**ML Studio (classic) module**

In Machine Learning Studio (classic), a module is a building block for creating experiments. Each module encapsulates a specific machine-learning algorithm, function, or code library that can act on data in your workspace. The modules are designed to accept connections from other modules, to share and modify data. The code that runs in each module comes from many sources. These include open-source libraries and languages, algorithms developed by Microsoft Research, and tools for working with Azure and other cloud services. By connecting and configuring modules, you can create a workflow that reads data from external sources, prepares it for analysis, applies machine learning algorithms, and generates results. When an experiment is open in Machine Learning Studio (classic), you can see the complete list of current modules in the navigation pane at left. You drag these building blocks into your experiment, and then connect them to create a complete machine-learning workflow, called an experiment.

Sometimes modules are updated to add new functionality or to remove older code. When this happens, any experiments that you created that use the module continue to run. But the next time you open the experiment, you are prompted to upgrade the module, or to use a different module.

This group contains most of the machine learning algorithms supported by Machine Learning.

It also contains modules intended to support the algorithms by training models, generating scores, and evaluating model performance.

**Evaluate**: After you have trained a model, use these tools to measure the model’s accuracy.

**Initialize:** These modules provide the machine learning algorithms, which you can customize by setting parameters. The algorithms in this section are grouped by type:

* Anomaly detection algorithms
* Classification algorithms
* Clustering algorithms
* Regression algorithms

**Score**

Use these modules to pass new data through the algorithm, and generate a set of results for evaluation. You can also use the results of scoring as part of a predictive service.

**Train**

These modules train an initialized machine learning model on the data you provide.

**Machine Learning – Evaluate**

Model evaluation is performed after training is complete, to measure the accuracy of the predictions and assess model fit.

Create and use machine learning models in Machine Learning Studio (classic)

The typical workflow for machine learning includes these phases:

1. Choose a suitable algorithm and set initial options.
2. Train the model by using compatible data.
3. Create predictions by using new data that's based on the patterns in the model.
4. Evaluate the model to determine whether the predictions are accurate, the amount of error, and whether overfitting occurs.

Machine Learning Studio (classic) supports a flexible, customizable framework for machine learning. Each task in this process is performed by a specific type of module. The module can be modified, added, or removed without breaking the rest of your experiment.

Use the modules in this category to evaluate an existing model. Model evaluation typically requires some kind of result dataset. If you don't have an evaluation dataset, you can generate results by scoring. You can also use a test dataset, or some other set of data that contains "ground truth" or known expected results

**Model evaluation**

In general, when evaluating a model, your options depend on the type of model you are evaluating, and the metric that you want to use. These topics list some of the most frequently used metrics:

* Evaluate Model
* Cross-Validate Model

Machine Learning Studio (classic) also provides a variety of visualizations, depending on the type of model you're using, and how many classes your model is predicting. Interpreting these statistics often requires a greater understanding of the particular algorithm on which the model was trained.

**View evaluation metrics**

In Machine Learning Studio (classic) to find the metric charts for each model type.

**Two-class classification models**

The default view for binary classification models includes an interactive ROC chart and a table of values for the principal metrics.



You have two options for viewing binary classification models:

* Right-click the module output, and then select **Visualize**.
* Right-click the module, select **Evaluation results**, and then select **Visualize**.

You can also use the slider to change the probability **Threshold** value. The threshold determines whether a result should be accepted as true or not. Then, you can see how these values change.

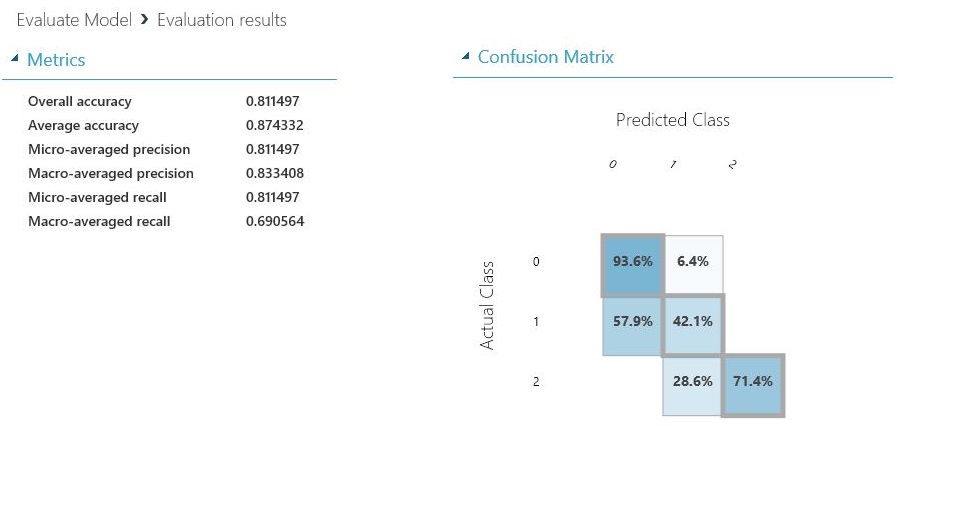
**Multiclass classification models**

The default metrics view for multi-class classification models includes a confusion matrix for all classes and a set of metrics for the model as a whole.

You have two options for viewing multi-class classification models:

* Right-click the module output, and then select **Visualize**.
* Right-click the module, select **Evaluation results**, and then select **Visualize**.

For simplicity, here are the two results, shown side by side:

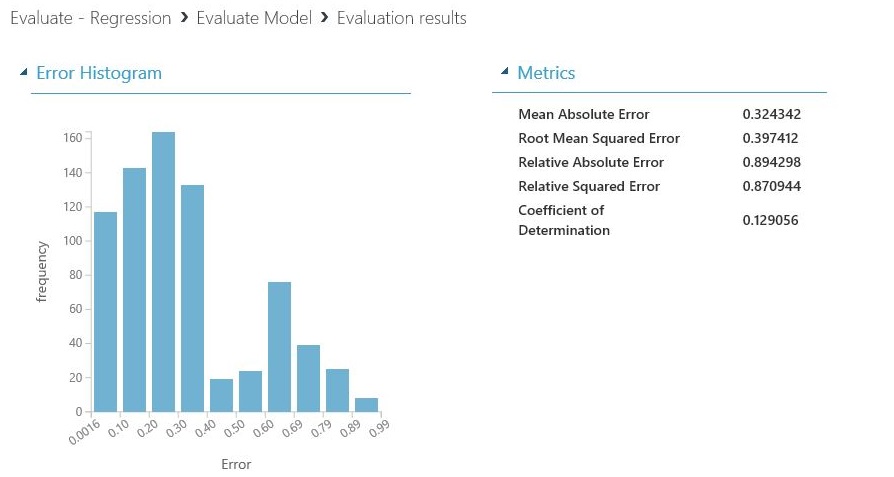


**Regression models**

The metrics view for regression models varies depending on the type of model that you created. The metrics view is based on the underlying algorithm interfaces, and on the best fit for the model metrics.

You have two options for viewing regression models:

* To view the accuracy metrics in a table, right-click the **Evaluate Model** module's output, and then select **Visualize**.
* To view an error histogram with the values, right-click the module, select **Evaluation results**, and then select **Visualize**.



The **Error Histogram** view can help you understand how error is distributed. It's provided for the following model types, and includes a table of default metrics, such as root mean squared error (RMSE).

**Cross-Validate Model**

Cross-validates parameter estimates for classification or regression models by partitioning the data. Cross-validation is an important technique often used in machine learning to assess both the variability of a dataset and the reliability of any model trained using that data.

The Cross-Validate Model module takes as input a labeled dataset, together with an untrained classification or regression model. It divides the dataset into some number of subsets (folds), builds a model on each fold, and then returns a set of accuracy statistics for each fold. By comparing the accuracy statistics for all the folds, you can interpret the quality of the data set and understand whether the model is susceptible to variations in the data. Cross-validate also returns predicted results and probabilities for the dataset, so that you can assess the reliability of the predictions.

**How cross-validation works**

1. Cross-validation randomly divides the training data into a number of partitions, also called folds.

* The algorithm defaults to 10 folds if you have not previously partitioned the dataset.
* To divide the dataset into a different number of folds, you can use the Partition and Sample module and indicate how many folds to use.

1. The module sets aside the data in fold 1 to use for validation (this is sometimes called the holdout fold), and uses the remaining folds to train a model.

For example, if you create five folds, the module would generate five models during cross-validation, each model trained using 4/5 of the data, and tested on the remaining 1/5.

1. During the testing of the model for each fold, multiple accuracy statistics are evaluated. Which statistics are used depends on the type of model that you are evaluating. Different statistics are used to evaluate classification models vs. regression models.
2. When the building and evaluation process is complete for all folds, Cross-Validate Model generates a set of performance metrics and scored results for all the data. You should review these metrics to see whether any single fold has particularly high or low accuracy

Advantages of cross-validation

A different, and very common way of evaluating a model is to divide the data into a training and test set using Split Data, and then validate the model on the training data. However, cross-validation offers some advantages:

* Cross-validation uses more test data.

Cross-validation measures the performance of the model with the specified parameters in a bigger data space. That is, cross-validation uses the entire training dataset for both training and evaluation, instead of some portion. In contrast, if you validate a model by using data generated from a random split, typically you evaluate the model only on 30% or less of the available data.

However, because cross-validation trains and validates the model multiple times over a larger dataset, it is much more computationally intensive and takes much longer than validating on a random split.

* Cross-validation evaluates the dataset as well as the model.

Cross-validation does not simply measure the accuracy of a model but also gives you some idea of how representative the dataset is and how sensitive the model might be to variations in the data.

**How to use the Cross-Validate Model**

There are two main ways to use cross-validation.

* For simple evaluation
* In combination with a parameter sweep

Cross-validation can take a long time to run if you use a lot of data. Therefore, you might use Cross-Validate Model in the initial phase of building and testing your model, to evaluate the goodness of the model parameters (assuming that computation time is tolerable), and then train and evaluate your model using the established parameters with the Train Model and Evaluate Model modules.

**Simple cross-validation**

In this scenario, you both train and test the model using Cross Validate Model.

1. Add the Cross Validate Model module to your experiment. You can find it in Machine Learning Studio (classic), in the Machine Learning category, under Evaluate.
2. Connect the output of any classification or regression model.

For example, if you are using a Two Class Bayes Point Machine for classification, configure the model with the parameters you want, and then drag a connector from the Untrained model port of the classifier to the matching port of Cross Validate Model.

1. On the Dataset port of Cross Validate Model, connect any labeled training dataset.
2. In the Properties pane of Cross Validate Model, click Launch column selector and choose the single column that contains the class label, or the predictable value.
3. Set a value for the Random seed parameter if you want to be able to repeat the results of cross-validation across successive runs on the same data.
4. Run the experiment.

**Cross-validation with a parameter sweep**

In this scenario, you use Tune Model Hyperparameters to identify the best model by conducting a parameter sweep, and then use Cross Validate Model to check its reliability. This is the easiest way to have Machine Learning identify the best model and then generate metrics for it.

1. Add the dataset for model training, and add one of the machine learning modules that creates a classification or regression model.
2. Add the Tune Model Hyperparameters module to your experiment. You can find it in the Machine Learning category, under Train.
3. Attach the classification or regression model to the Untrained model input of Tune Model Hyperparameters.
4. Add the Cross Validate Model module to your experiment. You can find it in Machine Learning Studio (classic), in the Machine Learning category, under Evaluate.
5. Locate the Trained best model output of Tune Model Hyperparameters, and connect it to the Untrained model input of Cross Validate Model.
6. Connect the training data to the Training dataset input of Cross Validate Model.
7. Run the experiment.
8. After reviewing the results, and the evaluation scores, to get a copy of the best model for later re-use, just right-click the Tune Model Hyperparameters module, select Trained best model, and then click Save as Trained Model.

**Results**

After all iterations are complete, Cross-Validate Model creates scores for the entire dataset, as well as performance metrics you can use to assess the quality of the model.

**Scored results**

The first output of the module provides the source data for each row, together with some predicted values and related probabilities.To view these results, in the experiment, right-click the Cross-Validate Model module, select Scored results, and click Visualize.

|  |  |
| --- | --- |
| New column Name | Description |
| Fold Assignments | Indicates the 0-based index of the fold each row of data was assigned to during cross-validation. |
| Scored Labels | This column is added at the end of the dataset, and contains the predicted value for each row |
| Scored Probabilities | This column is added at the end of the dataset, and indicates the estimated probability of the value in Scored Labels. |

**Evaluation results**

The second report is grouped by folds. Remember that, during execution, Cross-Validate Model randomly splits the training data into n folds (by default, 10). In each iteration over the dataset, Cross-Validate Model uses one fold as a validation dataset, and uses the remaining n-1 folds to train a model. Each of the n models is tested against the data in all the other folds. In this report, the folds are listed by index value, in ascending order. To order on any other column you can save the results as a dataset. To view these results, in the experiment, right-click the Cross-Validate Model module, select Evaluation results by fold, and click Visualize.

|  |  |
| --- | --- |
| Column Name | Description |
| Fold Number | An identifier for each fold. If you created 5 folds, there would be 5 subsets of data, numbered 0 to 4. |
| Number of Examples in Fold | The number of rows assigned to each fold. They should be roughly equal. |
| Model | The algorithm used in the model, identified by the API name |

The following metrics are included for each fold, depending on the type of model that you are evaluating.

* Classification models: Precision, recall, F-score, AUC, average log loss, training log loss
* Regression models: Negative log likelihood, mean absolute error, root mean squared error, relative absolute error, and coefficient of determination

**Evaluate Model**

To measure the accuracy of a trained model. You provide a dataset containing scores generated from a model, and the Evaluate Model module computes a set of industry-standard evaluation metrics.

The metrics returned by Evaluate Model depend on the type of model that you are evaluating:

* Classification Models
* Regression Models
* Clustering Models

**How to use Evaluate Model**

There are three ways to use the Evaluate Model module:

* Generate scores over your training data, and evaluate the model based on these scoresGenerate scores on the model, but compare those scores to scores on a reserved testing set
* Compare scores for two different but related models, using the same set of data

**Use the training data**

To evaluate a model, you must connect a dataset that contains a set of input columns and scores. If no other data is available, you can use your original dataset.

1. Connect the Scored datset output of the Score Model to the input of Evaluate Model.
2. Click Evaluate Model module, and select Run selected to generate the evaluation scores.

**Use testing data**

A common scenario in machine learning is to separate your original data set into training and testing datasets, using the Split module, or the Partition and Sample module.

1. Connect the Scored dataset output of the Score Model to the input of Evaluate Model.
2. Connect the output of the Split Data module that contains the testing data to the right-hand input of Evaluate Model.
3. Click Evaluate Model module, and select Run selected to generate the evaluation scores.

**Compare scores from two models**

You can also connect a second set of scores to Evaluate Model. The scores might be a shared evaluation set that has known results, or a set of results from a different model for the same data.

This feature is useful because you can easily compare results from two different models on the same data. Or, you might compare scores from two different runs over the same data with different parameters.

1. Connect the Scored dataset output of the Score Model to the input of Evaluate Model.
2. Connect the output of the Score Model module for the second model to the right-hand input of the Evaluate Model.
3. Right-click Evaluate Model, and select Run selected to generate the evaluation scores.

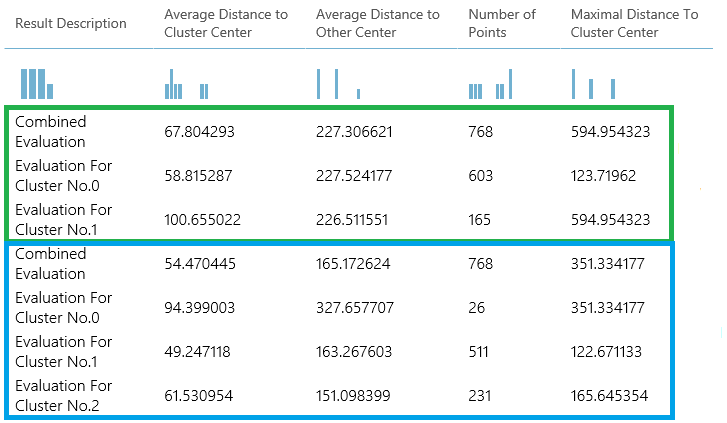
**Results**

After you run the Evaluate Model, right-click the module and select Evaluation results to see the results. You can:

* Save the results as a dataset, for easier analysis with other tools
* Generate a visualization in the Studio (classic) interface

If you connect datasets to both inputs of Evaluate Model, the results will contain metrics for both set of data, or both models. The model or data attached to the left port is presented first in the report, followed by the metrics for the dataset or model attached on the right port.

For example, the following image represents a comparison of results from two clustering models that were built on the same data, but with different parameters.

****

Because this is a clustering model, the evaluation results are different than if you compared scores from two regression models, or compared two classification mod

**Metrics**

This describes the metrics returned for the specific types of models supported for use with Evaluate Model:

* classification models
* regression models
* clustering models

**Metrics for classification models**

The following metrics are reported when evaluating classification models. If you compare models, they are ranked by the metric you select for evaluation.

* ***Accuracy*** measures the goodness of a classification model as the proportion of true results to total cases.
* ***Precision*** is the proportion of true results over all positive results.
* ***Recall*** is the fraction of all correct results returned by the model.
* ***F-score*** is computed as the weighted average of precision and recall between 0 and 1, where the ideal F-score value is 1.
* ***AUC*** measures the area under the curve plotted with true positives on the y axis and false positives on the x axis. This metric is useful because it provides a single number that lets you compare models of different types.
* ***Average log loss*** is a single score used to express the penalty for wrong results. It is calculated as the difference between two probability distributions – the true one, and the one in the model.
* ***Training log loss*** is a single score that represents the advantage of the classifier over a random prediction. The log loss measures the uncertainty of your model by comparing the probabilities it outputs to the known values (ground truth) in the labels. You want to minimize log loss for the model as a whole.

**Metrics for regression models**

The metrics returned for regression models are generally designed to estimate the amount of error. A model is considered to fit the data well if the difference between observed and predicted values is small. However, looking at the pattern of the residuals (the difference between any one predicted point and its corresponding actual value) can tell you a lot about potential bias in the model.

The following metrics are reported for evaluating regression models. When you compare models, they are ranked by the metric you select for evaluation.

* ***Negative log likelihood*** measures the loss function, a lower score is better. Note that this metric is only calculated for Bayesian Linear Regression and Decision Forest Regression; for other algorithms, the value is Infinity which means for nothing.
* ***Mean absolute error (MAE)*** measures how close the predictions are to the actual outcomes; thus, a lower score is better.
* ***Root mean squared error (RMSE***) creates a single value that summarizes the error in the model. By squaring the difference, the metric disregards the difference between over-prediction and under-prediction.
* ***Relative absolute error (RAE)*** is the relative absolute difference between expected and actual values; relative because the mean difference is divided by the arithmetic mean.
* ***Relative squared error (RSE)*** similarly normalizes the total squared error of the predicted values by dividing by the total squared error of the actual values.
* ***Mean Zero One Error (MZOE)*** indicates whether the prediction was correct or not. In other words: ZeroOneLoss(x,y) = 1 when x!=y; otherwise 0.
* ***Coefficient of determination***, often referred to as R2, represents the predictive power of the model as a value between 0 and 1. Zero means the model is random (explains nothing); 1 means there is a perfect fit. However, caution should be used in interpreting R2 values, as low values can be entirely normal and high values can be suspect.

**Metrics for clustering models**

Because clustering models differ significantly from classification and regression models in many respects, Evaluate Model also returns a different set of statistics for clustering models.

The statistics returned for a clustering model describe how many data points were assigned to each cluster, the amount of separation between clusters, and how tightly the data points are bunched within each cluster.

The statistics for the clustering model are averaged over the entire dataset, with additional rows containing the statistics per cluster.

For example, the following results show a portion of the results from a sample experiment that clusters the data in the PIMA Indian Diabetes Binary Classification dataset, which is available in Machine Learning Studio (classic).els. However, the overall presentation is the same.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result Description** | **Average Distance to Cluster Center** | **Average Distance to Other Center** | **Number of Points** | **Maximal Distance To Cluster Center** |
| Combined  Evaluation | 55.915068 | 169.897505 | 538 | 303.545166 |
| Evaluation For Cluster No.0 | 0 | 1 | 570 | 0 |
| Evaluation For Cluster No.1 | 0 | 1 | 178 | 0 |
| Evaluation For Cluster No.2 | 0 | 1 | 178 | 0 |

From these results, you get the following information:

* The ***Sweep Clustering*** module creates multiple clustering models, listed in order of accuracy. For simplicity, we've shown only the best-ranked model here. Models are measured using all possible metrics, but the models are ranked by using the metric that you specified. If you changed the metric, a different model might be ranked higher.
* The ***Combined Evaluation*** score at the top of the each section of results lists the averaged scores for the clusters created in that particular model.

This top-ranked model happened to create three clusters; other models might create two clusters, or four clusters. Therefore, this combined evaluation score helps you compare models with different number of clusters.

* The scores in the column, ***Average Distance to Cluster Center***, represent the closeness of all points in a cluster to the centroid of that cluster.
* The scores in the column, ***Average Distance to Other Center***, represent how close, on average, each point in the cluster is to the centroids of all other clusters.

You can choose any one of four metrics to measure this distance, but all measurements must use the same metric.

* The ***Number of Points*** column shows how many data points were assigned to each cluster, along with the total overall number of data points in any cluster.

If the number of data points assigned to clusters is less than the total number of data points available, it means that the data points could not be assigned to a cluster.

* The scores in the column, ***Maximal Distance to Cluster Center***, represent the sum of the distances between each point and the centroid of that point’s cluster.

If this number is high, it can mean that the cluster is widely dispersed. You should review this statistic together with the Average Distance to Cluster Center to determine the cluster’s spread.

Evaluate Recommender

Evaluates the accuracy of recommender model predictions. To measure the accuracy of predictions made by a recommendation model. Using this module, you can evaluate four different kinds of recommendations:

* Ratings predicted for a given user and item
* Items recommended for a given user
* A list of users found to be related to a given user
* A list of items found to be related to a given item

When you create predictions using a recommendation model, slightly different results are returned for each of these supported prediction types. The Evaluate Recommender module deduces the kind of prediction from the column format of the scored dataset. For example, the scored dataset might contain:

* user-item-rating triples
* users and their recommended items
* users and their related users
* items and their related items

The module also applies the appropriate performance metrics, based on the type of prediction being made.

**How to configure Evaluate Recommender**

The Evaluate Recommender module compares the predictions output by a recommendation model with the corresponding "ground truth" data. For example, the Score Matchbox Recommender module produces scored datasets that can be analyzed with Evaluate Recommender.

**Requirements**

Evaluate Recommender requires the following datasets as input.

**Test dataset**

The test dataset contains the "ground truth" data in the form of user-item-rating triples. If you already have a dataset containing user-item-rating triples, you can apply the Split Data module, using the RecommenderSplit option, to create a training dataset and a related test set from the existing dataset.

**Scored dataset**

The scored dataset contains the predictions that were generated by the recommendation model.

The columns in this second dataset depend on the kind of prediction you were performing during scoring. For example, the scored dataset might contain any of the following:

* Users, items, and the ratings the user would likely give for the item
* A list of users and items recommended for them
* A list of users, with users who are probably similar to them
* A list of items, together with smiliar items

**Metrics**

Performance metrics for the model are generated based on the type of input.

**Evaluate predicted ratings**

When evaluating predicted ratings, the scored dataset (the second input to Evaluate Recommender) must contain user-item-rating triples, meeting these requirements:

* The first column of the dataset contains user identifiers.
* The second column contains the item identifiers.
* The third column contains the corresponding user-item ratings.

**Evaluate Recommender** compares the ratings in the ground truth dataset to the predicted ratings of the scored dataset, and computes the mean absolute error (MAE) and the root mean squared error (RMSE).The other parameters of Evaluate Recommender have no effect on evaluation of rating predictions.

**Evaluate item recommendations**

When evaluating item recommendation, use a scored dataset that includes the recommended items for each user:

* The first column of the dataset must contain the user identifier.
* All subsequent columns should contain the corresponding recommended item identifiers, ordered by how relevant an item is to the user.

Before connecting this dataset, we recommend that you sort the dataset so that the most relevant items come first.

The other parameters of Evaluate Recommender have no effect on evaluation of item recommendations.

***Evaluate Recommender*** computes the average normalized discounted cumulative gain (NDCG) and returns it in the output dataset.

Because it is impossible to know the actual "ground truth" for the recommended items, Evaluate Recommender uses the user-item ratings in the test dataset as gains in the computation of the NDCG. To evaluate, the recommender scoring module must only produce recommendations for items with ground truth ratings (in the test dataset).

**Evaluate predictions of related users**

When evaluating predictions of related users, use a scored dataset that contains the related users for each user of interest:

* The first column must contain the identifiers for each user of interest.
* All subsequent columns contain the identifiers for the predicted related users. Related users are ordered by the strength of the relationship (most related user first).
* For Evaluate Recommender to work, the column names must be User, Related User 1, Related User 2, Related User 3, and so forth.

Evaluate Recommender computes the average normalized discounted cumulative gain (NDCG), based on Manhattan (L1 Sim NDCG) and Euclidean (L2 Sim NDCG) distances, and returns both values in the output dataset. Because there is no actual ground truth for the related users, Evaluate Recommender uses the following procedure to compute the average NDCGs.

For each user of interest in the scored dataset:

1. Find all items in the test dataset which have been rated by both the user of interest and the related user under consideration.
2. Create two vectors from the ratings of these items: one for the user of interest, and one for the related user under consideration.
3. Compute the gain as the similarity of the resulting two rating vectors, in terms of their Manhattan (L1) or Euclidean (L2) distance.
4. Compute the L1 Sim NDCG and the L2 Sim NDCG, using the gains of all related users.
5. Average the NDCG values over all users in the scored dataset.

In other words, gain is computed as the similarity (normalized Manhattan or Euclidian distances) between a user of interest (the entry in the first column of scored dataset) and a given related user (the entry in the n-th column of the scored dataset). The gain of this user pair is computed using all items for which both items have been rated in the original data (test set). The NDCG is then computed by aggregating the individual gains for a single user of interest and all related users, using logarithmic discounting. That is, one NDCG value is computed for each user of interest (each row in the scored dataset). The number that is finally reported is the arithmetic average over all users of interest in the scored dataset (i.e. its rows).

Hence, to evaluate, the recommender scoring module must only predict related users who have items with ground truth ratings (in the test dataset).