Status: In Progress

14 DAYS AI CHALLENGE

Day 13

Model Comparison & Feature Engineering

#Databricks #MachineLearning #Spark

INDIAN DATA CLUB

The Mission Parameters

1. Train

Initialize and fit 3 distinct regression models.

2. Compare

Utilize MLflow to log and analyze performance metrics.

3. Build

Construct a scalable Spark ML pipeline.

4. Select

Identify the optimal model for production.

#DatabricksWithIDC

Phase 1: Establishing the Control Group

```
Let's start by generating some synthetic regression data
and splitting it into training and testing sets.

from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split

X, y = make_regression(n_samples=200, n_features=5,
    noise=0.1, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
```

Lab Note

Data Source: Synthetic regression data (200 samples) ensures a clean testing ground.

Validation: Standard 80/20 split.

Defining the Contenders

```python
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

models = {
    'linear': LinearRegression(),
    'decision_tree': DecisionTreeRegressor(max_depth=5),
    'random_forest': RandomForestRegressor(n_estimators=100)
}
```

Modern Technical Documentation

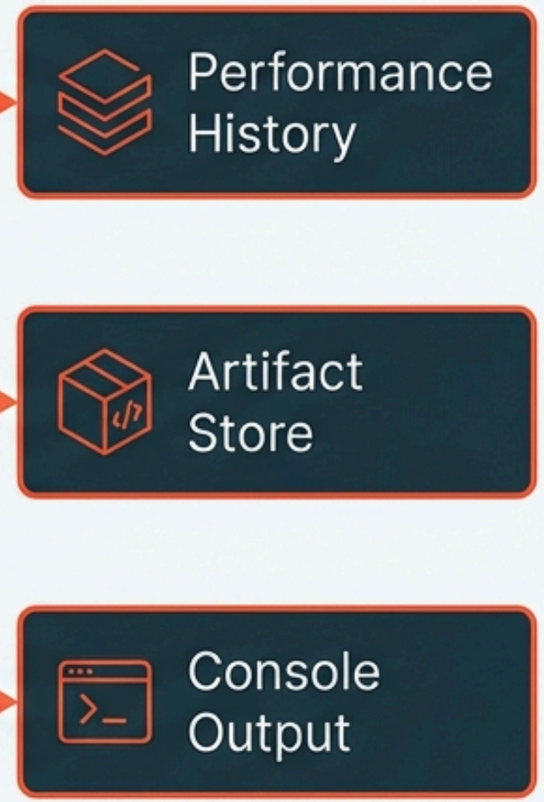
The Iteration Engine

```
import mlflow

for name, model in models.items():
    with mlflow.start_run(run_name=f'{name}_model'):
        mlflow.log_param('model_type', name)

        model.fit(X_train, y_train)
        score = model.score(X_test, y_test)
```

Automation: Iterating through the dictionary candidates.

Observability with MLflow

```
mlflow.log_metric('r2_score', score)
mlflow.sklearn.log_model(model, 'model')

print(f'{name}: R2 = {score:.4f}')
```



Scaling Up: From Experiment to Pipeline

Phase 1: Experimentation

Scikit-Learn
(In-Memory)

Productionizing

Phase 2: Engineering

Spark ML
(Distributed)

The Spark Toolkit

```
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression as SparkLR
```

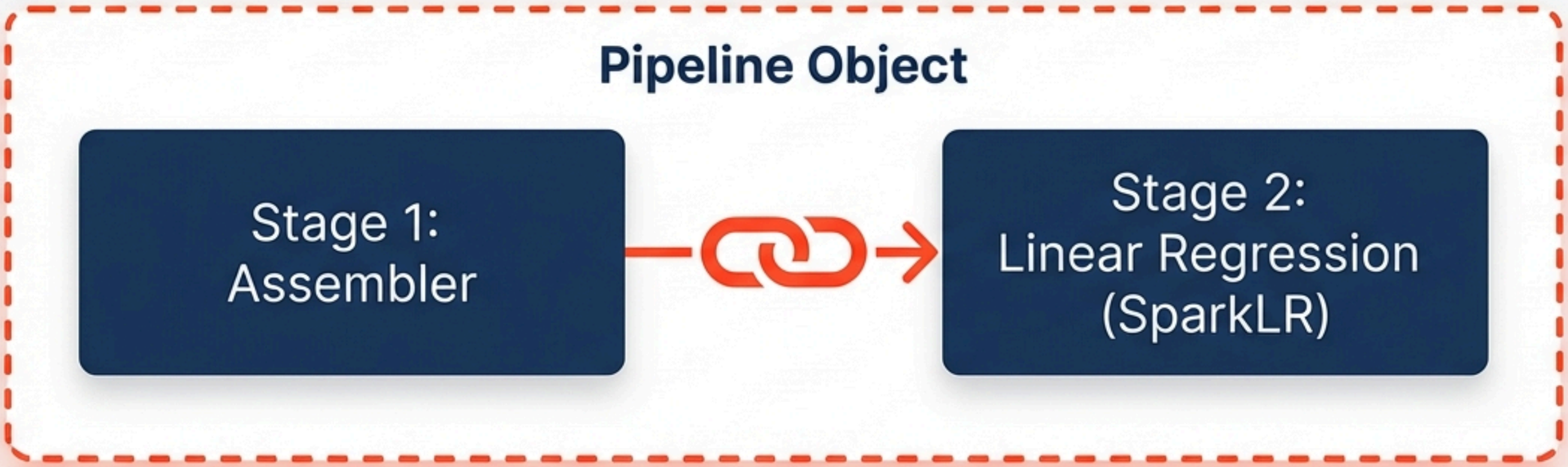
Critical Alias: We rename the Spark LinearRegression to 'SparkLR' to avoid conflicts with the Scikit-Learn library used in Phase 1.

Feature Engineering: The Vector Assembler

```
assembler = VectorAssembler(
    inputCols=['views', 'purchases'],
    outputCol='features'
)
```

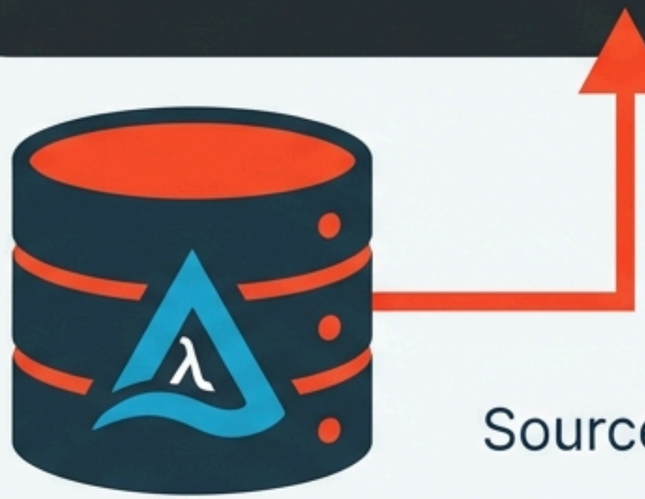
Architecture of the Pipeline

```
lr = SparkLR(featuresCol='features', labelCol='purchases')
pipeline = Pipeline(stages=[assembler, lr])
```



Ingesting Production Data

```
spark_df = spark.read.format('delta').load(
      '/Volumes/workspace/ecommerce/ecommerce_data/gold_products'
)
```



Source: ‘gold_products’ Delta Table.

Execution: Train and Split

```
train, test = spark_df.randomSplit([0.8, 0.2])
model = pipeline.fit(train)
```

Single Action: Fitting the pipeline triggers the data ingestion, vector assembly, and model training in one sequence.

The Data Profile

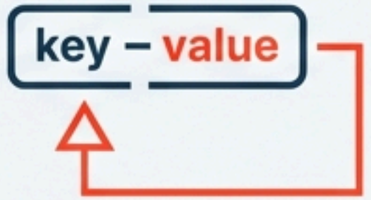
product_id	views	purchases	revenue	conversion_rate
8500290	1	1	10002.30	100
1005159	1	1	524205.44	100
17300014	1	1	968.8	100
12400055	1	1	116.09	100

Day 13: Technical Debrief

1

Iterative Comparison

Using Python dictionaries and loops beats manual redundancy.



2

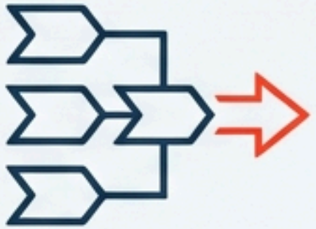
MLflow Observability

Logging metrics is essential for historical analysis.

3

Spark Pipelines

Encapsulating transformation and modeling **creates** portable production workflows.



Day 13 Status: Completed

Ready for Day 14

14 DAYS AI CHALLENGE

INDIAN DATA CLUB