Feature Decorrelation

Presented by Davide Perozzi and Fabian Vincenzi

Introduction

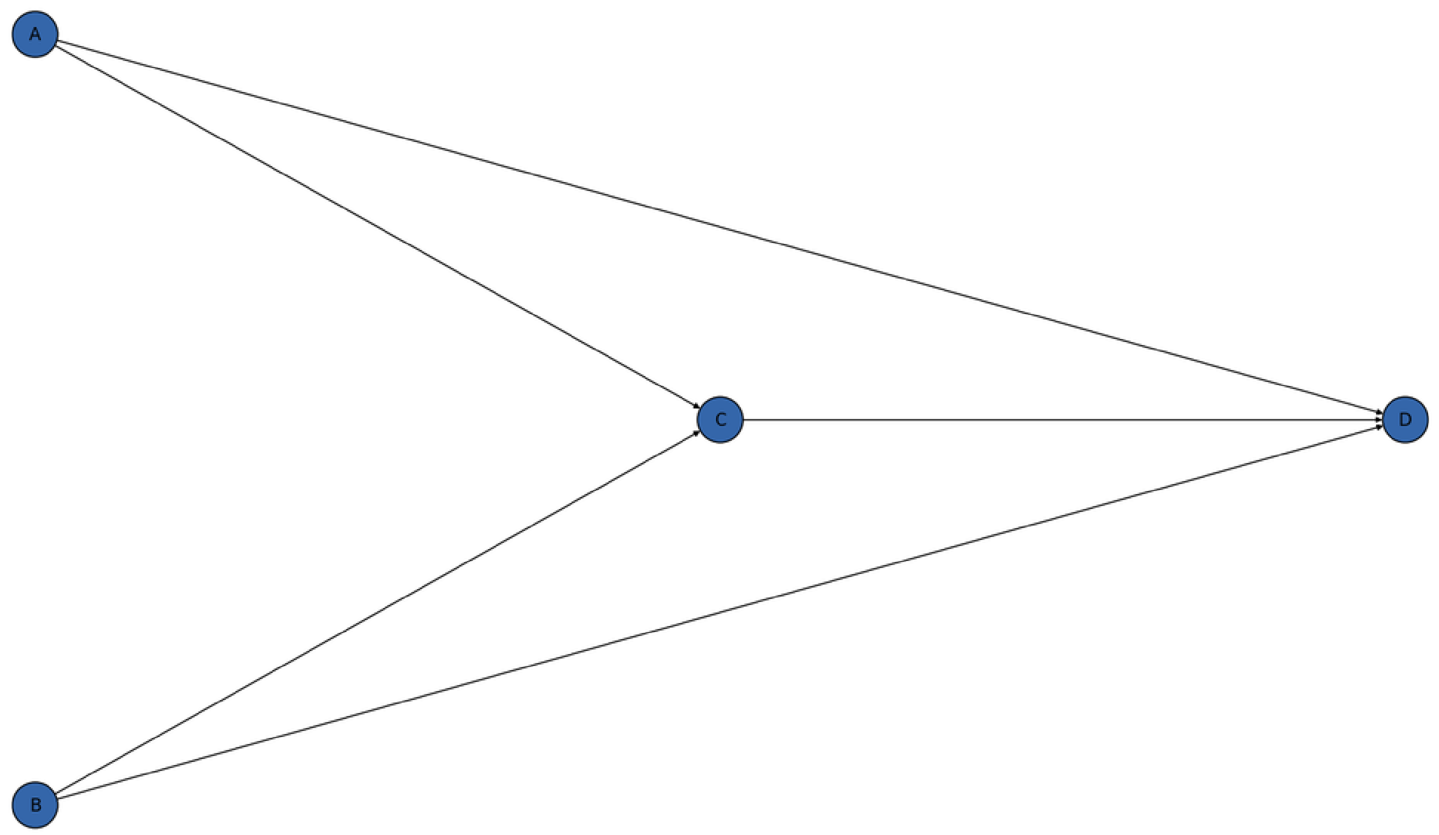
Problem overview

The task is to investigate the effect of feature decorrelation in model training, specifically the stability of the training.

To achieve this, we built a dataset and its decorrelated version with three different standard data distribution graphs and two with noise and observed the difference in training with two different model architectures.

Graphs

Using the Python library Datagen we created three different graphs:



Graph 1

A = dg . normal (mu=0, sigma =1)

B = dg . normal (mu=0, sigma =1)

C = A + B

D = A + B - 3 ∗ C

Graph 2

A = dg . normal (mu=0.5 , sigma =0.5)

B = dg . normal (mu=0.5 , sigma =0.5)

C = A + B

D = A + B - 3 ∗ C

Graph 3

A = dg . normal (mu=1.0 , sigma =0.2)

B = dg . normal (mu=0.8 , sigma =0.4)

C = A + B

D = A + B - 3 ∗ C

Graphs with noise

The first 2 graphs have also a noisy version defined so:

Graph with noise #1

A = dg . normal (mu=0, sigma =1)

B = dg . normal (mu=0, sigma =1)

C = A + B

D = A + B - 3 ∗ C + dg . n o i s e ( )

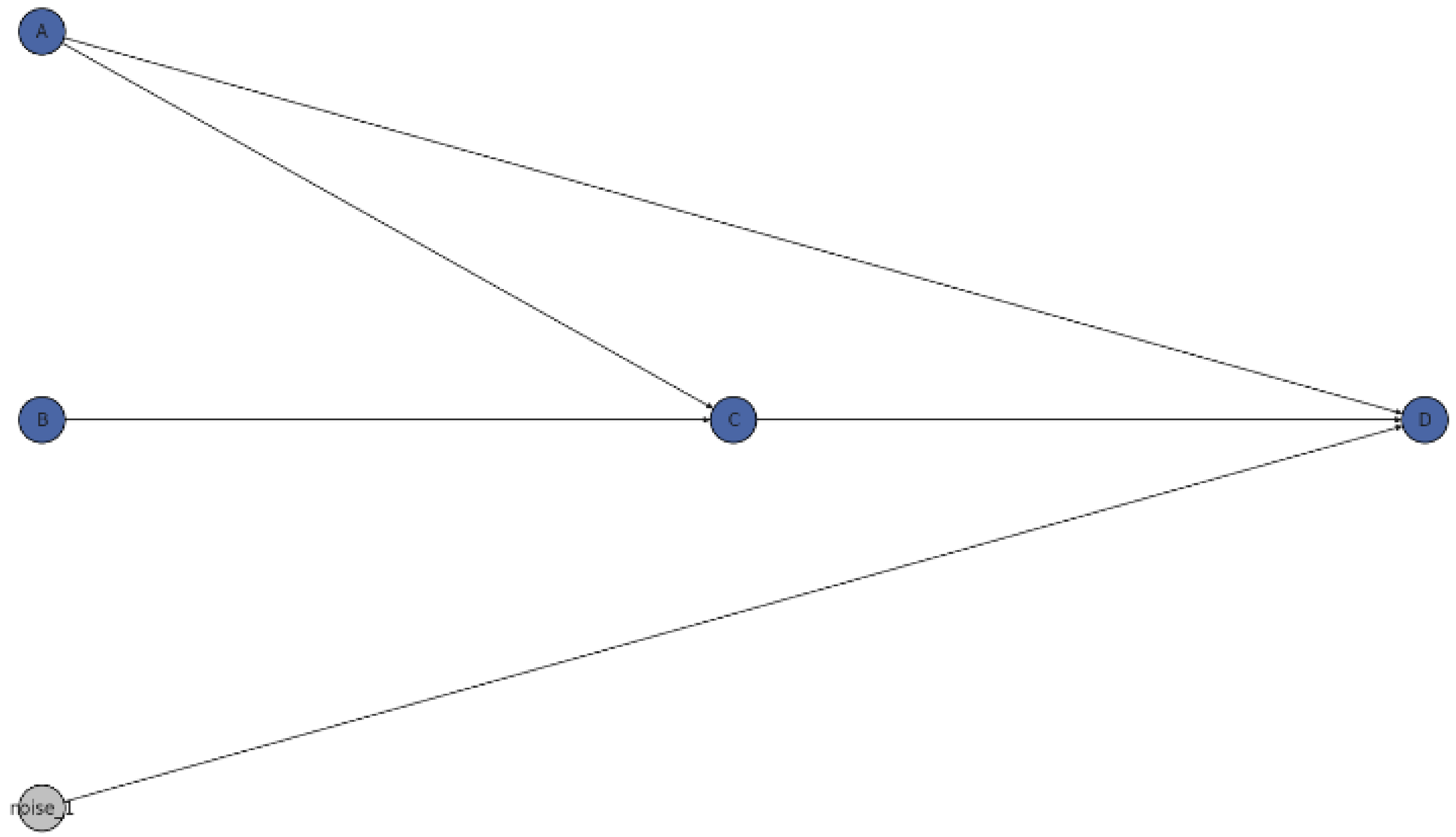
Graph with noise #2

A = dg . normal (mu=0.5 , sigma =0.5)

B = dg . normal (mu=0.5 , sigma =0.5)

C = A + B

D = A + B - 3 ∗ C + dg . n o i s e ( )

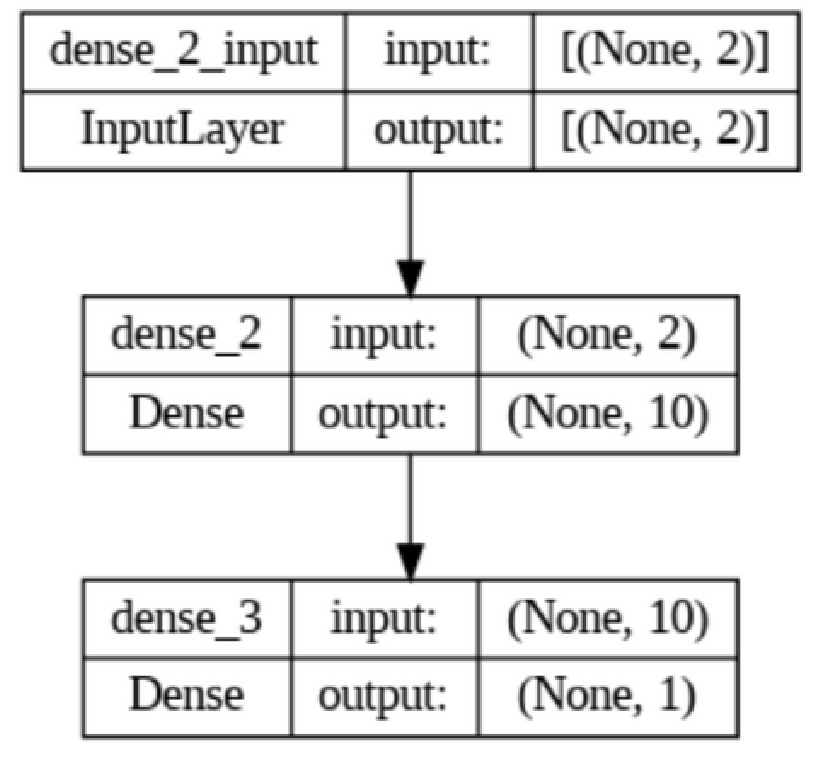


Dataset

We generated a total of 10,000 samples.

With them we compute the decorrelated feature C_prime.

To predict C_prime we trained a NN Model using Keras library.
With this Model we captured the correlation between features A,B and C.



Next we computed a new set of values for feature C, referred as C_prime:

C_prime = C - M(A,B)

This allowed us to decorrelate feature C from features A and B in the dataset, and thus investigate the impact of feature decorrelation on training stability.

	A	B	C	D	C_new	C_prime
0	0.875226	0.380171	1.255397	-1.255397	1.255379	0.000017
1	0.970282	-0.045318	0.924964	-0.924964	0.924950	0.000014
2	-0.475518	1.931827	1.456309	-1.456309	1.456267	0.000042
3	-0.151090	-0.001890	-0.152980	0.152980	-0.152998	0.000019
4	0.563920	-0.248622	0.315298	-0.315298	0.315284	0.000014

Training

We conducted experiments using two stochastic models from Keras and SciKit-Learn libraries to evaluate the impact of feature decorrelation

We trained both models with 25 different seeds for two epochs.

We collected the coefficients (feature importances) at the end of each training, and for each seed, we computed the distance of the coefficients vector from the mean coefficients vector using the L1 and L2 distance metrics.

Stochastic Linear Regression

```
keras.Sequential([
Dense (1, inputdim=3, activation='linear')
])
```

Stochastic Decision Tree

```
sklearn.tree.DecisionTreeRegressor(
splitter="random",
randomstate=seed
)
```

Results

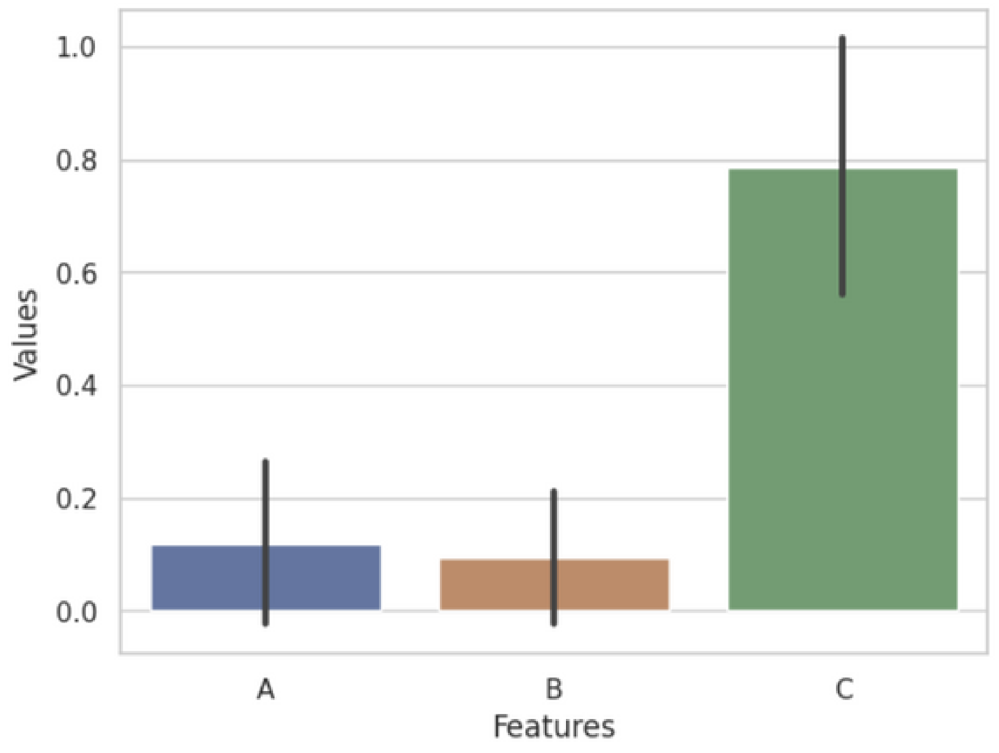
We conducted a comparison between the models trained on C_prime and C L1s and L2s metrics, specifically analyzing the ratio of them.

This comparison aimed to detect an improvement in the variability of the final coefficients.

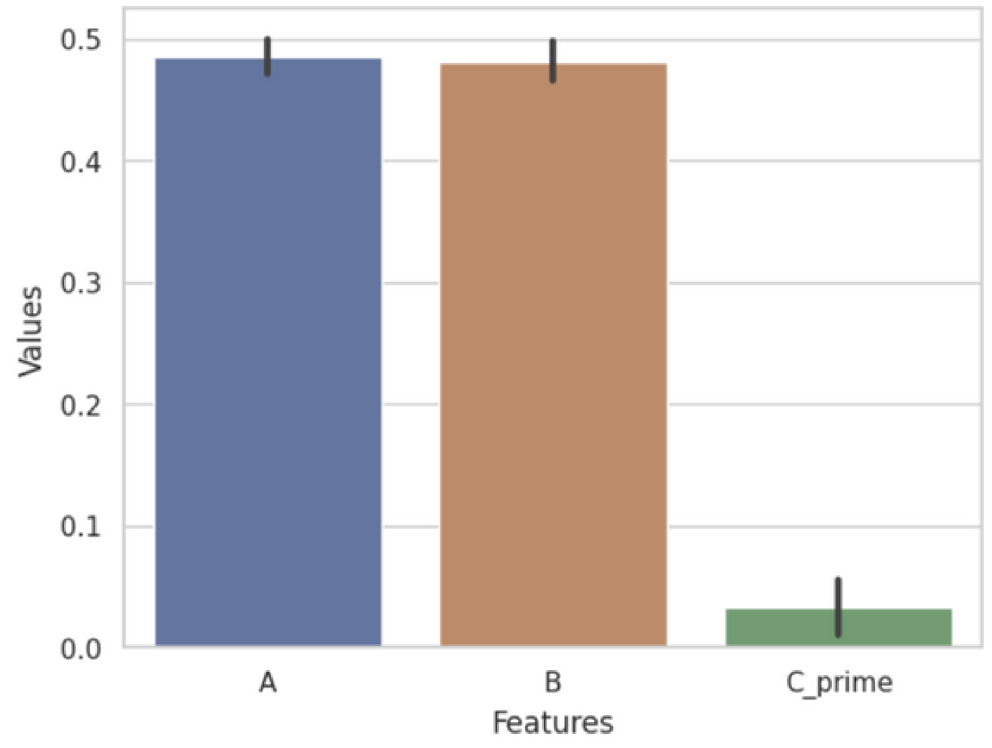
Our results in the table demonstrate that the ratios of the average L1 and L2 are consistently below 0.35 for SDT and below 0.95 for SLR. This indicates that feature decorrelation significantly improves training stability.

Notably, this improvement is particularly pronounced when using SDT (Stochastic Decision Tree) model.

This beaviour is also confirmed in presence of noise.



Train with C



Train with C_prime

Graph	Model	Ratio L1 C'/C	Ratio L2 C'/C
Graph 1	SLR	0.499974	0.864705
Graph 1	SDT	0.096018	0.097270
Graph 2	SLR	0.568582	0.864672
Graph 2	SDT	0.174069	0.170915
Graph 3	SLR	0.785187	0.931922
Graph 3	SDT	0.355453	0.339534
Graph w/ noise	SLR	0.498876	0.863772
Graph w/ noise	SDT	0.089377	0.091166
Graph w/ noise 2	SLR	0.568750	0.863956
Graph w/ noise 2	SDT	0.162088	0.165554

Error Analysis

We wanted to investigate a deteriorating of other performance indicators.

We collected the R2 loss score of the predictions for each seed and calculated the average for each model.

Our values indicate that for SLR the feature decorrelation increase the loss because the C_prime R2 loss score is consistently slightly lower thant the C R2 loss score.

This aspect should be carefully considered during a cost-benefit analysis.

R2 score C	R2 score C'
0.999999	0.999969
1.0	1.0
0.999696	0.991859
1.0	1.0
0.989710	0.956236
1.0	1.0
0.888754	0.888872
1.0	1.0
0.668298	0.663281
1.0	1.0

Conclusion

Findings:

- Feature decorrelation significantly improves training stability,
- Improvement is pronounced using SDT model,
- The improvement is achieved also in presence of noise,
- Loss is slightly increased.

In conclusion we can say that feature decorrelation offers benefits in terms of improving training stability but it is important to acknowledge that it can result in a higher loss.

THANK YOU FOR YOUR ATTENTION