CS5542 Big Data Apps and Analytics
LAB ASSIGNMENT #6

1.**Spark Programming:**

The spark programming is run on a video containing wildlife imagery. The data set contains the images of different wild animals, possibly which are present in the video. The first program is feature extraction/ Object detection. The main features are extracted from the video with training images provided. The images are of different classes or categories. For this lab, I have taken images from four categories: Elephant, Gazelle, Rhino, Zebra.

These are the images that are used for training :

Now the program is run on these images and the video and here are the screens:


Image
[41,31] [0.85490197, 0.8039216, 0.79607844]

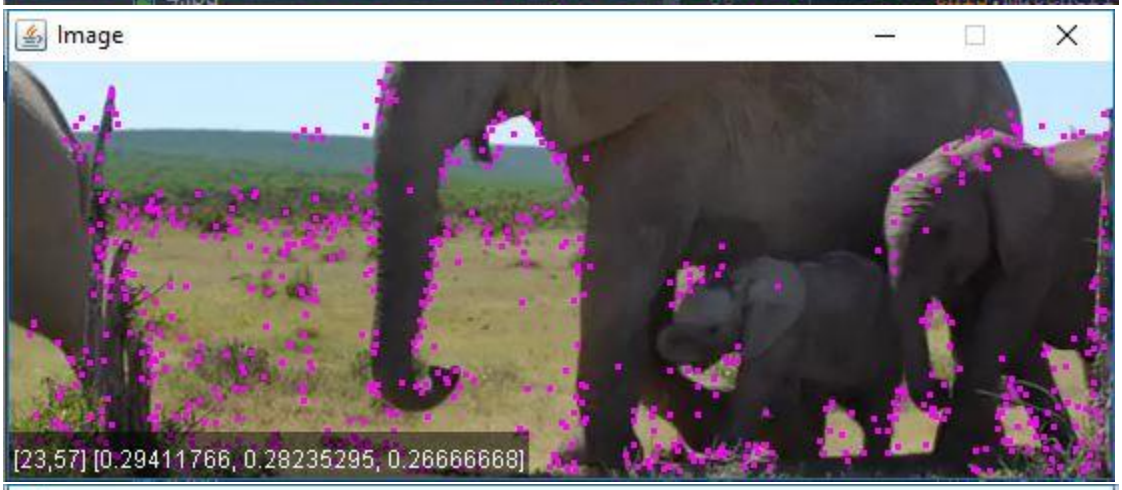


Image
[23,57] [0.29411766, 0.28235295, 0.26666668]


Image
[28,35] [0.16470589, 0.2901961, 0.20784314]

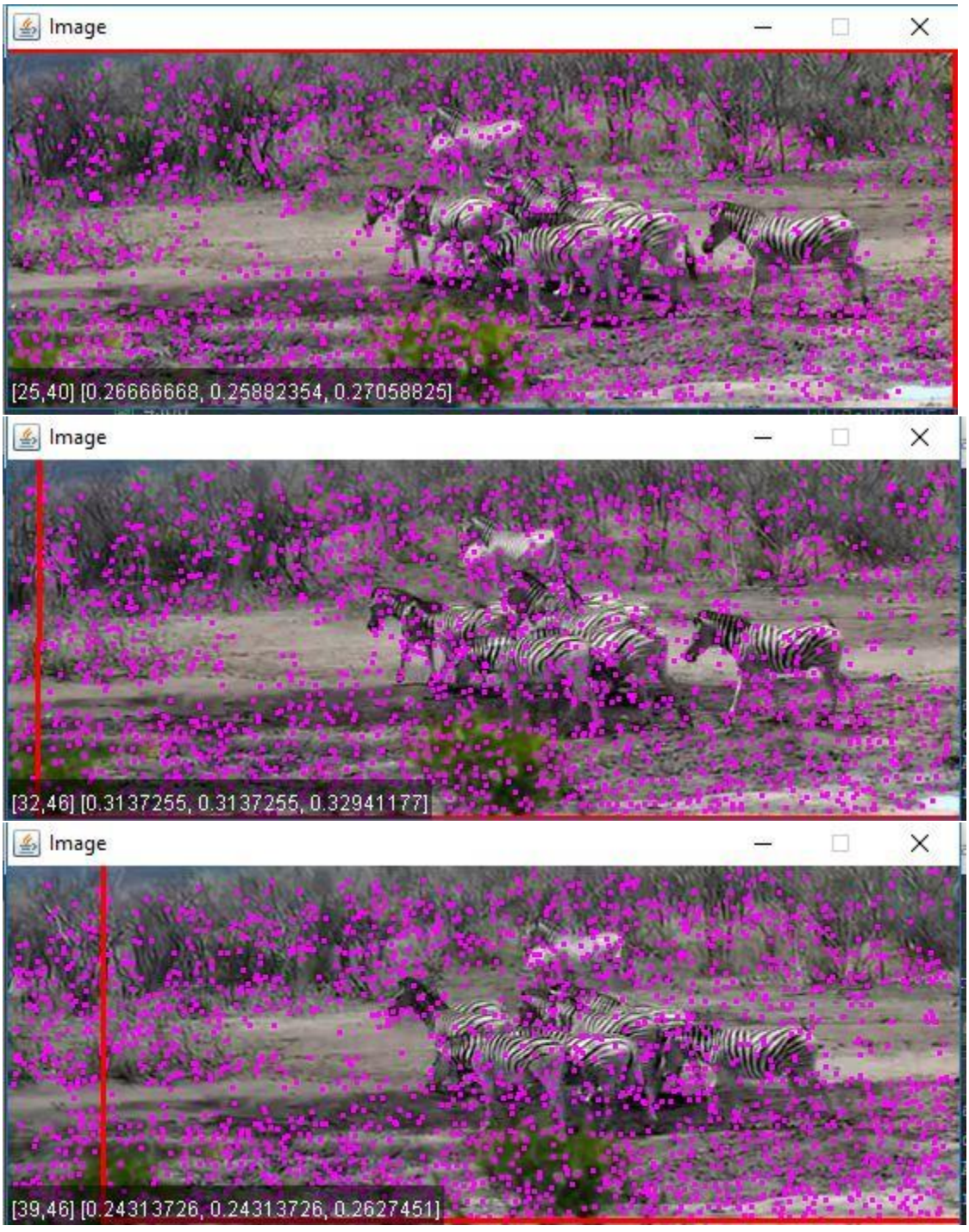


Image
[25,40] [0.26666668, 0.25882354, 0.27058825]
Image
[32,46] [0.3137255, 0.3137255, 0.32941177]
Image
[39,46] [0.24313726, 0.24313726, 0.2627451]

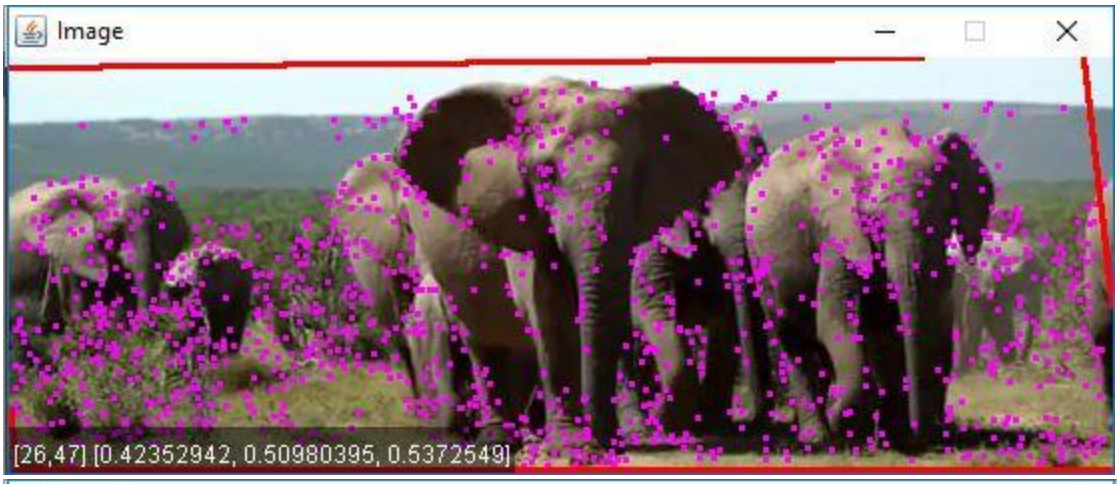


Image
[26,47] [0.42352942, 0.50980395, 0.5372549]

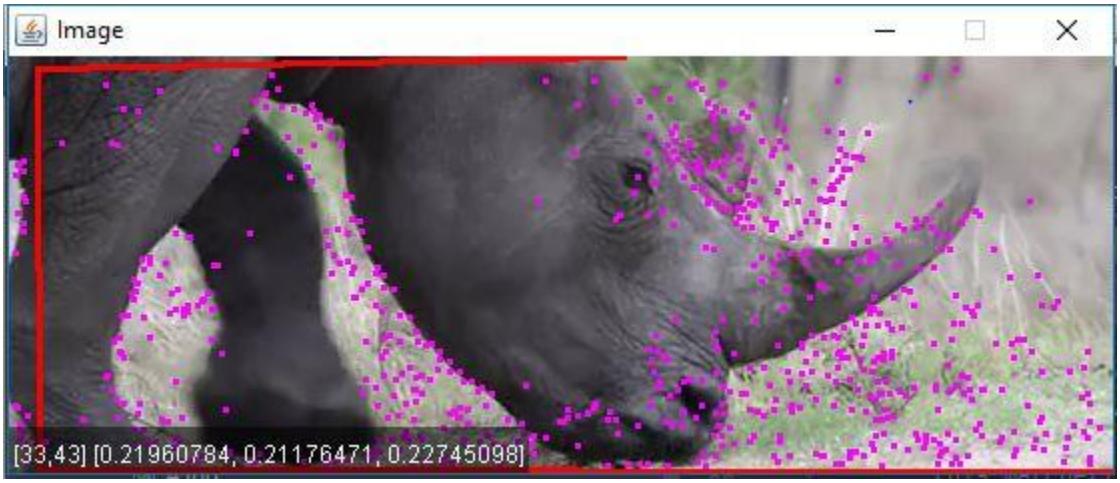


Image
[33,43] [0.21960784, 0.21176471, 0.22745098]

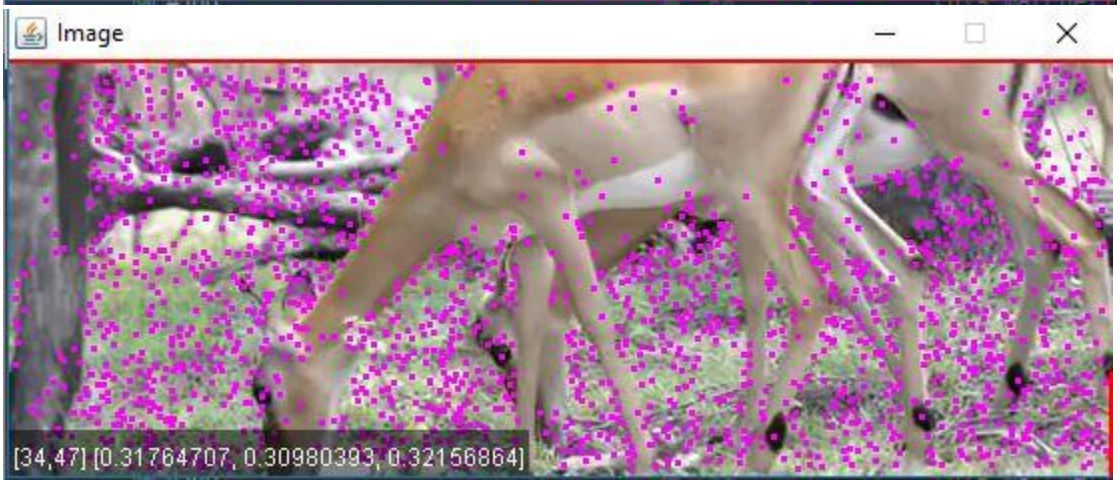
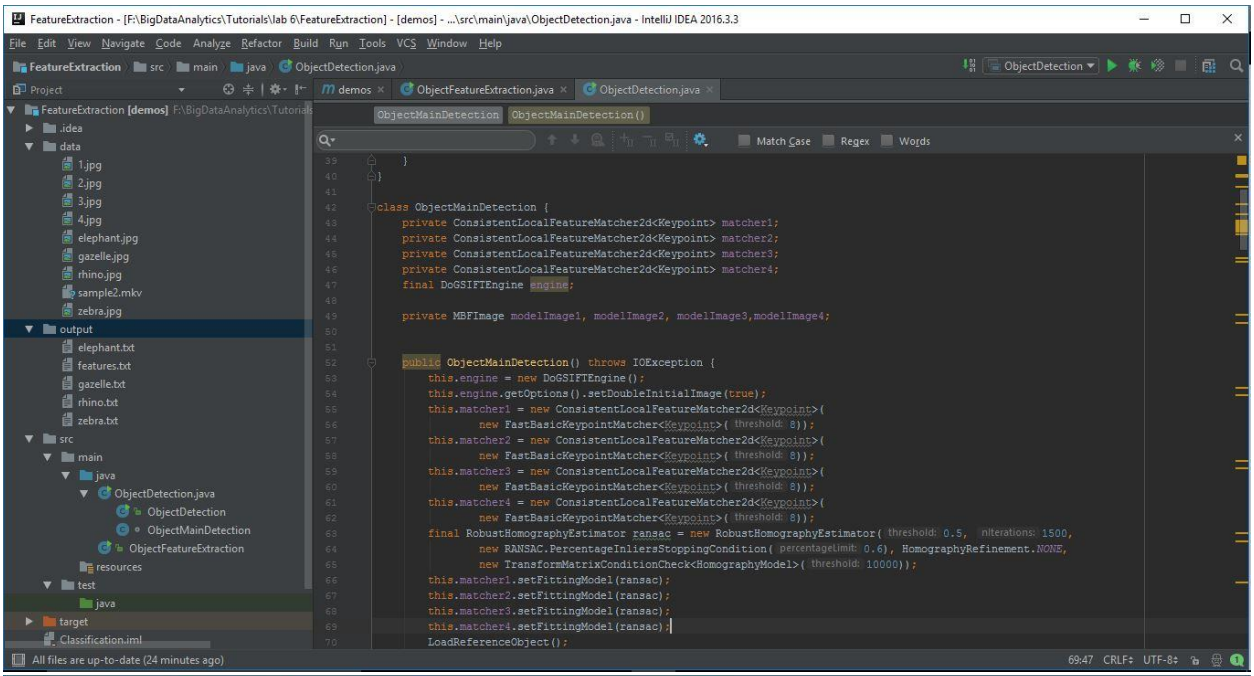
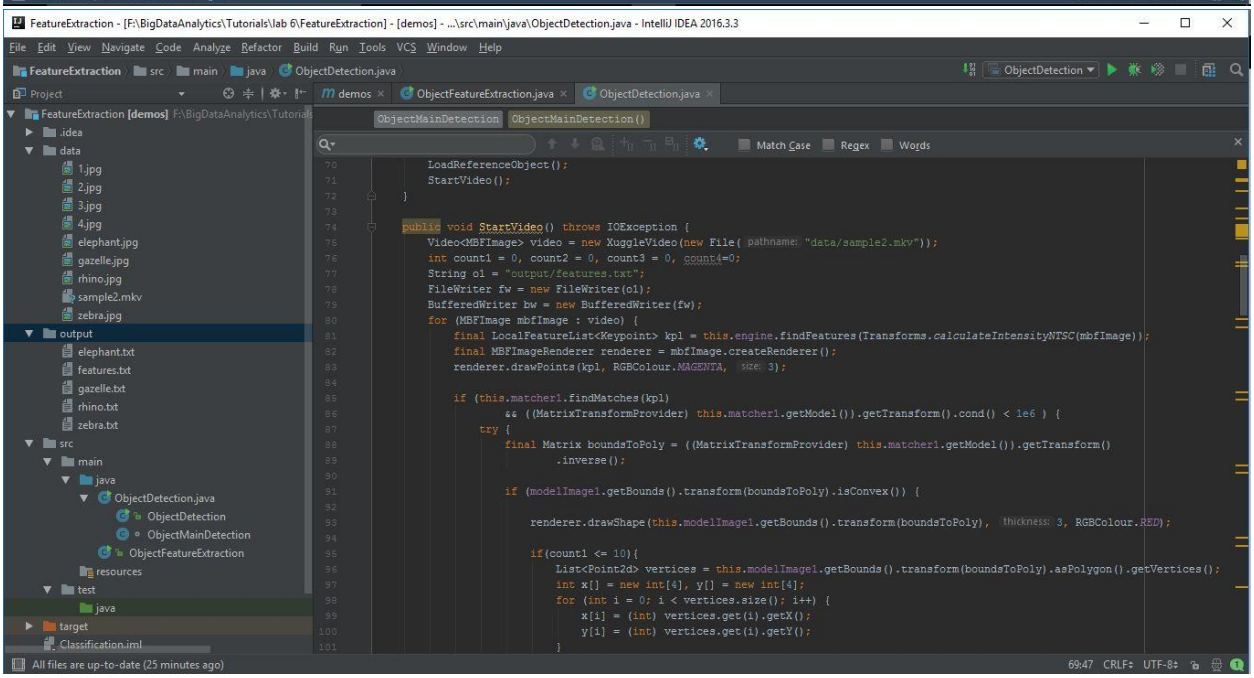
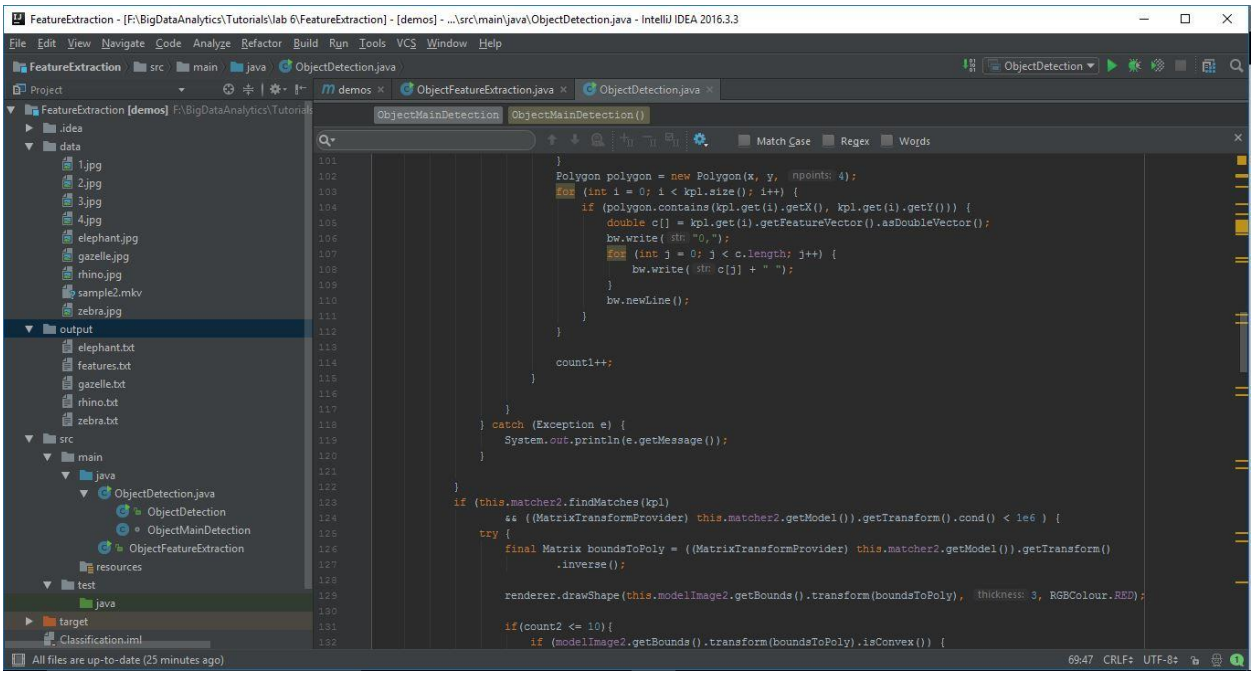
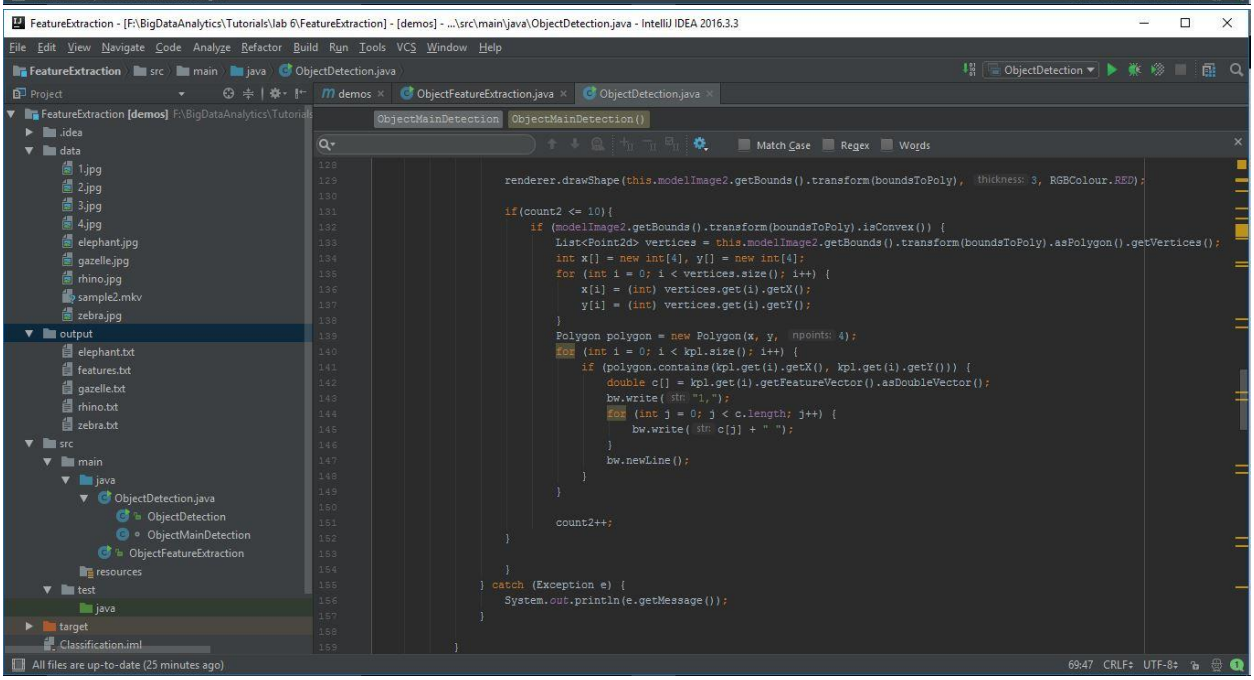


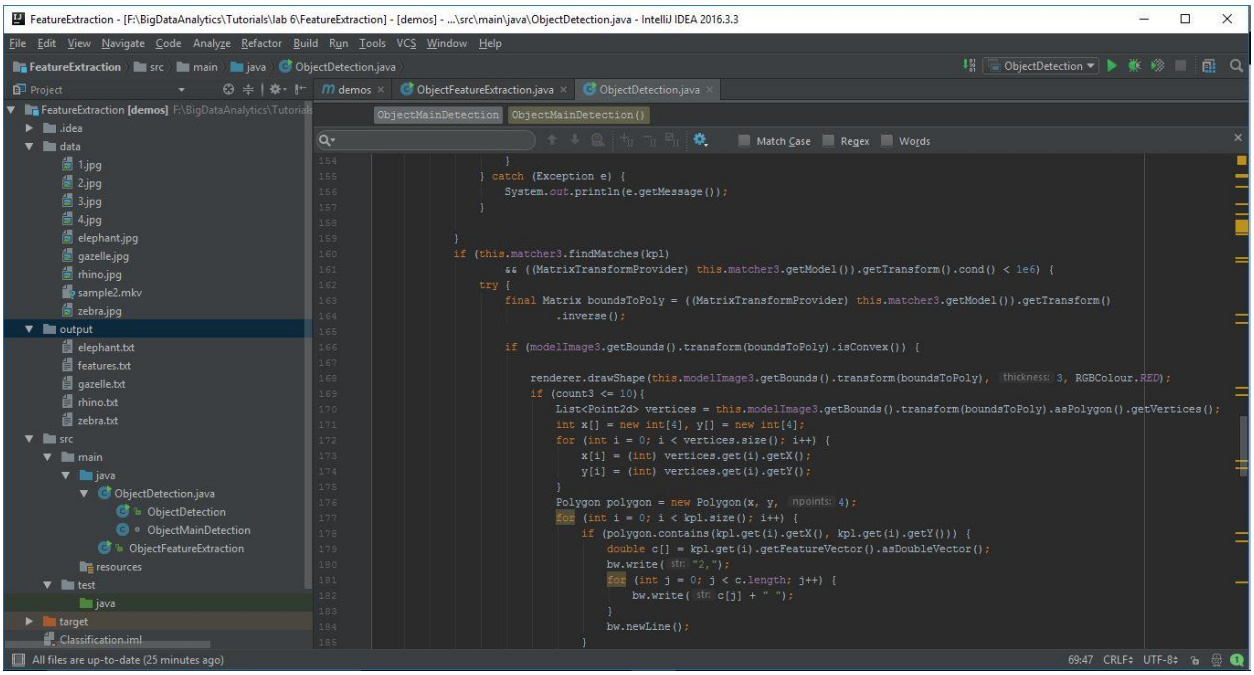
Image
[34,47] [0.31764707, 0.30980393, 0.32156864]

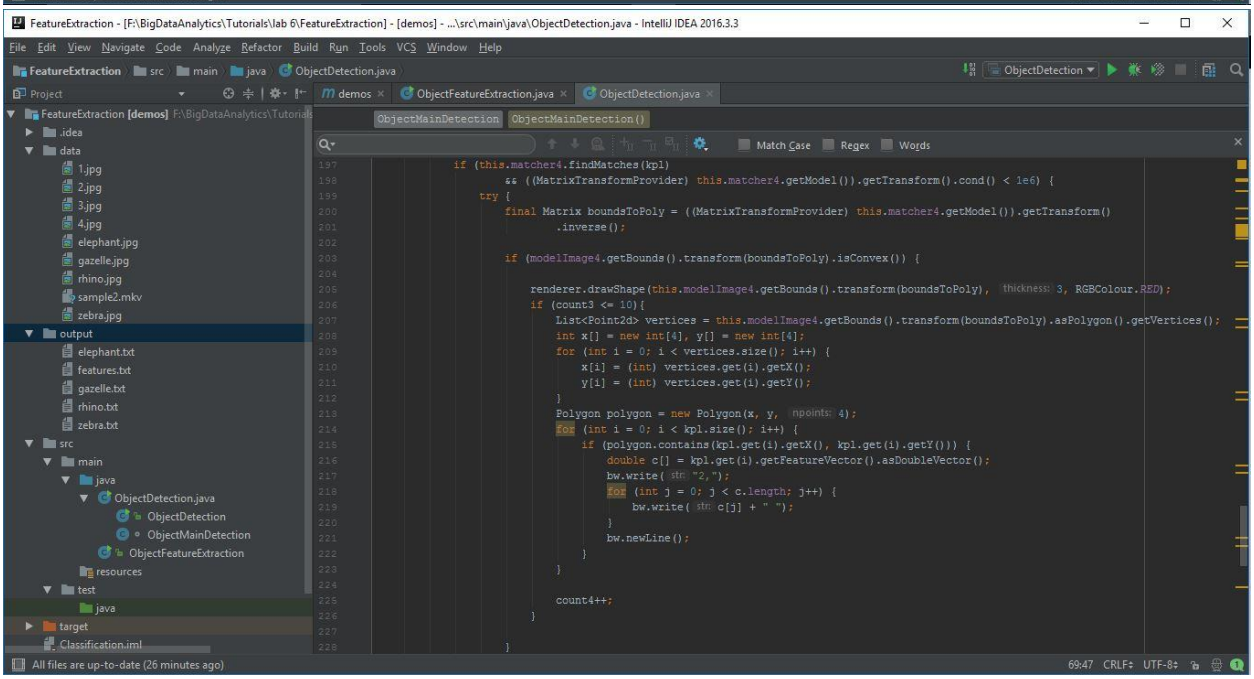

FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\src\main\java\ObjectDetection.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ObjectDetection
demos
ObjectFeatureExtraction.java
ObjectDetection.java
ObjectMainDetection
ObjectMainDetection()
Match Case
Regex
Words
FeatureExtraction [demos] F:\BigDataAnalytics\Tutorials
.idea
data
1.jpg
2.jpg
3.jpg
4.jpg
elephant.jpg
gazelle.jpg
rhino.jpg
sample2.mkv
zebra.jpg
output
elephant.txt
features.txt
gazelle.txt
rhino.txt
zebra.txt
src
main
java
ObjectDetection.java
ObjectDetection
ObjectMainDetection
ObjectFeatureExtraction
resources
test
java
target
Classification.iml
class ObjectMainDetection {
private ConsistentLocalFeatureMatcher2d<Keypoint> matcher1;
private ConsistentLocalFeatureMatcher2d<Keypoint> matcher2;
private ConsistentLocalFeatureMatcher2d<Keypoint> matcher3;
private ConsistentLocalFeatureMatcher2d<Keypoint> matcher4;
final DoGSIFTEngine engine;
private MBFImage modelImage1, modelImage2, modelImage3,modelImage4;
public ObjectMainDetection() throws IOException {
this.engine = new DoGSIFTEngine();
this.engine.getOptions().setDoubleInitialImage(true);
this.matcher1 = new ConsistentLocalFeatureMatcher2d<Keypoint>(
new FastBasicKeypointMatcher<Keypoint>(threshold: 8));
this.matcher2 = new ConsistentLocalFeatureMatcher2d<Keypoint>(
new FastBasicKeypointMatcher<Keypoint>(threshold: 8));
this.matcher3 = new ConsistentLocalFeatureMatcher2d<Keypoint>(
new FastBasicKeypointMatcher<Keypoint>(threshold: 8));
this.matcher4 = new ConsistentLocalFeatureMatcher2d<Keypoint>(
new FastBasicKeypointMatcher<Keypoint>(threshold: 8));
final RobustHomographyEstimator ransac = new RobustHomographyEstimator(threshold: 0.5, nIterations: 1500,
new RANSAC.PercentageInliersStoppingCondition(percentageLimit: 0.6), HomographyRefinement.NONE,
new TransformMatrixConditionCheck<HomographyModel>(threshold: 10000));
this.matcher1.setFittingModel(ransac);
this.matcher2.setFittingModel(ransac);
this.matcher3.setFittingModel(ransac);
this.matcher4.setFittingModel(ransac);
LoadReferenceObject();
All files are up-to-date (24 minutes ago)
69:47 CRLF UTF-8

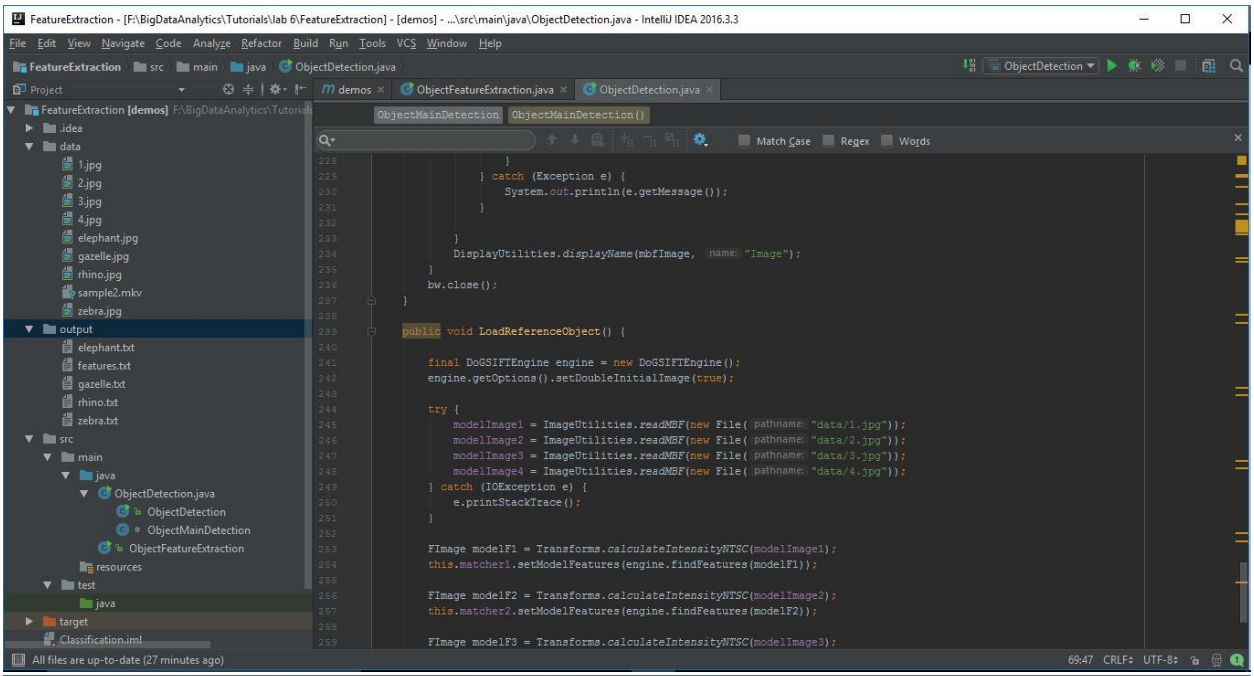

FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\src\main\java\ObjectDetection.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ObjectDetection
demos
ObjectFeatureExtraction.java
ObjectDetection.java
ObjectMainDetection
ObjectMainDetection()
Match Case
Regex
Words
FeatureExtraction [demos] F:\BigDataAnalytics\Tutorials
.idea
data
1.jpg
2.jpg
3.jpg
4.jpg
elephant.jpg
gazelle.jpg
rhino.jpg
sample2.mkv
zebra.jpg
output
elephant.txt
features.txt
gazelle.txt
rhino.txt
zebra.txt
src
main
java
ObjectDetection.java
ObjectDetection
ObjectMainDetection
ObjectFeatureExtraction
resources
test
java
target
Classification.iml
LoadReferenceObject();
StartVideo();
public void StartVideo() throws IOException {
Video<MBFImage> video = new XuggleVideo(new File(pathname: "data/sample2.mkv"));
int count1 = 0, count2 = 0, count3 = 0, count4=0;
String o1 = "output/features.txt";
FileWriter fw = new FileWriter(o1);
BufferedWriter bw = new BufferedWriter(fw);
for (MBFImage mbfImage : video) {
final LocalFeatureList<Keypoint> kpl = this.engine.findFeatures(Transforms.calculateIntensityNTSC(mbfImage));
final MBFImageRenderer renderer = mbfImage.createRenderer();
renderer.drawPoints(kpl, RGBColour.MAGENTA, size: 3);
if (this.matcher1.findMatches(kpl)
&& ((MatrixTransformProvider) this.matcher1.getModel()).getTransform().cond() < 1e6) {
try {
final Matrix boundsToPoly = ((MatrixTransformProvider) this.matcher1.getModel()).getTransform()
.inverse();
if (modelImage1.getBounds().transform(boundsToPoly).isConvex()) {
renderer.drawShape(this.modelImage1.getBounds().transform(boundsToPoly), thickness: 3, RGBColour.RED);
if(count1 <= 10){
List<Point2d> vertices = this.modelImage1.getBounds().transform(boundsToPoly).asPolygon().getVertices();
int x[] = new int[4], y[] = new int[4];
for (int i = 0; i < vertices.size(); i++) {
x[i] = (int) vertices.get(i).getX();
y[i] = (int) vertices.get(i).getY();
All files are up-to-date (25 minutes ago)
69:47 CRLF UTF-8

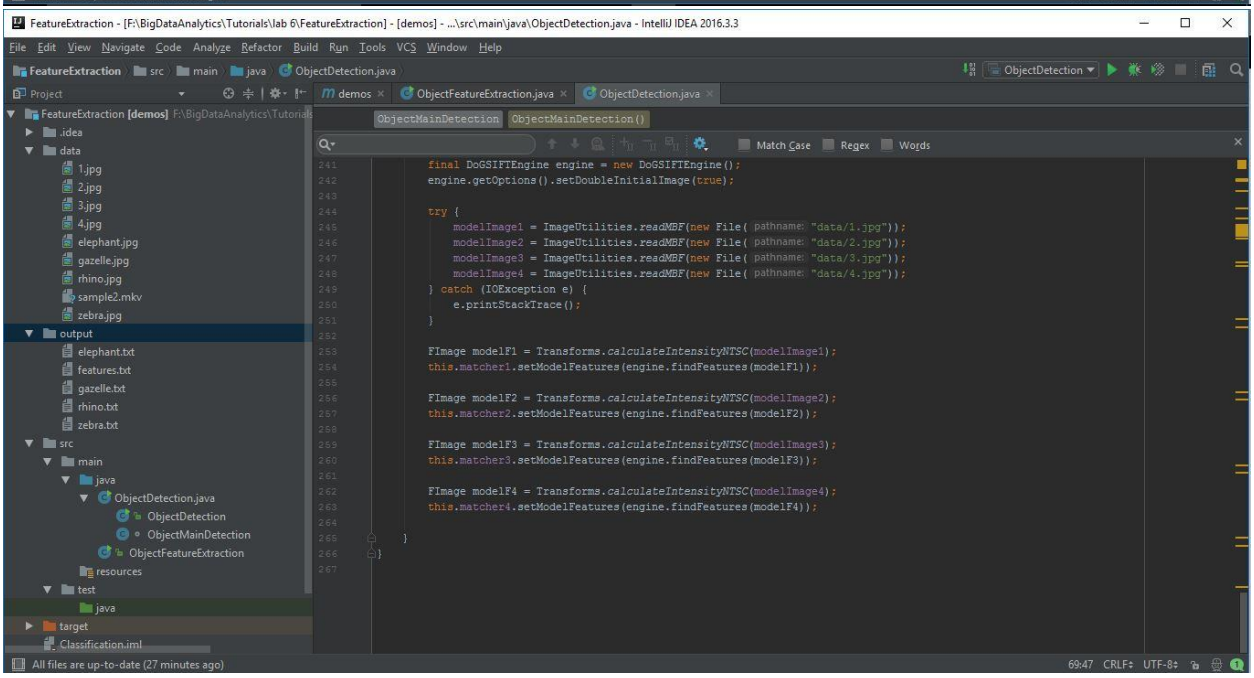

FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\src\main\java\ObjectDetection.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ObjectDetection
demos
ObjectFeatureExtraction.java
ObjectDetection.java
ObjectMainDetection
ObjectMainDetection()
Match Case
Regex
Words
FeatureExtraction [demos]
.idea
data
1.jpg
2.jpg
3.jpg
4.jpg
elephant.jpg
gazelle.jpg
rhino.jpg
sample2.mkv
zebra.jpg
output
elephant.txt
features.txt
gazelle.txt
rhino.txt
zebra.txt
src
main
java
ObjectDetection.java
ObjectDetection
ObjectMainDetection
ObjectFeatureExtraction
resources
test
java
target
Classification.iml
Polygon polygon = new Polygon(x, y, npoints: 4);
for (int i = 0; i < kpl.size(); i++) {
if (polygon.contains(kpl.get(i).getX(), kpl.get(i).getY())) {
double c[] = kpl.get(i).getFeatureVector().asDoubleVector();
bw.write(str: "0,");
for (int j = 0; j < c.length; j++) {
bw.write(str: c[j] + " ");
}
bw.newLine();
count1++;
} catch (Exception e) {
System.out.println(e.getMessage());
if (this.matcher2.findMatches(kpl)
&& ((MatrixTransformProvider) this.matcher2.getModel()).getTransform().cond() < 1e6) {
try {
final Matrix boundsToPoly = ((MatrixTransformProvider) this.matcher2.getModel()).getTransform()
.inverse();
renderer.drawShape(this.modelImage2.getBounds().transform(boundsToPoly), thickness: 3, RGBColour.RED);
if(count2 <= 10){
if (modelImage2.getBounds().transform(boundsToPoly).isConvex()) {
All files are up-to-date (25 minutes ago)
69:47 CRLF UTF-8

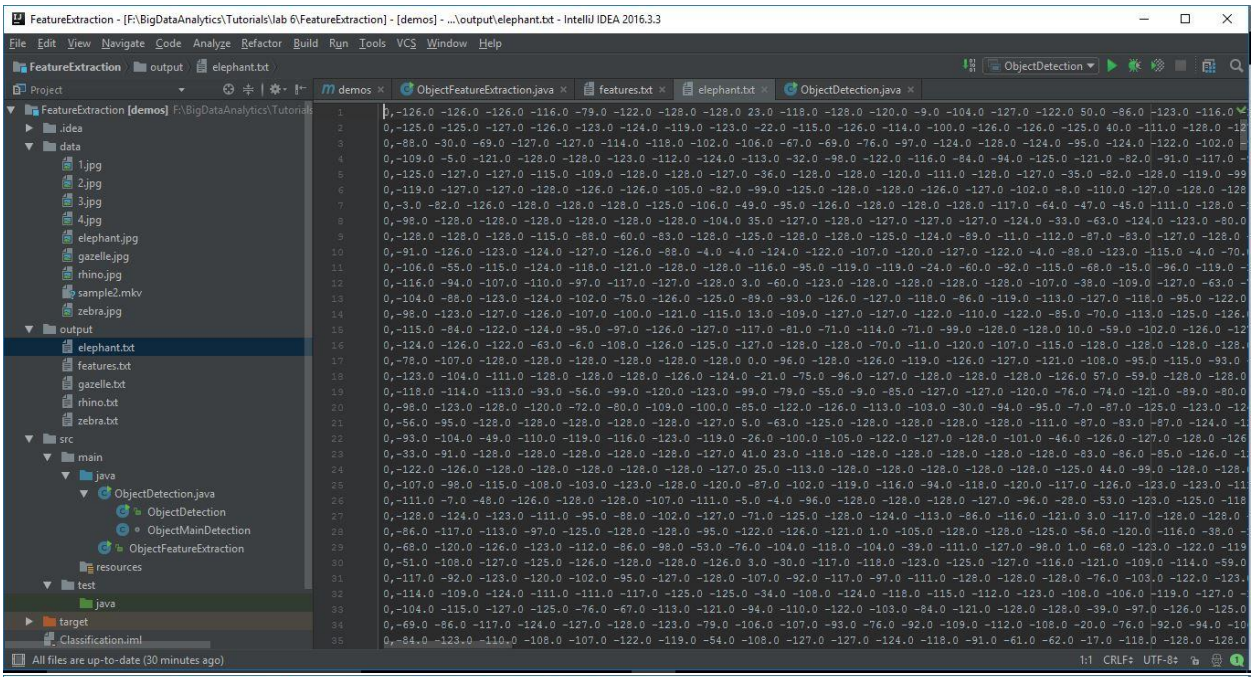

FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\src\main\java\ObjectDetection.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ObjectDetection
demos
ObjectFeatureExtraction.java
ObjectDetection.java
ObjectMainDetection
ObjectMainDetection()
Match Case
Regex
Words
FeatureExtraction [demos]
.idea
data
1.jpg
2.jpg
3.jpg
4.jpg
elephant.jpg
gazelle.jpg
rhino.jpg
sample2.mkv
zebra.jpg
output
elephant.txt
features.txt
gazelle.txt
rhino.txt
zebra.txt
src
main
java
ObjectDetection.java
ObjectDetection
ObjectMainDetection
ObjectFeatureExtraction
resources
test
java
target
Classification.iml
renderer.drawShape(this.modelImage2.getBounds().transform(boundsToPoly), thickness: 3, RGBColour.RED);
if(count2 <= 10){
if (modelImage2.getBounds().transform(boundsToPoly).isConvex()) {
List<Point2d> vertices = this.modelImage2.getBounds().transform(boundsToPoly).asPolygon().getVertices();
int x[] = new int[4], y[] = new int[4];
for (int i = 0; i < vertices.size(); i++) {
x[i] = (int) vertices.get(i).getX();
y[i] = (int) vertices.get(i).getY();
}
Polygon polygon = new Polygon(x, y, npoints: 4);
for (int i = 0; i < kpl.size(); i++) {
if (polygon.contains(kpl.get(i).getX(), kpl.get(i).getY())) {
double c[] = kpl.get(i).getFeatureVector().asDoubleVector();
bw.write(str: "1,");
for (int j = 0; j < c.length; j++) {
bw.write(str: c[j] + " ");
}
bw.newLine();
count2++;
} catch (Exception e) {
System.out.println(e.getMessage());
All files are up-to-date (25 minutes ago)
69:47 CRLF UTF-8

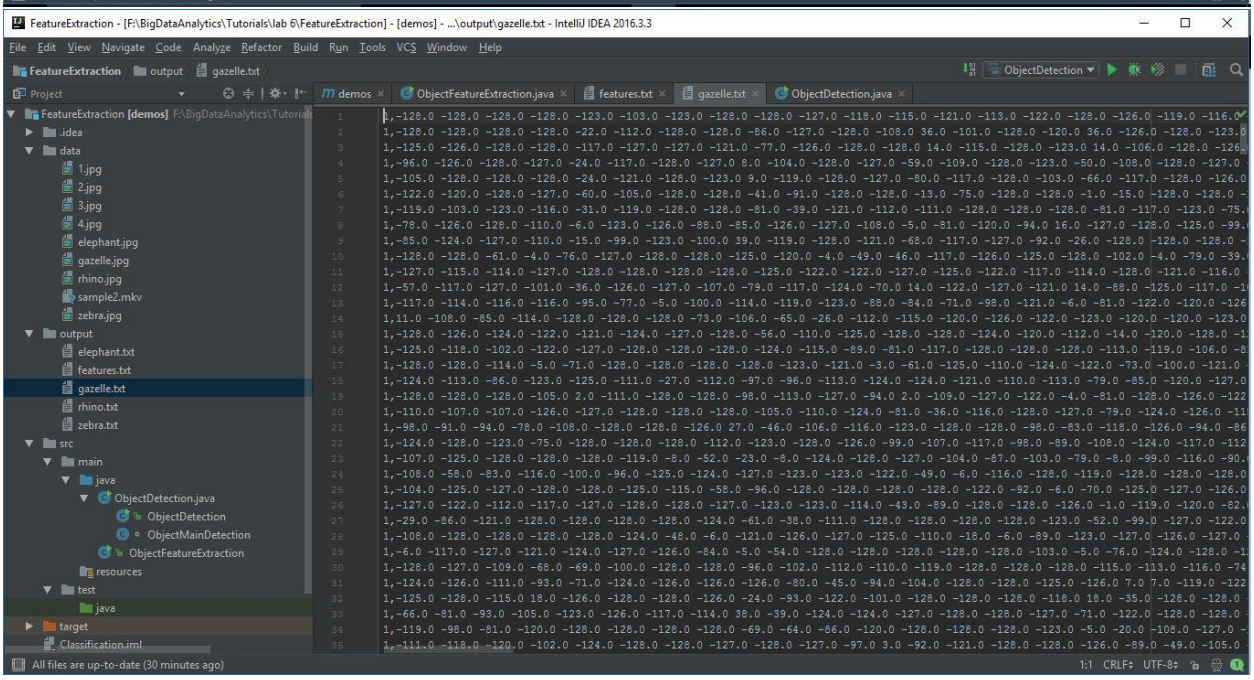

FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\src\main\java\ObjectDetection.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ObjectDetection.java
ObjectFeatureExtraction.java
ObjectMainDetection
ObjectMainDetection()
Match Case
Regex
Words
All files are up-to-date (25 minutes ago)
69:47 CRLF UTF-8

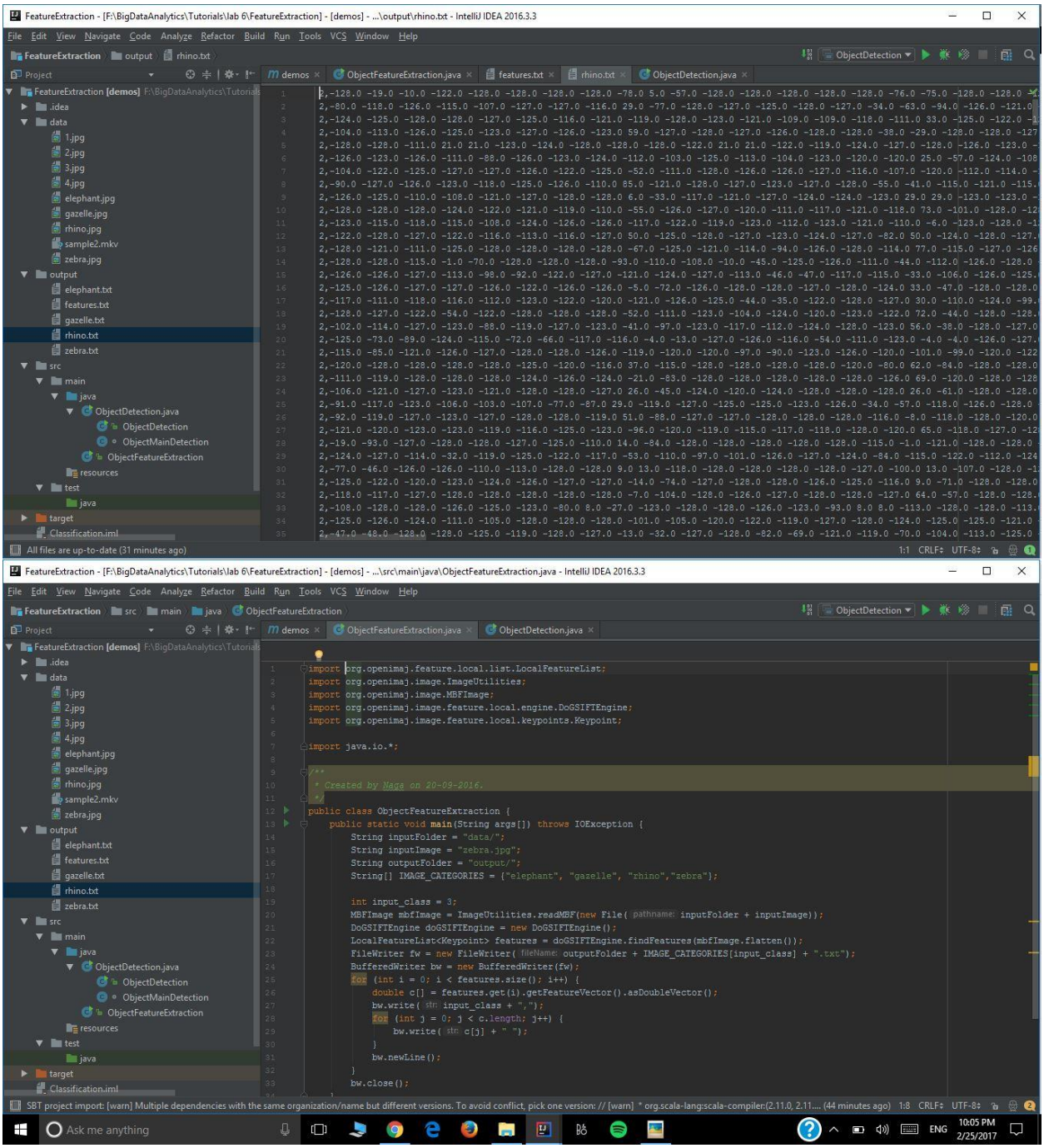

FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\src\main\java\ObjectDetection.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ObjectDetection.java
ObjectFeatureExtraction.java
ObjectMainDetection
ObjectMainDetection()
Match Case
Regex
Words
All files are up-to-date (26 minutes ago)
69:47 CRLF UTF-8


FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\src\main\java\ObjectDetection.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
FeatureExtraction src main java ObjectDetection.java
ObjectDetection
Project
demos
ObjectFeatureExtraction.java
ObjectDetection.java
ObjectMainDetection
ObjectMainDetection()
Match Case
Regex
Words
FeatureExtraction [demos] F:\BigDataAnalytics\Tutorials
.idea
data
1.jpg
2.jpg
3.jpg
4.jpg
elephant.jpg
gazelle.jpg
rhino.jpg
sample2.mkv
zebra.jpg
output
elephant.txt
features.txt
gazelle.txt
rhino.txt
zebra.txt
src
main
java
ObjectDetection.java
ObjectDetection
ObjectMainDetection
ObjectFeatureExtraction
resources
test
java
target
Classification.iml
} catch (Exception e) {
System.out.println(e.getMessage());
}
}
DisplayUtilities.displayName(mbfImage, name: "Image");
}
bw.close();
}
public void LoadReferenceObject() {
final DoGSIFTEngine engine = new DoGSIFTEngine();
engine.getOptions().setDoubleInitialImage(true);
try {
modelImage1 = ImageUtilities.readMBF(new File(pathname: "data/1.jpg"));
modelImage2 = ImageUtilities.readMBF(new File(pathname: "data/2.jpg"));
modelImage3 = ImageUtilities.readMBF(new File(pathname: "data/3.jpg"));
modelImage4 = ImageUtilities.readMBF(new File(pathname: "data/4.jpg"));
} catch (IOException e) {
e.printStackTrace();
}
FImage modelF1 = Transforms.calculateIntensityNTSC(modelImage1);
this.matcher1.setModelFeatures(engine.findFeatures(modelF1));
FImage modelF2 = Transforms.calculateIntensityNTSC(modelImage2);
this.matcher2.setModelFeatures(engine.findFeatures(modelF2));
FImage modelF3 = Transforms.calculateIntensityNTSC(modelImage3);
All files are up-to-date (27 minutes ago)
69:47 CRLF UTF-8

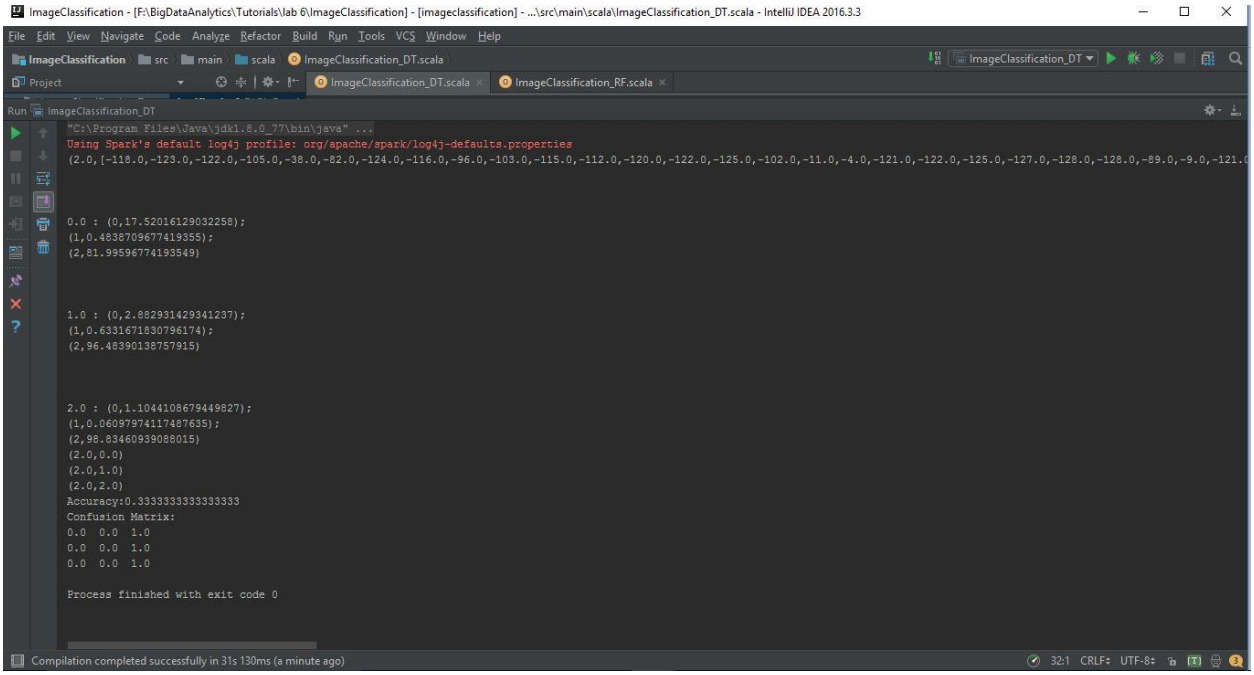

FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\src\main\java\ObjectDetection.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
FeatureExtraction src main java ObjectDetection.java
ObjectDetection
Project
demos
ObjectFeatureExtraction.java
ObjectDetection.java
ObjectMainDetection
ObjectMainDetection()
Match Case
Regex
Words
FeatureExtraction [demos] F:\BigDataAnalytics\Tutorials
.idea
data
1.jpg
2.jpg
3.jpg
4.jpg
elephant.jpg
gazelle.jpg
rhino.jpg
sample2.mkv
zebra.jpg
output
elephant.txt
features.txt
gazelle.txt
rhino.txt
zebra.txt
src
main
java
ObjectDetection.java
ObjectDetection
ObjectMainDetection
ObjectFeatureExtraction
resources
test
java
target
Classification.iml
final DoGSIFTEngine engine = new DoGSIFTEngine();
engine.getOptions().setDoubleInitialImage(true);
try {
modelImage1 = ImageUtilities.readMBF(new File(pathname: "data/1.jpg"));
modelImage2 = ImageUtilities.readMBF(new File(pathname: "data/2.jpg"));
modelImage3 = ImageUtilities.readMBF(new File(pathname: "data/3.jpg"));
modelImage4 = ImageUtilities.readMBF(new File(pathname: "data/4.jpg"));
} catch (IOException e) {
e.printStackTrace();
}
FImage modelF1 = Transforms.calculateIntensityNTSC(modelImage1);
this.matcher1.setModelFeatures(engine.findFeatures(modelF1));
FImage modelF2 = Transforms.calculateIntensityNTSC(modelImage2);
this.matcher2.setModelFeatures(engine.findFeatures(modelF2));
FImage modelF3 = Transforms.calculateIntensityNTSC(modelImage3);
this.matcher3.setModelFeatures(engine.findFeatures(modelF3));
FImage modelF4 = Transforms.calculateIntensityNTSC(modelImage4);
this.matcher4.setModelFeatures(engine.findFeatures(modelF4));
}
}
All files are up-to-date (27 minutes ago)
69:47 CRLF UTF-8

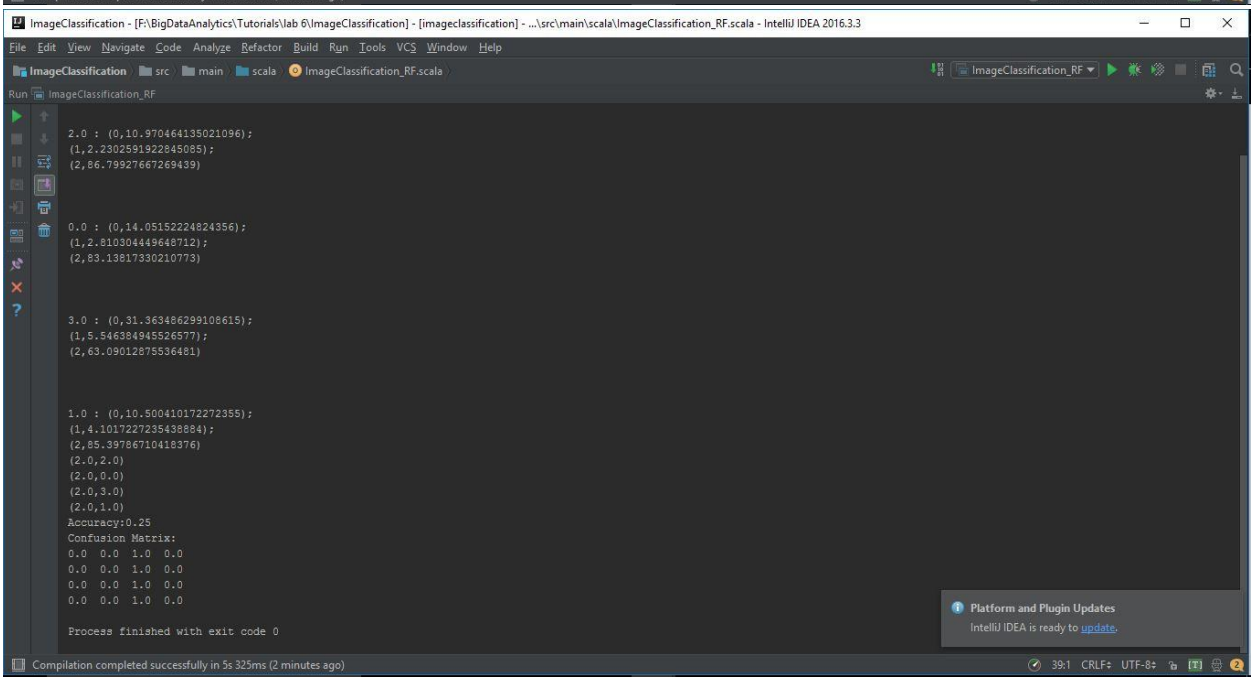

FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\output\elephant.txt - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
FeatureExtraction output elephant.txt
ObjectDetection
Project
demos
ObjectFeatureExtraction.java
features.txt
elephant.txt
ObjectDetection.java
FeatureExtraction [demos] F:\BigDataAnalytics\Tutorials
.idea
data
1.jpg
2.jpg
3.jpg
4.jpg
elephant.jpg
gazelle.jpg
rhino.jpg
sample2.mkv
zebra.jpg
output
elephant.txt
features.txt
gazelle.txt
rhino.txt
zebra.txt
src
main
java
ObjectDetection.java
ObjectDetection
ObjectMainDetection
ObjectFeatureExtraction
resources
test
java
target
Classification.iml
All files are up-to-date (30 minutes ago)
1:1 CRLF UTF-8

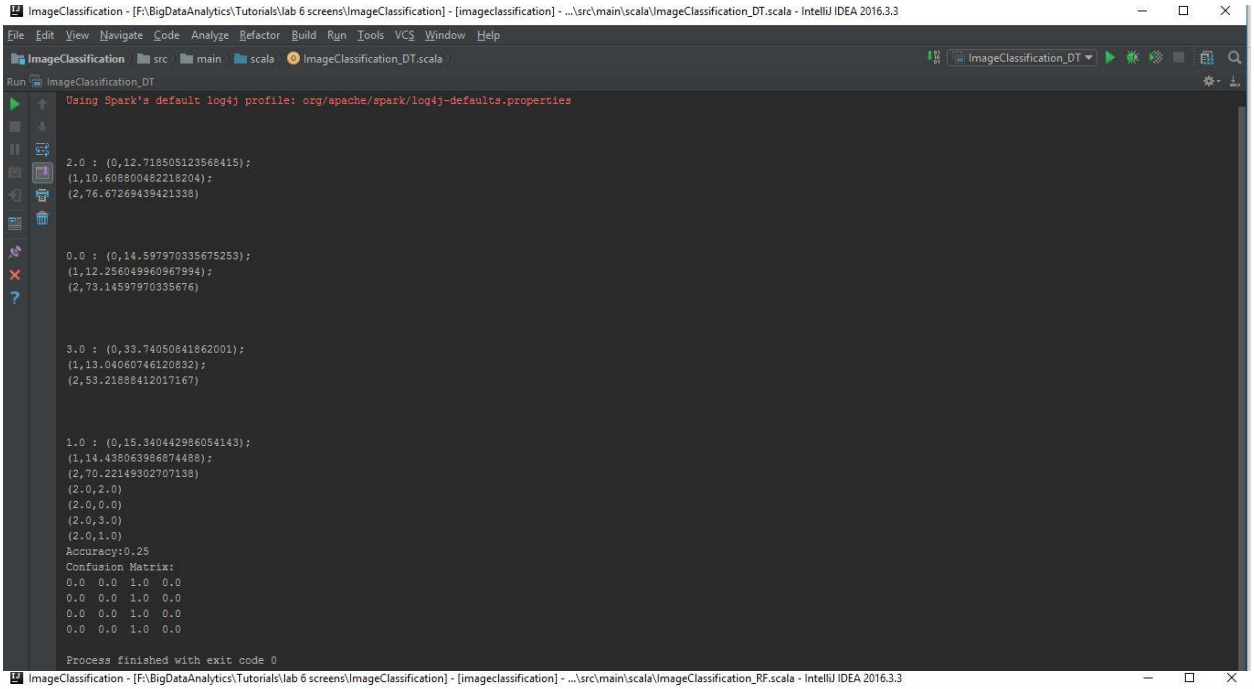

FeatureExtraction - [F:\BigDataAnalytics\Tutorials\lab 6\FeatureExtraction] - [demos] - ...\output\gazelle.txt - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
FeatureExtraction output gazelle.txt
ObjectDetection
Project
demos
ObjectFeatureExtraction.java
features.txt
gazelle.txt
ObjectDetection.java
FeatureExtraction [demos] F:\BigDataAnalytics\Tutorials
.idea
data
1.jpg
2.jpg
3.jpg
4.jpg
elephant.jpg
gazelle.jpg
rhino.jpg
sample2.mkv
zebra.jpg
output
elephant.txt
features.txt
gazelle.txt
rhino.txt
zebra.txt
src
main
java
ObjectDetection.java
ObjectDetection
ObjectMainDetection
ObjectFeatureExtraction
resources
test
java
target
Classification.iml
All files are up-to-date (30 minutes ago)
1:1 CRLF UTF-8

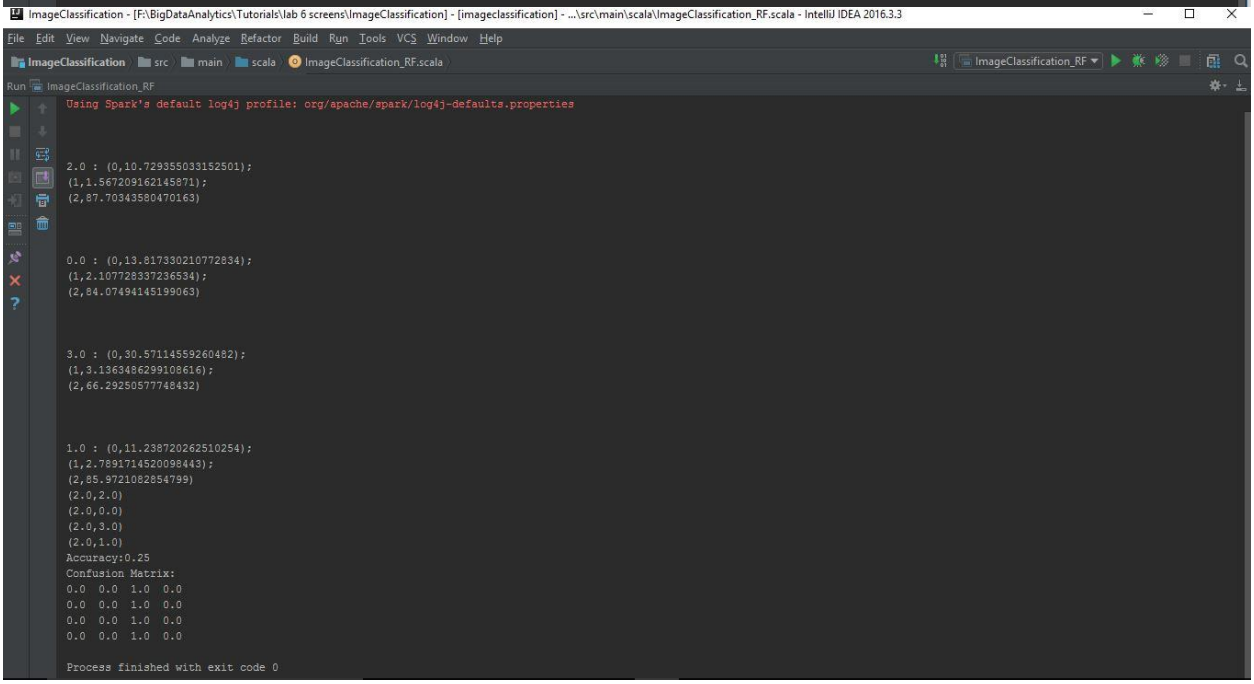


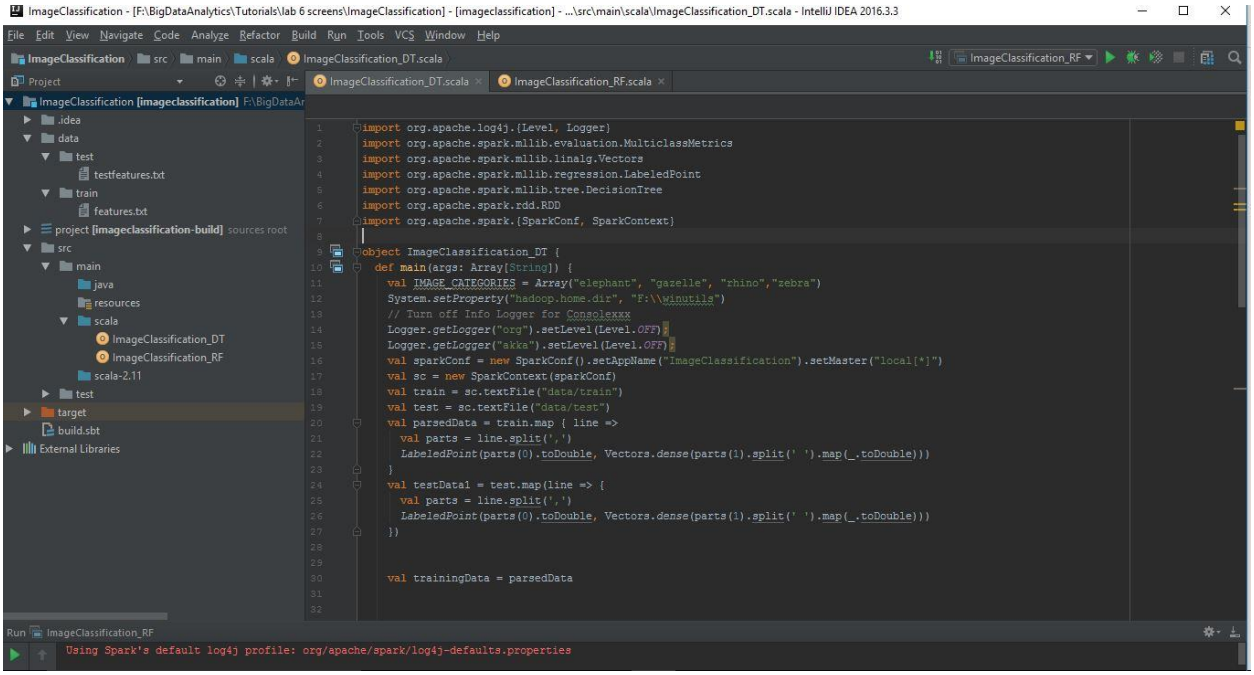
The features extracted from these are divided into test and train features, the image classification algorithm is run using this test and train features. Here are the screens:

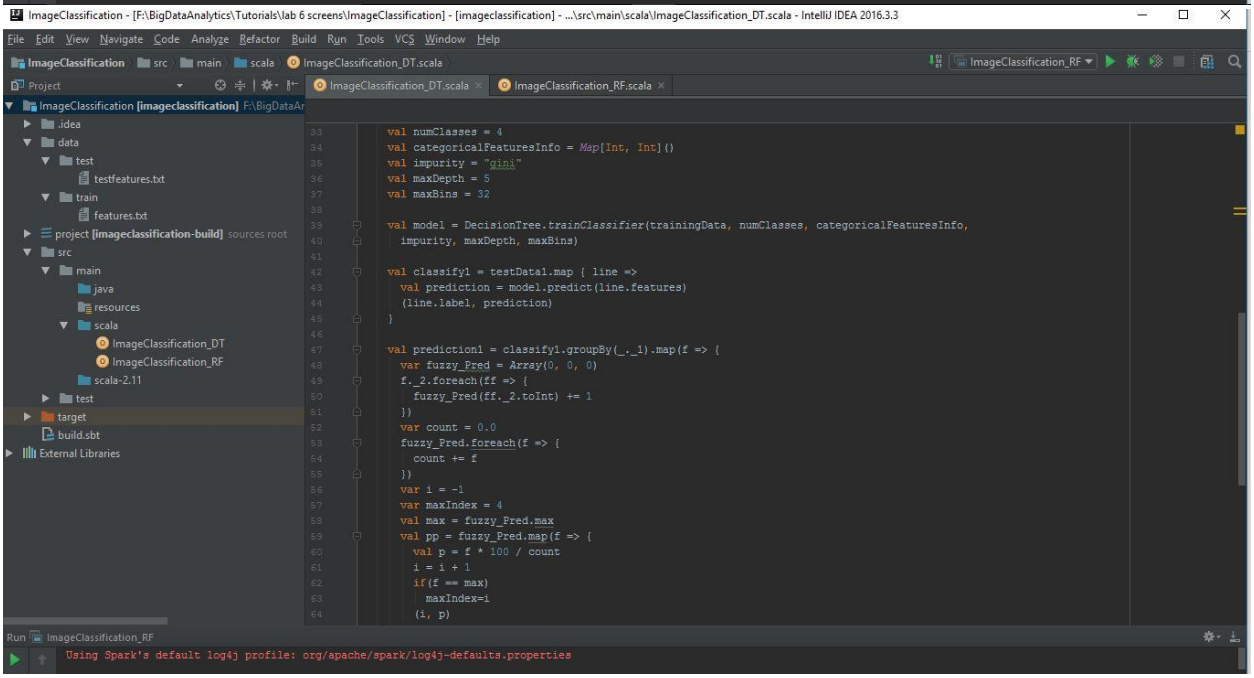

ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_DT.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification src main scala ImageClassification_DT.scala
ImageClassification_DT
Project
ImageClassification_DT.scala
ImageClassification_RF.scala
Run ImageClassification_DT
"C:\Program Files\Java\jdk1.8.0_77\bin\java" ...
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
(2.0,[-118.0,-123.0,-122.0,-105.0,-38.0,-82.0,-124.0,-116.0,-96.0,-103.0,-115.0,-112.0,-120.0,-122.0,-125.0,-102.0,-11.0,-4.0,-121.0,-122.0,-125.0,-127.0,-128.0,-128.0,-89.0,-9.0,-121.
0.0 : (0,17.52016129032258);
(1,0.4838709677419355);
(2,81.99596774193549)
1.0 : (0,2.882931429341237);
(1,0.6331671830796174);
(2,96.48390138757915)
2.0 : (0,1.1044108679449827);
(1,0.06097974117487635);
(2,98.83460939088015)
(2.0,0.0)
(2.0,1.0)
(2.0,2.0)
Accuracy:0.3333333333333333
Confusion Matrix:
0.0 0.0 1.0
0.0 0.0 1.0
0.0 0.0 1.0
Process finished with exit code 0
Compilation completed successfully in 31s 130ms (a minute ago)
32:1 CRLF: UTF-8:

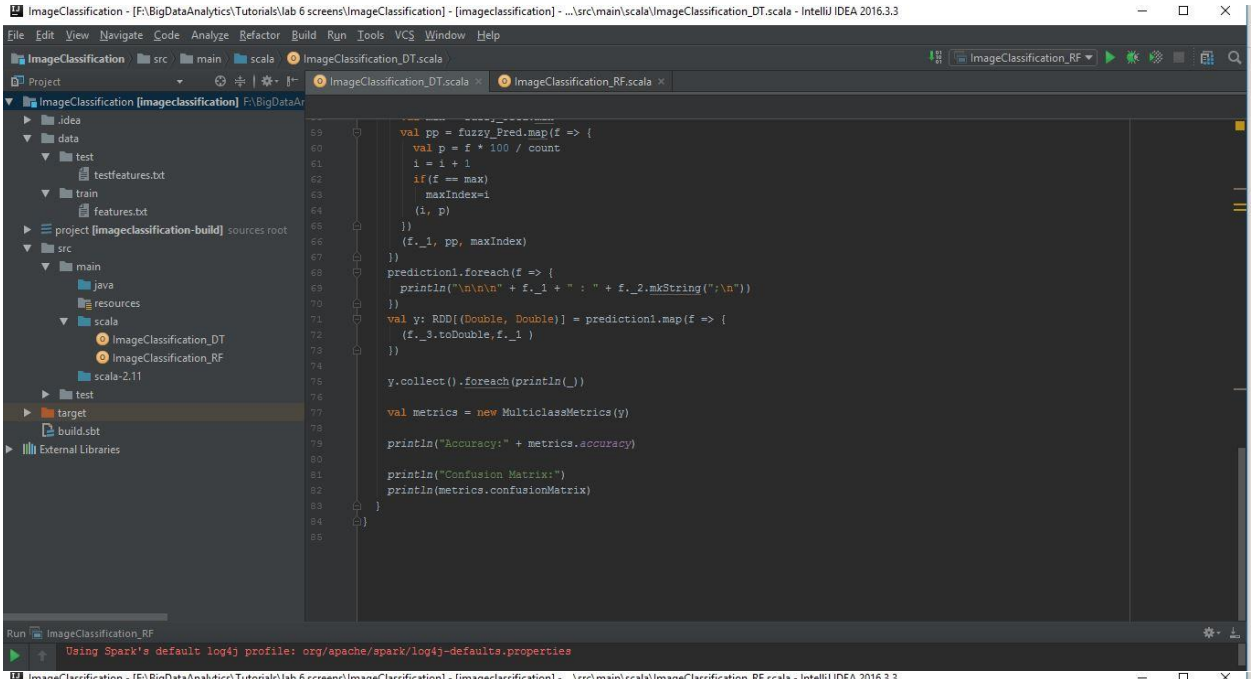

ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_RF.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification src main scala ImageClassification_RF.scala
ImageClassification_RF
Run ImageClassification_RF
2.0 : (0,10.970464135021096);
(1,2.2302591922845085);
(2,86.79927667269439)
0.0 : (0,14.05152224824356);
(1,2.810304449648712);
(2,83.13817330210773)
3.0 : (0,31.363486299108615);
(1,5.546384945526577);
(2,63.09012875536481)
1.0 : (0,10.500410172272355);
(1,4.1017227235438884);
(2,85.39786710418376)
(2.0,2.0)
(2.0,0.0)
(2.0,3.0)
(2.0,1.0)
Accuracy:0.25
Confusion Matrix:
0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0
Process finished with exit code 0
Platform and Plugin Updates
IntelliJ IDEA is ready to update.
Compilation completed successfully in 5s 325ms (2 minutes ago)
39:1 CRLF: UTF-8:

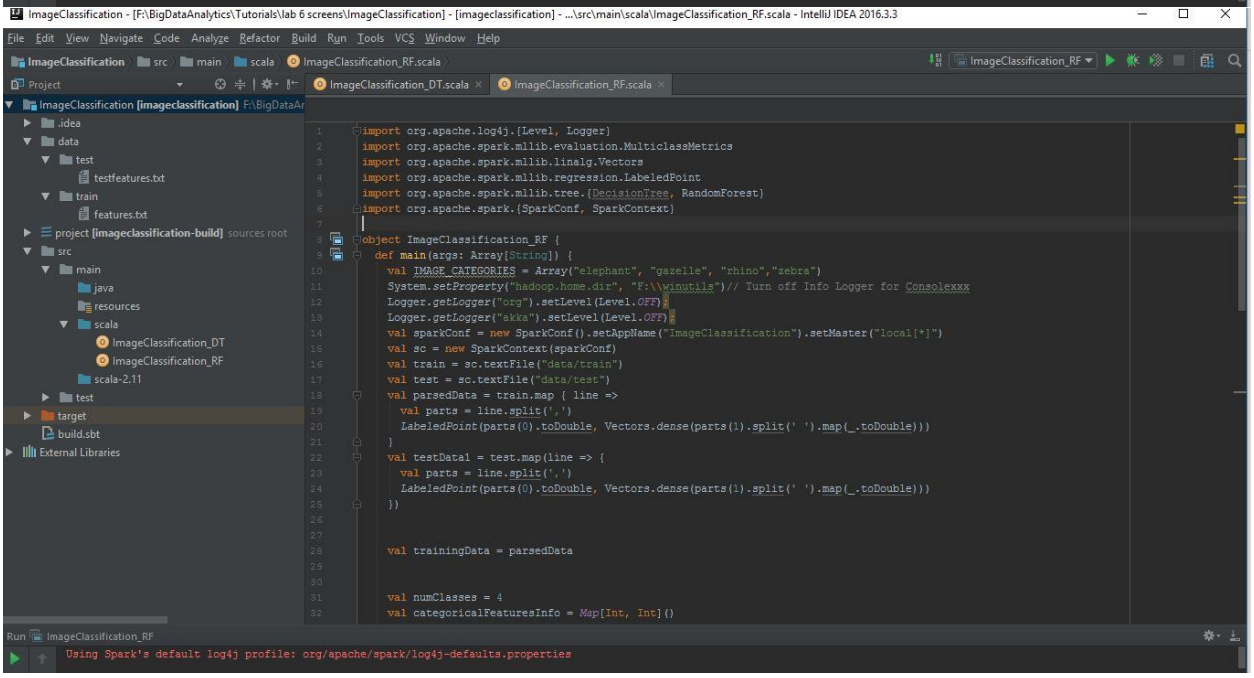

ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6 screens\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_DT.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification src main scala ImageClassification_DT.scala
ImageClassification_DT
Run ImageClassification_DT
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
2.0 : (0,12.718505123568415);
(1,10.608800482218204);
(2,76.67269439421338)
0.0 : (0,14.597970335675253);
(1,12.256049960967994);
(2,73.14597970335676)
3.0 : (0,33.74050841862001);
(1,13.04060746120832);
(2,53.2188412017167)
1.0 : (0,15.340442986054143);
(1,14.438063986874488);
(2,70.22149302707138)
(2.0,2.0)
(2.0,0.0)
(2.0,3.0)
(2.0,1.0)
Accuracy:0.25
Confusion Matrix:
0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0
Process finished with exit code 0

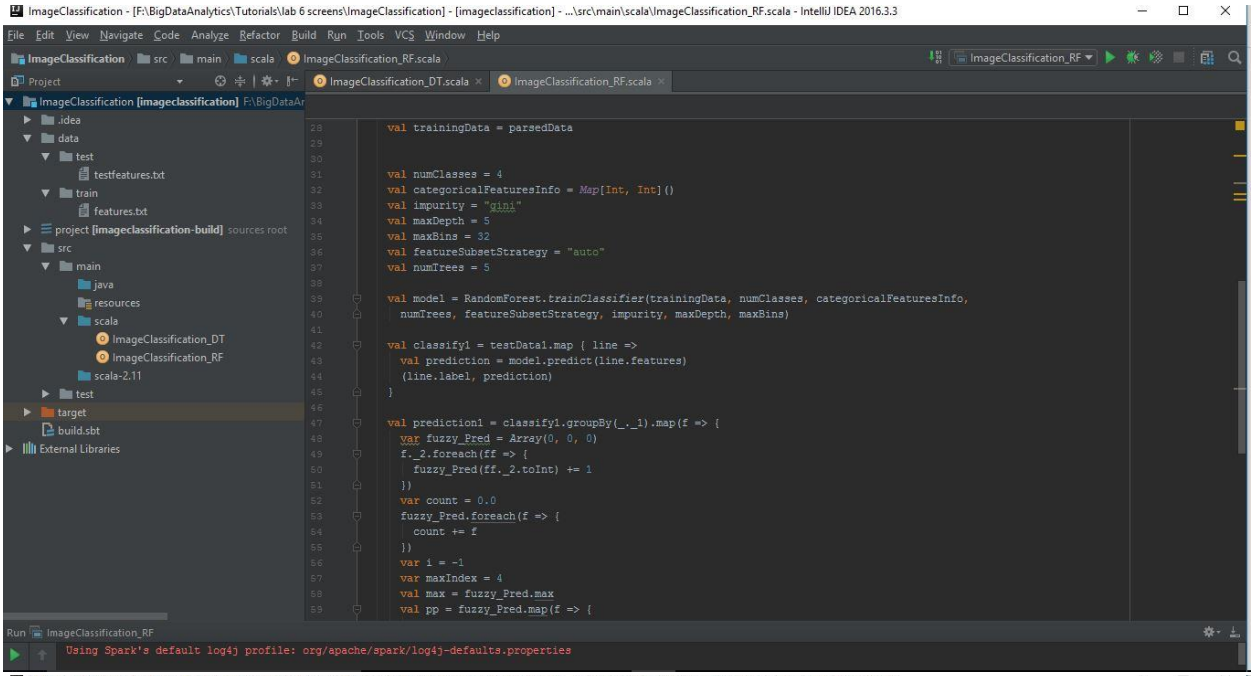

ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6 screens\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_RF.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification src main scala ImageClassification_RF.scala
ImageClassification_RF
Run ImageClassification_RF
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
2.0 : (0,10.729355033152501);
(1,1.567209162145871);
(2,87.70343580470163)
0.0 : (0,13.817330210772834);
(1,2.107728337236534);
(2,84.07494145199063)
3.0 : (0,30.57114559260482);
(1,3.1363486299108616);
(2,66.2925057748432)
1.0 : (0,11.238720262510254);
(1,2.7891714520098443);
(2,85.9721082854799)
(2.0,2.0)
(2.0,0.0)
(2.0,3.0)
(2.0,1.0)
Accuracy:0.25
Confusion Matrix:
0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0
0.0 0.0 1.0 0.0
Process finished with exit code 0

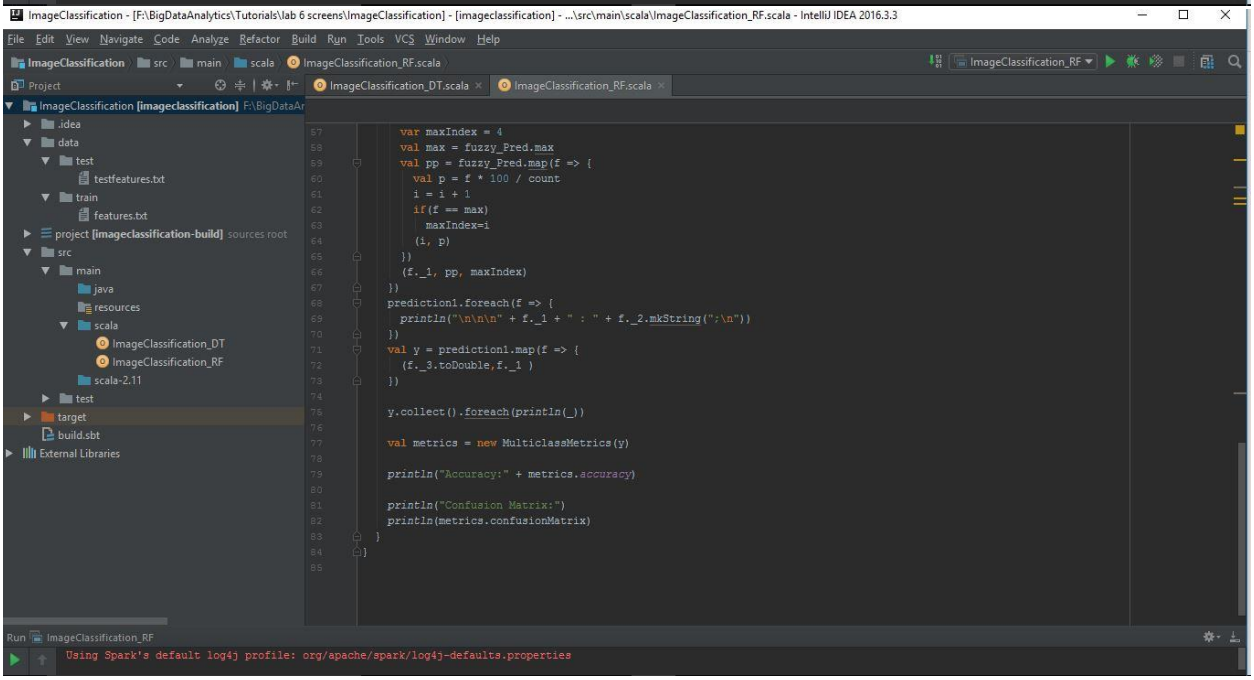

ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6 screens\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_DT.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification_DT.scala
ImageClassification_RF.scala
import org.apache.log4j.{Level, Logger}
import org.apache.spark.mllib.evaluation.MulticlassMetrics
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.tree.DecisionTree
import org.apache.spark.rdd.RDD
import org.apache.spark.{SparkConf, SparkContext}
object ImageClassification_DT {
def main(args: Array[String]) {
val IMAGE_CATEGORIES = Array("elephant", "gazelle", "rhino","zebra")
System.setProperty("hadoop.home.dir", "F:\\winutils")
// Turn off Info Logger for Consolexxx
Logger.getLogger("org").setLevel(Level.OFF);
Logger.getLogger("akka").setLevel(Level.OFF);
val sparkConf = new SparkConf().setAppName("ImageClassification").setMaster("local[*]")
val sc = new SparkContext(sparkConf)
val train = sc.textFile("data/train")
val test = sc.textFile("data/test")
val parsedData = train.map { line =>
val parts = line.split(',')
LabeledPoint(parts(0).toDouble, Vectors.dense(parts(1).split(' ').map(_.toDouble)))
}
val testData1 = test.map(line => {
val parts = line.split(',')
LabeledPoint(parts(0).toDouble, Vectors.dense(parts(1).split(' ').map(_.toDouble)))
})
val trainingData = parsedData
Run ImageClassification_RF
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties

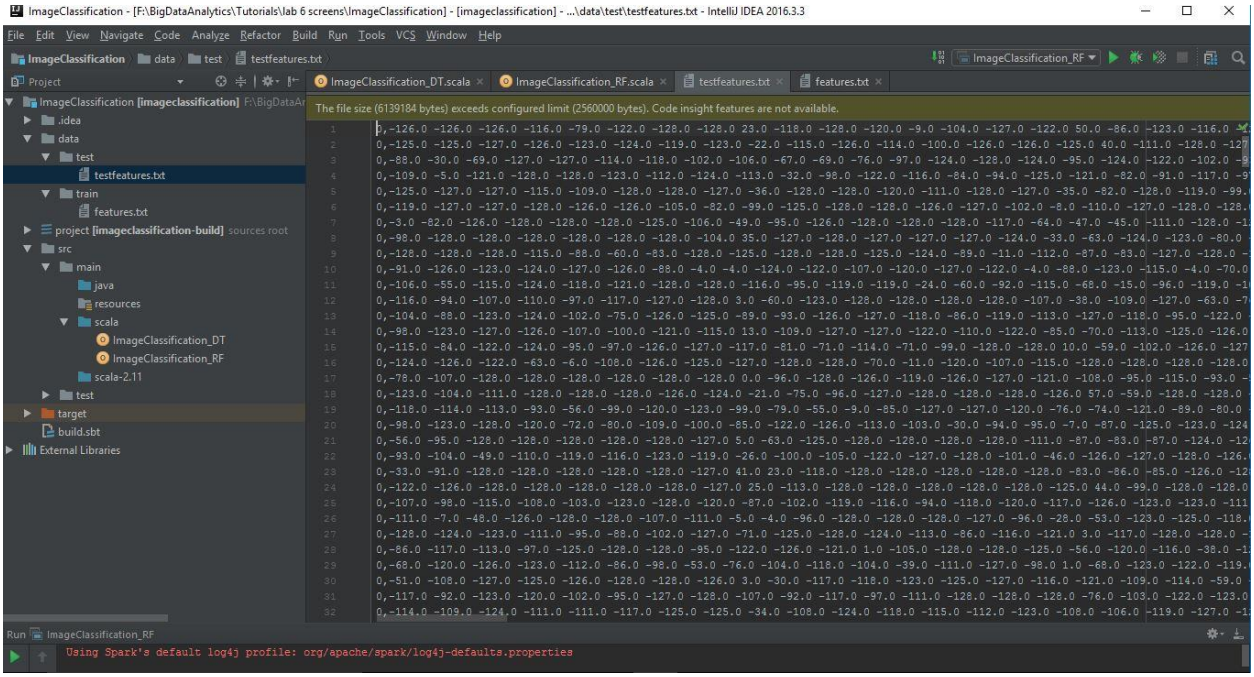

ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6 screens\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_DT.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification_DT.scala
ImageClassification_RF.scala
val numClasses = 4
val categoricalFeaturesInfo = Map[Int, Int]()
val impurity = "gini"
val maxDepth = 5
val maxBins = 32
val model = DecisionTree.trainClassifier(trainingData, numClasses, categoricalFeaturesInfo,
impurity, maxDepth, maxBins)
val classify1 = testData1.map { line =>
val prediction = model.predict(line.features)
(line.label, prediction)
}
val prediction1 = classify1.groupBy(_._1).map(f => {
var fuzzy_Pred = Array(0, 0, 0)
f._2.foreach(ff => {
fuzzy_Pred(ff._2.toInt) += 1
})
var count = 0.0
fuzzy_Pred.foreach(f => {
count += f
})
var i = -1
var maxIndex = 4
val max = fuzzy_Pred.max
val pp = fuzzy_Pred.map(f => {
val p = f * 100 / count
i = i + 1
if(f == max)
maxIndex=i
(i, p)
Run ImageClassification_RF
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties


ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6 screens\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_DT.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification_DT.scala
ImageClassification_RF.scala
ImageClassification [imageclassification]
.idea
data
test
testfeatures.txt
train
features.txt
project [imageclassification-build] sources root
src
main
java
resources
scala
ImageClassification_DT
ImageClassification_RF
scala-2.11
test
target
build.sbt
External Libraries
val pp = fuzzy_Pred.map(f => {
val p = f * 100 / count
i = i + 1
if(f == max)
maxIndex=i
(i, p)
})
(f._1, pp, maxIndex)
})
prediction1.foreach(f => {
println("\n\n\n" + f._1 + " : " + f._2.mkString(";\n"))
})
val y: RDD[(Double, Double)] = prediction1.map(f => {
(f._3.toDouble,f._1)
})
y.collect().foreach(println(_))
val metrics = new MulticlassMetrics(y)
println("Accuracy:" + metrics.accuracy)
println("Confusion Matrix:")
println(metrics.confusionMatrix)
}
}
Run ImageClassification_RF
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties


ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6 screens\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_RF.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification_DT.scala
ImageClassification_RF.scala
ImageClassification [imageclassification]
.idea
data
test
testfeatures.txt
train
features.txt
project [imageclassification-build] sources root
src
main
java
resources
scala
ImageClassification_DT
ImageClassification_RF
scala-2.11
test
target
build.sbt
External Libraries
import org.apache.log4j.{Level, Logger}
import org.apache.spark.mllib.evaluation.MulticlassMetrics
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.tree.{DecisionTree, RandomForest}
import org.apache.spark.{SparkConf, SparkContext}
object ImageClassification_RF {
def main(args: Array[String]) {
val IMAGE_CATEGORIES = Array("elephant", "gazelle", "rhino","zebra")
System.setProperty("hadoop.home.dir", "F:\\winutils")// Turn off Info Logger for Consolexxx
Logger.getLogger("org").setLevel(Level.OFF);
Logger.getLogger("akka").setLevel(Level.OFF);
val sparkConf = new SparkConf().setAppName("ImageClassification").setMaster("local[*]")
val sc = new SparkContext(sparkConf)
val train = sc.textFile("data/train")
val test = sc.textFile("data/test")
val parsedData = train.map { line =>
val parts = line.split(',')
LabeledPoint(parts(0).toDouble, Vectors.dense(parts(1).split(' ').map(_.toDouble)))
}
val testData1 = test.map(line => {
val parts = line.split(',')
LabeledPoint(parts(0).toDouble, Vectors.dense(parts(1).split(' ').map(_.toDouble)))
})
val trainingData = parsedData
val numClasses = 4
val categoricalFeaturesInfo = Map[Int, Int]()
Run ImageClassification_RF
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties


ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6 screens\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_RF.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification_DT.scala
ImageClassification_RF.scala
val trainingData = parsedData
val numClasses = 4
val categoricalFeaturesInfo = Map[Int, Int]()
val impurity = "gini"
val maxDepth = 5
val maxBins = 32
val featureSubsetStrategy = "auto"
val numTrees = 5
val model = RandomForest.trainClassifier(trainingData, numClasses, categoricalFeaturesInfo,
numTrees, featureSubsetStrategy, impurity, maxDepth, maxBins)
val classify1 = testData1.map { line =>
val prediction = model.predict(line.features)
(line.label, prediction)
}
val prediction1 = classify1.groupBy(_._1).map(f => {
var fuzzy_Pred = Array(0, 0, 0)
f._2.foreach(ff => {
fuzzy_Pred(ff._2.toInt) += 1
})
var count = 0.0
fuzzy_Pred.foreach(f => {
count += f
})
var i = -1
var maxIndex = 4
val max = fuzzy_Pred.max
val pp = fuzzy_Pred.map(f => {
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties


ImageClassification - [F:\BigDataAnalytics\Tutorials\lab 6 screens\ImageClassification] - [imageclassification] - ...\src\main\scala\ImageClassification_RF.scala - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ImageClassification_DT.scala
ImageClassification_RF.scala
var maxIndex = 4
val max = fuzzy_Pred.max
val pp = fuzzy_Pred.map(f => {
val p = f * 100 / count
i = i + 1
if(f == max)
maxIndex=i
(i, p)
})
(f._1, pp, maxIndex)
})
prediction1.foreach(f => {
println("\n\n\n" + f._1 + " : " + f._2.mkString(";\n"))
})
val y = prediction1.map(f => {
(f._3.toDouble,f._1)
})
y.collect().foreach(println(_))
val metrics = new MulticlassMetrics(y)
println("Accuracy:" + metrics.accuracy)
println("Confusion Matrix:")
println(metrics.confusionMatrix)
}
}
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties



The accuracy came out as 25%