אוניברסיטת בן-גוריון בנגב
המחלקה להנדסת תעשייה וניהול

לימוד מכונה – חלק ב'

עומר קידר

CONTENTS

MODEL TRAINING

הצגת ה **Dataset** שלנו לאחר בחירת הפיצ'רים מחלק א' :

Year_of_Release	Genre	Rating	Platform_General	General_Sales	Critic_Weight	User_Weight
0	Racing	T	Nintendo	0.05	0.234770354	0.023482367
1	Sports	E	Microsoft_Xbox	0.07	0.171414424	0.015145878
0	Fighting	T	Sony_Playstation	0.55	0.383691848	0.04274109
0	Puzzle	E	Nintendo	0.31	0.434956961	0.0330671
1	Action	T	Sony_Playstation	0.06	0.212420841	0.030777102

תזכורת לבחירת הפיצ'רים ניתן לראות (בנספח)

המרת משימת הלימוד למשימה **Classification**:

ביצענו המרה בעזרת פונקציית KBinsDiscretizer , כל מכירה באירופה שהייתה מעל החציון הומרה ל 1 והשאר ל 0.

- על מנת לבצע אימון למודל וכוונון פרמטרים, אנו נשתמש בשיטת K-fold cross-validation לצורך חלוקת הדאטה שלנו כפי שבחרנו בחלק א'.
- על הדאטה סט שהצגנו למעלה , נבצע מניפולציות מתאימות(כמו הפיכה למשתני דמה ונרמול) על מנת להתאים את הדאטה שלנו למודלים השונים אותם נבחן.

DECISION TREES:

על מנת להשתמש במודל של עץ החלטה , המרנו את כל המשתנים הקטגוריאליים למשתני דמה בעזרת הפונקציה get_dummies. ראשית בנינו עץ החלטה מלא עם ערכים דיפולטיביים , לאחר מכן ביצענו כוונון פרמטרים בעזרת אלגוריתם GridSearch ולבסוף בחרנו את הערכים שהניבו את המודל עם הדיוק הגבוה ביותר. בחרנו את את K להיות 10 משום שקראנו במספר מחקרים שזהו הערך הטוב ביותר לחלוקה ביחס לגודל דאטה דומה לשלנו וגם באופן כללי ועל-ידי כך נוצר סט אימון של 9 פולדים וסט ולידציה של סט אחד.

1. בניית עץ החלטה

```
Train accuracy    1.0
dtype: float64
Validation accuracy    0.750045
dtype: float64
```

עבור אימון עץ החלטה מלא קיבלנו את התוצאות בתמונה המצורפת. ניתן לראות כי דיוק המודל עבור סט האימון הוא 100% ועבור סט הולידציה 75%. מתוצאות אלו ניתן להבין כי הגענו למצב של Overfitting. העץ מתפצל ללא שום הגבלה עד העומק המרבי אליו הוא יכול להגיע וכך בעצם מגיע למצב בו הוא מסווג בצורה מושלמת את כלל הרשומות - כלומר המודל התאים את עצמו לסט האימון בצורה מושלמת. מצב זה הוא בעייתי משום שהמודל שלנו מותאם מידי לסט האימון והופך את המודל ללא גמיש. כאשר נבחן סט נתונים אחר עם המודל שלנו, המודל יספק תוצאות לא מספיק טובות. לגבי סט הולידציה אנו סבורים כי אחוז הדיוק שנקבל בהמשך ישתפר כתוצאה מכוונון הפרמטרים שנעשה שיהיו טובים יותר מהפרמטרים הדיפולטיביים ובנוסף המודל לא יהיה מאומן בצורה של Overfitting.

2. HYPERPARAMETER TUNING

לאחר שהרצנו את המודל הדיפולטיבי , ביצענו כוונון פרמטרים בעזרת GridSearchCV המשלב בצורה אוטומטית K-fold cross validation במטרה למצוא את הקונפיגורציה המיטבית עבור המודל שלנו, משום

שפונקציה זו עוברת על כל הקומבינציות האפשריות בטווח הערכים המוזן על ידינו. החלטנו לבחון את כוונון הפרמטרים : Criterion , Max depth ואלפא CCP. נציג על-ידי טבלאות שיצרנו בעזרת הפיתון ועל-ידי גרפים שהוצאנו את ערכי ההיפר פרמטרים שנבחנו כפונקציה של אחוז הדיוק על סט האימון וסט הולידציה.

הצגת גרפים וטבלאות

נציג כאן מספר מצומצם של רשומות מהטבלאות (טבלאות גדולות יותר וגרפים מופיעים בנספח).

Number	Parameters	Validation score
54	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 6}	0.884791
55	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 7}	0.881695
6	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 7}	0.88137
56	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 8}	0.880395

Number	Parameters	Train score
86	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 38}	1
90	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 42}	1
46	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 47}	1

מהטבלאות והגרפים שקיבלנו , כפי שגם טענו שקרה בעץ המלא שהרצנו , ניתן לראות כי על סט האימון , כאשר העץ מגיע לעומק גבוה (מספר פיצולים גדול) הוא מתאים את עצמו לסט האימון ומגיע לדיוק של 100% כלומר נמצא שוב במצב של Overfitting. כאשר הוא נמצא בעומקים נמוכים יותר הוא מתאים עצמו פחות לסט האימון ועל כן ניתן לראות שכאשר עומק העץ נמוך יותר התוצאות על סט הולידציה משתפרות ואחוזי הדיוק עולים.

ניתן לראות זאת בערכים האופטימליים שקיבלנו שהם:

```
The best parameters are: {'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 6}
```

בנוסף ניתן לראות שמקומבינציות מסויימות של ערכים (בעיקר גם בעצים רדודים מאוד) אנו מגיעים לערכי דיוק נמוכים מאוד ולמצב בו אנו במצב של Underfitting.

מוטיבציה לבחירת היפר הפרמטרים:

Max depth: עומק העץ , מאפיין על-פיו נבחר את הכמות המרבית של פיצולים בעץ. ככל שהעץ מתפצל יותר כך אנו עלולים להתקרב למצב בו יש Overfitting וכאשר העץ עם מספר פיצולים נמוך מאוד ישנה סכנה של Underfitting. לכן המוטיבציה שלנו לחפש את הערך האופטימלי של עומק העץ שיתן לנו גם את ההבנה לגבי המשתנים האינפורמטיביים ביותר ואלו שפחות וגם לגבי העומק הטוב ביותר שיניב תוצאה אופטימלית.

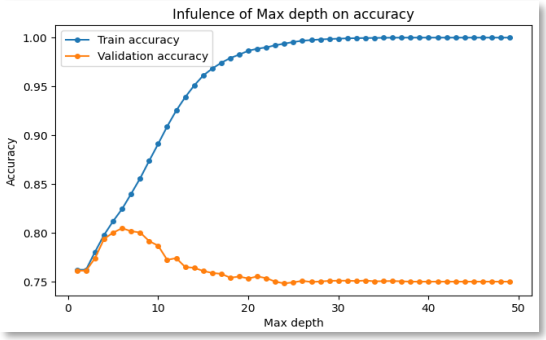
Criterion: קריטריון החישוב , ישנם שני קריטריוני חישוב שאותם בחנו , gini ו entropy , כל אחת מהשיטות מחשבת את האינפורמציה הנצברת בכל צומת בעץ בצורה שונה ועל-ידי כך מייצרת עצי החלטה שונים זה מזה. המוטיבציה שלנו הייתה למצוא את דרך החישוב הטובה ביותר אשר תוביל לעץ האופטימלי ביותר.

Ccp alpha: זהו פרטמר אשר מגדיר את הסף עבור ביצוע קטימה בעץ . האלפא עבור כל צומת נקבעת על-פי מובהקות הפיצול ולכן ככל שהאלפא נמוכה יותר הפיצול פחות מובהק ולפי כך ניתן להבין אם צריך לקטום את העץ באותו שלב או לא. המוטיבציה לכוונון פרמטר זה הייתה בכך שאנו משתמשים באלפא על מנת למנוע Overfitting ובכך נוכל לשמור על העץ אינפורמטיבי ומצד שני עץ בעל יכולת הכללה ולא מותאם לסט נתונים ספציפי.

המשמעות על העץ הנלמד כתוצאה מהגדלה / הקטנה של ערכי הפרמטרים:

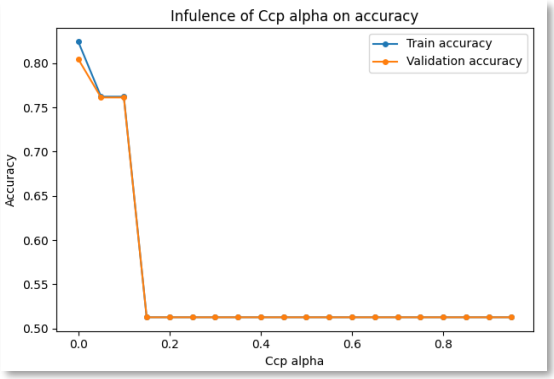
משום שהכוונון בוצע כתלות של הפרמטרים אחד בשני על ידי פונקציה מובנת , נשתמש בערכים האופטימליים שקיבלנו ונראה בכל פעם את ההשפעה של אחד או יותר מהערכים על המודל כאשר שאר הערכים אופטימליים.

השפעת עומק העץ על הדיוק:



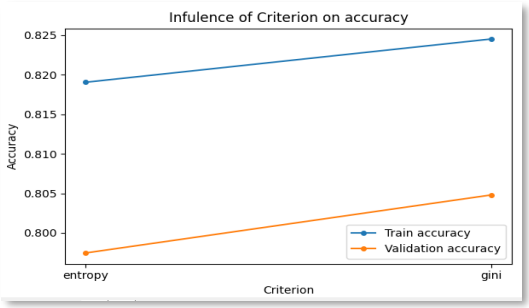
ניתן לראות כפי שכבר קיבלנו מהערכים האופטימליים , כי הדיוק הטוב ביותר מתקבל בעומק מקסימלי של 6. בנוסף ניתן לראות כי עבור עומקי עץ הגדולים מ 6 דיוק סט האימון ממשיך לעלות אך הדיוק על סט הולידציה יורד. מצב זה נובע מכך שהמודל מתאים את עצמו יותר ויותר לסט האימון (מה שצפוי שיקרה ככל שהעץ מתפצל יותר ויותר) ולכן כאשר אנו בוחנים את המודל על סט הולידציה אחוזי הדיוק שלו יורדים.
מכאן שעד עומד מקסימלי 6 ניתן להבין כי העץ מתפצל ונותן מידע אינפורמטיבי וחיוני אשר שומר על מצב בו אין Overfitting וכאשר העומקים גדלים מ6 אנו מתחילים להתקרב למצב זה.

השפעת Ccp alpha על הדיוק :



ניתן לראות כי האלפות השונות משפיעות בצורה משמעותית על דיוק המודל שלנו. המודל מניב דיוק מרבי כאשר אלפא שווה ל 0. בנוסף ניתן לראות כי גם סט האימון וגם סט הולידציה מושפעים בצורה דומה עד זהה מהשינוי באלפא ואחוזי הדיוק בשניהם יורדים עד לכדי 51% כאשר האלפא מגיע ל 0.15.

השפעת ה Criterion על דיוק העץ :



ניתן לראות כי שימוש ב 'gini' הניב אחוזי דיוק גבוהים יותר מ 'entropy' על סט הולידציה וגם על סט האימון ועל כן הוא נבחר לקריטריון האופטימלי שלנו למודל שנאמן .

3. אימון עץ ההחלטה עם הקונפיגורציה הטובה ביותר שהתקבלה

כפי שכבר הראנו הערכים האופטימליים על-ידם נאמן את העץ הם : Ccp alpha = 0 , Criterion='gini' , Max depth = 6.

תוצאות המודל שקיבלנו הם :

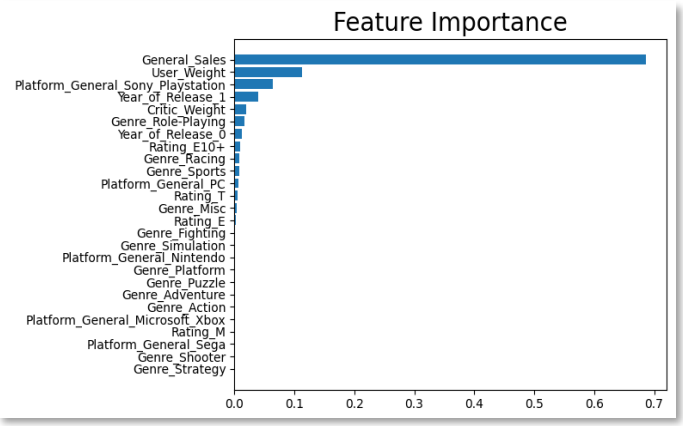
```
Train Best accuracy    0.824507
dtype: float64
Validation Best accuracy    0.804791
dtype: float64
```

מתוצאות אלו ניתן להסיק כי הדיוק המקסימלי על סט הולידציה מגיע כאשר דיוק סט האימון הוא 82.45% בלבד. כלומר אחוזי דיוק גבוהים על סט האימון לא מובילים בהכרח לתוצאות טובות יותר על סט הולידציה (או סט בחינה אחר) משום שאחוזי דיוק גבוהים על סט האימון ככל הנראה יובילו ל Overfitiing וכתוצאה מכך למודל לא גמיש ומותאם לסט האימון. השוני מסעיף 1 מתבטא במה שהסברנו מעלה ונגרם כתוצאה מכוונון הפרמטרים. כלומר כשסט האימון שלנו בעץ המלא הגיע לדיוק של 100% קיבלנו דווקא דיוק נמוך יותר על סט הולידציה של 75% לעומת דיוק של 80.479% על סט זה כאשר אחוזי הדיוק על סט האימון היו נמוכים יותר בערכים האופטימליים. (העץ המתקבל , קטום עד עומק 3 מוצג בנספח).

- אחד היתרונות של עץ החלטה הוא יכולת ההסברה שלו (Interpretability) , הכוונה בכך היא המידה בה אדם יכול להבין את הסיבות שהובילו להחלטה. כלומר בעץ, בכל צומת אנו יכולים להבין איזה פיצ'ר נבחר ולמה, ועל-פיו למה יסווג המודל שלנו. כל צומת נותנת מידע נוסף בעץ וניתן לראות את כל הדרך אשר הובילה לקבלת ההחלטה של העץ. עץ החלטה נותן מידע איכותי וברור לסיבות שהובילו להחלטתו. לכן עץ החלטה הוא בעל יכולת הסברה ופרשנות גובה.

על-פי מבנה העץ שלנו , ניתן ללמוד על הפיצ'רים האינפורמטיביים ביותר, על הערכים של כל אחד מהם ואיך העץ מחליט לסווג על פי כל אחד מהערכים הללו. ניתן לראות כי הפיצ'ר בעל ההשפעה הגבוה ביותר על המודל כפי שגם ציפינו שיהיה מהניתוחים בחלק א' הוא General Sales אשר מעיד על המכירות בעולם. דבר זה מתיישב והגיוני משום שאנו מנסים לסווג לגבי מכירות באירופה ולכן הגיוני שפיצ'ר זה ישפיע מאוד על קבלת ההחלטה בסיווג. גם ברמות הבאות של העץ ניתן לראות כי באמת פיצ'רים שראינו כי המתאם שלהם גבוה נבחרו מוקדם יותר User Weight ו Platform וסיפקו מידע אינפורמטיבי יותר וכאלו שפחות כמו למשל Year_of_Release נמצאים יותר עמוק בעץ.

מהפלט של Feature importance ניתן לראות את החשיבות של כל פיצ'ר בקבלת ההחלטות של עץ ההחלטה שבנינו. הפיצ'ר האינפורמטיבי ביותר הוא General_Sales וניתן לראות שהוא משפיע בצורה הכי משמעותית על העץ בפער די גדול. הפיצ'רים החשובים פחות הם דווקא Rating ו Genre שהמספקים הכי פחות אינפורמציה למודל שלנו. מסקנות אלו מתיישבות עם המסקנות מהסעיף הקודם. ניתן לראות כי שורש העץ מתחיל ב General_Sales שהוא האינפורמטיבי ביותר, לאחר מכן הפיצול השני הוא של User Weight ושל Platform_sony_playstaion שהם הבאים בדירוג וכך הלאה בהתאמה.



ARTIFICIAL NEURAL NETWORKS

עבור מודל זה השתמשנו בסט הנתונים שבו השתמשנו עבור מודל עץ ההחלטה, כאשר המשתנים הקטגוריאלים מיוצגים על ידי משתני דמה. בנוסף, ביצענו נירמול מסוג MIN-MAX על הפיצ'ר

'General_Saless' כאשר 0=min ו1=maxi בכדי שטווח הערכים בכל פיצ'ר יהיה 0-1. (נזכיר כי יש לנו פי'צרים רציפים אחרים שכבר ביצענו להם נרמול בחלק א(בנספח)

1. אימון ובחינת הרשת בערכי ברירת מחדל:

משמעות הקונפיגורציה

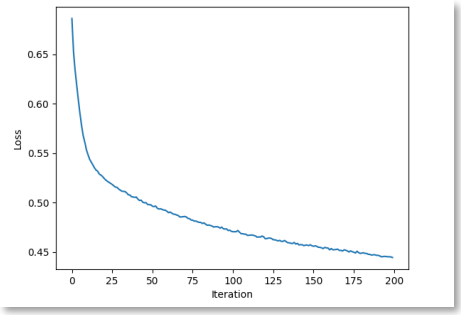
הערכים הדיפולטיים של קלאספייר MLP הם:

מספר נוירונים בשכבת הכניסה:26 מספר שכבות חבויות:1

מספר נוירונים בשכבה החבויה:100 מספר נוירונים בשכבת היציאה:2

מספר הנוירונים בשכבת הכניסה תמיד יהיה כמספר הפי'צרים שהמודל מקבל, במקרה שלנו מדובר על 26 פיצ'רים. מספר הנוירונים בשכבת היציאה במקרה שלנו הוא 2 - כמספר אופציות הקלסיפיקציה של המודל (סיווג בינארי). מספר השכבות החבויות וכמות הנוירונים בכל שכבה מגדיר את מספר יחידות העיבוד לכל מאפיין. ככל שהמודל עמוק ורחב יותר הוא ילמד בצורה מעמיקה ומדוייקת יותר, אך זה עלול גם ליצור מצב של overfitting ולכן המצב הזה לא תמיד רצוי.

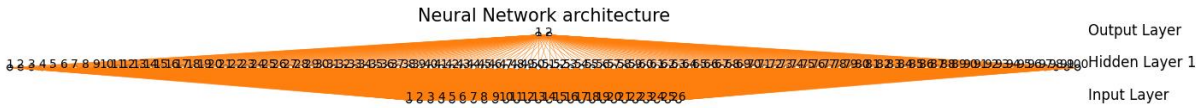
תוצאות דיוק ריצה ראשונית



הריצה הראשונית הייתה כאמור לפי הערכים הדיפולטיים של MLP. כיוון שעבור validation השתמשנו בשיטת K-Fold, הרצנו את המודל 10 פעמים ועשינו ממוצא על תוצאות המודל על ה-train ועל ה-validation.

Train accuracy: 0.782738 Validation accuracy: 0.757209

ניתן לראות כי המודל אינו over-fitted ואנו מקבלים תוצאות יחסית דומות בין האימון והוולידציה.



2. HYPERPARAMETER TUNING:

הצגת ממצאים

בשלב זה ביצענו GridSearch על מנת למצוא את הפרמטרים האופטימליים עבור מודל זה. כמו בכל העבודה, החלוקה ב cross-validation הייתה עבור 10 פולדים. הפרמטרים אותם כוונו הם Hidden layer size, maximum iterations, learning rate, solver. בחרנו לא לבדוק פונקציית אקטיבציה שונה מ-RelU כיוון שלידיעתנו היא הפונקציה המקובלת ביותר בתעשייה כיום. בנוסף, כיוון שלא נגענו בשיטות regularization בקורס, בחרנו שלא לבחון מודל עמוק מדי ולכן בדקנו עבור שכבה אחת חבויה או שתיים בלבד. ביצענו הרצה ראשונית כדי לסנן ערכים כך שנוכל למקד את כוונן הפרמטרים ולהגיע לתוצאה הטובה ביותר. מצאנו כי sgd

נתן תוצאות גרועות מאוד ביחס ל- adam ו lbfgs.התוצאות הטובות ביותר התקבלו עבור 500 אפוקים (ערך מקסימלי לפרמטר זה) ועם שכבה חבויה אחת או שתיים בעלות 7 נוירונים(נספח).

Best parameters: 'hidden_layer_sizes': (7,), 'learning_rate_init': 0.001, 'max_iter': 500, 'solver': 'lbfgs'

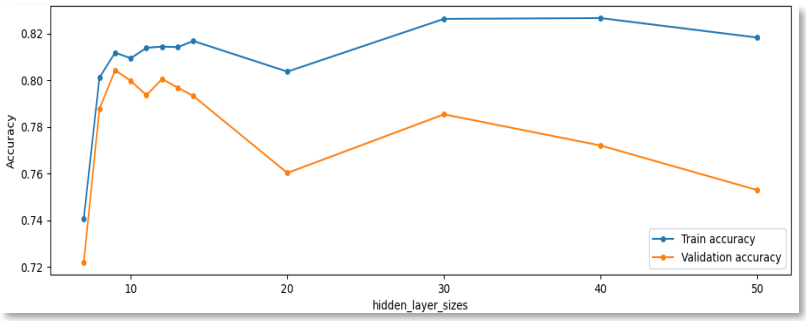
לאחר מכן ביצענו הרצה נוספת ממקודת יותר סביב התוצאות הללו וקיבלנו: (נספח).

Best parameters: 'hidden_layer_sizes': (10,), 'learning_rate_init': 0.001, 'max_iter': 500, 'solver': 'lbfgs'

לבסוף, לאחר הגעה לסט הערכים הטוב ביותר, ביצענו בדיקות נוספת עם טווח ערכים רחב יותר לכל פרמטר בנפרד כדי לבדוק האם אפשר לשפר עוד יותר את המודל. הפרמטר learning_rate לא נבדק כיוון שהוא קבוע עבור lbfgs (ה-solver שנבחר כטוב ביותר). (נספח).

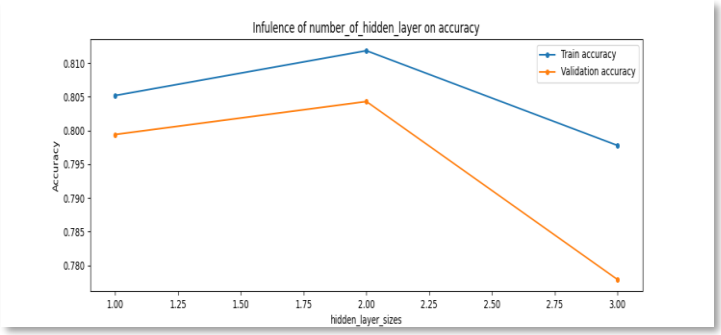
Best parameters: 'hidden_layer_sizes': (9, 9), 'learning_rate_init': 0.001, 'max_iter': 1000, 'solver': 'lbfgs'

השפעת מספר הנוירונים בשכבה החבויה על דיוק המודל (2 שכבות חבויות):



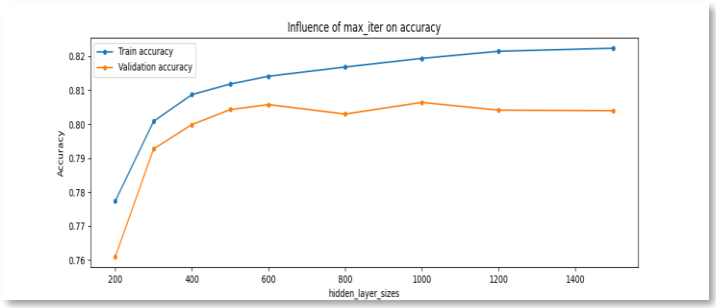
ניתן לראות כי הדיוק בהתחלה יחסית דומה בין train וה-validtation ולאחר מכן נפתח פער גדול ככל שמספר הנוירונים עולה. הסיבה לכך היא שככל שיש יותר נוירונים בשכבה, המודל מתאים עצמו יותר טוב לסט האימון וכך נוצר overfitting. התוצאה הטובה ביותר מתקבלת עבור 9 נוירונים בכל שכבה חבויה.

השפעת מספר השכבות החבויות על דיוק המודל:



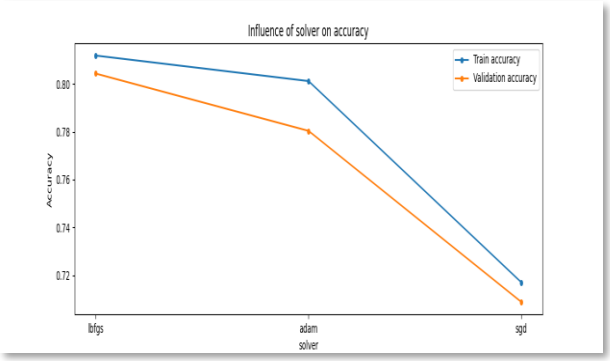
בדומה להשפעה של כמות הנוירונים בכל שכבה, בהתחלה יש דימיון בתוצאות train וה-validation והמודל משתפר במעבר משכבה אחת לשתיים, אך כשמגיעים כבר ל3 שכבות דיוק המודל צונח. ייתכן והסיבה לכך היא שעבור מודל עמוק יותר, יש צורך בכוח חישוב גדול יותר, כלומר במספר נוירונים גדול יותר בכל שכבה או בחלק מן השכבות. כאמור, בדיקה של 3 שכבות חבויות לא נבדקה ביחס למספר הנוירונים בעבודה זו. התוצאה הטובה ביותר היא עבור 2 שכבות חבויות.

השפעת מספר האיטרציות על דיוק המודל:



ניתן לראות כי עבור מספר האיטרציות, יש עליה כמעט עקבית בדיוק המודל עד לנקודה בה הוספה של איטרציות נוספות משפרות רק את דיוק ה-train אך מורידות את ה-validation (כלומר נוצר overfit). התוצאה הטובה ביותר מתקבלת עבור 1000 איטרציות.

השפעה של ה- solver על דיוק המודל:



גרף זה איננו מפתיע כיוון שראינו לכל אורך תהליך הtuning כי lbfgs היה הסולבר העדיף על פני השאר. כמובן כי ניתן היה לקבל תוצאות טובות יותר עבור sgd ו adam עם פרמטרים אחרים, אך לכל אורך התהליך לא מצאנו קונפיגורציה שהניבה תוצאות עדיפות על lbfgs.

היפר-פרמטרים

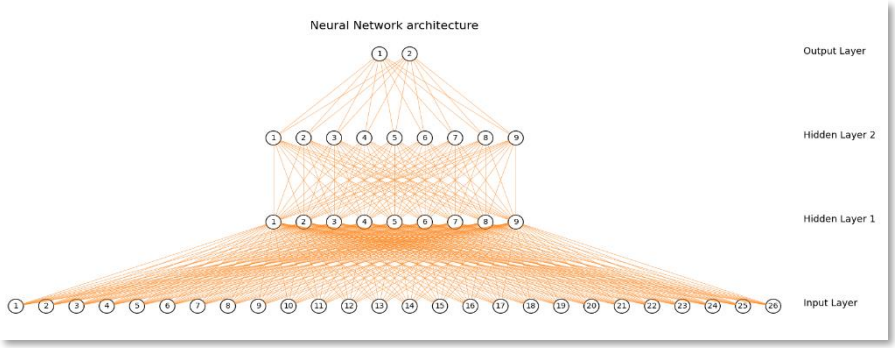
שם	מוטיבציה	משמעות
Hidden layer size	הגדרת מספר השכבות החבויות ומספר הנוירונים בכל שכבה משפיעה באופן מובהק על דיוק המודל.	ככל שמספר הנוירונים בשכבה החוביה עולה, כך גם רמת הדיוק משתפרת עד לנקודה בה נוצר overfit ודיוק הtrain מתייצב אך דיוק הtest מושפע לרעה. מספר רב של נוירונים גם יעלה את סיבוכיות החישוב של המודל.
Maximum iterations	כמות גבוהה יותר של אפוקים יכולה להניב תוצאות גבוהות יותר.	ככל שמספר האיטרציות עולה כך גם הדיוק עד התייצבות מסוימות. אולם מספר גבוהה מדי מייצר overfit. מספר גדול יותר איטרציות מגדיל את סיבוכיות החישוב.
Learning rate init	פרמטר זה מגדיר את הקצב שבו הרשת מעדכנת את המשקולות, הוא בין הפרמטרים החשובים ביותר בכיוון מודל ANN.	בהתאם לסולבר האידאלי שלנו, קצב הלמידה קבועה ולכן שינוי שלו לא ישפיע על תוצאות המודל. עבור סולברים אחרים, קצב הלמידה משפיעה על ההפרש בשיפור במודל עבורו המודל מפסיק את הלמידה. עבור קצב נמוך יותר, סיכויות הריצה תעלה.
Solver	השוני בין הסולברים יכול לתת תוצאות שונות בהתאם לסוג הדאטה וגודלו. כיוון שגודל הדאטה שלנו מצד אחד אינו קטן (מספר אלפים) אך גם לא ענק (לא בעשרות, מאות, או יותר של אלפים) בחרנו לבדוק את כולם כדי לגלות מה הכי מתאים לנו.	כיוון שכל סולבר עובד בצורה שונה ומגדיר את הדרך להגעה למשקל אופטימלי אחרת, הם נותנים תוצאות שונות על סוגי דאטה ודורשים כיוונון פרמטרים שונה.

הקונפיגורציה הסופית האופטימלית והתוצאות:

ניתן לראות כי המודל הסופי שלנו נותן תוצאות דומות מאוד בין ה-train וה-validation. תוצאה זו אינה מפתיעה היות וכוונו את הפרמטרים במודל כן שיתנו דיוק מקסימלי לסט הוולידציה, וימנעו overfitting. ההבדל בתוצאות מהמודל הראשוני הוא עליה בדיוק והקטנת השונות בין סט האימון וסט הוולידציה. גם במקרה זה, ההסבר לכך הוא שההיפר-פרמטרים נבחרו בדיוק למטרה זו - הגדלת הדיוק של המודל ללא overfitting.

Hidden layer size	(9, 9)
Maximum iterations	1000
Learning rate init	0.001
Solver	lbfgs

```
train_acc     0.819384
dtype: float64
val_acc     0.806418
dtype: float64
```



3. אימון הרשת עם קונפיגרציה אידיאלית

כדי למצוא את ארבעת המשחקים שעבורם המודל היה "הכי פחות בטוח בתוצאותיו" אימנו את המודל על כל סט האימון ואז בדקנו אותו והוצאנו מטריצת הסתברויות של השכבה האחרונה ולקחנו מכל עמודה (0, 1) את שני המשחקים שההסתברות שלהם להשתייך לעמודה הייתה הכי קרובה ל-0.5 מלמעלה. התוצאה:

```
The games that our model what the least sure about classifying as 0:
BioShock ,  Space Chimps

The games that our model what the least sure about classifying as 1:
Tiger Woods PGA Tour 09 All-Play ,  Need For Speed: Undercover
```

SVM:

אימון מודל SVM דיפולטיבי - ללא כוונון :

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=42, shrinking=True, tol=0.001,
    verbose=False)
Train accuracy     0.752503
dtype: float64
Validation accuracy     0.749553
```

1. HYPERPARAMETERS TUNING

החלטנו להציג את כוונון הפרמטרים רק על ההפרמט C וזאת בגלל סעיפי המשך הדורשים להציג את משוואת הקו המפריד (בפועל מודל SVM מסוג 'rbf' יצא בעל הדיוק הגבוה ביותר של כ 78%). מכאן שהמודל האופטימלי שנבחר הוא מודל SVM לינארי עם C=100 שהניב 75.4115% דיוק על סט הולידציה.

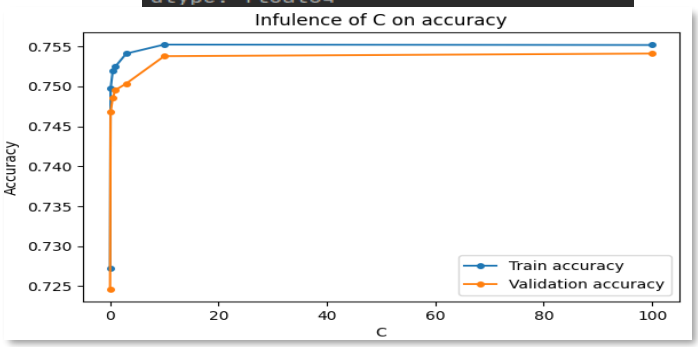
```
The best parameters are: {'C': 100}
```

Number	Parameters	Validation score
6	{'C': 100}	0.754115
5	{'C': 10}	0.753789
4	{'C': 3}	0.750367
3	{'C': 1}	0.749553
2	{'C': 0.5}	0.748575
1	{'C': 0.1}	0.746781
0	{'C': 0.01}	0.724616

Number	Parameters	Train score
5	{'C': 10}	0.755237
6	{'C': 100}	0.755183
4	{'C': 3}	0.754114
3	{'C': 1}	0.752503
2	{'C': 0.5}	0.751906
1	{'C': 0.1}	0.749715
0	{'C': 0.01}	0.727228

אחוזי דיוק על המודל הנבחר :

```
Train Best accuracy     0.755183
dtype: float64
Validation Best accuracy     0.754115
dtype: float64
```



השפעת כוונון C על המודל:

בערכי **C** קטנים מאוד המודל ממשיך לעלות , מגיע לערך כמעט מקסימלי ב **C**=10 ואחר מכן ניתן לראות שישנה עוד עליה קטנה ל **C**=100 ולכן שם עצרנו משום שהשפיעו כבר לא משמעותי ואף מתחיל לרדת בערכים גדולים יותר.

משוואת הישר המפריד:

בעזרת הפלטים מהפיתון הוצאנו את המשקלים של כל הפיצ'רים שלנו ואת ערך החותך, מהם נבנה את משוואת הישר המפריד שלנו:

```
The Weights of the Features are:
[General_Sales , Critic_Weight , User_Weight , Year_of_Release_0 , Year_of_Release_1 , Genre_Action ,
 Genre_Adventure , Genre_Fighting , Genre_Misc , Genre_Platform , Genre_Puzzle , Genre_Racing ,
 Genre_Role-Playing , Genre_Shooter, Genre_Simulation , Genre_Sports , Genre_Strategy , Rating_E
, Rating_E10+ , Rating_M , Rating_T , Platform_General_Microsoft_Xbox , Platform_General_Nintendo , Platform_General_PC
 , Platform_General_Sega ,Platform_General_Sony_Playstation ]
[[ 2.62725726  1.51408867  6.83998635 -0.25894209  0.25894209  0.10554074
  -0.07654567  0.05520021  0.12578513  0.17128639 -0.47778271  0.45420181
  -0.61824628  0.18605482  0.15248122  0.06365147 -0.14162713  0.11068064
  -0.10510547  0.00761266 -0.01318783  0.26440869  0.25360052 -0.09341175
  -1.13031909  0.70572163]]
The intercept is:
[-2.16746119]
```

לוקטור של המשקלים שקיבלנו נקרא W ולחותך שלנו נקרא b. מכאן שהישר המפריד שלנו הוא:

$$W*x - b = 0 \Rightarrow W*x - 2.16746119 = 0$$

ממשוואת הישר ניתן להבין את החשיבות של כל פיצ'ר למודל ה SVM על-פי וקטור המשקלים. ניתן לראות כי הפיצ'ר המשפיע ביותר על המודל הוא User_Weight , אחריו General_Sales ואז Critic_Weight , אלו הם שלושת הפיצ'רים המשפיעים ביותר. לעומת מסקנות קודמות, למשל בעץ, קיים שוני מסויים בין חשיבות הפיצ'רים שכן בעץ ההחלטה הפיצ'ר המשפיע ביותר היה General_Sales , בנוסף ניתן לראות שינויים גם בגודל ההשפעה של פלטפורמות שונות לעומת אחרות לעומת מה שהיה בעץ ההחלטה. באופן כללי כן ניתן להגיד שהפיצ'רים שהכי משפיעים על המודלים השונים די קבועים אך הסדר שלהם משתנה מעט. השינויים מתיישבים עם ההגיון משום שכל מודל פועל בדרכי החלטה אחרות ומבצע חישובים שונים ולכן הגיוני שיהיו שינויים כלשהם בסדר הפיצ'רים אך עדיין תשמר מגמה מסויימת קבועה של הפיצ'רים המשפיעים יותר ופחות.

מה היינו עושים במידה והיינו רוצים לבצע משימת קלסיפיקציה עם 3 מחלקות:

ב SVM ניתן לבצע משימות קלסיפיקציה עם מס' מחלקות הגדול מ2 על-ידי שימוש בשיטה הנקראת One Vs Rest. בשיטה זו כל קלאס משווה מול כל שאר הקלאסים האחרים וכך נוצר הסיווג עם יותר מ 2 מחלקות.

ניתן לממש זאת בעזרת `decision_function_shape='ovr'`

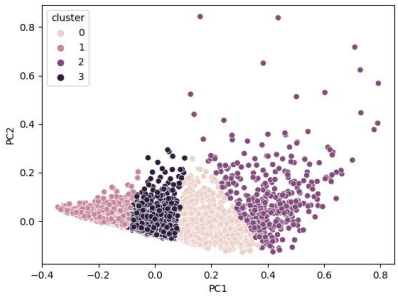
UNSUPERVISED LEARNING - CLUSTERING

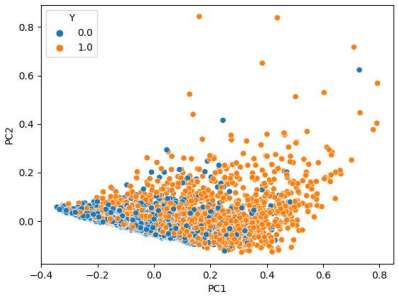
1. הרצת אלגוריתם K-MEANS

סט הנתונים שהשתמשנו בו לצורך הרצת האלגוריתם בתור התחלה הוא אותו סט נתונים שהשתמשנו עבור אלגוריתם ANN, כאשר הפי'צרים הקטגורים הומרו בשיטת one-hot-encoding והמשתנים הרציפים נורמלו לערכי 0-1. כמובן שהיות ואלגוריתם K-means הוא אלגוריתם של unsupervised learning, הדאטה נלקח ללא הלייבלים (ללא עמודת EU_Sales). כדי להבין האם יש דרך יעילה יותר להשתמש באלגוריתם פנינו למידע ברחבי האינטרנט לגבי שימוש של משתנים קטגוראלים בדאטה לשימוש ב-Kmeans וגילינו כי הדבר לא אידאלי. בנוסף, כפי שראינו במעבדה רצוי לבצע הורדת מימד על הנתונים ל-2 מימד בשביל שיהיה ניתן לעשות ויזואליזציה לאלגוריתם. שיטת להורדת מימד - PCA, אינה כדאית על דאטה קטגוריאלי לרוב, וזה בהחלט היה ניכר במקרה שלנו שכן הרצנו בדיקה וראינו שאחוז השונות המוסברת בהורדת כל הנתונים ל2 מימדים הוא כ 33% בלבד.

```
[0.18761364 0.13962275]
0.32723638831536594
```

לכן, אנו מסיקים מכך שיהיה נכון יותר להריץ את האלגוריתם רק על דאטה הרציף שלנו, לאחר שנבצע רק עליו הורדת מימד. לכן הסרנו את הדאטה הקטגוריאלי והורדנו למימד 2 על ידי PCA. כעת אפשר לראות כי באמת אחוז השונות המוסברת גבוהה (כ-98%). ניתן לראות גרף פיזור של הדאטה כעת (מתויג) ודוגמה להרצה של האלגוריתם עם k=4





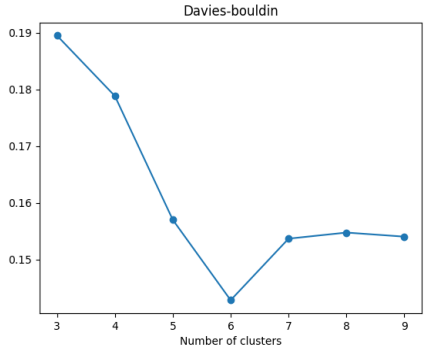

```
[0.8814357  0.10237079]
0.9838064908047754
```

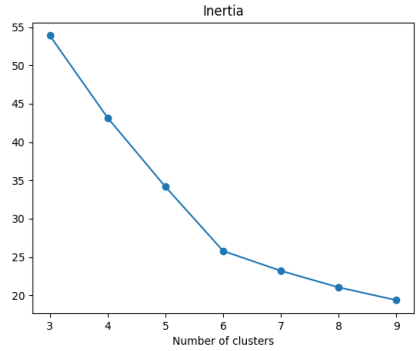
2. השוואת ערכי K

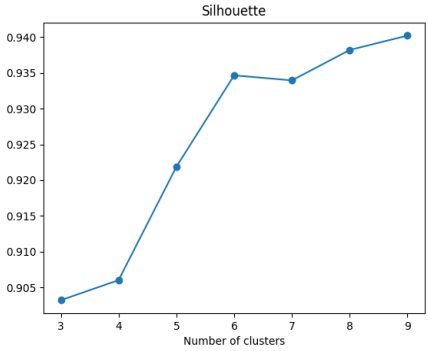
יצאנו מנקודת הנחה כי עבור סעיף זה צריך להניח שהlabel האמיתי של הדאטה לא ידוע. לכן, בחרנו להשתמש בהערכת מספר הקלאסטרים על פי מדד David-Bouldin measure אשר למדנו בהרצאה. בדקנו ערכי K של 3-9. בנוסף השתמשנו בשיטת k++ בביצוע המודל. פירוט של התוצאות ניתן לראות בנספח . התוצאה הטובה ביותר התקבלה עבור k=6.

```
For k= 6 , Davies-Bouldin score is: 0.14286902138001809
```

מדד זה נע בין 0-1 כאשר 0 זו התוצאה הטובה ביותר, לכן אנו שבעי רצון מתוצאה זו. כדי לוודא את התוצאה, החלטנו לבדוק מדד נוסף אשר ראינו במעבדה - שילואט. גם עבור מדד זה k=6 קיבל ציון טוב (0.935) אך הציון הטוב ביותר התקבל עבור k=9. בתקווה שנצליח "לשבור שיוויון", בחנו את מדד inertia אשר גם ראינו במעבדה. עבור מדד זה, התוצאה הטובה ביותר התקבלה עבורk=9 . ניתן לראות את סיכום התוצאות, ובעקבות תוצאת מדד inertia, אנו נבחר ב- 9 מחלקות.







3. בחינת מודל על פי חלוקת ה- CLUSTER

חילקנו את הדאטה של ל-9 מחלקות שונות, בדומה לחלוקה שביצענו בתחילת העבודה עבור 2 (דוגמה: 1/9 המשחקים שנמכרו הכי פחות הם לייבל 0, 1/9 המשחקים שנמכרו הכי הרבה הם לייבל 8). בחרנו שרירותית לבחון עבור מודל ANN. תוצאות הדיוק אשר קיבלנו הן:

```
train_acc    0.417993
```

```
val_acc    0.397107
```

ניתן לראות כי החלוקה החדשה הרעה מאוד את המודל ותוצאות הדיוק צנחו בחצי. אנו משערים שהסיבה לכך היא שעבור מספר גדול יותר של לייבלים, הרשת זקוקה לכמות שכבות או כמות נוירונים גבוהה יותר כדי להגיע לתוצאות טובות.

EVALUATION

על מנת להשוות בין המודלים שלנו בחרנו להשתמש ב Precision , TPR , FPR. המדד העיקרי להשוואה שבחרנו הוא Precision . מדד זה בוחן את היחס בין דגימות חיוביות אמיתיות לבין אלו שתויגו כחיוביות על ידע המסווג. מדד זה מחושב על-ידי הנוסחה הבאה:

$$Precision = \frac{TP}{TP + FP}$$

הסיבה שבחרנו להשוות בין המודלים על-ידי מדד זה היא משום שנרצה למקסם את הדיוק שלנו , כלומר כאשר אנו מסווגים משחק להיות משחק "רב מכר" אנו רוצים לצדוק ולא לסווג משחקים שאינם "רבי מכר" ככאלו שהם כן ובעצם לסווג FP. מדד זה מתבסס על דיוק גבוה בסיווג.

התוצאות שקיבלנו מחישוב מדד זה מראות כי מודל DT הוא הטוב ביותר מבין השלושה.

	SVM	DT	MLP
0	0.769052	0.811644	0.774743

על מנת להיות בטוחים בבחירת המודל שלנו בחרנו לבדוק את קטגוריות - FPR ו TPR.

התוצאות שקיבלנו היו :

```
FPR of SVM is:  FPR    0.38857
TPR of SVM is:  TPR    0.559285
```

```
FPR of DT is:  FPR    0.426802
TPR of DT is:  TPR    0.628634
```

```
FPR of MLP is:  FPR    0.425128
TPR of MLP is:  TPR    0.628584
```

TPR הוא מדד שמודד את שיעורים החיוביים שסווגו כחיוביים , FPR הוא מדד אשר מודד את ה False alarms. במדדים אלו אנו שואפים למקסם את מדד TPR ולמזער את FPR . ניתן לראות מהתוצאות כי במודל ה SVM הערכים של שני המדדים הללו הם הנמוכים ביותר ובמודלים של DT ו MLP מדדים אלו כמעט זהים. בשקלול של שלושת הקטגוריות שעל-פיהם בחנו את המודלים השונים , כאשר אנו מתחשבים בצורה המרבית ב Precision נרצה לבחור את עץ ההחלטה להיות

המודל שלנו. בחירה זו מתיישבת לנו גם עם אחוזי הדיוק שאותם כבר הראנו לכל מודל בשאלות הקודמות. ראינו כי מודל DT ומודל MLP נותנים לנו אחוזי דיוק כמעט זהים 0.8064 ל MLP לעומת 0.80479 ל DT ולכן הקטגוריות שבדקנו עזרו לנו לבחור כי המודל העדיף הוא DT. לא בחרנו במודל ה SVM לרמות שה FPR שלו הנמוך ביותר כי גם ה TPR שלו נמוך מהאחרים וגם אחוזי הדיוק ו ה precision של המודל פחות טובים.

IMPROVEMENTS

המודל הנבחר שלנו הוא DT. תחילה בחרנו לחזור ל data שלנו ולבצע שינויים אשר חשבנו שיעזרו לשפר את יכולות הסיווג של המודל שלנו. עוד בחלק א' ביצענו מספר שינויים שלא היינו בטוחים באשר לתרומתם למודל ולכן חזרנו על מנת לבדוק האם שינויים אלו דווקא פגעו במודל. גילינו לאחר מספר ניסויים על הדאטה כי הפיכת משתנה Year_of_Release לבינארי (אחרי ולפני שנת 2008) פגע במודל שלנו , בנוסף גילינו כי גם הפיכת Platform לכללי פגע במקצת באחוזי הדיוק של המודל , והפגיעה העיקרית בדיוק המודל הייתה בעקבות איחוד המכירות מהמקומות השונים בעולם לאחד. איחוד המכירות מראש בחלק א' העלה בנו חשש כי יכול להיות שיפגע בדיוק משום שמכירות במקומות שונים דווקא יכולים להעיד על מכירות במקום אחר ואילו האיחוד שלהם מאבד מידע באופן מסויים , החלטנו בחלק א' למרות זאת לאחד משום שהמתאם יצא גבוה יותר אך הסתבר בניסיון השיפור בחלק זה כשגוי. הפיצ'רים שלנו לאחר השינויים נראה כך :

Platform	Year_of_Release	Genre	NA_Sales	JP_Sales	Other_Sales	Critic_Weight	User_Weight	Rating
GC	2002	Racing	0.05	0	0	0.234770354	0.023482367	T
X360	2009	Sports	0.06	0	0.01	0.171414424	0.015145878	E
PS2	2004	Fighting	0.47	0	0.08	0.383691848	0.04274109	T
DS	2007	Puzzle	0.14	0.16	0.01	0.434956961	0.0330671	E
PSV	2014	Action	0	0.06	0	0.212420841	0.030777102	T
XOne	2015	Shooter	2.78	0.03	0.41	0.853921214	0.329028446	T
PS2	2003	Sports	0.47	0	0.12	0.37477754	0.05760871	E
PS3	2013	Action	0.01	0	0.01	0.422396139	0.089523123	E

ביצענו כוונון פרמטרים מחדש על בגלל החלפת הדאטה שלנו וקיבלנו כי העץ האופטימלי מתקבל כאשר:

```
The best parameters are: {'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 9}
+----------+-------------------------------------------------------------------------+--------------------+
|   Number | Parameters                                                              |   Validation score |
+==========+=========================================================================+====================+
|       17 | {'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 9}                 |           0.932867 |
+----------+-------------------------------------------------------------------------+--------------------+
```

```
Train Best accuracy     0.96618
dtype: float64
Validation Best accuracy     0.932867
dtype: float64
```

ותוצאות המודל החדש היו 96.66% על סט האימון ו 93.286% דיוק על סט הולדיציה לעומת 80.04% דיוק על סט הולדיציה עם הדאטה הקודם. שיפור אדיר שעיקרו כ- 10% בדיוק ,עלה בגלל פיצול המכירות למכירות המקוריות.

בחרנו לנסות לשפר את המודל שלנו עוד בעזרת boosting , הרעיון של שיטה זו היא הפיכת לומדים חלשים ללומד חזק יחיד. כלומר לקחת פיצ'רים שלא מספקים מידע רב למודל ולהפוך אותם למשמעותיים וכאלו שיעזרו למודל לקבל החלטות טובות יותר ובכך יעלו את אחוזי הדיוק של המודל. עשינו זו בעזרת פונקציה מובנת לעצי החלטה - GradientBoostingClassifier . ביצענו על שיטה זו גם כוונון פרמטרים נוסף מעבר לפרמטרים של העץ שכבר יש לנו כמו עומק 9 וקיבלנו את התוצאה הטובה ביותר עבור: (טבלה מלאה בנספח)

ניתן לראות כי המודל שלנו השתפר מ93.286% ל **95.22%%**.

	learning_rate	n_estimators	Train Accuracy	Validation Accuracy
52	0.25	100	1	0.952254

למרות שעל סט האימון אחוז הדיוק הוא 100 , נראה כי אנו לא במצב של OverFitting משום שאחוזי הדיוק על סט הולידציה עלו והמודל עדיין מצליח לסווג בצורה מעולה על סט הולידציה. בשלב זה החלטנו לעצור את ניסיון השיפור משום שלא מצאנו דרכים נוספות לשפר אותו מעבר לאחוזי הדיוק שהגענו.

הגשת חיזויים סופיים

מצורף קובץ החיזויים בפורמט CSV בשם y_test לקבצי הפרויקט שהוגשו.

נספחים:

תזכורת לפיצ'רים שנבחרו מחלק א':

הפיצ'רים איתם בחרנו להישאר מחלק א' הם :

Year_of_Release - שנת השחרור של המשחק , 1-אחרי שנת 2008 ו 0 לפני

Genre- ג'אנר המשחק

Rating- למי מיועד המשחק

Platform_General- פלטפורמת המשחק - איחדנו את הקונסולות תחת פלטפורמה כללית(למשל 2PS ו 4PS תחת Sony playstaion).

General_Sales- איחוד המכירות מהעולם שקיבלנו בדאטה המקורי למכירות כוללות.

Critic_Weight- שילוב של Critic_Score ו Critic_Count מהדאטה המקורי על פי נוסחה שהצגנו בחלק א'.

User_Weight- שילוב של User_Score ו User_Count מהדאטה המקורי על פי נוסחה שהצגנו בחלק א'.

DECISION TREES:

<u>גרף וטבלאות לערכי הפרמטרים שנבחנו כפונקציה של אחוז הדיוק (אימון וולידציה):</u>

שמנו תמונות של הערכים החשובים שמראים את השינויים החדים - עליהם גם הרחבנו בהסברים למעלה.

טבלאות של סט הולידציה:

Number	Parameters	Validation score
54	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 6}	0.804791
55	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 7}	0.801695
6	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 7}	0.80137
56	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 8}	0.800395
53	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 5}	0.800227
4	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 5}	0.799089
5	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 6}	0.797458
7	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8}	0.796809
8	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 9}	0.794364
52	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 4}	0.793872
57	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 9}	0.791596
3	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 4}	0.789965

39	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 40}	0.748741
40	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 41}	0.748578
34	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 35}	0.748415
25	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 26}	0.748414
30	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 31}	0.748252
72	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 24}	0.74825
32	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 33}	0.747598
1430	{'ccp_alpha': 0.7000000000000001, 'criterion': 'gini', 'max_depth': 10}	0.512631
1424	{'ccp_alpha': 0.7000000000000001, 'criterion': 'gini', 'max_depth': 4}	0.512631
1425	{'ccp_alpha': 0.7000000000000001, 'criterion': 'gini', 'max_depth': 5}	0.512631
1426	{'ccp_alpha': 0.7000000000000001, 'criterion': 'gini', 'max_depth': 6}	0.512631
1437	{'ccp_alpha': 0.7000000000000001, 'criterion': 'gini', 'max_depth': 17}	0.512631
1427	{'ccp_alpha': 0.7000000000000001, 'criterion': 'gini', 'max_depth': 7}	0.512631
1436	{'ccp_alpha': 0.7000000000000001, 'criterion': 'gini', 'max_depth': 16}	0.512631

טבלאות של סט האימון :

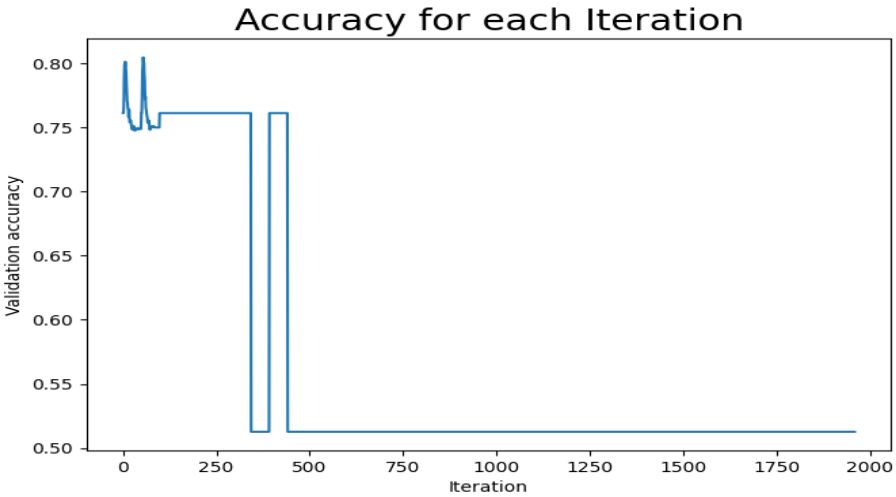
Number	Parameters	Train score
86	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 38}	1
90	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 42}	1
46	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 47}	1
47	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 48}	1
48	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 49}	1

15	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 16}	0.951152
62	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 14}	0.950971
14	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 15}	0.942588
61	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 13}	0.939004
13	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 14}	0.930929
60	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 12}	0.925189
12	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 13}	0.919142
59	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 11}	0.909003
11	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 12}	0.905147
58	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 10}	0.891116
10	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 11}	0.890663
9	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 10}	0.874912

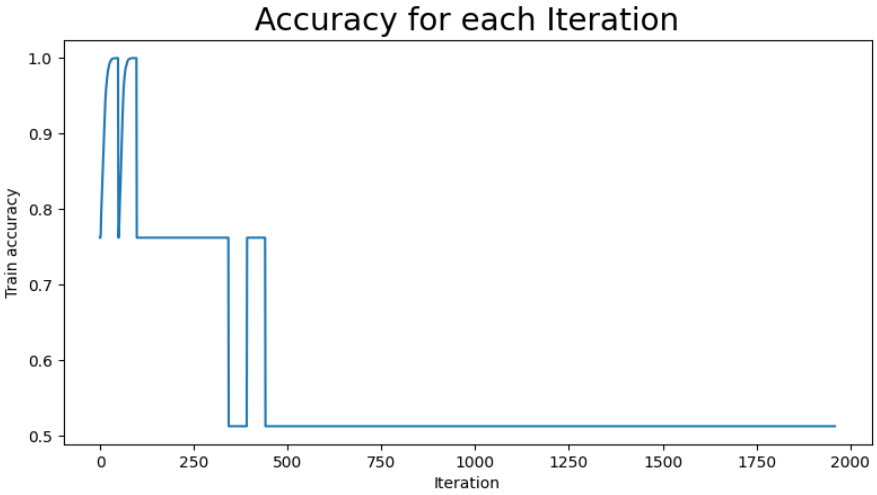
1029	{'ccp_alpha': 0.5, 'criterion': 'gini', 'max_depth': 1}	0.512628
1051	{'ccp_alpha': 0.5, 'criterion': 'gini', 'max_depth': 23}	0.512628
1050	{'ccp_alpha': 0.5, 'criterion': 'gini', 'max_depth': 22}	0.512628
1049	{'ccp_alpha': 0.5, 'criterion': 'gini', 'max_depth': 21}	0.512628
1048	{'ccp_alpha': 0.5, 'criterion': 'gini', 'max_depth': 20}	0.512628
1047	{'ccp_alpha': 0.5, 'criterion': 'gini', 'max_depth': 19}	0.512628
1046	{'ccp_alpha': 0.5, 'criterion': 'gini', 'max_depth': 18}	0.512628

גרפים:

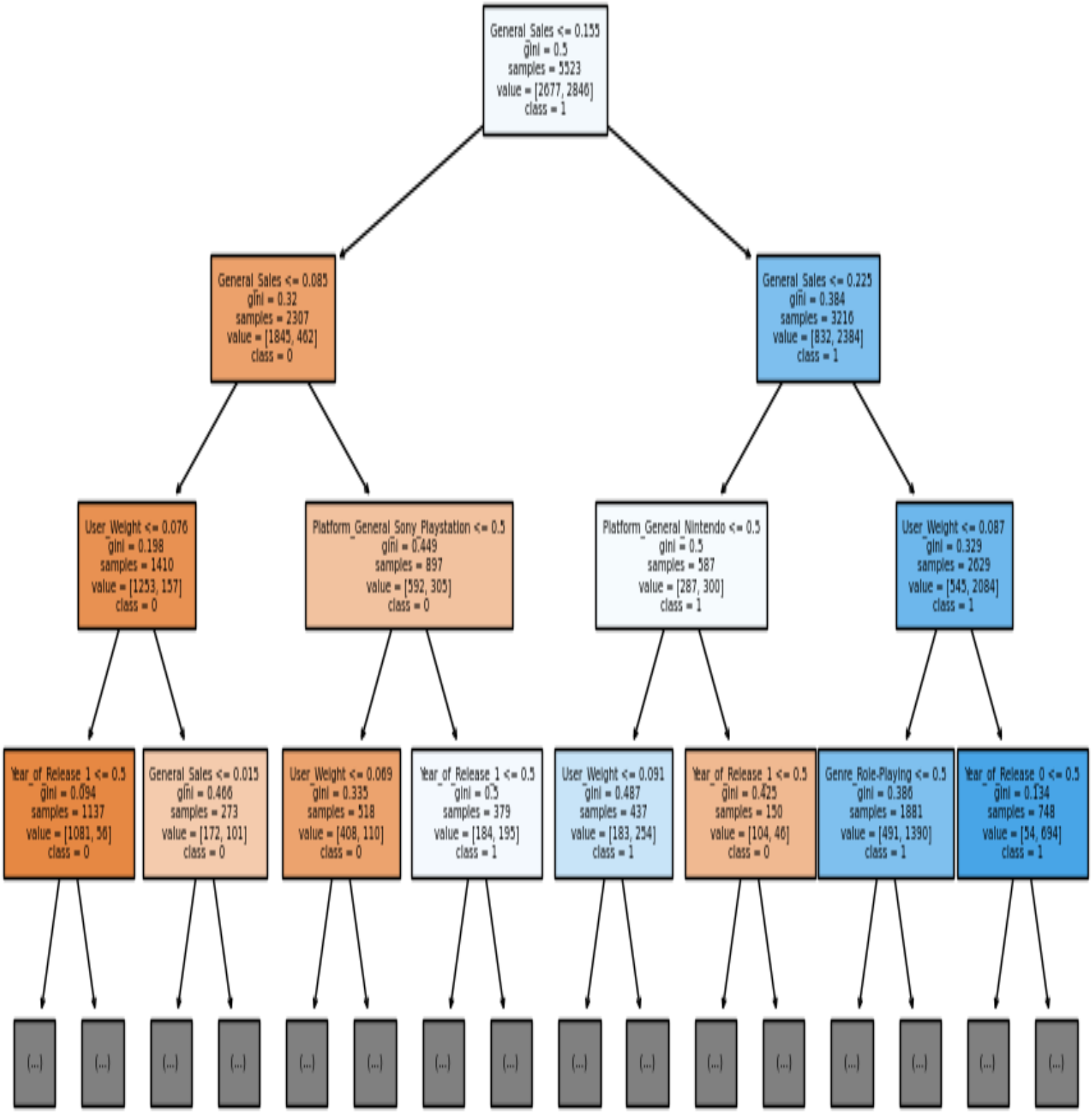
גרף של סט הולידציה:



גרף של סט האימון:



עץ החלטה מאומן על-פי הערכים האופטימליים אחרי כוונון(הוגבל לעומק 3 כדי שניתן יהיה לראות בבירור) :



ANN

נרמול פי'צרים שנעשה כבר בחלק א'.

D. Feature Representation

שלב זה הוא השלב בו נחליט כיצד לייצג את פיצ'רים שחילצנו בשלב ה Feature Extraction. הפיצ'רים הראשונים שנייצג יהיו הפיצ'רים שייצרנו מהפיצ'רים Critic Score ו User Score בשילוב עם Critic Count ו User Count. הפיצ'רים המקוריים היו בסקלות אחרות של 0-10 ו 0-100 ולכן גם הפיצ'ר החדש שחילצנו User_Weight ו Critic_Weight הם לא בעלי אותה סקלת ערכים. מכאן שנרצה לבצע נרמול ולהביא את שני הפיצ'רים החדשים שחילצנו לסקלה דומה בין 0-1. נעשה זאת על-ידי חלוקה של כל ערך בערך המקסימלי האבסולוטי על פי הנוסחה הבאה: $X = \frac{x}{\max(|x|)}$

תוצאות היפרטיונינג ראשון:

Number	Parameters	Train score
310	{'hidden_layer_sizes': (40, 40), 'learning_rate_init': 0.01, 'max_iter': 500, 'solver': 'adam'}	0.851303
307	{'hidden_layer_sizes': (40, 40), 'learning_rate_init': 0.01, 'max_iter': 250, 'solver': 'adam'}	0.851303
394	{'hidden_layer_sizes': (50, 50), 'learning_rate_init': 0.01, 'max_iter': 500, 'solver': 'adam'}	0.851285
391	{'hidden_layer_sizes': (50, 50), 'learning_rate_init': 0.01, 'max_iter': 250, 'solver': 'adam'}	0.851285
388	{'hidden_layer_sizes': (50, 50), 'learning_rate_init': 0.005, 'max_iter': 500, 'solver': 'adam'}	0.850597
385	{'hidden_layer_sizes': (50, 50), 'learning_rate_init': 0.005, 'max_iter': 250, 'solver': 'adam'}	0.850379
304	{'hidden_layer_sizes': (40, 40), 'learning_rate_init': 0.005, 'max_iter': 500, 'solver': 'adam'}	0.842866
301	{'hidden_layer_sizes': (40, 40), 'learning_rate_init': 0.005, 'max_iter': 250, 'solver': 'adam'}	0.842395
226	{'hidden_layer_sizes': (30, 30), 'learning_rate_init': 0.01, 'max_iter': 500, 'solver': 'adam'}	0.841146
223	{'hidden_layer_sizes': (30, 30), 'learning_rate_init': 0.01, 'max_iter': 250, 'solver': 'adam'}	0.841146
397	{'hidden_layer_sizes': (50, 50), 'learning_rate_init': 0.025, 'max_iter': 250, 'solver': 'adam'}	0.833614
400	{'hidden_layer_sizes': (50, 50), 'learning_rate_init': 0.025, 'max_iter': 500, 'solver': 'adam'}	0.833614
220	{'hidden_layer_sizes': (30, 30), 'learning_rate_init': 0.005, 'max_iter': 500, 'solver': 'adam'}	0.832365
217	{'hidden_layer_sizes': (30, 30), 'learning_rate_init': 0.005, 'max_iter': 250, 'solver': 'adam'}	0.831912
382	{'hidden_layer_sizes': (50, 50), 'learning_rate_init': 0.001, 'max_iter': 500, 'solver': 'adam'}	0.829522

The best parameters are: {'hidden_layer_sizes': (7,), 'learning_rate_init': 0.001, 'max_iter': 500, 'solver': 'lbfgs'}

Number	Parameters	Validation score
27	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.05, 'max_iter': 500, 'solver': 'lbfgs'}	0.796477
21	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.025, 'max_iter': 500, 'solver': 'lbfgs'}	0.796477
33	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.075, 'max_iter': 500, 'solver': 'lbfgs'}	0.796477
3	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.001, 'max_iter': 500, 'solver': 'lbfgs'}	0.796477
39	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.1, 'max_iter': 500, 'solver': 'lbfgs'}	0.796477
15	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.01, 'max_iter': 500, 'solver': 'lbfgs'}	0.796477
9	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.005, 'max_iter': 500, 'solver': 'lbfgs'}	0.796477
52	{'hidden_layer_sizes': (7, 7), 'learning_rate_init': 0.005, 'max_iter': 500, 'solver': 'adam'}	0.794033
130	{'hidden_layer_sizes': (14, 14), 'learning_rate_init': 0.001, 'max_iter': 500, 'solver': 'adam'}	0.793384
153	{'hidden_layer_sizes': (14, 14), 'learning_rate_init': 0.05, 'max_iter': 500, 'solver': 'lbfgs'}	0.793221
135	{'hidden_layer_sizes': (14, 14), 'learning_rate_init': 0.005, 'max_iter': 500, 'solver': 'lbfgs'}	0.793221
141	{'hidden_layer_sizes': (14, 14), 'learning_rate_init': 0.01, 'max_iter': 500, 'solver': 'lbfgs'}	0.793221
147	{'hidden_layer_sizes': (14, 14), 'learning_rate_init': 0.025, 'max_iter': 500, 'solver': 'lbfgs'}	0.793221

305	{'hidden_layer_sizes': (40, 40), 'learning_rate_init': 0.005, 'max_iter': 500, 'solver': 'sgd'}	0.730973
389	{'hidden_layer_sizes': (50, 50), 'learning_rate_init': 0.005, 'max_iter': 500, 'solver': 'sgd'}	0.729022
281	{'hidden_layer_sizes': (40,), 'learning_rate_init': 0.05, 'max_iter': 500, 'solver': 'sgd'}	0.727721
278	{'hidden_layer_sizes': (40,), 'learning_rate_init': 0.05, 'max_iter': 250, 'solver': 'sgd'}	0.727721
272	{'hidden_layer_sizes': (40,), 'learning_rate_init': 0.025, 'max_iter': 250, 'solver': 'sgd'}	0.727555
311	{'hidden_layer_sizes': (40, 40), 'learning_rate_init': 0.01, 'max_iter': 500, 'solver': 'sgd'}	0.727063
230	{'hidden_layer_sizes': (30, 30), 'learning_rate_init': 0.025, 'max_iter': 250, 'solver': 'sgd'}	0.727063
233	{'hidden_layer_sizes': (30, 30), 'learning_rate_init': 0.025, 'max_iter': 500, 'solver': 'sgd'}	0.727063
29	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.05, 'max_iter': 500, 'solver': 'sgd'}	0.726253
308	{'hidden_layer_sizes': (40, 40), 'learning_rate_init': 0.01, 'max_iter': 250, 'solver': 'sgd'}	0.726248
140	{'hidden_layer_sizes': (14, 14), 'learning_rate_init': 0.01, 'max_iter': 250, 'solver': 'sgd'}	0.726088
191	{'hidden_layer_sizes': (30,), 'learning_rate_init': 0.025, 'max_iter': 500, 'solver': 'sgd'}	0.726087
146	{'hidden_layer_sizes': (14, 14), 'learning_rate_init': 0.025, 'max_iter': 250, 'solver': 'sgd'}	0.726085
149	{'hidden_layer_sizes': (14, 14), 'learning_rate_init': 0.025, 'max_iter': 500, 'solver': 'sgd'}	0.726085
23	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.025, 'max_iter': 500, 'solver': 'sgd'}	0.726083
245	{'hidden_layer_sizes': (30, 30), 'learning_rate_init': 0.075, 'max_iter': 500, 'solver': 'sgd'}	0.725926
20	{'hidden_layer_sizes': (7,), 'learning_rate_init': 0.025, 'max_iter': 250, 'solver': 'sgd'}	0.725598
275	{'hidden_layer_sizes': (40,), 'learning_rate_init': 0.025, 'max_iter': 500, 'solver': 'sgd'}	0.725436

תוצאות הפרטיונינג שני:

The best parameters are: {'hidden_layer_sizes': (10,), 'max_iter': 500, 'solver': 'lbfgs'}

Number	Parameters	Validation score
8	{'hidden_layer_sizes': (10,), 'max_iter': 500, 'solver': 'lbfgs'}	0.802184
10	{'hidden_layer_sizes': (10, 10), 'max_iter': 500, 'solver': 'lbfgs'}	0.799743
4	{'hidden_layer_sizes': (7,), 'max_iter': 500, 'solver': 'lbfgs'}	0.796477
15	{'hidden_layer_sizes': (14, 14), 'max_iter': 500, 'solver': 'adam'}	0.793384
0	{'hidden_layer_sizes': (5,), 'max_iter': 500, 'solver': 'lbfgs'}	0.793223
14	{'hidden_layer_sizes': (14, 14), 'max_iter': 500, 'solver': 'lbfgs'}	0.793221
31	{'hidden_layer_sizes': (56, 56), 'max_iter': 500, 'solver': 'adam'}	0.784905
27	{'hidden_layer_sizes': (46, 46), 'max_iter': 500, 'solver': 'adam'}	0.783771
3	{'hidden_layer_sizes': (5, 5), 'max_iter': 500, 'solver': 'adam'}	0.783283
23	{'hidden_layer_sizes': (36, 36), 'max_iter': 500, 'solver': 'adam'}	0.78312
2	{'hidden_layer_sizes': (5, 5), 'max_iter': 500, 'solver': 'lbfgs'}	0.78117
19	{'hidden_layer_sizes': (26, 26), 'max_iter': 500, 'solver': 'adam'}	0.780512
7	{'hidden_layer_sizes': (7, 7), 'max_iter': 500, 'solver': 'adam'}	0.778715
11	{'hidden_layer_sizes': (10, 10), 'max_iter': 500, 'solver': 'adam'}	0.773175
18	{'hidden_layer_sizes': (26, 26), 'max_iter': 500, 'solver': 'lbfgs'}	0.773009
21	{'hidden_layer_sizes': (36,), 'max_iter': 500, 'solver': 'adam'}	0.766659
29	{'hidden_layer_sizes': (56,), 'max_iter': 500, 'solver': 'adam'}	0.76536

Number	Parameters	Train score
31	{'hidden_layer_sizes': (56, 56), 'max_iter': 500, 'solver': 'adam'}	0.831894
30	{'hidden_layer_sizes': (56, 56), 'max_iter': 500, 'solver': 'lbfgs'}	0.828943
23	{'hidden_layer_sizes': (36, 36), 'max_iter': 500, 'solver': 'adam'}	0.827132
27	{'hidden_layer_sizes': (46, 46), 'max_iter': 500, 'solver': 'adam'}	0.825177
22	{'hidden_layer_sizes': (36, 36), 'max_iter': 500, 'solver': 'lbfgs'}	0.817953
14	{'hidden_layer_sizes': (14, 14), 'max_iter': 500, 'solver': 'lbfgs'}	0.816794
26	{'hidden_layer_sizes': (46, 46), 'max_iter': 500, 'solver': 'lbfgs'}	0.815093
15	{'hidden_layer_sizes': (14, 14), 'max_iter': 500, 'solver': 'adam'}	0.812684
19	{'hidden_layer_sizes': (26, 26), 'max_iter': 500, 'solver': 'adam'}	0.812648
18	{'hidden_layer_sizes': (26, 26), 'max_iter': 500, 'solver': 'lbfgs'}	0.811309
10	{'hidden_layer_sizes': (10, 10), 'max_iter': 500, 'solver': 'lbfgs'}	0.809407
8	{'hidden_layer_sizes': (10,), 'max_iter': 500, 'solver': 'lbfgs'}	0.807905
4	{'hidden_layer_sizes': (7,), 'max_iter': 500, 'solver': 'lbfgs'}	0.802111
20	{'hidden_layer_sizes': (36,), 'max_iter': 500, 'solver': 'lbfgs'}	0.800934
28	{'hidden_layer_sizes': (56,), 'max_iter': 500, 'solver': 'lbfgs'}	0.799848
3	{'hidden_layer_sizes': (5, 5), 'max_iter': 500, 'solver': 'adam'}	0.79715
0	{'hidden_layer_sizes': (5,), 'max_iter': 500, 'solver': 'lbfgs'}	0.796969
7	{'hidden_layer_sizes': (7, 7), 'max_iter': 500, 'solver': 'adam'}	0.795811

תוצאות היפר טיוניג שלישי:

Number	Parameters	Train score
21	{'hidden_layer_sizes': (11, 9), 'max_iter': 500, 'solver': 'lbfgs'}	0.817174
13	{'hidden_layer_sizes': (14, 14), 'max_iter': 500, 'solver': 'lbfgs'}	0.816794
11	{'hidden_layer_sizes': (12, 12), 'max_iter': 500, 'solver': 'lbfgs'}	0.814404
18	{'hidden_layer_sizes': (10, 12), 'max_iter': 500, 'solver': 'lbfgs'}	0.814241
20	{'hidden_layer_sizes': (9, 11), 'max_iter': 500, 'solver': 'lbfgs'}	0.813897
9	{'hidden_layer_sizes': (11, 11), 'max_iter': 500, 'solver': 'lbfgs'}	0.813861
5	{'hidden_layer_sizes': (9, 9), 'max_iter': 500, 'solver': 'lbfgs'}	0.811834
19	{'hidden_layer_sizes': (12, 10), 'max_iter': 500, 'solver': 'lbfgs'}	0.811127
7	{'hidden_layer_sizes': (10, 10), 'max_iter': 500, 'solver': 'lbfgs'}	0.809407
17	{'hidden_layer_sizes': (10, 8), 'max_iter': 500, 'solver': 'lbfgs'}	0.809317
6	{'hidden_layer_sizes': (10,), 'max_iter': 500, 'solver': 'lbfgs'}	0.807905
15	{'hidden_layer_sizes': (9, 7), 'max_iter': 500, 'solver': 'lbfgs'}	0.806764
8	{'hidden_layer_sizes': (11,), 'max_iter': 500, 'solver': 'lbfgs'}	0.806746
4	{'hidden_layer_sizes': (9,), 'max_iter': 500, 'solver': 'lbfgs'}	0.805189
2	{'hidden_layer_sizes': (8,), 'max_iter': 500, 'solver': 'lbfgs'}	0.803867
0	{'hidden_layer_sizes': (7,), 'max_iter': 500, 'solver': 'lbfgs'}	0.802111
3	{'hidden_layer_sizes': (8, 8), 'max_iter': 500, 'solver': 'lbfgs'}	0.801098
10	{'hidden_layer_sizes': (12,), 'max_iter': 500, 'solver': 'lbfgs'}	0.794
16	{'hidden_layer_sizes': (8, 10), 'max_iter': 500, 'solver': 'lbfgs'}	0.793837

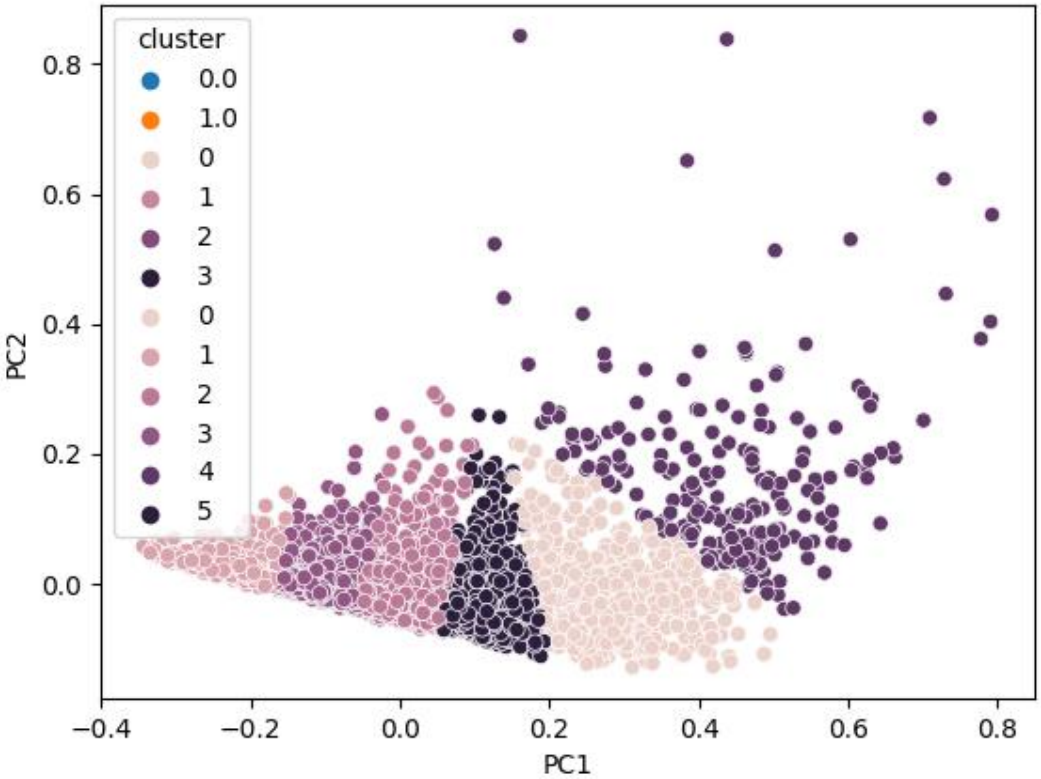
Number	Parameters	Validation score
5	{'hidden_layer_sizes': (9, 9), 'max_iter': 500, 'solver': 'lbfgs'}	0.8043
19	{'hidden_layer_sizes': (12, 10), 'max_iter': 500, 'solver': 'lbfgs'}	0.802345
6	{'hidden_layer_sizes': (10,), 'max_iter': 500, 'solver': 'lbfgs'}	0.802184
21	{'hidden_layer_sizes': (11, 9), 'max_iter': 500, 'solver': 'lbfgs'}	0.801695
20	{'hidden_layer_sizes': (9, 11), 'max_iter': 500, 'solver': 'lbfgs'}	0.801367
11	{'hidden_layer_sizes': (12, 12), 'max_iter': 500, 'solver': 'lbfgs'}	0.800551
7	{'hidden_layer_sizes': (10, 10), 'max_iter': 500, 'solver': 'lbfgs'}	0.799743
18	{'hidden_layer_sizes': (10, 12), 'max_iter': 500, 'solver': 'lbfgs'}	0.799576
4	{'hidden_layer_sizes': (9,), 'max_iter': 500, 'solver': 'lbfgs'}	0.799412
17	{'hidden_layer_sizes': (10, 8), 'max_iter': 500, 'solver': 'lbfgs'}	0.798445
8	{'hidden_layer_sizes': (11,), 'max_iter': 500, 'solver': 'lbfgs'}	0.79811
2	{'hidden_layer_sizes': (8,), 'max_iter': 500, 'solver': 'lbfgs'}	0.796809
0	{'hidden_layer_sizes': (7,), 'max_iter': 500, 'solver': 'lbfgs'}	0.796477
15	{'hidden_layer_sizes': (9, 7), 'max_iter': 500, 'solver': 'lbfgs'}	0.795994
9	{'hidden_layer_sizes': (11, 11), 'max_iter': 500, 'solver': 'lbfgs'}	0.793549
13	{'hidden_layer_sizes': (14, 14), 'max_iter': 500, 'solver': 'lbfgs'}	0.793221
3	{'hidden_layer_sizes': (8, 8), 'max_iter': 500, 'solver': 'lbfgs'}	0.787676
16	{'hidden_layer_sizes': (8, 10), 'max_iter': 500, 'solver': 'lbfgs'}	0.783444

CLUSTERING

מדד דיוויס-בולדוין עבור כל K:

```
For k= 3 , Davies-Bouldin score is: 0.18946695690802862
For k= 4 , Davies-Bouldin score is: 0.17882569619290625
For k= 5 , Davies-Bouldin score is: 0.15703110010914828
For k= 6 , Davies-Bouldin score is: 0.14286902138001809
For k= 7 , Davies-Bouldin score is: 0.1537050585315878
For k= 8 , Davies-Bouldin score is: 0.15477957428306852
For k= 9 , Davies-Bouldin score is: 0.15406012015001755
```

גרף עבור **k**=6 (אופטימאלי)



מדד שילואט עבור כל K

```
For k= 3 , Silhouette score is: 0.9032077164263895
For k= 4 , Silhouette score is: 0.9060150263302567
For k= 5 , Silhouette score is: 0.9218737132476607
For k= 6 , Silhouette score is: 0.9346615222852834
For k= 7 , Silhouette score is: 0.9339547145259732
For k= 8 , Silhouette score is: 0.938208524051991
For k= 9 , Silhouette score is: 0.9402214483401692
```

IMPROVEMENTS

טבלת BOOSTING

	learning_rate	n_estimators	Train Accuracy	Validation Accuracy
0	0.025	40	0.977839	0.9402
1	0.05	40	0.987489	0.944271
2	0.075	40	0.993881	0.945575
3	0.1	40	0.998008	0.947204
4	0.25	40	1	0.949158
5	0.5	40	1	0.947529
6	0.75	40	1	0.944596
7	1	40	0.999982	0.941174
8	0.025	50	0.980338	0.941177
9	0.05	50	0.990441	0.945249
10	0.075	50	0.997248	0.946554
11	0.1	50	0.999439	0.947041
12	0.25	50	1	0.951113
13	0.5	50	1	0.948996
14	0.75	50	1	0.944434
15	1	50	0.999982	0.941174
16	0.025	60	0.983108	0.942643
17	0.05	60	0.993555	0.946552
18	0.075	60	0.998841	0.945738
19	0.1	60	0.999946	0.946389
20	0.25	60	1	0.950949
21	0.5	60	1	0.94981
22	0.75	60	1	0.944434
23	1	60	0.999982	0.941174
24	0.025	70	0.985425	0.943946
25	0.05	70	0.996198	0.947204
26	0.075	70	0.999747	0.946553
27	0.1	70	1	0.948018
28	0.25	70	1	0.950298
29	0.5	70	1	0.94981
30	0.75	70	1	0.944434
31	1	70	0.999982	0.941174
32	0.025	80	0.987453	0.945412
33	0.05	80	0.997646	0.947367
34	0.075	80	0.999964	0.947204
35	0.1	80	1	0.949322
36	0.25	80	1	0.950787
37	0.5	80	1	0.94981
38	0.75	80	1	0.944434

39	1	80	0.999982	0.941174
40	0.025	90	0.989028	0.945739
41	0.05	90	0.998642	0.945574
42	0.075	90	1	0.94704
43	0.1	90	1	0.948996
44	0.25	90	1	0.95095
45	0.5	90	1	0.94981
46	0.75	90	1	0.944434
47	1	90	0.999982	0.941174
48	0.025	100	0.990296	0.944761
49	0.05	100	0.999294	0.945249
50	0.075	100	1	0.947692
51	0.1	100	1	0.949323
52	0.25	100	1	0.952254
53	0.5	100	1	0.94981
54	0.75	100	1	0.944434
55	1	100	0.999982	0.941174