Vilniaus Universitetas

Dimensijos mažinimas klasifikavime

Darbą atliko:

Vainius Gataveckas, Matas Gaulia, Dovydas Martinkus

Duomenų Mokslas

3 kursas 2 gr.

Vilnius, 2022

**Turinys**

[1 Tikslas ir uždaviniai 3](#_Toc102856727)

[2 Duomenų aibė 4](#_Toc102856728)

[3 Atliktos analizės aprašymas 5](#_Toc102856729)

[3.1 Pradinis apdorojimas 5](#_Toc102856730)

[3.2 k-NN klasifikatorius 7](#_Toc102856731)

[3.3 Naivus Bajeso klasifikatorius 9](#_Toc102856732)

[3.4 Sprendimų medžio klasifikatorius 11](#_Toc102856733)

[3.5 Atsitiktinio miško klasifikatorius 13](#_Toc102856734)

[3.6 Klasifikavimo kokybės įvertinimas ir modelių palyginimas 15](#_Toc102856735)

[4 Išvados 19](#_Toc102856736)

[5 Šaltiniai 20](#_Toc102856737)

[Priedas 21](#_Toc102856738)

# Tikslas ir uždaviniai

Tikslas: Naudojant pasirinktą duomenų aibę ištirti pasirinktus klasifikatorius, įvertinti klasifikavimo kokybę ir tarpusavyje palyginti klasifikavimo metodus.

Uždaviniai:

Pasirinktų klasifikatorių teorinis tyrimas.

Optimalių parametrų parinkimas.

Klasifikavimo šablonų sudarymas.

Klasifikavimo kokybės įvertinimas ir modelių palyginimas.

Apibendrintų išvadų pateikimas.

# Duomenų aibė

Spotify Past Decades Songs duomenų aibė

Duomenų aibės šaltinis: Kaggle

Nuoroda per internetą: <https://www.kaggle.com/cnic92/spotify-past-decades-songs-50s10s?select=1990.csv>

Duomenų aibę sudaro tokie požymiai:

„Number“ – (kategorinis, nominalusis) dainą identifikuojantis kodas

„Title“ – (kategorinis, nominalusis) dainos pavadinimas

„Artist“ – (kategorinis, nominalusis) atlikėjas arba grupė

„Top Genre“ – (kategorinis, nominalusis) dainos žanras

„Year“ – (kiekybinis, diskretusis, intervalinė skalė) išleidimo metai

„Decade“ – (kiekybinis, diskretusis, intervalinė skalė) išleidimo dešimtmetis

„Tempo“ – (kiekybinis, tolydus, santykių skalė) dainos tempas

“Loudness (dB)“ - (kiekybinis, tolydus, intervalų skalė) dainos garsumas

„Duration“ – (kiekybinis, tolydus, santykių skalė) dainos trukmė

„Energy“ – (kiekybinis, tolydus, santykių skalė) dainos energija

„Danceability“ – (kiekybinis, tolydus, santykių skalė) lengvumas šokti pagal dainą

„Liveness“ – (kiekybinis, tolydus, santykių skalė) kaip tikėtina, kad daina yra gyvas įrašas

„Valence“ – (kiekybinis, tolydus, santykių skalė) dainos pozityvumas

„Acousticness“ – (kiekybinis, tolydus, santykių skalė) dainos akustiškumas

„Speechiness“ – (kiekybinis, tolydus, santykių skalė) kiek dainoje yra kalbama

„Popularity“ - (kiekybinis, tolydus, santykių skalė) dainos populiarumas pagal perklausų skaičių

# Atliktos analizės aprašymas

## Pradinis apdorojimas

Klasifikavimui pasirinktas dainų išleidimo dešimtmetis (požymis „Decade“) su dvejomis galimomis reikšmėmis (80-ieji arba 2010-ieji). 2010-ųjų dainoms priskirta reikšmė 0, 80-ųjų dainoms – 1.

Abiem dešimtmečiams apskaičiuotos aprašomosios statistikos (žr. 1 ir 2 lentelės).

1 lentelė Aprašomosios statistikos charakteristikos 80-ųjų dainoms

|  |  |  |  |
| --- | --- | --- | --- |
| Požymis | Vidurkis | Standartinis nuokrypis | Mediana |
| tempo | 122.6 | 25.1 | 122.0 |
| energy | 64.9 | 20.2 | 68.0 |
| danceability | 62.3 | 13.4 | 63.0 |
| loudness | -9.1 | 3.7 | -9.0 |
| liveness | 16.4 | 13.9 | 11.0 |
| valence | 63.1 | 25.9 | 70.0 |
| duration | 258.9 | 51.2 | 251.0 |
| acousticness | 24.6 | 21.7 | 19.0 |
| speechiness | 4.4 | 2.8 | 4.0 |
| popularity | 67.7 | 7.3 | 68.0 |

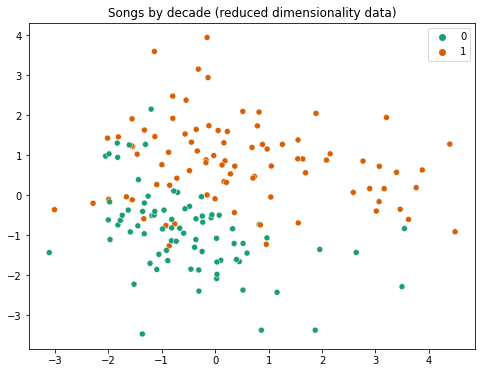
2 lentelė Aprašomosios statistikos charakteristikos 2010-ųjų dainoms

|  |  |  |  |
| --- | --- | --- | --- |
| Požymis | Vidurkis | Standartinis nuokrypis | Mediana |
| tempo | 118.7 | 22.4 | 120.0 |
| energy | 68.0 | 16.3 | 68.0 |
| danceability | 65.4 | 11.9 | 67.0 |
| loudness | -5.5 | 2.0 | -5.0 |
| liveness | 17.9 | 13.8 | 13.0 |
| valence | 46.4 | 20.9 | 47.0 |
| duration | 209.2 | 24.1 | 209.0 |
| acousticness | 14.7 | 17.9 | 9.0 |
| speechiness | 8.4 | 7.8 | 6.0 |
| popularity | 76.0 | 9.2 | 78.0 |

Duomenys padalinti į mokymo ir testavimo aibes naudojant santykį 80-20.

Visų skaitinių požymių matavimo skalės suvienodintos standartizuojant juos pagal vidurkį ir dispersiją. Standartizavimui naudotos vidurkio ir dispersijos reikšmės gautos iš mokymo aibės.

Siekiant palyginti klasifikavimo rezultatus naudojant dimensijos mažinimo metodus ir jų nenaudojant, duomenų aibės požymių dimensija sumažinta iki 2 naudojantis pagrindinių komponenčių analizės (angl. principal component analysis, toliau - PCA) metodu jį apmokant naudojant mokymo aibę (žr. 1 pav.).



1 pav. Dimensijos sumažinimas naudojantis PCA metodu mokymo aibės duomenims

## Naivus Bajeso klasifikatorius

Naivaus Bajeso (angl. naive Bayes) klasifikavimo metodas pagrįstas sąlyginės tikimybės modeliu gautu naudojantis Bajeso teorema. Metodas vadinamas „naivuoju“, nes taikoma požymių tarpusavio nepriklausomumo prielaida. Gautą tikimybinį modelį galima užrašyti:

Turint skaitinius požymius dažniausiai laikoma, kad turi normalujį skirstinį (2).

Iš šio tikimybinio modelio yra sudaromas klasifikatorius pasirenkant tą klasę, kurios tikimybė yra didžiausia. Modelyje naudojama visų požymių tarpusavio nepriklausomumo sąlyga turint tikras duomenų aibes yra pažeidžiama labai dažnai. Tai akivaizdu tais atvejais, kai tam tikros požymių kombinacijos iš viso nėra įmanomos, tačiau ir kitais atvejais didelio kiekio pasitaikančių požymių negalima laikyti nepriklausomais. (pvz. asmens ūgis ir svoris).

Nepaisant modeliui reikalingos prielaidos, tam tikrose taikymo srityse šiuo modeliu gaunami geri rezultatai. naivaus Bajeso klasifikatoriui nereikia parinkti esminių parametrų. Šis klasifikavimo metodas pasižymi greitu veikimu, gerais rezultatais požymių aibėje esant kategoriniams požymiams.

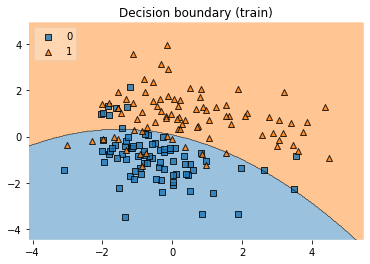
5 lentelė Naivaus Bajeso gauta sumaišymo matrica originalios dimensijos duomenims

|  |  |  |
| --- | --- | --- |
|  | Prognozuota reikšmė | |
| Tikra reikšmė | 18 | 4 |
| 2 | 17 |

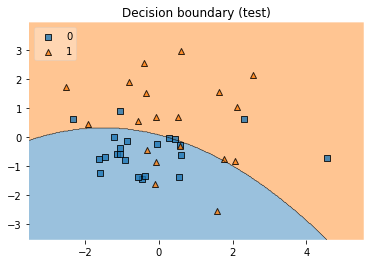
6 lentelė Naivaus Bajeso gauta sumaišymo matrica originalios dimensijos duomenims

|  |  |  |
| --- | --- | --- |
|  | Prognozuota reikšmė | |
| Tikra reikšmė | 18 | 4 |
| 7 | 12 |

Vizualiai pavaizduotas gautas sprendimo paviršius naudojant sumažintos dimensijos duomenis su mokymo ir testavimo aibių stebėjimais (žr. 4 ir 5 pav.).



4 pav. Sprendimo paviršius su mokymo aibės rezultatais naiviam Bajeso klasifikatoriui



5 pav. Sprendimo paviršius su testavimo aibės rezultatais naiviam Bajeso klasifikatoriui

## Sprendimų medžio klasifikatorius

Sprendimų medžių (angl. decision tree) konstravimo algoritmai paprastai veikia iš viršaus į apačią kiekviename žingsnyje pasirenkant tokį kintamąjį, kuris padalintų elementų rinkinį geriausiu būdu pagal tam tikrą naudojamą metriką. Dažnai naudojamos metrikos yra Gini priemaiša (Gini impurity), informacijos išlošis (information gain). Medžio lapo, į kurį pateko klasifikuojamas objektas, reikšmė atitinką modelio priimtą sprendimą (3).

Sprendimų medžiai sugeba lengvai prisitaikyti prie struktūrų, esančių mokymo duomenyse, tačiau rezultatai itin stipriai priklauso nuo to, kokie duomenys turėti mokymo aibėje. Dėl šios priežasties sprendimu medžiu tikėtina gauti stipriai prastesnius rezultatus klasifikuojant prieš tai nematytus stebėjimus (tai vadinama low-bias high-variance modeliu). Su šia problema susijęs modelio parametrų parinkimas, pavyzdžiui: *max\_depth* kontroliuoja maksimalų medžio gylį *, min\_samples\_split* parametru parenkamas minimalus stebėjimų kiekis, reikalingas norint dar kartą skaidyti duomenų aibę*, min\_samples\_leaf –* minimalus stebėjimų skaičius medžio lapuose. Šiuos parametrus siekiama parinkti taip, kad būtų išvengiama persimokymo (angl. overfitting).

Vienas iš sprendimų medžių privalumų yra beveik nereikalingas pradinis duomenų apdorojimas: Duomenys pateikiami neturi būti vienodoje skalėje, priklausomai nuo metodo implementacijos gali būti pateikiami objektai su praleistomis požymių reikšmėmis, nebūtina perkoduoti kategorinius kintamuosius.

Sprendimų medžiai lengvai suprantami ir interpretuojami, taip pat metode turimas savaiminis požymių parinkimo mechanizmas: šakoms sudaryti požymis naudojamas tik tada, kai juo gautas padalijimas yra geriausias galimas.

Geriausių parametrų rinkinys rastas naudojant parametrų tinklelį *max\_depth*={2,3,4}, *min\_samples\_split*={5,10,15,20,25}.

Geriausi rezultatai originalios dimensijos duomenims gauti su *max\_depth*=4 ir *min\_samples\_split*=5, tuo tarpu sumažintos dimensijos duomenims geriausi rezultatai gauti su ta pačia max\_depth reikšme, bet su *min\_samples\_split*=20. Originalios ir sumažintos dimensijos duomenims naudojant parametrus *min\_samples\_split*=5, max\_features=5, tuo tarpu sumažintos dimensijos duomenims geriausia *min\_samples\_split* parametro reikšmė gauta lygi 10.

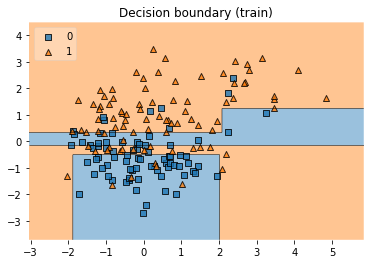
7 lentelė Spendimų medžio gauta sumaišymo matrica originalios dimensijos duomenims

|  |  |  |
| --- | --- | --- |
|  | Prognozuota reikšmė | |
| Tikra reikšmė | 18 | 4 |
| 6 | 13 |

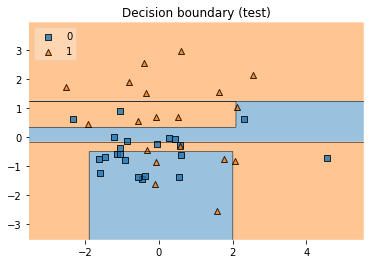
8 lentelė Sprendimų medžio gauta sumaišymo matrica sumažintos dimensijos duomenims

|  |  |  |
| --- | --- | --- |
|  | Prognozuota reikšmė | |
| Tikra reikšmė | 16 | 6 |
| 5 | 14 |

Vizualiai pavaizduotas gautas sprendimo paviršius naudojant sumažintos dimensijos duomenis su mokymo ir testavimo aibių stebėjimais (žr. 6 ir 7 pav.)



6 pav. Sprendimo paviršius su mokymo aibės rezultatais sprendimių medžio klasifikatoriui



7 pav. Sprendimo paviršius su testavimo aibės rezultatais sprendimų medžio klasifikatoriui

## Atsitiktinio miško klasifikatorius

Metodai, sukonstruojantys daugiau negu vieną medį vadinami ansamblių (angl. ensemble) metodais.

Bagging sprendimų medis (Bootstrap aggregating) sudaro kelis sprendimų medžius kiekvieną medį konstruodamas naudodamasis duomenų aibe, gauta imant tokio pačio dydžio imtį su gražinimu iš originalios duomenų aibės (saviranka, angl. bootstrap). Galutinis sprendimas gaunamas sujungiant visų medžių sprendimus į galutinį sprendimą.

Atsitiktinio miško metode (angl. random forest) kiekvieną kartą, kai medyje siekiama sudaryti šaką, geriausio galimo skaidymo (angl. split) ieškoma tik tarp atsitiktinai parinkto požymių poaibio (4). Šio į modelį įdedamo atsitiktinumo esmė yra sumažinti modelio priklausomumą nuo to, kokie duomenys buvo naudojami apmokyti modelį (variance). Naudojant atsitiktinį mišką gali sumažėti modelio gebėjimas prisitaikyti prie sudėtingų struktūrų duomenyse (bias), tačiau praktiškai šis sumažėjimas būna daug mažesnis už variance sumažėjimą.

Naudojamas medžių skaičius įprastai parenkamas iš anksto atsižvelgiant į duomenų sudėtingumą ir kiekį (sudėtingiems duomenims įprastai imamos reikšmės tūkstančiuose), todėl svarbiausias modelio parametras yra atsitiktinai parenkamų požymių skaičius *max\_features.* Galimi parinkti ir parametrai naudojami paprastiems sprendimų medžiams, tačiau jie nebėra tokie svarbūs.

Kadangi neturima daug stebėjimų medžių skaičius pasirinktas lygus 100. Geriausi parametrų rinkinys rastas naudojant parametrų tinklelį *max\_features*={2,3,4,5}, *min\_samples\_split*={5,10,15,20,25}. Sumažintos dimensijos duomenims nėra prasminga parinkti kitokią *max\_features* reikšmę negu 1, nes naudojant reikšmę 2 butų gaunamas bagging sprendimų medis.

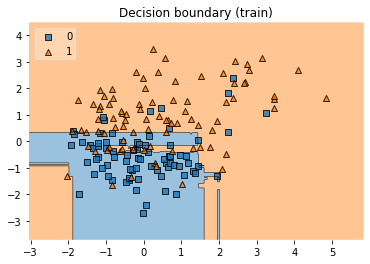
lentelė 9 Atsitiktiniu mišku gauta sumaišymo matrica originalios dimensijos duomenims

|  |  |  |
| --- | --- | --- |
|  | Prognozuota reikšmė | |
| Tikra reikšmė | 17 | 5 |
| 4 | 15 |

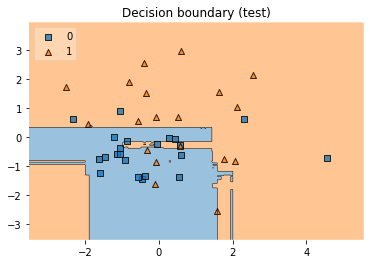
lentelė 10 Atsitiktiniu mišku gauta sumaišymo matrica sumažintos dimensijos duomenims

|  |  |  |
| --- | --- | --- |
|  | Prognozuota reikšmė | |
| Tikra reikšmė | 16 | 6 |
| 4 | 15 |

Pavaizduotas gautas sprendimo paviršius su mokymo ir testavimo aibių stebėjimais (žr. 8 ir 9 pav.)



8 pav. Sprendimo paviršius su mokymo aibės rezultatais atsitiktinio miško klasifikatoriui



9 pav. Sprendimo paviršius su testavimo aibės rezultatais atsitiktinio miško klasifikatoriui

## Klasifikavimo kokybės įvertinimas ir modelių palyginimas

Kadangi modelių palyginimui negali būti naudojama testavimo aibė, modeliai gali būti palyginti kitas strategijas: kryžminės validacijos (angl. k-fold cross validation) ir išlaikymo aibės (angl. validation / hold-out).

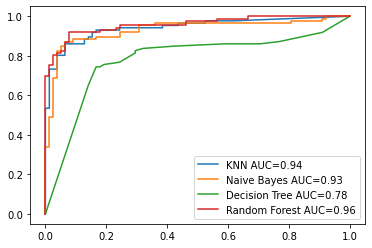
11 lentelė Klasifikavimo modelių metrikos naudojant kryžminę validaciją originalios dimensijos duomenims

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modelis | Klasė | Precision | Recall | F1-score | Accuracy |
| k-NN | 0 | 0.87 | 0.91 | 0.89 | 0.89 |
| k-NN | 1 | 0.91 | 0.87 | 0.89 | 0.89 |
| Naive Bayes | 0 | 0.86 | 0.91 | 0.88 | 0.88 |
| Naive Bayes | 1 | 0.91 | 0.86 | 0.89 | 0.88 |
| Decision Tree | 0 | 0.81 | 0.85 | 0.83 | 0.84 |
| Decision Tree | 1 | 0.86 | 0.83 | 0.84 | 0.84 |
| Random Forest | 0 | 0.91 | 0.91 | 0.91 | 0.91 |
| Random Forest | 1 | 0.92 | 0.92 | 0.92 | 0.91 |

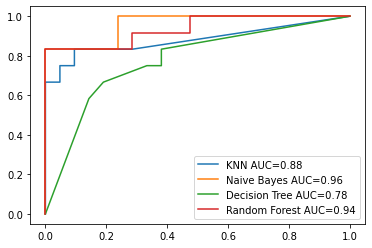
12 lentelė Klasifikavimo modelių metrikos naudojant išlaikymo aibę originalios dimensijos duomenims

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modelis | Klasė | Precision | Recall | F1-score | Accuracy |
| k-NN | 0 | 0.87 | 0.95 | 0.91 | 0.88 |
| k-NN | 1 | 0.9 | 0.75 | 0.82 | 0.88 |
| Naive Bayes | 0 | 0.91 | 1.0 | 0.95 | 0.94 |
| Naive Bayes | 1 | 1.0 | 0.83 | 0.91 | 0.94 |
| Decision Tree | 0 | 0.82 | 0.67 | 0.74 | 0.7 |
| Decision Tree | 1 | 0.56 | 0.75 | 0.64 | 0.7 |
| Random Forest | 0 | 0.91 | 1.0 | 0.95 | 0.94 |
| Random Forest | 1 | 1.0 | 0.83 | 0.91 | 0.94 |

Tiek naudojant kryžminę validaciją, tiek pasitelkiant išlaikymo duomenų rinkinį geriausi rezultatai pagal AUC gauti naudojant atsitiktinio miško ir naivaus Bajeso metodus (žr. atitinkamai 12 ir 13 pav.). Nuo šių modelių nestipriai atsiliko k-NN modelis, tuo tarpu paprastu sprendimų medžiu gauti stipriai prastesni rezultatai.



10 pav. ROC kreivė su AUC reikšmėmis naudojant kryžminę validaciją originalios dimensijos duomenims



11 pav. ROC kreivė su AUC reikšmėmis naudojant išlaikymo duomenis originalios dimensijos duomenims

13 lentelė Klasifikavimo modelių metrikos naudojant kryžminę validaciją sumažintos dimensijos duomenims

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modelis | Klasė | Precision | Recall | F1-score | Accuracy |
| k-NN | 0 | 0.73 | 0.79 | 0.76 | 0.76 |
| k-NN | 1 | 0.8 | 0.73 | 0.76 | 0.76 |
| Naive Bayes | 0 | 0.71 | 0.85 | 0.77 | 0.76 |
| Naive Bayes | 1 | 0.83 | 0.69 | 0.75 | 0.76 |
| Decision Tree | 0 | 0.79 | 0.74 | 0.77 | 0.79 |
| Decision Tree | 1 | 0.78 | 0.83 | 0.8 | 0.79 |
| Random Forest | 0 | 0.74 | 0.74 | 0.74 | 0.76 |
| Random Forest | 1 | 0.77 | 0.77 | 0.77 | 0.76 |

14 lentelė Klasifikavimo modelių metrikos naudojant išlaikymo aibę sumažintos dimensijos duomenims

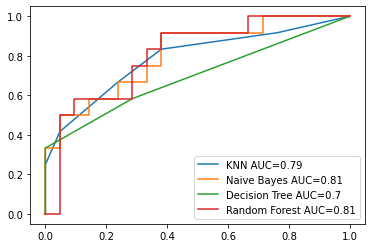
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modelis | Klasė | precision | recall | f1-score | accuracy |
| k-NN | 0 | 0.8 | 0.76 | 0.78 | 0.73 |
| k-NN | 1 | 0.62 | 0.67 | 0.64 | 0.73 |
| Naive Bayes | 0 | 0.77 | 0.81 | 0.79 | 0.73 |
| Naive Bayes | 1 | 0.64 | 0.58 | 0.61 | 0.73 |
| Decision Tree | 0 | 0.75 | 0.71 | 0.73 | 0.67 |
| Decision Tree | 1 | 0.54 | 0.58 | 0.56 | 0.67 |
| Random Forest | 0 | 0.79 | 0.71 | 0.75 | 0.7 |
| Random Forest | 1 | 0.57 | 0.67 | 0.62 | 0.7 |

Naudojant sumažintos dimensijos duomenis visų klasifikatorių rezultatai paveikti neigiamai (žr. atitinkamai 16 ir 17 pav.). Didžiausias nuostolis gautas naudojant atsitiktinio miško ir naivaus Bajeso klasifikatorius. Pasirinkta laikyti, kad šie rezultatai nėra netikėti, nes pradinė duomenų dimensija nebuvo didelė (n=10) ir naudojant dimensijos mažinimo metodus galimas (stiprus) informacijos praradimas.

.Paveikslėlis, kuriame yra žinutė, sportas

Automatiškai sugeneruotas aprašymas

12 pav. ROC kreivė su AUC reikšmėmis naudojant kryžminę validaciją sumažintos dimensijos duomenims



13 pav. ROC kreivė su AUC reikšmėmis naudojant išlaikymo duomenis sumažintos dimensijos duomenims

# Išvados

Duomenų aibę sudaro skirtingų dešimtmečių dainos su skaitiniais požymiais apie šias dainas. Duomenys

# Šaltiniai

1. *k-Nearest neighbour classifiers.* **P. Cunningham, S. Delany.** 2007 m., Mult Classif Syst, T. 54.

2. *An Empirical Study of the Naïve Bayes Classifier.* **Rish, I.** 2001 m., Empirical Methods for Artificial Intelligence.

3. **L. Breiman, J. Friedman, R. Olshen, C. Stone.** *Classification and Regression Trees.* Belmont, CA : Wadworth, 1984.

4. *Random Forests.* **Breiman, L.** 2001 m., Machine Learning, T. 45, p. 5-32.

# Priedas

Žemiau pateiktas naudotas programinis kodas:

# #### Read-in the data

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import seaborn as sns

def read\_clean\_data(filename):

df = pd.read\_csv(filename)[['title','artist','year','bpm', 'nrgy', 'dnce', 'dB','live', 'val', 'dur','acous', 'spch','pop']]

df = df.rename({'bpm':'tempo','nrgy':'energy','dnce':'danceability','dB':'loudness','live':'liveness',

'val':'valence','dur':'duration','acous':'acousticness','spch':'speechiness','pop':'popularity'},

axis = 1)

df['decade'] = filename[2:4] + 's'

return df

from sklearn.preprocessing import StandardScaler

filenames = ['1950.csv','1980.csv','2010.csv']

df = pd.concat([read\_clean\_data(i) for i in filenames]).reset\_index()

df\_id = df[["title","artist","decade"]]

df = df.iloc[:,4:len(df.columns)-1]

# pozymiai, pagal kuriuos klasteriuojama

df.describe().T.drop("count",axis=1)

x = StandardScaler().fit\_transform(df)

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN

from sklearn.manifold import MDS

from sklearn.decomposition import PCA

def do\_mds(x,n\_components=2,\*\*kwargs):

mds = MDS(n\_components,\*\*kwargs)

x\_trans = mds.fit\_transform(x)

return x\_trans

def do\_pca(x,n\_components=2,\*\*kwargs):

pca = PCA(n\_components,\*\*kwargs)

x\_trans = pca.fit\_transform(x)

return x\_trans

# klasterizavimas originaliems duomenims ir sumazinus dimensija

x\_small = do\_pca(x,n\_components=2)

df\_plot = pd.DataFrame(do\_pca(x))

df\_plot.columns = ["x","y"]

fig, ax = plt.subplots(1,1,figsize=(8, 6))

plot = sns.scatterplot(x="x",y="y",hue=df\_id["decade"],data=df\_plot,palette="Dark2",ax=ax)

plot.set\_title("K-means clustering (reduced dimensionality data)")

# #### KMeans

def do\_kmeans(x, standartize = True,\*\*kwargs):

model = KMeans(\*\*kwargs)

pred = model.fit\_predict(x)

return pred, model

from yellowbrick.cluster import SilhouetteVisualizer, KElbowVisualizer

visualizer = KElbowVisualizer(KMeans(), k=(2,10))

visualizer.fit(x)

visualizer.show()

visualizer = KElbowVisualizer(KMeans(), k=(2,10))

visualizer.fit(x\_small)

visualizer.show()

silhouette\_scores = []

for i in list(range(2,11)):

model = KMeans(i)

visualizer = SilhouetteVisualizer(model, colors='yellowbrick')

visualizer.fit(x)

silhouette\_scores.append(visualizer.silhouette\_score\_)

plt.clf()

plt.plot(range(2,11),silhouette\_scores)

plt.title("Silhoutte method optimal number of clusters (original dimensionality data)")

plt.xlabel("Number of clusters")

plt.ylabel("Average silhouette width")

silhouette\_scores = []

for i in list(range(2,11)):

model = KMeans(i)

visualizer = SilhouetteVisualizer(model, colors='yellowbrick')

visualizer.fit(x\_small)

silhouette\_scores.append(visualizer.silhouette\_score\_)

plt.clf()

plt.plot(range(2,11),silhouette\_scores)

plt.title("Silhoutte method optimal number of clusters (reduced dimensionality data)")

plt.xlabel("Number of clusters")

plt.ylabel("Average silhouette width")

for i in [2,5]:

model = KMeans(i,random\_state=123)

visualizer = SilhouetteVisualizer(model, colors='yellowbrick')

visualizer.fit(x)

visualizer.show()

for i in [2,5]:

model = KMeans(i,random\_state=123)

visualizer = SilhouetteVisualizer(model, colors='yellowbrick')

visualizer.fit(x\_small)

visualizer.show()

# elbow metodas

fig, ax = plt.subplots(1,2,figsize=(17, 6))

ax = ax.flatten()

df\_plot["cluster"], \_ = do\_kmeans(x,n\_clusters=5,random\_state=123)

plot = sns.scatterplot(x="x",y="y",hue="cluster",data=df\_plot,palette="Dark2",ax=ax[0])

plot.set\_title("K-means clustering (original dimensionality data)")

df\_plot["cluster"], \_ = do\_kmeans(x\_small,n\_clusters=5,random\_state=123)

plot = sns.scatterplot(x="x",y="y",hue="cluster",data=df\_plot,palette="Dark2",ax=ax[1])

plot.set\_title("K-means clustering (reduced dimensionality data)")

# kuo skiriasi klasteriai pagal desimtmecius

sns.set\_context("talk")

df\_plot["decade"] = df\_id["decade"]

sns.relplot(x="x",y="y",kind="scatter",data=df\_plot,col="cluster",hue="decade",palette="Set2",col\_wrap=3)

# silhouette metodas

sns.set\_context("notebook")

fig, ax = plt.subplots(1,2,figsize=(17, 6))

ax = ax.flatten()

df\_plot["cluster"], \_ = do\_kmeans(x,n\_clusters=2,random\_state=123)

df["cluster\_original"] = df\_plot["cluster"]

plot = sns.scatterplot(x="x",y="y",hue="cluster",data=df\_plot,palette="Dark2",ax=ax[0])

plot.set\_title("K-means clustering (original dimensionality data)")

df\_plot["cluster"], \_ = do\_kmeans(x\_small,n\_clusters=2,random\_state=123)

df["cluster\_reduced"] = df\_plot["cluster"]

plot = sns.scatterplot(x="x",y="y",hue="cluster",data=df\_plot,palette="Dark2",ax=ax[1])

plot.set\_title("K-means clustering (reduced dimensionality data)")

def statistics(df):

statistics\_1 = df.groupby("cluster\_original").describe().T

statistics\_1 = statistics\_1.reset\_index()[statistics\_1.reset\_index()["level\_1"].isin(["50%"])]

columns = list(statistics\_1.columns)

columns[0] = "variable"

columns[1] = "statistic"

for i in range(2,len(columns)):

columns[i] = str(columns[i]) + "\_original"

statistics\_1.columns = columns

statistics\_2 = df.groupby("cluster\_reduced").describe().T

statistics\_2 = statistics\_2.reset\_index()[statistics\_2.reset\_index()["level\_1"].isin(["50%"])]

columns = list(statistics\_2.columns)

columns[0] = "variable"

columns[1] = "statistic"

for i in range(2,len(columns)):

columns[i] = str(columns[i]) + "\_reduced"

statistics\_2.columns = columns

statistics = statistics\_1.merge(statistics\_2,how="inner")

statistics = statistics.reindex(sorted(statistics.columns), axis=1)

return statistic

# klasteriu palyginimas

print(df["cluster\_original"].value\_counts())

print(df["cluster\_reduced"].value\_counts())

statistics(df)

# kuo skiriasi klasteriai pagal desimtmecius

df\_plot["decade"] = df\_id["decade"]

sns.relplot(x="x",y="y",kind="scatter",data=df\_plot,col="cluster",hue="decade",palette="Set2")

df\_plot["energy"] = df["energy"]

sns.relplot(x="x",y="y",kind="scatter",hue="energy",size="energy",data=df\_plot,col="cluster",palette="Reds")

df\_plot["acousticness"] = df["acousticness"]

sns.relplot(x="x",y="y",kind="scatter",hue="acousticness",size="acousticness",data=df\_plot,col="cluster",palette="Greens",hue\_norm = (-10,100))

df\_plot["popularity"] = df["popularity"]

sns.relplot(x="x",y="y",kind="scatter",hue="popularity",size="popularity",data=df\_plot,col="cluster",palette="PuBuGn",

hue\_norm = (-10,100))

# #### Hierarchical

def do\_hierarchical(df, standartize = True,\*\*kwargs):

model = AgglomerativeClustering(\*\*kwargs)

pred = model.fit\_predict(x)

return pred, model

from scipy.cluster.hierarchy import dendrogram

def plot\_dendrogram(model, \*\*kwargs):

counts = np.zeros(model.children\_.shape[0])

n\_samples = len(model.labels\_)

for i, merge in enumerate(model.children\_):

current\_count = 0

for child\_idx in merge:

if child\_idx < n\_samples:

current\_count += 1 # leaf node

else:

current\_count += counts[child\_idx - n\_samples]

counts[i] = current\_count

linkage\_matrix = np.column\_stack(

[model.children\_, model.distances\_, counts]

).astype(float)

fig, ax = plt.subplots(1,1,figsize=(17, 6))

dendrogram(linkage\_matrix, \*\*kwargs, ax = ax)

\_, model = do\_hierarchical(x,distance\_threshold=0, n\_clusters=None,linkage="ward")

plot\_dendrogram(model, color\_threshold=0)

plt.title("Hierarchical Clustering Dendrogram Ward linkage")

\_, model = do\_hierarchical(x\_small,distance\_threshold=0, n\_clusters=None,linkage="ward")

plot\_dendrogram(model, color\_threshold=0)

plt.title("Hierarchical Clustering Dendrogram Ward linkage")

fig, ax = plt.subplots(1,2,figsize=(17, 6))

ax = ax.flatten()

df\_plot["cluster"], \_ = do\_hierarchical(x,n\_clusters=2,linkage="ward")

df["cluster\_original"] = df\_plot["cluster"]

plot = sns.scatterplot(x="x",y="y",hue="cluster",data=df\_plot,palette="Dark2",ax=ax[0])

plot.set\_title("Hierarchical clustering (original dimensionality data)")

df\_plot["cluster"], \_ = do\_hierarchical(x\_small,n\_clusters=2,linkage="ward")

df["cluster\_reduced"] = df\_plot["cluster"]

plot = sns.scatterplot(x="x",y="y",hue="cluster",data=df\_plot,palette="Dark2",ax=ax[1])

plot.set\_title("Hierarchical clustering (reduced dimensionality data)")

# #### DBSCAN

from sklearn.neighbors import NearestNeighbors

def do\_dbscan(df, \*\*kwargs):

model = DBSCAN(\*\*kwargs)

pred = model.fit\_predict(x)

return pred, model

x.shape, x\_small.shape

nn=NearestNeighbors(n\_neighbors=20).fit(x)

distances, indices = nn.kneighbors(x)

farthest = distances[:,-1]

fig, ax = plt.subplots(1,1,figsize=(5, 5))

plt.plot(np.sort(farthest))

plt.xlabel("index")

plt.ylabel("distance")

plt.show()

# ##### Originaliam duomenų rinkiniui: minSamples = 2n = 20 , eps = 3.5

nn=NearestNeighbors(n\_neighbors=min\_samples\_small).fit(x\_small)

distances, indices = nn.kneighbors(x\_small)

farthest = distances[:,-1]

fig, ax = plt.subplots(1,1,figsize=(5, 5))

plt.plot(np.sort(farthest))

plt.xlabel("index")

plt.ylabel("distance")

plt.show()

# ##### Sumažintos dimensijos duomenų rinkiniui: minSamples = 4, eps = 0.7

fig, ax = plt.subplots(1,2,figsize=(17, 6))

ax = ax.flatten()

df\_plot["cluster"], \_ = do\_dbscan(x,min\_samples=20,eps=3.5)

df\_noise = df\_plot[df\_plot["cluster"]==-1]

plot = sns.scatterplot(x="x",y="y",hue="cluster",data=df\_plot[df\_plot["cluster"]!=-1],palette="Dark2",ax=ax[0])

plot.scatter(df\_noise["x"],df\_noise["y"],c="grey",alpha=0.5)

plot.set\_title("DBSCAN clustering (original dimensionality data)")

df\_plot["cluster"], \_ = do\_dbscan(x\_small,min\_samples=4,eps=0.7)

df\_noise = df\_plot[df\_plot["cluster"]==-1]

plot = sns.scatterplot(x="x",y="y",hue="cluster",data=df\_plot[df\_plot["cluster"]!=-1],palette="Dark2",ax=ax[1])

plot.scatter(df\_noise["x"],df\_noise["y"],c="grey",alpha=0.5)

plot.set\_title("DBSCAN clustering (reduced dimensionality data)")