ML xgBoost

ML xgBoost

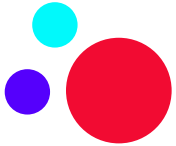
Agenda

1. **Redukcja wymiarowości (PCA, t-SNE)**
2. Gradient Boosting
3. xgBoost
4. Regularyzacja

ML xgBoost

Agenda

1. Redukcja wymiarowości (PCA, t-SNE)
2. **Gradient Boosting**
3. xgBoost
4. Regularyzacja



ML xgBoost

Agenda

1. Redukcja wymiarowości (PCA, t-SNE)
2. Gradient Boosting
3. **xgBoost**
4. Regularyzacja

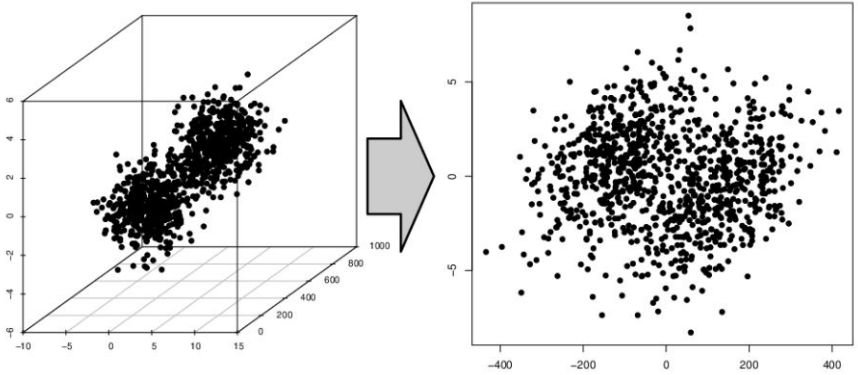
ML xgBoost

Agenda

1. Redukcja wymiarowości (PCA, t-SNE)
2. Gradient Boosting
3. xgBoost
4. **Regularyzacja**

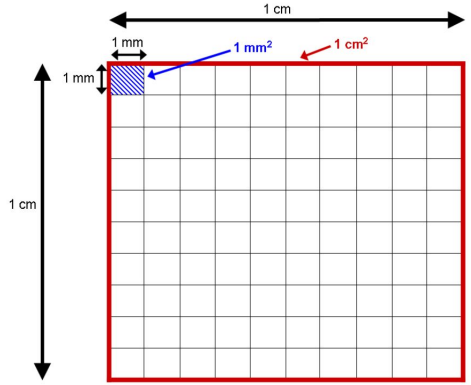
ML xgBoost

Problem wymiarowości w analizie danych



ML xgBoost

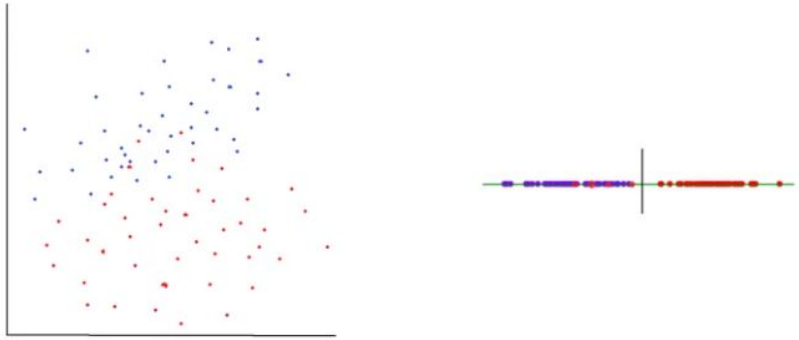
Przekleństwo wymiarowości



Im większy wymiar, tym znacznie więcej danych potrzebujemy. Tym samym wykładniczo rośnie liczba możliwych wariantów, co znacznie zwiększa złożoność obliczeniową naszych algorytmów.

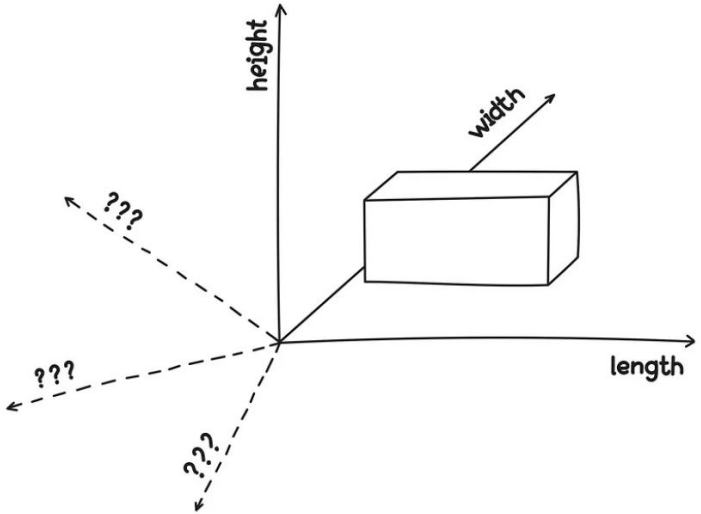
ML xgBoost

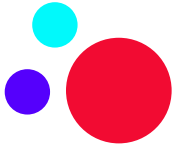
Redukcja wymiarowości



ML xgBoost

Wielowymiarowość

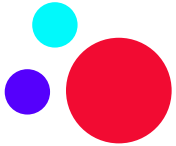




ML xgBoost

Zalety redukcji wymiarów

- Zmniejszenie rozmiaru danych, co przekłada się na wydajność algorytmów uczenia maszynowego,
- łatwiejsze zidentyfikowanie podstawowej struktury danych,
- zapobieganie przetrenowaniu modelu.



ML xgBoost

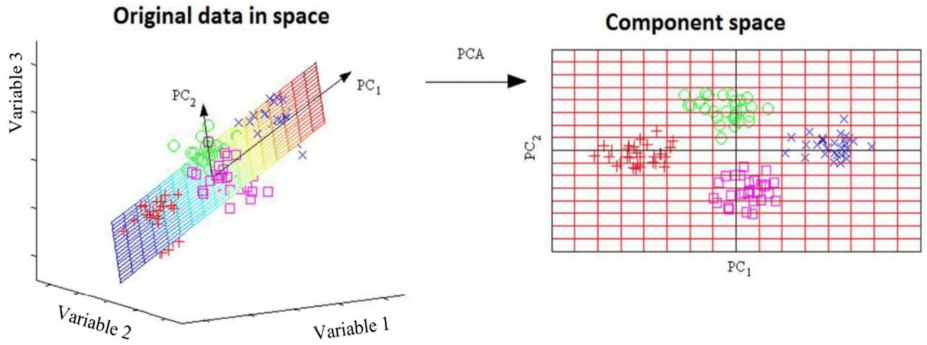
Techniki redukcji wymiarowości

- Metody w oparciu o szukanie czynników pomiędzy wymiarami,
- metody w oparciu o rzutowanie jednego wymiaru na inny.

ML xgBoost

Metody w oparciu o szukanie czynników pomiędzy wymiarami

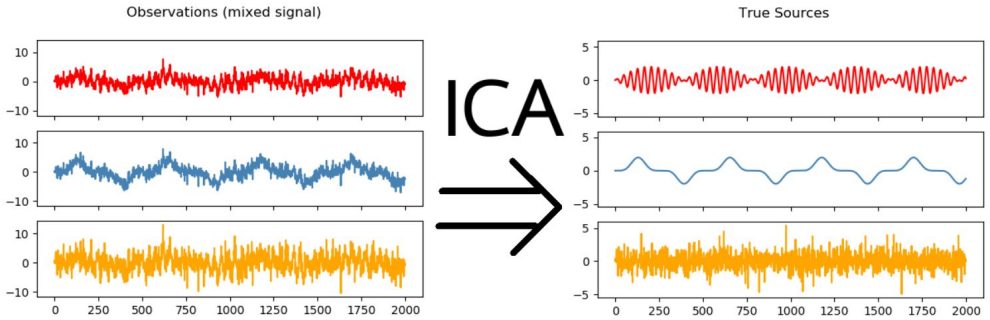
- Principal Component Analysis (PCA)



ML xgBoost

Metody w oparciu o szukanie czynników pomiędzy wymiarami

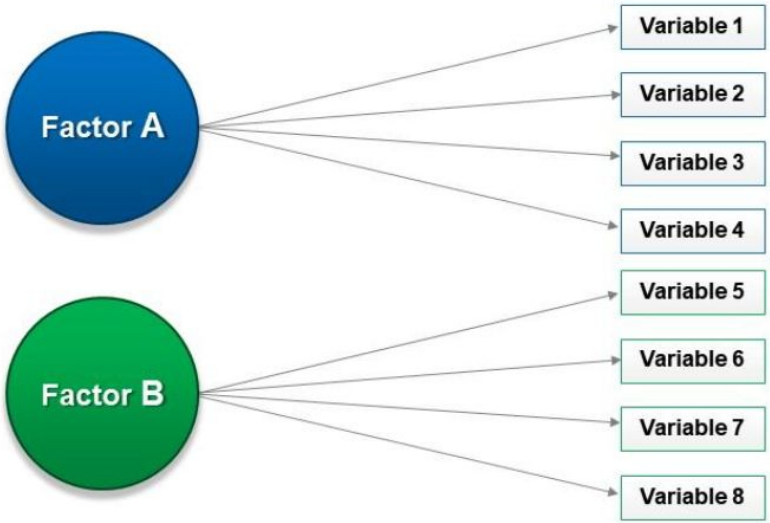
- Independent Component Analysis (ICA)



ML xgBoost

Metody w oparciu o szukanie czynników pomiędzy wymiarami

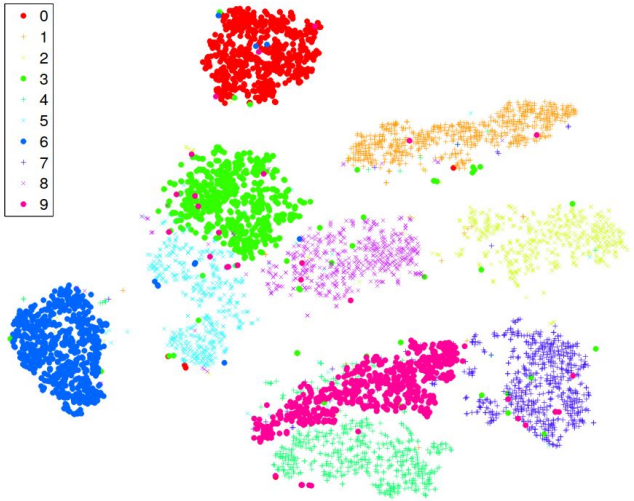
- Factor Analysis



ML xgBoost

Metody w oparciu o szukanie czynników pomiędzy wymiarami

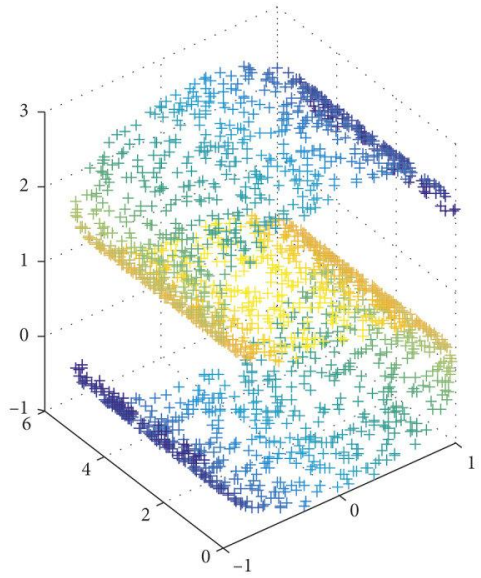
- t-SNE



ML xgBoost

Metody w oparciu o rzutowanie jednego wymiaru na inny

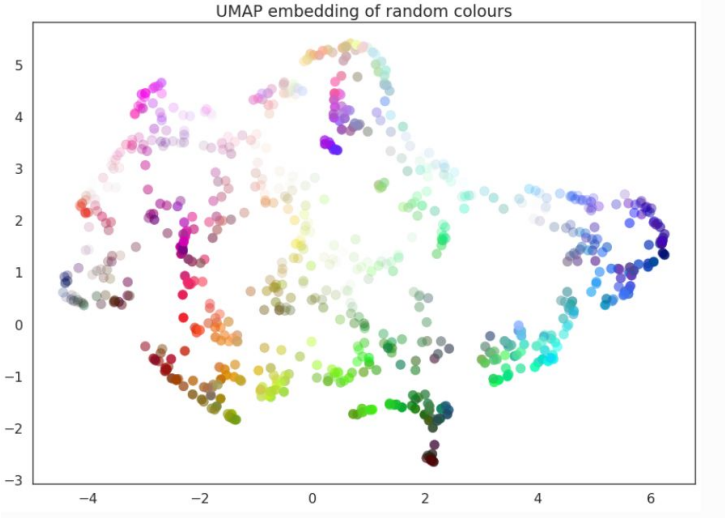
- ISOMAP



ML xgBoost

Metody w oparciu o rzutowanie jednego wymiaru na inny

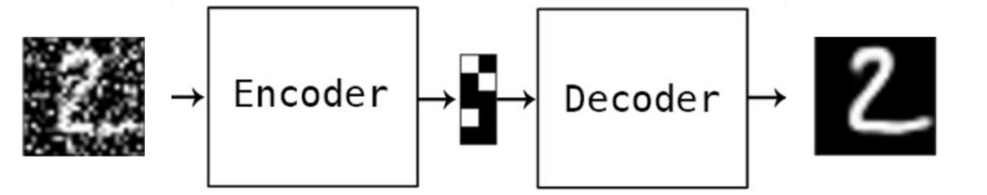
- UMAP



ML xgBoost

Metody w oparciu o rzutowanie jednego wymiaru na inny

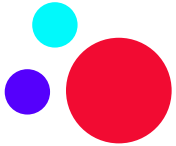
- UMAP



Noisiy input

Compressed representation

Denoised image



ML xgBoost

PCA

Principal

Components

Analysis

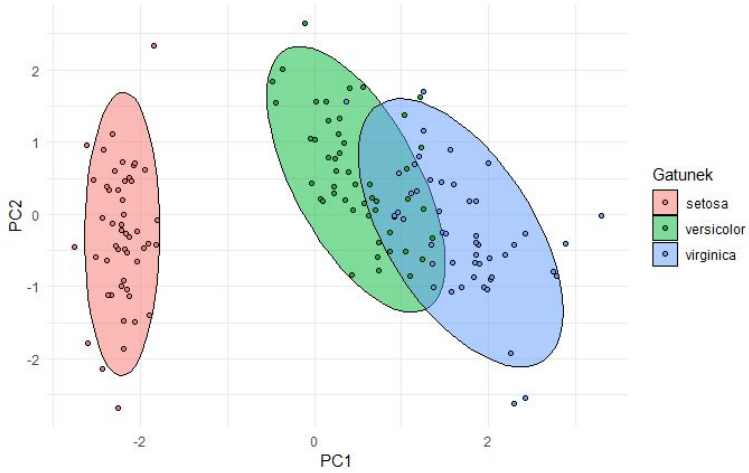
ML xgBoost

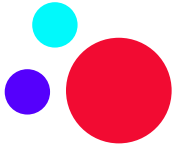
PCA – zastosowanie

1. Redukcja wymiarowości danych
2. Analiza wizualna i wizualizacja danych
3. Preprocessing danych przed modelem
4. Wykrywanie skorelowanych cech
5. Kompresja danych
6. Rozpoznawanie obrazów i analiza tekstur
7. Analiza genomiki i biologii molekularnej

ML xgBoost

PCA – składowe główne

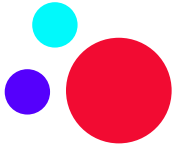




ML xgBoost

PCA – składowe główne

1. Nieskorelowane
2. Ułożone malejąco pod kątem zmienności
3. Wariancja jako maksymalizowana kryterium
4. Minimalna korelacja z poprzednimi składowymi



ML xgBoost

PCA - matematycznie

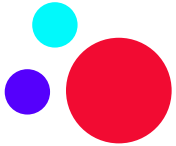
Jeśli X1,X2,...,Xp są oryginalnymi zmiennymi, a PC1,PC2,...,PCp są głównymi składowymi, to pierwsza główna składowa PC1 jest kombinacją liniową a1X1+a2X2+...+apXp, gdzie a1,a2,...,ap to współczynniki tak dobrane, aby maksymalizować wariancję PC1, przy czym a1^2+a2^2+...+ap^2=1.

ML xgBoost

Równania opisujące PCA

1. Średnia

$$\overline{X}_i = \frac{1}{n}\sum_{j=1}^{n} X_{ij}$$



ML xgBoost

Równania opisujące PCA

2. Standaryzacja

ML xgBoost

Równania opisujące PCA

3. Macierz kowariancji

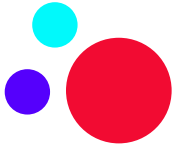
$$S = \frac{1}{n-1}\sum_{k=1}^{n}(X_k - \bar{X})(X_k - \bar{X})^T$$

ML xgBoost

Równania opisujące PCA

4. Wartości własne i wektory własne

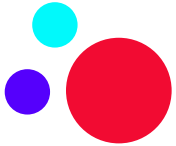
$$det(S - I\lambda) = 0$$



ML xgBoost

Równania opisujące PCA

5. Sortowanie wartości własnych



ML xgBoost

Równania opisujące PCA

6. Wybór głównych składowych

ML xgBoost

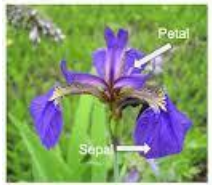
Równania opisujące PCA

7. Transformacja danych

ML xgBoost

Interpretacja wyników PCA

iris = datasets.load_iris()

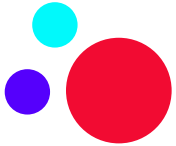


iris setosa

iris versicolor

iris virginica

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000



ML xgBoost

Interpretacja wyników PCA

iris = datasets.load_iris()

data = iris.data

columns = iris.feature_names

scaler = StandardScaler()

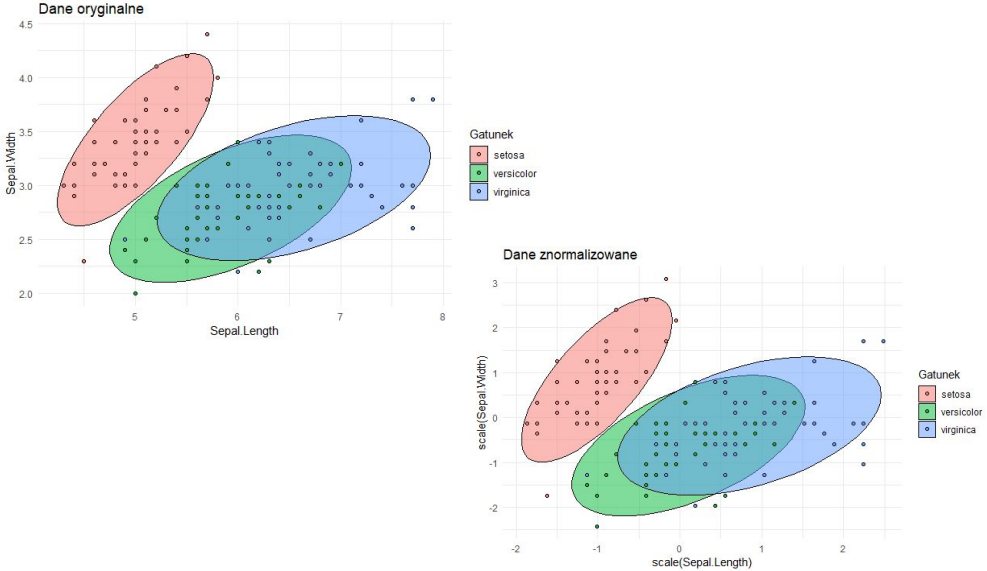
norm = scaler.fit_transform(df)

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444

ML xgBoost

Interpretacja wyników PCA



ML xgBoost

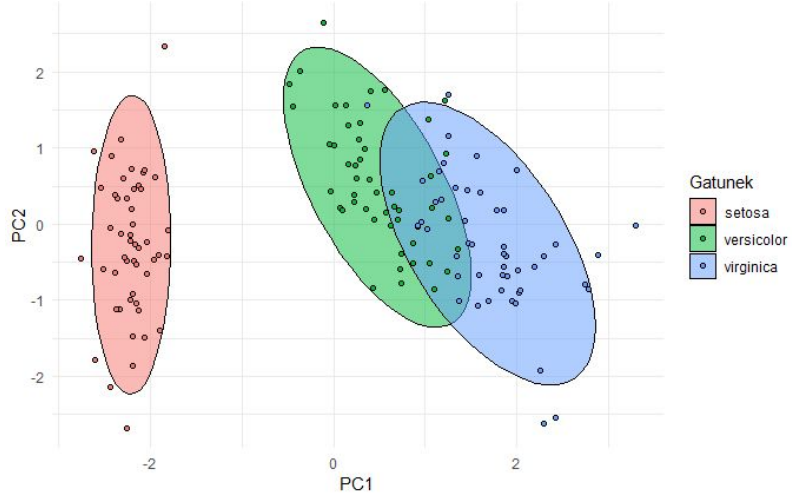
Interpretacja wyników PCA

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

principal_components = pca.fit_transform(normalized_df)

pc_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])
```

	PC1	PC2	Target
0	-2.264703	0.480027	0
1	-2.080961	-0.674134	0
2	-2.364229	-0.341908	0
3	-2.299384	-0.597395	0
4	-2.389842	0.646835	0

ML xgBoost

Interpretacja wyników PCA



infoShareAcademy.com

Zadanie 14.1 (instrukcja)

Przeprowadź Analizę Składowych Głównych (PCA) na zbiorze danych o winach.

```
from sklearn.datasets import load_wine
wine = load_wine()
```

ML xgBoost

Redukcja wymiarów a utrata informacji

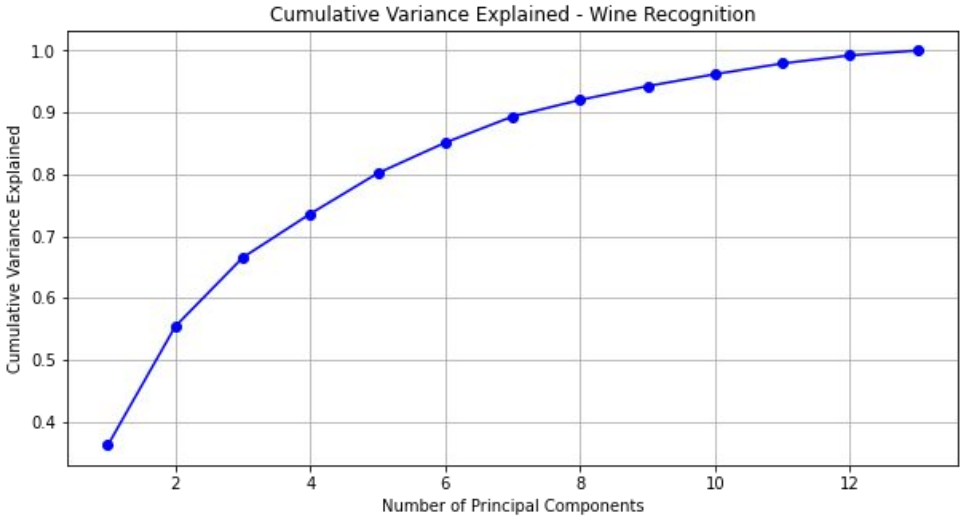
ML xgBoost

Interpretacja wyników PCA

```
def plot_information_loss(dataset, title):
    X = dataset.data
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    pca = PCA().fit(X_scaled)
    cumulative_variance = np.cumsum(pca.explained_variance_ratio_)

    plt.figure(figsize=(10, 5))
    plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance, marker='o', linestyle='-',
color='b')
    plt.title(f'Cumulative Variance Explained - {title}')
    plt.xlabel('Number of Principal Components')
    plt.ylabel('Cumulative Variance Explained')
    plt.grid(True)
    plt.show()

wine = load_wine()
plot_information_loss(wine, 'Wine Recognition')
```

ML xgBoost

Redukcja wymiarów a utrata informacji



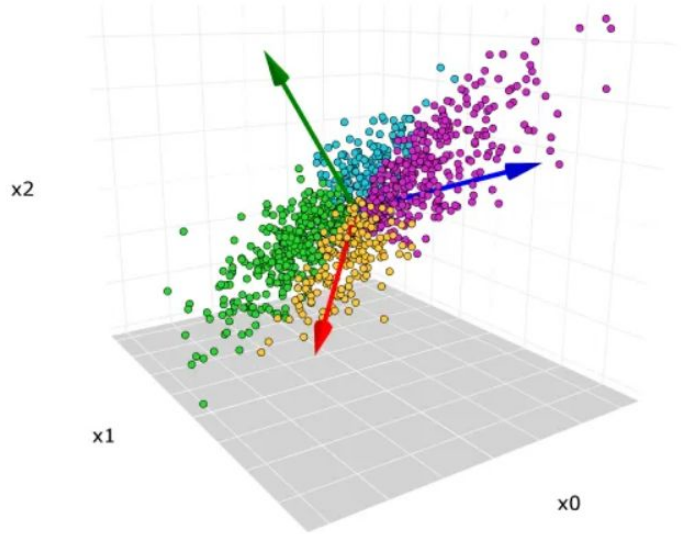
ML xgBoost

PCA – zastosowanie

1. Redukcja wymiarowości
2. Wizualizacja danych
3. Usuwanie korelacji
4. Przyśpieszenie algorytmów uczenia maszynowego
5. Analiza grup
6. Korekcja zakłóceń (denosing)
7. Ekstrakcja cech
8. Analiza genomiki i biologii
9. Kompresja danych

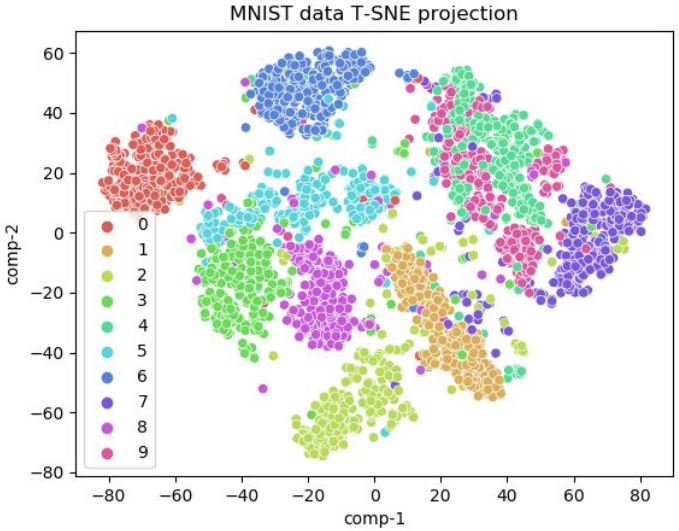
ML xgBoost

Podsumowanie



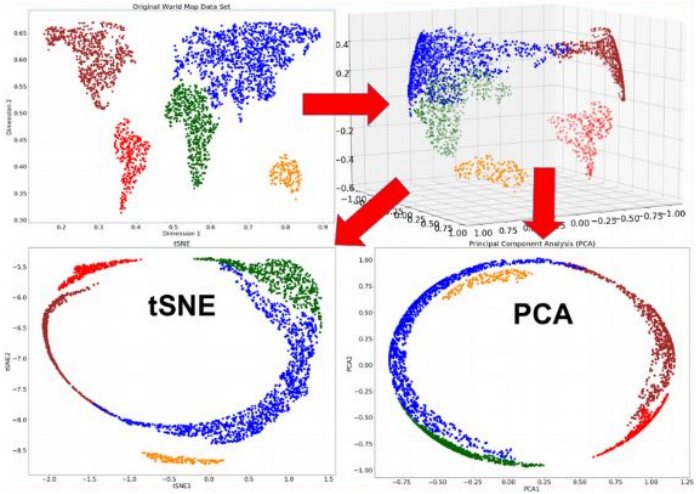
ML xgBoost

t-SNE



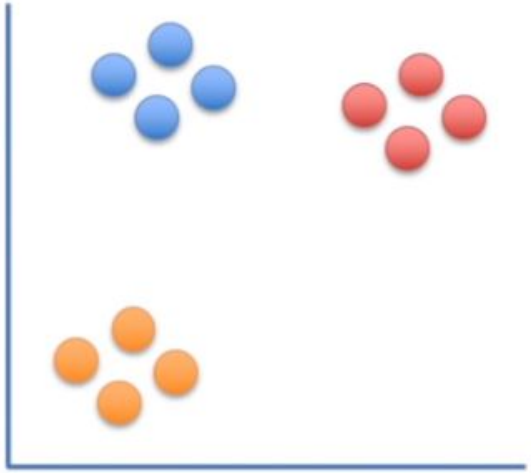
ML xgBoost

t-SNE czy PCA?



ML xgBoost

t-SNE – idea działania

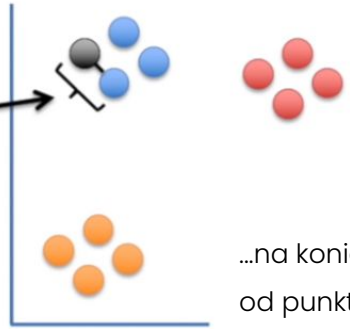


ML xgBoost

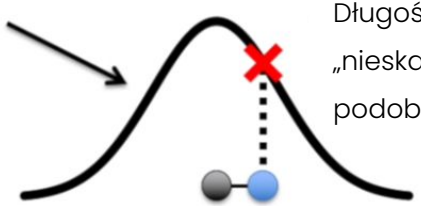
t-SNE – idea działania

Wyliczenie podobieństw:

Najpierw zmierz odległość między dwoma punktami...



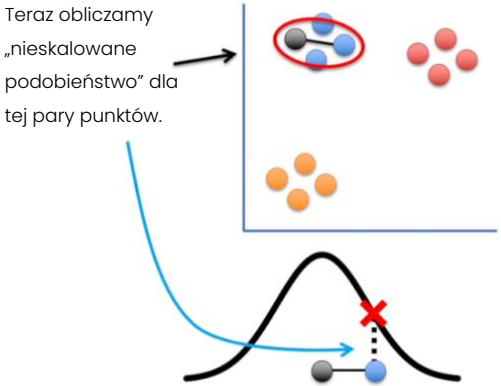
...następnie narysuj tę odległość na krzywej normalnej, której środkiem jest punkt zainteresowania...

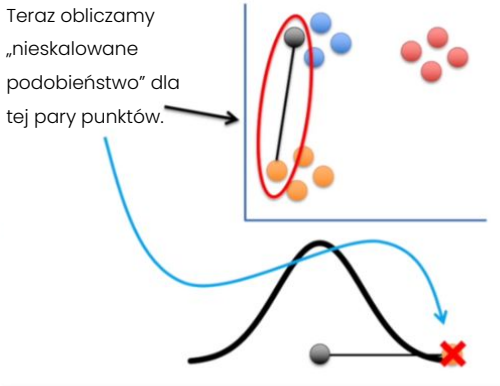


...na koniec narysuj linię od punktu do krzywej. Długość tej linii to „nieskalowane podobieństwo".

ML xgBoost

t-SNE – idea działania

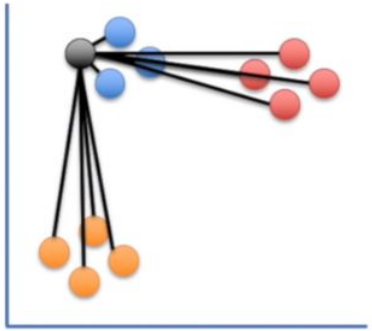




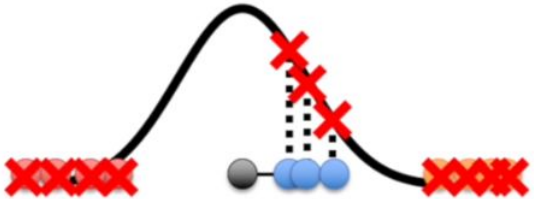
ML xgBoost

t-SNE – idea działania

Ostatecznie mierzymy odległości pomiędzy wszystkimi punktami a punktem zainteresowania...



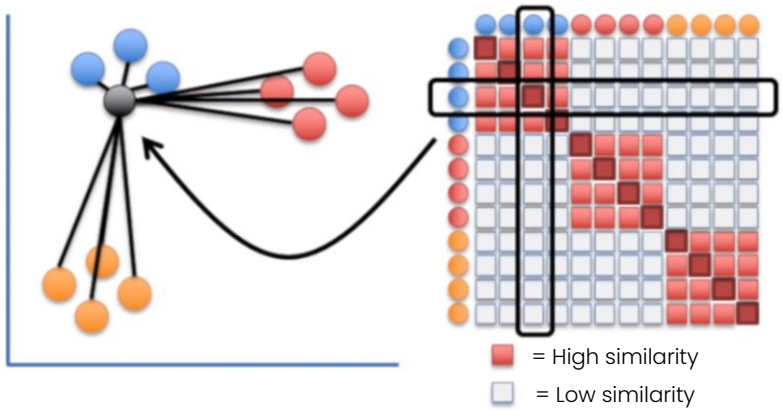
Narysuj je na krzywej normalnej...



...a następnie zmierzyć odległości od punktów do krzywej, aby uzyskać nieskalowane wyniki podobieństwa w odniesieniu do interesującego punktu.

ML xgBoost

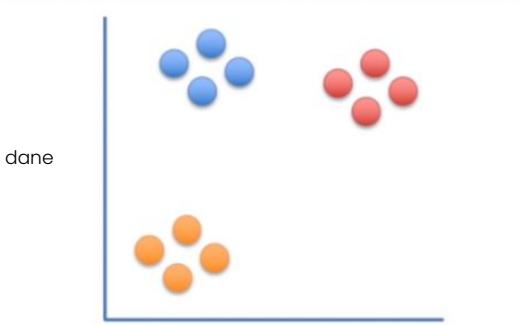
t-SNE – idea działania



ML xgBoost

t-SNE – idea działania

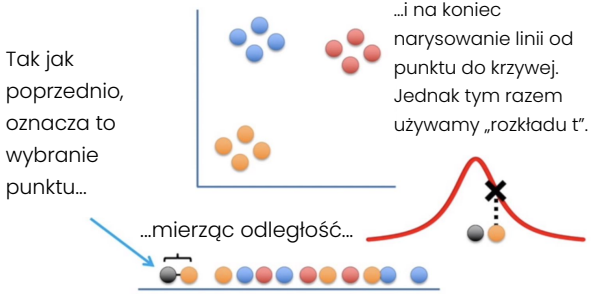
Teraz losowo rzutujemy dane na oś liczbową...

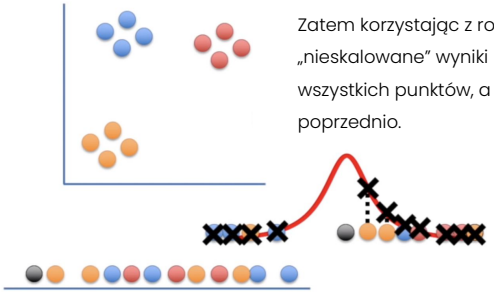


... i oblicz wyniki podobieństwa punktów na osi liczbowej.

ML xgBoost

t-SNE – idea działania

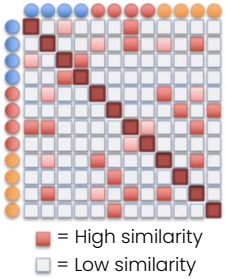


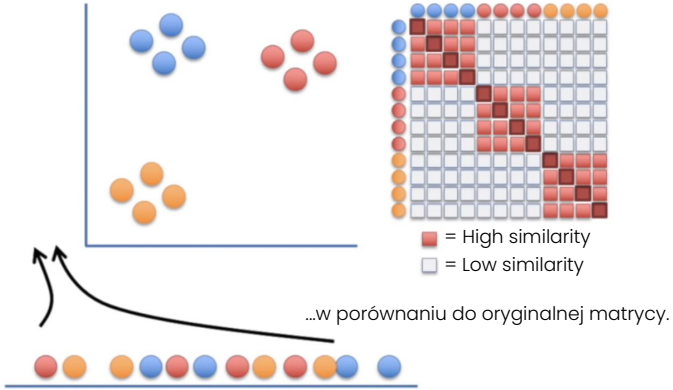


Zatem korzystając z rozkładu t, obliczamy „nieskalowane" wyniki podobieństwa dla wszystkich punktów, a następnie skalujemy je jak poprzednio.

ML xgBoost

t-SNE – idea działania



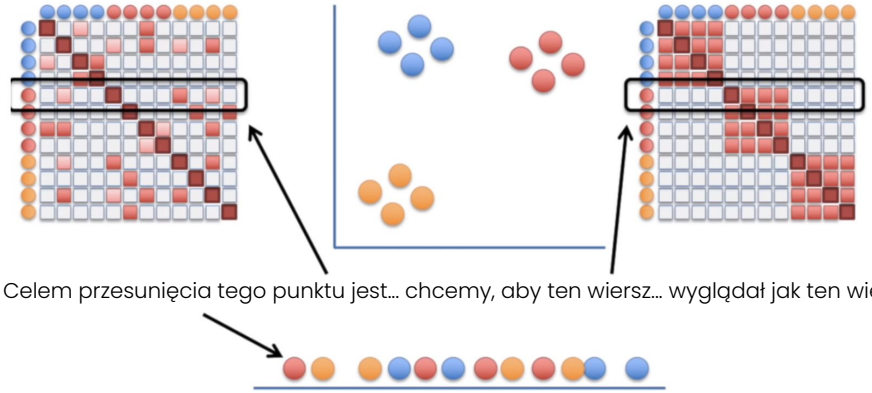


...w porównaniu do oryginalnej matrycy.

Podobnie jak poprzednio, otrzymujemy macierz wyników podobieństwa, ale w tej macierzy panuje bałagan...

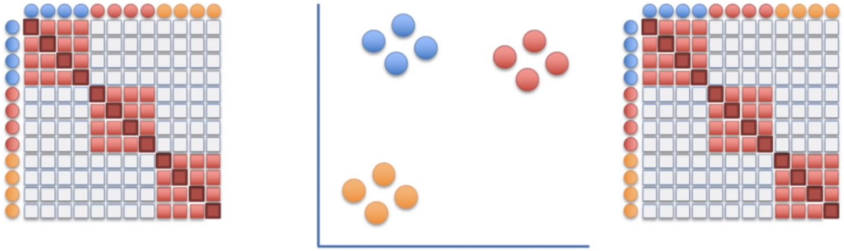
ML xgBoost

t-SNE – idea działania



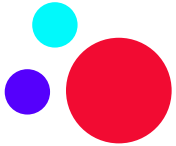
ML xgBoost

t-SNE – idea działania



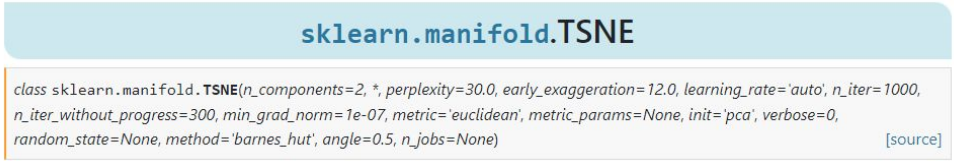
t-SNE przesuwa punkty po trochu i w każdym kroku wybiera kierunek, który sprawia, że macierz po lewej stronie bardziej przypomina macierz po prawej stronie.

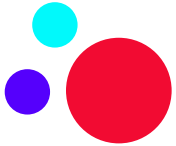
Używa małych kroków, ponieważ przypomina trochę grę w szachy i nie da się jej rozwiązać od razu. Zamiast tego wykonuje jeden ruch na raz.



ML xgBoost

Parametry t-SNE





ML xgBoost

t-SNE, a inne techniki

infoShare ACADEMY

Dlaczego t-SNE?

ML xgBoost

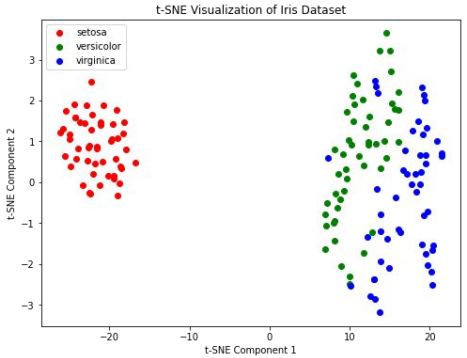
Implementacja

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler

iris = datasets.load_iris()
X = iris.data
y = iris.target

X_normalized = StandardScaler().fit_transform(X)

tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X_normalized)
```

ML xgBoost

implementacja

```
plt.figure(figsize=(8, 6))
for i, c in zip(range(3), ['red', 'green', 'blue']):
        plt.scatter(X_tsne[y == i, 0], X_tsne[y == i, 1], c=c,
label=iris.target_names[i])

plt.title('t-SNE Visualization of Iris Dataset')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.legend()
plt.show()
```



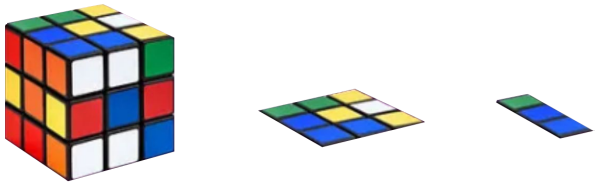
Zadanie 14.2 (instrukcja)

Przeprowadź analizę struktury zbioru danych dotyczącego samochodów za pomocą algorytmu t-SNE.

```
from sklearn.datasets import load_vehicles

vehicles = load_vehicles()
data = pd.DataFrame(vehicles.data,
columns=vehicles.feature_names)
target = vehicles.target
```

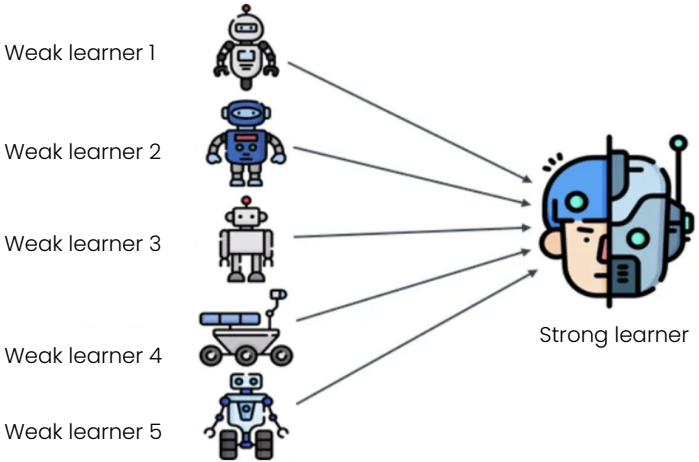
ML xgBoost

Redukcja wymiarowości - podsumowanie



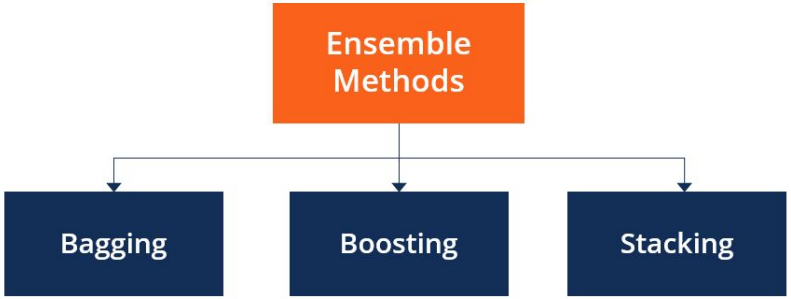
ML xgBoost

ensemble learning



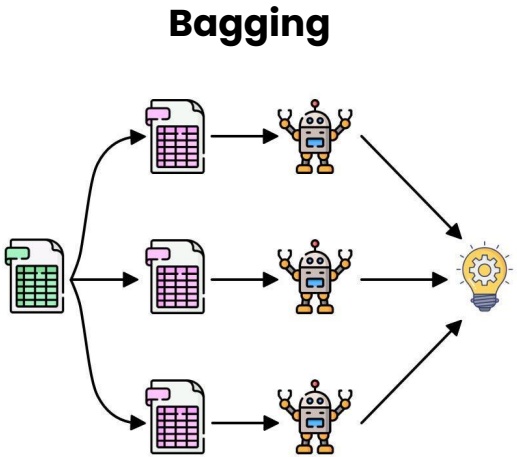
ML xgBoost

ensemble methods



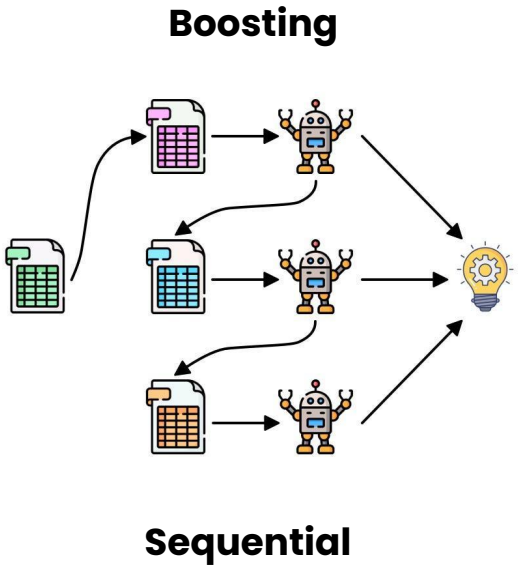
ML xgBoost

bagging



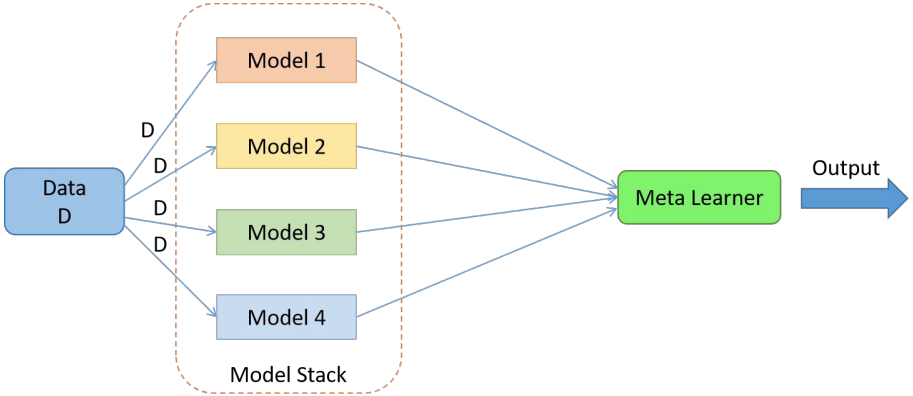
ML xgBoost

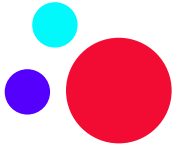
boosting



ML xgBoost

stacking





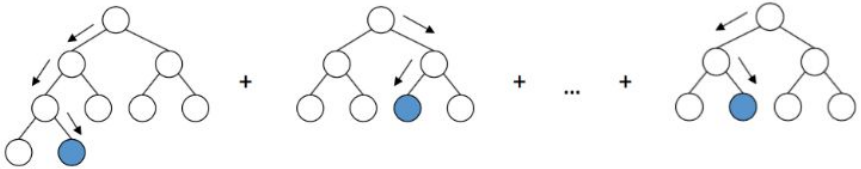
ML xgBoost

Charakterystyka słabych modeli

1. Niska złożoność
2. Niska dokładność indywidualna
3. Słaba generalizacja
4. Różnorodność (Diversity)
5. Niski koszt trenowania

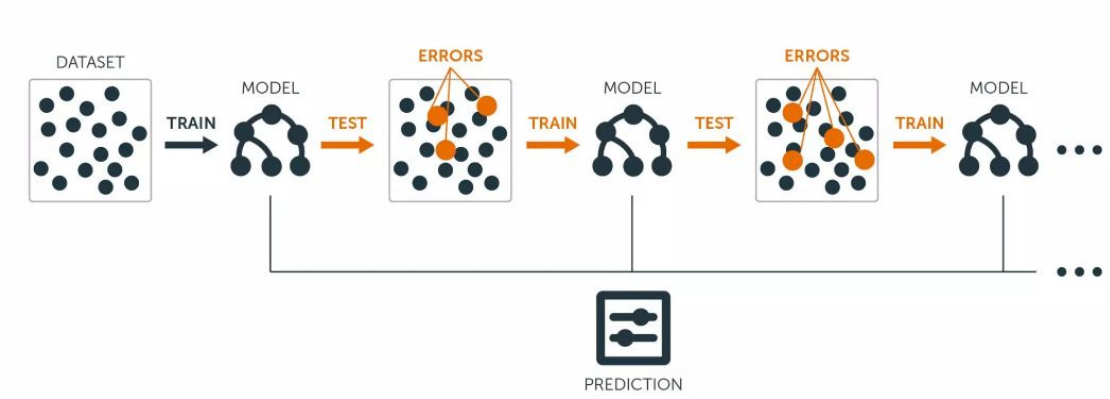
ML xgBoost

Algorytm gradient boosting



ML xgBoost

Trenowanie w Gradient Boostingu



ML xgBoost

Ekstremalny Gradient Boosting

Świadomość pamięci podręcznej i przetwarzanie poza rdzeniem.

Regularyzacja w celu uniknięcia nadmiernego dopasowania.

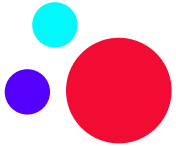
Przycinanie drzew metodą „najgłębszą".

Efektywna obsługa brakujących danych.

Równoległe budowanie drzewa.

Wbudowana funkcja sprawdzania krzyżowego.

infoShareAcademy.com



ML xgBoost

xgBoost - historia

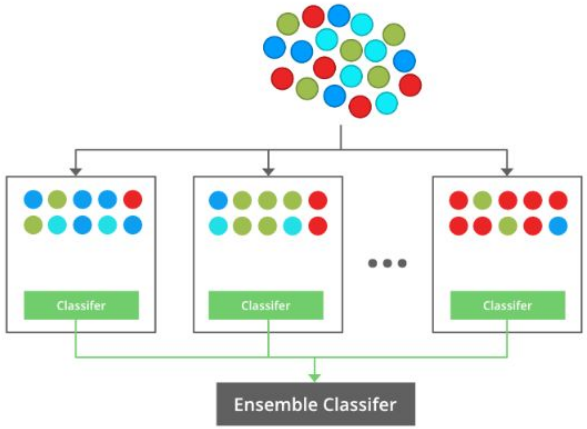
xgBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

ML xgBoost

xgBoost a Gradient Boosting



Original Data

Bootstrapping

Aggregating

Bagging

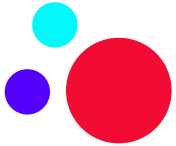
infoShareAcademy.com

ML XGBoost

Budowa xgBoost

Każdy kolejny model stara się przewidzieć błąd wszystkich modeli przed nim. Tzn. model 2 przewiduje błąd modelu 1 + średniej, model 1 modeluje błąd średniej.

Parametr C jest rozmiarem kroku, jaki robimy w naszej optymalizacji. Jest to tzw. learning rate.



ML xgBoost

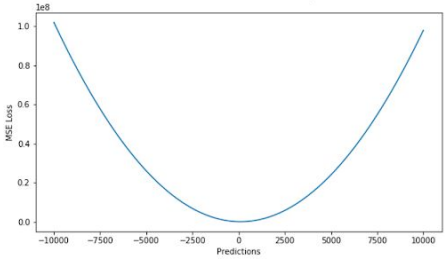
objective function

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta)$$

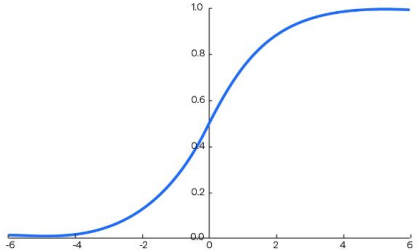
ML xgBoost

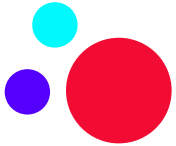
training loss

MSE



Funkcja Softmax





ML xgBoost

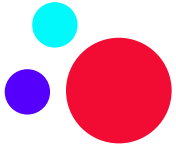
Regularyzacja

Lasso Regularization(L1)

$$\text{loss} = \sum_{i=0}^{n} (y_i - X_i\beta)^2 + \sum_{j=0}^{m} |\beta_j|$$

Lasso Regularization(L2)

$$\text{loss} = \sum_{i=0}^{n} (y_i - X_i\beta)^2 + \sum_{j=0}^{m} \beta_j^2$$



ML xgBoost

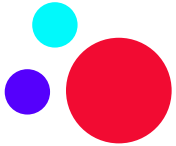
Regularyzacja L1

$$\text{Koszt} = \text{Funkcja Straty} + \alpha \sum_{i=1}^{n} |w_i|$$

ML xgBoost

Regularyzacja L2

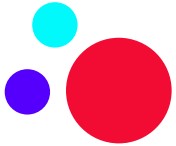
$$\text{Koszt} = \text{Funkcja Straty} + \alpha \sum_{i=1}^{n} w_i^2$$



ML xgBoost

Implementacja

```
import xgboost as xgb
```



ML xgBoost

Dane wejściowe

Pandas DataFrame

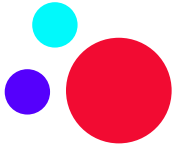
NumPy array

xgBoost DMatrix

ML xgBoost

Funkcja celu

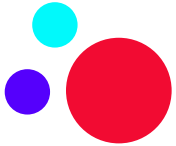
objective: określa użytą funkcję straty (loss)



ML xgBoost

learners

- Domyślnie ustawiony jest booster:gbtree, czy gradient boosted tree.
- Modyfikacją pierwszego jest booster:dart.
- Do dyspozycji jest jeszcze booster:gblinear.
- Nie jest zbyt popularny.
- Do treningu używamy funkcji .train() zamiast znanej .fit()



ML xgBoost

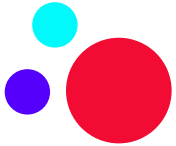
regularizations

Kara za złożoność modelu.

Istotne hiperparametry modelu:

alpha - regularyzacja L1 (lasso)

lambda - regularyzacja L2 (ridge)



ML xgBoost

Parametry

- **booster** (gbtree/gblinear/dart)
- **learning_rate**
- **gamma**
- **max_depth**
- **n_estimators**
- **subsample**
- **objective**
- **colsample_bytree**
- **n_jobs**
- **verbosity**
- **eval_metric**
- **missing**

ML xgBoost

Wizualizacja

Rysunek drzewa:

xgb.plot_tree()

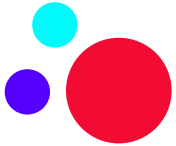
1 parametrem jest wytrenowany model.

2 parametrem jest głębokość rysunku.

Wykres istotności cech:

xgb.plot_importance()

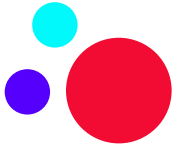
Parametrem jest model.



ML xgBoost

Zalety

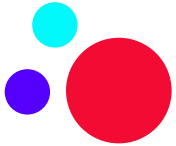
- Szybki i wydajny.
- Algorytm można zrównoleglić.
- Bardzo dokładny dla danych o ilości próbek większej niż liczba cech.
- Odpowiedni dla danych numerycznych oraz kategorycznych.
- Używany w problemach regresji i klasyfikacji.
- Kompatybilny z API scikit-learn.
- Szeroki wybór wbudowanych parametrów strojenia modelu.



ML xgBoost

Wady

- Nie jest odpowiedni do problemów lepiej rozwiązywalnych przez algorytmy uczenia głębokiego:
  - rozpoznawania obrazów,
  - przetwarzania języka naturalnego,
  - z małą liczbą przypadków uczących,
  - danych, gdzie ilość featurów jest porównywalna lub większa od ilości próbek.

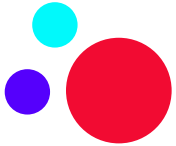


ML xgBoost

Implementacja xgBoost


```
import pandas as pd
import xgboost as xgb
import numpy as np
import time

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

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955	0	98178	47.5112	-122.257	1340	5650
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951	1991	98125	47.7210	-122.319	1690	7639
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933	0	98028	47.7379	-122.233	2720	8062
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965	0	98136	47.5208	-122.393	1360	5000
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987	0	98074	47.6168	-122.045	1800	7503



ML xgBoost

Implementacja xgBoost

Przykładowy trening:

```
y = data['price']
x = data.drop(['id', 'price', 'date', 'zipcode'], axis=1)

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

ML xgBoost

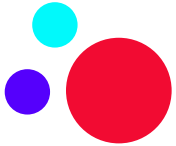
Implementacja xgBoost

```
xg_reg = xgb.XGBRegressor(random_state=123, n_estimators=5,
max_depth=7)

xg_reg.fit(x_train, y_train)

y_pred = xg_reg.predict(x_test)
```

```
array([736183.4 , 339161.88, 664775.1 , ..., 443761.  , 543238.06,
       285922.28], dtype=float32)
```

```
xg_reg.score(x_test, y_test)
```

```
0.7300266688026265
```



ML xgBoost

Implementacja xgBoost

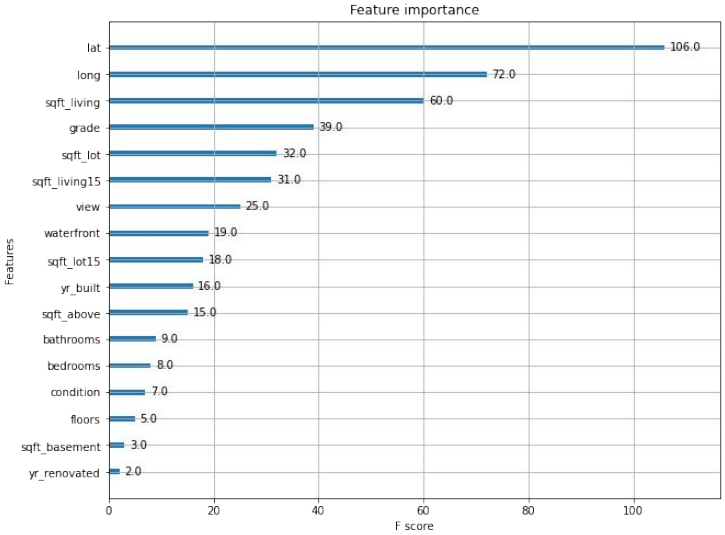

```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
import sklearn.metrics

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = sklearn.metrics.mean_absolute_error(y_test, y_pred)
```

```
RMSE - gbtree:  192431.47170763818
MAE - gbtree:  113103.9277917534
```

ML xgBoost

Implementacja xgBoost

xgb.plot_importance(xg_reg)



Zadanie 14.3 (instrukcja)

Dla zbioru danych dotyczącego cukrzycy (load_diabetes) dostępny w bibliotece scikit-learn, stwórz model regresji przy użyciu xgBoost, który przewiduje progresję choroby na podstawie różnych cech pacjentów.

```
from sklearn.datasets import load_diabetes
data = load_diabetes()
```

ML xgBoost

Implementacja

Klasyfikacja:

```
import xgboost as xgb
import numpy as np
import time
from sklearn import datasets
import matplotlib.pyplot as plt

x, y = datasets.make_classification(
    n_samples=2500, n_features=20, n_informative=3, n_redundant=2
)
```

ML xgBoost

Implementacja

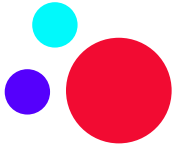
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=23)

xgb_cl = xgb.XGBClassifier(n_estimators=100, max_depth=3,
use_label_encoder=False, eval_metric='error')

xgb_cl.fit(x_train, y_train)
```

ML xgBoost

Implementacja

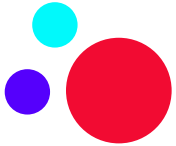
```
from sklearn.metrics import accuracy_score

y_pred = xgb_cl.predict(x_test)
acc = accuracy_score(y_test, y_pred)
print("acc: ", acc)
```

```
acc:  0.948
```



Zadanie 14.4 (instrukcja)

Wytrenuj klasyfikator xgboost na syntetycznym zbiorze danych i porównaj go z Random Forest (na tych samych parametrach).



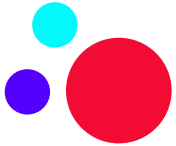
ML xgBoost

Implementacja

Zaawansowana klasyfikacja:

```
import xgboost as xgb
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("../data/income_evaluation.csv", sep='\s*,\s*', header=0,
encoding='ascii', engine='python')
df.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K



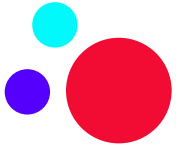
ML xgBoost

Implementacja

Konwersja danych kategorycznych do numerycznych:

sklearn.preprocessing.LabelEncoder

```
from sklearn.preprocessing import LabelEncoder

list_to_encode = [
    'workclass', 'education', 'marital-status', 'occupation',
    'relationship', 'race', 'sex', 'native-country'
]
```



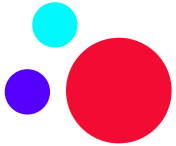
ML xgBoost

Implementacja


```
df_encoded = pd.get_dummies(df, columns=list_to_encode)
df_encoded.head()
```

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	income	workclass_?	workclass_Federal-gov	workclass_Local-gov	...	native-country_Portugal
0	39	77516	13	2174	0	40	<=50K	0	0	0	...	0
1	50	83311	13	0	0	13	<=50K	0	0	0	...	0
2	38	215646	9	0	0	40	<=50K	0	0	0	...	0
3	53	234721	7	0	0	40	<=50K	0	0	0	...	0
4	28	338409	13	0	0	40	<=50K	0	0	0	...	0

5 rows × 109 columns

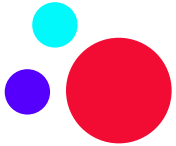


ML xgBoost

Implementacja

Wybór cech:

```
from sklearn.preprocessing import
LabelEncoder

le = LabelEncoder()

y = le.fit_transform(df_encoded['income'])
x = df_encoded.drop(['income'], axis=1)

data_dm = xgb.DMatrix(data=x.values, label=y)
```



ML xgBoost

Implementacja

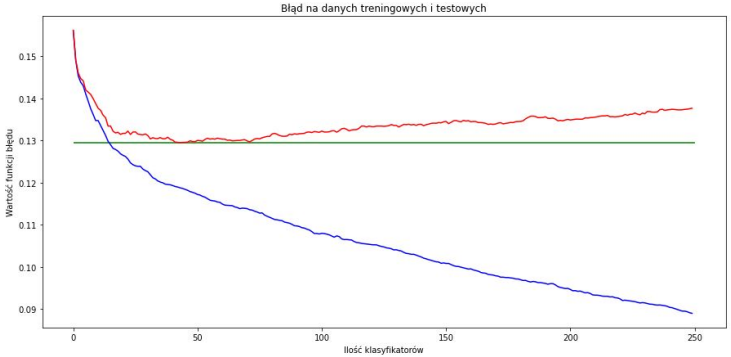
Kroswalidacja:

```
params = {
    "objective": "reg:logistic",
    "max_depth": 3,
    "booster": "gbtree",
    "learning_rate": 0.8
}
cv_results = xgb.cv(
    dtrain=data_dm,
    params=params,
    nfold=4,
    num_boost_round=250, metrics=["error",
"auc"], as_pandas=True)
```

ML xgBoost

Implementacja

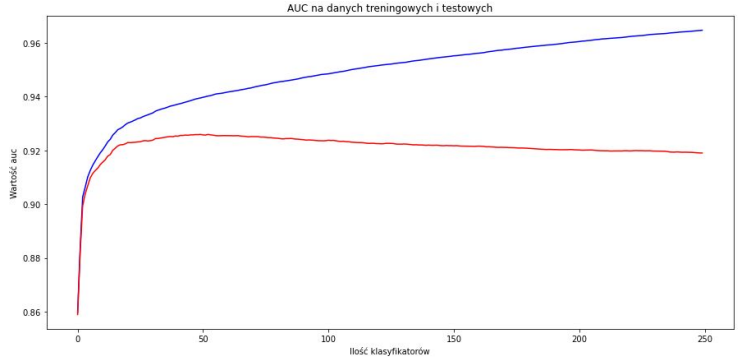
```
plt.figure(figsize=(15, 7))
plt.title("Błąd na danych treningowych i testowych")
plt.plot(cv_results["train-error-mean"], color="b")
plt.plot(cv_results["test-error-mean"], color="r")
# plt.ylim((0., 0.2))
plt.hlines([cv_results["test-error-mean"].min()], xmin=0, xmax=250, color="g")
plt.xlabel("Ilość klasyfikatorów")
plt.ylabel("Wartość funkcji błędu")
```



ML xgBoost

Implementacja

```
plt.figure(figsize=(15, 7))
plt.title("AUC na danych treningowych i
testowych")
plt.plot(cv_results["train-auc-mean"], color="b")
plt.plot(cv_results["test-auc-mean"], color="r")
plt.xlabel("Ilość klasyfikatorów")
plt.ylabel("Wartość auc")
```



ML xgBoost

Podsumowanie