**Feature Selection modules**

**Feature Selection**

Feature selection is an important tool in machine learning. Machine Learning Studio (classic) provides multiple methods for performing feature selection. Choose a feature selection method based on the type of data that you have, and the requirements of the statistical technique that's applied.

Each feature selection module in Machine Learning Studio (classic) uses a dataset as input. Then, the module applies well-known statistical methods to the data columns that are provided as input. The output is a set of metrics that can help you identify the columns that have the best information value.

In machine learning and statistics, feature selection is the process of selecting a subset of relevant, useful features to use in building an analytical model. Feature selection helps narrow the field of data to the most valuable inputs. Narrowing the field of data helps reduce noise and improve training performance.

Often, features are created from raw data through a process of feature engineering. For example, a timestamp in itself might not be useful for modeling until the information is transformed into units of days, months, or categories that are relevant to the problem, such as holiday versus working day.

New users of machine learning might be tempted to include all data that's available. They might expect that the algorithm will find something interesting by using more data. However, feature selection can usually improve your model, and prevent common problems:

* The data contains redundant or irrelevant features, which provide no more information than the currently selected features.
* The data contains irrelevant features that provide no useful information in any context. Including irrelevant fields not only increases the time required to train the data, but also can lead to poor results.
* With some algorithms, having duplicate information in the training data can lead to a phenomenon called multicollinearity. In multicollinearity, the presence of two highly correlated variables can cause the calculations for other variables to become much less accurate.

**Use feature selection in an experiment**

Feature selection typically is performed when you are exploring data and developing a new model. Keep these tips in mind when you use feature selection:

* When testing, add feature selection to your experiment to generate scores that inform your decision of which columns to use.
* Remove feature selection from the experiment when you operationalize a model.
* Run feature selection periodically to ensure that the data and best features haven't changed.

Feature selection is different from feature engineering, which focuses on creating new features out of existing data.

The following feature selection modules are provided in Machine Learning Studio (classic).

**Filter-Based Feature Selection** When you use the Filter-Based Feature Selection module, you can choose from among well-known feature selection methods. The module outputs both the feature selection statistics and the filtered dataset.

**Fisher Linear Discriminant Analysis**

Linear Discriminant Analysis is a supervised learning technique that you can use to classify numerical variables in conjunction with a single categorical target. The method is useful for feature selection because it identifies the combination of features or parameters that best separates the groups.

You can use the Fisher Linear Discriminant Analysis module to generate a set of scores for review, or you can use the replacement dataset that's generated by the module for training.

**Permutation Feature Importance**

Use the Permutation Feature Importance module to simulate the effect of any set of features on your dataset. The module computes performance scores for a model based on the random shuffling of feature values.

The scores that the module returns represent the potential change in the accuracy of a trained model if values change. You can use the scores to determine the effect of individual variables on the model.

**Machine learning algorithms that incorporate feature selection**

Some machine learning algorithms in Machine Learning Studio (classic) optimize feature selection during training. They might also provide parameters that help with feature selection. If you're using a method that has its own heuristic for choosing features, it's often better to rely on that heuristic instead of preselecting features.

These algorithms and feature selection methods are used internally:

* Boosted decision tree models for classification and regression

In these modules, a feature summary is created internally. Features that have a weight of 0 aren't used by any tree splits. When you visualize the best trained model, you can look at each of the trees. If a feature is never used in any tree, the feature is likely a candidate for removal. To optimize selection, it's also a good idea to use parameter sweeping.

* Logistic regression models and linear models

The modules for multiclass and binary logistic regression support L1 and L2 regularization. Regularization is a way of adding constraints during training to manually specify an aspect of the learned model. Regularization typically is used to avoid overfitting. Machine Learning Studio (classic) supports regularization for the L1 or L2 norms of the weight vector in linear classification algorithms:

* L1 regularization is useful if the goal is to have a model that's as sparse as possible.
* L2 regularization prevents any single coordinate in the weight vector from growing too much in magnitude. It's useful if the goal is to have a model with small overall weights.
* L1-regularized logistic regression is more aggressive about assigning a weight of 0 to features. It's useful in identifying features that can be removed.

**Filter-Based Feature Selection**

In general, feature selection refers to the process of applying statistical tests to inputs, given a specified output, to determine which columns are more predictive of the output. The Filter Based Feature Selection module provides multiple feature selection algorithms to choose from, including correlation methods such as Pearsons's or Kendall's correlation, mutual information scores, and chi-squared values. Machine Learning also supports feature value counts as an indicator of information value.

When you use the Filter Based Feature Selection module, you provide a dataset, identify the column that contains the label or dependent variable, and then specify a single method to use in measuring feature importance.

The module outputs a dataset that contains the best feature columns, as ranked by predictive power. It also outputs the names of the features and their scores from the selected metric.

**What is filter-based feature selection and why use it?**

This module for feature selection is called "filter-based" because you use the selected metric to identify irrelevant attributes, and filter out redundant columns from your model. You choose a single statistical measure that suits your data, and the module calculates a score for each feature column. The columns are returned ranked by their feature scores.

By choosing the right features, you can potentially improve the accuracy and efficiency of classification.

You typically use only the columns with the best scores to build your predictive model. Columns with poor feature selection scores can be left in the dataset and ignored when you build a model.

**How to choose a feature selection metric**

The Filter-Based Feature Selection provides a variety of metrics for assessing the information value in each column. This section provides a general description of each metric, and how it is applied. Additional requirements for using each metric are stated in the Technical Notes section and in the instructions for configuring each module.

1. **Pearson Correlation**

Pearson’s correlation statistic, or Pearson’s correlation coefficient, is also known in statistical models as the r value. For any two variables, it returns a value that indicates the strength of the correlation

Pearson's correlation coefficient is computed by taking the covariance of two variables and dividing by the product of their standard deviations. The coefficient is not affected by changes of scale in the two variables.

1. **Mutual Information**

The mutual information score measures the contribution of a variable towards reducing uncertainty about the value of another variable: namely, the label. Many variations of the mutual information score have been devised to suit different distributions.

The mutual information score is particularly useful in feature selection because it maximizes the mutual information between the joint distribution and target variables in datasets with many dimensions.

1. **Kendall Correlation**

Kendall's rank correlation is one of several statistics that measure the relationship between rankings of different ordinal variables or different rankings of the same variable. In other words, it measures the similarity of orderings when ranked by the quantities. Both this coefficient and Spearman’s correlation coefficient are designed for use with non-parametric and non-normally distributed data.

1. **Spearman Correlation**

Spearman's coefficient is a nonparametric measure of statistical dependence between two variables, and is sometimes denoted by the Greek letter rho. The Spearman’s coefficient expresses the degree to which two variables are monotonically related. It is also called Spearman rank correlation, because it can be used with ordinal variables.

1. **Chi Squared**

The two-way chi-squared test is a statistical method that measures how close expected values are to actual results. The method assumes that variables are random and drawn from an adequate sample of independent variables. The resulting chi-squared statistic indicates how far results are from the expected (random) result.

1. **Fisher Score**

The Fisher score (also called the Fisher method, or Fisher combined probability score) is sometimes termed the information score, because it represents the amount of information that one variable provides about some unknown parameter on which it depends.

The score is computed by measuring the variance between the expected value of the information and the observed value. When variance is minimized, information is maximized. Since the expectation of the score is zero, the Fisher information is also the variance of the score.

1. **Count Based**

Count-based feature selection is a simple yet relatively powerful way of finding information about predictors. The basic idea underlying count-based featurization is simple: by calculating counts of individual values within a column, you can get an idea of the distribution and weight of values, and from this, understand which columns contain the most important information.

Count-based feature selection is a non-supervised method of feature selection, meaning you don't need a label column. This method also reduces the dimensionality of the data without losing information.

**How to configure Filter-Based Feature Selection**

This module provides two methods for determining feature scores:

1. Generate feature scores using a traditional statistical metric

You choose a standard statistical metric, and the module computes the correlation between a pair of columns, the label column and a feature column

1. Use count-based feature selection

With the count-based method, the module calculates a score based purely on the values in the column.

**Generate feature scores using a traditional statistical metric**

1. Add the Filter-Based Feature Selection module to your experiment. You can find it in the Feature Selection category in Studio (classic).
2. Connect an input dataset that contains at least two columns that are potential features.

To ensure that a column should be analyzed and a feature score generated, use the Edit Metadata module to set the IsFeature attribute.

1. For Feature scoring method, choose one of the following established statistical methods to use in calculating scores.

|  |  |
| --- | --- |
| Method | Requirements |
| Pearson Correlation | Label can be text or numeric. Features must be numeric. |
| Mutual Information | Labels and features can be text or numeric. Use this method for computing feature importance for two categorical columns. |
| Kendall Correlation | Label can be text or numeric but features must be numeric. |
| Spearman Correlation | Label can be text or numeric but features must be numeric. |
| Chi Squared | Labels and features can be text or numeric. Use this method for computing feature importance for two categorical columns. |
| Fisher Score | Label can be text or numeric but features must be numeric. |

1. Select the Operate on feature columns only option to generate a score only for those columns that have been previously marked as features.

If you deselect this option, the module will create a score for any column that otherwise meets the criteria, up to the number of columns specified in Number of desired features.

1. For Target column, click Launch column selector to choose the label column either by name or by its index (indexes are one-based).

A label column is required for all methods that involve statistical correlation. The module returns a design-time error if you choose no label column or multiple label columns.

1. For Number of desired features, type the number of feature columns you want returned as a result.

* The minimum number of features you can specify is 1, but we recommend that you increase this value.
* If the specified number of desired features is greater than the number of columns in the dataset, then all features are returned, even those with zero scores.
* If you specify fewer result columns than there are feature columns, the features are ranked by descending score, and only the top features are returned.

1. Run the experiment, or select the Filter Based Feature Selection module and then click Run selected.

**Results of feature selection**

After processing is complete:

* To see a complete list of the feature columns that were analyzed, and their scores, right-click the module, select Features, and click Visualize.
* To view the dataset that is generated based on your feature selection criteria, right-click the module, select Dataset, and click Visualize.

If the dataset contains fewer columns than you expected, check the module settings, and the data types of the columns provided as input. For example, if you set Number of desired features to 1, the output dataset contains just two columns: the label column, and the most highly ranked feature column.

**Use count-based feature selection**

1. Add the Filter-Based Feature Selection module to your experiment. You can find it in the list of modules in Studio (classic), in the Feature Selection group.
2. Connect an input dataset that contains at least two columns that are possible features.
3. Select Count Based from the list of statistical methods in the Feature scoring method dropdown list.
4. For Minimum number of non-zero elements, indicate the minimum number of feature columns to include in the output.

By default, the module outputs all columns that meet the requirements. The module cannot output any column that gets a score of zero.

1. Run the experiment, or select just the module, and click Run Selected.

**Results of count-based feature selection**

* To see the list of feature columns with their scores, right-click the module, select Features, and click Visualize .
* To see the dataset containing the analyzed columns, right-click the module, select Dataset, and click Visualize.

Unlike other methods, the Count Based feature selection method does not rank the variables by highest scores, but returns all variables with a non-zero score, in their original order.

String features always get a zero (0) score and are thus are not output.

**Fisher Linear Discriminant Analysis**

To create a new feature dataset that captures the combination of features that best separates two or more classes.

This method is often used for dimensionality reduction, because it projects a set of features onto a smaller feature space while preserving the information that discriminates between classes. This not only reduces computational costs for a given classification task, but can help prevent overfitting.

To generate the scores, you provide a label column and set of numerical feature columns as inputs. The algorithm determines the optimal combination of the input columns that linearly separates each group of data while minimizing the distances within each group. The module returns a dataset containing the compact, transformed features, along with a transformation that you can save and apply to another dataset.

**Linear discriminant analysis**

Linear discriminant analysis is similar to analysis of variance (ANOVA) in that it works by comparing the means of the variables. Like ANOVA, it relies on these assumptions:

* Predictors are independent
* The conditional probability density functions of each sample are normally distributed
* Variances among groups are similar

Linear Discriminant Analysis is sometimes abbreviated to LDA, but this is easily confused with Latent Dirichlet Allocation. The techniques are completely different, so in this documentation, we use the full names wherever possible.

**How to configure Linear Discriminant Analysis**

1. Add your input dataset and check that the input data meets these requirements:

* Your data should be as complete as possible. Rows with any missing values are ignored.
* Values are expected to have a normal distribution. Before using Fisher Linear Discriminant Analysis, review the data for outliers, or test the distribution.
* You should have fewer predictors than there are samples.
* Remove any non-numeric columns. The algorithm examines all valid numeric columns included in the inputs, and return an error if invalid columns are included. If you need to exclude any numeric columns, add a Select Columns in Dataset module before Fisher Linear Discriminant Analysis, to create a view that contains only the columns you wish to analyze. You can rejoin the columns later using Add Columns. The original order of rows is preserved.

1. Connect the input data to the Fisher Linear Discriminant Analysis module.
2. For the Class labels column, click Launch column selector and choose one label column.
3. For the Number of feature extractors, type the number of columns that you want as a result.

For example, if your dataset contains eight numeric feature columns, you might type 3 to collapse them into a new, reduced feature space of only three columns.

It is important to understand that the output columns do not correspond exactly to the input columns, but rather represent a compact transformation of the values in the input columns.

If you use 0 as the value for Number of feature extractors, and n columns are used as input, n feature extractors are returned, containing new values representing the n-dimensional feature space.

1. Run the experiment.

**Results**

The algorithm determines the combination of values in the input columns that linearly separates each group of data while minimizing the distances within each group, and creates two outputs:

* ***Transformed features***. A dataset containing the specified number of feature extractor columns, named col1, col2, col3, and so forth. The output also includes the class or label variable as well.You can use this compact set of values for training a model.
* ***Fisher linear discriminant analysis transformation***. A transformation that you can save and then apply to a dataset that has the same schema. This is useful if you are analyzing many datasets of the same type and want to apply the same feature reduction to each. The dataset that you apply it to should have the same schema.

**Usage tips**

This method works only on continuous variables, not categorical or ordinal variables.

* Rows with missing values are ignored when computing the transformation matrix.
* If you save a transformation from an experiment, the transformations computed from the original experiment are reapplied to each new set of data, and are not recomputed. Therefore, if you want to compute a new feature set for each set of data, use a new instance of Fisher Linear Discriminant Analysis for each dataset.

Implementation details

The dataset of features is transformed using eigenvectors. The eigenvectors for the input dataset are computed based on the provided feature columns, also called a discrimination matrix. The transformation output by the module contains these eigenvectors, which can be applied to transform another dataset that has the same schema.

**Permutation Feature Importance**

To compute a set of feature importance scores for your dataset. You use these scores to help you determine the best features to use in a model. In this module, feature values are randomly shuffled, one column at a time, and the performance of the model is measured before and after. You can choose one of the standard metrics provided to measure performance.

The scores that the module returns represent the change in the performance of a trained model, after permutation. Important features are usually more sensitive to the shuffling process, and will thus result in higher importance scores.

**How to use Permutation Feature Importance**

To generate a set of feature scores requires that you have an already trained model, as well as a test dataset.

1. Add the Permutation Feature Importance module to your experiment. You can find this module in the Feature Selection category.
2. Connect a trained model to the left input. The model must be a regression model or classification model.
3. On the right input, connect a dataset, preferably one that is different from the dataset used for training the model. This dataset is used for scoring based on the trained model, and for evaluating the model after feature values have been changed.
4. For Random seed, type a value to use as seed for randomization. If you specify 0 (the default), a number is generated based on the system clock.

A seed value is optional, but you should provide a value if you want reproducibility across runs of the same experiment.

1. For Metric for measuring performance, select a single metric to use when computing model quality after permutation.

Machine Learning Studio (classic) supports the following metrics, depending on whether you are evaluating a classification or regression model:

* Classification

Accuracy, Precision, Recall, Average Log Loss

* Regression

Precision, Recall, Mean Absolute Error, Root Mean Squared Error, Relative Absolute Error, Relative Squared Error, Coefficient of Determination

1. Run the experiment.
2. The module outputs a list of feature columns and the scores associated with them, ranked in order of the scores, descending.