[](http://www.calstatela.edu/centers/hipic) CIS5560 Term Project Tutorial

#### Authors: [Monika](https://www.linkedin.com/in/monika-mishra-8b2a4115/) Mishra; [Amogh](https://www.linkedin.com/in/amoghmahesh/) Mahesh; [Aakanksha](https://www.linkedin.com/in/aakanksha-tasgaonkar-272ba393/) Tasgaonkar

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

#### Date: 05/19/2019

**Lab Tutorial**

Monika Mishra [(mmishra2@calstatela.edu)](mailto:(mmishra2@calstatela.edu))

Amogh Mahesh [(amahesh3@calstatela.edu)](mailto:(amahesh3@calstatela.edu))

Aakanksha Tasgaonkar [(atasgao@calstatela.edu)](mailto:(atasgao@calstatela.edu))

05/19/2019

**Applications of Machine Learning Models for Amazon Product Review Data**

**Objectives**

**List what your objectives are.** In this hands-on lab, you will learn how to implement the following machine learning algorithms:

* Match-box Recommender
* Decision Forest Regression
* Boosted Decision Tree Regression

**Platform Specification**

* Microsoft Azure Machine Learning Studio
* # of nodes: 1
* Total Memory Size: 10 GB

**Steps to create an experiment using ML studio:**

1. Data Preparation
2. Train the model
3. Evaluating the model

**Matchbox Recommender**

1. **Data Preparation**
2. 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.
3. Create a new blank experiment and give it the title **Amazon Recommendation - New**.
4. Upload the *Amazon Azure.csv* file and drag it to canvas
5. Search for the **Edit Metadata (Metadata Editor)** module and drag it onto the canvas.
6. Connect the output of the ***Amazon Azure*** dataset to the **Dataset** input of the **Edit Metadata (Metadata Editor)**.
7. Configure the properties of the **Edit Metadata (Metadata Editor)** to ensure the **star\_rating** column is of **Integer** type as required by the **Matchbox Recommender** module:
   * **Launch Column selector** and select the column: star\_rating
   * **Data type**: Integer
   * **Categorical**: Unchanged
   * **Fields**: Unchanged
   * **New column names**: blank
8. Search for the **Select Columns in Dataset (Project Columns)** module and drag it onto the canvas.
9. Connect the **Results dataset** output of the **Edit Metadata** module to the **Dataset input** of the **Select Columns in Dataset (Project Columns)** module.
10. 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

**customer\_id, product\_category, star\_rating**

1. Search for the **Remove Duplicate Rows** module and drag it onto the canvas.
2. Connect the **Select Columns in Dataset (Project Columns)** module’s output of the to the input of the **Remove Duplicate Rows** module.
3. Configure the properties of the **Remove Duplicate Rows** module as follows:
   * **Begin with: No Columns**
   * **Column Selector**: customer\_id, product\_category
   * **Retain first duplicate row**: checked

It should appear like this

A screenshot of a cell phone

Description automatically generated

1. **Train the Model**

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

1. **Evaluating the Model**

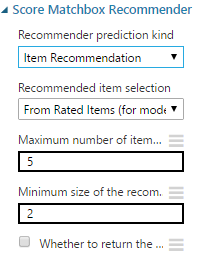
Since there is always a user bias towards a particular business or category, several methods and metrics are used to get a more accurate picture. In this model we will build the following recommenders:

**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.

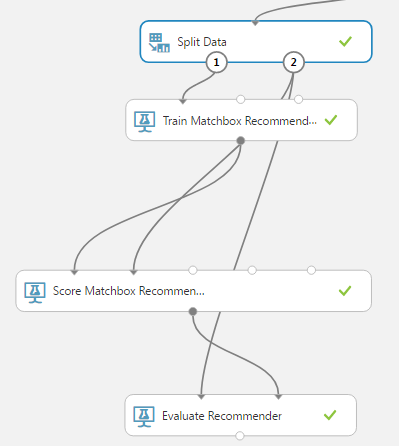
1. Search for the **Score Matchbox Recommender** module and drag it onto the canvas.
2. Connect the **Trained Matchbox recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox recommender** (left) input of the **Score Matchbox Recommender** module.
3. Connect the **Results 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 0.97. 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.

A screenshot of a cell phone

Description automatically generated

12. When the experiment has finished, visualize the output form the **Evaluate Recommender** module. Note that the **L1 Sim NDCG** is about 0.97 and the **L2 Sim NDCG** is about 0.97. 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.

A screenshot of a social media post

Description automatically generated

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

A screenshot of a cell phone

Description automatically generated

**Evaluate by Related Users**

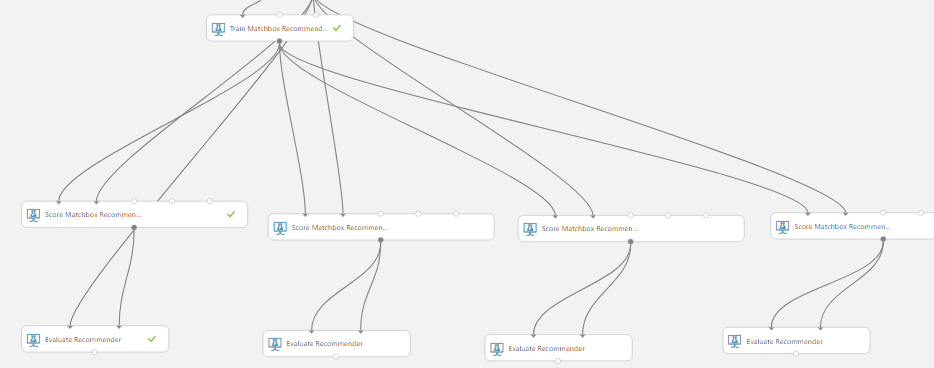
For pair of users who have rated the same categories, we can predict ratings of one user based on the ratings of the other user. Results are evaluated by the similarity of the ratings using both L1 and L2 average normalized discounted cumulative gain (NDCG) averaged over all the pairs selected. In both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

A screenshot of a cell phone

Description automatically generated

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

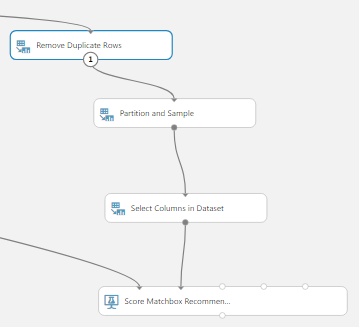


**Computing Category Recommendations**

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**: customer\_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 **customer\_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:

A screenshot of a cell phone

Description automatically generated

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

**Boosted Decision Tree Regression**

1. **Data Preparation**
2. 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.
3. Create a new blank experiment and give it the title **Amazon - BDTR**
4. Upload the *Amazon Azure.csv* file and drag it to canvas
5. Search for the **Select Columns in Dataset (Project Columns)** module and drag it onto the canvas. Exclude the columns 15 and column16.
6. Connect the output of the ***Amazon Azure*** dataset to the **Dataset** input of the **Select Columns in Dataset (Project Columns)**.
7. Search for the **Clean Missing Data** module and drag it onto the canvas. Select the columns review\_headline, review\_body, review date. Set cleaning mode as custom substitution value and replacement value as 0 as shown below:

A screenshot of a cell phone

Description automatically generated

1. Connect the output of the **Project Columns** dataset to the **Dataset** input of the **Clean Missing Data**.
2. Search for **Edit Metadata (Metadata Editor),** drag it to canvas, connect it’s input with the **Results dataset** output of the **Project Columns**.
3. Configure the properties of the **Edit Metadata (Metadata Editor)** as shown in the figure below:

A screenshot of a cell phone

Description automatically generated

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 **Edit Metadata** 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. Select the columns as shown in the figure:

A screenshot of a cell phone

Description automatically generated

It should appear like this

A screenshot of a cell phone

Description automatically generated

1. **Train the Model**

Now that the data is prepared, you can train the model.

1. Search for the **Split Data (Split)** module and drag it onto the Canvas.
2. Connect the **Results dataset** output of the **Select Columns in Dataset (Project Columns)** 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 shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for the **Boosted Decision Tree Regression** module and drag it onto the canvas. Set the property as shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for the **Cross Validate Model** and drag it onto the canvas. Set the property as shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for **Train Model**, drag it to canvas and on the properties pane, select the column **star\_rating**.
2. Search for the **Tune Model Hyperparameters** and drag it onto the canvas. Set the property as shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for the **Permutation Features Importance** modeland drag it onto the canvas. Set the property as shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for Score model and drag it on canvas twice.
2. Connect the **Results dataset1** (left) output of the **Split Data (Split) module** to the right input of the **Train Model** and to the middle input of the **Tune Model Hyperparameters**.
3. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the right input of the two **Score Model** and to the right input of the **Permutation Feature Importance**.
4. Connect the output of **Boosted Decision Tree Regression** model to the left inputs of the **Cross Validate Model**, **Train Model** and **Tune Model Hyperparameters**.
5. Connect the output of the **Select Columns in Dataset** to the right input of the **Cross Validate Model**.
6. Connect the output of the **Train Model** to the left input of the **Score Model.**
7. Connect the right output of the **Tune Model Hyperparameters** to the left input of the **Score Model.**
8. The figure should be like as given below:

A close up of a map

Description automatically generated

1. **Evaluating the Model**
2. Search for the **Evaluate** module and drag it onto the canvas.
3. Connect the left input of the Evaluate model from the output of one Score Model. Connect the right input from the output of the second Score Model.
4. It would appear as below:

A screenshot of a cell phone

Description automatically generated

1. Save and run the experiment.
2. When the experiment has finished, Visualize the output form the **Evaluate** module. You will see that the **Tune Model Hyperparameter** performed better.

A screenshot of a cell phone

Description automatically generated

1. You can also visualize the output from the **Permutation Feature Importance** module to see the features weight and can prune features with less importance. This might improve the model.

A screenshot of a cell phone

Description automatically generated

1. Save the experiment **Amazon - BDTR** as **Amazon – BDTR – New.**
2. On the properties pane of **Select Columns in Dataset**, remove the columns with low importance – review\_date, marketplace, customer\_id, review\_id and review\_body.
3. Save and run the experiment.
4. When the experiment has finished, Visualize the output form the **Evaluate** module to see if any improvement is there. It turned out that no improvement was there.

A screenshot of a cell phone

Description automatically generated

**Decision Forest Regression**

1. **Data Preparation**
2. 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.
3. Create a new blank experiment and give it the title **Amazon - DFR**
4. Upload the *Amazon Azure.csv* file and drag it to canvas
5. Search for the **Select Columns in Dataset (Project Columns)** module and drag it onto the canvas. Exclude the columns 15 and column16.
6. Connect the output of the ***Amazon Azure*** dataset to the **Dataset** input of the **Select Columns in Dataset (Project Columns)**.
7. Search for the **Clean Missing Data** module and drag it onto the canvas. Select the columns review\_headline, review\_body, review date. Set cleaning mode as custom substitution value and replacement value as 0 as shown below:

A screenshot of a cell phone

Description automatically generated

1. Connect the output of the **Project Columns** dataset to the **Dataset** input of the **Clean Missing Data**.
2. Search for **Edit Metadata (Metadata Editor),** drag it to canvas, connect it’s input with the **Results dataset** output of the **Project Columns**.
3. Configure the properties of the **Edit Metadata (Metadata Editor)** as shown in the figure below:

A screenshot of a cell phone

Description automatically generated

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 **Edit Metadata** 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. Select the columns as shown in the figure:

A screenshot of a cell phone

Description automatically generated

It should appear like this

A screenshot of a cell phone

Description automatically generated

1. **Train the Model**

Now that the data is prepared, you can train the model.

1. Search for the **Split Data (Split)** module and drag it onto the Canvas.
2. Connect the **Results dataset** output of the **Select Columns in Dataset (Project Columns)** 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 shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for the **Decision Forest Regression** module and drag it onto the canvas. Set the property as shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for the **Cross Validate Model** and drag it onto the canvas. Set the property as shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for **Train Model**, drag it to canvas and on the properties pane, select the column **star\_rating**.
2. Search for the **Tune Model Hyperparameters** and drag it onto the canvas. Set the property as shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for the **Permutation Features Importance** modeland drag it onto the canvas. Set the property as shown below:

A screenshot of a cell phone

Description automatically generated

1. Search for Score model and drag it on canvas twice.
2. Connect the **Results dataset1** (left) output of the **Split Data (Split) module** to the right input of the **Train Model** and to the middle input of the **Tune Model Hyperparameters**.
3. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the right input of the two **Score Model** and to the right input of the **Permutation Feature Importance**.
4. Connect the output of **Decision Forest Regression** model to the left inputs of the **Cross Validate Model**, **Train Model** and **Tune Model Hyperparameters**.
5. Connect the output of the **Select Columns in Dataset** to the right input of the **Cross Validate Model**.
6. Connect the output of the **Train Model** to the left input of the **Score Model.**
7. Connect the right output of the **Tune Model Hyperparameters** to the left input of the **Score Model.**
8. The figure should be like as given below:

A close up of a map

Description automatically generated

1. **Evaluating the Model**
2. Search for the **Evaluate** module and drag it onto the canvas.
3. Connect the left input of the Evaluate model from the output of one Score Model. Connect the right input from the output of the second Score Model.
4. It would appear as below:

A screenshot of a cell phone

Description automatically generated

1. Save and run the experiment.
2. When the experiment has finished, Visualize the output form the **Evaluate** module. You will see that the **Tune Model Hyperparameter** performed better.

A screenshot of a cell phone

Description automatically generated

1. You can also visualize the output from the **Permutation Feature Importance** module to see the features weight and can prune features with less importance. This might improve the model.

A screenshot of a cell phone

Description automatically generated

1. Save the experiment **Amazon - DFR** as **Amazon – DFR – New.**
2. On the properties pane of Select Columns in Dataset, and select only 5 columns from above which have non zero score and the label column star\_rating.
3. Save and run the experiment.
4. When the experiment has finished, Visualize the output form the **Evaluate** module to see if any improvement is there. It turned the model did improve a bit.

A screenshot of a cell phone

Description automatically generated

**References**

1. URLs of Data Source: https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon\_reviews\_multilingual\_US\_v1\_00.tsv.gz
2. URL of our Github: https://github.com/monika2403/mmishra2/tree/master/CIS%205560
3. URL of References : Microsoft's DAT203x, Data Science and Machine Learning Essentials