למידת מכונה – פרויקט גמר

סיווג תמונות של בגדים לשתי מחלקות (ילדים / מבוגרים) באמצעות מספר שיטות של למידת מכונה

מגישים –

1. עדן שקורי 208449256
2. יוסף שוורץ 313419483

מרצה – ד"ר לי-עד גוטליב,

המחלקה למדעי המחשב,

אוניברסיטת אריאל שבשומרון.

מבוא:

במסגרת פרויקט הגמר בקורס "למידת מכונה", מצאנו מאגר נתונים המכיל תמונות של פרטי לבוש מגוונים השייכים למבוגרים וילדים.

הרצנו מספר אלגוריתמים ללמידת מוכנה (KNN, SVM, Logistic regression, MLP, CNN),

במטרה לסווג את התמונות לשתי מחלקות (ילדים / מבוגרים).

בנוסף, השתמשנו בטכניקת PCA על מנת להוריד את ממד התמונות.

נרחיב על אלגוריתמים אלו בהמשך, כמו כן נבצע אנאליזות שונות ונפרט על בעיות בהן נתקלנו במהלך הפרויקט, הפתרונות שלהם והמסקנות שנבעו מבניית המודלים השונים.

מאגר הנתונים:

https://www.kaggle.com/agrigorev/clothing-dataset-full

מאגר הנתונים שלנו מגיע מאתר Kaggle, בקישור שמעל.

הוא מכיל כ-5762 תמונות צבעוניות בגדלים שונים, וקובץ csv המכיל מידע עבור התמונות (סיווג למחלקת פריטי לבוש – לא נעסוק בכך בפרויקט, סיווג למבוגרים/ילדים, ועוד).

לדוגמא:



עיבוד מקדים:

כחלק מעיבוד המקדים החלטנו על מנת להקל על תהליך הלמידה בחלק מהמודלים, ועל מנת לשמור על אחידות הפרוייקט, החלטנו לבצע מס' פעולות:

1. מעבר על הנתונים ובדיקת שלמותם – עברנו על כלל התמונות והקבצים המצורפים על מנת לאתר תקלות שיכולות להקשות בהמשך, גילינו פערים בין מה שמופיע בקובץ למה שקיים במאגר, ולכן מחקנו תמונות אשר אין לגביהן מידע או תיוג.
   נשארנו עם כ-5403 תמונות.
2. הגדרת גודל אחיד – על מנת להקל על תהליך הלמידה, ועל מנת לאפשר מודל אחיד (לדוגמא בCNN) הגדרנו גודל אחיד לכלל התמונות – 32X32X3.
   זאת על מנת שנוכל להריץ את המודלים במחשבים אישיים (ללא GPU) ולקבל תוצאות בזמן סביר.
3. הכנת הפיצ'רים – שמרנו את מאגר התמונות כמערך 4 ממדי, ולכן היה צורך "לשטח" את המערכים כדי שהמודלים השונים יוכלו להשתמש בפיקסלים (פיצ'רים) בנוחות.
   ממערך של 5403X32X32X3, המרנו למערך דו-ממדי של 5403X3072.
4. נרמול הנתונים – מכיוון שאנו עוסקים בתמונות, כל פיצ'ר הוא בעצם פיקסל שערכו הוא בטווח 0 ועד 255, על מנת להקל על האלגוריתמים השונים נרמלנו את הנתונים לטווח [0,1] ע"י soft-normalization.
5. פיצול לקבוצת אימון וקבוצת בדיקה ע"י sklearn.model_selection.train_test_split, כאשר הדאטה מחולקת ל20% מבחן, ו80% אימון.
   בחלק מהאלגוריתמים ישנה קבוצת בדיקה (validation) שהיא מהווה 25% מקבוצת האימון (כלומר, 20% מכלל המאגר).

אלגוריתם KNN:

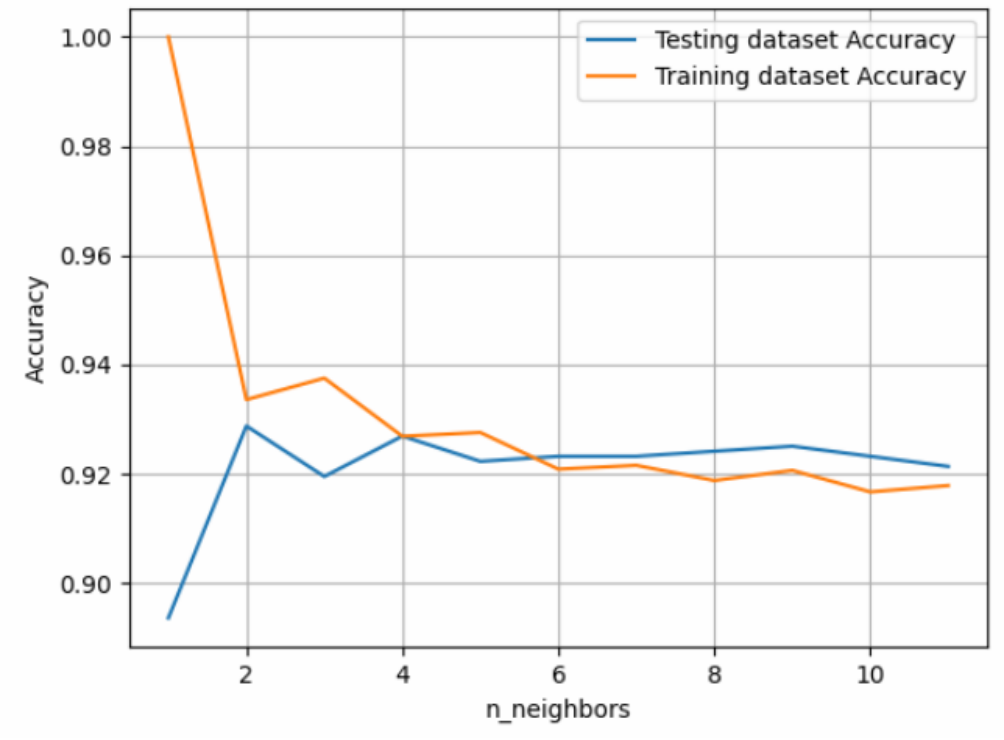
אלגוריתם K-NN הינו טכניקה בלמידת מכונה המסווגת נתון חדש למחלקה שאליה שייכים רוב ה"שכנים" שלו מתוך הK שכנים הקרובים ביותר אליו.

לדוגמא, נניח שנתון חדש, ב9-NN קרוב ל5 אדומים, 3 ירוקים ו1 שחור, הנתון יסווג כאדום.

האלגוריתם דורש זיכרון גדול מכיוון שהוא זוכר את פרטי כלל הנתונים, ולפיהם יודע לסווג דאטה חדש.

נשים לב שקיימים סוגים שונים של מרחקים (מרחק מנהטן, מרחק אוקלידי, ועוד), ניתן להשתמש בכל מרחק כל עוד הממד זהה.

את האלגוריתם הרצנו על k מ1 ועד 10, התוצאות של הAccuracy מוצגות בגרף:



ניתן לראות כי עבור k=1 הAccuracy=100% אך זה מכיוון שזה מעין Overfitting מכוון.

האלגוריתם בעצם מסווג את הדאטה לפי השכן הכי קרוב אליו (שכן אחד) שהוא בעל תיוג, ומכיוון שבאימון כלל התמונות הינן מתויגות, אזי שהוא בוחר בעצם את המחלקה של עצמו ומכאן ה100% הצלחה.

אולם, עבור הבדיקה (test) אחוזי ההצלחה נמוכים יותר (~89%).

מהגרף לעיל ניתן להסיק כי עבור הדאטה שלנו הk המתאים ביותר עבור הבדיקה הינו k=2, עם 93% הצלחה.

אלגוריתם SVM:

אלגוריתם SVM הינו אלגוריתם ללמידת מכונה המחפש אחר מישור מפריד בין כלל הנקודות, בנוסף האלגוריתם מחפש גם את המרווח הגדול ביותר, לעיתים הוא יעדיף "לוותר" על נקודה וזה על מנת לקבל מרווח גדול יותר.

מבחינה ויזואלית נוכל להבין זאת ב2D ולכאורה נחפש קו שיפריד בין הנקודות, אך במידה והנקודות המתויגות כ1 הן במרכז ועוטפות אותן נקודות המתויגות כ0 מכל הכיוונים, האלגוריתם יבצע טכניקה שבה הוא מעלה את ממד הנקודות ל3D על מנת להפריד ביניהם באמצעות מישור.

בתחילה ניסנו להשתמש בטכניקה "linear" וקיבלנו את התוצאה הבאה-

SVM: train accuracy = 100.00%, test accuracy = 86.11%

הבנו שאנו נמצאים בoverfitting ולכן פונקציה זו ייתכן ואינה מתאימה לרב מימד, לכן בחרנו לבדוק גם את פונקציית "poly" והתקבלה התוצאה הבאה-

SVM: train accuracy = 92.82%, test accuracy = 88.89%

ניתן לראות שכעת יש יותר דמיון בין האימון לבדיקה ולכן, בחרנו להיצמד לפונקציה זו.

Logistic Regression:

בטכניקה זו האלגוריתם מקבל את מאגר האימון, ומציב לכל פיצ'ר משקולת, ולכל דגימה גם מאזן (bias), לאלגוריתם ישנה פונקציית טעות (Loss) הבודקת כמה אנו טועים בחיזוי לעומת האמת, בהתאם לתוצאה שלה היא מעדכנת את ערכי המשקולות כדי שבהינתן דאטה חדש נוכל להתייחס לכל פיקסל ביחס מתאים וזה על מנת לחזות כמה שיותר נכון.

האלגוריתם מעדכן את המשקולות בצורה שנקראת Gradient descent והיא מעדכנת את המשקולות בכיוון הפוך לנגזרת לפי קצת למידה α .

ניתן להגדיל או להקטין את קצב הלמידה בהתאם להתכנסות.

הרצנו את האלגוריתם על המאגר והתקבלה התוצאה הבאה –

Final accuracy: 89.36%

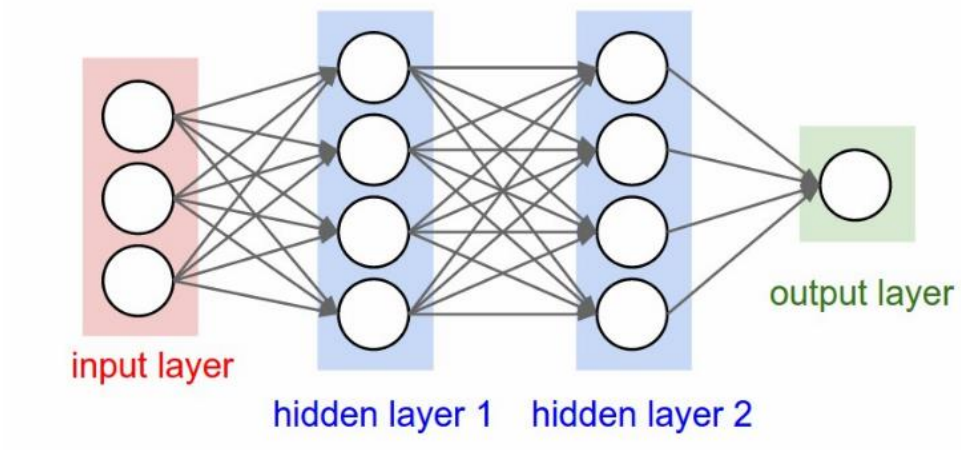
רשת MLP:

אלגוריתם זה עובד בצורה דומה לLogistic Regression שדנו עליו כעת, אך הוא מכיל בתוכו שכבות נסתרות.

השכבה הראשונה הינה בגודל הקלט (כמות הפיצ'רים לכל דוגמא).

השכבה האחרונה הינה בגודל הפלט (במקרה שלנו, סיווג בינארי דורש חוליה אחת).

בין שכבה לשכבה החוליות מחוברות בתצורה שנקראת Fully Connected, כלומר, כל קודקוד בשכבה l, מחובר לכל הקודקודים בשכבה $l+1$.



פרטים טכניים –

64 – Batch size

25% – Validation ממאגר האימון.

RELU – Activation function

Sigmoid – Output layer activation

6 – Epochs

binary-crossentropy – Loss function

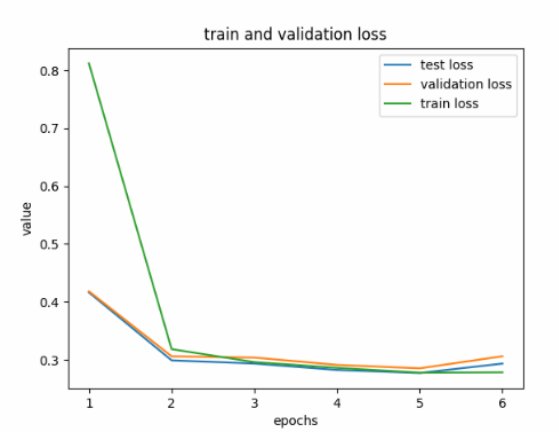
Adam – Optimaizer

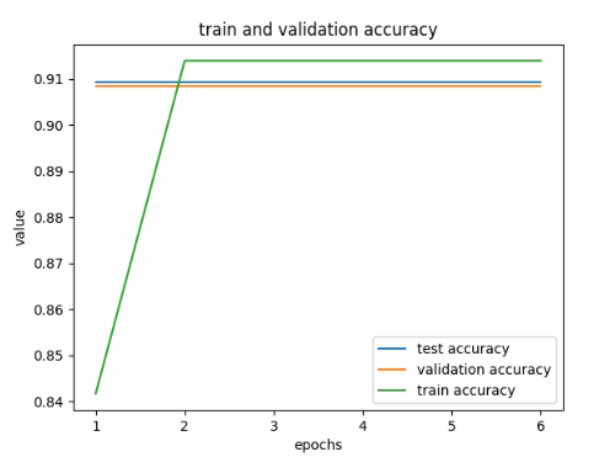
ארכיטקטורת המודל בצורה משפך (כיוון הזרימה הוא משכבה גדולה לקטנה)

ניסיונות-

בסבב נסיונות הראשון יצרנו 3 שכבות פנימיות (1024,512,256) –

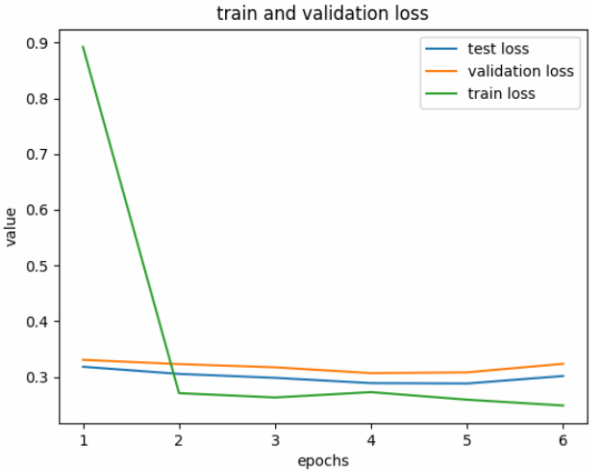
ניתן לראות כי אחוז ההצלחה על מאגר הבדיקה הינו סביב ה30%, לעומת מאגר האימון ששם אחוז ההצלחה הינו סביב ה92%, בהתאם ניתן להסתכל על פונקציית הטעות ולהבין כי הטעות על הבדיקה גבוהה 90%, לעומת הטעות באימונים ששם הטעות יורדת עד 28% לאורך האימון.
כלומר, במודל זה קיים Overfitting ולכן יש צורך לשפר את המודל.

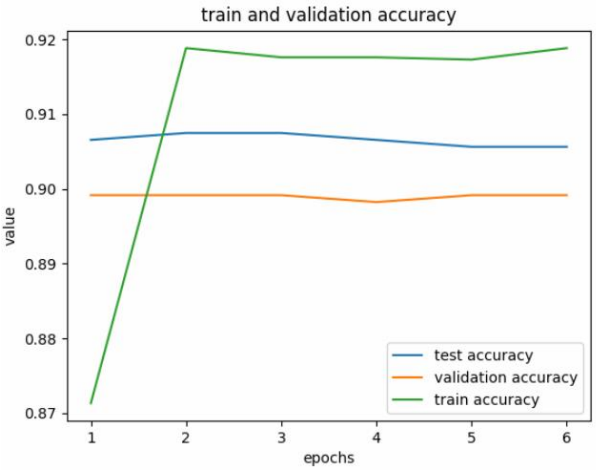




בסבב ניסיונות השני ניסינו לבצע שיטות רגולריזציה כמו DropOut אך באימונים צפינו בהתנהגות לא צפויה ולכן לא המשכנו עם טכניקה זו.

בסבב ניסיונות השלישי בחרנו להוריד שכבות פנימיות ולהישאר רק עם שכבה נסתרת אחת בעלת 512 חוליות-





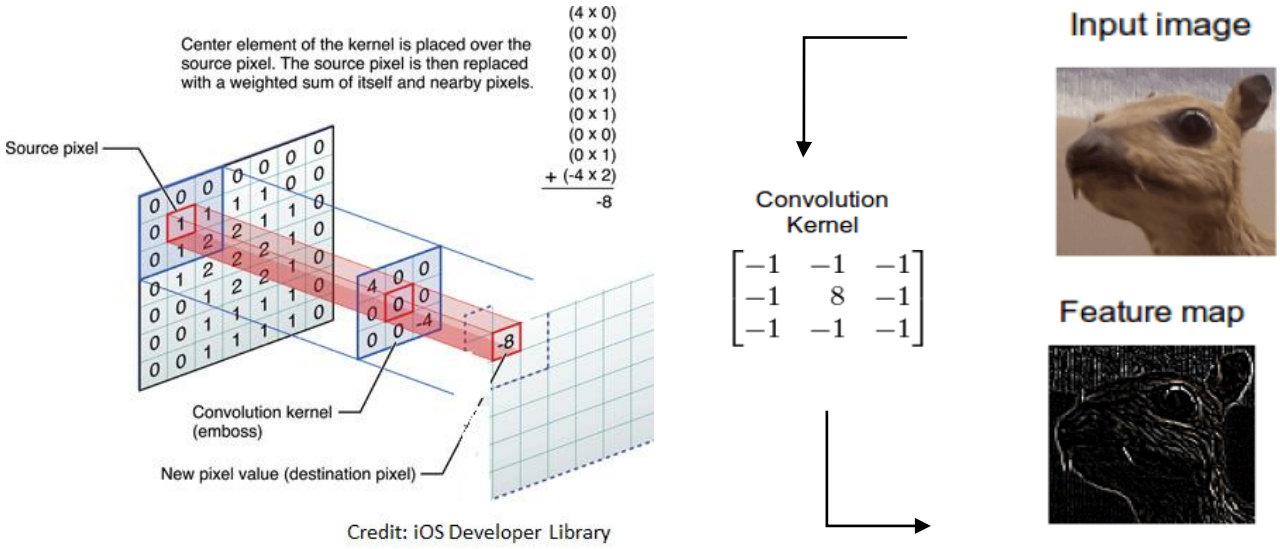
בניסיונות אלו קיבלנו תוצאות טובות ביותר ולכן בחרנו להיצמד למודל זה.

רשת CNN-

CNN הינה רשת נוירונים המתאימה לעבודה על תמונות, מורכבת מקונבולוציות, כלומר עיבוד נתונים של הפיקסלים בתמונה.

שכבת קונבלוציה מורכבת ממטריצת kernel שהיא "סורקת" את המטריצה של התמונה ומבצעת מכפלה של איבר באיבר, ומתקדמת ימינה ולמטה בהתאם לגודל מטריצת הקלט.

שכבת הקונבלוציה נועדה להדגיש תכונות מסיימות בתמונה, לפי פילטרים מסוימים הנקבעים במהלך הלמידה, למשל, הדגשה של צורות מסוימות (קווים, עיגולים וכו').



לאחר שכבת קונבולציה הגדרנו שכבת pooling שזו שכבה שבעצם נועדה הוריד את הממד, בחרנו בmax-pooling, כלומר בוחר את הפיקסל עם הערך הגבוה ביותר במטריצת הpooling.

בחרנו להשתמש גם כאן בפונקציית אקטיבציה RELU, וגדלי הקונבלוציות הם 3X3.

להלן הארכיטקטורה שלנו –

שכבת קלט של (3X32X32)

שכבת קונבולציה של (3X3) עם 128 פילטרים (padding=valid ,stride=1)

פונקציית אקטיבציה RELU

שכבת max-pooling (2X2)

שכבת קונבולציה של (3X3) עם 64 פילטרים (padding=valid ,stride=1)

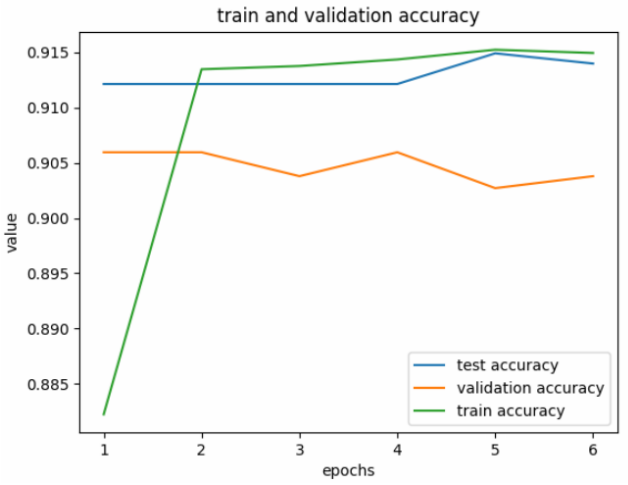
פונקציית אקטיבציה RELU

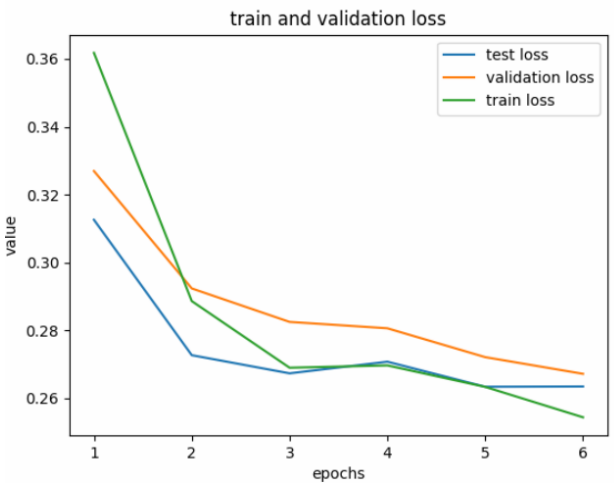
שכבת max-pooling (2X2)

שכבת Fully-Connected ל32 נוירונים

שכבת Output של נוירון אחד, עם פונקציית אקטיבציה "Sigmoid".

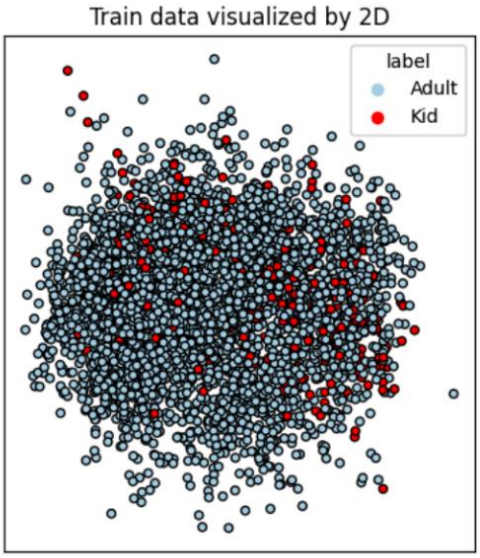
לאחר אימונים עם שימוש בbatch size של 128, התקבלה התוצאה הבאה-

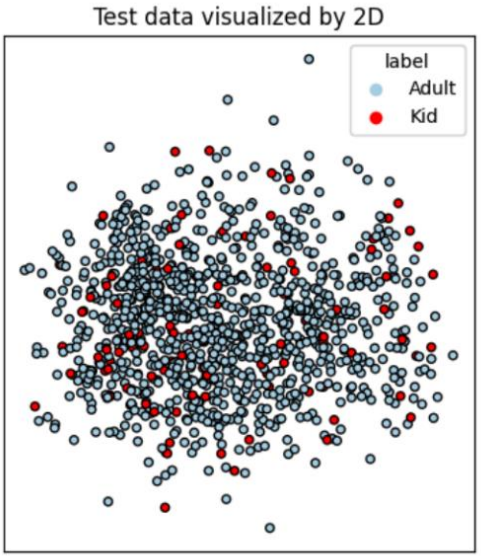




- PCA

PCA הינה שיטה להורדת ממד, השתמשנו בשיטה זו ע"מ להוריד את ממד התמונות מ3072 פיצ'רים (32x32x3) ל2, ע"מ לבחון האם ניתן לסווג את המידע בממד נמוך יותר ובצורה מספקת ובכך לאפשר לאלגוריתמים לעבוד עם ממד נמוך יותר ולחסוך בזמן וזיכרון.

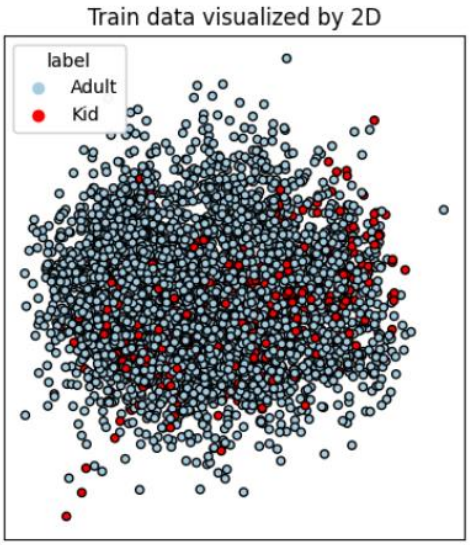


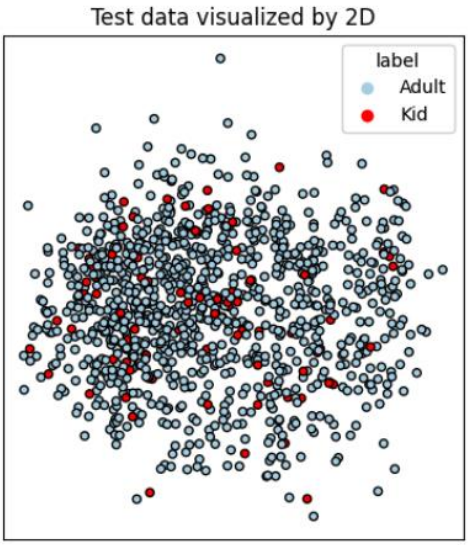


ניתן לראות שלאחר הורדת הממד ניתן להבחין ב2 המחלקות לפי הצבעים, אך ניתן גם לראות ששתי המחלקות מעורבבות, כלומר אין מרחקים בין המחלקות ולכן נצפה לתוצאות פחות טובות.

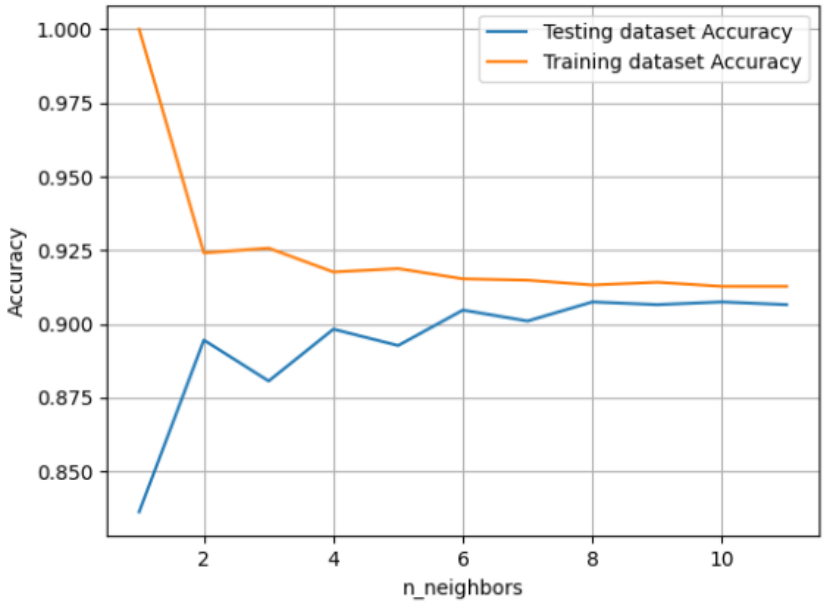
ניסינו לסווג את המידע לאחר הורדת ממד לפי **KNN :**

לאחר הורדת הממד:





לאחר הרצת KNN:



בזכור, התוצאות בהרצת KNN ללא הורדת ממד היו הכי גבוהות כאשר הK=2 עם כ93.5% על הtrain וכ93% על הtest ולעומת זאת כאן כאשר הK=2 התוצאות הן כ92.5% על הtrain וכ89% על הtest, כלומר תוצאות נמוכות יותר כאשר הK זהה.

ניתן לראות שלאחר הורדת הממד גם בתרחיש הכי טוב (עבור הtest) כאשר הK=8 התוצאות הן נמוכות יותר - כ91% על הtrain ועל הtest.

SVM:

לאחר הרצת SVM עם kernel ליניארי קיבלנו את התוצאות הבאות (לאחר הורדת הממד):

SVM: train accuracy = 91.30%, test accuracy = 90.75%

נציין שהרצה עם kernel פולינומי רץ בזמן לא סביר.

וכזכור, התוצאות בSVM ללא הורדת הממד היו -

עם kernel ליניארי:

SVM: train accuracy = 100.00%, test accuracy = 86.11%

ועם kernel פולינומי:

SVM: train accuracy = 92.82%, test accuracy = 88.89%

ניתן לראות שבמקרה זה קיבלנו תוצאות טובות יותר ואף עקביות יותר בין המאגרים (בין הtrain לtest).

מכיוון שבSVM עם kernel ליניארי, המודל אמור למצוא קו ישר שיפריד בין 2 המחלקות, לפי הפיזור שראינו לאחר הPCA נראה שאין קו שיכול להפריד בצורה טובה בין 2 המחלקות מכיוון שהנקודות מעורבבות, ולכן המודל כנראה בוחר להעביר קו "מחוץ" לתחום הנקודות, כלומר ממש בקצה ו"בוחר" לטעות על כל הנקודות האדומות (בגדי הילדים), כלומר לתת תיוג של מבוגר לכל התמונות משום שיש מיעוט של בגדי ילדים בדאטה (ב8% מכלל התמונות), ולכן הדיוק יהיה גבוה אך נספוג הרבה false negative.

(באופן דומה, במטלה 3 שמנו לב לאותה התנהגות חריגה במימוש הAdaboost כאשר האלגוריתם בחר את הקו הראשון)

מסקנות:

סיווג מידע בממד נמוך יותר לרוב תשיג הצלחה פחות טובה (כאשר הממד הוא 2) אך יתכן שכאשר נשתמש בממד קטן אך גדול יותר מ2 נצליח להשיג תוצאות טובות יותר מכיוון שכך נאבד פחות מהפיצ'רים החשובים והסיווג יהיה מדויק יותר.

עם זאת, הורדת ממד מאפשרת לנו להשתמש בפחות זיכרון וזמן ריצה קצר יותר.

לאחר התיקון

עוד לפני ההצגה הבחנו כי הדאטה שלנו לא מאוזן, אך לאחר ההצגה הבנו שכל התוצאות שלנו לא היו מדויקות עקב חוסר האיזון – כ8% מהתמונות הן תמונות של בגדי ילדים וכל השאר הן תמונות של בגדי מבוגרים, ולכן כמעט בכל השיטות \ האלגוריתמים קיבלנו accuracy של כ-90%, כלומר ניתן להבחין כי כנראה בכל המקרים כל המאגר מקבל "אוטומטית" תיוג של 0 (בגד מבוגרים) וטועה אוטומטית על כל בגדי הילדים ע"מ לקבל הצלחה יותר טובה.

מה שנבצע בשלב זה ע"מ להתגבר על חוסר האיזון הוא "להשמיט" חלק מהתמונות של בגדי המבוגרים כדי שיהיה לנו יחס שווה בין המחלקות.

במקור – כמות בגדי הילדים הייתה 476 מתוך 5403 סה"כ.

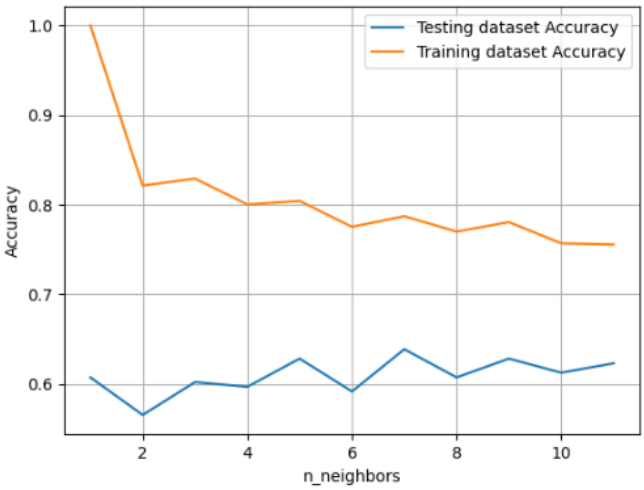
כעת אחרי השמטת חלק מבגדי המבוגרים (לאחר ערבוב הדאטה) יש לנו 476 בגדי ילדים ו476 בגדי מבוגרים, כלומר גודל המאגר הנוכחי הוא 952 תמונות.

נבחן שוב את התוצאות על אותם המודלים לאחר שינוי המאגר:

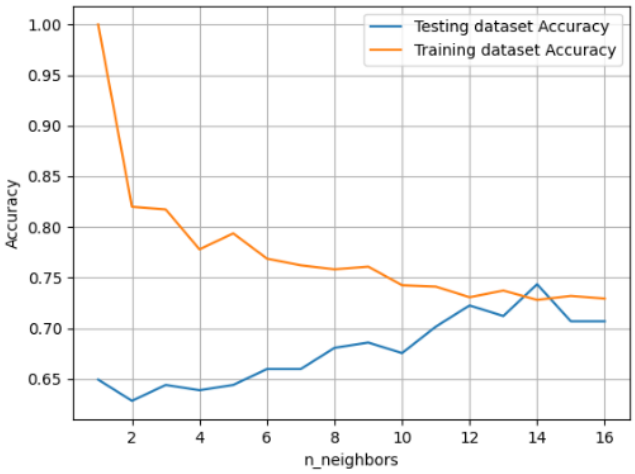
KNN –

באופן דומה ללפני התיקון גם כאן הרצנו את המודל על K מ1 ועד 10, אך כאן ניתן לראות שהמודל עוד לא "מתכנס" ולכן החלטנו לתת לו לרוץ עד K=17.

ניסיון ראשון



ניסיון שני



ניתן לראות שבניסיון השני הK שאיתו יש תוצאות טובות ביותר הוא K=14 , כלומר ניתן תיוג לפי 14 השכנים הכי קרובים (מרחק אוקלידי). התוצאות שקיבלנו בK=14 הם:

train accuracy = ~73.5% , test accuracy = 75%

לעומת המצב לפני התיקון של train accuracy = 93% , test accuracy = ~89%.

נציין שלאחר בדיקה – **כאשר הK הוא זוגי ויש תיקו בין מספר השכנים שמתויגים 0 לבין מספר השכנים שמתויגים 1 האלגוריתם נותן לאותו קודקוד סיווג לפי הסדר של המאגר, כלומר אם ה k/2 הראשונים היו עם תיוג 0 הוא ייתן תיוג 0.

– SVM

עם הרצת kernel ליניארי: SVM: train accuracy = 99.87%, test accuracy = 62.30%

עם הרצת kernel פולינומי: SVM: train accuracy = 50.85%, test accuracy = 46.60%

ניתן לראות שעם kernel ליניארי יש לנו overfitting מאוד משמעותי, אך עם kernel פולינומי התוצאות עקביות בין הtrain לtest.

נזכיר שלפני התיקון התוצאות שלנו היו:

עם הרצת kernel ליניארי: SVM: train accuracy = 100.00%, test accuracy = 86.11%

עם הרצת kernel פולינומי: SVM: train accuracy = 92.82%, test accuracy = 88.89%

ניתן לראות שההתנהגות של הoverfitting בליניארי והעקביות בפולינומי זהה גם לפני התיקון.

- Logistic Regression

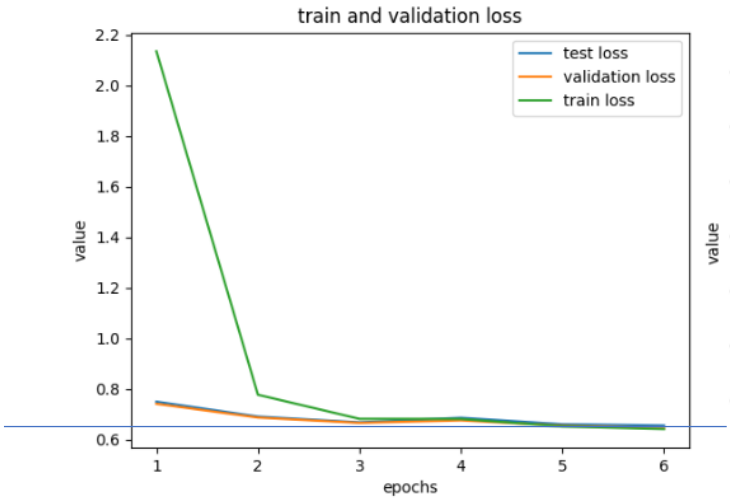
הרצנו את האלגוריתם על המאגר המאוזן והתקבלה התוצאה הבאה – Final accuracy: 59.69%

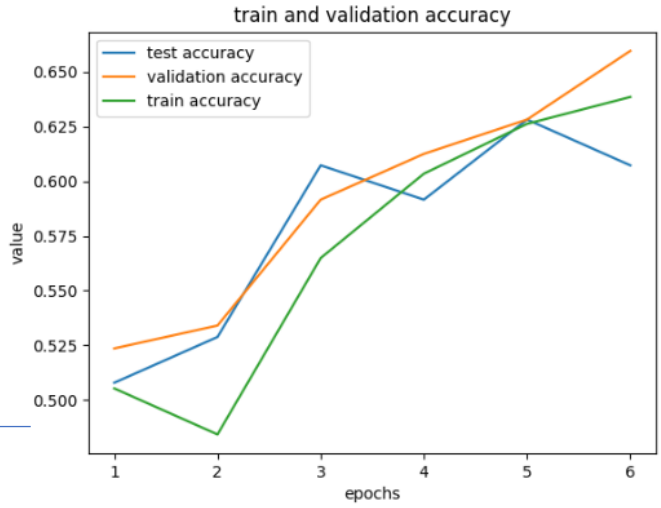
לעומת התוצאה שהתקבלה במאגר הלא מאוזן -Final accuracy: 89.36%.

לאור ההסבר שלנו בחלק הראשון לפני התיקון לא נרחיב שוב על האלגוריתם.

– MLP

הרצנו את אותו המודל עם אותו מבנה על המאגר המאוזן, התוצאות:



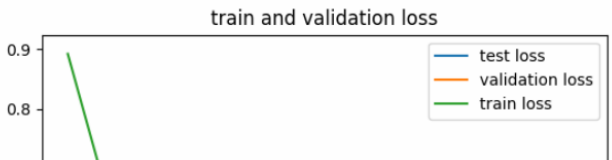


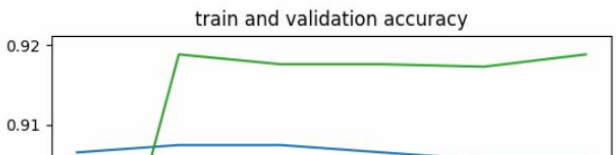
ניתן לראות שאלו תוצאות הרבה יותר הגיוניות כבר מהמבנה של הגרפים באיורים, התוצאות שקיבלנו הם (לאחר 5 epochs):

Train accuracy = Test accuracy = 63%

Train Loss = Test Loss = 0.61

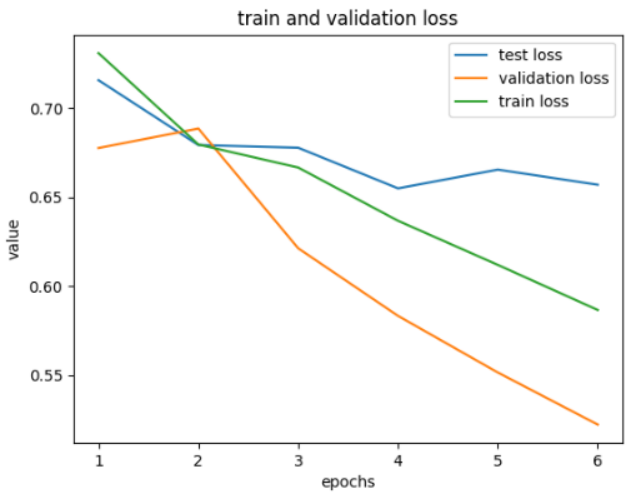
נזכיר כאן את התוצאות שהיו כאשר המאגר היה לא מאוזן לשם הנוחות:

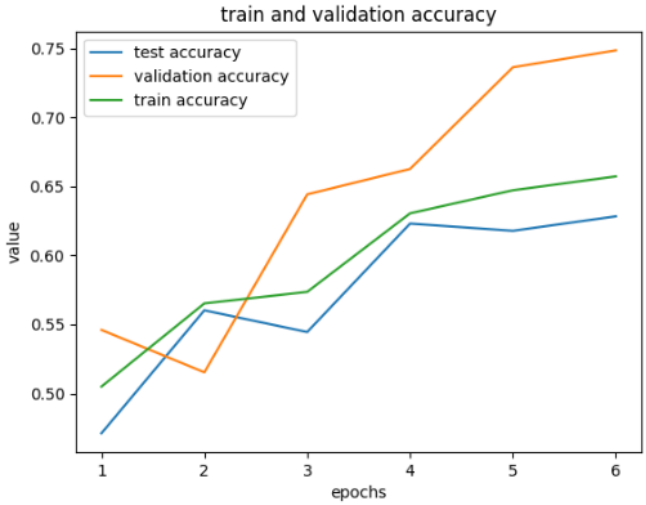




– CNN

גם כאן הרצנו את אותו המודל מלפני ללא שינוי:





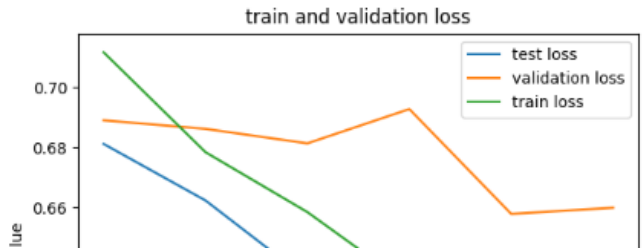
ניתן לראות שקיבלנו תוצאות דיי דומות למודל הMLP –

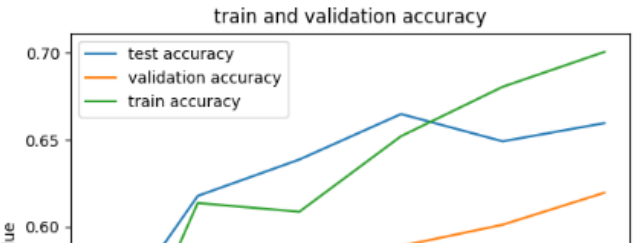
Train accuracy = 65.5%, Test accuracy = 63%

Train Loss =0.65, Test Loss = 0.59

התוצאות נראו לנו קצת חריגות משום שבCNN יש לנו שכבות קונבולוציה שאמורות לעזור לנו יותר ולכן אנו אמורים להצליח לקבל תוצאות טובות יותר בCNN מאשר בMLP.

ולכן החלטנו להוסיף עוד שכבת קונבולוציה של 256 פילטרים ואחריה שכבת max pooling לפני השכבות הקיימות והגענו לתוצאות הבאות:



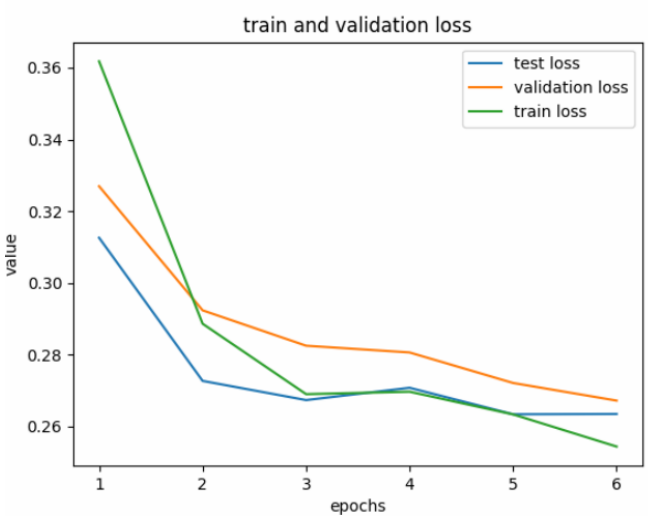


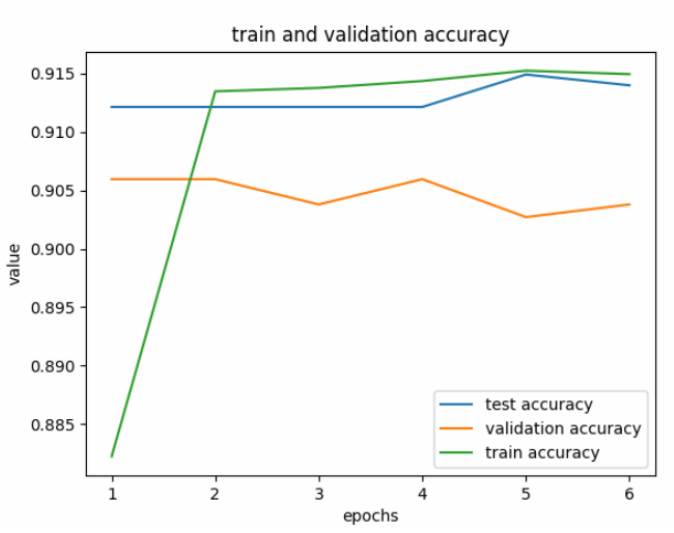
וכפי שחשבנו הצלחנו לשפר מעט את התוצאות ולקבל:

Train accuracy = ~70%, Test accuracy = ~66%

Train Loss = Test Loss = 0.57

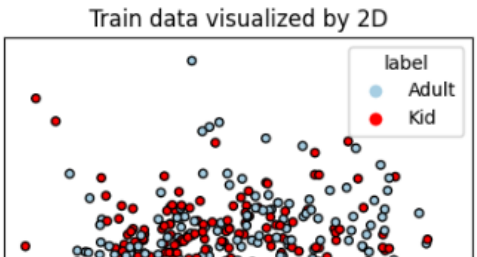
נציין גם את התוצאות שהיו לנו כאשר המודל לא היה מאוזן:

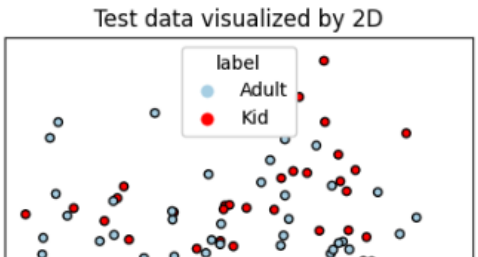




- PCA

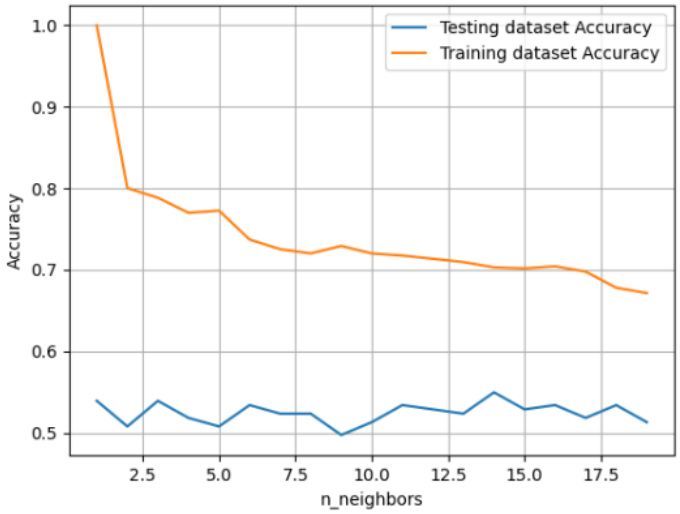
שוב בחנו את שיטת הורדת הממד ל2 -





ניתן להבחין שגם כאן נראה שאין דרך טובה לחלק את המאגר ל2 קבוצות משום שהנקודות מעורבבות.

KNN לאחר PCA:



ניתן לראות שכמעט בכל K אנו מקבלים תוצאות דומות ולכן גם בk=3 ניתן להסתכל על התוצאות כאשר אנו מקבלים accuracy של כ- 54% (לעומת המצב ללא PCA בו קיבלנו accuracy של כ- 75% ולכן ניסן להסיק שבמקרה זה הורדת הממד ל2 לא כ"כ תורמת).

SVM לאחר PCA:

עם kernel ליניארי: SVM: train accuracy = 62.94%, test accuracy = 43.98%

עם kernel פולינומי: SVM: train accuracy = 61.63%, test accuracy = 50.26%

כלומר ניתן לראות שגם לאחר הורדת הממד עדיין יש לנו overfitting עם kernel ליניארי ואף גם עם kernel פולינומי.

נשים לב כי התוצאות שהיו לנו ללא הורדת הממד היו

עם הרצת kernel ליניארי: SVM: train accuracy = 99.87%, test accuracy = 62.30%

עם הרצת kernel פולינומי: SVM: train accuracy = 50.85%, test accuracy = 46.60%

ולכן המקרה של שימוש בkernel פולינומי הורדת הממד מעט תרמה לנו.

מסקנות כלליות:

אנו יכולים להסיק מכלל העבודה על הפרויקט שנדרש תמיד לבדוק דבר ראשון ה אם המאגר שלנו מאוזן או לא ובמידה ולא לנקוט בפעולות מתאימות ע"מ לאזן אותו כמו למשל מה שביצענו- השמטה של חלק מהמאגר עם התיוג של הקבוצה הגדולה יותר.

כמובן ש"תיקון המאגר" כנראה יוריד לנו את הaccuracy אך התוצאות שלנו יהיו "נכונות יותר" והמודלים יתנו תיוג יורת מכוון לאחר למידה ולא יתנו לכמעט כל המאגר את התיוג של הקבוצה הגדולה.