Module 7: Machine Learning using Spark MLlib

Project Solution

Project Solution

Uber Dataset Analysis using MLlib

Answer:

```
import org.apache.spark._
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.types._
import org.apache.spark.sql.functions._
import org.apache.spark.sql. _
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.clustering.KMeans
import org.apache.spark.storage.StorageLevel
import scala.io.Source
import scala.collection.mutable.HashMap
import java.io.File
import org.apache.spark.sql.Row
import scala.collection.mutable.ListBuffer
import org.apache.spark.util.IntParam
import org.apache.spark.util.StatCounter
import org.apache.spark.rdd.RDD
import org.apache.spark.rdd._
```

```
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
import sqlContext.implicits._
import sqlContext._
```

```
val schema = StructType(Array(StructField("dt", StringType, true), StructField("lat",
        DoubleType, true), StructField("lon", DoubleType, true), StructField("base",
        StringType, true)))
```

//add path to the csv file uploaded on hdfs

```
val df = spark.read.option("header",
        "true").schema(schema).csv("hdfs://nameservice1/user/edureka_253770/ub
        er1.csv")
```

```
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), 8743)
val kmeans = new
        KMeans().setK(8).setFeaturesCol("features").setPredictionCol("prediction")
val model = kmeans.fit(df2)
println("Final Centers: ")
model.clusterCenters.foreach(println)
```

```
val categories = model.transform(df2)
```

```
val q = categories.select(hour($"dt").alias("hour"), $"prediction").groupBy("hour",
        "prediction").agg(count("prediction").alias("count")).orderBy(desc("count"))
val t = categories.select(hour($"dt").alias("hour"), $"lat", $"lon",
        $"prediction").filter($"hour".isNotNull)
categories.createOrReplaceTempView("categories")
t.createOrReplaceTempView("t")
```

```
q.createOrReplaceTempView("q")
import org.apache.spark._
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.types._
import org.apache.spark.sql.functions._
import org.apache.spark.sql._
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.clustering.KMeans
import org.apache.spark.storage.StorageLevel
import scala.io.Source
import scala.collection.mutable.HashMap
import java.io.File
import org.apache.spark.sql.Row
import scala.collection.mutable.ListBuffer
import org.apache.spark.util.IntParam
import org.apache.spark.util.StatCounter
import org.apache.spark.rdd.RDD
import org.apache.spark.rdd._
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext =
        org.apache.spark.sql.SQLContext@3399cc
import sqlContext.implicits._
import sqlContext._
schema: org.apache.spark.sql.types.StructType =
        StructType(StructField(dt,StringType,true), StructField(lat,DoubleType,true),
        StructField(lon,DoubleType,true), StructField(base,StringType,true))
df: org.apache.spark.sql.DataFrame = [dt: string, lat: double ... 2 more fields]
featureCols: Array[String] = Array(lat, lon)
```

```
assembler: org.apache.spark.ml.feature.VectorAssembler =
        vecAssembler_67d420874b0d

df2: org.apache.spark.sql.DataFrame = [dt: string, lat: double ... 3 more fields]

kmeans: org.apache.spark.ml.clustering.KMeans = kmeans_257f64e2a4dd

model: org.apache.spark.ml.clustering.KMeansModel = kmeans_257f64e2a4dd

Final Centers:

[40.72558487769913,-74.00663509247732]

[40.78319953048764,-73.87016222235611]

[40.6729193088002,-73.98290215039974]

[40.65586990514615,-73.7791239652199]

[40.75370251056143,-73.98312052115025]

[40.78138051668325,-73.96010637358509]

[40.70732481637808,-73.9459141779388]

[40.97125845213848,-73.61148625254575]

categories: org.apache.spark.sql.DataFrame = [dt: string, lat: double ... 4 more fields]

q: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [hour: int, prediction: int
        ... 1 more field]

t: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [hour: int, lat: double ...
        2 more fields]

val p = q.filter($"hour".isNotNull)

//val p = spark.sql("DELETE FROM q where hour IS NULL")

p.createOrReplaceTempView("p")
```