package com.twitter.search.common.encoding.features;

import com.google.common.base.Preconditions;

/\*\*

\* A normalizer that normalizes the prediction score from a machine learning classifier, which

\* ranges within [0.0, 1.0], to an integer value by multiplying by (10 ^ precision), and returns

\* the rounded value. The lower the precision, the less amount of bits it takes to encode the score.

\* @see #precision

\*

\* This normalizer also could denormalize the normalized value from integer back to double using the

\* same precision.

\*/

public class PredictionScoreNormalizer {

private final int precision;

private final double normalizingBase;

public PredictionScoreNormalizer(int precision) {

this.precision = precision;

this.normalizingBase = Math.pow(10, this.precision);

}

/\*\*

\* Returns the normalized int value for prediction score {@code score} by multiplying

\* by {@code normalizingBase}, and round the result.

\* @throws IllegalArgumentException when parameter {@code score} is not within [0.0, 1.0]

\*/

public int normalize(double score) {

Preconditions.checkArgument(isScoreWithinRange(score));

return (int) Math.round(score \* this.normalizingBase);

}

/\*\*

\* Converts the normalized int value back to a double score by dividing by {@code normalizingBase}

\* @throws IllegalStateException when the denormalized value is not within [0.0, 1.0]

\*/

public double denormalize(int normalizedScore) {

double denormalizedValue = normalizedScore / this.normalizingBase;

if (!isScoreWithinRange(denormalizedValue)) {

throw new IllegalStateException(

String.format("The denormalized value %s is not within [0.0, 1.0]", denormalizedValue)

);

}

return denormalizedValue;

}

private static boolean isScoreWithinRange(double score) {

return 0.0 <= score && score <= 1.0;

}

}