Brief Explanation of Preprocessing and Analysis in FAIVOR Metrics Library

Saba Amiri

May 6, 2025

1 Preprocessing

1.1 Probability Calibration

When model outputs are not in standard probability range $[0, 1]$, we apply one of three transformations based on the observed value distribution:

1.1.1 Sigmoid Transformation (for Logits)

For outputs resembling logits (typically with values outside the range $[-1, 2]$), we apply the sigmoid function:

$$p(x) = \frac{1}{1 + e^{-x}}$$

This transforms unbounded values to the range $(0, 1)$, which is the theoretically correct transformation for logistic regression and neural network outputs.

1.1.2 Min-Max Scaling

For outputs in a consistent range $[a, b]$ where $a \neq b$, we apply min-max normalization:

$$p(x) = \frac{x - \min(x)}{\max(x) - \min(x)}$$

This preserves the relative ordering of predictions while rescaling to $[0, 1]$.

1.1.3 Clipping

As a fallback method, we simply clip values to the valid range:

$$p(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases}$$

1.2 Threshold-Based Binary Classification

Given a set of binary ground truth labels $y \in \{0,1\}$ and probability predictions $\hat{p} \in [0,1]$, we analyze performance across different decision thresholds $\tau \in [0,1]$.

1.2.1 Confusion Matrix Elements

For each threshold τ, we convert probability predictions to binary decisions:

$$\hat{y}_\tau = \begin{cases} 1 & \text{if } \hat{p} \geq \tau \\ 0 & \text{if } \hat{p} < \tau \end{cases}$$

Then we compute confusion matrix elements:

- True Positives: $TP(\tau) = \sum_{i=1}^{n} \mathbf{1}(y_i = 1 \wedge \hat{y}_{\tau,i} = 1)$
- True Negatives: $TN(\tau) = \sum_{i=1}^{n} \mathbf{1}(y_i = 0 \wedge \hat{y}_{\tau,i} = 0)$
- False Positives: $FP(\tau) = \sum_{i=1}^{n} \mathbf{1}(y_i = 0 \wedge \hat{y}_{\tau,i} = 1)$
- False Negatives: $FN(\tau) = \sum_{i=1}^{n} \mathbf{1}(y_i = 1 \wedge \hat{y}_{\tau,i} = 0)$

where $\mathbf{1}(\cdot)$ is the indicator function.

1.2.2 Performance Metrics at Each Threshold

Using the confusion matrix elements, we calculate:

- Accuracy: $\text{Acc}(\tau) = \frac{TP(\tau)+TN(\tau)}{TP(\tau)+TN(\tau)+FP(\tau)+FN(\tau)}$
- Precision (Positive Predictive Value): $\text{Prec}(\tau) = \frac{TP(\tau)}{TP(\tau)+FP(\tau)}$
- Recall (Sensitivity, True Positive Rate): $\text{Rec}(\tau) = \frac{TP(\tau)}{TP(\tau)+FN(\tau)}$
- Specificity (True Negative Rate): $\text{Spec}(\tau) = \frac{TN(\tau)}{TN(\tau)+FP(\tau)}$
- F1-Score (Harmonic mean of precision and recall): $\text{F1}(\tau) = \frac{2\cdot\text{Prec}(\tau)\cdot\text{Rec}(\tau)}{\text{Prec}(\tau)+\text{Rec}(\tau)}$
- False Positive Rate: $\text{FPR}(\tau) = \frac{FP(\tau)}{FP(\tau)+TN(\tau)} = 1 - \text{Spec}(\tau)$

Note: We implement division with zero-handling:

$$\text{SafeDivide}(a,b) = \begin{cases} \frac{a}{b} & \text{if } b \neq 0 \\ 0 & \text{if } b = 0 \end{cases}$$

1.3 ROC Curve Analysis

The Receiver Operating Characteristic (ROC) curve plots True Positive Rate against False Positive Rate at various threshold settings:

$$\text{ROC} = \{(\text{FPR}(\tau), \text{TPR}(\tau)) : \tau \in [0, 1]\}$$

where $\text{TPR}(\tau) = \text{Rec}(\tau)$.

1.3.1 Area Under the ROC Curve (AUC)

The AUC provides a single measure of classifier performance:

$$\text{AUC} = \int_0^1 \text{TPR}(FPR^{-1}(t))dt$$

Computationally, this is approximated using the trapezoidal rule:

$$\text{AUC} \approx \sum_{i=1}^{n-1} \frac{1}{2}(\text{TPR}_i + \text{TPR}_{i+1})(\text{FPR}_i - \text{FPR}_{i+1})$$

where thresholds are sorted such that $\text{FPR}_i \geq \text{FPR}_{i+1}$.

1.4 Precision-Recall Curve Analysis

The Precision-Recall (PR) curve plots Precision against Recall at various threshold settings:

$$\text{PR} = \{(\text{Rec}(\tau), \text{Prec}(\tau)) : \tau \in [0, 1]\}$$

1.4.1 Average Precision (AP)

Average Precision summarizes the PR curve:

$$\text{AP} = \sum_{i=1}^{n} (\text{Rec}_i - \text{Rec}_{i-1})\text{Prec}_i$$

where thresholds are sorted such that $\text{Rec}_i \geq \text{Rec}_{i-1}$ with $\text{Rec}_0 = 0$.

1.5 Implementation Details

We use 101 evenly spaced thresholds from 0 to 1 with a step size of 0.01:

$$\tau_i = \frac{i}{100}, \quad i \in \{0, 1, 2, ..., 100\}$$

For ROC and PR curves, we use scikit-learn's `roc_curve` and `precision_recall_curve` functions, which adaptively select thresholds based on the unique prediction values.

1.6 Combined ROC and PR Curve Analysis for Comprehensive Model Evaluation

Within the FAIVOR-ML-Validator, we combine both ROC curve and Precision-Recall curve analyses to provide a comprehensive view of model performance across decision thresholds. Both ROC and PR curves evaluate the same model but emphasize different aspects of performance. Therefore, the system calculates and presents both analyses simultaneously:

$$\text{Threshold Analysis} = \{\text{ROC}, \text{PR}, \text{Metrics}(\tau) \text{ for } \tau \in [0, 1]\}$$

The two curve analyses offer complementary perspectives.**ROC curve** evaluates the tradeoff between true positive rate and false positive rate, providing an overall measure of discrimination ability regardless of class distribution. **Precision-Recall curve** focuses on the tradeoff between precision and recall, particularly valuable for imbalanced datasets where accurately identifying the positive class is crucial.

For any model, both analyses are calculated and presented to stakeholders, ensuring that no critical performance aspect is overlooked.

1.6.1 Implementation Details

For a classification model, the validator performs a comprehensive threshold analysis:

```
results = {
    "probability_preprocessing": transformation_message,
    "roc_curve": roc_curve_data,     # FPR, TPR, thresholds, AUC
    "pr_curve": pr_curve_data,       # Precision, Recall, thresholds, AP
    "threshold_metrics": threshold_metrics  # Detailed metrics at 101 thresholds
}
```

This allows stakeholders to view AUC-ROC for overall discrimination ability, examine Average Precision (AP) for performance on positive class identification, and select optimal decision thresholds based on application-specific requirements using the detailed `threshold_metrics`.

2 Subgroup Analysis

Subgroup analysis partitions data according to categorical features and evaluates model performance independently for each partition. For a dataset $\mathcal{D} = \{(x_i, y_i, \hat{y}_i)\}_{i=1}^{n}$ containing features, ground truth, and predictions, we analyze performance across different subpopulations.

2.1 Conditional Performance Evaluation

For a categorical feature A with values $\{a_1, a_2, \ldots, a_k\}$, we define subgroups:

$$\mathcal{D}_j = \{(x_i, y_i, \hat{y}_i) \in \mathcal{D} \mid A(x_i) = a_j\}, \quad j \in \{1, 2, \ldots, k\}$$

For each subgroup $\mathcal{D}_j$ with $n_j = |\mathcal{D}_j|$ samples, we compute performance metrics. For a metric ϕ:

$$\phi_j = \phi(\{(y_i, \hat{y}_i) \mid (x_i, y_i, \hat{y}_i) \in \mathcal{D}_j\})$$

From a probabilistic perspective, we estimate:

$$\phi_j \approx \mathbb{E}[\phi(Y, \hat{Y}) \mid A = a_j]$$

2.2 Implementation Details

Our implementation:

1. Extracts categorical features from column metadata
2. For each categorical feature:
   (a) Identifies unique values in the feature
   (b) For each value, filters samples where feature equals that value
   (c) Computes the same metrics as for the overall dataset
   (d) Records the sample size for context

2.3 Statistical Considerations

The standard error for a metric ϕ_j in subgroup j scales as:

$$\mathrm{SE}(\phi_j) \propto \frac{1}{\sqrt{n_j}}$$

Smaller subgroups have higher uncertainty. Our implementation includes sample size reporting to contextualize metric reliability.

2.4 Diagnostic Applications

Isolating errors within subgroups helps diagnose model weaknesses:

$$\mathrm{Error}_j = \{(x_i, y_i, \hat{y}_i) \in \mathcal{D}_j \mid y_i \neq \hat{y}_i\}$$

For a subgroup with significantly worse performance:

$$\phi_j \ll \frac{1}{k} \sum_{i=1}^{k} \phi_i$$

This indicates potential knowledge gaps or representation issues in the training data.