ML Wstęp, Regresje

ML Wstęp, Regresje

Agenda

1. **Rodzaje uczenia maszynowego**
2. Inżynieria cech (enkodowanie, skalowanie, standaryzacja i normalizacja)
3. Metryki klasyfikacji i regresji
4. Regresja liniowa

ML Wstęp, Regresje

Agenda

1. Rodzaje uczenia maszynowego
2. **Inżynieria cech (enkodowanie, skalowanie, standaryzacja i normalizacja)**
3. Metryki klasyfikacji i regresji
4. Regresja liniowa

ML Wstęp, Regresje

Agenda

1. Rodzaje uczenia maszynowego
2. Inżynieria cech (enkodowanie, skalowanie, standaryzacja i normalizacja)
3. **Metryki klasyfikacji i regresji**
4. Regresja liniowa

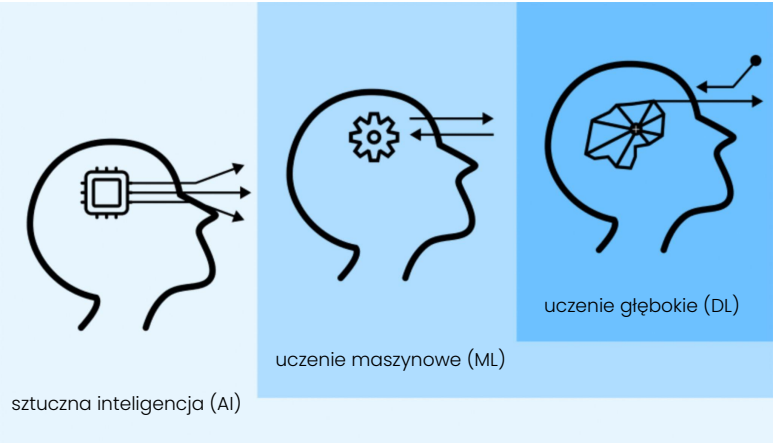
ML Wstęp, Regresje

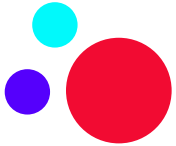
Agenda

1. Rodzaje uczenia maszynowego
2. Inżynieria cech (enkodowanie, skalowanie, standaryzacja i normalizacja)
3. Metryki klasyfikacji i regresji
4. **Regresja liniowa**

ML Wstęp, Regresje

Uczenie maszynowe



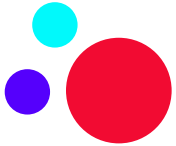


ML Wstęp, Regresje

Podstawowa klasyfikacja metod

Ze względu na sposób uczenia:

- Uczenie nadzorowane (ang. supervised learning)
- Uczenie nienadzorowane (ang. unsupervised learning)
- Uczenie ze wzmocnieniem (ang. reinforcement learning)

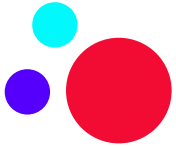


ML Wstęp, Regresje

Uczenie nadzorowane

Model uczy się odwzorowywać dane wejściowe na dane wyjściowe $X \rightarrow Y$.

W tym zagadnieniu dane są pary wartości wejściowych x_i oraz wyjściowych y_i.



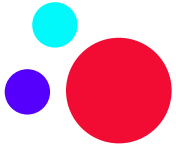
ML Wstęp, Regresje

Uczenie nadzorowane

Przykłady modeli i algorytmów.

Parametryczne:

- regresja liniowa/nieliniowa (ang. linear/non-linear regression),
- logistyczna (ang. logistic regression).



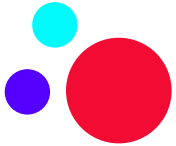
ML Wstęp, Regresje

Uczenie nadzorowane

Przykłady modeli i algorytmów.

Nieparametryczne:

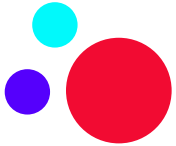
- maszyna wektorów wspierających (ang. support vector machine - SVM),
- drzewa losowe (ang. random trees),
- XGBoost.



ML Wstęp, Regresje

Uczenie nienadzorowane

Algorytm wyszukuje w danych wzorce (ang. pattern) nie mając informacji wyjściowej (Y). Wyszukiwanie wzorców odbywa się z wykorzystaniem optymalizacji (maksymalizacja funkcji celu lub minimalizacja funkcji kosztu).

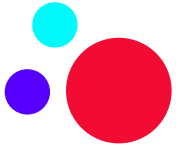


ML Wstęp, Regresje

Uczenie nienadzorowane

Przykłady modeli i algorytmów (nieparametryczne):

- algorytm centroidów (k-means),
- t-distributed Stochastic Neighbor Embedding (t-SNE),
- Uniform Manifold Approximation and Projection for Dimension Reduction (uMAP).



ML Wstęp, Regresje

Klasyfikacja metod

Ze względu na typ modelu:

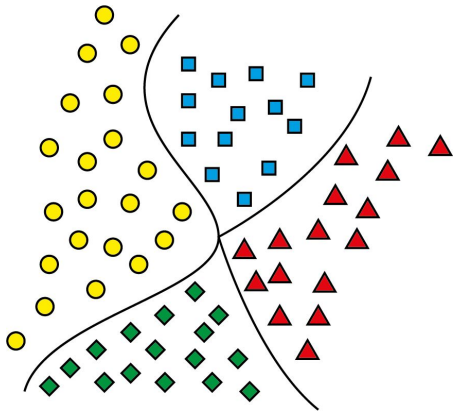
- Klasyfikacja (ang. classification)
- Regresja (ang. regression)
- Klasteryzacja (ang. clustering)
- Inne:
  - Redukcja wymiarowości (ang. dimensionality reduction)
  - Uczenie zależności (ang. association rule mining)
  -...

ML Wstęp, Regresje

Klasyfikacja

Kluczowe aspekty:

- cel klasyfikacji,
- przykłady zastosowań,
- dane treningowe i etykiety.

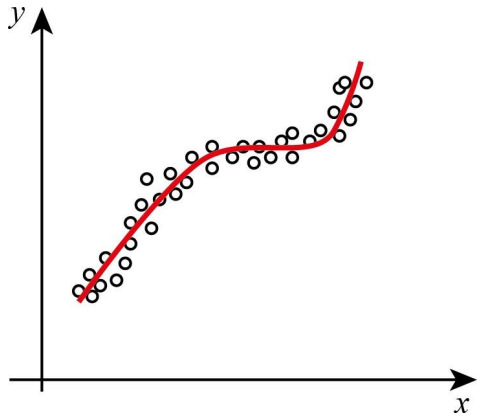


ML Wstęp, Regresje

Regresja

Kluczowe aspekty:

- cel regresji,
- przykłady zastosowań,
- dane treningowe i etykiety.

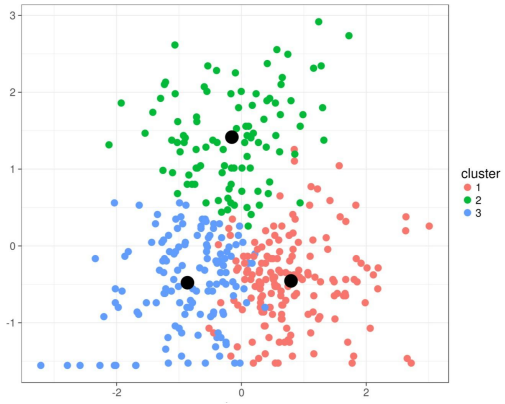


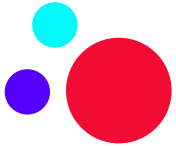
ML Wstęp, Regresje

Klasteryzacja

Kluczowe aspekty:

- cel klasteryzacji,
- przykłady zastosowań,
- dane treningowe i etykiety.



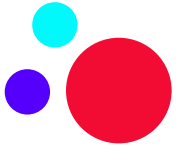


ML Wstęp, Regresje

Redukcja wymiarowości

Kluczowe aspekty:

- cel redukcji wymiarowości,
- przykłady zastosowań,
- dane treningowe i etykiety.

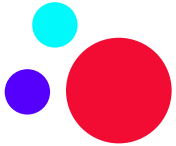


ML Wstęp, Regresje

Uczenie zależności

Kluczowe aspekty:

- cel uczenia zależności,
- przykłady zastosowań,
- dane treningowe i etykiety.

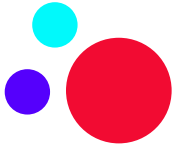


ML Wstęp, Regresje

Szeregi czasowe

Kluczowe aspekty:

- charakterystyka szeregów czasowych,
- zastosowania w Machine Learningu,
- dane treningowe i testowe.

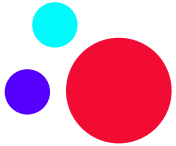


ML Wstęp, Regresje

Klasyfikacja metod

Ze względu na typ modelu:

- parametryczne (ang. parametric),
- nieparametryczne (ang. non-parametric).



ML Wstęp, Regresje

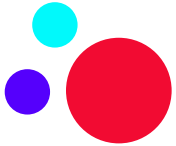
Modele parametryczne

Modele parametryczne mają z góry określoną postać funkcyjną, a uczenie modelu polega na określeniu parametrów. Nadają się raczej do nieskomplikowanych zagadnień.

Przykłady modeli parametrycznych:

- regresja liniowa i nieliniowa,
- regresja logistyczna.

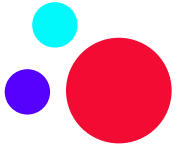
Np. w przypadku regresji liniowej $y=a \cdot x+b$ parametrami są a i b.



ML Wstęp, Regresje

Modele nieparametryczne

Modele nieparametryczne nie posiadają określonej formy funkcyjnej. Są to zwykle bardzo elastyczne modele umożliwiające tworzenie bardzo skomplikowanych odwzorowań.



ML Wstęp, Regresje

Inżynieria cech

Inżynieria cech (ang. feature engineering) polega na przekształcaniu istniejącego zbioru danych w celu zwiększenia efektywności modelu, np.:

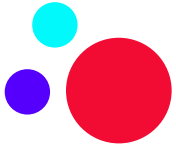
- zamiana danego zbioru cech na zbiór cech nieskorelowanych (np. analiza głównych składowych PCA),
- dodanie nowej cechy zawierającej wartość 1, gdy inna cecha posiada pewną własność i 0 gdy jej nie ma (one-hot encoding),
- rozbicie jednej cechy na kilka np.: zamiana dat na dni, miesiące itp.,
- transformacje cech (np. logarytmowanie, podnoszenie do kwadratu),
- skalowanie cech,
- standaryzacja cech,
- normalizacja cech.

ML Wstęp, Regresje

one-hot encoding

Polega na wprowadzeniu jednej lub więcej cech (nowych kolumn) w oparciu o wartości danej cechy. Np.: załóżmy, że pewna cecha oznacza grupę wiekową: młodzież, dorośli, seniorzy.
Możliwe zatem są trzy wartości. One-hot encoding polega na zamianie tej kolumny na trzy nowe:

wiek		młodzież	dorośli	seniorzy
młodzież		1	0	0
seniorzy		0	0	1
dorośli		0	1	0
dorośli		0	1	0
młodzież	→	1	0	0
seniorzy		0	0	1
seniorzy		0	0	1
młodzież		1	0	0
seniorzy		0	0	1



ML Wstęp, Regresje

one-hot encoding w praktyce

One-hot encoding polega na zamianie wartości (całkowitych, nazw kategorii itp.) na nowe kolumny danych, które zawierają 1 w miejscu wystąpienia danej wartości i 0 w przypadku jej braku.

```
import numpy as np
from sklearn.preprocessing import OneHotEncoder
```

ML Wstęp, Regresje

one-hot encoding – dla pojedynczej cechy (kolumny danych)

Poniżej utworzona zostanie macierz zawierająca nazwy kategorii `a`, `b` lub `c`.

```
# x = df["nazwa_kolumny"].to_numpy()
x = np.array(["a","b","c","a","c","a","c","b","c","a","b"],ndmin=2).transpose()
```

```
array([['a'],
       ['b'],
       ['c'],
       ['a'],
       ['c'],
       ['a'],
       ['c'],
       ['b'],
       ['c'],
       ['a'],
       ['b']], dtype='<U1')
```

wynik

ML Wstęp, Regresje

one-hot encoding – dla pojedynczej cechy (kolumny danych)

Utworzenie obiektu klasy OneHotEncoder wykonuje się poprzez konstruktor klasy `OneHotEncoder`. Domyślnie tworzona jest macierz rzadka, tu w celu lepszej ilustracji zagadnienia wymuszono utworzenie macierzy pełnej poprzez podanie argumentu `sparse=False`.

```
my_encoder = OneHotEncoder(sparse=False)
```

W chwili obecnej obiekt my_encoder jest "pusty". nie zawiera żadnych danych. W celu jego pełnego określenia należy użyć metody fit, a jako argumentu użyć tablicy z kategoriami.

```
my_encoder.fit(x)
```

ML Wstęp, Regresje

one-hot encoding – dla pojedynczej cechy (kolumny danych)

my_encoder.categories_

[array(['a', 'b', 'c'], dtype='<U1')]

ML Wstęp, Regresje

one-hot encoding – dla pojedynczej cechy (kolumny danych)

```
xohe = my_encoder.transform(x)
xohe
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]])
```

ML Wstęp, Regresje

one-hot encoding – dla wielu cech (kolumn danych)

```
X = np.array([["a","b","c","a","c","a","c","b","c","a"],
              [ 1 , 4 , 5 , 1 , 5 , 4 , 3 , 1 , 3 , 1 ],
              ["u","z","z","v","v","u","z","v","u","z"]]).transpose()
```

ML Wstęp, Regresje

one-hot encoding – dla wielu cech (kolumn danych)

Utworzenie enkodera i wypisanie określonych kategorii:

```
my_encoder1 = OneHotEncoder(sparse=False)
my_encoder1.fit(X)
my_encoder1.categories_
```

```
my_encoder1.transform(X)
```

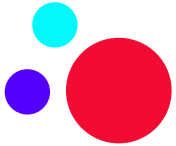
```
array([[1., 0., 0., 1., 0., 0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0., 1., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 0., 1., 0., 0., 1.],
       [1., 0., 0., 1., 0., 0., 0., 0., 1., 0.],
       [0., 0., 1., 0., 0., 0., 1., 0., 1., 0.],
       [1., 0., 0., 0., 0., 1., 0., 1., 0., 0.],
       [0., 0., 1., 0., 1., 0., 0., 0., 0., 1.],
       [0., 1., 0., 1., 0., 0., 0., 0., 1., 0.],
       [0., 0., 1., 0., 1., 0., 0., 1., 0., 0.],
       [1., 0., 0., 1., 0., 0., 0., 0., 0., 1.]])
```

Zadanie 12.1

one-hot encoding (instrukcja)

Zastosuj one-hot encoding, aby przekształcić poniższy array zawierający informacje na temat preferencji muzycznych na reprezentację binarną.

preferencje_muzyczne = np.array(['Rock', 'Pop', 'Jazz', 'Hip-Hop', 'Rock', 'Jazz', 'Pop', 'Hip-Hop'])



ML Wstęp, Regresje

Skalowanie cechy

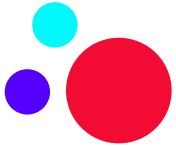
Skalowanie polega na zmianie zakresu zmiennej do pewnego ustalonego zakresu, zwykle od 0 do 1 lub -1 do 1.

Skalowanie do zakresu od 0 do 1:

$$X_{[0;1]} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Skalowanie do nowego zakresu: $[X_{s,min}, X_{s,max}]$

$$X_s = X_{s,min} + \left(\frac{X - X_{min}}{X_{max} - X_{min}}\right)(X_{s,max} - X_{s,min})$$



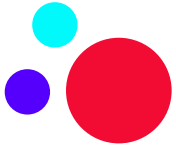
ML Wstęp, Regresje

Skalowanie cech – scikit-learn – pojedyncza kolumna

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler

x = np.array([0.1, 10., 15., -5.1, 6.2],ndmin=2).transpose()

        array([[ 0.1],
               [10. ],
               [15. ],
               [-5.1],
               [ 6.2]])
```



ML Wstęp, Regresje

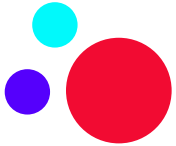
Skalowanie cech – scikit-learn – pojedyncza kolumna

```
my_scaler = MinMaxScaler()

my_scaler.fit(x)

[my_scaler.data_min_, my_scaler.data_max_]

        [array([-5.1]), array([15.])]
```

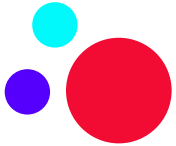


ML Wstęp, Regresje

Skalowanie cech – scikit-learn – pojedyncza kolumna

```
xs = my_scaler.transform(x)
```

```
array([[0.25870647],
       [0.75124378],
       [1.        ],
       [0.        ],
       [0.56218905]])
```



ML Wstęp, Regresje

Skalowanie cech – scikit-learn – wiele kolumn

```
X = np.array([[ 2. , 6.  , 200.  , -450.,  105.  , 344.1, -33.],
              [ 0.1, 0.01,   0.02,    0.02,  0.01,   0.04,  0.032]]).transpose()

              my_scaler1 = MinMaxScaler()
              my_scaler1.fit(X)
              [my_scaler1.data_min_, my_scaler1.data_max_]
```

```
[array([-4.5e+02,  1.0e-02]), array([3.441e+02, 1.000e-01])]
```

ML Wstęp, Regresje

Skalowanie cech – scikit-learn – wiele kolumn

```
Xs = my_scaler1.transform(X)
```

```
array([[0.56919783, 1.        ],
       [0.57423498, 0.        ],
       [0.81853671, 0.11111111],
       [0.        , 0.11111111],
       [0.69890442, 0.        ],
       [1.        , 0.33333333],
       [0.52512278, 0.24444444]])
```

Zadanie 12.2

Skalowanie cech (instrukcja)

Rozważmy zestaw danych o ocenach filmów, który zawiera różne cechy, takie jak "Ocena Krytyków", "Ocena Użytkowników", "Czas Trwania" itp. Twoim zadaniem jest przeprowadzenie skalowania cech, aby dostosować je do określonego zakresu.

```
film_data = np.array([
        [75, 8.0, 120, 3.5],
        [90, 7.5, 150, 4.0],
        [60, 6.0, 100, 3.2],
        [80, 8.5, 130, 4.5],
        [70, 7.0, 110, 3.8]
])
```

ML Wstęp, Regresje

Standaryzacja cechy

Standaryzacja składa się z centrowania i skalowania cechy względem odchylenia standardowego. Polega na odjęciu od cechy jej średniej wartości i podzieleniu przez odchylenie standardowe.

$$u = \frac{x - x}{s}$$

x - średnia, s – odchylenie standardowe, x – skalowana wartość.

ML Wstęp, Regresje

Standaryzacja cechy – scikit-learn - pojedyncza kolumna

import numpy as np
from sklearn.preprocessing import StandardScaler

Poniżej utworzona zostanie macierz zawierająca liczby zmiennoprzecinkowe.

x = np.array([0.1, 10., 15., -5.1, 6.2],ndmin=2).transpose()

```
array([[ 0.1],
       [10. ],
       [15. ],
       [-5.1],
       [ 6.2]])
```

ML Wstęp, Regresje

Standaryzacja cechy – scikit-learn - pojedyncza kolumna

```
my_scaler = StandardScaler()

my_scaler.fit(x)

[my_scaler.mean_, my_scaler.var_]

    [array([5.24]), array([50.4344])]
```

ML Wstęp, Regresje

Standaryzacja cechy – scikit-learn - pojedyncza kolumna

```
xs = my_scaler.transform(x)

    array([[-0.72376852],
           [ 0.67026034],
           [ 1.37431532],
           [-1.4559857 ],
           [ 0.13517856]])
```

Zadanie 12.3

Standaryzacja (instrukcja)

Rozważmy zbiór danych obejmujący informacje o wzroście (w centymetrach) i wadze (w kilogramach) grupy osób. Twoim zadaniem jest zastosowanie standaryzacji, aby dostosować te cechy do jednolitego zakresu.

```
data = np.array([
        [160, 55],
        [170, 65],
        [155, 50],
        [180, 75],
        [175, 70]
])
```

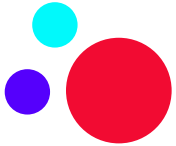
ML Wstęp, Regresje

Normalizacja cechy

Normalizacja cechy polega na skalowaniu cechy względem wybranej miary (np. normy L2). Zwykle skalowaniu podlegają wiersze, ale można również skalować kolumny.

i	x_1	x_{fi}	x_3
1	-5.1	1.4	3.8
2	2.0	4.1	1.6
3	1.0	-1.0	3.0
4	-6.2	3.4	1.1
5	55.1	31	13.3
...	...	...	...

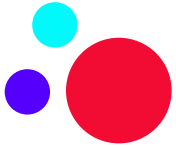
sklearn.preprocessing.normalize



ML Wstęp, Regresje

Odległości między obserwacjami

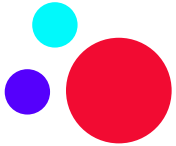
Niektóre algorytmy wymagają określenia odległości między obserwacjami. Odległości oblicza się między danymi obserwacjami tj. wierszami tabeli. Obserwacja może składać się z danych różnego typu, zatem nie w każdym przypadku sensowne jest stosowanie odległości euklidesowej (tzw. długości wektora).



ML Wstęp, Regresje

Odległości między obserwacjami

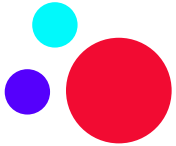
Wartości różnych zmiennych mogą się bardzo różnić, więc odległości zostaną zdominowane przez zmienne o większych wartościach. W takich przypadkach warto zastosować normalizację i/lub standaryzację.



ML Wstęp, Regresje

Metryki między obserwacjami

Metryki (ang. metrics, scoring) służą do oceny efektywności modelu.
W różnych zagadnieniach stosuje się różne metryki. Metryk jest bardzo wiele np. lista dostępnych metryk w scikit-learn.



ML Wstęp, Regresje

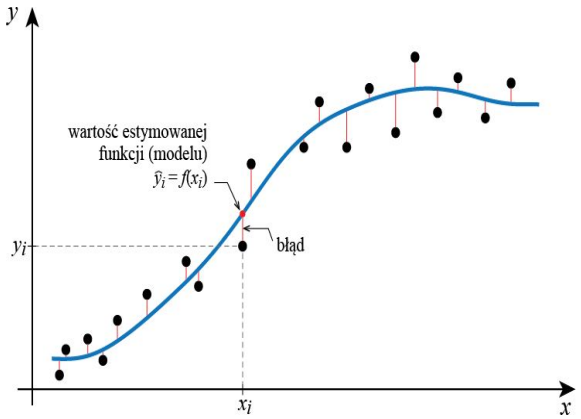
Metryki między obserwacjami

W zagadnieniach klasyfikacji, regresji i klasteryzacji używa się różnych metryk. Stosując różne metryki można uzyskać inne (lepsze/gorsze) wyniki dla danego modelu o identycznych danych.

ML Wstęp, Regresje

Regresja

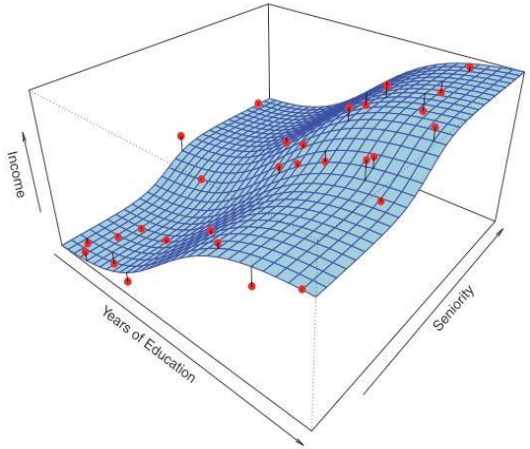
Błąd w regresji.



ML Wstęp, Regresje

Błąd w regresji

Błąd w regresji.



ML Wstęp, Regresje

Regresja - przykładowy wynik

indeks	obserwacje (dane)	model (predykcja)	błąd
i	y_i	$\hat{y}_i$	$y_i - \hat{y}_i$
1	2,4	2,5	-0,1
2	25,3	27,5	-2,2
3	4,5	3,8	0,7
...	...	...	...
1024	12,1	11,8	0,3

ML Wstęp, Regresje

Podstawowe metryki

Podstawowymi metrykami dopasowania modelu są:

- średni błąd kwadratowy (mean square error)

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

ML Wstęp, Regresje

Podstawowe metryki

Podstawowymi metrykami dopasowania modelu są:

- pierwiastek MSE (root mean square error)

$$RMSE = \sqrt{MSE}$$

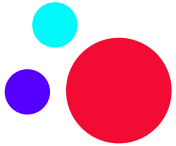
ML Wstęp, Regresje

Podstawowe metryki

Podstawowymi metrykami dopasowania modelu są:

- średni błąd bezwzględny (mean absolute error)

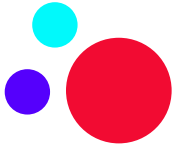
$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i |$$



ML Wstęp, Regresje

Klasyfikacja - przykładowy wynik

indeks	obserwacje (dane)	model (predykcja)
i	y_i	$\hat{y}_i$
1	klasa 1	klasa 1
2	klasa 2	klasa 3
3	klasa 1	klasa 1
...	...	...
256	klasa 3	klasa 3



ML Wstęp, Regresje

Klasyfikacja - przykładowy wynik

Przykładowy wynik klasyfikacji binarnej – tablica pomyłek (ang. confusion matrix):

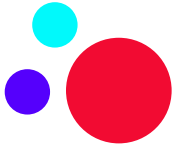
		obserwacje (dane)	
		positive	negative
predykcja (model)	positive	**true positive (TP)**	**false positive (FP)**
	negative	**false negative (FN)**	**true negative (TN)**

ML Wstęp, Regresje

Podstawowe metryki

Podstawowymi metrykami efektywności modelu klasyfikacji binarnej są:

- Accuracy
- Precision
- Recall
- F1-score



ML Wstęp, Regresje

Podstawowe metryki

Accuracy porównuje liczbę poprawnie sklasyfikowanych obserwacji do liczby wszystkich obserwacji.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

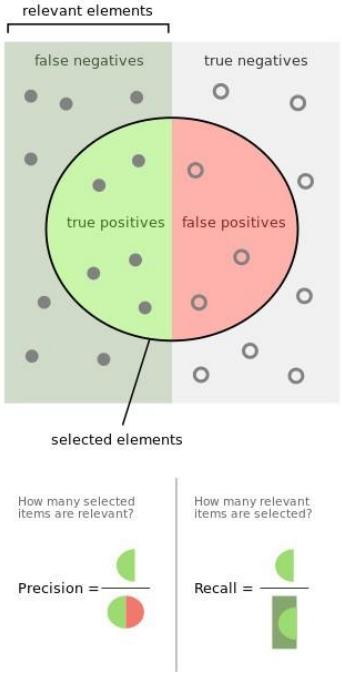
ML Wstęp, Regresje

Metryki klasyfikacji

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

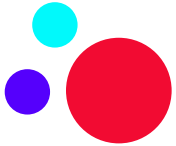
$$F_1 = 2\frac{Precision \cdot Recall}{Precision + Recall}$$



ML Wstęp, Regresje

Krzywa ROC

Receiver operating curve (ROC) – krzywa prezentująca skuteczność klasyfikatora binarnego przy różnych poziomach odcinania (threshold).



ML Wstęp, Regresje

Krzywa ROC

Na osi poziomej odznacza się False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + TN}$$

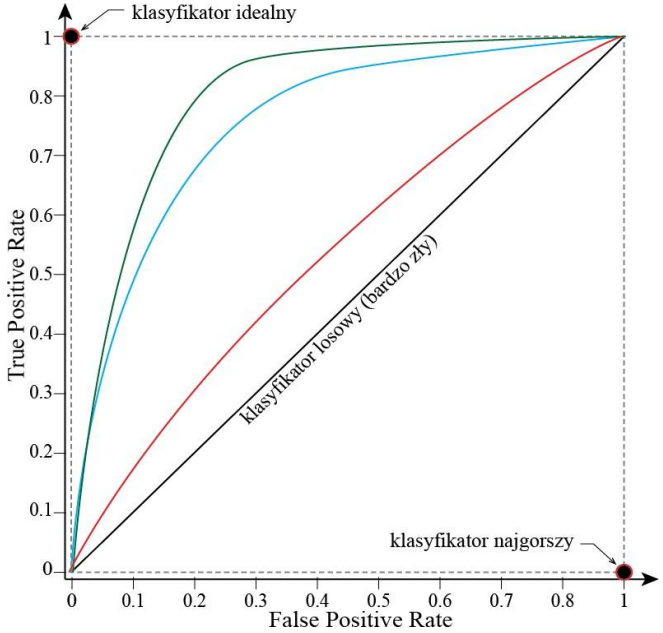
Na osi pionowej odznacza się True Positive Rate (TPR):

$$TPR = \frac{TP}{TP + FN}$$

Istotne jest pole pod krzywą ROC czyli tzw. AUC, dla klasyfikatora idealnego AUC=1.

ML Wstęp, Regresje

Krzywa ROC



ML Wstęp, Regresje

Klasyfikacja - podstawowe metryki

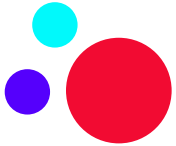
Przykładowy wynik klasyfikacji wieloklasowej: Macierz pomyłek 4x4 przy klasyfikacji wieloklasowej (4 klasy).

		obserwacje (dane)			
model (predykcja)	klasa	1	2	3	4
	1	**153**	**34**	**36**	**36**
	2	**33**	**155**	**25**	**21**
	3	**26**	**22**	**155**	**33**
	4	**28**	**30**	**38**	**175**

ML Wstęp, Regresje

Trenowanie modelu

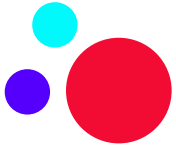
- Dostępne dane dzieli się na dwa lub trzy zbiory: treningowy i testowy (np. 8:2) lub treningowy, walidacyjny i testowy (np. 6:2:2 , 7:1,5:1,5, 8:1:1).
- Zbiór testowy nie powinien zawierać obserwacji, które znajdują się w zbiorze treningowym.



ML Wstęp, Regresje

Trenowanie modelu

- Dopasowanie modelu do danych wykonuje się na zbiorze treningowym.
- Zbiór walidacyjny wykorzystuje się do obiektywnego określenia efektywności modelu w trakcie jego uczenia. Jest to zbiór, dla którego określa się wartość metryki w trakcie procesu uczenia.
- Diagnostykę modelu przeprowadza się dla zbioru testowego.



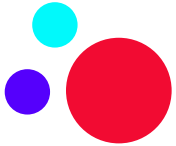
ML Wstęp, Regresje

Model - definicja

Model jest odwzorowaniem X→Y otrzymanym na podstawie danych.
Jest to estymator $f(x)$ odwzorowania $f(x)$ dla, którego zachodzi:

$$Y = f(X) + \varepsilon$$

X – predykatory (wejście, zmienne objaśniające, zmienne niezależne),
Y – odpowiedź (wyjście, zmienna objaśniana, zmienna zależna),
ε – oznacza błąd losowy.



ML Wstęp, Regresje

Obciążenie modelu - bias

Obciążenie modelu oznacza różnicę (błąd) między wartością oczekiwaną predykcji $E\,f(x)$, a prawdziwą (nieznaną) wartością funkcji $f(x)$.

$$Bias\ \ f(x) = E\,f(x) - f(x)$$

W praktyce obciążenie modelu oznacza, że model zawsze dokonuje predykcji z nadmiarem (niedomiarem).

Obciążenie powinno być jak najmniejsze.

ML Wstęp, Regresje

Wariancja modelu - variance

Model jest odwzorowaniem X→Y otrzymanym na podstawie danych. Jest to estymator $f(x)$ odwzorowania $f(x)$ dla, którego zachodzi:

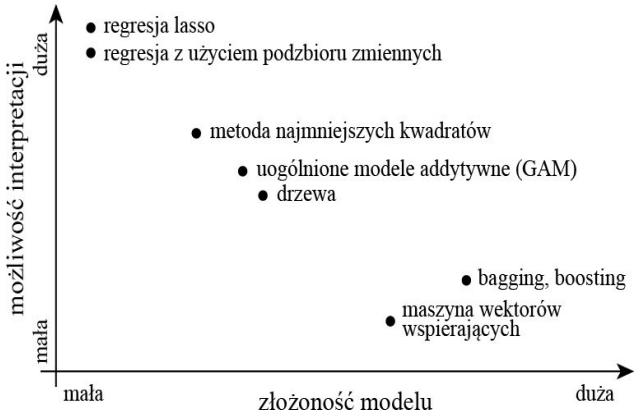
$$Var(f(x)) = E\left[(f(x) - E[f(x)])^2\right]$$

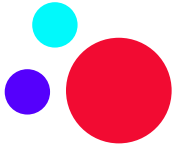
Duża wariancja oznacza, że predykcje dla nieodległych od siebie wartości x będą mocno zróżnicowane, "rozstrzelone".

Wariancja powinna być jak najmniejsza.

ML Wstęp, Regresje

Własności modelu vs interpretacja





ML Wstęp, Regresje

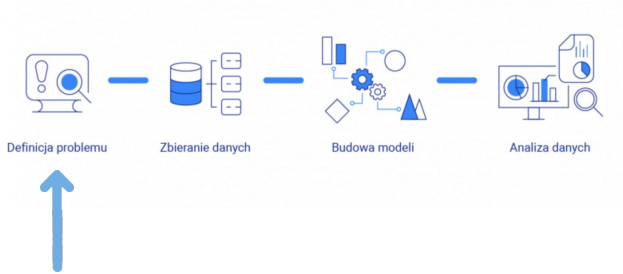
Workflow pracy z modelami

1. Analiza problemu.
2. Analiza i eksploracja danych.
3. Inżynieria cech (ang. feature engineering).
4. Wybór modelu.
5. Wybór metryk.
6. Trenowanie modelu.
7. Testowanie.
8. Analiza wyników.

ML Wstęp, Regresje

Workflow pracy z modelami

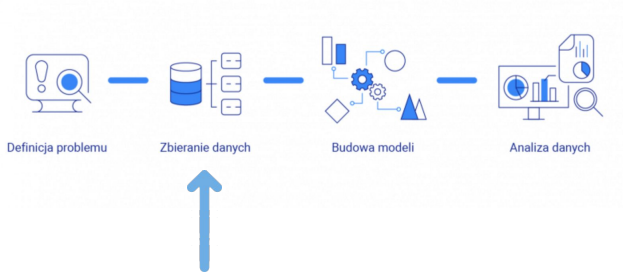
1. Analiza problemu, który ma zostać rozwiązany przy wykorzystaniu danych (np. oszacowanie liczby klientów w danym dniu roku, segmentacja klientów, analiza powiązań) i wybór odpowiedniej metody uczenia maszynowego do rozwiązania danego zagadnienia.



ML Wstęp, Regresje

Workflow pracy z modelami

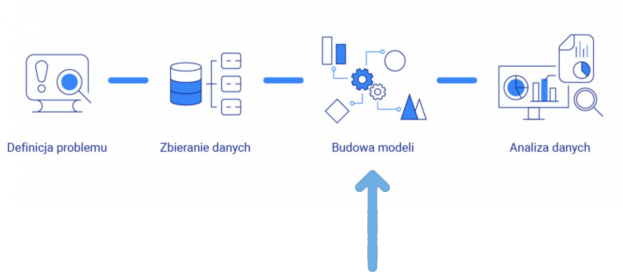
2. Analiza i eksploracja danych w celu wyboru modelu i możliwie najlepszego przygotowania danych (np. usuwanie obserwacji odstających, usuwanie danych skorelowanych, itp.).



ML Wstęp, Regresje

Workflow pracy z modelami

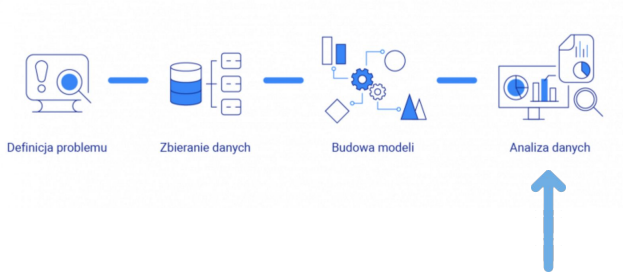
3. Inżynieria cech (ang. feature engineering) – wprowadzenie dodatkowych danych (np. one-hot encoding, flagi klasteryzacji), redukcja wymiarowości itp.



ML Wstęp, Regresje

Workflow pracy z modelami

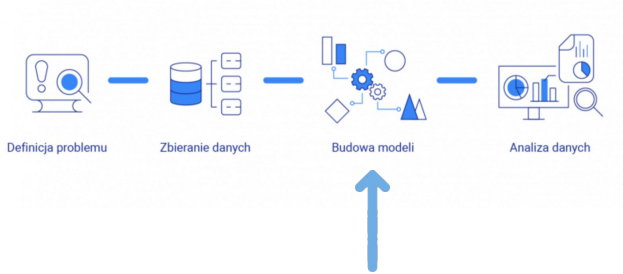
4. Wybór modelu będzie zależał od charakteru danych i celu jaki należy osiągnąć. W niektórych zagadnieniach akceptowalna będzie niższa skuteczność modelu, jeśli model będzie oferował możliwość interpretacji.



ML Wstęp, Regresje

Workflow pracy z modelami

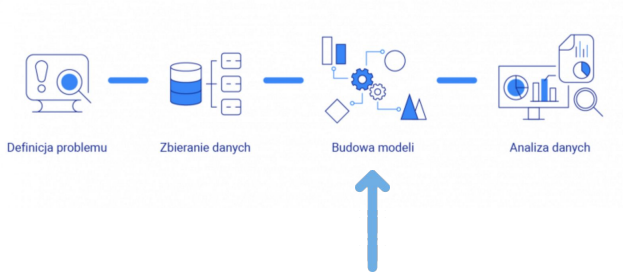
5. Wybór metryk adekwatnych do modelu i zagadnienia. Błąd modelu można mierzyć na różne sposoby, ważne jest aby dokonać właściwego wyboru metryki. ogólnie rzecz biorąc nie ma metryk lepszych i gorszych, wszystko zależy od konkretnego problemu.



ML Wstęp, Regresje

Workflow pracy z modelami

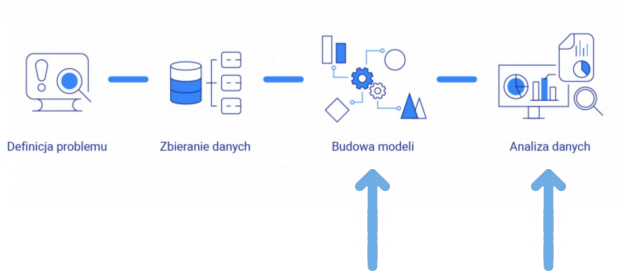
6. Trenowanie modelu. Polega na określeniu postaci estymatora (np. parametrów modelu) na podstawie danych.



ML Wstęp, Regresje

Workflow pracy z modelami

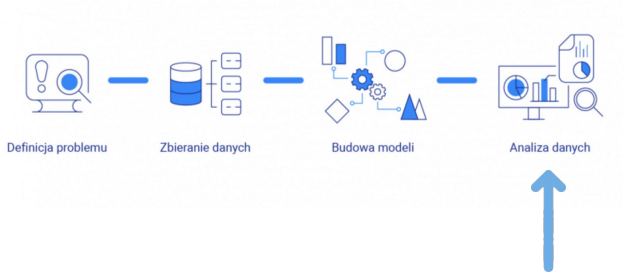
7. Testowanie modelu. Polega na sprawdzeniu efektywności modelu na danych, które nie zostały wykorzystane do uczenia (trenowania).



ML Wstęp, Regresje

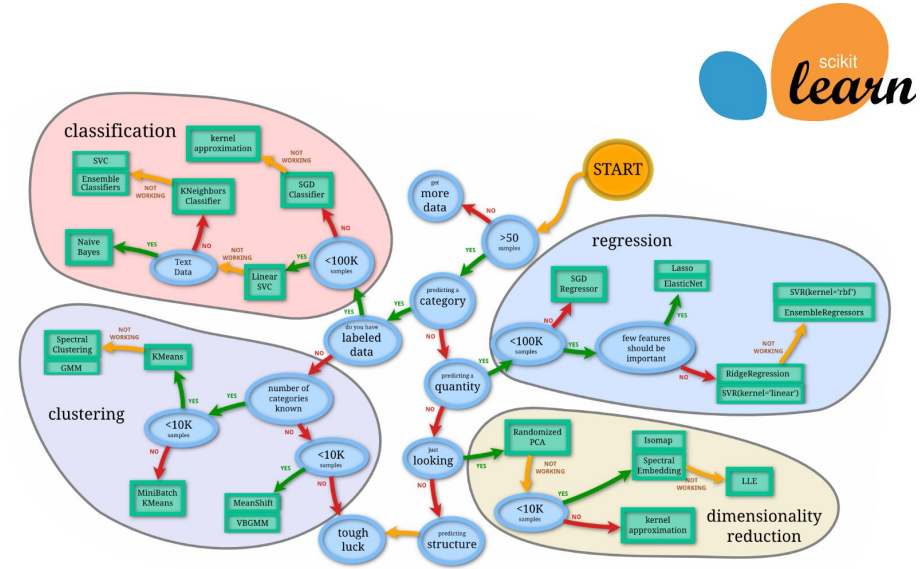
Workflow pracy z modelami

8. Analiza wyników. Stwierdzenie czy model jest skuteczny (tj. pozwala na osiągnięcie założonego na początku celu), sprawdzenie czy model nie jest obciążony (ang. bias) oraz czy nie jest nadmiernie dopasowany (ang. overfitting).



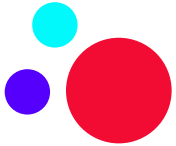
ML Wstęp, Regresje

scikit-learn w API – uczenie maszynowe w Pythonie



infoShareAcademy.com

infoShare ACADEMY



ML Wstęp, Regresje

Workflow pracy z modelami

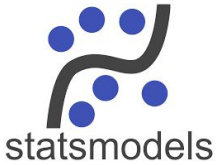
Scikit-learn oferuje bardzo spójne API. Przepływ pracy jest następujący:

1. Utworzenie obiektów (encoder, scaler, classifier, regressor, itp.).
2. Użycie metody fit z argumentem w postaci danych.
3. Użycie metody transform lub predict w celu obliczenia transformacji lub predykcji.
4. (użycie pipeline)

ML Wstęp, Regresje

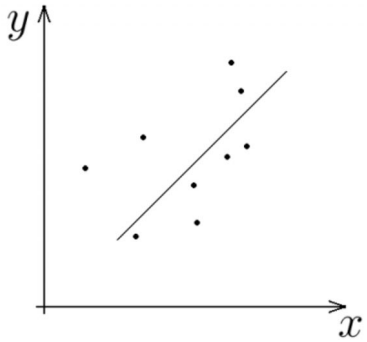
Biblioteki ML Python

- scikit-learn, statsmodels, XGBoost, uMAP
- TPOT
- pyCARET



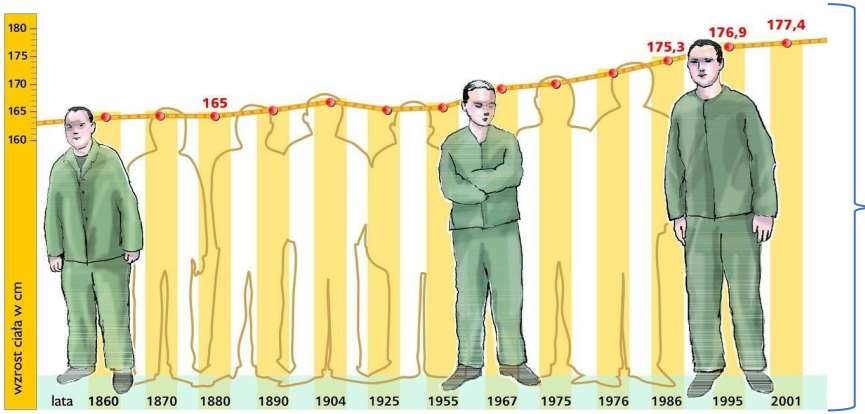
ML Wstęp, Regresje

Regresja liniowa



ML Wstęp, Regresje

Regresja liniowa – wzrost społeczeństwa



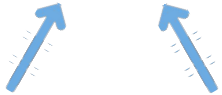
185 cm

2030 rok

ML Wstęp, Regresje

Czym jest funkcja?

$$y = f(x)$$



Zmienna zależna

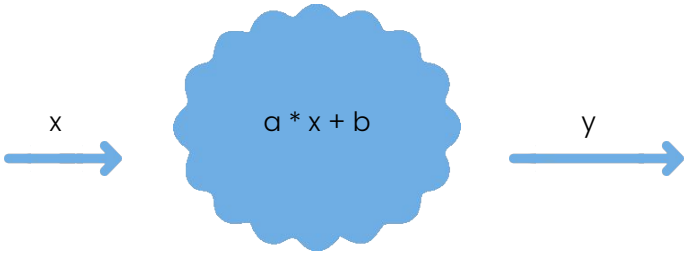
Zmienna niezależna

ML Wstęp, Regresje

Funkcja

ML Wstęp, Regresje

Funkcja liniowa



X – wejście funkcji, zmienna objaśniająca, niezależna

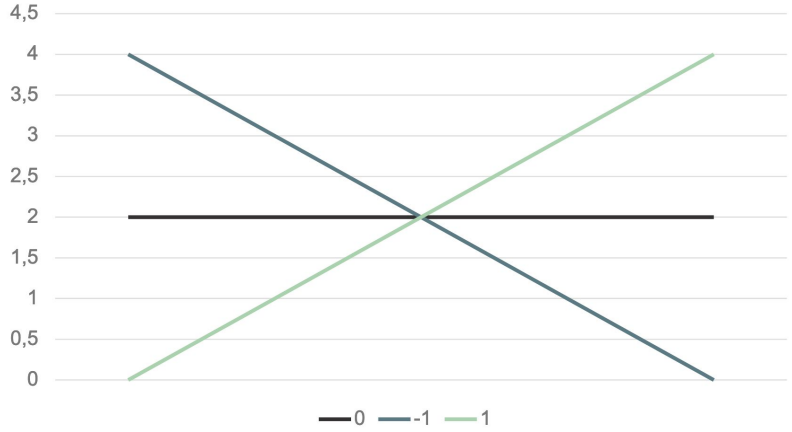
Y – odpowiedź funkcji, zmienna objaśniana, zależna

a – współczynnik ‚slope', kierunkowy

b – współczynnik ‚intercept', przesunięcie

ML Wstęp, Regresje

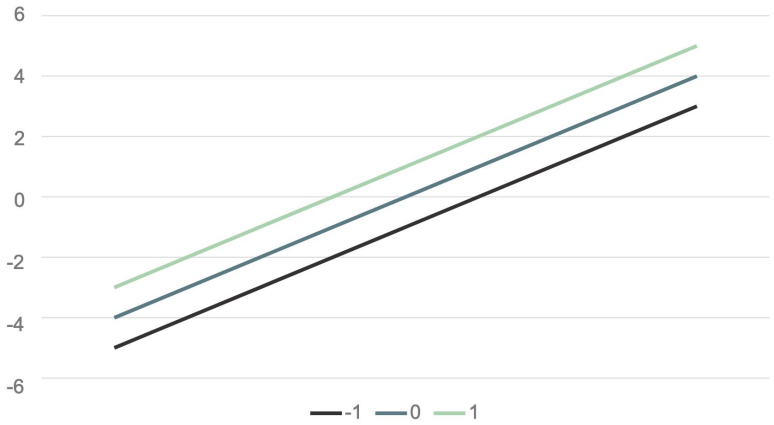
Funkcja liniowa



Wykres funkcji liniowej zależny od współczynnika a.

ML Wstęp, Regresje

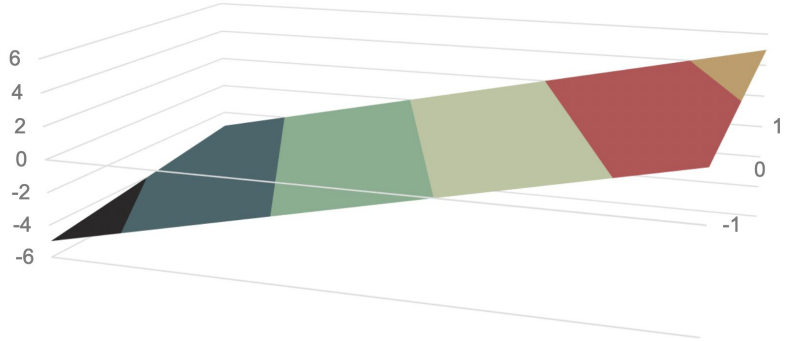
Funkcja liniowa



Wykres funkcji liniowej zależny od współczynnika b.

ML Wstęp, Regresje

Funkcja liniowa wielu zmiennych - hiperpłaszczyzna



infoShareAcademy.com

Zadanie 12.4 (instrukcja)

1. Napisz funkcję, która wyliczać będzie odpowiedź funkcji liniowej na podstawie wejścia oraz dwóch współczynników.
2. Narysuj wykres funkcji liniowej na podstawie danych syntetycznych.
3. Stwórz wykres nakładający na siebie kilka funkcji liniowych o różnych współczynnikach slope.
4. Stwórz wykres nakładający na siebie kilka funkcji liniowych o różnych współczynnikach intercept.
5. Odległości między dwiema funkcjami liniowymi o różnych współczynnikach slope.

ML Wstęp, Regresje

Funkcja liniowa

Tworzenie na podstawie dwóch punktów, A i B:

$$a = \frac{y_B - y_A}{x_B - x_A}$$

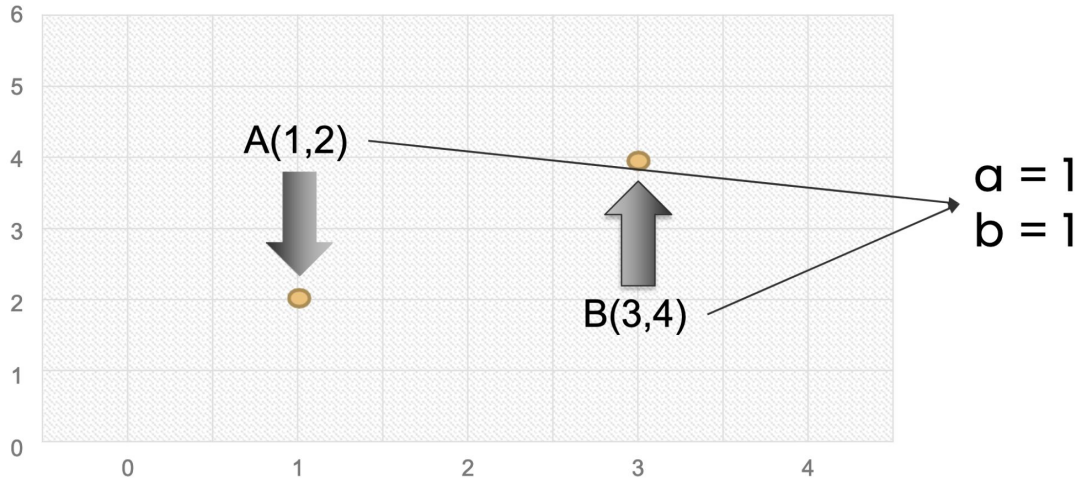
$$\mathrm{b} = y_A - a x_A$$

xA, xB, yA, yB - wartości współrzędnych x i y punktu A i B

ML Wstęp, Regresje

Funkcja liniowa

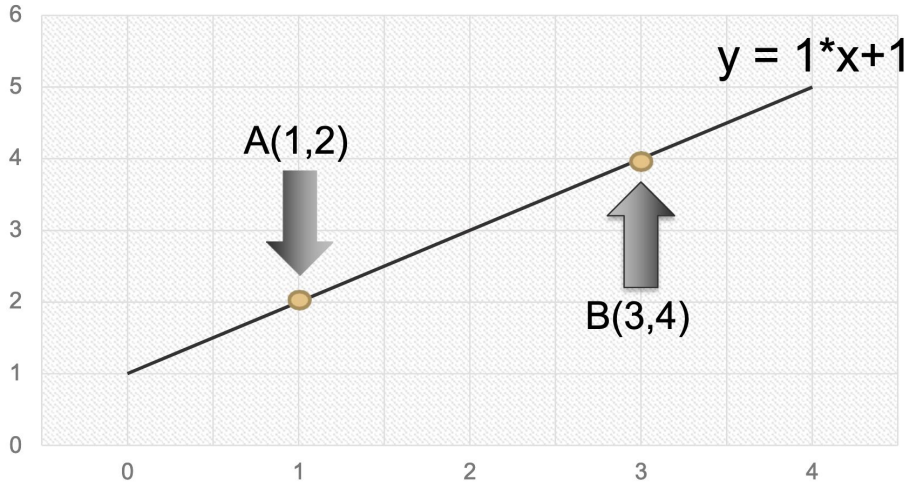
Wyznaczanie funkcji liniowej na podstawie dwóch punktów:



ML Wstęp, Regresje

Funkcja liniowa

Wyznaczanie funkcji liniowej na podstawie dwóch punktów:



ML Wstęp, Regresje

Regresja liniowa (definicja)

- Przedstawia korelację dwóch zmiennych: x, y.
- X nazywamy zmienną objaśniającą.
- Y nazywamy zmienną objaśnianą.
- Modelem jest funkcja liniowa: y = ax + b.
- Wyznaczenie funkcji liniowej polega na minimalizacji wartości błędu określanej metodą najmniejszych kwadratów (y - _y)^2.

ML Wstęp, Regresje

Regresja liniowa (implementacja)

$$a = \frac{\sum_{k=1}^{n}(x_i - x_{mean})(y_i - y_{mean})}{\sum_{k=1}^{n}(x_i - x_{mean})^2}$$

$$b = y_{mean} - a x_{mean}$$

n – ilość punktów do treningu

yi , xi - wartość współrzędnych kolejnych punktów

$xmean$, $ymean$ - wartość średnia współrzędnych ze wszystkich punktów

ML Wstęp, Regresje

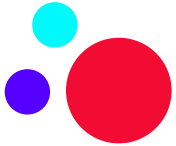
Regresja liniowa (współczynnik determinacji)

$$R^2 = \frac{\sum_{i=1}^{n} (y_i^* - y_{mean})^2}{\sum_{i=1}^{n} (y_i - y_{mean})^2}$$

y_i^* - kolejna wartość predykcji

y_i - kolejna wartość referencyjna

y_{mean} - wartość średnia ze wszystkich referencji



ML Wstęp, Regresje

Regresja liniowa (metryki reszty)

- Wielkości odchylenia odpowiedzi modelu od danych referencyjnych.
- Pokazują jak mocno na przestrzeni danych nasz model odbiega od rzeczywistości.
- Jeśli wartości reszt są mniej więcej podobne dla większości punktów, oznacza to dobre zastosowanie regresji liniowej.

Zadanie 12.5 (instrukcja)

1. Na początku wczytaj dane height.csv.

2. Narysuj wykres wysokości w czasie.

3. Z pierwszego i ostatniego punktu wyznacz równanie prostej na podstawie poniższych wzorów:

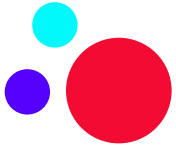
$$a = \frac{y_B - y_A}{x_B - x_A}$$

$$b = y_A - a x_A$$

A,B - skrajne punkty o współrzędnych (x,y)

a - slope

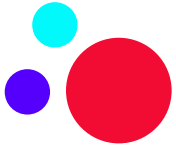
b - intercept



ML Wstęp, Regresje

Regresja liniowa (błąd średniokwadratowy)

- Stosowany do porównywania jakości modeli regresji.
- Przedstawia średni błąd odchylenia wszystkich próbek testowanych od referencyjnych.
- W treningu regresji liniowej lepiej sprawdza się r^2, jednak jako dodatek również można go zastosować.



ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn)

- **Pobieramy pakiet modeli liniowych:**

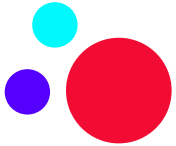
  from sklearn.linear_model import LinearRegression
- **Tworzymy model regresji liniowej:**

  model = LinearRegression()
- **Trening:**

  model.fit(???)
- **Współczynniki:**

  model.intercept_, model.coef_
- **Predykcja:**

  model.predict(???)



ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn)


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

Modelowanie regresji liniowej w sklearn

```
df = pd.read_csv("data/salary.csv")
df.head()
```

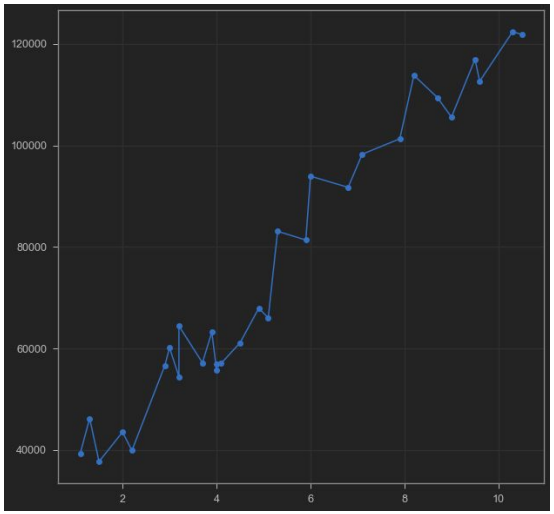
	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn)

```
x = df.YearsExperience.to_numpy()
y = df.Salary.to_numpy()
```

```
fig = plt.figure(figsize=(10,10))
plt.plot(x,y,marker='o')
plt.grid()
plt.show()
```



ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn)

Modelowanie

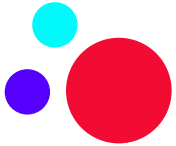
```
model = LinearRegression()

x = x.reshape(-1, 1)
y = y.reshape(-1, 1)

model.fit(x,y)

model.score(x,y)
```

R2 = 0.9569



ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn)

Rysowanie wykresu regresji liniowej:

model.intercept_

array([25792.20019867]) → b

model.coef_

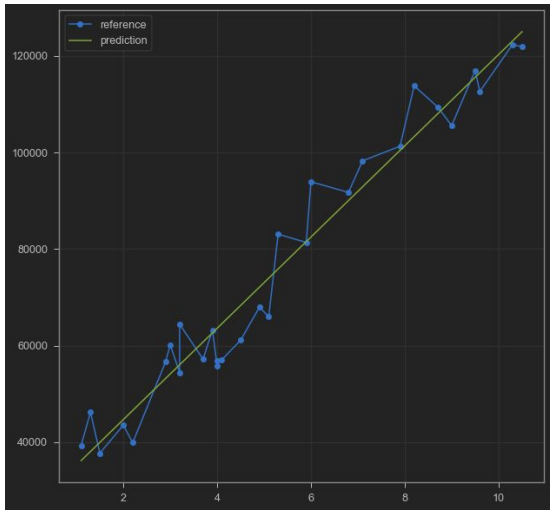
array([[9449.96232146]]) → b

ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn)

```
def linear_function(a,b,x):
    """
    nasza funkcja liniowa
    """
    return a*x+b
```

```
y_pred = linear_function(a,b,x)
```



ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn) – r kwadrat

```
from sklearn.metrics import r2_score

print("R^2 dla modelu liniowego:\n", r2_score(y,y_pred))
```

R^2 dla modelu liniowego:

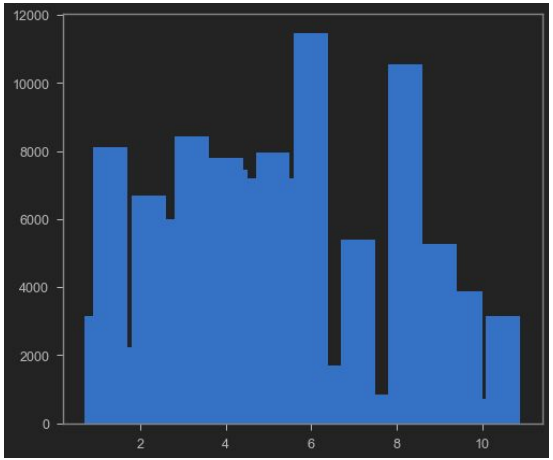
0.9569566641435086

ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn) – wykres reszt

yerr = y - y_pred

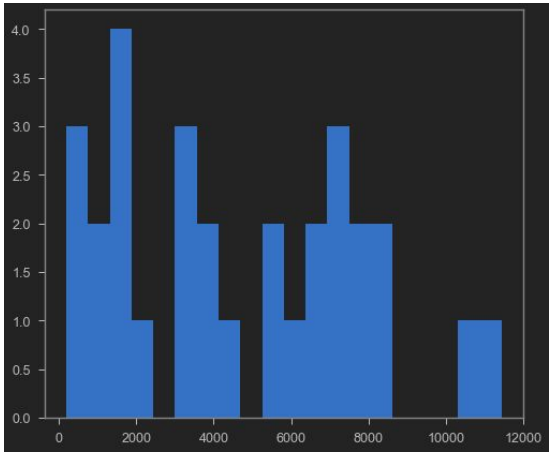
plt.bar(x.reshape(-1),np.abs(yerr.reshape(-1)))
plt.show()



ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn) – wykres reszt

```
plt.hist(np.abs(yerr.reshape(-1)),bins=20)
plt.show()
```

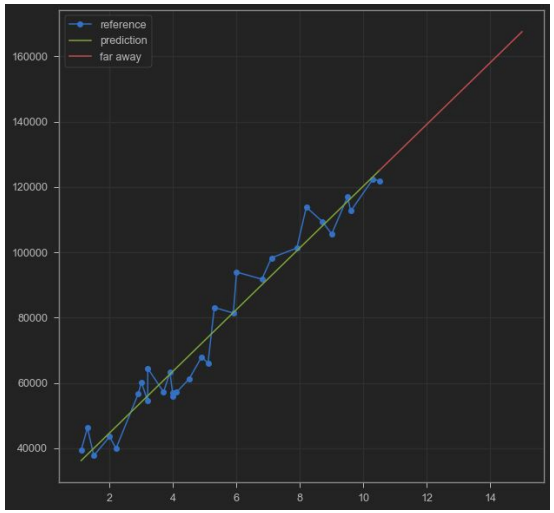


ML Wstęp, Regresje

Regresja liniowa (implementacja w sklearn)

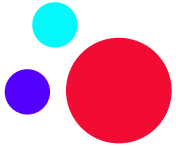
```
x_tmp = np.array([x[-1][0],15]).reshape(-1,1)
y_far_away = model.predict(x_tmp)
y_far_away
```

```
array([[125016.80457395],
       [167541.63502049]])
```



Zadanie 12.6 (instrukcja)

Wyznacz model regresji liniowej dla zestawienia "Y house price of unit area" od "X2 house age" w datasecie "estate.csv". Zbadaj poziom korelacji za pomocą metryki r2.



ML Wstęp, Regresje

Podsumowanie

infoShare ACADEMY