package com.twitter.product\_mixer.component\_library.scorer.deepbird

import com.twitter.cortex.deepbird.{thriftjava => t}

import com.twitter.ml.prediction\_service.BatchPredictionRequest

import com.twitter.ml.prediction\_service.BatchPredictionResponse

import com.twitter.product\_mixer.component\_library.scorer.common.ModelSelector

import com.twitter.product\_mixer.core.feature.datarecord.BaseDataRecordFeature

import com.twitter.product\_mixer.core.feature.featuremap.datarecord.FeaturesScope

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

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

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

import com.twitter.util.Future

/\*\*

\* Configurable Scorer that calls any Deepbird Prediction Service thrift.

\* @param identifier Unique identifier for the scorer

\* @param predictionService The Prediction Thrift Service

\* @param modelSelector Model ID Selector to decide which model to select, can also be represented

\* as an anonymous function: { query: Query => Some("Ex") }

\* @param queryFeatures The Query Features to convert and pass to the deepbird model.

\* @param candidateFeatures The Candidate Features to convert and pass to the deepbird model.

\* @param resultFeatures The Candidate features returned by the model.

\* @tparam Query Type of pipeline query.

\* @tparam Candidate Type of candidates to score.

\* @tparam QueryFeatures type of the query level features consumed by the scorer.

\* @tparam CandidateFeatures type of the candidate level features consumed by the scorer.

\* @tparam ResultFeatures type of the candidate level features returned by the scorer.

\*/

case class DeepbirdV2PredictionServerScorer[

Query <: PipelineQuery,

Candidate <: UniversalNoun[Any],

QueryFeatures <: BaseDataRecordFeature[Query, \_],

CandidateFeatures <: BaseDataRecordFeature[Candidate, \_],

ResultFeatures <: BaseDataRecordFeature[Candidate, \_]

](

override val identifier: ScorerIdentifier,

predictionService: t.DeepbirdPredictionService.ServiceToClient,

modelSelector: ModelSelector[Query],

queryFeatures: FeaturesScope[QueryFeatures],

candidateFeatures: FeaturesScope[CandidateFeatures],

resultFeatures: Set[ResultFeatures])

extends BaseDeepbirdV2Scorer[

Query,

Candidate,

QueryFeatures,

CandidateFeatures,

ResultFeatures

](identifier, modelSelector, queryFeatures, candidateFeatures, resultFeatures) {

override def getBatchPredictions(

request: BatchPredictionRequest,

modelSelector: t.ModelSelector

): Future[BatchPredictionResponse] =

predictionService.batchPredictFromModel(request, modelSelector)

}