**Machine Learning – Score**

Scoring is also called prediction, and is the process of generating values based on a trained machine learning model, given some new input data. The values or scores that are created can represent predictions of future values, but they might also represent a likely category or outcome. The meaning of the score depends on the type of data you provide, and the type of model that you created.

**Create and use models in Machine Learning Studio (classic)**

* The typical workflow for machine learning includes these phases:
* Choosing a suitable algorithm, and setting initial options.
* Training the model on compatible data.
* Creating predictions using new data, based on the patterns in the model.
* Evaluating the model to determine if the predictions are accurate, how much error there is, and if there is any overfitting.

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, which can be modified, added, or removed, without breaking the rest of your experiment. The modules in this section include tools for scoring. In this phase of machine learning, you apply a trained model to new data, to generate predictions. You can either send those predictions to an application that consumes machine learning results, or use the results of scoring to evaluate the accuracy and usefulness of the model.

Scoring is widely used in machine learning to mean the process of generating new values, given a model and some new input. The generic term "score" is used, rather than "prediction," because the scoring process can generate so many different types of values:

* A list of recommended items and a similarity score.
* Numeric values, for time series models and regression models.
* A probability value, indicating the likelihood that a new input belongs to some existing category.
* The name of a category or cluster to which a new item is most similar.
* A predicted class or outcome, for classification models.

When you add one of these modules in your experiment, you must attach an already trained machine learning model, and some new data. When you run the experiment or the selected module, the scoring module ingests the new data, computes scores based on the model, and returns the scores in a table.

**Data used for scoring**

The new data that you provide as input generally needs to have the same columns that were used to train the model, minus the label, or outcome column.

Columns that are used solely as identifiers are usually excluded when training a model, and thus should be excluded when scoring as well. However, identifiers such as primary keys can easily be re-combined with the scoring dataset later, by using the Add Columns module. This module works without you having to specify a join key, as long as the dataset size has not changed.

Before you perform scoring on your dataset, always check for missing values and nulls. When data used as input for scoring has missing values, the missing values are used as inputs. Because nulls are propagated, the result is usually a missing value.

**List of scoring modules**

Machine Learning Studio (classic) provides many different scoring modules. You select one depending on the type of model you are using, or the type of scoring task you are performing:

* Apply Transformation: Applies a well-specified data transformation to a dataset. Use this module to apply a saved process to a set of data.
* Assign Data to Clusters: Assigns data to clusters by using an existing trained clustering model. Use this module if you want to cluster new data based on an existing K-Means clustering model. This module replaces the Assign to Clusters (deprecated) module, which has been deprecated but is still available for use in existing experiments.
* Score Matchbox Recommender: Scores predictions for a dataset by using the Matchbox recommender. Use this module if you want to generate recommendations, find related items or users, or predict ratings.
* Score Model: Scores predictions for a trained classification or regression model. Use this module for all other regression and classification models, as well as some anomaly detection models.

**Apply Transformation**

To modify an input dataset based on a previously computed transformation. For example, if you used z-scores to normalize your training data by using the Normalize Data module, you would want to use the z-score value that was computed for training during the scoring phase as well. In Machine Learning Studio (classic), you can do this easily by saving the normalization method as a transform, and then using Apply Transformation to apply the z-score to the input data before scoring.

Machine Learning Studio (classic) provides support for creating and then applying many different kinds of custom transformations. For example, you might want to save and then re-use transformations that do the following:

* Remove or replace missing values, using Clean Missing Data
* Bin, scale, and normalize data, using Normalize Data or Group Data into Bins
* Create a set of compact features by calculating joint probability distribution for a dataset, using the Learning with Counts modules.

**How to use Apply Transformation**

1. Add the Apply Transformation module to your experiment. You can find thi module under Machine Learning, in the Score category.
2. Locate an existing transformation to use as an input.

If the transformation was created earlier in the experiment (for example, as part of a cleaning or data scaling operation) typically the ITransform interface object is available on the module's right-hand output. Connect that output to the left-hand input of Apply Transformation. Previously saved transformations can be found in the Transforms group in the left navigation pane.

1. Connect the dataset that you want to transform. The dataset should have exactly the same schema (number of columns, column names, data types) as the dataset for which the transformation was first designed.
2. No other parameters need to be set; all customization is done when defining the transformation.
3. To apply a transformation to the new dataset, run the experiment.

**Assign Data to Clusters**

To generate predictions using a clustering model that was trained using the K-Means clustering algorithm. The module returns a dataset that contains the probable assignments for each new data point. It also creates a PCA (Principal Component Analysis) graph to help you visualize the dimensionality of the clusters.

**How to use Assign Data to Clusters**

1. In Machine Learning Studio (classic), locate a previously trained clustering model. You can create and train a clustering model by using either of these methods:

* Configure the K-means algorithm using the K-Means Clustering module, and then train the model using a dataset and the Train Clustering Model module.
* Configure a range of options for the K-means algorithm using K-Means Clustering and then train the model using the Sweep Clustering module.

You can also add an existing trained clustering model from the Saved Models group in your workspace.

1. Attach the trained model to the left input port of Assign Data to Clusters.
2. Attach a new dataset as input. In this dataset, labels are optional. Generally, clustering is an unsupervised learning method so it is not expected that you will know categories in advance. However, the input columns must be the same as the columns that were used in training the clustering model, or an error occurs.
3. Leave the option Check for Append or Uncheck for Result Only selected if you want the results to contain the full input dataset, together with a column indicating the results (cluster assignments). If you deselect this option, you get back just the results. This might be useful when creating predictions as part of a web service.
4. Run the experiment.

**Results**

The Assign Data to Clusters module returns two types of results on the Results dataset output:

* To see the separation of clusters in the model, click the output of the module and select Visualize

This command displays a Principal Component Analysis (PCA) graph that maps the collection of values in each cluster to two component axes.

* The first component axis is the combined set of features that captures the most variance in the model. It is plotted on the x-axis (Principal Component 1).
* The next component axis represents some combined set of features that is orthogonal to the first component and that adds the next most information to the chart. It is plotted on the y-axis (Principal Component 2).

From the graph, you can see the separation between the clusters, and how the clusters are distributed along the axes that represent the principal components.

* To view the table of results for each case in the input data, attach the Convert to Dataset module, and visualize the results in Studio (classic).

This dataset contains the cluster assignments for each case, and a distance metric that gives you some indication of how close this particular case is to the center of the cluster.

| **Output column name** | **Description** |
| --- | --- |
| Assignments | A 0-based index that indicates which cluster the data point was assigned to. |
| DistancesToClusterCenter no. *n* | For each data point, this value indicates the distance from the data point to the center of the assigned cluster, and the distance to other clusters. The metric used to calculate distance is determined when you configure the K-means clustering model. |

**Score Matchbox Recommender**

Scores predictions for a dataset using the Matchbox recommender.To create predictions based on a trained recommendation model, based on the Matchbox algorithm from Microsoft Research.

The Matchbox recommender can generate four different kinds of predictions:

* Predict ratings for a given user and item
* Recommend items to a given user
* Find users related to a given user
* Find items related to a given item

When creating the latter three kinds of predictions, you can operate in either production mode or evaluation mode.

* ***Production mode*** considers all users or items, and is typically used in a web service. You can create scores for new users, not just users seen during training.
* ***Evaluation mode*** operates on a reduced set of users or items that can be evaluated, and is typically used during experimentation.

**Matchbox recommender**

The goal of creating a recommendation system is to recommend one or more "items" to "users" of the system. Examples of an item could be a movie, restaurant, book, or song. A user could be a person, group of persons, or other entity with item preferences.

There are two principal approaches to recommender systems. The first is the content-based approach, which makes use of features for both users and items. Users may be described by properties such as age and gender, and items may be described by properties such as author and manufacturer. Typical examples of content-based recommendation systems can be found on social matchmaking sites. The second approach is collaborative filtering, which uses only identifiers of the users and the items and obtains implicit information about these entities from a (sparse) matrix of ratings given by the users to the items. We can learn about a user from the items they have rated and from other users who have rated the same items.

The Matchbox recommender combines collaborative filtering with a content-based approach. It is therefore considered a hybrid recommender. When a user is relatively new to the system, predictions are improved by making use of the feature information about the user, thus addressing the well-known "cold-start" problem. However, once there are a sufficient number of ratings from a particular user, it is possible to make fully personalized predictions for them based on their specific ratings rather than on their features alone. Hence, there is a smooth transition from content-based recommendations to recommendations based on collaborative filtering. Even when user or item features are not available, Matchbox still works in its collaborative filtering mode.

**How to configure Score Matchbox Recommender**

This module supports different types of recommendations, each with different requirements. Click the link for the type of data you have and the type of recommendation you want to create.

* Predict ratings
* Recommend items
* Find related users
* Find related items

***Predict ratings***

When you predict ratings, the model calculates how a given user will react to a particular item, given the training data. Therefore, the input data for scoring must provide both a user and the item to rate.

1. Add a trained recommendation model to your experiment, and connect it to Trained Matchbox recommender. You must create the model by using Train Matchbox Recommender.
2. Recommender prediction kind: Select Rating Prediction. No further parameters are required.
3. Add the data for which you wish to make predictions, and connect it to Dataset to score. To predict ratings, the input dataset must contain user-item pairs. The dataset can contain an optional third column of ratings for the user-item pair in the first and second columns, but the third column will be ignored during prediction.
4. (Optional). If you have a dataset of user features, connect it to User features.

The dataset of user features should contain the user identifier in the first column. The remaining columns should contain values that characterize the users, such as their gender, preferences, location, etc.

Features of users who have rated items are ignored by Score Matchbox Recommender, because they have already been learned during training. Therefore, filter your dataset in advance to include only cold-start users, or users who have not rated any items.

1. If you have a dataset of item features, you can connect it to Item features.

The item features dataset must contain an item identifier in the first column. The remaining columns should contain values that characterize the items. Features of rated items are ignored by Score Matchbox Recommender as they have already been learned during training. Therefore, restrict your scoring dataset to cold-start items, or items that have not been rated by any users.

1. Use the optional fifth input port, named Training Dataset, to remove items that have already been rated from the prediction results. To apply this filter, connect the original training dataset to the input port.
2. Run the experiment.

**Results for rating predictions**

The output dataset contains three columns, containing the user, the item, and the predicted rating for each input user and item.

Additionally, the following changes are applied during scoring:

* Missing values in a user or item feature columns are automatically replaced with the mode of its non-missing training set values.
* All user and item features are rescaled by the corresponding maximum absolute values seen in training.
* No translation is applied to the feature values, to maintain their sparsity.
* String-valued features are converted into a set of binary-valued indicator features.

***Recommend***

To recommend items for users, you provide a list of users and items as input. From this data, the model uses its knowledge about existing items and users to generate a list of items with probable appeal to each user. You can customize the number of recommendations returned, and set a threshold for the number of previous recommendations that are required in order to generate a recommendation.

1. Add a trained recommendation model to your experiment, and connect it to Trained Matchbox recommender. You must create the model by using Train Matchbox Recommender.
2. To recommend items for a given list of users, set Recommender prediction kind to Item Recommendation.
3. Recommended item selection: Indicate whether you are using the scoring module in production or for model evaluation, by choosing one of these values:

* **From Rated Items (for model evaluation):** Select this option if you are developing or testing a model. This option enables evaluation mode, and the module makes recommendations only from those items in the input dataset that have been rated.
* **From All Items:** Select this option if you are setting up an experiment to use in a Web service or production. This option enables production mode, and the module makes recommendations from all items seen during training.

1. Add the dataset for which you want to make predictions, and connect it to Dataset to score.

* If you choose the option, From All Items, the input dataset should consist of one and only one column, containing the identifiers of users for which to make recommendations. If the dataset contains more than one column, an error is raised. Use the Select Columns in Dataset module to remove extra columns from the input dataset.
* If you choose the option, From Rated Items (for model evaluation), the input dataset should consist of user-item pairs. The first column should contain the user identifier. The second column should contain the corresponding item identifiers. The dataset can include a third column of user-item ratings, but this column is ignored.

1. (Optional). If you have a dataset of user features, connect it to User features.

The first column in the user features dataset should contain the user identifier. The remaining columns should contain values that characterize the user, such as their gender, preferences, location, etc.

Features of users who have rated items are ignored by Score Matchbox Recommender, because these features have already been learned during training. Therefore, you can filter your dataset in advance to include only cold-start users, or users who have not rated any items.

1. (Optional) If you have a dataset of item features, you can connect it to Item features.

The first column in the item features dataset must contain the item identifier. The remaining columns should contain values that characterize the items.

Features of rated items are ignored by Score Matchbox Recommender, because these features have already been learned during training. Therefore, you can restrict your scoring dataset to cold-start items, or items that have not been rated by any users.

1. Maximum number of items to recommend to a user: Type the number of items to return for each user. By default, 5 items are recommended.
2. Minimum size of the recommendation pool per user: Type a value that indicates how many prior recommendations are required. By default, this parameter is set to 2, meaning the item must have been recommended by at least two other users. This option should be used only if you are scoring in evaluation mode. The option is not available if you select From All Items.
3. Run the experiment.

**Results of item recommendation**

The scored dataset returned by Score Matchbox Recommender lists the recommended items for each user.

* The first column contains the user identifiers.
* A number of additional columns are generated, depending on the value you set for Maximum number of items to recommend to a user. Each column contains a recommended item (by identifier). The recommendations are ordered by user-item affinity, with the item with highest affinity put in column, Item 1.

***Find related users***

The option to find related users is useful if you are recommending "people like you", or if you are creating a pool of similar users on which to base other types of predictions.

1. Add a trained recommendation model to your experiment, and connect it to Trained Matchbox recommender. You must create the model by using Train Matchbox Recommender.
2. Recommender prediction kind: Select Related Users.
3. Related user selection: Indicate how you will be using the model for scoring, and specify the pool of users on which to base the scores as follows:

* **From All Users**: Select this option if you are setting up an experiment to use in a Web service or production, or if you need to make predictions for new users. This option enables production mode, and the module bases its recommendation only on users seen during training.
* **From Users That Rated Items (for model evaluation):** Select this option if you are developing or testing a model. This option enables evaluation mode, and the model bases its recommendations on the users in the test set who have rated some common items.

1. Connect a dataset that contains the users for which to generate predictions. The format for this dataset depends on whether you are using the scoring module in production mode or evaluation mode.

* Production mode, using From All Items

The dataset to score must consist of users for which you wish to find related users. The first and only column should contain the user identifiers. If other columns are included, an error is raised. Use the Select Columns in Dataset module to remove unnecessary columns.

* Evaluation mode, using From Rated Items (for model evaluation)

The dataset to score should consist of 2-3 columns, containing user-item pairs. The first column should contain user identifiers. The second column should contain item identifiers. The dataset can include a third column of ratings (by the user in column 1 for the item in column 2), but the ratings column will be ignored.

1. Maximum number of related users to find for a user: Type a number that indicates the maximum number of predictions you want for each user. The default is 5, meaning that at most five related users can be returned, but in some cases there might be fewer than 5.
2. In evaluation mode (From Users That Rated Items), configure these additional parameters:

* **Minimum number of items that the query user and the related user must have rated in common**: This value sets a threshold for recommendations. The number that you type represents the minimum number of items that your target user and the potential related user must have rated. The default value is 2, meaning that, at minimum, two items must have been rated by both users.
* **Minimum size of the related user pool for a single user:** This value controls the minimum number of similar users needed to create a recommendation. By default, the value is 2, meaning that if you have as few as two users who are related by virtue of rating the same items, you can consider them related and generate a recommendation.

1. (Optional). If you have a dataset of user features, connect it to User features.

The first column in the user features dataset should contain the user identifier. The remaining columns should contain values that characterize the user, such as gender, preferences, location, etc.

Features of users who have rated items are ignored by Score Matchbox Recommender as these features have already been learned during training. Therefore, filter your dataset in advance to include only cold-start users, or users who have not rated any items.

1. (Optional) If you have a dataset of item features, connect it to Item features.

The first column in the item features dataset must contain the item identifier. The remaining columns should contain values that characterize the items.

Features of rated items are ignored by Score Matchbox Recommender as these features have already been learned during training. Therefore, you can restrict your scoring dataset to cold-start items, or items which have not been rated by any users.

1. Run the experiment.

**Results for related users**

The scored dataset returned by Score Matchbox Recommender lists the users who are related to each users in the input dataset.

For each user specified in the input dataset, the result dataset contains a set of related users.

* The first column contains the identifier of the target user (the user provided as input).
* Additional columns are generated containing the identifiers of related users. The number of additional columns depends on the value you set in the option, Maximum number of related users to find for a user.
* Related users are ordered by the strength of the relation to the target user, with the most strongly related user in the column, Related User 1.

**Find related items**

By predicting related items, you can generate recommendations for users based on items that have already been rated.

1. Add a trained recommendation model to your experiment, and connect it to Trained Matchbox recommender. You must create the model by using Train Matchbox Recommender.
2. Recommender prediction kind: Select Related Items.
3. Connect a dataset that contains the users for which to generate predictions. The format for this dataset depends on whether you are using the scoring module in production mode or evaluation mode.

* Production mode, using From All Items

The dataset to score must consist of items for which you wish to find related users. The first and only column should contain the item identifiers. If other columns are included, an error is raised. Use the Select Columns in Dataset module to remove unnecessary columns.

* Evaluation mode, using From Rated Items (for model evaluation)

The dataset to score should consist of 2-3 columns, containing user-item pairs. The first column should contain user identifiers. The second column should contain item identifiers. The dataset can include a third column of ratings (by the user in column 1 for the item in column 2), but the ratings column are ignored.

1. Maximum number of related items to find for an item>: Type a number that indicates the maximum number of predictions you want for each item. The default is 5, meaning that at most five related items can be returned, but there might be fewer than 5.
2. If you are using evaluation mode (From Users That Rated Items), configure these additional parameters:

* **Minimum number of items that the query item and the related item must have been rated by in common:** This value sets a threshold for recommendations. The number that you type represents the minimum number of items that have been rated by the target user and some related user. The default value is 2, meaning that, at minimum, two items must have been rated by the target user and the related user.
* **Minimum size of the related item pool for a single item:** This value controls the minimum number of similar items needed to create a recommendation. By default, the value is 2, meaning that, if you have as few as two items that are related by virtue of having been rated by the same users, you can consider them related and generate a recommendation.

1. (Optional). If you have a dataset of user features, connect it to User features.

The first column in the user features dataset should contain the user identifier. The remaining columns should contain values that characterize the user, such as their gender, preferences, location, etc. Features of users who have rated items are ignored by Score Matchbox Recommender, because these features have already been learned during training. Therefore, you can filter your dataset in advance to include only cold-start users, or users who have not rated any items.

1. (Optional) If you have a dataset of item features, you can connect it to Item features.

The first column in the item features dataset must contain the item identifier. The remaining columns should contain values that characterize the item. Features of rated items are ignored by Score Matchbox Recommender, because these features have already been learned during training. Therefore, you can restrict your scoring dataset to cold-start items, or items which have not been rated by any users.

1. (Optional) In a predictive experiment, you can use a fifth input port, named Training Dataset, to remove existing users that were included in the model training data from the prediction results. To apply this filter, connect the original training dataset to the input port.
2. Run the experiment.

**Results for related items**

The scored dataset returned by Score Matchbox Recommender lists the related items for each item in the input dataset.

* The first column contains the identifier of the target item (the item provided as input).
* Additional columns are generated containing the identifiers of related items. The number of additional columns depends on the value you set in the option, Maximum number of related items to find for an item.The related items are ordered by the strength of the relation to the target item, with the most strongly related item in the column, Related Item 1.

**Cold-start users and recommendations**

Typically, to create recommendations, the Score Matchbox Recommender module requires the same inputs that you used when training the model, including a user ID. That is because the algorithm needs to know if it has learned something about this user during training. However, for new users, you might not have a user ID, only some user features such as age, gender, and so forth.

You can still create recommendations for users who are new to your system, by handling them as cold-start users. For such users, the recommendation algorithm does not use past history or previous ratings, only user features.

For purposes of prediction, a cold-start user is defined as a user with an ID that has not been used for training. To ensure that IDs do not match IDs used in training, you can create new identifiers. For example, you might generate random IDs within a specified range, or allocate a series of IDs in advance for cold-start users. However, if you do not have any collaborative filtering data, such as a vector of user features, you are better of using a classification or regression learner.

**Production use of the Matchbox recommender**

If you have experimented with the Matchbox recommender and then move the model to production, be aware of these key differences when using the recommender in evaluation mode and in production mode:

* Evaluation, by definition, requires predictions that can be verified against the ground truth in a test set. Therefore, when you evaluate the recommender, it must predict only items that have been rated in the test set. This necessarily restricts the possible values that are predicted.

However, when you operationalize the model, you typically change the prediction mode to make recommendations based on all possible items, in order to get the best predictions. For many of these predictions, there is no corresponding ground truth, so the accuracy of the recommendation cannot be verified in the same way as during experimentation.

* If you do not provide a user ID in production, and provide only a feature vector, you might get as response a list of all recommendations for all possible users. Be sure to provide a user ID.

To limit the number of recommendations that are returned, you can also specify the maximum number of items returned per user.

* It is not possible to generate predictions only for items that have not previously been rated. This is by design.

The reason is that, in order to recommend only the items that have not been rated, the recommender would need to store the entire training data set with the model, which would increase your use of storage. If you want to recommend only items that have not been seen by your user, you can request more items to recommend, and then manually filter out the rated ones.

**Module parameters**

| **Name** | **Range** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- | --- |
| Recommender prediction kind | List | Prediction kind | Item Recommendation | Specify the type of prediction the recommender should output |
| Recommended item selection | List | Item selection | From Rated Items (for model evaluation) | Select the set of items to make recommendations from |
| Related user selection | List | User selection | From Users That Rated Items (for model evaluation) | Select the set of users to use when finding related items |
| Related item selection | List | [Item selection | From Rated Items (for model evaluation) | Select the set of items to use when finding related items |

**Score Model**

Scores predictions for a trained classification or regression model.

**How to use Score Model**

1. Add the Score Model module to your experiment in Studio (classic).
2. Attach a trained model and a dataset containing new input data. The data should be in a format compatible with the type of trained model you are using. The schema of the input dataset should also generally match the schema of the data used to train the model.
3. Run the experiment.

**Results**

After you have generated a set of scores using Score Model:

* To generate a set of metrics used for evaluating the model’s accuracy (performance). you can connect the scored dataset to Evaluate Model,
* Right-click the module and select Visualize to see the a sample of the results.
* Save the results to a dataset.

The score, or predicted value, can be in many different formats, depending on the model and your input data:

* For classification models, Score Model outputs a predicted value for the class, as well as the probability of the predicted value.
* For regression models, Score Model generates just the predicted numeric value.
* For image classification models, the score might be the class of object in the image, or a Boolean indicating whether a particular feature was found.