The Search for Categorical Correlation

Shaked Zychlinski Follow
Feb 23, 2018 · 6 min read

All the code appearing in this post is available as part of the `dython` *library on my GitHub page.*
For any code related questions, please open an issue on the library's GitHub page.

Not long ago I stumbled across a data-set of mushrooms on Kaggle, where over 20 different features of edible and poisonous mushrooms were collected and sorted into categories. The idea of seeking patterns that might point on how safe it is to eat a random mushroom seemed like a nice challenge — I even found myself creating a whole storyline of a lost man in the woods behind the kernel I published later on.

While going through other users' kernels, it was easy to see that Random Forests and other simple methods reach extremely high accuracy without too much effort, so I saw no reason doing so too — I've decided to see if can find by myself which features point towards which mushroom I can safely eat, if I'll ever need to. I realized what I'm actually looking for is the correlation between the features and the mushroom's type — but that's a problem, as the features are all categorical, and correlation isn't defined in that case.

What is correlation?

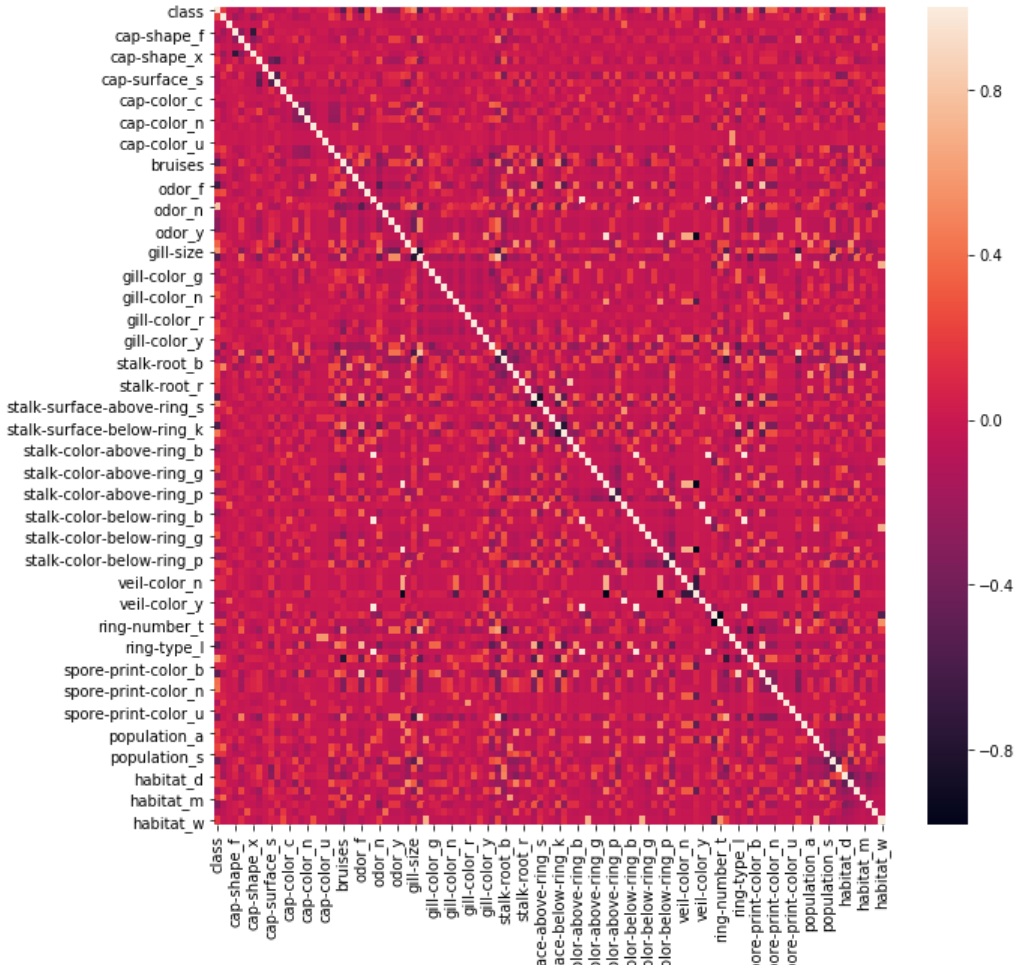
Before we can discuss about what correlation is not, let's talk about what it is. In human language, correlation is the measure of how two features are, well, correlated; just like the month-of-the-year is correlated with the average daily temperature, and the hour-of-the-day is correlated with the amount of light outdoors. Formalizing this mathematically, the definition of correlation usually used is Pearson's R for a data sample (which results in a value in the range [-1,1]):

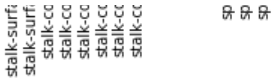
$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Pearson's R for data sample. Taken from Wikipedia

But, as can be seen from the above equation, Pearson's R isn't defined when the data is categorical; let's assume that *x* is a color feature — how do you subtract *yellow* from the *average of colors*? We need something else here.

One common option to handle this scenario is by first using one-hot encoding, and break each possible option of each categorical feature to 0-or-1 features. This will then allow the use of correlation, but it can easily become too complex to analyse. For example, one-hot encoding converts the 22 categorical features of the mushrooms data-set to a 112-features data-set, and when plotting the correlation table as a heat-map, we get something like this:





Correlation of the mushrooms data-set, transformed using one-hot encoding

This is not something that can be easily used for gaining new insights. So we still need something else.

Going categorical

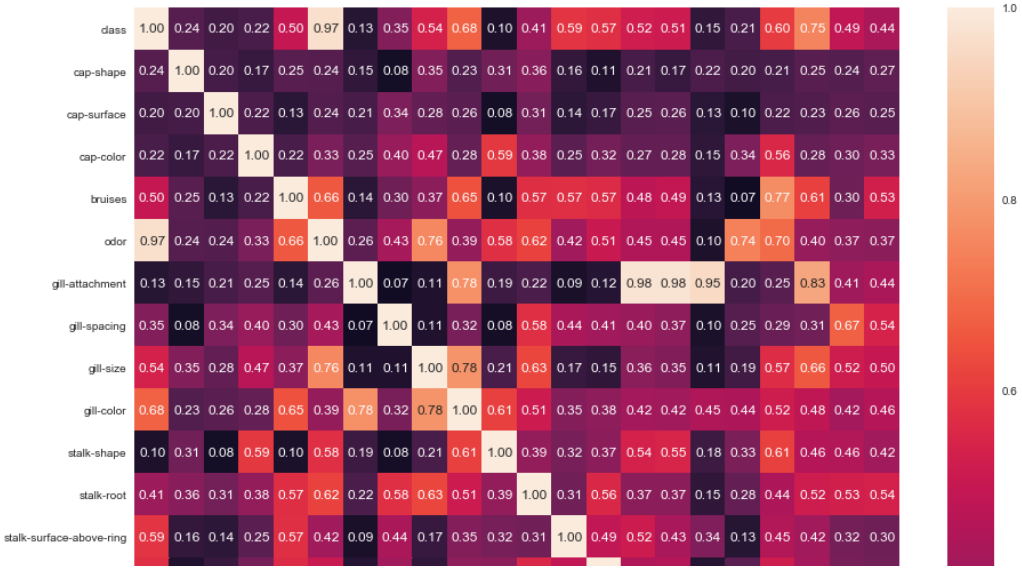
What we need is something that will *look* like correlation, but will work with categorical values — or more formally, we're looking for a *measure of association* between two categorical features. Introducing: Cramér's V. It is based on a nominal variation of Pearson's Chi-Square Test, and comes built-in with some great benefits:

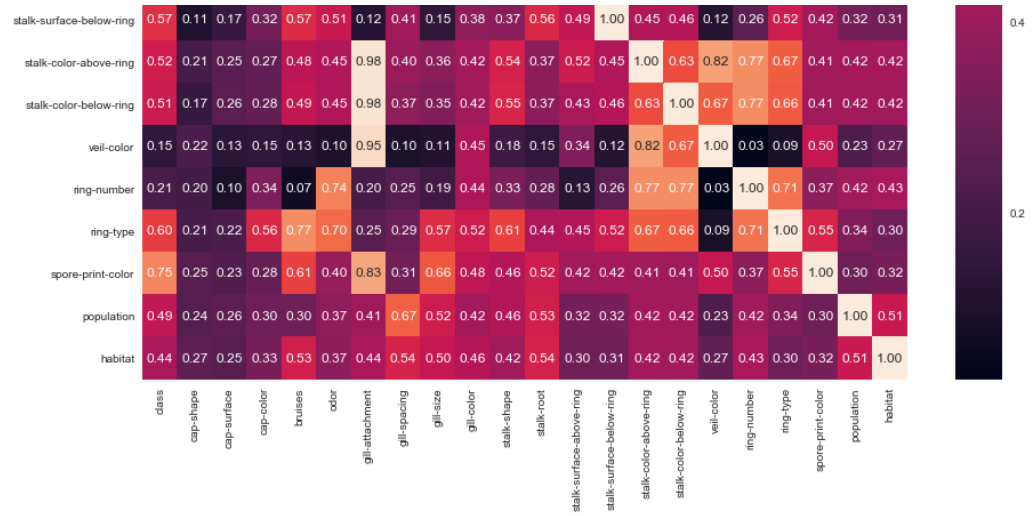
1. Similarly to correlation, the output is in the range of [0,1], where 0 means no association and 1 is full association. (Unlike correlation, there are no negative values, as there's no such thing as a negative association. Either there is, or there isn't)
2. Like correlation, Cramer's V is symmetrical — it is insensitive to swapping *x* and *y*

And what was even better — someone already implemented that as a Python function. And here's my edited version of the original:

```
def cramers_v(x, y):
    confusion_matrix = pd.crosstab(x,y)
    chi2 = ss.chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2/n
    r,k = confusion_matrix.shape
    phi2corr = max(0, phi2-((k-1)*(r-1))/(n-1))
    rcorr = r-((r-1)**2)/(n-1)
    kcorr = k-((k-1)**2)/(n-1)
    return np.sqrt(phi2corr/min((kcorr-1),(rcorr-1)))
```

When applied to the mushrooms data-set, it looks like this:





Cramer's V calculated for the mushrooms data-set

Well isn't that pretty? Just by looking at this heat-map we can see that the *odor* is highly associated with the *class* (edible/poisonous) of the mushroom, and that the *gill-attachment* feature is highly associated with three others.

The curse of symmetry

Thinking over the output of Cramer's V, I realized I'm losing valuable information due to the symmetry of it. To better demonstrate that, consider the following data-set:

We can see that if the value of x is known, the value of y still can't be determined, but if the value of y is known — then the value of x is guaranteed. This valuable information is lost when using Cramer's V due to its symmetry, so to preserve it we need an *asymmetric* measure of association between categorical features. And this is exactly what Theil's U is.

Theil's U, also referred to as the Uncertainty Coefficient, is based on the *conditional entropy* between x and y — or in human language, given the value of x, how many possible states does y have, and how often do they occur. Just like Cramer's V, the output value is on the range of [0,1], with the same interpretations as before — but unlike Cramer's V, it is asymmetric, meaning $U(x,y) \neq U(y,x)$ (while $V(x,y) = V(y,x)$, where V is Cramer's V). Using Theil's U in the simple case above will let us find out that knowing y means we know x, but not vice-versa.

Implementing the formula as a Python function yields this *(full code with the* `conditional_entropy` *function can be found on my Github page — link at the top of the post)*:

```
def theils_u(x, y):
    s_xy = conditional_entropy(x,y)
    x_counter = Counter(x)
    total_occurrences = sum(x_counter.values())
    p_x = list(map(lambda n: n/total_occurrences,
x_counter.values()))
    s_x = ss.entropy(p_x)
    if s_x == 0:
        return 1
    else:
        return (s_x - s_xy) / s_x
```

Applying this to the mushrooms data-set:

Theil's U calculated for the mushrooms data-set

This new calculation shed much more light on the associations we've seen from Cramer's V — for example, we now see that while knowing the *odor* gives a lot of information over

the mushroom's *class*, this is not in the case the other way around. Theil's U indeed gives us much more information on the true relations between the different features.

What happens when we mix things up?

So now we have a way to measure the correlation between two continuous features, and two ways of measuring association between two categorical features. But what about a pair of a continuous feature and a categorical feature? For this, we can use the Correlation Ratio (often marked using the greek letter *eta*). Mathematically, it is defined as the weighted variance of the mean of each category divided by the variance of all samples; in human language, the Correlation Ratio answers the following question: *Given a continuous number, how well can you know to which category it belongs to?* Just like the two coefficients we've seen before, here too the output is on the range of [0,1].

Implementation in Python looks like this:

```
def correlation_ratio(categories, measurements):
    fcat, _ = pd.factorize(categories)
    cat_num = np.max(fcat)+1
    y_avg_array = np.zeros(cat_num)
    n_array = np.zeros(cat_num)
    for i in range(0,cat_num):
        cat_measures = measurements[np.argwhere(fcat ==
i).flatten()]
        n_array[i] = len(cat_measures)
        y_avg_array[i] = np.average(cat_measures)
    y_total_avg =
np.sum(np.multiply(y_avg_array,n_array))/np.sum(n_array)
    numerator =
np.sum(np.multiply(n_array,np.power(np.subtract(y_avg_array,y_total_
avg),2)))
    denominator =
np.sum(np.power(np.subtract(measurements,y_total_avg),2))
    if numerator == 0:
        eta = 0.0
    else:
        eta = np.sqrt(numerator/denominator)
    return eta
```

Final words

I believe I can declare the search for a measure of association for categorical features a successful one, especially as certain requirements — such as the need for an asymmetric measure — were not expected when starting. These three new metrics are very useful when exploring a data-set which contains categorical features, and helped me gain more insights on data-sets I've explored. I can only hope this will be useful to you as it was to me, and if not — well, at least you now know how to identify edible mushrooms.

Data Analysis Data Visualization Correlation