VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Papildomi vizualizavimo skyriai

Praktinis darbas Nr.2

Darbą atliko:

Roland Gulbinovič ir Matas Amšiejus

Duomenų mokslas III kursas, 2 grupė

Vilnius 2022

Turinys

Įvadas

Duomenys

Darbui naudojama *Diamonds* duomenų aibė. Šiame duomenų rinkinyje yra beveik 54 000 įrašų apie deimantus. Duomenų atributai:

- *Carat* –masė (karatais (1 karatas = 200 mg)) (kiekybinis tolydus, m.s. - santykių);
- *Cut* – deimanto forma (priklausomas kategorinis kint. (matavimų skalė (toliau m.s.) nominalioji);
- *Color* – deimanto spalva (kategorinis kint., m.s. – nominalioji);
- *Clarity* – deimanto skaidrumas (kategorinis kint., m.s. – ranginė);
- *Depth* – deimanto aukštis padalintas iš jo juostos (%, kiekybinis tolydus kint., m.s. – santykių);
- *Table* – deimanto viršūnė išreikšta procentais per vidutinį deimanto diametrą (%, kiekybinis tolydus kint., m.s. – santykių);
- *Price* – deimanto kaina($) (kiekybinis tolydus, m.s. - santykių);
- *Length* – ilgis(mm) (kiekybinis tolydus, m.s. - santykių);
- *Width* – plotis(mm) (kiekybinis tolydus, m.s. - santykių);
- *Height* – aukštis(mm) (kiekybinis tolydus, m.s. - santykių).

Mūsų priklausomas požymis „Cut“ turi 5 galimas reikšmes: [Fair, Good, Very Good, Premium, Ideal].

Kadangi duomenų yra labai daug, mes išsirinkome tik po 200 įrašų kiekvienos deimanto formos (žr. kodą). Taigi darbui naudosime tik 1000 deimantų imtį.

Tyrimo tikslas

Šio darbo tikslas yra panaudoti dimensijos mažinimo metodus daugiamačių duomenų vizualizavimui, ištirti metodų galimybes bei pateikti pasirinktos aibės vizualizavimo rezultatus ir gautų rezultatų interpretaciją.

Tyrimo uždaviniai

- Rasti duomenų aibę;
- Pateikti pasirinktos aibės aprašomąją statistiką, aprašyti duomenų aibės specifiką;
- Sunormuoti duomenų aibę pagal vidurkį ir dispersiją;
- Sumažinti duomenų aibės dimensiją iki $dim = 2$, naudojant tris pasirinktus dimensijos mažinimo metodus;
- Palyginti rezultatus normuotos ir nenormuotos duomenų aibės;
- Vizualizuoti rezultatus naudodami taškinius grafikus;
- Apibendrinti gautus rezultatus;

- Įvardinti tirtų dimensijos mažinimo metodų privalumus ir trūkumus.

Duomenų apdorojimas

Duomenų aprašomoji statistika

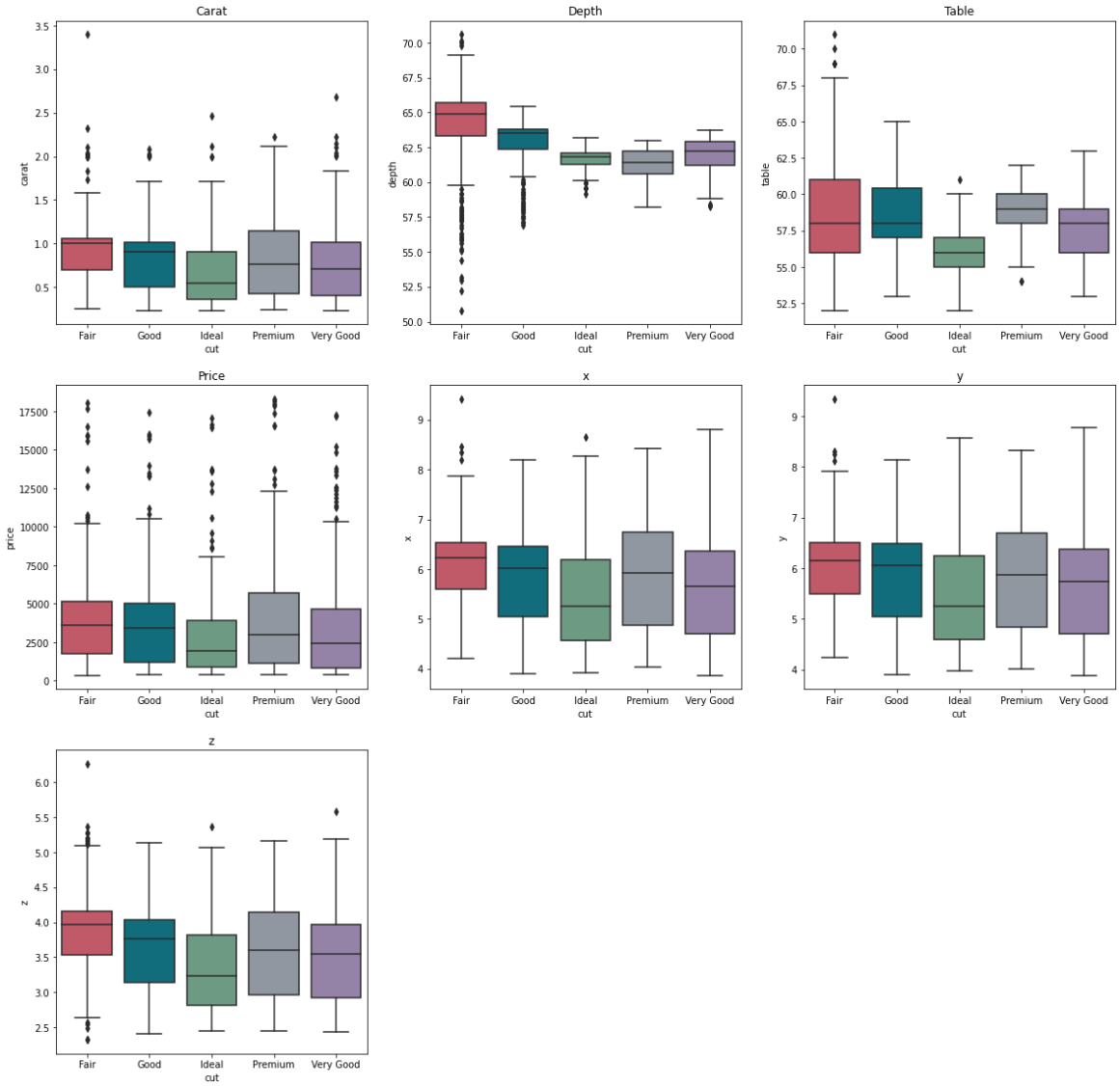
Iš pradžių galime pažiūrėti duomenų aprašomąją statistiką.

1 lentelė. Aprašomoji statistika

	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

Matome, kad *price* kovariantės vidurkis ir mediana žymiai skiriasi, o standartinis nuokrypis yra didelis. Tai gali indikuoti potencialias išskirtis. Su *carat* galime matyti, kad maksimali reikšmė yra žymiai didesnė už trečią kvartilį. Taip pat yra ir su *depth, table* reikšmėmis, kurių maksimumai ar minimumai žymiai skiriasi nuo vidurkio. Taigi, bus verta patikrinti, ar imtis turi išskirčių.

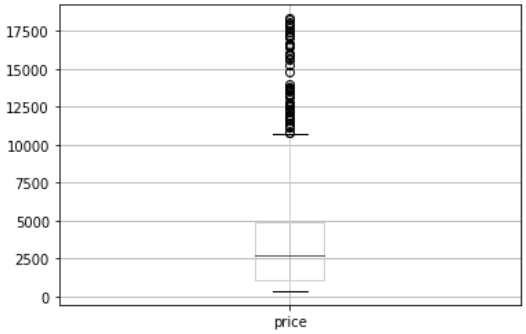
Dabar patikrinsime kaip duomenys pasiskirstę pagal formas. Tam naudosime stačiakampes diagramas. Iš pirmo paveikslėlio matome, kad duomenyse yra daug išskirčių. Taip pat galime pastebėti, kad nėra daug kovariančių, pagal kurias deimantų klasės smarkiai skirtųsi (labiausiai skiriasi pagal *depth, table*). Dar galime matyti, kad klasės *Good* ir *Very Good* yra labai panašios, todėl jas sujungsime.



1 pav. Cut stačiakampės diagramos

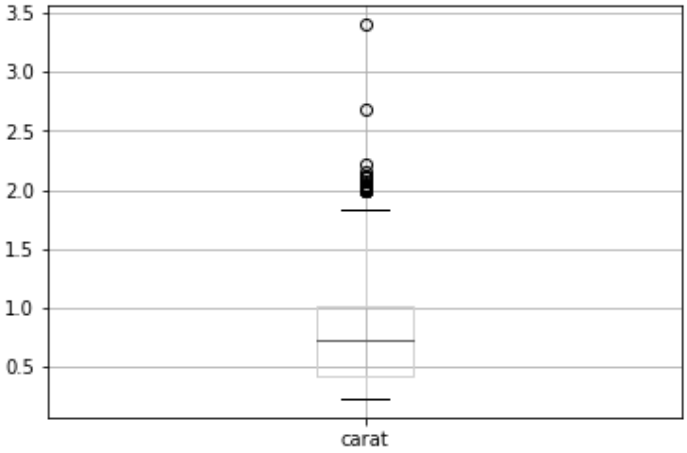
Išskirčių šalinimas

Susidarėme nuomonę, kad imtyje gali būti išskirčių. Pirma patikrinkime *price* kovariantę, nes ji atrodė pati nepastoviausia.



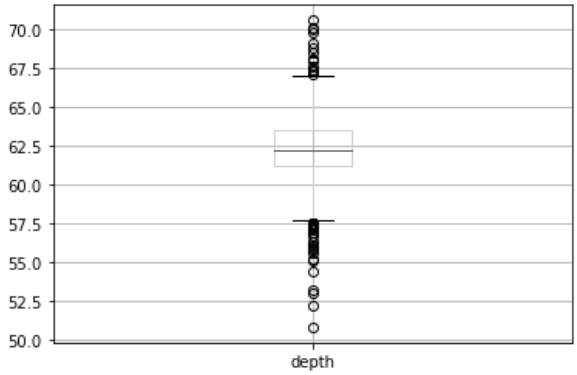
2 pav. Price stačiakampė diagrama

Matome, kad yra daug sąlyginių išskirčių. Patikrinus randame, kad yra 6 išskirtys. Jas šaliname. Toliau tikriname *carat* išskirtis.



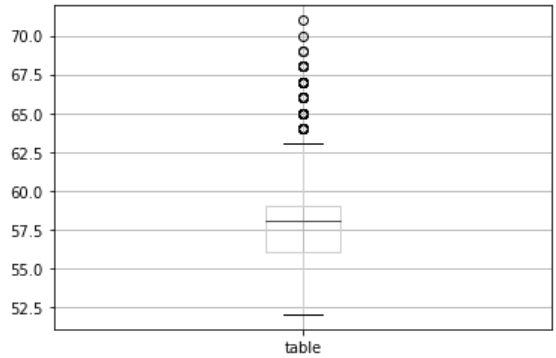
3 pav. Carat stačiakampė diagrama

Aiškiai matome vieną išskirtį, ją ir šaliname. Toliau tikriname *depth*.



4 pav. Depth stačiakampė diagrama

Šį kartą turime išskirtis kurios mažesnės nei $Q1 - 3 \cdot IQR$, jas šaliname. Toliau tikriname *table* kovariantę.



5 pav. Table stačiakampė diagrama

Matome daugiau taškų atsiskyrėlių. Patikrinus, iš jų vienuolika buvo išskirtys. Jas šaliname. Daugiau išskirčių nebeliko.

Koreliacijos

Nubrėžę koreliacijų matricą pamatėme, kad *carat, x, y, z* stipriai koreliuoja. Ištriname *y* ir *z*.

2 lentelė. Koreliacijų matrica

	carat	depth	table	price	x	y	z
carat	1.000000	0.196192	0.127079	0.900640	0.974481	0.973111	0.975622
depth	0.196192	1.000000	-0.318614	0.102759	0.108694	0.096885	0.297459
table	0.127079	-0.318614	1.000000	0.087506	0.160665	0.156849	0.089089
price	0.900640	0.102759	0.087506	1.000000	0.870403	0.874377	0.858085
x	0.974481	0.108694	0.160665	0.870403	1.000000	0.998235	0.980504
y	0.973111	0.096885	0.156849	0.874377	0.998235	1.000000	0.978037
z	0.975622	0.297459	0.089089	0.858085	0.980504	0.978037	1.000000

Normavimas

Visuose dimensijos mažinimo metoduose naudojome normavimą pagal vidurkį ir dispersiją (žr. kodą):

$$x_{norm} = \frac{x - \bar{x}}{\sqrt{\sigma^2}},$$

Kur σ^2 – požymio reikšmių dispersija, $\bar{x}$ – požymio reikšmių vidurkis.

3 lentelė. Nenormuoti duomenys

carat	cut	depth	table	price	x
0.30	Fair	61.7	66.0	593	4.25
0.50	Fair	65.1	59.0	1175	5.02
1.09	Fair	64.4	60.0	5085	6.49
0.50	Fair	66.9	57.0	1431	4.95
1.73	Fair	65.9	58.0	6007	7.52
...	...	...	...	...	...
1.59	Very Good	62.4	56.0	9925	7.46
0.30	Very Good	62.1	62.0	575	4.21
0.73	Very Good	63.2	58.0	2679	5.71
1.01	Very Good	62.8	59.0	6499	6.34
0.40	Very Good	60.0	56.0	853	4.81

4 lentelė. Sunormuoti duomenys

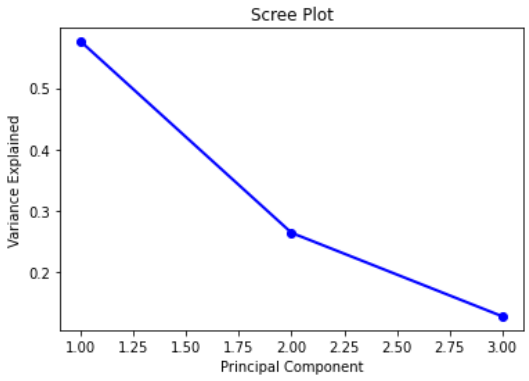
carat	depth	table	price	x
-1.185781	-0.277132	3.174828	-0.954069	-1.480072
-0.713494	1.297340	0.425901	-0.765335	-0.719690
0.679752	0.973184	0.818605	0.502621	0.731948
-0.713494	2.130883	-0.359506	-0.682318	-0.788816
2.191071	1.667804	0.033198	0.801612	1.749082
...	...	...	...	...
1.860470	0.047024	-0.752210	2.072162	1.689831
-1.185781	-0.091900	1.604013	-0.959906	-1.519573
-0.170364	0.417488	0.033198	-0.277610	-0.038309
0.490838	0.232256	0.425901	0.961160	0.583821
-0.949638	-1.064368	-0.752210	-0.869755	-0.927067

Principinių komponenčių analizė (PCA)

Pirmas dimensijos mažinimo metodas kurį naudosime yra *principinių komponenčių analizė*.

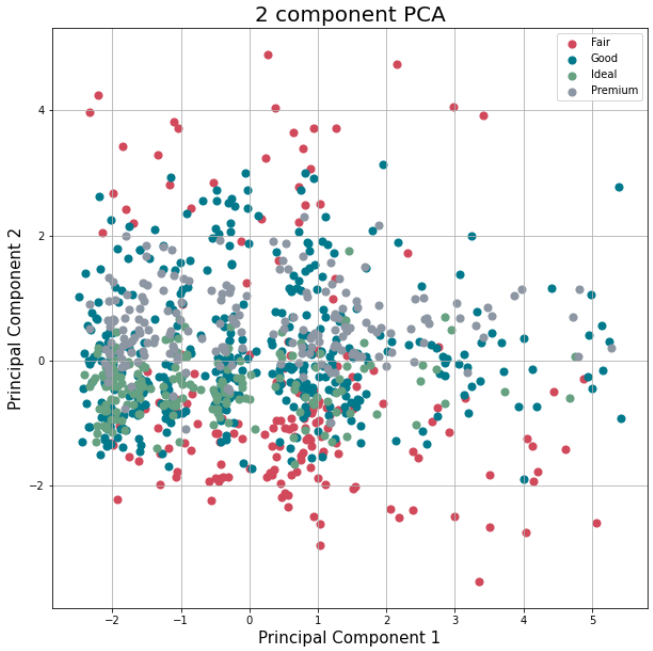
PCA tikslas – sumažinti sudėtingų duomenų dimensiją į lengvai suprantamą, prarandant kuo mažiau informacijos. Tai pasiekti galima transformuojant duomenis pagal principines komponentes (toliau PC), kurios yra nekoreliuotos ir pirmosios išlaiko didžiausią dispersiją (žr. literatūrą).

Pritaikius PCA metodą mūsų duomenims gauname, kad pirmos dvi komponentės išlaiko apie 80% bendros dispersijos (pirma komponentė – 57,5%, antra komponentė – 26%, trečia – 13%). Taigi matome, kad šios dvi komponentės išlaiko didžiąją dalį informacijos, tačiau trečio irgi gali parodyti papildomos informacijos.



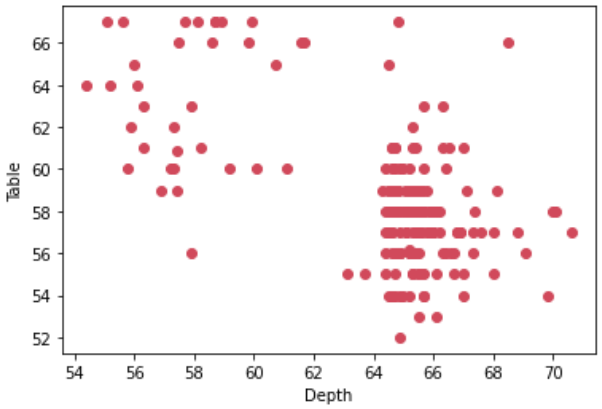
6 pav. Dispersijos pasiskirstymas

Dabar galime šias principines komponentes atvaizduoti naudojant taškinį grafiką.



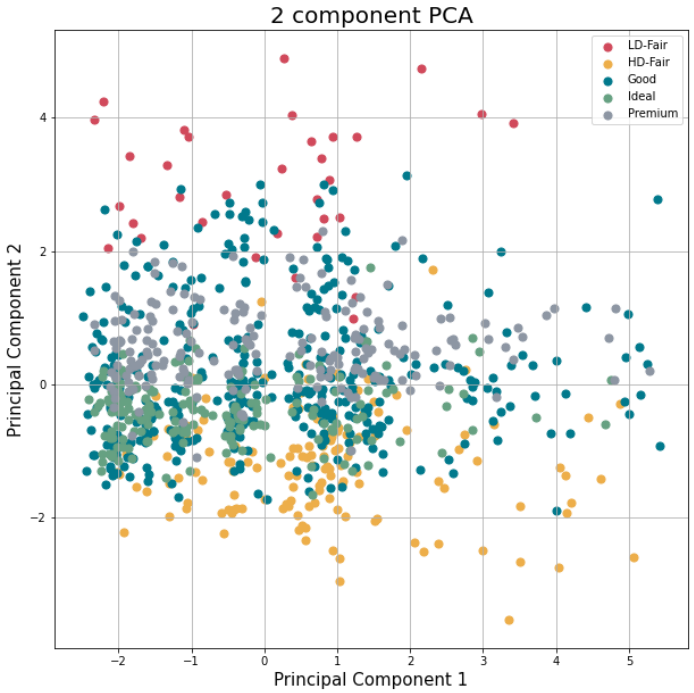
7 pav. Taškinė diagrama pagal principines komponentes

Iš grafiko matome, kad visos deimantų grupės atsiskiria sunkiai. Nors galime įžvelgti įdomų pastebėjimą. Grupė *Fair* išsibarsčiusi plačiau, bet daugiausiai – aplink *good* ir *ideal* grupes (pagal 2-ą PC). Galime teigti, kad pastaroji grupė potencialiai apjungia du klasterius. Pabandykime nustatyti, pagal ką jie atsiranda. Pabandykime šiai klasei nusibrėžti sklaidos diagramą pagal *depth* (matėme didžiausias išskirtis, potencialiai dėl susigrupavimo) ir *table*.



8 pav. Fair sklaidos diagrama pagal depth ir table

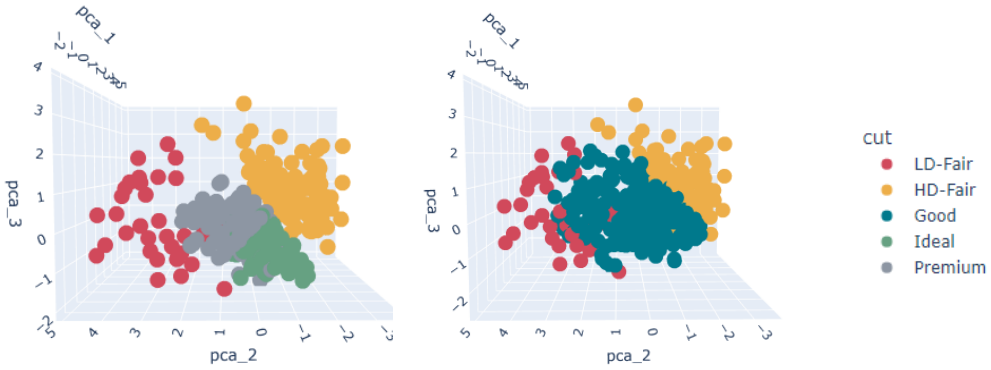
Matome, kad klasė tikrai susiskirsto į du klasterius, todėl pabandysime išvesti naujas klases pagal *depth*. Jei *depth* < 63, tai klasę priskirsime *LD-Fair* (nuo *low depth*). Kitu atveju – *HD-Fair*. Taigi dabar turime 5 klases.



9 pav. PCA su išskaidyta Fair

Dabar matome, kad abi *fair* grupės atsiskiria kaip tikėtasi.

Naudojant tris dimensijas, matome aiškesnius atsiskyrimus ir tarp kitų grupių.

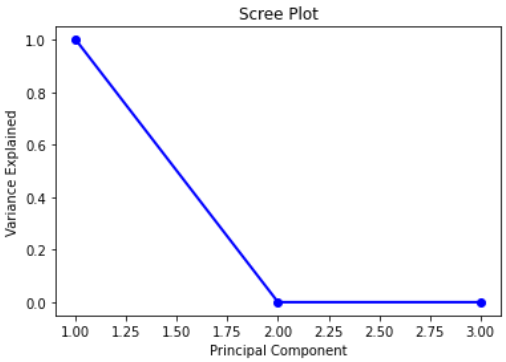


10 pav. 3D PCA

Nors *good* persidengia su *ideal* ir *premium*, tačiau atsiskiria nuo *fair*. Taip pat *ideal* ir *premium* irgi atsiskiria viena nuo kitos.

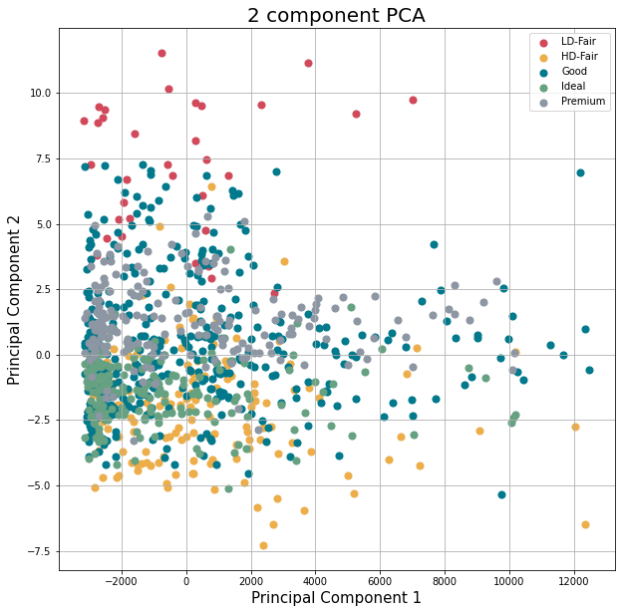
Nenormuoti duomenys

Pabandžius tą patį padaryti su nenormuotais duomenimis, matome, kad ant pirmos PC yra beveik visa dalis bendros dispersijos.



11 pav. Nenormuotų duomenų dispersijos pasiskirstymas

Iš 12 paveikslėlio matome, kad duomenys yra labiau susitelkę kairėje pusėje, taškai sunkiau atskiriami.

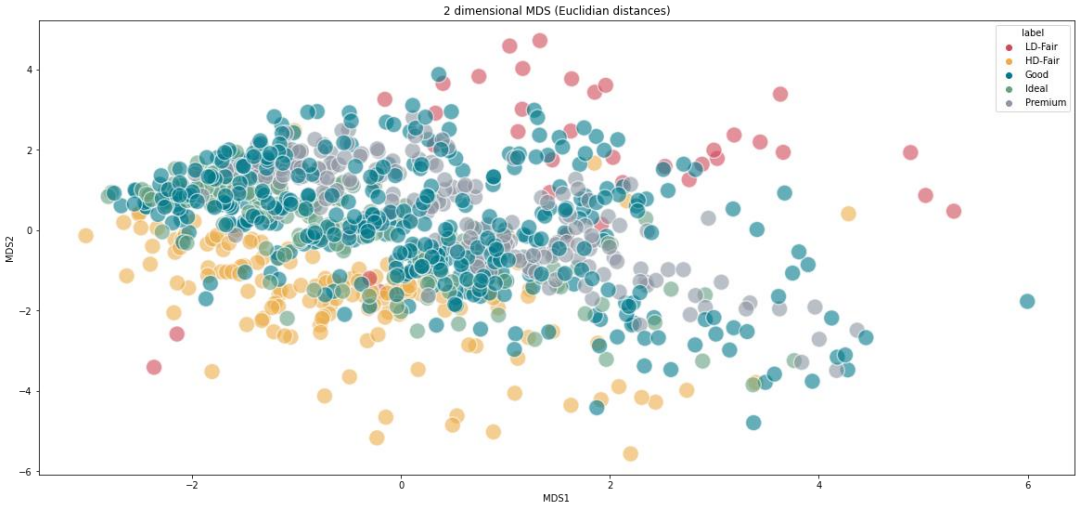


12 pav. Nenormuotų duomenų PCA

Daugiamatės skalės (MDS)

Toliau, kad sumažinti duomenų dimensiją naudojome *daugiamatės skalės* metodą.

MDS tikslas - kiekvieną objektą iš didesnės dimensijos duomenų erdvės transformuoti į iš anksto parinkto dydžio mažesnės dimensijos erdvę (vadinama vaizdo erdve). Naudojant MDS ieškoma daugiamačių duomenų projekcijų vaizdo erdvėje, siekiant išlaikyti atstumus tarp objektų (žr. literatūra).



13 pav. Vizualizacija pagal MDS

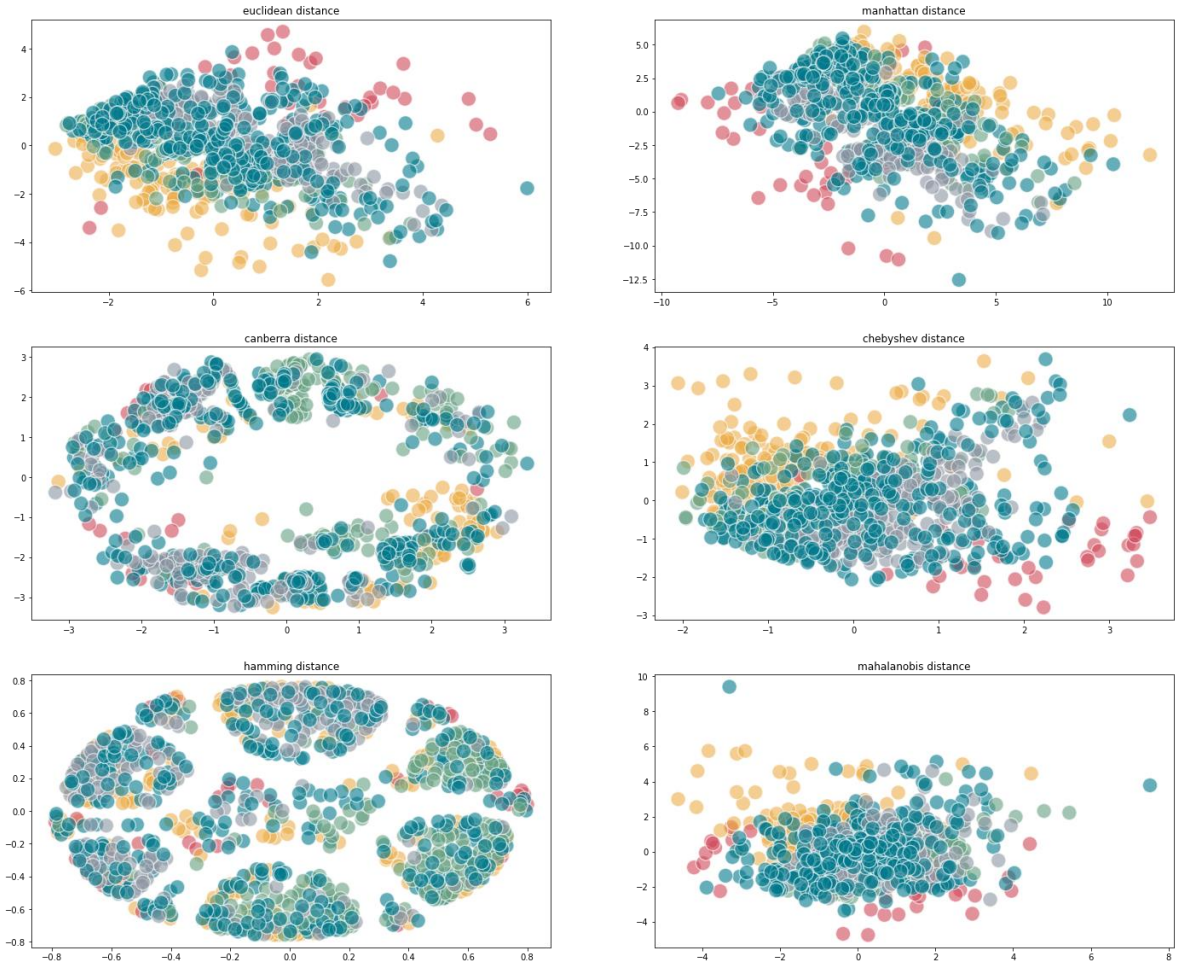
Matome, kad kaip ir su PCA, grupės *good*, *ideal* ir *premium* persidengia, tačiau *fair* ir vėl yra „iš kraštų“.

Iš įtempimo grafiko matome, kad perėjus prie dviejų komponenčių įtempimas smarkiai krenta. Paėmus tris komponentes, įtempimas nuo dviejų pasikeis nežymiai.

14 pav. Įtempimo funkcija

Skirtingi grafikai keičiant algoritmo parametrus

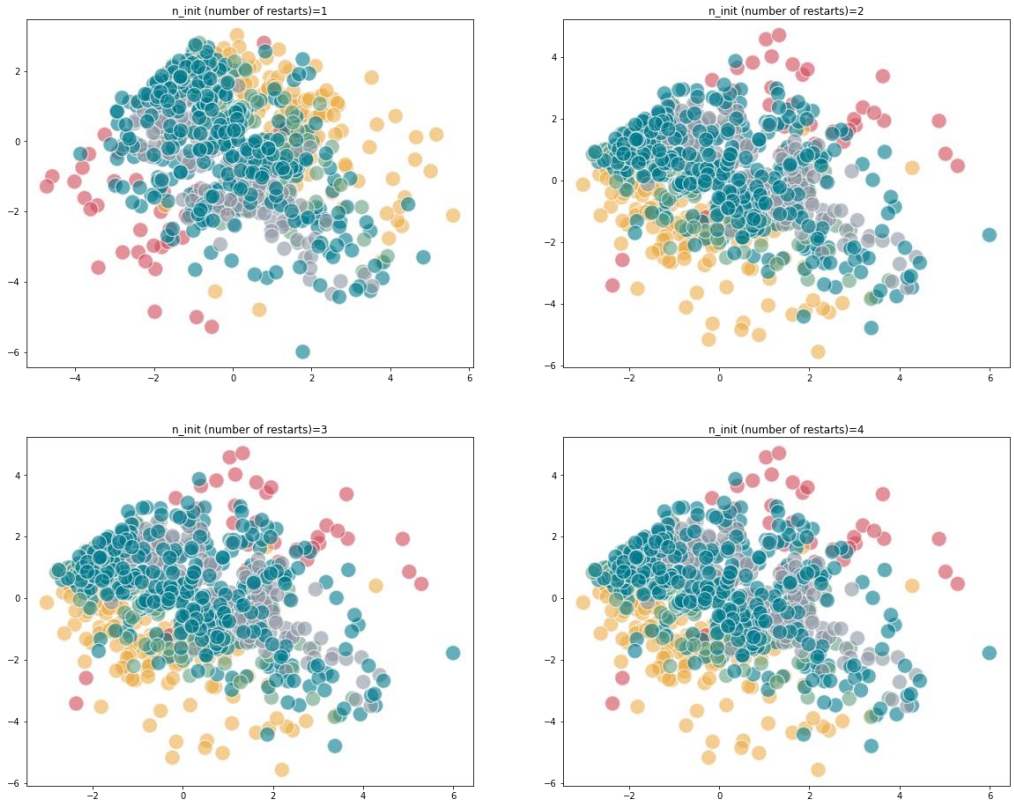
Pagal skirtingas atstumų metrikas:



15 pav. vizualizacija naudojant MDS keičiant atstumų metrikas

Iškart matome, kad euklidinė atstumų metrika yra ne veltui pasirenkama pagal nutylėjimą, nes beveik visi likę būdai klasių atsiskyrimus vaizduoja prasčiau. Kitos neblogos alternatyvos: Manheteno, Čebyševo atstumai. Apie duomenis kažko naujo atrasti nepavyko.

Pagal įtempimą minimizuojančios funkcijos SMACOF kartojimų skaičių (*n_init*). Vizualizuojamas tik geriausias variantas.

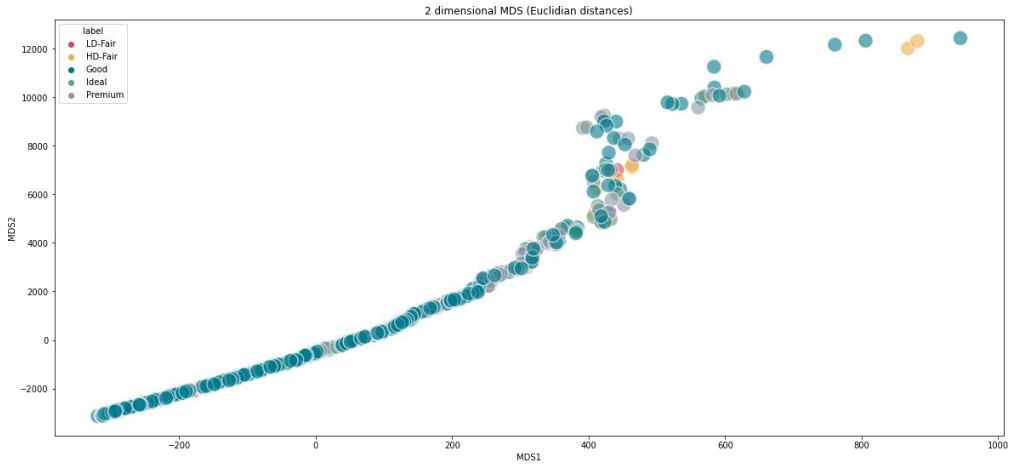


16 pav. vizualizacija naudojant MDS keičiant n_init

Matome, kad šį kartą vizualizacijai n_init didelės įtakos neturėjo. Tačiau kartais maža reikšmė gali nulemti algoritmo konvergavimą į lokalų minimumą, taip sukuriant prastesnę vizualizaciją. Pabandžius nubrėžti 3D grafikus, gavome labai panašius į 3D PCA.

Nenormuoti duomenys

Nubrėžę grafiką galime teigti, kad duomenų nenormavimas gali visiškai pakeisti galutinį vaizdą į blogąją pusę. Šiuo atveju nieko negalėtume pasakyti apie duomenis.



17 pav. MDS su nenormuotais duomenimis

t-SNE

Trečia dimensijos mažinimo metodą pasirinkome *t-paskirstytą stochastinį kaimynų įterpimą.*

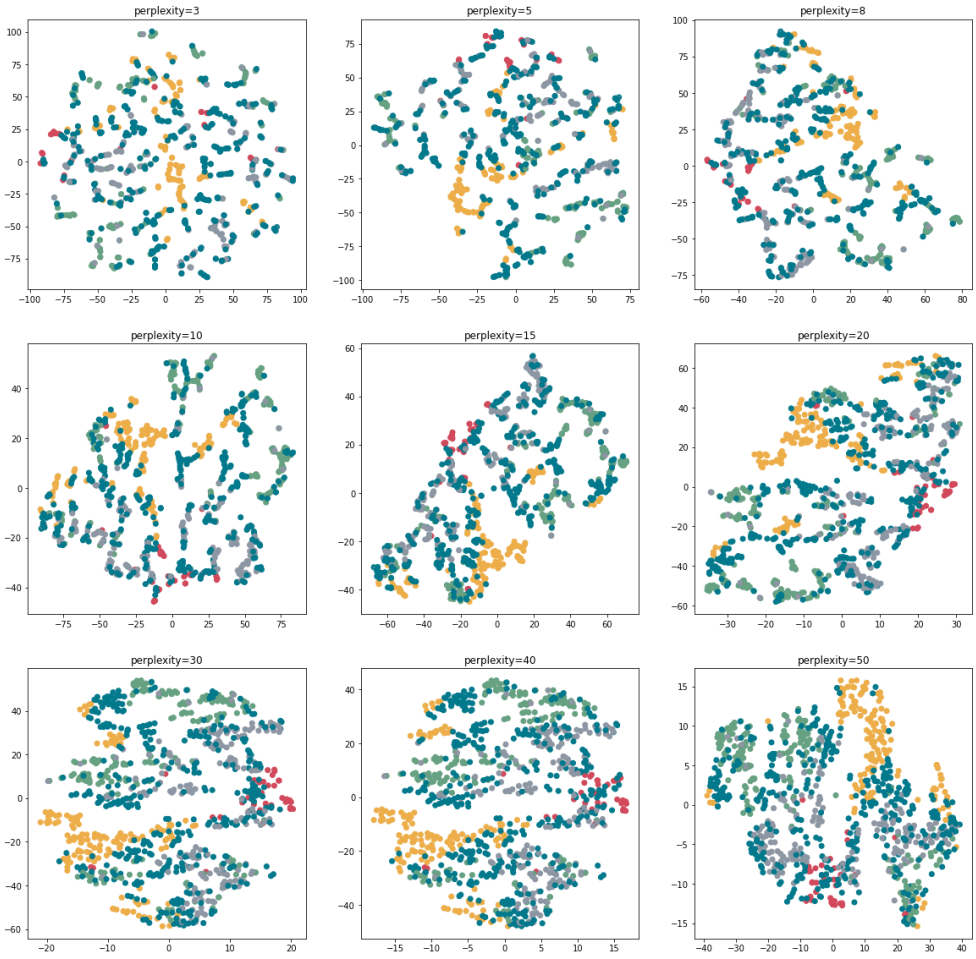
t-SNE yra patobulintas *Hintono* ir *Rowieso* pradinio stochastinio kaimyno įterpimo (SNE) variantas. Pagrindinė SNE idėja – minimizuoti skirtumą tarp sąlyginių tikimybių skirstinių, atspindinčių panašumus, apskaičiuotus duomenų taškams didelės ir mažos dimensijos vaizduose (žr. literatūrą).

18 pav. t-SNE taškinė diagrama

Deja t-SNE irgi nepadėjo mums geriau atskirti deimantų formų. Iš šio grafiko matome dar daugiau klasterių, kurie tik apsunkina interpretaciją arba indikuoja, kad mums trūksta informacijos.

Skirtingi grafikai keičiant algoritmo parametrus

Grafikai pagal skirtingus *„perplexity“*.

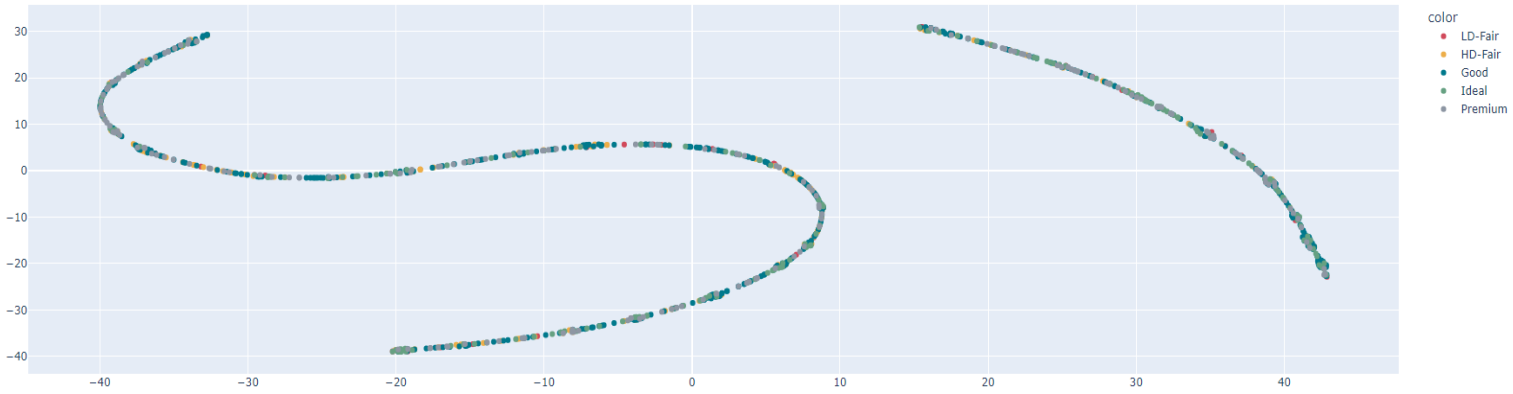


19 pav. vizualizacija naudojant t-SNE keičiant perplexity

Matome, kad nei vienas variantas nesuteikia daugiau informacijos nei turėjome prieš tai. Taip pat tam, kad pradėtų atsiskirti *fair* klasė, reikia gana didelio *perplexity*.

Nenormuoti duomenys

Šį kartą nenormuotų duomenų t-SNE grafikas yra dar sunkiau interpretuojamas, grupės visiškai neatsiskiria.



20 pav. Nenormuotų duomenų t-SNE

Panaudotų dimensijos mažinimo metodų privalumai ir trūkumai

PCA

Šio metodo privalumas – paprastumas. Modelis greitai pateikė atsakymus turimai aibei, kurie šį kartą niekuo nenusileido kitiems dimensijos mažinimo metodams. Šio metodo problema – su didėjančiu kovariančių skaičiumi krinta jo tikslumas (dispersija labiau išsibarsčiusi). Tačiau šiems duomenims vizualizacija 2D erdvėje buvo pakankamai gera (išlaikoma ~80 % visos dispersijos). Taip pat išskirtys gali iškraipyti rezultatus (jautri išskirtims).

MDS

Šis metodas leidžia pastebėti netiesinius sąryšius tarp taškų ir juos vizualizuoti išsaugant duomenų topologiją, kas praplatina dimensijos mažinimo galimybes. Taip pat šis metodas nėra toks jautrus išskirtims. Jo problema – ilgas veikimo laikas ir galimybė algoritmui konverguoti į lokalų minimumą (vietoje globalaus).

t–SNE

Šio metodo didelis privalumas – išsaugojimas vietinės struktūros, nesutelkimas taškų centre. Problema, su kuria susidūrėme ir mes – prarasta informacija apie grupių padėtį. Turėjome daug mažesnių klasterių, kurie nebūtinai taip išsidėstytų didesnės dimensijos erdvėje.

Išvados

Iš pradinės analizės nustatėme, kad klasės good ir very good reikšmingai nesiskiria, todėl jas sujungėme. Pirmą kart nubrėžę PCA pamatėme, kad klasę Fair sudaro du klasteriai. Juos išskaidėme į dvi klases. Iš PCA ir MDS mažinimo metodų gavome panašius rezultatus. Pagal 3D grafikus matėme, kad viena nuo kitos atsiskyrė *ideal* ir *premium* klasės. Nuo *fair* (abiejų) aiškiai atsiskyrė visos likusios klasės. *Good* klasė persidengė su *premium* ir *ideal* klasėmis. Tai indikuoja, kad mums trūksta duomenų norint atskirti deimantų formų (*cut*) klases pagal turimus parametrus. Tad jeigu norėtume tęsti darbą su šiais duomenimis (pavyzdžiui atlikdami regresijos polinominio atsako uždavinį), turėtume pabandyti rasti papildomų kintamųjų ir daugiau informacijos . Nepavyko panaudoti kategorinių kintamųjų papildomos informacijos, kas galbūt leistų susidaryti papildomas ir tikslesnes įžvalgas.

Šaltiniai

PCA

PCA. „Use of principal component analysis (PCA) and hierarchical cluster analysis (HCA) for multivariate association between bioactive compounds and functional properties in foods: A critical perspective“. Autoriai - Daniel Granatoa, Jânio S. Santosa , Graziela B. Eschera , Bruno L. Ferreirab , Rubén M. Maggio. 2018 vasaris. Nuoroda: Use of principal component analysis (PCA) and hierarchical cluster analysis (HCA) for multivariate association between bioactive compounds and functional properties in foods: A critical perspective - ScienceDirect

A Step-by-Step Explanation of Principal Component Analysis (PCA). Autorius: Zakaria Jaadi. 2021 04 01. Nuoroda: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

MDS

Knyga „Data Visualization With Multidimensional Scaling“. Autoriai Andreas Buja, Deborah F Swayne, Michael L Littman, Nathaniel Dean, Heike Hofmann, Lisha Chen. 2012 vasario 1. Nuoroda: Data Visualization With Multidimensional Scaling: Journal of Computational and Graphical Statistics: Vol 17, No 2 (tandfonline.com)

t-SNE

Straipsnis „Visualizing data using t-SNE“. Autoriai: L.v.d. Maaten ir G. Hinton, 2008. „Journal of Machine Learning Research“, 9 tomas. Nuoroda: https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf?fbclid=IwA

Wattenberg, et al., "How to Use t-SNE Effectively", Distill, 2016. Nuoroda: How to Use t-SNE Effectively (distill.pub)

Kodas

```
"""Diamonds_galutinis.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1gCX8VHxPKq4tvMsm3qUl7rEjVOdiTyWP

# PCA

## Duomenu nuskaitymas ir paruošimas

#### Naudojame 'cut' kaip musu target kintamaji
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from keras.datasets import mnist
from sklearn.manifold import MDS, smacof
import seaborn as sns
import plotly.express as px
from matplotlib.ticker import MaxNLocator

from sklearn.metrics.pairwise import pairwise_distances

# Naudojant google colab:
from google.colab import files
uploaded = files.upload()

# Nuskaitomi duomenys
df_visas = pd.read_csv("diamonds.csv")

df_visas = df_visas.drop("color", axis = "columns")
df_visas = df_visas.drop("clarity", axis = "columns")

# istriname pirma stulpeli, nes ten tiesiog ID
df_visas = df_visas.iloc[: ,1:]

"""#### Aprašomoji statistika"""

df_visas.describe()

# Toliau bus lenteles suskirstytos i cut grupes pagal:
# carat
df_visas[['cut', 'carat']].groupby('cut').describe()

# depth
df_visas[['cut', 'depth']].groupby('cut').describe()

# table
df_visas[['cut', 'table']].groupby('cut').describe()

# price
df_visas[['cut', 'price']].groupby('cut').describe()

# x
df_visas[['cut', 'x']].groupby('cut').describe()

# y
df_visas[['cut', 'y']].groupby('cut').describe()

# z
df_visas[['cut', 'z']].groupby('cut').describe()
```

```
"""#### Imties ėmimas"""

# Issirenkame po 200 kiekvieno tipo
df = df_visas.groupby('cut').apply(lambda x: x.sample(200, random_state =
1)).reset_index(drop=True)

# Turime po 300 kiekvieno 'cut'
df['cut'].value_counts()

"""### Išskirčių tikrinimas"""

# Staciakampes diagramos pagal cut
sns.set_palette(sns.color_palette(['#d1495b', '#00798c', '#66a182', '#8d96a3', '#957DAD']))

fig, axes = plt.subplots(3, 3, figsize=(20, 20))
sns.boxplot(ax=axes[0, 0], data=df, x='cut', y='carat').set(title = "Carat")
sns.boxplot(ax=axes[0, 1], data=df, x='cut', y='depth').set(title = "Depth")
sns.boxplot(ax=axes[0, 2], data=df, x='cut', y='table').set(title = "Table")
sns.boxplot(ax=axes[1, 0], data=df, x='cut', y='price').set(title = "Price")
sns.boxplot(ax=axes[1, 1], data=df, x='cut', y='x').set(title = "x")
sns.boxplot(ax=axes[1, 2], data=df, x='cut', y='y').set(title = "y")
sns.boxplot(ax=axes[2, 0], data=df, x='cut', y='z').set(title = "z")
fig.delaxes(ax=axes[2,1])
fig.delaxes(ax=axes[2,2])

df.boxplot('price')

def outlier(df):
    dfi = df._get_numeric_data()


    q1 = dfi.quantile(0.25)
    q3 = dfi.quantile(0.75)

    iqr = q3 - q1

    lower_bound = q1 -(3 * iqr)
    upper_bound = q3 +(3 * iqr)

    return (lower_bound, upper_bound)

# visi Q1 - 3iqr ir Q3 + 3 IQR
outlier(df)

# tik price
price_out = outlier(df)[1][3]
price_out

# Ismetame visas isskirtis:
df.drop(df[df.price >= price_out].index, inplace=True)
max(df.iloc[:,4])

# Isskirtis su carat
df.boxplot('carat')

out_carat = price_out = outlier(df)[1][0]
out_carat

# Ismetame carat isskirti:
df.drop(df[df.carat >= out_carat].index, inplace = True)
max(df.iloc[:,0])

# Isskirtis su depth
df.boxplot('depth')

# Matome isskirtis Q1-3IQR. Saliname kaip anksciau
out_depth = outlier(df)[0][1]
out_depth

# Ismetame carat isskirti:
```

```
df.drop(df[df.depth <= out_depth].index, inplace = True)
min(df.iloc[:,2])

# Isskirtis su table
df.boxplot('table')

out_table = outlier(df)[1][2]
df.drop(df[df.table >= out_table].index, inplace = True)
max(df.iloc[:,3])

df

"""Tyrime pirmuoju bandymu mūsų cut grupė 'fair' atrodė, lyg turėtų du klasterius, todėl dabar tikriname, dėl ko 'fair' sudaro du klasterius"""

plt.scatter(df[df["cut"] == "Fair"]["depth"], df[df["cut"] == "Fair"]["table"])
plt.xlabel("Depth")
plt.ylabel("Table")
plt.show()

"""Matome, kad klasė smarkiai atsiskiria pagal depth parametrą, todėl skirstysime į dvi grupes: LD-fair (low depth) ir HD-fair (high depth)"""

backup = df.copy()

df.loc[(df["depth"] >= 63 ) & (df["cut"] == "Fair"), "cut"] = "HD-Fair"
df.loc[(df["depth"] < 63) & (df["cut"] == "Fair"), "cut"] = "LD-Fair"

"""Tap pat iš stačiakampių diagramų matome, kad good ir very good mažai skiriasi, todėl šias grupes sujungiame."""

df.loc[df["cut"] == "Very Good", "cut"] = "Good"

df['cut'].unique()

"""### Koreliacijų matrica"""

df.corr()

"""Matome, kad kovaraintės carat, x, y, z koreliuoja, atmetame y (labai panašus į x, tai nėra didelio skirtumo kurį atmetame), z (turi išskirčių, daugiau koreliuoja ir su likusiomi kovariantėmis)."""

df = df.drop("y", axis = "columns")
df = df.drop("z", axis = "columns")

"""### Duomenų normavimas paga vidurkį ir dispersiją"""

from sklearn.preprocessing import StandardScaler
features = ['carat', 'depth', 'table', 'price', 'x']# stulpeliu pavadinimai
x_ro = df.loc[:, features].values # paimti duomenys be klases
y = df.loc[:,['cut']].values#  paimta klase
x_ = StandardScaler().fit_transform(x_ro) # duomenys normuoti pagal vidurki ir dispersija

df['cut'].unique()

# Sunormuotu duomenu lentele
dfnorm = pd.DataFrame(x_, columns = ['carat', 'depth', 'table', 'price', 'x'])
dfnorm

"""## PCA

### Normuoti duomenys
"""

n_comp = 3 # kiek komponentu norime
pca_list = []
for i in range(n_comp): # sukuriamas list objektas su stulpeliu pavadinimais (pca_1 pca_2 pca_3...)
    pca_list.append("pca_" + str(i+1))
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=n_comp)
principalComponents = pca.fit_transform(x_)
principalDf = pd.DataFrame(data = principalComponents, columns = pca_list)

principalDf

y_pca = df[['cut']].reset_index(drop = True)

finalDf = pd.concat([principalDf, y_pca], axis = 1) # prijungiami klasifikatoriaus reiksmes

finalDf

pca.explained_variance_ratio_

"""#### Vizualizavimas normuotų"""

# Normuoti
PC_values = np.arange(pca.n_components_) + 1
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2, color='blue')
plt.title('Scree PLot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()

fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['LD-Fair', 'HD-Fair', 'Good', 'Ideal', 'Premium']
colors = ['#d1495b', '#edae49', '#00798c', '#66a182', '#8d96a3']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['cut'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'pca_1']
               , finalDf.loc[indicesToKeep, 'pca_2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()

fig = plt.figure(figsize = (10,10))
ax = plt.axes(projection ="3d")
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_zlabel('Principal Component 3', fontsize = 15)
ax.set_title('3 component PCA', fontsize = 20)
targets = ['LD-Fair', 'HD-Fair', 'Good', 'Ideal', 'Premium']
colors = ['#d1495b', '#edae49', '#00798c', '#66a182', '#8d96a3']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['cut'] == target
    ax.scatter3D(finalDf.loc[indicesToKeep, 'pca_1']
               , finalDf.loc[indicesToKeep, 'pca_2']
               , finalDf.loc[indicesToKeep, 'pca_3']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()

# Papildomas argumetas 3d spalvoms del vientisumo
color_map={"LD-Fair": '#d1495b',
                    "HD-Fair": '#edae49',
                    "Good": '#00798c',
                    "Premium": '#8d96a3',
                    "Ideal": '#66a182'}

px.scatter_3d(finalDf, x='pca_1', y='pca_2', z='pca_3', color='cut',title="3 dimensional
PCA",color_discrete_sequence=px.colors.qualitative.Alphabet, color_discrete_map=color_map)

"""### Nenormuoti duomenys"""
```

```
pca = PCA(n_components=n_comp)
principalComponents = pca.fit_transform(x_ro)
principalDf = pd.DataFrame(data = principalComponents, columns = pca_list)

finalDf_ro = pd.concat([principalDf, y_pca], axis = 1) # prijungiami klasifikatoriaus reiksmes
print(pca.explained_variance_ratio_)

"""#### Vizualizavimas nenormuotų"""

# Nenormuoti scree plot
PC_values = np.arange(pca.n_components_) + 1
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()

# Nenormuoti taskine diagrama
fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['LD-Fair', 'HD-Fair', 'Good', 'Ideal', 'Premium']
colors = ['#d1495b', '#edae49', '#00798c', '#66a182', '#8d96a3']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf_ro['cut'] == target
    ax.scatter(finalDf_ro.loc[indicesToKeep, 'pca_1']
               , finalDf_ro.loc[indicesToKeep, 'pca_2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()

"""# MDS"""

def do_mds(data, labels, n_components = 2,metric=True,n_init=4,max_iter=1000,random_state=3000,
           dissimilarity="euclidean",verbose=0):
    mds=MDS(n_components=n_components,
          metric=metric,
          n_init=n_init,
          max_iter=max_iter,
          verbose=verbose,
          eps=0.00001,
          n_jobs=None,
          random_state = random_state,
          dissimilarity=dissimilarity)

    data_transformed = mds.fit_transform(data)
    stress = round(mds.stress_,0)
    print('Iterations: ', mds.n_iter_)
    print('Stress: ', stress)

    columns = ["MDS" + str(i) for i in range(1,n_components + 1)]
    toplot = pd.DataFrame(data_transformed,columns=columns)
    toplot["label"] = labels

    return toplot, stress

def plot_mds(data,title,ax=None,**kwargs):
    legend = None
    if ax is None:
        fig = plt.figure(figsize=(20, 9))
        ax = fig.add_subplot(1,1,1)
        legend = "auto"
    ax.set_title(title)
    sns.scatterplot(x="MDS1",y="MDS2",hue="label",data=data,alpha =
0.6,s=300,ax=ax,legend=legend,**kwargs)

def plot_mds3d(data):
    plot = px.scatter_3d(data, x='MDS1', y='MDS2', z='MDS3',
```

```
                color='label',title="3 dimensional MDS (Euclidian distances)",
color_discrete_sequence=px.colors.qualitative.Alphabet, color_discrete_map=color_map)
    return plot

"""### Normuoti duomenys"""

# vizualizacija sumazinus dimensija iki 2
sns.set_palette(sns.color_palette(['#d1495b', '#edae49', '#00798c', '#66a182', '#8d96a3']))

dfMDS , _ = do_mds(x_, y)

plot_mds(dfMDS,"2 dimensional MDS (Euclidian distances)")

def stress_plot(data,labels):
    stress = []
    n_dimensions = range(1,data.shape[1])
    for i in n_dimensions:
        _ , stress_value = do_mds(data,labels,n_components=i)
        stress.append(stress_value)

    ax = sns.lineplot(x=n_dimensions,y=stress)
    ax.xaxis.set_major_locator(MaxNLocator(integer=True))
    ax.set_xlabel("n_components")
    ax.set_ylabel("stress")
    ax.set_title("Stress by remaining components")

# matome, kaip stress stiprai padideja tik jeigu noretume sumazinti dim iki 1 (daroma pagal
kiekviena stulpeli)
stress_plot(x_, y)

dist_types = ['euclidean','manhattan', 'canberra', 'chebyshev', 'hamming', 'mahalanobis']

fig, ax = plt.subplots(3,2,figsize=(24, 20))
ax = ax.flatten()

for i,j in enumerate(dist_types):
    dist = pairwise_distances(x_, metric=j)
    dfMDS, _ = do_mds(dist,y,dissimilarity="precomputed")
    plot_mds(dfMDS,j + " distance",ax[i])
    ax[i].set_xlabel("")
    ax[i].set_ylabel("")

# n_init nustato kiek kartu atlikti SMACOF algoritma -> grazinamas tik geriausias rezultatas
# matoma, kad naudojant tik viena iteracija siuo atveju randamas tik lokalus minimumas

n_init = [1,2,3,4]

fig, ax = plt.subplots(2,2,figsize=(20, 16))
ax = ax.flatten()


for i,j in enumerate(n_init):
    dfMDS, _ = do_mds(x_,y,n_init=j)
    plot_mds(dfMDS,"n_init (number of restarts)=" + str(j),ax[i])
    ax[i].set_xlabel("")
    ax[i].set_ylabel("")

# vizualizacija sumazinus dimensija iki 3
dfMDS , _ = do_mds(x_,y,n_components=3)

plot_mds3d(dfMDS)

dfMDS

"""### Nenormuoti duomenys"""

# vizualizacija sumazinus dimensija iki 2
sns.set_palette(sns.color_palette(['#d1495b', '#edae49', '#00798c', '#66a182', '#8d96a3']))
dfMDS , _ = do_mds(x_ro, y)

plot_mds(dfMDS,"2 dimensional MDS (Euclidian distances)")
```

```
"""# T-SNE"""

# Sukuriame spalvu stulpeli del vientisumo
df1 = df.copy()
df1.loc[df1["cut"] == 'LD-Fair', "cutID"] = "#d1495b"
df1.loc[df1["cut"] == 'HD-Fair', "cutID"] = "#edae49"
df1.loc[df1["cut"] == 'Good', "cutID"] = "#00798c"
df1.loc[df1["cut"] == 'Ideal', "cutID"] = "#66a182"
df1.loc[df1["cut"] == 'Premium', "cutID"] = "#8d96a3"

# Modelio kurimas
from sklearn.manifold import TSNE
import plotly.express as px

tsne = TSNE(n_components=2, random_state=0)
projections = tsne.fit_transform(x_)

# Taskine diagrama
fig = px.scatter(
    projections, x=0, y=1,
    color=df1.cut, color_discrete_sequence=px.colors.qualitative.Alphabet,
color_discrete_map=color_map
)
fig.show()

# 3D TSNE
tsne = TSNE(n_components=3, random_state=0)
projections = tsne.fit_transform(x_)

fig = px.scatter_3d(
    projections, x=0, y=1, z=2,
    color=df1.cut, color_discrete_sequence=px.colors.qualitative.Alphabet,
color_discrete_map=color_map
)
fig.update_traces(marker_size=7)
fig.show()

projections

df1

data1 = TSNE(n_components=2, perplexity=3).fit_transform(x_)
data2 = TSNE(n_components=2, perplexity=5).fit_transform(x_)
data3 = TSNE(n_components=2, perplexity=8).fit_transform(x_)
data4 = TSNE(n_components=2, perplexity=10).fit_transform(x_)
data5 = TSNE(n_components=2, perplexity=15).fit_transform(x_)
data6 = TSNE(n_components=2, perplexity=20).fit_transform(x_)
data7 = TSNE(n_components=2, perplexity=30).fit_transform(x_)
data8 = TSNE(n_components=2, perplexity=40).fit_transform(x_)
data9 = TSNE(n_components=2, perplexity=50).fit_transform(x_)

plt.rcParams["figure.figsize"] = (20,20)
#plot 1:
plt.subplot(3, 3, 1)
plt.scatter(data1[:, 0], data1[:, 1], c=df1.cutID,cmap='tab10')
plt.title('perplexity=3')
#plot 2:
plt.subplot(3, 3, 2)
plt.scatter(data2[:, 0], data2[:, 1], c=df1.cutID,cmap='tab10')
plt.title('perplexity=5')
#plot 3:
plt.subplot(3, 3, 3)
plt.scatter(data3[:, 0], data3[:, 1], c=df1.cutID,cmap='tab10')
plt.title('perplexity=8')
#plot 4:
plt.subplot(3, 3, 4)
plt.scatter(data4[:, 0], data4[:, 1], c=df1.cutID,cmap='tab10')
plt.title('perplexity=10')
#plot 5:
plt.subplot(3, 3, 5)
```

```
plt.scatter(data5[:, 0], data5[:, 1], c=df1.cutID,cmap='tab10')
plt.title('perplexity=15')
#plot 6:
plt.subplot(3, 3, 6)
plt.scatter(data6[:, 0], data6[:, 1], c=df1.cutID,cmap='tab10')
plt.title('perplexity=20')
#plot 7:
plt.subplot(3, 3, 7)
plt.scatter(data7[:, 0], data7[:, 1], c=df1.cutID,cmap='tab10')
plt.title('perplexity=30')
#plot 8:
plt.subplot(3, 3, 8)
plt.scatter(data8[:, 0], data8[:, 1], c=df1.cutID,cmap='tab10')
plt.title('perplexity=40')
#plot 9:
plt.subplot(3, 3, 9)
plt.scatter(data9[:, 0], data9[:, 1], c=df1.cutID,cmap='tab10')
plt.title('perplexity=50')

"""### Nenormuoti duomenys"""

# Modelio kurimas
from sklearn.manifold import TSNE
import plotly.express as px

tsne = TSNE(n_components=2, random_state=0)
projections = tsne.fit_transform(x_ro)

# Taskine diagrama
fig = px.scatter(
    projections, x=0, y=1,
    color=df1.cut, color_discrete_sequence=px.colors.qualitative.Alphabet,
color_discrete_map=color_map
)
fig.show()
```