Neptune Tutorial

Setup:

For installation and quick introduction to Neptune, follow the instructions from this link.
Installation and setup - Documentation

tldr

```
pip install scikit-learn neptune-client neptune-sklearn
scikit-learn
```

NOTE: Create a neptune account ahead of the experiment.

Tracking data

You can find information on how to track different assets in Neptune here
https://docs.neptune.ai/how-to-guides/experiment-tracking/organize-ml-experiments

Examples:

Model

To upload a model to neptune.ai; we can upload the model as a pickle file. The following command, uses neptune-sklearn's get_pickled_model( ) to log model:

```
run["model"] = npt_utils.get_pickled_model(model)
```

Parameters

To log the parameters we run the following command. Note that this also gets logged with the model.pkl (above):

```
parameters={'n_estimators': 50, 'max_depth': 5,
'min_samples_split': 6}
# We will log the parameters to params in neptune.ai
Run["params"] = parameters
```

Evaluation metrics

Similar to the parameters, we log each of the evaluation metrics in the following manner:

```
# Generating the Evaluation metrics
```

```
y_test_predict = model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict)))
r2 = r2_score(y_test, y_test_predict)

# Logging each metric to neptune
run['scores/max_error'] = max_error(y_test, y_test_predict)
run['scores/mean_absolute_error'] =
mean_absolute_error(y_test, y_test_predict)
run['scores/r2_score'] = r2_score(y_test, y_test_predict)
```

Dataset

The dataset generated will be of type numpy. Therefore, we will need to convert it into a DataFrame using pandas. The dataset will then be downloaded as .csv file.:

```
# We will save the dataset as .csv file
dataset = pd.DataFrame(data=data['data'], columns =
data['feature_names']).to_csv("data.csv", header=None, index=None)
```

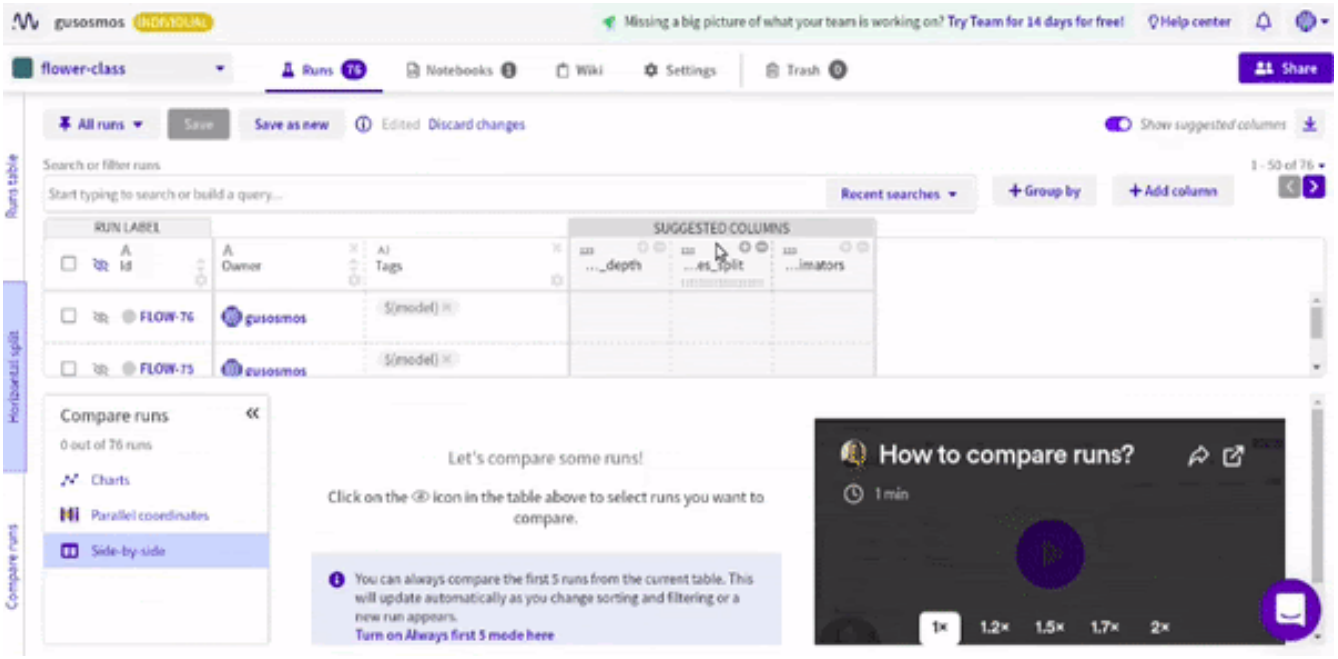
The data.csv file is now saved in the same directory as the python file, so to log it we specify the name data.csv. `.upload("path/to/file")`

```
# Logging the data.csv
run['data'].upload("data.csv")
```

Retrieve Data

Neptune provides a web dashboard that shows details of multiple experiment runs. Neptune also provides a programmatic way of retrieve experiment details.

The data logged for each run can be found by clicking on the desired run. Here you will find the logged data, source code, and other information from that specific run.



To programmatically fetch the data from a previous run, you need to specify the run ID in init().

Evaluation metrics

Here we are getting the r2_score from the run.

```
      run = neptune.init(
   project='<YOUR_WORKSPACE/YOUR_PROJECT>',
   api_token='<YOUR_API_TOKEN>',
   run='<RUN_ID>' # for example 'EX-8'
   )
# query a field
R2_score = run['scores/r2_score'].fetch()
```

Model

If you want to **download** a specific model from a given run. You specify the **run ID** in:

```
run = neptune.init(
   project='<YOUR_WORKSPACE/YOUR_PROJECT>',
   api_token='<YOUR_API_TOKEN>',
   run='<RUN_ID>' # for example 'FLOW-88'
   )
```

Then download the model by running:

```
run["model"].download()
```

Parameter

Using the code

First we need to specify which run we want to grab the parameters from. You need to specify the **run ID** in:

```
run = neptune.init(
    project='<YOUR_WORKSPACE/YOUR_PROJECT>',
    api_token='<YOUR_API_TOKEN>',
    run='<RUN_ID>' # for example 'FLOW-8'
    )
```

Then you can use the fetch() to grab a specific parameter. Below we are selecting the max_depth parameter from the specified run.

```
parameters = {'n_estimators': 50, 'max_depth': 5, 'min_samples_split': 6}

# Logging parameters to parameters directory in neptune.ai
run["parameters"] = parameters

# Retrieving max_depth from <RUN_ID>
max_depth = run['models/parameters/max_depth'].fetch()
```

Dataset

If you want to **download** a specific **dataset** from a given run. You specify the **run ID** in:

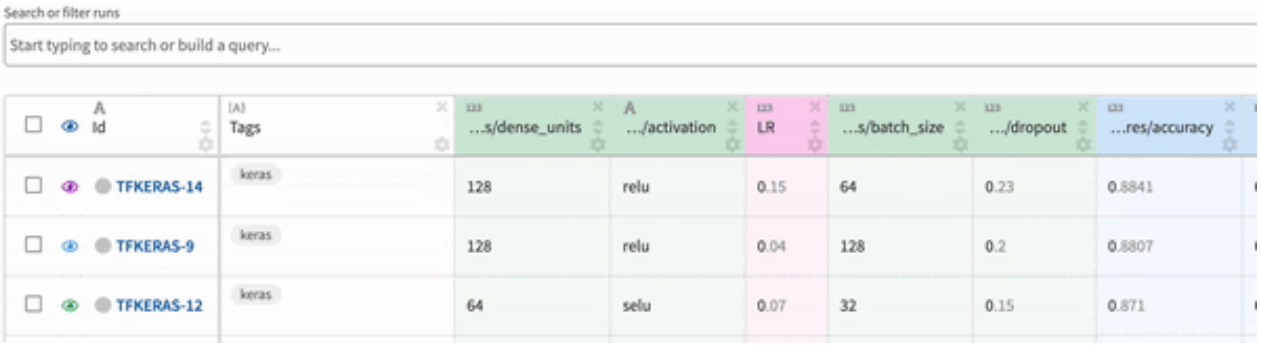
```
run = neptune.init(
    project='<YOUR_WORKSPACE/YOUR_PROJECT>',
    api_token='<YOUR_API_TOKEN>',
    run='<RUN_ID>' # for example 'FLOW-1'
    )
```

Then download the model by running:

```
run["data"].download("path")
```

Query & Comparing the multiple runs

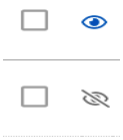
You can search and filter runs using the neptune.ai dashboard interface. To filter *Runs*, type in the *Fields'* name such as, “rmse”, into the filter box and select the field you are looking for. Select the desired operator and provide a reference value if required by the chosen operator.

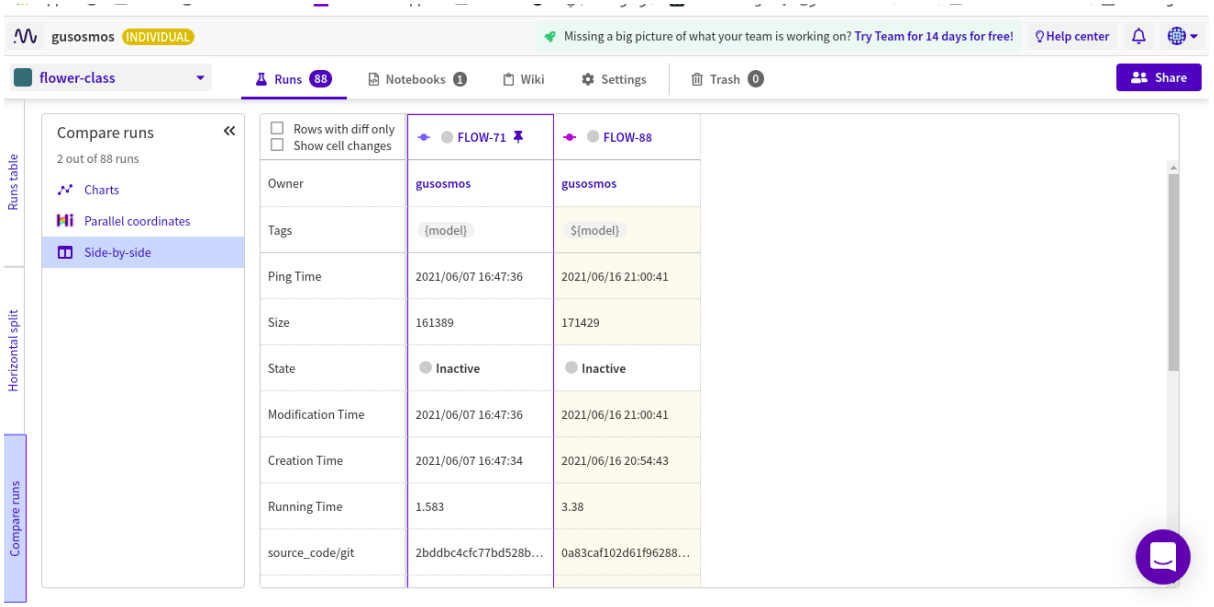


For further information:
[Organizing and filtering Runs - Documentation]

Comparing Multiple runs

In the table of runs, there is a column that displays an ‘eye’ icon beside each run. Clicking on the eye icon will enable comparison and will put the run column on a separate table where you will be able to compare it with other runs:





For further reading:
Next Comparing runs

Adding Tags to runs

If you want the individual runs to have tags next to them, e.g. having an RFR tag to runs that were run with the RFR algorithm, then you do the following:

Add tag attribute, followed by the tag name you want it to have:

```
run = neptune.init(
   project='<YOUR_WORKSPACE/YOUR_PROJECT>',
   api_token='<YOUR_API_TOKEN>',
   tags=['list-of', 'tags', 'goes-here'] # e.g 'RFR' or 'LR'
   )
```