RIP correlation. Introducing the Predictive Power Score

We define and open-source the Predictive Power Score (PPS). The PPS is an alternative to the correlation that finds more patterns in your data.

Florian Wetschoreck Follow
Apr 23 · 13 min read ★

. . .

Too many problems with the correlation

It is Friday afternoon and your boss tells you that the data delivery surprisingly arrived early — after only 4 weeks of back and forth. This was the missing piece for your predictive model. You're excited but also a little bit anxious because you know what's next: exploring the data. All. 45. Columns. This will take many hours but you know it's worth it because without data understanding you are walking blind. One obvious step is to have a look at all the univariate column distributions. But this won't be enough.

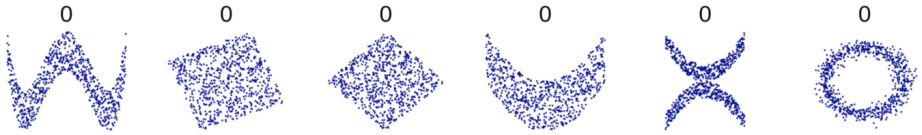
> You ask yourself: what relationships exist between columns?

To answer this question, you just repeat the typical drill: calculate a correlation matrix and check for some surprising relationships. Whenever you are surprised, you take a moment to plot a scatterplot of the two columns at hand and see if you can make any sense of it. Hopefully you can but too often you cannot because you don't even know what the columns mean in the first place. But this is a story for another day.

> After inspecting the correlation matrix you move on and you don't even know what you don't know (scary).

Let's take a moment to review the correlation. The score ranges from -1 to 1 and indicates if there is a **strong linear relationship** — either in a positive or negative direction. So far so good. However, there are **many non-linear relationships** that the score simply won't detect. For example, a sinus wave, a quadratic curve or a mysterious step function. The score will just be 0, saying: "Nothing interesting here". Also,

OneHotEncoding will create a matrix that has more values than there are atoms in the universe.



Too many scenarios where the correlation is 0. This makes me wonder if I missed something... (Excerpt from the image by Denis Boigelot)

If you are a little bit too well educated you know that the **correlation matrix is symmetric**. So you basically can throw away one half of it. Great, we saved ourselves some work there! Or did we? Symmetry means that the correlation is the same whether you calculate the correlation of A and B or the correlation of B and A. However, relationships in the real world are rarely symmetric. More often, **relationships are asymmetric**. Here is an example: The last time I checked, my zip code of 60327 tells strangers quite reliably that I am living in Frankfurt, Germany. But when I only tell them my city, somehow they are never able to deduce the correct zip code. Pff … amateurs. Another example is this: a column with 3 unique values will never be able to perfectly predict another column with 100 unique values. But the opposite might be true. Clearly, asymmetry is important because it is so common in the real world.

> Thinking about those shortcomings of correlation, I started to wonder: can we do better?

The requirements: One day last year, I was dreaming about a score that would tell me if there is any relationship between two columns — no matter if the relationship is linear, non-linear, gaussian or only known by aliens. Of course, the score should be asymmetric because I want to detect all the weird relationships between cities and zip codes. The score should be 0 if there is no relationship and the score should be 1 if there is a perfect relationship. And as the icing on the cake, the score should be able to handle categoric and numeric columns out of the box. Summing it up for all my academic friends: an asymmetric, data-type-agnostic score for predictive relationships between two columns that ranges from 0 to 1.

. . .

Calculating the Predictive Power Score (PPS)

> *First of all,* ***there is not the one and only way*** *to calculate the predictive power score. In fact, there are* ***many possible ways*** *to calculate a score that satisfies the requirements mentioned before. So, let's rather think of the predictive power score as a framework for a family of scores.*

evaluation metric. When the target is numeric we can use a Decision Tree Regressor and calculate the Mean Absolute Error (MAE). When the target is categoric, we can use a Decision Tree Classifier and calculate the weighted F1. You might also use other scores like the ROC etc but let's put those doubts aside for a second because we have another problem:

> Most evaluation metrics are meaningless if you don't compare them to a baseline

I guess you all know the situation: you tell your grandma that your new model has a F1 score of 0.9 and somehow she is not as excited as you are. In fact, this is very smart of her because she does not know if anyone can score 0.9 or if you are the first human being who ever scored higher than 0.5 after millions of awesome KAGGLErs tried. So, we need to "normalize" our evaluation score. And how do you normalize a score? You define a lower and an upper limit and put the score into perspective. So what should the lower and upper limit be? Let's start with the upper limit because this is usually easier: a perfect F1 is 1. A perfect MAE is 0. Boom! Done. But what about the lower limit? Actually, we cannot answer this in absolute terms.

> The lower limit depends on the evaluation metric and your data set. It is the value that a naive predictor achieves.

If you achieve a F1 score of 0.9 this might be super bad or really good. If your super fancy cancer detection model always predicts "benign" and it still scores 0.9 on that highly skewed dataset then 0.9 is obviously not so good. So, we need to calculate a score for a very naive model. But what is a naive model? For a classification problem, always predicting the most common class is pretty naive. For a regression problem, always predicting the median value is pretty naive.

Let's have a look at a detailed, fictional example:

Getting back to the example of the zip codes and the city name. Imagine both columns are categoric. First, we want to calculate the PPS of zip code to city. We use the weighted F1 score because city is categoric. Our cross-validated Decision Tree Classifier achieves a score of 0.95 F1. We calculate a baseline score via always predicting the most common city and achieve a score of 0.1 F1. If you normalize the score, you will get a final PPS of 0.94 after applying the following normalization formula: (0.95–0.1) / (1–0.1). As we can see, a PPS score of 0.94 is rather high, so **the zip code seems to have a good predictive power towards the city**. However, if we calculate the PPS in the opposite direction, we might achieve a PPS of close to 0 because the Decision Tree Classifier is not substantially better than just always predicting the most common zip code.

> *Please note: the normalization formula for the MAE is different from the F1. For MAE lower*

Comparing the PPS to correlation

In order to get a better feeling for the PPS and its differences to the correlation, let's have a look at the following two examples:

Example 1: Non-linear effects and asymmetry

Example 1: Comparison of Correlation to PPS for $y=x^2$

Let's use a typical quadratic relationship: the feature x is a uniform variable ranging from -2 to 2 and the target y is the square of x plus some error. In this case, x can predict y very well because there is a clear non-linear, quadratic relationship — after all that's how we generated the data. However, this is not true in the other direction from y to x. For example, if y is 4, it is impossible to predict whether x was roughly 2 or -2. Thus, the predictive relationship is asymmetric and the scores should reflect this.

What are the values of the scores in this example? If you don't already know what you are looking for, the correlation will leave you hanging because the **correlation is 0**. Both from x to y and from y to x because the correlation is symmetric. However, the **PPS from x to y is 0.88**, detecting the non-linear relationship and saving the day. Nevertheless, the PPS is not 1 because there exists some error in the relationship. In the other direction, the **PPS from y to x is 0** because there is no relationship that y can predict if it only knows its own value. This is consistent with the observations that we made before.

Example 2: Categorical columns and hidden patterns

Let's compare the correlation matrix to the PPS matrix on the Titanic dataset. "The Titanic dataset? Again??" I know, you probably think you already have seen everything about the Titanic dataset but maybe the PPS will give you some new insights.

Example 2: Comparing the Pearson correlation matrix (left) with the PPS matrix (right) for the Titanic dataset.

Two findings about the correlation matrix:

1. The correlation matrix is smaller and leaves out many interesting relationships. Of course, that makes sense because columns like Sex, TicketID or Port are categoric and the correlation cannot be computed for them.

2. The correlation matrix shows a negative correlation between TicketPrice and Class of medium strength (-0.55). We can double-check this relationship if we have a look at the PPS. We will see that the TicketPrice is a strong predictor for the Class (0.9 PPS) but not vice versa. The Class only predicts the TicketPrice with a PPS of 0.2. This makes sense because whether your ticket did cost 5.000$ or 10.000$ you were most likely in the highest class. In contrast, if you know that someone was in the highest class you cannot say whether they paid 5.000$ or 10.000$ for their ticket. In this scenario, the asymmetry of the PPS shines again.

Four findings about the PPS matrix:

1. The first row of the matrix tells you that the **best univariate predictor** of the column Survived is the column Sex. This makes sense because women were prioritized during the rescue. (We could not find this information in the correlation matrix because the column Sex was dropped.)

2. If you have a look at the column for TicketID, you can see that TicketID is a fairly good predictor for a range of columns. If you further dig into this pattern, **you will find out that multiple persons had the same TicketID**. Thus, the TicketID is actually referencing a latent group of passengers who bought the ticket together, for example the big Italian Rossi family that turns any evening into a spectacle. Thus, the PPS helped me to **detect a hidden pattern**.

3. What's even more surprising than the strong predictive power of TicketID is the strong predictive power of TicketPrice across a wide range of columns. Especially, the fact that the TicketPrice is fairly good at predicting the TicketID (0.67) and vice versa (0.64). Upon further research you will find out that tickets often had unique prices. For example, only the Italian Rossi family paid a price of 72,50$. This is a critical insight! It means that **the TicketPrice contains information about the TicketID** and thus about our Italian family. An information that you need to have when considering **potential information leakage**.

be surprised why the TicketPrice has predictive power on the survival rate (PPS 0.39). But if you know that the Class influences your survival rate (PPS 0.36) and that the TicketPrice is a good predictor for your Class (PPS 0.9), then you might have found an explanation.

. . .

Applications of the PPS and the PPS matrix

After we learned about the advantages of the PPS, let's see where we can use the PPS in the real life.

> *Disclaimer: There are use cases for both the PPS and the correlation. The PPS clearly has some advantages over correlation for **finding predictive patterns** in the data. However, once the patterns are found, the correlation is still a great way of **communicating found linear relationships**.*

- **Find patterns in the data:** The PPS finds every relationship that the correlation finds — and more. Thus, you can use the PPS matrix as an alternative to the correlation matrix to detect and understand linear or nonlinear patterns in your data. This is possible across data types using a single score that always ranges from 0 to 1.
- **Feature selection:** In addition to your usual feature selection mechanism, you can use the predictive power score to find good predictors for your target column. Also, you can eliminate features that just add random noise. Those features sometimes still score high in feature importance metrics. In addition, you can eliminate features that can be predicted by other features because they don't add new information. Besides, you can identify pairs of mutually predictive features in the PPS matrix — this includes strongly correlated features but will also detect non-linear relationships.
- **Detect information leakage:** Use the PPS matrix to detect information leakage between variables — even if the information leakage is mediated via other variables.
- **Data Normalization:** Find entity structures in the data via interpreting the PPS matrix as a directed graph. This might be surprising when the data contains latent structures that were previously unknown. For example: the TicketID in the Titanic dataset is often an indicator for a family.

. . .

How to use the PPS in your own (Python) project

If you are still following along you are one of the rare human beings who still have an attention span — you crazy beast! If you can't wait to see what the PPS will reveal on your own data, we have some good news for you: we open-sourced an implementation

Installing the package:

```
pip install ppscore
```

Calculating the PPS for a given pandas dataframe:

```
import ppscore as pps
pps.score(df, "feature_column", "target_column")
```

You can also calculate the whole PPS matrix:

```
pps.matrix(df)
```

How fast is the PPS in comparison to the correlation?

Although the PPS has many advantages over the correlation, there is some drawback: it takes longer to calculate. But how bad is it? Does it take multiple weeks or are we done in a couple of minutes or even seconds? When calculating a **single PPS** using the Python library, the time should be no problem because it usually takes around **10–500ms**. The calculation time mostly depends on the data types, the number of rows and the used implementation. However, when calculating the whole PPS matrix for 40 columns this results in 40*40=**1600 individual calculations** which **might take 1–10 minutes**. So you might want to start the calculation of the PPS matrix in the background and go on that summer vacation you always dreamed of! 🚢 For our projects and datasets the computational performance was always good enough but of course there is room for improvement. Fortunately, we see many ways how the calculation of the PPS can be improved to achieve speed gains of a factor of 10–100. For example, using intelligent sampling, heuristics or different implementations of the PPS. If you like the PPS and are in need of a faster calculation, please reach out to us.

. . .

Limitations

We made it — you are excited and want to show the PPS to your colleagues. However, you know they are always so critical about new methods. That's why you better be prepared to know the limitations of the PPS:

1. The calculation is slower than the correlation (matrix).
2. The score cannot be interpreted as easily as the correlation because it does not tell you anything about the **type of relationship** that was found. Thus, the PPS is better for finding patterns but the correlation is better to communicate found linear relationships.

4. There are limitations of the components used underneath the hood. Please remember: you might exchange the components e.g. using a GLM instead of a Decision Tree or using ROC instead of F1 for binary classifications.

5. If you use the PPS for feature selection you still want to perform forward and backward selection in addition. Also, the PPS cannot detect interaction effects between features towards your target.

. . .

Conclusion

After years of using the correlation we were so bold (or crazy?) to suggest an alternative that can detect linear and non-linear relationships. The PPS can be applied to numeric and categoric columns and it is asymmetric. We proposed an implementation and open-sourced a Python package. In addition, we showed the differences to the correlation on some examples and discussed some new insights that we can derive from the PPS matrix.

Now it is up to you to decide what you think about the PPS and if you want to use it on your own projects. We have been using the PPS for over a year as part of the library bamboolib where the PPS is essential to add some advanced features and thus we wanted to share the PPS with the broader community. Therefore, we hope to receive your feedback about the concept and we would be thrilled if you try the PPS on your own data. In case that there might be a positive reception, we are happy to hear about your requests for adjustments or improvements to the implementation. As we mentioned before, there are many ways on how to improve the speed and on how to adjust the PPS for more specific use cases.

Github: https://github.com/8080labs/ppscore

Email: florian AT 8080labs.com

Newsletter: if you want to hear more about the PPS and our other upcoming Data Science projects and tools, you can **subscribe here**. We will not write about paid products, you can unsubscribe anytime and — sad that we even have to mention this — we will never give away your email.

. . .

Originally published at https://8080labs.com on April 23, 2020.

Thanks to Yenson Lau.

Data Science Exploratory Data Analysis Correlation Python Open Source