# [https://avatars2.githubusercontent.com/u/4156894?v=3&s=100](http://www.calstatela.edu/centers/hipic) CIS5560 Term Project Tutorial

#### Authors: Ruchi Singh; Kulwinder Kaur; Ovanes Mikaelian

#### Instructor: [Jongwook Woo](https://www.linkedin.com/in/jongwook-woo-7081a85)

#### Date: 05/18/2017

**Lab Tutorial**

Ruchi Singh ([rsingh26@@calstatela.edu](mailto:rsingh26@@calstatela.edu))

Kulwinder Kaur ([kkaur7@@calstatela.edu](mailto:kkaur7@@calstatela.edu))

Ovanes Mikaelian ([omikael@calstatela.edu](mailto:omikael@calstatela.edu))

**Applications of Machine Learning Algorithms Using AzureML**

**Objectives**

**List what your objectives are.** In this hands-on lab, you will learn how to:

* Construct and evaluate a recommender using Local business and Tip files from Yelp dataset
* Provide users with recommendations for business categories based on their previous ratings for a category and the ratings of other users

**Requirements**

* An Azure ML account
* A web browser and Internet connection

**Platform Specifications**

* Microsoft Azure Machine Learning Studio
* Number of nodes: 1
* Total Memory Size: 10 GB
* Number of modules per experiment: 100

**Creating a Recommender**

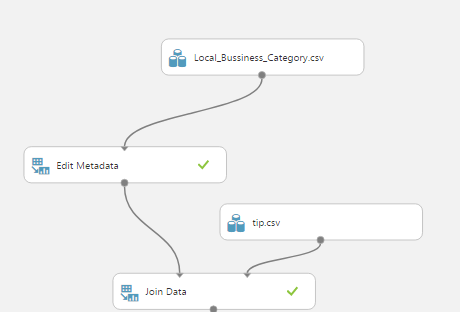
This tutorial shows how to create a Matchbox Recommender using AzureML

## **Data Preparation**

In this step the data is uploaded into AzureML, then it is transformed and cleaned so that it is suitable for the recommender algorithm.

1. Open a browser and browse to [https://studio.azureml.net.](https://studio.azureml.net/) Then sign in using the Microsoft account associated with your Azure ML account.
2. Create a new blank experiment, and give it the title **Yelp** **Category Recommendation**.
3. Download the *business.csv* file and drag it to canvas
4. Search for the **Edit Metadata (Metadata Editor)** module and drag it onto the canvas.
5. Connect the output of the **Business** dataset to the **Dataset** input of the **Edit Metadata (Metadata Editor)**.
6. Configure the properties of the **Edit Metadata (Metadata Editor)** to ensure the **Stars** column is of **Integer** type as required by the **Matchbox Recommender** module:
   * **Lunch Column selector** and select the column: Stars
   * **Data type**: Integer
   * **Categorical**: Unchanged
   * **Fields**: Unchanged
   * **New column names**: blank
7. Download the *tip.csv* dataset and drag it into canvas. This dataset maps user\_ IDs to business\_IDs and provides an info about the likes and the tips about the business.
8. Search for the **Join Data (Join)** module and drag it onto the canvas.
9. Connect the **Results dataset** output of the **Edit Metadata (Metadata Editor)** to the **Dataset1** (left) input of the **Join Data (Join)** module.
10. Connect the output of the **Tip** dataset to the **Dataset2** (right) input of the **Join Data (Join)** module.
11. Configure the properties of the **Join Data (Join)** module as follows:
    * **Column Selector for L** (left): business\_id
    * **Column Selector for R** (right): business\_id
    * **Match case**: Checked
    * **Join type**: Inner Join
    * **Keep right key column**: Unchecked

Our experiment looks like this:



1. Search for the **Select Columns in Dataset (Project Columns)** module and drag it onto the canvas.
2. Connect the **Results dataset** output of the **Join Data (Join)** module to the **Dataset input** of the **Select Columns in Dataset (Project Columns)** module.
3. Launch and Configure the **Column Selector** of the **Select Columns in Dataset (Project Columns)** module. As shown in the below figure, select the **Allow duplicates and preserve column order in selection** box, and then select the following columns in the order shown below
   * **User\_id, categories, stars**
4. Search for the **Clean Missing Data** module and drag it into the canvas
5. Connect the **Results dataset** output of the **Select Columns in Dataset (Project Columns)** module to the input of the **Clean Missing Data** module.
6. Configure the properties of the **Clean Missing Data** module as follows:
   * **Begin with: No Columns**
   * **Column Selector**: User\_Id, categories , stars
   * **Retain first duplicate row**: checked
7. Search for the **Remove Duplicate Rows** module and drag it onto the canvas.
8. Connect the **Cleaned dataset** output of the **Clean Missing Data** module to the input of the **Remove Duplicate Rows** module.
9. Configure the properties of the **Remove Duplicate Rows** module as follows:
   * **Begin with: No Columns**
   * **Column Selector**: User\_Id, categories
   * **Retain first duplicate row**: checked

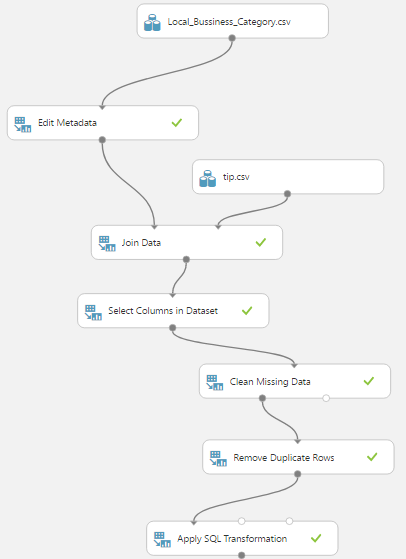
1. Search for the **Apply SQL Transformation** module and drag it onto the canvas.

In the SQL Query Command type the following:  
SELECT user\_id, categories, avg(stars) FROM t1

GROUP BY user\_id, categories;

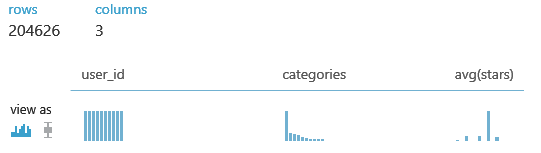
(The SQL command calculates the average star rating each user gives to a particular category.)

Our experiment looks like this:



1. Save and run the experiment. When the experiment has finished, visualize the output of the **Apply SQL Transformation** module.

It should look like this:





1. Search for the **Convert to CSV** module and drag it onto the canvas
2. Connect the output of **Apply SQL Transformation** module to input of **Convert to CSV** module
3. Download the csv. file and save it as reviewstar.csv. This file will be used for SparkML in another project

## **Train a Recommender**

Now that the data is prepared, you can train a recommender.

1. Search for the **Split Data (Split)** module and drag it onto the Canvas.
2. Connect the **Results dataset** output of the **Remove Duplicate Rows** module to the input of the **Split Data (Split) module**.
3. On the properties pane of the **Split Data (Split) module**, configure the properties as follows:
   * **Splitting mode**: Recommender Split
   * **Fraction of training-only users**: 0.75
   * **Fraction of test user ratings for training**: 0.25
   * **Fraction of cold users**: 0
   * **Fraction of cold items**: 0
   * **Fraction of ignored users**: 0
   * **Fraction of ignored items**: 0
   * **Remove occasionally produced cold items**: unchecked
   * **Random seed for Recommender**: 5432
4. Search for the **Train Matchbox Recommender** module and drag it onto the canvas.
5. Connect the **Results dataset1** (left) output of the **Split Data (Split) module** to the **Training dataset of user- item-rating triples** (left) input of the **Train Matchbox Recommender** module.
6. On the properties pane for the **Train Matchbox Recommender** module, configure the properties as follows:
   * **Number of traits**: 20
   * **Number of recommendation algorithm iterations**: 10
   * **Number of training batches**: 4

# Evaluating a Recommender

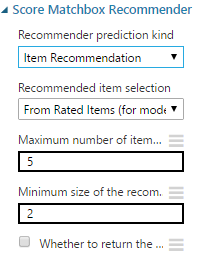
Since there is always a user bias towards a particular business or category, several methods and metrics are used to get a more accurate picture. In this model we will build the following recommenders:

**Item recommendation, Related Items, Rating Prediction, Related Users**

## **Evaluate by Item (Category) Recommendation**

This recommender provides recommendations for a category based on user’s rating. Results are evaluated by NDCG (Normalized Discounted Cumulative Gain). An ideal result has a value of 1.0.

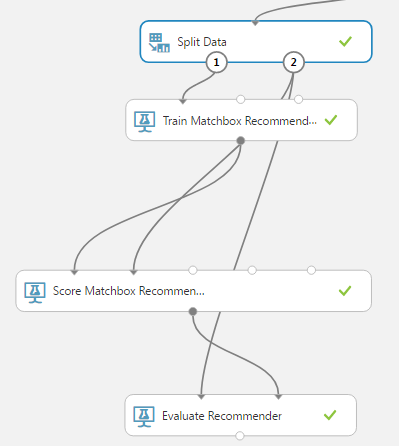
1. Search for the **Score Matchbox Recommender** module and drag it onto the canvas.
2. Connect the **Trained Matchbox recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox recommender** (left) input of the **Score Matchbox Recommender** module.
3. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Dataset to score** (second from left) input to the **Score Matchbox Recommender** module; look at the connect in the figure below.
4. On the properties pane for the **Score Matchbox Recommender** module, ensure that the following properties are specified:
   * **Recommender prediction kind**: Item Recommendation
   * **Recommended item selection**: From Rated Items (for model evaluation)
   * **Maximum number of items to recommend to a user**: 5
   * **Minimum size of the recommendation pool for a single user**: 2



5. Search for the **Evaluate Recommender** module and drag it onto the canvas.

1. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Test dataset** (left) input of the **Evaluate Recommender** module.
2. Connect the **Scored dataset** (right) output of the **Score Matchbox Recommender** module to the **Scored dataset** (right) input of the **Evaluate Recommender** module.
3. On the properties pane of the **Evaluate Recommender** module, verify that the properties are set as follows:
   * **Minimum number of items that the query user and the related user must have rated in common:** 2
   * **Minimum number of users that the query item and the related item must have been rated by in common:** 2

After training and evaluating the recommender our screen should be like this:::



1. Save and run the experiment.
2. When the experiment has finished, Visualize the output form the **Evaluate Recommender** module. Note that the **NDCG** is about 0.98. This is an encouraging result, not too far from the ideal.

## **Evaluate by Related Items (Categories)**

This metric predicts stars for one category based on the stars of other categories. Here we consider related pairs of categories that a group of users has rated. Results are evaluated by the similarity of the ratings using both L1 and L2 NDCG. Ideal recommender will give a value of 0.0 if the ratings are identical in all cases.

1. Copy the **Score Matchbox Recommender** module and the **Evaluate Recommender** module.
2. Paste these modules onto the canvas and drag them to one side.
3. Connect the **Trained Matchbox recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox recommender** (left most) input of the new **Score Matchbox Recommender** module.
4. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Dataset to score** (second from left) input to the new **Score Matchbox Recommender** module.
5. On the properties pane of the new **Score Matchbox Recommender** module configure the following properties:
   * **Recommender prediction kind**: Related Items
   * **Related item selection**: From Rated Items (for model evaluation)
   * **Maximum number of related items to find for an item**: 5
   * **Minimum number of users that the query item and the related item must have been rated by in common**: 2
   * **Minimum size of the related item pool for a single user**: 2

1. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Test dataset** (left hand) input of the new **Evaluate Recommender** module.
2. Ensure the **Scored dataset** (right) output of the new **Score Matchbox Recommender** module is connected to the **Scored dataset** (right hand) input of the new **Evaluate Recommender** module.
3. On the properties pane of the new **Evaluate Recommender** module configure the parameters as follows:
   * **Minimum number of items that the query user and the related user must have rated in common**: 2
   * **Minimum number of users that the query item and the related item must have been rated by in common**: 2

1. Save and run the experiment. 

12. When the experiment has finished, visualize the output form the **Evaluate Recommender** module. Note that the **L1 Sim NDCG** is about 0.94 and the **L2 Sim NDCG** is about 0.95. These values aren’t very good, since the scale is 0 to 5.

## **Evaluate by Rating Predictions**

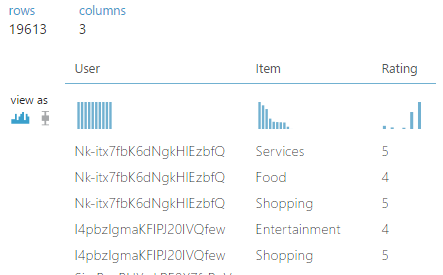
Rating Prediction compare predicted stars to actual star values using mean absolute error (MAE) and root mean square error (RMSE). Ideal results are 0.0 in both cases.

1. Copy the newest **Score Matchbox Recommender** module and the newest **Evaluate Recommender** module.
2. Paste these modules onto the canvas and drag them to one side.
3. Connect the **Trained Matchbox recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox recommender** (left most) input of the newest **Score Matchbox Recommender** module.
4. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Dataset to score** (second from left) input to the newest **Score Matchbox Recommender** module.
5. On the properties pane of the newest **Score Matchbox Recommender** module, set the

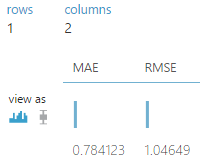
Recommender prediction kind property to Rating Prediction

1. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Test dataset** (left hand) input of the newest **Evaluate Recommender** module.
2. Ensure the **Scored dataset** (right) output of the newest **Score Matchbox Recommender** module is connected to the **Scored dataset** (right hand) input of the newest **Evaluate Recommender** module.
3. On the properties pane of the newest **Evaluate Recommender** module configure the parameters as follows:
   * **Minimum number of items that the query user and the related user must have rated in common**: 2
   * **Minimum number of users that the query item and the related item must have been rated by in common**: 2
4. Save and run the experiment.

Visualize the output of



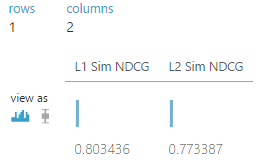
1. When the experiment has finished, visualize the output form the Evaluate Recommender module.



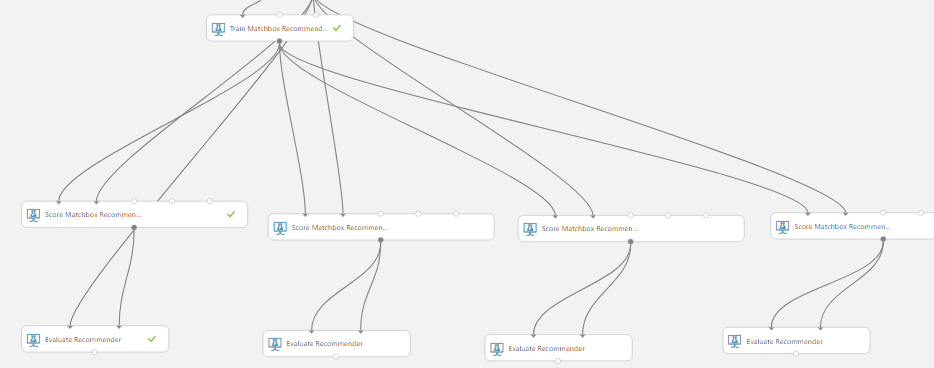
**Evaluate by Related Users**

For pair of users who have rated the same categories, we can predict ratings of one user based on the ratings of the other user. Results are evaluated by the similarity of the ratings using both L1 and L2 average normalized discounted cumulative gain (NDCG) averaged over all the pairs selected. In both cases an ideal value is 0.0.

1. Copy the newest **Score Matchbox Recommender** module and the newest **Evaluate Recommender** module.
2. Paste these modules onto the canvas and drag them to one side.
3. Connect the **Trained Matchbox recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox recommender** (left) input of the newest **Score Matchbox Recommender** module.
4. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Dataset to score** (second from left) input to the newest **Score Matchbox Recommender** module.
5. On the properties pane of configure the following properties:
   * **Recommender prediction kind**: Related Users
   * **Related user selection**: From Users That Rated Items (for model evaluation)
   * **Maximum number of related Users to find for a User**: 5
   * **Minimum number of items that the query user and the related user must have rated in common**: 2
   * **Minimum size of the related user pool for a single user**: 2
6. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Test dataset** (left hand) input of the newest **Evaluate Recommender** module.
7. Ensure the **Scored dataset** (right) output of the newest **Score Matchbox Recommender** module is connected to the **Scored dataset** (right) input of the newest **Evaluate Recommender** module.
8. On the properties pane of the newest **Evaluate Recommender** module configure the parameters as follows:
   * **Minimum number of items that the query user and the related user must have rated in common**: 2
   * **Minimum number of users that the query item and the related item must have been rated by in common**: 2
9. Save and run the experiment.
10. When the experiment has finished, visualize the output form the Evaluate Recommender module. The numbers can be better, but aren’t too bad either.



After all four metrics are added, the bottom part of our screen looks like the following:

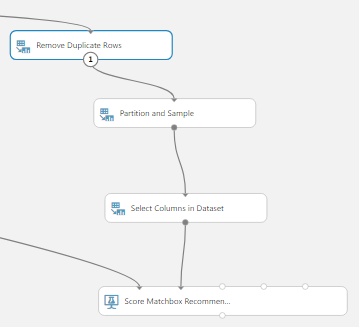


# Computing Category Recommendations

Now that we have constructed the recommender and tested the four metrics, we can compute category recommendations for selected users.

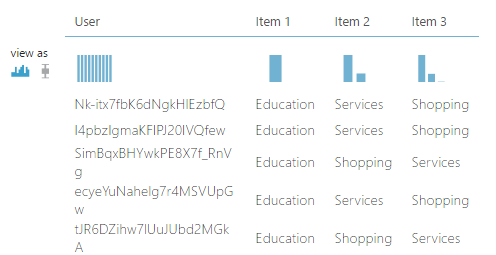
## **Compute Recommendations**

1. Search for the **Remove Duplicate Rows** module, and drag a new one onto the canvas.
2. Connect the output of the existing **Remove Duplicate Rows** module to the input of the new **Remove Duplicate Rows** module.
3. In the properties pane of the new **Remove Duplicate Rows** module set the following:
   * **Column selector**: user\_id
   * **Retain first duplicate row**: checked
4. Search for the **Partition and Sample** module, and drag it onto the canvas.
5. Connect the output of the new **Remove Duplicate Rows** module to the input of the **Partition and Sample** module.
6. On the properties pane of the **Partition and Sample** module set the following parameters:
   * **Partition or sample mode**: Head
   * **Number of rows to select**: 100
7. Search for the **Select Columns in Dataset (Project Columns)** module, and drag it onto the canvas.
8. Launch the **column selector** of the **Select Columns in Dataset (Project Columns)** module and select only the **user\_id** column. Connect **Partition and Sample** to **Select Columns in Dataset (Project Columns)** module.
9. Search for the **Score Matchbox Recommender** module, and drag a new one onto the canvas.
10. Connect the **Trained Matchbox Recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox Recommender** input of the **Score Matchbox Recommender** module.
11. Connect the output of the newest **Select Columns in Dataset (Project Columns)** module to the **Dataset to Score** (second from the left) input of the **Score Matchbox Recommender** module.
12. On the properties pane of the **Score Matchbox Recommender** module set the following parameters:
    * **Recommender prediction kind**: Item Recommendation
    * **Recommended item selection**: From All Items
    * **Maximum number of items to recommend to a user**: 3
13. The new parts of our experiment resemble the lower right part of this diagram:



1. Save and run the experiment.
2. When the experiment has finished running visualize the output of the **Score Matchbox**

**Recommender** module. Examine the output:



There are 100 rows, one for each of the users. Each row contains category recommendation for each user.

**Summary**

In this lab we have constructed and evaluated a Yelp Category recommender. Specifically, we:

* Evaluated the category recommender using four different metrics:
* Determined the performance of the recommender was reasonably good based on these metrics.
* Used the recommender to create category recommendations for 100 users.

**References**

* 1. URLs of Data Source:

<https://s3.amazonaws.com/hipicdatasets/yelp_raw_fall_2016.csv>

<https://www.yelp.com/dataset_challenge/dataset>

* 1. URL of our Github

<https://github.com/rsingh26/DataScience/tree/master/MachineLearning>

* 1. URL of Refereces :

[https://courses.edx.org/asset-v1:Microsoft+DAT203x+1T2016+type@asset+block/DAT203x-](https://courses.edx.org/asset-v1:Microsoft+DAT203x+1T2016+type@asset+block/DAT203x-       Lab_5A.pdf)

[Lab\_5A.pdf](https://courses.edx.org/asset-v1:Microsoft+DAT203x+1T2016+type@asset+block/DAT203x-       Lab_5A.pdf)