Regresi Linear

Ali Akbar Septiandri

Universitas Al-Azhar Indonesia

aliakbars@live.com

October 25, 2019

Selayang Pandang

1. Simple Linear Regression
2. Multicollinearity
3. Basis Function Regression
4. Regularisation

Bahan Bacaan

1. VanderPlas, J. (2016). Python Data Science Handbook. (In Depth: Linear Regression) `http://nbviewer.jupyter.org/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.06-Linear-Regression.ipynb`
2. McElreath, R. (2018). Statistical Rethinking. (Chapter 4, Chapter 8) `https://xcelab.net/rm/statistical-rethinking/`
3. Murray, I. (2016). MLPR class notes. (Linear Regression; Regression and Gradients) `http://www.inf.ed.ac.uk/teaching/courses/mlpr/2016/notes/` (graduate level)

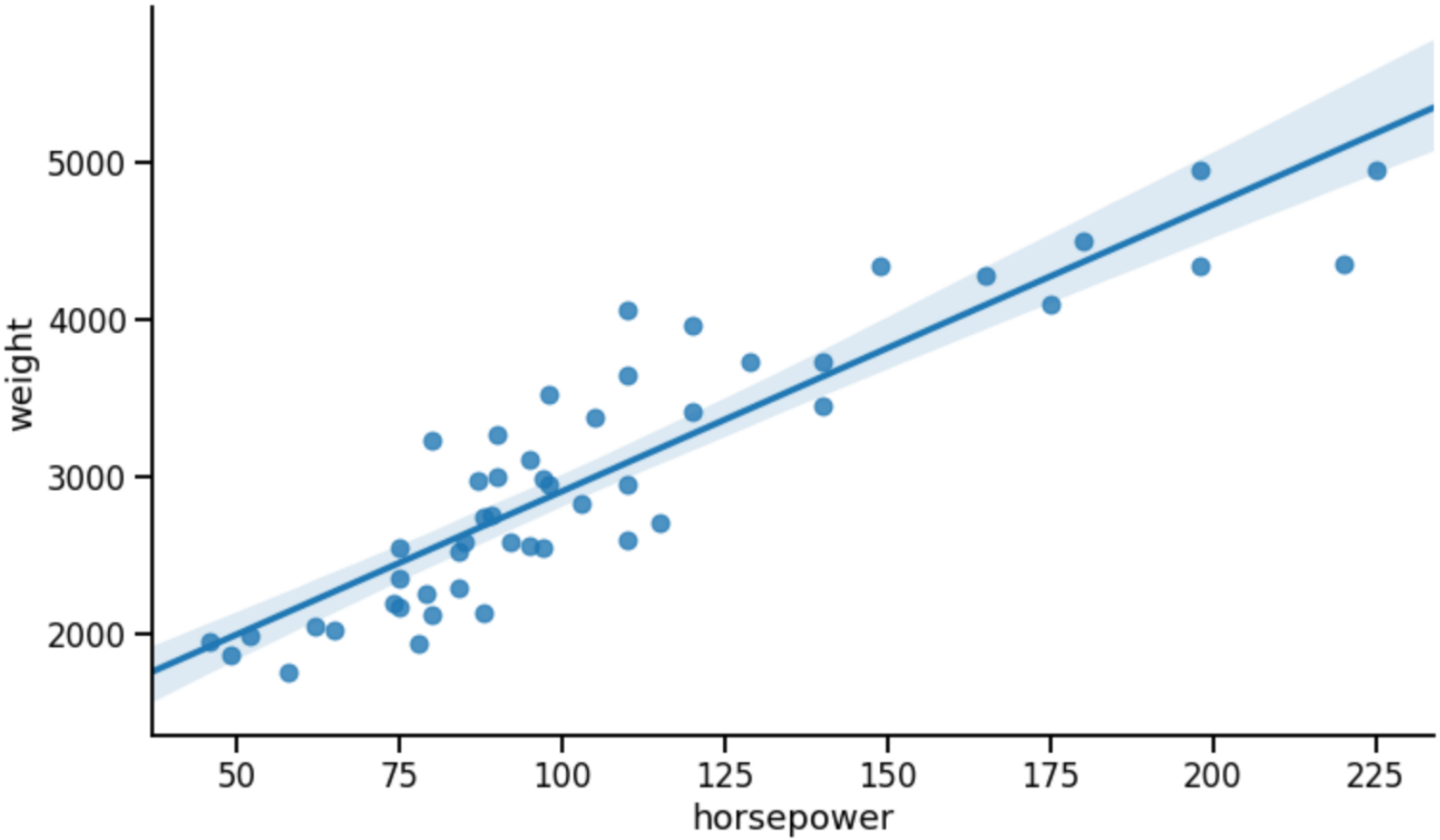
Gambar: Golem of Prague

"...scientific models are neither true nor false, neither prophets nor charlatans. Rather they are constructs engineered for some purpose."

- Richard McElreath

Simple Linear Regression

Prediksi hubungan antara dua variabel



Gambar: weight $= 1083.77 + 18.21$ horsepower

Simple Linear Regression

Fungsi linear

Kasus paling sederhana adalah mencocokkan garis lurus ke sekumpulan data

$$y = ax + b$$

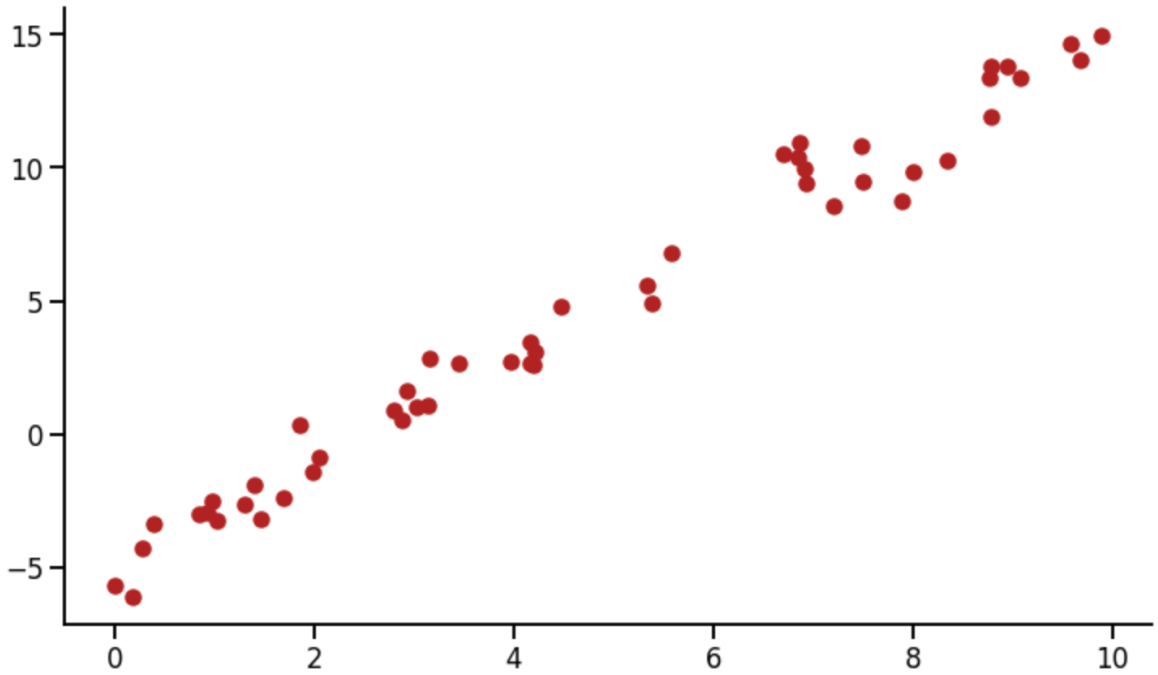
dengan a adalah *slope*, sedangkan b dikenal dengan nama *intercept*.

Notasi lain

$$y = w_0 + w_1 x_1$$

dengan w adalah bobot atau koefisien.

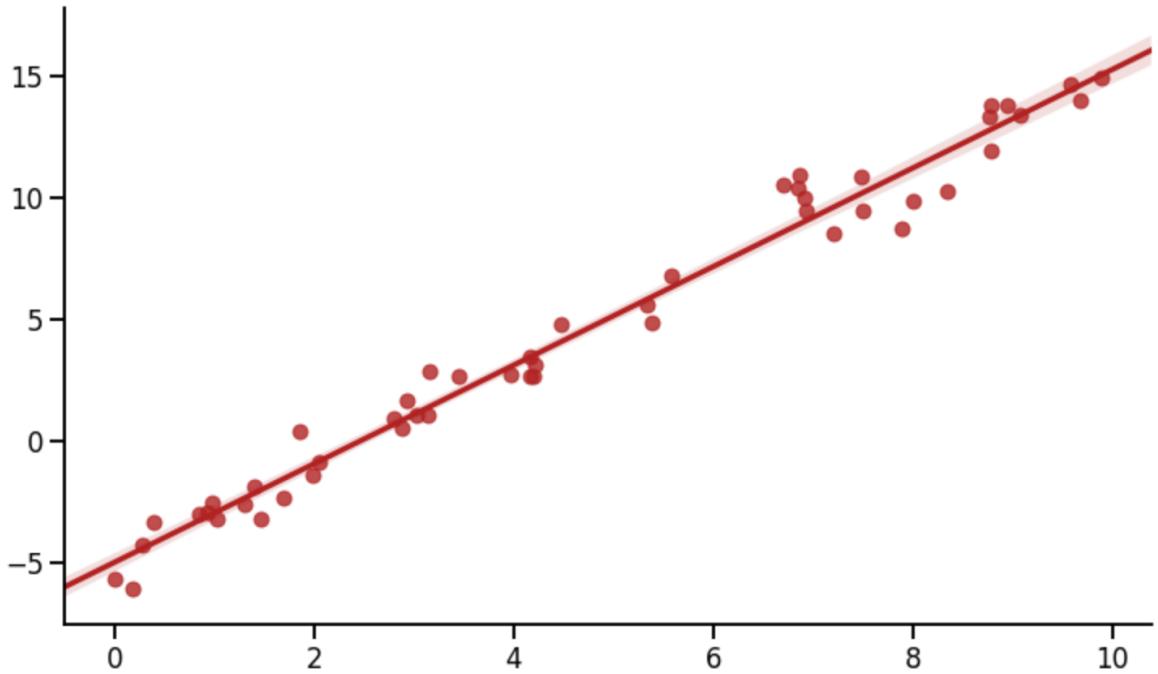
Simple Linear Regression



Example

```
1 rng = np.random.RandomState(1)
2 x = 10 * rng.rand(50)
3 y = 2 * x - 5 + rng.randn(50)
4 plt.scatter(x, y);
```

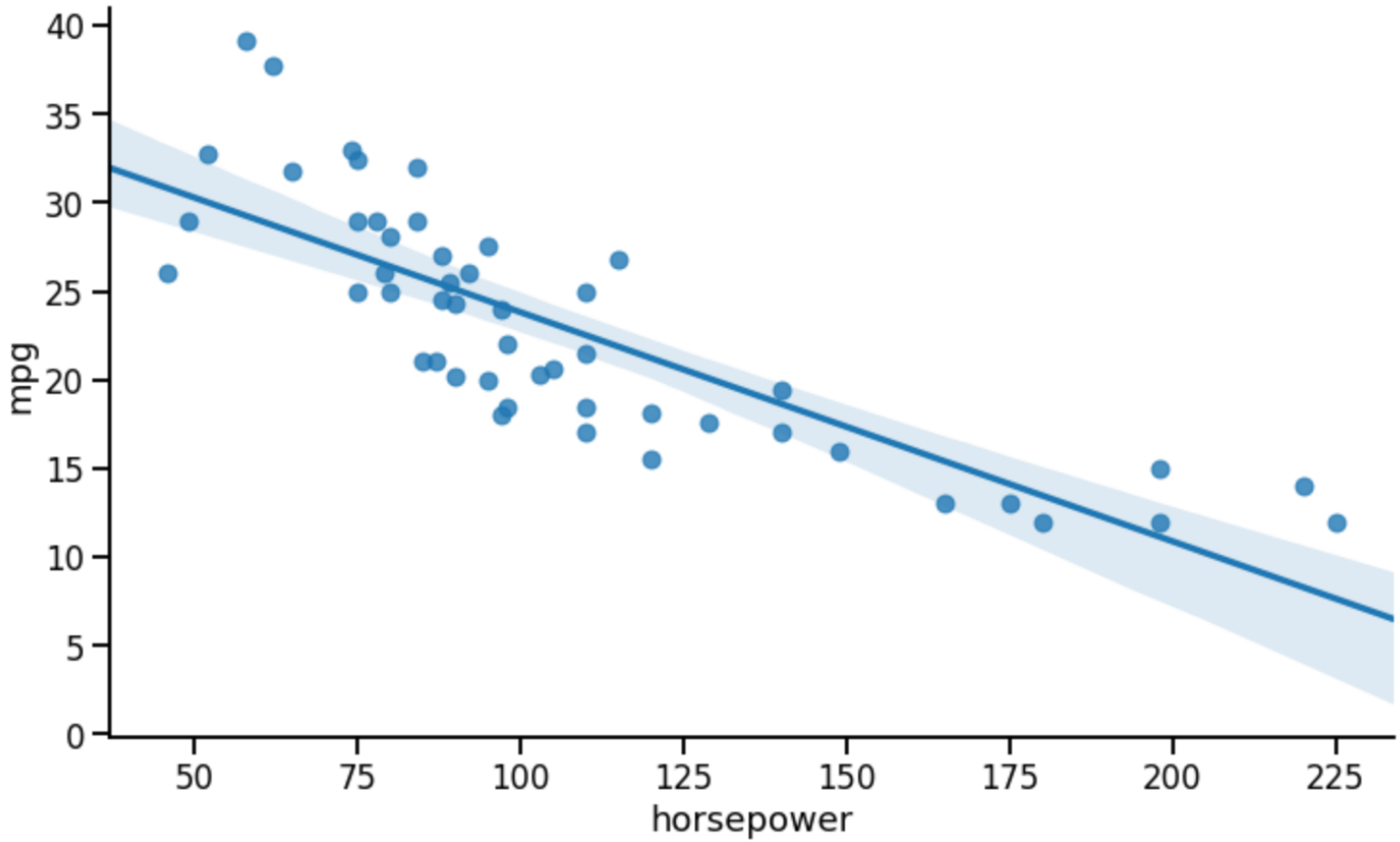
Mencocokkan Garis



Gambar: Hasil pencocokan garis

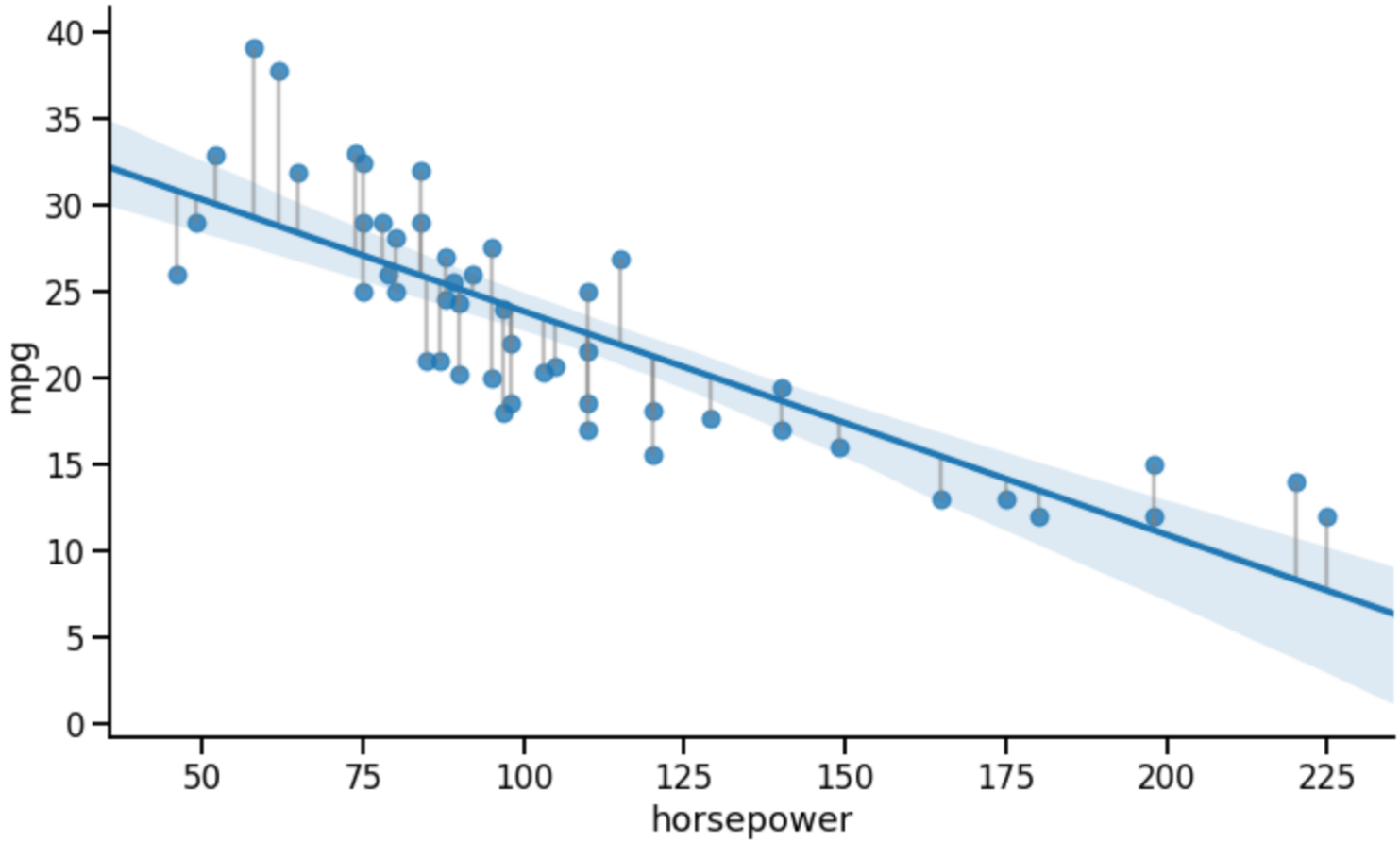
```
Model slope:  2.02720881036
Model intercept:  -4.99857708555
```

Prediksi hubungan tenaga mobil dan konsumsi bahan bakar



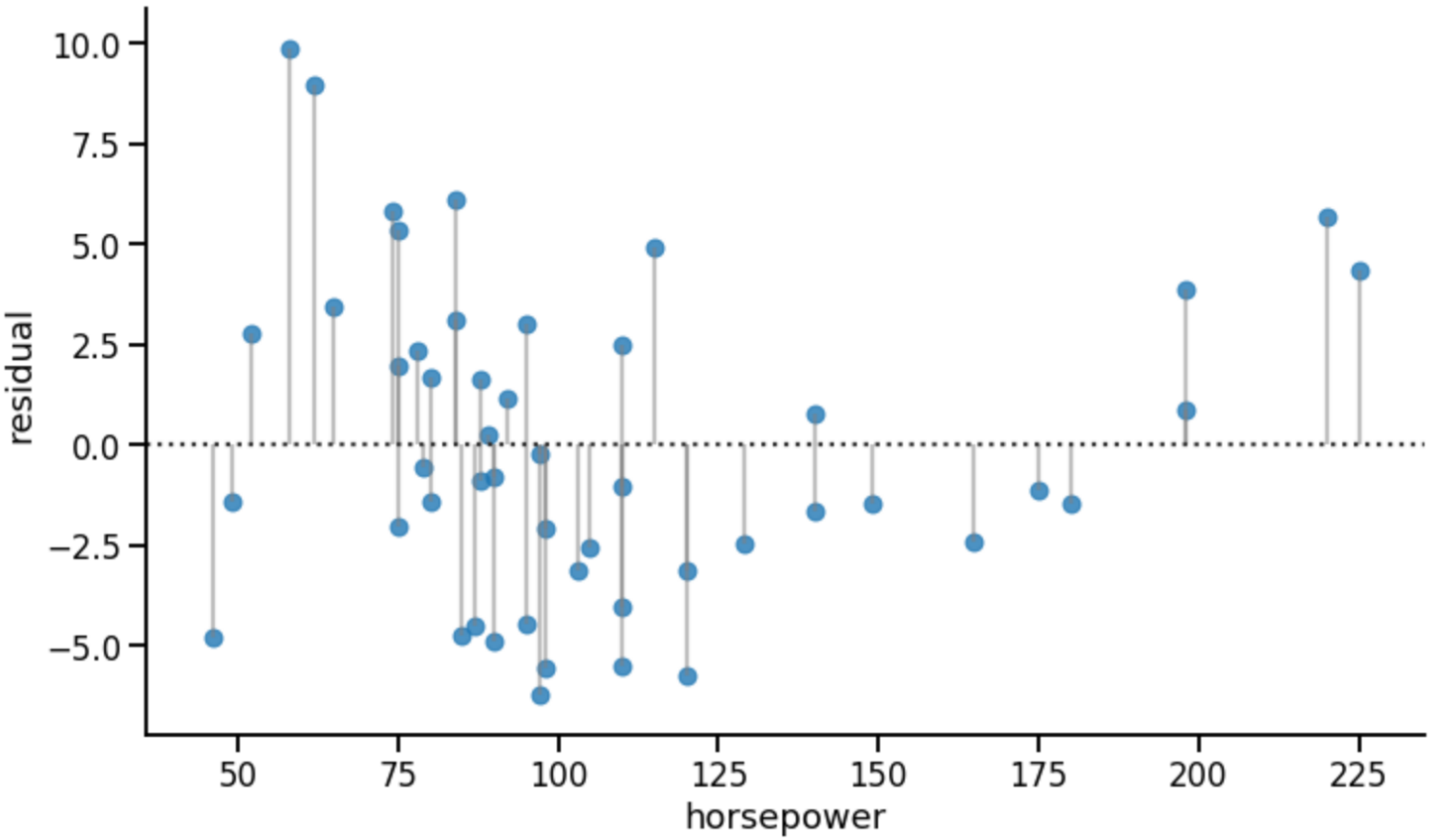
Gambar: $\text{mpg} = 36.75 - 0.13\ \text{horsepower}$

Residual



Gambar: Kita menginginkan garis (fungsi) yang meminimalkan residual

Residual Plot



Gambar: Residual plot untuk menggambarkan kinerja model

Bagaimana kalau ada lebih dari dua variabel
yang ingin kita lihat hubungannya?

Multivariable Linear Regression

Model

$$y = w_0 x_0 + w_1 x_1 + w_2 x_2 + ... + w_D x_D = \sum_{j=0}^{D} w_j x_j$$

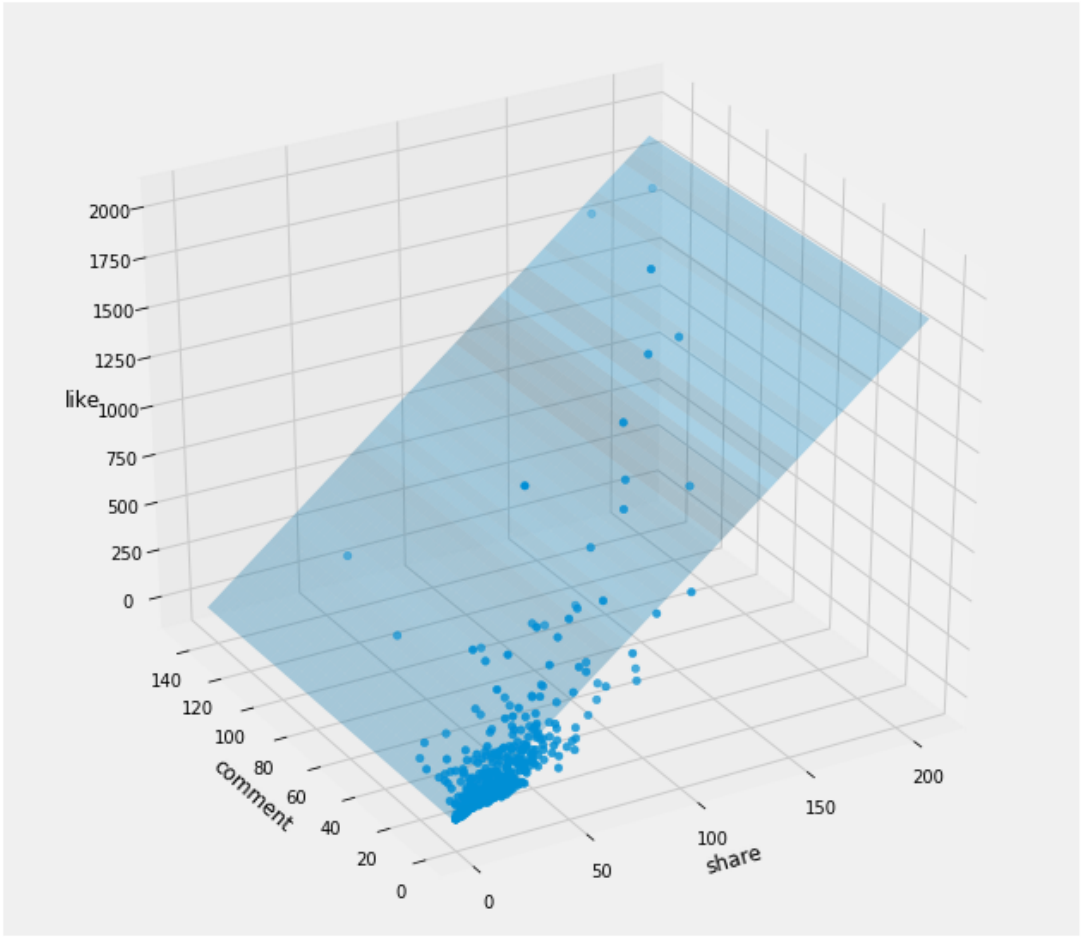
dengan $x_0 = 1$

Notasi matriks-vektor

$$y = \phi \mathbf{w}$$

dengan $\phi = (1, \mathbf{x}^T)$

Regresi linear untuk dua variabel



Gambar: Hubungan antara ‘share’, ‘comment’, dan ‘like’ pada foto di Facebook

Prediktor linear (contoh)

Vektor bobot $\mathbf{w} \in \mathbb{R}^D$
bias: -20.24
share: 6.65
comment: 3.53

Vektor fitur $\phi(x) \in \mathbb{R}^D$
bias: 1
share: 147
comment: 58

$$\begin{aligned}\hat{y} &= \mathbf{w} \cdot \phi(x) \\ &= \sum_{j=1}^{D} w_j \phi_j(x) \\ &= -20.24(1) + 6.65(147) + 3.53(58) = 1162.05\end{aligned}$$

Jadi, *diprediksi* bahwa untuk foto dengan *share* $= 147$ dan *comment* $= 58$, foto tersebut akan mendapatkan ≈ 1162.05 *likes*.

Kita sudah tahu nilai y dan ϕ,
tapi berapa nilai $\mathbf{w}$?

Nyatanya, kita tidak bisa mencari nilai ϕ^{-1}

Loss Function

- ϕ bukan matriks bujur sangkar dan datanya mengandung *noise*
- Harus menggunakan *loss function* $O(\mathbf{w})$ yang dapat diminimalkan
- Pilihan umum: *squared error*

$$
\begin{aligned}
O(\mathbf{w}) &= \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \\
&= (\mathbf{y} - \phi\mathbf{w})^T (\mathbf{y} - \phi\mathbf{w})
\end{aligned}
$$

Solusi

- Jawaban: Minimalkan $O(\mathbf{w}) = \sum_{i=1}^{n}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$ dengan mencari turunan parsial yang diatur sama dengan 0
- Solusi analitis:

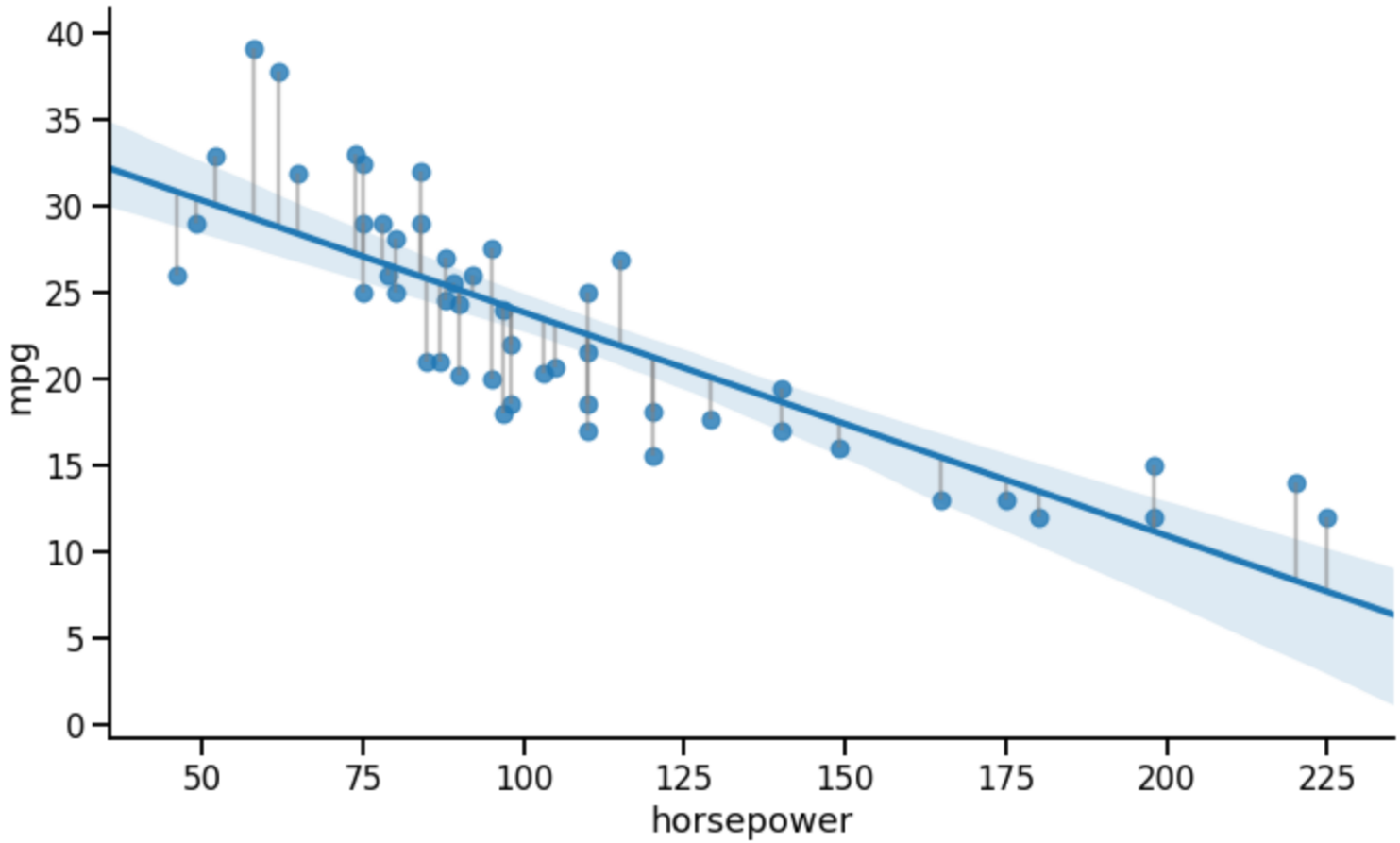
$$\hat{\mathbf{w}} = (\phi^T\phi)^{-1}\phi^T\mathbf{y}$$

- Bagian $(\phi^T\phi)^{-1}\phi^T$ dikenal sebagai *pseudo-inverse*

Multicollinearity

Apakah prediksi selalu lebih baik saat prediktornya ditambah?

Kembali ke contoh mobil



Gambar: $\text{mpg} = 36.75 - 0.13$ horsepower

Mean Absolute Error (MAE) = 3.21

Mari tambahkan variabel prediktor!

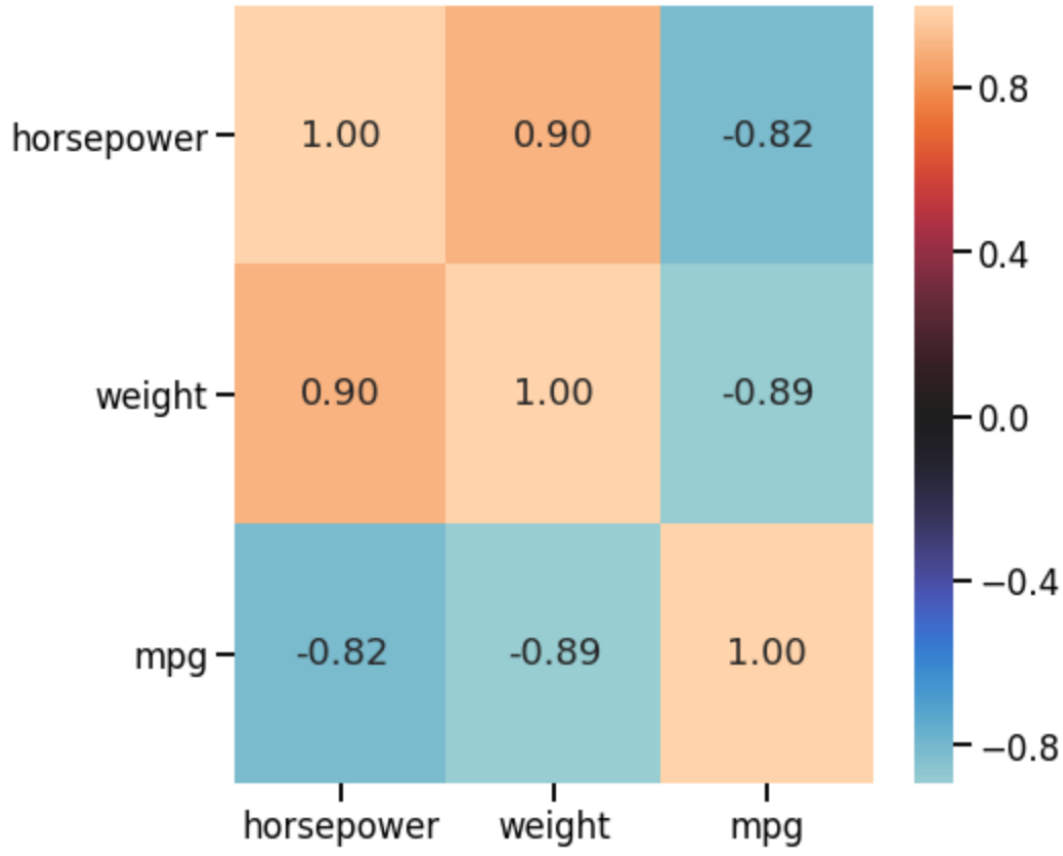
$$\text{mpg} = 43.99 - 0.01 \text{ horsepower} - 0.01 \text{ weight}$$

$$\text{MAE} = 2.38$$

Perhatikan kembali koefisiennya!

Meski MAE lebih rendah, koefisiennya menjadi semakin kecil, yang artinya variabel prediktornya makin tidak berguna(?)

Multikolinearitas



Gambar: Heatmap korelasi antarvariabel, ‘horsepower’ dan ‘mpg’ berkorelasi negatif kuat

Multikolinearitas

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	43.9918	1.751	25.120	0.000	40.469	47.515
horsepower	-0.0077	0.024	-0.322	0.749	-0.056	0.041
weight	-0.0067	0.001	-5.614	0.000	-0.009	-0.004

Omnibus:	3.240	**Durbin-Watson:**	2.367
Prob(Omnibus):	0.198	**Jarque-Bera (JB):**	2.550
Skew:	0.550	**Prob(JB):**	0.279
Kurtosis:	3.126	**Cond. No.**	1.26e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.26e+04. This might indicate that there are strong multicollinearity or other numerical problems.

Gambar: Pustaka statsmodels dapat mendeteksi kasus multikolinearitas

StatsModels

```
import statsmodels.formula.api as smf

model = smf.ols(
  'mpg ~ horsepower + weight',
  data=df
).fit()
model.summary()
```

Prediksi tinggi badan

Asumsikan kita ingin membuat model untuk memprediksi tinggi badan dari panjang kaki

$$\text{height} = 44.71 + 1.62\ \text{left\_leg}$$

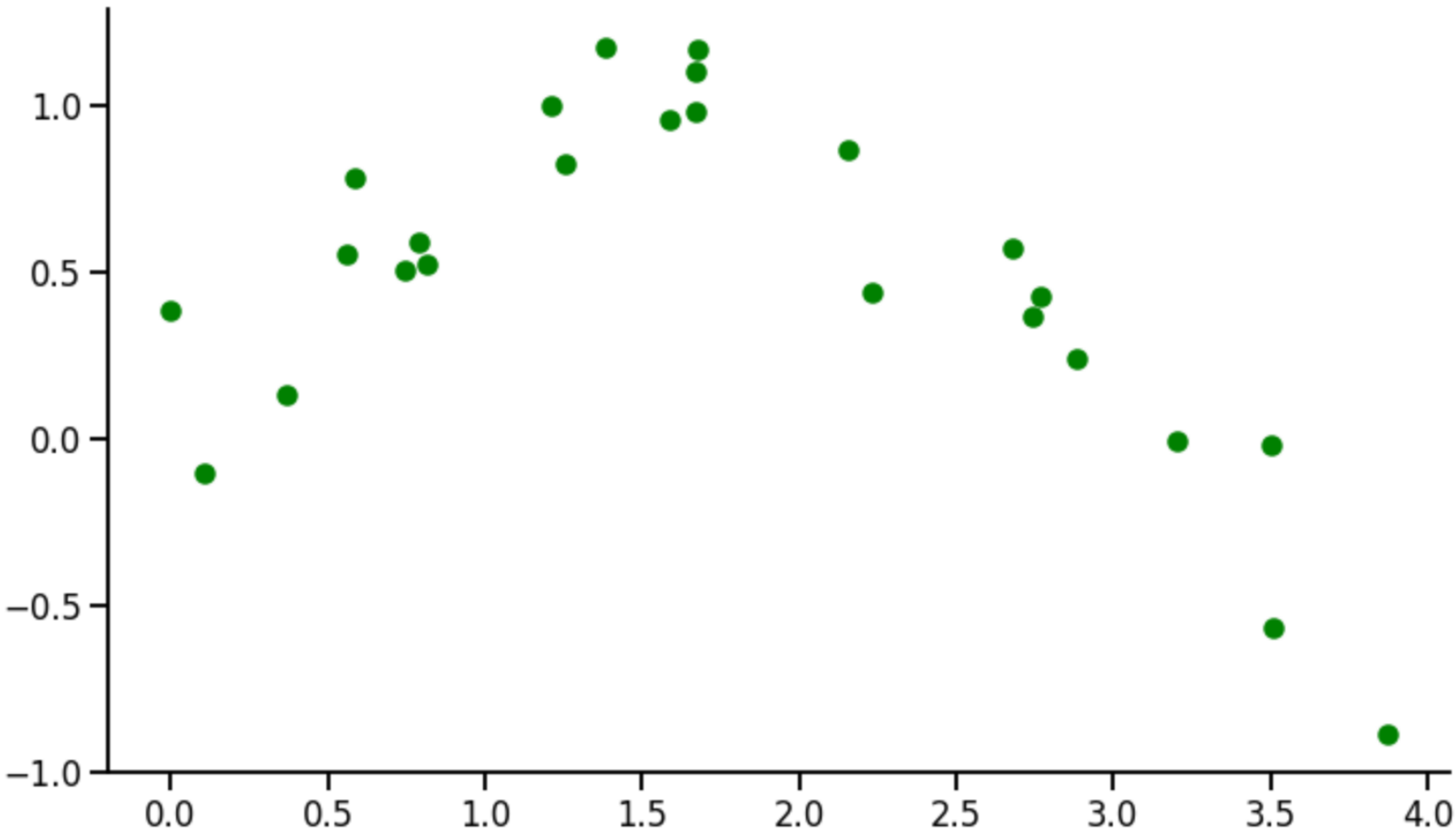
Multikolinearitas

$$\text{height} = 44.57 - 19.27\ \text{leg\_left} + 20.88\ \text{leg\_right}$$

Dengan kata lain: kalau kaki kiri lebih panjang, tinggi badan akan lebih pendek(?)

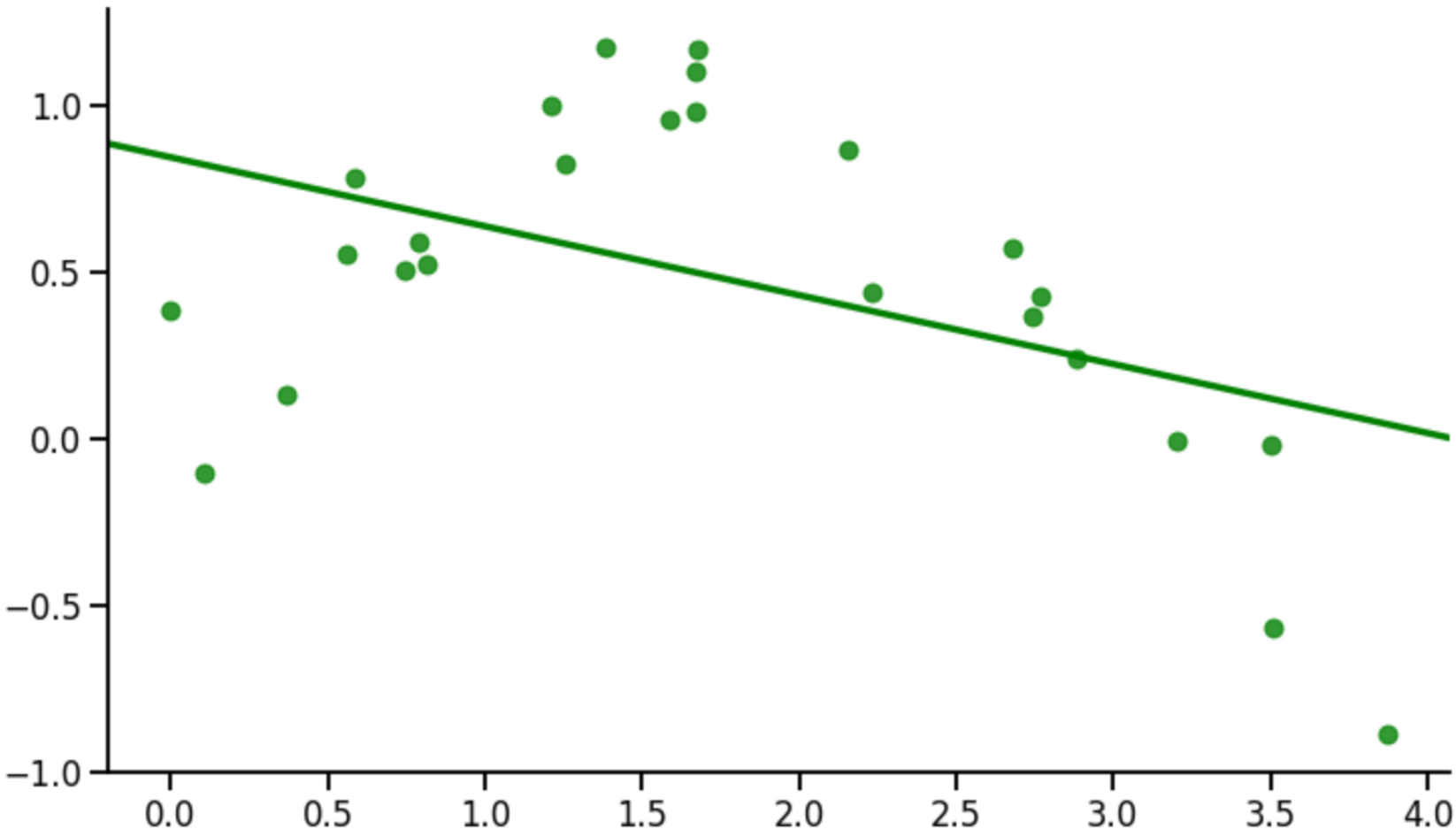
Basis Function Regression

Non-linearity



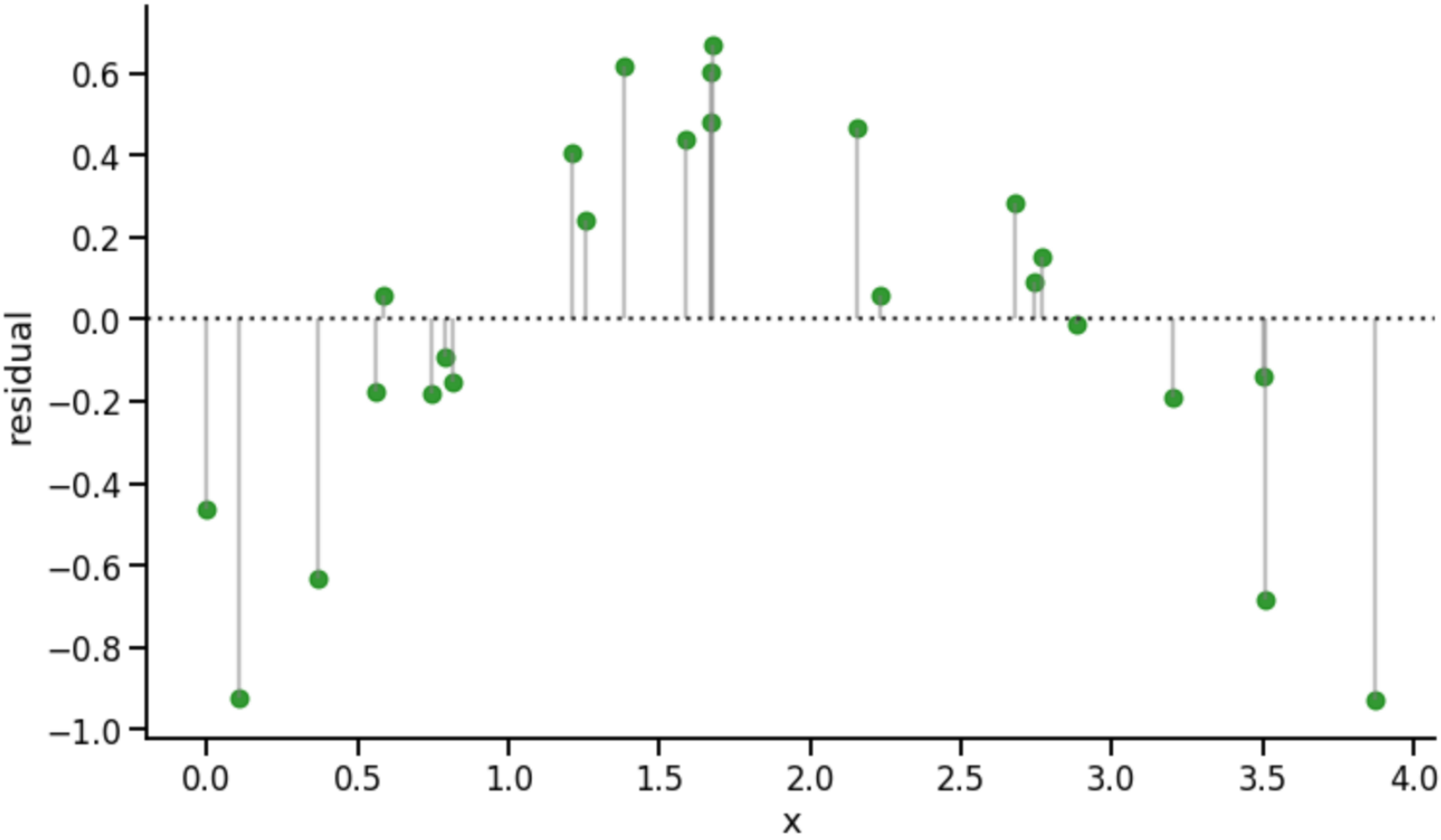
Gambar: Data yang dihasilkan dari fungsi sin dengan *noise*

Underfitting



Gambar: Hasil *fitting* regresi linear sederhana

Residual Plot



Gambar: Residual plot dari model yang mengalami *underfitting*

Jika model yang dihasilkan lebih sederhana
dibandingkan data yang seharusnya dicocokkan,
maka model tersebut disebut mengalami underfitting.

Polynomial Basis Functions

Regresi linear dengan fungsi basis polinomial

Jika kita mengubah $x_p = f_p(x)$, dengan $f_p()$ adalah fungsi transformasi, maka untuk $f_p() = x^p$ dan x adalah input berdimensi satu, modelnya menjadi

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + ...$$

Polynomial Basis Functions

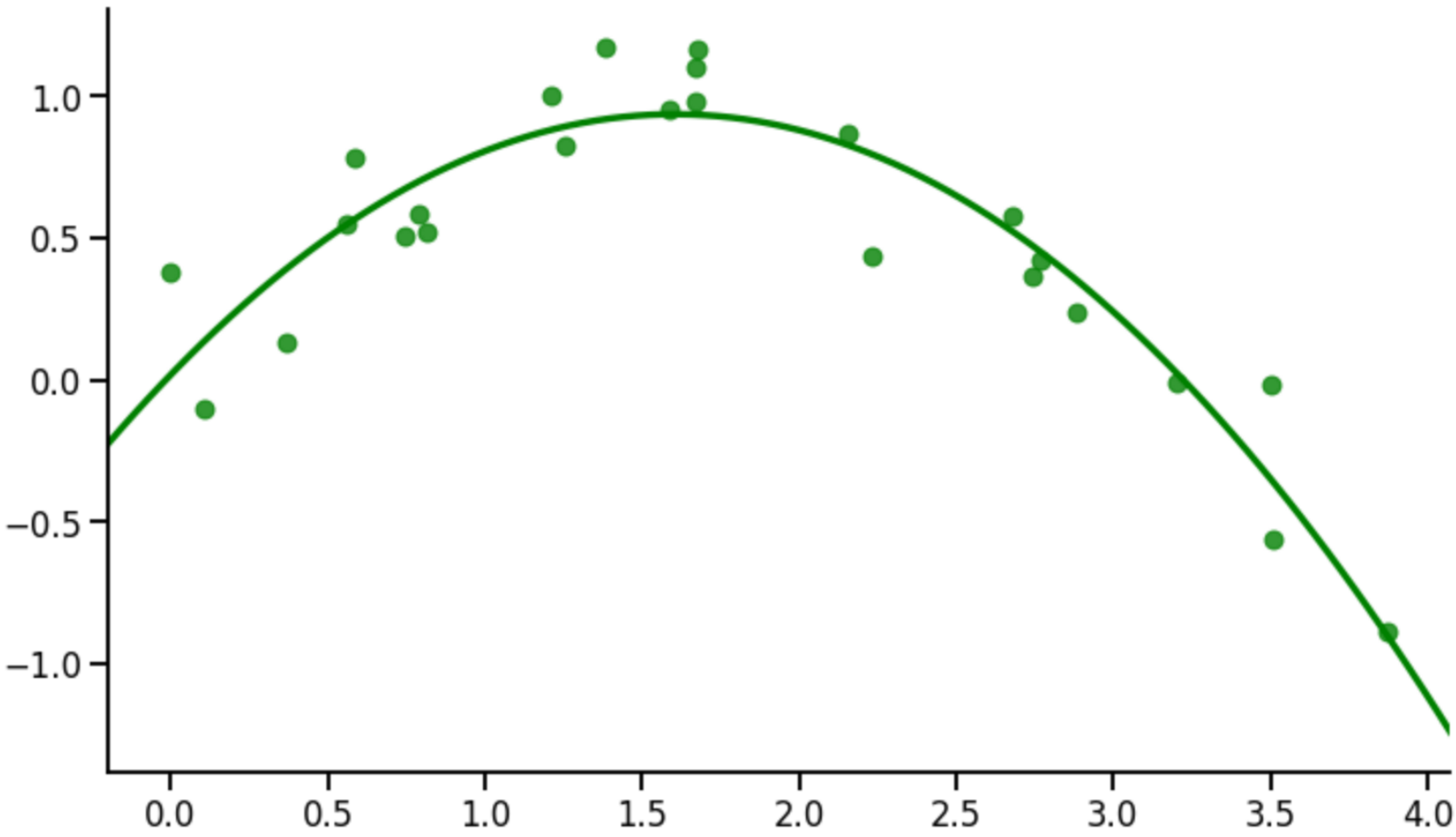
In

```
from sklearn.preprocessing import PolynomialFeatures
x = np.array([2, 3, 4])
poly = PolynomialFeatures(3, include_bias=False)
poly.fit_transform(x[:, None])
```

Out

```
array([[  2.,   4.,   8.],
[  3.,   9.,  27.],
[  4.,  16.,  64.]])
```

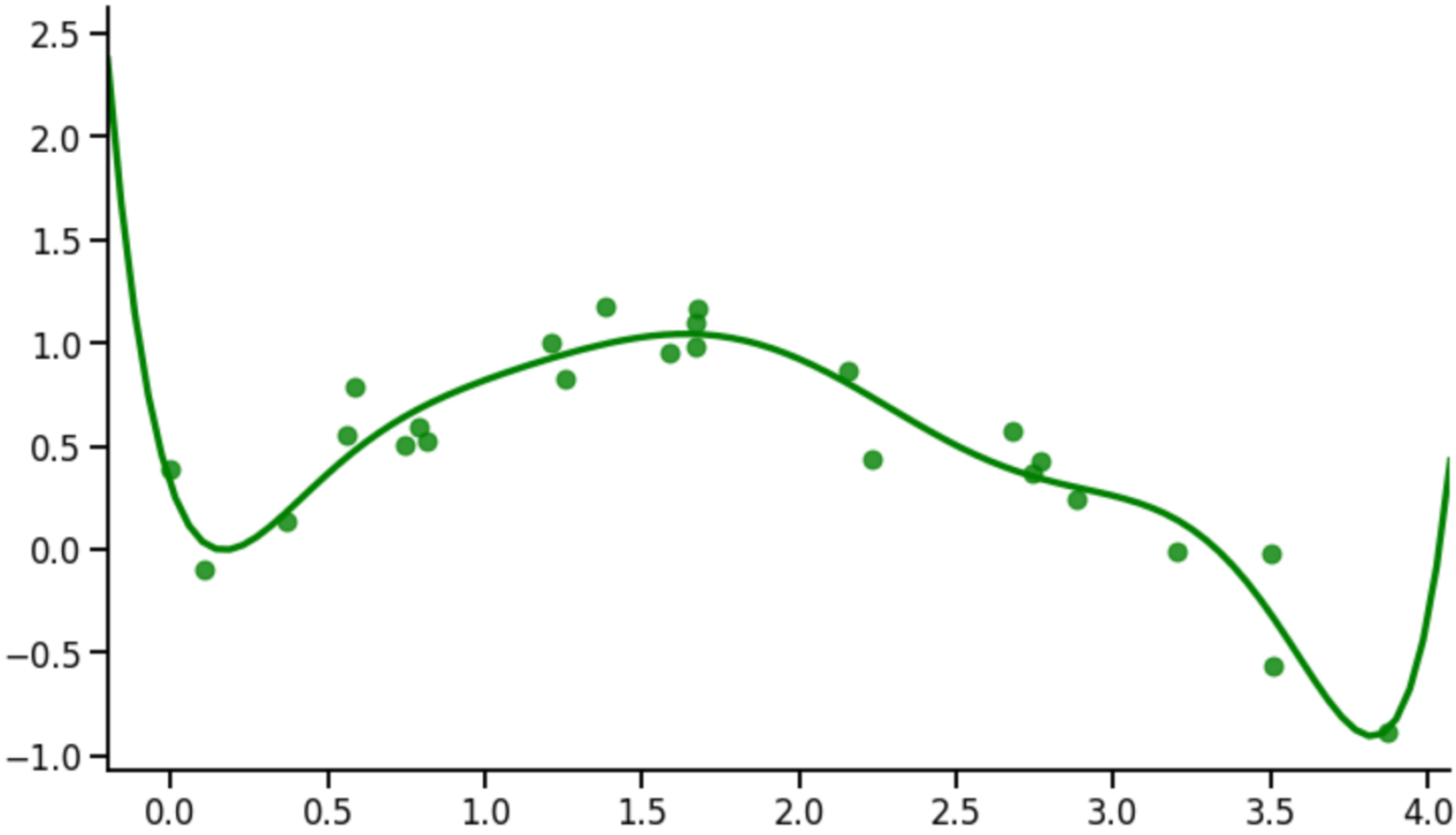
Best-fit



Gambar: Hasil *fitting* fungsi basis polinomial $p = 2$

Apa yang terjadi jika p dibuat lebih besar?

Overfitting



Gambar: Hasil *fitting* fungsi basis polinomial $p = 8$

Jika model yang dihasilkan lebih kompleks ($\sim$ parameternya banyak)
dibandingkan data yang seharusnya dicocokkan,
maka model tersebut disebut mengalami overfitting.

Regularisation

Bagaimana cara menghindari *overfitting*?

Ridge Regression

- Digunakan untuk menghindari *overfitting*
- Dikenal juga sebagai L_2 *regularisation* atau *Tikhonov regularisation*
- Pemberian penalti untuk koefisien model

$$P = \alpha \sum_{j=1}^{p} w_j^2 = \alpha \|\mathbf{w}\|_2^2$$

Loss Function pada Ridge Regression

- *Loss function* yang harus diminimalkan menjadi

$$O(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \alpha \|\mathbf{w}\|_2^2$$

dengan $\|\mathbf{w}\|_d = (\sum_{j=1}^{p} |w_j|^d)^{\frac{1}{d}}$

Loss Function pada Ridge Regression

- *Loss function* yang harus diminimalkan menjadi

$$O(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \alpha \|\mathbf{w}\|_2^2$$

  dengan $\|\mathbf{w}\|_d = (\sum_{j=1}^{p} |w_j|^d)^{\frac{1}{d}}$
- Parameter α (terkadang juga ditulis sebagai λ) bernilai bebas (ditentukan oleh pengguna)

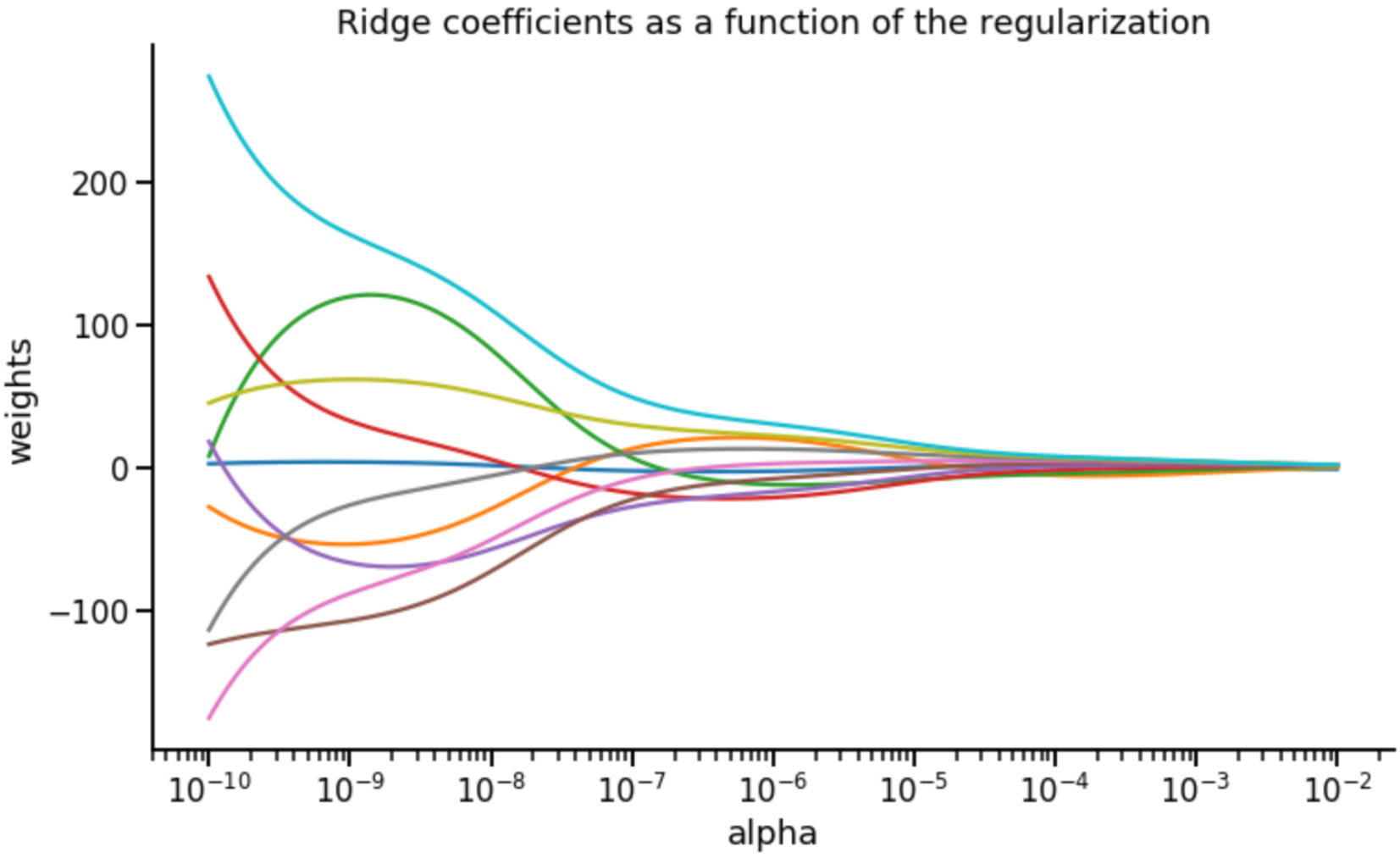
Loss Function pada Ridge Regression

- *Loss function* yang harus diminimalkan menjadi

$$O(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \alpha \|\mathbf{w}\|_2^2$$

  dengan $\|\mathbf{w}\|_d = (\sum_{j=1}^{p} |w_j|^d)^{\frac{1}{d}}$
- Parameter α (terkadang juga ditulis sebagai λ) bernilai bebas (ditentukan oleh pengguna)
- Solusi analitis:

$$\hat{\mathbf{w}} = (\boldsymbol{\phi}^T \boldsymbol{\phi} + \alpha I_p)^{-1} \boldsymbol{\phi}^T \mathbf{y}$$



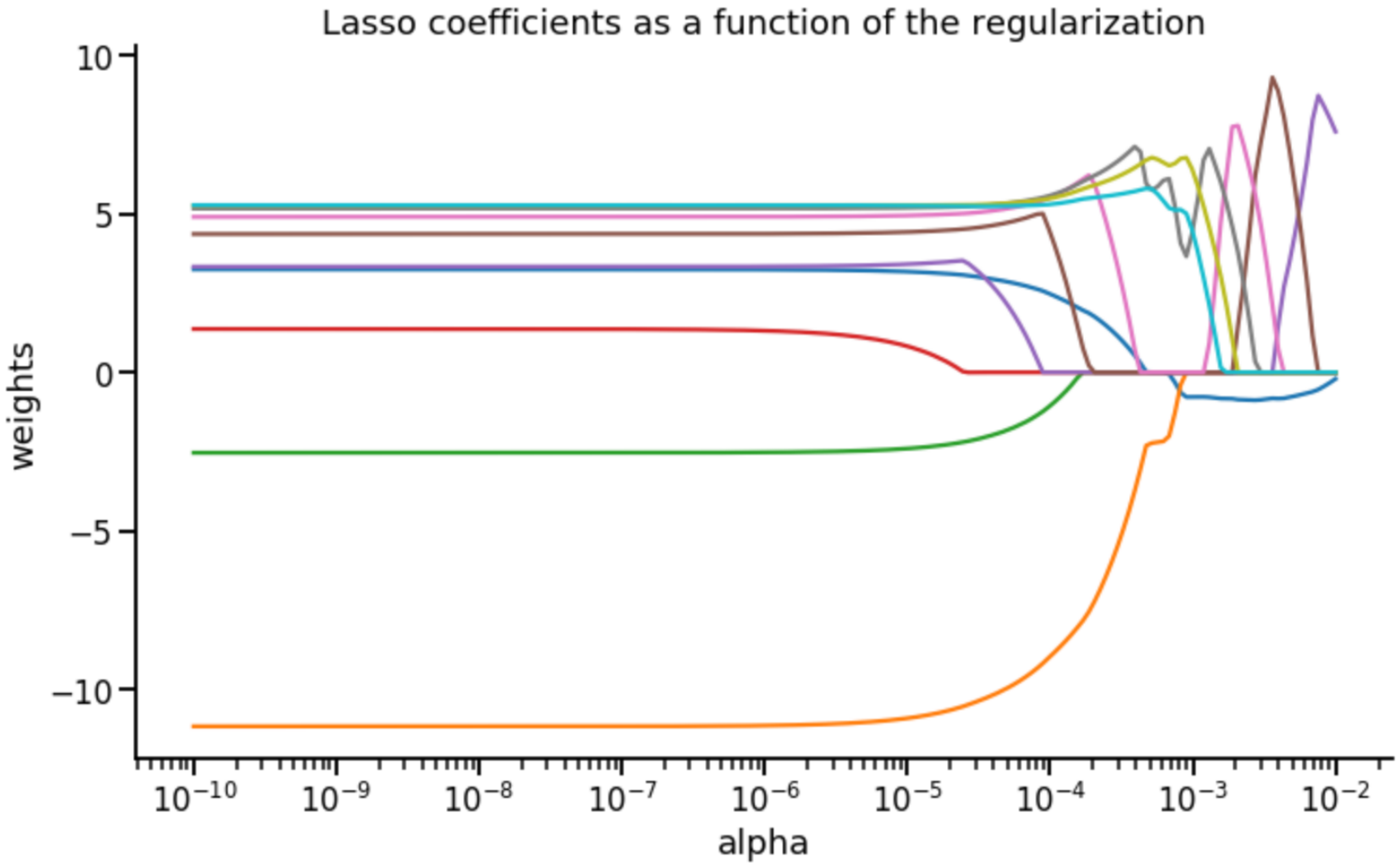
Gambar: Semakin besar nilai α, nilai koefisien mengerucut ke nol

Lasso Regression

- Secara konsep mirip seperti *ridge regression*
- Penalti dengan jumlah nilai absolut dari koefisien (1-norms; L_1 *regularisation*)

$$P = \alpha \sum_{j=1}^{p} |w_j|$$

- Bekerja dengan membuat banyak koefisien bernilai nol



Gambar: Penalti yang diberikan lasso lebih “keras”

Referensi

Jake VanderPlas (2016)

In Depth: Linear Regression

Python Data Science Handbook

Sebastian Raschka (2015)

Single-Layer Neural Networks and Gradient Descent

`http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html`

Terima kasih