# Objectives Part 2 – Code Walkthrough

## Step 1: Load Dataset from GitHub

%pyspark  
import pandas as pd  
  
url = "https://raw.githubusercontent.com/ryumatsu/soccer-betting-ml/refs/heads/main/E0\_cleaned.csv"  
pdf = pd.read\_csv(url)  
  
df = spark.createDataFrame(pdf).cache()  
  
df.printSchema()  
print("Total rows:", df.count())  
df.select(df.columns[:6]).limit(3).show(truncate=False)

## Step 2: Create Binary Label & Drop Nulls

%pyspark  
from pyspark.sql.functions import when, col  
  
features = ["B365H","B365D","B365A"]  
  
df\_bin = (df  
 .withColumn("label", when(col("Result") == "H", 1).otherwise(0))  
 .select(["label"] + features)  
 .dropna())  
  
print("Rows after cleaning:", df\_bin.count())  
df\_bin.limit(5).show()

## Step 3: Assemble Features & Train/Test Split

%pyspark  
from pyspark.ml.feature import VectorAssembler  
  
vec = VectorAssembler(inputCols=features, outputCol="features")  
data = vec.transform(df\_bin).select("label", "features")  
  
train, test = data.randomSplit([0.7, 0.3], seed=42)  
  
print(f"Train rows: {train.count()}, Test rows: {test.count()}")

## Step 4: Train Four Classifiers

%pyspark  
from pyspark.ml.classification import (  
 LogisticRegression, RandomForestClassifier,  
 DecisionTreeClassifier, GBTClassifier)  
  
lr = LogisticRegression(labelCol="label", featuresCol="features")  
rf = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=100)  
dt = DecisionTreeClassifier(labelCol="label", featuresCol="features")  
gbt = GBTClassifier(labelCol="label", featuresCol="features", maxIter=50)  
  
lr\_m = lr.fit(train)  
rf\_m = rf.fit(train)  
dt\_m = dt.fit(train)  
gbt\_m = gbt.fit(train)

## Step 5: Evaluate Models (AUC & Accuracy)

%pyspark  
from pyspark.ml.evaluation import BinaryClassificationEvaluator  
  
eval\_auc = BinaryClassificationEvaluator(labelCol="label", metricName="areaUnderROC")  
results = []  
  
for name, model in [("LogReg", lr\_m), ("RandForest", rf\_m), ("DecTree", dt\_m), ("GBT", gbt\_m)]:  
 pred = model.transform(test)  
 auc = eval\_auc.evaluate(pred)  
 acc = pred.filter(col("prediction") == col("label")).count() / test.count()  
 results.append((name, round(auc, 3), round(acc, 3)))  
  
spark.createDataFrame(results, ["Model", "AUC", "Accuracy"]).show()

## Step 6: Random Forest Feature Importance

%pyspark  
for f, imp in zip(features, rf\_m.featureImportances.toArray()):  
 print(f"{f}: {imp:.4f}")

## Step 7: Cross-Validate Random Forest

%pyspark  
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder  
  
grid = (ParamGridBuilder()  
 .addGrid(rf.numTrees, [50, 100])  
 .addGrid(rf.maxDepth, [3, 5])  
 .build())  
  
cv = CrossValidator(estimator=rf,  
 estimatorParamMaps=grid,  
 evaluator=eval\_auc,  
 numFolds=3)  
  
cv\_model = cv.fit(train)  
best\_auc = eval\_auc.evaluate(cv\_model.transform(test))  
  
print("Best RF AUC after CV:", round(best\_auc, 3))  
print("Best params – trees:", cv\_model.bestModel.getNumTrees,  
 ", depth:", cv\_model.bestModel.getOrDefault('maxDepth'))

## Step 8: Confusion Matrix (Best Model)

%pyspark  
from pyspark.mllib.evaluation import MulticlassMetrics  
  
best\_pred = lr\_m.transform(test)  
  
cm = MulticlassMetrics(best\_pred.select("prediction", "label")  
 .rdd.map(lambda r: (float(r[0]), float(r[1])))).confusionMatrix().toArray()  
  
print("Confusion Matrix:")  
for row in cm:  
 print(row)

## Step 9: Results Summary

- Dataset: EPL 2023/24 betting odds (225 matches)  
- Binary Task: Predict Home Win (1) vs Draw/Away (0)  
  
| Model | AUC | Accuracy |  
|--------------------|-------|----------|  
| Logistic Regression| 0.730 | 0.721 |  
| Random Forest | 0.672 | 0.559 |  
| Decision Tree | 0.569 | 0.574 |  
| Gradient Boosted | 0.550 | 0.574 |  
  
- Best Model (CV): Random Forest (50 trees, depth 5) with AUC ≈ 0.680  
- Key Features: B365H > B365D > B365A  
- Platform: Databricks (2 × i3.xlarge, 8 vCPUs, 61 GiB RAM)  
- Total Runtime: Less than 5 minutes  
- Metrics met the 70% accuracy/AUC goal