```
%pip install ydata-profiling nbconvert
```

```
import pandas as pd
import numpy as np
import kagglehub
import seaborn as sns
import matplotlib.pyplot as plt
from ydata_profiling import ProfileReport
from sklearn.preprocessing import RobustScaler, MinMaxScaler
```

Upgrade to ydata-sdk

Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.

```
# Downloading the latest dataset from Kaggle
path = kagglehub.dataset_download("mlg-ulb/creditcardfraud")

print("Path to dataset files:", path)
```

```
Using Colab cache for faster access to the 'creditcardfraud' dataset.
Path to dataset files: /kaggle/input/creditcardfraud
```

```
df = pd.read_csv(path+'/creditcard.csv')
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

```
df.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V2
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+0
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	1.654067e-16	-3.568593e-16	2.578648e-16	4.473266e-1
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01	7.257016e-01	6.244603e-01	6.056471e-0
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836627e+0
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545861e-0
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02	6.781943e-03	-1.119293e-02	4.097606e-0
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01	5.285536e-01	1.476421e-01	4.395266e-0
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01	1.050309e+01	2.252841e+01	4.584549e+0

8 rows × 31 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

```
report = ProfileReport(df)
report
```

```
Summarize dataset: 100%                                    940/940 [02:40<00:00,  1.37it/s, Completed]

  0%|          | 0/31 [00:00<?, ?it/s]
  3%|▏         | 1/31 [00:00<00:16,  1.81it/s]
  6%|▌         | 2/31 [00:00<00:10,  2.80it/s]
 10%|▉         | 3/31 [00:01<00:13,  2.06it/s]
 13%|█▏        | 4/31 [00:01<00:09,  2.71it/s]
 16%|█▌        | 5/31 [00:02<00:15,  1.68it/s]
 19%|█▊        | 6/31 [00:03<00:13,  1.82it/s]
 23%|██▏       | 7/31 [00:03<00:14,  1.69it/s]
 26%|██▌       | 8/31 [00:04<00:11,  1.95it/s]
 29%|██▉       | 9/31 [00:04<00:11,  1.89it/s]
 32%|███▏      | 10/31 [00:04<00:09,  2.12it/s]
 35%|███▌      | 11/31 [00:05<00:10,  1.98it/s]
 39%|███▊      | 12/31 [00:05<00:08,  2.31it/s]
 42%|████▏     | 13/31 [00:06<00:09,  1.93it/s]
 48%|████▊     | 15/31 [00:07<00:07,  2.17it/s]
 52%|█████▏    | 16/31 [00:07<00:05,  2.69it/s]
 55%|█████▍    | 17/31 [00:08<00:06,  2.04it/s]
 61%|██████    | 19/31 [00:09<00:05,  2.22it/s]
 65%|██████▍   | 20/31 [00:09<00:04,  2.71it/s]
 68%|██████▊   | 21/31 [00:10<00:04,  2.05it/s]
 71%|███████   | 22/31 [00:10<00:03,  2.46it/s]
 74%|███████▍  | 23/31 [00:10<00:03,  2.04it/s]
 77%|███████▋  | 24/31 [00:11<00:02,  2.43it/s]
 81%|████████  | 25/31 [00:11<00:02,  2.07it/s]
 84%|████████▍ | 26/31 [00:12<00:02,  2.44it/s]
 87%|████████▋ | 27/31 [00:12<00:01,  2.13it/s]
 90%|█████████ | 28/31 [00:12<00:01,  2.34it/s]
 94%|█████████▍| 29/31 [00:13<00:00,  2.40it/s]
100%|██████████| 31/31 [00:13<00:00,  2.27it/s]
Generate report structure: 100%                            1/1 [00:09<00:00,  9.07s/it]

Render HTML: 100%                                          1/1 [00:05<00:00,  5.94s/it]
```

YData Profiling Report

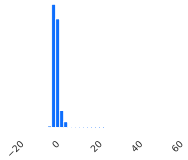
Mean	$1.0178873 \times 10^{-15}$	**Memory size**	2.2 MiB

More details

V6

Real number (ℝ)

Distinct	275663	**Minimum**	-26.160506
Distinct (%)	96.8%	**Maximum**	73.301626
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	176633
Infinite (%)	0.0%	**Negative (%)**	62.0%
Mean	$1.4976915 \times 10^{-15}$	**Memory size**	2.2 MiB



More details

V7

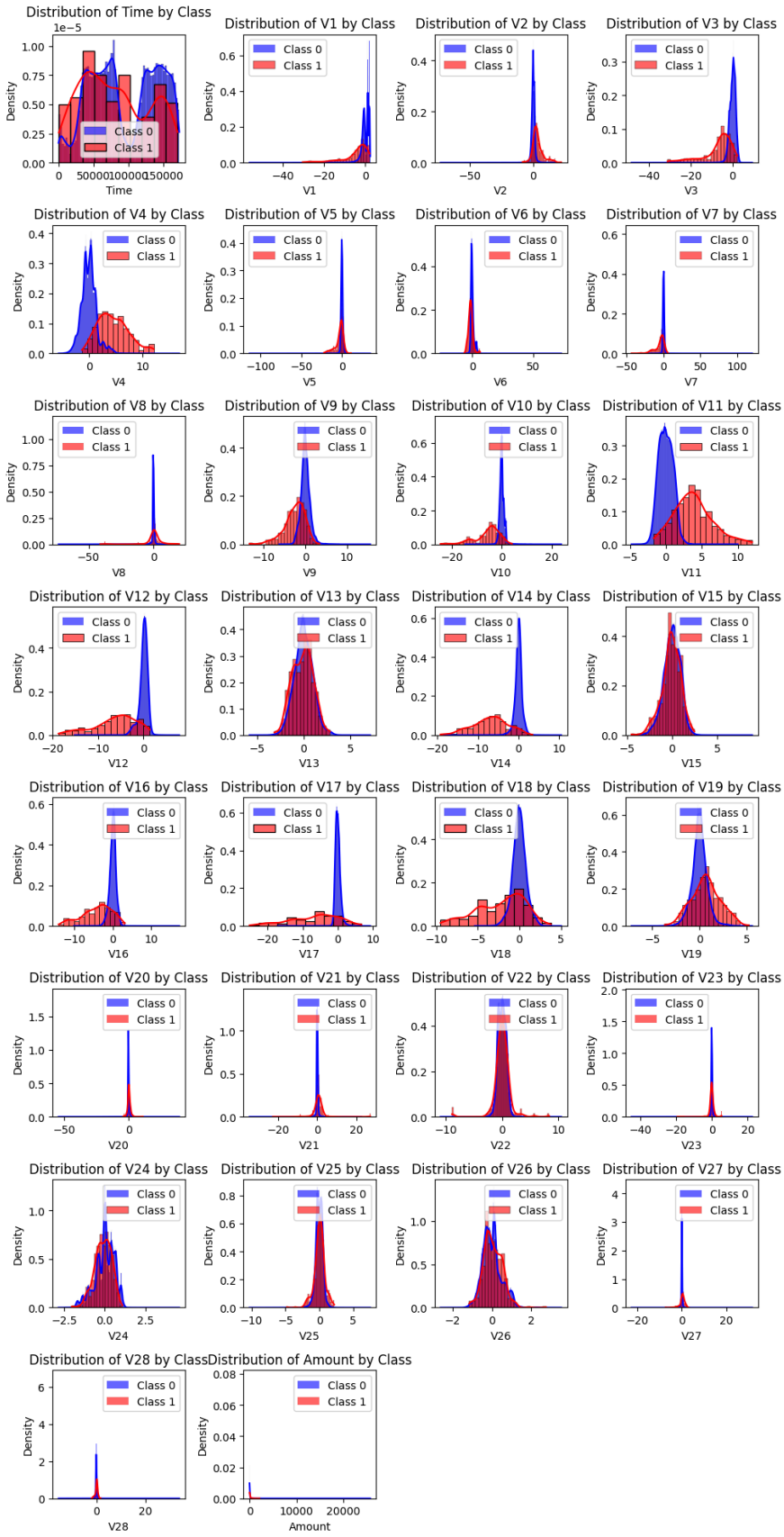
Real number (ℝ)

Distinct	275663	**Minimum**	-43.557242


```
1 report.to_file('report.html')
```

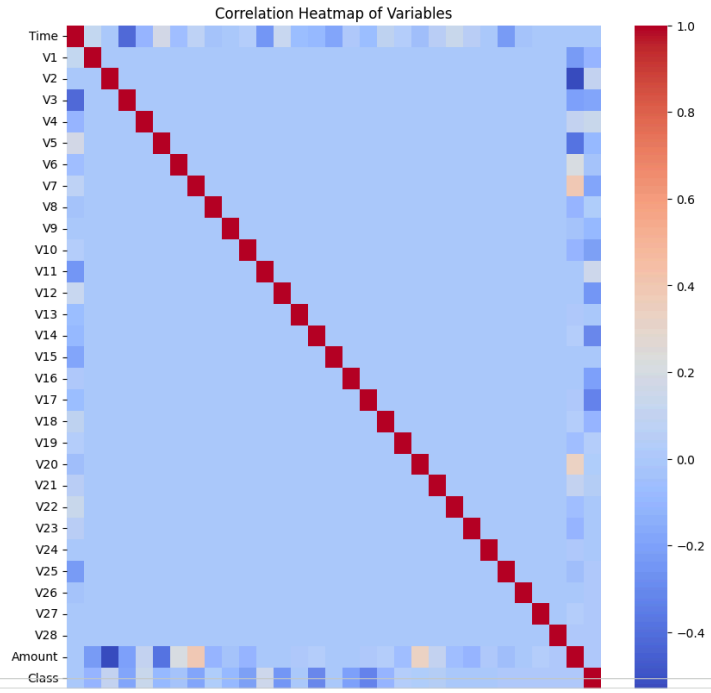
```
Export report to file: 100%                                1/1 [00:00<00:00,  4.22it/s]
```

```
 1 import matplotlib.pyplot as plt
 2 import seaborn as sns
 3
 4 # Separate data by class
 5 df_class_0 = df[df['Class'] == 0]
 6 df_class_1 = df[df['Class'] == 1]
 7
 8 # Get column names, excluding 'Class' itself for the histograms
 9 columns_to_plot = [col for col in df.columns if col != 'Class']
10
11 # Determine number of rows and columns for subplots dynamically
12 num_cols = 4
13 num_rows = (len(columns_to_plot) + num_cols - 1) // num_cols
14
15 fig, axes = plt.subplots(num_rows, num_cols, figsize=(10, 2.5 * num_rows))
16 axes = axes.flatten() # Flatten the 2D array of axes for easy iteration
17
18 for i, col in enumerate(columns_to_plot):
19     if i < len(axes): # Ensure we don't go out of bounds
20         sns.histplot(df_class_0[col], ax=axes[i], color='blue', label='Class 0', kde=True, stat='density', alpha=0.6)
21         sns.histplot(df_class_1[col], ax=axes[i], color='red', label='Class 1', kde=True, stat='density', alpha=0.6)
22         axes[i].set_title(f'Distribution of {col} by Class')
23         axes[i].legend()
24
25 # Hide any unused subplots
26 for j in range(i + 1, len(axes)):
27     fig.delaxes(axes[j])
28
29 plt.tight_layout()
30 plt.show()
```




```
1 Start coding or generate with AI.
```

```
1 # Correlation Heatmap to visualize
2 plt.figure(figsize=(10,10))
3 sns.heatmap(df.corr(), annot=False, cmap='coolwarm')
4 plt.title('Correlation Heatmap of Variables')
5 plt.show()
```

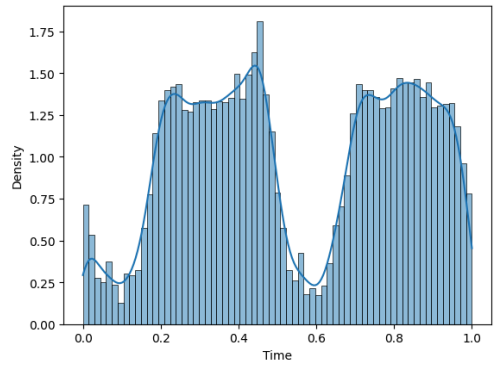



```
# Scaling Amount and Time column to normalize the scale

new_df = df.copy()
new_df['Amount'] = RobustScaler().fit_transform(new_df['Amount'].to_numpy().reshape(-1,1))
time = new_df['Time']
new_df['Time'] = (time - time.min())/(time.max()-time.min())
sns.histplot(new_df['Time'], kde=True, stat='density')

```

```
<Axes: xlabel='Time', ylabel='Density'>
```




```
new_df.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V2
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+0
mean	0.548717	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	1.654067e-16	-3.568593e-16	2.578648e-16	4.473266e-1
std	0.274828	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01	7.257016e-01	6.244603e-01	6.056471e-0
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836627e+0
25%	0.313681	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545861e-0
50%	0.490138	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02	6.781943e-03	-1.119293e-02	4.097606e-0
75%	0.806290	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01	5.285536e-01	1.476421e-01	4.395266e-0
max	1.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01	1.050309e+01	2.252841e+01	4.584549e+0

8 rows × 31 columns

```
Start coding or generate with AI.
```