Machine learning creates static models from historical data. But, once deployed in production, ML models become unreliable and obsolete and degrade with time. There might be changes in the data distribution in production, thus causing biased predictions. User behavior itself might have changed compared to the baseline data the model was trained on, or there might be additional factors in real-world interactions which would have impacted the predictions. Data drift is a major reason model accuracy decreases over time. Thus, monitoring the changes continuously in our model’s behavior is of utmost importance. Flagging such drifts and automating certain jobs for retraining the model with new data or manual intervention of any kind ensures that the model remains relevant in production and gives fair and unbiased predictions over time.

Data drift refers to the phenomenon where the statistical properties of the input data used to train a machine learning model change over time. This can lead to a decrease in the model's performance as it may no longer accurately reflect the underlying patterns in the data. There are various types of data drift, and addressing them is crucial for maintaining the effectiveness of machine learning models. Let's explore the types of data drift and potential solutions, focusing on Databricks, AWS, and Azure as platforms.

## Types of Data Drift

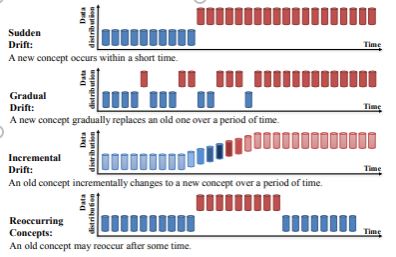
## 1) Concept Drift

Concept drift means that the statistical properties of the target variable, which the model is trying to predict, change over time. This causes problems because the predictions become less accurate and become unreliable.

**Example:**  
The abrupt changes in consumer behavior brought on by COVID-19 had a major impact on the accuracy of forecasting models that rely on historical data to inform their predictions. This can be treated as an example of concept drift.

One of the main reasons for concept drift to occur is the non-stationarity of data i.e., change in statistical properties of data with time.

**Solution**: Regularly retrain the model using updated data to adapt to changing patterns.



**2) Covariate Drift**

Covariate shift is the change in the distribution of one or more of the independent variables or input variables of the dataset. This means that even though the relationship between feature and target variable remains unchanged, the distribution of the feature itself has changed. When statistical properties of this input data change, the same model which has been built before will not provide unbiased results. This leads to inaccurate predictions.

**Example**: Suppose a model is trained with a salary variable that ranges from 200$ to 300$ and is in production. Over time, salary increases and the model encounters real-time data with higher salary figures of 1000$,1200$, and so on. And the model will see an increase in mean and variance, and therefore it leads to a data drift.

**Solution:** Periodically update the training dataset to include recent data and retrain the model.

### **Comparison:**

* **Nature of Change:**

**Concept Drift:** Involves changes in the relationship between features and the target variable.

**Covariate Drift:** Involves changes in the distribution of input features.

* **Impact on Models:**

**Concept Drift:** Affects the model's ability to predict the target variable accurately.

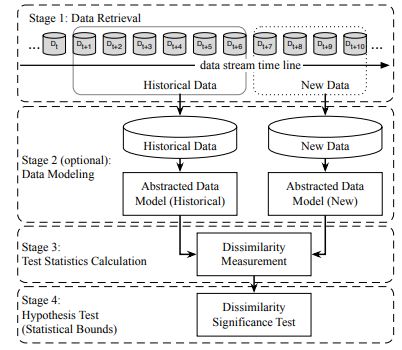
**Covariate Drift:** Affects the model's ability to generalize from the training distribution to the new distribution.

* **Adaptation Strategy:**

**Concept Drift:** Requires continuous monitoring and periodic model retraining to adapt to changing relationships.

**Covariate Drift:** May involve domain adaptation techniques to align the source and target distributions.

## Data Drift Detection Framework



**Stage 1 (Data Retrieval)** is used to retrieve data from data streams in chunks since a single data point cannot carry enough information to infer the overall distribution.

**Stage 2 (Data Modeling)** is used to extract the key features, that is, the features of the data that most impact a system if they drift.

**Stage 3 (Test Statistics Calculation)** is to measure the drift and calculate test statistics for the hypothesis test.

**Stage 4 (Hypothesis Test)** is used to evaluate the statistical significance of the change observed in Stage 3 or the p-value.

## Methods for Detecting Data Drift

All the methods for detecting data drift are lagging indicators of drift. Only after they have processed enough data after any kind of drift that has occurred, that the actual drift is detected.

**1) Kolmogorov-Smirnov (K-S) test:**

The K-S test is a nonparametric test that compares the cumulative distributions of two data sets, in this case, the training data and the post-training data. The null hypothesis for this test states that the data distributions from both the datasets are same. If the null is rejected then we can conclude that there is adrift in the model.

In our analysis, we have only considered numerical columns for the test.

For generating our final Data Drift analysis, the chi-squared test can be applied for the categorical features to identify data drift.

**2) Population Stability Index:**

It compares the distribution of the target variable in the test dataset to a training data set that was used to develop the model.



Steps for calculation:

1) Divide the expected (test) dataset and the actual (training dataset) into buckets and define the boundary values of the buckets based on the minimum and maximum values of that column in train data.

2) Calculate the % of observations in each bucket for both expected and actual datasets.

3) Calculate the PSI as given in the formula

**a) When PSI<=1**  
This means there is no change or shift in the distributions of both datasets.

**b) 0.1< PSI<0.2**

This indicates a slight change or shift has occurred.

**c) PSI>0.2**

This indicates a large shift in the distribution has occurred between both datasets.

**3) Model-Based Approach**

A Machine Learning-based model approach can also be used to detect data drift between two populations.

We need to label our data which has been used to build the current model in production as 0 and the real-time data gets labeled as 1. We now have to build a model and evaluate the results.

If the model gives high accuracy, it means that it can easily discriminate between the two sets of data. Thus, we could conclude that a covariate shift has occurred and the model will need to be recalibrated. On the other hand, if the model accuracy is around 0.5, it means that it is as good as a random guess. This means that a significant data shift has not occurred and we can continue to use the model.

The disadvantage of this model is that every time new input data is made available, the training and testing process needs to be repeated which can become computationally expensive.

**4) Using specialized drift detection techniques such as** **Adaptive Windowing (ADWIN)**:

The Adaptive Windowing (ADWIN) algorithm uses a sliding window approach to detect concept drift. Window size is fixed and ADWIN slides the fixed window for detecting any change on the newly arriving data. When two sub-windows show distinct means in the new observations the older sub-window is dropped.

A user-defined threshold is set to trigger a warning that drift is detected. If the absolute difference between the two means derived from two sub-windows exceeds the pre-defined threshold, an alarm is generated. This method is applicable for univariate data

**5) Page-Hinkley method**:

This drift detection method calculates the mean of the observed values and keeps updating the mean as and when new data arrives. A drift is detected if the observed mean at some instant is greater than a threshold value lambda.

## Handling data drift in production

In production, there are multiple ways to respond to data drift.

Some of the methods which are generally followed in the industry are:

**1) Blindly update model:**

This is a naïve approach. There is no proactive drift detection. Models are periodically retrained and updated with recent data. Without drift detection in place, it is difficult to estimate the time interval for re-training and model re-deployment.

**2) Training with weighted data:**

When a new model is trained instead of discarding old training data, use weight inversely proportional to the age of data.

**3) Incremental learning:**

As new data arrives, the models are continuously retrained and updated. As a result, the model is always adapting to the changes in the data distribution. This approach will work with machine learning models which allow incremental learning one instance of data at a time.

**Tools/services available in:**

**Azure:**

1. Azure Machine Learning (AML) Data Drift Monitoring:

Azure Machine Learning includes a Data Drift Monitor that helps in monitoring changes in the data over time. It enables us to compare the data used to train the model with the data the model is making predictions on.

2. Azure Machine Learning Datasets:

Azure Machine Learning Datasets allow to version and track changes in the datasets over time. We can use this feature to identify and manage data drift.

3. Azure Data Factory:

Azure Data Factory is a cloud-based data integration service that allows to create data-driven workflows for orchestrating and automating data movement and data transformation. Monitoring data flow over time can help detect data drift.

4. Azure Stream Analytics:

For real-time data processing scenarios, Azure Stream Analytics can be used to process and analyze streaming data. Monitoring the input data streams can help identify changes and drift in real-time.

5. Azure Monitor:

Azure Monitor provides a comprehensive solution for collecting, analyzing, and acting on telemetry from Azure and on-premises environments. It can be used to monitor data sources and detect changes that might indicate data drift.

6. Azure Synapse Analytics:

Azure Synapse Analytics is a cloud-based integrated analytics service. It includes capabilities for data warehousing, big data, and data integration. Monitoring and analyzing data stored in Azure Synapse Analytics can help identify drift.

7. Azure Data Explorer:

Azure Data Explorer is a fast and highly scalable data exploration service. It can be used to analyze large volumes of data and identify patterns or changes indicative of data drift.

8. Azure Cognitive Services:

In some cases, using Azure Cognitive Services for anomaly detection can help identify unexpected patterns or changes in data that may indicate data drift.

**Services/tools in Databricks**

1. MLflow Tracking:

- Databricks integrates with MLflow, an open-source platform for managing the end-to-end machine learning lifecycle. MLflow Tracking allows you to log and query experiments to compare and analyze model performance over time. This can help in detecting data drift by comparing model metrics across different runs.

2. Delta Lake:

- Delta Lake is an open-source storage layer that brings ACID transactions to Apache Spark and big data workloads. It allows for versioning of data, making it easier to track changes and manage data drift. You can use Delta Lake to capture and store different versions of datasets over time.

3. Databricks Delta Time Travel:

- Delta Time Travel is a feature of Delta Lake that allows you to query and access previous versions of your Delta tables. This feature can be valuable for comparing and analyzing changes in data over time.

4. Databricks ML Model Monitoring:

- Databricks ML Model Monitoring provides capabilities to monitor the performance of machine learning models in real-time. It allows you to set up alerts based on various metrics, helping you detect and respond to data drift as it occurs.

5. Databricks SQL Analytics:

- Databricks SQL Analytics provides a collaborative environment for data analysts and data scientists to query and visualize data. Using SQL Analytics, you can create queries and visualizations to monitor and analyze changes in your data.

6.\*Databricks Jobs and Notebooks:

- Databricks allows you to schedule and automate the execution of notebooks and jobs. You can leverage this functionality to periodically run data quality checks, model evaluations, and other monitoring tasks to identify data drift.

7. Databricks MLflow Registry:

- MLflow Registry, part of the MLflow platform, provides a centralized repository for managing and versioning machine learning models. It helps in keeping track of different model versions, making it easier to monitor changes and updates.

8. Databricks Monitoring and Logging:

- Databricks provides monitoring and logging capabilities, allowing you to track cluster performance, job execution, and other metrics. Monitoring these metrics can help you identify issues related to data drift or changes in your data processing workflows.

**Services/tools in AWS:**

1. AWS DataBrew:

- AWS DataBrew is a visual data preparation tool that makes it easy to clean and normalize data for analytics and machine learning. While not specifically designed for data drift monitoring, it can be used as part of a broader data preparation process to ensure that your data remains consistent over time.

2. Amazon SageMaker Model Monitor:

- Amazon SageMaker Model Monitor is a service that automatically detects and alerts you to concept drift, data drift, and quality issues in your machine learning models. It allows you to set up monitoring schedules to continuously analyze your data and model performance.

3. AWS Glue:

- AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy to prepare and load data for analysis. While not specifically focused on drift monitoring, AWS Glue can be part of a data pipeline that helps maintain data consistency.

4. Amazon CloudWatch:

- CloudWatch is a monitoring service that provides data and actionable insights for AWS resources. You can use CloudWatch to set up alarms and dashboards to monitor various metrics related to your data sources and machine learning models.

5. AWS CloudTrail:

- AWS CloudTrail records API calls made on your account, providing visibility into user activity, resource changes, and more. While not directly focused on data drift, it can help you track changes and activities within your AWS environment.

6. Amazon Simple Notification Service (SNS) and Amazon Simple Queue Service (SQS):

- You can use SNS and SQS to set up notifications and alerts based on changes or anomalies detected in your data. This can be part of a custom monitoring solution tailored to your specific needs.

7. Amazon QuickSight:

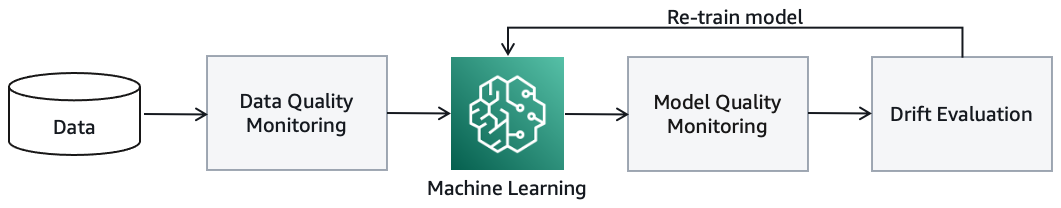
- Amazon QuickSight is a business analytics service that can be used for creating visualizations and dashboards. While not designed specifically for data drift monitoring, it can be used to visualize and analyze data changes over time.

8. AWS Lambda:

- You can use AWS Lambda to create custom functions that respond to changes in your data or model performance. For example, you could trigger Lambda functions based on CloudWatch alarms or other events.

**AWS (Sagemaker):**

There are three stages to detecting data drift: data quality monitoring, model quality monitoring, and drift evaluation:

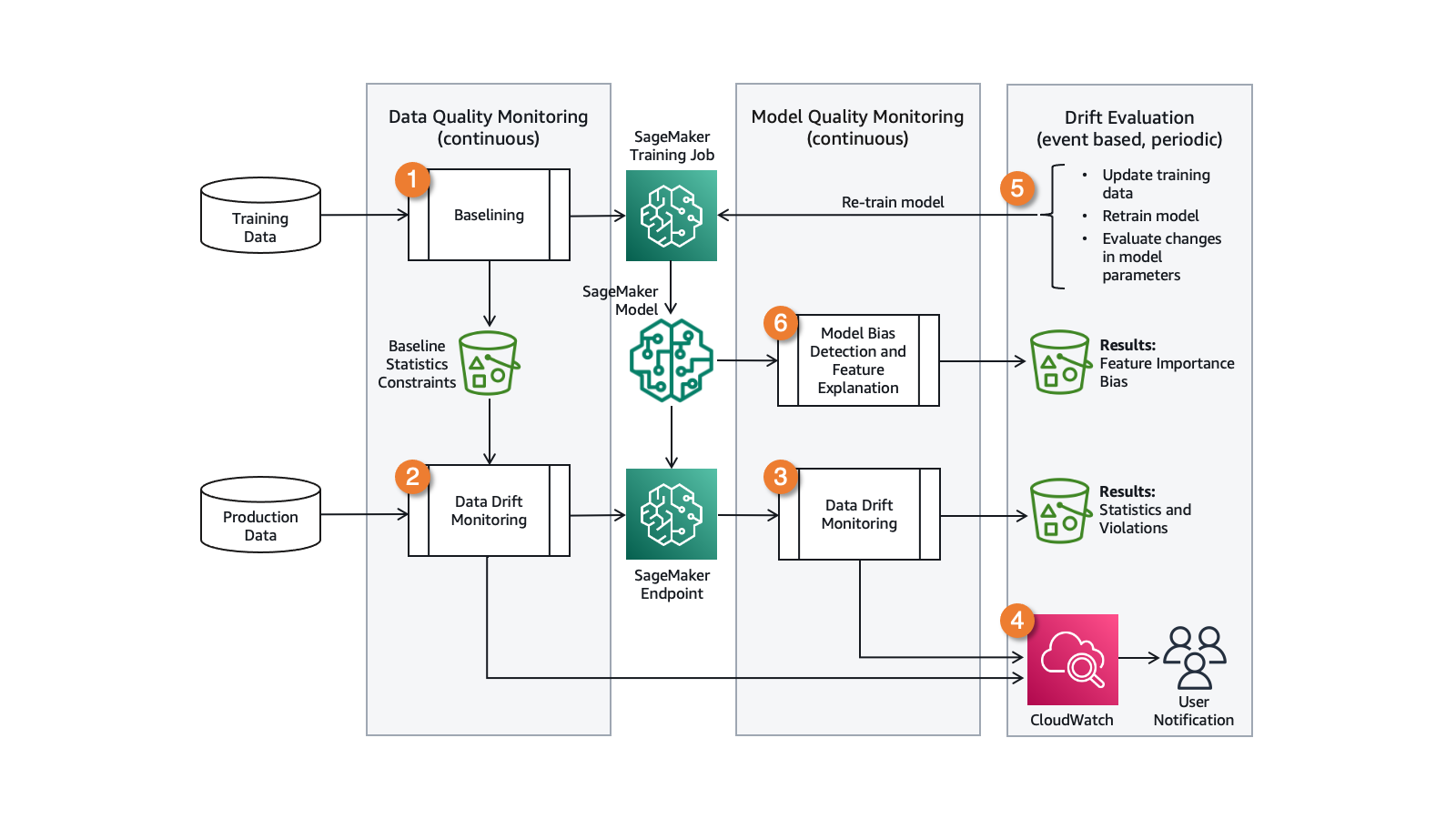


**Data quality monitoring** establishes a profile of the input data during model training, and then continuously compares incoming data with the profile. Deviations in the data profile signal a drift in the input data.

You can also detect drift through **model quality monitoring**, which requires capturing actual values that can be compared with the predictions. For example, using weekly demand forecasting, you can compare the forecast quantities one week later with the actual demand. Some use cases can require extra steps to collect actual values. For example, product recommendations may require you to ask a selected group of consumers for their feedback to the recommendation.

SageMaker Clarify provides insights into your trained models, including importance of model features and any biases towards certain segments of the input data. Changes of these attributes between re-trained models also signal drift. Drift evaluation constitutes the monitoring data and mechanisms to detect changes and triggering consequent actions. With Amazon CloudWatch, you can define rules and thresholds that prompt drift notifications.

Figure 1 illustrates a basic architecture with the data sources for training and production (on the left) and the observed data concerning drift (on the right). You can use Amazon SageMaker Data Wrangler, a visual data preparation tool, to clean and normalize your input data for your ML task. You can store the features that you defined for your models in the Amazon SageMaker Feature Store, a fully managed, purpose-built repository to store, update, retrieve, and share ML features.

The white, rectangular boxes in the architecture diagram represent the tasks for detecting data and model drift. You can integrate those tasks into your ML workflow with Amazon SageMaker Pipelines.

The drift observation data can be captured in tabular format, such as comma-separated values or Parquet, on Amazon Simple Storage Service (S3) and analyzed with Amazon Athena and Amazon QuickSight.

## How to build a feedback loop

The baselining task establishes a data profile from training data. It uses Amazon SageMaker Model Monitor and runs before training or re-training the model. The baseline profile is stored on Amazon S3 to be referenced by the data drift monitoring job.

The data drift monitoring task continuously profiles the input data, compares it with baseline, and the results are captured in CloudWatch. This tasks runs on its own computation resources using Deequ, which checks that the monitoring job does not slow down your ML inference flow and scales with the data. The frequency of running this task can be adjusted to control cost, which can depend on how rapidly you anticipate that the data may change.

The model quality monitoring task computes model performance metrics from actuals and predicted values. The origin of these data points depends on the use case. Demand forecasting use cases naturally capture actuals that can be used to validate past predictions. Other use cases can require extra steps to acquire ground-truth data.

CloudWatch is a monitoring and observability service with which you can define rules to act on deviation in model performance or data drift. With CloudWatch, you can setup alerts to users via e-mail or SMS, and it can automatically start the ML model re-training process.

Run the baseline task on your updated data set before re-training your model. Use the SageMaker model registry to catalog your ML models for production, manage model versions, and control the associate training metrics.

## Gaining insight into data and models

SageMaker Clarify provides greater visibility into your training data and models, helping identify and limit bias and explain predictions. For example, the trained models may consider some features more strongly than others when generating predictions. Compare the feature importance and bias between model-provided versions for a better understanding of the changes.

## Data Drift in Azure Machine Learning

Having learnt what is Data Drift and Concept Drift, wouldn’t it be great if we can detect the same? Fortunately, with the democratization of Machine Learning, we have tools at our disposal to ease our life. One of them is Azure Machine Learning. So, how useful is Azure Machine Learning in detecting Drift? We found no way to detect concept drift. However, we have a way to detect data drift.

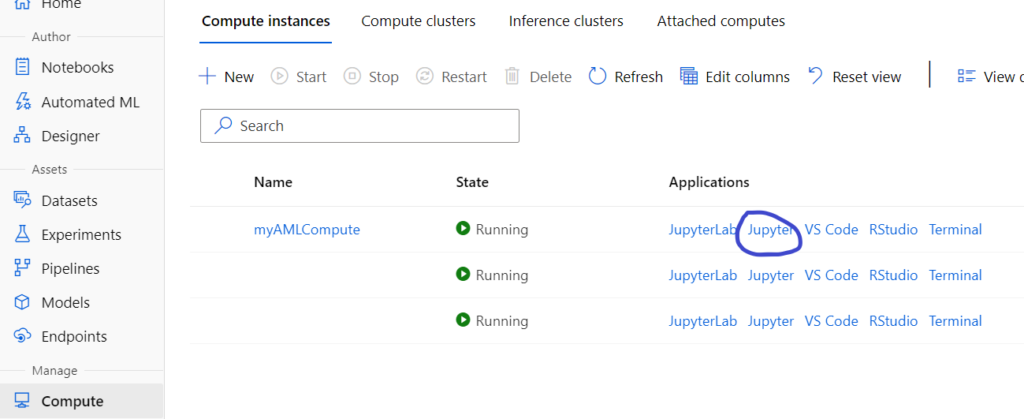
### Setting up

We assume you are familiar with Azure Machine Learning. If not, refer to this [**link**](https://azure.microsoft.com/en-in/services/machine-learning/). Azure Machine Learning is Microsoft Azure’s flagship ML as a service platform for training, deployment, orchestration, monitoring Machine Learning models. In this article, we will focus on the monitoring aspect of ML.

One feature of Azure Machine Learning is the Azure ML SDK, which is a custom python library by Microsoft. It comprises a library called *azureml-datadrift*.

But, before that, let’s perform some housekeeping. Assuming that you have created an AML workspace, first create a compute. You can either use a CLI or the Azure ML studio. To create your compute, refer to this [**link**](https://docs.microsoft.com/en-us/azure/machine-learning/how-to-create-attach-compute-studio#amlcompute).

After creating the compute, click on Jupyter. It will open a Jupyter environment.



Next, create your folder and open a terminal to clone this git [**repository**](https://github.com/MicrosoftLearning/mslearn-dp100). For detailed instructions, refer to this [**link**](https://microsoftlearning.github.io/mslearn-dp100/instructions/17-monitor-data-drift.html).

### Creating baseline dataset

Once you are ready with the setup, navigate to the folder in which you have cloned the above repository and open notebook *17 – Monitor Data Drift.*

The first step checks for the *azureml-datadrift* library.

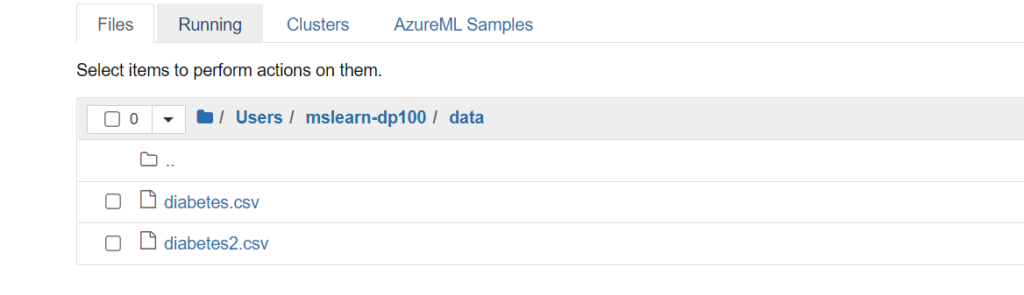
Next, connect to the Azure ML workspace using the following script.

Load the workspace from the saved config file

Now, comes the important step of creating a *dataset.* A dataset is like a data view created from a *datastore*. A datastore is a linked service or a connection to any data source like Azure Blob store, Azure SQL DB, Azure Data Lake store, etc. To read more about it, read this link on [**datastores**](https://docs.microsoft.com/en-us/azure/machine-learning/how-to-connect-data-ui#create-datastores).

We can create datasets using either code or studio. We will use the former.

Here, the toy data *diabetes* is in the data folder of the cloned repository. We will create a dataset using this data.



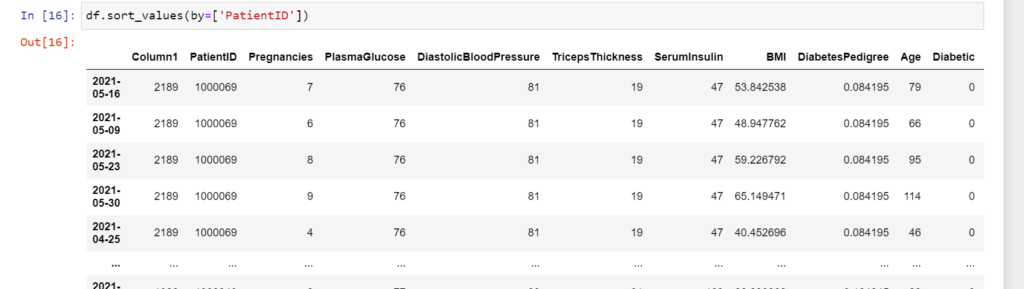
The next script will create a data set using the two files. First, it will upload the data to the workspace blob store and create a view on top of it.

Upload the baseline data  
Create and register the baseline dataset  
This creates our baseline dataset.

### Creating Target dataset

For simulation, we will create a target dataset and add certain drift to the data week over week for the past 7 weeks. Fundamentally, you create a time series dataset using this method. Below is the code to do the same.

Load the smaller of the two data files  
We'll generate data for the past 6 weeks  
Upload the files  
Register the target dataset  
To check how the target dataset looks like, convert it into Pandas Dataframe.



### Creating a Data Drift Monitor

Before creating a Data Drift Monitor, create a compute cluster and connect to it

Next, create a data drift monitor. This specifies the baseline and target dataset to be compared along with properties like frequency of monitoring, compute cluster, feature list, etc.

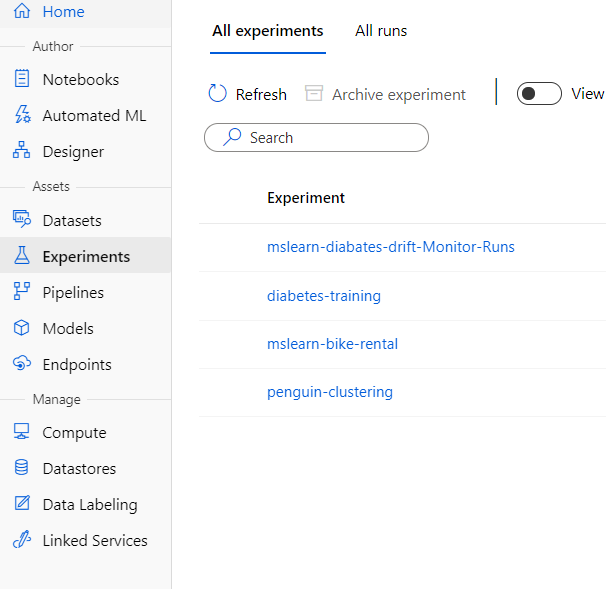
monitor

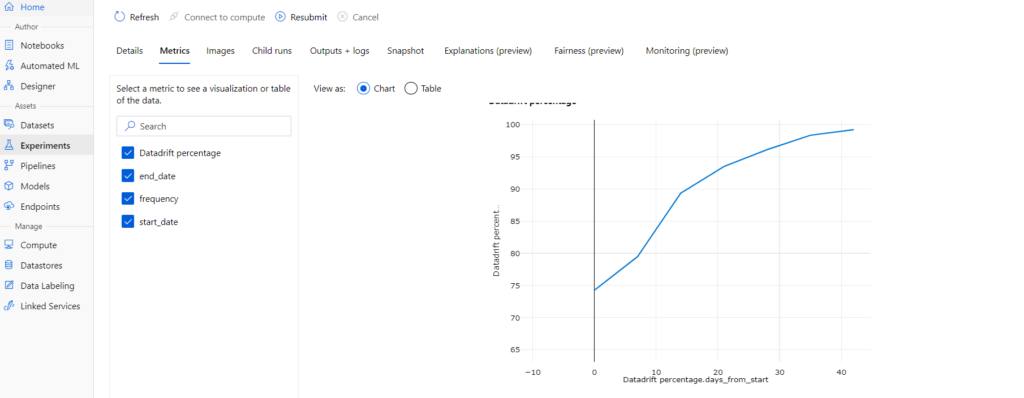
### Backfill the Data Drift Monitor

Now, in order to analyze data drift trends over time, you need to find a backfill job from your data drift monitor. This backfill job run will analyze how data has drifted over a time frame. The backfill function takes start date and end date as parameters and uses the time series data created above to generate the drift analysis.

Finally, after running the backfill job, Analyze the data drift:

However, you can view a graphical representation of the backfill run in Azure ML studio.





**Databricks**

## Step 1 – Installing required libraries

Importing and installing necessary libraries like evidently(tests, test\_suite, ) json mlflow numpy pandas datetime   
databricks.feature\_store(FeatureStoreClient)

**Step 2 – Fetching Data from the Feature Store**

Accessing data from feature store  
Create a FeatureStoreClient  
Specify the name of the feature table to read from  
Read data from the feature store  
converting to pandas df

* After importing all the required libraries, we will fetch our data from the Feature Store.
* To do that, we need to create a FeatureStoreClient() first.
* Then we need to give a table name from which we need to fetch the data.
* Then finally we will run the feature\_store.read\_table(table\_name) command to read the table from the feature store. This command will return a Spark Dataframe and we will store that in feature\_df.
* Then finally for easy operations, we will convert this Spark Dataframe to a Panda Dataframe. For this, we will use feature\_df.toPandas() command.

## Step 3 – Creating Reference and Current Data

creating reference data and current data

* Now we will split this data into 2 sets. Reference Dataset and Current Dataset.
* We will keep **90% of our data in the Reference Dataset** and the rest **10% in the Current Dataset**.
* We will name them **ref** and **cur** respectively.

## Step 4 – Let’s Check Data Drift in DataBricks

Running Drift Test  
Converting to JSON  
Creating a Dataframe of it for easy visualization

* Now in this step, we will create a TestSuite object and pass a list of all the tests we want to perform on our data.
* In our case, we just want to perform **TestNumberOfDriftedColumns()** test.
* We will run the test and pass our **reference** and **current dataset**.
* Then we are simply converting the results into a JSON file to create a Dataframe out of it for easier visualization.
* And then we are simply printing our Dataframe.

## Step 5 – Logging the results in a mlflow experiment

Logging this drift report in a mlflow experiment

* Now finally we will log all this information in a mlflow experiment.
* We will start a mlflow run using **with mlflow.start\_run() as run** and we will start logging parameters in that.
* Following are the parameters we are logging in this experiment:
  + **date**
  + **reference\_data**
  + **current\_data**
  + **n\_features**
  + **features**
  + **n\_drifted\_features**
  + **drifted\_features**
  + **drifted\_features\_p\_vals**