(https://cognitiveclass.ai)

Welcome to exercise one of week four of “Apache Spark for Scalable Machine Learning on BigData”. In this exercise we’ll work on classification.

Let’s create our DataFrame again:

This notebook is designed to run in a IBM Watson Studio default runtime (NOT the Watson Studio Apache Spark Runtime as the default runtime with 1 vCPU is free of charge). Therefore, we install Apache Spark in local mode for test purposes only. Please don't use it in production.

In case you are facing issues, please read the following two documents first:

https://github.com/IBM/skillsnetwork/wiki/Environment-Setup
(https://github.com/IBM/skillsnetwork/wiki/Environment-Setup)

https://github.com/IBM/skillsnetwork/wiki/FAQ (https://github.com/IBM/skillsnetwork/wiki/FAQ)

Then, please feel free to ask:

https://coursera.org/learn/machine-learning-big-data-apache-spark/discussions/all
(https://coursera.org/learn/machine-learning-big-data-apache-spark/discussions/all?cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBMDeveloperSkillsNetwork-ML0201EN-SkillsNetwork-20647446&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newslette

Please make sure to follow the guidelines before asking a question:

https://github.com/IBM/skillsnetwork/wiki/FAQ#im-feeling-lost-and-confused-please-help-me
(https://github.com/IBM/skillsnetwork/wiki/FAQ#im-feeling-lost-and-confused-please-help-me)

If running outside Watson Studio, this should work as well. In case you are running in an Apache Spark context outside Watson Studio, please remove the Apache Spark setup in the first notebook cells.

In [1]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown('# <span style="color:red">'+string+'</span>'))


if ('sc' in locals() or 'sc' in globals()):
    printmd('<<<<<!!!!! It seems that you are running in a IBM Watson Studio Apache Spa
rk Notebook. Please run it in an IBM Watson Studio Default Runtime (without Apache Spar
k) !!!!!>>>>>')
```

In [2]:

```
!pip install pyspark==2.4.5
```

```
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/secretstorage/
dhcrypto.py:16: CryptographyDeprecationWarning: int_from_bytes is deprecat
ed, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/secretstorage/
util.py:25: CryptographyDeprecationWarning: int_from_bytes is deprecated,
use int.from_bytes instead
  from cryptography.utils import int_from_bytes
Collecting pyspark==2.4.5
  Downloading pyspark-2.4.5.tar.gz (217.8 MB)
     |████████████████████████████████| 217.8 MB 11 kB/s s eta 0:00:01
Collecting py4j==0.10.7
  Downloading py4j-0.10.7-py2.py3-none-any.whl (197 kB)
     |████████████████████████████████| 197 kB 67.4 MB/s eta 0:00:01
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-2.4.5-py2.py3-none-any.whl s
ize=218257927 sha256=b231381c0606b57a4fbef778445a724fa1d9fc5f73003efde9da5
7a90e3637c4
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/01/c0/03/1c241c9c482b
647d4d99412a98a5c7f87472728ad41ae55e1e
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.7 pyspark-2.4.5
```

In [3]:

```
try:
    from pyspark import SparkContext, SparkConf
    from pyspark.sql import SparkSession
except ImportError as e:
    printmd('<<<<<!!!!! Please restart your kernel after installing Apache Spark !!!!!>
>>>')
```

In [4]:

```
sc = SparkContext.getOrCreate(SparkConf().setMaster("local[*]"))

spark = SparkSession \
    .builder \
    .getOrCreate()
```

In [5]:

```
# delete files from previous runs
!rm -f hmp.parquet*

# download the file containing the data in PARQUET format
!wget https://github.com/IBM/coursera/raw/master/hmp.parquet

# create a dataframe out of it
df = spark.read.parquet('hmp.parquet')

# register a corresponding query table
df.createOrReplaceTempView('df')
```

```
--2021-04-12 23:52:11--  https://github.com/IBM/coursera/raw/master/hmp.pa
rquet
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/IBM/skillsnetwork/raw/master/hmp.parquet [fol
lowing]
--2021-04-12 23:52:12--  https://github.com/IBM/skillsnetwork/raw/master/h
mp.parquet
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/IBM/skillsnetwork/master/hmp.p
arquet [following]
--2021-04-12 23:52:12--  https://raw.githubusercontent.com/IBM/skillsnetwo
rk/master/hmp.parquet
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.19
9.108.133, 185.199.109.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 932997 (911K) [application/octet-stream]
Saving to: ‘hmp.parquet’

hmp.parquet         100%[===================>] 911.13K  --.-KB/s    in 0.0
2s

2021-04-12 23:52:12 (58.3 MB/s) - ‘hmp.parquet’ saved [932997/932997]
```

Since this is supervised learning, let’s split our data into train (80%) and test (20%) set.

In [6]:

```
splits = df.randomSplit([0.8, 0.2])
df_train = splits[0]
df_test = splits[1]
```

Again, we can re-use our feature engineering pipeline

In [7]:

```
from pyspark.ml.feature import StringIndexer, OneHotEncoder
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import Normalizer

indexer = StringIndexer(inputCol="class", outputCol="label")

vectorAssembler = VectorAssembler(inputCols=["x","y","z"],
                                  outputCol="features")

normalizer = Normalizer(inputCol="features", outputCol="features_norm", p=1.0)
```

Now we use LogisticRegression, a simple and basic linear classifier to obtain a classification performance baseline.

In [9]:

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline

lr = LogisticRegression(maxIter = 10, regParam = 0.3, elasticNetParam = 0.8)
pipeline = Pipeline(stages = [indexer, vectorAssembler, normalizer, lr])
model = pipeline.fit(df_train)
prediction = model.transform(df_test)
```

If we look at the schema of the prediction dataframe we see that there is an additional column called prediction which contains the best guess for the class our model predicts.

In [10]:

```
prediction.printSchema()
```

```
root
 |-- x: integer (nullable = true)
 |-- y: integer (nullable = true)
 |-- z: integer (nullable = true)
 |-- source: string (nullable = true)
 |-- class: string (nullable = true)
 |-- label: double (nullable = false)
 |-- features: vector (nullable = true)
 |-- features_norm: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

Let's evaluate performance by using a build-in functionality of Apache SparkML.

In [11]:

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
MulticlassClassificationEvaluator().setMetricName("accuracy").evaluate(prediction)
```

Out[11]:

```
0.20650522741488697
```

So we get 20% right. This is not bad for a baseline. Note that random guessing would give us only 7%. Of course we need to improve. You might have notices that we're dealing with a time series here. And we're not making use of that fact right now as we look at each training example only individually. But this is ok for now. More advanced courses like "Advanced Machine Learning and Signal Processing" (https://www.coursera.org/learn/advanced-machine-learning-signal-processing/ (https://www.coursera.org/learn/advanced-machine-learning-signal-processing?cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBMDeveloperSkillsNetwork-ML0201EN-SkillsNetwork-20647446&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newslette
_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBMDeveloperSkillsNetwork-ML0201EN-SkillsNetwork-20647446&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newslette will teach you how to improve accuracy to the nearly 100% by using algorithms like Fourier transformation or wavelet transformation. But let's skip this for now. In the following cell, please use the RandomForest classifier (you might need to play with the "numTrees" parameter) in the code cell below. You should get an accuracy of around 44%. More on RandomForest can be found here:

https://spark.apache.org/docs/latest/ml-classification-regression.html#random-forest-classifier (https://spark.apache.org/docs/latest/ml-classification-regression.html#random-forest-classifier?cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBMDeveloperSkillsNetwork-ML0201EN-SkillsNetwork-20647446&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newslette

In [12]:

```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

forest = RandomForestClassifier(numTrees = 10)
pipeline2 = Pipeline(stages = [indexer, vectorAssembler, normalizer, forest])
model2 = pipeline2.fit(df_train)
prediction2 = model2.transform(df_test)
```

In [13]:

```
prediction2.printSchema()
```

```
root
 |-- x: integer (nullable = true)
 |-- y: integer (nullable = true)
 |-- z: integer (nullable = true)
 |-- source: string (nullable = true)
 |-- class: string (nullable = true)
 |-- label: double (nullable = false)
 |-- features: vector (nullable = true)
 |-- features_norm: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

In [14]:

```
MulticlassClassificationEvaluator().setMetricName("accuracy").evaluate(prediction2)
```

Out[14]:

```
0.44124743097131625
```

Thank you for completing this lab!

This notebook was created by Romeo Kienzler (https://linkedin.com/in/romeo-kienzler-089b4557) I hope you found this lab interesting and educational. Feel free to contact me if you have any questions!

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-09-29	2.0	Srishti	Migrated Lab to Markdown and added to course repo in GitLab

In [ ]: