Podstawy Sztucznej Inteligencji (PSI)

Kwestie formalne

Formuła przedmiotu

- wykład co tydzień, dr Pohl i prof. Dobrowolski
- **7 laboratoriów**, co 2 tygodnie, obecność obowiązkowa
- dopuszczalna **1 nieobecność nieusprawiedliwiona**
- **odrabianie zajęć:**
  - druga grupa (nie musi być ten sam temat)
  - grupa innego laboranta (musi być ten sam temat)
  - konsultacje indywidualne (w wyjątkowych przypadkach)

Zasady oceniania

- **każde laboratorium max 10 punktów**
- **z każdego zestawu trzeba mieć 50% (5/10 punktów)**
- zadania dodatkowe max 2 punkty za każde, nie są uwzględniane do zaliczenia (3.0)
- na oddanie zestawu jest 2 tygodnie, do godziny rozpoczęcia następnych zajęć
- za **zestaw spóźniony** dostaje się min(5, liczba punktów z zestawu)
    - **ale:** jeden zestaw można oddać spóźniony bez konsekwencji (zgłosić przy ocenianiu)
- przesyłanie zadań prawdopodobnie przez Teamsy lub maila

Zadania

- https://github.com/apohllo/sztuczna-inteligencja
- jak zauważysz błąd / literówkę -> PRs welcome

Konfiguracja środowiska

Stos technologiczny

- opcja 1: Google Colab
- opcja 2:
  - Python 3.9
  - Anaconda
  - Jupyter Notebook
  - PyCharm

Środowisko wirtualne

- Python uruchamia się w izolowanym **środowisku wirtualnym (virtual environment, venv)**
- venv'y mają różne rodzaje: bazowy (venv), pipenv, Poetry env, **conda**, …
- **Anaconda** - dystrybucja Pythona do data science
- **Jupyter Notebook** - środowisko do tworzenia interaktywnych notebooków z kodem i tekstem
- **kernel** - "programming language specific process"; dla Pythona to **IPython**
- **ważne combo:** venv + kernel, zawiera wtedy te same paczki;

Konfiguracja venv'a i kernela

- stworzenie venv'a: `conda create -n <env_name> python=3.9`
- aktywacja venv'a: `conda activate <env_name>`
- instalacja IPython'a: `conda install -c anaconda ipykernel`
- instalacja kernela: `python -m ipykernel install --user --name=<env_name>`
- włączenie Jupyter Notebooka: `jupyter notebook`
- aktywacja kernela: GUI Jupyter Notebooka (Kernel -> Change kernel)

Wstęp do uczenia maszynowego

Uczenie nadzorowane

- posiadamy **etykietę (label)**, opisującą przewidywaną cechę, dla każdego przykładu
- **klasyfikacja:**
  - dyskretny, skończony, nieuporządkowany zbiór **klas (classes)**
  - np. zdrowy/chory, gatunek ptaka na zdjęciu, text sentiment (negatywny/pozytywny)
- **regresja:**
  - wartość ciągła - **regression target**
  - np. klikalność reklamy (CTR), wartość domu, rozpuszczalność molekuły

Dane tabelaryczne

- wiersze to **próbki / przykłady (samples)**
- kolumny to **cechy / zmienne (features / variables)**
- liczba cech = **wymiarowość (dimensionality)**
- cechy mają **typy:**
  - binarne (logiczne)
  - kategoryczne (skończone, nieuporządkowane)
  - numeryczne (dyskretne lub ciągłe)

	A	B	C	D	E
1	Region	Market	Business Segment	Sales Period	Sales Amount
2	Europe	France	Accessories	Jan	1,706
3	Europe	France	Accessories	Feb	3,767
4	Europe	France	Accessories	Mar	1,219
5	Europe	France	Accessories	Apr	3,091
6	Europe	France	Accessories	May	7,057
7	Europe	France	Accessories	Jul	5,930
8	Europe	France	Accessories	Aug	9,628
9	Europe	France	Accessories	Sep	4,279
10	Europe	France	Accessories	Oct	2,504
11	Europe	France	Accessories	Nov	7,493
12	Europe	France	Accessories	Dec	2,268
13	Europe	France	Bikes	Jan	64,895
14	Europe	France	Bikes	Feb	510,102
15	Europe	France	Bikes	Mar	128,806
16	Europe	France	Bikes	Apr	81,301
17	Europe	France	Bikes	May	619,594

Pojęcia podstawowe

- **algorytmy** - konkretne metody do klasyfikacji lub regresji
- algorytm to tak naprawdę **funkcja**, która dostaje dane i zwraca wynik, czyli klasę lub wartość regresji
- **trening** - nauka **parametrów** funkcji, aby dostosować ją do naszego **zbioru danych**
- taki algorytm modeluje rzeczywistość, więc jest **modelem**
- model zawiera pewne założenia "z góry", sterujące jego zachowaniem, czyli **hiperparametry (hyperparameters)**

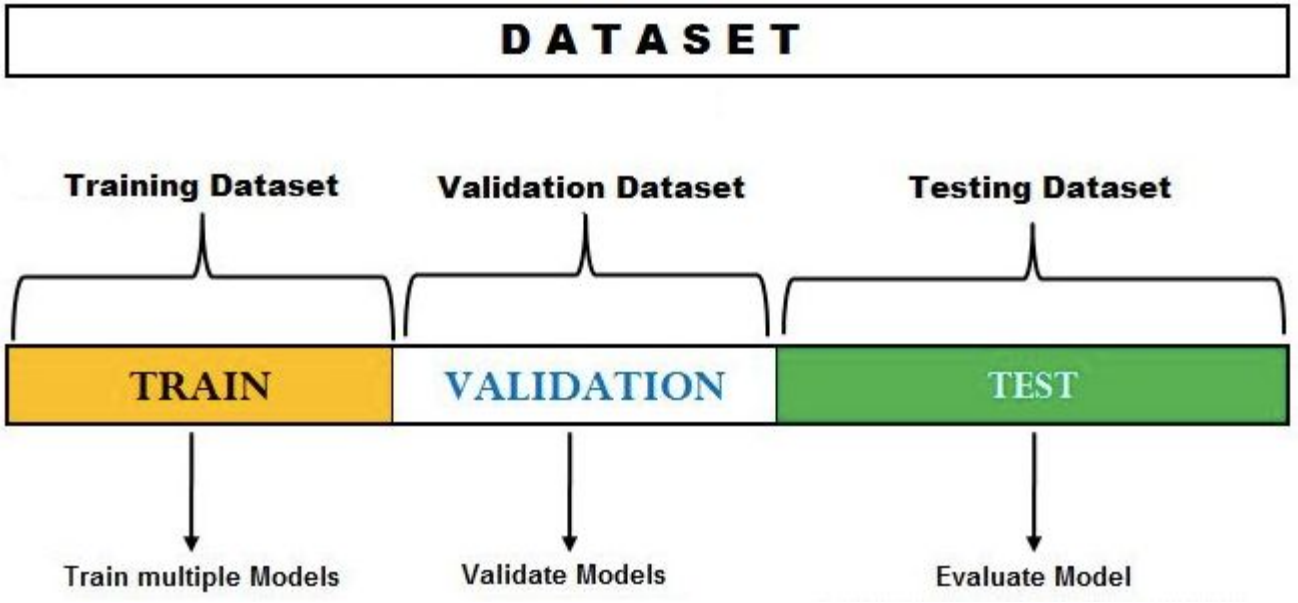
Ocena modelu

Ocena modelu

- **jakość** wytrenowanego modelu jest niejednoznaczna
- wiele **metryk (metrics)** jakości
- **zawsze** trzeba sprawdzać jakość modelu na danych, których nie widział w trakcie treningu!
  - **zbiór treningowy (training set)** - na nim uczymy model
  - **zbiór walidacyjny (validation set)** - sprawdzamy na nim na bieżąco kolejne modele, np. do wyboru hiperparametrów
  - **zbiór testowy (test set)** - sprawdzamy na nim ostateczny model, raportujemy metryki na nim

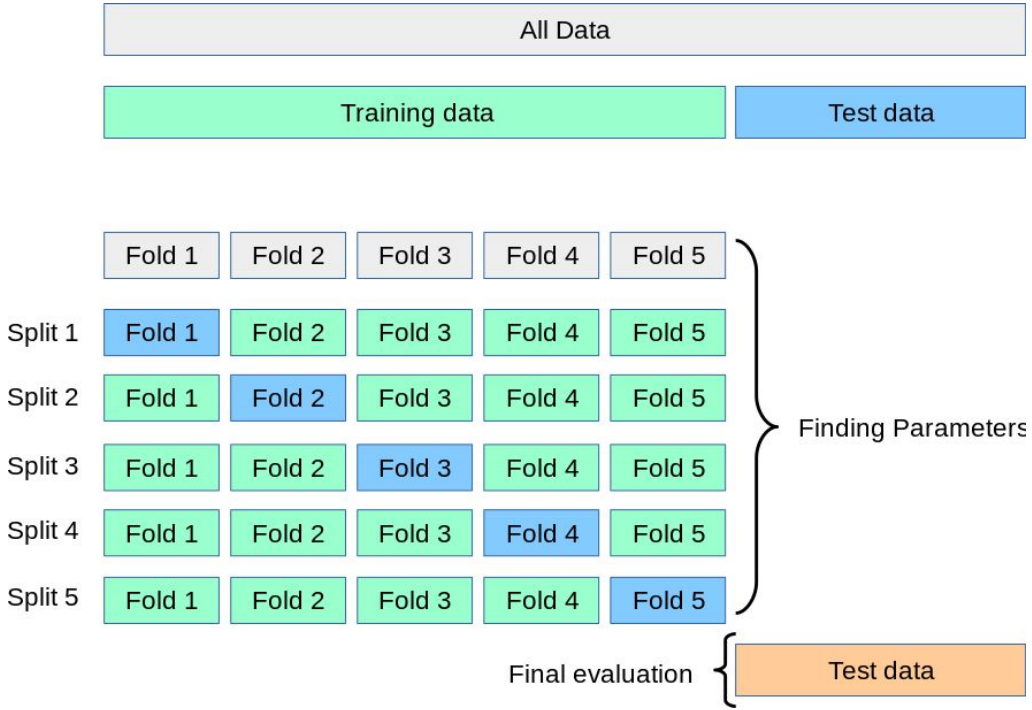
Metoda holdout

- tniemy zbiór na 3 kawałki
- typowe proporcje:
  - 60-20-20%
  - 80-10-10%



Walidacja skrośna (cross-validation)

- tniemy dane treningowe na k foldów, po kolei każdy jest zbiorem walidacyjnym, a reszta treningowym
- daje k wyników, które uśredniamy
- bardziej precyzyjne, ale większy koszt obliczeniowy
- typowe wartości k: 5, 10



Predykcje w klasyfikacji binarnej

- model + klasyfikacja binarna: 0/1, chory/zdrowy, odmówić/udzielić kredytu, …
- możliwe sytuacje:
  - **true positive (TP)** - predykcja 1, prawda 1
  - **true negative (TN)** - predykcja 0, prawda 0
  - **false positive (FP)** - predykcja 1, prawda 0
  - **false negative (FN)** - predykcja 0, prawda 1
- **chcemy:** jak najwięcej TP i TN, jak najmniej FP i FN

Macierz pomyłek (confusion matrix)

- pozwala obliczyć inne metryki
- **uwaga:** trzeba ogarnąć, gdzie są predykcje, a gdzie wartości prawdziwe

$$P = TP + FN$$

$$N = TN + FP$$

	Actual 1	Actual 0
Predicted 1	True Positive (TP)	False Positive (FP)
Predicted 0	False Negative (FN)	True Negative (TN)

Celność (accuracy)

- podstawowa miara jakości
- wyznacza, jak często model przewiduje prawdziwą klasę
- **problem:** nie działa dobrze, kiedy liczność klas jest bardzo różna - **klasyfikacja niezbalansowana (imbalanced classification)**

$$Accuracy = \frac{TP + TN}{P + N}$$

	Actual 1	Actual 0
Predicted 1	True Positive (TP)	False Positive (FP)
Predicted 0	False Negative (FN)	True Negative (TN)

Predykcje w regresji

- **błąd średniokwadratowy (mean squared error, MSE):**
  - często używany podczas treningu
  - różniczkowalny
  - mocno penalizuje duże błędy (nie zawsze pożądane)

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

- **błąd bezwzględny (mean absolute error, MAE):**
  - często używany podczas ewaluacji
  - penalizuje błąd wprost proporcjonalnie do jego wielkości

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Generalizacja i dostrajanie modeli

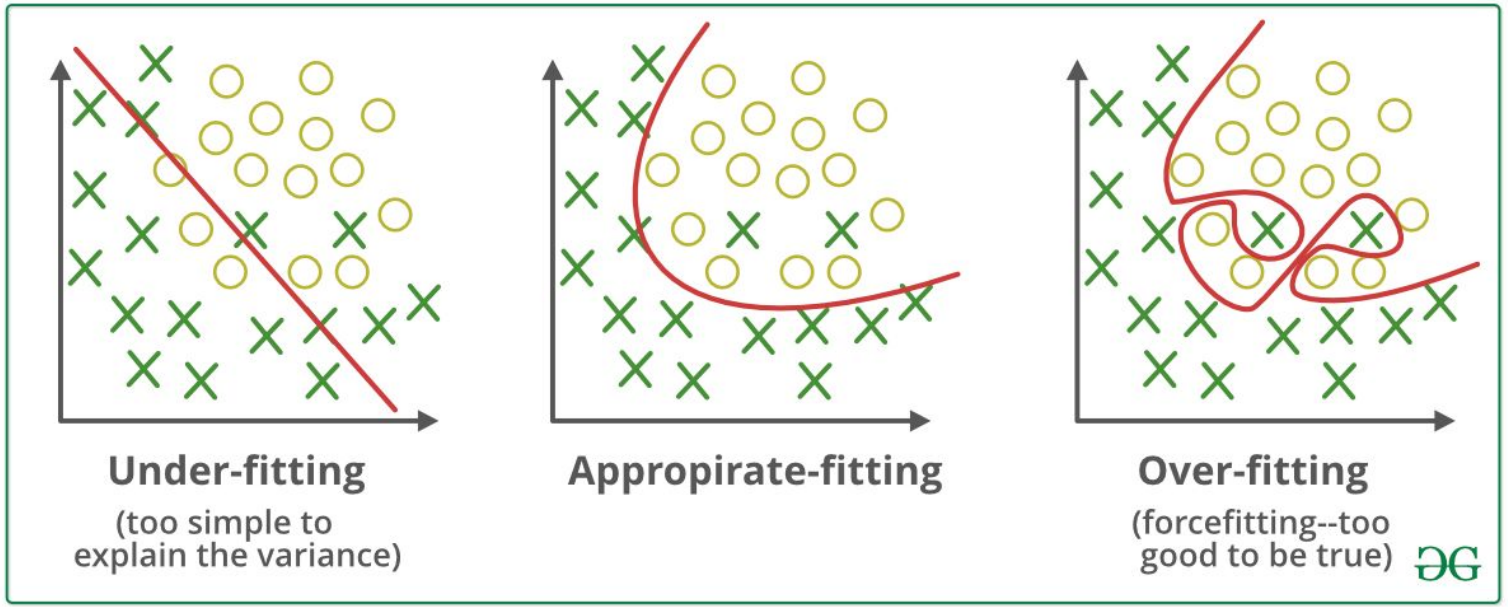
Co chcemy osiągnąć?

- **chcemy:** wysokiego wyniku na zbiorach treningowym oraz testowym
- **w praktyce:** testowy niższy od treningowego
- chcemy modelu, który się dobrze **generalizuje (generalization)**
- **pojemność (capacity)** - miara, jak bardzo złożone relacje w danych potrafi zrozumieć model
- dobra generalizacja wymaga odpowiedniej pojemności

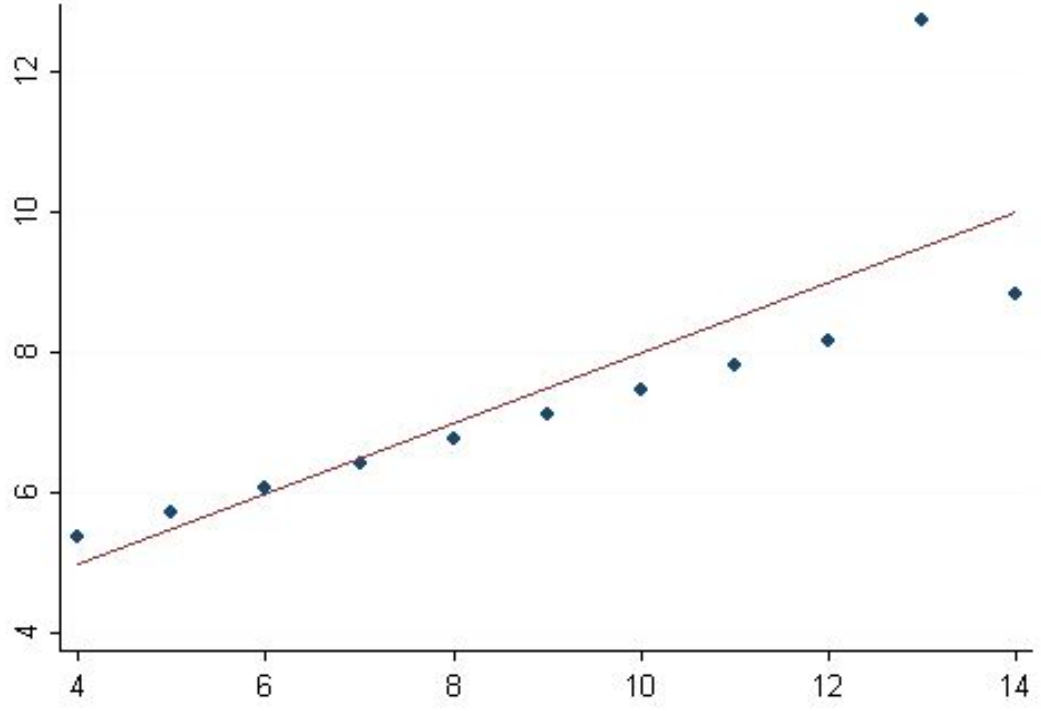
Underfitting i overfitting

- słabe wyniki treningowy i testowy
- zbyt prosty model

- dużo wyższy wynik treningowy od testowego
- zbyt złożony model

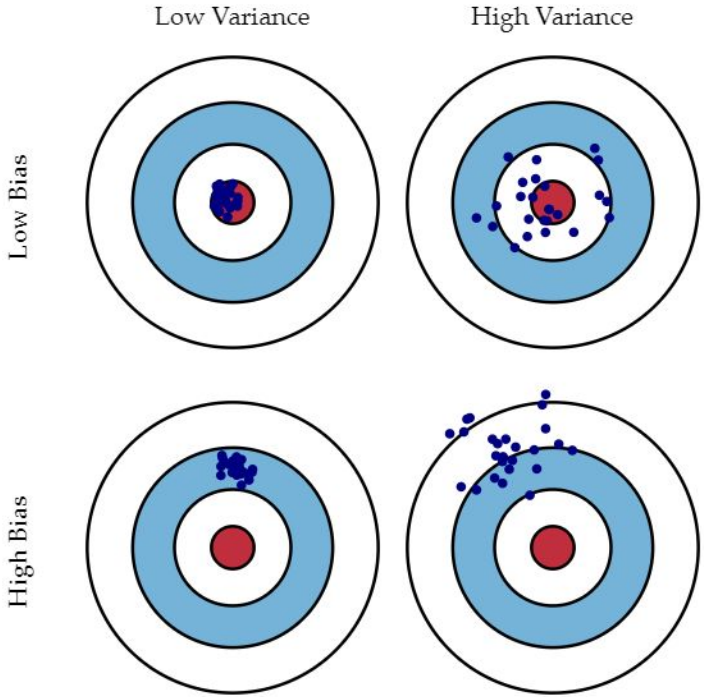


Przykład - overfitting w regresji liniowej



Błąd modelu - bias i variance

- **bias** - błąd wynikający z nieprawdziwych założeń modelu
- **variance** - błąd wynikający z nauki losowych wahań w danych
- **chcemy:** niski bias (dokładny model) i niskiej wariancji (model odporny na szum)
- **bias-variance tradeoff** - jedno rośnie, drugie maleje



Regularyzacja

- **regularyzacja (regularization)** - zmniejszanie pojemności modelu
- zmniejsza wariancję błędu, zwiększa bias - **zmniejsza overfitting**
- "wygładza" predykcje modelu, zmniejsza czułość
- najważniejsza technika to **karanie dużych wartości parametrów**, np. dodawanie wielkości wag jako kary dla modelu
- moc regularyzacji to typowo najważniejszy **hiperparametr (hyperparameter)** modeli

Hiperparametry (hyperparameters)

- czynione z góry założenia co do modelu, parametry narzucane przed treningiem, a nie uczone z danych
- **przykłady:** siła regularyzacji, liczba drzew w lesie losowym
- dobieranie wartości:
  - **grid search** - sprawdzamy wszystkie możliwe kombinacje
  - **random search** - sprawdzamy losowe N kombinacji
- trzeba mieć pojęcie o sensownych zakresach wartości - za to nam płacą

Regresja liniowa

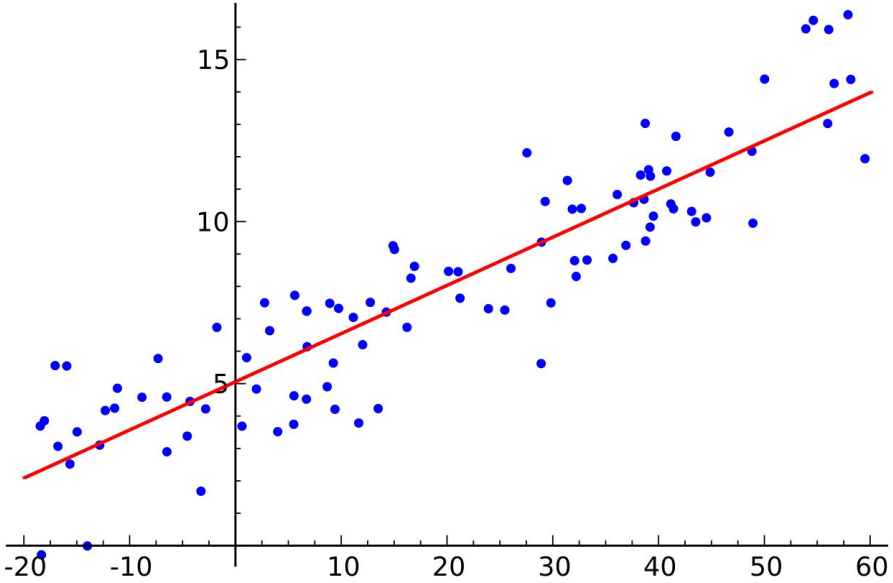
Model

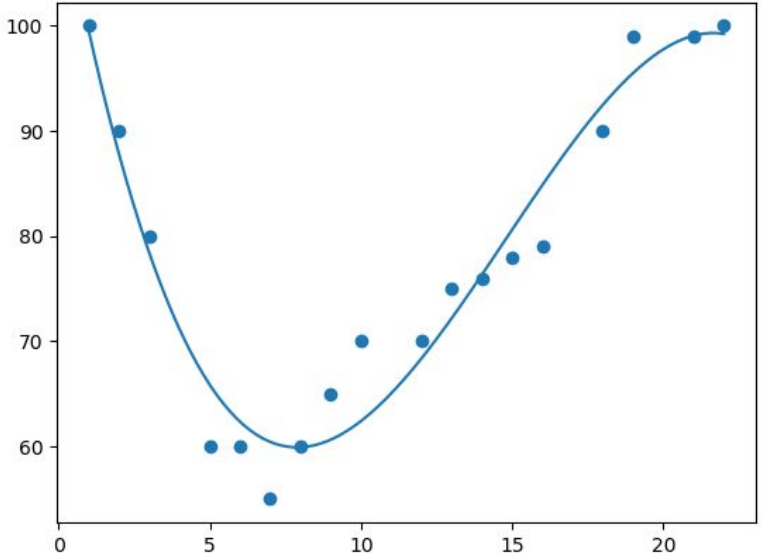
- model to **kombinacja liniowa** współczynników `w` i wektora cech `x`:

$$\hat{y} = w_0 1 + w_1 x_1 + ... + w_d x_d = x^T w$$

- uczymy się **wektora współczynników (wag) `w`**
- **bias**, punkt przecięcia ze środkiem układu współrzędnych, to sztuczna zmienna z samymi 1
- wbrew nazwie **nie musi** być linią prostą / płaszczyzną / hiperpłaszczyzną!

Przykład graficzny



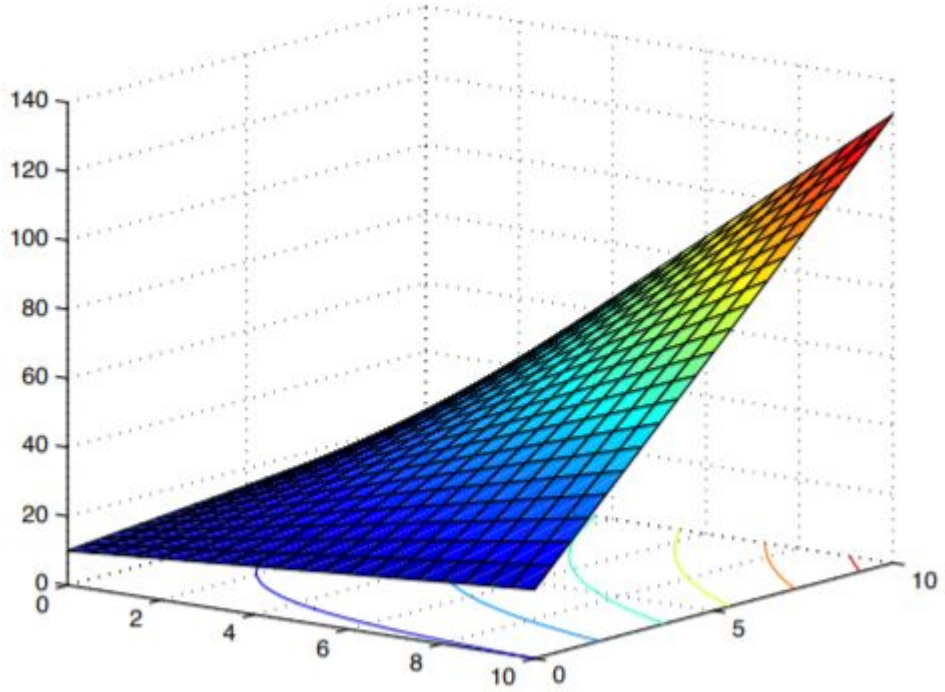


Przykład graficzny 2

$$y = 10 + x_1 + x_2 + x_1 x_2 = w \cdot x$$

$$x = [1, x_1, x_2, x_1 x_2]$$

$$w = [10, 1, 1, 1]$$



Obliczanie współczynników `w`

- trzeba zminimalizować sumę kwadratów błędów

$$\arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

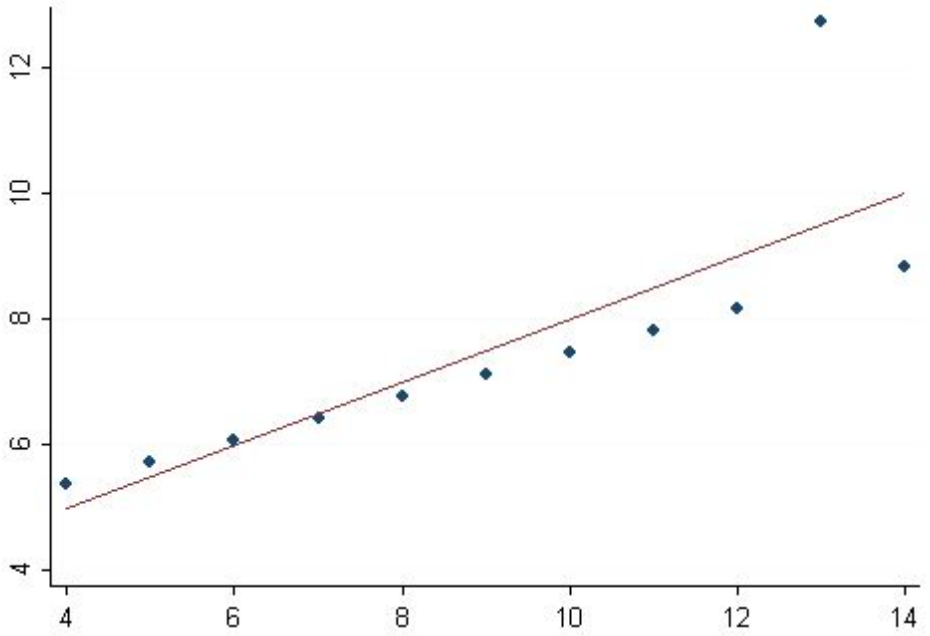
- opcja 1:
  - **analitycznie**, wyprowadzając wzór; nie zawsze się da, ale w regresji liniowej tak
  - w praktyce wykorzystuje **Singular Value Decomposition (SVD)**
- opcja 2:
  - numerycznie zminimalizować funkcję kosztu, typowo **spadkiem wzdłuż gradientu (gradient descent)**
  - używane w sieciach neuronowych

Funkcja kosztu

- wykorzystuje się **sumę kwadratów błędów (Sum of Squared Errors, SSE):**

$$C(X) = \sum_{i=1}^{N} (y - \hat{y})^2$$

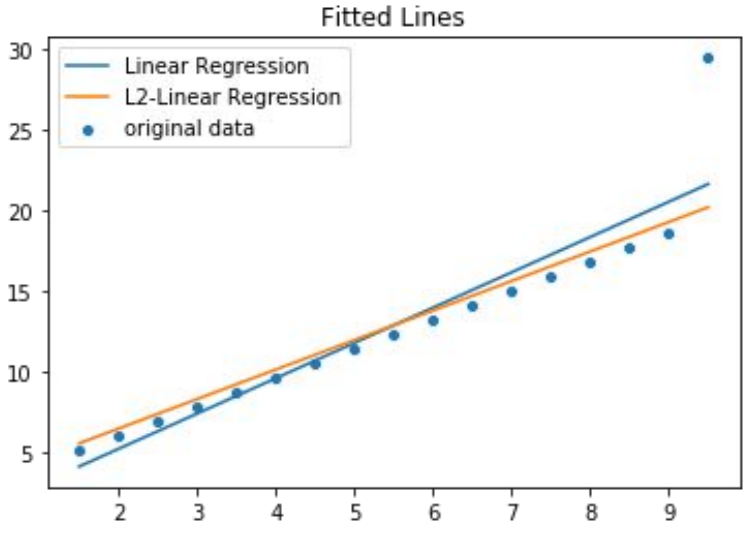
- **zalety:** proste, różniczkowalne
- **wady:** bardzo zwraca uwagę na **punkty odstające (outliers)**, przez co łatwo przeucza



Regularyzacja L2

$$C(w) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{N} w_i^2 = ||y - \hat{y}||_2^2 + \lambda ||w||_2^2$$

- dodajemy do funkcji kosztu sumę kwadratów wag (metryka L2) - formalnie to **ridge regression**
- **penalizuje duże wagi**, które oznaczają zwykle overfitting
- współczynnik regularyzacji λ to **najważniejszy hiperparametr** regresji liniowej



Regularyzacja L1

$$C(w) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{N} |w| = ||y - \hat{y}||_2^2 + \lambda ||w||_1$$

- dodajemy do funkcji kosztu sumę wartości bezwzględnych wag (metryka L1) - formalnie to **LASSO regression**
- penalizuje duże wagi, ale szczególnie **premiuje wagi równe dokładnie 0**
- waga 0 = **selekcja cech (feature selection)**
- **nieróżniczkowalne** - wymaga odpowiedniego solwera
- drugi po L2, albo najważniejszy hiperparametr regresji liniowej

Regularyzacja ElasticNet

$$C(w) = ||y - \hat{y}||_2^2 + \lambda||w||_1 + \gamma||w||_2^2$$

- regularyzacja L1 oraz L2 naraz
- wymaga specjalnego solwera
- **typowo najlepsze wyniki**
- więcej hiperparametrów

Regresja liniowa - zalety i wady

- **zalety:**
  - prostota i interpretowalność
  - szybkość i skalowalność
  - mało hiperparametrów, łatwo dostroić
- **wady:**
  - często zbyt prosty model

Regresja logistyczna

Model

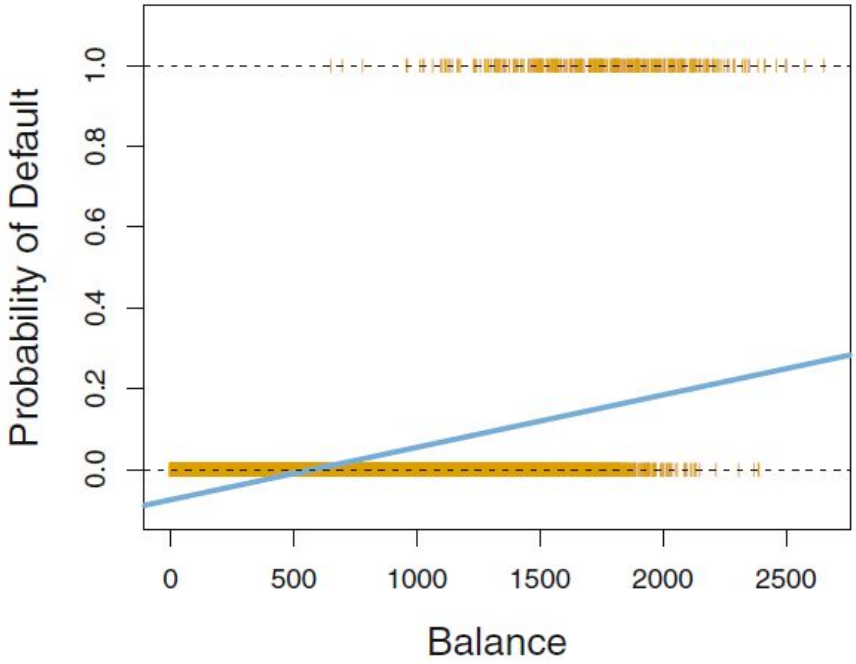
- model to przekształcenie regresji liniowej z użyciem **funkcji logistycznej (sigmoidy):**

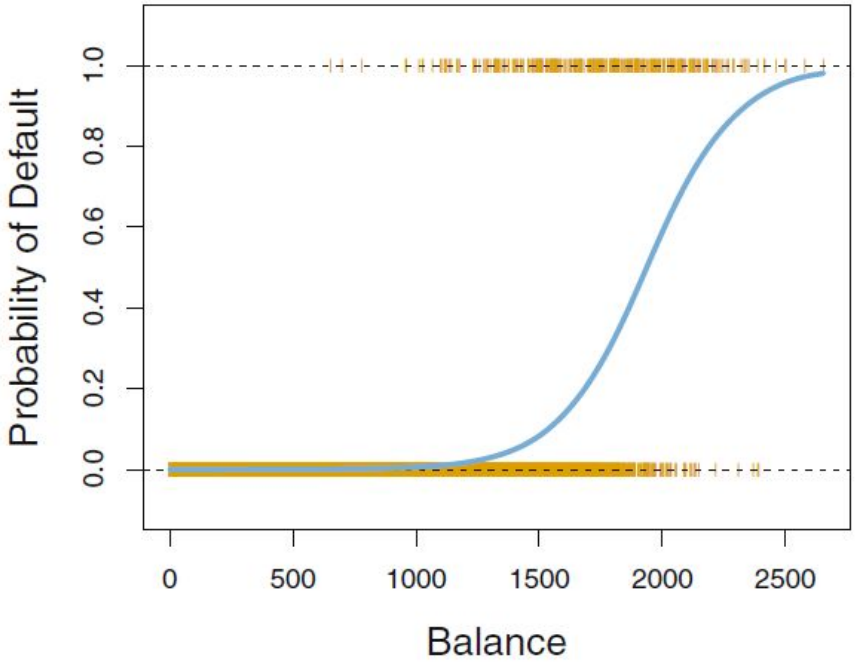
$$\hat{y} = \sigma(Xw) = \frac{1}{1 + \exp(-Xw)} = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \ldots + w_d x_d)}}$$

- to przekształcenie zamienia zakres wartości z (-∞, +∞) na [0, 1], czyli mamy **prawdopodobieństwo klasy pozytywnej**

$$\hat{y} = p(x) = P(y = 1|x)$$

Regresja liniowa vs logistyczna





Obliczanie współczynników `w`

- **nie ma wzoru analitycznego** - trzeba obliczać numerycznie
- metoda największej wiarygodności **(maximum likelihood estimation, MLE)**
- **funkcja wiarygodności** to prawdopodobieństwo wygenerowania naszego zbioru danych przez model:

$$L = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

- w praktyce minimalizuje się **negative log likelihood (NLL):**

$$NLL = -\log(L) = \sum_{i=1}^{n} -y_i \ln p(x_i) - (1 - y_i) \ln(1 - p(x_i))$$

Obliczanie współczynników `w`

- dla każdego przykładu mamy **entropię krzyżową (cross-entropy)**, czyli miarę różnicy między prawdziwym rozkładem klas a przewidywanym przez model

$$-y_i \ln p(x_i) - (1 - y_i) \ln(1 - p(x_i))$$

- można użyć spadku wzdłuż gradientu lub dowolnego innego optymalizatora, żeby zoptymalizować według wektora wag `w`

$$-y_i \ln \sigma\left(x_i^T w\right) - (1 - y_i) \ln\left(1 - \sigma\left(x_i^T w\right)\right)$$

Regularyzacja

- działa analogicznie jak w przypadku regresji liniowej
- dodajemy po prostu dodatkowy czynnik do funkcji kosztu: L1, L2 lub ElasticNet

$$C(w) = NLL(w) + \lambda||w||_1 + \gamma||w||_2^2$$

- L1 i ElasticNet wymagają specjalnych solwerów
- rodzaj regularyzacji i moc regularyzacji to **najważniejsze hiperparametry**

Regresja logistyczna - zalety i wady

- **zalety:**
  - prostota i interpretowalność
  - szybkość i skalowalność
  - mało hiperparametrów, łatwo dostroić
  - zwraca prawdopodobieństwa
- **wady:**
  - często zbyt prosty model
  - tylko klasyfikacja binarna