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

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.product\_mixer.component\_library.scorer.tensorbuilder.ModelInferRequestBuilder

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

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 javax.inject.Inject

import javax.inject.Singleton

@Singleton

class CortexManagedInferenceServiceTensorScorerBuilder @Inject() (

statsReceiver: StatsReceiver) {

/\*\*

\* Builds a configurable Scorer to call into your desired Cortex Managed ML Model Service.

\*

\* If your service does not bind an Http.Client implementation, add

\* [[com.twitter.product\_mixer.component\_library.module.http.FinagleHttpClientModule]]

\* to your server module list

\*

\* @param scorerIdentifier Unique identifier for the scorer

\* @param resultFeatureExtractors The result features an their tensor extractors for each candidate.

\* @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.

\*/

def build[Query <: PipelineQuery, Candidate <: UniversalNoun[Any]](

scorerIdentifier: ScorerIdentifier,

modelInferRequestBuilder: ModelInferRequestBuilder[

Query,

Candidate

],

resultFeatureExtractors: Seq[FeatureWithExtractor[Query, Candidate, \_]],

client: MLModelInferenceClient

): Scorer[Query, Candidate] =

new CortexManagedInferenceServiceTensorScorer(

scorerIdentifier,

modelInferRequestBuilder,

resultFeatureExtractors,

client,

statsReceiver.scope(scorerIdentifier.name)

)

}