



Insights Hub

Predictive Learning Essentials

System Manual

05/2025

Welcome to Predictive Learning
Essentials Help

1

Introduction to Predictive Learning

2

Navigating Predictive Learning

3

Data Management

4

Environments

5

Model Management and Execution

6

Managing Docker Models

7

Running Docker Containers as Jobs

8

Jupyter Notebook Samples

9

Usage Metrics

10




Open Source Software

11

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.
 WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.
 CAUTION
indicates that minor personal injury can result if proper precautions are not taken.
NOTICE
indicates that property damage can result if proper precautions are not taken.


If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

 WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

1. Welcome to Predictive Learning Essentials Help.....	4
2. Introduction to Predictive Learning.....	5
3. Navigating Predictive Learning.....	6
4. Data Management.....	9
5. Environments.....	11
6. Model Management and Execution.....	14
7. Managing Docker Models.....	25
8. Running Docker Containers as Jobs.....	28
9. Jupyter Notebook Samples.....	30
10. Usage Metrics.....	41
11. Open Source Software.....	43

Welcome to Predictive Learning Essentials Help **1**

Welcome to Predictive Learning Essentials Help

The Predictive Learning Essentials (Predictive Learning) system provides the infrastructure and tooling that the data scientists need when creating statistical and machine learning algorithms for rapidly building and executing models. This helps to achieve the highest level of product performance and quality, translating into increased customer satisfaction, improved net promoter scores, and enhanced brand reputation.

When predictive learning is used to build machine learning algorithms and applied to IoT, service, field, and other data streams, it can provide unmatched insights into the future of your machines, processes, and products. This insight enables you to refine small and large aspects of your systems, foresee maintenance needs, and pinpoint root causes of process disruptions.

Predictive Learning facilitates:

- Easy access to Integrated Data Lake and IoT data.
- Integrating your organization's systems via Predictive Learning's public APIs.
- Rapidly building and executing models using statistical and machine learning algorithms.
- Import and run Docker models created outside Predictive Learning.
- Schedule models to run automatically for recurring reports and analysis.

With your valuable feedback, future releases will enhance further model development and execution experience.

Introduction to Predictive Learning

2

Introduction to Predictive Learning

Predictive (PrL) Learning enables data scientists to write or bring their own machine learning (ML) algorithms using statistical functions, transformations, and filtering to bring the most comprehensive and flexible means for accessing and working with both historical and near real-time data. PrL supports multiple data sources, such as:

- Integrated Data Lake (IDL) : Folders in IDL can be referenced from PrL.
- Internet of Things (IoT) : Asset data can be made available to PrL.

Features

PrL features allow users to:

- Validate and execute models within environments
- Create Jupyter notebooks and save them locally while environments continue to run
- Use Python scripts to read data from APIs
- Bring locally build models for execution as Dockers
- Schedule the model execution
- Use variety of data sources for Input and Output

Roles and Permissions

Predictive Learning adopts user roles and permission from the "Settings" application on the Launchpad. For more information on assigning user roles, refer to [user management](#) and [roles](#). If the "Settings" application does not appear on the Launchpad, please contact your administrator.



Users with the Predictive Learning Admin role are allowed to create, update, delete, or view a list of environment configurations based on the available configuration template, while users with the Predictive Learning User role can only select, start, or stop an environment to run your model based on the environment configurations created and saved by your administrator.

Navigating Predictive Learning

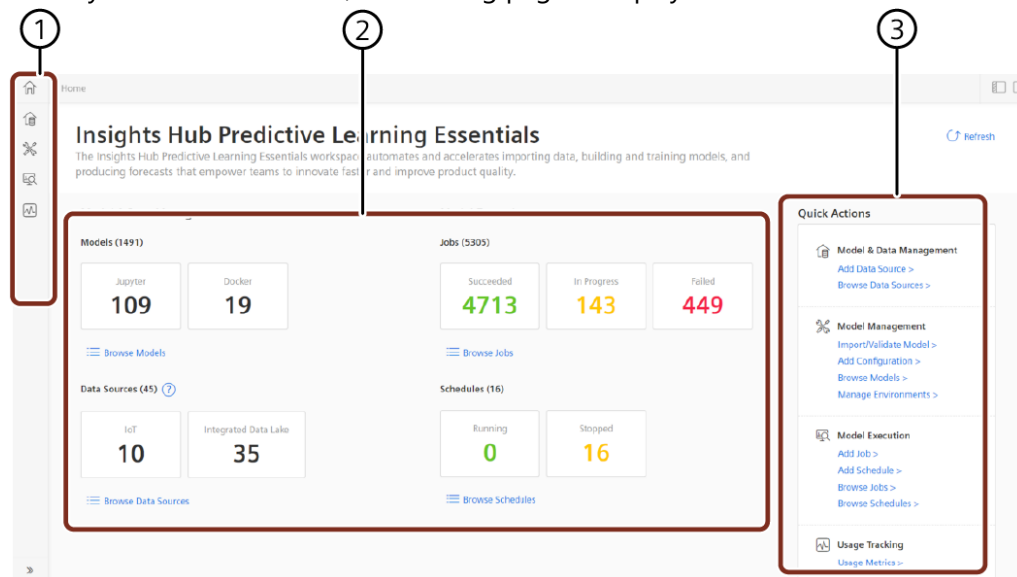
3

Navigating Predictive Learning

Predictive Learning (PrL) icon can be accessed from the Launchpad by clicking on the following icon:



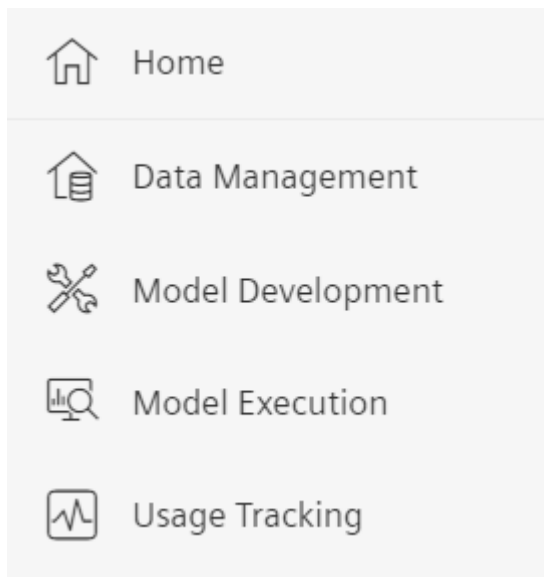
When you click the PrL icon, the landing page is displayed as shown in the below image:



- ① Navigation area
- ② Displays the number of available models, data sources, jobs and schedules
- ③ Quick Actions

Navigation area

The expandable menu bar in the left navigation area provides access to all PrL features. Click the double-chevron at the bottom of the window to expand and collapse the menu bar. Here is an example of the expanded menu bar and the features accessible through the menu items:



Feature Summary Blocks

The PrL landing page displays two blocks of features where summary information and browse links are displayed:

- Data Management
- Model Management and Execution

Data Management

Data Management facilitates bringing in data stored in data sources, like:

- Integrated Data Lake (IDL)
- Internet of Things (IoT)

Model Management and Execution

Model Management and Execution facilitates validating models and environments for working with your data. You can also import existing models for use in PrL. PrL supports these models:

- Jupyter Notebooks
- Docker Images

Browse Links and Quick Actions

"Browse" links appear beneath each feature; click a browse link to open a table of the items you've created or that have been shared with you.

The "Quick Actions" section provides access to all PrL features. Click any "Quick Action" link to jump to main PrL features.

Usage Tracking

Your PrL activity is tracked in real time, and you can view your usage data on the Usage Metrics page, which displays a table of usage history, including:

- Environment instance used
- Compute hours used
- Transaction time

Data Management

4

Data Management

Data Management features include creating and maintaining the data sources. For more information, refer to the following topics:

- [Creating a Data Source](#)
- [Searching Data Sources](#)

Data Sources

A data source is a configuration that specifies where to retrieve the data from and the amount of data to be fetched for model execution. The jobs user runs in PRL uses the data that is derived from data sources. A data source can be used as input data source or output data source by a job. Predictive learning enables a data scientist to consume data from multiple data storages by providing below data sources:

- Integrated Data Lake (IDL): A reference to the folders in IDL helps models to access the data to read and write to IDL.
- Internet of Things (IoT): A configuration to read the timeseries data from an Asset.

Example of Data Sources Page

Below is an example of the Data Sources Page that illustrates the possible actions that you can take with data sources:

The screenshot displays the 'Data Sources' page in the Siemens Xcelerator interface. The page has a sidebar with navigation icons and a main content area. The main content area is titled 'Data Sources' and includes a search bar and an 'Add Data Source' button. Below this is a table with the following columns: Name, Created By, and Asset Name. The table lists several data sources, including 'demo', 'IOTDL_65petInput', 'TestAsset', 'TestAssetName', 'Ds1Hr', '1689247838061_IOT2DL_Input_E2E', 'ITD5ju', and '168748257702_JM_IOT_PRL_Input_...'. On the right side, there is a 'Data Sources Overview' panel showing counts: Total 1155, IoT 333, and Integrated Data Lake 822.

Creating a Data Source

To create a data source, follow the below steps:

1. In the "Data Management" section, click "Add Data Source".

2. Select a data source location, and click "Next".
3. Enter a name for the data source.
4. For IoT, select the number of hours, and click "browse" to select an asset and aspect or, for IDL, select the folder and file.
5. Click "Save".

Actions Available for All Data Sources

All data sources allow users to search for a data source. Enter a data source name in the search bar at the top of the Data Sources table. Only data source names can be searched. As you enter search characters, the UI displays the Data Source locations that contain files with matching names. Click the Data Source location to expand it and view the actual files.

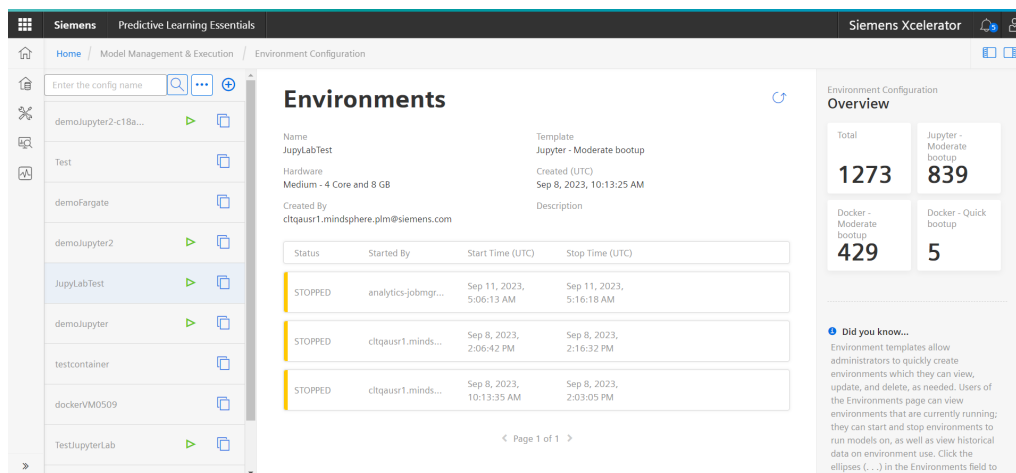
Environments

5

Environments

Environments are platforms that facilitate creating and running your PrL analytical models. Since Predictive analysis can involve a great deal of resource consumption, it is important to configure the right environment and instance type for the work you plan to do.

User Interface



Possible Actions with Environments

The Environment Configuration page displays a list of environments that you created or that were created for you. Use the icons on the right side of the Environments table for tasks, such as:

- **Viewing** environments: To view environments,
 - Select "Manage Environments" in the Quick Actions panel.
 - Select "Mine Only" from the ellipses drop-down list to filter environments created by your user only.
- **Searching** for an environment: Enter characters in the "Search by Name" field at the top of the Environment Configuration page.
- **Cloning** an environment: Click the Clone icon on the right side of the Environment Configurations table. This feature is not available in Local private cloud environment.

Creating New Environments

Begin creating an environment from the landing page by clicking the: - **Create a Configuration** link in Quick Actions, or - **Create a Configuration** link at the top of the Environment Configuration page.



- Create configuration is not available in local private cloud environment.
- Any prl user of the tenant can shutdown the environment irrespective of who started it
- Every environment started gets automatically shutdown after 8 hours if not stopped by user

About Environment Templates

PrL provides templates for many standard environment types. Templates contain settings and terms that are specific The environment template you select determines the instance types available. Here are the environment selections and the Instance Types available for them:

	Instance Name	vCPU	Memory (GiB)	AWS EC2	AWS Container Support	Azure VM	Azure Container Support	Private Cloud Container Support
1	XSmall	1	2	t2.small	Yes	No	Yes	Yes
2	Small	2	4	c5.large	Yes	Standard_F2s_v2	Yes	Yes
3	Medium	4	8	c5.xlarge	Yes	Standard_F4s_v2	Yes	Yes
4	Large	8	16	c5.2xlarge	Yes	Standard_F8s_v2	No	No
5	XLarge	16	32	c5.4xlarge	Yes	Standard_F16s_v2	No	No

About Instances

PrL provides flexible instance options so you can select one that fits the model and data you are running.

Instances can be thought of as virtual servers or environments and they are optimized for compute-intensive workloads. In PrL, each of the instance types combines various amounts of compute power, memory, and networking capacity, depending on the job you are running and the amount and type of data (structured, unstructured, or semi-structured) being analyzed.

The Instance Type drop-down list populates instance types available for the template you select.

How to Create a New Environment

To configure a new environment, follow these steps:

1. Click the "Create an Environment" link on the Environments page or click the "Create an Environment" link in Quick Actions.

2. Enter a Name for the new environment.
3. Enter an optional description.
4. Select a template from the "Template" drop-down list.
5. Select an instance from the "Instance Type" drop-down list.
6. Click "Save". The saved environment displays at the top of the Environments table.

How to Start or Stop an Environment

When you start an environment, it will run and automatically shut down after eight hours unless a user stops it before that time.

To start or stop an environment, follow these steps:

1. Click "Manage Environments" in Quick Actions. The "Environment History" page is displayed.
2. Select a configuration from the configuration drop-down list.
3. Click "Start" to start the environment, or click "Stop" to stop it.

Model Management and Execution

6

Model Management and Execution

This section describes the models available in Predictive Learning and the steps required to develop a model.

This also includes descriptions and step-by-step instructions for the following:

- Validating a New Model
- Importing a Model
- Actions You can Take with Models
- Executing a Model
- Advanced Docker Model Settings
- Executing Models on a Schedule

About Models and Model Types

Model is an analytical script written in form of Jupyter notebook, Zeppelin Notebook or bundled into a Docker image.

Model Types

The following types of models can be created in PrL:

- Jupyter Notebook
- Docker Image

Viewing Models

The "Models" page displays a table that lists information about the models you have access to. Each model's details as well as icons that allow you to perform various actions with models are displayed.



Maximum file size for the model uploads is limited to 50 MB.

Possible actions on models

Action icons appear on the right side of the models table. The following actions are possible on the available models:

- **Launch icon:** click on the launch icon to open a model.
- **Ellipses icon:** click the ellipses icon to view actions you can take, that include:
 - **Deleting a model:** displays a pop-up window to prevent unintended deletions.
 - **Editing a model:** displays a pop-up dialog in which the name and description can be updated. However, it is not possible to change the model type.
 - **Creating a version:** select whether to create a minor or major version (major = 2.0; minor=1.1), set an expiration date and model type, and browse for a model to upload, up to 50MB.
 - **Downloading a model:** downloads the model on your local system.

Importing a Model

When importing an existing model, follow the below steps:

1. Click "Import/Validate Model" on the "Models" page. The "Import/Validate Model" pop-up window is displayed.
2. Click the "Import Model" tab.
3. Enter a name and description.
4. Select an expiration date from the built-in calendar.
5. Select a model type from the Type drop-down list, or select "Browse" to locate and select a model file.
6. Click "Save".

Your imported model is displayed in the Models list.

Import/Validate Model

Validate Model

Import Model

Name *

Visual inspection model

Description

Expiration Date

25/11/2023 12:00 AM

Author

deepak.vijay@siemens.com

Type *

Jupyter Notebook

Select model file(up to 50MB) *

↑

Browse for Model File

Name	Type	Size
Visual Inspection model.ipynb		22.4 KB

Cancel

Save

Importing Docker Images

If "Docker Image" is selected as the model Type, then the "Browse for Model File" button will be replaced with the "Generate Token" button. This is necessary because of the way Docker images are imported into the application. In general, Docker images are developed locally or, it can be imported from an external source.

Import/Validate Model

×

Validate Model

Import Model

Name *

Select a Name →

Description

Expiration Date *

dd/mm/yyyy hh:mm

Author

@siemens.com

Type *

Docker Image

Generate Token for Docker

Generate Token

Image repository URI (with tag) *

Cancel

Save

Clicking the "Generate Token" will provide you with a temporary session credentials that will allow you to upload the Docker image to our Docker registry. This is required in order to allow secure and high-performance on any usages of your Docker image. Once your Docker image is uploaded, we will store a secure and private copy that can only be accessed by your tenant. In addition, we attach the necessary metadata to the image for execution, map its inputs and outputs, and enable the display of logs generated during its execution. Click "Generate Token":

Docker push

Steps for uploading a Docker Image in Model Management:

1. Please save this URI, it will be required to associate the image with an analytical model:


```
7e324168-432f-4526-ae89-3d5e995bf397.dkr.ecr.eu-central-1.amazonaws.com/integ/modelmanagement/r.../repository/7e324168-432f-4526-ae89-3d5e995bf397
```
2. On your local environment tag your model using:


```
docker tag <local_image_ID> 7e324168-432f-4526-ae89-3d5e995bf397.dkr.ecr.eu-central-1.amazonaws.com/integ/modelmanagement/r.../repository/7e324168-432f-4526-ae89-3d5e995bf397
```
3. On your local environment use the following command to log in your docker to the image repository (password expires at 2023-07-19T15:25:04.000):


```
docker login -u AWS-eyJWYXlsb2FkLjoiTXINTkUwOnhLR25TbkdWazdTMOF3NWxKNTJNVWlxMXV1TzJQUt1bUhyZDdzWDBCN3ZxekpFUGZodksybH0ZW8wL1F0cWkzTS9yZEt2amRsMEtFL2FsdUkYyVTMRjdjZxcnZgWkNjdjZGwV11Mk9mRzdmclUNNTlhaazZCV3BWBdVQOM1VqZ09FMjU2dDZTRlI3Z05PWEFjUG9VOHJEdEtEQURVYU1NjImdmhSaWVYQXc3anlnSDI3SGRmMFErZlNaRnJuWWxRT293ZHVQOQRuU21a5jFGZUp6bjITOWVWXR2THRWmN1R1Z0cjhySXJNVHgzOHdld0lwZjhoOaW8vdUxpYyRldGdYUklnQWdZYkZDOTA3L3daSDcxL3ZXVmdFMm5kYs3b3hh50ptVlFaMnh6b244TWlncmduUUYzMjYwZS9RR2hiUjBuN3VWQmN0elQ2dzhNUlVaZytlaTlKUktMbXRDV1BZlYlBzVWVhQbDdsbkVOZXJFbzJlOXFGM3RyRE96ZnBwZVJPYXQ0eUJzcy82Q1NKdF11WHI2R3lFRS55VdrOXNuc9LNU8rUktLaFc4M2ZaV2FFZ1d6b01uSkVvL2t1cnFQL29mZHFwBDQvemZ3b0F4WHI3bVR3cXJreWVCY1IsK1MxbWF4RXJ5RjYbZs3SjZlVUJkZkZRWIEjYOSXR4THE0WWcwQjJlGaWlZMjNOSW11Wm13dEJPU0F0bWwAMFECCFcaZDZAAAMF3VlOAYs3R5FhWkTic2Lk3E7LdWMDIDOCSEdse6OulvY7Bd4CldYzodiEIMlueEFQulJkL2DlC2bFMMNkKsFOa
```
4. On your local environment use the next command to upload your image:

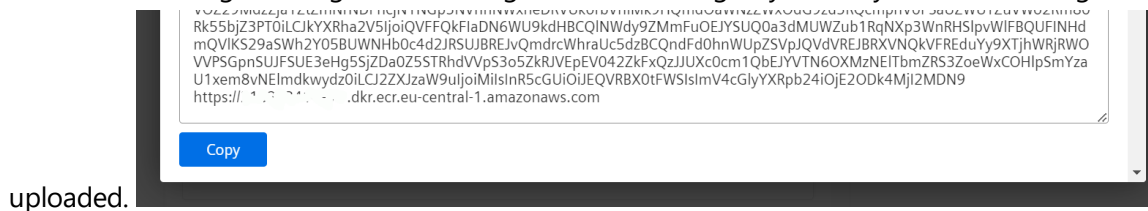

```
docker push 7e324168-432f-4526-ae89-3d5e995bf397.dkr.ecr.eu-central-1.amazonaws.com/integ/modelmanagement/r.../repository/7e324168-432f-4526-ae89-3d5e995bf397
```

After credentials were generated, you have 24 hours to create a Docker model in Model Management (through the "New Model" dialog), where you have the use the correct repository and tag URI.

Already built Docker images tend to be large files as they contain complete setups of operating systems with your own additions. This allows replicating environments that you have built and prepared, as well as their execution in most of the other external environments, such as public or private cloud environments. Docker images pack everything into an hierarchical structure (layers) and contain the metadata needed to interact with the exterior and with its own container engine. These images are built with a Docker compliant engine following a set of instructions that are described in a file named Dockerfile. A Docker engine compiles these instructions into a Docker image that can be distributed and instantiated as a Docker container by any container compliant engine. Building the image is often done with the help of a command line interface, and we are requiring the same Docker compliant command line to upload the image into our system. Therefore, the instructions in the pop-up are meant to be used with such a command line, but they target the Docker CLI.

1. This is meant for reference only, our system designates an URI that will indicate where your Docker image will be uploaded. This is immutable and attempting to change it, will make our system unaware of where you have uploaded your Docker image.
2. Tag your local image with the instructions from this step. You need to replace `<local_image_ID>` with your local IMAGE_ID, that you find by using "docker images" command. The `<local_image_ID>` can be found under the "IMAGE_ID" column.

3. Login to our Docker registry using the command provided at this step. You can expand the textbox containing the long session string to reveal the registry where your Docker image will be



uploaded.

4. After you get a successful login at the above step, you can start "pushing" (uploading) your local Docker image to our registry using the command from this step.

Now you can close the pop-up.

Note that your local image might have a "tag" that is usually the string that follows after the URL and is separated by a colon, like in "URL:tag". The tag is helpful to denote versions for example, like "v1.0.1" or "final-v1.0". If your tagged image at step 2 above includes this tag, then after closing the pop-up, you need to paste the URL stated at step 1 above in the "Image Repository URI (with tag)" field, including the tag as in the image below.

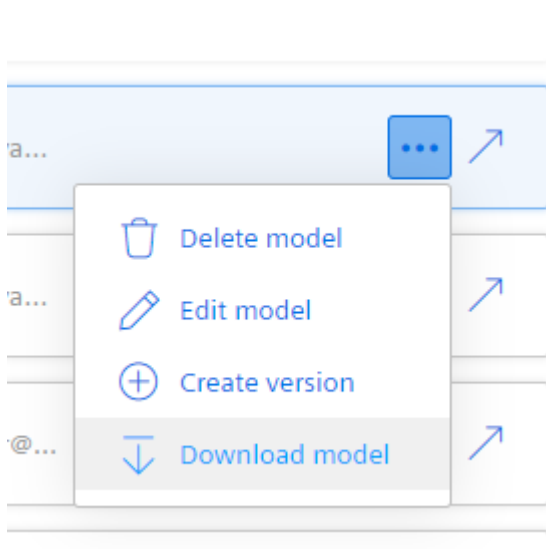
The screenshot shows a dialog box titled 'Generate Token for Docker'. It has a dropdown menu labeled 'Type' with 'Docker Image' selected. To the right is a blue button labeled 'Generate Token'. Below the dropdown is a text field labeled 'Image repository URI (with tag) *' containing the text 'men*/l... repository/3b58c5f6-224c-4442-811a-0564af14a41d:v1.0-fina'. At the bottom right are two blue buttons: 'Cancel' and 'Save'.

Click "Save" only after the upload your Docker image is complete.

Clicking "Save" will instruct the system to verify the Docker's image existence in our registry and its validity.

Downloading a Docker Image

You can download a previously uploaded Docker image by using similar steps as the ones above. Instead of pushing, you will be able to download (pull) a Docker image once you have a valid temporary session with our Docker registry. From the Models list, click the "..." button and use the "Download model" action menu. This will not download the actual image, but the access session in the form of a JSON file. From the JSON file, you can depict the keys needed to login to our registry.



Using Docker CLI, you can proceed using a similar "docker login -u AWS -p <password> <registry>" where <password> and <registry> are provided in the downloaded file. Once you logged in, you can use "docker pull <uri>" where <uri> is also provided in the downloaded JSON file.

Provided JSON file contains two types of authentication:

1. First part for Docker compliant CLIs under the "credentials" key, contains "user", "password". These can be used with Docker CLI to connect to our registry.
2. Second part, "providerCredentials" containing "accessKey", "secret" and "sessionToken" for AWS CLI. For the second option, you can use the AWS CLI tools to interact with your image. It provides additional but limited to AWS ECR functionality than Docker CLI (e.g. docker image scanning). The list of capabilities can be explored directly from the AWS CLI once you logged in the registry.

Validating a New Model

When validating a model, the process begins with the "Import/Validate Model" pop-up Window.

Import/Validate Model

Validate Model **Import Model**

Environment Configuration

TestJupyterLab ▼

Jupyter - Moderate bootup Medium - 4 Core and 8 GB

▶ Start

↻

Status	Configuration	Start Time (UTC)
RUNNING	6e231387-da81-4a59-94b6-eac0ead61d56	2023-10-26T10:21:56.179

↗

Cancel Save

To develop a new model, follow these steps:

1. Click "Add/Develop Model" on the Landing or Models page. The Import/Develop a Model pop-up window displays.
2. Make sure you are on the "Develop a New Model" tab.
3. Select an environment from the drop-down list and click "Start".
4. When the environment configuration displays a "Running" status, click the arrow icon. Your environment configuration opens in Jupyter Notebook.

Executing a Model

Executing a model involves running an analytical model against source data in a specific environment. Sometimes, this is called "running a job" in Predictive Learning.

To run a basic PrL job, you need to configure the following:

- **Input:** source (location) from which PrL reads the data for the job
- **Output:** location to which PrL writes the job results data
- **Model:** the mathematical model that runs against the input data
- **Environment:** start and stop environments for running jobs

Model Validation and Job execution environments file system

- Users cannot create new files or folders in their Jupyter notebook and Docker container, except in a specific directory during model validation and job execution.

- The file system is locked to read-only during model validation and job execution, except for the /tmp directory.
- Any new files or folders created during model validation and job execution should be placed in the /tmp directory only.

Executing a PrL Model

To execute your model, follow these steps:

1. Click "Add Job" in "Quick Actions".
2. Enter a name and an optional description.
3. Select a model from the drop-down list.
4. Select an environment configuration from the drop-down list.
5. Select a data source for input and output.
6. Before clicking the "Add Job" button, if you want to schedule the job to run, refer [Adding a Schedule to a Job](#).
7. Click "Add Job".



In case of LPC, each model will run in its own pod, and there is no maximum limit based on resource availability within the kubernetes cluster nodes.

It is a known limitation if a pod cannot start due to node unavailability, it remains in a "Pending" state, impacting operational efficiency and slowing down the application performance.

Executing Models on a Schedule

PrL jobs can be run ad-hoc or you can schedule them to run for a specific amount of time. If you want to run your job on a schedule, you have to set it up while you are creating the job.

Adding a Schedule to a Model Job

The "Enable Scheduling" toggle is located at the bottom of the "Add a Job" page.

To add a schedule to a job that you create, follow these steps:

1. Slide the "Enable Scheduling" toggle to the right.
2. Enter a number in the "Days" to run field.
3. Select a time increment from the drop-down list.
4. Click "Add Schedule".

Add a Job

Job Details

Name *
Enter the name
Enter the name →

Description
Enter the description

► Models

Configuration *
Select the environment configuration

Maximum Run Hours *
1

► Job data sources

► Advanced details

Enable Scheduling ☒

Days to Run * 7 Every 10 Minute

Run Job with schedule ☐ Run Job

Add Job Cancel



- Quick boot-up environments allow for a minimum scheduling interval of 5 minutes, whereas moderate boot-up environments permit a minimum scheduling interval of one hour.
- The Scheduling feature is available on the AWS platform only.

Advanced Docker Model Settings

When using Docker models, you can also customize the model using these additional settings:

- External reference IDs
- Start Command
- Maximum Run Hours
- Environment Variables (key/value pairs)
- Entry Point
- Scheduling

Adding Advanced Settings to a Docker Job

With the exception of the Maximum Run Hours field, all of the advanced option fields are optional. To add advanced options to a job, follow these steps:

1. Click "Advanced Details".
2. Enter an optional external reference ID.
3. Click the icon in the "Environment Variables" field to add Key / Value pairs in the pop-up window.

4. Click "OK".
5. Enter a Start Command and Entry point.
6. Enter a number in the "Maximum Run Hours" field.
7. Click "Add Job".

Managing Docker Models

7

Managing Docker Models

Docker Image Overview and Constraints

In addition to supporting Python (2, 3) and R models developed in Jupyter or Notebook, Docker is among the types of models PrL supports. Docker models have the advantage of being able to run any custom code, in any program language, and also Linux distribution preferred by users. The default operating system for all other types of models is the AWS AMI Linux distribution. There are few constraints related to data ingestion and persistence functions in the docker image setup. The Docker image persisted in Model Management:

- Consumes data from the /data/input folder
- Persists data in the /data/output folder

These folders are set up for automated execution by the Job Manager service, which retrieves the data for the job and persists it in /data/input, as well as the data written to the /data/output folder. Job Manager service places the data in the designated persistence service location, like Data Exchange, Predictive Learning Storage, or Integrated Data Lake (IDL).

About Creating a Docker Image to use in Model Execution

If you want to create your own Docker image to hold your code or model, you will require at a very minimum a Dockerfile. Usually, you inherit one of the public images that provides minimal support for your code or model. Here's a short example:

```
ARG BASE_CONTAINER=python:3.9-slim-bullseye
FROM $BASE_CONTAINER
```

```
USER root
```

```
RUN ["mkdir", "/tmp/input"]
RUN ["mkdir", "/tmp/output"]
RUN chmod 777 -R /tmp
RUN ["mkdir", "/data"]
RUN ["mkdir", "/data/input"]
RUN ["mkdir", "/data/output"]
RUN chmod 777 -R /data
RUN ["mkdir", "/iot_data"]
```

```
RUN ["mkdir", "/iot_data/input"]
RUN ["mkdir", "/iot_data/output"]
RUN ["mkdir", "/iot_data/datasets"]
RUN chmod 777 -R /iot_data
RUN ["mkdir", "/prl_storage_data"]
RUN chmod 777 -R /prl_storage_data
```

```
RUN pip install awscli
RUN apt-get update
RUN apt-get install wget -y
RUN apt-get install curl -y
RUN apt-get install jq -y
```

```
COPY . .
```

```
ENTRYPOINT ["python3", "./my_python_script.py"]
```

The lines that create folders `RUN ["mkdir", ...]` will create the proper folders for Job Manager to copy in input files, or to copy from results. If you do not pass in any inputs or outputs to your container when the image is executed as a job, then, these are not needed. In addition, if you want your Docker image to contain additional libraries, you can install these here using `RUN apt-get install ...`. These commands depend on your operating system, and they should be adapted to each. For detailed instructions on how to design your Dockerfile, refer [Dockerfile reference](#). For more instructions on building your Docker image, refer to [Docker build](#).

Persisting a Docker Image in Model Management

To create a new Docker model, follow these steps:

1. Click the "New Version" button on the Manage Analytical Models page. The "Create New Version" pop-up window opens.
2. Select "Docker Image" from the "Type" drop-down list. The system displays two Docker-relevant controls: a "Generate Token" button, and a text field.
3. Enter a complete Docker image repository path and tag version in the text field.
4. Do not click the "Generate Token" button. Read the information below, then proceed with the steps below.

Generate Token Option

Before a Docker Image can be associated with a Model, it must be brought into the Predictive Learning (PrL) service, which requires that you push the Docker Image to the PrL service repository.

Important: Time Constraints Involved with the Generating Token Option

Please note the following time constraints before proceeding:

- If you fail to complete all steps involved in generating a token within the 2-hour and 24-hour time windows discussed below, you will need to start the process over again. Also, once the two-hour window closes you cannot make updates to the uploaded Docker image.
- Within 24 hours of the token being generated you must link the Docker image, tag, and repository path to the model. If not completed within this time frame, the repository is automatically deleted.

Finish Generating a Token for a Docker Image

Follow the steps below and complete them within two hours of generating your token:

1. Click the "Generate Token" button. The service generates a unique repository.
2. Create a tag for your Docker image to serve as a reference to the upload version.
3. Log in and push the tagged Docker image into the service's repository using the Docker push command.

Next Steps

Once you upload your Docker image, you can:

- Associate the Docker image with your model by referencing the correct repository and tag in Managing Models.
- Safely close dialog window between uploading the Docker image and creating the Model Management entry.

Running Docker Containers as Jobs

8

Running Docker Containers as Jobs

You can execute Docker images in Predictive Learning (PrL), just as you can with any model stored in PrL.

About Docker Images

Refer to the following important points about pushing Docker images or when executing them:

- Do not store access data such as usernames, tokens, and secrets within Docker images, even though the storage location itself is secure.
- The Docker containers you create have limited external connectivity. For example, they can only download Python or R libraries. Therefore, you should include in the Docker image all the extra data and libraries they require.
- Jupyter notebooks created using the Predictive Learning environments may not work as is from customers local environment.
- Exposing ports from your Docker container has no effect because they run in isolation and other components cannot access them.

Additional Code Required in Dockerfiles

Depending on the type of input your container requires, the Dockerfile used in building the container requires a few extra lines of code to allow the environment to prepare input data, and to extract outputs.

For Data Exchange or Data Lake Input

The Dockerfile for Data Exchange or Integrated Data Lake input requires the following:

```
RUN ["mkdir", "/tmp/input"]  
RUN ["mkdir", "/tmp/output"]  
RUN chmod 777 -R /tmp
```

For IoT Input

The Dockerfile for IoT input requires the following:

```
RUN ["mkdir", "/tmp/input"]  
RUN ["mkdir", "/tmp/output"]  
RUN chmod 777 -R /tmp
```

For PrL Storage Inputs or Outputs

The Dockerfile for PrL storage inputs and outputs requires the following:

```
RUN ["mkdir", "/tmp/input"]  
RUN ["mkdir", "/tmp/output"]  
RUN chmod 777 -R /tmp
```

Jupyter Notebook Samples

9

Jupyter Notebooks Samples

Using Jupyter Notebooks

You can use different Insights Hub APIs such as Data Exchange, Model Management, and IoT Time Series from a Jupyter Notebook, a model development workspace in PrL. You will need valid access to the Insights Hub APIs you want to use in PrL.

Jupyter Notebook can be accessed from the Manage Environments page and the information provided here will aid you in launching a Jupyter Notebook.



- PrL environments do not retain the Jupyter Notebooks you have created. Ensure to save them on your local machine before you stop the environment. Clusters, on the other hand, do save new work and modifications applied to a Notebook.
- The dataset feature is going to be deprecated from all the platforms very soon. Currently, it is not available in Azure Private Cloud environment. For more information, refer to [datasets](#)

Model Validation & Job execution environments file system

- Users cannot create new files or folders in their Jupyter notebook and Docker container, except in a specific directory during model validation and job execution.
- The file system is locked to read-only during model validation and job execution, except for the /tmp directory.
- Any new files or folders created during model validation and job execution should be placed in the 9* directory only.

Checking your Configuration

When you begin working with your scripts, your environment will require a certain set of libraries.

Some libraries required to run the minimal services within the cluster come preinstalled. We have included links to the libraries in the Open Source Software topic in this Help.

Run these commands to examine installed packages from Jupyter:

```
%pip freeze
```

It is recommended that the required packages are installed at the beginning of the notebook and executed each time the cluster is started. The custom actions you take are not stored between stops and starts.

Installing Your Own Python Libraries



For the LPC environment, ensure you install an older version of the AWS SDK due to known issues [fix: MissingContentLength in boto3 version 1.36.1 · Issue #4398 · boto/boto3](#). Use a command like `pip install boto3"<1.36.0" awscli"<1.37.0"`

Please use the following whenever you need to install your libraries:

```
#upgrade pip and install required libraries
```

```
%pip install --upgrade pip
```

```
%pip install requests
```

```
%pip install pandas
```

```
%pip install pyarrow
```

Insights Hub API call from jupyter:

AWS

```
import os
import requests
import json
dlpath = '/datalake/v3/generateAccessToken'
gw = os.environ['GATEWAY_ENDPOINT'] + 'gateway/'
# increment_value = 1
headers = {
    'Content-Type': 'application/json'
}
payload="{ \"subtenantId\": \"\" } "
dl_url = gw + dlpath
response = requests.post(dl_url, data=payload, headers=headers)
#print(response.status_code)
dl = json.loads(response.text)
```

```

os.environ["AWS_ACCESS_KEY_ID"] = dl['credentials']['accessKeyI
d']
os.environ["AWS_SECRET_ACCESS_KEY"] = dl['credentials']['secretAc
cessKey']
os.environ["AWS_SESSION_TOKEN"] = dl['credentials']['sessionToke
n']

```

Not all external library repositories are allowed. If you require additional external sources for your project, please contact your organization's Predictive Learning administrator.

When an environment stops, all libraries and modifications performed on it are lost, which requires users to run the installation paragraphs each time Jupyter imports the note. Running the installation paragraphs insures that your machine is up-to-date.

Uploading data in Integrated Data Lake

AWS

```

# upgrade pip and install required libraries
%pip install --upgrade pip
%pip install requests --force-reinstall --upgrade
%pip install pandas --force-reinstall --upgrade
%pip install pyarrow --force-reinstall --upgrade
import datetime
import requests
import os
import json
import re
HEADERS = {
    'Accept': '*/*',
    'Accept-Encoding': 'gzip, deflate, br',
    'Connection': 'keep-alive',
    'Content-Type': 'application/json'
}
GATEWAY = os.environ['GATEWAY_ENDPOINT'] + '/gateway/'
OUTPUT_FOLDER = 'OUTPUT_FOLDER'
#Get a signed URL for down/upload of data. The function
#attempts for 5 times to obtain the URL and then raises an except
ion.
def getSignedURL(fileName, folder, attempt=0, upload=True):
    if upload:
        IDLpath = 'datalake/v3/generateUploadObjectUrls'
    else:
        IDLpath = 'datalake/v3/generateDownloadObjectUrls'

```

```

IDLFilePath = '/%s/%s' % (folder, fileName)
url = GATEWAY + IDLpath
body='{"paths": [{"path": "%s"}]}' % IDLFilePath
response = requests.post(url, headers=HEADERS, data=body)
try:
    return json.loads(response.text)['objectUrls'][0]['signedUr
l']
except KeyError:
    if attempt < 5:
        attempt += 1
        return getSignedURL(fileName, attempt, upload)
    else:
        raise Exception('Failed to get a signed URL')
!echo "This is a test!" >> test.txt
fileName='test.txt'
signedURL = getSignedURL(fileName, OUTPUT_FOLDER)
requests.put(signedURL, headers=HEADERS, data=fileName)

```

VPC/LPC

```

key_id = os.environ["CLIENT_ID"]
key_secret = os.environ["CLIENT_SECRET"]
tenant = os.environ["tenant"]
subTenant = os.environ.get("subtenant")
authServer = os.environ["TOKEN_ENDPOINT"]
coreGateway= os.environ["CORE_GATEWAY"]

class JsonWebToken():
    # Handles getting JWT from environment controller using HMAC.
    # Refreshes token whenever it gets expired.

    def __init__(self):
        self.token = self.__refreshToken()

    def __refreshToken(self):
        #Gets new JWT token using HMAC
        sign, timestamp = self.__getHmacSignature()
        headers = {
            'Authorization': 'HMAC-SHA256 Credential=' + key_id +
            '&SignedHeaders=host;x-msg-timestamp&Signature=' + sign,
            'x-msg-timestamp': timestamp
        }
        response = requests.get(authServer, headers=headers, veri
fy=False)

```

```

        print(response.status_code)
        return json.loads(response.text)["access_token"]

    def __getHmacSignature(self):
        # Creates HMAC Signature for getting JWT
        payload, timestamp = self.__getPayload()
        sign = hmac.new(key_secret.encode(), payload.encode(), ha
shlib.sha256).hexdigest()
        return sign, timestamp

    def __isExpired(self):
        # Checks whether existing JWT is expired or not.
        decoded = jwt.decode(self.token, verify=False, options=
{'verify_signature': False})
        current = int(datetime.now().timestamp()) + 10
        if decoded['exp'] <= current:
            return True
        return False

    def getToken(self):
        # Gets token if expired. This method should be used by us
ers of this class.
        if self.__isExpired():
            print("token is expired. getting new one...")
            self.__refreshToken()
            return self.token

    def __getPayload(self):
        timestamp = datetime.utcnow().strftime('%Y%m%dT%H%M%SZ')
        payload = "GET\n" + urlparse(authServer).netloc + "\n" +
timestamp + "\n" + tenant
        print(payload)
        if subTenant != None:
            payload = payload + "\n" + subTenant

        return payload, timestamp

    jwtobj = JsonWebToken()

    # Generate Upload URL

    # create random string
    random_string = ''.join(random.choices(string.ascii_lowercas

```

```

e, k = 10))
    print(random_string)
    filename = 'readme.json'

    path = random_string + "/" + filename
    body = json.dumps({
        "paths": [
            {
                "path": path
            }
        ],
        "subtenantId": "",
        "isMicrosoftRoutingEnabled": False
    })

    with open(filename, 'w') as f:
        random_content = "{ 'random_string': '" + ''.join(random.c
hoices(string.ascii_lowercase, k = 50)) + "' }"
        f.write(random_content)

    upload_url = coreGateway + "/api/datalake/v3/generateUploadOb
jectUrls"

    headers = {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
        'Authorization': 'Bearer ' + jwtobj.getToken()
    }

    response = requests.post(upload_url, data=body, headers=heade
rs)

    print("response code: " + str(response.status_code))
    print(response.content)

    signedUploadUrl = json.loads(response.content)['objectUrls']
[0]['signedUrl']
    print("Signed Upload URL: " + signedUploadUrl)

```

Reading data from IoT

Run the following commands to read data from IoT sources:

```
%pip install --upgrade pip
%pip install requests --force-reinstall --upgrade
%pip install awscli --force-reinstall --upgrade
%pip install pandas sklearn seaborn matplotlib joblib
import json
import io
import os
import datetime
import time
from dateutil import parser
import random
from threading import Thread
import requests
import pandas as pd
import tempfile

def read_iot(entity_id = "<<iot_entity_id_GUID>>",
            aspect_name = "<<aspect_name>>",
            tenant = "tenantname",
            max_results = 2000, #max is 2000
            from_dt = "2020-06-01T13:09:37.029Z",
            to_dt = "2020-07-01T08:02:27.962Z",
            variable = "pressure",
            sort = "asc"):
    if variable is not None:
        url = "?from=" + from_dt + "&to=" + to_dt + "&sort=" + sort + "&limit=" + str(max_results) + "&select=" + variable
    else:
        url = "?from=" + from_dt + "&to=" + to_dt + "&sort=" + sort + "&limit=" + str(max_results)
    #this is the IoT Timeseries API base URL
    TSpaath = 'iottimeseries/v3/timeseries'
    #this is the Predictive Gateway URL that handles authentication for your API calls
    gw = os.environ['GATEWAY_ENDPOINT'] + '/gateway/'
    headers = {
        'Content-Type': 'application/json'
    }
    iot_url = gw + TSpaath + "/" + entity_id + "/" + aspect_name + url
    response = requests.get(iot_url, headers=headers)
    return response

import pandas as pd
import tempfile
```



```

start = datetime.datetime.utcnow() - datetime.timedelta(days=70)
end = start + datetime.timedelta(days=30)
response = read_iot(entity_id = "<<iot_entity_id_GUID>>",
                    aspect_name = "<<aspect_name>>",
                    tenant = "tenantname",
                    max_results = 2000, #max is 2000
                    from_dt = start.strftime('%Y-%m-%dT%H:%M:%S.%f')[:-3] + 'Z',
                    to_dt = end.strftime('%Y-%m-%dT%H:%M:%S.%f')[:-3] + 'Z',
                    sort = "asc",
                    variable = None)
if response.status_code == 200:
    f = tempfile.TemporaryFile()
    f.write(response.content)
    f.seek(0)
    #we read the IoT data into a Pandas DataFrame
    data = pd.read_json(f.read())
    f = tempfile.TemporaryFile()
    f.write(response.content)
    f.seek(0)
    print(data.shape)
else:
    print(response.status_code)
    print(response.content)

```

Using Inputs from Job Execution

All job executions require parameters, and you can use any of the following:

- IoT (Internet of Things)
- IDL (Integrated Data Lake)

Using the first three from the list above ensures Job Manager copies the input to a temporary location available to your code. In Jupyter notebooks, there are three variables available:

```

inputFolder
outputFolder
datasetName

```

You can employ the Jupyter magic command `%store`, as follows:

```

%store -r inputFolder #-r specifies a read %store -r outputFolder %store -
r datasetName

```

The `datasetName` variable will only contain a value when the IoT input type is being used. The `inputFolder` will be prefilled by the job execution engine with a value pointing to the temporary location that holds the input files or data. That will be an S3 path on AWS or a blob storage on Azure. It does not contain the associated prefix like `s3://`. You can then use the `outputFolder` variable in a Jupyter notebook as in:

AWS

```
!aws s3 cp /tmp/mylocalfile.txt s3://$outputFolder+'/myfile.txt'
```

VPC

```
!pip install azure_cli
%store -r inputFolder
%store -r outputFolder

input = inputFolder
output = outputFolder
!az storage blob download --account-name prlstorageanls -c jobman
ager -f data.csv -n $input'/data.csv' --auth-mode login
!az storage blob upload --account-name prlstorageanls -c jobmanag
er -f data.csv -n $output'/output.csv' --auth-mode login
```

LPC

```
inputPath='s3://' + os.environ.get('inputFolder') + '/'
outputPath='s3://' + os.environ.get('outputFolder') + '/'
endpoint=os.environ.get('AWS_ENDPOINT_URL') + '/'

# Create the directory
directory_path = '/tmp/upload_dir'
os.makedirs(directory_path, exist_ok=True)
os.chmod(directory_path, stat.S_IRWXU)

!aws s3 cp $inputPath /tmp/upload_dir --recursive --endpoint-url
$endpoint

!aws s3 cp /tmp/upload_dir $outputPath --recursive --endpoint-url
$endpoint
```

For Jupyter notebooks, we do not provide a built-in library for loading the dataset, however, there are various ways to achieve this by using Python. If you encounter any issues in loading your dataset, feel free to contact us for guidance.



Both `inputFolder` and `outputFolder` variables are remote storage paths, not local folders; therefore most of the commonly-used file functions do not work against it; however, CLI and shell commands will use them as long as they use the correct prefix. For Python or Scala libraries that can work with remote storage services, we recommend checking the documentation for each respective library; for example, the pandas Python library is able to save and read files from AWS S3 storage.

Using exported IoT Datasets

We prepare the environments that you start with our `prlutils` library that handles reading of parquet dataset files into a Pandas Dataframe. Please use the bellow snippet to show the available datasets and then load a single dataset.

```
from prlutils import datasetutils
import boto3
import os
import json
import s3fs
```

```
du = datasetutils.DatasetUtils()
datasetnames = du.get_dataset_names()
print('Dataset names: ' + str(datasetnames))
#ds.shape
```

You will get a list of datasets like in:

```
['test_asset_2',
 'Last30DaysAsset2Filtered',
 'Last30DaysAsset2']
```

You can load the dataset you want with the:

```
ds = du.load_dataset_by_name(datasetnames[0])
ds
```

and check its data immediately:

	_time	time	flow	flow_qc	pressure	pressure_qc	temp	temp_qc	pk	rk	tll	_itime
0	1586060055730	NaN	105	NaN	35	NaN	52.0	NaN	None	NaN	NaN	1586166501255
1	1586060066750	NaN	105	NaN	36	NaN	66.0	NaN	None	NaN	NaN	1586166511032
2	1586060077330	NaN	116	NaN	36	NaN	51.0	NaN	None	NaN	NaN	1586166521100
3	1586060088248	NaN	117	NaN	37	NaN	64.0	NaN	None	NaN	NaN	1586166530858
4	1586060098822	NaN	109	NaN	36	NaN	57.0	NaN	None	NaN	NaN	1586166540960
...
974	1586304058495	NaN	169	NaN	0	NaN	0.0	NaN	None	NaN	NaN	1586262761348
975	1586304346027	NaN	149	NaN	0	NaN	0.0	NaN	None	NaN	NaN	1586264739121
976	1586304738584	NaN	149	NaN	0	NaN	0.0	NaN	None	NaN	NaN	1586264745255
977	1586304918584	NaN	149	NaN	0	NaN	0.0	NaN	None	NaN	NaN	1586264750490
978	1586305222970	NaN	149	NaN	0	NaN	0.0	NaN	None	NaN	NaN	1586264755822

979 rows x 12 columns

Then, you can use your dataset just like with any other Pandas Dataframe:

```

filteredDataset = ds[ds['temp']>60]
print("Number of entries AFTER filtering: "+ str(filteredDataset.shape))
try:
    path = "s3://" + outputFolder
    filteredDataset.write.csv(path)
    filteredDataset.write.csv('s3://prl-storage-216273414971/prlteam/data/')
except:
    print('Output folder is None.')
else:
    print('Filtered dataset written to outputFolder' + outputFolder)

```

If you encounter issues with using our library, make sure that the libraries used by our Datasets utility does not conflict with your previously installed Python libraries. Our utility library uses the following:

```

%pip install pyarrow fastparquet fss pec s3fs boto3 awscli

```

More About Jupyter Notebook

Jupyter is a powerful tool that allows multiple customizations and languages. These resources can help you explore further:

- <https://jupyter.org/documentation>
- <https://jupyter-notebook.readthedocs.io/en/stable/>
- <https://ipython.readthedocs.io/en/stable/interactive/magics.html>

Usage Metrics

10

Usage Metrics in Predictive Learning

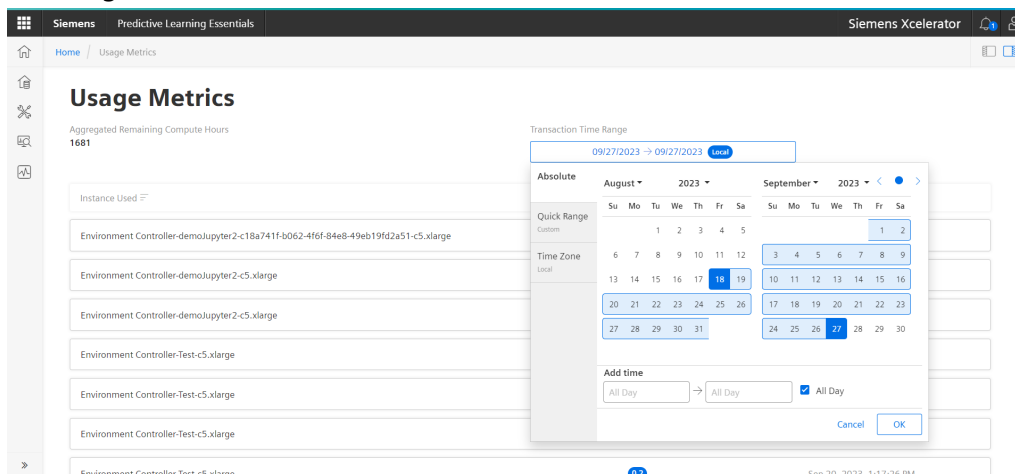
The Usage Metrics page shows the number of compute hours that remain for your organization and also lists your individual transactions and the number of compute hours you have used.



Usage metrics feature is not available on LPC platform.

Usage transaction Time Range

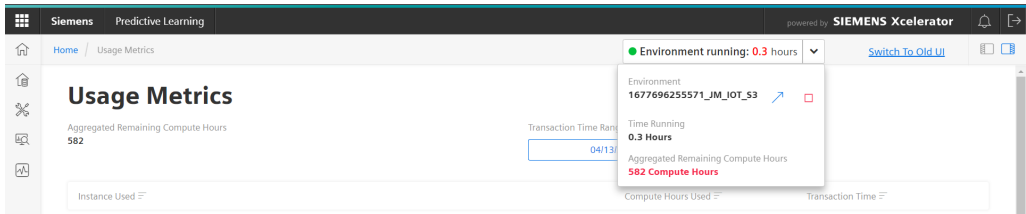
Here is an example of the the calendar display that opens when you click the calendar icon. Click a starting date and an ending date/time for the environment instances you want to display in the Usage Metrics table:



Environment Hours Left

Click the drop-down arrow in the "Environment Hours Left" field to see additional details about compute hours remaining and time left for a selected environment. You can also start and stop an environment using the "Start/Stop Environment" toggle in the Environment details drop-down list.

Here is an example of the Environment Hours Left" drop-down display:



Compute hours

"Together with the application you get an amount of a consumption metric named compute-hours. During development and job execution or job scheduling the system consumes these hours based on the load and complexity of the used environments. The accumulated load values will be subtracted from the pool of compute-hours. Additional capacities can be bought using the store."

Open Source Software

11

Open Source Software

Features of Predictive Learning / PrL Essentials Environments provided in the Analytics Workspace expose installed libraries through the Apache notebooks and Jupyter Notebook developer tools. Predictive Learning continues to support Python 2 libraries to keep the old notebooks running, but we strongly encourage users to upgrade their scripts to Python 3. With upgraded scripts, when Python 3 environments start up, they come with a minimal set of libraries, while users can continue to install their required libraries.

For Python 2 environments and, in accordance with to the GNU Lesser General Public License, this table displays the available libraries. The open source resources code is available via the links.

Library Name	Version	Source Code
awscli	1.18.14	awscli
boto	2.49.0	boto
docutils	0.15.2	docutils
scikit-image	0.13.1	scikit-image
scikit-learn	0.19.1	scikit-learn
python36-sagemaker-pyspark	1.2.1	python36-sagemaker-pyspark
PyYAML	5.3.1	PyYAML
requests	2.24.0	requests