

ML to proces wykorzystania danych (przyrodniczych, wytworzonych przez człowieka, wygenerowanych przez algorytm) do budowy **modeli**, które pozwalają na przeprowadzanie prognozowania lub praktycznego rozwiązania.

Modele wykorzystują dane i ich strukturę a nie reguły if-then-else .

Więcej danych i polepszanie się ich jakości będzie powodowało lepsze **modele** dające lepsze prognozy i rozwiązania.

Rodzaje ML to:

supervised, semi-supervised, unsupervised i reinforcement learning.

Co to jest Machine Learning?

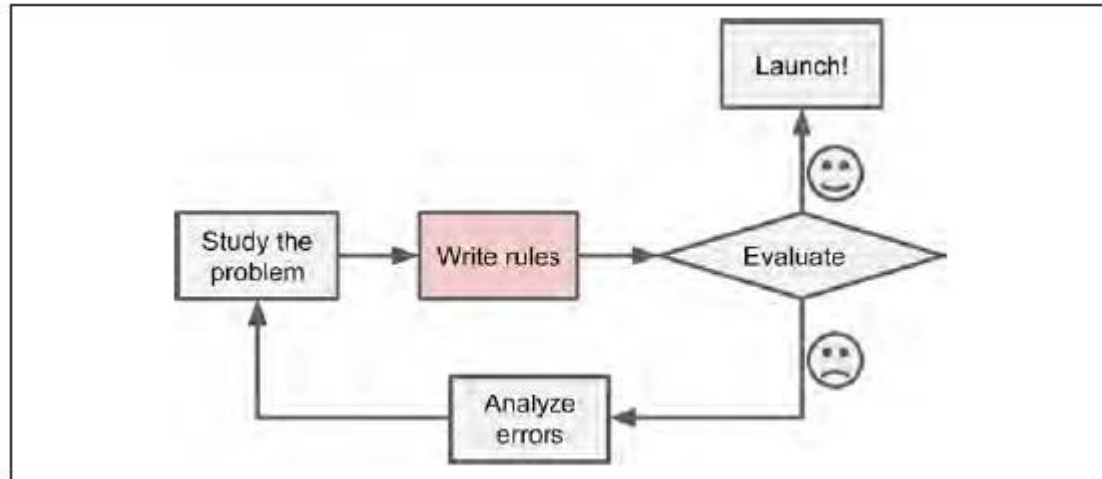


Figure 1-1. The traditional approach

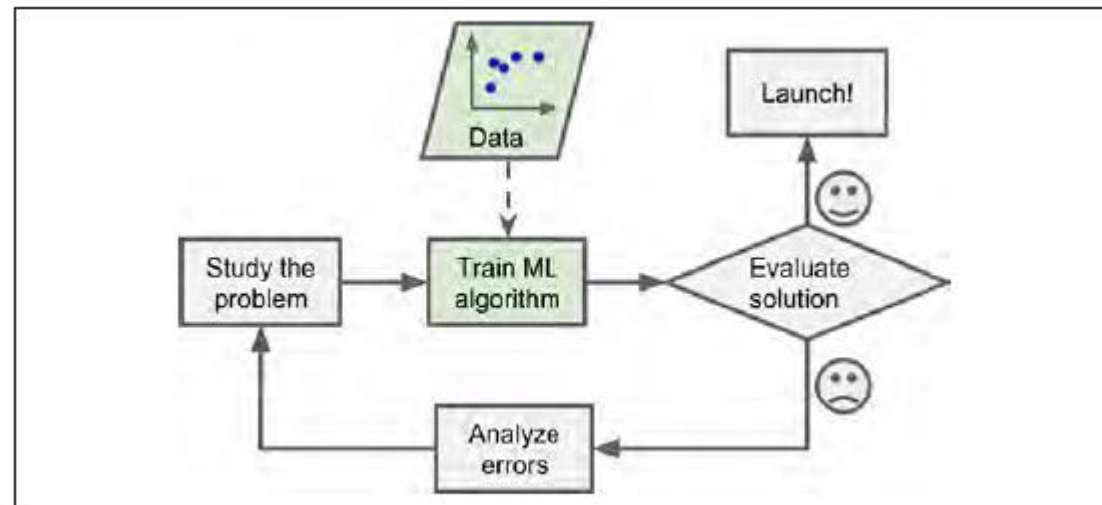


Figure 1-2. Machine Learning approach

Zastosowanie technik ML do analizy dużych ilości danych pozwala wykrywać niewidoczne zależności - **data mining**.

ML jest przydatne do:

- Problemów dla których istniejące rozwiązania wymagają uciążliwego ręcznego dopasowywania lub długiej listy reguł. Pojedynczy ML algorytm może dać prostszy kod i lepsze wyniki.
- Złożone problemy dla których brak jest dobrego rozwiązania przy zastosowaniu tradycyjnych metod.
- Środowiska z ciągłym dopływem danych.
- Poszukiwanie złożonych zależności w dużych ilościach danych.

Instance-based learning

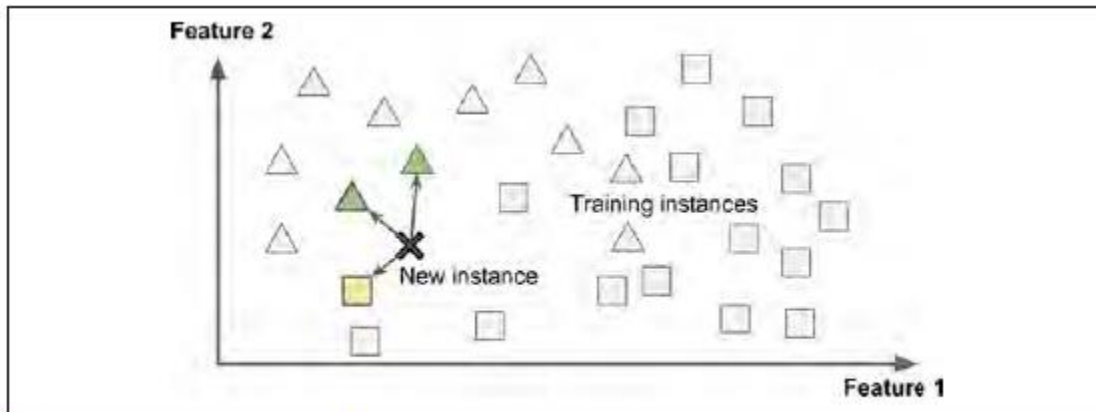


Figure 1-15. Instance-based learning

wymaga *measure of similarity* between

Model-based learning

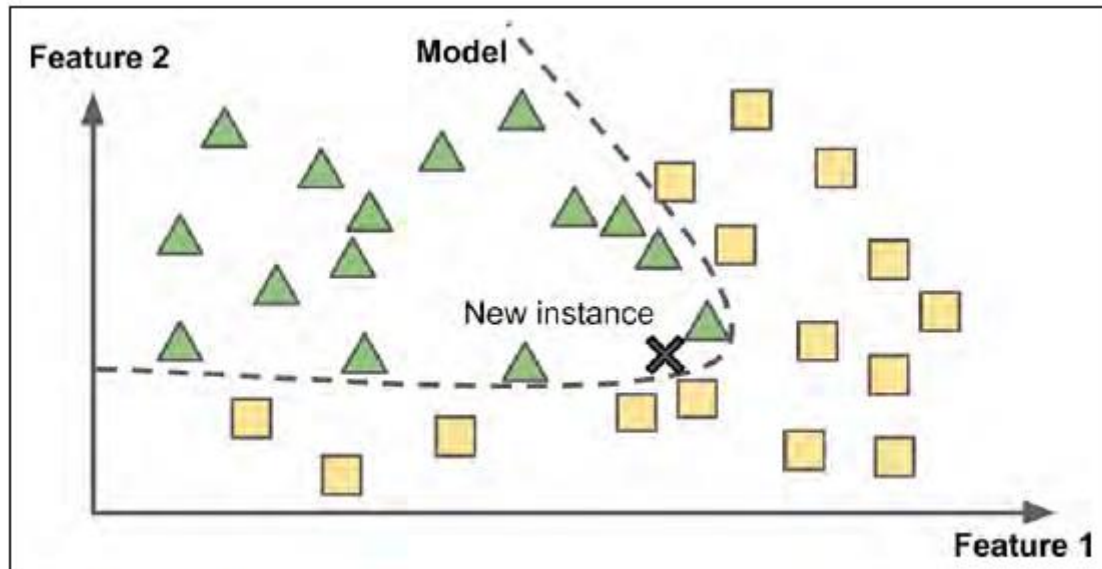
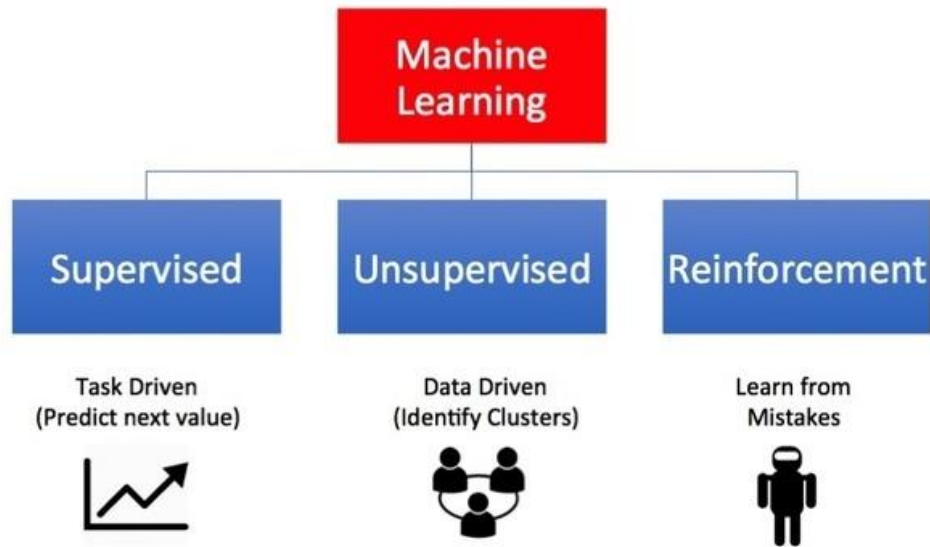


Figure 1-16. Model-based learning

Types of Machine Learning



Celem jest wykorzystanie zbioru danych do stworzenia modelu, który na podstawie **feature vector** dedukuje **label**.

W **supervised learning** zbiór danych to **labeled examples**:

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^N$$

Każdy element \mathbf{x}_i jest nazywany **feature vector**,

jest to wektor, którego każdy wymiar $j = 1, \dots, D$

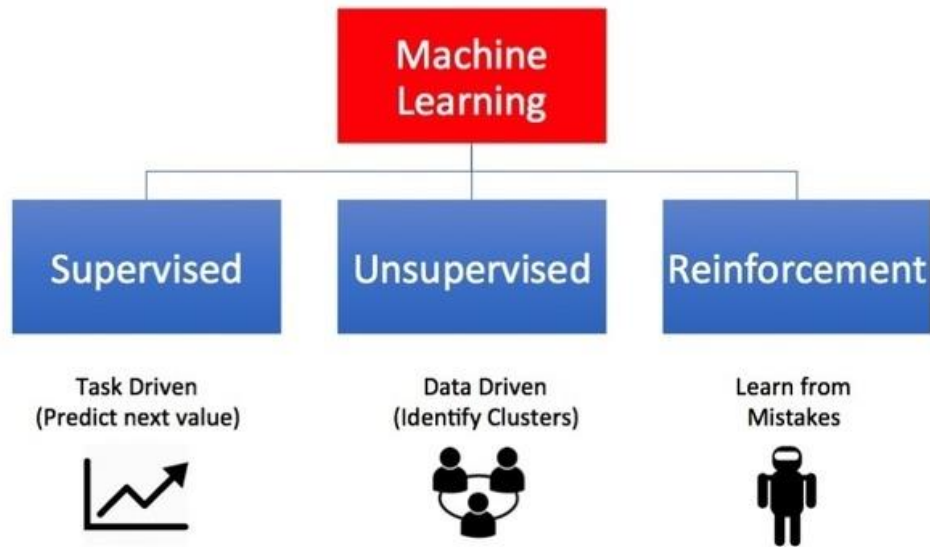
zawiera value opisujące example nazywane **feature**

i jest oznaczone jako $x^{(j)}$

Label y_i należy albo do skończonego zbioru klas $\{1, 2, \dots, C\}$

albo jest liczbą rzeczywistą (lub bardziej złożoną strukturą)

Types of Machine Learning



Celem jest wykorzystanie zbioru danych do stworzenia modelu, który na podstawie **feature vector** tworzy numer klastra (**clustering**), nowy wektor (**dimensionality reduction**) lub liczbę rzeczywistą opisującą w jakim stopniu wektor różni się od typowego wektora (**outlier detection**).

W **unsupervised learning** zbiór danych to **unlabeled examples**:

$$\{\mathbf{x}_i\}_{i=1}^N$$

Każdy element \mathbf{x}_i jest nazywany **feature vector**,

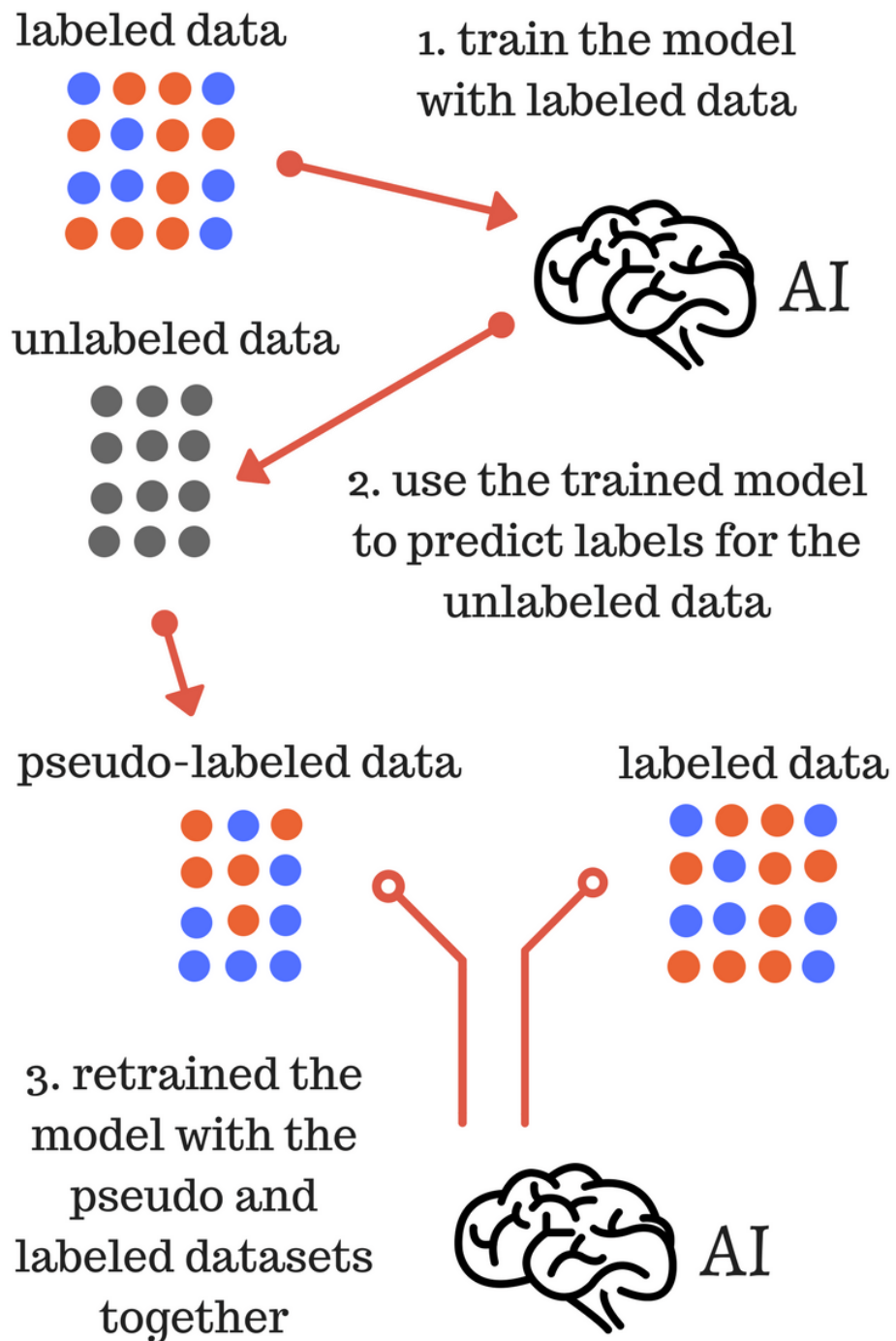
jest to wektor, którego każdy wymiar $j = 1, \dots, D$

zawiera value opisujące example nazywane **feature**

i jest oznaczone jako $x^{(j)}$

Label y_i należy albo do skończonego zbioru klas $\{1, 2, \dots, C\}$

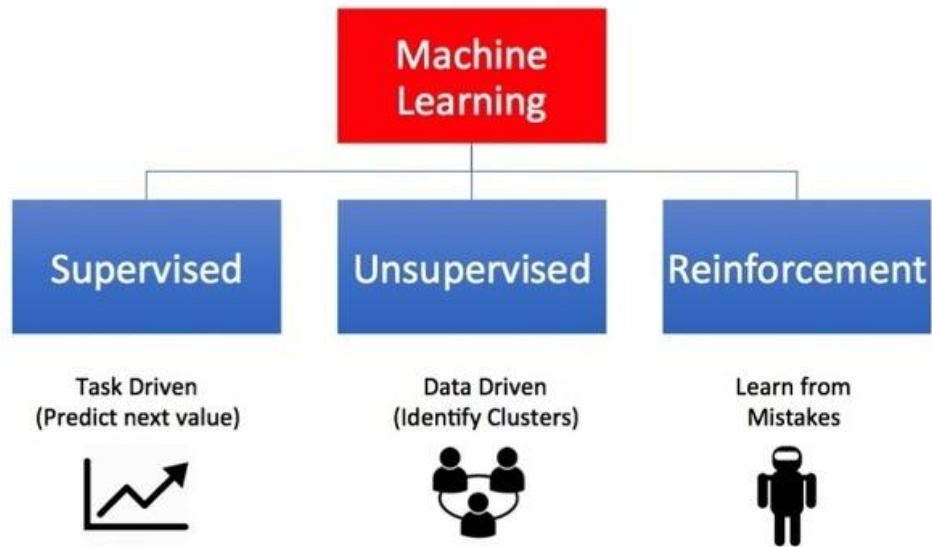
albo jest liczbą rzeczywistą (lub bardziej złożoną strukturą)



W Semi-Supervised learning zbiór danych zawiera examples z labels i bez labels. Zwykle liczba danych bez labels jest dużo większa niż examples z labels.

Cel jest taki sam jak w metodzie supervised.

Types of Machine Learning



Celem jest **learn a policy** – funkcji, która pobiera feature vector stanu i zwraca optymalną akcję (o maksymalnej **expected average reward**)

W **reinforcement learning** maszyna „żyje” w środowisku i jest w stanie pozyskiwać stan środowiska jako feature vector. Maszyna może podejmować akcję w każdym stanie środowiska co skutkuje różną skalą nagród i kar oraz może powodować zmianę środowiska.

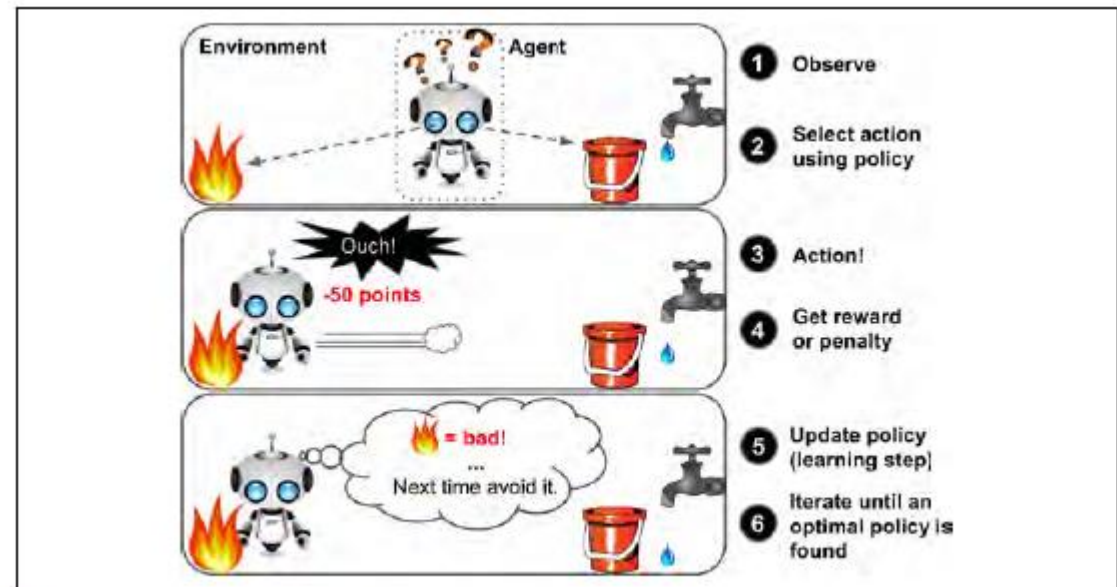


Figure 1-12. Reinforcement Learning

ALGORYTMY ML

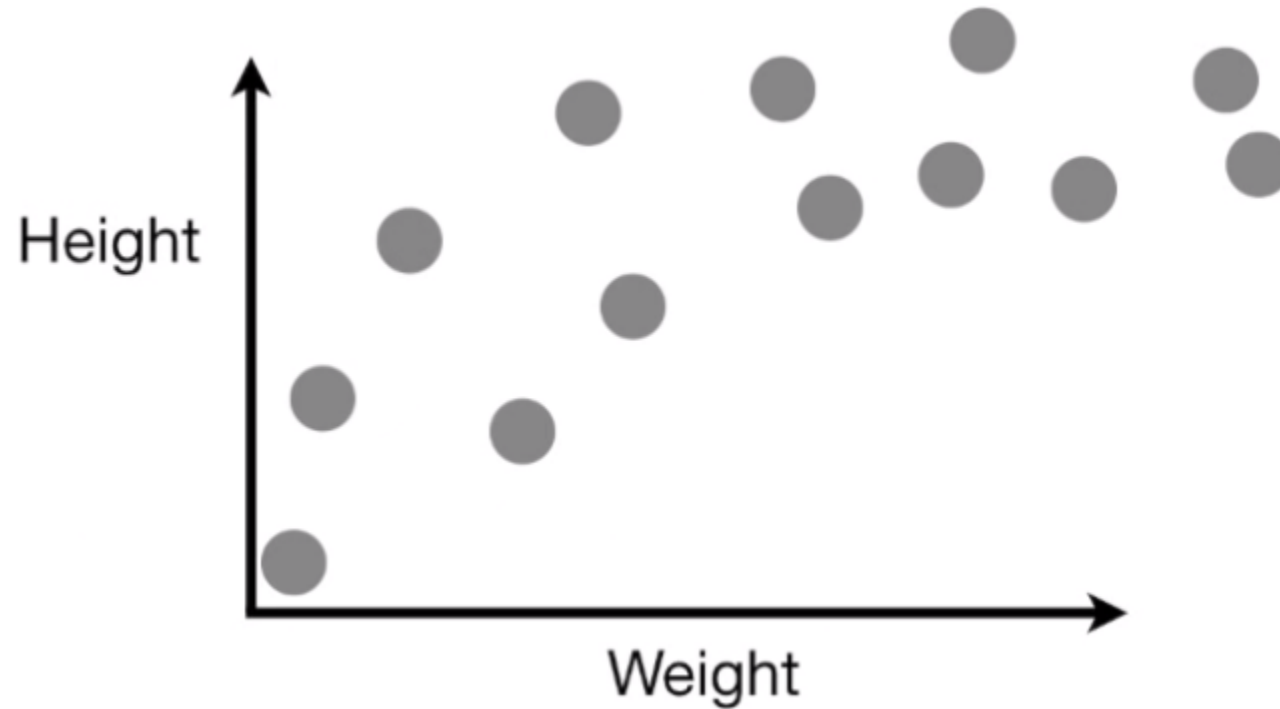
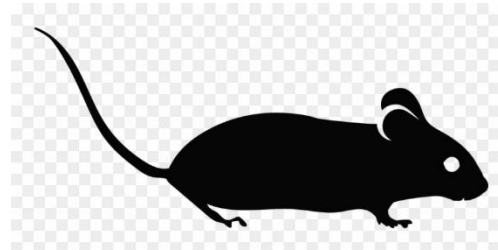
Najważniejsze supervised learning algorithms

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks₂

Najważniejsze unsupervised learning algorithms

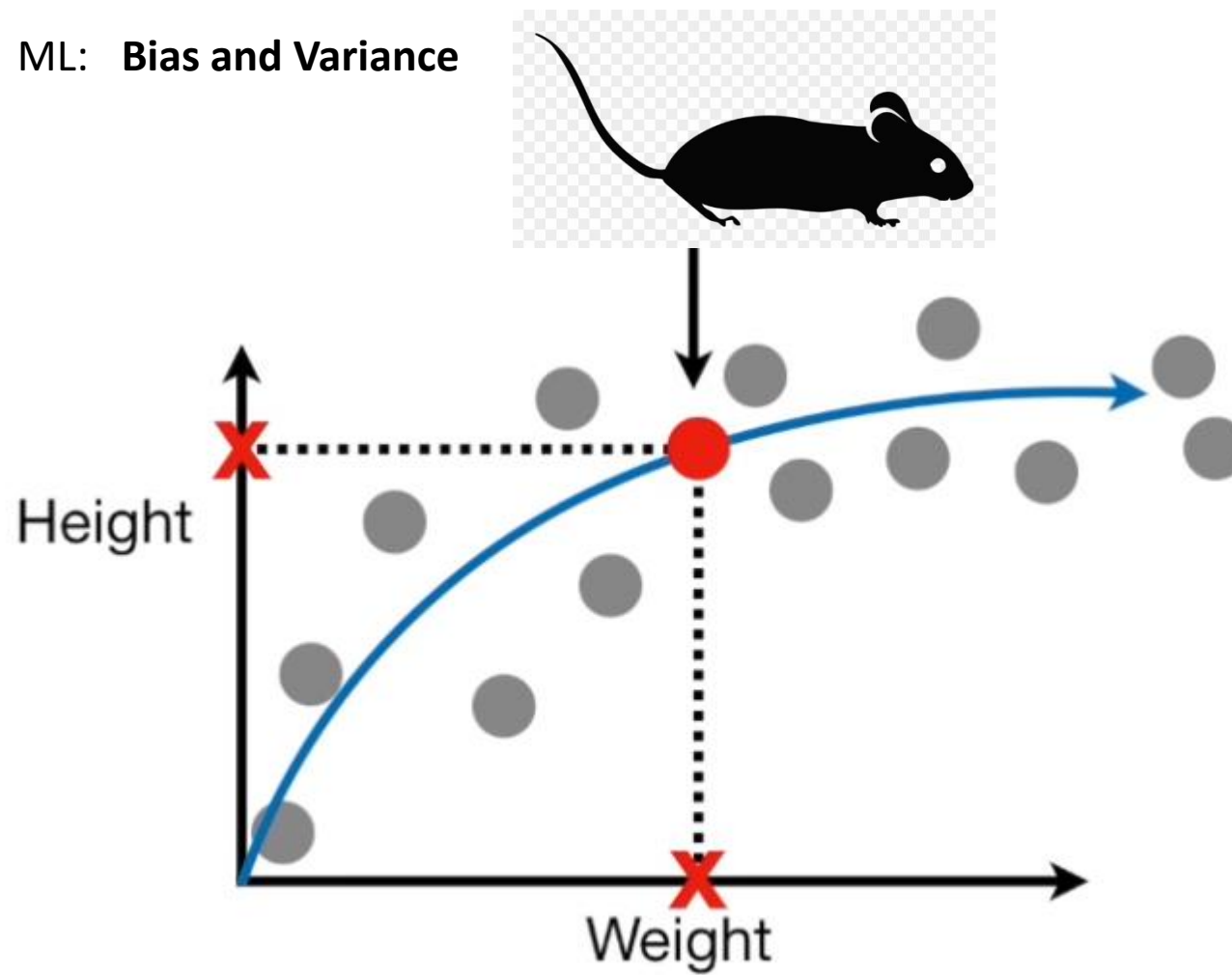
- **Clustering**
 - k-Means
 - Hierarchical Cluster Analysis (HCA)
 - Expectation Maximization
- Visualization and **dimensionality reduction**
 - Principal Component Analysis (PCA)
 - Kernel PCA
 - Locally-Linear Embedding (LLE)
 - t-distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning
 - Apriori
 - Eclat

PODSTAWOWE POJĘCIA: **Bias and Variance**



Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

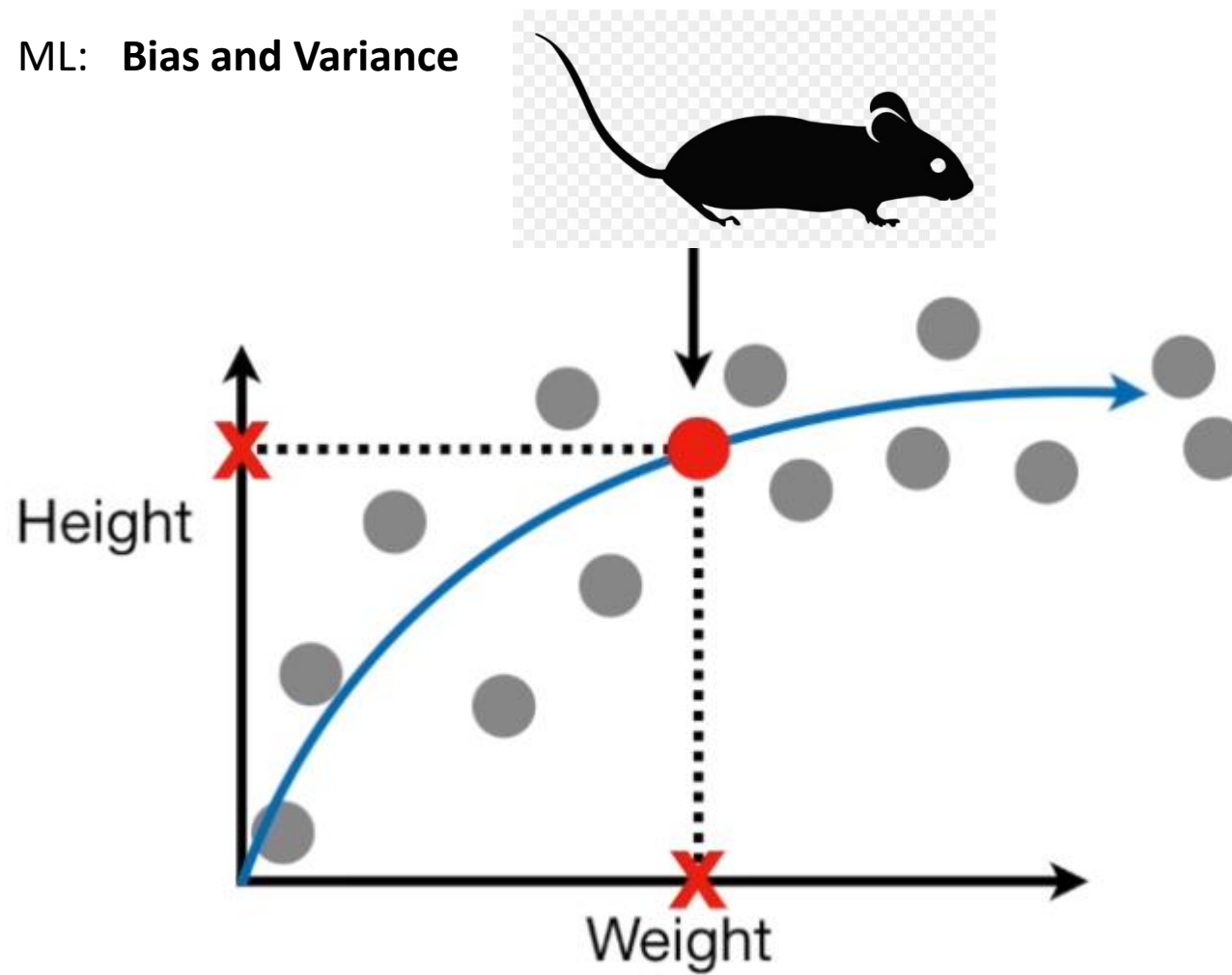
PODSTAWOWE POJĘCIA ML: **Bias and Variance**



Sytuacja idealna:
znane jest wyrażenie

Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

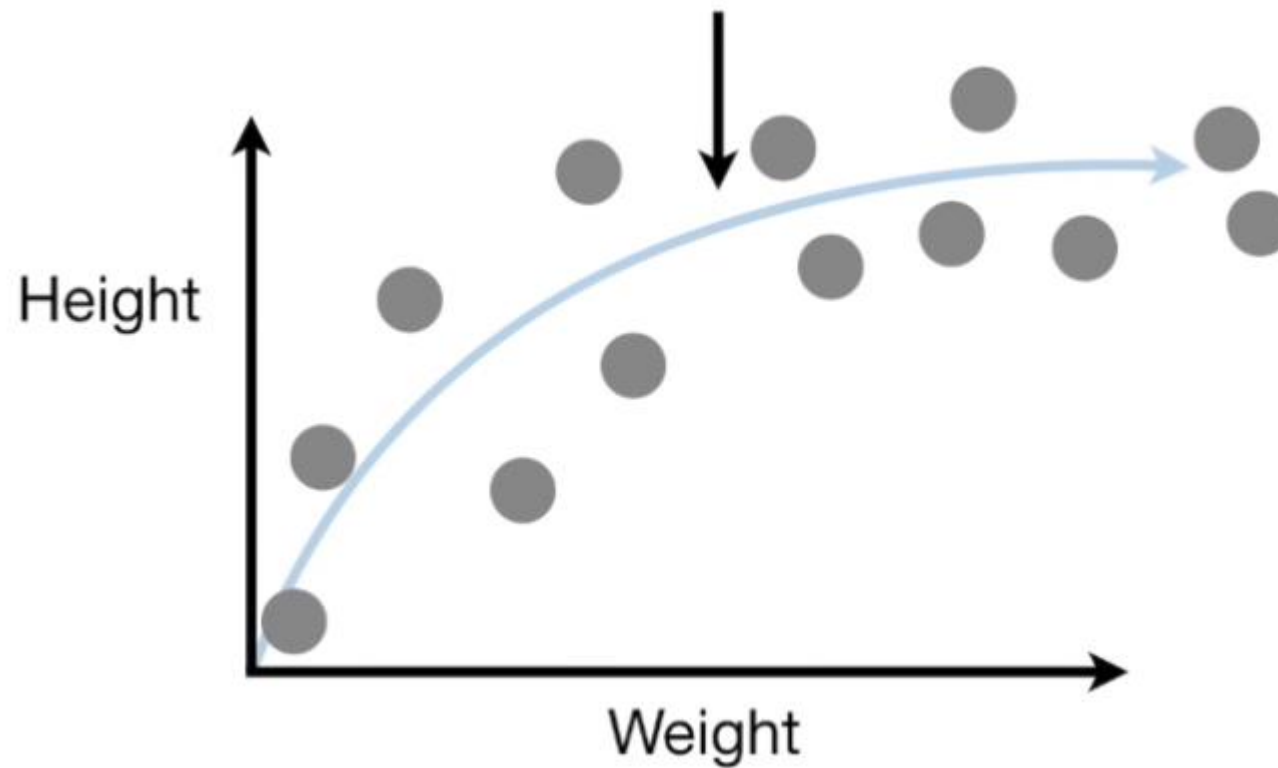
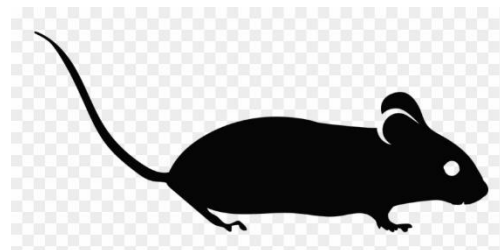
PODSTAWOWE POJĘCIA ML: **Bias and Variance**



~~Syst. nie idzie do przodu:
zmniejszenie błędów jest niemożliwe~~

Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

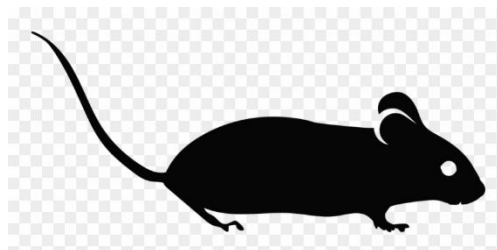
PODSTAWOWE POJĘCIA ML: **Bias and Variance**



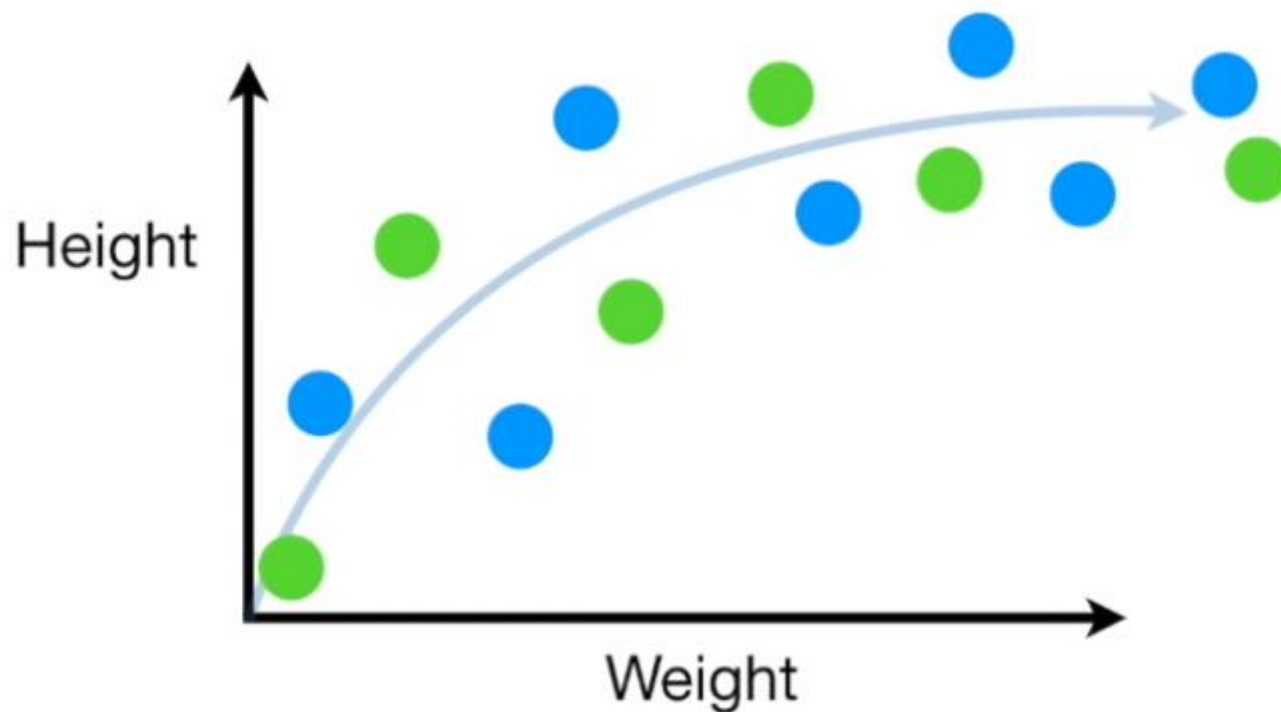
Wykorzystanie ML do
modelowania tej
zależności

Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

PODSTAWOWE POJĘCIA ML: **Bias and Variance**



Wykorzystanie ML do
modelowania tej
zależności



Krok 1.

Dzielimy dane na dwie grupy:

training set

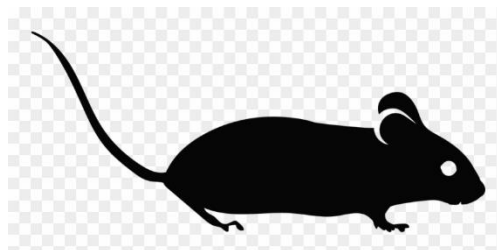


Testing set

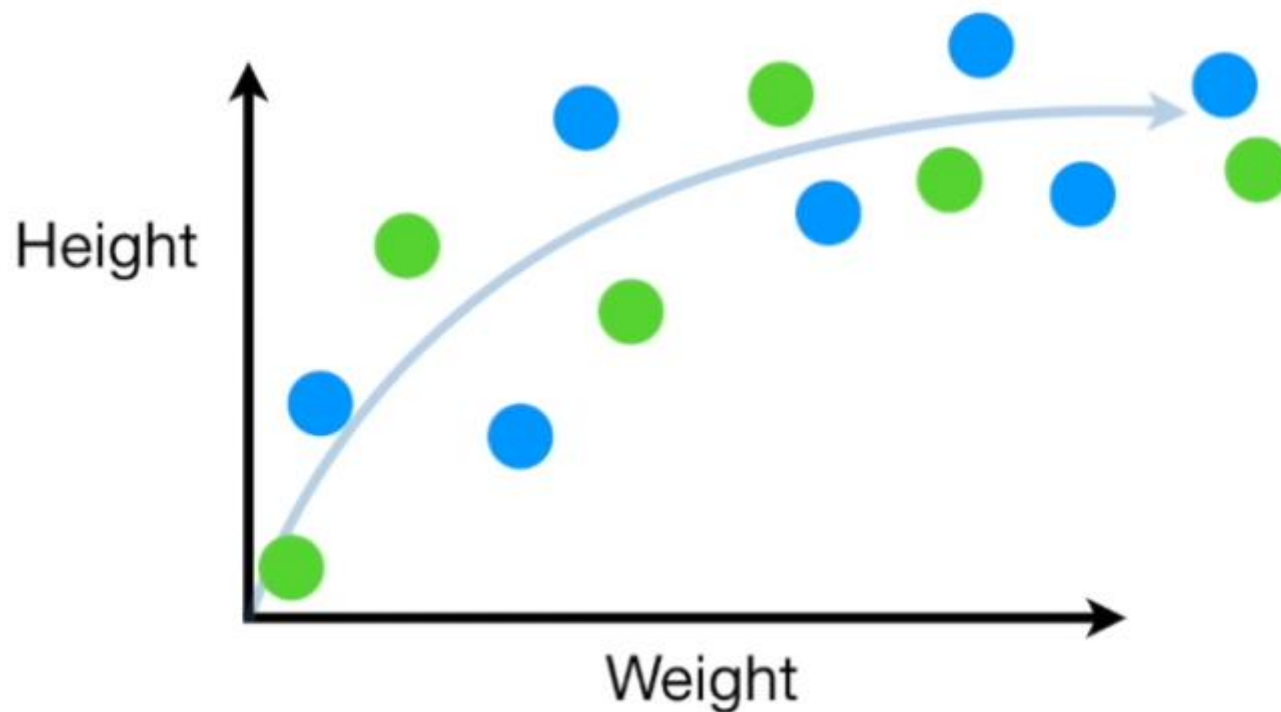


Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

PODSTAWOWE POJĘCIA ML: **Bias and Variance**



Wykorzystanie ML do modelowania tej zależności



Krok 1.

Dzielimy dane na dwie grupy:

training set

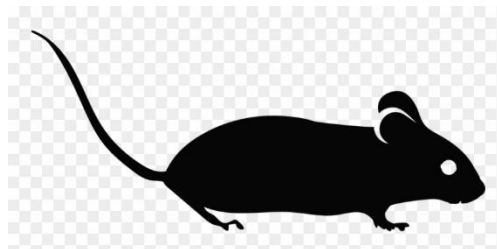


testing set



Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

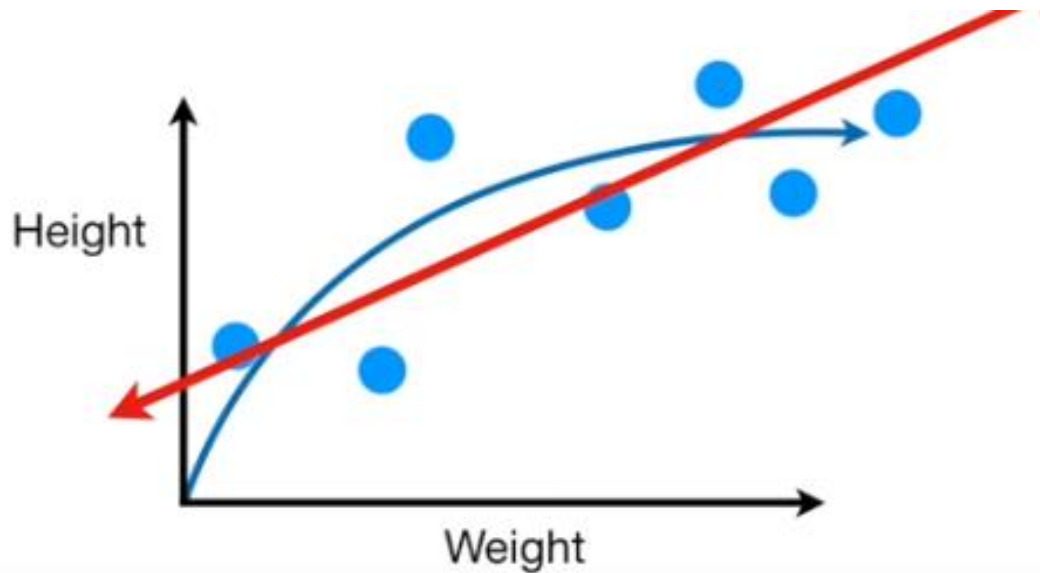
PODSTAWOWE POJĘCIA ML: **Bias and Variance**



Wykorzystanie ML do modelowania tej zależności

Pierwszy ML algorytm

Metoda przypisuje prostą do danych



**Ze względu na brak
możliwego zakrzywienia
linii bias jest duży**

Brak możliwości danej metody ML odzwierciedlenia prawdziwej relacji nazywa się **bias**.

Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

Krok 1.

Dzielimy dane na dwie grupy:

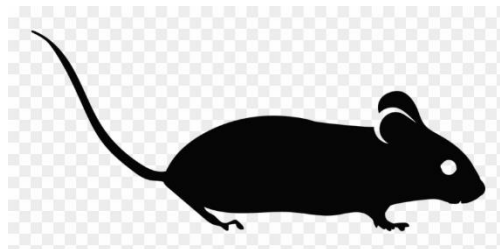
training set



testing set



PODSTAWOWE POJĘCIA ML:

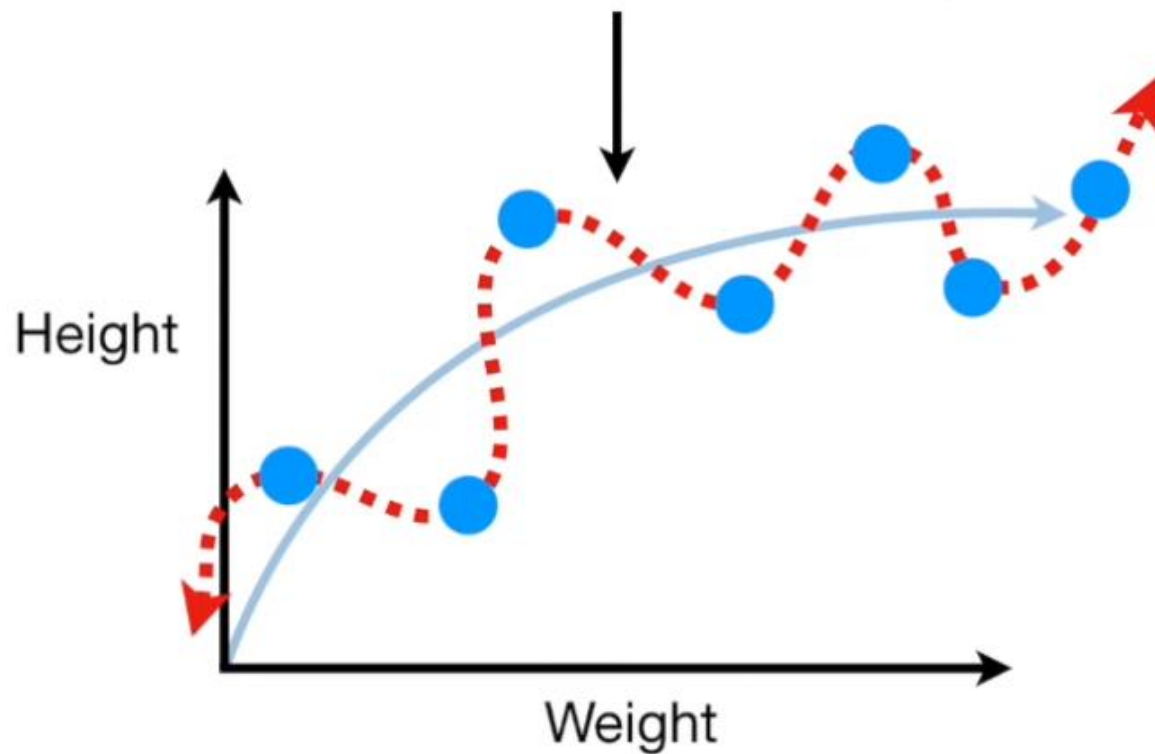


Wykorzystanie ML do modelowania tej zależności

ML algorytm

Inna metoda przypisuje krzywą do danych

Ze względu na możliwe zakrzywienia linii bias jest mały



Krok 1.

Dzielimy dane na dwie grupy:

training set

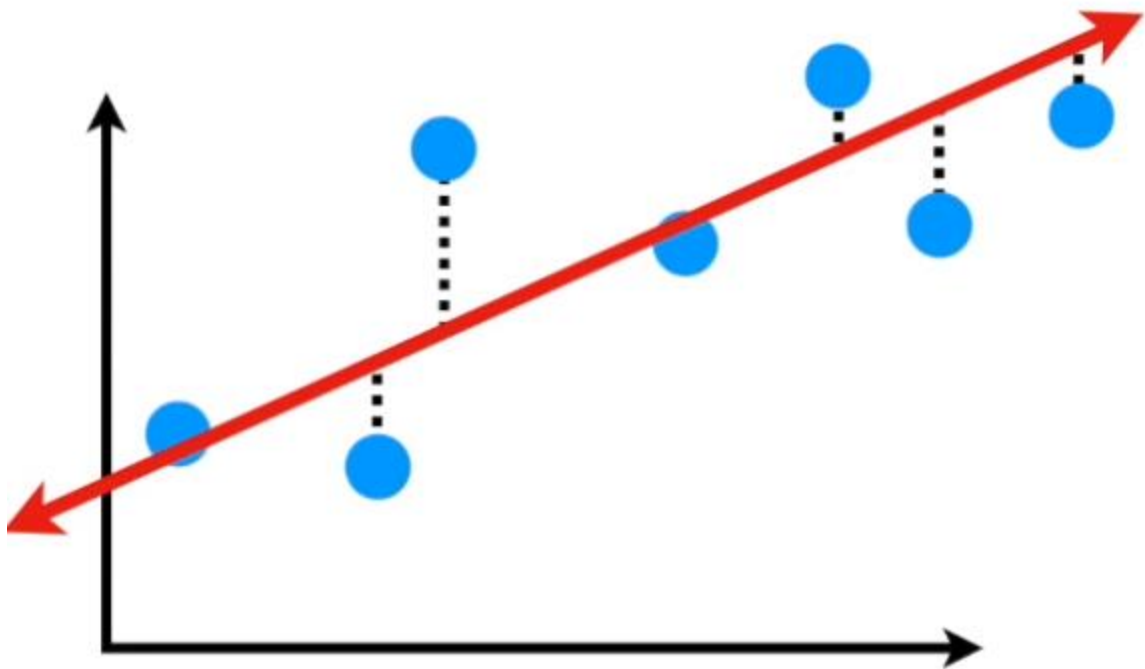


Testing set

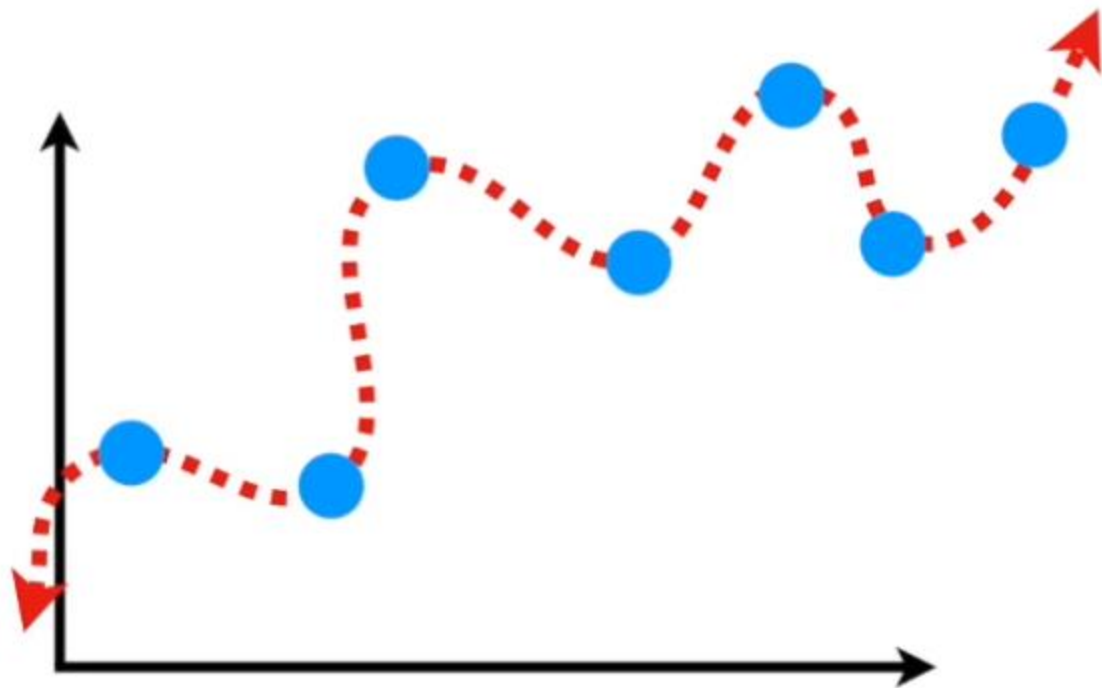


Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

TRAINING



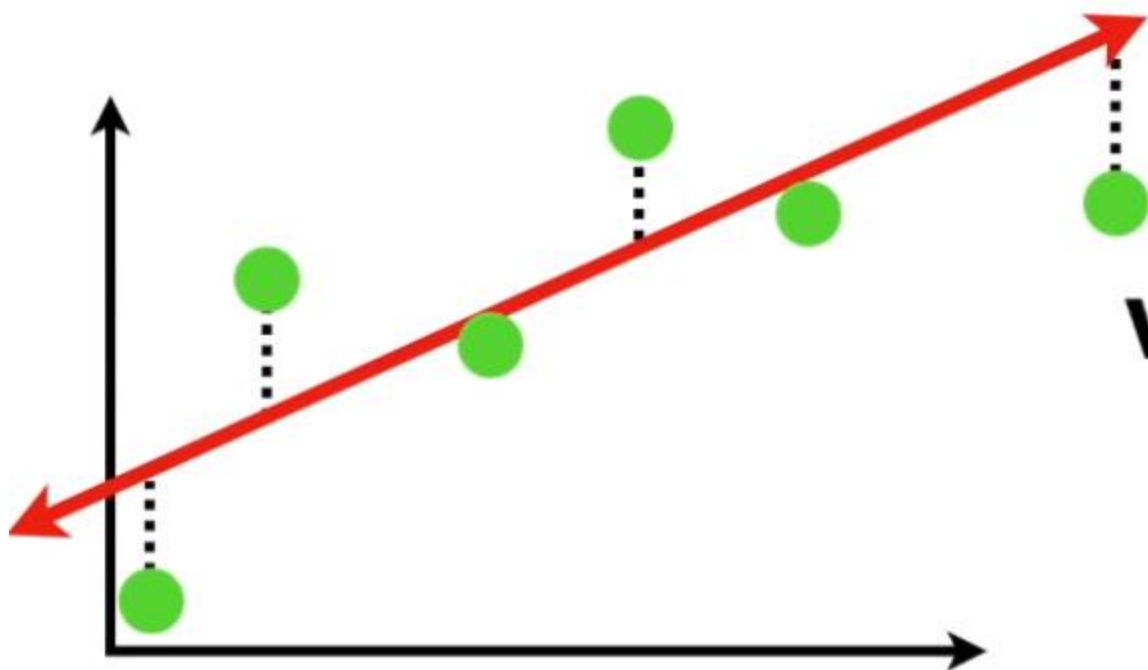
LEPIEJ



Miara suma kwadratów (im większa tym gorzej - **loss function**)

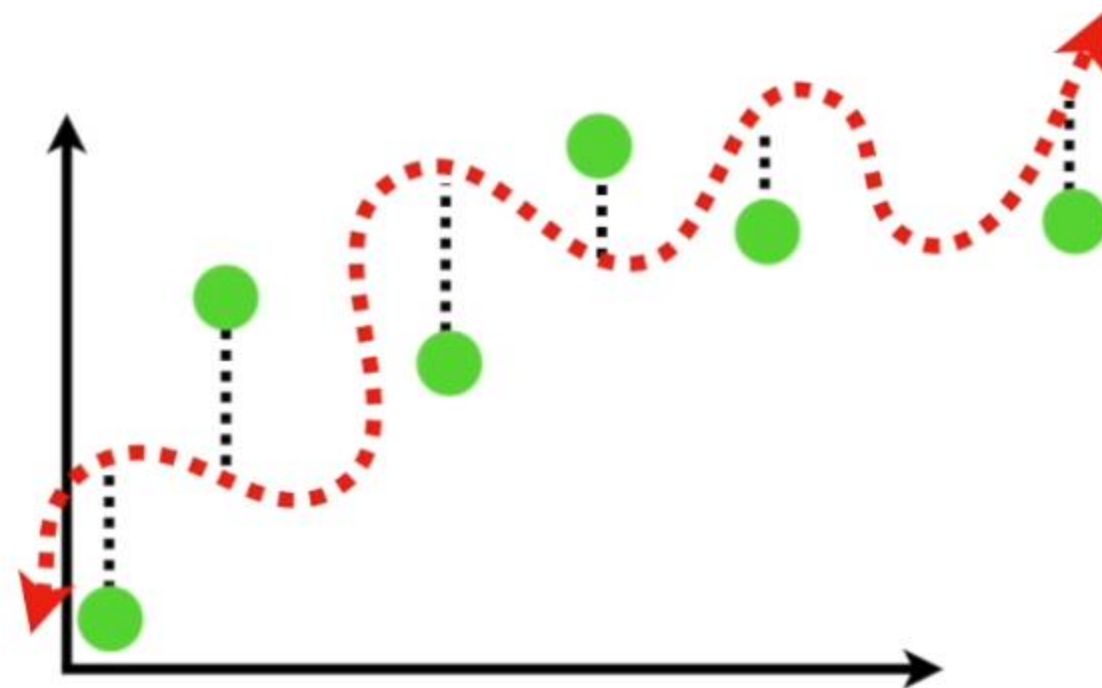
TESTING

LEPIEJ



VS.

RÓŻNICA w dopasowaniu do zbiorów
danych nazywa się variance, w tym
przypadku jest ona większa

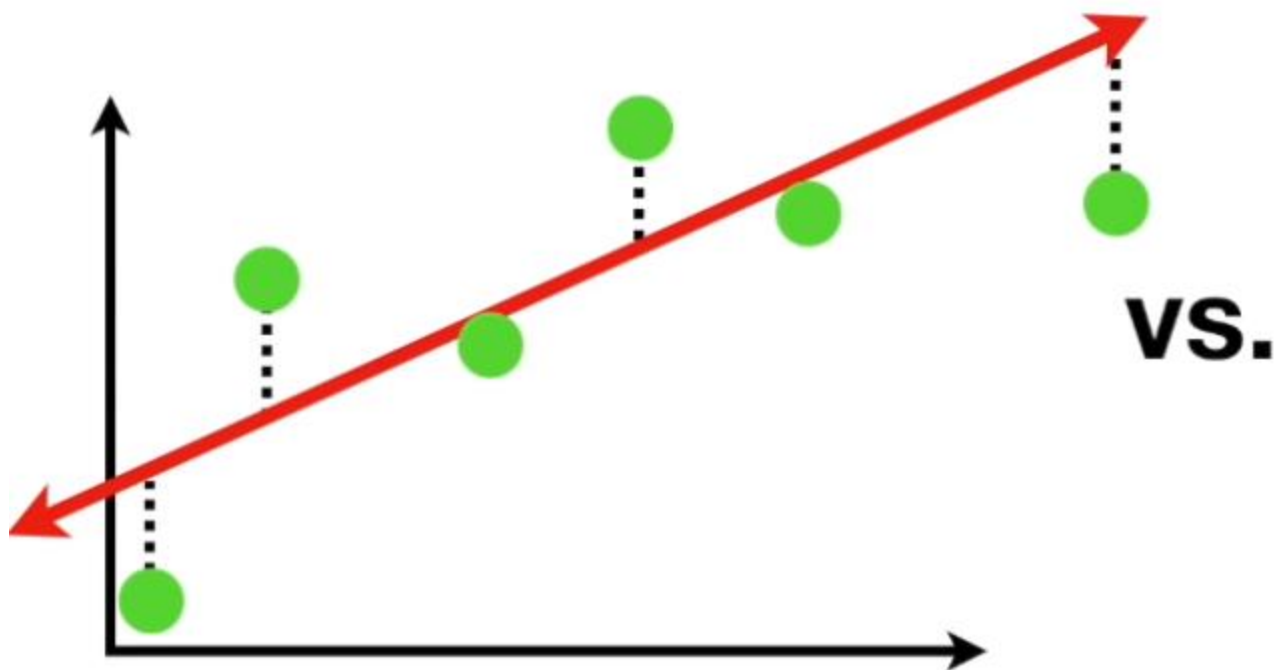


Miara suma kwadratów (im większa tym gorzej - **loss function**)

W większości algorytmów celem jest minimalizacja **loss function**

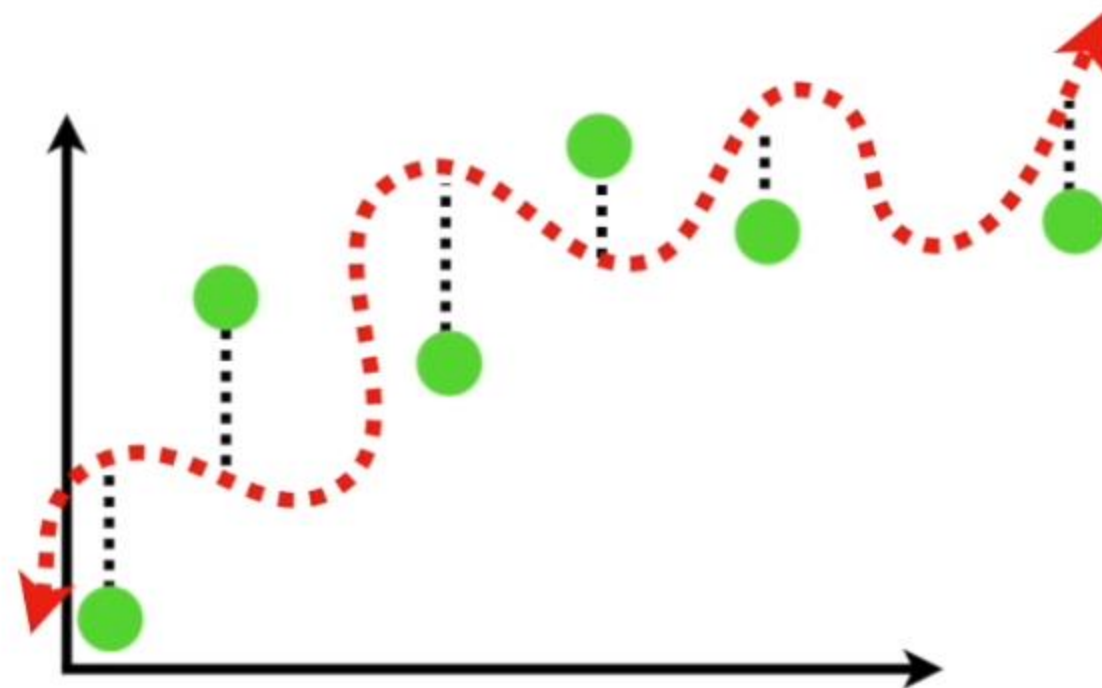
TESTING

LEPIEJ

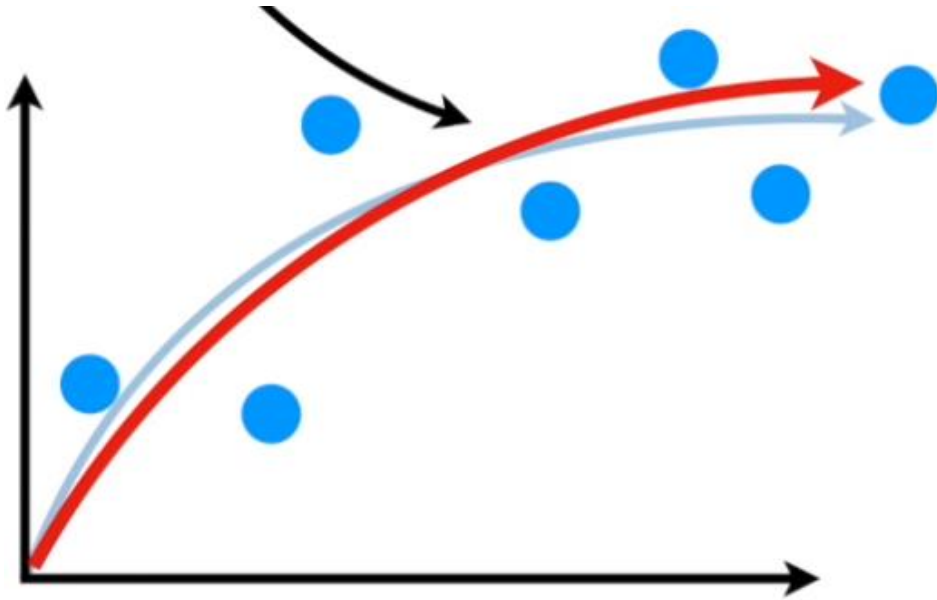


Miara suma kwadratów (im większa tym gorzej - **loss function**)

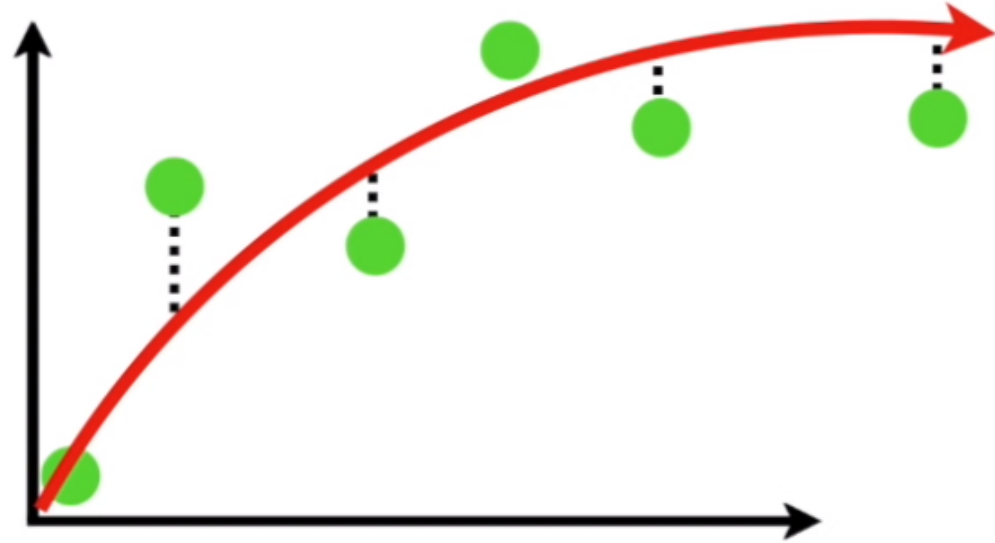
RÓŻNICA w dopasowaniu do zbiorów
danych nazywa się variance, w tym
przypadku jest ona większa



Ponieważ krzywa pasuje do zbioru treningowego
bardzo dobrze, ale nie pasuje do zbioru
testowego is **overfit**



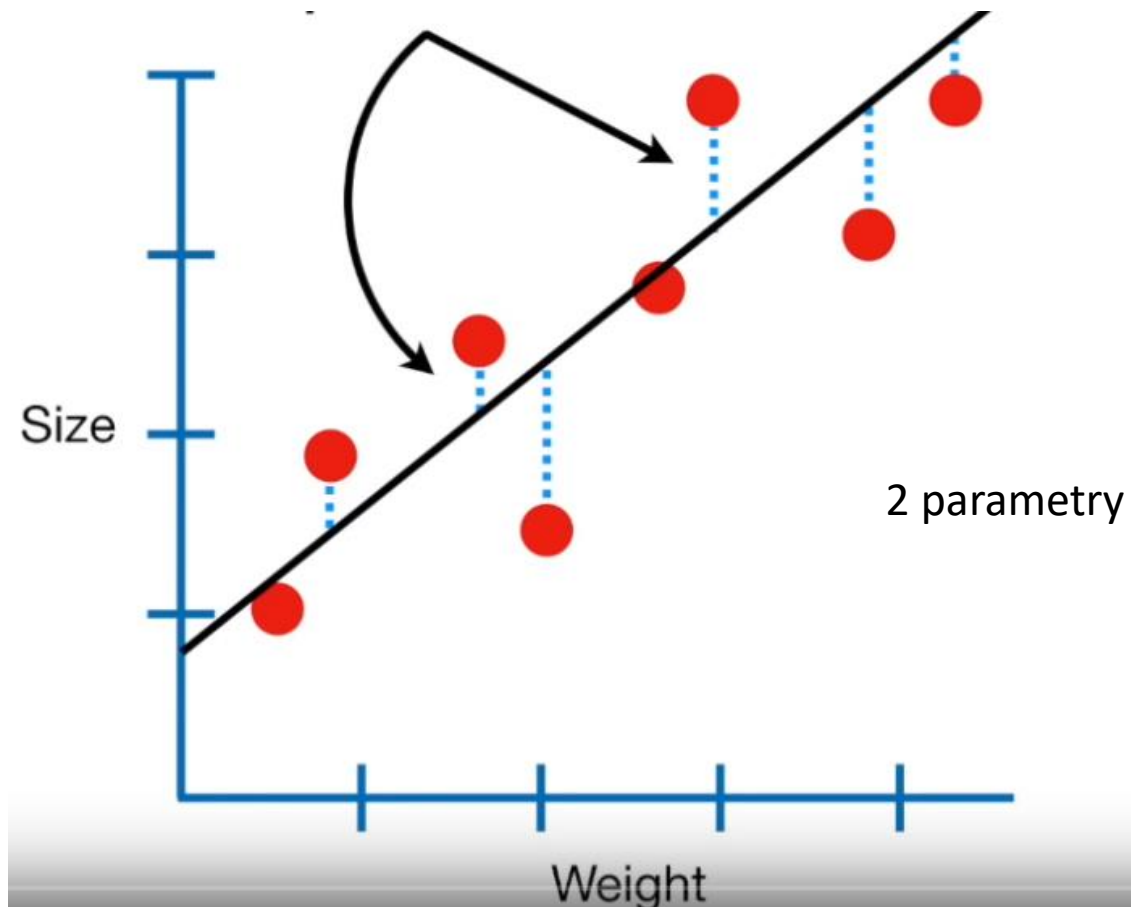
W ML idealny algorytm ma **low bias** i **low variability**



Podstawowe metody znajdowania złotego środka pomiędzy prostymi metodami a złożonymi to:

regularization
boosting
bagging

JAK działają te metody **Regularization > Ridge Regression**



Znajdujemy linię, która daje:

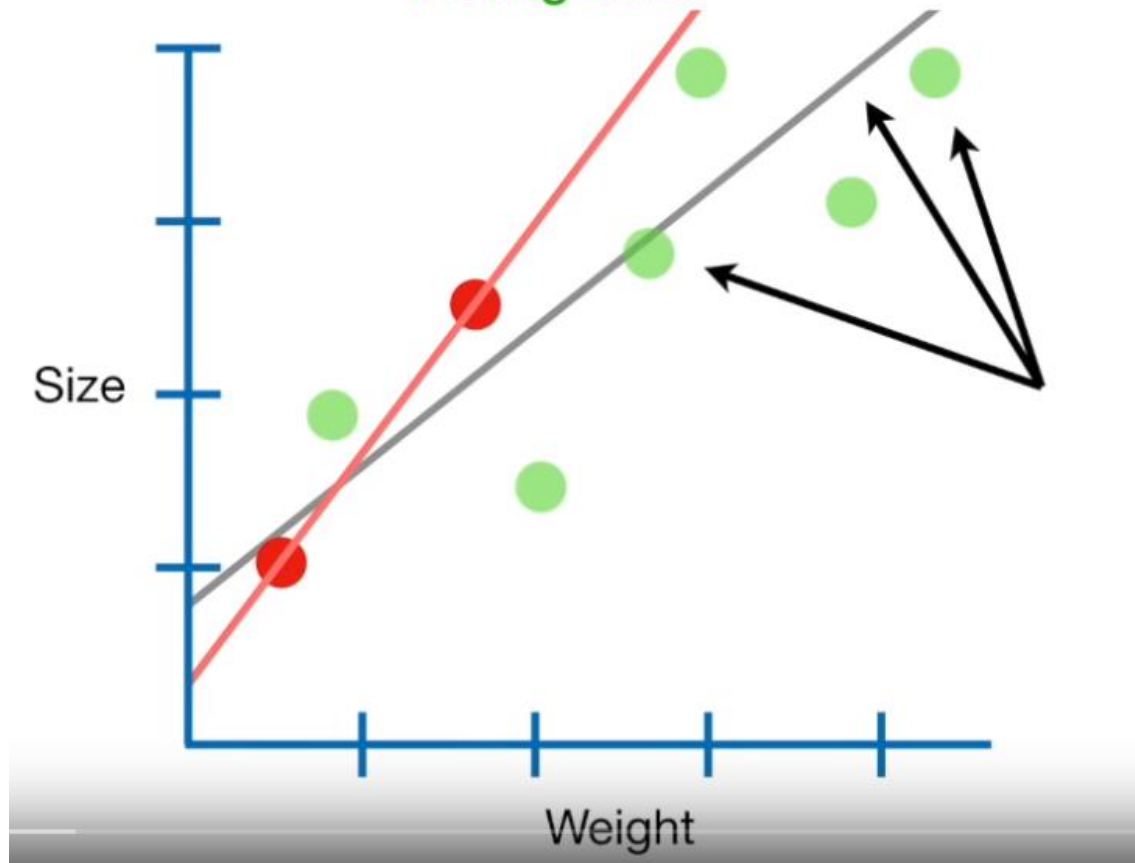
minimum sum of squared residuals

y-axis intercept.

$$\text{Size} = 0.9 + 0.75 \times \text{Weight}$$

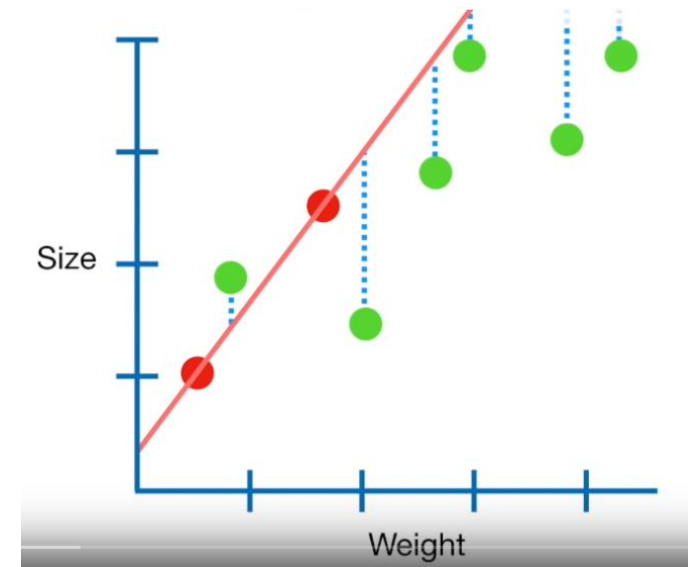
slope.

Let's call the **Two Red Dots** the **Training Data**, and the remaining **Green Dots** the **Testing Data**.



Jeśli mamy tylko dwa punkty:

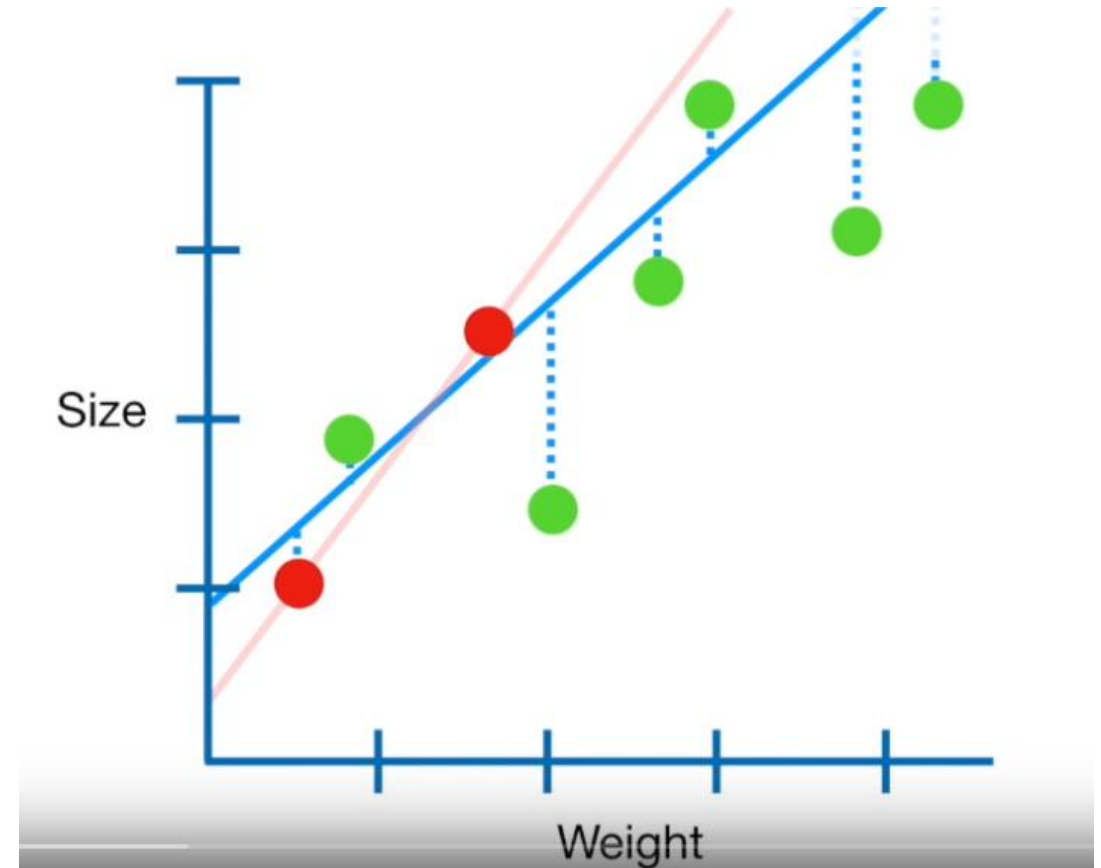
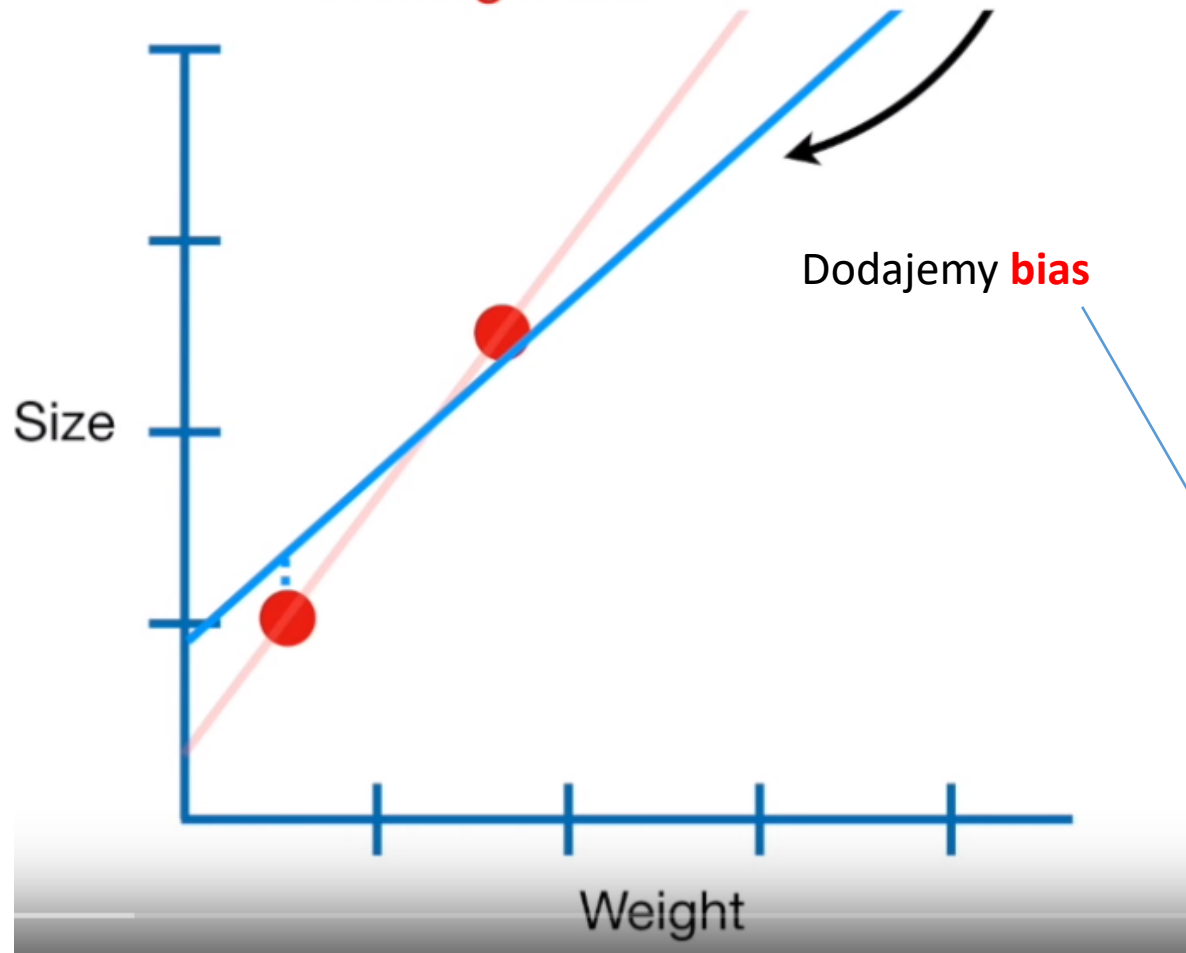
Sum for squared residuals for train = 0
Sum for square residuals for test LARGE



New line has High Variance

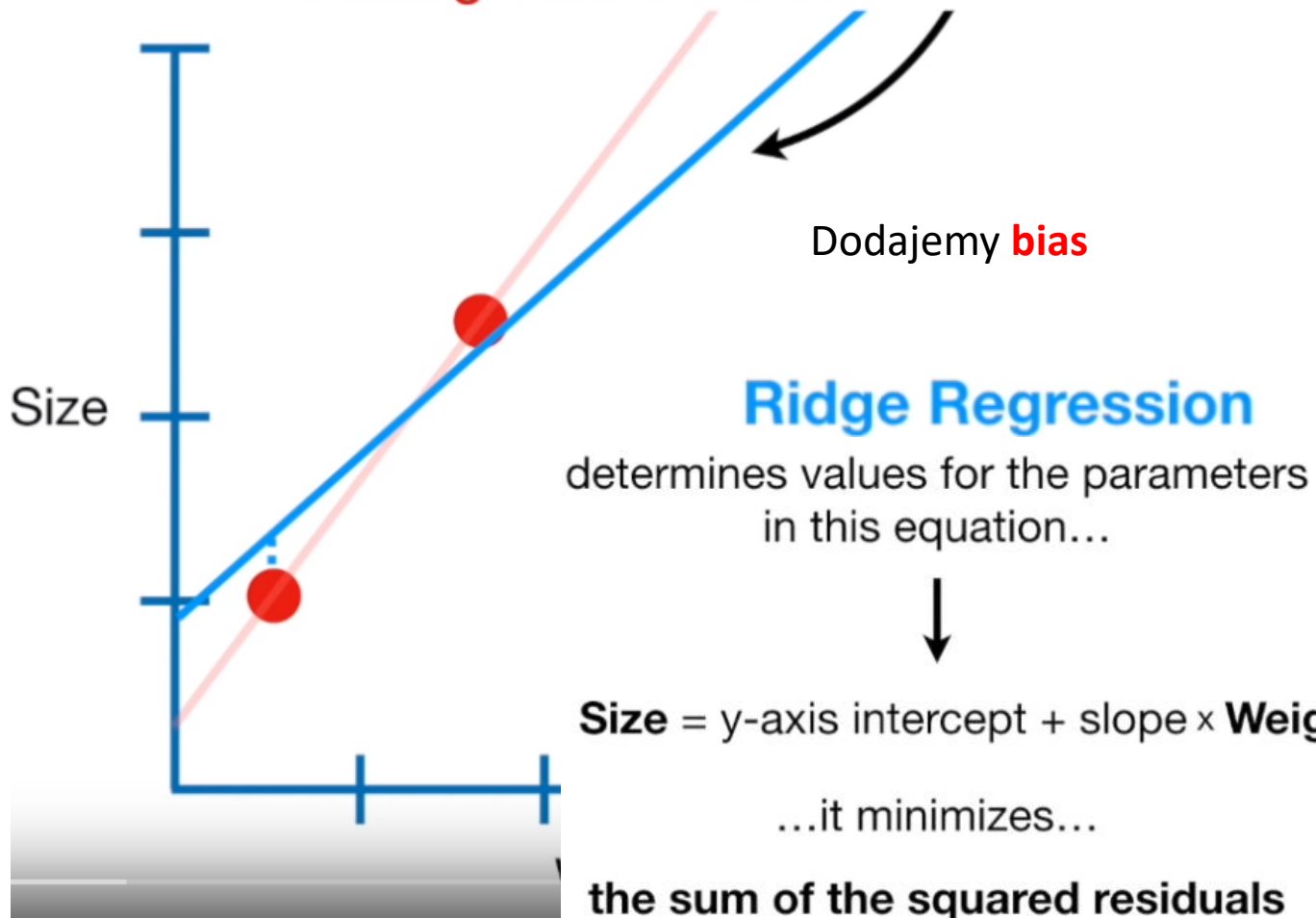
New Line is **Over Fit** to the **Training Data**.

The main idea behind **Ridge Regression** is to find a **New Line** that doesn't fit the **Training Data** as well...



Zmniejsza się dzięki temu **variance**

The main idea behind **Ridge Regression** is to find a **New Line** that doesn't fit the **Training Data** as well...



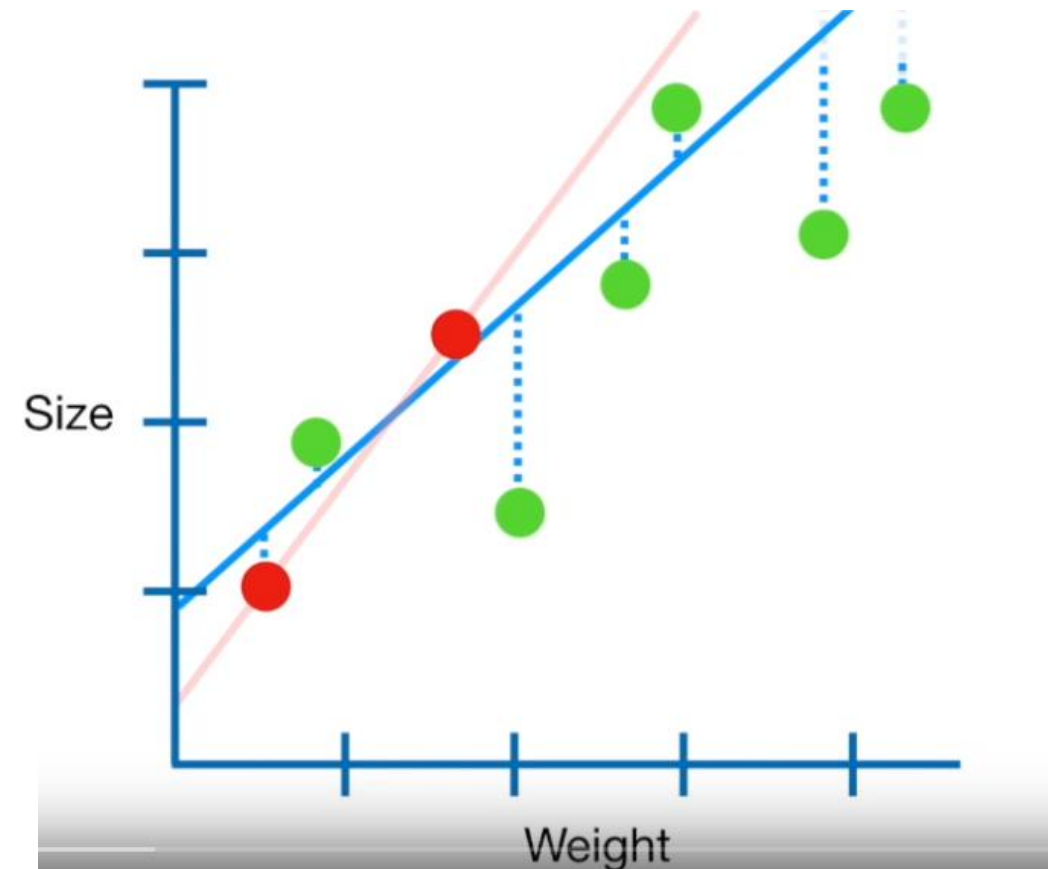
$$\text{Size} = \text{y-axis intercept} + \text{slope} \times \text{Weight}$$

...it minimizes...

the sum of the squared residuals

+

$\lambda \times \text{the slope}^2$



Zmniejsza się dzięki temu **variance**

Ridge Regression

determines values for the parameters
in this equation...



Size = y-axis intercept + slope x **Weight**

...it minimizes...

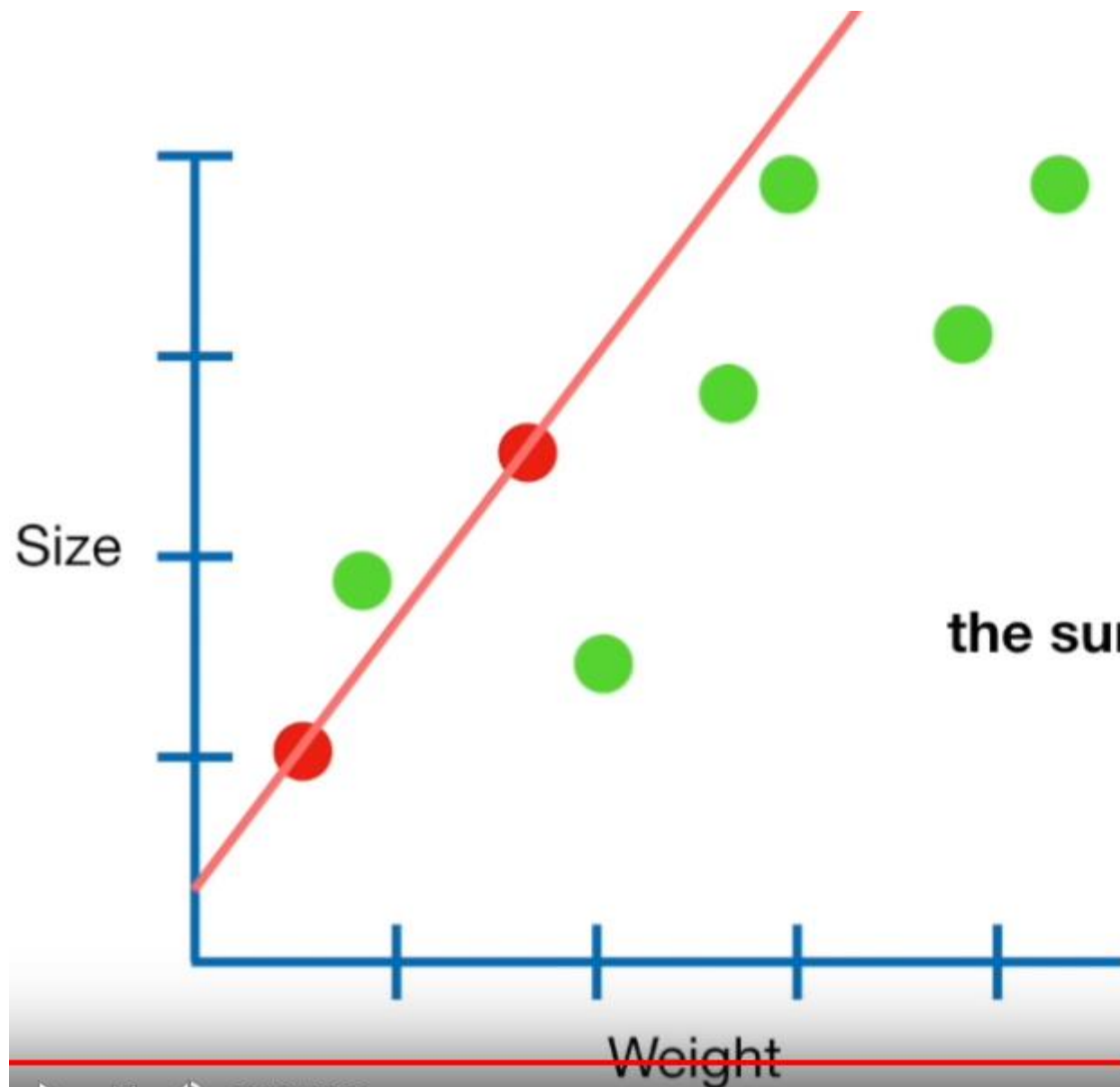
the sum of the squared residuals

+

$\lambda \times \text{the slope}^2$



dodaje penalty do tradycyjnej Least Squares method

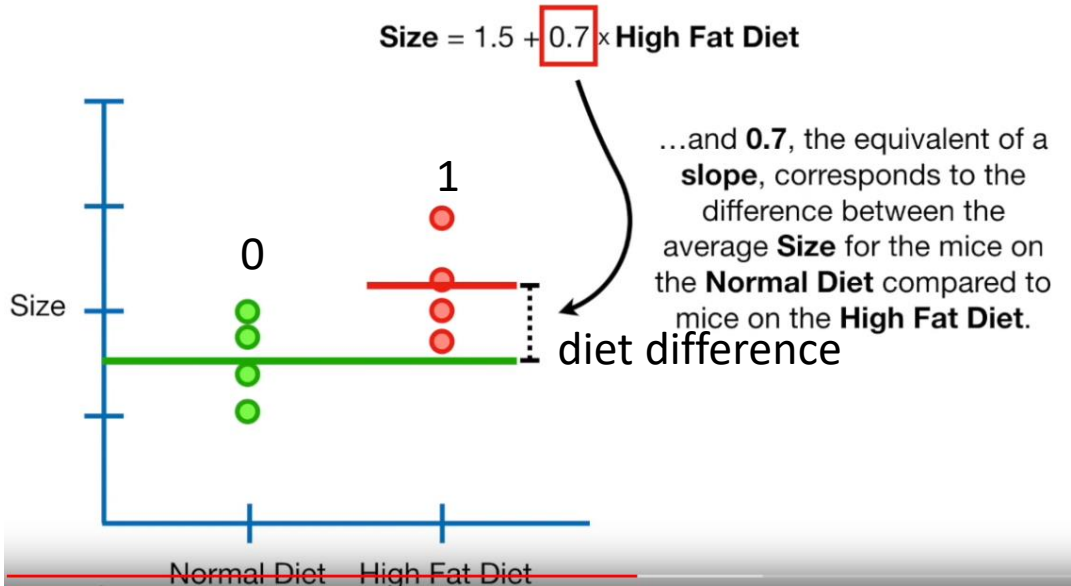
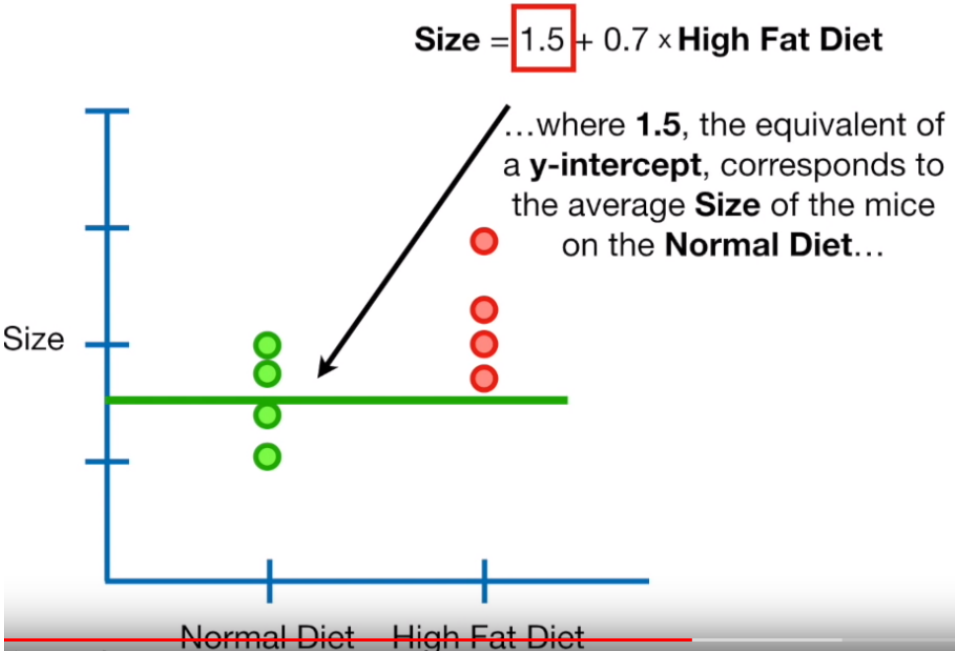


We just try a bunch of values for λ and use **Cross Validation**, typically **10-fold Cross Validation**, to determine which one results in the lowest **Variance**.

the sum of the squared residuals
+
 $\lambda \times \text{the slope}^2$

Weight

Dla danych dyskretnych:



Algorytmy ML składają się z trzech części:

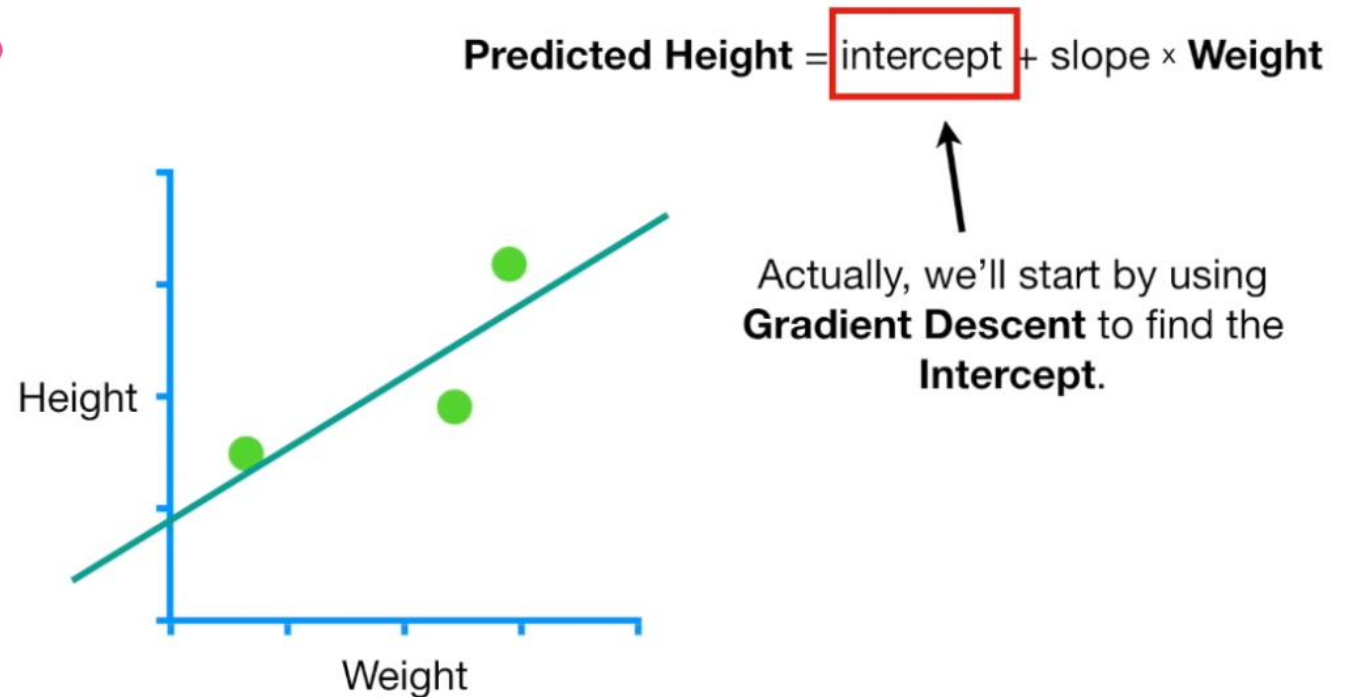
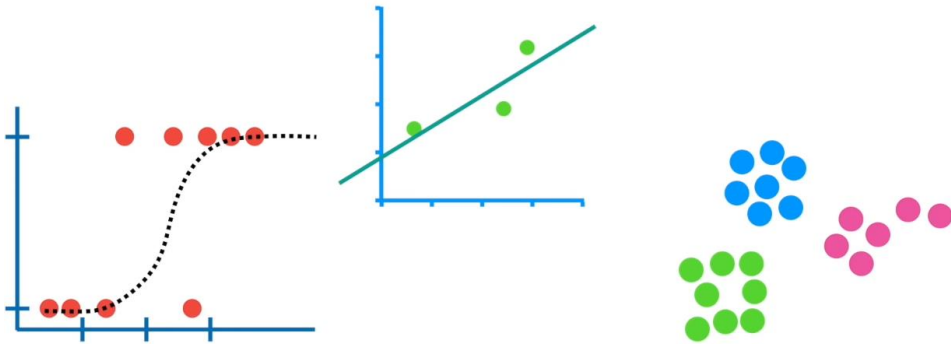
1. Loss function
2. Kryterium optymalizacji bazującym na loss function
3. Metodzie optymalizacji, która na podstawie danych treningowych znajduje rozwiązanie kryterium optymalizacji

Niektóre algorytmy ML (najstarsze) nie zawierają globalnej automatyzacji

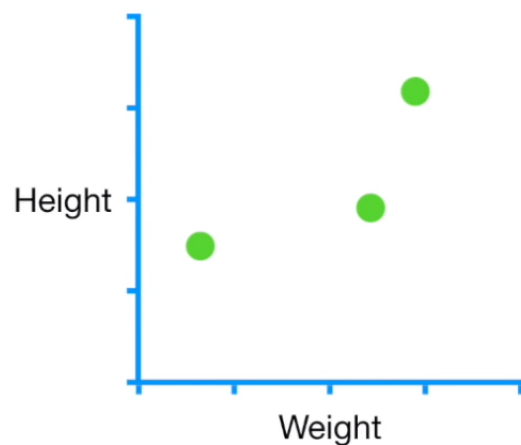
Najczęściej używanymi algorytmami optymalizacji są:
gradient descent i **stochastic gradient descent**

Gradient Descent – optymalizacja parametrow

<https://www.youtube.com/watch?v=sDv4f4s2SB8>



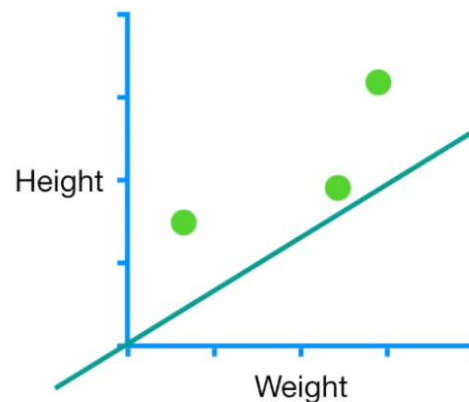
$$\text{Predicted Height} = \text{intercept} + 0.64 \times \text{Weight}$$



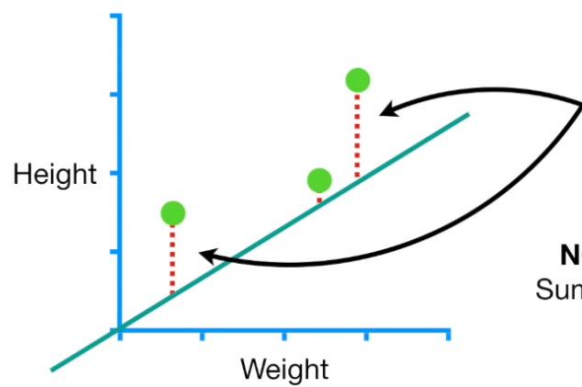
The first thing we do is pick a random value for the **Intercept**.

This is just an initial guess that gives **Gradient Descent** something to improve upon.

$$\text{Predicted Height} = 0 + 0.64 \times \text{Weight}$$



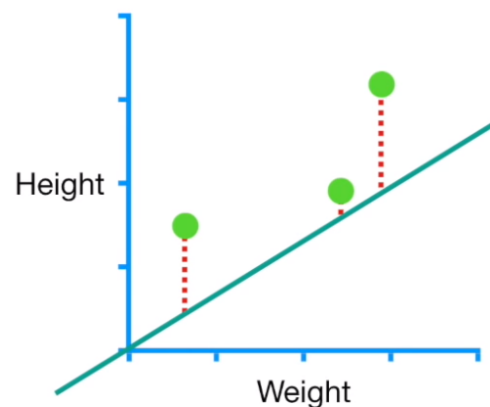
And that gives us the equation for this line.



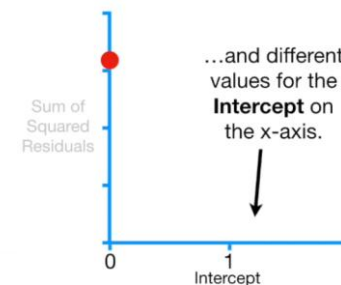
In this example, we will evaluate how well this line fits the data with the **Sum of the Squared Residuals**.

NOTE: In Machine Learning lingo, The Sum of the Squared Residuals is a type of **Loss Function**.

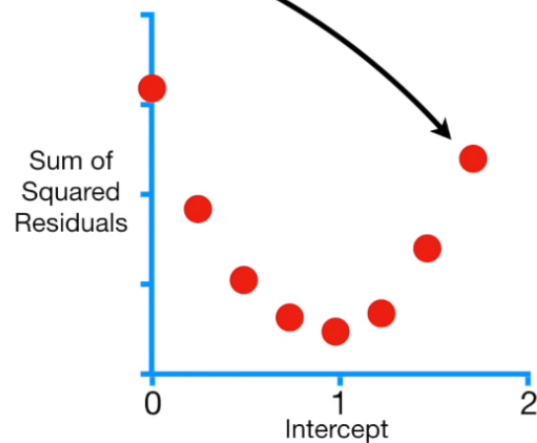
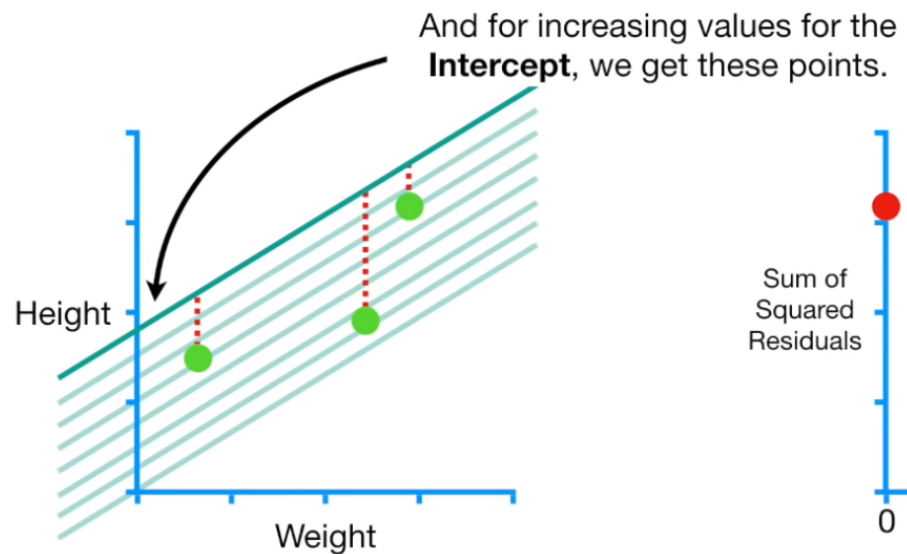
$$\text{Sum of squared residuals} = 1.1^2 + 0.4^2 + 1.3^2 = 3.1$$



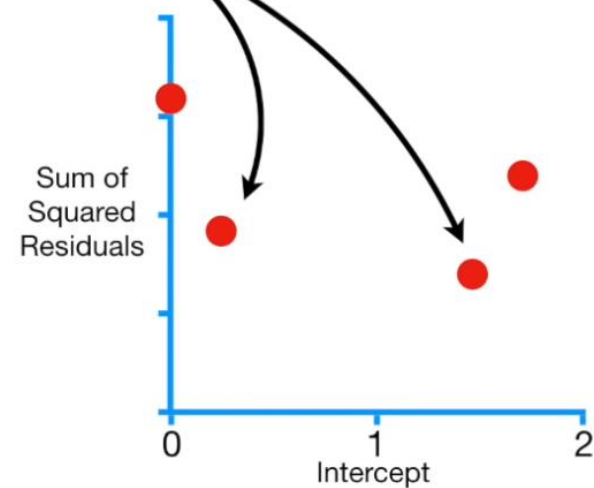
In the end, **3.1** is the Sum of the Squared Residuals.



...and different values for the **Intercept** on the x-axis.

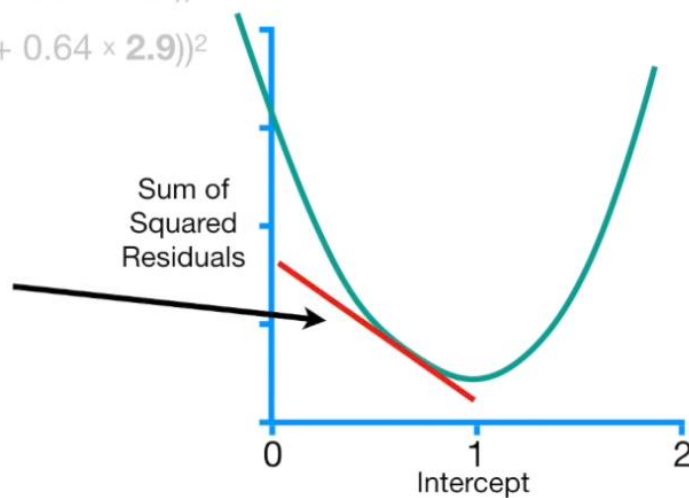


Gradient Descent only does a few calculations far from the optimal solution...



$$\begin{aligned} \text{Sum of squared residuals} &= (1.4 - (\text{intercept} + 0.64 \times 0.5))^2 \\ &+ (1.9 - (\text{intercept} + 0.64 \times 2.3))^2 \\ &+ (3.2 - (\text{intercept} + 0.64 \times 2.9))^2 \end{aligned}$$

...and we can take the derivative of this function and determine the slope at any value for the **Intercept**.



$$\frac{d}{d \text{ intercept}} \text{ Sum of squared residuals} =$$

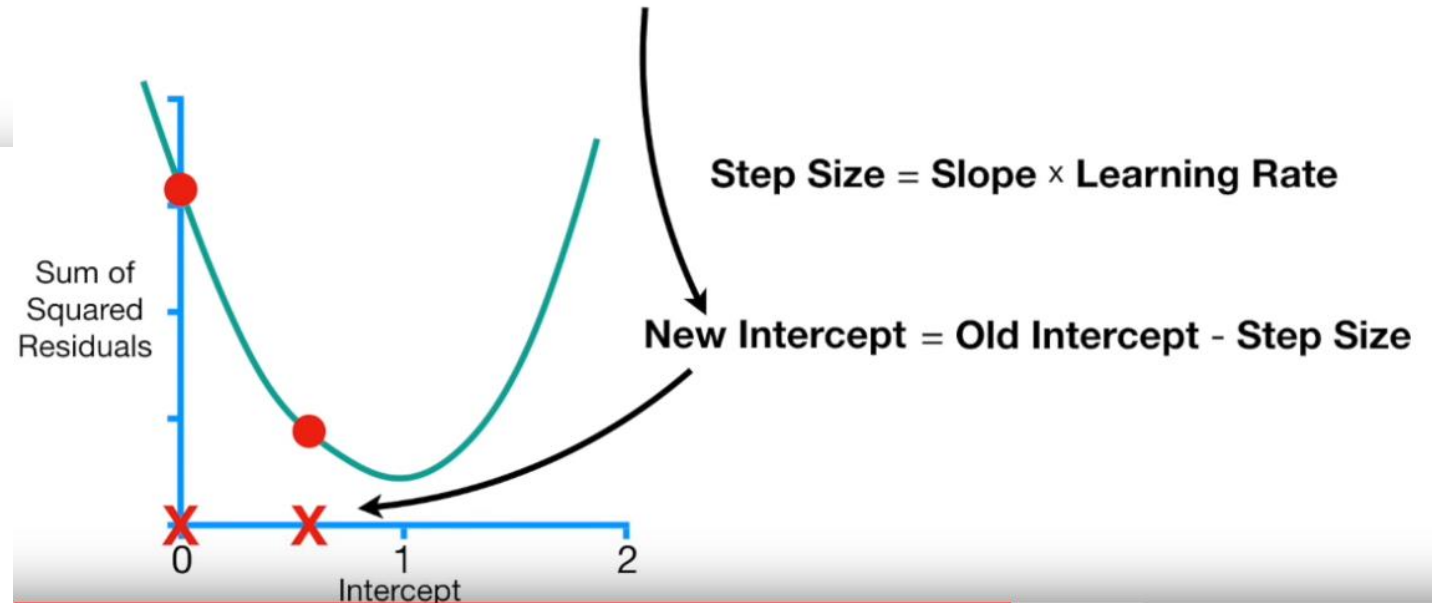
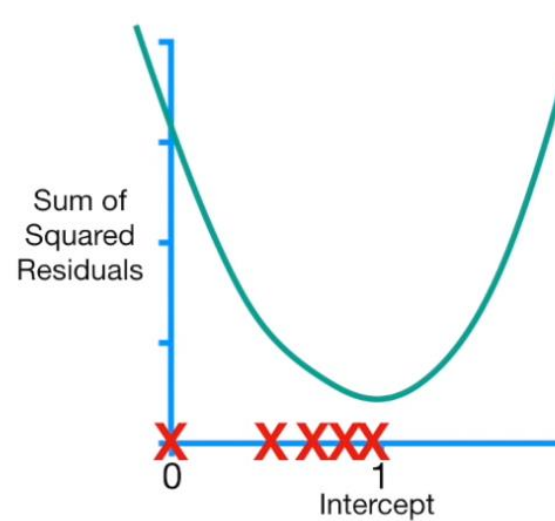
$$-2(1.4 - (\text{intercept} + 0.64 \times 0.5))$$

$$+ -2(1.9 - (\text{intercept} + 0.64 \times 2.3))$$

$$+ -2(3.2 - (\text{intercept} + 0.64 \times 2.9))$$

This makes **Gradient Descent** very useful when it is not possible to solve for where the derivative = 0, and this is why **Gradient Descent** can be used in so many different situations.

Regardless of which **Loss Function** you use, **Gradient Descent** works the same way.



Jak pracuje się w ML

Używa się bibliotek z algorytmami: *scikit-learn*

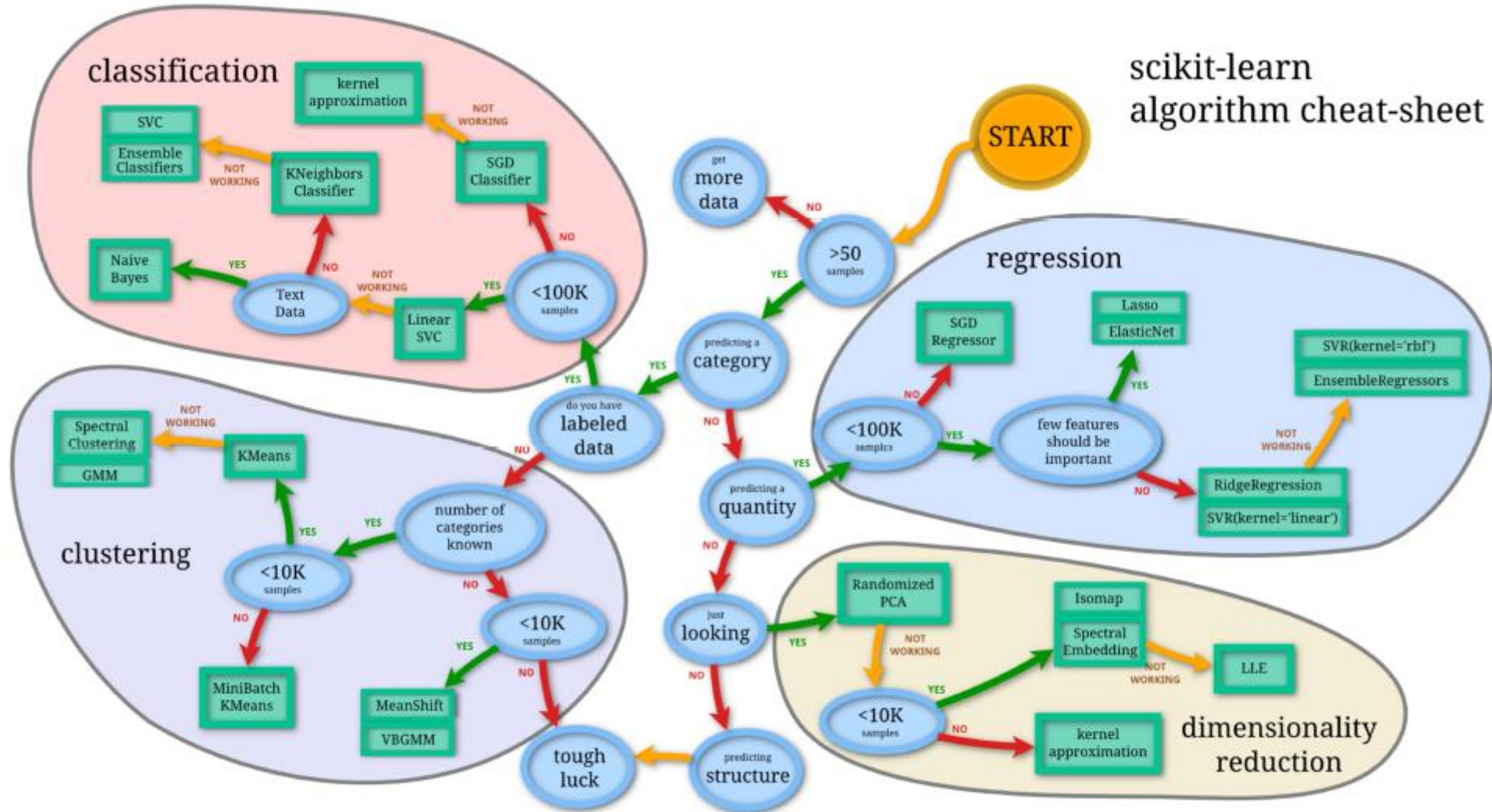


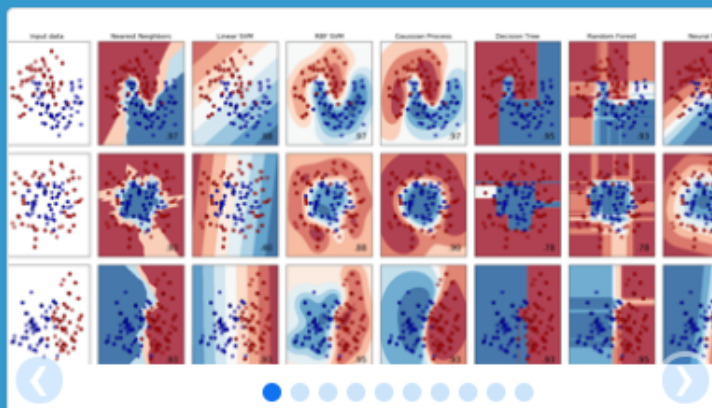
```
1 def train(x, y):
2     from sklearn.linear_model import LinearRegression
3     model = LinearRegression().fit(x,y)
4     return model
5
6 model = train(x,y)
7
8 x_new = 23.0
9 y_new = model.predict(x_new)
10 print(y_new)
```

Każdy inny algorytm może tu być
Wstawiony

Algorytmy zawierają hyperparametry, które
mogą służyć do jego optymalizacji dla danego
zagadnienia.

scikit-learn algorithm cheat-sheet





scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: preprocessing, feature extraction. — Examples

ZASADY PRACY Z ML

Feature engineering

Budowa zbioru danych. Przekształcanie surowych danych w zbiór danych ML

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^N$$

Wyzwaniem jest pozyskiwanie każdego rodzaju informacji:

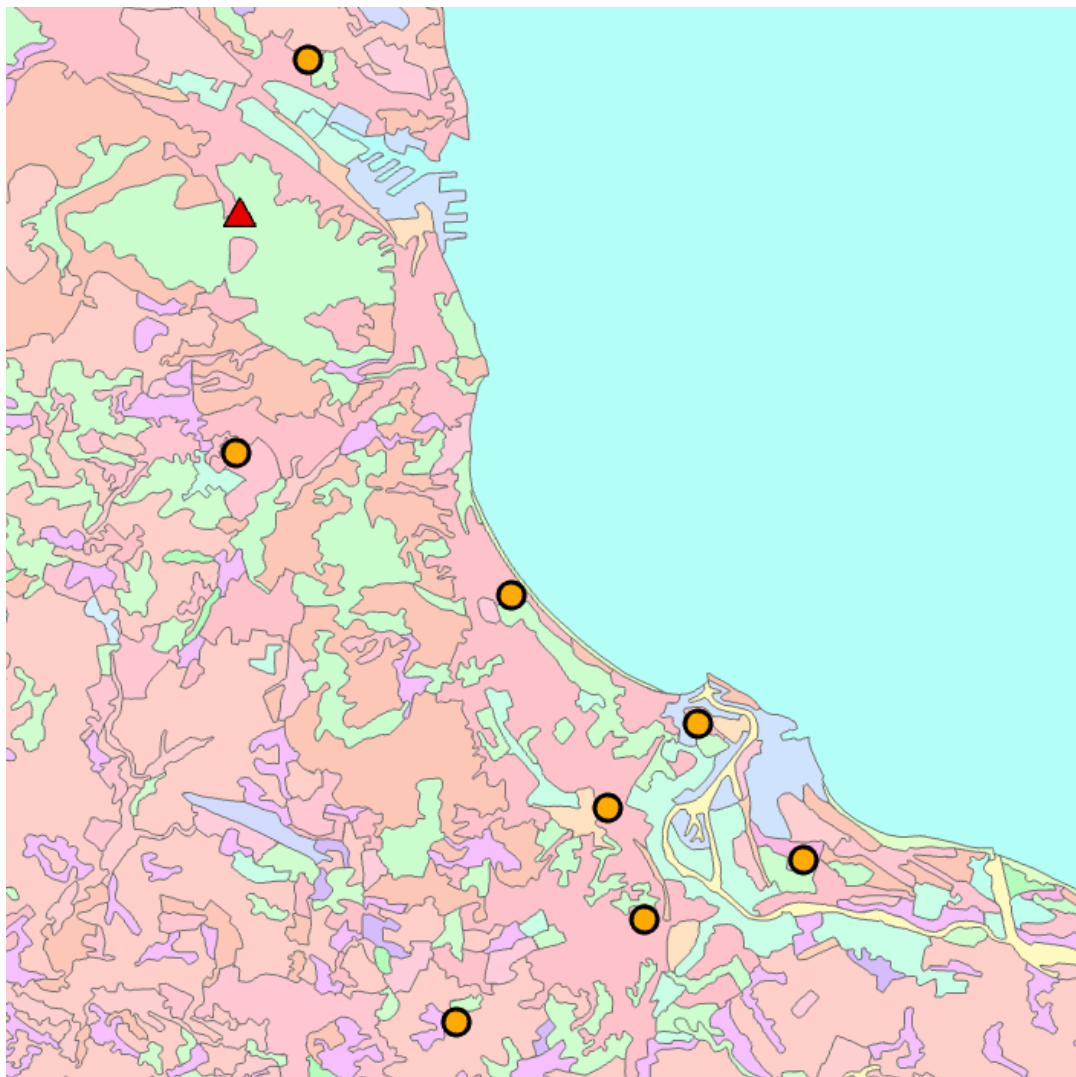
Informative feature – która pozwoli algorytmom ML budowę modelu, który będzie miał wysoką predictive power

```
In [8]: 1 polut.columns
```

```
Out[8]: Index(['Code', 'Concentration', 'TimeSM', 'T', 'H%', 'Wind_dir', 'Wind_vel',  
              'UR200', 'WA200', 'GR200', 'IN200', 'RO200', 'UR600', 'WA600', 'GR600',  
              'IN600', 'RO600', 'UR1000', 'WA1000', 'GR1000', 'IN1000', 'RO1000',  
              'UR2000', 'WA2000', 'GR2000', 'IN2000', 'RO2000', 'UR4000', 'WA4000',  
              'GR4000', 'IN4000', 'RO4000', 'CorIN2', 'CorOUT2', 'CorIN6', 'CorOUT6',  
              'CorIN10', 'CorOUT10', 'CorIN15', 'CorOUT15', 'CorIN30', 'CorOUT30'],  
              dtype='object')
```

	Code	Concentration	TimeSM	T	H%	Wind_dir	Wind_vel
0	PL0496A	38.2	2015-12-31 23:00:00	-13.6	84.0	Wind blowing from the east- southeast	1.0
1	PL0496A	107.6	2016-01-01 00:00:00	-14.5	84.0	Wind blowing from the east	1.0
2	PL0496A	129.2	2016-01-01 01:00:00	-15.0	84.0	Wind blowing from the east	1.0

Uwzględnienie pokrycia terenu




```
In [8]: 1 polut.columns
```

```
Out[8]: Index(['Code', 'Concentration', 'TimeSM', 'T', 'H%', 'Wind_dir', 'Wind_vel',  
              'UR200', 'WA200', 'GR200', 'IN200', 'RO200', 'UR600', 'WA600', 'GR600',  
              'IN600', 'RO600', 'UR1000', 'WA1000', 'GR1000', 'IN1000', 'RO1000',  
              'UR2000', 'WA2000', 'GR2000', 'IN2000', 'RO2000', 'UR4000', 'WA4000',  
              'GR4000', 'IN4000', 'RO4000', 'CorIN2', 'CorOUT2', 'CorIN6', 'CorOUT6',  
              'CorIN10', 'CorOUT10', 'CorIN15', 'CorOUT15', 'CorIN30', 'CorOUT30'],  
              dtype='object')
```

UR200	WA200	GR200	IN200	RO200	UR600	WA600	GR600	IN600	RO600	UR1000	WA1000	GR1000	IN1000	RO1000
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------

62.93	0.0	2.66	20.68	6.74	52.55	0.0	12.26	16.26	13.1	46.9	0.16	17.2	18.07	10.76
-------	-----	------	-------	------	-------	-----	-------	-------	------	------	------	------	-------	-------

62.93	0.0	2.66	20.68	6.74	52.55	0.0	12.26	16.26	13.1	46.9	0.16	17.2	18.07	10.76
-------	-----	------	-------	------	-------	-----	-------	-------	------	------	------	------	-------	-------

62.93	0.0	2.66	20.68	6.74	52.55	0.0	12.26	16.26	13.1	46.9	0.16	17.2	18.07	10.76
-------	-----	------	-------	------	-------	-----	-------	-------	------	------	------	------	-------	-------

62.93	0.0	2.66	20.68	6.74	52.55	0.0	12.26	16.26	13.1	46.9	0.16	17.2	18.07	10.76
-------	-----	------	-------	------	-------	-----	-------	-------	------	------	------	------	-------	-------

62.93	0.0	2.66	20.68	6.74	52.55	0.0	12.26	16.26	13.1	46.9	0.16	17.2	18.07	10.76
-------	-----	------	-------	------	-------	-----	-------	-------	------	------	------	------	-------	-------

```
In [8]: 1 polut.columns
```

```
Out[8]: Index(['Code', 'Concentration', 'TimeSM', 'T', 'H%', 'Wind_dir', 'Wind_vel',  
              'UR200', 'WA200', 'GR200', 'IN200', 'RO200', 'UR600', 'WA600', 'GR600',  
              'IN600', 'RO600', 'UR1000', 'WA1000', 'GR1000', 'IN1000', 'RO1000',  
              'UR2000', 'WA2000', 'GR2000', 'IN2000', 'RO2000', 'UR4000', 'WA4000',  
              'GR4000', 'IN4000', 'RO4000', 'CorIN2', 'CorOUT2', 'CorIN6', 'CorOUT6',  
              'CorIN10', 'CorOUT10', 'CorIN15', 'CorOUT15', 'CorIN30', 'CorOUT30'],  
              dtype='object')
```

UR2000	WA2000	GR2000	IN2000	RO2000	UR4000	WA4000	GR4000	IN4000	RO4000
--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

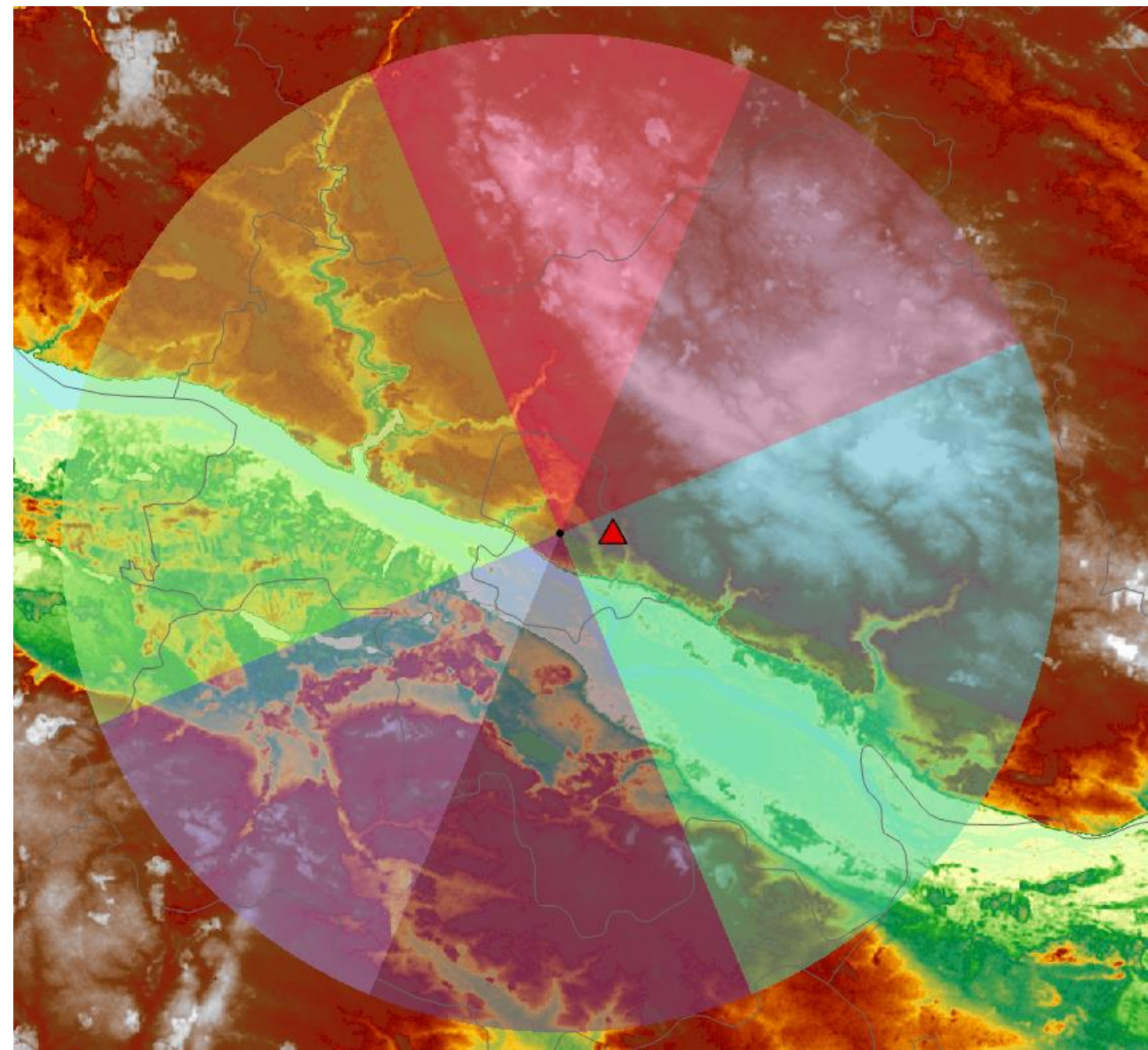
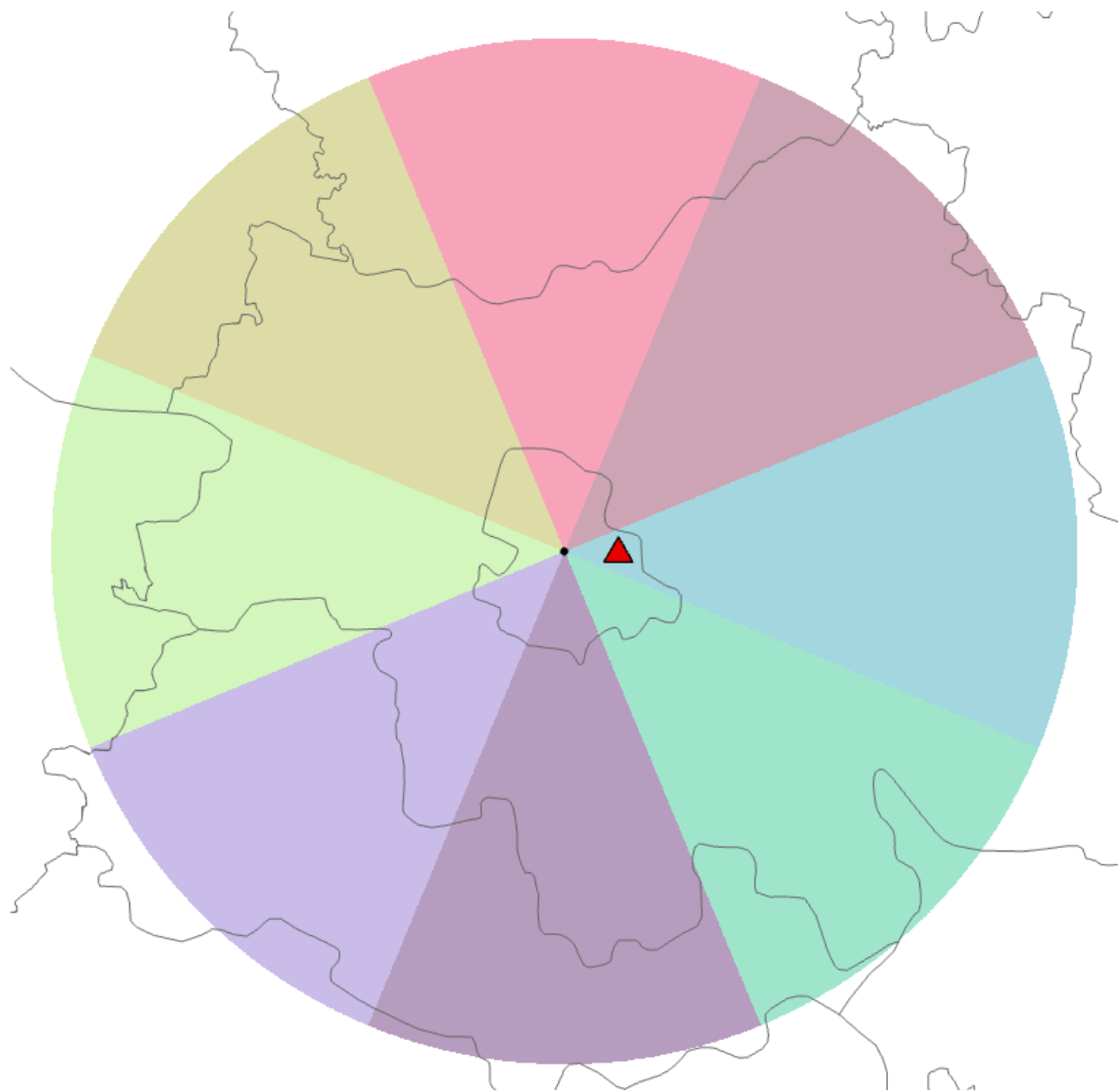
38.72	0.25	25.0	17.11	8.58	26.18	1.04	40.96	16.98	6.31
-------	------	------	-------	------	-------	------	-------	-------	------

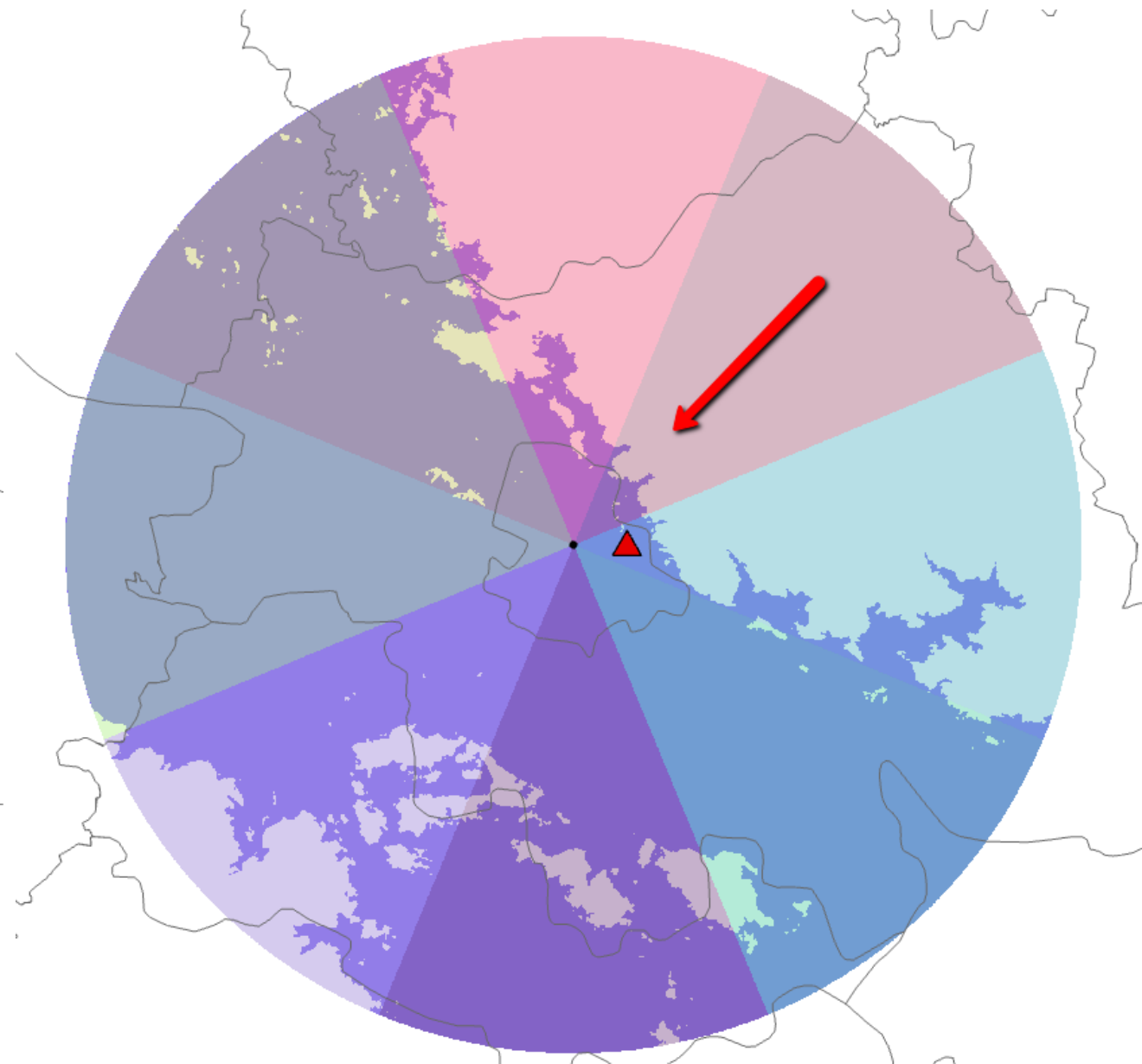
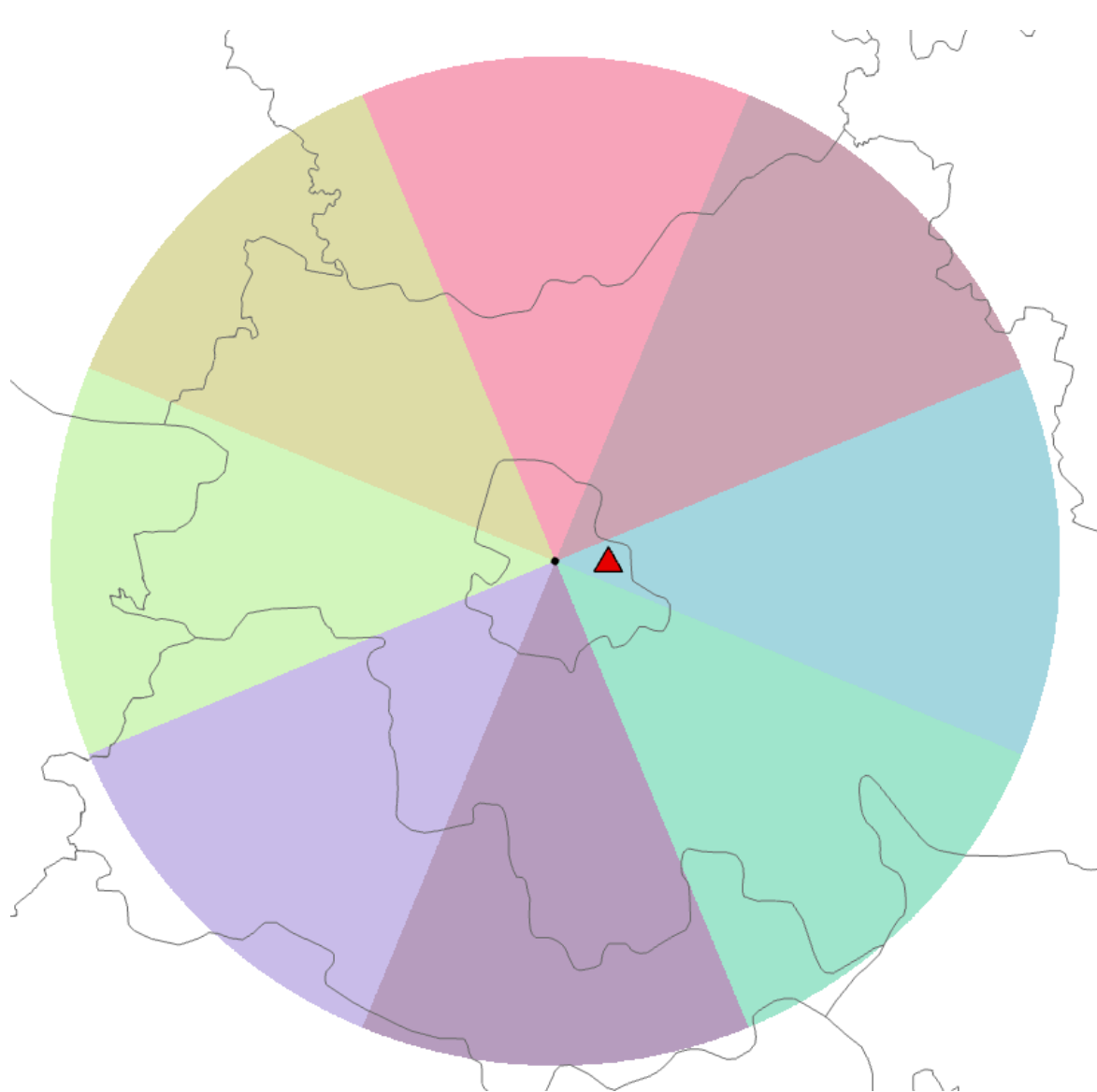
38.72	0.25	25.0	17.11	8.58	26.18	1.04	40.96	16.98	6.31
-------	------	------	-------	------	-------	------	-------	-------	------

38.72	0.25	25.0	17.11	8.58	26.18	1.04	40.96	16.98	6.31
-------	------	------	-------	------	-------	------	-------	-------	------

38.72	0.25	25.0	17.11	8.58	26.18	1.04	40.96	16.98	6.31
-------	------	------	-------	------	-------	------	-------	-------	------

38.72	0.25	25.0	17.11	8.58	26.18	1.04	40.96	16.98	6.31
-------	------	------	-------	------	-------	------	-------	-------	------





DEM korytarze wiatrowe

```
In [8]: 1 polut.columns
```

```
Out[8]: Index(['Code', 'Concentration', 'TimeSM', 'T', 'H%', 'Wind_dir', 'Wind_vel',  
              'UR200', 'WA200', 'GR200', 'IN200', 'RO200', 'UR600', 'WA600', 'GR600',  
              'IN600', 'RO600', 'UR1000', 'WA1000', 'GR1000', 'IN1000', 'RO1000',  
              'UR2000', 'WA2000', 'GR2000', 'IN2000', 'RO2000', 'UR4000', 'WA4000',  
              'GR4000', 'IN4000', 'RO4000', 'CorIN2', 'CorOUT2', 'CorIN6', 'CorOUT6',  
              'CorIN10', 'CorOUT10', 'CorIN15', 'CorOUT15', 'CorIN30', 'CorOUT30'],  
              dtype='object')
```

CorIN2	CorOUT2	CorIN6	CorOUT6	CorIN10	CorOUT10	CorIN15	CorOUT15	CorIN30	CorOUT30
0.95	0.955	0.54	0.87	0.26	0.925	0.115	0.83	0.315	0.845
0.92	0.910	0.40	0.97	0.24	0.960	0.110	0.97	0.230	0.970
0.92	0.910	0.40	0.97	0.24	0.960	0.110	0.97	0.230	0.970
0.92	0.910	0.40	0.97	0.24	0.960	0.110	0.97	0.230	0.970
0.92	0.910	0.40	0.97	0.24	0.960	0.110	0.97	0.230	0.970

One-Hot Encoding

Niektóre algorytmy pracują tylko z danymi numerycznymi

Np. kolor = czerwony, żółty, zielony

Czerwony	Żółty	Zielony
1	0	0
0	1	0
1	0	0

Binning, bucketing

Konwersja numerical features na categorical features

Zamiana ciągłych wartości na wiele bins:

Wiek:

Bin0_5

Bin6_10

Bin11-15

Bin15_30

Bin30_50

Bin50_70

Bin70-90

Wiek	Bin0_5	Bin6_10	Bin11_15	Bin15_30	Bin30_50	Bin50_70	Bin70_90
22	0	0	0	1	0	0	0
12	0	0	1	0	0	0	0

Normalization

Zamiana faktycznego zakresu wartości na zakres [-1,1] lub [0,1]

$$\bar{x}^{(j)} = \frac{x^{(j)} - \min^{(j)}}{\max^{(j)} - \min^{(j)}},$$

Nie jest bezwzględnie wymagana. Najczęściej przyspiesza działanie procedury. Unika się także potencjalnych problemów z pracą komputera z małymi i dużymi wartościami.

Standarization (z-score normalization)

Przeskalowanie do standardowego rozkładu normalnego

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}}.$$

Unsupervised LM alorytmy działają lepiej po standaryzacji niż normalizacji.

Standaryzacja jest także wskazana, kiedy features mają rozkłady zbliżone do normalnego.

Standaryzacja jest wskazana, kiedy występują duże i małe „outliers” (normalizacja ścieśni wtedy wartości w wąskim zakresie)

W pozostałych sytuacjach wskazana jest normalizacja.

Postępowanie z brakującymi danymi

1. Usunięcie features z brakującymi danymi.
2. Wykorzystanie algorytmu ML, który radzi sobie z brakiem danych.
3. Wykorzystanie **imputation technique**

Najprostsza technika polega na zastąpieniu brakujących wartości średnią

$$\hat{x}^{(j)} \leftarrow \frac{1}{N} x^{(j)}.$$

Bardziej zaawansowana może wykorzystywać regresję liniową.

Podział danych na 3 zbiory (najczęściej tylko 2 treningowy i testowy)

1. Zbiór treningowy (70%)
2. Zbiór walidacyjny (15%)
3. Zbiór testowy (15%)

Przy bardzo dużych liczbach danych (95%,2.5%,2.5%)

Zbiór walidacyjny jest używany do:

1. Wyboru algorytmu
2. Wyznaczeniu hiperparametrów

Ocena działania modelu

Dla danych ciągłych wyznacza się MSE (dla zbioru treningowego i testowego – powinny być porównywalne – jak nie możliwy overfitting)

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

Dla klas (danych jakościowych) stosuje się szereg metod:

1. Confusion matrix
2. Accuracy
3. Cost-sensitive accuracy
4. Precision / recall

Confusion matrix

	spam (predicted)	not_spam (predicted)
spam (actual)	23 (TP)	1 (FN)
not_spam (actual)	12 (FP)	556 (TN)

Z 24 przykładów, które były spamem 23 zostały poprawnie sklasyfikowane (TP – true positives), model błędnie sklasyfikował 1 jako spam (false negative – FN). 556 przykładów, które nie były spamem zostały sklasyfikowane jako true negative TN, a 12 jako false positive – FP.

Na podstawie **confusion matrix**,
można wyznaczyć **precision i recall**

$$\text{precision} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{recall} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Cross-Validation

First, you fix the values of the hyperparameters you want to evaluate.

Then you split your training set into several subsets of the same size. Each subset is called a *fold*. Typically, five-fold cross-validation is used in practice. With five-fold cross-validation, you randomly split your training data into five folds: $\{F_1, F_2, \dots, F_5\}$. Each F_k , $k = 1, \dots, 5$ contains 20% of your training data. Then you train five models as follows. To train the first model, f_1 , you use all examples from folds F_2, F_3, F_4 , and F_5 as the training set and the examples from F_1 as the validation set. To train the second model, f_2 , you use the examples from folds F_1, F_3, F_4 , and F_5 to train and the examples from F_2 as the validation set. You continue building models iteratively like this and compute the value of the metric of interest on each validation set, from F_1 to F_5 .

Then you average the five values of the metric to get the final value.