package com.twitter.search.common.util.ml.prediction\_engine;

import java.util.Collections;

import java.util.Map;

import java.util.function.Supplier;

import com.google.common.base.Preconditions;

import com.twitter.ml.prediction.core.PredictionEngine;

import com.twitter.ml.prediction.core.PredictionEngineFactory;

import com.twitter.ml.prediction.core.PredictionEngineLoadingException;

import com.twitter.ml.vw.constant.SnapshotConstants;

import com.twitter.search.common.file.AbstractFile;

import com.twitter.search.common.util.ml.models\_manager.BaseModelsManager;

/\*\*

\* Loads PredictionEngine models from a model provider (config or fixed directory)

\* and keeps them in memory. Can also reload models periodically by querying the

\* same model provider source.

\*/

public class PredictionEngineModelsManager extends BaseModelsManager<PredictionEngine> {

PredictionEngineModelsManager(

Supplier<Map<String, AbstractFile>> activeModelsSupplier,

boolean shouldUnloadInactiveModels,

String statsPrefix) {

super(activeModelsSupplier, shouldUnloadInactiveModels, statsPrefix);

}

@Override

public PredictionEngine readModelFromDirectory(AbstractFile modelBaseDir)

throws PredictionEngineLoadingException {

// We need to add the 'hdfs://' prefix, otherwise PredictionEngine will treat it as a

// path in the local filesystem.

PredictionEngine predictionEngine = new PredictionEngineFactory()

.createFromSnapshot(

"hdfs://" + modelBaseDir.getPath(), SnapshotConstants.FIXED\_PATH);

predictionEngine.initialize();

return predictionEngine;

}

/\*\*

\* Creates an instance that loads the models specified in a configuration file.

\*

\* Note that if the configuration file changes and it doesn't include a model that was present

\* before, the model will be removed (i.e. it unloads models that are not active anymore).

\*/

public static PredictionEngineModelsManager createUsingConfigFile(

AbstractFile configFile, String statsPrefix) {

Preconditions.checkArgument(

configFile.canRead(), "Config file is not readable: %s", configFile.getPath());

return new PredictionEngineModelsManager(new ConfigSupplier(configFile), true, statsPrefix);

}

/\*\*

\* Creates a no-op instance. It can be used for tests or when the models are disabled.

\*/

public static PredictionEngineModelsManager createNoOp(String statsPrefix) {

return new PredictionEngineModelsManager(Collections::emptyMap, false, statsPrefix) {

@Override

public void run() { }

};

}

}