package com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator.real\_time\_aggregates

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.constant.SharedFeatures.TIMESTAMP

import com.twitter.util.Duration

/\*\*

\* The default TimeDecay implementation for real time aggregates.

\*

\* @param featureIdToHalfLife A precomputed map from aggregate feature ids to their half lives.

\* @param timestampFeatureId A discrete timestamp feature id.

\*/

case class RealTimeAggregateTimeDecay(

featureIdToHalfLife: Map[Long, Duration],

timestampFeatureId: Long = TIMESTAMP.getFeatureId) {

/\*\*

\* Mutates the data record which is just a reference to the input.

\*

\* @param record Data record to apply decay to (is mutated).

\* @param timeNow The current read time (in milliseconds) to decay counts forward to.

\*/

def apply(record: DataRecord, timeNow: Long): Unit = {

if (record.isSetDiscreteFeatures) {

val discreteFeatures = record.getDiscreteFeatures

if (discreteFeatures.containsKey(timestampFeatureId)) {

if (record.isSetContinuousFeatures) {

val ctsFeatures = record.getContinuousFeatures

val storedTimestamp: Long = discreteFeatures.get(timestampFeatureId)

val scaledDt = if (timeNow > storedTimestamp) {

(timeNow - storedTimestamp).toDouble \* math.log(2)

} else 0.0

featureIdToHalfLife.foreach {

case (featureId, halfLife) =>

if (ctsFeatures.containsKey(featureId)) {

val storedValue = ctsFeatures.get(featureId)

val alpha =

if (halfLife.inMilliseconds != 0) math.exp(-scaledDt / halfLife.inMilliseconds)

else 0

val decayedValue: Double = alpha \* storedValue

record.putToContinuousFeatures(featureId, decayedValue)

}

}

}

discreteFeatures.remove(timestampFeatureId)

}

}

}

}