**Impute Missing Data:**

*Replace using MICE*

*For each missing value, this option assigns a new value, which is calculated by using a method described in the statistical literature as "Multivariate Imputation using Chained Equations" or "Multiple Imputation by Chained Equations".*

*Custom substitution value*

*Use this option to specify a placeholder value (such as a 0 or NA) that applies to all missing values. The value that you specify as a replacement must be compatible with the data type of the column.*

*Replace with mean*

*Calculates the column mean and uses the mean as the replacement value for each missing value in the column.*

*Replace with mode*

*Calculates the mode for the column and uses that as the replacement value for every missing value in the column.*

*Replace with median*

*Calculates the column median value and assigns that as the replacement for any missing value in the column.*

*Remove entire row*

*Completely removes any row in the dataset that has one or more missing values. This is useful if the missing value can be considered randomly missing.*

*Remove entire column*

*Completely removes any column in the dataset that has one or more missing values.*

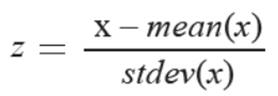
*Replace using Probabilistic PCA*

*Replaces the missing values by using a linear model that analyzes the correlations between the columns and estimates a low-dimensional approximation of the data, from which the full data is reconstructed.*

**Normalize Data:**

Zscore

This option converts all values to a z-score. The values in the column are transformed using the following formula:

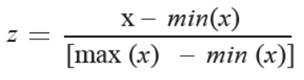


MinMax

The min-max normalizer linearly rescales every feature to the [0,1] interval.

Rescaling to the [0,1] interval is done by shifting the values of each feature so that the minimal value is 0, and then dividing by the new maximal value (which is the difference between the original maximal and minimal values).

The values in the column are transformed using the following formula:



LogNormal

This option converts all values to a lognormal scale.

The values in the column are transformed using the following formula:

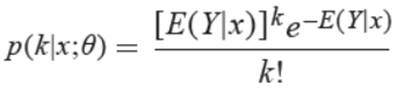
C:\Users\jboswell\Downloads\image027.jpg

Here μ and σ are the parameters of the distribution, computed empirically from the data as maximum likelihood estimates, for each column separately.

TanH

All values are converted to a hyperbolic tangent.

The values in the column are transformed using the following formula:



**Linear Regression:**

Online gradient descent

Gradient descent is a method that minimizes the amount of error at each step of the model training process. There are many variations on gradient descent and its optimization for various learning problems has been extensively studied.

Ordinary least squares

Least squares linear regression is one of the most commonly used techniques in predictive analytics. This method assumes that there is a fairly strong linear relationship between the inputs and the dependent variable. Ordinary least squares refers to the loss function, which computes error as the sum of the square of distance from the actual value to the predicted line, and fits the model by minimizing the squared error.

**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.

**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.

Mean Absolute Error (MAE): The average of absolute errors (an error is the difference between the predicted value and the actual value).

Root Mean Squared Error (RMSE): The square root of the average of squared errors of predictions made on the test dataset.

Relative Absolute Error: The average of absolute errors relative to the absolute difference between actual values and the average of all actual values.

Relative Squared Error: The average of squared errors relative to the squared difference between the actual values and the average of all actual values.

Coefficient of Determination: Also known as the R squared value, this is a statistical metric indicating how well a model fits the data.

**Boosted Decision Tree Regression:**

**Single Parameter**

If you know how you want to configure the model, you can provide a specific set of values as arguments. You might have learned these values by experimentation or received them as guidance.

**Parameter Range**

If you are not sure of the best parameters, you can find the optimal parameters by specifying multiple values and using a parameter sweep to find the optimal configuration.

For **Maximum number of leaves per tree**, indicate the maximum number of terminal nodes (leaves) that can be created in any tree. By increasing this value, you potentially increase the size of the tree and get better precision, at the risk of overfitting and longer training time.

For **Minimum number of samples per leaf node**, indicate the minimum number of cases required to create any terminal node (leaf) in a tree. By increasing this value, you increase the threshold for creating new rules. For example, with the default value of 1, even a single case can cause a new rule to be created. If you increase the value to 5, the training data would have to contain at least 5 cases that meet the same conditions.

For **Learning rate**, type a number between 0 and 1 that defines the step size while learning. The learning rate determines how fast or slow the learner converges on the optimal solution. If the step size is too big, you might overshoot the optimal solution. If the step size is too small, training takes longer to converge on the best solution.

For **Number of trees constructed**, indicate the total number of decision trees to create in the ensemble. By creating more decision trees, you can potentially get better coverage, but training time will increase.

For **Random number seed**, you can type a non-negative integer to use as the random seed value.

Select **Allow unknown categorical levels** option to create a group for unknown values in the training and validation sets. If you deselect this option, the model can accept only the values that are contained in the training data.

**K-Means Clustering:**

Single Parameter. If you know the exact parameters you want to use in the clustering model, you can provide a specific set of values as arguments.

Parameter Range. If you are not sure of the best parameters, you can find the optimal parameters by specifying multiple values and using the Sweep Clustering module to find the optimal configuration.

**For this experiment**, we will use Single parameter.

**For Number of Centroids**, type the number of clusters you want the algorithm to begin with.

The properties Initialization are used to specify the algorithm that is used to define the initial cluster configuration.

*First N*. Some initial number of data points are chosen from the data set and used as the initial means

*Random*. The algorithm randomly places a data point in a cluster and then computes the initial mean to be the centroid of the cluster's randomly assigned points.

*K-Means++*. This is the default method for initializing clusters.

*K-Means++Fast*. A variant of the K-means ++ algorithm that was optimized for faster clustering.

*Evenly*. Centroids are located equidistant from each other in the d-Dimensional space of n data points.

*Use label column*. The values in the label column are used to guide the selection of centroids.

**For Random number seed**, optionally type a value to use as the seed for the cluster initialization. This value can have a significant effect on cluster selection.

**For Metric**, choose the function to use for measuring the distance between cluster vectors, or between new data points and the randomly chosen centroid. Azure Machine Learning supports the following cluster distance metrics:

*Euclidean*. The Euclidean distance is commonly used as a measure of cluster scatter for K-means clustering. This metric is preferred because it minimizes the mean distance between points and the centroids.

*Cosine*. The cosine function is used to measure cluster similarity. Cosine similarity is useful in cases where you do not care about the length of a vector, only its angle.

For Iterations, type the number of times the algorithm should iterate over the training data before finalizing the selection of centroids. You can adjust this parameter to balance accuracy vs. training time.

**For Assign label mode**, choose an option that specifies how a label column if present in the dataset, should be handled.

Because K-means clustering is an unsupervised machine learning method, labels are optional. However, if your dataset already has a label column, you can use those values to guide selection of the clusters, or you can specify that the values be ignored.

*Ignore label column*. The values in the label column are ignored and are not used in building the model.

*Fill missing values*. The label column values are used as features to help build the clusters. If any rows are missing a label, the value is imputed by using other features.

*Overwrite from closest to center*. The label column values are replaced with predicted label values, using the label of the point that is closest to the current centroid.

**Anomaly Detection:**

Currently, Azure Machine Learning Studio supports the following anomaly detectors.

One-Class Support Vector Machine: One class support vector machine implements a binary classifier where the training data consists of examples of only one class (normal data). The model attempts to separate the collection of training data from the origin using maximum margin. By default, a radial basis kernel is used.

PCA-based: This algorithm uses PCA (Principal Component Analysis) to approximate the subspace containing the normal class. The subspace is spanned by orthonormal eigenvectors associated with the top eigenvalues of the data covariance matrix. For each new input, the anomaly detector first computes its projection on the eigenvectors, then it computes the normalized reconstruction error. This norm error is the anomaly score. The higher the error, the more anomalous the instance is.

Trainer

Anomaly detection has a separate trainer module since labels are optional during training. For all other supervised learning tasks, labels are required.

Learner Parameters

The parameters of the learner can be specified in two different ways using the Create Trainer mode option found in the module properties.

Single Parameter: In this mode, the parameters are specified manually.

Parameter Sweep: This mode is meant to be used in conjunction with the Sweep Parameters module. Multiple values or a range of values can be specified for each of the tunable parameters. Sweep Parameters module can then be used to optimize these values.

In the experiment, we use the Parameter Sweep mode and specify a grid of value for each of the parameters. The parameter sweep is then executed to optimize F-score value of the detector. Other metrics relevant to classification such as AUC, ROC, precision/recall can also be optimized instead.

Parameter Sweep

The parameter sweep takes an untrained learner, training data and optionally validation data. For the anomaly detector, we need to specify both training and validation data since the training data consists of examples from a single class whereas validation data consists of examples from both classes. This excludes the use of cross-validation to optimize parameters.

The Sweep Parameters module outputs a learner with the best settings. We can use this learner directly or re-train the learner with this setting using the entire training data.

Scoring

Predictions from the anomaly detectors can be obtained using the Score Model module. The output of one-class SVM anomaly detectors consists of uncalibrated scores that may be possibly unbounded. We normalize the scores to match the dynamic range of scores generated by the PCA anomaly detector (which lies in [0,1] range).

Results

Anomaly detectors can be evaluated on the same metrics as binary classifiers. Area under the ROC curve provides a good way to measure the discriminatory power of the anomaly detectors.