**Spark-Zeppelin-Uber**

Log file location: s3://aws-logs-995025397421-ap-south-1/elasticmapreduce/

Cluster: Spark master class

Global DNS: ec2-13-127-13-225.ap-south-1.compute.amazonaws.com

Zeppelin: ec2-13-127-13-225.ap-south-1.compute.amazonaws.com:8890

Data location: 's3://hivetableau/Uber/uber.csv'

Part I:

import org.apache.spark.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.\_

import org.apache.spark.sql.Dataset

import org.apache.spark.sql.types.TimestampType

import org.apache.spark.ml.feature.VectorAssembler

import org.apache.spark.ml.clustering.KMeans

val spark: SparkSession = SparkSession.builder().appName("uber").getOrCreate()

import spark.implicits.\_

val schema = StructType(Array(

StructField("dt", TimestampType, true),

StructField("lat", DoubleType, true),

StructField("lon", DoubleType, true),

StructField("base", StringType, true)

))

val df = spark.read.option("inferSchema", "false").schema(schema).csv("s3://hivetableau/Uber/uber.csv")

df.cache

df.show

df.schema

val featureCols = Array("lat", "lon")

val assembler = new VectorAssembler().setInputCols(featureCols).setOutputCol("features")

val df2 = assembler.transform(df)

val Array(trainingData, testData) = df2.randomSplit(Array(0.7, 0.3), 5043)

val kmeans = new KMeans().setK(20).setFeaturesCol("features").setMaxIter(5)

val model = kmeans.fit(trainingData)

println("Final Centers: ")

model.clusterCenters.foreach(println)

val categories = model.transform(testData)

categories.show

categories.createOrReplaceTempView("uber")

categories.select(month($"dt").alias("month"), dayofmonth($"dt").alias("day"), hour($"dt").alias("hour"), $"prediction").groupBy("month", "day", "hour", "prediction").agg(count("prediction").alias("count")).orderBy("day", "hour", "prediction").show

categories.select(hour($"dt").alias("hour"), $"prediction").groupBy("hour", "prediction").agg(count("prediction")

.alias("count")).orderBy(desc("count")).show

categories.groupBy("prediction").count().show()

spark.sql("select prediction, count(prediction) as count from uber group by prediction").show

spark.sql("SELECT hour(uber.dt) as hr,count(prediction) as ct FROM uber group By hour(uber.dt)").show

model.write.overwrite().save(“s3://hivetableau/Uber/model")

val res = spark.sql("select dt, lat, lon, base, prediction as cid FROM uber order by dt")

res.write.format("json").save("s3://hivetableau/Uber/model1")

Part II:

import org.apache.spark.\_

import org.apache.spark.rdd.RDD

import org.apache.spark.util.IntParam

import org.apache.spark.sql.SQLContext

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.\_

import org.apache.spark.mllib.stat.Statistics

import sqlContext.implicits.\_

import sqlContext.\_

val df = sqlContext.read.json("s3://hivetableau/Uber/model1").cache()

df.printSchema

df.show

df.select(month($"dt").alias("month"), dayofmonth($"dt").alias("day"), $"cid").groupBy("month", "day", "cid").agg(count("cid").alias("count")).orderBy(desc("count")).show

df.select(month($"dt").alias("month"), dayofmonth($"dt").alias("day"), hour($"dt").alias("hour"), $"cid").groupBy("month", "day", "hour", "cid").agg(count("cid").alias("count")).orderBy(desc("count"), desc("cid")).show

df.select(month($"dt").alias("month"), dayofmonth($"dt").alias("day"), hour($"dt").alias("hour"), $"cid").groupBy("month", "day", "cid").agg(count("cid").alias("count")).orderBy(asc("day")).show

df.select(month($"dt").alias("month"), dayofmonth($"dt").alias("day"), hour($"dt").alias("hour"), $"cid").groupBy("month", "day", "hour", "cid").agg(count("cid").alias("count")).orderBy("day", "hour", "cid").show

df.select(month($"dt").alias("month"), dayofmonth($"dt").alias("day"), hour($"dt").alias("hour"), $"cid").groupBy("month", "day", "hour", "cid").agg(count("cid").alias("count")).orderBy("day", "hour", "cid").show