**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Научный руководитель  доцент департамента  программной инженерии  факультета компьютерных наук,  канд. техн. наук  Родригес Залепинос Р.А.  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  2018 г. | УТВЕРЖДЕНО  Академический руководитель  образовательной программы  «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** В.В. Шилов  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  2017 г. |

|  |  |
| --- | --- |
| **Подп. и дата** |  |
| **Инв. № дубл.** |  |
| **Взам. инв. №** |  |
| **Подп. и дата** |  |
| **Инв. № подл** |  |

**ПРОГРАММА ОБНАРУЖЕНИЯ ИЗМЕНЕНИЙ ЗЕМЛЕПОЛЬЗОВАНИЯ ПО МУЛЬТИСЕНОСОРНЫМ СПУТНИКОВЫМ ДАННЫМ**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.16 12 01-1-ЛУ**

|  |  |
| --- | --- |
|  | Исполнитель:  студент группы БПИ152  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** /А.А. Лукин/  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  2018 г |
|  |  |
|  |  |

УТВЕРЖДЕН

RU.17701729.04.16 12 01-1-ЛУ

**ПРОГРАММА ОБНАРУЖЕНИЯ ИЗМЕНЕНИЙ ЗЕМЛЕПОЛЬЗОВАНИЯ ПО МУЛЬТИСЕНСОРНЫМ СПУТНИКОВЫМ ДАННЫМ**

|  |  |
| --- | --- |
| **Подп. и дата** |  |
| **Инв. № дубл.** |  |
| **Взам. инв. №** |  |
| **Подп. и дата** |  |
| **Инв. № подл** |  |

**Текст программы**

**RU.17701729.04.16 12 01-1**

**Листов 136**

# **Аннотация**

В данном программном документе приведено техническое задание для «Программы обнаружения изменений землепользования по мультисенсорным спутниковым данным». Данная программа предназначена для обнаружения изменений землепользования по спутниковым снимкам.

Оформление программного документа произведено по требованиям ГОСТ 19.401-78 «Текст программы. Требования к содержанию и оформлению» [1] .

**Содержание**

[**Аннотация** 2](#_Toc514059108)

[**Терминология** 4](#_Toc514059109)

[**1.** **Текст программы** 1](#_Toc514059110)

[1.1. app.js 1](#_Toc514059111)

[1.2. ChangeDetection 1](#_Toc514059112)

[1.3. Classification 13](#_Toc514059113)

[1.4. ClassificationEnum 35](#_Toc514059114)

[1.5. Controller 35](#_Toc514059115)

[1.6. LandUseChangeDetection 55](#_Toc514059116)

[1.7. LandUseChangeDetection 56](#_Toc514059117)

[1.8. Resolution 57](#_Toc514059118)

[1.9. SentinelData 57](#_Toc514059119)

[1.10. SentinelLevelUpdater 76](#_Toc514059120)

[1.11. Utils 78](#_Toc514059121)

[1.12. Data 85](#_Toc514059122)

[1.13. Setup.sql 89](#_Toc514059123)

[1.14. LevelUpForm 90](#_Toc514059124)

[1.15. ProgressForm 97](#_Toc514059125)

[1.16. SearchAndDownloadForm 99](#_Toc514059126)

[1.17. TrainNextGisForm 110](#_Toc514059127)

[**Список использованной литературы** 135](#_Toc514059128)

# **Терминология**

**Землепользование (Land Use) -**  управление и изменение окружающей природной среды или дикой природы в построенных условиях, таких как поселения и полу-естественные среды обитания, такие как пахотные поля , пастбища и управляемые леса.

**Мультисенсорные (мультиспектральные) снимки -** набормоноспектральных изображений одной и той же сцены, полученных одновременно, но в разных спектральных каналах. Поочередный синтез отдельных каналов позволяет решать многочисленные тематические задачи, а также помогает при дешифрировании снимков**.**

**Гранула Sentinel-2 –** мультисенсорное изображение Sentinel-2, покрывающее территорию 100х100 км2.

**Метод опорных векторов (Support Vector Machine, SVM) -** метод решения задачи классификации и регрессии путем построения нелинейной плоскости, разделяющей решения. Благодаря особенностям природы пространства признаков, в котором строятся границы решения, метод опорных векторов обладает высокой степенью гибкости при решении задач регрессии и классификации различного уровня сложности

# **Текст программы**

Программа состоит из.

## app.js

## ChangeDetection

package LandUseChangeDetection;

import LandUseChangeDetection.data.Data;

import com.vividsolutions.jts.geom.Geometry;

import com.vividsolutions.jts.geom.GeometryCollection;

import com.vividsolutions.jts.geom.MultiPolygon;

import com.vividsolutions.jts.geom.Polygon;

import com.vividsolutions.jts.simplify.TopologyPreservingSimplifier;

import org.geotools.coverage.grid.GridCoverage2D;

import org.geotools.coverage.grid.GridCoverageFactory;

import org.geotools.data.simple.SimpleFeatureCollection;

import org.geotools.data.simple.SimpleFeatureIterator;

import org.geotools.factory.CommonFactoryFinder;

import org.geotools.feature.DefaultFeatureCollection;

import org.geotools.feature.simple.SimpleFeatureBuilder;

import org.geotools.feature.simple.SimpleFeatureTypeBuilder;

import org.geotools.process.raster.PolygonExtractionProcess;

import org.geotools.referencing.crs.DefaultGeographicCRS;

import org.opengis.feature.simple.SimpleFeature;

import org.opengis.feature.simple.SimpleFeatureType;

import org.opengis.filter.Filter;

import org.opengis.filter.FilterFactory2;

import org.opengis.referencing.FactoryException;

import javax.media.jai.RasterFactory;

import java.awt.image.DataBuffer;

import java.awt.image.Raster;

import java.awt.image.WritableRaster;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.stream.IntStream;

class ChangeDetector {

/\*\*

\* Before Sentinel 2 data

\*/

private SentinelData beforeSentinelData;

/\*\*

\* After Sentinel 2 data

\*/

private SentinelData afterSentinelData;

private SimpleFeatureCollection beforeClassification;

private SimpleFeatureCollection afterClassification;

private SimpleFeatureCollection changeDetection;

SimpleFeatureCollection getChangeDetection() {

return this.changeDetection;

}

private List<LandUseChangeDetectionResult> areas;

List<LandUseChangeDetectionResult> getAreas() {

return this.areas;

}

ChangeDetector(SentinelData beforeSentinelData, SentinelData afterSentinelData) throws Exception {

if (beforeSentinelData.getSensingDate().before(afterSentinelData.getSensingDate())) {

this.beforeSentinelData = beforeSentinelData;

this.afterSentinelData = afterSentinelData;

} else {

this.beforeSentinelData = afterSentinelData;

this.afterSentinelData = beforeSentinelData;

}

cropScenes();

}

ChangeDetector(SentinelData beforeSentinelData, SentinelData afterSentinelData, SimpleFeatureCollection collection) throws Exception {

if (beforeSentinelData.getSensingDate().before(afterSentinelData.getSensingDate())) {

this.beforeSentinelData = beforeSentinelData;

this.afterSentinelData = afterSentinelData;

} else {

this.beforeSentinelData = afterSentinelData;

this.afterSentinelData = beforeSentinelData;

}

cropScenes();

cropByROI(collection);

}

private void cropScenes() throws Exception {

if (beforeSentinelData == null || afterSentinelData == null) {

throw new Exception("Error, data is null");

}

beforeSentinelData.cropBands(afterSentinelData.getEnvelope());

afterSentinelData.cropBands(beforeSentinelData.getEnvelope());

}

private Geometry roi;

private void cropByROI(SimpleFeatureCollection collection) throws Exception {

if (beforeSentinelData == null || afterSentinelData == null || collection == null) {

return;

}

if (beforeSentinelData.getCRS() != collection.getSchema().getCoordinateReferenceSystem()) {

collection = Utils.transformToCRS(collection, beforeSentinelData.getCRS());

}

// Get geom union

Geometry union = null;

SimpleFeatureIterator it = collection.features();

while (it.hasNext()) {

Geometry geometry = (Geometry) it.next().getDefaultGeometry();

if (geometry == null) {

continue;

}

if (union == null) {

union = geometry;

} else {

union = union.union(geometry);

}

}

this.roi = union;

beforeSentinelData.cropBands(union);

afterSentinelData.cropBands(union);

}

private Float[][] beforeClassificationMatrix;

private Float[][] afterClassificationMatrix;

void certificate() throws Exception {

if (beforeSentinelData.getHeight() != afterSentinelData.getHeight()

|| beforeSentinelData.getWidth() != afterSentinelData.getWidth()) {

throw new Exception("Error, classification bands with different sizes");

}

Classification beforeSVM = Classification.getInstance(beforeSentinelData.getType());

Classification afterSVM = Classification.getInstance(afterSentinelData.getType());

Raster beforeMask = beforeSentinelData.getCloudsAndSnowMask().getRenderedImage().getData();

Raster afterMask = afterSentinelData.getCloudsAndSnowMask().getRenderedImage().getData();

int[] beforeMaskPixels = new int[beforeMask.getWidth() \* beforeMask.getHeight()];

beforeMask.getPixels(beforeMask.getMinX(), beforeMask.getMinY(), beforeMask.getWidth(), beforeMask.getHeight(), beforeMaskPixels);

int[] afterMaskPixels = new int[afterMask.getWidth() \* afterMask.getHeight()];

afterMask.getPixels(afterMask.getMinX(), afterMask.getMinY(), afterMask.getWidth(), afterMask.getHeight(), afterMaskPixels);

beforeClassificationMatrix = new Float[beforeSentinelData.getWidth()][beforeSentinelData.getHeight()];

afterClassificationMatrix = new Float[afterSentinelData.getWidth()][afterSentinelData.getHeight()];

beforeSentinelData.getPixelVector(0);

afterSentinelData.getPixelVector(0);

int width = beforeSentinelData.getWidth();

int height = beforeSentinelData.getHeight();

IntStream.range(0, width).parallel().forEach(x ->

IntStream.range(0, height).parallel().forEach(y -> {

int i = y \* width + x;;

if (beforeMaskPixels[i] != 1 && beforeMaskPixels[i] != -9999 && beforeMaskPixels[i] != Float.NaN

&& afterMaskPixels[i] != 1 && afterMaskPixels[i] != -9999 && afterMaskPixels[i] != Float.NaN) {

double[] bPixels = beforeSentinelData.getPixelVector(i);

double[] aPixels = afterSentinelData.getPixelVector(i);

if (bPixels != null && aPixels != null) {

beforeClassificationMatrix[x][y] = new Float(beforeSVM.predict(bPixels));

afterClassificationMatrix[x][y] = new Float(afterSVM.predict(aPixels));

} else {

beforeClassificationMatrix[x][y] = Float.NaN;

afterClassificationMatrix[x][y] = Float.NaN;

}

} else {

beforeClassificationMatrix[x][y] = Float.NaN;

afterClassificationMatrix[x][y] = Float.NaN;

}

})

);

}

void checkAndFixPixels() {

// Check pixels

checkPixels(beforeClassificationMatrix);

checkPixels(afterClassificationMatrix);

}

void extractPolygons() throws IOException {

GridCoverageFactory factory = new GridCoverageFactory();

WritableRaster before = RasterFactory.createBandedRaster(DataBuffer.TYPE\_FLOAT,

beforeSentinelData.getWidth(), beforeSentinelData.getHeight(), 1, null);

for (int i = 0; i < beforeClassificationMatrix.length; ++i) {

Float[] row = beforeClassificationMatrix[i];

for (int j = 0; j < row.length; ++j) {

before.setSample(i, j, 0, row[j]);

}

}

WritableRaster after = RasterFactory.createBandedRaster(DataBuffer.TYPE\_FLOAT,

afterSentinelData.getWidth(), afterSentinelData.getHeight(), 1, null);

for (int i = 0; i < afterClassificationMatrix.length; ++i) {

Float[] row = afterClassificationMatrix[i];

for (int j = 0; j < row.length; ++j) {

after.setSample(i, j, 0, row[j]);

}

}

GridCoverage2D beforeClassesGrid = factory.create("Before classes", before, beforeSentinelData.getEnvelope());

GridCoverage2D afterClassesGrid = factory.create("After classes", after, afterSentinelData.getEnvelope());

// Raster to vector

final PolygonExtractionProcess process = new PolygonExtractionProcess();

this.beforeClassification = process.execute(beforeClassesGrid, 0, true,

this.roi, Collections.singletonList(Float.NaN), null, null);

// System.out.println("Polygon Extraction After");

this.afterClassification = process.execute(afterClassesGrid, 0, true,

this.roi, Collections.singletonList(Float.NaN), null, null);

}

void detectLandUseChanges() throws FactoryException {

this.changeDetection = getIntersections(this.beforeClassification, this.afterClassification);

this.changeDetection = Utils.transformChangeDetectionCollectionCRS(this.changeDetection, DefaultGeographicCRS.WGS84);

}

void calculateLUCDAreas() throws Exception {

this.areas = Data.getSquares(this.changeDetection);

}

/\*\*

\* Checking for pixel neighbors

\* @param pixels pixels matrix

\*/

private void checkPixels(Float[][] pixels) {

IntStream.range(0, pixels.length - 1).parallel().forEach(x ->

IntStream.range(0, pixels[x].length - 1).parallel().forEach(y -> {

Float val = pixels[x][y];

if (val != -1) {

List<Float> neighbors = new ArrayList<>();

for (int i = x - 1; i <= x + 1; ++i) {

for (int j = y - 1; j <= y + 1; ++j) {

if ((x == i && y == j)

|| i < 0

|| j < 0

|| i >= pixels.length

|| j >= pixels[0].length

|| pixels[i][j] == Float.NaN) {

continue;

}

neighbors.add(pixels[i][j]);

}

}

if (neighbors.size() > 0) {

float c = neighbors.get(0);

boolean flag = true;

for (float v : neighbors) {

if (v != c) {

flag = false;

break;

}

}

if (flag) {

pixels[x][y] = c;

}

}

}

})

);

}

public String getWKT() throws Exception {

if (this.changeDetection == null) {

throw new Exception("Error, calculate lucd before");

}

StringBuilder builder = new StringBuilder(this.changeDetection.getSchema().getCoordinateReferenceSystem().toWKT());

builder.append("\n\n");

SimpleFeatureIterator it = this.changeDetection.features();

while (it.hasNext()) {

SimpleFeature feature = it.next();

builder.append(beforeSentinelData.getSensingDate().toString());

builder.append(": ");

builder.append(getLandUseClass((int)feature.getAttribute("before")));

builder.append(" ");

builder.append(afterSentinelData.getSensingDate().toString());

builder.append(": ");

builder.append(getLandUseClass((int)feature.getAttribute("after")));

builder.append(" ");

builder.append(TopologyPreservingSimplifier.simplify((Geometry) feature.getDefaultGeometry(), 0).toString());

builder.append("\n");

}

return builder.toString();

}

/\*\*

\* Filter factory

\*/

private static final FilterFactory2 ff = CommonFactoryFinder.getFilterFactory2();

/\*\*

\* Get land-use collections changes

\* @param beforeCollection before sensed land-use collection

\* @param afterCollection after sensed land-use collection

\* @return land-use change collection

\*/

private static SimpleFeatureCollection getIntersections(SimpleFeatureCollection beforeCollection, SimpleFeatureCollection afterCollection) {

SimpleFeatureTypeBuilder typeBuilder = new SimpleFeatureTypeBuilder();

typeBuilder.setName("LUCD");

typeBuilder.setCRS(beforeCollection.getSchema().getCoordinateReferenceSystem());

typeBuilder.add("the\_geom", MultiPolygon.class);

typeBuilder.add("before", Integer.class);

typeBuilder.add("after", Integer.class);

final SimpleFeatureType featureType = typeBuilder.buildFeatureType();

DefaultFeatureCollection collection = new DefaultFeatureCollection(null, null);

SimpleFeatureBuilder featureBuilder = new SimpleFeatureBuilder(featureType);

String geomPropertyName = afterCollection.getSchema().getGeometryDescriptor().getLocalName();

SimpleFeatureIterator it = beforeCollection.features();

while (it.hasNext()) {

SimpleFeature feature = it.next();

Geometry geometry = (Geometry) feature.getDefaultGeometry();

Filter filter = ff.intersects(ff.property(geomPropertyName), ff.literal(geometry));

SimpleFeatureCollection interCollection = afterCollection.subCollection(filter);

if (interCollection != null) {

SimpleFeatureIterator sfi = interCollection.features();

while (sfi.hasNext()) {

SimpleFeature afterFeature = sfi.next();

Geometry afterGeometry = (Geometry) afterFeature.getDefaultGeometry();

if (afterGeometry instanceof Polygon || afterCollection instanceof MultiPolygon) {

Geometry intersection = geometry.intersection(afterGeometry);

if (!intersection.isEmpty()) {

intersection.setUserData(geometry);

Object beforeClass = feature.getAttribute("value");

Object afterClass = afterFeature.getAttribute("value");

if (intersection instanceof Polygon || intersection instanceof MultiPolygon) {

featureBuilder.add(intersection);

featureBuilder.add(beforeClass);

featureBuilder.add(afterClass);

collection.add(featureBuilder.buildFeature(null));

} else if (intersection instanceof GeometryCollection) {

GeometryCollection geometryCollection = (GeometryCollection)intersection;

for (int i = 0; i < geometryCollection.getNumGeometries(); ++i) {

Geometry geom = geometryCollection.getGeometryN(i);

if (!geom.isEmpty() && (geom instanceof Polygon || geom instanceof MultiPolygon)) {

featureBuilder.add(geom);

featureBuilder.add(beforeClass);

featureBuilder.add(afterClass);

collection.add(featureBuilder.buildFeature(null));

}

}

}

}

}

}

}

}

return collection;

}

private static String getLandUseClass(int c) throws Exception {

if (c == 0) return "water";

if (c == 1) return "agriculture";

if (c == 2) return "urban";

if (c == 3) return "forest";

throw new Exception("Error, unknown land-land use class");

}

}

## Classification

package LandUseChangeDetection;

import com.vividsolutions.jts.geom.Geometry;

import com.vividsolutions.jts.geom.MultiPolygon;

import org.geotools.coverage.grid.GridCoverage2D;

import org.geotools.coverage.grid.GridCoverageFactory;

import org.geotools.data.DataStore;

import org.geotools.data.simple.SimpleFeatureCollection;

import org.geotools.data.simple.SimpleFeatureIterator;

import org.geotools.data.simple.SimpleFeatureSource;

import org.geotools.feature.DefaultFeatureCollection;

import org.geotools.feature.simple.SimpleFeatureBuilder;

import org.geotools.feature.simple.SimpleFeatureTypeBuilder;

import org.geotools.filter.ConstantExpression;

import org.geotools.geometry.jts.JTS;

import org.geotools.geometry.jts.ReferencedEnvelope;

import org.geotools.process.vector.VectorToRasterProcess;

import org.geotools.referencing.CRS;

import org.opengis.feature.simple.SimpleFeature;

import org.opengis.feature.simple.SimpleFeatureType;

import org.opengis.referencing.FactoryException;

import org.opengis.referencing.crs.CoordinateReferenceSystem;

import org.opengis.referencing.operation.MathTransform;

import org.opengis.referencing.operation.TransformException;

import smile.classification.SVM;

import smile.math.kernel.GaussianKernel;

import javax.media.jai.JAI;

import javax.media.jai.RenderedOp;

import java.awt.image.Raster;

import java.awt.image.renderable.ParameterBlock;

import java.io.\*;

import java.util.\*;

public class Classification implements Serializable {

/\*\*

\* Classification singleton instance

\*/

private static Classification instance;

/\*\*

\* Classification B singleton instance

\*/

private static Classification instanceB;

/\*\*

\* Path ot serializable object

\*/

public static final File svmModelPath = new File("src/resources/model.svm");

/\*\*

\* Path ot serializable object

\*/

public static final File svmBModelPath = new File("src/resources/modelB.svm");

/\*\*

\* SVM A model

\*/

private SVM<double[]> svm;

/\*\*

\* Classes

\*/

private static final List<List<String>> LAND\_USE\_CLASSES = Collections.unmodifiableList(Arrays.asList(

// Water class

Collections.unmodifiableList(Arrays.asList(

"water" // natural

)),

// Agriculture

Collections.unmodifiableList(Arrays.asList(

"farmland",

"meadow",

"orchard",

"plant\_nursery",

"vineyard",

"farm",

"allotments",

"farmyard",

"greenhouse\_horticulture"

)),

// Urban classes

// Build up

Collections.unmodifiableList(Arrays.asList(

"commercial",

"garages",

"industrial",

"religious",

"residential",

"retail",

"school",

"brownfield",

"construction",

"landfill",

"quarry",

"salt\_pond"

)),

// Forest class

Collections.unmodifiableList(Arrays.asList(

"forest",

"wood"

))

));

/\*\*

\* Private constructor

\* @param svm model

\*/

private Classification(SVM<double[]> svm) {

this.svm = svm;

}

/\*\*

\* SVM classification singleton

\* @return Classification instance

\*/

public static Classification getInstance() {

if (instance == null) {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(svmModelPath))) {

SVM<double[]>svm = (SVM<double[]>) ois.readObject();

instance = new Classification(svm);

} catch (ClassNotFoundException | IOException e) {

instance = new Classification(null);

}

}

return instance;

}

public static Classification getInstance(ClassificationEnum type) {

if (type == ClassificationEnum.A) {

return getInstance();

}

if (instanceB == null) {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(svmBModelPath))) {

SVM<double[]>svm = (SVM<double[]>) ois.readObject();

instanceB = new Classification(svm);

} catch (ClassNotFoundException | IOException e) {

instanceB = new Classification(null);

}

}

return instanceB;

}

// /\*\*

// \* Serialize trained SVM model

// \*/

// private void serializeSVMObject() throws IOException {

// try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(svmModelPath))) {

// oos.writeObject(this.svm);

// }

// }

/\*\*

\* Serialize trained SVM model

\*/

private void serializeSVMObject(ClassificationEnum type) throws IOException {

if (type == ClassificationEnum.A) {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(svmModelPath))) {

oos.writeObject(this.svm);

}

return;

}

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(svmBModelPath))) {

oos.writeObject(this.svm);

}

}

int predict(double[] vector) {

return this.svm.predict(vector);

}

public int[] predict(double[][] vectors) {

return this.svm.predict(vectors);

}

private void trainAndValidateModel(SVMData svmData) throws IOException {

// TODO: Grid search values range

SVM<double[]> selectedSVM = null;

double selectedS = 0;

double selectedC = 0;

double totalAccuracy = 0;

double[] classAccuracies = new double[LAND\_USE\_CLASSES.size()];

SVMSet[] sets = svmData.getCrossValidationData();

// Grid search and

for (double s = 0.5; s <= Math.pow(2, 13); s \*= 2) {

for (double c = 0.5; c <= Math.pow(2, 13); c \*= 2) {

double accuracy = 0;

double maxAccuracy = 0;

double[] accuracies = new double[LAND\_USE\_CLASSES.size()];

double[] maxAccuracies = new double[LAND\_USE\_CLASSES.size()];

SVM<double[]> currentSVM = null;

// Cross validation

for (int i = 0; i < sets.length; ++i) {

SVM<double[]> svm = new SVM<>(new GaussianKernel(s), c, LAND\_USE\_CLASSES.size(), SVM.Multiclass.ONE\_VS\_ALL);

for (int j = 0; j < sets.length; ++j) {

if (j != i) {

svm.learn(sets[j].vectors, sets[j].labels);

}

}

svm.finish();

int[] predictions = svm.predict(sets[i].vectors);

int[] counts = new int[LAND\_USE\_CLASSES.size()];

int[] sizes = new int[LAND\_USE\_CLASSES.size()];

for (int j = 0; j < predictions.length; ++j) {

if (sets[i].labels[j] == predictions[j]) {

++counts[predictions[j]];

}

++sizes[sets[i].labels[j]];

}

int count = 0;

for (int val : counts) {

count += val;

}

double currentAccuracy = (double) count / predictions.length;

accuracy += currentAccuracy;

double[] currentAccuracies = new double[LAND\_USE\_CLASSES.size()];

for (int j = 0; j < accuracies.length; ++j) {

currentAccuracies[j] = (double)counts[j] / sizes[j];

accuracies[j] += currentAccuracies[j];

}

// System.out.println((double)wCount / predictions.length);

if (currentAccuracy > maxAccuracy) {

maxAccuracy = currentAccuracy;

maxAccuracies = currentAccuracies;

currentSVM = svm;

}

}

accuracy = accuracy / sets.length;

for (int i = 0; i < accuracies.length; ++i) {

accuracies[i] /= sets.length;

}

System.out.println("S = " + s + "; C = " + c + "; accuracy = " + accuracy + " " + Arrays.toString(accuracies));

if (accuracy > totalAccuracy) {

totalAccuracy = accuracy;

classAccuracies = accuracies;

selectedSVM = currentSVM;

selectedC = c;

selectedS = s;

}

System.out.println("RES: S = " + selectedS + "; C" + selectedC + "; ac = " + totalAccuracy + " " + Arrays.toString(classAccuracies) + "\n");

System.out.println();

}

}

System.out.println("c = " + selectedC + "; s" + selectedS + "; ac = " + totalAccuracy + " c: " + Arrays.toString(classAccuracies));

// Online training

sets = svmData.getCrossValidationData();

for (int i = 1; i < sets.length; ++i) {

selectedSVM.learn(sets[i].vectors, sets[i].labels);

}

selectedSVM.finish();

int[] predictions = selectedSVM.predict(sets[0].vectors);

int[] counts = new int[LAND\_USE\_CLASSES.size()];

int[] sizes = new int[LAND\_USE\_CLASSES.size()];

for (int j = 0; j < predictions.length; ++j) {

if (sets[0].labels[j] == predictions[j]) {

++counts[predictions[j]];

}

++sizes[sets[0].labels[j]];

}

int count = 0;

for (int val : counts) {

count += val;

}

double currentAccuracy = (double) count / predictions.length;

System.out.println(currentAccuracy);

this.svm = selectedSVM;

System.out.println();

}

private void learn(double[][] data, int[] label) {

svm.learn(data, label);

svm.finish();

}

// /\*\*

// \* Train svm model by NextGIS shapefiles

// \* @param nextShp

// \* @param s2DataFile

// \* @throws Exception

// \*/

// public void trainByNextGISData(File nextShp, File s2DataFile) throws Exception {

// SentinelData sData = new SentinelData(s2DataFile, Resolution.R60m);

// GridCoverage2D mask = getNextGISCoverage(nextShp, sData);

// SVMData svmData = getTrainingAndValidationData(sData, mask);

// sData = null;

// trainAndValidateModel(svmData);

// svmData = null;

// serializeSVMObject();

// }

public void trainByNextGISData(File nextShp, File s2DataFile, ClassificationEnum type, Resolution resolution) throws Exception {

SentinelData sData = new SentinelData(s2DataFile, resolution, type);

GridCoverage2D mask = getNextGISCoverage(nextShp, sData);

SVMData svmData = getTrainingAndValidationData(sData, mask);

sData = null;

trainAndValidateModel(svmData);

svmData = null;

serializeSVMObject(type);

}

// private void trainByNextGISData(SentinelData sData, File nextSHP) throws Exception {

// GridCoverage2D mask = getNextGISCoverage(nextSHP, sData);

// SVMData svmData = getTrainingAndValidationData(sData, mask);

// sData = null;

// trainAndValidateModel(svmData);

// svmData = null;

// }

/\*\*

\* NextGIS shapefile with water features

\*/

private final static String WATER\_POLYGON\_SHP\_NAME = "water-polygon.shp";

/\*\*

\* NextGIS shapefile with lu (agriculture)

\*/

private final static String LAND\_USE\_POLYGON\_SHP\_NAME = "landuse-polygon.shp";

/\*\*

\* NextGIS shapefile with vegetation (forest, wood)

\*/

private final static String VEGETATION\_POLYGON\_SHP\_NAME = "vegetation-polygon.shp";

/\*\*

\* Resterization of NextGIS data for classes

\* @param nextShp NextGIS shapefiles directory

\* @param sentinelData Sentinel 2 data

\* @return Array of masks by groups

\* @throws IOException Cannot open files

\* @throws FactoryException Cannot create feature builder

\* @throws TransformException Cannot change CRS

\*/

private GridCoverage2D getNextGISCoverage(File nextShp, SentinelData sentinelData) throws Exception {

// Checking for directory

if (!nextShp.isDirectory()) {

throw new FileNotFoundException("Error, NextSHP file is not directory");

}

// Water class (water-polygon.shp)

File waterShpFile = new File(nextShp.getAbsolutePath() + File.separator + WATER\_POLYGON\_SHP\_NAME);

if (!waterShpFile.exists()) {

throw new FileNotFoundException("Error, NextGIS doesn't contain water file");

}

DataStore waterDataStore = Utils.openShapefile(waterShpFile);

if (waterDataStore == null || waterDataStore.getTypeNames() == null || waterDataStore.getTypeNames().length == 0){

throw new NullPointerException("Water vector data store is null");

}

String waterTypeName = waterDataStore.getTypeNames()[0];

SimpleFeatureSource waterFeatureSource = waterDataStore.getFeatureSource(waterTypeName);

SimpleFeatureCollection waterFC = waterFeatureSource.getFeatures();

waterFC = Utils.transformToCRS(waterFC, sentinelData.getCRS());

// Land use shapefile

File landUseFile = new File(nextShp.getAbsolutePath() + File.separator + LAND\_USE\_POLYGON\_SHP\_NAME);

if (!landUseFile.exists()) {

throw new FileNotFoundException("Error, NextGIS doesn't contain land-use file");

}

DataStore landUseDataStore = Utils.openShapefile(landUseFile);

if (landUseDataStore == null || landUseDataStore.getTypeNames() == null ||landUseDataStore.getTypeNames().length == 0) {

throw new NullPointerException("Land-Use vector store is null");

}

String landUseTypeName = landUseDataStore.getTypeNames()[0];

SimpleFeatureSource landUseFeatureSource = landUseDataStore.getFeatureSource(landUseTypeName);

SimpleFeatureCollection landUseFC = landUseFeatureSource.getFeatures();

// Extract tags from land use shapefile

DefaultFeatureCollection[] featureCollections = extractClassesFeatures(null, landUseFC, "LANDUSE", sentinelData.getCRS());

// Vegetation shape file

File vegetationFile = new File(nextShp.getAbsolutePath() + File.separator + VEGETATION\_POLYGON\_SHP\_NAME);

if (!landUseFile.exists()) {

throw new FileNotFoundException("Error, NextGIS doesn't contain vegetation file");

}

DataStore vegetationDataStore = Utils.openShapefile(vegetationFile);

if (vegetationDataStore == null || vegetationDataStore.getTypeNames() == null || vegetationDataStore.getTypeNames().length == 0) {

throw new NullPointerException("Vegetation vector store is null");

}

String vegetationTypeName = vegetationDataStore.getTypeNames()[0];

SimpleFeatureSource vegetationFeatureSource = vegetationDataStore.getFeatureSource(vegetationTypeName);

SimpleFeatureCollection vegetationFC = vegetationFeatureSource.getFeatures();

// Convert

featureCollections[1] = extractClassesFeatures(featureCollections, vegetationFC, "NATURAL", sentinelData.getCRS())[1];

// Convert result to simple feature collection array

SimpleFeatureCollection[] simpleFeatureCollections = new SimpleFeatureCollection[featureCollections.length];

System.arraycopy(featureCollections, 0, simpleFeatureCollections, 0, featureCollections.length);

// Resterization

GridCoverage2D[] masks = new GridCoverage2D[simpleFeatureCollections.length];

// Water mask

GridCoverage2D waterCoverage = VectorToRasterProcess.process(waterFC, ConstantExpression.constant(1),

sentinelData.getGridDimension(), sentinelData.getEnvelope(), "waterMask", null);

masks[0] = waterCoverage;

for (int i = 1; i < masks.length; i++) {

masks[i] = VectorToRasterProcess.process(simpleFeatureCollections[i], ConstantExpression.constant((int)Math.pow(2, i)),

sentinelData.getGridDimension(), sentinelData.getEnvelope(), String.valueOf(i), null);

}

// Merge masks

GridCoverage2D last = masks[0];

for (int i = 1; i < masks.length; ++i) {

ParameterBlock maskOp = new ParameterBlock();

maskOp.addSource(last.getRenderedImage());

maskOp.addSource(masks[i].getRenderedImage());

RenderedOp op = JAI.create("add", maskOp);

GridCoverageFactory factory = new GridCoverageFactory();

ReferencedEnvelope envelope = new ReferencedEnvelope(masks[0].getEnvelope());

last = factory.create("ClassesMask", op, envelope);

}

return last;

}

/\*\*

\* Feature type builder

\*/

private SimpleFeatureTypeBuilder typeBuilder = null;

/\*\*

\* Feature builder

\*/

private SimpleFeatureBuilder featureBuilder = null;

/\*\*

\* Extract features from collection according croup tags

\* @param featureCollections Collections for updating

\* @param fc Collection with new features

\* @param attributeName Attribute of feature for looking

\* @param crs Purposed coordinate reference system

\* @return Updated feature collections

\* @throws FactoryException Cannot create builder

\* @throws TransformException Cannot transform features

\*/

private DefaultFeatureCollection[] extractClassesFeatures(DefaultFeatureCollection[] featureCollections,

SimpleFeatureCollection fc, String attributeName,

CoordinateReferenceSystem crs) throws FactoryException, TransformException {

// Type builder

if (this.typeBuilder == null) {

typeBuilder = new SimpleFeatureTypeBuilder();

typeBuilder.setName("Training");

typeBuilder.setCRS(crs);

typeBuilder.add("geom", MultiPolygon.class);

SimpleFeatureType featureType = typeBuilder.buildFeatureType();

featureBuilder = new SimpleFeatureBuilder(featureType);

}

// Init collection array

if (featureCollections == null) {

featureCollections = new DefaultFeatureCollection[LAND\_USE\_CLASSES.size()];

for (int i = 0; i < featureCollections.length; i++) {

featureCollections[i] = new DefaultFeatureCollection(null, null);

}

}

// CRS transformer

MathTransform transform = null;

if (!CRS.equalsIgnoreMetadata(fc.getSchema().getCoordinateReferenceSystem(), crs)) {

transform = CRS.findMathTransform(fc.getSchema().getCoordinateReferenceSystem(), crs, true);

}

// Extract features

try (SimpleFeatureIterator it = fc.features()) {

while (it.hasNext()) {

SimpleFeature feature = it.next();

String featureTag = (String) (feature.getAttribute(attributeName));

for (int i = 0; i < LAND\_USE\_CLASSES.size(); i++) {

if (LAND\_USE\_CLASSES.get(i).contains(featureTag)) {

Geometry geometry = (Geometry) feature.getDefaultGeometry();

if (transform != null) {

geometry = JTS.transform(geometry, transform);

}

featureBuilder.add(geometry);

SimpleFeature tagFeature = featureBuilder.buildFeature(null);

featureCollections[i].add(tagFeature);

break;

}

}

}

}

return featureCollections;

}

/\*\*

\* Random generator

\*/

private static Random random = new Random();

/\*\*

\* Extract training data and divide to training and validation data

\* @param sentinelData Sentinel 2 Data

\* @param mask Classes masks

\* @return Extracted and divided data

\*/

private SVMData getTrainingAndValidationData(SentinelData sentinelData, GridCoverage2D mask) throws Exception {

SVMData svmData = new SVMData();

int height = sentinelData.getHeight();

int width = sentinelData.getWidth();

float[] maskPixels = new float[height \* width];

float[] availablePixels = new float[height \* width];

// Get cloud and snow mask

Raster cloudsAndSnowRaster = sentinelData.getCloudsAndSnowMask().getRenderedImage().getData();

cloudsAndSnowRaster.getPixels(cloudsAndSnowRaster.getMinX(), cloudsAndSnowRaster.getMinY(), width, height, availablePixels);

// Get classes mask

Raster maskRaster = mask.getRenderedImage().getData();

maskRaster.getPixels(maskRaster.getMinX(), maskRaster.getMinY(), width, height, maskPixels);

for (int i = 0; i < maskPixels.length; ++i) {

int c = (int)maskPixels[i];

if (availablePixels[i] == 0.0F && c > 0 && c <= Math.pow(2, LAND\_USE\_CLASSES.size())) {

double[] vector = sentinelData.getPixelVector(i);

if (vector == null) continue;

for (int j = 0; j < LAND\_USE\_CLASSES.size(); ++j) {

if (Math.pow(2, j) == c) {

svmData.add(new SVMVector(vector, j));

break;

}

}

}

}

return svmData;

}

private class SVMVector{

double[] vector;

int label;

SVMVector(double[] vector, int label) {

this.vector = vector;

this.label = label;

}

}

/\*\*

\* SVM training or validation set

\*/

private class SVMSet{

double[][] vectors;

int[] labels;

SVMSet(double[][] vectors, int[] labels) {

this.vectors = vectors;

this.labels = labels;

}

}

/\*\*

\* SVM training and validation data

\*/

private class SVMData{

/\*\*

\* List of data

\*/

private List<SVMVector> data;

/\*\*

\* SVM data initializer

\*/

SVMData() {

data = new LinkedList<>();

}

/\*\*

\* Add vector and class label

\* @param data data label

\*/

void add(SVMVector data) {

this.data.add(data);

}

/\*\*

\* Get training and validation data in relation 20 : 80

\* @return Training and validation sets

\*/

SVMSet[] getCrossValidationData() {

SVMSet[] sets = new SVMSet[5];

for (int i = 0; i < sets.length; ++i) {

int size;

if (this.data.size() < 1000) {

size = this.data.size();

} else {

size = 1000;

}

double[][] tempVectors = new double[size][];

int[] tempLabels = new int[size];

for (int j = 0; j < size; ++j) {

SVMVector data = this.data.remove(random.nextInt(this.data.size()));

tempVectors[j] = data.vector;

tempLabels[j] = data.label;

}

sets[i] = new SVMSet(tempVectors, tempLabels);

}

return sets;

}

}

// static File result = new File("C:\\Users\\lukin\\Desktop\\res.txt");

// static BufferedWriter writer;

//

//// static {

//// try {

//// writer = new BufferedWriter(new FileWriter(result));

//// } catch (IOException e) {

//// e.printStackTrace();

//// }

//// }

//

// static void findBestBandsSet(File nextShp, File s2DataFile) throws Exception {

// SentinelData data = new SentinelData(s2DataFile, Resolution.R60m);

// List<GridCoverage2D> bands = data.getBands();

// List<List<Integer>> sets = new ArrayList<>();

// for (int i = 0; i < (1 << bands.size()); ++i) {

// sets.add(new ArrayList<>());

// for (int j = 0; j < bands.size(); ++j) {

// if ((i & (1 << j)) != 0) {

// sets.get(i).add(j);

// }

// }

// }

// sets = Lists.reverse(sets);

// for (List<Integer> list : sets) {

// if (list.size() < 10) {

// continue;

// }

// List<GridCoverage2D> currentBands = new ArrayList<>();

// for (Integer i : list) {

// currentBands.add(bands.get(i));

// }

// data.setBands(currentBands);

// writer.write(Arrays.toString(list.toArray()));

// writer.newLine();

// writer.flush();

// Classification classification = new Classification(null);

// classification.trainByNextGISData(data, nextShp);

// }

// writer.close();

// }

}

## ClassificationEnum

package LandUseChangeDetection;

public enum ClassificationEnum {

A,

B

}

## Controller

package LandUseChangeDetection;

import LandUseChangeDetection.forms.ProgressForm;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.concurrent.Task;

import javafx.concurrent.Worker;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.control.TextArea;

import javafx.scene.image.Image;

import javafx.scene.layout.BorderPane;

import javafx.scene.web.WebEngine;

import javafx.scene.web.WebView;

import javafx.stage.FileChooser;

import javafx.stage.Modality;

import javafx.stage.Stage;

import jdk.nashorn.api.scripting.JSObject;

import org.apache.commons.io.FileUtils;

import org.esa.s2tbx.dataio.VirtualPath;

import org.geotools.data.DataStore;

import org.geotools.data.collection.ListFeatureCollection;

import org.hsqldb.lib.FileUtil;

import java.io.\*;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Arrays;

import java.util.List;

import java.util.Scanner;

import java.util.stream.Collectors;

import java.util.stream.Stream;

public class Controller {

/\*\*

\* Application form

\*/

public BorderPane appForm;

public ComboBox beforeGranulesComboBox;

public ComboBox afterGranulesComboBox;

public WebView webMap;

/\*\*

\* Data resolutions

\*/

private final ObservableList<String> resolutions = FXCollections.observableArrayList( "60m", "20m");

public ComboBox resolutionBox;

public Label waterLabel;

public Label buildLevel;

public Label forestLabel;

public Label waLabel;

public Label wbLabel;

public Label wfLabel;

public Label awLabel;

public Label abLabel;

public Label afLabel;

public Label bwLabel;

public Label baLabel;

public Label bfLabel;

public Label faLabel;

public Label fwLabel;

public Label fbLabel;

public Label agriLabel;

public Label roiLabel;

public TextArea wktTextArea;

WebEngine webEngine;

@FXML

void initialize(){

resolutionBox.setItems(resolutions);

resolutionBox.setValue("60m");

this.webEngine = webMap.getEngine();

File mapIndexFile = new File("src/resources/AppWebForm/index.html");

webEngine.load("file:" + mapIndexFile.getAbsolutePath());

webEngine.setJavaScriptEnabled(true);

}

/\*\*

\* Show level up form

\* @param actionEvent level up form opening action event

\*/

public void getLevelUpFormHandler(ActionEvent actionEvent) {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/levelUpForm.fxml"));

try {

Parent root = loader.load();

Stage stage = new Stage();

stage.initModality(Modality.APPLICATION\_MODAL);

stage.setTitle("Level Up Sentinel 2 Data");

stage.setScene(new Scene(root, 600, 270));

stage.setResizable(false);

stage.getIcons().add(new Image("/icon.png"));

stage.showAndWait();

} catch (IOException e) {

e.printStackTrace();

}

}

public void openDownloadingForm(ActionEvent actionEvent) {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/searchAndDownloadForm.fxml"));

try {

Parent root = loader.load();

Stage stage = new Stage();

stage.initModality(Modality.APPLICATION\_MODAL);

stage.setTitle("Search and Download Sentinel 2 Data");

stage.setMinWidth(800);

stage.setMinHeight(600);

stage.setScene(new Scene(root, 800, 600));

stage.getIcons().add(new Image("/icon.png"));

stage.showAndWait();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* Open learn form based on OSM data

\* @param actionEvent OSM learn form opening action event

\*/

public void openLearnOSMForm(ActionEvent actionEvent) {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/learnNextGISForm.fxml"));

try {

Parent root = loader.load();

Stage stage = new Stage();

stage.initModality(Modality.APPLICATION\_MODAL);

stage.setTitle("Learn SVM by Open Street Map");

stage.setScene(new Scene(root, 600, 400));

stage.getIcons().add(new Image("/icon.png"));

stage.setResizable(false);

stage.showAndWait();

} catch (IOException e) {

e.printStackTrace();

}

}

private File beforeSentinelData;

public void selectBeforeData(ActionEvent actionEvent) {

FileChooser fc = new FileChooser();

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("XML files (\*.xml)", "\*.xml");

fc.setSelectedExtensionFilter(filter);

this.beforeSentinelData = fc.showOpenDialog(appForm.getScene().getWindow());

if (this.beforeSentinelData == null) {

Utils.showErrorMessage("Before Sentinel 2 file is not selected",

"Before Sentinel 2 file is not selected",

"Please, select Sentinel 2 data file");

} else {

try {

List<VirtualPath> granules = SentinelData.checkAndGetGranules(this.beforeSentinelData);

ObservableList<String> granulesList = FXCollections.observableArrayList(

granules.stream().map(VirtualPath::getFullPathString)

.collect(Collectors.toList()));

beforeGranulesComboBox.setItems(granulesList);

beforeGranulesComboBox.setValue(granulesList.get(0));

} catch (Exception e) {

Utils.showErrorMessage("Error",

e.getMessage(),

Arrays.toString(e.getStackTrace()));

}

}

}

private File afterSentinelData;

public void selectAfterData(ActionEvent actionEvent) {

FileChooser fc = new FileChooser();

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("XML files (\*.xml)", "\*.xml");

fc.setSelectedExtensionFilter(filter);

this.afterSentinelData = fc.showOpenDialog(appForm.getScene().getWindow());

if (this.afterSentinelData == null) {

Utils.showErrorMessage("After Sentinel 2 file is not selected",

"After Sentinel 2 file is not selected",

"Please, select Sentinel 2 data file");

} else {

try {

List<VirtualPath> granules = SentinelData.checkAndGetGranules(this.afterSentinelData);

ObservableList<String> granulesList = FXCollections.observableArrayList(

granules.stream().map(VirtualPath::getFullPathString)

.collect(Collectors.toList()));

afterGranulesComboBox.setItems(granulesList);

afterGranulesComboBox.setValue(granulesList.get(0));

} catch (Exception e) {

Utils.showErrorMessage("Error",

e.getMessage(),

Arrays.toString(e.getStackTrace()));

}

}

}

private ChangeDetector lucd;

public void detectChanges(ActionEvent actionEvent) {

if (beforeSentinelData == null || afterSentinelData == null) {

return;

}

try {

File beforeSentinelGranuleFile = new File(beforeGranulesComboBox.getValue().toString());

File afterSentinelGranuleFile = new File(afterGranulesComboBox.getValue().toString());

final Resolution resolution = resolutionBox.getValue().equals("60m") ? Resolution.R60m : Resolution.R20m;

ProgressForm form = new ProgressForm();

Task<ChangeDetector> task = new Task<ChangeDetector>() {

@Override

protected ChangeDetector call() throws Exception {

updateProgress(0, 1);

updateMessage("Opening and parsing of " + beforeSentinelGranuleFile.getName());

SentinelData firstSentinelData = new SentinelData(beforeSentinelGranuleFile, resolution, SentinelData.getType(beforeSentinelData));

updateProgress(0.1, 1);

updateMessage("Opening and parsing of " + afterSentinelGranuleFile.getName());

SentinelData secondSentinelData = new SentinelData(afterSentinelGranuleFile, resolution, SentinelData.getType(afterSentinelData));

updateProgress(0.2, 1);

updateMessage("Bands cropping...");

ChangeDetector detector;

if (roiFile == null) {

detector = new ChangeDetector(firstSentinelData, secondSentinelData);

} else {

DataStore store = Utils.openShapefile(roiFile);

if (store == null || store.getTypeNames() == null || store.getTypeNames().length == 0) {

throw new NullPointerException("ROI vector data store is null");

}

String waterTypeName = store.getTypeNames()[0];

detector = new ChangeDetector(firstSentinelData, secondSentinelData,

Utils.openShapefile(roiFile).getFeatureSource(waterTypeName).getFeatures());

}

updateProgress(0.3, 1);

updateMessage("Bands classification and clouds and snow removing...");

detector.certificate();

updateProgress(0.5, 1);

updateMessage("Land-use classification results checking and fixing...");

detector.checkAndFixPixels();

updateProgress(0.6, 1);

updateMessage("Land-use classification vector extraction...");

detector.extractPolygons();

updateProgress(0.7, 1);

updateMessage("Land-use change detection...");

detector.detectLandUseChanges();

updateProgress(0.8, 1);

updateMessage("Land-use change areas calculation...");

detector.calculateLUCDAreas();

updateProgress(0.9, 1);

return detector;

}

};

form.activateProgressBar(task);

task.setOnSucceeded(event -> {

try {

this.lucd = task.getValue();

List<LandUseChangeDetectionResult> areas = lucd.getAreas();

for (LandUseChangeDetectionResult area : areas) {

switch (area.getBefore()) {

case 0: {

switch (area.getAfter()) {

case 0: {

waterLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 1: {

waLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 2: {

wbLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 3: {

wfLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

}

break;

}

case 1: {

switch (area.getAfter()) {

case 0: {

awLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 1: {

agriLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 2: {

abLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 3: {

afLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

}

break;

}

case 2: {

switch (area.getAfter()) {

case 0: {

bwLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 1: {

baLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 2: {

buildLevel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 3: {

bfLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

}

break;

}

case 3: {

switch (area.getAfter()) {

case 0: {

fwLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 1: {

faLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 2: {

fbLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 3: {

forestLabel.textProperty().setValue(area.getArea() + " m²" );

break;

}

}

break;

}

}

}

Utils.writeGeoJSON(this.lucd.getChangeDetection(), "src/resources/AppWebForm/res/result.json");

this.wktTextArea.textProperty().setValue(this.lucd.getWKT());

this.webEngine.executeScript("showResult();");

} catch (Exception e) {

Utils.showErrorMessage("Error",

e.getMessage(),

Arrays.toString(e.getStackTrace()));

} finally {

form.getDialogStage().close();

}

});

task.setOnFailed(e -> {

Utils.showErrorMessage("Error",

task.getException().getMessage(),

Arrays.toString(task.getException().getStackTrace()));

form.getDialogStage().close();

});

((Stage)this.appForm.getScene().getWindow()).setOnHiding(event -> task.cancel());

new Thread(task).start();

} catch (Exception e) {

Utils.showErrorMessage("Error",

e.getMessage(),

Arrays.toString(e.getStackTrace()));

}

}

private File roiFile;

public void selectROI(ActionEvent actionEvent) {

FileChooser fc = new FileChooser();

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("Shapefile (\*.shp)", "\*.shp");

fc.setSelectedExtensionFilter(filter);

this.roiFile = fc.showOpenDialog(appForm.getScene().getWindow());

if (this.roiFile == null) {

roiLabel.setText("");

} else {

roiLabel.setText(roiFile.getAbsolutePath());

}

}

public void openLUCD(ActionEvent actionEvent) {

FileChooser fc = new FileChooser();

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("XML files (\*.lucd)", "\*.lucd");

fc.setSelectedExtensionFilter(filter);

File file = fc.showOpenDialog(appForm.getScene().getWindow());

if (file == null || !file.exists()) {

Utils.showErrorMessage("Error",

"Please select LUCD file",

"");

return;

}

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file))) {

CDSer lucd = (CDSer) ois.readObject();

List<LandUseChangeDetectionResult> areas = lucd.resultList;

for (LandUseChangeDetectionResult area : areas) {

switch (area.getBefore()) {

case 0: {

switch (area.getAfter()) {

case 0: {

waterLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 1: {

waLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 2: {

wbLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 3: {

wfLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

}

break;

}

case 1: {

switch (area.getAfter()) {

case 0: {

awLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 1: {

agriLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 2: {

abLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 3: {

afLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

}

break;

}

case 2: {

switch (area.getAfter()) {

case 0: {

bwLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 1: {

baLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 2: {

buildLevel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 3: {

bfLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

}

break;

}

case 3: {

switch (area.getAfter()) {

case 0: {

fwLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 1: {

faLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 2: {

fbLabel.textProperty().setValue(area.getArea() + " m²");

break;

}

case 3: {

forestLabel.textProperty().setValue(area.getArea() + " m²" );

break;

}

}

break;

}

}

}

PrintWriter writer = new PrintWriter(new File("src/resources/AppWebForm/res/result.json"));

writer.write(lucd.json);

writer.close();

this.wktTextArea.textProperty().setValue(lucd.wkt);

this.webEngine.executeScript("showResult();");

} catch (Exception e) {

Utils.showErrorMessage("Error",

"Cannot open LUCD file",

"");

}

}

public void saveLUCD(ActionEvent actionEvent) {

if (this.lucd == null) {

Utils.showErrorMessage("Error",

"Please, make Land Use Change Detection Before",

"");

return;

}

FileChooser fc = new FileChooser();

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("XML files (\*.lucd)", "\*.lucd");

fc.setSelectedExtensionFilter(filter);

File file = fc.showSaveDialog(appForm.getScene().getWindow());

if (file != null) {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(file))) {

String gson = "";

FileInputStream inputStream = new FileInputStream("src/resources/AppWebForm/res/result.json");

Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF\_8.name());

while (scanner.hasNextLine()) {

gson += scanner.nextLine();

}

scanner.close();

inputStream.close();

CDSer s = new CDSer(this.lucd.getAreas(), gson, this.wktTextArea.getText());

oos.writeObject(s);

} catch (IOException e) {

e.printStackTrace();

Utils.showErrorMessage("Error",

"Cannot write LUCD file",

"");

}

}

}

public void saveSHP(ActionEvent actionEvent) {

if (this.lucd == null) {

Utils.showErrorMessage("Error",

"Please, make Land Use Change Detection Before",

"");

return;

}

FileChooser fc = new FileChooser();

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("XML files (\*.shp)", "\*.shp");

fc.setSelectedExtensionFilter(filter);

File file = fc.showSaveDialog(appForm.getScene().getWindow());

if (file != null) {

try {

Utils.writeShapefile(this.lucd.getChangeDetection(), file.getAbsolutePath());

} catch (IOException e) {

Utils.showErrorMessage("Error",

"Cannot write Shapefile",

"");

}

}

}

}

## LandUseChangeDetection

package LandUseChangeDetection;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.stage.Stage;

/\*\*

\* Hello world!

\*

\*/

public class LandUseChangeDetection extends Application

{

@Override

public void start(Stage primaryStage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("/app.fxml"));

primaryStage.setTitle("Land-Use Change Detector");

primaryStage.setScene(new Scene(root, 800, 700));

primaryStage.setMinWidth(800);

primaryStage.setMinHeight(700);

primaryStage.getIcons().add(new Image("/icon.png"));

primaryStage.show();

}

public static void main( String[] args )

{

launch(args);

}

}

## LandUseChangeDetection

package LandUseChangeDetection;

import java.io.Serializable;

public class LandUseChangeDetectionResult implements Serializable {

private double area;

private int before;

private int after;

public LandUseChangeDetectionResult(double area, int before, int after) {

this.area = area;

this.before = before;

this.after = after;

}

public double getArea() {

return area;

}

public int getBefore() {

return before;

}

public int getAfter() {

return after;

}

}

## Resolution

package LandUseChangeDetection;

/\*\*

\* Sentinel data resolutions

\*/

public enum Resolution {

R10m,

R20m,

R60m

}

## SentinelData

package LandUseChangeDetection;

import com.vividsolutions.jts.geom.Geometry;

import it.geosolutions.jaiext.JAIExt;

import org.apache.commons.io.FilenameUtils;

import org.esa.s2tbx.dataio.VirtualPath;

import org.esa.s2tbx.dataio.s2.S2Config;

import org.esa.s2tbx.dataio.s2.S2ProductNamingUtils;

import org.gdal.gdal.Dataset;

import org.gdal.gdal.gdal;

import org.gdal.gdalconst.gdalconstConstants;

import org.gdal.osr.SpatialReference;

import org.geotools.coverage.CoverageFactoryFinder;

import org.geotools.coverage.grid.GridCoordinates2D;

import org.geotools.coverage.grid.GridCoverage2D;

import org.geotools.coverage.grid.GridCoverageFactory;

import org.geotools.coverage.grid.GridGeometry2D;

import org.geotools.coverage.processing.CoverageProcessor;

import org.geotools.coverage.processing.Operations;

import org.geotools.coverage.processing.operation.Crop;

import org.geotools.coverageio.jp2k.JP2KReader;

import org.geotools.data.simple.SimpleFeatureCollection;

import org.geotools.gce.geotiff.GeoTiffWriter;

import org.geotools.geometry.jts.ReferencedEnvelope;

import org.opengis.coverage.grid.GridGeometry;

import org.opengis.geometry.Envelope;

import org.opengis.parameter.ParameterValueGroup;

import org.opengis.referencing.crs.CoordinateReferenceSystem;

import javax.media.jai.Interpolation;

import javax.media.jai.JAI;

import javax.media.jai.RasterFactory;

import javax.media.jai.RenderedOp;

import java.awt.\*;

import java.awt.image.DataBuffer;

import java.awt.image.Raster;

import java.awt.image.WritableRaster;

import java.awt.image.renderable.ParameterBlock;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.StandardCopyOption;

import java.text.SimpleDateFormat;

import java.util.\*;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class SentinelData {

static {

gdal.AllRegister();

JAIExt.initJAIEXT();

}

/\*\*

\* JP2k Extension

\*/

private static final String JP2K\_EXTENSION = "jp2";

/\*\*

\* Projection file

\*/

private static final String PRJ\_EXTENSION = ".prj";

/\*\*

\* World file

\*/

private static final String WDL\_EXTENSION = ".j2w";

/\*\*

\* Sentinel data bands

\*/

private List<GridCoverage2D> bands;

/\*\*

\* Bands getter

\* @return list of bands

\*/

List<GridCoverage2D> getBands() {

return bands;

}

void setBands(List<GridCoverage2D> bands) {

this.pixels = null;

this.bands = bands;

}

/\*\*

\* Sentinel data resolution

\*/

private Resolution resolution;

/\*\*

\* Resolution getter

\* @return Bands resolution

\*/

public Resolution getResolution() {

return resolution;

}

/\*\*

\* Get bands CRS

\* @return bands CRS

\*/

public CoordinateReferenceSystem getCRS() {

if (this.bands == null || this.bands.size() == 0) {

return null;

}

return this.bands.get(0).getCoordinateReferenceSystem2D();

}

public int getWidth() {

return this.bands.get(0).getRenderedImage().getWidth();

}

public int getHeight() {

return this.bands.get(0).getRenderedImage().getHeight();

}

/\*\*

\* Get bands' grid dimension

\* @return grid dimension

\*/

public Dimension getGridDimension() {

if (this.bands == null || this.bands.size() == 0) {

return null;

}

return new Dimension(this.getWidth(), this.getHeight());

}

/\*\*

\* Get raster's envelope

\* @return Raster envelope

\*/

public Envelope getEnvelope() {

if (this.bands == null || this.bands.size() == 0) {

return null;

}

return this.bands.get(0).getEnvelope();

}

/\*\*

\* Data pixels

\*/

private double[][] pixels;

/\*\*

\* Get pixel data vector

\* @param pixel pixel number

\* @return Data vector

\*/

double[] getPixelVector(int pixel) {

// Lazy initialization

if (pixels == null) {

pixels = new double[bands.size()][];

for (int i = 0; i < bands.size(); i++) {

Raster band = bands.get(i).getRenderedImage().getData();

pixels[i] = new double[this.getWidth() \* this.getHeight()];

band.getPixels(band.getMinX(), band.getMinY(), this.getWidth(), this.getHeight(), pixels[i]);

}

}

if (this.pixels[0][pixel] < 0) {

return null;

}

double[] vector = new double[bands.size()];

for (int i = 0; i < vector.length; i++) {

vector[i] = pixels[i][pixel];

}

return vector;

}

/\*\*

\* Check product structure

\* @param dataFile Path to product XML

\* @return List of granules

\*/

public static List<VirtualPath> checkAndGetGranules(File dataFile) throws Exception {

// Check Sentinel 2 data

VirtualPath xmlPath = new VirtualPath(dataFile.getAbsolutePath());

if (!S2ProductNamingUtils.checkStructureFromProductXml(xmlPath)) {

throw new Exception("Error, invalid Sentinel 2 product");

}

S2Config.Sentinel2ProductLevel level = S2ProductNamingUtils.getLevel(

xmlPath, S2Config.Sentinel2InputType.INPUT\_TYPE\_PRODUCT\_METADATA);

if (level != S2Config.Sentinel2ProductLevel.L2A) {

throw new Exception("Error, Sentinel 2 level should be Level-2A, not " + level.toString() +

" Please, use Level updater");

}

if (!S2ProductNamingUtils.hasValidStructure(

S2Config.Sentinel2InputType.INPUT\_TYPE\_PRODUCT\_METADATA, xmlPath)) {

throw new Exception("Error, Sentinel 2 Product doesn't have valid structure");

}

return S2ProductNamingUtils.getTilesFromProductXml(xmlPath);

}

/\*\*

\* Sentinel 2 Data sensing datetime

\*/

private Date sensingDate;

/\*\*

\* Sensing date getter

\* @return Sentinel 2 sensing datetime

\*/

Date getSensingDate() {

return this.sensingDate;

}

/\*\*

\* Date extracting pattern

\*/

private static final Pattern DATE\_PATTERN = Pattern.compile("(.\*)\_(\\d{8}T\\d{6})(.\*)");

/\*\*

\* Date layout

\*/

private static final SimpleDateFormat FORMATTER = new SimpleDateFormat("yyyyMMdd'T'HHmmss");

/\*\*

\* Extracting Sentinel 2 data sensing date and time

\* @param granuleDir Sentinel 2 data granule

\* @return sensing datetime

\* @throws Exception if granule is not exist or contains incorrect sensing date

\*/

private static Date getSensingDate(File granuleDir) throws Exception {

Matcher m = DATE\_PATTERN.matcher(granuleDir.getName());

if (!m.find()) {

throw new Exception("Error, granule directory doesn't contains sensing date info");

}

String sensingDateString = m.group(2);

return FORMATTER.parse(sensingDateString);

}

private ClassificationEnum type;

public ClassificationEnum getType() {

return type;

}

public static ClassificationEnum getType(File file) throws Exception {

if (file.getParentFile().getName().startsWith("S2A")) {

return ClassificationEnum.A;

} else if (file.getParentFile().getName().startsWith("S2B")) {

return ClassificationEnum.B;

}

throw new Exception("Please select valid Sentinel file");

}

SentinelData(File granuleDir, Resolution r, ClassificationEnum type) throws Exception {

this.type = type;

this.resolution = r;

// Get date

this.sensingDate = getSensingDate(granuleDir);

// Get bands' files

StringBuilder fileBuilder = new StringBuilder(granuleDir.getAbsolutePath());

fileBuilder.append(File.separator);

fileBuilder.append("IMG\_DATA");

fileBuilder.append(File.separator);

File imgDataFile = new File(fileBuilder.toString());

fileBuilder.append(r);

fileBuilder.append(File.separator);

File file = new File(fileBuilder.toString());

// Checking for old scl data

File[] imgDataDirFiles = imgDataFile.listFiles();

if (imgDataDirFiles != null && imgDataDirFiles.length > 1) {

for (File f : imgDataDirFiles) {

if (f.getName().contains("SCL")) {

File target = new File(file.getAbsolutePath() + File.separator + f.getName());

if (!target.exists()) {

Files.copy(f.toPath(), target.toPath());

}

}

}

}

File[] files = file.listFiles();

if (files == null) {

throw new NullPointerException("Error, incorrect SL2 Data");

}

// TODO: Filter bands

// TODO: Interpolation 10m

bands = new ArrayList<>(files.length);

for (File bandFile : files) {

if (FilenameUtils.getExtension(bandFile.getName()).equals(JP2K\_EXTENSION)) {

if (!bandFile.getName().contains("TCI") && !bandFile.getName().contains("AOT")) {

bands.add(openSentinelData(bandFile));

}

}

}

fileBuilder = new StringBuilder(granuleDir.getAbsolutePath());

fileBuilder.append(File.separator);

fileBuilder.append("QI\_DATA");

fileBuilder.append(File.separator);

file = new File(fileBuilder.toString());

files = file.listFiles();

if (files == null) {

throw new NullPointerException("Error, incorrect SL2 Data");

}

// TODO: 10m

String resolutionMarker = r.toString().substring(1);

for (File qFile : files) {

if (FilenameUtils.getExtension(qFile.getName()).equals(JP2K\_EXTENSION)){

if (qFile.getPath().endsWith(resolutionMarker + ".jp2") && qFile.getPath().contains("CLD")) {

this.cloudsMaskFile = qFile;

continue;

}

if (qFile.getPath().endsWith(resolutionMarker + ".jp2") && qFile.getPath().contains("SNW")) {

this.snowMaskFile = qFile;

}

}

}

if (this.cloudsMaskFile == null || this.snowMaskFile == null ) {

throw new Exception("Error, granule not contain quality data");

}

}

/\*\*

\* Open Sentinel 2 Data

\* @param bandFile band file

\* @return Band data grid coverage

\* @throws IOException if band file doesn't exists

\*/

private GridCoverage2D openSentinelData(File bandFile) throws Exception {

// Checking for projection and world file

if (!checkPRJAndJ2W(bandFile)) {

writeJP2Info(bandFile);

}

// Open jp2k sentinel band

JP2KReader reader = new JP2KReader(bandFile);

return reader.read(null);

}

/\*\*

\* Check for existing of prj and world file

\* @param bandFile Sentinel band file

\* @return existing

\*/

private boolean checkPRJAndJ2W(File bandFile) {

String pathWithoutExtension = FilenameUtils.removeExtension(bandFile.getAbsolutePath());

File prjFile = new File(pathWithoutExtension + PRJ\_EXTENSION);

if (!prjFile.exists()) {

return false;

}

File wdlFile = new File(pathWithoutExtension + WDL\_EXTENSION);

return wdlFile.exists();

}

/\*\*

\* Create prj and world file

\* @param bandFile Sentinel band file

\*/

private void writeJP2Info(File bandFile) throws Exception {

//gdal.AllRegister();

String[] args = new String[]{bandFile.getAbsolutePath()};

args = gdal.GeneralCmdLineProcessor(args);

Dataset hDataSet = gdal.Open(args[0], gdalconstConstants.GA\_ReadOnly);

if (hDataSet == null) {

throw new Exception("GDALOpen failed, " + gdal.GetLastErrorNo() + " " + gdal.GetLastErrorMsg());

}

String fileWithoutExtension = FilenameUtils.removeExtension(bandFile.getAbsolutePath());

writeProjection(hDataSet, new File(fileWithoutExtension + PRJ\_EXTENSION));

writeWorldFile(hDataSet, new File(fileWithoutExtension + WDL\_EXTENSION));

hDataSet.delete();

}

/\*\*

\* Write jp2k projection file

\* @param jp2kDataSet jp2k data set

\* @param prjFile prj file

\* @throws IOException Output stream

\*/

private void writeProjection(Dataset jp2kDataSet, File prjFile) throws IOException {

String prjStr = jp2kDataSet.GetProjectionRef();

SpatialReference srs = new SpatialReference(prjStr);

if (prjStr.length() != 0) {

String[] prjPrettyWKT = new String[1];

srs.ExportToPrettyWkt(prjPrettyWKT, 0);

prjStr = prjPrettyWKT[0];

}

srs.delete();

try (BufferedWriter writer = new BufferedWriter(new FileWriter(prjFile.getAbsolutePath()))) {

writer.write(prjStr);

}

}

/\*\*

\* Write jp2k world file

\* @param hDataSet Sentinel data set

\* @param j2wFile World file

\* @throws IOException Output

\*/

private void writeWorldFile(Dataset hDataSet, File j2wFile) throws IOException {

double[] geoTransform = new double[6];

hDataSet.GetGeoTransform(geoTransform);

StringBuilder wb = new StringBuilder();

wb.append(geoTransform[1]);

wb.append(System.lineSeparator());

wb.append(geoTransform[2]);

wb.append(System.lineSeparator());

wb.append(geoTransform[4]);

wb.append(System.lineSeparator());

wb.append(geoTransform[5]);

wb.append(System.lineSeparator());

double temp = geoTransform[0] + geoTransform[1] / 2;

wb.append(temp);

wb.append(System.lineSeparator());

temp = geoTransform[3] + geoTransform[5] / 2;

wb.append(temp);

try (BufferedWriter writer = new BufferedWriter(new FileWriter(j2wFile.getAbsolutePath()))) {

writer.write(wb.toString());

}

}

/\*\*

\* Crop bands

\* @param envelope cropping envelope

\*/

void cropBands(Envelope envelope) throws Exception {

final CoverageProcessor processor = new CoverageProcessor();

ParameterValueGroup params = processor.getOperation("CoverageCrop").getParameters();

params.parameter("Envelope").setValue(envelope);

ListIterator<GridCoverage2D> it = this.bands.listIterator();

while (it.hasNext()) {

GridCoverage2D band = it.next();

params.parameter("Source").setValue(band);

band = (GridCoverage2D) processor.doOperation(params);

it.set(band);

}

if (this.cloudAndSnowMask == null) {

this.getCloudsAndSnowMask();

}

params.parameter("Source").setValue(this.cloudAndSnowMask);

this.cloudAndSnowMask = (GridCoverage2D) processor.doOperation(params);

// ListIterator<GridCoverage2D> ita = this.bands.listIterator();

// while (ita.hasNext()) {

// GridCoverage2D band = ita.next();

// File file = new File("C:\\Users\\Arthur\\Desktop\\Java\\" + ita.nextIndex() + ".tiff");

// GeoTiffWriter writer = new GeoTiffWriter(file);

// writer.write(band, null);

// }

}

void cropBands(Geometry geometry) throws Exception {

final CoverageProcessor processor = new CoverageProcessor();

ParameterValueGroup params = processor.getOperation("CoverageCrop").getParameters();

params.parameter(Crop.PARAMNAME\_ROI).setValue(geometry);

ListIterator<GridCoverage2D> it = this.bands.listIterator();

while (it.hasNext()) {

GridCoverage2D band = it.next();

params.parameter("Source").setValue(band);

band = (GridCoverage2D) processor.doOperation(params);

it.set(band);

}

if (this.cloudAndSnowMask == null) {

this.getCloudsAndSnowMask();

}

params.parameter("Source").setValue(this.cloudAndSnowMask);

this.cloudAndSnowMask = (GridCoverage2D) processor.doOperation(params);

// ListIterator<GridCoverage2D> ita = this.bands.listIterator();

// while (ita.hasNext()) {

// GridCoverage2D band = ita.next();

// File file = new File("C:\\Users\\Arthur\\Desktop\\Java\\" + ita.nextIndex() + ".tiff");

// GeoTiffWriter writer = new GeoTiffWriter(file);

// writer.write(band, null);

// }

}

// TODO: comment

private static GridCoverage2D interpolateData(GridCoverage2D coverage, GridGeometry2D gridGeometry2D) {

Interpolation interpolation = Interpolation.getInstance(Interpolation.INTERP\_NEAREST);

return (GridCoverage2D) Operations.DEFAULT.resample(coverage, coverage.getCoordinateReferenceSystem(), gridGeometry2D, interpolation);

}

/\*\*

\* Clouds mask file

\*/

private File cloudsMaskFile;

/\*\*

\* Snow mask file

\*/

private File snowMaskFile;

/\*\*

\* Low value for excluding values

\*/

private static final Double BORDER\_VALUE = 90.0;

/\*\*

\* Clouds and snow mask

\*/

private GridCoverage2D cloudAndSnowMask;

/\*\*

\* Get clouds and snow mask

\* @return clouds and snow mask

\*/

public GridCoverage2D getCloudsAndSnowMask() throws Exception {

if (this.cloudAndSnowMask == null) {

GridCoverage2D cloudsMask = openSentinelData(cloudsMaskFile);

GridCoverage2D snowMask = openSentinelData(snowMaskFile);

// JAI operations

ParameterBlock maskOp = new ParameterBlock();

maskOp.addSource(cloudsMask.getRenderedImage());

maskOp.addSource(snowMask.getRenderedImage());

RenderedOp cloudAndSnowMask = JAI.create("Or", maskOp);

ParameterBlock selectOp = new ParameterBlock();

selectOp.addSource(cloudAndSnowMask);

selectOp.add(BORDER\_VALUE);

cloudAndSnowMask = JAI.create("Binarize", selectOp);

GridCoverageFactory factory = new GridCoverageFactory();

ReferencedEnvelope envelope = new ReferencedEnvelope(cloudsMask.getEnvelope());

this.cloudAndSnowMask = factory.create("CloudAndSnowMask", cloudAndSnowMask, envelope);

}

return this.cloudAndSnowMask;

}

// /\*\*

// \* Merge masks

// \* @param first first mask

// \* @param second second mask

// \* @return merged mask

// \* @throws Exception if mask is null or not equals

// \*/

// public static GridCoverage2D mergeMasks(GridCoverage2D first, GridCoverage2D second) throws Exception {

// if (first == null || second == null) {

// throw new Exception("Error, mask cannot be null");

// }

// Envelope a = first.getEnvelope2D();

// Envelope b = second.getEnvelope2D();

// if (first.getEnvelope() != second.getEnvelope()) {

// //throw new Exception("Error, masks must have equal envelopes");

// }

// // JAI operations

// ParameterBlock maskOp = new ParameterBlock();

// maskOp.addSource(first.getRenderedImage());

// maskOp.addSource(second.getRenderedImage());

// RenderedOp mask = JAI.create("Or", maskOp);

//

// GridCoverageFactory factory = new GridCoverageFactory();

// ReferencedEnvelope envelope = new ReferencedEnvelope(first.getEnvelope());

//

// return factory.create("Mask", mask, envelope);

// }

}

## SentinelLevelUpdater

package LandUseChangeDetection;

import java.io.File;

import java.io.IOException;

import java.util.Map;

import java.util.concurrent.Semaphore;

public class SentinelLevelUpdater {

/\*\*

\* Sen2Cor semaphore

\*/

public static final Semaphore SEMAPHORE = new Semaphore(1, true);

/\*\*

\* Update progress

\*/

private Process process;

/\*\*

\* Progress getter

\* @return Update process

\*/

public Process getProcess() {

return this.process;

}

/\*\*

\* Level up Sentinel 2 data

\* @param sentinel1CLevel Sentinel 2 data directory

\* @param param Sentinel data resolutions

\* @throws IOException Ability to call to Sen2Cor

\* @throws InterruptedException Semaphore checking

\*/

public void levelUp(File sentinel1CLevel, String param) throws IOException, InterruptedException {

this.process = upTo2ALevel(sentinel1CLevel, param);

}

/\*\*

\* Update Sentinel 1C level to 2A level

\* @param sentinel1CLevel Sentinel 2 data directory

\* @param param Sentinel data resolutions

\* @throws IOException Ability to call to Sen2Cor

\* @throws InterruptedException Semaphore checking

\*/

private Process upTo2ALevel(File sentinel1CLevel, String param) throws IOException, InterruptedException {

SEMAPHORE.acquire();

// TODO: Change to Linux

ProcessBuilder pb = new ProcessBuilder("cmd.exe", "/C", "start", "/B",

"L2A\_Process", sentinel1CLevel.getAbsolutePath(), param);

Map<String, String> env = pb.environment();

Map<String, String> sEnv = System.getenv();

for (Map.Entry<String, String> entry : sEnv.entrySet()) {

env.put(entry.getKey(), entry.getValue());

}

pb.redirectErrorStream(true);

return process = pb.start();

}

}

## Utils

package LandUseChangeDetection;

import com.vividsolutions.jts.geom.Geometry;

import com.vividsolutions.jts.geom.MultiPolygon;

import javafx.scene.control.Alert;

import org.geotools.coverage.grid.GridCoverage2D;

import org.geotools.coverage.processing.CoverageProcessor;

import org.geotools.data.DataStore;

import org.geotools.data.DataStoreFinder;

import org.geotools.data.DefaultTransaction;

import org.geotools.data.Transaction;

import org.geotools.data.shapefile.ShapefileDataStore;

import org.geotools.data.shapefile.ShapefileDataStoreFactory;

import org.geotools.data.simple.SimpleFeatureCollection;

import org.geotools.data.simple.SimpleFeatureIterator;

import org.geotools.data.simple.SimpleFeatureSource;

import org.geotools.data.simple.SimpleFeatureStore;

import org.geotools.feature.DefaultFeatureCollection;

import org.geotools.feature.simple.SimpleFeatureBuilder;

import org.geotools.feature.simple.SimpleFeatureTypeBuilder;

import org.geotools.geojson.feature.FeatureJSON;

import org.geotools.geometry.jts.JTS;

import org.geotools.referencing.CRS;

import org.opengis.feature.simple.SimpleFeature;

import org.opengis.feature.simple.SimpleFeatureType;

import org.opengis.geometry.Envelope;

import org.opengis.parameter.ParameterValueGroup;

import org.opengis.referencing.FactoryException;

import org.opengis.referencing.crs.CoordinateReferenceSystem;

import org.opengis.referencing.operation.MathTransform;

import org.opengis.referencing.operation.TransformException;

import java.io.\*;

import java.net.MalformedURLException;

import java.util.HashMap;

import java.util.ListIterator;

import java.util.Map;

import java.util.TreeMap;

public class Utils {

/\*\*

\* Read shapefile layers

\* @param shpFile ERSI shapefile

\* @return shapefile data store

\* @throws IOException if file is not exists

\*/

static DataStore openShapefile(File shpFile) throws IOException {

Map<String, Object> map = new TreeMap<>();

try {

map.put("url", shpFile.toURI().toURL());

} catch (MalformedURLException e) {

e.printStackTrace();

}

return DataStoreFinder.getDataStore(map);

}

static SimpleFeatureCollection transformToCRS(SimpleFeatureCollection fc, CoordinateReferenceSystem crs) throws FactoryException {

CoordinateReferenceSystem vectorCRS = fc.getSchema().getCoordinateReferenceSystem();

if (!CRS.equalsIgnoreMetadata(vectorCRS, crs)) {

MathTransform transform = CRS.findMathTransform(vectorCRS, crs, true);

// Create transformed feature collection

SimpleFeatureTypeBuilder typeBuilder = new SimpleFeatureTypeBuilder();

typeBuilder.setName(fc.getSchema().getName());

typeBuilder.setCRS(crs);

typeBuilder.add("the\_geom", MultiPolygon.class);

final SimpleFeatureType featureType = typeBuilder.buildFeatureType();

DefaultFeatureCollection transformedFC = new DefaultFeatureCollection(null, null);

SimpleFeatureBuilder featureBuilder = new SimpleFeatureBuilder(featureType);

try (SimpleFeatureIterator it = fc.features()) {

while (it.hasNext()) {

SimpleFeature feature = it.next();

Geometry geometry = (Geometry) feature.getDefaultGeometry();

geometry = JTS.transform(geometry, transform);

featureBuilder.add(geometry);

SimpleFeature transformedFeature = featureBuilder.buildFeature(null);

transformedFC.add(transformedFeature);

}

} catch (TransformException e) {

e.printStackTrace();

}

// Replace collection

fc = transformedFC;

}

return fc;

}

/\*\*

\*

\* @param fc

\* @param crs

\* @return

\* @throws FactoryException

\*/

static SimpleFeatureCollection transformChangeDetectionCollectionCRS(SimpleFeatureCollection fc, CoordinateReferenceSystem crs) throws FactoryException {

CoordinateReferenceSystem vectorCRS = fc.getSchema().getCoordinateReferenceSystem();

if (!CRS.equalsIgnoreMetadata(vectorCRS, crs)) {

MathTransform transform = CRS.findMathTransform(vectorCRS, crs, true);

// Create transformed feature collection

SimpleFeatureTypeBuilder typeBuilder = new SimpleFeatureTypeBuilder();

typeBuilder.setName(fc.getSchema().getName());

typeBuilder.setCRS(crs);

typeBuilder.add("the\_geom", MultiPolygon.class);

typeBuilder.add("before", Integer.class);

typeBuilder.add("after", Integer.class);

final SimpleFeatureType featureType = typeBuilder.buildFeatureType();

DefaultFeatureCollection transformedFC = new DefaultFeatureCollection(null, null);

SimpleFeatureBuilder featureBuilder = new SimpleFeatureBuilder(featureType);

try (SimpleFeatureIterator it = fc.features()) {

while (it.hasNext()) {

SimpleFeature feature = it.next();

Geometry geometry = (Geometry) feature.getDefaultGeometry();

geometry = JTS.transform(geometry, transform);

featureBuilder.add(geometry);

featureBuilder.add(feature.getAttribute("before"));

featureBuilder.add(feature.getAttribute("after"));

SimpleFeature transformedFeature = featureBuilder.buildFeature(null);

transformedFC.add(transformedFeature);

}

} catch (TransformException e) {

e.printStackTrace();

}

// Replace collection

fc = transformedFC;

}

return fc;

}

/\*\*

\* Check for level

\* @param file sentinel 2 data

\* @return existing

\*/

public static boolean isLevel2A(File file) {

if (!file.isDirectory()) {

return false;

}

String[] files = file.list();

if (files == null) {

return false;

}

for (String s : files) {

if (s.equals("MTD\_MSIL2A.xml")) {

return true;

}

}

return false;

}

/\*\*

\* Show JavaFX error message

\* @param title message window title

\* @param header message header

\* @param content message content

\*/

public static void showErrorMessage(String title, String header, String content) {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle(title);

alert.setHeaderText(header);

alert.setContentText(content);

alert.showAndWait();

}

/\*\*

\* Crop grid coverage by envelope

\* @param scene grid coverage

\* @param envelope envelope

\* @return cropped grid coverage

\*/

public static GridCoverage2D cropGridCoverage(GridCoverage2D scene, Envelope envelope) {

final CoverageProcessor processor = new CoverageProcessor();

ParameterValueGroup params = processor.getOperation("CoverageCrop").getParameters();

params.parameter("Envelope").setValue(envelope);

params.parameter("Source").setValue(scene);

return (GridCoverage2D)processor.doOperation(params);

}

static void writeShapefile(SimpleFeatureCollection collection, String path) throws IOException {

ShapefileDataStoreFactory dataStoreFactory = new ShapefileDataStoreFactory();

File file = new File(path);

Map<String, Serializable> params = new HashMap<>();

params.put("url", file.toURI().toURL());

params.put("create spatial index", Boolean.TRUE);

ShapefileDataStore dataStore = (ShapefileDataStore) dataStoreFactory.createDataStore(params);

dataStore.createSchema(collection.getSchema());

Transaction transaction = new DefaultTransaction("create");

String typeName = dataStore.getTypeNames()[0];

SimpleFeatureSource featureSource = dataStore.getFeatureSource(typeName);

if (featureSource instanceof SimpleFeatureStore) {

SimpleFeatureStore featureStore = (SimpleFeatureStore) featureSource;

featureStore.setTransaction(transaction);

try {

featureStore.addFeatures(collection);

transaction.commit();

} catch (Exception ex) {

ex.printStackTrace();

transaction.rollback();

} finally {

transaction.close();

}

}

}

static void writeGeoJSON(SimpleFeatureCollection collection, String path) throws IOException {

FeatureJSON featureJSON = new FeatureJSON();

File file = new File(path);

try (FileOutputStream fos = new FileOutputStream(file)){

featureJSON.writeFeatureCollection(collection, fos);

}

}

}

## Data

package LandUseChangeDetection.data;

import LandUseChangeDetection.LandUseChangeDetectionResult;

import org.geotools.data.DataStore;

import org.geotools.data.DataStoreFinder;

import org.geotools.data.DefaultTransaction;

import org.geotools.data.Transaction;

import org.geotools.data.simple.SimpleFeatureCollection;

import org.geotools.data.simple.SimpleFeatureStore;

import org.postgresql.PGConnection;

import org.postgresql.util.PGobject;

import java.io.IOException;

import java.sql.\*;

import java.util.\*;

public class Data {

/\*\*

\* Sql connection

\*/

private static Connection connection;

static {

try {

Class.forName("org.postgresql.Driver");

connection = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5433/LUCD",

"postgres", "admin");

// Add geometry type to the connection

((PGConnection)connection).addDataType("geometry", (Class<? extends PGobject>) Class.forName("org.postgis.PGgeometry"));

} catch (ClassNotFoundException | SQLException e) {

e.printStackTrace();

System.exit(0);

}

}

public static List<LandUseChangeDetectionResult> getSquares(SimpleFeatureCollection collection) throws Exception {

// Insert both collection to PostGIS

List<LandUseChangeDetectionResult> results;

try {

insertCollection(collection);

results = getLandUseChanges();

} finally {

clearDB();

}

return results;

}

/\*\*

\* Insert clssificated collection into PostGIS

\* @param collection classificated collection

\*/

private static void insertCollection(SimpleFeatureCollection collection) throws IOException {

Map<String, Object> params = new HashMap<>();

params.put("dbtype", "postgis");

params.put("host", "localhost");

params.put("port", 5433);

params.put("schema", "public");

params.put("database", "LUCD");

params.put("user", "postgres");

params.put("passwd", "admin");

DataStore pgStore = DataStoreFinder.getDataStore(params);

SimpleFeatureStore store = (SimpleFeatureStore) pgStore.getFeatureSource("landuses");

Transaction tx = new DefaultTransaction("Add");

store.setTransaction(tx);

try {

store.addFeatures(collection);

tx.commit();

} catch (Exception ex) {

tx.rollback();

throw ex;

} finally {

tx.close();

}

pgStore.dispose();

}

/\*\*

\* Land use change areas result

\*/

private static final String CHANGE\_SQUARES\_QUERY;

static {

CHANGE\_SQUARES\_QUERY = "SELECT sum(st\_area(the\_geom)), before, after " +

"FROM landuses " +

"GROUP BY before, after ";

}

/\*\*

\* Get land use change areas

\* @return list of change results

\*/

private static List<LandUseChangeDetectionResult> getLandUseChanges() throws Exception {

PreparedStatement statement = connection.prepareStatement(CHANGE\_SQUARES\_QUERY);

ResultSet resultSet = statement.executeQuery();

List<LandUseChangeDetectionResult> results = new ArrayList<>();

while (resultSet.next()) {

results.add(new LandUseChangeDetectionResult(

resultSet.getDouble(1),

resultSet.getInt(2),

resultSet.getInt(3)));

}

return results;

}

private static void clearDB() throws SQLException {

PreparedStatement statement = connection.prepareStatement("TRUNCATE landuses");

statement.execute();

}

}

## Setup.sql

-- -- Enable PostGIS (includes raster)

-- CREATE EXTENSION postgis;

-- -- Enable Topology

-- CREATE EXTENSION postgis\_topology;

-- -- Enable PostGIS Advanced 3D

-- -- and other geoprocessing algorithms

-- -- sfcgal not available with all distributions

-- CREATE EXTENSION postgis\_sfcgal;

-- -- fuzzy matching needed for Tiger

-- CREATE EXTENSION fuzzystrmatch;

-- -- rule based standardizer

-- CREATE EXTENSION address\_standardizer;

-- -- example rule data set

-- CREATE EXTENSION address\_standardizer\_data\_us;

-- -- Enable US Tiger Geocoder

-- CREATE EXTENSION postgis\_tiger\_geocoder;

DROP TABLE landUses;

CREATE TABLE landUses(

id SERIAL PRIMARY KEY,

the\_geom geography NOT NULL,

before INTEGER NOT NULL,

after INTEGER NOT NULL

);

## LevelUpForm

package LandUseChangeDetection.forms;

import LandUseChangeDetection.SentinelLevelUpdater;

import LandUseChangeDetection.Utils;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.concurrent.Task;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.\*;

import javafx.scene.layout.AnchorPane;

import javafx.stage.DirectoryChooser;

import javafx.stage.Stage;

import java.io.BufferedReader;

import java.io.File;

import java.io.InputStreamReader;

import java.util.Arrays;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class LevelUpForm {

/\*\*

\* Level up form's anchor pane

\*/

public AnchorPane levelUpForm;

/\*\*

\* Sentinel 2 Data path label

\*/

public Label pathLabel;

/\*\*

\* Current task label

\*/

public Label taskLabel;

/\*\*

\* Sentinel 2 Level up progress bar

\*/

public ProgressBar progressBar;

/\*\*

\* Start level up button

\*/

public Button create2ADataButton;

/\*\*

\* Interrupt creation button

\*/

public Button cancelButton;

/\*\*

\* Select file button

\*/

public Button openSentinel1;

/\*\*

\* Data resolutions

\*/

private final ObservableList<String> resolutions = FXCollections.observableArrayList("All", "10m", "20m", "60m");

/\*\*

\* Resolution choice box

\*/

public ChoiceBox resChoiceBox;

/\*\*

\* Sentinel 2 Level 1C file

\*/

private File sentinel1CLevelFile = null;

@FXML

void initialize() {

resChoiceBox.setItems(resolutions);

resChoiceBox.setValue("All");

}

/\*\*

\* Open Sentinel 2 Level 1C file

\* @param actionEvent Opening action event

\*/

public void openSentinel1CLevelData(ActionEvent actionEvent) {

DirectoryChooser dc = new DirectoryChooser();

this.sentinel1CLevelFile = dc.showDialog(levelUpForm.getScene().getWindow());

if (this.sentinel1CLevelFile == null) {

pathLabel.setText("Data not selected");

create2ADataButton.setDisable(true);

cancelButton.setDisable(true);

} else {

pathLabel.setText(this.sentinel1CLevelFile.getAbsolutePath());

create2ADataButton.setDisable(false);

}

}

/\*\*

\* Sentinel Level updater

\*/

private SentinelLevelUpdater updater;

/\*\*

\* Create Sentinel 2 Level 2A Data

\* @param actionEvent Creating action event

\*/

public void createL2ADataHandler(ActionEvent actionEvent) {

if (this.sentinel1CLevelFile == null) {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Error Dialog");

alert.setHeaderText("Sentinel 2 Data Not Selected");

alert.setContentText("Please, select Sentinel 2 Level 1C File");

alert.showAndWait();

return;

}

if (Utils.isLevel2A(this.sentinel1CLevelFile)) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Info");

alert.setHeaderText("Converting is not needed");

alert.setContentText("Sentinel 2 Data already on Level 2A");

alert.showAndWait();

return;

}

openSentinel1.setDisable(true);

create2ADataButton.setDisable(true);

cancelButton.setDisable(false);

resChoiceBox.setDisable(true);

String temp = (String)resChoiceBox.getValue();

if (temp.equals("All")){

temp = "";

} else {

temp = "--resolution=" + temp.substring(0, temp.length() - 1);

}

final String param = temp;

this.updater = new SentinelLevelUpdater();

// Update Task

Task updateTask = new Task() {

@Override

protected Object call() {

try {

updater.levelUp(sentinel1CLevelFile, param);

BufferedReader reader = new BufferedReader(new InputStreamReader(updater.getProcess().getInputStream(), "cp866"));

String line;

Pattern pattern = Pattern.compile("Progress\\[%]: (\\d+\\.\\d+) : (.\*)");

while ((line = reader.readLine()) != null) {

// Checking for cancel

if (this.isCancelled()) {

Runtime rt = Runtime.getRuntime();

if (System.getProperty("os.name").toLowerCase().contains("windows")) {

for (int i = 0; i < 3; ++i) {

rt.exec("taskkill /F /IM python.exe");

}

} else {

rt.exec("kill -9 python.exe");

}

updater.getProcess().destroy();

break;

}

Matcher m = pattern.matcher(line);

if (m.find()) {

updateMessage(m.group(2));

updateProgress(Double.parseDouble(m.group(1)), 100.0);

} else {

updateMessage(line);

}

}

updateMessage("Canceled");

updateProgress(0.0, 100.0);

// Show finished info

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setHeaderText("Level up Finished");

alert.setTitle("Level up process finished");

alert.setContentText("Level up process to Level-2A Product finished");

alert.showAndWait();

} catch (Exception e) {

SentinelLevelUpdater.SEMAPHORE.release();

// Show error info

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Error");

alert.setHeaderText(e.getMessage());

alert.setContentText(Arrays.toString(e.getStackTrace()));

alert.showAndWait();

} finally {

SentinelLevelUpdater.SEMAPHORE.release();

openSentinel1.setDisable(false);

cancelButton.setDisable(true);

create2ADataButton.setDisable(false);

resChoiceBox.setDisable(false);

}

return null;

}

};

progressBar.progressProperty().bind(updateTask.progressProperty());

taskLabel.textProperty().bind(updateTask.messageProperty());

// Task cancel

cancelButton.setOnMouseClicked(event -> updateTask.cancel());

// Stage exit

((Stage)levelUpForm.getScene().getWindow()).setOnHiding(event -> updateTask.cancel());

new Thread(updateTask).start();

}

}

## ProgressForm

package LandUseChangeDetection.forms;

import javafx.concurrent.Task;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.control.ProgressBar;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

public class ProgressForm {

private final Stage dialogStage;

private final ProgressBar pb = new ProgressBar();

private final Label status = new Label();

public ProgressForm() {

dialogStage = new Stage();

dialogStage.initStyle(StageStyle.UTILITY);

dialogStage.setResizable(false);

dialogStage.initModality(Modality.APPLICATION\_MODAL);

dialogStage.setTitle("Loading...");

dialogStage.setWidth(400);

dialogStage.setHeight(150);

pb.setProgress(-1F);

pb.setPrefWidth(380);

final VBox hb = new VBox();

hb.setSpacing(5);

hb.setAlignment(Pos.CENTER);

hb.getChildren().addAll(pb, status);

Scene scene = new Scene(hb, 600, 400);

dialogStage.setScene(scene);

}

public void activateProgressBar(final Task<?> task) {

pb.progressProperty().bind(task.progressProperty());

status.textProperty().bind(task.messageProperty());

dialogStage.show();

}

public Stage getDialogStage() {

return dialogStage;

}

}

## SearchAndDownloadForm

package LandUseChangeDetection.forms;

import LandUseChangeDetection.Utils;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.\*;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.VBox;

import javafx.scene.web.WebEngine;

import javafx.scene.web.WebView;

import javafx.util.Callback;

import javafx.util.converter.NumberStringConverter;

import org.apache.abdera.Abdera;

import org.apache.abdera.model.Document;

import org.apache.abdera.model.Entry;

import org.apache.abdera.model.Feed;

import org.apache.abdera.protocol.Response;

import org.apache.abdera.protocol.client.AbderaClient;

import org.apache.abdera.protocol.client.ClientResponse;

import org.apache.commons.httpclient.UsernamePasswordCredentials;

import org.apache.commons.httpclient.auth.AuthScope;

import java.awt.\*;

import java.io.File;

import java.io.IOException;

import java.io.InputStream;

import java.net.URI;

import java.net.URISyntaxException;

import java.net.URLConnection;

import java.time.Instant;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.time.ZoneId;

import java.util.Base64;

import java.util.List;

public class SearchAndDownloadForm {

/\*\*

\* ESA open hub portal url

\*/

private static final String ESA\_OPEN\_HUB\_PORTAL\_URL = "https://scihub.copernicus.eu";

/\*\*

\* ESA Open Search base

\*/

private static final String OPEN\_SEARCH\_QUERY\_BASE = "https://scihub.copernicus.eu/apihub/search?start=0&rows=100&q=";

/\*\*

\* ESA open hub api url

\*/

private static final String esaOpenHubURL = "https://scihub.copernicus.eu/apihub/odata/v1/";

public PasswordField passwordTextField;

public TextField loginTextField;

public DatePicker sensingStartDate;

public DatePicker sensingFinishDate;

public TextField maxCloudPercentage;

public WebView webMap;

public SplitPane splitPane;

public Button changeButton;

public Button loginButton;

public ListView resultListView;

public TabPane tab;

public Button searchButton;

public Hyperlink signInLink;

/\*\*

\* Open search client

\*/

private AbderaClient abderaClient;

public void changeLogin(ActionEvent actionEvent) {

this.loginTextField.setDisable(false);

this.passwordTextField.setDisable(false);

this.loginButton.setDisable(false);

this.changeButton.setDisable(true);

this.sensingStartDate.setDisable(true);

this.sensingFinishDate.setDisable(true);

this.maxCloudPercentage.setDisable(true);

this.searchButton.setDisable(true);

}

private WebEngine webEngine;

/\*\*

\* Download form initialization

\*/

@FXML

void initialize(){

maxCloudPercentage.setTextFormatter(new TextFormatter<>(new NumberStringConverter()));

maxCloudPercentage.setText("100");

sensingStartDate.setValue(LocalDate.now());

sensingFinishDate.setValue(LocalDate.now());

final Callback<DatePicker, DateCell> dayCellFactory = (final DatePicker datePicker) -> new DateCell() {

@Override

public void updateItem(LocalDate item, boolean empty) {

super.updateItem(item, empty);

if (item.isAfter(LocalDate.now())) { //Disable all dates after required date

setDisable(true);

setStyle("-fx-background-color: #ffc0cb;"); //To set background on different color

}

}

};

sensingStartDate.setDayCellFactory(dayCellFactory);

sensingFinishDate.setDayCellFactory(dayCellFactory);

WebEngine webEngine = webMap.getEngine();

File mapIndexFile = new File("src/resources/SaDWebForm/index.html");

webEngine.load("file:" + mapIndexFile.getAbsolutePath());

this.webEngine = webEngine;

this.webEngine.setJavaScriptEnabled(true);

this.signInLink.setOnAction(e -> {

if (Desktop.isDesktopSupported()){

try {

Desktop.getDesktop().browse(new URI("https://scihub.copernicus.eu/dhus/#/self-registration"));

} catch (IOException | URISyntaxException e1) {

e1.printStackTrace();

}

}

});;

}

/\*\*

\* Check for login and password filling

\* @return empty or not

\*/

private boolean checkLoginAndPassword() {

return loginTextField.getText().length() != 0 && passwordTextField.getText().length() != 0;

}

/\*\*

\* Login acton handler

\* @param actionEvent login action event

\*/

public void loginHandler(ActionEvent actionEvent) throws URISyntaxException {

this.loginTextField.setDisable(true);

this.passwordTextField.setDisable(true);

if (!checkLoginAndPassword()) {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Empty login or password");

alert.setHeaderText("Error, empty login or password fields");

alert.setContentText("Please, fill empty field or sing up in https://scihub.copernicus.eu/");

alert.showAndWait();

this.loginTextField.setDisable(false);

this.passwordTextField.setDisable(false);

return;

}

String login = loginTextField.getText();

String password = passwordTextField.getText();

// Create OpenSearchConsumer

Abdera abdera = new Abdera();

this.abderaClient = new AbderaClient(abdera);

this.abderaClient.addCredentials(

ESA\_OPEN\_HUB\_PORTAL\_URL,

AuthScope.ANY\_REALM,

AuthScope.ANY\_SCHEME,

new UsernamePasswordCredentials(login, password)

);

this.loginButton.setDisable(true);

this.changeButton.setDisable(false);

this.sensingStartDate.setDisable(false);

this.sensingFinishDate.setDisable(false);

this.maxCloudPercentage.setDisable(false);

this.searchButton.setDisable(false);

}

public void searchDataHandler(ActionEvent actionEvent) throws URISyntaxException {

Instant sensingStartDate = null;

Instant sensingFinishDate = null;

if (!this.sensingStartDate.getEditor().getText().isEmpty()) {

LocalDate startDate = this.sensingStartDate.getValue();

sensingStartDate = Instant.from(startDate.atStartOfDay(ZoneId.systemDefault()));

}

if (!this.sensingFinishDate.getEditor().getText().isEmpty()) {

LocalDate finishDate = this.sensingFinishDate.getValue();

sensingFinishDate = Instant.from(finishDate.atStartOfDay(ZoneId.systemDefault()));

}

if (this.maxCloudPercentage.getText().isEmpty()) {

Utils.showErrorMessage("Max cloud percentage error",

"Max cloud percentage should be integer number between 0 and 100",

"");

return;

}

int maxCloudsPercentage = Integer.parseInt(maxCloudPercentage.getText());

if (maxCloudsPercentage < 0 || maxCloudsPercentage > 100) {

Utils.showErrorMessage("Max cloud percentage error",

"Max cloud percentage should be integer number between 0 and 100",

"");

return;

}

if (sensingStartDate != null && sensingFinishDate != null && sensingStartDate.isAfter(sensingFinishDate)) {

Utils.showErrorMessage("Error",

"Sensing start date must be before sensing finish date",

"");

return;

}

// Create query url

StringBuilder queryBuilder = new StringBuilder();

queryBuilder.append("platformname:Sentinel-2");

// Set up periods

if (sensingStartDate != null && sensingFinishDate != null) {

LocalDateTime start = LocalDateTime.ofInstant(sensingStartDate, ZoneId.systemDefault());

LocalDateTime finish = LocalDateTime.ofInstant(sensingFinishDate, ZoneId.systemDefault());

queryBuilder.append("%20AND%20")

.append("beginposition:%5B")

.append(start.getYear() + "-" + start.getMonthValue() + "-" + start.getDayOfMonth() + "T00:00:00.000Z")

.append("%20TO%20")

.append(finish.getYear() + "-" + finish.getMonthValue() + "-" + finish.getDayOfMonth() + "T23:59:59.000Z")

.append("%5D");

}

// Coverage intersection

String geometry = (String) this.webEngine.executeScript("getGeometry();");

if (geometry != null && !geometry.equals("") && !geometry.equals("undefined")) {

queryBuilder.append("%20AND%20footprint%3A%22Intersects%28")

.append(geometry

.replace(" ", "%20")

.replace(",", "%2C")

.replace("(", "%28")

.replace(")", "%29")

)

.append("%29%22");

}

// Set up clouds percentage

if (maxCloudsPercentage != 100) {

queryBuilder.append("%20AND%20cloudcoverpercentage%3A%5B0%20TO%20").append(maxCloudsPercentage).append("%5D");

}

// Create open search query

System.out.println(geometry);

ClientResponse response = this.abderaClient.get(OPEN\_SEARCH\_QUERY\_BASE + queryBuilder.toString());

List<Entry> entries = null;

if (response.getType() == Response.ResponseType.SUCCESS) {

Document<Feed> doc = response.getDocument();

Feed feed = doc.getRoot();

entries = feed.getEntries();

} else {

Utils.showErrorMessage("Error", "Open Search error", response.getType().toString());

return;

}

if (resultListView.getItems().size() > 0) {

resultListView.getItems().clear();

}

if (entries == null || entries.size() == 0) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Result");

alert.setHeaderText("Data not found");

alert.setContentText("Please, change request params");

alert.showAndWait();

} else {

ObservableList<Entry> e = FXCollections.observableArrayList();

e.addAll(entries);

resultListView.setCellFactory(c -> new SentinelDataResponse());

resultListView.setItems(e);

tab.getSelectionModel().select(1);

}

}

private class SentinelDataResponse extends ListCell<Entry> {

private BorderPane content = new BorderPane();

Label title = new Label();

Label summary = new Label();

Hyperlink link = new Hyperlink();

ImageView imageView = new ImageView();

public SentinelDataResponse() {

VBox vBox = new VBox();

vBox.setSpacing(4);

vBox.getChildren().addAll(title, link, summary);

content.setLeft(imageView);

content.setCenter(vBox);

//content.getChildren().addAll(imageView, title, link, summary);

setContentDisplay(ContentDisplay.GRAPHIC\_ONLY);

setGraphic(content);

}

@Override

public void updateItem(Entry entry, boolean empty) {

super.updateItem(entry, empty);

if (entry == null || empty) {

setText(null);

setGraphic(null);

} else {

title.setText(entry.getTitle());

link.setText(entry.getAlternateLink().getHref().toString());

summary.setText(entry.getSummary());

try {

URLConnection uc = new java.net.URL(entry.getLink("icon").getHref().toString()).openConnection();

String userpass = "artur7" + ":" + "9063228328a!";

String basicAuth = "Basic " + new String(Base64.getEncoder().encode(userpass.getBytes()));

uc.setRequestProperty ("Authorization", basicAuth);

InputStream in = uc.getInputStream();

Image image = new Image(in, 120, 120, false, false);

imageView.setImage(image);

} catch (IOException e) {

e.printStackTrace();

}

link.setOnAction(e -> {

if (Desktop.isDesktopSupported()){

try {

Desktop.getDesktop().browse(new URI(link.getText()));

} catch (IOException | URISyntaxException e1) {

e1.printStackTrace();

}

}

});

setGraphic(content);

}

}

}

}

## TrainNextGisForm

package LandUseChangeDetection.forms;

import LandUseChangeDetection.\*;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.concurrent.Task;

import javafx.event.ActionEvent;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.layout.AnchorPane;

import javafx.stage.DirectoryChooser;

import javafx.stage.FileChooser;

import javafx.stage.Stage;

import org.esa.s2tbx.dataio.VirtualPath;

import java.io.File;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.StandardCopyOption;

import java.util.Arrays;

import java.util.List;

import java.util.Objects;

import java.util.Observable;

import java.util.stream.Collectors;

public class TrainNextGisForm {

/\*\*

\* Learn SVM by NextGIS data anchor pane

\*/

public AnchorPane learnNextGISForm;

public Label esaDataLabel;

public Button selectGranuleButton;

public ComboBox granuleSelectionBox;

public ComboBox resolutionBox;

public Button selectTrainingVectorButton;

public Label nextGisVectorFileLabel;

public Button trainButton;

public Button exportAButton;

public Button exportBButton;

public Button cancelButton;

public Button importAButton;

public Button importBButton;

public Button trainingSentinelDataChooser;

/\*\*

\* Training sentinel file

\*/

private File sentinel2ALevelFile = null;

/\*\*

\* Training granule

\*/

private VirtualPath granulePath = null;

public void initialize() {

this.resolutionBox.setItems(resolutions);

this.resolutionBox.setValue(resolutions.get(0));

}

/\*\*

\* Selection of Sentinel training files

\* @param actionEvent Selection action event

\*/

public void selectSentinelTrainingData(ActionEvent actionEvent) {

FileChooser dc = new FileChooser();

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("XML files (\*.xml)", "\*.xml");

dc.setSelectedExtensionFilter(filter);

this.sentinel2ALevelFile = dc.showOpenDialog(learnNextGISForm.getScene().getWindow());

if (this.sentinel2ALevelFile == null) {

esaDataLabel.setText("Data not selected");

selectGranuleButton.setDisable(true);

} else {

esaDataLabel.setText(this.sentinel2ALevelFile.getAbsolutePath());

selectGranuleButton.setDisable(false);

}

}

/\*\*

\* Training shapefile

\*/

private File trainingShapefile;

/\*\*

\* Training shapefile selection

\* @param actionEvent action event

\*/

public void selectTrainingVectorFile(ActionEvent actionEvent) {

DirectoryChooser dc = new DirectoryChooser();

this.trainingShapefile = dc.showDialog(learnNextGISForm.getScene().getWindow());

if (this.trainingShapefile == null) {

nextGisVectorFileLabel.setText("Data not selected");

trainButton.setDisable(true);

} else {

nextGisVectorFileLabel.setText(this.trainingShapefile.getAbsolutePath());

trainButton.setDisable(false);

}

}

/\*\*

\* Train SVM model

\* @param actionEvent training action event

\*/

public void trainSVMModel(ActionEvent actionEvent) {

if (this.sentinel2ALevelFile == null || this.trainingShapefile == null) {

Utils.showErrorMessage("Error",

"Please, choose files for training",

"");

return;

}

File granule = new File(granuleSelectionBox.getValue().toString());

ClassificationEnum type;

if (sentinel2ALevelFile.getParentFile().getName().startsWith("S2A")) {

type = ClassificationEnum.A;

} else if (sentinel2ALevelFile.getParentFile().getName().startsWith("S2B")) {

type = ClassificationEnum.B;

} else {

Utils.showErrorMessage("Error",

"Plese select valid Sentinel file",

sentinel2ALevelFile.getParentFile().getName());

return;

}

Resolution resolution;

if (resolutionBox.getValue().equals("60m")) {

resolution = Resolution.R60m;

} else {

resolution = Resolution.R20m;

}

Classification svm = Classification.getInstance(type);

Task task = new Task() {

@Override

protected Object call() {

try {

trainingSentinelDataChooser.setDisable(true);

selectGranuleButton.setDisable(true);

granuleSelectionBox.setDisable(true);

resolutionBox.setDisable(true);

selectTrainingVectorButton.setDisable(true);

importAButton.setDisable(true);

importBButton.setDisable(true);

exportAButton.setDisable(true);

exportBButton.setDisable(true);

trainButton.setDisable(true);

cancelButton.setDisable(false);

svm.trainByNextGISData(trainingShapefile, granule, type, resolution);

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Success");

alert.setHeaderText("Training process finished successfully");

alert.showAndWait();

} catch (Exception e) {

Utils.showErrorMessage("Error",

e.getMessage(),

Arrays.toString(e.getStackTrace()));

}

return null;

}

};

this.cancelButton.setOnMouseClicked(event -> {

task.cancel();

trainingSentinelDataChooser.setDisable(false);

selectGranuleButton.setDisable(false);

granuleSelectionBox.setDisable(false);

resolutionBox.setDisable(false);

selectTrainingVectorButton.setDisable(false);

cancelButton.setDisable(true);

importAButton.setDisable(false);

importBButton.setDisable(false);

exportAButton.setDisable(false);

exportBButton.setDisable(false);

trainButton.setDisable(false);

});

((Stage)learnNextGISForm.getScene().getWindow()).setOnHiding(event -> task.cancel());

new Thread(task).start();

}

/\*\*

\* Resolutions list

\*/

private final static ObservableList<String> resolutions = FXCollections.observableArrayList("60m", "20m");

/\*\*

\* Selection granules handler

\* @param actionEvent granule selection action handler

\*/

public void selectGranuleButtonHandler(ActionEvent actionEvent) {

if (this.sentinel2ALevelFile == null) {

Utils.showErrorMessage("Sentinel 2 Data Error",

"Error, Sentinel data not selected",

"");

return;

}

try {

List<VirtualPath> granules = SentinelData.checkAndGetGranules(this.sentinel2ALevelFile);

ObservableList<String> granulesList = FXCollections.observableArrayList(

granules.stream().map(VirtualPath::getFullPathString)

.collect(Collectors.toList()));

granuleSelectionBox.setItems(granulesList);

granuleSelectionBox.setValue(granulesList.get(0));

granuleSelectionBox.setDisable(false);

resolutionBox.setDisable(false);

selectTrainingVectorButton.setDisable(false);

} catch (Exception e) {

Utils.showErrorMessage("Granules extracting error",

"Error, granules extracting ",

e.getMessage());

}

}

public void exportAHandler(ActionEvent actionEvent) {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Export SVM A Model");

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("SVM files (\*.svm)", "\*.svm");

fileChooser.setSelectedExtensionFilter(filter);

File file = fileChooser.showSaveDialog(learnNextGISForm.getScene().getWindow());

if (file != null) {

try {

Files.copy(Classification.svmModelPath.toPath(), file.toPath(), StandardCopyOption.REPLACE\_EXISTING);

} catch (IOException e) {

Utils.showErrorMessage("Exporting error",

"Cannot export SVM A Model",

Arrays.toString(e.getStackTrace()));

}

}

}

public void exportBHandler(ActionEvent actionEvent) {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Export SVM B Model");

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("SVM files (\*.svm)", "\*.svm");

fileChooser.setSelectedExtensionFilter(filter);

File file = fileChooser.showSaveDialog(learnNextGISForm.getScene().getWindow());

if (file != null) {

try {

Files.copy(Classification.svmBModelPath.toPath(), file.toPath(), StandardCopyOption.REPLACE\_EXISTING);

} catch (IOException e) {

Utils.showErrorMessage("Exporting error",

"Cannot export SVM B Model",

Arrays.toString(e.getStackTrace()));

}

}

}

public void importAHandler(ActionEvent actionEvent) {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Import SVM A Model");

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("SVM files (\*.svm)", "\*.svm");

fileChooser.setSelectedExtensionFilter(filter);

File file = fileChooser.showOpenDialog(learnNextGISForm.getScene().getWindow());

if (file != null) {

try {

Files.copy(file.toPath(), Classification.svmModelPath.toPath(), StandardCopyOption.REPLACE\_EXISTING);

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Import");

alert.setHeaderText("SVM A model imported successfully. Please, restart program for correct work");

alert.showAndWait();

} catch (IOException e) {

Utils.showErrorMessage("Importing error",

"Cannot import SVM A Model",

Arrays.toString(e.getStackTrace()));

}

}

}

public void importBHandler(ActionEvent actionEvent) {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Import SVM B Model");

FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter("SVM files (\*.svm)", "\*.svm");

fileChooser.setSelectedExtensionFilter(filter);

File file = fileChooser.showOpenDialog(learnNextGISForm.getScene().getWindow());

if (file != null) {

try {

Files.copy(file.toPath(), Classification.svmBModelPath.toPath(), StandardCopyOption.REPLACE\_EXISTING);

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Import");

alert.setHeaderText("SVM B model imported successfully. Please, restart program for correct work");

alert.showAndWait();

} catch (IOException e) {

Utils.showErrorMessage("Exporting error",

"Cannot import SVM B Model",

Arrays.toString(e.getStackTrace()));

}

}

}

}

## AppWebForm.index.html

<!DOCTYPE html>

<html lang="en">

<head>

<title>Lukin Artur Homework</title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no"/>

<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>

<link rel="stylesheet" href="http://code.jquery.com/ui/1.11.1/themes/smoothness/jquery-ui.css">

<link rel="stylesheet" href="http://libs.cartocdn.com/cartodb.js/v3/3.15/themes/css/cartodb.css"/>

<link rel="stylesheet" href="http://jqueryui.com/resources/demos/style.css">

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.3.1/dist/leaflet.css" />

<!--<link rel="stylesheet" href="styles/leaflet.draw.css"/>-->

<style>

html, body {

height: 100%;

padding: 0;

margin: 0;

}

#map {

height: 100%;

padding: 0;

margin: 0;

}

#dashboard {

height: 33%;

padding: 25px;

margin: 0;

border-top: 2px solid #333;

}

#dash {

margin: 25px 0;

width: 400px;

}

button {

margin-right: 10px;

font-family: "Proxima Nova W01", "Helvetica Neue", Helvetica, Arial, sans-serif;

background: transparent;

}

p {

font-size: 17px;

margin-bottom: 10px;

}

</style>

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>

<script src="http://code.jquery.com/ui/1.11.1/jquery-ui.js"></script>

<script type="text/javascript" src="http://fast.fonts.net/jsapi/b6470a43-5105-49d3-95ae-75217a436bf8.js"></script>

</head>

<body>

<div id="map"></div>

<script src="https://unpkg.com/leaflet@1.3.1/dist/leaflet.js"></script>

<script src="scripts/leaflet.ajax.min.js"></script>

<!--<script src="scripts/leaflet.js"></script>-->

<!--<script src="http://libs.cartocdn.com/cartodb.js/v3/3.15/cartodb.js"></script>-->

<!--<script src="https://unpkg.com/leaflet@1.3.1/dist/leaflet.js"></script>-->

<!--<script src="scripts/leaflet.draw.js"></script>-->

<!--<script src="scripts/Leaflet.draw.drag.js"></script>-->

<!--<script src="scripts/jsts.min.js"></script>-->

<!--<script src="scripts/leaflet.ajax.min.js"></script>-->

<!--<script src="scripts/terraformer.min.js"></script>-->

<!--<script src="scripts/terraformer-wkt-parser.js"></script>-->

<!--<script src="scripts/ActiveLayers.js"></script>-->

<!--<script src="scripts/SelectLayers.js"></script>-->

<script language="JavaScript">

// Map styles

var OpenTopoMap = L.tileLayer('https://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {

maxZoom: 17,

attribution: 'Map data: &copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>, <a href="http://viewfinderpanoramas.org">SRTM</a> | Map style: &copy; <a href="https://opentopomap.org">OpenTopoMap</a> (<a href="https://creativecommons.org/licenses/by-sa/3.0/">CC-BY-SA</a>)'

});

// Choose center and zoom level

var options = {

center: [55.792191, 49.113994], // Kazan

zoom: 15,

layers: OpenTopoMap

};

// Instantiate map on specified DOM element

var map\_object = new L.Map('map', options);

var geoJsonLayer;

function showResult() {

if (geoJsonLayer !== undefined) {

map\_object.removeLayer(geoJsonLayer);

}

geoJsonLayer = new L.geoJson();

geoJsonLayer.addTo(map\_object);

$.ajax({

dataType: "json",

url: "res/result.json",

success: function(data) {

$(data.features).each(function(key, data) {

geoJsonLayer.addData(data);

});

geoJsonLayer.eachLayer(function (data) {

switch (data.feature.properties.before) {

case 0: {

switch (data.feature.properties.after) {

case 0: {

data.setStyle({

color: '#1e90ff',

fillOpacity: 1,

fillColor: '#1e90ff'

});

break;

}

case 1: {

data.setStyle({

color: '#b1e1ac',

fillOpacity: 1,

fillColor: '#b1e1ac'

});

break;

}

case 2: {

data.setStyle({

color: '#bee7ff',

fillOpacity: 1,

fillColor: '#bee7ff'

});

break;

}

case 3: {

data.setStyle({

color: '#7f4e33',

fillOpacity: 1,

fillColor: '#7f4e33'

});

break;

}

}

break;

}

case 1: {

switch (data.feature.properties.after) {

case 0: {

data.setStyle({

color: '#00f4dd',

fillOpacity: 1,

fillColor: '#00f4dd'

});

break;

}

case 1: {

data.setStyle({

color: '#1fff28',

fillOpacity: 1,

fillColor: '#1fff28'

});

break;

}

case 2: {

data.setStyle({

color: '#6e0793',

fillOpacity: 1,

fillColor: '#6e0793'

});

break;

}

case 3: {

data.setStyle({

color: '#f40000',

fillOpacity: 1,

fillColor: '#f40000'

});

break;

}

}

break;

}

case 2: {

switch (data.feature.properties.after) {

case 0: {

data.setStyle({

color: '#0f2b90',

fillOpacity: 1,

fillColor: '#0f2b90'

});

break;

}

case 1: {

data.setStyle({

color: '#a4ff09',

fillOpacity: 1,

fillColor: '#a4ff09'

});

break;

}

case 2: {

data.setStyle({

color: '#afb9c3',

fillOpacity: 1,

fillColor: '#afb9c3'

});

break;

}

case 3: {

data.setStyle({

color: '#ff8e00',

fillOpacity: 1,

fillColor: '#ff8e00'

});

break;

}

}

break;

}

case 3: {

switch (data.feature.properties.after) {

case 0: {

data.setStyle({

color: '#712bc2',

fillOpacity: 1,

fillColor: '#712bc2'

});

break;

}

case 1: {

data.setStyle({

color: '#2b9722',

fillOpacity: 1,

fillColor: '#2b9722'

});

break;

}

case 2: {

data.setStyle({

color: '#ff00e9',

fillOpacity: 1,

fillColor: '#ff00e9'

});

break;

}

case 3: {

data.setStyle({

color: '#144616',

fillOpacity: 1,

fillColor: '#144616'

});

break;

}

}

break;

}

}

});

map\_object.flyToBounds(geoJsonLayer.getBounds(), 11);

}

}).error(function() {});

}

</script>

</body>

</html>

## SaDWebForm.index.html

<!DOCTYPE html>

<html lang="en">

<head>

<title>Lukin Artur Homework</title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no"/>

<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>

<link rel="stylesheet" href="http://code.jquery.com/ui/1.11.1/themes/smoothness/jquery-ui.css">

<link rel="stylesheet" href="http://libs.cartocdn.com/cartodb.js/v3/3.15/themes/css/cartodb.css"/>

<link rel="stylesheet" href="http://jqueryui.com/resources/demos/style.css">

<link rel="stylesheet" href="styles/leaflet.draw.css"/>

<style>

html, body {

height: 100%;

padding: 0;

margin: 0;

}

#map {

height: 100%;

padding: 0;

margin: 0;

}

#dashboard {

height: 33%;

padding: 25px;

margin: 0;

border-top: 2px solid #333;

}

#dash {

margin: 25px 0;

width: 400px;

}

button {

margin-right: 10px;

font-family: "Proxima Nova W01", "Helvetica Neue", Helvetica, Arial, sans-serif;

background: transparent;

}

p {

font-size: 17px;

margin-bottom: 10px;

}

</style>

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>

<script src="http://code.jquery.com/ui/1.11.1/jquery-ui.js"></script>

<script type="text/javascript" src="http://fast.fonts.net/jsapi/b6470a43-5105-49d3-95ae-75217a436bf8.js"></script>

</head>

<body>

<div id="map"></div>

<script src="http://libs.cartocdn.com/cartodb.js/v3/3.15/cartodb.js"></script>

<script src="scripts/leaflet.draw.js"></script>

<script src="scripts/Leaflet.draw.drag.js"></script>

<script src="scripts/jsts.min.js"></script>

<script src="scripts/leaflet.ajax.min.js"></script>

<script src="scripts/terraformer.min.js"></script>

<script src="scripts/terraformer-wkt-parser.js"></script>

<script src="scripts/ActiveLayers.js"></script>

<script src="scripts/SelectLayers.js"></script>

<script>

// Map styles

var OpenTopoMap = L.tileLayer('https://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {

maxZoom: 17,

attribution: 'Map data: &copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>, <a href="http://viewfinderpanoramas.org">SRTM</a> | Map style: &copy; <a href="https://opentopomap.org">OpenTopoMap</a> (<a href="https://creativecommons.org/licenses/by-sa/3.0/">CC-BY-SA</a>)'

});

// Choose center and zoom level

var options = {

center: [55.792191, 49.113994], // Kazan

zoom: 5,

layers: OpenTopoMap

};

// Instantiate map on specified DOM element

var map\_object = new L.Map('map', options);

// Draw geometries

var drawnItems = new L.FeatureGroup();

map\_object.addLayer(drawnItems);

// Draw control

var drawControl = new L.Control.Draw({

draw: {

position: 'topleft',

polygon: false,

polyline: false,

rectangle: {

shapeOptions: {

color: '#9100ff',

opacity: 1

}

},

circle: false,

marker: false

},

edit: {

featureGroup: drawnItems

}

});

map\_object.addControl(drawControl);

var geometry;

// Select geometry

map\_object.on('draw:created', function (e) {

drawnItems.eachLayer(function (layer) {

drawnItems.removeLayer(layer);

});

drawnItems.addLayer(e.layer);

geometry = Terraformer.WKT.convert(e.layer.toGeoJSON().geometry);

});

// Edit geometry

map\_object.on('draw:edited', function () {

geometry = '';

});

// Edit geometry

map\_object.on('draw:deleted', function () {

geometry = '';

});

function getGeometry() {

return geometry;

}

</script>

</body>

</html>

# **Список использованной литературы**

1. European Space Agency [Электронный ресурс] // ESA: [сайт]. [2018]. URL: https://www.esa.int/ESA (дата обращения: 10.01.2018).
2. Open Access Hub [Электронный ресурс] // ESA: [сайт]. [2018]. URL: https://scihub.copernicus.eu/twiki/do/view/SciHubWebPortal/APIHubDescription (дата обращения: 10.02.2018).
3. Sentinel 2 Product Types [Электронный ресурс] // ESA: [сайт]. [2018]. URL: https://earth.esa.int/web/sentinel/user-guides/sentinel-2-msi/product-types (дата обращения: 10.02.2018).
4. Open Street Map Tags [Электронный ресурс] // Open Street Map: [сайт]. [2018]. URL: https://wiki.openstreetmap.org/wiki/Tags (дата обращения: 10.02.2018).
5. Windows [Электронный ресурс] // Mocrosoft: [сайт]. [2018]. URL: https://www.microsoft.com/ru-ru/windows (дата обращения: 05.05.2018).
6. Java SE Development KIT [Электронный ресурс] // Oracle: [сайт]. [2018]. URL: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html (дата обращения: 05.05.2018).
7. Python:Anaconda [Электронный ресурс] // Anaconda: [сайт]. [2018]. URL: https://anaconda.org/anaconda/python (дата обращения: 05.05.2018).
8. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ Р 7.02-2006 Консервация документов на компакт-дисках. Общие требования. – М.: ИПК Издательство стандартов, 2006
10. ГОСТ 18300-87 Спирт этиловый ректификованный технический. Технические условия. – М.:ИПК Издательство стандартов, 1997
11. ГОСТ 9805-84. Спирт изопропиловый. Технические условия. – М.: ИПК Издательство стандартов, 1984.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ** | | | | | | | | | |
| Изм. | Номера листов (страниц) | | | | Всего  листов  (страниц)  в докум. | № докум. | Входящий № сопроводитель-ного документа и дата | Подпись | Дата |
| изме-ненных | заме-  ненных | новых | анну-  лиро-  ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |