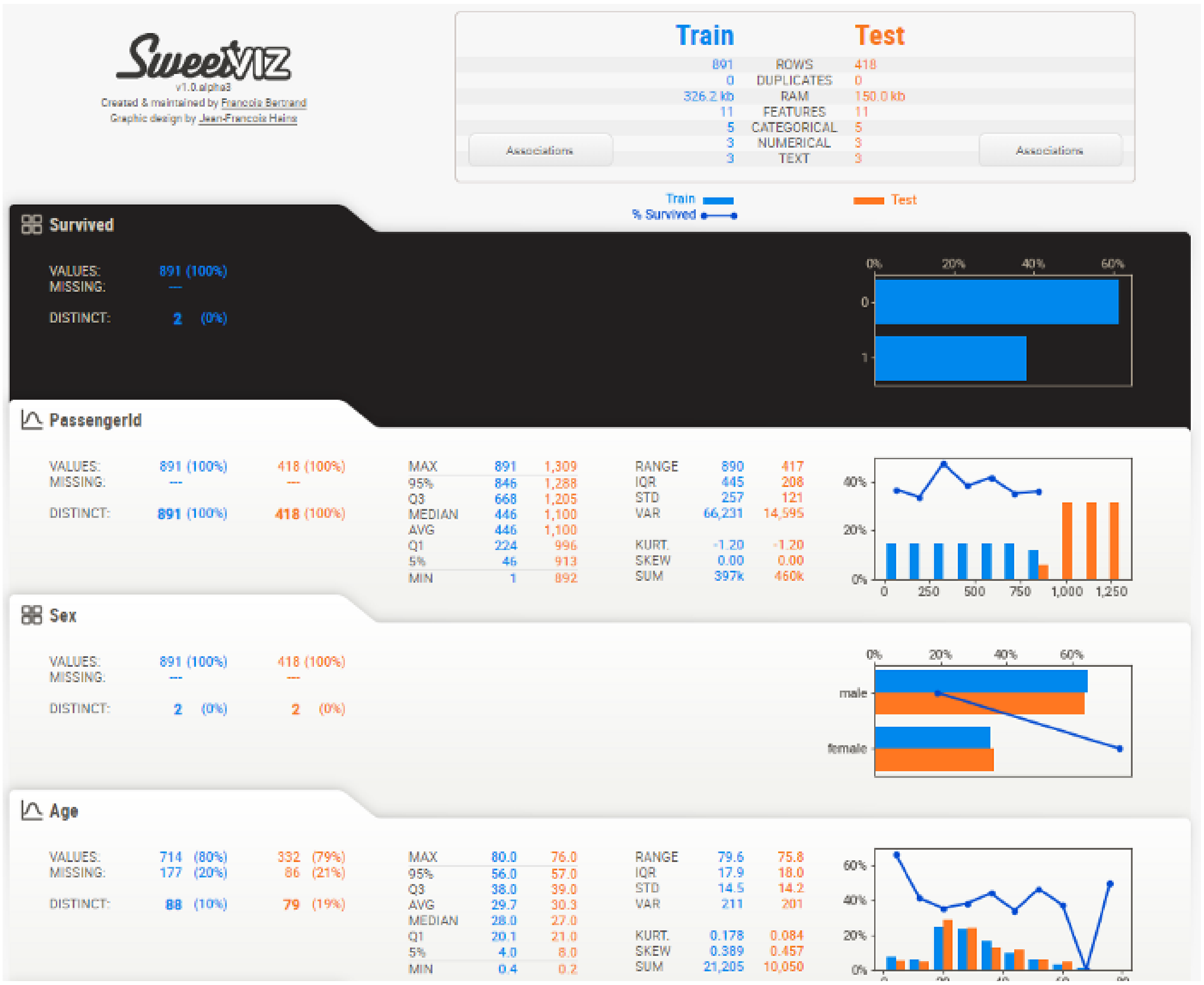
Powerful EDA (Exploratory Data Analysis) in just two lines of code using Sweetviz

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

Francois Bertrand Follow
Jun 3 · 11 min read ★



Sweetviz
v1.0.alpha3
Created & maintained by Francois Bertrand
Graphic design by Jean-Francois Hains

Train		Test
891	ROWS	418
0	DUPLICATES	0
326.2 kb	RAM	150.0 kb
11	FEATURES	11
5	CATEGORICAL	5
3	NUMERICAL	3
3	TEXT	3

Associations Associations

Train ▬ Test ▬
% Survived ●—●

Survived

	Train
VALUES:	891 (100%)
MISSING:	---
DISTINCT:	2 (0%)

PassengerId

	Train	Test
VALUES:	891 (100%)	418 (100%)
MISSING:	---	---
DISTINCT:	891 (100%)	418 (100%)

	Train	Test
MAX	891	1,309
95%	846	1,288
Q3	668	1,205
MEDIAN	446	1,100
AVG	446	1,100
Q1	224	996
5%	46	913
MIN	1	892

	Train	Test
RANGE	890	417
IQR	445	208
STD	257	121
VAR	66,231	14,595
KURT.	-1.20	-1.20
SKEW	0.00	0.00
SUM	397k	460k

Sex

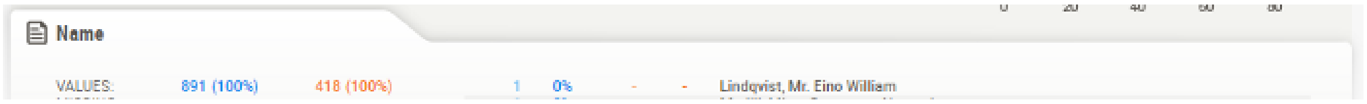
	Train	Test
VALUES:	891 (100%)	418 (100%)
MISSING:	---	---
DISTINCT:	2 (0%)	2 (0%)

Age

	Train	Test
VALUES:	714 (80%)	332 (79%)
MISSING:	177 (20%)	86 (21%)
DISTINCT:	88 (10%)	79 (19%)

	Train	Test
MAX	80.0	76.0
95%	56.0	57.0
Q3	38.0	39.0
AVG	29.7	30.3
MEDIAN	28.0	27.0
Q1	20.1	21.0
5%	4.0	8.0
MIN	0.4	0.2

	Train	Test
RANGE	79.6	75.8
IQR	17.9	18.0
STD	14.5	14.2
VAR	211	201
KURT.	0.178	0.084
SKEW	0.389	0.457
SUM	21,205	10,050



Sweetviz report created using only two lines of code

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 packs a powerful punch; in addition to creating insightful and beautiful visualizations with just two lines of code, it provides analysis that would take a lot more time to generate manually, including some that no other library provides so quickly such as:

- **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")
```

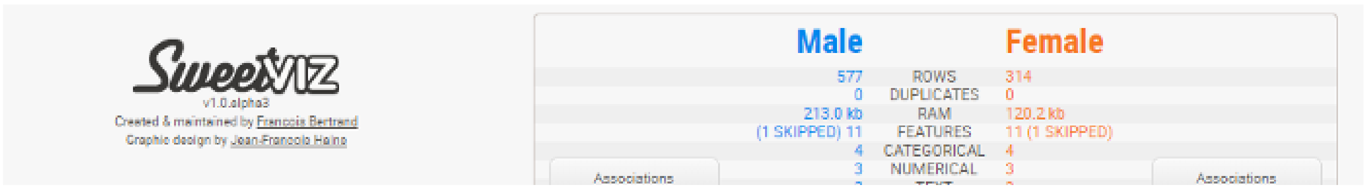
We now have 2 dataframes (train and test), and we would like to analyze the target value "Survived". I want to point out in this case we know the name of the target column in advance, but it is always optional to specify a target column. We can generate a report with this line of code:

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

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

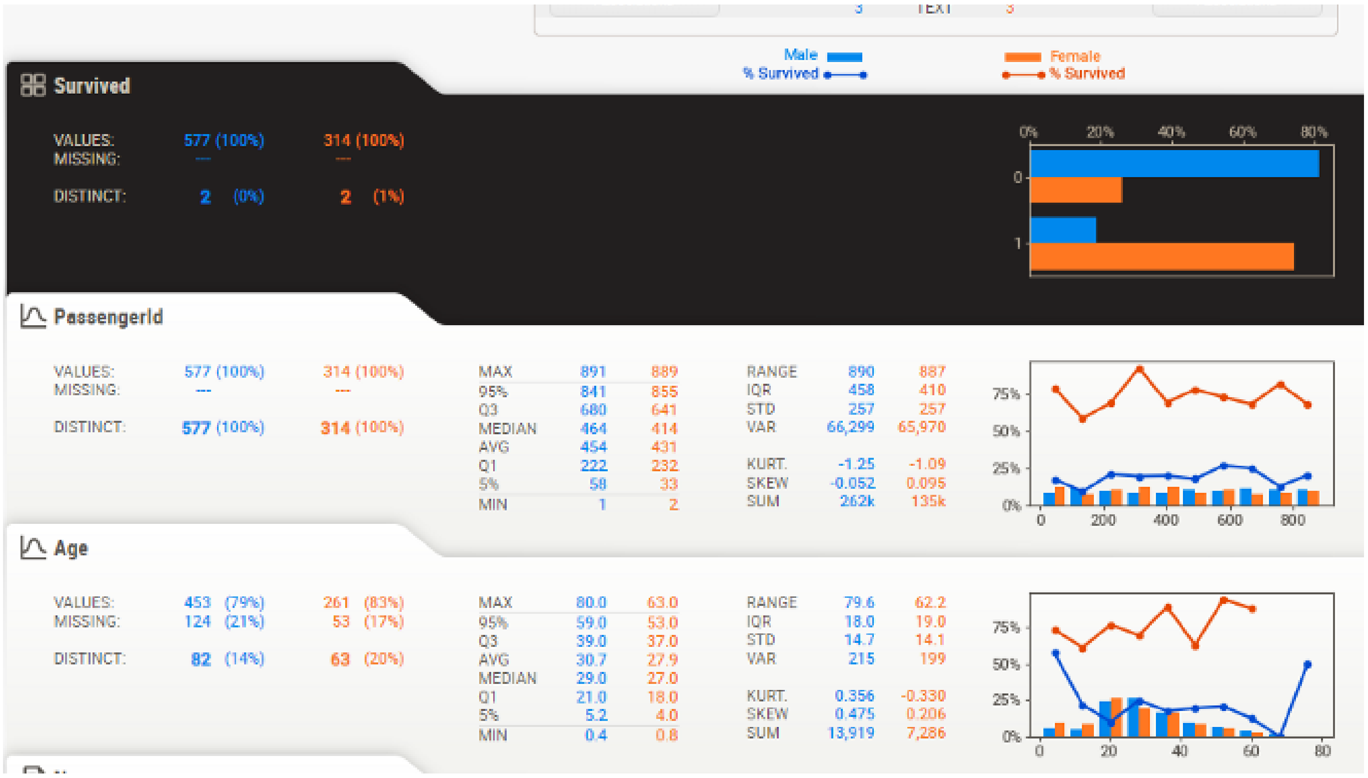
```
my_report.show_html("Report.html") # Not providing a filename will
default to SWEETVIZ_REPORT.html
```

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



Sweetviz
v1.0.alpha3
Created & maintained by Francois Bertrand
Graphic design by Jean-Francois Hains

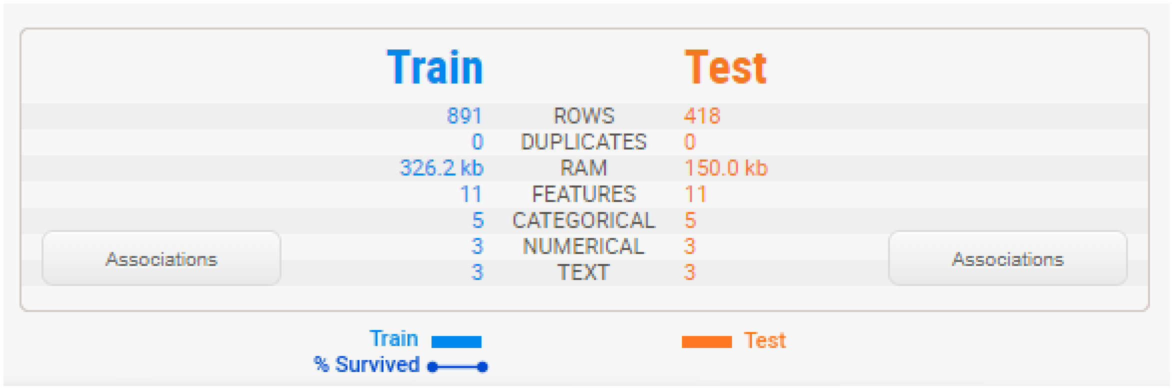
```
              Male                   Female
               577     ROWS          314
                 0     DUPLICATES    0
          213.0 kb     RAM           120.2 kb
    (1 SKIPPED) 11     FEATURES      11 (1 SKIPPED)
                 4     CATEGORICAL   4
                 3     NUMERICAL     3
```



```
Survived

VALUES:      577 (100%)     314 (100%)
MISSING:     ---            ---

DISTINCT:    2   (0%)       2   (1%)

PassengerId

VALUES:      577 (100%)     314 (100%)    MAX       891     889     RANGE    890      887
MISSING:     ---            ---           95%       841     855     IQR      458      410
                                          Q3        680     641     STD      257      257
DISTINCT:    577 (100%)     314 (100%)    MEDIAN    464     414     VAR      66,299   65,970
                                          AVG       454     431
                                          Q1        222     232     KURT.    -1.25    -1.09
                                          5%        58      33      SKEW     -0.052   0.095
                                          MIN       1       2       SUM      262k     135k

Age

VALUES:      453  (79%)     261  (83%)    MAX       80.0    63.0    RANGE    79.6     62.2
MISSING:     124  (21%)     53   (17%)    95%       59.0    53.0    IQR      18.0     19.0
                                          Q3        39.0    37.0    STD      14.7     14.1
DISTINCT:    82   (14%)     63   (20%)    AVG       30.7    27.9    VAR      215      199
                                          MEDIAN    29.0    27.0
                                          Q1        21.0    18.0    KURT.    0.356    -0.330
                                          5%        5.2     4.0     SKEW     0.475    0.206
                                          MIN       0.4     0.8     SUM      13,919   7,286
```

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

Summary display



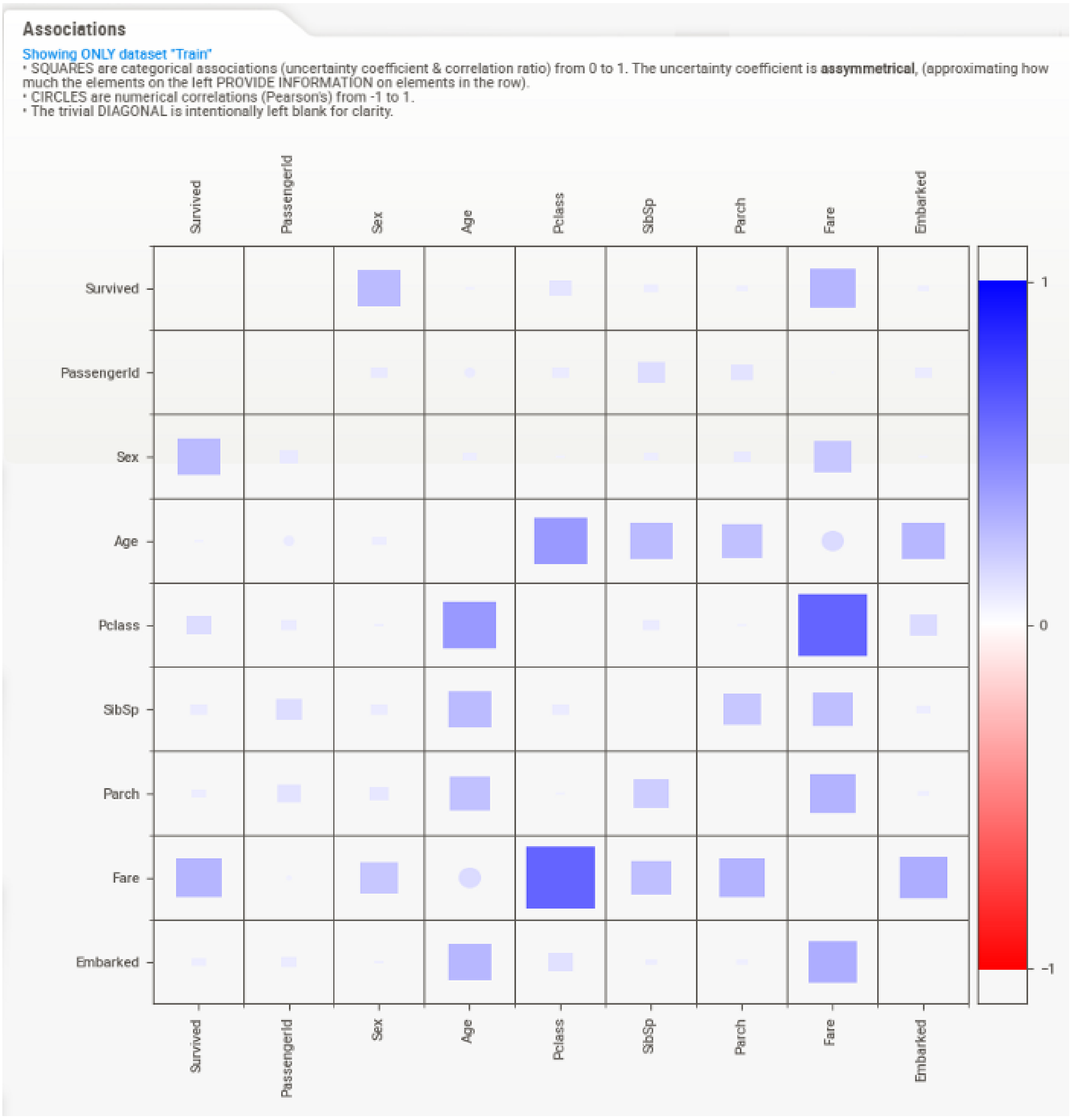
Train		Test
891	ROWS	418
0	DUPLICATES	0
326.2 kb	RAM	150.0 kb
11	FEATURES	11
5	CATEGORICAL	5
3	NUMERICAL	3
3	TEXT	3

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:



Associations

Showing ONLY dataset "Train"
• SQUARES are categorical associations (uncertainty coefficient & correlation ratio) from 0 to 1. The uncertainty coefficient is **assymmetrical**, (approximating how much the elements on the left PROVIDE INFORMATION on elements in the row).
• CIRCLES are numerical correlations (Pearson's) from -1 to 1.
• The trivial DIAGONAL is intentionally left blank for clarity.

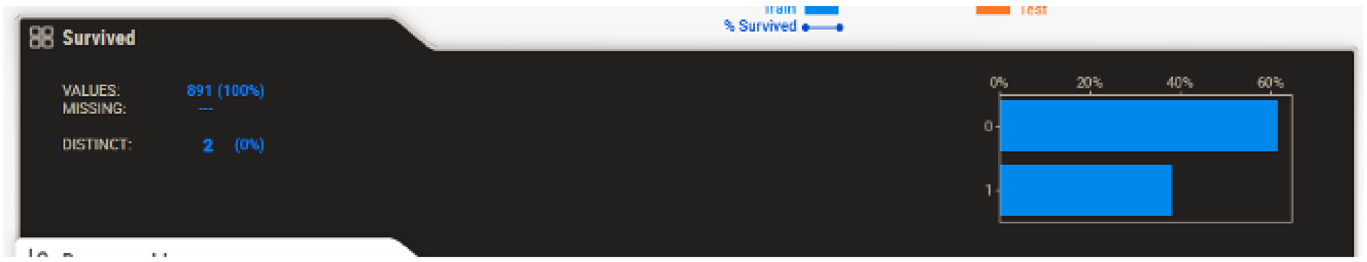
This graph is a composite of the visuals from Drazen Zaric: Better Heatmaps and Correlation Matrix Plots in Python and concepts from Shaked Zychlinski: The Search for Categorical Correlation.

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: categorical-categorical associations (provided by the uncertainty coefficient) are ASSYMMETRICAL, meaning that each row represents how much the row title (on the left) gives information on each column. For example, "Sex", "Pclass" and "Fare" are the elements that give the most information on "Survived". For the Titanic dataset, this 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



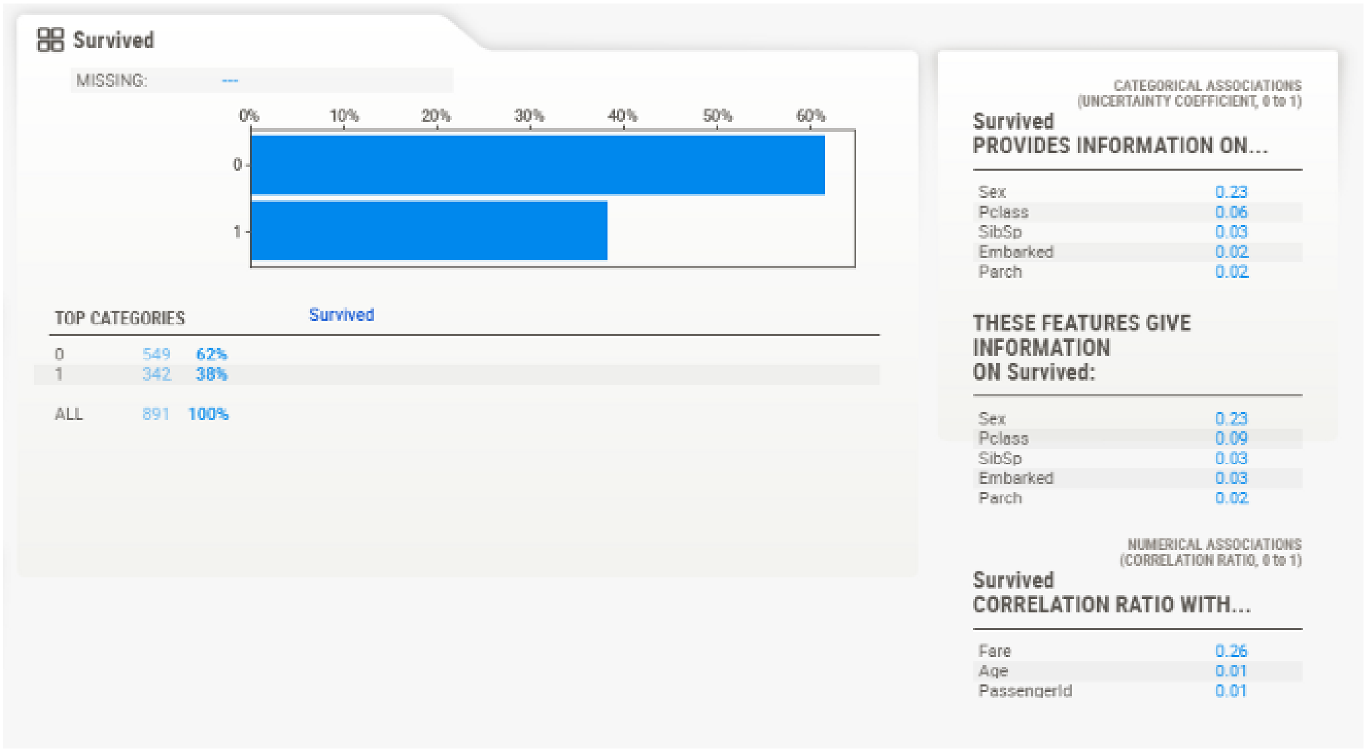
When a target variable is specified, it will show up first, in a special black box.

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:



Survived

MISSING: ---

TOP CATEGORIES	Survived	
0	549	62%
1	342	38%
ALL	891	100%

CATEGORICAL ASSOCIATIONS (UNCERTAINTY COEFFICIENT, 0 to 1)

Survived PROVIDES INFORMATION ON...

Sex	0.23
Pclass	0.06
SibSp	0.03
Embarked	0.02
Parch	0.02

THESE FEATURES GIVE INFORMATION ON Survived:

Sex	0.23
Pclass	0.09
SibSp	0.03
Embarked	0.03
Parch	0.02

NUMERICAL ASSOCIATIONS (CORRELATION RATIO, 0 to 1)

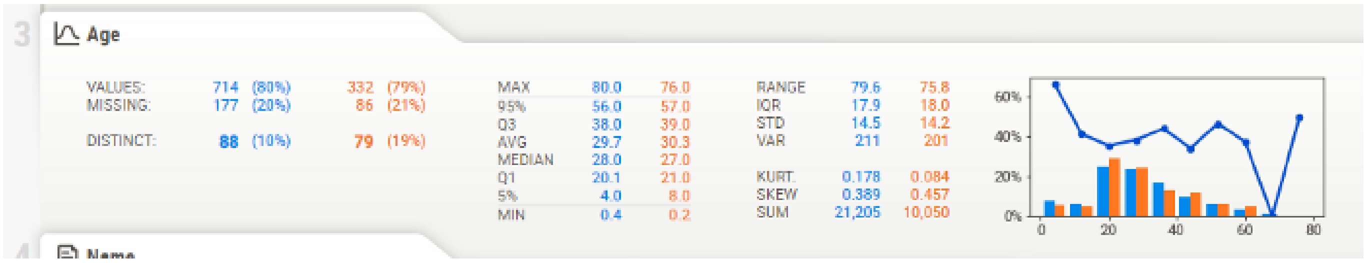
Survived CORRELATION RATIO WITH...

Fare	0.26
Age	0.01
PassengerId	0.01

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



3 Age

VALUES:	714	(80%)	332	(79%)
MISSING:	177	(20%)	86	(21%)
DISTINCT:	88	(10%)	79	(19%)

MAX	80.0	76.0
95%	56.0	57.0
Q3	38.0	39.0
AVG	29.7	30.3
MEDIAN	28.0	27.0
Q1	20.1	21.0
5%	4.0	8.0
MIN	0.4	0.2

RANGE	79.6	75.8
IQR	17.9	18.0
STD	14.5	14.2
VAR	211	201
KURT.	0.178	0.084
SKEW	0.389	0.457
SUM	21,205	10,050

4 Name

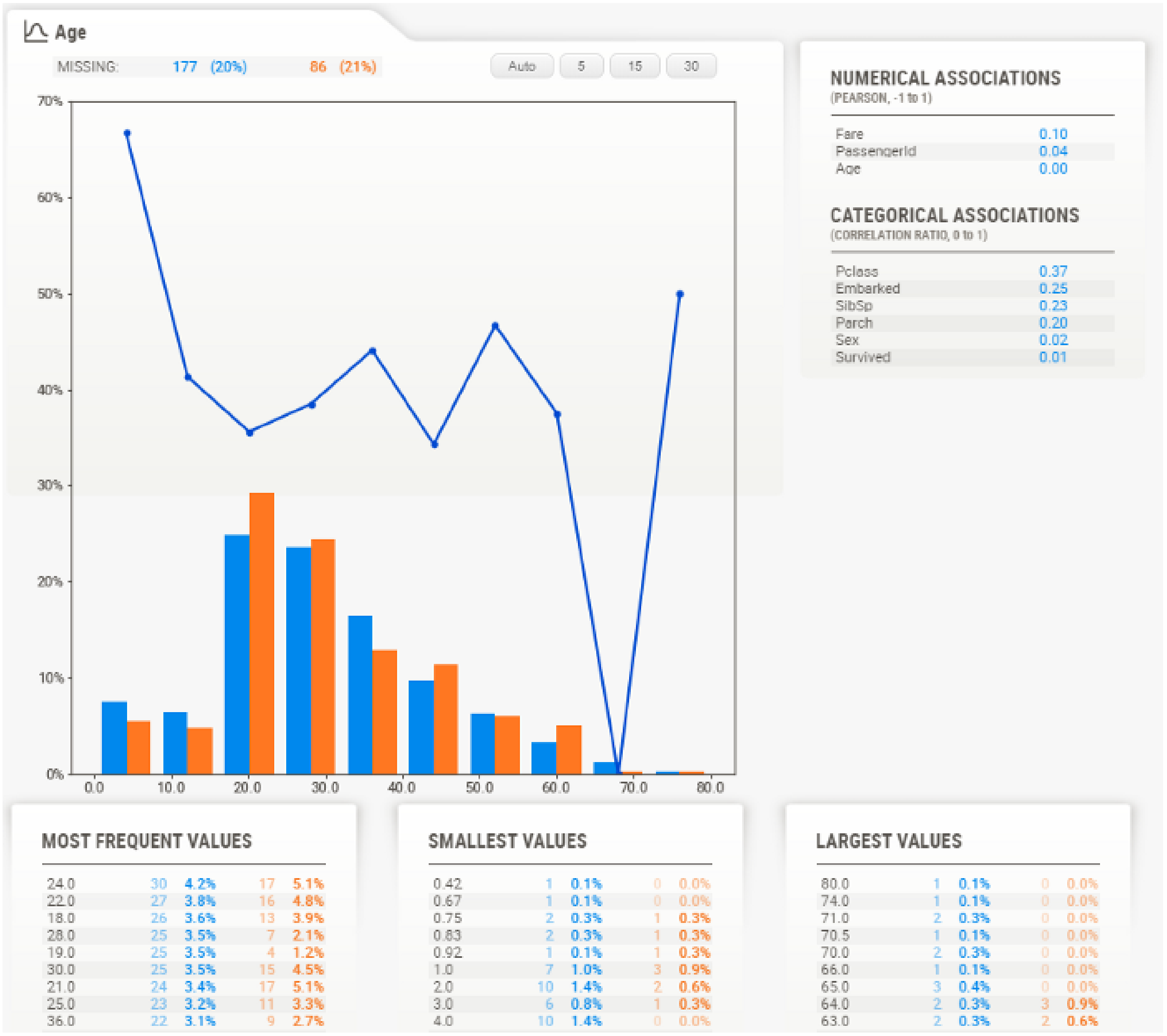
Numerical data shows more information on its summary. Here, we can see that in this case about 20% of data is missing (21% in the test data, which is very consistent).

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.

Detail area (numerical)



NUMERICAL ASSOCIATIONS
(PEARSON, -1 to 1)

Fare	0.10
PassengerId	0.04
Age	0.00

CATEGORICAL ASSOCIATIONS
(CORRELATION RATIO, 0 to 1)

Pclass	0.37
Embarked	0.25
SibSp	0.23
Parch	0.20
Sex	0.02
Survived	0.01

MOST FREQUENT VALUES

24.0	30	4.2%	17	5.1%
22.0	27	3.8%	16	4.8%
18.0	26	3.6%	13	3.9%
28.0	25	3.5%	7	2.1%
19.0	25	3.5%	4	1.2%
30.0	25	3.5%	15	4.5%
21.0	24	3.4%	17	5.1%
25.0	23	3.2%	11	3.3%
36.0	22	3.1%	9	2.7%

SMALLEST VALUES

0.42	1	0.1%	0	0.0%
0.67	1	0.1%	0	0.0%
0.75	2	0.3%	1	0.3%
0.83	2	0.3%	1	0.3%
0.92	1	0.1%	1	0.3%
1.0	7	1.0%	3	0.9%
2.0	10	1.4%	2	0.6%
3.0	6	0.8%	1	0.3%
4.0	10	1.4%	0	0.0%

LARGEST VALUES

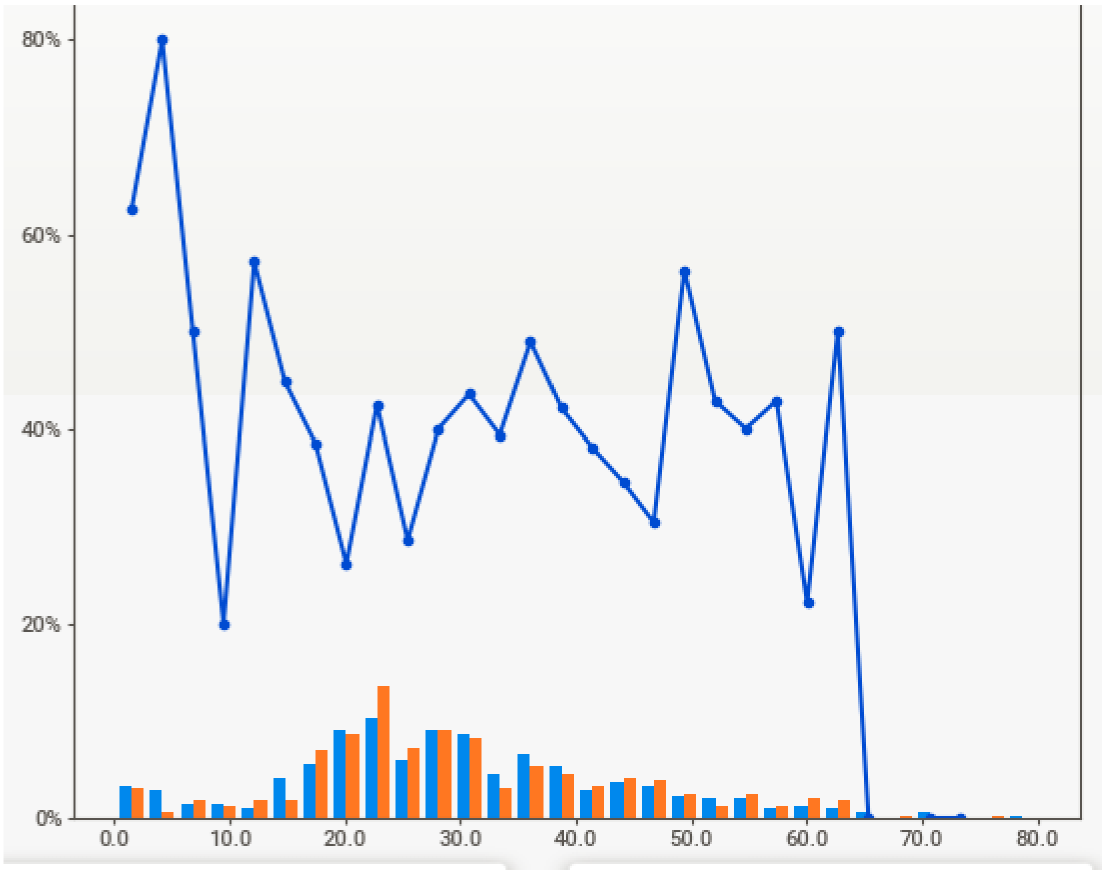
80.0	1	0.1%	0	0.0%
74.0	1	0.1%	0	0.0%
71.0	2	0.3%	0	0.0%
70.5	1	0.1%	0	0.0%
70.0	2	0.3%	0	0.0%
66.0	1	0.1%	0	0.0%
65.0	3	0.4%	0	0.0%
64.0	2	0.3%	3	0.9%
63.0	2	0.3%	2	0.6%

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:

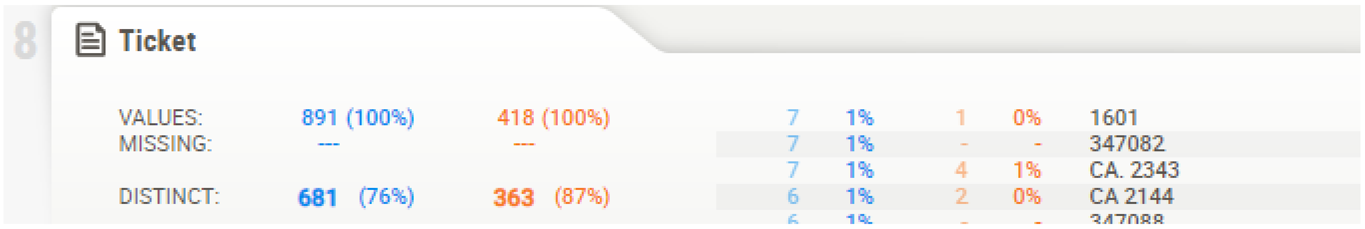
- Auto
- 5
- 15
- 30

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 and you can access the detail area.

For example selecting "30" yields a much more granular graph:

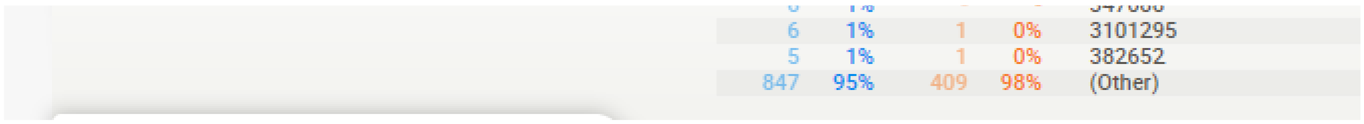


Text data



8 Ticket

VALUES:	891 (100%)	418 (100%)	7	1%	1	0%	1601
MISSING:	---	---	7	1%	-	-	347082
			7	1%	4	1%	CA. 2343
DISTINCT:	681 (76%)	363 (87%)	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.

To do all this, simply create a FeatureConfig object and pass it in to the analyze/compare function. You can specify either a string or a list to kwargs `skip`, `force_cat` and `force_text`:

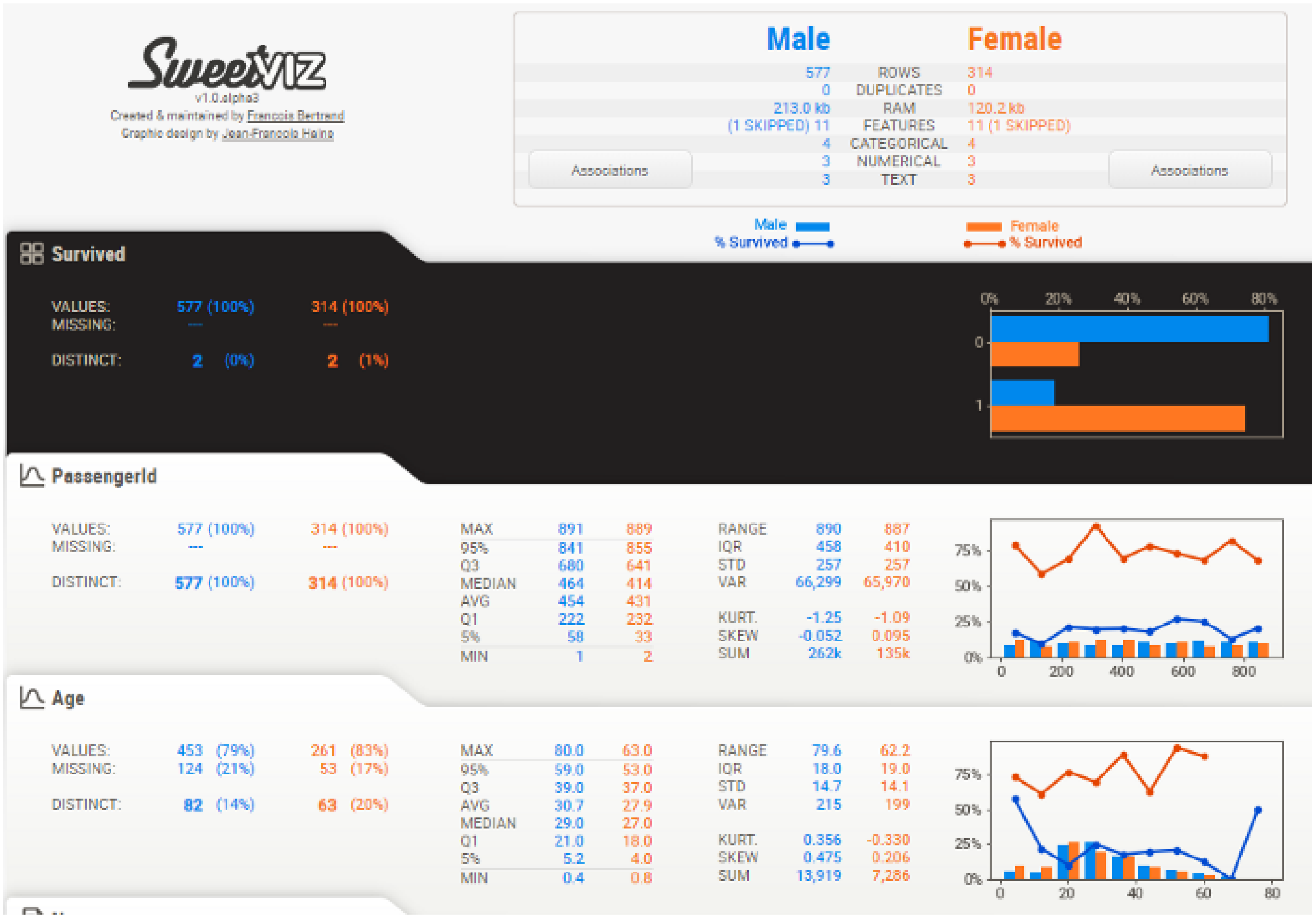
```
feature_config = sweetviz.FeatureConfig(skip="PassengerId",
force_cat=["Ticket"])

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)



Sweetviz v1.0.alpha3
Created & maintained by Francois Bertrand
Graphic design by Jean-Francois Hains

Male		Female
577	ROWS	314
0	DUPLICATES	0
213.0 kb	RAM	120.2 kb
(1 SKIPPED) 11	FEATURES	11 (1 SKIPPED)
4	CATEGORICAL	4
3	NUMERICAL	3
3	TEXT	3

Survived

	Male	Female
VALUES:	577 (100%)	314 (100%)
MISSING:	---	---
DISTINCT:	2 (0%)	2 (1%)

PassengerId

	Male	Female
VALUES:	577 (100%)	314 (100%)
MISSING:	---	---
DISTINCT:	577 (100%)	314 (100%)

	Male	Female
MAX	891	889
95%	841	855
Q3	680	641
MEDIAN	464	414
AVG	454	431
Q1	222	232
5%	58	33
MIN	1	2

	Male	Female
RANGE	890	887
IQR	458	410
STD	257	257
VAR	66,299	65,970
KURT.	-1.25	-1.09
SKEW	-0.052	0.095
SUM	262k	135k

Age

	Male	Female
VALUES:	453 (79%)	261 (83%)
MISSING:	124 (21%)	53 (17%)
DISTINCT:	82 (14%)	63 (20%)

	Male	Female
MAX	80.0	63.0
95%	59.0	53.0
Q3	39.0	37.0
AVG	30.7	27.9
MEDIAN	29.0	27.0
Q1	21.0	18.0
5%	5.2	4.0
MIN	0.4	0.8

	Male	Female
RANGE	79.6	62.2
IQR	18.0	19.0
STD	14.7	14.1
VAR	215	199
KURT.	0.356	-0.330
SKEW	0.475	0.206
SUM	13,919	7,286

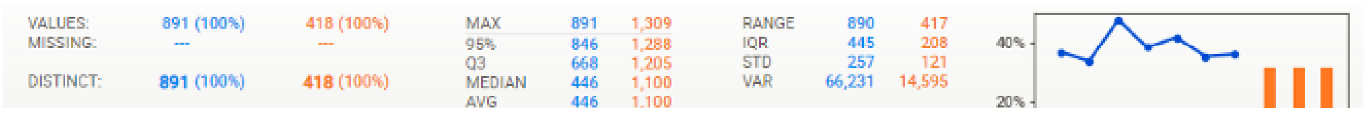
Note that the target value ("Survived" in this case) is now plotted as separate lines, one for each compared dataset (e.g. male in blue, female in orange).

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

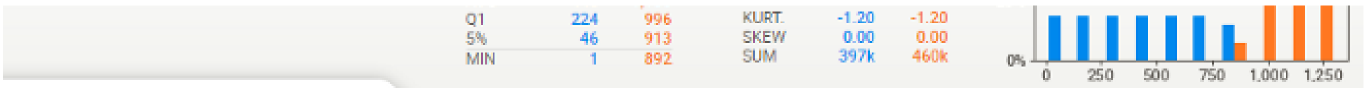
PassengerId



VALUES:	891 (100%)	418 (100%)
MISSING:	---	---
DISTINCT:	891 (100%)	418 (100%)

MAX	891	1,309
95%	846	1,288
Q3	668	1,205
MEDIAN	446	1,100
AVG	446	1,100

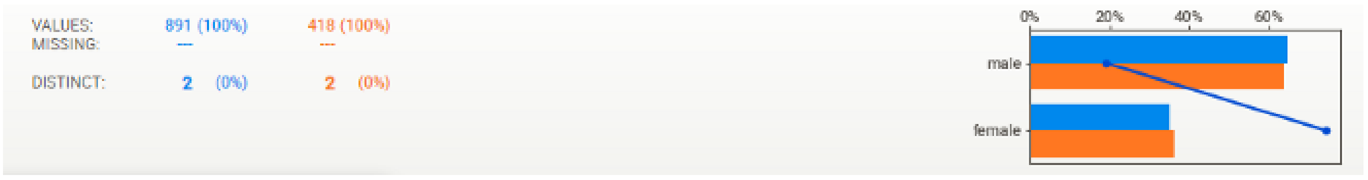
RANGE	890	417
IQR	445	208
STD	257	121
VAR	66,231	14,595



```
Q1      224   996     KURT.  -1.20  -1.20
5%       46   913     SKEW    0.00   0.00
MIN       1   892     SUM     397k   460k
```

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

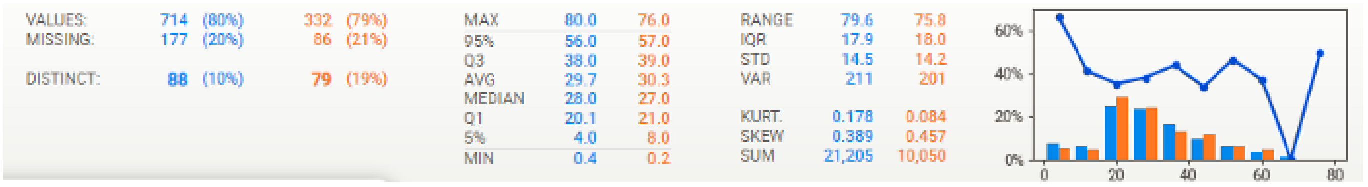
Sex



```
VALUES:     891 (100%)    418 (100%)
MISSING:    ---           ---

DISTINCT:     2   (0%)      2   (0%)
```

- 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



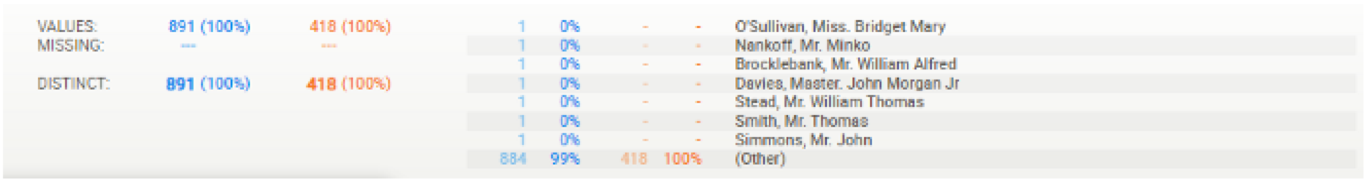
```
VALUES:    714  (80%)    332  (79%)     MAX     80.0   76.0     RANGE    79.6    75.8
MISSING:   177  (20%)     86  (21%)     95%     56.0   57.0     IQR      17.9    18.0
                                        Q3      38.0   39.0     STD      14.5    14.2
DISTINCT:   88  (10%)     79  (19%)     AVG     29.7   30.3     VAR       211     201
                                        MEDIAN  28.0   27.0
                                        Q1      20.1   21.0     KURT.   0.178   0.084
                                        5%       4.0    8.0     SKEW    0.389   0.457
                                        MIN      0.4    0.2     SUM    21,205  10,050
```

- 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 <= 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

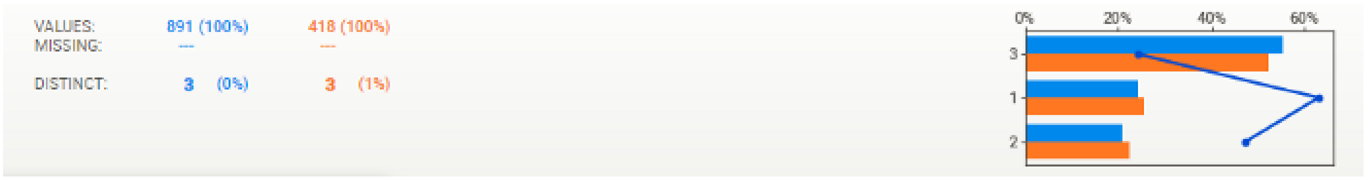


VALUES:	891 (100%)	418 (100%)
MISSING:	---	---
DISTINCT:	891 (100%)	418 (100%)

1	0%	-	-	O'Sullivan, Miss. Bridget Mary
1	0%	-	-	Nankoff, Mr. Minko
1	0%	-	-	Brocklebank, Mr. William Alfred
1	0%	-	-	Davies, Master. John Morgan Jr
1	0%	-	-	Stead, Mr. William Thomas
1	0%	-	-	Smith, Mr. Thomas
1	0%	-	-	Simmons, Mr. John
884	99%	418	100%	(Other)

- No missing data, data seems pretty clean
- All names are distinct, which is not surprising

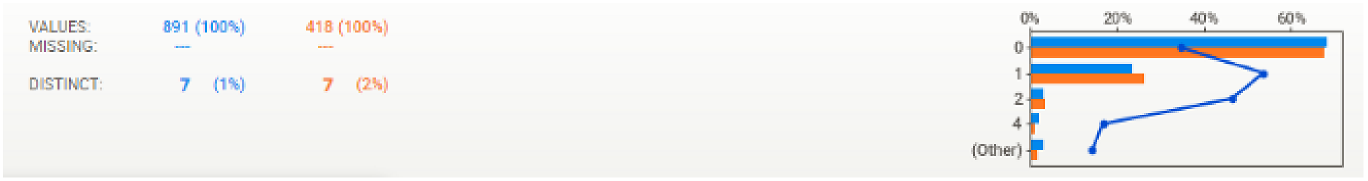
Pclass



VALUES:	891 (100%)	418 (100%)
MISSING:	---	---
DISTINCT:	3 (0%)	3 (1%)

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

SibSp

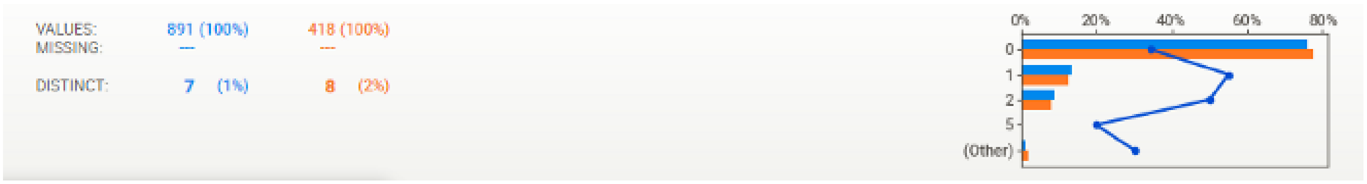


VALUES:	891 (100%)	418 (100%)
MISSING:	---	---
DISTINCT:	7 (1%)	7 (2%)

- 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



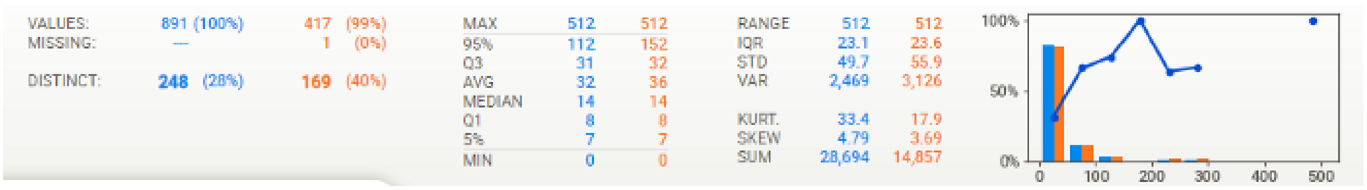
VALUES:	891 (100%)	418 (100%)
MISSING:	---	---
DISTINCT:	7 (1%)	8 (2%)

- Similar distribution between Train and Test
- No missing data

Ticket

- ~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



VALUES:	891 (100%)	417 (99%)
MISSING:	---	1 (0%)
DISTINCT:	248 (28%)	169 (40%)

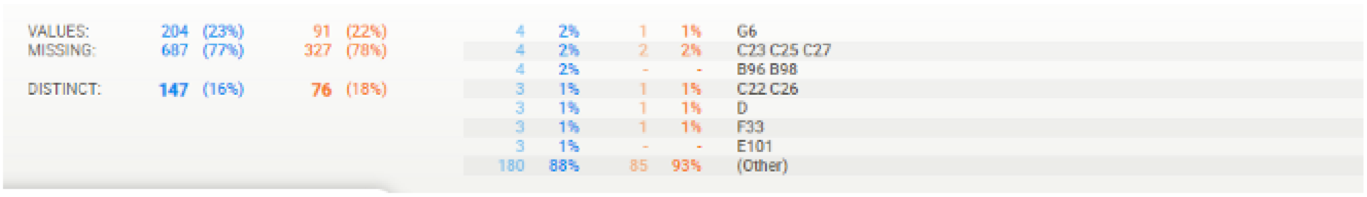
MAX	512	512
95%	112	152
Q3	31	32
AVG	32	36
MEDIAN	14	14
Q1	8	8
5%	7	7
MIN	0	0

RANGE	512	512
IQR	23.1	23.6
STD	49.7	55.9
VAR	2,469	3,126
KURT.	33.4	17.9
SKEW	4.79	3.69
SUM	28,694	14,857

- 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 recordu in the Test set, data pretty consistent between Train and Test

Cabin

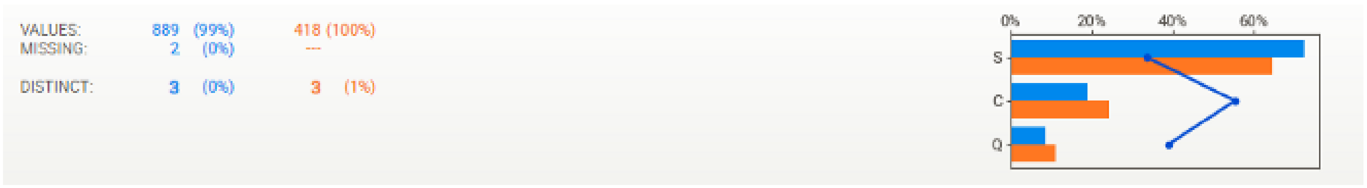


VALUES:	204	(23%)	91	(22%)
MISSING:	687	(77%)	327	(78%)
DISTINCT:	147	(16%)	76	(18%)

4	2%	1	1%	G6
4	2%	2	2%	C23 C25 C27
4	2%	-	-	B96 B98
3	1%	1	1%	C22 C26
3	1%	1	1%	D
3	1%	1	1%	F33
3	1%	-	-	E101
180	88%	85	93%	(Other)

- 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



VALUES:	889	(99%)	418	(100%)
MISSING:	2	(0%)	---	
DISTINCT:	3	(0%)	3	(1%)

- 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 a 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)

Conclusion

All this information from just two lines of code!

Using Sweetviz easily gives me a significant jump-start when I start looking at a new dataset. It's worth pointing out I also find it useful later in the analysis process, for example during 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.

Eda Visualization Exploratory Data Analysis Pandas Towards Data Science