עבודת גמר - למידה לא מפוקחת 88-780

מגיש: רותם גזית, ת"ז 318446697

סמסטר חורף 2020

תוכן העניינים

1 הקדמה

המידע אותו אני מנסה לנתח כאן הוא ניתוח הסרטים המופיעים בנטפליקס[1], כשהמטרה הכללית היא לחלק אותם בסופו של דבר לקבוצת סרטים ש"שווה לצפות בהם" וקבוצת סרטים ש"לא שווה לצפות בהם".

המידע מכיל בעבור כל סרט את שמו, שנת הפצתו, אורכו, הבמאים והשחקנים שלו, דירוג קבוצת הגיל המומלצת לצפייה, הקטגוריות אליו שויך הסרט בנטפליקס, המדינות בהן הופק וצולם, ותיאור הסרט במילים.

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
60011649	Movie	Indiana Jones and the Raiders of the Lost Ark	Steven Spielberg	Harrison Ford, Karen Allen, Paul	United States	January 1, 2019	1981	PG	116 min	Action & Adventure, Children & Family Movies, Classic Movies	When Indiana Jones is hired by the government to locate the legendary A
60010488	Movie	Indiana Jones and the Temple of Doom	Steven Spielberg	Harrison Ford, Kate Capshaw, A	United States	January 1, 2019	1984	PG	119 min	Action & Adventure, Children & Family Movies, Classic Movies	Indiana Jones, his young sidekick and a spoiled songbird get more than t
60010487	Movie	Indiana Jones and the Last Crusade	Steven Spielberg	Harrison Ford, Sean Connery, D	United States	January 1, 2019	1989	PG-13	127 min	Action & Adventure, Children & Family Movies, Classic Movies	Accompanied by his father, Indiana Jones sets off on his third adventure

את המידע הצמדתי לשני נתונים נוספים: מה הדירוג הממוצע (1 עד 10) שנתנו לו צופים באתר IMDB[2], ובכמה פרסי אוסקר זכה הסרט[3]. הקוד לתהליך ההצמדה נמצא בנתיב python_code/data_prep והוא מכיל מספר ניואנסים קטנים (למשל חלק מהמקומות השתמשו ב & במקום במילה and בשמות חלק מהסרטים). לאחר תהליך העיבוד הראשוני המידע נשמר בקובץ data/netflix/stage.json.

ביציאה לדרך אני מניח מספר הנחות עבודה על עולם הבעיה:

1. סביר להניח שישנם שחקנים טובים ושחקנים גרועים, וששחקן טוב ישחק בעיקר בסרטים טובים (וסרט טוב בעיקר יכיל שחקנים טובים).
   כלומר, טיבו של סרט יכול להיות קשור לסרט אחר לפי השחקנים שבו.
2. פרס אוסקר הוא מדד אבסלוטי לאיכות הסרט. הייתי רוצה למצוא קורולציה בין "קבוצת הסרטים ששווה לצפות בהם" והזכיות שלהם בפרסי אוסקר.
3. ישנם סינונים בסיסיים שאני יכול לעשות שימקדו אותי במציאת "קבוצת הסרטים ששווה לצפות בהם", למשל:
   (א) ככל הנראה שלא אכליל סרטים הודים בתוך הקבוצה הזו, היות וזה לא הטעם שאני מחפש בסרט טוב.
   (ב) דירוג נמוך מאוד לסרט ב IMDB הוא חיתוך פשוט שאפשר לעשות. למשל כל סרט שקיבל דירוג פחות מ 5, ישירות יעבור ל"קבוצת הסרטים שלא שווה לצפות בהם".

תכנית העבודה היא כזו:

1. לבצע סינונים נאיביים לפי הנחות העבודה שלי על המידע
2. ליצור גרף קשרים בין הסרטים, בהתבסס בעיקר על השחקנים המשותפים בניהם
3. למצוא קהילות בגרף במגוון שיטות: אלגוריתם Louvain, או לחילופין ביצוע MDS ולאחריו אלגוריתמים לזיהוי קהילות במרחב
4. לאחד את הקהילות שנמצא לשתי קבוצות - קבוצת "הסרטים הטובים" וקבוצת "הסרטים הלא טובים"
5. לבצע מבחנים לאיכות הקהילות שמצאנו - מבחנים סטטיסטיים למובהקות ההפרדה שיצרנו, ואל מול ה "Groudtruth" של סרטים טובים - שהוא מבחינתי זכייה בפרסי אוסקר.

הנדסת תכנה

הפרויקט נבנה כך שיהיה ניתן בקלות מאוד להחליף את שיטת הקליסטור או את שיטת בניית הגרף. לצורך כך קיימים שני Interfaces:

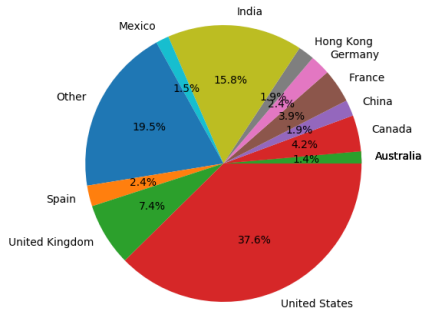
1. עבור יצירת הקשתות לגרף - IEdgeLogic
   מקבל את ה dataframe ומחזיר קשתות ממושקלות
2. עבור חלוקה לקהילות - ICluster
   מקבל את הגרף ומחזיר שיוך של כל node לקהילה

השיטה הזו מאפשרת לנו להחליף בסה"כ את המימוש הנבחר לאותו ממשק ב app.py כדי להחליף שיטת עבודה ולבדוק מיד את התוצאות. כל אחד מהמימושים יוסבר כאן ובנוסף מתועד כ docstring בקוד עצמו.

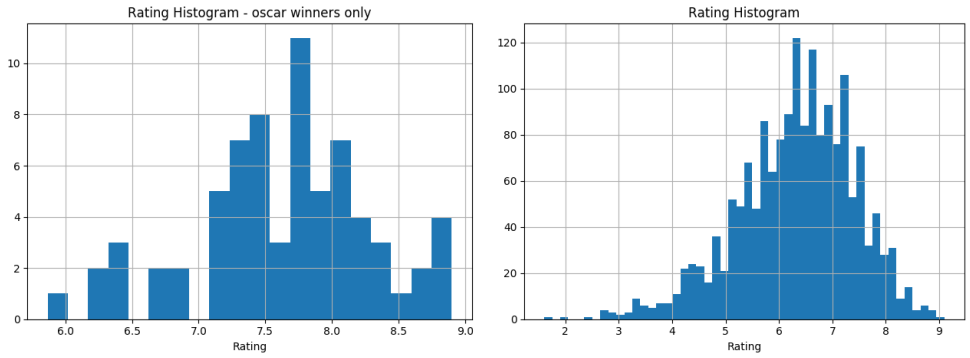
[1] https://www.kaggle.com/shivamb/netflix-shows/tasks?taskId=123
[2] https://www.kaggle.com/ashirwadsangwan/imdb-dataset
[3] https://www.kaggle.com/unanimad/the-oscar-award

2 ניתוח המידע

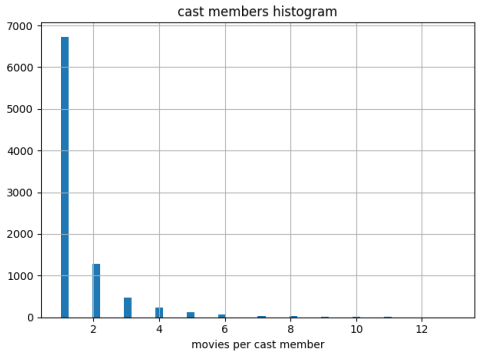
כלל התרשימים המוצגים נוצרים בקובץ python_code/learning/stats/data_analysis.py.
התרשים הבא מציג את הפילוח של הסרטים לפי מדינת המקור שלהן:



אני מסנן רק על סרטים אמריקאים ובריטים כי זו קבוצת העניין שלי. הסינון מתבצע ב python_code/learning/data_loader.py.
שני התרשימים הבאים מציגים את היסטוגרמת הדירוגים שניתנו לסרטים הללו בנטלפיקס, ואת היסטוגרמת הציונים של סרטים זוכי אוסקר.



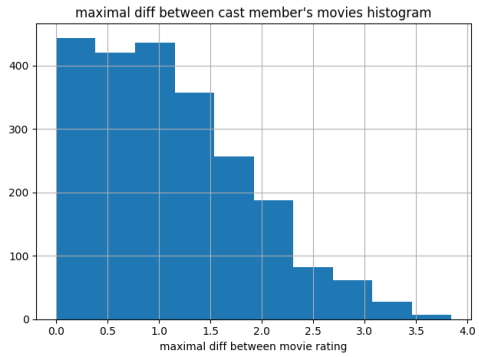
אפשר לראות מכך שני דברים מעניינים: דבר ראשון - סרט זוכה אוסקר מקבל תמיד ציון של מעל 5, אך אולי בניגוד למצופה - אין מתאם חד משמעי בין הציון לזכייה. כשמחשבים את מקדם מתאם פירסון בין שני הנתונים[4] מקבלים $R = 0.391695$, כלומר קשר ליניארי חיובי אך לא משמעותי מאוד.
כיוון שאין טעם להתעסק בסרטים עם דירוג נמוך מאוד, אנחנו מבצעים סינון על סרטים שקיבלו רק דירוג גדול מ 5 (מתבצע ב data_loader.py)
התרשים הבא הוא היסטוגרמה של כמות הסרטים שכל שחקן שיחק בהם. ניתן לראות שבאופן מובהק רוב השחקנים (במידע שלנו) הופיעו בסרט יחיד.



נבדוק את הנחת העבודה שלנו - "סביר להניח שישנם שחקנים טובים ושחקנים גרועים, וששחקן טוב ישחק בעיקר בסרטים טובים".
התרשים הבא הוא היסטוגרמה[5] של הפרשי הציונים בין הסרט הטוב ביותר לגרוע ביותר של כל שחקן (בעבור אלו עם יותר מסרט

[4] corrolation_between_raiting_and_oscars
[5] cast_member_rating_variance

אחד). כלומר - הפרש 0 אומר שכל הסרטים שלו קיבלו את אותו הציון, והפרש 10 אומר שהיה לו סרט מושלם וסרט נורא ואיום. הממוצע הוא 1.111, סטיית התקן היא 0.78, ואחוזון 95 עומד על 2.597. כלומר, המידע אכן מקיים במידת מה את ההנחה שלנו - ההפרשים בין הסרטים של שחקן מסוים אינם עולים על 30%.



3 קליסטור המידע

3.1 בניית גרף

3.1.1 יצירת הקשתות

המטרה שלנו אם כן היא לבנות תחילה את גרף הקשרים בין הסרטים. את הגרף תמיד נבנה באופן הבא: ניצור קשת בין שני סרטים אם יש להם שחקן משותף. כעת נשאלת השאלה מה יהיו משקלי הקשתות המחברות סרטים. לצורך כך נבדקו מספר גישות.

תהי $\{M_i\}_{i=1}^N$ קבוצת הסרטים השונים, ויהי $C_i = \{c_i^1, ..., c_i^k\}$ קבוצת השחקנים המשחקים בסרט i. יהי S_i הדירוג של הסרט i.

הגישה הפשוטה ביותר[6] אומרת את הדבר הבא: הציון לקשת בין הסרט M_i לסרט M_j יהיה ההפרש בין הדירוגים של הסרטים (במידה ויש להם שחקנים משותפים). כיוון שאנחנו רוצים שציון גדול יותר יעיד על קשר חזק יותר, וההפרש בין הציונים חסום (10 הוא הציון המקסימלי), נכתוב כך:

$$w_{i,j} = \begin{cases} 10 - |S_i - S_j| & C_i \cap C_j \neq \phi \\ 0 & \text{otherwise} \end{cases}$$

גישה נוספת [7] מגדירה את הציון בין הסרט M_i לסרט M_j ע"י:

$$w_{i,j} = |C_i \cap C_j|$$

הגישה הזו נתנה תוצאות טובות, אבל גישה בריאה יותר היא לתת ציון שונה לכל אחד מהשחקנים בסרט. הסיבה היא שמניתוח ה dataset עולה באופן מאוד ברור שרשימת השחקנים בסרט ממויינת, כלומר ככל שהשחקן מופיע מוקדם יותר ברשימה - כך חשיבותו בסרט עולה.

בנוסף, צריך להוסיף את במאי הסרט לרשימה (היות והוא לא חלק מצוות השחקנים). התוספת הזו מתבצעת ב data_loader.py.

לכן, הגרסה המתקדמת יותר[8] של המשקל הזה תהיה תוך הנחה שיש חשיבות לסדר בקבוצה C_i. נגדיר את המשקל החדש כך: תהי הפונקציה

$$C(i,c) = \begin{cases} j & \exists j : c_i^j = c \\ 0 & \text{otherwise} \end{cases}$$

כלומר $C(i,c)$ מחזירה את האינדקס של השחקן c בקבוצה C_i. אזי המשקל החדש יהיה

$$w_{i,j} = \sum_{c \in C_i \cap C_j} 1 + 2^{-C(i,c)} + 2^{-C(j,c)}$$

[6] python_code/learning/graph/edges_logics/rating_diff.py
[7] python_code/learning/graph/edges_logics/cast_count.py
[8] python_code/learning/graph/edges_logics/cast_priority_factor.py

כלומר בעבור שני סרטים $C_1 = \{a,b,c\}$ $C_2 = \{a,c\}$ אזי $w_{1,2} = \underbrace{1+1+1}_{a} + \underbrace{1+\frac{1}{4}+\frac{1}{2}}_{c} = 4.75$.

היוריסטיקה נוספת שנרצה להוסיף היא התייחסות לסוג הסרט[9]. סביר להניח שזה ששחקן x שיחק בשתי קומדיות יצביע על קשר גבוה יותר בין טיב הסרטים לעומת קומדיה מול דרמה. על כן אם $G_i = \{g_1, ..., g_t\}$ קבוצת הז'אנרים של הסרט i, נגדיר את המשקל החדש כך:

$$w'_{i,j} = w_{i,j} \cdot (|G_i \cap G_j| + 1)$$

3.1.2 מדידת איכות המשקלים

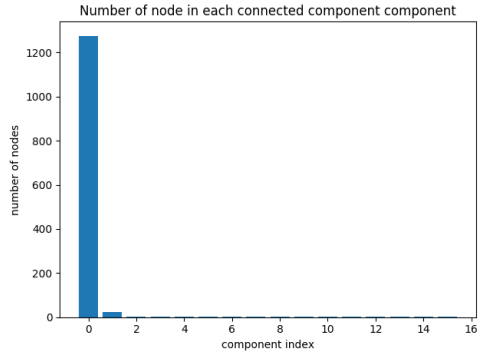
הדרך הטובה ביותר שיש לי כרגע כדי למדוד את איכות המשקלים של הקשתות ביחס לבעיה אותה אני מנסה לפתור, היא לחשב בעבור כל לוגיקה את מקדם המתאם של פירסון ביחס להפרש בין הציונים של הסרטים. כלומר, היינו רוצים לראות ש $R(|S_i - S_j|, w_{i,j}) < 0$, כלומר קיים יחס ליניארי שלילי בניהם (ככל שהציון לקשת גבוה יותר - כך ההפרש בין הדירוגים קטן יותר).
נאמר ששיטה אחת טובה יותר מהשנייה אם מקדם המתאם של הראשונה גדול בערכו המוחלט מזו של השנייה (קשר מובהק יותר).
צריך לזכור שזו היוריסטיקה בלבד (אם היה ברור שזה המשקל שאיתו נוכל לחלק את הגרף לשתי הקבוצות המדוברות - היינו נשארים עם הדרך הראשונה, שבה ברור שהמקדם הוא -1)
הטבלה המצורפת מציגה את תוצאות הניסוי, הקוד שיוצר אותה נמצא בpython_code/learning/stats/edge_logic_test.py.

לוגיקה	R
הפרש בין הדירוגים של סרטים עם שחקנים משותפים	-1
התבססות על כמות שחקנים משותפים	-0.039672
התבססות על כמות שחקנים משותפים לפי סדרם ברשימה	-0.042627
התבססות על כמות קטגוריות משותפות	-0.014543
שילוב לוגיקה 2 ולוגיקה 4	-0.06095
שילוב לוגיקה 3 ולוגיקה 4	-0.058427

אפשר לשים לב שהקשר ממש לא מובהק באף אחת מהלוגיקות, ואף הקשר של "שילוב לוגיקות 2 ו 4" נראה על פניו טוב יותר מהקשר של "שילוב לוגיקות 3 ו 4", אבל במבחן התוצאה דווקא הלוגיקה אחרונה הייתה המוצלחת ביותר מבניהן, ולכן זו הלוגיקה שאיתה נמשיך בפרויקט.
מכאן והלאה נבחן את החלוקה שלנו בראי שתי שיטות ליצירת הקשתות: שיטת ההפרש בין הציונים, ושיטת "שילוב לוגיקות 3 ו 4" (שלה נקרא מעתה "שיטת רשימת השחקנים").

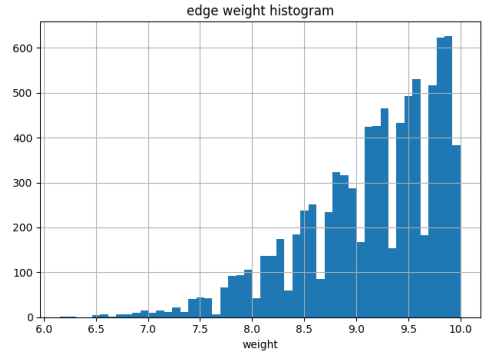
3.1.3 סטטיסטיקות על הגרף

התרשים הבא מראה את כמות הסרטים הנמצאים בכל אחד מרכיבי הקשירות בגרף. זה נתון חשוב מאוד, כי אם יש לנו הרבה מאוד רכיבי קשירות קטנים - אין לנו בעצם במה למצוא את האשכולות שלנו. נשים לב - אנחנו תמיד בונים את הגרף באותה הצורה, ולכן הגרף רלוונטי לשתי הלוגיקות.

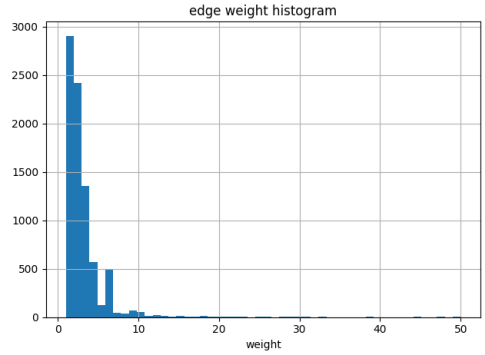


התרשימים הבאים מציגים היסטוגרמה של משקלי הקשתות בגרף. מימין - הלוגיקה מבוססת הפרש דירוגי הסרטים, משמאל - הלוגיקה המבוססת על שיטת רשימת השחקנים.

[9] python_code/learning/graph/edges_logics/category.py



איור 1: לוגיקת הפרש דירוגים



איור 2: לוגיקת רשימת השחקנים

3.2 חלוקת הגרף לאשכולות

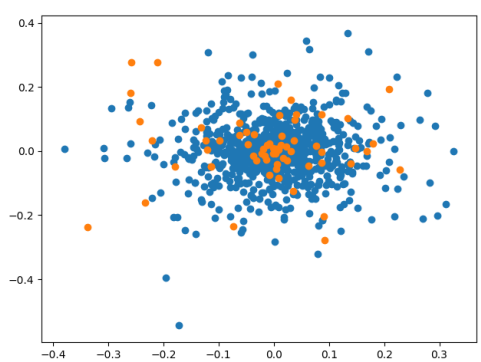
נבדוק מספר גישות למציאת אשכולות בגרף שיצרנו.

3.2.1 המרת הגרף לקאורדינטות

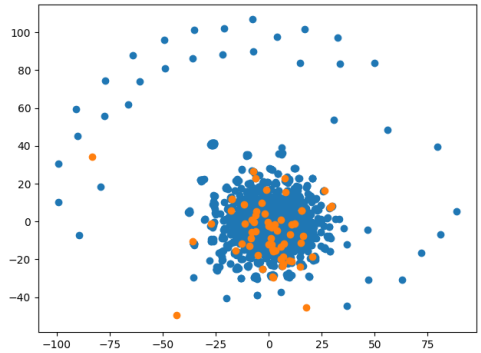
אלגוריתם MDS מאפשר לקחת את מטריצת המרחקים $d^2_{i,j}$ בין הקשתות בגרף, ולהפוך כל קודקוד i לנקודה $\bar{x}_i$ כך ש $d(x_i, x_j)^2 = d^2_{i,j}$.
לאחר הפיכת הגרף לנקודות במרחב, אפשר להפעיל אלגוריתמים למציאת אשכולות כמו Kmeans, GMM וכו'. המימוש קורה ב python_code/learning/clustering/mds_clustering.py.
צריך לשים לב שבין חלק מהסרטים אין לנו קשת בכלל (לחילופין - קיימת קשת במשקל 0), ואנחנו רוצים שככל שהקשת "כבדה" יותר - כך יצוג הנקודות יהיה קרוב יותר. אלגוריתם MDS כפי שממומש ב sklearn מתייחס ל0 בתור "קשת לא קיימת", ועל כן צריך לשמור שערכים אלו יהיו 0.

$$d^{\epsilon}_{i,j} = \begin{cases} \frac{1}{w_{i,j}} & w_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases}$$

ההמרה הזו מתבצעת לפני הרצת MDS על הגרף. לאחר הרצת ההמרה ל2 מימדים, קיבלנו את התמונה הבאה. בכתום - סרטים שהם זוכי אוסקר.



התמונה שקיבלנו בעצם מציגה שלא ניתן להגיע לאישכול משמעותי (ביחס לזוכי האוסקר) על גבי ההמרה הזו.
ננסה להחליף את האלגוריתם של MDS למשהו אחר: ננסה להפעיל את אלגוריתם TSNE. גם כאן התמונה לא טובה.



תמונות דומות התקבלו גם כשהחלפתי את הקשתות בגרף ל"שיטת רשימת השחקנים", וגם כשניסיתי להגדיל את מספר המימדים. כלומר נראה שהדרך לפתרון לא תוכל להעזר בהמרת הגרף לקאורדינטות.

3.2.2 אלגוריתמים גרפיים

3.2.3 אלגוריתם Louvain

שיטה נוספת לחלוקת הגרף לאשכולות היא תוך שימוש באלגוריתמים גרפים.

אלגוריתם Louvain הוא אלגוריתם שפותח בשנים האחרונות, ומתבסס על ה loss function הבא (אליו הוא מחפש מינימום בצורה חמדנית):

$$L = -\frac{1}{2m}\left[\sum_{i,j}(w_{i,j} - \frac{\gamma k_i k_j}{2m})c(i,j)\right]$$

$$c(i,j) = \begin{cases} 1 & i,j \quad \text{in\_same\_community} \\ 0 & \text{otherwise} \end{cases}$$

$$k_i = \sum_j w_{i,j}$$

$$m = \sum_i k_i$$

ה loss function כפי שהצגנו בכיתה, מייצג את הציפיה שלנו מ cluster מוצלח.

הפרמטר $\gamma \in [0,\infty]$ מאפשר לקבוע את גודל האשכולות, כאשר $\gamma = 0$ כולם באותו אשכול, $\gamma = \infty$ כולם באשכולות יחידים.

הפעלת האלגוריתם מתבצעת ב python_code/learning/clustering/louvain.py.

לאחר הפעלת האלגוריתם וקבלת קהילות, נרצה לחשב בעבור כל אחת מספר נתונים: כמה סרטים יש באשכול, מה ממוצע הדירוגים של הסרטים באשכול[10], ומה ה precision וה recall אל מול ה groundtruth - סרטים זוכי אוסקר[11]. הטבלה המוצגת כאן מציגה את תוצאות ההרצה בעבור $\gamma = 1$ (על בסיס גרף הבנוי ב"שיטת הפרש דירוגים").

community	title count	win count	averageRating	stdRating	max_rating	min_rating
0	174	2	6.2008	0.7462	8.5000	5.0000
1	128	12	6.8749	0.7954	8.7999	5.0000
2	117	17	6.8066	0.8529	8.9000	5.0000
3	112	2	6.4519	0.7905	8.5000	5.0000
4	108	5	6.4751	0.7817	8.5000	5.1000
5	99	4	6.5523	0.8577	8.2000	5.0000
6	89	2	6.5880	0.7962	8.9026	5.2000
7	75	0	6.4893	0.7608	8.1000	5.1000
8	69	6	6.5837	0.9712	8.9000	5.0577
9	68	4	6.4569	0.7092	8.0000	5.0202
10	67	3	6.4464	1.0104	8.4994	5.0000
11	64	0	6.9064	0.7736	8.5000	5.4000
12	62	4	6.4503	0.8441	8.0000	5.0000
13	30	1	6.7290	0.6338	8.0000	5.5000
14	27	2	6.2964	0.7801	8.1000	5.3000
15	24	0	6.4297	0.8141	8.0000	5.1000
16	4	0	7.2443	0.4341	7.7772	6.8000
17	3	0	8.6000	0.1000	8.7000	8.5000
18	2	1	8.1721	0.6052	8.6000	7.7442
19	2	0	7.4000	0.8485	8.0000	6.8000
20	2	0	7.2500	0.2121	7.4000	7.1000
21	2	0	7.3500	0.4950	7.7000	7.0000
22	2	0	6.4087	0.0123	6.4173	6.4000
23	2	0	6.6233	1.0228	7.3465	5.9000
24	2	0	6.4500	0.4950	6.8000	6.1000
25	2	0	6.8642	0.3736	7.1283	6.6000
26	2	0	5.9000	1.2728	6.8000	5.0000
27	2	0	7.2000	0.0000	7.2000	7.2000
28	2	0	6.1000	1.1314	6.9000	5.3000
29	2	0	6.3500	0.0707	6.4000	6.3000

[10] community_stats

[11] precision_recall

אבחנה בין האשכולות השונים

כעת על מנת לייצר מה clusters את שתי הקבוצות המבוקשות, צריך למצוא קריטריון לאבחנה בין הקבוצות. הרצתי את האלגוריתם מספר פעמים, בכל פעם עם γ שונה, ובדקתי מהו הפרמטר עם הקורולציה הגדולה ביותר לכמות הסרטים זוכי האוסקר ב cluster. ה"סרט בעל הדירוג המקסימלי" בכל אשכול נראה כמו הבחירה הטובה ביותר.

gamma	min_raiting	avg_rating	std_rating	max_rating	size	best corr
0.1	-0.348	-0.125	0.187	0.473	1.000	size
1	-0.435	-0.090	0.289	0.535	0.635	size
10	0.040	0.271	0.255	0.409	0.126	**max_rating**
15	0.057	0.331	0.234	0.389	0.070	**max_rating**
20	0.134	0.393	0.189	0.411	0.120	**max_rating**
25	0.133	0.321	0.136	0.363	0.101	**max_rating**
30	0.173	0.355	0.135	0.378	0.097	**max_rating**
35	0.142	0.335	0.172	0.371	0.135	**max_rating**
40	0.114	0.331	0.162	0.379	0.156	**max_rating**
45	0.129	0.335	0.176	0.384	0.145	**max_rating**
50	0.228	0.382	0.116	0.384	0.087	**max_rating**
60	0.232	0.373	0.096	0.381	0.101	**max_rating**
70	0.228	0.343	0.103	0.358	0.063	**max_rating**
80	0.221	0.342	0.121	0.357	0.065	**max_rating**
90	0.273	0.353	0.103	0.345	0.023	avg_rating
100	0.299	0.359	0.079	0.343	-0.018	avg_rating

את החלוקה לשתי קבוצות נעשה באופן הבא: נבחר ערך r. אשכול i שבו הדירוג לסרט הטוב ביותר הוא r_i יהיה בקטגוריה 1 אם $r_i \geq r$, אחרת יהיה בקטגוריה 0.
כדי למצוא את הערך r האופטימלי, נריץ שוב את התהליך, ובעבור כל אחת נחפש את ה max rating שיתן לנו את ה precision הגדול ביותר (כל עוד ה recall נותר מעל 50%)[12]. כיוון שיש לנו מספר סופי של אשכולות בכל פעם, מספיק לרוץ עליהם בצורה ממויינת לפי הדירוג המקסימלי בכל אשכול, בסדר יורד. במילים אחרות:
אם $\{M_i\}$ כלל הסרטים, S_i דעירוג הסרט ה i, $\{I_i\}_{i=1}^{k}$ קבוצות האינדקסים (זרות בזוגות) המייצגים אשכול - כלומר M_i, M_j באותו אשכול $i,j \in I_s \Leftrightarrow s$. נגדיר $m_i = \max_{j \in I_i} S_j$. נמיין את I_i כך ש: $m_i \geq m_j$ לכל $i < j$. כעת עלינו לפתור את:

$$\max_{1 \leq t \leq k} Precision(\{I_1, ..., I_t\})$$
$$S.T \quad Recall(\{I_1, ..., I_t\}) \geq 0.5$$

שזה כמובן ניתן לפתרון ע"י ריצה על $1 \leq t \leq k$ האפשריים.
לפי התוצאות מוצגות כאן (על בסיס גרף הבנוי ב"שיטת הפרש דירוגים"):

gamma	max_rating	Cluster size	Cluster wins	Cluster precision	Cluster recall
0.1	8.600	1282	65	0.051	1.000
1	8.600	408	38	0.093	0.585
10	8.000	407	36	0.088	0.554
15	8.000	368	35	0.095	0.538
20	7.960	389	36	0.093	0.554
25	8.000	333	33	0.099	0.508
30	8.000	308	33	0.107	0.508
35	7.900	315	38	0.121	0.585
40	7.900	289	39	0.135	0.600
45	7.900	271	37	0.137	0.569
50	7.900	260	36	0.138	0.554
60	7.900	241	35	0.145	0.538
70	7.801	253	33	0.130	0.508
80	7.801	236	33	0.140	0.508
90	7.645	306	41	0.134	0.631
100	7.800	214	33	0.154	0.508

נראה שלקחת $r = 7.9$ (הממוצע שלהם) יתן לנו תוצאות מיטביות.

3.2.4 שיטות נוספות

במעלה הדרך, נבדקו שיטות גרפיות נוספות לחלוקת הגרף, כולן לא הוציאו תוצאות טובות במיוחד:

- **אשכול היררכי**[13] - נתן לכל היותר precision 5%, recall 76%, כלומר יצר אשכול גדול עם דיוק נמוך
- **אשכול ספקטרלי** - נתן תוצאה דומה, גם ממנה לא היה ניתן להפיק ערך.

[12] find_best_cut_by_max_rating
[13] python_code/learning/clustering/hierarchy_clustering.py

3.3 מדידת איכות התוצר

בסופו של דבר החלוקה לאשכולות מתבצעת בדרך הבאה:

- בניית גרף לפי לוגיקת קשתות מסוימת
- הרצת אלגוריתם Louvain
- כל אשכול המכיל סרט שדירוגו גבוה מ 7.9 נכנס לקטגוריה 1, אחרת נכנס לקטגוריה 0

לאחר אופטימיזצית hyper parameters, בחרתי $\gamma = 60$. מוצגות כאן התוצאות על החלוקה לקבוצות 0, 1 באופן הטוב ביותר לשתי השיטות:

group	titles	wins	max rating	min rating	avg rating	precision	recall
0	1043	29	7.900	5.000	6.472	2.780	44.615
1	277	36	8.903	5.000	7.345	12.996	55.385

איור 3: תוצאות אשכול - לוגיקת רשימת שחקנים

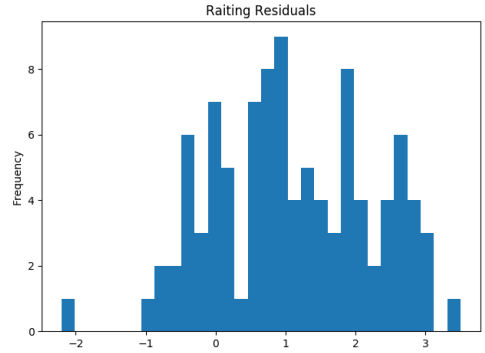
group	titles	wins	max rating	min rating	avg rating	precision	recall
0	1098	30	7.900	5.000	6.443	2.732	46.154
1	246	35	8.903	5.020	7.488	14.228	53.846

איור 4: תוצאות אשכול - לוגיקת הפרש דירוגים

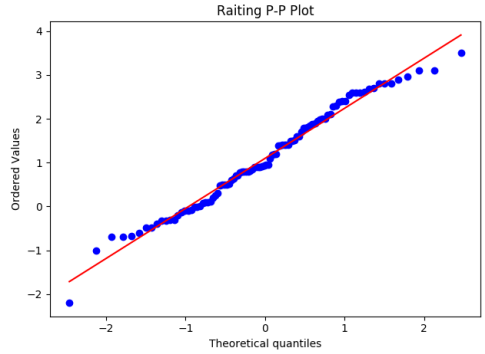
כעת לאחר קבלת התוצאות, נרצה להבין כמה ההפרדה בין הקבוצות הנ"ל (בכל אחת מהלוגיקות) היא מובהקת (significant) מבחינה סטטיסטית. החלק הזה קורה ב python_code/learning/stats/hypothesis_test.py. נבחן את טיב ההפרדה בניהם לפי הדירוג של הסרטים שהם מכילים. נרצה לעשות זאת בעזרת מבחן t.
מבחן t דורש מאיתנו שני תנאי קדם: שונות שווה בין הקבוצות, ושההפרש בין הקבוצות יתפלג נורמלית. לפני שנוכל להפעיל את מבחן t, עלינו תחילה להפעיל מבחנים נוספים שיעזרו לנו לאשש את הנחות היסוד האלו. את כל התהליך אנחנו מריצים על $N = 100$ רשומות רנדומיות מכל אחת מהקבוצות[14].
יהיו $\{S_i^1\}_{i=1}^N$, $\{S_i^0\}_{i=1}^N$ שתי הסדרות של המ"מ המייצגים את הציון של הסרטים מקבוצות 0,1. השערת האפס שלנו היא ששתיהן מגיעות מאותה התפלגות.

בעבור לוגיקת הפרש הדירוגים:

כדי לבחון תחילה שהשונות שווה בין הקבוצות, נפעיל את מבחן Levene. לאחר ההפעלה נקבל $p = 0.335 > 0.05$, כלומר ה pvalue איננו מובהק ולכן ההנחה שלנו מחזיקה.
כעת נסתכל על סדרת המ"מ $\{D_i\}_{i=1}^N$ כאשר $D_i = S_i^0 - S_i^1$. נרצה לבדוק את הטענה שהמ"מ הנ"ל מתפלג נורמלית. התרשימים הבאים מציגים היסטוגרמה של המ"מ וגרף p-p. היינו מצפים מההיסטוגרמה להיראות כמו פעמון, ומגרף p-p להיצמד לקו הישר.



איור 5: היסטוגרמה - D_i



איור 6: גרף p-p - D_i

נפעיל את מבחן שפירו על $\{D_i\}$ ונקבל $p = 0.21378 > 0.05$, כלומר גם הוא לא מחזיר pvalue שהוא significat ועל כן גם ההנחה הזו מחזיקה.
לאחר שבדקנו את שתי ההנחות ניתן להפעיל את מבחן t והוא מחזיר לנו $p = 9.60642 \cdot 10^{-18} < 0.05$, כלומר באופן מובהק ניתן להגיד שהנחת האפס שלנו נדחית וההפרדה שיצרנו בין שתי הקבוצות הינה משמעותית.

[14] split_to_groups

בעבור לוגיקת רשימת השחקנים:

נרצה להפעיל את אותו התהליך. מבחן שפירו על סדרת ההפרשים החזיר $p = 0.3652 > 0.05$, אבל מבחן Levene החזיר $p = 1.7011 \cdot 10^{-5} < 0.05$ ולכן השערת האפס שלו (שהשונויות בין הקבוצות שוות) נדחית, כלומר לא ניתן להפעיל את מבחן t.

במקרה הזה נוכל להפעיל את Welch's t-test - הוא איננו מניח שהשונות של הסדרות שווה ולכן יהיה יעיל בהתמודדות עם המצב הזה. כדי לעבור בין ttest ל welch's בפייתון צריך בסה"כ להעביר פרמטר שונה לפונקציה scipy.stats.ttest_ind.
לאחר הפעלתו נקבל $p = 5.7311 \cdot 10^{-7} < 0.05$ כלומר גם כאן השערת האפס של המבחן נדחית וניתן להגיד שההפרדה בין הקבוצות הינה משמעותית.

4 סיכום

לפי ה dataset הנתון, קיימים ב netflix 4283 סרטים שונים. בסופו של התהליך המוצג כאן, המתבסס על הצמדת הסרטים לדירוגיהם באתר IMDB, החלת היוריסטיקות והפעלת אלגוריתם Louvain, קיבלנו קבוצת סרטים בגודל 246 סרטים (5.7% מהכמות המקורית), המכילים 54% מכלל הסרטים זוכי האוסקר בנטפליקס, ודירוגיהם שונים מהותית מדירוגי יתר הסרטים, וזאת בהתבססות על שחקנים משותפים בין סרטים בלבד.

ביציאה לדרך, הנחתי שככל הנראה ישנה קורולציה בין אוכלוסיית השחקנים בסרטים לזכייה באוסקר, בתור אינדיקטור ליצירת "קבוצת הסרטים ששווה לצפות בהם". לאחר הפעלת התהליך, קיבלנו הפרדה משמעותית על גרף הקשרים בין הסרטים המבוססים על שחקניהם, כך שבסופו של דבר קיבלנו קבוצת סרטים המכילה את מרבית הסרטים זוכי האוסקר.