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

import com.twitter.ml.api.\_

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

import com.twitter.util.Time

import java.lang.{Double => JDouble}

import java.lang.{Long => JLong}

case class TypedSumSqMetric() extends TypedSumLikeMetric[JDouble] {

import AggregationMetricCommon.\_

override val operatorName = "sumsq"

/\*

\* Transform feature -> its squared value in the given record

\* or 0 when feature = None (sumsq has no meaning in this case)

\*/

override def getIncrementValue(

record: DataRecord,

feature: Option[Feature[JDouble]],

timestampFeature: Feature[JLong]

): TimedValue[Double] = feature match {

case Some(f) => {

val featureVal =

Option(SRichDataRecord(record).getFeatureValue(f)).map(\_.toDouble).getOrElse(0.0)

TimedValue[Double](

value = featureVal \* featureVal,

timestamp = Time.fromMilliseconds(getTimestamp(record, timestampFeature))

)

}

case None =>

TimedValue[Double](

value = 0.0,

timestamp = Time.fromMilliseconds(getTimestamp(record, timestampFeature))

)

}

}

/\*\*

\* Syntactic sugar for the sum of squares metric that works with continuous features.

\* See EasyMetric.scala for more details on why this is useful.

\*/

object SumSqMetric extends EasyMetric {

override def forFeatureType[T](

featureType: FeatureType

): Option[AggregationMetric[T, \_]] =

featureType match {

case FeatureType.CONTINUOUS =>

Some(TypedSumSqMetric().asInstanceOf[AggregationMetric[T, Double]])

case \_ => None

}

}