Week 4 commands

```
val rawData = sc.textFile("hdfs:///user/user1/kddcup.data")

rawData.count

rawData.take(10).foreach(println)


import org.apache.spark.mllib.linalg._


val labelsAndData = rawData.map { line =>
val buffer = line.split(',').toBuffer
buffer.remove(1,3)
val label = buffer.remove(buffer.length-1)
val vector = Vectors.dense(buffer.map(_.toDouble).toArray)
(label,vector)
}

val sample = labelsAndData.sample(false,0.05).values.cache()

import org.apache.spark.mllib.clustering._

val kmeans = new KMeans()

kmeans.setK(100)

val model = kmeans.run(sample)

model.clusterCenters.foreach(println)
```

```
def distance(a: Vector, b: Vector) =
math.sqrt(a.toArray.zip(b.toArray).map(p => p._1 - p._2).map(d =>
d*d).sum)

def distanceToCentroid(datum: Vector, model: KMeansModel) = {
val cluster = model.predict(datum)
val centroid = model.clusterCenters(cluster)
distance(centroid,datum)
}

val distances = sample.map(datum =>
distanceToCentroid(datum,model))

distances.top(10).foreach(println)

val threshold = distances.top(100).last


val originalAndData = rawData.zip(labelsAndData.values)

val anomalies = originalAndData.filter { case (str,datum) =>
distanceToCentroid(datum,model) > threshold }.keys

anomalies.count


anomalies.toDebugString


anomalies.take(2).foreach(println)
```