Preparing\_data\_4ML

bokola

February 16, 2019

Packages

# **Data Preparation for Machine Learning**

**Data preparation** is a vital step in the machine learning pipeline. Just as visualization is necessary to understand the relationships in data, proper preparation or data munging is required to ensure machine learning models work optimally. The process of data preparation is highly interactive and iterative. A typical process includes at least the following steps: 1. **Visualization** of the dataset to understand the relationships and identify possible problems with the data. 2. **Data cleaning and transformation** to address the problems identified. It many cases, step 1 is then repeated to verify that the cleaning and transformation had the desired effect. 3. **Construction and evaluation of a machine learning models**. Visualization of the results will often lead to understanding of further data preparation that is required; going back to step 1.

In this we explore the following: *Recode character strings to eliminate characters that will not be processed correctly.* Find and treat missing values. *Set correct data type of each column.* Transform categorical features to create categories with more cases and coding likely to be useful in predicting the label. *Apply transformations to numeric features and the label to improve the distribution properties.* Locate and treat duplicate cases.

## **Examples**

As an example we begin by looking at the automotive data

## **Treat missing values**

Missing values are a common problem in data set. Failure to deal with missing values before training a machine learning model will lead to biased training at best, and in many cases actual failure. Many R models will not process arrays with missing values. There are two problems that must be dealt with when treating missing values: 1. First you must find the missing values. This can be difficult as there is no standard way missing values are coded. Some common possibilities for missing values are: *Coded by some particular character string, or numeric value like -999.* A NULL value or numeric missing value such as a NaN. 2. You must determine how to treat the missing values: *Remove features with substantial numbers of missing values. In many cases, such features are likely to have little information value.* Remove rows with missing values. If there are only a few rows with missing values it might be easier and more certain to simply remove them. *Impute values. Imputation can be done with simple algorithms such as replacing the missing values with the mean or median value. There are also complex statistical methods such as the expectation maximization (EM) or SMOTE algorithms.* Use nearest neighbor values. Alternatives for nearest neighbor values include, averaging, forward filling or backward filling.

Carefully observe the first few cases from the data frame and notice that missing values are coded with a ‘?’ character. Execute the code in the cell below to identify the columns with missing values.

#(auto\_prices == '?')  
lapply(auto\_prices, function(x) {any(x== '?')})

## $symboling  
## [1] FALSE  
##   
## $normalized.losses  
## [1] TRUE  
##   
## $make  
## [1] FALSE  
##   
## $fuel.type  
## [1] FALSE  
##   
## $aspiration  
## [1] FALSE  
##   
## $num.of.doors  
## [1] TRUE  
##   
## $body.style  
## [1] FALSE  
##   
## $drive.wheels  
## [1] FALSE  
##   
## $engine.location  
## [1] FALSE  
##   
## $wheel.base  
## [1] FALSE  
##   
## $length  
## [1] FALSE  
##   
## $width  
## [1] FALSE  
##   
## $height  
## [1] FALSE  
##   
## $curb.weight  
## [1] FALSE  
##   
## $engine.type  
## [1] FALSE  
##   
## $num.of.cylinders  
## [1] FALSE  
##   
## $engine.size  
## [1] FALSE  
##   
## $fuel.system  
## [1] FALSE  
##   
## $bore  
## [1] TRUE  
##   
## $stroke  
## [1] TRUE  
##   
## $compression.ratio  
## [1] FALSE  
##   
## $horsepower  
## [1] TRUE  
##   
## $peak.rpm  
## [1] TRUE  
##   
## $city.mpg  
## [1] FALSE  
##   
## $highway.mpg  
## [1] FALSE  
##   
## $price  
## [1] TRUE

Execute the code in the cell below to display the data types of each column and a sample of the values.

str(auto\_prices)

## 'data.frame': 205 obs. of 26 variables:  
## $ symboling : int 3 3 1 2 2 2 1 1 1 0 ...  
## $ normalized.losses: chr "?" "?" "?" "164" ...  
## $ make : chr "alfa-romero" "alfa-romero" "alfa-romero" "audi" ...  
## $ fuel.type : chr "gas" "gas" "gas" "gas" ...  
## $ aspiration : chr "std" "std" "std" "std" ...  
## $ num.of.doors : chr "two" "two" "two" "four" ...  
## $ body.style : chr "convertible" "convertible" "hatchback" "sedan" ...  
## $ drive.wheels : chr "rwd" "rwd" "rwd" "fwd" ...  
## $ engine.location : chr "front" "front" "front" "front" ...  
## $ wheel.base : num 88.6 88.6 94.5 99.8 99.4 ...  
## $ length : num 169 169 171 177 177 ...  
## $ width : num 64.1 64.1 65.5 66.2 66.4 66.3 71.4 71.4 71.4 67.9 ...  
## $ height : num 48.8 48.8 52.4 54.3 54.3 53.1 55.7 55.7 55.9 52 ...  
## $ curb.weight : int 2548 2548 2823 2337 2824 2507 2844 2954 3086 3053 ...  
## $ engine.type : chr "dohc" "dohc" "ohcv" "ohc" ...  
## $ num.of.cylinders : chr "four" "four" "six" "four" ...  
## $ engine.size : int 130 130 152 109 136 136 136 136 131 131 ...  
## $ fuel.system : chr "mpfi" "mpfi" "mpfi" "mpfi" ...  
## $ bore : chr "3.47" "3.47" "2.68" "3.19" ...  
## $ stroke : chr "2.68" "2.68" "3.47" "3.40" ...  
## $ compression.ratio: num 9 9 9 10 8 8.5 8.5 8.5 8.3 7 ...  
## $ horsepower : chr "111" "111" "154" "102" ...  
## $ peak.rpm : chr "5000" "5000" "5000" "5500" ...  
## $ city.mpg : int 21 21 19 24 18 19 19 19 17 16 ...  
## $ highway.mpg : int 27 27 26 30 22 25 25 25 20 22 ...  
## $ price : chr "13495" "16500" "16500" "13950" ...

Compare the columns with missing values to their data types. In all cases, the columns with missing values have a character type as a result of using the ‘?’ code. As a result, some columns that should be numeric (bore, stroke, horsepower, peak.rpm, and price) are coded as character.

The next question is how many missing values are in each of these character type columns? Execute the code in the cell below to display the counts of missing values.

for(col in names(auto\_prices)) {  
 if(is.character(auto\_prices[, col])) {  
 count <- sum(ifelse(auto\_prices[, col] == '?', 1, 0))  
 cat(paste(col, as.character(count), '\n'))  
 }  
}

## normalized.losses 41   
## make 0   
## fuel.type 0   
## aspiration 0   
## num.of.doors 2   
## body.style 0   
## drive.wheels 0   
## engine.location 0   
## engine.type 0   
## num.of.cylinders 0   
## fuel.system 0   
## bore 4   
## stroke 4   
## horsepower 2   
## peak.rpm 2   
## price 4

The normalize.losses column has a significant number of missing values and will be removed. Columns that should be numeric, but contain missing values, are processed in the following manner: The ‘?’ values are replaced with R NA values. Rows containing NA values are removed with complete.cases. Execute this code, noticing the resulting shape of the data frame.

# Drop column with too many missing value  
auto\_prices[, 'normalized.losses'] = NULL  
# Remove rows with missing values, accounting for missing values codes as '?'  
cols <- c('price', 'bore', 'stroke', 'horsepower', 'peak.rpm')  
auto\_prices[, cols] <- lapply(auto\_prices[, cols], function(x) {ifelse(x == '?', NA, x)})  
auto\_prices <-auto\_prices[complete.cases(auto\_prices[, cols]), ]  
dim(auto\_prices)

## [1] 195 25

The data set now contains 195 cases and 25 columns. 10 rows have been dropped by removing missing values.

## **Transform column data type**

As has been previously noted, there are five columns in this dataset which do not have the correct type as a result of missing values. This is a common situation, as the methods used to automatically determine data type when loading files can fail when missing values are present.

The code in the cell below iterates over a list of columns setting them to numeric. Execute this code and observe the resulting types.

auto\_prices[, cols] <- lapply(auto\_prices[, cols], as.numeric)  
str(auto\_prices[, cols])

## 'data.frame': 195 obs. of 5 variables:  
## $ price : num 13495 16500 16500 13950 17450 ...  
## $ bore : num 3.47 3.47 2.68 3.19 3.19 3.19 3.19 3.19 3.13 3.5 ...  
## $ stroke : num 2.68 2.68 3.47 3.4 3.4 3.4 3.4 3.4 3.4 2.8 ...  
## $ horsepower: num 111 111 154 102 115 110 110 110 140 101 ...  
## $ peak.rpm : num 5000 5000 5000 5500 5500 5500 5500 5500 5500 5800 ...

## **Feature engineering and transforming variables**

In most cases, machine learning is not performed using raw features. Features are transformed, or combined to form new features in forms which are more predictive This process is known as **feature engineering**. In many cases, good feature engineering is more important than the details of the machine learning model used. It is often the case that good features can make even poor machine learning models work well, whereas, given poor features even the best machine learning model will produce poor results. Some common approaches to feature engineering include: \* **Aggregating categories** of categorical variables to reduce the number. Categorical features or labels with too many unique categories will limit the predictive power of a machine learning model. Aggregating categories can improve this situation, sometime greatly. However, one must be careful. It only makes sense to aggregate categories that are similar in the domain of the problem. Thus, domain expertise must be applied. \* **Transforming numeric variables** to improve their distribution properties to make them more covariate with other variables. This process can be applied not only to features, but to labels for regression problems. Some common transformations include, **logarithmic** and **power** included squares and square roots. \* **Compute new features** from two or more existing features. These new features are often referred to as interaction terms. An interaction occurs when the behavior of say, the produce of the values of two features, is significantly more predictive than the two features by themselves. Consider the probability of purchase for a luxury mens’ shoe. This probability depends on the interaction of the user being a man and the buyer being wealthy. As another example, consider the number of expected riders on a bus route. This value will depend on the interaction between the time of day and if it is a holiday.

## **Aggregating categorical variables**

When a dataset contains categorical variables these need to be investigated to ensure that each category has sufficient samples. It is commonly the case that some categories may have very few samples, or have so many similar categories as to be meaningless. As a specific case, you will examine the number of cylinders in the cars. Execute the code in the cell below to print a frequency table for this variable and examine the result.

auto\_prices %>%  
 select(num.of.cylinders) %>%  
 table()

## .  
## eight five four six three twelve   
## 4 10 155 24 1 1

Notice that there is only one car with three and twelve cylinders. There are only four cars with eight cylinders, and 10 cars with five cylinders. It is likely that all of these categories will not have statistically significant difference in predicting auto price. It is clear that these categories need to be aggregated.

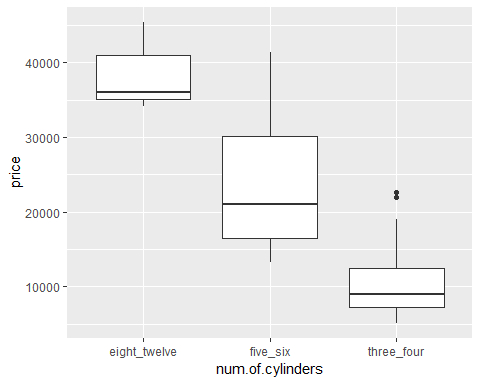
The code in the cell below uses a list with named elements to recode the number of cylinder categories into a smaller number categories. Notice that out vector is defined in advance. Execute this code and examine the resulting frequency table.

cylinder\_categories <- c('three' = 'three\_four', 'four' = 'three\_four',   
 'five' = 'five\_six', 'six' = 'five\_six',  
 'eight' = 'eight\_twelve', 'twelve' = 'eight\_twelve')  
out <- rep('i', length.out = nrow(auto\_prices))  
i = 1  
for (x in auto\_prices[, 'num.of.cylinders']) {  
 out[i] <- cylinder\_categories[[x]]  
 i = i +1  
}  
auto\_prices[, 'num.of.cylinders'] = out  
table(auto\_prices[, 'num.of.cylinders'])

##   
## eight\_twelve five\_six three\_four   
## 5 34 156

There are now three categories. One of these categories only has five members. However, it is likely that these autos will have different pricing from others. Next, execute the code in the cell below to make box plots of the new cylinder categories.

ggplot(auto\_prices, aes(num.of.cylinders, price)) +  
 geom\_boxplot()



Indeed, the price range of these categories is distinctive. It is likely that these new categories will be useful in predicting the price of autos. Now, execute the code in the cell below and examine the frequency table for the body.style feature.

auto\_prices %>%   
 select(body.style) %>%  
 table()

## .  
## convertible hardtop hatchback sedan wagon   
## 6 8 63 94 24

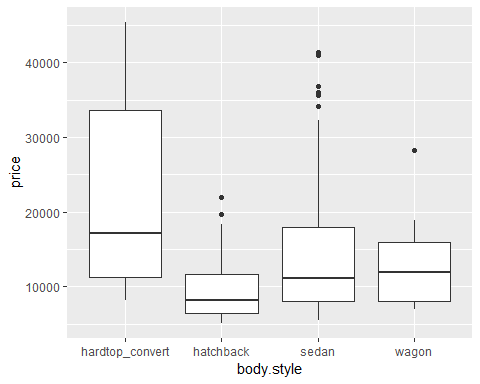
Two of these categories have a limited number of cases. These categories can be aggregated to increase the number of cases using a similar approach as used for the number of cylinders. Execute the code in the cell below to aggregate these categories.

body\_cats = c('sedan' = 'sedan', 'hatchback' = 'hatchback', 'wagon' = 'wagon',   
 'hardtop' = 'hardtop\_convert', 'convertible' = 'hardtop\_convert')  
  
out = rep('i', length.out = nrow(auto\_prices))  
i = 1  
for(x in auto\_prices[,'body.style']){  
 out[i] = body\_cats[[x]]  
 i = i + 1  
}  
auto\_prices[,'body.style'] = out  
  
table(auto\_prices[,'body.style'])

##   
## hardtop\_convert hatchback sedan wagon   
## 14 63 94 24

To investigate if this aggregation of categories was a good idea, execute the code in the cell below to display a box plot. Then, answer **Question 1** on the course page.

ggplot(auto\_prices, aes(body.style, price)) +  
 geom\_boxplot()

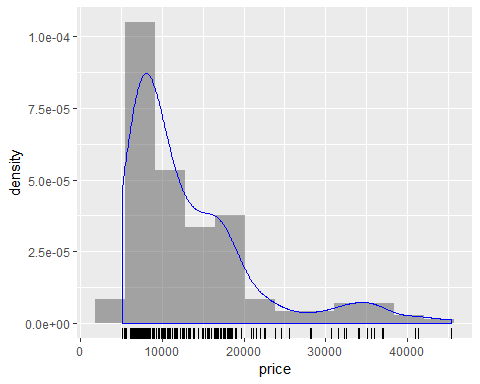
 The hardtop\_convert category appers to have values distinct from other body style

## **Transforming Numeric Variablee**

Transformations are done to improve the performance of ML models by making relationships between variables more linear. In other cases transformations are done to make distributions closeer to normal, or at least more symmetric. These transformations include taking logarithms, exponents and power transformations.

Here we transform the label price

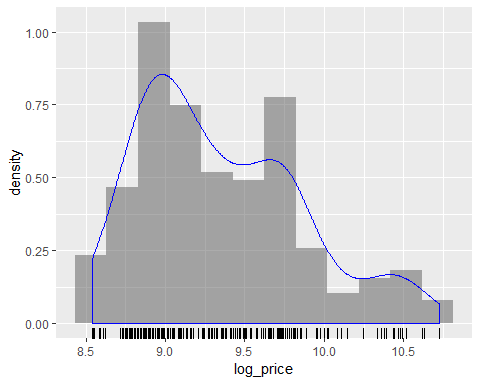
plot\_hist <- function(df, col = 'price', bins = 10){  
 options(repr.plot.width = 4, repr.plot.height = 3.5)  
 bw <- (max(df[, col]) - min(df[, col])) / (bins+1)  
 p <- ggplot(df, aes\_string(col)) +  
 geom\_histogram(binwidth = bw, aes(y = ..density..),alpha = 0.5) +  
 geom\_density(aes(y = ..density..), color = 'blue') +  
 geom\_rug()  
 print(p)  
}  
plot\_hist(auto\_prices)



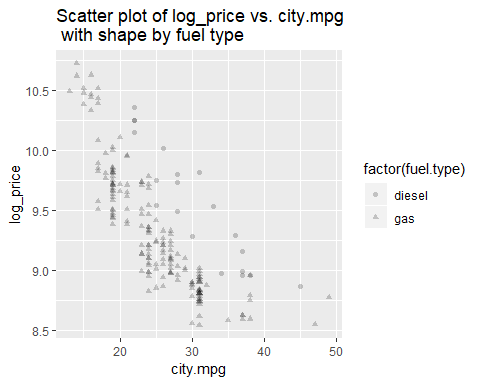
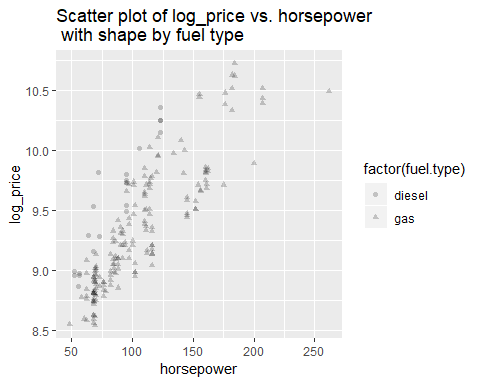
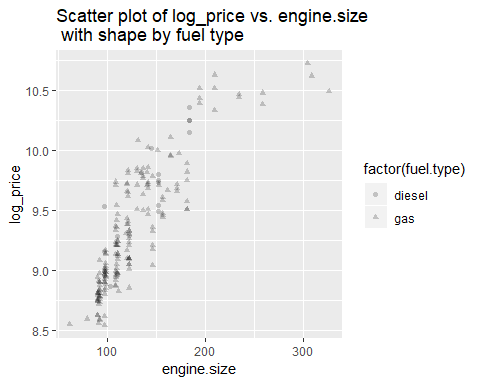
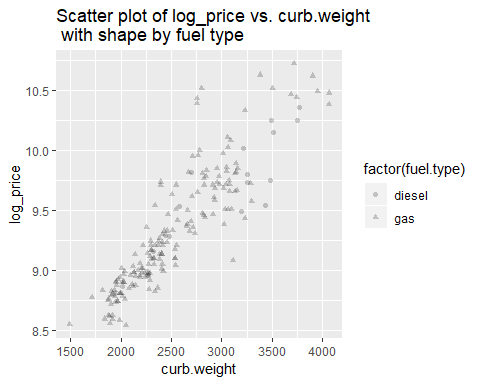
The distribution of auto price is both quite skewwed to the right and multi-modal. Given the skew and the fact that there are no values less than or equal to zero, a log transformation might be appropriate.

The code in the cell below displays a histogram of the logarithm of prices. Execute this code and examine the result.

auto\_prices %<>%  
 mutate(log\_price = log(price))  
plot\_hist(auto\_prices, col = 'log\_price')

 The distribution of the logarithm of price is more symmetric, but still shows some multi-modal tendency and skew. None-the-less, this is an improvement so we will use these values as our label. The next question is, how does this transformation change the relationship between the label and some of the features? To find out, execute the code in the cell below.

plot\_scatter\_sp <- function(df, cols, col\_y = 'log\_price', alpha = 1.0) {  
 options(repr.plot.width = 4, repr.plot.height = 3.5)  
 for(col in cols) {  
 p <- ggplot(df, aes\_string(col, col\_y)) +  
 geom\_point(aes(shape = factor(fuel.type)), alpha = alpha) +  
 ggtitle(paste("Scatter plot of", col\_y, 'vs.', col, '\n with shape by fuel type' ))  
 print(p)  
   
 }  
}  
num\_cols <- c('curb.weight', 'engine.size', 'horsepower', 'city.mpg')  
plot\_scatter\_sp(auto\_prices, num\_cols, alpha = 0.2)

 Comparing the results to those obtained in the visualization lab, it does appear that the relationships between curb.weight and log\_price and city.mpg and log\_price are more linear. The relationship with the log\_price and categorical variables should likely also be investigated. It is also possible that some type of power transformation should be applied to, say horsepower or engine size. In the interest of brevity, these ideas are not pursued here. Before proceeding, answer **Question 2** on the course page.

# Let's save the dataframe to a csv file   
# We will use this in the next module so that we don't have to re-do the steps above  
# You don't have to run this code as the csv file has been saved under the next module's folder  
write.csv(auto\_prices, file = file.path(data.path, 'Auto\_Prices\_Preped.csv'), row.names = FALSE)