KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

INTELEKTIKOS PAGRINDAI (P176B101)

Komandinio darbo ataskaita.

Atliko:
IFF-1/6 gr. studentai
Lukas Kuzmickas
Gustas Miliukas

Priėmė:
jaun.asist. Nakrošis Arnas

KAUNAS 2024

Turinys

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

1. ĮVADAS

Šiame laboratoriniame darbe naudojamas rinkinys, kuriame pateikiamos „Vinho Verde“ vyno kokybės duomenų variantas. Duomenų rinkinys apibūdina įvairių vyne esančių cheminių medžiagų kiekį ir jų poveikį jo kokybei.

Išskirti tolydiniai duomenų rinkinio atributai: fixed acidity, volatile acidity, citric acid,

residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates,

alcohol. Išmetame atributą id.

Išskirti kategoriniai duomenų rinkinio atributai: quality.

Metodus, kuriuos planuojame naudoti:

- SOM algoritmas: SOM (savireguliuojantys žemėlapiai) algoritmas yra naudojamas duomenų vizualizavimui ir klasterizavimui. Jis organizuoja duomenis mažos dimensijos tinkle, kur kiekvienas neuronas prisitaiko prie panašių duomenų grupių, atskleisdamas paslėptas struktūras.
- K-vidurkių algoritmas: K-vidurkių algoritmas yra klasterizavimo metodas, kuris grupuoja duomenų taškus į K klasterių, minimizuodamas atstumus tarp taškų ir jų klasterių centrų (vidurkių). Iteratyviai perskaičiuoja centrus, kol jie stabilizuojasi.

Dažniausiai naudojama metrika tiek SOM, tiek K-vidurkių algoritmuose - Euklido atstumas, jame matuojame tiesinį atstumą tarp dviejų taškų daugiamatėje erdvėje, apskaičiuojant kvadratinių skirtumų sumos kvadratinę šaknį. Euklido atstumas yra populiarus dėl savo paprastumo ir efektyvumo.

Inercija (angl. "inertia") yra metrika, matuojanti kiekvieno taško atstumą iki artimiausio klasterio centro ir sumuojanti šiuos atstumus visiems taškams. Ji naudojama įvertinti, kaip gerai duomenų taškai yra susitelkę aplink klasterio centrus. Mažesnė inercija rodo geresnį klasterizavimą, nes duomenų taškai yra arčiau savo klasterio centro.

Silueto koeficientas (angl. "silhouette coefficient") vertina klasterizavimo kokybę, atsižvelgiant į du kriterijus:

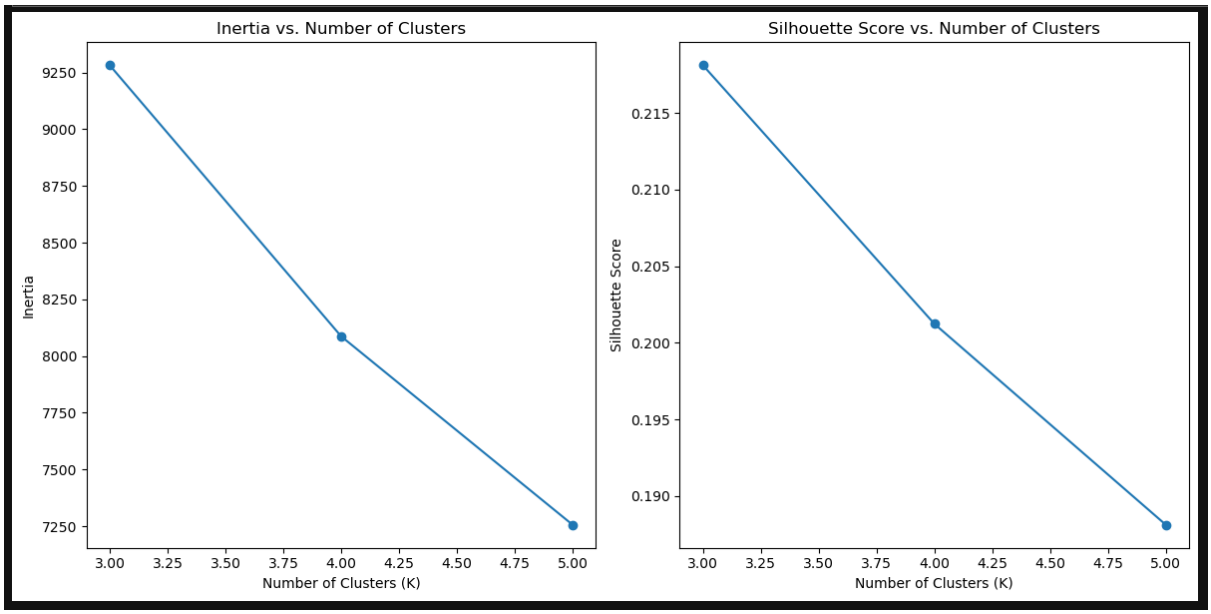
- Vidutinį atstumą tarp taško ir visų kitų to paties klasterio taškų (vidinis sanglaudumas).
- Vidutinį atstumą tarp taško ir kitų artimiausio klasterio taškų (atskirtumas).

Silueto koeficientas svyruoja nuo -1 iki 1. Aukštesnis koeficientas rodo geresnį klasterizavimą, nes taškas yra labiau susijęs su savo klasteriu nei su gretimu klasteriu.

Šie rodikliai padeda nustatyti optimalų klasterių skaičių ir vertinti klasterizavimo rezultatų kokybę.

2. K-VIDURKIŲ IR SOM ALGORITMAI

Pritaikome K-vidurkių algoritmą, mūsų tolydžioms įvesties reikšmėms:



Pav. 1 Viso duomenų rinkinio rezultatai nuo klasterių skaičiaus (K=3,4,5)

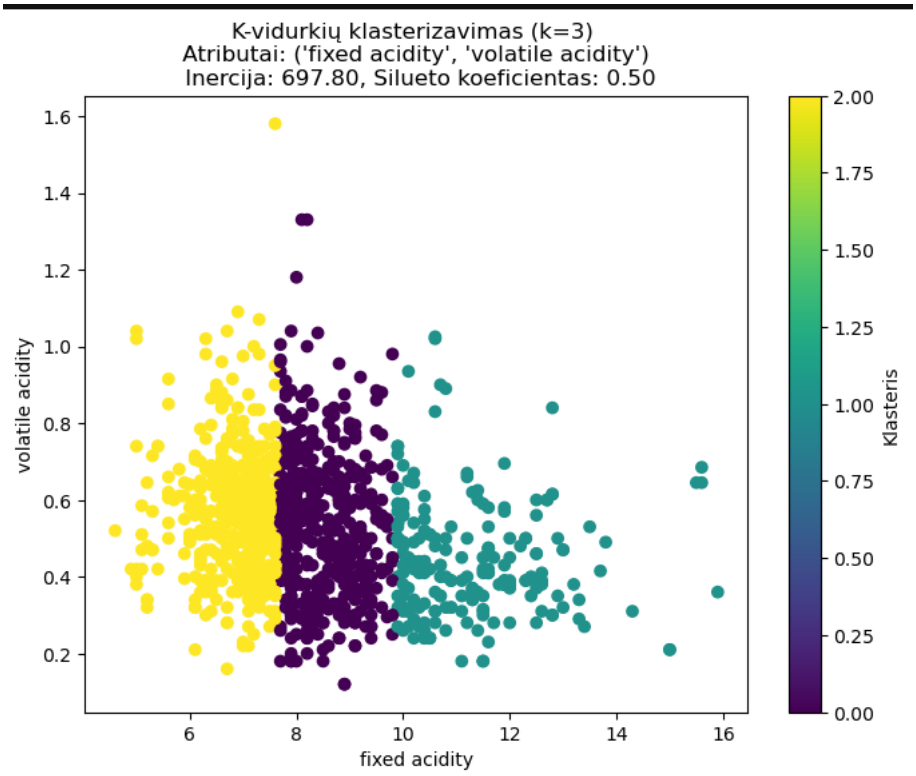
Sudarome mūsų įvesties atributų poras, kurias norėsime sugrupuoti:

```
attributes_combinations = [
    ("fixed acidity", "volatile acidity"),
    ("residual sugar", "chlorides"),
    ("total sulfur dioxide", "density"),
    ("alcohol", "free sulfur dioxide")
]
```

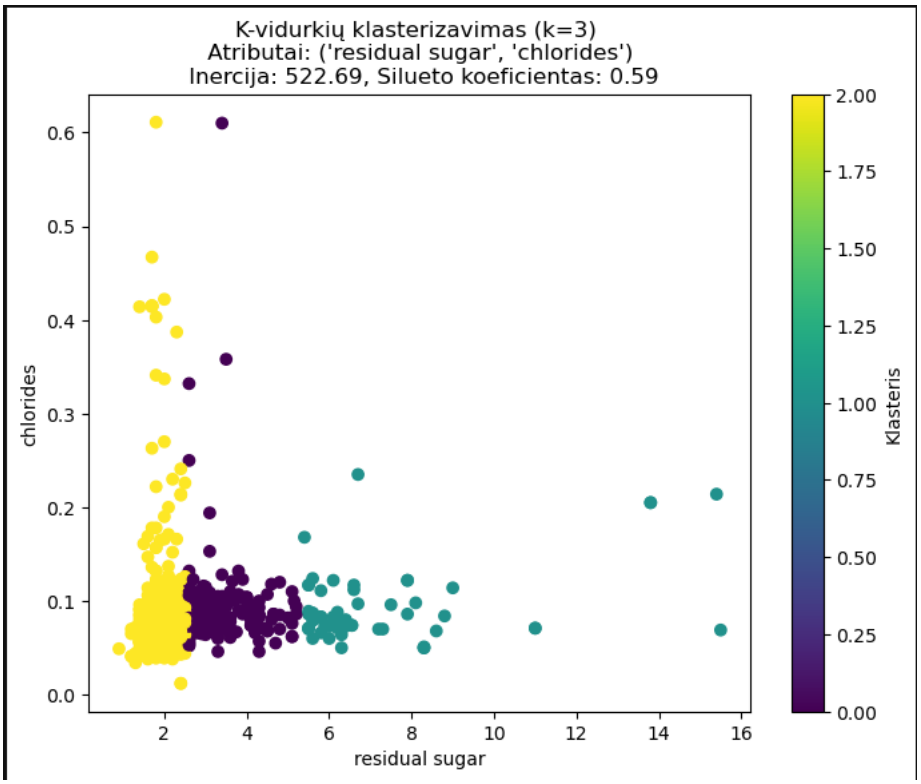
Pav. 2 Įvesties duomenų poros grupavimui.

Pasirinkti klasterių skaičiai, kurie bus keičiami (K=3,4,5).

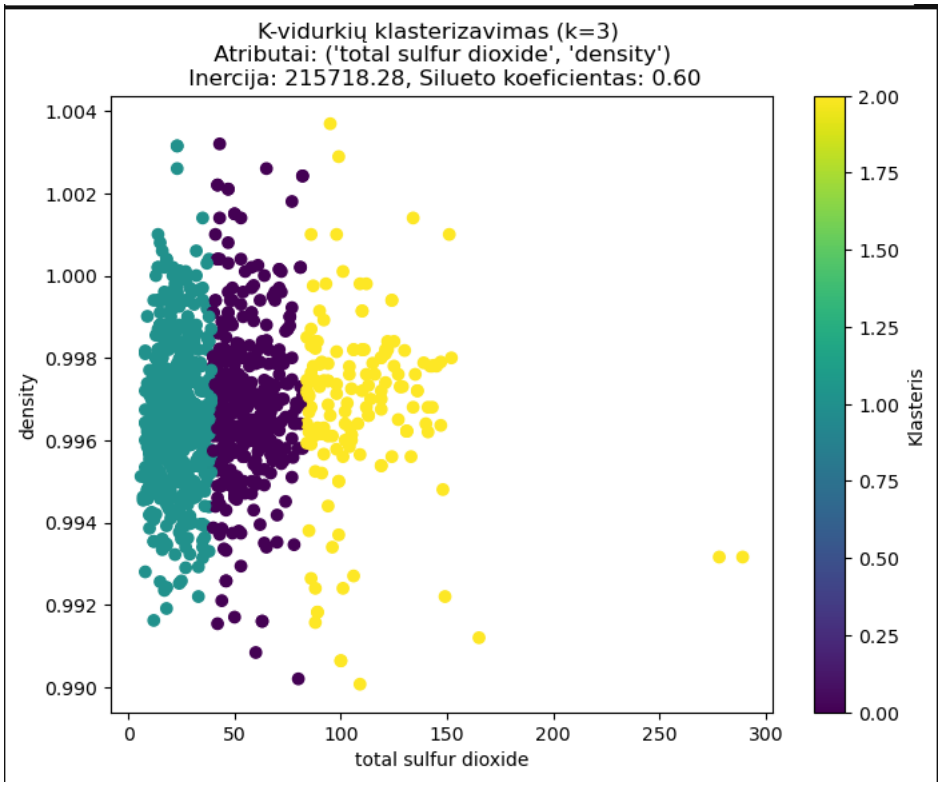
Kai mūsų klasterių skaičius yra K=3, rezultatai (naudojame K-vidurkių algoritmą):



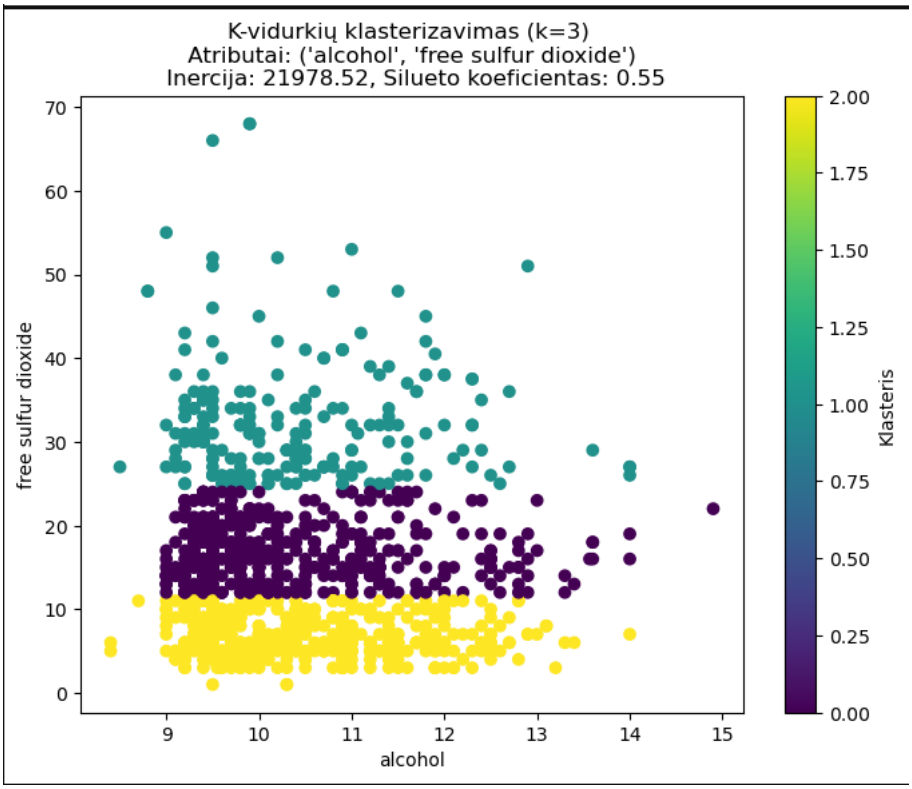
Pav. 3 K-vidurkių algoritmo 1-sios poros rezultatai, kai K=3.



Pav. 4 K-vidurkių algoritmo 2-sios poros rezultatai, kai K=3.

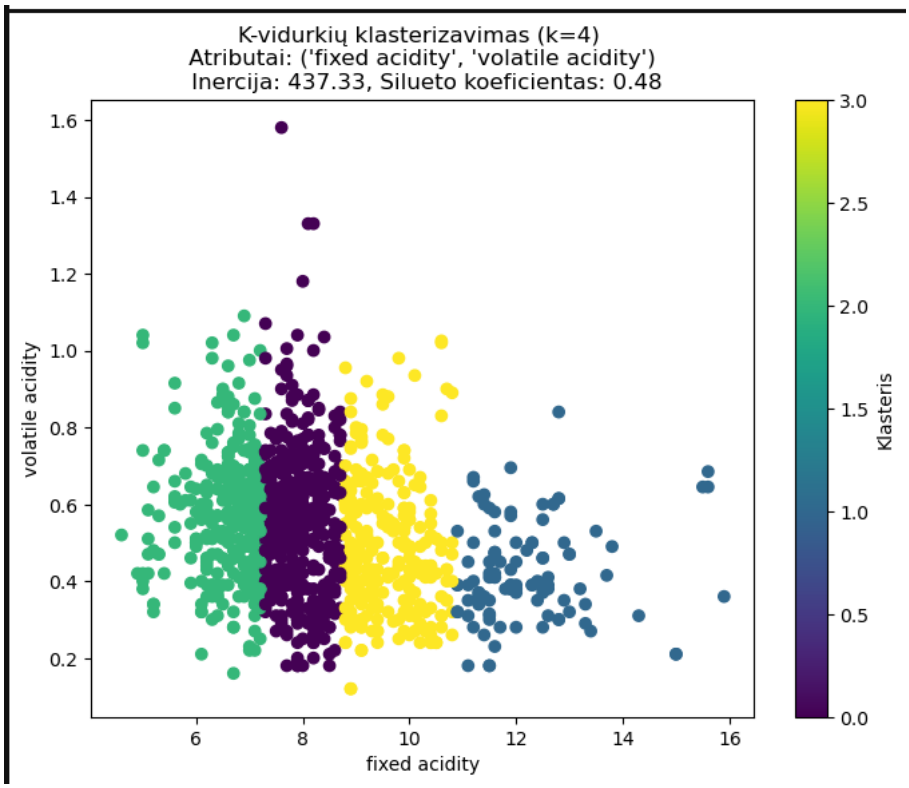


Pav. 5 K-vidurkių algoritmo 3-sios poros rezultatai, kai K=3.

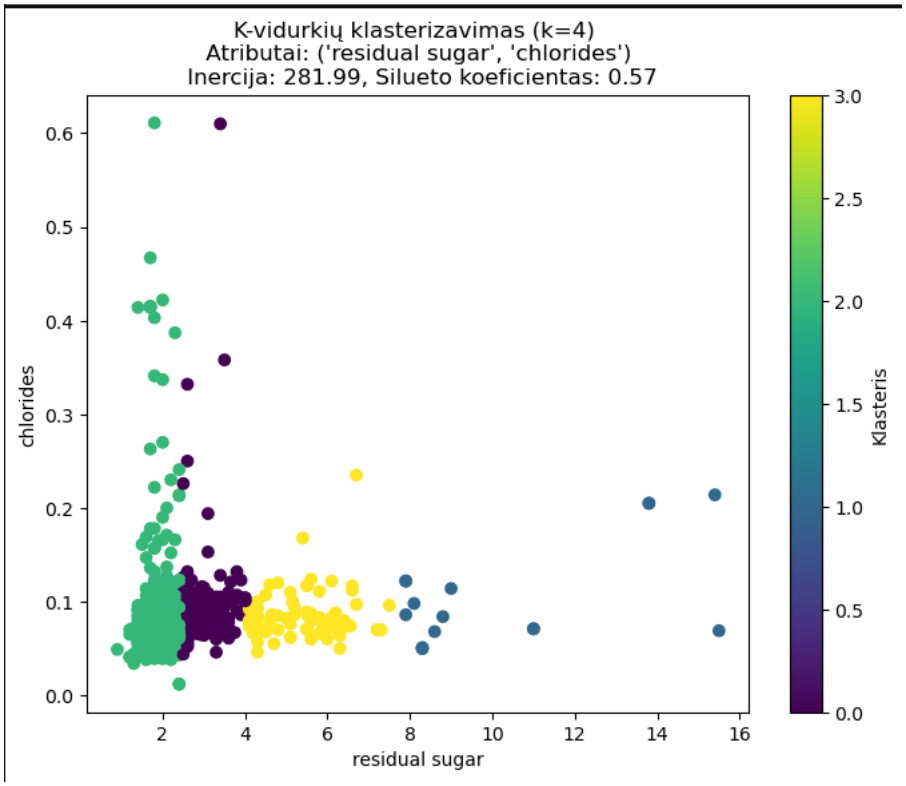


Pav. 6 K-vidurkių algoritmo 4-sios poros rezultatai, kai K=3.

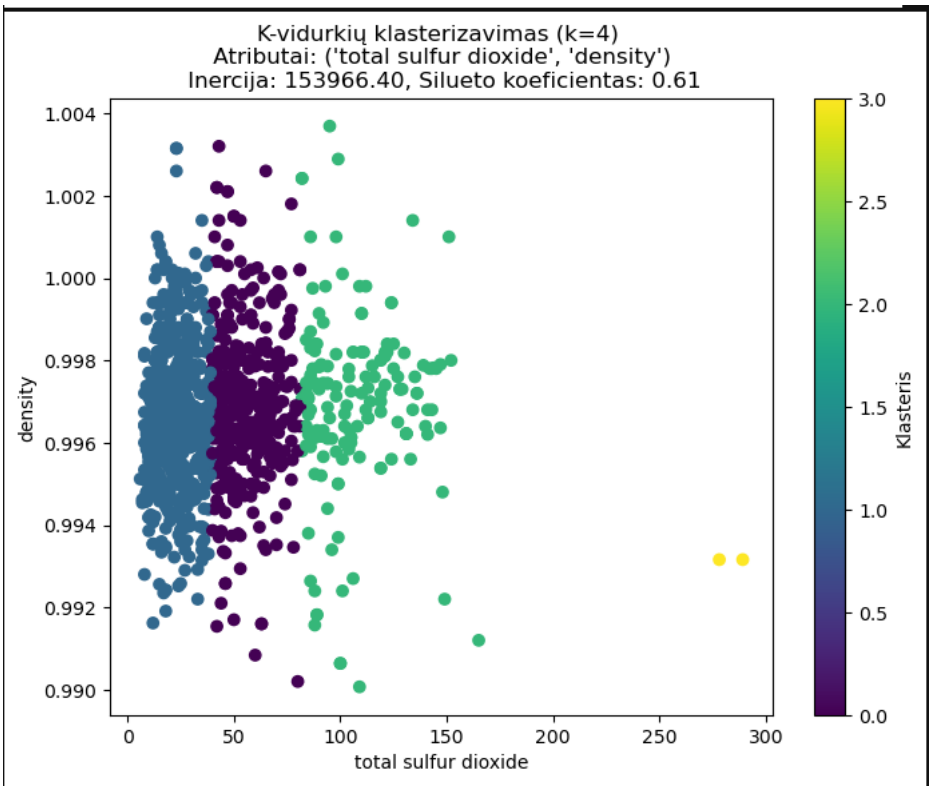
Kai mūsų klasterių skaičius yra K=4, rezultatai (naudojame K-vidurkių algoritmą):



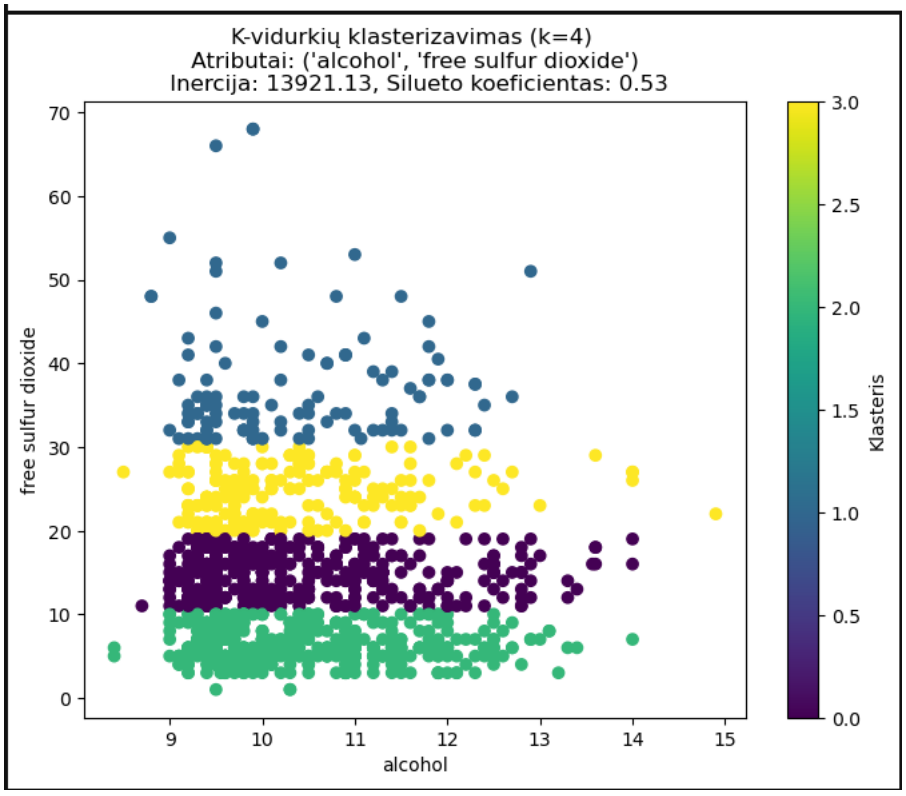
Pav. 7 K-vidurkių algoritmo 1-sios poros rezultatai, kai K=4.



Pav. 8 K-vidurkių algoritmo 2-sios poros rezultatai, kai K=4.

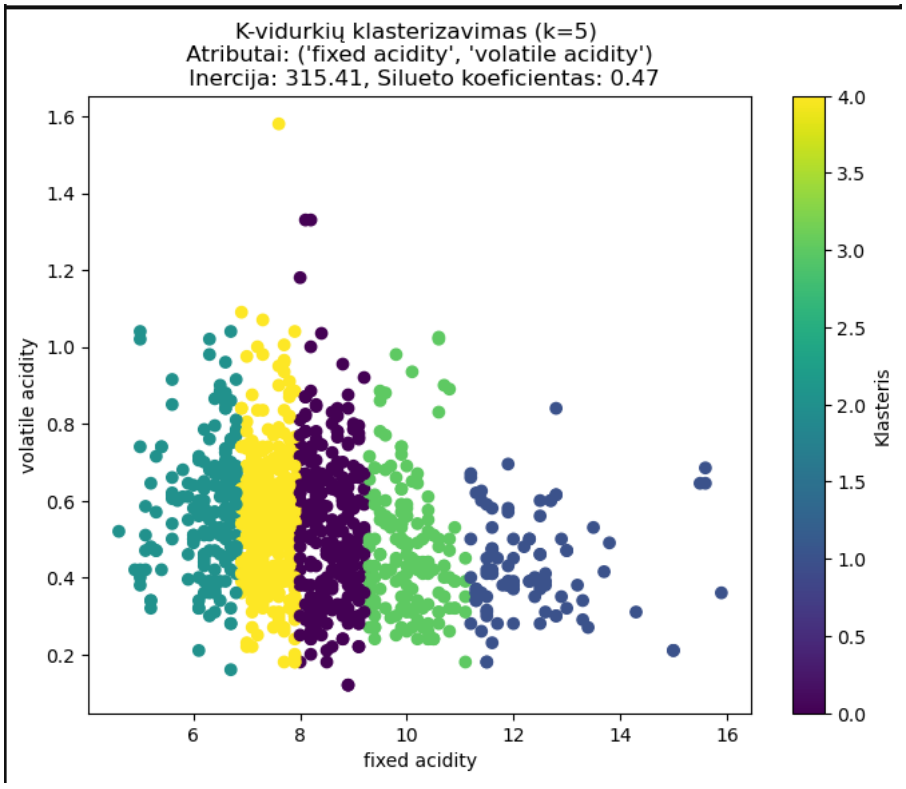


Pav. 9 K-vidurkių algoritmo 3-sios poros rezultatai, kai K=4.

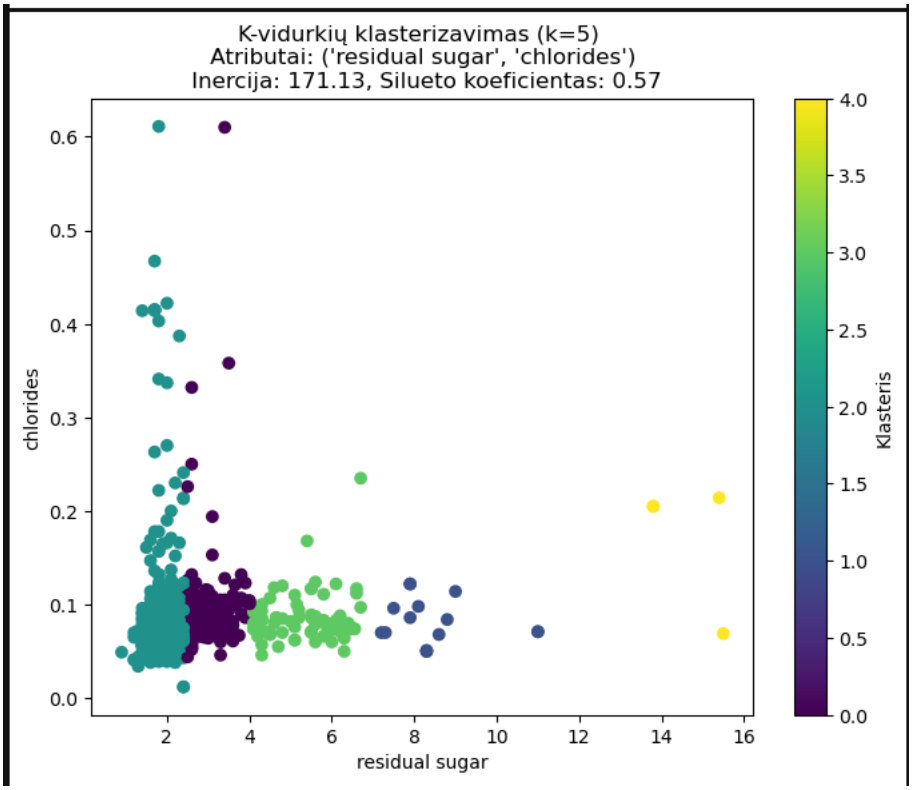


Pav. 10 K-vidurkių algoritmo 4-sios poros rezultatai, kai K=4.

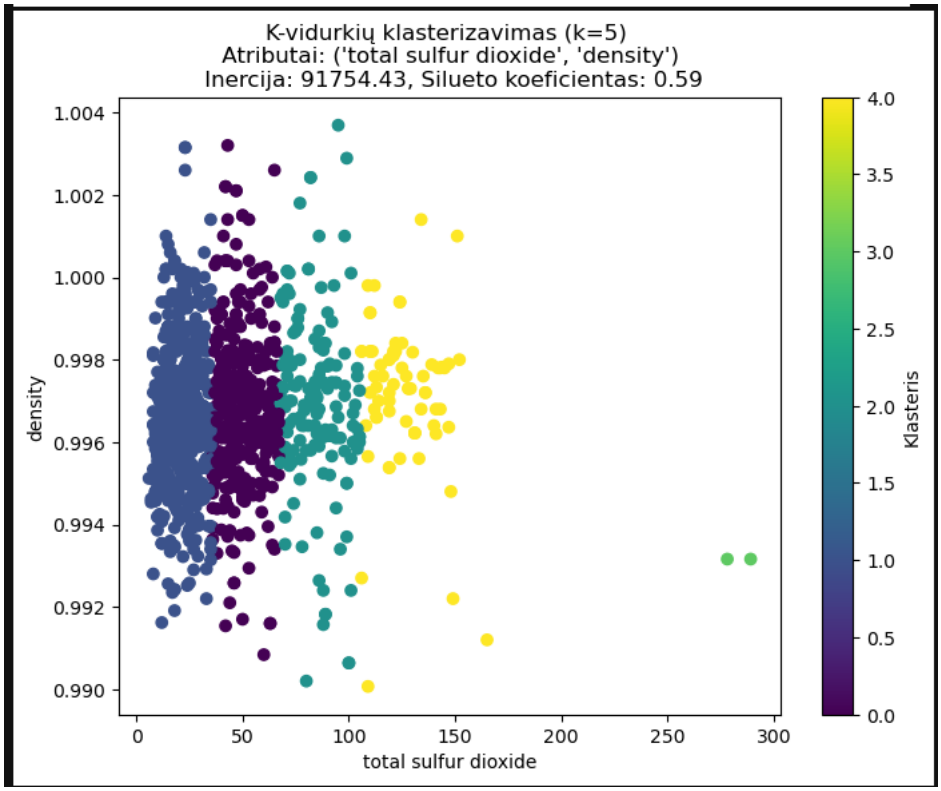
Kai mūsų klasterių skaičius yra K=5, rezultatai (naudojame K-vidurkių algoritmą):



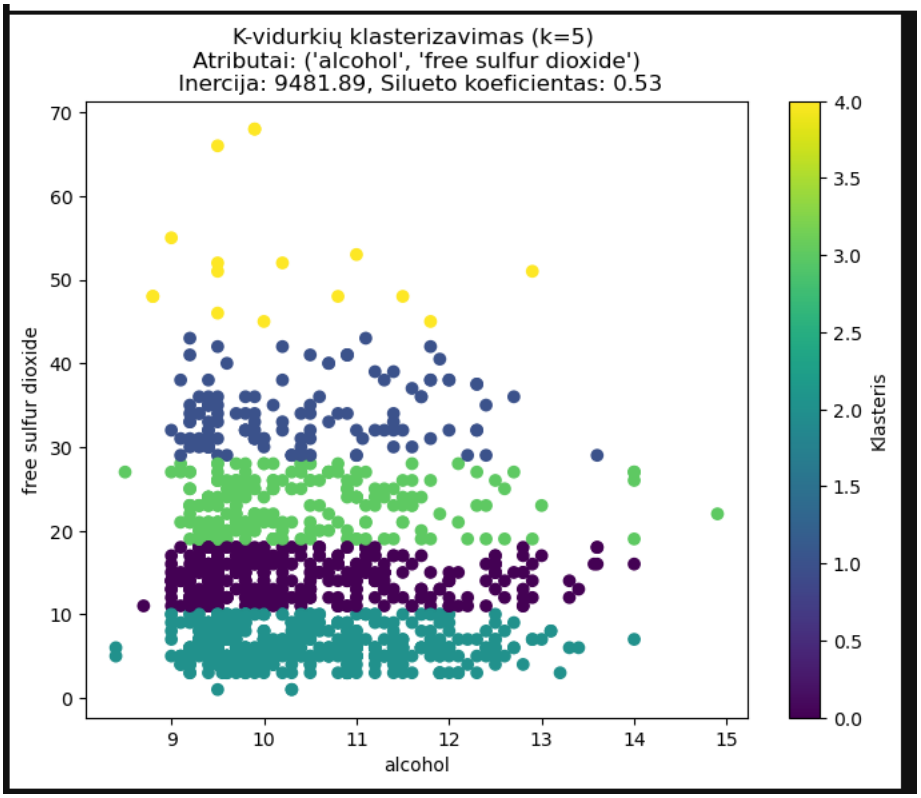
Pav. 11 K-vidurkių algoritmo 1-sios poros rezultatai, kai K=5.



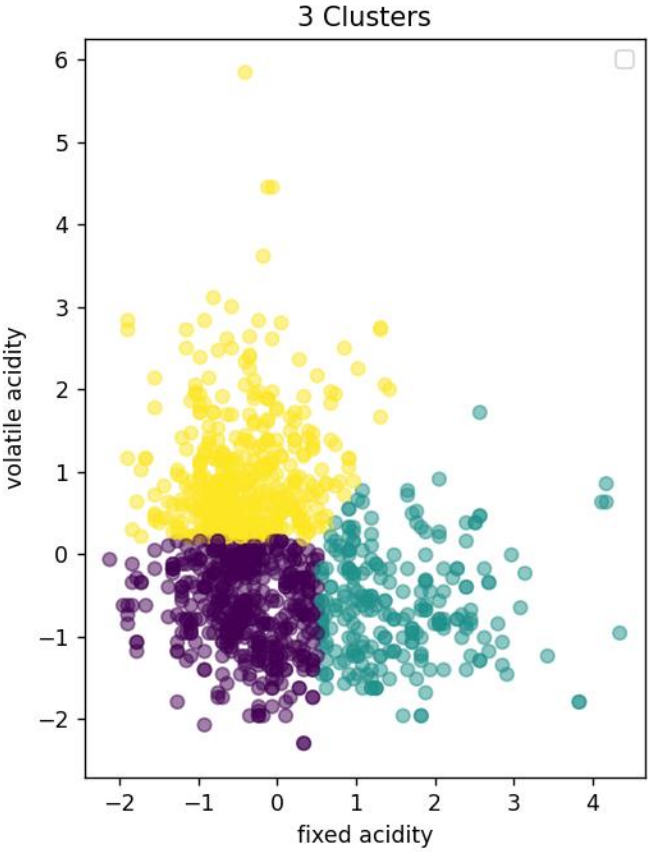
Pav. 12 K-vidurkių algoritmo 2-sios poros rezultatai, kai K=5.



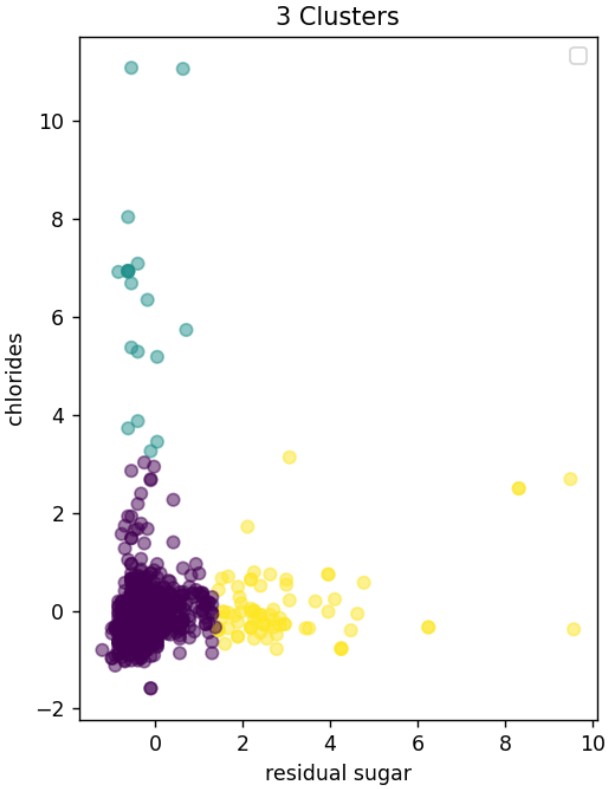
Pav. 13 K-vidurkių algoritmo 3-sios poros rezultatai, kai K=5.



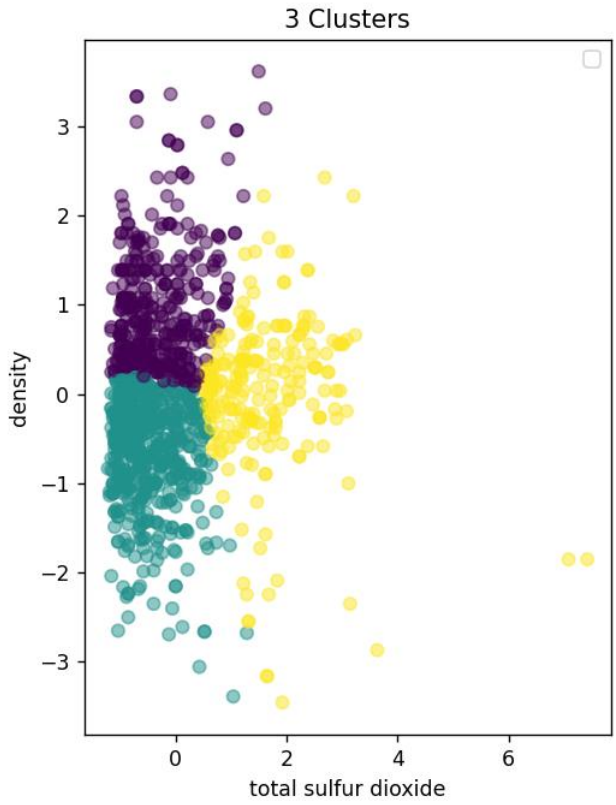
Pav. 14 K-vidurkių algoritmo 4-sios poros rezultatai, kai K=5.



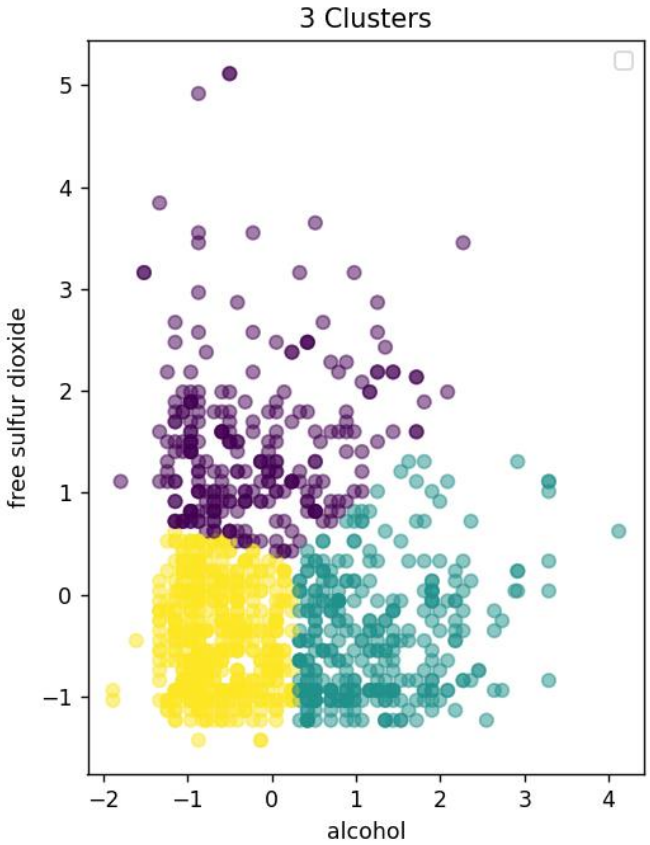
Pav. 15 SOM 1-osios poros rezultatai, kai K = 3



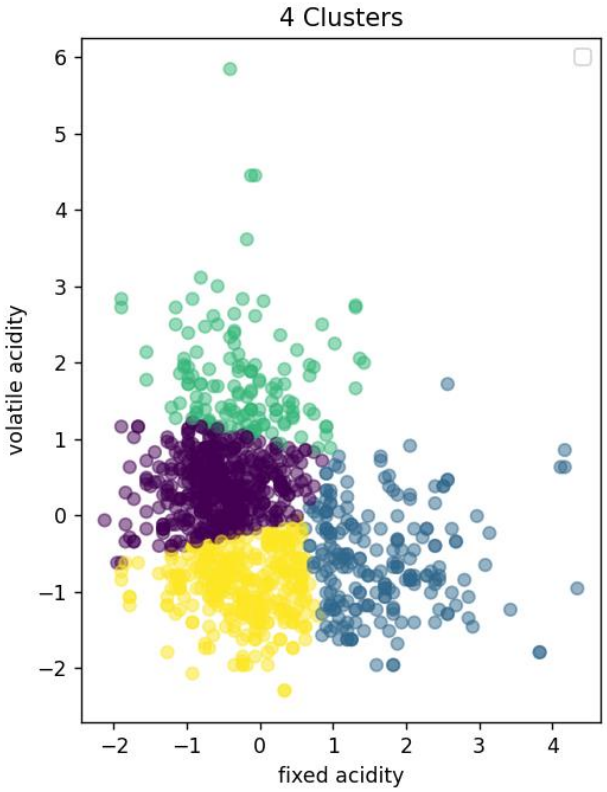
Pav. 16 SOM 2-osios poros rezultatai, kai K = 3



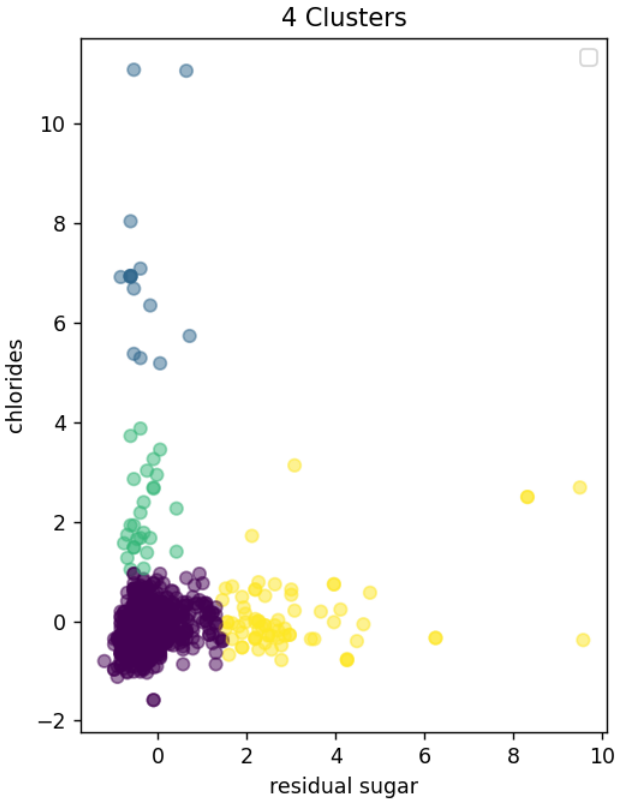
Pav. 17 SOM 3-osios poros rezultatai, kai K = 3



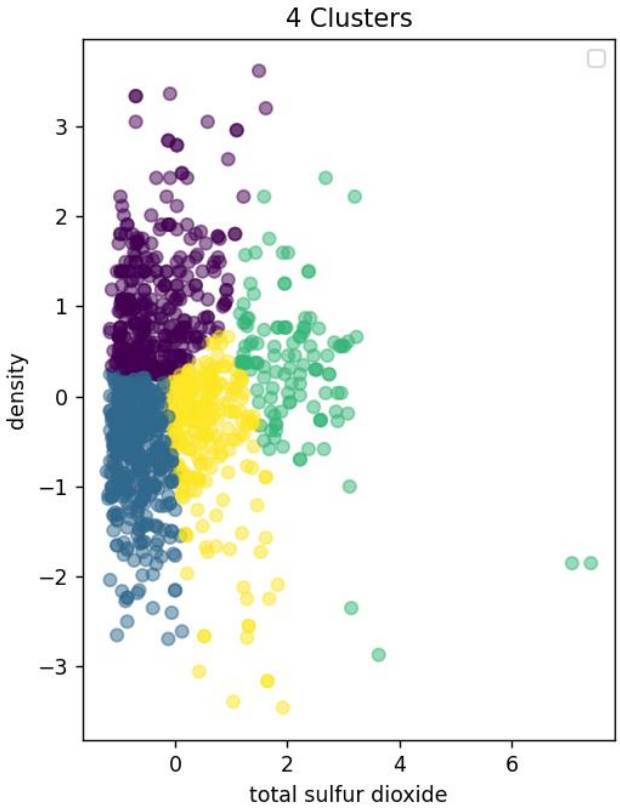
Pav. 18 SOM 4-osios poros rezultatai, kai K = 3



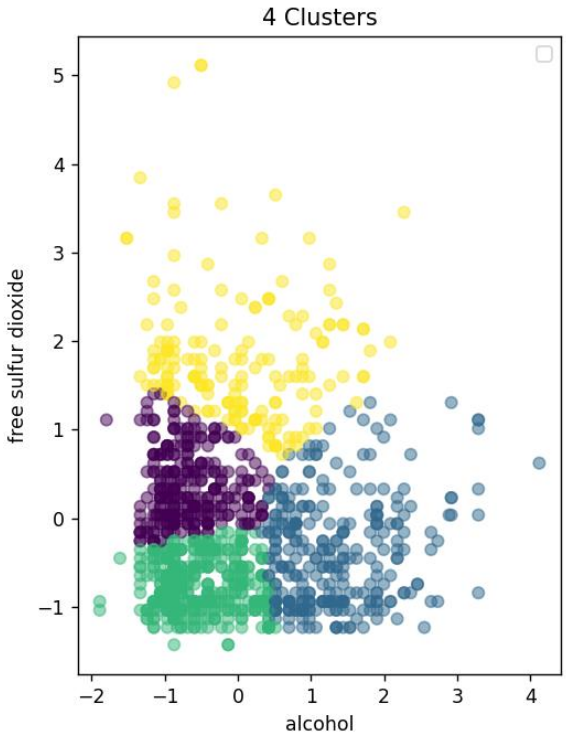
Pav. 19 SOM 1-osios poros rezultatai, kai K = 4



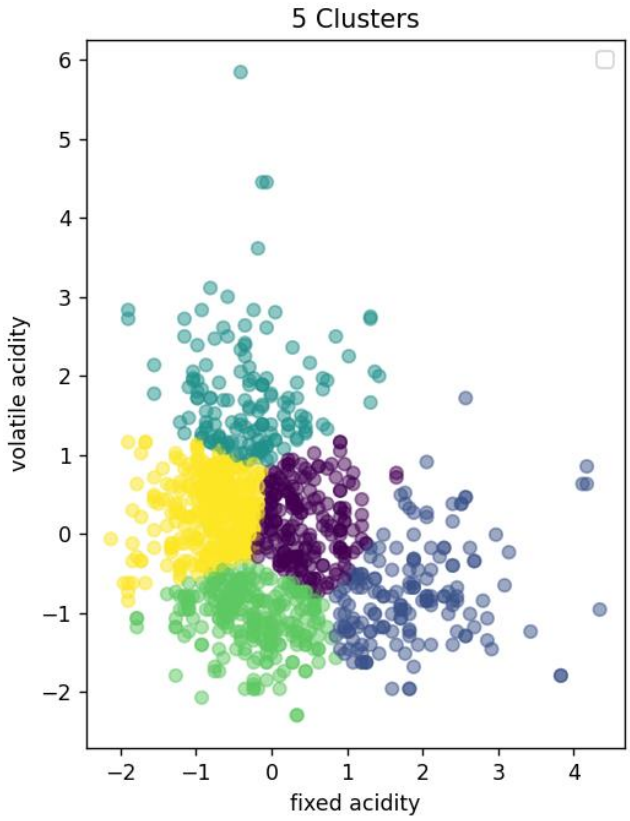
Pav. 20 SOM 2-osios poros rezultatai, kai K = 3



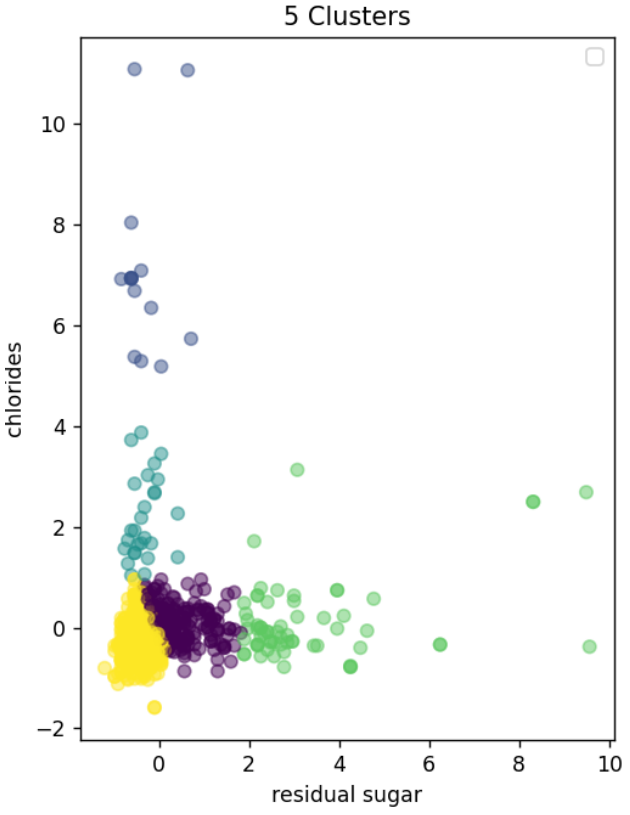
Pav. 21 SOM 3-osios poros rezultatai, kai K = 4



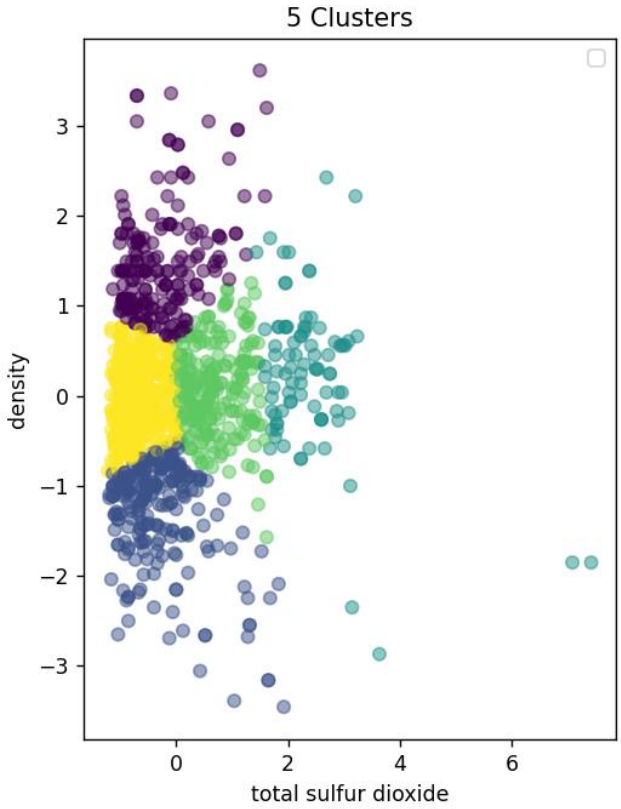
Pav. 22 SOM 4-osios poros rezultatai, kai K = 4



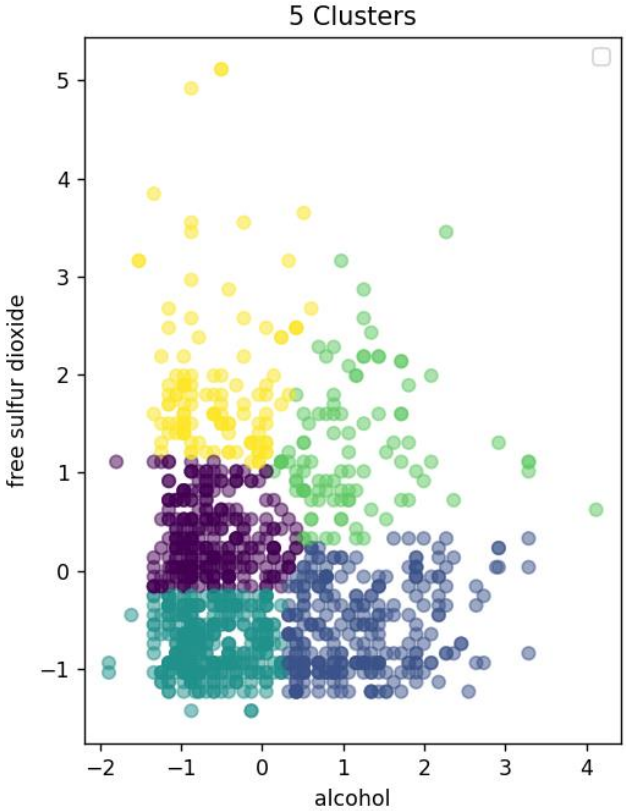
Pav. 23 SOM 1-osios poros rezultatai, kai K = 5



Pav. 24 SOM 2-osios poros rezultatai, kai K = 5



Pav. 25 SOM 3-osios poros rezultatai, kai K = 5



Pav. 26 SOM 4-osios poros rezultatai, kai K = 5

Rezultatai pateikiami lentelėje (žr. 1 lent.):

Lentelė 1 K-vidurkių (K=3,4,5) rezultatai.

Klasterių skaičius	Atributai	Inercija	Silueto koef.
3	('fixed acidity', 'volatile acidity')	697.80	0.50
3	('residual sugar', 'chlorides')	522.69	0.59
3	('total sulfur dioxide', 'density')	215718.28	0.60
3	('alcohol', 'free sulfur dioxide')	21978.52	0.55
4	('fixed acidity', 'volatile acidity')	437.33	0.48
4	('residual sugar', 'chlorides')	281.99	0.57
4	('total sulfur dioxide', 'density')	153966.40	0.61
4	('alcohol', 'free sulfur dioxide')	13921.13	0.53
5	('fixed acidity', 'volatile acidity')	315.41	0.47
5	('residual sugar', 'chlorides')	171.13	0.57
5	('total sulfur dioxide', 'density')	91754.43	0.59
5	('alcohol', 'free sulfur dioxide')	9481.89	0.53

Inercija mažėja didėjant klasterių skaičiui. Tai normalu, nes didėjant klasterių skaičiui, kiekvienas klasteris apima mažesnį duomenų taškų skaičių, o duomenų taškai yra arčiau savo klasterio centro. Tačiau labai mažos inercijos reikšmės gali reikšti, kad klasterių skaičius yra per didelis ir klasterizacija tampa per daug detalizuota (overfitting).

Silueto koeficientas išlieka gana stabilus, tačiau šiek tiek mažėja didėjant klasterių skaičiui. Tai gali reikšti, kad didinant klasterių skaičių, kai kurie klasteriai tampa mažiau aiškiai apibrėžti.

Aukščiausi silueto koeficientai matomi su atributų pora ('total sulfur dioxide', 'density'), kas gali reikšti, kad šie atributai labiausiai tinka klasterizavimui.

Lentelė 2 SOM algoritmo (K=3,4,5) rezultatai.

Klasterių skaičius	Atributai	Inercija	Silueto koef.
3	('fixed acidity', 'volatile acidity')	897.2338874490879	0.36906398060980733
3	('residual sugar', 'chlorides')	761.3961622476089	0.7507560503938105
3	('total sulfur dioxide', 'density')	975.1082369557818	0.3878601112998161
3	('alcohol', 'free sulfur dioxide')	846.6699326357589	0.42930808742796595
4	('fixed acidity', 'volatile acidity')	712.0180709345759	0.35769519279751816
4	('residual sugar', 'chlorides')	633.4432682350557	0.6908236665449374
4	('total sulfur dioxide', 'density')	819.4445711494948	0.3562288065199641
4	('alcohol', 'free sulfur dioxide')	701.4777838602441	0.3548787005407565
5	('fixed acidity', 'volatile acidity')	587.1668683255605	0.35439427899562004
5	('residual sugar', 'chlorides')	505.97733764697807	0.42982277061503094
5	('total sulfur dioxide', 'density')	644.4769107101107	0.35116371470780844
5	('alcohol', 'free sulfur dioxide')	575.9086910062229	0.37178981958728824

Pagal gautus rezultatus matome inercijos įvertis naudojant SOM yra ganėtinai žemas, kas parodo, kad SOM metodas naudojamus duomenys klasterizuoja gana efektyviai, o didinant klasterių skaičių inercija vis mažėja, nes tuomet klasterių dydžiai mažėja ir kiekvieno klasterio taško atstumas iki jo klasterio centro taip pat mažėja.

Tuo tarpu pagal gautus silueto koeficientus matome, kad klasterio taškai yra sunkiau atskiriami nuo kitų klasterių. Tai galime matyti ir gautuose grafikuose, nes visi klasteriai yra arti vienas kito. Šis koeficientas mažai kinta keičiantis klasterių skaičiui.

3. IŠVADOS

- Apibendrinant, SOM algoritmas geriau suklasterizuoja taškus;
- SOM greitis ir kokybė priklauso nuo iteracijų skaičiaus;
- Didėjant klasterių skaičiui mažėja inercija;
- K-vidurkių rezultatas labiau priklauso nuo klasterių skaičiaus, tuo tarpu SOM rezultatai mažiau priklauso nuo klasterių skaičiaus;
- Geriausiai klasterizuota atributų pora iš duomenų rinkinio: „**('residual sugar', 'chlorides')**“.

4. PRIEDAI

Kvidurkiai.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from minisom import MiniSom
FILE = "WineQT.csv"
NUMERICAL = ["fixed acidity", "volatile acidity", "citric acid", "residual sugar", "chlorides"
, "free sulfur dioxide",
             "total sulfur dioxide", "density", "pH", "sulphates", "alcohol"]
CATEGORICAL = ["quality"]
data = pd.read_csv(FILE)
data = data.drop(columns=['Id'])
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data[NUMERICAL])
cluster_numbers = [3, 4, 5]
attributes_combinations = [
    ("fixed acidity", "volatile acidity"),
    ("residual sugar", "chlorides"),
    ("total sulfur dioxide", "density"),
    ("alcohol", "free sulfur dioxide")
]
def perform_kmeans(data, attributes, n_clusters):
    X = data[list(attributes)].values
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    labels = kmeans.fit_predict(X)

    inertia = kmeans.inertia_
    silhouette_avg = silhouette_score(X, labels)
    return labels, inertia, silhouette_avg
for n_clusters in cluster_numbers:
    print(f"Klasterių skaičius: {n_clusters}")
    for attributes in attributes_combinations:
        labels, inertia, silhouette_avg = perform_kmeans(data, attributes, n_clusters)
        print(f"Atributai: {attributes}")
        print(f"Inercija: {inertia:.2f}")
        print(f"Silueto koeficientas: {silhouette_avg:.2f}")
        X = data[list(attributes)].values
        plt.figure(figsize=(8, 6))
        plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
        plt.xlabel(attributes[0])
        plt.ylabel(attributes[1])
        plt.title(f"K-vidurkių klasterizavimas (k={n_clusters}) \n Atributai: {attributes} \n
Inercija: {inertia:.2f}, Silueto koeficientas: {silhouette_avg:.2f}")
        plt.colorbar(label='Klasteris')
        plt.show()
```

SOM.py

```
import numpy as np
import pandas as pd
from matplotlib import colors, cm
from scipy.spatial import distance
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

```
from minisom import MiniSom
import matplotlib.pyplot as plt
from pylab import bone, pcolor, colorbar, plot, show

def train_som(data, som_shape=(10, 10), sigma=1.0, learning_rate=0.5, num_epochs=100):
    som = MiniSom(som_shape[0], som_shape[1], data.shape[1], sigma=sigma,
learning_rate=learning_rate)
    som.random_weights_init(data)
    som.train_batch(data, num_epochs)
    return som

def plot_cluster_distribution(ax, data, labels, attribute_pair):
    ax.scatter(data[:, 0], data[:, 1], c=labels, cmap='viridis', alpha=0.5)
    ax.set_xlabel(attribute_pair[0])
    ax.set_ylabel(attribute_pair[1])

def calculate_silhouette_inertia(data, labels):
    silhouette_avg = silhouette_score(data, labels)
    inertia = np.sum(np.min(np.array([np.sum((data - center) ** 2, axis=1) for center in
kmeans.cluster_centers_]), axis=0))
    return silhouette_avg, inertia

FILE = "WineQT.csv"
NUMERICAL = ["fixed acidity", "volatile acidity", "citric acid", "residual sugar",
"chlorides", "free sulfur dioxide",
             "total sulfur dioxide", "density", "pH", "sulphates", "alcohol"]
CATEGORICAL = ["quality"]


data = pd.read_csv(FILE)
data = data.drop(columns=['Id'])
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data[NUMERICAL])

cluster_numbers = [3, 4, 5]
for num_clusters in cluster_numbers:
    print(f"Training SOM for {num_clusters} clusters")
    som = train som(data scaled, som shape=(num clusters, num clusters), num epochs=2000)

attributes_combinations = [
    ("fixed acidity", "volatile acidity"),
    ("residual sugar", "chlorides"),
    ("total sulfur dioxide", "density"),
    ("alcohol", "free sulfur dioxide")
]

for pair in attributes_combinations:
    fig, axes = plt.subplots(1, len(cluster_numbers), figsize=(8 * len(cluster_numbers), 6))
    attribute_data = data[[pair[0], pair[1]]].values
    scaled_attribute_data = scaler.fit_transform(attribute_data)

    for i, num_clusters in enumerate(cluster_numbers):
        kmeans = KMeans(n_clusters=num_clusters, random_state=42)
        labels = kmeans.fit_predict(scaled_attribute_data)
        ax = axes[i] if len(cluster_numbers) > 1 else axes

        ax.legend()
        ax.set_title(f"{num_clusters} Clusters")
        plot_cluster_distribution(ax, scaled_attribute_data, labels, pair)

        silhouette_avg, inertia = calculate_silhouette_inertia(scaled_attribute_data, labels)
        print(f"Silhouette Score for {pair[0]} and {pair[1]} with {num_clusters} clusters:
{silhouette_avg}")
        print(f"Inertia for {pair[0]} and {pair[1]} with {num_clusters} clusters:
{inertia}\n")

    plt.xlabel(pair[0])
    plt.ylabel(pair[1])
    plt.show()
```