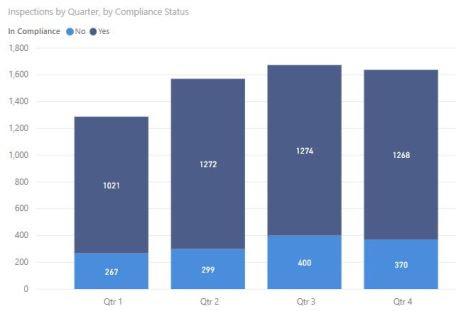
Not all data scientists were computer scientists who discovered their exceptional data literacy skills. They come from all walks of life, and sometimes that can mean optimizing for data structures and performance isn’t the top priority. That’s perfectly fine! There may come a time where you find yourself executing a chunk of code and consciously noting you could go take a short nap, and that’s where you’ve wondered where you could to be more productive. This short example provides help in how to profile using an extremely powerful and user-friendly package, profvis.

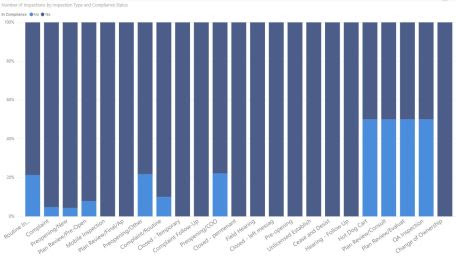
Data for this example: <https://data.detroitmi.gov/Public-Health/Restaurant-Inspections-All-Inspections/kpnp-cx36>

In this tutorial, we’ll create and profile a simple classifier. The dataset linked to above provides all restaurant inspection data for the city of Detroit, from August 2016 to January 2019.

After extensive analysis and exploration in Power BI, some patterns emerge. Quarter 3 is the busiest for inspections, and Quarter 1 is the slowest. Routine inspections occur primarily on Tuesday, Wednesday, or Thursday. Hot dog carts are a roll of the dice.







This doesn’t seem too complex, and we theorize that we can create a classifier that predicts whether a restaurant is in compliance, by taking into account the number of violations in each of three categories (priority, core, and foundation).

To do so, we throw together some simple code that ingests the data, splits into a test and training set, creates the classifier model, and provides us the confusion matrix.

# Import the restaurant inspection dataset

df.rst.insp <- read.csv("Restaurant\_Inspections\_-\_All\_Inspections.csv", header = TRUE)

# A count of rows in the dataset

num.rows <- nrow(df.rst.insp)

# Create a shuffled subset of rows

subset.sample <- sample(num.rows, floor(num.rows\*.75))

# Create a training dataset using a shuffled subset of rows

df.training <- df.rst.insp[subset.sample,]

# Create a test dataset of all rows NOT in the training dataset

df.test <- df.rst.insp[-subset.sample,]

# Create the generalized linear model using the training data frame

mdlCompliance <- glm(In.Compliance ~ Core.Violations + Priority.Violations + Foundation.Violations, family = binomial, data = df.training)

# Predict the compliance of the test dataset

results <- predict(mdlCompliance, newdata=df.test, type = "response")

# Turn the response predictions into a binary yes or no

results <- ifelse(results < 0.5, "No", "Yes")

# Add the results as a new column to the data frame with the actual results

df.test$results <- results

# Output the confusion matrix

table(df.test$In.Compliance, df.test$results)

# Output the confusion matrix

library(caret)

confMat <- table(df.test$In.Compliance, df.test$results)

confusionMatrix(confMat, positive = "Yes")

An accuracy rate of 81.5%! That’s pretty great! Admittedly, a human wouldn’t have much trouble seeing a slew of priority violations and predicting a restaurant shutdown, but this classifier can perform the analysis at a much faster rate.

At this point, we have a good model we trust and expect to use for many years. Let’s pretend to fast forward a decade. Detroit’s meteoric rise has continued, the dataset has grown to massive amounts, and we begin to think we could improve the runtime. Easy enough! Profvis is here to give us the most user-friendly introduction to profiling available. To begin, simply install and load the package.

install.packages("profvis")

library("profvis")

Wrap your code in a profvis call, placing all code inside of braces. The braces are important, and be sure to put every line you want to profile. Maybe your confusion matrix is the bad part, or maybe you read the CSV in an inefficient way!

profvis({

df.rst.insp <- read.csv("Restaurant\_Inspections\_-\_All\_Inspections.csv", header = TRUE)

num.rows <- nrow(df.rst.insp)

subset.sample <- sample(num.rows, floor(num.rows\*.75))

df.training <- df.rst.insp[subset.sample,]

df.test <- df.rst.insp[-subset.sample,]

mdlCompliance <- glm(In.Compliance ~ Core.Violations + Priority.Violations + Foundation.Violations, family = binomial, data = df.training)

results <- predict(mdlCompliance, newdata=df.test, type = "response")

results <- ifelse(results < 0.5, "No", "Yes")

df.test$results <- results

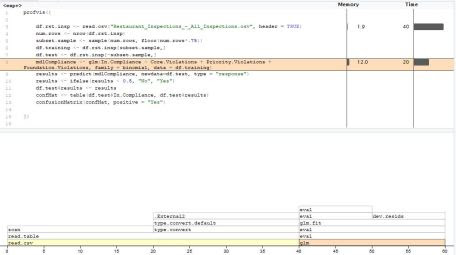
confMat <- table(df.test$In.Compliance, df.test$results)

confusionMatrix(confMat, positive = "Yes")

})

The output can help pinpoint poor-performing sections, and you can appropriately improve code where necessary.

The FlameGraph tab gives us a high-info breakdown. The Data tab gives us the bare-bones stats we need to get started.



profvis Data tab

In this example, we would certainly choose to improve the way we read in the data, since it accounts for two-thirds of the total run time in that single step!

The result here might be a minor gain, but we can easily understand how larger datasets would see massive performance improvements with a series of tweaks.