**096235 - מערכות נבונות אינטראקטיביות**

דו"ח סיכום פרויקט: מימוש אלגוריתם  
 " DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation"

מגישים: יעל קיסלמן ועודד סלטון

**הקדמה**

מוזיקה מלווה אותנו מאז שחר ההיסטוריה, המוזיקה מהווה אחת מדרכי הביטוי של האדם כך שלאורך השנים נכתבו שירים שמבטאים כל רגש שחווה האדם ומעוצבים כך שהשירים נותנים לאדם להתחבר אליהם ולהנות מההקשבה למוזיקה. במשך השנים התפתחו המון סוגי מוזיקה שונים ומגוונים אשר מכוונים לטעמם של המאזינים השונים.

במרוצת השנים האחרונות ועם התפתחות הטכנולוגיה, התפתחו מערכות המלצה שונות על מנת להקל על המשתמש בבחירת השירים שהינו שומע ובצירת רשימות השמעה שונות. מערכות ההמלצה הפכו להיות קריטיות וכיום הן משפיעות על חווית המשתמש לא פחות מהמוזיקה עצמה ושירותי המוזיקה הגדולים כמו Youtube, Spotify משקיעים מאמצים רבים בשדרוג מערכות ההמלצה שלהן.

הפרויקט שלנו עוסק במודל המלצה מבוסס למידת חיזוק , בשם DJ\_MC שפותח במאמר הבא : ["DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation"](https://dl.acm.org/doi/pdf/10.5555/2772879.2772954) ועוסק במידול הבעיה של המלצת השירים כתהליך החלטה מרקובית שדרכו ניתן להשיג רשימת שירים מותאמת למשתמש. במאמר לומדים לא רק את ערך השיר למשתמש אלא שמים חשיבות גם לסוגי המעברים בין השירים שהמשתמש עושה.

לפרויקט שלנו מספר אבני דרך :

• מציאת דאטה מתאים לפרויקט - השתמשנו בדאטה של ספוטיפיי שמכיל את השירים הכי מושמעים בכל שנה בין השנים 2010 ל 2019 - לחץ כאן

• מימוש האלגוריתם כפי שקיים במאמר כך שהפלט של האלגוריתם הוא השיר הבא שהמערכת ממליצה למשתמש.

• בדיקת האלגוריתם שלנו אל מול האלגוריתם החמדן.

• ניתוח התוצאות והסקת המסקנות.

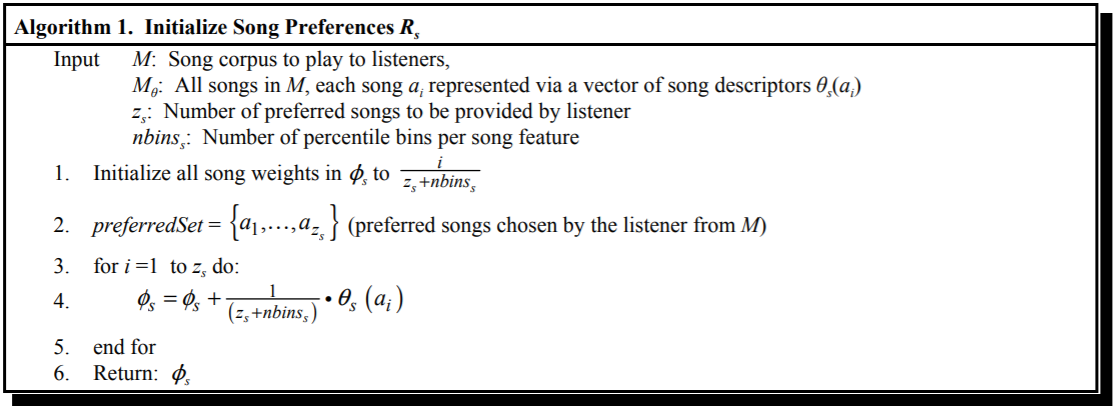
**סקירת ספרות**

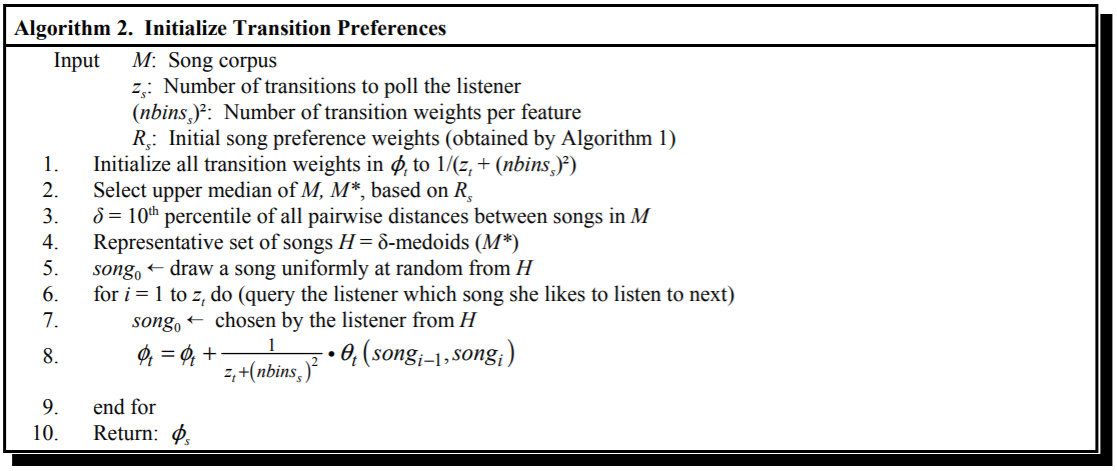
* Chen, S., Moore, J. L., Turnbull, D., and Joachims, T. 2012. “Playlist Prediction via Metric Embedding,” in Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York: ACM, pp. 714-722.  
  מאמר זה עוסק גם הוא על חיזוי פלייליסטים בעזרת הטבעה מטרית ע"י שימוש במודל מרקובי במרחב סמוי. ממאמר זה נלקח על הדאטה לאימון האלגוריתם.
* X. Wang, Y. Wang, D. Hsu, and Y. Wang. Exploration in interactive personalized music recommendation: A reinforcement learning approach. arXiv preprint arXiv:1311.6355, 2013  
  במאמר זה משתמשים גם בלמידה באמצעות חיזוקים לצורך יצירת מערכת המלצות לשירים וממדלים את בעית ההמלצה כבעית multi-armed bandit.
* Palmer, C. 2005. “Sequence Memory in Music Performance,” Current Directions in Psychological Science (14:5), pp. 247-250  
  במאמר זה מאששים את ההנחה כי קיימת חשיבות לסדר נגינת השירים עבור המאזין וכי ההנאה ממוזיקה מושפעת מסדר השירים.
* A. Berenzweig, B. Logan, D. P. Ellis, and B. Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. Computer Music Journal, 28(2):63–76, 2004.  
  מאמר זה עוסק בדימיון בין שירים ודרכים ללמדידתו. אומנם המאמר מביא כלים למדידת דימיון בין שירים- דבר שיכול להיות שימושי בבנית מערכ' המלצה אך הקישור היחיד בין המאמר שלנו לזה הוא בדאטה. במאמר, החוקרים טענו כי קיים דימיון מסוים בין שירים ברשימות השמעה שנוצרו ע"י משתמשים ולכן השתמשו בדאטה מאתר The Art of the Mix ולקחו עשרות אלפי רשימות האזנה שנוצרו ידנית.  
  האלג' שלנו משתמש בדאטה הנ"ל בכדי לסמלץ מודלים של מאזינים ע"מ לבדוק את האלג' DJ-MC.
* E. Zheleva, J. Guiver, E. Mendes Rodrigues, and N. Mili´c-Frayling. Statistical models of music-listening sessions in social media. In Proceedings of the 19th international conference on World wide web, pages 1019–1028. ACM, 2010.  
  במאמר זה מוצג מודל סטטיסטי אשר מניב טעם מוזיקלי של משתמש דרך היסטורית האזנות ומזהה קבוצות שירים המזוהות עם הטעם הנ"ל וקבוצות מאזינים בעלי טעם מוזיקלי דומה.
* R. S. Sutton and A. G. Barto. Introduction to Reinforcement Learning. MIT Press, Cambridge, MA, USA, 1st edition, 1998.  
  ספר על למידת חיזוק – המודל למידה עליו מבוסס האלג' במאמר שלנו
* Hao Liu ; Jun Hu ; Matthias Rauterberg. H. Liu, J. Hu and M. Rauterberg, "Music Playlist Recommendation Based on User Heartbeat and Music Preference," 2009 International

במאמר , מנסים לקשר את מערכת ההמלצה לפרמטרים פיזיים של המשתמש – heart beat - כך שהמטרה היא לשים את המוזיקה בצורה מותאמת לכך שתשמור את קצב דפיקות הלב של המשתמש נורמלי , במצבים בהם הקצב גבוה המשתמש יקבל רשימת שירים ייחודית שתותאם אליו ותחזיר במהרה את דפיקות הלב למצב של רוגע .   
מידול הבעיה כאן כמעט זהה למידול הבעיה שלנו שכן גם כאן משתמשים ב MDP למידול הבעיה .

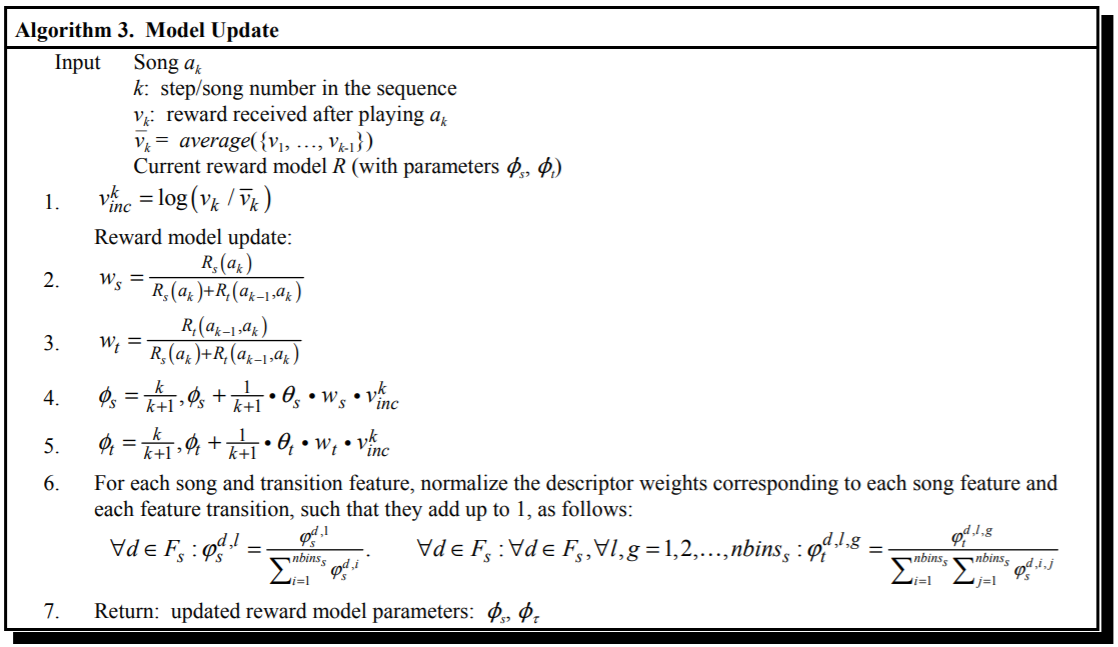
* Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions  
  המאמר מתאר את הטכנולוגיות שקיימות נכון ליום בו יצא , את נושא מערכות ההמלצה , המאמר סוקר את המערכות ומקטרג אותן לשלוש קטגוריות שונות : מבוססות תוכן , שיתופיות והיברידיות . כמו כן המאמר סוקר את הבעיות שקיימות כיום בתחום .
* Hirai, Tatsunori, Hironori Doi, and Shigeo Morishima. "MusicMixer: computer-aided DJ system based on an automatic song mixing." Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology. 2015.   
  המאמר מציע סוכן דמה, הנקרא music mixer , הסוכן הפעם אמור לעזור לדי ג'יים באירועים, כך שכעת הוא מציע אפשרות לעשות חיתוכים בתוך שירים בדיוק בזמן המתאים מתוך הבנה שהנושא קריטי במוזיקה בדגש על מסיבות ומאפשר יצירה של המיקס המושלם.
* Liebman, Elad, Benny Chor, and Peter Stone. "Representative selection in nonmetric datasets." Applied Artificial Intelligence 29.8 (2015): 807-838.  
  המאמר מדבר על אלגוריתם שמימשנו באלגוריתם 2 למציאת קרבה בין השירים השונים.

**מתודולוגיה** האלגוריתם אשר מימשנו משתמש במודל הלמידה באמצעות חיזוקים (RL) אשר מבוסס על תהליך למידה מרקובי (MDP).  
אנו מסתמכים על העבר (רשימת השירים שהושמעו) על מנת ליצור רצף שירים אופטימלי למשתמש תוך כדי לכידת העדפותיו הן מהשירים עצמם והן מהמעברים בין השירים.  
אנו מגדירים את עולם המצבים S כרשימת השירים שהושמעו עד כה, סט הפעולות A כבחירת השיר הבא ברשימה, פונקצית מעבר דטרמ' P (1 או 0) ופונקצית הרווח R – ההנאה של המשתמש משמיעת שיר a במצב s.  
את פונקצית הרווח של המשתמש אנו מחלקים ל-2 : רווח משיר Rs ורווח ממעבר בין שירים Rt ומעדכנים אותה בכל פעם שהמשתמש מזין פידבק לשיר שהושמע לו (כאשר הפידבק הוא 1 או 0 – like או unlike בהתאמה).

* **דאטה**השתמשנו בקובץ המכיל את כ-600 מהשירים הכי מושמעים ב - Spotify בעשור האחרון "Top Spotify songs from 2010-2019" כאשר כל שיר מאופין ב10 פיצ'רים שונים:  
  BPM, Energy, Danceability, Loudness, Liveness, Valence, Length, Acousticness, Speechiness, Popularity.  
  כימתנו כל מאפיין לבינים של 10 אחוז (10-percentile bins) ובכך למעשה יצרנו וקטור מאפיינים בינארי θs בגודל 100 (כאשר כל תת-וקטור בגודל 10 שייך למאפיין שונה) שבו יש "1" במקום המתאים לאחוזון בו נופל השיר לכל מאפיין (סה"כ בכל תת-וקטור של 10 בינים יש אחד "1" ותשע "0" – סה"כ סכום האינדיקטורים הוא 10).
* **אלגוריתם 1 – אתחול העדפות שירים Rs**המשתמש בוחר K שירים מועדפים מתוך רשימת שירים M ואנו מאתחלים את וקטור המשקולות φs.
* **אלגוריתם 2 – אתחול העדפות שירים Rt**  
  המערכת מקבלת את רשימת השירים M, וקטור המשקלים מאלגוריתם 1 מספר המעברים האפשריים שכן בדוגמא שלנו אם לכל פיצ'ר יש עשרה אחוזונים, אזי קיימים 100 מעברים אפשריים בין אחוזונים בין שני שירים עוקבים ועשינו אתחול.  
  חשוב לציין שבמימוש האלגוריתם שלנו הנחנו כי מעבר בין שני אחוזונים בכל פיצ'ר זהה ולדוגמא המשקל של מעבר בין אחוזון חמש לשש בפיצ'ר טמפו זהה למעבר בפיצ'ר לאוודנס.

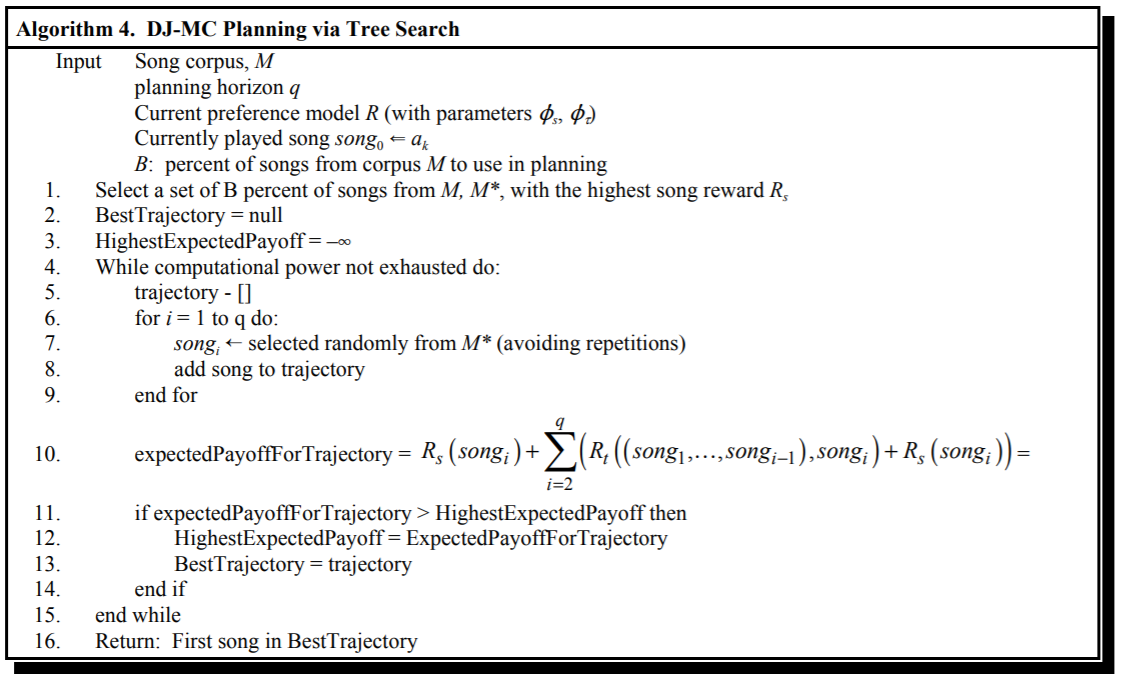
מכיוון שבמאמר מדברים על סוגי שינויים בין שירים שהמשתמש אוהב לדוגמא יש משתמשים שמעדיפים רשימת שירים של שירים מאותו ז'אנר וישנם משתמשים שמעדיפים שירים מז'אנרים שונים פיצלנו את רשימת השירים לקלאסטרים כל שכל קלאסטר מהווה שירים שנמצאים באותו מרחב השמעה (המשמעות היא שמבחינת האלגוריתם בחירה בין שירים מאותו קלאסטר זהים למשתמש שמעדיף רשימת שירים מאותו ז'אנר ) על מנת למצוא את הקלאסטר מימשנו את אלגוריתם ה כפי שמתואר במאמר מספר עשר (lieberman,shor,stone ,2015).   
האלגוריתם מחזיר את H שהוא סט השירים המייצגים של כל מעבר אפשרי וככל שגודלה של דלתה משתנה כך הסט יותר קטן על מנת לא ליצור יותר מדי סטים אבל גם לא פחות מדי הגדרנו את דלתא להיות שווה 24 אחוזונים.  
לאחר מכן, סימלצנו בחירה של המשתמש רשימה באורך עשרה שירים מתוך הקבוצה H (עם חזרות) והתאמנו את מטריצת המשקלים להעדפת האחוזונים של המשתמש.

* **אלגוריתם 3 – עדכון המודל**

**בשלב זה נכנס האלמנט של RL , המשתמש נותן את הפידבק על השיר שהושמע לו עד עכשיו ולאחר מכן המודל מעדכן את וקטורי המשקלים כך שהעדכון קורה בצורה הבא: המערכת מקבלת את הפידבק מחשבת את שמוגדר להיות לוג ערך הפידבק של השיר חלקי ממוצע הפידבקים ולאחר מכן מחשבת את המשקלים שמהווים הקשר בין הערך של המשתמש משיר לערך שלו למעבר בין שירים. בשלב הבא חישבנו את עדכון המשקלים כך שהוא מתחשב במספר השיר (ככל ש k יותר גדול כך הפידבק על השיר האחרון פחות משפיע) ומנרמלים.**  
לאחר קבלת משוב מהמשתמש על השיר שמתנגן (ניזכר כי הקלט הוא 1 או 0) אנו מעדכנים את וקטורי המשקלים של השירים והמעברים בהתאם.

* **אלגוריתם 4 – תכנון באמצעות עץ חיפוש**

ראשית נציין שבעיה של בחירת תת קבוצה מכל תתי הקבוצות באורך מסויים עם חשיבות לסדר היא בעיה קשה ולכן על מנת להיות יעילים יותר בבחירת תתי הקבוצות השתמשנו תחילה בבחירה של 50% מהשירים בעלי הערך הכי גבוה ברשימה M (צמצום הרשימה) לאחר מכן הגבלנו את מספר הפרמוטציות שנבדוק על מנת להתחשב בזמן ריצה (בכל ניסוי השתמשנו במספר פרמוטציות אחר אבל תחילה בחרנו 100 פרמוטציות ואז עברנו לאלף – מתוך 50 שירים בחירה של עשר מכילה כ (כך שבטרייד אוף בין הסתכלות על כל הפרמוטציות לבחירת האפשרות הטובה ביותר היינו צריכים לשלם מחיר מסויים)

שנית האלגוריתם הגריל בכל פעם פרמוטציה וחישב את הערך שלה למשתמש החל מהשיר הבא ועד השיר האחרון לאחר מכן בחרנו את הפרמוטציה שהניבה לנו את הערך הכי גבוה ובהתאם לכך נבחר השיר הבא שהמשתמש ישמע.

**מערכת ההמלצה -**   
מערכת הממלצה היא בעלת ממשק לוקאלי כך שע"י שימוש בGUI של פייתון יצרנו ממשק שבוא המשתמש יכול להחליט האם הוא אוהב או לא אוהב את השיר. למשתמש מופיע שם השיר והאם הוא אוהב אותו או לא , התגובה נרשמת במשתנים המיועדים בפייתון וכך ממשיכה ריצת האלגוריתם הלאה להמלצת השיר הבא זאת עד שתסתיים הרשימה.  


**תוצאות**בשל מגבלות מסגרת העבודה שלנו, האלג' שונה מעט שונה מהאלגוריתם המקורי, בפרט בכמות הקומבינציות האפשריות שאנו מריצים באלגוריתם 4. החלטנו לבדוק על משתמשים אנושיים את האלגוריתם שלנו לעומת האלגוריתם החמדן בכדי לאשש את תוצאות המחקר ובפרט לוודא שעל אף השינויים המינוריים שלנו – האלג' עדין טוב יותר מהחמדן. בנוסף, הבדל נוסף בין המאמר לבין המימוש שלנו הוא שלא יכולנו לבקש מהמשתמשים להאזין לכל שיר ורק לאחר כך להחליט האם הם אוהבים אותו או לא (במאמר פיתחו לכך אפליקציה וסימלצו על משתמשים ופה היה לנו קושי לבקש ממשתמש לשמוע מספר שירים בכל ניסוי ורק לאחר מכן לדרג) ולכן המשתמש פשוט קיבל את השיר הבא ברשימה ודירג אותו זה היה טריגר נוסף לבחירת דאטה שונה מהמאמר שכן השירים של ספוטיפיי מוכרים ממש ומוגדרים top hits ולכן הסיכוי שמשתמש לא יכיר את אחד השירים נמוך ומכך שהדאטה יותר מותאם לאופי המערכת שלנו. בכל ניסוי סימלצנו רשימה רנדומלית של שירים שהמשתמש אוהב.  
הרצו 3 ניסויים שונים כאשר בכל ניסוי היו 20 משתמשים -  
תחילה, הגדרנו את מספר הפרמוטציות בכל הרצה מאלגוריתם 4 להיות 100 וכל משתמש קיבל רשימה של עשרה שירים מהאלגוריתם החמדן אל מול עשרה שירים מהאלגוריתם שלנו.   
תוצאות הניסוי -  
בניסוי הראשון היה ניתן לראות מספר מגמות – ראשית ממוצע השירים שהמשתמש אהב באלגוריתם החמדן היה 7.6 אל מול 7.05 באלגוריתם שלנו, כמו כן, בשירים הראשונים, האלג' החמדן מניב תוצאות יותר טובות מהאלגוריתם שלנו, נקודה שהובילה אותנו לניסוי מספר שלוש.   
הגרפים המצורפים מציגים את תוצאות הניסוי, כאשר בכל ניסוי מוצגים שני גרפים:  
גרף 1 שמראה את כמות המשתמשים שאהבו את השיר ה-K (ציר X מייצג שיר, וציר Y את כמות המשתמשים שאהבו את השר)  
גרף 2 שמראה את כמות השירים שמשתמשים אהבו מתוך העשר (ציר X מייצג את כמות השירים האהובים, וציר Y את סך המשתמשים)

ניסוי מספר 2 -   
הניסוי התבצע בצורה זהה לניסוי מספר אחד עם אופק רשימה באורך עשר אך כעת מספר הפרמוטציות שנבדוק באלגוריתם 4 שונה מ 100 ל1000 .  
תוצאות : כעת ניתן היה לראות שיפור באלגוריתם שלנו שהוביל לממוצע של 7.7 שירים שהמשתמש אוהב מתוך הרשימה. ובנוסף ניתן לראות כי ככל שמתקדמים ברשימה נראה המשתמש מעדיף את השירים שהמערכת שלנו מציעה.

ניסוי מספר 3 -  
בניסוי השלישי החזרנו את מספר הפרמוטציות להיות 100 (אלגוריתם 4) אך הגדלו את אורך רשימת השירים מעשר לעשרים שירים – בכדי לבדוק האם האלגוריתם שלנו יותר טוב מהחמדן לאורך זמן.

תוצאות –  
  
כאן ניתן לראות שהאלגוריתם החמדן הוציא תוצאה פחות טובה מהניסויים הראשונים, שכן בניסויים האלו ממוצע השירים האהובים שהניב החמדן הינו באזור ה7.5 שירים, לעומת זאת בניסוי השלישי, כאשר העלנו את מספר השירים ל-20 הממוצע של 13.55 שירים מתוך 20 לעומת 14.4 באלגוריתם שלנו.  
למעשה, ניתן לראות כי ככל שהאלגוריתם לומד יותר ויותר את העדפותיו של המשתמש הוא מניב תוצאות טובות יותר וכמתואר במאמר – אנו מצפים שבשלב כלשהו האלגוריתם יתכנס ויפסיק את תהליך הלמידה כך שהמשתמש יוכל להפסיק לתת פידבק כל פעם.

**דיון**  
במסגרת הפרויקט בחרנו להתמקד ולהתנסות בתחום הלמידת חיזוק וקבלת החלטות רציפה כפי שלמדנו בכיתה בהרצאה השמינית תחת הנושא explainable AI: reinforcement learning.   
נכנסו לתוך עולם המערכות המלצה ומימשו מערכת המלצות מוזיקלית מבוססת RL ליצירת רשימת השמעה אופטימלית למשתמש אשר למתחשבת גם במעברים בין השירים בעת ההמלצה.  
עוד מתחילת קריאת המאמר הסכמנו עם הרעיון המרכזי – אנחנו מאמינים כי ישנה חשיבות גדולה למעברים השירים בעת יצירת פלייליסט ולא רק להעדפות, ולאחר מימוש האלג' והרצתו נוכחנו לגלות כי המאמר אכן צודק.  
למאזין האנושי חשוב שהשיר הבא שיתנגן יהיה לטעמו והוא לא יצטרך פשוט "להעביר" הלאה עד שיגיע לשיר שירצה לשמוע ברגע נתון. אם נחשוב על זה, אנחנו אוהבים שירים מז'אנרים שונים (פופ, רוק, מטאל, קלאסי..) אבל זה לא אומר שבהכרח נשמח לשמוע שיר קלאסי לאחר שיר מטאל – יש חשיבות לרצף השירים שמתנגן לנו באוזן.  
למדנו כיצד להתאים את את בעית ההמלצה לעולם הלמידה מחיזוקים – החל ממידול מצב העולם כרשימת השירים שהושמעו עד כה, בניית פונקצית רווח שמחולקת ל2 תתי פונק' ומושפעת מפידבק מהמשתמש, ועד שימוש בפידבק של המשתמש בכדי למצוא ולשפר את הפוליסה הטובה ביותר עבור המודל (בחירת השיר הבא).  
למעשה, בכל צעד, המודל שלנו בוחר לבצע את הפעולה (שיר) שתניב לו את הרווח הטוב ביותר בהתאם למדיניות שגיבש, ובתגובה הוא מקבל פידבק מהמשתמש לפיו הוא מעדכן את המדיניות עבור האיטרציות הבאות.  
אילו היה לנו יותר זמן ומשאבים לפרויקט (משתתפים בתשלום) היינו בונים ממשק יותר טוב שמאפשר למשתתף קודם לשמוע את השיר ורק אחרי זה לפדבק אותו, בנוסף היינו מנסים למצוא את מספר האיטרציות האופטימלי למערכת בכל גודל של רשימת שירים ובכך היינו מצליחים להפיק תוצאות משמעותיות בהרבה.  
כמו כן, הינו רוצים לשנות את פונקצית הניקוד של המשתמש לשיר (כזכור, פונקציה בינארית Like/unlike) ולבססה על משך זמן הניגון של כל שיר.  
למעשה, אנו חושבים כי משך זמן הניגון הינו סממן למידת אהדתו של המשתמש מהשיר שמתנגן – לדוגמא, אם לאחר קבלת השיר הבא ברשימה המשתמש "מעביר הלאה" לשיר אחר תוך 5 שניות, אז זה מעיד שהוא אינו אוהב את השיר ומקביל לunlike כפידבק. אבל אם המשתמש מאזין לשיר כמעט עד הסוף זה אומר שהוא מרוצה מהשיר וכך למעשה נרצה למדל את הפידבק מהמשתמש – פונק' שאינה בינארית אלא סקאלה בין 0 ל1 שמתאימה למשך הניגון של שיר. בשביל לעשות זאת נצטרך כמובן גם לבנות ממשק האזנה למשתמש, לשמור את משך הניגון של כל שיר לאחר לאחר שלבי האתחול ולעדכן את המודל בהתאם.   
בממשק שאנחנו בנינו המשתמש רק צריך ללחוץ אוהב/לא אוהב לשיר שמתנגן ולכן הרצת האלגוריתם על משתמשים אנושיים לא לקחה זמן ארוך והקלט לפונקציה היה פשוט (1 או 0). אם היה לנו עוד זמן הינו בונים ממשק משופר שהקלט הינו משך הזמן מרגע הרצת השיר ועד השיר הבא/עד לחיצת כפתור וממפים אותו לניקוד בהתאם.

הקוד שלנו בגיט האב : <https://github.com/odedsalton/interactive_machins>