package com.twitter.timelines.data\_processing.ad\_hoc.earlybird\_ranking.training\_data\_generation

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

import com.twitter.ml.api.DataSetPipe

import com.twitter.ml.api.Feature

import com.twitter.timelines.data\_processing.ad\_hoc.earlybird\_ranking.common.LabelInfo

import com.twitter.timelines.data\_processing.ad\_hoc.earlybird\_ranking.common.LabelInfoWithFeature

import com.twitter.timelines.prediction.features.recap.RecapFeatures

import java.lang.{Double => JDouble}

import scala.util.Random

/\*\*

\* Adds an IsGlobalEngagement label to records containing any recap label, and adjusts

\* weights accordingly. See [[weightAndSample]] for details on operation.

\*/

class EarlybirdExampleSampler(

random: Random,

labelInfos: List[LabelInfoWithFeature],

negativeInfo: LabelInfo) {

import com.twitter.ml.api.util.FDsl.\_

private[this] val ImportanceFeature: Feature[JDouble] =

SharedFeatures.RECORD\_WEIGHT\_FEATURE\_BUILDER

.extensionBuilder()

.addExtension("type", "earlybird")

.build()

private[this] def uniformSample(labelInfo: LabelInfo) =

random.nextDouble() < labelInfo.downsampleFraction

private[this] def weightedImportance(labelInfo: LabelInfo) =

labelInfo.importance / labelInfo.downsampleFraction

/\*\*

\* Generates a IsGlobalEngagement label for records that contain any

\* recap label. Adds an "importance" value per recap label found

\* in the record. Simultaneously, downsamples positive and negative examples based on provided

\* downsample rates.

\*/

def weightAndSample(data: DataSetPipe): DataSetPipe = {

val updatedRecords = data.records.flatMap { record =>

val featuresOn = labelInfos.filter(labelInfo => record.hasFeature(labelInfo.feature))

if (featuresOn.nonEmpty) {

val sampled = featuresOn.map(\_.info).filter(uniformSample)

if (sampled.nonEmpty) {

record.setFeatureValue(RecapFeatures.IS\_EARLYBIRD\_UNIFIED\_ENGAGEMENT, true)

Some(record.setFeatureValue(ImportanceFeature, sampled.map(weightedImportance).sum))

} else {

None

}

} else if (uniformSample(negativeInfo)) {

Some(record.setFeatureValue(ImportanceFeature, weightedImportance(negativeInfo)))

} else {

None

}

}

DataSetPipe(

updatedRecords,

data.featureContext

.addFeatures(ImportanceFeature, RecapFeatures.IS\_EARLYBIRD\_UNIFIED\_ENGAGEMENT)

)

}

}