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

import java.util.Map;

import com.google.common.base.Preconditions;

import com.twitter.search.common.features.thrift.ThriftSearchResultFeatures;

import com.twitter.search.modeling.common.TweetFeaturesUtils;

/\*\*

\* Score accumulator for schema-based features.

\*/

public class SchemaBasedScoreAccumulator extends BaseScoreAccumulator<ThriftSearchResultFeatures> {

public SchemaBasedScoreAccumulator(LightweightLinearModel model) {

super(model);

Preconditions.checkState(model.isSchemaBased(),

"Cannot create SchemaBasedScoreAccumulator with a non-schema-based model: %s",

model.getName());

}

@Override

protected final void updateScoreWithFeatures(ThriftSearchResultFeatures featureData) {

// go through all features available and apply all those available in the model

addSchemaBooleanFeatures(featureData.getBoolValues());

addSchemaContinuousFeatures(featureData.getIntValues());

addSchemaContinuousFeatures(featureData.getLongValues());

addSchemaContinuousFeatures(featureData.getDoubleValues());

}

private void addSchemaBooleanFeatures(Map<Integer, Boolean> booleanMap) {

if (booleanMap == null || booleanMap.isEmpty()) {

return;

}

for (Map.Entry<Integer, Boolean> entry : booleanMap.entrySet()) {

if (entry.getValue()) {

score += model.binaryFeaturesById.getOrDefault(entry.getKey(), 0.0);

}

}

}

private void addSchemaContinuousFeatures(Map<Integer, ? extends Number> valueMap) {

if (valueMap == null || valueMap.isEmpty()) {

return;

}

for (Map.Entry<Integer, ? extends Number> entry : valueMap.entrySet()) {

Integer id = entry.getKey();

if (TweetFeaturesUtils.isFeatureDiscrete(id)) {

continue; // we don't process any discrete features now

}

Double weight = model.continuousFeaturesById.get(id);

if (weight != null) {

// found non-discretized entry

score += weight \* entry.getValue().doubleValue();

} else {

DiscretizedFeature discretizedFeature = model.discretizedFeaturesById.get(id);

if (discretizedFeature != null) {

// Use only the weight of the discretized feature (there's no need to multiply it)

score += discretizedFeature.getWeight(entry.getValue().doubleValue());

}

}

}

}

}