Skills Network

Hands On Lab - Saving and loading a SparkML model

Objectives:

In this lab you will

- Create a simple Linear Regression Model
- Save the SparkML model
- Load the SparkML model
- Make predictions using the loaded SparkML model

Install pyspark

In [1]:
```
!pip install pyspark
!pip install findspark
```

```
Collecting pyspark
  Downloading pyspark-3.4.1.tar.gz (310.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 310.8/310.8 MB 1.4 MB/s eta 0:00:0000:0100:01
  Preparing metadata (setup.py) ... done
Collecting py4j==0.10.9.7 (from pyspark)
  Downloading py4j-0.10.9.7-py2.py3-none-any.whl (200 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 200.5/200.5 kB 32.6 MB/s eta 0:00:00
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.4.1-py2.py3-none-any.whl size=311285398 sha256=68bab3fc596f8157f
c2c78579e5aac400b247ff534e7adb5485fec682f5c6a25
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/b7/8e/8f/ba5d017af5f502964eb1358e1d496a8519de1645936b0
1810e
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.7 pyspark-3.4.1
Collecting findspark
  Downloading findspark-2.0.1-py2.py3-none-any.whl (4.4 kB)
Installing collected packages: findspark
Successfully installed findspark-2.0.1
```

Import libraries

In [2]:
```
import findspark
findspark.init()
```

In [3]:
```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
```

Creating the spark session and context

In [4]:
```
# Creating a spark context class
sc = SparkContext()

# Creating a spark session
spark = SparkSession \
    .builder \
    .appName("Saving and Loading a SparkML Model").getOrCreate()
```

```
23/08/10 15:58:12 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

Importing Spark ML libraries

In [5]:
```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
```

Create a DataFrame with sample data

In [6]:
```
# Create a simple data set of infant height(cms) weight(kgs) chart.

mydata = [[46,2.5],[51,3.4],[54,4.4],[57,5.1],[60,5.6],[61,6.1],[63,6.4]]

# Mention column names of dataframe
columns = ["height", "weight"]
```

```
# creating a dataframe
mydf = spark.createDataFrame(mydata, columns)

# show data frame
mydf.show()
```

```
+------+------+
|height|weight|
+------+------+
|    46|   2.5|
|    51|   3.4|
|    54|   4.4|
|    57|   5.1|
|    60|   5.6|
|    61|   6.1|
|    63|   6.4|
+------+------+
```

Converting data frame columns into feature vectors

In this task we use the `VectorAssembler()` function to convert the dataframe columns into feature vectors. For our example, we use the horsepower ("hp) and weight of the car as input features and the miles-per-gallon ("mpg") as target labels.

In [7]:
```
assembler = VectorAssembler(
    inputCols=["height"],
    outputCol="features")

data = assembler.transform(mydf).select('features','weight')
```

In [8]:
```
data.show()
```

```
+--------+------+
|features|weight|
+--------+------+
|  [46.0]|   2.5|
|  [51.0]|   3.4|
|  [54.0]|   4.4|
|  [57.0]|   5.1|
|  [60.0]|   5.6|
|  [61.0]|   6.1|
|  [63.0]|   6.4|
+--------+------+
```

Create and Train model

We can create the model using the `LinearRegression()` class and train using the `fit()` function.

In [9]:
```
# Create a LR model
lr = LinearRegression(featuresCol='features', labelCol='weight', maxIter=100)
lr.setRegParam(0.1)
# Fit the model
lrModel = lr.fit(data)
```

```
23/08/10 16:11:54 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLA
S
23/08/10 16:11:54 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
23/08/10 16:11:54 WARN netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSystemL
APACK
23/08/10 16:11:54 WARN netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRefLAPA
CK
```

Save the model

In [10]:
```
lrModel.save('infantheight2.model')
```

Load the model

In [12]:
```
# You need LinearRegressionModel to load the model
from pyspark.ml.regression import LinearRegressionModel
```

In [13]:
```
model = LinearRegressionModel.load('infantheight2.model')
```

Make Prediction

Predict the weight of an infant whose height is 70 CMs.

In [14]:
```
# This function converts a scalar number into a dataframe that can be used by the model to predict.
def predict(weight):
    assembler = VectorAssembler(inputCols=["weight"],outputCol="features")
    data = [[weight,0]]
    columns = ["weight", "height"]
    _ = spark.createDataFrame(data, columns)
    __ = assembler.transform(_).select('features','height')
    predictions = model.transform(__)
    predictions.select('prediction').show()
```

In [15]:
```
predict(70)
```

```
+-----------------+
|       prediction|
+-----------------+
|7.863454719775907|
+-----------------+
```

Practice exercises

Save the model as `babyweightprediction.model`

In [16]:
```
lrModel.save('babyweightprediction.model')
```

Double-click **here** for the solution.

Load the model `babyweightprediction.model`

In [17]:
```
model = LinearRegressionModel.load('babyweightprediction.model')
```

Double-click **here** for the solution.

Predict the weight of an infant whose height is 50 CMs.

In [ ]:
```
predict(50)
```

Double-click **here** for the solution.

In [ ]:
```

```