Building and Evaluating Machine Learning Models in Orange

A Step-by-Step Guide for Classification Tasks

Machine Learning with Orange

1 Introduction

This guide will walk you through the complete process of building, evaluating, and comparing classification models using Orange. You'll learn how to create a machine learning workflow that transforms raw data into actionable predictions while understanding each step's importance.

Orange uses a visual programming approach where you connect widgets (components) to create data analysis workflows. Think of widgets as building blocks and connections as the flow of data between them. This visual approach makes it easier to understand and experiment with different machine learning pipelines.

Dataset Overview: You'll work with a retail campaign dataset containing information about customers and whether they responded to a marketing campaign. Your goal is to predict which customers will respond to future campaigns.

2 Step 1: Loading and Understanding Your Data

2.1 Loading the Dataset

1. Open Orange and create a new project by clicking **File** → **New**.

2. Find the **File** widget in the Data section (left panel). Drag it onto the canvas.

3. Double-click the File widget to open it. Click the folder icon and navigate to your `retail_campaign_dataset.xlsx` file.

4. Once loaded, you'll see a summary showing:

   - Number of instances (rows): 200 customers
   - Number of features (columns): 12
   - Target variable: CampaignResponse

Pro Tip: Orange automatically detects your target variable (the one you want to predict). Look for the target icon next to CampaignResponse. If it's not selected as target, click on its role and change it to "target".

2.2 Initial Data Exploration

Now let's understand what we're working with:

1. Add a **Data Table** widget (from Data section) and connect the File widget to it by:

   - Click and hold on the File widget's right side
   - Drag the connection line to the Data Table widget's left side
   - Release to create the connection

2. Double-click Data Table to see your data in spreadsheet format. Notice:

   - Numerical features: Age, Income_K, AvgTransactionValue, etc.
   - Categorical features: LoyaltyMember, PreferredChannel, etc.
   - Target variable: CampaignResponse (Yes/No)

3. Add a **Feature Statistics** widget and connect File → Feature Statistics. This shows:

   - Distribution of each feature

- Missing values (we have none - the data is clean!)
- Basic statistics (mean, std dev, min, max)

3 Step 2: Understanding Feature Relationships

Before building models, let's explore which features might be good predictors:

3.1 Visualizing Distributions

1. Add a **Distributions** widget and connect File → Distributions.
2. In Distributions, select different features to see how they relate to CampaignResponse:
   - Try **LoyaltyMember**: Notice how "Yes" members have higher response rates
   - Try **DaysSinceLastPurchase**: Recent purchasers respond more
   - Try **CustomerSegment**: VIP customers show highest response rates

What to Look For: Features where the distribution differs significantly between "Yes" and "No" responses are likely good predictors. These visual patterns help you understand what your models will learn.

3.2 Examining Correlations

1. Add a **Correlations** widget and connect File → Correlations.
2. Look for features highly correlated with CampaignResponse. You'll see:
   - Positive correlations: Features that increase response likelihood
   - Negative correlations: Features that decrease response likelihood

4 Step 3: Preparing for Model Building

4.1 Setting Up Train-Test Split

Machine learning models need to be tested on data they haven't seen during training. This gives us an honest assessment of how well they'll perform on new customers.

1. Add a **Data Sampler** widget and connect File → Data Sampler.
2. Configure Data Sampler:
   - Sampling type: **Random Sampling**
   - Sample size: **70%** (this will be training data)
   - Check **Sample with replacement**: No
   - Check **Stratified sampling**: Yes (maintains response rate proportions)
   - Set **Random seed**: 42 (for reproducibility)
3. The Data Sampler has two outputs:
   - **Data Sample**: 70% for training (140 customers)
   - **Remaining Data**: 30% for testing (60 customers)

Important: Always use stratified sampling for imbalanced datasets. This ensures both training and test sets have similar proportions of Yes/No responses.

5 Step 4: Building Classification Models

Now we'll build five different classification models and compare their performance. Each model has different strengths and assumptions.

5.1 Creating the Model Workflow

1. From the Model section, add these five widgets to your canvas:
   - **SVM** (Support Vector Machine)
   - **Logistic Regression**
   - **Tree** (Decision Tree)
   - **Random Forest**
   - **kNN** (k-Nearest Neighbors)
2. Connect the **Data Sample** output from Data Sampler to each model widget.
3. Configure each model:

5.1.1 SVM Configuration

- Double-click SVM widget
- Cost (C): 1.0 (controls regularization)
- Kernel: RBF (handles non-linear relationships)
- Leave other settings as default

5.1.2 Logistic Regression Configuration

- Double-click Logistic Regression widget
- Regularization type: Ridge (L2)
- Strength (C): 1.0
- Leave other settings as default

5.1.3 Decision Tree Configuration

- Double-click Tree widget
- Max depth: 5 (prevents overfitting)
- Min samples split: 5
- Min samples leaf: 2

5.1.4 Random Forest Configuration

- Double-click Random Forest widget
- Number of trees: 100
- Max features: sqrt (square root of total features)
- Leave other settings as default

5.1.5 kNN Configuration

- Double-click kNN widget
- Number of neighbors: 5
- Weight: Uniform
- Metric: Euclidean

6 Step 5: Evaluating Model Performance

6.1 Using Test and Score Widget

The Test and Score widget is your central hub for model evaluation. It runs multiple models and compares their performance.

1. Add a **Test and Score** widget to your canvas.
2. Connect all five model widgets to Test and Score (they go to the left side).
3. Connect the **Remaining Data** output from Data Sampler to Test and Score's **Test Data** input.
4. Double-click Test and Score to see results. You'll see a table with:
    - **AUC**: Area Under the ROC Curve (higher is better, max 1.0)
    - **CA**: Classification Accuracy (percentage correctly classified)
    - **F1**: Harmonic mean of precision and recall
    - **Precision**: Of predicted "Yes", how many were actually "Yes"
    - **Recall**: Of actual "Yes", how many did we predict correctly

Understanding Metrics:

- **Use Accuracy** when classes are balanced and errors are equally costly
- **Use Precision** when false positives are expensive (e.g., spam detection)
- **Use Recall** when false negatives are expensive (e.g., disease detection)
- **Use F1** when you need balance between precision and recall
- **Use AUC** for overall model quality regardless of threshold

6.2 Cross-Validation for Robust Evaluation

Test set evaluation gives one estimate, but cross-validation provides more reliable performance estimates:

1. In Test and Score, click on the **Test on train data** section.
2. Change from "Test on train data" to **Cross-validation**.
3. Set:
   - Number of folds: 10
   - Stratified: Yes
   - Random seed: 42
4. Now you'll see performance metrics with standard deviations, showing how consistent each model is.

7 Step 6: Detailed Model Analysis

7.1 Confusion Matrix Analysis

Understanding where models make mistakes is crucial for improvement:

1. Add a **Confusion Matrix** widget.
2. Connect Test and Score → Confusion Matrix.
3. The confusion matrix shows:
   - True Positives (TP): Correctly predicted "Yes"
   - True Negatives (TN): Correctly predicted "No"
   - False Positives (FP): Predicted "Yes" but was "No"
   - False Negatives (FN): Predicted "No" but was "Yes"
4. Select different models in Test and Score to see their confusion matrices.

Business Context: In marketing, false positives (targeting non-responders) waste money on campaigns. False negatives (missing potential responders) mean lost sales. Consider which is more costly for your business.

7.2 ROC Curve Analysis

ROC curves show model performance across all possible thresholds:

1. Add a **ROC Analysis** widget.
2. Connect Test and Score → ROC Analysis.
3. The ROC curve plots:
   - X-axis: False Positive Rate (1 - Specificity)
   - Y-axis: True Positive Rate (Sensitivity/Recall)
   - Diagonal line: Random classifier performance
   - Area Under Curve (AUC): Overall performance measure
4. Models with curves closer to the top-left corner perform better.

7.3 Understanding Feature Importance

Different models show feature importance differently:

7.3.1 For Tree-Based Models

1. Add a **Tree Viewer** widget.
2. Connect Tree → Tree Viewer.
3. The tree shows which features make the most important splits.

7.3.2 For Logistic Regression

1. Add a **Nomogram** widget.
2. Connect Logistic Regression → Nomogram.
3. Features are ordered by importance (coefficient magnitude).

8 Step 7: Making Predictions on New Data

8.1 Using the Predictions Widget

Once you've selected your best model, you can use it to make predictions:

1. Add a **Predictions** widget.
2. Connect:
   - Your best model (e.g., Random Forest) → Predictions
   - File → Predictions (as new data to predict)
3. Predictions widget shows:
   - Original features
   - Predicted class (Yes/No)
   - Prediction probabilities
4. You can select customers with high probability of responding for targeted campaigns.

9 Step 8: Model Selection and Deployment Considerations

9.1 Choosing the Best Model

Consider these factors when selecting your final model:

1. **Performance Metrics**: Which model has the best scores for your business needs?
2. **Interpretability**:
   - Decision Trees: Highly interpretable
   - Logistic Regression: Interpretable coefficients
   - Random Forest, SVM, kNN: Black box models
3. **Prediction Speed**:
   - Logistic Regression, Decision Tree: Very fast
   - SVM, Random Forest: Moderate
   - kNN: Slow for large datasets
4. **Handling New Data Types**:
   - Tree-based models: Handle mixed data well
   - SVM, kNN: Require scaling for numerical features

Deployment Warning: A model that's 2% more accurate but takes 100x longer to make predictions might not be worth it in production. Balance accuracy with practical constraints.

10 Advanced Tips and Best Practices

10.1 Feature Engineering Ideas

While our dataset is pre-cleaned, in practice you might create new features:

- Customer lifetime value = AvgTransactionValue × PreviousPurchases
- Recency score = 1 / (DaysSinceLastPurchase + 1)
- Engagement score = Combination of email opens and app usage

11 Summary Workflow Checklist

Your complete Orange workflow should include:

✓ Data loading and initial exploration

✓ Train-test split with stratification

✓ Multiple model training

✓ Comprehensive evaluation (accuracy, ROC, confusion matrix)

✓ Feature importance analysis

✓ Final model selection based on business needs

Final Advice: Save your Orange workflow frequently (File → Save). You can share workflows with classmates and reuse them for similar problems. Each widget's settings are preserved, making it easy to reproduce results.