```
#Name: Glory Joe-Ibekwe
#Student number: 2144448
```

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

```
!wget -q https://archive.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz
```

```
!tar xf spark-3.2.0-bin-hadoop3.2.tgz
```

```
!pip install -q findspark
```

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.2.0-bin-hadoop3.2"
```

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

```
#Simple Linear Regression Model
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
{}
```

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
dataset = spark.read.csv('BostonHousing.csv', inferSchema=True, header=True)
dataset.printSchema()
```

```
root
 |-- crim: double (nullable = true)
 |-- zn: double (nullable = true)
 |-- indus: double (nullable = true)
 |-- chas: integer (nullable = true)
 |-- nox: double (nullable = true)
 |-- rm: double (nullable = true)
 |-- age: double (nullable = true)
 |-- dis: double (nullable = true)
 |-- rad: integer (nullable = true)
 |-- tax: integer (nullable = true)
 |-- ptratio: double (nullable = true)
 |-- b: double (nullable = true)
 |-- lstat: double (nullable = true)
 |-- medv: double (nullable = true)
```

```
#Input all the features in one vector column
assembler = VectorAssembler(inputCols=['crim','zn','indus','chas','nox','rm','age','dis','rad
output = assembler.transform(dataset)
#input vs output
finalized_data = output.select("Attributes", "medv")
finalized_data.show()
```

```
+--------------------+----+
|          Attributes|medv|
+--------------------+----+
|[0.00632,18.0,2.3...|24.0|
|[0.02731,0.0,7.07...|21.6|
|[0.02729,0.0,7.07...|34.7|
|[0.03237,0.0,2.18...|33.4|
|[0.06905,0.0,2.18...|36.2|
|[0.02985,0.0,2.18...|28.7|
|[0.08829,12.5,7.8...|22.9|
|[0.14455,12.5,7.8...|27.1|
|[0.21124,12.5,7.8...|16.5|
|[0.17004,12.5,7.8...|18.9|
|[0.22489,12.5,7.8...|15.0|
|[0.11747,12.5,7.8...|18.9|
|[0.09378,12.5,7.8...|21.7|
|[0.62976,0.0,8.14...|20.4|
|[0.63796,0.0,8.14...|18.2|
|[0.62739,0.0,8.14...|19.9|
|[1.05393,0.0,8.14...|23.1|
|[0.7842,0.0,8.14,...|17.5|
|[0.80271,0.0,8.14...|20.2|
|[0.7258,0.0,8.14,...|18.2|
+--------------------+----+
only showing top 20 rows
```

```
#slipt training and test data
train_data, test_data = finalized_data.randomSplit([0.80, 0.2])
regression = LinearRegression(featuresCol= 'Attributes', labelCol='medv')


#learn to fit the model from trainig set
regression = regression.fit(train_data)
#to predict the prices on testing set
pred = regression.evaluate(test_data)


#predict the model
pred.predictions.show()
```

```
+--------------------+----+------------------+
|          Attributes|medv|        prediction|
+--------------------+----+------------------+
|[0.01965,80.0,1.7...|20.1|20.942059375047997|
|[0.02729,0.0,7.07...|34.7| 30.45913909535283|
|[0.02899,40.0,1.2...|26.6| 22.84051236798047|
|[0.03427,0.0,5.19...|19.5| 20.17825409117635|
|[0.0351,95.0,2.68...|48.5|  42.1188680790035|
|[0.03578,20.0,3.3...|45.4|38.481007992886944|
|[0.03584,80.0,3.3...|23.5|30.273731654141216|
|[0.04203,28.0,15....|22.9|28.751109352962597|
|[0.04297,52.5,5.3...|24.8| 27.40148345030581|
|[0.04337,21.0,5.6...|20.5|23.772705281119805|
|[0.04379,80.0,3.3...|19.4| 25.46029542455749|
|[0.05188,0.0,4.49...|22.5|22.322549402843883|
|[0.0536,21.0,5.64...|25.0|27.994047888655807|
|[0.05515,33.0,2.1...|36.1| 33.16359805447096|
|[0.05644,40.0,6.4...|32.4| 36.14175926061993|
|[0.06466,70.0,2.2...|22.5| 29.14071748722086|
|[0.06642,0.0,4.05...|29.9| 30.78475934631303|
|[0.0686,0.0,2.89,...|33.2|32.072305644032724|
|[0.07022,0.0,4.05...|23.2|25.393829236612426|
|[0.08664,45.0,3.4...|36.4| 33.04606015731522|
+--------------------+----+------------------+
only showing top 20 rows
```

```
#coefficient of the regression model
coeff = regression.coefficients
#x and Y intercept
intr = regression.intercept
print("The coefficient of the model is : %a" %coeff)
print("The Intercept of the model is : %f" %intr)
```

```
The coefficient of the model is : DenseVector([-0.1075, 0.0441, 0.0388, 2.993, -15.1559,
The Intercept of the model is : 31.489172
```

```
from pyspark.ml.evaluation import RegressionEvaluator
eval = RegressionEvaluator(labelCol="medv", predictionCol="prediction", metricName="rmse")
```

```
#Root Mean Square Error
rmse = eval.evaluate(pred.predictions)
print("RMSE: %.3f" %rmse)
```

```
RMSE: 5.527
```

```
#mean Square Error
mse = eval.evaluate(pred.predictions, {eval.metricName:"mse"})
print("MSE: %.3f" %mse)
```

```
     MSE: 30.549
```

```
#mean Absolute Error
mae = eval.evaluate(pred.predictions, {eval.metricName:"mae"})
print("MAE: %.3f" %mae)
```

```
     MAE: 3.555
```

```
# r2 - coefficient of determination
r2 = eval.evaluate(pred.predictions, {eval.metricName:"r2"})
print("r2: %.3f" %r2)
```

```
     r2: 0.687
```