Model Feedback API vs Data Point Comparison

Overview

The `/feedback` endpoint is a crucial component of the MLOps monitoring system that allows tracking and evaluating model performance in real-time. It accepts actual outcomes and compares them with predictions to calculate various performance metrics.

Endpoint Specifications

- **URL**: `/feedback`
- **Method**: `POST`
- **Content-Type**: `application/json`

Input Validation

1. **Required Fields**: Both `prediction` and `actual` must be present.
2. **Value Constraints**: Both values must be either `0` or `1`.
3. **Error Handling**: Returns **400 Bad Request** if validation fails.

Metrics Updated

The endpoint updates several Prometheus metrics:

1. Confusion Matrix Metrics

- `true_positives_total`: When `prediction=1` and `actual=1`.
- `true_negatives_total`: When `prediction=0` and `actual=0`.
- `false_positives_total`: When `prediction=1` and `actual=0`.
- `false_negatives_total`: When `prediction=0` and `actual=1`.

2. Performance Metrics

All metrics are automatically calculated based on the confusion matrix:

- **Accuracy** = (TP + TN) / (TP + TN + FP + FN)

- **Precision** = TP / (TP + FP)
- **Recall** = TP / (TP + FN)
- **F1 Score** = 2 × (Precision × Recall) / (Precision + Recall)

Implementation Details

1. The endpoint uses **atomic counters** for thread-safe metric updates.
2. Performance metrics are calculated **only when there's at least one feedback entry**.
3. **All errors are logged** for debugging purposes.
4. Metrics are **persisted using Prometheus** and can be visualized in dashboards.

Best Practices

1. **Send feedback as soon as actual outcomes are available.**
2. **Monitor error logs** for validation or processing issues.
3. **Use the metrics endpoint** to track model performance over time.
4. **Set up alerts** for significant drops in performance metrics.

Integration with Monitoring

- All metrics are **exposed via the Prometheus metrics endpoint**.
- Can be **integrated with Grafana dashboards**.
- Supports **alerts based on performance thresholds**.
- Enables **tracking model drift and performance degradation**.

Use Cases

The feedback system is essential for:

- Continuous **model monitoring**.
- **Performance tracking**.
- **Model drift detection**.
- **Quality assurance**.
- **Compliance reporting**.

Below is an example of starting feedback simulation...

This will send 20 feedback requests with varying accuracy patterns
You can monitor the metrics in Prometheus (http://localhost:9090)

Query to watch: model_accuracy

```
Unset
Starting feedback simulation...
This will send 20 feedback requests with varying accuracy patterns
You can monitor the metrics in Prometheus (http://localhost:9090)
Query to watch: model_accuracy


=== Scenario 1: Good Predictions ===
Error sending feedback: {"error":"Internal server error"}

[19:26:12] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=1.000, Precision=1.000, Recall=1.000, F1=1.000
[19:26:13] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=1.000, Precision=1.000, Recall=1.000, F1=1.000
[19:26:14] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=1.000, Precision=1.000, Recall=1.000, F1=1.000
[19:26:16] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=1.000, Precision=1.000, Recall=1.000, F1=1.000

=== Scenario 2: Mixed Performance ===
[19:26:17] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=1.000, Precision=1.000, Recall=1.000, F1=1.000
[19:26:18] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=1.000, Precision=1.000, Recall=1.000, F1=1.000
[19:26:19] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=1.000, Precision=1.000, Recall=1.000, F1=1.000
[19:26:20] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=1.000, Precision=1.000, Recall=1.000, F1=1.000
[19:26:22] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=1.000, Precision=1.000, Recall=1.000, F1=1.000

=== Scenario 3: Poor Performance ===
[19:26:23] Feedback sent - Prediction: 0, Actual: 1
Current Metrics: Accuracy=0.909, Precision=1.000, Recall=0.857, F1=0.923
[19:26:24] Feedback sent - Prediction: 0, Actual: 1
Current Metrics: Accuracy=0.833, Precision=1.000, Recall=0.750, F1=0.857
[19:26:25] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.846, Precision=1.000, Recall=0.750, F1=0.857
[19:26:26] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.857, Precision=1.000, Recall=0.750, F1=0.857
[19:26:27] Feedback sent - Prediction: 1, Actual: 0
Current Metrics: Accuracy=0.800, Precision=0.857, Recall=0.750, F1=0.800
```

```
=== Scenario 4: Recovery ===
[19:26:29] Feedback sent - Prediction: 0, Actual: 1
Current Metrics: Accuracy=0.750, Precision=0.857, Recall=0.667, F1=0.750
[19:26:30] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.765, Precision=0.875, Recall=0.700, F1=0.778
[19:26:31] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.778, Precision=0.875, Recall=0.700, F1=0.778
[19:26:32] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.789, Precision=0.875, Recall=0.700, F1=0.778
[19:26:33] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.800, Precision=0.889, Recall=0.727, F1=0.800

=== Sending Sample Feedback ===
Response: {'error': 'Internal server error'}
➜  mlops-monitoring git:(master) ✗ python scripts/send_feedback.py
Starting feedback simulation...
This will send 20 feedback requests with varying accuracy patterns
You can monitor the metrics in Prometheus (http://localhost:9090)
Query to watch: model_accuracy


=== Scenario 1: Good Predictions ===
[19:26:52] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.810, Precision=0.900, Recall=0.750, F1=0.818
[19:26:53] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.818, Precision=0.900, Recall=0.750, F1=0.818
[19:26:54] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.826, Precision=0.900, Recall=0.750, F1=0.818
[19:26:55] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.833, Precision=0.900, Recall=0.750, F1=0.818
[19:26:56] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.840, Precision=0.909, Recall=0.769, F1=0.833

=== Scenario 2: Mixed Performance ===
[19:26:58] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.846, Precision=0.917, Recall=0.786, F1=0.846
[19:26:59] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.852, Precision=0.923, Recall=0.800, F1=0.857
[19:27:00] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.857, Precision=0.923, Recall=0.800, F1=0.857
[19:27:01] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.862, Precision=0.929, Recall=0.812, F1=0.867
[19:27:02] Feedback sent - Prediction: 1, Actual: 1
```

```
Current Metrics: Accuracy=0.867, Precision=0.933, Recall=0.824, F1=0.875

=== Scenario 3: Poor Performance ===
[19:27:03] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.871, Precision=0.933, Recall=0.824, F1=0.875
[19:27:05] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.875, Precision=0.938, Recall=0.833, F1=0.882
[19:27:06] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.879, Precision=0.941, Recall=0.842, F1=0.889
[19:27:07] Feedback sent - Prediction: 0, Actual: 1
Current Metrics: Accuracy=0.853, Precision=0.941, Recall=0.800, F1=0.865
[19:27:08] Feedback sent - Prediction: 1, Actual: 0
Current Metrics: Accuracy=0.829, Precision=0.889, Recall=0.800, F1=0.842

=== Scenario 4: Recovery ===
[19:27:09] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.833, Precision=0.889, Recall=0.800, F1=0.842
[19:27:10] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.838, Precision=0.895, Recall=0.810, F1=0.850
[19:27:12] Feedback sent - Prediction: 1, Actual: 1
Current Metrics: Accuracy=0.842, Precision=0.900, Recall=0.818, F1=0.857
[19:27:13] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.846, Precision=0.900, Recall=0.818, F1=0.857
[19:27:14] Feedback sent - Prediction: 0, Actual: 0
Current Metrics: Accuracy=0.850, Precision=0.900, Recall=0.818, F1=0.857
```

Using Data Points as “Ground Truth” falls short

1. Overfitting & Unrealistic Performance Metrics

- Models are trained to **minimize error** on training data, meaning they can **memorize** patterns rather than generalizing to unseen data.
- Performance metrics on training data **almost always appear artificially high**, leading to **overconfidence** in the model's capabilities.

2. Lack of Generalization

- The true goal of a model is to make accurate predictions on **new, unseen data** (i.e., real-world scenarios).

- If performance is only evaluated on training data, we **can't assess how well the model generalizes** to new inputs.

3. Data Leakage & Bias

- If the training data is used for performance evaluation, **data leakage** can occur, where the model has access to information it wouldn't have in a real-world setting.
- This creates a **false sense of reliability**, hiding potential weaknesses in the model.

4. No Insight into Model Drift

- Models degrade over time due to **shifting data distributions** (i.e., concept drift).
- Tracking model performance requires evaluating predictions **on fresh, real-world data**, not the same dataset it was trained on.

5. Incorrect Business Decision-Making

- If stakeholders rely on **inflated training performance metrics**, they might **overestimate the model's capabilities**, leading to incorrect or costly decisions.
- A proper evaluation should include **validation data, test data, and real-world feedback loops**.