Both `infer_signature(X_test, y_pred)` and `infer_signature(y_test, y_pred)` can be correct, but they serve **different purposes**. Here's how to choose the right one:

---

1. `signature = infer_signature(X_test, y_pred)`

✅ **Correct for most cases**

- **What it does**: Infers the model's input/output schema from:
  - **Input (`X_test`)**: Features the model expects (e.g., `DataFrame` / `np.array` of features).
  - **Output (`y_pred`)**: Predictions generated by the model.
- **When to use**:
  - Logging a **predictive model** (e.g., scikit-learn, TensorFlow).
  - Ensures the signature matches the model's actual API (`model.predict(X)`).

Example:

```python
from mlflow.models import infer_signature

# X_test = Feature matrix (input), y_pred = Model predictions (output)
signature = infer_signature(X_test, y_pred)
mlflow.sklearn.log_model(model, "model", signature=signature)
```

---

2. `signature = infer_signature(y_test, y_pred)`

⚠️ **Rarely useful**

- **What it does**: Infers a signature where:
  - **Input (`y_test`)**: True labels (ground truth).
  - **Output (`y_pred`)**: Predicted labels.
- **When to use**:
  - Only if you're logging a **metric calculator** (e.g., a function that compares `y_test` and `y_pred`).
  - Not for standard predictive models!

Example (Edge Case):

```python
# This is ONLY correct if you're logging a "metric function"
def calculate_metrics(y_true, y_pred):
    return {"mse": mean_squared_error(y_true, y_pred)}

# Signature: Input=true labels, Output=metrics
signature = infer_signature(y_test, y_pred)
mlflow.pyfunc.log_model(
    artifact_path="metric_fn",
    python_model=calculate_metrics,
    signature=signature
)
```

Key Differences

Scenario	Correct Signature	Purpose
Predictive Model	`infer_signature(X_test, y_pred)`	Tracks input features → predictions.
Metric/Evaluation Function	`infer_signature(y_test, y_pred)`	Tracks true labels → metrics.

Best Practices

1. For **most models**, use:

   ```python
   signature = infer_signature(X_test, y_pred)
   ```

2. Validate the signature matches your model's API:

   ```python
   model = mlflow.pyfunc.load_model("runs:/<RUN_ID>/model")
   model.predict(X_test[:1])  # Should work without errors
   ```

3. Always provide an `input_example` for clarity:

   ```python
   mlflow.sklearn.log_model(
       model,
       "model",
   ```

```
    signature=signature,
    input_example=X_test[:1]
)
```

Common Mistakes

- Using `y_test` instead of `X_test` for predictive models → MLflow will expect labels as input (causing errors in deployment).
- Not inferring a signature at all → Triggers the warning you saw earlier.