Converts a parquet file to CSV file with header using ApacheSpark

In [6]:

```
%%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> install.log
    if [ $? == 0 ]; then
        echo 'Please <<RESTART YOUR KERNEL>> (Kernel->Restart Kernel and Clear All Outputs)
    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> install.log
    if [ $? == 0 ]; then
        echo 'Please <<RESTART YOUR KERNEL>> (Kernel->Restart Kernel and Clear All Outputs)
    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 need a di
    exit -1
fi
```

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

In [7]:

```
# @param data_dir temporal data storage for local execution
# @param data_csv csv path and file name (default: data.csv)
# @param data_parquet path and parquet file name (default: data.parquet)
# @param master url of master (default: local mode)
```

In [8]:

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
import os
import shutil
import glob
```

In [34]:

```
# Proceeding ahead using parquet file with logistic regression
```

In [15]:

```
data_csv = os.environ.get('data_csv', 'data.csv')
data_parquet = os.environ.get('data_parquet', 'data.parquet')
master = os.environ.get('master', "local[*]")
data_dir = os.environ.get('data_dir', '../../data/')
```

In [16]:

```
data_parquet = 'data.parquet'
data_csv = 'trends.csv'
```

In [17]:

```
skip = False
if os.path.exists(data_dir + data_csv):
    skip = True
```

In [18]:

```
if not skip:
    sc = SparkContext.getOrCreate(SparkConf().setMaster(master))
    spark = SparkSession.builder.getOrCreate()
```

In [19]:

```
if not skip:
    df = spark.read.parquet(data_dir + data_parquet)
```

In [20]:

```
if not skip:
    if os.path.exists(data_dir + data_csv):
        shutil.rmtree(data_dir + data_csv)
    df.coalesce(1).write.option("header", "true").csv(data_dir + data_csv)
    file = glob.glob(data_dir + data_csv + '/part-*')
    shutil.move(file[0], data_dir + data_csv + '.tmp')
    shutil.rmtree(data_dir + data_csv)
    shutil.move(data_dir + data_csv + '.tmp', data_dir + data_csv)
```

Spark Train Logistic Regression

Train Logistic Regression classifier with Apache SparkML

In [21]:

```
%%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> install.log
    if [ $? == 0 ]; then
        echo 'Please <<RESTART YOUR KERNEL>> (Kernel->Restart Kernel and Clear All Outputs)
    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> install.log
    if [ $? == 0 ]; then
        echo 'Please <<RESTART YOUR KERNEL>> (Kernel->Restart Kernel and Clear All Outputs)
    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 need a di
    exit -1
fi
```

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

In [22]:

```
from pyspark import SparkContext, SparkConf, SQLContext
import os
from pyspark.ml.classification import LogisticRegression
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
import logging
import shutil
import site
import sys
import wget
import re
```

In [23]:

```
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 case '
                    'you need a different version please open an issue at '
                    'https://github.com/IBM/claimed/issues')
```

In [24]:

```
data_parquet = os.environ.get('data_parquet',
                              'data.parquet')  # 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 [25]:

```
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-Za-z0-9]
                sys.argv
            )
    )))

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

In [26]:

```
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
```

In [27]:

```
df = spark.read.parquet(data_dir + data_parquet)
```

In [28]:

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

In [29]:

```
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 [30]:

```
splits = df.randomSplit([0.8, 0.2], seed = 1)
df_train = splits[0]
df_test = splits[1]
indexer = StringIndexer(inputCol="class", outputCol="label")

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

normalizer = MinMaxScaler(inputCol="features", outputCol="features_norm")
lr = LogisticRegression(maxIter=1000, regParam=2.0, elasticNetParam=1.0)
pipeline = Pipeline(stages=[indexer, vectorAssembler, normalizer, lr])
model = pipeline.fit(df_train)
prediction = model.transform(df_train)
binEval = MulticlassClassificationEvaluator(). \
    setMetricName("accuracy"). \
    setPredictionCol("prediction"). \
    setLabelCol("label")

binEval.evaluate(prediction)
```

```
22/11/11 12:36:16 WARN netlib.BLAS: Failed to load implementation from: co
m.github.fommil.netlib.NativeSystemBLAS
22/11/11 12:36:16 WARN netlib.BLAS: Failed to load implementation from: co
m.github.fommil.netlib.NativeRefBLAS
```

Out[30]:

```
0.20652429598763358
```

In [32]:

```
# pmmlBuilder = PMMLBuilder(sc, df_train, model)
# pmmlBuilder.buildFile(data_dir + model_target)
```

In [ ]:

```
```