**Why Violin Plots Are Awesome for Feature Engineering:**

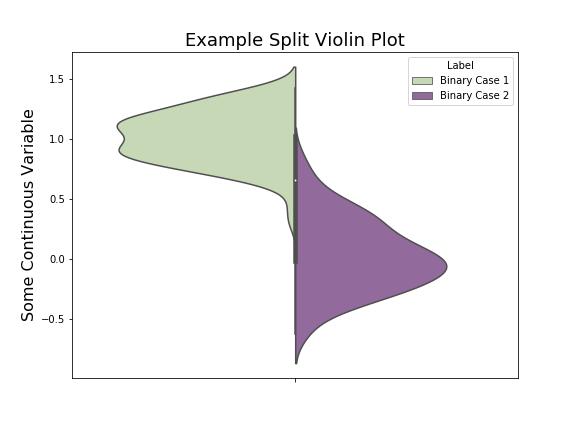
**An Example Using NLP To Identify Similar Products**

*At Wayfair, technology and data expertise enable data scientists to transform new web datasets into intelligent machine algorithms that re-imagine how traditional commerce works. In this post, we introduce how visual tools like Violin Plots amplify our data acumen to unlock deep insights. The savvy data scientist recognizes the value of a Violin Plot when engineering new model features. We share how this method is applied in an e-commerce example where fuzzy text matching systems are developed to identify similar products sold online.*

*Key article takeaways:*

* *Skillful usage of Violin Plots can improve feature engineering and selection.*
* *A good Violin Plot communicates more information about data irregularities than standard summary statistics and correlation coefficients*

Good data visualizations are helpful at every step of a data science project. When starting out, good data visualizations can inform how one should formulate the problem. Visualizations also can help guide decisions surrounding which data inputs to use, and are helpful when evaluating model accuracy and feature importance. When debugging an existing model, visualizations help diagnose data irregularities and bias in model predictions. Finally, when communicating with business stakeholders, the right visualization makes a clear point without any additional explanation.

One data visualization that is particularly helpful when working on binary classification problems is the **split violin plot**. In my experience, this is a type of plot that is not nearly as famous as it should be. In brief, a split violin plot takes a variable grouped by two categories and plots a smoothed histogram of the variable in each group on opposite sides of a shared axis.

**What I like most about violin plots is that they show you the entire distribution of your data. If data inputs violate your assumptions (e.g. multimodal, full of null values, skewed by bad imputation or extreme outliers) you see the problems at a quick glance and in incredible detail.** This is better than a few representative percentiles as in a box and whisker plot, or a table of summary statistics. They avoid the problem of oversaturation prevalent in scatter plots with lots of points, and reveal outliers more clearly than you would in a histogram without a lot of fine-tuning.

We’ll illustrate these advantages in a simple example where we use fuzzy string matching to engineer features for a binary classification problem.

**An Example using NLP to Identify Similar Products**

At Wayfair, we develop sophisticated algorithms to parse large product catalogs and identify similar products. Part of this project involves engineering features for a model which flags two products as the same or not. Let’s start from a dataset that provides several pairs of product names and a label indicating whether or not they refer to the same item.

|  |  |  |
| --- | --- | --- |
| Product1 | Product2 | Match |
| Da-Plex Rigid Rear Black Fixed Frame Projection Screen | Kohler K527E1SN DTV Prompt Shower Interface with ECO Mode | False |
| Ruby Coronet Oval Platter 16" Ruby Coronet | 1 Door Outdoor Enclosed Bulletin Board Size: 3' H x 2' W | False |
| Lower Case Letter Painting Print on Wrapped Canvas 28" x 28" - Yel... | Letter - Lower Case 'p' Stretched Wall Art Size: 28" x 28" | True |
| 67" x 29.5" Soaking Bathtub Kit | Fanmats Alabama State Football Rug 20.5"x32.5" | False |

**Fuzzywuzzy Similarity Scores**

For the purpose of this fuzzy text matching illustration, let’s use an open-source Python library called [fuzzywuzzy](https://github.com/seatgeek/fuzzywuzzy) (developed by the fine folks at [SeatGeek](https://seatgeek.com)). This library contains several functions for measuring the similarity between two strings. Each function takes in two strings and returns a number between 0 and 100 representing the similarity between the strings. Functions differ in their conventions and different functions will produce different similarity results.

from fuzzywuzzy import fuzz

fuzz.QRatio('brown leather sofa', '12ft leather dark brown sofa')  
>>> 57

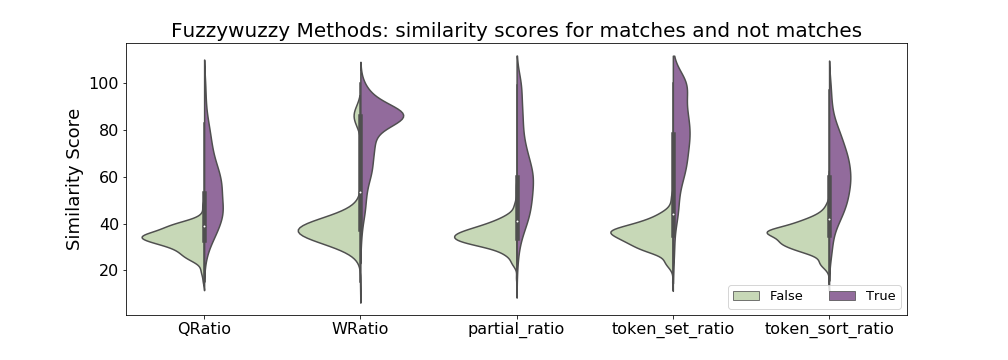
fuzz.WRatio('brown leather sofa', '12ft leather dark brown sofa')  
>>> 86

fuzz.token\_set\_ratio('brown leather sofa', '12ft leather dark brown sofa')  
>>> 100

It’s rarely obvious which function is best for a given problem. Let’s consider five different fuzzy matching methods and compute similarity scores for each pair of strings. Using these scores, we’ll create some violin plots to determine which method is best for distinguishing between matches and not matches. (You could also consider combinations of scores, though this comes at a higher computational cost.)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Product1 | Product2 | Match | QRatio | WRatio | partial | token\_set | token\_sort |
| Da-Plex Rigid Rear Black Fixed Frame Projection Screen 60" H x 60" W | Kohler K527E1SN DTV Prompt Shower Interface with ECO Mode | False | 33 | 36 | 30 | 37 | 34 |
| Ruby Coronet Oval Platter 16" Ruby Coronet | 1 Door Outdoor Enclosed Bulletin Board Size: 3' H x 2' W | False | 33 | 34 | 31 | 32 | 36 |
| Lower Case Letter Painting Print on Wrapped Canvas 28" x 28" - Yellow - p Letters | Letter - Lower Case 'p' Stretched Wall Art Size: 28" x 28" | True | 45 | 64 | 50 | 67 | 59 |
| 67" x 29.5" Soaking Bathtub Kit | Fanmats Alabama State Football Rug 20.5"x32.5" | False | 24 | 41 | 27 | 42 | 43 |

A few lines of code are all we need to generate split violin plot using the [Seaborn](https://seaborn.pydata.org/) library. The purple distribution depicts a smoothed (sideways) histogram of fuzzy matching scores when Match is True, while the light-green shows the distribution of similarity scores when Match is False. **When two distributions have little or no overlap along the y-axis, the fuzzy matching function will do a better job distinguishing between our binary classes.**



Generally, these fuzzy matching scores do a good job in distinguishing between observations where the two names refer to the same product. For any method, a pair of names with a similarity score of 50 or more will probably refer to the same product.

Still, we can see that some fuzzy matching functions do a better job than others in distinguishing between True and False observations. The *token\_set\_ratio* plot seems to have the least overlap between the True and False distributions, followed by the plots for *token\_sort\_ratio* and *WRatio*. Of our five similarity scores, the scores from these methods should perform the best in any predictive model. In comparison, notice how much more the True and False distributions overlap for the *partial\_ratio* and *QRatio* methods. Scores from these methods will be less helpful as features.

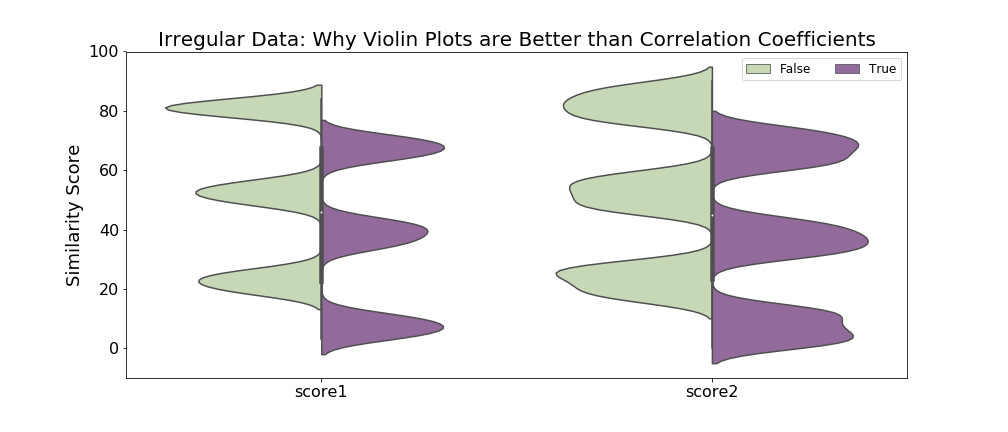
**Conclusion: Violin plots suggest that of our five similarity scores, *token\_set\_ratio* would be the best feature in a predictive model, especially compared to the *partial\_ratio* or *QRatio* methods.**

**Why Violin Plots are Superior to More Conventional Analyses**

For comparison, let’s look at the Pearson correlation coefficients between our fuzzy-matching scores and our indicator variable for whether the pair is a match or not.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | QRatio | WRatio | partial\_ratio | token\_set\_ratio | token\_sort\_ratio |
| Match | 0.68 | 0.80 | 0.71 | **0.84** | 0.76 |

For this data, the correlation coefficients give a similar ranking as achieved using the violin plots. The *token\_set\_ratio* method gives the strongest correlation to the Match variable while the *QRatio* method gives the weakest correlation. If our goal was only to identify the best fuzzywuzzy function to use, we apparently could have made our selection using correlation coefficients instead of violin plots. In general, however, violin plots are much more reliable and informative. Consider the following (pathological) example.



In these violin plots, the similarity scores on the left appear to be more helpful in separating between matches and not-matches. There is less overlap between the True and False observations and the observations are more tightly clustered into their respective groups.

However, notice that the relationship between the similarity scores and the True/False indicator is not at all linear or even monotone. As a result, correlation coefficients can fail to correctly guide our decision on which set of scores to use. Is this true? Let’s take a look.

|  |  |  |
| --- | --- | --- |
|  | score1 | score2 |
| Match | -0.28 | -0.30 |

Here, the correlation coefficients of *score1* and *score2* against the outcome variable are quite close. However, the plot on the right –the one that doesn’t cleanly separate True and False observations– has the stronger correlation coefficient. **If we blindly took the series with the strongest correlation, we would choose the less helpful of the two features.**

**Wrapping up…**

To summarize:

1. **Split violin plots** are a great way to visualize your data at a quick glance, especially when dealing with binary classification problems.
2. Violin plots can guide us in **feature engineering and selection** by revealing the variables that best separate the two classification outcomes.
3. **Correlation coefficients**, in comparison, can accomplish a similar task when the relationship between the features and labels are linear. When linearity is violated, the correlation coefficients are misleading in comparison to the violin plots.

There are certainly limits to this approach. Nothing that requires an “eye test” is scalable to many features. Also, violin plots have a few important parameters which, if not properly set, can hide important patterns in the data. Still, when properly used, split violin plots are a great tool for binary classification type problems.

*Acknowledgements*

Special thanks to **Zhenyu Lai, Aditya Karan, Brad Fay** and **Laura Tengelsen** for excellent editing and feedback.

**Authors: Benjamin Tengelsen**