GES 488 Lab 11

Justin Drew

12/03/2019

Questions

Q1: You need to decide which of these variables is the predicted variable, and which is the predictor variable(s) in our classification. y ~ x just means y as a function of x. Write a line of R code that labels the columns correctly. The next line of R code is wrong–you correct it to run the classification model (a decision tree). What is written below says that the first column is predicted by (the y variable) the variables in columns 2 and 3 (the x variables). That's nonsense, you can improve it right?

Q2: Calculate the overall accuracy of your model using a confusion matrix function in R. Then calculate, for each class, the error of omission and the error of commission (not the accuracy). You can get the confusion matrix table using the code below, but you need to look up a confusion matrix function.

Hint: If you figure it out properly, it will involving installing a new library (remember to set dependencies=TRUE), waiting, and then using a simple function on your table. If for some reason you can't use a package, calculate it in R "by hand". And for our purposes, Sensitivity = (1 - error of omission), while Positive Predicted Value = (1 - error of commission).

Q3: Why do you think the accuracy of your classification model is so high? Hint: it's really not that high. randomForest estimates accuracy by witholding neighboring pixels from your training polygons before running the classification model (the out-of-bag (OOB) sample).

Q4: Use the predict() function to make a new classified raster using your classification model, predicting for the whole raster file (r1). Plot that raster, then write that classified raster out as a tiff file. Your answer is in two parts: A) your R code, and it should only be three lines; and B) your predicted raster.

Q5: If I gave you a similar image to this one (same location, three years later (in 2016), still Landsat 8, and taken in late spring as well), could you expect to use the same classification model to predict it accurately? Why or why not?

Q6: What about if the image was similar, taken three years later, but taken in late fall? Why or why not?

Q7: What about if you wanted to create a new classification model for the image in Q6–could you use the same shapefile training data without modifying it? Why or why not?

Q8: Your employers only care about the land near the lake. Subset your predicted land cover raster down to the lake and its immediate vicinity.

Homework

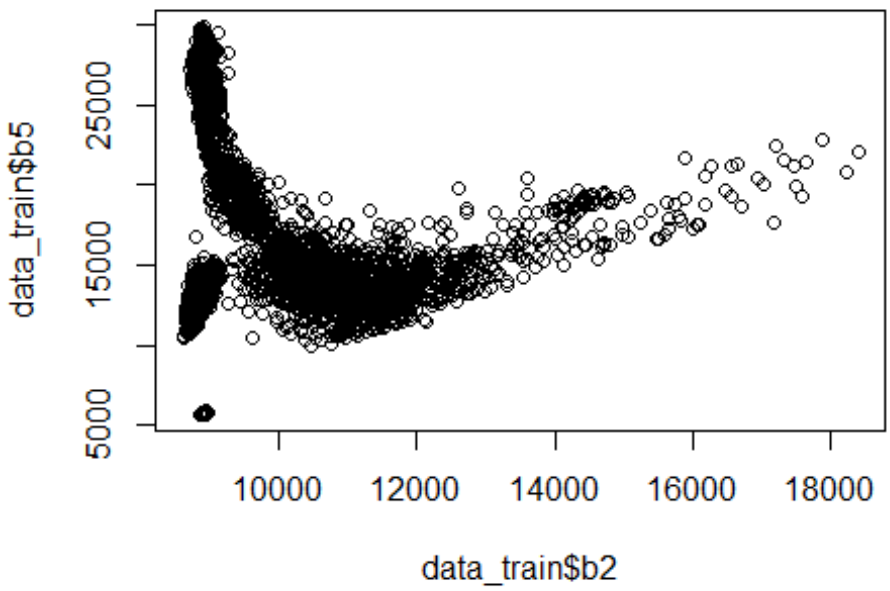
Q9: Tell me, in simple pseucocode instructions, how you would go about assessing the accuracy of this image using a combination of ArcGIS (or QGIS) and R. Assume you don't get to travel to Italy to ground-truth. Your description should walk me through the process in enough detail that you could hand it to your neighbor and they would know what to do. You might want to review this: http://web.pdx.edu/~nauna/resources/9-accuracyassessment.pdf

Bonus (+2):

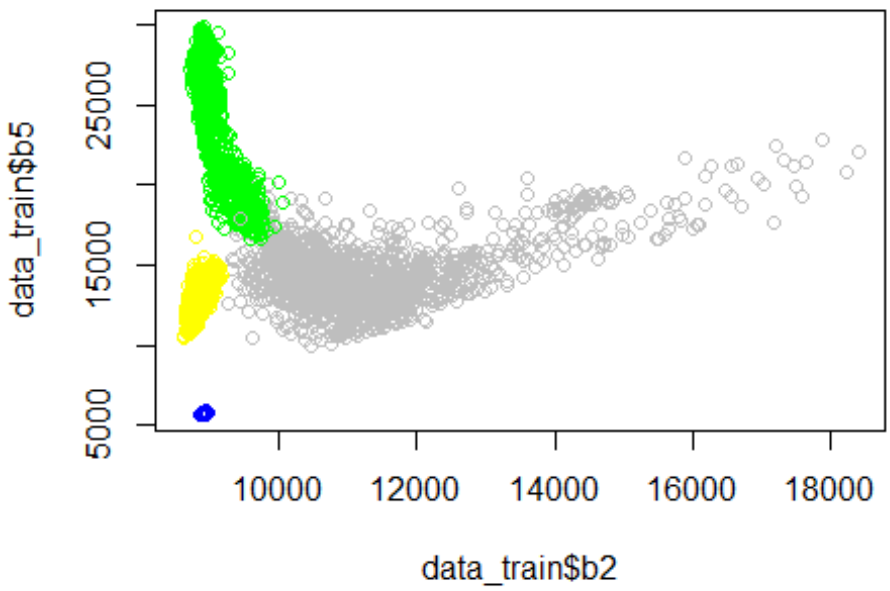
Actually do the above homework you designed, with at least 20 locations per class. Include any R code.

Setup

```
# unzip("Frascati_sample_image.zip")
# unzip("TrainingData_utm33.zip")

r1 <- brick("Frascati_sample_image/sample_image.tif")
names(r1)<-paste0("b", 2:7)

train1<-shapefile("TrainingData_utm33.shp")
ext_data<-extract(r1, train1, df=TRUE)

dim(train1@data)

## [1] 11  3

dim(ext_data)

## [1] 5014    7

data_train<-merge(train1@data[,1:2], ext_data, by.x=1, by.y=1)
head(data_train)

##   id Class   b2   b3   b4   b5   b6   b7
## 1  1 Water 8865 7898 6499 5699 5173 5099
## 2  1 Water 8895 7913 6480 5718 5170 5099
## 3  1 Water 8891 7931 6482 5708 5173 5107
## 4  1 Water 8887 7930 6496 5712 5182 5093
## 5  1 Water 8903 7927 6491 5723 5171 5107
## 6  1 Water 8906 7942 6499 5710 5176 5096

plot(data_train$b2, data_train$b5)
```




```
#
plot(data_train$b2, data_train$b5, col=ifelse(data_train$Class=="Water",
"blue", ifelse(data_train$Class=="BareSoil", "yellow",
ifelse(data_train$Class=="Veg", "green", ifelse(data_train$Class=="Urb",
"gray", "red")))))
```



Answer to Question 1

```
mod1<-randomForest(y=factor(data_train[,2]), x=data_train[,3:8],
data=data_train)
mod1

##
## Call:
##  randomForest(x = data_train[, 3:8], y = factor(data_train[, 2]),
data = data_train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.54%
## Confusion matrix:
##          BareSoil  Urb  Veg Water class.error
## BareSoil     1295    0    0     0  0.00000000
## Urb             1 1599   14     0  0.00929368
## Veg             0   12 1058     0  0.01121495
## Water           0    0    0  1035  0.00000000

importance(mod1)

##    MeanDecreaseGini
## b2         785.4752
## b3         637.5702
```

```
## b4          770.6988
## b5          741.8659
## b6          329.2917
## b7          452.1044
```

Answer to Question 2

Functions

```
accuracy <- function(preds,ref){
  stopifnot(length(preds)==length(ref))
  sum(preds==ref)/length(preds)
}

classSensitivity <- function(preds,ref,class.name){
  stopifnot(length(preds)==length(ref))
  pred1 <- preds == class.name
  ref1 <- ref == class.name
  TP <- sum(pred1 & ref1)
  class.ct <- sum(ref1)
  TP/class.ct
}

classPPV <- function(preds,ref,class.name){
  stopifnot(length(preds)==length(ref))
  pred1 <- preds == class.name
  ref1 <- ref == class.name
  TP <- sum(pred1 & ref1)
  pred.ct <- sum(pred1)
  TP/pred.ct
}
```

Code & Outputs

```
confMat<-as.table(mod1$confusion[,1:4])
print(confMat)
```

```
##          BareSoil  Urb  Veg Water
## BareSoil     1295    0    0     0
## Urb             1 1599   14     0
## Veg             0   12 1058     0
## Water           0    0    0  1035
```

```
acc <- round(accuracy(mod1$predicted,data_train$Class),4)
print(paste("Overall Accuracy:",acc))
```

```
## [1] "Overall Accuracy: 0.9946"
```

```
for(cn in rownames(confMat)) {
  om.err <- round(1 - classSensitivity(mod1$predicted,data_train$Class,cn),4)
  com.err <- round(1 - classPPV(mod1$predicted,data_train$Class,cn),4)
```

```
  print(paste0("Error for ",cn,":"))
  print(paste("Omission Error:",om.err))
  print(paste("Commission Error:",com.err))
}
```

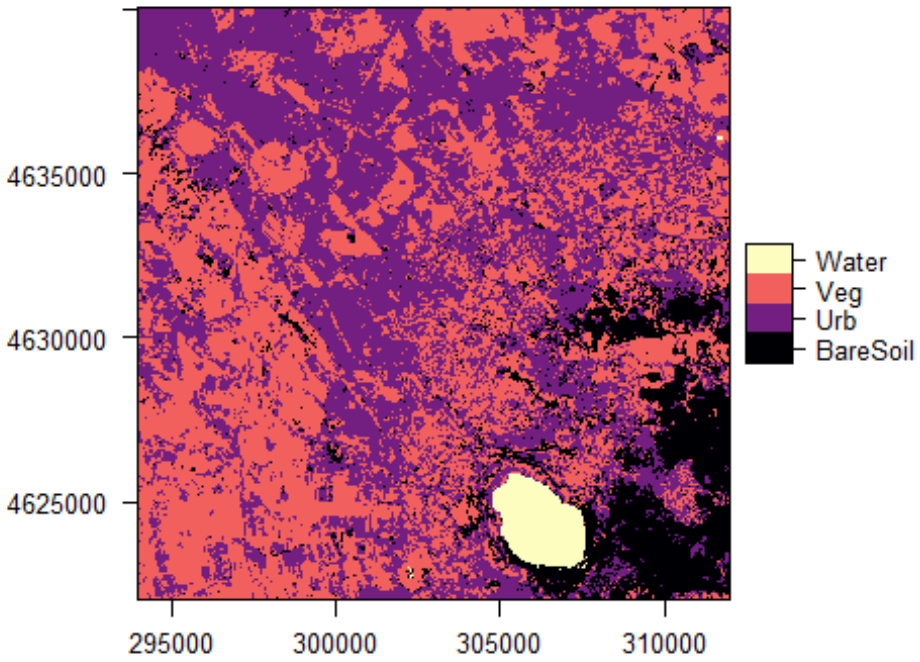
```
## [1] "Error for BareSoil:"
## [1] "Omission Error: 0"
## [1] "Commission Error: 8e-04"
## [1] "Error for Urb:"
## [1] "Omission Error: 0.0093"
## [1] "Commission Error: 0.0074"
## [1] "Error for Veg:"
## [1] "Omission Error: 0.0112"
## [1] "Commission Error: 0.0131"
## [1] "Error for Water:"
## [1] "Omission Error: 0"
## [1] "Commission Error: 0"
```

Answer to Question 3

The accuracy is so high because it's fit on the training data, meaning it's defining how it predicts new data based on what the training data looks like. As a result, this internal confusion matrix, which shows the result of predicting on the training data, should show high accuracy as long as the model was able to find effective splits to differentiate the classes.

Answer to Question 4

```
pred.val <- predict(mod1,values(r1))
r.pred <- raster(r1)
r.pred[] <- pred.val
writeRaster(r.pred,filename = "Frascati_Classy.tif",
            format="GTiff",overwrite=T)
rasterVis::levelplot(r.pred)
```



Answer to Question 5

I would expect that the same classification model would be able to accurately predict a similar image taken years later at the same time of year. Since this scene already has significant urbanization, it's unlikely that later development would introduce new land cover classes that would confuse the prediction more. Also, since it's the same time of year, you could also expect that the reflectance values would be similar enough that the land cover classes present in the new image would still resemble those present in the original image well enough to get an accurate prediction.

Answer to Question 6

If the later image was taken in late Fall, the prediction would not be as accurate. The most important factor in this is that the vegetation will have changed in response to the colder weather, like deciduous trees losing their leaves, causing their spectral response to shift and make the predictions for that class unreliable. Consider nearly half of the above prediction was "Veg", this alone would reduce the overall accuracy significantly.

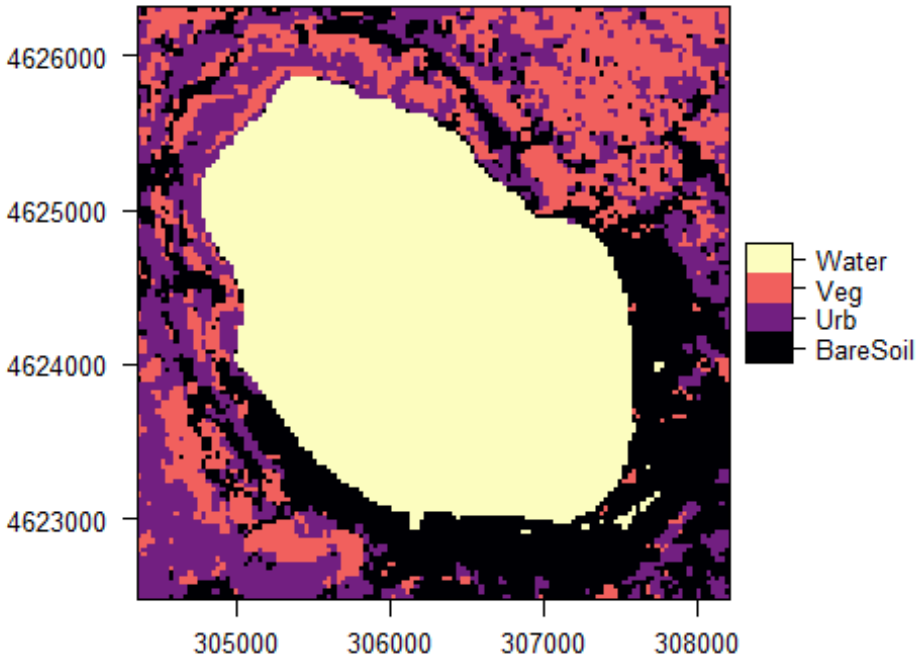
Answer to Question 7

You most likely could use the same training shapefiles for the image in Q6 as the one used for the existing prediction. The current areas being covered by the training shapefiles are

currently all a single class, and with the image already showing established urbanization, it seems reasonable to expect an image taken only a few years later would still be relatively homogeneous under each polygon.

Answer to Question 8

```
e <- extent(304372.3,308191.1,4622483,4626302) # taken from drawExtent()
r.pred.c <- crop(r.pred,e)
rasterVis::levelplot(r.pred.c)
```



Answer to Question 9 (Homework)

1. Create a polygon from the extent of the above cropped prediction raster
2. Bring the polygon into QGIS
3. Use the Random Points tool (Vector > Research Tools > Random Points Inside Polygons) to create 100 points and save the result as a shapefile
4. Attribute each of the random points to have the predicted class the point appears on as a “predicted” field
5. Compare each point to high-resolution (<=30m) imagery from the same time period and add the true land cover class as a “truth” field
6. Bring the points into R and create two vectors from the “predicted” and “truth” fields
7. Create the confusion matrix of the classification from these two vectors
8. Calculate accuracy statistics