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

import com.twitter.util.Duration

import com.twitter.ml.api.\_

import java.lang.{Boolean => JBoolean}

/\*\*

\* Case class used as shared argument for

\* getAggregateValue() and setAggregateValue() in AggregationMetric.

\*

\* @param aggregatePrefix Prefix for aggregate feature name

\* @param feature Simple (non-aggregate) feature being aggregated. This

is optional; if None, then the label is aggregated on its own without

being crossed with any feature.

\* @param label Label being paired with. This is optional; if None, then

the feature is aggregated on its own without being crossed with any label.

\* @param halfLife Half life being used for aggregation

\*/

case class AggregateFeature[T](

aggregatePrefix: String,

feature: Option[Feature[T]],

label: Option[Feature[JBoolean]],

halfLife: Duration) {

val aggregateType = "pair"

val labelName: String = label.map(\_.getDenseFeatureName()).getOrElse("any\_label")

val featureName: String = feature.map(\_.getDenseFeatureName()).getOrElse("any\_feature")

/\*

\* This val precomputes a portion of the feature name

\* for faster processing. String building turns

\* out to be a significant bottleneck.

\*/

val featurePrefix: String = List(

aggregatePrefix,

aggregateType,

labelName,

featureName,

halfLife.toString

).mkString(".")

}

/\* Companion object with util methods. \*/

object AggregateFeature {

def parseHalfLife(aggregateFeature: Feature[\_]): Duration = {

val aggregateComponents = aggregateFeature.getDenseFeatureName().split("\\.")

val numComponents = aggregateComponents.length

val halfLifeStr = aggregateComponents(numComponents - 3) + "." +

aggregateComponents(numComponents - 2)

Duration.parse(halfLifeStr)

}

}