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

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

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.Feature

import com.twitter.scalding\_internal.multiformat.format.keyval.KeyValInjection

import com.twitter.scalding\_internal.multiformat.format.keyval.KeyValInjection.Batched

import com.twitter.scalding\_internal.multiformat.format.keyval.KeyValInjection.JavaCompactThrift

import com.twitter.scalding\_internal.multiformat.format.keyval.KeyValInjection.genericInjection

import com.twitter.summingbird.batch.BatchID

import scala.collection.JavaConverters.\_

object OfflineAggregateInjections {

val offlineDataRecordAggregateInjection: KeyValInjection[AggregationKey, (BatchID, DataRecord)] =

KeyValInjection(

genericInjection(AggregationKeyInjection),

Batched(JavaCompactThrift[DataRecord])

)

private[aggregation\_framework] def getPdts[T](

aggregateGroups: Iterable[T],

featureExtractor: T => Iterable[Feature[\_]]

): Option[Set[PersonalDataType]] = {

val pdts: Set[PersonalDataType] = for {

group <- aggregateGroups.toSet[T]

feature <- featureExtractor(group)

pdtSet <- feature.getPersonalDataTypes.asSet().asScala

javaPdt <- pdtSet.asScala

scalaPdt <- PersonalDataType.get(javaPdt.getValue)

} yield {

scalaPdt

}

if (pdts.nonEmpty) Some(pdts) else None

}

def getInjection(

aggregateGroups: Set[TypedAggregateGroup[\_]]

): KeyValInjection[AggregationKey, (BatchID, DataRecord)] = {

val keyPdts = getPdts[TypedAggregateGroup[\_]](aggregateGroups, \_.allOutputKeys)

val valuePdts = getPdts[TypedAggregateGroup[\_]](aggregateGroups, \_.allOutputFeatures)

KeyValInjection(

genericInjection(AggregationKeyInjection, keyPdts),

genericInjection(Batched(JavaCompactThrift[DataRecord]), valuePdts)

)

}

}