


```
!pip install pyspark
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder \
    .appName("RetailMLModel") \
    .getOrCreate()
```


Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.4)  
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (0.10.9.7)

```
from google.colab import files
uploaded = files.upload()
```

  Online.csv  
**Online.csv**(application/vnd.ms-excel) - 45580670 bytes, last modified: n/a - 100% done  
Saving Online.csv to Online.csv


```
import pandas as pd
```

```
# Read using pandas for preview
df = pd.read_csv("Online.csv")
df.head()
```



	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	C
0	536365	85123A	WHITE HANGING HEART T- LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	K
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	K

```
spark_df = spark.createDataFrame(df)
spark_df.printSchema()
spark_df.show(5)
```

 root

```
|-- InvoiceNo: string (nullable = true)
|-- StockCode: string (nullable = true)
|-- Description: string (nullable = true)
|-- Quantity: long (nullable = true)
|-- InvoiceDate: string (nullable = true)
```

```
-- UnitPrice: double (nullable = true)
-- CustomerID: double (nullable = true)
-- Country: string (nullable = true)
```

```
+-----+-----+-----+-----+-----+-----+
|InvoiceNo|StockCode|Description|Quantity|InvoiceDate|UnitPrice|CustomerI
+-----+-----+-----+-----+-----+-----+
| 536365| 85123A|WHITE HANGING HEA...| 6|12/1/2010 8:26| 2.55| 17850.
| 536365| 71053| WHITE METAL LANTERN| 6|12/1/2010 8:26| 3.39| 17850.
| 536365| 84406B|CREAM CUPID HEART...| 8|12/1/2010 8:26| 2.75| 17850.
| 536365| 84029G|KNITTED UNION FLA...| 6|12/1/2010 8:26| 3.39| 17850.
| 536365| 84029E|RED WOOLLY HOTTIE...| 6|12/1/2010 8:26| 3.39| 17850.
+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
from pyspark.sql.functions import col
```

```
spark_df = spark_df.withColumn("TotalValue", col("Quantity") * col("UnitPrice"))
```

```
spark_df = spark_df.na.drop()
```

```
from pyspark.ml.feature import VectorAssembler
```

```
assembler = VectorAssembler(
    inputCols=["Quantity", "UnitPrice"],
    outputCol="features"
)
```

```
data = assembler.transform(spark_df).select("features", "TotalValue")
data.show(5)
```



```
+-----+-----+
| features|TotalValue|
+-----+-----+
|[6.0,2.55]|15.299999999999999|
|[6.0,3.39]| 20.34|
|[8.0,2.75]| 22.0|
|[6.0,3.39]| 20.34|
|[6.0,3.39]| 20.34|
+-----+-----+
only showing top 5 rows
```

```
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
```

```

from pyspark.ml.regression import LinearRegression

lr = LinearRegression(featuresCol="features", labelCol="TotalValue")
model = lr.fit(train_data)

predictions = model.transform(test_data)
predictions.select("TotalValue", "prediction").show(10)

```

TotalValue	prediction
-6539.400000000001	-4351.476826467478
-216.0	-1882.1603000138573
-201.6	-1336.6126888202634
-835.1999999999999	-800.427110730613
-216.0	-413.72442820493904
-1188.0	-333.22994521817577
-138.24	-262.6384377455033
-103.67999999999999	-195.48910865242075
-302.40000000000003	-196.6120888329962
-102.0	-162.0202320939047

only showing top 10 rows

```

from pyspark.ml.evaluation import RegressionEvaluator

evaluator = RegressionEvaluator(
    labelCol="TotalValue", predictionCol="prediction", metricName="rmse"
)

rmse = evaluator.evaluate(predictions)
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

```

Root Mean Squared Error (RMSE): 203.40

```

assembler = VectorAssembler(inputCols=["Quantity", "UnitPrice"], outputCol="features")
lr = LinearRegression(featuresCol="features", labelCol="TotalValue")
model = lr.fit(train_data)

```

```

rmse = evaluator.evaluate(predictions)
print(f"RMSE: {rmse}")

```

RMSE: 203.39757647210158

