From “Classification of Gem Materials Using Machine Learning”

Matthew F. Hardman, Artitaya Homkrajae, Sally Eaton-Magaña, Christopher M. Breeding, Aaron C. Palke, and Ziyin Sun

Gems & Gemology, Vol. 60, No. 3, pp. 306–329

APPENDIX 1

All statistical analysis in this study is conducted using R version 4.3.2.

Dataset Preparation

We define several variables/terms for use with all methods. These names are examples only and can be replaced by other names, if the user prefers.

> dataset

Datasets of pearls, alexandrite, or CVD-grown diamond spectroscopic data can be saved as .csv files and named appropriately, and then imported into R. In this supplementary file, these files are referred to as “dataset” in all following scripts. In these .csv files, individual samples are given as rows in the data table. Columns correspond to metadata (such as sample ID, treatment, locality, etc.) as well as compositional variables or peak position data.

> variable_list

This is the list of names for the variables used for classification purposes. The “variable_list” is itself a variable that can be referenced in statistical scripts. For pearls and alexandrite, the variables to be included in “variable_list” correspond to the concentrations of various elements (Mg, Fe, etc.). For CVD-grown diamonds, “variable_list” includes variables that correspond to the baselined peak intensity for various peak positions in photoluminescence emission spectra. The variable list can be defined using the script

variable_list <- c(“Variable 1”,”Variable 2”, “Variable 3”, …)

with “Variable 1,” “Variable 2,” “Variable 3” corresponding to the names of the columns in the dataset, referencing the variables that are to be used in the statistical analysis (e.g., “Mg,” “Fe,” etc. for saltwater pearls, and peak position variables for CVD-grown diamonds).

> outcome_variable

This is the generalized name of the variable that is the primary outcome/classification variable for the chosen dataset. In this study, for pearls and alexandrite this corresponds to "Locality," and for CVD-grown diamonds this corresponds to "Treatment." All methods in this study are described in general terms, and the scripts can be modified to suit any particular dataset.

Spectrum Baselining

Install and load the "baseline" package in R.

```
> install.packages("baseline")
> library(baseline)
```

The user must specify the spectrum to be baselined, and format the data for the spectrum such that the .csv file is formatted into two columns, with the first being the wavelength position of the analysis and the second being the counts at that position. This file is henceforth referred to as "spectrum." This spectrum file then corresponds to the wavelength position and intensity values for a photoluminescence emission spectrum, and can be baselined using the following script.

```
> spectrum_counts <- t(spectrum[,2])
> baselined.spectrum <- baseline(spectrum_counts, lambda = 6, hwi = 50, it = 10, int = 2000, method = 'fillpeaks')
```

The arguments *hwi* (half-width of the local windows for the baselining), *it* (number of iterations), and *int* (number of divisions to divide spectra into) can be adjusted to suit the available spectra. The "fillpeaks" method is one of many approaches to spectrum baselining. Details and descriptions of these parameters, as well as alternative baselining methods, are available at cran.r-project.org/web/packages/baseline/baseline.pdf.

Principal Component Analysis (PCA)

The principal component solution is derived using the prcomp() function from base R. Data are first transformed using the scale() and log() functions in R, using the variables included within variable_list.

```
> pca.solution <- prcomp(scale(log(dataset[,variable_list])))
```

A biplot can be generated using the biplot() function.

```
> biplot(pca.solution)
```

Linear Discriminant Analysis (LDA)

LDA in this study is conducted using the "MASS" package. Install and load the MASS library, then generate the LDA solution.

```
> install.packages("MASS")
> library(MASS)
> lda.solution(as.factor(dataset$outcome_variable) ~ ., data = dataset[,variable_list])
```

Random Forest (RF)

Random forest in this study is conducted using the randomForest() function in the "randomForest" package in R. The number of trees (*ntree*) as well as *mtry* (the number of variables to try at each split) can be specified. *nmin* corresponds to the number of samples in the outcome class with the fewer samples. *class_num* corresponds to the number of outcome classes. Provided values for *ntree* and *mtry* in this example are arbitrary.

```
> install.packages("randomForest")
> library(randomForest)
> randomforest.solution <- randomForest(as.factor(dataset[,outcome_variable]) ~ ., data =
  dataset[,variable_list], ntree = 1000, mtry = 3, strata = dataset[,outcome_variable],
  sampsize = rep(nmin, class_num))
```

Support Vector Machines (SVM)

SVM in this study is conducted using the "e1071" package. Install and load the e1071 library, then generate the SVM solution.

```
> install.packages("e1071")
> library(e1071)
> svm.solution <- svm(as.factor(dataset$outcome_variable) ~ ., data = dataset[,variable_list], method = "C-classification", kernel = "radial", gamma = 0.1, cost = 10)  # method, kernel, gamma, and cost are parameters that can be specified by the user
```

Artificial Neural Networks (ANN)

ANN models in this study are produced using the "nnet" package. Install and load the nnet library, then generate the ANN solution.

```
> install.packages("nnet")
> library(nnet)
> ann.solution <- nnet(as.factor(dataset$outcome_variable) ~ ., data = dataset[,variable_list], size = 5, maxit = 1000)# Size indicates the number of hidden neurons. Maxit specifies the number of iterations for model training.
```

Feature Selection

Feature selection in this study is conducted using the Boruta() function, which is available in the "Boruta" package in R. Install and load the Boruta library.

```
> install.packages("Boruta")
> library(Boruta)
```

Conduct the feature selection procedure

```
> boruta.solution <- Boruta(as.factor(dataset$outcome_variable) ~ ., data =
  dataset[,variable_list])
```

View the list of important, tentative, and not important variables

```
> print(boruta.solution)
```

Model Validation

In this study, all models are validated using *k*-fold cross validation with $k = 10$ folds. The dataset is subdivided into ten equally sized subsets ("folds"), each produced using stratified subsampling (the proportion of each group/outcome is the same in the full database as well as in each fold). These processes can be conducted using the caret package in R.

Install and load the "caret" package:

```
>	install.packages("caret")
>	library(caret)
```

Subset the starting dataset into ten folds:

```
>	fold <- createFolds(factor(dataset$outcome_variable), k = 10, list = FALSE)
>	dataset$fold <- fold
>	training_subset <- subset(dataset, fold == 1) # substitute 1 for 2, 3, 4, … 10
>	testing_subset <- subset(dataset, fold == 1) # substitute 1 for 2, 3, 4, … 10
```

For all *k* values (in this example, from 1 to 10), the training_subset and testing_subset are generated. A statistical model is produced using the former and tested with the latter. The error rates for each of the ten procedures are averaged, and the model error rate reported as mean $\pm 2\sigma$ for the ten folds.