

Klasyczny model regresji liniowej

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,m} \\ x_{2,1} & & x_{2,m} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,m} \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad B = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_m \end{bmatrix}$$

$$Y = XB + \varepsilon$$

$$\hat{Y} = X\hat{B}$$

$$e = Y - \hat{Y} = Y - X\hat{B} \quad (\text{błąd/reszty})$$

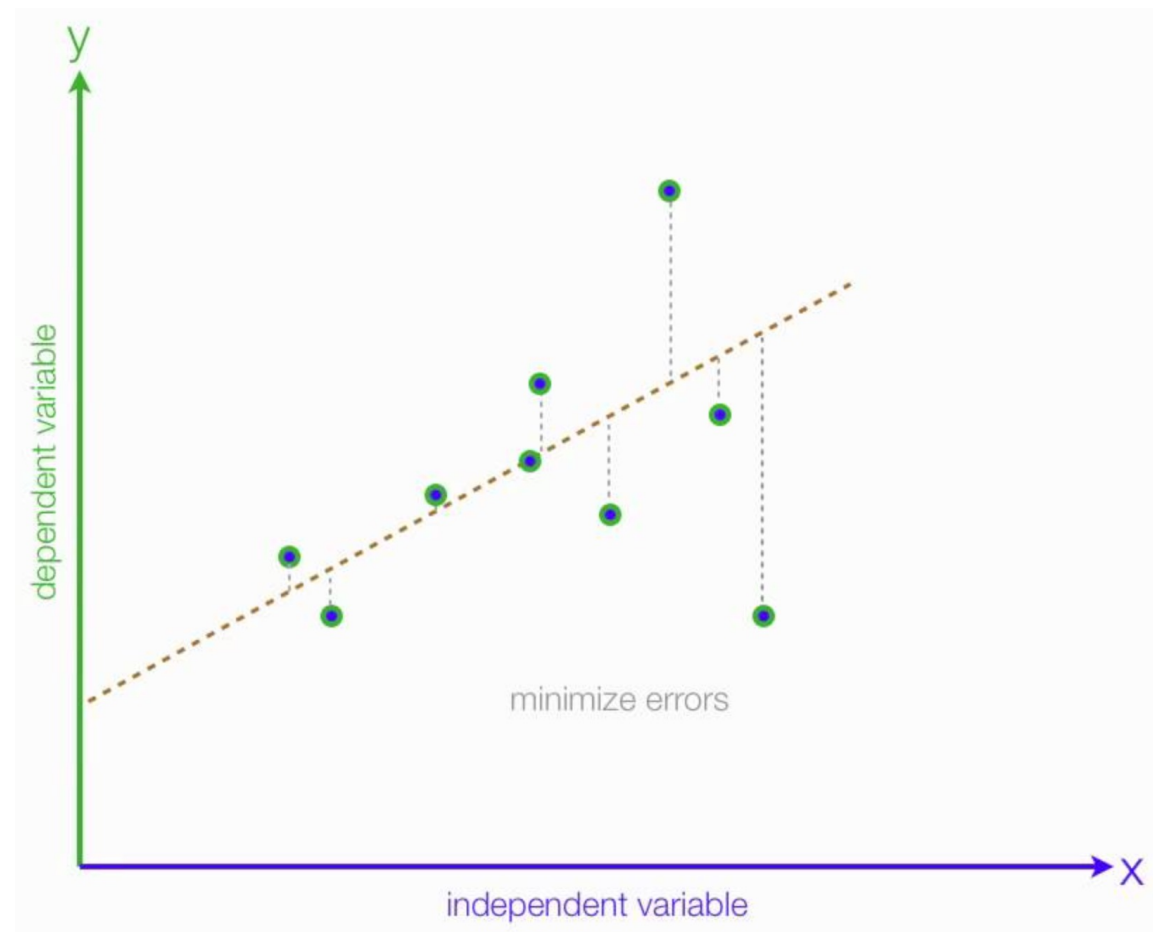
Metoda najmniejszych kwadratów (*ordinary least square/OLS*):

$$RSS = e^T e = (Y - X\hat{B})^T (Y - X\hat{B}) = (Y^T - \hat{B}^T Y^T)(Y - X\hat{B})$$

$$RSS = Y^T Y - 2\hat{B}^T X^T Y + \hat{B}^T X^T X \hat{B} \xrightarrow{B} \min$$

Równanie normalne:

$$\hat{B} = (X^T X)^{-1} X^T Y$$



Klasyczny model regresji liniowej

$$\hat{Y} = X\hat{B}$$

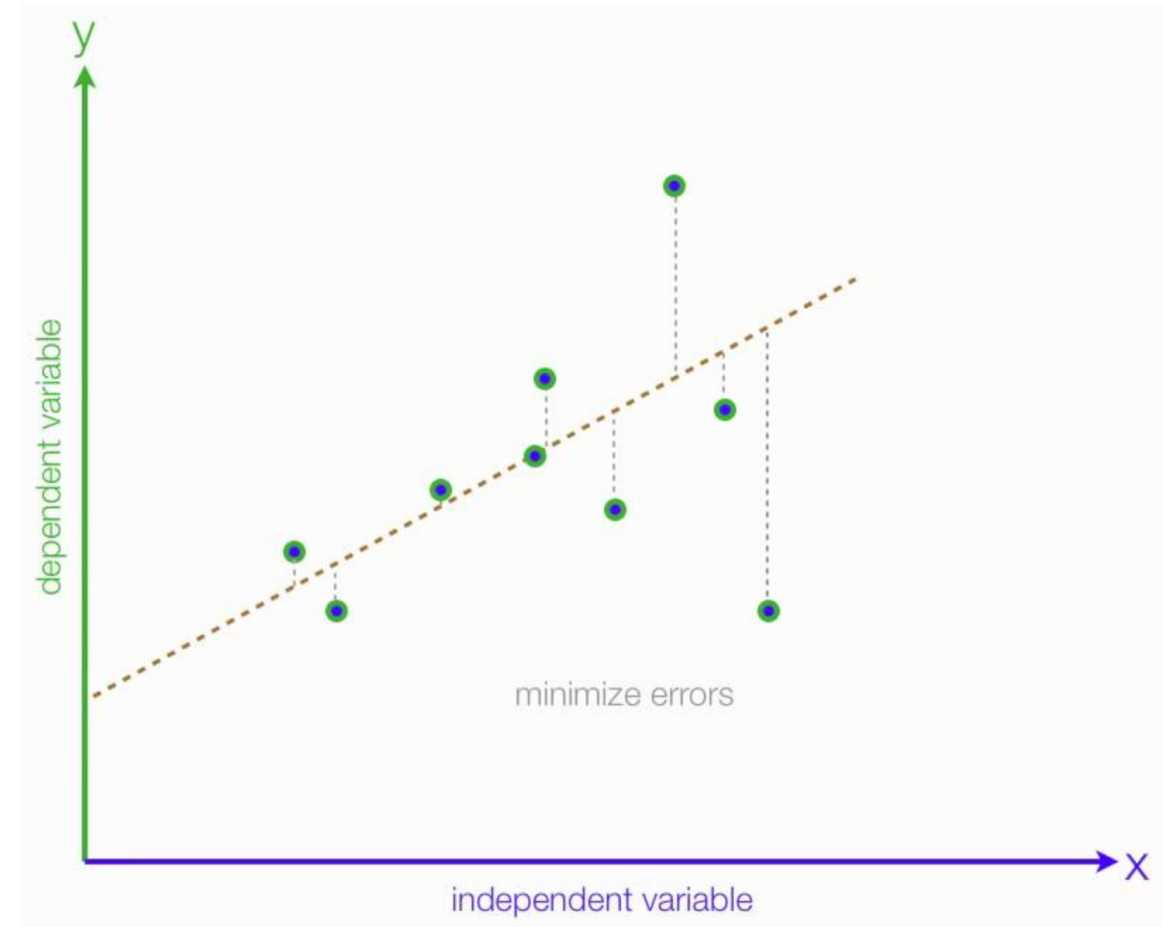
$$e = Y - \hat{Y} = Y - X\hat{B}$$

$$RSS = e^T e = (Y - X\hat{B})^T (Y - X\hat{B}) = (Y^T - \hat{B}^T Y^T)(Y - X\hat{B})$$

$$RSS = [e_1 \quad e_2 \quad \dots \quad e_n] \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} = e_1^2 + e_2^2 + e_3^2 + \dots + e_n^2$$

$$RSS = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - \mathbf{x}_i \hat{\mathbf{b}})^2$$

$$e = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix} = \begin{bmatrix} y_1 - \hat{b}_1 x_1 + b_0 \\ y_2 - \hat{b}_1 x_2 + b_0 \\ \vdots \\ y_n - \hat{b}_1 x_n + b_0 \end{bmatrix}$$



Klasyczny model regresji liniowej

1. Liniowość - zmienne losowe $Y_i, X_{i,k}$ należą do L^2 i spełniają:

$$Y = XB + \varepsilon$$

2. Ścisła egzogeniczność:

$$\mathbb{E}(\varepsilon|X) = 0$$

3. Liniowa niezależność obserwacji.

4. Sferyczność błędu, czyli:

1. Homoskedastyczność (stałość wariancji):

$$\mathbb{E}(\varepsilon^2|X) = \sigma^2$$

2. Brak korelacji reszt

$$\mathbb{E}(\varepsilon_i \varepsilon_j | X) = 0$$

5. Gaussowskość błędu:

$$\varepsilon|X \sim \mathcal{N}$$

Twierdzenie Gaussa-Markova:

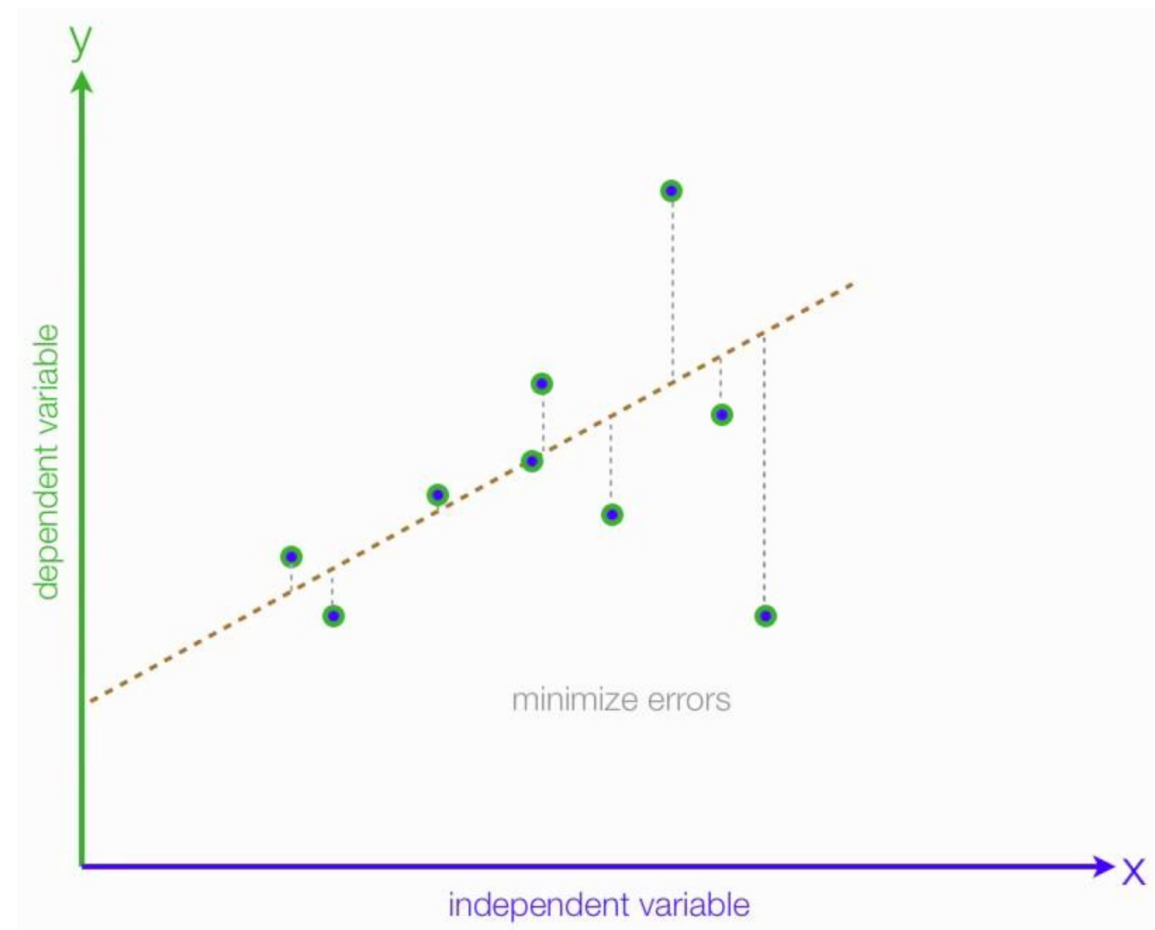
Jeśli zachodzą 1 – 4 to estymator najmniejszych kwadratów:

$$B = (X^T X)^{-1} X^T Y$$

jest:

najlepszym nieobciążonym liniowym estymatorem.

(BLUE: best linear unbiased estimator)



Klasyczny model regresji liniowej

Ocena jakości modelu R^2 ; *adjusted* R^2

$$R^2 := \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Współczynnik determinacji, inaczej zwany współczynnikiem określoności lub R-kwadrat jest miarą tego, jaki procent zmienności zmiennej zależnej (objaśnianej) jest wyjaśniany za pomocą zmiennej niezależnej (czynnik zmienna objaśniająca, predyktor) bądź modelu statystycznego. Innymi słowy, współczynnik determinacji informuje nas, ile nasz model, nasz badany czynnik wyjaśnia zgromadzone dane pomiarowe (zmienną zależną).

Z R^2 (R-kwadrat) wiąże się z nieodłącznym problemem – dodatkowe zmienne wejściowe sprawiają, że R-kwadrat pozostanie taki sam lub wzrośnie (jest to spowodowane tym, jak R-kwadrat jest obliczany matematycznie). Dlatego nawet jeśli dodatkowe zmienne wejściowe nie wykazują związku ze zmiennymi wyjściowymi, R-kwadrat wzrośnie.

$$R_{adj}^2 = 1 - (1 - R^2) \frac{n - 1}{n - k - 1}$$

n – ilość obserwacji

k – ilość zmiennych objaśniających (bez wyrazu wolnego)

Metoda Najmniejszych Kwadratów

Złożoność obliczeniowa

Równanie normalne:

$$\hat{B} = (X^T X)^{-1} X^T Y$$

Najbardziej złożone obliczeniowo jest znalezienie macierzy odwrotnej:

$$W^{-1} = (X^T X)^{-1}$$

Wyznaczanie:

Metoda dopełnień algebraicznych

$$W^{-1} = \frac{1}{\det W} (W^D)^T$$

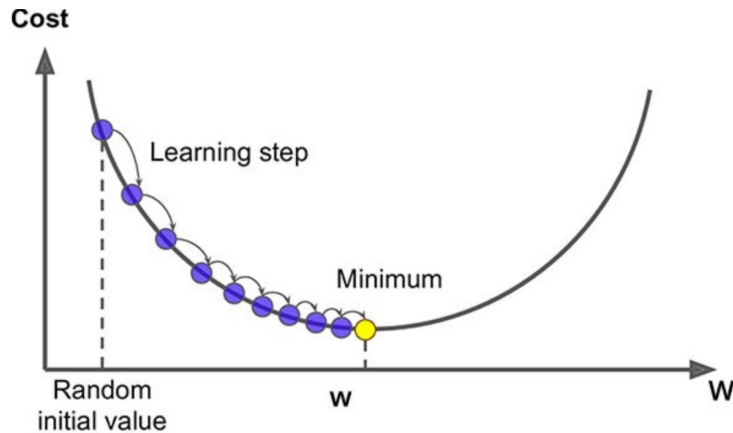
Metoda eliminacji Gaussa-Jordana

$$[W|I] \mapsto [I|W^{-1}]$$

W zależności od implementacji algorytmu złożoność obliczeniowa odwrócenia macierzy wynosi zazwyczaj od około $O(n^{2.4})$ do $O(n^3)$.

Podwojenie liczby kolumn wydłuża czas obliczeń o około od $2^{2.4} = 5.3$ do $2^3 = 8$.

Gradient prosty *Gradient descent*



$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

η – learning rate (0.01)

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & \cdots & x_{2,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

$$\mathbf{x}_i = [x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,n}]$$

$$RSS(\theta) = \mathbf{e}^T \mathbf{e} = (\mathbf{Y} - \mathbf{X}\theta)^T (\mathbf{Y} - \mathbf{X}\theta) = (\mathbf{Y}^T - \theta^T \mathbf{Y}^T) (\mathbf{Y} - \mathbf{X}\theta)$$

$$RSS(\theta) = \mathbf{Y}^T \mathbf{Y} - 2\theta^T \mathbf{X}^T \mathbf{Y} + \theta^T \mathbf{X}^T \mathbf{X} \theta$$

$$\frac{\partial}{\partial \theta} RSS(\theta) = -2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X} \theta = 2\mathbf{X}^T (\mathbf{X}\theta - \mathbf{Y})$$

$$\frac{1}{m} RSS(\theta) = J(\theta)$$

```
for iteration in range(n_iterations):
    gradients = 2/m * X.T.dot(X.dot(theta) - y)
    theta = theta - eta * gradients
```

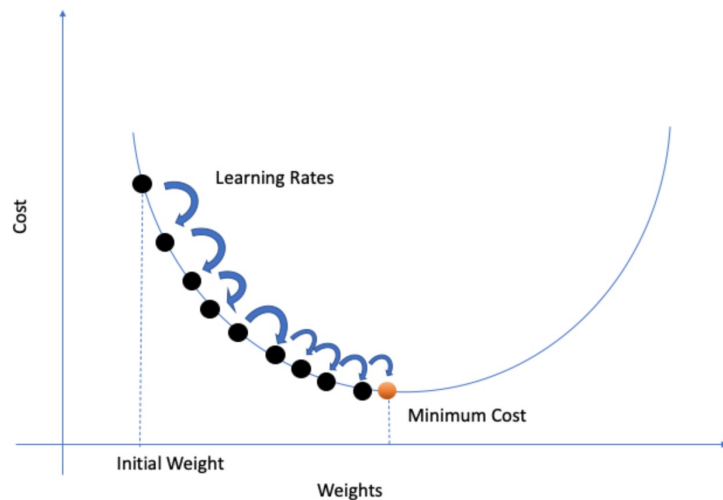
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i)^2$$

$$h_{\theta}(x_i) = \boldsymbol{\theta}^T \mathbf{x}_i$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\boldsymbol{\theta}^T \mathbf{x}_i - y_i)^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m 2(\boldsymbol{\theta}^T \mathbf{x}_i - y_i) x_{i,j} = \frac{1}{m} \sum_{i=1}^m (\boldsymbol{\theta}^T \mathbf{x}_i - y_i) x_{i,j}$$

Stochastyczny Gradient *Stochastic Gradient Descent*



$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J_i(\theta_0, \theta_1)$$

η – learning rate (0.01)

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & \cdots & x_{2,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

$$\mathbf{x}_i = [x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,n}]$$

i – ustalone, losowe

$$J_i(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i)^2$$

$$J_i(\theta_0, \theta_1) = (h_{\theta}(x_i) - y_i)^2$$

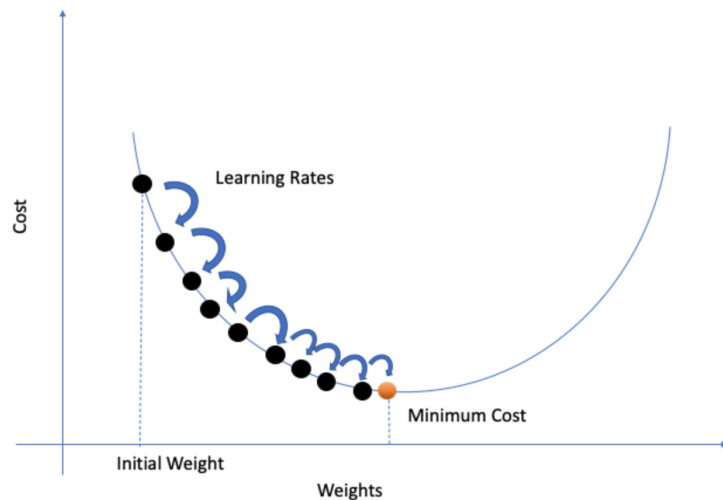
$$h_{\theta}(x_i) = \boldsymbol{\theta}^T \mathbf{x}_i$$

$$J_i(\theta_0, \theta_1) = (\boldsymbol{\theta}^T \mathbf{x}_i - y_i)^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = 2(\boldsymbol{\theta}^T \mathbf{x}_i - y_i) x_{i,j}$$

```
for epoch in range(n_epochs):
    for i in range(m):
        random_index = np.random.randint(m)
        xi = X[random_index:random_index+1]
        yi = y[random_index:random_index+1]
        gradients = 2 * xi.T.dot(xi.dot(theta) - yi)
```

Gradient z minigrupami *Mini-batch Gradient Descent*



$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J_I(\theta_0, \theta_1)$$

η – learning rate (0.01)

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & \cdots & x_{2,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

$$\mathbf{x}_i = [x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,n}]$$

i – ustalone, losowe

$$J(\theta_0, \theta_1) = \frac{1}{2|I|} \sum_{i \in I} (h_{\theta}(\mathbf{x}_i) - y_i)^2$$

$$h_{\theta}(\mathbf{x}_i) = \boldsymbol{\theta}^T \mathbf{x}_i$$

$$J(\theta_0, \theta_1) = \frac{1}{2|I|} \sum_{i \in I} (\boldsymbol{\theta}^T \mathbf{x}_i - y_i)^2$$

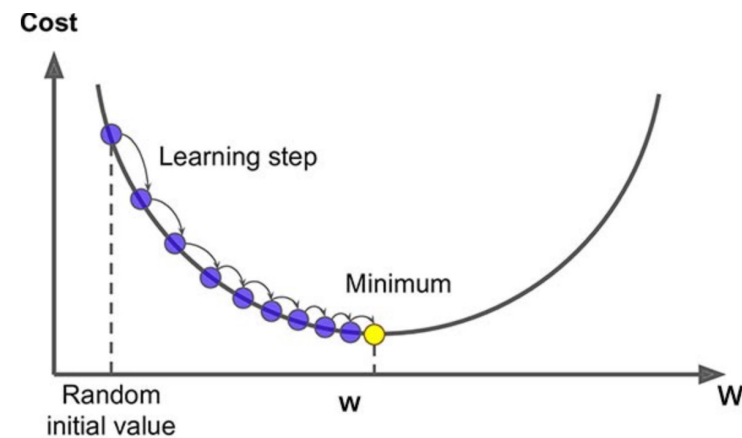
$$\frac{\partial}{\partial \theta_j} J_I(\theta_0, \theta_1) = \frac{1}{2|I|} \sum_{i \in I} 2(\boldsymbol{\theta}^T \mathbf{x}_i - y_i) x_{i,j} = \frac{1}{2|I|} \sum_{i \in I} (\boldsymbol{\theta}^T \mathbf{x}_i - y_i) x_{i,j}$$

Spadek po gradiencie

Gradient prosty
Gradient descent

Stochastyczny gradient
Stochastic Gradient Descent

Gradient z minigrupami
Mini-batch Gradient Descent



$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

η – learning rate (0.01)

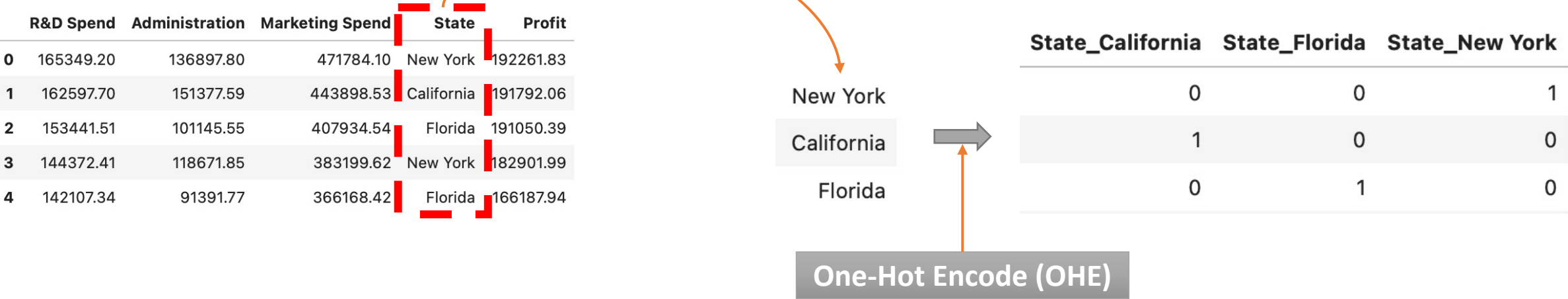
Obliczamy koszt i pochodną dla całego zbioru obserwacji x .

Obliczamy koszt i pochodną dla losowo wybranej obserwacji x .

Obliczamy koszt i pochodną dla losowo wybranego podzbioru (batch) obserwacji x .

Zmienne katagoryczne/dyskretne *Dummy Variable*

One-Hot Encode (OHE)



	R&D Spend	Administration	Marketing Spend	Profit	State_California	State_Florida	State_New York
0	165349.20	136897.80	471784.10	192261.83	0	0	1
1	162597.70	151377.59	443898.53	191792.06	1	0	0
2	153441.51	101145.55	407934.54	191050.39	0	1	0
3	144372.41	118671.85	383199.62	182901.99	0	0	1
4	142107.34	91391.77	366168.42	166187.94	0	1	0

Pułapka zmiennych kategorycznych *Dummy Variable Trap*

Dla modelu regresji

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

	R&D Spend	Administration	Marketing Spend	Profit	State_California	State_Florida	State_New York
0	165349.20	136897.80	471784.10	192261.83	0	0	1
1	162597.70	151377.59	443898.53	191792.06	1	0	0
2	153441.51	101145.55	407934.54	191050.39	0	1	0
3	144372.41	118671.85	383199.62	182901.99	0	0	1
4	142107.34	91391.77	366168.42	166187.94	0	1	0

R&D Spend	Administration	Marketing Spend	Profit	State_California	State_Florida	State_New York	intercept
165349.20	136897.80	471784.10	192261.83	0	0	1	1
162597.70	151377.59	443898.53	191792.06	1	0	0	1
153441.51	101145.55	407934.54	191050.39	0	1	0	1
144372.41	118671.85	383199.62	182901.99	0	0	1	1
142107.34	91391.77	366168.42	166187.94	0	1	0	1

$$B = (X^T X)^{-1} X^T Y$$

$$\det(X^T X) = 0$$
$$(X^T X)^{-1} - \textit{nie istnieje}$$

Zmienne kategoryczne/dyskretne

Inne metody

https://contrib.scikit-learn.org/category_encoders/

```
pip install category_encoders
```

```
import category_encoders as ce

encoder = ce.BackwardDifferenceEncoder(cols=[...])
encoder = ce.BaseNEncoder(cols=[...])
encoder = ce.BinaryEncoder(cols=[...])
encoder = ce.CatBoostEncoder(cols=[...])
encoder = ce.CountEncoder(cols=[...])
encoder = ce.GLMMEncoder(cols=[...])
encoder = ce.HashingEncoder(cols=[...])
encoder = ce.HelmertEncoder(cols=[...])
encoder = ce.JamesSteinEncoder(cols=[...])
encoder = ce.LeaveOneOutEncoder(cols=[...])
encoder = ce.MEstimateEncoder(cols=[...])
encoder = ce.OneHotEncoder(cols=[...])
encoder = ce.OrdinalEncoder(cols=[...])
encoder = ce.SumEncoder(cols=[...])
encoder = ce.PolynomialEncoder(cols=[...])
encoder = ce.TargetEncoder(cols=[...])
encoder = ce.WOEEncoder(cols=[...])
encoder = ce.QuantileEncoder(cols=[...])

encoder.fit(X, y)
X_cleaned = encoder.transform(X_dirty)
```

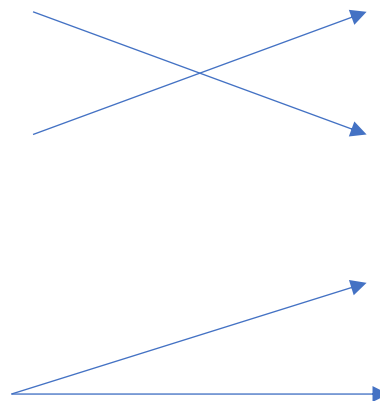
Regresyjne Lasy Losowe (*Bagging*)

Step 1: Bootstrapping

Losujemy obserwacje ze zbioru danych aby stworzyć zbiór o tym samym rozmiarze.

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Sugar
No	No	No	125	90.01
Yes	Yes	Yes	180	121.3
Yes	Yes	No	210	96.50
Yes	No	Yes	167	137.67



Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Sugar
Yes	Yes	Yes	180	121.3
No	No	No	125	90.01
Yes	No	Yes	167	137.67
Yes	No	Yes	167	137.67

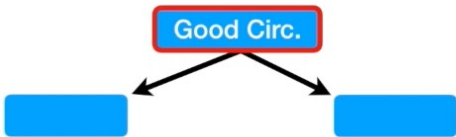
Regresyjne Lasy Losowe

Step 2: Tworzymy drzewo losowe na wybranym zbiorze używając losowego podzbioru kolumn.

Losowo wybieramy podzbiór



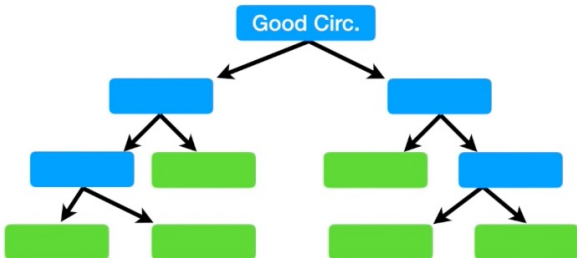
Znajdujemy najlepszy podział korzenia.



Losowo wybieramy podzbiór dla węzła



Budujemy drzewo rozważając tylko podzbiór cech

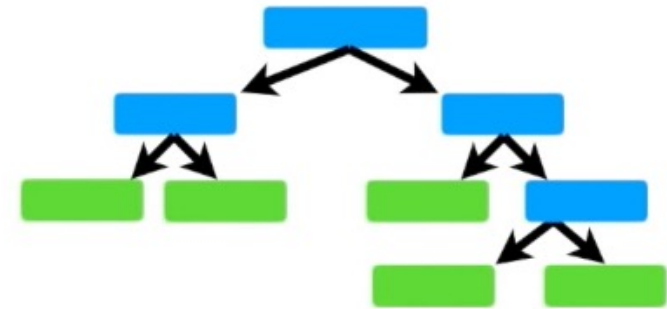
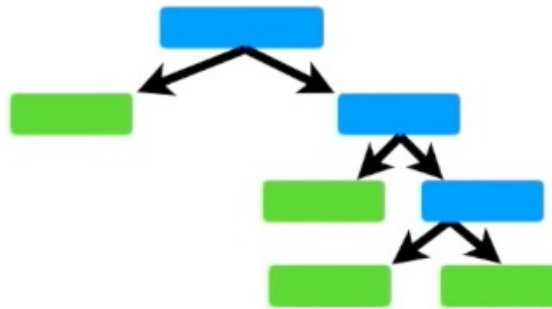
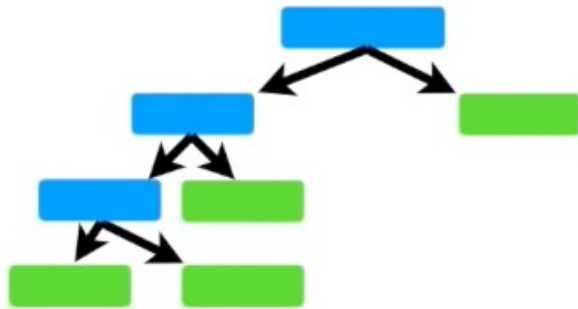
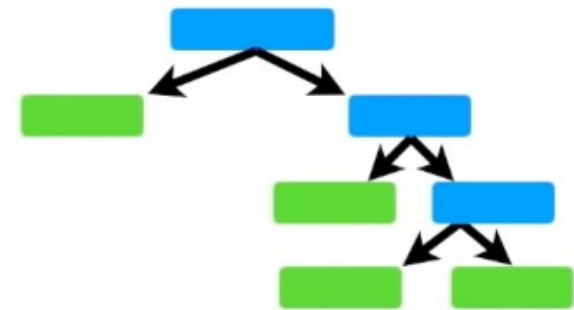
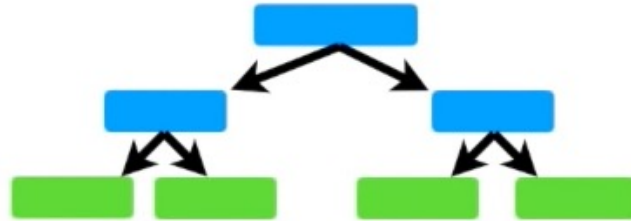
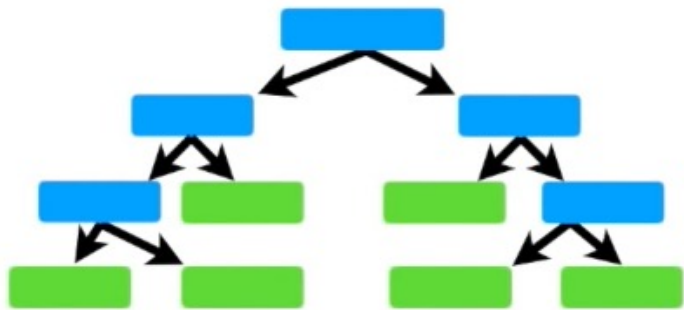


Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Sugar
Yes	Yes	Yes	180	121.3
No	No	No	125	90.01
Yes	No	Yes	167	137.67
Yes	No	Yes	167	137.67

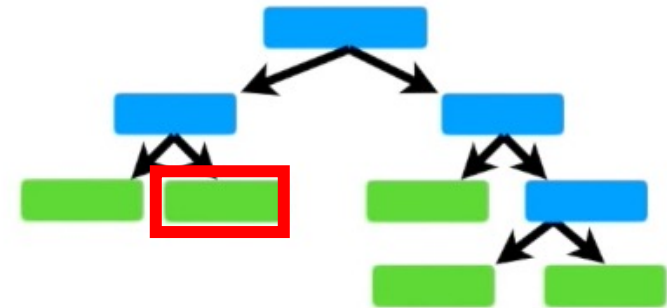
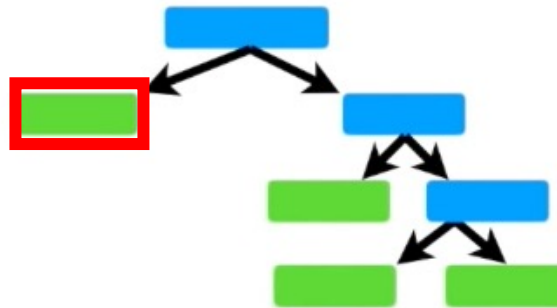
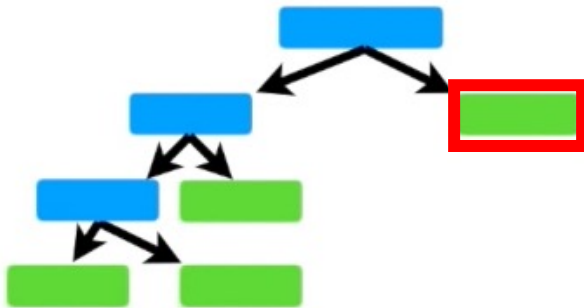
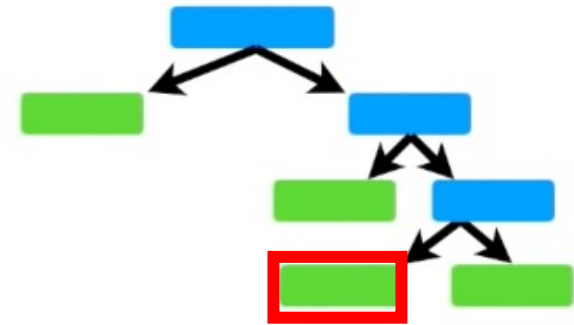
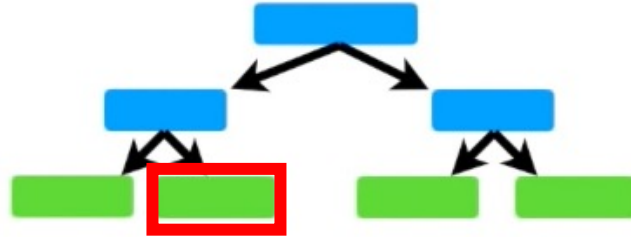
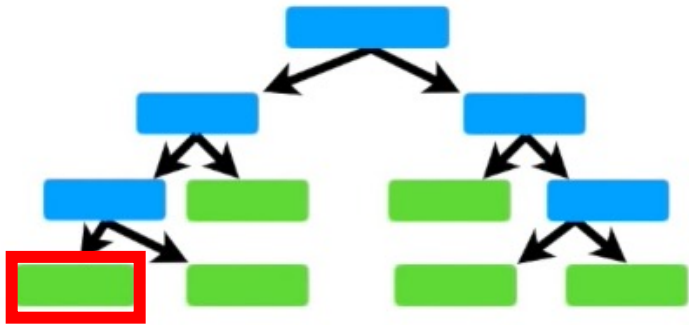
Regresyjne Lasy Losowe

Powtarzamy Step 1 i Step 2 tworząc kolejne drzewa.



Regresyjne Lasy Losowe

Wartość przewidywanej wartości to średnia ze wszystkich drzew.



Regresyjne Lasy Losowe

Out-Of-Bag Dataset

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Sugar
No	No	No	125	90.01
Yes	Yes	Yes	180	121.3
Yes	Yes	No	210	96.50
Yes	No	Yes	167	137.67

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Sugar
Yes	Yes	Yes	180	121.3
No	No	No	125	90.01
Yes	No	Yes	167	137.67
Yes	No	Yes	167	137.67

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Sugar
Yes	Yes	No	210	96.50

Możemy przewidzieć wartości dla obserwacji z OOB a następnie policzyć na nim pewną metrykę.

W bibliotece sklearn moduł `sklearn.ensemble.RandomForestRegressor` posiada atrybut **`oob_score_`**, który zwraca R^2

`oob_score_` : float

Score of the training dataset obtained using an out-of-bag estimate. This attribute exists only when `oob_score` is True.

Machine Learning:

- Nadzorowane
 - Klasyfikacja
 - Regresja
- Nienadzorowane
 - Klasteryzacja
 - Redukcja wymiarowości
 - *Detekcja anomalii

*Klasyczny Model Regresji Liniowej:

- Założenia modelu
- Metoda Najmniejszych Kwadratów (OLS)
- Twierdzenie Gaussa Markowa

Model:

- LinearRegression
- DecisionTreeRegressor
- *RandomForrestRegressor

Regresja Liniowa:

- Regresja jednowymiarowa (Simple Linear Regression)
- Regresja wielowymiarowa (multivariate Linear Regression)
- Regresja wielomianowa (polynomial Linear Regression)

Funkcja kosztu:

- Błąd średniokwadratowy

Spadek po gradiencie:

- Metoda Gradientu Prostego
- *Metoda Stochastycznego Gradientu
- *Metoda Gradientu z Mini-grupami

Jakość modelu:

- Przetrenowanie/Niedotrenowanie
- Bias-Variance Trade Off

Ocena modelu:

- train_test_split
- KFold
- LeaveOneOut

Metryki:

- R^2 – (r-squared)
- *adjR2 – (adjusted R^2)
- MAE, MAPE, WMAPE
- MSE, RMSE

*Regularyzacja:

- Lasso (L1)
- Ridge (L2)
- ElasticNet

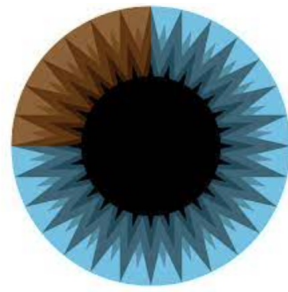
*Sklearn Pipeline

TODO:

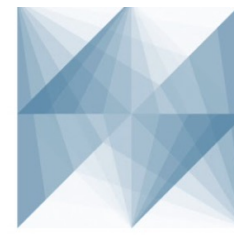
- AdaBoost
- Gradient Tree Boost (XGBoost)
- SVM Regressor



<https://machinelearningmastery.com>



<https://www.youtube.com/c/3blue1brown>

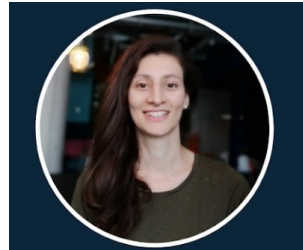


Machine Learning Study Groups

<https://www.youtube.com/channel/UCMEQFEKrsRFBXnUIreTACxg>

towards data science

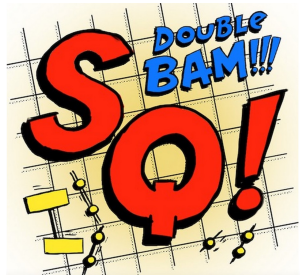
<https://towardsdatascience.com>



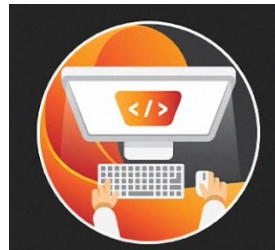
<https://www.youtube.com/c/TechWorldwithNana>



<https://www.youtube.com/c/TensorFlow>



<https://www.youtube.com/c/joshstarmar>



<https://www.youtube.com/c/TechWithTim>

kaggle

