intedact

Release 0.1.0

Matt Boggess

Dec 26, 2022

ABOUT

CHAPTER

ONE

INTEDACT: INTERACTIVE EDA

PyPI Version License

Interactive EDA for pandas DataFrames directly in your Jupyter notebook. intedact makes common, standardized EDA visual summaries available in an interactive manner with one function call. Using ipywidgets, you can quickly cycle through different variables or combinations of variables and produce useful visual summaries when exploring the dataset. Each summary will have additional plot parameters you can tweak to adjust the visualizations to work for your dataset.

Full documentation at intedact.readthedocs.io

CHAPTER

TWO

GETTING STARTED

2.1 Installation

Install via pip:

```
pip install intedact
```

Download the following nltk resources for the ngram text summaries.

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

2.2 Univariate EDA

Univariate EDA refers to the process of visualizing and summarizing a single variable.

For interactive univariate EDA simply import the `univariate_eda_interact` function in a jupyter notebook and pass in a pandas dataframe:

```
from intedact import univariate_eda_interact
univarate_eda_interact(
    data, notes_file="optional_file_to_save_notes_to.json"
)
```

At the top level, one selects the column and the summary type for that column to display. To explore the full dataset, just toggle through each of the column names. Current supported summary types:

- categorical: Summarize a categorical or low cardinality numerical column
- numeric: Summarize a high cardinality numerical column
- datetime: Summarize a datetime column
- text: Summarize a free form text column
- collection: Summarize a column with collections of values (i.e. lists, tuples, sets, etc.)
- url: Summarize a column containing urls

For each column, one can then adjust parameters for the given summary type to fit your particular dataset. These summaries try to automatically set good default parameters, but sometimes you need to make adjustments to get the full picture.

See the documentation for examples of how to statically call the individual univariate summary functions.

2.3 Bivariate EDA

Bivariate EDA refers to the process of visualizing and summarizing a pair of variables.

Like with univariate EDA, simply import the `bivariate_eda_interact` function in a jupyter notebook and pass in a dataframe:

```
from intedact import bivariate_eda_interact
bivarate_eda_interact(
    data, notes_file="optional_file_to_save_notes_to.json"
)
```

At the top level, one selects a pair of columns to display (one as the independent and the second as the dependent). Current supported summary types:

- categorical-categorical: Summarize a pair of categorical columns
- numeric-categorical: Summarize an independent numeric variable against a dependent categorical variable
- categorical-numeric: Summarize an independent categorical variable against a dependent numeric variable
- numeric-numeric: Summarize a pair of numeric columns

CHAPTER

THREE

DESIGN PHILOSOPHY

The motivation for intedact comes from the following observations:

1. There is a standard set of visualizations that should be always applied to different individual and combinations of variables depending on their type when performing EDA. For example, it is always good to visualize the distribution of a numerical variable using a histogram. intedact's goal is to save you from having to constantly copy-paste this code across columns, projects, etc.
2. These visualizations often need some degree of adjustment to get the information you need. For example, really skewed variables with outliers might need some outlier filtering and/or a log transform to actually be able to visualize the histogram properly. intedact's goal is to give you additional control over the visualization with interactive widgets that you can repeatedly adjust until you get the visualization you need.

Given the above, intedact tries to produce visualizations that give you the visual understanding you are seeking for 95% of cases when you pass in the defaults. For the other 5%, we give you additional parameters you can tweak via the widgets so you can still get the insights you need without having to leave the interface.

intedact is not a single click EDA summary generation tool. Many of those exist and we recommend pairing them with intedact (pandas-profiling is a great one for example). Where these fall short, is they don't focus on the visualizations and give you the power to adjust them to your dataset when the defaults don't suffice. Use intedact when you want to dig deeper and really visually understand a variable or the relationship between variable(s).

CHAPTER

FOUR

UNIVARIATE SUMMARY EXAMPLES

4.1 Univariate URL Summary

Example of univariate eda summary for an url variable

The URL summary computes the following:

- Countplot for the invididual unique urls
- Countplot for the domains of the urls
- Countplot for the domain suffixes of the urls
- Countplot for the file types of the urls

```
import pandas as pd
import plotly

import intedact
```

Here we take a look at the source URL's for countries GDPR violations recordings.

```
data = pd.read_csv(
    "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-
↪04-21/gdpr_violations.tsv",
    sep="\t",
)

fig = intedact.url_summary(data, "source", fig_width=700)
plotly.io.show(fig)
```

Total running time of the script: (0 minutes 3.239 seconds)

4.2 Univariate Numeric Summary

Example of univariate eda summary for a numeric variable.

The numeric summary computes the following:

- A histogram
- A boxplot

```
import pandas as pd
import plotly

import intedact
```

Here we take a look at some GDPR violation prices and showcase some parameters:

- log transformation
- outlier filtering
- custom bin count

```
data = pd.read_csv(
    "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-
↪04-21/gdpr_violations.tsv",
    sep="\t",
)
fig = intedact.numeric_summary(
    data, "price", bins=20, transform="log", upper_quantile=0.95, fig_width=700
)
plotly.io.show(fig)
```

Total running time of the script: (0 minutes 0.196 seconds)

4.3 Univariate Collection Summary

Example of univariate eda summary for a collection variable (lists, tuples, sets, etc.).

The collection summary computes the following:

- Three separate countplots: - Counts for all the unique collections - Counts for all the unique entries - Counts for the number of entries in each collection

```
import pandas as pd
import plotly

import intedact
```

Here we take a look at which articles of GDPR countries violated. We first have to process the column so it is a list and not a string. One can also choose whether to sort the values (ignore order of how they're listed) and remove duplicates (only consider unique entries)

```
data = pd.read_csv(
    "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-
↪04-21/gdpr_violations.tsv",
    sep="\t",
)
data["article_violated"] = data["article_violated"].apply(lambda x: x.split("|"))

fig = intedact.collection_summary(data, "article_violated", fig_width=700)
plotly.io.show(fig)
```

Total running time of the script: (0 minutes 0.208 seconds)

4.4 Univariate Categorical Summary

Example of univariate eda summary for a categorical variable.

The categorical summary computes the following:

- A countplot with counts and percentages by level of the categorical
- A table with summary statistics

```
import pandas as pd
import plotly

import intedact
```

For our first example, we plot the name of countries who have had GDPR violations. By default, the plot will try to order and orient the columns appropriately. Here we order by descending count and the plot was flipped horizontally due to the number of levels in the variable.

```
data = pd.read_csv(
    "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-
↪04-21/gdpr_violations.tsv",
    sep="\t",
)
fig = intedact.categorical_summary(data, "name", fig_width=700)
plotly.io.show(fig)
```

We can do additional things such as condense extra columns into an "Other" column, add a bar for missing values, and change the sort order to sort alphabetically.

```
fig = intedact.categorical_summary(
    data,
    "name",
    include_missing=True,
    order="sorted",
    max_levels=5,
    fig_width=700,
)
plotly.io.show(fig)
```

Out:

```
No missing values for column: name
```

Total running time of the script: (0 minutes 0.222 seconds)

4.5 Univariate Text Summary

Example of univariate eda summary for a text variable

The text summary computes the following:

- Histogram of # of tokens / document
- Histogram of # of characters / document
- Boxplot of # of unique observations of each document
- Countplots for the most common unigrams, bigrams, and trigams

```
import nltk
import pandas as pd
import plotly

import intedact

nltk.download("punkt")
nltk.download("stopwords")
```

Out:

```
[nltk_data] Downloading package punkt to /Users/mboggess/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/mboggess/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

True
```

Here we take a look at the summaries for GDPR violations.

```
data = pd.read_csv(
    "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-
↪04-21/gdpr_violations.tsv",
    sep="\t",
)

fig = intedact.text_summary(data, "summary", fig_width=700)
plotly.io.show(fig)
```

By default, the summary does a lot of text cleaning: removing punctuation and stop words, lower casing. We can turn all of these off.

```
fig = intedact.text_summary(
    data,
    "summary",
    remove_stop=False,
    remove_punct=False,
    lower_case=False,
    fig_width=700,
)
plotly.io.show(fig)
```

Total running time of the script: (0 minutes 0.851 seconds)

4.6 Univariate Datetime Summary

Example of univariate eda summary for a datetime variable. Here we look at posting times for TidyTuesday tweets.

The datetime summary computes the following:

- A time seriesplot aggregated according to the *ts_freq* parameter
- Barplots showing counts by day of week, month, hour of day, day of month

```
import pandas as pd
import plotly

import intedact

data = pd.read_csv(
    "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/tidytuesday_
↪tweets/data.csv"
)
data["created_at"] = pd.to_datetime(data.created_at)
fig = intedact.datetime_summary(data, "created_at", fig_width=700)
plotly.io.show(fig)
```

By default, the summary tries to infer reasonable units for the time series. We can change these by using time unit strings for the *ts_freq* parameter.

```
fig = intedact.datetime_summary(data, "created_at", ts_freq="1 day", fig_width=700)
plotly.io.show(fig)
```

Example of changing plot type, removing trend line, and removing outliers.

```
fig = intedact.datetime_summary(
    data,
    "created_at",
    ts_type="markers",
    trend_line="none",
    upper_quantile=0.99,
    fig_width=700,
)
plotly.io.show(fig)
```

Total running time of the script: (0 minutes 4.192 seconds)

CHAPTER

FIVE

BIVARIATE SUMMARY EXAMPLES

5.1 Bivariate Numeric-Categorical Summary

Example of bivariate eda summary for a numeric independent variable and a categorical dependent variable.

The summary computes the following:

- Lineplot with fractions for each level of the categorical variable against quantiles of the numeric variable

```
import pandas as pd
import plotly

import intedact
```

Here we look at how diamond cut quality changes with carats.

```
data = pd.read_csv(
    "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/diamonds.csv"
)
fig = intedact.numeric_categorical_summary(
    data, "carat", "cut", num_intervals=5, fig_width=700
)
plotly.io.show(fig)
```

Total running time of the script: (0 minutes 0.692 seconds)

5.2 Bivariate Categorical-Numeric Summary

Example of bivariate eda summary for a categorical independent variable and a numeric dependent variable.

The summary computes the following:

- Overlapping histogram/kde plots of distributions by level
- Side by side boxplots per level

```
import pandas as pd
import plotly

import intedact
```

Here we look at how diamond price changes with cut quality

```
data = pd.read_csv(
    "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/diamonds.csv"
)
data["cut"] = pd.Categorical(
    data["cut"],
    categories=["Fair", "Good", "Very Good", "Premium", "Ideal"],
    ordered=True,
)
fig = intedact.categorical_numeric_summary(data, "cut", "price", fig_width=700)
plotly.io.show(fig)
```

Total running time of the script: (0 minutes 1.243 seconds)

5.3 Bivariate Numeric-Numeric Summary

Example of bivariate eda summary for a pair of numeric variables.

The summary computes the following:

- A scatterplot with trend line
- A 2d histogram
- Boxplots of the dependent variable against quantiles of the independent variable

```
import pandas as pd
import plotly

import intedact
```

Here we take a look at relationship between carat and price in the diamonds dataset

```
data = pd.read_csv(
    "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/diamonds.csv"
).sample(n=10000)
fig = intedact.numeric_numeric_summary(data, "carat", "price", fig_width=700)
plotly.io.show(fig)
```

By default, it is hard to see much since the distributions are very skewed with outliers. We can tweak the plot to actually visualize the distributions in more detail.

```
fig = intedact.numeric_numeric_summary(
    data,
    "carat",
    "price",
    upper_quantile1=0.98,
    hist_bins=100,
    num_intervals=10,
    opacity=0.4,
    fig_width=700,
)
plotly.io.show(fig)
```

Total running time of the script: (0 minutes 8.355 seconds)

5.4 Bivariate Categorical-Categorical Summary

Example of bivariate eda summary for a pair of categorical variables

The summary computes the following:

- Categorical heatmap with counts and percentages for each level combo
- Barplot showing distribution of column2's levels within each level of column1
- Lineplot showing distribution of column2's levels across each level of column1

```
import pandas as pd
import plotly

import intedact
```

Here we look at how diamond cut quality and clarity quality are related.

```
data = pd.read_csv(
    "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/diamonds.csv"
)
data["cut"] = pd.Categorical(
    data["cut"],
    categories=["Fair", "Good", "Very Good", "Premium", "Ideal"],
    ordered=True,
)
data["clarity"] = pd.Categorical(
    data["clarity"],
    categories=["I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF"],
    ordered=True,
)
fig = intedact.categorical_categorical_summary(
    data, "clarity", "cut", barmode="group", fig_width=700
)
plotly.io.show(fig)
```

Total running time of the script: (0 minutes 0.585 seconds)

CHAPTER

SIX

UNIVARIATE SUMMARY FUNCTIONS

Functions for creating univariate EDA summaries.

categorical_summary	Creates a univariate EDA summary for a provided categorical data column in a pandas DataFrame.
numeric_summary	Creates a univariate EDA summary for a high cardinality numeric data column in a pandas DataFrame.
datetime_summary	Creates a univariate EDA summary for a datetime data column in a pandas DataFrame.
text_summary	Creates a univariate EDA summary for a text variable column in a pandas DataFrame.
collection_summary	Creates a univariate EDA summary for a collections column in a pandas DataFrame.
url_summary	Creates a univariate EDA summary for a url column in a pandas DataFrame.

6.1 categorical_summary

categorical_summary(*data: DataFrame, column: str, fig_height: int = 600, fig_width: int = 1200, order: Union[str, List] = 'auto', max_levels: int = 20, flip_axis: Optional[bool] = None, include_missing: bool = False, display_figure: bool = False*) → Figure

Creates a univariate EDA summary for a provided categorical data column in a pandas DataFrame.

Parameters

- **data** – pandas DataFrame with data to be plotted
- **column** – column in the dataframe to plot
- **fig_width** – figure width in inches
- **fig_height** – figure height in inches
- **order** – Order in which to sort the levels of the variable for plotting:
  - **'auto'**: sorts ordinal variables by provided ordering, nominal variables by descending frequency, and numeric variables in sorted order.
  - **'descending'**: sorts in descending frequency.
  - **'ascending'**: sorts in ascending frequency.
  - **'sorted'**: sorts according to sorted order of the levels themselves.
  - **'random'**: produces a random order. Useful if there are too many levels for one plot.

Or you can pass a list of level names in directly for your own custom order.

- **max_levels** – Maximum number of levels to attempt to plot on a single plot. If exceeded, only the max_level - 1 levels will be plotted and the remainder will be grouped into an 'Other' category. size and number of levels.
- **flip_axis** – Whether to flip the plot so labels are on y axis. Useful for long level names or lots of levels. Default tries to infer based on number of levels and label_rotation value.
- **include_missing** – Whether to include missing values as an additional level in the data
- **display_figure** – Whether to display the figure in addition to returning it

6.2 numeric_summary

numeric_summary(*data: DataFrame, column: str, fig_height: int = 600, fig_width: int = 1200, bins: int = 0, transform: str = 'identity', lower_quantile: float = 0, upper_quantile: float = 1, display_figure: bool = False*) → Figure

Creates a univariate EDA summary for a high cardinality numeric data column in a pandas DataFrame.

Parameters

- **data** – pandas DataFrame to perform EDA on
- **column** – A string matching a column in the data to visualize
- **fig_height** – Height of the plot in pixels
- **fig_width** – Width of the plot in pixels
- **bins** – Number of bins to use for the histogram. Default (0) is to determines # of bins from the data
- **transform** – Transformation to apply to the data for plotting:
  - 'identity': no transformation
  - 'log': apply a logarithmic transformation (zero and negative values will be filtered out)
  - 'sqrt': apply a square root transformation
- **lower_quantile** – Lower quantile to filter data above
- **upper_quantile** – Upper quantile to filter data below
- **display_figure** – Whether to display the figure in addition to returning it

6.3 datetime_summary

datetime_summary(*data: DataFrame, column: str, fig_height: int = 1000, fig_width: int = 1200, ts_freq: str = 'auto', ts_type: str = 'lines', trend_line: str = 'auto', lower_quantile: float = 0, upper_quantile: float = 1, display_figure: bool = False*) → Figure

Creates a univariate EDA summary for a datetime data column in a pandas DataFrame.

Parameters

- **data** – pandas DataFrame to perform EDA on
- **column** – A string matching a column in the data

- **fig_height** – Height of the plot in inches
- **fig_width** – Width of the plot in inches
- **ts_freq** – String describing the frequency at which to aggregate data in one of two formats:
  - A pandas offset string.
  - A human readable string in the same format passed to date breaks (e.g. "4 months")

  Default is to attempt to intelligently determine a good aggregation frequency.
- **ts_type** – 'lines', 'markers', or 'lines+markers' to plot a line, points, or line + points
- **trend_line** –

  Trend line to plot over data. "None" produces no trend line. Other options are passed to geom_smooth.

  a time period ranging from seconds to years. (e.g. '1 year', '3 minutes')
- **lower_quantile** – Lower quantile to filter data above
- **upper_quantile** – Upper quantile to filter data below
- **display_figure** – Whether to display the figure in addition to returning it

6.4 text_summary

text_summary(*data: DataFrame*, *column: str*, *fig_height: int = 1000*, *fig_width: int = 1200*, *top_ngrams: int = 10*, *remove_punct: bool = True*, *remove_stop: bool = True*, *lower_case: bool = True*, *display_figure: bool = False*) → Figure

Creates a univariate EDA summary for a text variable column in a pandas DataFrame. Currently only supports English.

Parameters

- **data** – Dataset to perform EDA on
- **column** – A string matching a column in the data
- **fig_height** – Height of the plot in pixels
- **fig_width** – Width of the plot in pixels
- **top_ngrams** – Maximum number of ngrams to plot for the top most frequent unigrams to trigrams
- **remove_punct** – Whether to remove punctuation during tokenization
- **remove_stop** – Whether to remove stop words during tokenization
- **lower_case** – Whether to lower case text for tokenization
- **display_figure** – Whether to display the figure in addition to returning it

6.5 collection_summary

collection_summary(*data: DataFrame*, *column: str*, *fig_height: int = 1000*, *fig_width: int = 1000*, *top_entries: int = 10*, *sort_collections: bool = False*, *remove_duplicates: bool = False*, *display_figure: bool = False*) → Figure

Creates a univariate EDA summary for a collections column in a pandas DataFrame.

The provided column should be an object type containing lists, tuples, or sets.

Parameters

- **data** – Dataset to perform EDA on
- **column** – A string matching a column in the data
- **fig_height** – Height of the plot in inches
- **fig_width** – Width of the plot in inches
- **top_entries** – Max number of entries to show for countplots
- **sort_collections** – Whether to sort collections and ignore original order
- **remove_duplicates** – Whether to remove duplicate entries from collections
- **display_figure** – Whether to display the figure in addition to returning it

6.6 url_summary

url_summary(*data: DataFrame*, *column: str*, *fig_height: int = 1000*, *fig_width: int = 1200*, *top_entries: int = 10*, *display_figure: bool = False*) → Figure

Creates a univariate EDA summary for a url column in a pandas DataFrame. The provided column should be a string/object column containing urls.

Parameters

- **data** – Dataset to perform EDA on
- **column** – A string matching a column in the data
- **fig_height** – Height of the plot in inches
- **fig_width** – Width of the plot in inches
- **top_entries** – Max number of entries to show for countplots
- **display_figure** – Whether to display the figure in addition to returning it

CHAPTER

SEVEN

BIVARIATE SUMMARY FUNCTIONS

Functions for creating bivariate EDA summaries.

`categorical_categorical_summary`	Generates an EDA summary of two categorical variables
`categorical_numeric_summary`	Generates an EDA summary of the relationship between a categorical variable as the independent variable and a numeric variable as the dependent variable.
`numeric_categorical_summary`	Generates an EDA summary of the relationship of a numeric variable on a categorical variable.
`numeric_numeric_summary`	Creates a bivariate EDA summary for two numeric data columns in a pandas DataFrame.

7.1 categorical_categorical_summary

categorical_categorical_summary(*data: DataFrame*, *column1: str*, *column2: str*, *fig_height: int = 1000*, *fig_width: int = 1200*, *order1: Union[str, List] = 'auto'*, *order2: Union[str, List] = 'auto'*, *barmode: str = 'stack'*, *max_levels: int = 30*, *include_missing: bool = False*, *display_figure: bool = False*) → Figure

Generates an EDA summary of two categorical variables

Parameters

- **data** – pandas DataFrame with data to be plotted
- **column1** – First categorical column in the data to plot as independent variable
- **column2** – Second categorical column in the data to plot as dependent variable
- **fig_width** – Figure width in pixels
- **fig_height** – Figure height in pixels
- **order1** – Order in which to sort the levels of the first variable:
  - **'auto'**: sorts ordinal variables by provided ordering, nominal variables by descending frequency, and numeric variables in sorted order.
  - **'descending'**: sorts in descending frequency.
  - **'ascending'**: sorts in ascending frequency.
  - **'sorted'**: sorts according to sorted order of the levels themselves.
  - **'random'**: produces a random order. Useful if there are too many levels for one plot.

  Or you can pass a list of level names in directly for your own custom order.

- **order2** – Same as order1 but for the second variable
- **barmode** – Type of bar plot aggregation. One of [‘stack’, ‘group’, ‘overlay’, ‘relative’]
- **max_levels** – Maximum number of levels to attempt to plot on a single plot. If exceeded, only the max_level - 1 levels will be plotted and the remainder will be grouped into an ‘Other’ category.
- **include_missing** – Whether to include missing values as an additional level in the data to be plotted
- **display_figure** – Whether to display the figure in addition to returning it

7.2 categorical_numeric_summary

categorical_numeric_summary(*data: DataFrame, column1: str, column2: str, fig_height: int = 1000, fig_width: int = 1200, order: Union[str, List] = 'auto', max_levels: int = 10, include_missing: bool = False, lower_quantile: float = 0, upper_quantile: float = 1, hist_bins: Optional[int] = None, dist_type: str = 'kde_only', transform: str = 'identity', display_figure: bool = False*) → Figure

Generates an EDA summary of the relationship between a categorical variable as the independent variable and a numeric variable as the dependent variable.

Parameters

- **data** – pandas DataFrame with data to be plotted
- **column1** – Categorical column in the data to be used as independent variable
- **column2** – Numeric column in the data to be used as dependent variable
- **fig_height** – Height of the figure in pixels
- **fig_width** – Width of the figure in pixels
- **order** – Order in which to sort the levels of the categorical variable:
  - **’auto’**: sorts ordinal variables by provided ordering, nominal variables by descending frequency, and numeric variables in sorted order.
  - **’descending’**: sorts in descending frequency.
  - **’ascending’**: sorts in ascending frequency.
  - **’sorted’**: sorts according to sorted order of the levels themselves.
  - **’random’**: produces a random order. Useful if there are too many levels for one plot.

  Or you can pass a list of level names in directly for your own custom order.
- **max_levels** – Maximum number of levels to attempt to plot on a single plot. If exceeded, only the max_level - 1 levels will be plotted and the remainder will be grouped into an ‘Other’ category.
- **include_missing** – Whether to include missing values as an additional level in the data to be plotted
- **lower_quantile** – Lower quantile to filter numeric column above
- **upper_quantile** – Upper quantile to filter numeric column below
- **hist_bins** – Number of bins to use for the histogram. Default will use plotly defaults

- **dist_type** – Type of distribution to plot:
  - **'norm_hist+kde'**: Plots histograms with overlaid KDE normalized to be a probabililty density
  - **'norm_hist_only'**: Plots just histograms normalized to be a probabililty density
  - **'unnorm_hist_only'**: Plots just unnormalized histograms with counts
  - **'kde_only'**: Plots just KDEs normalized to be a probabililty density
- **transform** – Transformation to apply to the numeric column for plotting:
  - 'identity': no transformation
  - 'log': apply a logarithmic transformation (zero and negative values will be filtered out)
  - 'sqrt': apply a square root transformation
- **display_figure** – Whether to display the figure in addition to returning it

7.3 numeric_categorical_summary

numeric_categorical_summary(*data: DataFrame, column1: str, column2: str, fig_height: int = 600, fig_width: int = 1200, order: Union[str, List] = 'auto', num_intervals: int = 4, interval_type: str = 'quantile', max_levels: int = 30, include_missing: bool = False, display_figure: bool = False*) → Figure

Generates an EDA summary of the relationship of a numeric variable on a categorical variable.

Parameters

- **data** – pandas DataFrame with data to be plotted
- **column1** – Numeric column in the data to be plotted as independent variable
- **column2** – Categorical column in the data to be plotted as dependent variable
- **fig_height** – Height of the figure in pixels
- **fig_width** – Width of the figure in pixels
- **order** – Order in which to sort the levels of the categorical variable:
  - **'auto'**: sorts ordinal variables by provided ordering, nominal variables by descending frequency, and numeric variables in sorted order.
  - **'descending'**: sorts in descending frequency.
  - **'ascending'**: sorts in ascending frequency.
  - **'sorted'**: sorts according to sorted order of the levels themselves.
  - **'random'**: produces a random order. Useful if there are too many levels for one plot.

  Or you can pass a list of level names in directly for your own custom order.
- **num_intervals** – Number of intervals to bin column1 into
- **interval_type** – Type of intervals to bin column1 into. 'quantile' or 'equal width'
- **max_levels** – Maximum number of levels to attempt to plot on a single plot. If exceeded, only the max_level - 1 levels will be plotted and the remainder will be grouped into an 'Other' category.

- **include_missing** – Whether to include missing values as an additional level in the data to be plotted
- **display_figure** – Whether to display the figure in addition to returning it

7.4 numeric_numeric_summary

numeric_numeric_summary(*data: DataFrame, column1: str, column2: str, fig_height: int = 1200, fig_width: int = 1200, trend_line: str = 'auto', opacity: float = 1.0, hist_bins: Optional[int] = None, lower_quantile1: float = 0, upper_quantile1: float = 1, lower_quantile2: float = 0, upper_quantile2: float = 1, num_intervals: int = 4, interval_type: str = 'quantile', transform1: str = 'identity', transform2: str = 'identity', display_figure: bool = False*) → Figure

Creates a bivariate EDA summary for two numeric data columns in a pandas DataFrame.

Parameters

- **data** – pandas DataFrame to perform EDA on
- **column1** – name of numeric column to plot as independent variable
- **column2** – name of numeric column to plot as dependent variable
- **fig_height** – Height of the plot in pixels
- **fig_width** – Width of the plot in pixels
- **opacity** – Level of opacity to apply to points in scatterplot (0 = fully transparent, 1 = fully opaque)
- **trend_line** – Trend line to plot over data. Default is to plot no trend line. Other options are passed to geom_smooth.
- **hist_bins** – Number of bins to use for the histogram. Default will use plotly defaults
- **lower_quantile1** – Lower quantile to filter data above for column1
- **upper_quantile1** – Upper quantile to filter data below for column1
- **lower_quantile2** – Lower quantile to filter data above for column2
- **upper_quantile2** – Upper quantile to filter data below for column2
- **num_intervals** – Number of intervals to bin column1 into for the boxplots
- **interval_type** – Type of intervals to bin column1 into for the boxplots. 'quantile' or 'equal width'
- **transform1** – Transformation to apply to the column1 for plotting:
  - 'identity': no transformation
  - 'log': apply a logarithmic transformation (zero and negative values will be filtered out)
  - 'sqrt': apply a square root transformation
- **transform2** – Transformation to apply to the column2 data for plotting. Same options as for column1.
- **display_figure** – Whether to display the figure in addition to returning it

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

C

D

N

T

U