In [1]:
```
import findspark
findspark.init()
```

In [2]:
```
from pyspark.sql import SparkSession
spark=SparkSession.builder.appName("DfApp").getOrCreate()
```

In [3]:
```
df=spark.read.option("header","true").csv('D:\weight-height.csv',inferSchema=True)
df.show()
```

```
+------+-----------+-----------+
|Gender|     Height|     Weight|
+------+-----------+-----------+
|  Male|73.84701702|241.8935632|
|  Male|68.78190405|162.3104725|
|  Male|74.11010539|212.7408556|
|  Male| 71.7309784|220.0424703|
|  Male|69.88179586|206.3498006|
|  Male|67.25301569|152.2121558|
|  Male|  65.865443|183.9278886|
|Female|68.34851551|167.9711105|
|  Male|67.01894966|175.9294404|
|  Male|  56.234555|156.3996764|
|  Male|71.19538228|186.6049256|
|  Male|71.64080512|213.7411695|
|  Male|64.76632913|167.1274611|
|  Male| 65.3456677|189.4461814|
|Female|69.24373223| 186.434168|
|  Male| 67.6456197|172.1869301|
|  Male|72.41831663|196.0285063|
|  Male|63.97432572|172.8834702|
|Female| 69.6400599|185.9839576|
|  Male|  65.654333| 182.426648|
+------+-----------+-----------+
only showing top 20 rows
```

In [4]:
```
df.printSchema()
```

```
root
 |-- Gender: string (nullable = true)
 |-- Height: double (nullable = true)
 |-- Weight: double (nullable = true)
```

In [5]:
```
# Convert of (independent feature) string col to numeric col
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import VectorAssembler
```

In [6]:
```
indexer=StringIndexer(inputCol='Gender',outputCol='Gender_num')
df1=indexer.fit(df).transform(df)
df1.show()
```

```
+------+-----------+-----------+----------+
|Gender|     Height|     Weight|Gender_num|
+------+-----------+-----------+----------+
|  Male|73.84701702|241.8935632|       1.0|
|  Male|68.78190405|162.3104725|       1.0|
|  Male|74.11010539|212.7408556|       1.0|
|  Male| 71.7309784|220.0424703|       1.0|
|  Male|69.88179586|206.3498006|       1.0|
|  Male|67.25301569|152.2121558|       1.0|
|  Male|  65.865443|183.9278886|       1.0|
|Female|68.34851551|167.9711105|       0.0|
|  Male|67.01894966|175.9294404|       1.0|
|  Male|  56.234555|156.3996764|       1.0|
|  Male|71.19538228|186.6049256|       1.0|
|  Male|71.64080512|213.7411695|       1.0|
|  Male|64.76632913|167.1274611|       1.0|
|  Male| 65.3456677|189.4461814|       1.0|
|Female|69.24373223| 186.434168|       0.0|
|  Male| 67.6456197|172.1869301|       1.0|
|  Male|72.41831663|196.0285063|       1.0|
|  Male|63.97432572|172.8834702|       1.0|
|Female| 69.6400599|185.9839576|       0.0|
|  Male|  65.654333| 182.426648|       1.0|
+------+-----------+-----------+----------+
only showing top 20 rows
```

In [7]:
```
df1.count()
```

Out[7]:
```
10000
```

In [8]:
```
df1.na.drop(how='any')
```

Out[8]:
```
DataFrame[Gender: string, Height: double, Weight: double, Gender_num: double]
```

In [9]:
```
df1.count()
```

Out[9]:
```
10000
```

In [10]:
```
featureAss=VectorAssembler(inputCols=['Height','Gender_num'],outputCol='inputfeatures')
outputframe=featureAss.transform(df1)
outputframe.show(5,False)
```

```
+------+-----------+-----------+----------+-----------------+
|Gender|Height     |Weight     |Gender_num|inputfeatures    |
+------+-----------+-----------+----------+-----------------+
|Male  |73.84701702|241.8935632|1.0       |[73.84701702,1.0]|
|Male  |68.78190405|162.3104725|1.0       |[68.78190405,1.0]|
|Male  |74.11010539|212.7408556|1.0       |[74.11010539,1.0]|
|Male  |71.7309784 |220.0424703|1.0       |[71.7309784,1.0] |
|Male  |69.88179586|206.3498006|1.0       |[69.88179586,1.0]|
+------+-----------+-----------+----------+-----------------+
only showing top 5 rows
```

In [11]:
```
finaldata=outputframe.select('inputfeatures','Weight')
finaldata.show(5,False)
```

```
+-----------------+-----------+
|inputfeatures    |Weight     |
+-----------------+-----------+
|[73.84701702,1.0]|241.8935632|
|[68.78190405,1.0]|162.3104725|
|[74.11010539,1.0]|212.7408556|
|[71.7309784,1.0] |220.0424703|
|[69.88179586,1.0]|206.3498006|
+-----------------+-----------+
only showing top 5 rows
```

In [12]:
```
finaldata.printSchema()
```

```
root
 |-- inputfeatures: vector (nullable = true)
 |-- Weight: double (nullable = true)
```

In [13]:
```
train,test=finaldata.randomSplit([.70,.30])
```

In [14]:
```
from pyspark.ml.regression import LinearRegression
lr=LinearRegression(featuresCol='inputfeatures',labelCol='Weight')
lrt=lr.fit(train)
```

In [15]:
```
print(lrt.coefficients)
print(lrt.intercept)
```

```
[5.874750924403333,19.768394298239173]
-238.24743210526913
```

In [16]:
```
res=lrt.evaluate(test)
res.predictions.show(5,False)
```

```
C:\Users\user\anaconda3\lib\site-packages\pyspark\sql\context.py:125: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
+-----------------+-----------+-----------------+
|inputfeatures    |Weight     |prediction       |
+-----------------+-----------+-----------------+
|[54.26313333,0.0]|64.70012671|80.53496058616969|
|[55.14855736,0.0]|88.81241211|85.73660622249011|
|[55.73973682,0.0]|108.1219685|89.20963830402434|
|[55.85121382,0.0]|103.7671373|89.86453791282406|
|[55.97919788,0.0]|85.41753362|90.61641238761797|
+-----------------+-----------+-----------------+
only showing top 5 rows
```

In [17]:
```
print("R2",res.r2)
print("Mean Absolute Error is",res.meanAbsoluteError)
print("Root Mean Square Error(RMSE) is",res.rootMeanSquaredError)
```

```
R2 0.8995479948065372
Mean Absolute Error is 8.2483495480659
Root Mean Square Error(RMSE) is 10.359710317841316
```

In [ ]:
```
```