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

import java.io.IOException;

import java.util.Collections;

import java.util.Map;

import java.util.function.Supplier;

import com.google.common.base.Preconditions;

import com.twitter.ml.api.FeatureContext;

import com.twitter.mlv2.trees.predictor.CartTree;

import com.twitter.mlv2.trees.scorer.DecisionForestScorer;

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

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

/\*\*

\* Loads Decision Forest based models and keep them in memory. Can also be scheduled to reload

\* models periodically.

\*

\* Note: Each instance is tied to a single {@link FeatureContext} instance. So, to load models

\* for different tasks, you should use different instances of the this class.

\*/

public class DecisionForestModelsManager extends BaseModelsManager<DecisionForestScorer<CartTree>> {

private static final String MODEL\_FILE\_NAME = "model.json";

private final FeatureContext featureContext;

DecisionForestModelsManager(

Supplier<Map<String, AbstractFile>> activeModelsSupplier,

FeatureContext featureContext,

boolean shouldUnloadInactiveModels,

String statsPrefix

) {

super(activeModelsSupplier, shouldUnloadInactiveModels, statsPrefix);

this.featureContext = featureContext;

}

@Override

public DecisionForestScorer<CartTree> readModelFromDirectory(AbstractFile modelBaseDir)

throws IOException {

String modelFilePath = modelBaseDir.getChild(MODEL\_FILE\_NAME).getPath();

return DecisionForestScorer.createCartTreeScorer(modelFilePath, featureContext);

}

/\*\*

\* 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 DecisionForestModelsManager createUsingConfigFile(

AbstractFile configFile, FeatureContext featureContext, String statsPrefix) {

Preconditions.checkArgument(

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

return new DecisionForestModelsManager(

new ConfigSupplier(configFile), featureContext, true, statsPrefix);

}

/\*\*

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

\*/

public static DecisionForestModelsManager createNoOp(String statsPrefix) {

return new DecisionForestModelsManager(

Collections::emptyMap, new FeatureContext(), false, statsPrefix) {

@Override

public void run() { }

};

}

}