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

import com.twitter.finagle.Http

import com.twitter.finagle.grpc.FinagleChannelBuilder

import com.twitter.finagle.grpc.FutureConverters

import com.twitter.mlserving.frontend.TFServingInferenceServiceImpl

import com.twitter.stitch.Stitch

import tensorflow.serving.PredictionServiceGrpc

import inference.GrpcService.ModelInferRequest

import inference.GrpcService.ModelInferResponse

import io.grpc.ManagedChannel

import io.grpc.Status

/\*\*

\* Client wrapper for calling a Navi Inference Service (go/navi).

\* @param httpClient Finagle HTTP Client to use for connection.

\* @param modelPath Wily path to the ML Model service (e.g. /s/role/service).

\*/

case class NaviModelClient(

httpClient: Http.Client,

modelPath: String)

extends MLModelInferenceClient {

private val channel: ManagedChannel =

FinagleChannelBuilder

.forTarget(modelPath)

.httpClient(httpClient)

// Navi enforces an authority name.

.overrideAuthority("rustserving")

// certain GRPC errors need to be retried.

.enableRetryForStatus(Status.UNKNOWN)

.enableRetryForStatus(Status.RESOURCE\_EXHAUSTED)

// this is required at channel level as mTLS is enabled at httpClient level

.usePlaintext()

.build()

private val inferenceServiceStub = PredictionServiceGrpc.newFutureStub(channel)

def score(request: ModelInferRequest): Stitch[ModelInferResponse] = {

val tfServingRequest = TFServingInferenceServiceImpl.adaptModelInferRequest(request)

Stitch

.callFuture(

FutureConverters

.RichListenableFuture(inferenceServiceStub.predict(tfServingRequest)).toTwitter

.map { response =>

TFServingInferenceServiceImpl.adaptModelInferResponse(response)

}

)

}

}