Final Project of Machine Learning with Spark

Author - Rohit Rokde

In [1]:

```
import findspark
findspark.init('/u/rrokde/spark-2.4.4-bin-hadoop2.7')
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

conf = SparkConf().setMaster("local").setAppName("Project Assignment - ML with Spark")
sc = SparkContext(conf=conf)
sc.setLogLevel("ERROR")
spark = SparkSession.builder.getOrCreate()
```

In [2]:

```
# 1. Data Loading

data = spark.read.load('/u/rrokde/mls/default of credit card clients1.csv',
                       format='csv', header=False, inferSchema=True, sep=',')
```

In [3]:

```
# 2. Data Transformation

#We need to remove the garbage values from the first row.
from itertools import islice
data = data.rdd.mapPartitionsWithIndex(lambda idx, it: islice(it, 1, None) if (idx == 0 ) else it).toDF()
firstRow = data.head(1)

#Second row contains the column names.
columnNames = []
for i in range(len(firstRow[0])):
    columnNames.append( str(firstRow[0][i]).replace(' ', '_') )
data = data.rdd.mapPartitionsWithIndex(lambda idx, it: islice(it, 1, None) if (idx == 0 ) else it).toDF()

#Next we tidy up the dataframe by setting the right column names and casting column data to the right type
for i in range(len(firstRow[0])):
    data = data.withColumnRenamed("_c" + str(i), columnNames[i])
    data = data.withColumn(columnNames[i],data[columnNames[i]].cast('double'))
    # Source: https://forums.databricks.com/questions/9147/how-to-infer-csv-schema-default-all-columns-like-s.html

#Now we optimize our data for machine learning. That is extracting features and then normalizing it.
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.feature import MinMaxScaler

#Code taken from 1st assignment
df = data
list_of_columns = df.columns
list_of_columns.pop()     #Remove the target column from feature vector
assembler = VectorAssembler().setInputCols(list_of_columns).setOutputCol("features")
transformed = assembler.transform(df)
scaler = MinMaxScaler(inputCol="features",outputCol="scaledFeatures")
scalerModel =  scaler.fit(transformed.select("features"))
scaledData = scalerModel.transform(transformed)

#Remove unnecessary columns. We have all the data in vectorised column - scaledFeatures
for col in scaledData.columns:
    if( (col == 'default_payment_next_month') or (col == 'scaledFeatures')):
        pass
    else:
        scaledData = scaledData.drop(col)

print("Dataframe structure used for training and evaluation:-")
scaledData.printSchema()
```

```
Dataframe structure used for training and evaluation:-
root
 |-- default_payment_next_month: double (nullable = true)
 |-- scaledFeatures: vector (nullable = true)
```

In [4]:

```
# 3. Model Learning


# Split the data into training and test sets
(trainingData, testData) = scaledData.randomSplit([0.7, 0.3])

#Source: https://spark.apache.org/docs/latest/ml-classification-regression.htm
l#binomial-logistic-regression
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml import Pipeline

# Train a DecisionTree model.
dt = DecisionTreeClassifier(labelCol="default_payment_next_month", featuresCol
="scaledFeatures")

# Chain indexers and tree in a Pipeline
#pipeline = Pipeline(stages=[labelIndexer, featureIndexer, dt])
pipeline = Pipeline(stages=[dt])

# Train model.  This also runs the indexers.
model = pipeline.fit(trainingData)
```

In [5]:

```
# 4. Model evaluation

# Make predictions.
predictions = model.transform(testData)

from pyspark.ml.evaluation import MulticlassClassificationEvaluator
# Select (prediction, true label) and compute test error
evaluator = MulticlassClassificationEvaluator(
    labelCol="default_payment_next_month", predictionCol="prediction", metricN
ame="accuracy")


accuracy = evaluator.evaluate(predictions)
print("DecisionTreeClassifier accuracy = ", str(format( (accuracy)*100, '.2f')
) + "%")
```

```
DecisionTreeClassifier accuracy =  81.69%
```

Accuracy of the model is around 82%

In [6]:

```
# Select example rows to display.
print("Some values to verify our results:-")
predictions.select("prediction", "default_payment_next_month").show(10)
```

```
Some values to verify our results:-
+----------+-------------------------+
|prediction|default_payment_next_month|
+----------+-------------------------+
|       0.0|                      0.0|
|       0.0|                      0.0|
|       0.0|                      0.0|
|       0.0|                      0.0|
|       0.0|                      0.0|
|       0.0|                      0.0|
|       0.0|                      0.0|
|       0.0|                      0.0|
|       0.0|                      0.0|
|       0.0|                      0.0|
+----------+-------------------------+
only showing top 10 rows
```