week_3

VERSION AUTHOR	LAST UPDATED	LANGUAGE
Unknown	28 Aug 2018, 5:50 PM	Python 3.5 with Spark 2.1

This is the third assignment for the Coursera course "Advanced Machine Learning and Signal Processing"

Again, please insert to code to your ApacheCouchDB based Cloudant instance below using the "Insert Code" function of Watson Studio (you've done this in Assignment 1 and 2 before)

Done, just execute all cells one after the other and you are done - just note that in the last one you must update your email address (the one you've used for coursera) and obtain a submission token, you get this from the programming assignment directly on coursera.

Please fill in the sections labelled with "###YOUR_CODE_GOES_HERE###"

In [54]:
```
credentials_1 = {
  "username": "12b91d1c-8f7c-4caa-bb21-4e6430b7126a-bluemix",
  "password": "b4cb0c9142010ca255ca2f1d5ff4707c176bf26a13e92884a2f6764f
a875a63e",
  "host": "12b91d1c-8f7c-4caa-bb21-4e6430b7126a-bluemix.cloudant.com",
  "port": 443,
  "custom_url": "https://12b91d1c-8f7c-4caa-bb21-4e6430b7126a-bluemix:b
4cb0c9142010ca255ca2f1d5ff4707c176bf26a13e92884a2f6764fa875a63e@12b91d1
c-8f7c-4caa-bb21-4e6430b7126a-bluemix.cloudant.com"
}
```

Let's create a SparkSession object and put the Cloudant credentials into it

In [55]:
```
spark = SparkSession\
    .builder\
    .appName("Cloudant Spark SQL Example in Python using temp tables")\
    .config("cloudant.host",credentials_1['custom_url'].split('@')[1])\
    .config("cloudant.username", credentials_1['username'])\
    .config("cloudant.password",credentials_1['password'])\
    .getOrCreate()
```

Now it’s time to have a look at the recorded sensor data. You should see data similar to the one exemplified below….

In [56]:
```
df=spark.read.load('shake_classification2', "org.apache.bahir.cloudant"
)
df.createOrReplaceTempView("df")
spark.sql("SELECT * from df").show()
```

```
+-----+--------+-----+-----+-----+--------------------+---------------
----+
|CLASS|SENSORID|    X|    Y|    Z|                 _id|
_rev|
+-----+--------+-----+-----+-----+--------------------+---------------
----+
|    0|asdfghjk| 0.09| 0.09| 0.09|08a0090048132feb2...|1-bf7aa9edc5a599
2...|
|    0|asdfghjk| 0.04| 0.04| 0.04|08a0090048132feb2...|1-8b6948def4b339
d...|
|    0|asdfghjk|-0.03|-0.03|-0.03|08a0090048132feb2...|1-9e51a2b988a979
d...|
|    0|asdfghjk|-0.02|-0.02|-0.02|08a0090048132feb2...|1-2bf1b84ff00070
```

```
3...|
|    0|asdfghjk| 0.03| 0.03| 0.03|08a0090048132feb2...|1-a05a2cbd1889ec
2...|
|    0|asdfghjk| 0.03| 0.03| 0.03|08a0090048132feb2...|1-a05a2cbd1889ec
2...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|
|    0|asdfghjk|  0.0|  0.0|  0.0|08a0090048132feb2...|1-1ed1d73a44dea3
6...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|
|    0|asdfghjk|-0.02|-0.02|-0.02|08a0090048132feb2...|1-2bf1b84ff00070
3...|
|    0|asdfghjk|-0.02|-0.02|-0.02|08a0090048132feb2...|1-2bf1b84ff00070
3...|
|    0|asdfghjk|-0.03|-0.03|-0.03|08a0090048132feb2...|1-9e51a2b988a979
d...|
|    0|asdfghjk| 0.02| 0.02| 0.02|08a0090048132feb2...|1-25badf12404083
a...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|
|    0|asdfghjk|-0.06|-0.06|-0.06|08a0090048132feb2...|1-4b64be7c1c427e
4...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|
|    0|asdfghjk|-0.09|-0.09|-0.09|08a0090048132feb2...|1-6785b12cd3c82b
c...|
|    0|asdfghjk| 0.02| 0.02| 0.02|08a0090048132feb2...|1-25badf12404083
a...|
+-----+--------+-----+-----+-----+--------------------+---------------
----+
only showing top 20 rows
```

Let’s check if we have balanced classes – this means that we have roughly the same number of examples for each class we want to predict. This is important for classification but also helpful for clustering

In [57]:
```
spark.sql("SELECT count(class), class from df group by class").show()
```

```
+------------+-----+
|count(class)|class|
+------------+-----+
|         510|    0|
|         616|    1|
+------------+-----+
```

Let's create a VectorAssembler which consumes columns X, Y and Z and produces a column “features”

In [58]:
```
from pyspark.ml.feature import VectorAssembler
vectorAssembler = VectorAssembler(inputCols=["X","Y","Z"],
                                  outputCol="features")
```

Please insatiate a clustering algorithm from the SparkML package and assign it to the clust variable. Here we don't need to take care of the "CLASS" column since we are in unsupervised learning mode – so let's pretend to not even have the "CLASS" column for now – but it will become very handy later in assessing the clustering performance. PLEASE NOTE – IN REAL-WORLD SCENARIOS THERE IS NO CLASS COLUMN – THEREFORE YOU CAN'T ASSESS CLASSIFICATION PERFORMANCE USING THIS COLUMN

In [59]:
```
from pyspark.ml.clustering import GaussianMixture
clust = GaussianMixture().setK(2).setSeed(1)
```

Let's train...

In [60]:
```
from pyspark.ml import Pipeline
pipeline = Pipeline(stages=[vectorAssembler, clust])
model = pipeline.fit(df)
```

...and evaluate...

In [61]:
```
prediction = model.transform(df)
prediction.show()
```

```
+-----+--------+-----+-----+-----+--------------------+---------------
----+------------------+----------+--------------------+
|CLASS|SENSORID|    X|    Y|    Z|                 _id|
_rev|           features|prediction|         probability|
+-----+--------+-----+-----+-----+--------------------+---------------
----+------------------+----------+--------------------+
|    0|asdfghjk| 0.09| 0.09| 0.09|08a0090048132feb2...|1-bf7aa9edc5a599
2...|   [0.09,0.09,0.09]|         1|[0.15332615004498...|
|    0|asdfghjk| 0.04| 0.04| 0.04|08a0090048132feb2...|1-8b6948def4b339
d...|   [0.04,0.04,0.04]|         1|[0.04007966847636...|
|    0|asdfghjk|-0.03|-0.03|-0.03|08a0090048132feb2...|1-9e51a2b988a979
d...|[-0.03,-0.03,-0.03]|         1|[0.03424729720081...|
|    0|asdfghjk|-0.02|-0.02|-0.02|08a0090048132feb2...|1-2bf1b84ff00070
3...|[-0.02,-0.02,-0.02]|         1|[0.03073826153248...|
|    0|asdfghjk| 0.03| 0.03| 0.03|08a0090048132feb2...|1-a05a2cbd1889ec
2...|   [0.03,0.03,0.03]|         1|[0.03441073261793...|
|    0|asdfghjk| 0.03| 0.03| 0.03|08a0090048132feb2...|1-a05a2cbd1889ec
2...|   [0.03,0.03,0.03]|         1|[0.03441073261793...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|[-0.04,-0.04,-0.04]|         1|[0.03982752837615...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|[-0.04,-0.04,-0.04]|         1|[0.03982752837615...|
|    0|asdfghjk|  0.0|  0.0|  0.0|08a0090048132feb2...|1-1ed1d73a44dea3
6...|          (3,[],[])|         1|[0.02821114159692...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|[-0.04,-0.04,-0.04]|         1|[0.03982752837615...|
|    0|asdfghjk|-0.02|-0.02|-0.02|08a0090048132feb2...|1-2bf1b84ff00070
3...|[-0.02,-0.02,-0.02]|         1|[0.03073826153248...|
|    0|asdfghjk|-0.02|-0.02|-0.02|08a0090048132feb2...|1-2bf1b84ff00070
3...|[-0.02,-0.02,-0.02]|         1|[0.03073826153248...|
|    0|asdfghjk|-0.03|-0.03|-0.03|08a0090048132feb2...|1-9e51a2b988a979
d...|[-0.03,-0.03,-0.03]|         1|[0.03424729720081...|
|    0|asdfghjk| 0.02| 0.02| 0.02|08a0090048132feb2...|1-25badf12404083
a...|   [0.02,0.02,0.02]|         1|[0.03083633589894...|
```

```
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|[-0.04,-0.04,-0.04]|         1|[0.03982752837615...|
|    0|asdfghjk|-0.06|-0.06|-0.06|08a0090048132feb2...|1-4b64be7c1c427e
4...|[-0.06,-0.06,-0.06]|         1|[0.06099632327040...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|[-0.04,-0.04,-0.04]|         1|[0.03982752837615...|
|    0|asdfghjk|-0.04|-0.04|-0.04|08a0090048132feb2...|1-10b2ea101b1aa1
d...|[-0.04,-0.04,-0.04]|         1|[0.03982752837615...|
|    0|asdfghjk|-0.09|-0.09|-0.09|08a0090048132feb2...|1-6785b12cd3c82b
c...|[-0.09,-0.09,-0.09]|         1|[0.15141594019802...|
|    0|asdfghjk| 0.02| 0.02| 0.02|08a0090048132feb2...|1-25badf12404083
a...|   [0.02,0.02,0.02]|         1|[0.03083633589894...|
+-----+--------+-----+-----+-----+--------------------+---------------
----+------------------+----------+--------------------+
only showing top 20 rows
```

In [62]:
```
prediction.createOrReplaceTempView('prediction')
spark.sql('''
select max(correct)/max(total) as accuracy from (

    select sum(correct) as correct, count(correct) as total from (
        select case when class != prediction then 1 else 0 end as corre
ct from prediction
    )

    union

    select sum(correct) as correct, count(correct) as total from (
        select case when class = prediction then 1 else 0 end as correc
t from prediction
    )
)
''').rdd.map(lambda row: row.accuracy).collect()[0]
```

Out[62]:
```
0.9307282415630551
```

If you reached at least 55% of accuracy you are fine to submit your predictions to the grader. Otherwise please experiment with parameters setting to your clustering algorithm, use a different algorithm or just re-record your data and try to obtain. In case you are stuck, please use the Coursera Discussion Forum. Please note again – in a real-world scenario there is no way in doing this – since there is no class label in your data. Please have a look at this further reading on clustering performance evaluation https://en.wikipedia.org/wiki/Cluster_analysis#Evaluation_and_assessment (https://en.wikipedia.org/wiki/Cluster_analysis#Evaluation_and_assessment)

In [63]:
```
!rm -Rf a2_m3.json
```

In [64]:
```
!rm -f rklib.py
!wget https://raw.githubusercontent.com/romeokienzler/developerWorks/ma
ster/coursera/ai/rklib.py
```

```
--2018-08-28 10:50:13--  https://raw.githubusercontent.com/romeokienzle
r/developerWorks/master/coursera/ai/rklib.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.
101.48.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|15
```