

Cost estimation for precast columns

Raphael Ziegler

24. November 2022

Outline

- ① Dataset
- ② Descriptive Statistics
- ③ Models
- ④ one more thing...
- ⑤ Conclusion

Source Data

Source Excel file

Datum	Bezeichnung	StützenTyp		StützenForm		Abmessungen			Last	Stück	Hersteller	Eingekauft durch			Preis			Ratios					
		statisch rechteckig	statisch oval	statisch rechteckig	statisch oval	a	b	i				intern	Rabatt inkl. Skonto	Transport I	intern Ankunfts- Frachtkosten & Transport	extern Rabatt inkl. Skonto	extern Ankunfts- Frachtkosten & Transport	A	V	N/mm ²	CHF/m ²	Twerring	
		w	w	w	w	[m]	[m]	[m]				%	%	[CHF]	%	[CHF]	%	[m ²]	[m ²]	%			
12.01.2011	x	x	x	x	0.300				4.77	7'300	3	Firma 2	x	2'545	18.0%	2'730		0.0962	0.459	73.8	5'949	30.0%	8'190
12.01.2011	x	x	x	x	0.300				4.77	4'800	10	Firma 2	x	2'065	18.0%	2'201		0.0967	0.337	67.9	4'529	30.0%	22'013
12.01.2011	x	x	x	x	0.300				2.85	6'000	2	Firma 2	x	1'698	18.0%	1'810		0.0967	0.201	64.9	8'985	30.0%	2'630
12.01.2011	x	x	x	x	0.300				2.83	5'000	3	Firma 2	x	1'484	18.0%	1'582		0.0967	0.200	70.7	7'908	30.0%	4'746
12.01.2011	x	x	x	x	0.300				2.80	7'300	15	Firma 2	x	2'058	18.0%	2'194		0.0967	0.198	303.4	17'054	30.0%	32'807
12.01.2011	x	x	x	x	0.300				2.80	8'000	3	Firma 2	x	2'219	18.0%	2'472		0.0962	0.269	89.4	9'176	30.0%	7'416
12.01.2011	x	x	x	x	0.300				2.73	7'000	8	Firma 2	x	1'988	18.0%	2'119		0.0967	0.193	99.0	10'982	30.0%	32'755
12.01.2011	x	x	x	x	0.300				2.73	8'300	5	Firma 2	x	2'090	18.0%	2'228		0.0964	0.263	66.3	8'482	30.0%	21'140
12.01.2011	x	x	x	x	0.300				2.66	2'500	5	Firma 2	x	954	18.0%	974		0.0967	0.189	35.4	5'143	30.0%	974
12.01.2011	x	x	x	x	0.300				2.63	4'000	10	Firma 2	x	1'170	18.0%	1'247		0.0967	0.186	56.6	6'799	30.0%	32'472
12.01.2011	x	x	x	x	0.300				3.28	8'000	5	Firma 2	x	1'714	18.0%	1'827		0.0962	0.314	68.0	5'790	30.0%	1'827
12.01.2011	x	x	x	x	0.300				2.96	4'000	2	Firma 2	x	1'134	18.0%	1'209		0.0967	0.211	56.6	5'779	30.0%	2'418
12.01.2011	x	x	x	x	0.300				2.95	6'000	14	Firma 2	x	2'729	18.0%	2'822		0.0967	0.209	93.4	8'737	30.0%	25'906
12.01.2011	x	x	x	x	0.300				2.95	7'500	7	Firma 2	x	1'872	18.0%	1'996		0.0967	0.209	106.1	9'570	30.0%	12'869
12.01.2011	x	x	x	x	0.300				2.90	7'200	5	Firma 2	x	814	18.0%	866		0.0967	0.205	45.3	4'233	30.0%	4'239
12.01.2011	x	x	x	x	0.300				2.86	6'000	3	Firma 2	x	1'540	18.0%	1'642		0.0967	0.204	92.4	8'064	30.0%	4'926
12.01.2011	x	x	x	x	0.300				2.81	4'000	2	Firma 2	x	1'087	18.0%	1'159		0.0967	0.199	54.0	5'154	30.0%	2'317
12.01.2011	x	x	x	x	0.300				2.75	4'000	3	Firma 2	x	970	18.0%	1'034		0.0967	0.194	56.6	5'219	30.0%	3'352
12.01.2011	x	x	x	x	0.300				2.70	2'000	2	Firma 2	x	692	18.0%	728		0.0967	0.191	28.3	3'865	30.0%	1'475
12.01.2011	x	x	x	x	0.300				2.66	4'000	3	Firma 2	x	951	18.0%	1'014		0.0967	0.188	56.6	5'292	30.0%	3'091
12.01.2011	x	x	x	x	0.300				2.28	3'500	3	Firma 2	x	780	18.0%	838		0.0967	0.161	49.5	5'139	30.0%	2'514
12.01.2011	x	x	x	x	0.300	0.30	0.30		2.28	3'000	1	Firma 2	x	650	18.0%	699		0.0960	0.205	33.3	3'405	30.0%	699
12.01.2011	x	x	x	x	0.300				3.80	6'000	26	Firma 2	x	2'036	18.0%	2'213		0.0967	0.269	93.4	8'239	30.0%	67'538
12.01.2011	x	x	x	x	0.300				3.80	4'000	24	Firma 2	x	1'247	18.0%	1'329		0.0967	0.269	56.6	4'949	30.0%	21'903
12.01.2011	x	x	x	x	0.319				2.62	700	18	Firma 2	x	744	12.0%	851		0.0969	0.052	36.3	18'361	30.0%	10'120
12.11.2011	x	x	x	x	0.333				2.62	450	19	Firma 2	x	601	12.0%	668		0.0969	0.058	32.4	18'889	30.0%	37'063
27.08.2012	x	x	x	x	0.40	0.40	3.80	4'200	18	Firma 6	x	680	24.0%	685		0.1603	0.576	28.3	1'180	30.0%	32'235		
27.08.2012	x	x	x	x	0.30	0.30	3.80	700	12	Firma 6	x	527	24.0%	521		0.0960	0.324	7.8	7'867	30.0%	6'248		
27.08.2012	x	x	x	x	0.30	0.30	3.27	2'900	35	Firma 6	x	550	24.0%	543		0.0960	0.294	32.2	1'866	30.0%	5'782		

Source Data

DataFrame.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3447 entries, 0 to 3446
Data columns (total 33 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   Ort              0 non-null     float64 
 1   Projektname      0 non-null     float64 
 2   Datum            3447 non-null   datetime64[ns]
 3   Bezeichnung      0 non-null     float64 
 4   Stahlbetonstütze 3312 non-null   object  
 5   Stahlbetonverbundstütze 135 non-null   object  
 6   rund             724 non-null    object  
 7   eckig            2373 non-null   object  
 8   oval              253 non-null    object  
 9   d                 724 non-null    float64 
 10  a                2626 non-null   float64 
 11  b                2626 non-null   float64 
 12  l                3447 non-null   float64 
 13  Nd               3447 non-null   int64  
 14  Md               98 non-null    float64 
 15  Stück            3447 non-null   int64  
 16  Hersteller       3447 non-null   object  
 17  Einkauf_LoMa    3447 non-null   object  
 18  Einkauf_Baumeister 0 non-null    float64 
 19  LoMa             3447 non-null   float64 
 20  Rabatt inkl. Skonto LoMa 3382 non-null   float64 
 21  Transport         2212 non-null   float64 
 22  Stückpreis        3447 non-null   float64 
 23  Baumeister       0 non-null    float64 
 24  Rabatt inkl. Skonto Baumeister 0 non-null   float64 
 25  Baumeister inkl. Rabatt & Teuerung 0 non-null   float64 
 26  A                 3447 non-null   float64 
 27  V                 3447 non-null   float64 
 28  N/mm2          3447 non-null   float64 
 29  CHF/m3         3447 non-null   float64 
 30  Teuerung          3447 non-null   float64 
 31  Total Preis       3447 non-null   float64 
 32  Bemerkungen      1493 non-null   object  
dtypes: datetime64[ns](1), float64(22), int64(2), object(8)
memory usage: 888.8+ KB
```

Source Data

some words on metadata



A Column types

A.1 steel/concrete composite columns

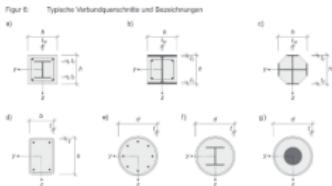


FIGURE A.1: SIA 264 [8] Figure 6, steel/concrete composite columns

A.2 reinforced concrete columns

FIGURE 14: Druckglieder mit Umschließungsbewehrung

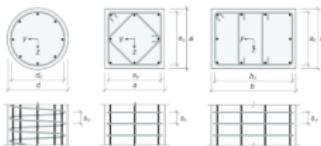


FIGURE A.2: SIA 262 [9] Figure 14, reinforced concrete columns

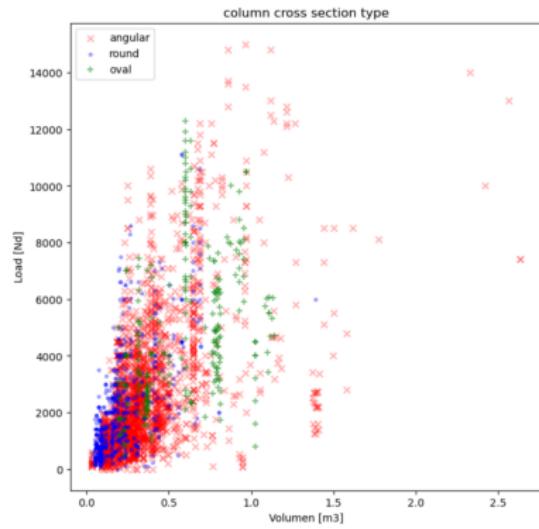
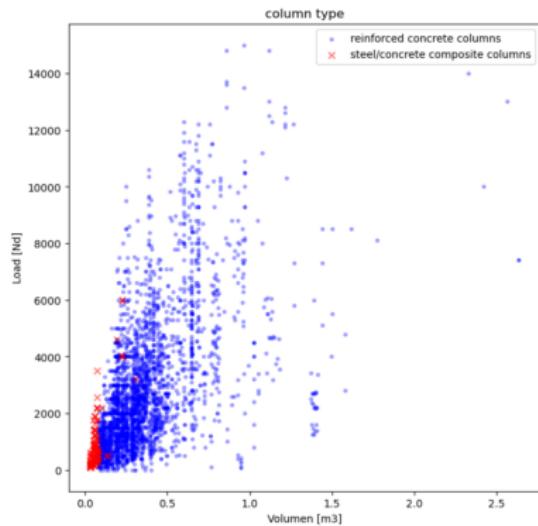
Raphael Ziegler, Cost estimation for precast columns

Data cleaning

Considered data range:

load:	$0 < N_d < 15'000$
bending moment:	$M_d < 50$
diameter:	$0.10 < d < 0.50$
width:	$0.10 < a < 1.0$
length:	$0.10 < b < 1.0$
column length:	$2.0 < l < 15.0$

Scatter plots



Procedure

top down (with linear model and neural network)

- ① all data
- ② column type
- ③ column type and cross-section
- ④ cross-section

The prediction accuracy should not fall below 70%

remarks

TABLE 7.2: Considered columns/feature

column/feature	variable name
load	Nd
cross-section area	A
column length	l
volume	V
unit price	Stückpreis

TABLE 7.3: Normalization of columns/feature

column/feature	adjustment
load	Nd/10000
cross-section area	A x 10
column length	l/10
volume	V
unit price	Stückpreis/10000

Neural Network

- input shape 4
- 3 Layers with activation function *relu*
- last layer without activation function, as I'm looking for numbers
- loss function *mse*
- metrics function r^2
- epoch standard 256

all data - linear model

MSE:
Train: 324
Test: 363

R2:
Train: 0.7097
Test: 0.6972

MSE:
Train: 340
Test: 296

R2:
Train: 0.7097
Test: 0.6928

MSE:
Train: 325
Test: 356

R2:
Train: 0.7287
Test: 0.6028

MSE:
Train: 329
Test: 343

R2:
Train: 0.7041
Test: 0.7202

all data - linear model

MSE:
Train: 329
Test: 343

R2:
Train: 0.7041
Test: 0.7202

Intercept: -70.2744
Coeficent Nd: 0.1313
Coeficent A: -5440.8351
Coeficent V: 1548.3899
Coeficent l: 175.4642

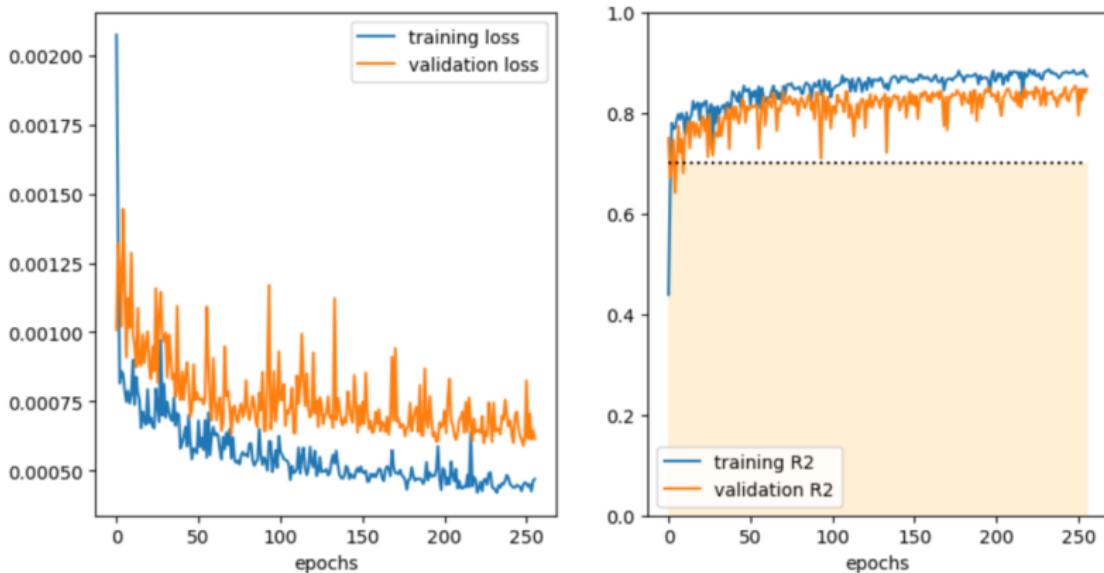
OLS Regression Results

Dep. Variable:	Stückpreis	R-squared:	0.704			
Model:	OLS	Adj. R-squared:	0.704			
Method:	Least Squares	F-statistic:	1605.			
Date:	Sun, 20 Nov 2022	Prob (F-statistic):	0.00			
Time:	15:02:10	Log-Likelihood:	-19499.			
No. Observations:	2703	AIC:	3.901e+04			
Df Residuals:	2698	BIC:	3.904e+04			
Df Model:	4					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	-70.2744	51.818	-1.356	0.175	-171.881	31.333
Nd	0.1313	0.004	36.027	0.000	0.124	0.138
A	-5440.8351	397.038	-13.704	0.000	-6219.364	-4662.306
V	1548.3899	112.573	13.754	0.000	1327.651	1769.129
l	175.4642	15.991	10.973	0.000	144.109	206.819
Omnibus:	1120.405	Durbin-Watson:	1.952			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	12892.447			
Skew:	1.638	Prob(JB):	0.00			
Kurtosis:	13.185	Cond. No.	2.39e+05			

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 2.39e+05. This might indicate that there are strong multicollinearity or other numerical problems.

all data - neural network



22/22 - 0s - loss: 6.1461e-04 - r_square: 0.8474 - 37ms/epoch - 2ms/step

```
1 xin = np.array([[2500/10000, .25*.25*10, .25*.25*3, 3/10]])
2 print(f"NN.predict(NNmodel, xin)[0][0]*10000:.0f")
```

1/1 [=====] - 0s 19ms/step
726

compound columns - linear model

MSE:
Train: 190
Test: 416

R2:
Train: 0.5475
Test: 0.7510

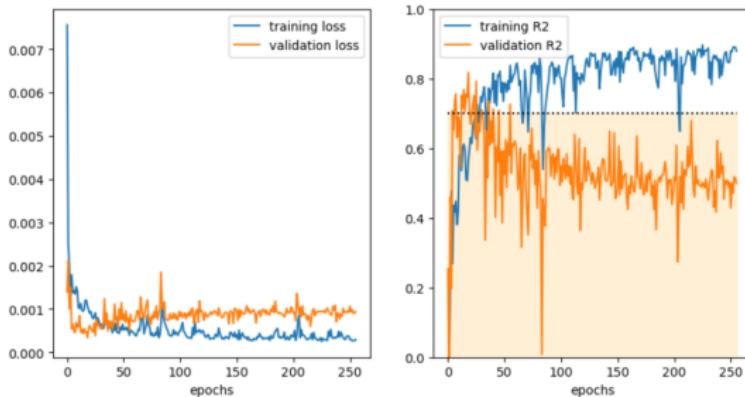
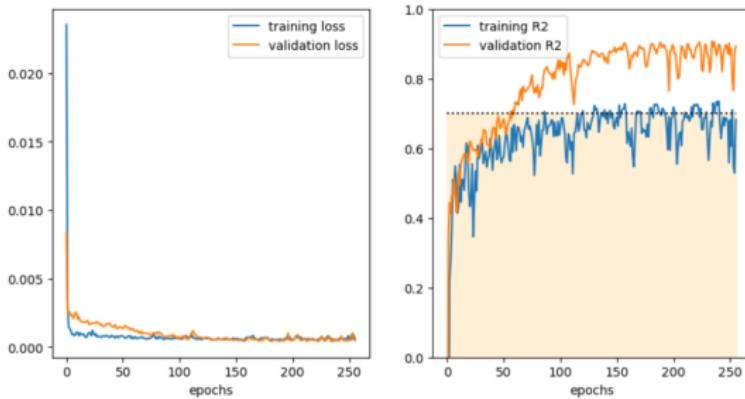
MSE:
Train: 244
Test: 236

R2:
Train: 0.7517
Test: 0.7120

MSE:
Train: 245
Test: 211

R2:
Train: 0.7701
Test: 0.5621

compound colums - neural network



concret columns - linear model

MSE:

Train: 325

Test: 319

MSE:

Train: 328

Test: 308

MSE:

Train: 323

Test: 327

R2:

Train: 0.7282

Test: 0.7102

R2:

Train: 0.7214

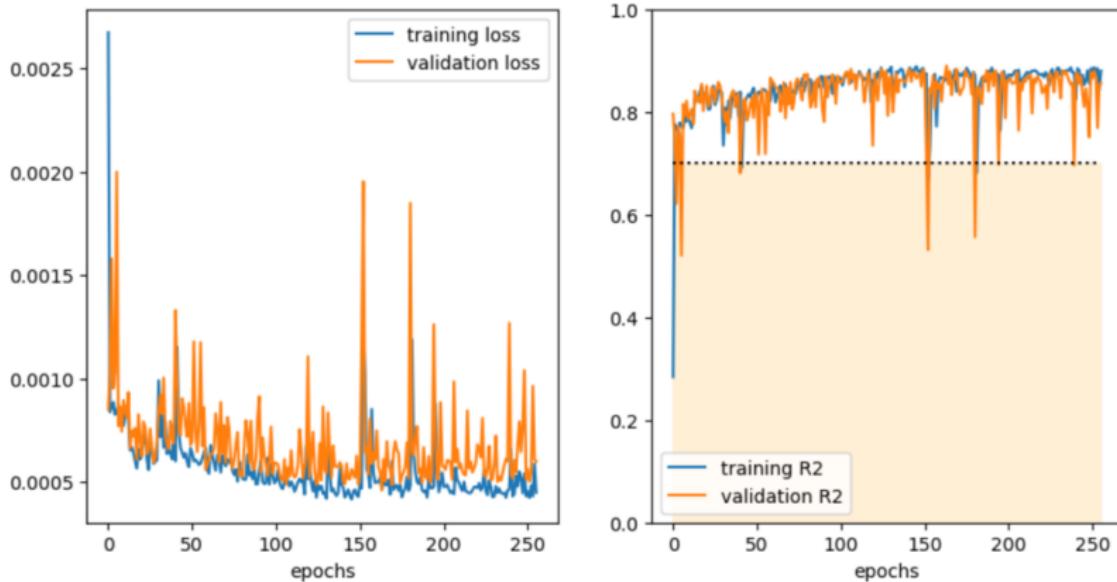
Test: 0.7405

R2:

Train: 0.7217

Test: 0.7367

concret columns - neural network



21/21 - 0s - loss: 5.9896e-04 - r_square: 0.8563 - 33ms/epoch - 2ms/step

concret columns - round

MSE:

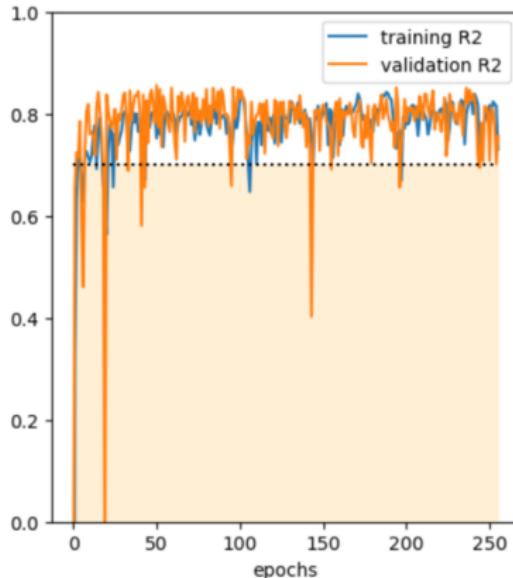
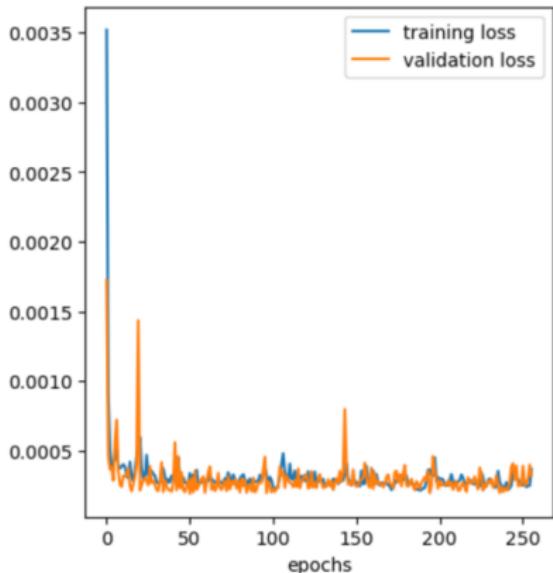
198

169

R2:

0.7244

0.7432



concret columns - angular

MSE :

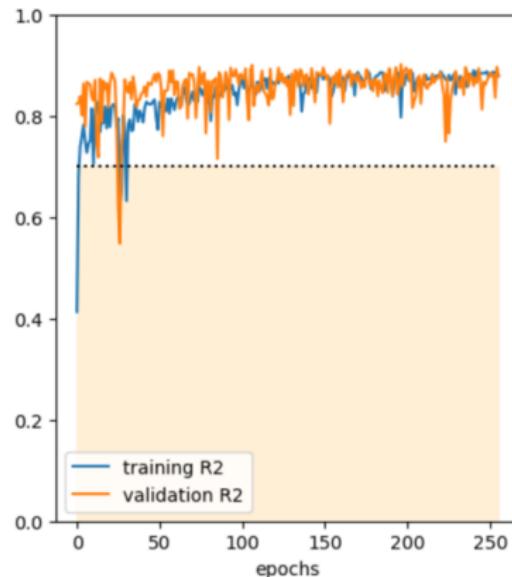
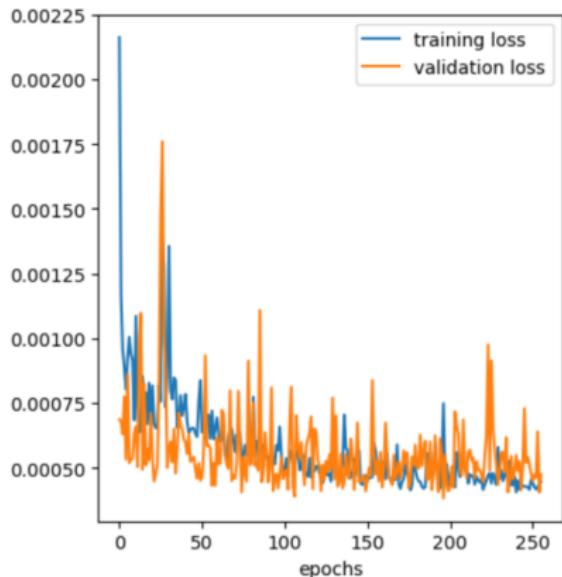
327

331

R2 :

0.7017

0.7454



concret columns - oval

MSE :

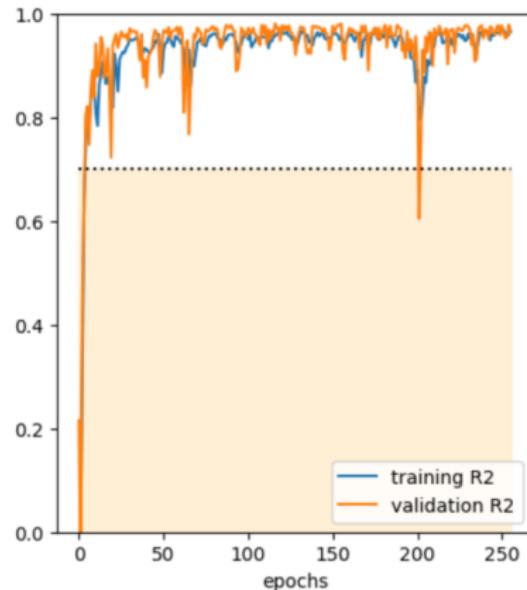
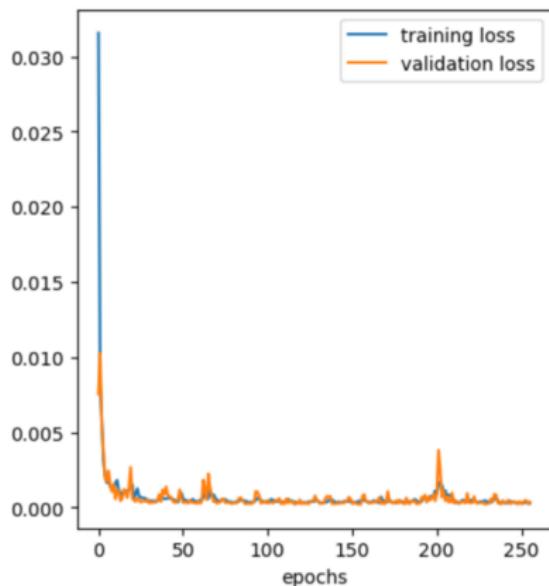
424

432

R2:

0.7893

0.7949



round - linear model

MSE :
224
191

R2:
0.6642
0.6680

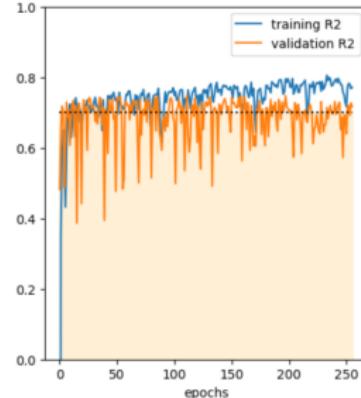
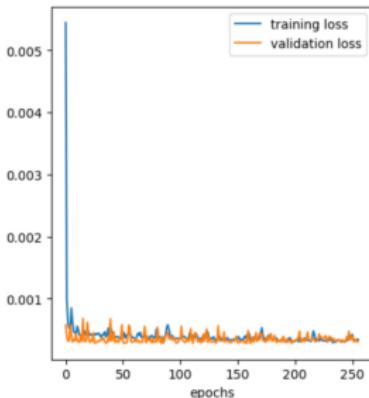
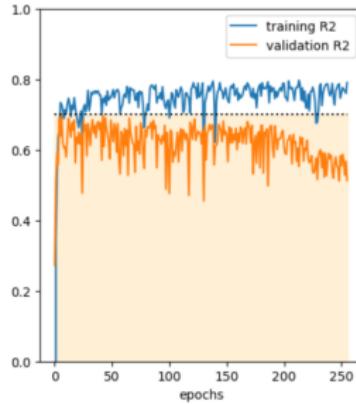
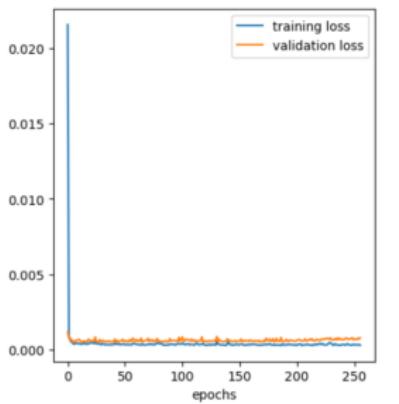
MSE :
216
224

R2:
0.6803
0.6041

MSE :
217
217

R2:
0.6518
0.7156

round - neural network



angular - linear model

MSE:

332

347

R2:

0.6998

0.6972

MSE:

316

403

R2:

0.7080

0.6718

MSE:

342

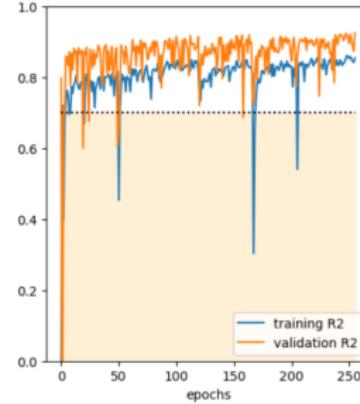
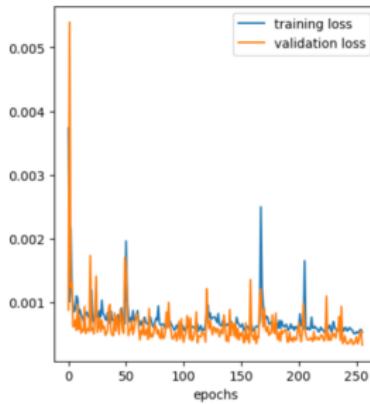
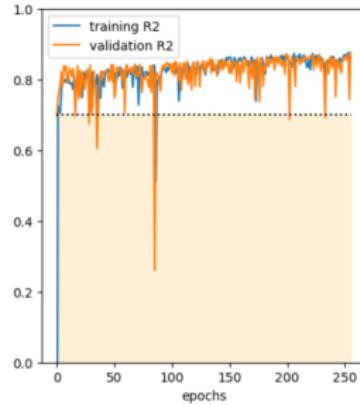
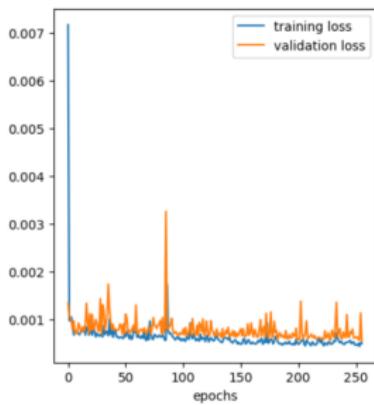
304

R2:

0.6966

0.7119

angular - neural network



What if...

TABLE 7.2: Considered columns/feature

column/feature	variable name
load	Nd
cross-section area	A
column length	l
volume	V
unit price	Stückpreis

What if I used more/different features?

Principal Component Analysis

Reformatting the DataFrame:

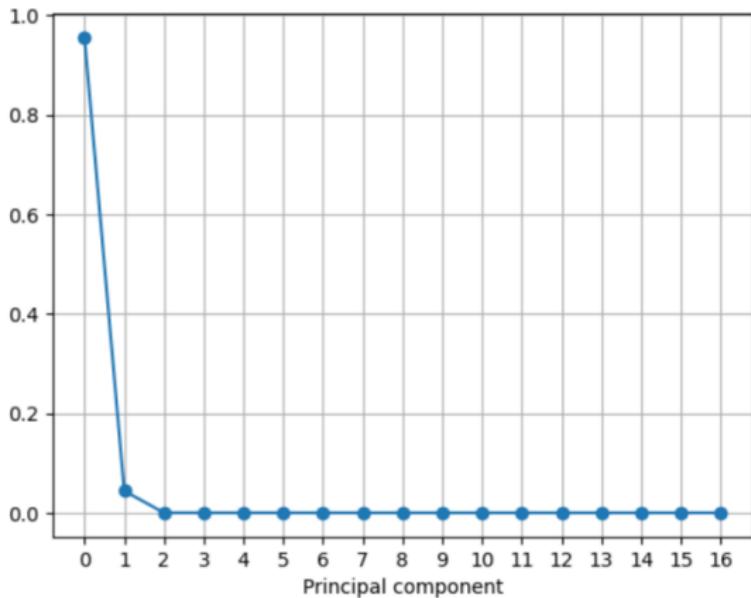
- Dropping some features
- fill Nan values with 0
- replace x, X values with 1

```
df_pca = df.drop(["Datum", "Stückpreis", "Ort", "Projektname", "Bezeichnung", "Einkauf_Baumeister",
                  "Baumeister", "Rabatt inkl. Skonto Baumeister", "Baumeister inkl. Rabatt & Teuerung",
                  "Bemerkungen", "Einkauf_LoMa", "Rabatt inkl. Skonto LoMa", "Teuerung", "Hersteller", "CHF/m³",
                  "Total Preis"], axis=1)

df_pca = df_pca.fillna(0, inplace=False)

df_pca = df_pca.replace(to_replace=["x", "X"], value=[1, 1])
```

Principal Component Analysis

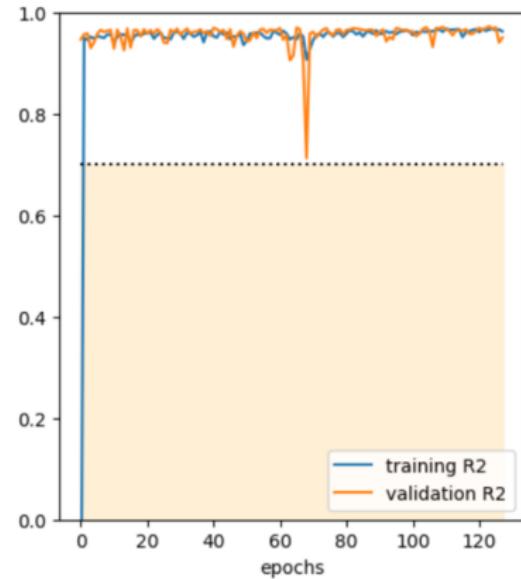
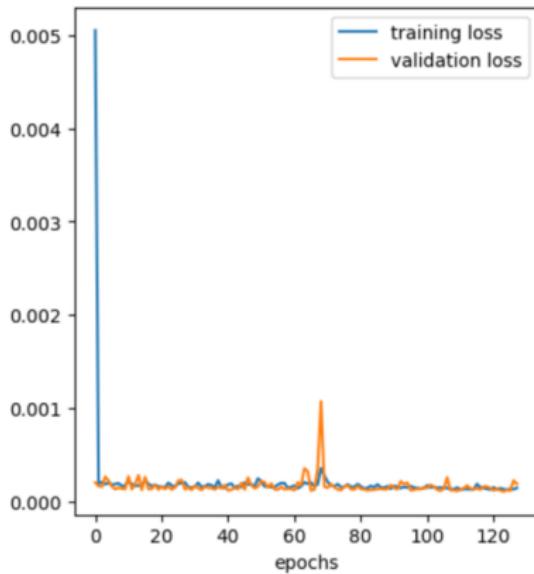


PCA - linear model & neural network

```
1 print(f"np.sum(pca.explained_variance_ratio_[:3]):{0.6f}")  
0.999944
```

MSE:
130
118

R2:
0.9555
0.9623



22/22 - 0s - loss: 1.8603e-04 - r_square: 0.9500 - 42ms/epoch - 2ms/step

PCA - prediction

```
1 Stahlbetonstütze = 1
2 Stahlbetonverbundstütze = 0
3 rund = 0
4 eckig = 1
5 oval = 0
6 d = 0
7 a = 0.25
8 b = 0.25
9 l = 3.0
10 Nd = 2000
11 Md = 0
12 Stück = 1
13 LoMa = 1
14 Transport = 10
15 A = a * b
16 V = A * l
17 N_mm2 = Nd*1000/(A*100*100*100)
```

```
1/1 [=====] - 0s 24ms/step
194
```

PCA - prediction

```
1 Stahlbetonstütze = 1
2 Stahlbetonverbundstütze = 0
3 rund = 0
4 eckig = 1
5 oval = 0
6 d = 0
7 a = 0.25
8 b = 0.25
9 l = 3.0
10 Nd = 2000
11 Md = 0
12 Stück = 1
13 LoMa = 1
14 Transport = 10
15 A = a * b
16 V = A * l
17 N_mm2 = Nd*1000/(A*100*100*100)
```

```
1/1 [=====] - 0s 24ms/step
194
```

this is not a realistic value!

PCA - prediction

```
1 Stahlbetonstütze = 1
2 Stahlbetonverbundstütze = 0
3 rund = 0
4 eckig = 1
5 oval = 0
6 d = 0
7 a = 0.25
8 b = 0.25
9 l = 3.0
10 Nd = 2000
11 Md = 0
12 Stück = 1
13 LoMa = 1
14 Transport = 10
15 A = a * b
16 V = A * l
17 N_mm2 = Nd*1000/(A*100*100*100)
```

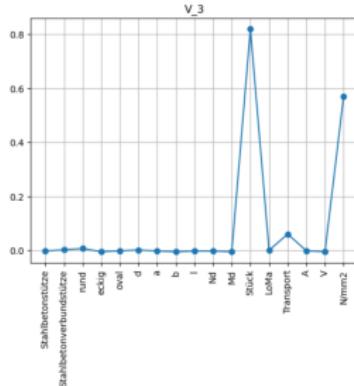
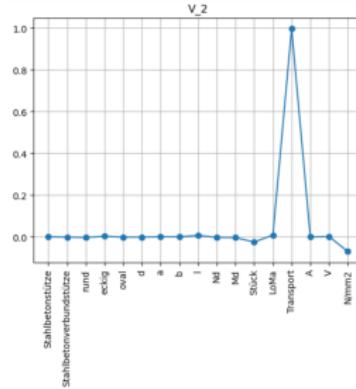
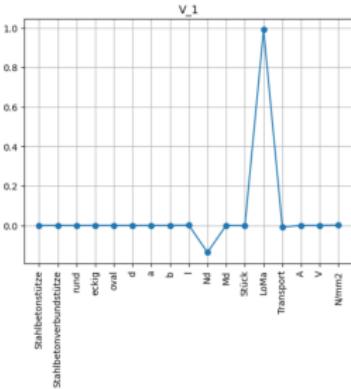
```
1/1 [=====] - 0s 24ms/step
194
```

this is not a realistic value!

What happen?

Principal Component Analysis

first 3 components:



PCA round 2

Reformatting the DataFrame:

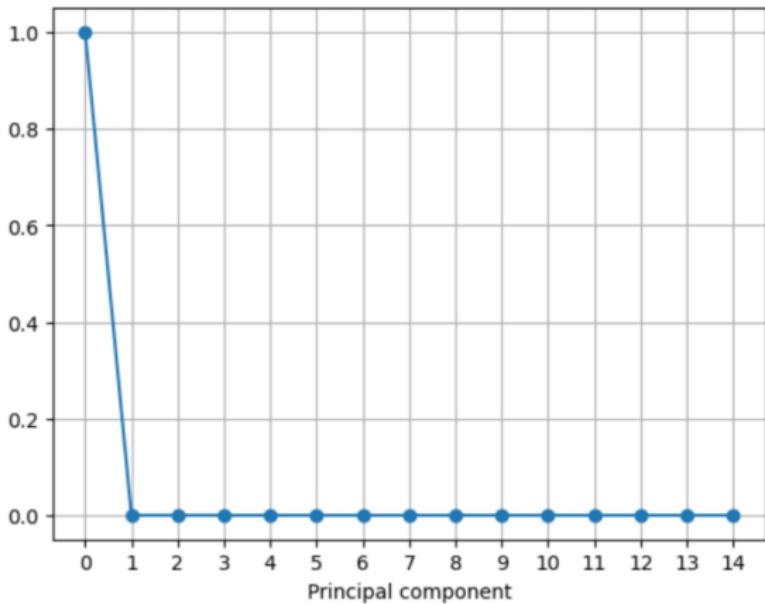
- Dropping some features
- fill Nan values with 0
- replace x, X values with 1

```
df_pca = df.drop(["Datum", "Stückpreis", "Ort", "Projektname", "Bezeichnung", "Einkauf_Baumeister",
                  "Baumeister", "Rabatt inkl. Skonto Baumeister", "Baumeister inkl. Rabatt & Teuerung",
                  "Bemerkungen", "Einkauf_LoMa", "Rabatt inkl. Skonto LoMa", "Teuerung", "Hersteller", "CHF/m3",
                  "LoMa", "Transport", "Total Preis"], axis=1)

df_pca = df_pca.fillna(0, inplace=False)

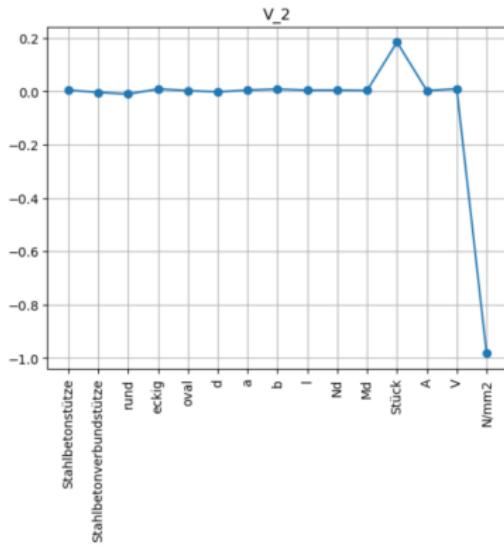
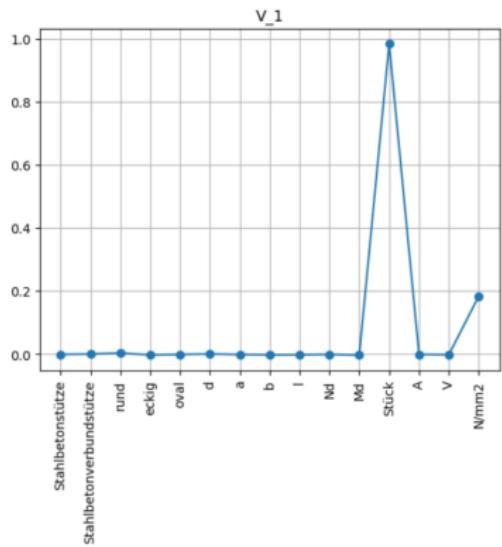
df_pca = df_pca.replace(to_replace=["x", "X"], value=[1, 1])
```

PCA round 2



Principal Component Analysis

first 2 components:



PCA round 2

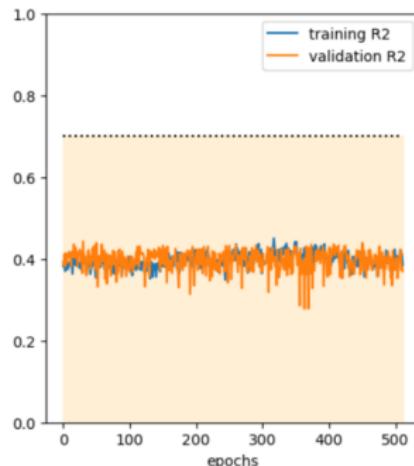
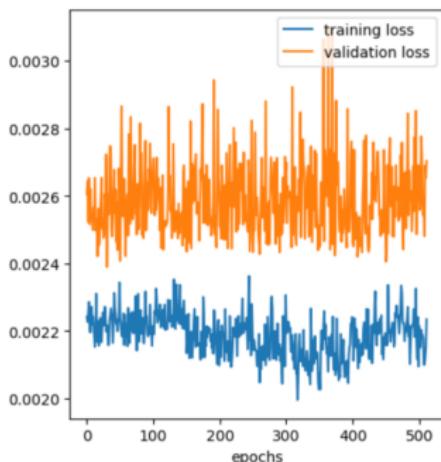
linear model & neural network

```
1 print(f"np.sum(pca.explained_variance_ratio_[:2]):{0.6f}")
```

0.999842

MSE:
Train: 497
Test: 540

R2:
Train: 0.3197
Test: 0.3200



22/22 - 0s - loss: 0.0027 - r_square: 0.3697 - 33ms/epoch - 2ms/step

PCA round 2 - prediction

```
1 Stahlbetonstütze = 1
2 Stahlbetonverbundstütze = 0
3 rund = 0
4 eckig = 1
5 oval = 0
6 d = 0
7 a = 0.25
8 b = 0.25
9 l = 3.0
10 Nd = 2000
11 Md = 0
12 Stück = 1
13 # LoMa = 1
14 # Transport = 10
15 A = a * b
16 V = A * l
17 N_mm2 = Nd*1000/(A*100*100*100)
```

```
1/1 [=====] - 0s 61ms/step
532
```

Better result even if the accuracy is only around 36%

Conclusion

Conclusion

- ① the neural network meets the requirements for accuracy, I can predict the column cost.

Conclusion

- ① the neural network meets the requirements for accuracy, I can predict the column cost.
- ② test/train split can make a difference

Conclusion

- ① the neural network meets the requirements for accuracy, I can predict the column cost.
- ② test/train split can make a difference
- ③ more data is better

Conclusion

- ① the neural network meets the requirements for accuracy, I can predict the column cost.
- ② test/train split can make a difference
- ③ more data is better
- ④ the neural network performs better than the linear model

Conclusion

- ① the neural network meets the requirements for accuracy, I can predict the column cost.
- ② test/train split can make a difference
- ③ more data is better
- ④ the neural network performs better than the linear model
- ⑤ shit in shit out

Conclusion

- ① the neural network meets the requirements for accuracy, I can predict the column cost.
- ② test/train split can make a difference
- ③ more data is better
- ④ the neural network performs better than the linear model
- ⑤ shit in shit out
- ⑥ know your data

