package com.twitter.home\_mixer.functional\_component.scorer

import com.twitter.home\_mixer.model.HomeFeatures.InNetworkFeature

import com.twitter.home\_mixer.model.HomeFeatures.IsRetweetFeature

import com.twitter.home\_mixer.model.HomeFeatures.ScoreFeature

import com.twitter.product\_mixer.component\_library.model.candidate.TweetCandidate

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.scorer.Scorer

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

import com.twitter.product\_mixer.core.model.common.identifier.ScorerIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.stitch.Stitch

/\*\*

\* Scales scores of each out-of-network tweet by the specified scale factor

\*/

object OONTweetScalingScorer extends Scorer[PipelineQuery, TweetCandidate] {

override val identifier: ScorerIdentifier = ScorerIdentifier("OONTweetScaling")

override val features: Set[Feature[\_, \_]] = Set(ScoreFeature)

private val ScaleFactor = 0.75

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = {

Stitch.value {

candidates.map { candidate =>

val score = candidate.features.getOrElse(ScoreFeature, None)

val updatedScore = if (selector(candidate)) score.map(\_ \* ScaleFactor) else score

FeatureMapBuilder().add(ScoreFeature, updatedScore).build()

}

}

}

/\*\*

\* We should only be applying this multiplier to Out-Of-Network tweets.

\* In-Network Retweets of Out-Of-Network tweets should not have this multiplier applied

\*/

private def selector(candidate: CandidateWithFeatures[TweetCandidate]): Boolean = {

!candidate.features.getOrElse(InNetworkFeature, false) &&

!candidate.features.getOrElse(IsRetweetFeature, false)

}

}