package com.twitter.follow\_recommendations.common.models

import com.twitter.follow\_recommendations.thriftscala.DebugDataRecord

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.FeatureContext

import com.twitter.util.Try

import com.twitter.util.logging.Logging

import scala.collection.convert.ImplicitConversions.\_

// contains the standard dataRecord struct, and the debug version if required

case class RichDataRecord(

dataRecord: Option[DataRecord] = None,

debugDataRecord: Option[DebugDataRecord] = None,

)

trait HasDataRecord extends Logging {

def dataRecord: Option[RichDataRecord]

def toDebugDataRecord(dr: DataRecord, featureContext: FeatureContext): DebugDataRecord = {

val binaryFeatures: Option[Set[String]] = if (dr.isSetBinaryFeatures) {

Some(dr.getBinaryFeatures.flatMap { id =>

Try(featureContext.getFeature(id).getFeatureName).toOption

}.toSet)

} else None

val continuousFeatures: Option[Map[String, Double]] = if (dr.isSetContinuousFeatures) {

Some(dr.getContinuousFeatures.flatMap {

case (id, value) =>

Try(featureContext.getFeature(id).getFeatureName).toOption.map { id =>

id -> value.toDouble

}

}.toMap)

} else None

val discreteFeatures: Option[Map[String, Long]] = if (dr.isSetDiscreteFeatures) {

Some(dr.getDiscreteFeatures.flatMap {

case (id, value) =>

Try(featureContext.getFeature(id).getFeatureName).toOption.map { id =>

id -> value.toLong

}

}.toMap)

} else None

val stringFeatures: Option[Map[String, String]] = if (dr.isSetStringFeatures) {

Some(dr.getStringFeatures.flatMap {

case (id, value) =>

Try(featureContext.getFeature(id).getFeatureName).toOption.map { id =>

id -> value

}

}.toMap)

} else None

val sparseBinaryFeatures: Option[Map[String, Set[String]]] = if (dr.isSetSparseBinaryFeatures) {

Some(dr.getSparseBinaryFeatures.flatMap {

case (id, values) =>

Try(featureContext.getFeature(id).getFeatureName).toOption.map { id =>

id -> values.toSet

}

}.toMap)

} else None

val sparseContinuousFeatures: Option[Map[String, Map[String, Double]]] =

if (dr.isSetSparseContinuousFeatures) {

Some(dr.getSparseContinuousFeatures.flatMap {

case (id, values) =>

Try(featureContext.getFeature(id).getFeatureName).toOption.map { id =>

id -> values.map {

case (str, value) =>

str -> value.toDouble

}.toMap

}

}.toMap)

} else None

DebugDataRecord(

binaryFeatures = binaryFeatures,

continuousFeatures = continuousFeatures,

discreteFeatures = discreteFeatures,

stringFeatures = stringFeatures,

sparseBinaryFeatures = sparseBinaryFeatures,

sparseContinuousFeatures = sparseContinuousFeatures,

)

}

}