Uniwersytet Pedagogiczny im. Komisji Edukacji Narodowej
Instytut Bezpieczeństwa i Informatyki

PROJEKT INŻYNIERSKI
DOKUMENTACJA UŻYTKOWA

EKSPLORACJA DANYCH ZA POMOCĄ TOPOLOGICZNEJ ANALIZY DANYCH NA PRZYKŁADZIE AMERYKAŃSKO-KANADYJSKIEJ LIGI KOSZYKARSKIEJ NBA

wykonany przez:

Piotr Maliga
Nr albumu: 145892
&
Krzysztof Wszołek
Nr albumu: 145948
&

pod opieką:
dr. Marcin Żelawski
mgr. Patryk Mazurek

Kraków 2023
(ostatnia aktualizacja: 17:55:08, 2023-01-29)

Spis treści

1 Szczegółowa dokumentacja projektowa **1**

1.1 Projekt bazy danych 1

1.2 Szczegółowa dokumentacja kodu 3

1.3 Środowisko programistyczne 8

Bibliografia9

1 Szczegółowa dokumentacja projektowa

W zależności od specyfiki projektu! Wymienione niżej podpunkty mają charakter orientacyjny.

1.1 Projekt bazy danych

Opis kolumn

- RK - ranking w danym sezonie
- Team - nazwa drużyny
- TRB - całkowita lczba zbiórek (w offensywie i w defensywie)
- AST - Liczba asyst
- STL - przechwycenie piłki
- BLK - liczba bloków
- PTS - liczba zdobytych punktów

Tabele:

Rk	Team	TRB	AST	STL	BLK	PTS
1	Denver Nuggets	4050	2005	856	406	9828
2	Golden State Warriors*	3419	1954	803	378	9564
3	Portland Trail Blazers*	3763	2254	724	410	9407
4	Phoenix Suns*	3730	2209	687	535	9348
5	Indiana Pacers*	3394	2181	658	357	9159
6	Boston Celtics*	3785	2160	672	565	9145
7	Chicago Bulls*	3490	2212	822	438	9024
8	Atlanta Hawks*	3655	1864	729	374	9003
9	San Antonio Spurs*	3788	2140	670	571	8782
10	Houston Rockets*	3783	1906	796	409	8753
11	Seattle SuperSonics*	3395	2042	861	380	8744
12	Milwaukee Bucks*	3241	2075	894	330	8727
13	Los Angeles Lakers*	3518	2091	642	384	8717
14	Orlando Magic	3662	1809	602	306	8684
15	Philadelphia 76ers*	3480	1824	678	479	8641
16	Utah Jazz*	3341	2217	652	451	8527
17	Los Angeles Clippers	3746	2119	725	507	8491
18	New York Knicks*	3489	2172	638	418	8455
19	New Jersey Nets	3748	1782	748	600	8441
20	Charlotte Hornets	3227	2019	759	304	8428
21	Miami Heat	3534	1904	756	387	8349
22	Cleveland Cavaliers	3340	2240	643	450	8343
23	Washington Bullets	3563	2081	588	468	8313
24	Detroit Pistons*	3658	1825	487	367	8205
25	Dallas Mavericks	3344	1821	581	397	8195
26	Minnesota Timberwolves	3388	1885	712	440	8169
27	Sacramento Kings	3245	1991	631	513	7928
	League Average	**3547**	**2029**	**704**	**431**	**8717**

Rk	Team	TRB	AST	STL	BLK	PTS
1	Golden State Warriors*	3513	2064	854	375	9732
2	Indiana Pacers*	3647	2398	705	393	9197
3	Phoenix Suns*	3646	2202	673	582	9194
4	Portland Trail Blazers*	3843	2065	753	410	9135
5	Chicago Bulls*	3612	2279	672	480	9011
6	Charlotte Hornets	3531	2284	822	309	8980
7	Cleveland Cavaliers*	3491	2260	616	621	8926
8	Utah Jazz*	3640	2188	715	448	8877
9	Boston Celtics*	3678	2072	636	484	8745
10	Seattle SuperSonics*	3539	1877	775	448	8737
11	Atlanta Hawks	3786	2123	793	320	8711
12	New Jersey Nets*	3904	1937	736	615	8641
13	Milwaukee Bucks	3469	2018	863	317	8609
14	Miami Heat*	3553	1749	670	373	8608
15	Sacramento Kings	3408	1957	727	618	8549
16	San Antonio Spurs*	3781	2010	729	608	8524
17	Los Angeles Clippers*	3525	2053	824	498	8440
18	Washington Bullets	3414	2011	713	422	8395
19	Houston Rockets	3506	2058	656	571	8366
20	Philadelphia 76ers	3367	1755	692	482	8358
21	Orlando Magic	3500	1792	643	367	8330
22	New York Knicks*	3674	2130	634	382	8328
23	Minnesota Timberwolves	3335	2025	619	526	8237
24	Los Angeles Lakers*	3352	1803	756	400	8229
25	Denver Nuggets	3702	1553	773	461	8176
26	Detroit Pistons*	3631	1899	546	357	8113
27	Dallas Mavericks	3633	1630	536	349	8007
	League Average	**3581**	**2007**	**709**	**452**	**8635**

Rk	Team	TRB	AST	STL	BLK	PTS
1	Phoenix Suns*	3651	2087	752	455	9298
2	Charlotte Hornets*	3603	2161	639	473	9030
3	Golden State Warriors	3603	2010	693	383	9014
4	Portland Trail Blazers*	3733	1969	770	425	8898
5	Seattle SuperSonics*	3476	1906	944	409	8884
6	Sacramento Kings	3418	2075	768	348	8847
7	Indiana Pacers*	3675	2144	615	403	8836
8	Cleveland Cavaliers*	3425	2349	615	536	8832
9	Atlanta Hawks*	3634	2084	806	278	8814
10	Los Angeles Clippers*	3543	2242	847	491	8783
11	Utah Jazz*	3504	2177	746	344	8709
12	Orlando Magic	3606	1952	542	467	8652
13	San Antonio Spurs*	3461	2012	582	514	8652
14	Denver Nuggets	3830	1735	651	565	8626
15	Chicago Bulls*	3573	2133	783	410	8625
16	Philadelphia 76ers	3462	2038	672	566	8556
17	Los Angeles Lakers*	3391	2013	782	431	8546
18	Houston Rockets*	3517	2115	682	543	8531
19	Boston Celtics*	3512	1999	647	458	8502
20	Miami Heat	3518	1688	609	350	8495
21	New Jersey Nets*	3797	1872	693	526	8431
22	Milwaukee Bucks	3163	2084	863	393	8392
23	Washington Bullets	3348	2110	673	359	8353
24	New York Knicks*	3810	2125	680	372	8328
25	Detroit Pistons	3608	1941	580	249	8252
26	Dallas Mavericks	3499	1683	649	355	8141
27	Minnesota Timberwolves	3144	2001	649	455	8046
	League Average	**3537**	**2026**	**701**	**428**	**8632**

Rk	Team	TRB	AST	STL	BLK	PTS
1	Chicago Bulls*	3658	2033	745	345	8625
2	Seattle SuperSonics*	3403	1999	882	393	8572
3	Orlando Magic*	3367	2080	663	406	8571
4	Phoenix Suns*	3510	2001	623	331	8552
5	Boston Celtics	3477	1792	653	406	8495
6	San Antonio Spurs*	3523	2044	645	536	8477
7	Los Angeles Lakers*	3298	2080	722	516	8438
8	Charlotte Hornets	3243	1907	582	277	8431
9	Dallas Mavericks	3787	1913	642	342	8409
10	Washington Bullets	3257	1815	592	506	8408
11	Houston Rockets*	3374	1982	645	476	8404
12	Utah Jazz*	3366	2139	667	418	8404
13	Golden State Warriors	3458	1889	706	470	8334
14	Sacramento Kings*	3459	1829	643	436	8163
15	Los Angeles Clippers	3169	1672	703	411	8153
16	Portland Trail Blazers*	3737	1760	594	417	8145
17	Indiana Pacers*	3272	1917	579	323	8144
18	Atlanta Hawks*	3330	1609	771	319	8059
19	Minnesota Timberwolves	3256	1867	650	481	8024
20	Denver Nuggets	3544	1851	521	597	8013
21	Toronto Raptors	3284	1927	745	493	7994
22	New York Knicks*	3278	1822	645	377	7971
23	Miami Heat*	3494	1752	574	439	7909
24	Milwaukee Bucks	3137	1755	582	307	7837
25	Detroit Pistons*	3324	1610	506	352	7822
26	Philadelphia 76ers	3192	1629	643	420	7746
27	New Jersey Nets	3853	1752	627	571	7684
28	Cleveland Cavaliers*	2922	1818	674	340	7473
29	Vancouver Grizzlies	3127	1706	728	333	7362

Rk		TRB	AST	STL	BLK	PTS
1	Michael Jordan	492	453	223	83	2580
2	Scottie Pippen	595	511	193	93	1461
3	Horace Grant	659	178	95	69	1000
4	Bill Cartwright	486	126	32	15	760
5	John Paxson	91	297	62	3	710
6	B.J. Armstrong	149	301	70	4	720
7	Stacey King	208	65	24	42	419
8	Cliff Levingston	225	56	29	43	314
9	Will Perdue	336	47	23	57	307
10	Craig Hodges	42	97	34	2	362
11	Dennis Hopson	109	65	25	14	264
12	Scott Williams	98	16	12	13	127
		3490	**2212**	**822**	**438**	**9024**

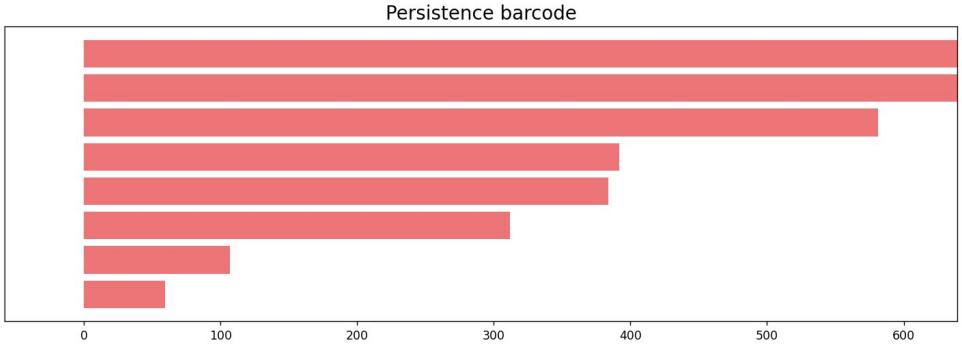
Rk		TRB	AST	STL	BLK	PTS
1	Chris Mullin	443	329	173	63	2107
2	Tim Hardaway	332	793	214	12	1881
3	Mitch Richmond	452	238	126	34	1840
4	Rod Higgins	354	113	52	37	776
5	Alton Lister	483	93	20	90	491
6	Tom Tolbert	275	76	35	38	500
7	Tyrone Hill	383	19	33	30	390
8	Šarūnas Marčiulionis	118	85	62	4	545
9	Jim Petersen	200	27	13	41	279
10	Kevin Pritchard	65	81	30	8	243
11	Mario Elie	109	44	19	10	231
12	Paul Mokeski	67	9	8	3	57
13	Steve Johnson	57	17	4	4	90
14	Larry Robinson	23	11	9	1	56
15	Les Jepsen	37	1	1	3	28
16	Vincent Askew	11	13	2	0	33
17	Mike Smrek	7	1	2	0	14
18	Bart Kofoed	3	4	0	0	3
		3419	**1954**	**803**	**378**	**9564**

1.2 Szczegółowa dokumentacja kodu

main.py

```
1   import matplotlib.pyplot as plt
2   import gudhi
3
4   ...
18
19  off_file = 'chicago91.off'
20  point_cloud = gudhi.read_points_from_off_file(off_file=off_file)
21
22  rips_complex = gudhi.RipsComplex(points=point_cloud, max_edge_length=700)
23  simplex_tree = rips_complex.create_simplex_tree(max_dimension=1)
24  diag = simplex_tree.persistence(min_persistence=2)
25
26  gudhi.plot_persistence_barcode(diag)
27  plt.show()
28
29
```

Output:



Zmienne:

- off-ile - zmienna z plikiem w tym przypadku chicago91.off
- point-coud - zmienna przechowująca chmurę punktów pobraną z pliku .off
- rips-complex - zmienna przechowująca wartośc z wykonania funkcji RipsComplex(). czyli dystans diagramu.
- simplex-tree - zmienna przechowuje wynik operacji creat-simplex-tree() wykonanej na obiekcie rips-complex (Maksymalna liczba wymiaru)
- diag - zmienna przechowująca wynik metody persistance() wykonanej na obiekcie simplex-tree. Minimalny rozmiar interwału trwałości.

Metody:

gudhi.read_points_from_off_file()

Read points from an **OFF file**.

Parameters:	**off_file** (*string*) – An OFF file style name.
Returns:	The point set.
Return type:	List[List[float]]

- **__init__()**

  SimplexTree constructor.

Parameters:	**other** (***SimplexTree*** *(Optional)*) – If *other* is *None* (default value), an empty *SimplexTree* is created. If *other* is a *SimplexTree*, the *SimplexTree* is constructed from a deep copy of *other*.
Returns:	An empty or a copy simplex tree.
Return type:	**SimplexTree**
Raises:	**TypeError** – In case *other* is neither *None*, nor a *SimplexTree*.
Note:	If the *SimplexTree* is a copy, the persistence information is not copied. If you need it in the clone, you have to call `compute_persistence()` on it even if you had already computed it in the original.

`persistence()`

This function computes and returns the persistence of the simplicial complex.

Parameters:	• **homology_coeff_field** (*int*) – The homology coefficient field. Must be a prime number. Default value is 11. Max is 46337. • **min_persistence** (*float*) – The minimum persistence value to take into account (strictly greater than min_persistence). Default value is 0.0. Set min_persistence to -1.0 to see all values. • **persistence_dim_max** (*bool*) – If true, the persistent homology for the maximal dimension in the complex is computed. If false, it is ignored. Default is false.
Returns:	The persistence of the simplicial complex.
Return type:	list of pairs(dimension, pair(birth, death))

•

`gudhi.plot_persistence_barcode(persistence=[], persistence_file='', alpha=0.6, max_intervals=20000, inf_delta=0.1, legend=False, colormap=None, axes=None, fontsize=16)` [source]

This function plots the persistence bar code from persistence values list , a np.array of shape (N x 2) (representing a diagram in a single homology dimension), or from a **persistence diagram** file.

Parameters:	• **persistence** (*an array of (dimension, array of (birth, death)) or an array of (birth, death).*) – Persistence intervals values list. Can be grouped by dimension or not. • **persistence_file** (*string*) – A **persistence diagram** file style name (reset persistence if both are set). • **alpha** (*float.*) – barcode transparency value (0.0 transparent through 1.0 opaque - default is 0.6). • **max_intervals** (*int.*) – maximal number of intervals to display. Selected intervals are those with the longest life time. Set it to 0 to see all. Default value is 20000. • **inf_delta** (*float.*) – Infinity is placed at `((max_death - min_birth) x inf_delta)` above `max_death` value. A reasonable value is between 0.05 and 0.5 - default is 0.1. • **legend** (*boolean.*) – Display the dimension color legend (default is False). • **colormap** (*tuple of colors (3-tuple of float between 0. and 1.).*) – A matplotlib-like qualitative colormaps. Default is None which means `matplotlib.cm.Set1.colors`. • **axes** (*matplotlib.axes.Axes*) – A matplotlib-like subplot axes. If None, the plot is drawn on a new set of axes. • **fontsize** (*int*) – Fontsize to use in axis.
Returns:	(*matplotlib.axes.Axes*): The axes on which the plot was drawn.

•

Przykłady użytych bibliotek:

Gudhi:

```
import matplotlib.pyplot as plot
import gudhi

__author__ = "Vincent Rouvreau"
__copyright__ = "Copyright (C) 2016 Inria"
__license__ = "MIT"

print("#####################################################################")
print("Show barcode persistence example")

persistence = [
    (2, (1.0, float("inf"))),
    (1, (1.4142135623730951, float("inf"))),
    (1, (1.4142135623730951, float("inf"))),
    (0, (0.0, float("inf"))),
    (0, (0.0, 1.0)),
    (0, (0.0, 1.0)),
    (0, (0.0, 1.0)),
]
gudhi.plot_persistence_barcode(persistence)
plot.show()

print("#####################################################################")
print("Show diagram persistence example")

gudhi.plot_persistence_diagram(persistence)
plot.show()

print("#####################################################################")
print("Show diagram persistence example with a confidence band")

gudhi.plot_persistence_diagram(persistence, band=0.2)
plot.show()

print("#####################################################################")
print("Show barcode and diagram persistence side by side example")
fig, axes = plot.subplots(nrows=1, ncols=2)
gudhi.plot_persistence_barcode(persistence, axes=axes[0])
gudhi.plot_persistence_diagram(persistence, axes=axes[1])
fig.suptitle("barcode versus diagram")
plot.show()
```

Matplotlib:

Example

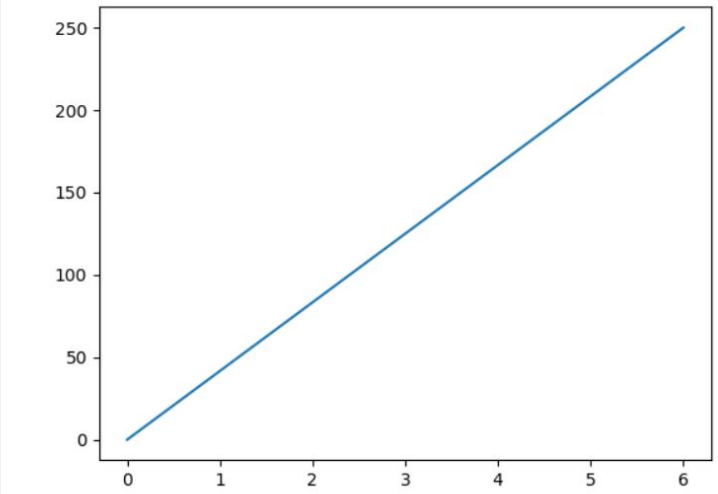
Draw a line in a diagram from position (0,0) to position (6,250):

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

Result:



1.3 Środowisko programistyczne

Wymagania:

Srodowisko PyCharm Community Edition 2020.2.3

- Platforam sprzętowa - Windows, macOS, Linux.
- System operacyjny - Windows 8/10/11 (64 bit), macOS 10.14 lub nowszy, dowolna dystrybucja Linuksa obsługująca Gnome, KDE lub Unity DE.
- Procesor - Wielordzeniowy procesor
- Pmięć RAM - min 4GB
- Miejsce na dysku twardym - 2,5GB i 1GB pamięci podręcznej
- Wielordzeniowy procesor.
- Minimum 4 GB.
- 2,5 GB i 1 GB na pamięć podreczną
- Ekran. 1024x768.

Instalacja bibliotek gudhi i matplotlib

- python -m pip install -U pip
- python -m pip install -U matplotlib
- python pip install gudhi

Literatura

[1] *Encyklopedia Zarządzania - Analiza Danych*

[2] *Zastosowanie Topologicznej Analizy Danych - Autorstwa: Artura Żuwała*

[3] *Moneyball 2.0: How Missile Tracking Cameras Are Remaking The NBA*

[4] *Analyzing NBA basketball data with R*

[5] *Game of Waveforms*

Książki

[6] ***Lutz Mark*** - *Python - Wprowadzenie*

[7] ***Vaughan Lee*** - *Python z życia wzięty. Rozwiązywanie Problemów za pomocą kilku linii kodu*

Kursy online

[8] ***Kanał o Wszystkim*** - *Kurs Python - Programowanie*

[9] ***Greg Hogg*** - *Complete Beginner's Tutorial to Google Colab*

Linki, które były pomocne w czasie realizacji pracy

[10] *https://gudhi.inria.fr/python/latest/simplex$_t$ree$_r$ef.html*
https://gudhi.inria.fr/python/latest/persistence$_g$raphical$_t$ools$_r$ef.html
https://gudhi.inria.fr/doc/3.4.1
https://www.w3schools.com/python/matplotlib$_p$yplot.asp
https://snyk.io/advisor/python/gudhi/functions/gudhi.plot$_p$ersistence$_b$arcode

Prace własne

[11] ***Piotr Maliga*** - *Metody-Badawcze-w-Informatyce-Piotr-Maliga*