(https://cognitiveclass.ai)

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 9.5 kB/s  eta 0:00:01
Collecting py4j==0.10.7
  Downloading py4j-0.10.7-py2.py3-none-any.whl (197 kB)
     |████████████████████████████████| 197 kB 35.7 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=e838a0faae379755093e18d9e10f60632f665824df8979d5888c7
42977e02845
  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()
```

Welcome to exercise two of week three of “Apache Spark for Scalable Machine Learning on BigData”. In this exercise we’ll work on clustering.

Let’s create our DataFrame again:

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')
```

```
--2021-04-08 06:08:07--  https://github.com/IBM/coursera/raw/master/hmp.pa
rquet
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/IBM/skillsnetwork/raw/master/hmp.parquet [fol
lowing]
--2021-04-08 06:08:08--  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]
--2021-04-08 06:08:08--  https://raw.githubusercontent.com/IBM/skillsnetwo
rk/master/hmp.parquet
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.19
9.108.133, 185.199.111.133, 185.199.110.133, ...
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

2021-04-08 06:08:08 (40.6 MB/s) - ‘hmp.parquet’ saved [932997/932997]
```

Let’s reuse our feature engineering pipeline.

In [6]:

```
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler, Normalize
r
from pyspark.ml.linalg import Vectors
from pyspark.ml import Pipeline

indexer = StringIndexer(inputCol="class", outputCol="classIndex")
encoder = OneHotEncoder(inputCol="classIndex", outputCol="categoryVec")
vectorAssembler = VectorAssembler(inputCols=["x","y","z"],
                                  outputCol="features")
normalizer = Normalizer(inputCol="features", outputCol="features_norm", p=1.0)

pipeline = Pipeline(stages=[indexer, encoder, vectorAssembler, normalizer])
model = pipeline.fit(df)
prediction = model.transform(df)
prediction.show()
```

```
+---+---+---+--------------------+-----------+----------+--------------+--
--------------+--------------------+
|  x|  y|  z|              source|      class|classIndex|   categoryVec|
features|       features_norm|
+---+---+---+--------------------+-----------+----------+--------------+--
--------------+--------------------+
| 22| 49| 35|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
2.0,49.0,35.0]|[0.20754716981132...|
| 22| 49| 35|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
2.0,49.0,35.0]|[0.20754716981132...|
| 22| 52| 35|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
2.0,52.0,35.0]|[0.20183486238532...|
| 22| 52| 35|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
2.0,52.0,35.0]|[0.20183486238532...|
| 21| 52| 34|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
1.0,52.0,34.0]|[0.19626168224299...|
| 22| 51| 34|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
2.0,51.0,34.0]|[0.20560747663551...|
| 20| 50| 35|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
0.0,50.0,35.0]|[0.19047619047619...|
| 22| 52| 34|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
2.0,52.0,34.0]|[0.20370370370370...|
| 22| 50| 34|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
2.0,50.0,34.0]|[0.20754716981132...|
| 22| 51| 35|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
2.0,51.0,35.0]|[0.20370370370370...|
| 21| 51| 33|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
1.0,51.0,33.0]|[0.2,0.4857142857...|
| 20| 50| 34|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
0.0,50.0,34.0]|[0.19230769230769...|
| 21| 49| 33|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
1.0,49.0,33.0]|[0.20388349514563...|
| 21| 49| 33|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
1.0,49.0,33.0]|[0.20388349514563...|
| 20| 51| 35|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[2
0.0,51.0,35.0]|[0.18867924528301...|
| 18| 49| 34|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[1
8.0,49.0,34.0]|[0.17821782178217...|
| 19| 48| 34|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[1
9.0,48.0,34.0]|[0.18811881188118...|
| 16| 53| 34|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[1
6.0,53.0,34.0]|[0.15533980582524...|
| 18| 52| 35|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[1
8.0,52.0,35.0]|[0.17142857142857...|
| 18| 51| 32|Accelerometer-201...|Brush_teeth|       6.0|(13,[6],[1.0])|[1
8.0,51.0,32.0]|[0.17821782178217...|
+---+---+---+--------------------+-----------+----------+--------------+--
--------------+--------------------+
only showing top 20 rows
```

Now let’s create a new pipeline for kmeans.

In [7]:

```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator

kmeans = KMeans(featuresCol="features").setK(14).setSeed(1)
pipeline = Pipeline(stages=[vectorAssembler, kmeans])
model = pipeline.fit(df)
predictions = model.transform(df)

evaluator = ClusteringEvaluator()

silhouette = evaluator.evaluate(predictions)
print("Silhouette with squared euclidean distance = " + str(silhouette))
```

```
Silhouette with squared euclidean distance = 0.41244594513295846
```

We have 14 different movement patterns in the dataset, so setting K of KMeans to 14 is a good idea. But please experiment with different values for K, do you find a sweet spot? The closer Silhouette gets to 1, the better.

https://en.wikipedia.org/wiki/Silhouette_(clustering) (https://en.wikipedia.org/wiki/Silhouette_(clustering)?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 [10]:

```
# please change the pipeline the check performance for different K, feel free to use a
 loop
for ki in range(3, 15):
    kmeans = KMeans(featuresCol = 'features').setK(ki).setSeed(1)
    pipeline = Pipeline(stages = [vectorAssembler, kmeans])
    model = pipeline.fit(df)
    predictions = model.transform(df)
    evaluator = ClusteringEvaluator()
    silhouette = evaluator.evaluate(predictions)
    print('For k = ', ki, ' ===> Silhouette value = ', silhouette)
```

```
For k =  3  ===> Silhouette value =  0.6147915951361759
For k =  4  ===> Silhouette value =  0.6333227654128869
For k =  5  ===> Silhouette value =  0.5937447997439024
For k =  6  ===> Silhouette value =  0.592463658820136
For k =  7  ===> Silhouette value =  0.5484627422401509
For k =  8  ===> Silhouette value =  0.46686489256383346
For k =  9  ===> Silhouette value =  0.48034893889849645
For k =  10  ===> Silhouette value =  0.47370428136987536

Exception ignored in: <function JavaWrapper.__del__ at 0x7f5a227157a0>
Traceback (most recent call last):
  File "/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/pyspar
k/ml/wrapper.py", line 40, in __del__
    if SparkContext._active_spark_context and self._java_obj is not None:
AttributeError: 'KMeans' object has no attribute '_java_obj'

For k =  11  ===> Silhouette value =  0.4819049717562352
For k =  12  ===> Silhouette value =  0.40964155503229643
For k =  13  ===> Silhouette value =  0.4153293521373778
For k =  14  ===> Silhouette value =  0.41244594513295846
```

Final Result: The best k = 4 ==> with silhouette value = 0.633327 [more closer than 1]

Normalized features

Now please extend the pipeline to work on the normalized features. You need to tell KMeans to use the normalized feature column and change the pipeline in order to contain the normalizer stage as well.

In [27]:

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

In [30]:

```
kmeans2 = KMeans(featuresCol = "features_norm").setK(2).setSeed(1)
pipeline2 = Pipeline(stages = [vectorAssembler, normalizer, kmeans2])
model2 = pipeline2.fit(df)

predictions2 = model2.transform(df)

evaluator2 = ClusteringEvaluator()

silhouette2 = evaluator2.evaluate(predictions2)
print("Silhouette with squared euclidean distance = " + str(silhouette2))
```

```
Silhouette with squared euclidean distance = 0.6462988404434188
```

Final Result: The best k = 2 ==> with silhouette value = 0.646298

Denormalized features

Sometimes, inflating the dataset helps, here we multiply x by 10, let's see if the performance inceases.

In [22]:

```
from pyspark.sql.functions import col
df_denormalized = df.select([col('*'),(col('x')*10)]).drop('x').withColumnRenamed('(x *
10)','x')
```

In [23]:

```
kmeans = KMeans(featuresCol="features").setK(12).setSeed(1)
pipeline = Pipeline(stages=[vectorAssembler, kmeans])
model = pipeline.fit(df_denormalized)
predictions = model.transform(df_denormalized)

evaluator = ClusteringEvaluator()

silhouette = evaluator.evaluate(predictions)
print("Silhouette with squared euclidean distance = " + str(silhouette))
```

```
Silhouette with squared euclidean distance = 0.6032474280054309
```

Final Result: a good result is k = 12 ==> with silhouette value = 0.603247

Gaussian Mixture Algorithm

Apache SparkML can be used to try many different algorithms and parametrizations using the same pipeline. Please change the code below to use GaussianMixture over KMeans. Please use the following link for your reference.

https://spark.apache.org/docs/latest/ml-clustering.html#gaussian-mixture-model-gmm (https://spark.apache.org/docs/latest/ml-clustering.html#gaussian-mixture-model-gmm?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 [31]:

```
from pyspark.ml.clustering import GaussianMixture

gmm = GaussianMixture().setK(3).setSeed(1)
pipeline = Pipeline(stages = [vectorAssembler, gmm])

model = pipeline.fit(df)

predictions = model.transform(df)

evaluator = ClusteringEvaluator()

silhouette = evaluator.evaluate(predictions)
print("Silhouette with squared euclidean distance = " + str(silhouette))
```

```
Silhouette with squared euclidean distance = 0.41320597955203525
```

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