GEM 520 Lab 6 - Supervised Image Classification

Agness Chisale

30 October 2025

Contents

The R code to analyze the polygons that you have delineated, split the data into training and validation sets, classify the image with the Maximum Likelihood algorithm and generate the confusion matrix is provided. You just need to set the file path to your `classification_polygons_YourInitials.shp` at the beginning of PART 3.

However, you will have to type your own code here to calculate the overall accuracy, user accuracy and producer accuracy from the confusion matrix. **Note that the overall accuracy must be more than 75% to receive full marks**.

Make also sure to answer Q1, Q2, Q3, Q4 and Q5 in your R Markdown report.

This document reads in the Landsat imagery and the delineated polygons with relative file paths from the root directory of the lab folder, where the Rmd file is located. Make sure that `classification_polygons_YourInitials` files are stored in the `outputs` folder and that your Rmd file is located at the root of the directory

PART 1 - Supervised Classification

See PDF

PART 2 - Defining areas of training data

See PDF

PART 3 - Image classification and accuracy assessment

SET THE FILE NAME OF YOUR `classification_polygons_YourInital.shp` file in the following code chunk

```
my_polygons <- "outputs/classification_polygons_AC.shp"
```

The following packages need to be installed on your machine. *DO NOT install the packages in the R Markdown document. Run the install.packages() commands in the console, in a fresh and clean R Studio session*

```
install.packages('terra', repos='https://rspatial.r-universe.dev')
install.packages("sf")
install.packages("RStoolbox")
install.packages("tidyverse")
install.packages("caret")
```

Once installed, we attach the packages

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.5.1

## Warning: package 'ggplot2' was built under R version 4.5.1

## Warning: package 'tibble' was built under R version 4.5.1

## Warning: package 'tidyr' was built under R version 4.5.1

## Warning: package 'readr' was built under R version 4.5.1

## Warning: package 'purrr' was built under R version 4.5.1

## Warning: package 'dplyr' was built under R version 4.5.1

## Warning: package 'forcats' was built under R version 4.5.1

## Warning: package 'lubridate' was built under R version 4.5.1
```

```
library(terra)
```

```
## Warning: package 'terra' was built under R version 4.5.1
```

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.5.1
```

```r
library(RStoolbox)
```

```
## Warning: package 'RStoolbox' was built under R version 4.5.1
```

We start by reading the Landsat image into R

```r
ls_image <- rast("data/LC09_L2SP_047026_20240716_20240717_02_T1_SR_BSTACK.tif")

ls_image
```

```
## class       : SpatRaster
## size        : 1407, 1194, 6  (nrow, ncol, nlyr)
## resolution  : 30, 30  (x, y)
## extent      : 466215, 502035, 5371545, 5413755  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS 84 / UTM zone 10N (EPSG:32610)
## source      : LC09_L2SP_047026_20240716_20240717_02_T1_SR_BSTACK.tif
## names       : blue, green, red, nir, swir1, swir2
## min values  :    0,     0,   0,   0,     0,     0
## max values  :    1,     1,   1,   1,     1,     1
```

```r
terra::plotRGB(ls_image, r = 3, g = 2, b = 1, stretch = "lin")
```

Then, we load the delineated polygons

```
class_poly <- st_read(my_polygons)
```

```
## Reading layer 'classification_polygons_AC' from data source
##   'C:\Users\achisale.stu\Documents\MGEM\GEM 520\Labs\Lab 6\GEM520_Lab6_classification\outputs\classi
##   using driver 'ESRI Shapefile'
## Simple feature collection with 271 features and 1 field
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 466215.2 ymin: 5371513 xmax: 501997.3 ymax: 5413746
## Projected CRS: WGS 84 / UTM zone 10N
```

```
# Make sure that the geometry is valid
class_poly <- st_make_valid(class_poly)

# Tranform lc_class to factor
class_poly <- class_poly %>%
  mutate(lc_class = factor(lc_class,
                           levels = c("Broadleaf Forest", "Coniferous Forest", "Exposed soil and rocks"

# Plot
terra::plotRGB(ls_image, r = 3, g = 2, b = 1, stretch = "lin")
plot(class_poly[, "lc_class"], add = TRUE)
```

Here is a summary of the number of polygons per class

```
poly_summary <- class_poly %>%
  st_drop_geometry() %>%
  group_by(lc_class) %>%
  summarize(n_poly = n())

poly_summary
```

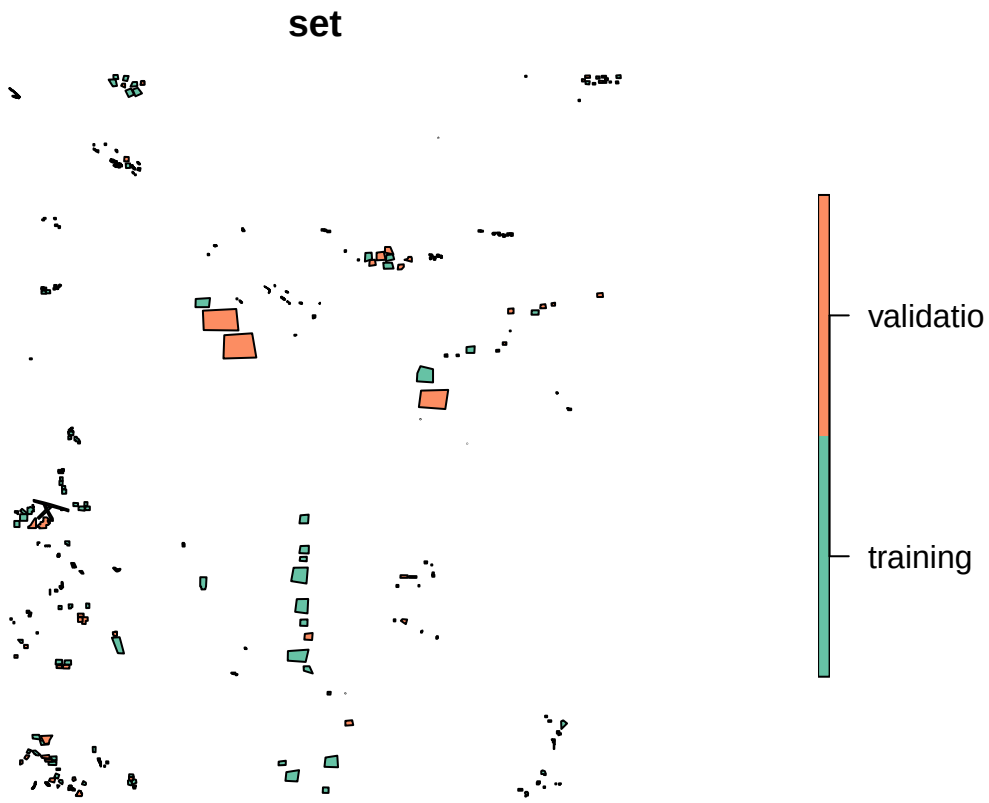
```
## # A tibble: 7 x 2
##   lc_class                n_poly
##   <fct>                    <int>
## 1 Broadleaf Forest            33
## 2 Coniferous Forest           33
## 3 Exposed soil and rocks      35
## 4 High density developed      50
## 5 Low density developed       36
## 6 Non-forest vegetation       33
## 7 Water                       51
```

For each land cover class will use 70% of the polygons to train the classification algorithm and the remaining 30% for validation.

```
# Assign a unique ID to each polygon
class_poly <- tibble::rowid_to_column(class_poly, var = "ID")

set.seed(1234)

# Sample 70% of the polygons in each land cover class
poly_train <- class_poly %>%
  group_by(lc_class) %>%
  sample_frac(0.7) %>%
  mutate(set = "training") %>% st_cast(to = 'POLYGON')
```

```
## Warning in st_cast.MULTIPOLYGON(X[[i]], ...): polygon from first part only
```

```
# Use the ID field to select the polygons for validation
poly_val <- class_poly %>%
  filter(!ID %in% poly_train$ID) %>%
  mutate(set = "validation") %>% st_cast(to = 'POLYGON')

poly_set <- rbind(poly_train,
                  poly_val)

# Plot poly_set
plot(poly_set[, "set"], key.width = lcm(3.2))
```



We now extract the values of the Landsat image pixels in the polygons

```
# Add join column to our polygon set and remove the old one
poly_set <- poly_set %>% tibble::rowid_to_column(var = "join_id") %>% select(-ID)

# Extract pixel values for each polygon
poly_set_vals <- terra::extract(ls_image, vect(poly_set))

# Join with LC class via an inner_join (where new join_id == ID from terra)
poly_set_vals <- inner_join(poly_set, poly_set_vals, join_by(join_id == ID)) %>%
  st_drop_geometry()
```

Each row of `poly_set_vals` corresponds to one pixel of the Landsat image.

```
poly_set_vals
```

```
## # A tibble: 24,611 x 9
## # Groups:   lc_class [7]
##    join_id lc_class         set        blue  green    red   nir  swir1  swir2
##  *   <dbl> <fct>            <chr>     <dbl>  <dbl>  <dbl> <dbl>  <dbl>  <dbl>
##  1       1 Broadleaf Forest training 0.0141 0.0410 0.0228 0.366 0.122  0.0504
##  2       1 Broadleaf Forest training 0.0149 0.0383 0.0232 0.313 0.115  0.0476
##  3       1 Broadleaf Forest training 0.0111 0.0300 0.0162 0.316 0.0943 0.0349
##  4       1 Broadleaf Forest training 0.0136 0.0382 0.0223 0.384 0.129  0.0501
##  5       1 Broadleaf Forest training 0.0143 0.0382 0.0230 0.357 0.118  0.0484
```

```
##  6       1 Broadleaf Forest training 0.0179 0.0471 0.0292 0.417 0.172  0.0694
##  7       1 Broadleaf Forest training 0.0176 0.0427 0.0299 0.388 0.159  0.0639
##  8       1 Broadleaf Forest training 0.0181 0.0483 0.0311 0.420 0.171  0.0710
##  9       1 Broadleaf Forest training 0.0198 0.0498 0.0334 0.423 0.180  0.0771
## 10       1 Broadleaf Forest training 0.0166 0.0407 0.0233 0.335 0.131  0.0541
## # i 24,601 more rows
```

We can check the number of pixels per class and training / validation set

```
poly_stats <- poly_set_vals %>%
  group_by(set, lc_class) %>%
  summarize(n_px = n())
```

```
## `summarise()` has grouped output by 'set'. You can override using the `.groups`
## argument.
```

```
poly_stats
```

```
## # A tibble: 14 x 3
## # Groups:   set [2]
##    set        lc_class                n_px
##    <chr>      <fct>                  <int>
##  1 training   Broadleaf Forest         322
##  2 training   Coniferous Forest       1553
##  3 training   Exposed soil and rocks   299
##  4 training   High density developed  1835
##  5 training   Low density developed   1001
##  6 training   Non-forest vegetation   1624
##  7 training   Water                   6389
##  8 validation Broadleaf Forest         125
##  9 validation Coniferous Forest        942
## 10 validation Exposed soil and rocks   125
## 11 validation High density developed  1128
## 12 validation Low density developed    752
## 13 validation Non-forest vegetation    502
## 14 validation Water                   8014
```

We can pivot the data from a wide to long format

```
poly_set_vals_long <- pivot_longer(poly_set_vals,
                                   blue:swir2,
                                   names_to = "band",
                                   values_to = "reflectance")

poly_set_vals_long
```

```
## # A tibble: 147,666 x 5
## # Groups:   lc_class [7]
##    join_id lc_class         set      band  reflectance
##      <dbl> <fct>            <chr>    <chr>       <dbl>
##  1       1 Broadleaf Forest training blue       0.0141
```

```
##  2       1 Broadleaf Forest training green      0.0410
##  3       1 Broadleaf Forest training red        0.0228
##  4       1 Broadleaf Forest training nir        0.366
##  5       1 Broadleaf Forest training swir1      0.122
##  6       1 Broadleaf Forest training swir2      0.0504
##  7       1 Broadleaf Forest training blue       0.0149
##  8       1 Broadleaf Forest training green      0.0383
##  9       1 Broadleaf Forest training red        0.0232
## 10       1 Broadleaf Forest training nir        0.313
## # i 147,656 more rows
```

And calculate some summary statistics for each band and land cover class: mean, 5^{th} quantile and 95^{th} quantile of reflectance.

```
spectral_sign <- poly_set_vals_long %>%
  group_by(lc_class, band) %>%
  summarize(r_mean = mean(reflectance, na.rm = TRUE),
            r_q05 = quantile(reflectance, 0.05, na.rm = TRUE),
            r_q95 = quantile(reflectance, 0.95, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'lc_class'. You can override using the
## `.groups` argument.
```

```
spectral_sign
```

```
## # A tibble: 42 x 5
## # Groups:   lc_class [7]
##    lc_class          band  r_mean   r_q05  r_q95
##    <fct>             <chr>  <dbl>   <dbl>  <dbl>
##  1 Broadleaf Forest  blue  0.0212 0.0130  0.0369
##  2 Broadleaf Forest  green 0.0424 0.0309  0.0651
##  3 Broadleaf Forest  nir   0.365  0.281   0.443
##  4 Broadleaf Forest  red   0.0298 0.0159  0.0713
##  5 Broadleaf Forest  swir1 0.150  0.101   0.249
##  6 Broadleaf Forest  swir2 0.0625 0.0367  0.122
##  7 Coniferous Forest blue  0.0116 0.00947 0.0145
##  8 Coniferous Forest green 0.0290 0.0247  0.0334
##  9 Coniferous Forest nir   0.244  0.207   0.288
## 10 Coniferous Forest red   0.0153 0.0115  0.0194
## # i 32 more rows
```

We can now visualize the spectral signature of each land cover class

```
# Wavelength corresponding to each band
bands_wavelength <- read_csv("data/bands_wavelength.csv")
```

```
## Rows: 6 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): band
## dbl (1): wavelength
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

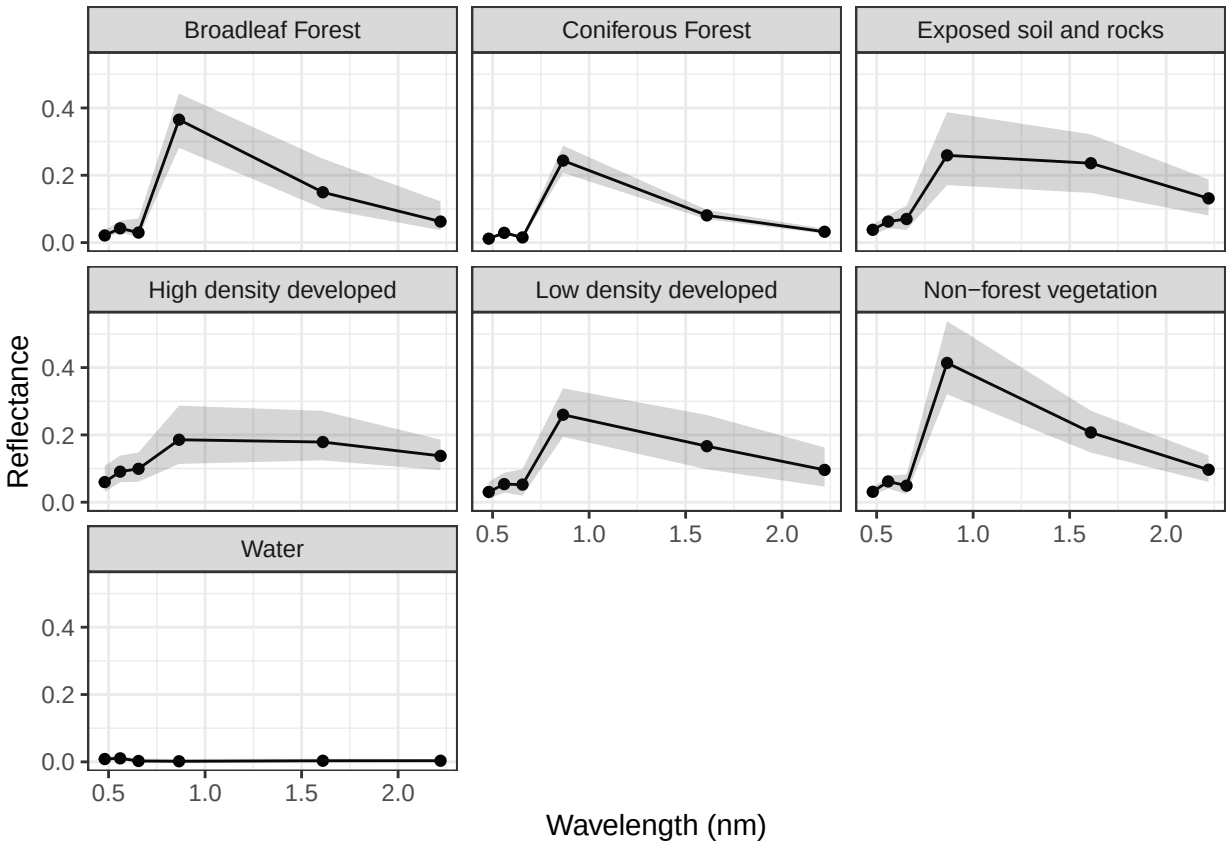
```r
bands_wavelength
```

```
## # A tibble: 6 x 2
##   band  wavelength
##   <chr>      <dbl>
## 1 blue       0.48
## 2 green      0.56
## 3 red        0.655
## 4 nir        0.865
## 5 swir1      1.61
## 6 swir2      2.22
```

```r
# Join wavelength
spectral_sign <- inner_join(spectral_sign, bands_wavelength)
```

```
## Joining with `by = join_by(band)`
```

```r
# Graph
ggplot(spectral_sign, aes(x = wavelength, y = r_mean, group = 1)) +
  geom_point() +
  geom_line() +
  geom_ribbon(aes(ymin = r_q05, ymax = r_q95), alpha = 0.2) +
  facet_wrap(vars(lc_class)) +
  theme_bw() +
  labs(x = "Wavelength (nm)",
       y = "Reflectance")
```



We can now use the function `superClass()` from the `RSToolbox` package to perform the classification and accuracy assessment. The argument `model = "mlc"` is used to select the Maximum Likelihood algorithm for classification. We provide the polygons used for training and validation under the arguments `trainData` and `valData`. The function will sample `500` pixels from the training polygons (argument `nSamples = 500`) per land cover class and use this sample to train the classification algorithm. Similarly, validation will be performed on a sample of `500` pixels of the validation polygons per land cover class. Note that for some classes there might not be enough pixels to reach `500` observations. In that case, training / validation would be performed on $<$ `500` pixels per land cover class. It is important to try to balance the number of observation per classes before training the classification model and assessing its accuracy. Otherwise, the overall accuracy derived from the confusion matrix could be biased towards the classes with the largest number of observations.

For example, it is easier to draw large polygons across water (and hence getting a lot of training/validation data) than it is over broadleaf forest. Let's assume that we use 1000 pixels to assess the accuracy of water and 20 pixels to assess the accuracy of broadleaf vegetation. For water, 900 out of 1000 pixels (90%) were classified correctly whereas only 5 out of 20 pixels were classified correctly for broadleaf.

The overall accuracy would be:

```
OA_unbalanced <- (900 + 5) / (1000 + 20)

OA_unbalanced
```

```
## [1] 0.8872549
```

Now let's assume that 20 pixels were used for both water and broadleaf to assess the classification accuracy. For water, 18 out of 20 pixels (still 90%) were classified correctly and 5 out of 20 pixels were classified correctly for broadleaf.

The overall accuracy would be:

```
OA_balanced <- (18 + 5) / (20 + 20)

OA_balanced
```

```
## [1] 0.575
```

In the first case where classes are unbalanced we obtain an overall accuracy of 89%. Because there are much more pixels in water than in broafleaf in the first case, the overall accuracy is very high and doesn't reflect the fact that the broadleaf class is poorly classified. In the second case, where classes are balanced, we obtain an overall accuracy of 58%. This illustrates that the classification is not as good as we might think with the first case.

The superClass() function might take a few minutes to run

```
set.seed(1234)

poly_train <- poly_train %>% rename(class = lc_class)
poly_val <- poly_val %>% rename(class = lc_class)

mlc_model <- superClass(img = ls_image,
                        trainData = poly_train,
                        valData = poly_val,
                        responseCol = "class",
                        model = "mlc",
                        nSamples = 500)
```

```
## Warning: package 'caret' was built under R version 4.5.1
```

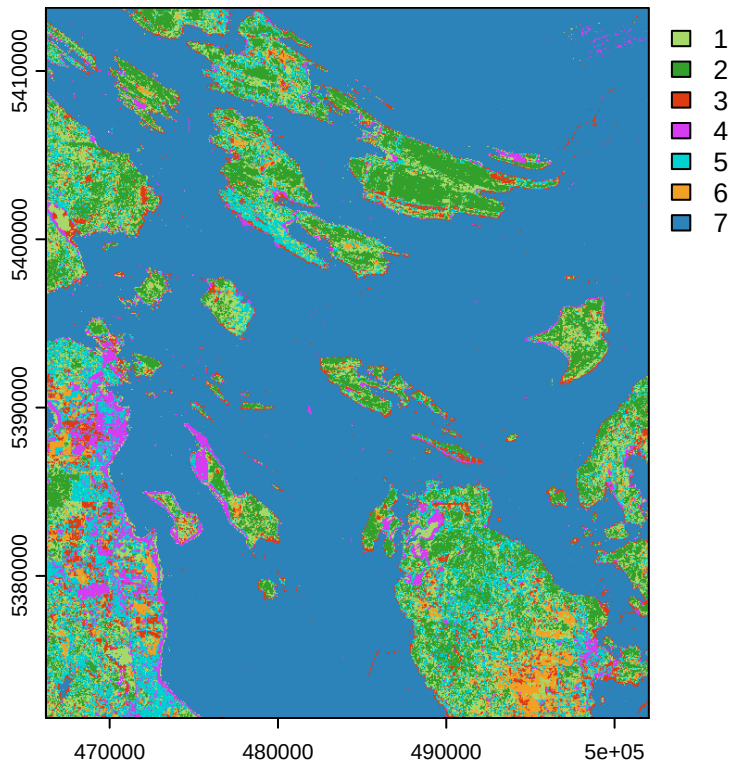
```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

The `superClass()` function returns a list with multiple objects. The classified map is stored in the element of the list called `map`. The trained model is stored in the element `model`. The predictions of the model at the validation data are stored in a data.frame called `validationSamples` located in the element of `mlc_model` called `validation`. The column `reference` is the land cover class that you have assigned and the column `prediction` is the land cover class predicted by the model.

```
classified_map <- mlc_model$map

# Write the classified map as a tif file
writeRaster(classified_map,
            filename = "outputs/classified_map.tif",
            overwrite = TRUE)

# Plot with colors
plot(classified_map,
     col = c('#A6D96A','#33A02C','#DE3B13','#D63CF1','#00D2D2','#F1A026','#2B83BA'))
```




```
# Validation df
val_preds <- mlc_model$validation$validationSamples

head(val_preds)
```

```
## Simple feature collection with 6 features and 3 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 473940 ymin: 5413200 xmax: 474090 ymax: 5413320
## Projected CRS: WGS 84 / UTM zone 10N
##               reference            prediction  cell               geometry
## 1 Non-forest vegetation      Broadleaf Forest 18170 POINT (474000 5413290)
## 2 Non-forest vegetation      Broadleaf Forest 16978 POINT (474060 5413320)
## 3 Non-forest vegetation Non-forest vegetation 20560 POINT (474060 5413230)
## 4 Non-forest vegetation Non-forest vegetation 16979 POINT (474090 5413320)
## 5 Non-forest vegetation      Broadleaf Forest 21750 POINT (473940 5413200)
## 6 Non-forest vegetation Non-forest vegetation 20557 POINT (473970 5413230)
```

The confusion matrix can be created from `val_preds` using the `table()` function (base R package). The columns represent the reference classes (classes you have assigned) and the rows represent the predictions of the model.

```
conf_matrix <- table(st_drop_geometry(val_preds[, c("prediction", "reference")]))

knitr::kable(conf_matrix)
```

	Broadleaf Forest	Coniferous Forest	Exposed soil and rocks	High density developed	Low density developed	Non-forest vegetation	Water
Broadleaf Forest	69	2	3	0	48	67	0
Coniferous Forest	0	748	0	0	14	0	0
Exposed soil and rocks	0	0	41	23	30	0	0
High density developed	0	0	0	751	84	0	0
Low density developed	0	0	16	105	302	4	1
Non-forest vegetation	9	0	0	2	15	355	0
Water	0	0	0	0	0	0	1667

```
conf_matrix
```

```
##                          reference
## prediction               Broadleaf Forest Coniferous Forest
##   Broadleaf Forest                     69                 2
##   Coniferous Forest                     0               748
##   Exposed soil and rocks                0                 0
##   High density developed                0                 0
##   Low density developed                 0                 0
##   Non-forest vegetation                 9                 0
##   Water                                 0                 0
##                          reference
## prediction               Exposed soil and rocks High density developed
##   Broadleaf Forest                            3                      0
##   Coniferous Forest                           0                      0
##   Exposed soil and rocks                     41                     23
##   High density developed                      0                    751
##   Low density developed                      16                    105
##   Non-forest vegetation                       0                      2
##   Water                                       0                      0
##                          reference
## prediction               Low density developed Non-forest vegetation Water
##   Broadleaf Forest                          48                     67     0
##   Coniferous Forest                         14                      0     0
##   Exposed soil and rocks                    30                      0     0
##   High density developed                    84                      0     0
##   Low density developed                    302                      4     1
##   Non-forest vegetation                     15                    355     0
##   Water                                      0                      0  1667
```

The `table()` function returns a `matrix` object with the predicted classes in the rows and the reference classes in the columns. The diagonal of a matrix can be returned as a `vector` using the `diag()` function. The row and column sums can be returned with the `rowSums()` and `colSums()` functions. The sum of all elements of a matrix can be obtained with the `sum()` function.

Use the functions described above to calculate the overall accuracy (`OA`), producer accuracy (`PA`) and user accuracy (`UA`) of your classification.

Note that the overall accuracy must be more than 75% to receive full marks

Code answer

```
# Calculate accuracy metrics here
# Assuming you have a confusion matrix
#conf_matrix <- table(predicted, reference)
# Overall Accuracy
OA <- sum(diag(conf_matrix)) / sum(conf_matrix)

# Producer Accuracy
PA <- diag(conf_matrix) / colSums(conf_matrix)

# User Accuracy
UA <- diag(conf_matrix) / rowSums(conf_matrix)

# Print results
print(paste("Overall Accuracy:", round(OA, 3)))
```

```
## [1] "Overall Accuracy: 0.903"
```

```
print("Producer Accuracy:")
```

```
## [1] "Producer Accuracy:"
```

```
print(round(PA, 3))
```

```
##       Broadleaf Forest      Coniferous Forest Exposed soil and rocks
##                  0.885                  0.997                  0.683
## High density developed  Low density developed  Non-forest vegetation
##                  0.852                  0.613                  0.833
##                  Water
##                  0.999
```

```
print("User Accuracy:")
```

```
## [1] "User Accuracy:"
```

```
print(round(UA, 3))
```

```
##       Broadleaf Forest      Coniferous Forest Exposed soil and rocks
##                  0.365                  0.982                  0.436
## High density developed  Low density developed  Non-forest vegetation
##                  0.899                  0.706                  0.932
##                  Water
##                  1.000
```

Answer the following questions in brief answers (1-5 lines each)

Question 1

What is a supervised classification process and how does it differs from unsupervised classification? Supervised classification uses training data selected by the analyst to guide how the software assigns pixels to known land cover classes.It differs from unsupervised classification, where the software automatically groups pixels by spectral similarity and the analyst labels them afterwards.

Question 2

Why do we need distinct training and validation data, and why is important they do not overlap?

Distinct training and validation data ensure that model accuracy is tested on independent samples. If they overlap, the classifier would be tested on data it has already seen, leading to biased accuracy results.

Question 3

How does Maximum Likelihood classification work? Maximum Likelihood classification assumes that the data for each class follow a normal (Gaussian) distribution. It assigns each pixel to the class where it has the highest probability of belonging, based on its spectral values.

Question 4

Which classes achieved the worst producer and user accuracies, why do you think this is? Broadleaf Forest had the lowest user accuracy and Low-density Developed had the lowest producer accuracy . The low accuracy for Broadleaf Forest likely occurred because its spectral reflectance overlapped with nearby non-forest vegetation and coniferous areas, causing misclassification. Similarly, low-density developed zones contain mixed surfaces—vegetation, bare soil, and small buildings, creating spectral confusion.

Question 5

Which classes achieved the best results, why do you think this is? Water and coniferous forest achieved the highest accuracies. These classes have distinct and consistent spectral patterns. Water strongly absorbs in NIR and SWIR bands, while coniferous forests show a unique dark green reflectance due to dense canopy and needle structure, reducing confusion with other landcover types.

Part 4 - Classification Examination

Now, load the classified map `outputs/classified_map.tif` back into QGIS. Navigate around the classified image and compare it to the original image. Ask yourself: was this a successful classification? Does the distribution of classes make sense? Are there any areas of major misclassification? Would you feel comfortable and confident passing along this map to someone who would then make "real world" decisions based off of it? In order to answer these questions (in your head... these are not actual graded questions), feel free to review the Google Satellite imagery with this classified map.