# Spark Train Random Forest Classifier

Train Random Forest classifier with Apache SparkML

In [1]:
```
%%bash
export version=`python --version |awk '{print $2}' |awk -F"." '{print $1$2}'`

echo $version

if [ $version == '36' ] || [ $version == '37' ]; then
    echo 'Starting installation...'
    pip3 install pyspark==2.4.8 wget==3.2 pyspark2pmml==0.5.1 > install.log 2> inst
    if [ $? == 0 ]; then
        echo 'Please <<RESTART YOUR KERNEL>> (Kernel->Restart Kernel and Clear All
    else
        echo 'Installation failed, please check log:'
        cat install.log
    fi
elif [ $version == '38' ] || [ $version == '39' ]; then
    pip3 install pyspark==3.1.2 wget==3.2 pyspark2pmml==0.5.1 > install.log 2> inst
    if [ $? == 0 ]; then
        echo 'Please <<RESTART YOUR KERNEL>> (Kernel->Restart Kernel and Clear All
    else
        echo 'Installation failed, please check log:'
        cat install.log
    fi
else
    echo 'Currently only python 3.6, 3.7 , 3.8 and 3.9 are supported, in case you r
    exit -1
fi
```

```
37
Starting installation...
Please <<RESTART YOUR KERNEL>> (Kernel->Restart Kernel and Clear All Outputs)
```

In [2]:
```
from pyspark import SparkContext, SparkConf, SQLContext
import os
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark2pmml import PMMLBuilder
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import MinMaxScaler
from pyspark.ml.feature import OneHotEncoder
import logging
import shutil
import site
import sys
import wget
import re
```

In [3]:
```
if sys.version[0:3] == '3.9':
    url = ('https://github.com/jpmml/jpmml-sparkml/releases/download/1.7.2/'
           'jpmml-sparkml-executable-1.7.2.jar')
    wget.download(url)
    shutil.copy('jpmml-sparkml-executable-1.7.2.jar',
```

```
                site.getsitepackages()[0] + '/pyspark/jars/')
elif sys.version[0:3] == '3.8':
    url = ('https://github.com/jpmml/jpmml-sparkml/releases/download/1.7.2/'
           'jpmml-sparkml-executable-1.7.2.jar')
    wget.download(url)
    shutil.copy('jpmml-sparkml-executable-1.7.2.jar',
                site.getsitepackages()[0] + '/pyspark/jars/')
elif sys.version[0:3] == '3.7':
    url = ('https://github.com/jpmml/jpmml-sparkml/releases/download/1.5.12/'
           'jpmml-sparkml-executable-1.5.12.jar')
    wget.download(url)
elif sys.version[0:3] == '3.6':
    url = ('https://github.com/jpmml/jpmml-sparkml/releases/download/1.5.12/'
           'jpmml-sparkml-executable-1.5.12.jar')
    wget.download(url)
else:
    raise Exception('Currently only python 3.6 , 3.7, 3,8 and 3.9 is supported, in
                    'you need a different version please open an issue at '
                    'https://github.com/IBM/claimed/issues')
```

In [4]:
```
data_new_csv = os.environ.get('data_new_csv',
                              'trends.csv')  # input file name (parquet)
master = os.environ.get('master',
                        "local[*]")  # URL to Spark master
model_target = os.environ.get('model_target',
                              "model.xml")  # model output file name
data_dir = os.environ.get('data_dir',
                          '../../data/')  # temporary directory for data
input_columns = os.environ.get('input_columns',
                               '["x", "y", "z"]')  # input columns to consider
```

In [5]:
```
parameters = list(
    map(lambda s: re.sub('$', '"', s),
        map(
            lambda s: s.replace('=', '="'),
            filter(
                lambda s: s.find('=') > -1 and bool(re.match(r'[A-Za-z0-9_]*=[.\/A-
                sys.argv
            )
    )))

for parameter in parameters:
    logging.warning('Parameter: ' + parameter)
    exec(parameter)
```

In [6]:
```
conf = SparkConf().setMaster(master)
#if sys.version[0:3] == '3.6' or sys.version[0:3] == '3.7':
conf.set("spark.jars", 'jpmml-sparkml-executable-1.5.12.jar')

sc = SparkContext.getOrCreate(conf)
sqlContext = SQLContext(sc)
spark = sqlContext.sparkSession
```

```
22/11/11 12:58:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel
(newLevel).
22/11/11 12:58:37 WARN util.Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
22/11/11 12:58:37 WARN util.Utils: Service 'SparkUI' could not bind on port 4041.
Attempting port 4042.
```

In [7]:
```
#df = spark.read.parquet(data_dir + data_parquet)
df = spark.read.option("header", True).csv(data_dir + data_new_csv)
```

In [8]:
```
df.head()
```

Out[8]:
```
Row(x='33', y='36', z='51', source='Accelerometer-2011-03-24-13-21-39-eat_meat-f1.
txt', class='Eat_meat')
```

In [9]:
```
# register a corresponding query table
df.createOrReplaceTempView('df')
```

In [10]:
```
from pyspark.sql.types import DoubleType
df = df.withColumn("x", df["x"].cast(DoubleType()))
df = df.withColumn("y", df.y.cast(DoubleType()))
df = df.withColumn("z", df.z.cast(DoubleType()))
```

In [11]:
```
indexer = StringIndexer(inputCol="class", outputCol="label")
indexed = indexer.fit(df).transform(df)
indexed.show()
```

```
+----+----+----+--------------------+--------+-----+
|   x|   y|   z|              source|   class|label|
+----+----+----+--------------------+--------+-----+
|33.0|36.0|51.0|Accelerometer-201...|Eat_meat|  5.0|
|33.0|36.0|51.0|Accelerometer-201...|Eat_meat|  5.0|
|33.0|35.0|53.0|Accelerometer-201...|Eat_meat|  5.0|
|31.0|37.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
|32.0|36.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
|32.0|36.0|51.0|Accelerometer-201...|Eat_meat|  5.0|
|32.0|36.0|51.0|Accelerometer-201...|Eat_meat|  5.0|
|33.0|36.0|53.0|Accelerometer-201...|Eat_meat|  5.0|
|33.0|35.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
|33.0|36.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
|32.0|35.0|53.0|Accelerometer-201...|Eat_meat|  5.0|
|33.0|36.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
|32.0|38.0|53.0|Accelerometer-201...|Eat_meat|  5.0|
|32.0|37.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
|33.0|35.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
|32.0|36.0|53.0|Accelerometer-201...|Eat_meat|  5.0|
|32.0|36.0|53.0|Accelerometer-201...|Eat_meat|  5.0|
|32.0|36.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
|34.0|36.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
|33.0|36.0|52.0|Accelerometer-201...|Eat_meat|  5.0|
+----+----+----+--------------------+--------+-----+
only showing top 20 rows
```

In [12]:
```
splits = indexed.randomSplit([0.8, 0.2])
```

```
df_train = splits[0]
df_test = splits[1]
```

In [13]:
```
#indexer = StringIndexer(inputCol="class", outputCol="label")
# all string(categorical) variables will be encoded into numbers, each category by
sindexer = StringIndexer(inputCol="label",
                         outputCol="label1",
                         handleInvalid='keep',
                         stringOrderType='frequencyDesc')

vectorAssembler = VectorAssembler(inputCols=eval(input_columns),
                                  outputCol="features")

normalizer = MinMaxScaler(inputCol="features", outputCol="features_norm")
```

In [14]:
```
rfc= RandomForestClassifier(numTrees=10,
                            maxDepth=5,
                            labelCol='label',
                            seed=1)
```

In [15]:
```
pipeline = Pipeline(stages=[vectorAssembler, normalizer, rfc])
```

In [16]:
```
model = pipeline.fit(df_train)
```

In [18]:
```
prediction = model.transform(df_train)
```

In [19]:
```
binEval = MulticlassClassificationEvaluator(). \
    setMetricName("accuracy"). \
    setPredictionCol("prediction"). \
    setLabelCol("label")

binEval.evaluate(prediction)
```

Out[19]:
```
0.44570934081629227
```

In [20]:
```
#lr = LogisticRegression(maxIter=10, regParam=0.01, elasticNetParam=0.0)
rfc= RandomForestClassifier(numTrees=20,
                            maxDepth=7,
                            labelCol='label',
                            seed=1)
```

In [21]:
```
pipeline = Pipeline(stages=[vectorAssembler, normalizer, rfc])
model = pipeline.fit(df_train)
```

In [22]:
```
prediction = model.transform(df_train)
```

In [23]:
```
binEval = MulticlassClassificationEvaluator(). \
    setMetricName("accuracy"). \
    setPredictionCol("prediction"). \
    setLabelCol("label")

binEval.evaluate(prediction)
```

Out[23]:
```
0.46767372954928615
```

In [24]:
```
pmmlBuilder = PMMLBuilder(sc, df_train, model)
pmmlBuilder.buildFile(data_dir + model_target)
```

Out[24]:
```
'/resources/labs/BD0231EN/claimed/component-library/transform/../../data/model.xml'
```

In [25]:
```
prediction = model.transform(df_test)
```

In [26]:
```
binEval = MulticlassClassificationEvaluator(). \
    setMetricName("accuracy"). \
    setPredictionCol("prediction"). \
    setLabelCol("label")

binEval.evaluate(prediction)
```

Out[26]:
```
0.4689383014164179
```

In [ ]:
```

```