Weakness Detection in ML Models

Using ModEva Framework for Identifying Underperforming Regions

Machine Learning Expert

July 5, 2025

Outline

Understanding Weakness Detection

Definition

Weakness detection is the process of identifying areas in the input space where a machine learning model underperforms or makes incorrect predictions.

Characteristics of Weak Regions:

- High residual errors
- Poor prediction accuracy
- Patterns of bias or inconsistency
- Unexpected behavior on certain data subsets

Goal: Identify and understand these regions to improve model reliability, robustness, and fairness across all data segments.

Why Weakness Detection is Important

Improve Model Performance

Identify struggling areas and implement targeted improvements

Guide Data Collection

Pinpoint where additional data can enhance model performance

Enhance Trustworthiness

Address systematic biases and recurring errors to build confidence

Mitigate Risks

Detect and address weaknesses before deployment

Key Approaches to Weakness Detection

Residual Analysis

- Examine prediction errors
- Identify patterns in residuals
- Locate regions with systematic errors

Data Slicing

- Divide dataset into smaller subsets
- Evaluate performance in each slice
- Compare across slices

Feature Sensitivity

- Identify features linked to poor performance
- Analyze feature interactions
- Understand sensitivity to feature changes

Visualization

- Plot performance across feature ranges
- Visualize error distributions
- Highlight underperforming regions

Introduction to Error Slicing

Concept

Error slicing involves dividing data into segments to assess model performance in specific regions of the feature space.

Primary Binning Methods in ModEva:

1. **Uniform Binning**: Equal-sized intervals across feature range
2. **Quantile Binning**: Equal number of samples in each bin
3. **Automatic Binning**: Using tree-based methods (XGBoost) to find optimal splits
4. **User-defined Binning**: Custom-defined bin boundaries

Goal: Identify specific feature ranges or data segments where model performance drops below acceptable thresholds.

Uniform Binning

How It Works:

- Divides feature range into equal-sized intervals
- Simple and easy to interpret
- Works well with uniformly distributed features

Disadvantages:

- Can result in empty or sparse bins for skewed distributions

```python
# Analyze residual feature importance
results = ts.diagnose_residual_fi(
    method="uniform")
results.plot()

# Uniform binning numerical feature
results = ts.diagnose_slicing_accuracy(
    features=(("LIMIT_BAL", ), ("PAY_1", )),
    method="uniform",
    bins=10,
    metric="AUC",
    threshold=0.65)
results.plot()
```

Quantile Binning

How It Works:

- Divides data so each bin has equal number of samples
- Handles skewed distributions effectively
- Ensures equal representation in bins

Disadvantages:

- Bin widths may vary, complicating interpretation
- Sensitive to outliers

```python
# Quantile binning
results = ts.diagnose_slicing_accuracy(
    features="LIMIT_BAL",
    method="quantile",
    bins=10,
    metric="AUC",
    threshold=0.65)
results.plot()
```

This generates a visualization of performance across quantile bins

Automatic Binning with Tree-Based Models

How It Works:

- Uses depth-1 or depth-2 XGBoost trees
- Automatically finds optimal split points
- Splits based on relationship with target

Advantages:

- Optimized for target performance
- Captures meaningful feature-target relationship
- More intelligent than fixed-width binning

```
# Automatic binning
results = ts.diagnose_slicing_accuracy(
    features="LIMIT_BAL",
    method="auto-xgb1",
    bins=10,
    metric="AUC",
    threshold=0.75)
results.plot()  # Display results in plot
results.table   # Display results in table

```

This automatically identifies meaningful bins based on model performance

Custom Binning

```
# Custom binning
results = ts.diagnose_slicing_accuracy(
    features="LIMIT_BAL",
    method="precompute",
    bins={"LIMIT_BAL": (0.0, 50000, 1000000)},
    metric="AUC")
results.table  # Display results in table

```

This allows defining custom bin boundaries for specific feature ranges of interest

When to use custom binning:

- When specific feature thresholds are meaningful for the business context
- To focus on particular regions of interest (e.g., high-value customers)
- When domain expertise suggests particular breakpoints
- For comparing with established industry benchmarks

Multiple Feature Slicing

```python
# Slicing for a set of features
results = ts.diagnose_slicing_accuracy(
    features=(("PAY_1", ), ("BILL_AMT1",), ("PAY_AMT1", )),
    method="quantile",
    metric="AUC",
    threshold=0.6)
results.table

```

This analyzes performance across multiple features independently

Benefits:

- Provides a comprehensive view of performance across multiple feature dimensions
- Identifies which features are most associated with weak performance areas
- Allows prioritization of feature improvement efforts

Feature Interaction Slicing

```
# 2-Feature interaction slicing
results = ts.diagnose_slicing_accuracy(
    features=("PAY_1", "PAY_AMT1"),
    method="uniform",
    bins=10,
    metric="AUC",
    threshold=0.5)
results.table

```

This examines performance across combinations of feature values

Why interaction slicing matters:

- Models may perform well for individual feature ranges but struggle with specific combinations
- Reveals complex relationships that cause model weaknesses
- Identifies rare but important feature interaction scenarios
- Helps detect potential biases in specific feature combinations

Analyzing Weak Regions

```python
# Retrieving samples below threshold value
from modeva.testsuite.utils.slicing_utils import get_data_info
data_info = get_data_info(res_value=results.value)[("PAY_1", "PAY_AMT1")]
data_info

# Testing distribution difference between weak samples and the rest
data_results = ds.data_drift_test(
    **data_info,
    distance_metric="PSI",
    psi_method="uniform",
    psi_bins=10)
data_results.plot("summary")
```

This identifies data points in weak regions and compares their distribution to the overall dataset

Visualizing Weak Regions

```
# To get the list of figure names in the "data_results" object
data_results.get_figure_names()

# Example of plotting from the list of figures
data_results.plot(('density', 'PAY_1'))

```

This visualizes the distribution differences between weak samples and normal samples

Key visualizations for understanding weak regions:

- Density plots showing feature distributions
- PSI (Population Stability Index) summary plots
- Feature importance plots for weak regions
- Interaction heatmaps highlighting problematic combinations

Comparing Weaknesses Across Models

```python
# Compare models on numerical features
tsc = TestSuite(ds, models=[model_lgbm, model_xgb])
results = tsc.compare_slicing_accuracy(
    features="PAY_AMT1",
    method="quantile",
    bins=10,
    metric="AUC")
results.plot()

```

This compares performance of different models across feature slices

Benefits of comparative weakness analysis:

- Identifies which model performs better in specific regions
- Reveals complementary strengths across models
- Informs potential ensemble strategies
- Guides targeted model improvement efforts

Comparing Categorical Feature Performance

```python
# Compare models on categorical features
tsc = TestSuite(ds, models=[model_lgbm, model_xgb])
results = tsc.compare_slicing_accuracy(
    features="EDUCATION",
    metric="AUC",
    threshold=0.6)
results.plot()

```

This compares model performance across categorical feature values

What to look for:

- Categories where models show significant performance differences
- Segments where all models struggle (potential data issues)
- Categories with inconsistent performance across models
- Low-frequency categories with high performance variance

Beyond Basic Weakness Detection

Robustness Testing

- Tests model sensitivity to input noise
- Identifies regions vulnerable to small perturbations
- Evaluates stability of predictions

Reliability Testing

- Focuses on prediction uncertainty
- Identifies regions with low confidence
- Evaluates calibration of probability estimates

Resilience Testing

- Tests performance under distribution shift
- Evaluates behavior across heterogeneous data
- Measures degradation under changing conditions

Integration with Other Tests

- Combine with fairness analysis
- Link to explainability assessments
- Connect with feature importance

For details, refer to the corresponding sections in ModEva documentation

From Weakness Detection to Model Improvement

Data Strategies

- Collect more data in weak regions
- Balance representation of underperforming segments
- Engineer new features to address specific weaknesses
- Apply targeted transformations to problematic features

Model Strategies

- Adjust hyperparameters to focus on weak areas
- Create specialized models for challenging segments
- Implement higher capacity approaches such as mixture of experts

From Weakness Detection to Model Deployment

Deployment Strategies

- Implement guardrails for detecting edge cases
- Add uncertainty estimates to flag low-confidence predictions
- Create monitoring dashboards focused on weak regions
- Design fallback mechanisms for known weak spots

Business Integration

- Communicate limitations to stakeholders
- Align model capabilities with business risk tolerance
- Design workflows that accommodate model weaknesses
- Prioritize improvements based on business impact

Summary: Weakness Detection with MoDeVa

1. **Comprehensive Approach:** MoDeVaprovides multiple methods for identifying model weaknesses across feature spaces
2. **Flexible Binning:** Choose from uniform, quantile, automatic, or custom binning to effectively slice data
3. **Feature Interactions:** Identify weaknesses in specific feature value combinations
4. **Distribution Analysis:** Understand the characteristics of weak regions compared to the overall dataset
5. **Model Comparison:** Compare weakness patterns across different models to guide improvement strategies
6. **Advanced Diagnostics:** Link weakness detection to robustness, reliability, and resilience testing

Key Takeaway

Weakness detection transforms model development from a metrics-focused process to one that addresses specific underperforming regions.