package com.twitter.frigate.pushservice.predicate

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.frigate.pushservice.params.PushFeatureSwitchParams

import com.twitter.frigate.pushservice.params.PushParams

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.hermit.predicate.Predicate

import com.twitter.util.Future

object MlModelsHoldbackExperimentPredicate {

val name = "MlModelsHoldbackExperimentPredicate"

private val alwaysTruePred = PredicatesForCandidate.alwaysTruePushCandidatePredicate

def getPredicateBasedOnCandidate(

pc: PushCandidate,

treatmentPred: Predicate[PushCandidate]

)(

implicit statsReceiver: StatsReceiver

): Future[Predicate[PushCandidate]] = {

Future

.join(Future.value(pc.target.skipFilters), pc.target.isInModelExclusionList)

.map {

case (skipFilters, isInModelExclusionList) =>

if (skipFilters ||

isInModelExclusionList ||

pc.target.params(PushParams.DisableMlInFilteringParam) ||

pc.target.params(PushFeatureSwitchParams.DisableMlInFilteringFeatureSwitchParam) ||

pc.target.params(PushParams.DisableAllRelevanceParam) ||

pc.target.params(PushParams.DisableHeavyRankingParam)) {

alwaysTruePred

} else {

treatmentPred

}

}

}

def apply()(implicit statsReceiver: StatsReceiver): NamedPredicate[PushCandidate] = {

val stats = statsReceiver.scope(s"predicate\_$name")

val statsProd = stats.scope("prod")

val counterAcceptedByModel = statsProd.counter("accepted")

val counterRejectedByModel = statsProd.counter("rejected")

val counterHoldback = stats.scope("holdback").counter("all")

val jointDauQualityPredicate = JointDauAndQualityModelPredicate()

new Predicate[PushCandidate] {

def apply(items: Seq[PushCandidate]): Future[Seq[Boolean]] = {

val boolFuts = items.map { item =>

getPredicateBasedOnCandidate(item, jointDauQualityPredicate)(statsReceiver)

.flatMap { predicate =>

val predictionFut = predicate.apply(Seq(item)).map(\_.headOption.getOrElse(false))

predictionFut.foreach { prediction =>

if (item.target.params(PushParams.DisableMlInFilteringParam) || item.target.params(

PushFeatureSwitchParams.DisableMlInFilteringFeatureSwitchParam)) {

counterHoldback.incr()

} else {

if (prediction) counterAcceptedByModel.incr() else counterRejectedByModel.incr()

}

}

predictionFut

}

}

Future.collect(boolFuts)

}

}.withStats(stats)

.withName(name)

}

}