

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 3](#_Toc73299468)

[2. Naudotas duomenų rinkinys 4](#_Toc73299469)

[2.1. Tolydinio tipo kintamieji 4](#_Toc73299470)

[2.2. Kategorinio tipo kintamieji 4](#_Toc73299471)

[3. SOM realizuojantys metodai 5](#_Toc73299472)

[4. K-vidurkių algoritmo realizuojantys metodai 5](#_Toc73299473)

[5. Atstumo metrikos 6](#_Toc73299474)

[6. Atlikti eksperimentai 7](#_Toc73299475)

[7. Išvados 19](#_Toc73299476)

[8. Programos kodas 20](#_Toc73299477)

[Papildinys 22](#_Toc73299478)

# Paveiksliukų sąrašas

[pav. 1 Tolydinio tipo kintamųjų analizės lentelė prieš duomenų koregavimą 4](#_Toc73299479)

[pav. 2 Tolydinio tipo kintamųjų analizės lentelė po duomenų koregavimo 4](#_Toc73299480)

[pav. 3 Kategorinio tipo kintamųjų analizės lentelė prieš duomenų koregavimą 4](#_Toc73299481)

[pav. 4 Kategorinio tipo kintamųjų analizės lentelė po duomenų koregavimo 4](#_Toc73299482)

[pav. 5 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 3 7](#_Toc73299483)

[pav. 6 Antrojo bandymo SOM žemėlapio grafikas, kai k = 3 7](#_Toc73299484)

[pav. 7 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 4 8](#_Toc73299485)

[pav. 8 Antrojo bandymo SOM žemėlapio grafikas, kai k = 4 8](#_Toc73299486)

[pav. 9 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 5 9](#_Toc73299487)

[pav. 10 Antrojo bandymo SOM žemėlapio grafikas, kai k = 5 9](#_Toc73299488)

[pav. 11 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 6 10](#_Toc73299489)

[pav. 12 Antrojo bandymo SOM žemėlapio grafikas, kai k = 6 10](#_Toc73299490)

[pav. 13 Atributų "Age" ir "Total years of experience" inercijos ir klasterių priklausomybės kreivė 11](#_Toc73299491)

[pav. 14 Siluetų analizė "Age" ir "Total years of experience" atributams 12](#_Toc73299492)

[pav. 15 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=2 13](#_Toc73299493)

[pav. 16 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=3 13](#_Toc73299494)

[pav. 17 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=4 14](#_Toc73299495)

[pav. 18 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=5 14](#_Toc73299496)

[pav. 19 Atributų "Total years of experience" ir "Yearly brutto salary" inercijos ir klasterių priklausomybės kreivė 15](#_Toc73299497)

[pav. 20 Siluetų analizė "Total years of experience" ir „Yearly brutto salary“ atributams 16](#_Toc73299498)

[pav. 21 Atributų "Total years of experience" ir "Yearly brutto salary" "Scatter plot" diagrama kai k=2 16](#_Toc73299499)

[pav. 22 Atributų "Total years of experience" ir "Yearly brutto salary" "Scatter plot" diagrama kai k=3 17](#_Toc73299500)

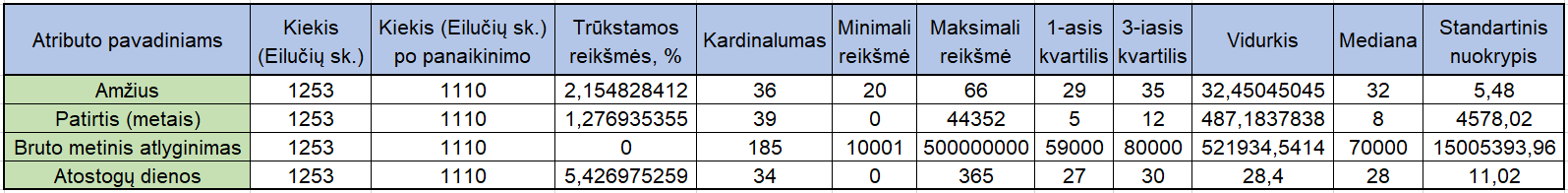
[pav. 23 Atributų "Total years of experience" ir "Yearly brutto salary" "Scatter plot" diagrama kai k=4 17](#_Toc73299501)

[pav. 24 Atributų "Total years of experience" ir "Yearly brutto salary" "Scatter plot" diagrama kai k=5 18](#_Toc73299502)

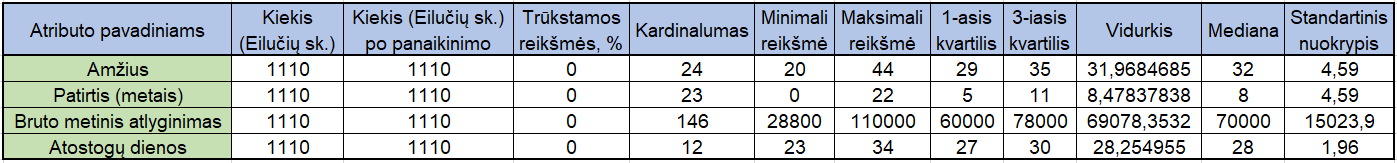
# Naudotas duomenų rinkinys

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.

# Tolydinio tipo kintamieji

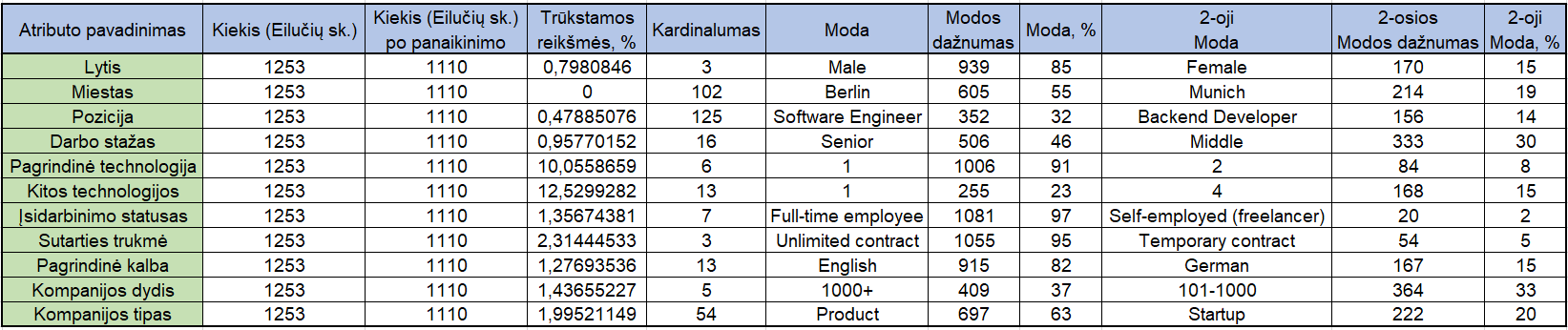


pav. 1 Tolydinio tipo kintamųjų analizės lentelė prieš duomenų koregavimą

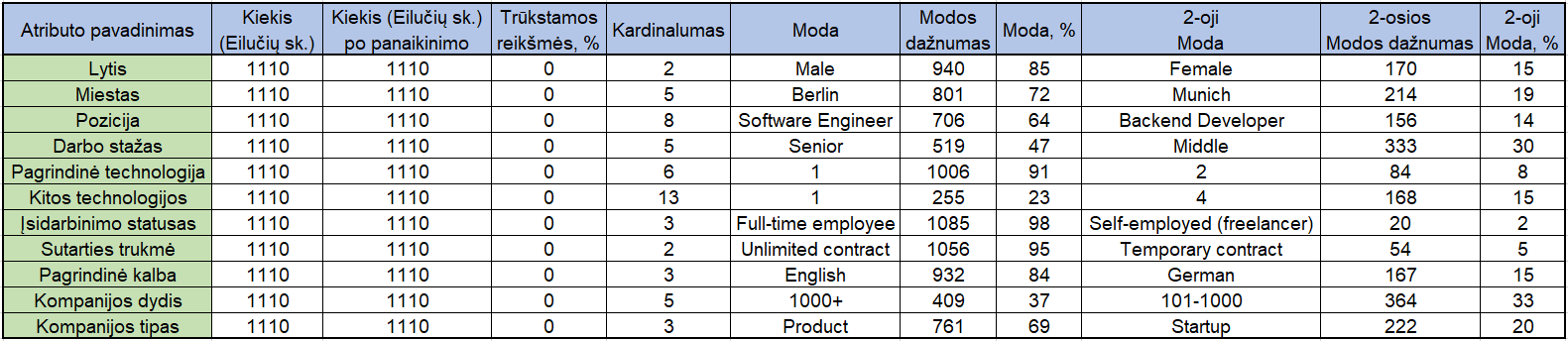


pav. 2 Tolydinio tipo kintamųjų analizės lentelė po duomenų koregavimo

# Kategorinio tipo kintamieji



pav. 3 Kategorinio tipo kintamųjų analizės lentelė prieš duomenų koregavimą



pav. 4 Kategorinio tipo kintamųjų analizės lentelė po duomenų koregavimo

# 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.

# 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

# Atstumo metrikos

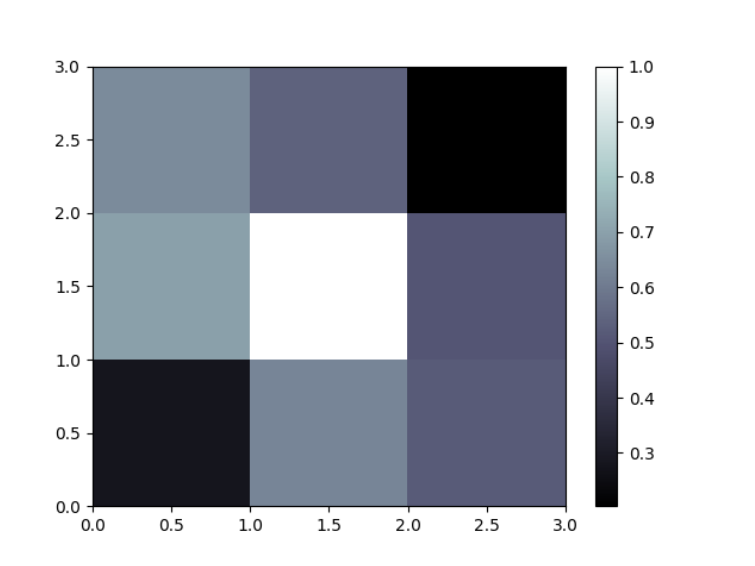
Šio darbo realizacijoje naudotas Euklido atstumas, kitaip tariant atstumo tarp parinkto neurono ir įvesties vektoriaus kvadratas. Nors šiuo konkrečiu atveju Euklido atstumo skaičiavimo metodas ir atlieka skaičiavimus korektiškai, tačiau didėjant duomenų kiekiui, kai kalbama apie milžiniškus duomenų rinkinius,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

# Atlikti eksperimentai

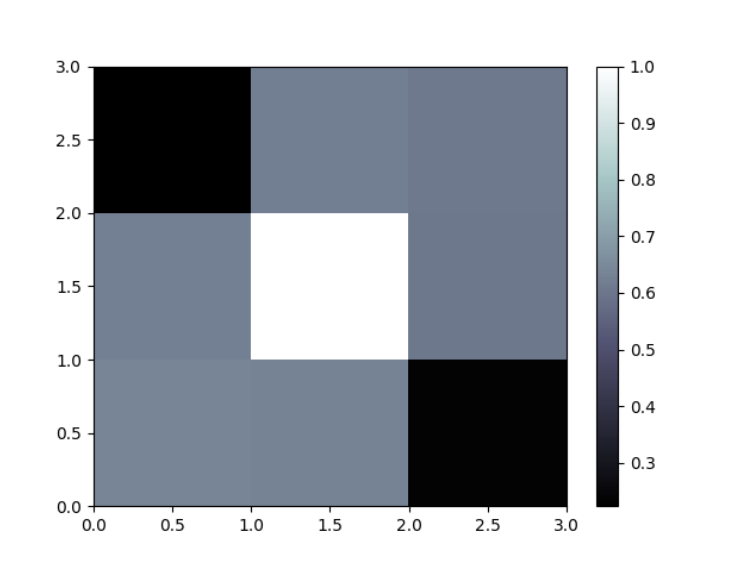
Prieš atliekant eksperimentus, buvo priimtos sekančių konstantų reikšmės, bei paaiškinimai:

* 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.

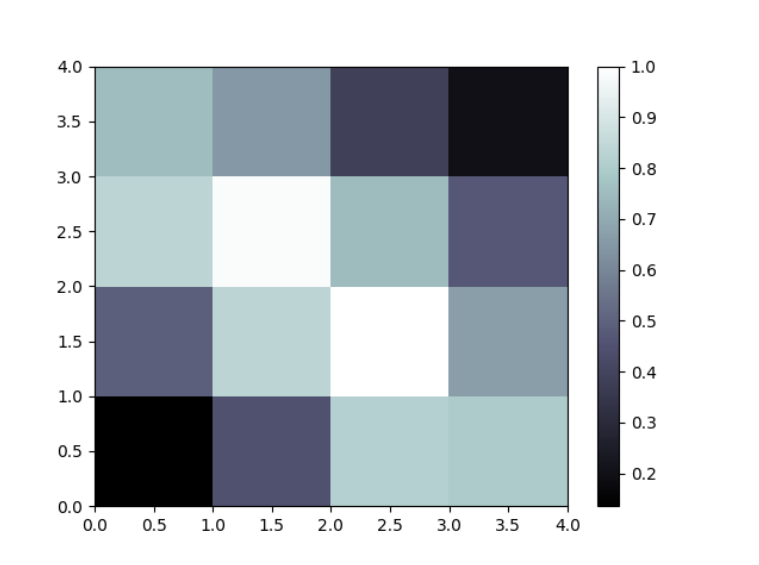


pav. 5 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 3

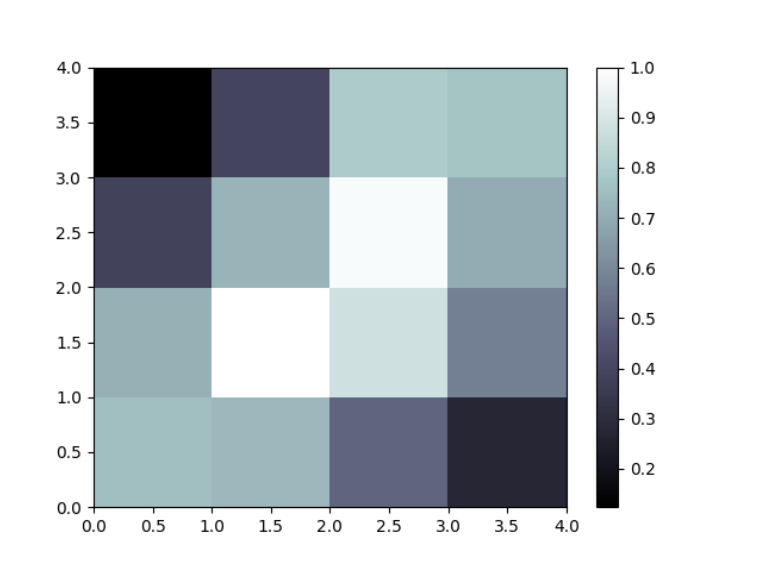


pav. 6 Antrojo bandymo SOM žemėlapio grafikas, kai k = 3

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

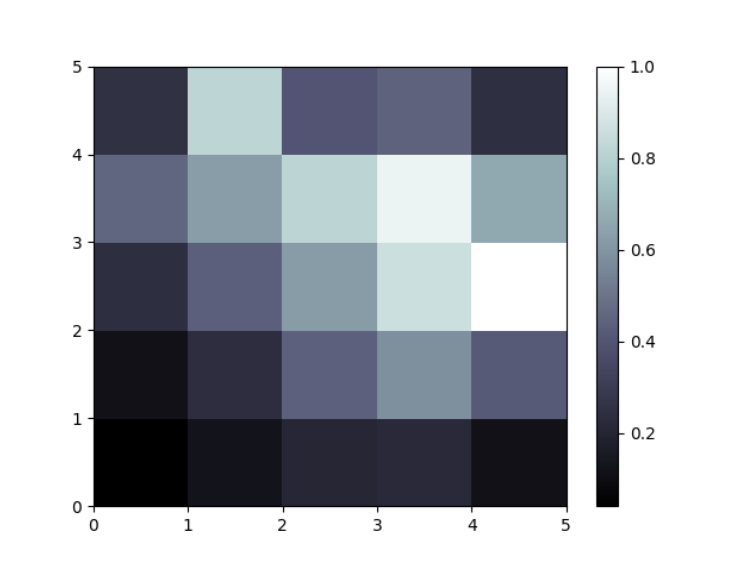


pav. 7 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 4

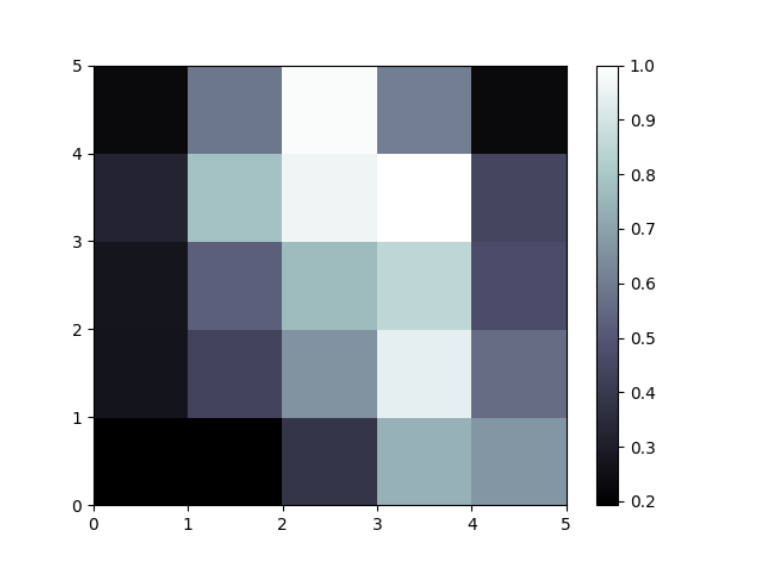


pav. 8 Antrojo bandymo SOM žemėlapio grafikas, kai k = 4

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

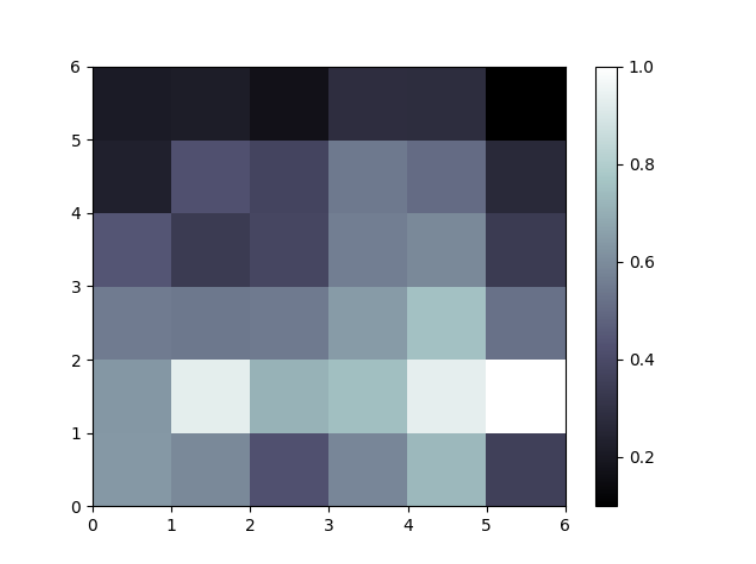


pav. 9 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 5

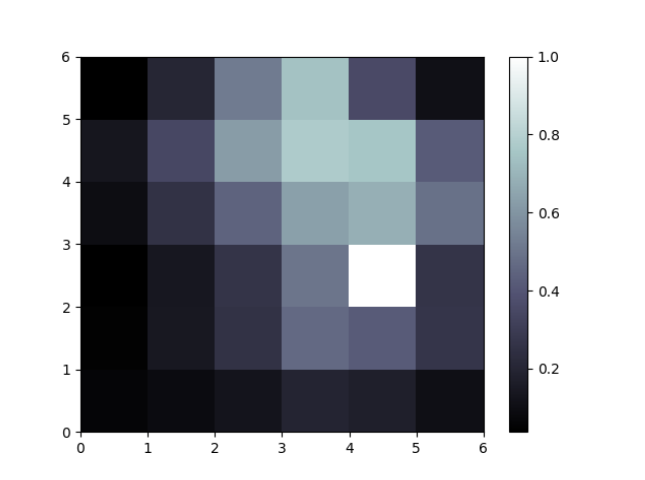


pav. 10 Antrojo bandymo SOM žemėlapio grafikas, kai k = 5

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



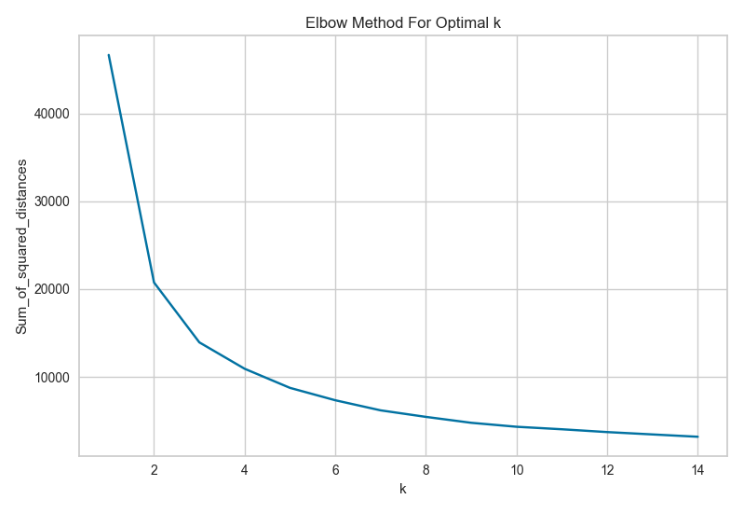
pav. 11 Pirmojo bandymo SOM žemėlapio grafikas, kai k = 6



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

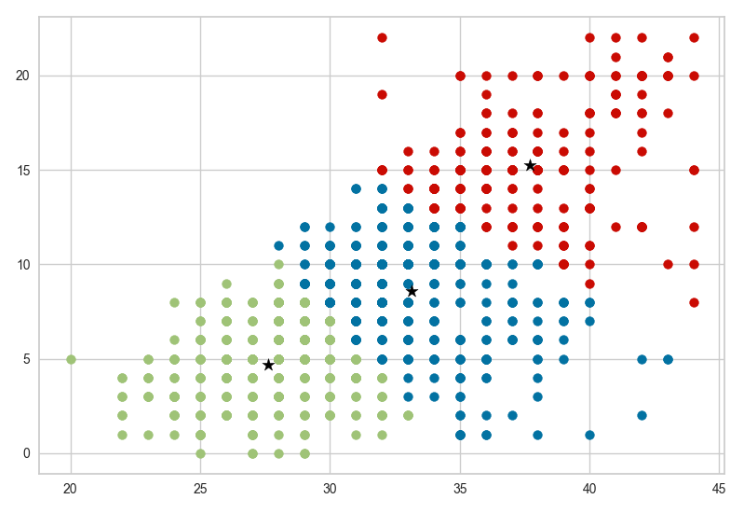
K-vidurkių algoritmui pirmąjąm ekspermentui buvo pasirinkti „Age“ ir „Total years of experience“ atributai.

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

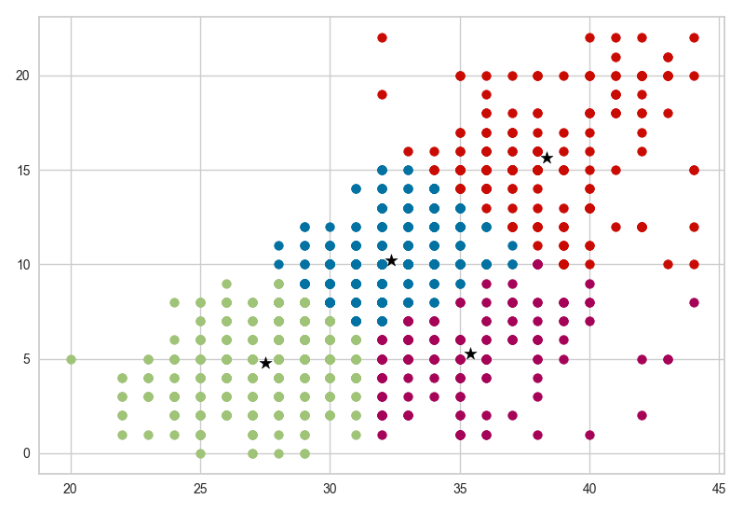


pav. 13 Atributų "Age" ir "Total years of experience" inercijos ir klasterių priklausomybės kreivė

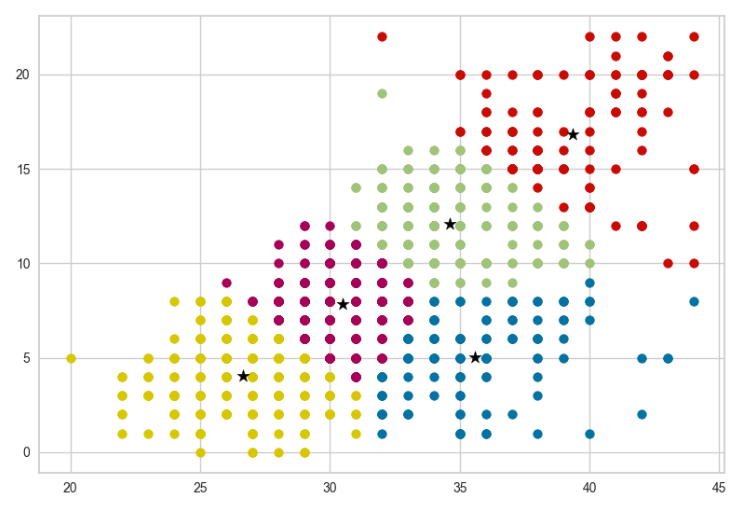
|  |  |
| --- | --- |
|  |  |
| 1. Silueto analizė kai k=2 | 1. Silueto analizė kai k=3 |
|  |  |
| 1. Silueto analizė kai k=4 | 1. Silueto analizė kai k=5 |
| pav. 14 Siluetų analizė "Age" ir "Total years of experience" atributams | |
| pav. 15 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=2 | |



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



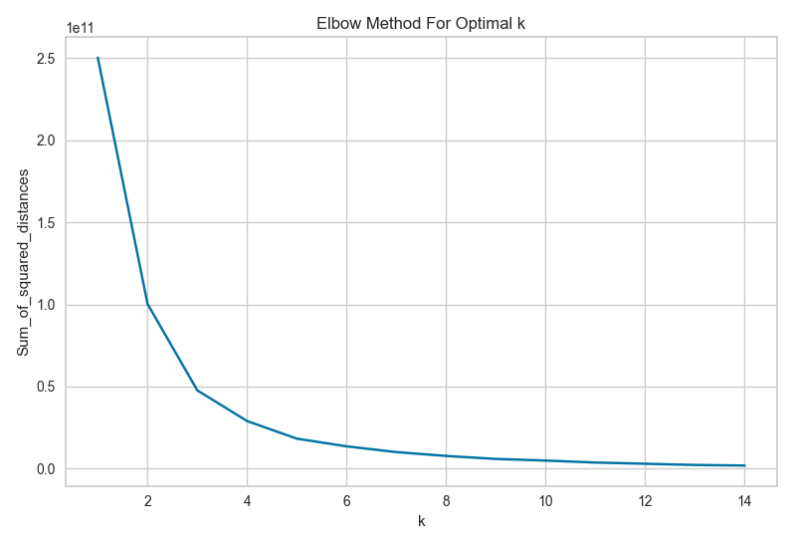
pav. 17 Atributų "Age"ir "Total years of experience" "Scatter plot" diagrama kai k=4



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

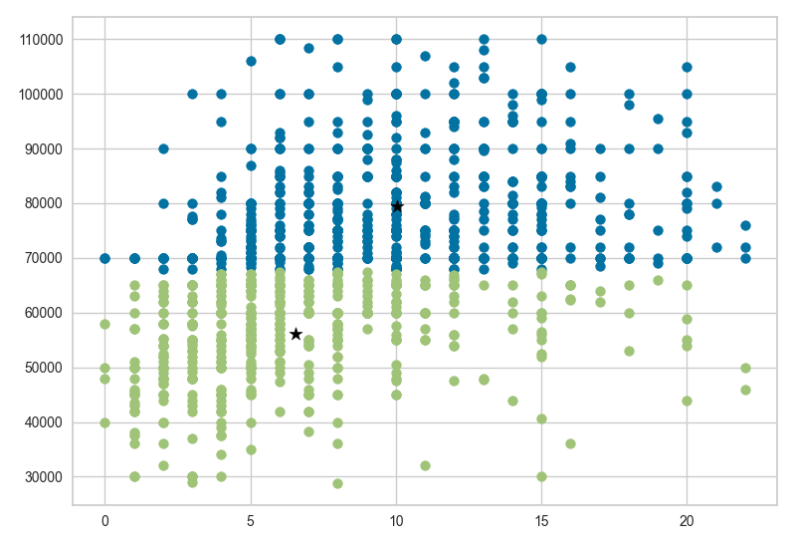
Antrasis bandymas buvo atliktas su „Total years of experience“ ir „Yearly brutto salary“

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

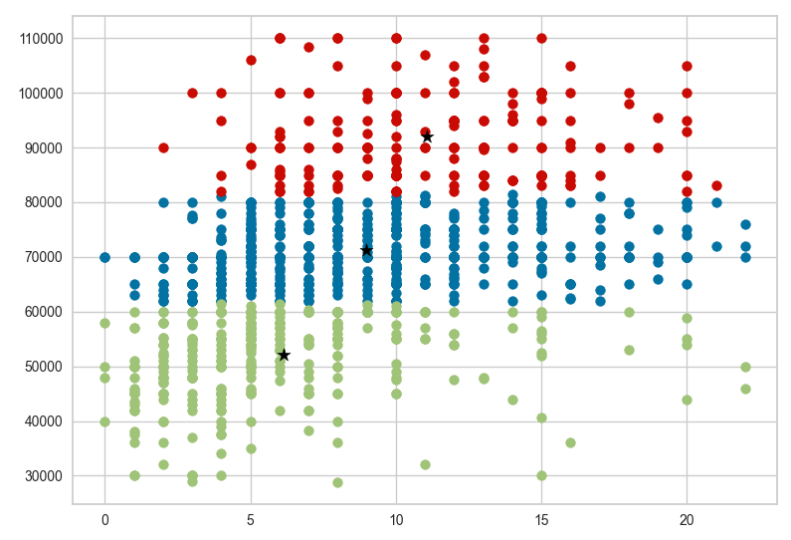


pav. 19 Atributų "Total years of experience" ir "Yearly brutto salary" inercijos ir klasterių priklausomybės kreivė

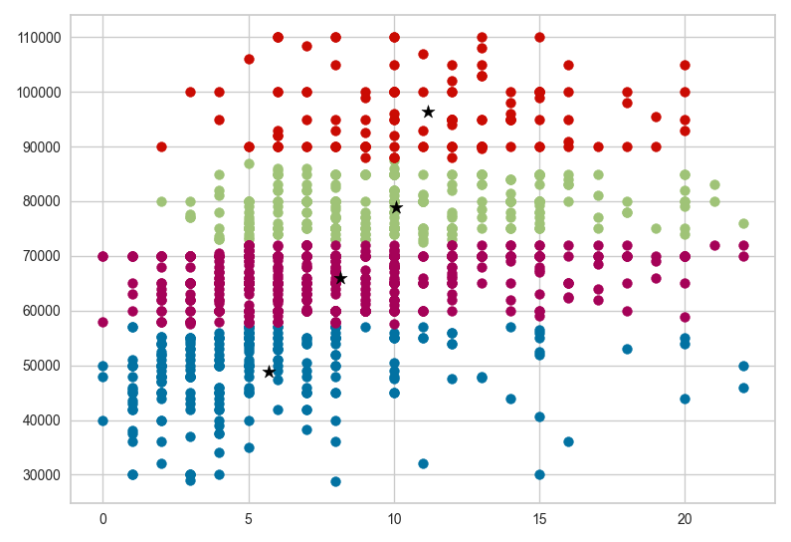
|  |  |
| --- | --- |
|  |  |
| 1. Silueto analizė kai k=2 | 1. Silueto analizė kai k=3 |
|  |  |
| 1. Silueto analizė kai k=4 | 1. Silueto analizė kai k=5 |
| pav. 20 Siluetų analizė "Total years of experience" ir „Yearly brutto salary“ atributams | |



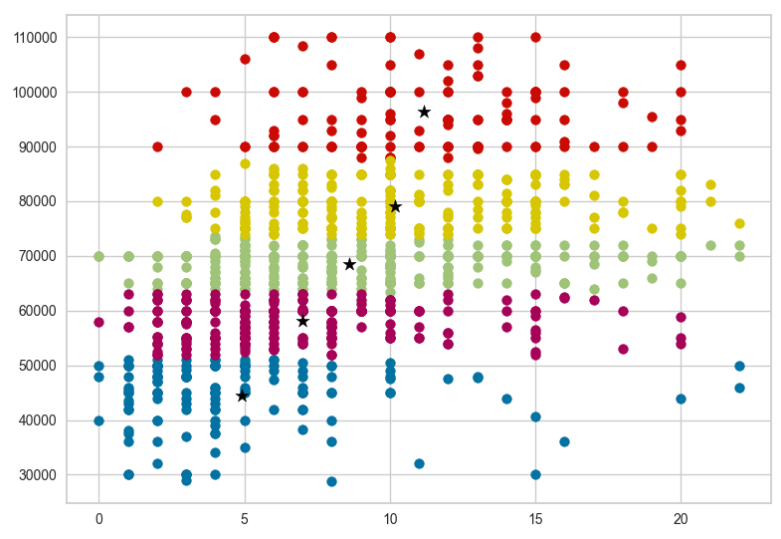
pav. 21 Atributų "Total years of experience" ir "Yearly brutto salary" "Scatter plot" diagrama kai k=2



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

# 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.

# Programos kodas

1. **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