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.{Long => JLong}

case class TypedCountMetric[T](

) extends TypedSumLikeMetric[T] {

import AggregationMetricCommon.\_

import ConversionUtils.\_

override val operatorName = "count"

override def getIncrementValue(

record: DataRecord,

feature: Option[Feature[T]],

timestampFeature: Feature[JLong]

): TimedValue[Double] = {

val featureExists: Boolean = feature match {

case Some(f) => SRichDataRecord(record).hasFeature(f)

case None => true

}

TimedValue[Double](

value = booleanToDouble(featureExists),

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

)

}

}

/\*\*

\* Syntactic sugar for the count metric that works with

\* any feature type as opposed to being tied to a specific type.

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

\*/

object CountMetric extends EasyMetric {

override def forFeatureType[T](

featureType: FeatureType,

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

Some(TypedCountMetric[T]())

}