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

import com.twitter.finagle.Http

import com.twitter.product\_mixer.component\_library.module.http.FinagleHttpClientModule.FinagleHttpClientModule

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

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.datarecord.TensorDataRecordCompatible

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

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.Named

import javax.inject.Singleton

@Singleton

class CortexManagedInferenceServiceDataRecordScorerBuilder @Inject() (

@Named(FinagleHttpClientModule) httpClient: Http.Client) {

/\*\*

\* Builds a configurable Scorer to call into your desired DataRecord-backed 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 modelPath MLS path to model

\* @param modelSignature Model Signature Key

\* @param modelSelector [[ModelSelector]] for choosing the model name, can be an anon function.

\* @param candidateFeatures Desired candidate level feature store features to pass to the model.

\* @param resultFeatures Desired candidate level feature store features to extract from the model.

\* Since the Cortex Managed Platform always returns tensor values, the

\* feature must use a [[TensorDataRecordCompatible]].

\* @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.

\*/

def build[

Query <: PipelineQuery,

Candidate <: UniversalNoun[Any],

QueryFeatures <: BaseDataRecordFeature[Query, \_],

CandidateFeatures <: BaseDataRecordFeature[Candidate, \_],

ResultFeatures <: BaseDataRecordFeature[Candidate, \_] with TensorDataRecordCompatible[\_]

](

scorerIdentifier: ScorerIdentifier,

modelPath: String,

modelSignature: String,

modelSelector: ModelSelector[Query],

queryFeatures: FeaturesScope[QueryFeatures],

candidateFeatures: FeaturesScope[CandidateFeatures],

resultFeatures: Set[ResultFeatures]

): Scorer[Query, Candidate] =

new CortexManagedDataRecordScorer(

identifier = scorerIdentifier,

modelSignature = modelSignature,

modelSelector = modelSelector,

modelClient = ManagedModelClient(httpClient, modelPath),

queryFeatures = queryFeatures,

candidateFeatures = candidateFeatures,

resultFeatures = resultFeatures

)

}