databricks Bike Accessment

Import Data

```
from pyspark.sql import SparkSession
sc = SparkSession.builder.master("local[*]").getOrCreate()
```

```
data = spark.read.csv("/FileStore/tables/Bike_Rental_UCI_dataset-
bb6c6.csv",inferSchema=True,header=True)
```

```
data.show()
```

```
+------+---+----+---+-------+----------+----------+----+----+---------+-----
----+----+------+
|season| yr|mnth| hr|holiday|workingday|weathersit|temp| hum|windspeed|dayOf
Week|days|demand|
+------+---+----+---+-------+----------+----------+----+----+---------+-----
----+----+------+
|     1|  0|   1|  0|      0|         0|         1|0.24|0.81|      0.0|
Sat|   0|    16|
|     1|  0|   1|  1|      0|         0|         1|0.22| 0.8|      0.0|
Sat|   0|    40|
|     1|  0|   1|  2|      0|         0|         1|0.22| 0.8|      0.0|
Sat|   0|    32|
|     1|  0|   1|  3|      0|         0|         1|0.24|0.75|      0.0|
Sat|   0|    13|
|     1|  0|   1|  4|      0|         0|         1|0.24|0.75|      0.0|
Sat|   0|     1|
|     1|  0|   1|  5|      0|         0|         2|0.24|0.75|   0.0896|
Sat|   0|     1|
|     1|  0|   1|  6|      0|         0|         1|0.22| 0.8|      0.0|
Sat|   0|     2|
|     1|  0|   1|  7|      0|         0|         1| 0.2|0.86|      0.0|
Sat|   0|     3|
|     1|  0|   1|  8|      0|         0|         1|0.24|0.75|      0.0|
Sat|   0|     8|
|     1|  0|   1|  9|      0|         0|         1|0.32|0.76|      0.0|
Sat|   0|    14|
|     1|  0|   1| 10|      0|         0|         1|0.38|0.76|   0.2537|
Sat|   0|    36|
|     1|  0|   1| 11|      0|         0|         1|0.36|0.81|   0.2836|
Sat|   0|    56|
|     1|  0|   1| 12|      0|         0|         1|0.42|0.77|   0.2836|
Sat|   0|    84|
|     1|  0|   1| 13|      0|         0|         2|0.46|0.72|   0.2985|
Sat|   0|    94|
|     1|  0|   1| 14|      0|         0|         2|0.46|0.72|   0.2836|     
Sat|   0|   106|
|     1|  0|   1| 15|      0|         0|         2|0.44|0.77|   0.2985|     
Sat|   0|   110|
|     1|  0|   1| 16|      0|         0|         2|0.42|0.82|   0.2985|     
Sat|   0|    93|
|     1|  0|   1| 17|      0|         0|         2|0.44|0.82|   0.2836|     
Sat|   0|    67|
|     1|  0|   1| 18|      0|         0|         3|0.42|0.88|   0.2537|     
Sat|   0|    35|
|     1|  0|   1| 19|      0|         0|         3|0.42|0.88|   0.2537|     
Sat|   0|    37|
+------+---+----+---+-------+----------+----------+----+----+---------+-----
----+----+------+
only showing top 20 rows
```

Feature Engineering

```
from pyspark.ml.classification import
(LogisticRegression,DecisionTreeClassifier,RandomForestClassifier)
from pyspark.ml import Pipeline
from pyspark.ml.tuning import ParamGridBuilder,CrossValidator
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.feature import StringIndexer


indexer = StringIndexer(inputCol="dayOfWeek",outputCol="day_cat")


indexed_data = indexer.fit(data).transform(data)
indexed_data.show()
```

```
+------+---+----+---+-------+----------+----------+----+----+---------+-----
----+----+------+-------+
|season| yr|mnth| hr|holiday|workingday|weathersit|temp| hum|windspeed|dayOf
Week|days|demand|day_cat|
+------+---+----+---+-------+----------+----------+----+----+---------+-----
----+----+------+-------+
|     1|  0|   1|  0|      0|         0|         1|0.24|0.81|      0.0|     
Sat|   0|    16|    0.0|
|     1|  0|   1|  1|      0|         0|         1|0.22| 0.8|      0.0|     
Sat|   0|    40|    0.0|
|     1|  0|   1|  2|      0|         0|         1|0.22| 0.8|      0.0|     
Sat|   0|    32|    0.0|
|     1|  0|   1|  3|      0|         0|         1|0.24|0.75|      0.0|     
Sat|   0|    13|    0.0|
|     1|  0|   1|  4|      0|         0|         1|0.24|0.75|     0.0|
Sat|   0|     1|    0.0|
|     1|  0|   1|  5|      0|         0|         2|0.24|0.75|  0.0896|
Sat|   0|     1|    0.0|
|     1|  0|   1|  6|      0|         0|         1|0.22| 0.8|     0.0|
Sat|   0|     2|    0.0|
|     1|  0|   1|  7|      0|         0|         1| 0.2|0.86|     0.0|
Sat|   0|     3|    0.0|
|     1|  0|   1|  8|      0|         0|         1|0.24|0.75|     0.0|
Sat|   0|     8|    0.0|
|     1|  0|   1|  9|      0|         0|         1|0.32|0.76|     0.0|
Sat|   0|    14|    0.0|
|     1|  0|   1| 10|      0|         0|         1|0.38|0.76|  0.2537|
Sat|   0|    36|    0.0|
|     1|  0|   1| 11|      0|         0|         1|0.36|0.81|  0.2836|
Sat|   0|    56|    0.0|
|     1|  0|   1| 12|      0|         0|         1|0.42|0.77|  0.2836|
Sat|   0|    84|    0.0|
|     1|  0|   1| 13|      0|         0|         2|0.46|0.72|  0.2985|
Sat|   0|    94|    0.0|
|     1|  0|   1| 14|      0|         0|         2|0.46|0.72|  0.2836|
Sat|   0|   106|    0.0|
|     1|  0|   1| 15|      0|         0|         2|0.44|0.77|  0.2985|
Sat|   0|   110|    0.0|
|     1|  0|   1| 16|      0|         0|         2|0.42|0.82|  0.2985|
Sat|   0|    93|    0.0|
|     1|  0|   1| 17|      0|         0|         2|0.44|0.82|  0.2836|
Sat|   0|    67|    0.0|
|     1|  0|   1| 18|      0|         0|         3|0.42|0.88|  0.2537|
Sat|   0|    35|    0.0|
|     1|  0|   1| 19|      0|         0|         3|0.42|0.88|  0.2537|
Sat|   0|    37|    0.0|
+------+---+----+---+-------+----------+----------+----+----+--------+-----
----+----+------+-------+
only showing top 20 rows
```

```
indexed_data.select('day_cat').distinct().orderBy('day_cat').show()
```

```
+-------+
|day_cat|
+-------+
|    0.0|
|    1.0|
|    2.0|
|    3.0|
|    4.0|
|    5.0|
|    6.0|
+-------+
from pyspark.ml.feature import VectorAssembler

vec = VectorAssembler(inputCols=
['season','yr','mnth','hr','holiday','workingday','weathersit','temp','hum',
'windspeed','days','day_cat'],outputCol='features')


data1 = vec.transform(indexed_data)


data1.show()

+------+---+----+---+-------+----------+----------+----+----+---------+-----
----+----+------+-------+--------------------+
|season| yr|mnth| hr|holiday|workingday|weathersit|temp| hum|windspeed|dayOf
Week|days|demand|day_cat|            features|
+------+---+----+---+-------+----------+----------+----+----+---------+-----
----+----+------+-------+--------------------+
|     1|  0|   1|  0|      0|         0|         1|0.24|0.81|      0.0|
Sat|   0|    16|    0.0|(12,[0,2,6,7,8],[...|
|     1|  0|   1|  1|      0|         0|         1|0.22| 0.8|      0.0|
Sat|   0|    40|    0.0|(12,[0,2,3,6,7,8]...|
|     1|  0|   1|  2|      0|         0|         1|0.22| 0.8|      0.0|
Sat|   0|    32|    0.0|(12,[0,2,3,6,7,8]...|
|     1|  0|   1|  3|      0|         0|         1|0.24|0.75|      0.0|
Sat|   0|    13|    0.0|(12,[0,2,3,6,7,8]...|
|     1|  0|   1|  4|      0|         0|         1|0.24|0.75|      0.0|
Sat|   0|     1|    0.0|(12,[0,2,3,6,7,8]...|
|     1|  0|   1|  5|      0|         0|         2|0.24|0.75|   0.0896|
Sat|   0|     1|    0.0|[1.0,0.0,1.0,5.0,...|
|     1|  0|   1|  6|      0|         0|         1|0.22| 0.8|      0.0|
Sat|   0|     2|    0.0|(12,[0,2,3,6,7,8]...|
|     1|  0|   1|  7|      0|         0|         1| 0.2|0.86|      0.0|
Sat|   0|     3|    0.0|(12,[0,2,3,6,7,8]...|
|     1|  0|   1|  8|      0|         0|         1|0.24|0.75|      0.0|
Sat|   0|     8|    0.0|(12,[0,2,3,6,7,8]...|
|     1|  0|   1|  9|      0|         0|         1|0.32|0.76|      0.0|
Sat|   0|    14|    0.0|(12,[0,2,3,6,7,8]...|
|     1|  0|   1| 10|      0|         0|         1|0.38|0.76|   0.2537|
Sat|   0|    36|    0.0|[1.0,0.0,1.0,10.0...|
|     1|  0|   1| 11|      0|         0|         1|0.36|0.81|   0.2836|
Sat|   0|    56|    0.0|[1.0,0.0,1.0,11.0...|
|     1|  0|   1| 12|      0|         0|         1|0.42|0.77|   0.2836|
Sat|   0|    84|    0.0|[1.0,0.0,1.0,12.0...|
|     1|  0|   1| 13|      0|         0|         2|0.46|0.72|   0.2985|
Sat|   0|    94|    0.0|[1.0,0.0,1.0,13.0...|
|     1|  0|   1| 14|      0|         0|         2|0.46|0.72|   0.2836|
Sat|   0|   106|    0.0|[1.0,0.0,1.0,14.0...|
|     1|  0|   1| 15|      0|         0|         2|0.44|0.77|   0.2985|
Sat|   0|   110|    0.0|[1.0,0.0,1.0,15.0...|
|     1|  0|   1| 16|      0|        0|         2|0.42|0.82|   0.2985|
Sat|   0|    93|    0.0|[1.0,0.0,1.0,16.0...|
|     1|  0|   1| 17|      0|        0|         2|0.44|0.82|   0.2836|
Sat|   0|    67|    0.0|[1.0,0.0,1.0,17.0...|
|     1|  0|   1| 18|      0|        0|         3|0.42|0.88|   0.2537|
Sat|   0|    35|    0.0|[1.0,0.0,1.0,18.0...|
|     1|  0|   1| 19|      0|        0|         3|0.42|0.88|   0.2537|
Sat|   0|    37|    0.0|[1.0,0.0,1.0,19.0...|
+------+---+----+---+-------+---------+----------+----+----+---------+-----
----+----+------+-------+--------------------+
only showing top 20 rows
```

```
modelData = data1.select('features','demand')
```

```
modelData.describe().show()
```

```
+-------+------------------+
|summary|            demand|
+-------+------------------+
|  count|             17379|
|   mean|189.46308763450142|
| stddev| 181.3875990918646|
|    min|                 1|
|    max|               977|
+-------+------------------+
```

```
modelData.show(truncate=False)
```

```
+-------------------------------------------------------+------+
|features                                               |demand|
+-------------------------------------------------------+------+
|(12,[0,2,6,7,8],[1.0,1.0,1.0,0.24,0.81])               |16    |
|(12,[0,2,3,6,7,8],[1.0,1.0,1.0,1.0,0.22,0.8])          |40    |
|(12,[0,2,3,6,7,8],[1.0,1.0,2.0,1.0,0.22,0.8])          |32    |
|(12,[0,2,3,6,7,8],[1.0,1.0,3.0,1.0,0.24,0.75])         |13    |
|(12,[0,2,3,6,7,8],[1.0,1.0,4.0,1.0,0.24,0.75])         |1     |
|[1.0,0.0,1.0,5.0,0.0,0.0,2.0,0.24,0.75,0.0896,0.0,0.0] |1     |
|(12,[0,2,3,6,7,8],[1.0,1.0,6.0,1.0,0.22,0.8])          |2     |
|(12,[0,2,3,6,7,8],[1.0,1.0,7.0,1.0,0.2,0.86])          |3     |
|(12,[0,2,3,6,7,8],[1.0,1.0,8.0,1.0,0.24,0.75])         |8     |
|(12,[0,2,3,6,7,8],[1.0,1.0,9.0,1.0,0.32,0.76])         |14    |
|[1.0,0.0,1.0,10.0,0.0,0.0,1.0,0.38,0.76,0.2537,0.0,0.0]|36    |
|[1.0,0.0,1.0,11.0,0.0,0.0,1.0,0.36,0.81,0.2836,0.0,0.0]|56    |
|[1.0,0.0,1.0,12.0,0.0,0.0,1.0,0.42,0.77,0.2836,0.0,0.0]|84    |
|[1.0,0.0,1.0,13.0,0.0,0.0,2.0,0.46,0.72,0.2985,0.0,0.0]|94    |
|[1.0,0.0,1.0,14.0,0.0,0.0,2.0,0.46,0.72,0.2836,0.0,0.0]|106   |
|[1.0,0.0,1.0,15.0,0.0,0.0,2.0,0.44,0.77,0.2985,0.0,0.0]|110   |
|[1.0,0.0,1.0,16.0,0.0,0.0,2.0,0.42,0.82,0.2985,0.0,0.0]|93    |
|[1.0,0.0,1.0,17.0,0.0,0.0,2.0,0.44,0.82,0.2836,0.0,0.0]|67    |
|[1.0,0.0,1.0,18.0,0.0,0.0,3.0,0.42,0.88,0.2537,0.0,0.0]|35    |
|[1.0,0.0,1.0,19.0,0.0,0.0,3.0,0.42,0.88,0.2537,0.0,0.0]|37    |
+-------------------------------------------------------+------+
only showing top 20 rows
```

Seperate data for Train & Test

```
trainData,testData = modelData.randomSplit([0.7,0.3])
```

```
trainData = trainData.withColumnRenamed(('demand'),('label'))
```

```
testData = testData.withColumnRenamed(('demand'), ('label'))
```

```
trainData.show(truncate=False)
```

```
+--------------------------------------------------------------------+-----+
|features                                                            |label|
+--------------------------------------------------------------------+-----+
|(12,[0,2,3,6,7,8],[1.0,1.0,1.0,1.0,0.22,0.8])                       |40   |
|(12,[0,2,3,6,7,8],[1.0,1.0,2.0,1.0,0.22,0.8])                       |32   |
|(12,[0,2,3,6,7,8],[1.0,1.0,3.0,1.0,0.24,0.75])                      |13   |
|(12,[0,2,3,6,7,8],[1.0,1.0,6.0,1.0,0.22,0.8])                       |2    |
|(12,[0,2,3,6,7,8],[1.0,1.0,8.0,1.0,0.24,0.75])                      |8    |
|(12,[0,2,3,6,7,8],[1.0,1.0,9.0,1.0,0.32,0.76])                      |14   |
|(12,[0,2,6,7,8,10],[1.0,1.0,1.0,0.18,0.55,13.0])                    |28   |
|(12,[0,2,6,7,8,10],[2.0,4.0,2.0,0.3,0.61,87.0])                     |32   |
|(12,[0,2,6,7,8,10],[2.0,6.0,1.0,0.56,0.52,150.0])                   |93   |
|(12,[0,2,6,7,8,10],[3.0,7.0,1.0,0.82,0.56,199.0])                   |101  |
|(12,[0,2,6,7,8,10],[3.0,8.0,1.0,0.6,0.88,227.0])                    |128  |
|(12,[0,2,6,7,8,10],[4.0,10.0,1.0,0.48,0.72,275.0])                  |89   |
|[1.0,0.0,1.0,0.0,0.0,0.0,1.0,0.04,0.45,0.2537,19.0,0.0]             |13   |
|[1.0,0.0,1.0,0.0,0.0,0.0,1.0,0.04,0.57,0.1045,20.0,1.0]             |22   |
|[1.0,0.0,1.0,0.0,0.0,0.0,1.0,0.22,0.64,0.3582,25.0,0.0]             |28   |
|[1.0,0.0,1.0,0.0,0.0,0.0,1.0,0.26,0.56,0.0,14.0,1.0]                |39   |
|[1.0,0.0,1.0,0.0,0.0,1.0,1.0,0.06,0.41,0.19399999999999998,21.0,3.0]|7    |
|[1.0,0.0,1.0,0.0,0.0,1.0,1.0,0.12,0.5,0.19399999999999998,12.0,2.0] |14   |
|[1.0,0.0,1.0,0.0,0.0,1.0,1.0,0.12,0.5,0.2836,8.0,3.0]               |5    |
|[1.0,0.0,1.0,0.0,0.0,1.0,1.0,0.14,0.59,0.2836,11.0,5.0]             |7    |
+--------------------------------------------------------------------+-----+
only showing top 20 rows
```

```
testData.show(truncate=False)
```

```
+-------------------------------------------------------+-----+
|features                                               |label|
+-------------------------------------------------------+-----+
|(12,[0,2,3,6,7,8],[1.0,1.0,4.0,1.0,0.24,0.75])         |1    |
|(12,[0,2,3,6,7,8],[1.0,1.0,7.0,1.0,0.2,0.86])          |3    |
|(12,[0,2,6,7,8],[1.0,1.0,1.0,0.24,0.81])               |16   |
|(12,[0,2,6,7,8,10],[3.0,9.0,1.0,0.62,0.94,247.0])      |116  |
|[1.0,0.0,1.0,0.0,0.0,0.0,1.0,0.1,0.42,0.3881,7.0,1.0]  |25   |
|[1.0,0.0,1.0,0.0,0.0,0.0,1.0,0.16,0.8,0.1045,26.0,1.0] |33   |
|[1.0,0.0,1.0,0.0,0.0,0.0,2.0,0.18,0.51,0.1642,6.0,0.0] |25   |
|[1.0,0.0,1.0,0.0,0.0,0.0,2.0,0.46,0.88,0.2985,1.0,1.0] |17   |
|[1.0,0.0,1.0,0.0,0.0,1.0,1.0,0.14,0.59,0.1045,9.0,6.0] |12   |
|[1.0,0.0,1.0,0.0,0.0,1.0,1.0,0.26,0.56,0.3881,17.0,5.0]|13   |
|[1.0,0.0,1.0,0.0,0.0,1.0,2.0,0.2,0.75,0.1343,24.0,2.0] |9    |
|[1.0,0.0,1.0,0.0,0.0,1.0,2.0,0.22,0.93,0.0,17.0,4.0]   |3    |
|[1.0,0.0,1.0,1.0,0.0,0.0,1.0,0.14,0.8,0.0,26.0,1.0]    |29   |
|[1.0,0.0,1.0,1.0,0.0,0.0,2.0,0.16,0.59,0.0896,13.0,0.0]|20   |
|[1.0,0.0,1.0,1.0,0.0,1.0,2.0,0.2,0.69,0.2239,5.0,2.0]  |7    |
|[1.0,0.0,1.0,1.0,0.0,1.0,2.0,0.24,0.65,0.1343,23.0,4.0]|5    |
|[1.0,0.0,1.0,1.0,0.0,1.0,3.0,0.22,0.93,0.1343,17.0,4.0]|7    |
|[1.0,0.0,1.0,2.0,0.0,0.0,2.0,0.18,0.55,0.0,6.0,0.0]    |16   |
|[1.0,0.0,1.0,2.0,0.0,1.0,1.0,0.14,0.86,0.1343,10.0,4.0]|1    |
|[1.0,0.0,1.0,2.0,0.0,1.0,1.0,0.16,0.64,0.0,4.0,5.0]    |2    |
+-------------------------------------------------------+-----+
only showing top 20 rows
```

Linear Regression

```
from pyspark.ml.regression import LinearRegression
lr = LinearRegression(featuresCol ='features', labelCol='label', maxIter=10,
regParam=0.3, elasticNetParam=1)
lr_model = lr.fit(trainData)
print("Coefficients: " + str(lr_model.coefficients))
print("Intercept: " + str(lr_model.intercept))
trainingSummary = lr_model.summary
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)
```

```
Coefficients: [19.6914190944,53.0943621795,-2.09734576019,7.56361284259,-18.
6125249833,4.89479402101,-3.94093165934,282.052218461,-201.599320208,30.2210
596181,0.0748403062638,-0.265478966667]
Intercept: -2.5016762683676923
RMSE: 142.436392
r2: 0.385282
```

```
# Make predictions.
predictions = lr_model.transform(testData)

# Select example rows to display.
predictions.select("prediction", "label", "features").show(5)

# Select (prediction, true label) and compute test error
evaluator1 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="rmse")
rmse = evaluator1.evaluate(predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
evaluator2 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="r2")
r2 = evaluator2.evaluate(predictions)
print("R2 on test data = %g" % r2)
```

```
+------------------+-----+--------------------+
|        prediction|label|            features|
+------------------+-----+--------------------+
|-42.101040948732376|    1|(12,[0,2,3,6,7,8]...|
|  -52.8682163823581|    3|(12,[0,2,3,6,7,8]...|
| -84.45145153157792|   16|(12,[0,2,6,7,8],[...|
|  37.61010761104589|  116|(12,[0,2,6,7,8,10...|
| -33.32783081995555|   25|[1.0,0.0,1.0,0.0,...|
+------------------+-----+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 140.911
R2 on test data = 0.39197
```

LR Cross Validation

```
from pyspark.ml import Pipeline
from pyspark.ml.tuning import ParamGridBuilder,CrossValidator
from pyspark.ml.evaluation import RegressionEvaluator
```

```
evaluator = RegressionEvaluator(metricName = 'r2')
```

```
evaluator.explainParam('metricName')
```

```
Out[118]: 'metricName: metric name in evaluation - one of:\n
rmse - root mean squared error (default)\n                      mse - mean
 squared error\n                       r2 - r^2 metric\n
mae - mean absolute error. (default: rmse, current: r2)'
```

```
pipeline = Pipeline(stages=[lr])
```

```
gridBuilder=ParamGridBuilder()\
  .addGrid(lr.regParam, [0.1, 0.01]) \
  .build()
```

```
cv =
CrossValidator(estimator=pipeLine,estimatorParamMaps=gridBuilder,evaluator=e
valuator,numFolds=2)
```

```
cvm = cv.fit(trainData)
```

```
predictions = cvm.transform(testData)
```

```
evaluator.evaluate(predictions)
```

```
Out[124]: 0.10068787153472136
```

The result is unexpected, we need more time to investigate the reason.

Display with group by

```
data1.groupby('season').mean('demand').show()
```

```
+------+------------------+
|season|       avg(demand)|
+------+------------------+
|     1|111.11456859971712|
|     3|236.01623665480426|
|     4|198.86885633270322|
|     2|208.34406894987526|
+------+------------------+
```

```
data1.groupby('hr').mean('demand').show()
```

```
+---+------------------+
| hr|       avg(demand)|
+---+------------------+
| 12|253.31593406593407|
| 22|131.33516483516485|
|  1|  33.3756906077348|
| 13|253.66117969821673|
|  6| 76.04413793103448|
| 16| 311.9835616438356|
|  3|11.727403156384504|
| 20|226.03021978021977|
|  5| 19.88981868898187|
| 19|311.52335164835165|
| 15| 251.2331961591221|
|  9|219.30949105914718|
| 17|461.45205479452056|
|  4| 6.352941176470588|
|  8|359.01100412654745|
| 23| 87.83104395604396|
|  7| 212.0646492434663|
| 10| 173.6685006877579|
| 21|172.31456043956044|
| 11| 208.1430536451169|
+---+------------------+
only showing top 20 rows
```

Other Models

```
from pyspark.ml import Pipeline
from pyspark.ml.regression import DecisionTreeRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

# Train a DecisionTree model.
dt = DecisionTreeRegressor(featuresCol="features")

# Chain indexer and tree in a Pipeline
pipeline = Pipeline(stages=[dt])

# Train model.  This also runs the indexer.
model = pipeline.fit(trainData)

# Make predictions.
predictions = model.transform(testData)

# Select example rows to display.
predictions.select("prediction", "label", "features").show(5)

# Select (prediction, true label) and compute test error
evaluator1 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="rmse")
rmse = evaluator1.evaluate(predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
evaluator2 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="r2")
r2 = evaluator2.evaluate(predictions)
print("R2 on test data = %g" % r2)
```

```
+------------------+-----+--------------------+
|        prediction|label|            features|
+------------------+-----+--------------------+
|14.869767441860466|    1|(12,[0,2,3,6,7,8]...|
|             107.7|    3|(12,[0,2,3,6,7,8]...|
| 66.51470588235294|   16|(12,[0,2,6,7,8],[...|
|124.61904761904762|  116|(12,[0,2,6,7,8,10...|
| 66.51470588235294|   25|[1.0,0.0,1.0,0.0,...|
+------------------+-----+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 107.384
R2 on test data = 0.646884
```

```
from pyspark.ml import Pipeline
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

# Train a RandomForest model.
rf = RandomForestRegressor(featuresCol='features')

# Chain indexer and forest in a Pipeline
pipeline = Pipeline(stages=[rf])

# Train model.  This also runs the indexer.
model = pipeline.fit(trainData)

# Make predictions.
predictions = model.transform(testData)

# Select example rows to display.
predictions.select("prediction", "label", "features").show(5)

# Select (prediction, true label) and compute test error
evaluator1 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="rmse")
rmse = evaluator1.evaluate(predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
evaluator2 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="r2")
r2 = evaluator2.evaluate(predictions)
print("R2 on test data = %g" % r2)
```

```
+------------------+-----+--------------------+
|        prediction|label|            features|
+------------------+-----+--------------------+
| 19.70270694650672|    1|(12,[0,2,3,6,7,8]...|
| 74.45332008243942|    3|(12,[0,2,3,6,7,8]...|
|30.302337702024925|   16|(12,[0,2,6,7,8],[...|
| 67.10513669994847|  116|(12,[0,2,6,7,8,10...|
|33.211464342172775|   25|[1.0,0.0,1.0,0.0,...|
+------------------+-----+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 110.492
R2 on test data = 0.626147
```

```
from pyspark.ml import Pipeline
from pyspark.ml.regression import GBTRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

# Train a GBT model.
gbt = GBTRegressor(featuresCol='features', maxIter=50)

# Chain indexer and GBT in a Pipeline
pipeline = Pipeline(stages=[gbt])

# Train model.  This also runs the indexer.
model = pipeline.fit(trainData)

# Make predictions.
predictions = model.transform(testData)

# Select example rows to display.
predictions.select("prediction", "label", "features").show(5)

# Select (prediction, true label) and compute test error
evaluator1 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="rmse")
rmse = evaluator1.evaluate(predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
evaluator2 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="r2")
r2 = evaluator2.evaluate(predictions)
print("R2 on test data = %g" % r2)
```

```
+------------------+-----+--------------------+
|        prediction|label|            features|
+------------------+-----+--------------------+
|-13.52827471851879|    1|(12,[0,2,3,6,7,8]...|
| -44.0988006791547|    3|(12,[0,2,3,6,7,8]...|
| 28.40194416751362|   16|(12,[0,2,6,7,8],[...|
|106.60344271916942|  116|(12,[0,2,6,7,8,10...|
|29.900054705811844|   25|[1.0,0.0,1.0,0.0,...|
+------------------+-----+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 48.446
R2 on test data = 0.928129
```

For this Bike Rental Dataset, the best model is gradient-boosted tree regression, the RMSE is just 48.446 and the r2 is up to 0.928129 !