI)
mlflow.set_experiment(EXPERIMENT_NAME)
mlflow.sklearn.autolog()


def run_register_model(data_path: str, top_n: int):

    client = MlflowClient(MLFLOW_TRACKING_URI)

    # Retrieve the top_n model runs and log the models
    experiment = client.get_experiment_by_name(HPO_EXPERIMENT_NAME)
    runs = client.search_runs(
```

```
        experiment_ids=experiment.experiment_id,
        run_view_type=ViewType.ACTIVE_ONLY,
        max_results=top_n,
        order_by=["metrics.rmse ASC"]
    )

    print("Selected best runs:")
    for run in runs:
        print(run.info.run_name)
        train_and_log_model(data_path=data_path, params=run.data.params)

    # Select the model with the lowest test RMSE
    experiment = client.get_experiment_by_name(EXPERIMENT_NAME)
    best_run = client.search_runs(
        experiment_ids=experiment.experiment_id,
        run_view_type=ViewType.ACTIVE_ONLY,
        order_by=["metrics.test_rmse ASC"]
    )[0]

    print(f"Best run name = {best_run.info.run_name}")
    print(f"Best run id = {best_run.info.run_id}")

    # Register the best model
    mlflow.register_model(
        model_uri=f"runs:/{best_run.info.run_id}/model",
        name="nyc-taxi-green"
    )
```

[10]:
```
run_register_model("./output", 5)
```

```
Selected best runs:
hilarious-trout-414
```

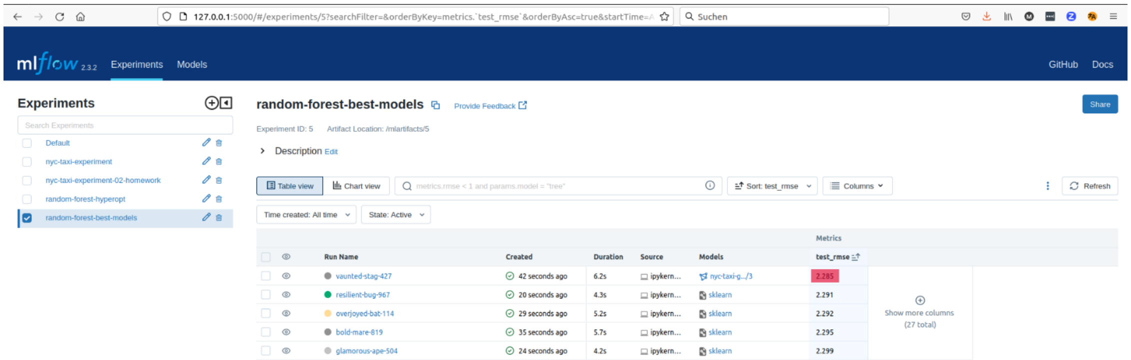
```
2023/05/26 08:20:22 WARNING mlflow.utils.autologging_utils: MLflow autologging
encountered a warning: "/opt/conda/lib/python3.10/site-
packages/_distutils_hack/__init__.py:33: UserWarning: Setuptools is replacing
distutils."
```

```
upset-horse-810
unruly-koi-3
secretive-shrew-736
bustling-sheep-896
Best run name = unequaled-lynx-848
Best run id = 4202140c1a014e14a7759624d29db1b4
```

```
Registered model 'nyc-taxi-green' already exists. Creating a new version of this
model…
2023/05/26 08:20:43 INFO mlflow.tracking._model_registry.client: Waiting up to
300 seconds for model version to finish creation. Model name: nyc-taxi-green,
```

```
version 4
Created version '4' of model 'nyc-taxi-green'.
```

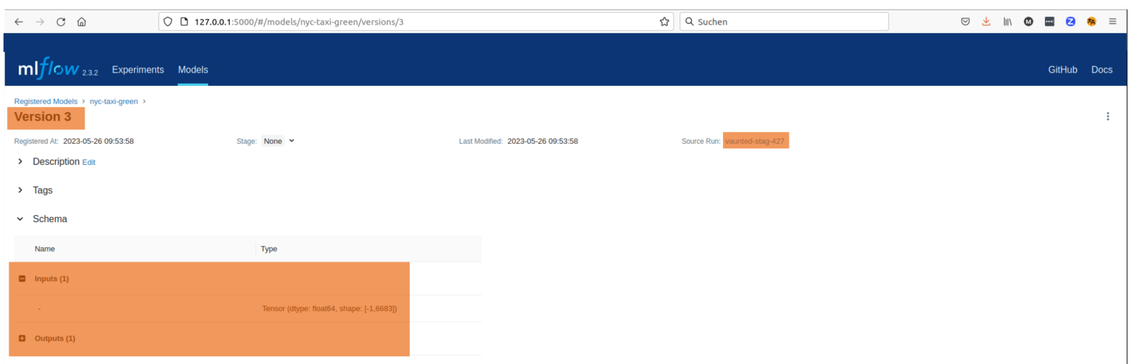
What is the test RMSE of the best model?



1.6 Q6. Model metadata

Now explore your best model in the model registry using UI. What information does the model registry contain about each model?

- Version number **YES**
- Source experiment **YES (source experiment id)**
- Model signature **YES**
- All the above answers are correct **YES**



[ ]: