



Open in app

Get started



Published in Towards Data Science

You have **2** free member-only stories left this month. [Sign up for Medium and get an extra one](#)



Francois Bertrand

Follow

Jun 3, 2020 · 11 min read · ✨ · 🎧 Listen



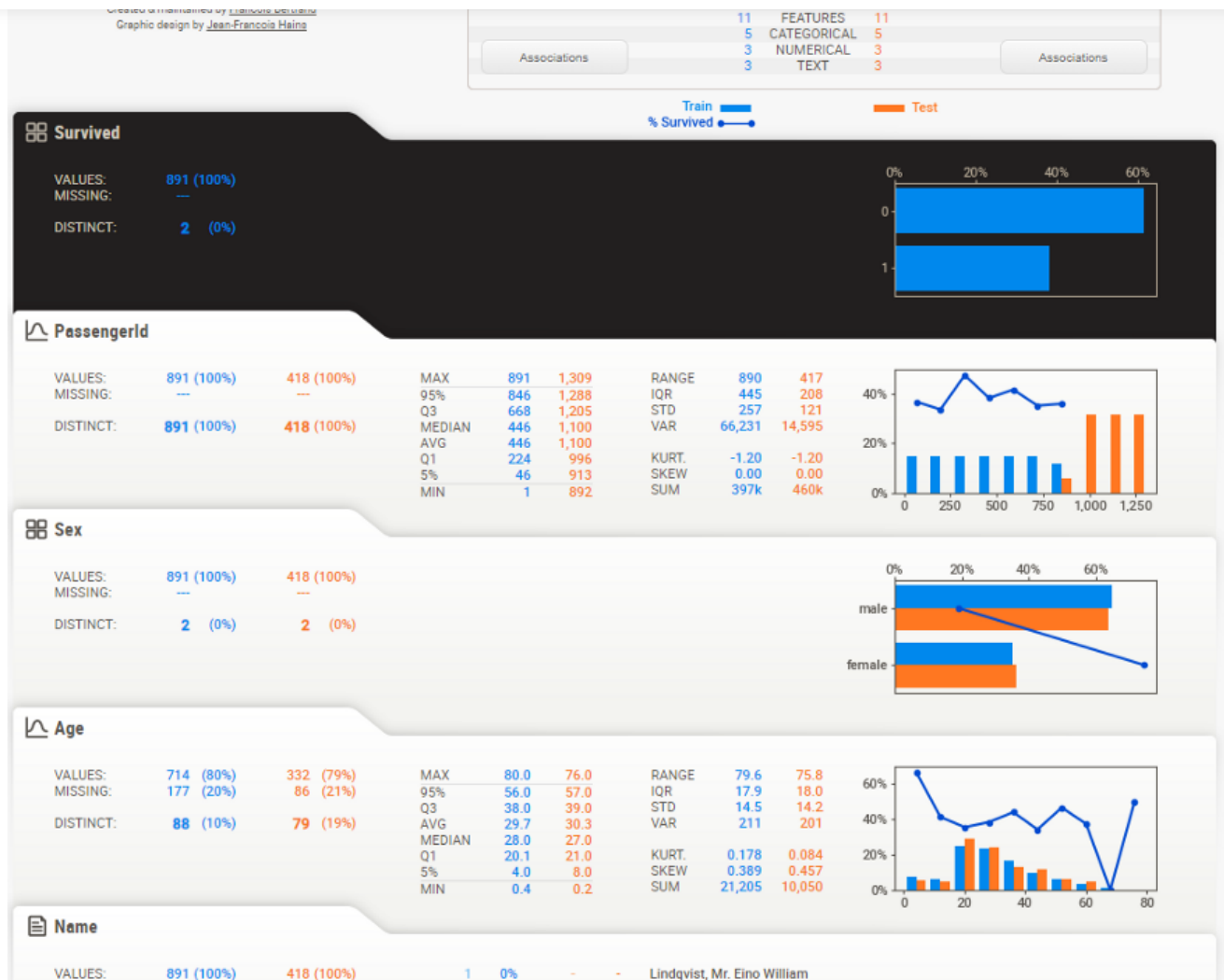
Save



# Powerful EDA (Exploratory Data Analysis) in just two lines of code using Sweetviz

Get to know your data much faster with this new Python library



[Open in app](#)[Get started](#)

Sweetviz report created using only two lines of code

## Update January 2021: Sweetviz now supports notebook integration (you can insert reports inline) and new features!

Exploratory data analysis (EDA) is an essential early step in most data science projects and it often consists of taking the same steps to characterize a dataset (e.g. find out data types, missing information, distribution of values, correlations, etc.). Given the repetitiveness and similarity of such tasks, there are a few libraries that automate and help kickstart the process.

One of the latest is a new open-source Python library called Sweetviz ([GitHub](#)), created by a few contributors and myself for just that purpose. It takes pandas dataframes and creates a self-contained HTML report.

It makes a powerful punch, in addition to creating insightful and beautiful





- **Comparison of 2 datasets** (e.g. Train vs Test)
- **Visualization of the target value** against all other variables (e.g. “What was the survival rate of male vs female” etc.)

Here is a link to a report generated by Sweetviz for the well-known sample Titanic Survivor dataset. We will be analyzing this report in this article.

## EDA made... fun?!

Being able to get so much information so quickly about the target value and compare different areas of the dataset almost instantly transforms this initial step from being out all about tedium to being quicker, interesting and even to some degree... fun! (to this data geek, at least!) Of course EDA is a much longer process but at least that first step is a lot smoother. Let's see how it works out with the well-known sample dataset.

## Analyzing the Titanic dataset

For this article, we will be analyzing the sample Titanic Survivor dataset you can find [here](#).

After installation of Sweetviz (using `pip install sweetviz`), simply load the pandas dataframes as you normally would, then call either `analyze()`, `compare()` or `compare_intra()` depending on your need (more on that below). The full documentation can be found on [GitHub](#). For now, let's start with the case at hand, loading it as so:

```
import sweetviz
import pandas as pd
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

We now have 2 dataframes (train and test), and we would like to analyze the target value “Survived”. I want to point out in this case we know the name of the target column in advance, but it is always optional to specify a target column. We can generate a report with this line of code:





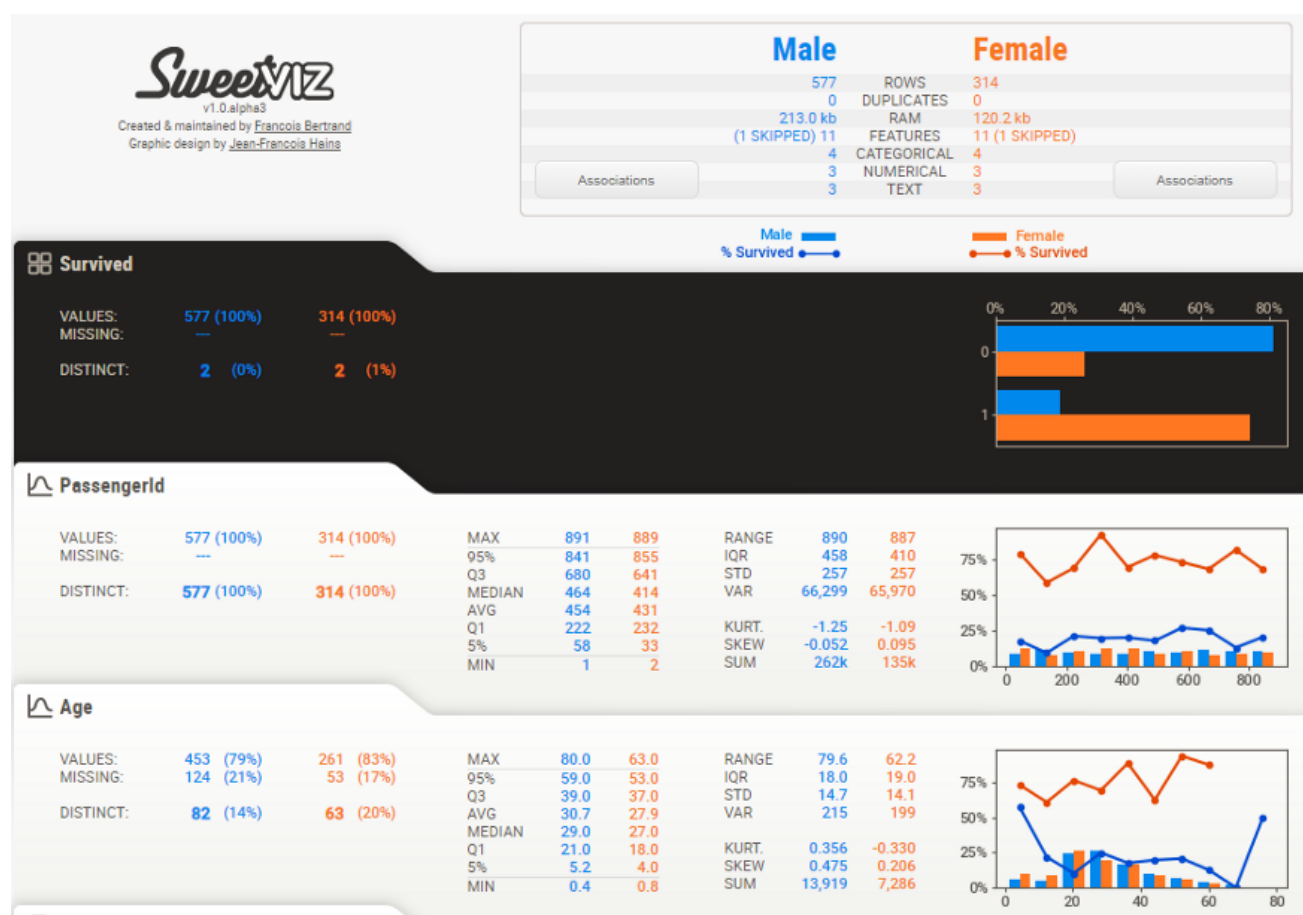
Open in app

Get started

Running this command will perform the analysis and create the report object. To get the output, simply use the `show_html()` command:

```
my_report.show_html("Report.html") # Not providing a filename will default to SWEETVIZ_REPORT.html
```

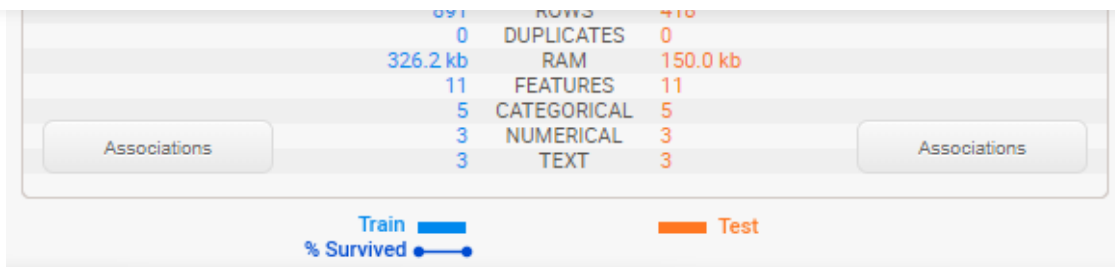
After generating the file, it will open it through your default browser and should look something like this:



There's a lot to unpack, so let's take it one step at a time!

## Summary display



[Open in app](#)[Get started](#)

The summary shows us the characteristics of both dataframes side-by-side. We can immediately identify that the testing set is roughly half the size of the training set, but that it contains the same features. That legend at the bottom shows us that the training set does contain the “Survived” target variable, but that the testing set does not.

Note that Sweetviz will do a best guess at determining the data type of each column, between numerical, category/boolean and text. These can be overridden, more on that below.

## Associations

Hovering your mouse over the “Associations” button in the summary will make the Associations graph appear on the right-hand side:

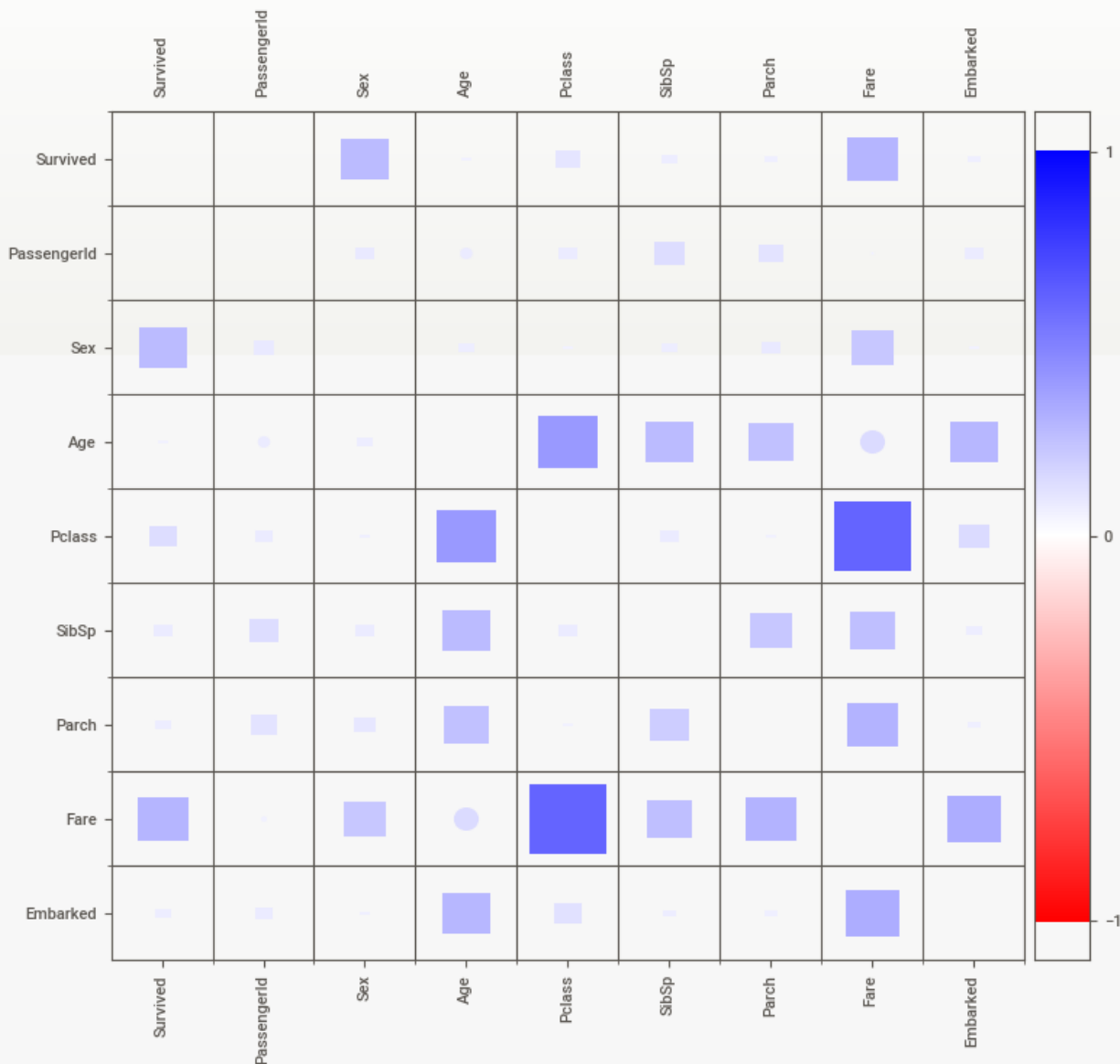




Open in app

Get started

• CIRCLES are numerical correlations (Pearson's) from -1 to 1.  
• The trivial DIAGONAL is intentionally left blank for clarity.



This graph is a composite of the visuals from [Drazen Zaric: Better Heatmaps and Correlation Matrix Plots in Python](#) and concepts from [Shaked Zychlinski: The Search for Categorical Correlation](#).

Basically, in addition to showing the traditional numerical correlations, it unifies in a single graph both numerical correlation but also the uncertainty coefficient (for categorical-categorical) and correlation ratio (for categorical-numerical). Squares represent categorical-featured-related variables and circles represent numerical-numerical correlations. Note that the trivial diagonal is left empty, for clarity.

IMPORTANT: This visualization is a composite of the visuals from Drazen Zaric: Better Heatmaps and Correlation Matrix Plots in Python and concepts from Shaked Zychlinski: The Search for Categorical Correlation.

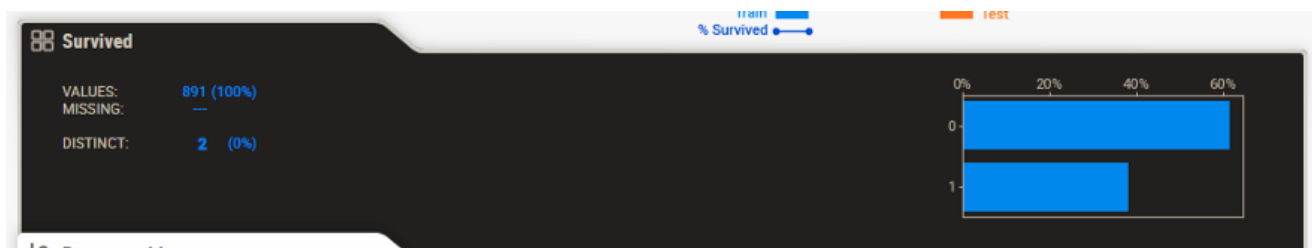


[Open in app](#)[Get started](#)

information is rather symmetrical but it is not always the case.

Finally, it is worth noting these correlation/association methods shouldn't be taken as gospel as they make some assumptions on the underlying distribution of data and relationships. However they can be a very useful starting point.

## Target variable



When a target variable is specified, it will show up first, in a special black box.

**IMPORTANT: only numerical and boolean features can be targets currently.**

We can gather from this summary that “Survived” has no missing data in the training set (891, 100%), that there are 2 distinct possible values (accounting for less than 1% of all values), and from the graph it can be estimated that roughly 60% did not survive.

## Detail area (categorical/boolean)

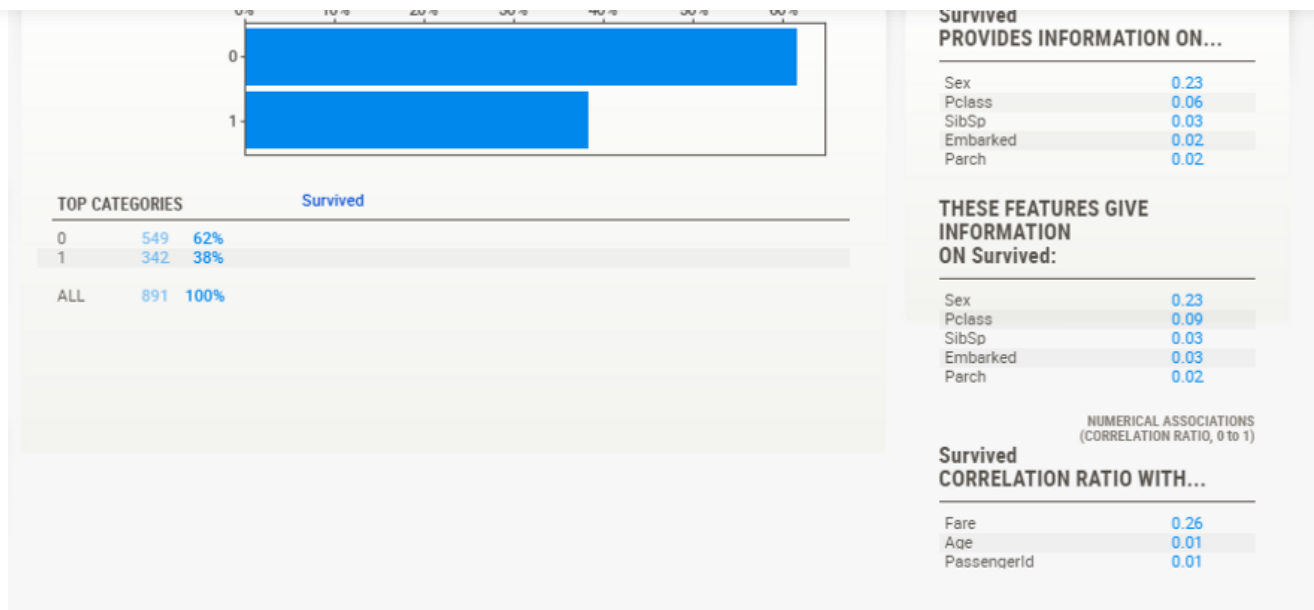
When you move the mouse to hover over any of the variables, an area to the right will showcase the details. The content of the details depends on the type of variable being analyzed. In the case of a categorical (or boolean) variable, as is the case with the target, the analysis is as follows:





Open in app

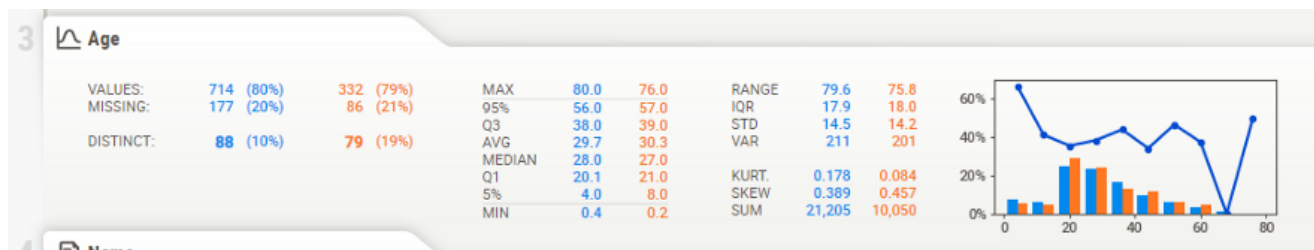
Get started



*IMPORTANT: a “widescreen” monitor is required to see the full detail area for the moment.*

Here, we can see the exact statistics for each class, where 62% did not survive and 38% survived. You also get the detail of the associations for each of the other features.

## Numerical data



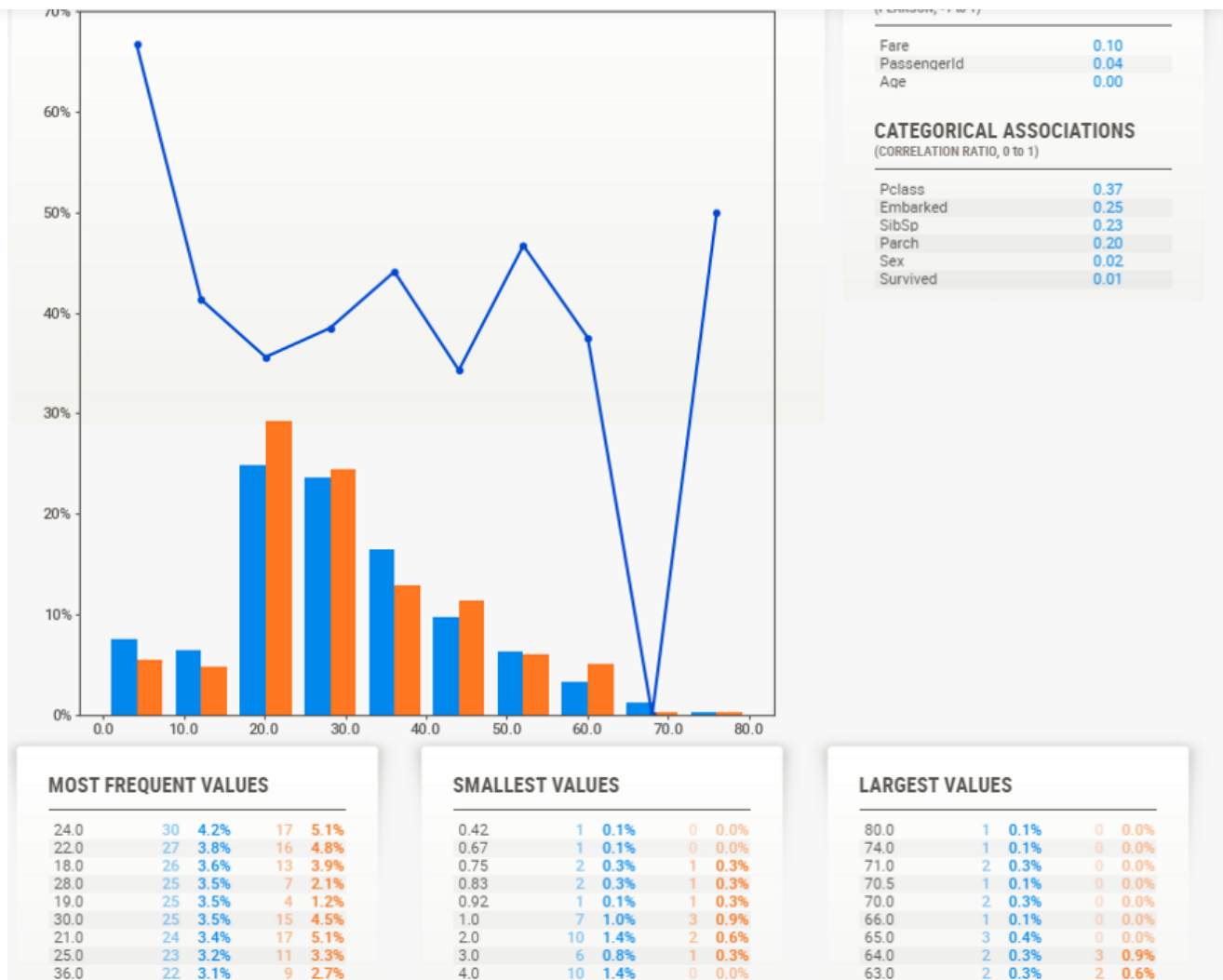
Numerical data shows more information on its summary. Here, we can see that in this case about 20% of data is missing (21% in the test data, which is very consistent).

**Note that the target value (“Survived” in this case) is plotted as a line, right over the distribution graph. This enables an instant analysis of the target distribution with regard to other variables.**

Interestingly, we can see from the graph on the right that the survival rate is pretty consistent across all ages, except for the youngest which have a higher survival rate. It would look like “women and children first” was not just talk.





[Open in app](#)[Get started](#)

As with the categorical data type, the numerical data type shows some extra information in its detail area. Noteworthy here are the buttons on top of the graph. These buttons change how many “bins” are shown in the graph. You can select the following:

- Auto
- 5
- 15
- 30

**Note:** to get to these buttons, you need to “lock in place” the current feature by clicking on it. The feature then has a RED OUTLINE to show it is locked in place



[Open in app](#)[Get started](#)

## Text data

8 Ticket									
VALUES:	891 (100%)	418 (100%)	7	1%	1	0%	1601		
MISSING:	---	---	7	1%	-	-	347082		
DISTINCT:	681 (76%)	363 (87%)	7	1%	4	1%	CA. 2343		
			6	1%	2	0%	CA 2144		
			6	1%	-	-	347088		
			6	1%	1	0%	3101295		
			5	1%	1	0%	382652		
			847	95%	409	98%	(Other)		

For now, anything that the system does not consider numerical or categorical will be deemed as “text”. Text features currently only show count (percentage) as stats.

## FeatureConfig: forcing data types, skipping columns

In many cases, there are “label” columns that you may not want to analyze (although target analysis can provide insights on the distribution of target values based on labeling). In other cases, you may want to force some values to be marked as categorical even though they are numerical in nature.



[Open in app](#)[Get started](#)

```
feature_config = sweetviz.FeatureConfig(skip="PassengerId",
force_cat=["Ticket"])

my_report = sweetviz.compare([train, "Train"], [test, "Test"],
"Survived", feature_config)
```

## Comparing sub-populations (e.g. Male vs Female)

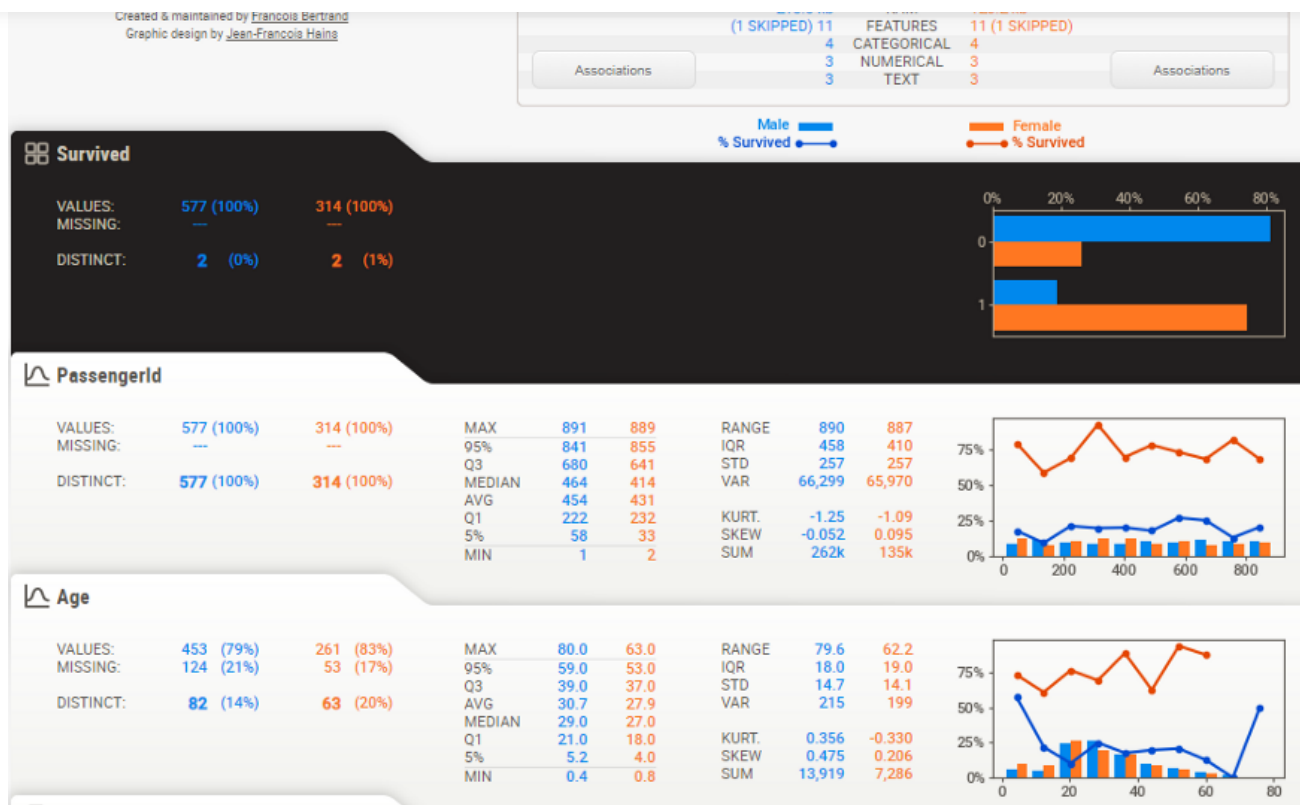
Even if you are only looking at a single dataset, it can be very useful to study the characteristics of different subpopulations within that dataset. To do so, Sweetviz provides the `compare_intra()` function. To use it, you provide a boolean test that splits the population (here we try `train["Sex"] == 'male'` , to get a sense of the different gender populations), and give a name to each subpopulation. For example:

```
my_report = sweetviz.compare_intra(train, train["Sex"] == 'male',
["Male", "Female"], 'Survived')

my_report.show_html() # Not providing a filename will default to
SWEETVIZ_REPORT.html
```

Yields the following analysis: (for this screenshot I used `feature_config` to skip showing the analysis of the “Sex” feature, as it is redundant)



[Open in app](#)[Get started](#)

Note that the target value (“Survived” in this case) is now plotted as separate lines, one for each compared dataset (e.g. male in blue, female in orange).

## Putting it all together

EDA is a fluid, artistic process that must be uniquely adapted to each set of data and situation. However, a tool like Sweetviz can help kickstart the process and get rid of a lot of the initial minutiae of characterizing datasets to provide insights right off the bat. Let’s go through all the features for the Titanic dataset to see what that could look like.

### Individual fields

#### PassengerId



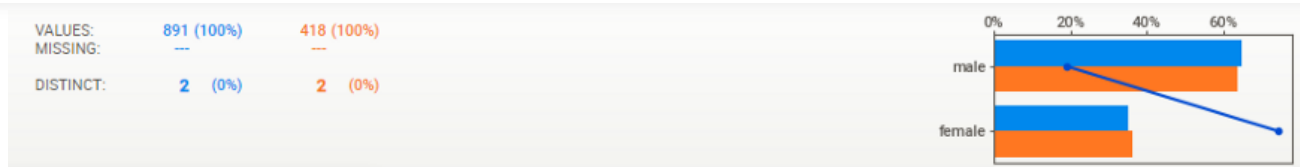
- The distribution of ID’s and survivability is even and ordered as you would hope/expect, so no surprises here.





Open in app

Get started



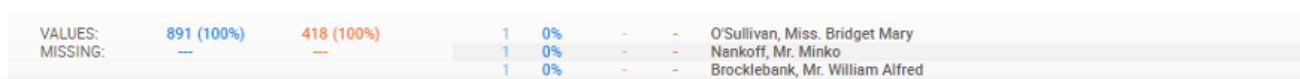
- About twice as many males as females, but...
- Females were much more likely to survive than males
- Looking at the correlations, Sex is correlated with Fare which is and isn't surprising...
- Similar distribution between Train and Test
- No missing data

## Age



- 20% missing data, consistent missing data and distribution between Train and Test
- Young-adult-centric population, but ages 0–70 well-represented
- Surprisingly evenly distributed survivability, except for a spike at the youngest age
- Using 30 bins in the histogram in the detail window, you can see that this survivability spike is really for the youngest (about  $\leq 5$  years old), as at about 10 years old survivability is really low.
- Age seems related to Siblings, Pclass and Fare, and a bit more surprisingly to Embarked

## Name



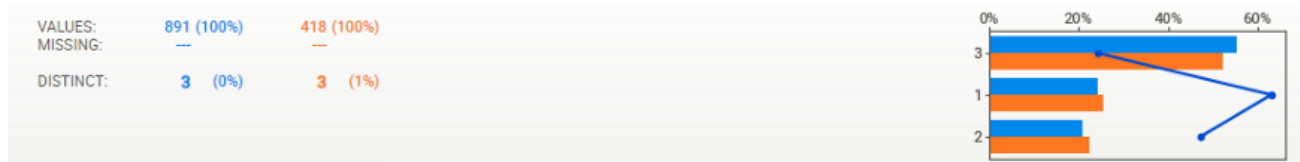


Open in app

Get started

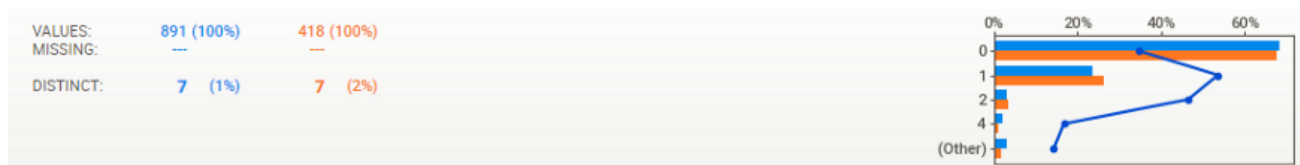
- All names are distinct, which is not surprising

## Pclass



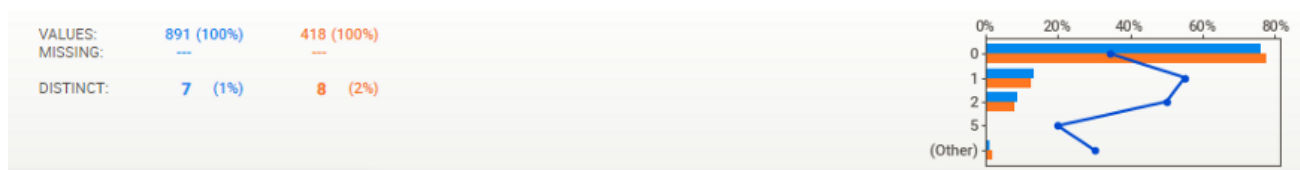
- Survivability closely follows class (first class most likely to survive, third class least likely)
- Similar distribution between Train and Test
- No missing data

## SibSp



- There seems to be a survival spike at 1 and to some degree 2, but (looking at the detail pane not shown here) there is a sharp drop-off at 3 and greater. Large families couldn't make it or perhaps were poorer?
- Similar distribution between Train and Test
- No missing data

## Parch



- Similar distribution between Train and Test



[Open in app](#)[Get started](#)

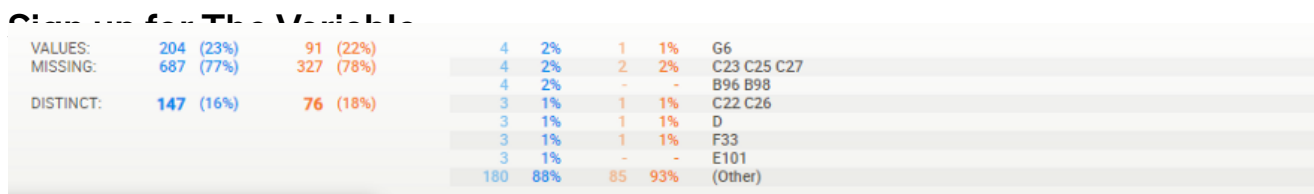
- ~80% distinct values, so about 1 in 5 shared tickets on average
- The highest frequency ticket was 7, which is generally consistent with the maximum number of siblings (8)
- No missing data, data seems pretty clean

## Fare



- As expected, and as with Pclass, the higher fares survived better (although sample size gets pretty thin at higher levels)
- A Correlation Ratio of 0.26 for “Survived” is relatively high so it would tend to support this theory
- About 30% distinct values feels a bit high as you would expect fewer set prices but looks like there is a lot of granularity so that’s ok
- Only 1 missing record in the Test set, data pretty consistent between Train and Test

## Cabin



By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.

- A lot of missing data (up to 78%), but consistent between Train and Test
- Maximum frequency is 4, which would make sense to have 4 people maximum in a cabin

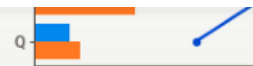
## Embarked





Open in app

Get started



- 3 distinct values (S, C, Q)
- Only 2 missing rows, in Train data. Data seems pretty consistent between Train and Test
- Survivability somewhat higher at C; could this be a location with richer people?
- Either way, “Embarked” shows a Uncertainty Coefficient of only 0.03 for “Survived”, so it may not be very significant

### General analysis

- Overall, most data is present and seems consistent and make sense; no major outliers or huge surprises

### Test versus Training data

- Test has about 50% fewer rows
- Train and Test are very closely matched in the distribution of missing data
- Train and Test data values are very consistent across the board

### Association/correlation analysis

- Sex, Fare and Pclass give the most information on Survived
- As expected, Fare and Pclass are highly correlated
- Age seems to tell us a good amount regarding Pclass, siblings and to some degree Fare, which would be somewhat expected. It seems to tell us a lot about “Embarked” which is bit more surprising.

### Missing data

- There is no significant missing data except for Age (~20%) and Cabin (~77%) (and an odd one here and there on other features)







[Open in app](#)

[Get started](#)

feature-generation, to get a quick overview of how new features play out. I hope you will find it as useful a tool in your own data analysis.

