KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Intelektikos pagrindai

Laboratorinis darbas Nr. 1

Duomenų apdorojimas rinkinio analizė

Atliko:

IFF-7/13 grup. Stud:

Tautvydas Dikšas

Vertino:

Lekt. Germanas Budnikas

doc. Agnė Paulauskaitė-Tarasevičienė

KAUNAS, 2020

TURINYS

1. UŽDUOTIS

Pasirinkti duomenys turintys 16 stulpelių bei 16719 įrašų. Iš jų yra 10 tolydiniai ir 6 kategoriniai.
Laboratoriniui darbui atlikti naudotasi Python programavimo kalba.
Taip pat naudojamos bibliotekos: Pandas – duomenų skaitymui ir diagramų braižymui.
Naudojamos bibliotekos kode:

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
import seaborn as sn
from numpy import log as ln
from docx import Document
from docx.shared import Pt
```

Duomenų nuoroda: https://www.kaggle.com/sidtwr/videogames-sales-dataset#Video_Games_Sales_as_at_22_Dec_2016.csv

2. TOLYDINIO TIPO DUOMENŲ RINKINIO KOKYBĖS ANALIZĖ

Šioje lentelėje pateikta tolydinio tipo atributų kokybės analizė. Lentelė parodo, kad yra stulpelių, kuriems nemažai trūksta duomenų, todėl potencialiai galima jas nenaudoti.

Pavadinimas	Bendras reikšmių skaičius	Trūkstamų skaičius	Kardinalumas	Minimali reikšmė	Maksimali reikšmė	1-asis kvartilis	2-asis kvartilis	Vidurkis	Mediana	Standartinis nuokrypis
Year_of_Release	16719	1.609%	39	1980.0	2020.0	2003.0	2010.0	2006.487	2007.0	5.899
NA_Sales	16719	0.0%	402	0.0	41.36	0.0	0.24	0.263	0.08	0.855
EU_Sales	16719	0.0%	307	0.0	28.96	0.0	0.11	0.145	0.02	0.524
JP_Sales	16719	0.0%	244	0.0	10.22	0.0	0.04	0.078	0.0	0.318
Other_Sales	16719	0.0%	155	0.0	10.57	0.0	0.03	0.047	0.01	0.193
Global_Sales	16719	0.0%	629	0.01	82.53	0.06	0.47	0.534	0.17	1.637
Critic_Score	16719	51.331%	82	13.0	98.0	60.0	79.0	68.968	71.0	13.971
Critic_Count	16719	51.331%	106	3.0	113.0	12.0	36.0	26.361	21.0	18.983
User_Score	16719	54.603%	95	0.0	9.7	6.4	8.2	7.125	7.5	1.505
User_Count	16719	54.603%	888	4.0	10665.0	10.0	81.0	162.23	24.0	561.245

Tolydinių atributų lentelės rezultatų realizavimui naudojamas kodas:

```
class Con:
    def __init__(self, dataframe):
        self.__dataframe = dataframe
        self.__datalist = dataframe.values.tolist()

    @property
    def dataframe(self):
        return self.__dataframe

    @property
    def datalist(self):
        return self.__datalist

    def clearlist(self):
        return sorted([val for val in self.dataframe if not np.isnan(val)])

    def length(self):
        return len(self.dataframe)

    def missingpercent(self):
        return sum(np.isnan(val) for val in self.dataframe) / self.length() * 100

    def cardinality(self):
        return len(set(self.clearlist()))

    def max(self):
        return max(self.clearlist())

    def min(self):
        return min(self.clearlist())

    def firstquarter(self):
        index = len(self.clearlist()) % 4
        quarter = int(len(self.clearlist()) / 4)
        if index == 0 or index == 1:
            return (self.clearlist()[quarter] + self.clearlist()[quarter - 1]) / 2
        elif index == 2 or index == 3:
            return self.clearlist()[quarter]
```

```
    def thirdquarter(self):
        index = len(self.clearlist()) * 3 % 4
        quarter = int(len(self.clearlist()) * 3 / 4)
        if index == 0:
            return (self.clearlist()[quarter] + self.clearlist()[quarter -1]) / 2
        elif index == 3:
            return (self.clearlist()[quarter] + self.clearlist()[quarter + 1]) / 2
        elif index == 2 or index == 1:
            return self.clearlist()[quarter]

    def avg(self):
        return sum(self.clearlist()) / len(self.clearlist())

    def median(self):
        index = len(self.clearlist()) % 2
        middle = int(len(self.clearlist()) / 2)
        return self.clearlist()[middle] if index == 0 else (self.clearlist()[middle] +
 self.clearlist()[middle + 1]) / 2

    def variance(self):
        avg = int(self.avg())
        return sum([(val - avg) ** 2 for val in self.clearlist()]) / len(self.clearlis
t())

    def standartdev(self):
        return math.sqrt(self.variance())
```

3. KATEGORINIO DUOMENŲ RINKINIO KOKYBĖS ANALIZĖ

Šioje lentelėje pateikta kategorinio tipo atributų kokybės analizė. Kategorinių atributų problema yra aukštas kardinalumas.

Pavadinimas	**Bendras reikšmių skaičius**	**Trūkstamų skaičius**	**Kardinalumas**	**Moda**	**Modos dažnumas**	**Modos dažnumas procentais**	**2-oji moda**	**2-osios modos dažnumas**	**2-osios modos dažnumas procentais**
Name	16719	0.012%	11562	Need for Speed: Most Wanted	12	0.072%	Ratatouille	9	0.054%
Platform	16719	0.0%	31	PS2	2161	12.925%	DS	2152	12.872%
Genre	16719	0.012%	12	Action	3370	20.157%	Sports	2348	14.044%
Publisher	16719	0.323%	582	Electronic Arts	1356	8.111%	Activision	985	5.892%
Developer	16719	39.614%	1696	Ubisoft	204	1.22%	EA Sports	172	1.029%
Rating	16719	40.487%	8	E	3991	23.871%	T	2961	17.71%

Kategorinių atributų lentelės rezultatų realizavimui naudojamas kodas:

```
class Categoric:
    def __init__(self, dataframe):
        self.__dataframe = dataframe
        self.__datalist = dataframe.values.tolist()

    @property
    def dataframe(self):
        return self.__dataframe

    @property
    def datalist(self):
        return self.__datalist

    def clearlist(self):
        return [val for val in self.datalist if val is not np.nan]

    def length(self):
        return len(self.dataframe)

    def missingpercent(self):
        return sum( 1 for val in self.dataframe if val is np.nan) / self.length() * 10
0


    def cardinality(self):
        return len(set(self.clearlist()))

    def valueoccurences(self):
        alldata = self.clearlist()
        dictionary = {val:alldata.count(val) for val in set(self.clearlist())}
        return {k: v for k, v in sorted(dictionary.items(), key=lambda item: item[1])}

    def mode(self):
        return list(self.valueoccurences().keys())[-1]

    def modecount(self):
        return list(self.valueoccurences().values())[-1]

    def modepercentage(self):
        return self.modecount() / self.length() * 100

    def mode2(self):
        return list(self.valueoccurences().keys())[-2]

    def mode2count(self):
        return list(self.valueoccurences().values())[-2]
```

4. ATRIBUTŲ HISTOGRAMOS

Tolydinių atributų histogramoms realizuoti naudojamas kodas:

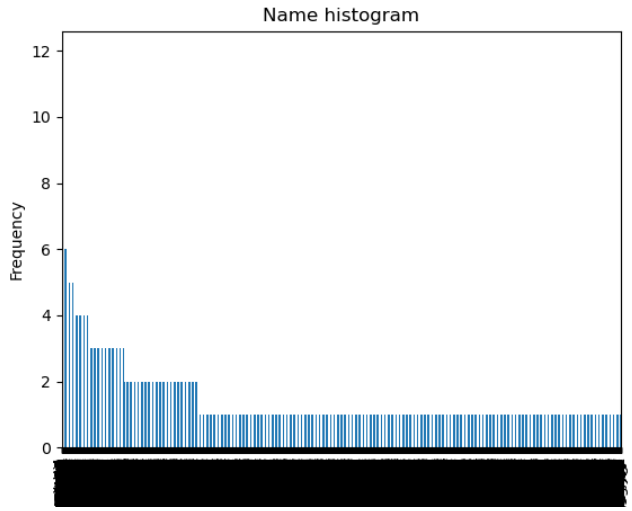
```
def drawhistogram(data):
    cont, categ = sepheaders(data)
    for header in cont:
        plt.figure(num=header)
        binvalue =int(1 + 3.22 * ln(len(data[header])))
        data[header].hist(bins=binvalue)
        plt.xlabel('Values')
        plt.ylabel('Frequency')
        plt.title(header+" histogram")
    plt.show()
```

Kategorinio tipo atributų histogramoms realizuoti naudojamas kodas:

```
def drawbarplotcategoric(data):
    cont, categ = sepheaders(data)
    for header in categ:
        plt.figure(num=header)
        data[header].value_counts().plot(kind='bar')
        plt.xlabel('Values')
        plt.ylabel('Frequency')
        plt.title(header+" histogram")
    plt.show()
```

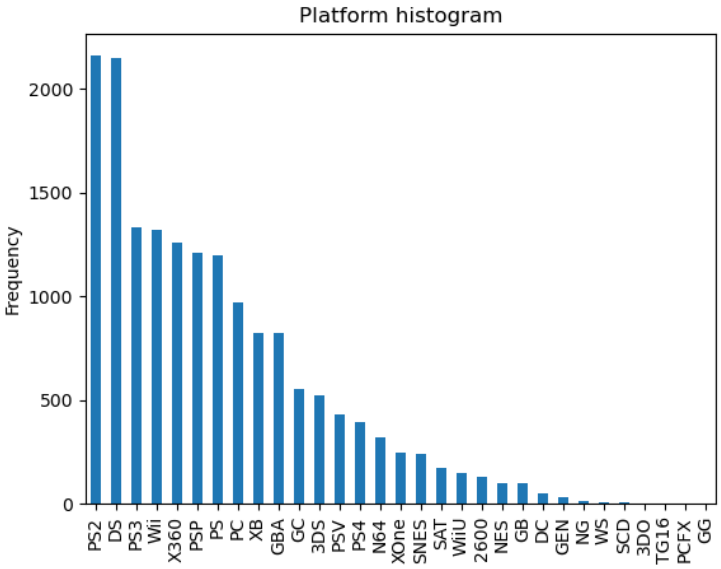
Histogramos:

Pav. 4.1 matome kaip atrodo situaciją, kai yra daug skirtingų pavadinimų t.y. didelis kardinalumas.



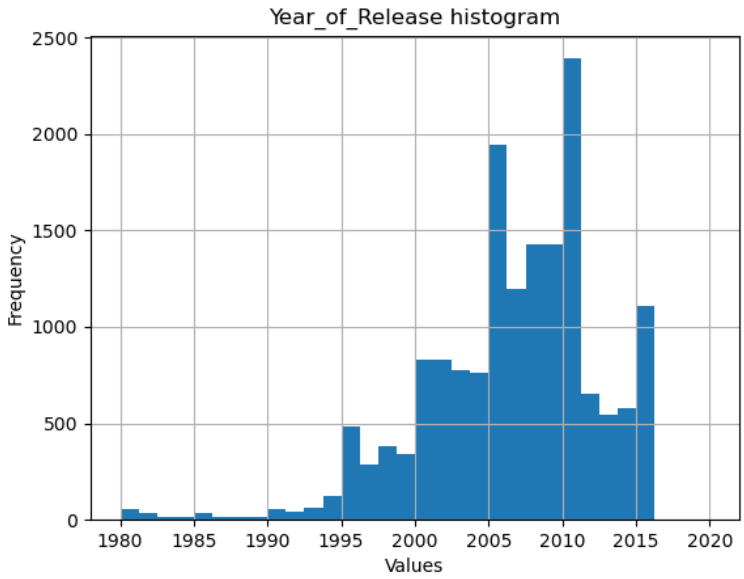
pav. 4.1 Atributo Name histograma

Pav. 4.2 mažėjančiai pasiskirsčiusias platformų kategorijas. ,,Right-skewed" pasiskirstymas.



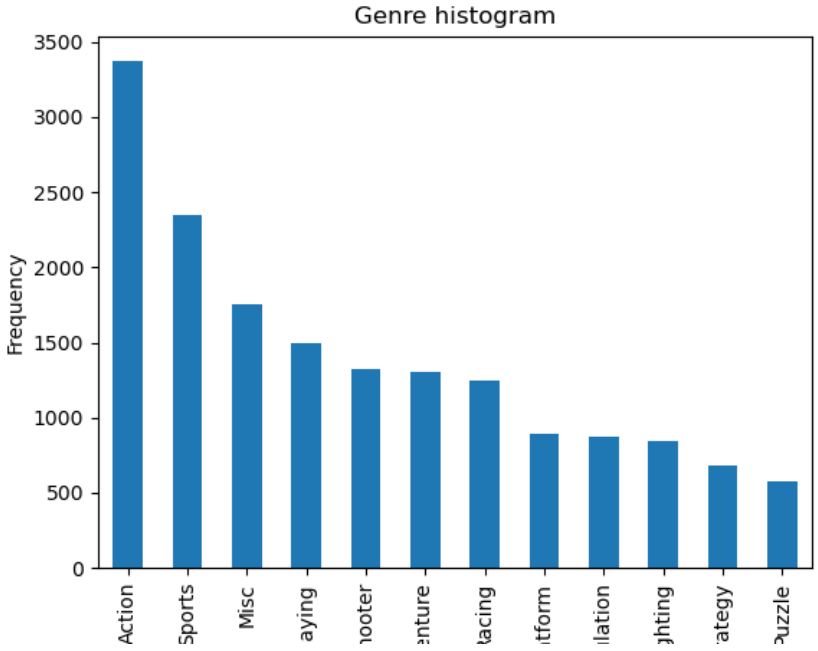
pav. 4.2 Atributo Platform histograma

Pav. 4.3 atvaizduoja year_of_Release histogramą. „Random” pasiskirstymas.



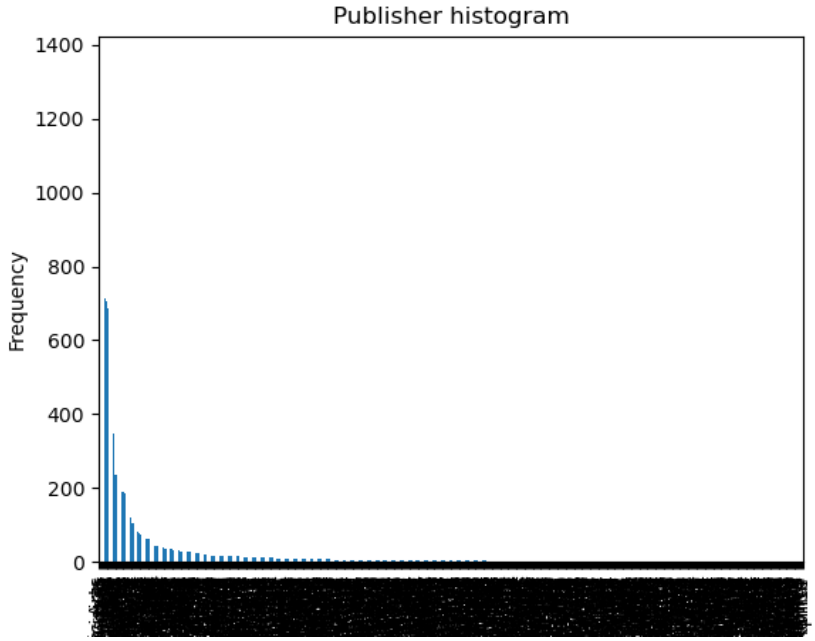
pav. 4.3 Atributo Year of Release histograma

Pav. 4.4 atvaizduoja Genre histogramą. Dėšinėje nukreiptas pasiskirstymas. „Right-skewed” pasiskirstymas



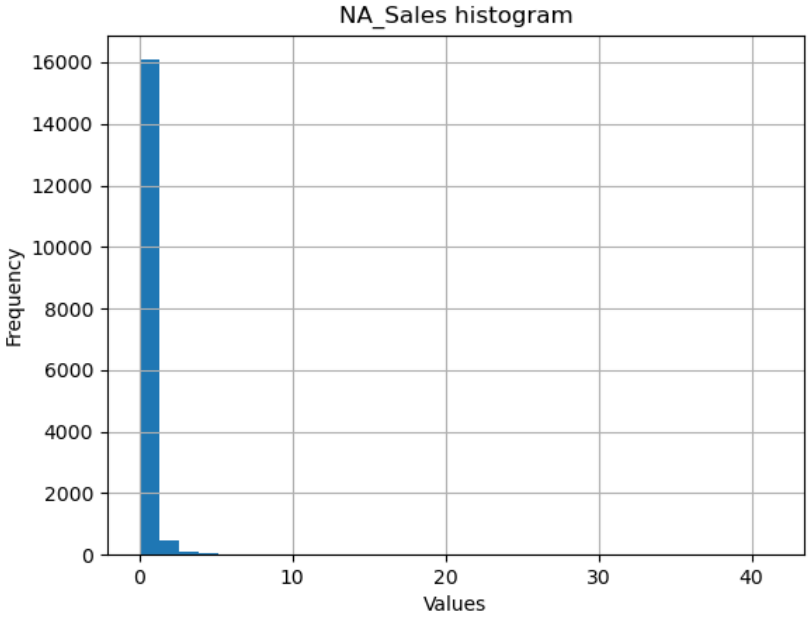
pav. 4.4 Atributo Genre histograma

Pav. 4.5 atvaizduoja publisher histogramą. Diagramoje matoma didelio kardinalumo problema. Eksponentinis mažėjantis pasiskirstymas.



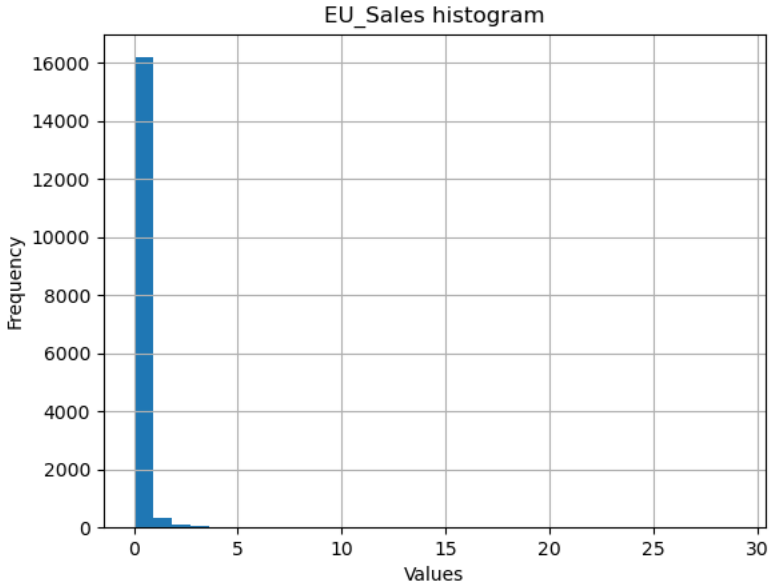
pav. 4.5 Atributo Publisher histograma

Pav. 4.6 atvaizduoja NA_Sales histogramą. Joje matome, kad turi ekstremalių reikšmių, todėl vienas „bin" yra žymiai aukštesnis už kitus. Reikia pasirinkti interval, kad diagrama išsilygintų.



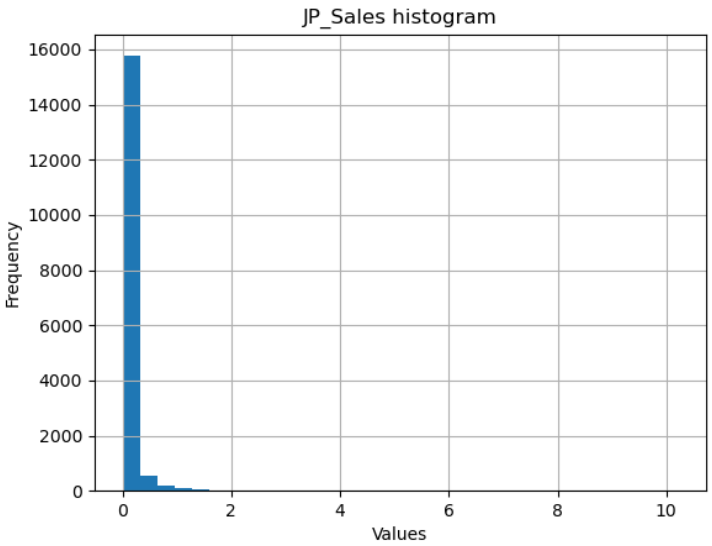
pav. 4.6 Atributo NA Sales histograma

Pav. 4.7 atvaizduoja EU_Sales histogramą. Joje matome, kad turi ekstremalių reikšmių, todėl vienas „bin" yra žymiai aukštesnis už kitus. Reikia pasirinkti interval, kad diagrama išsilygintų.



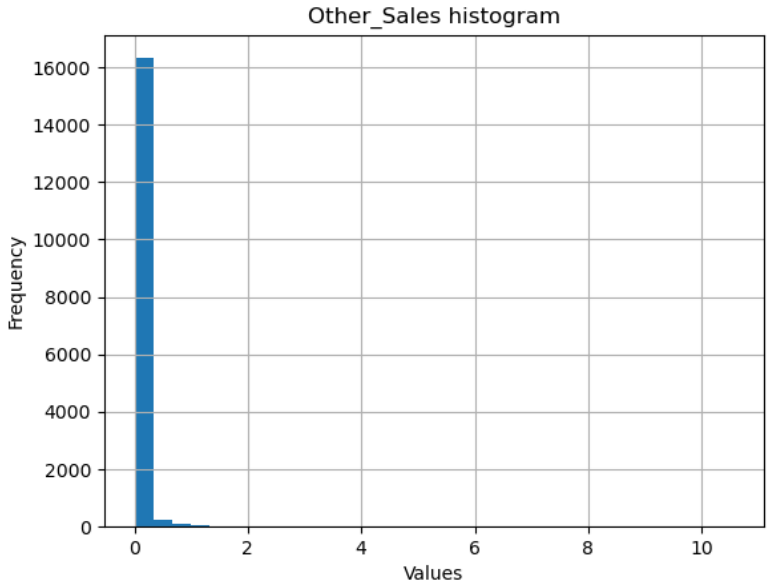
pav. 4.7 Atributo EU Sales histograma

Pav. 4.8 atvaizduoja JP_Sales histogramą. Joje matome, kad turi ekstremalių reikšmių, todėl vienas „bin" yra žymiai aukštesnis už kitus. Reikia pasirinkti interval, kad diagrama išsilygintų.



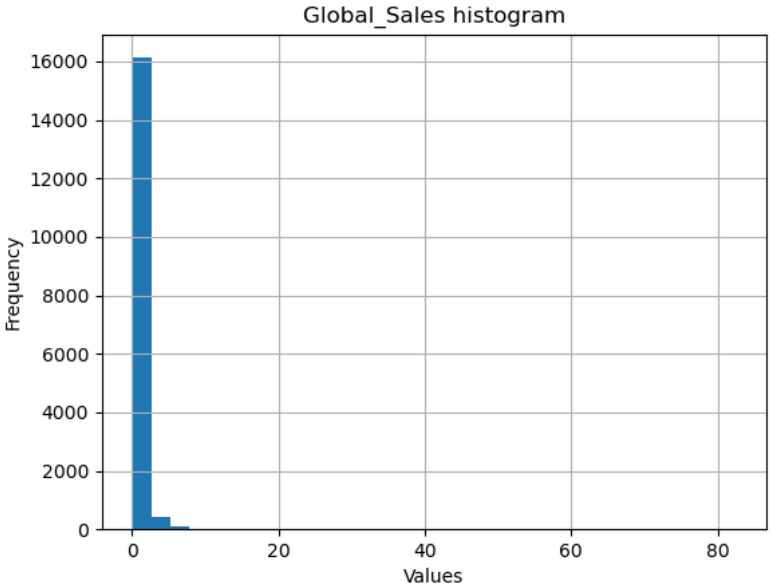
pav. 4.8 Atributo JP Sales histograma

Pav. 4.9 atvaizduoja Other_Sales histogramą. Joje matome, kad turi ekstremalių reikšmių, todėl vienas „bin" yra žymiai aukštesnis už kitus. Reikia pasirinkti interval, kad diagrama išsilygintų.



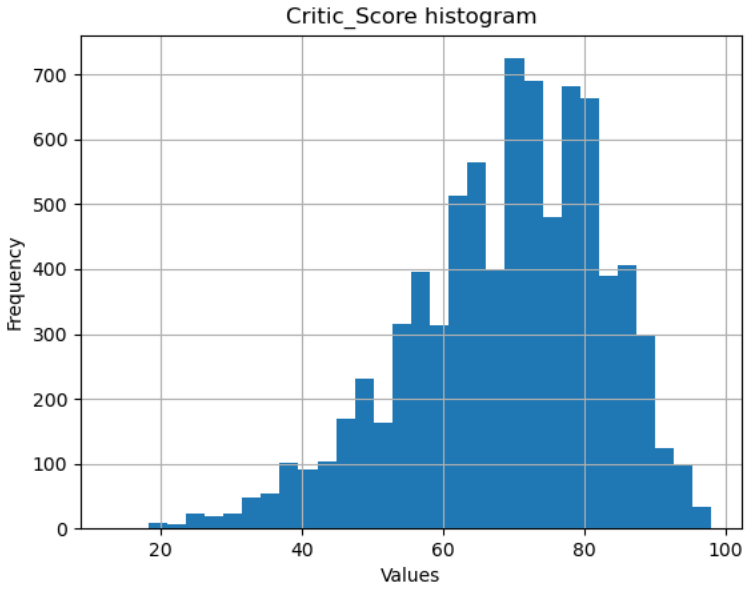
pav. 4.9 Atributo Other Sales histograma

Pav. 4.10 atvaizduoja Global_Sales histogramą. Joje matome, kad turi ekstremalių reikšmių, todėl vienas „bin" yra žymiai aukštesnis už kitus. Reikia pasirinkti interval, kad diagrama išsilygintų.



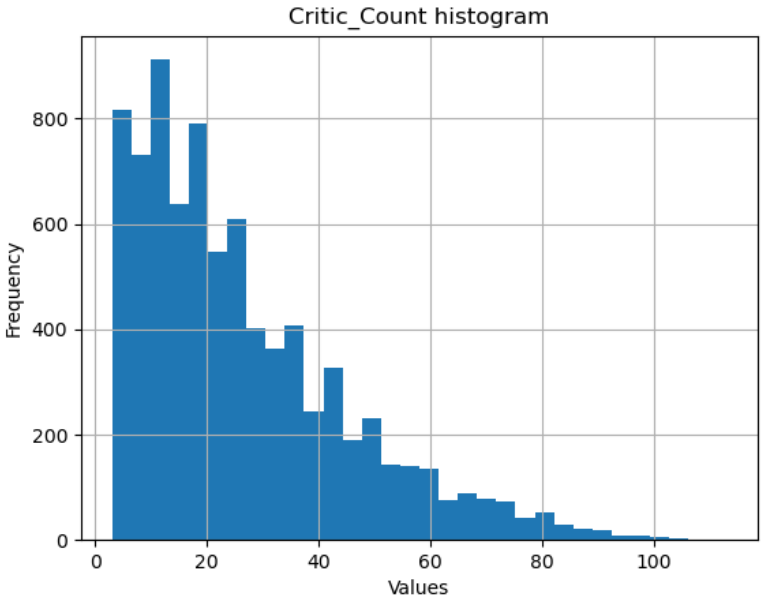
pav. 4.10 Atributo Global Sales histograma

Pav. 4.11 parodo atributo Critic_Score histogramą. Beveik normalusis pasiskirstymas



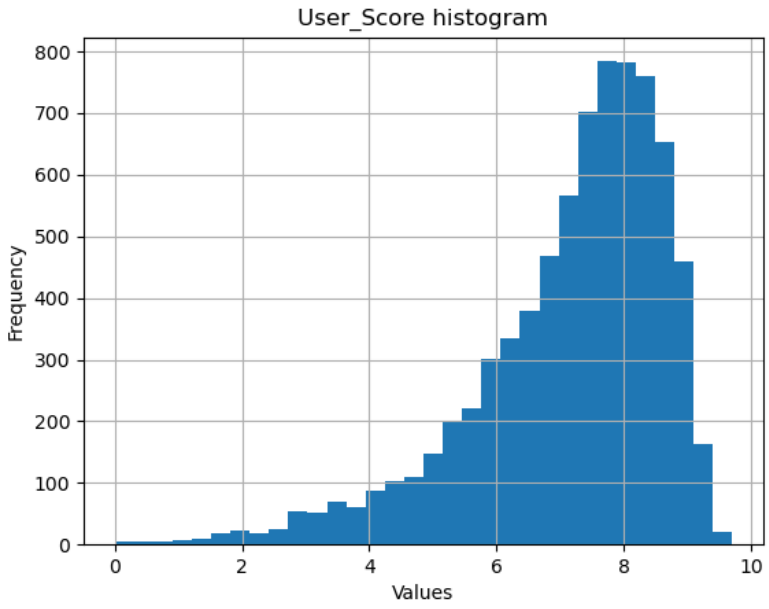
pav. 4.11 Atributo Critic Score histograma

Pav. 4.12 atvaizduoja Critic_Count histogramą. „Right-Skewed" pasiskirstymas



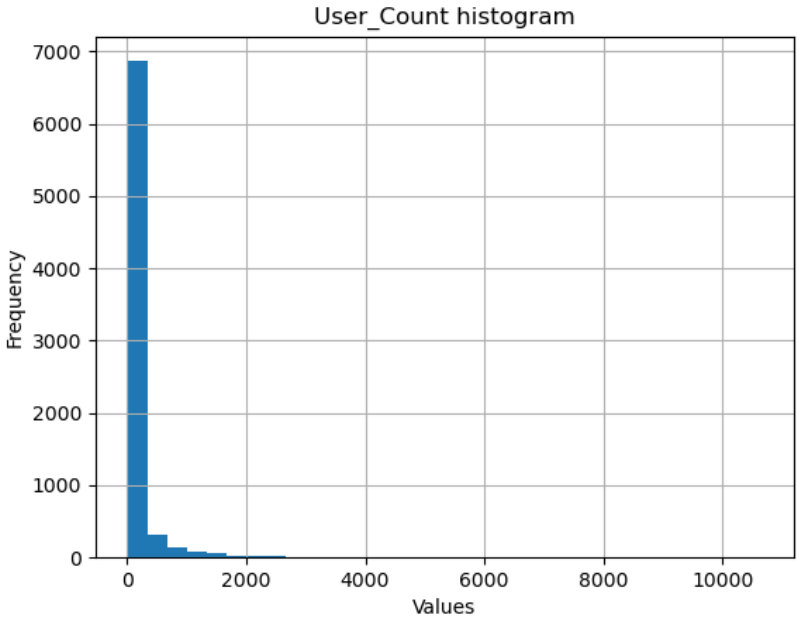
pav. 4.12 Atributo Critic Count histograma

Pav. 4.13 atvaizduota User_Score histograma. Eksponentinis didėjantis pasiskirstymas.



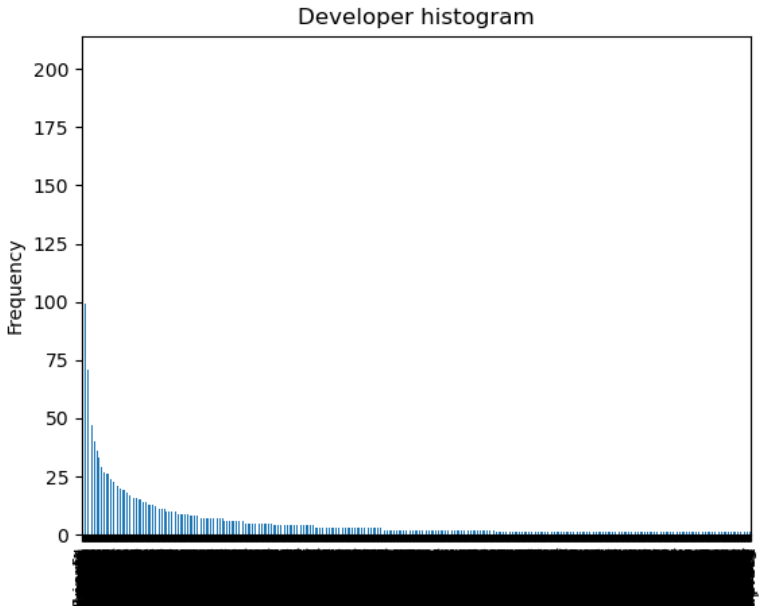
pav. 4.13 Atributo User Score histograma

Pav. 4.14 atvaizduoja User_Count histogramą. Atributo duomenyse yra ekstremalios reikšmės, todėl reikia pasirinkti intervalą, kad diagrama susilygintų.



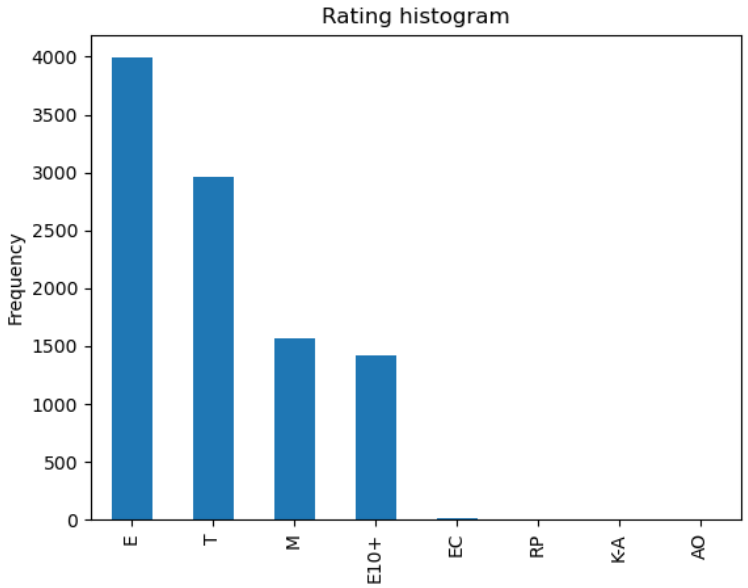
pav. 4.14 Atributo User Count histograma

Pav. 4.15 parodo atributo Developer histrogramą. Eksponentinis mažėjantis



pav. 4.15 Atributo Developer histograma

Pav. 4.16 atvaizduoja Rating atributo diagrama. Dešinėje pusėje nukreiptas pasiskirstymas (mažėjantis) t.y. „right skewed"



pav. 4.16 Atributo Rating histograma

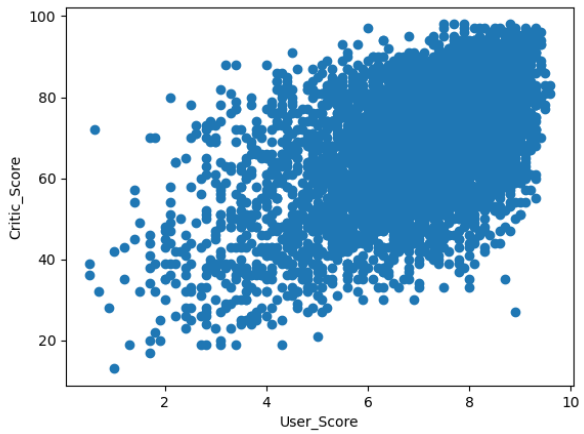
5. DUOMENŲ KOKYBĖS PROBLEMŲ IDENTIFIKAVIMAS

Duomenų rinkinyje buvo rastos kelios kokybės problemos. Pirma problema buvo trūkstamos reikšmės. Atvaizduojant histogramas tos eilutės buvo praleidžiamos. Antroji problema – ekstremalios reikšmės. Šios problemos sprendimo būdas gali būti intervalo nustatymas.

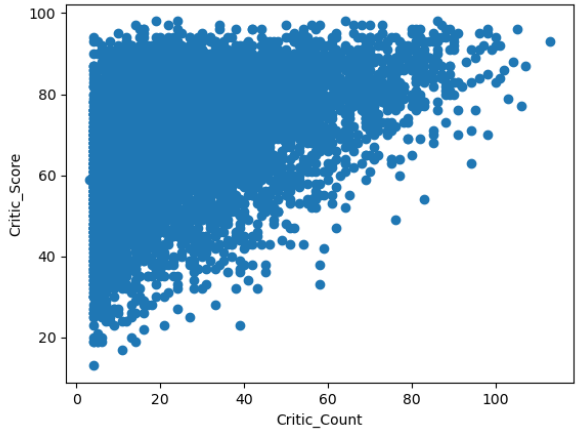
6. ATRIBUTŲ SĄRYŠIŲ NUSTATYMAS

Stipri koreliacija:

Stipri koreliacija parodo, kad yra ryšys tarp dviejų atributų. Būtent tai parodo Pav. 6.1 ir Pav. 6.2



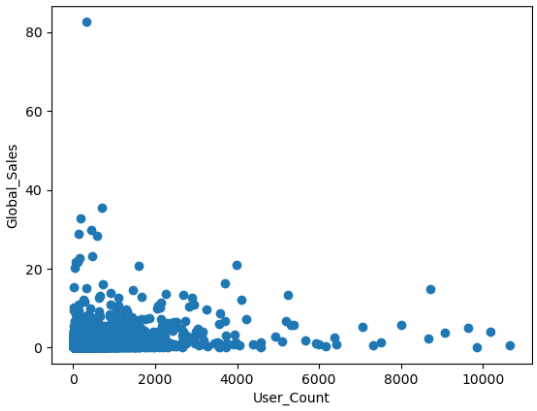
pav. 6.1 Atributų Critic_Score ir User_Score priklausomybė



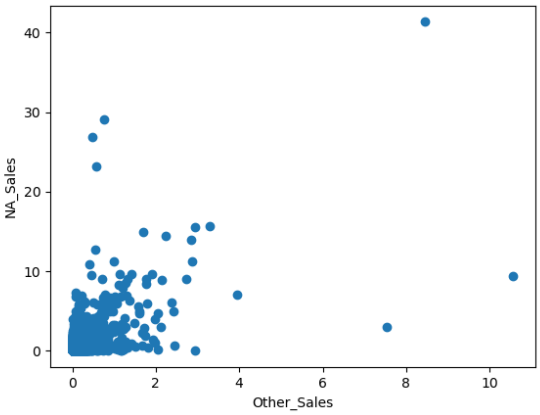
pav. 6.2 Atributų Critic_Score ir Critic_Count priklausomybė

Silpna koreliacija:

Silpna koreliacija parodo, kad ryšio tarp atributų tikriausiai nėra. Pav. 6.3 ir Pav. 6.4 parodo, kad pas juos silpna koreliacija ir pasikeitus duomenims jų diagramai įtakos neturės.



pav. 6.3 Atributų Global_Sales ir User_Count priklausomybė



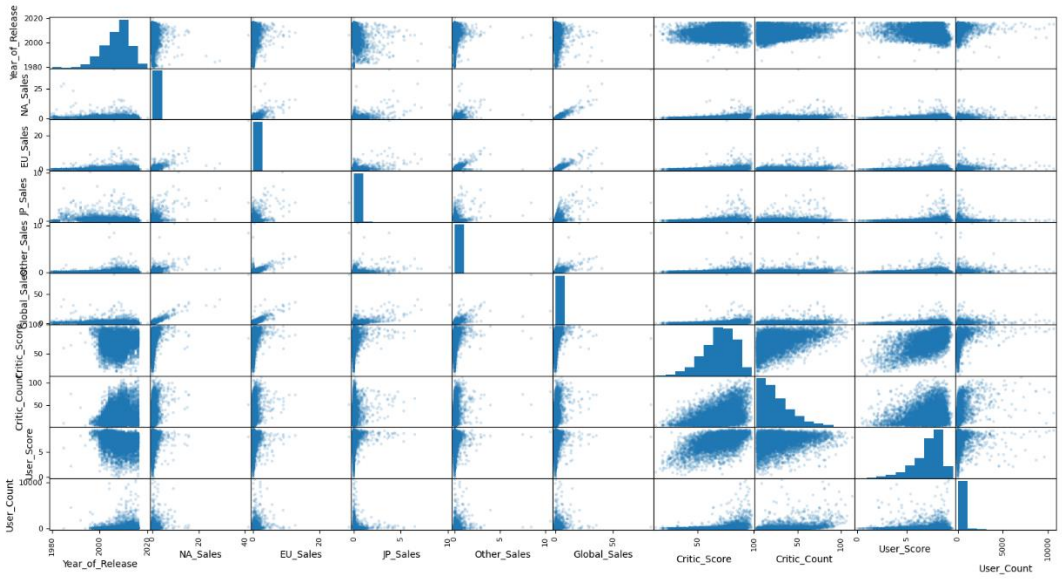
pav. 6.4 Atributų NA_Sales ir Other_Sales priklausomybė

Realizavimui naudotas kodas:

```
def drawscatter(data, header, header1):
    plt.figure(num=header+header1)
    plt.scatter(data[header], data[header1])
    plt.xlabel(header)
    plt.ylabel(header1)
    plt.show()
```

6.1. SCATTER PLOT MATRIX DIAGRAMA:

Pav. 6.1 atvaiduoja scatter plot matrix diagramą, kuri parodo atributų priklausomybes. Diagramoje esančios histogramos rodo, kad priklausomybė lyginima su savimi.



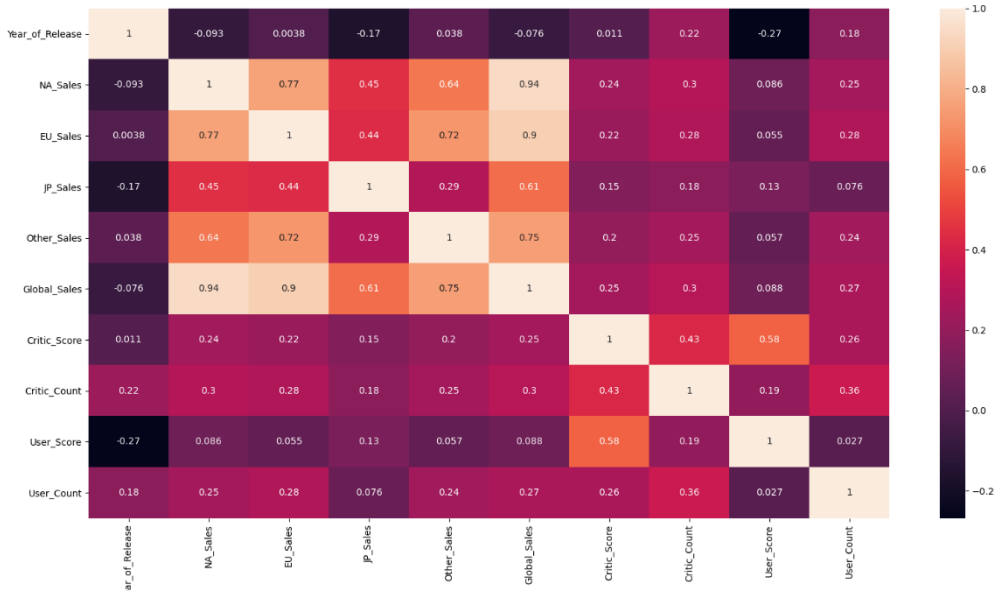
pav. 6.5 SPLOM diagrama

Diagramos atvaizdavimas kodu:

```
def splom(data):
    cont, categ = sepheaders(data)
    pd.plotting.scatter_matrix(data, alpha=0.2)
    plt.show()
```

7. KORELIACIJOS MATRICA

Pav. 7.1 išspausdinta tolydinių atributų koreliacijos matrica. Joje matome, kad stipri koreliacija yra tarp NA_Sales ir Global_Sales, NA_Sales ir EU_Sales, EU_Sales ir Global_Sales. Nėra koreliacijos matoma su Year_of_Release atributu, todėl sprendimas būtų išmesti jį arba paversti kategoriniu.



pav. 7.1 Tolydinių atributų koreliacijos matrica

Šios diagramos atvaizdavimas kodu:

```
def drawcorrelation(data):
    corrMatrix = data.corr()
    sn.heatmap(corrMatrix, annot=True)
    plt.show()
```

8. IŠVADOS

- Analizuojant duomenų rinkinį nustatyta, kurie atributai tarpusavyje priklausomi
- Atliekant duomenų rinkinio analizę pastebėta, kad yra stulpelių, kuriems trūksta >50% duomenų
- Duomenų rinkinyje pastebėta daug stulpelių, kurie visiškai nepriklausantys, todėl buvo galima juos ištrinti.