Remote sensing for land classification

Namesh Nazar

1/3/2021

Introduction

This report uses machine learning approaches to predict land classification using multispectral satellite data. This is a supervised machine learning classification problem. The dataset has been donated by University of Strathclyde, Glasgow. The application of machine learning to analyze multispectral satellite data has immense potential to gather real-time and historic data for research and monitoring in the areas of crop sensing, urban planning, water management, climate change, and forecasting and management of disasters etc.

Goal of the project

The aim of the analysis is to use multi-spectral data to predict land classification.

Dataset

The source of the data is Landsat Multi Spectral Scanner (MSS) imagery. Landsat is **moderate spatial-resolution (30-meter) imagery** that provides large areas of repeated data coverage at a scale that enables users to see detailed human-scale processes, such as urbanization, but not individual houses. The MSS provides **four spectral bands** of Earth surface reflection in the visible and near-infrared regions of the electromagnetic spectrum. One frame of Landsat MSS imagery consists of *four digital images (made of pixels) of the same scene in different spectral bands.* In case of Landsat MSS imagery, the resolution of the image is such that each pixel is about 80m x 80m. Each image contains 2340 x 3380 such pixels.

Our data set consists of a small sample (82 x 100 pixels) of Landsat MSS data. For each pixel, data for four spectral bands is provided to make predictions about the land classification. Documentation provided with the data set is used to better understand it. The data set is available at UCI Machine Learning Repository on the following link: https://archive.ics.uci.edu/ml/datasets/Statlog+%28Landsat+Satellite%29

The data is provided in two parts: a training set and a test set. Since the sample is so small the donor of the data already alerted users to not use cross-validation for training of models. Parts of the data have been removed and the order has been randomized to avoid possibility of recreating of original image. The binary pixel data has also been converted into ASCII format already and classifications have been verified through site visits.

Steps for loading data:

- Load data from url into a variable.
- Identify the class of the data and use appropriate function to turn it into a dataframe
- Look at the dataframe to confirm that it is ready to use.

```r
# We load the data sets
url_train <- "https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/satimage/sat.trn"

# We see what the data looks like.
class(url_train)
```

```
## [1] "character"
```

```r
read_lines(url_train, n_max = 3)
```

```
## [1] "92 115 120 94 84 102 106 79 84 102 102 83 101 126 133 103 92 112 118 85 84 103 104 81 102 126 13
## [2] "84 102 106 79 84 102 102 83 80 102 102 79 92 112 118 85 84 103 104 81 84 99 104 78 88 121 128 10
## [3] "84 102 102 83 80 102 102 79 84 94 102 79 84 103 104 81 84 99 104 78 84 99 104 81 84 107 113 87 8
```

```r
# We find that the data is separated by space. We therefore use read.table
# function to turn it into a dataframe.
train_set <- read.table(url_train)

# We then have a look at the data and structure to see if it worked.
str(train_set)
```

```
## 'data.frame':    4435 obs. of  37 variables:
##  $ V1 : int  92 84 84 80 84 80 76 76 76 76 ...
##  $ V2 : int  115 102 102 102 94 94 102 102 89 94 ...
##  $ V3 : int  120 106 102 102 102 98 106 106 98 98 ...
##  $ V4 : int  94 79 83 79 79 76 83 87 76 76 ...
##  $ V5 : int  84 84 80 84 80 80 76 80 76 76 ...
##  $ V6 : int  102 102 102 94 94 102 102 98 94 98 ...
##  $ V7 : int  106 102 102 102 98 102 106 106 98 102 ...
##  $ V8 : int  79 83 79 79 76 79 87 79 76 72 ...
##  $ V9 : int  84 80 84 80 80 76 80 76 76 76 ...
##  $ V10: int  102 102 94 94 102 102 98 94 98 94 ...
##  $ V11: int  102 102 102 98 102 102 106 102 102 90 ...
##  $ V12: int  83 79 79 76 79 79 79 76 72 76 ...
##  $ V13: int  101 92 84 84 84 76 80 80 80 76 ...
##  $ V14: int  126 112 103 99 99 99 107 112 95 91 ...
##  $ V15: int  133 118 104 104 104 104 118 118 104 104 ...
##  $ V16: int  103 85 81 78 81 81 88 88 74 74 ...
##  $ V17: int  92 84 84 84 76 76 80 80 76 76 ...
##  $ V18: int  112 103 99 99 99 99 112 107 91 95 ...
##  $ V19: int  118 104 104 104 104 108 118 113 104 100 ...
##  $ V20: int  85 81 78 81 81 85 88 85 74 78 ...
##  $ V21: int  84 84 84 76 76 76 80 80 76 76 ...
##  $ V22: int  103 99 99 99 99 103 107 95 95 91 ...
##  $ V23: int  104 104 104 104 108 118 113 100 100 100 ...
##  $ V24: int  81 78 81 81 85 88 85 78 78 74 ...
##  $ V25: int  102 88 84 84 84 84 79 79 75 75 ...
##  $ V26: int  126 121 107 99 99 103 107 103 91 91 ...
##  $ V27: int  134 128 113 104 104 104 113 104 96 96 ...
##  $ V28: int  104 100 87 79 79 79 87 83 75 71 ...
##  $ V29: int  88 84 84 84 84 79 79 79 75 79 ...
##  $ V30: int  121 107 99 99 103 107 103 103 91 87 ...
```

```
##  $ V31: int  128 113 104 104 104 109 104 104 96 93 ...
##  $ V32: int  100 87 79 79 79 87 83 79 71 71 ...
##  $ V33: int  84 84 84 84 79 79 79 79 79 79 ...
##  $ V34: int  107 99 99 103 107 107 103 95 87 87 ...
##  $ V35: int  113 104 104 104 109 109 104 100 93 93 ...
##  $ V36: int  87 79 79 79 87 87 79 79 71 67 ...
##  $ V37: int  3 3 3 3 3 3 3 3 4 4 ...
```

```
# We do the same steps for test data
url_test <- "https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/satimage/sat.tst"
read_lines(url_test, n_max = 3)
test_set <- read.table(url_test)
str(test_set)
```

Description of variables

Each line of data corresponds to a ***9 pixels in 3x3 square neighbourhood of pixels***. For each pixel ***four spectral values*** are provided. First, the four spectral values for the top-left pixel are given, followed by the four spectral values for the top-middle pixel and then those for the top-right pixel,and so on with the pixels read out in sequence left-to-right and top-to-bottom. Finally, each row also contains a column on the ***land classification labels*** of the central pixel. All data is in the form of integers.

NUMBER OF EXAMPLES

- training set 4435
- test set 2000

NUMBER OF ATTRIBUTES: 36 (= 4 spectral bands x 9 pixels in neighbourhood)

CLASS: There are 6 decision classes: 1,2,3,4,5 and 7.

Key steps to be performed

The following key steps will be performed in the data:

1. Load data from source and convert it into a data frame to be used for analysis.
2. Process the data frame for ease of analysis. This includes giving descriptive column names and classification labels. Columns to be used for analysis will also be selected to reduce computation required to process the data.
3. Visualize the data to better understand variables.
4. Perform analysis through different models to predict classification.
5. Report on the best performing model.

Method and Analysis

Data processing

The data was provided in the form of text separated by space which can easily be converted into data frame for analysis. Once we have the data frame we provide the columns descriptive names. From the documentations provided with the dataset we know that ***each column corresponds to one (out of 4)***

spectral value for one (out of 9) pixel. The names given to the columns include the position of the pixel and a number (1-4) for four spectral values. The final column is the land classification label which is converted from numbers (1-7) to text that describes the labels.

The land classification labels are as follows:

1. red soil
2. cotton crop
3. grey soil
4. damp grey soil
5. soil with vegetation stubble
6. mixture class (all types present)
7. very damp grey soil

```
train_set <- train_set %>% setNames(c("Top_left_spec1", "Top_left_spec2", "Top_left_spec3",
    "Top_left_spec4", "Top_center_spec1", "Top_center_spec2", "Top_center_spec3",
    "Top_center_spec4", "Top_right_spec1", "Top_right_spec2", "Top_right_spec3",
    "Top_right_spec4", "Middle_left_spec1", "Middle_left_spec2", "Middle_left_spec3",
    "Middle_left_spec4", "Middle_center_spec1", "Middle_center_spec2", "Middle_center_spec3",
    "Middle_center_spec4", "Middle_right_spec1", "Middle_right_spec2", "Middle_right_spec3",
    "Middle_right_spec4", "Bottom_left_spec1", "Bottom_left_spec2", "Bottom_left_spec3",
    "Bottom_left_spec4", "Bottom_center_spec1", "Bottom_center_spec2", "Bottom_center_spec3",
    "Bottom_center_spec4", "Bottom_right_spec1", "Bottom_right_spec2", "Bottom_right_spec3",
    "Bottom_right_spec4", "Classification_code"))

# We give the classifications descriptive names as well.
train_set <- train_set %>% mutate(Classification = ifelse(Classification_code ==
    1, "red soil", ifelse(Classification_code == 2, "cotton crop", ifelse(Classification_code ==
    3, "grey soil", ifelse(Classification_code == 4, "damp grey soil", ifelse(Classification_code ==
    5, "soil with vegetation stubble", ifelse(Classification_code == 6, "mixture ",
    "very damp grey soil")))))))

str(train_set)
```

```
## 'data.frame':    4435 obs. of  38 variables:
##  $ Top_left_spec1     : int  92 84 84 80 84 80 76 76 76 76 ...
##  $ Top_left_spec2     : int  115 102 102 102 94 94 102 102 89 94 ...
##  $ Top_left_spec3     : int  120 106 102 102 102 98 106 106 98 98 ...
##  $ Top_left_spec4     : int  94 79 83 79 79 76 83 87 76 76 ...
##  $ Top_center_spec1   : int  84 84 80 84 80 80 76 80 76 76 ...
##  $ Top_center_spec2   : int  102 102 102 94 94 102 102 98 94 98 ...
##  $ Top_center_spec3   : int  106 102 102 102 98 102 106 106 98 102 ...
##  $ Top_center_spec4   : int  79 83 79 79 76 79 87 79 76 72 ...
##  $ Top_right_spec1    : int  84 80 84 80 80 76 80 76 76 76 ...
##  $ Top_right_spec2    : int  102 102 94 94 102 102 98 94 98 94 ...
##  $ Top_right_spec3    : int  102 102 102 98 102 102 106 102 102 90 ...
##  $ Top_right_spec4    : int  83 79 79 76 79 79 79 76 72 76 ...
##  $ Middle_left_spec1  : int  101 92 84 84 84 76 80 80 80 76 ...
##  $ Middle_left_spec2  : int  126 112 103 99 99 99 107 112 95 91 ...
##  $ Middle_left_spec3  : int  133 118 104 104 104 104 118 118 104 104 ...
##  $ Middle_left_spec4  : int  103 85 81 78 81 81 88 88 74 74 ...
##  $ Middle_center_spec1: int  92 84 84 84 76 76 80 80 76 76 ...
##  $ Middle_center_spec2: int  112 103 99 99 99 99 112 107 91 95 ...
##  $ Middle_center_spec3: int  118 104 104 104 104 108 118 113 104 100 ...
```

```
##  $ Middle_center_spec4: int  85 81 78 81 81 85 88 85 74 78 ...
##  $ Middle_right_spec1 : int  84 84 84 76 76 76 80 80 76 76 ...
##  $ Middle_right_spec2 : int  103 99 99 99 99 103 107 95 95 91 ...
##  $ Middle_right_spec3 : int  104 104 104 104 108 118 113 100 100 100 ...
##  $ Middle_right_spec4 : int  81 78 81 81 85 88 85 78 78 74 ...
##  $ Bottom_left_spec1  : int  102 88 84 84 84 84 79 79 75 75 ...
##  $ Bottom_left_spec2  : int  126 121 107 99 99 103 107 103 91 91 ...
##  $ Bottom_left_spec3  : int  134 128 113 104 104 104 113 104 96 96 ...
##  $ Bottom_left_spec4  : int  104 100 87 79 79 79 87 83 75 71 ...
##  $ Bottom_center_spec1: int  88 84 84 84 84 79 79 79 75 79 ...
##  $ Bottom_center_spec2: int  121 107 99 99 103 107 103 103 91 87 ...
##  $ Bottom_center_spec3: int  128 113 104 104 104 109 104 104 96 93 ...
##  $ Bottom_center_spec4: int  100 87 79 79 79 87 83 79 71 71 ...
##  $ Bottom_right_spec1 : int  84 84 84 84 79 79 79 79 79 79 ...
##  $ Bottom_right_spec2 : int  107 99 99 103 107 107 103 95 87 87 ...
##  $ Bottom_right_spec3 : int  113 104 104 104 109 109 104 100 93 93 ...
##  $ Bottom_right_spec4 : int  87 79 79 79 87 87 79 79 71 67 ...
##  $ Classification_code: int  3 3 3 3 3 3 3 3 4 4 ...
##  $ Classification     : chr  "grey soil" "grey soil" "grey soil" "grey soil" ...
```

```
# We do the same for test set
test_set <- test_set %>% setNames(c("Top_left_spec1", "Top_left_spec2", "Top_left_spec3",
    "Top_left_spec4", "Top_center_spec1", "Top_center_spec2", "Top_center_spec3",
    "Top_center_spec4", "Top_right_spec1", "Top_right_spec2", "Top_right_spec3",
    "Top_right_spec4", "Middle_left_spec1", "Middle_left_spec2", "Middle_left_spec3",
    "Middle_left_spec4", "Middle_center_spec1", "Middle_center_spec2", "Middle_center_spec3",
    "Middle_center_spec4", "Middle_right_spec1", "Middle_right_spec2", "Middle_right_spec3",
    "Middle_right_spec4", "Bottom_left_spec1", "Bottom_left_spec2", "Bottom_left_spec3",
    "Bottom_left_spec4", "Bottom_center_spec1", "Bottom_center_spec2", "Bottom_center_spec3",
    "Bottom_center_spec4", "Bottom_right_spec1", "Bottom_right_spec2", "Bottom_right_spec3",
    "Bottom_right_spec4", "Classification_code"))
test_set <- test_set %>% mutate(Classification = ifelse(Classification_code == 1,
    "red soil", ifelse(Classification_code == 2, "cotton crop", ifelse(Classification_code ==
        3, "grey soil", ifelse(Classification_code == 4, "damp grey soil", ifelse(Classification_code ==
        5, "soil with vegetation stubble", ifelse(Classification_code == 6, "mixture ",
        "very damp grey soil")))))))

str(test_set)
```

Each row gives data on ***4 spectral values on a set of 9 pixels in 3x3 neighbourhood***. For ease of analysis we choose the central pixel for classification of land. This gives us *4 predictor, one for each spectral value*, to classify land into labels. The data documentation also mentions that the classification label is based on the location of central pixel.

```
# We know that the dataset provides 4 spectral values for 9 pixels in a 3x3 pixel
# set. For ease of analysis we use the middle pixels.

train_set <- train_set %>% select(c("Middle_center_spec1", "Middle_center_spec2",
    "Middle_center_spec3", "Middle_center_spec4", "Classification"))
str(train_set)
```

```
## 'data.frame':    4435 obs. of  5 variables:
##  $ Middle_center_spec1: int  92 84 84 84 76 76 80 80 76 76 ...
```

```
##  $ Middle_center_spec2: int  112 103 99 99 99 99 112 107 91 95 ...
##  $ Middle_center_spec3: int  118 104 104 104 104 108 118 113 104 100 ...
##  $ Middle_center_spec4: int  85 81 78 81 81 85 88 85 74 78 ...
##  $ Classification     : chr  "grey soil" "grey soil" "grey soil" "grey soil" ...
```

```
# Repeat same for test set
test_set <- test_set %>% select(c("Middle_center_spec1", "Middle_center_spec2", "Middle_center_spec3",
    "Middle_center_spec4", "Classification"))
str(test_set)
```

Data exploration and visualization

We start exploring the data by pulling out its summary. We can see that all the variables have different ranges but the mean and medians are not very off, which indicates our data may be approximated by a normal distribution. We also see that our classification variable is of class "Character". We would need to change that to factor when we start using machine learning algorithms from the caret package.

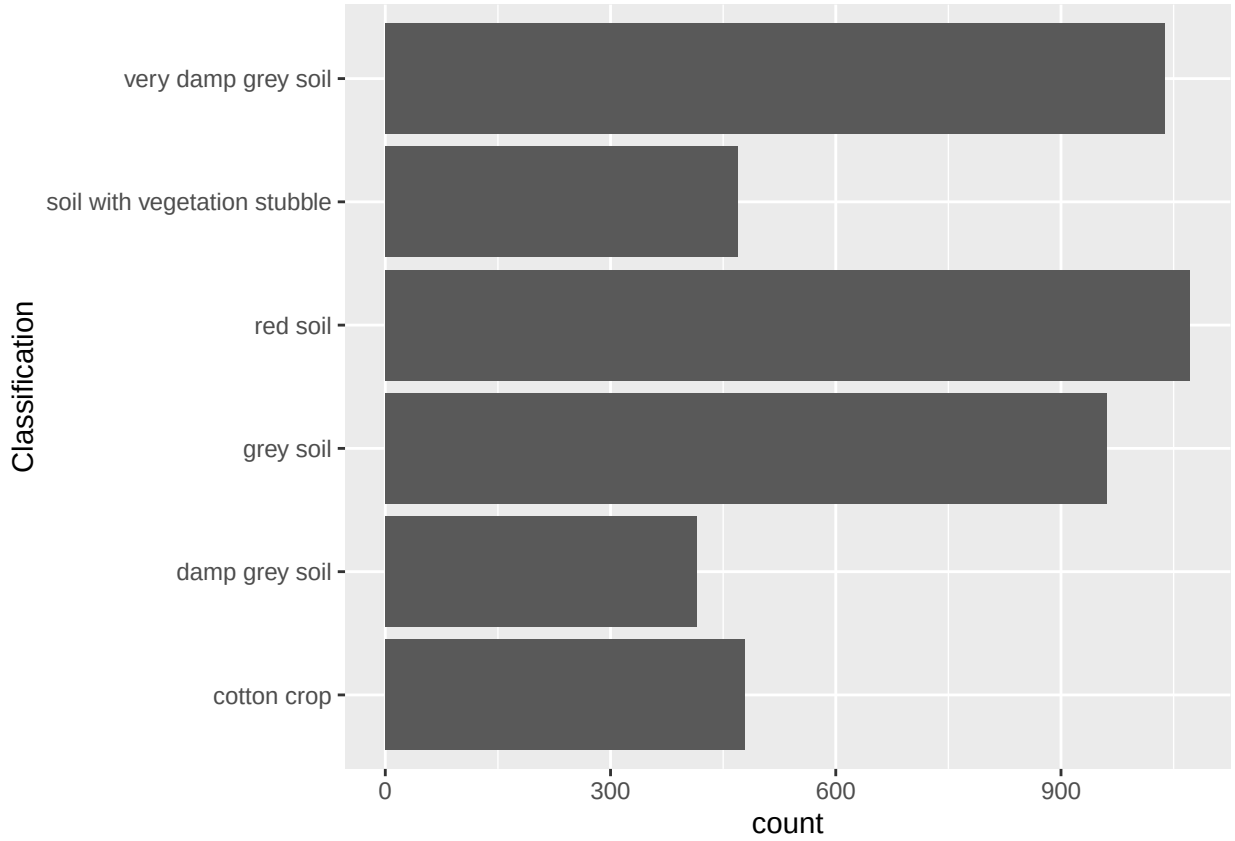
```
summary(train_set)
```

```
##  Middle_center_spec1 Middle_center_spec2 Middle_center_spec3
##  Min.   : 40.00      Min.   : 27.00      Min.   : 56.00
##  1st Qu.: 60.00      1st Qu.: 71.00      1st Qu.: 85.00
##  Median : 68.00      Median : 85.00      Median :101.00
##  Mean   : 69.13      Mean   : 83.43      Mean   : 99.24
##  3rd Qu.: 79.00      3rd Qu.:103.00      3rd Qu.:113.00
##  Max.   :104.00      Max.   :130.00      Max.   :139.00
##  Middle_center_spec4 Classification
##  Min.   : 34.00      Length:4435
##  1st Qu.: 69.00      Class :character
##  Median : 81.00      Mode  :character
##  Mean   : 82.62
##  3rd Qu.: 92.00
##  Max.   :157.00
```

Now we look at the frequency distribution of our land classification. We can see that we have no data on "mixture" land. So we only have 6 types of land classifications.

```
# How many times do different classifications appear

count_of_classifications <- train_set %>% ggplot(aes(Classification)) + geom_bar() +
    coord_flip()
count_of_classifications
```

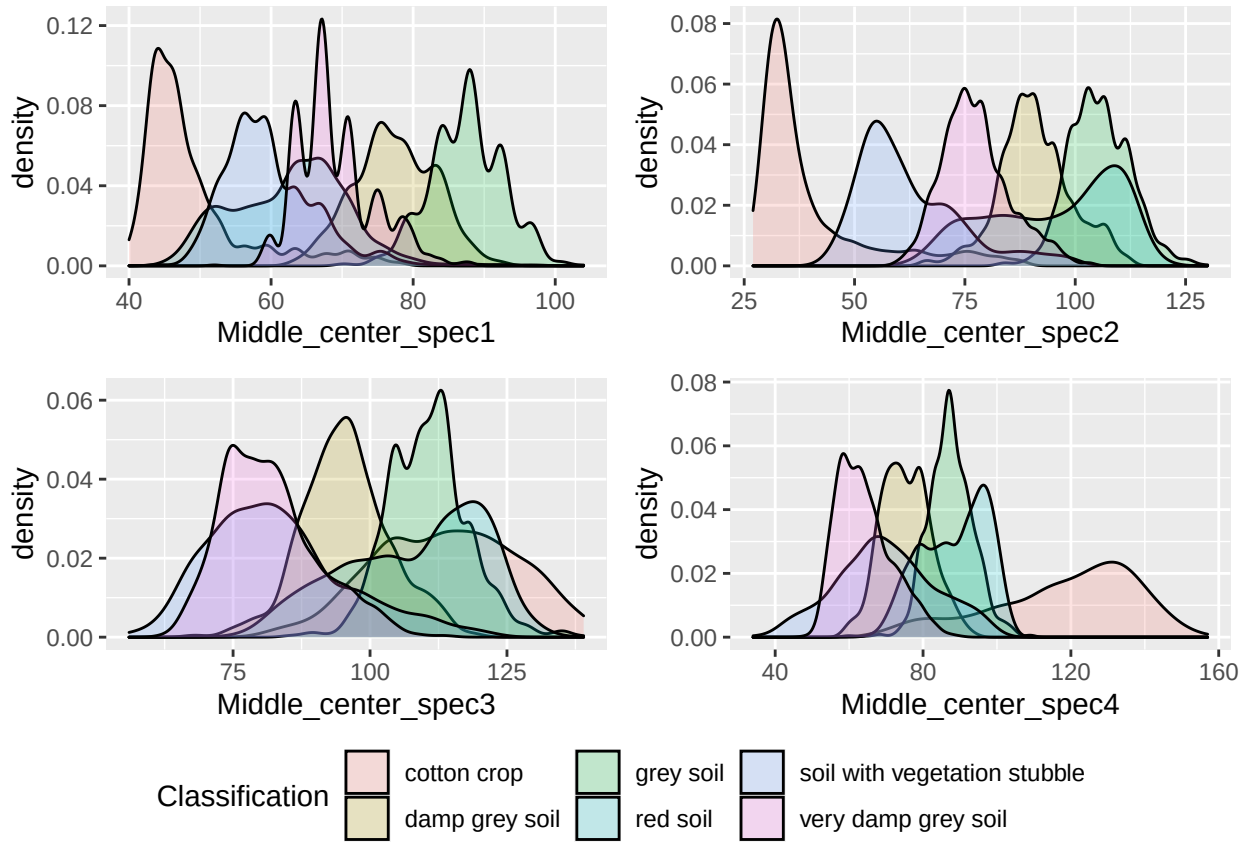


We then visualize different predictors in our data set. We see that for each land classification each variable demonstrates a distinct distribution. These distributions do overlap with each other but they peak at different points. This is a good sign for the predictive power of these variables for land labels. We also see that they seem to be bell shaped which indicates normal distribution.

```
# We make density plots for the four spectral bands for each land classification

dens_spec1 <- train_set %>% ggplot(aes(Middle_center_spec1, fill = Classification)) +
    geom_density(alpha = 0.2)
dens_spec2 <- train_set %>% ggplot(aes(Middle_center_spec2, fill = Classification)) +
    geom_density(alpha = 0.2)
dens_spec3 <- train_set %>% ggplot(aes(Middle_center_spec3, fill = Classification)) +
    geom_density(alpha = 0.2)
dens_spec4 <- train_set %>% ggplot(aes(Middle_center_spec4, fill = Classification)) +
    geom_density(alpha = 0.2)

dens_grid <- ggarrange(dens_spec1, dens_spec2, dens_spec3, dens_spec4, ncol = 2,
    nrow = 2, common.legend = TRUE, legend = "bottom")
dens_grid
```



Following are the distribution statistics estimates of each predictor for different classifications.

```
# Mean and Standard deviation of predictors distribution by classification

train_set %>% group_by(Classification) %>% summarize(mean_spec1 = mean(Middle_center_spec1),
    sd_spec1 = sd(Middle_center_spec1), mean_spec2 = mean(Middle_center_spec2), sd_spec2 = sd(Middle_ce
    mean_spec3 = mean(Middle_center_spec3), sd_spec3 = sd(Middle_center_spec3), mean_spec4 = mean(Middl
    sd_spec4 = sd(Middle_center_spec4))
```

```
## # A tibble: 6 x 9
##   Classification mean_spec1 sd_spec1 mean_spec2 sd_spec2 mean_spec3 sd_spec3
##   <chr>               <dbl>    <dbl>      <dbl>    <dbl>      <dbl>    <dbl>
## 1 cotton crop          48.8     7.57       39.9    13.5       114.     12.6
## 2 damp grey soil       77.4     5.54       90.9     8.16       95.6     7.91
## 3 grey soil            87.5     5.04      105.      6.87      111.      7.23
## 4 red soil             62.8     8.02       95.3    14.5       108.     12.6
## 5 soil with veg~       59.6     6.09       62.3    11.6        83.0    12.6
## 6 very damp gre~       69.0     5.38       77.4     7.69       81.6     8.74
## # ... with 2 more variables: mean_spec4 <dbl>, sd_spec4 <dbl>
```

We confirm normal distribution of data through a qq plot. The plot confirms that our data is normally distributed.

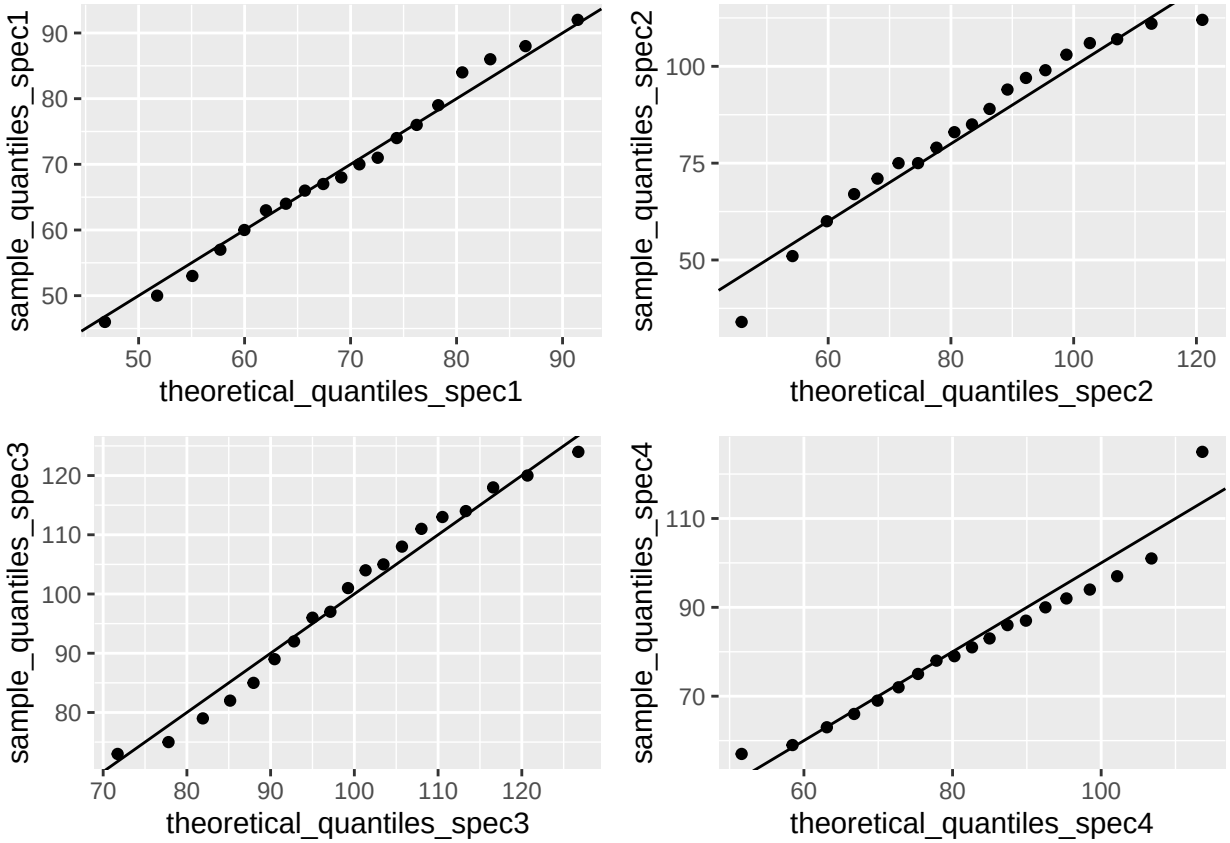
```
# Develop QQ plots
```

```r
p <- seq(0.05, 0.95, 0.05)
sample_quantiles_spec1 <- quantile(train_set$Middle_center_spec1, p)
sample_quantiles_spec2 <- quantile(train_set$Middle_center_spec2, p)
sample_quantiles_spec3 <- quantile(train_set$Middle_center_spec3, p)
sample_quantiles_spec4 <- quantile(train_set$Middle_center_spec4, p)

theoretical_quantiles_spec1 <- qnorm(p, mean = mean(train_set$Middle_center_spec1),
    sd = sd(train_set$Middle_center_spec1))
theoretical_quantiles_spec2 <- qnorm(p, mean = mean(train_set$Middle_center_spec2),
    sd = sd(train_set$Middle_center_spec2))
theoretical_quantiles_spec3 <- qnorm(p, mean = mean(train_set$Middle_center_spec3),
    sd = sd(train_set$Middle_center_spec3))
theoretical_quantiles_spec4 <- qnorm(p, mean = mean(train_set$Middle_center_spec4),
    sd = sd(train_set$Middle_center_spec4))

spec1_qq <- qplot(theoretical_quantiles_spec1, sample_quantiles_spec1) + geom_abline()
spec2_qq <- qplot(theoretical_quantiles_spec2, sample_quantiles_spec2) + geom_abline()
spec3_qq <- qplot(theoretical_quantiles_spec3, sample_quantiles_spec3) + geom_abline()
spec4_qq <- qplot(theoretical_quantiles_spec4, sample_quantiles_spec4) + geom_abline()

qq_grid <- ggarrange(spec1_qq, spec2_qq, spec3_qq, spec4_qq, ncol = 2, nrow = 2,
    common.legend = TRUE, legend = "bottom")
qq_grid
```



Results

We now start using machine learning algorithms to predict land classification labels. We use caret package for this analysis since it provides very useful algorithms for training and testing of models.

As we noted earlier, our land classification label variable is a character. We know that caret package functions require the dependent variable to be of class "factor". We, therefore, transform the variable from a character to a factor.

```
# Convert our classification variable from 'character' to 'factor' to use caret
# package functions
class(train_set$Classification)
```

```
## [1] "character"
```

```
train_set$Classification <- as.factor(train_set$Classification)
class(train_set$Classification)
```

```
## [1] "factor"
```

```
# We to the same for test set
class(test_set$Classification)
test_set$Classification <- as.factor(test_set$Classification)
class(test_set$Classification)
```

Baseline

We start developing from setting a baseline of performance. We do this by randomly assigning a classification to each row without using any of the predictors. We evaluate the accuracy of this model and use this as a baseline. For any model to be useful to us, it needs to perform better than this baseline.

```
# Lets start with simple random classification to set a baseline. Note: we have
# not used 'mixture' classification because we know it is not a part of the
# dataset
set.seed(1)
y_hat <- sample(c("red soil", "cotton crop", "grey soil", "damp grey soil", "soil with vegetation stubbl
    "very damp grey soil"), length(nrow(test_set)), replace = TRUE) %>% factor(levels = levels(test_set$

baseline_acc <- mean(y_hat == test_set$Classification)
baseline_acc
```

```
## [1] 0.2305
```

As expected, random predictions perform quite poorly and we hardly make the correct predictions a quarter of the times.

As discussed earlier, the data set provided is a very small sample of a bigger satellite image. Because of the small size of data we will not be using cross-validation. Cross-validation approach uses randomly generated smaller samples that are not used for training, and instead used to estimate the true error. This approach is used to prevent over-training on training dataset. When we overtrain a model we underestimate the error in our predictions and the model performs poorly on new data. We know that caret package by default uses

cross-validation in its algorithms. We, therefore, stop it from doing so through trainControl function. If we had a bigger dataset it would have been better to use cross-validation approach but given the size of our dataset we will use method "none" which only fits one model to the entire training set.

```
# Set trainControl to 'none' so the models apply to entire dataset instead of
# cross-validating using smaller datasets
control <- trainControl(method = "none")
```

Machine Learning Model using different algorithms

We start developing our model and try the following algorithms to evaluate their performance: LDA, QDA, KNN, RF, svmLinear, svmRadial, and multinom. Since we have a small sample and we are not using cross-validation, we should be careful not to over-train our model on the training data. We do this by comparing the accuracy of our model on training and test set. If the accuracy on the training set is significantly higher than the accuracy on test set it means the model is over-trained on the training set and cannot be generalized to any other data.

Generative models are a wide class of machine learning algorithms which make predictions by modelling joint distribution P(y, x), often by assuming some distribution for the conditional distribution of P (x|y).

Discriminative models are a class of supervised machine learning models which explicitly model the actual distribution of each class by estimating conditional probabilities P(y|x) which are used make predictions.

- **Linear discriminant analysis (LDA)/ Quadratic Discriminant Analysis (QDA)**: Both these models and generative. We anticipate that QDA will perform better than LDA since we have six classification labels to be predicted using four predictors. This would require more flexibility in classification than a linear model can provide. We test both models to see which fits better. Our results confirm that QDA model performs better than LDA confirming linear approximation of the decision boundary separating different classifications is not sufficient to make our predictions.
- **K nearest neighbours (KNN)**: KNN is also a discriminant model which predicts classifications based on classification of nearest neighbours. We find that the accuracy of KNN is slighly poorer than QDA. This is not surprising given the small dataset. When we have a small dataset, the fact the QDA makes an assumption about the structure of the decision boundary (quadratic) helps make better predictions than KNN that makes no assumptions about form. For KNN to perform better we need a bigger dataset.

- **Random forest**: We test random forest model which is a popular discriminant algorithm to improve prediction performance and reduce instability by averaging multiple decision trees (a forest of trees constructed with randomness). Each decision tree used a random sample of observations and a random number of features. Because we have just 4 features and a small dataset using random forests may be tricky. We observe a significant difference between training set and test set accuracy, however, our test set accuracy is similar to what we have achieved with other models. As datasets get bigger the acceptable gap between train and test set accuracy gets smaller. Since in this case the data set is small and the test set accuracy is similar to other models we accept this difference and do not consider our model over-trained.
- **svmLinear/svmRadial**: Support Vector Machine ML technique uses generative algorithm for classification problems. SVM works by identifying the optimal decision boundary that separates data points from different groups, and then applies this separation boundary to predict group in new data. svmLinear is used when the boundary between classes is linear while radial is used on non-linear cases. We try both to see which performs better and as expected the non-linear model gives better results.
- **multinom**: Multinomial logistic regression is used to model categorical outcome variables like the land classification labels, in which the log odds of the outcomes are modeled as a linear combination of the predictor variables.

Below we summarize the performance of different algorithms. We see QDA and svmRadial perform the best which is in line with the non-linear nature of classification problem.

```
Accuracy_results <- tibble(method = c("LDA", "QDA", "KNN", "Random Forest", "svmLinear",
    "svmRadial", "multinom"), Test_Accuracy = c(lda_acc_test, qda_acc_test, knn_acc_test,
    rf_acc_test, svmLinear_acc_test, svmRadial_acc_test, multinom_acc_test), Train_Accuracy = c(lda_acc
    qda_acc_train, knn_acc_train, rf_acc_train, svmLinear_acc_train, svmRadial_acc_train,
    multinom_acc_train))
Accuracy_results %>% mutate(difference = Train_Accuracy - Test_Accuracy)
```

```
## # A tibble: 7 x 4
##   method        Test_Accuracy Train_Accuracy difference
##   <chr>                 <dbl>          <dbl>      <dbl>
## 1 LDA                   0.807          0.828    0.0207
## 2 QDA                   0.844          0.850    0.00606
## 3 KNN                   0.843          0.886    0.0427
## 4 Random Forest         0.833          0.959    0.126
## 5 svmLinear             0.833          0.850    0.0173
## 6 svmRadial             0.846          0.863    0.0166
## 7 multinom              0.826          0.843    0.0176
```

Ensemble

We observe that the accuracy of all models is within 5 percentage points. We develop an ensemble to see if we can improve our predictions using a combination of various models. We remove the LDA/svmLinear and retain QDA/svmRadial since our previous results have already shown that their non-linear classification algorithms perform better. To make ensemble prediction we use the approach of voting. Any classification for which more than 50% of the models have made the same prediction will be given that label. In cases where all labels get less than 50% vote a "poor prediction" label is given.

```
# We then try the approach of ensembles to see if it can improve our predictions.
models <- c("qda", "knn", "rf", "multinom", "svmRadial")
fits <- lapply(models, function(model) {
    print(model)
    train(Classification ~ ., method = model, trControl = control, data = train_set)
})
```

```
## [1] "qda"
## [1] "knn"
## [1] "rf"
## [1] "multinom"
## # weights:  36 (25 variable)
## initial  value 7946.453246
## iter  10 value 5058.897621
## iter  20 value 4465.567192
## iter  30 value 2040.638134
## iter  40 value 1964.518408
## iter  50 value 1891.616486
## iter  60 value 1880.556499
## iter  70 value 1879.566720
## iter  80 value 1877.549898
## final  value 1877.549564
```

```
## converged
## [1] "svmRadial"
```

```
# We assign each fit with the name of the model used to train
names(fits) <- models

# We then get predictions for each model in the form of a data frame
pred <- sapply(fits, function(object) predict(object, newdata = test_set))
dim(pred)
```

```
## [1] 2000    5
```

```
head(pred)
```

```
##      qda              knn                   rf
## [1,] "red soil"       "red soil"            "red soil"
## [2,] "grey soil"      "grey soil"           "grey soil"
## [3,] "damp grey soil" "grey soil"           "grey soil"
## [4,] "damp grey soil" "grey soil"           "grey soil"
## [5,] "damp grey soil" "very damp grey soil" "very damp grey soil"
## [6,] "damp grey soil" "damp grey soil"      "damp grey soil"
##      multinom         svmRadial
## [1,] "red soil"       "red soil"
## [2,] "grey soil"      "grey soil"
## [3,] "damp grey soil" "grey soil"
## [4,] "damp grey soil" "grey soil"
## [5,] "damp grey soil" "damp grey soil"
## [6,] "damp grey soil" "damp grey soil"
```

```
# We confirm the dimensions of our results. We should have one column for each
# model and one row for each row in test set.
length(models)
```

```
## [1] 5
```

```
length(test_set$Classification)
```

```
## [1] 2000
```

```
# Now we compute accuracy for each model
acc <- colMeans(pred == test_set$Classification)
acc
```

```
##       qda       knn        rf  multinom svmRadial
##    0.8440    0.8415    0.8440    0.8255    0.8470
```

```
# Now we compute average accuracy across models
mean(acc)
```

```
## [1] 0.8404
```

```r
# We use voting approach (more than 50% votes) to pick ensemble prediction.
# First we find the proportion of models that have predicted each classification
# in each row.
votes_cotton_crop <- rowMeans(pred == "cotton crop")
votes_damp_grey_soil <- rowMeans(pred == "damp grey soil")
votes_grey_soil <- rowMeans(pred == "grey soil")
votes_red_soil <- rowMeans(pred == "red soil")
votes_soil_with_vegetation_stubble <- rowMeans(pred == "soil with vegetation stubble")
votes_very_damp_grey_soil <- rowMeans(pred == "very damp grey soil")

# Based on these proportions we made the ensemble's prediction using the
# classification given by more than 50% of the models
y_hat <- ifelse(votes_cotton_crop > 0.5, "cotton crop", ifelse(votes_damp_grey_soil >
    0.5, "damp grey soil", ifelse(votes_grey_soil > 0.5, "grey soil", ifelse(votes_red_soil >
    0.5, "red soil", ifelse(votes_soil_with_vegetation_stubble > 0.5, "soil with vegetation stubble",
    ifelse(votes_very_damp_grey_soil > 0.5, "very damp grey soil", "poor prediction"))))))

# We then test its accuracy which is higher than individual accuracies of all
# models
en_acc_test <- mean(y_hat == test_set$Classification)
```

The final ensemble model accuracy

We find that our ensemble performs better than all individual models with an accuracy of over 85%. However, in case of 11 classifications it does not make a prediction due to less than 50% votes for each label. Given the high overall accuracy, the number of failures of classification is quite small.

```r
en_acc_test
```

```
## [1] 0.8535
```

```r
# How many predictions have less than 50% of models predicting one classificatiom
length(which(y_hat == "poor prediction"))
```

```
## [1] 11
```

```r
pred[which(y_hat == "poor prediction"), ]
```

```
##       qda                           knn
##  [1,] "damp grey soil"              "grey soil"
##  [2,] "soil with vegetation stubble" "cotton crop"
##  [3,] "damp grey soil"              "very damp grey soil"
##  [4,] "soil with vegetation stubble" "damp grey soil"
##  [5,] "soil with vegetation stubble" "cotton crop"
##  [6,] "grey soil"                   "very damp grey soil"
##  [7,] "very damp grey soil"         "damp grey soil"
##  [8,] "damp grey soil"              "very damp grey soil"
##  [9,] "soil with vegetation stubble" "soil with vegetation stubble"
## [10,] "soil with vegetation stubble" "soil with vegetation stubble"
## [11,] "soil with vegetation stubble" "very damp grey soil"
```

```
##       rf                    multinom
##  [1,] "very damp grey soil" "damp grey soil"
##  [2,] "cotton crop"         "soil with vegetation stubble"
##  [3,] "grey soil"           "very damp grey soil"
##  [4,] "cotton crop"         "very damp grey soil"
##  [5,] "cotton crop"         "soil with vegetation stubble"
##  [6,] "very damp grey soil" "damp grey soil"
##  [7,] "grey soil"           "very damp grey soil"
##  [8,] "very damp grey soil" "damp grey soil"
##  [9,] "grey soil"           "cotton crop"
## [10,] "grey soil"           "cotton crop"
## [11,] "very damp grey soil" "soil with vegetation stubble"
##       svmRadial
##  [1,] "grey soil"
##  [2,] "very damp grey soil"
##  [3,] "damp grey soil"
##  [4,] "very damp grey soil"
##  [5,] "very damp grey soil"
##  [6,] "grey soil"
##  [7,] "damp grey soil"
##  [8,] "grey soil"
##  [9,] "red soil"
## [10,] "grey soil"
## [11,] "damp grey soil"
```

Conclusion

An ensemble of different models that support non-linear classification problems provides us with the most accurate predictions of land classification. In order to build this area of machine learning bigger dataset with more detailed site validation data is required. This includes ground data on types of crops, low-income slum housing versus high-income urban housing, irrigation channels and water courses etc. Machine learning for land classification can greatly improve the quality and reduce the cost of data available to policy makers.