Leon Shpaner, M.S.
Oscar Gil, M.S.

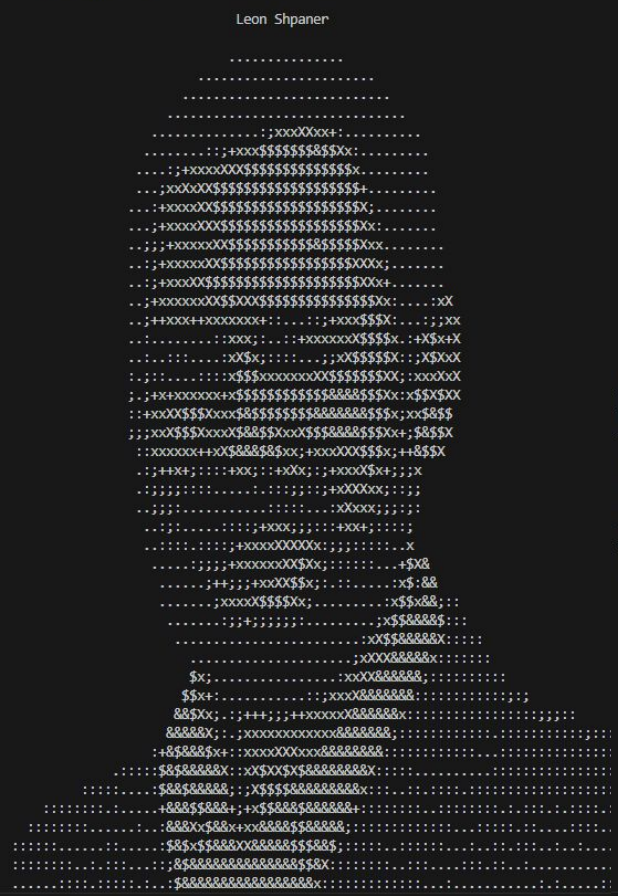
jupytercon

EDA-Toolkit

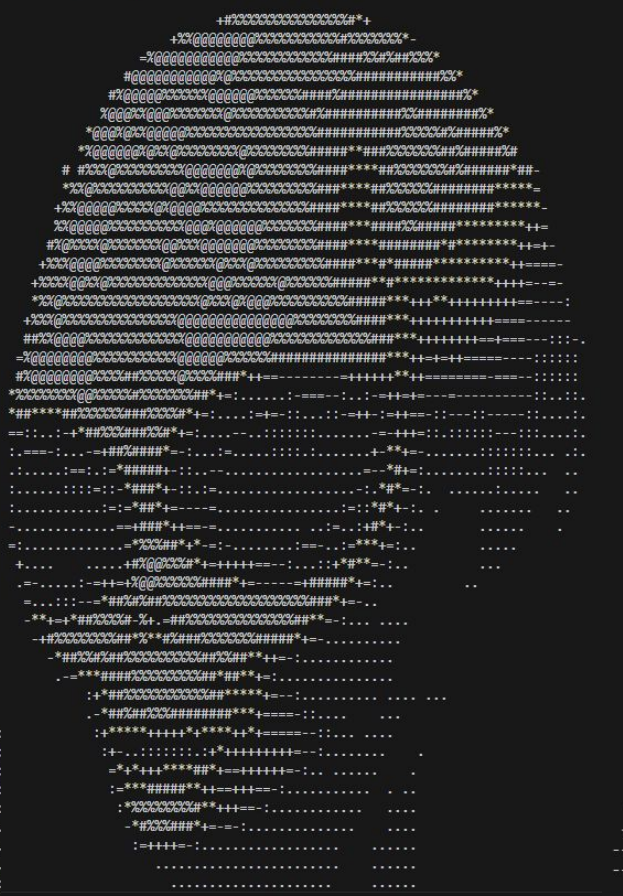


#JupyterCon

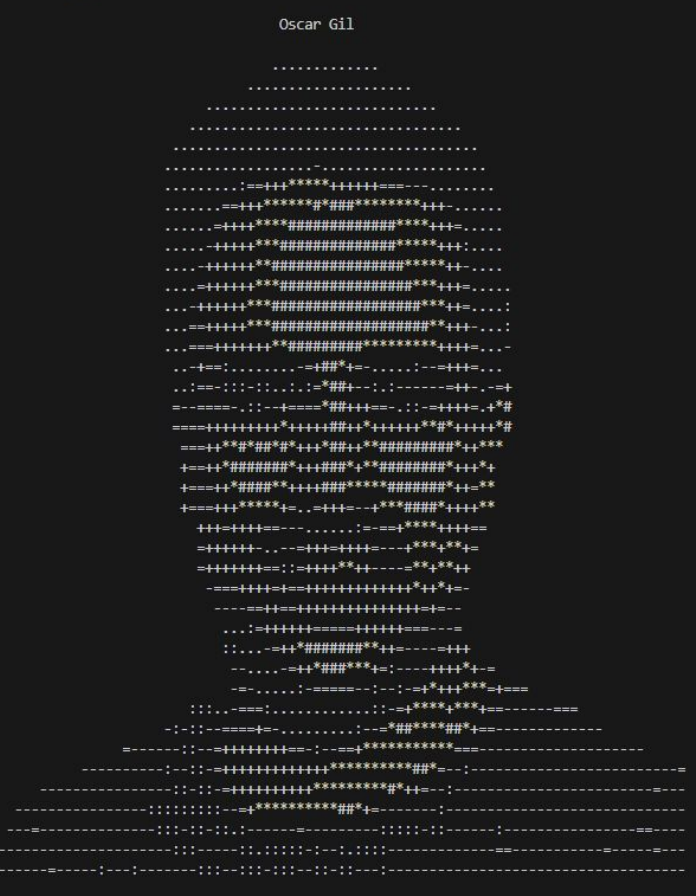
```
(venv_eda_311) lshpaner@LAPTOP-JRSVC86M:~/Python_Projects/eda_toolkit$ python
>>> print_art("leon_shpaner_bb")
--- leon_shpaner_bb ---
```




```
(venv_eda_311) lshpaner@LAPTOP-JRSVC86M:~/Python_Projects/eda_toolkit$ python
>>> print_art("grey_alien_bb")
--- grey_alien_bb ---
```



```
(venv_eda_311) lshpaner@LAPTOP-JRSVC86M:~/Python_Projects/eda_toolkit$ python
>>> print_art("oscar_gil_bb")
--- oscar_gil_bb ---
```



Adult Income Dataset

UCI Machine Learning Repository: https://archive.ics.uci.edu/dataset/2/adult

Adult

Donated on 4/30/1996

Predict whether annual income of an individual exceeds $50K/yr based on census data. Also known as "Census Income" dataset.

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate	Social Science	Classification

Feature Type	# Instances	# Features
Categorical, Integer	48842	14

Dataset Information

Additional Information

Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

...

SHOW MORE

Has Missing Values?

Yes

Variables Table

DOWNLOAD (605.7 KB)

IMPORT IN PYTHON

CITE

257 citations

541052 views

Keywords

census

Creators

Barry Becker
Silicon Graphics

Ronny Kohavi
ronnyk@live.com
Consultant/Stanford

DOI

10.24432/C5XW20

jupytercon

We built generate_table1



generate_table1

jupytercon

```python
from eda_toolkit import generate_table1

table1 = generate_table1(
    df=df,
    continuous_cols=["hours-per-week", "age", "education-num"],
    value_counts=False,
    max_categories=3,
    export_markdown=True,
    decimal_places=0,
)

print(table1)
```

Variable	Type	Mean	SD	Median	Min	Max	Mode	Missing (n)	Missing (%)	Count	Proportion (%)
hours-per-week	Continuous	40	12	40	1	99	40	0	0	48,842	100
age	Continuous	39	14	37	17	90	36	0	0	48,842	100
education-num	Continuous	10	3	10	1	16	9	0	0	48,842	100

generate_table1

jupytercon

```python
from eda_toolkit import generate_table1

table1 = generate_table1(
    df=df,
    continuous_cols=["hours-per-week", "age", "education-num"],
    categorical_cols=["sex", "race", "workclass"],
    value_counts=False,
    max_categories=3,
    export_markdown=True,
    decimal_places=0,
)

print(table1)
```

Variable	Type	Mean	SD	Median	Min	Max	Mode	Missing (n)	Missing (%)	Count	Proportion (%)
hours-per-week	Continuous	40	12	40	1	99	40	0	0	48,842	100
age	Continuous	39	14	37	17	90	36	0	0	48,842	100
education-num	Continuous	10	3	10	1	16	9	0	0	48,842	100
sex	Categorical						Male	0	0	48,842	100
race	Categorical						White	0	0	48,842	100
workclass	Categorical						Private	963	2	47,879	98

generate_table1

jupytercon

```
table1 = generate_table1(
    df=df,
    categorical_cols=["sex", "race", "workclass"],
    value_counts=True,
    max_categories=3,
    export_markdown=True,
    drop_columns=["Mode"]
    decimal_places=0,
)
print(table1)
```

Variable	Mean	SD	Median	Min	Max	Missing (n)	Missing (%)	Count	Proportion (%)
age	39	14	37	17	90	0	0	48,842	100
capital-gain	1,079	7,452	0	0	99,999	0	0	48,842	100
capital-loss	88	403	0	0	4,356	0	0	48,842	100
education-num	10	3	10	1	16	0	0	48,842	100
fnlwgt	189,664	105,604	178,144	12,285	1,490,400	0	0	48,842	100
hours-per-week	40	12	40	1	99	0	0	48,842	100
sex						0	0	48,842	100
sex = Male						0	0	32,650	67
sex = Female						0	0	16,192	33
race						0	0	48,842	100
race = White						0	0	41,762	86
race = Black						0	0	4,685	10
race = Asian-Pac-I						0	0	1,519	3
workclass						963	2	47,879	98
workclass = Privat						963	2	33,906	69
workclass = Self-e						963	2	3,862	8
workclass = Local-						963	2	3,136	6

```
p_value_table_1_cat = generate_table1(
    df_table1_cat,
    include_types="categorical",
    value_counts=True,
    export_markdown=True,
    groupby_col="income",
    drop_columns=["Missing (n)", "Missing (%)", "income", "Type", "Mode"],
    drop_variables="income",
    markdown_path=os.path.join(data_output, "p_val_table1_summary.md"),
)
print(p_value_table_1_cat )
```

```
Using Chi-squared test for variable: fnlwgt_bin
Using Chi-squared test for variable: age_group
Using Chi-squared test for variable: income
```

Variable	Count	Proportion (%)	<=50K (n = 37,155)	>50K (n = 11,687)	P-value
fnlwgt_bin	48,842	100.00	37,155	11,687	0.66
fnlwgt_bin = Bin 1	42,729	87.48	32,517 (87.52%)	10,212 (87.38%)	
fnlwgt_bin = Bin 2	5,898	12.08	4,466 (12.02%)	1,432 (12.25%)	
fnlwgt_bin = Bin 3	186	0.38	148 (0.40%)	38 (0.33%)	
fnlwgt_bin = Bin 4	22	0.05	18 (0.05%)	4 (0.03%)	
fnlwgt_bin = Bin 5	7	0.01	6 (0.02%)	1 (0.01%)	
age_group	48,842	100.00	37,155	11,687	0.00
age_group = 18-29	13,920	28.50	13,174 (35.46%)	746 (6.38%)	
age_group = 30-39	12,929	26.47	9,468 (25.48%)	3,461 (29.61%)	
age_group = 40-49	10,724	21.96	6,738 (18.13%)	3,986 (34.11%)	
age_group = 50-59	6,619	13.55	4,110 (11.06%)	2,509 (21.47%)	
age_group = 60-69	3,054	6.25	2,245 (6.04%)	809 (6.92%)	
age_group = 70-79	815	1.67	668 (1.80%)	147 (1.26%)	
age_group = < 18	595	1.22	595 (1.60%)	0 (0.00%)	
age_group = 80-89	131	0.27	115 (0.31%)	16 (0.14%)	
age_group = 90-99	55	0.11	42 (0.11%)	13 (0.11%)	
age_group = 100 +	0	0.00	0 (0.00%)	0 (0.00%)	

jupytercon

We built stacked_crosstab_plot



```
# Define the func_col to use in the loop in order of usage
func_col = ["sex", "income"]

# Define the legend_labels to use in the loop
legend_labels_list = [
    ["Male", "Female"],  # Corresponds to "sex"
    ["<=50K", ">50K"],  # Corresponds to "income"
]

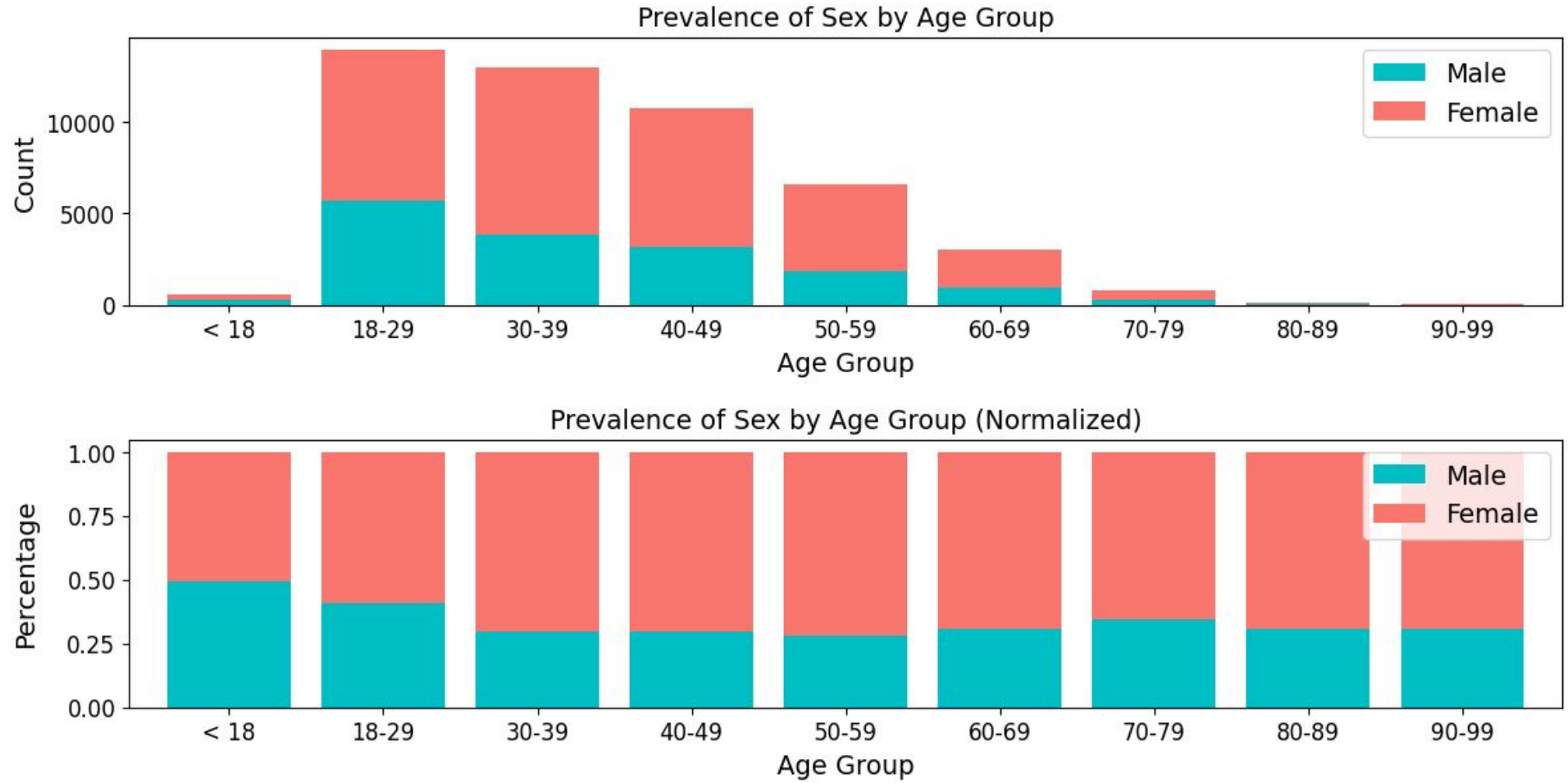
# Define titles for the plots
title = [
    "Sex",
    "Income",
]
```

```
from eda_toolkit import kde_distributions

stacked_crosstabs = stacked_crosstab_plot(
    df=df,
    col="age_group",
    func_col=func_col,
    legend_labels_list=legend_labels_list,
    title=title,
    kind="bar",
    width=0.8,
    rot=0,
    custom_order=None,
    color=["#00BFC4", "#F8766D"],
    output="plots_only",
    return_dict=True,
    x=14,
    y=8,
    p=10,
    logscale=False,
    plot_type="both",
    show_legend=True,
    label_fontsize=14,
    tick_fontsize=12,
)
```

```
crosstab_age_sex = stacked_crosstabs["sex"]


crosstab_age_sex
```


Prevalence of Sex by Age Group
Count
10000
5000
0
Male
Female
< 18
18-29
30-39
40-49
50-59
60-69
70-79
80-89
90-99
Age Group
Prevalence of Sex by Age Group (Normalized)
Percentage
1.00
0.75
0.50
0.25
0.00
Male
Female
< 18
18-29
30-39
40-49
50-59
60-69
70-79
80-89
90-99
Age Group

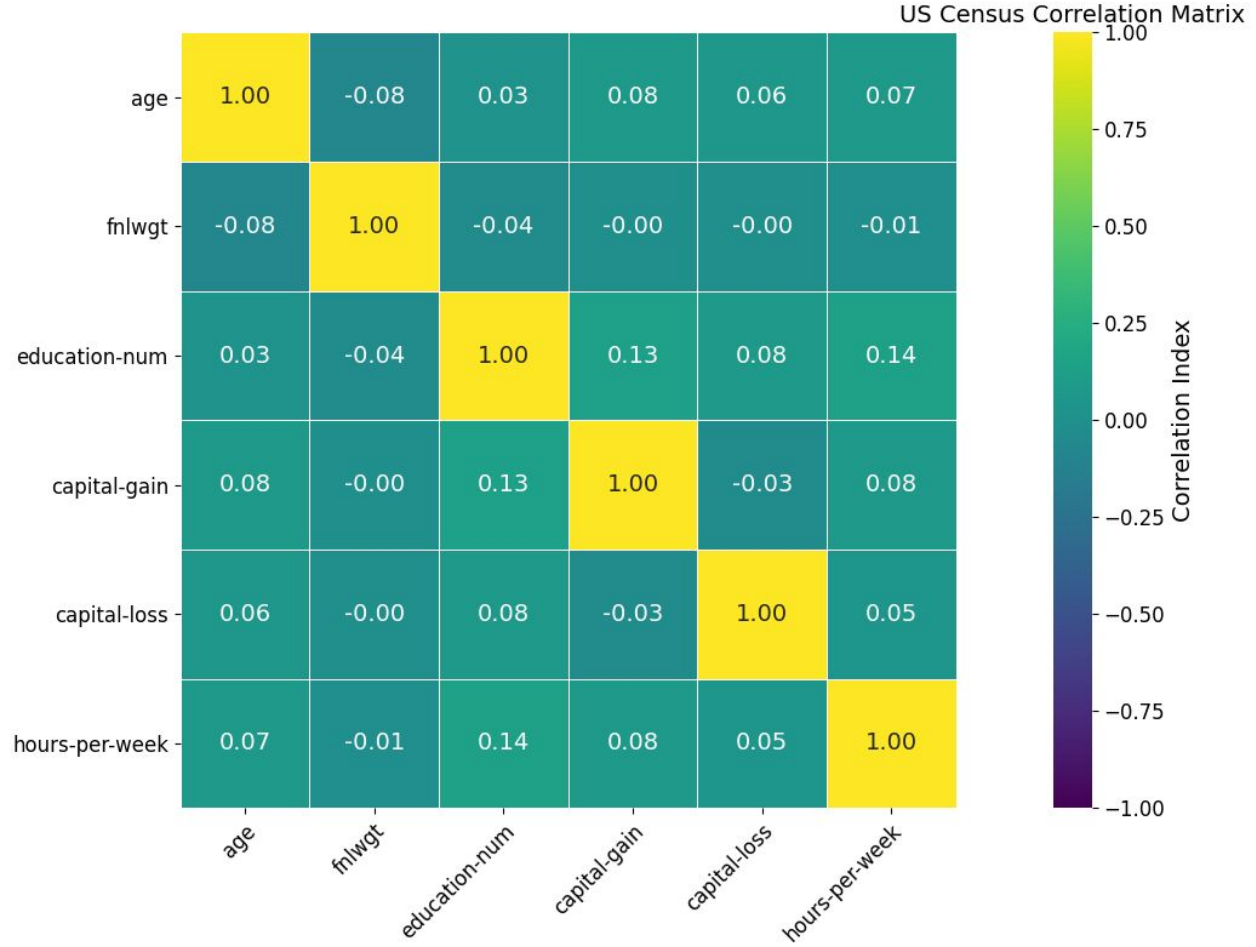
Crosstab for sex

sex	Male	Female	Total	Male_%	Female_%
age_group					
< 18	295	300	595	49.58	50.42
18-29	5707	8213	13920	41.00	59.00
30-39	3853	9076	12929	29.80	70.20
40-49	3188	7536	10724	29.73	70.27
50-59	1873	4746	6619	28.30	71.70
60-69	939	2115	3054	30.75	69.25
70-79	280	535	815	34.36	65.64
80-89	40	91	131	30.53	69.47
90-99	17	38	55	30.91	69.09
Total	16192	32650	48842	33.15	66.85

jupytercon

We built flex_corr_matrix

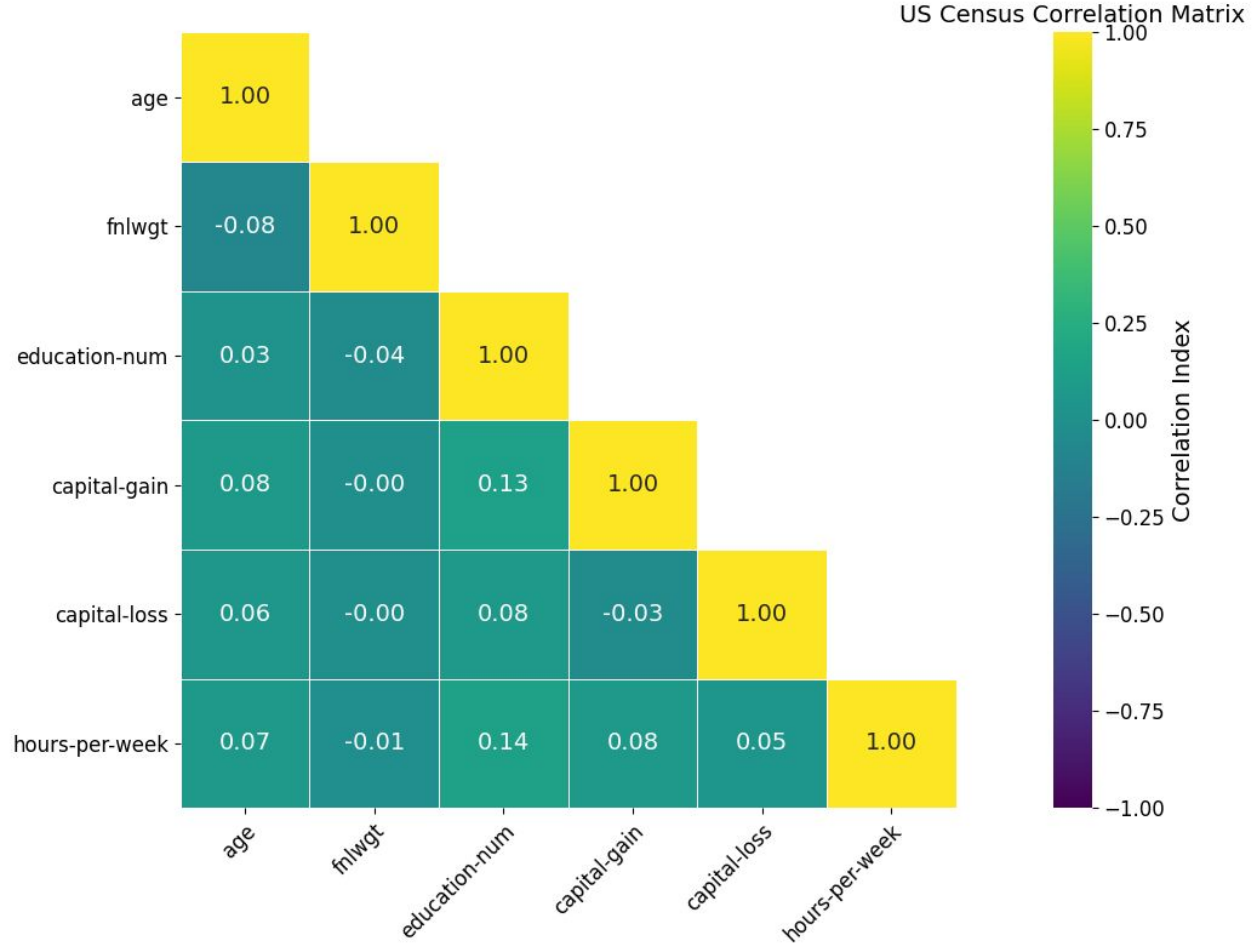





```python
from eda_toolkit import flex_corr_matrix

flex_corr_matrix(
    df=df,
    cols=df_num.columns.to_list(),
    annot=True,
    cmap="viridis",
    figsize=(10, 8),
    title="US Census Correlation Matrix",
    xlabel_alignment="right",
    label_fontsize=14,
    tick_fontsize=12,
    xlabel_rot=45,
    ylabel_rot=0,
    text_wrap=50,
    vmin=-1,
    vmax=1,
    cbar_label="Correlation Index",
    triangular=False,

)
```




```python
from eda_toolkit import flex_corr_matrix

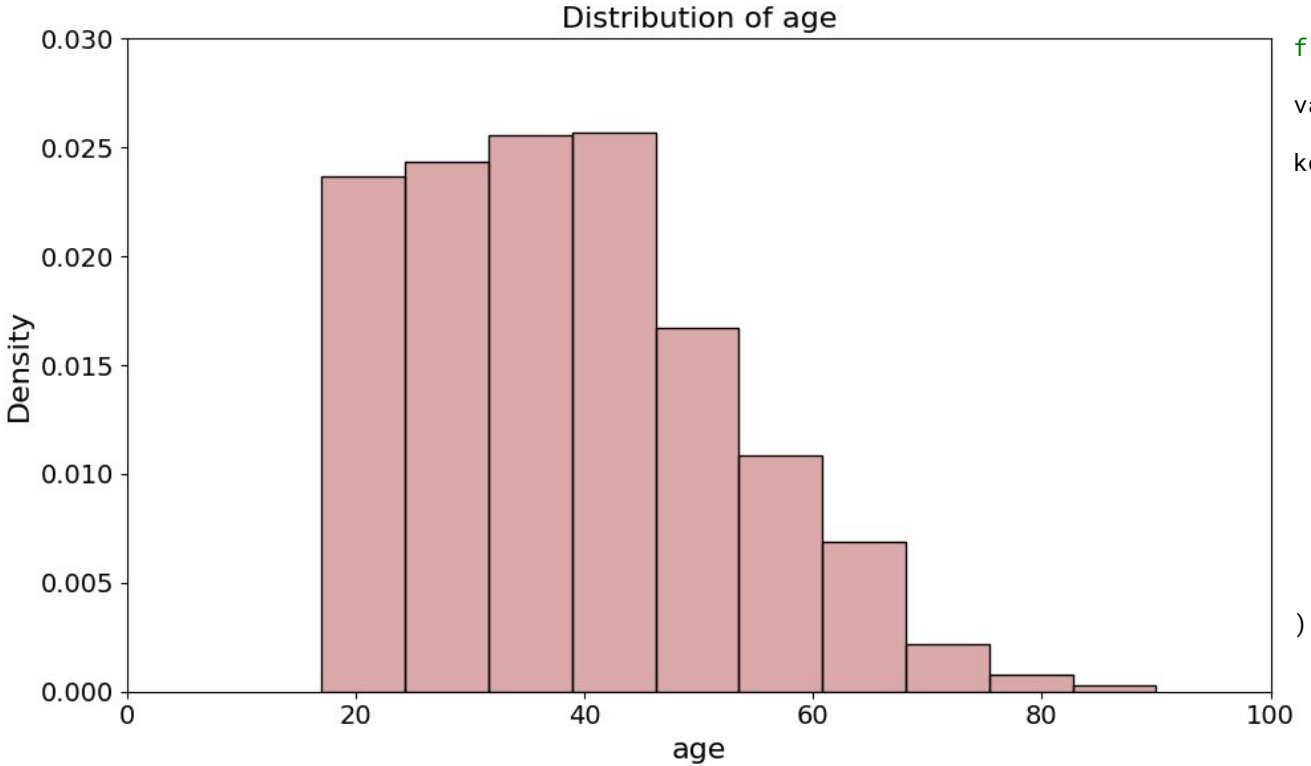
flex_corr_matrix(
    df=df,
    cols=df_num.columns.to_list(),
    annot=True,
    cmap="viridis",
    figsize=(10, 8),
    title="US Census Correlation Matrix",
    xlabel_alignment="right",
    label_fontsize=14,
    tick_fontsize=12,
    xlabel_rot=45,
    ylabel_rot=0,
    text_wrap=50,
    vmin=-1,
    vmax=1,
    cbar_label="Correlation Index",
    triangular=True,

)
```

jupytercon

We built kde_distributions

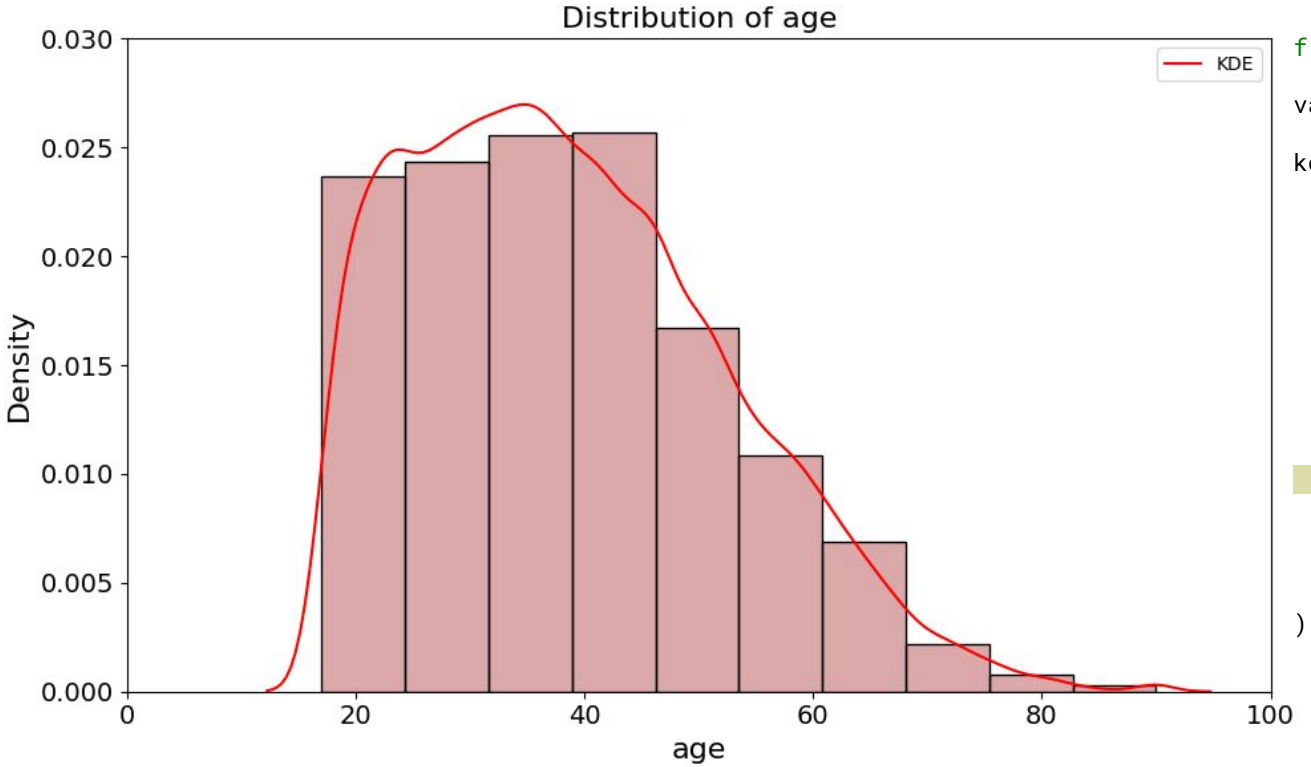





```python
from eda_toolkit import kde_distributions

vars_of_interest = ["age"]

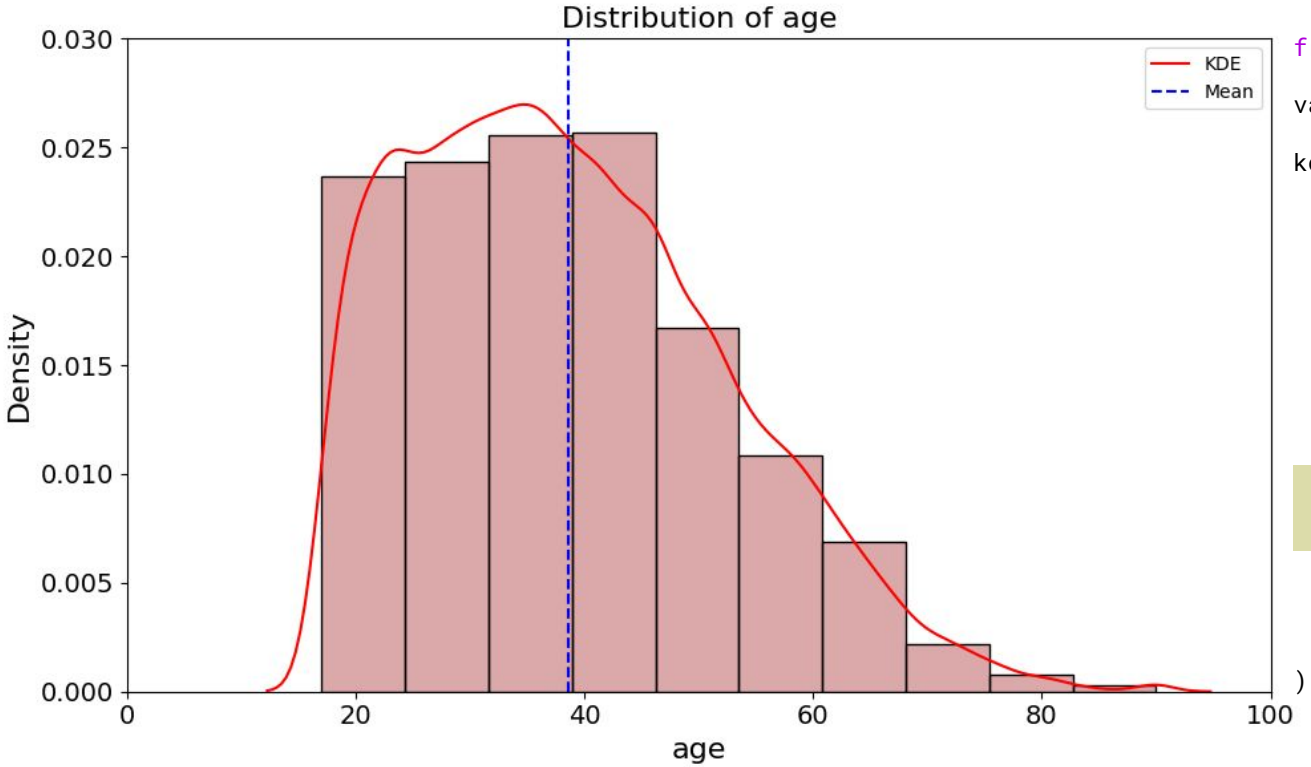
kde_distributions(
    df=df,
    figsize=(10, 6),
    hist_color="brown",
    fill=True,
    fill_alpha=0.40,
    text_wrap=50,
    bbox_inches="tight",
    vars_of_interest=vars_of_interest,
    y_axis_label="Density",
    bins=10,
    plot_type="hist",
    label_fontsize=16,
    tick_fontsize=14,
    xlim=(0, 100),
    ylim=(0, 0.03),
)
```




```python
from eda_toolkit import kde_distributions

vars_of_interest = ["age"]

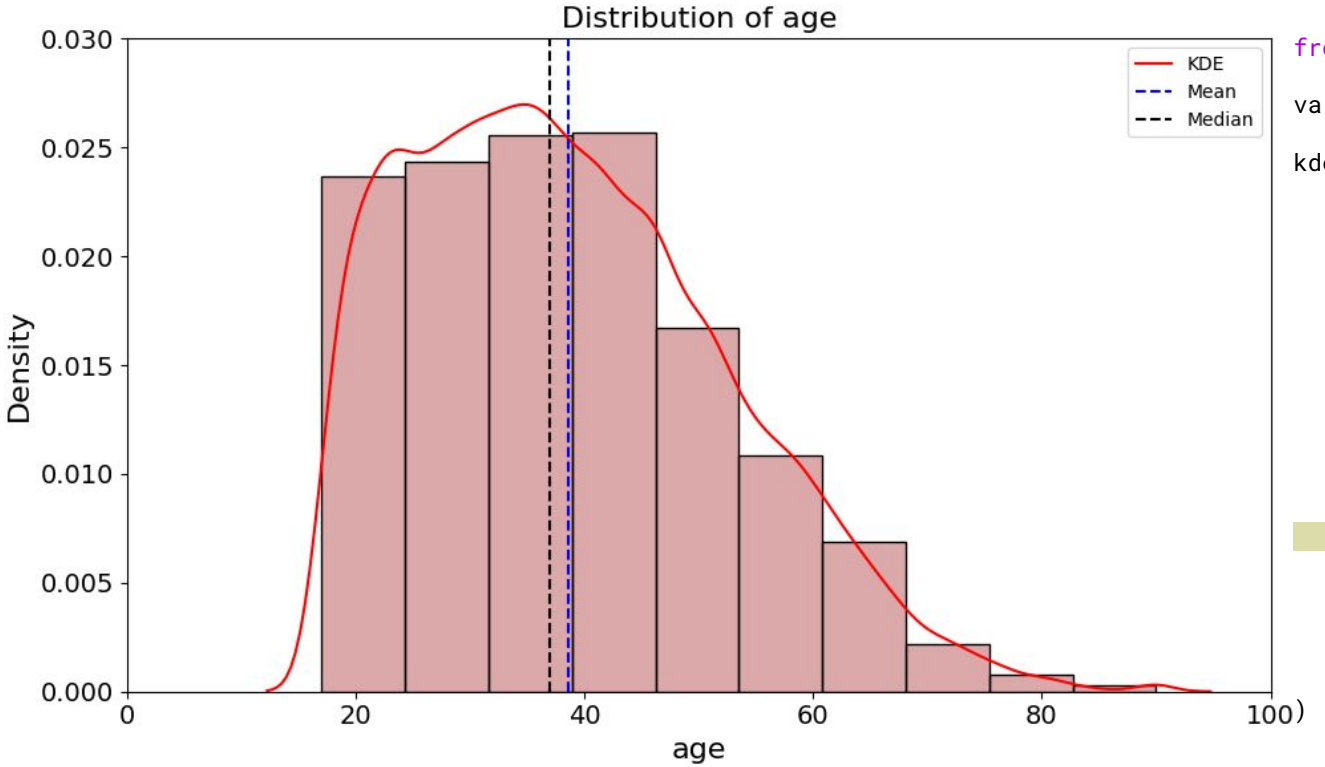
kde_distributions(
    df=df,
    figsize=(10, 6),
    hist_color="brown",
    fill=True,
    fill_alpha=0.40,
    text_wrap=50,
    bbox_inches="tight",
    vars_of_interest=vars_of_interest,
    y_axis_label="Density",
    bins=10,
    plot_type="both",
    label_fontsize=16,
    tick_fontsize=14,
    xlim=(0, 100),
    ylim=(0, 0.03),
)
```




```python
from eda_toolkit import kde_distributions

vars_of_interest = ["age"]

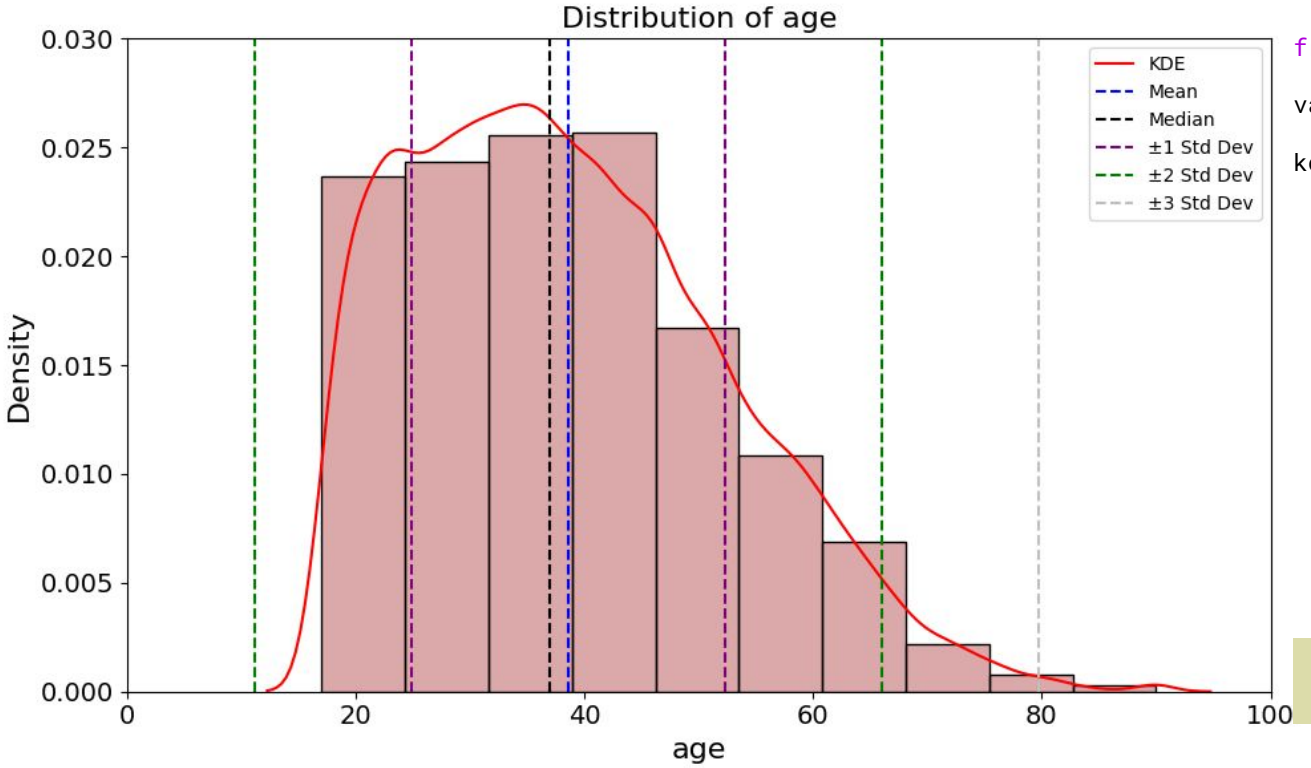
kde_distributions(
    df=df,
    figsize=(10, 6),
    hist_color="brown",
    fill=True,
    fill_alpha=0.40,
    text_wrap=50,
    bbox_inches="tight",
    vars_of_interest=vars_of_interest,
    y_axis_label="Density",
    bins=10,
    plot_type="both",
    plot_mean=True
    mean_color=”blue”
    label_fontsize=16,
    tick_fontsize=14,
    xlim=(0, 100),
    ylim=(0, 0.03),
)
```




```python
from eda_toolkit import kde_distributions

vars_of_interest = ["age"]

kde_distributions(
    df=df,
    figsize=(10, 6),
    hist_color="brown",
    fill=True,
    fill_alpha=0.40,
    text_wrap=50,
    bbox_inches="tight",
    vars_of_interest=vars_of_interest,
    y_axis_label="Density",
    bins=10,
    plot_type="both",
    plot_mean=True,
    plot_median=True,
    mean_color="blue"
    label_fontsize=16,
    tick_fontsize=14,
    xlim=(0, 100),
    ylim=(0, 0.03),
)
```




```python
from eda_toolkit import kde_distributions

vars_of_interest = ["age"]

kde_distributions(
    df=df,
    figsize=(10, 6),
    hist_color="brown",
    fill=True,
    fill_alpha=0.40,
    text_wrap=50,
    bbox_inches="tight",
    vars_of_interest=vars_of_interest,
    y_axis_label="Density",
    bins=10,
    plot_type="both",
    plot_mean=True,
    mean_color="blue",
    plot_median=True,
    label_fontsize=16,
    tick_fontsize=14,
    std_dev_levels=[1, 2, 3],
    std_color=[“purple”,
               "green","silver"],
    xlim=(0, 100),
    ylim=(0, 0.03),
)
```

jupytercon

We built summarize_all_combinations



```python
from eda_toolkit import summarize_all_combinations

# Define unique variables for the analysis
unique_vars = ["age_group","workclass",
               "education","occupation",
               "Race","sex","income",]

# Generate summary tables for all combinations of the specified
variables
summary_tables, all_combinations = summarize_all_combinations(
    df=df,
    data_path=data_output,
    variables=unique_vars,
    data_name="census_summary_tables.xlsx",
)

# Print all combinations of variables
print(all_combinations)
```

```
Generating combinations: 100%|██████████| 120/120 [00:01<00:00, 76.56it/s]
Writing summary tables: 100%|██████████| 120/120 [00:41<00:00,  2.87it/s]
Finalizing Excel file: 100%|██████████| 1/1 [00:00<00:00, 13706.88it/s]
Data saved to ../data_output/census_summary_tables.xlsx


(age_group, workclass), (age_group, education), (age_group, occupation), ...
(workclass, sex, income), (education, race, income), ...
(age_group, education, race, sex), ...
(workclass, education, occupation, race, sex, income),
(age_group, workclass, education, occupation, race, sex, income)
```

A1 Sheet Name

	A	B	C
1	**Sheet Name**	**Description**	
2	age_group_workclass	Summary for age_group, workclass	
3	age_group_education	Summary for age_group, education	
4	age_group_occupation	Summary for age_group, occupation	
5	age_group_race	Summary for age_group, race	
6	age_group_sex	Summary for age_group, sex	
7	age_group_income	Summary for age_group, income	
8	workclass_education	Summary for workclass, education	
9	workclass_occupation	Summary for workclass, occupation	
10	workclass_race	Summary for workclass, race	
11	workclass_sex	Summary for workclass, sex	
12	workclass_income	Summary for workclass, income	
13	education_occupation	Summary for education, occupation	
14	education_race	Summary for education, race	
15	education_sex	Summary for education, sex	
16	education_income	Summary for education, income	
17	occupation_race	Summary for occupation, race	
18	occupation_sex	Summary for occupation, sex	
19	occupation_income	Summary for occupation, income	
20	race_sex	Summary for race, sex	
21	race_income	Summary for race, income	
22	sex_income	Summary for sex, income	
23	age_group_workclass_education	Summary for age_group, workclass, education	
24	age_group_workclass_occupation	Summary for age_group, workclass, occupation	
25	age_group_workclass_race	Summary for age_group, workclass, race	
26	age_group_workclass_sex	Summary for age_group, workclass, sex	
27	age_group_workclass_income	Summary for age_group, workclass, income	
28	age_group_education_occupation	Summary for age_group, education, occupation	
29	age_group_education_race	Summary for age_group, education, race	
30	age_group_education_sex	Summary for age_group, education, sex	
31	age_group_education_income	Summary for age_group, education, income	
32	age_group_occupation_race	Summary for age_group, occupation, race	
33	age_group_occupation_sex	Summary for age_group, occupation, sex	
34	age_group_occupation_income	Summary for age_group, occupation, income	
35	age_group_race_sex	Summary for age_group, race, sex	
36	age_group_race_income	Summary for age_group, race, income	
37	age_group_sex_income	Summary for age_group, sex, income	

Table of Contents | age_group_workclass | age_group_education | age_group_occupation | age_group_race | age_group_sex | age_group_income | workclass_education | workclass_occupation | workclass_rac

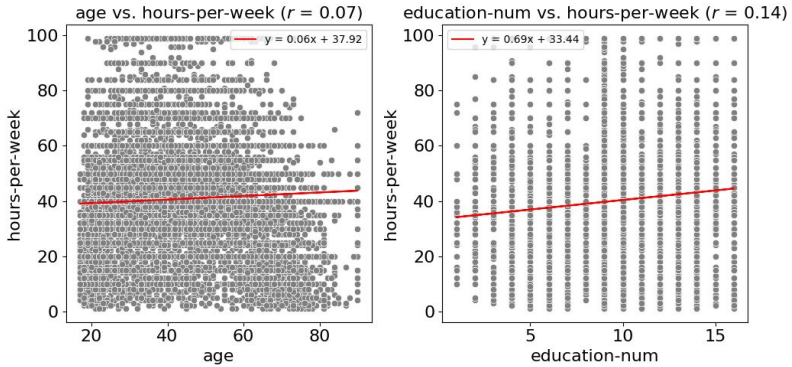
jupytercon

We built scatter_fit_plot



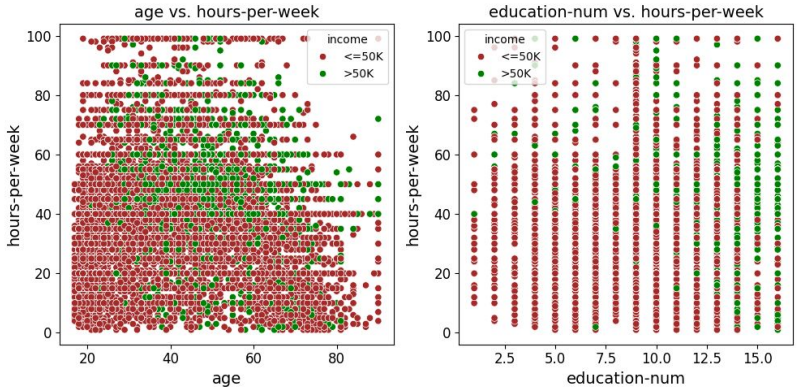
```
from eda_toolkit import scatter_fit_plot

scatter_fit_plot(
    df=df,
    x_vars=["age", "education-num"],
    y_vars=["hours-per-week"],
    show_legend=True,
    show_plot="subplots",
    subplot_figsize=None,
    label_fontsize=14,
    tick_fontsize=12,
    add_best_fit_line=True,
    scatter_color="#808080",
    show_correlation=True,
)
```

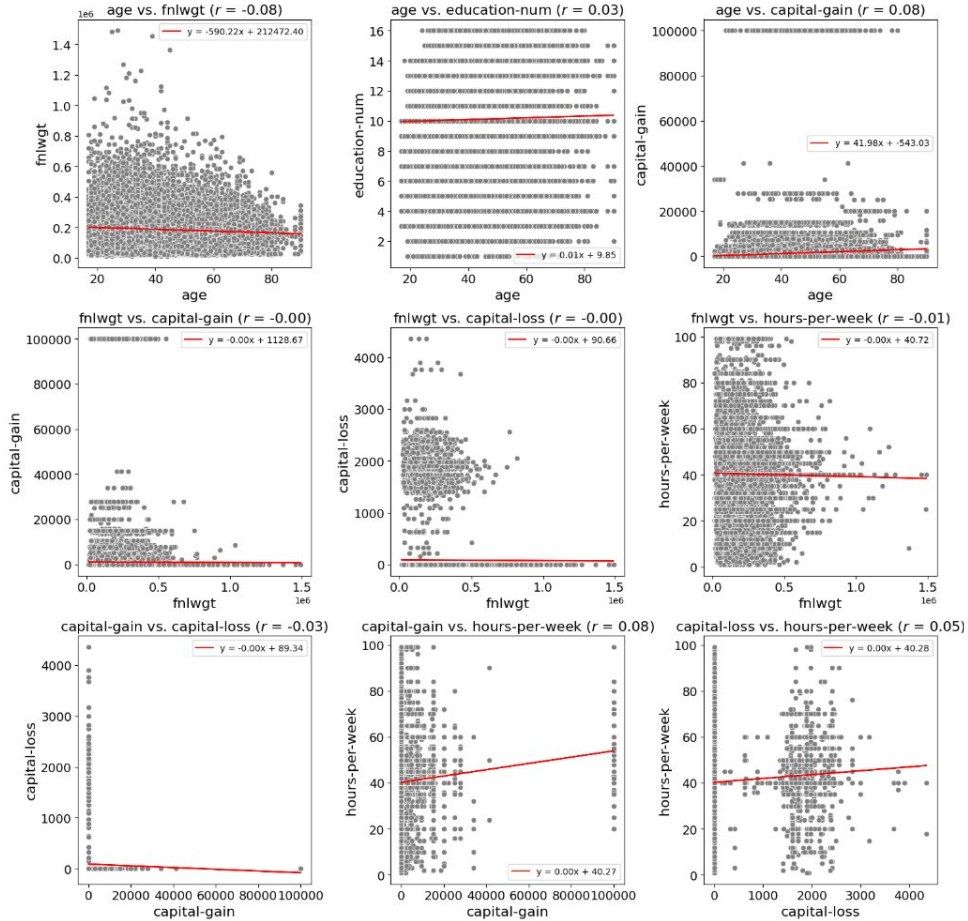



```
hue_dict = {"<=50K": "brown", ">50K": "green"}

scatter_fit_plot(
    df=df,
    x_vars=["age", "education-num"],
    y_vars=["hours-per-week"],
    show_legend=True,
    show_plot="subplots",
    label_fontsize=14,
    tick_fontsize=12,
    add_best_fit_line=False,
    scatter_color="#808080",
    hue="income",
    hue_palette=hue_dict,
    show_correlation=False,
)
```



scatter_fit_plot




```python
hue_dict = {"<=50K": "brown", ">50K": "green"}

scatter_fit_plot(
    df=df,
    x_vars=["age", "education-num"],
    all_vars=df.select_dtypes(np.number).columns.to_list(),
    y_vars=["hours-per-week"],
    show_legend=True,
    show_plot="subplots",
    label_fontsize=14,
    tick_fontsize=12,
    add_best_fit_line=False,
    scatter_color="#808080",
    hue="income",
    hue_palette=hue_dict,
    show_correlation=False,
)
```

jupytercon

We built data_doctor

- Wanted one call that can:
  analyze → transform → visualize → (optionally) write back
- Works for **any numeric feature**



jupytercon

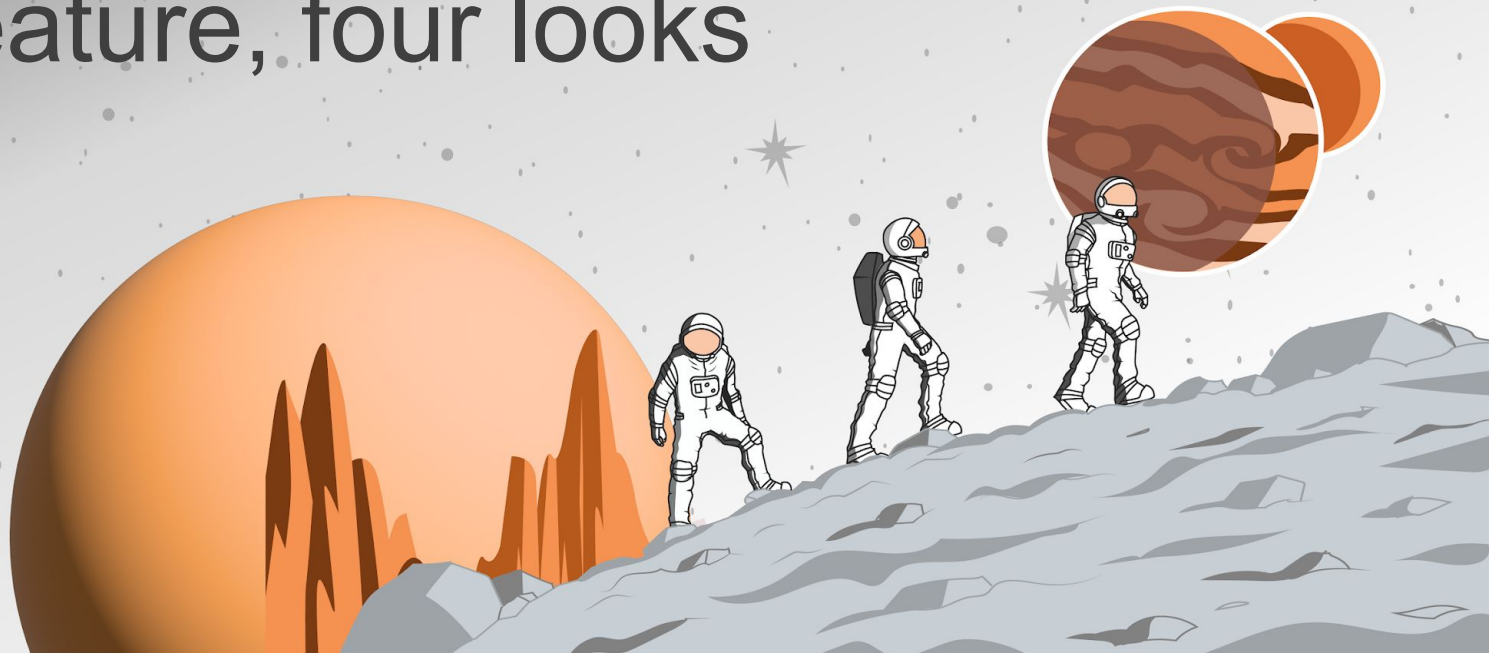
What data_doctor(...) does in one go

- Profiles the feature
- Plots it the same way every time
- Applies a named scaler
- Can **save** the transformed version back to df



jupytercon

One feature, four looks

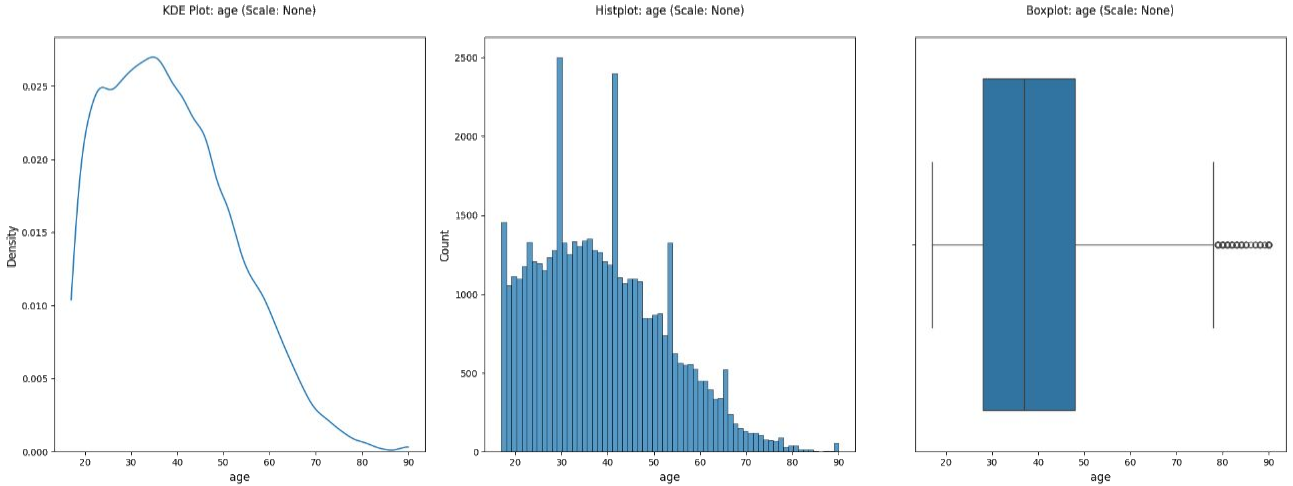


#JupyterCon

```
et.data_doctor(df=X, feature_name='age')
```

DATA DOCTOR SUMMARY REPORT

Feature	age
Statistic	**Value**
Min	17.0000
Max	90.0000
Mean	38.6436
Median	37.0000
Std Dev	13.7104
Quartile	**Value**
Q1 (25%)	28.0000
Q2 (50% = Median)	37.0000
Q3 (75%)	48.0000
IQR	20.0000
Outlier Bound	**Value**
Lower Bound	-2.0000
Upper Bound	78.0000

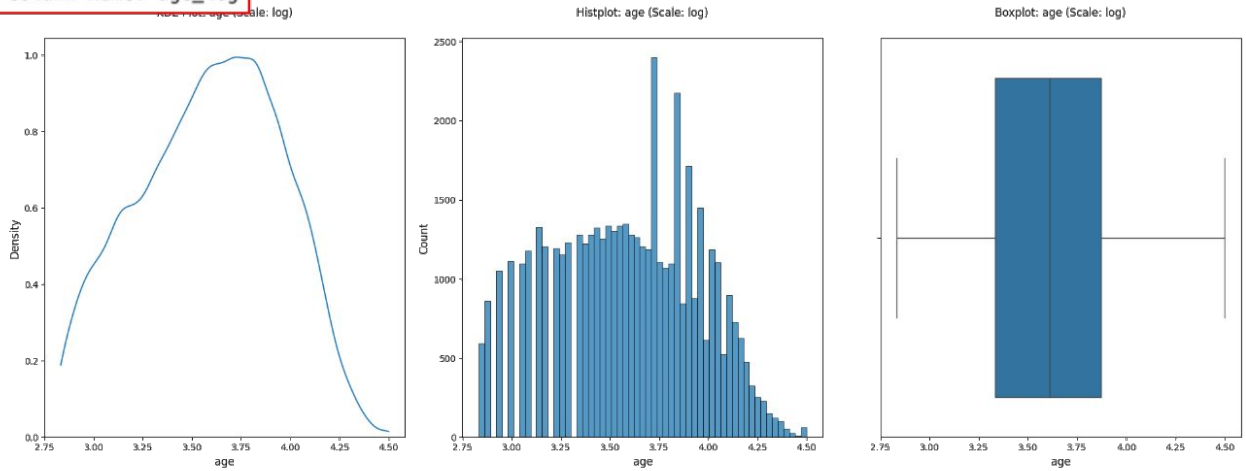


Lower cutoff: 17 | Upper cutoff: 90

```
et.data_doctor(df=X, feature_name='age', scale_conversion='log', apply_as_new_col_to_df=True)
```

```
DATA DOCTOR SUMMARY REPORT
+-----------------------------+---------------------+
| Feature                     | age                 |
+-----------------------------+---------------------+
| Statistic                   | Value               |
+-----------------------------+---------------------+
| Min                         |              2.8332 |
| Max                         |              4.4998 |
| Mean                        |              3.5905 |
| Median                      |              3.6109 |
| Std Dev                     |              0.3617 |
+-----------------------------+---------------------+
| Quartile                    | Value               |
+-----------------------------+---------------------+
| Q1 (25%)                    |              3.3322 |
| Q2 (50% = Median)           |              3.6109 |
| Q3 (75%)                    |              3.8712 |
| IQR                         |              0.5390 |
+-----------------------------+---------------------+
| Outlier Bound               | Value               |
+-----------------------------+---------------------+
| Lower Bound                 |              2.5237 |
| Upper Bound                 |              4.6797 |
+-----------------------------+---------------------+
```

```
New Column Name: age_log
```

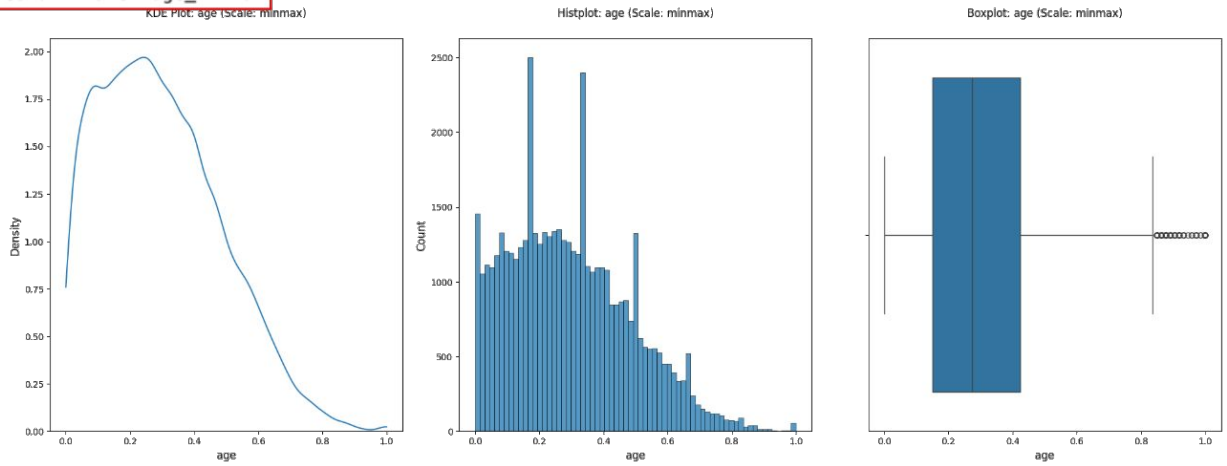


Lower cutoff: 3 | Upper cutoff: 4

```
et.data_doctor(df=X, feature_name='age', scale_conversion='minmax', apply_as_new_col_to_df=True)
```

```
                DATA DOCTOR SUMMARY REPORT
+-------------------------------+-----------------------+
| Feature                       | age                   |
+-------------------------------+-----------------------+
| Statistic                     | Value                 |
+-------------------------------+-----------------------+
| Min                           |                0.0000 |
| Max                           |                1.0000 |
| Mean                          |                0.2965 |
| Median                        |                0.2740 |
| Std Dev                       |                0.1878 |
+-------------------------------+-----------------------+
| Quartile                      | Value                 |
+-------------------------------+-----------------------+
| Q1 (25%)                      |                0.1507 |
| Q2 (50% = Median)             |                0.2740 |
| Q3 (75%)                      |                0.4247 |
| IQR                           |                0.2740 |
+-------------------------------+-----------------------+
| Outlier Bound                 | Value                 |
+-------------------------------+-----------------------+
| Lower Bound                   |               -0.2603 |
| Upper Bound                   |                0.8356 |
+-------------------------------+-----------------------+
```

```
New Column Name: age_minmax
```



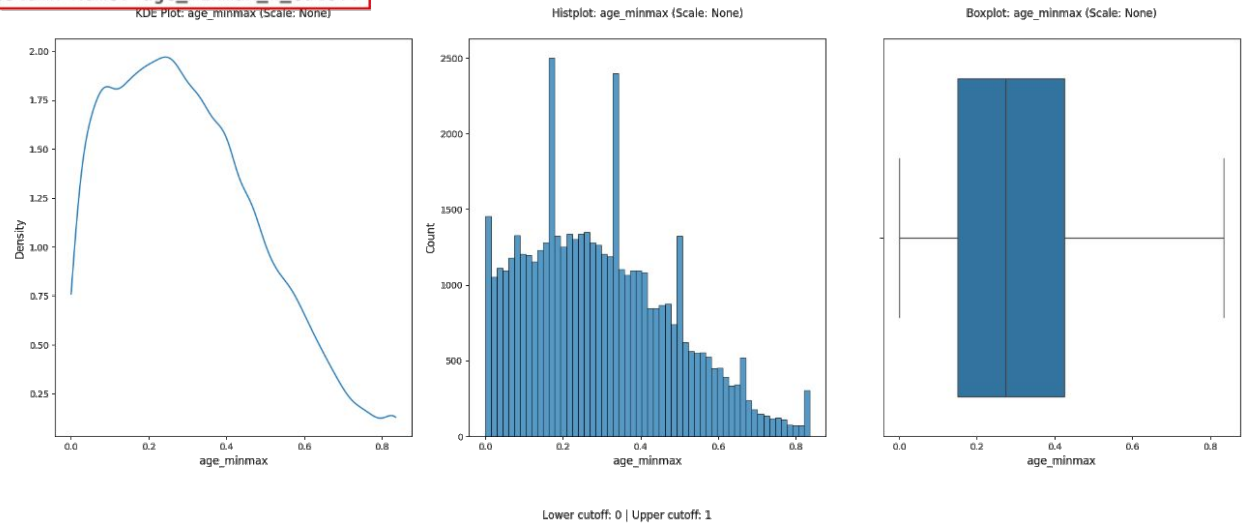

```
Lower cutoff: 0 | Upper cutoff: 1
```

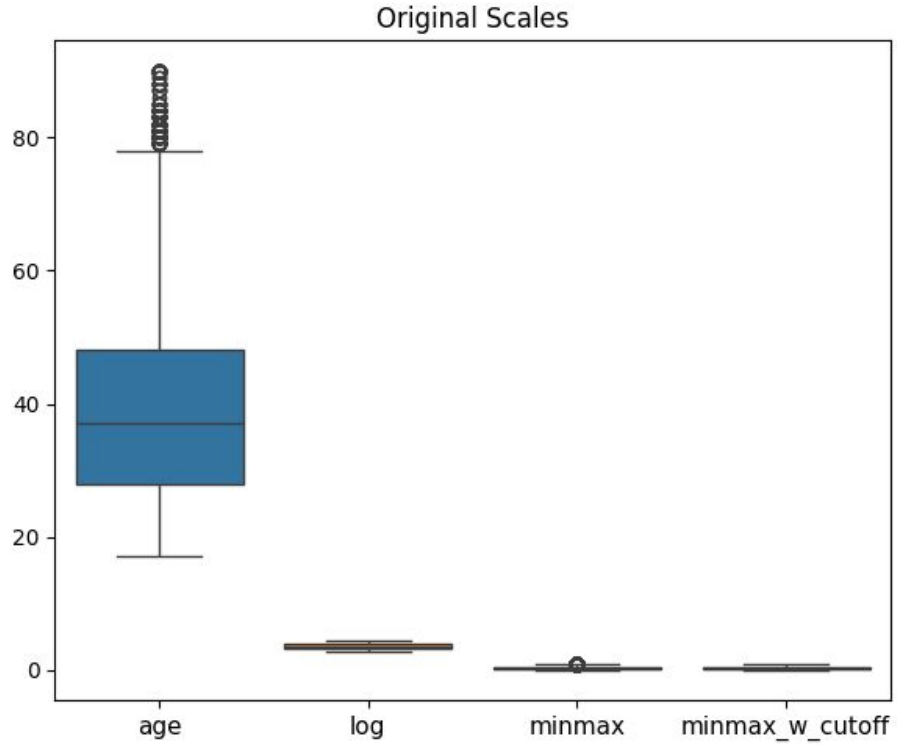
```
et.data_doctor(df=X, feature_name='age_minmax', upper_cutoff=.8356, apply_cutoff=True, apply_as_new_col_to_df=True)
```

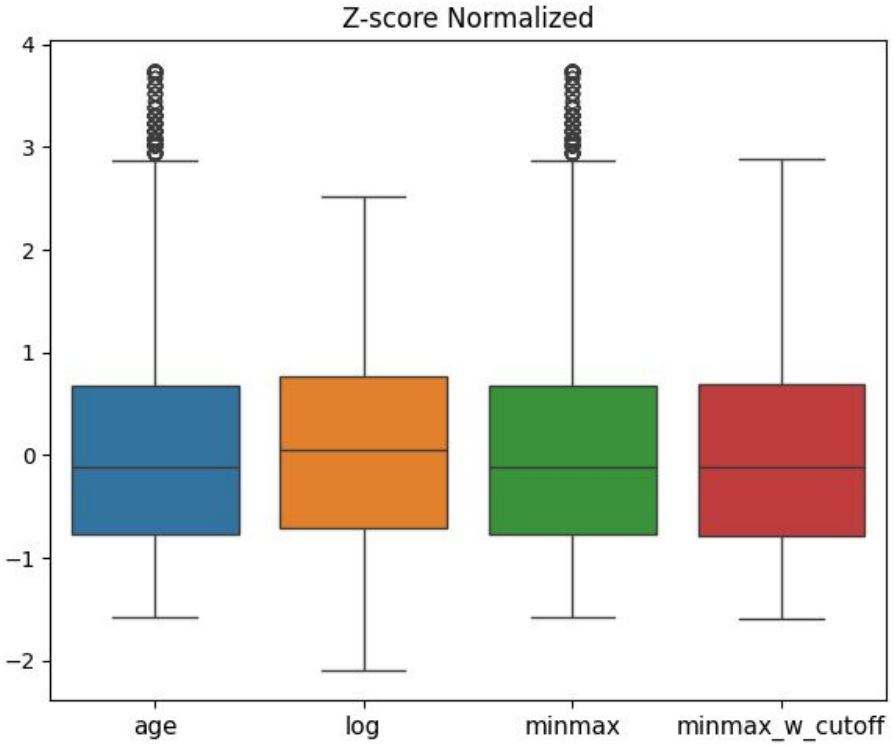
DATA DOCTOR SUMMARY REPORT

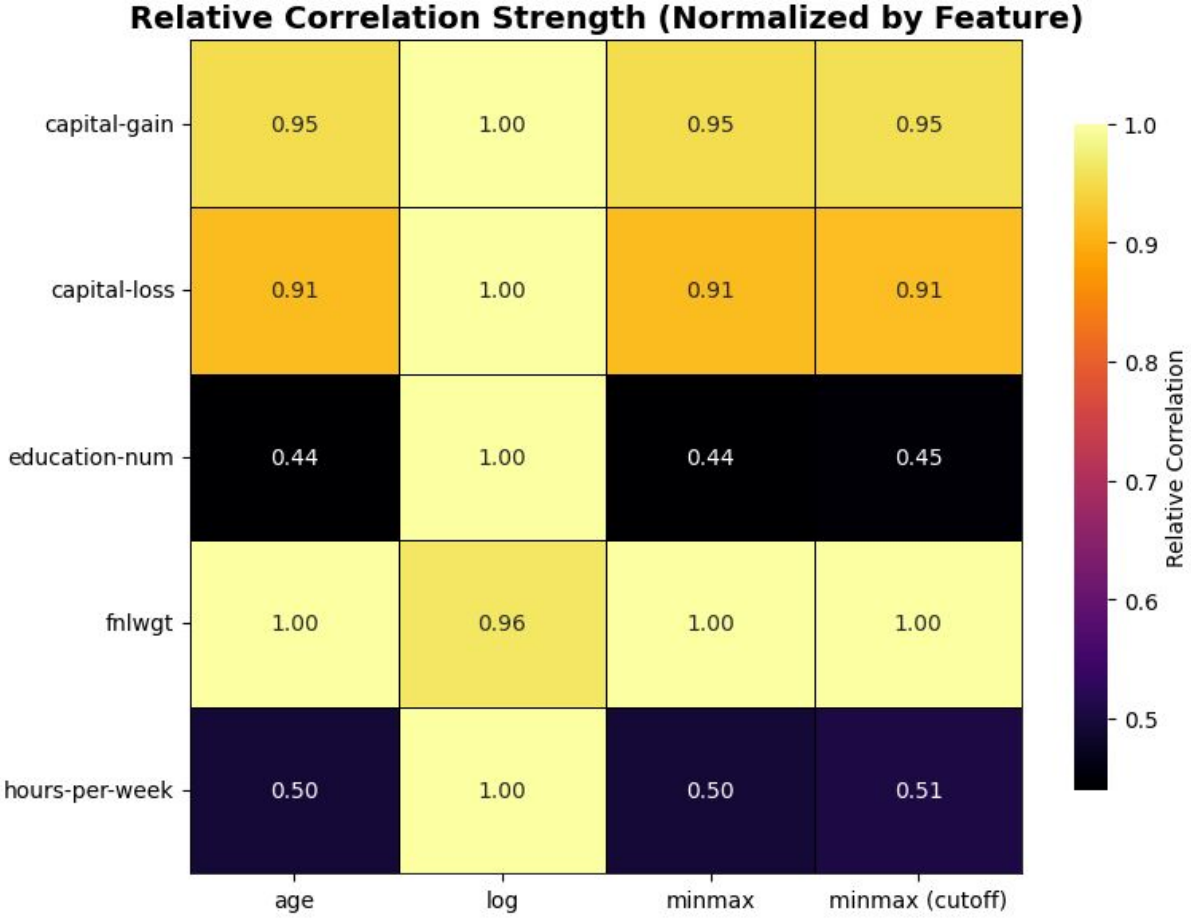
Feature	age_minmax
Statistic	**Value**
Min	0.0000
Max	0.8356
Mean	0.2961
Median	0.2740
Std Dev	0.1867
Quartile	**Value**
Q1 (25%)	0.1507
Q2 (50% = Median)	0.2740
Q3 (75%)	0.4247
IQR	0.2740
Outlier Bound	**Value**
Lower Bound	-0.2603
Upper Bound	0.8356

```
New Column Name: age_minmax_w_cutoff
```




Original Scales
80
60
40
20
0
age
log
minmax
minmax_w_cutoff

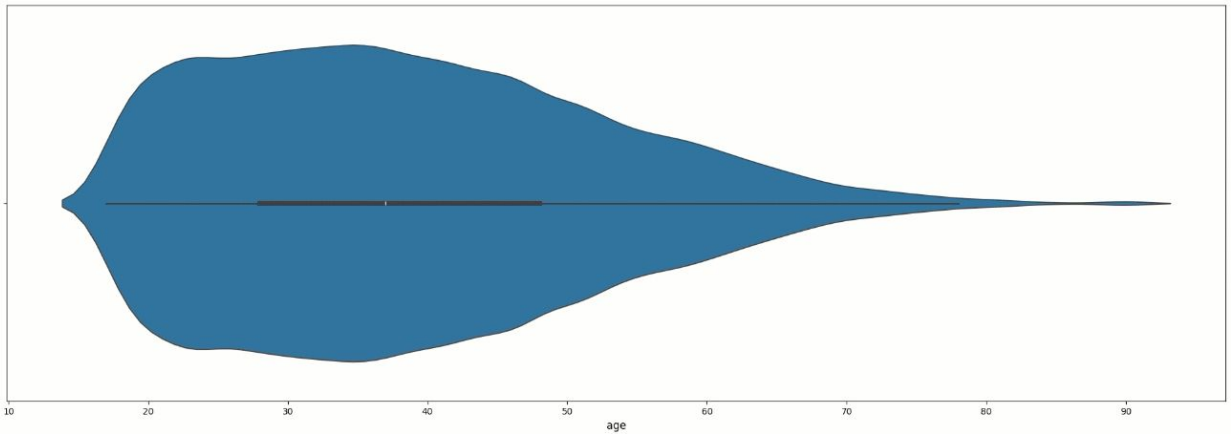

Z-score Normalized
4
3
2
1
0
−1
−2
age
log
minmax
minmax_w_cutoff


Relative Correlation Strength (Normalized by Feature)
capital-gain
capital-loss
education-num
fnlwgt
hours-per-week
0.95
1.00
0.95
0.95
0.91
1.00
0.91
0.91
0.44
1.00
0.44
0.45
1.00
0.96
1.00
1.00
0.50
1.00
0.50
0.51
age
log
minmax
minmax (cutoff)
Relative Correlation
1.0
0.9
0.8
0.7
0.6
0.5

```
et.data_doctor(df=X, feature_name='age', plot_type='box_violin', box_violin='violinplot')
```

```
            DATA DOCTOR SUMMARY REPORT
+------------------------------+--------------------+
| Feature                      | age                |
+------------------------------+--------------------+
| Statistic                    | Value              |
+------------------------------+--------------------+
| Min                          |            17.0000 |
| Max                          |            90.0000 |
| Mean                         |            38.6436 |
| Median                       |            37.0000 |
| Std Dev                      |            13.7104 |
+------------------------------+--------------------+
| Quartile                     | Value              |
+------------------------------+--------------------+
| Q1 (25%)                     |            28.0000 |
| Q2 (50% = Median)            |            37.0000 |
| Q3 (75%)                     |            48.0000 |
| IQR                          |            20.0000 |
+------------------------------+--------------------+
| Outlier Bound                | Value              |
+------------------------------+--------------------+
| Lower Bound                  |            -2.0000 |
| Upper Bound                  |            78.0000 |
+------------------------------+--------------------+
```

Violinplot: age (Scale: None)

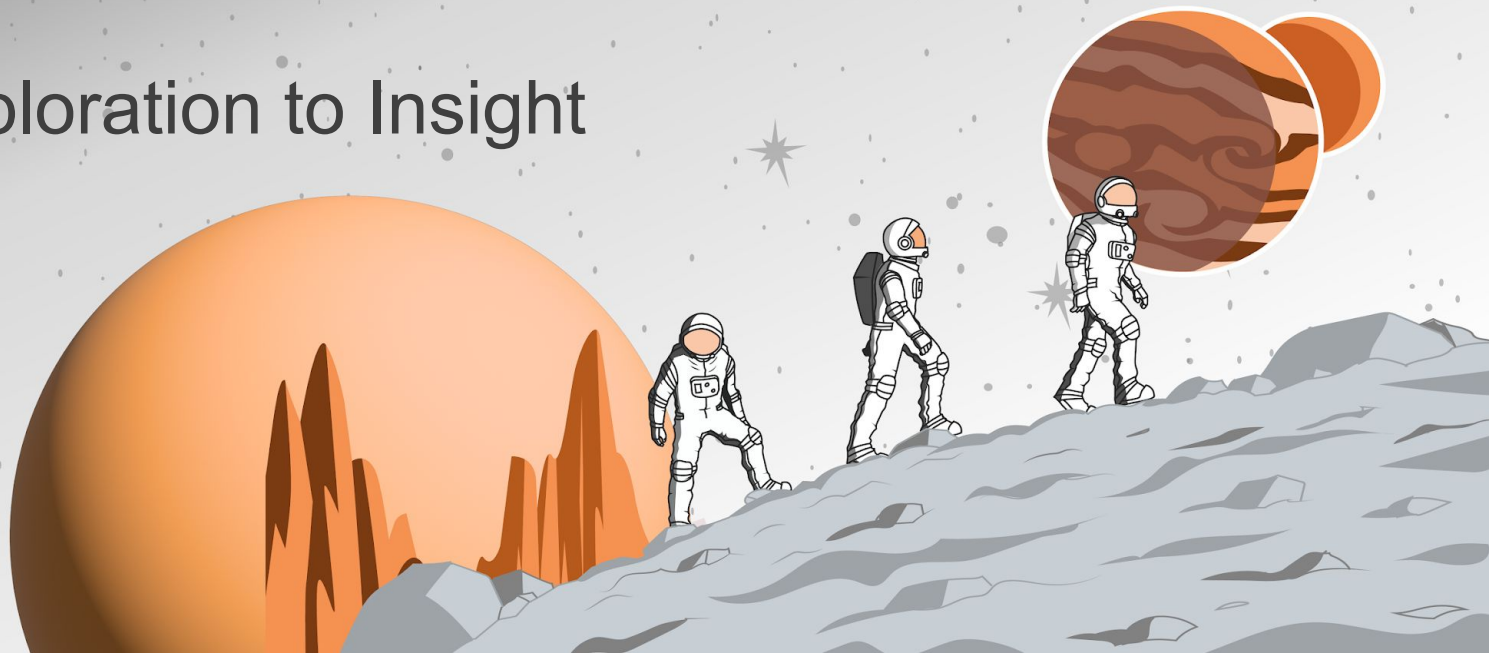


Lower cutoff: 17 | Upper cutoff: 90

jupytercon

In Conclusion

From Exploration to Insight



#JupyterCon

Thank you to:

Dr. Ebrahim Tarshizi
Academic Director, MS in Applied Data Science (MSADS)
University of San Diego

UCLA Health, UCLA CTSI (with gratitude to Alex Bui, PhD, Panayiotis Petousis, PhD, and Arthur Funnell)

University of California, Riverside

JupyterCon
for the opportunity to share our work with the open-source data science community.

#JupyterCon

