package com.twitter.product\_mixer.core.feature.featuremap.datarecord

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.DataRecordMerger

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.datarecord.\_

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

object DataRecordConverter {

val merger = new DataRecordMerger

}

/\*\*

\* Constructs a FeatureMap from a DataRecord, given a predefined set of features from a FeaturesScope.

\*

\* @param featuresScope scope of predefined set of BaseDataRecordFeatures that should be included in the output FeatureMap.

\*/

class DataRecordConverter[DRFeature <: BaseDataRecordFeature[\_, \_]](

featuresScope: FeaturesScope[DRFeature]) {

import DataRecordConverter.\_

def toDataRecord(featureMap: FeatureMap): DataRecord = {

// Initialize a DataRecord with the Feature Store features in it and then add all the

// non-Feature Store features that support DataRecords to DataRecord. We don't

// need to add Feature Store features because they're already in the initial DataRecord.

// If there are any pre-built DataRecords, we merge those in.

val richDataRecord = featuresScope.getFeatureStoreFeaturesDataRecord(featureMap)

val features = featuresScope.getNonFeatureStoreDataRecordFeatures(featureMap)

features.foreach {

case \_: FeatureStoreDataRecordFeature[\_, \_] =>

case requiredFeature: DataRecordFeature[\_, \_] with DataRecordCompatible[\_] =>

richDataRecord.setFeatureValue(

requiredFeature.mlFeature,

requiredFeature.toDataRecordFeatureValue(

featureMap.get(requiredFeature).asInstanceOf[requiredFeature.FeatureType]))

case optionalFeature: DataRecordOptionalFeature[\_, \_] with DataRecordCompatible[\_] =>

featureMap

.get(

optionalFeature.asInstanceOf[Feature[\_, Option[optionalFeature.FeatureType]]]).foreach {

value =>

richDataRecord

.setFeatureValue(

optionalFeature.mlFeature,

optionalFeature.toDataRecordFeatureValue(value))

}

case dataRecordInAFeature: DataRecordInAFeature[\_] =>

merger.merge(richDataRecord.getRecord, featureMap.get(dataRecordInAFeature))

}

richDataRecord.getRecord

}

}