Highlight Note

Creating a Data Drift Monitor

Azure Machine Learning supports data drift monitoring through the use of *datasets*. You can capture new feature data in a dataset and compare it to the dataset with which the model was trained.

Monitoring Data Drift by Comparing Datasets

It's common for organizations to continue to collect new data after a model has been trained. For example, a health clinic might use diagnostic measurements from previous patients to train a model that predicts diabetes likelihood, but continue to collect the same diagnostic measurements from all new patients. The clinic's data scientists could then periodically compare the growing collection of new data to the original training data, and identify any changing data trends that might affect model accuracy.

To monitor data drift using registered datasets, you need to register two datasets:

- A *baseline* dataset - usually the original training data.
- A *target* dataset that will be compared to the baseline based on time intervals. This dataset requires a column for each feature you want to compare, and a timestamp column so the rate of data drift can be measured.

[!NOTE] You can configure a deployed service to collect new data submitted to the model for inferencing, which is saved in Azure blob storage and can be used as a target dataset for data drift monitoring. See **Collect data from models in production** in the Azure Machine Learning documentation for more information.

After creating these datasets, you can define a *dataset monitor* to detect data drift and trigger alerts if the rate of drift exceeds a specified threshold. You can create dataset monitors using the visual interface in Azure Machine Learning studio, or by using the **DataDriftDetector** class in the Azure Machine Learning SDK as shown in the following example code:

```
from azureml.datadrift import DataDriftDetector

monitor = DataDriftDetector.create_from_datasets(workspace=ws,
                                                 name='dataset-drift-detector',
                                                 baseline_data_set=train_ds,
                                                 target_data_set=new_data_ds,
                                                 compute_target='aml-cluster',
                                                 frequency='Week',
                                                 feature_list=['age','height', 'bmi'],
                                                 latency=24)
```

After creating the dataset monitor, you can *backfill* to immediately compare the baseline dataset to existing data in the target dataset, as shown in the following example, which backfills the monitor based on weekly changes in data for the previous six weeks:

```
import datetime as dt

backfill = monitor.backfill( dt.datetime.now() - dt.timedelta(weeks=6), dt.datetime.now())
```