Model Performance & Residual Analysis

July 5, 2025

Outline

Why Measure Model Performance?

- **Accuracy Assessment**: Quantify how well models predict outcomes for unseen data
- **Model Comparison**: Select the best-performing model among multiple candidates
- **Bias-Variance Analysis**: Detect overfitting (high variance) or underfitting (high bias)
- **Business Impact**: Translate metrics into actionable insights

Key Concepts

- **Evaluation Metrics**: Classification vs. Regression metrics
- **Data Splitting**: Training, validation, testing datasets
- **Cross-Validation**: Estimate performance across multiple splits
- **Real-World Validation**: Simulate practical scenarios

Classification Metrics I

Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Proportion of correctly classified samples

Precision

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

True positives among predicted positives

Recall (Sensitivity)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

True positives among all actual positives

F1-Score

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Harmonic mean of precision and recall

Classification Metrics II

AUC-ROC

$$\int_0^1 TPR(FPR^{-1}(x))dx$$

Measures model's ability to distinguish between classes

Log Loss (Cross-Entropy)

$$-\frac{1}{n}\sum_{i=1}^{n}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

Penalizes confident but incorrect predictions

Brier Score

$$\frac{1}{n}\sum_{i=1}^{n}(p_i - y_i)^2$$

Evaluates accuracy of predicted probabilities

When to use each metric:

- Imbalanced classes → Precision, Recall, F1
- Ranking performance → AUC-ROC
- Probability calibration → Log Loss, Brier Score

Regression Metrics

Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Average squared difference between predictions and actuals

Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

Average absolute difference between predictions and actuals

R-Squared (Coefficient of Determination)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Proportion of variance explained by the model

When to use each metric:

- Sensitive to outliers $\rightarrow$ MAE
- Penalize large errors $\rightarrow$ MSE
- Compare models $\rightarrow$ R-Squared

Challenges in Measuring Model Performance

Data Imbalance

- Metrics like accuracy can be misleading
- Better metrics: precision, recall, F1-score

Overfitting/Underfitting

- Overfitting: Model performs well on training data but poorly on unseen data
- Underfitting: Insufficient learning of data patterns

Real-World Applicability

- Lab performance $\neq$ Real-world performance
- Need for temporal validation or external datasets

Heteroscedasticity

- Error variance changes across feature values
- May indicate need for transformations

Performance Evaluation: Model Setup & Training

```
# Regression tasks using lightGBM and xgboost
from modeva.models import MoLGBMRegressor, MoXGBRegressor
# for lightGBM
model_lgbm = MoLGBMRegressor(name = "LGBM_model",
                             max_depth=2,
                             n_estimators=100)
# for xgboost
model_xgb = MoXGBRegressor(name = "XGB_model",
                           max_depth=2,
                           n_estimators=100)

# Train models with input: ds.train_x and target: ds.train_y
model_lgbm.fit(ds.train_x, ds.train_y)
model_xgb.fit(ds.train_x, ds.train_y)
```

Performance Evaluation in ModEva: Reporting

```python
# Create a testsuite that bundles dataset and model
from modeva import TestSuite
ts = TestSuite(ds, model_lgbm) # store bundle of dataset and model

# Evaluate performance and summarize into table
results = ts.diagnose_accuracy_table(
    train_dataset="train",
    test_dataset="test",
    metric=("MAE", "MSE", "R2")
)
results.table  # Display results in a tabular format

```

This produces a table with metrics for both training and test sets

Performance Comparison Between Models

```
# Create TestSuite with multiple models
tsc = TestSuite(ds, models=[model_lgbm, model_xgb])

# Performance comparison of 2 models
results = tsc.compare_accuracy_table(
    train_dataset="train",
    test_dataset="test",
    metric=("MAE", "MSE", "R2")
)
results.plot()  # Visualize the comparison

```

This generates comparative visualizations to help identify the better performing model

What is Residual Analysis?

Definition

Residual analysis is a diagnostic tool to evaluate model performance by analyzing differences between actual and predicted values.

For regression: $r_i = y_i - \hat{y}_i$
For classification: Residuals are expressed as misclassification errors or differences between predicted probabilities and true labels.

Purpose of Residual Analysis:

- Assess model fit and identify systematic errors
- Examine error distribution patterns
- Detect outliers where model performs poorly
- Identify heteroscedasticity (non-constant variance)
- Validate model assumptions

Techniques for Residual Analysis

Residual Plots

- Residuals vs. Predicted: Should show random scatter around zero
- Residuals vs. Features: Identifies feature-specific weaknesses
- Histogram of Residuals: Checks distribution pattern

Quantile-Quantile (Q-Q) Plot

- Assesses if residuals follow normal distribution
- Compares quantiles of residuals to a theoretical normal distribution

Interpreting Results

- Random distribution $\rightarrow$ Model captures data well
- Systematic patterns $\rightarrow$ Model misspecification
- High variance $\rightarrow$ Heteroscedasticity
- Large residuals $\rightarrow$ Outliers or edge cases

Applications

- Model debugging
- Feature engineering
- Outlier detection
- Assumption validation

Basic Residual Analysis

```
# Create a TestSuite as before
ts = TestSuite(ds, model_lgbm)

# Perform residual analysis with feature "hr"
results = ts.diagnose_residual_analysis(
    features="hr",
    dataset="train"
)
results.plot()  # Generate residual visualization
```

This produces visualization showing residuals vs. feature "hr"

- Identify patterns in residuals across different hours of the day
- Check if the model performs consistently throughout the day
- Detect time periods with higher error rates

Advanced Residual Analysis: Supervised Approach

Concept

MoDeVa uses supervised ML to analyze residuals, enabling targeted model improvements.

- **Traditional Approach Limitations:**
  - Binning and clustering often miss complex non-linear relationships
  - Difficult to identify why errors occur in high-dimensional spaces
- **ModEva's Approach:**
  - Uses interpretable GBDT models to explicitly model residual errors
  - Employs Random Forest proximity matrices for similarity-based clustering
  - Identifies high-error regions in feature space
  - Provides actionable insights for targeted model improvements

Methodology 1: Residual Modeling with Interpretable GBDT

1. Train interpretable XGBoost model (depth-1 or depth-2) to predict residuals
2. Use absolute residuals $r = |y - \hat{y}|$ as target variable
3. Extract feature importance and effects from the residual model

```
# Train interpretable GBDT model to analyze residuals
results = ts.diagnose_residual_interpret(
    dataset='test', n_estimators=100, max_depth=2
)
results.plot("feature_importance")  # Plot feature
    importance
results.plot("effect_importance")   # Plot effect importance

# Get effect of specific feature on residuals
ts_residual = results.value["TestSuite"]
ts_residual.interpret_effects("hr", dataset="test").plot()
```

Methodology 2: Proximity-Based Clustering

1. Train Random Forest model on dataset
2. Extract proximity matrix measuring similarity between data points
3. Cluster similar samples based on this proximity
4. Analyze how errors distribute across clusters

```
# Cluster based on RF proximity matrix
results = ts.diagnose_residual_cluster(
    dataset="test",
    response_type="abs_residual",
    metric="MAE",
    n_clusters=10,
    cluster_method="pam",
    sample_size=2000,
    rf_n_estimators=100,
    rf_max_depth=5
)
results.table  # Table of cluster performance
results.plot("cluster_residual")
results.plot("cluster_performance")
results.plot("feature_importance")
```

Identifying & Interpreting High-Error Regions

1. Identify clusters with highest average error
2. Analyze feature composition of problematic clusters
3. Use data drift analysis to compare high-error regions to overall distribution
4. Develop targeted interventions to improve model performance

```
# Analyze a specific high-error cluster
cluster_id = 2  # Cluster with high error rate

# Compare cluster distribution to overall distribution
data_results = ds.data_drift_test(
    **results.value["clusters"][cluster_id]["data_info"],
    distance_metric="PSI",
    psi_method="uniform",
    psi_bins=10
)
data_results.plot("summary")  # Overall drift summary
data_results.plot(name=('density', 'hr'))  # Feature distribution

```

Actionable Insights from Residual Analysis

Model Improvements

- **Feature Transformations:** Apply non-linear transformations to features with high residual effect
- **Feature Engineering:** Create new features for high-error regions
- **Model Architecture:** Adjust model complexity or algorithm

Data Enhancements

- **Targeted Sampling:** Collect more data in high-error regions
- **Outlier Handling:** Develop specific strategies for edge cases
- **Segmented Models:** Create specialized models for challenging subgroups

Key Takeaway

Residual analysis transforms model evaluation from a simple metric comparison to a targeted diagnostic process that directly informs model improvements.

Summary: Model Performance & Residual Analysis

1. **Performance Metrics:** Choose appropriate metrics based on problem type and business needs
2. **Residual Analysis:** Critical diagnostic tool for understanding model weaknesses
3. **MoDeVa Framework:** Provides integrated tools for comprehensive model evaluation
4. **Supervised Learning for Residuals:** Powerful approach to identify and interpret error patterns
5. **High-Error Region Identification:** Enables targeted model improvements

Next Steps

- Apply these techniques to your own models
- Combine performance metrics with residual analysis
- Explore other MoDeVa capabilities for comprehensive model validation