**BJ- ML\_Orange- t-SNE-NNs- parameters**

**Widget Rank**

The **Rank** widget scores variables according to their correlation with discrete or numeric target variable, based on applicable internal scorers.

1. Select scoring methods. See the options for classification, regression and unsupervised data in the **Scoring methods** section.
2. Select attributes to output. *None* won't output any attributes, while *All* will output all of them. With manual selection, select the attributes from the table on the right. *Best ranked* will output n best ranked attributes. If *Send Automatically* is ticked, the widget automatically communicates changes to other widgets.
3. Status bar. Produce a report by clicking on the file icon. Observe input and output of the widget. On the right, warnings and errors are shown.

Scoring methods (classification)

1. Information Gain: the expected amount of information (reduction of entropy)
2. [Gain Ratio](https://en.wikipedia.org/wiki/Information_gain_ratio): a ratio of the information gain and the attribute's intrinsic information, which reduces the bias towards multivalued features that occurs in information gain
3. [Gini](https://en.wikipedia.org/wiki/Gini_coefficient): the inequality among values of a frequency distribution
4. [ANOVA](https://en.wikipedia.org/wiki/One-way_analysis_of_variance): the difference between average values of the feature in different classes
5. [Chi2](https://en.wikipedia.org/wiki/Chi-squared_distribution): dependence between the feature and the class as measured by the chi-square statistic
6. [ReliefF](https://en.wikipedia.org/wiki/Relief_(feature_selection)): the ability of an attribute to distinguish between classes on similar data instances
7. [FCBF (Fast Correlation Based Filter)](https://www.aaai.org/Papers/ICML/2003/ICML03-111.pdf): entropy-based measure, which also identifies redundancy due to pairwise correlations between features

Additionally, you can connect certain learners that enable scoring the features according to how important they are in models that the learners build (e.g. [Logistic Regression](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/logisticregression.html), [Random Forest](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/randomforest.html), [SGD](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/stochasticgradient.html)). Please note that the data is normalized before ranking.

Scoring methods (regression)

1. [Univariate Regression](https://en.wikipedia.org/wiki/Simple_linear_regression): linear regression for a single variable
2. [RReliefF](http://www.clopinet.com/isabelle/Projects/reading/robnik97-icml.pdf): relative distance between the predicted (class) values of the two instances.

Additionally, you can connect regression learners (e.g. [Linear Regression](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/linearregression.html), [Random Forest](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/randomforest.html), [SGD](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/stochasticgradient.html)). Please note that the data is normalized before ranking.

Scoring method (unsupervised)

Currently, only [PCA](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/unsupervised/PCA.html) is supported for unsupervised data. Connect PCA to Rank to obtain the scores. The scores correspond to the correlation of a variable with the individual principal component.

Scoring with learners

Rank can also use certain learners for feature scoring. See [Learners as Scorers](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/learners-as-scorers/index.html) for an example.

**Widget t-SNE:**

**1. Exaggeration Parameter:**

The Exaggeration parameter in the t-SNE widget controls the emphasis placed on separating similar data points in the visualization. It acts like a zoom lens:

* **Low Exaggeration (default):** Data points are clustered more tightly, highlighting global structure in the data.
* **High Exaggeration:** Data points are spread out more, emphasizing local differences and potentially revealing finer-grained clusters.

**2. PCA Components vs. Ranked Attributes:**

There's a key distinction between the number of PCA components in t-SNE and the best-ranked attributes in the Rank widget:

* **PCA Components (t-SNE):** These components are derived from Principal Component Analysis (PCA), a dimensionality reduction technique. In t-SNE, a smaller number of components (typically 2 or 3) captures the most significant variance in the data for visualization in a lower-dimensional space.
* **Ranked Attributes (Rank):** This widget ranks features based on their importance or relevance to the target variable (in classification tasks). It doesn't directly reduce dimensionality but highlights potentially informative features.

**3. Labels in t-SNE Widget:**

* **No Label:** Use this option for purely exploratory visualization, where you're primarily interested in observing the overall data distribution and potential cluster patterns.
* **Indicate Something:** Consider using labels if you have specific attributes or categories you want to visualize separately. This helps identify clusters or patterns associated with those labels.

**4. t-SNE After Rank vs. Direct from Preprocess:**

The choice between using t-SNE after the Rank widget or directly after the Preprocess widget depends on your goals:

* **t-SNE After Rank:** This approach helps focus the visualization on potentially more informative features identified by the Rank widget. You're essentially using the ranked attributes to guide the t-SNE dimensionality reduction, potentially revealing more meaningful clusters or patterns.
* **t-SNE Directly from Preprocess:** This bypasses feature ranking and lets t-SNE work with the entire preprocessed dataset. This can be helpful for general exploration or if you're unsure which features might be most relevant.

**General Recommendations:**

* Start with a low Exaggeration value in t-SNE for a broad overview of the data distribution.
* Experiment with different numbers of PCA components to see how it affects the visualization.
* Consider using labels in t-SNE if you have specific features or categories you want to explore.
* Use the Rank widget before t-SNE if you want to focus on potentially more informative attributes.
* Try both approaches (Rank before t-SNE and directly from Preprocess) to see which one provides the most insightful visualization for your specific dataset and analysis goals.

**After t-SNE can be used (coonecting with )**

**1. KMeans:**

* **Purpose:** Performs K-Means clustering on the data points transformed by t-SNE. K-Means attempts to partition the data into a predefined number of clusters (k).
* **Benefits:**
  + Identifies distinct clusters within the t-SNE visualization.
  + Helps interpret the structure revealed by t-SNE in terms of meaningful clusters.
  + Provides cluster labels that can be used for further analysis, such as comparing cluster properties or evaluating model performance.

**2. DBSCAN:**

* **Purpose:** Performs Density-Based Spatial Clustering of Applications with Noise (DBSCAN) on the t-SNE data points. Unlike K-Means, DBSCAN doesn't require a pre-defined number of clusters and can identify clusters of arbitrary shapes.
* **Benefits:**
  + Useful for datasets where the number of clusters is unknown or potentially varies in density.
  + Can identify outliers or noise points that might not fall into well-defined clusters in the t-SNE space.

**3. Silhouette Plot:**

* **Purpose:** Evaluates the quality of clustering results (typically from KMeans or DBSCAN) based on the Silhouette score. The Silhouette score measures how well data points are assigned to their respective clusters.
* **Benefits:**
  + Helps assess the effectiveness of the clustering performed after t-SNE, especially when using KMeans with a pre-defined number of clusters.
  + Provides insights into whether clusters are well-separated or if there might be overlapping data points.

**4. Heatmap:**

* **Purpose:** Visualizes the pairwise correlations or distances between data points in the t-SNE space using a color-coded matrix. Darker colors indicate stronger correlations or smaller distances, while lighter colors represent weaker relationships.
* **Benefits:**
  + Explores relationships between data points within the t-SNE embedding.
  + Helps identify potential patterns or groups of highly correlated data points.

The best choice for you depends on what you want to achieve after the t-SNE visualization. Here's a quick guide:

* **KMeans/DBSCAN:** Identify clusters within the t-SNE embedding.
* **Silhouette Plot:** Evaluate the quality of clustering results (after KMeans or DBSCAN).
* **Heatmap:** Explore pairwise relationships/distances between data points in t-SNE.

this is not an exhaustive list

**NN Widget, Solver:**

By default, the NN widget in Orange typically uses the **L-BFGS *Solver***

* **L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) algorithm as the solver for training the neural network. L-**BFGS is an efficient optimization algorithm well-suited for moderate-sized datasets and problems with smooth gradients.

However, Orange might also offer options for other solvers like **SGD (Stochastic Gradient Descent)** and **Adam (Adaptive Moment Estimation)** within the NN widget settings. These can be advantageous in certain scenarios:

* **SGD:**
  + Simpler and computationally cheaper than L-BFGS.
  + Works well for large datasets where processing all data points at once becomes impractical.
  + May require more iterations to converge compared to L-BFGS.
* **Adam:**
  + Combines the benefits of SGD and RMSProp (another optimization algorithm).
  + Adaptively adjusts learning rates for each parameter, potentially leading to faster convergence and better performance in some cases.

**2. Number of Iterations in NN Widget:**

The NN widget usually provides a setting for the **maximum number of iterations** allowed for training the neural network. This parameter controls how long the optimization algorithm will run before stopping.

* **Choosing the Right Number of Iterations:**
  + Too few iterations might lead to underfitting (model fails to learn the underlying patterns).
  + Too many iterations can result in overfitting (model memorizes training data and performs poorly on unseen data).
  + It's often a good practice to monitor the training process (e.g., using a validation set) and adjust the number of iterations based on when the model starts to overfit.

**3. Stochastic Gradient Descent (SGD) Widget:**

The **Stoch. Grad. Disc.** widget (Stochastic Gradient Descent) in Orange *is a separate widget* that allows you to implement SGD explicitly in your workflow. *Here's how it differs from the NN widget's internal solver*:

* **Explicit Control:** The Stoch. Grad. Disc. widget gives you more granular control over the SGD hyperparameters, such as learning rate, momentum, and batch size. You can fine-tune these parameters for specific problems.
* **Flexibility:** This widget might provide additional functionalities beyond basic classification or regression tasks offered by the NN widget.

**Connecting SGD Widget with NN:**

* You might consider connecting the Stoch. Grad. Disc. widget with the NN widget if you need more control over SGD parameters or require functionalities beyond the NN widget's capabilities.
* However, if the NN widget's default L-BFGS solver or other built-in options like SGD or Adam with their default settings meet your needs, you can directly use the NN widget without needing the Stoch. Grad. Disc. widget.

**Choosing the Right Approach:**

* For most common neural network tasks in Orange, the NN widget with its built-in solvers is likely sufficient.
* Consider using the Stoch. Grad. Disc. widget if you require finer control over SGD or need features not available in the NN widget.
* Experiment with both approaches to see which one works best for your specific problem and data.

**Exploring Orange's Test & Score Widget and Related Concepts:**

The Test & Score widget in Orange is a valuable tool for evaluating the performance of machine learning models. Let's delve into your questions:

**1. Repeat Train/Test:**

This option specifies how many times the training and testing process should be repeated. In each repetition:

* The data is randomly **split** into training and testing sets based on the specified training test size (e.g., 70% for training, 30% for testing).
* The model is **trained** using the training set.
* The trained model is **evaluated** on the testing set.

**2. Example with Repeat Train/Test = 2:**

* With 1000 initial examples and Repeat Train/Test = 2, the widget performs the following:
  + **First Repetition:**
    - Splits data randomly into 700 training examples (70%) and 300 testing examples (30%).
    - Trains the model using the 700 training examples.
    - Evaluates the model on the **300 testing examples** (not 1000). You'll see results based on this evaluation in the Confusion Matrix.
  + **Second Repetition:**
    - Performs another random split (potentially different from the first split).
    - Trains and evaluates the model again using the new splits.
  + The Confusion Matrix might show **around 600 examples** (300 from each repetition) if both repetitions used exactly the same split (unlikely). Otherwise, it might show slightly less depending on how many examples overlapped in the training sets from each repetition.

**The following metrics data in Test and Scoring widget:**Train Test AUC CA F1 Prec Recall MCC Spec LogLoss

0.034 0.02 0.98 0.96 0.96 0.96 0.96 0.94 0.98 0.82

1. **Train:**
   * This refers to the performance of the model (kNN or other) on the training dataset. It's a measure of how well the model fits the training data.
   * In your case, the value is 0.034, indicating that the model performed well on the training set.
2. **Test:**
   * This refers to the performance of the model on a separate testing dataset (not used during training). It gives an indication of how well the model generalizes to new, unseen data.
   * The value of 0.02 suggests that the model performed well.
3. **AUC (Area Under the ROC Curve):**
   * AUC is a measure of the model's ability to distinguish between positive and negative classes. It ranges from 0 to 1, where a higher value indicates better performance.
   * Your AUC value of 0.98 is quite high, suggesting that the model has a strong discriminatory power.
4. **CA (Classification Accuracy):**
   * This is the ratio of correctly predicted instances to the total instances. It provides a general measure of model accuracy.
   * A value of 0.96 indicates that the model correctly classified 96% of *instances in both the training and testing datasets*.
5. **F1 Score:**
   * The F1 score is the *harmonic mean of precision and recall*. It provides a balance between precision and recall.

F1 = 2 \* (P \* R) / (P + R).

* + Your F1 score of 0.96 is high, indicating a good balance between precision and recall.

1. **Precision:**
   * Precision measures the accuracy of positive predictions. It is the ratio of true positive predictions to the total positive predictions.
   * A precision value of 0.96 suggests that the model has a high accuracy in predicting positive instances.
2. **Recall:**

Recall represents the ratio of correctly predicted positive cases out of all the actual positive cases in the dataset.

It answers the question: "Of all the actual positive cases in the data, how many did the model correctly predict as positive?"

***Explanations***: **Precision and Recall** are two crucial metrics in machine learning, especially for classification tasks, and their definitions can seem very similar.

**Precision:**

* **Focus:** Accuracy of positive predictions.
* **Calculation:** Ratio of true positive predictions (TP) to the total positive predictions (TP + FP).
* **Interpretation:**
  + A high precision value (closer to 1) indicates that the model is mostly predicting positive cases that are actually positive. There are few false positives (FP).
  + A low precision value (closer to 0) means the model is predicting many false positives. It might be "overly enthusiastic" about classifying things as positive.

**Recall:**

* **Focus:** Completeness of positive predictions.
* **Calculation:** Ratio of true positive predictions (TP) to all actual positive cases in the dataset (TP + FN).
* **Interpretation:**
  + A high recall value (closer to 1) indicates the model is catching most of the actual positive cases. There are few false negatives (FN).
  + A low recall value (closer to 0) means the model is missing many actual positive cases. It might be too cautious in classifying things as positive.

**Understanding the Trade-off:**

Precision and Recall often have a trade-off relationship. Here's why:

* **Increasing Precision:** If you adjust your model to be more precise (fewer false positives), you might inadvertently decrease recall (more false negatives). The model becomes stricter about classifying things as positive.
* **Increasing Recall:** If you adjust your model to have higher recall (fewer false negatives), you might see a decrease in precision (more false positives). The model becomes more lenient about classifying things as positive.

**Choosing the Right Metric:**

The importance of Precision vs. Recall depends on the specific problem you're trying to solve. Here are some examples:

* **Spam Filtering:** High precision might be desirable here. A false positive (mistakenly labeling a non-spam email as spam) is less harmful than a false negative (missing an actual spam email).
* **Fraud Detection:** High recall might be critical. Missing a fraudulent transaction (false negative) could be very costly, while a false positive (mistakenly flagging a legitimate transaction) can be inconvenient but rectified.

**Illustrative Example:**

Imagine a medical test for a disease. Here's how Precision and Recall would be interpreted:

* **Precision:** The percentage of people who test positive and actually have the disease.
* **Recall:** The percentage of people with the disease who are correctly identified by the test.

**In conclusion:**

Understanding the distinction between Precision and Recall is essential for evaluating the performance of your machine learning models in classification tasks. By considering the trade-off between these metrics and the specific problem you're addressing, you can choose the metric that best reflects the success of your model.

**SUMMARY**

1. **True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN)**: These are the four basic elements of a confusion matrix:
   * True Positives (TP): The number of instances correctly predicted as belonging to the positive class.
   * True Negatives (TN): The number of instances correctly predicted as not belonging to the positive class.
   * False Positives (FP): Also known as Type I errors, these are instances incorrectly predicted as belonging to the positive class when they actually belong to the negative class.
   * False Negatives (FN): Also known as Type II errors, these are instances incorrectly predicted as not belonging to the positive class when they actually belong to the positive class.
2. **Interpretation**: The ***confusion matrix*** allows you to gain insights into the performance of your classifier. By examining the values in the matrix, you can assess how well your model is performing in terms of correctly and incorrectly classifying instances across different classes.
3. **Metrics Derivation**: Various evaluation metrics can be derived from the confusion matrix, including:
   * Accuracy: (TP + TN) / Total
   * Precision: TP / (TP + FP)
   * Recall (*Sensitivity* or True Positive Rate): TP / (TP + FN)
   * Specificity (True Negative Rate): TN / (TN + FP)
   * F1 Score: 2 \* (Precision \* Recall) / (Precision + Recall)
4. **MCC (Matthews Correlation Coefficient):**
   * MCC is a correlation coefficient between the observed and predicted binary classifications. It considers true and false positives and negatives.
   * A value of 0.94 indicates a strong correlation between predicted and observed classifications.
5. **Spec (Specificity):** Spec = TN / (TN + FP)
   * Specificity measures the ability of the model to correctly identify negative instances.
   * A value of 0.98 suggests that the model has a high specificity, meaning it is good at identifying true negatives.
6. **LogLoss:** LogLoss = - (y \* log(p) + (1 - y) \* log(1 - p))

y: True label of the data point (0 for negative, 1 for positive)

p: Predicted probability of the data point belonging to the positive class (between 0 and 1)

* + LogLoss is a measure of how well the predicted probabilities match the true labels. Lower values are better.
  + Your LogLoss value of 0.82 is reasonable, and lower values would indicate better-calibrated probability predictions.

**MNIST Case, where we have 0-...- 9 outputs/classes. How are all those metrics intrepreted for MNIST case.:**

1. **True Positives (TP):**
   * In a multi-class setting like MNIST, TP refers to the instances where the model correctly predicts a specific digit among the 10 possible digits.
2. **False Negatives (FN):**
   * FN occurs when the model fails to predict a certain digit, which is actually present in the true labels.
3. **True Negatives (TN):**
   * In multi-class problems, TN is less applicable because it represents instances that are correctly predicted as not belonging to a specific class. It's more commonly used in binary classification.
4. **False Positives (FP):**
   * Similarly, FP in a multi-class context refers to instances that are incorrectly predicted as belonging to a specific class when they do not.

Now, let's discuss the other metrics:

1. **Precision:**
   * In the MNIST case, precision for a specific digit measures how many instances predicted as that digit are actually that digit. It's the ratio of true positives to the sum of true positives and false positives for that digit.
2. **Recall (Sensitivity):**
   * Recall for a specific digit measures how many instances of that digit were correctly predicted out of all instances that are actually that digit. It's the ratio of true positives to the sum of true positives and false negatives for that digit.
3. **F1 Score:**
   * The F1 score is the harmonic mean of precision and recall for a specific digit. It provides a balanced measure of how well the model performs for that particular digit.
4. **Accuracy:**
   * Accuracy in a multi-class setting is the overall correctness of predictions across all classes. It's the ratio of correct predictions (true positives and true negatives for all classes) to the total number of instances.
5. **Macro/Micro-Average:**
   * When dealing with multiple classes, you might see macro and micro-average metrics. Micro-average considers all instances and classes equally, while macro-average calculates metrics for each class independently and then takes the average.
6. **AUC (Area Under the ROC Curve):**

* In a multi-class context, AUC is often calculated using the one-vs-rest (OvR) approach. It measures the ability of the model to distinguish between one class and the rest.

1. **LogLoss:**

* LogLoss is a measure of how well the predicted probabilities match the true labels across all classes. Lower values indicate better-calibrated probability predictions.

In summary, when evaluating a multi-class classification model like the one for the MNIST dataset, it's important to consider precision, recall, F1 score, accuracy, and potentially AUC and LogLoss, with specific attention to each digit's performance in addition to overall model performance.

In the Test & Score widget in Orange, the options **"Stratified" and "Replicate**" have the following meanings:

1. **Stratified**: When this option is selected, the data is divided into folds while ensuring that each fold maintains the same class distribution as the original dataset. This is particularly useful for classification tasks to ensure that each class is represented proportionally in both the training and testing sets.
2. **Replicate**: This option allows you to specify the number of times you want to repeat the evaluation process. Each replication involves randomly splitting the data into training and testing sets, performing the specified evaluation metric, and then averaging the results across all replications. Replicating the evaluation process helps to obtain more robust and reliable performance estimates, especially when dealing with small datasets or when evaluating the performance of complex models.

By utilizing these options appropriately, you can ensure more reliable and informative evaluations of your machine learning models in Orange.

**Confusion Matrix in Multi-Class Classification**

A confusion matrix is a powerful tool used to evaluate the performance of a classification model, especially when dealing with multiple classes. It provides a clear visualization of how the model classified the data points and allows you to identify areas for improvement.

**Structure:**

* The confusion matrix is a square table with rows representing the **actual classes** of the data points and columns representing the **predicted classes** by the model.
* Each cell of the table contains the number of data points that belong to a particular **actual class** (row) but were **predicted** by the model as belonging to a different **predicted class** (column).

**Interpreting the Matrix:**

* **Perfect Classification:** If the model perfectly classified all data points, the diagonal elements (where the actual class and predicted class match) would contain all the values, and other cells would be zero.
* **Misclassifications:** Off-diagonal elements represent **misclassifications**. The higher the value in a non-diagonal cell, the more data points were incorrectly classified into that category.

**Key Metrics Derived from Confusion Matrix:**

* **Accuracy:** Overall percentage of correctly classified data points.
* **Precision:** Ratio of correctly predicted positive cases to the total predicted positive cases.

(For class i: Precision\_i = TP\_i / (TP\_i + FP\_i))

* **Recall:** Ratio of correctly predicted positive cases to the total actual positive cases. (For class i: Recall\_i = TP\_i / (TP\_i + FN\_i))
* **F1-Score:** Harmonic mean of precision and recall, combining their importance.

(For class i: F1\_i = 2 \* (Precision\_i \* Recall\_i) / (Precision\_i + Recall\_i))

**Confusion Matrix Example (3 Classes):**

|  |  |  |  |
| --- | --- | --- | --- |
| Predicted Class | Class A | Class B | Class C |
| **Class A** | **TP\_A** (Correct) | FP\_B | FP\_C |
| **Class B** | FN\_A | **TP\_B** (Correct) | FP\_C |
| **Class C** | FN\_A | FN\_B | **TP\_C** (Correct) |

drive\_spreadsheetЭкспортировать в Таблицы

**Benefits of Confusion Matrix:**

* **Visualization:** Provides a clear visual representation of classification performance.
* **Identification of Errors:** Helps pinpoint specific classes where the model struggles, allowing for targeted improvement strategies.
* **Evaluation of Multi-Class Models:** More informative than just accuracy for multi-class problems, as it reveals issues in specific class predictions.

**Additional Considerations:**

* When dealing with imbalanced datasets (unequal distribution of classes), accuracy alone might not be sufficient. Analyzing metrics like precision, recall, and F1-score for each class is crucial.
* Confusion matrix interpretation should be done in conjunction with other evaluation metrics to gain a comprehensive understanding of the model's performance.

By utilizing the confusion matrix and the derived metrics, you can effectively assess the strengths and weaknesses of your multi-class classification model and take steps to improve its performance.

1. **Basic Structure**: A confusion matrix is essentially a table where each row represents the instances in an actual class while each column represents the instances in a predicted class (or vice versa). The diagonal elements of the matrix represent the number of instances that were correctly classified, while off-diagonal elements represent misclassifications.
2. **True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN)**: These are the four basic elements of a confusion matrix:
   * True Positives (TP): The number of instances correctly predicted as belonging to the positive class.
   * True Negatives (TN): The number of instances correctly predicted as not belonging to the positive class.
   * False Positives (FP): Also known as Type I errors, these are instances incorrectly predicted as belonging to the positive class when they actually belong to the negative class.
   * False Negatives (FN): Also known as Type II errors, these are instances incorrectly predicted as not belonging to the positive class when they actually belong to the positive class.
3. **Interpretation**: The confusion matrix allows you to gain insights into the performance of your classifier. By examining the values in the matrix, you can assess how well your model is performing in terms of correctly and incorrectly classifying instances across different classes.
4. **Metrics Derivation**: Various evaluation metrics can be derived from the confusion matrix, including:
   * Accuracy: (TP + TN) / Total
   * Precision: TP / (TP + FP)
   * Recall (Sensitivity or True Positive Rate): TP / (TP + FN)
   * Specificity (True Negative Rate): TN / (TN + FP)
   * F1 Score: 2 \* (Precision \* Recall) / (Precision + Recall)
5. **Visualization**: Confusion matrices can also be visualized graphically, making it easier to interpret the results, especially when dealing with multiple classes.

In summary, a confusion matrix is a crucial tool for evaluating the performance of a classification model, especially in scenarios involving multiple classes, by providing detailed insights into the model's predictive capabilities across different categories.

**3. ROC Analysis:**

The Receiver Operating Characteristic (ROC) Analysis widget visualizes the performance of a classification model using the ROC curve. The ROC curve plots the True Positive Rate (TPR) on the y-axis against the False Positive Rate (FPR) on the x-axis. It helps assess how well the model distinguishes between positive and negative classes across different classification thresholds.

**4. Performance Curve:**

This term might refer to different visualizations depending on the task (classification vs. regression).

* **Classification:** In Orange, the Performance Curve could represent the **lift curve** for a classification model. The lift curve shows the cumulative ratio of positive examples identified by the model compared to a random selection.
* **Regression:** For regression tasks, it might refer to a curve depicting the **model's predicted values versus the actual target values**.

**5. Calibration Plot:**

The Calibration Plot assesses how well the predicted probabilities of a classification model align with the actual class frequencies. Ideally, the predicted probability should closely match the true probability of an example belonging to a specific class.

**6. Recommendations for Using These Notions in Orange:**

* **Set Repeat Train/Test:** Use a reasonable number of repetitions (e.g., 5-10) to account for randomness in data splits and obtain a more robust evaluation:

[After N repetitions, we calculate the mean (average) of each metric across all N repetitions. This provides a more reliable estimate of the model's true performance, taking into account the randomness in data splits].

* **Analyze Confusion Matrix:** Look at the confusion matrix to understand the model's performance in terms of true positives, false positives, true negatives, and false negatives.
* **Interpret ROC Curve:** A high Area Under the ROC Curve (AUC) indicates good model performance at distinguishing classes. Use ROC analysis to compare different models or to fine-tune hyperparameters.
* **Evaluate Performance Curve:** For classification, a lift curve that rises quickly above the baseline indicates a good model for identifying positive examples. For regression, a plot where predicted values closely follow the actual values suggests good model performance.
* **Inspect Calibration Plot:** A well-calibrated model shows a diagonal line in the calibration plot, indicating that the predicted probabilities accurately reflect the true class probabilities.

**Additional:**

* Use these widgets along with other evaluation metrics like accuracy, precision, recall, and F1-score for a comprehensive understanding of model performance.
* Consider using cross-validation techniques to obtain a more reliable estimate of model generalizability.