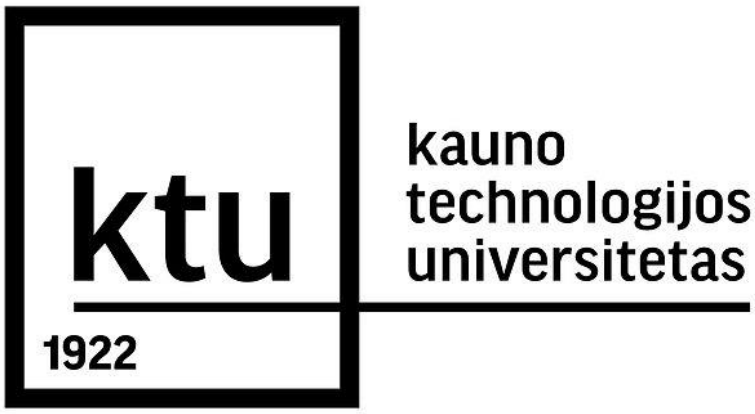


INFORMATIKOS FAKULTETAS

P176B101 Intelektikos pagrindai

komandinio darbo ataskaita

Studentas: Ignas Šakys, IFF-7/5
Dovydas Zamas, IFF-8/3

Dėstytoja: Doc. Agnė Paulauskaitė - Tarasevičienė

KAUNAS 2021

Turinys

1. Paveiksliukų sąrašas

2. Naudotas duomenų rinkinys

Darbui atlikti buvo pasirinktas pirmojo laboratorinio darbo metu naudotas duomenų rinkinys. Šiam rinkiniui atlikus duomenų analizę matome, jog dėl tuščių reikšmių laukuose, buvo pašalinta 143 eilutės ir duomenų rinkinyje liko 1110 eilučių su kuriomis dirbsime toliau. Žvelgiant į kitus rodiklius galime pastebėti, kad duomenys yra iškraipyti dėl duomenyse esančių neadekvačių reikšmių, jas sutvarkysime šalindami duomenų kokybės problemas.

2.1. Tolydinio tipo kintamieji

Atributo pavadiniams	Kiekis (Eilučių sk.)	Kiekis (Eilučių sk.) po panaikinimo	Trūkstamos reikšmės, %	Kardinalumas	Minimali reikšmė	Maksimali reikšmė	1-asis kvartilis	3-iasis kvartilis	Vidurkis	Mediana	Standartinis nuokrypis
Amžius	1253	1110	2,154828412	36	20	66	29	35	32,45045045	32	5,48
Patirtis (metais)	1253	1110	1,276935355	39	0	44352	5	12	487,1837838	8	4578,02
Bruto metinis atlyginimas	1253	1110	0	185	10001	500000000	59000	80000	521934,5414	70000	15005393,96
Atostogų dienos	1253	1110	5,426975259	34	0	365	27	30	28,4	28	11,02

pav. 1 Tolydinio tipo kintamųjų analizės lentelė prieš duomenų koregavimą

Atributo pavadiniams	Kiekis (Eilučių sk.)	Kiekis (Eilučių sk.) po panaikinimo	Trūkstamos reikšmės, %	Kardinalumas	Minimali reikšmė	Maksimali reikšmė	1-asis kvartilis	3-iasis kvartilis	Vidurkis	Mediana	Standartinis nuokrypis
Amžius	1110	1110	0	24	20	44	29	35	31,9684685	32	4,59
Patirtis (metais)	1110	1110	0	23	0	22	5	11	8,47837838	8	4,59
Bruto metinis atlyginimas	1110	1110	0	146	28800	110000	60000	78000	69078,3532	70000	15023,9
Atostogų dienos	1110	1110	0	12	23	34	27	30	28,254955	28	1,96

pav. 2 Tolydinio tipo kintamųjų analizės lentelė po duomenų koregavimo

2.2. Kategorinio tipo kintamieji

Atributo pavadinimas	Kiekis (Eilučių sk.)	Kiekis (Eilučių sk.) po panaikinimo	Trūkstamos reikšmės, %	Kardinalumas	Moda	Modos dažnumas	Moda, %	2-oji Moda	2-osios Modos dažnumas	2-oji Moda, %
Lytis	1253	1110	0,7980846	3	Male	939	85	Female	170	15
Miestas	1253	1110	0	102	Berlin	605	55	Munich	214	19
Pozicija	1253	1110	0,47885076	125	Software Engineer	352	32	Backend Developer	156	14
Darbo stažas	1253	1110	0,95770152	16	Senior	506	46	Middle	333	30
Pagrindinė technologija	1253	1110	10,0558659	6	1	1006	91	2	84	8
Kitos technologijos	1253	1110	12,5299282	13	1	255	23	4	168	15
Įsidarbinimo statusas	1253	1110	1,35674381	7	Full-time employee	1081	97	Self-employed (freelancer)	20	2
Sutarties trukmė	1253	1110	2,31444533	3	Unlimited contract	1055	95	Temporary contract	54	5
Pagrindinė kalba	1253	1110	1,27693536	13	English	915	82	German	167	15
Kompanijos dydis	1253	1110	1,43655227	5	1000+	409	37	101-1000	364	33
Kompanijos tipas	1253	1110	1,99521149	54	Product	697	63	Startup	222	20

pav. 3 Kategorinio tipo kintamųjų analizės lentelė prieš duomenų koregavimą

Atributo pavadinimas	Kiekis (Eilučių sk.)	Kiekis (Eilučių sk.) po panaikinimo	Trūkstamos reikšmės, %	Kardinalumas	Moda	Modos dažnumas	Moda, %	2-oji Moda	2-osios Modos dažnumas	2-oji Moda, %
Lytis	1110	1110	0	2	Male	940	85	Female	170	15
Miestas	1110	1110	0	5	Berlin	801	72	Munich	214	19
Pozicija	1110	1110	0	8	Software Engineer	706	64	Backend Developer	156	14
Darbo stažas	1110	1110	0	5	Senior	519	47	Middle	333	30
Pagrindinė technologija	1110	1110	0	6	1	1006	91	2	84	8
Kitos technologijos	1110	1110	0	13	1	255	23	4	168	15
Įsidarbinimo statusas	1110	1110	0	3	Full-time employee	1085	98	Self-employed (freelancer)	20	2
Sutarties trukmė	1110	1110	0	2	Unlimited contract	1056	95	Temporary contract	54	5
Pagrindinė kalba	1110	1110	0	3	English	932	84	German	167	15
Kompanijos dydis	1110	1110	0	5	1000+	409	37	101-1000	364	33
Kompanijos tipas	1110	1110	0	3	Product	761	69	Startup	222	20

pav. 4 Kategorinio tipo kintamųjų analizės lentelė po duomenų koregavimo

3. SOM realizuojantys metodai

Pirmiausia nuskaitomi ir sutvarkomi duomenys iš pasirinkto duomenų failo. Toliau inicijuojama SOM klasė ir kviečiamos šios klasės metodas “train()“. Galiausiai gautas SOM žemėlapis atvaizduojamas grafiškai.

Norint realizuoti SOM bus vykdomi sekantys žingsniai:

1. SOM modelio iniciavimas, kai modeliui paduodamos žemėlapio dimensijų dydžiai, mokymosi greitis, bei nuskaityti ir į duomenų rinkinį suformuoti duomenys. Programa pasinaudodama jai suteiktais kintamaisiais, sugeneruos žemėlapio duomenų struktūrą, kurioje bus inicijuojamos atsitiktinės vektorių reikšmės intervale nuo 0 iki 1.
2. Išrenkamas atsitiktinis vektorius iš įvesties duomenų rinkinio.
3. Skaičiuojamas BMU (*angl. Best Matching Unit*) arba kitaip dar vadinamas neuronas – nugalėtojas. Šio neurono vektorius yra labiausiai panašus į atsitiktinai parinkto vektoriaus iš įvesties duomenų. Neuronas – nugalėtojas išrenkamas naudojant atstumo tarp dviejų vektorių apskaičiavimo metodus, šiuo atveju metodas remiasi Euklido atstumo formule.
4. Žinant BMU yra apskaičiuojamas vadinamosios neurono kaimynystės dydis. Kitaip tariant surandami šalia šio neurono esantys neuronai.
5. BMU ir kaimynystei priklausančių neuronų svorių perskaičiavimas.
6. Mokymosi greičio mažinimas.
7. Žingsniai nuo 2 iki 6 vykdomi tol, kol suformuojamas galutinis SOM žemėlapis.

4. K-vidurkių algoritmo realizuojantys metodai

1. Pirmiausiai nusiskaitome tolydinio tipo stulpelius iš duomenų rinkinio
2. Pasirenkame klasterių skaičių
3. Pasirenkame centroidus
4. Paskaičiuojame visų taškų *i* atstumus iki kiekvieno centroido
5. Kiekvienam taškui *i* priskiriame klasterį *m* iki kurio atstumas yra mažiausias
6. Perskaičiuojame klasterio *m* coordinates
7. Kartojame 4-6 žingsnius iki tol, kol nebekinta taškams priskirti klasteriai bei centroidų koordinatės.

Pastaba: Šiems žingsniams atlikti buvo pasinaudota python bibliotekomis:

- from sklearn.cluster import KMeans
- from sklearn.metrics import silhouette_score
- from yellowbrick.cluster import SilhouetteVisualizer

5. Atstumo metrikos

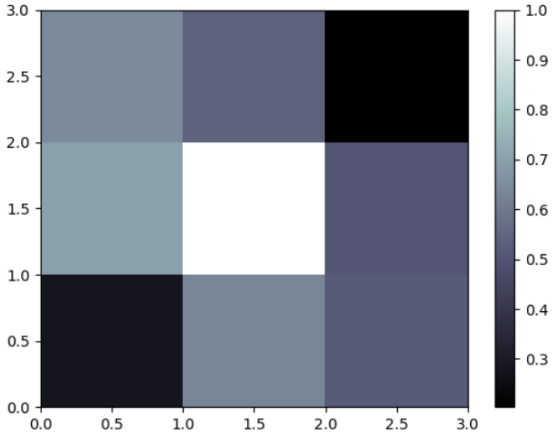
Šio darbo realizacijoje naudotas Euklido atstumas, kitaip tariant atstumo tarp parinkto neurono ir įvesties vektoriaus kvadratas. Nors šiuo konkrečiu atveju Euklido atstumo skaičiavimo metodas ir atlieka skaičiavimus korektiškai, tačiau didėjant duomenų kiekiui, kai kalbama apie milžiniškus duomenų rinkinius, specialistai renkasi kitokius alternatyvius metodus.

6. Atlikti eksperimentai

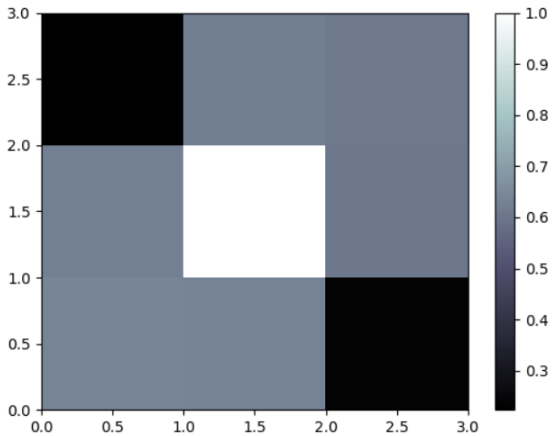
Prieš atliekant eksperimentus, buvo priimtos sekančių konstantų reikšmės, bei paaiškinimai:

- Mokymosi greičio pradinė reikšmė – 0.5;
- Epochų skaičius – 250;
- Kuo grafiko langelio spalva tamsesnė, tuo atstumas tarp neuronų yra mažesnis;
- Kuo grafiko langelio spalva šviesesnė, tuo atstumas tarp neuronų yra didesnis;
- Tamsesnių langelių susitelkimas atvaizduoja susiformavusius klasterius;
- Šviesesnių langelių susitelkimas atvaizduoja atskirtį tarp susiformavusių klasterių;

Pirmojo bandymo metu klasterių skaičius buvo lygus 3. Programa kompiliuota du kartus, gauti du skirtingi SOM žemėlapių grafikai pavaizduoti 5 ir 6 paveikslėliuose.

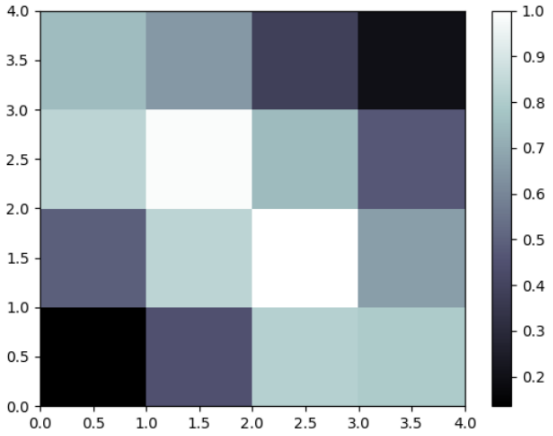


pav. 5 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 3

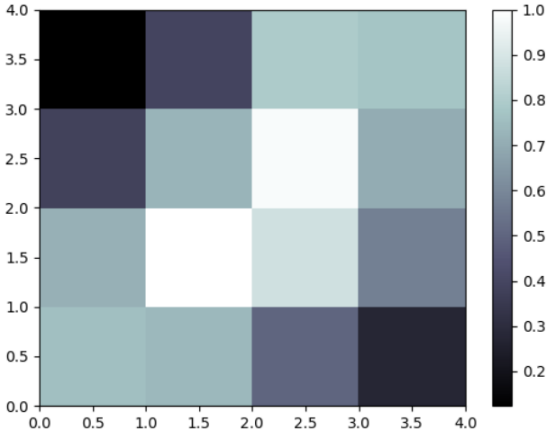


pav. 6 Antrojo bandymo SOM žemėlapio grafikas, kai k = 3

Antrojo bandymo metu klasterių skaičius buvo lygus 4. Programa kompiliuota du kartus, gauti du skirtingi SOM žemėlapių grafikai pavaizduoti 7 ir 8 paveikslėliuose.

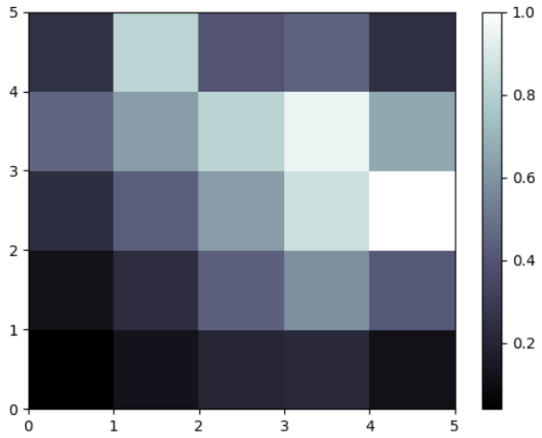


pav. 7 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 4

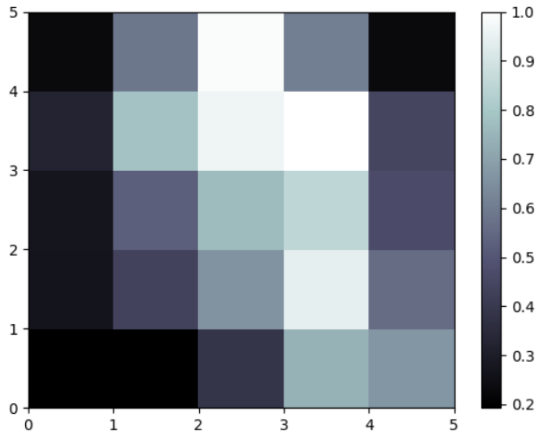


pav. 8 Antrojo bandymo SOM žemėlapio grafikas, kai k = 4

Trečiojo bandymo metu klasterių skaičius buvo lygus 5. Programa kompiliuota du kartus, gauti du skirtingi SOM žemėlapių grafikai pavaizduoti 9 ir 10 paveikslėliuose.

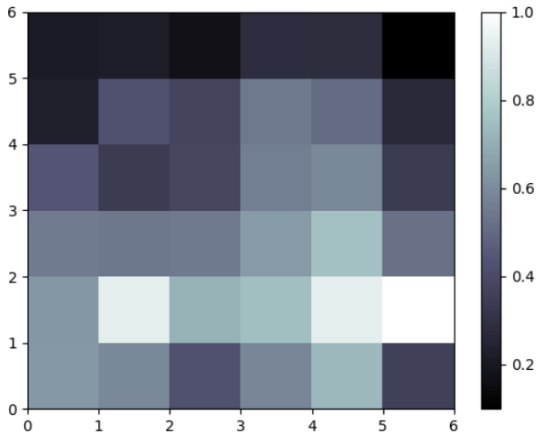


pav. 9 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 5

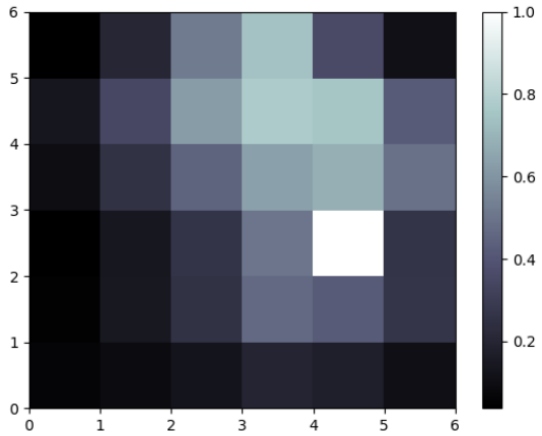


pav. 10 Antrojo bandymo SOM žemėlapio grafikas, kai k = 5

Ketvirtojo bandymo metu klasterių skaičius buvo lygus 5. Programa kompiliuota du kartus, gauti du skirtingi SOM žemėlapių grafikai pavaizduoti 11 ir 12 paveikslėliuose.



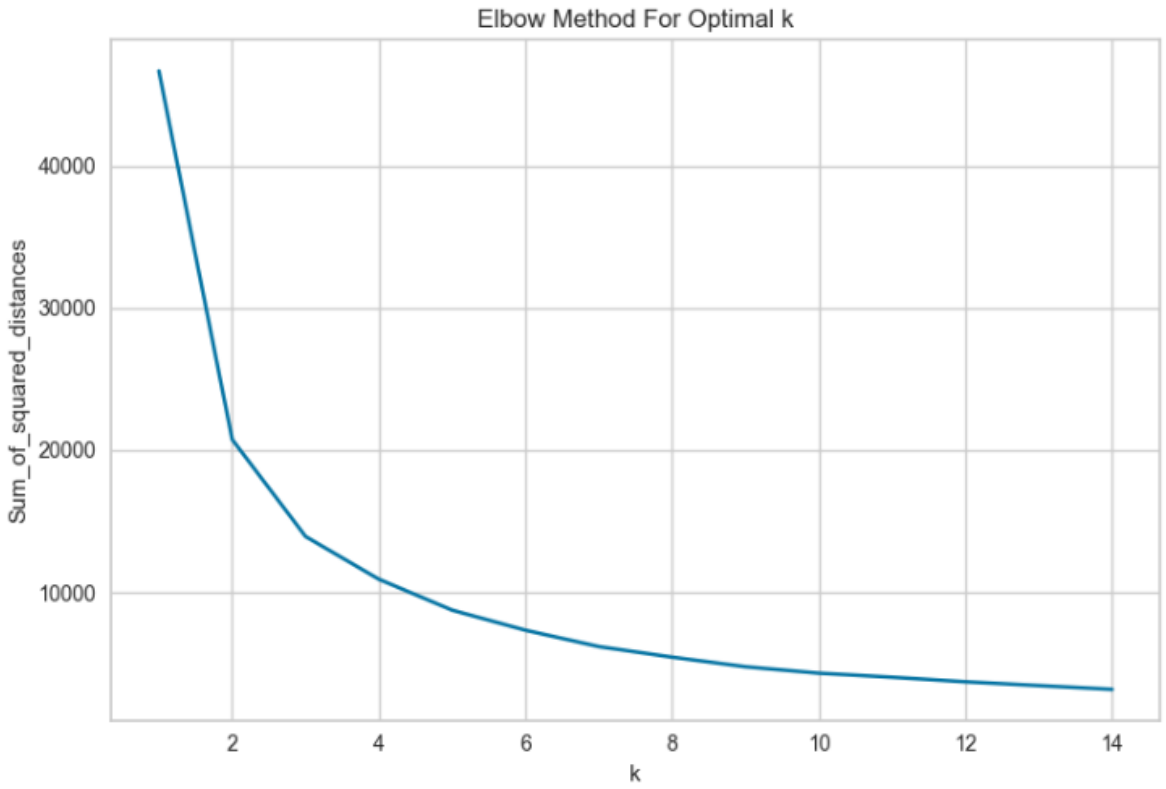
pav. 11 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 6



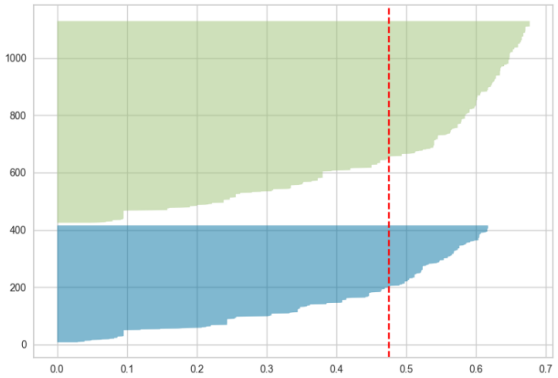
pav. 12 Antrojo bandymo SOM žemėlapio grafikas, kai k = 6

K-vidurkių algoritmui pirmąjam ekspermentui buvo pasirinkti „Age“ ir „Total years of experience“ atributai.

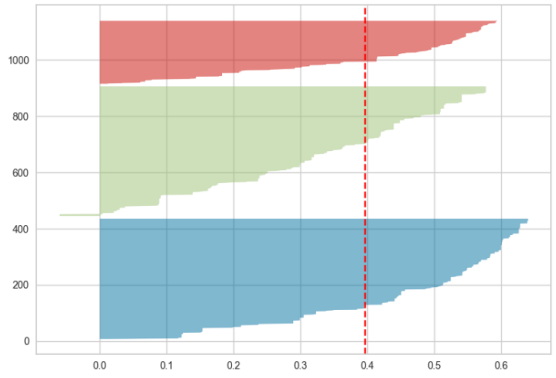
Silhouetter Score: 0.475 – silueto koeficientas “Age” ir “Total years of experience” atributams



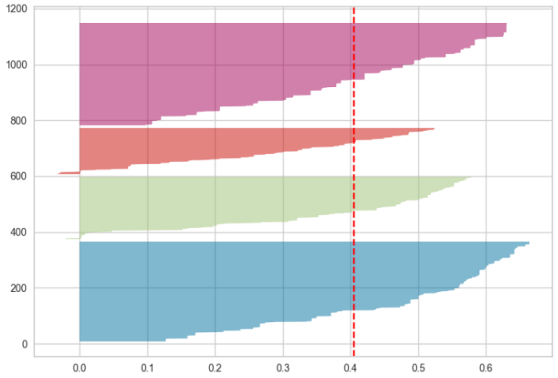
pav. 13 Atributų "Age" ir "Total years of experience" inercijos ir klasterių priklausomybės kreivė



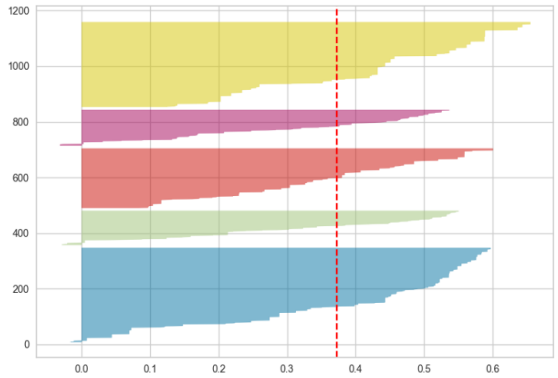
a) Silueto analizė kai k=2



b) Silueto analizė kai k=3

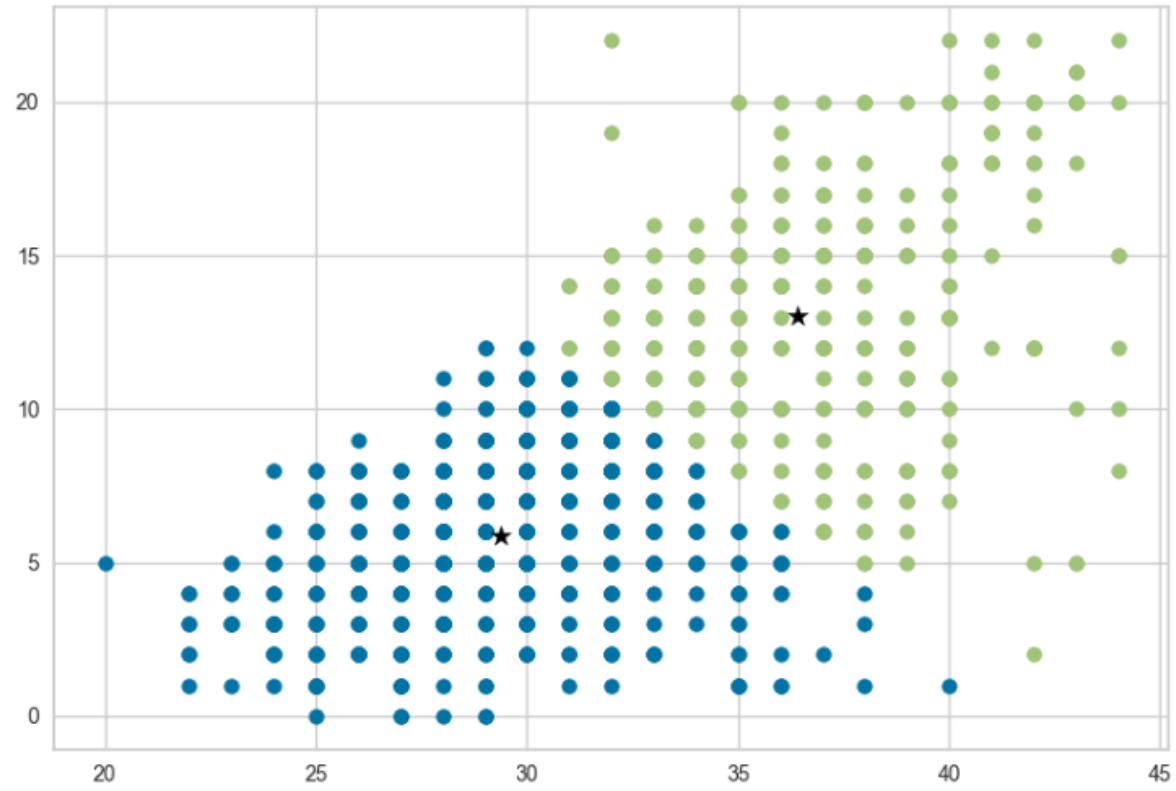


c) Silueto analizė kai k=4

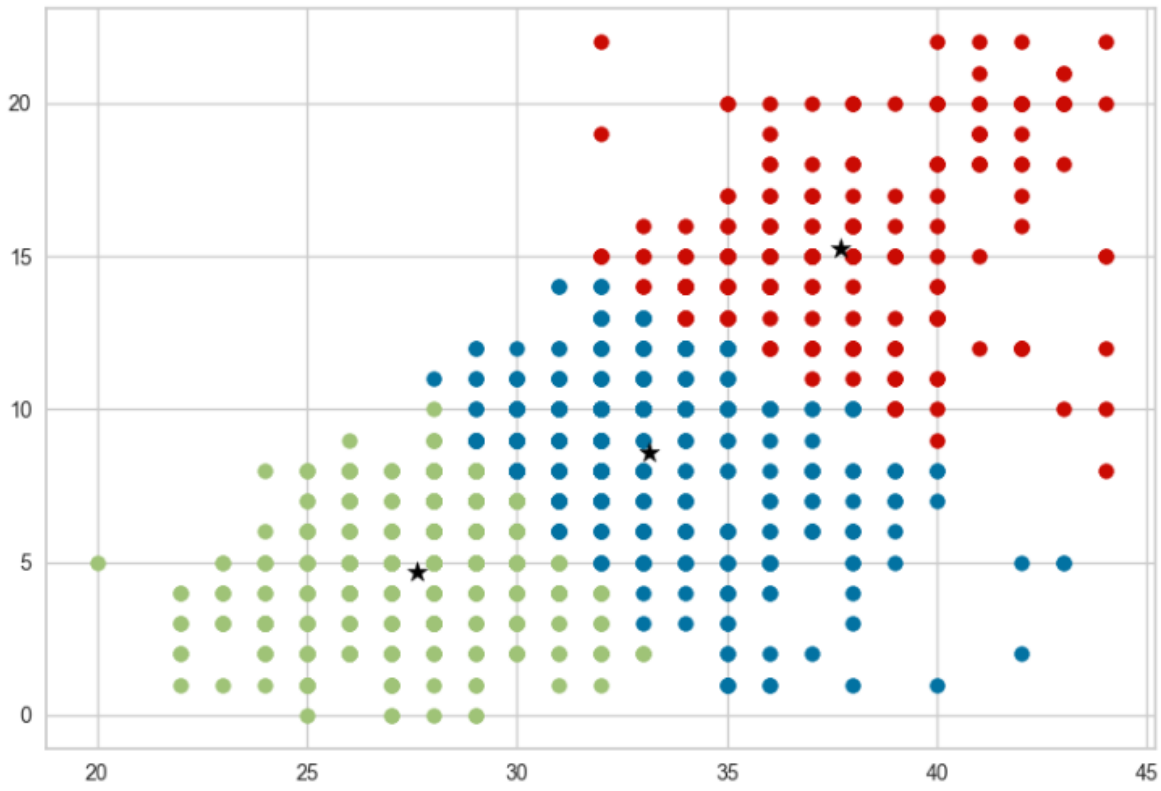


d) Silueto analizė kai k=5

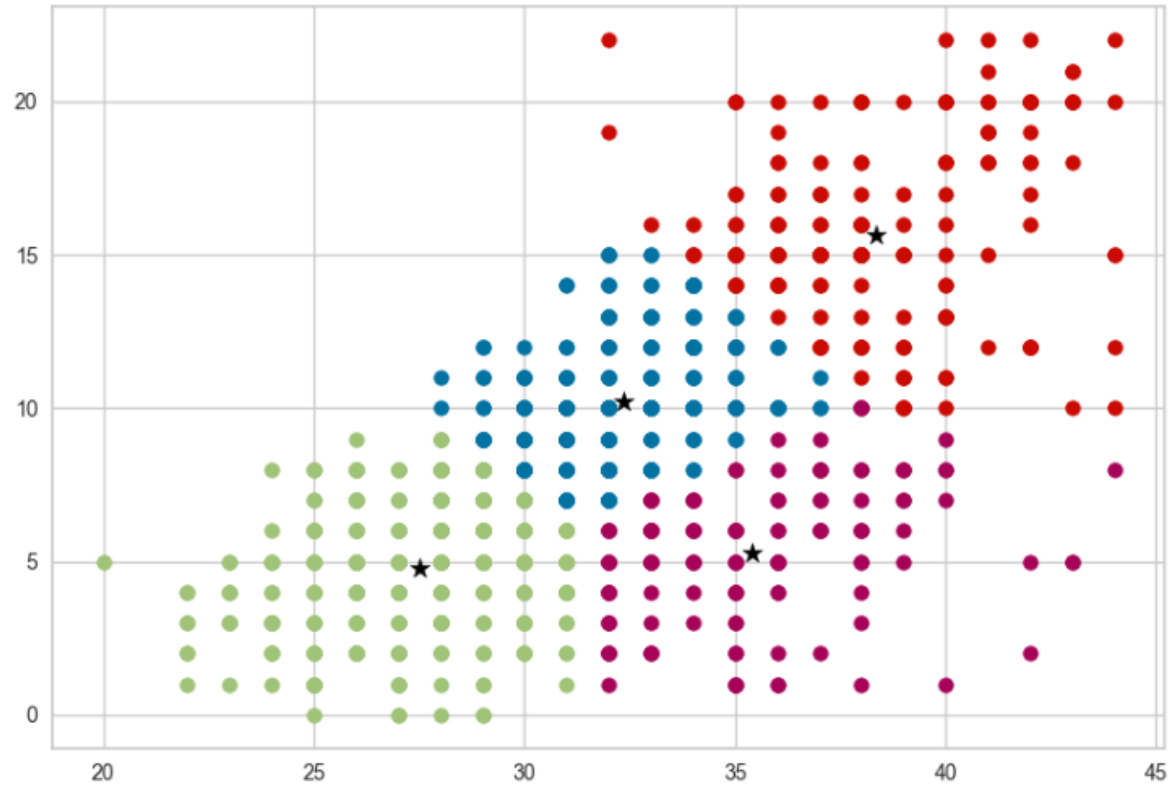
pav. 14 Siluetų analizė "Age" ir "Total years of experience" atributams



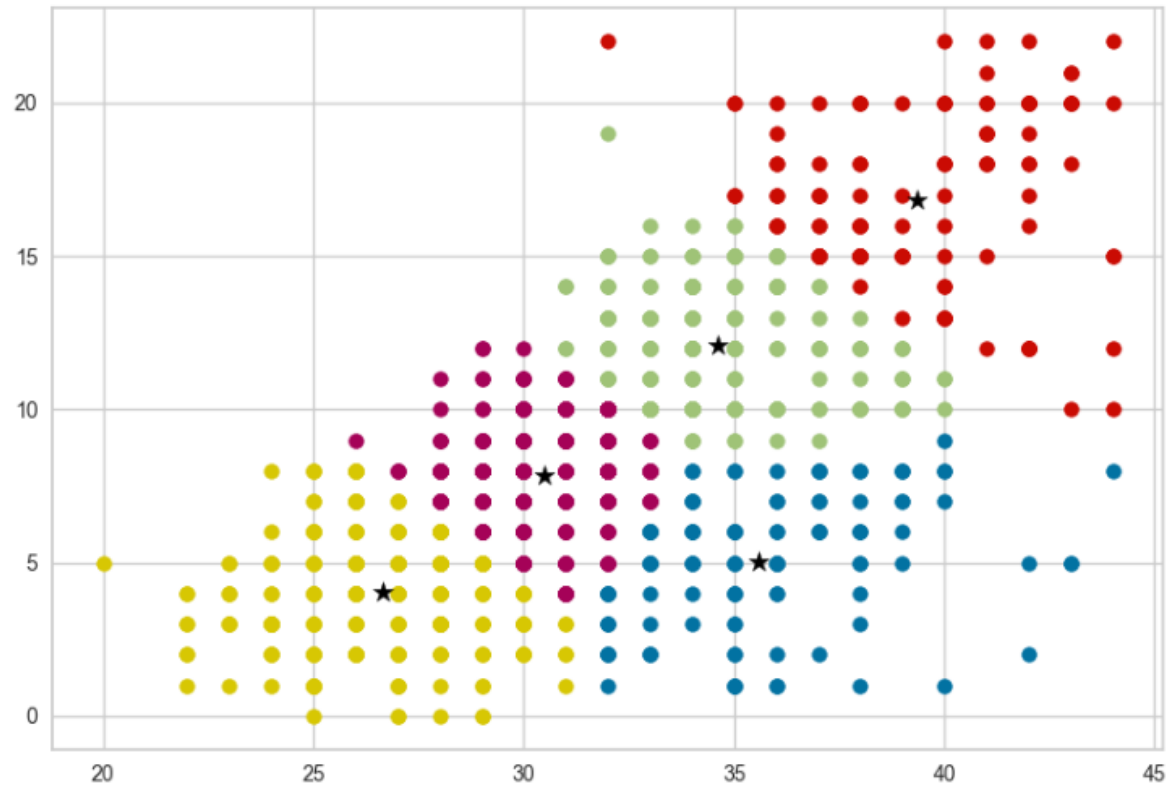
pav. 15 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=2



pav. 16 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=3



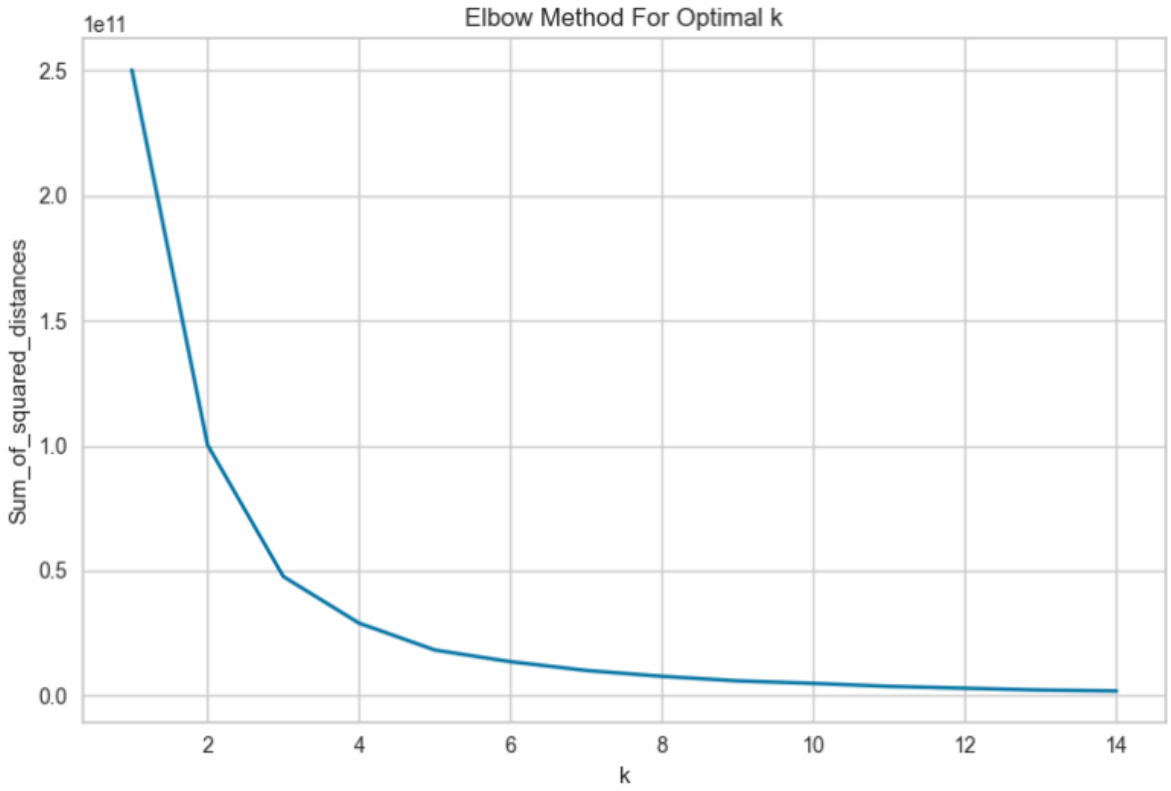
pav. 17 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=4



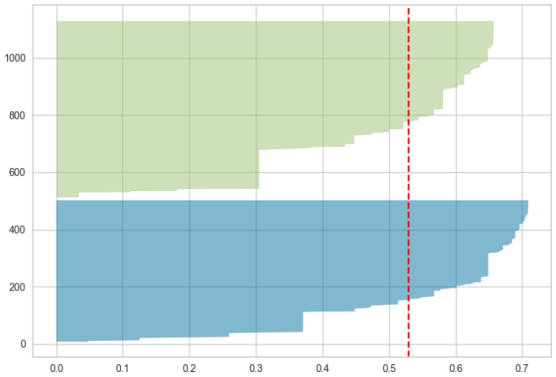
pav. 18 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=5

Antrasis bandymas buvo atliktas su „Total years of experience“ ir „Yearly brutto salary“

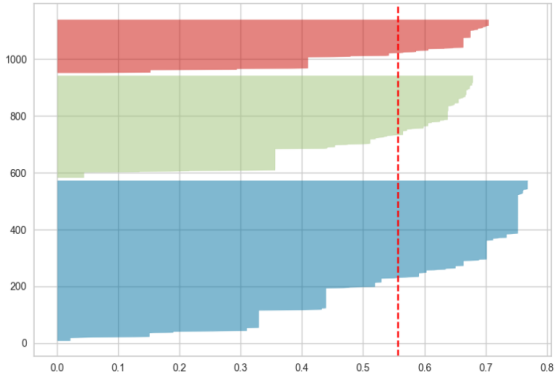
Silhouetter Score: 0.529– silueto koeficientas “Age” ir “Total years of experience” atributams



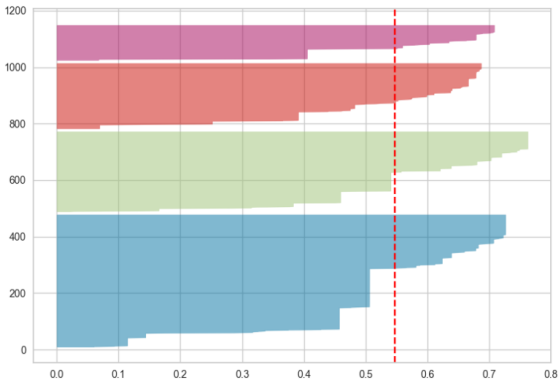
pav. 19 Atributų "Total years of experience" ir "Yearly brutto salary" inercijos ir klasterių priklausomybės kreivė



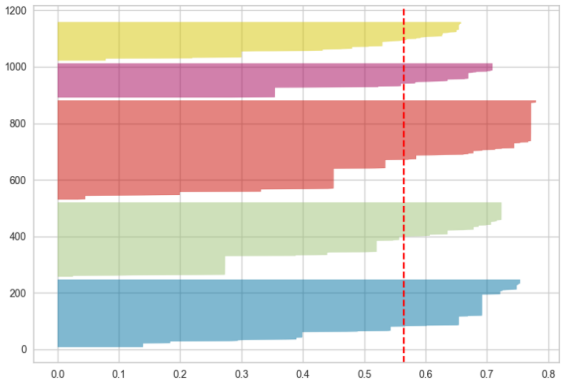
a) Silueto analizė kai k=2



b) Silueto analizė kai k=3

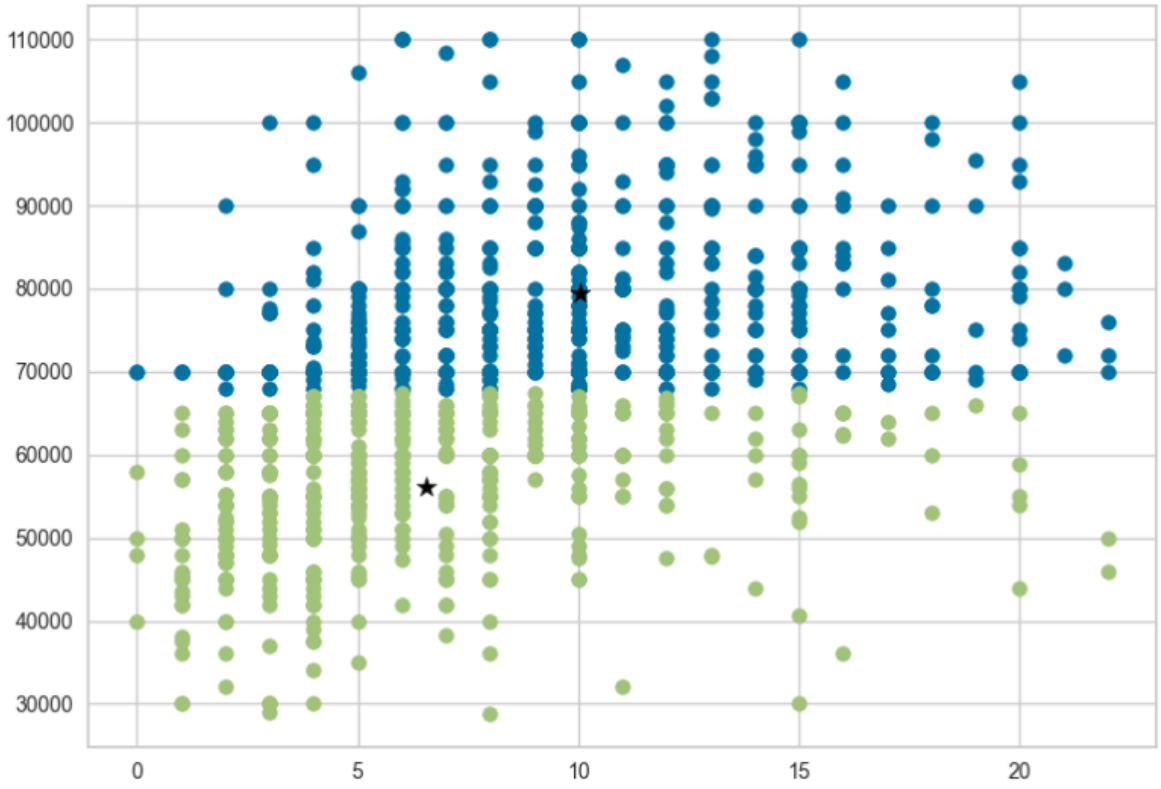


c) Silueto analizė kai k=4

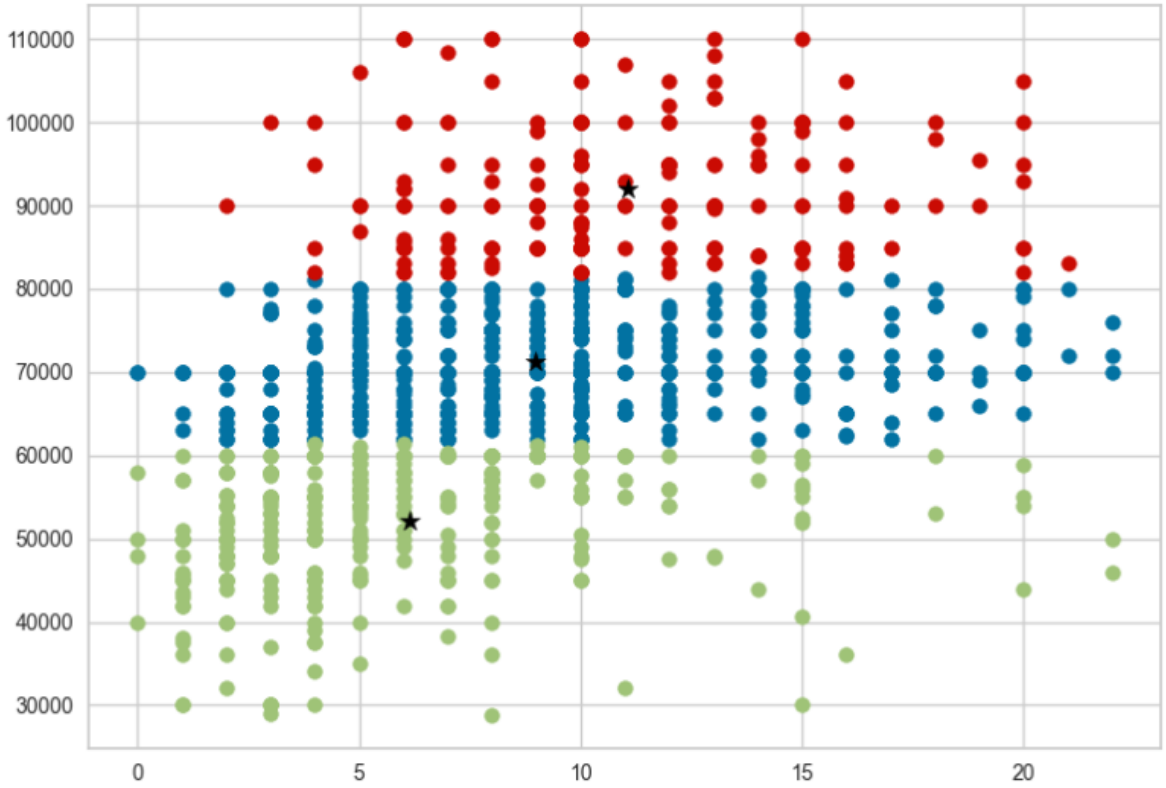


d) Silueto analizė kai k=5

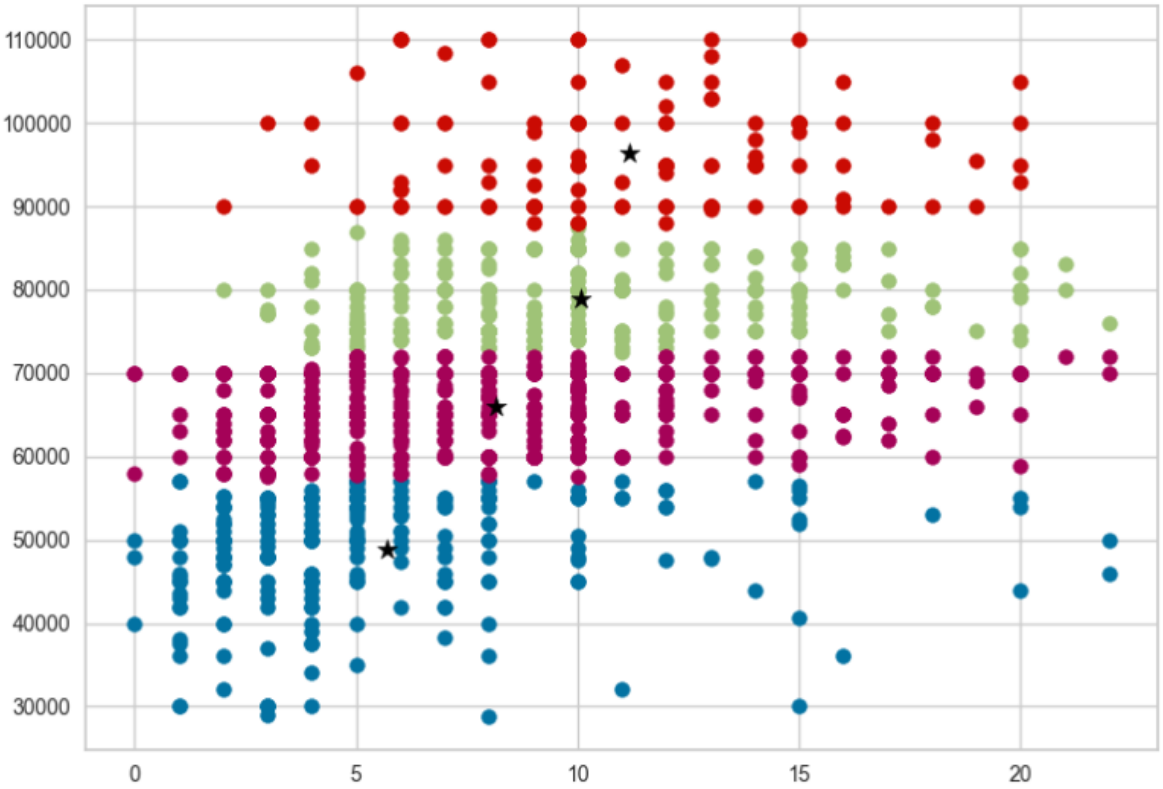
pav. 20 Siluetų analizė "Total years of experience" ir „Yearly brutto salary“ atributams



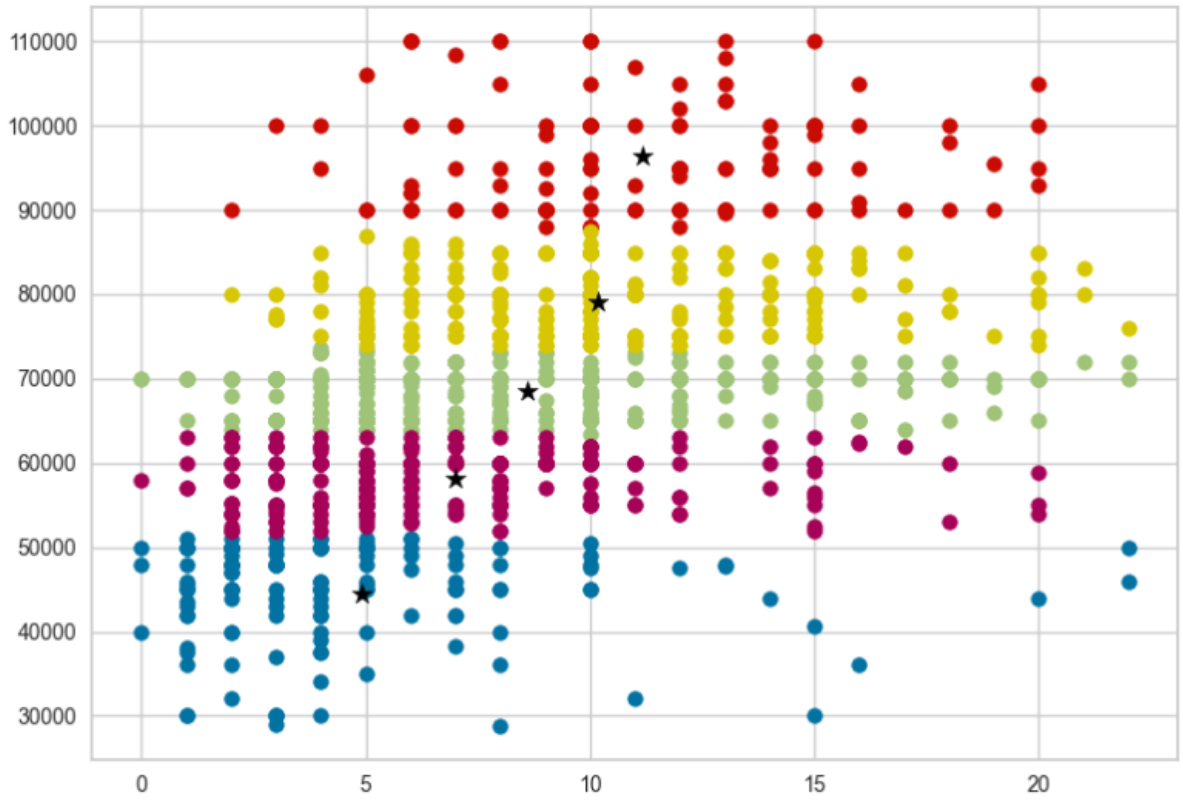
pav. 21 Atributų "Total years of experience" ir "Yearly brutto salary" "Scatter plot" diagrama kai k=2



pav. 22 Atributų "Total years of experience" ir "Yearly brutto salary" "Scatter plot" diagrama kai k=3



pav. 23 Atributų "Total years of experience" ir "Yearly brutto salary" "Scatter plot" diagrama kai k=4



pav. 24 Atributų "Total years of experience" ir "Yearly brutto salary" "Scatter plot" diagrama kai k=5

7. Išvados

Atlikus SOM realizaciją ir išnagrinėjus grafiškai pateiktus rezultatus, kuriuose atitinkama spalva atvaizduojami svorių pasiskirstymai suformuotame neuronų žemėlapyje, padarytos sekančios išvados:

1. Gautuose SOM žemėlapių grafikuose, matomi sudaryti klasteriai, tačiau dėl pasirinkto duomenų rinkinio, ne visuose grafikuose matoma aiški klasterių atskirtis.
2. Euklido atstumo metodas tinkamas mažesniems duomenų rinkiniams, didesniuose duomenų rinkiniuose yra naudojami kitokie atstumo apskaičiavimo metodai.
3. Eksperimentų atlikimo eigoje, pastebėta, jog geresni klasterių suformavimo rezultatai gaunami naudojant 250 epochų. Panaudojus 100 epochų aptikta neapsimokymo situacija, t. y. per 100 epochų nebūdavo suformuojami tokie aiškūs klasteriai.

Atlikus K-vidurkių realizacija ir išnagrinėjus grafiškai gautus rezultatus galima teigti, jog K-vidurkių metodas pagal tam tikras duomenų savybes suskirsto juos į atskirus klasterius pvz., atlikus pirmąjį bandymą sunku nusakyti pagal kokias savybes duomenys yra suklasterizuoti, nes klasterių pasiskirstymas yra įvairus (pav. 18 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=5). Antrajame bandyme matome jog klasterių pasiskirstymas yra linijinis iš to galima spręsti, kad klasteriai sudaryti pagal metinę algą pvz., pasirenkame 2 klasterius, tai pirmasis klasteris apims duomenų dalį kur metinė alga yra žemiau vidutinės metinės algos, antrasis – daugiau. Optimaliam klasterių skaičiui pasirinkti buvo nubraižyti inercijos ir siluetų grafikai. Iš inercijos grafikų matome alkūnių taškus, kurie nurodo optimalų klasterių skaičių, nes po to inercijos vertės pokytis nėra reikšmingas. Iš silueto analizės matome jog šiuo atveju galime pasirinkti nuo 2 iki 5 klasteriu, nes nėra neigiamų reikšmių ir koeficientai yra panašūs, tačiau pirmuoju atveju optimaliausias klasterių skaičius yra 2, o antru atveju – 5, nes koeficientai yra arčiausiai 1.

8. Programos kodas

```
import copy

import matplotlib.pyplot as plt
import pandas as pd
from minisom import MiniSom
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from yellowbrick.cluster import SilhouetteVisualizer



def convert_categorical_to_continuous_data(df):
    for column_name in df:
        if type(df[column_name][0]) == str:
            df[column_name] = pd.Categorical(df[column_name])
            df[column_name] = df[column_name].cat.codes
    return df

def read_data(filename:str):
    init_data = pd.read_excel(filename)
    df = copy.deepcopy(init_data)
    df = convert_categorical_to_continuous_data(df=df)
    y = df.pop('Company size').to_numpy().astype(int)
    X = df.to_numpy()
    return X, y, init_data


def plot_minisom_results(som, training_data, labels, init_data):
    plt.bone()
    plt.pcolor(som.distance_map().T)
    plt.colorbar()
    # markers = ['o', 's', 'D', 'p', 'w']
    # colors = ['r', 'g', 'b', 'y', 'o']
    for i, x in enumerate(training_data):
        w = som.winner(x)
        plt.plot(w[0] + 0.5,
                 w[1] + 0.5,
                 # markers[labels[i]],
                 # markeredgecolor=colors[labels[i]],
                 # markerfacecolor='None',
                 # markersize=10,
                 # markeredgewidth=2
                 )
    plt.show()

def execute_minisom(som_dimensions, training_data, labels, init_data):
    som = MiniSom(x=som_dimensions[0], y=som_dimensions[1],
input_len=training_data.shape[1], sigma=1.0, learning_rate=0.5)
    som.random_weights_init(training_data)
    som.train(data=training_data, num_iteration=1000)
    plot_minisom_results(som=som, training_data=training_data,
labels=labels, init_data=init_data)

def plot_KMeans_results(n_clusters, dataset, y_km, clusters):
    for i in range(0,n_clusters):
        plt.scatter(dataset[y_km == i, 0], dataset[y_km == i, 1], s=50)
        plt.scatter(clusters[i][0], clusters[i, 1], marker='*', s=100,
color='black')
    plt.show()
```

```
def plot_Elbow(dataset):
    Sum_of_squared_distances = []
    K = range(1, 15)
    for k in K:
        km = KMeans(n_clusters=k)
        km = km.fit(dataset)
        Sum_of_squared_distances.append(km.inertia_)
    plt.plot(K, Sum_of_squared_distances, 'bx-')
    plt.xlabel('k')
    plt.ylabel('Sum_of_squared_distances')
    plt.title('Elbow Method For Optimal k')
    plt.show()

def plot_Silhoutte(dataset):
    for i in [2, 3, 4, 5]:
        '''
        Create KMeans instance for different number of clusters
        '''
        km = KMeans(n_clusters=i, init='k-means++', n_init=10,
max_iter=100)
        '''
        Create SilhouetteVisualizer instance with KMeans instance
        Fit the visualizer
        '''
        visualizer = SilhouetteVisualizer(km, colors='yellowbrick')
        visualizer.fit(dataset)
        plt.show()

def execute_KMeans(dataset, n_clusters):
    km = KMeans(n_clusters=n_clusters, random_state=42)
    points = dataset[:, [4, 5]]
    km.fit(points)
    plt.scatter(dataset[:, 4], dataset[:, 5])
    plt.show()

    clusters = km.cluster_centers_
    y_km = km.fit_predict(points)
    plot_KMeans_results(n_clusters,points,y_km,clusters)
    plot_Elbow(points)
    score = silhouette_score(points, km.labels_, metric='euclidean')
    print('Silhouetter Score: %.3f' % score)
    plot_Silhoutte(points)

if __name__ == '__main__':
    """
    Reading data from excel file
    """
    X, y, init_data = read_data(filename=r'IT Salary Survey EU
2020.xls')
    som_dimensions = [10, 10]

    # execute_minisom(som_dimensions=som_dimensions, training_data=X,
labels=y, init_data=init_data)
    execute_KMeans(X,5)
```

Papildinys

SOM metodą realizavo Ignas Šakys

K-vidurkių metodą realizavo Dovydas Zamas