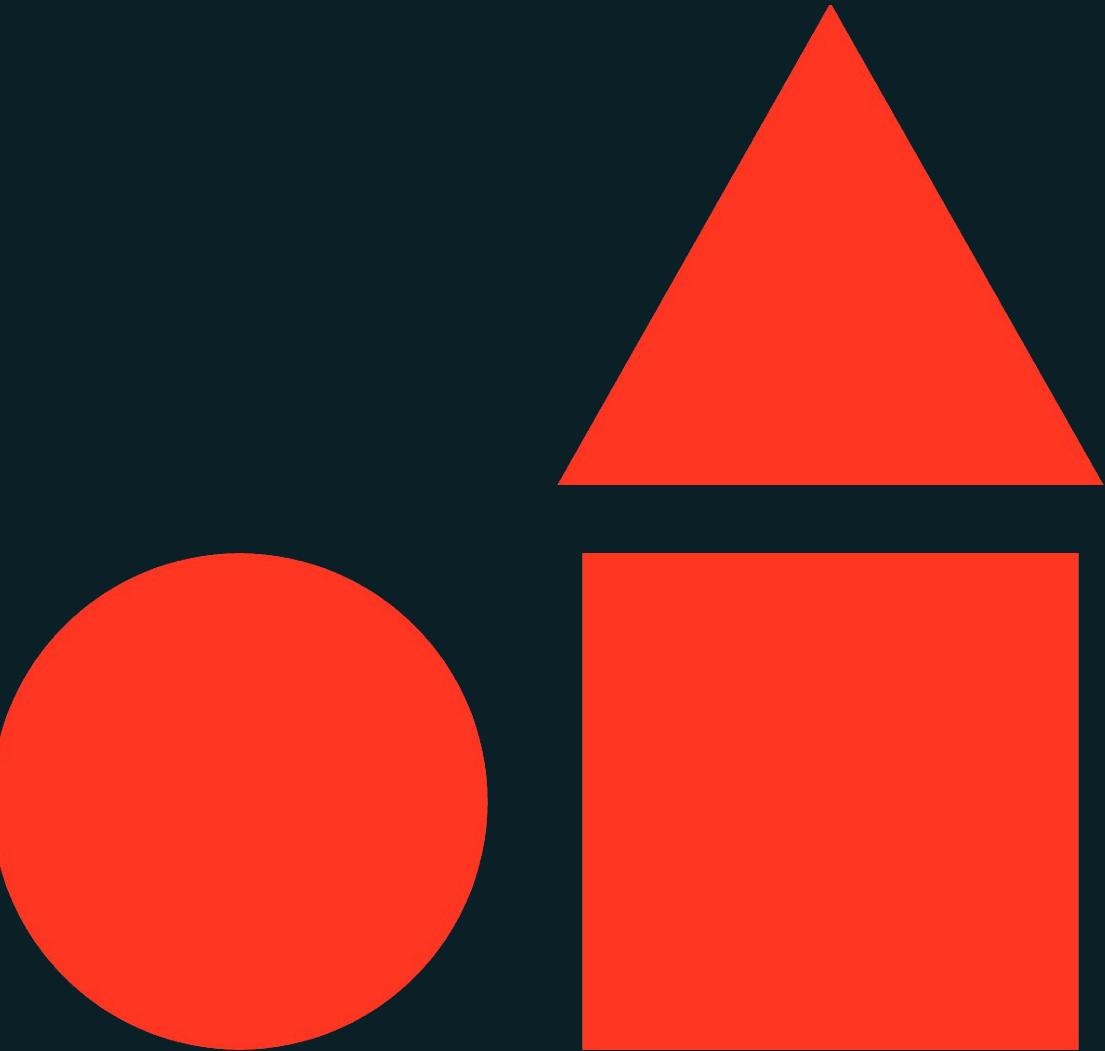




Machine Learning Model Development

Databricks Academy



Course Learning Objectives

- Describe fundamental concepts of developing ML models
- Describe the main components of MLflow for model development
- Describe hyperparameter tuning and leverage Optuna for HPO
- Utilize MLflow for model tracking and development
- Leverage the AutoML API and Databricks UI for AutoML to run an experiment, identify the model based on a preferred metric, and modify a generated model



Learning Goals

Upon completion of this content, you should be able to:

- Describe fundamental concepts of machine learning.
- Describe the main components of MLflow for model development.
- Describe hyperparameter tuning and methods.
- Utilize MLflow for model tracking and model tuning with Optuna.
- Create an AutoML experiment, identify the best model, and modify the generated models.



Prerequisites/Technical Considerations

Things to keep in mind before you work through this course

Prerequisites

- 1 Knowledge of fundamental concepts of regression, classification, and clustering methods.
- 2 Familiarity with Databricks workspace and notebooks.
- 3 Intermediate level knowledge of Python.

Technical Considerations

- 1 A cluster running on **DBR ML 16.3**
- 2 **Unity Catalog** enabled workspace



AGENDA

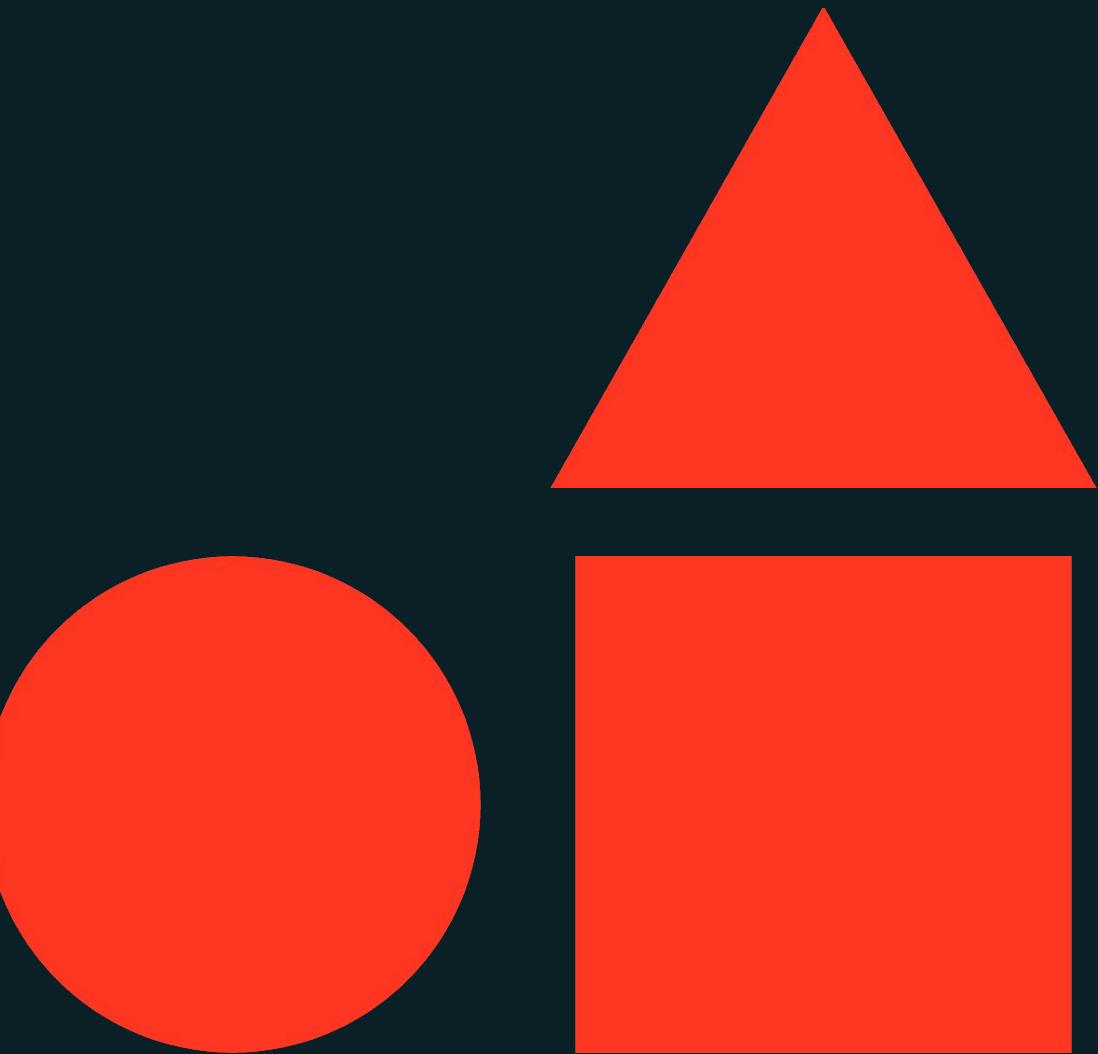
1. Model Development Workflow	DEMO	LAB
Model Development and MLflow	✓	✓
Evaluating Model Performance		
02. Hyperparameter Tuning		
Hyperparameter Tuning Fundamentals		
Hyperparameter Tuning with Optuna	✓	✓
03. AutoML		
Automated Model Development with AutoML	✓	✓





Machine Learning Model Development

Machine Learning Model Development



Problem Statement

Things you'll be able to do after completing this module

- Describe principle categories of machine learning (supervised, unsupervised, reinforcement learning, etc.).
- Describe principle types of supervised learning: regression, classification, forecasting.
- Describe traditional steps of the machine learning model training workflow.
- Describe the benefits of MLflow on Databricks
- Describe Databricks Autologging.
- Review of key metrics for ML algorithms (regression, classification, clustering)





Model Development Workflow

LECTURE

Model Development and MLflow



© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

Types of Machine Learning

The basics of supervised, unsupervised, and reinforcement learning

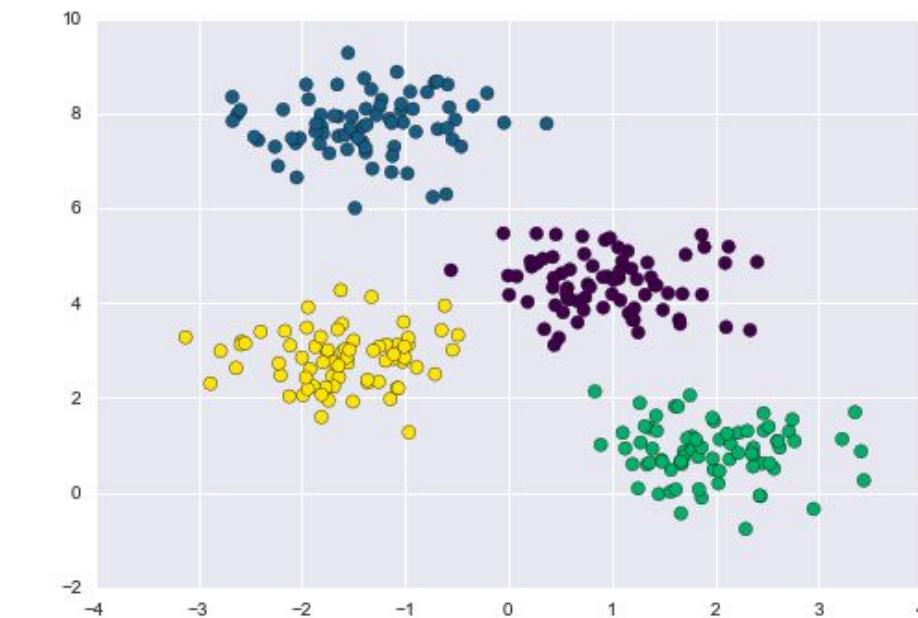
Supervised Learning

- Data has a set of input records with **associated labels**.
- Goal is to predict the output labels given an **unlabeled input**.
- Model the conditional probability:
 $P(Y | X_1, X_2, \dots, X_n)$



Unsupervised Learning

- Unlabeled data (no known function output)
- Clustering (group records with similar features)
- Dimensionality reduction (reduce feature space)
- Model the joint probability: $P(X_1, X_2, \dots, X_n)$

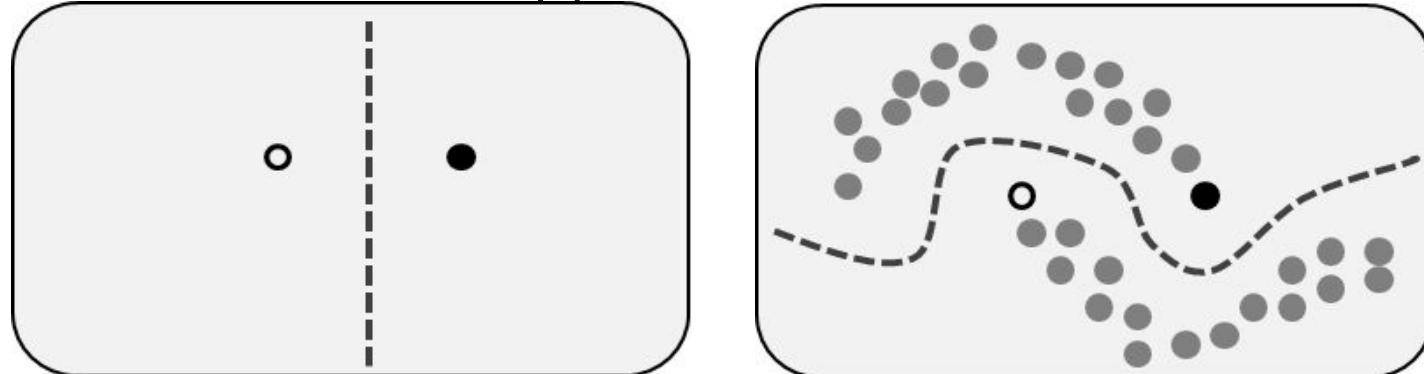


Types of Machine Learning

The basics of supervised, unsupervised, and reinforcement learning

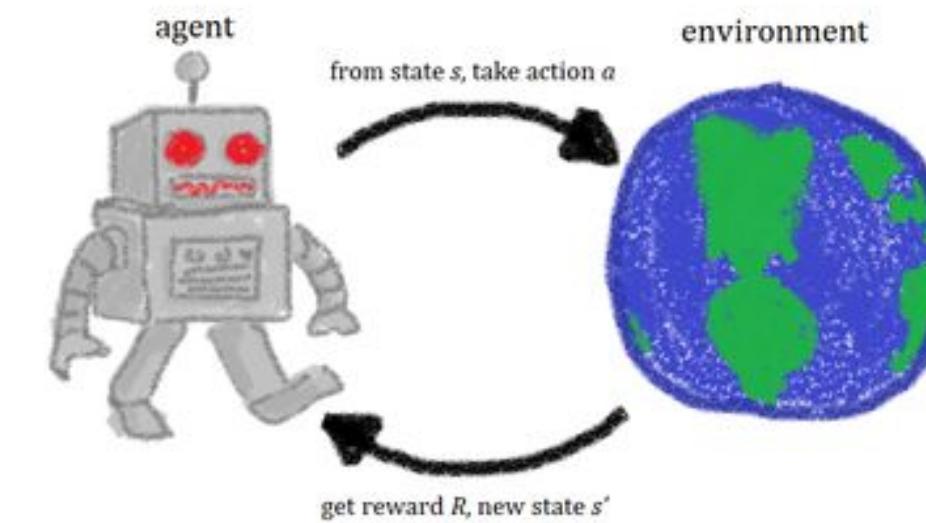
Hybrids

- Combines supervised learning and unsupervised learning
- Labeled and unlabeled data, mostly unlabeled
- Semi-supervised:
 - Use small set of labeled data to infer labels to a larger set
- Self-supervised:
 - Learn patterns of data in one (larger) dataset that can be applied to another



Reinforcement Learning

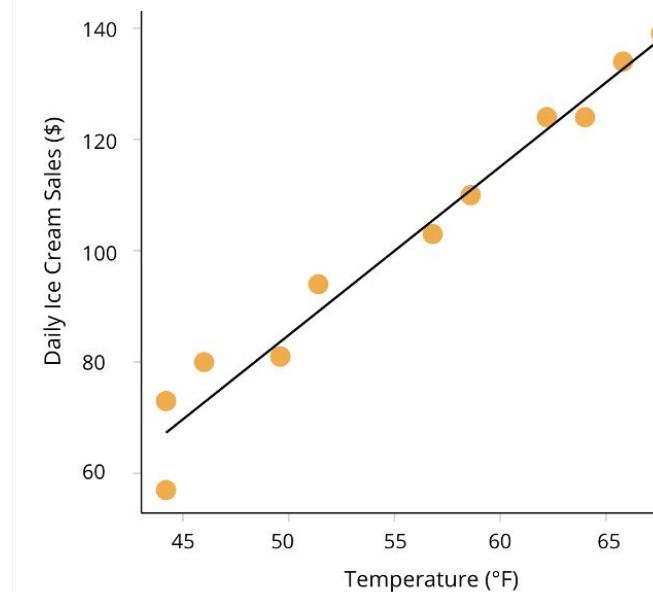
- States, actions, and rewards
- Useful for exploring spaces and exploiting information to maximize expected cumulative rewards
- Frequently **utilizes neural networks and deep learning**



Types of Supervised Learning

Regression

- Predicting a **continuous** output
- **Example:** predict sales, predict number of viewers



Classification

- Predicting a **categorical/discrete** output
- **Example:** yes/no, image classification, predict disease



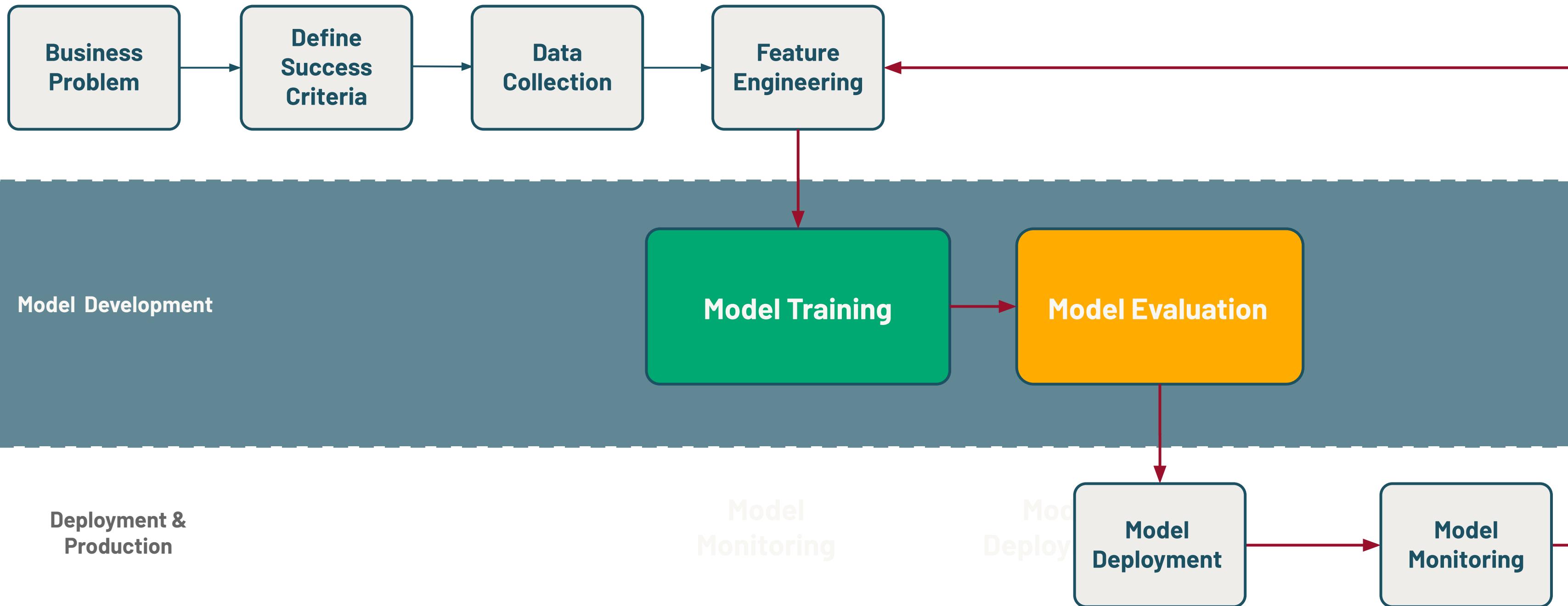
Forecasting

- Predicting future values based on **historical** data
- **Example:** demand forecasting



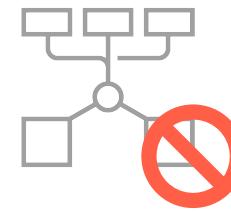
The Machine Learning Lifecycle

Model development and evaluation



Core Machine Learning Dev. Issues

Modern ML lifecycle comes with many challenges



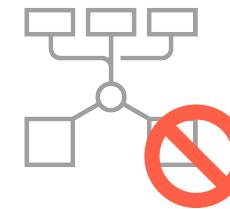
Reproducibility and Experiment Tracking

Difficulty in ensuring consistent results across different environments and effectively documenting and organizing experiments



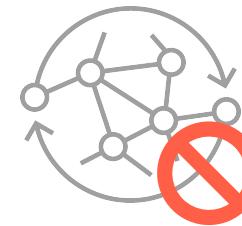
Core Machine Learning Dev. Issues

Modern ML lifecycle comes with many challenges



Reproducibility and Experiment Tracking

Difficulty in ensuring consistent results across different environments and effectively documenting and organizing experiments.



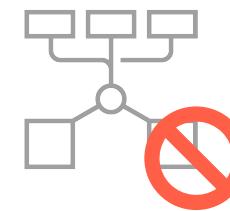
Model Evaluation and Comparison

Challenges in accurately comparing and selecting the best-performing model due to variations in evaluation metrics and design.



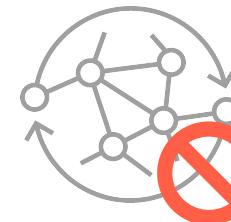
Core Machine Learning Dev. Issues

Modern ML lifecycle comes with many challenges



Reproducibility and Experiment Tracking

Difficulty in ensuring consistent results across different environments and effectively documenting and organizing experiments.



Model Evaluation and Comparison

Challenges in accurately comparing and selecting the best-performing model due to variations in evaluation metrics and design.



Deployment and Standardization

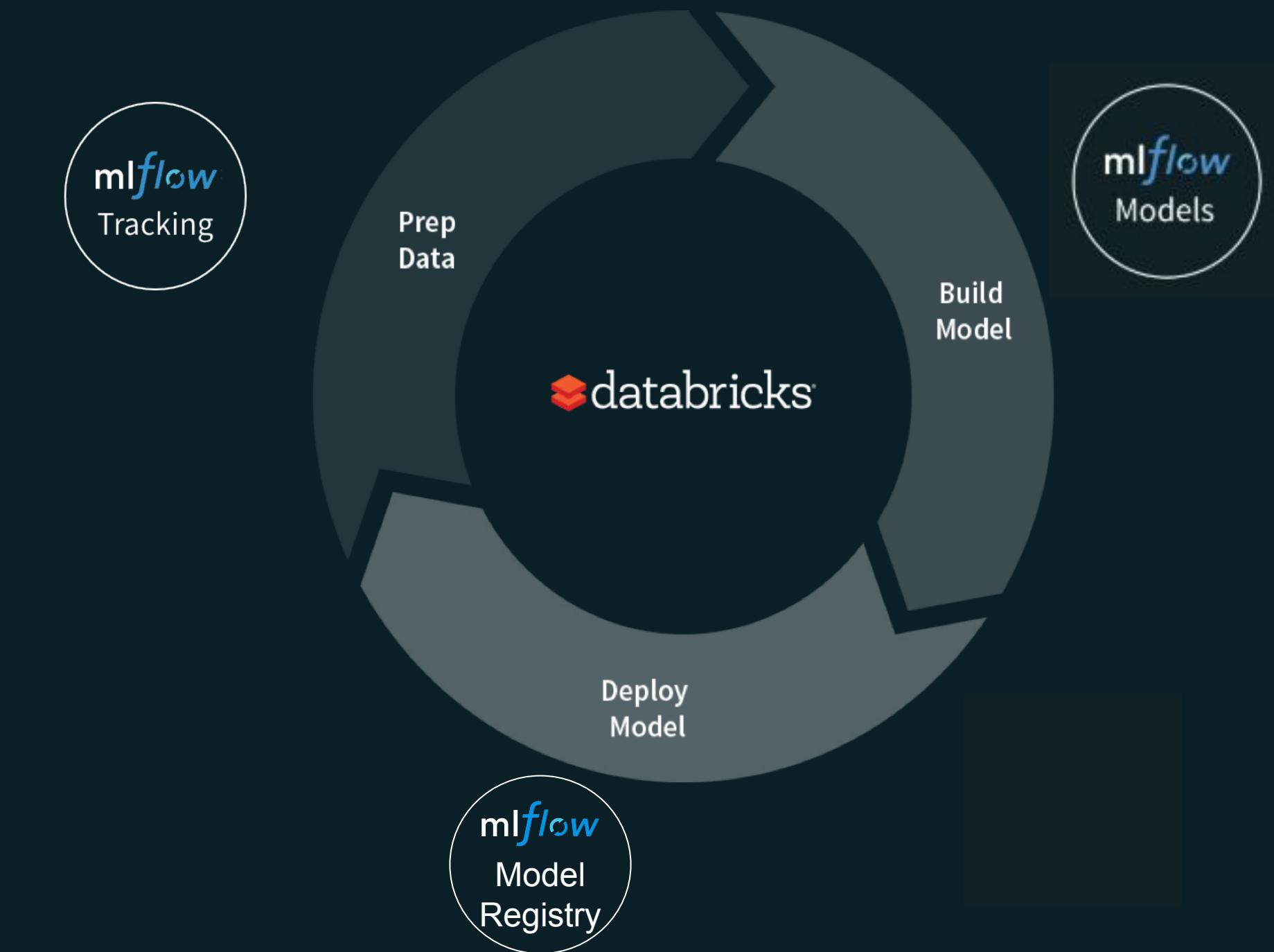
Complexities in transitioning model, ensuring seamless integration, standardized deployment processes.



MLflow

MLflow addresses common ML development issues.

- Open-source platform for machine learning lifecycle
- Operationalizing machine learning
- Developed by Databricks
- Pre-installed on the Databricks Runtime for ML



MLflow Components

The four components of MLflow

mlflow™

Tracking

Record and query experiments: code, data, config, results

Focus of this course

mlflow™

Model

General **S** model format that supports diverse deployment tools

mlflow™

Model

Registry

Centralized and collaborative model lifecycle management

APIs: CLI, Python, R, Java, REST

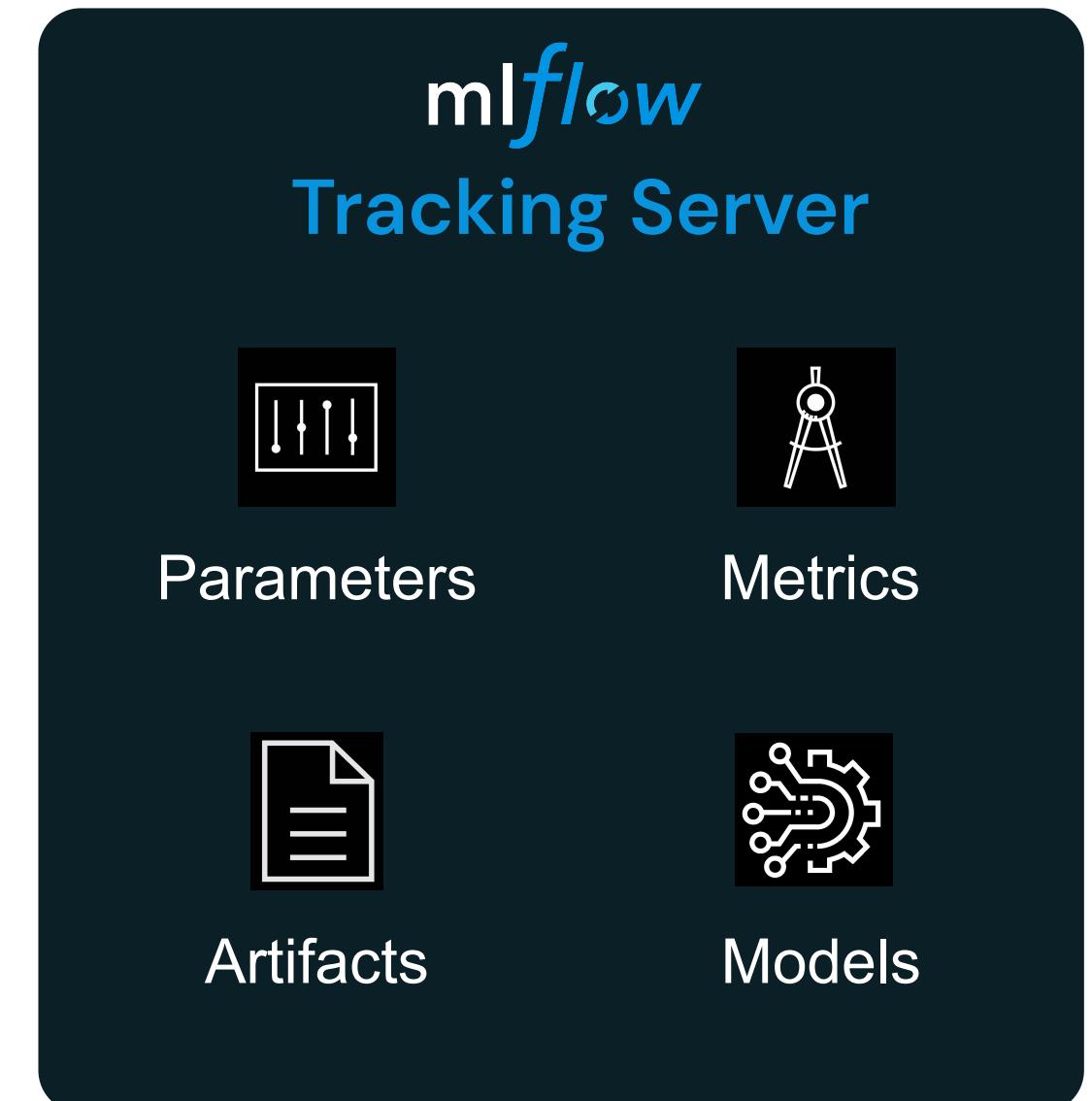


© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

MLflow Model Tracking

Make your ML workflow more manageable and transparent

- Record experiment **parameters** such as hyperparameters and model configurations.
- Log **metrics** and compare them for performance measures.
- Store and manage output **artifacts** such as trained models and plots.
- Store model's **source code** from the run.



Model Tracking and Custom Logging

Supports for custom logging

Custom logging for;

- **Parameters** (such as custom model hyperparameters),
- **Custom metrics**
- **Artifacts** (such as the trained model and additional files) within the context of the run.



```
1 # Log metrics
2 y_pred = model.predict(X_test)
3 super_score = custom_score(y_test, y_pred)
4 mlflow.log_metric('super_score', super_score)
5
6 # Log artifacts (e.g., model file)
7 model_path = 'random_forest_model.pkl'
8 mlflow.sklearn.log_model(model, model_path)
9 mlflow.log_artifact(model_path)
10
11 # Log params
12 mlflow.log_params({'param1': 123, 'param2': 'abc'})
```



Model Tracking and Autologging

Model Tracking Demo [Provide feedback](#)

Overview

Model metrics

System metrics

Artifacts

inspect, visualize and Compare Metrics

`mlflow.autolog()`

Automatically track ML development with one line of code: parameters, metrics, data lineage, model, and environment.

Reproduce Run

Model registered

Artifacts (Figures, datasets, text files)

Model Details

Run ID	70e02bb8bec2418197fc6033c0484342
Duration	13.2s
Datasets used	dataset (98736df7) Training +2
Tags	spa... : path=dbfs:/mnt/dbacademy-datasets/m...
Source	1.2 - Model Tracking with MLflow
Logged models	sklearn
Registered models	menaf_gul_8vg5_da.default.diabetes-predictions v1

Parameters (4)

Search parameters	
Parameter	Value
criterion	gini
max_depth	50

Metrics (8)

Search metrics	
Metric	Value
train_f1	0.81349218257...
test_recall	0.6965090090...

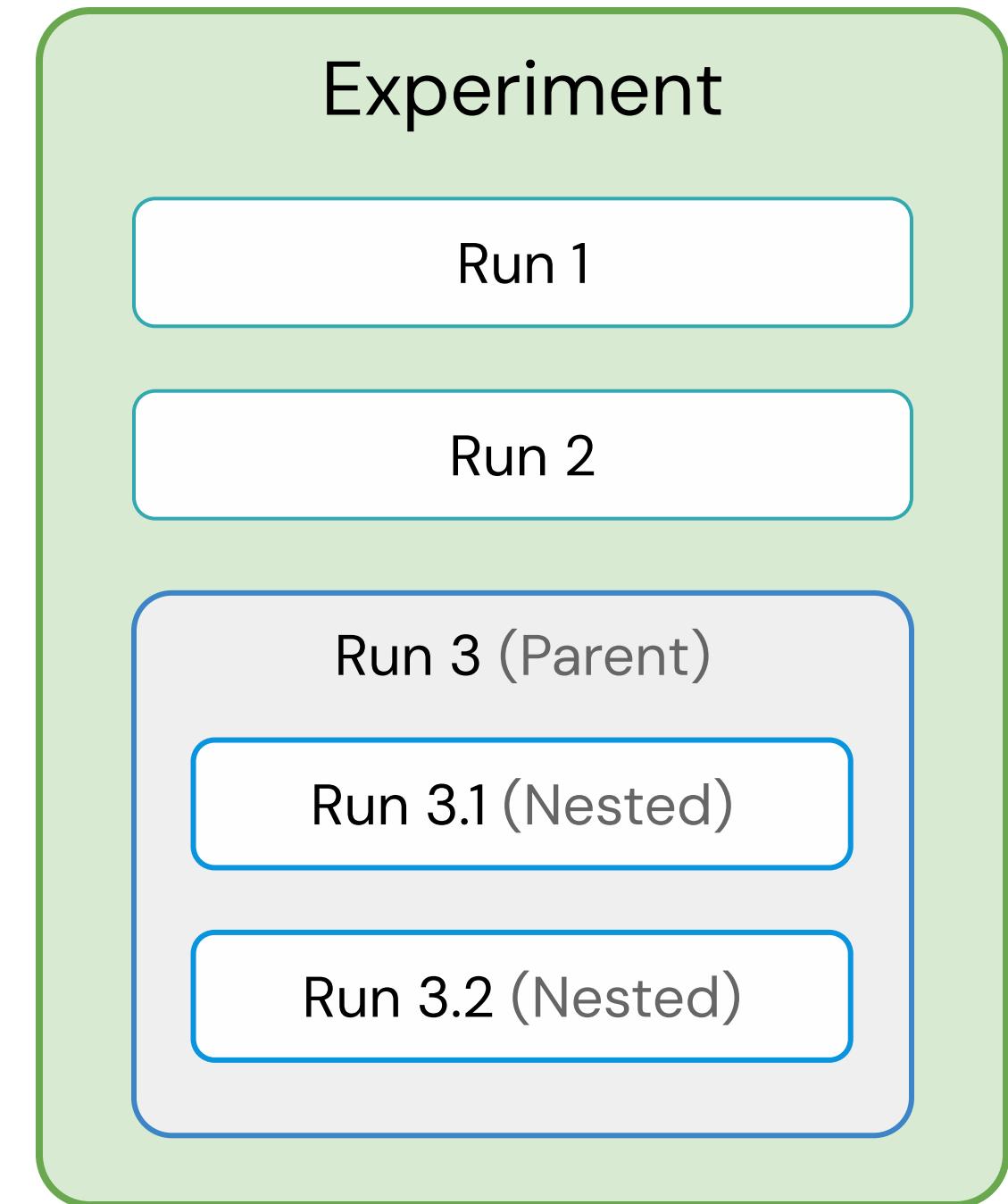


MLflow Concepts

Experiment → Runs → Metadata / Artifacts

MLflow experiment:

- A higher-level organizational unit that **encompasses a set of runs**.
- Groups and organizes related runs, typically conducted to explore different configurations, parameters, or algorithms.
- Two options
 - Notebook Experiment: tied to the notebook
 - Workspace experiment: Manually specify the path. Multiple notebooks can create runs in it.

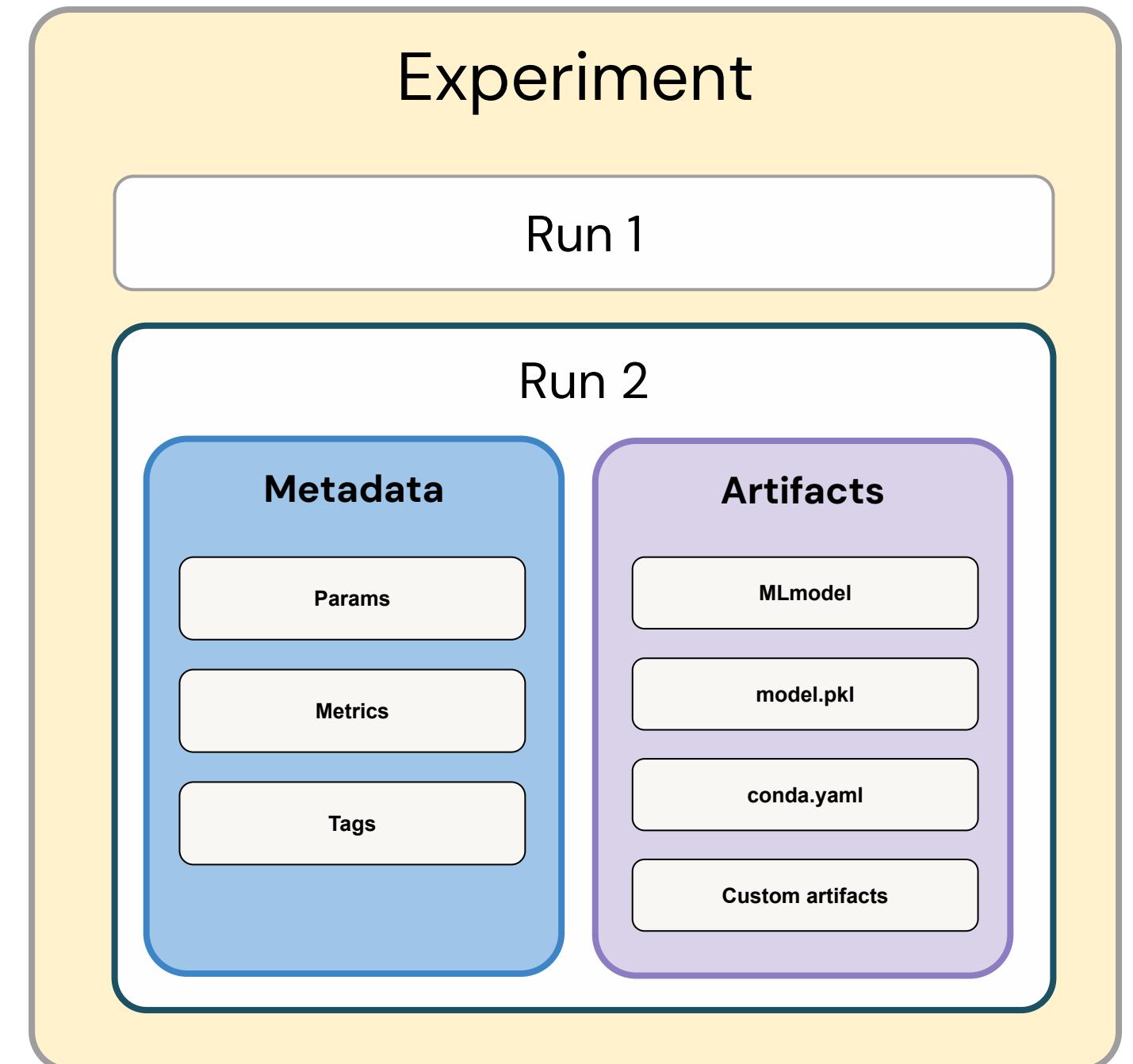


MLflow Concepts

Experiment → Runs → Metadata / Artifacts

MLflow run:

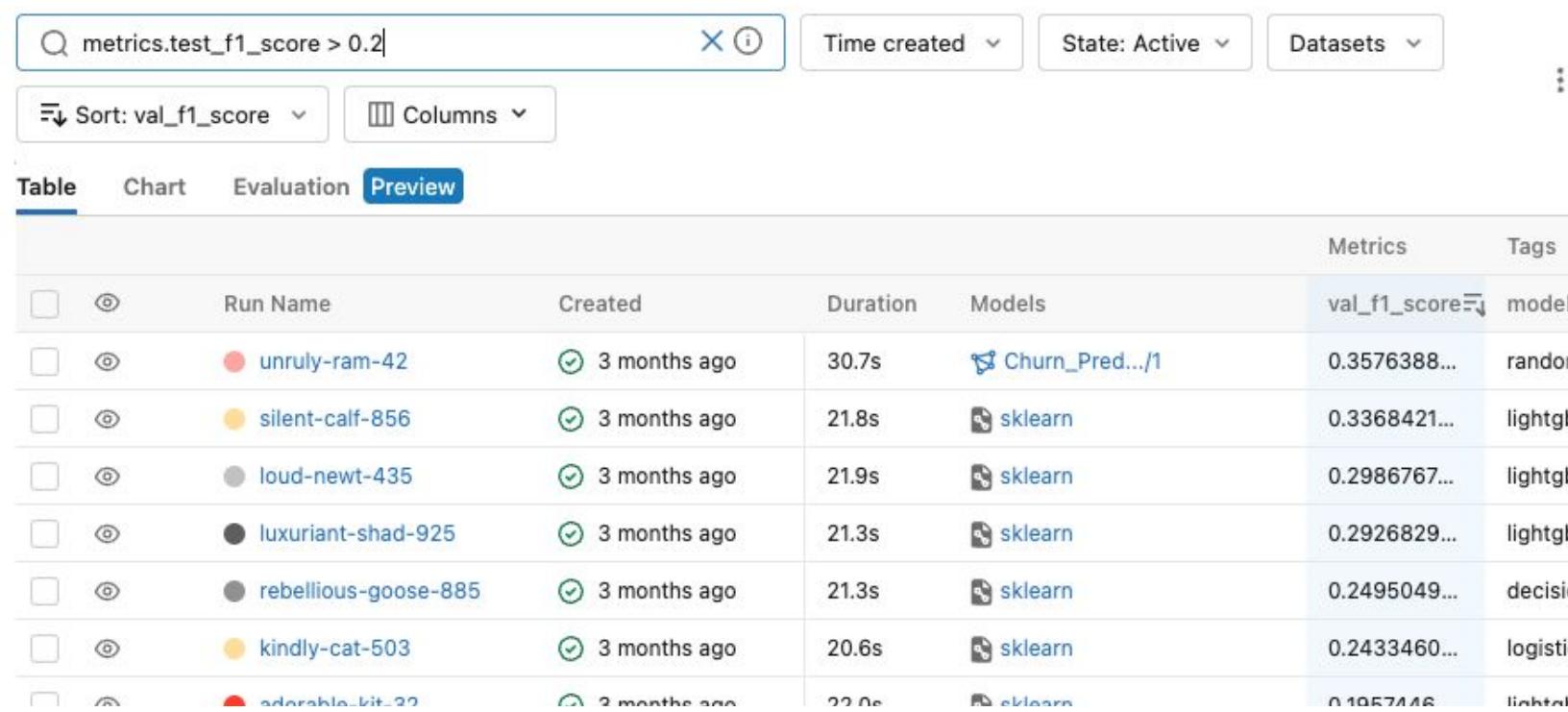
- Executions of data science code.
- Belongs to only one experiment.
- Contains
 - Metadata
 - Artifacts
- Has a default Py Func flavor which is wrapper around the native model.



Examining Past Runs

Via MLflow UI

- List and filter runs by metrics
- Built in to Databricks platform



A screenshot of the MLflow UI interface. At the top, there is a search bar containing the query "metrics.test_f1_score > 0.2". Below the search bar are several filters: "Time created", "State: Active", "Datasets", "Sort: val_f1_score", and "Columns". The main area shows a table of runs. The columns are: Run Name, Created, Duration, Models, Metrics, and Tags. The "Preview" tab is selected at the bottom of the table header. The table contains the following data:

	Run Name	Created	Duration	Models	Metrics	Tags
1	unruly-ram-42	3 months ago	30.7s	Churn_Pred.../1	val_f1_score=0.3576388...	random_forest
2	silent-calf-856	3 months ago	21.8s	sklearn	val_f1_score=0.3368421...	lightgbm
3	loud-newt-435	3 months ago	21.9s	sklearn	val_f1_score=0.2986767...	lightgbm
4	luxuriant-shad-925	3 months ago	21.3s	sklearn	val_f1_score=0.2926829...	lightgbm
5	rebellious-goose-885	3 months ago	21.3s	sklearn	val_f1_score=0.2495049...	decision_tree
6	kindly-cat-503	3 months ago	20.6s	sklearn	val_f1_score=0.2433460...	logistic_regression
7	adorable-kit-22	3 months ago	22.0s	sklearn	val_f1_score=0.1057446...	lightgbm

Via MLflow API

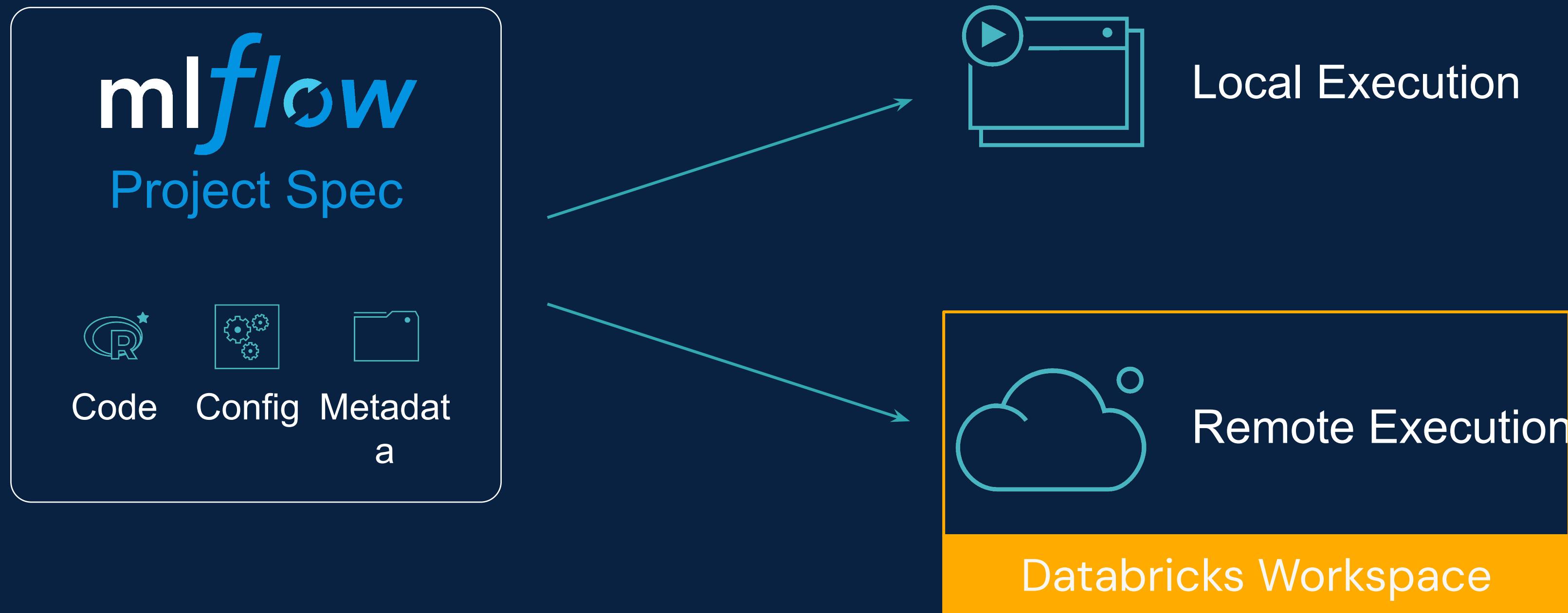
- MLflowClient Object
- List experiments
- Search runs by metrics, model etc.



```
1 from mlflow import MlflowClient
2
3 runs = MlflowClient().search_runs(
4     experiment_ids="0",
5     filter_string="params.model = 'random_forest'",
6     max_results=5,
7     order_by=["metrics.accuracy DESC"],
8 )
```



mlflow Projects



MLflow Project

Package in a reusable and reproducible manner

Key aspects:

- **Packaging**: entire project and dependencies
- **Reproducibility**
- **Easy execution**: Via CLI and APIs
- **Versioning and sharing**



Example MLflow Project

```
my_project/
  └── MLproject
      ├── conda.yaml
      ├── main.py
      └── model.py
      ...
      . . .
```

```
conda_env: conda.yaml

entry_points:
  main:
    parameters:
      training_data: path
      lambda: {type: float, default: 0.1}
    command: python main.py {training_data} {lambda}
```

```
$ mlflow run git://<my_project>
```

```
mlflow.run("git://<my_project>", ...)
```





Model Development Workflow

LECTURE

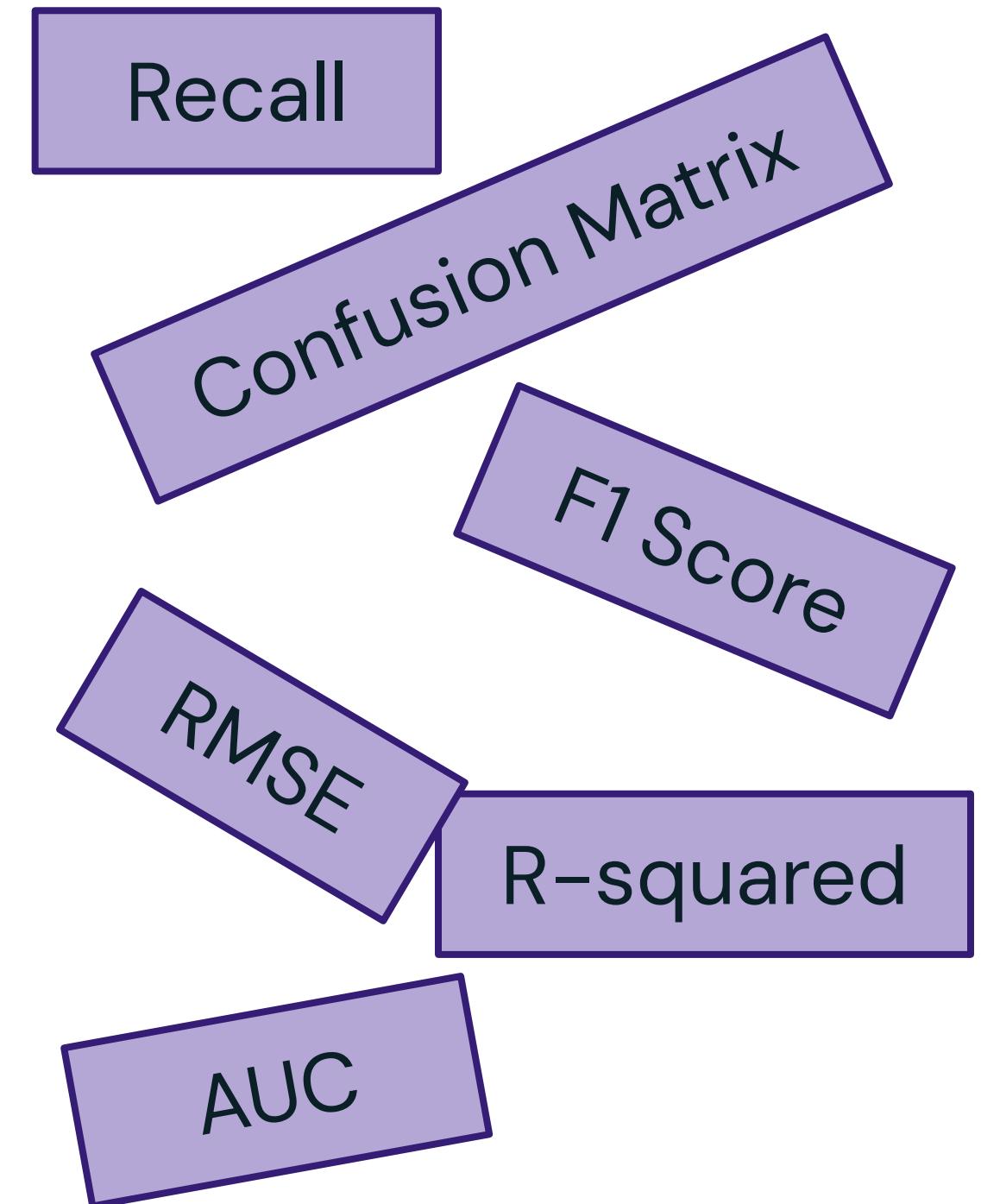
Evaluating Model Performance



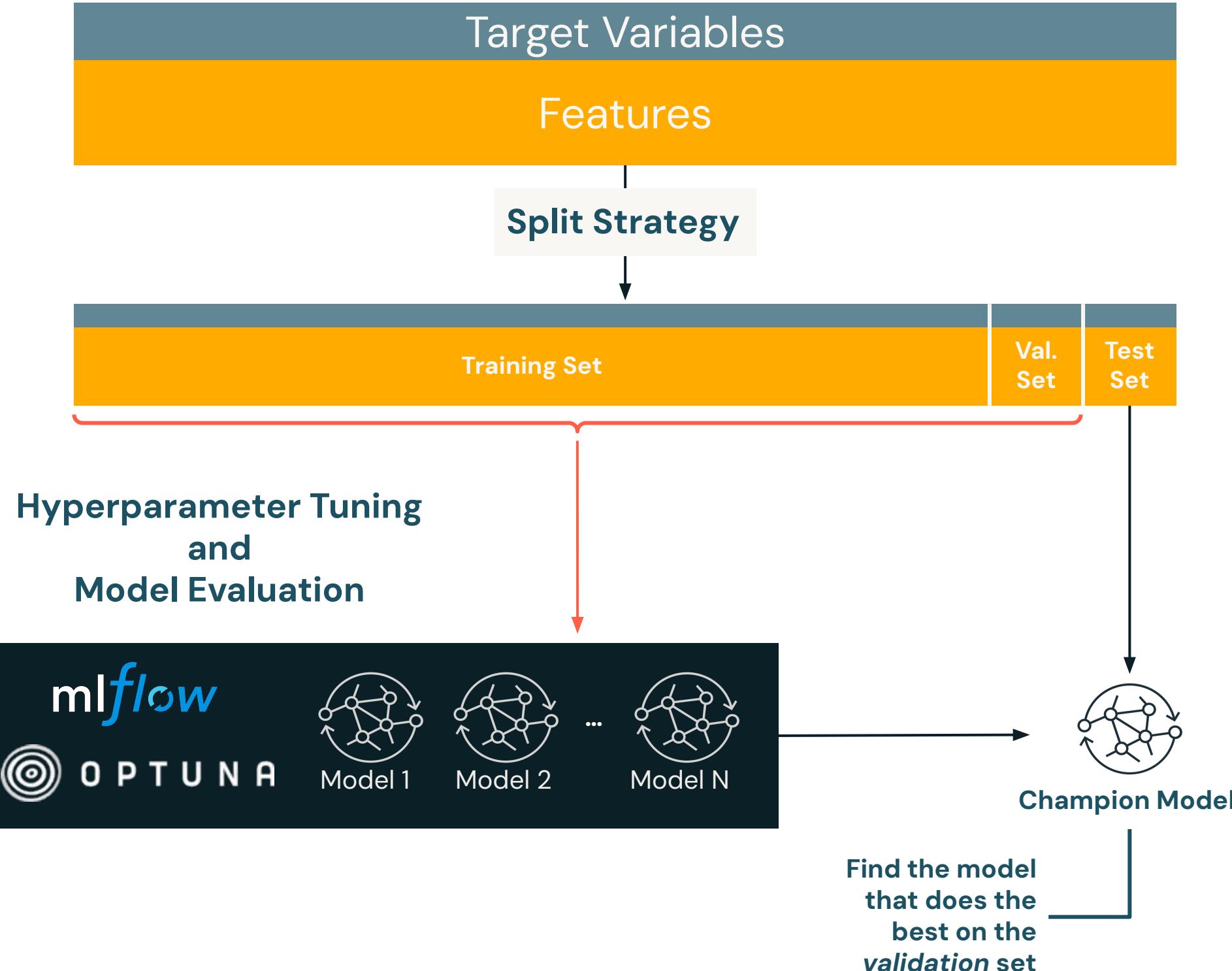
© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

Purpose of Computing Evaluation Metrics

- Evaluation metrics provide a numerical representation of how well a model is doing.
- Metrics enable the comparison of different models or different versions of the same model.
- Evaluation metrics guide the fine-tuning of models.
- Metrics prove an objective basis for choosing between different models or approaches.



How to Build and Evaluate Models?



Dataset Type	Purpose	Usage	Model Impact
Training	Training data for model	Minimize loss during optimization	Direct impact on internal model parameters
Validation	Hyperparameter tuning	Used for preventing overfitting	Indirect impact on model parameters
Test	Evaluate generalizability	Assesses real-world performance on unseen data	No influence on model training or selection



Review of Common Metrics

Supervised Learning – Regression

Metric	Description	Values	Code

[More Details: Appendix A](#)



© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

Review of Common Metrics

Supervised Learning – Regression

Metric	Description	Values	Code
R ²	Measures how well the model explains the variation in the target variable.	Range: $(-\infty, 1]$ Larger is better	<pre>from sklearn.metrics import r2_score r2 = r2_score(y_true, y_pred)</pre>

[More Details: Appendix A](#)



© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

Review of Common Metrics

Supervised Learning – Regression

Metric	Description	Values	Code
R ²	Measures how well the model explains the variation in the target variable.	Range: $(-\infty, 1]$ Larger is better	<pre>from sklearn.metrics import r2_score r2 = r2_score(y_true, y_pred)</pre>
Mean Absolute Error	Measures the average absolute difference between actual and predicted values.	Range: $[0, \infty)$ Lower is better	<pre>from sklearn.metrics import mean_absolute_error mae = mean_absolute_error(y_true, y_pred)</pre>

[More Details: Appendix A](#)



© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

Review of Common Metrics

Supervised Learning – Regression

Metric	Description	Values	Code
R ²	Measures how well the model explains the variation in the target variable.	Range: $(-\infty, 1]$ Larger is better	<pre>from sklearn.metrics import r2_score r2 = r2_score(y_true, y_pred)</pre>
Mean Absolute Error	Measures the average absolute difference between actual and predicted values.	Range: $[0, \infty)$ Lower is better	<pre>from sklearn.metrics import mean_absolute_error mae = mean_absolute_error(y_true, y_pred)</pre>
Mean Squared Error	Measures the average of squared differences between actual and predicted values. Squaring gives higher weight to large errors.	Range: $[0, \infty)$ Lower is better	<pre>from sklearn.metrics import mean_squared_error mse = mean_squared_error(y_true, y_pred)</pre>

[More Details: Appendix A](#)



Review of Common Metrics

Supervised Learning – Regression

Metric	Description	Values	Code
R ²	Measures how well the model explains the variation in the target variable.	Range: $(-\infty, 1]$ Larger is better	<pre>from sklearn.metrics import r2_score r2 = r2_score(y_true, y_pred)</pre>
Mean Absolute Error	Measures the average absolute difference between actual and predicted values.	Range: $[0, \infty)$ Lower is better	<pre>from sklearn.metrics import mean_absolute_error mae = mean_absolute_error(y_true, y_pred)</pre>
Mean Squared Error	Measures the average of squared differences between actual and predicted values. Squaring gives higher weight to large errors.	Range: $[0, \infty)$ Lower is better	<pre>from sklearn.metrics import mean_squared_error mse = mean_squared_error(y_true, y_pred)</pre>
Root Mean Squared Error	Similar to MSE but takes the square root, making it more interpretable to the original units.	Range: $[0, \infty)$ Lower is better	<pre>from sklearn.metrics import mean_squared_error mse = mean_squared_error(y_true, y_pred, squared=False)</pre>

[More Details: Appendix A](#)



Review of Common Metrics

Supervised Learning - Classification

Metric	Description
Accuracy	
Precision	
Recall (Sensitivity)	
F1 Score	

		Prediction	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Confusion Matrix

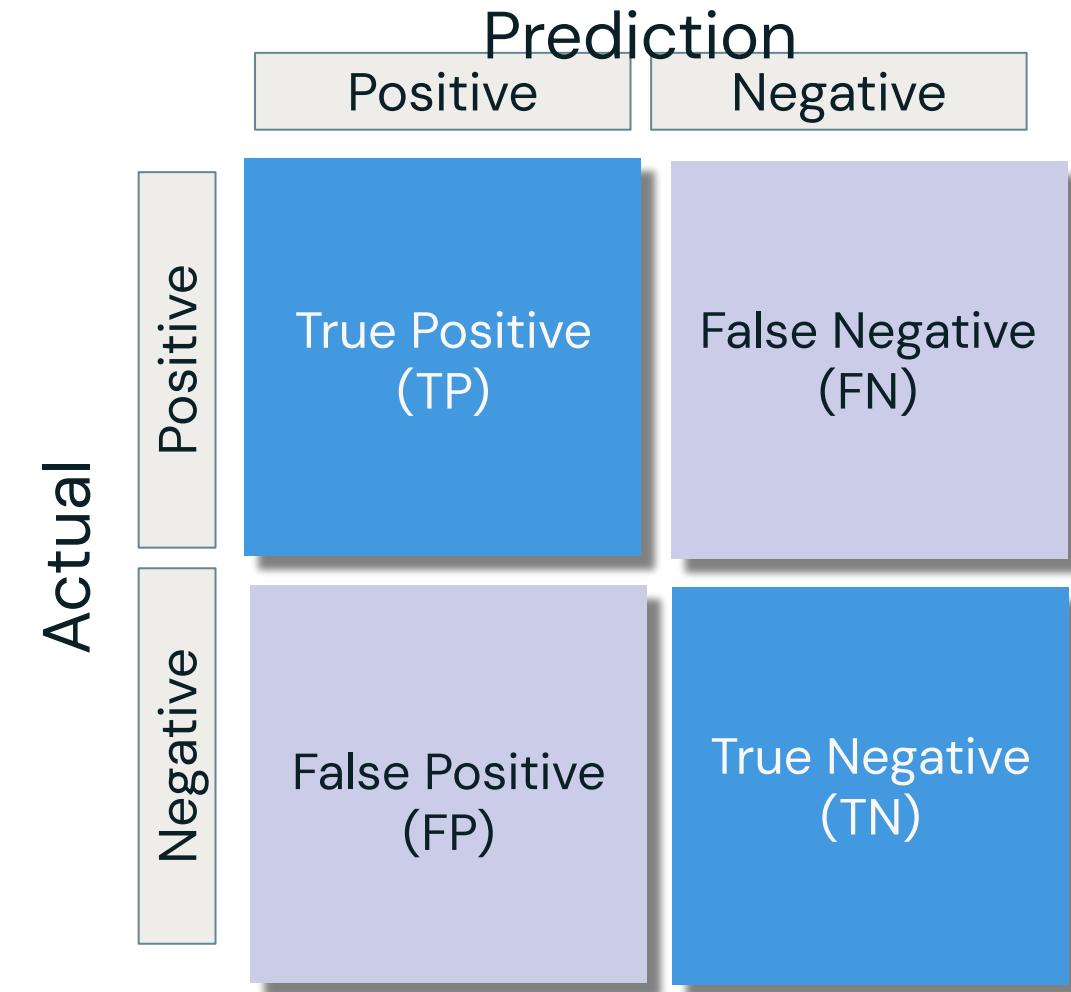
```
from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(y_true, y_pred)
```



Review of Common Metrics

Supervised Learning – Classification

Metric	Description
Accuracy	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.
Precision	
Recall (Sensitivity)	
F1 Score	



Confusion Matrix

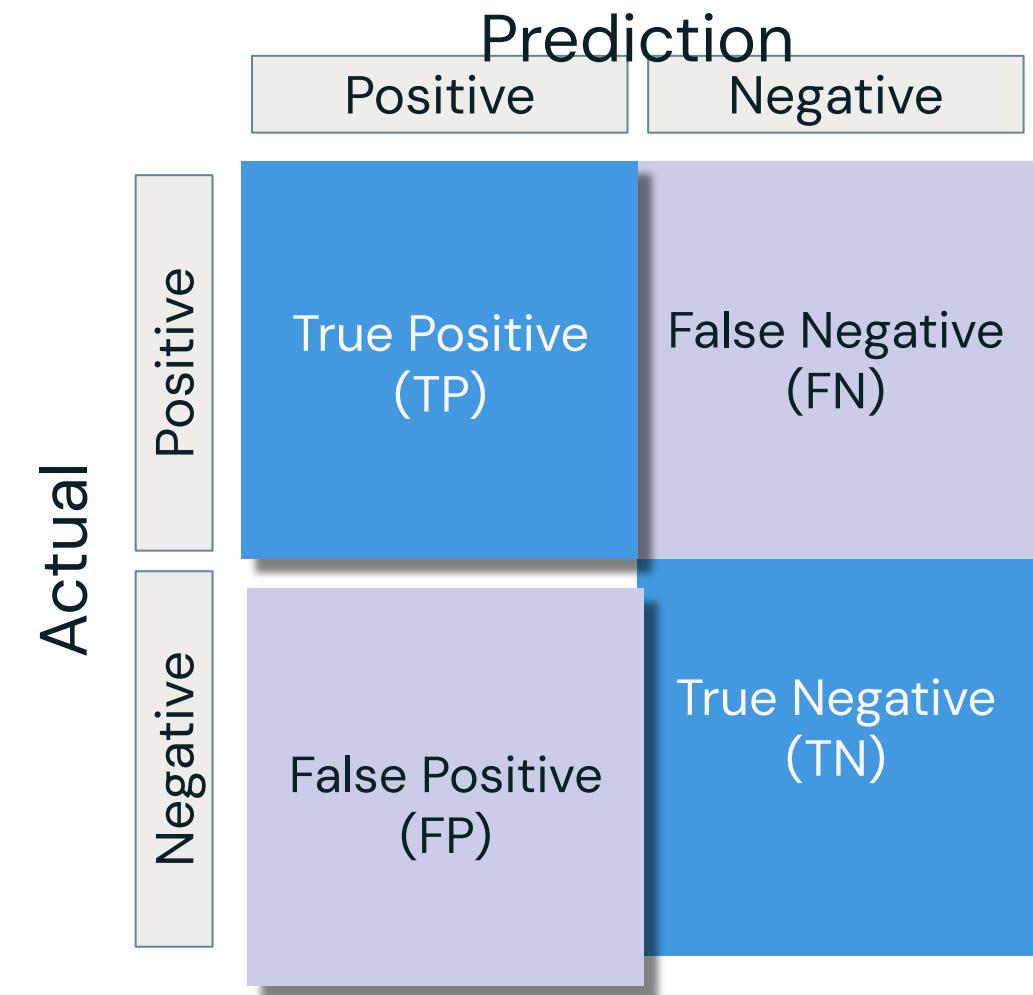
```
from sklearn.metrics import accuracy_score  
accuracy = accuracy_score(y_true, y_pred)
```



Review of Common Metrics

Supervised Learning – Classification

Metric	Description
Accuracy	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.
Precision	Measures how many predicted positives are actually correct. Important when false positives are costly.
Recall (Sensitivity)	
F1 Score	



Confusion Matrix

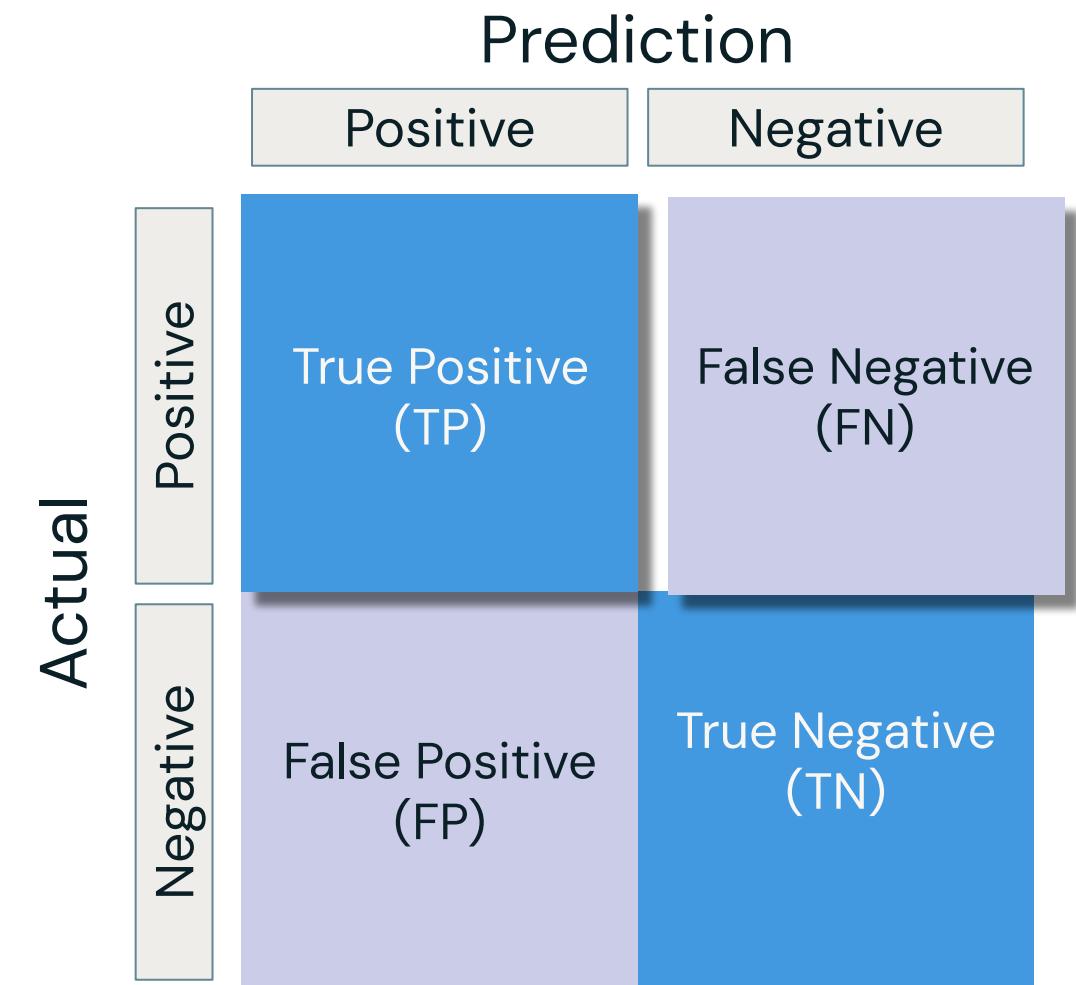
```
from sklearn.metrics import precision_score  
precision = precision_score(y_true, y_pred)
```



Review of Common Metrics

Supervised Learning – Classification

Metric	Description
Accuracy	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.
Precision	Measures how many predicted positives are actually correct. Important when false positives are costly.
Recall (Sensitivity)	Measures how many actual positives were correctly identified. Important when false negatives are costly.
F1 Score	



Confusion Matrix

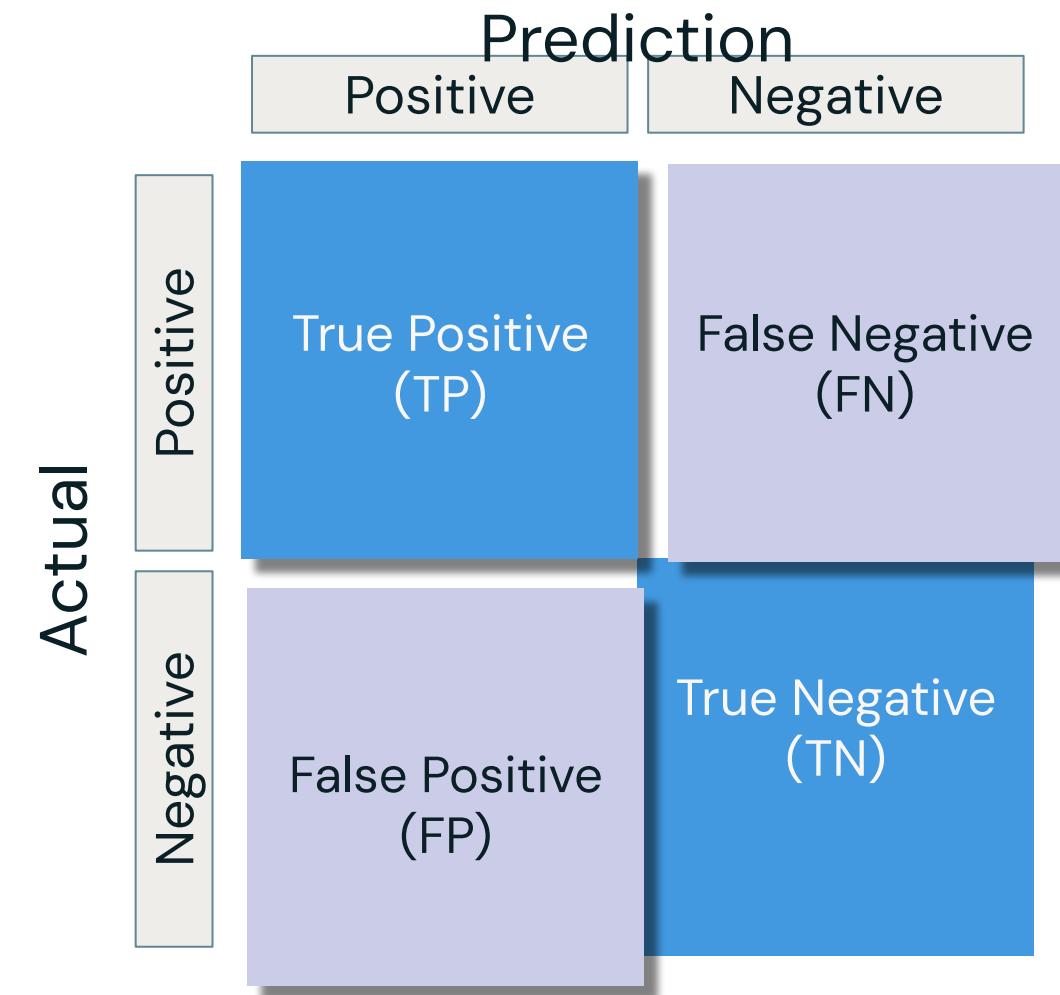
```
from sklearn.metrics import recall_score  
recall = recall_score(y_true, y_pred)
```



Review of Common Metrics

Supervised Learning – Classification

Metric	Description
Accuracy	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.
Precision	Measures how many predicted positives are actually correct. Important when false positives are costly.
Recall (Sensitivity)	Measures how many actual positives were correctly identified. Important when false negatives are costly.
F1 Score	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.



Confusion Matrix

```
from sklearn.metrics import f1_score  
f1 = f1_score(y_true, y_pred)
```



Precision, Recall, F1-Score

Which one to choose?

Precision

Suitable when minimizing false positives is crucial, even at the expense of missing some positive instances.

Example: In a **spam email classifier**, high precision ensures that emails marked as spam are indeed spam, reducing the chances of incorrectly classifying important emails as spam.

Recall

Appropriate when capturing all positive instances is vital, even if it means tolerating some false positives.

Example: In a **medical diagnosis system**, high recall ensures identifying all patients with a certain disease, even if it leads to a few false alarms.

F1 Score

Useful when seeking a balance between precision and recall.

Example: In a search engine **ranking algorithm**, F1 score might be used to evaluate how well the algorithm retrieves relevant documents while minimizing irrelevant ones.

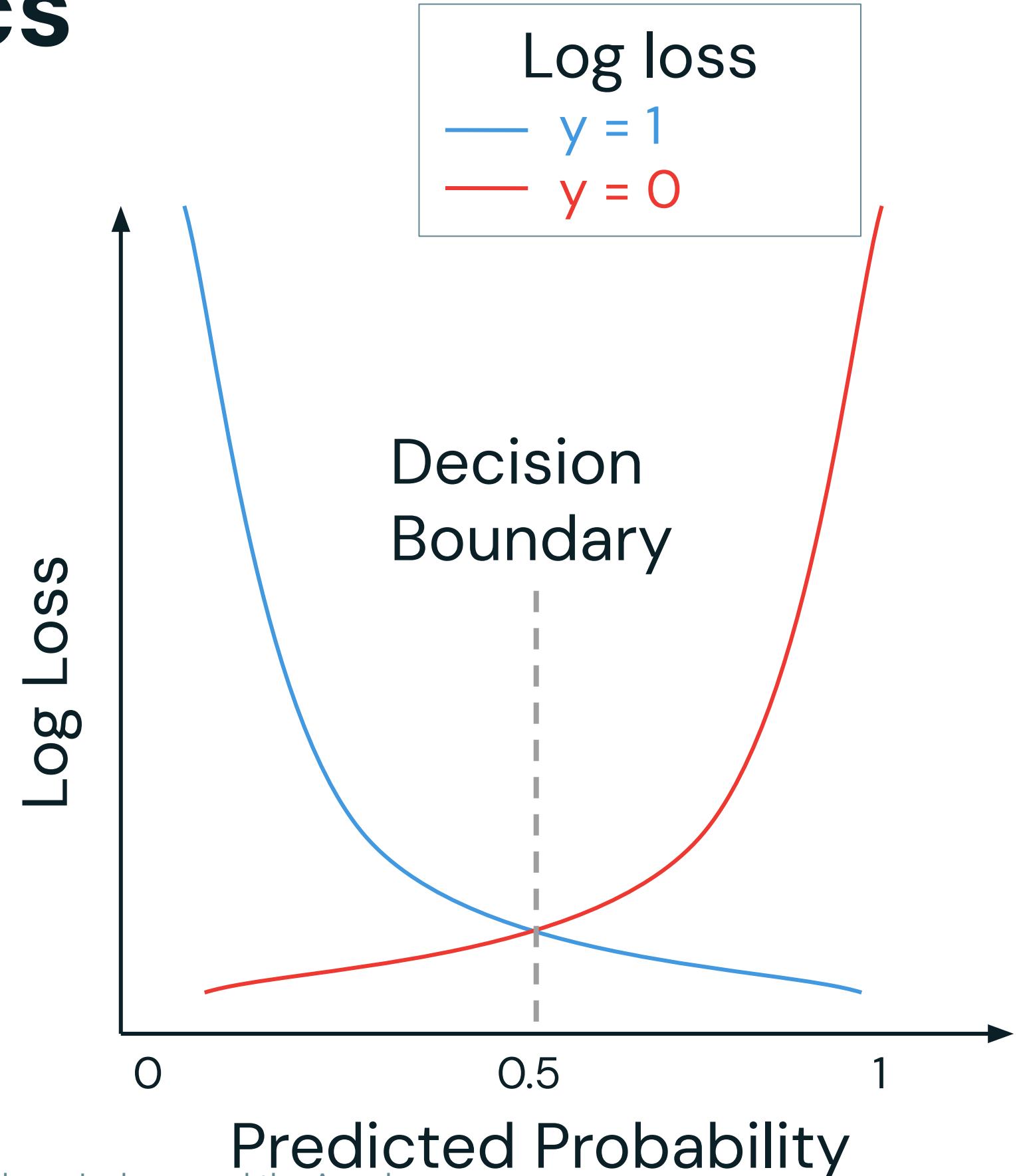


Review of Common Metrics

Supervised Learning – Classification

Metric	Description
Log loss (Cross-Entropy Loss)	Measures by comparing predicted probabilities to the actual class labels. Penalizes confident but incorrect predictions heavily. Lower values indicate better calibrated models.
ROC/AUC	

```
from sklearn.metrics import log_loss  
ll_value = log_loss(y_true, y_pred)
```

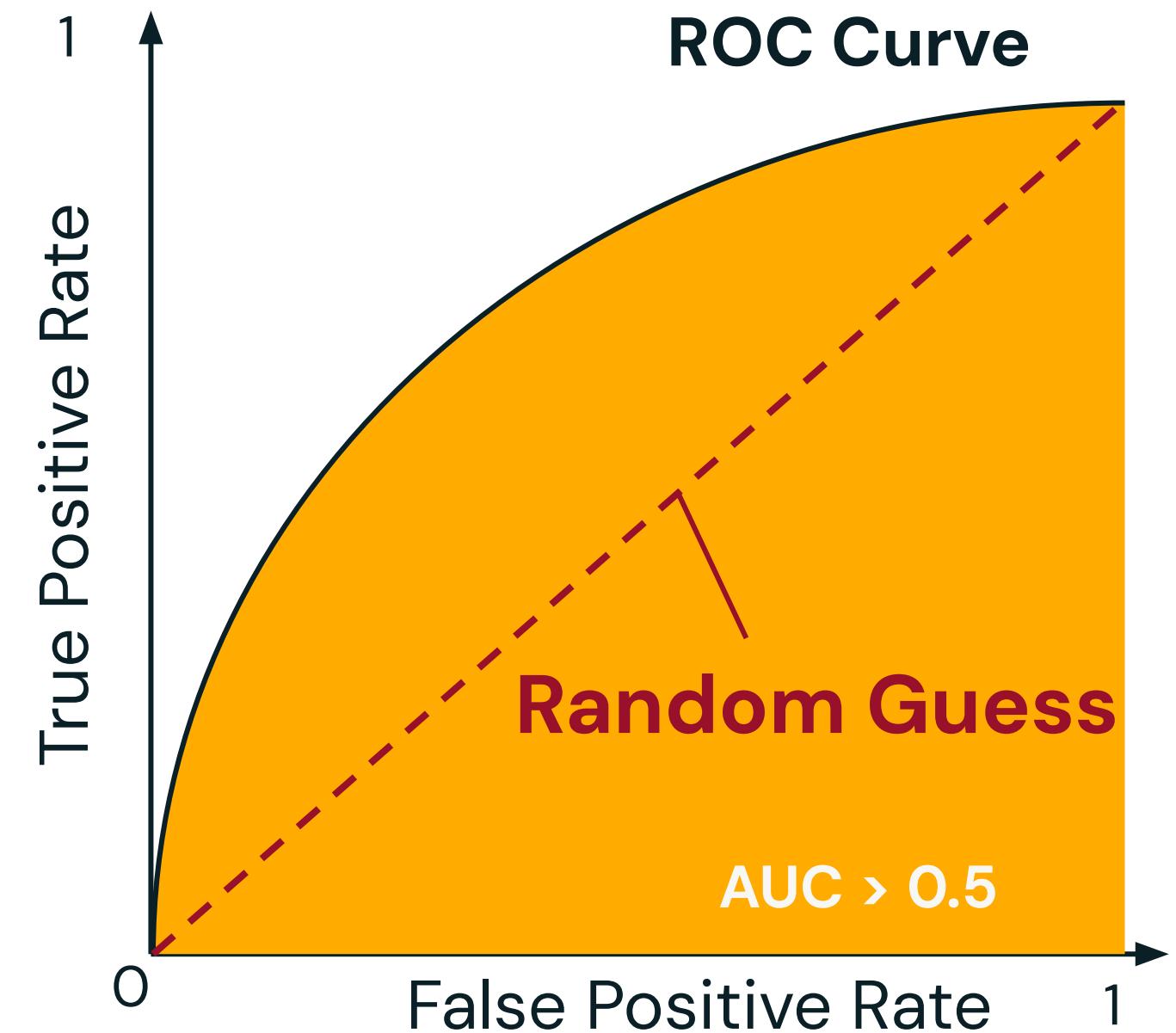


Review of Common Metrics

Supervised Learning – Classification

Metric	Description
Log loss (Cross-Entropy Loss)	Measures by comparing predicted probabilities to the actual class labels. Penalizes confident but incorrect predictions heavily. Lower values indicate better calibrated models.
ROC/AUC	Measures the model's ability to distinguish between classes across different thresholds. AUC (Area Under the Curve) closer to 1 indicates a better model.

```
from sklearn.metrics import roc_auc_score
auc_score = roc_auc_score(y_true, y_pred)
```

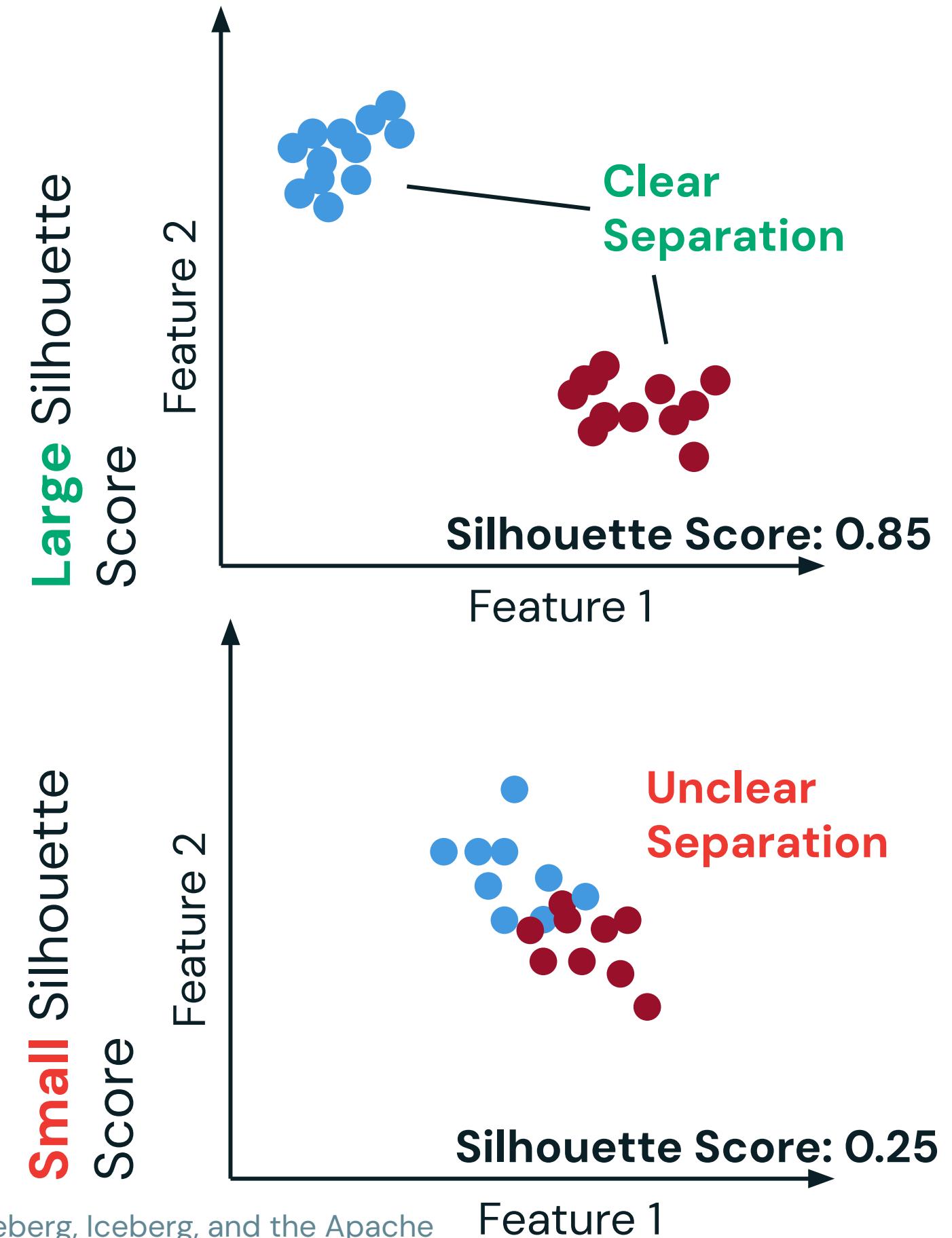


Review of Common Metrics

Unsupervised Learning - Clustering

Metric	Description
Silhouette Score	Measures how well-defined clusters are. Higher values indicate better separation.
Elbow Method	

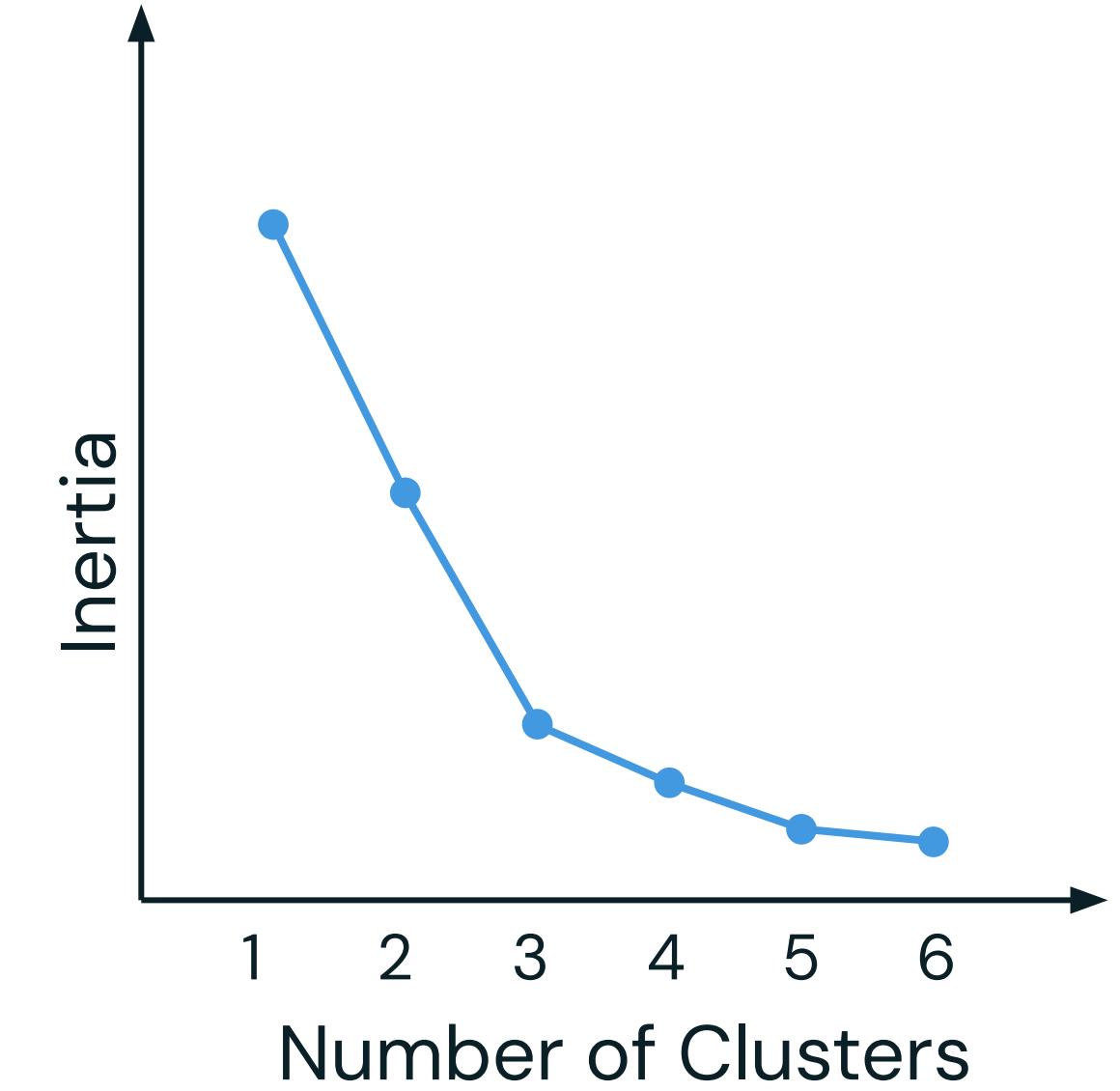
```
from sklearn.metrics import silhouette_score
From sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
Labels = kmeans.fit_predict(X)
s_score = silhouette_score(X, labels)
```



Review of Common Metrics

Unsupervised Learning – Clustering

Metric	Description
Silhouette Score	Measures how well-defined clusters are. Higher values indicate better separation.
Elbow Method	Plots variance within clusters vs the number of clusters to find the elbow point where adding more clusters doesn't improve separation.



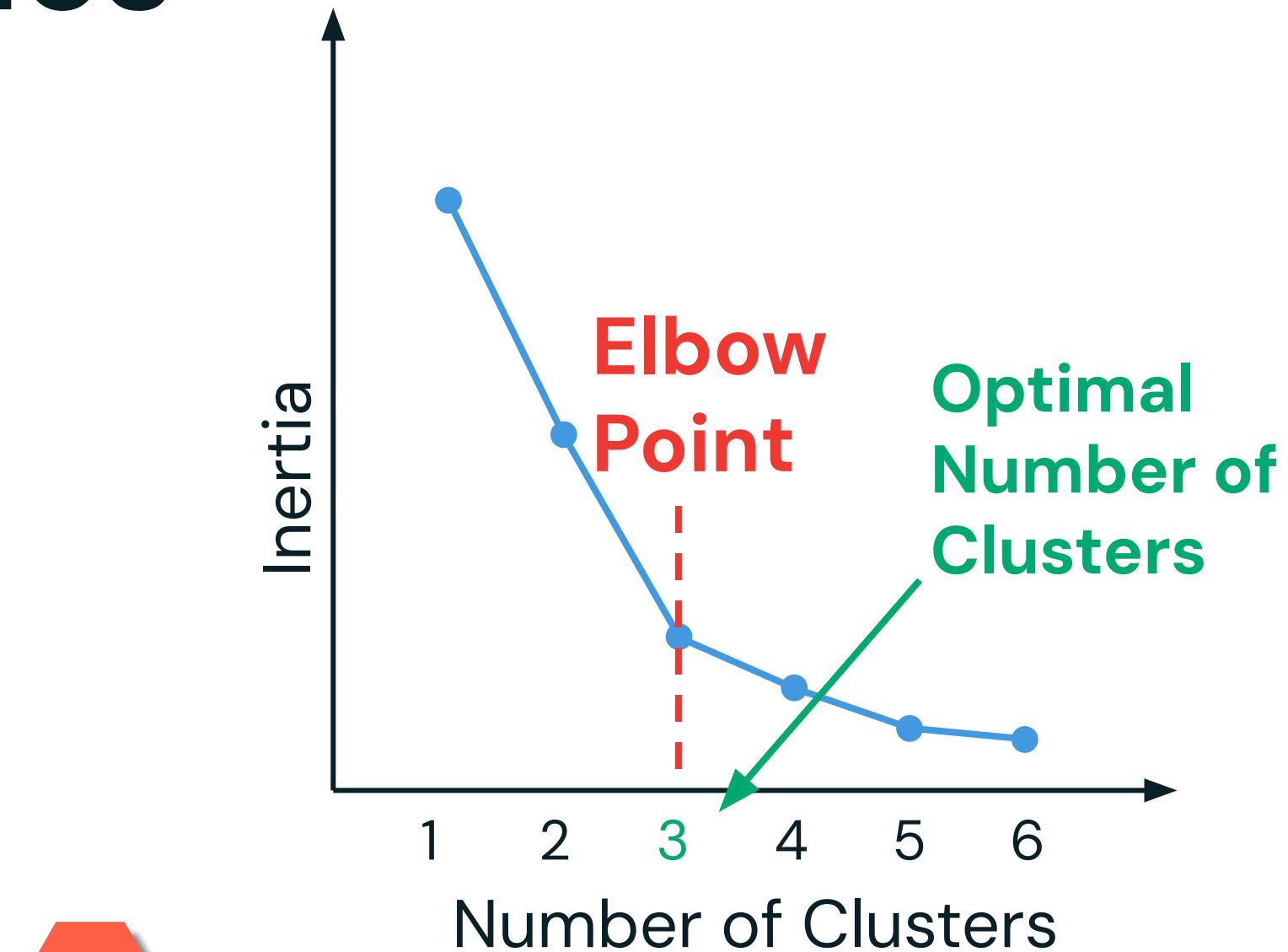
Review of Common Metrics

Unsupervised Learning – Clustering

Metric	Description
Silhouette Score	Measures how well-defined clusters are. Higher values indicate better separation.
Elbow Method	Plots variance within clusters vs the number of clusters to find the elbow point where adding more clusters doesn't improve separation.

1

```
from sklearn.metrics import silhouette_score
From sklearn.cluster import KMeans
k_values = range(1, 7)
```



```
inertia_values = [KMeans(n_clusters=k,
random_state=42, n_init=10).fit(X).inertia_ for k
in k_values]

knee_locator = KneeLocator(k_values,
inertia_values, curve="convex",
direction="decreasing")

optimal_k = knee_locator.elbow
```

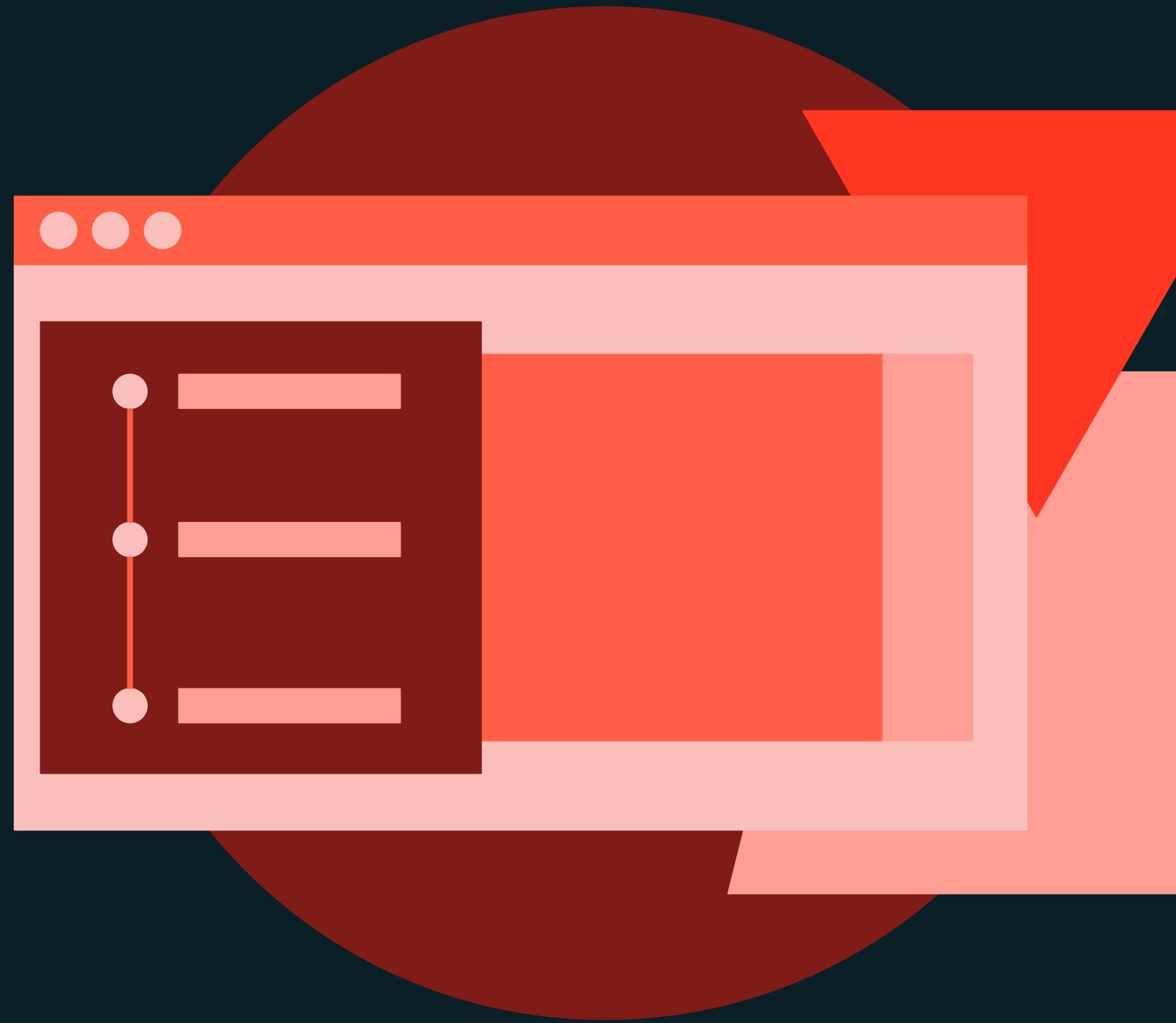




Model Development Workflow

DEMONSTRATION

Supervised Learning



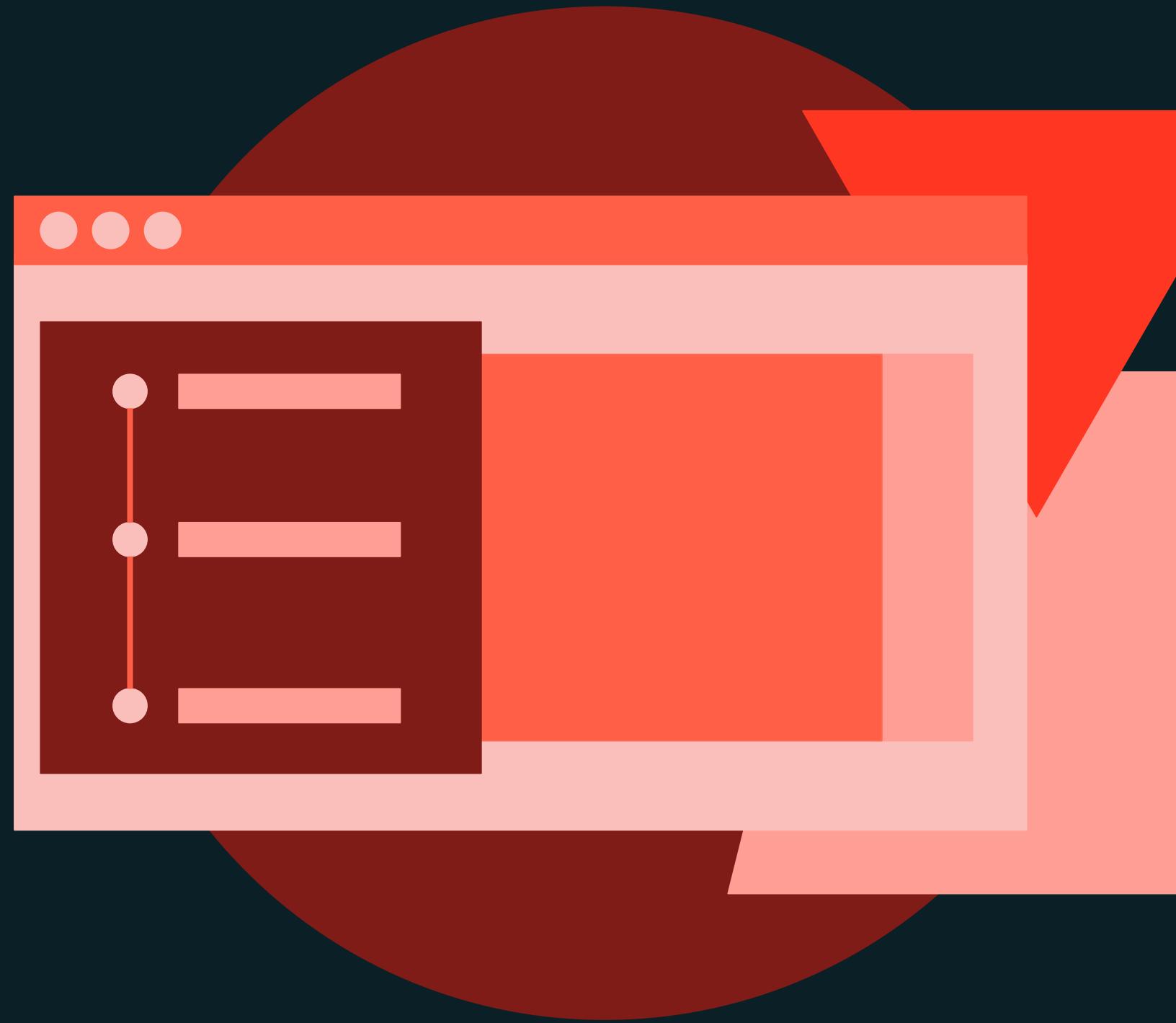
© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).



Model Development Workflow

DEMONSTRATION

Unsupervised Learning



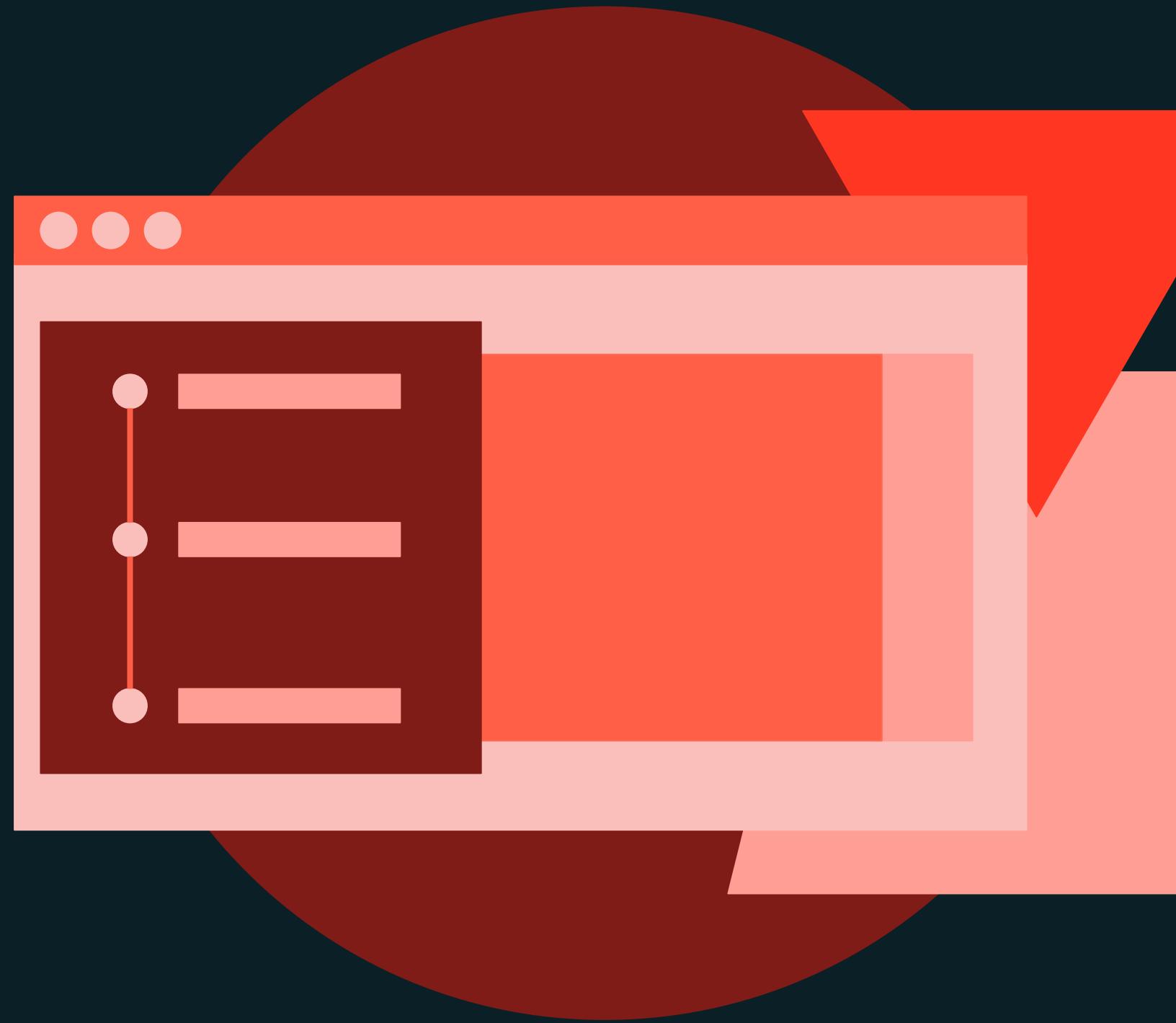
© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).



Model Development Workflow

DEMONSTRATION

Model Tracking with MLflow





Model Development Workflow

LAB EXERCISE

Model Development Tracking with MLflow



Appendix A

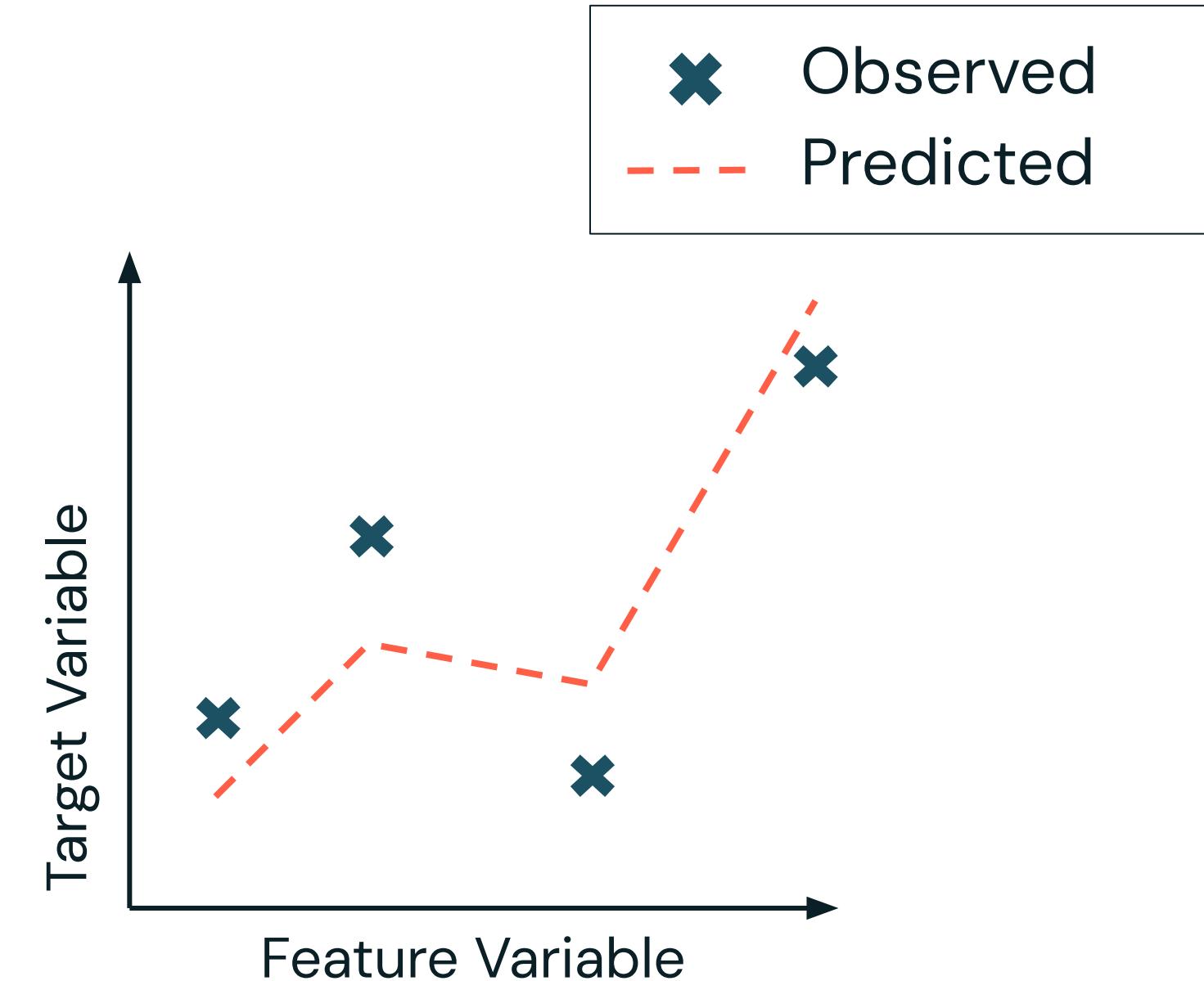


© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

Review of Some Common Metrics

Supervised Learning – Regression

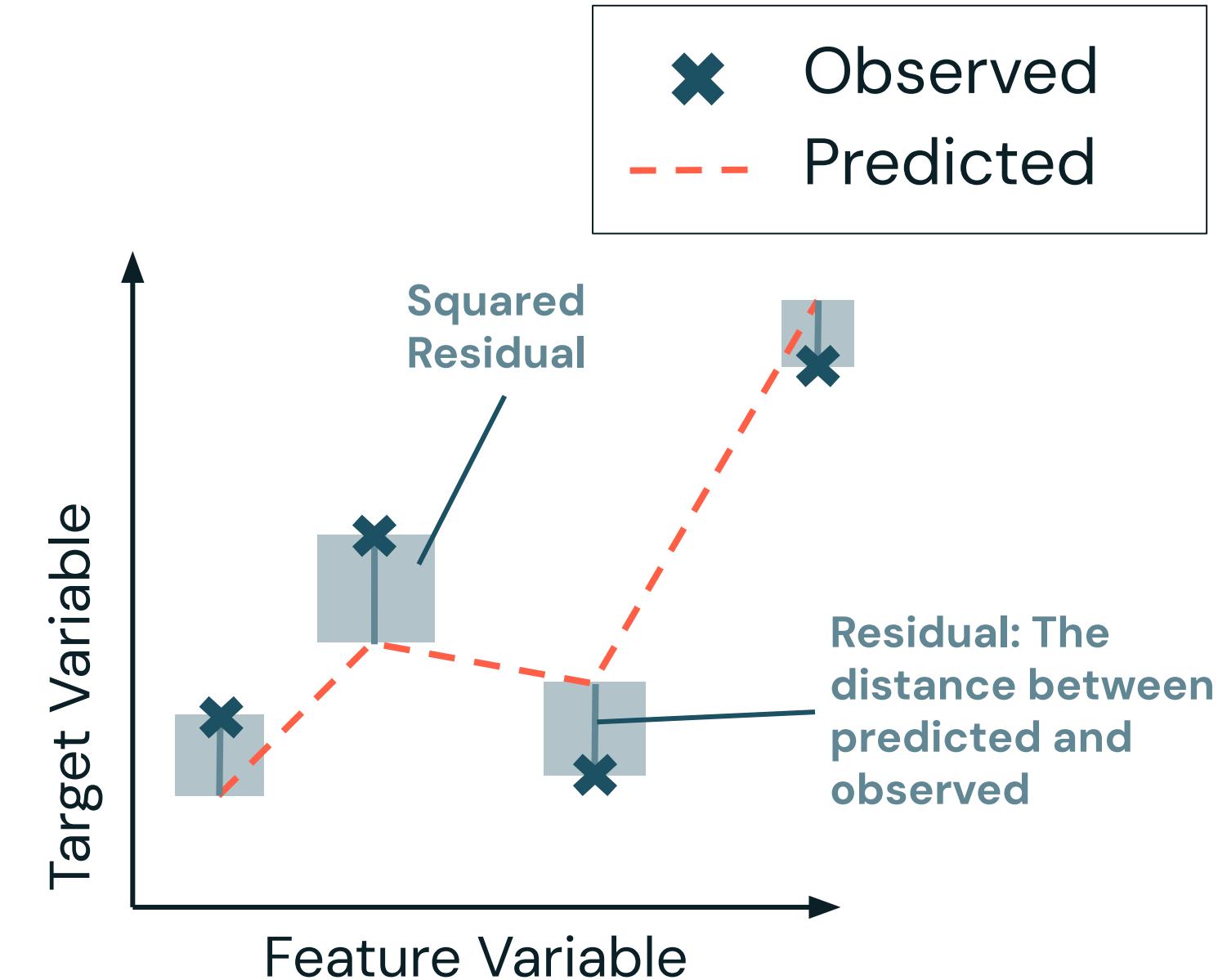
Metric	Description
R^2	
Mean Absolute Error	
Mean Squared Error	
Root Mean Squared Error	



Review of Some Common Metrics

Supervised Learning – Regression

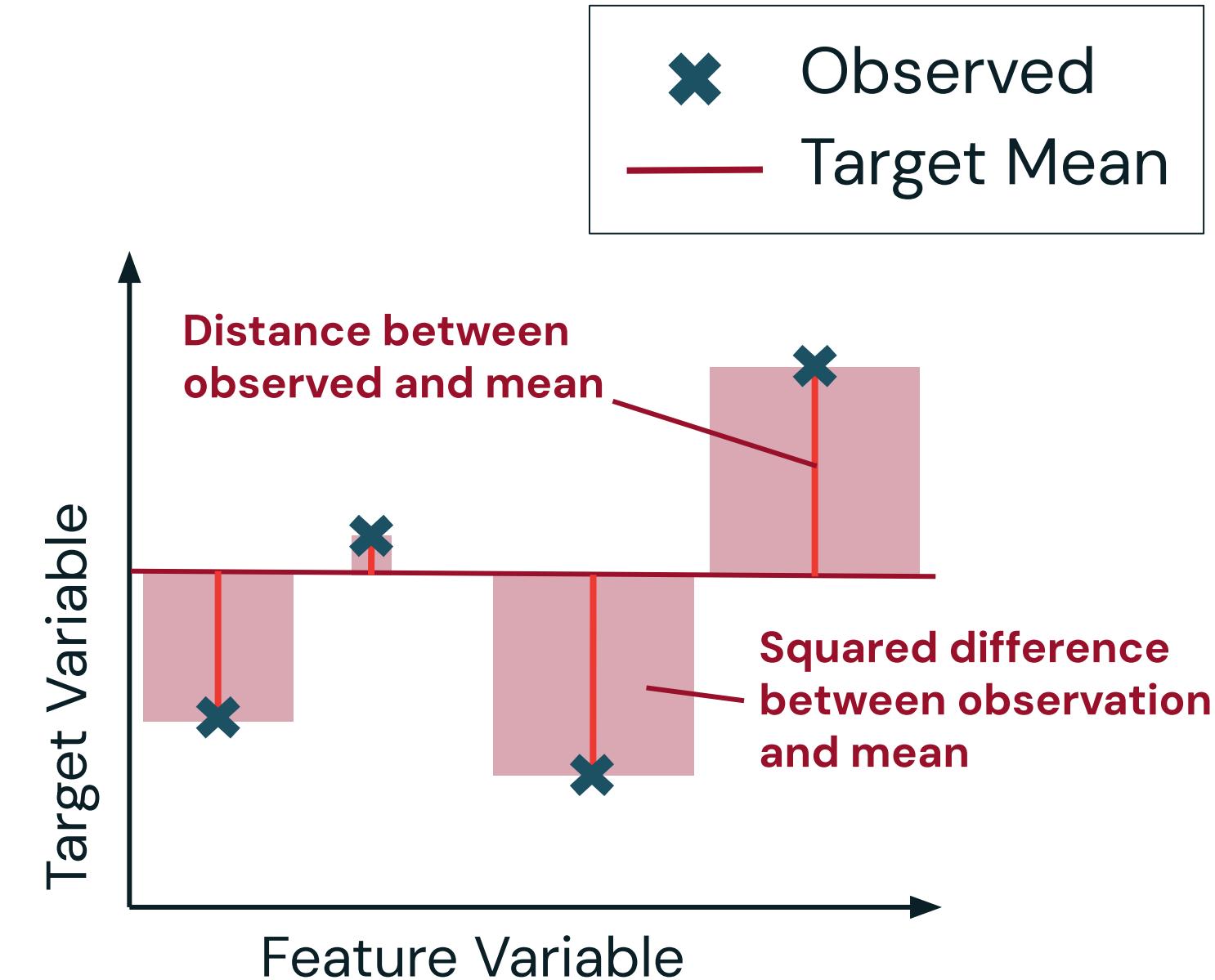
Metric	Description
R^2	Measures the proportion of variance explained by the model. Higher values indicate a better fit.
Mean Absolute Error	
Mean Squared Error	
Root Mean Squared Error	



Review of Some Common Metrics

Supervised Learning – Regression

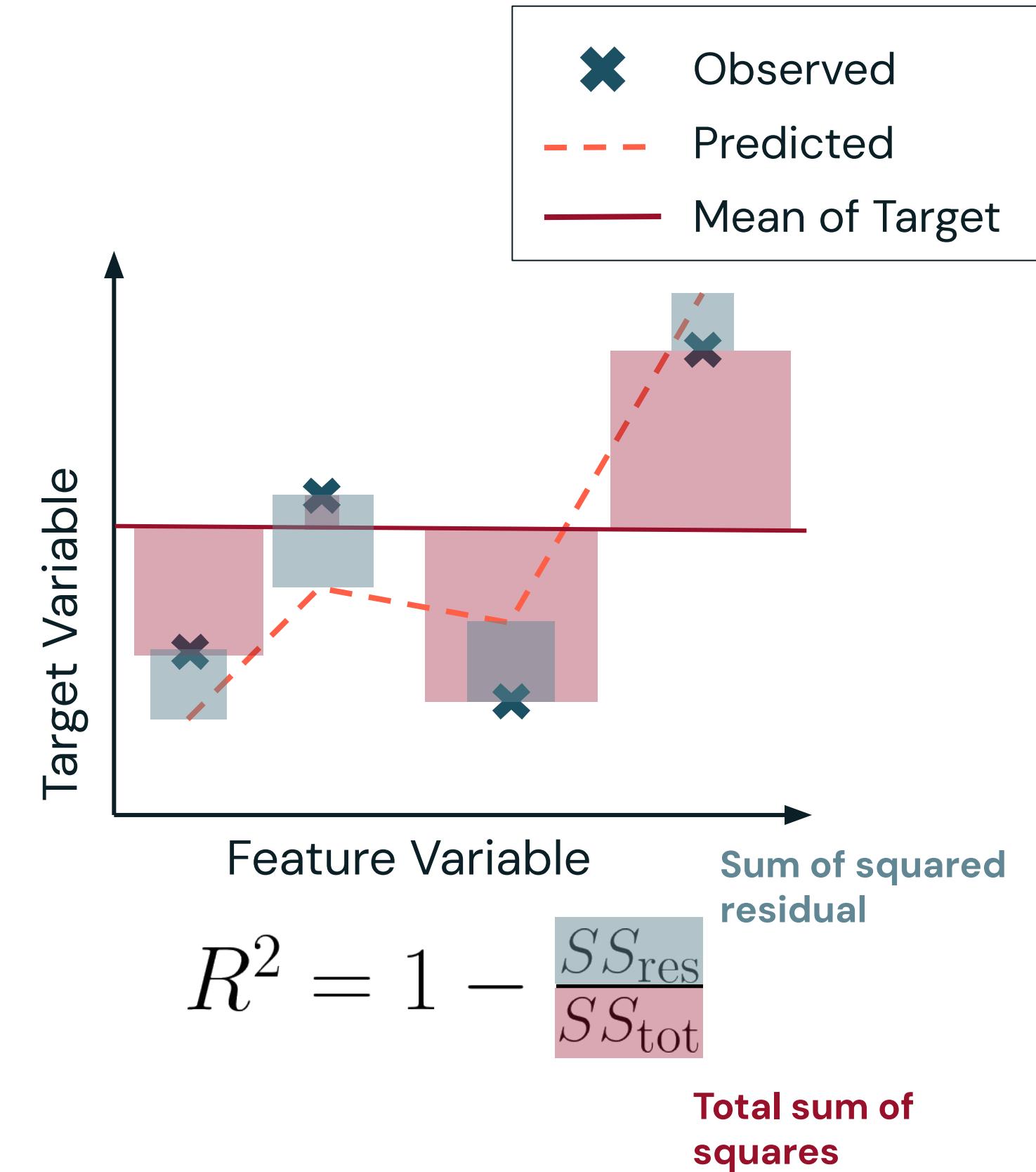
Metric	Description
R^2	Measures the proportion of variance explained by the model. Higher values indicate a better fit.
Mean Absolute Error	
Mean Squared Error	
Root Mean Squared Error	



Review of Some Common Metrics

Supervised Learning – Regression

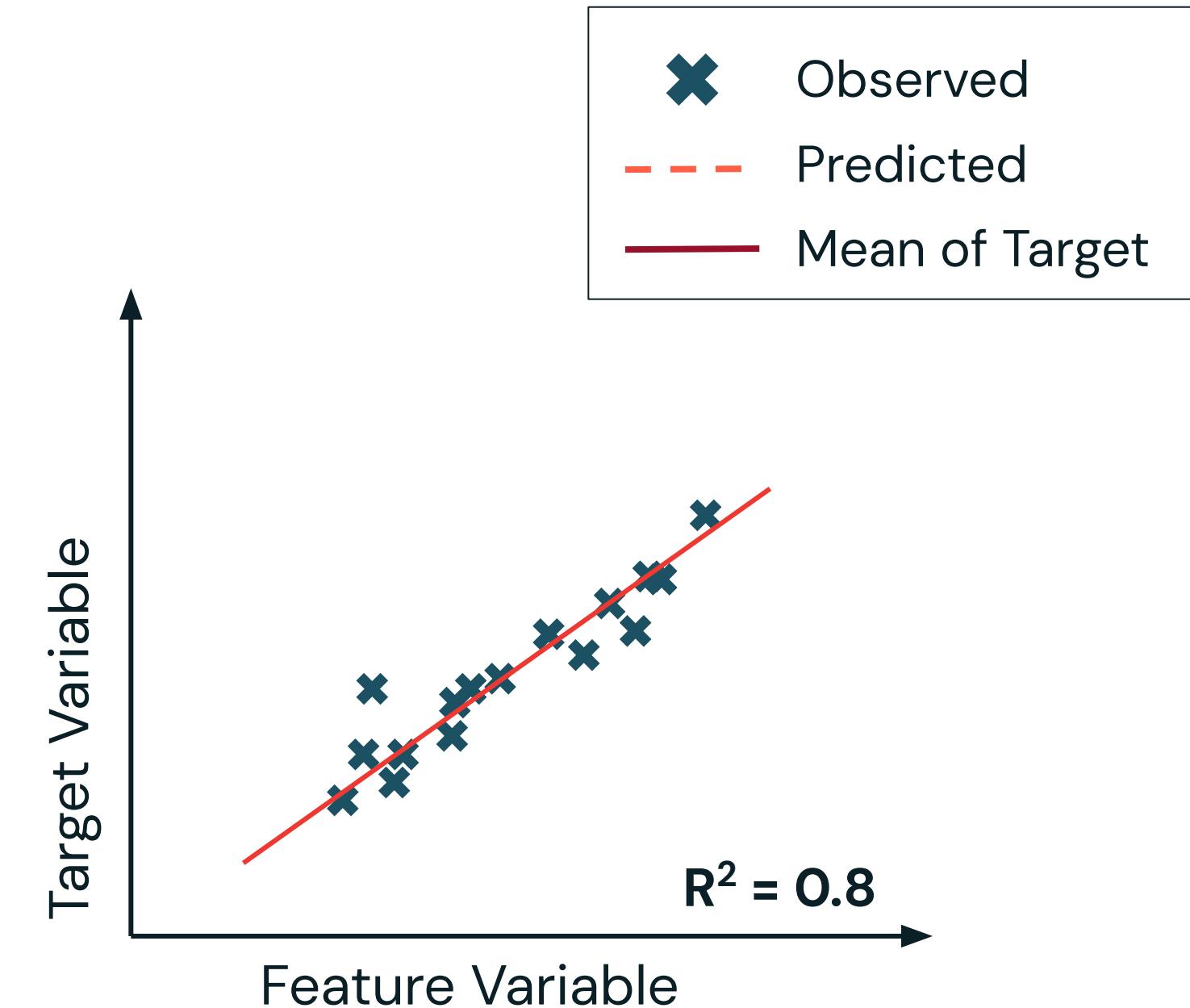
Metric	Description
R ²	Measures the proportion of variance explained by the model. Higher values indicate a better fit.
Mean Absolute Error	
Mean Squared Error	
Root Mean Squared Error	



Review of Some Common Metrics

Supervised Learning – Regression

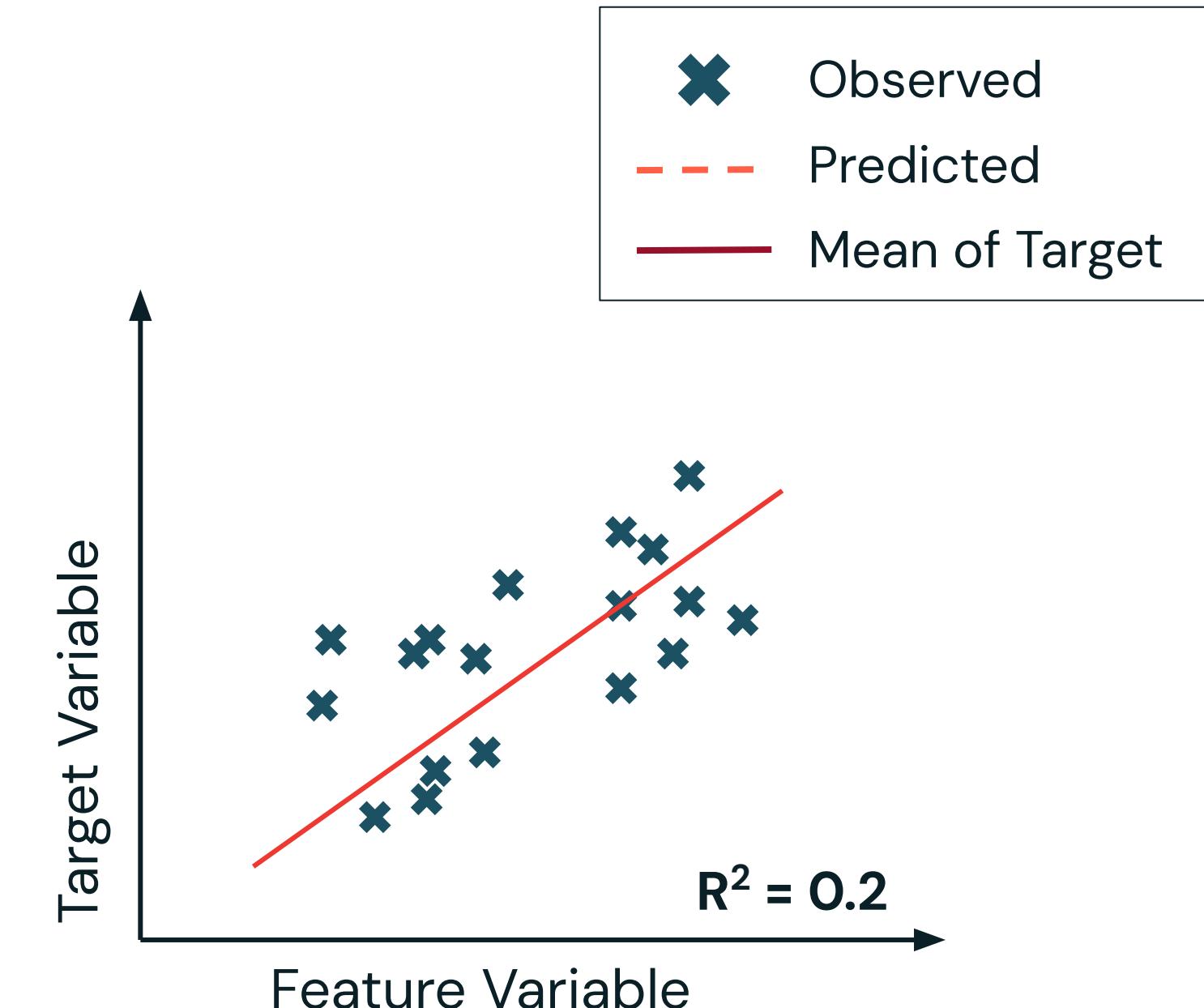
Metric	Description
R^2	Measures the proportion of variance explained by the model. Higher values indicate a better fit.
Mean Absolute Error	
Mean Squared Error	
Root Mean Squared Error	



Review of Some Common Metrics

Supervised Learning – Regression

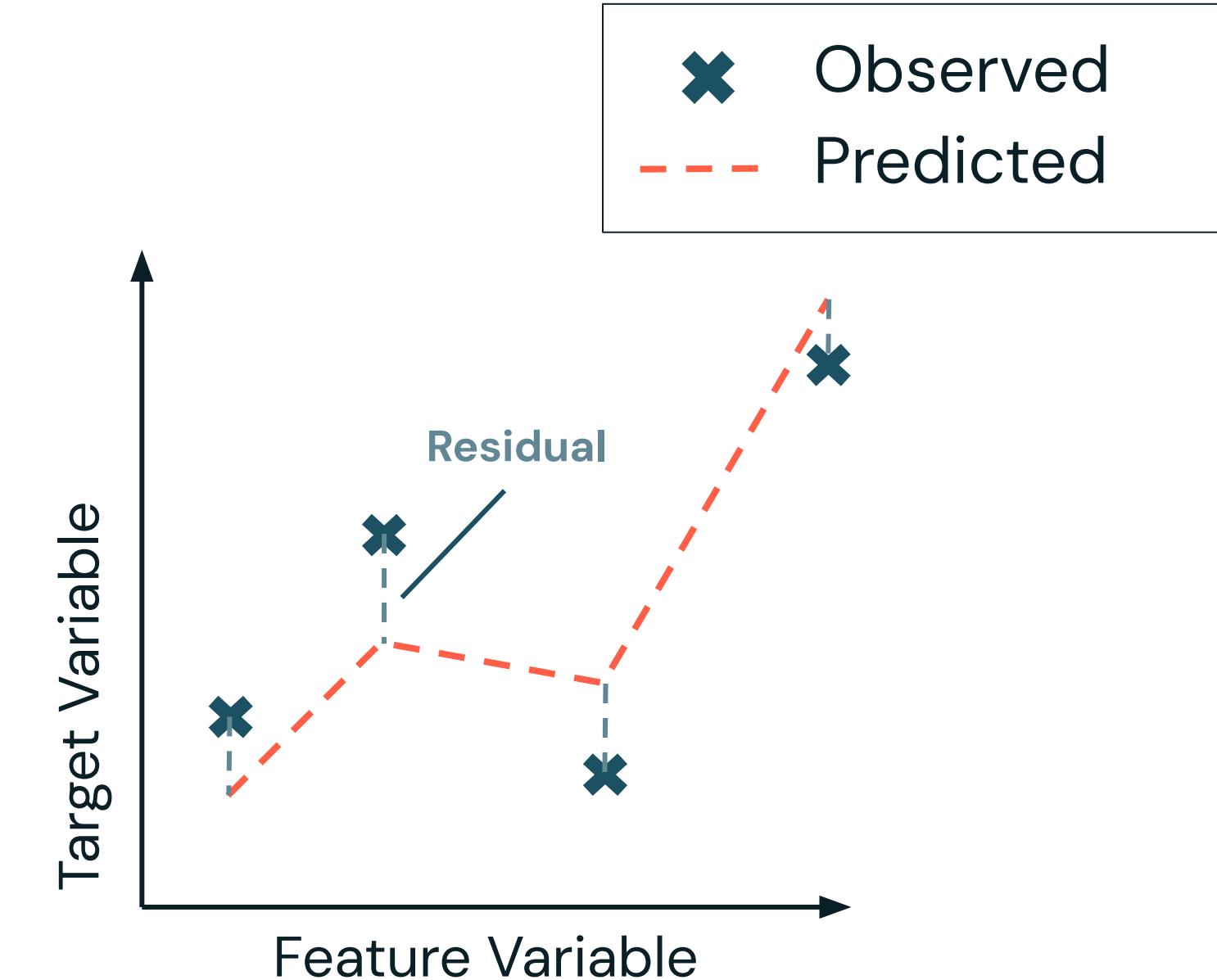
Metric	Description
R^2	Measures the proportion of variance explained by the model. Higher values indicate a better fit.
Mean Absolute Error	
Mean Squared Error	
Root Mean Squared Error	



Review of Some Common Metrics

Supervised Learning – Regression

Metric	Description
R ²	Measures the proportion of variance explained by the model. Higher values indicate a better fit.
Mean Absolute Error	Computes the average of the absolute difference between actual and predicted values. Less sensitive to outliers.
Mean Squared Error	
Root Mean Squared Error	



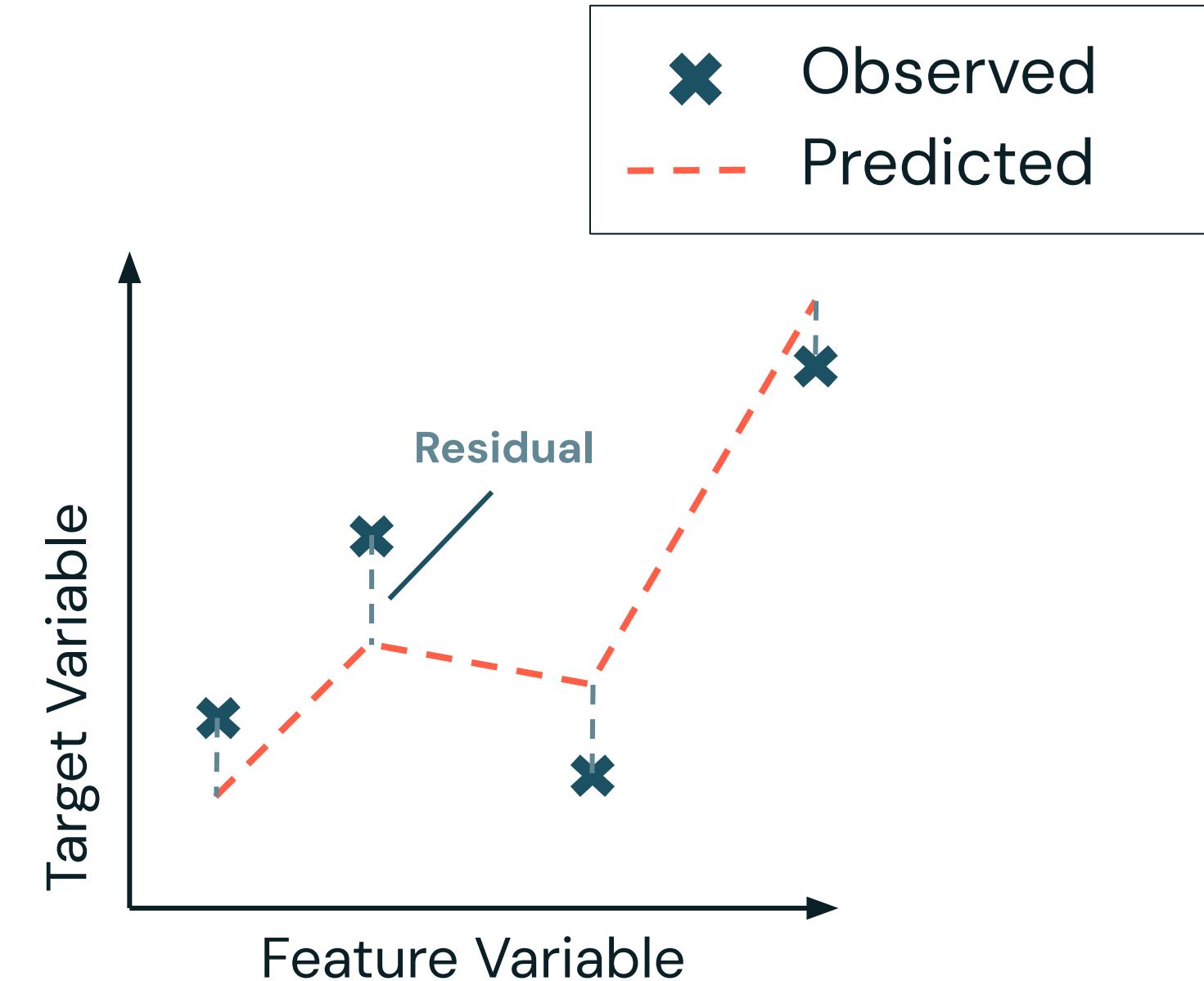
$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$



Review of Some Common Metrics

Supervised Learning – Regression

Metric	Description
R ²	Measures the proportion of variance explained by the model. Higher values indicate a better fit.
Mean Absolute Error	Computes the average of the absolute difference between actual and predicted values. Less sensitive to outliers.
Mean Squared Error	Penalizes larger errors more than MAE due to squaring. Common for Optimization.
Root Mean Squared Error	



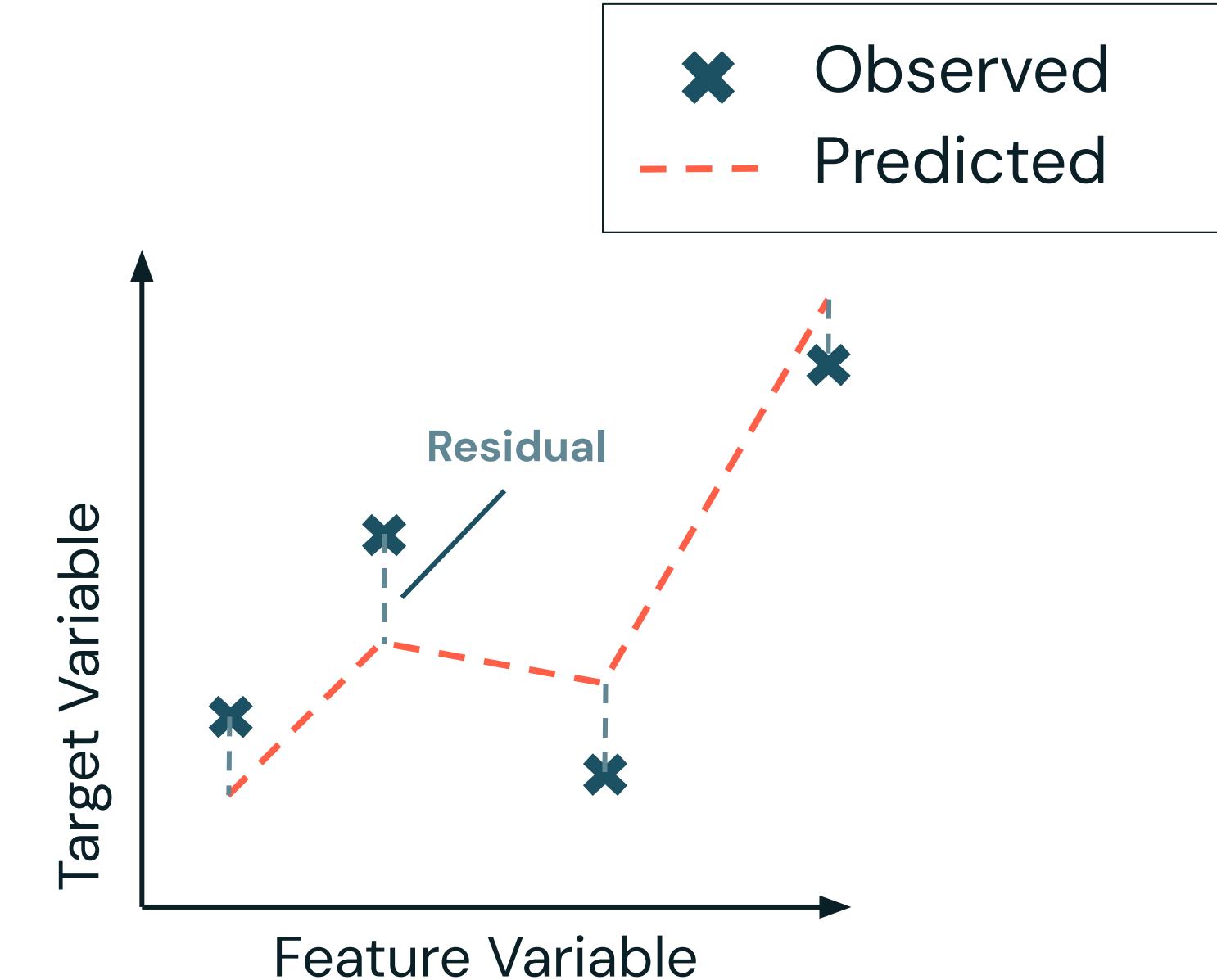
$$MSE = \frac{1}{n} \sum_{i=1}^n (\text{Residual})^2$$



Review of Some Common Metrics

Supervised Learning – Regression

Metric	Description
R ²	Measures the proportion of variance explained by the model. Higher values indicate a better fit.
Mean Absolute Error	Computes the average of the absolute difference between actual and predicted values. Less sensitive to outliers.
Mean Squared Error	Penalizes larger errors more than MAE due to squaring. Common for Optimization.
Root Mean Squared Error	Similar to MSE but takes the square root, making it more interpretable to the original units.



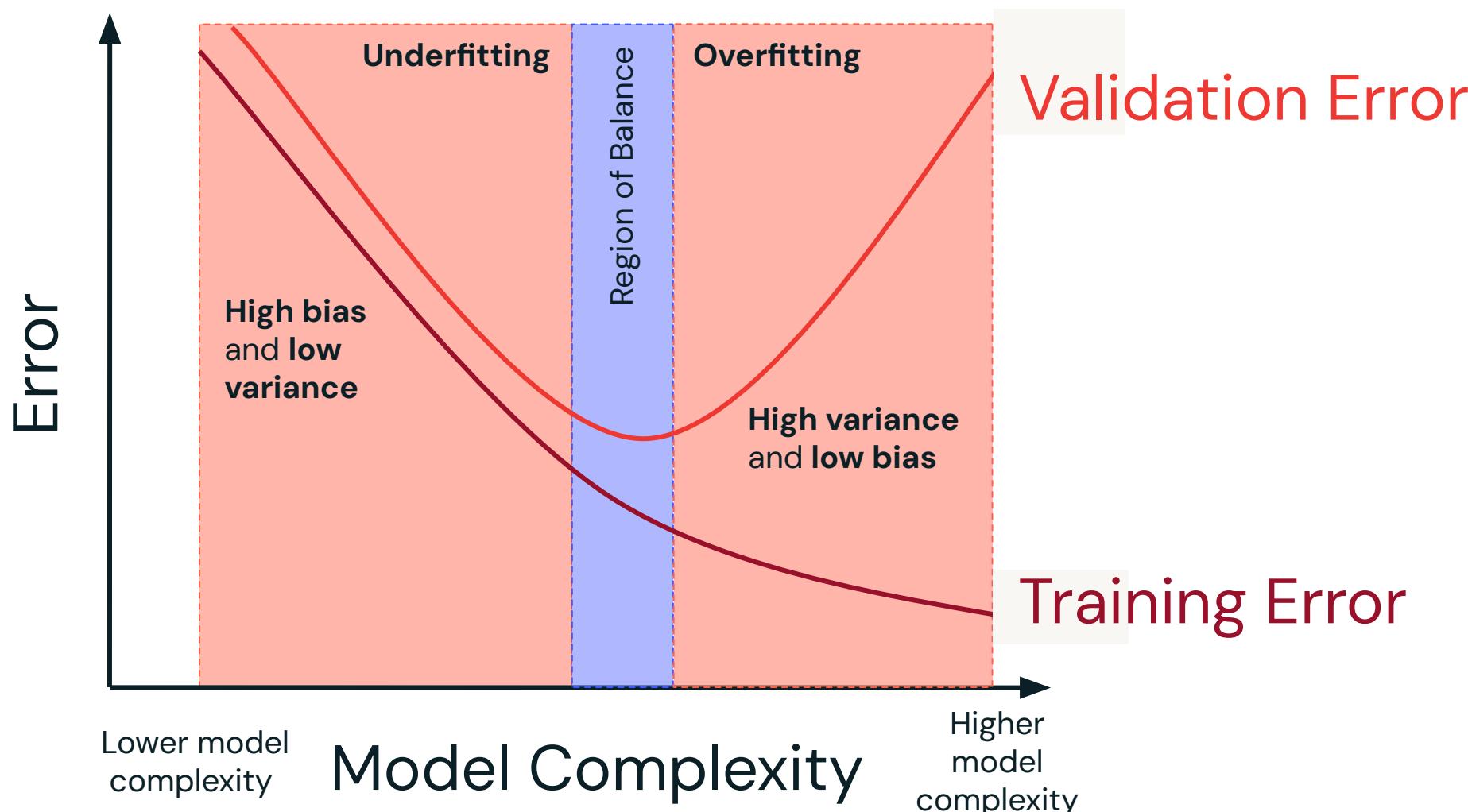
$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Residual



Bias-Variance Tradeoff

Underfitting vs Overfitting

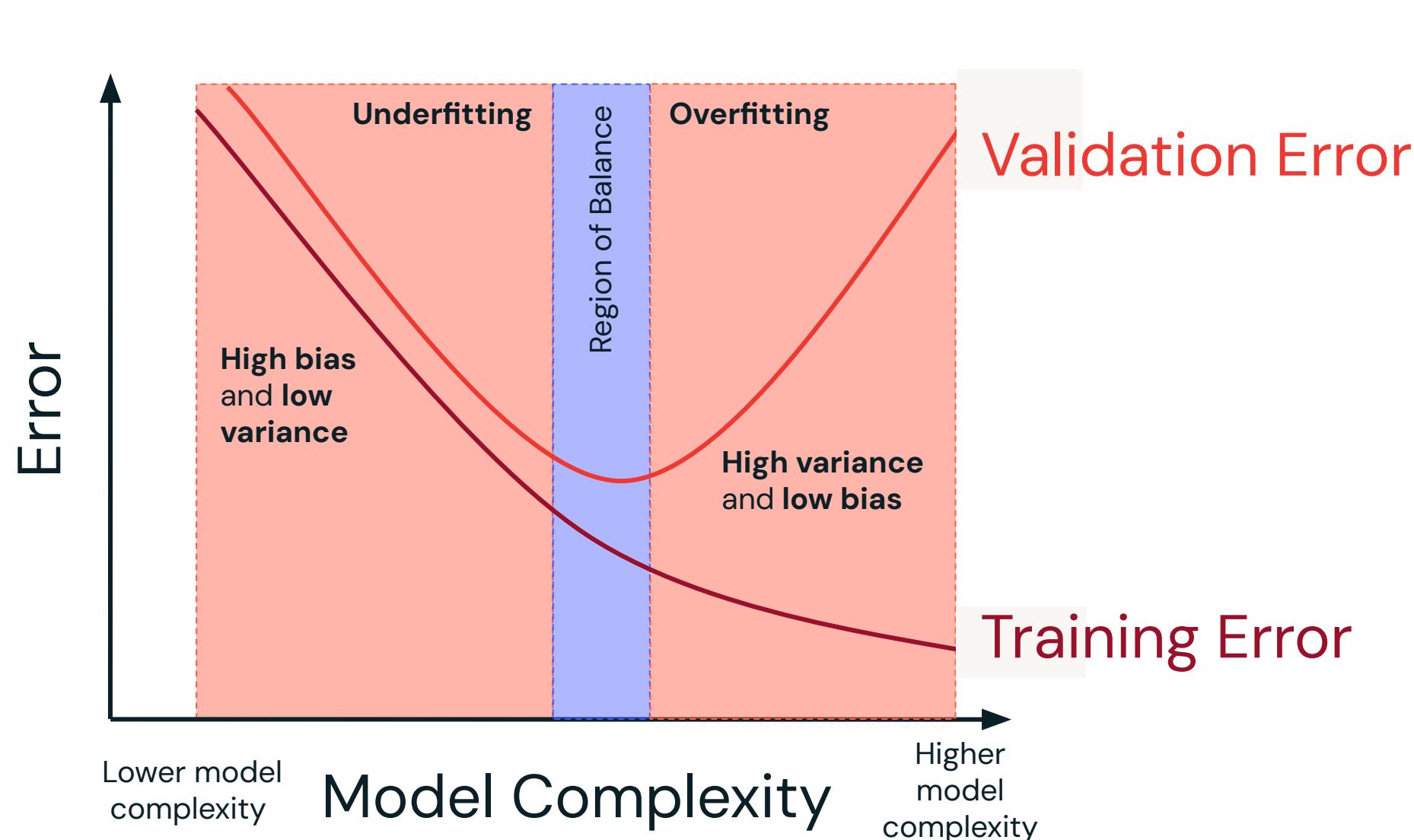


	Description of Error	Mitigation Strategies
Bias	Error due to overly simplistic assumptions in the model.	Add more features or increasing model complexity.
Variance	Error due to sensitivity to small fluctuations in the training data.	Add more data to help with generalizability. Add regularization, reduce model complexity, or using cross-validation.



Bias-Variance Tradeoff

Underfitting vs Overfitting



Reason	Increase/Decrease
High Bias	Model is too simple. ▲ FP & ▲ FN
High Variance	Model does not generalize well, leading to precision-recall tradeoffs. ▲ FP & ▼ FN or ▼ FP & ▲ FN



Review of Some Common Metrics

Supervised Learning – Classification

Metric	Description
Accuracy	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.
Precision	
Recall (Sensitivity)	
F1 Score	

		Prediction	
		Positive	Negative
Actual	Positive	True Positive (TP) 4	False Negative (FN) 1
	Negative	False Positive (FP) 1	True Negative (TN) 4

$$\frac{TP+TN}{TP+TN+FP+FN} = \frac{4}{5}$$



Review of Some Common Metrics

Supervised Learning – Classification

Metric	Description
Accuracy	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.
Precision	Measures how many predicted positives are actually correct. Important when false positives are costly.
Recall (Sensitivity)	
F1 Score	

		Prediction	
		Positive	Negative
Actual	Positive	True Positive (TP) 4	False Negative (FN) 1
	Negative	False Positive (FP) 1	True Negative (TN) 4

$$\frac{TP}{TP+FP} = \frac{4}{5}$$



Review of Some Common Metrics

Supervised Learning – Classification

Metric	Description
Accuracy	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.
Precision	Measures how many predicted positives are actually correct. Important when false positives are costly.
Recall (Sensitivity)	Measures how many actual positives were correctly identified. Important when false negatives are costly.
F1 Score	

		Prediction	
		Positive	Negative
Actual	Positive	True Positive (TP) 4	False Negative (FN) 1
	Negative	False Positive (FP) 1	True Negative (TN) 4

$$\frac{TP}{TP+FN} = \frac{4}{5}$$



Review of Some Common Metrics

Supervised Learning - Classification

Metric	Description
Accuracy	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.
Precision	Measures how many predicted positives are actually correct. Important when false positives are costly.
Recall (Sensitivity)	Measures how many actual positives were correctly identified. Important when false negatives are costly.
F1 Score	Measures the proportion of correctly classified instances. It can be misleading for imbalance datasets.

		Prediction	
		Positive	Negative
Actual	Positive	True Positive (TP) 4	False Negative (FN) 1
	Negative	False Positive (FP) 1	True Negative (TN) 4

$$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{4}{5}$$

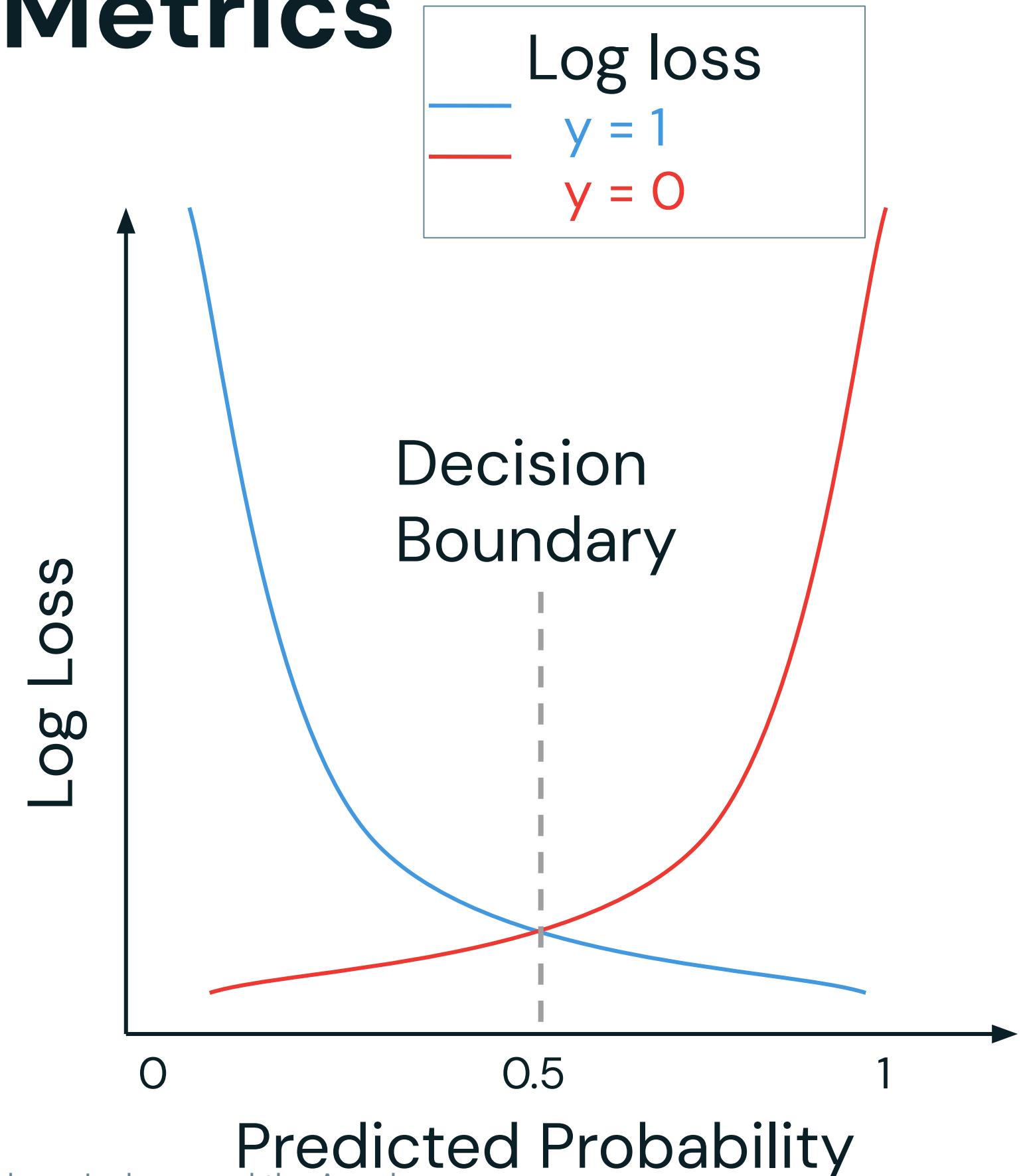


Review of Some Common Metrics

Supervised Learning – Classification

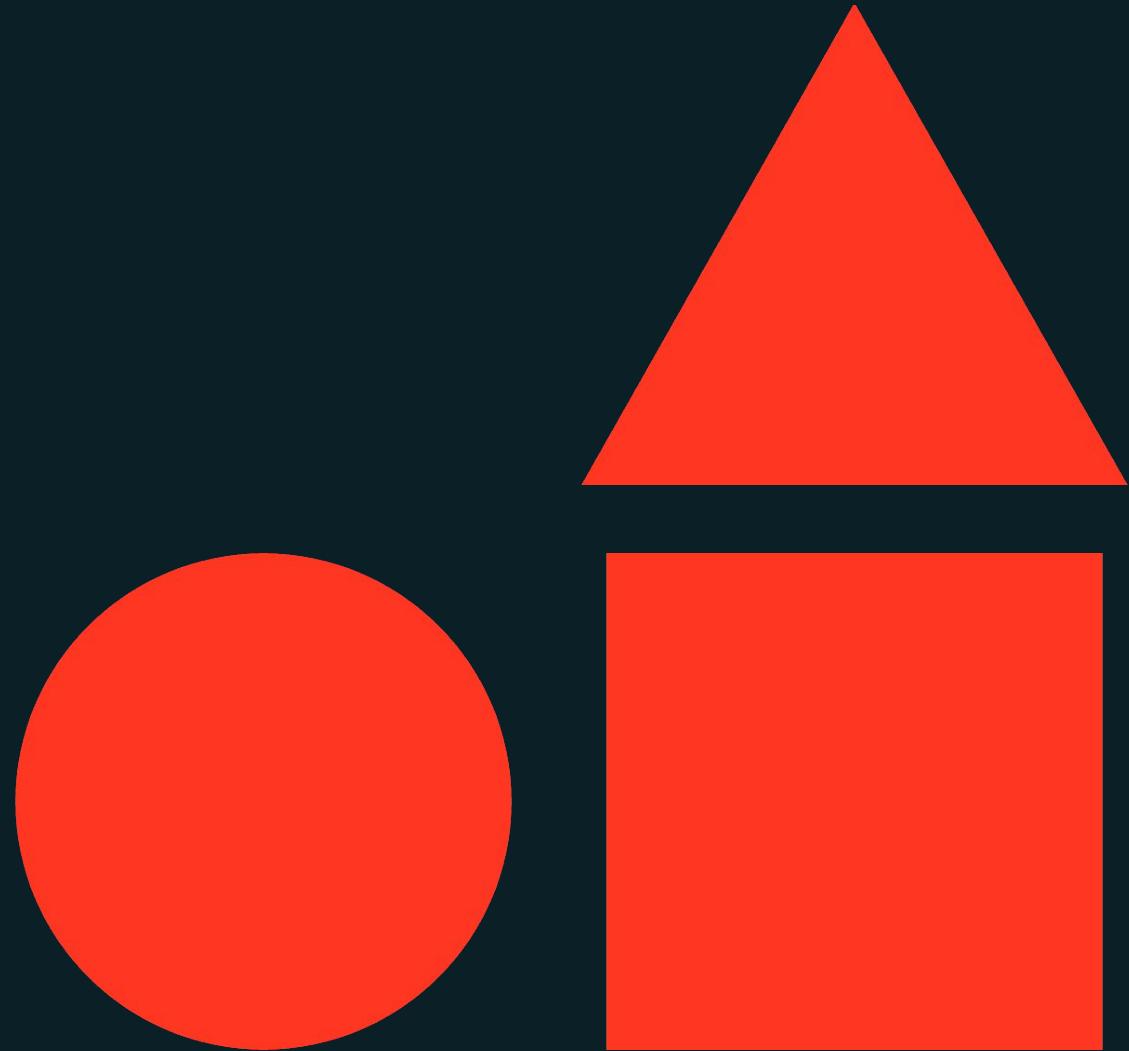
Metric	Description
Log loss (Cross-Entropy Loss)	Measures how many predicted positives are actually correct. Important when false positives are costly.
ROC/AUC	

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





Hyperparameter Tuning



Machine Learning Model Development



© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

Learning objectives

Things you'll be able to do after completing this module

- Define a hyperparameters
- Describe, compare, and contrast grid, random, and Bayesian search methods for hyperparameter tuning
- Describe cross-validation as a method for robust performance estimation
- Describe Optuna as an open-source hyperparameter optimization (HPO) framework





Hyperparameter Tuning

LECTURE

Hyperparameter Tuning Fundamentals



What is a Hyperparameter

Definitions

- **Hyperparameter** is a parameter whose value is used to **control** the training process.
- **Hyperparameter Tuning** (model tuning) is the process of adjusting the hyperparameters of a model **to find the best configuration** that **maximizes its performance** on a given dataset.

Example Hyperparameters:

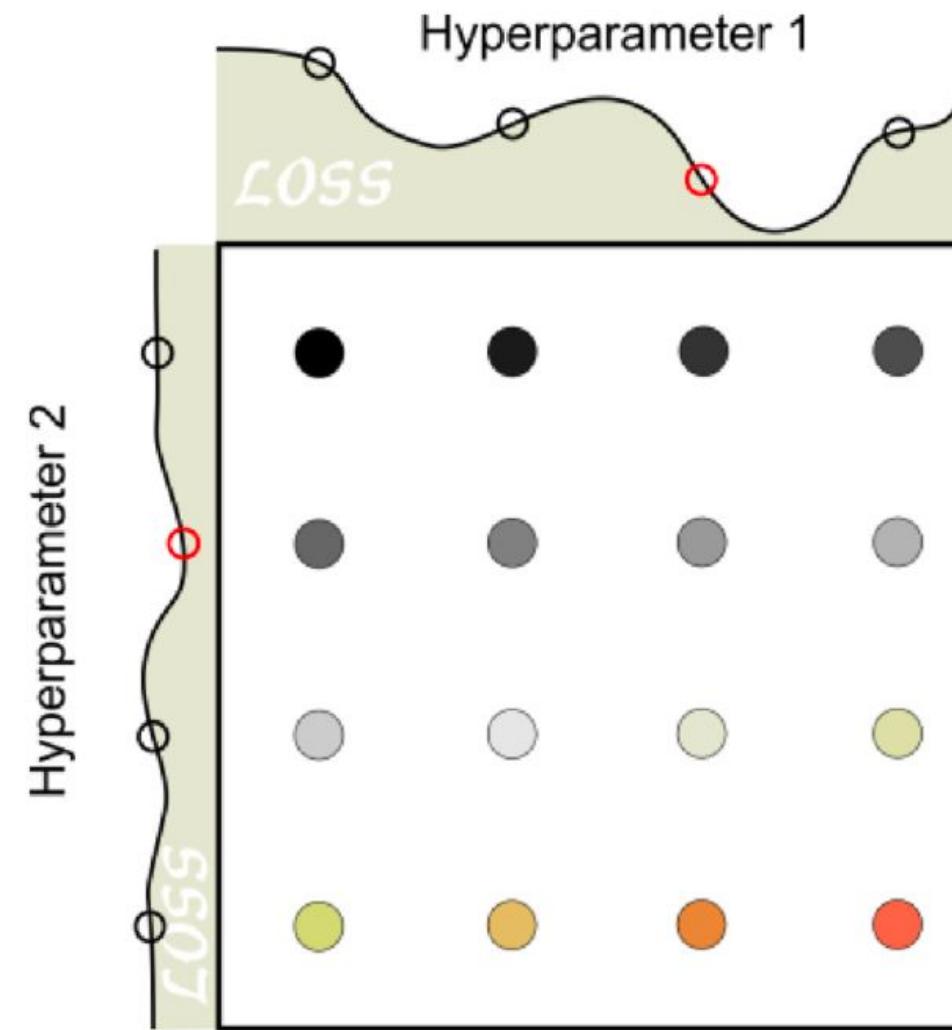
- Random Forest:
 - Maximum depth of trees
 - Number of trees
 - Number of features
- Linear regression
 - Normalize features or not
- Neural Networks
 - Activation function
 - Number of neurons in each hidden layer



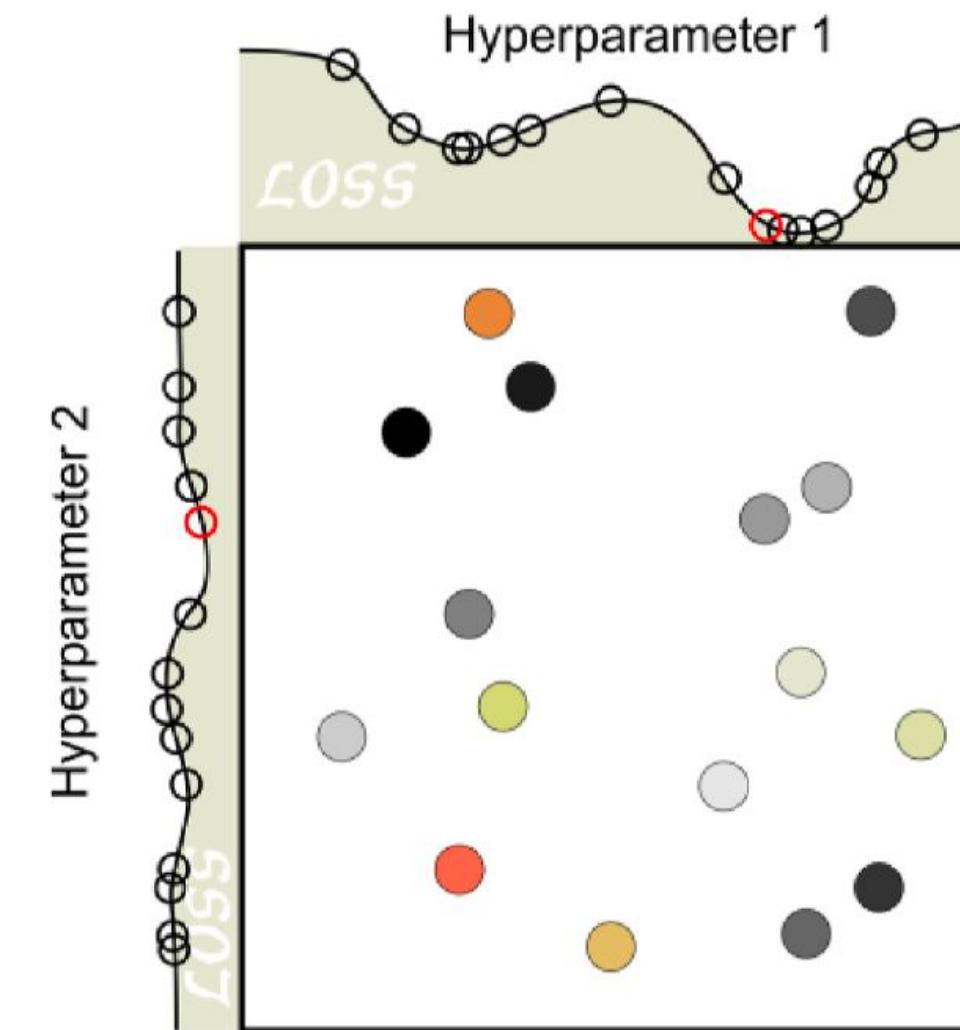
Hyperparameter Search Methods

How to pick a hyperparameter?

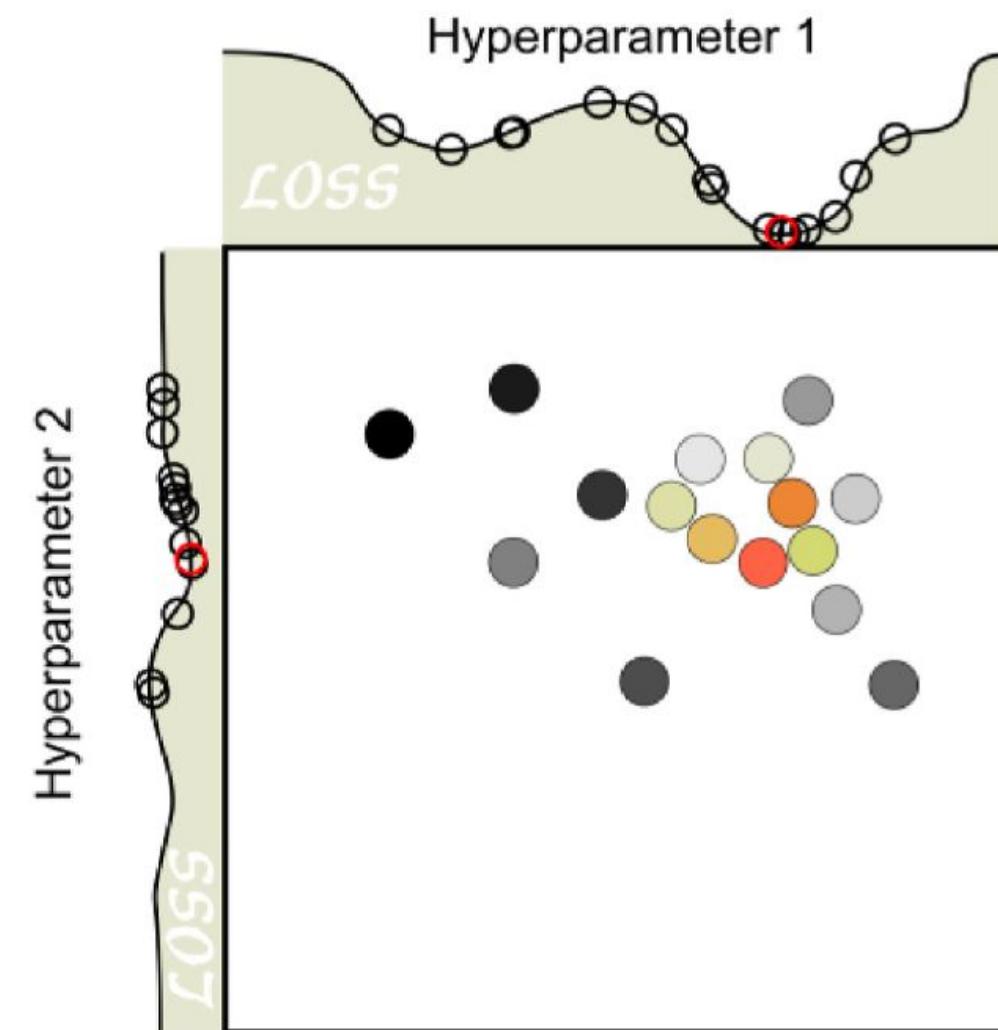
Grid Search



Random Search



Bayesian Optimization



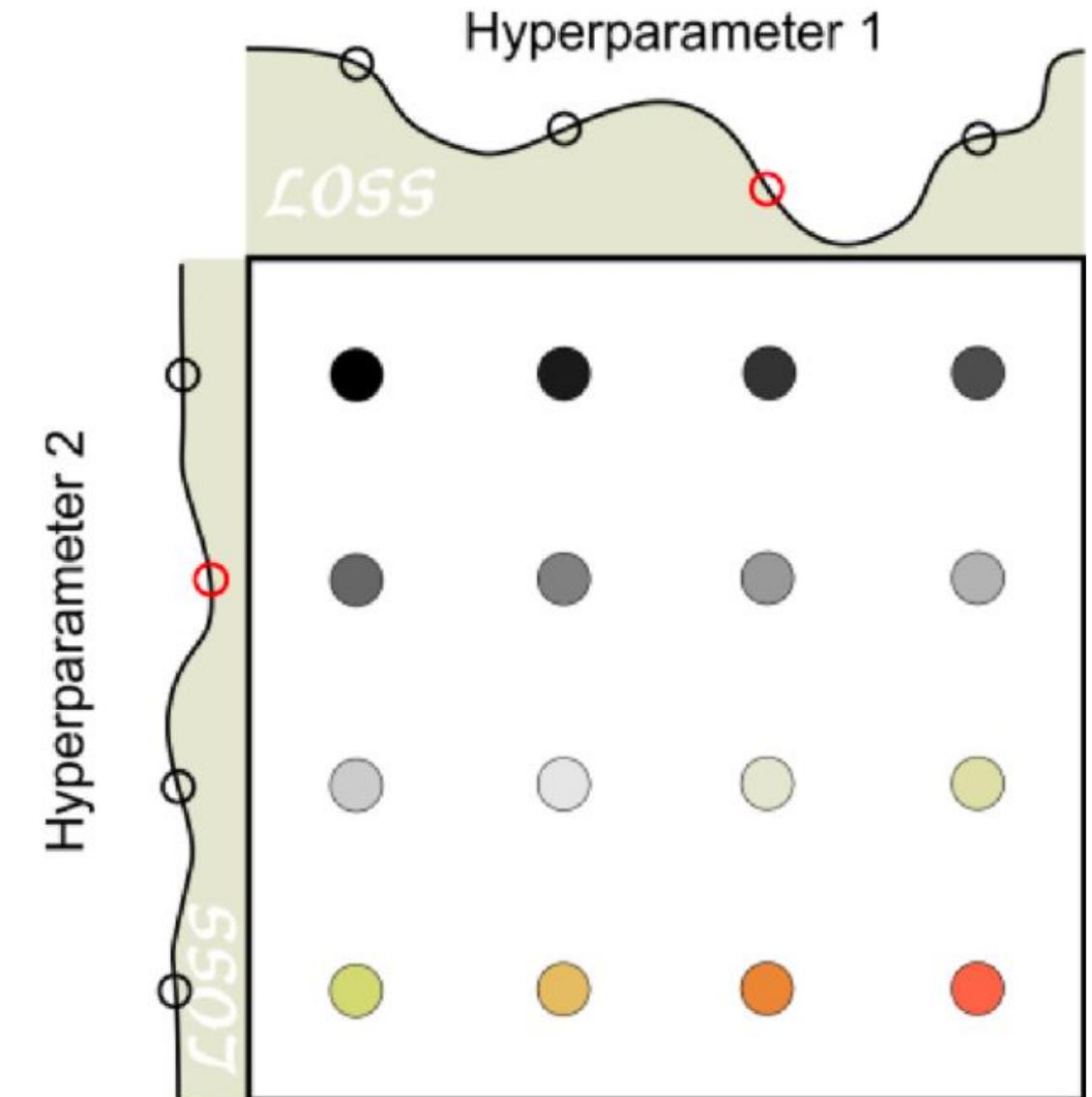
[Image Source](#)



Grid Search

Systematic parameter selection

- Define a grid of hyperparameter values and exhaustively searching through all possible combinations.
- Evaluate each combination using cross-validation and selects the best set of hyperparameters.
- Can be computationally expensive, especially when the number of hyperparameters and their possible values is large



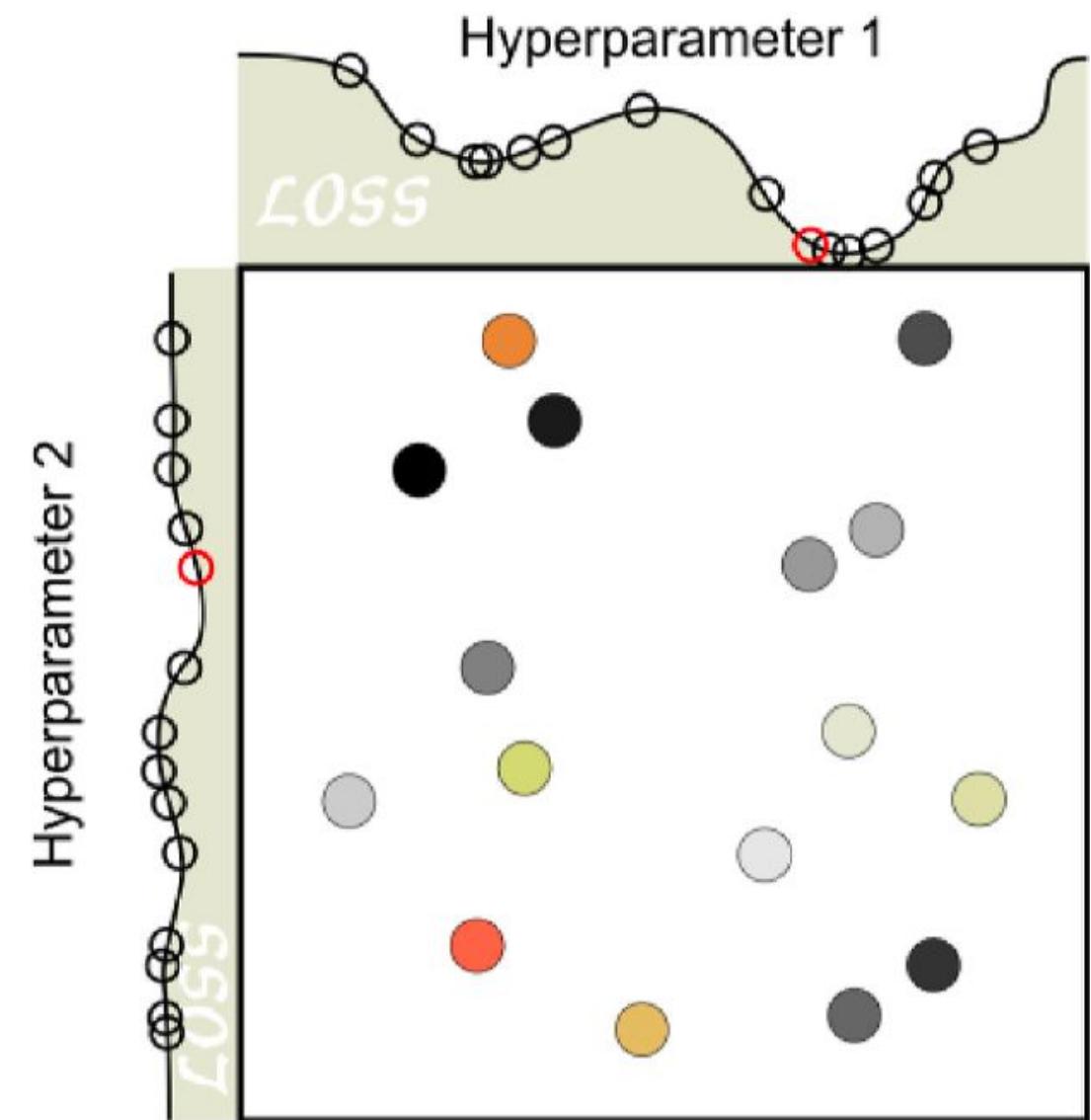
Grid Search will always find the best parameter, but is computational expensive.



Random Search

Stochastic parameter selection

- Randomly samples hyperparameter values from specified distributions.
- Random search is more efficient than grid search, especially when dealing with large datasets or hyperparameters.
- It doesn't guarantee the best combination.



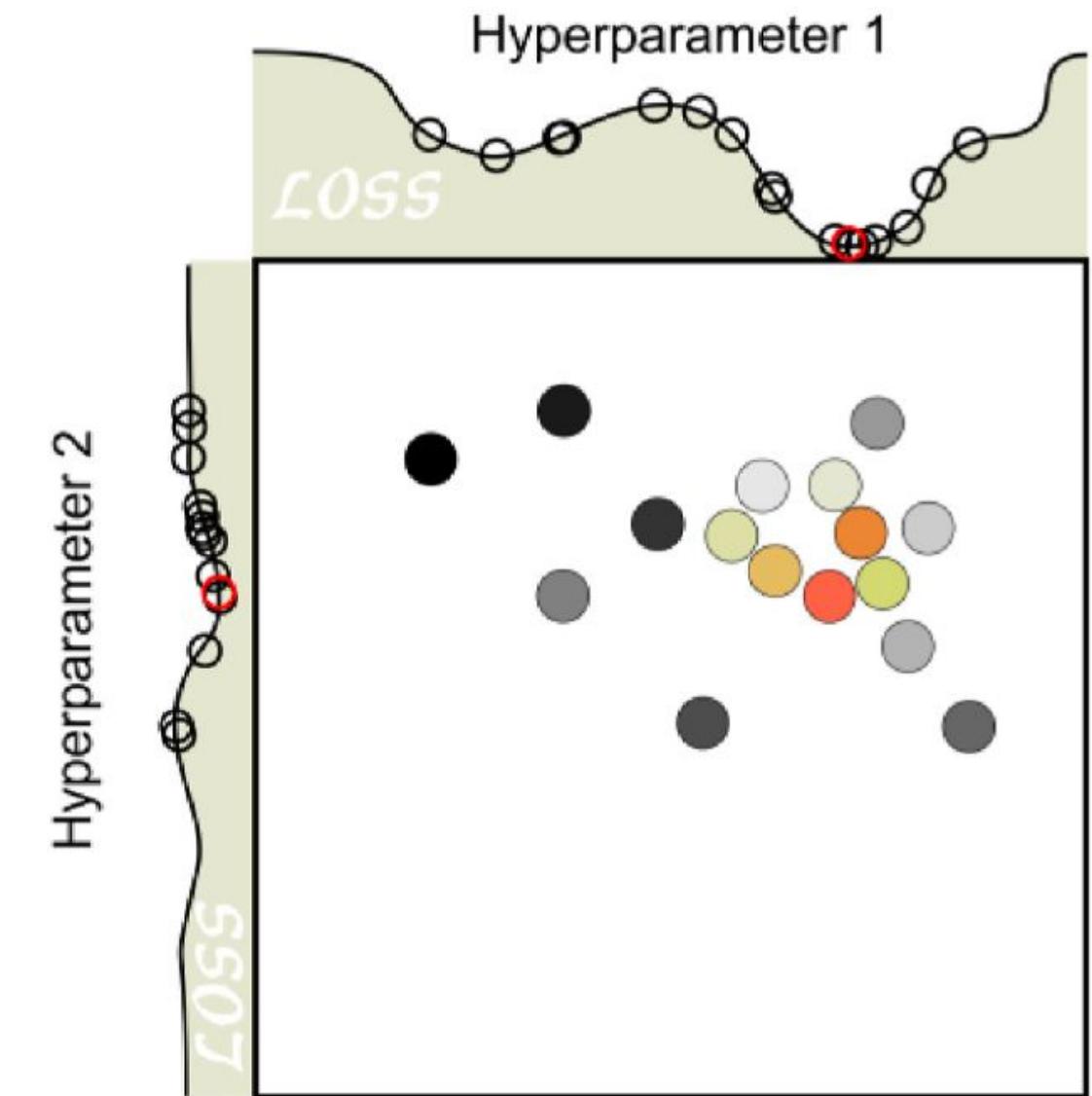
Random Search generally outperforms grid search, but does not systematically cover all possible values.



Bayesian Optimization

Probabilistic parameter selection

- Aiming to **minimize validation error** by constructing a **probability model** of the objective function.
- Guides the search based on **previous evaluations** to focus on promising regions of the hyperparameter space.
- **The Tree of Parzen Estimators (TPE)** algorithm is a popular Bayesian method.

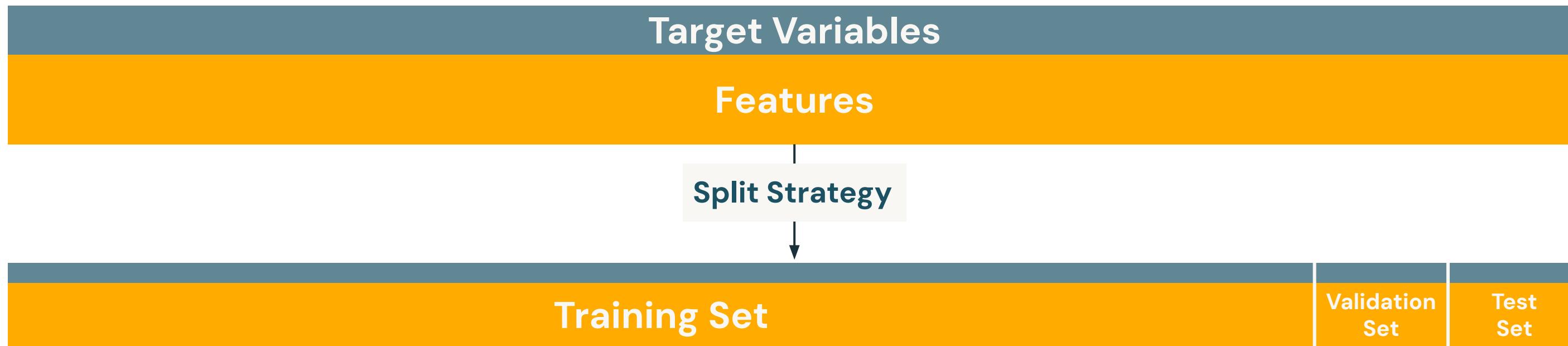


Bayesian approaches generally outperform grid search and random search, but more complex to implement.



Validation

- The validation set is one of our holdout datasets
- Build a model for each set of hyperparameter values
- Evaluate each model to identify the optimal hyperparameter values
- What dataset should we use to train and evaluate?

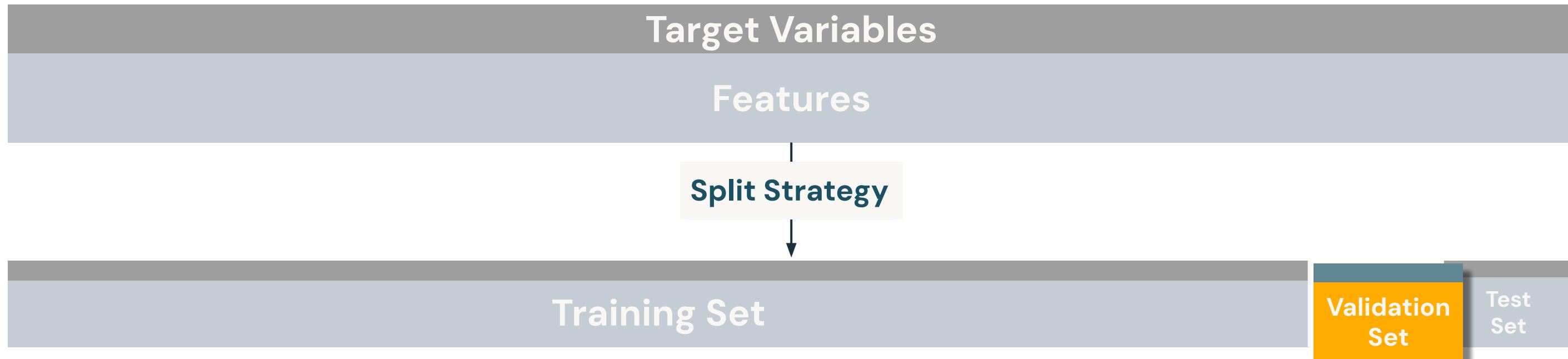


**What if there isn't enough data to split into three separate sets?
Risk of overfitting due to non-random initial split?**



Validation

- The validation set is one of our holdout datasets
- Build a model for each set of hyperparameter values
- Evaluate each model to identify the optimal hyperparameter values
- What dataset should we use to train and evaluate?

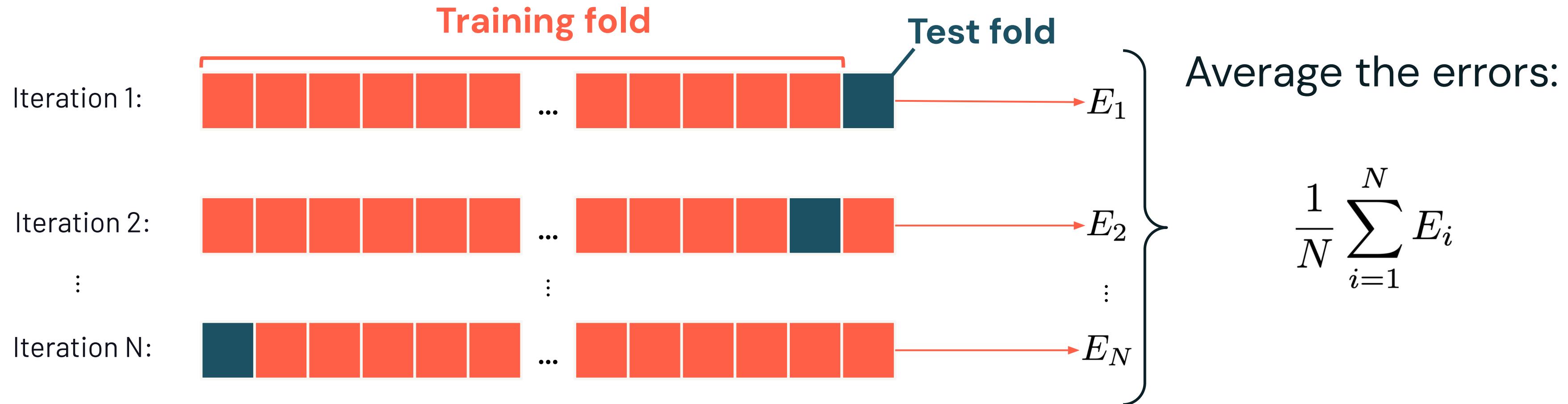


**What if there isn't enough data to split into three separate sets?
Risk of overfitting due to non-random initial split?**



K-Fold Cross Validation

A method for robust performance estimation



Final Pass:

Training with Optimal Hyperparameters

Test



Tuning Can be Expensive!

A Grid Search example

Train and validate **every unique combination** of hyperparameters

Tree Depth	Number of Trees
5	2
8	4



Tree Depth	Number of Trees
5	2
5	4
8	2
8	4

✗ K (folds)

✚ Best model

Question: With 3-fold cross validation, how many models will this build?



Optuna



© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

Popular Tools for Hyperparameter Opt.



- **Scikit-learn's GridSearchCV and RandomizedSearchCV:** Offer grid search and random search for hyperparameter tuning.



- **Hyperopt:** A popular library for performing hyperparameter optimization using Bayesian methods. **Hyperopt will be removed with DBR ML version 17.0+.**



- **Ray:** Ray Tune provides a scalable library for hyperparameter tuning, offering both grid and advanced search algorithms.



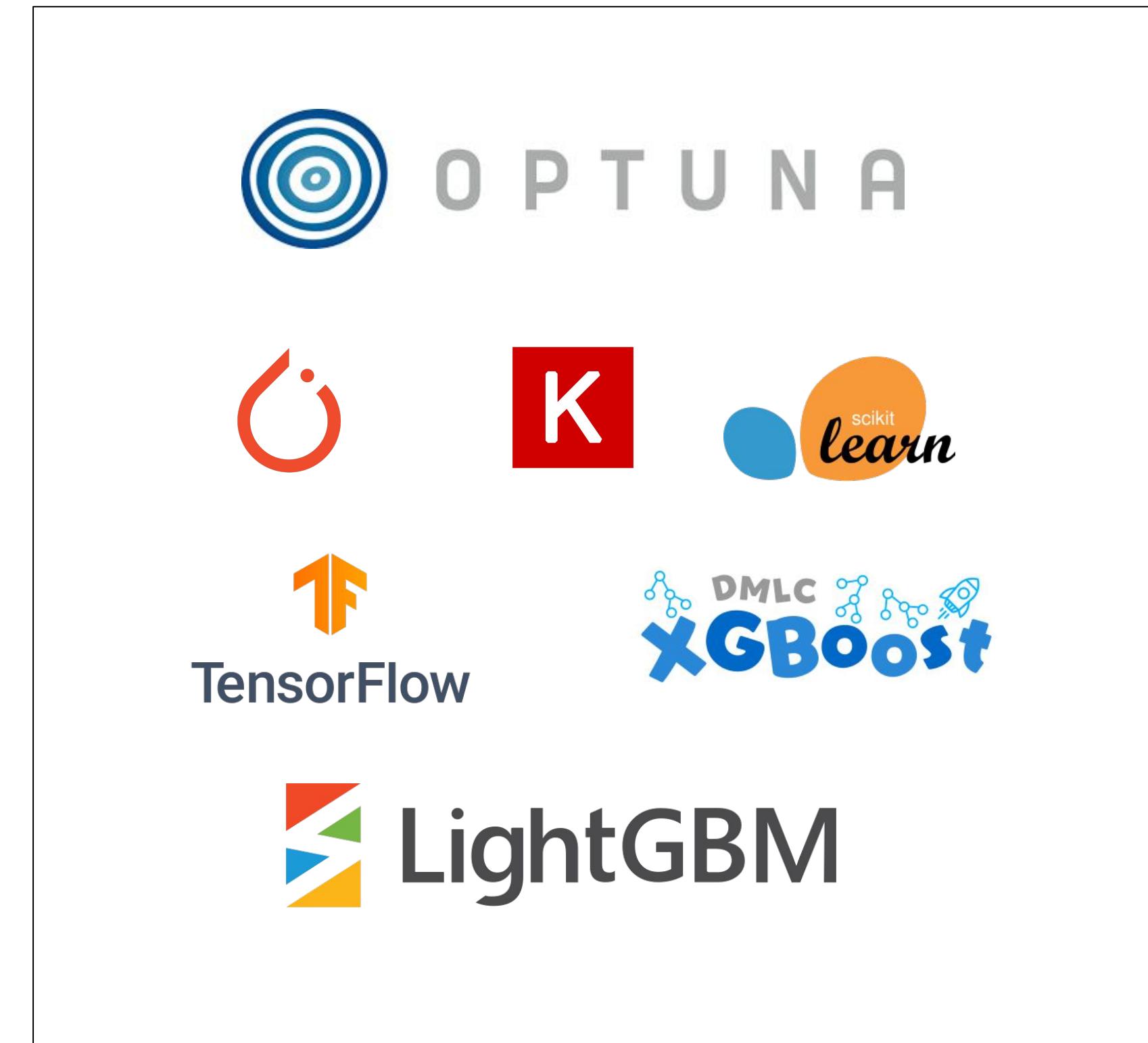
- **Optuna:** An open source hyperparameter optimization framework to automate hyperparameter search.



Optuna: Lightweight, Versatile, and Platform Agnostic

Simple installation and minimal requirements

- Optuna is written entirely in Python with few dependencies.
- Intuitive API:
 - Define parameters within your code.
 - No need for framework-specific syntax.
- Platform and Framework Agnostic
 - e.g. PyTorch, TensorFlow, Keras, Scikit-Learn, XGBoost, LightGBM, etc.



Optuna Terminology

Definitions

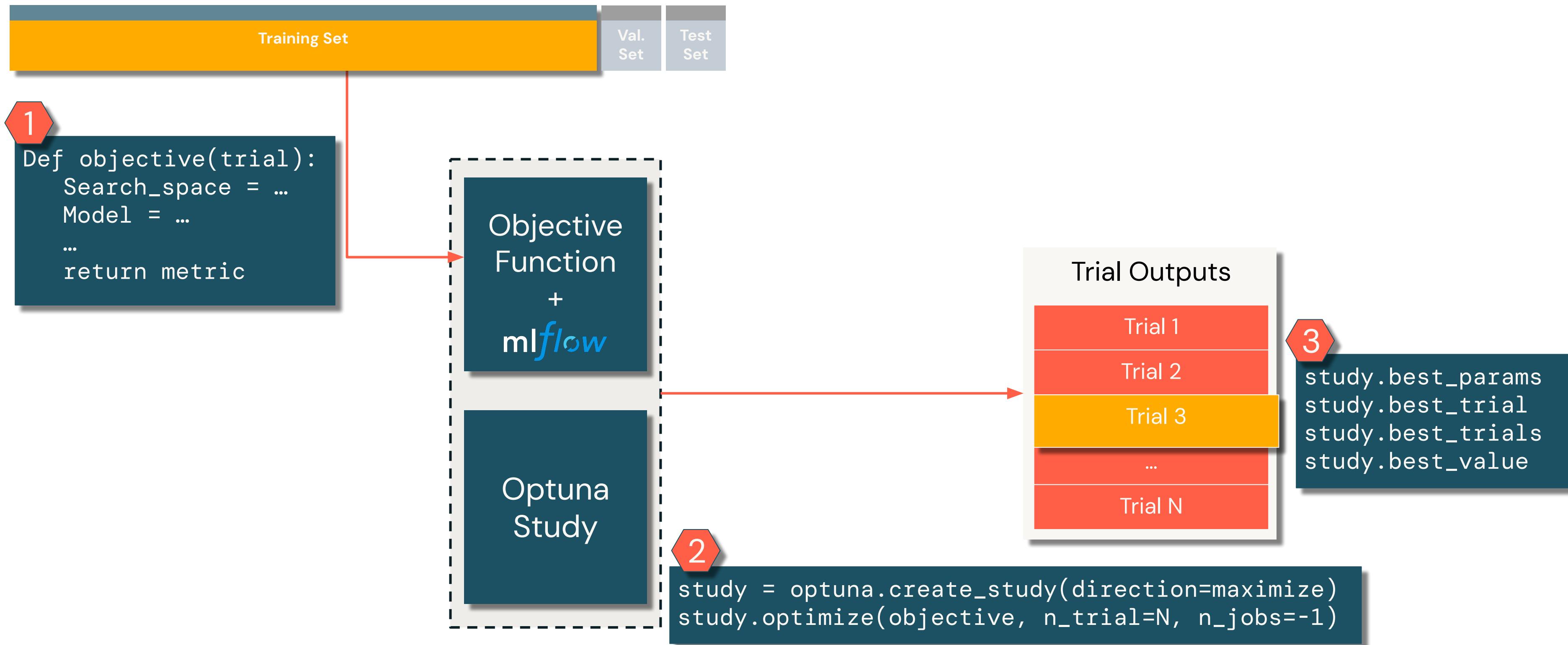
- A **trial** is a single call of the objective function.
- A **study** is an optimization session, which is a collection of trials.
- A **parameter** is a variable whose value is to be optimized, which is typically the loss or accuracy of the model.

Optuna is a framework designed for the *automation* and the *acceleration* of the *optimization* studies.



Hyperparameter Optimization Framework

Automating Hyperparameter Search



Key Concepts

Search space and algorithms

Search Space Definition with Python Syntax

- Hyperparameter search space can be defined using functions such as;
 - `suggest_discrete_uniform`
 - `suggest_categorical`
 - `suggest_float`
 - `suggest_int`
 - `suggest_uniform`

Sampling Strategy

- Sampling strategy determines how to decide the next hyperparameter.
- Continuously refine the search space using past evaluation results to focus on promising areas.
- Some of the supported strategies; Grid Search, Random Search, **Tree-structured Parzen Estimator algorithm (default)**

Pruning Strategy*

- Early stopping halts unpromising trials early based on intermediate results.
- Call `report()` and `should_prune()` after each step to activate pruning.

*Not covered in this course.



Optuna - MLflow Integration

Track hyperparameters and metrics of all the Optuna trials

MLflowCallback: This callback enables automatic logging of hyperparameters and metrics for each trial into MLflow, streamlining experiment tracking.

Parent-Child Run Organization: MLflow's support for hierarchical run structures allows grouping of all Optuna trials under a single parent run.

```
from optuna.integration.mlflow import MLflowCallback
mlflow_callback = MLflowCallback(
    tracking_uri = "databricks",
    create_experiment= False,
    mlflow_kwargs = {
        "experiment_id": <experiment_id>,
        "Nested": True
    }
)
study.optimize(
    objective,
    n_trials = N,
    callbacks = [mlflow_callback])
```

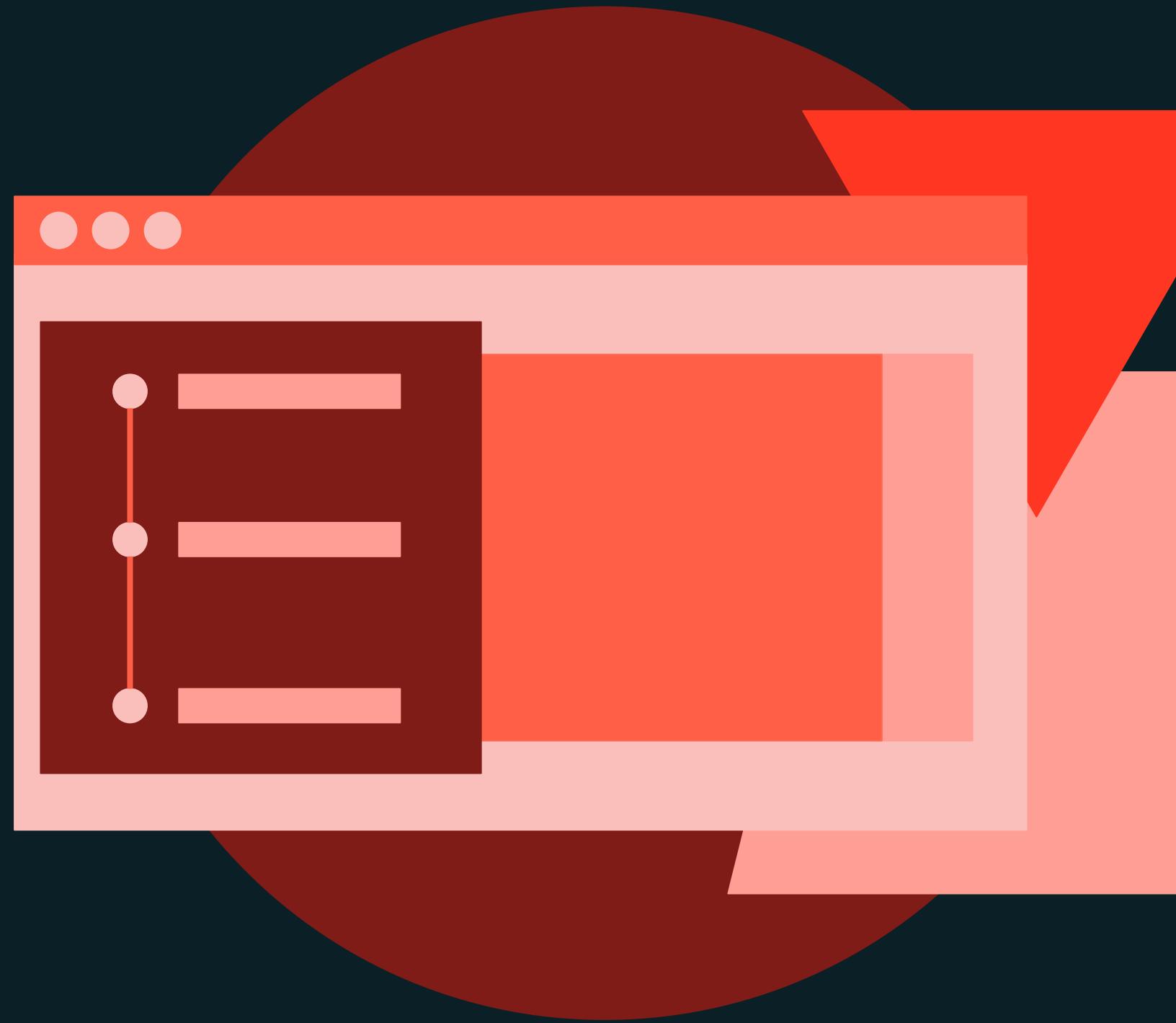




Hyperparameter Tuning

DEMONSTRATION

Hyperparameter Tuning with Optuna





Hyperparameter Tuning

LAB EXERCISE

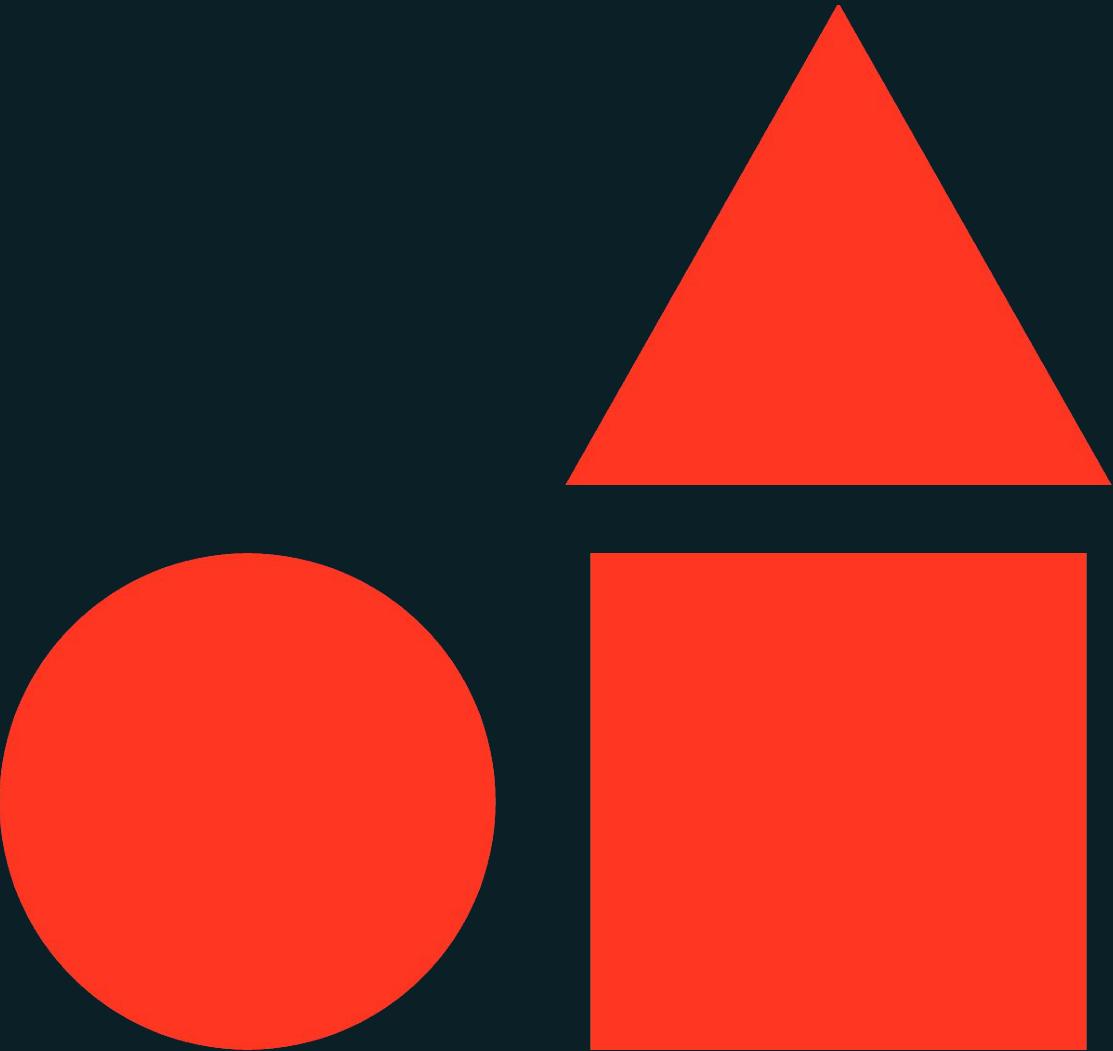
Hyperparameter Tuning with Optuna





AutoML

Machine Learning Model Development



Learning objectives

Things you'll be able to do after completing this module

- Describe the benefits of using AutoML to generate machine learning models.
- Describe the glass box nature of AutoML.
- Start an AutoML experiment via the UI and the API.
- Open and edit a notebook generated by AutoML.
- Identify the best model generated by AutoML based on a given metric.
- Modify the best model generated by AutoML.





AutoML

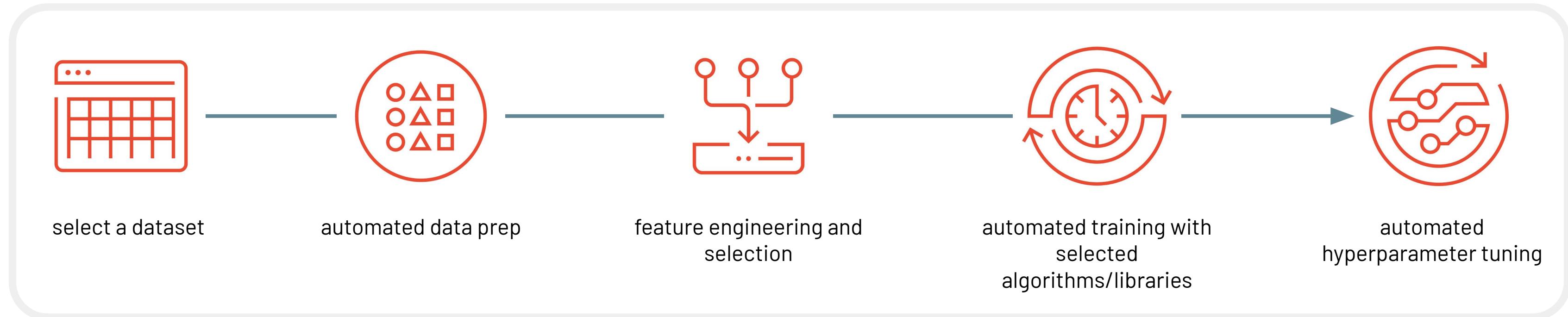
LECTURE

Introduction to AutoML



What is AutoML?

Automated machine learning (AutoML) is a fully-automated model development solution seeking to “democratize” machine learning. While the scope of the automation varies, AutoML technologies usually **automate the ML process from data to model selection**.



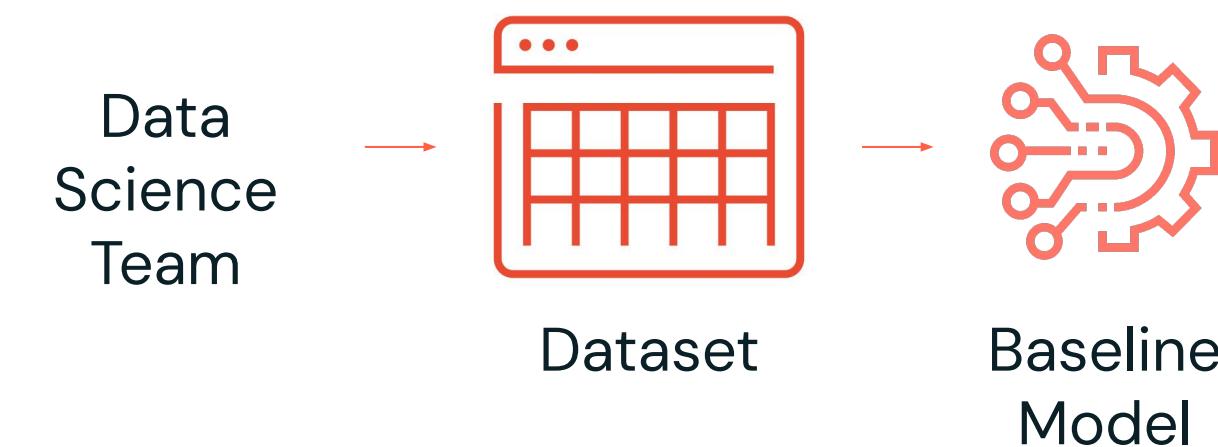
Two Key Pain Points Solved with AutoML

Quickly Verify the Predictive Power of a Dataset



"Can this dataset be used to predict customer churn?"

Get a Baseline Model to Guide Project Direction

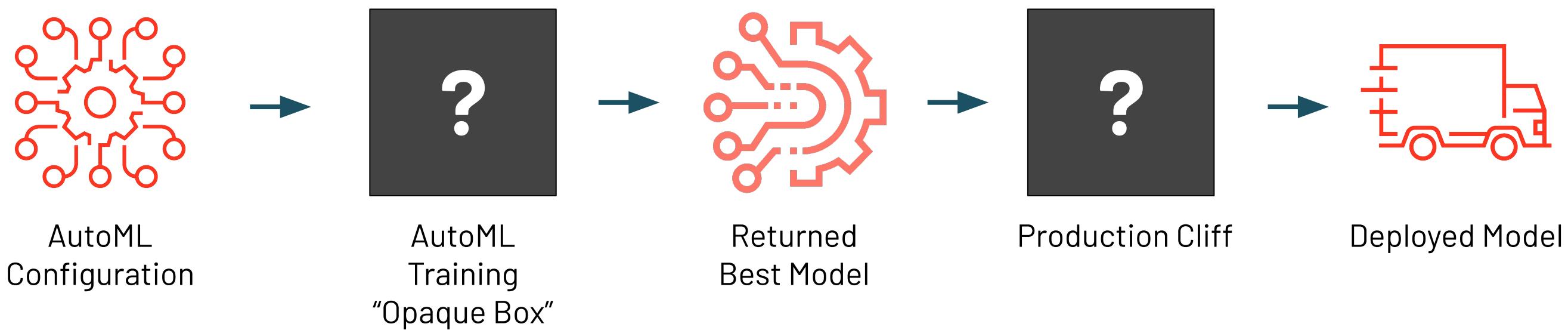


"What direction should I go in for this ML project and what benchmark should I aim to beat?"



Problems with Existing AutoML Solutions

Opaque-Box and Production Cliff Problems in AutoML

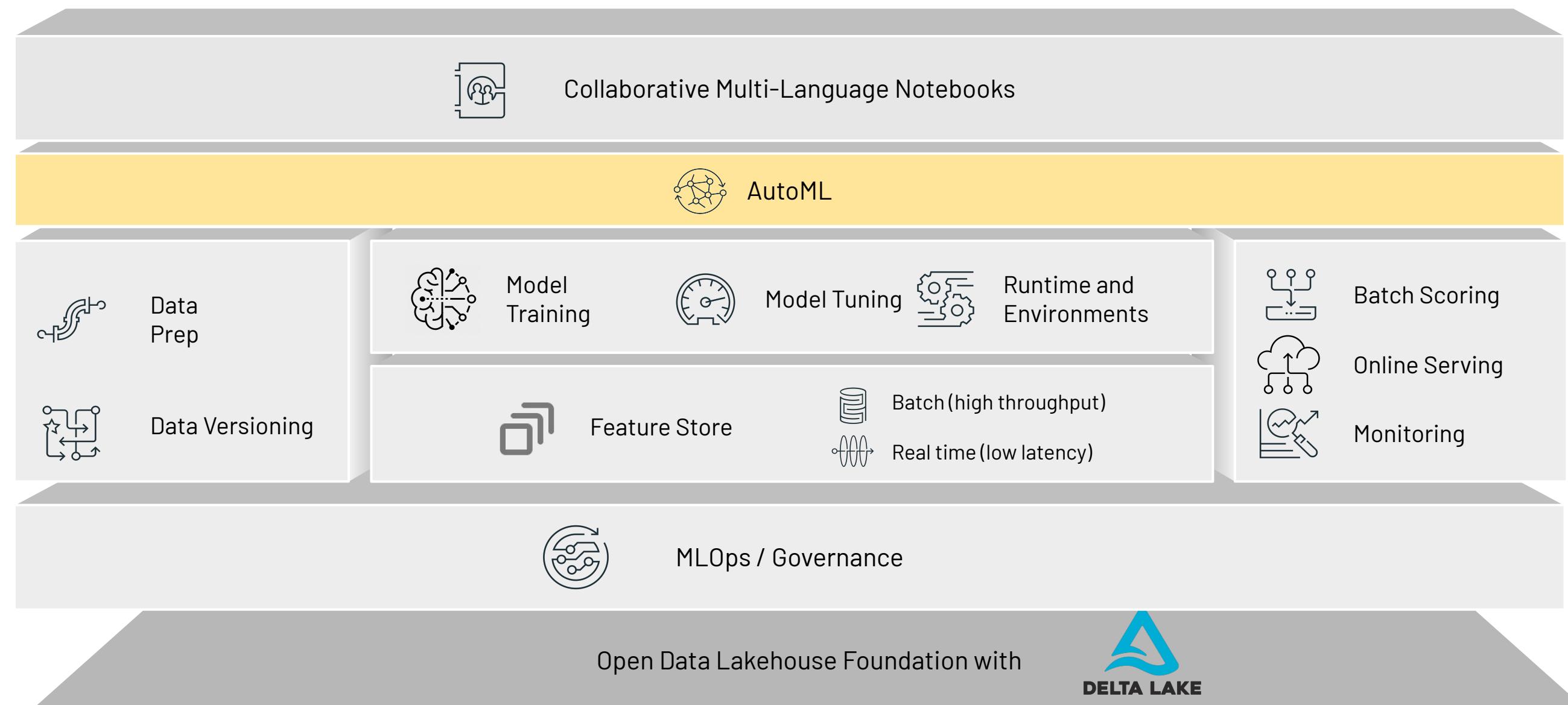


Problem	Result / Pain Points
<ul style="list-style-type: none">⚠️ Data scientists need to be able to explain how they trained a model for regulatory purposes (e.g., FDA, GDPR, etc.) - "opaque box" models do not help.⚠️ A "production cliff" exists where data scientists need to modify the returned "best" model using their domain expertise.	<ul style="list-style-type: none">⚠️ Time and energy required to reverse engineering these "opaque-box" returned models to modify and/or explain them.⚠️ The "best" model returned is often not good enough to deploy



Solution: Databricks AutoML

Rapid, simplified machine learning built on the Lakehouse Architecture



Solution: Databricks AutoML

Rapid, simplified machine learning **for everyone**

Quick-start ML initiatives

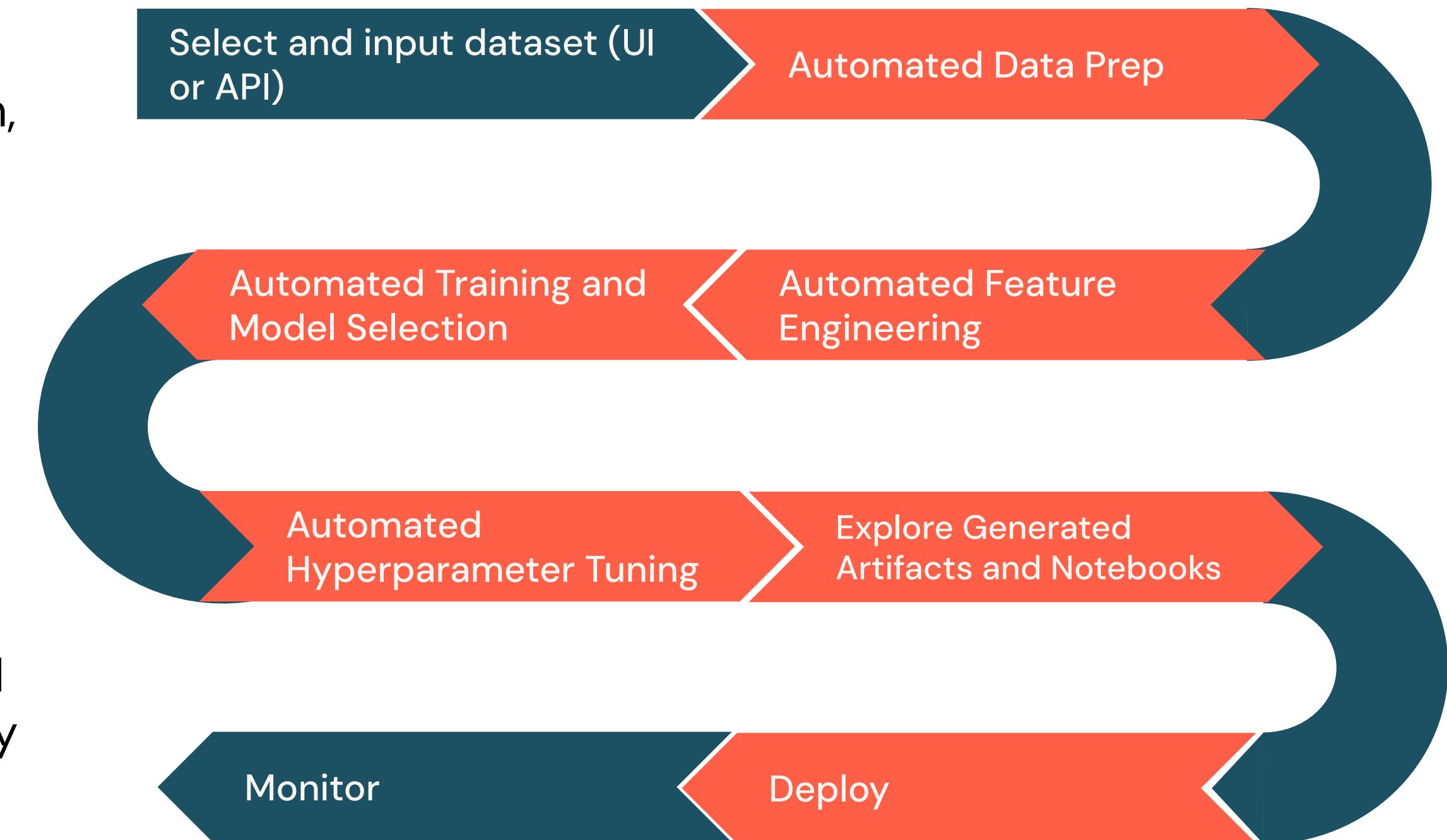
Accelerate your time to production,
Save weeks on ML projects

Auto-generated notebooks

Ensure best practices.
Customize baseline models with
your domain expertise

Wide range of problems

Solve classification, regression, and
forecasting problems from a variety
of ML libraries



Databricks AutoML

A glass-box solution empowering data teams without taking away control

UI and API to start AutoML training

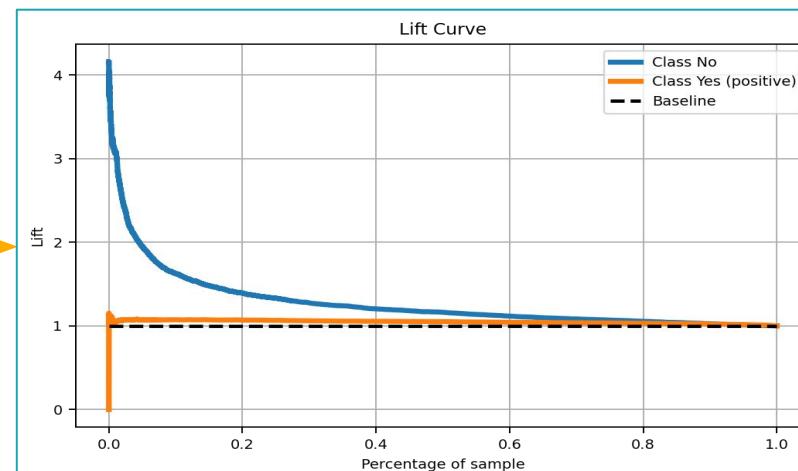
Configure AutoML experiment

Compute Configuration

Experiment Configuration

Advanced Configuration (optional)

Run Name	Created	Duration	Source	Models	val_f1_score
rebellious-finch-526	11 days ago	1.9min	Notebook	sklearn	0.865
vaulted-calf-990	11 days ago	1.9min	Notebook	sklearn	0.864
smiling-lark-727	11 days ago	2.2min	Notebook	sklearn	0.864
Experiment 2	11 days ago	1.8min	Notebook	sklearn	0.862
languid-vole-290	11 days ago	1.9min	Notebook	sklearn	0.862
industrious-colt-828	11 days ago	1.9min	Notebook	sklearn	0.862
debonair-gnat-986	11 days ago	1.8min	Notebook	sklearn	0.862
capable-skink-256	11 days ago	1.7min	Notebook	sklearn	0.862
painted-bear-922	11 days ago	2.0min	Notebook	sklearn	0.862
fun-steed-935	11 days ago	1.7min	Notebook	sklearn	0.862



XGBoost Classifier training

Table of contents

Code 1

Code 2

Code 3

Load Data

Auto-created
MLflow Experiment
to track models and
metrics

Easily deploy to
Model Registry

Auto-generated
Data Exploration
notebook

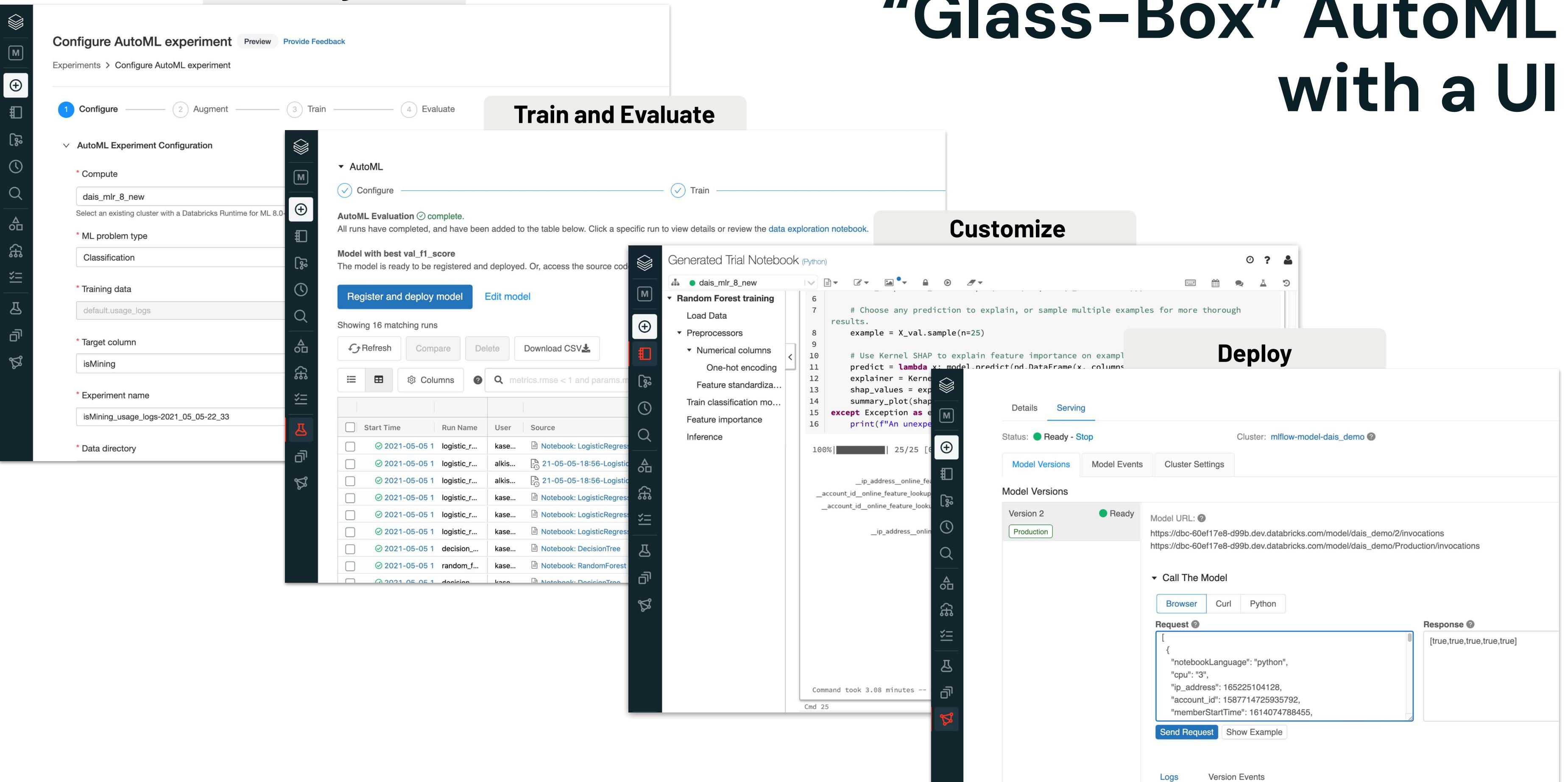
Understand and
debug data
quality and
preprocessing

Auto-generated
notebooks with
source code

Iterate further on
models from
AutoML, adding
your expertise



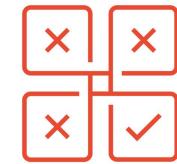
“Glass-Box” AutoML with a UI



AutoML Supports Common Problem Types

Support for problem, model, feature types

Problem Types



Classification



Regression



Time Series
Forecasting
(Serverless)

Models / Tuning



XGBoost



HYPEROPT



PROPHET

Features



Numerical



Categorical



Timestamp



Text



Complete Integration

AutoML integrated with other products

AutoML integrates with;

- **Feature Store:** You can expand the input training dataset using existing feature tables.
- **Model Tracking:** All trial run metrics and parameters are tracked.
- **Model Registry:** You can register generated models to Unity Catalog.
- **Model Serving:** You can deploy generated models with Model Serving for real-time inference.

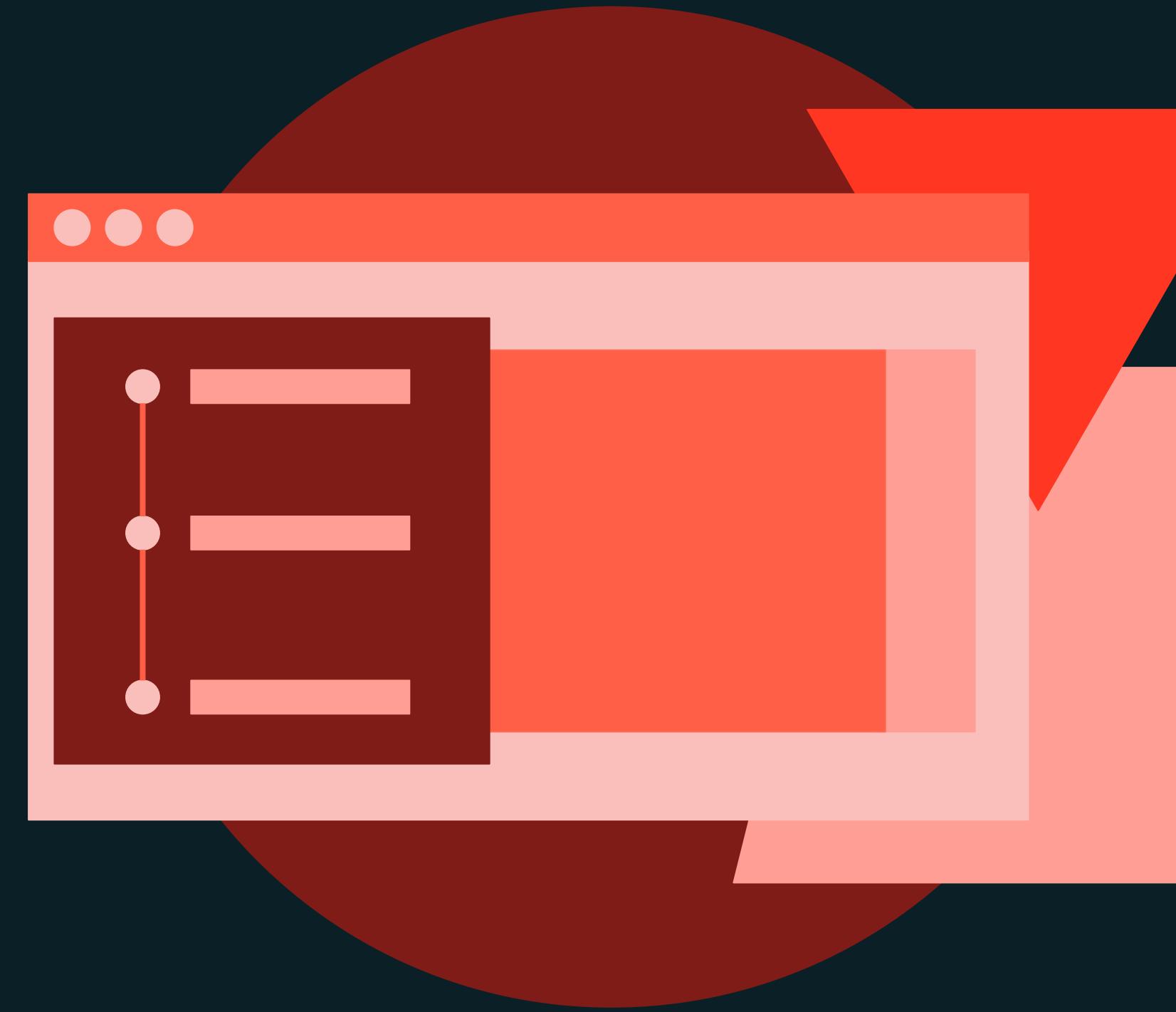




AutoML

DEMONSTRATION

Automated Model Development with AutoML



Demo

Outline

What we'll cover:

- Prepare data
- AutoML Experiment with the UI
 - Create an experiment
 - View the best run
- AutoML Experiment with the API
 - Start an experiment
 - Search for best run
 - Import notebooks for other experiments





AutoML

LAB EXERCISE

AutoML



© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).

Lab

Outline

What you'll do:

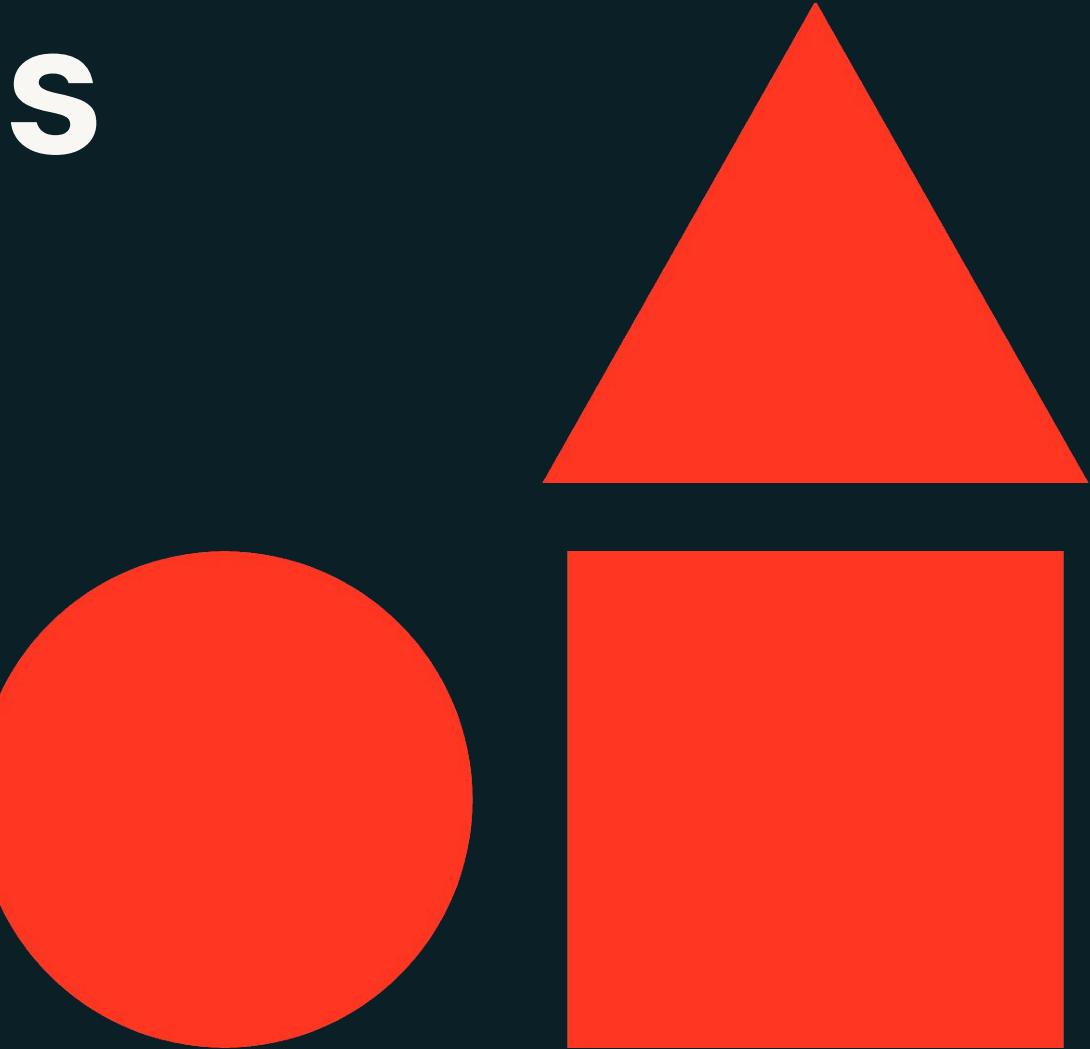
- Load dataset
- Create a classification experiment using the AutoML UI
- Create a classification experiment with the AutoML API
- Retrieve the best run and show the model URI
- Import the notebook for a run





Summary and Next Steps

Machine Learning Model Development





databricks



© Databricks 2025. All rights reserved. Apache, Apache Spark, Spark, the Spark Logo, Apache Iceberg, Iceberg, and the Apache Iceberg logo are trademarks of the [Apache Software Foundation](#).