DATA624 - Homework 2

Esteban Aramayo

2022-06-10

# Week 2: Data Pre-processing & Exponential Smoothing [ Jun 6 - Jun 12]

## Data Pre-processing (Applied Predictive Modeling KJ)

### Exercise 3.1

The UC Irvine Machine Learning Repository6 contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. There are nine predictors, including the refractive index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe.

The data can be accessed via:

library(mlbench)  
data(Glass)

str(Glass)

#> 'data.frame': 214 obs. of 10 variables:  
#> $ RI : num 1.52 1.52 1.52 1.52 1.52 ...  
#> $ Na : num 13.6 13.9 13.5 13.2 13.3 ...  
#> $ Mg : num 4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...  
#> $ Al : num 1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...  
#> $ Si : num 71.8 72.7 73 72.6 73.1 ...  
#> $ K : num 0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...  
#> $ Ca : num 8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...  
#> $ Ba : num 0 0 0 0 0 0 0 0 0 0 ...  
#> $ Fe : num 0 0 0 0 0 0.26 0 0 0 0.11 ...  
#> $ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...

#### Exercise 3.1.a

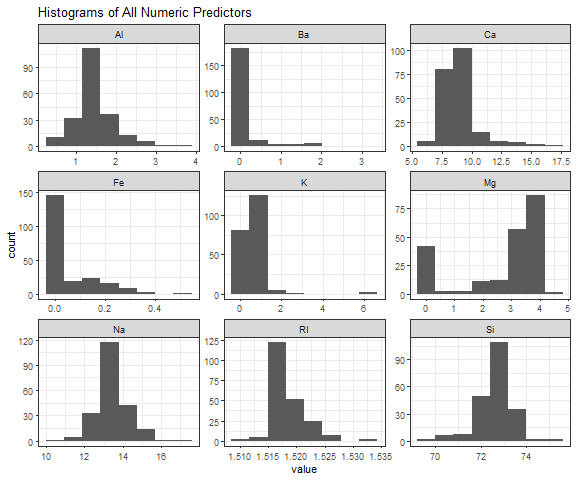
Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.

#### Solution

To understand the distribution of each predictor, we generate 9 histograms for the numeric predictors and one bar plot for the non-numeric factor. The table below shows the description of the distributions observed from their corresponding histograms.

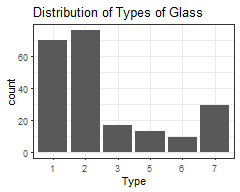
| Predictor | Distribution Description |
| --- | --- |
| Al | slightly right skewed |
| Ba | right skewed and mostly centered around 0 |
| Ca | right skewed |
| Fe | right skewed and mostly centered around 0.0 |
| K | right skewed |
| Mg | left skewed and bi-modal |
| Na | nearly normal with a slight right tail |
| RI | slightly right skewed |
| Si | nearly normal with a slight left tail |
| Type | right skewed and mostly centered around 1,2, and 7 |

library(dplyr)  
library(purrr)  
library(tidyr)  
library(ggplot2)  
library(corrplot)  
library(e1071)  
  
Glass %>%  
 purrr::keep(is.numeric) %>%  
 tidyr::gather() %>%  
 ggplot2::ggplot(aes(value)) +  
 geom\_histogram(bins = 8) +   
 facet\_wrap(~key, scales = 'free') +  
 ggtitle("Histograms of All Numeric Predictors") +  
 theme\_bw(base\_size=8)



Bar plot for “Type” non-numeric factor predictor.

Glass %>%  
 ggplot2::ggplot() +  
 geom\_bar(aes(x = Type)) +  
 ggtitle("Distribution of Types of Glass") +  
 theme\_bw(base\_size=8)



To determine the relation between the numeric predictors we generate a correlation matrix plot. Which shows us the following:

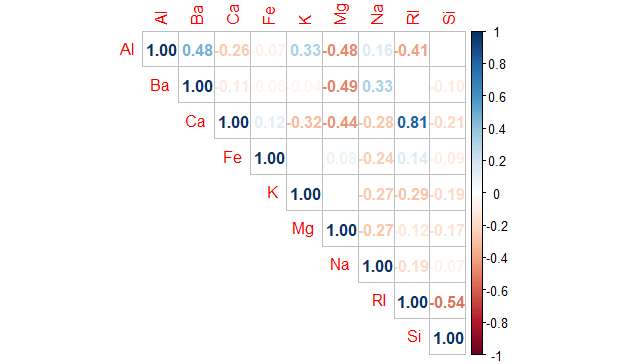
* There is positive correlation among the predictors below. The strongest positive correlation is between and .

| Predictor | Predictor | Correlation Value |
| --- | --- | --- |
|  |  | 0.81 |
|  |  | 0.48 |
|  |  | 0.33 |
|  |  | 0.33 |

* There is negative correlation among the predictors below. The strongest negative correlation is between and .

| Predictor | Predictor | Correlation Value |
| --- | --- | --- |
|  |  | -0.54 |
|  |  | -0.49 |
|  |  | -0.48 |
|  |  | -0.44 |
|  |  | -0.41 |

Glass %>%  
 purrr::keep(is.numeric) %>%  
 cor() %>%  
 corrplot(method = 'number', order = 'alphabet', type = 'upper')

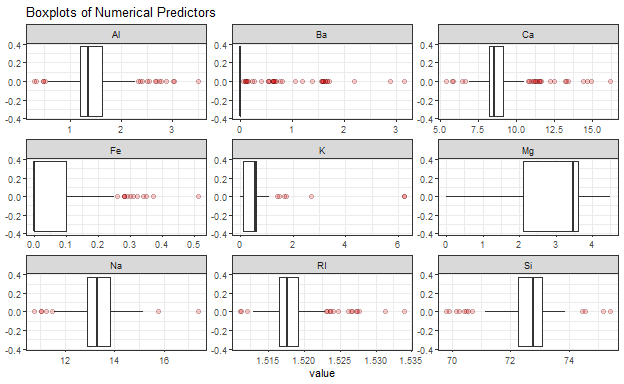


#### Exercise 3.1.b

Do there appear to be any outliers in the data? Are any predictors skewed?

To determine the presence of outliers we generate boxplots for each of the numeric predictors. The boxplots reveal that with exception of all other predictors have potential outliers.

Glass %>%  
 purrr::keep(is.numeric) %>%  
 tidyr::gather() %>%  
 ggplot2::ggplot(aes(value)) +   
 geom\_boxplot(outlier.color = "black",  
 outlier.shape = 21,  
 outlier.fill = "red",  
 outlier.alpha = 0.2) +  
 facet\_wrap(~key, scales = 'free') +  
 ggtitle("Boxplots of Numerical Predictors") +  
 theme\_bw(base\_size=8)



Based on the previous histograms from Exercise 3.1.a, the boxplots above, and the skewness values below, we can confirm that all predictors have some degree of skewness. The predictors and are left skewed, while the rest of the predictors are right skewed. Overall, predictor shows the highest degree of skewness (right skewness).

Glass %>%  
 purrr::keep(is.numeric) %>%  
 apply(., 2, e1071::skewness) %>%  
 round(4)

#> RI Na Mg Al Si K Ca Ba Fe   
#> 1.6027 0.4478 -1.1365 0.8946 -0.7202 6.4601 2.0184 3.3687 1.7298

#### Exercise 3.1.c

Are there any relevant transformations of one or more predictors that might improve the classification model?

Depending on the type of classification model that one decides to use, there might be two problems in the data that need to be dealt with, skewness and outliers.

**Handling skewness**

From Exercises 3.1.a and 3.1.c we know that all predictors have some degree of skewness. So, to it might be a good idea to try to reduce the skew by applying transformations to the data such as log, square root, or inverse.

For those skewed predictors that contain values greater than zero, we could apply the Box and Cox procedure to each predictor to determine the appropriate transformation parameter . Which in turn, would help us apply the right transformation to the data to reduce the skew.

**Handling outliers**

Exercise 3.1.b revealed that with exception of all other predictors have potential outliers.

To handle the outliers in our predictors, we would first need to analyze all the suspected outliers to determine if they are true outliers or not. If the values are proved to be errors, then probably it would be better to remove them. Ot it might be possible that they might be a clustered group of observations that make sense or some other reason that justifies their existence. Second, if after the analysis we conclude that there are legitimate outliers, then one can apply the “spatial sign” transformation to the predictors as a group. This is to try to reduce an exceptional influence of the outliers on the model.

### Exercise 3.2

The soybean data can also be found at the UC Irvine Machine Learning Repository. Data were collected to predict disease in 683 soybeans. The 35 predictors are mostly categorical and include information on the environmental conditions (e.g., temperature, precipitation) and plant conditions (e.g., left spots, mold growth). The outcome labels consist of 19 distinct classes.

The data can be loaded via:

library(mlbench)  
data(Soybean)  
  
## See ?Soybean for details

#### Exercise 3.2.a

Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?

From the chapter, predictors with degenerate distributions are the ones that meet one of the following criteria:

* If a predictor variable has a single unique value; we refer to this type of data as a zero variance predictor.
* Similarly, some predictors might have only a handful of unique values that occur with very low frequencies. These “near-zero variance predictors” may have a single value for the vast majority of the samples.

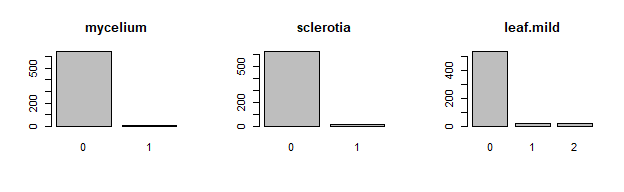
Below is a list of predictors with either zero variance or near-zero variance detected in the Soybean data set. In it one can see that predictors , , and appear to have degenerate distributions.

library(caret)  
  
pred <- Soybean[, 2:36]  
  
nzvars <- caret::nearZeroVar(x = pred, names = TRUE, saveMetrics = TRUE)  
  
degen <- nzvars %>%  
 filter(zeroVar == TRUE | nzv == TRUE) %>%  
 arrange(percentUnique)  
  
print(degen)

#> freqRatio percentUnique zeroVar nzv  
#> mycelium 106.50 0.2928258 FALSE TRUE  
#> sclerotia 31.25 0.2928258 FALSE TRUE  
#> leaf.mild 26.75 0.4392387 FALSE TRUE

Below are the distributions of the predictors , , and , which appear to have degenerate distributions.

par(mfrow = c(1,3))  
plot(Soybean$mycelium, main='mycelium')  
plot(Soybean$sclerotia, main='sclerotia')  
plot(Soybean$leaf.mild, main='leaf.mild' )

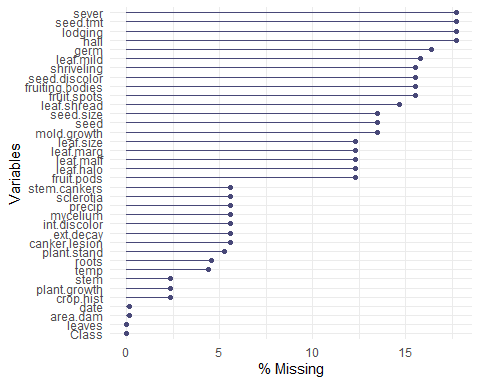


#### Exercise 3.2.b

Roughly 18% of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?

The plot below shows the percentage of missing values for each predictor. Every predictor is missing values. Some more than others.

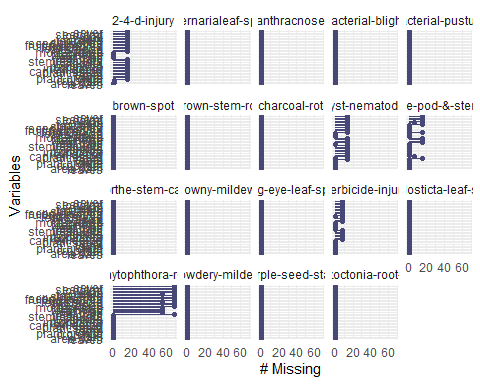
library(naniar)  
  
Soybean %>%  
 gg\_miss\_var(show\_pct = TRUE)



Next, the plot below shows the list of missing values per predictor but grouped by class. Such plot shows that only the 5 classes below have missing values.

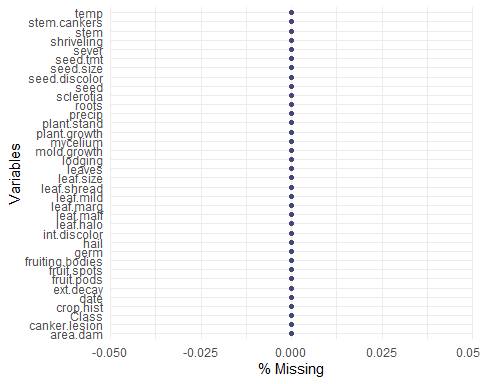
* phytophthora-rot
* diaporthe-pod-&-stem-blight
* cyst-nematode
* 2-4-d-injury
* herbicide-injury

gg\_miss\_var(Soybean, facet = Class)



Below is a plot of missing values for the predictors of the Soybean dataset, but without including the observations for the 5 classes mentioned above. Such plot shows that after excluding the 5 classes, none of the predictors are missing values. This would confirm that the pattern of missing values is related to the 5 classes.

Soybean %>%  
 filter(!Class %in% c("phytophthora-rot", "diaporthe-pod-&-stem-blight", "cyst-nematode",  
 "2-4-d-injury", "herbicide-injury")) %>%  
 gg\_miss\_var(show\_pct = TRUE)



#### Exercise 3.2.c

Develop a strategy for handling missing data, either by eliminating predictors or imputation.

## Data Pre-processing (Applied Predictive Modeling KJ)

## Chapter 7 Exponential smoothing (Forecasting: Principle and Practice HA)

### Exercise 7.8.1 (aka 7.1)

Consider the pigs series — the number of pigs slaughtered in Victoria each month.

#### Exercise 7.8.1.a

Use the () function in R to find the optimal values of and , and generate forecasts for the next four months.

#### Solution

Below is the forecast for the next four months using simple exponential smoothing (ses) forecasts applied to pigs time series.

library(fpp2)  
library(forecast)  
  
fc <- ses(y = pigs, h = 4)  
  
fc

#> Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
#> Sep 1995 98816.41 85605.43 112027.4 78611.97 119020.8  
#> Oct 1995 98816.41 85034.52 112598.3 77738.83 119894.0  
#> Nov 1995 98816.41 84486.34 113146.5 76900.46 120732.4  
#> Dec 1995 98816.41 83958.37 113674.4 76092.99 121539.8

Extract the optimal values of and from the model component of the previously calculated forecast.

optimal\_alpha <- fc$model$par[1]  
optimal\_l0 <- fc$model$par[2]  
  
print(optimal\_alpha)

#> alpha   
#> 0.2971488

print(optimal\_l0)

#> l   
#> 77260.06

The optimal values are: = **0.2971488** and = **77260.06**.

#### Exercise 7.8.1.b

Compute a 95% prediction interval for the first forecast using where is the standard deviation of the residuals. Compare your interval with the interval produced by R.

#### Solution

Below we can see that our first forecast corresponds to September 1995.

print(fc)

#> Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
#> Sep 1995 98816.41 85605.43 112027.4 78611.97 119020.8  
#> Oct 1995 98816.41 85034.52 112598.3 77738.83 119894.0  
#> Nov 1995 98816.41 84486.34 113146.5 76900.46 120732.4  
#> Dec 1995 98816.41 83958.37 113674.4 76092.99 121539.8

Calculate the standard deviation of the residuals

s <- sd( fc$residuals )  
  
print(s)

#> [1] 10273.69

Calculate the Lower and Upper Confidence values for a 95% prediction interval using the residuals of the forecast and using the results from the model.

# residuals results  
Lo95 <- fc$mean[1] - 1.96 \* s  
Hi95 <- fc$mean[1] + 1.96 \* s  
  
# model results  
R.Lo95 <- fc$lower[1,2]   
R.Hi95 <- fc$upper[1,2]

When we compare our calculated values vs the values produced by R we see that the values obtained by both methods are close but not the same. The lower value is higher for the residuals method, while the upper value is smaller for the residuals method.

| Calculation Method | Lower 95 | Upper 95 |
| --- | --- | --- |
| Residuals based () | 78679.97 | 118952.84 |
| R model based | 78611.97 | 119020.84 |

### Exercise 7.8.2 (aka 7.2)

Write your own function to implement simple exponential smoothing. The function should take arguments (the time series), alpha (the smoothing parameter ) and level (the initial level ). It should return the forecast of the next observation in the series. Does it give the same forecast as ()?

### Solution

Let’s create our own version of the () function using the formula for weighted average form

myses <- function(y, alpha, level) {  
   
 # set initial estimated y with level  
 y\_hat <- level  
   
 # traverse elements of series  
 for(i in 1:length(y)) {  
   
 # calculate the next estimated y  
 y\_hat <- alpha \* y[i] + (1 - alpha) \* y\_hat  
   
 }  
   
 return(y\_hat)  
   
}

Now let’s see if the forecast of the next observation in the series returned by our () function matches the value returned by the () function.

fc\_myses <- myses(y = pigs, alpha = optimal\_alpha, level = optimal\_l0)  
  
fc\_ses <- ses(y = pigs, h = 4)  
  
print(fc\_myses)

#> alpha   
#> 98816.41

print(fc\_ses$mean[1])

#> [1] 98816.41

When comparing the forecast results for the next observation we can see that the two calculation methods yield very close results.

| Calculation Method | Forecast value |
| --- | --- |
| myses() | 98816.41 |
| ses() | 98816.41 |

### Exercise 7.8.3 (aka 7.3)

Modify your function from the previous exercise to return the sum of squared errors rather than the forecast of the next observation. Then use the () function to find the optimal values of and . Do you get the same values as the () function?

### Solution

Based on the () function, create a function to return the Sum of Squared Errors (SSE).

mySSE <- function( pars = c(alpha, level), y ) {  
   
 # unpack pars array to get alpha and level values  
 alpha <- pars[1]  
 level <- pars[2]  
   
 # set initial estimated y with level  
 y\_hat <- level  
   
 err <- 0  
   
 SSE <- 0  
  
 # traverse elements of series  
 for(i in 1:length(y)) {  
   
 # calculate error by subtracting estimated y from actual y  
 err <- y[i] - y\_hat  
   
 # sum up and accumulate squared errors  
 SSE <- SSE + err ^ 2  
   
 # calculate the next estimated y  
 y\_hat <- alpha \* y[i] + (1 - alpha) \* y\_hat  
  
 }  
   
 return(SSE)  
   
}

Let’s use our () function to calculate the optimum values and compare them to the values from R’s () function.

result\_mySSE <- optim( par = c(0.5, pigs[1]), y = pigs, fn = mySSE )  
  
  
mySSE\_optimal\_alpha <- result\_mySSE$par[1]  
mySSE\_optimal\_l0 <- result\_mySSE$par[2]  
  
ses\_optimal\_alpha <- fc\_ses$model$par[1]  
ses\_optimal\_l0 <- fc\_ses$model$par[2]

When comparing the results of both methods, we see that the values are very close. The Optimal for the () function is slightly bigger than that of the () function. While the Optimal for the () function is slightly smaller than that of the () function.

| Calculation Method | Optimal | Optimal |
| --- | --- | --- |
| () function | 0.2990081 | 76379.27 |
| R () function | 0.2971488 | 77260.06 |