# Machine Learning with Spark MLIB and PySpark

## Step 1: Log in to Worker Nodes and Install the Library

1. Log in to Worker Node 1:  
docker-compose exec worker1 bash

2. Install the Library using pip3:  
pip3 install numpy

3. Exit Worker Node 1:  
exit

4. Log in to Worker Node 2:  
docker-compose exec worker2 bash

5. Install the Library using pip3:  
pip3 install numpy

6. Exit Worker Node 2:  
exit

## Step 2: Log in to the Master Node and Install the Library

1. Log in to the Master Node:  
docker-compose exec master bash

2. Install the Library using pip3:  
pip3 install numpy

## Step 3: Load grades.csv into HDFS

hdfs dfs -put /data/grades.csv /

## Step 3: Load grades.csv into Hive

1. Open Hive Shell:

hive

2. Create the Hive Table:

CREATE TABLE gradesml (  
 last\_name STRING,  
 first\_name STRING,  
 ssn STRING,  
 test1 INT,  
 test2 INT,  
 test3 INT,  
 test4 INT,  
 final\_score INT,  
 grade STRING  
)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
STORED AS TEXTFILE;

3. Load Data from CSV into the Hive Table:

LOAD DATA INPATH '/grades.csv' INTO TABLE gradesml;

## Step 4: Use Spark to Run Machine Learning (MLlib)

1. Enter pyspark

pyspark

2. Here’s the PySpark code to use test scores to predict the final score using MLlib (Linear Regression):

from pyspark.sql import SparkSession  
from pyspark.ml.feature import VectorAssembler  
from pyspark.ml.regression import LinearRegression  
  
# Step 1: Create a Spark session  
spark = SparkSession.builder.appName("MLlib GradesML Prediction").enableHiveSupport().getOrCreate()  
  
# Step 2: Load the data from the Hive table 'gradesml' into a Spark DataFrame  
grades\_df = spark.sql("SELECT test1, test2, test3, test4, final\_score FROM gradesml")  
  
# Step 3: Handle null values by either dropping or filling them  
grades\_df = grades\_df.na.drop() # Drop rows with null values  
  
# Step 4: Prepare the data for MLlib by assembling features into a vector  
assembler = VectorAssembler(  
 inputCols=["test1", "test2", "test3", "test4"],  
 outputCol="features",  
 handleInvalid="skip" # Skip rows with null values  
)  
assembled\_df = assembler.transform(grades\_df).select("features", "final\_score")  
  
# Step 5: Split the data into training and testing sets  
train\_data, test\_data = assembled\_df.randomSplit([0.7, 0.3])  
  
# Step 6: Initialize and train a Linear Regression model  
lr = LinearRegression(labelCol="final\_score")  
lr\_model = lr.fit(train\_data)  
  
# Step 7: Evaluate the model on the test data  
test\_results = lr\_model.evaluate(test\_data)  
  
# Step 8: Print the model performance metrics  
print(f"RMSE: {test\_results.rootMeanSquaredError}")  
print(f"R^2: {test\_results.r2}")  
  
# Step 9: Stop the Spark session  
spark.stop()