‏7/1/2021

אלון קרמונה ויהונתן להט

312469083, 308147248

לימוד מכונה

חלק ב'

תוכן עניינים

[הכנת הנתונים לאימון ובחינת מערכת לומדת 2](#_Toc44364162)

[Decision Trees 2](#_Toc44364163)

[בניית עץ החלטה מלא 2](#_Toc44364164)

[כוונון פרמטרים 3](#_Toc44364165)

[בניית עץ ההחלטה האופטימלי 3](#_Toc44364166)

[רשתות נוירונים מלאכותיות 5](#_Toc44364167)

[בנייה ואימון המודל הראשוני 5](#_Toc44364168)

[Hyperparameter Tuning 6](#_Toc44364169)

[משימת Binary Classification באמצעות רשת נוירונים 7](#_Toc44364170)

[SVM 8](#_Toc44364171)

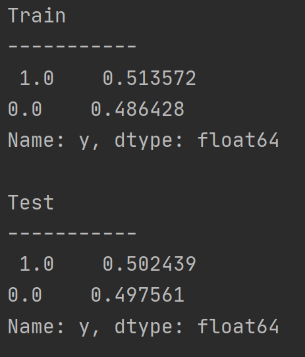
[Unsupervised Learning - Clustering 8](#_Toc44364171)

[השוואה בין מודלים- Evaluation 11](#_Toc44364176)

שיפור [המודל הנבחר 12](#_Toc44364180)

[נספחים 13](#_Toc44364183)

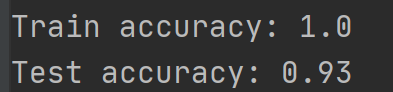
# הכנת הנתונים לאימון ובחינת מערכת לומדת

בהמשך לחלק א, ביצענו עוד מספר מניפולציות לבסיס הנתונים שלנו טרם התחלת אימון המודל. ראשית, החלטנו להפוך את שני המשתנים הקטגוריאליים developer ו-publisher לשני פיצ'רים בינאריים שעונים על השאלה " האם המפתח/המשווק של המשחק בטופ 10", אנו מאמינים שאין צורך במספר כה גדול של קטגוריות בפיצ'רים אלה. כמו כן, כאשר ביצענו הדפסה של גרף החלוקה על פי מפתחים/משווקים, ראינו כי ישנו מספר מצומצם של מפתחים/משווקים שמהווים שחקי מפתח בשוק משחקי המחשב – ורצינו לתת להם את ההתייחסות הנכונה. משתנה המטרה שלנו, כמות מכירות באירופה, הינו רציף. נשים לב כי משימת הלימוד המתאימה לסוג משתנה מטרה זה הינו משימת חיזוי. כמבוקש, ביצענו המרה של משימת הלימוד למשימת Classification בעזרת פונקציית KBinsDiscretizer, כאשר התוצאה היא חלוקה מאוזנת של מספר הרשומות בכל קלסיפיקציה, כפי שניתן לראות בפלט משמאל.  
חילקנו את מחלקות כמות המכירות באירופה בצורה אורדינלית, כך ש-0 מייצג את הקלאס בו נעשה פחות מכירות ו-1 מייצג את המשחקים שנמכרו בצורה יותר נרחבת. יחס הערכים של 1 ו-0 במשתנה המטרה y - נשמר יחסית זהה למה שהתקבל עבור סט הנתונים הכולל.

חלוקת הנתונים לאימון וולידציה: בחרנו בשיטת הולידציה holdout. על פי שיטה זו, יש לחלק את סט הנתונים לסט אימון וסט ולידציה- כך שנאמן את המודלים על סט האימון בלבד, ואת תהליך Hyperparameter Tuning נעשה על ידי מקסום המטריקה הנבחרת.   
חילקנו את סט הנתונים הכולל לשני סטים שונים כאשר משתנה test\_size הוגדר להיות 0.1. משמעות קביעת פרמטר זו היא למעשה ש-90% מן נתונים יישאר בסט האימון – והנתונים הנותרים יופנו לתהליך הולידציה. מספר גבוהה של נתונים בסט האימון יסייע בתהליך הלמידה של המודל – עם זאת יש לוודא כי ישארו מספיק נתונים בסט הולידציה בכדי שיהוו מדגם מייצג לבחינת המודל. משתנה random\_state הוגדר להיות 42, עליו נשמור למען האחידות שבבחינת המודלים השונים שניצור.   
מספר התצפיות שהתקבלו בסט האימון – 5524 תצפיות, בסט הולידציה – 614 תצפיות.

# Decision Trees

1. **בניית עץ החלטה מלא**



ראשית, ביצענו מודל של עץ החלטה מלא המבוסס על סט האימון – ללא הגבלת רמות פיצול. מכיוון שהמודל אומן על כל ההסתעפויות האפשריות הקיימות בנתונים (כל סאמפל סווג על פי ערכי הפיצ'ארים שלו) - אחוז הדיוק שהתקבל על סט האימון הוא 100%. המודל נמצא במצב הנקרא over-fitting, בו הוא מעיד על תאימות מלאה בין המודל לסט האימון ולכן אחוז הדיוק אינו מפתיע.  
אך כשקיבל מידע חדש שלא היה בסט האימון – ירד אחוז ה-Test accuracy.   
לכן נשאף למודל קומונאלי ופשוט יותר, שידע להבין את הכלל לפי הפרט, ובו סביר שאחוז הדיוק על סט האימון לא יהיה מושלם וירד מערך 1. אחוז הדיוק שהתקבל על ה-Test set הוא 93% . ניתן להסיק כי זהו ערך טוב למדי אך ניתן לשפרו, כפי שנראה בהמשך.

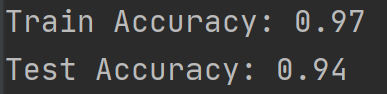
1. **כוונון פרמטרים**

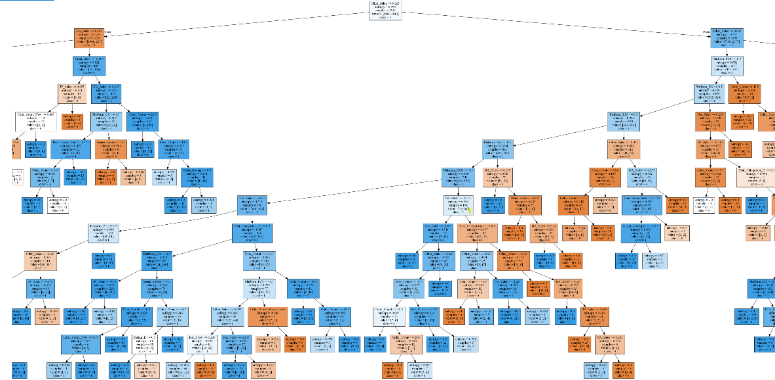
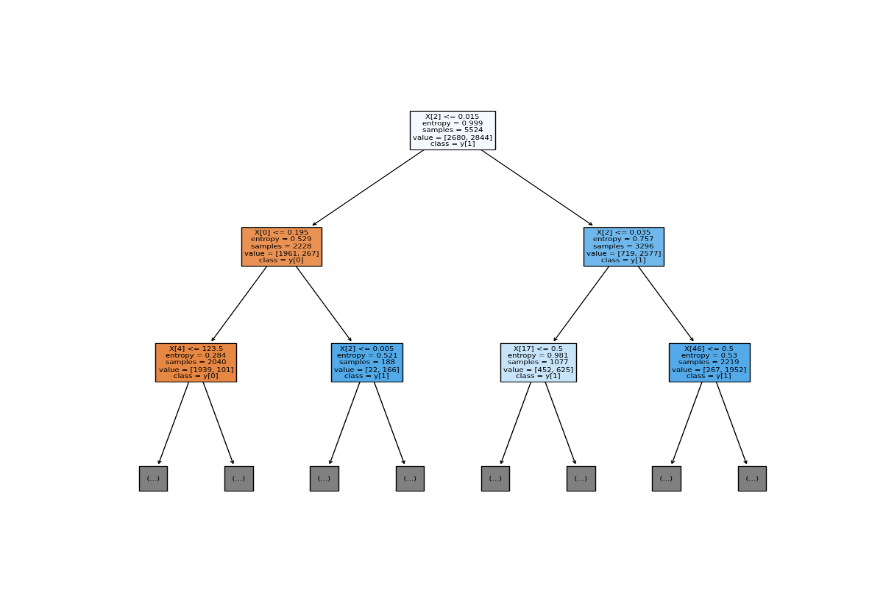
ביצענו שלב זה בשיטת Grid-Search. שיטה זו למעשה בוחנת את כל הקומבינציות האפשריות (במקרה שלנו 1176), כך שהיא מכסה את כל האופציות, ולכן בעלת זמן ריצה ארוך. בהרצת שיטה זו על הנתונים שלנו, זמן הריצה ארך כ-3 דקות, זמן לא קצר אך מקובל על ידינו בשביל יתרון כיסוי מרחב האפשרויות שלה וזאת על מנת למקסם ערכים ולשפר את המודל. הפרמטרים שבחרנו לבדוק:

|  |  |  |
| --- | --- | --- |
| פרמטר | אופציות | הסבר |
| max\_depth | 1-25 | מייצג את עומק העץ המקסימלי האפשרי. עץ בעל עומק רב יכיל יותר הסתעפויות שמשקפות את תהליך הלמידה המדויק שלו את נתוני סט האימון. אך מנגד, תהליך למידה ממוקד יתר על המידה יכול לגרום ל- over-fitting (כפי שהסברנו לעיל). לכן זוהי המוטיבציה לכוונונו. בחרנו לבדוק טווח של 1-25, שמייצג את הספקטרום הסביר של התוצאות שאנו עשויים לקבל (זאת מכיוון שבעץ ההחלטה המלא- קיבלנו עץ בעל 24 רמות).  **הערך האופטימאלי שנבחר עבור פרמטר זה הוא 13.** |
| criterion | "gini", "entropy" | בבואנו לבחור את הפיצול הטוב ביותר לפי מאפיין מסוים, נחשב את מידת השפעתו בבחירת חלוקת ענפי העץ ישנם שני מדדים שונים – אנטרופיה (הבודק את רמת ה-"אי-סדר" בתצפיות) וג'יני (המחשב הסתברויות אפריוריות לסיווג לא נכון).  **הערך האופטימאלי שנבחר עבור פרמטר זה הוא "entropy".** |
| Max\_features | 'Auto',  'sqrt',  'log2',  None | מגביל את כמות המאפיינים בהם יש להתחשב כאשר מחפשים את ההסתעפות הטובה ביותר. טווח הערכים (במחרוזות) מייצגים נוסחאות חישוב שונות שנלמדו בהרצאה. במיוחד לאחר הוספת משתני הדמה, המוטיבציה להגבלת כמות המאפיינים היא אפשרות הוצאת חלק מהפיצ'רים שאינם תורמים לדיוק המודל- ובכך לפשט את העץ.  **הערך האופטימאלי שנבחר עבור פרמטר זה הוא "None".** |
| min\_samples\_leaf | 1,2,3 | מספר התצפיות המינימלי הנדרש בעלים בבסיס העץ. נצפה שכשערך זה יגדל, הדבר יקטין את הover-fitting (יפחית מקריות). נבדוק את הערכים 1-3 עבור פרמטר זה.  **הערך האופטימאלי שנבחר עבור פרמטר זה הוא 3.** |



1. **בניית עץ ההחלטה האופטימלי**

ניתן לראות מתוצאות המודל החדש (שמעודכן לערכי ההיפר-פרמטרים שנבחרו), כי על אף הירידה באחוז הדיוק על סט האימון- ישנה **הגדלה** באחוז הדיוק של סט הבחינה. העץ קטום, בניגוד לעץ ההחלטה מלא כמו בסעיף 1, בשילוב עם שינויי הפרמטרים האחרים (כמו מינימום 3 סמפלים בעלים)- גרמה לכך שאחוזי הדיוק על סט האימון אינם 100% וכי **הפחתנו משמעותית את מצב ה- over-fitting.** יש לציין כי בנוגע לכמות הפיצ'רים המדד הנבחר הוא NONE. משמע, האלגוריתם בחר לא להוריד פיצ'רים בחלק זה (נדון בבחירת הפיצ'רים על פי חשיבותם בהמשך).

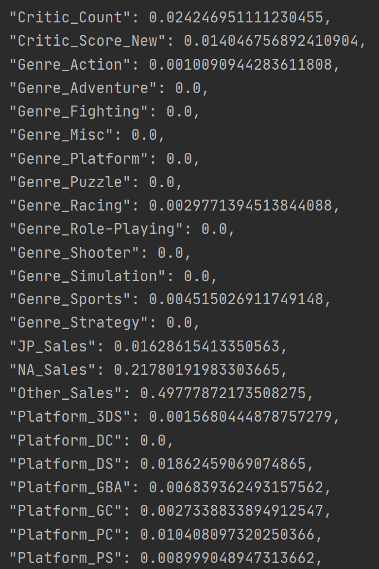


***"אחד היתרונות של מודל עץ החלטה הוא יכולת ההסברה של המודל (interpretability)"***

כוונתו של המשורר היא בראש ובראשונה שעץ החלטה הוא מודל הנחשב כפשוט להבנה ולפירוש, כלומר אנשים מסוגלים להבין דגמים של עצי החלטה במהירות ובקלות ביחס למודלים אחרים מורכבים יותר. מכיוון שעץ החלטה מורכב מצמתי החלטה שבכל אחד מהם נבדק תנאי על מאפיין מסוים של התצפיות ומעלים המכילים את הערך החזוי עבור התצפית המתאימה למסלול שמוביל אליהם בעץ, המודל מאפשר הדמיה ויזואלית של קבלת ההחלטות- דבר שתורם ליכולת ההסברה שלו, ובעצם הופכת את עץ ההחלטה לייצוג פשוט לסיווג דוגמאות.

ע"י התבוננות במבנה העץ שהתקבל ניתן ללמוד מספר תובנות, ראשית, ניתן להבחין כי משתנה other\_sales ממוקם בראש העץ וכן הוא בעל ערך ממד האנטרופיה הגבוה ביותר. אנו לומדים מכך שהחשיבות של פיצ'ר זה בסיווג הנתונים היא הגבוהה ביותר. לעומת זאת, ישנם משתנים אחרים שלא מופיעים כלל בעץ ההחלטה שהתקבל ולכן ניתן להסיק כי הם לא נמצאו כמסווגים טובים למודל. את המשתנים שלא נכללו כלל בעץ נוכל לראות בתת הסעיף הבא על ידי מדד חשיבות המשתנה שיהיה שווה אצלם ל0.

הפלט המתקבל מהרצת פונקציית feature\_importances על עץ ההחלטה מוצג משמאל, הנתונים מייצגים את התרומה של כל מאפיין לסיווג המודל. הפלט מחושב על ידי השמת אחוז התרומה למודל מתוך כלל הפיצ'רים. כאמור, ניתן לראות בפלט שהתקבל עבור המודל, שהמאפיין בעל הערך הגדול ביותר (0.497) הוא *other\_sales* שמופיע בראש העץ. בנוסף, אפשר לראות כי משתנה *NA\_Sales* ניצב אחריו עם ערך גבוה של (0.217) – תרומה רבה ביחס לשאר הפיצ'ארים. הדבר עולה בקנה אחד עם ההיגיון, שמודל פרדיקציה עבור מכירות באירופה - ייתן חשיבות גדולה לכמות המכירות בארצות אחרות. לעומת זאת, אנו יכולים לראות כי ישנם מספר רב של המאפיינים בעלי ערך של 0, ולכן ערכים נמוכים אלו לא מופיעים כלל בעץ (לדוגמא *Platform\_DC*). פיצ'רים אלו לא יתרמו דבר למשימת הקלסיפיקציה שלנו, ונשקול לבצע להם מניפולציה בשלב שיפור המודל.

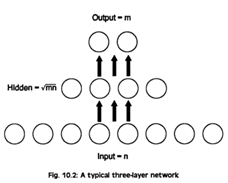


# רשתות נוירונים מלאכותיות:

נעשה שימוש במודל לימוד מכונה נוסף אשר מקורו בניסיון להתחקות אחר הדרך בה המוח האנושי עובד. המודל בנוי מנוירונים שלקשרים ביניהם משמעות חישובית. ב"שכבת הכניסה" מספר נוירונים שקולטים את המידע מהפיצ'רים השונים של כל סאמפל, על כל נוירון יש משקל שיכול להתעדכן ולהשתנות במהלך תהליך הטריינינג. יכולות להיות מספר שכבות שמוגדרות כ-Hidden שמטרתן לשקלל את ערכי הנוריונים הקשורים אליהן, בהתחשב במשקלים השונים, ולבצע פעולת אקטיבציה – טרם שליחת המידע המעובד לשכבה הבאה. בהתאם למשימת הלמידה שלנו, הפלט הסופי יהיה הסיווג שניתן לתצפית.   
תהליך הלמידה של המודל נעשה לאחר קבלת הפלטים- המודל יחשב את השגיאה ויבצע תהליך של עדכון המשקלים שברשת ע"י גזירת פונקציית האקטיבציה – במטרה למזער את השגיאה.  
הדרך בה המודל מאופיין גורמת לו להיות "רגיש" לסקאלת השונות של הפיצ'רים. על כן, יש לבצע נורמליזציה או סטנדרטיזציה לנתונים טרם הזנתם. ננרמל את הפיצ'רים הרציפים כך שיהיו בעלי תוחלת 0 וסטיית תקן 1 (ניתן לראות בקוד המצורף ביצוע תהליך זה). כמו כן, עבור הפיצ'רים המוגדרים כקטגוריאליים נגדיר משתנה דמה שערכיהן בקבוצה {1,-1} – וזאת על מנת לתת יכולת השפעה למשתנים על פונקציית האקטיבציה. בכדי למנוע את סטיית הנתונים שבסט הולידציה – נבצע תהליך נירמול שיתאים לטריינינג סט המעודכן.

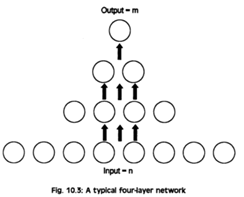
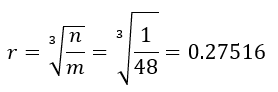
**בנייה ואימון המודל הראשוני**

בעיית הלמידה וסט הנתונים המעודכן מגדירים לנו את גודל שכבת הקלט והפלט של המודל. עם זאת, אלגוריתם ANN מאפשר לנו לשנות את מספר השכבות – ע"י הוספת שכבות נסתרות ואת מספר הנוירונים שיהיו בכל שכבת נסתרת.   
על פי הנלמד בהרצאה והספרות המדעית – מודל פשוט מדי יכול להוביל למצב של Under-fitting שכן תצורתו לא תספק את המשאבים הנדרשים לפתירת הבעיה. לעומת זאת, מודל מורכב ידרוש זמן ריצה ומשאבים רבים יותר ובנוסף לגרום ל-Over-fitting של המודל לטריינינג סט.

כשניגשנו לספרות המקצועית מצאנו את הספר Practical Neural Network Recipes in C++”", בספר מדובר על החשיבות שבהוספת שכבה נסתרת- אך שההמלצה היא להשאיר את המודל כמה שיותר פשוט ולהתחיל מהשמת שכבה נסתרת אחת בלבד ולאחר מכן לבחון הוספת שכבה נוספת. כשזה נוגע למספר הנוירונים בשכבה הנסתרת במודל תלת שכבתי (שכבת קלט, שכבה נסתרת ושכבת הפלט), נעשית אפרוקסמיזציה באמצעות geometric pyramid rule שמציעה את היורוסטיקה הבאה: עבור מודל תלת שכבתי, כאשר שכבת הקלט הינה בגודל m ושכבת הפלט בגודל n נוירונים – השכבה הנסתרת תכיל את שורש המכפלה של m ו-n.  
במקרה שלנו, הפלט הנדרש הוא למעשה סיווג בינארי של משחק המחשב, ועל כן n=1.  
בשלב זה הסט מכיל 48 פיצ'רים – ביניהם משתני דמה קטגוריאליים ומשתנים רציפים מנורמלים, לכן m=48. נחשב את כמות הנוירונים הרצויה בשכבה הנסתרת:   
 כלומר, שבעה נוירונים.

בהרצאה הוצגו לנו מספר פונקציות אקטיבציה בהן נעשה שימוש באלגוריתמי ANN, מטרת הפונקציות היא לקיחת הסכימה שבתוך הנוירון והתמרת התוצאה לערך אחר. נבחר להשתמש בשלב זה בפונקציית logistic sigmoid, זוהי פונקציית פחיסה שיכולה למעשה לקבל טווח נרחב של ערכים אותה היא פוחסת לטווח בתחום (0,1). פונקציה זו מתאימה למשימות קלסיפיקציה. יתרון נוסף של פונקציה זו היא גזירתה הפשוטה יחסית בה יעשה שימוש בתהליך כיוונון המשקולות.  
הרצנו את פונקציית MLPClassifier עם max\_iter = 1000 ושאר הערכים דיפולטיביים כמבוקש.  
מכיוון שלאלגוריתם רכיב רנדומלי- נשמור על מספר Seed קבוע בכדי לעקוב אחר התהליך.

להלן נתוני הריצה וגרף של פונקציית ה-loss ביחס למספר האיטרציה. ניתן לראות כי נעשו כל ה-1000 איטרציות וכי נגזרת ה-loss בערך מוחלט קטנה ככל שמספר האיטרציות עולה. בנוסף, ניתן להבחין כי בסביבות האיטרציה ה-70 ישנו שינוי מגמה של שיפוע הגרף.  
עבור המודל הראשוני הגענו לדיוק של 94% על סט האימון ושל 83% עבור סט הולידציה. כאשר מדובר במשימת קלסיפקציה בעלת 2 סיווגים אפשריים – ניתן לומר כי מדובר בנתון סביר עבור סט הולידציה וזאת לעומת אחוז דיוק גבוה יחסית עבור סט האימון. ומכאן נסיק כי יכול להיות שהמודל הותאם יתר על המידה לסט האימון ונמצא במצב של Overfitting.

כעת ננסה את המודל השני שהוצע במאמר, מודל ארבע-שכבתי ובו שתי שכבות נסתרות. על פי המודל מספרי הנוירונים מחושבים בצורה הבאה:

נריץ את המודל כעת עם שתי שכבות נסתרות, בגדלים 13 ו-4.  
עבור המודל הנ"ל הגענו לדיוק של 95% על סט האימון ושל 83% עבור סט הולידציה. ניתן לומר שהשיפור הוא מינורי.

**Hyperparameter Tuning**נבצע תהליך של היפר-פרמטריזציה על מנת לנסות לאתר את התצורה שתיטיב עם המודל ותשפר את תוצאותיו על סט הולידציה. במודל רשתות נוירונים ישנם מספר גדול יחסית של אפשרויות עבור הפרמטרים, בנוסף, כמות הרשומות הגבוהה יחסית יעמיס על החישובציה – לכן, נבחר בשיטה היעילה יותר מבחינת זמן הריצה. שיטת randomSearch היא שיטה הדוגמת באופן רנדומלי ערכי פרמטרים מתוך רשימה מוגדרת מראש, בכל איטרציה בוחנת שילוב של פרמטרים שונים תוך השמת דגש על פרמטרים בעלי השפעה. כעת נכתוב את המוטיבציה בבדיקת כל היפר-פרמטר:  
הרצנו את השיטה עשר פעמים ומתוכם בחרנו את הפרמטרים שמביאים לתוצאה הטובה ביותר עבור ה-Accuracy בולידציה, בכל ריצה טווח הפרמטרים היה שונה וכך גם מספר הרצות השיטה.

|  |  |
| --- | --- |
| פרמטר | הסבר |
| Hidden\_  Layer\_ Sizes | כל השמת ערך תוסיף שכבה נסתרת למודל, הערך הינו מספר הנוירונים שבשכבה. מספר קטן של שכבות ומספר נוירונים קטן בהן עלול לא להביא להתקרבות הרצויה ל"פונקציה האמיתית" ובכך למצב של Under-fitting. לעומת זאת, שכבות נסתרות רבות ובהן מספר נוירונים רב יכול ליצור מודל שיהיה מורכב יתר על המידה, המודל ידרוש זמן ריצה ומשאבים רבים יותר. בנוסף, מודל כזה צפוי לגרום ל-Over-fitting של המודל לטריינינג סט. **הערך שנבחר עבור פרמטר זה הוא - (58, 82).** |
| Activation | פרמטר אשר מקבל ערך string המייצג את פונקציית האקטיבציה המבצעת התמרה לערך אותה היא מקבלת. נבחר להתמקד בשתי פונקציות עליהן הייתה התייחסות בהרצאה – פונקציית logistic ופונקציית relu. **הערך שנבחר עבור פרמטר זה - "logistic".** |
| Alpha | ערך זה הוא למעשה גורם ה"ענישה" על גודל המשקולות, אלפא גדול ישאף לפחות פניות חדות על הגרף. המטרה של פרמטר זה הינה למנוע בעיית fitting. **הערך שנבחר עבור פרמטר זה - 0.001** |
| Learning\_ Rate\_Init | מקבל ערך מספרי חיובי הקטן מאחד, בוחן את קצב הלמידה של המודל כתוצאה מכיוונון המשקולות שנעשה באיטרציות. כאשר השינוי בקצב הלמידה קטן מהערך – התהליך ייפסק. ערך גדול יוביל לכך שייתכן והמודל יתמקד בנקודת אופטימום לוקאלית במקום להתקדם למודל טוב יותר. לעומת זאת, ערך קטן מדיי עלול להוביל למצב של overfitting. **הערך שנבחר עבור פרמטר זה הוא 0.01.** |
| Max\_Iter | מספר האיטרציות שאם יגיע אליו המודל – הוא ייעצר ללא קשר לשאר הפרמטרים. ערך קטן עשוי לגרום לכך שהמודל יסיים את תהליך הלמידה מוקדם מהנדרש ויחמיץ מודל שהיה מביא תוצאות טובות יותר. מאידך, ערך גבוהה ייכשל במניעת מצב של overfitting. **הערך שנבחר עבור פרמטר זה הוא 46.** |
| Early\_ stopping | פרמטר בוליאני. אם יקבל ערך True המודל יחלק סט ולידציה אותו יבחן במהלך האיטרציות. אם אחוז הדיוק בסט זה לא ישתפר עבור יותר מ-10 איטרציות – המודל ייפסק. בדומה לפרמטרים אחרים, עלול לפספס מודל טוב יותר אך גם עשוי למנוע overfitting על סט האימון. **הערך הנבחר הוא True.** |

נראה כי תהליך היפר-פרמטריזציה שנעשה באמצעות שיטת Randomsearch , הביא לשיפור של אחוז אחד במידת הדיוק בסט האימון ו-2% בסט הולידציה – לעומת המודל הראשוני שהורץ. השיפור הקטן ב-Accuracy של ה-Train set עשוי להיות תוצר של סיבוך המודל ע"י הוספת נוירונים נוספים לשכבות הנסתרות. השיפור בדיוק של סט הולידציה עשוי להיות מושפע מההפחתה המשמעותית במספר האיטרציות. כאמור, מספר איטרציות גבוהה מדי עלול לגרום להתאמת יתר של המודל לסט האימון.

**משימת Binary classification באמצעות רשת נוירונים**

במשימת קלאסיפקציה בינארית במודל רשתות נוירונים, לאחר שהמודל עושה התאמה לטריינינג סט הוא מקבל את הפיצ'רים של כל סאמפל ונותן לו סיווג 0 או 1 בהתאם להסתברות שהסאמפל שייך לקלאס - על פי תהליך הלמידה של המודל. בעזרת פונקציית predict\_proba שנעשתה על המודל הנבחר לאחר תהליך ההיפר-פרמטריזציה – קיבלנו את ערכי ההסתברות להשתייכות הסאמפל לכל סיווג. כמובן שהערכים עבור שני הקלאסים משלימים ל-1. הסיווג נעשה על פי הקלאס שקיבל הסתברות הגדולה מ-0.5.   
להלן דוגמה לרשומה:  
על פי דוגמה זו, המודל חוזה כי המשחק Call of Duty: Black Ops II שייך לקלאס 1 שהוא הקלאס המייצג את המשחקים הנמכרים יותר. ההסתברות לכך על פי היא מעל 96%.

|  |  |  |  |
| --- | --- | --- | --- |
| Real Y Value | Prob. for class 1 | Prob. for class 0 | Game Name |
| 0.05 | 0.455859236672 | 0.544140763327 | Super Princess Peach |
| 0.13 | 0.460071639568 | 0.539928360431 | Dragon Ball Z: Supersonic Warriors |
| 0.25 | 0.507222709546 | 0.492777290453 | Call of Duty: World at War |
| 0.13 | 0.500457342779 | 0.499542657220 | Tim Burton's The Nightmare Before Christmas: Oogie's Revenge |

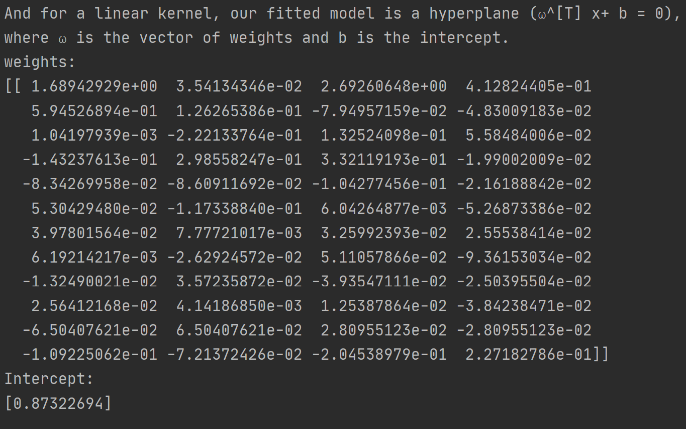
התבקשנו להביא את ארבעת המשחקים שהמודל שיצרנו הכי פחות בטוח בסיווג שניתן לו, שניים מעל סף ההחלטה ושניים מתחתיו. להלן טבלה המציגה משחקים אלה:

**SVM**

המסווג שנוצר ב- SVM הוא [המפריד הליניארי](https://he.wikipedia.org/w/index.php?title=%D7%9E%D7%A4%D7%A8%D7%99%D7%93_%D7%9C%D7%99%D7%A0%D7%99%D7%90%D7%A8%D7%99&action=edit&redlink=1) אשר יוצר מרווח גדול ככל האפשר בינו לבין הדוגמאות הקרובות לו ביותר בשתי הקטגוריות.  אנו רוצים למקסם את רוחב השוליים מתוך מטרה לצמצם את התצפיות החריגות ולתת מרחב בטחון.   
על מנת למצוא את הקונפיגורציה המיטבית, הרצנו את המודל עם *SVMGrid* כאשר בחנו מספר פרמטרים אותם האלגוריתם ייבחן:

|  |  |  |
| --- | --- | --- |
| פרמטר | אופציות | הסבר |
| Kernel | ,''linear  'poly',  'rbf',  'sigmoid' | "תעלול הגרעין" – בפרמטר זה אנו יכולים להחליף את המכפלה הפנימית בה משתמשים עבור המפריד הליניארי, בפונקציית גרעין (kernel function) אשר מדמה את פיזורם מחדש של הווקטורים המקוריים במרחב עשיר יותר, ללא עלות חישובית משמעותית. לפרמטר זה מספר אופציות וייתכן שכל אחת מן האופציות תטייב את הדיוק שלנו בהמשך. עם זאת, בכדי להציג את המשקולות המוקצות לתכונות (המקדמים בבעיה הראשונית) בעזרת 'coef' שזמין רק במקרה הליניארי, אנו **נבחר להשתמש בlinear**. |
| C | 1 - 5  בקפיצות של 0.5 | אלגוריתם הלמידה מנסה לספק שתי מטרות: סווג נכון של דוגמאות האימון ומיקסום של השוליים. לא אחת- המטרות נמצאות בסתירה. במצה כזה, יש צורך להגדיר איזו מטרה יותר חשובה. על כן נגדיר קבוע ענישה על סיווג לא נכון - C. ערך C גבוה פירושו העדפת הסיווג הנכון על פני שוליים רחבים, ו- C נמוך מעדיף הכללה (שוליים רחבים), גם במחיר שדוגמאות האימון הספציפיות אינן מסווגות נכון. על כן ישנו טווח ערכים אופציונאליים, בחרנו לבדוק ערך בין 1 ל-5 בקפיצות של 0.5.  **הערך הנבחר ע"י האלגוריתם C= 3.5.** |
| Gamma | 'Scale' | פרמטר הגמא מגדיר עד כמה ההשפעה של דוגמת אימון אחת משפיעה, כאשר ערכים נמוכים פירושם 'מרחק רחוק' וערכים גבוהים פירושם 'מרחק קרוב'. במקרה שלנו, מפני שנבחר ב-linear, אין אופציית בחירה **והערך יהיה scale** (ערך גמא של 1 / (n\_features \* X.var())). |

לאחר שקיבלנו את ערכי הפרמטרים הנבחרים- הרצנו מודל SVM מותאם. קיבלנו את אחוזי הדיוק של המודל הנבחר על סט האימון והוולידציה:



נציג את משוואת הישר המפריד המיוצגת ע"י המטריצה:

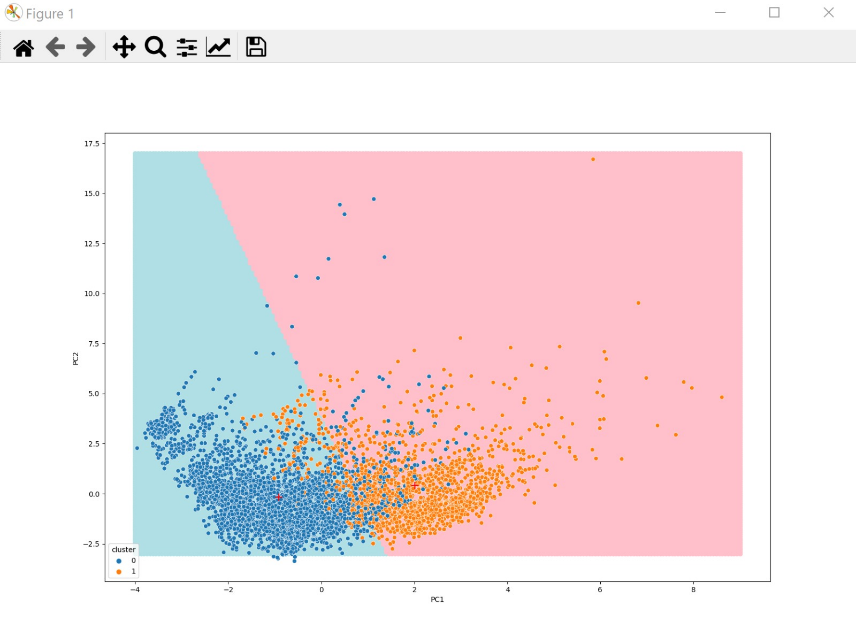
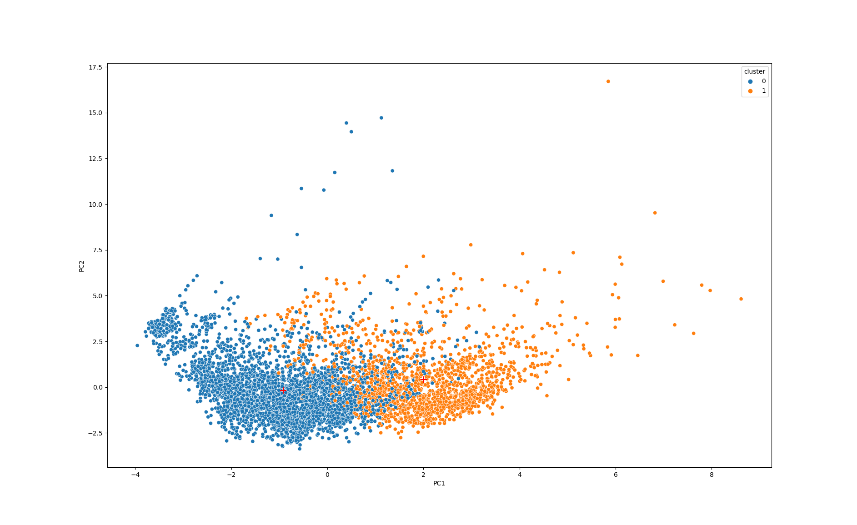
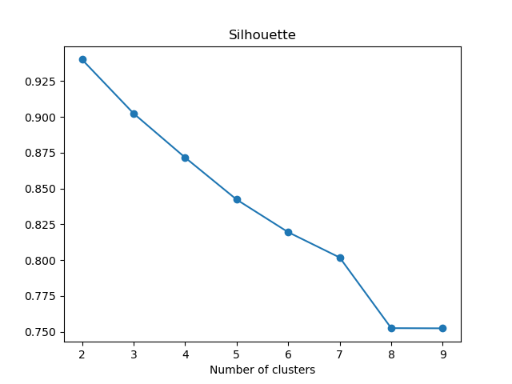
ממשואת הישר ניתן להסיק בנוגע לחשיבות כל פיצ'ר בתהליך הסיווג. ככל שהערך יותר גדול כך הפיצ'ר בעל חשיבות גבוהה יותר ותורם יותר לסיווג ומיקסום המדד שלנו. דבר זה תואם מסקנות קודמות שלנו, כאשר אנו רואים קורלציה בין התוצאה במשוואת הישר לבין מדד הfeature importance במודל העץ. לדוגמא, הפיצ'ר במקום ]2][0] במטריצה שהתקבלה בעל הערך הגבוה ביותר ואחריו זה שבמקום ה-[0][0] במטריצה, שהם התכונות other\_sales ו NA\_sales בהתאמה, והם קיבלו את הערכים הכי גבוהים גם במדד בעץ.

**במידה והיינו רוצים לבצע משימת קלסיפיקציה עם 3 מחלקות**, ניתן לעשות זאת בעזרת SVM.

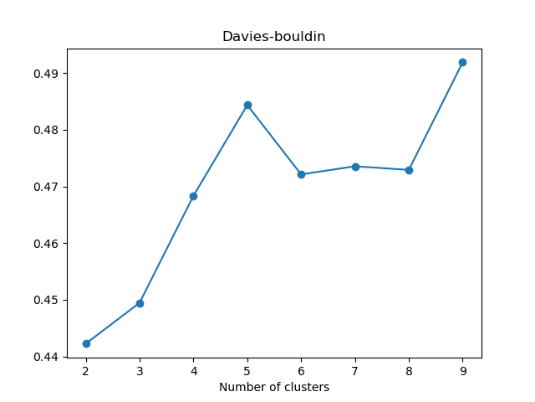
בסוג הפשוט ביותר שלו, SVM אינו תומך ב-multiclass classification. המודל תומך בסיווג בינארי והפרדת data points לשתי מחלקות. לכן, במידה ונרצה לבצע משימת סיווג של 3 מחלקות נשתמש באותו העיקרון, לאחר פירוק בעיית הסיווג למספר בעיות סיווג בינאריות. הרעיון הוא למפות data points למרחב ממדי גבוה כדי להשיג הפרדה לינארית הדדית בין כל שני Classes. הדבר נקרא גישת One-to-One approach, המפרקת את בעיית הmulticlass לבעיות סיווג בינאריות מרובות- מסווג בינארי לכל זוג Classes.   
גישה אחרת שאפשר להשתמש בה היא One-to-Rest. בגישה זו, ההתפלגות מוגדרת כמסווג בינארי לכל מחלקה ומתבצעת על כל קלאס בנפרד- אל מול השאר.

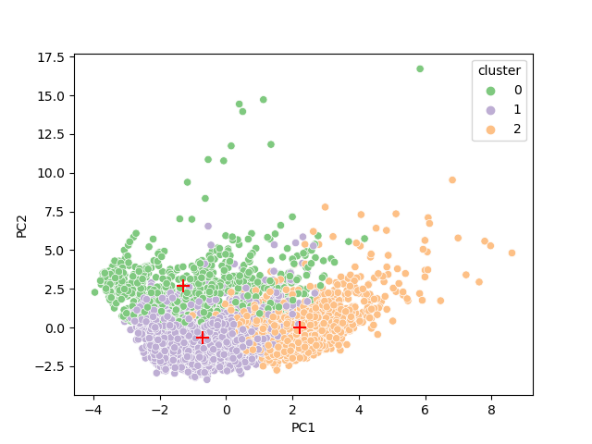
**Unsupervised Learning - Clustering**

שיטה נוספת לסיווג היא שיטת **K-means**. בשיטה זו לא נעשה שימוש בערכי ה-Y המייצגים את הסיווגים, אלא נעשה תהליך של Unsupervised Learning ובו הלימוד נעשה על פי הפיצ'ארים עצמם בלבד. היתרון של תהליך לימוד זה הוא שהוא לא מצריך דאטה מתויג – ובכך חוסך עבודה ידנית רבה במקרה וסיווגים אלו לא קיימים. לכן, נתעלם כאמור מהסיווגים הקיימים ונאפשר לאלגוריתם לבצע אשכול לסאמפלים ובכך למעשה לסווג על פי מספר המחלקות שניתן לו. בכדי לבחון את המודל – נוכל להשוות את הסיווגים לאלו הקיימים בסט האימון. כאן ביצענו נרמול של הערכים, על מנת להימנע מסטיות בחלוקה כתוצאה מהפערים הגדולים בטווחי המשתנים. לאחר מכן, השתמשנו בשיטת PCA להורדת מימדים של מאפיינים. ניתן לראות בנספחים דוגמא לחלוקה לפני ואחרי הנרמול. הרצנו את המודל הראשוני עם ערכי ברירת מחדל ועם מספר אשכולות כמספר ערכי Y – 2. לאחר ששקללנו את מרכזי האשכולות, קיבלנו את הגרף השמאלי ובו חלוקת הסאמפלים לאשכולות – על פי מרחקם ממרכזי שני האשכולות.

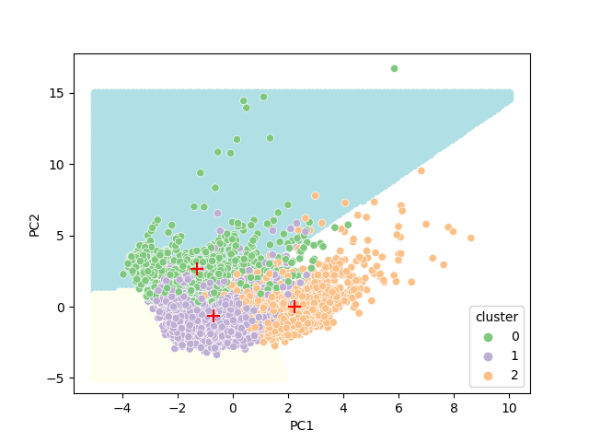
****מבדיקת חלוקת המרחב של האשכולות-התקבל הגרף הימני. על אף שהתצפיות לא מתחלקות באופן אחיד בין שני האשכולות, בהחלט ניתן לראות שהמסה הקריטית של תצפיות ששייכות לסיווג של מספר מכירות נמוך באירופה- שויכו לאשכול השמאלי התחתון. באופן משלים, תצפיות המתאימות לערך 1 שויכו לאשכול הימני עליון. אך מכיוון שמטרת המודל הינה אשכול ולא קלסיפיקציה – לא ניתן לדעת מהן המחלקות האמיתיות.   
בכדי לבחור את ה-K הטוב ביותר - שינינו את מספר האשכולות בכל אימון של מודל. בדקנו את המודל באמצעות המדדים שנלמדו בשיעור:

**Silhouette** – כל רשומה מקבלת ערך בטווח [-1,1], והוא מייצג את המרחק בין כל רשומה באשכול לכל הרשומות באשכול האחר. לבסוף, ערך המדד מתקבל מממוצע כל הערכים שהתקבלו עבור כל התצפיות. אנו נשאף למקסם מדד זה, כלומר נרצה לבחור בערך K אשר יגדיל את המרחקים בין האשכולות.

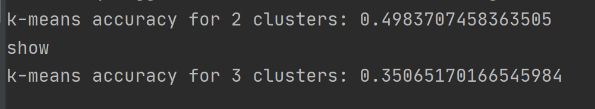
**Davies - Bouldin** – ערך המדד מחושב על פי רמת הדמיון בין אשכול לבין אשכול אחר הדומה לו. הערכת רמת הדמיון נעשית למעשה על בסיס המרחקים של תצפיות בתוך האשכול למרחקים בין האשכולות. נרצה למזער מדד זה, לכן הערך הטוב ביותר יתקבל כאשר הגרף יכיל אשכולות שמרכזיהם רחוקים זה מזה – בעוד התצפיות השייכות להן יהיו קרובות זו לזו.

בדומה למספר המחלקות הקיימות – ניתן לראות כי הערכים הטובים ביותר בשני המדדים הינו K=2.

בהתאם להנחיות הפרויקט, לאחר שראינו כי 2 הוא אכן מספר המחלקות האופטימלי על פי 2 המדדים, נבחר את האופציה השנייה בטיבה. על פי הגרפים אנו רואים שהיא חלוקה ל-3 קלאסטרים. ביצענו חלוקה מחדש של ה Labels, באמצעות פונקציית KBinsDiscretizer כך שכעת הפרמטר של הדליים הינו n\_bins = 3. בגרף הבא ניתן לראות את חלוקת הסאמפלים לשלושת האשכולות:

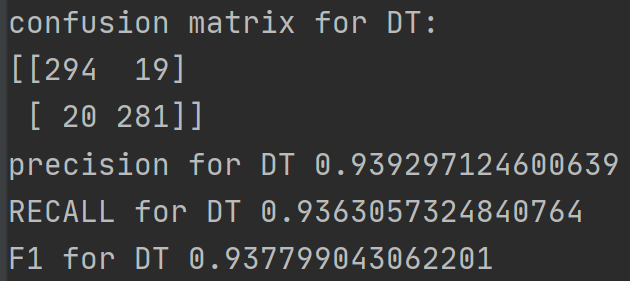
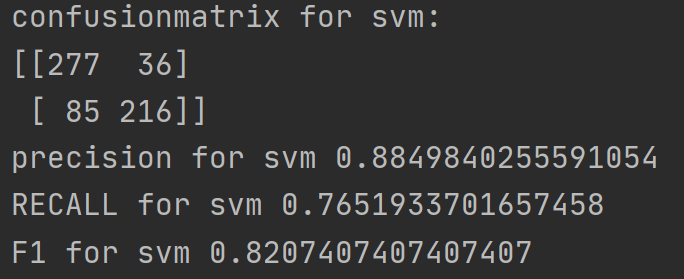
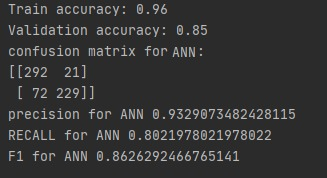
כאשר הוספנו את קווי ההפרדה הליניאריים של כל קלאסטר לפי הלייבל שלו קיבלנו את הגרף הבא:

נבדוק את טיב ההתאמה באמצעות **מדד purity**. למעשה, עבור כל אשכול נחשב את מספר המופעים של סוג התצפית שמופיעה בו מקסימום פעמים. זה ייתן אינדקציה ל"חיזוי נכון". נסכום את ה"חיזויים הנכונים" עבור כל ­­­אשכול ונחלק במספר התצפיות הכולל. ככל שערך המדד יותר גדול (מקסימום 1) כך ההתאמה טובה יותר. עבור 2 אשכולות: טיב ההתאמה בין האשכולות למחלקות הינו 0.49.

עבור 3 אשכולות: טיב ההתאמה בין האשכולות למחלקות הינו 0.35.

*­­­*קיבלנו כי הערך של מדד purity עבור k = 3 מעט נמוך יותר מערך המדד כאשר k = 2. נציין כי ייתכן שלפי מדדים שונים יתקבלו ערכים שונים לטיב התאמת המודל, ולכן זהו רק מדד אחד.

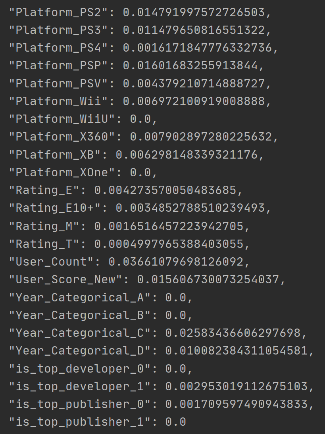
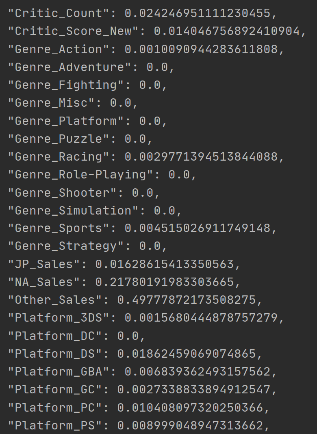
# השוואה בין המודלים – Evaluation

על מנת לבחור לפי מה להשוות את ביצועי שלושת המודלים (SVM, DT,MLP) פנינו למספר מאמרים הקשורים לנושא וחקרנו אודות ההבדלים בין המדדים. ראינו שלבעיות הקלסיפיקציה שלנו על פרדיקציה של משחקי מחשב, יטיב עימנו לחשב את precision & recall. לפי המאמרים, הבנו שדיוק וזיכרון הם שני מדדי הערכת מודלים חשובים ביותר. בעוד שדיוק מתייחס לאחוז התוצאות הרלוונטיות שלנו, זיכרון מתייחס לאחוז סך התוצאות הרלוונטיות המסווגות כראוי על ידי האלגוריתם שלנו. למרבה הצער, לא ניתן למקסם את שני המדדים הללו בו זמנית, מכיוון שאחד עולה מחיר של אחר ולכן מתבצע trade of בלתי נמנע ביניהם. עם זאת לאחר נבירה נוספת, מצאנו מדד נוסף הנקרא F-1 score, שהוא למעשה ממוצע הרמוני של דיוק וזיכרון. לבעיות בהן אין עדיפות לדיוק וזיכרון (כמו אצלנו), ניתן לבחור מודל שממקסם את ציון ה- F-1. כך F1 ייתן איזון בין Precision ו- Recall. לאחר ביישום המדד ב3 הקטגוריות התוצאות הן:

לפי מדד זה אנו רואים שהמודל בעל התוצאות הטובות ביותר הינו DT ! מסקנותינו אם כך כי מודל זה ממקסם את אחוז התוצאות הרלוונטיות שלנו ואחוז סך התוצאות הרלוונטיות המסווגות כראוי על ידי האלגוריתם שלנו. על כן, נמליץ להשתמש במודל הdesicion tree מבין שלושתם, לפי מדד זה.

\*מאמרים בנספחים

# שיפור המודל הנבחר



על מנת לשפר את המודל הנבחר (DT) אנו ראשית רצינו לדון שוב בחשיבות הפיצ'רים שיש לנו כיום לאחר כל השלבים שביצענו. כעת אנו יודעים יותר ויכולים לבצע מניפולציה אחרת לחלוקת המידע על מנת לנסות להטיב עם המודל. יצרנו dictionary על מנת לבחון את הפלטים של טיב הפיצ'רים. נרצה לנסות להוריד פיצ'רים שלא תורמים כלל על ידי איחוד קטגוריות מחדש. הדבר יגרום לפחות הסתעפויות בעץ, הקטנת הover fitting ובתקווה להעלאת אחוז הדיוק. לכן הפעולות שביצענו הן (תיעוד בנספחים):

1. איחדנו את כל הGenre שבמדד החשיבות שלהם יצאו 0 לקטגוריה אחת genre\_other)), בתקווה שביחד הם יקבלו משמעות וייעלו את נכונות הסיווג.
2. איחדנו את Year\_Categorical\_a ו Year\_Categorical\_b מאותה סיבה (שניהם 0 בחשיבותם)
3. ישנן 3 סמפלים בודדים בעלי קטגוריה שונה בפיצ'ר הRating שבתחילת העבודה התעלמנו מהם. בראייה לאחור הדבר שגוי והיינו רוצים להכניס אותם, ולכן איחדנו את שלושתם לעמודה חדשה הRating\_others.
4. בחלק זה ניתחנו את התוצאות לפי הידע שצברנו. חקרנו לראות מה הבעיות שיכולות להשפיע של אי דיוק מודל והגענו למושג **"קללת המימד" –Curse of dimensionality ,** שמדבר על כך שככל שהמימד יותר גדול (יותר פיצ'רים) זה ייצר משהו שהוא יותר over-fit לtrain, ולכן צריך להיזהר מיותר מידי ממדים. בניתוח המודל שיצא לנו, ראינו כי אמנם אין over-fitting מובהק כאשר האחוז דיוק של האימון הוא 97% והtest 94%, אך מצד שני עומק האופטימלי שיצא לנו הוא עומק של 13. בהסתכלות על הfeature importance, אנו רואים שיש הבדל מובהק בין מספר בודד של פיצ'רים, לבין לא מעט שלא תורמים כלל ולמעשה רק 2 מתוך 48 פיצ'רים תורמים מעל 0.2 אחוז! הדבר מרמז על כך שאין סיבה לעץ בעומק 13 עם המון התפצלויות ופיצ'רים כאשר הם בקושי תורמים. לכן שקלנו להסיר כמה מהם. אך בנגד, למרות האינטואיציה שלנו, החישוביזציה של הgrid\_search הראה אחרת כאשר במדד max\_features האלגוריתם בחר להשאיר את כולם (NONE)! לכן אנו מבינים שהורדה בהכרח לא תועיל כאן לדיוק.

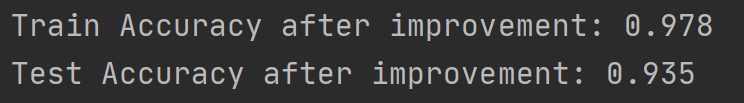
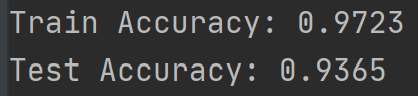
לכן, כעת המחשבה שלנו הייתי לנסות להוריד עוד יותר את הover\_fitting, על ידי כך שננסה למצוא פרמטרים שייתכן שאמנם את הדיוק של הtraining אך יעלו את הדיוק של הtest. בהתבוננות על תוצאת min\_samples\_leaf ראינו שמתוך טווח של 1-3 הערך הנבחר יצא 3 (המקסימלי) ולכן ראינו לנכון להגדיל את טווח הערכים בין 1 ל8, על מנת שאולי הגדלת המינימום בכל עלה בבסיס תוריד את המקריות של אירועים בודדים שיוצרים הסתעפות בעץ ותקטין את מורכבות העץ כפי שאנו רוצים.

את טווח העץ נבחר בין 4 ל20, נשאף שייצא קטן אך האלגוריתם יחליט מה מטיב עם הדיוק. בנוסף, בשביל השוואת המודל שבחרנו את f1, ורצינו גם לשאוף לשפר אותו פה על ידי השמת scoring = f1, אך מכיוון שאנחנו לא נמדדים עליו אלא על accuracy השארנו את הערך הדיפולטיבי שלו.

אם לא די בכך, דבר נוסף ששינינו הוא שינוי הcv מ10 ל5, שכן במקום שאלגוריתם ייאמן 10 פעמים עם 90 אחוז training ו10 אחוז test, כעת הוא ירוץ 5 פעמים בחלוקה של 80 training ו20 test. הדבר גם כן אמור להוריד את over\_fitting ולהועיל לדיוק הtest שלנו על חשבון הtraining.

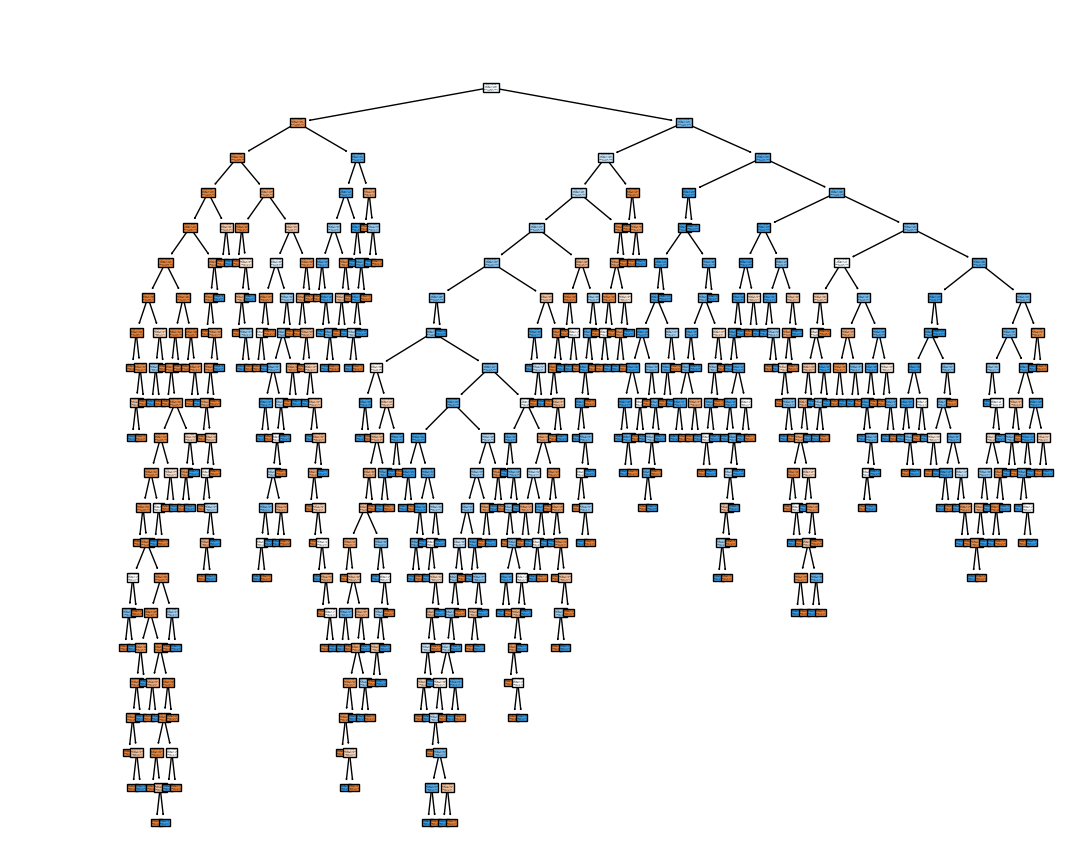
כאשר סיימנו עם כל שיפור המודל, לצערנו לא הצלחנו לשפר אותו על אף כל הנסיונות, אם כי אפשר להבין שהתוצאה לפני השיפור היא מאוד טובה ויהיה קשה לשפרה בקלות 😊.

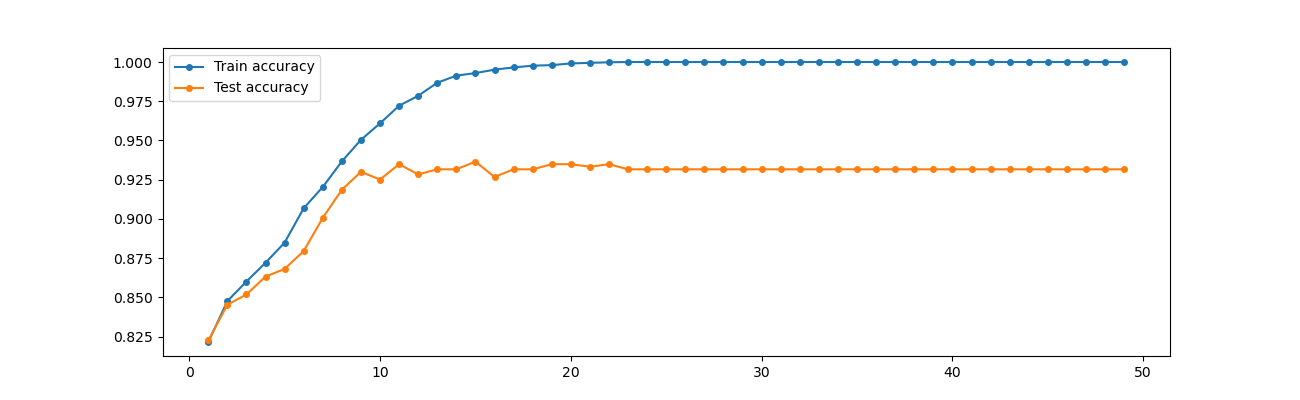
\*\*באופן כללי, בסעיף זה אנו מנסים לשפר את הaccuracy של המודל לפי הנתבקש ולכן בוחרים בפרמטרים שממקסמים את הדיוק, אך בעולם האמיתי יש עוד שיקולים ומדדים מלבד הaccuracy, הן במדדים אחרים והן בשיקולים כמו זמן ריצה ופשטות העץ. לדוגמא הרצנו לשם הבדיקה את אותם פרמטרים מלבד עומק העץ שהגבלנו ל7, והדיוק בtest ירד ל0.9. כאמור האלגוריתם לא יבחר בו, אך ייתכן כי פשטות העץ וההכללה הגדולה יותר שלו כעת כשהוא בעל עומק של 7 ולא 18 (כפי שהאידיאלי יצא) תועדף על ידינו, וייתכן כי במדדים אחרים אף יגבר על הקודם.



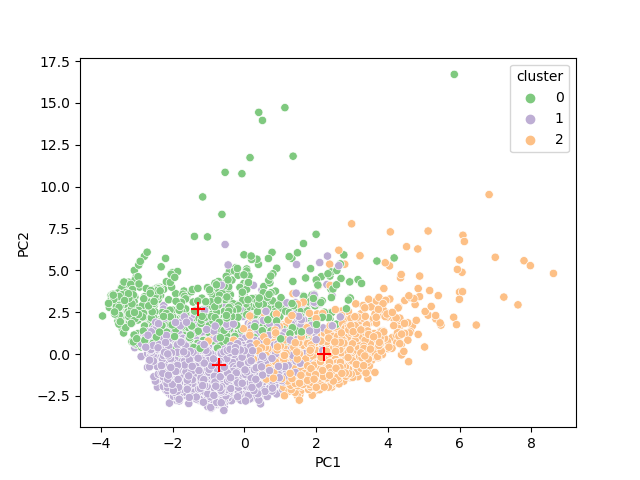
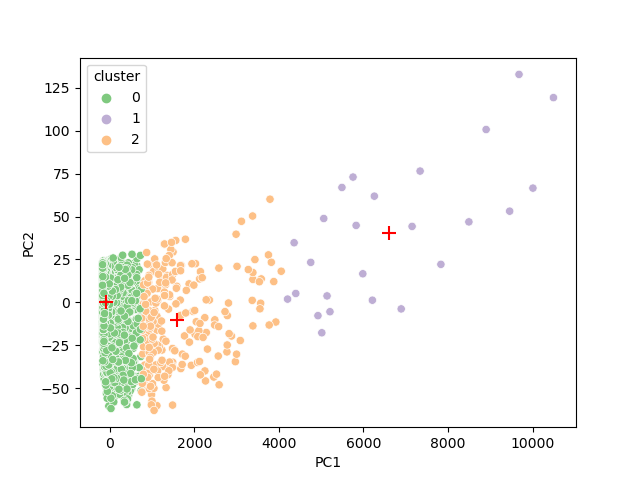
**נספחים:**

1. עץ החלטה מלא + גרף עומק עץ לפני היפר פרמטר



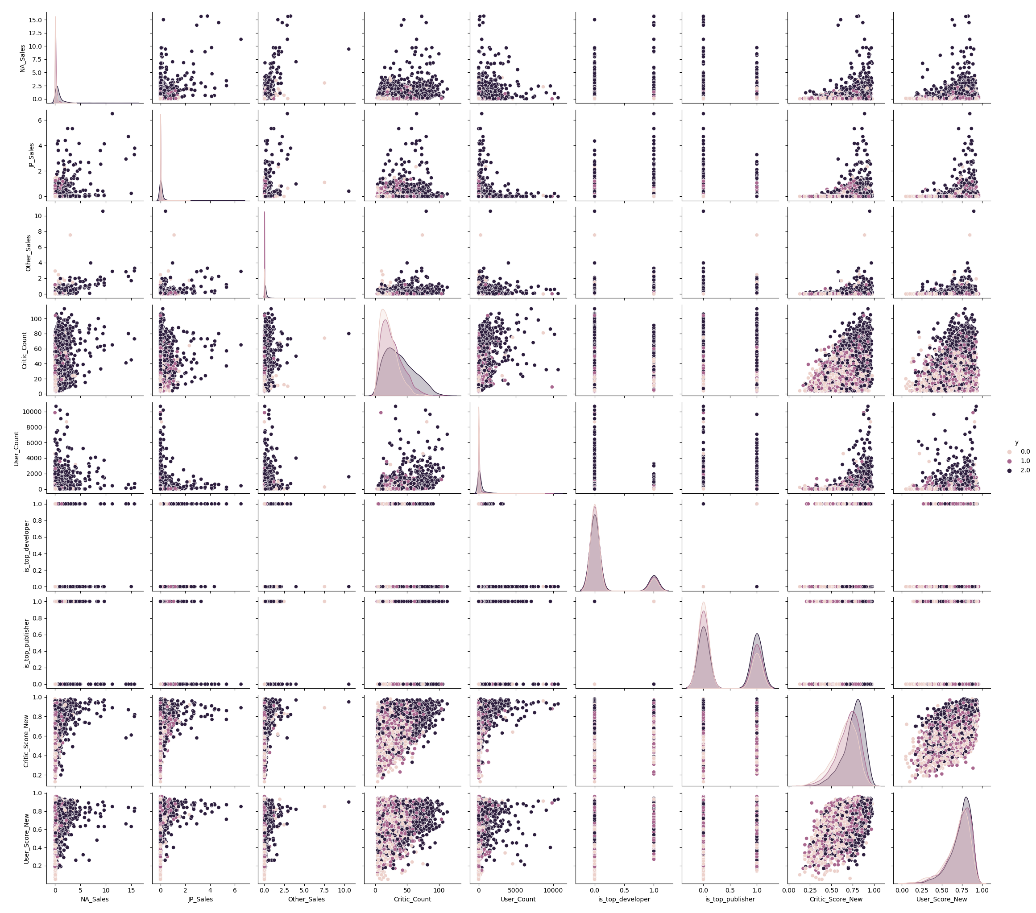
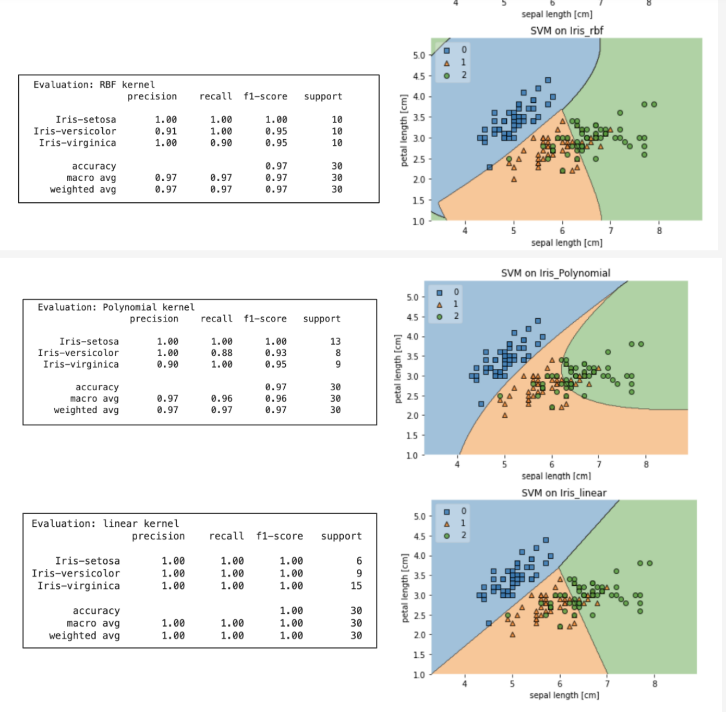


1. K means חלוקה ל3 לפני ואחרי הנרמול:

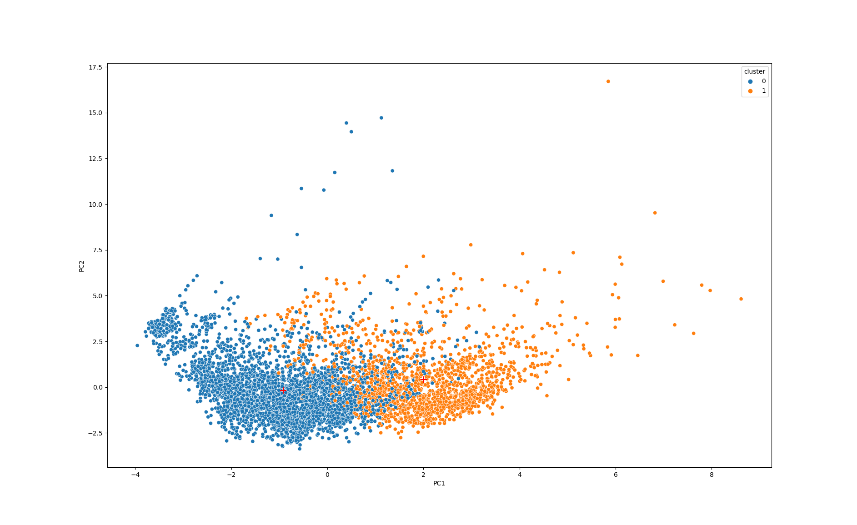
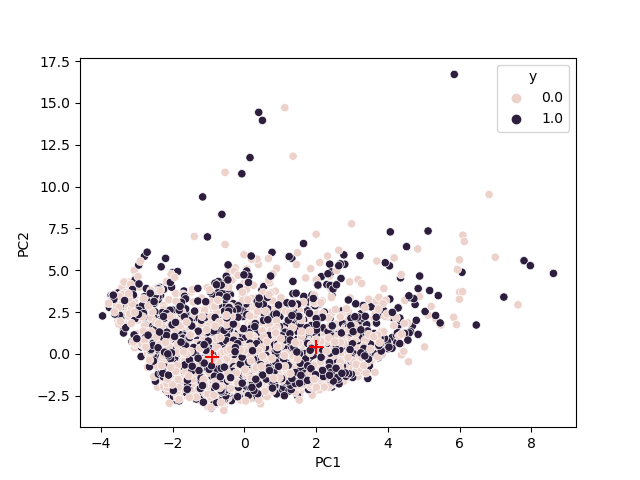


קלסטרים - חלוקה לאשכולות

שיטות אשכול שונות לרבי אשכולות (יותר מ2)



דוגמא לאשכול אל מול חלוקת הלייבלינג האמיתי

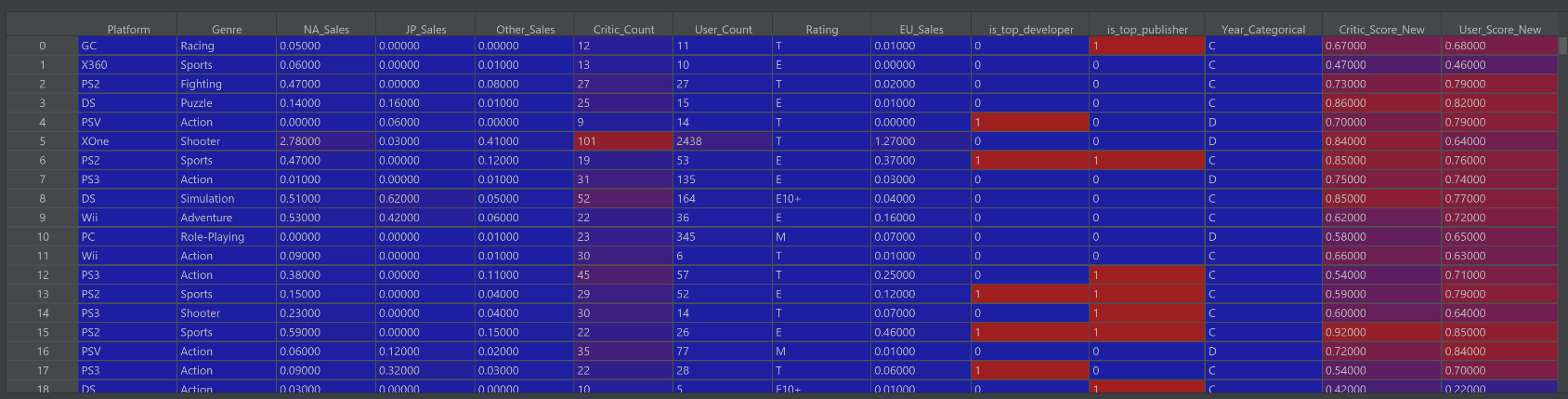




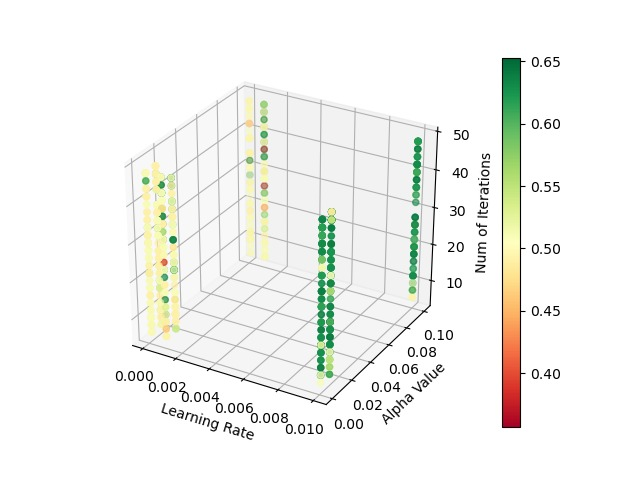
1. שיפורים- מאמרים לF1 + קוד פונקציות שיפורים

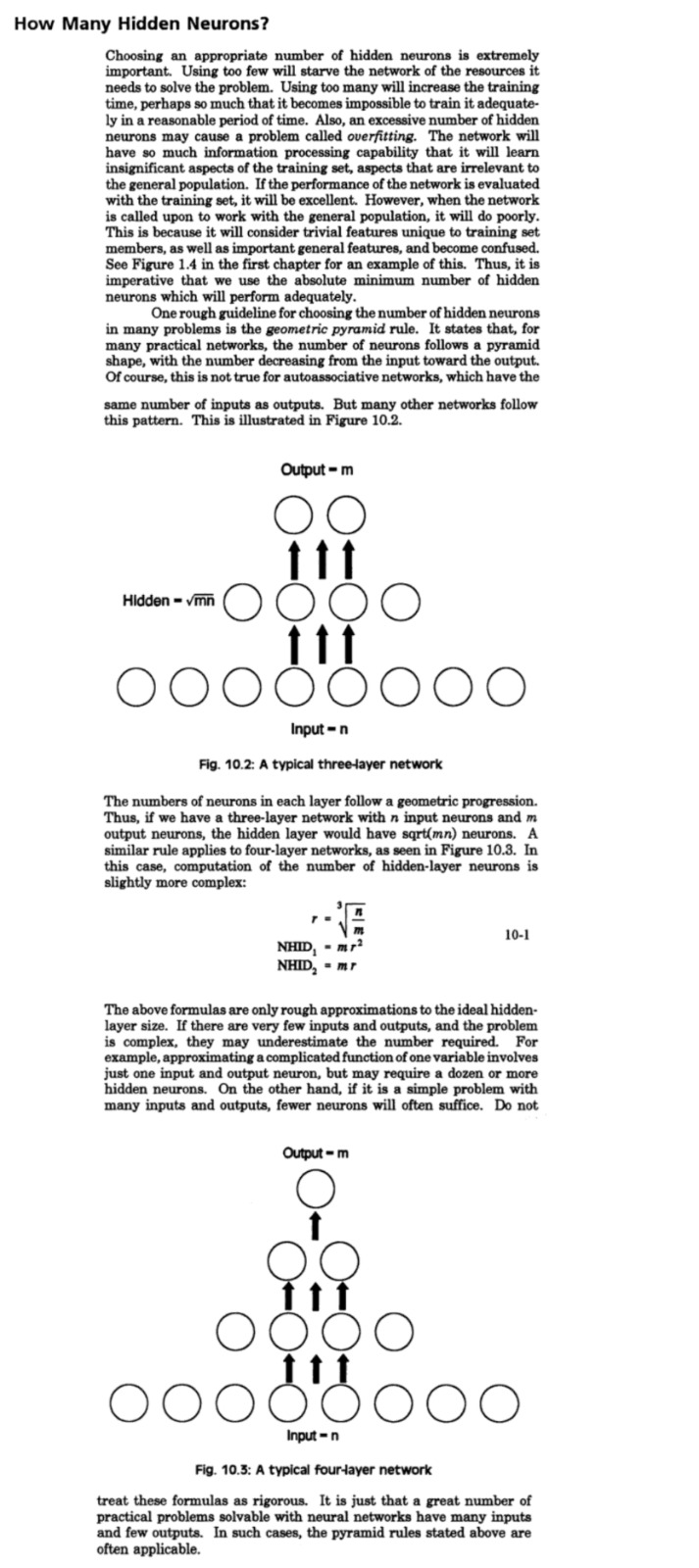
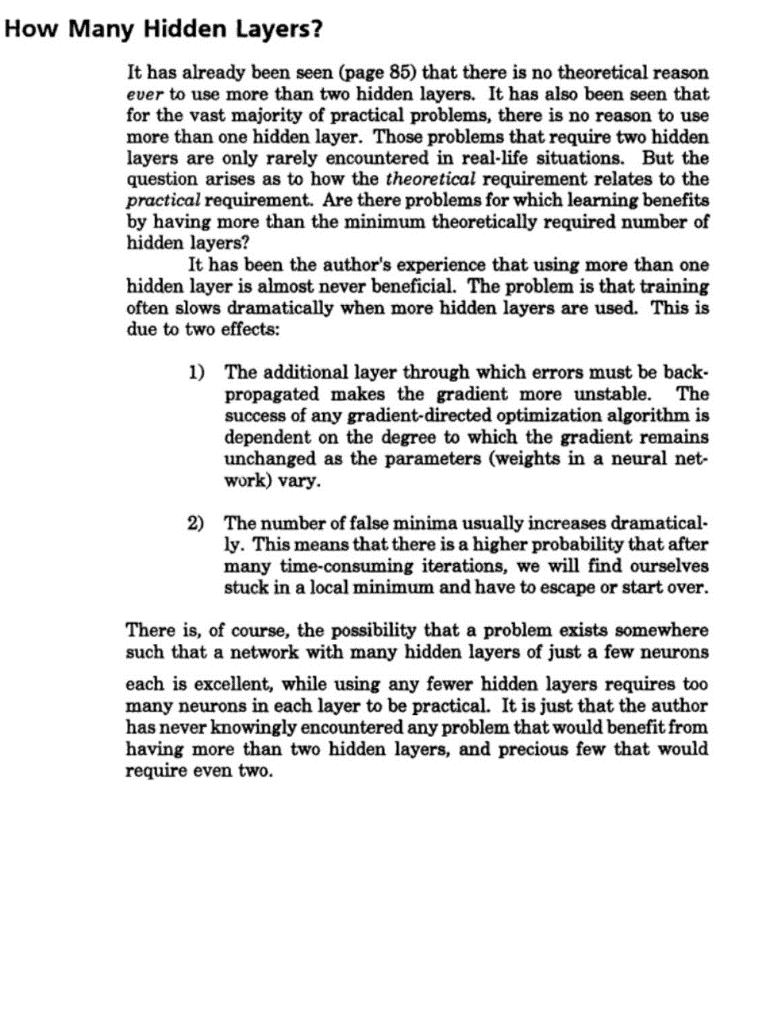
 Powers, David M W (2011). [*"Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation"*](https://www.researchgate.net/publication/228529307). Journal of Machine Learning Technologies. **2** (1): 37–63.

Buckland, Michael, and Fredric Gey. "The relationship between recall and precision." *Journal of the American society for information science* 45.1 (1994): 12-19.‏

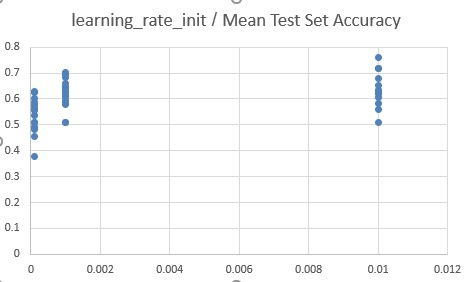
1. דוגמא לפיצ'רים לפני המניפולציה
2. נוירונים:

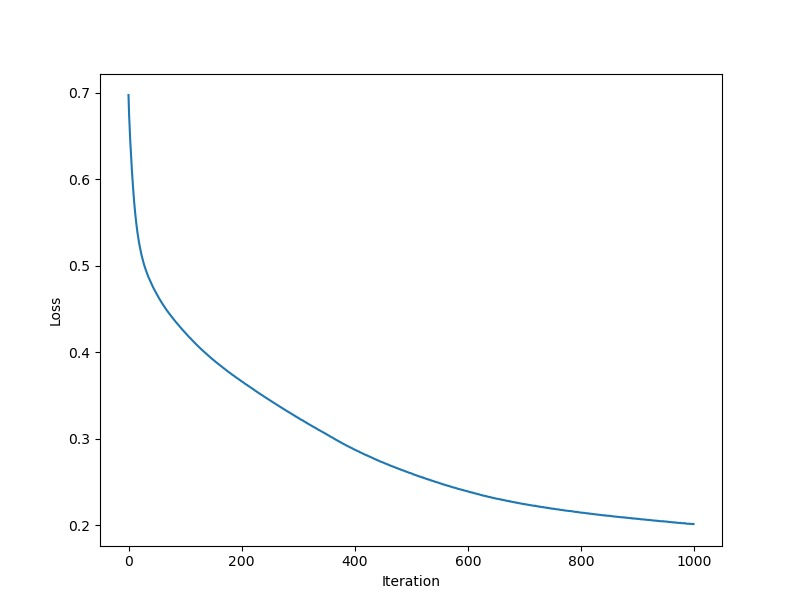
גרף 1 לrandom search





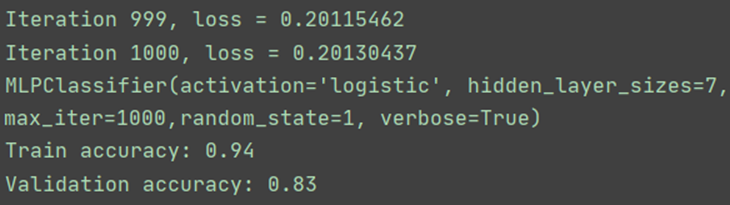
גרף 2 ו3 לrandom search





פלט לריצה ראשונית

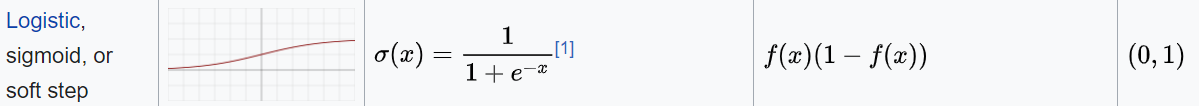
גרף loss ריצה ראשונית ברירת מחדל



פלט מודל נבחר







המאמר ממנו שאבנו את המידע:

T. Masters

(4th ed.), Practical Neural Network Recipes in C++, Academic Press, New York (1993)