Welcome to exercise one of week four of “Apache Spark for Scalable Machine Learning on BigData”. In this exercise we’ll work on classification.

Let’s create our DataFrame again:

```
In [ ]: # 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')
```

Since this is supervised learning, let’s split our data into train (80%) and test (20%) set.

```
In [ ]: splits = df.randomSplit([0.8, 0.2])
        df_train = splits[0]
        df_test = splits[1]
```

Again, we can re-use our feature engineering pipeline

```
In [ ]: from pyspark.ml.feature import StringIndexer, OneHotEncoder
        from pyspark.ml.linalg import Vectors
        from pyspark.ml.feature import VectorAssembler
        from pyspark.ml.feature import Normalizer


        indexer = StringIndexer(inputCol="class", outputCol="label")

        vectorAssembler = VectorAssembler(inputCols=["x","y","z"],
                                          outputCol="features")

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

Now we use LogisticRegression, a simple and basic linear classifier to obtain a classification performance baseline.

```
In [ ]: from pyspark.ml.classification import LogisticRegression
        from pyspark.ml import Pipeline

        lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)
        pipeline = Pipeline(stages=[indexer, vectorAssembler, normalizer,lr])
        model = pipeline.fit(df_train)
        prediction = model.transform(df_test)
```

If we look at the schema of the prediction dataframe we see that there is an additional column called prediction which contains the best guess for the class our model predicts.

```
In [ ]: prediction.printSchema()
```

Let’s evaluate performance by using a build-in functionality of Apache SparkML.

```
In [ ]: from pyspark.ml.evaluation import MulticlassClassificationEvaluator
        MulticlassClassificationEvaluator().setMetricName("accuracy").evaluate
        (prediction)
```

So we get 20% right. This is not bad for a baseline. Note that random guessing would give us only 7%. Of course we need to improve. You might have notices that we’re dealing with a time series here. And we’re not making use of that fact right now as we look at each training example only individually. But this is ok for now. More advanced courses like “Advanced Machine Learning and Signal Processing” (https://www.coursera.org/learn/advanced-machine-learning-signal-processing/) will teach you how to improve accuracy to the nearly 100% by using algorithms like Fourier transformation or wavelet transformation. But let’s skip this for now. In the following cell, please use the RandomForest classifier (you might need to play with the “numTrees” parameter) in the code cell below. You should get an accuracy of around 44%. More on RandomForest can be found here:

https://spark.apache.org/docs/latest/ml-classification-regression.html#random-forest-classifier

```
In [ ]: $$
```