1. Uczenie maszynowe nienadzorowane i nadzorowane. Przykłady.

- **Uczenie nadzorowane:**
  - System otrzymuje zbiór przykładów uczących, w postaci (x_i, y_i) i ma na celu nauczyć się funkcji f, takiej że $f(x_i) = y_i$ dla każdego i.
  - Liczba wartości y_i jest określona i nazywamy je klasami.
  - Uczenie maszynowe nadzorowane można podzielić:
    - Klasyfikacje - odpowiedź należy do jednej z klas. Np. “Co to za zwierzę na zdjęciu?” -> “Pies”
    - Regresja - odpowiedź jest rzeczywista wartością. Np. “Jaka będzie wartość 50m mieszkania za rok? -> “200 000”.
  - Przykłady algorytmów:
    - Linear regression (regresja)
    - Random forest (regresja, klasyfikacja)
    - logistic regression
    - *Support Vector Machine*, **SVM (ultra ważne, często o tym na wykładzie mówił!!!)**
    - Artificial neural network
- **Uczenie nienadzorowane:**
  - Dostarczamy tylko dane wejściowe bez odpowiadających im danych wyjściowych.
  - Głównym celem uczenia nienadzorowanego jest stworzeniu modelu opisującego ukryte struktury i relacje w nieetykietowanym zbiorze danych.
  - Uczenie maszynowe nienadzorowane można podzielić na:
    - Clustering - grupowanie obiektów o podobnych cechach.
    - Association - wykrywanie relacji pomiędzy zmiennymi opisującymi obiekty.
  - Przykłady algorytmów:
    - K-Means (clustering)
    - Apriori algorithm (association)

W **uczeniu nadzorowanym** (ang. supervised learning) dane uczące przekazywane algorytmowi zawierają dołączone rozwiązania problemu, tzw. **etykiety** (ang. labels);
Klasycznym zadaniem uczenia nadzorowanego jest **klasyfikacja**. Filtr spamu stanowi tu dobry przykład: jest on trenowany za pomocą dużej liczby przykładowych wiadomości e-mail należących do danej **klasy** (spamu lub hamu), dzięki którym musi być w stanie klasyfikować wiadomości.
Innym typowym zadaniem uczenia nadzorowanego jest przewidywanie docelowej wartości numerycznej, takiej jak cena samochodu, przy użyciu określonego zbioru cech (przebieg, wiek, marka itd.), zwanych **czynnikami prognostycznymi/predykcyjnymi** (ang. predicators). Ten typ zadania nosi nazwę **regresji**. Aby wyuczyć dany system, należy mu

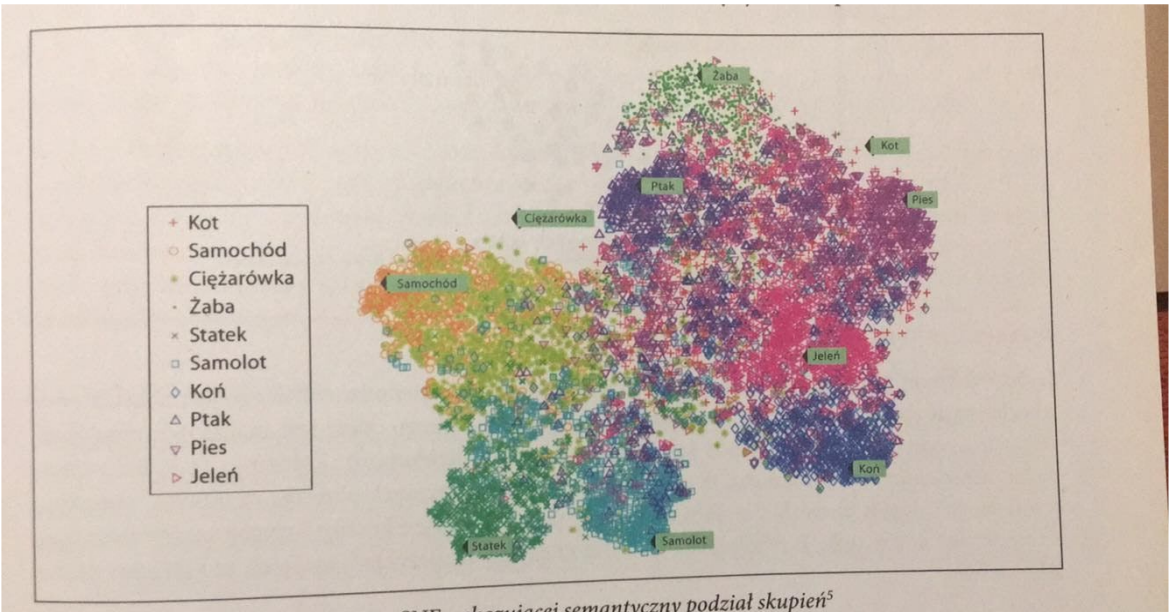
podać wiele przykładów samochodów, w tym zarówno etykiety (np. ceny), jak i czynniki prognostyczne.

W **uczeniu nienadzorowanym** dane uczące są nieoznakowane. System próbuje uczyć się bez nauczyciela. Kilka najważniejszych algorytmów uczenia nienadzorowanego:

- analiza skupień
  - metoda k-średnich
  - hierarchiczna analiza skupień (HCA)
  - algorytm oczekiwanie-maksymalizacja
- wizualizacja i redukcja wymiarowości
  - analiza głównych składowych (PCA)
  - jądrowa analiza głównych składowych
  - lokalnie liniowe zanurzanie (LLE0)
  - stochastyczne zanurzanie sąsiadów przy użyciu rozkładu t
- uczenie przy użyciu reguł asocjacyjnych
  - algorytm Apriori
  - algorytm Eclat

Załóżmy, że masz do dyspozycji mnóstwo danych dotyczących osób odwiedzających Twój blog. Możesz chcieć skorzystać z analizy skupień (ang. **clustering),** aby spróbować określić grupy podobnych użytkowników. W dowolnym momencie możesz sprawdzić, do której grupy zalicza się dana osoba odwiedzająca: powiązania pomiędzy poszczególnymi użytkownikami są określane bez Twojej pomocy. Algorytm ten może, przykładowo, zauważyć, że 40% odwiedzających osób to męscy miłośnicy porno, przeglądający Twoją stronę głównie wieczorami, podczas gdy grupa 20% innych użytkowników to młodociani czytelnicy fantastyki naukowej, którzy zaglądają na Twój blog tylko w weekendy. Jeśli użyjesz algorytmu hierarchicznej analizy skupień, może on dodatkowo rozdzielić grupy na mniejsze podjednostki. W ten sposób możesz łatwiej określić, jakie wpisy umieszczać dla poszczególnych grup.

Dobrym przykładem algorytmów uczenia nienadzorowanego są również algorytmy wizualizujące: wprowadzasz dużo złożonych, nieoznakowanych danych, które następnie są wyświetlane w postaci punktów na dwu- lub trójwymiarowym wykresie.



Rysunek 1.9. Przykład wizualizacji t-SNE wskazującej semantyczny podział skupień[5]

2. Regresja liniowa, korelacje (statistics).

- Regresja liniowa jest algorytmem bazującym na uczeniu nadzorowanym
- Jest metodą modelowania liniowej zależności między jedną lub większą liczbą zmiennych wejściowych *x*, a pojedynczą zmienną wyjściową *y*.
- W przypadku pojedynczej zmiennej *x* metodę określa się jako **prostą regresję liniową**. Dla prostej regresji liniowej, prosta regresji jest postaci: $y = b0 + b1 * x$, gdzie *b0* nazywamy przecięciem, b1 nachyleniem linii regresji.
- W przypadku wielu zmiennych wejściowych metodę określa się jako **wielokrotną regresję liniową**. Dla wielokrotnej regresji liniowej wzór ma postać: $y = b0 + (b1 * x1) + ... + bnxn$

Funkcja kosztu określa jak dobrze funkcja hipotezy h odwzorowuje wartości Y dla aktualnych wartości parametrów. Jedną z funkcji określania kosztu funkcji hipotezy jest funkcja najmniejszych kwadratów. Funkcja najmniejszych kwadratów oblicza sumę kwadratów różnic pomiędzy wartościami przybliżonymi przez funkcję hipotezy, a przybliżanymi wartościami Y.

$$J(\theta) = J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2$$

Korelacja:

- zależność korelacyjna pomiędzy cechami X i Y charakteryzuje się tym, że wartościom jednej cechy są przyporządkowane ściśle określone wartości średnie drugiej cechy.
- celem analizy korelacji jest stwierdzenie, czy między badanymi zmiennymi zachodzą jakieś zależności, jaka jest ich siła, jaka jest ich postać i kierunek

- Współzależność między zmiennymi może być dwojakiego rodzaju:
    - funkcyjna - polega na tym, że zmiana wartości jednej zmiennej powoduje ścisle określoną zmianę wartości drugiej zmiennej.
    - probabilistyczna - występuje wtedy, gdy wraz ze zmianą wartości jednej zmiennej zmienia się rozkład prawdopodobieństwa drugiej zmiennej.
- Współczynnik korelacji liniowej Pearsona r_{xy}:
    - Współczynnik r_{xy} jest miernikiem siły związku prostoliniowego między dwiema cechami mierzalnymi.

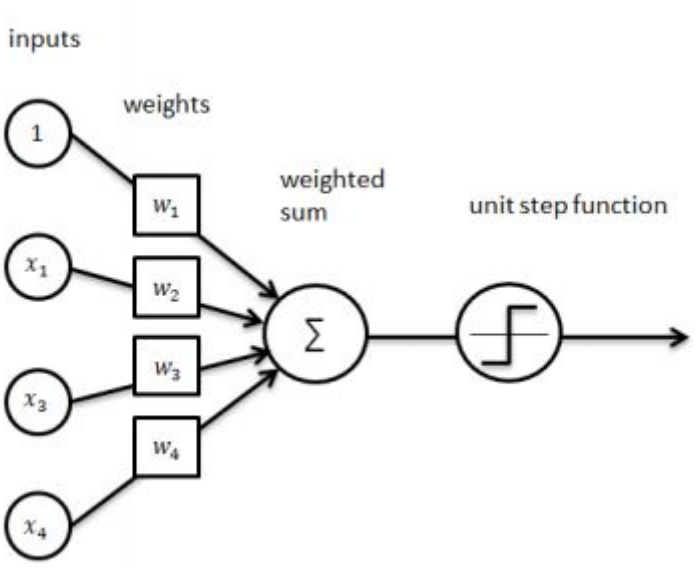
$$r_{xy} = r_{yx} = \frac{\operatorname{cov}(x,y)}{s(x)s(y)}$$

3. Perceptron:

- Składa się z:
    - n wejść $x_1, \ldots, x_n$ (argumenty funkcji,
    - n wag stowarzyszonych z wejściami $w_1, \ldots, w_n$
    - funkcji aktywacji:

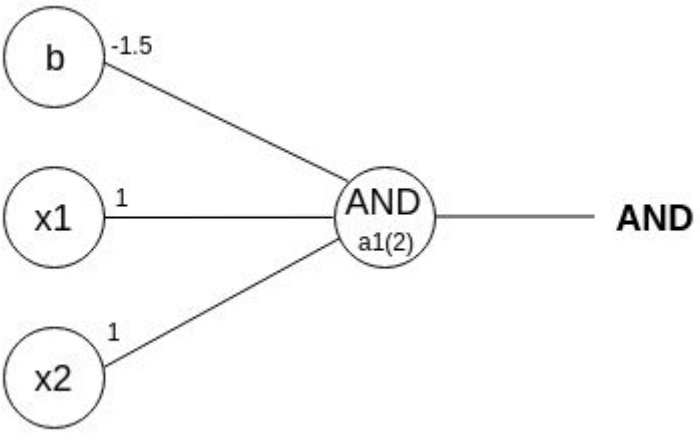
$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Budowa:

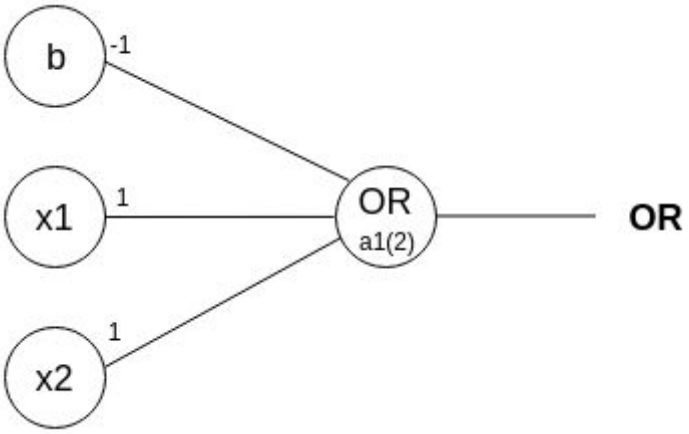


- Używany do tworzenia klasyfikatorów binarnych.
- Perceptrony dzielimy na jednowarstwowe i wielowarstwowe.

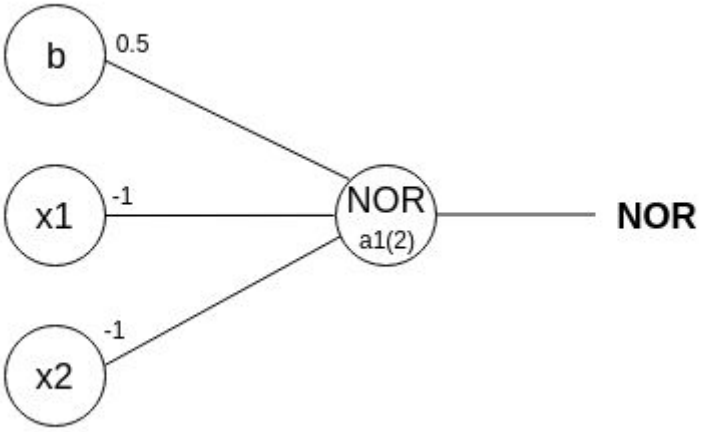
- Perceptron otrzymuje wiele sygnałów wejściowych, jeśli suma sygnałów będzie większa niż wartość progowa, to perceptron generuje sygnał.
- Bramki logiczne na perceptronie:
    - AND:



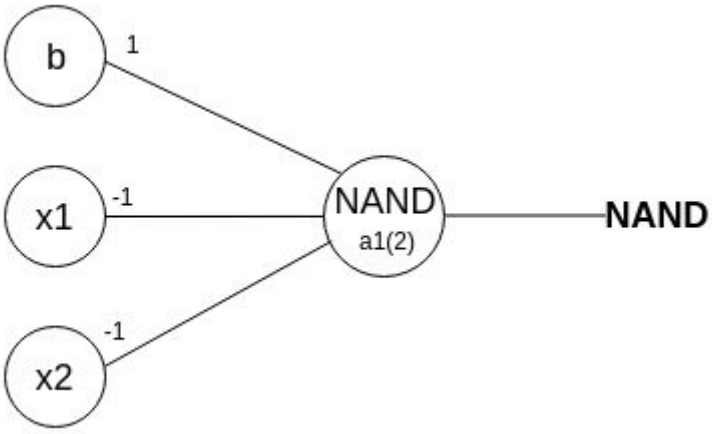
    - OR:



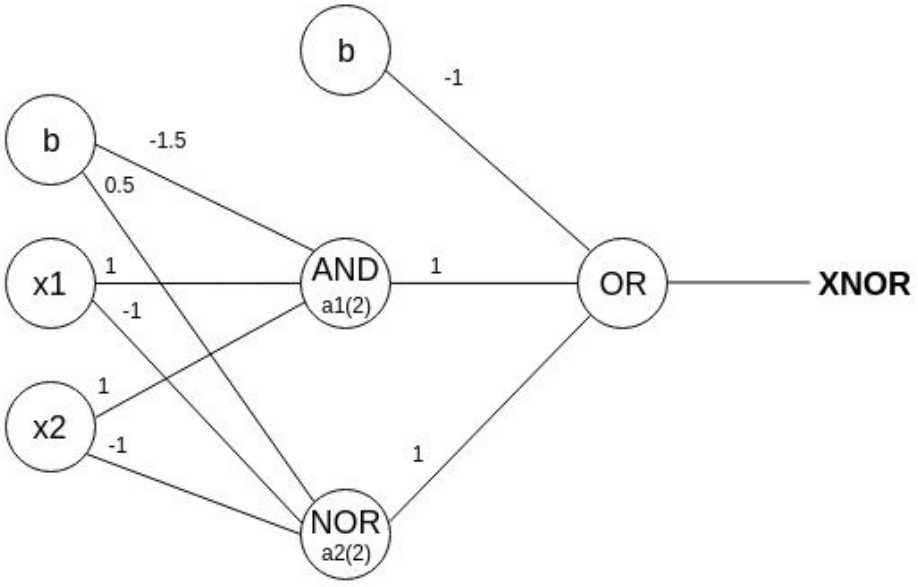
    - NOR:



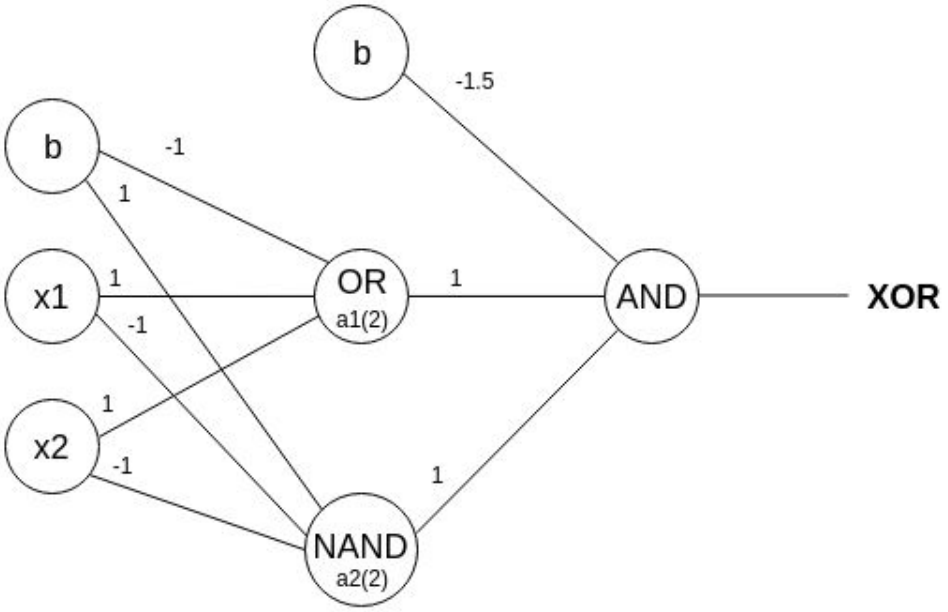
- NAND:


b
1
x1
-1
NAND
a1(2)
NAND
-1
x2

- XNOR:


b
-1
b
-1.5
0.5
x1
1
AND
a1(2)
1
OR
XNOR
-1
1
x2
-1
1
NOR
a2(2)

- XOR:



4. Sieci neuronowe, backpropagation.

Propagacja wsteczna (ang. backpropagation) jest powszechnie używanym algorytmem do uczenia jednokierunkowych sieci neuronowych podczas nadzorowanego uczenia maszynowego. Polega na “przenoszeniu” błędu, jaki popełniła sieć, w kierunku od warstwy wyjściowej do warstwy wejściowej (wstecz w stosunku do kierunku przepływu informacji). Cykl uczenia metodą wstecznej propagacji błędu składa się z następujących etapów:

- Wyznaczenie odpowiedzi neuronów warstwy wyjściowej oraz warstw ukrytych na zadany sygnał wejściowy.
- Wyznaczenie błędu popełnionego przez neurony znajdujące się w warstwie wyjściowej i przesłanie go w kierunku warstwy wejściowej.
- Adaptacja wag.

Podsumowując, algorytm wstecznej propagacji błędu można zapisać następująco:

- Wygeneruj losowo wektor wag
- Podaj wybrany wzorzec na wejście sieci
- Wyznacz odpowiedzi dla wszystkich neuronów wyjściowych sieci:

$$y_k^{wyj} = \mathrm{f}\left(\sum_{j=1}^{J} w_{kj}^{wyj} y_j^{wyj-1}\right)$$

- Oblicz błędy wszystkich neuronów warstwy wyjściowej:

$$\delta_k^{wyj} = z_k - y_k^{wyj}$$

- Oblicz błędy w warstwach ukrytych:

$$\delta_j^{h-1} = \frac{d\,\mathrm{f}(u_j^{h-1})}{du_j^{h-1}} \sum_{k=1}^{l} \delta_k^h w_{kj}^h$$

- Zmodyfikuj wagi wg zależności:

$$w_{ji}^{h-1} = w_{ji}^{h-1} + \eta \delta_j^{h-1} y_i^{h-1}$$

- Jeżeli wartość funkcji celu jest zbyt duża wróć do pkt 2.

CNN – Convolutional neural network

- Rozpoznaje wzorce w przestrzeni;
- Jednostronny przepływ informacji;
- Stosowany do rozpoznawania komponentów obrazów (cech, kształtów, twarzy);
- Wymaga stałej długości wejścia, generuje stałej długości wyjście;

RNN – Recursive neural network

- Rozpoznaje wzorce w czasie;
- Przepływ informacji działa w obie strony (dostaje feedback z wyjścia – takie sprzężenie zwrotne). Przykładowo znaczenie słowa “dog” zmieni się w zależności od tego, czy występuje przed nim fraza “hot”;
- Stosowany do rozpoznawania tekstów, mowy;
- Używa zmiennej długości wejścia/wyjścia;

5. Regresja z regularyzacją: metody ridge, lasso.

Metoda ridge (regresja grzbietowa):

Kryterium zawiera sumę kwadratów wag jako składnik kary:

$$E(h_{\mathbf{w}}) = \sum_{\mathbf{x} \in P} (y - h_{\mathbf{w}}(\mathbf{x}))^2 + \lambda \sum_{i=1}^{n} w_i^2 = (\mathbf{Y} - \mathbf{X}\mathbf{w})^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$$

$\lambda \geq 0$ jest parametrem określającym stopień uwzględnienia kary w kryterium

Dla $\lambda = 0$ otrzymujemy zwykły model regresji liniowej. Dla $\lambda = \infty$ otrzymujemy zerowe wagi.

Aby wyrównać wpływ poszczególnych wag na wartość kary przed wykonaniem obliczeń należy sprowadzić wartości wszystkich atrybutów do tej samej skali.

Metoda LASSO jest metodą regularyzacji modelu regresji liniowej, w której kryterium zawiera sumę modułów wag jako składnik kary:

$$E(h_{\mathbf{w}}) = \sum_{\mathbf{x} \in P} (y - h_{\mathbf{w}}(\mathbf{x}))^2 + \lambda \sum_{i=1}^{n} |w_i|$$

Wprowadzenie kary w postaci sumy modułów zamiast sumy kwadratów wag ma ciekawe konsekwencje. W regresji grzbietowej wagi zmniejszają się wraz ze wzrostem λ, ale nigdy nie osiągają zera. W LASSO wagi mogą się zerować przy odpowiednio dużych wartościach λ. LASSO jest jednocześnie algorytmem regularyzacji i selekcji atrybutów. Wyznaczenie wartości wag minimalizujących kryterium używane w LASSO nie jest możliwe na drodze analitycznej, lecz wymaga algorytmu iteracyjnego.

6. Examples of machine learning models.

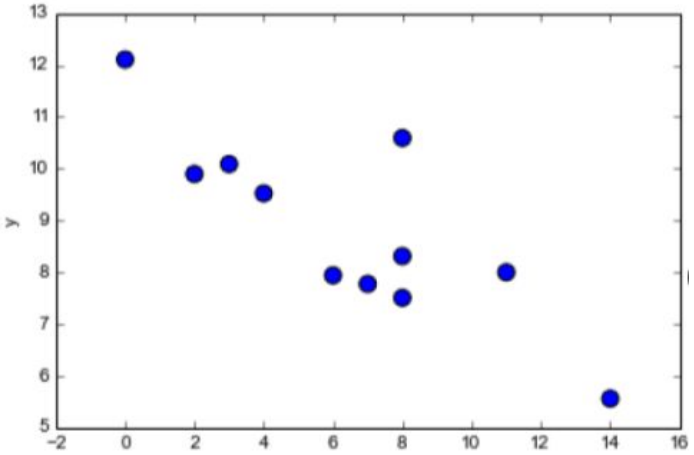
- Linear Regression
- Logistic Regression
- Decision Tree
- Support Vector Machine
- Naive Bayes
- kNN
- K-Means
- Random Forest
- Dimensionality Reduction Algorithms

- oprogramowanie do rozpoznawania mowy:
    - automatyczne tłumaczenie
    - rozpoznawanie mowy ludzkiej
    - dyktowanie komputerowi
    - interfejsy użytkownika sterowane głosem
    - automatyzacja głosem czynności domowych
    - interaktywne biura obsługi
    - rozwój robotów
- automatyczna nawigacja i sterowanie:
    - kierowanie pojazdem (ALVINN)
    - odnajdywanie drogi w nieznanym środowisku
    - kierowanie statkiem kosmicznym (NASA Remote Agent)
    - automatyzacja systemów produkcji i wydobycia (przemysł, górnictwo)
- analiza i klasyfikacja danych:
    - systematyka obiektów astronomicznych (NASA Sky Survey)
    - rozpoznawanie chorób na podstawie symptomów
    - modelowanie i rozwijanie terapii lekowych
    - rozpoznawania pisma na podstawie przykładów
    - klasyfikowanie danych do grup tematycznych według kryteriów
    - aproksymacja nieznanej funkcji na podstawie próbek
    - ustalanie zależności funkcyjnych w danych
    - przewidywanie trendów na rynkach finansowych na podstawie danych mikro- i makroekonomicznych
    - wykrywanie prania pieniędzy

7. K-Nearest Neighbors.

KNN - K-Nearest Neighbors - może być używany zarówno przy problemie klasyfikacji jak i regresji. Algorytm używa podobieństw cech do przewidzenia wartości dla nowych obiektów populacji. Oznacza to, że nowemu punktowi przypisuje się wartość na podstawie tego, jak bardzo przypomina on punkty w zbiorze treningowym.

$$D = \begin{bmatrix} (8, & 8.31) \\ (14, & 5.56) \\ (0, & 12.1) \\ (6, & 7.94) \\ (3, & 10.09) \\ (2, & 9.89) \\ (4, & 9.52) \\ (7, & 7.77) \\ (8, & 7.51) \\ (11, & 8.0) \\ (8, & 10.59) \end{bmatrix}$$



https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/

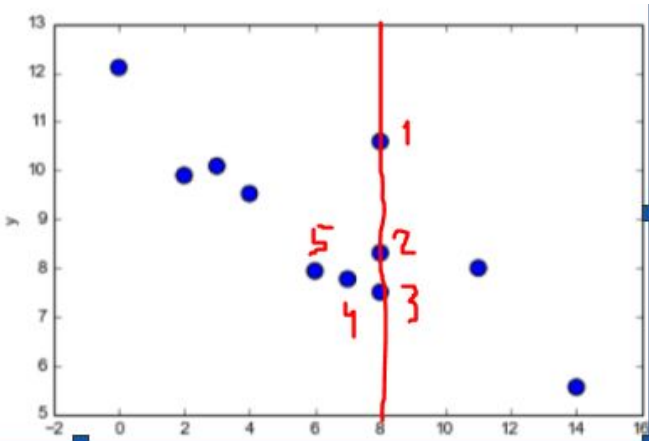
Opis przykładu z linka

1. Tabela zawierająca wartości <x, y, z> czyli wzrost, wiek, waga. Będziemy poszukiwać brakującej wartości z.
2. Narysowany wykres prezentuje układ dwuwymiarowy dla x i y - ponieważ tylko te wartości dla wszystkich osób są znane.
3. Na podstawie szukanej osoby obliczamy z jej znanych x oraz y najbliższych sąsiadów. (do obliczania najbliższych sąsiadów posługujemy się tylko kolumnami w których wszystkie wartości są znane, nie wykorzystujemy z ponieważ właśnie tego poszukujemy).
4. W przykładzie wybranych jest 3 najbliższych sąsiadów.
5. Z tych sąsiadów wyciągamy wartości z i na ich podstawie liczymy średnią arytmetyczną nasze poszukiwane z.

Przełożenie na nasze zadanie:

1. U nas mamy tabelę z wartościami <x, y> z czego poszukujemy y.
2. Znamy wszystkie x, dlatego tylko na podstawie x możemy wyliczać odległości i znajdować sąsiadów.
3. x = 8, zatem poszukujemy 5 sąsiadów co ich wartość x jest najbliżej 8.
4. Znaleźliśmy 1, 2, 3, 4, 5 - zaznaczone na wykresie.
5. Liczymy nasze brakujące y na podstawie ich y.
6. (8.31 + 10.59 + 7.51 + 7.77 + 7.94) / 5 = 8.424 to jest nasze poszukiwane y

$$D = \begin{bmatrix} (8, & 8.31) \\ (14, & 5.56) \\ (0, & 12.1) \\ (6, & 7.94) \\ (3, & 10.09) \\ (2, & 9.89) \\ (4, & 9.52) \\ (7, & 7.77) \\ (8, & 7.51) \\ (11, & 8.0) \\ (8, & 10.59) \end{bmatrix}$$



8. Drzewa decyzyjne, XGBOOST.

Drzewa decyzyjne są graficzną metodą wspomagania procesu decyzyjnego. Drzewo składa się z korzenia oraz gałęzi prowadzących z korzenia do kolejnych wierzchołków. Wierzchołki, z których wychodzi co najmniej jedna krawędź, są nazywane węzłami, a pozostałe wierzchołki – liśćmi. Węzły są opisywane przez atrybuty eksplorowanej relacji, krawędzie opisują możliwe wartości dla atrybutu. W każdym węźle sprawdzany jest pewien warunek dotyczący danej obserwacji, i na jego podstawie wybierana jest jedną z gałęzi prowadząca do kolejnego wierzchołka. Klasyfikacja danej obserwacji polega na przejściu od korzenia do liścia i przypisaniu do tej obserwacji klasy zapisanej w danym liściu.

Przykład drzewa decyzyjnego - sprawdzenie, czy warunki atmosferyczne pozwalają na grę w tenisa

zachmurzenie	temperatura	wilgotność	wiatr	decyzja
słońce	gorąco	wysoka	słaby	nie
słońce	gorąco	wysoka	silny	nie
pochmurno	gorąco	wysoka	słaby	tak
deszcz	średnio	wysoka	słaby	tak
deszcz	chłodno	normalna	słaby	tak
deszcz	chłodno	normalna	silny	nie
pochmurno	chłodno	normalna	silny	tak
słońce	średnio	wysoka	słaby	nie
słońce	chłodno	normalna	słaby	tak
deszcz	średnio	normalna	słaby	tak
słońce	średnio	normalna	silny	tak
pochmurno	średnio	wysoka	silny	tak
pochmurno	gorąco	normalna	słaby	tak
deszcz	średnio	wysoka	silny	nie

IF zachmurzenie = słońce AND wilgotność = normalna THEN graj

Estymacja błędu klasyfikowania:

Błąd klasyfikowania: $e = \frac{b}{n}$

Dokładność klasyfikowania: 1 - e

gdzie **n** to liczba wszystkich klasyfikowanych obiektów, a **b** to liczba błędnie sklasyfikowanych obiektów

Entropia - niepewność wystąpienia danego zdarzenia w następnej chwili. Jest miarą informacji. Jeśli zdarzenie występuje z prawdopodobieństwem równym 1, to entropia wynosi 0 (nie ma niepewności).

$$H(Y) = -\sum_{i=1}^{K} p_k \, log_2 p_k$$

$\mathbf{p_k}$ - prawdopodobieństwo zajścia zdarzenia k

Entropia dla przykładu podanego powyżej:

$$H(Y) = -\sum_{i=1}^{K} p_k \, log_2 p_k = -\frac{5}{14} log_2 \frac{5}{14} - \frac{9}{14} log_2 \frac{9}{14} = 0.94$$

- 5/14, bo występuje 5 odpowiedzi “nie”, a 9/14, bo występuje 9 odpowiedzi “tak”
- 0.94 to entropia prawie jeden, co oznacza, że podział decyzyjny jest taki, że jest dużo jednej i drugiej opcji (5 nie do 9 tak).

Information gain - przyrost informacji, miara zmiany entropii, mierzy ile informacji o klasie daje nam funkcja.

$$InfoGain(Y, t) = H(Y) - H(Y|t)$$

H(Y) - entropia zdarzenia Y
H(Y|t) - entropia warunkowa dla zdarzenia Y z danym atrybutem t

Information gain dla przykładu podanego powyżej (wilgotności):

Humidity left – patrzymy na normal, dla normal jest 6 **yes** i jedno **no**. HL to podział

$$InfoGain(Humidity) = H(Y) - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R = 0.94 - \frac{7}{14}H_L - \frac{7}{14}H_R$$

$$H_L = -\frac{6}{7}log_2\frac{6}{7} - \frac{1}{7}log_2\frac{1}{7} = 0.592$$

$$H_R = -\frac{3}{7}log_2\frac{3}{7} - \frac{4}{7}log_2\frac{4}{7} = 0.985$$

$$InfoGain(Humidity) = H(Y) - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R = 0.94 - \frac{7}{14}0.592 - \frac{7}{14}0.985 = 0.94 - 0.296 - 0.4925 = 0.1515$$

- Wartości dla Humidity to **High** i **Normal**.na 6/7 i 1/7.
- **Humidity right** – dla high mamy 3 **yes** na 4 **no**.
- InfoGain chcemy mieć jak najwyższy. Gdyby wszystkie normal były **yes** i wszystkie high byłyby **no**, to byłby koniec drzewa decyzyjnego, nie musimy wtedy sprawdzać reszty atrybutów. Wartości HL i HR byłyby równe 0.

XGBOOST

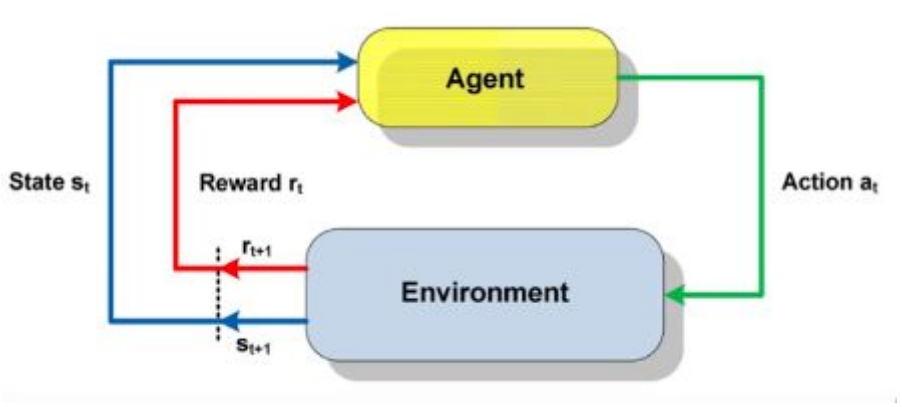
- Dobrze się skaluje,
- Idealny do obliczeń równoległych i rozproszonych
- Jedna z metod uczenia zespołowego (ensemble learning) - zamiast polegać na wyniku z jednego modelu, wykorzystuje się moc predykcji wielu modelów. Rezultatem jest jeden model agregujący wyniki z kilku modeli.
- Techniki uczenia zespołowego:
    - Bagging:
        - Oznacza agregację bootstrap.
        - Różnorodność klasyfikatorów uzyskuje się za pomocą replik danych treningowych bootstrap. Oznacza to, że różne podzbiory danych treningowych są losowo rysowane z wymianą z całego zbioru danych treningowych.
        - Każdy podzbiór danych treningowych służy do szkolenia innego klasyfikatora tego samego typu. Poszczególne klasyfikatory są następnie łączone, podejmując decyzję zwykłą większością głosów.
    - Boosting:
        - W boostingu drzewa budowane są sekwencyjnie, tak, że każde kolejne drzewo ma na celu zredukować błędy poprzedniego drzewa.
        - Podstawowe modele w boostingu są słabe, a moc ich predykcji jest podobna do zgadywania.
        - Boosting tworzy małe, nie głębokie drzewa.

https://xgboost.readthedocs.io/en/latest/tutorials/model.html <- artykuł z oficjalnej strony xgboost

https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/ <- coś takiego nam pokazywał, słowo w słowo, przerobię to na prostszy język na dniach

9. Uczenie ze wzmocnieniem - reinforcement learning.

Uczenie ze wzmocnieniem (reinforcement learning) **NIE** jest metodą uczenia nadzorowanego lub nienadzorowanego. Opiera się na otrzymywaniu nagród (rewards) oraz kar (penalties) od środowiska w którym operuje. Przykład: w grze Pong nagrodą jest otrzymanie punktu, karą jest zdobycie punktu przez przeciwnika. W jaki sposób algorytm wnioskuje jakie podjęte kroki wpłynęły na zdobycie nagrody? Ten problem zwie się problemem przypisania zasługi (Credit Assignment Problem). W uczeniu ze wzmocnieniem dobre wnioskowanie zwykle wymaga długiego czasu trenowania.



Obserwacja + **informacja o nagrodzie** powodują **akcję**.
Historia jest sekwencją **obserwacji (Observation)**, **akcji (Action)** i **nagród (Reward)**:

$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t$$

Stan to informacja używana do zdeterminowania co się stanie następnie. Formalnie jest funkcją historii:

$$S_t = f(H_t)$$

Stan środowiska (S_t^e) nie jest zwykle widoczny dla agenta.

Stan informacji (stan Markova) zawiera wszystkie ważne informacje z historii

Stan S_t jest stanem Markova wtedy i tylko wtedy gdy:

$$P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_1,\ldots,S_t]$$

Stan środowiska S_t^e oraz Historia H_t są stanami Markova.

W pełni obserwowalne środowiska

Pełna obserwowalność: agent bezpośrednio obserwuje stan środowiska

$$O_t = S_t^a = S_t^e$$

- stan agenta = stan środowiska = stan informacji
- formalnie, jest to proces decyzyjny Markova (MDP)

Częściowo obserwowalne środowiska

- częściowa obserwowalność: agent pośrednio obserwuje środowisko
    - robot z kamerą nie jest informowany o jego bezwzględnej lokalizacji
    - handlujący agent tylko obserwuje obecne ceny
    - grający w pokera agent tylko obserwuje odkryte karty
- w tym przypadku stan agenta =/= stan środowiska

Uczenie ze wzmocnieniem (ang. reinforcement learning) - system uczący zwany w tym kontekście agentem, może obserwować środowisko, dobierać i wykonywać czynności, a także odbierać nagrody i kary. Musi następnie samodzielnie nauczyć się najlepszej strategii (polityki), aby uzyskiwać jak największą nagrodę. Polityka definiuje rodzaj działania, jakie agent powinien wybrać w danej sytuacji. Np. metody uczenia przez wzmacnianie są używane w wielu modelach robotów do nauki chodzenia.

10. Word embedding.

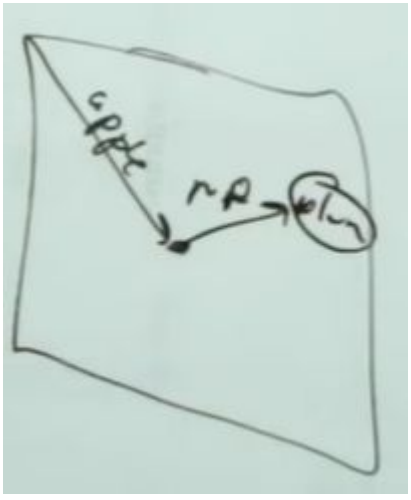
[Filmik]
[Filmik2]

Word Embedding => Zbiorowy termin dla modeli, które nauczyły się mapować zestaw słów lub zwrotów w słownictwie na wektory wartości liczbowych.

Word2Vec (stworzony przez googla. Model ma 300 wymiarów stworzony na 3 milionach słów z google news data) - proces uczenia maszynowego podobny do

klasyfikacji czy regresji.

Jest to model, który wytwarza coś nazywanego **Word Embedding.** Każde słowo może być przedstawiono jako numery (vector). Czyli np. “apple” może zostać przedstawione jako jakaś liczba lub tablica liczb. (tych liczb może być 1,2 albo i 100). Załóżmy, że mamy 2 liczby (2 wymiary). Pozwala to na “dodawanie” do siebie słów i np “apple” + “purple” jako wektory może wskazywać na jakieś inne słowo czy też wyszukiwanie słów o podobnym znaczeniu. Wykorzystywany, aby znaleźć relacje pomiędzy słowami. Np, że dog i cat mają ze sobą coś wspólnego. Tak samo blue czy red.



Word Embedding polega na zwiększeniu możliwości sieci neuronowych aby mogły się uczyć z tekstu. Wektory te nazywa się “Embedding”.

Jednym z większych ograniczeń w “Word embedding” jest to, że jedno słowo może mieć wiele znaczeń. Np zamek. a przez model jest stosowane jako pojedyncza reprezentacja.