We will first setup all the services that are required in our data processing steps. The services need Python and Java installed in the machine where we host the services.

1. addDFT Service
    a. Install the web.py framework. Follow the steps in the official link http://webpy.org/install
    b. Open the console / terminal
    c. Locate the addDFTREST.py script files in the python script directory under KarmaProcessing and execute the script to start the web service
        python addDFTREST.py 8080
    The command starts the addDFT service on port 8080
2. getLabel Service
    a. Open a new terminal session
    b. Locate the getLabelPOSTService.py script files in the python script directory under KarmaProcessing and execute the script to start the web service
    c. Execute the getLabelPOSTService.py script
        python getLabelPOSTService.py 8083

3. svmService
    Follow the steps in setting up the Karma SVM service.
    https://github.com/InformationIntegrationGroup/karma-svm-service

Steps using Karma

The steps using karma are divided into two categories:

1. Karma setup
    These are tasks that are performed only the first time. They include modeling the different services – addDFT, getLabels, and svmTraining, as well as modeling the individual dataset – AccelerometerSensor and LocationProbe. All transformations and processing done here is recorded by karma and can be played automatically for the other data sets.

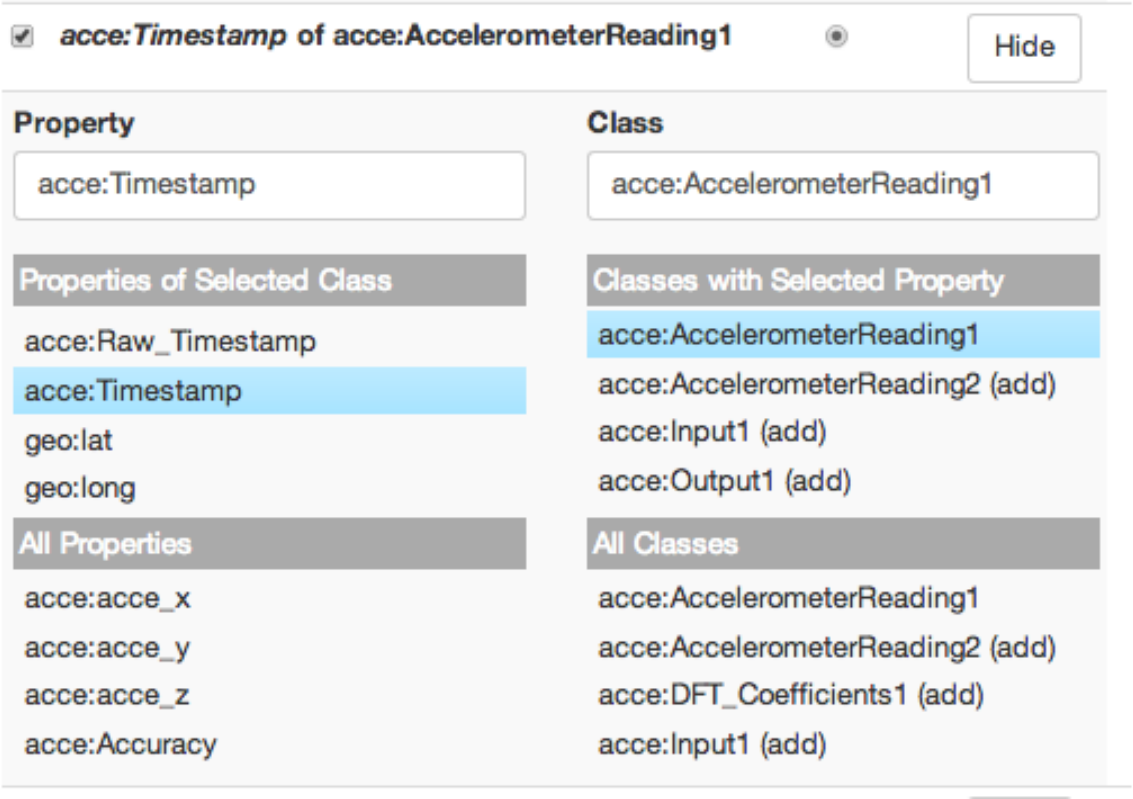
2. Karma execution
    The Karma execution tasks are ones that are repeated for each datasets. They mainly include tasks like service invocation, joining data sets, and publishing RDFs.

Now we will discuss each part separately, starting with the setup process.

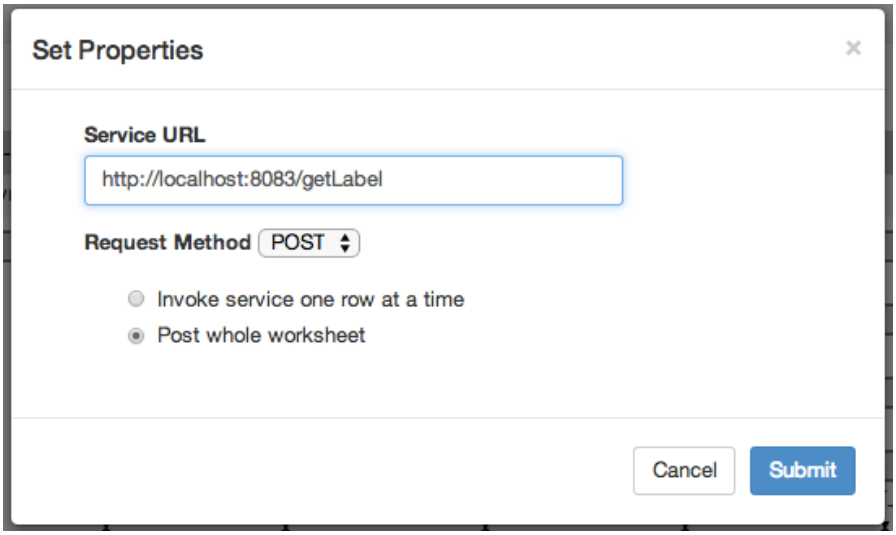
Karma Setup

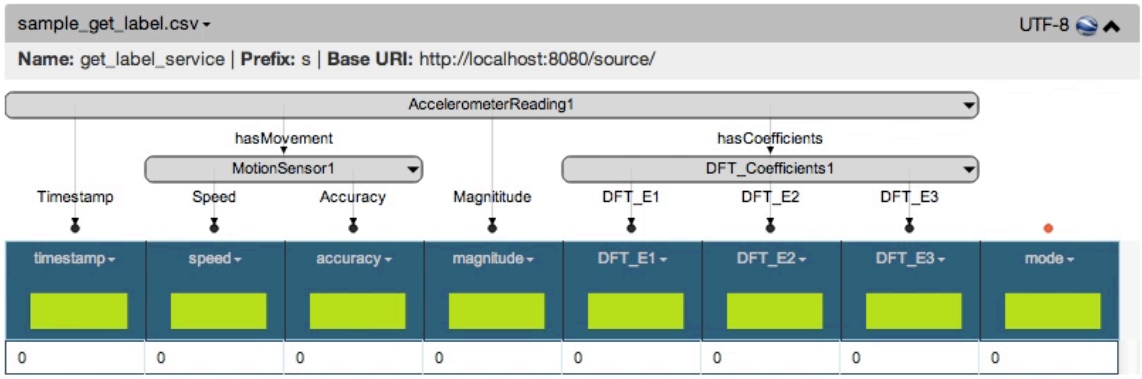
1. Create a sample CSV file with columns – timestamp, speed, accuracy, magnitude, DFT_E1, DFT_E2, DFT_E3, mode. Add one additional row that contains values. The values can have any random numbers

2. Modeling the getLabel service
    a. Load the sample CSV file in karma
    b. Model each column for its semantic type.

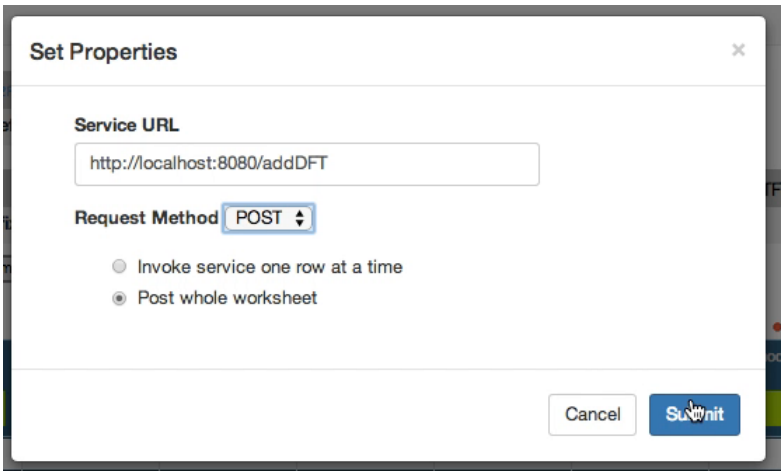


    c. Set the label for the model by changing the 'Name' of the worksheet. Click the top left corner of the current worksheet to change the Name.
    d. Set the service properties from the drop down menu after clicking the worksheet title bar and publish the model

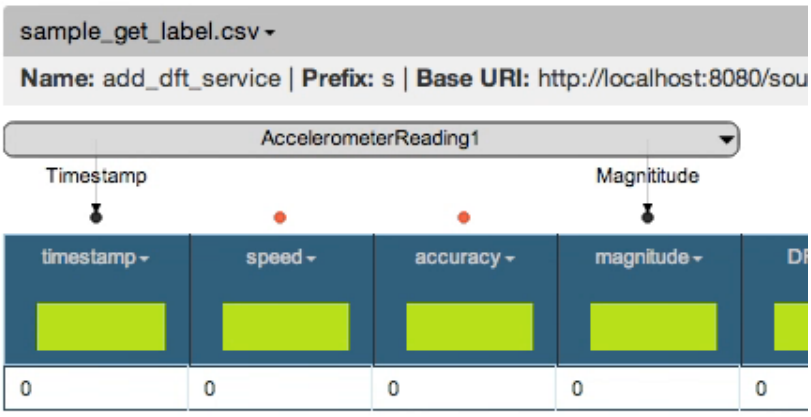




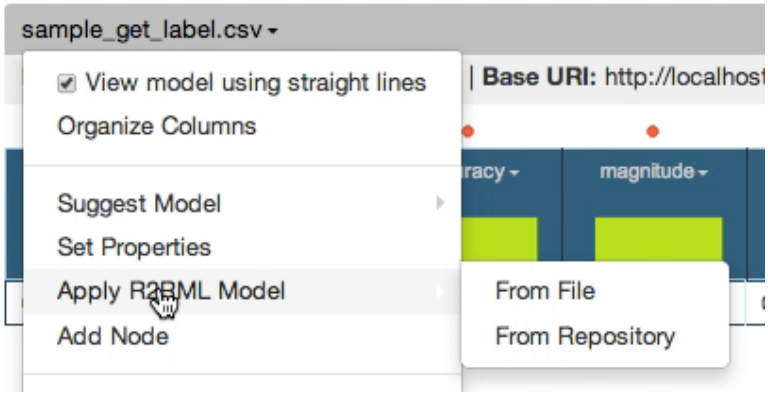
3. Modeling the addDFT service
    a. Load the sample CSV file
    b. Model the required columns for its semantic types
    c. Set the Name of the model
    d. Set the service properties



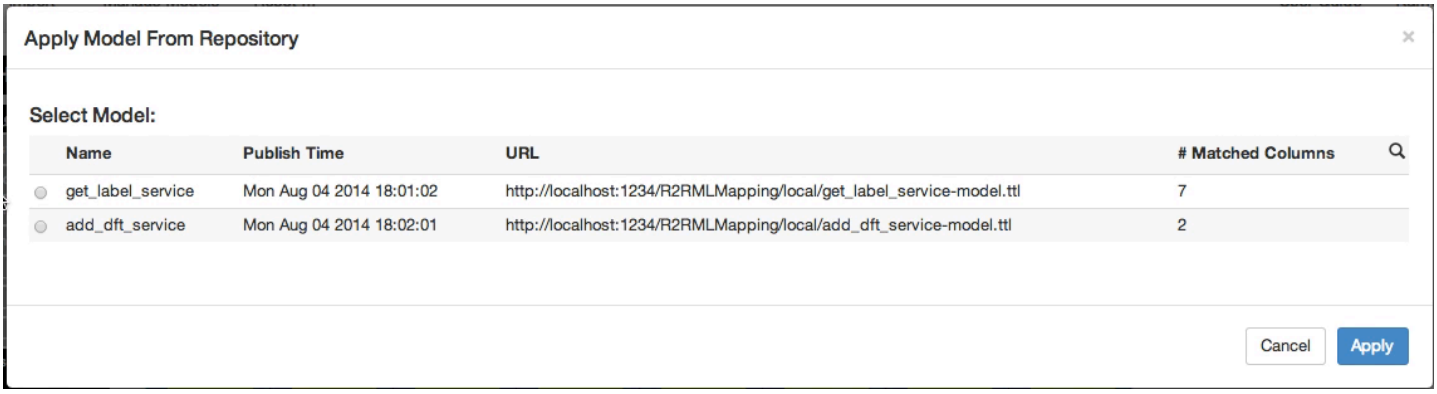
    e. Publish the model



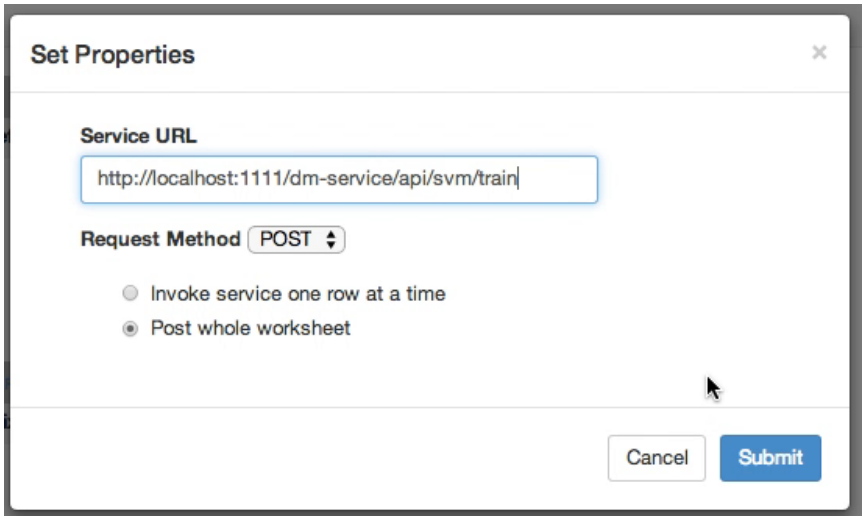
4. Modeling the SVM service
    a. Load the sample CSV file
    b. Apply the model for the getLabel service from the repository



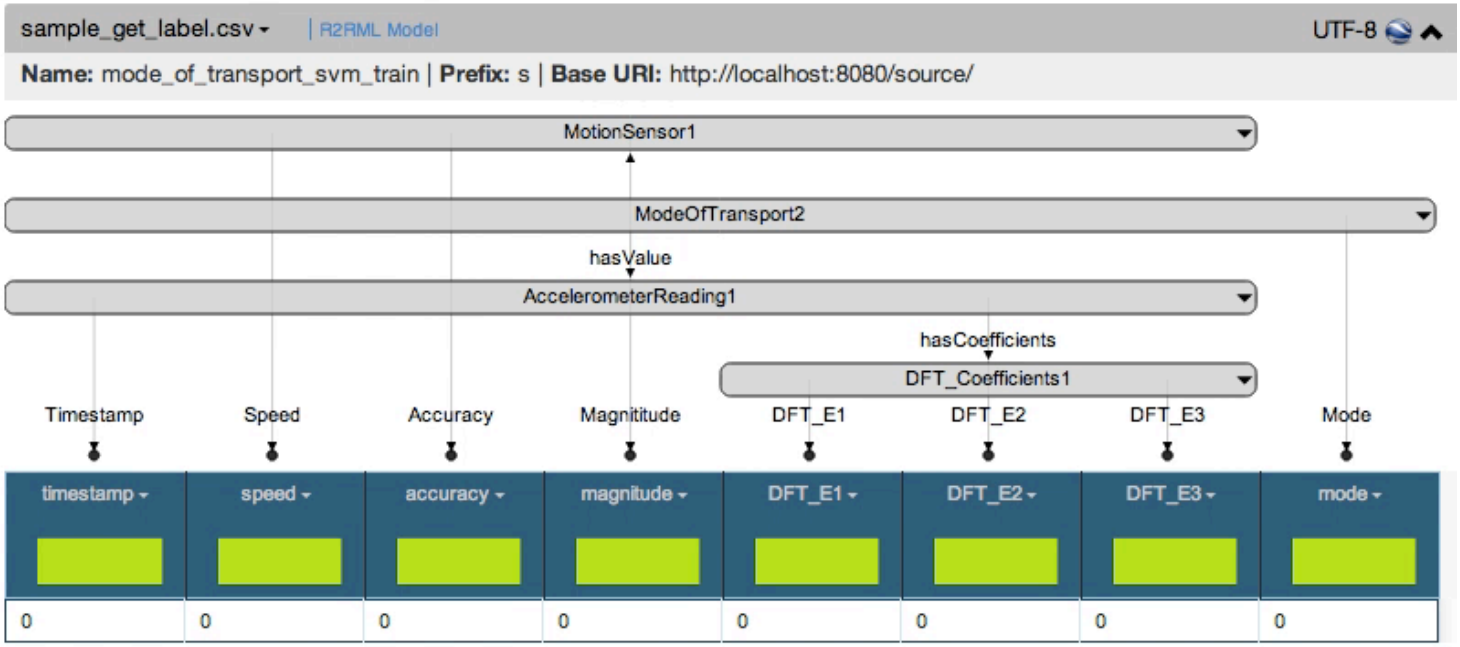
    c. Add the additional semantic type mapping for the “mode” column
    d. Change the Name of the worksheet by clicking the top left of the worksheet



e. Set the service properties
Use a fixed model name when setting the training service The model name can be specified as
http://localhost:8081/dm-service/api/svm/train/{the svm model name}



f. Publish the model

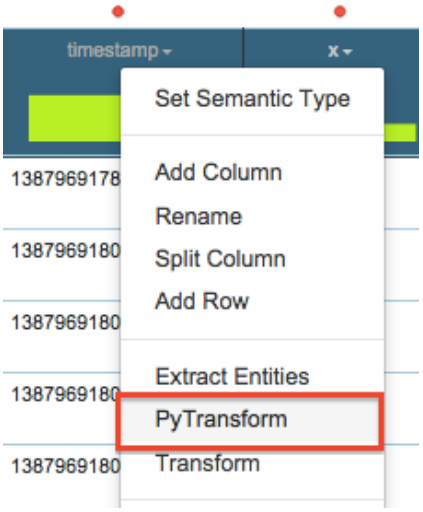


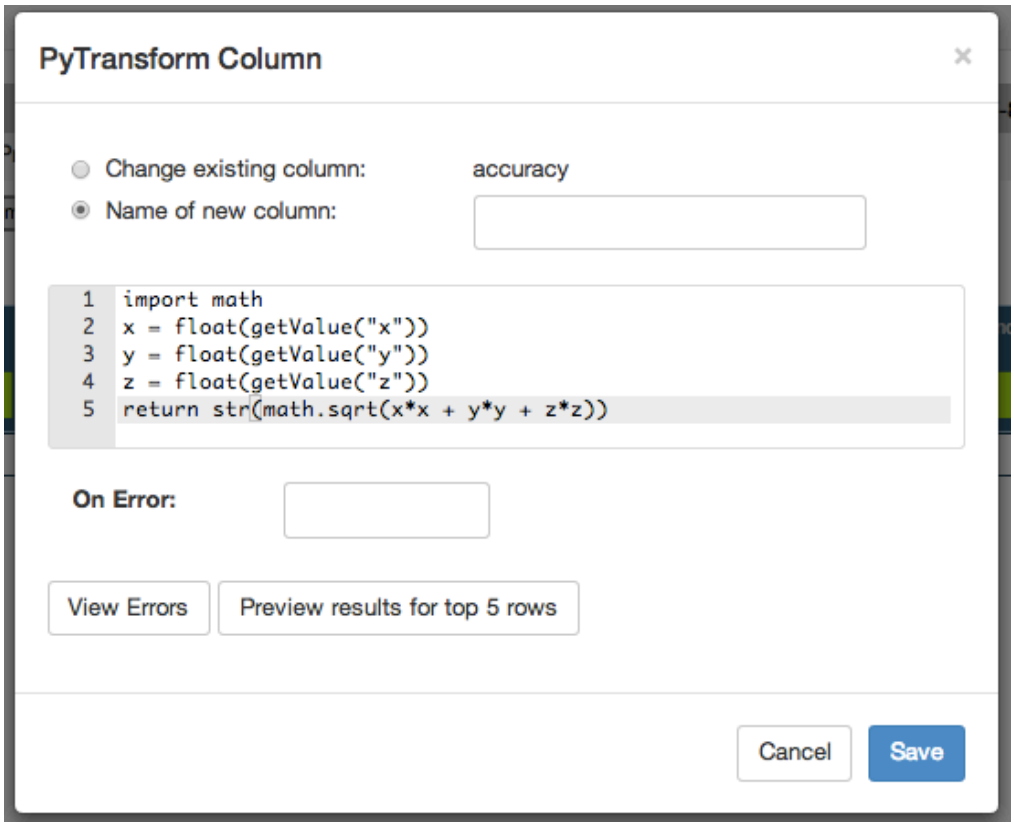
g. Change the name of the model to ‘svm test’
h. Set the service properties to contain the url for the svm testing
Use a previously user model name when setting the testing service url.
The model name can be specified as

http://localhost:8081/dm-service/api/svm/test/{the svm model name}

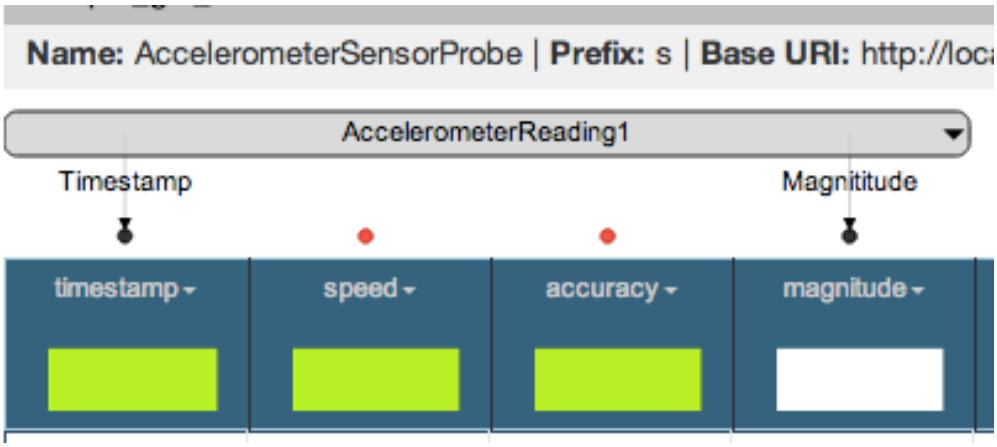
i. Publish the model

5. Modeling the AccelerometerSensor data set.
   a. Load the AccelerometerSensor1.csv file and import only 5 rows into the worksheet. We import a small set of data to speed up the process of modeling.
   b. Add the magnitude column by using a Python transform

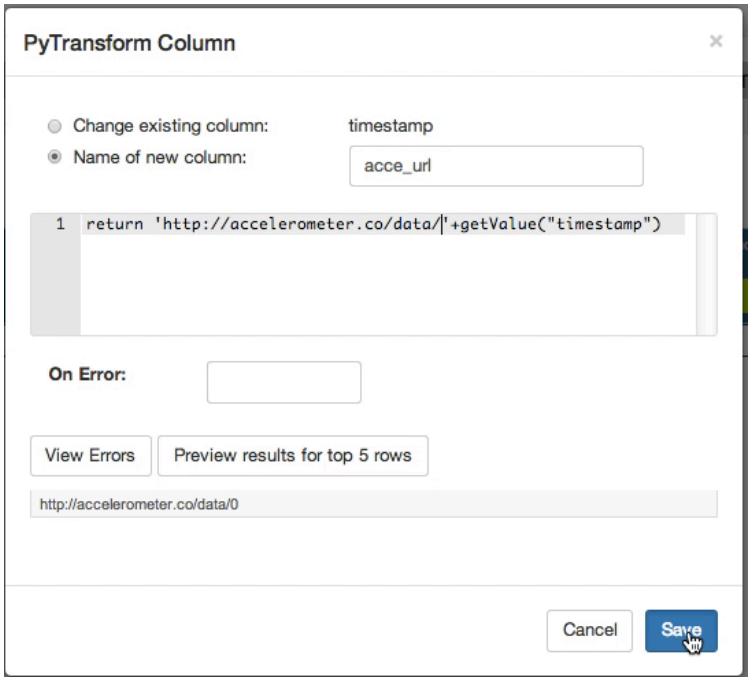




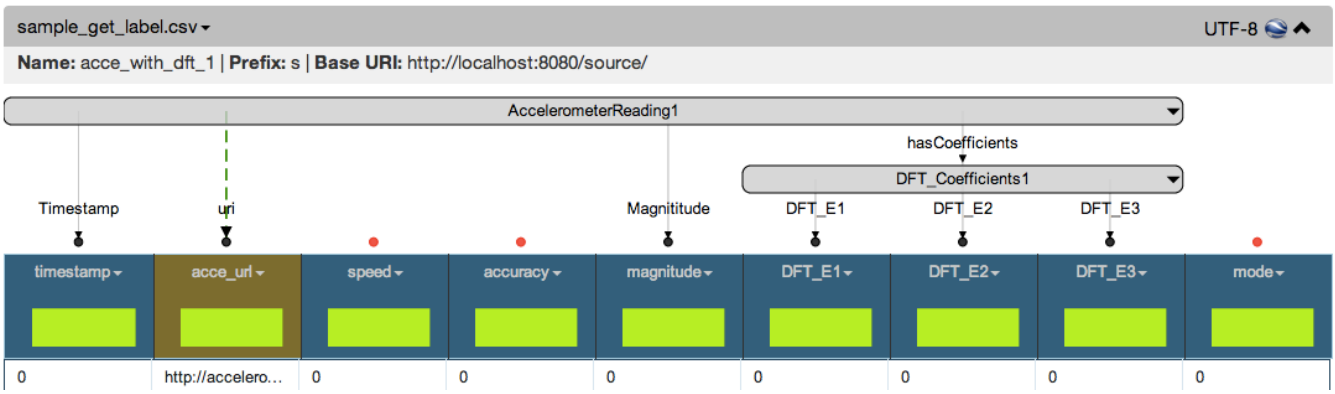
c. Set semantic types for the required columns
d. Publish the model



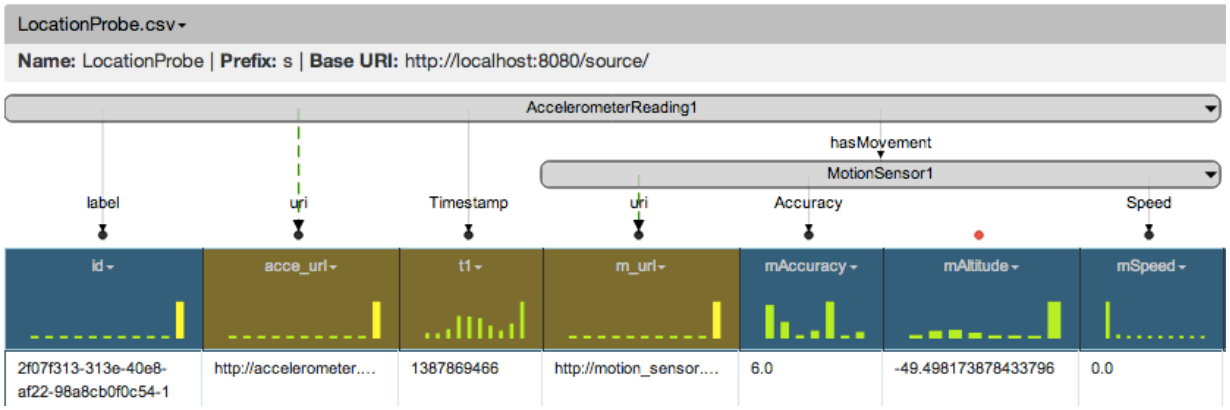
6. Modeling the AccelerometerData with DFT coefficient
   a. Load the sample CSV file
   b. Create Uris using python transform for the AccelerometerReading class



c. Set semantic types for the required columns
d. Publish the model

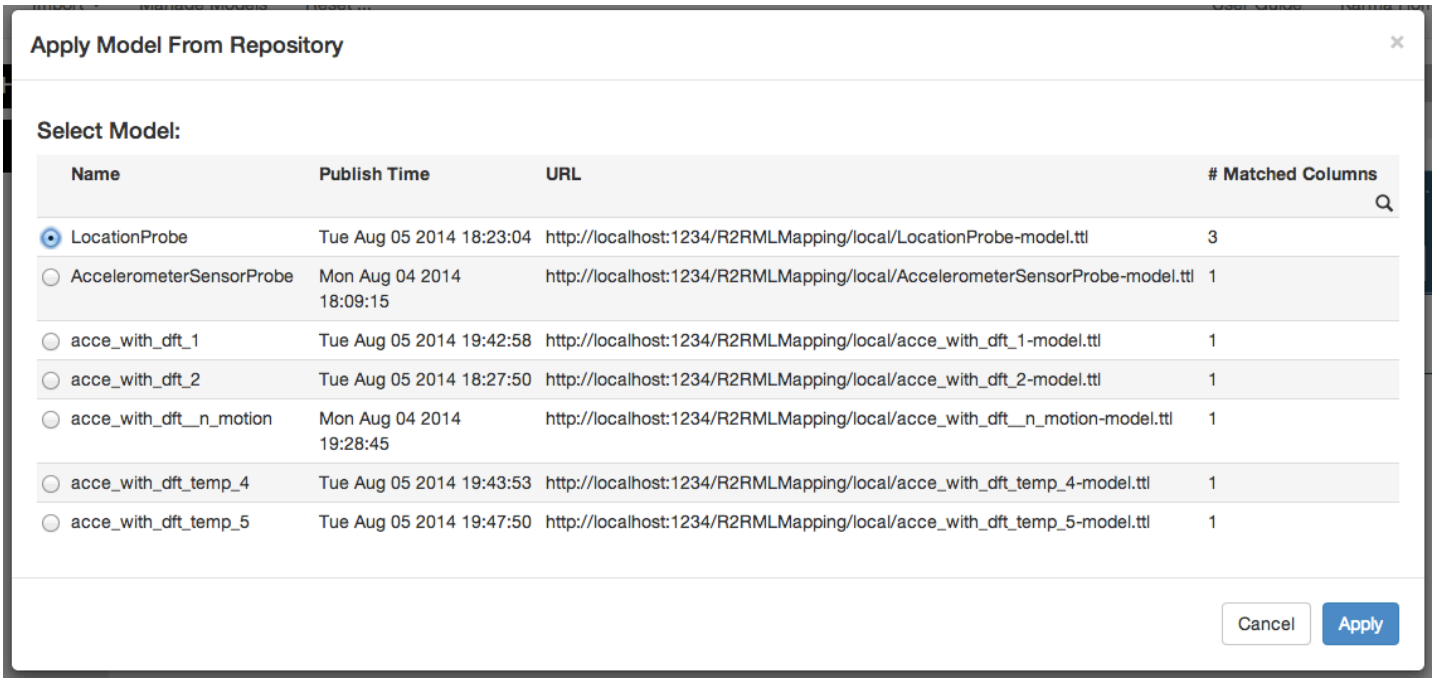


7. Modeling the LocationProbe.csv file
    a. Load the LocationProbe.csv file and import only 1 row.
    b. Generate uris using python transform for MotionSensor and AccelerometerReading classes.
    c. Set the sematic types for the required columns
    d. Publish the model

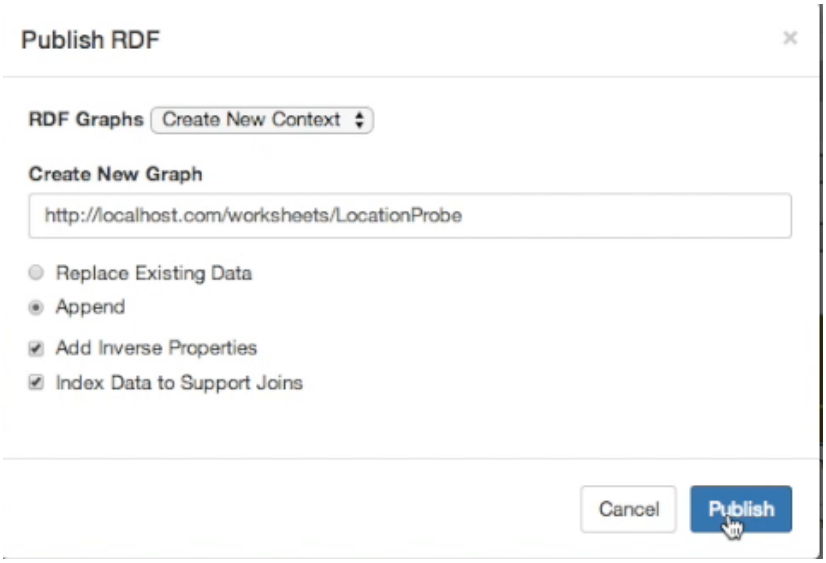


Karma Execution

1. Load the LocationProbe.csv file and import all rows by setting 0 in the rows text field
2. Apply the model "LocationProbe" form the repository

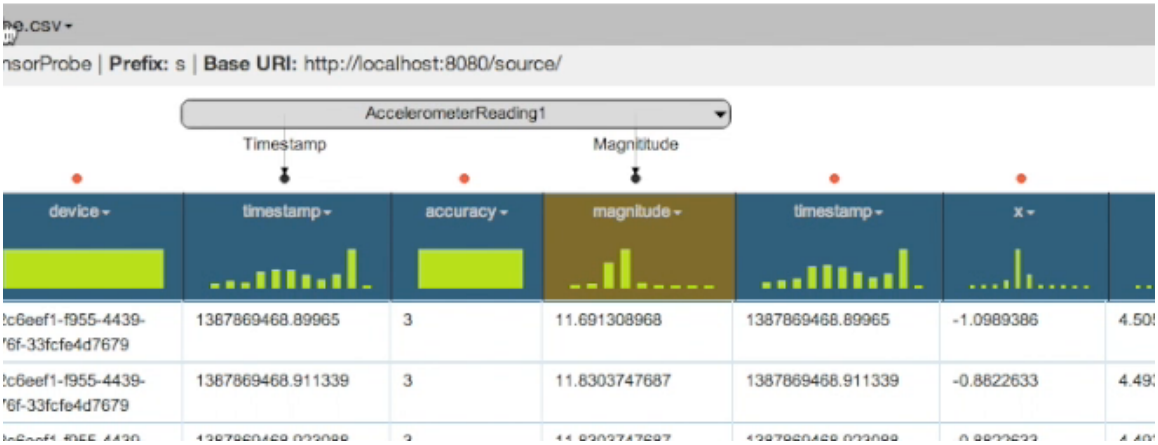


3. Publish the RDF

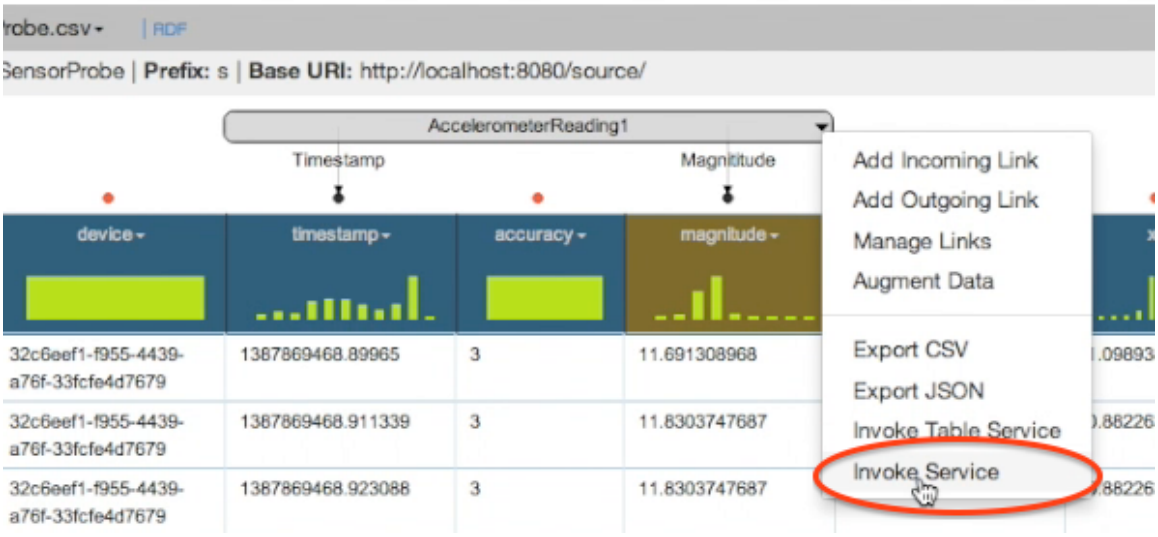


4. Load the AccelerometerSensor1.csv file

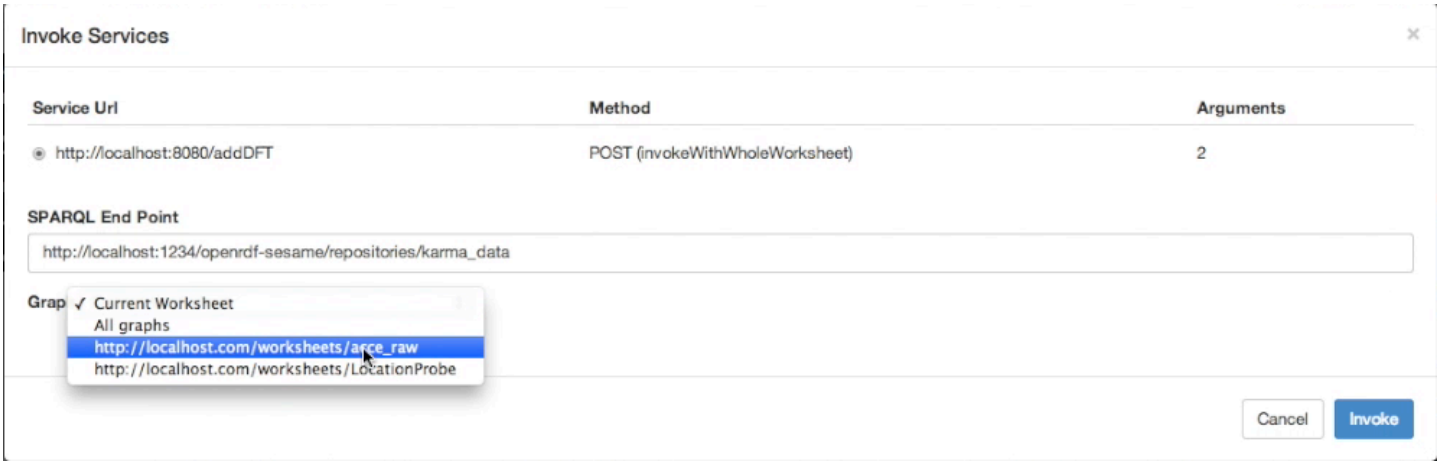
5. Apply the model “AccelerometerSensorModel” from the repository



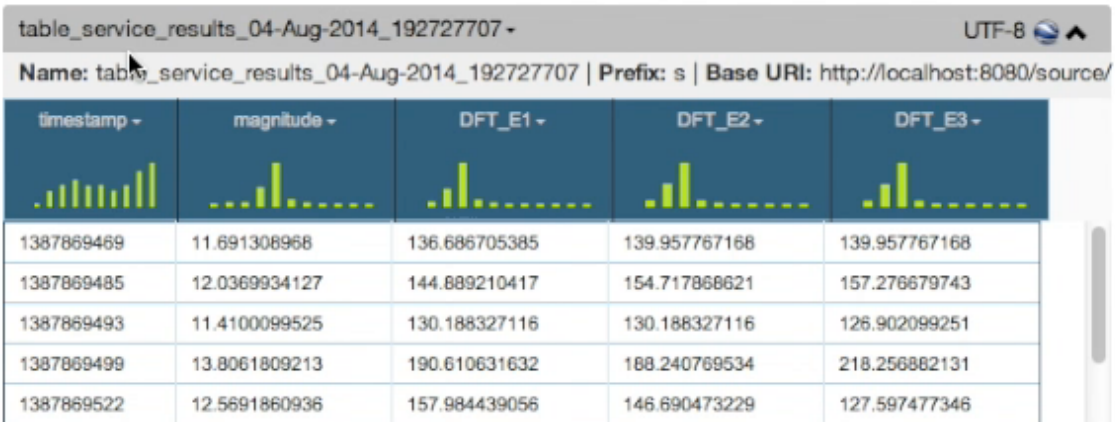
6. Publish the RDF
7. Invoke the addDFT service



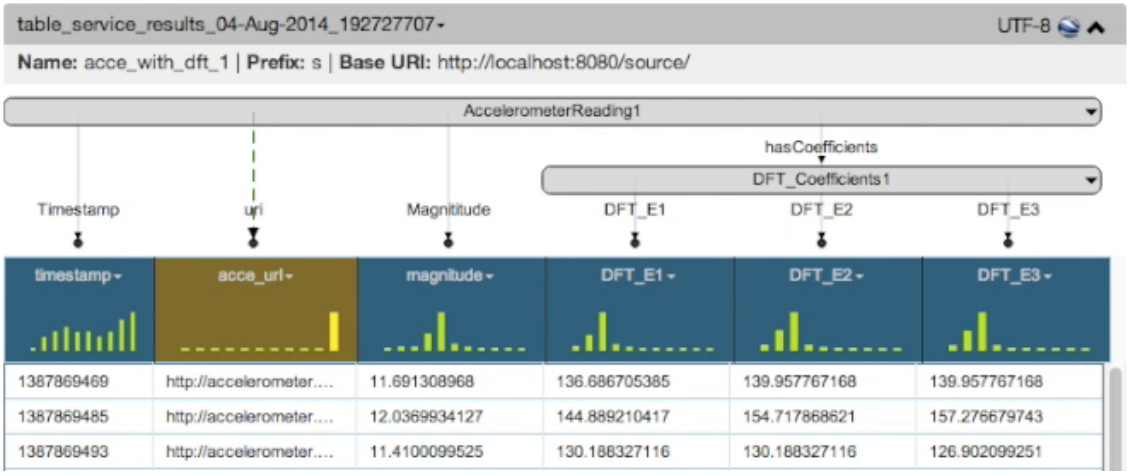
8. Karma lists the services that could be invoked from the given model



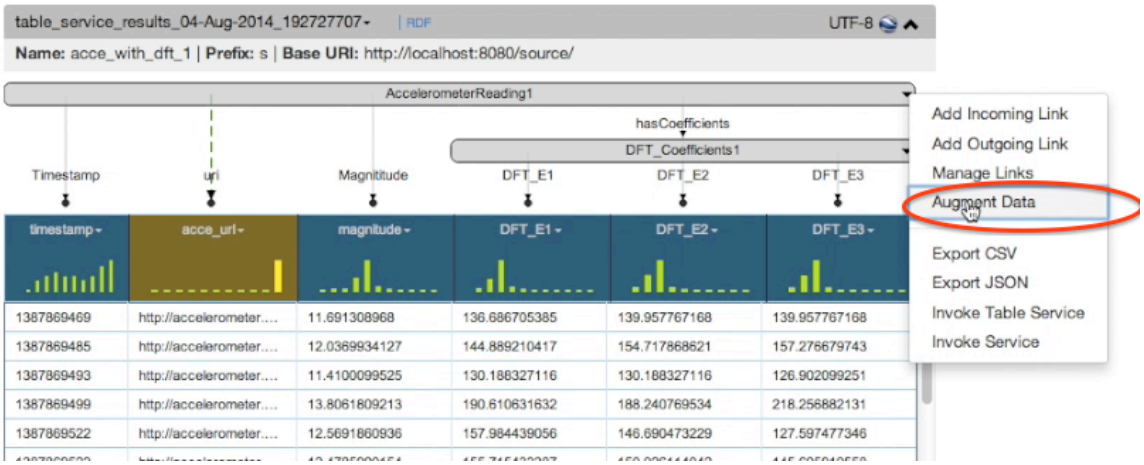
9. The output of the DFT service is loaded into a new worksheet



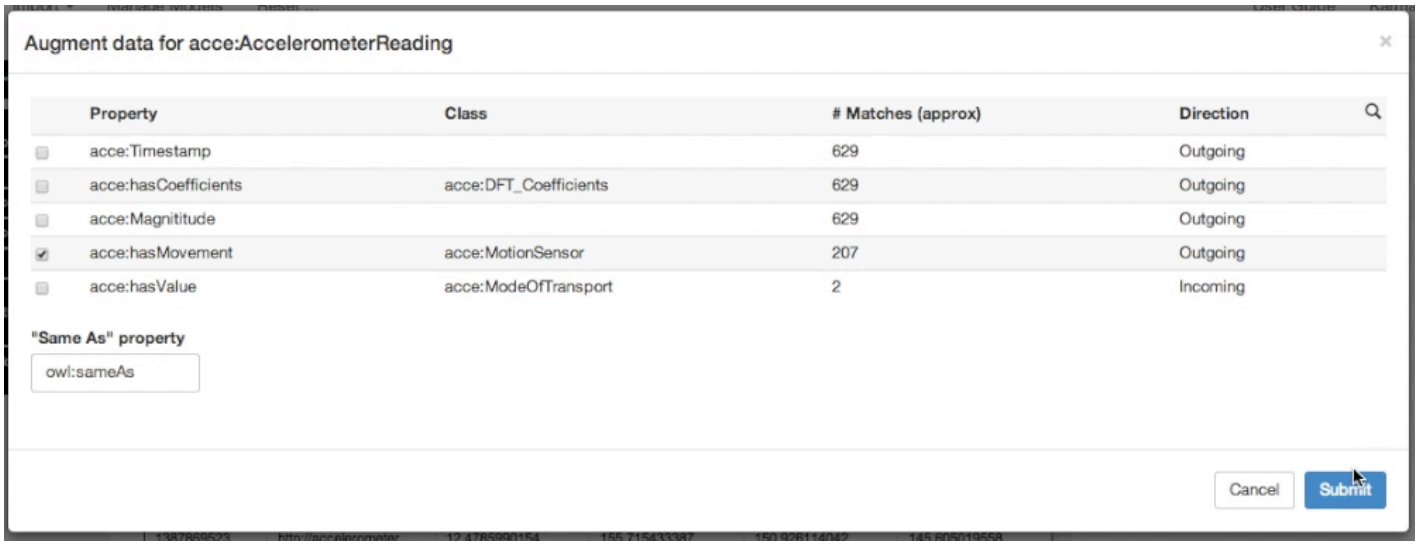
10. Apply the "Acce_with_dft" model on the new worksheet



11. Publish the RDF for this worksheet in a new graph named “acce_with_dft_1”
12. Select “Augment Data” option to perform the join between MotionSensor data and the AccelerometerData

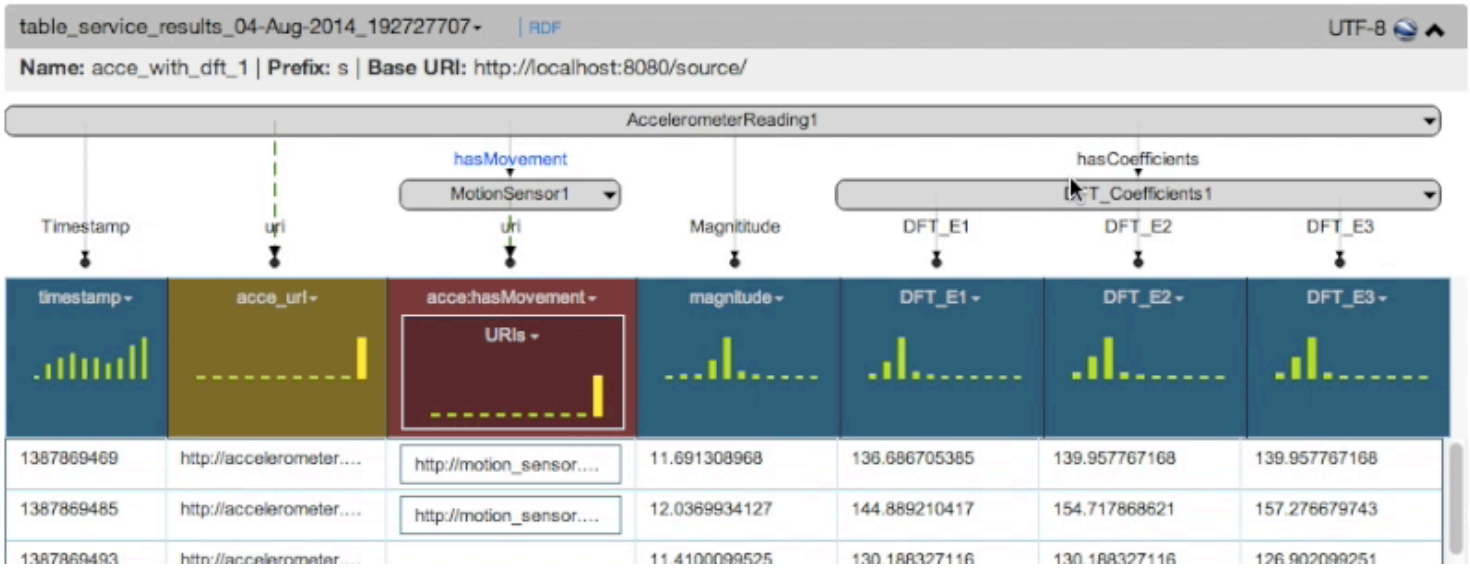


13. Karma will list the properties that were matched

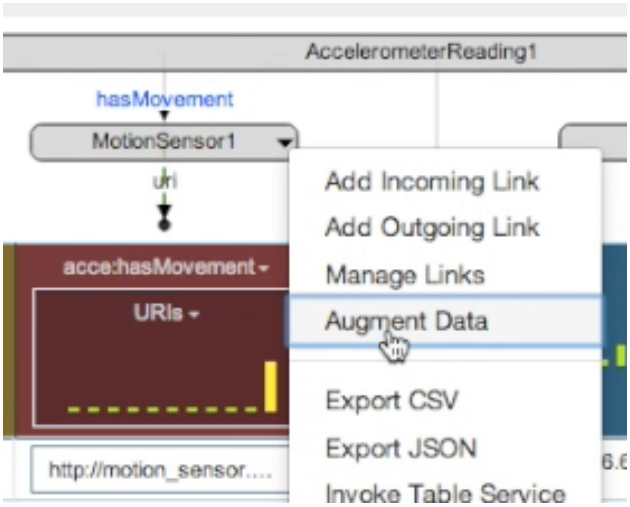


14. Select the “hasMoment” property. The AccelerometerReading class has this object property that maps it to the MotionSensor class, which has data properties like “accuracy” and “speed”

15. Once the data is augmented, a new column for MotionSensor url with the semantic types are added to the worksheet



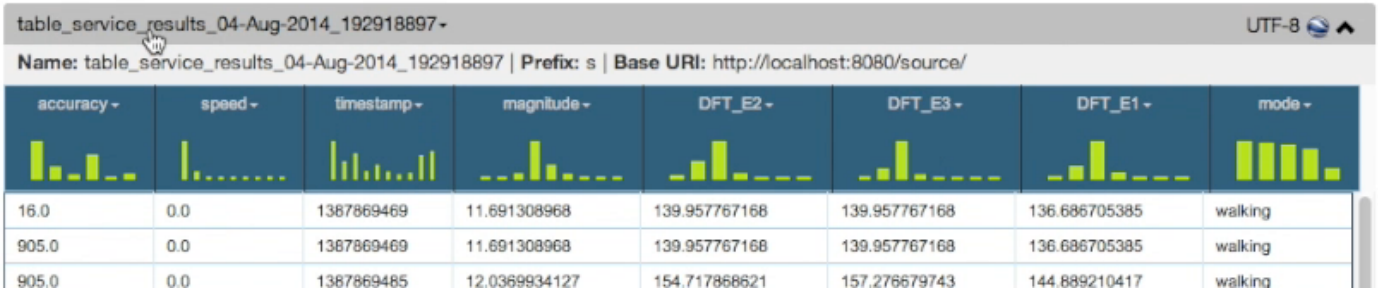
16. Now we again augment the MontionSensor url to get its data properties for "speed" and "accuracy"



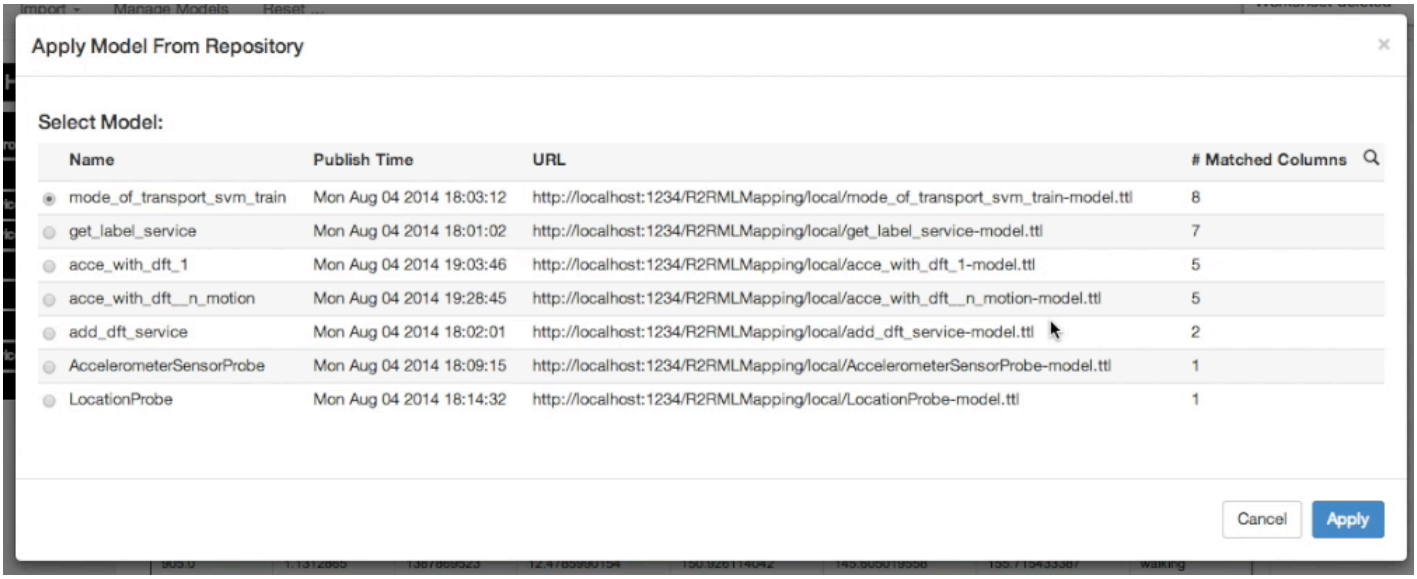
17. After joining we have all our columns that are required for the getLabel service

18. Since this model was created by augmenting data sets, we will publish the model
19. Publish the RDF
20. Now we again invoke a service from the AccelerometerReading class
21. Karma shows two services that could be invoked with the model. We select getLabel

22. The results are loaded into a new worksheet.



23. We now apply the mode of transport model on this worksheet.



24. Publish the RDF

25. If this is the first data set, then we invoke the SVM training service. The output of the training returns us