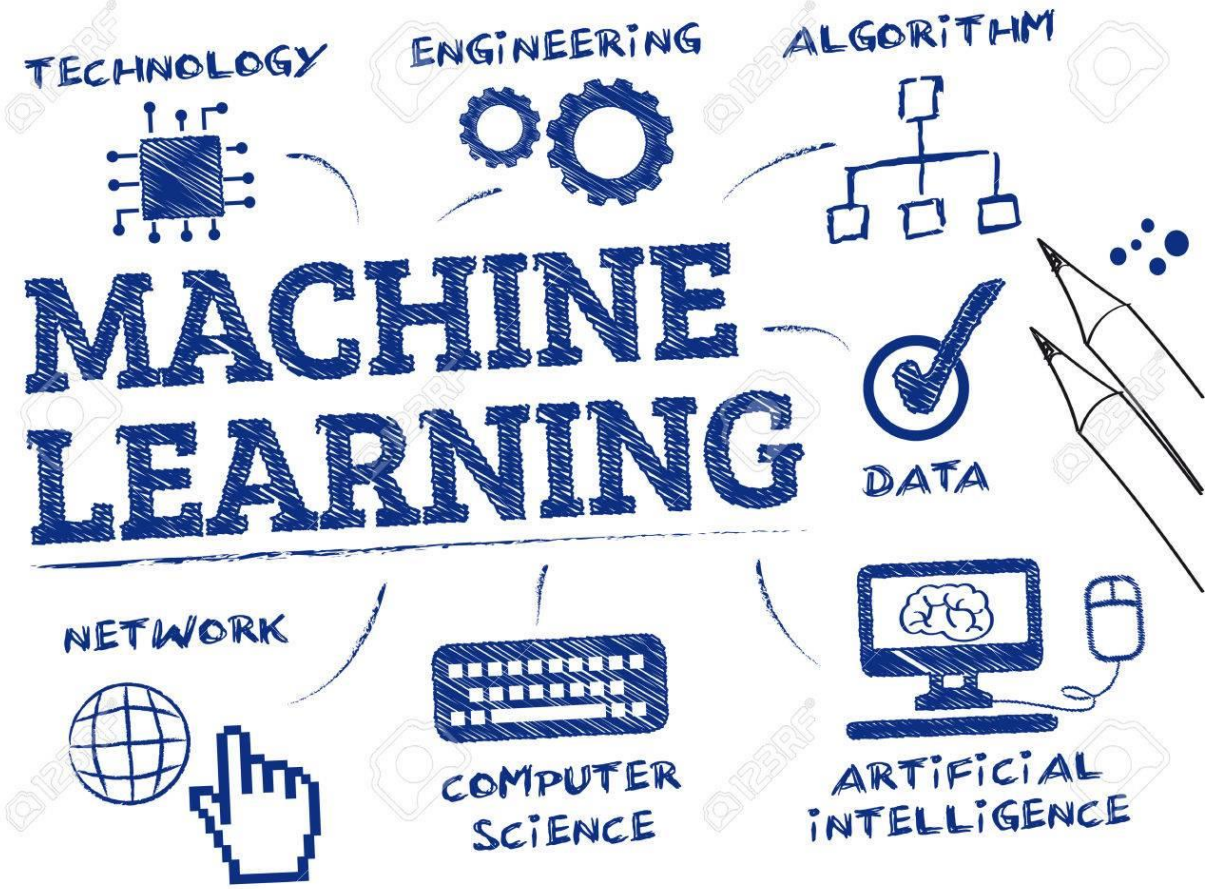


ML to proces wykorzystania danych (przyrodniczych, wytworzonych przez człowieka, wygenerowanych przez algorytm) do budowy modeli, które pozwalają na przeprowadzanie prognozowania lub wygenerowanie praktycznego rozwiązania.

Modele wykorzystują dane i ich strukturę a nie reguły typu 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?

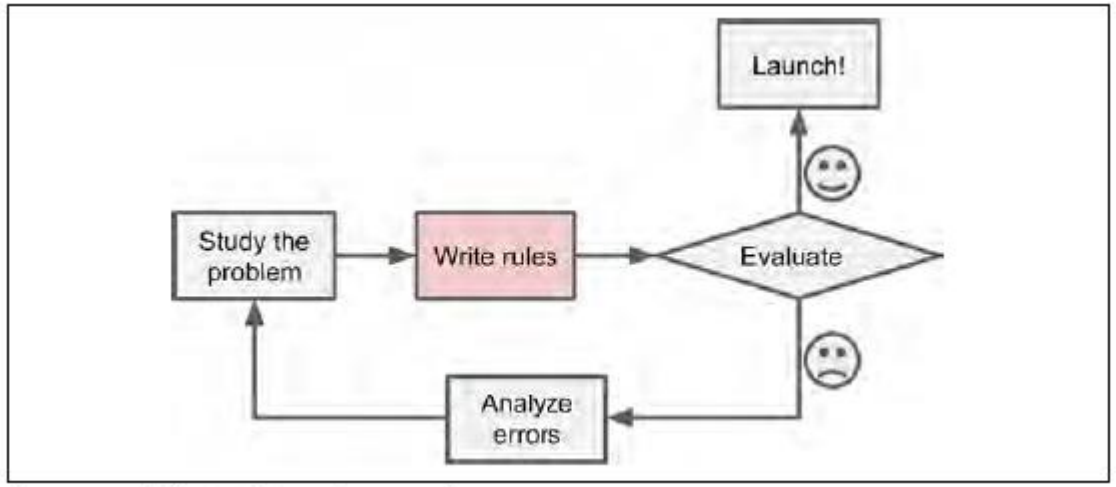


Figure 1-1. The traditional approach

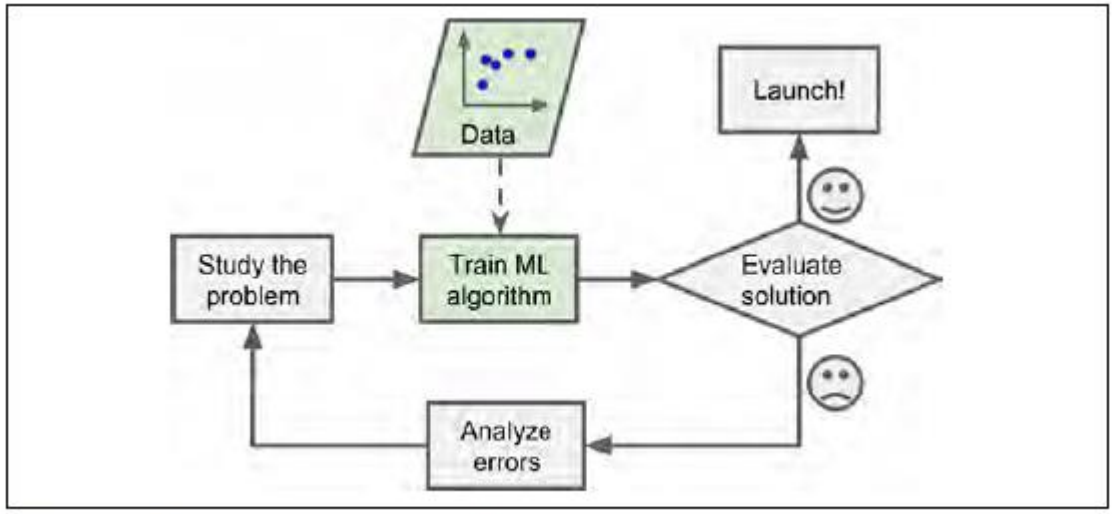


Figure 1-2. Machine Learning approach

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

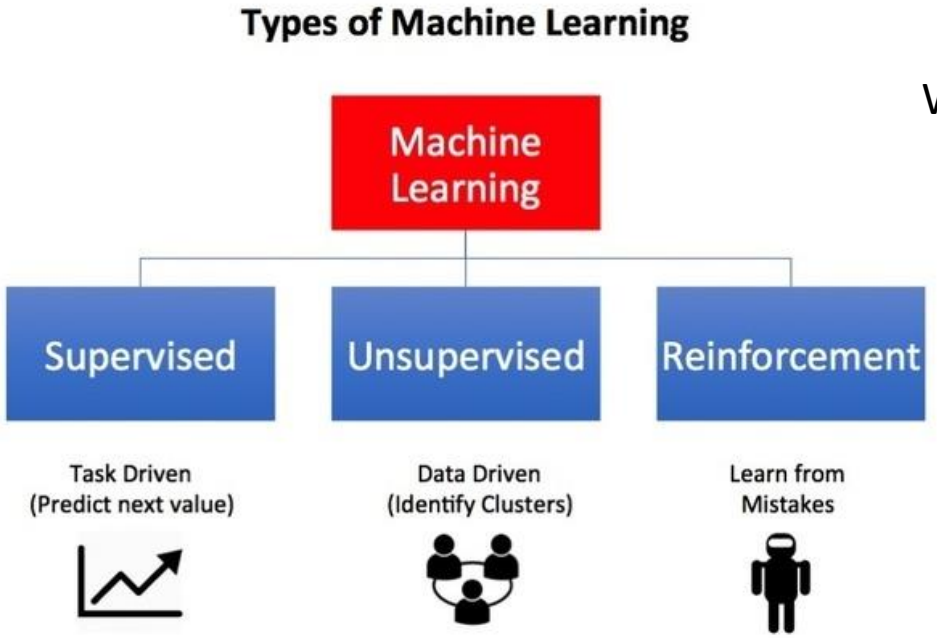
ML jest przydatne do:

• Problemów dla których istniejące rozwiązania wymagają uciążliwego ręcznego dopasowywania lub długiej listy reguł. Pojedyńczy 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.



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

W **supervised learning** zbiór danych to **labeled examples (observations)**:

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

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

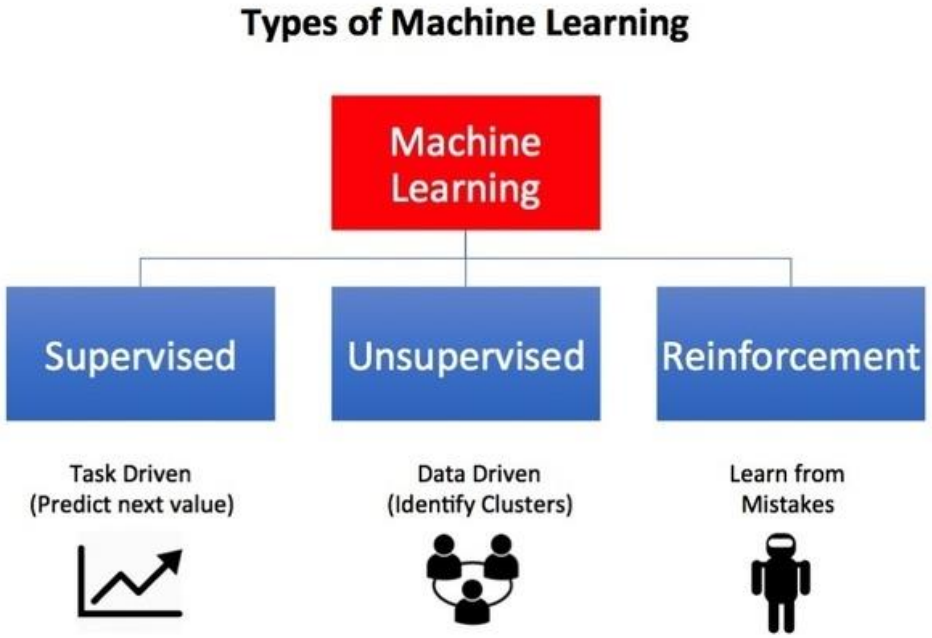
jest to wektor, którego każdy wymiar $j = 1, \ldots, D$

zawiera value (zmienne) opisujące example nazywane **feature (atribute)**

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

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

albo jest liczbą rzeczywistą (lub bardziej złożoną strukturą)



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

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

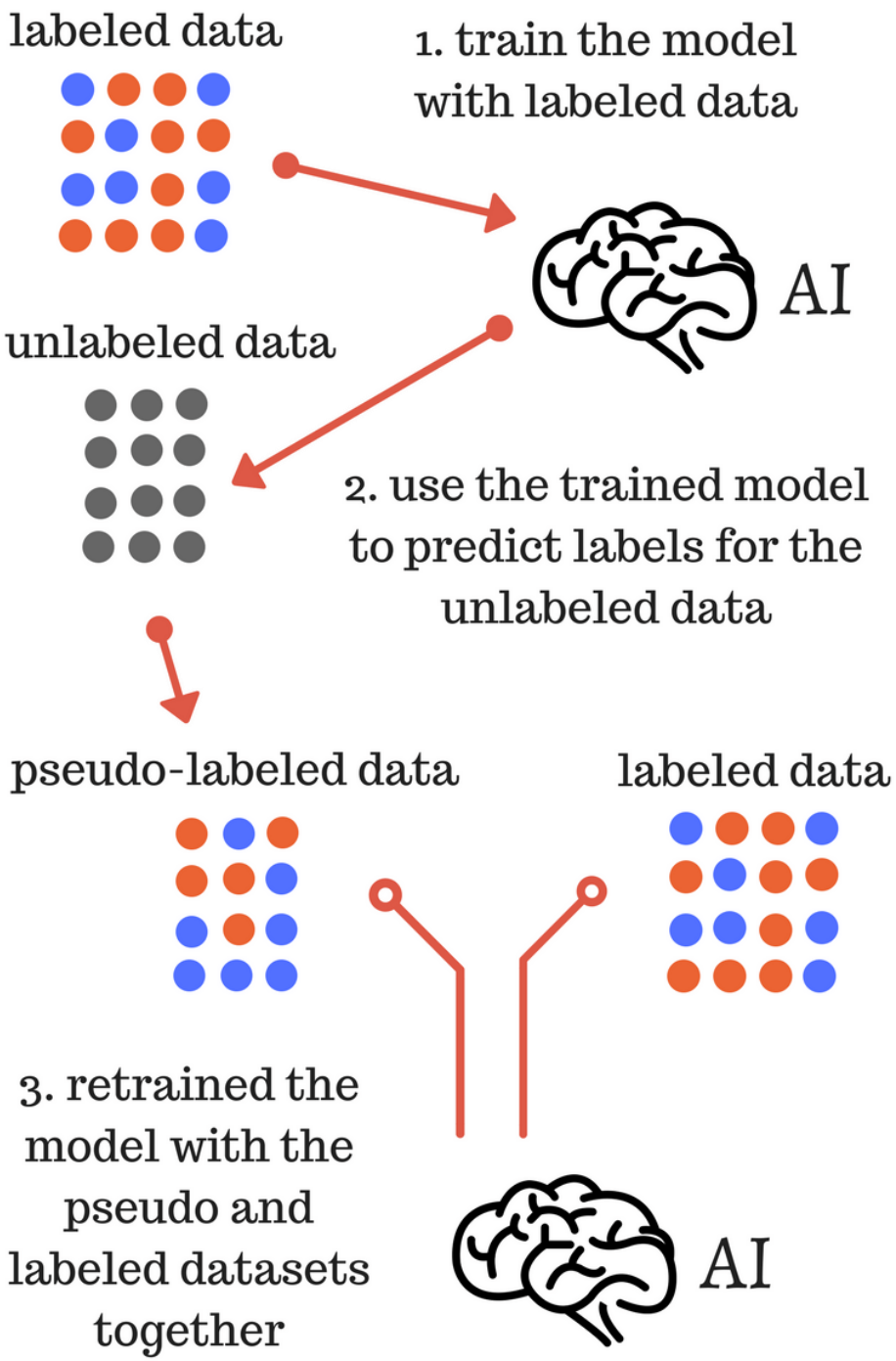
Każdy element $\mathbf{x}_i$ jest nazywany **feature vector ,**

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

zawiera value opisujące example nazywane **feature**

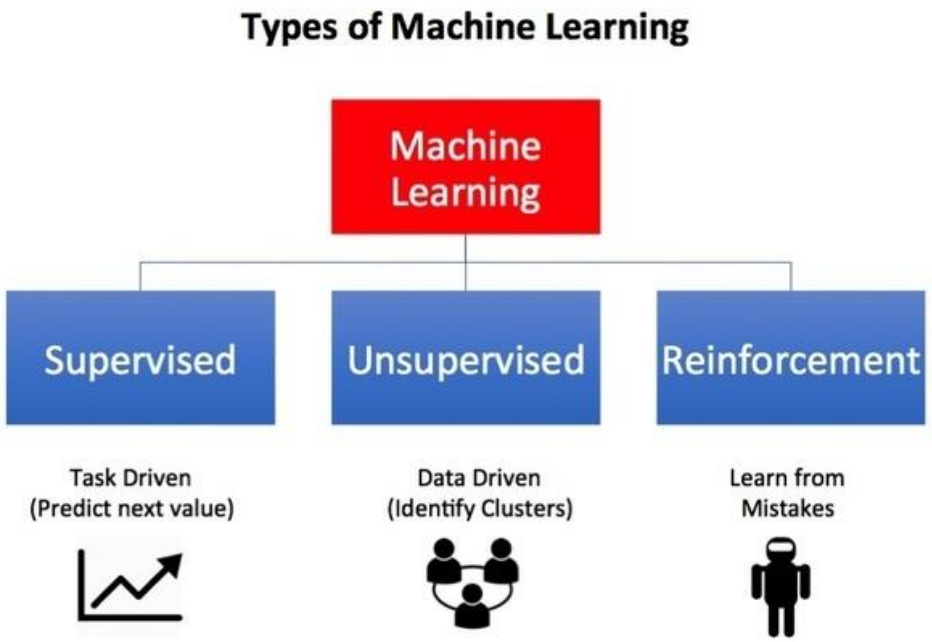
i jest oznaczone jako $x^{(j)}$

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 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.



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.

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

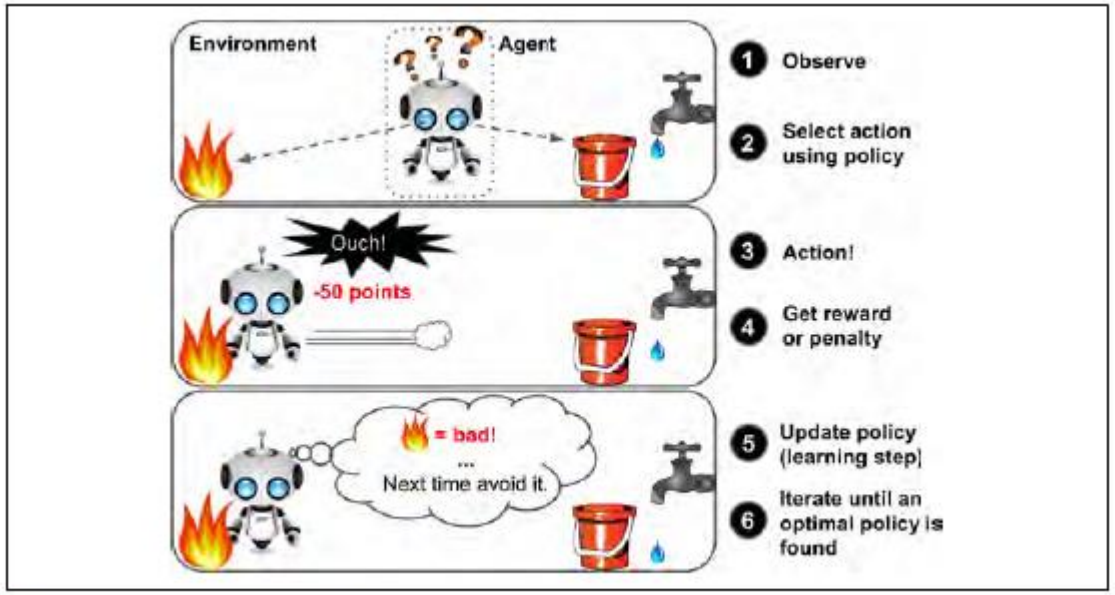


Figure 1-12. Reinforcement Learning

ALGORYTMY ML

Najważniejsze supervised learning algorithms

- Gaussian Naive Bayes
- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees
- Random Forests
- Neural networks

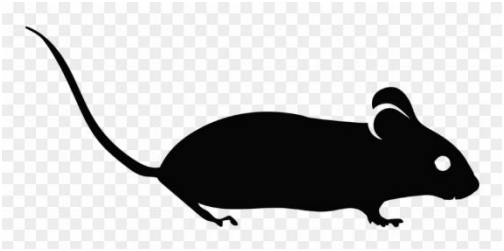
Najważniejsze unsupervised learning algorithms

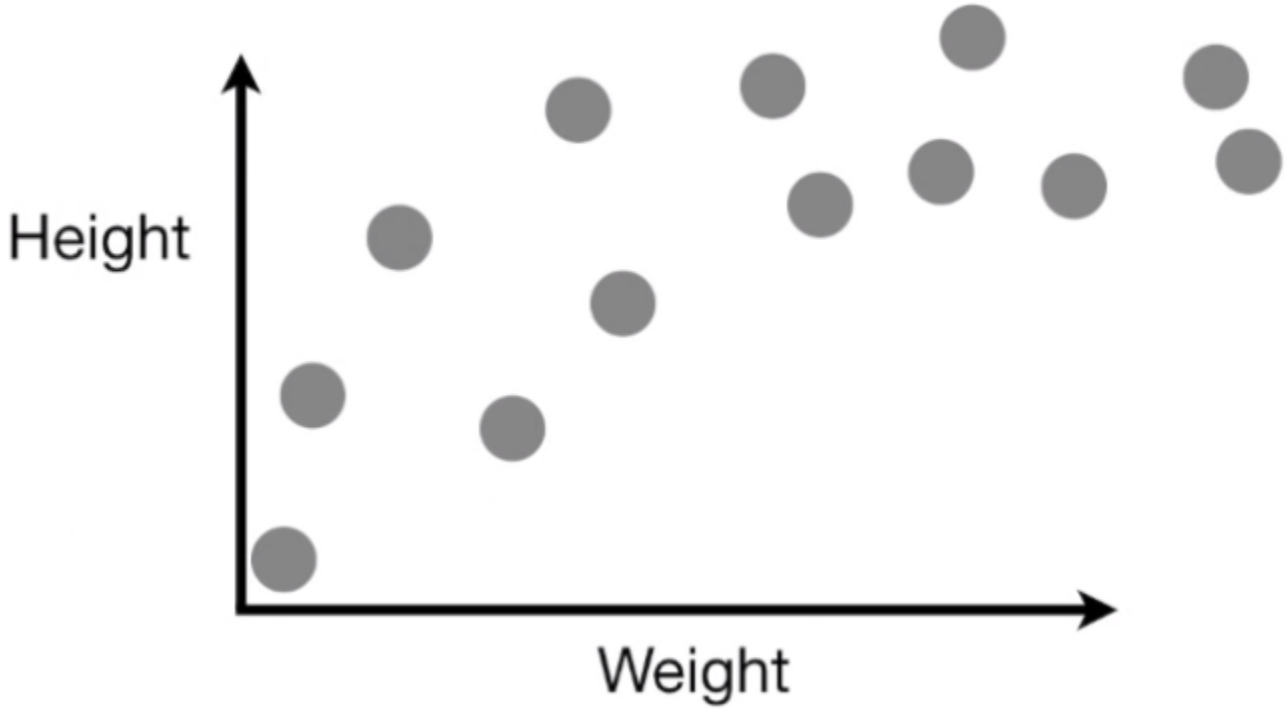
- **Clustering**
  - k-Means
  - Hierarchical Cluster Analysis (HCA)
  - Expectation Maximization
- **Visualization** and **dimensionality reduction**
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)

TECHNIKI ML

https://www.youtube.com/watch?v=EuBBz3bI-aA&t=26s

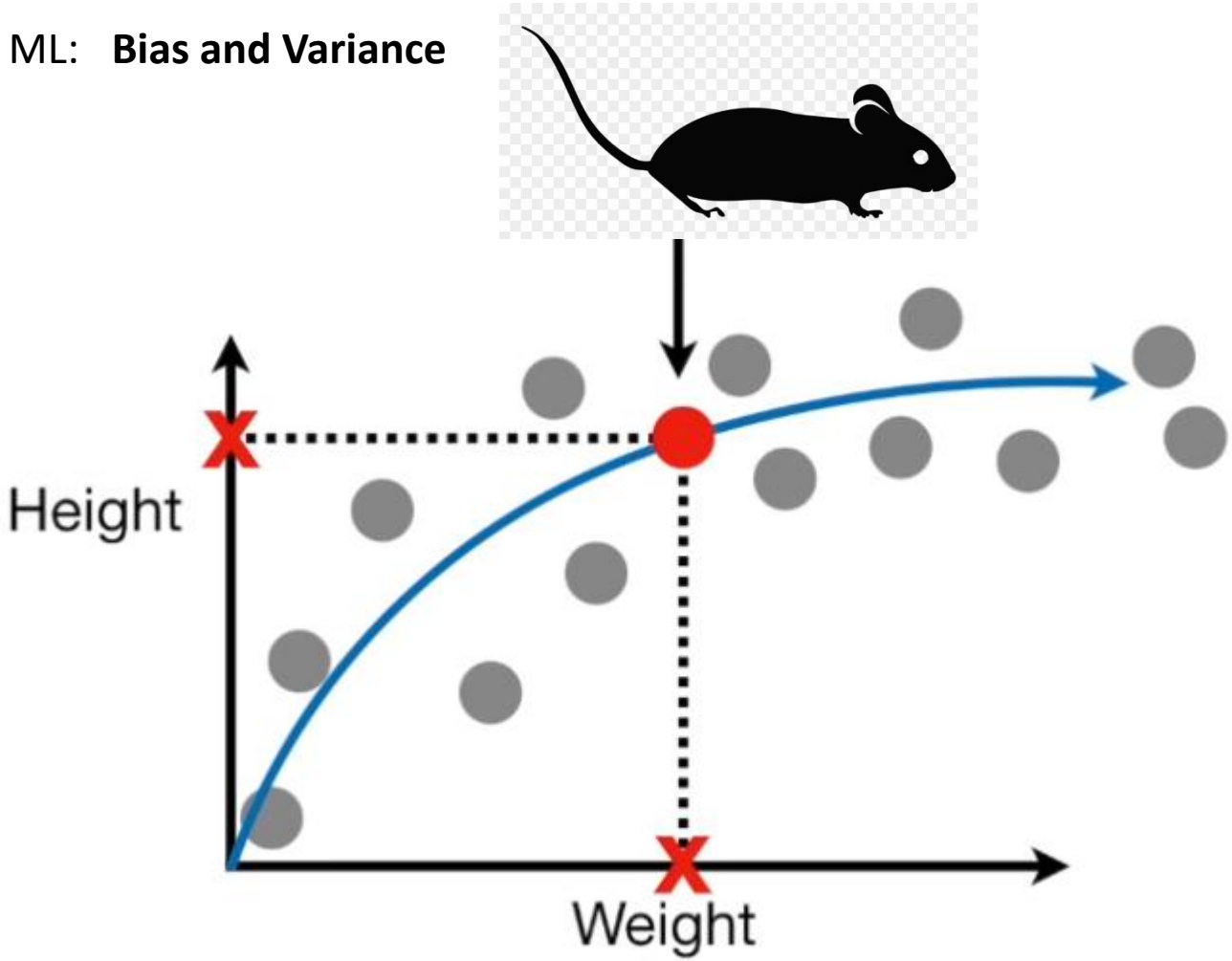
PODSTAWOWE POJĘCIA: **Bias and Variance**





Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

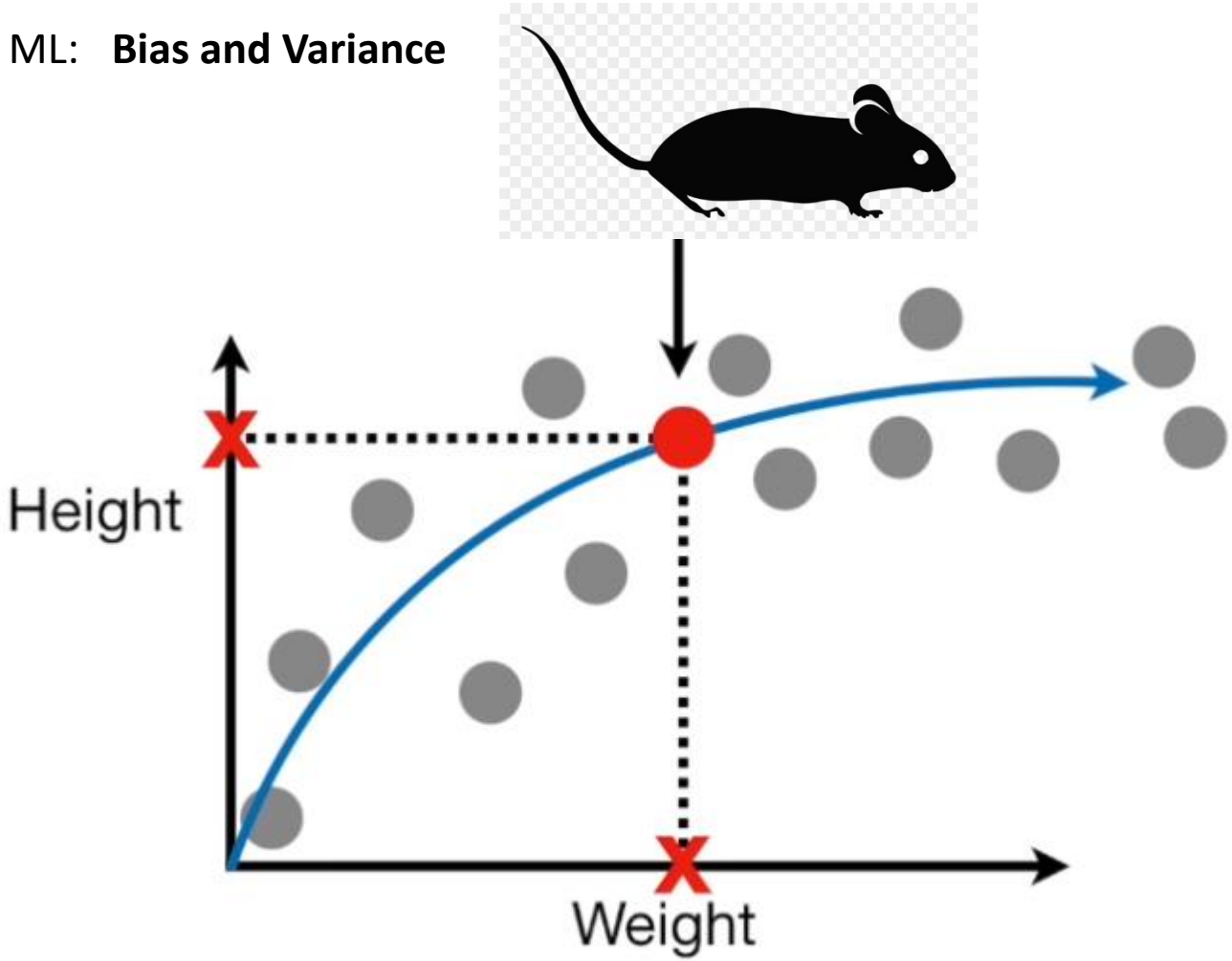
PODSTAWOWE POJĘCIA ML: **Bias and Variance**



Sytuacja idealna:
znane jest wyrażenie

Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

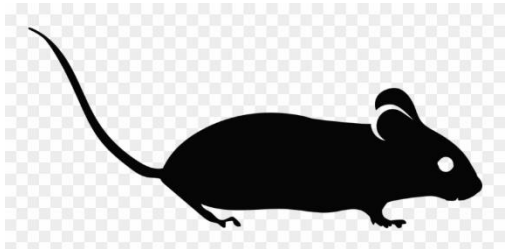
PODSTAWOWE POJĘCIA ML: **Bias and Variance**

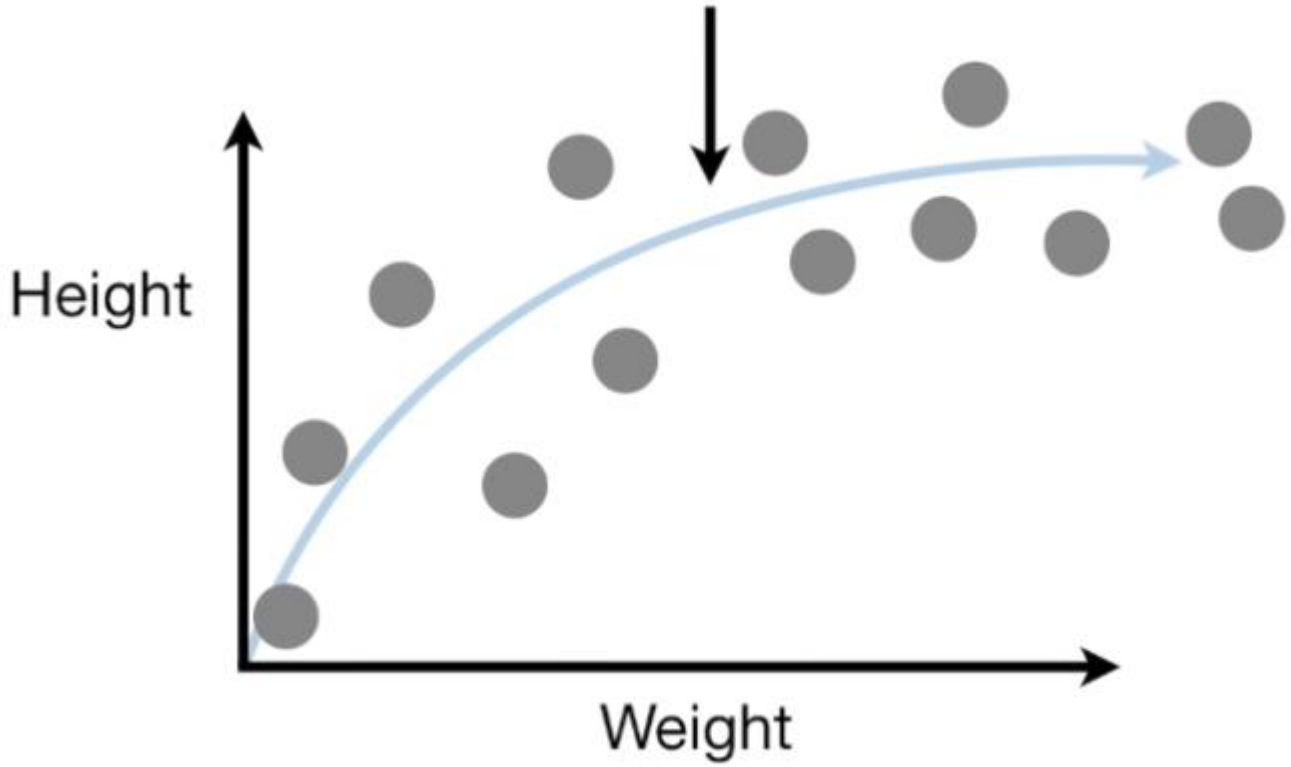


Sy[illegible]
zr[illegible]enie

Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

PODSTAWOWE POJĘCIA ML: **Bias and Variance**

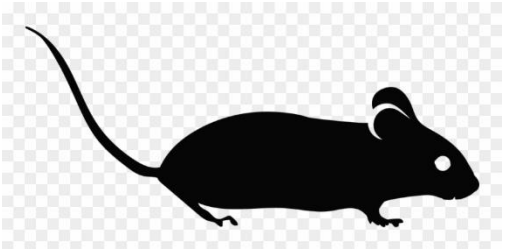




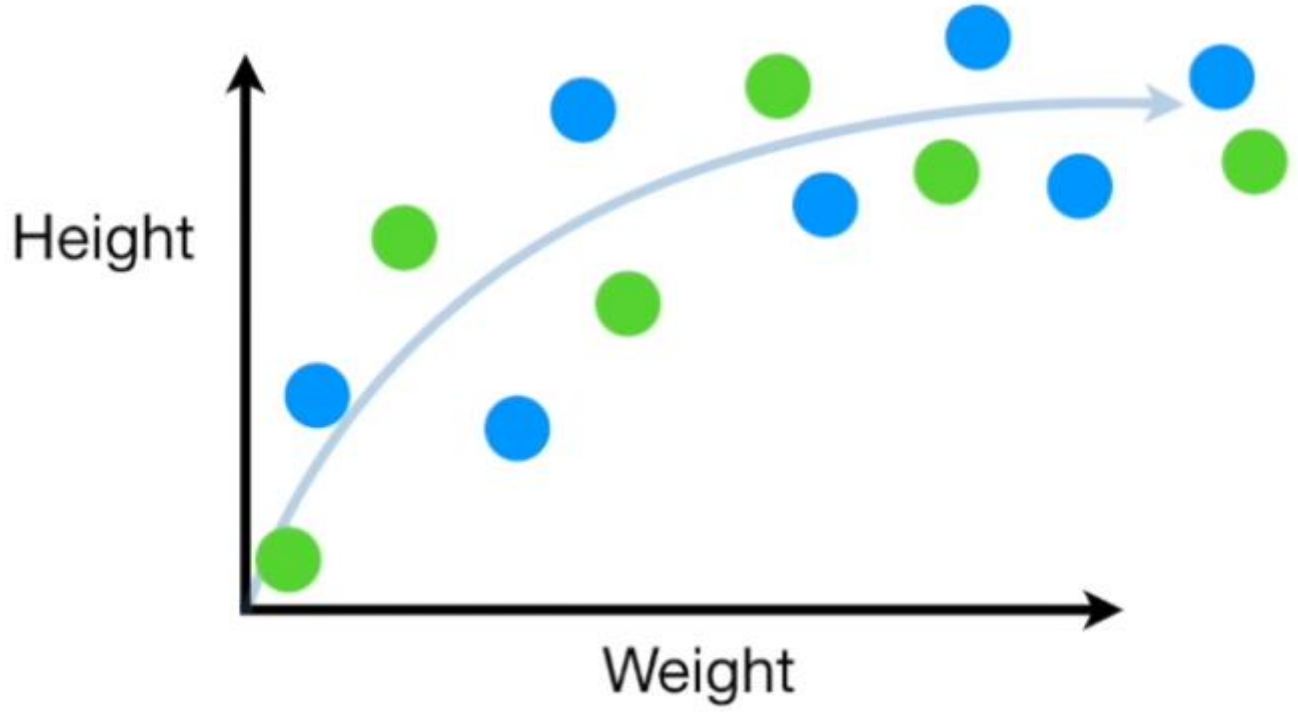
Wykorzystanie ML do modelowania tej zależności

Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

PODSTAWOWE POJĘCIA ML: **Bias and Variance**



Wykorzystanie ML do modelowania tej zależności



Krok 1.

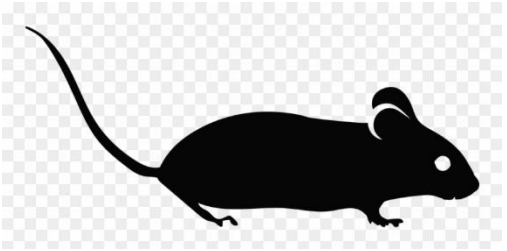
Dzielimy dane na dwie grupy:

training set

testing set

Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

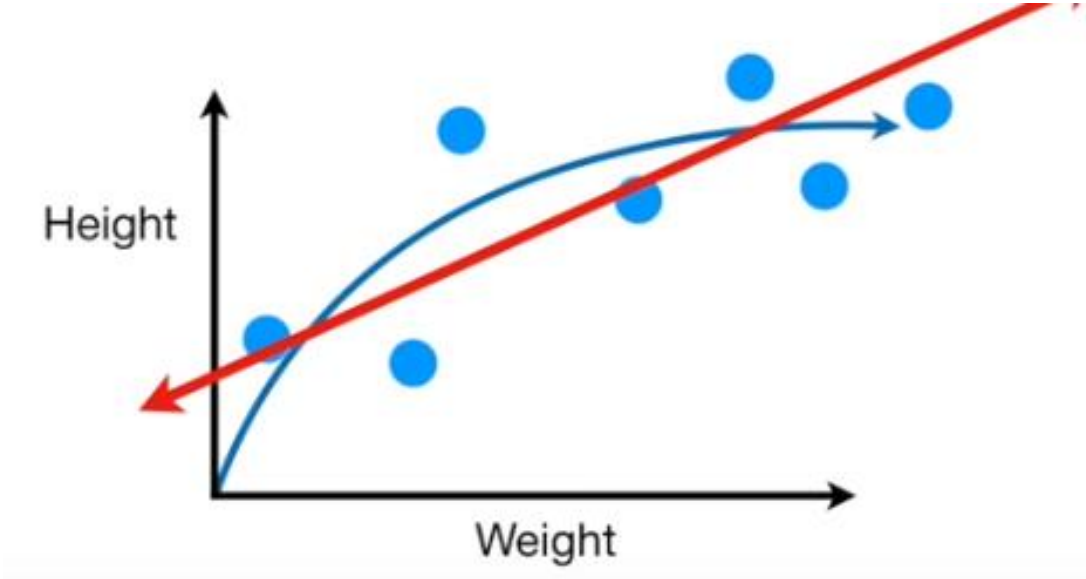
PODSTAWOWE POJĘCIA ML: **Bias and Variance**



Wykorzystanie ML do modelowania tej zależności

Pierwszy ML algorytm

Metoda przypisuje prostą do danych



Krok 2.

Pracujemy tylko na:

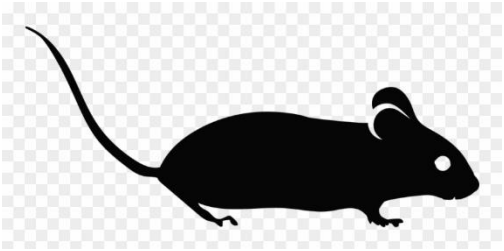
training set

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

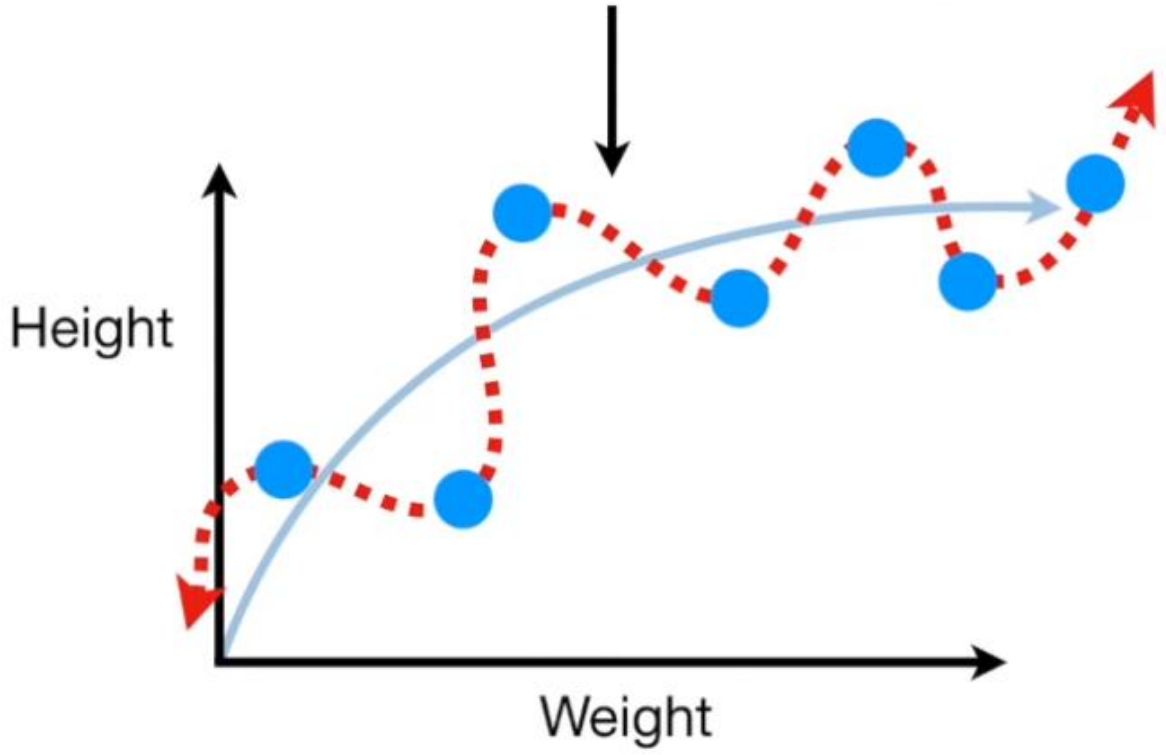
PODSTAWOWE POJĘCIA ML:



Wykorzystanie ML do modelowania tej zależności

ML algorytm

Inna metoda przypisuje krzywą do danych

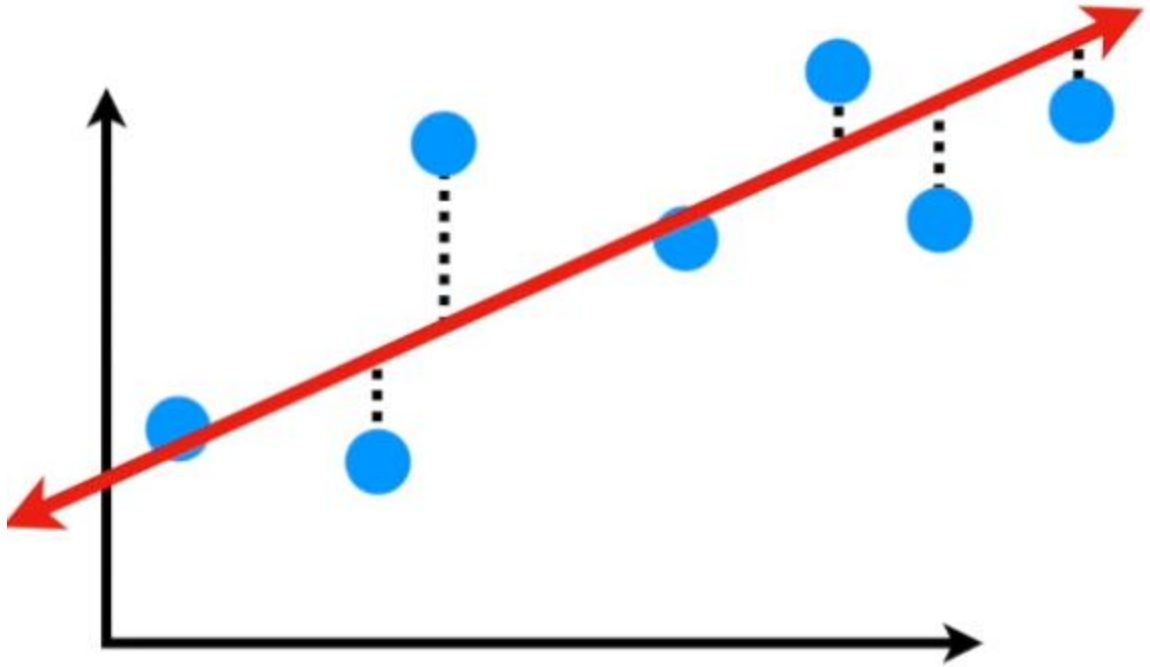


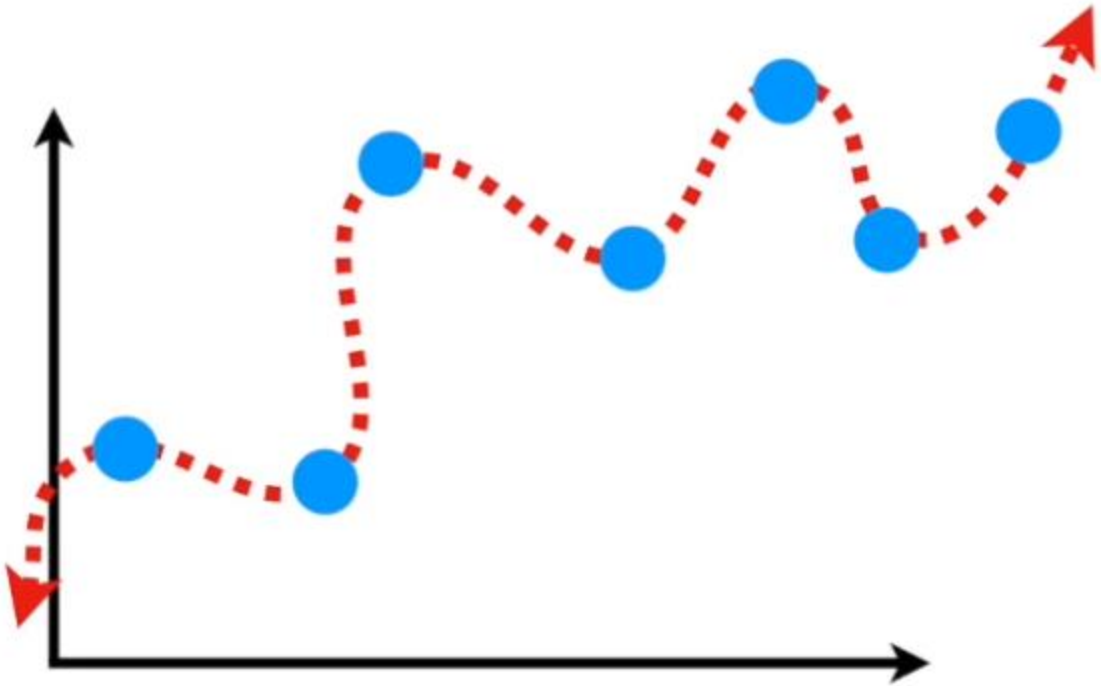
Ze względu na możliwe zakrzywienia linii bias jest mały

Zadanie: Przewidzieć wysokość myszy na podstawie jej wagi

TRAINING

LEPIEJ



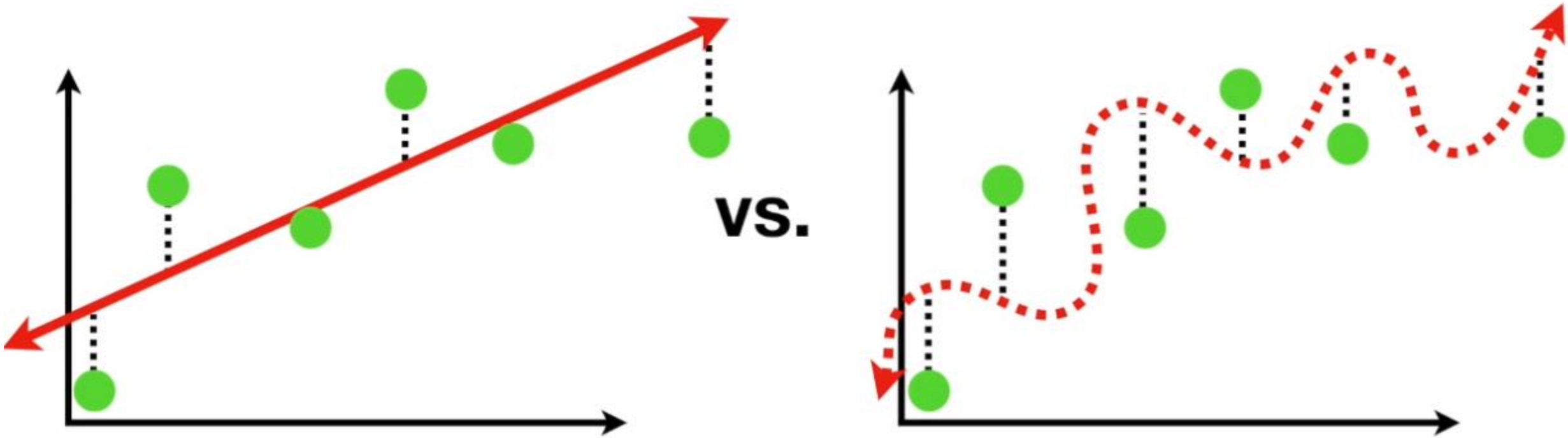


Miara suma kwadratów (im większa tym gorzej - **loss function**)

TESTING

LEPIEJ

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

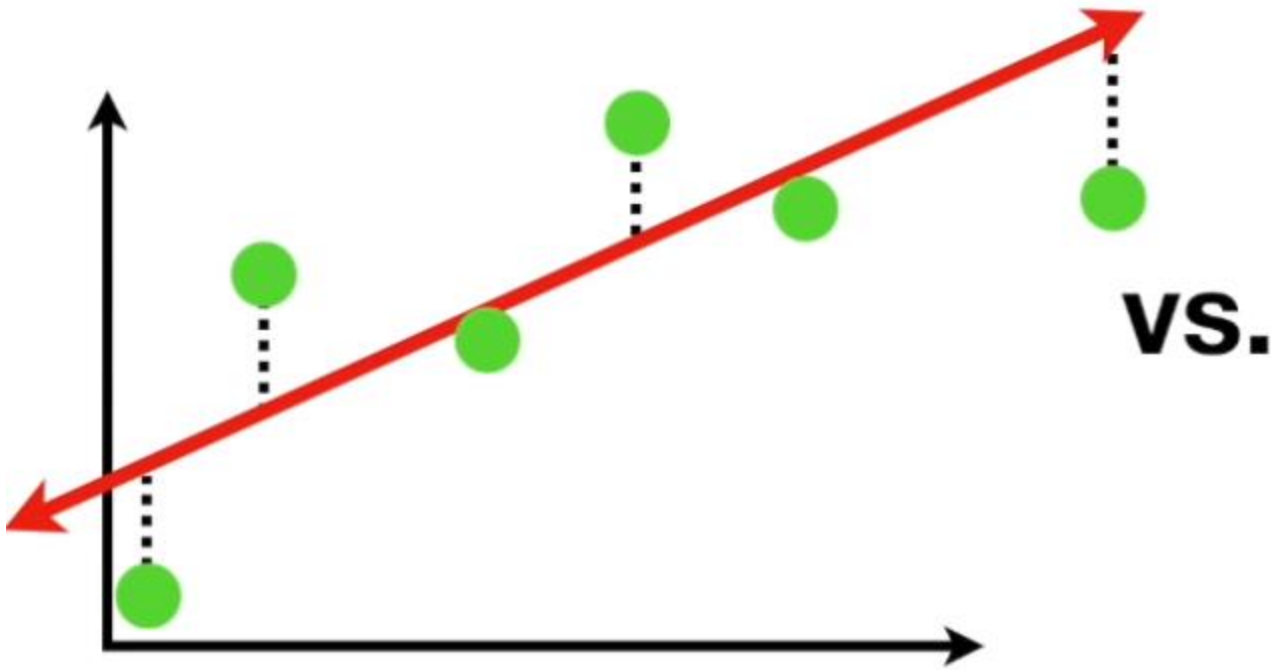


Miara suma kwadratów (im większa tym gorzej - **loss function**)

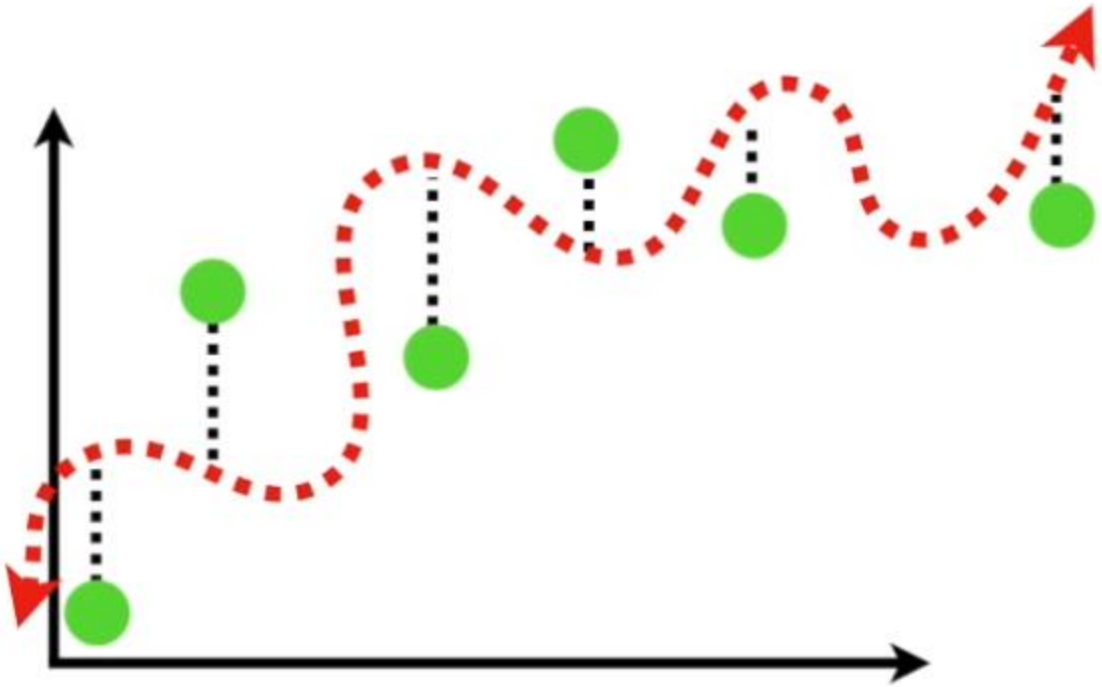
W większości algorytmów celem jest minimalizacja **loss function**

TESTING

LEPIEJ

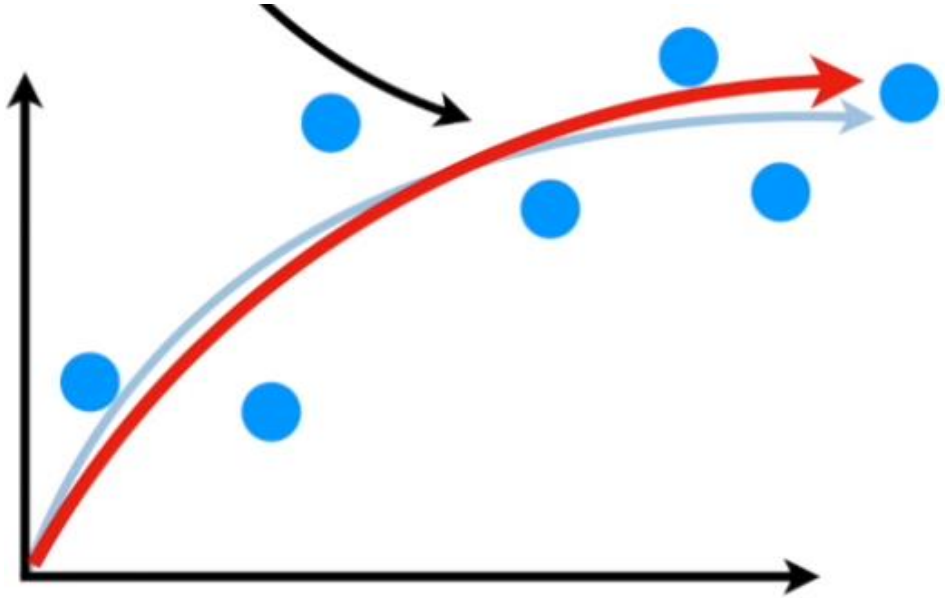


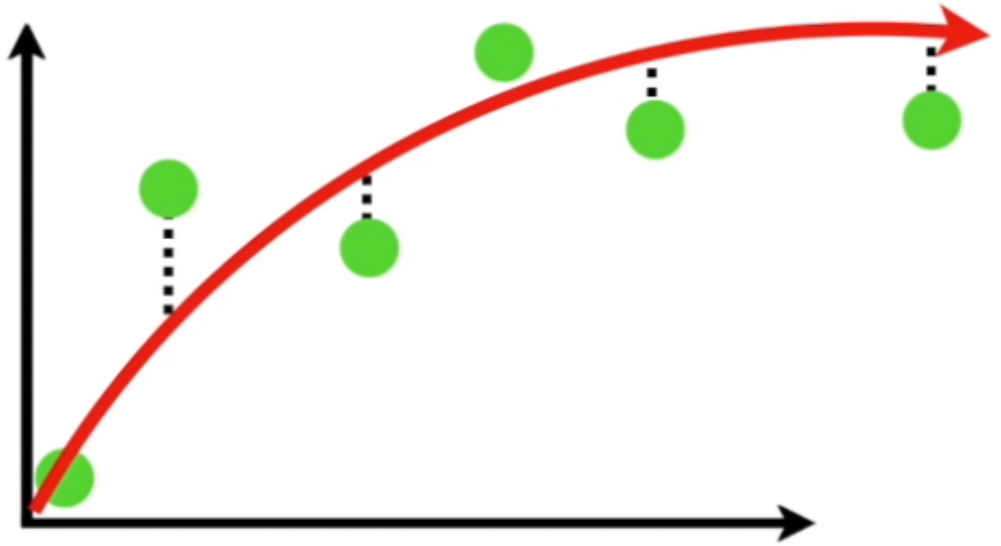
RÓŻNICA w dopasowaniu do zbiorów danych nazywa się variance, w tym przypadku jest ona większa



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

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



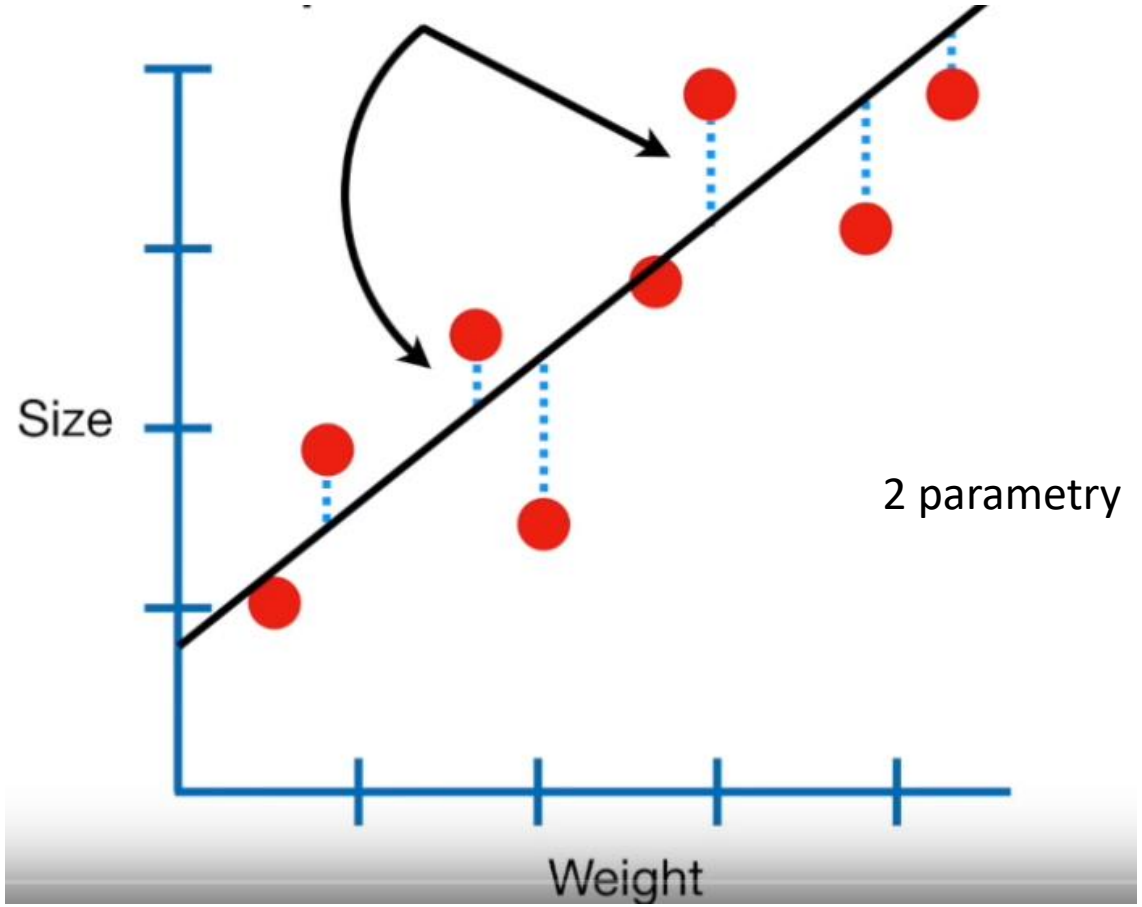


W ML idealny algorytm ma **low bias** i **low variability**

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

regularization
boosting
bagging

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



2 parametry

Znajdujemy linię, która daje:

minimum sum of squared residuals

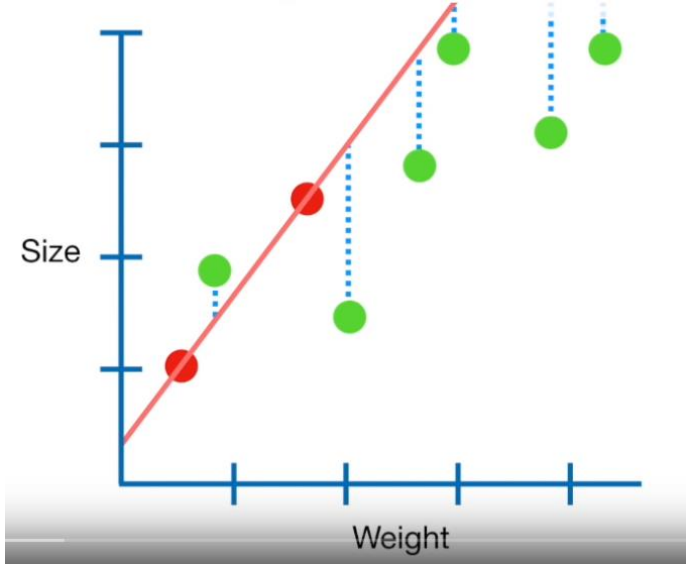
y-axis intercept

Size = 0.9 + 0.75 x **Weight**

slope

Jeśli mamy tylko dwa punkty:

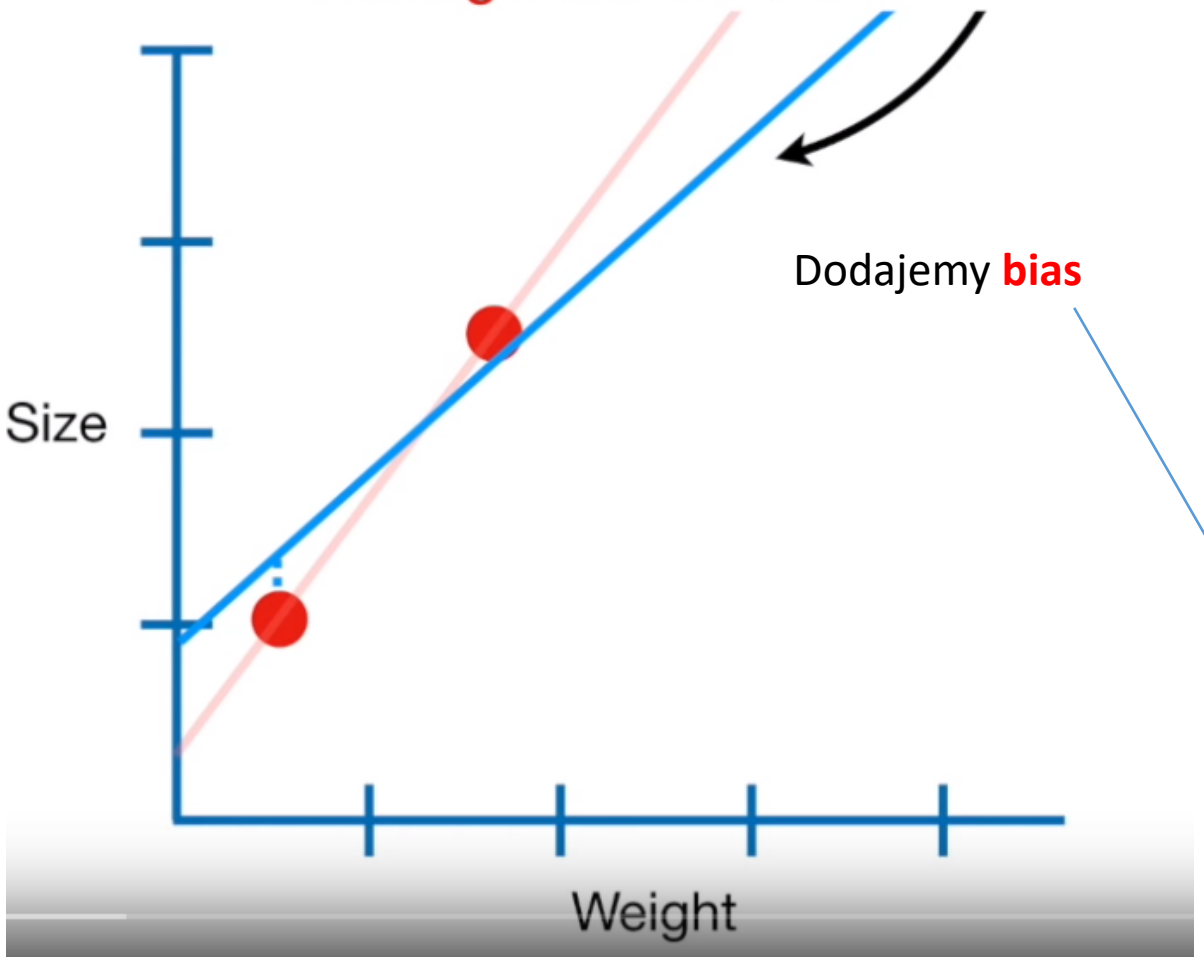
Sum for squared residiuals for train = 0
Sum for square residuals for test LARGE



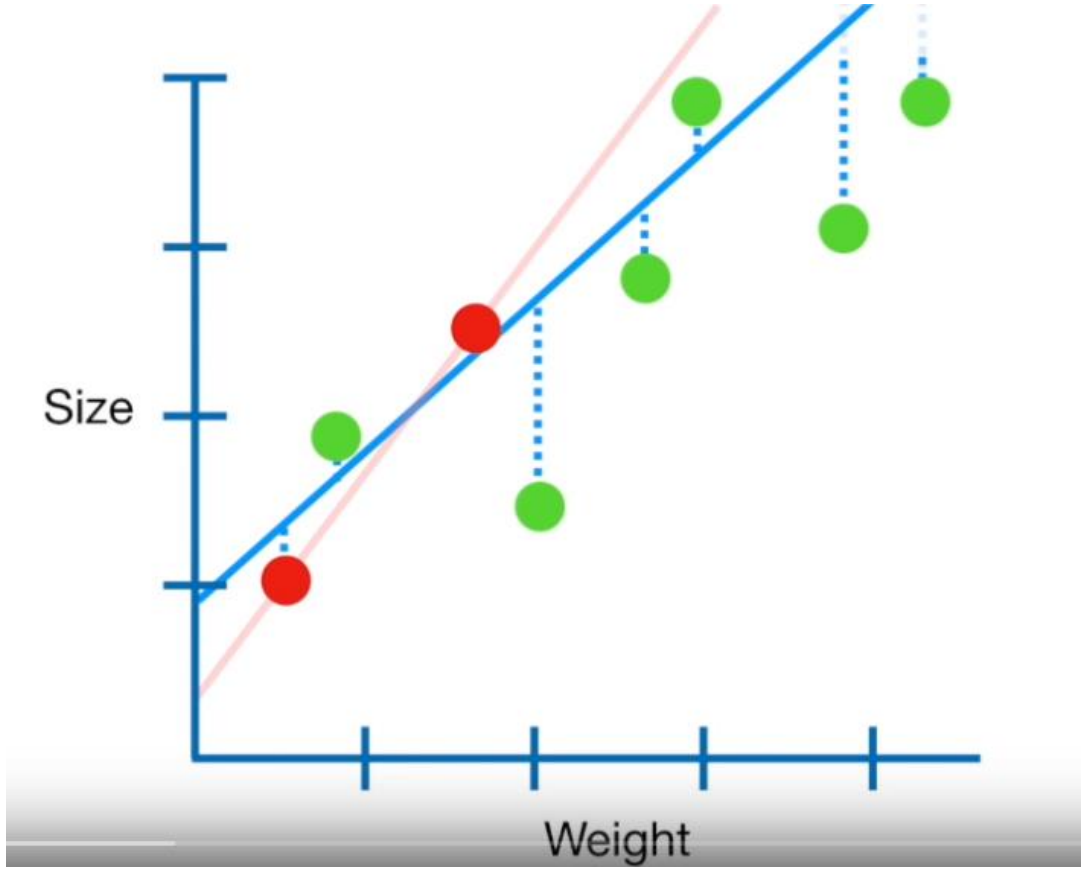
New line has High Variance

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…

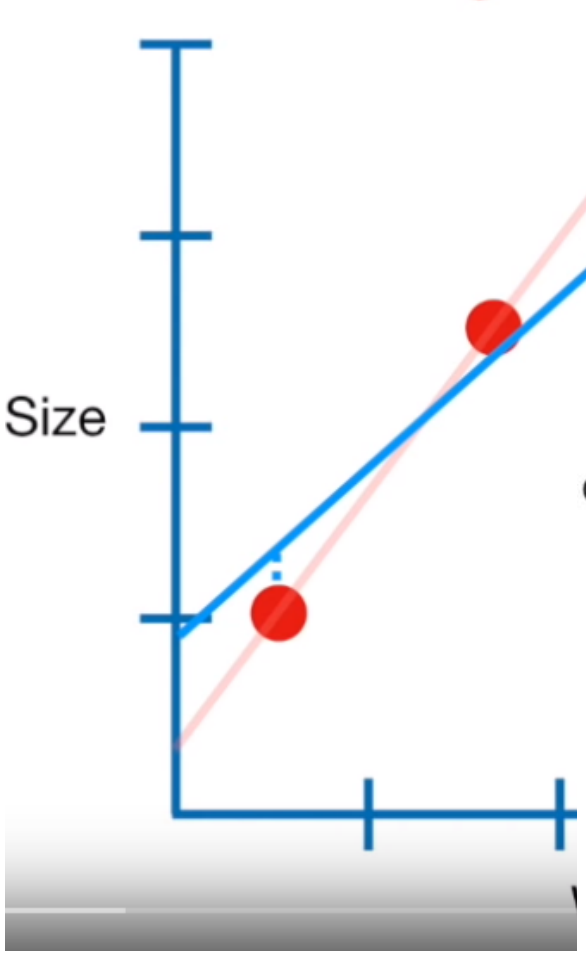


Dodajemy **bias**



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...



Dodajemy **bias**

Ridge Regression

determines values for the parameters in this equation...

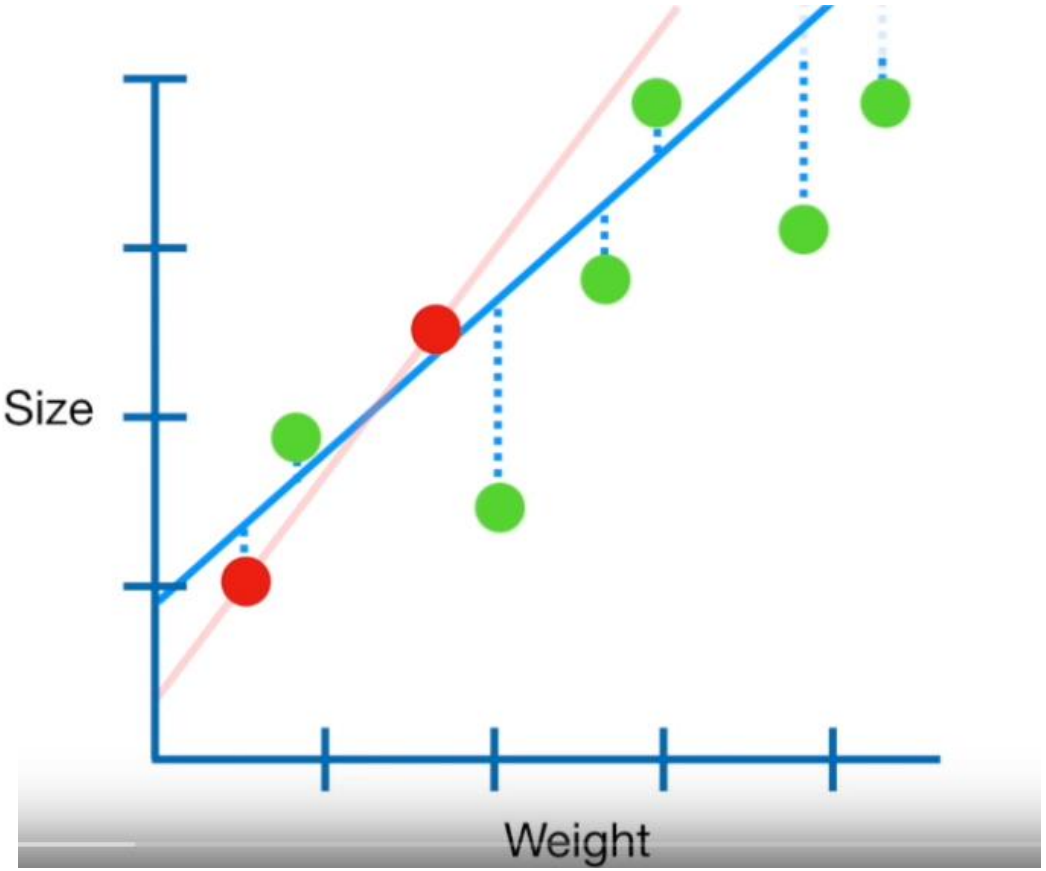
Size = y-axis intercept + slope × **Weight**

...it minimizes...

the sum of the squared residuals

+

λ × the slope²



Zmniejsza się dzięki temu **variance**

Ridge Regression

determines values for the parameters
in this equation…

↓

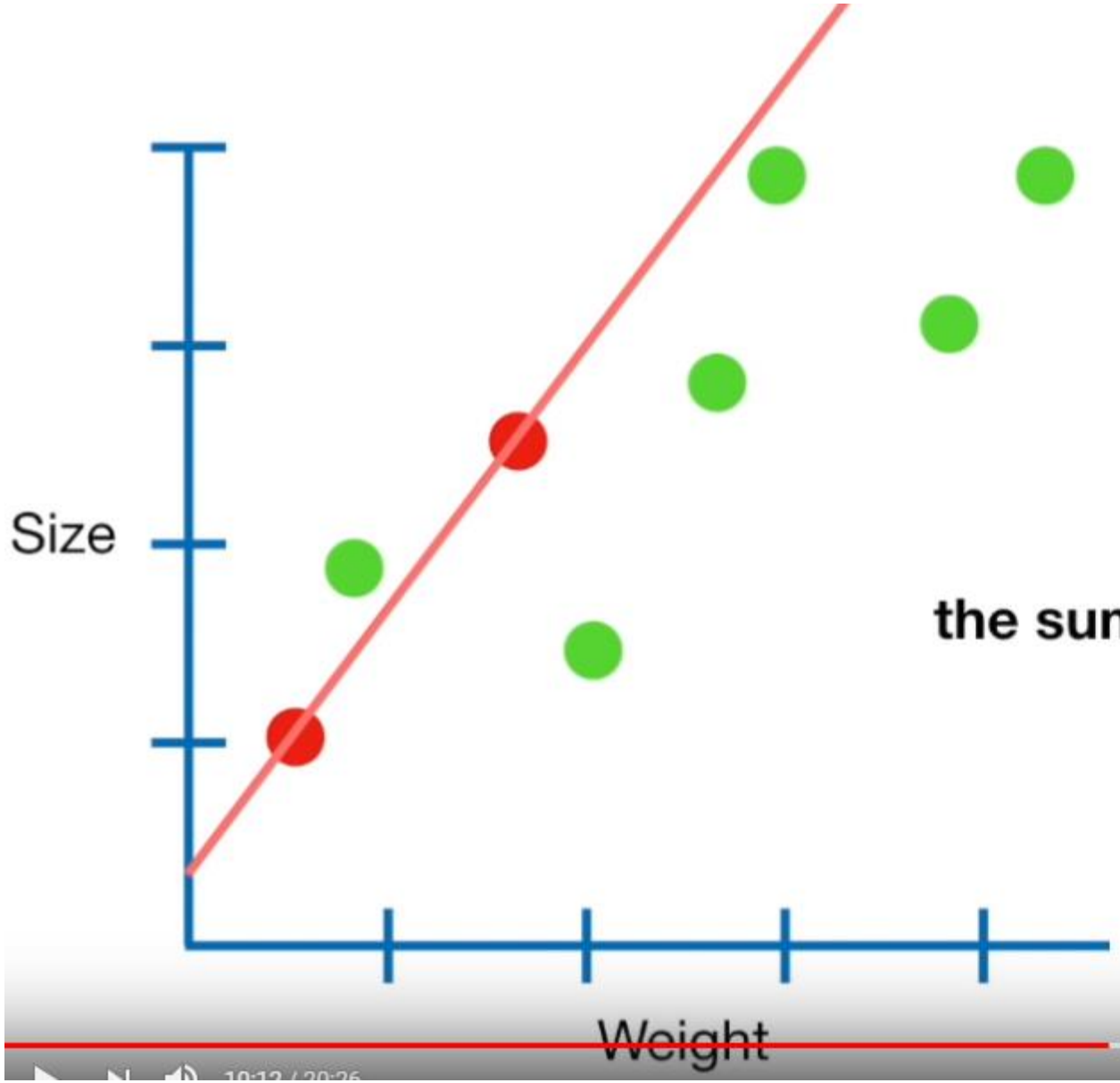
Size = y-axis intercept + slope × **Weight**

…it minimizes…

the sum of the squared residuals

+

λ × the slope² ← dodaje penalty do tradycyjnej Least Squares method



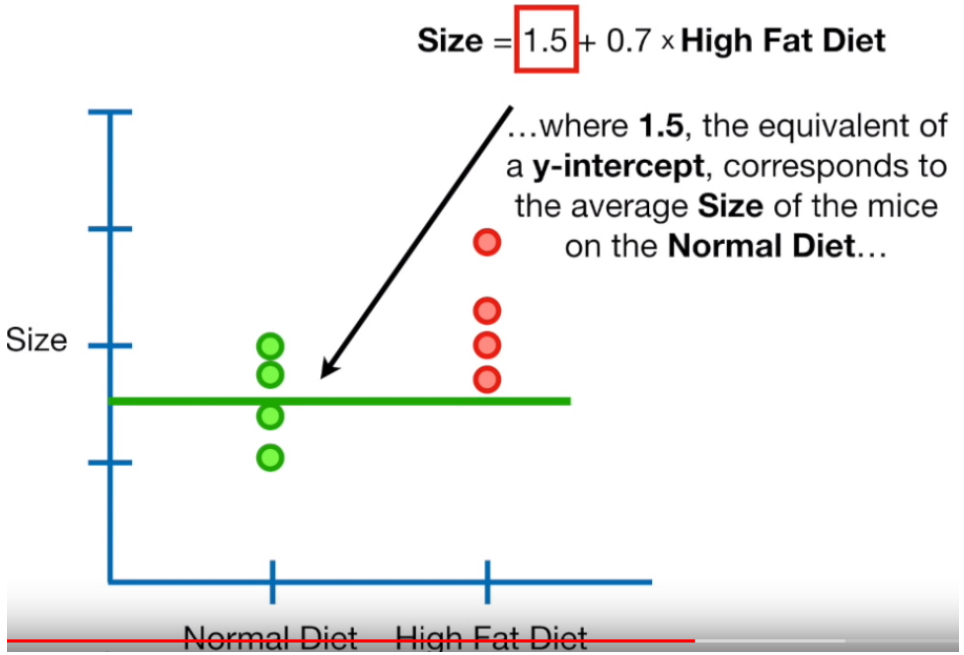
We just try a bunch of values for **λ** and use **Cross Validation**, typically **10-fold Cross Validation**, to determine which one results in the lowest **Variance**.

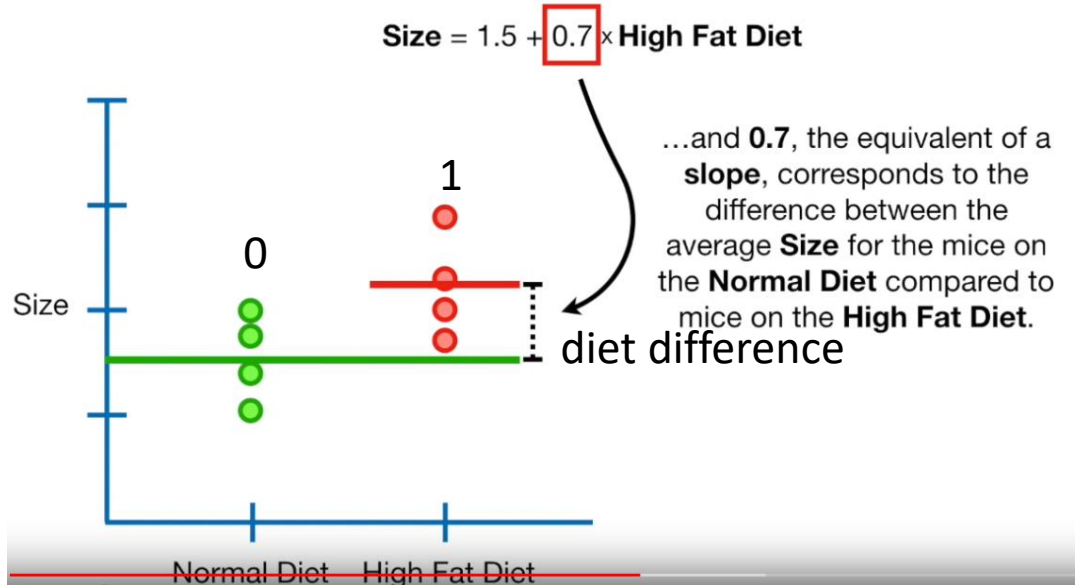
the sum of the squared residuals

+

λ x the slope²

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

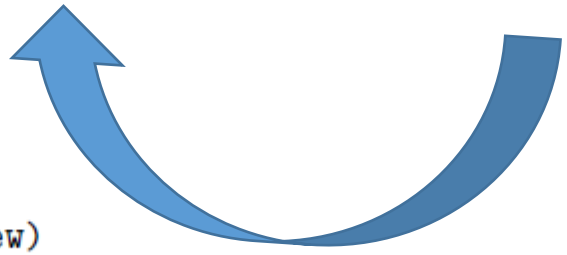
Niektóre algorytmy ML (najstarsze) nie zawierają globalnej automatyzacji

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

Jak pracuje się w ML

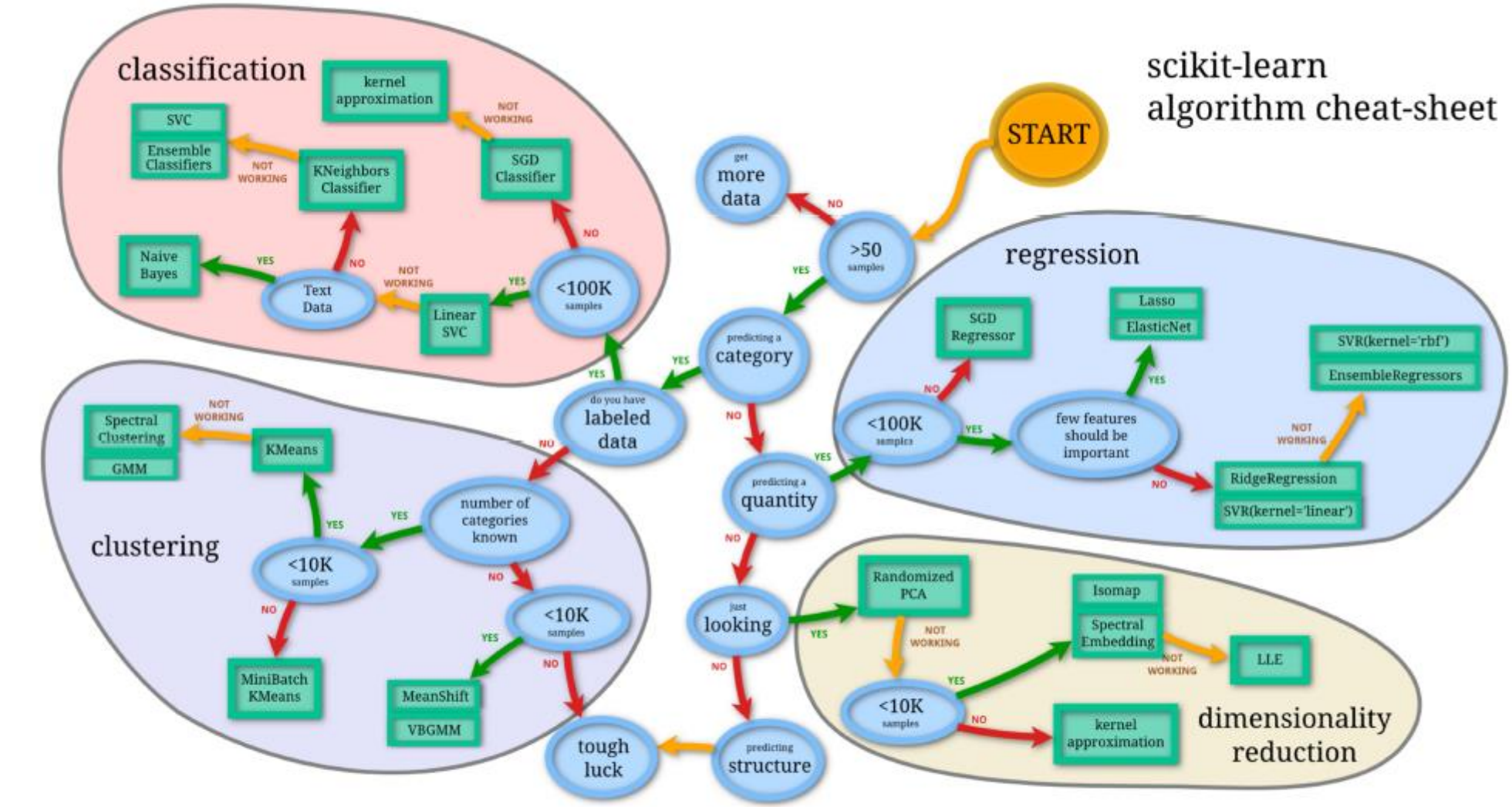
Używa się bibliotek z algorytmami: ***scikit-learn***


```
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 (TA SAMA PROCEDURA !)

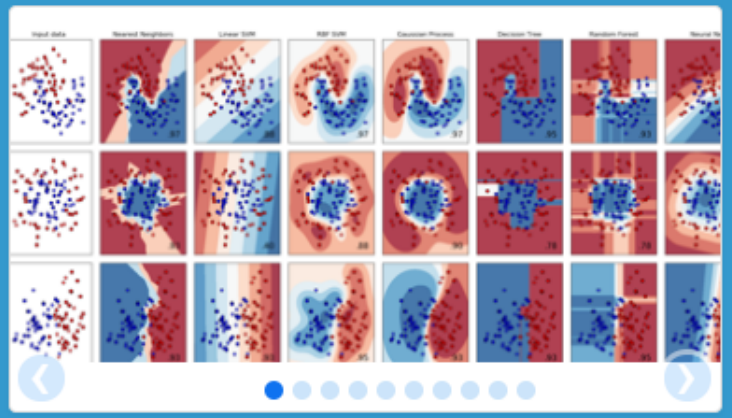
Algorytmy zawierają hyperparametry, które mogą służyć do jego optymalizacji dla danego zagadnienia.


scikit-learn
algorithm cheat-sheet
START
get more data
NO
>50 samples
YES
predicting a category
YES
do you have labeled data
YES
<100K samples
NO
Naive Bayes
YES
Text Data
NO
KNeighbors Classifier
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
kernel approximation
NOT WORKING
SGD Classifier
NO
NOT WORKING
Linear SVC
YES
classification
NO
Spectral Clustering
GMM
NOT WORKING
KMeans
YES
number of categories known
YES
<10K samples
NO
MiniBatch KMeans
NO
<10K samples
YES
MeanShift
VBGMM
NO
clustering
NO
predicting a quantity
YES
<100K samples
NO
SGD Regressor
YES
few features should be important
YES
Lasso
ElasticNet
NO
RidgeRegression
SVR(kernel='linear')
NOT WORKING
SVR(kernel='rbf')
EnsembleRegressors
regression
NO
just looking
YES
Randomized PCA
NOT WORKING
<10K samples
YES
Isomap
Spectral Embedding
NOT WORKING
LLE
NO
kernel approximation
dimensionality reduction
NO
predicting structure
tough luck

scikit learn

Home Installation Documentation Examples

Google Custom Search



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

https://scikit-learn.org/stable/

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:

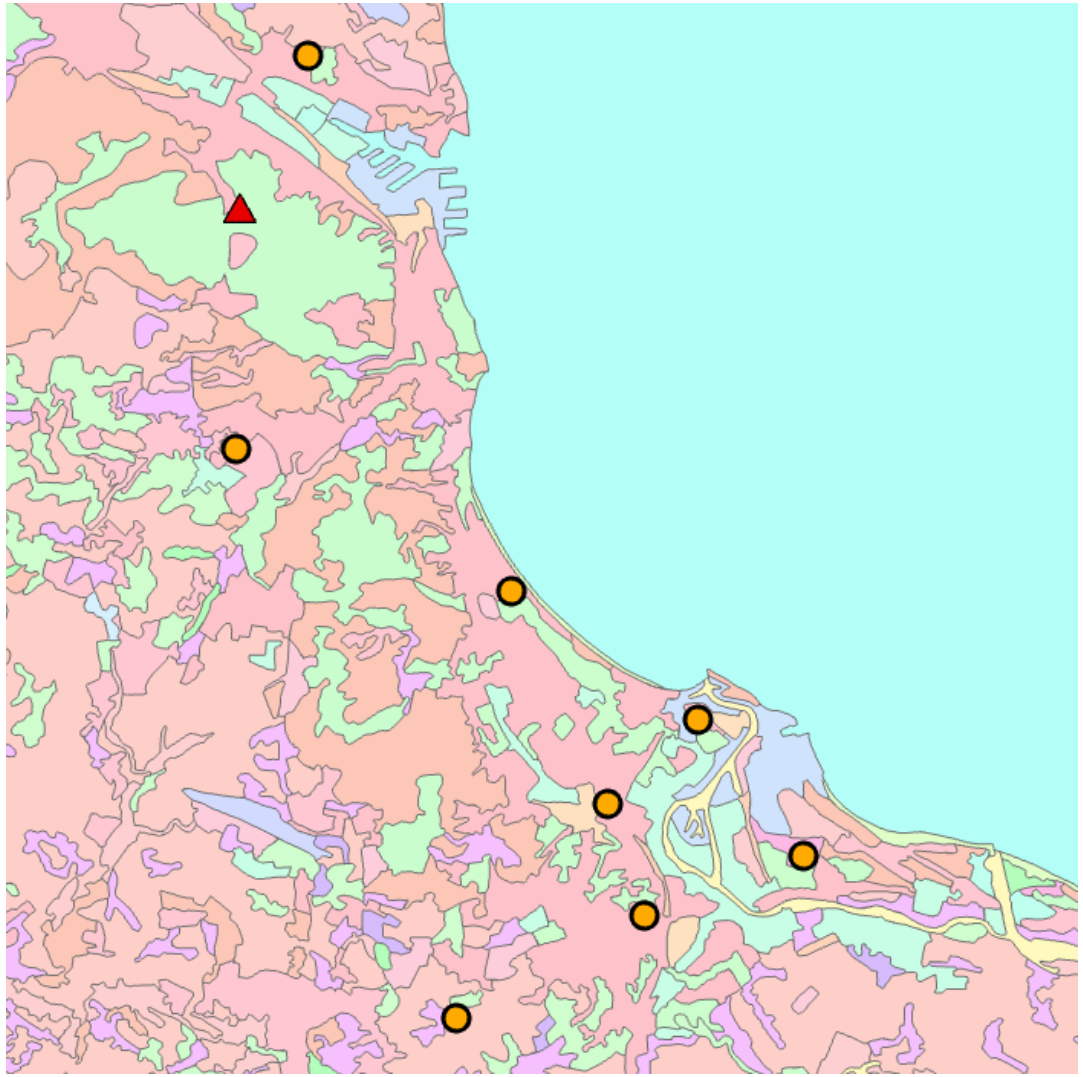
Informative feature – która pozwoli algorytmom ML budowę modelu, który będzie miał wysoką **predictive power**

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]:

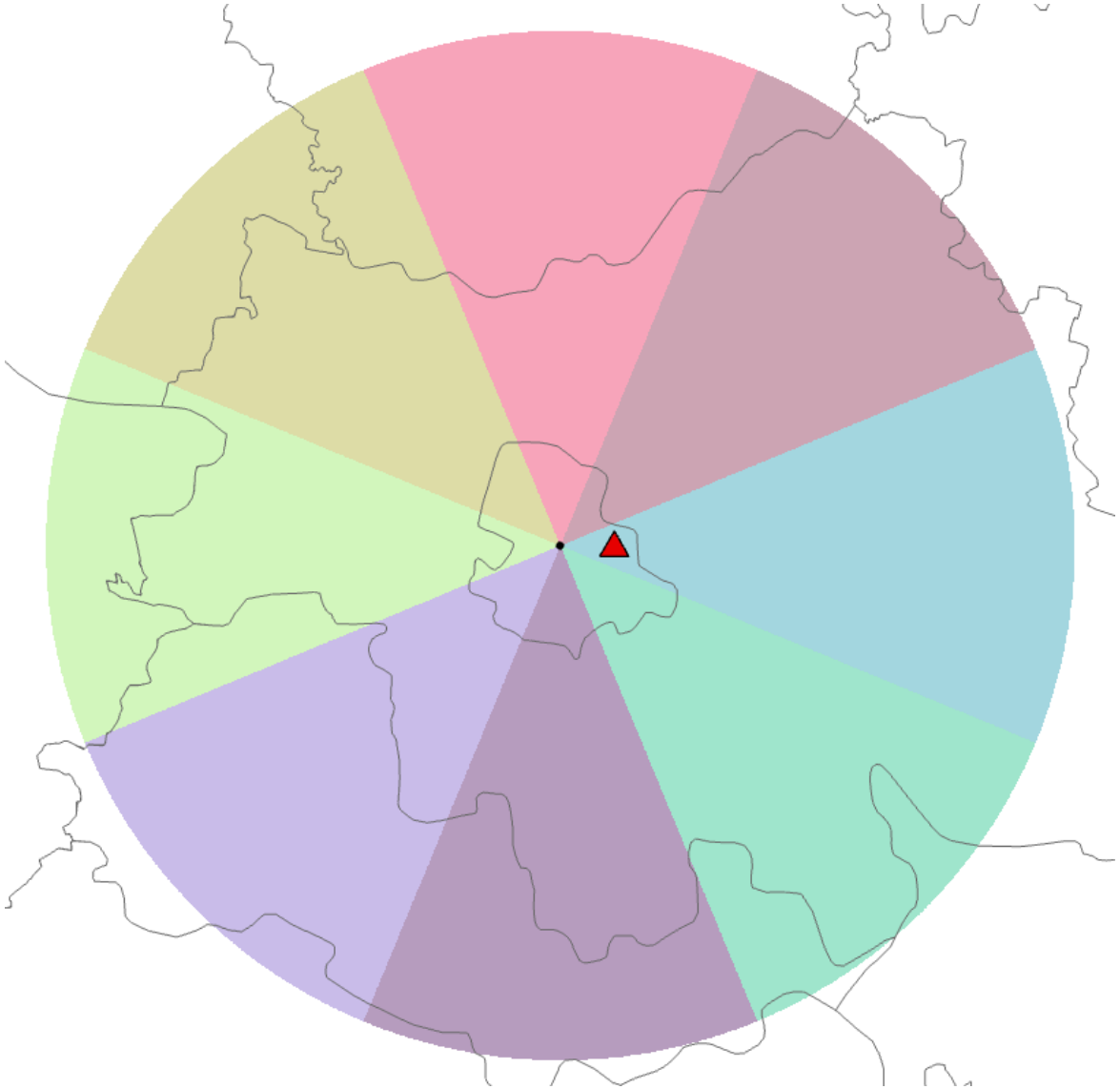
```
Index(['Code', 'Concentration', 'TimeSM', 'T', 'H%', 'Wind_dir', 'Wind_vel',
       'UR200', 'WA200', 'GR200', 'IN200', 'RO200', 'UR600', 'WA600', 'GR600',
       'IN600', 'RO600', 'UR1000', 'WA1000', 'GR1000', 'IN1000', 'RO1000',
       'UR2000', 'WA2000', 'GR2000', 'IN2000', 'RO2000', 'UR4000',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

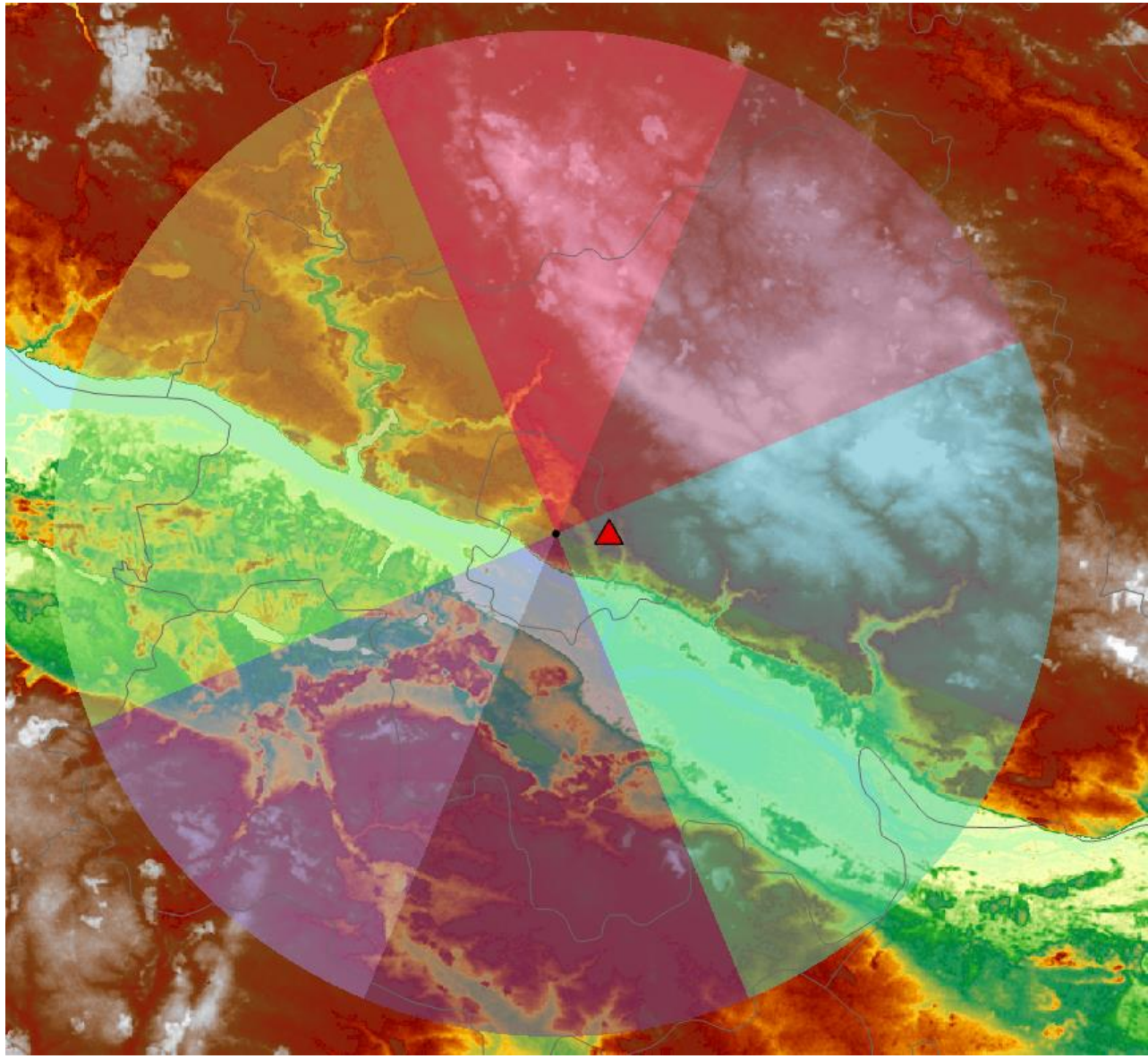
UR200	WA200	GR200	IN200	RO200	UR600	WA600	GR600	IN600	RO600	UR1000	WA1000	GR1000	IN1000	RO1000
62.93	0.0	2.66	20.68	6.74	52.55	0.0	12.26	16.26	13.1	46.9	0.16	17.2	18.07	10.76
62.93	0.0	2.66	20.68	6.74	52.55	0.0	12.26	16.26	13.1	46.9	0.16	17.2	18.07	10.76
62.93	0.0	2.66	20.68	6.74	52.55	0.0	12.26	16.26	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

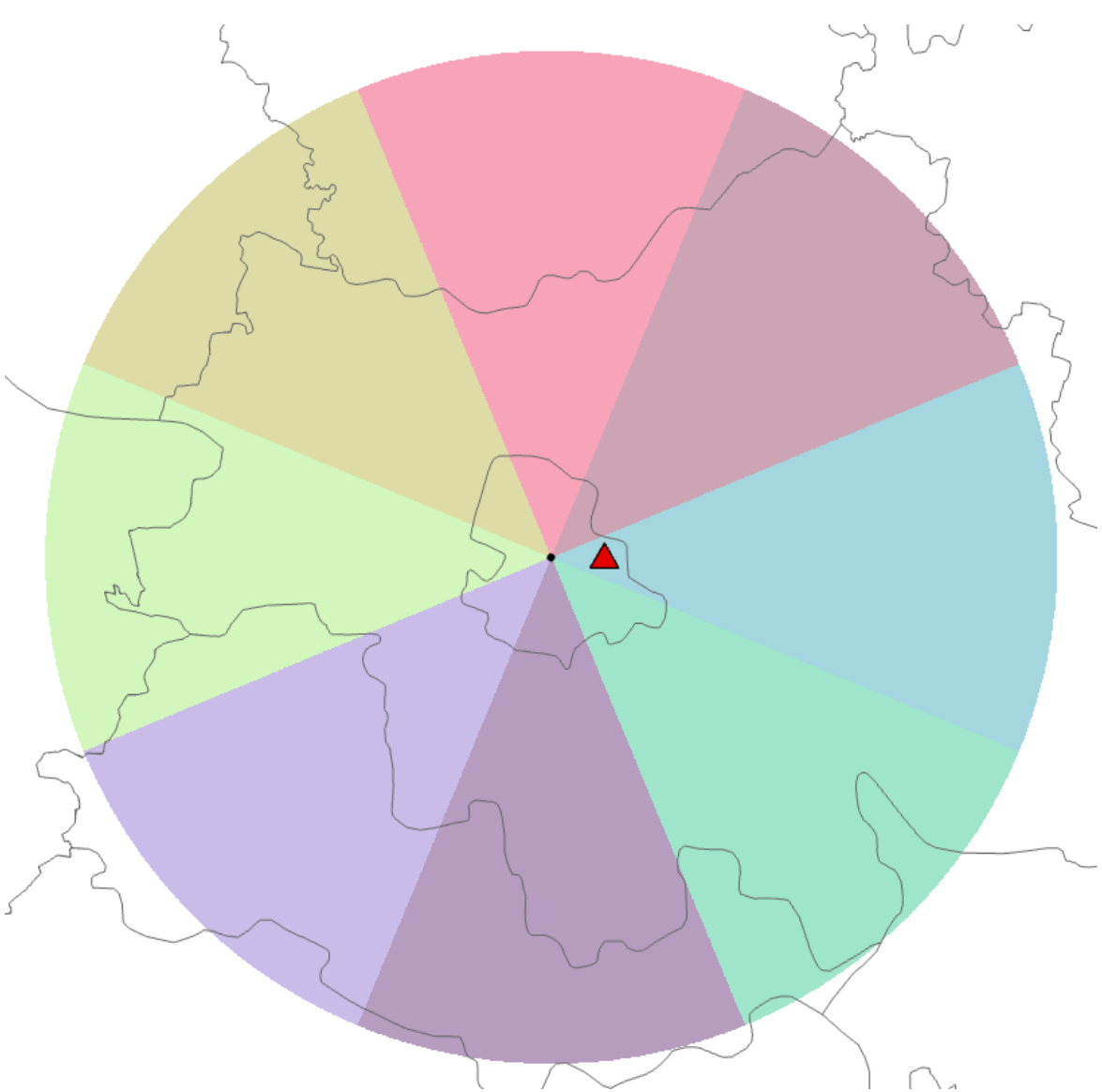
In [8]:
```
polut.columns
```

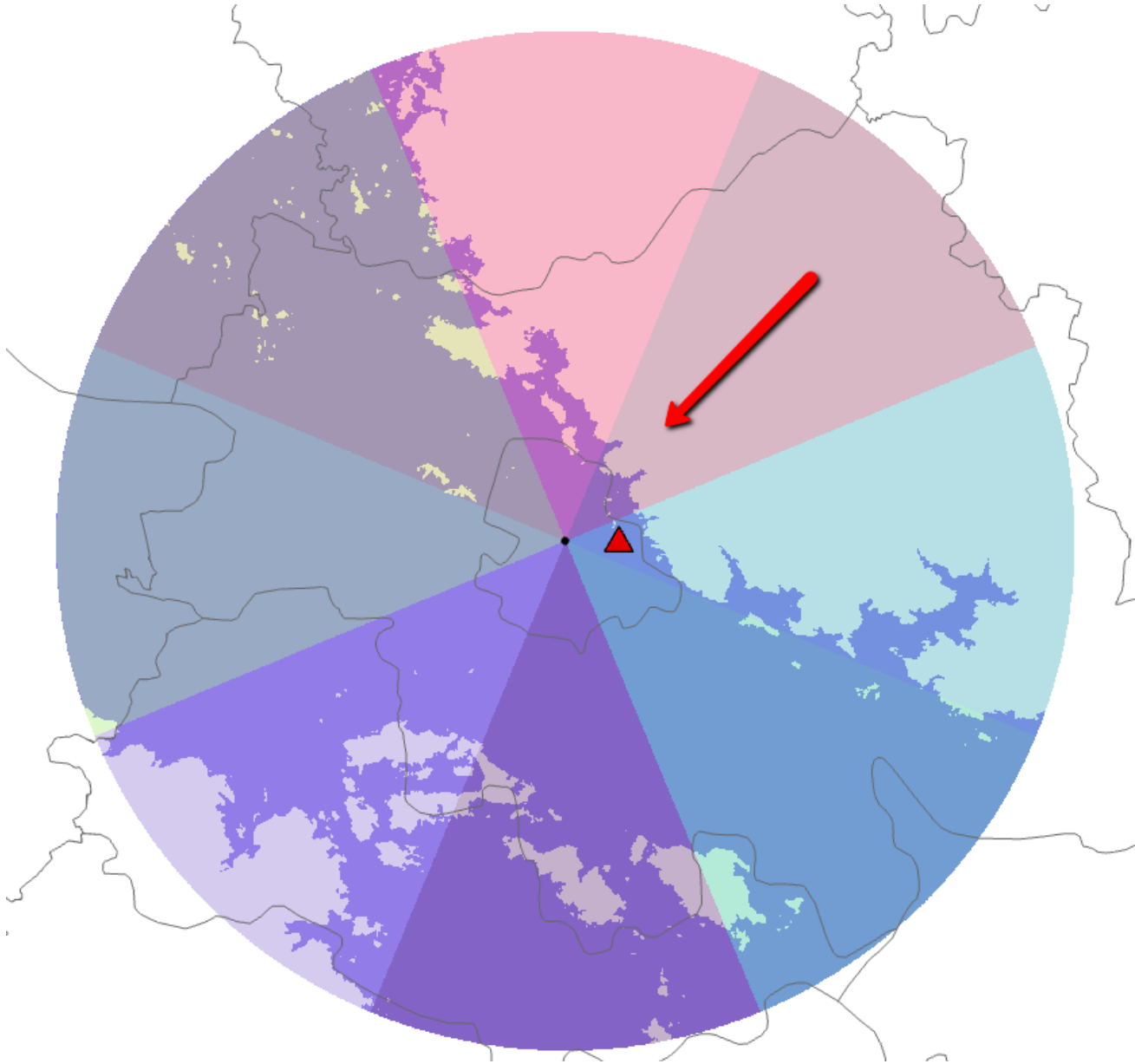
Out[8]:
```
Index(['Code', 'Concentration', 'TimeSM', 'T', 'H%', 'Wind_dir', 'Wind_vel',
       'UR200', 'WA200', 'GR200', 'IN200', 'RO200', 'UR600', 'WA600', 'GR600',
       'IN600', 'RO600', 'UR1000', 'WA1000', 'GR1000', 'IN1000', 'RO1000',
       'UR2000', 'WA2000', 'GR2000', 'IN2000', 'RO2000', 'UR4000', '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 categorial features

Zamiana ciągłych wartości na wiele bins:

Wiek:

Bin0_5
Bin6_10
Bin11-15
Bin15_30
Bin30_50
Bin50_70
Bin70-90

Wiek
22
12

Bin0_5	Bin6_10	Bin11_15	Bin15_30	Bin30_50	Bin50_70	Bin70_90
0	0	0	1	0	0	0
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 rozkadu 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.

Postepowanie 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 tylko2 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

treningowy > model > walidacyjny > decyzja

Ocena działania modelu

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

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

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 ostały sklasyfikowane jako true negative TN, a 12 jako false positive – FP.

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

$$\text{precision} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

$$\text{recall} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Cross-Validation – ocena działania modelu (określanie dowolnej miary)

1. Określamy (ustalamy) hiperpatrametry w modelu.
2. Dzielimy zbiór treningowy na szereg podzbiorów o tym samym rozmiarze (każdy podzbiór nazywa się **fold**)
3. Najczęściej używana jest five-fold cross-validation (losowy podział na 5 folds: $\{F_1, F_2, \ldots, F_5\}$. Każdy F_k, $k = 1, \ldots, 5$ zawiera 20% danych treningowych.
4. Trenujemy 5 modeli w następujący sposób. W pierwszym modelu, f_1, używamy wszystkich obserwacji z folds F_2, F_3, F_4, i F_5 jako zbiór treningowy i F_1 jako zbiór walidacyjny . W drugim modelu, f_2, używamy wszystkich obserwacji z folds F_1, F_3, F_4, i F_5 jako zbiór treningowy i F_1 jako zbiór walidacyjny.
5. Kontynuujemy iteracyjne budowanie modeli i wyznaczamy wartość dla każdego zbioru walidacyjnego od F_1 do F_5.
6. Wyznaczamy średnią wartość z 5 jako wartość końcową.