# Objectives Part 1 – Code Walkthrough

This section outlines the full pipeline for loading, preparing, and modeling soccer betting data to predict match outcomes using Apache Spark and Logistic Regression.

## Step 1: Load Dataset from GitHub

We load the cleaned dataset directly from GitHub using Pandas, and then convert it to a Spark DataFrame for further processing.

import pandas as pd  
  
# Load directly from GitHub  
url = "https://raw.githubusercontent.com/ryumatsu/soccer-betting-ml/main/E0\_cleaned.csv"  
df = pd.read\_csv(url)  
  
# Convert to Spark  
spark\_df = spark.createDataFrame(df)  
spark\_df.show(5)

## Step 2: Register DataFrame as Temporary SQL View

Allows us to use SQL queries for inspection and feature selection.

temp\_table\_name = "E0\_cleaned\_csv"  
spark\_df.createOrReplaceTempView(temp\_table\_name)

%sql  
SELECT \* FROM `E0\_cleaned\_csv`

## Step 3: Feature Engineering and Label Preparation

We cast betting odds to float and convert the label to binary (Home Win = 1, else = 0).

from pyspark.sql.functions import col, when  
  
spark\_df = spark\_df.select(  
 col("B365H").cast("float"),  
 col("B365D").cast("float"),  
 col("B365A").cast("float"),  
 col("Label").cast("int")  
).dropna()  
  
spark\_df = spark\_df.withColumn("Label", when(col("Label") == 1, 1).otherwise(0))  
spark\_df.groupBy("Label").count().show()

## Step 4: Assemble Features

Use VectorAssembler to bundle the odds columns into a single feature vector.

from pyspark.ml.feature import VectorAssembler  
  
assembler = VectorAssembler(  
 inputCols=["B365H", "B365D", "B365A"],  
 outputCol="features"  
)  
  
assembled = assembler.transform(spark\_df)  
assembled.select("features", "Label").show(5)

## Step 5: Train-Test Split

Split the dataset into 80% training and 20% test data.

train\_data, test\_data = assembled.randomSplit([0.8, 0.2], seed=42)

## Step 6: Train Logistic Regression Model

from pyspark.ml.classification import LogisticRegression  
  
lr = LogisticRegression(featuresCol="features", labelCol="Label", maxIter=10)  
lr\_model = lr.fit(train\_data)

## Step 7: Make Predictions and Evaluate Accuracy

predictions = lr\_model.transform(test\_data)  
predictions.select("features", "Label", "prediction").show(5)  
  
from pyspark.ml.evaluation import MulticlassClassificationEvaluator  
  
evaluator = MulticlassClassificationEvaluator(  
 labelCol="Label",  
 predictionCol="prediction",  
 metricName="accuracy"  
)  
  
accuracy = evaluator.evaluate(predictions)  
print(f"Test Accuracy: {accuracy:.2f}")

## Step 8: Generate Confusion Matrix

predictions.groupBy("Label", "prediction").count().orderBy("Label", "prediction").show()