package com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.metrics

import com.twitter.algebird.DecayedValue

import com.twitter.algebird.DecayedValueMonoid

import com.twitter.algebird.Monoid

import com.twitter.dal.personal\_data.thriftjava.PersonalDataType

import com.twitter.ml.api.\_

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

import com.twitter.ml.api.util.SRichDataRecord

import com.twitter.util.Duration

import java.lang.{Long => JLong}

import java.util.{HashSet => JHashSet}

import java.util.{Set => JSet}

object AggregationMetricCommon {

/\* Shared definitions and utils that can be reused by child classes \*/

val Epsilon: Double = 1e-6

val decayedValueMonoid: Monoid[DecayedValue] = DecayedValueMonoid(Epsilon)

val TimestampHash: JLong = SharedFeatures.TIMESTAMP.getDenseFeatureId()

def toDecayedValue(tv: TimedValue[Double], halfLife: Duration): DecayedValue = {

DecayedValue.build(

tv.value,

tv.timestamp.inMilliseconds,

halfLife.inMilliseconds

)

}

def getTimestamp(

record: DataRecord,

timestampFeature: Feature[JLong] = SharedFeatures.TIMESTAMP

): Long = {

Option(

SRichDataRecord(record)

.getFeatureValue(timestampFeature)

).map(\_.toLong)

.getOrElse(0L)

}

/\*

\* Union the PDTs of the input featureOpts.

\* Return null if empty, else the JSet[PersonalDataType]

\*/

def derivePersonalDataTypes(features: Option[Feature[\_]]\*): JSet[PersonalDataType] = {

val unionPersonalDataTypes = new JHashSet[PersonalDataType]()

for {

featureOpt <- features

feature <- featureOpt

pdtSetOptional = feature.getPersonalDataTypes

if pdtSetOptional.isPresent

pdtSet = pdtSetOptional.get

} unionPersonalDataTypes.addAll(pdtSet)

if (unionPersonalDataTypes.isEmpty) null else unionPersonalDataTypes

}

}