Create Synthetic data

For generate the data we gonna use the SDV or Synthetic Data Vault. SVD generates synthetic data by applying mathematical techniques and machine learning models such as the deep learning model.

Even if the data contain multiple data types and missing data, SDV will handle it, so we only need to provide the data.

To use the SDMetrics library, you'll need:

-Real data, loaded as a pandas DataFrame
-Synthetic data, loaded as a pandas DataFrame
-Metadata, represented as a dictionary format

```
! pip install sdv
```

```
import pandas as pd
import os
import numpy as np
from sdv.tabular import CTGAN, GaussianCopula, CopulaGAN
from sdv.evaluation import evaluate
from sdmetrics.reports.single_table import QualityReport
from sdmetrics.reports.utils import get_column_plot
import warnings
warnings.filterwarnings('ignore')
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
data_drive_path = os.path.join('drive', 'MyDrive', 'Colab Notebooks', 'SP-project')
```

```
infringement_path = os.path.join(data_drive_path, 'dataset', 'infringement_dataset.csv')
```

Load data

We will choose only 1000 sample of the original data to generate sintetic data since the dataset is too large

```
data = pd.read_csv(infringement_path)
data = data.sample(int(data.shape[0]/10))
```

```
to_drop=["address",
"appendix_a",
"appendix_b",
"appendix_c",
"appendix_d",
"appendix_e",
"appendix_f",
"appendix_g",
"appendix_h",
"appendix_i",
"appendix_j",
"appendix_k",
"appendix_l",
"appendix_m",
"appendix_n",
"appendix_o",
"appendix_p",
"appendix_q",
"appendix_r",
"appendix_s",
"appendix_t",
"car_age",
"first_name",
"last_name",
"num_req_bureau_day",
```

```
"num_req_bureau_hour",
"num_req_bureau_month",
"num_req_bureau_qrt",
"num_req_bureau_week",
"num_req_bureau_year",
"provided_email",
"provided_homephone",
"provided_mobilephone",
"provided_workphone",
"region_rating",
"score_ext_1",
"score_ext_2",
"score_ext_3"]

data = data.drop(to_drop, axis=1)
data
```

	loan_id	infringed	contract_type	gender	has_own_car	has_own_realty	nu
283559	428406	0	Cash loans	F	Y	Y	
126426	246616	0	Cash loans	F	N	Y	
16138	118822	0	Cash loans	F	Y	N	
219598	354404	0	Cash loans	F	N	N	
246579	385372	1	Cash loans	M	N	N	
...	...	...	...	...	...	...	
192961	323769	0	Cash loans	F	N	N	
192	100224	0	Cash loans	F	Y	Y	
218401	353036	0	Cash loans	F	N	N	
24236	128192	0	Cash loans	M	N	Y	
147424	270932	0	Cash loans	F	N	Y	

30751 rows × 31 columns

Create and train model - CTGAN

```
%%time
model_ctgan = CTGAN(epochs=15, generator_dim=(256, 256), discriminator_dim=(256, 256), verbose=True)
model_ctgan.fit(data)
```

```
Epoch 1, Loss G:  0.2502,Loss D: -0.0510
Epoch 2, Loss G: -0.1003,Loss D: -0.0557
Epoch 3, Loss G: -0.2420,Loss D:  0.0630
Epoch 4, Loss G: -0.4728,Loss D:  0.0130
Epoch 5, Loss G: -0.9434,Loss D:  0.0150
Epoch 6, Loss G: -0.9446,Loss D:  0.0335
Epoch 7, Loss G: -0.7393,Loss D: -0.4376
Epoch 8, Loss G: -1.1872,Loss D: -0.4655
Epoch 9, Loss G:  0.2402,Loss D: -0.3041
Epoch 10, Loss G: -0.0918,Loss D: -0.8371
Epoch 11, Loss G: -0.5632,Loss D: -0.7223
Epoch 12, Loss G: -0.8689,Loss D: -0.6275
Epoch 13, Loss G: -1.3744,Loss D: -0.8704
Epoch 14, Loss G: -1.7406,Loss D: -0.0588
Epoch 15, Loss G: -2.2546,Loss D: -0.4588
CPU times: user 1min 34s, sys: 35.6 s, total: 2min 9s
Wall time: 2min 7s
```

After fitting the model we gonna use it to generate the new data

```
%%time
synthetic_data_ctgan = model_ctgan.sample(num_rows=data.shape[0])
synthetic_data_ctgan
```

```
CPU times: user 1.89 s, sys: 52.3 ms, total: 1.94 s
Wall time: 1.95 s
```

	loan_id	infringed	contract_type	gender	has_own_car	has_own_realty	num
0	390548	0	Cash loans	F	N	Y	
1	330113	0	Cash loans	M	N	Y	
2	297794	0	Cash loans	M	Y	Y	
3	404005	0	Cash loans	F	N	Y	
4	333130	0	Cash loans	M	Y	Y	
...	...	...	...	...	...	...	
30746	220406	0	Cash loans	F	Y	N	
30747	350135	1	Cash loans	M	N	Y	
30748	263321	0	Cash loans	M	N	Y	
30749	456247	0	Cash loans	F	N	Y	
30750	356989	0	Cash loans	M	Y	N	

30751 rows × 31 columns

```
save_path = os.path.join(data_drive_path, 'dataset', 'synthetic_data_CTGAN.csv')
synthetic_data_ctgan.to_csv(save_path, index=False)
```

Evaluate results

```
model_score = evaluate(synthetic_data_ctgan, data)
model_score
```

```
0.9259618593516252
```

Computing performance score

This report evaluates the shapes of the columns (marginal distributions) and the pairwise trends between the columns (correlations).

```
report_ctgan = QualityReport()

report_ctgan.generate(data, synthetic_data_ctgan,model_ctgan.get_metadata().to_dict())
```

```
Creating report: 100%|██████████| 4/4 [00:04<00:00,  1.23s/it]

Overall Quality Score: 89.61%

Properties:
Column Shapes: 90.29%
Column Pair Trends: 88.92%
```

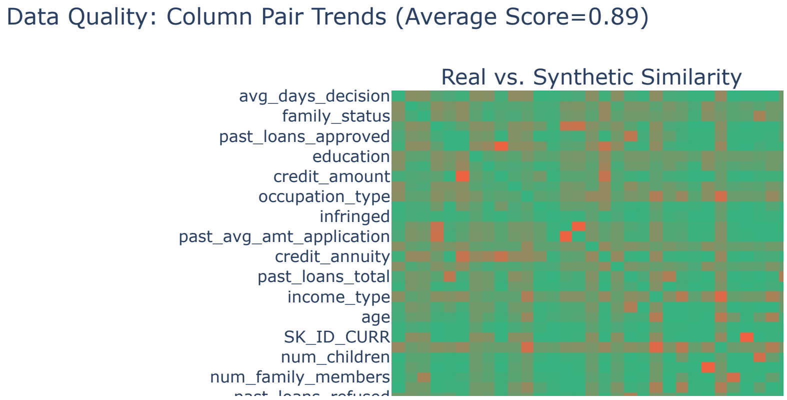
```
details = report_ctgan.get_details(property_name='Column Shapes')
details
```

	Column	Metric	Quality Score
0	loan_id	KSComplement	0.940750
1	infringed	KSComplement	0.983383
2	num_children	KSComplement	0.934474
3	annual_income	KSComplement	0.922799
4	credit_amount	KSComplement	0.905206
5	credit_annuity	KSComplement	0.728931
6	goods_valuation	KSComplement	0.690744
7	age	KSComplement	0.912653
8	days_employed	KSComplement	0.831485
9	mobilephone_reachable	KSComplement	0.975871
10	num_family_members	KSComplement	0.940425
11	SK_ID_CURR	KSComplement	0.897734
12	avg_days_decision	KSComplement	0.851140
13	past_avg_amount_annuity	KSComplement	0.795955
14	past_avg_amt_application	KSComplement	0.930968
15	past_avg_amt_credit	KSComplement	0.909461
16	past_loans_approved	KSComplement	0.947825
17	past_loans_refused	KSComplement	0.915932
18	past_loans_canceled	KSComplement	0.917256
19	past_loans_unused	KSComplement	0.968184

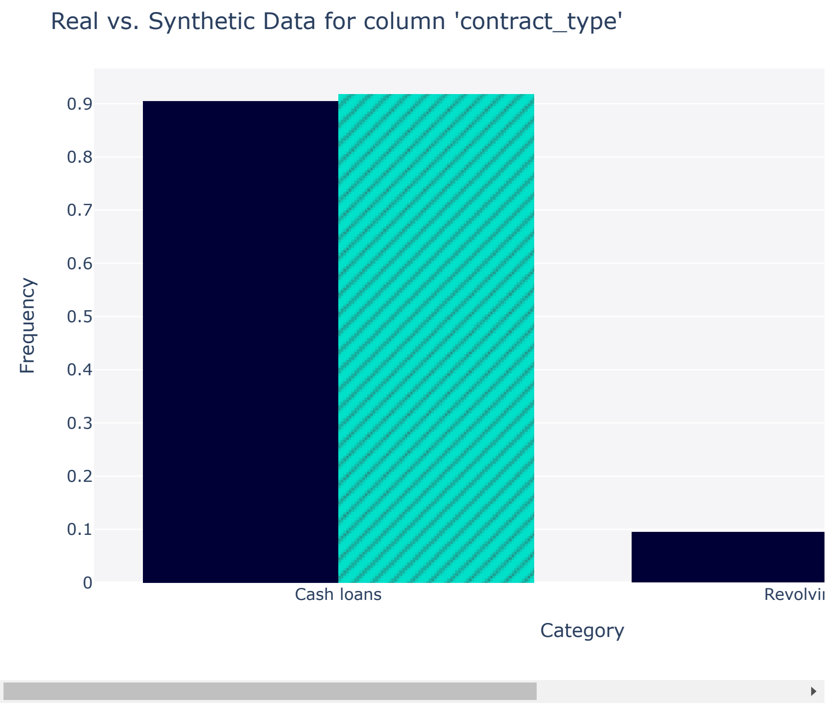
```
print('Column with more quality',details[details['Quality Score'] == details['Quality Score'].max()]['Column'], details['
print('Column with less quality',details[details['Quality Score'] == details['Quality Score'].min()]['Column'], details['
```

```
Column with more quality 21    contract_type
Name: Column, dtype: object 0.988000390231212
Column with less quality 6    goods_valuation
Name: Column, dtype: object 0.6907437225471798
```

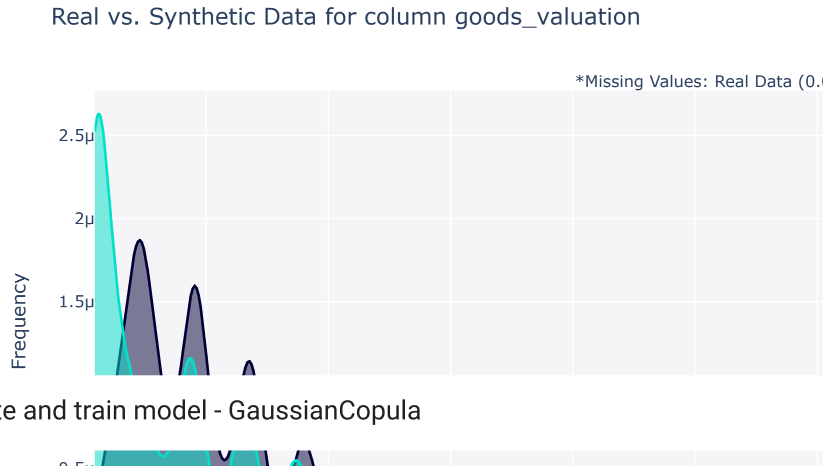
```
report_ctgan.get_visualization(property_name='Column Pair Trends')
```




```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_ctgan,
    metadata=model_ctgan.get_metadata().to_dict(),
    column_name='contract_type'
)

fig.show()
```




```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_ctgan,
    metadata=model_ctgan.get_metadata().to_dict(),
    column_name='goods_valuation'
)

fig.show()
```



Create and train model - GaussianCopula

```
%%time
model_gauscopula = GaussianCopula()
model_gauscopula.fit(data)
```

```
CPU times: user 3.17 s, sys: 50.3 ms, total: 3.22 s
Wall time: 3.23 s
```

After fitting the model we gonna use it to generate the new data

```
%%time
synthetic_data_gausscopula = model_gauscopula.sample(num_rows=data.shape[0])
synthetic_data_gausscopula
```

```
CPU times: user 1.01 s, sys: 142 ms, total: 1.15 s
Wall time: 1.03 s
```

	loan_id	infringed	contract_type	gender	has_own_car	has_own_realty	num
0	193504	0	Cash loans	F	N	Y	
1	365977	0	Cash loans	M	Y	Y	
2	198816	0	Cash loans	F	N	Y	
3	422608	0	Cash loans	F	N	Y	
4	329338	0	Cash loans	F	N	Y	
...	...	...	...	...	...	...	
30746	159078	0	Cash loans	F	Y	Y	
30747	415951	0	Cash loans	F	N	Y	
30748	282187	0	Cash loans	M	Y	Y	
30749	407171	1	Cash loans	M	N	Y	
30750	153332	0	Cash loans	F	N	N	

30751 rows × 31 columns

```
save_path = os.path.join(data_drive_path, 'dataset', 'synthetic_data_GaussianCopula.csv')
synthetic_data_gausscopula.to_csv(save_path, index=False)
```

Evaluate results

```
model_score_gauss = evaluate(synthetic_data_gausscopula, data)
model_score_gauss
```

```
0.9015870799206975
```

Computing performance score

This report evaluates the shapes of the columns (marginal distributions) and the pairwise trends between the columns (correlations).

```
report_gausscopula = QualityReport()

report_gausscopula.generate(data, synthetic_data_gausscopula,model_gauscopula.get_metadata().to_dict())
```

```
Creating report: 100%|██████████| 4/4 [00:04<00:00,  1.18s/it]

Overall Quality Score: 90.16%

Properties:
Column Shapes: 88.11%
Column Pair Trends: 92.21%
```

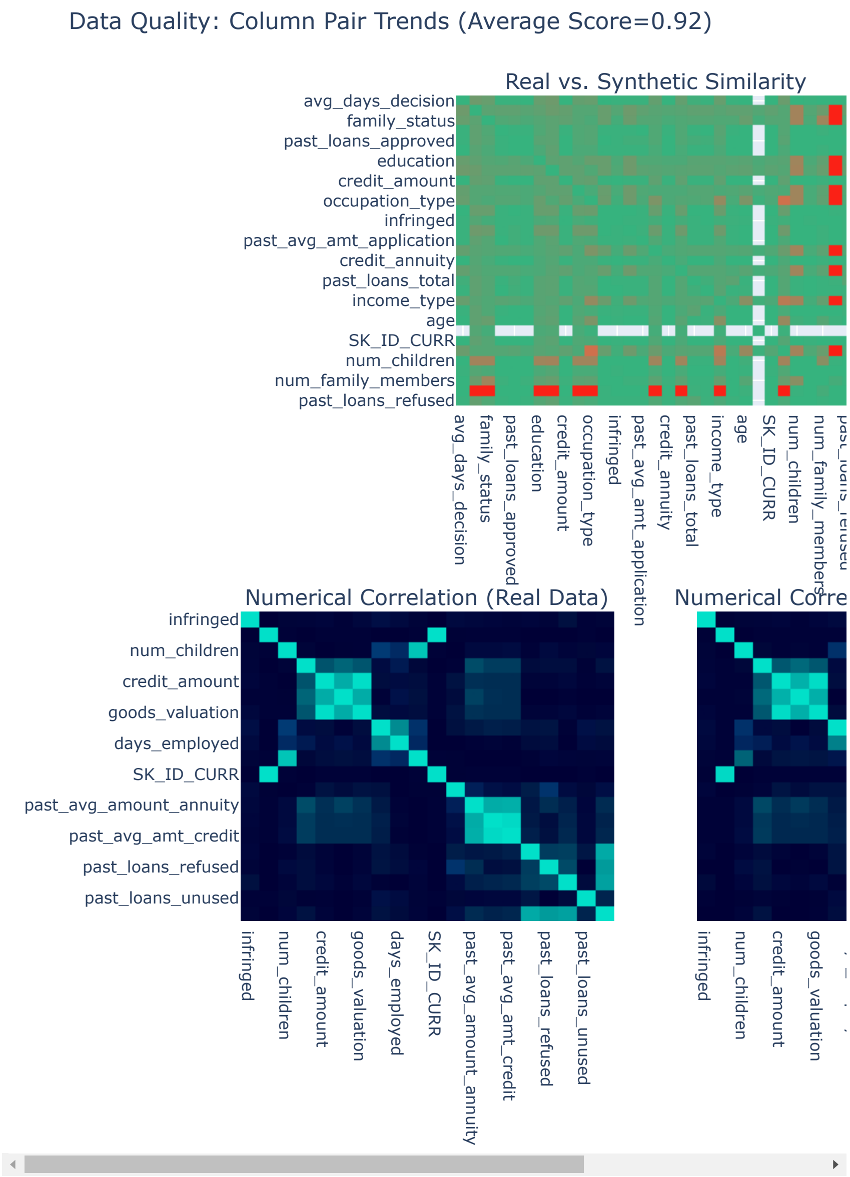
```
details_gausscopula = report_gausscopula.get_details(property_name='Column Shapes')
details
```

	Column	Metric	Quality Score
0	loan_id	KSComplement	0.940750
1	infringed	KSComplement	0.983383
2	num_children	KSComplement	0.934474
3	annual_income	KSComplement	0.922799
4	credit_amount	KSComplement	0.905206
5	credit_annuity	KSComplement	0.728931
6	goods_valuation	KSComplement	0.690744
7	age	KSComplement	0.912653
8	days_employed	KSComplement	0.831485
9	mobilephone_reachable	KSComplement	0.975871
10	num_family_members	KSComplement	0.940425
11	SK_ID_CURR	KSComplement	0.897734
12	avg_days_decision	KSComplement	0.851140
13	past_avg_amount_annuity	KSComplement	0.795955
14	past_avg_amt_application	KSComplement	0.930968
15	past_avg_amt_credit	KSComplement	0.909461
16	past_loans_approved	KSComplement	0.947825
17	past_loans_refused	KSComplement	0.915932
18	past_loans_canceled	KSComplement	0.917256
19	past_loans_unused	KSComplement	0.968184
20	past_loans_total	KSComplement	0.779650
21	contract_type	TVComplement	0.988000
22	gender	TVComplement	0.903775
23	has_own_car	TVComplement	0.948099
24	has_own_realty	TVComplement	0.954148
25	income_type	TVComplement	0.902865
26	education	TVComplement	0.908003
27	family_status	TVComplement	0.968391
28	housing_type	TVComplement	0.953010
29	occupation_type	TVComplement	0.900496
30	organization_type	TVComplement	0.881825

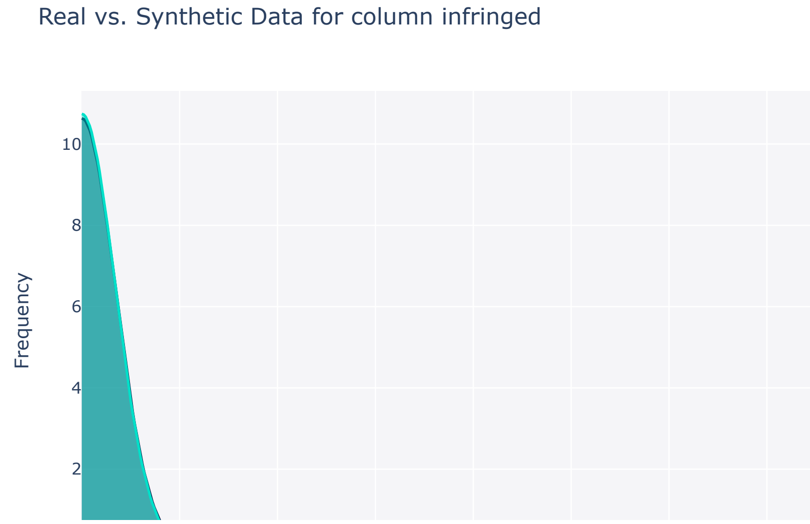
```
print('Column with more quality',details_gausscopula[details_gausscopula['Quality Score'] == details_gausscopula['Quality Score'].max()]['Col
print('Column with less quality',details_gausscopula[details_gausscopula['Quality Score'] == details_gausscopula['Quality Score'].min()]['Col
```

```
Column with more quality 1                infringed
9    mobilephone_reachable
Name: Column, dtype: object 0.9984715944196937
Column with less quality 8    days_employed
Name: Column, dtype: object 0.2606094110760625
```

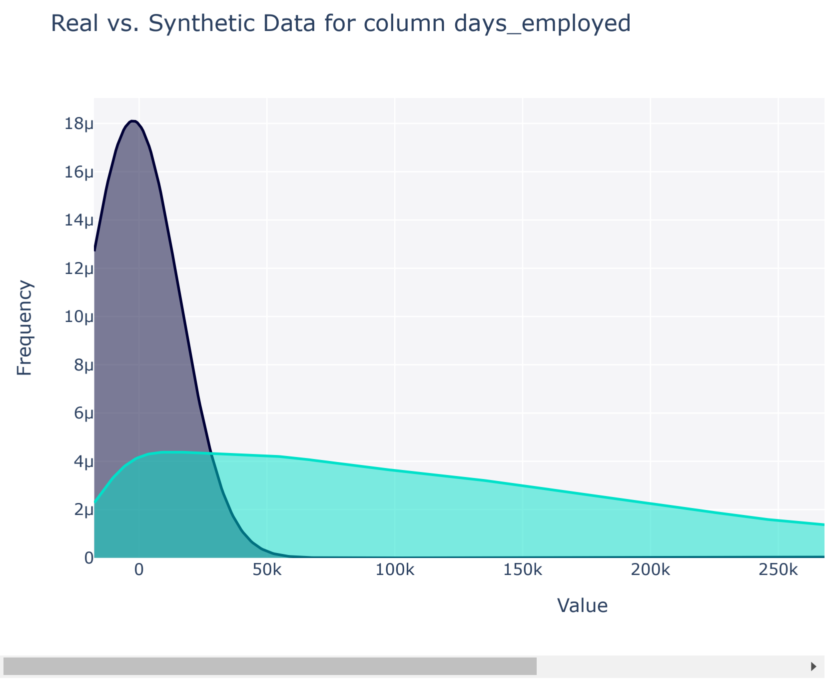
```
report_gausscopula.get_visualization(property_name='Column Pair Trends')
```




```
from sdmetrics.reports.utils import get_column_plot

fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_gausscopula,
    metadata=model_gauscopula.get_metadata().to_dict(),
    column_name='infringed'
)

fig.show()
```




```
from sdmetrics.reports.utils import get_column_plot

fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_gausscopula,
    metadata=model_gauscopula.get_metadata().to_dict(),
    column_name='days_employed'
)

fig.show()
```



Create and train model - CopulaGAN

```
%%time
model_copulagan = CopulaGAN(epochs=15, generator_dim=(256, 256), discriminator_dim=(256, 256), verbose=True)
model_copulagan.fit(data)
```

```
Epoch 1, Loss G:  0.9265,Loss D:  0.0935
Epoch 2, Loss G:  0.3005,Loss D:  0.0339
Epoch 3, Loss G:  0.0527,Loss D:  0.2966
Epoch 4, Loss G:  0.2026,Loss D: -0.1428
Epoch 5, Loss G: -0.4846,Loss D: -0.0646
Epoch 6, Loss G: -0.1811,Loss D: -0.1495
Epoch 7, Loss G:  0.3966,Loss D: -0.8832
```

```
Epoch 8, Loss G:  0.5739,Loss D: -1.0648
Epoch 9, Loss G:  0.3714,Loss D: -0.8372
Epoch 10, Loss G: -0.5608,Loss D: -0.7625
Epoch 11, Loss G:  0.3631,Loss D: -0.2006
Epoch 12, Loss G: -0.9812,Loss D:  0.0280
Epoch 13, Loss G: -1.0042,Loss D: -0.3290
Epoch 14, Loss G: -1.6039,Loss D: -0.2833
Epoch 15, Loss G: -1.7142,Loss D:  0.1150
CPU times: user 1min 36s, sys: 34.8 s, total: 2min 11s
Wall time: 2min 10s
```

After fitting the model we gonna use it to generate the new data

```
%%time
synthetic_data_copulagan = model_copulagan.sample(num_rows=data.shape[0])
synthetic_data_copulagan
```

```
CPU times: user 2.53 s, sys: 52.8 ms, total: 2.58 s
Wall time: 2.72 s
```

	loan_id	infringed	contract_type	gender	has_own_car	has_own_realty	num
0	455856	0	Cash loans	M	Y	N	
1	227417	0	Cash loans	F	N	Y	
2	338245	0	Cash loans	F	N	Y	
3	202030	0	Cash loans	F	N	N	
4	387233	0	Cash loans	M	N	Y	
...	...	...	...	...	...	...	
30746	168754	0	Cash loans	M	Y	Y	
30747	279642	0	Cash loans	F	Y	Y	
30748	416746	0	Revolving loans	F	Y	N	
30749	403427	0	Cash loans	F	N	Y	
30750	333208	0	Cash loans	F	Y	Y	

30751 rows × 31 columns

```
save_path = os.path.join(data_drive_path, 'dataset', 'synthetic_data_CopulaGAN.csv')
synthetic_data_copulagan.to_csv(save_path, index=False)
```

Evaluate results

```
model_score_copulagan = evaluate(synthetic_data_copulagan, data)
model_score_copulagan
```

```
0.9149543433086731
```

Computing performance score

This report evaluates the shapes of the columns (marginal distributions) and the pairwise trends between the columns (correlations).

```
report_copulagan = QualityReport()
```

```
report_copulagan.generate(data, synthetic_data_copulagan,model_copulagan.get_metadata().to_dict())
```

```
Creating report: 100%|██████████| 4/4 [00:07<00:00,  1.86s/it]

Overall Quality Score: 88.33%

Properties:
Column Shapes: 89.69%
Column Pair Trends: 86.97%
```

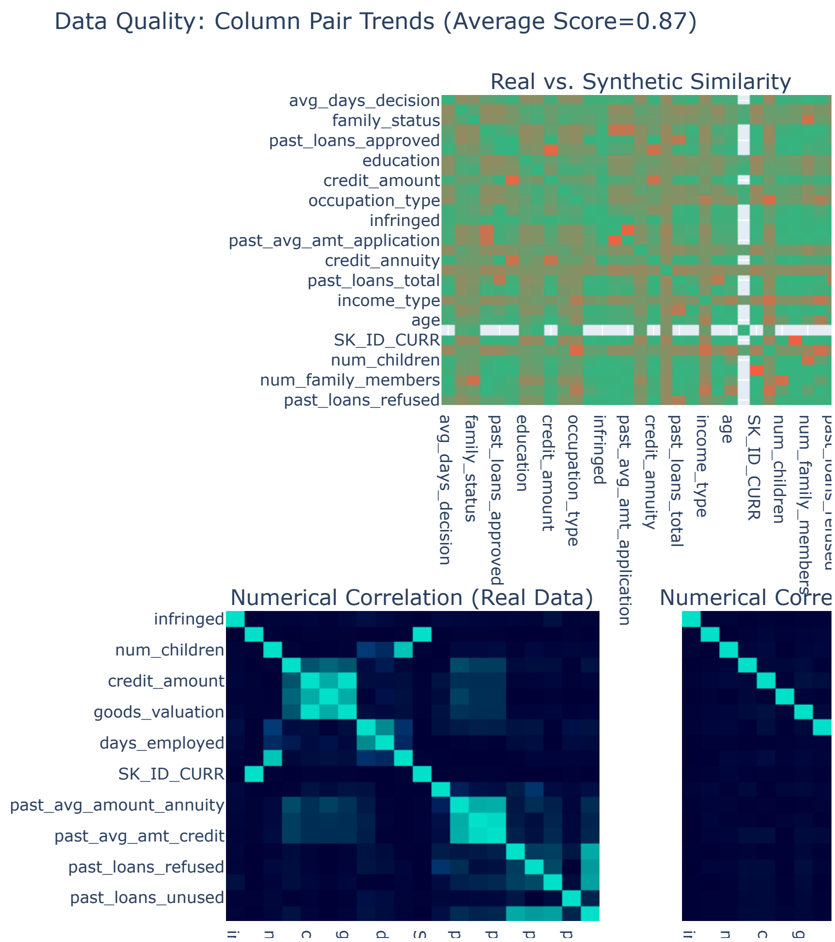
```
details_copulagan = report_copulagan.get_details(property_name='Column Shapes')
details_copulagan
```

	Column	Metric	Quality Score
0	loan_id	KSComplement	0.877207
1	infringed	KSComplement	0.961790
2	num_children	KSComplement	0.889825
3	annual_income	KSComplement	0.674417
4	credit_amount	KSComplement	0.888849
5	credit_annuity	KSComplement	0.879534
6	goods_valuation	KSComplement	0.783997
7	age	KSComplement	0.945563
8	days_employed	KSComplement	0.692888
9	mobilephone_reachable	KSComplement	0.998472
10	num_family_members	KSComplement	0.877955
11	SK_ID_CURR	KSComplement	0.890177
12	avg_days_decision	KSComplement	0.949637
13	past_avg_amount_annuity	KSComplement	0.850508
14	past_avg_amt_application	KSComplement	0.961016
15	past_avg_amt_credit	KSComplement	0.752688
16	past_loans_approved	KSComplement	0.961496
17	past_loans_refused	KSComplement	0.922155
18	past_loans_canceled	KSComplement	0.970082
19	past_loans_unused	KSComplement	0.990118
20	past_loans_total	KSComplement	0.882821
21	contract_type	TVComplement	0.996098
22	gender	TVComplement	0.891743
23	has_own_car	TVComplement	0.811388
24	has_own_realty	TVComplement	0.975903
25	income_type	TVComplement	0.851940
26	education	TVComplement	0.927417
27	family_status	TVComplement	0.959741
28	housing_type	TVComplement	0.934148
29	occupation_type	TVComplement	0.951155
30	organization_type	TVComplement	0.903060

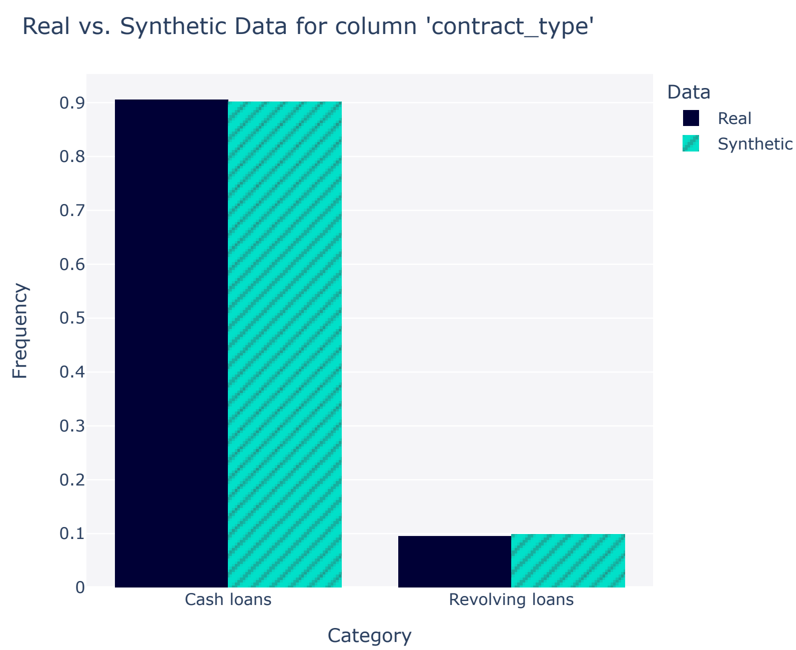
```
print('Column with more quality',details_copulagan[details_copulagan['Quality Score'] == details_copulagan['Quality Score'].max()]['Column'],
print('Column with less quality',details_copulagan[details_copulagan['Quality Score'] == details_copulagan['Quality Score'].min()]['Column'],
```

```
Column with more quality 9    mobilephone_reachable
Name: Column, dtype: object 0.99847159441969 37
Column with less quality 3    annual_income
Name: Column, dtype: object 0.6744170921270852
```

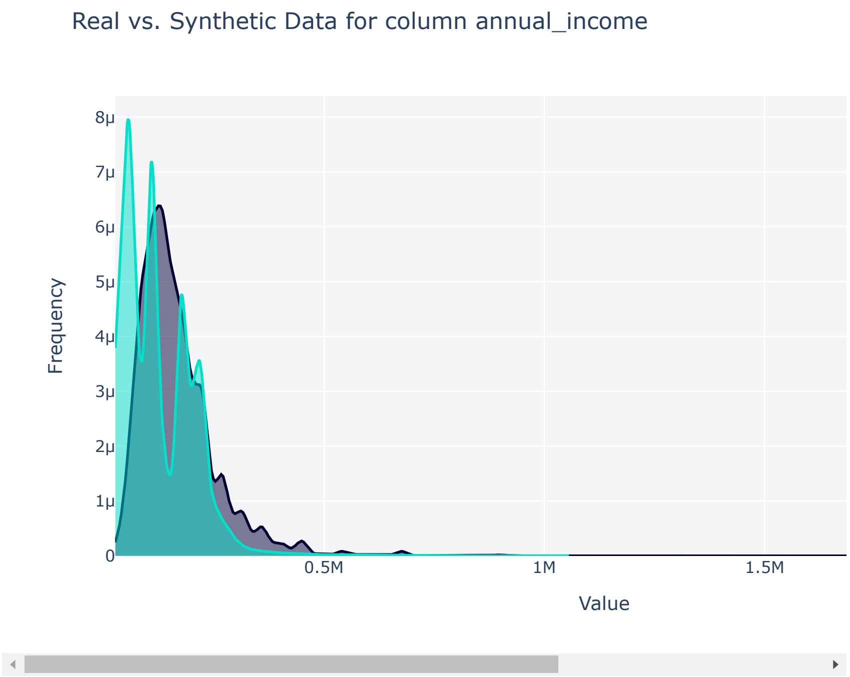
```
report_copulagan.get_visualization(property_name='Column Pair Trends')
```




```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_copulagan,
    metadata=model_copulagan.get_metadata().to_dict(),
    column_name='contract_type' # second best since the fist has some bugs
)

fig.show()
```



```
fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data_copulagan,
    metadata=model_copulagan.get_metadata().to_dict(),
    column_name='annual_income'
)

fig.show()
```



Result analysis

By observing the results, we can see that the overall quality score of our new dataset is 81%. This result is good, but we can do a deeper analysis and see the scores on the individual columns and between the correlation in the columns. The result is basically the same, so we can conclude that our synthetic data is good.

[Pros and Cons of synthetic data]

Advantages:

- **Data quality** - Higher data quality, balance, and variety are ensured with synthetic data. Artificially created data can apply labels and automatically fill in missing quantities, allowing for more precise prediction;
- **Scalability** - Synthetic data is used to cover the gaps left by real-world data;
- **Utilization simplicity** - Synthetic data guarantees all data has a consistent format and labelling, getting rid of errors and duplicates.

Disadvantages:

- Outliers are challenging to map because synthetic data merely approximates real-world data, it is not a duplicate.Therefore, some outliers that are present in original data may not be covered by synthetic data;
- The quality of the model depends on the data source.

0s completed at 3:50 PM