Prescription Mashup: Setup and Use

1 Introduction

The Prescription Mashup is designed to demonstrate making a Prescription (i.e. optimizing a result) using Thingworx Analytics.

The mashup uses a simple dataset showing different inputs to concrete (e.g. water, coarse aggregate, fine aggregate, age etc) and the resulting compressive strength. A prediction model predicts compressive strength based on different inputs entered. Furthermore an optimized score is calculated based on adjusting some values. The prescribed adjustments are also shown.

The dataset was taken from this site:
https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

2 Thingworx Version

The entities were created and tested with TWX version 9.0

3 Naming Convention

In this procedure we will name all datasets, models and entities starting with our initials, for example TS. for Tanveer Saifee. For example my dataset might be named TS.concrete

In the explanations below we use XX as a placeholder to refer to your initials.

4 Import Entities

This step only needs to be performed if you are missing these entities: Shared.ConcretePrescMashup, Shared.ConcreteHelper, Shared.ConcreteDT

Import entities from folder 1-prediction-methods-entities

Now import entities from folder 2-prescriptive-scoring-entities

5 Copy your entities

Duplicate the following entities, replacing “Shared” with your initials:

Shared.concretePrescMashup

Shared.ConcreteHelper

Shared.ConcreteDT

So you should now have the following entities (where XX represents your initials):

XX.concretePrescMashup

XX.ConcreteHelper

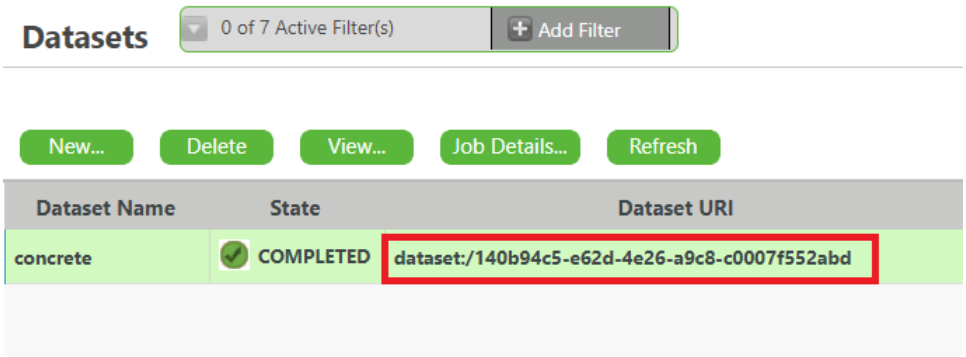
XX.ConcreteDT

6 Set Dataset URI

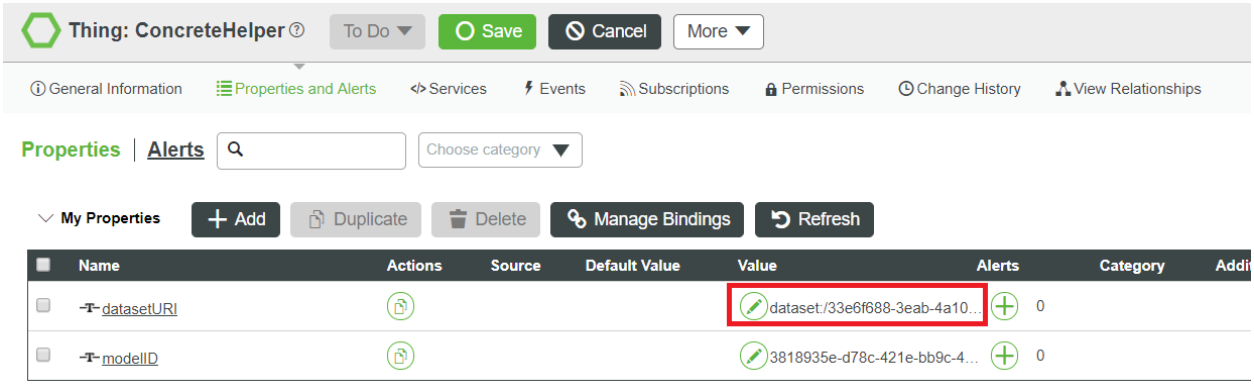
The following sections assume that you have created the concrete dataset and model as per the Prediction Methods Mashup training guide.

Open Analytics Builder … Data …

Copy Dataset URI to the clipboard as shown below:



Navigate to Thing: XX.ConcreteHelper and set datasetURI value to the value in your clipboard as shown below:



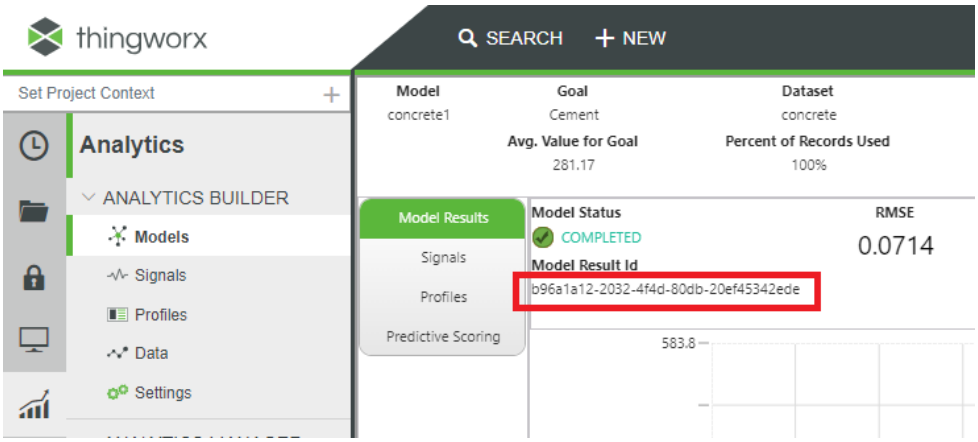
An example value for datasetURI is **dataset:/140b94c5-e62d-4e26-a9c8-c0007f552abd**

Save changes to Thing: XX.ConcreteHelper.

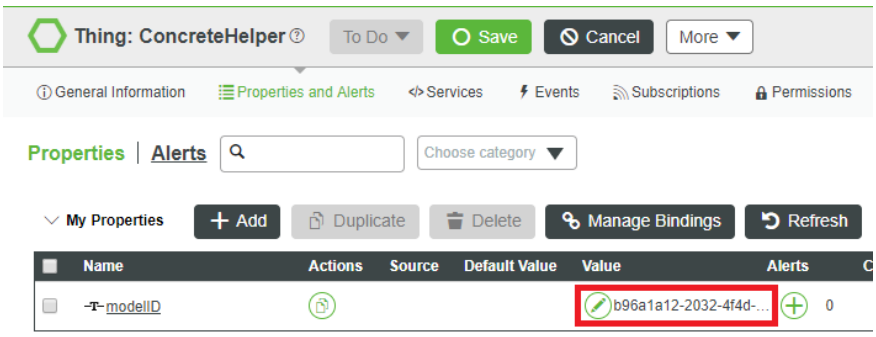
7 Set Model ID

Open Analytics Builder … View Model …

Copy Model Result Id to the clipboard as shown below:



Navigate to Thing: XX.ConcreteHelper and set modelID value to the value in your clipboard as shown below:



An example value for modelID is **db17b49d-43a6-4685-8aa4-bc1944df0ae3**

Save changes to Thing: XX.ConcreteHelper.

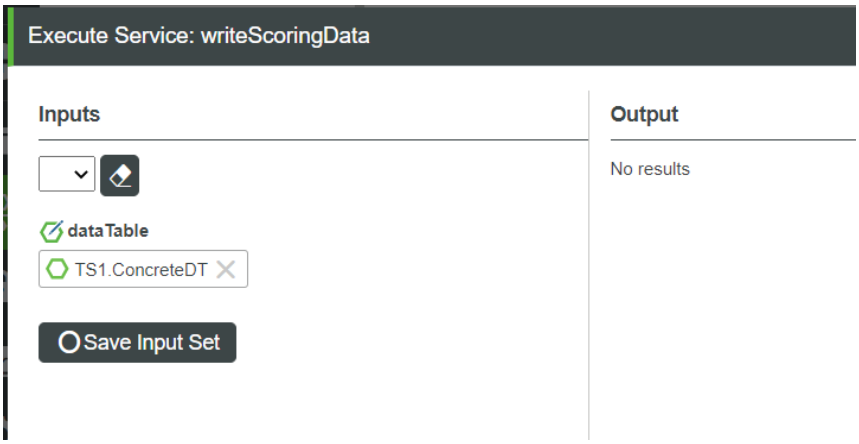
8 Write Scoring Data

We now need to write some records for scoring to your data table to make it easy to use the mashup (by loading pre-set values). In order to do this you need to execute the following service:

XX.ConcreteHelper.writeScoringData

For the input dataTable, select your data table XX.ConcreteDT

The service does not display any outputs after completion, see below:



You might get the following error:

“Error executing service writeScoringData. Message :: Entry With This Key Already Exists”

This can be ignored as it indicates that data has already been loaded to the data table.

9 Using the Mashup

You can open the mashup by adapting this URL and then pasting into your browser:

https://<your-servername:port>/Thingworx/Mashups/XX.concretePrescMashup

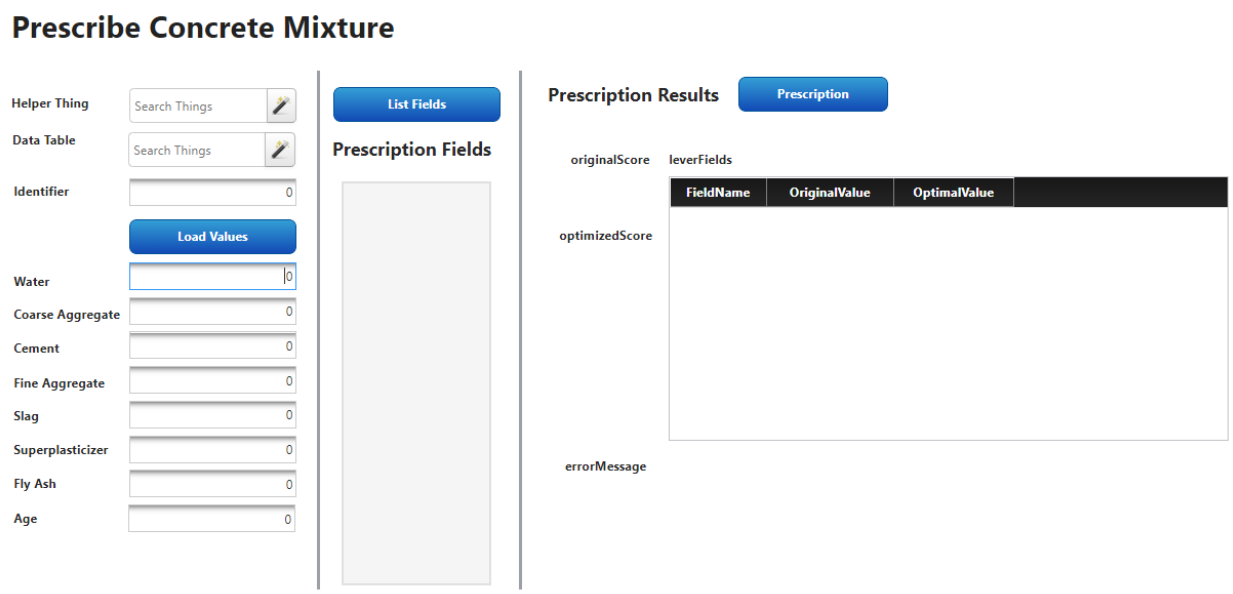
or

http://<your-servername:port>/Thingworx/Mashups/XX.concretePrescMashup

(depending whether you’re using HTTPS or HTTP)

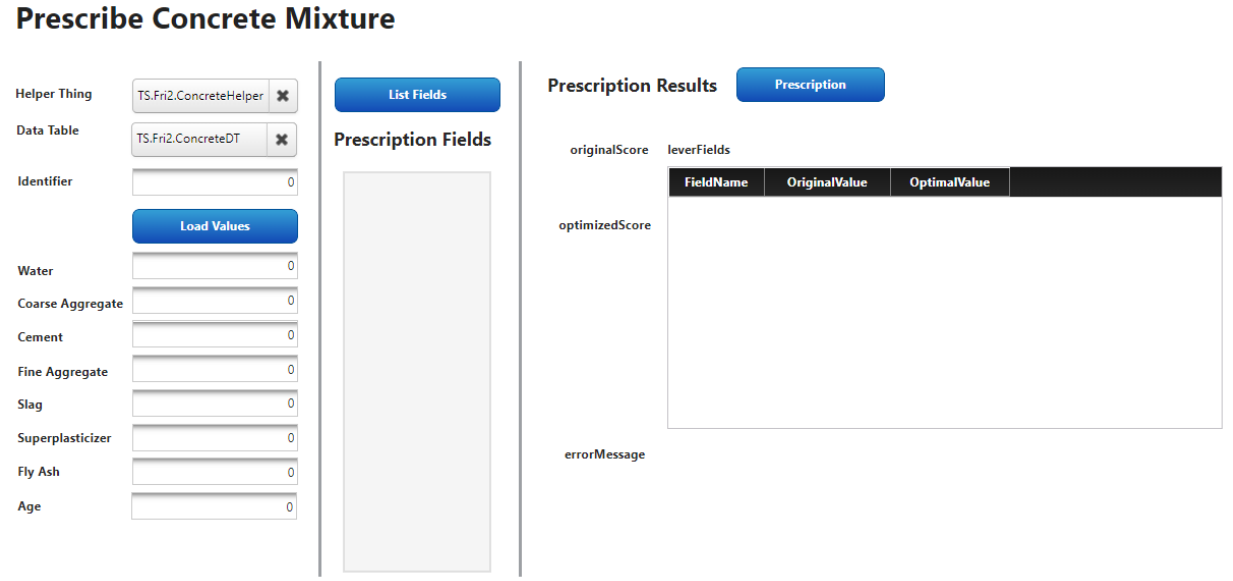
(Note that the mashup name is case sensitive, so check the exact naming in case it’s not found)

You should see the blank mashup load similar to what’s shown below:



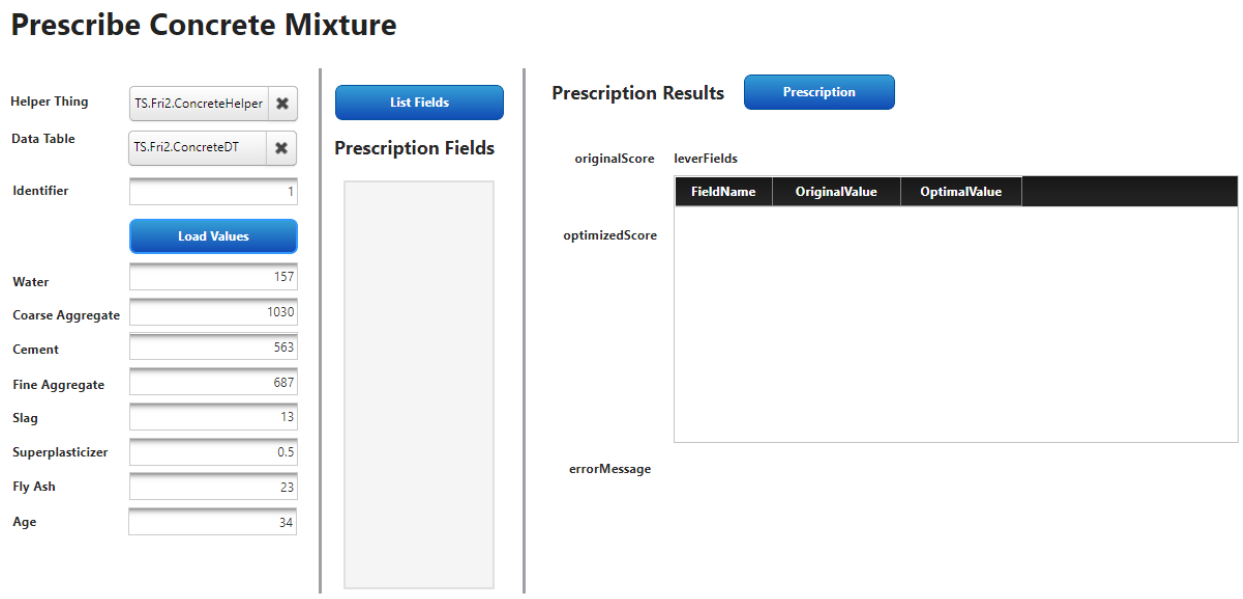
9.1 Select Entities

Now select your Helper Thing and Data Table:



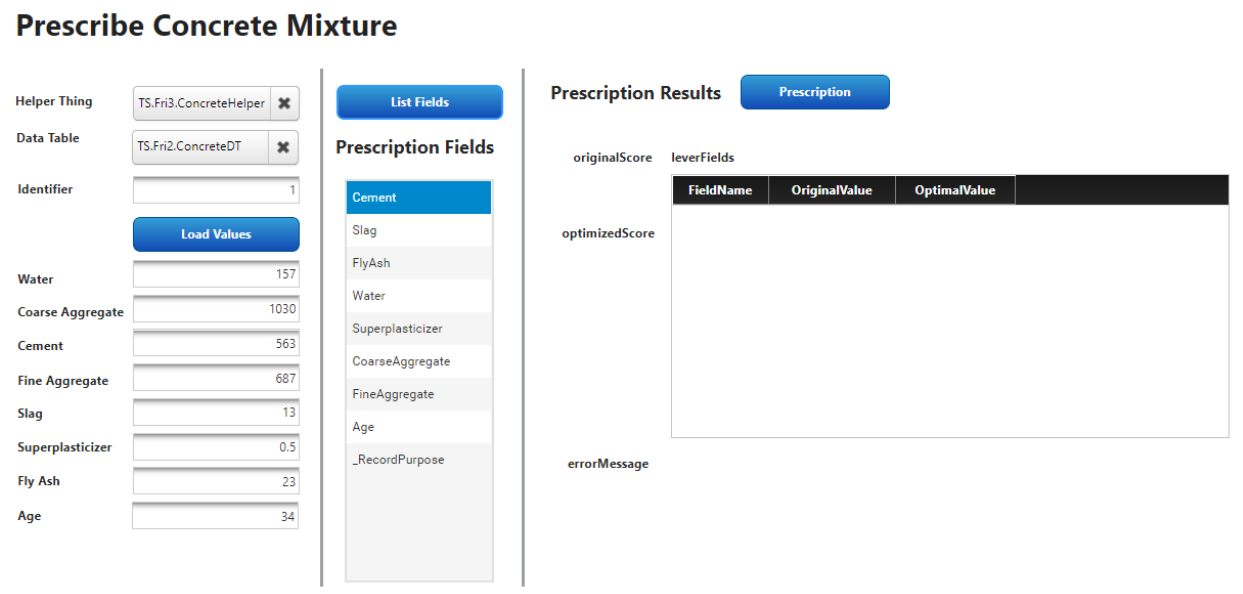
9.2 Load Test Data

Input an identifier between 1 and 20 and click Load Values to load up some test data (to avoid having to type out all fields). You can change these values as needed for your testing.



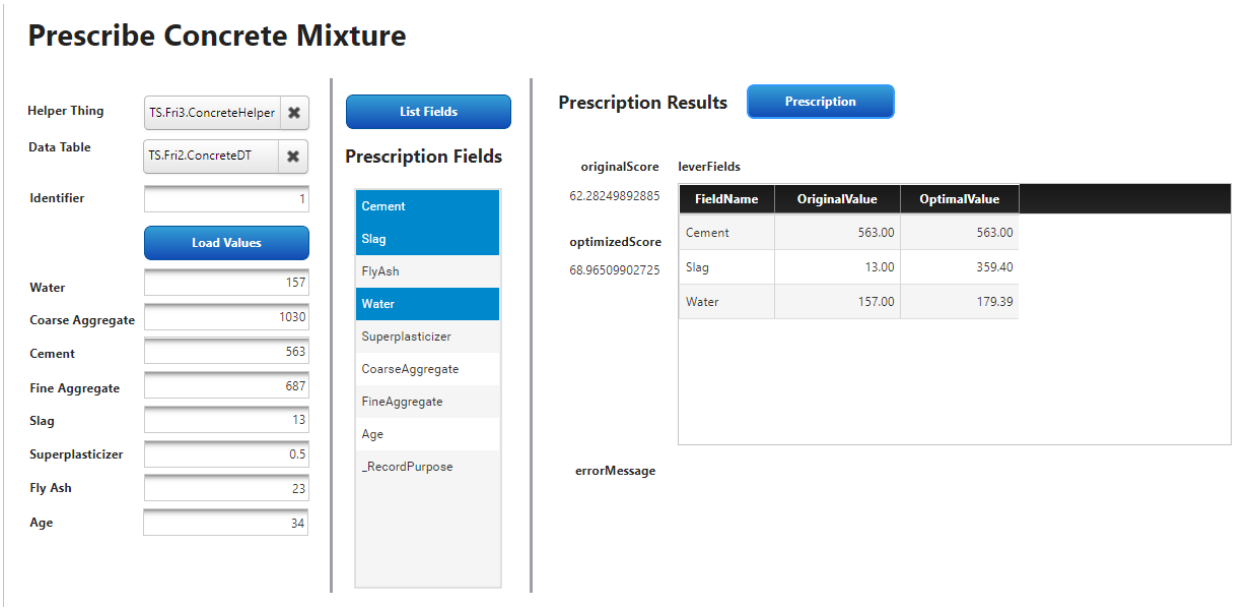
9.3 List Lever Fields

Now you can click List Fields to see the fields available on the prediction model. Select one or more of these fields (using Windows:Ctrl + Click, Mac: Command + Click).



9.4 Generate Prescriptions

Now click on the Prescription button to see how far the prediction can be optimized by making changes to the lever fields you selected. The system will determine, based on the fixed inputs (i.e. unselected fields) and the variable inputs (i.e. the selected fields), optimal values for the selected fields to maximize the Compressive Strength of the concrete.



Here's an explanation of the fields shown in the Prescription Results:

9.4.1 Original Score

This is the prediction for Compressive Strength based on the input values you provided

9.4.2 Optimized Score

This is the increased Compressive Strength score which would result from making changes as recommended in the Lever Fields table.

9.4.3 Lever Fields

This is a table showing some fields with their original values (i.e. the values you provided on the far left of the screen) and the optimal values (the results of prescription). If you change the input values to these optimal values then you will maximise the outcome you are predicting: Compressive Strength.

Fields which you selected in the middle of the screen for optimization are included in the Lever Fields output table.

9.4.4 Error Message

This field displays any error from the Prediction microservice.

10 Main Services

In order to use this functionality in your own environment you may want to copy the services to use as code samples. Below are overviews of the services called to enable the functionality in the mashup.

10.1 ConcreteHelper.writeScoringData

Writes static data into a data table for loading later into the mashup (to avoid typing out all values).

10.2 ConcreteHelper.listFields

Runs when the mashup loads. It retrieves all field names from the dataset excluding field CompressiveStrength (which is the goal). These fields are used in the mashup to populate a list so that the user can select which fields to use for optimization.

10.3 ConcreteHelper.prescribeStrength

Gets the input values for prediction as well as the selected fields for optimization as inputs. Then calls RealtimeScore on the Prescriptive Microservice and returns all results including optimization values.